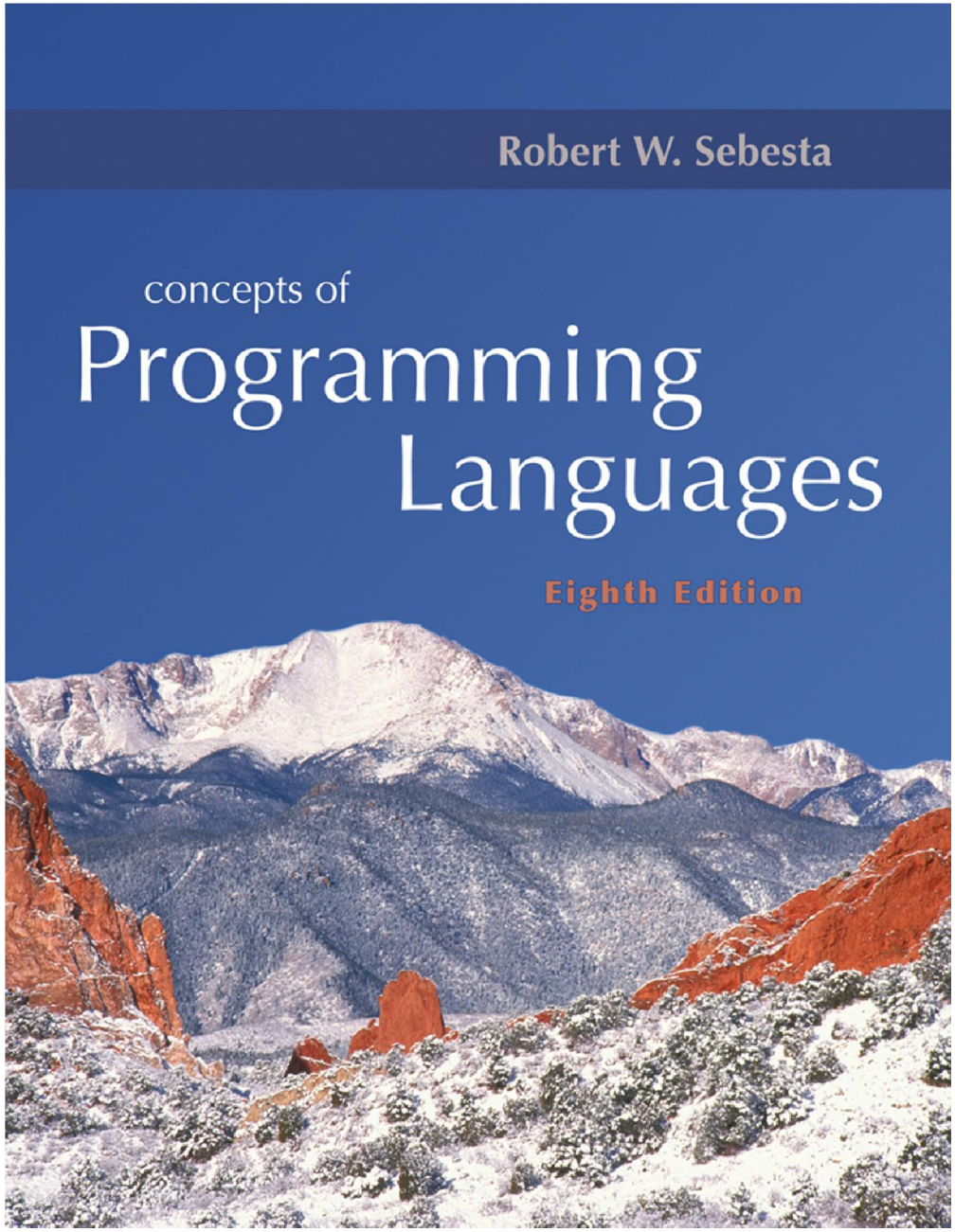
Bölüm 5

İsimler, Bağlamalar, Tip Kontrolü, Etki Alanları



ISBN 0-321-49362-1

5. bölüm konuları

- Giriş
- İsimler
- Değişkenler
- Bağlama kavramı
- Tip kontrolü
- Kuvvetli tipleme
- Tip Eşdeğerliği
- Etki alanı (Scope)
- Etki alanı ve ömür
- Referans Çevreleri (Referencing Environments)
- İsimli Sabitler

Giriş

- Komutlu diller von Neumann mimarisinin soyutlanmış halidir
  - Hafıza
  - İşlemci

İsimler

- Tasarım problemi:
  - Büyük harf – küçük harf farkeder mi?
  - Özel kelimeler “ayrılmış” mı yoksa “anahtar kelime” mi?

