

BÖLÜM 1. GİRİŞ

BÖLÜM 2. ALGORİTMALARA GENEL BAKIŞ

BÖLÜM 3. ALGORİTMALAR

BÖLÜM 4. ALGORİTMA VE AKIŞ ŞEMALARI

BÖLÜM 5. DOSYALAMA SİSTEMLERİ

**BÖLÜM 6. ÇEŞİTLİ KONULARDA ALGORİTMA VE AKIŞ
ŞEMALARI**

ALGORİTMALARA GENEL BAKIŞ

✓ Algoritmanın Tanımı

✓ Algoritmanın Yapısı

✓ Algoritmanın Dili

ALGORİTMANIN TANIMI

Algoritmalar, problemleri çözmek için adım adım procedürlerdir.

Algoritma, bilgisayarda problemlerin bir sınıfını çözmek için bir metottur.

Algoritma, bir mekanik kural veya otomatik metod veya bazı matematiksel işlemlerin düzenlenmesi için programdır.

Algoritma, soruların herhangi verilen bir sınıfına cevaplar bulmakta kullanılabilen bir hesaplama prosedürü için etkili komutların kümesidir.

Algoritma, açık olarak tanımlanmış olan ve herhangi bir bilgisayara icra edilen bir prosedürdür.



ALGORİTMANIN YAPISI

- ✓ Atama adımları (Bir değişkene bazı değerlerin atanması gibi)
- ✓ Aritmetik adımlar (Toplama, bölme, çıkarma, çarpma gibi)
- ✓ Mantiki adımlardır (İki sayının karşılaştırılması gibi)



$:=$ atama sembolüdür.

örnek : $\text{Max} := a$

(a'nın değeri max değişkenine atanır.)

örnek : $b := 5$

(b değişkenine 5 değeri atanır.)



ARİTMETİK ADIMLAR

- (+) Toplama İşlemi
- (-) Çıkarma İşlemi
- (*) Çarpma İşlemi
- (/) Bölme İşlemi
- (=) Aktarma ve Eşitlik
- (^) Üs Alma İşlemi
- (< >) Eşit Değil (Farklı) İşlemi
- (<) Küçüktür İşlemi
- (>) Büyüktür İşlemi
- (< =) Küçük ya da Eşit İşlemi
- (> =) Büyük ya da Eşit İşlemi



MANTIKİ ADIMLAR

örnek :

$5 < 3 \longrightarrow \text{false}$

$6 > 2 \longrightarrow \text{true}$

örnek :

$a := 4$

$6 > a \longrightarrow \text{true}$

$a < 2 \longrightarrow \text{false}$



ALGORİTMANIN DİLİ

✓ Kodlama

✓ Şartlı Yapılar

✓ Döngü Yapıları



KODLAMA

1) Procedure = Bir algoritmanın kodlanmasına başlanan ilk ifadedir. Bu ifadede ;

örnek: Procedure max(L = list of integers)

Burada algoritmanın adı max iken tamsayıların listesinin maksimumunu bulur (L).

2) Assignments = Atamalar ve ifadelerin diğer tipleri
Assignments ifadesi değişkenlere değer atamada kullanılır. Bu ifadede sol taraf değerın adını alırken sağ taraf ise prosedürlerle tanımlanan fonksiyonları , değerleri atanana değişkenleri , sabitleri içeren bir ifade yada deyimdir. Sağ tarafta ayrıca aritmetik işlemlerin herhangi biri de bulunabilir.

örnek: Max := a

örnek: b := 5 + 3



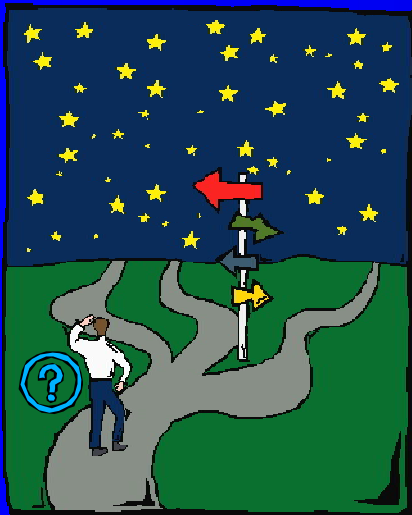
IF – THEN VE IF – THEN - ELSE

Genel Yazılımı : If kontrol ifadesi **Then** ifade1;

 If kontrol ifadesi **Then** ifade1
 else ifade2;

Açıklama : ‘Kontrol ifadesi’ sonucu doğruysa ‘ifade1’ ile belirtilen ifade ya da ifade grubunun yapılmasını sağlayan deyimdir.

Bazı durumlarda karşılaştırma yanlışsa, hiçbir işlem yapmaya gerek yoktur. Böyle durumlarda ‘*else*’ kısmına gerek kalmaz ve if de- yimi, Then kısmından sonraki ifade ile bitirilir.



Örnek 1: If Ort > 50 then Writeln(‘Geçti’);
 writeln(‘ne olacak şimdi’);

} Tek
Dallanma

Örnek 2: If Ort > 50 then Writeln(‘Geçti’)
 else writeln(‘ne olacak şimdi’);

} Çift
Dallanma



Balık elimde
olduğu
sürece dolan



WHILE - DO

Topu
yakalayıncaya
kadar dön



REPEAT - UNTIL

5 kez
dön



FOR - DO





WHILE - DO

Genel Yazılımı : While kontrol ifadesi **Do** tek veya blok ifade;

Açıklama : Do kelimesinden sonraki tek veya blok ifadeyi, Kontrol ifadesi doğru olduğu müddetçe işletir. Buradaki kontrol ifadesi, mantıksal ifade veya mantıksal değişkendir. Tek veya blok ifadeyi işletmeden önce, kontrol ifadesini test eder ve yanlışsa ifadeyi hiç işletmeden döngüden çıkar.

Örnek Program :

```
Var k:integer;  
Begin  
  k:=1;  
  while k < 27 do begin  
    write(k+5);  
    k:=k*2;  
  end;  
  writeln;  
  writeln(k);  
End.
```



k	Kontrol ifadesi (k<27)	Çıktı
1	1 < 27 doğru	6
2	2 < 27 doğru	7
4	4 < 27 doğru	9
8	8 < 27 doğru	13
16	16 < 27 doğru	21
32	32 < 27 yanlış	-
32	-	32

Topu
yakalayınca
kadar dön



REPEAT - UNTIL

Genel Yazılımı : **Repeat** komut veya komutlar **Until** şart ;

Açıklama : Program akışı döngü içerisine girdiği anda, **Until** komutunda belirtilen şart sağlanıncaya kadar iki komut arasındaki işlemler sürekli olarak tekrarlanır.

While döngüsünün **Repeat** ' den en önemli farkı, önce kontrol ifadesine bakılır, sonra döngü bloğu işletilir. **Repeat** ise sonradan kontrollü döngüdür. Yani, önce döngü bloğu işletilir, sonra kontrol ifadesine bakılır.



FOR - DO

5 kez
dön



Genel Yazılımı : **For** **değişken:=** başlangıç değeri **To\DownTo** bitiş değeri **Do**

Açıklama : Belirlenen işlem ya da işlemleri istenilen sayıda tekrarlamak veya istenen iki aralıkta değer elde etmek için kullanılır.

