Programlama Dilleri Prensipleri

Ders 6. Data Types (Veri Türleri)

Konular

- İlkel Veri Türleri (Primitive Data Types)
- Character String Türleri
- Enumeration Türleri
- Array Türleri
- İlişkisel Diziler (Associative Arrays)
- Record Türleri
- Tuple Türleri
- List Türleri
- Union Türleri
- Pointer and Reference Türleri
- Tür Denetimi (Type Checking)
- Tür Eşdeğerliği (Equivalence)
- Teori and Veri Türleri

Giriş

- Bir veri türü (data type), bir veri nesneleri koleksiyonunu ve bu nesneler üzerinde önceden tanımlanmış bir dizi işlemi tanımlar
- Bir tanımlayıcı (descriptor), bir değişkenin özniteliklerinin koleksiyonudur
- Bir nesne (object), kullanıcı tanımlı (soyut veri) türünün bir örneğini temsil eder
- Tüm veri türleri için bir tasarım sorunu: Hangi işlemler tanımlanır ve nasıl belirlenir?

İlkel Veri Türleri (Primitive Data Types)

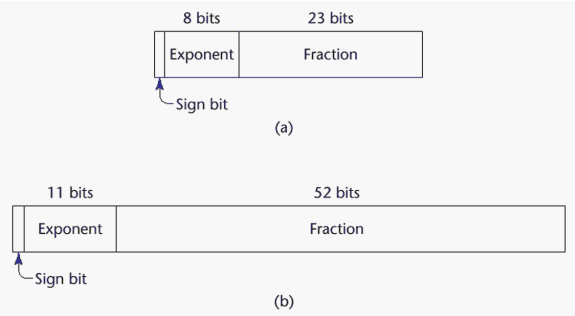
- Hemen hemen tüm programlama dilleri bir dizi ilkel veri türü sağlar
- İlkel veri türleri: Diğer veri türleri açısından tanımlanmamış olanlar
- Bazı ilkel veri türleri yalnızca donanımın yansımalarıdır
- Diğerleri, uygulamaları için yalnızca biraz donanım dışı destek gerektirir

İlkel Veri Türleri
Integer

- Neredeyse her zaman donanımın tam bir yansımasıdır
- Bir dilde sekiz farklı tam sayı türü olabilir
- Java'nın signed (işaretli) tam sayı tipleri: byte, short, int, long

İlkel Veri Türleri
Floating Point (Kayan Noktalı)

- Real sayıları modellemesi
- Bilimsel kullanım için diller en az iki kayan nokta türünü destekler (ör. float ve double; bazen daha fazla)
- Genellikle tam olarak donanım gibidir, ancak her zaman değil
  - IEEE Kayan Nokta Standart 754



İlkel Veri Türleri
Complex

- Bazı diller complex bir türü destekler, örneğin, Fortran ve Python
- Her değer iki kayan noktadan oluşur, real kısım ve hayali kısım
- Değişmez biçim (Python'da):
  - (7 + 3j), burada 7 real kısım ve 3 hayali kısımdır

İlkel Veri Türleri
Decimal

- İş uygulamaları için (para)
  - COBOL için gerekli
  - C# bir decimal (ondalık) veri türü sunar
- Kodlanmış biçimde (BCD) sabit sayıda ondalık basamak saklar
- Avantaj: doğruluk
- Dezavantajları: sınırlı uzunluk, hafızayı boşa harcar

İlkel Veri Türleri
Boolean

- Hepsinden daha basit
- Değer aralığı: iki öğesi vardır: biri **true (doğru)** ve biri **false (yanlış)**
  - Bazı programlama dillerinde false ve true yerine 0 ve 1 kullanılabilir.
- Avantaj: okunabilirlik

İlkel Veri Türleri
Character

- Sayısal kodlama olarak saklanır
- En yaygın kullanılan kodlama: ASCII
- Alternatif, 16 bit kodlama: Unicode (UCS-2)
  - En doğal dillerden karakterler içerir
  - Başlangıçta Java'da kullanıldı
  - C# ve JavaScript ayrıca Unicode'u destekler
- 32 bit Unicode (UCS-4)
  - 2003'ten itibaren Fortran tarafından desteklenmektedir

Character : String

- Değerler karakter dizileridir
- Tasarım Sorunları:
  - İlkel bir tür mü yoksa sadece özel bir dizi(array) mi?
  - String'lerin uzunluğu statik mi yoksa dinamik mi olmalıdır?

