BMB214 Programlama Dilleri Prensipleri

Ders 1. Giriş ve Ön Bilgiler

Erdinç Uzun
Tekirdağ Namık Kemal Üniversitesi
Bilgisayar Mühendisliği Bölümü

Ders için Önerilen Kaynaklar

- Ders Slaytları, Video ve Kodlar
  - adys.nku.edu.tr
  - erdincuzun.com
  - Youtube
- Tavsiye Edilen Kitaplar
  - **Programlama Dilleri Prensipleri**, Prof. Dr. Nejat YUMUŞAK, Dr. M. Fatih ADAK, Seçkin Yayıncılık
  - **Concepts of Programming Languages**, 12th edition, Robert W Sebesta, Pearson
- Değerlendirme
  - Ara Sınav: %30
  - Final: %70
  - Finalde ödevlerden 1 soru olacaktır.

Diğer Üniversiteler

- Boğaziçi, Hacettepe, ODTÜ, Ege, Sakarya, Düzce
- Stanford, MIT
  - CS242 website (stanford.edu)
  - Programming Languages | Electrical Engineering and Computer Science | MIT OpenCourseWare

Bu dersin açılmasının nedenleri

- Program geliştirme fikirleri ifade etme becerisini arttırma
- Projeye uygun programlama dillerini seçmek
- Yeni diller öğrenme becerisini attırma
- Uygulamanın öneminin daha iyi anlaşılması
- Zaten bilinen dillerin daha iyi kullanılması
- Gelişime ayak uydurma

Konular

- Derse Giriş
- Programlama Dillerinin Seviyesine Göre Sınıflandırılması
- Programlama Alanları
- Dil Değerlendirme Kriterleri
- Dil Tasarımına Etkiler
- Dil Kategorileri
- Dil Tasarım Değişimi Uygulama Yöntemleri
- Programlama Ortamları

Programlama Dillerinin Seviyesine Göre Sınıflandırılması

- Seviye, bir programlama dilinin insan algısına olan yakınlığı olarak ifade edilebilir.
  - Makine Dili: 1 ve 0
  - Düşük Seviyeli: Assembly
  - Orta Seviye: C, C++
  - Yüksek Seviye: Pascal, Basic, Fortran, Cobol
    - Çok Yüksek Seviye: Dbase, Visual Basic, Java, SQL, Paradox, Access, FileMaker, C#

Programlama Alanları

- Bilimsel ve mühendislik uygulamaları
  - Dizi ve matrislerin etkin kullanımı
  - Fortran, C, Pascal
- Mesleki uygulamalar
  - Raporlar oluşturun
  - COBOL
- Yapay zeka uygulamaları
  - Sayılar yerine semboller; bağlantılı listeleri(Linked List) kullanır
  - LISP, Prolog, Python
- Sistem programlama
  - Sürekli kullanım nedeniyle verimliliğe ihtiyaç var
  - C, Assembler
- Web Yazılımı
  - Farklı programlama dilleri: biçimlendirme (ör. HTML), İstemci taraflı programlama (Javascript), komut dosyası oluşturma (ör. PHP, Java, C#)

Programlama Dili Değerlendirme Kriterleri

- **Okunabilirlik (Readability)**: programların okunma ve anlaşılma kolaylığı
- **Yazılabilirlik (Writability)**: bir dilin programları oluşturmak için kullanılma kolaylığı
- **Güvenilirlik (Reliability)**: spesifikasyonları yerine getirme
- **Maliyet (Cost)**: nihai toplam maliyet

Programlama Dili Değerlendirme Kriterleri

	CRITERIA		
Characteristic	READABILITY	WRITABILITY	RELIABILITY
Simplicity	•	•	•
Orthogonality	•	•	•
Data types	•	•	•
Syntax design	•	•	•
Support for abstraction		•	•
Expressivity		•	•
Type checking			•
Exception handling			•
Restricted aliasing			•

Programlama Dili Değerlendirme Kriterleri : Okunabilirlik (Readability)

