5. İsimler, Bağlamalar ve Kapsamlar

5.1. İsimler

Değişkenlerin yanısıra alt yordamlar, parametreler ve diğer program yapılarının da isimleri vardır. İsim kelimesi yerine tanımlayıcı (identifier) kelimesi de kullanılmaktadır.

5.1.1. Özel kelimeler

Çoğu dilde özel kelimeler değişken veya fonksiyon adı olarak kullanılamaz. Fakat bazı dillerde anahtar kelimeler (keyword) vardır ve bunların yeniden tanımlanmasına izin verilmektedir. Örneğin, Fortan dilinde özel kelimeler anahtar kelimelerdir. Bu ayrımın olduğu yaygın kullanılan tek dildir. Örneğin Integer kelimesinin aşağıdaki kullanımlarında, Integer'ın anlamı içeriğe bakarak kararlaştırılır. İlk ifadede Integer tipinde Apple isimli bir değişken tanımlanıyorken, ikinci ifadede Integer adlı değişkene 4 değeri atanmaktadır.

```
Integer Apple
Integer = 4
```

Ayrılmış kelimeler (reserved words), programlama dilinin isim olarak kullanılamayan özel kelimeleridir. Bir dil tasarlarken anahtar kelimeler yerine ayrılmış kelimeler tercih edilmelidir. Çünkü anahtar kelimelerin kullanımı kafa karıştırıcı olabilir. Örneğin Fortran'da şu ili ifade de geçerlidir:

```
Real Integer
Integer Real
```

İlk ifadede Real tipinde Integer adında bir değişken tanımlanıyorken, ikinci ifadede Integer tipinde Real adında bir değişken tanımlanmaktadır.

Çok fazla ayrılmış kelime kullanımı da tercih edilmez çünkü kullanıcı anlamlı değişken ismi bulma sıkıntısı yaşayabilir.

5.2. Değişkenler

Bir değişken, bir bellek hücresinin veya hücreler topluluğunun soyutlamasıdır. Genellikle değişkeni isimlendirirken bir bellek bölgesini isimlendirdiğimizi düşünürüz ama bir isim değişken için bundan daha fazlasıdır.

Makine dilinden assembly diline geçiş, gerçek bellek adresleri yerine isim kullanımını beraberinde getirmiştir. Bu da okunabilirliği artırmıştır.

Bir değişken altı özellikle tanımlanabilir: isim, adres, tip, yaşam süresi ve kapsam.

5.2.1. İsim

Bir isim, programdaki bir varlığı tanımlamak için kullanılan karakter dizisidir. Bazı dillerde isimlerin uzunluğu sınırlanmıştır. Çeşitli formlarda tanımlama kuralları olabilir. Örneğin, PHP'de değişken adları $ işareti ile başlar. Perl'de adının başındaki $, @ veya % sembolleri değişkenin tipini belirler. Çoğu dilde büyük/küçük harf ayrımı vardır.

5.2.2. Adres

Değişkenin adresi, kendisi ile ilgili olan bellek adresidir. Bu çok basit bir olay gibi gözükse de, çoğu programlama dilinde program süresince aynı değişken farklı bellek adresleri ile ilişkilendirilebilir.

Farklı değişkenlerin aynı adrese sahip olmaları da mümkündür. Aynı bellek bölgesine erişebilen birden fazla değişkene aliases denir. Aliasing, okunabilirlik için bir ayak bağıdır çünkü bir değişkenin değeri başka bir değişkenin değeri değiştirildiğinde değişmektedir. Aliasing'e örnek olarak, C dilindeki Union, aynı bellek bölgesini işaret eden pointerlar verilebilir.

Bir değişkenin bir adresle eşlenme zamanı programlama dillerini anlamada büyük önem taşır.

5.2.3. Tip

Değişkenlerin tipleri, değişkenin alabileceği değer aralığına ve o değişken üzerinde yapılabilecek işlemlerin ne olabileceğine karar verir.