İsimler...

- Uzunluk
  - Çok kısa ise anlamlı olmaz
  - Örnekler:
    - FORTRAN I: en fazla 6
    - COBOL: en fazla 30
    - C#, Ada, and Java: sınır yok, tüm karakterler önemli
    - C++: sınır yok, ama pratikte var

İsimler…

- Büyük-küçük hassasiyeti
  - Dezavantaj: okunabilirlik (benzer görünen isimler gerçekte farklı)
    - C-tabanlı dillerde büyük-küçük hassasiyeti var.
    - Diğerlerinde yok.

İsimler…

- Özel kelimeler
  - *Anahtar kelime*: sadece bazi yerlerde özel, başka yerde : kullanıcı-tarafından tanımlanmış isim olarak kullanılabilir, ör. Fortran'da
    - `Real VarName (Real` *veritipi*`),`
    - `Real = 3.4 (Real` *değişken*`)`
  - *Ayrılmış kelime* : kullanıcı-tarafından tanımlanmış isim olarak kullanılamaz (If, while vs.)

Değişkenler

- Bir değişken bir hafıza hücresinin soyut halidir
- Değişkenlerin altı özelliği bulunur
  - Adı
  - Addresi
  - Değeri
  - Tipi
  - Ömrü
  - Etki alanı

Değişken özellikleri

- İsim – her değişkenin adı olmayabilir
- Adres – değişkenin ilişkili olduğu hafıza adresi
  - Bir değişkenin (x, y, z vs.) değişik zamanlarda değişik adresleri olabilir.
  - Bir değişkenin programın değişik yerlerinde değişik adresleri olabilir.
  - İki değişken ismi ayni hafiza hücresini gösteriyorsa, bu iki değişken birbirinin takma adıdır (alias).
  - Takma adlar işaretçilerle ve referans değişkenleri ile yaratılırlar
  - Takma adlar okunabilirliğe zarar verir.

Değişken özellikleri...

- *Tip* – değişkenin hangi değerleri alabileceğini ve tip değerleri üzerinde hangi operasyonalrın tanımladığını belirler, dolayısı ile değişkenin hangi operasyonların içinde yer alabileceğini belirler.
- *Değer* – değişkenin ilişkili olduğu adresteki değer.
  - Değişkenin sol-değeri = değişkenin ilişkili olduğu
  - Değişkenin sağ-değeri = değişkenin değeri
- *Soyut hafıza hücresi:* değişkenin ilişkili olduğu adrestedki bir veya daha çok fiziki hücre (ör: integer 4 bayt yer tutar)

Bağlama Kavramı

Bağlama bir ilişkilendirmedir. Örnek: işlem ile sembol (toplama ile + gibi) vs.

- *Bağlama zamanı* bağlamanın gerçekleştiği zamandır.

Olabilecek Bağlama Zamanları

- Dil tasarım zamanı-- ör: operatör sembollerının işlemlere bağlanması
- Dil gerçeklenmesi zamanı: -- ör: kayan nokta tipinin belli bir temsil şekline bağlanması
- Derleme zamanı – ör: C dilinde bir değişkenin bir tipe bağlanması
- Yükleme zamanı– ör: C `static` değikeninin bir hafıza hücresine bağlanması
- Çalışma zamanı – statik olmayan lokal değişkenlerin bir hafıza hücresine bağlanması

Statik ve Dinamik Bağlama

- Bağlama, eğer çalışma öncesi gerçekleşirse va çalışma süresince değişmezse, buna *statik bağlama* denir.
- Bağlama, eğer ilk kez çalışma esnasında gerçekleşirse, veya çalışma esnasında değişebilirse, buna *dinamik bağlama* denir.