- Genel basitlik (Simplicity)
- Ortogonalite (Orthogonality)
- Veri tipleri (Data Types)
- Sözdizimi ile ilgili hususlar (Syntax Design)

Programlama Dili Değerlendirme Kriterleri : Okunabilirlik (Readability)

- Genel basitlik
  - Yönetilebilir bir dizi özellik ve yapı:
    - Temel Yapılar sahip dil (Array: C# ve Java)
  - Özellik Çokluğu (Feature Multiplicity) :

Java arttırma örneği

```
count = count + 1
count += 1
count++
++count
```

  - Minimal operatör aşırı yüklemesi: a + b
    - a ve b int ise toplama işlemi sonuç int
    - a ve b string ise birleştirme işlemi sonuç string
    - Programlama dilinine farklılıklar olabilir.

https://en.wikipedia.org/wiki/Operator_overloading

Programlama Dili Değerlendirme Kriterleri : Okunabilirlik (Readability)

- Ortogonalite
  - Ortogonalite, "A'yı değiştirmek B'yi değiştirmez" anlamına gelen özelliktir.
  - Bir programlama dilinde ortogonallik, nispeten küçük bir ilkel yapı kümesinin, dilin kontrol ve veri yapılarını oluşturmak için nispeten az sayıda yolla birleştirilebileceği anlamına gelir.
  - opcode [operand] [operand] …

https://en.wikipedia.org/wiki/Orthogonality_(programming)#:~:text=The%20term%20is%20most%2Dfrequently,data%20structures%20of%20the%20language.

Programlama Dili Değerlendirme Kriterleri : Okunabilirlik (Readability)

- Veri tipleri
  - Yeterli önceden tanımlanmış veri türleri
  - C'de Boolean veri tipi yoktur
    - timeout = 0
  - Java, C#, Python… gibi dillerde Boolean veri tipi vardır.
    - timeout = false

Programlama Dili Değerlendirme Kriterleri : Okunabilirlik (Readability)

- Sözdizimi ile ilgili hususlar
  - Özel Kelimeler
    - Döngü için özel kelimeler: while, class, and for
  - Birleşik özel kelimeler
    - static kelimesi (compile time'da yüklenir)
    - public static

Programlama Dili Değerlendirme Kriterleri
Yazılabilirlik (Writability)

- Basitlik ve Ortogonalite
  - Az sayıda yapı, az sayıda ilkel, bunları birleştirmek için küçük bir kurallar dizisi
- Soyutlama desteği
  - Karmaşık yapıları veya işlemleri ayrıntıların göz ardı edilmesine izin verecek şekilde tanımlama ve kullanma yeteneği (Soyutlama konusu programcı yeteneğidir.)
- İfade
  - İşlemleri belirlemenin nispeten uygun bir yolu
  - Operatörlerin gücü ve sayısı ve önceden tanımlanmış işlevler

Programlama Dili Değerlendirme Kriterleri
Güvenilirlik (Reliability)

- Tip kontrolü
  - Derleme zamanında yapılmalıdır (Çalışma zamanında kötü sonuçlar doğurabilir.)
  - Tip hatalarını: int, float, string
    - int a = 10.2; ?

Programlama Dili Değerlendirme Kriterleri
Güvenilirlik (Reliability)

- İstisna işleme (Exception Handling)
  - Çalışma zamanı hatalarını önleyin ve düzeltici önlemler alın.
    - try-catch
  - Ada, C++, Java, and C# bu konuya önem vermiştir.

Programlama Dili Değerlendirme Kriterleri
Güvenilirlik (Reliability)

- Takma ad (Aliasing)
  - Aynı bellek konumu için iki veya daha fazla farklı referans alma yönteminin varlığı
  - Özellikle Nesne Yönelimli Programlama dillerinde yapılabilir.

Programlama Dili Değerlendirme Kriterleri
Güvenilirlik (Reliability)

- Okunabilirlik ve yazılabilirlik
  - Bir algoritmayı ifade etmenin "doğal" yollarını desteklemeyen bir dil, "doğal olmayan" yaklaşımların kullanılmasını ve dolayısıyla daha düşük güvenilirlik gerektirir.
  - Bir program ne kadar kolay yazılırsa, doğru olma olasılığı da o kadar yüksektir.

