Yazılım Nedir?
En yalın tanımıyla ***yazılım;*** elektronik bir donanımı, belirli bir işi yapması için derlenmiş komutların bütünüdür. Bu komutlar işlemcilerde işlenerek bir olaya dönüştürülür. Peki **Türk Dil Kurumu Sözlüğü**'ne göre **yazılım nedir?** diye baktığımızda şu tanımla karşılaşıyoruz; *Bir bilgisayarda donanıma hayat veren ve bilgi işlemde kullanılan programlar, yordamlar, programlama dilleri ve belgelemelerin tümü.* (TDK Sözlük)
Yazılım aslında hayatımızın her alanındadır. Bu haliyle yazılım aslında hayatımızı kolaylaştırır. Bilgisayarlar, telefonlar, televizyonlar, mobil teknoloji, internet, sanayide kullanılan yeni nesil cihazların neredeyse tamamı, otomotiv, inşaat, eğitim, reklam, pazarlama, iletişim, medya, eğlence, sağlık başta olmak üzere hemen hemen tüm sektörlerde, uzay sanayisinde, günlük hayatta kullanılan bazı teknik aksesuarlarda kısacası "yazılım" gerçekten de yaşamın her alanındadır.

Yazılıma Başlarken Nelere Dikkat Edilmelidir?
Bir fikrinizi hayata geçirmek yada size verilen bir yazılım işini yapabilmek için ilk önce *donanımı* ve/veya *işletim sistemini* seçmeniz gerekir.
Eğer elektronik bir donanım yapılıyorsa ihtiyaca göre en uygun performanslı ve en uygun fiyatlı işlemci ve donanımlar seçilmelidir. İşlemciler günümüzde 5 TL'den başlayıp binlerce liraya kadar çıkabilmektedir. Bu yüzden doğru işlemci seçimi çok önemlidir. Ardından bu işlemcinin desteklediği *dil* ve *dile uygun derleyici* belirlenmelidir. Her işlemcinin her dile ait desteği olmadığı için, desteklediği diller arasındaki seçim bu dillerin sağladığı hız ve kolaylığa göre olmalıdır.
Eğer bilgisayar için bir yazılım yapılıyorsa öncelikle hangi işletim sistemi için yazılım yapılacağı seçilir. Ardından hangi programlama dilinin kullanılacağı belirlenir. Bunun akabinde derleyici yardımı ile yazılan kodlar makine diline çevrilir. *Yazılan dile uyumlu bir derleyici kullanılması bu yüzden zorunludur.* Bilgisayarda dil ve derleyici uyumu elektronik cihazlara göre daha çeşitli ve kolay erişilebilir olduğu için kısa bir araştırma ile ihtiyaçlar kolaylıkla bulunabilir. Burada önemli olan programı hangi işletim sistemi için derleyeceğinizdir.

Yazılım Çeşitleri Nelerdir?
Bu başlıkta yazılımı iki ana başlık altında inceleyeceğiz:

- Bilgisayar Yazılımları
- Elektronik Yazılımları

Bilgisayar, temelde elektronik bir cihaz olsa da içindeki yazılım mantığı temel elektronik cihazlardan biraz daha farklı olduğu için bunları ayrı iki kategoriye ayırmak daha doğru olur. Bu arada mobil cihazlar bilgisayar ile aynı kategoride anlatılabilir, aralarında pek fark yoktur.

Bilgisayar Yazılımları
Bilgisayar yazılımları da kendi içerisinde işlev olarak üçe ayrılır. Bunlar;

- Uygulama Yazılımları
- Sistem Yazılımları
- Bilgisayar Programlama Araçları

Uygulama Yazılımları
Bilgisayarda kullanılan, bir görevi yapmak için yazılmış yazılımlardır. Mesela; *Web programları, Ofis Programları, Resim ve Video Düzenleme Programları, Oyunlar* gibi birçok kategoride uygulamalar bulunmaktadır. Kısacası uygulama yazılımları; insanların çalışmalarını hızlandırmak, bir işlemi bir veya birkaç tuşla yapabilmek için yazılmış yazılımlardır.

Sistem Yazılımları
Herkesin bildiği gibi *Windows, Android, iOS* gibi kullanıcının ilk karşılaştığı, donanımların ve yazılımların uyumlu çalışmasını sağlayan temel yazılımlardır. Sistem yazılımları için uygulama yazılımlarından daha derin bir bilgisayar ve yazılım bilgisi gerekmektedir.

Bilgisayar Programlama Araçları
Bu yazılımlar, yazılan kodları bilgisayar diline çevirerek donanımlara ne yapması gerektiğini söyler. Bu sayede de bilgisayar,/makine bu uygulamaları çalıştırabilir. Eğer yazılım diliyle, kullanılan programlama aracı uyuşmuyorsa

veya doğru işletim sistemine ait değilse o program o cihazda çalışmayacaktır. Bunu Türkçe bilmeyen birine Türkçe bir şeyler anlatmak gibi düşünebilirsiniz.

Elektronik Yazılımları
Bir veya birkaç görevi yapması için yazılan, genellikle işlemcinin pin giriş-çıkışlarına bağlı sensörlerden veri okumak ve işlemek, giriş-çıkışlara bağlı olan motor veya led gibi elektronik cihaza bir iş yaptırmak amacıyla yazılan yazılımlardır. Bu yazılımlar küçük projelerden, sanayide kullanılan büyük cihazlara kadar her alanda kullanılmaktadır.

Bu arada elektronik yazılım dilleri ile bilgisayar dilleri aslında fark yoktur. Yani birçok ortak yazılım dili bulunmaktadır.

Elektronik yazılımının bilgisayar yazılımından farkı; elektronik yazılımlarda, *programlanan işlemciyi mutlaka bir elektronik devreyle; gerekiyorsa da mekanik tasarım ile birleştirilip kullanmak gerekmektedir.* Yani elektronik yazılımlarda; kimi zaman kamera, kimi zaman ise motor gibi fiziksel bir karşılık mutlaka bulunmalıdır.
Programlama Yaparken Hangi Yazılım Dili Kullanılmalıdır?
Aslında bu soru çok fazla sorulmasına rağmen çok genel bir soru olduğu için tek bir karşılığı yani yanıtı yoktur. O nedenle bu soruyu parçalara bölerek cevaplamak gerektiğini düşünüyorum.

Kullanıcı Arabirimine Sahip Uygulamalar İçin;
Kullanıcı arabirimine sahip uygulama yazılımı yapmak isteniyorsa arayüzü sürükle bırak mantığı ile daha kolay ve hızlı yapılabildiği için **C#, Visual Basic** veya **Java** tercih edilebilir. Oyun yazmaya yeni başlayan yazılımcılar genellikle hazır motor kullandığı için **C#** dilini öğrenmeleri kesinlikle gereklidir. Dillerin üçünü de denediğim için *en kolay anlaşılabilir dilin* **Visual Basic** olduğunu ve başlangıçta programlama mantığını öğrenmek için ideal olduğunu düşünüyorum. *Java* dili, yazılım işinde profesyonelleşmek, bu konuda bir işte çalışmak için mutlaka geliştirilmesi gereken çok önemli bir dildir.

