



BLM5504 NESNEYE DAYALI KAVRAMLAR VE PROGRAMLAMA
Yrd. Doç. Dr. Yunus Emre SELÇUK

DERS NOTLARI:
B. DENETİM AKIŞI AYRINTILARI

1



DENETİM AKIŞI

DENETİM AKIŞI

- Denetim akışı: Kodların yürütüldüğü sıra.
 - En alt düzeyde ele alındığı zaman bir bilgisayar programı, çeşitli komutların belli bir sıra ile yürütülmesinden oluşur.
 - Komutların peş peşe çalışması bir nehrin akışına benzetilebilir.
 - Komutların kod içerisinde veriliş sırası ile bu komutların yürütüldüğü sıra aynı olmayabilir.
 - Belli bir komut yürütülmeye başlandığı zaman ise o komut için denetimi ele almış denilebilir.
 - Bu benzetmelerden yola çıkarak, kodların yürütüldüğü sıraya denetim akışı adı verilebilir.

2

DENETİM AKIŞI

KARAR VERME İŞLEMLERİ – IF DEYİMİ

- Karar verme işlemleri:
 - Bir koşulu sınyarak ne yapılacağına karar vermek için if komutu kullanılır.

```
if( ifade )  
  
    ifadenin sonucu doğruysa çalışacak tek bir komut;  
  
if( koşul ) {  
    //komutlar  
}
```

- İfade/koşul çeşitleri:
 - Karşılaştırma (ilişkişel işleçler(operator)): < > <= >= == !=
 - Mantıksal işlemler: Boole cebiri. & | !
- İfadeler ve/veya mantıksal işlemleri ile birleştirilebilir.
 - Çift işleç kullanılır: && ||
- Karmaşık ifadeler işlem önceliği ve okunabilirliği arttırmak için parantezlenebilir.

3

DENETİM AKIŞI

KARAR VERME İŞLEMLERİ – IF DEYİMİ

- Karşılaştırma:

```
if( yas >= 18 )  
    System.out.println("Tebrikler, siz bir yetişkinsiniz.");  
  
if( x != y )  
    System.out.println("x ile y'nin değeri farklıdır.");  
  
if( x == y )  
    System.out.println("x ile y'nin değeri aynıdır.");
```

4

DENETİM AKIŞI

KARAR VERME İŞLEMLERİ – IF DEYİMİ

- Mantıksal işlemler:

```
boolean dogru = true, yanlis = false, mantikli = true;
```

```
if( !dogru )
```

```
    System.out.println("Söylenen yalan.");
```

```
if( !dogru & mantikli )
```

```
    System.out.println("Söylenen yalan ama mantıklı!");
```

- Birleştirme:

```
if( dogru && mantikli )
```

```
    System.out.println("Söylenen hem doğru hem mantıklı.");
```

- Sınamalarda işlem yerine birleştirme tercih edin (& yerine &&, | yerine ||)

5

DENETİM AKIŞI

KARAR VERME İŞLEMLERİ – IF DEYİMİ

- Karar verme işlemleri:

- Bir koşulu hem doğruluk hem de yanlışlık açısından değerlendirmek için if komutu else anahtar kelimesi ile birlikte kullanılır.

```
if( koşul ) {
```

```
    //koşul doğru ise çalışacak komutlar
```

```
}
```

```
else {
```

```
    //koşul yanlış ise çalışacak komutlar
```

```
}
```

- Örnek:

```
if( yas >= 18 ) {
```

```
    System.out.println("Tebrikler, siz bir yetişkinsiniz.");
```

```
}
```

```
else {
```

```
    System.out.println("Henüz bir yetişkin değilsiniz.");
```

```
}
```

6

DENETİM AKIŞI

KARAR VERME İŞLEMLERİ – IF DEYİMİ

- Örnek program ve akış şeması:

```
package temeller;
import java.util.*;
public class KararVerme01 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("Yetişkinlik yaşı sınırını girin: ");
        int sinir = in.nextInt();
        System.out.print("Kişinin yaşını girin: ");
        int yas = in.nextInt();
        if( yas >= sinir )
            System.out.println("Kişi kanunen bir yetişkindir.");
        else
            System.out.println("Kişi henüz reşit değildir.");
        in.close();
    }
}
```

