Programlama Dillerinin Prensipleri

HAFTA 5

BAĞLAMA KAVRAMLARI VE İSİM KAPSAMLARI

DR. ÖĞR. ÜYESİ DENİZ BALTA

Bağlama (Binding)

- Bir özellikle (isim, adres, değer vb) bir program elemanı (değişken, altprogram vb.) arasında ilişki kurulmasına **bağlama** (*binding*) denir.

- Bağlamanın gerçekleştiği zamana bağlama zamanı denir.

Bağlama (Binding)

- Bu özelliklerin bağlanma zamanına göre ve bağlamanın durağan yada dinamik olmasına göre diller farklılık gösterir.
- Dilin tasarımı, gerçekleştirimi ve derlenmesi zamanında yapılan bağlamalara statik bağlama denir.
- Çalışma zamanında yapılan bağlamalara ise dinamik bağlama denir.

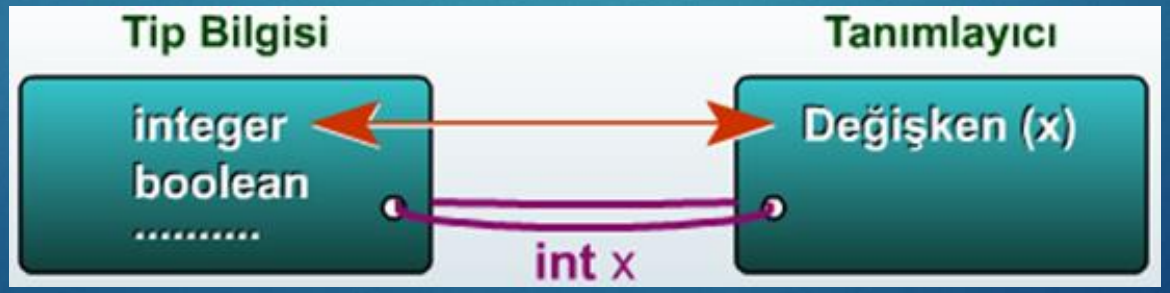
Bağlama zamanı

int hesap; … hesap=hesap+10;	
Hesap için olası tipler	Dilin tasarım zamanında
Hesap değişkeninin tipi	Dilin derlenmesi zamanında
Hesap değişkeninin olası değerleri	Derleyici tasarım zamanı
Hesabın değeri	Bu deyimin yürütülmesi zamanında
+ işlemcisinin muhtemel anlamları	Dilin tanımlanması zamanında
+ işlemcisinin bu deyimdeki anlamı	Derlenme süreci
10 literalinin ara gösterimi	Derleyici tasarımı zamanında
Hesap değişkeninin alacağı son değer	Çalışma zamanında

- Bir programlama dilinin semantik olarak anlaşılması için program elemanlarının özelliklerinin bağlama zamanlarının tam olarak anlaşılması şarttır.
- Örneğin bir alt programın ne yaptığının anlaşılması için bir çağrıdaki gerçek parametrenin altprogramın tanımındaki formal parametreye nasıl bağlandığının anlaşılması şarttır.
- Bir değişenin o anki değerinin bilinmesi için belleğe ne zaman bağlandığının bilinmesi gerekir

Tip bağlama

- Bir tanımlayıcı (id) bir tip bilgisi ile ilişkilendirilince o tiple bağlanmış olur.
- Bir programlama dilinde bir değişken kullanılmadan önce isimlendirilmeli, bir tip ile bağlanmalıdır.
- Böylece o değişkenin hangi değerleri alabileceği ve üzerinde hangi işlemlerin yapılabileceği belirlenmiş olur.
- Semantik anlam analizi için bu çok önemlidir.



Statik tip bağlama

Durağan Tip Bağlama	Dinamik Tip Bağlama
Derleme Zamanında	bir değişkenin tipi çalışma zamanında, değişkenin bağlandığı değer ile belirleniyorsa
bir değişken, *integer* tipi ile bağlanmışsa	bir değişken, atama sembolünün sağ tarafında bulunan değerin, değişkenin veya ifadenin tipine bağlanır ve değişkenin tipi, çalışma zamanında değişkenin yeni değerler alması ile değiştirilir. A=1.5 A=14 Avantaj:Esneklik (örneğin sıralama)
FORTRAN, Pascal, C ve C++'da bir değişkenin tip bağlaması durağan olarak gerçekleşir ve çalışma süresince değiştirilemez.	APL, LISP, SMALLTALK, SNOBOL4
derleyici, tip hatalarını, program çalıştırılmadan önce yakalar.	Derleyicinin hata yakalama yeteneği zayıftır. Statik tip kontrolü yapılamaz Yorumlayıcı kullanırlar

