BLG339
PROGRAMLAMA DİLLERİ KAVRAMI

Hafta 5
Veri Tipleri

Yrd. Doç. Dr. Melike Şah Direkoğlu

Alındığı kaynak:
Addison-Wesley's Programming Language Concepts slaytları ve Prof. Dr. Tuğrul Yılmaz' ın ders notlarından faydalanarak hazırlanmıştır.

Konular

- Giriş
- İlkel/Temel Veri Tipleri
- Karakter String Tipleri
- Kullanıcı Tanımlı Tipler
- Dizi Tipleri
- Birleşmiş Diziler
- Kayıt Tipleri
- Küme Tipleri
- İşaretçi ve Referans Tipleri

Giriş

- Bir veri tipi, bir değerler kümesini ve bu değerler üzerindeki işlemleri tanımlar.
  - Bilgisayarlar sonuçları veriyi işleyerek sunar.
- Değişkenleri özellikleri olan bir tanımlayıcı (descriptor) kolleksiyonu olarak düşünübiliriz.
  - değişken statik ise, tanımlayıcı özellikleri dereleme sırasında gereklidir
  - veya değişken dinamik ise, tanımlayıcı özellikleri çalışma sırasında gereklidir
- Tüm veri tipleri üzerinde tasarımda düşünülen temel problem "bu veri tipleriyle hangi işlemler nasıl yapılacaktır?" şeklindedir.

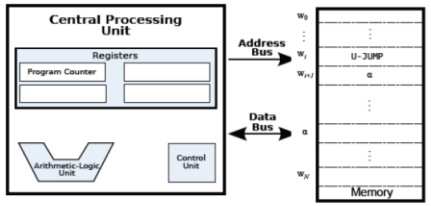
Giriş

- Bir programlama dilindeki veri tipleri,
  - programlardaki ifade yeteneğini,
  - gerçek hayattaki problemlere nasıl uygulanacagını
  - programların güvenilirliğini doğrudan etkiledikleri için, bir programlama dilinin değerlendirilmesinde önemli bir yer tutarlar.

Giriş – Bilgisayarda Veri (Data) Nedir?

- Bir problemin çözümü esnasında bilgisayarda tutulan, CPU komutu olmayan her türlü bilgiye veri denir.
- Von Neumann mimarisi CPU ile belleği belirgin bir şekilde ayırır.
- Bellek içeriği oldukça karışıktır. Bellekte:
  - Her türlü CPU komutları: yazmaç (register), bellek transferleri, aritmetik işlemler, karşılaştırma, vs.
  - Bazı işlemleri yapabilmek için ek bilgi: transfer etmek için adresler, vs.
  - İşlenecek değerler, adres bilgileri gibi **veriler (data)**.
- Belleğe erişim tamamen adrese göredir.
- Von Neumann makinesinde işlenecek bütün bilgiler ikili (binary) sayısal sisteme dönüştürülmek zorundadır.

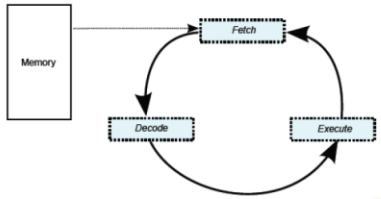
Giriş – Veri Nedir?



Von Neumann mimarisi

```
main:
        pushq   %rbp
        movq    %rsp, %rbp
        movl    alice(%rip), %edx
        movl    bob(%rip), %eax
        imull   %edx, %eax
        movl    %eax, carol(%rip)
        movl    $0, %eax
        leave
        ret
alice:
        .long   123
bob:
        .long   456
```

Çevirici dili (assembly language)
İki ayrı bellek noktasındaki tam sayıları çarpıp sonucu başka bir bellek noktasına koyan program



Getir (Fetch), kodu çöz (decode), yap (execute)

```
01010101 01001000 10001001 11101101 10001011 00010101 10110010 00000011
00100000 00000000 10001011 00000101 10100000 00000011 00100000 00000000
00001111 10101111 11000010 10001001 00000101 00111011 00000011 00100000
00000000 10111000 00000000 00000000 00000000 00000000 11001001 11000011
...
11001000 00000001 00000000 00000000  00000000 00000000
```

Yukarıdaki programın ikili sayı sisteminde bellekteki durumu.

