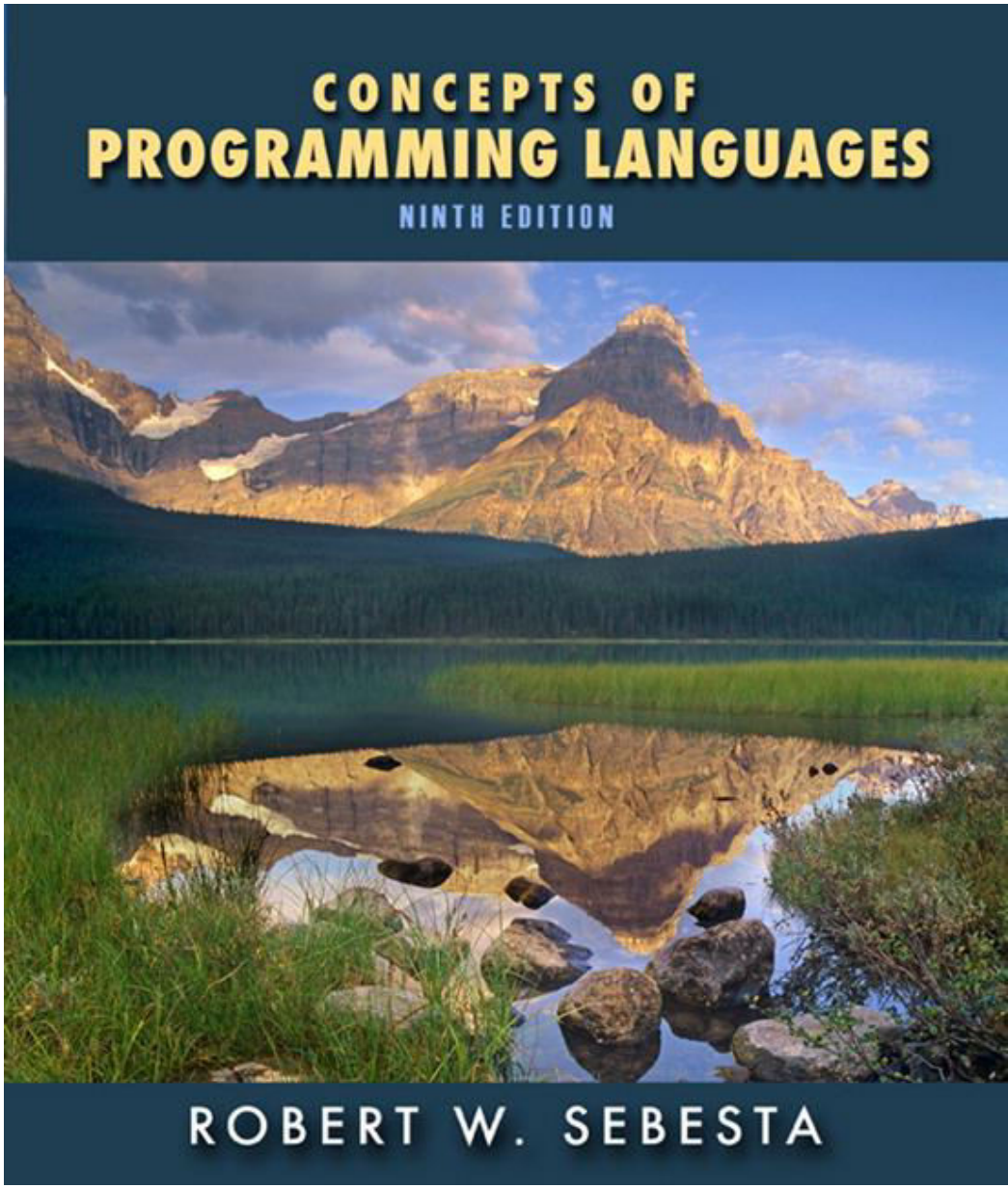
# Bölüm 1

## Hazırlık



# Bölüm 1 Konular

- Programlama Dilleri Kavramlarının Öğrenilmesinin Nedenleri
- Programlama Alanları
- Dil Değerlendirme Kriterleri
- Dil Tasarımına Etki Eden Faktörler
- Dil Kategorileri
- Dil Tasarım Ödünleşmesi (Bir şeyi kazanmak için başka bir şeyden fedakarlık etme)
- Uygulama Yöntemleri
- Programlama Çevresi