Statik tip bağlama

Örtülü (implicit) Tip Bağlama :

- Herhangi bir tanımlama deyiminin kullanılmaksızın, bazı varsayılan kurallar ile tip bağlaması yapılıyorsa buna Örtülü tip bağlama denir.
- Örneğin bir değişken isminin programda ilk kullanıldığı deyim ile ilişkili olarak tipini bağlanması örtülü tip bağlamadır.
- FORTRAN, PL/I, BASIC dilleri örtülü tanımlamalara sahiptir. FORTRAN'da bir değişenin ismi I,J, K, L, M, N harflerinden biri ile başlıyorsa bu değişken örtülü olarak INTEGER tipi ile aksi hallerde REAL tipi ile bağlanır.
- BASIC dilinde ise son karakteri $ olan değişkenler char tipi ile bağlanırlar.

Statik tip bağlama

Örtülü (implicit) Tip Bağlama :

- Örtülü tanımlamalar, programlama dilinin güvenilirliğini tehlikeye atabilir.
- Çünkü bu örtülü tip bağlamaları bazı tip hatalarının ve programcı hatalarının derleyici tarafından yakalanmasına engel olabilir.
- Programcının tanımlamayı unuttuğu bir değişkene örtülü olarak tip atamasının yapılması fark edilemeyen hatalar oluşturabilir.
- Bu yüzden PL/I, BASIC, Perl ve FORTRAN gibi dillerde örtülü tanımlamalar bulunmasına karşın günümüzde çoğu programlama dili dışsal tanımlama yolunu tercih etmektedir.

Statik tip bağlama

Dışsal Tip Bağlama:

- Bu yöntemde tip bağlaması için **int**, **float** ve **char** gibi bir deyim kullanılır.
- Bir çok dil güvenlik nedeniyle bu yöntemi tercih eder.

```
public static void main (String [] args)
{
    String x;
    x="Mustafa";
}
```

Dinamik tip bağlama

- Bir değişkenin tipi çalışma zamanında ve değişkenin bağlandığı değer ile belirleniyorsa, bu dil **dinamik tip bağlamalı** bir dildir.
- Burada bir bildirim deyimi kullanılarak tip bağlaması yapılmaz, bunun yerine bir atama deyiminde bu değişkene bir değer atandığı zaman tip bağlaması yapılır.
- İlgi değişken, atama sembolünün sağ tarafında bulunan değerin, değişkenin veya ifadenin tipine bağlanır. Eğer değişken çalışma zamanında yeni değerler alırsa değişkenin tipi değiştirilir.

Dinamik tip bağlama

- Dinamik tip bağlamalı dillerde derleyicinin hata yakalama şansı statik tip bağlamalı dillere göre daha zayıftır.
- Çünkü bir atama operatörünün sağ ve sol taraflarında herhangi iki tip görünebilir.
- Bu durumda atama operatörünün sağındaki yanlış tip hata olarak algılanmaz ve sol tarafın tipi böylece yanlış bir tipe dönüşebilir.

Dinamik tip bağlama

- Dinamik tip bağlamalı dillerde tip kontrolü çalışma zamanında yapılmak zorundadır.
- Bir değişkenin değeri için kullanılan bellek değişken boyutta olmalıdır.
- Çünkü farklı tipler farklı miktarda yer kaplamaktadır.
- Dolayısıyla dinamik tip bağlamalı dillerde statik tip kontrolü yapmak mümkün olmamaktadır.
- Dinamik tip bağlamalı dillerin gerçeklenmesinde bu yüzden yorumlayıcı kullanılmaktadır.
- Çünkü dinamik olarak değişen tipleri makine koduna çevirmek zor olacaktır.

Dinamik tip bağlama

Avantajları:

- Değişkenlere Dinamik olarak tip bağlanması, programlamaya esneklik sağlar.
- Dinamik tip bağlamalı bir dilde farklı tipteki değerlerin sıralanması mümkündür.
- Sıralama programındaki değişkenlerin tipleri çalışma zamanında belirlenebiliyorsa dinamik tip bağlamalı dil bir avantajdır.
- Durağan tip bağlamalı programlama diller sadece tek bir veri tipi için bir sıralama programı yazılabilir. Ayrıca bu veri tipi başlangıçta bilinmelidir.

