Bilgisayar Mühendisliğine Giriş

Yrd.Doç.Dr.Hacer KARACAN

PROGRAMLAMA DİLLERİ

- "Programlama Dilleri" ni neden öğrenmeliyiz?
- Programlama alanları
- Dil değerlendirme kriterleri
- Dil tasarımı üstündeki etkiler
- Dil kategorileri
- Gerçekleştirim metodları
- Programlama ortamları

“Programlama Dilleri” ni neden öğrenmeliyiz?

- Fikirlerimizi ifade etmemizi kolaylaştırır.
- Uygun dil seçmek için konuyla ilgili altyapımızı sağlamlaştırır.
- Yeni dil öğrenme becerimiz artar.
- Gerçekleştirimin (implementation) önemini kavramamızı sağlar.
- Bilgisayar programlama çalışmalarımızda çok daha başarılı olmamızı sağlar.

Programlama Alanları (Programming Domains)

- Bilimsel uygulamalar
  - Çok sayıda kayan nokta (floating-point) hesaplamaları
  - Fortran
- İş yaşamı uygulamaları
  - Rapor oluşturma
  - COBOL
- Yapay zeka
  - Sayılar yerine sembollerle yapılan hesaplamalar
  - LISP
- Sistem programlama
  - Sürekli kullanımdan ötürü hızlı bir sistem yaratılması
  - C
- Web yazılımları
  - Birçok dilin derlenmiş koleksiyonu: markup (örn. XHTML), scripting (örn. PHP), genel amaçlı (örn. Java)

Dil Değerlendirme Kriterleri (Language Evaluation Criteria)

- **Okunabilirlik (Readability)**: O dille yazılan bir programın okunmasının kolay ve anlaşılırlığının yüksek olması gereklidir.
- **Yazılabilirlik (Writability)**: Program yaratmak için dilin kolaylıkla kullanılabiliyor olması gereklidir.
- **Güvenilirlik (Reliability)**: Dilin beklentilere ve standartlara uygun olması gereklidir.
- **Maliyet (Cost)**: Faydası maliyetinden yüksek olmalıdır.

Değerlendirme Kriterleri: Okunabilirlik

- "Overall Simplicity"
  - Yönetimi kolay özellikler/yapılar bulunması
  - Aynı operasyonlar için farklı farklı özelliklerin/yapıların bulunmaması
  - Operatör yüklemelerinin en az seviyede olması
- "Orthogonality"
  - Az sayıda ilkel/basit yapıların yine az sayıda yöntem kullanılarak birleştirilmesi
  - Her kombinasyonun doğru (legal) olması
- "Control statements"
  - İyi organize edilmiş kontrol deyimlerinin bulunması (örn., while deyimi)
- "Data types and structures"
  - Veri yapılarının iyi tanımlanmış olması
- "Syntax considerations"
  - Karmaşık ifadelerin oluşturulması için belirli özel deyimlerin kullanılması
  - Anlamlı anahtar sözcükler ve anlaşılabilir yapılar kullanılması

Değerlendirme Kriterleri: Yazılabilirlik

- "Simplicity and orthogonality"
  - Az sayıda ilkel/basit yapıların yine az sayıda kural kullanılarak birleştirilmesi
- "Support for abstraction"
  - Karmaşık yapıların ya da operasyonların ,gereksiz detayları göz ardı etmeye olanak sağlayacak şekilde, tanımlanabilmesi ve kullanılabilmesi
- "Expressivity"
  - Operasyonları ifade etmek için uygun deyimler bulunması
  - Örn: for deyiminin tüm modern dillere eklenmesi

Değerlendirme Kriterleri: Güvenilirlik

- "Type checking"
  - Yazım hatalarına karşı testleri yapılmış olmalı
- "Exception handling"
  - Yürütme süresi (run-time) hataları önlenmeli, önlenemediği durumlarda ise program tarafından düzeltilip çalışmaya devam edilmeli
- "Aliasing"
  - Aynı hafıza lokasyonu için iki veya daha fazla sayıda ayrı referans metodunun bulunması
- "Readability and writability"
  - Bir programı yazmak ve kodunu anlamak ne kadar kolaysa, doğru olarak çalışma şansı da o kadar yüksek olacaktır.