# Programlama Dilleri Kavramlarının Öğrenilmesinin Nedenleri

- Fikirlerin ifade edilmesi için artırılmış yetenek
- Uygun dillerin seçimi için geliştirilmiş geçmiş
- Yeni dilleri öğrenebilmek için artırılmış yetenek
- Uygulamanın önemini daha iyi anlama
- Bilinen dillerin daha iyi kullanımı
- Hesaplamanın etraflıca ilerletilmesi

# Programlama Alanları

- Bilimsel uygulamalar (scientific)
  - Kayan noktalı hesaplamaların büyüklüğü; dizilerin kullanımı
  - Fortran
- İş uygulamaları (business)
  - Raporların üretimi, ondalık rakamların ve karakterlerin kullanımı
  - COBOL
- Yapay zeka (artificial intellegence)
  - Sayılarla uğraşmak yerine sembollerin kullanılması; yönlendirilmiş listelerin kullanımı
  - LISP
- Sistem programlaması (system programming)
  - Sürekli kullanım nedeniyle  verim gerektirir
  - C
- Web yazılımı (web software)
  - Dillerin seçim koleksiyonu: biçimleme (XHTML), komut dizisi oluşturma (PHP), genel amaçlı (Java)

# Dil Değerlendirme Kriterleri

- **Okunabilirlik–readability**: programın okunabilir ve anlaşılabilir kolaylığı
- **Yazılabilirlik–writability**: program oluşturmada kullanılacak dilin kolaylığı (uygunluğu)
- **Güvenilirlik–reliability**: şartnamelerin uygunluğu (örneğin, kendi şartnamelerin yapılması)
- **Maliyet–cost**: nihai toplam maliyet

# Değerlendirme Kriteri: Okunabilirlik

- Bütün yönüyle kolaylık
  - Özelliklerin ve yapılandırma setinin yönetilebilirliği
  - Özellik çeşitliliğinin minimum seviyede tutulması
  - Aşırı işlem yüklenmesinin minimum seviyede tutulması
- Ortogonallik
  - İlkel yapılandırmanın göreceli küçük bir seti, göreceli küçük bir sayı usulleri ile birleştirilebilir
  - Mümkün olan her kombinasyon yasaldır
- Veri tipleri
  - Uygun ön tanımlı veri tipleri
- Sözdizimin (syntax) kurallarının göz önüne alınması
  - İşaretleyici formlar: esnek kompozisyonlar
  - Bileşim ifadelerin şekillendirilmesinin özel kelimeleri ve yöntemleri
  - Biçim ve anlamlar: kendinden tanımlı yapılar, anlamlı anahtar sözcükler

# Değerlendirme Kriteri: Yazılabilirlik

- Basitlik ve Ortogonallik
  - Daha az yapılar, ilkel küçük sayıları, onları birleştirmek için küçük sayıda kurallar seti
- Soyutlamayı desteklemeli
  - Detayların ihmal edilmesine müsaade edecek biçimde kompleks yapıların tanımlanması ve kullanılması yeteneği
- Anlatımcılık
  - Belirlenmiş işlemlerin göreceli uygun usullerinin bir seti
  - Dayanıklık ve işlemlerin sayısı ve ön tanımlı fonksiyonlar

# Değerlendirme Kriteri: Güvenirlirlik

- Tip kontrolü
  - Tip hatalarının test edilmesi
- Hariç tutma kullanımı
  - Run-time hata kesişmesi ve  düzeltici ölçmelerin alınması
- Örtüşme
  - Aynı hafıza lokasyonu için bir veya daha fazla farklı referans verme yöntemlerinin mevcudiyeti
- Okunabilirlik ve yazılabilirlik
  - Bir algoritmayı ifade etmede «doğal» yolları desteklemeyen bir dil, «doğal olmayan» yaklaşımların kullanılmasını gerektirir ve bu yüzden de güvenliği azalır.