Bellek bağlama

- Bir değişkenin ulaşılabilir bir bellek hücresi ile ilişkilendirilmesine **bellek yeri ataması** (*memory allocation*) denir.
- Değişkenin bu bellek hücresini iade etmesi ise **belleğin serbest bırakılması** (*deallocation*) olarak adlandırılır.
- Bir değişkenin bu bellek hücresi ile ilişkili kaldığı süreye ise (**bellek yeri ataması ile belleğin serbest bırakılması arası**) değişken için yaşam süresi (lifetime) denir.

Bellek bağlama

Program çalışma zamanı bellek düzeni:

- Programın derlenmiş hali yani derlenmiş kod parçası belleğin derlenmiş program kısmında saklanır.
- Program boyunca geçerli değişkenler (global değişkenler) statik (yığıt) bellek bölgesi kısmında tutulur.
- Yığın (Heap) bellek kısmı dinamik bellek değerleri için kullanılır. Dinamik bellek bölümü, gerektikçe büyüyebilir ve kullanılan bellek hücreleri iade edilebilir.

Bellek bağlama

Değişkenlerin bellek yeri bağlaması

etkinlik (*activation*) **kaydı**

Pascal-*dispose* Java-otomatik

aynı bellek bölümünün yeniden kullanılabilmesi

C'deki malloc fonksiyonu
C++ 'daki new işlemcisi

Doğrudan adresleme

Bellek bağlama

Değişkenlerin bellek yeri bağlaması

Statik değişkenler:

- Statik değişkenler, programın yürütülmesi başlamadan bellek hücrelerine bağlanırlar ve bellek hücreleri ile programın çalışması sonlanıncaya kadar bağlı kalırlar.
- Derleme zamanında bu değişkenler için bellek ayrılması gerçekleşir.
- Dolaylı adresleme gerektiren değişken türlerine erişim yavaş iken Doğrudan adreslemeden dolayı Statik değişkenler verimlidir, erişim hızlıdır.
- Statik değişkenler esneklik kriterine olumsuz etki etmektedir.

Bellek bağlama

Değişkenlerin bellek yeri bağlaması

Yığıt dinamik (stack-dynamic) değişkenler:

- Yığıt dinamik değişkenlerin bellek yeri bağlamaları kendilerine ilişkin tanımlama deyimleri çalıştığında gerçekleşir.
- Yığıt dinamik değişkenler için bellek yeri, çalışma zamanında bellekteki yığıt bellekten ayrılır.
- Dolayısıyla Yığıt dinamik değişkenler için ne kadar belleğe ihtiyaç olduğu derleme zamanında hesaplanamaz.
- ALGOL 60 ve bu çizgideki diller yığıt dinamik değişkenleri tanımlamaktadır. FORTRAN77 ve FORTRAN90 yerel olarak yığıt dinamik değişkenlere izin vermektedir. Pascal, C ve C++'da, lokal değişkenler, varsayılan olarak yığıt_dinamik değişkenlerdir.

Bellek bağlama

Değişkenlerin bellek yeri bağlaması

Dışsal heap dinamik değişkenler:

- Dışsal yığın dinamik değişkenler için ne kadar bellek gerektiği önceden bilinmez.
- Dolayısıyla bellek yeri bağlaması çalışma zamanında gerçekleşir.
- Çalışma zamanında veriler oldukça belleğe atanır ve bellek yeri yığın bellekten alınır ve daha sonra yığın belleğe iade edilir.
- Bu verilere sadece işaretçi (pointer) değişkenler aracılığıyla ulaşılabilir. Bu değişkenlerin tip bağlaması derleme zamanında gerçekleşir.

Bellek bağlama

Değişkenlerin bellek yeri bağlaması

Örtülü heap dinamik değişkenler:

- Örtülü değişkenler, sadece kendisine bir değer atandığı zaman belleğe bağlanırlar ve her yeni atamada tip ve bellek özellikleri yeniden belirlenebilir.
- Örtülü dinamik değişkenler kod yazımına esneklik kazandırmaktadırlar.
- Bunun yanında değişen tüm özelliklerin çalışma zamanında izlenmesi zorunluluğundan dolayı bir hız kaybı vardır.
- Ayrıca esneklikten dolayı derleyicilerin yakalayamayacağı hatalar olabilir.