Arayüz Yerine Hız Gerektiren İşlemler İçin;
Hesaplama, dosya okuma ve yazma gibi arayüz gerektirmeyen, hız gerektiren işlemler için yazılacaksa **C, C++** veya **Python** öğrenmek çok daha idealdir. Aynı zamanda **C** dili elektronik devrelerin neredeyse tamamında kullanılmaktadır. Diğer dillerle yapılan çalışma süresi hız karşılaştırmasına göre C dili, makine diline en yakın dil olduğu için en hızlı çalışan programlama dilidir. Öğrenmek için biraz daha zor bir dil olsa da hız ve kaynak kontrolü açısından mutlaka öğrenilmesi gereken çok önemli bir dildir. **Assembly** dilini bu kıyaslamanın dışında tuttum, çünkü bir bilgisayar programı yazmak için Assembly dili çok fazla zaman ve enerji sarf ettirecektir. Elektronik devrelerde ise daha basit işlemler için Assembly dili tercih edilebilmektedir, fakat karmaşık işlemlerde bilgisayarda olduğu gibi zaman ve enerji kaybı üst seviyede olacaktır. **C++** dilini kullanmadığımdan, yanlış bilgi vermemek adına onun hakkında bilgi yazmadım. Eğer C++ dili hakkında bilginiz varsa yorum yapabilirsiniz. Okumak ve öğrenmek isterim.

Web Sitesi Yapmak İçin;
Web sitesi veya web uygulaması yapılmak isteniyorsa mutlaka **HTML, CSS** ve **JavaScript** bilinmelidir. HTML ve CSS tek başına kullanılamadıkları için biri **HTML** olmak üzere en az ikisi mutlaka bilinmelidir. Bu üç dilin internette ve kitapçılarda oldukça fazla kaynağı ve ücretsiz örnek çalışmaları var. O yüzden günümüzde öğrenilmesi çok kolay olan dillerdir. Aynı zamanda sitenin hayata geçmesi için **PHP** gibi sunucu üstünde çalışan programlama dillerinin de bilinmesi gerekmektedir. **Django, Flask** gibi **Python** tabanlı web çatıları da web sitesi oluştururken kullanılmaktadır.

Algoritma Nedir?

Algoritma, belli bir problemi çözmek veya belirli bir amaca ulaşmak için tasarlanan yoldur. Algoritma bir problemin çözümünün basit, net, sıralı biçimde belirtilmiş halidir. Matematikte ve bilgisayar biliminde bir işi yapmak için tanımlanan, bir başlangıç durumundan başladığında, açıkça belirlenmiş bir son durumunda sonlanan, sonlu işlemler kümesidir.

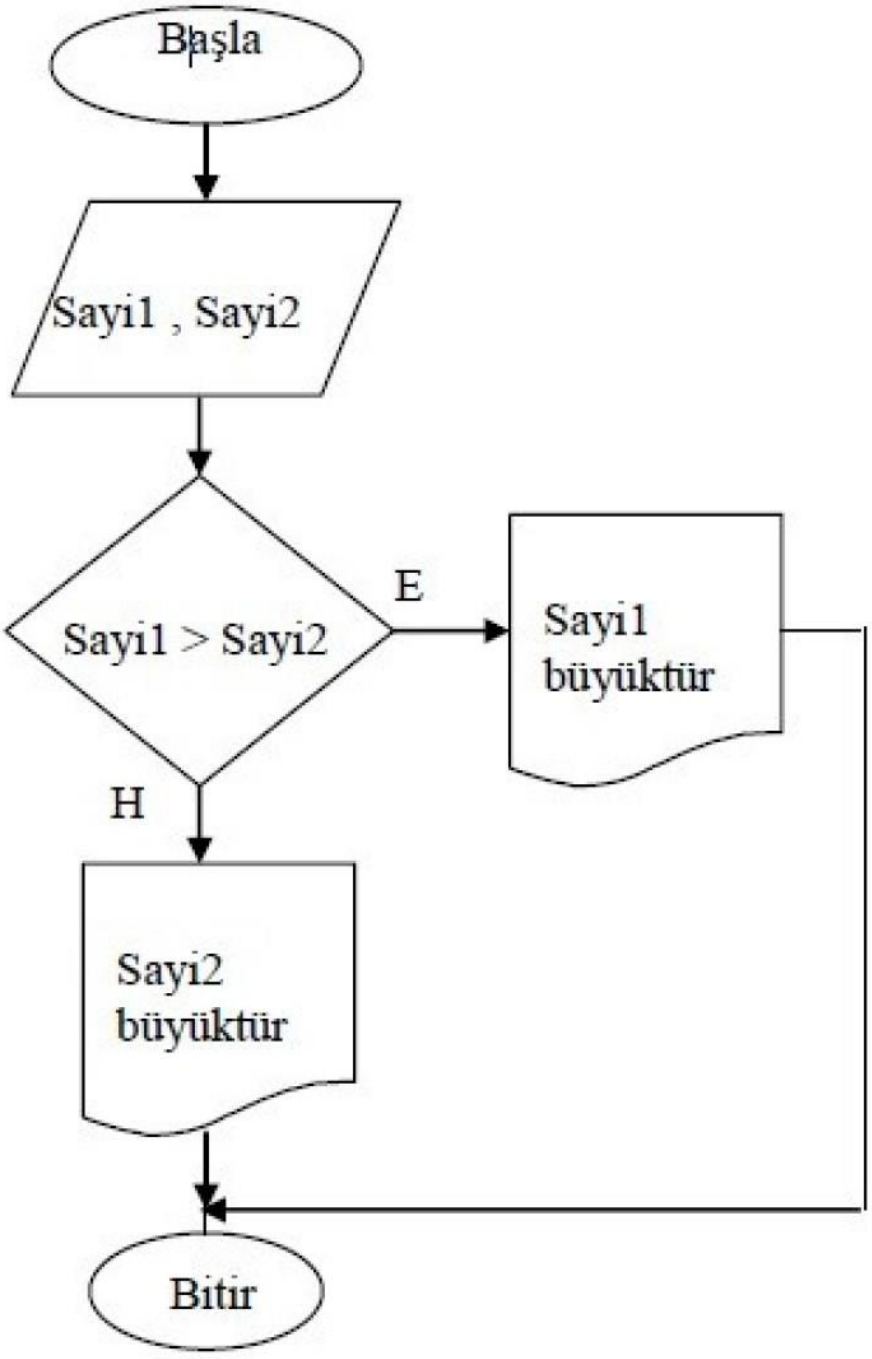
Akış Diyagramı

Bu kapsamlı yazı, akış diyagramı (akış şeması) hakkında tanımlar, tarihçe, kullanım durumları, semboller, ipuçları ve akış şeması başlangıcı için gerekli olan program bilgileri ve akış şeması örneklerini içerir.

Akış Diyagramı Nedir

Bir işlemi, sistemi veya bilgisayar algoritmasını gösteren bir diyagramdır. Net, anlaşılması kolay diyagramlarla genellikle karmaşık süreçleri belgelemek, incelemek, planlamak, iyileştirmek ve iletmek için birçok alanda yaygın olarak kullanılırlar.