Değerlendirme Kriterleri: Maliyet

- Maliyet için gözönüne alınması gerekenler:
  - Dili kullanabilmeleri için programcıların eğitilmesi
  - Programların yazılması
  - Programların derlenmesi (compile)
  - Programların çalıştırılması (execute)
  - Programın hayata geçirilmesi: ücretsiz derleyicilerin (compiler) bulunması
  - Güvenilirlik →düşük güvenilirlik yüksek maliyet demektir
  - Program güncellemelerinin ve olası değişikliklerin uygulanması

Diğer Değerlendirme Kriterleri

- "Portability"
  - Programların bir uygulamadan diğerine geçişlerinin kolay olması
- "Generality"
  - Çok sayıda uygulama alanı olması
- "Well-definedness"
  - Dilin bütünlük ve kesinlik içeren bir tanımlamasının bulunması

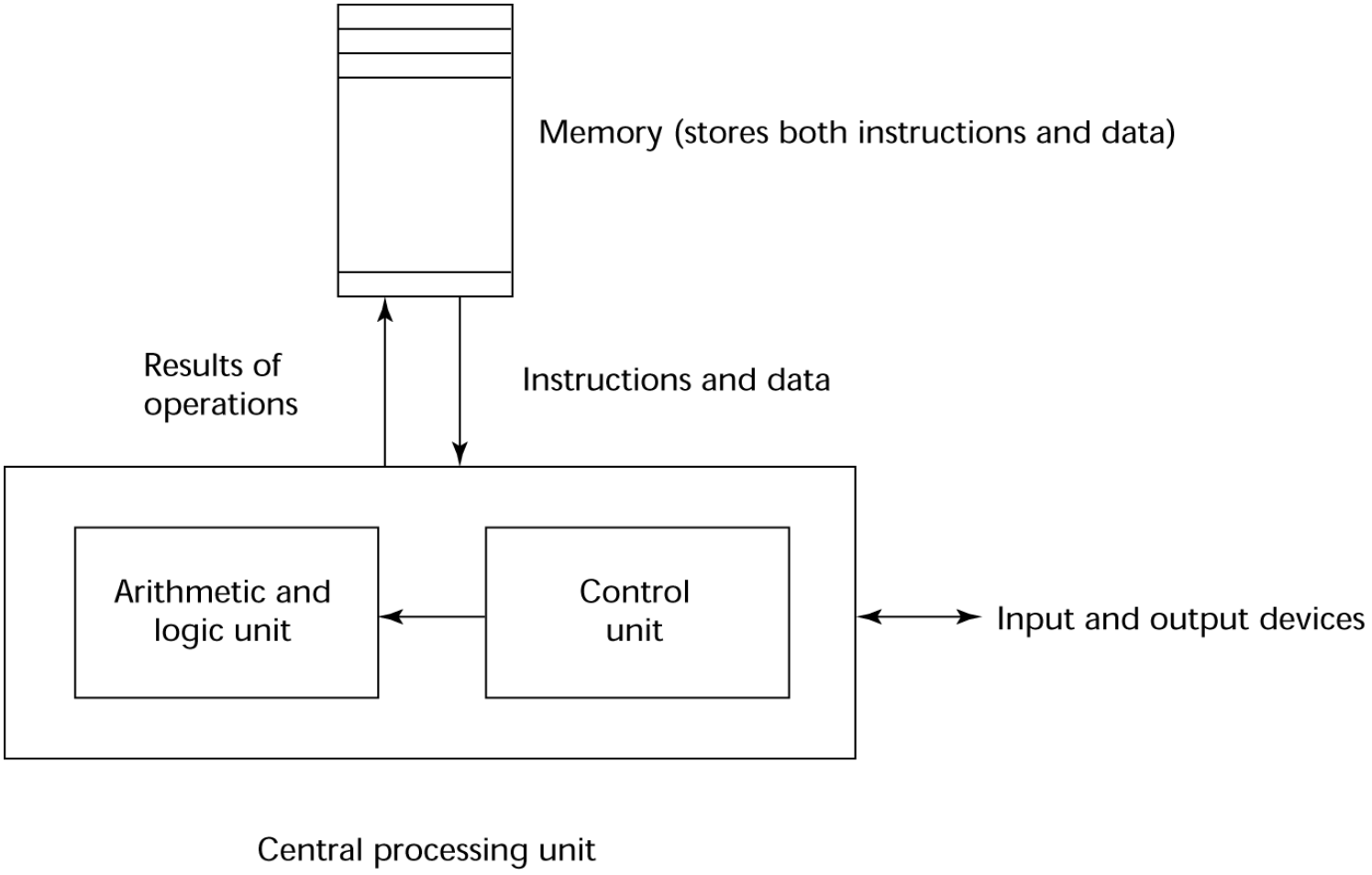
Dil Tasarımı Üzerindeki Etkiler

- Bilgisayar Mimarisi
  - Programlama dilleri yaygın bilgisayar mimarisine (*von Neumann*) uygun olarak geliştirilir
- Programlama Metodolojileri
  - Yeni yazılım geliştirme metodolojileri (örn. nesne yönelimli yazılım geliştirme) yeni programlama metodlarının ve dolayısıyla yeni programlama dillerinin ortaya çıkmasını sağlar

Bilgisayar Mimarisinin Etkileri

- En bilinen bilgisayar mimarisi: Von Neumann
  - Veri ve programlar hafızada saklanır
  - Hafıza CPU'dan ayrıdır
  - Komutlar ve veriler hafızadan CPU'ya aktarılır
  - Bu mimariye uygun baskın dillerin özellikleri
    - Değişkenler hafıza hücrelerinin birer modelidir
    - Atama ifadeleri (assignment statements) veri aktarımının birer modelidir
    - İterasyonlar verimlidir

von Neumann Mimarisi



Von Neumann Bottleneck

- Bir bilgisayarın hafızasıyla işlemcisi arasındaki bağlantı hızı bilgisayarın hızını belirler
- Program yönergeleri genellikle bu bağlantı hızından çok daha hızlı bir şekilde çalıştırılır ve bu sebeple bilgisayarın hızı bir darboğaz (bottleneck) haline gelir