5.2.4. Değer

Değişken ile ilişkilendirilmiş bellek hücresi veya hücrelerinin içeriği, değişkenin değeridir.

5.3. Bağlama (Binding)

Bir varlığı, bir özelliği ile eşlemeye bağlama diyoruz. Örneğin bir değişken ile tipini veya değerini eşleme gibi. Bağlamanın yapıldığı zaman ve bağlama, programlama dillerinin anlamsallığında öne çıkan konulardır. Bağlamalar, dil tasarım zamanında, dil gerçekleştirim zamanında, derleme zamanında, yükleme, link ve çalıştırma zamanlarında yapılabilir. Örneğin * işareti, çarpma işlemi ile dil tasarım zamanında bağlanır. C dilindeki int veri tipi, alabileceği değer aralığına dilin gerçekleştirim zamanında bağlanır. Java dilindeki bir değişken, tipine derleme zamanında bağlanır. Ya da Java' daki alt yordam değişkenleri, bellek hücresine çalışma zamanında bağlanır. Kütüphane çağrıları, link zamanında alt programlara bağlanabilir.

Şu Java ifadesini ele alalım:
count = count +5;

- count değişkeninin tipi derleme zamanında bağlanır.
- count'un alabileceği değerler kümesi derleyici tasarımı zamanında bağlanır.
- + işaretinin anlamı derleme zamanında bağlanır. Çünkü, işleme girenlerin ne olduğuna bağlı olarak anlamı değişebilir.
- 5 rakamının tanımı derleyici tasarımı zamanında bağlanır.
- count'un bu ifade sonrasında değeri çalışma zamanında bağlanır.

Bağlama zamanlarının tamamen anlaşılması, programlama dilinin anlamsallığını kavramada ön koşuldur.

5.3.1. Değişkenlere Özelliklerin Bağlanması

Çalışma zamanı başlamadan önce yapılan ve programın çalışması boyunca değişmeyen bağlamalara statik bağlama denir. Eğer bağlama çalışma zamanında yapıldıysa veya programın çalışması boyunca değişiyorsa bu dinamik bağlama olarak adlandırılır.

5.3.2. Bağlama

Bir değişken program içinde kullanılmadan önce bir veri tipi iile bağlanmak zorundadır.

5.3.2.1. Statik Tip Bağlama

Açık (Explicit) tanımlama, bir program içindeki değişkenleri ve tiplerini bir ifade ile tanımlamadır. Dolaylı (Implicit) tanımlama, değişkenlere varsayılan eğilimleri dikkate alınarak bir tip eşlemektir. Dolaylı tanımlama, bir değişkenin adının ilk görüldüğü yerde tip tanımlamasının yapılmasıdır. Her iki tanımlama da statik tip bağlama oluşturur. 1960' lardan sonra geliştirilen çoğu programlama dilinde açık tanımlama yapmak gerekmektedir. Dolaylı tanımlamaya örnek olarak Fortran dilinde bir değişkenin adı **I, J, K, L, M** ile başlıyorsa tipinin **integer**, bunun haricinde de **real** olarak tanımlanmasını verebiliriz. Bir başka dolaylı tanımlama örneği C# dilinde **var** kelimesi ile tanımlanan değişkenin tipinin tanımlanması için ilk değerinin atanması gerekliliğidir.

Örn:
var sum = 0; // integer olarak tanımlandı
var total = 0.0; // float olarak tanımlandı
var name = “Mehmet”; // string olarak tanımlandı

Yukarıdaki örnekte görülen değişkenlerin tipleri dolaylı olarak tanımlanmıştır ve program sonuna kadar sabit kalır. C#' tan başka Visual Basic 9.0+' da ve F# gibi bazı fonksiyonel dillerde dolaylı tip tanımlama vardır.