Programlama Dili Değerlendirme Kriterleri
Cost (Maliyet)

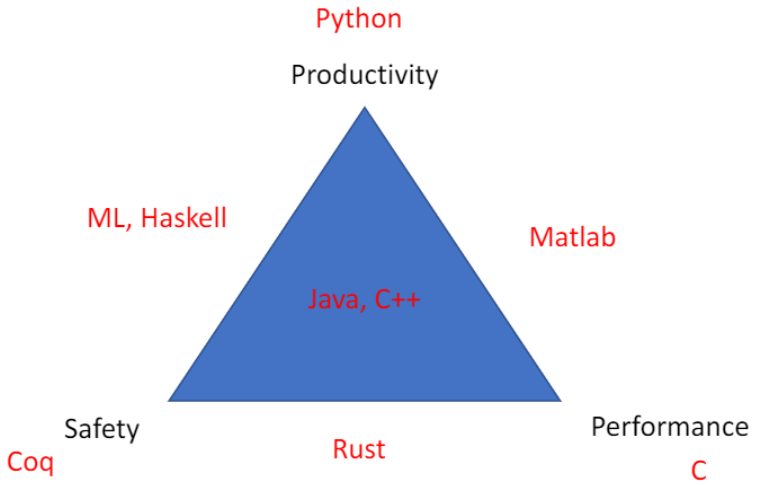
- Programcıları dili kullanmak için eğitmek
- Program yazma (belirli uygulamalara yakınlık)
- Programları derleme: Zaman önemli
- Programları çalıştırma
- Ücretsiz derleyicilerin kullanılabilirliği
- Güvenilirlik: zayıf güvenilirlik yüksek maliyetlere yol açar
- Programların bakımı

Programlama Dili Değerlendirme Kriterleri
Diğerleri

- **Taşınabilirlik (Portability)**
  - Programların bir uygulamadan diğerine taşınabilme kolaylığı
- **Genellik (Generality)**
  - Geniş bir uygulama yelpazesine uygulanabilirlik
- **İyi tanımlılık (Well-definedness)**
  - Dilin resmi tanımının tamlığı ve kesinliği

Programlama Dilinin Hedefleri

- Performans
- Üretkenlik (Productivity)
- Güven (Safety)



https://web.stanford.edu/class/cs242/materials.html

Programlama Dili Tasarımı Etkileri

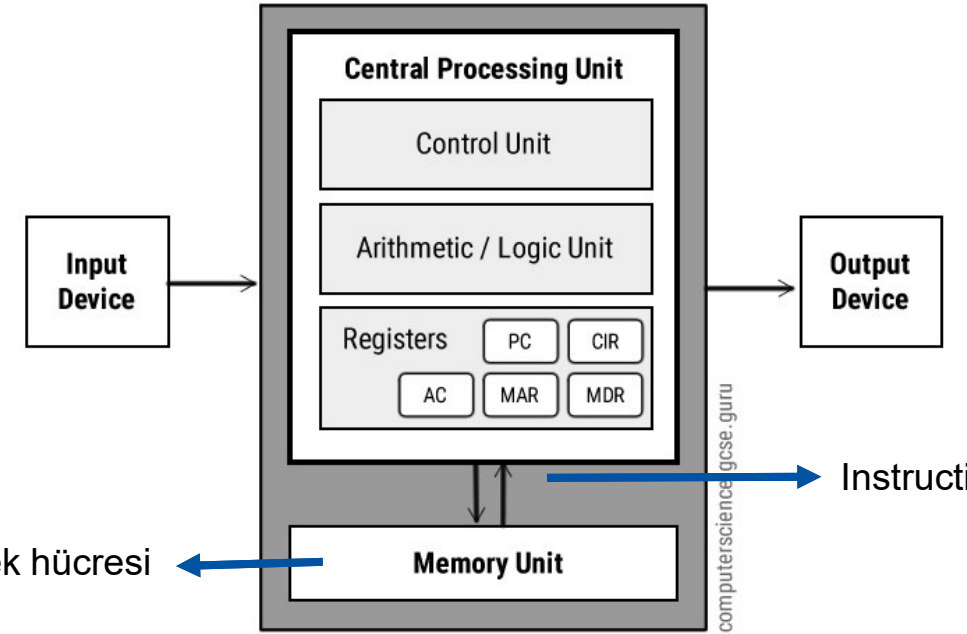
- Bilgisayar Mimarisi Diller:
  - von Neumann mimarisi olarak bilinen yaygın bilgisayar mimarisi olarak kullanılır.
- Program Tasarım Metodolojileri
  - Yeni yazılım geliştirme metodolojileri (ör. Nesne yönelimli yazılım geliştirme) yeni programlama paradigmalarına ve dolayısıyla yeni programlama dillerine yol açtı