# Değerlendirme Kriteri: Maliyet

- Dili kullanmak için programcıların eğitimi
- Programların yazılması (Belirli uygulamalara yakınlık)
- Programların derlenmesi
- Programların çalıştırılması
- Dil uygulama sistemi: ücretsiz derleyicilerin mevcudiyeti
- Güvenilirlik: Zayıf güvenilirlik maliyetlerin artmasına neden olur
- Programların bakımı

# Değerlendirme Kriteri: Diğerleri

- Taşınabilirlik
  - Bir uygulamadan diğer bir uygulamaya bir programın taşınabilirliğindenki kolaylık
- Genellik
  - Uygulamaların geniş bir alana uygulanabilirliği
- İyi tanımlama
  - Dilin resmi tanımının bütünlüğü ve kesinliği

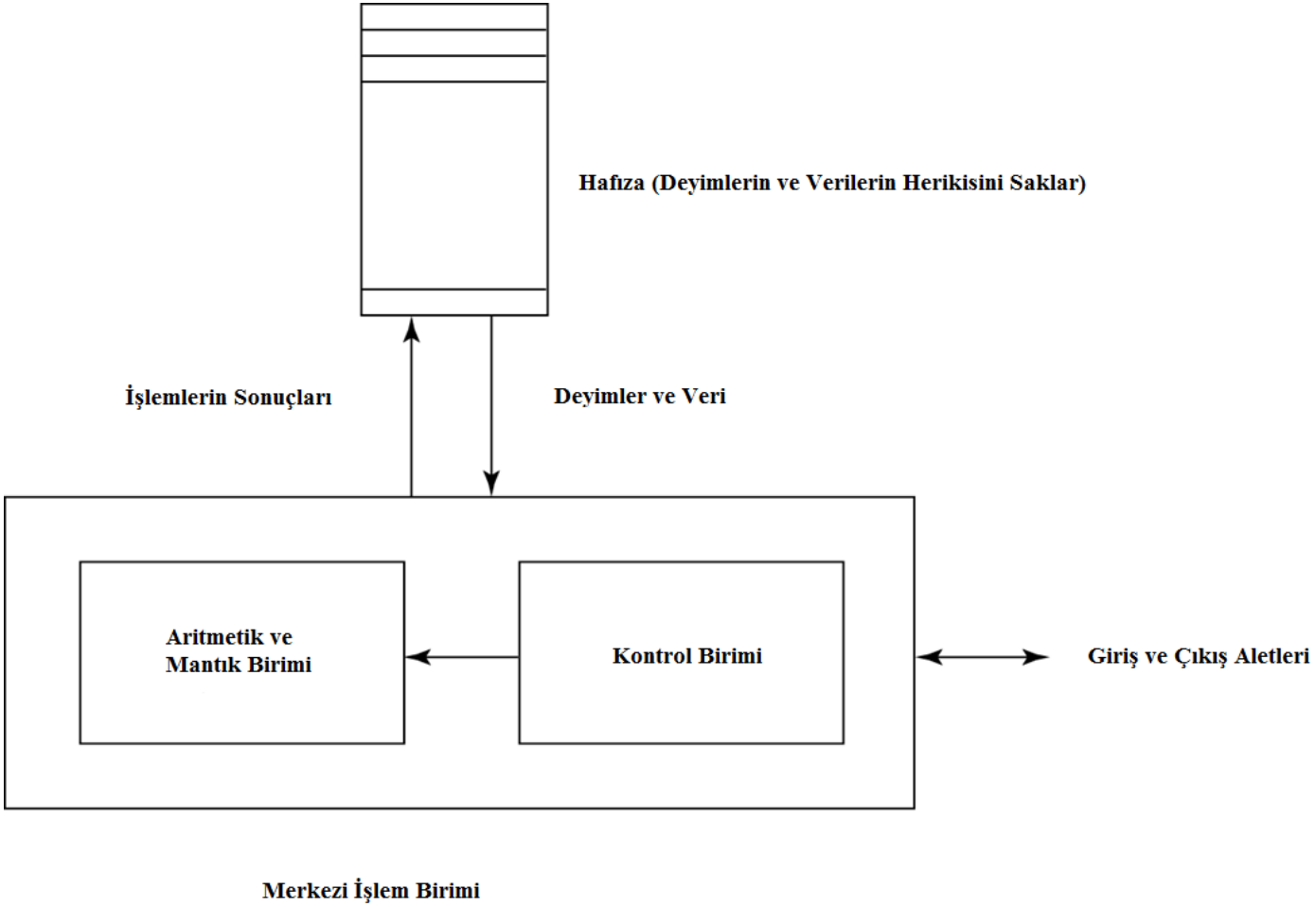
# Dil Tasarımına Etki Eden Faktörler

- Bilgisayar Mimarisi
  - Diller *von Neumann* mimarisi olarak bilinen yaygın bilgisayar mimarisi üzerine geliştirilir
- Programlama metodolojileri
  - Yeni yazılım geliştirme metodolojileri (örneğin nesne tabanlı yazılım geliştirilmesi) yeni programlama paradigmalarının ve uzantılarının, yeni programlama dillerinin doğmasına neden olmaktadır

# Bilgisayar Mimarisi Etkisi

- İyi bilinen bilgisayar mimarisi: Von Neumann