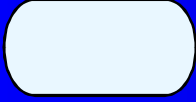
Örnek Program : 1'den 100'e kadar olan sayıların toplamını bulan program.

```
Var k, toplam:integer;  
Begin  
  toplam:=0;  
  for k:=1 to 100 do  
    toplam:=toplam+k;  
  writeln(toplam);  
End.
```

k	Toplam:=Toplam+k	Açıklama
?	0	Toplam değişkenine 0 değerini ver
1	$0 + 1 = 1$	k' ya 1 değeri ver ve toplama ekle
2	$1 + 2 = 3$	k' yı 1 artır ve toplama ekle
3	$3 + 3 = 6$	k' yı 1 artır ve toplama ekle
...		
100	$4950 + 100 = 5050$	k' yı 1 artır ve toplama ekle



AKIŞ ŐEMALARI



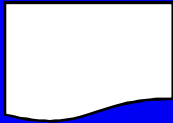
Başlama, Bitiş ve bağlantı İşlemleri



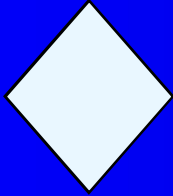
Giriş ve Okutma İşlemleri



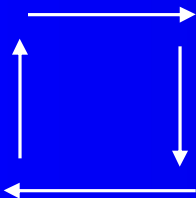
Atama ve Hesaplama İşlemleri



Yazdırma İşlemleri



Karar ve Kontrol İşlemleri



Akış yönünü belirten işlemler

Akış Őemalarında
kullanılacak Őekiller ve
bunların anlamı



ALGORİTMALAR

Algoritmaların Oluşturulması

Algoritmaların Özellikleri

Arama Algoritmaları

Sıralı Arama (Sequential Search) Algoritması

İkili Arama (Binary Search) Algoritması

Sıralama Algoritmaları

Bubble Sort

Shell Sort

Selection Sort

Quick Sort

Şifreleme Algoritmaları

Simetrik (Gizli) Anahtar Algoritmaları

Asimetrik Kripto-Algoritmalar

Algoritmaların Özellikleri

Giriş : Bir algoritma açıkça belirtilen bir kümeden giriş değerlere sahiptir.

Çıkış : Bir algoritmanın her bir giriş değerinin kümesinden, çıkış değerinin kümesi üretilir. Çıkış değerleri problemin sonucunu içerir.

Tanımlılık : Algoritmanın adımları tam olarak tanımlanmalıdır.

Sonluluk : Bir algoritma herhangi bir giriş kümesi için sonlu sayıdaki adımlardan sonra istenilen sonucu üretmelidir.

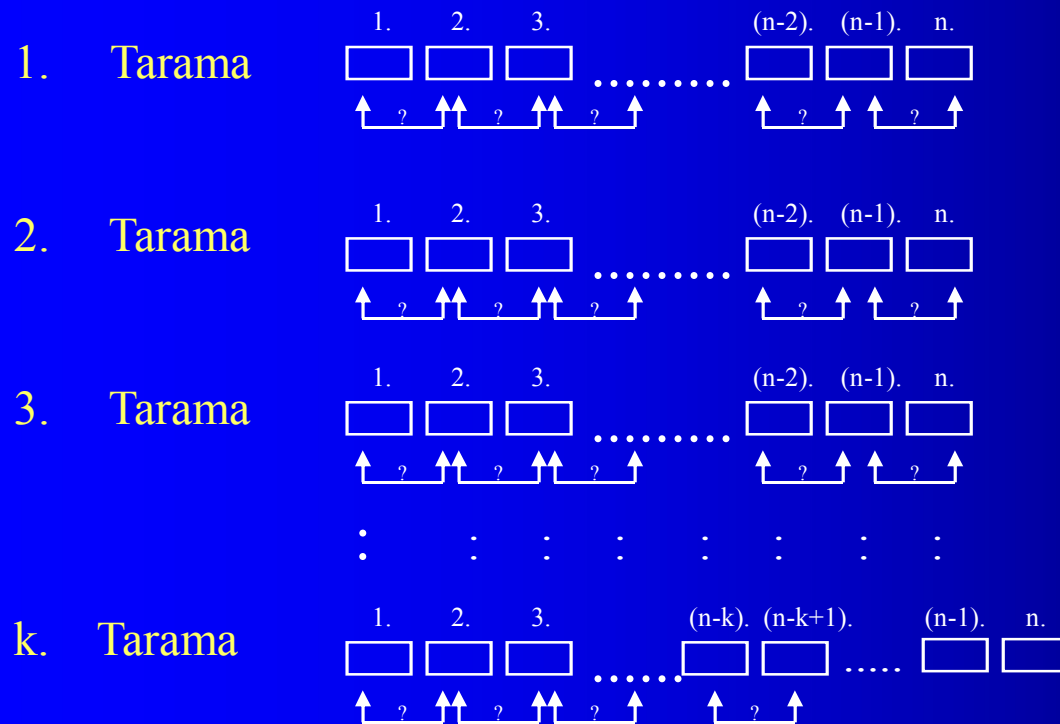
Etkinlik : Algoritmanın her bir adımı tam olarak ve sınırlı bir zamanda gerçekleşebilmelidir.

Genellik : Prosedür, sadece belli giriş değerleri için değil istenilen formdaki bütün problemler için uygulanabilir olmalıdır.



Bubble Sort

Dizide her bir eleman, sırasıyla kendisinden sonraki eleman ile karşılaştırılır ve gerektiğinde yer değiştirme yapılabilir. Bu sıralama işlemi, yer değiştirme olduğu sürece devam edecektir.



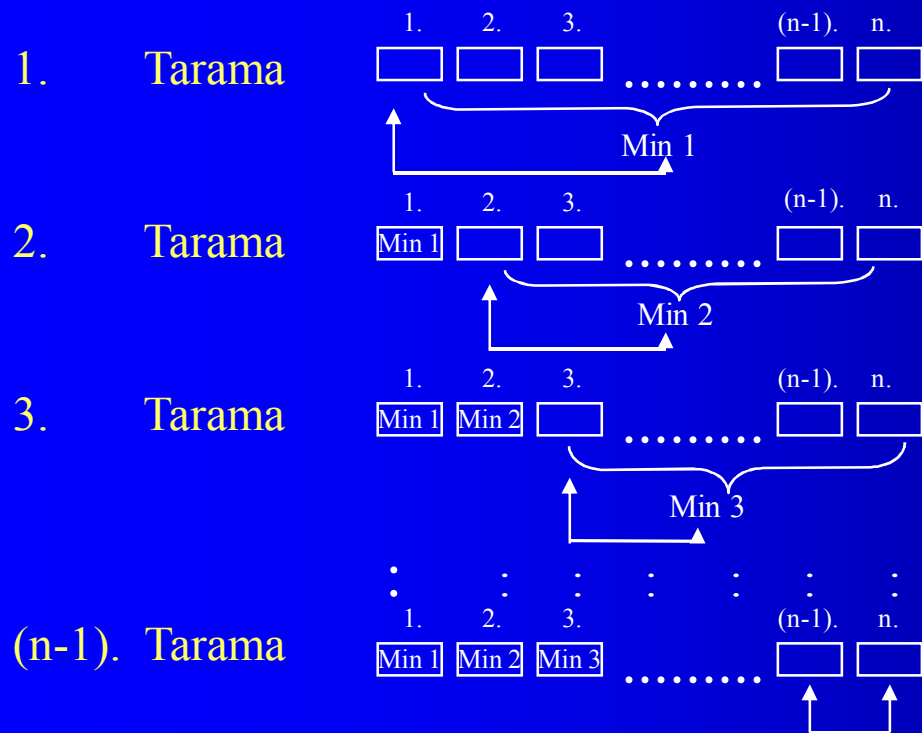
Örnek :