7

DENETİM AKIŞI

KARAR VERME İŞLEMLERİ – IF DEYİMİ

- Zincirleme karar verme işlemleri:

```
if( koşul1 ) {
    //koşul1 doğru ise çalışacak komutlar
}
else if( koşul2 ){
    //koşul1 yanlış ve koşul2 doğru ise çalışacak komutlar
}
else if( koşul3 ){
    //koşul1,2 yanlış; koşul3 doğru ise çalışacak komutlar
}
...
else {
    /*tüm koşullar yanlış ise çalışacak komutlar.
    seçimliktir, bulunması şart değildir.
    Birden fazla if varsa, else en yakın if'e aittir,
    ama siz yine de kıvrık parantezleri kullanın.*/
}
```

8

DENETİM AKIŞI

KARAR VERME İŞLEMLERİ – IF DEYİMİ

- Zincirleme karar verme işlemleri örneği:

```
if( sicaklik > enSicakGun ) {  
    System.out.println("Bugün olağanüstü sıcak bir gün.");  
}  
else if( sicaklik < enSogukGun ){  
    System.out.println("Bugün olağanüstü soğuk bir gün.");  
}  
else {  
    System.out.println("Bugün sıradan bir gün.");  
}
```

- Akış şemasını da çiz.

9

DENETİM AKIŞI


KARAR VERME İŞLEMLERİ – IF DEYİMİ

- if blokları iç içe alınabilir :

```
if( koşul1 ) {  
    //koşul1 doğru ise çalışacak komutlar  
    if( koşul2 ){  
        //koşul2 de doğru ise çalışacak komutlar  
    }  
}  
else {  
    if( koşul3 ){  
        //koşul1 yanlış ve koşul3 doğru ise çalışır.  
    }  
}
```

- Akış şemasını da çiz.

10




DENETİM AKIŞI

KARAR VERME İŞLEMLERİ – IF DEYİMİ

- Örnek: Basit bir piyango çekilişi
 - Bilgisayar iki basamaklı rastgele bir sayı üretir.
 - Kullanıcı bir tahmin girer.
 - Kullanıcı sayıyı doğru tahmin ederse 10,000 TL kazanır.
 - Kullanıcı sayının basamaklarını ters sırada tahmin ederse 3,000TL kazanır.
 - Eğer kullanıcı tek bir basamağı tahmin ederse 1,000TL kazanır.

11



DENETİM AKIŞI

KARAR VERME İŞLEMLERİ – IF DEYİMİ

```
package temeller;
import java.util.*;
public class KararVerme02 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int piyango = (int)(Math.random() * 100);
        System.out.print("Tahmininizi girin: ");
        int tahmin = in.nextInt();
        System.out.println("Çekilen sayı: " + piyango);
        if (tahmin == piyango)
            System.out.println("Doğru bildiniz: Ödülünüz 10,000TL");
        else if (tahmin % 10 == piyango / 10 && tahmin / 10 == piyango % 10)
            System.out.println("İki rakam bildiniz: Ödülünüz 3,000TL");
        else if (tahmin % 10 == piyango / 10 || tahmin % 10 == piyango % 10
            || tahmin / 10 == piyango / 10 || tahmin / 10 == piyango % 10)
            System.out.println("Tek rakam bildiniz: Ödülünüz 1,000TL");
        else
            System.out.println("Üzgünüm, kaybettiniz.");
        in.close();
    }
}
```

12

DENETİM AKIŞI

DÖNGÜLER – FOR DEYİMİ

- Şimdiye kadar verdiğimiz komutlar yalnız bir kez işlendi veya koşula bağlı olarak hiç işlenmedi.
- Aynı komutu birden fazla kez işletmek istediğimizde çevrim (döngü:loop) ifadeleri kullanırız.
- for ifadesi ile döngü: Bir komutu belli bir sayıda yinelemek için.

```
for( baslangicIfadesi; devamIfadesi; artimIfadesi ) {  
    komutlar;  
}
```

