YOZGAT BOZOK ÜNİVERSİTESİ

# Programlama Dilleri Prensipleri

Dr. Öğr. Üyesi R. Sinan ARSLAN

# Giriş

- ilkel veri tipleri
- türetilmiş veri tipleri
- diziler
- gösterge tipi
- bellek bölgeleme
- kuvvetli tipleme

# Veri tipleri ve yapıları

- bir verinin bellekte nasıl tutulacağını, değerinin nasıl yorumlanacağını ve veri üzerinde hangi işlemleri yapılabileceğini belirleyen bilgiye veri tipi denir.
- Programlama dilleri için veri tipi önemlidir.
- Programlama dilini ifade gücü ve güvenliliğini etkiler.
- Veri tipleri, basit(primitive) ve türetilmiş(user defined) veri tipleri olarak iki gruba ayrılır.

# İlkel Veri tipleri

- dilin tasarımında kararlaştırılmış olup, dilin kurallarına göre varlığı kesin olan türlerdir.
- Her programlama dilinde ilkel veri tipi grubu vardır.
- Örnek olarak C dilinde 11 farklı veri tipi vardır.
- Diller arasında bazı veri tipleri için temel farklar olsa da benzer veri tipleri bulunur.
- C/C++ dilindeki veri tipleri
  - enum(16 bit), unsigned int(16 bit), short int(16 bit), int(16 bit), unsigned long(32 bit), long(32 bit), float(32 bit), double(64 bit), long double(80 bit), unsigned char(8 bit), char(8 bit)
  - unsigned olanlar 0 dan başlarlar. negatif değer alamazlar.
  - veri tiplerinin alabileceği değer sınırları $-2^{bitsayisi}/2$ ile $+2^{bitsayisi}/2$-1 aralığındadır.
- Sayısal veri tiplerinin yanında, karakter, mantıksal, karakter dizisi, kullanıcı tanımlı tipler de popüler dillerin çoğunda bulunur.

# Sayısal Veri Tipleri

- Tamsayı
  - tamsayı değer alır.
  - bellekte en sol bit işaret biti olmak üzere bir dizi ikili bit ile gösterilir.
  - C dilinde 4 tamsayı türü ve unsigned/signed olanlar ile birlikte 8 adettir.
  - unsigned sadece pozitif sayıları, signed olanlar hem negatif hem pozitif sayıları alabilirler.
  - int: tamsayı, long: tamsayıdır, uzun sayıları tutar. gerekli hallerde alt çizgi ile ayrılarak ta gösterilebilir. int id = 1234_5678_L; unsigned: işaretsiz değerler alır. char, int ve long türlerinde uygulanabilir. Java da unsigned değişken tanımlaması yoktur.
  - Ada, C dillerinde bu temel tiplerin yanında short int, int , long int gibi tip tanımları da mümkündür.

# Sayısal Veri Tipleri

- Kayan Noktalı Sayı:
  - reel sayıları modellmek için kayan noktalı(floating point) veri tipleri tanımlanır.
  - Kesir ve üs olarak iki bölümde ifade edilebilirler.
  - Kayan noktalı sayıların tanımlanmasında duyarlılık(precision) ve alan(range) terimleri vardır. duyarlılık değerin kesin bölümünün tamlığını, alan ise kesir ve üstlerin birleşmesini ifade eder.
  - float ve double veri türleri duyarlılıkları farklı olan iki veri tipidir.
  - double float a göre daha hassasiyet yüksektir.

```
double a =10;
double b = 3;
printf(a/b) ->
ÇIKTI : 3.333333333
```

```
float a =10;
float b = 3;
printf(a/b) ->
ÇIKTI : 3.3333332539
```