Sırasız Sayılar

Sırasız Sayılar			25	12	35	10	18
			1.	2.	3.	4.	5.
1.	Tarama	1.1	25	12	35	10	18
		1.2	12	25	35	10	18
		1.3	12	25	35	10	18
		1.4	12	25	10	35	18
			12	25	10	18	35
2.	Tarama	2.1	12	25	10	18	35
		2.2	12	25	10	18	35
		2.3	12	10	25	18	35
			12	10	18	25	35
3.	Tarama	3.1	12	10	18	25	35
		3.2	10	12	18	25	35
		3.3	10	12	18	25	35
4.	Tarama	4.1	10	12	18	25	35
			10	12	18	25	35

Selection Sort

Sırasıyla her bir eleman kendisinden sonraki elemanlardan minimum olanı ile yer değiştirir. Bu sıralamada geçerli olan minimum ile yer değiştirme, artan düzen içindir. Azalan düzende yapılacak sıralama için, her eleman kendisinden sonraki maksimum eleman ile yer değiştirilir.



Sırasız Sayılar

1. Tarama

2. Tarama

3. Tarama

4. Tarama

Sıralı Sayılar

1.	2.	3.	4.	5.
25	12	35	10	18

1.	2.	3.	4.	5.
25	12	35	10	18

Min=D=10

1.	2.	3.	4.	5.
10	12	35	25	18

Min=D=18

1.	2.	3.	4.	5.
10	12	35	25	18

Min=D=18

1.	2.	3.	4.	5.
10	12	18	25	35

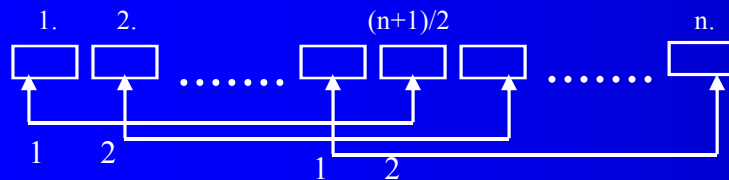
Min=D=18



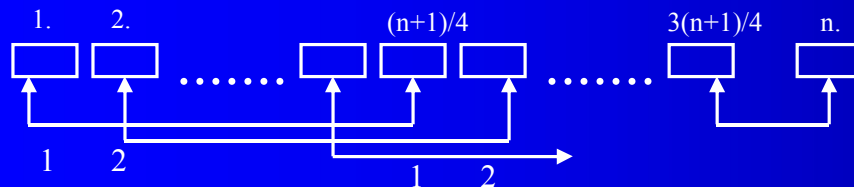
Shell Sort

Elemanlar dizisi ikiye bölünür. Bunun sol ve sağındaki elemanlar karşılıklı taraflar arası kontrol edilip yer değiştirilir. Bu işlem daha küçük ikili bölmelere ayırarak devam eder.

1. Tarama Aralık=(n+1) / 2



1. Tarama Aralık=(n+1) / 4



⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮

k. Tarama Aralık=1



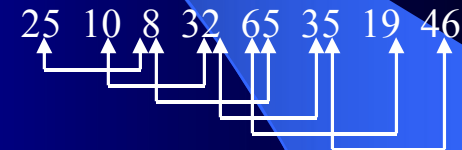
1. Tarama

Aralık:(1+8)/2=4



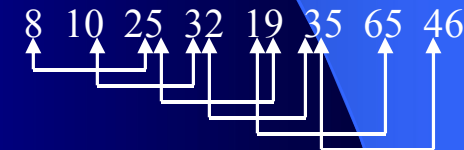
2. Tarama

Aralık:4/2=2



3. Tarama

Aralık:2



4. Tarama

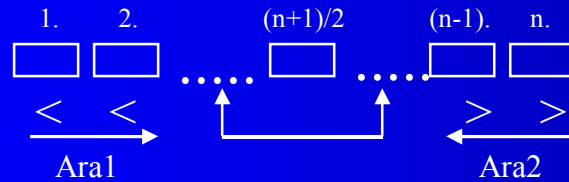
Aralık:2/2=1



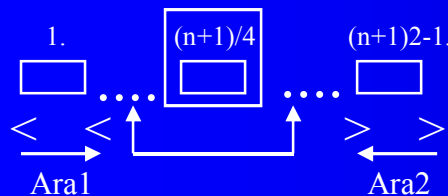
Quick Sort

Dizi ikiye bölünüp bir orta eleman alınır. Sol taraftaki bir bölmenin sıra düzenine uymayan orta değerden büyük ilk elemanı ile, sağ taraftaki bölmenin sıra düzenine uymayan orta değerden küçük ilk elemanı ile yer değiştirilir. Solu ve sağlı her bölme tekrar kendi içinde ikiye bölünüp, taraflar arası yer değiştirmelere devam edilir.

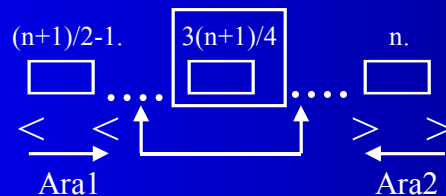
Hızlı Sıralama(1,n)



Hızlı Sıralama(1,((n+1)/2)-1)



Hızlı Sıralama(((n+1)/2)+1,n)

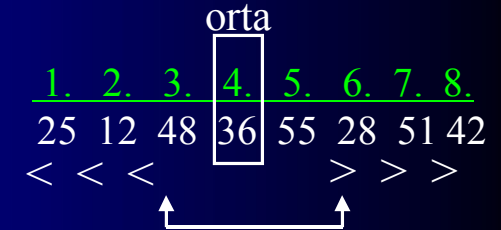


1. Tarama

Hızlısırala(1,8)

Ara1=1

Ara2=8



2. Tarama

Hızlısırala(1,3)

Ara1=1

Ara2=3

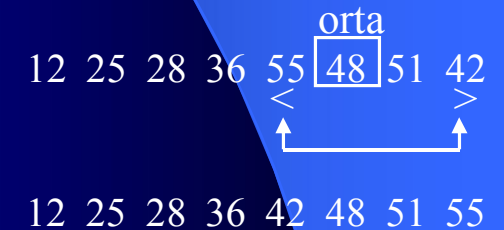


4. Tarama

Hızlısırala(5,8)

Ara1=5

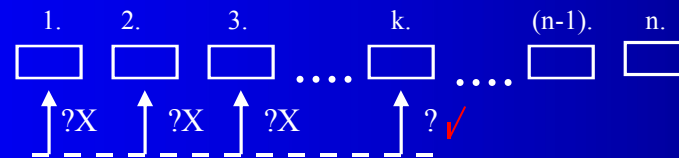
Ara2=8



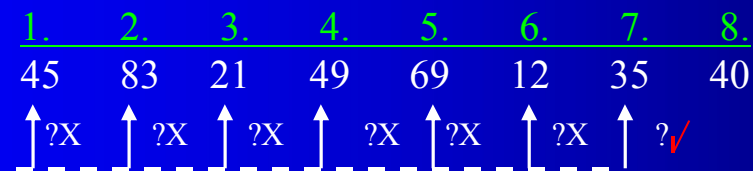
Sıralı Arama

Bu metodun uygulanmasında, arama yapılan dizinin sıralı ya da sırasız düzende olması önemli değildir. Aranılan eleman, dizinin ilkinden başlanıp bulunana kadar, teker teker karşılaştırılır. Bulduğunda arama işlemine son verilir. Aranılan elemanın yok olduğu, ancak dizinin baştan sona taranması ile anlaşılabilir.