- Başlangıç ifadesi:
 - Döngüyü yineleme sayısını, bir sayaca bağlı olarak belirleriz.
 - Başlangıç ifadesinde sayaca ilk değer ataması yapılır.
 - Sayaç, başlangıç ifadesi içerisinde de tanımlanabilir.
 - `int i = 0;`

13

DENETİM AKIŞI

DÖNGÜLER – FOR DEYİMİ

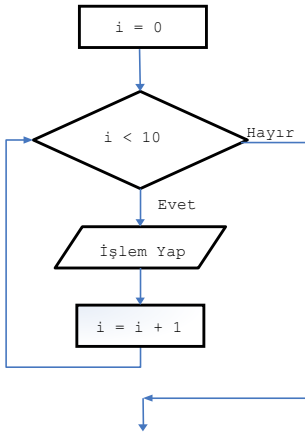
- İleri doğru sayım:

```
for( i = 0; i < 10; i++ ) {  
    System.out.println(i+ " ");  
}
```

 - Çıktısı: 0 1 2 3 4 5 6 7 8 9
- Geriye doğru sayım:

```
for( i = 10; i >= 0; i = i - 2 ) {  
    System.out.println(i+ " ");  
}
```

 - Çıktısı: 10 8 6 4 2 0
- Sınır değerlere dikkat.
- Devam ifadesinde != yerine < > <= >= kullan.
 - Özellikle ondalıklı sayılarda.
- Döngünün tamamlanmasını beklemeden sonlandırmak istiyorsak **break** komutunu uygun bir biçimde kullanabiliriz.
- Verilen kurallara aykırı işler yapabilirsiniz ancak döngü aşırı karmaşıklıkla, tavsiye etmiyoruz.



14

DÖNGÜLER – FOR DEYİMİ

- Örnek program: Verilen bir N sayısının asal olup olmadığının tespiti.
 - Algoritma: N'in 2-N/2 aralığındaki sayılarla tam bölünüp bölünemediğine bakarız. Bir tane bile tam bölen bulursak N asal değildir.

```
package temeller;
import java.util.*;
public class Donguler01For {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("Asallığı sınanacak sayıyı girin: ");
        int N = in.nextInt();
        boolean asal = true;
        for( int i=2; i<=N/2; i++ ){
            if( N % i == 0 ) {
                asal = false;
                break;
            }
        }
        if( asal )
            System.out.println("Girilen sayı asaldır.");
        else
            System.out.println("Girilen sayı asal değildir.");
        in.close();
    }
}
```

15

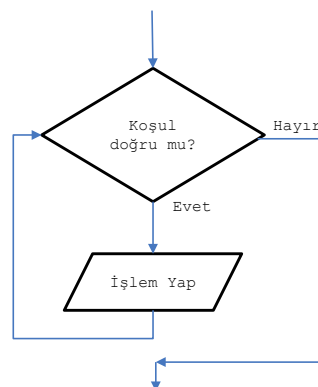
DENETİM AKIŞI

DÖNGÜLER – WHILE DEYİMİ

- while ifadesi ile döngü: Bir komutu belli bir koşul geçerli olduğu sürece yinelemek için.

```
while( kosul ) {
    komutlar;
}
```

- Döngüye girmeden önce döngüye girmeyi kesinleştirmek için koşulu doğrulamak gerekir (bkz. örnek kod).
- Çevrim, koşulun geçersiz olduğu anda değil, koşul geçersiz olduktan sonraki ilk kontrolde biter (bkz. örnek kod çıktısı).