Bazen **akış şeması** olarak yazılan akış diyagramları, akış ve sekansı tanımlamak için bağlantı oklarıyla birlikte adım tipini tanımlamak için dikdörtgenler, ovaller, elmaslar ve potansiyel olarak sayısız başka şekiller kullanır. Basit, elle çizilmiş grafiklerden çok sayıda adım ve güzergahı gösteren kapsamlı bilgisayarla çizilmiş diyagramlara kadar çeşitlilik gösterir.
Akış şemalarının tüm biçimlerini göz önüne alırsak, teknik ve teknik olmayan insanlar tarafından çok sayıda alanda kullanılan, gezegendeki en yaygın diyagramlardan biridir. Akış çizelgeleri bazen Süreç Akış Şeması, Süreç Haritası, İşlevsel Akış Şeması, İş Süreçleri Haritalaması, İş Süreçleri Modellemesi ve Notasyonu (BPMN) veya Süreç Akış Şeması (PFD) gibi daha özel isimlerle adlandırılır. Bunlar Veri Akış Diyagramları (DFD'ler) ve Unified Modeling Language (UML) Aktivite Diyagramları gibi diğer popüler diyagramlarla ilgilidir.



Akış Diyagramı Tarihçesi

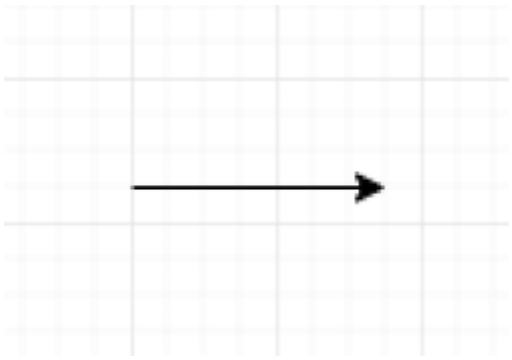
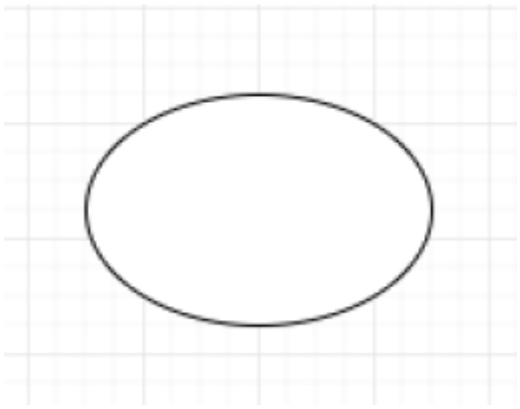
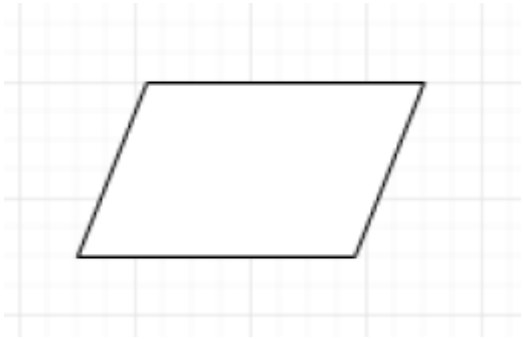
İş süreçlerini belgelemek için akış çizelgeleri 1920'lerde ve 30'larda kullanılmıştır. 1921 yılında, endüstri mühendisleri Frank ve Lillian Gilbreth, Amerikan Makine Mühendisleri Derneğine (ASME) "Akış Süreci Şeması" nı sundu.

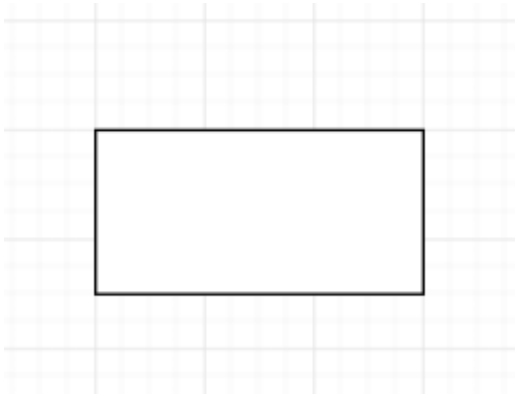
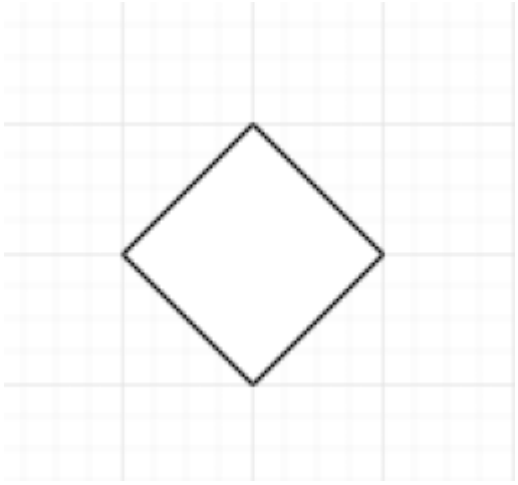
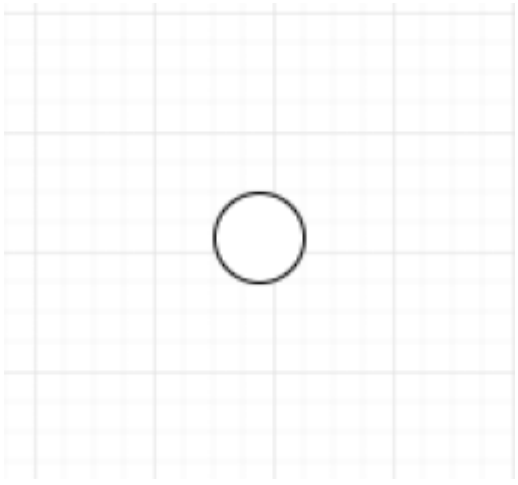
1930'ların başlarında, endüstri mühendisi Allan H. Morgensen, şirketindeki iş adamlarına işlerini daha verimli hale getirmek için konferanslar sunmak için Gilbreth'in araçlarını kullandı. 1940'larda, iki Morgensen öğrencisi, Art Spinanger ve Ben S. Graham, metotları daha geniş bir