String
Tipik İşlemler

- Atama ve kopyalama
- Karşılaştırma (=,>, vb.)
- Birleştirme
- Alt string (substring) referansı
- Desen eşleştirme (pattern matching)

String
Farklı Programlama Dillerinde

- C ve C ++
  - İlkel değil
  - Char arrays ve işlemleri sağlayan fonksiyonların kitaplığını kullanır
- SNOBOL4 (bir string işleme dili)
  - İlkel
  - Ayrıntılı desen eşleştirme dahil birçok işlem
- Fortran ve Python
  - Atama ve birkaç işlem içeren ilkel tür
- Java
  - String sınıfı aracılığıyla ilkel
- Perl, JavaScript, Ruby ve PHP
  - Desen eşleştirmesi regular expression'lar kullanarak sağlar

String
Length (Uzunluk) Seçeneği

- Statik: COBOL, Java'nın String sınıfı
- Sınırlı Dinamik Uzunluk: C ve C ++
  - Bu dillerde, uzunluğu korumak yerine bir dizenin karakterlerinin sonunu belirtmek için özel bir karakter kullanılır.
- Dinamik (maksimum yok): SNOBOL4, Perl, JavaScript

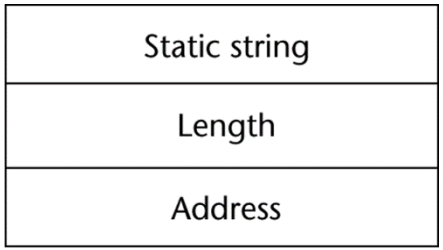
String Değerlendirmesi

- Yazılabilirliğe yardım
- Statik uzunluğa sahip ilkel bir tür olarak, sağlamak ucuzdur - neden olmasın?
- Dinamik uzunluk güzel, ancak masrafa değer mi?

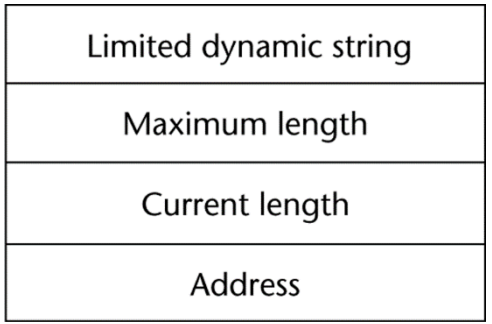
String Uygulaması

- Statik uzunluk: derleme (compile) zamanı tanımlayıcısı
- Sınırlı dinamik uzunluk: uzunluk için bir çalışma zamanı (run-time) tanımlayıcısına ihtiyaç duyabilir (ancak C ve C ++ 'da değil)
- Dinamik uzunluk: çalışma zamanı (run-time) tanımlayıcısına ihtiyaç duyar; Tahsis / tekrar tahsis (allocation ve deallocation) en büyük uygulama sorunudur

String
Derleme ve Çalışma Zamanında Tanımlama



Statik string'ler için derleme zamanında tanımlama



Sınırlı dinamik string'ler için Çalışma Zamanında Tanımlama

Kullanıcı Tanımlı Ordinal Türleri

- Ordinal tür, olası değerler aralığının pozitif tamsayılar kümesiyle kolayca ilişkilendirilebildiği türdür.
- Java'daki ilkel ordinal türleri örnekleri
  - integer
  - char
  - boolean

Enumeration (Numaralandırma) Türü

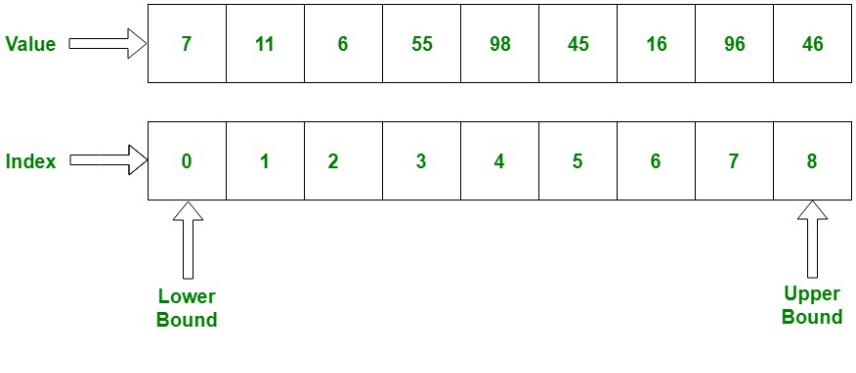
- Sabit olarak adlandırılan tüm olası değerler tanımda verilir.
  - C# örneği
- Enum days {mon, tue, wed, thu, fri, sat, sun};
- Tasarım Sorunları:
  - Bir enumeration sabitinin birden fazla tür tanımında görünmesine izin veriliyor mu ve eğer öyleyse, bu sabitin oluşumunun türü nasıl kontrol edilir?
  - enumeration değerleri tam sayıya mı zorlanıyor?
  - enumeration türüne zorlanan başka bir tür var mı?