Tip Bağlaması

- Tipler ne şekilde belirtilir?
- Bağlama ne zaman gerçeklesir? (statik–dinamik)
- Statik ise, *açık (explicit)* veya *kapalı (implicit)* sekilde tip belirtilebilir.

Açıkça/üstü kapalı Deklarasyon

- *Açıkça* deklarasyon değişkenlerin tipini deklare etmek için kullanılan bir ifadedir (ör: int x)
- *Üstü kapalı* deklarasyon otomatik olarak ilk kullanıldıkları yerde tipini belirlemekdir (ör: y=6.3)
- FORTRAN, PL/I, BASIC, ve Perl'de üstü kapalı Deklarasyon vardır
  - Avantaj: Yazma kolaylığı
  - Dezavantaj: güvenirlik

Dinamik tip bağlama

- Örneğin JavaScript and PHP'de var
- Atama komutu ile belirtilir.

```
list = [2, 4.33, 6, 8];
list = 17.3;
```

  - Avantaj: esneklik
  - Dezavantajlar:
    - Yüksek maliyet (dinamik tip kontrolü)
    - Derleyiçinin tip kontrolü yapması zor

Tip çıkarımı

- Kullanıldığı dillerden bazıları: ML, Miranda, and Haskell
  - Açık deklarasyona ihtiyaç yok. Tipler kullanım şeklinden yola çıkarak derleyici tarafından otomatik olarak bulunur.

Değişken özellikleri...

- Depolama Bağlamaları ve Ömür
  - Yer alma (allocation) – hafıza hücreleri havuzundan hüce alma
  - Yer verme (deallocation) – alınan hücreyi havuza geri verme
- Bir değişkenin *ömrü* bir hafıza hücresine bağlı olduğu zaman süresidir.

Ömür açısından değişken kategorileri

- Statik—program çalışmaya başlamadan hafıza hücrelerine bağlanır ve program çalışması süresince ayni hücreye bağlı kalır. Ör: C ve C++ `static` değişkenleri
  - Avantajları:
    - verimlilik (direkt adresleme – yığın üzeriden değil)
    - Tarihçeye duyarlı fonksiyon desteği
  - Dezavantajları: esneklik az (özyinelemeyi desteklemez)

Ömür açısından değişken kategorileri

- Yığıt-dinamik
  - Yer sistem yığıtı üzerinde alınır
  - Fonksiyonların lokal değişkenleri için
  - Değişkenlerin yer bağlaması değişkenlerin içinde bulundukları fonksiyonlar çağrıldığı zaman gerçekleşir
- Avantajları:
  - Özyinelemeye fırsat tanır
  - Yerden tasarruf (fonksiyon çalışmadığı zamanda yer alınmış olmaz)
- Dezavantajları:
  - Yer alma-verme fazladan masrafı
  - Fonksıyonlar tarihe duyarlı değil
  - Endirekt adresleme (yığıt çerçevesi başlangıcı + uzantı)

Ömür açısından değişken kategorileri

- *Açıkça yığın dinamik*
  - *Komutlarla program çalışırken yer alıp verme sağlanır (malloc, new, delete vs.)*
  - İşaretçilerle veya referanslarla erişim sağlanır
- Avantajları:
  - Dinamik yer yönetimi
  - İleri seviyeli veri yapılarınına olanak sağlar
- Dezavantajları:
  - Verimsiz
  - Güvenilmez (hata yapmaya elverişli)
  - Hafıza kaçakları

Ömür açısından değişken kategorileri

- *Üstü kapalı yığın-dinamik*
  - String, dizi ve diğer objelerin yer alımı-geri verilmesi **otomatik** olarak yığın (heap) üzerinde yapılır.
  - Bir değişkene atama yapıldığında (ör: x="abc") veya bir literal değer kullanıldığında (ör: print("abc")) başlatılır
- Avantaj
  - Kolay kullanım
- Dezavantajlar:
  - Verimsiz (bütün bilgi obje üzerinde saklanır)
  - Hata tespiti zordur (dinamik tipleme)

Tip kontrolü

- İşlem (operator) ve işlenilen (operand) kavramının atamalara ve foksiyonlara genişletilmesi olayı (operatörler doğru tipteki işlenilenle çalışırlar)
- *Tip kontrolü* operatörün parametrelerinin operatörün beklnetileri ile uyumlu olduğun doğrulnamasıdır
- *Uyumlu* bir tip ya operatörün beklediği tiptir, ya da derleyici tarafından üretilen kod tarafından otomatik olarak beklenilen tipe çevrilebilen bir tiptir. Otomatik çevrilmenin adı: *zorlama (coersion)*
- *Tip hatası:* Operatörün kabul edemeyeceği bir parametre tipine uygulanması

Tip kontrolü...