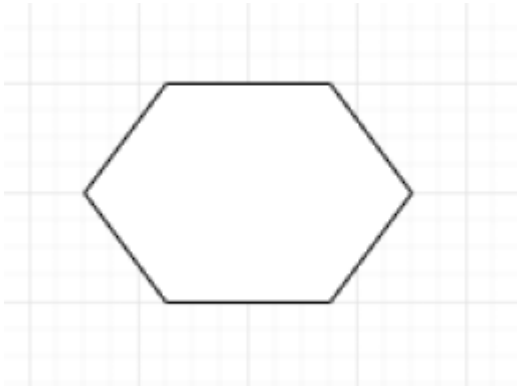
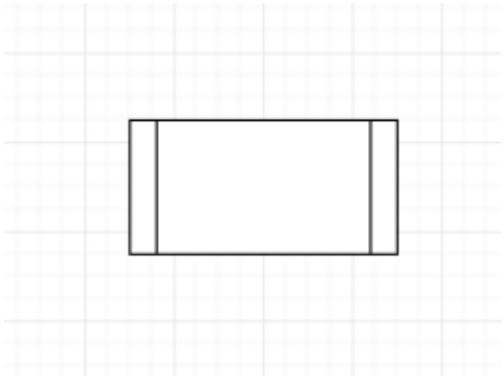
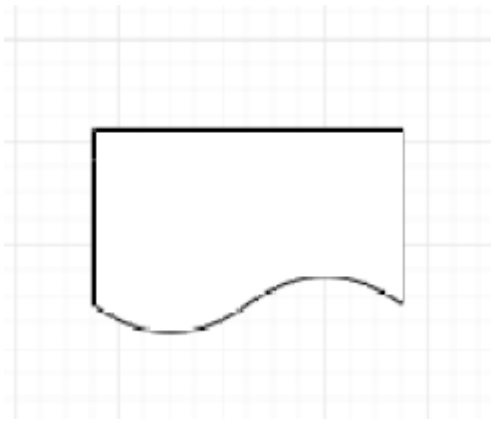
alana yaydı. Spinanger, Procter ve Gamble'a iş basitleştirme yöntemlerini tanıttı. Standard Register Industrial'ın direktörü olan Graham, akış işlem çizelgelerini bilgi işlemeye uyarladı. 1947'de ASME, Gilbreth'in orijinal eserinden türetilen Flow Process Charts için bir sembol sistemi benimsedi.

Akış Diyagramı Şekilleri

Akış şemasındaki farklı durumlar için farklı semboller kullanılır, Örneğin: Giriş / Çıkış ve karar verme farklı sembollere sahiptir. Aşağıdaki tabloda akış çizelgesi hazırlanmasında kullanılan çoğu semboller açıklanmaktadır. **Akış şeması sembolleri** olarak da bilinen liste şu şekildedir.

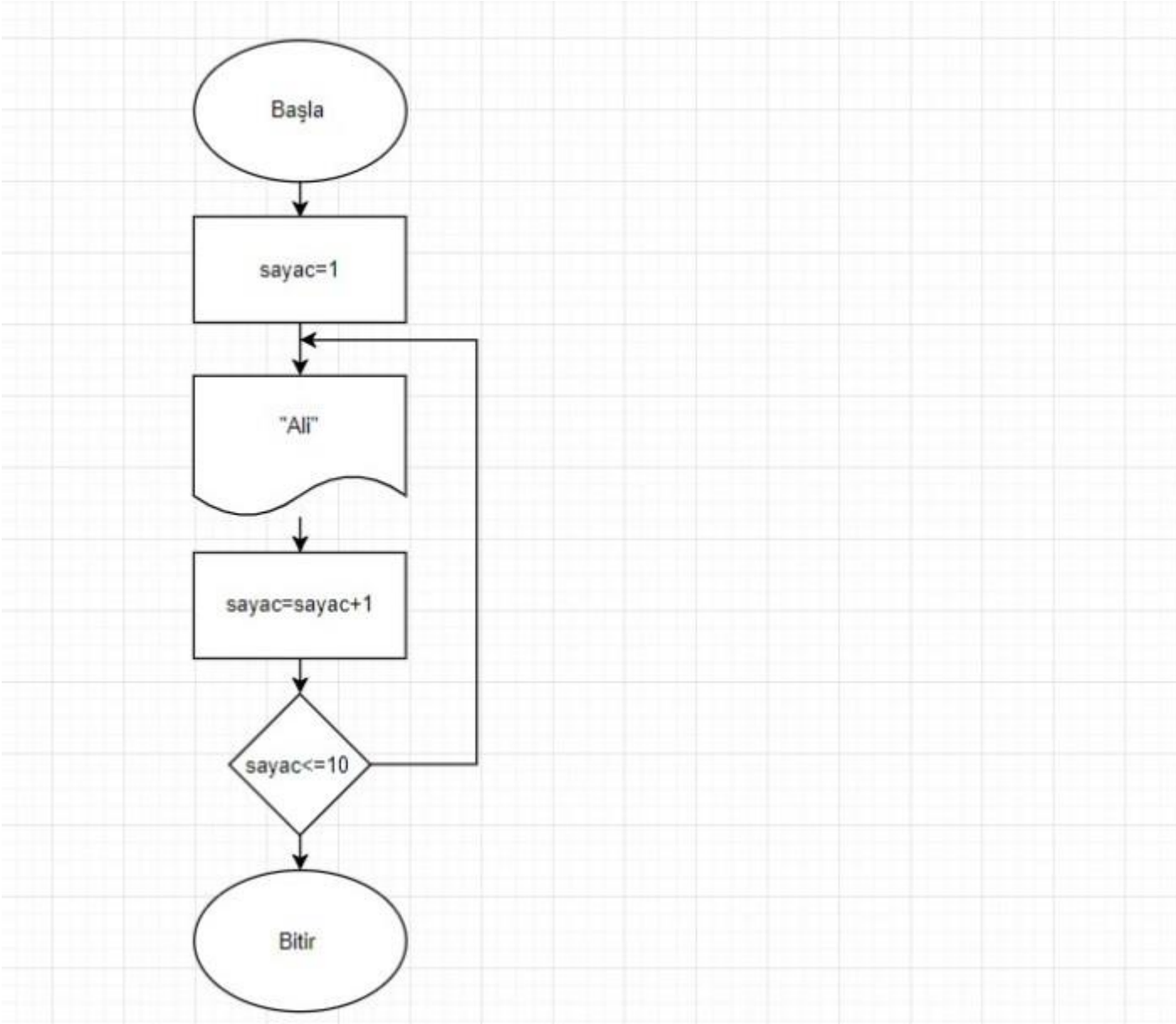
Sembol	**Amaç**	**Tanım**
	Akış Yönü	Akışın yönünü göstermek için kullanılır.
	Akışı Başlat/Durdur	Akış şemasının başlaması ve durması için kullanılır.
	Giriş / Çıkış	Bilgi giriş çıkışı için kullanılır. (Kullanıcıdan bilgi alma ve ekrana bilgi verme)

Sembol	Amaç	Tanım
	İşlem	Veri üzerinde aritmetik ve mantıksal işlemler yapmak için kullanılır.
	Karar	İşlem sonucunda iki yöne dallanma imkanı sunar. İşlem doğru ise bir yöne, yanlış ise diğer yöne dallanma sağlanır.(Mantıksal kararları yapmak için kullanılır.)
	Sayfa bağlantı konnektörü (Düğüm)	Farklı akış şemasını yada kollarını bağlamak için kullanılır.