16

DENETİM AKIŞI

DÖNGÜLER – WHILE DEYİMİ

- Örnek kod: Kullanıcı çıkmak isteyene kadar girdiği sayının karekökünü hesaplayan program

```
package temeller;
import java.util.*;
public class Donguler02While {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("Girilen sayıların kareköklerini hesaplayan");
        System.out.println("bu programdan çıkmak için 0 girebilirsiniz.");
        double deger = 7.6;
        while( deger != 0 ) {
            System.out.print("Bir sayı girin: ");
            deger = in.nextDouble();
            if( deger > 0 )
                System.out.println("Karekökü: " + Math.sqrt(deger) );
        }
        in.close();
    }
}
```

17

DENETİM AKIŞI

DÖNGÜLER – WHILE DEYİMİ

- Örnek çıktı:

```
Girilen sayıların kareköklerini hesaplayan
bu programdan çıkmak için 0 girebilirsiniz.
Bir sayı girin: 9
Karekökü: 3.0
Bir sayı girin: 42
Karekökü: 6.48074069840786
Bir sayı girin: -5
Bir sayı girin: 92
Karekökü: 9.591663046625438
Bir sayı girin: 0
```

18

DENETİM AKIŞI

DÖNGÜLER – WHILE DEYİMİ

- Örnek çıktı:

```
Girilen sayıların kareköklerini hesaplayan
bu programdan çıkmak için 0 girebilirsiniz.
Bir sayı girin: 5.5
Exception in thread "main" java.util.InputMismatchException
at java.util.Scanner.throwFor(Unknown Source)
at java.util.Scanner.next(Unknown Source)
at java.util.Scanner.nextDouble(Unknown Source)
at temeller.Donguler02While.main(Donguler02While.java:11)
```

19

DENETİM AKIŞI

DÖNGÜLER – DO/WHILE DEYİMİ

```
do {
    komutlar;
} while( kosul );
```

- Farkı:
 - Kontrolün sonda yapılması.
 - Böylece ilk değer atama sorunu kalkar.
 - Noktalı virgül sonda.

```
graph TD
    Entry(( )) --> İşlemYap[/İşlem Yap/]
    İşlemYap --> Koşul{Koşul doğru mu?}
    Koşul -- EVET --> İşlemYap
    Koşul -- HAYIR --> Exit(( ))
```

20

İÇ İÇE DÖNGÜLER

- ```
System.out.println("i,j");
for(int i = 1; i < 5; i++) {
 for(int j = 1; j < 5; j++) {
 System.out.println("(" + i + ", " + j + ")");
 }
}
```

21

## DÖNGÜLERDEN ÇIKMA: BREAK DEYİMİ

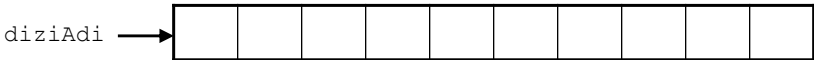
- ```
System.out.println("i,j");
for( int i = 1; i < 5; i++ ) {
    for( int j = 1; j < 5; j++ ) {
        if( i == 2 )
            break;
        System.out.println("(" + i + ", " + j + ")");
    }
}
```

22

DİZİLER

DİZİLER

- Birçok programlama probleminin çözümü için, birbiri ile ilişkili ve aynı tipten verileri ayrı ayrı değişkenlerde tutmak yerine, bunları birlikte saklamak daha uygundur.
- Birlikte saklanan veriler tek bir birleşik "veri yapısı" içerisinde yer alır.
- Bu derste dizi (array) adlı veri yapısını inceleyeceğiz.
- Dizi tanımlama:
`TipAdi[] diziAdi;
diziAdi = new TipAdi[elemanSayisi];`
- Tek adımda tanımlama:
`TipAdi[] diziAdi = new TipAdi[elemanSayisi];`
- Alternatif tanımlama:
`TipAdi diziAdi[]; //C usulü`
- Java usulü, değişkenin bir dizi olduğunu daha iyi vurguluyor.
- Bellekteki durum: Herbiri bir eleman alabilecek, ardışıl konumlanmış hücreler.