Veri tipleri ikiye ayrılır:
İlkel/temel veri tipleri
Yapısal veri tipleri

İlkel Veri Tipleri (Primitive Data Types)

- Neredeyse tüm programlama dilleri ilkel veri tiplerini sağlamaktadır.
- Başka veri tipleri aracılığıyla tanımlanmayan veri tiplerine ilkel/temel (primitive) veri tipleri denir.
- Bazı veri tipleri donanımın bir yansımasıdır. Diğerleri donanım harici desteğe ihtiyaç duyabilir.
- Integer sayilar, donanıma bağlıdır. Gerçekleştirimi limitlidir (4 – 8 bayt gibi).

İlkel Veri Tipleri(Primitive Data Types)

- Önceleri programlama dillerinde sadece sayısal temel veri tipleri tanımlanmışken, günümüzde karakter, mantıksal, karakter dizgi, kullanıcı tanımlı sıralı tipler gibi çeşitli temel veri tipleri bulunmaktadır.

Tamsayı (Integer)

- Hemen herzaman CPU özelliklerine göredir.
- Bir dilde 8 farklı tamsayı olabilir.
  - Java: byte, short, int, long
  - C, C#, C++ (ek olarak) : unsigned int (sadece pozitif tamsayılar).
  - Işaret biti genellikle en solda temsil edilir.

İşaret ve büyüklük
sign and magnitude

İkili (binary)	signed	unsigned
00000000	0	0
00000001	1	1
...	...	...
01111111	127	127
10000000	−0	128
...	...	...
11111111	−127	255

IBM 7090 gibi ilk bilgisayarlarda kullanılmıştır.

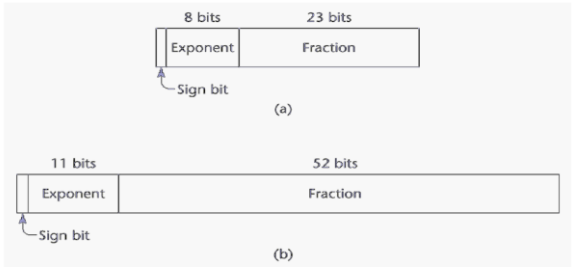
Kayan Noktalı Sayılar (Floating Point)

- Kayan noktalı (*floating point*) veri tipleri, gerçel sayıları modellerler.
- Modelleme yaklaşıktır (estimation). Örneğin π veya *e* sayıları gösterilemez.
- Bilimsel hesaplamalarda kullanılan dillerde en az iki noktalı sayı destekler(örn., float ve double)
- Genellikle tamamen donanım (CPU) ile uyumludur.
  - IEEE Floating-Point Standard 754

IEEE Kayar Noktalı Formatı

- Kayan noktalı sayılar, kesir (fraction) ve üst (exponent) olarak iki bölümde ifade edilirler.
- Aşağıdaki şekilde görüldüğü gibi kayan noktalı veri tipi, gerçel (real) ve çift-duyarlılık (double) olmak üzere iki tiple gösterilebilirler.

$$\left(1+\sum_{n=1}^{23} \text{bit}_{[23-n]} \times 2^{-n}\right) \times 2^{\text{exponent}-127}$$



Ondalık (Decimal, BCD)

- İş uygulamaları için (para) kullanılır.
- Bu veri tipi, az sayıda programlama dilinde vardır: Örneğin; PL/I, Cobol, C#
- Sabit sayıda on tabanlı karakterleri saklar (kodlanmış – BCD binary coded decimal)
- Avantaj: Kesinlik
- Dezavantaj: Kapsama sınırlıdır, çok bellek kullanılır. Onlu veri tipi, onlu değerleri tam olarak saklayabilirse de, üsler bulunmadığı için gösterilebilecek değer alanı sınırlıdır. Her basamak için bir sekizli (byte) gerekli olması nedeniyle, belleği etkin olarak kullanmaz.
- İşlemler CPU desteği varsa CPU tarafından yapılır (Intel de yok), yoksa yazılımla benzetimlenir (simulate).