- Tip bağlamaları statik ise, hemen hemen tüm tip kontrolleri statik olabilir (program çalışmadan önce yapılabilir)
- Tip bağlamaları dinamik ise tip kontolü de dinamik olmalı
- Bir dilde tüm tip hataları bulunabiliyorsa bu dil *kuvvetlice tiplenmiş* demektir
  - Kuvvetlice tiplenmiş olmanın avantajı: tip hatasına sebebiyet veren değişken kullanım hatalarını bulur

Kuvvetlice tiplenme

Dil örnekleri:

- C ve C++: kuvvetlice tiplenmiş değil. (parametrelerin tip kontolunün yapılmasından kaçınılabilir)
- Ada büyük ölçüde kuvetlice tiplenmiştir
- Java ve C#: Ada gibi

Kuvvetlice tiplenme...

- Zorlama (coersion) kuvvetlice tiplemeyi zayıflatır (değişkenin kullanımı yanlış olsa bile bu yanlışlık zorlama yüzünden keşfedilemez)
- Java'da C++a göre zorlamaların ancak yarısı olsa da, Ada'ya göre kuvvetlice tipleme yönünden çok zayıftır

İsim Tip Eşdeğerliği

- İki değişkenin ancak ayni deklarasyonda tanımlanmışlarsa, veya ayni tanımlı tipi kullanıyorlarsa *isim tip eşdeğerliği* vardır.
- Gerçeklenmesi (implementatin) kolay, ama çok sınırlaryıcı:
  - Integer tiplerin ald dizileri integer ile uyumlu olmaz

Yapısal tip eşdeğerliği

- İki değişkenin iç yapıları ayni ise *yapısal tip eşdeğerlikleri* var demektir.
- Daha esnek, ama gerçeklenmesi daha zor

Değişken özellikleri: Etki alanı

- Bir değişkenin *etki alanı* görünebildiği ifadelerdir
- *Yerel olmayan* değişkenler bir program ünitesinde görünebilen, ama orada tanimli olmayan değişkenlerdir.
- Dilin etki alanı kuralları isimlerin hangi hafıza yerleri ile ilişkilendirileceğini belirler.

Statik etki alanı

- Program metnine bağlı
- Kullanılan değişkenin nerede tanımlı olduğunu bulmak gerekiyor
- Arama yöntemi: Tanımları içden dışa doğru ara
- Bir etki alanının dışındaki etki alanları onun statik atalarıdır. En yakındaki etki alanı ise statik ebeveynidir
- Bazı dillerde içiçe etki alanlarına sebebiyet veren içiçe fonksiyon tanımları mümkündür (ör: Ada, JavaScript, PHP)

Statik etki alanı …

- Değişkenler bir birimden/üniteden ayni isimde, daha içeride/yakında bir değişken tanımlamak sureti ile saklanabilir
- C++ ve Ada dillerinde bu "saklanmış" değişkenlere erişmek mümkündür
  - Ada: `birim.isim`
  - C++: `sınıf_adı::isim`

Bloklar

- Program birimi içinde statik etki alanı yaratma yöntemi
- Örnekler

C-tabanlı diller:

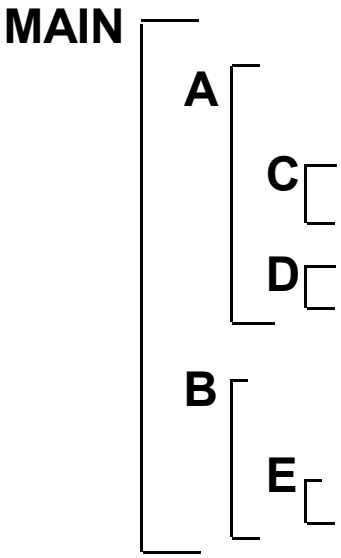
```
while (...) {
    int index;
  ...
}
```