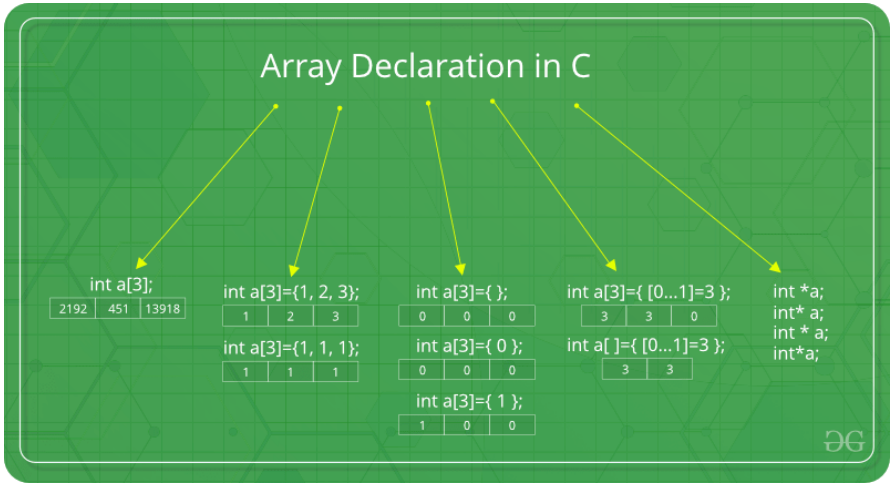
Enumeration Değerlendirilmesi

- Okunabilirliğe yardım eder, örneğin bir rengi sayı olarak kodlamaya gerek yoktur
- Güvenilirlik yardımı, ör. Derleyici şunları kontrol edebilir:
  - işlemler (renklerin eklenmesine izin vermeyin)
  - Hiçbir enumeration değişkenine tanımlı aralığı dışında bir değer atanamaz
  - C# ve Java 5.0, enumeration için C++'dan daha iyi destek sağlar çünkü bu dillerdeki enumeration türü değişkenleri tamsayı türlerine zorlanmamaktadır

Array Türü

- Bir array, tek bir öğenin kümedeki konumu ile tanımlandığı homojen bir veri öğeleri birleşimidir.





Array Tasarım Sırasında Karşılaşılabilecek Konular

- Array için hangi veri türlerine destek veriyor?
- array elemanlarının indeks aralığı dışına çıkınca ne oluyor?
- Tahsis ne zaman gerçekleşir?
- Düzensiz veya çok boyutlu dizilere veya her ikisine de izin veriliyor mu?
- Bir array farklı veri türlerine sahip olabilir mi?
- Dizi nesneler üzerinden başlatılabilir mi?
- Herhangi bir tür dilim destekleniyor mu? (alt diziler oluşturmak için)

Array İndeksleri

- Indexing (Indeksleme) (veya subscripting) indeks ile öğe arasında bir haritalamadır.
  - array_name (index_value_list) → an element (Öğe)
- Index Dizimi
- Fortran and Ada parantez kullanır
  - Ada, array referansları ve fonksiyon çağrıları arasındaki tekdüzeliği göstermek için açıkça parantez kullanır çünkü her ikisi de eşlemedir.
- Birçok programlama dili köşeli parantezi kullanır.

Array
İndeks Türleri

- FORTRAN, C: sadece integer
- Java: sadece integer türleri
- Indeks aralık kontrolü
  - C, C++, Perl, and Fortran aralık kontrolü yapmaz
  - Java, ML, C# aralık kontrolü yapar