Bilgisayar Mimarisi Etkisi

- Tanınmış bilgisayar mimarisi: Von Neumann
  - Bellekte (Memory) saklanan veriler ve programlar
  - Bellek CPU'dan ayrıdır
  - Komutlar (Instruction) ve veriler bellekten CPU'ya aktarılır
  - Programlama dillerin açısından
    - Değişkenler (variables) bellektedir.
    - Atamalarda, aktarım olur.
    - Iterasyon (Iteration) etkilidir

Von Neumann Mimarisi

von Neumann bilgisayarı, hem veriler hem de programlar aynı bellekte saklanır.



Instructions and data

Variables (Değişken): bellek hücresi

Von Neumann Mimarisi

- Fetch-execute-cycle

Program sayacını (Program Counter) başlat

sonsuza kadar tekrar et

sayacın işaret ettiği komutu (instruction) getir

sayacı artır (increment)

komutu çözmek (decode)

komutu çalıştır (execute)

tekrar sonu

Programlama Metodolojilerinin Etkileri

- 1950'ler ve 1960'ların başı: Basit uygulamalar; makine verimliliği konusu sorun
- 1960'ların sonu: İnsan verimliliği önemli hale geldi; okunabilirlik, daha iyi kontrol yapıları
  - yapısal programlama
  - yukarıdan aşağıya (top-down) tasarım ve adım adım iyileştirme
  - Bu dönemde, donanım maliyetleri düştükçe ve programcı maliyetleri arttıkça, temel bilgi işlem maliyetinin donanımdan yazılıma kaydı.
- 1970'lerin sonu: Veriye yönelik süreç odaklı
  - Veri soyutlama (data abstraction)
- 1980'lerin ortası: Nesne yönelimli programlama
  - Veri soyutlama + kalıtım (inheritance) + polimorfizm

Programlama Dili Kategorileri

- Imperative (Emir Esaslı) Programlama
  - Merkezi özellikler değişkenler, atama ifadeleri ve iterasyondur.
  - Nesne yönelimli programlamayı destekleyen diller
  - Betik dilleri (Scripting languages)
  - Görsel dilleri
  - Örnekler: C, Java, Perl, JavaScript, Ruby, Rust, Visual BASIC .NET, C ++, C#
- Fonksiyonel (Functional) Programlama
  - Hesaplama yapmanın ana yolu, verilen parametrelere fonksiyon uygulamaktır.
  - Örnekler: LISP, Scheme, ML, F#, Haskell
- Mantık (Logic) Programlama
  - Kural tabanlı
  - Örnek: Prolog
- Biçimlendirme / programlama hibrit
  - Bazı programları desteklemek için genişletilen biçimlendirme dilleri
  - Örnekler: JSTL, XSLT