# Sayısal Veri Tipleri

- Onlu(decimal):
  - ticari işletmelerde kullanılırlar.
  - Ondalık noktanın sabit bir yerde bulunduğu sabit sayıda onlu basamak içeren bir veri tipidir.
  - decimal tipler BCD(binary coded decimal) olarak bellekte saklanacağından bellek açısından verimli değillerdir.
  - Java da yoktur ancak c# destekler.
  - decimal türü double türüne göre daha hassastır.

```
double a =10;
double b = 3;
printf(a/b) ->
ÇIKTI : 3.333333333
```

```
float a =10;
float b = 3;
printf(a/b) ->
ÇIKTI : 3.33333
```

```
decimal a =10;
decimal b = 3;
printf(a/b) ->
ÇIKTI : 3.3333333333333333
```

# Sayısal Veri Tipleri

- Mantıksal(boolean):
  - tüm programlama dillerinde bulunur.
  - sadece true ve false değerlerini alabilir.
  - mantıksal olarak 1 ve 0 yani 1 bit yer işgal ettiği halde derleyiciler 1 byte yer ayırırlar.
  - 1 bite erişim 1 byte erişimden daha fazla zaman alır.
  - C ve python dillerinde mantıksal tür bulunmaz iken c++ da doğrudan tip olarak bulunur.
  - Python dilinde bir değişkene true ve false değeri atanabilir.
  - c ve pythonda ilişkisel ifadenin sonucu 1 ve 0 olarak değerlendirilir. C de 0 değeri false diğer tüm değerler true olarak değerlendirilir.

# Sayısal Veri Tipleri

- Karakter:
  - Hesaplamalarda geçerli değildir.
  - Sadece karakter olarak bilgi saklar. ASCII kodlaması ile saklanır. 128 karakter barındırır.
  - C dilinde char ve int veri tipleri dönüşümlü olarak kullanılabilirler.
  - c++'da char anahtar kelimesi kullanılır.
  - Java dilinde ise Unicode olarak karakterler saklanır. bellekte 2 byte yer kaplar. 65535 tane karakter yazılabilir. Böylece latin harfler dışında bir çok farklı karakteri barındırır.

# Sayısal Veri Tipleri

- Karakter katarı(string):
  - karakter dizisini tutar.
  - temel veri tipi olarak tanımlanmamış dillerde(c, c++ gibi) tek karakterli bir karakter dizisi olarak saklanırlar.
  - Java dilinde string temel veri tipidir.
  - c dilinde string veri tipi bulunmaz.
    - char *str = "bozok" -> str[0] değeri b karakterini verir.

# Kullanıcı tanımlı tipler

- Enumeration(enum), sayılama:
  - kullancıya kendine ait veri tipini sunma yöntemidir.
  - enum içerisinde tanımlı veri tipleri tamsayı veri tipine eşlenir.

typedef enum gunler {pazartesi, salı, çarşamba, perşembe}g;

g hafta;

hafta = pazartesi;

printf("%d", hafta); -> 0 olacaktır.

salı -> 1 , çarşama->2 değerini verir.

# Türetilmiş Veri Tipleri

- Programlama dillerinin genelince ilkel veya farklı veri tipleri kullanılarak türetilmiş veri tipleri oluşturulabilir.
- Diziler örnek olarak verilebilir.
- Bellekte bir dizi yerde saklanırlar.

# Türetilmiş Veri Tipi-Diziler

- Diziler art arda gelen aynı tip bilgiyi saklayan bellek elemanlarıdır.
- Aynı anda bir değişken ismi altında birden fazla veriyi tutmak için kullanılır.
- Programların anlaşılırlığını ve esnekliğini artırır.
- dizide elemanların yeri ilk elemana göre belirlenebilir.
- dizi de bulunan tüm elemanları için bir temel veri tipi bulunur.
- indislerin gösteriminde genelde [ ] köşeli parantez kullanılır.

int öğrenciler[10] ifadesi int veri tipinde 10 elaman barındırabilir ve öğrenciler isimli bir dizi tanımlamasıdır.

- 10 değeri indis numarası değil bellekte ayrılacak hücre sayısını ifade eder.
- dizilerde elamanlara erişim ilk elemanın adresi tutularak yapılır.