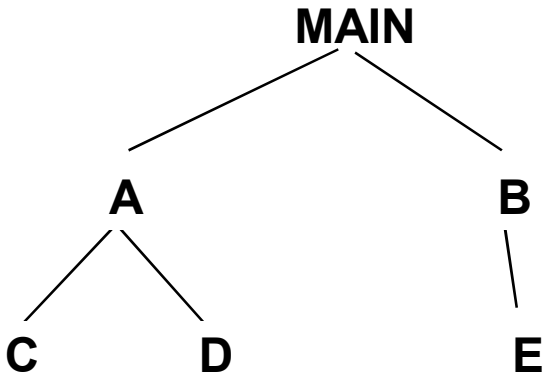
Ada:

```
declare Temp : Float;
begin
    ...
end
```

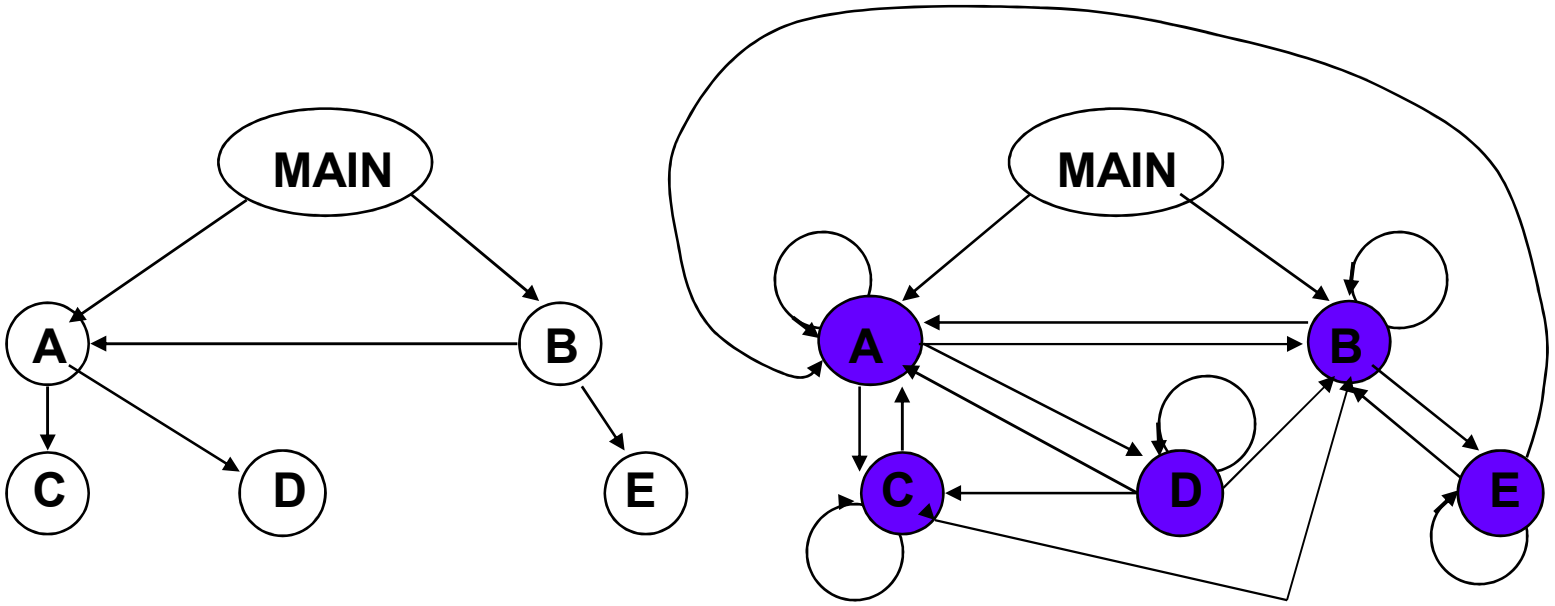
Statik etki alanı örneği

- MAIN, A ve B'yi çagırır
  A, C ve D'yi çağırır
  B, A ve E'yi çağırır





Statik etki alanı örneği ...