Indeks Bağlama ve Array Kategorileri

- Statik: indeks aralıkları statik olarak bağlıdır ve depolama tahsisi statiktir (çalışma zamanından önce)
  - Avantaj: verimlilik (dinamik ayırma yok)
- Fixed stack-dynamic: indeks aralıkları statik olarak bağlıdır, ancak tahsis tanımlama zamanında (declaration time) yapılır
  - Avantaj: alan verimliliği
- Fixed heap-dynamic: Fixed stack-dynamic 'e benzer: depolama bağlama dinamiktir ancak tahsisten sonra sabittir (yani bağlama istendiğinde yapılır ve depolama stack'ten değil, heap'ten tahsis edilir)
- Heap-dynamic: indeks aralıklarının bağlanması ve depolama tahsisi dinamiktir ve herhangi bir sayıda değiştirilebilir
  - Avantaj: esneklik (diziler program yürütülürken büyüyebilir veya küçülebilir)

Indeks Bağlama ve Array Kategorileri Programlama Dili Örnekleri

- C and C++ array'ler, static anahtar kelimesi ile static'tir.
- C and C++ array'ler, static anahtar kelimesi yoksa fixed stack-dynamic'tir.
- C and C++, fxied heap-dynamic array'leri sağlar
- C# ArrayList sınıfı fixed heap-dynamic'tir
- Perl, JavaScript, Python, and Ruby heap-dynamic array'leri destekler

Array Başlangıç Değerleri Atama

- Bazı diller, depolama tahsisi sırasında başlangıç değeri atamaya izin verir
  - C, C++, Java, C# example
    int list [] = {4, 5, 7, 83}
  - C and C++'de character string
    char name [] = "freddie";
  - C and C++ array string'leri
    char *names [] = {"Bob", "Jake", "Joe"];
  - Java string nesnelerine atama
    String[] names = {"Bob", "Jake", "Joe"};

Heterojen Array'ler

- Heterojen bir array, öğelerin aynı türde olması gerekmeyen dizidir
- Perl, Python, JavaScript ve Ruby tarafından desteklenir

Array Başlangıç Değeri Atama
List karşılaştırması (Python)

- C-based languages
  - int list [] = {1, 3, 5, 7}
  - char *names [] = {"Mike", "Fred", "Mary Lou"};
- Python
  - List örneği

```
list = [x ** 2 for x in range(12) if x % 3 == 0]
```

list çıktısı [0, 9, 36, 81]

Array İşlemleri

- APL, vektörler ve matrisler ile tekli operatörler için en güçlü dizi işleme işlemlerini sağlar (örneğin, sütun öğelerini tersine çevirmek için)
- Python'un dizi atamaları, ancak bunlar yalnızca referans değişiklikleridir. Python ayrıca dizi birleştirme ve öğe üyelik işlemlerini de destekler
- Ruby ayrıca dizi birleştirme sağlar

Rectangular and Jagged Array'ler

- Bir Rectangular Array, tüm satırların aynı sayıda öğeye sahip olduğu ve tüm sütunların aynı sayıda öğeye sahip olduğu çok boyutlu bir dizidir.
- Bir jagged matris, değişen sayıda öğeye sahip satırlara sahiptir
  - Çok boyutlu diziler aslında dizi dizileri olarak görmek mümkündür
- C, C ++ ve Java jagged array'leri destekler
- F# ve C# rectangular ve jagged array'leri destekler

Slices (Dilimler)