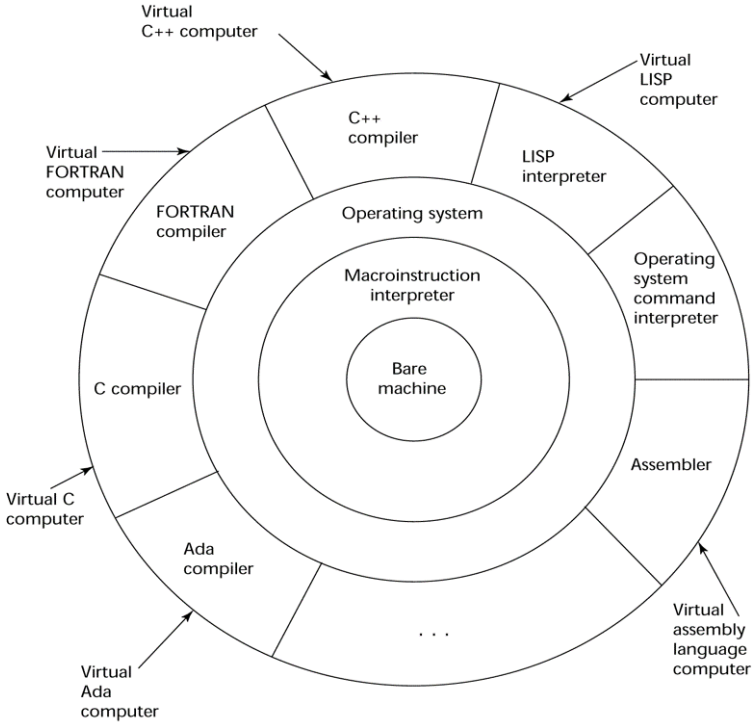
Dil tasarımda karşılaştırmalar

- Güvenilirlik vs. yürütme maliyeti
  - Örnek: Java, dizi öğelerine yapılan tüm referansların düzgün indeksleme için kontrol edilmesini ister, bu da yürütme maliyetlerinin artmasına neden olur
- Okunabilirlik vs. yazılabilirlik
  - Örnek: APL, karmaşık hesaplamaların kompakt bir programda ancak okunabilirliğin düşük olması pahasına yazılmasına olanak tanıyan birçok güçlü operatör (ve çok sayıda yeni sembol) sağlar
- Yazılabilirlik (esneklik) vs. güvenilirlik
  - Örnek: C ++ işaretçileri güçlü ve çok esnektir ancak güvenilmezdir

Uygulama Metotları

- Derleme (Compilation)
  - Programlar, Makine diline çevrilir; JIT (Just-in-Time) sistemlerini içerir
  - Kullanım: Büyük ticari uygulamalar
- Yorumlama (Interpretation)
  - Programlar, yorumlayıcı (interpreter) olarak bilinen başka bir program tarafından yorumlanır
  - Kullanım: Küçük programlar veya verimlilik sorun olmadığında
- Hibrit Uygulama Sistemleri
  - Derleyiciler ve yorumlayıcılar arasında bir uzlaşma
  - Kullanım: Verimlilik ilk konu olmadığında küçük ve orta ölçekli sistemler

