



BMB214 Programlama Dilleri Prensipleri

Ders 5. Names , Variables , Bindings, and Scopes

Konular

- ◎ Names
- ◎ Değişkenler (Variables)
- ◎ Binding Kavramı
- ◎ Scope (Kapsam) Kavramı
- ◎ Scope ve Yaşam Zamanı (Ömür – Lifetime)
- ◎ Referans Ortamları (Referencing Environments)
- ◎ Adlandırılmış Sabitler (Named Constants)

Giriş

- ◎ Emir esaslı programlama dilleri von Neumann mimarisinin soyutlamalarıdır.
 - Hafıza (Memory)
 - İşlemci (Processor)
- ◎ Değişkenler (Variables)
- ◎ Bir tür (type) tasarlamak için scope, ömür, tür denetimi, başlatma (initialization) ve tür uyumluluğunu (compatibility) dikkate alınmalıdır.

Names

- ◎ Name: Değişkenleri, sabitleri, fonksiyonlara, türlere, işlemlere vb. ifade etmemizi sağlayan tanımlayıcılar
- ◎ Bir programlama name konusunda dikkat edilecek hususlar:
 - Name büyük / küçük harfe duyarlı mıdır?
 - Özel kelimeler ayrılmış kelimeler mi yoksa anahtar kelimeler mi?

Names ... Uzunluk

- ◎ Çok kısa bir name tanımlanırsa programcı/lar koda geri döndüğünde tekrar düzenlemesi kolay olmaz.
 - Bir sayaç değişkeni tutmak için x değişken ismi de kullanılabilir, count veya sayac gibi bir isimde kullanılabilir.
- ◎ Programlama dili örnekleri:
 - C99: sınır yok ama yalnızca ilk 63 önemli; ayrıca, harici adlar maksimum 31 ile sınırlıdır
 - C # ve Java: sınır yok ve hepsi önemli

Names ...

Özel karakterler

- ◎ PHP: tüm değişken isimleri dolar (\$) işaretleriyle başlamalıdır
- ◎ Perl: tüm değişken adları, değişkenin türünü belirten özel karakterlerle başlar
- ◎ Ruby: @ ile başlayan değişken isimleri örnek değişkenlerdir; @@ ile başlayanlar sınıf değişkenleridir

Names ...

Büyük küçük harf duyarlılığı

- ◎ Dezavantaj: okunabilirlik (birbirine benzeyen isimler farklıdır)
 - C tabanlı ve Python dillerdeki isimler büyük / küçük harfe duyarlıdır
 - C ++, Java ve C# 'de daha kötüdür çünkü önceden tanımlanmış adlar büyük / küçük harf karışıktır (ör. `IndexOutOfBoundsException`)

Names ...

Özel Kelimeler

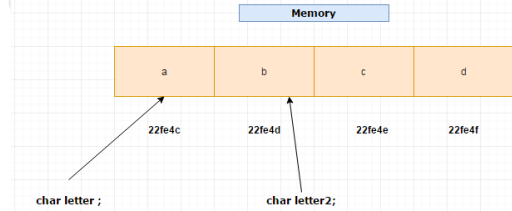
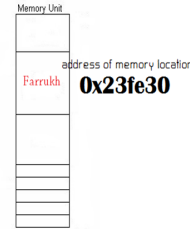
- ◎ Okunabilirliğe yardım; ifade maddelerini sınırlandırmak veya ayırmak için kullanılır
- ◎ Bir anahtar kelime, yalnızca belirli durumlarda özeldir.
- ◎ Ayrılmış kelime (Reserved Word), kullanıcı tanımlı ad olarak kullanılamayan özel bir kelimedir.
 - Ayrılmış sözcüklerle ilgili olası sorun: Çok fazla varsa, çok sayıda çarpışma meydana gelir (örneğin, COBOL'de 300 ayrılmış sözcük vardır!)