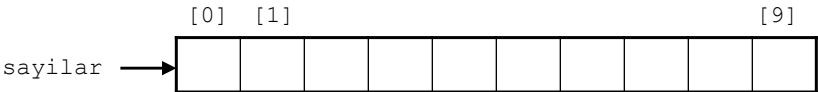


23

DİZİLER

DİZİLER

- Örnek tanımlama:
`int[] sayilar = new int[10];`
- Bellekte oluşan durum:



- DİKKAT: Dizi elemanlarının dizin numarası (indeksi) sıfırdan başlar, eleman sayısı - 1'de biter.
- Diziler genelde çevrim ifadeleri ile kullanılır.
 - Dizilerin tüm elemanlarına sıralı olarak ulaşmak için for kullanılır.
 - Yaz: Üstteki diziye 10,20,...,100 atayan kod.
- Değer atayarak tanımlama:
`int[] sayilar = { 10, 20, 30, ... ,100 };`

24

DİZİLER

DİZİLER

- Dizideki eleman sayısını (dizinin boyutunu) öğrenmek için:

```
int[] a = new int[100];  
int j = a.length;
```
- İstenilen boyutta bir dizi oluşturmak için:

```
int boyut = 100;  
int[] a = new int[ boyut ];
```

25

DİZİLER

DİZİLER ve FOR EACH DÖNGÜSÜ

- Dizilerin elemanlarına sıralı erişim amaçlı özel bir for çevrimi:

```
for( degisken : dizi ) {  
    komutlar;  
}
```
- Örnek:

```
for( sayi : sayilar ) {  
    System.out.println(sayi);  
}
```
- Değişken, dizi elemanları ile aynı tipte olmalıdır.
- Dizilerden başka veri yapıları ile de for-each çevrimi kullanılabilir (ileride anlatılacak).
- Avantajlar:
 - Sınır değerleri ile uğraşmak yok.
 - Daha basit gösterim.
- Dezavantajlar:
 - İndeks değeri kullanmak istiyorsak bir anlamı olmaz (Ör: diziye 10,20,... atamak).

26

DİZİLER

DİZİLER İLE İLGİLİ PROBLEMLER

- Dizideki en küçük veya (en büyük) elemanı bulmak.
 - Değerin kaç olduğunu bulmak yetmez, en küçük elemanın dizideki yerini de bulmak gerekir.
 - Sorunu çözmek için elemanın indeksini saklamak yeter.
 - Algoritmayı yaz:
 - Dizinin ilk elemanını en küçük farzet.
 - Bunu sırayla dizinin diğer elemanları ile karşılaştır.
 - Karşılaştırdığın eleman baktığından küçükse, artık o elemanı en küçük farzet.

27

DİZİLER

DİZİDEKİ EN KÜÇÜK VE EN BÜYÜK ELEMANI BULMAK

```
package temeller;
import java.util.Scanner;
public class DiziOrnekleri01 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int diziBoyutu;
        System.out.print( "Dizi kaç eleman içersin? " );
        diziBoyutu = in.nextInt( );
        int[] dizi = new int[ diziBoyutu ];
        for( int i = 0; i < diziBoyutu; i++ ) {
            System.out.print( (i+1) + ". elemanı girin: " );
            dizi[i] = in.nextInt( );
        }
        int enKucuk = 0, enBuyuk = 0;
        for( int i = enKucuk+1; i < dizi.length; i++ ) {
            if( dizi[enKucuk] > dizi[i] )
                enKucuk = i;
            if( dizi[enBuyuk] < dizi[i] )
                enBuyuk = i;
        }
        System.out.println( "Dizinin en küçük elemanı: " + dizi[enKucuk] );
        System.out.println( "Dizinin en büyük elemanı: " + dizi[enBuyuk] );
        in.close();
    }
}
```