- von Neumann bilgisayarları nedeniyle, emir dilleri daha baskındır
  - Veri ve programlar hafızada saklanır
  - Hafıza CPU'dan ayrıdır
  - Komutlar ve veriler hafızadan CPU'ya iletilir
  - Emir dillerini esas alır
    - Değişken model hafıza hücreleri
    - Atama ifadeleri model iletme
    - İterasyon (adım adım) etkindir

# von Neumann Mimarisi



# von Neumann Mimarisi

- Fetch (bilgisayarda emirlerin getirilmesi)-çalıştırma-döngüsü (von Neumann mimarisi bilgisayarda)

```
Program sayıcısını başlangıç durumuna getir
repeat sureklidon
  sayac tarafindan isretlenen deyimleri getir
  sayaci artir
  komutu coz
  komutu calistir
end repeat
```

# Programlama Metodolojilerinin Etkileri

- 1950li yıllar ve1960'lı yılların başı: Basit uygulamalar; makine verimliliği hakkında kaygılar
- 1960'lı yılların sonu: İnsan verimliliği önem kazandı; okunabilirlik, daha iyi kontrol yapıları
  - Yapılandırılmış programlama
  - Yukarıdan aşağıya tasarım ve adım usulü arıtma (düzeltme)
- 1970'li yılların sonu: İşlem tabanlıdan veri tabanlıya geçiş
  - Veri soyutlama
- 1980'li yılların ortaları: Nesne tabanlı programlama
  - Veri soyutlama + kalıtsallık + Çok biçimlilik