Katmanlı Görünüm: İşletim sistemi ve Programlama dilleri



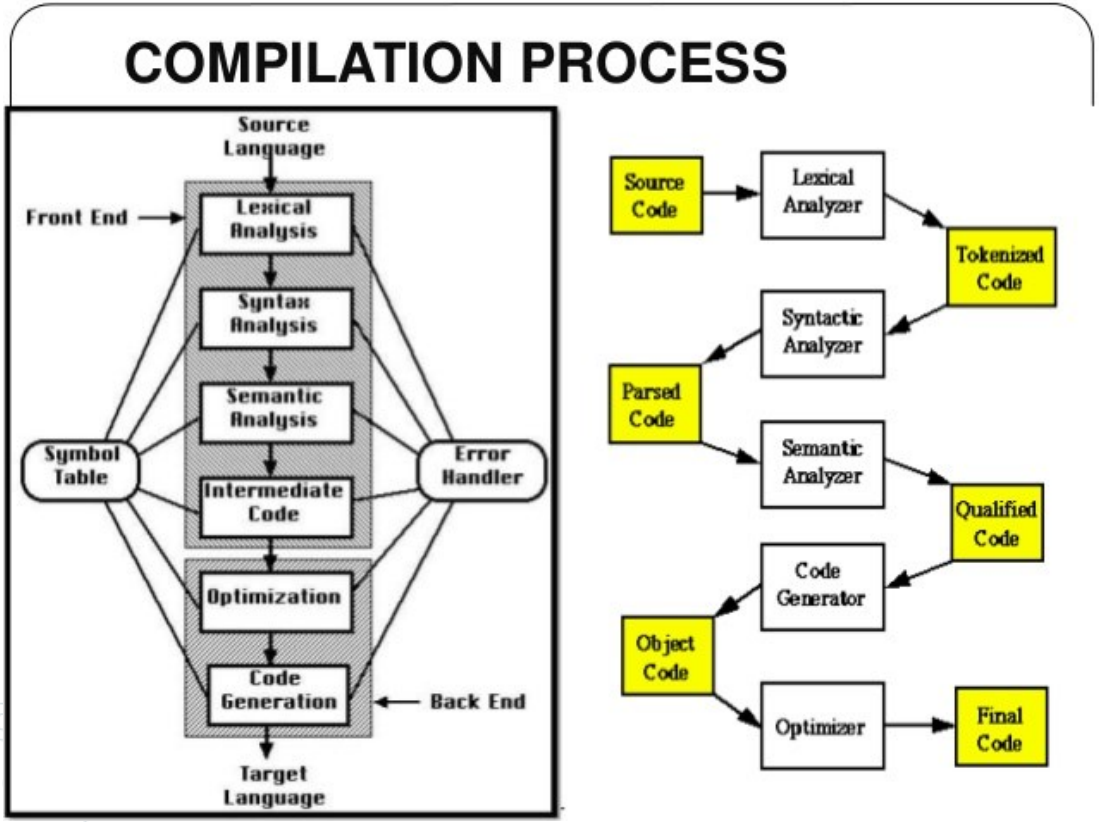
Derleme (Compilation)

- Üst düzey programı (kaynak dil – source language) makine koduna (makine dili) çevirir
- Yavaş çeviri, hızlı uygulama
- Derleme işleminin birkaç aşaması vardır:
  - Sözcük analizi (lexical anaysis): kaynak programdaki karakterleri sözcük birimlerine dönüştürür
  - Sözdizimi (syntax) analizi: sözcük birimlerini programın sözdizimsel yapısını temsil eden ayrıştırma ağaçlarına dönüştürür
  - Anlambilim (semantics) analizi: ara kod oluştur
  - Kod üretimi (code generation): makine kodu üretilir

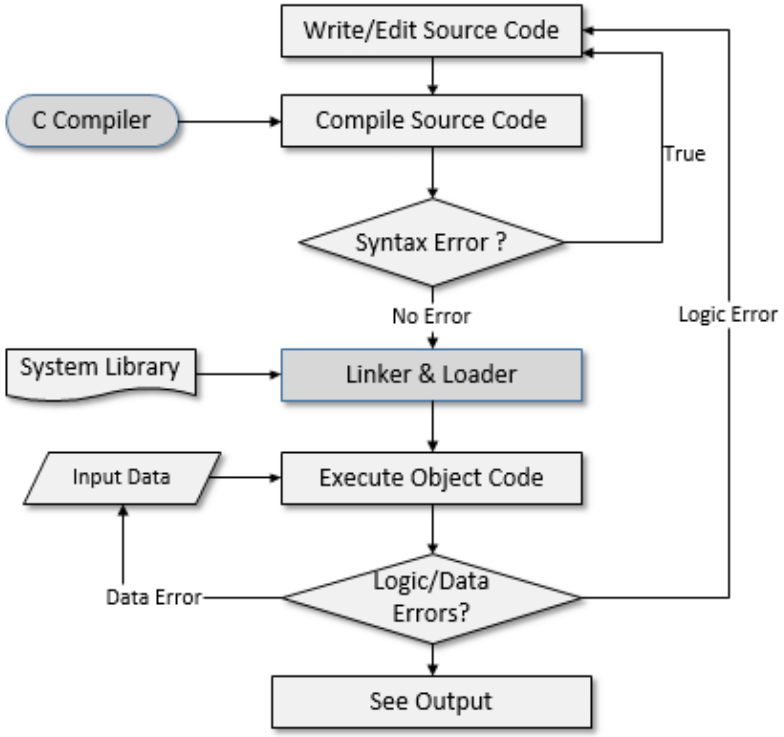
Derleme (Compilation)