Aranan: k.



Aranan: 35



Bulundu

Sıra: 7

Aranan: 25



Bulunamadı

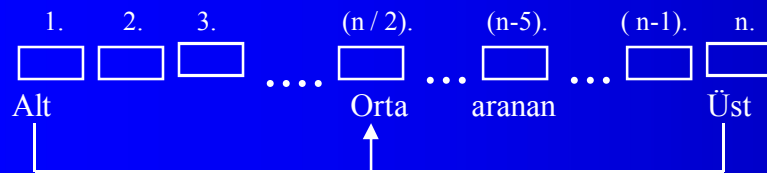
Sıra: 0



İkili Arama

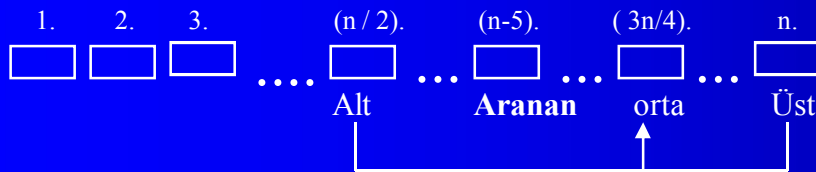
Bu metodun uygulanmasında arama yapılan dizinin sıralı düzende olması gerekir. İstenilen eleman, dizinin ortasından başlanarak aranır. Bu eleman, orta elemandan küçükse ilk yarısı, büyükse son yarısı, daha dar arama kesimi olarak ele alınır. Bu şekilde arama işlemi aranan bulununcaya yada daralmakta olan kesimin bitimine kadar sürer.

1. Arama



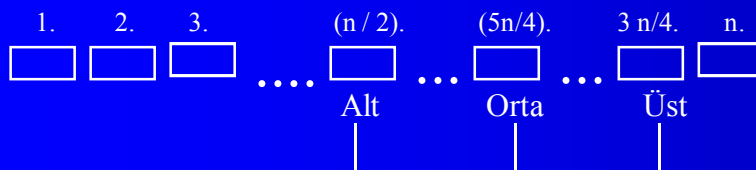
Eğer $D(n/2) = \text{Aranan}$ ise bulundu. Değilse 2. arama

2. Arama



Eğer $D(3n/2) = \text{Aranan}$ ise bulundu. Değilse 3. arama

3. Arama



Eğer $D(5n/8) = \text{Aranan}$ ise bulundu.
Değilse bulunamadı

Aranan : 35

1.	2.	3.	4.	5.	6.	7.	8.
12	21	35	40	45	49	69	83
alt			orta				üst

1.	2.	3.	4.	5.	6.	7.	8.
12	21	35	40	45	49	69	83
alt	orta	üst					

1.	2.	3.	4.	5.	6.	7.	8.
12	21	35	40	45	49	69	83
	alt	orta	üst				

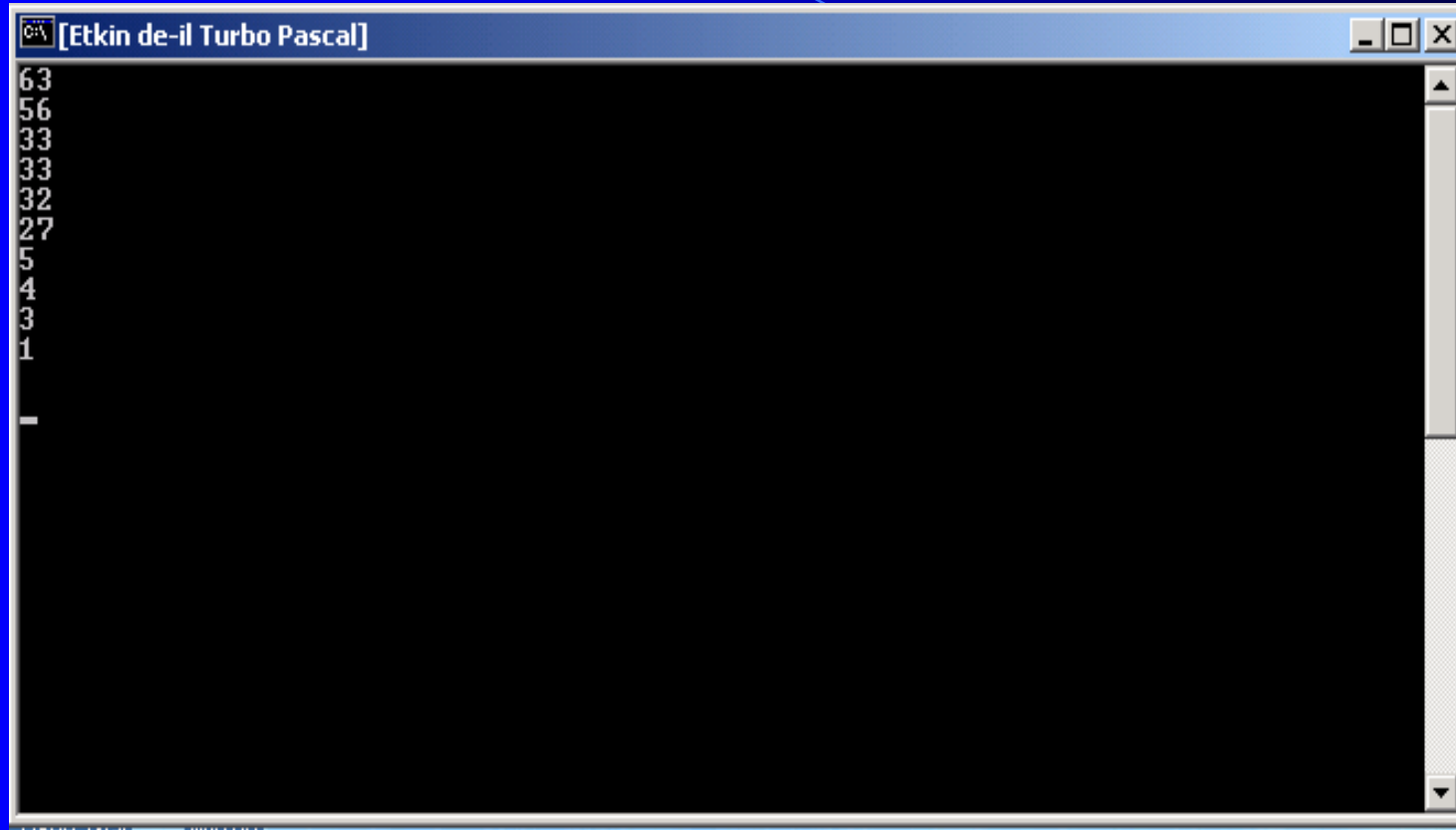


Program : Verilen dizideki elemanları büyükten küçüğe doğru sıralayan program

```
Program Bubble_Sort;  
Uses crt;  
Type  
    Stip = Array[1..10] of integer;  
Const  
    S:Stip = (27, 3, 4, 5, 32, 56, 33, 33, 63, 1);  
    N=10;  
Var  
    i, j: byte;  
Procedure Degis(var a,b: integer);  
Var  
    c:integer;  
Begin  
    c:=a; a:=b; b:=c;  
end;  
Procedure Bubble (var s:stip; N:integer);  
Begin  
    For i:=2 to n do  
        For j:=n downto i do  
            If s[j-1] < s[j] then degis (s[j-1], s[j]);  
End;  
Begin  
    Bubble(s,n);  
    Clrscr;  
    For i:=1 to n do writeln(s[i]); Readln;  
End.
```