# Dil Kategorileri

- Emir-Imperative
  - Merkezi özelikler değişkenlerdir, atama ifadeleri ve iterasyon
  - Nesne tabanlı programlamayı destekleyen dilleri kapsar
  - Script dillerini içerir
  - Görsel dilleri içerir
  - Örnekler: C, Java, Perl, JavaScript, Visual BASIC .NET, C++
- Fonksiyonel-Functional
  - Hesaplamaları icra ederken asıl araç, verilen parametreler için fonksiyonların uygulanmasıdır
  - Örnek: LISP, Scheme
- Mantıksal- Logical
  - Kurala dayalı (kurallar belirli bir sırada özelleştirilmez)
  - Örnek: Prolog
- Biçimleme/hibrid programlama-Markup/programming hybrid
  - Biçimleme dilleri bazı programlama dillerini desteklemek için genişletilmiştir
  - Örnekler: JSTL, XSLT

# Dil Tasarım Ödünleşim (Trade-Offs)

- Güvenilirlik, yürütme maliyeti
  - Örnek: Java uygun indeksleme için dizi elemanlarının tümünün kontrol edilmesini talep eder, bu durum yürütme maliyetini artırır.
- Okunabilirlik yazılabilirlik

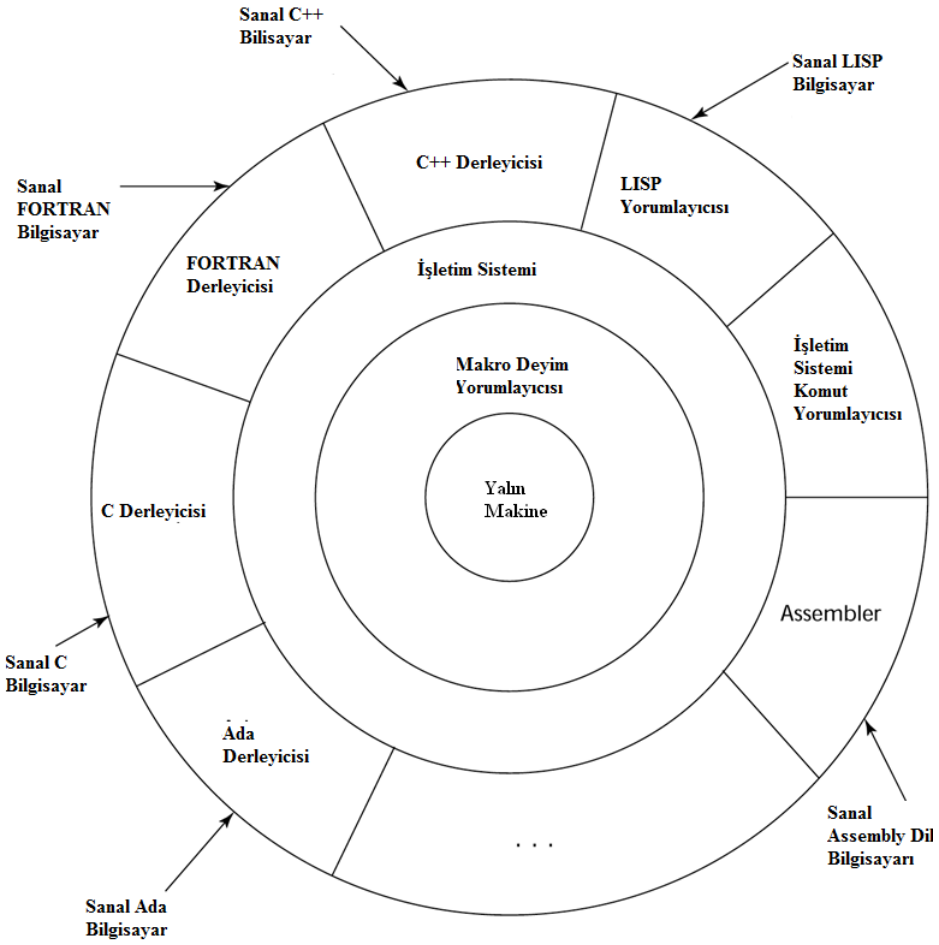
  Örnek: APL birçok güçlü operatörü ( ve büyük miktarlarda yeni sembolleri) sağlar. Bu durum kompakt bir program içinde kompleks hesaplamaların yazılmasına müsaade eder. Fakat zayıf okunabilirlik maliyetin artmasına neden olur.
- Yazılabilirlik (esneklik) ve güvenilirlik
  - Örnek: C++ işaretleyicileri güçlüdür ve çok esnektir, fakat güvenilir değildirler

