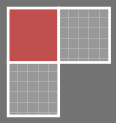
2010

# Programlama Dilleri

## Ders Notları

Bu doküman 2009-2010 yıllarında Gebze Yüksek Teknoloji Enstitüsü Bilgisayar Mühendisliği bölümünde verilen Programlama Dilleri dersi notlarından derlenmiştir. Ayrıca, ders notlarının derlenmeside özellikle Robert W. Sebesta'ya ait; Concepts of Programming Languages isimli eserden faydalanılmıştır.

HIDAYETTAKCI
GYTE Bilgisayar Mühendisliği
31.08.2010



# 1. Birinci Bölüm : Programlama Dillerine Giriş

**Bu bölümde**

- **Neden programlama dilleri dersi?**
- **Programlama dili nedir?**
- **Programlama dillerinde tercih edilen özellikler**
- **Programlama dilleri taksonomisi**
- **Programlama dillerinin uygulama alanlarına göre sınıflandırılması**
- **Programlama dilleri değerlendirme kriterleri**

**Konuları işlenecektir.**

## 1.1. Programlama dili nedir?

En kaba ifadeyle algoritmaları gerçekleştirmek için kullanılan simgeler dizisine program adı verilir. Programlama dili ise programlar oluşturmak için kullanılan yazılım ortamlarıdır. Programlama dilleri hem kaynak kodun yazılması hem de kaynak kodun çalışabilir koda çevrilmesinde çatı vazifesi görmektedir. Kaynak kodun çalışabilir koda çevrimi programlama dillerinin sahip olduğu dil işlemcileri sayesinde yerine getirilir. İki tip dil işlemci vardır: derleyici ve yorumlayıcı. Programlama dilleri işletim sistemi değildir fakat işletim sistemi için lüzumlu uygulamalar üretmeye yarar.

### 1.1.1. Derleyicilere karşi yorumlayicilar

(Compilers vs. Interpreters)

Kaynak kodun (source code) çalıştırılabilir koda (executable code) çevirimi dil işlemcileri (language processors) tarafından yerine getirilir. İki tip dil işlemcisi vardır: derleyici (compiler) ve yorumlayıcı (interpreter). İkisi arasında çeşitli farklılıklar vardır. Örneğin çeviri konusunda;

Programlama dillerinin bazısı derleyici kullanır (örneğin pascal, c ve c++) bazısı ise yorumlayıcı (örneğin basic). Bu arada hem derleme hem de yorumlama yapan dillerde vardır. Bunlara birleşik uygulama (örneğin Java) adı verilir.

### 1.1.2. Neden yorumlayici?

- Esneklik (yürütüm zamani oluşan geç bağlantilar nedeniyle) – kodlar yorumlandıkça bellek bağlamalar yapılır
- Yürütüm zamani durum desteği – hatalar tek tek, sırayla düzeltilir
- Komut dosyalari (Perl, Shells, Python,TCL) için uygundur
- Dinamik ortamlar (Basic, APL, LISP) için uygundur
- Sanal makineler (JVM, Emulators, CPUs) için uygundur

### 1.1.3. Neden derleyici?

- Temel mühendislik prensipleri – bütüncül yaklaşım
- Doğruluk – erken statik hata kontrolü
- Maliyet – derleme, program maliyetini düşürür.
- Performans – hizli çalişir
    - Bir defa derle (maliyet), birçok defa yürüt (fayda)

## Derleme

Derleme, yüksek seviyeli bir dilde yazılan ve insanlar tarafından kolayca anlaşılan kodların, bilgisayarlar tarafından anlaşılan makine diline çevrimidir. Bir dönüşümü ifade eder. Bu dönüşüm sırasında ne kadar fazla işlem yapılırsa derleyici performansı o oranda düşecektir.

### Sözdizim analiz (Syntax Analysis)

Doğru kodlama yapılıp yapılmadığı lexical olarak kontrol edilir.

. Tarama: program tokenlara ayrılır

. Ayriştirma: tokenlar analiz edilir, doğru yazılıp yazılmadığı

**Anlambilimsel (semantic) analiz**
Mantıksal hata yapılıp yapılmadığı kontrol edilir. Örneğin sonsuz döngü durumları bu aşamada ortaya çıkarılır.
. Anlambilimsel analiz son kisimdir.
. Özet sözdizim analiz ağaci kullanilir.