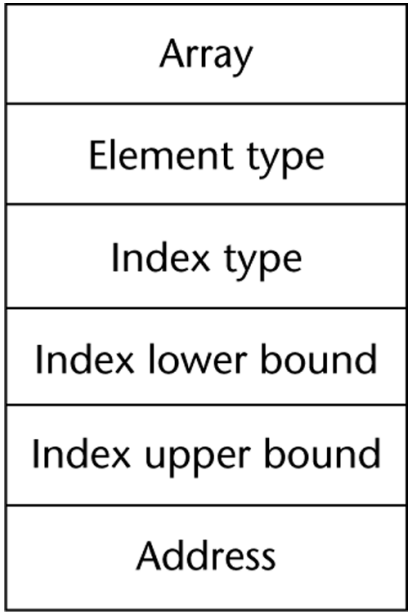
- Bir slice, bir dizinin bazı alt dizisidir
- Dilimler yalnızca dizi işlemlerine sahip dillerde kullanışlıdır
- Python

```
vector = [2, 4, 6, 8, 10, 12, 14, 16]
mat = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

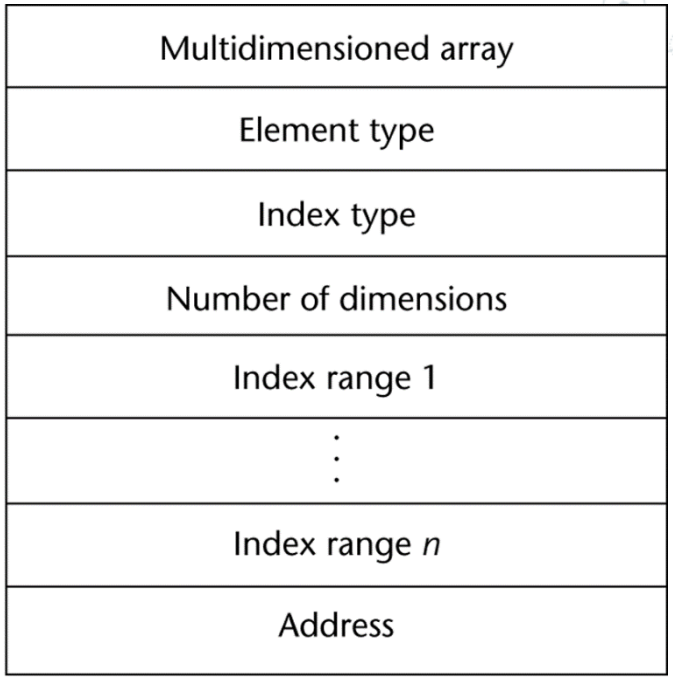
`vector (3:6)` öğelik bir dizi döndür
`mat[0][0:2]` birinci satırdaki ilk ve ikinci öğeyi döndürür

- Ruby'de bu işlemler için slice metodu vardır
  `list.slice(2, 2)` slice metodu üç ve dördüncü öğeyi döndürür

Derleme Zamanı (Compile -Time) Tanımlayıcıları



Tek Boyutlu Array



Çok boyutlu array

İlişkisel Diziler (Associative Arrays)

- Bir ilişkisel dizi, anahtar adı verilen eşit sayıda değerle indekslenen, sırasız veri öğeleri koleksiyonudur.
  - Kullanıcı tanımlı anahtarlar saklanmalıdır
- Tasarım Zorlukları:
  - Elemanlara yapılan referansın şekli nedir?
  - Boyut statik mi yoksa dinamik mi?
- Perl (Hash), Python (Dictionary), Ruby (Hash) ve Lua'da (Table) yerleşik tür bulunur.

İlişkisel Diziler
Perl Örneği

- İsimler % ile başlar ve veriler parantezler (…) arasında bulunur

```
%hi_temps = ("Mon" => 77, "Tue" => 79, "Wed" => 65, …);
```

- Indeks üzerinden ulaşım için $, süslü parantez ve anahtar değer kullanılır

```
$hi_temps{"Wed"} = 83;
```

- Öğeler anahtar üzerinden delete ile silinebilir

```
delete $hi_temps{"Tue"};
```

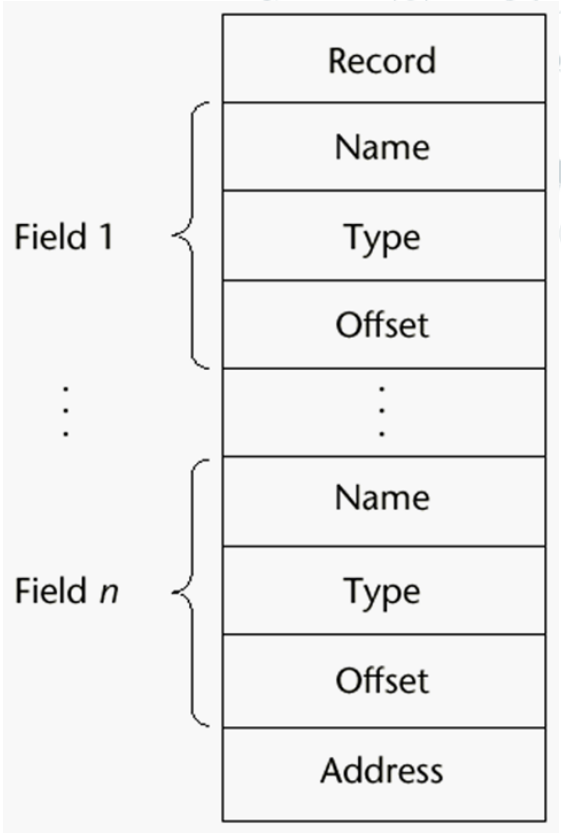
Record Türleri

- Bir record (kayıt), tek tek öğelerin adlarla tanımlandığı, muhtemelen heterojen bir veri öğeleri toplamıdır.
- Tasarım Zorlukları:
  - Alana yapılan sözdizimsel referans formu nedir?
  - Eliptik referanslara izin veriliyor mu?

Record
Cobol

- COBOL, iç içe geçmiş kayıtları göstermek için düzey numaralarını kullanır; diğerleri yinelemeli tanım kullanır