# Uygulama Yöntemleri

- Derleme- Compilation
  - Programlar makine diline çevrilir
- Sade Yorumlama-Pure Interpretation
  - Programlar, yorumlayıcı olarak adlandırılan diğer bir program tarafından yorumlanır
- Hibrid Yorumlama Sistemleri-Hybrid Implementation Systems
  - Derleyiciler ve sade yorumlayıcılar arasındaki uzlaşmayı sağlar

# Bilgisayarın Tabakalaştırılmış Görünümü

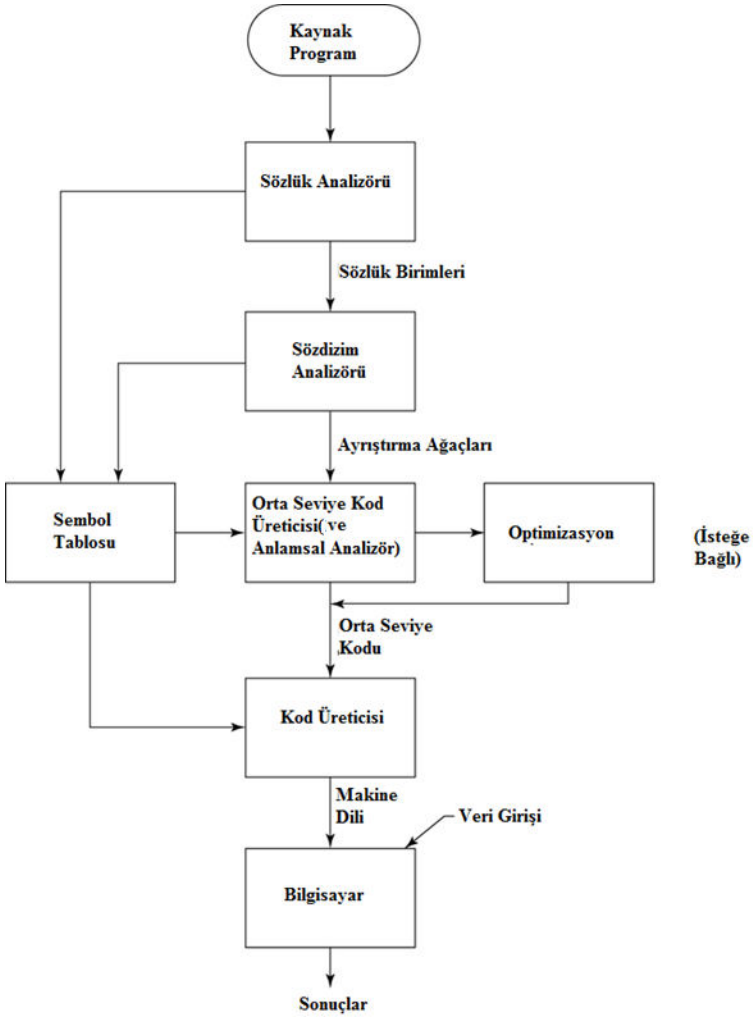
- İşletim sistemi ve dil yorumlaması bir bilgisayarın makine arayüzü üzerinden tabakalaştırılır



# Derleme

- Yüksek seviyedeki programı (kaynak dil) makine koduna (makine diline) çevirir
- Yavaş çevirme, hızlı yürütme
- Yürütme işleminin birkaç fazı vardır:
  - Sözcük analizi- lexical : Kaynak programdaki karakterleri sözcük birimlerine çevirir
  - Söz dizim analizi-syntax: sözcük birimlerini programın sözdizimsel yapısını temsil eden ayrıştırma ağaçlarına (parse units) dönüştürür
  - Anlamsal analiz-semantic: orta düzey kodları üretir
  - Kod üretimi: makine kodu üretilir