## 1.2. Neden programlama dilleri dersi?

Fikirlerimizi somutlaştırma ihtiyacı
Algoritmik süreç önemli bir konudur fakat aynı zamanda soyut bir süreci ifade eder. Bir algoritmanın uygulanabilmesi için bir programlama dili ile kodlanması gerekir. Soyut şeylerin somutlaşması, fikirlerin uygulamaya dönüşebilmesi için programlama dilleri gereklidir.

Seçeneklerimizin ne olduğunu görebilmek için,
Bir fikrin uygulamaya dönüşmesinde birçok seçenek mevcuttur. Fikrimizi gerçekleştirmede kullanabileceğimiz seçeneklerin neler olduğunu ancak programlama dilleri bilgisi ile görebiliriz.

Dil öğrenmede yetkinlik
Dillerin özelliklerini bilmeyen, belli bir dille çalişmaya alişmiş kişi, farkli bir dili öğrenmesi gerektiğinde zorlanir. Fakat programlama dillerinin kavramlarına aşina olan birisi bir dilden başkasına daha kolay geçiş yapabilir.

Belli bir dilin önemli özelliklerini anlayarak daha iyi kullanabilmek için
Hangi dilin hangi alanlarda üstün olduğuna karar verebilmek için programlama dillerinin kavramlarına hakim olmak lazımdır. Örneğin ticari uygulamalar için Delphi, mühendislik uygulamalar için C++ dillerinin daha uygun olduğunu söyleyebilmek için dil özelliklerini iyi bilmek lazımdır.

Hesaplamanin gelişmesi için
Dilleri daha iyi değerlendirebilirsek, doğru seçimler yapariz, doğru teknolojilerin gelişmesine destek olmuş oluruz.

Hata ayiklarken özelliklerini bilmemiz faydalidir.
Her dilde hata durumları ile mücadele yöntemi farklıdır. Eğer biz dilin hata ayıklama konusundaki özellikleri iyi bilirsek o zaman hata bulmamız ve daha hızlı uygulama geliştirmemiz daha kolay hale gelir.

Özellikleri öğreniriz, olmayan özellikler için emulasyonlar yaparız.
Örneğin dilimiz hata ayıklama özelliğini sağlamıyorsa o özelliği biz programlarımızda kendimiz tamamlarız.

## 1.3. Programlama dillerinde tercih edilen özellikler:

**Teknik özellikler**
- Kolay kullanim
- Problem tipine uygunluk

- Yazma kolayliği
- Performansın yüksek olması
- Esneklik/Gelişmişlik

**Teknik olmayan faktörler**
- Eylemsizlik (eskiden vazgeçememe)
- Büyük destekçiler (yeni diller için) – problem çıktığında yazılım üreticisi destek sağlıyor mu?
- Derleyici/yorumlayici erişilebilirliği
- Kişisel tercihler/inanişlar

## 1.4. Programlama Dilleri Taksonomisi

**Taksonomi**: Bir bilgi dağarciğini parçalara ayırıp, parçalari arasindaki ilişkiyi tanimlayan yöntem.

Programlama dilleri için tarihten günümüze çeşitli yaklaşımlar ortaya atılmıştır. Bu yaklaşımlar arasındaki farklar; bazen tarihsel gelişimin doğal bir sonucu bazen de çözülecek problemin doğasından kaynaklanmıştır.

### 1.4.1. Komut Merkezli Diller

Her programlama dilinde komutlar vardır ve komutlar bir araya gelerek ifadeleri (statement) oluştururlar. İfadeler, bir programın cümleleri gibidir. Komut-merkezli programlamada ise en önemli şey ifadelerin sırayla çalıştırılmasıdır. Bu sırayı programcı belirler ve kodunu bu şekilde yazar. Yazılan kod, bir derleyici (compiler) tarafından makina diline çevrilir ve üretilen kod da çalışma zamanında makina üzerinde çalıştırılır. Komut-merkezli programlama, en eski ve en oturmuş yaklaşımdır. Yapısı, makina diline ve yapısına çok benzediği için, genelde performansı en yüksek diller komut-merkezli dillerdir.