# Türetilmiş Veri Tipi-Diziler

int sayilar [] = {1,2,3,4,4,5};

int *p  = sayilar;

cout<<*p -> ifadesinin sonucunda işaretçi otomatik olarak ilk indisten başladığı için çıktı 1 olacaktır. !!

- istenilmesi halinde birden fazla boyutlu dizilerin tanımlanması da mümkündür.
- genel olarak tüm indisler için alt değer varsayılan olarak 0 dır.

# Diziler- Dizi Tipleri Adresleme

- Dizilerde adresleme için ilk indisin bulunduğu adres ve her bir eleman için ayrılan bellek alanı kullanılır.
- Tek boyutlu diziler için
  - ilk adres + i*a (i. inci eleman ve her eleman için a birim bellek alanı ayrılıyor ise ) şeklinde ilgili elemanın adres alanına ulaşılabillir.
- İki boyutlu diziler de tek boyutlu diziler gibidir.

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 |

iki boyutlu dizisinde adresleme [0][0] ile başlar ve [3][3] ile tamamlanır.
Bellekte bulunma sırası
1 2 3 4 5 6 7 8 ... 16 şeklinde olacaktır. !!

# Diziler- Dizi Tipleri Adresleme

- Dizilerde uzunluk sabittir anca bu uzunluğu belirleme zamanı değişebilir. çalşıma anında veya derleme anında dizilerin uzunlukları belirlenebilir.
- bir kez belirlendikten sonra uzunlukları değiştirilemez.
- Dizilerde boyut büyütebilmenin tek yolu yeni bir dizi yaratarak önceki dizideki değerleri yeni diziye aktarmaktır.
- Çalışma anında dizi uzunluklarının belirlenmesinde dikkatli olunmalıdır. Çünkü blok içerisinde tanımlanan dizilerde local blok kapandığı anda dizi silineceği için daha sonra kullanılamaz.
- Bu sorunu aşabilmek için dizi Heap bellek bölgesinde oluşturulmalıdır. Bunun için C++ dilinde aşağıdaki gibi kodlama yapılmalıdır.

int sayiler[uzunluk]; -> local tanım

int *sayilar = new int [uzunluk](C++) -> Heap bölgesi tanımıdır..

int [] dizi = new int [uzunluk ](Java) -> Heap bölgesi tanımı

# Kayıt Tipi(record type)

- alt alan olarak birden fazla isimlendirme sayesinde, dizi gibi illa numerik olarak ulaşılmasına ihtiyaç duyulmadan durumları ifade eder.
- isim ile erişim sağlanabilir.
- Kayıtlarda heterojen bir dağılım mümkündür.
- indis numarası yerine isim ile erişim sağlanır.
- Java sınıf oluşturup içerisinde bulunan özellikleri oluşturulacak bir nesne ile erişim sağlanması buna örnek olarak verilebilir.

# Union(Ortaklık) tipi

- aynı bellek bölgesinin farklı değişkenler tarafından kullanılmasının sağlayan veri tipidir.
- farklı zamanlarda kullanılacak birden fazla değişken için ayrı ayrı yer ayırma zorunluluğunu ortadan kaldırarak bellekten kazanç amacıyla kullanılır.
- struct yapısında her bir değişken için ayrı ayrı bellek alanı ayrılırken, union yapısında tek bir bellek alanı kullanılır.

```
union bir {
  int id;
  double dd;
  char cd;
} bd;
```

```
bd.id = 1;
printf("%d", bd.id);
bd.id = 2;
printf("%d", bd.id);
```

# Set(Küme) tipi

- aynı tipte ve birbiriyle ilgili bilgilerin tutulduğu yapıdır.
- Java, c++ gibi güncel diller tarafından desteklenmez. Bunun sebebi kelime boyutuna göre bellekte yer ayırır ve kelime boyutu farklı mimarilerde farklı olabileceği için taşınabilirliği azaltmasıdır.
- Pascal dilinde bu özellik sunulur.