Variables (Değişkenler)

- ◎ Bir değişken, bir hafıza hücresinin bir soyutlamasıdır
- ◎ Değişkenlerde ortaya çıkan kavramlar:
 - Name
 - Adres
 - Value (Değer)
 - Tür (Type)
 - Ömür (Lifetime)
 - Scope (Kavram)

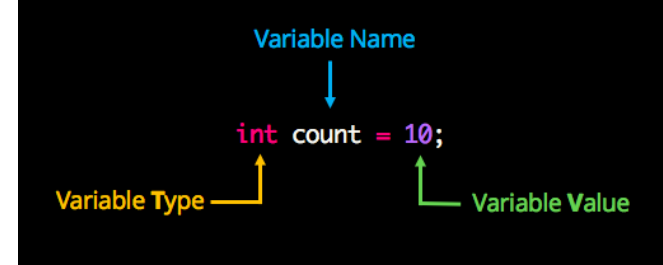
Variables ...

```
int main(int argc, char** argv) {  
    string name="Farrukh";  
    return 0;  
}
```



- ◎ Name - tüm değişkenler onlara sahip değildir
- ◎ Adres - ilişkili olduğu hafıza adresi
 - Bir değişken, yürütme sırasında farklı zamanlarda farklı adreslere sahip olabilir.
 - Bir değişken, bir programın farklı yerlerinde farklı adreslere sahip olabilir.
 - Aynı bellek konumuna erişmek için iki değişken adı kullanılabiliyorsa, bunlar takma adlar (aliases) olarak adlandırılır.
 - ◎ C ve C ++ gibi dillerde takma adlar işaretçiler, referans değişkenleri oluşturulur
 - ◎ Takma adlar okunabilirlik açısından zararlıdır (program okuyucuları hepsini hatırlamalıdır)

Variables ...



- ◎ Type (Tür) - değişkenlerin değer aralığını ve bu türdeki değerler için tanımlanan işlem kümesini belirler; kayan nokta (floating point) durumunda, yazım aynı zamanda hassasiyeti de belirler
- ◎ Value (Değer) - değişkenin ilişkilendirildiği konumun içeriği
 - Bir değişkenin sol değeri adresidir
 - Bir değişkenin sağ değeri onun değeridir
- ◎ Soyut bellek hücresi - bir değişkenle ilişkili fiziksel hücre veya hücre koleksiyonu

Binding (Bağlama)



- ◎ Binding, bir varlık ile bir öznitelik arasındaki bir ilişkidir, Örneğin
 - bir değişken ile türü değeri arasında
 - bir işlem ile bir sembol arasında
- ◎ Binding time (Bağlanma zamanı), bir bağlamanın gerçekleştiği zamandır.

Binding time (Bağlanma zamanı)

- ⊙ Dil tasarım süresi - operatör sembollerini operasyonlara bağlama
- ⊙ Dil uygulama süresi - kayan nokta türünü bir gösterime bağlama
- ⊙ Derleme zamanı (Compile Time) - bir değişkeni C, C# veya Java'daki bir türe bağlama
- ⊙ Yükleme zamanı (Load time) - bir C, C#, Java veya C++ statik değişkeni bir bellek hücreğine bağlama
- ⊙ Çalışma zamanı (Runtime) - statik olmayan bir yerel değişkeni bir bellek hücreğine bağlayın

Static and Dynamic Binding (Statik ve Dinamik Bağlama)

- ◎ Bir binding, ilk olarak çalışma zamanından (runtime) önce meydana gelirse ve programın yürütülmesi boyunca değişmeden kalırsa statiktir.
- ◎ Bir binding, ilk olarak yürütme (execution) sırasında meydana gelirse dinamiktir veya programın yürütülmesi sırasında değişebilir.

Type Binding

- ◎ Bir tür nasıl belirlenir?
- ◎ Bağlanma ne zaman gerçekleşir?
 - Statik ise tür, açık veya örtük bir bildirimle belirtilebilir

Explicit /Implicit Declaration (Açık/Örtülü Bildirim)

- ⦿ Açık bir bildirim, değişken türlerini bildirmek için kullanılan bir program ifadesidir
- ⦿ Örtülü bir bildirim, bildirim ifadeleri yerine varsayılan kurallar aracılığıyla değişken türlerini belirtmek için varsayılan bir mekanizmadır.
 - Basic, Perl, Ruby, JavaScript ve PHP örtük bildirimler sağlar
 - Avantaj: yazılabilirlik (küçük bir kolaylık)
 - Dezavantaj: güvenilirlik (Perl ile daha az sorun)

Explicit /Implicit Declaration ...