İsim kapsamları

- Programda tanımlanan bir isim için hangi komutların ve deyimlerin bu isme ulaşabileceği ve bu ismin geçerli ve etkin olduğu program alanına isim kapsamı *(name scope)* denir.
- İsim kapsam, ismin tanımlandığı noktadan başlar ve o programlama dilinin kabul ettiği isim kapsamı kurallarına bağlı olarak sonraki bir noktaya kadar devam eder.
- İsimler geçerli olduğu kapsam alanı için lokal değişkendir. Bir kapsama onu saran kapsamalardan alınan isimler lokal olmayan değişkenlerdir.
- Ana program blokundaki bir isim ise global değişkendir.

İsim kapsamları

Statik Kapsam Bağlama:

- Statik kapsam bağlamada değişkenlerin kapsam alanları programın lexical – metinsel düzenine göre yapılır.
- Bir değişken ismi ile karşılaşıldığında değişkenin tanımı öncelikle bulunduğu blokta aranır.
- Eğer burada bu değişken bildirilmemişse programın lexical incelemesi ile fiziksel olarak kendisine en yakın blokta değişkene başvuru yapılır.
- Değişkenin bildirimin yapıldığı ilk yerdeki değerlere göre işlemler yapılır. Ve ilk çağrım noktasına dönülür.

İsim kapsamları

Statik Kapsam Bağlama:

Statik kapsam bağlamanın değerlendirilmesi:

- Programlama dillerinde Blok kavramı ile altprogramların ayrıştırılması kolaylaşmakla birlikte statik kapsam bağlamada, iç içe altprogramlarda fazla sayıda genel değişken kullanımına sebep olabilir.
- Programdaki genel değişkenler, gerekli olmasa bile, tüm altprogramlara görünür olacaklardır. Bu ise programlama dilinin güvenilirlik kriterine zarar verecektir.

İsim kapsamları

Dinamik Kapsam Bağlama:

- Eğer bir ismin kapsamı, altprogramların metinsel düzenine (lexical scope) değil de altprogramların çağrılış sırasına göre çalışma zamanında belirleniyorsa bu bağlamaya **dinamik kapsam bağlama** olarak adlandırılır.
- Dinamik kapsam bağlamada bir değişken ismi çalışma zamanında aynı isimli yeni bir değişken bulunana kadar, kendisinden sonra çalıştırılan tüm deyimlerde geçerlidir.

İsim kapsamları

Dinamik Kapsam Bağlama:

Dinamik Kapsam Bağlamanın Değerlendirilmesi:

- Dinamik kapsam bağlama kurallarının uygulanması kolaydır.
- Fakat procedure'de bir değişkene yapılan başvuru, deyimin her çalışmasında farklı değişkenleri gösterebilir.
- Ayrıca bir değişkenin aldığı değer çalışma zamanında değişebilmekte ve kullanıcı bunu programı okuyarak kolayca izleyememektedir.
- Bu ise programın anlaşılabilirliğini azaltmakta ve dillerin okunabilirlik ölçütünü olumsuz etkilemektedir.

Bloklar

- Blok, herhangi bir bildirimin, bağlamanın kapsamının sınırlandırıldığı program bölgesidir.
- Bir program blokunda deyimler bir araya getirilir ve bu deyimlere özgü yerel değişkenler tanımlanır. Bir blok içerisinde tanımlanan değişkenlere bu blokun **yerel değişkenleri** denir.
- O blok içinde görünebilen fakat orada bildirilmemiş değişkenlere ise o blok için **yerel olmayan değişkenler** denir.
- Her programlama dilinin kendine has blok yapısı vardır.
- Bir programlama dilinde altprogramlar içi içe yuvalanabiliyorsa bu dil **blok yapılı** bir dildir.

Bloklar

- Pascal, altprogramların yuvalanmasına izin verdiği için blok yapılı bir dil olarak nitelenmesine karşın, altprogram olmayan bloklar, Pascal programlarında yer alamaz.
- C'de altprogramlar yuvalanamaz ve isimsiz bloklar bulunabilir.
- C ve C++'da "**{... }**" arasındaki birleşik (*compound*) deyimler bir bloktur ve blok içerisinde yeni kapsamlar tanımlanabilir.
- Bloklarda tanımlanmış değişkenler, yığıt dinamik değişkenlerdir ve bu blok çağrıldığı zaman bellek kullanırlar.

Bloklar

- Modern programlama dilleri içiçe blok yapılarına izin vermektedir.
- Blok yapısı programlama dillerinde okunabilirliği ve ifade gücünü yükselten bir tekniktir.