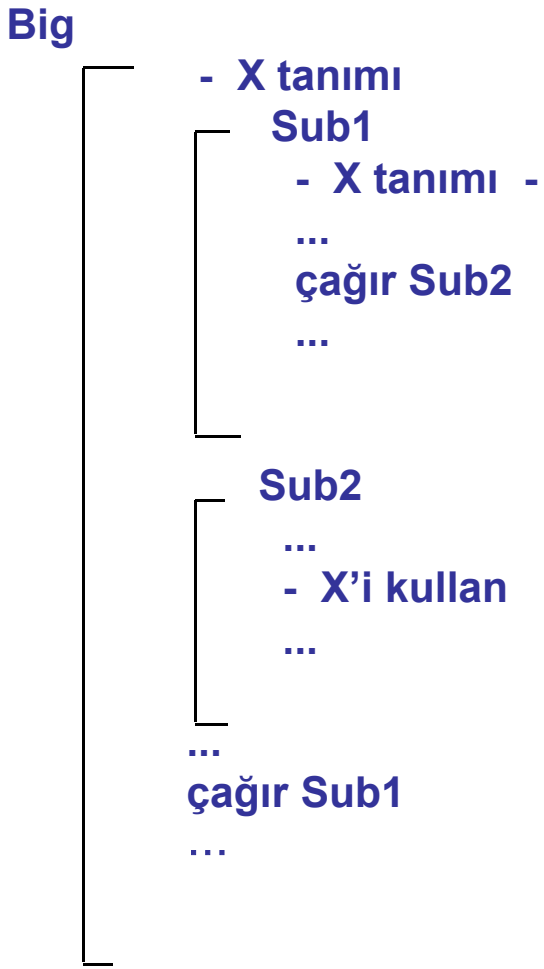
Dinamik etki alanı

- Program ünitelerinin çağrilma sırasına bağlı, içiçeliğe değil
- Değişkenler çağırma zincirini geriye doğru arayarak bulunur
- Evaluation of Dynamic Scoping:
  - Advantage: convenience (called subprogram is executed in the context of the caller)
  - Disadvantage: poor readability

Dinamik etki alanı değerlendirmesi

- Avantajı
  - Gerçeklenmesi kolay
- Dezavantajları
  - Okunabilirlik kötü (kullanılan bir yerel-olmayan değişkenin gerçekte nerede tanımlı olduğunu bulmak zor)
  - Tehlikeli
    - Bir fonksiyonun hangi fonksiyon tarafından çağrılcağının garantisi olmadığından istemeden yanlış değişkene erişmesi/değiştirmesi mümkün
    - Çağıran fonksiyonun kendinden habersiz değişkenlerinin değerlerinin değişmesi mümkğün

Etki alanı örneği



Big, Sub1'i çagırır
Sub1, Sub2'yi çağırır
Sub2 X'i kullanır

Etki alanı örneği ...

- Statik etki alanı
  - Big'deki X kullanılır
- Dinamik etki alanı
  - Sub1'deki X kullanılır X

Etki alanı ve ömür

- Etki alanı görünürlükle ilgili
- Ömür bir değişkenin ne zaman hafıza yerine bağlandığı, ne zaman bu bağın koparıldığı ile ilgili
- C veya C++ fonksiyonlarındaki `static` değişkenler
  - Etki alanı: fonksiyonun içi
  - Ömrü: Programın başladığı andan bittiği ana kadar

Referans Çevreleri

- Bir ifadenin *referans çevresi* o ifadedeki görülebilen tüm isimlerdir
- Statik etki alanlı bir dilde, lokal değişkenler + dıştaki etki alanlarındaki değişkenler
- Dinamik etki alanlı bir dilde, lokal değişkenler + çağrı zincirindeki fonksiyonların görünen değişkenleri

Özet

- Küçük/büyük harf duyarlılığı
- Değişkenleri tanımlayan 6 özellik: ad, adres, değer, tip, ömür, etki alanı
- Bağlama: özelliklerin program nesneleri ile ilişkilendirilmesi
- Sayısal değişkenlerin sınıflandırılması: statik, yığıt dinamik, açıkça yığın dinamik, üstü kapalı yığın dinamik
- Kuvvetli tipleme, tüm tip hatalarının keşfedilme garantisi demektir