28

DİZİLER

DİZİLER İLE İLGİLİ PROBLEMLER

- Diziyi sıralamak: Artan veya azalan sırada.
 - Algoritma: Selection Sort
 - Dizideki en küçük elemanı önceki şekilde bul.
 - Bulduğun en küçük eleman ile dizinin başındaki elemanın yerini değiştir: İlk eleman yerine oturdu.
 - Dizinin 2. en küçük elemanını bul. Dizinin başından başlama, çünkü orada 1. en küçük eleman var. Dolayısıyla dizinin 2. elemanından başla.
 - Bulduğun 2. en küçük eleman ile dizinin başındaki elemanın yerini değiştir: 2. eleman yerine oturdu.
 - Bu şekilde tüm elemanları yerine oturt.
 - İşlem:
 - 1. elemanı 2., 3., ... elemanlarla karşılaştır.
 - 2. elemanı 3., 4., ... elemanlarla karşılaştır.
 - ... =>Yukarıdan aşağıya 1,2, ... soldan sağa sütun başı +1'den başlayan bir seri var. Her seri bir döngü: İç içe iki döngü.

29

DİZİYİ ARTAN SIRADA SIRALAMA

```
package temeller;
import java.util.Scanner;
public class DiziOrnekleri02 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int diziBoyutu;
        System.out.print( "Dizi kaç eleman içersin? " );
        diziBoyutu = in.nextInt( );
        int[] dizi = new int[ diziBoyutu ];
        for( int i = 0; i < diziBoyutu; i++ ) {
            System.out.print( (i+1) + ". elemanı girin: " );
            dizi[i] = in.nextInt( );
        }
        int i, j;
        for( i = 0; i < dizi.length-1; i++ ) {
            int minIndex = i;
            for( j = i+1; j < dizi.length; j++ ) {
                if( dizi[j] < dizi[minIndex] ) {
                    minIndex = j;
                }
            }
            int temp = dizi[i];
            dizi[i] = dizi[minIndex];
            dizi[minIndex] = temp;
        }
        System.out.print( "Sıralanmış dizi: " );
        for( int eleman: dizi )
            System.out.print( eleman + " " );
        in.close();
    }
}
```

30

DİZİLER

DİZİYİ ARTAN SIRADA SIRALAMA

- İç içe 2 döngü: $O(n^2)$ karmaşıklık.
 - Özyinelemeli algoritmalarla daha düşük karmaşıklık.
 - Ör: Quicksort
 - Hazır kullanım: `java.util.Arrays.sort` metodu.

```
for( i = 0; i < dizi.length-1; i++ ) {  
    int minIndex = i;  
    for( j = i+1; j < dizi.length; j++ ) {  
        if( dizi[j] < dizi[minIndex] ) {  
            minIndex = j;  
        }  
    }  
    int temp = dizi[i];  
    dizi[i] = dizi[minIndex];  
    dizi[minIndex] = temp;  
}
```

`Arrays.sort(dizi);`

31

DİZİLER

DİZİLER İLE İLGİLİ PROBLEMLER

- Sıralı dizide en hızlı eleman arama algoritması: İkili arama (binary search)
 - Aranacak değeri dizinin ortasındaki elemanla karşılaştır.
 - bas: dizinin başını, son: dizinin sonunu gösteren değişken.
 - Başlangıçta `bas = 0`, `son = uzunluk-1`, `orta := (bas + son) / 2`
 - Eşitlerse, bulunmuştur.
 - Dizinin ortasındaki eleman aranan elemandan küçükse, dizinin sadece sol tarafında ara (`bas = orta + 1`).
 - Dizinin ortasındaki eleman aranan elemandan büyükse, dizinin sadece sağ tarafında ara (`son = orta - 1`).
 - `bas > son` ise dizide arayacak yer kalmamış demektir.
 - Sonuç: Aranacak eleman dizide mevcutsa elemanın indeksi
 - Aranacak eleman dizide yoksa `-1` sonucu verilsin
 - Gerçekleme ayrıntıları: Dizi bir kez girilsin, sonra kullanıcı 0 girene dek girdiği her değer dizide aransın.