Program Çıktısı :



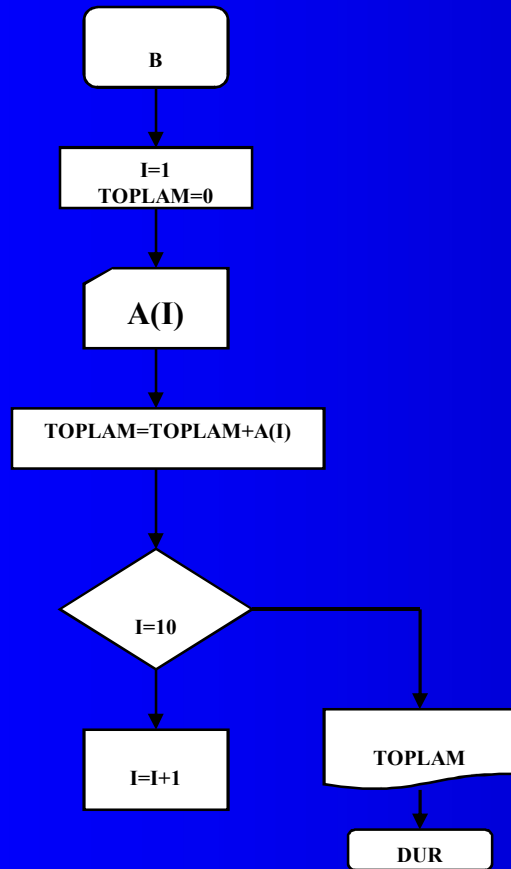
A screenshot of a Turbo Pascal console window titled "[Etkin de-il Turbo Pascal]". The window has a black background with white text. The text displays a list of numbers: 63, 56, 33, 33, 32, 27, 5, 4, 3, 1, followed by a hyphen on a new line. The window includes standard Windows-style window controls (minimize, maximize, close) in the top right corner and a vertical scrollbar on the right side.

```
[Etkin de-il Turbo Pascal]
63
56
33
33
32
27
5
4
3
1
-

```



Örnek:10 elemanlı bir sayı dizisinin elemanlarının toplamını bulan algoritma ve akış şemasının oluşturulması.



akış şeması

- A1. Başla,
- A2. $I=1, TOPLAM=0$ al,
- A3. $A(I)$ ' yı gir,
- A4. $TOPLAM=TOPLAM+A(I)$ al,
- A5. Eğer $I=10$ ise A7. adıma git,
- A6. $I=I+1$ al ve A3. adıma geri dön,
- A7. $TOPLAM$ ' ı yaz,
- A8. Dur.





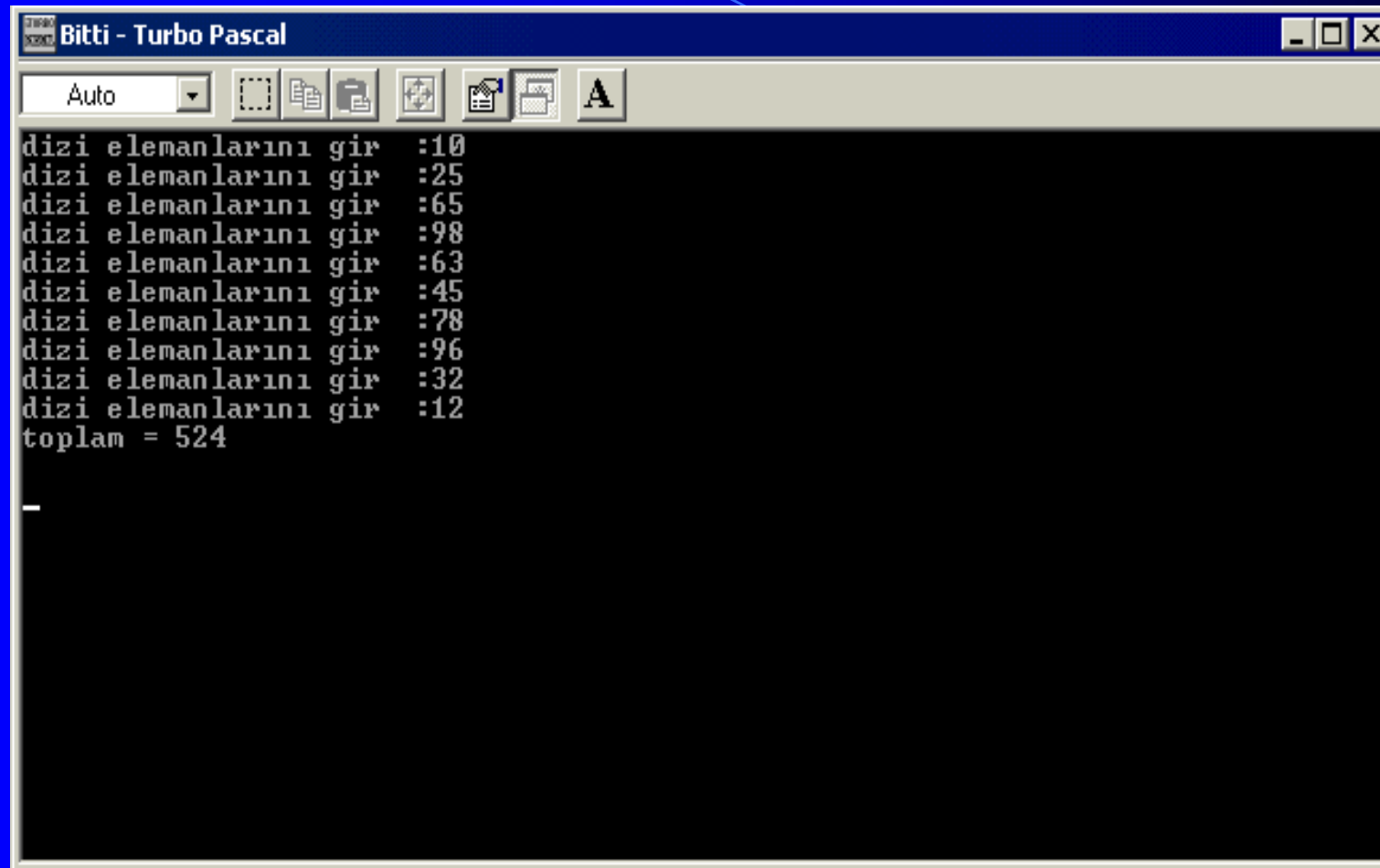
Algoritmanın Pascal Dilindeki Karşılığı:

```
Program dizi_toplamı;  
Var  
    Toplam,i:integer;  
    A:=array[1..10] of integer;  
Begin  
    Toplam:=0;  
    For i:=1 to 10 do begin  
        write('dizi elemanlarını gir :');  
        readln(a[i]);  
        toplam:=toplam+a[i];  
    end;  
    writeln('toplam = ',toplam);  
end.
```

Verilen örnekte A2. adımda, bir I indisi ve TOPLAM değişkeni tanımlamıştır. Burada tanımlanan I indisi 1' den 10' a kadar artırılarak A dizisinin elemanlarının girişi yapılmaktadır ve girilen her eleman TOPLAM değişkene ilave edilerek toplatılmaktadır. A3. adımda A dizisinin I' yıncı elemanı girilerek A4. Adımda bu eleman TOPLAM değişkene ilave edilmektedir. Bu işlem I indisi 10' a kadar devam etmektedir. Sorgulama işlemi A5. adımda yapılarak bu doğrultuda algoritma yönlendirilmektedir.