Type

numbers = set of [0..10] şeklinde tanımlanır.

küme tipinde bileşim, keşişim, küme farkı, küme eşitliği gibi işlemler yapılabilir.

* -> kümelerin ortak elemanı, + kümelerin tüm elemanaları, - bir kümede olup diğer kümede olmayan elemanları, = küme elemanları arasında karşılaştırmayı verir.

# Pointer(Gösterge) tipi

- İşaretçiler değişkenlerin kendisini değil adreslerini gösterirler.
- işaretçilere veriler değil bunların saklı olduğu adresler atanır.
- işaretçi tipindeki değişkenler, gösterdikleri veri ne olursa olsun sabit uzunluktadırlar ve tek bir bellek alanında saklanırlar.
- Bu sebeple bellek kullanımı ve yönetimi pointer sayesinde etkindir.
- fonksiyonlar arası dizi, string gibi uzun veri katarlarını aktarmak için faydalıdır.
- ayrıca bir fonksiyondan geriye de pointer sayesinden istenilen büyüklükte veri alınabilir.

int *point;

işaretçilerin birbirine atanması için & işareti kullanılır. Bu adresin atanması gerektiğini gösterir.

int * = &point;

değeri okumak içinde yine * ile değişkeni çağırmak gereklidir.

# Pointer(gösterge) tipi

| adres | değer | değişken |
|---|---|---|
| 2001 | 50 | y |
| 2002 | 25 | z |
| 2003 | 15 | t |

```
int x = &y;
int v = &z;
```

| adres | değer | değişken |
|---|---|---|
| 1900 | 2001 | x |
| 1901 | 2002 | v |
| | | |

# Kullanıcı Tanımlı Veri Türleri

- struct yapısı kullanıcı tanımlı veri türüdür.
- içerisinde farklı ilkel veri tipleri barındırabilir.
- C dilindeki struct yapısında fonksiyon vb. olmadığı için tam bir sınıf mantığında çalışmaz. Ancak c++ ta bulunan struct yapısı sınıf mantığında çalışır. içinde hem ilkel veri tipleri hem de fonksiyonlar barındırabilrir.

```
typedef struct bozok{
        int ogrencino;
}BB;
BB nesne1;
nesne1.ogrencino = 100;
```

# Bellek Yönetimi

- Programda bulunan değişkenler, sınıflar, fonksiyonlar mutlaka bir bellek alanında saklanmaları gereklidir.
- Bellekte tutuldukları yerler bakımında 4 bölge vardır:
  - statik bellek bölgesi
  - çalışma anı yığını
  - heap bellek bölgesi
  - derlenmiş kod bölgesi

# Statik Bellek Bölgesi

- Bu bellek bölgesinde bulunacak değişkenler program başlamdan bellidir.
- global değişkenler, sabitler, static olarak tanımlanmış local değişkenler bu bellek alanında tutulurlar.
- program çalıştığı sürece bellekte kalırlar ve program sonlanınca bellekten silinirler.
- Program başında bir kere tanımlanıp program sonuna kadar kaldıkları için kullanımları sırasında en son kaldıkları değerler her zaman hafızada tutulurlar.
- Java dilinde metod içerisinde statik kullanımına izin verilmez. Sadece sınıf içerisindeki elemanlar statik olabilir. Bu sayede bu eleman sınıftan türetilecek bütün nesneler için ortaktır ve aynıdır.
- Global değişkenlerin statik bellek bölgesinde saklanması ve program boyunca değişip değişmediklerinin kontrol edilmesi zor olduğu için sık kullanılması tavsiye edilmez.

# Çalışma Anı yığını

- Stack yapısındadır ve stack yapısında çalışır. bu sebeple ilk giren değer son çıkar.
- Program çalıştığı sürece aktif olarak büyük küçülür ve içerisinde ihtiyaç duyulan değerler saklanır.
- fonksiyon ve metod çağrımları ve local değişkenler tutulur.
- fonksiyonlar gönderilen parametreler ve dönüş parametleri bu bellek alanında tutulur.
- çağrılmış fonksiyondan geri dönüldüğünde, yığında bulunan ilk adresine dönüldüğü için bir daha o dönüş değerine erişilemez.