32

DİZİLER

İKİLİ ARAMA

- Program kodu:

```
package temeller;
import java.util.*;
public class DiziOrnekleri03 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int diziBoyutu;
        System.out.print( "Dizi kaç eleman içersin? " );
        diziBoyutu = in.nextInt( );
        int[] dizi = new int[ diziBoyutu ];
        for( int i = 0; i < diziBoyutu; i++ ) {
            System.out.print( (i+1) + ". elemanı girin: " );
            dizi[i] = in.nextInt( );
        }
        Arrays.sort(dizi);
        System.out.print( "Sıralanmış dizi: " );
        for( int eleman: dizi )
            System.out.print( eleman + " " );
        System.out.println();
    }
}
```

33

- Program kodu (devam):

```
char c; int aranan, bas, son, orta, yer;
do {
    System.out.print("Aramak istediğiniz elemanı girin: ");
    aranan = in.nextInt();
    bas = 0; son = dizi.length-1; yer = -1;
    while( bas <= son ) {
        orta = ( bas + son ) / 2;
        System.out.print(" bas: " + bas);
        System.out.print(" son: " + son);
        System.out.print(" orta: " + orta);
        System.out.println(" dizi[orta]: " + dizi[orta]);
        if( dizi[orta] == aranan ) {
            yer = orta; break;
        }
        else {
            if( dizi[orta] < aranan )
                bas = orta + 1;
            else
                son = orta - 1;
        }
    }
    if( yer != -1 )
        System.out.println("Aranan değer " + (yer+1) + ". sırada bulundu.");
    else
        System.out.println("Aranan değer bulunamadı.");
    System.out.print("Devam etmek istiyor musunuz (e/h)? ");
    in.nextLine();
    c = in.nextLine().charAt(0);
} while( c == 'E' || c == 'e' );
in.close();
}
```

34

DİZİLER

İKİLİ ARAMA

- Program çıktısı (kısmi):

```
Sıralanmış dizi: 13 19 27 28 45 57 58 76 83 99
Aramak istediğiniz elemanı girin: 27
  bas: 0 son: 9 orta: 4 dizi[orta]: 45
  bas: 0 son: 3 orta: 1 dizi[orta]: 19
  bas: 2 son: 3 orta: 2 dizi[orta]: 27
Aranan değer 3. sırada bulundu.
Devam etmek istiyor musunuz (e/h)? e
Aramak istediğiniz elemanı girin: 88
  bas: 0 son: 9 orta: 4 dizi[orta]: 45
  bas: 5 son: 9 orta: 7 dizi[orta]: 76
  bas: 8 son: 9 orta: 8 dizi[orta]: 83
  bas: 9 son: 9 orta: 9 dizi[orta]: 99
Aranan değer bulunamadı.
Devam etmek istiyor musunuz (e/h)? h
```

- Algoritmanın karmaşıklığı: $O(\ln N)$
 - $N = 2^k$ eleman için en fazla k adımda aranan elemana ulaşılır.

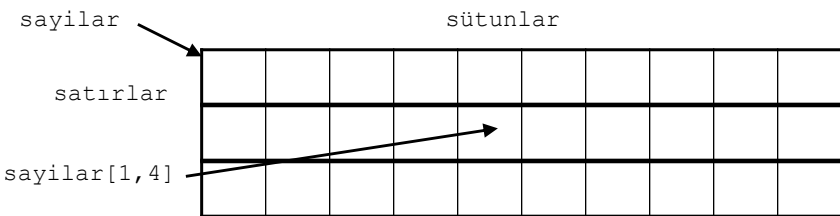
35

DİZİLER

ÇOK BOYUTLU DİZİLER

- Şimdiye dek gördüğümüz diziler doğrusal bir yapıydı = tek boyutluydu.
- Bir değişkeni 0 boyutlu bir varlık, yani nokta gibi düşünebilirsiniz. Dizileri ise yan yana dizilmiş noktalardan oluşan bir çizgi gibi, yani bir boyutlu olarak düşünebilirsiniz.
- Doğal olarak düşüncemiz 2 ve 3 boyutlu dizilere genişleyecektir.
- 2 boyutlu dizi: Bir satranç tahtası gibi, satırlar ve sütunlardan oluşmuş, dizilerin dizisi.
- 2 boyutlu dizi tanımlama: [satır,sütun], dikkat: önce satır, sonra sütun.

```
int[][] sayilar = new int[3][10];
```



36