Program Çıktısı :



```
dizi elemanlarını gir :10
dizi elemanlarını gir :25
dizi elemanlarını gir :65
dizi elemanlarını gir :98
dizi elemanlarını gir :63
dizi elemanlarını gir :45
dizi elemanlarını gir :78
dizi elemanlarını gir :96
dizi elemanlarını gir :32
dizi elemanlarını gir :12
toplam = 524
_
```



DOSYALAMA SİSTEMLERİ

Dosyalama Sistemleri

Sırasal Erişimli Dosyalar

Doğrudan Erişimli Dosyalar

Sırasal Erişimli Dosyalara İlişkin Algoritma
ve Akış Şemaları

Doğrudan Erişimli Dosyalara İlişkin Algoritma
ve Akış Şemaları



DOSYALAMA SİSTEMLERİ

Dosyalama sistemlerini, bilgilerin kalıcı olmalarını sağlamak amacıyla, verilerin disk, disket ve kaset gibi manyetik yüzeylerde saklanarak gerektiğinde kullanılabilmesini sağlayan sistemler olarak adlandırılabilirler. Diğer bir ifade ile, genel olarak birbirleriyle ilişkili verilerin birer kayıt biçiminde saklandıkları ortam olarak da tanımlanabilir.



Sırasal Erişimli Dosyalar: Bilgiler kaydediliş sırasına göre dosya içerisinde yer alırlar. Bu yüzden istenilen bir bilgiye ulaşmakta sırasallık gerekmektedir. Ulaşılmak istenilen kayda ilişkin herhangi bir bilgi verilerek dosya baştan itibaren taranmak suretiyle istenilen bilgiye ulaşmak mümkün olmaktadır.

Sırasal erişimli dosyalarda istenilen bir kaydın silinebilmesi için geçici bir dosyanın oluşturulması gerekir. Silinecek kişiye ait herhangi bir bilgi girildikten sonra bu bilgiye göre dosya taranarak silinmesi istenilen kaydın dışındaki diğer kayıtlar geçici olarak adlandırılan dosyaya yazılır. Daha sonra ana dosya olarak adlandırılan ilk dosya silinerek geçici olarak oluşturulan dosya ana dosya olarak yeniden tanımlanır.

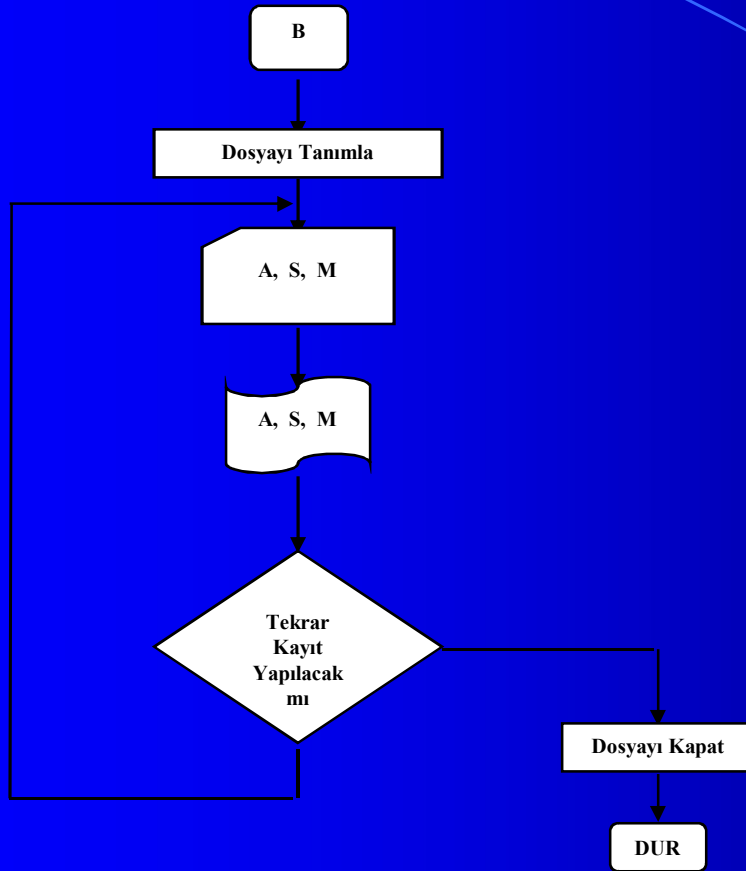
Sırasal erişimli dosyalarda istenilen bir kayda ulaşmak için, bu kayda ilişkin herhangi bir bilgi ile diğer kayıtların bu bilgilerinin karşılaştırılması gerekmektedir. Dolayısıyla çok fazla kayıt içeren dosyalarda bir bilgiye ulaşmak zaman kaybına yol açabilmektedir.

örnek :

Örnek: Sırasal erişimli bir dosyaya çalışanlara ait ad-soyad, sicil numarası ve maaş bilgilerini kaydeden algoritma ve akış şemasının oluşturulması

- A1. Başla,
- A2. Dosyayı Tanımla,,
- A3. A' yı gir {ad-soyad}
- A4. S' yi gir {sicil numarası}
- A5. M' yi gir {maaş}
- A6. A,S ve M' yi dosyaya yaz,
- A7. Tekrar kayıt yapılacak mı? { E ya da H }
- A8. Eğer 'E' ise A3. Adıma geri dön,
- A9. Dosyayı kapat ve dur.