Mantıksal (Boolean)

- Hepsinin en basitidir.
- Değerler: iki eleman, biri doğru“true” ve diğeri yanlış “false”
- Bitler ile gerçeklenebilirler, bununla birlikte genellikle byte lar (sekizliler) ile gerçeklenirler (saklanırlar).
- Avantaj: Okunabilirlik(readability)

Karakter (Character)

- Karakter veri tipi, tek bir karakterlik bilgi saklayabilen ve bilgisayarlarda sayısal kodlamalar olarak saklanan bir veri tipidir.
- En yaygın sayısal kodlama: ASCII
  - 128 farklı karakteri göstermek için, 0..127 arasındaki tamsayı değerleri kullanır.
- Alternatif, 16-bit kodlama: Unicode
  - Tüm doğal dillerin karakterlerini barındırır.
  - Burada karakterler 1-4 bayt ile gösterilirler.
  - Java, C# ve JavaScript de Unicode u destekliyor.

Karakter Dizgi Tipi (Character String Types)

- Bir **karakter dizgi** veri tipinde, nesneler karakterler dizisi olarak bulunur.
- Karakter dizgi veri tipi bazı programlama dillerinde temel bir veri tipi olarak (dolayısıyla dizilim tipi indeksli kullanım yok), bazılarında ise özel bir karakter dizilimi olarak yer almıştır.
- Önemli tasarım özelliklerinden birisi de boyunun statik veya dinamik mi olmasıdır.

Karakter String Tipi

- İşlemler:
  - Atama, tanımlama (char *str = “özellikler”;)
  - Karşılaştırma (=, >, strcmp, vs.)
  - Birleştirme
  - Alt dizgiye erişim
  - Örüntülü eşleme (Pattern matching)

Belirli Dillerde Karakter String Tipleri

- C and C++
  - İlkel değildir
  - char dizileri kullanılır ve işlemler için kütüphane fonksiyonları vardır (strcpy(src,dst)).
- SNOBOL4 (bir string işleme dili)
  - İlkel veri tipidir. Birçok işlem hazır vardır.
- Java
  - String sınıfıyla ilkeldir.
  - statik dizgi nesneleri yaratır. Nesneler değiştirilemez.
  - StringBuffer sınıfı değiştirilebilir dizgi nesnelerinin sınıfıdır.
- Ada, FORTRAN 90, and BASIC
  - Temel veri.
  - Atama, karşılaştırma, birleştirme, alt dizgiye erişim
  - FORTRAN da örüntülü eşleme var.
- Perl, JavaScript, C++, C#
  - Patterns düzenli ifadeler (regular expressions) ile tanımlanır. Çok güçlü bir özellik
  - Örneğin : /[A-Za-z][A-Za-z\d]+/ veya /\d+\.?\d*|\.\d+/

Karakter String Uzunluk(Length) Seçenekleri

1. Statik - FORTRAN 77, Ada, COBOL
   - Örneğin. (FORTRAN 90)
   - **CHARACTER (LEN = 15) NAME;**

2. Limitli Dinamik Uzunluk - C ve C++ da geçek boy sondaki null karakterinden anlaşılır.

3. Dinamik - SNOBOL4, Perl, JavaScript

4. Ada her üç tip dizgiyi de ayrı ayrı destekler.

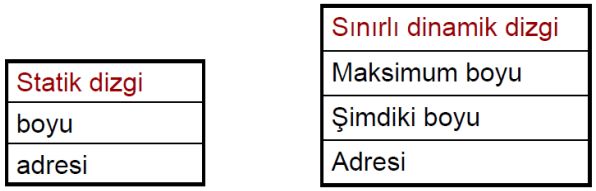
Karakter String Tipi Değerlendirme

- Yazabilmek için araç.
- Sabit boylu temel tip olarak desteklenmesi kolay.
- Dinamik boy güzel ancak harcanan kaynaklara değer mi?
  - Dinamik String genelde yorumlanan dillerde tercih ediliyorlar.