```
01 EMP-REC.
   02 EMP-NAME.
      05 FIRST PIC X(20).
      05 MID   PIC X(10).
      05 LAST  PIC X(20).
   02 HOURLY-RATE PIC
  99V99.
```



Array ve Record
Değerlendirme ve Karşılaştırma

- Veri değerlerinin toplanması heterojen olduğunda record kullanılır
- Array öğelerine erişim, record alanlarına erişimden çok daha yavaştır, çünkü array indekslemesi dinamiktir (record içindeki alan adları statiktir)
- Dinamik indeksler record alanı erişimiyle kullanılabilir, ancak tür kontrolüne izin vermez ve çok daha yavaş olur

Tuple Türleri

- Bir tuple, öğelerin adlandırılmaması dışında bir kayda benzer bir veri türüdür.
- Python, ML ve F#'da fonksiyonların birden çok değer döndürmesine izin vermek için kullanılır
- Python
  - Listeleriyle yakından ilişkili, ancak değişmez (readonly)
  - Örnek
    myTuple = (3, 5.8, ′ elma ′)
    Indeks 1 ile başlar, iki tüple birleştirilebilir, del ile silinebilir

Tuple Türleri
Örnekler

- ML

```
val myTuple = (3, 5.8, 'apple');
```

- Erişim:

`#1(myTuple)` :birinci elemana erişiö

- Yeni bir tüple türü tanımlanabilir

```
type intReal = int * real;
```

- F#

```
let tup = (3, 5, 7)
let a, b, c = tup
```

Tuple değerleri 3 değişkene aktarıldı

List Türleri

- Lisp ve Scheme'deki listeler parantezlerle sınırlandırılmıştır ve virgül kullanmaz.
  (A B C D) ve (A (B C) D)
- Veri ve kod aynı biçime sahiptir
  Veri olarak, (A B C) tam anlamıyla ne ise Kod olarak,
  (A B C), A fonksiyonu B ve C parametreleri temsil eder
- Yorumlayıcı, listenin hangisi olduğunu bilmesi gerekir, bu nedenle veri ise, bir kesme işaretiyle alıntı yaparız
  ' (A B C)
  veri anlamına gelir

List Türleri
Scheme

- Scheme'daki İşlemleri Listeleme
  - CAR, liste parametresinin ilk öğesini döndürür
    (CAR ′ (A B C)), A değerini döndürür
  - CDR, ilk öğe kaldırıldıktan sonra liste parametresinin kalanını döndürür
    (CDR ′ (A B C)), (B C) döndürür
  - CONS, yeni bir liste oluşturmak için ilk parametresini ikinci parametresi olan bir listeye koyar
    (CONS ′ A (B C)), (A B C) döndürür
  - LIST, parametrelerinin yeni bir listesini döndürür
    (LIST ′ A ′ B ′ (C D)) şunu döndürür: (A B (C D))

List Türleri
ML ve F#

- ML’de List işlemleri
  - Listeler parantez içinde yazılır ve öğeler virgülle ayrılır
  - Liste öğeleri aynı türde olmalıdır
  - Scheme CONS fonksiyonu, ML'de bir ikili operatördür, ::
    3 :: [5, 7, 9], [3, 5, 7, 9] olarak değerlendirilir
  - Scheme CAR ve CDR işlevleri sırasıyla hd ve tl olarak adlandırılır.
- F#’da List
  - ML’deki gibi, öğelerin noktalı virgülle ayrılması dışında ve hd ve tl, List sınıfının yöntemleridir.

Liste Türleri
Python

- Python List
  - List veri türü aynı zamanda Python dizileri olarak hizmet eder
  - Scheme, Common Lisp, ML ve F #'dan farklı olarak Python'un listeleri değiştirilebilir
  - Öğeler herhangi bir türde olabilir
  - Örnek:
    myList = [3, 5.8, "programlama"]
  - Liste öğelerine, sıfırdan başlayan indesklere başvurulur
    x = myList[1] x'e 5,8 aktarılır
  - Del ile liste öğeleri silinebilir
    del myList[1]
  - Bir Liste Oluşturma Tekniği (List Comprehension): küme gösteriminden türetilmiştir