# Derleme İşlemi



# İlave Derleme Termolojileri

- **Yükleme modülü** (çalıştırılabilir imajı): kullanıcı ve sistem kodu birlikte
- **Yönlendirme ve yükleme**: sistem program birimlerinin toplanması işlemi ve onları bir kullanıcı programına yönlendirme

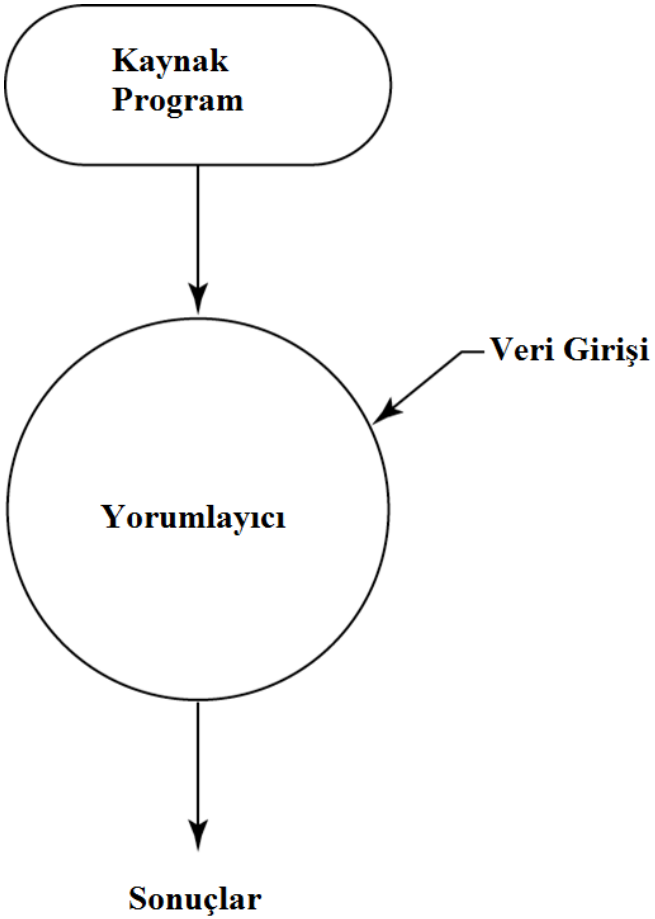
# Von Neumann Darboğazı

- Bir bilgisayar hafızası ile onun işlemcisi arasındaki bağlantı hızı, bilgisayarın hızını tanımlar
- Genelde program komutları (deyimleri) bağlantı hızından daha hızlı çalışır; bu yüzden bağlantı hızı bir dar boğaza (tıkanıklığa) sebebiyet verir
- *Von Neumann darboğazı,* bilgisayar hızını öncelikli sınırlayan faktörüdür

# Sade Yorumlama

- Çevirme yapılmamaktadır
- Programlar daha kolay uygulanır (çalışma zamanındaki hatalar kolayca ve anında görüntülenebilir)
- Daha yavaş çalışma (derlenen programlardan 10'dan 100'e kadar daha yavaştır)
- Genelde daha fazla yer gerektirir
- Geleneksel yüksek seviyeli diller için şimdi daha nadir kullanılır
- Bazı web skript dillerinde tekrar kullanılmaya başlanılmıştır (JavaScript, PHP)