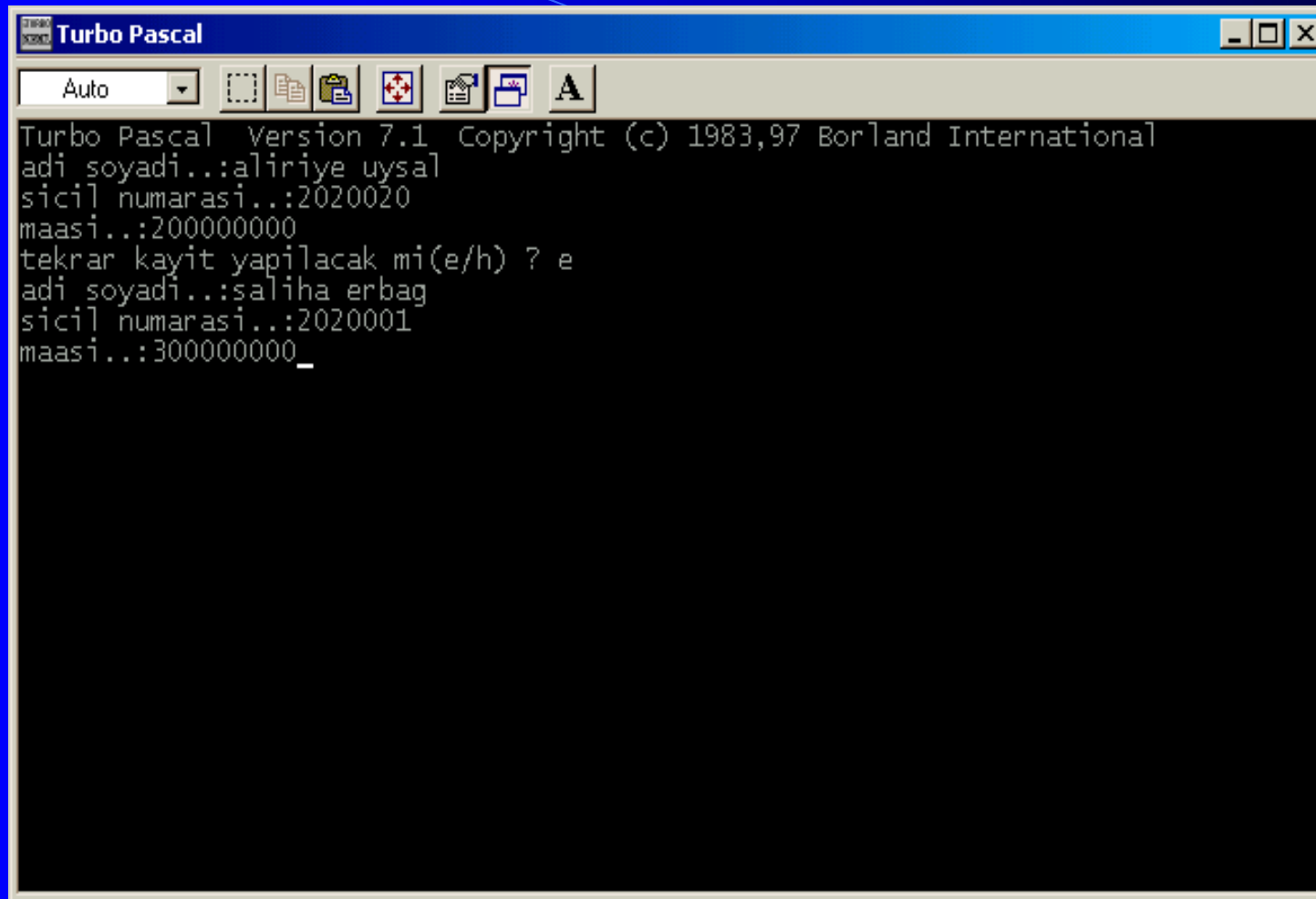
Sırasal erişimli bir dosyaya kayıt yapılmasına ilişkin akış şeması



Algoritmanın Pascal dilindeki yazılımı:

```
program kayıt;
var
    a:string[20];
    s:integer;
    m:longint;
    c:char;
    t:text;
begin
    assign(t,'bilgi'); rewrite(t);
    c:='e';
    while(c<>'h') do begin
        write('adı soyadı...:');
        readln(a);
        write('sicil numarası...:');
        readln(s);
        write('maası...:');
        readln(m);
        writeln(t,a);
        writeln(t,s);
        writeln(t,m);
        write('tekrar kayıt yapılacak mı(e/h) ? '); readln(c);
    end;
    close(t);
readln;
end.
```





```
Turbo Pascal Version 7.1 Copyright (c) 1983,97 Borland International
adi soyadi...:aliriye uysal
sicil numarasi...:2020020
maasi...:200000000
tekrar kayıt yapılacak mı(e/h) ? e
adi soyadi...:saliha erbag
sicil numarasi...:2020001
maasi...:300000000_
```



Verilen örnekte, A2. adımda kayıt amacıyla dosya tanımlanmaktadır. A3, A4 ve A5. adımlarda dosyaya kaydedilecek bilgilerin girişı yapılmaktadır. Klavyeden girilen bilgilerin dosyaya yazdırılması işlemi A6. adımda gerçekleştirilmektedir. A7. adımda dosyaya tekrar kayıt yapıp yapılmayacağı sorgulanmaktadır. Eğer girilen cevap E ise A3. adıma geri dönölerek yeniden bilgi girişı istenmektedir. Aksi halde dosya kapatılarak işlemlere son verilmektedir.



Doğrudan Erişimli Dosyalar: Bilgiler birer kayıt numarası ile dosya içerisinde saklanmaktadır. Bu kayıt numaraları bilgilerin adreslerini tanımlarlar. İstenilen bir kayda ulaşmak; o kayda ilişkin kayıt numarasının girilmesi ile mümkün olmaktadır. Bu tür dosyalama sistemleri sırasal erişimli dosyalara göre daha kullanışlıdır.

Doğrudan erişimli dosyalara kayıt yapılırken belirli bir kayıt sırası yoktur, yani kayıt numaralarının arka arkaya verilmesi gerekmez.. Örneğin önce 1 numaralı kayıt yapıldıktan sonra 20 numaralı kayıt arkasından yapılabilir. İstenilen kayda erişildiğinde bu kayıttaki ilgili bilgi kontrol edilerek başlangıçta verilen işaretin olup olmadığı sorgulanabilir ve bu doğrultuda ilgili kaydın boş olup olmadığı anlaşılabilir.

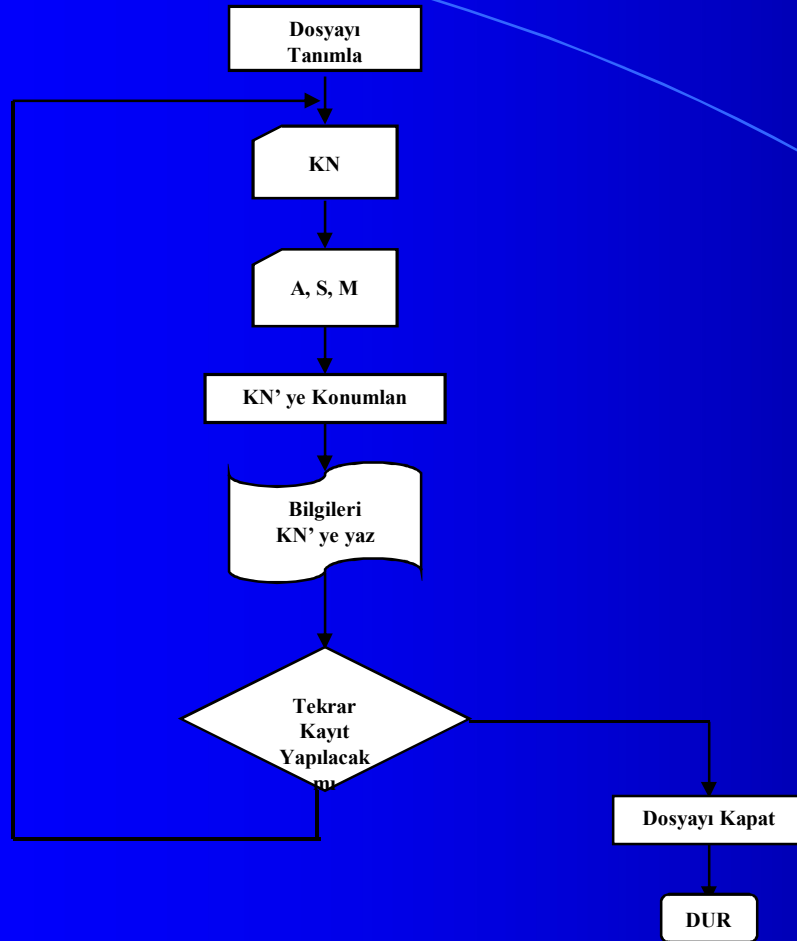
Doğrudan erişimli dosyalarda istenilen bir kayda ulaşmak sırasal erişimli dosyalara göre daha kolaydır. Bu tür dosyalamada karşılaştırma işlemleri yoktur. İstenilen kayda ilişkin kayıt numarası verilerek doğrudan o kayda ulaşmak mümkün olmaktadır.