- ◎ Bazı diller, değişken türlerini (bağlam) belirlemek için tür çıkarımını kullanır
 - C# - bir değişken, var ve bir başlangıç değeri ile bildirilebilir.
 - Başlangıç değeri türü ayarlar Visual Basic 9.0+, ML, Haskell ve F# tür çıkarımını kullanır. Bir değişkenin görünümünün bağlamı, türünü belirler.

Dynamic Type Binding (Dinamik Tür Bağlama)

- ◎ Dinamik Tür Bağlama (JavaScript, Python, Ruby, PHP ve C # (sınırlı))
- ◎ Bir atama ifadesiyle belirtilir, ör. JavaScript

```
liste = [2, 4,33, 6, 8];
liste = 17.3;
```

 - Avantaj: esneklik (genel program birimleri)
 - Dezavantajları:
 - Yüksek maliyet (dinamik tip kontrol ve yorumlama)
 - Derleyici tarafından tür hatası tespiti zordur

Variables ...

◎ Depolama Bağlamaları ve Ömrü

- Allocation (tahsis) - bazı uygun hücre havuzlarından bir hücre almak
- Deallocation (Ayrılma) - havuza hücreyi geri koymak
- Bir değişkenin yaşam süresi, belirli bir hafıza hücresine bağlı olduğu süredir.

Yaşam Sürelerine Göre Değişken Kategorileri

Statik

- ◎ Statik - yürütme başlamadan önce bellek hücrelerine bağlanır ve yürütme boyunca aynı bellek hücresine bağlı kalır, örneğin, işlevlerdeki C ve C ++ statik değişkenleri
 - Avantajlar: verimlilik (doğrudan adresleme), geçmişe duyarlı alt program desteği
 - Dezavantaj: esneklik eksikliği (özyineleme yok)

Yaşam Sürelerine Göre Değişken Kategorileri...

Stack-dinamik

- ◎ Stack-dinamik (Stack-dynamic) - Bildirim ifadeleri detaylandırıldığında değişkenler için depolama bağlamaları oluşturulur.
Bir bildirim, kendisiyle ilişkili çalıştırılabilir kod çalıştırıldığında detaylandırılır)
- ◎ Skaler (Scaler) ise, adres dışındaki tüm öznitelikler statik olarak bağlıdır.
 - C alt programlarındaki (statik olarak bildirilmemiş) ve Java yöntemlerinde yerel değişkenler
 - Avantaj: özyinelemeye izin verir; depolamayı korur
 - Dezavantajları:
 - Tahsis ve tahsisi kaldırmanın ek yükü
 - Alt programlar geçmişe duyarlı olamaz
 - Verimsiz referanslar (dolaylı adresleme)

Yaşam Sürelerine Göre Değişken Kategorileri...

Explicit heap-dynamic

- ⊙ Explicit heap-dynamic - Programcı tarafından belirtilen, yürütme sırasında etkili olan açık yönergelerle ayrılır ve serbest bırakılır
- ⊙ Yalnızca işaretçiler veya referanslar aracılığıyla referans verildiğinde, örneğin C ++ 'da dinamik nesneler (new ve delete yoluyla), Java'daki tüm nesneler
 - Avantaj: dinamik depolama yönetimi sağlar
 - Dezavantaj: verimsiz ve güvenilmez

Yaşam Sürelerine Göre Değişken Kategorileri...