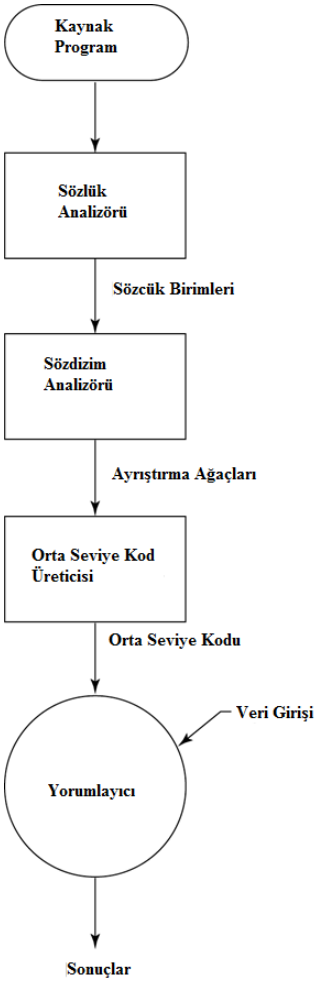
# Sade Yorumlama İşlemi



# Hibrid Yorumlama Sistemleri

- Derleyici ve sade yorumlayıcılar arasında uzlaşmayı sağlar
- Bir yüksek seviye dil programı, kolay yorumlamaya müsaade eden bir orta dil seviyesine çevrilir
- Sade yorumlamadan daha hızlıdır
- Örnekler
  - Yorumlama yapmadan önce hataları bulmak için Perl programlar kısmen derlenir
  - Java'nın başlangıçtaki uygulamaları hibrid idi; orta seviye biçimi, bayt kod; bayt kod yorumlayıcı ve run-time sistemi (birlikte bunlar Java Sanal Makine olarak adlandırılır) olan herhangi bir makineye taşınabilirlik sağlar.

# Hibrid Uygulama İşlemi



# Tam zamanında (Just-in-Time) Uygulama Sistemleri

- Başlangıçta programları orta seviyedeki dile çevirir
- Daha sonra alt programların orta seviyedeki dilini çağrıldıklarında makine koduna derler
- Makine kod versiyonu müteakip çağrılmalar için muhafaza edilir
- Java programları için JIT sistemleri yaygın kullanılır
- .NET dilleri JIT sistemi ile uygulanır

# Ön İşlemciler

- Ön işlemci makroları (deyimleri) genellikle diğer dosya kodlarını içerisine alacak şekilde belirlemek için kullanılır
- Bir ön işlemci, gömülü ön işlemci makrolarını genişletmek için program derlenmeden önce bir programı anında işler
- Çok iyi bilinen bir örnek: C önişlemci
  - `#include, #define` komutları ve benzer makrolar

# Programlama Çevresi

- Yazılım geliştirilmesinde kullanılan araçların bir koleksiyonu
- UNIX
  - Eski bir işletim sistemi ve araç koleksiyonu
  - Günümüzde sıkça GUI ile birlikte kullanılır (CDE, KDE, veya GNOME) Bunlar UNIX'in üst seviyesinde çalışır
- Microsoft Visual Studio.NET
  - Büyük, kompleks görsel çevre
- Web uygulamalarını ve Web olmayan herhangi bir .NET dili kullanılır
- NetBeans
  - Java'daki web uygulamaları hariç olmak üzere, Visual Studio .NET ile ilgilidir

# Özet

- Programlama dillerini öğrenmek, birkaç nedenden ötürü önemlidir:
  - Farklı yapıları kullanmak için kapasitemizi artırır
  - Dilleri daha akıllıca seçmemize imkan tanır
  - Yeni dilleri öğrenmemizi kolaylaştırır
- Programlama dillerini değerlendirmek için daha önemli kriterler aşağıdakileri kapsar:
  - Okunabilirlik, Yazılabilirlik, Güvenilirlik, Maliyet
- Dil tasarımında asıl etkiler makine mimarisi ve yazılım geliştirme metodolojileridir
- Programlama dillerinin uygulamalarının asıl (temel) yöntemleri: derleme, sade yorumlama ve hibrid uygulamadır