Karakter String Tipi Gerçekleştirim (implementation)

- Statik boy – derleme zamanı niteleyici/betimleyici.
- Sınırlı dinamik boy – yürütme zamanı betimleyici gerekli olabilir (fakat C ve C++ da yok.)
- Dinamik boy – Yürütme zamanı betimleyici gerekir. Bellekten yer alma ve geri verme en zor uygulama problemi.

Karakter String Tipi Gerçekleştirim



Kullanıcı Tanımlı Sıralı Tipler

Kullanıcı Tanımlı Sıralı Tipler (User-Defined Ordinal Types)

- Bir sıralı (ordinal) tip, olası değerlerin pozitif tamsayılar kümesi ile ilişkilendirilebildiği veri tipidir.
- Bir çok programlama dilinde kullanıcılar, **sayılama** (*enumeration*) ve **altalan** (*subrange*) olmak üzere iki tür sıralı tip tanımlayabilir. Bu tip tanımlarındaki amaç, programcılara modellenen gerçek dünya nesnelerine karşı gelebilecek yeni tipler oluşturma olanağı sağlamaktır.
- Örnekler:
  - C++: enum renkler {kirmizi, mavi, yesil, sari, siyah};
  - C: typedef enum {kirmizi, mavi, yesil, sari, siyah} renkler;

Sayılama (Enumeration) Tipi

- Sayılama (enumeration) tipi, gerçek hayattaki verilerin tamsayı (*integer*) veri tipine eşleştirilmesi için kullanılan veri tipidir.
- C# örneği
  - enum days {mon, tue, wed, thu, fri, sat, sun};
- Tasarım Konuları
  - Tanımlanan sayma tipi başka yerde tekrar tanımlanıp kullanılabilir mi?
  - Sayılama tipleri (enumeration) değeleri sadece integer mi?

Sayılama Tiplerinin Değerlendirilmesi

- Okunabilirliğe yardım eder. örn., rengi sayı olarak kodlamaya gerek yoktur.
- Güvenilirliğe yardım eder. örn., derleyici şunları kontrol edebilir.
  - İşlemleri (renklerin toplanması izin vermez)
  - Tanımlandığı aralık dığında değerler atanamaz
  - Ada, C# ve Java 5.0; C++ dan daha iyi sayma tip idesteği verir.

Alt Aralık (Subrange) Tipi

- Alt aralık (subrange) tipi, kullanıcı tanımlı sıralı tiplerin bir alt grubudur.
- Bir alt aralık tipinin tanımındaki değerler, daha önceden tanımlanmış veya dilde tanımlı olan (*built-in*) sıralı tiplerle ilişkilendirir. Böylece, yeni tanımlanan tip ile alt grubu olduğu ana sınıf arasında bağ kurulur.
  - Altalan tipinin ana sınıfına uygulanabilen tüm işlemciler, altalan tiplerine de uygulanabilmektedir.
- Tasarım sorusu: nasıl kullanılabilirler?
  - Pascal'da: **type pos = 0 .. MAXINT;**

Alt Aralık Tipi Değerlendirilmesi

- Okunabilirliği arttırır.
- Güvenilirlik – sınırlandırılmış değerler hata kontrolünü sağlar.

Kullanıcı Tanımlı Tiplerin Gerçekleştirimi

- Sayma tipleri(Enumeration types) tamsayılar olarak gerçekleştirilirler.
- Aralık tipleri derleyici tarafından bazı şartlar eklenerek ebeveyn tip gibi gerçekleştirilirler.