Implicit heap-dynamic

- ◎ Implicit heap-dynamic - atama ifadelerinin neden olduğu allocation ve deallocation
 - APL'deki tüm değişkenler;
 - Perl, JavaScript ve PHP'deki tüm string ve arrays
- ◎ Avantaj: esneklik (generic kode)
- ◎ Dezavantajları:
 - Verimsiz, çünkü tüm özellikler dinamik
 - Hata algılama kaybı

Scope (Kapsam)

- ◎ Bir değişkenin kapsamı, üzerinde görünür olduğu statement aralığıdır.
- ◎ Bir program biriminin yerel değişkenleri (local variables), o birimde bildirilenlerdir.
- ◎ Bir program biriminin yerel olmayan değişkenleri (nonlocal variables), birimde görünen ancak orada bildirilmeyen değişkenlerdir.
- ◎ Global değişkenler (Global variables), yerel olmayan değişkenlerin özel bir kategorisidir
- ◎ Bir dilin kapsam kuralları, isimlere yapılan referansların değişkenlerle nasıl ilişkilendirileceğini belirler.

Static Scope

- ◎ Bir değişkene bir ad başvurusu bağlamak için, siz (veya derleyici) bildirimi bulmalısınız
- ◎ Arama süreci(Search process): verilen ad için bir tane bulunana kadar ilk önce yerel olarak, ardından giderek daha geniş kapsamlı kapsamlarda arama bildirimleri yapar.
- ◎ Çevreleyen statik kapsamlar (belirli bir kapsamda), statik ataları (static ancestors) olarak adlandırılır; en yakın statik ataya statik ebeveyn (static parent) adı verilir
- ◎ Bazı diller, iç içe yerleştirilmiş statik kapsamlar oluşturan iç içe geçmiş alt program tanımlarına izin verir (örneğin Ada, JavaScript, Common Lisp, Scheme, Fortran 2003+, F # ve Python)

Scope...

- ◎ Değişkenler, aynı ada sahip "daha yakın" bir değişkene sahip olarak bir birimden gizlenebilir

Blocks

- ◎ Program birimleri içinde statik kapsamlar oluşturma yöntemi

Example in C:

```
void sub() {  
    int count;  
    while (...) {  
        int count;  
        count++;  
        ...  
    }  
    ...  
}
```

- ◎ Not: C ve C ++ 'da yasal, ancak Java'da değil ve C # - çok hataya açık

LET Construct (Yapısı)

- ⊙ İşlevsel dillerin çoğu, bir tür let yapı biçimi içerir
- ⊙ Bir let yapının iki bölümü vardır
 - İlk bölüm isimleri değerlere bağlar
 - İkinci kısım, birinci kısımda tanımlanan isimleri kullanır.
- ⊙ Scheme dilinde

```
(LET (  
  (name1 expression1)  
  ...  
  (namen expressionn)  
)
```

LET Construct ...

⊙ ML:

let

val name₁ = expression₁

...

val name_n = expression_n

in

expression

end;

⊙ F#:

- Birinci bölüm: let left_side = ifade
- (left_side bir ad veya bir tuple modelidir)
- Takip eden tek şey ikinci bölüm

⊙ Javascript

- https://www.w3schools.com/js/js_let.asp

Tanımlama Sırası

- ◎ C++, Java ve C# değişken tanımlamaları bir statement'ın görünebileceği her yerde görünmesine izin verir
- ◎ C++ ve Java'da, tüm yerel değişkenlerin kapsamı tanımlamadan bloğun sonuna kadardır
- ◎ C # 'da, bir blokta tanımlanan herhangi bir değişkenin kapsamı, bloktaki tanımlamanın konumuna bakılmaksızın tüm bloktur
- ◎ Bununla birlikte, bir değişken kullanılmadan önce yine de bildirilmelidir
- ◎ C ++, Java ve C # 'de, değişkenler for deyimlerinde bildirilebilir. Bu tür değişkenlerin kapsamı, for construct ile sınırlıdır.

Global Scope

- ◎ C, C ++, PHP ve Python, bir dosyada bir dizi fonksiyon tanımından oluşan bir program yapısını destekler
 - Bu diller, değişken bildirimlerin fonksiyon tanımlarının dışında görünmesine izin verir
- ◎ C ve C ++ hem bildirime (yalnızca özniteliklere) hem de tanımlara (öznitelikler ve depolama) sahiptir
 - Bir fonksiyon tanımının dışındaki bir bildirim, başka bir dosyada tanımlandığını belirtir