- Yükleme modülü (Load module - executable image) : kullanıcı ve sistem kodu birlikte bulunduğu modül.
- Bağlama ve yükleme (Linking and loading): sistem program birimlerini toplama ve bunları bir kullanıcı programına bağlama süreci

Derleme İşlemi (Compilation Process)



Derleme İşlemi (Compilation Process)



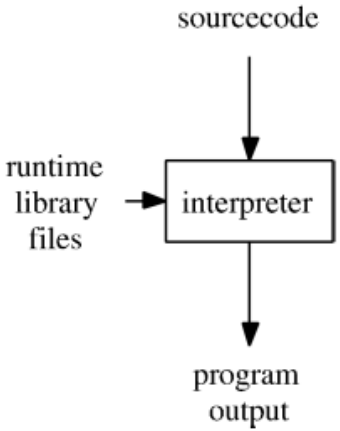
Von Neumann Darboğazı

- Bir bilgisayarın belleği ile işlemcisi arasındaki bağlantı hızı, bilgisayarın hızını belirler
- Program komutları genellikle bağlantı hızından çok daha hızlı yürütülebilir; bağlantı hızı böylece bir darboğazla sonuçlanır
- Von Neumann darboğazı olarak bilinir; bilgisayarların hızındaki birincil sınırlayıcı faktördür

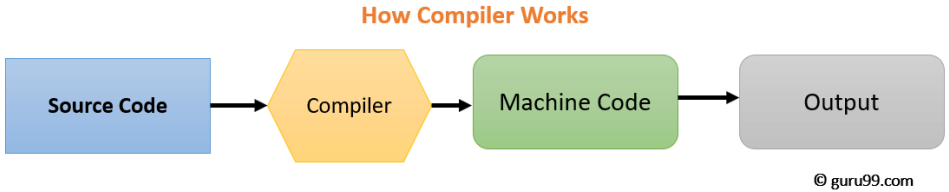
Yorumlayıcı

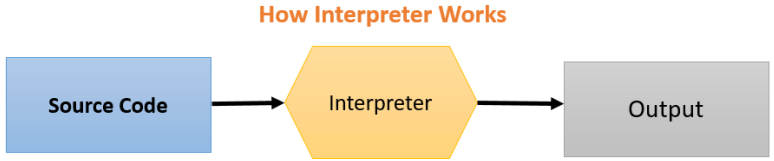
- Kaynak kodunuz, bir yorumlayıcı program tarafından yorumlanır.
- Programların daha kolay uygulanması (çalışma zamanı hataları kolayca ve anında görüntülenebilir)
- Daha yavaş yürütme (derlenmiş programlardan 10 ila 100 kat daha yavaş)
- Genellikle daha fazla alan gerektirir
- Artık geleneksel yüksek seviyeli diller için çok nadir örnekleri vardır.
  - Bazı Web kodlama dilleriyle (ör. JavaScript, PHP) önemli geri dönüş