```
[x * x for x in range(6) if x % 3 == 0]
```

Oluşturulan liste: [0, 9, 36]

List Türleri
Comprehension

- Haskell - List Comprehension
  [n * n | n <- [1..10]]
- F# - List Comprehensions
  let myArray = [|for i in 1 .. 5 -> [i * i) |]
- Hem C # hem de Java, sırasıyla heap-dynamic collection sınıfları olan List ve ArrayList aracılığıyla listeleri destekler

Unions Türleri

- Bir union, değişkenlerinin yürütme sırasında farklı zamanlarda farklı tür değerleri depolamasına izin verilen bir türdür
- Tasarım zorluğu
  - Veri türü kontrolü gerekli olmalı mı?

Discriminated vs. Free Unions

- C ve C ++, tür denetimi için dil desteğinin olmadığı union yapıları sağlar; bu dillerdeki birliğe «free union» denir
- Tür kontrolü için discriminant adı verilen bir tür göstergesi içermesini gerektirir
  - ML, Haskell ve F # tarafından desteklenir

Unions
F#

- Tanımlama

```
type intReal =
    | IntValue of int
    | RealValue of float;;
```

  - intReal yeni bir tür, IntValue ve RealValue içeriyor

- intReal'i oluşturmak için

```
let ir1 = IntValue 17;;
let ir2 = RealValue 3.4;;
```

Union Değerlendirmesi

- Free union güvensizdir
  - Tür kontrolüne izin verme
- Java ve C# union desteklemez
  - Programlama dilinde artan güvenlik endişelerini yansıtıyor

Pointer (İşaretci) ve Reference (Referans) Türleri

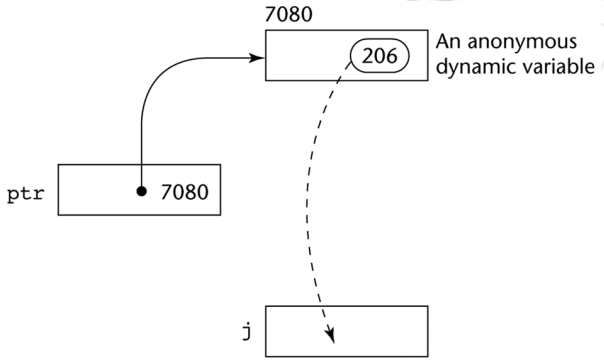
- Pointer türü bir değişken, bellek adreslerinden ve özel bir değerden oluşan bir değer aralığına sahiptir.
- Dolaylı adreslemenin gücünü sağlar
- Dinamik belleği yönetmenin bir yolunu sağlar
- Depolamanın dinamik olarak oluşturulduğu alandaki bir konuma erişmek için bir pointer kullanılabilir (genellikle heap olarak adlandırılır)

Pointer
Tasarım Zorlukları

- Bir pointer değişkeninin kapsamı ve ömrü nedir?
- Bir heap-dynamic değişkenin ömrü nedir?
- Pointer'lar, işaret edebilecekleri değer türü açısından sınırlı mı?
- Pointer'lar dinamik depolama yönetimi, dolaylı adresleme veya her ikisi için mi kullanılıyor?
- Programlama dillerinde pointer türlerini, referans türlerini veya her ikisini de desteklemeli mi?

Pointer İşlemleri

- İki temel işlem: atama ve pointer değeri
- Atama, bir pointer değişkeninin değerini bazı yararlı adreslere ayarlamak için kullanılır
- Pointer değeri içindeki değer, işaretçinin değeriyle temsil edilen konumda depolanan değeri verir
- Başvurudan çıkarma explicit or implicit olabilir
  C ++, * aracılığıyla explicit bir işlem kullanır
  j = *ptr
  j'ye ptr'de bulunan değeri atar



Pointer Problemleri

- Dangling Pointers (tehlikeli)
  - Bir pointer, ayrılan bir heap-dynamic değişkene işaret ediyor
- Heap-dynamic değişkeni kaybolması
  - Artık kullanıcı programı tarafından erişilemeyen ayrılmış bir heap-dynamic değişkeni (genellikle çöp olarak adlandırılır)
    - Pointer p1, yeni oluşturulan bir heap-dynamic değişkeni gösterecek şekilde ayarlandı
    - Pointer p1 daha sonra yeni oluşturulan başka bir heap-dynamic değişkeni gösterecek şekilde ayarlanır
    - Heap-dynamic değişkenleri kaybetme sürecine bellek sızıntısı (*memory leakage)* denir

Pointers
C and C++

- Son derece esnektir ancak dikkatli kullanılmalıdır
- Pointer, ne zaman ve nerede tahsis edildiğine bakılmaksızın herhangi bir değişkene işaret edebilir
- Dinamik depolama yönetimi ve adresleme için kullanılır
- İşaretçi aritmetiği mümkündür
- Etki alanı türünün sabitlenmesine gerek yoktur (void *)
  Void * herhangi bir türe işaret edebilir

Pointer Aritmetiği
C and C++