Örnek :

Örnek Doğrudan erişimli bir dosyada ad soyad, sicil numarası ve maaş bilgilerini kaydeden algoritma ve akış şemasının oluşturulması

- A1. Başla,
- A2. Dosyayı tanımla,
- A3. KN' yi gir {kayıt numarası},
- A4. A' ya git {adı soyadı},
- A5. S' yi gir {sicil numarası},
- A6. M' yi gir {maaş},
- A7. KN' ye konumlan,
- A8. Dosyayı yaz,
- A9. Tekrar kayıt yapılacak mı? {E ya da H},
- A10. Eğer E ise A3. adıma geri dön,
- A11. Dosyayı kapat,
- A12. Dur.





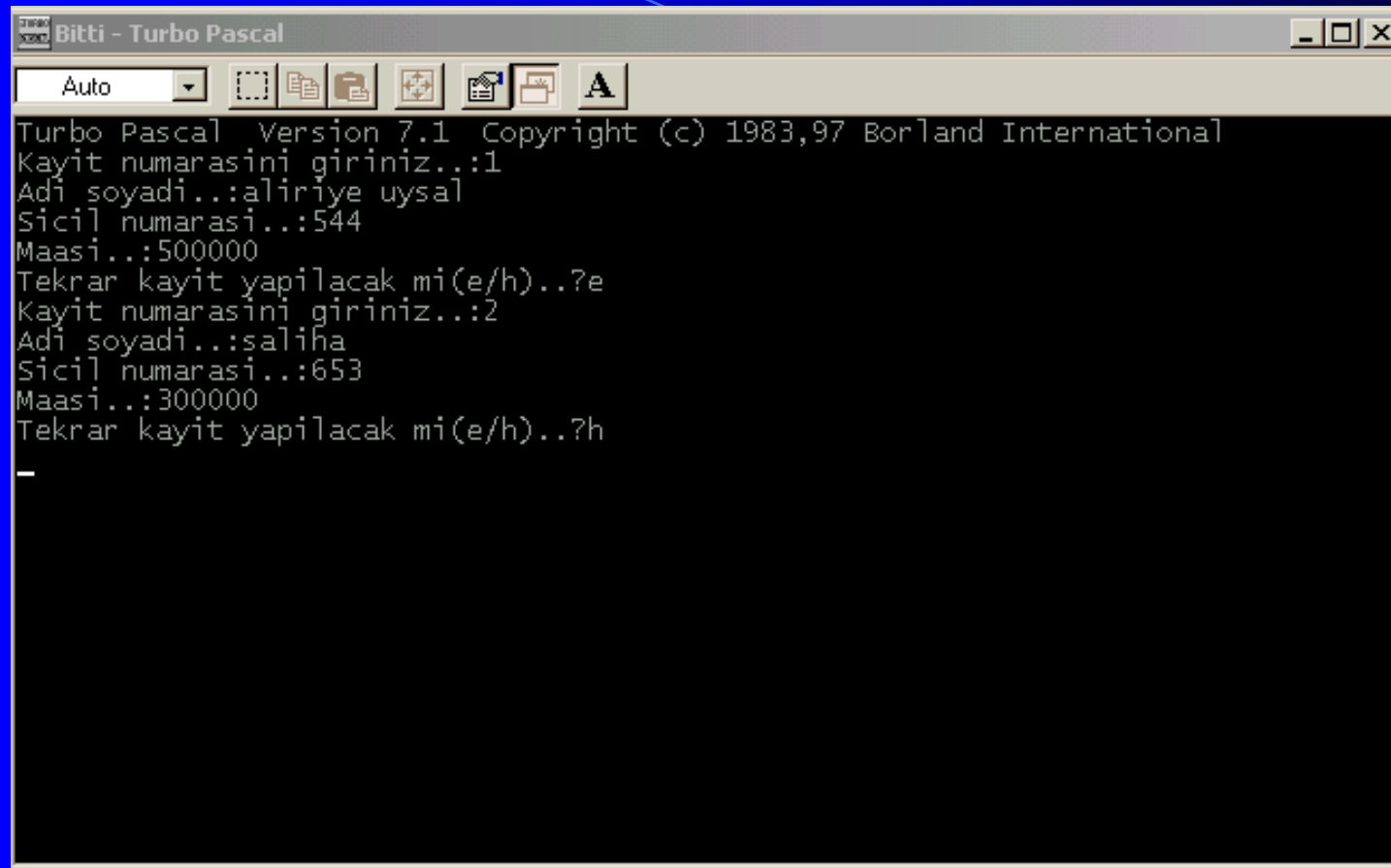
Doğrudan erişimli dosyada kayda ilişkin akış şeması



Algoritmanın Pascal dilindeki yazılımı:

```
program kayıt;  
type  
    bilgi=record  
        a:string[20];  
        s,m:longint;  
    end;  
var  
    dosya:file of bilgi;  
    kay:bilgi;  
    kn:integer;  
    c:char;  
begin  
    assign(dosya,'maas.dat');  
    {$I-}; reset(dosya); {$I+};  
    if(IOResult<>0) then rewrite(dosya);  
    c:='e';  
    while(c<>'h') do begin  
        write('Kayit numarasini giriniz...');readln(kn);  
        write('Adi soyadi...');readln(kay.a);  
        write('Sicil numarası...');readln(kay.s);  
        write('Maasi...');readln(kay.m);  
        seek(dosya,kn);  
        write(dosya,kay);  
        write('Tekrar kayıt yapılacak mi(e/h)..?');readln(c);  
    end;  
    close(dosya);  
end.
```





```
Turbo Pascal Version 7.1 Copyright (c) 1983,97 Borland International
Kayit numarasini giriniz...:1
Adi soyadi...:aliriye uysal
Sicil numaras...:544
Maasi...:500000
Tekrar kayit yapilacak mi(e/h)..?e
Kayit numarasini giriniz...:2
Adi soyadi...:saliha
Sicil numaras...:653
Maasi...:300000
Tekrar kayit yapilacak mi(e/h)..?h
-
```



Sırasal erişimli dosyalardan farklı olarak bu tür dosyalama sistemlerinde bilgiler önceden verilen kayıt numaraları ile belirlenen adreslere kaydedilmektedir.

Bu doğrultuda A3. adımda verilen KN değişkeni kaydedilecek bilgilere ilişkin kayıt bölgesinin yerini tanımlamak üzere girilmektedir. Sonraki adımlarda girilen bilgiler sonucunda, A7. adımda dosyadan KN' ye konumlanarak girilen bilgilerin buralara yazılması sağlanmaktadır. Bu işlemler istenildiği kadar kaydın yapılmasına kadar devam etmektedir.



Örnek:Girilen iki sayının OBEB ve OKEK'ini bulan pogramın yazılması.

Uses crt;

Var

a,b,t,k:byte;

Begin

Clrscr;

Write('1. sayiyi giriniz : '); readln(a);

Write('2. sayiyi giriniz : '); readln(b);

t:=a*b;

if (a<b) then begin

k:=a;

a:=b;

b:=k;

end;

repeat

k:=b;

b:=a mod b;

a:=k;

until b=0;

writeln('Obeb : ',a);

write('Okek : ',t/a:6:5);

readln;

end.



```
1. sayiyi giriniz : 2  
2. sayiyi giriniz : 6  
Obeb : 2  
Okek : 6.000000
```