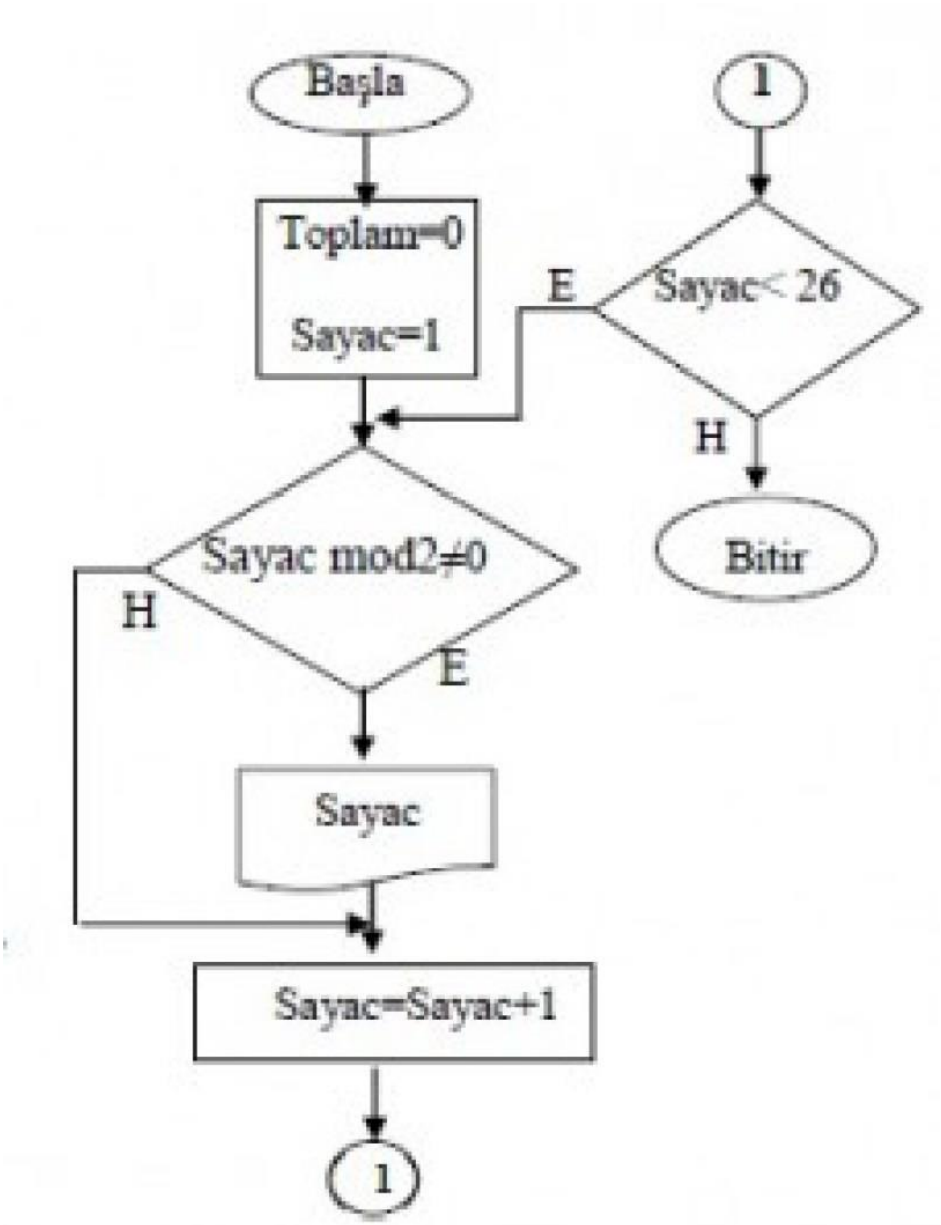
Sembol	Amaç	Tanım
	Döngü (Tekrar)	Düzenli tekrar eden durumları uygulamak için kullanılır.
	Alt program	Bir işlem sürecinde yapılan bir grup işlemi göstermek için kullanılır. Alt program yada alt fonksiyon olarak ifade edilir.
	Çıktı	Akışı yazdırmak için kullanılır. (Genellikle yazıcı ile…)

Örnek 1- Ekrana 10 defa programcının adını yazan algoritmayı yapınız".

```
1  BAŞLA
2  Sayac=1
3 YAZ "AHMET"
4  Sayac=Sayac+1
5 EĞER Sayac<=10 İSE GİT Adım 3
6  Dur
```



Örnek 2 1'den 100'e kadar tek sayıları yazdıran algoritma ve akış diyagramını yapınız.
Akış Diyagramı

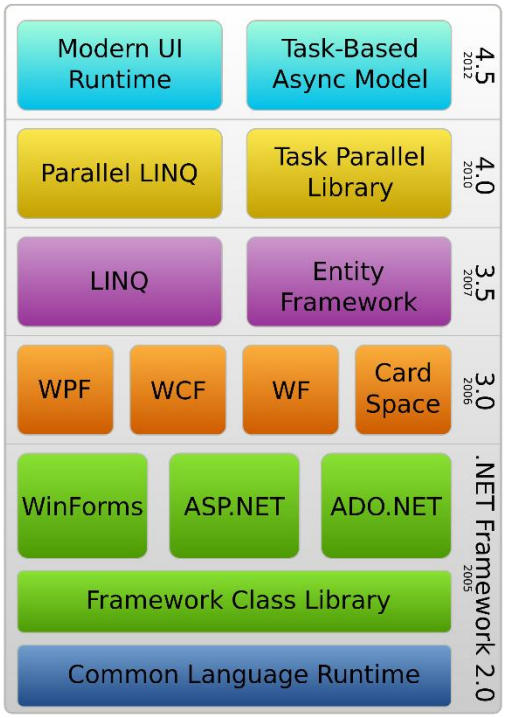

Başla
Toplam=0
Sayac=1
Sayac mod2≠0
H
E
Sayac
Sayac=Sayac+1
1
1
Sayac< 26
E
H
Bitir

.Net Framework

.Net Framework, içerisinde çeşitli kütüphaneler barındıran, internet protokolleri ve standartları kullanan, program yürütme ve makine – kullanıcı iletişimi sistemleri ile uygulama geliştirme ortamı sağlayan bir çatıdır.

