Bağlama ve Kapsam Kavramları

İçerik

- Bağlama Kavramı
  - Bağlama zamanı,
  - Tip bağlama, Bellek bağlama
- İsim Kapsamları
  - Durağan Kapsam Bağlama
  - Dinamik Kapsam Bağlama konuları

Bağlama(Binding)

- Bir özellikle bir program elemanı arasında ilişki kurulmasına **bağlama** (*binding*) denir.

- Çeşitli programlama dilleri, özelliklerin program elemanlarına **bağlanma zamanı** ve bu özelliklerin **durağan** (*static*) veya **dinamik** (*dynamic*) olması açısından farklılıklar göstermektedir.

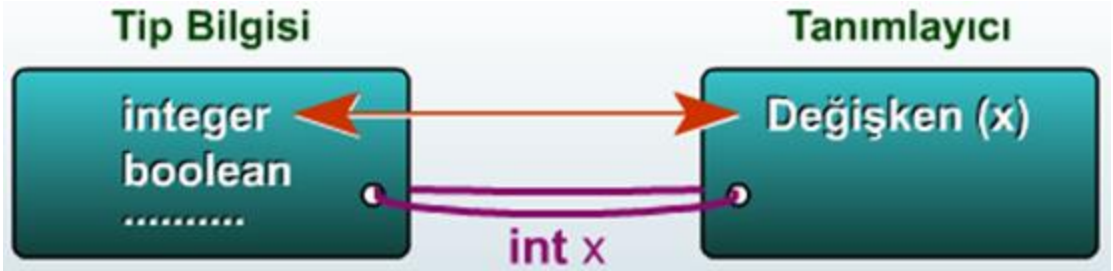
Bağlama Zamanı

- Bir programlama dilinde çeşitli bağlamalar farklı zamanlarda gerçekleşebilir.

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

Tip Bağlama

- Bir tanımlayıcı (id) bir tip bilgisi, 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.



Bağlama Zamanı

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

Durağan Tip Bağlaması iki türlüdür

Örtülü (*implicit*) ve **Dışsal** (*explicit*)

Tanımlama deyimleri kullanılmaz ve değişkenlerin tipleri, varsayılan (*default*) kurallar ile belirlenir

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:son karakteri $ ola değişkenler karakter tipi ile bağlanır.

Yazım yanlışlığı hataların derleme sırasında yakalanması engellenebilir.
Programlama dilinin güvenilirliğini azaltırlar.

Örnek: PL/I, BASIC, PERL ve FORTRAN

Değişkene, programda yer alan bir tanımlama deyimi ile bir tip ile bağlanır.

```
procedure D;
var n: char;
    begin
n:= "D" ;
          W;
    end;
```

```
{ int t=a;

a=b;

b=t;

}
```

Bellek Bağlama

- (*allocation*) (*deallocation*) lifetime

etkinlik (*activation*) **kaydı**

aynı bellek bölümünün yeniden kullanılabilmesi

Pascal-*dispose* Java-otomatik
C'deki malloc fonksiyonu
C++ 'daki new işlemcisi

Doğrudan adresleme

Statik değişkenler, programın yürütülmesi başlamadan bellek hücrelerine bağlanırlar ve bellek hücreleri ile programın çalışması sonlanıncaya kadar bağlı kalırlar. FORTRAN I, II ve FORTRAN IV'de hepsi statik. C, C++ ve Java **static** anahtarını kullanır.

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.

Dışsal yığın dinamik değişkenlerin bellek yeri bağlaması çalışma zamanında gerçekleşir. Ne kadar bellek gerektiği önceden bilinmez. 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, bellek yeri bağlaması ise çalışma zamanında gerçekleşir.

	Statik	*Stack*	*Heap*
Ada		Lokal değişkenler, altprogram parametreleri	*implicit*: local değişkenler; *explicit*: new (garbage collection)
C	global değişkenler; statik local değişkenler	Lokal değişkenler, altprogram parametreleri	*explicit* : **malloc** ve **free**
C++	C ile aynı, static sınıf üyeleri	C ile aynı	*Explicit:* **new** ve **delete**
Java		Sadece ilkel tipli local değişkenler	*Implicit*: her sınıf(garbage collection)
Fortran77	global değişkenler(bloklar), lokal değişkenler ve altprogram parametreleri (implementation dependent); **SAVE,** static bellek atamasını düzenler	Lokal değişkenler, altprogram parametreleri (implementation dependent)	
Pascal	global değişkenler(compiler bağımlı)	global değişkenler(compiler dependent), local değişkenler altprogram parametreleri	Explicit: **new** ve **dispose**

İSİM KAPSAMLARI (Name Scope)

- Belirli isim tanımlarının etkin olduğu bir program alanına **isim kapsamı** denir.

Statik isim kapsam	Dinamik Kapsam
Değişkenlerin kapsamları, programın metinsel düzenine göre, fiziksel yakınlığa göre, belirlenir.	Bir ismin kapsamının, altprogramların fiziksel yakınlıklarına göre değil, altprogramların çağrılma sırasına göre çalışma zamanında belirlenmesi **dinamik kapsam bağlama** olarak adlandırılır.
ALGOL 60'ı izleyen çok sayıda dilde tanımlıdır. Altprogramlar iç içe yuvalanabilir. (C++ ve FORTRAN hariç)	LISP, APL dillerinin ilk sürümleri
1. Altprogramların yuvalanması sonucu gereğinden fazla genel değişken kullanımı olabilir. 2. Bir programda genel olarak tanımlanan değişkenler tüm altprogramlara görünebilir olacakları için güvenilirlik azalmaktadır.	1. Bir altprogramda bir değişkene yapılan başvuru, deyimin her çalışmasında farklı değişkenleri gösterebilir. 2. Programların anlaşılabilirliğini azaltmaktadır

ÖRNEK

```
program L;
    var n: char ;                    {n, L' de bildirilmiş     }
    procedure W;
            begin
                writeln(n)     {W de n 'ye başvuru var.}
            end;
    procedure D;
            var n: char;             {D de n tekrar bildirilmiş}
            begin
                n:= "D" ;
                W;                   { D' deki W' i çağırdı}
            end;
    begin  {L}
        n:= "L";
        W;                           {Ana program L' den W' u çağrıldı}
        D;
    end.
```

statik kapsam bağlama kuralına göre

L
L

Dinamik kapsam bağlama kuralına göre

L
D

Örnek: Aşağıdaki program parçasının çıkışını

a) statik kapsam bağlama kurallarına göre
b) Dinamik kapsam bağlama kurallarına göre bulunuz.

```
int x;

int main() {
  x = 2;
  f();
  g();
}

void f() {
  int x = 3;
  h();
}

void g() {
  int x = 4;
  h();
}

void h() {
  printf("%d\n",x);
}
```

Dinamik kapsam bağlamaya göre çıkış: 3 4

Statik kapsam bağlamaya göre çıkış: 2 2

Özet

- Bu hafta
- Programlama elemanlarıyla çeşitli özelliklerinin ilişkilendirilmesini sağlayan bağlama konusu, durağan-dinamik tip, bellek ve isim kapsamı bağlama yönüyle incelenmiştir.
- İsim kapsamlarının durağan kapsam bağlama ve dinamik kapsam bağlama seçenekleri incelenmiş ve karşılaştırılmıştır.