5.3.2.2. Dinamik Tip Bağlama

Bir değişken bir tipe bir atama ifadesi içinde bağlanır. Atama ifadesi çalıştırıldığında değişken eşitliğin sağ tarafındaki değerin tipine bağlanır. Böyle bir atama sonunda değişken bir adrese ve bir bellek hücresine de bağlanmış olabilir. Her değişkene her veri tipinde değer atanabilir. Bir değişkenin tipi program boyunca defalarca değişebilir. Dinamik tip bağlama ile bağlanan değişkenlerin geçici olabileceği unutulmamalıdır. Bu tip bağlamayı kullanan dillerde esneklik yüksektir. LISP, Python, Ruby, JavaScript ve PHP dinamik tip bağlama kullanan dillere örnektir. C# 2010'da dinamik tip bağlama özelliği eklenmiştir ve bu özelliği kullanmak için değişken, isminin başında dynamic kelimesi koyularak tanımlanır. Örn: dynamic any;

Dinamik tip bağlama, programı daha az güvenilir yapmaktadır çünkü derleyicinin hata yakalama yeteneği statik tip bağlama kullanan derleyicilere kıyasla azalmıştır. Hatalı değer ataması yoktur. Atamada, sağ tarafın değerine göre sol tarafın tipi değiştirilir. Dinamik tip bağlama kullanan diller saf yorumlama kullanarak gerçekleştirilmişlerdir. Burada programın çalışma hızının önemi devreye girmektedir.

5.3.3. Depo Bağlama ve Yaşam Süresi

Bir değişkene bağlanacak bellek hücresi, uygun bellek havuzundan alınmak zorundadır. Bu işleme tahsis etme (allocation) diyoruz. Geri bırakma (deallocation) da bir değişkenle bağı koparılan bellek hücresini bellek havuzuna yerleştirmektir.

Bir değişkenin yaşam süresi, belirli bir bellek bölgesine bağlı olduğu zaman zarfıdır. Dolayısıyla bir değişkenin yaşamı bir bellek hücresine bağlandığında başlar ve o bellek hücresinden ayrıldığında da sona erer. Değişkenleri yaşam sürelerine göre 4 gruba ayırabiliriz: statik, stack-dinamik, açık (explicit) heap-dinamik, dolaylı (implicit) heap-dinamik

5.3.3.1. Statik Değişkenler

Statik değişkenler bellek bölgesine program çalışmaya başlamadan bağlanırlar ve çalışma sona erene kadar aynı bellek hücresine bağlı kalırlar. Bir alt program yeniden çağrıldığında, bir önceki çağrıda oluşan değeri kaybetmek istemediğimiz durumlarda lokal statik değişkenler kullanırız. Global olarak erişilebilen değişkenler de program boyunca aynı bellek bölgesine bağlı kalırlar.

Statik bağlamanın dezavantajı esnekliği azaltmasıdır. Sadece statik değişkenlere sahip bir dilde özyineleme desteği bulunamaz. Diğer bir dezavantaj da belleğin değişkenler arasında paylaşılamamasıdır. Çok büyük diziler üzerinde işlem yapan iki alt yordam düşünelim. İki alt yordamın hiçbir zaman aynı anda çalışmadıkları ve statik dizi kullandıklarını da varsayarsak çok fazla bellek işgali olacaktır.

5.3.3.2. Stack-Dinamik Değişkenler

Bu tip değişkenler hafızaya çalışma zamanında bağlanır ancak hafızada sabit olarak kalırlar. İçerikleri zamanla değişebilir.

5.3.3.3. Açık (Explicit) Heap-Dinamik Değişkenler

Bu değişkenler, tipleri kullanıcı tarafından açıkça belirtilerek kullanılabilen ve dinamik olarak tahsis edilip geri bırakılabilen bellek alanlarıdır. Programlama dillerinde dinamik hafıza yönetimi özelliği sayesinde kullanıcı hafızadan istediği boyutta birbölge ayırabilir (allocation). C ve C++'ta bulunan **malloc**, **realloc** ve **calloc** komutları bu amaçla kullanılır. **Free** komutu ise bir bellek bölgesinin bellek havuzuna geri bırakılması için kullanılır. Yine, Java, C++ ve C# gibi dillerde bu iş için **new** komutu kullanılmaktadır. Örneğin,

```
insan ayse; // ayse insan tipinden nesneleri gösterebilir
ayse = new insan(); //ayse için bellekte yer ayrılır.
```