- **Bellek yönetimi** : Birçok programlama dilinde, programcılar belleği tahsis etme ve serbest bırakmadan ve nesne kullanım ömrü işlemesinden sorumludur..NET Framework uygulamalarında CLR, uygulama adına bu hizmetleri sağlar.
- **Ortak tür sistemi** : Geleneksel programlama dillerinde, temel türler diller arası çalışabilirliği karmaşıklaştıran bir derleyici tarafından tanımlanır..NET Framework'de, temel türler .net Framework tür sistemi tarafından tanımlanır ve .NET Framework'ü hedefleyen tüm diller için ortaktır.
- **Kapsamlı bir sınıf kitaplığı** : Düşük düzeydeki programlama işlemlerini kullanmak için çok büyük miktarda kod yazmak yerine programcılar .NET Framework Sınıf Kitaplığı'ndan alınan tür ve üyelerin erişilebilir kitaplığını kullanabilirler.
- **Geliştirme çerçeveleri ve teknolojileri** : .NET Framework; web uygulamaları için ASP.NET, veri erişimi için ADO.NET ve hizmet odaklı uygulamalar için Windows Communication Foundation gibi uygulama geliştirmenin belirli alanlarına ilişkin kitaplıklar içerir.
- **Diller arası çalışabilirlik** : .NET Framework'ü hedefleyen dil derleyicileri, ortak dil çalışma zamanı tarafından çalışma zamanında derlenmiş Ortak Ara Dil (CIL) adında bir ara kod üretir.Bu özellik ile, tek bir dilde yazılan yordamlar diğer diller için erişilebilir ve programcılar tercih ettikleri dil veya dillerde uygulamalar oluşturmaya odaklanabilirler.
- **Sürüm uyumluluğu :** Nadir istisnalar ile, belirli bir .net Framework sürümü kullanılarak geliştirilen uygulamalar, sonraki bir sürümü üzerinde hiçbir değişikliğe gerek olmadan çalıştırabilirsiniz.
- **Yan yana yürütme :** .NET Framework, ortak dil çalışma zamanının birden çok sürümünün aynı bilgisayarda bulunmasına izin vererek sürüm çakışmalarının çözümlenmesine yardımcı olur.Bu da, uygulamaların birden çok sürümünün birlikte var olabileceği ve bir uygulamanın, oluşturulduğu .NET Framework sürümünde çalıştırılabileceği anlamına gelir.
- **Çoklu Sürüm Desteği** : .NET Framework taşınabilir sınıf kitaplığı hedefleyen tarafından geliştiriciler platformlarda birden çok .NET Framework, Windows 7, Windows 8, Windows 8.1, Windows Phone ve Xbox 360 gibi iş derlemeleri oluşturabilir.

Ne zaman çıktı?

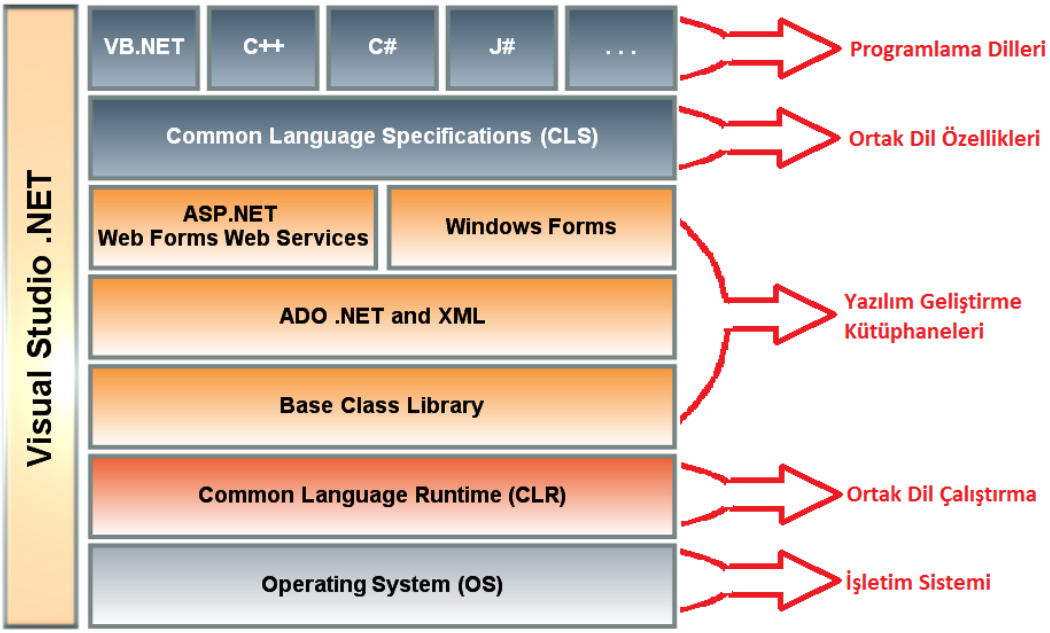


Version number	CLR version	Release date
1.0	1.0	2002-02-13
1.1	1.1	2003-04-24
2.0	2.0	2005-11-07
3.0	2.0	2006-11-06
3.5	2.0	2007-11-19
4.0	4	2010-04-12
4.5	4	2012-08-15
4.5.1	4	2013-10-17
4.5.2	4	2014-05-05
4.6	4	2015-07-20
4.6.1	4	2015-11-17

1990'lı yıllarda .Net Framework'un geliştirmesine başlayan Microsoft, orjinal ismi "Next Generation Windows Services" (Yeni Nesil Windows Servisleri) olan .Net 1.0'ın beta sürümünü Ağustos 2000'de yayınladı. .Net 1.0 tam sürümü ise 13 Şubat 2002 yılında yayınlandı.

İlk versiyondan sonra Microsoft dokuz güncelleme daha yayınladı. Bunlardan yedi tanesi Visual Studio'nun yeni versiyonu olarak yayınlanırken geriye kalan iki tanesi ise eski CLR (Common Language Runtime) sisteminin güncellenmesidir.

.Net Framework Yapısı

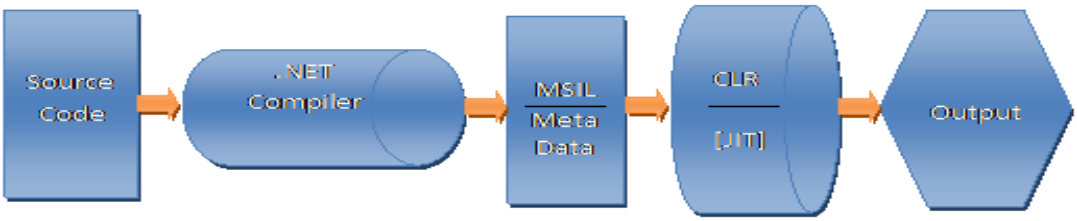


Yukarıdaki resimde görünen yapıyıincelemek gerekirse; **programlama dilleri** programcıların bir bir algoritmayı takip ederek bilgisayara ne yapması gerektiğini anlattığı farklı özelliklere sahip iletişim araçlarıdır.

Programlama dilleri ile yazılan kodların geçtiği sonraki aşama **CLS(Common Language Specifications)** kısmıdır. |> CLS yapısında .NET çatısı altındaki programlama dillerinin ortak noktalarını barındırır. Bünyesinde barındırdığı birtakım yapıları ve kısıtları ile kütüphane(library) ve derleyici(compiler) yazabilmek için rehberlik yapmaktadır. CLS, yazılan bir kütüphanenin CLS'yi destekleyen diğer programlama dilleri ile entegre şekilde çalışabilmesini ve bu diller tarafından da kullanılabilmesini sağlamaktadır. CLS'nin kriterleri ve kuralları göz önünde bulundurularak yazılan bir API(Application Program Interface) diğer programlama dilleri içerisinden kullanılabilmekte. <|

CLS içinde **CTS (Common Type System) (Ortak Tür Sistemi)** diye bir yapı vardır. |> Ortak tür sistemi .NET çatısında herhangi bir programlama diliyle yazdığınız bir bileşenin (Dll kütüphanesi veya User Control Nesneleri gibi) başka herhangi bir .NET dili ile kullanabilmenize olanak sağlar. CTS yapısında temel nesne türlerini barındırır. Bunlar Object, Integer, String, Char, Double, Decimal… vb. türlerdir. <|

Yazılım Geliştirme Kütüphaneleri framework çatısı altındaki uygulamalar için hazırlanmış kütüphanelerdir. Örnek vermek gerekirse windows forms uygulamaları için hazırlanmış form classlarının ve tool classlarının bulunduğu, mvc için action classları gibi controller classları gibi classların bulunduğu kütüphanelerdir. O uygulama ve ya teknoloji için tekrardan buttonların ve uygulamaya özel yapıların kodlarının yazılasına gerek kalmaz, bu kütüphanelerde hazır bulunur.