Global Scope PHP

- ◎ Programlar HTML içine, herhangi bir sayıda parçaya, bazı statement'lara ve bazı fonksiyon tanımlarına gömülüdür.
- ◎ Bir fonksiyonda (örtük olarak) tanımlanan bir değişkenin kapsamı, fonksiyon için yereldir.
- ◎ Dış fonksiyon tanımlanmasında örtük olarak bildirilen bir değişkenin kapsamı, tanımlanan programın sonuna kadardır, ancak araya giren fonksiyonları atlar
 - Global değişkenlere **\$GLOBALS** dizisi aracılığıyla veya **global** olarak tanımlama ile bir fonksiyon içinden erişilebilir

Global Scope Python

- © Fonksiyonlarda global bir değişkene referans verilebilir, ancak bir fonksiyona sadece fonksiyonda **global** olarak bildirilmişse atanabilir

Statik Kapsamın Değerlendirilmesi

◎ Birçok durumda iyi çalışır

◎ Problemler

- Çoğu durumda, çok fazla erişim mümkündür
- Bir program geliştikçe, başlangıç yapısı bozular ve yerel değişkenler genellikle küresel hale gelir; alt programlar da iç içe geçmekten ziyade küresel olmaya doğru yönelir

Dynamic Scope (Dinamik Kapsam)

- ⦿ Metin düzenlerine değil, program birimlerinin çağrı sıralarına göre çalışır.
- ⦿ Değişkenlere yapılan referanslar, yürütmeyi bu noktaya zorlayan alt program çağrıları zincirinde geriye doğru arama yapılarak bildirimlere bağlanır.

Dynamic Scope...

```
function big() {  
  function sub1()  
    var x = 7;  
    function sub2() {  
      var y = x;  
    }  
  var x = 3;  
}
```

big calls sub1
sub1 calls sub2
sub2 uses x

- Statik kapsama
 - sub2 içindeki x big'in x'dir
- Dinamik kapsama
 - sub2 içindeki x sub1'in x'dir

Dinamik Kapsam Değerlendirmesi

◎ Avantaj: kolaylık

◎ Dezavantajları:

- Bir alt program yürütülürken, değişkenleri çağırdığı tüm alt programlar tarafından görülebilir.
- Statik olarak kontrol etmek imkansız
- Zayıf okunabilirlik - statik olarak mümkün değildir bir değişkenin türünü belirle

Scope ve Yaşam Zamanı

- ◎ Kapsam ve ömür bazen yakından ilişkilidir, ancak farklı kavramlardır
- ◎ Bir C veya C++ işlevinde statik bir değişken düşünün

Referans Ortamları (Referencing Environments)

- ◎ Bir statement'ın referans ortamı, statement'ta görünen tüm adların koleksiyonudur.
- ◎ Statik kapsamlı bir dilde, yerel değişkenler artı tüm çevreleyen kapsamlardaki görünür değişkenlerin tümüdür
- ◎ Yürütülmeye başladıysa ancak henüz sonlandırılmadıysa bir alt program etkindir
- ◎ Dinamik kapsamlı bir dilde, referans ortamı yerel değişkenler artı tüm **aktif alt programlardaki** tüm görünür değişkenlerdir.

Adlandırılmış Sabitler (Named Constants)

- ◎ Adlandırılmış sabit, yalnızca depolamaya bağlandığında bir değere bağlanan bir değişkendir
- ◎ Avantajlar: okunabilirlik ve değiştirilebilirlik
- ◎ Programları parametrelendirmek için kullanılır
- ◎ Değerlerin adlandırılmış sabitlere bağlanması statik (bildirim sabitleri olarak adlandırılır) veya dinamik olabilir
 - Diller: C ++ ve Java: dinamik olarak bağlı her türden ifade
 - C # 'nin iki türü vardır, readonly ve const
 - const isimli sabitlerin değerleri derleme zamanı bağlanır
 - readonly isimli sabitler dinamik olarak bağlanır

Online Compilers

- ◎ <https://www.onlinegdb.com/>
- ◎ Online compiler and debugger
- ◎ Desteklediği diller:
 - C, C++, Python, Java, PHP, Ruby, Perl, C#, VB, Swift, Pascal, Fortran, Haskell, Objective-C, Assembly, JS, SQLite, Prolog