# heap bellek bölgesi

- c, c++ gibi dillerde bu bölgenin kontrolü programcıya bırakılmıştır bu sebeple dikkatli kullanılmalıdır.
- bu bölge iyi kullanılmaz ise bellek taşması, yanlış değere erişim gibi problemler olur.
- dinamik olarak oluşturulan yapıların boyutları değişken olacağı ve çalışma anında belirleneceği için bu bölgeye ihtiyaç duyulmaktadır.
- Heap bellek bölgesinde yer ayırmak için C dilinde malloc, Java da ise new keywordü kullanılır.
- C dilinde malloc kullanabilmek için stdlib.h kütüphanesine ihtiyaç duyulur.
- Heap bellek bölgesi kullanıldıktan sonra kullanıcı tarafından belleğe iade edilmelidir. Aksi halde garbage oluşur. ve Java da bulunan garbage collector gibi yapılara ihtiyaç duyulur.
- Java da belirli aralıklara otomatik olarak bellek temizliği yapılrken C dilinde free komutu kullanılmalıdır. Java el ile yapılmak isteniyor null kullanılmalıdır.

# void keyword

- türü olmayan göstericidir.
- yer geldiğinde bir tamsayı , yeri geldiğinde bir kayan noktalı sayı olabilir.

```
int x = 100;

void *obj;

obj = &x;
```

Java dilinde ise

```
Object x = 100; // java da object nesnesi vardır.
```

# Kuvvetli Tipleme

- farklı veri tiplerinin etkileşimlerini kısıtlanmasına kuvvetli tipleme denir.
- programlama dilinin tüm tip hatalarını yakalayabilmesi onu kuvvetli bir tip yapar.
- c, c++ dilleri kuvvetli tiplemeyi desteklemez.

# Tip Dönüşümleri

- deyimler birden fazla operatör, sabit veya değişken barındırabilir ve bunların farklı veri tipinde olması olağandır.
- sonucun hangi tipte olacağına ise deyim içerisindeki değerlerin tipine göre karar verilir.
- deyim içerisindeki en büyük bellek alanına sahip olan veri tipi sonuç veri tipi olarak belirlenir.
- Farklı veri tipleri ile çalışıldığı durumda problem yaşamamak için
  - deyimde yer alan ifadelerden birisini büyük veri tipi olarak belirleme
  - tip dönüşümü yapma yöntemleri kullanılabilir.
- karışık şekilde deyimlerde operatörler farklı tipte operand alırlarsa tip dönüşümüne ihtiyaç duyulur. tip dönüşümünde bir esne kendi tipindeki tüm değerleri içermeyen bir tipe dönüştürülür ise daralan dönüşüm denir. kayan noktalı sayıdan tamsayıya dönüşüm bu şekildedir. kendi tipinin tüm değerlerini içeren bir tipe dönüşüm var ise genişleyen dönüşüm denir. tamsayının kayan noktalı sayıya dönüşü örnektir.

# Tip Dönüşümleri

- farklı tipteki değerin işlem sonucunun zorunlu olarak tip dönüşümüne tabi tutulmasına örtülü dönüşüm denir. Bu özellik derleyici tasarımında karar verilir.

```
int a = 10;
float b = 2.3;
double x = a*b;
```

- Derleyicinin otomatik olarak yapmadığı tip dönüşümleri programı tarafından da yapılabilir. Buna dışsal casting denir.

```
int top = (int)t + 10;
```

- Bu tip dönüşümleri sadece gerekli hallerde kullanılmalıdır. Bu deyimlerde ortaya çıkacak hataları derleyiciler yakalayamaz.

# Sorular

- aynı sayıların float ve double olmasına rağmen sonucun değişmesinin nedeni nedir?
- union veri tipi nedir? neden ihtiyaç duyulur.
- kuvvetli tipleme nedir* açıklayınız.