```
float stuff[100];
float *p;
p = stuff;
```

`*(p+5)` anlamı `stuff[5]` ve `p[5]`

`*(p+i)` almanı `stuff[i]` and `p[i]`

Reference Türleri

- C ++, öncelikle biçimsel parametreler için kullanılan, reference türü olarak adlandırılan özel bir tür pointer türü içerir.
  - Hem referans bazında hem de değer bazında geçişin avantajları
- Java, C++'ın reference değişkenlerini genişletir ve pointer'ların tamamen değiştirmelerine olanak tanır
  - Referanslar, adres olmaktan çok nesnelere yapılan referanslardır
- C#, hem Java referanslarını hem de C++ pointer'ları içerir
- Soru C# veya Java'da iki değişkenin yerlerini return yapmadan bir void fonksiyon üzerinden değiştirin.

  a=5; b=10;
  degistir(a, b)
  a=10; b=5; olsun

Pointer ve Reference Değerlendirmesi

- İşaretçiler goto'lar gibidir - bir değişkenle erişilebilen hücre aralığını genişletirler
- Dinamik veri yapıları için pointer’lar veya referanslar gereklidir - bu nedenle onlar olmadan bir programlama dili tasarlayamayız

Tür Denetimi (Type Checking)

- Alt programları ve atamaları içerecek şekilde operand ve operatör kavramını geneller
- Tür denetimi, bir operatörün operand'larına uyumlu türlerde olmasını sağlama etkinliğidir
- Bir uyumlu tür (compatible type), operatör için yasal olan veya dil kurallarına göre derleyici tarafından üretilen kod tarafından implicit olarak yasal bir türe dönüştürülmesine izin verilen türdür
  - Bu otomatik dönüşüme zorlama (coercion) denir
- Bir tür hatası (type error), bir operatörün uygun olmayan tipte bir operand'a uygulanmasıdır

Tür Denetimi (Type Checking)

- Tüm tür bağlamaları statikse, neredeyse tüm tür denetimleri statik olabilir
- Tür bağlamaları dinamikse, tür denetimi dinamik olmalıdır
- Tür hataları her zaman tespit edilirse, bir programlama dili güçlü bir tür yönetimine (Strong Typing) sahiptir.
  - Güçlü tür yönetiminin avantajı: tür hatalarıyla sonuçlanan değişkenlerin yanlış kullanımlarının tespitine izin verir

Güçlü Tür Denetimi (Strong Typing)

- C ve C++ değildir: parametre türü kontrolünden kaçınılabilir; union türü kontrol edilmez
- Java ve C#, neredeyse (açık tür çevirme nedeniyle) güçlü tür denetimine sahiptir.

Tür Eşdeğerliği (Type Equivalence)
Name Type Equivalence

- İsim türü eşdeğerliği, iki değişkenin aynı bildirimde veya aynı tür adını kullanan bildirimlerdeyse eşdeğer türlere sahip olduğu anlamına gelir.
- Uygulaması kolay ancak oldukça kısıtlayıcı:
  - Tam sayı türlerinin alt aralıkları, tam sayı türleriyle eşdeğer değildir (örneğin byte veri türü…)
  - Biçimsel parametreler, karşılık gelen gerçek parametreleriyle aynı türde olmalıdır

Tür Eşdeğerliği (Type Equivalence)
Structure Type Equivalence

- Yapı türü eşdeğerliliği, eğer türleri aynı yapıya sahipse, iki değişkenin eşdeğer türleri olduğu anlamına gelir
  - Daha esnek, ancak uygulaması daha zor

Teori ve Veri Türü

- Tür teorisi matematik, mantık, bilgisayar bilimi ve felsefede geniş bir çalışma alanıdır.
- Bilgisayar biliminde tür teorisinin iki dalı:
  - Practical (Pratik) - ticari programlama dillerdeki veri türleri
  - Abstract (Soyut) - lambda calculus türleri
- Bir tür sistemi, bir dizi türdür ve programlarda kullanımlarını yöneten kurallardır.