- Bu durum von Neumann darboğazı olarak adlandırılır ve bilgisayarın hızıyla ilgili temel sınırlama faktörüdür

Programlama Metodolojilerinin Etkileri

- 1950'ler ve 1960'ların başı: Basit uygulamalar; Makine verimliliğine yönelik kaygılar
- 1960'ların sonu: İnsan verimliliği; okunabilirlik ve daha iyi kontrol yapıları önem kazandı
  - Yapısal (structured) programming
  - Yukardan-aşağıya (top-down) tasarım ve adım-adım düzenleme
- 1970'lerin sonu: Süreç-tabanlı tasarımlardan veri-tabanlı tasarımlara geçiş
  - Veri soyutlama (data abstraction)
- 1980'lerin ortaları: Nesne yönelimli programlama
  - Veri soyutlama + kalıtım + polimorfizm

Programlama Dili Kategorileri

- "Imperative"
  - Merkez yapıyı değişkenler, atama ifadeleri ve iterasyon oluşturur
  - Örn: C, Pascal
- "Functional"
  - Verilen parametrelere fonksiyonları uygulayarak hesaplamalar yapma temeline dayanır
  - Örn.: LISP, Scheme
- "Logic"
  - Kural-tabanlı (Rule-based)
  - Örn: Prolog
- "Object-oriented"
  - Veri soyutlama, kalıtım
  - Örn: Java, C++
- Markup
  - Programlama dili değil, dokumanların genel gorunumlerini düzenlemek amacıyla kullanılır
  - Örn: XHTML, XML

Programlama Dilleri Tasarımı

- Güvenilirlik **X** Yürütme giderleri
  - Örn: Java tüm dizi elemanlarının referanslarının düzgün indexleme için kontrol edilmesini talep eder ama bu yürütme giderlerini arttıran bir durumdur

- Okunabilirlik **X** Yazılabilirlik
  - Örn: APL karmaşık programların yazılabilirliği arttırmak için birçok güçlü operatör ve çok sayıda yeni sembol sunar ancak bu durum programların okunabilirliğini ciddi şekilde düşürür

- Yazılabilirlik **X** Güvenilirlik
  - Örn: C++ işaretçileri oldukça güçlü ve esnek bir yapıdadır ancak güvenilir bir biçimde kullanılmaları zordur