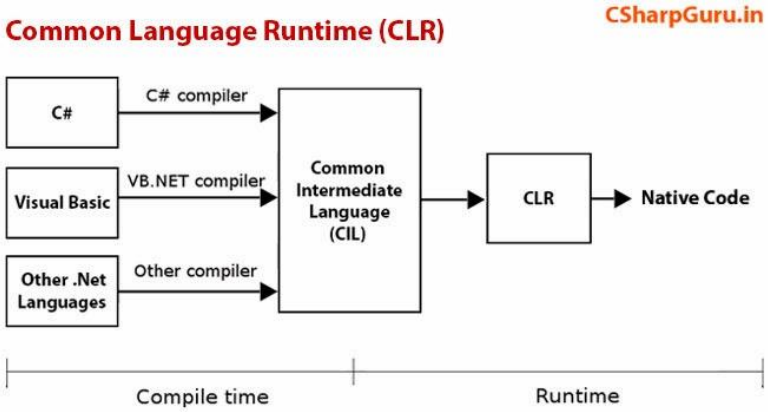
|> **CLR(Common Language Runtime) (Ortak Dil Çalıştırma)** tüm dillerin derlenmesinde sanal makine (programları gerçek bir bilgisayar sistemindeki gibi çalıştıran mekanizmaların yazılım uyarlamasıdır. Sanal Makine, işletim sistemi ile bilgisayar platformu arasında bir sanal ortam yaratır ve bu sanal ortam üzerinde yazılımların çalıştırabilmesine olanak sağlar) gibi çalışır. Tüm .net dilleri CLR tarafından dayatılan kural ve standartlara uymak zorundadır. Örneğin :

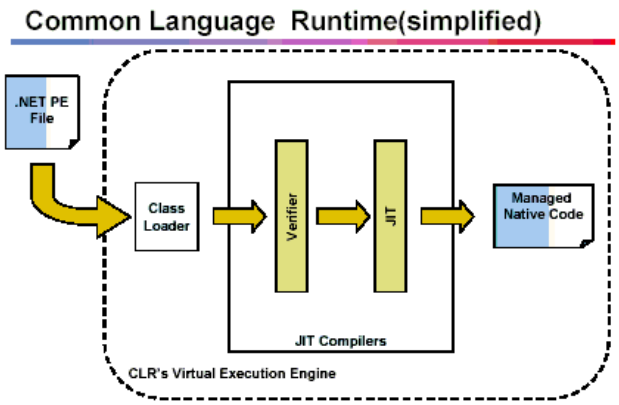
Nesne bildirimi, oluşumu ve kullanımı : CLR nesneyi zorunlu kılar. (Nesne : etrafımızda gördüğümüz ve tabiri eğer uygunsa "şey" olarak tanımladığımız varlıktır. Örnek verirsek; kalem, kağıt, defter, ben, onlar, Ahmet vs bunların hepsi birer nesnedir diyebiliriz)

Data türleri (.NET Framework çatısı altında çalışacak dillerin uyması gereken tür tanımlama standartlarıdır. Hangi veri türünü kullanacağı ve bu türlerin bellekte kaç byte yer işgal edeceği gibi)

Hata ve İstisna Yakalama (Istisnai durumlar programlarin çalisma zamaninda hata üretmeleri veya kodda beklenmedik durumlarin olusmasi sonucunda CLR tarafindan üretilirler. <|

|> CLR .NET Framework için yazılan uygulamaların çalışmasını sağlayan birimdir. Bir .NET uygulaması yazdığınızda (C#.NET, VB.NET gibi dillerle) elde ettiğiniz .exe uzantılı dosya aslında çalıştırılabilir bir dosya değildir. Bu çıktı bir





MSIL (Microsoft Intermediate Language) dosyasıdır bu dosyaya taşınabilir Assembly de diyebiliriz. CLR yüklü olan bir bilgisayarda bu uygulamayı çalıştırdığınızda otomatik olarak JIT (Just-In-Time) derleyicisi çağrılır ve JIT kodunuzun kullanılan bölümlerini uygun işlemci Assembly kodlarına dönüştürür ve bu sayede uygulamanız .NET Framework yüklü olan her platformda sorunsuz bir şekilde çalışabilir. Bu işlemin amacı programınızın taşınabilir olmasıdır. <|

MSIL (MicroSoft Intermediate Language) : Microsoft .NET'i çıkarmasının amacı Java'ya rakip olabilmekti. Hangi alanda rakip olacaktı peki JAVA'ya. İlk çıktığı yıllar birinci amaç JAVA gibi her yerde çalışabilir olmasını sağlamaktı. Bu uyumu çok iyi yaparlarsa .NET'in de kullanabileceği alan yaygınlaşacaktı.

Microsoft Intermediate Language denilen kavram ise bu düşüncenin biraz daha geliştirilmiş hali. Şöyle ki .NET'in içerisin de farklı kodlar yazabiliyoruz. Bunların en bilineni ise C# ve Visual Basic .NET diyelim. Bu iki dilde de bir uygulama yaptık. Bir birinden farklı uygulamalar diyelim.

Uygulamalarımız derlediğimiz an arka planda bambaşka bir durum oluyor. Hangi programlama dili olmasına bakmaksızın Microsoft Intermediate Language adlandırılan ortak bi dil yapısına dönüşüyor programlarımız.

Bu durum şu kolaylığı sağlıyor, .NET platformlarının bütün dillerinin başka bir yükleme gerektirmeden kullanabilmesine olanak sağlamakta.

JIT (Just In Time) (O anda – Çalışma anında derleme): CLR ,programlarımızı değişik şekilde derleyebilir. Varsayılan derleme türü JIT(Just IN TIME- çalışam anında derleme) 'dır. Program çalışırken daha önce derlenmemiş bir parçasına gelince hemen o kısmı da derler ve bunu hafızda chach'e koyar. Tekrar aynı program parçasını çalıştırmak gerekirse burayı hafızadan çalıştırır. Eğer RAM 'imizi yeteri kadar büyükse, programın tamamı derlenmiş ve hafızada depolanmış durumda olabilir. Bu durumda programımız çok hızlı çalışır.

Hafızamızın yeteri kadar büyük olmadığı durumlarda EconoJIT (Ekonomik JIT) derleyicisini kullanabiliriz. Bu derleyici ile programın derlenmiş kısımları hafızada depolanmaz ve her seferinde aynı program parçası derlenir. Tabi ki bu derleyici normal JIT'e göre programlarımızı daha yavaş çalıştırır. Ama RAM 'imizi çok daha az kullanır.