Yorumlayıcı Çalışması



Derleyici vs. Yorumlayıcı

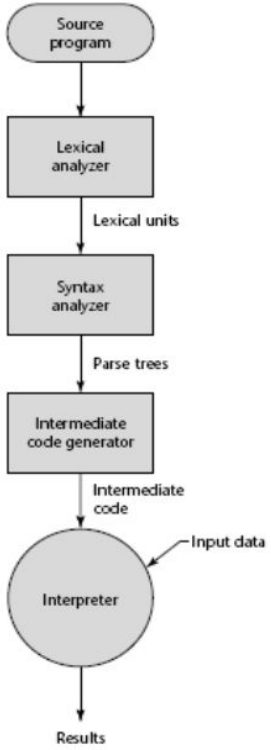




Hibrit Uygulama Sistemleri

- Derleyiciler ve yorumlayıcı arasında
- Üst düzey (high-level) bir dil programı, kolay yorumlamaya izin veren bir ara dile çevrilir
- Yorumlamadan daha hızlı
- Örnekler
  - Perl programları, yorumlamadan önce hataları tespit etmek için kısmen derlenir
  - Java'nın ilk uygulamaları hibritti; ara kod (bayt kodu), bayt kodu yorumlayıcısı ve çalışma zamanı sistemine sahip herhangi bir makineye taşınabilirlik sağlar (bunlara birlikte Java Sanal Makinesi (Java Virtual Machine - JVM) denir)

Hibrit Uygulama Sistemlerinin Çalışması



JIT (Just-in-Time) Uygulama Sistemleri

- Başlangıçta programları bir ara dile çevirin
- Ardından, alt programların ara dilini çağrıldıklarında makine koduna derler.
- Makine kodu sürümü sonraki çağrılar için saklanır
- JIT sistemleri Java programları için yaygın olarak kullanılmaktadır
- .NET dilleri bir JIT sistemi ile uygulanmaktadır
- Özünde, JIT sistemleri gecikmeli derleyicilerdir

Ön işlemciler (Preprocessors)

- Ön işlemci makroları (komutlar), başka bir dosyadaki kodun ekleneceğini belirtmek için yaygın olarak kullanılır.
- Bir önişlemci, gömülü önişlemci makrolarını genişletmek için program derlenmeden hemen önce bir programı işler.
- Örnek: C ön işlemcisi #include, #define
  - #include "myLib.h "
  - #define max(A, B) ((A) > (B) ? (A) : (B))
    - x = max(2 * y, z / 1.73);
    - x = ((2 * y) > (z / 1.73) ? (2 * y) : (z / 1.73); //ön işleme

Programlama Ortamları

- Derleyici ve yorumlayıcı olmalı
  - C, C++, ADA, Fortran: gcc.gnu.org
  - C# ve F#: .Net Framework, .Net Core, .Net 5.0
  - Java: java.sun.com
  - Haskell: haskell.org
  - Scheme: https://www.scheme.com/
  - Perl: www.perl.com
  - Python: www.python.com
  - Ruby: https://www.ruby-lang.org/
  - Node.js: https://nodejs.org/en/
  - Rust: https://www.rust-lang.org/
  - Javasript: Bir browser ☺

IDE (Integrated Development Environment - Tümleşik geliştirme ortamı)

- Bilgisayar programcılarının hızlı ve rahat bir şekilde yazılım geliştirebilmesini amaçlayan, geliştirme sürecini organize edebilen birçok araç ile birlikte geliştirme sürecinin verimli kullanılmasına katkıda bulunan araçların tamamını içerisinde barındıran bir yazılım türüdür.
  - Programlama diline göre sözdizimi renklendirmesi yapabilen kod yazım editörü.
  - Kod dosyalarının hiyerarşik olarak görülebilmesi amacıyla hazırlanmış gerçek zamanlı bir dizelge.
  - Tümleşik bir derleyici, yorumlayıcı ve hata ayıklayıcı.
  - Yazılımın derlenmesi, bağlanması, çalışmaya tümüyle hazır hale gelmesi ve daha birçok ek işi otomatik olarak yapabilmek amacıyla küçük inşa araçları.

https://tr.wikipedia.org/wiki/T%C3%BCmle%C5%9Fik_geli%C5%9Ftirme_ortam%C4%B1

IDE (Integrated Development Environment - Tümleşik geliştirme ortamı)

- Borland JBuilder
- Microsoft Visual Studio .NET
- NetBeans
- Eclipse
- Code::Blocks
- Dev-C++
- Anjuta
- KDevelop
- **Visual Studio Code**
  - **Extension desteği sayesinde birçok programlama diline desteği var. Açık Kaynak Kod.**

Ortam hazırlanıyor...

- C, C++, ADA, Fortran Derleyicilerinin kurulumu
- VS Code ile bağlantısı