5.3.3.4. Dolaylı (Implicit) Heap-Dinamik Değişkenler

Bu tip değişkenlere değer atandığı zaman bir yığıta bağlanırlar. Örn: JavaScript' teki şu atama ifadesi:

```
highs = [74, 84, 86, 90, 71]
```

Burada highs daha önce ne tipte bir değişken olduğundan bağımsız olarak bu atamadan sonra bir dizi (array) olarak tanımlanmaktadır.

5.4. Kapsam

Bir değişkenin görünür olduğu program bölümlerine kapsam denir. Bir değişken bir ifade tarafından kullanılabiliyorsa, o değişken o ifade için görünürdür.

Bir değişken bir program birimi veya bir blok içinde tanımlanmışsa yerel (local) değişken olarak adlandırılır. Yerel olmayan değişkenler bir program birimi veya blok içinden erişilebilir ancak orada tanımlanmamışlardır. Global değişkenler de yerel olmayan değişkenlerin özel bir alt grubudur.

5.4.1. Statik Kapsam

Değişkenlerin kapsamı çalışma zamanından önce statik olarak belirlenir.

Örn:

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

Yukarıdaki örnekte, statik kapsam sözkonusuyken, sub2 içinde kullanılan x değişkeni, big fonksiyonu içinde tanımlanmış olan x'tir. Sub2 içinde x kullanıldığında x'in en yakında nerede tanımlandığına bakılır ve x'in, sub2 içinde bir tanımı olmadığı görülür. Bu durumda sub2'yi kapsayan ilk fonksiyona gidilir yani bir üst kapsama çıkılır. Bir üst fonksiyonda yani big fonksiyonunda x'in tanımını görmek mümkündür. Dolayısıyla sub2 içinde kullanılan x, big içinde tanımlanan x' tir. Sub1 içindeki x tanımı sub2 için bir anlam ifade etmez çünkü sub1 sub2'yi kapsayan bir fonksiyon değildir.

5.4.2. Bloklar

Çoğu programlama dili çalıştırılabilir kodun orta yerinde yeni statik kapsamlar oluşturmaya izin verirler. Böylece bir kod bölümü kendi yerel değişkenlerine sahip olabilir. Bu değişkenler stack-dinamik' tir; o kod bölümüne girildiğinde bellekteki yerleri ayrılır ve kod bölümünün sonuna gelindiğinde kendileri için ayrılan yer belleğe geri bırakılır. Bu tip kod bölümlerine blok diyoruz. C-tabanlı dillerde {} arasına yazılan ifadelerin birleşimi bir blok oluşturur. Örneğin, **list** integer tipinde bir dizi olsun. Aşağıdaki ifadede bir if bloğu görmekteyiz. **temp** değişkeninin kapsamı yalnızca bu if bloğudur.

```
if (list[i] < list[j]) {
        int temp;
        temp = list[i];
        list[i] = list[j];
        list[j] = temp;
}
```

Kapsamlar bloklarla oluşturulur. Bloklar daha büyük bloklar tarafından kapsanmış olabilirler. Bir blok içinde adı geçen değişken o blok içinde tanımlanmamışsa üst seviyelerdeki bloklara bakılır.
Örn:

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

Yukarıdaki örnekte while bloğu içinde kullanılan count değişkeni while bloğu için yerel bir değişkendir ve bir üst seviyede sub() içinde tanımlanan count değişkeninden bağımsızdır.

5.4.3. Global Kapsam

C, C++, PHP, JavaScript ve Python gibi bazı diller, değişkenlerin fonksiyonlar dışında tanımlanmasına imkan verirler. Bu değişkenlere global değişkenler denir ve o dosyadaki tüm fonksiyonlar tarafından görünür olurlar.