**İlk yaklaşım: Makine ve Assembly Dili Yaklaşımı**

İlk yıllarda programlama doğrudan makine dilinde yapılıyordu. Ortaya çıkan koda makine kodu adı verilmekteydi. Programlar küçük, donanımlar ise ilkeldi. Önemli olan performanstı ve makine dili bunu sağlıyordu. Makine kodu komutları 0 ve 1'lerden oluşmakta idi. Buna birinci nesil kodlama adı verilmişti. Ardından çok az sayıda, ADD, PUSH, PULL gibi komutlar programlamada kullanılmaya başlanmıştır. Dilin adı Assembly'dir ve bu kodlama ikinci nesil kodlama olarak bilinir.

**İlk Yüksek Seviyeli Yaklaşım: Komut-merkezli Programlama**

Programlamayı daha kolaylaştırmak maksadıyla üçüncü nesil programlama dilleri geliştirilmiştir. Ilk yüksek seviyeli dil **Fortran**`dır. Bir IBM çalışanı olan John Backus liderliğindeki bir takım tarafından geliştirilen bu dil, 1957 yılında ticari olarak piyasaya sürülmüştür. Fortran'ın en temel özelliği, matematiksel problemleri kolayca çözmeye yarayacak yapıları içermesiydi.

**Ilerleme: Yordamsal Programlama**

FORTRAN alt yordam (subroutine) yapısına sahip olmasına rağmen, yordamların, programlarda yoğun olarak kullanılması ve merkezi bir önem kazanması için biraz zaman geçmesi gerekti. 1960'larda popüler olan yapısal programlama (structured programming), programı, alt parçalara bölmeyi tavsiye ediyordu. Bu yaklaşımda, akışı kontrol eden ana parça dışında, adına *birim* (module) denen irili ufaklı pek çok program parçası vardı. Birimler, ana birimden ya da diğer birimlerden değer alıp onlara değer döndürebiliyorlardı. Bu birimler, kendi yerel değişkenlerine (local variables) ve yerel ifadelere sahiptiler. Bu şekilde, ayrık işler/görevler birimlerde yazılıyor ve ana birimden tekrar tekrar çağrılabiliyordu. Böylece birimsel (modular) ya da başka bir deyişle yordamsal (procedural) programlama ortaya çıktı.

**Bir Sonraki Aşama: Nesneye Dayalı Programlama**

Yordamsal programlamadan ve komuta dayalı programlamadan oldukça ileri bir programlama tekniği olan nesneye dayalı programlama, yazılımı nesne özellik ve olaylarına dayalı olarak çözmektedir. Yeni sınıf ve nesnelerin kolayca oluşturulabilmesi, önceden meydana getirilmiş sınıfların yeniden kullanımı, programlama güvenliği ve buna benzer birçok yenilik getiren bir teknoloji olmuştur.

### 1.4.2. Bildirimci (declarative) Diller

Eğer bir program bir ilişki ve fonksiyon belirtiyorsa bildirimsel bir dil olarak kabul edilebilir. Bu modelde program değişkenlerine atama yapılmaz. Derleyici ya da yorumlayıcı belleği bizim için yönetir. İşlemci, geleneksel dillere göre daha uzaktan yönetilir. Bu dillerde başlıca üç yaklaşımdan bahsedilebilir:

**Fonksiyonel Diller (Functional Programming)**

Sadece fonksiyonlar üzerine kurulmuş bir modeldir. Fonksiyonlar bir çok değer alır ve geriye sadece bir değer döndürürler. Fonksiyonlar başka fonksiyonları çağırır ya da başka fonksiyonun parametresi olur.

**Mantıksal Diller(Logic Programming)**

Bir çeşit sembolik mantık sistemi olan önerme hesaplarına (predicate calculus) dayalı dillerdir. Önerme hesabı bilinen gerçeklerden yeni gerçekler ortaya çıkarmak için kurallar ve aksiyomlar (varsayım) sağlar. Bu temelli bir program, bir dizi varsayım ya da olgu, çıkarım kuralları, ispatlanmış sorgu ve teoremlerden oluşur. Eğer gerçekler sorguyu destekliyorsa sonuç doğru, aksi halde yanlıştır. Bu modeldeki en tanınmış örnek Prolog programlama dilidir.