Gerçekleştirim Metodları

- Derleme (Compilation)
  - Programlar makine diline çevrilir
- Yorumlama (Interpretation)
  - Programlar bir yorumlayıcı (interpreter) vasıtasıyla yorumlanır
- Hibrid gerçekleştirim sistemleri
  - Derleyiciler ve yorumlayıcıların birlikte kullanımı

Derleme (Compilation)

- Üst-seviye programı (kaynak dili) makine koduna (makine dili) çevirir
- Çevirim yavaş, yürütme hızlıdır
- Derleme işleminin birçok fazı bulunmaktadır:
  - **Sözcüksel (lexical) analiz:** kaynak program içindeki karakterleri sözcüksel birimlere dönüştürür
  - **Sözdizim (syntax) analizi:** sözcüksel birimleri programın sözdizimsel yapısını ifade eden ayrıştırma ağaçlarına (parse tree) dönüştürür
  - **Anlambilim (semantics) analizi:** orta seviye program kodunu üretir
  - **Kod üretme:** makine kodu üretilir

Yorumlama (Interpretation)

- Çeviri yoktur
- Programların gerçekleştirimini kolaylaştırır (yürütüm aşaması hataları kolaylıkla ve vakit kaybetmeden görüntülenir)
- Daha yavaş çalışır (derlenen programlardan 10 ila 100 kat daha yavaştır)
- Genellikle geniş hafıza alanına ihtiyaç duyar
- Üst-seviye dillerde kullanımı yaygın değildir
- Bazı "web scripting" dilleriyle (örn. JavaScript) ciddi bir geridönüş yapmıştır

Hibrit Gerçekleştirim Sistemleri

- Derleyiciler ve yorumlayıcıların birlikte kullanımı

- Yüksek-seviyeli bir dille yazılan program kolayca yorumlanmasını sağlayacak orta-seviye bir dile çevrilir

- Tek başına yorumlama sürecinin işlediği durumlardan daha hızlıdır

- Örnek:
  - Perl programları hataları yakalayabilmek için yorumlanmadan önce kısmen derlenir

  - Java dilinin ilk uygulamaları hibrit bir gerçekleştirim sistemi kullanıyordu; orta-seviye form (*byte kod*) byte kod yorumlayıcısı ve run-time sistemi olan her makinede (*Java Virtual Machine*) programın çalışmasını sağlıyordu.

Just-in-Time Gerçekleştirim Sistemleri

- Başlangıçta programları orta-seviye bir dile çevirir
- Daha sonra bu dili makine koduna derler (compilation)
- Makine kodu versiyonu daha sonraki çağrılar için saklanır
- JIT sistemleri Java programları için sıklıkla kullanılır
- .NET dilleri de JIT sistemleriyle gerçekleştirilir

Programlama Ortamları

- Yazılım geliştirmek için kullanılan araçların oluşturduğu bir koleksiyon olarak tanımlanabilir.
- UNIX
  - Eski bir işletim sistemi ve araç koleksiyonu
  - Günümüzde genellikle UNIX üzerinde çalışan bir GUI (e.g., CDE, KDE, or GNOME) ile kullanılır
- Borland JBuilder
  - Java dili için geliştirme ortamıdır
- Microsoft Visual Studio.NET
  - Geniş, karmaşık bir görsel ortamdır
  - C#, Visual BASIC.NET, Jscript, J#, ve C++ dilleriyle yazılan programlarda kullanılır

Özet

- Programlama dillerini öğrenmek birçok açıdan yararlıdır
  - Farklı yapıları kullanmamızı kolaylaştırır
  - Uygun programlama dilini kolaylıkla seçmemize yardımcı olur
  - Yeni bir programlama dili öğrenmemizi kolaylaştırır
- Bir programlama dilini değerlendirmek için en önemli kriterler:
  - Okunabilirlik, yazılabilirlik, güvenilirlik, maliyet
- Programlama dilleri üzerindeki temel etkileri makine mimarisi ve yazılım geliştirme metodolojileri yapmaktadır
- Programlama dillerini gerçekleştirmek için ana metodlar şunlardır:
  - Derleme
  - Yorumlama
  - Hibrit gerçekleştirim sistemleri