CLR ile gelen bir başka çeşidi olan PreJIT(ön JIT derleyicisi) ise derleme işini program çalışmadan önce yapar ve tüm makine kodlarını bir yerde saklar. Çalışma anında çok hızlı olan programımız diğer JIT derleyicileriyle derlenmiş olanlara nazaran çok hızlı çalışır.

Kısaca yazılan kodlar derlendiğinde doğrudan windows işletim sistemleri üzerinde çalışmaz; üzerinde kurulan sanal CLR üzerinde çalışır. Çünkü kodlar derlendiğinde çalışabilir(executable) kodlara değil, önce MSIL kodlarına dönüşmektedir. CLR programımızın çalışmasını idare eden sistemdir. Örneğin C# dilinde int tipi bir değişken tanımlandığında, derleyici onu .NET Framework karşılığı olan Int32 tipine dönüştürür. Kaynak kodu hangi dilde yazarsanız yazın, deklare edilen her şey .NET'deki karşılıklarına dönüşür. Yani .NET'in desteklediği herhangi bir dilde yazılan veri tipleri .NET'in CTS(Common Type System) denilen veri tiplerine dönüşür.

C# Console Application

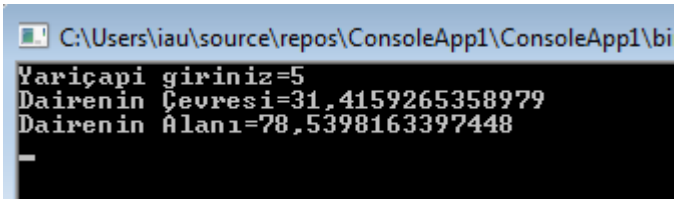
Değişken tanımlamaları

```
int sayi = 5;
int x, y = 8, z;//Aynı türdeki değişkenler bir arada kullanılabilir.
bool aktif = true;
float f=3.14f;//Değerin sonuna eklediğimiz f harfi değerin float olduğunu gösterir.
double d=3.2;
char c = 'c';//Char tipindeki değişkenler tek tırnak içine yazılmalıdır.

string ad = "Yaren Nilay";//String tipibdeki ifadeler çift tırnak kullanılarak
yazılmalıdır.
```

Klavyeden girilen yarıçap değerleriyle dairenin çevresini ve alanını hesaplayan kod satırlarını yazınız.

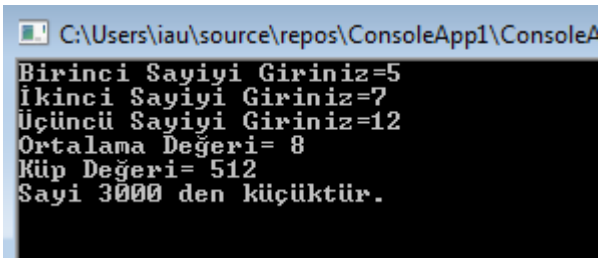
```
namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Yariçapi giriniz=");
            double yaricap = Convert.ToInt16(Console.ReadLine());
```

```
            double cevre,alan;
            cevre = 2 * Math.PI * yaricap;
            alan = Math.PI * Math.Pow(yaricap, 2);
            Console.WriteLine("Dairenin Çevresi="+cevre);
            Console.WriteLine("Dairenin Alanı="+alan);
            Console.ReadKey();

        }
    }
```



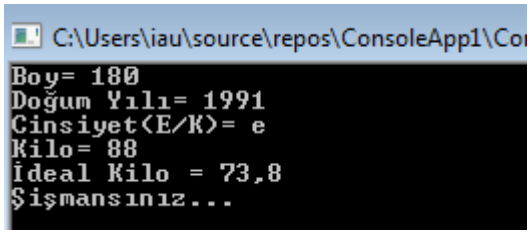
Klavyeden girilen 3 tane sayının ortalamasını bulup küpünü alan,3000 den büyük veya küçükse ekrana yazdıran kod satırlarını yazınız

```
            int sayi1, sayi2, sayi3;
            double ortalama,kup;
            Console.Write("Birinci Sayiyi Giriniz=");
            sayi1 = Convert.ToInt16(Console.ReadLine());
            Console.Write("İkinci Sayiyi Giriniz=");
            sayi2 = Convert.ToInt16(Console.ReadLine());
            Console.Write("Üçüncü Sayiyi Giriniz=");
            sayi3 = Convert.ToInt16(Console.ReadLine());

            ortalama = (sayi1+sayi2+sayi3)/3;
            kup = Math.Pow(ortalama, 3);
            Console.WriteLine("Ortalama Değeri= " +ortalama);
            Console.WriteLine("Küp Değeri= " +kup);

            if (kup < 3000) { Console.Write("Sayi 3000 den küçüktür."); }
            else { Console.Write("Sayi 3000 den büyüktür."); }


            Console.ReadKey();
```



Klavyeden girilen boy, kilo ve yaş bilgileriyle ideal kilonuzu hesaplayan kod satırlarını yazınız.

```
Console.Write("Boy= ");
int boy = Convert.ToInt16(Console.ReadLine());

Console.Write("Doğum Yılı= ");
int dYili = Convert.ToInt16(Console.ReadLine());

Console.Write("Cinsiyet(E/K)= ");
string cinsiyet =Console.ReadLine();

Console.Write("Kilo= ");
int kilo = Convert.ToInt16(Console.ReadLine());

double k, ideal;

if(cinsiyet == "E" || cinsiyet == "e"){ k = 0.9; }
else if (cinsiyet == "K" || cinsiyet == "k") { k = 0.8; }
else { k = 0; }

int yas = DateTime.Now.Year - dYili;
ideal = (boy - 100 + yas / 10) * k;

Console.WriteLine("İdeal Kilo = " + ideal);

if (ideal == kilo) { Console.WriteLine("Fitsiniz..."); }
else if (ideal < kilo) { Console.WriteLine("Şişmansınız..."); }
else if(ideal > kilo) { Console.WriteLine("Zayıfsınız..."); }

Console.ReadKey();
```

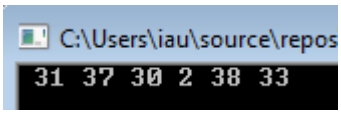


Ekrana 1 ile 10 arasında rastgele sayı üreten kod satırlarını yazınız.

```
Random rastgele = new Random();
int sayi = rastgele.Next(1, 10);
Console.Write(sayi);
```

1 den 49 a kadar olan sayılardan rastgele altı tane seçip sayısal loto örneği kod satırlarını yazınız.

```
Random rastgele = new Random();

for(int i = 0; i < 6; i++)
{   int sayi = rastgele.Next(1, 50);
    Console.Write(" " +sayi);}


Console.ReadKey();
```



10 satır 6 kolon olarak sayısal loto tahmini yapan program

```
Random rnd = new Random();


        for(int i=0;i<=10;i++)
        {
           for (int j = 0; j < 6; j++)
           {
              int sayi = rnd.Next(1, 50);
              Console.Write(sayi + " ");
           }
           Console.WriteLine();
```