**Veritabanı Dilleri (Database Languages)**

Veritabanı yönetimine yardımcı olmak üzere, başka dillerle beraber çalışabilen ve verilere odaklı dillerdir. Kendi içinde ikiye ayrılırlar: Verilerin yapısını ve ilişkisini tanımlayan kısım ve Veriler üzerinde işlemler yapan kısım. En iyi örnek SQL dilidir ve her ikisini de içerir.

## 1.5. Programlama dillerinin uygulama alanlarına göre sınıflandırılması:

Bilimsel ve Mühendislik Diller: Bu diller daha çok bilimsel ve mühendislik problemlerinin çözümünde tercih edilirler. PASCAL ve C dillerini, birde geleceği pek parlak olmayan ve hala ısrarla kullanılan 90 canlı, dünyanın ilk yüksek seviyeli dili FORTRAN' ı buna örnek verebiliriz.

Veritabanı Programlama Dilleri: Bu diller veritabanlarının genel olarak yönetiminde kullanılan dillerdir: DBASE, PARADOX, FOXPRO, SQL.. Kişisel bilgisayarlarda yaygın olarak kullanılanlardan bazılarıdır.

Yapay Zekâ Dilleri: bu diller insan davranışını taklit etmeye yönelik yapay zekâ içeren programların yazımında kullanılan mantıksal dillerdir. En ünlüleri: LISP ve PROLOG.

Genel Amaçlı Diller: Çok çeşitli konularda uygulama geliştirmek amacıyla kullanılan dillerdir. C ve PASCAL' ı örnek verebiliriz.

Sistem Programlama Dilleri: Sistem programlarının yazımında kullanılan dillerdir. C ' yi sembolik makine dillerini bu grup içinde ele alabiliriz.

## 1.6. Programlama Dilleri Değerlendirme Kriterleri

Programlama Dillerinin Gelişimi bölümünde tanitildiği gibi, çeşitli programlama amaçlarina uygun çok sayida programlama dili vardir.
Programlama dilleri arasinda seçim yapmak için çeşitli değerlendirme kriterlerine ihtiyaç duyulmaktadir.

Bir programlama dilinin değerlendirilmesinde göz önüne alinmasi gereken kavramlar çok sayidadir ve dilin kullanim amacina göre farklilik gösterebilmektedir. Ancak genel olarak bir programlama dilinin değerlendirilmesi için bazı kriterler belirlenmiştir. Bu kriterlerin en önde gelenleri, okunabilirlik, yazilabilirlik ve güvenilirlik olarak sayilabilir.

### Okunabilirlik

Bir programlama dilinin değerlendirilmesinde en önemli kriterlerden birisi, programların okunabilme ve anlaşılabilme kolaylığıdır. Programlama dillerinin okunabilir olmaları, programlarda hata olasılığını azaltır ve programların bakımını kolaylaştırır.

Bir programlama dilinde yer alan kavramlar, yapılar ve dilin sözdizimi, dilin okunabilirliğini doğrudan etkiler. Karmaşık bir sözdizimi, bir programın yazımı sırasında kısa yollar sağlayabilir ancak programın daha sonra değiştirilmek amacıyla okunmasını ve anlaşılmasını zorlaştırır.

### Yazılabilirlik

Bir programlama dilinde program yazma kolaylığını belirleyen en önemli etkenlerden birincisi, programlama dilinin sözdizimidir. Buna ek olarak, programlama dilinin soyutlama yeteneği, dilin yazılabilirliğini önemli ölçüde etkilemektedir.

Programlama dilleri, işlem veya veri olarak iki ayri şekilde soyutlama sağlayabilirler. Bunlara ek olarak, yazılabilirlik açisindan söz edilmesi gereken bir diğer konu, seçilen dilin, eldeki problem konusuna uygunluğudur.

**Güvenilirlik**

Bir programlama dilinin güvenilirliği, o dil kullanilarak geliştirilen programlarin güvenilir olmasidir.
Programlama dillerinde güvenilirlik, çeşitli faktörler tarafindan belirlenir. Bunlara örnek olarak, dilde bulunan tip denetimi ve istisnai durum işleme verilebilir.

Programin doğruluğunun sağlanmasi için programlama ortaminda sunulan araçlar da, güvenilir programlar geliştirilmesini etkiler.