BÖLÜM 1

GİRİŞ

Bilgisayarların istenilen amaçlara cevap verebilmesi için doğru programlanması gerekir. Mevcut problemleri çözmek için etkili algoritmalar geliştirmek günümüzün gelişen teknolojisi ile birlikte oldukça önem kazanmıştır. Dolayısıyla problemlerin çözümüne ulaşabilmek için algoritmaların analizinin iyi irdelenmesi gerektiği bir gerçektir.

Bir programın oluşturulmasındaki temel adımlara bakıldığında; ilk aşama verilen probleme ilişkin verilerin ve istenilenlerin doğru tahmin edilmesidir. Daha sonra problemi çözmeye yönelik çözüm yöntemlerinin araştırılması ve gerekli yöntemin tespitinden sonra problemin çözümüne ilişkin mantıksal akışların oluşturulması gerekir. Bu mantıksal akışlara algoritma adı verilmektedir. Algoritmalar çözümde esas alınan aritmetik ve mantıksal işlemleri birleştirerek problemin çözümüne ulaşabilmeye sağlayan ön programlar olarak düşünülebilir. Algoritmalar, her bir adımının geometrik şekillerle ifade edilmesi anlamına gelen akış şemaları ile desteklendiğinde çözüm akışının daha net izlenmesi mümkün olmaktadır. Problemin çözümüne ilişkin algoritma ve akış şemaları oluşturularak doğru olduğuna karar verildikten sonra bilgisayarın anlayacağı dilde kodlanma işlemlerine geçilir.

Kitabın ikinci bölümünde algoritmaların tanımı, genel yapısı ve tipleri hakkında bilgi verilerek, üçüncü bölümde algoritmaların oluşturulması, özellikleri ve algoritma tiplerine (arama algoritmaları, sıralama algoritmaları, şifreleme algoritmaları vb.) değinilmiştir. Dördüncü bölümde, algoritmalar ve akış şemaları birlikte ele alınmıştır. Değişken tanımlama, aritmetik işlemler, diziler ve matrisler gibi konulara değinilerek örnekler sunulmuştur. Beşinci bölümde, dosyalama sistemleri (sırasal erişimli, doğrudan erişimli) anlatılmış ve altıncı bölümde de çeşitli konularda Pascal dili ile hazırlanmış algoritmalara yer verilmiştir.

Kitapta, daha çok örneklere yer verilerek konunun anlaşılırlığını arttırmak amaçlanmıştır.

BÖLÜM 2

2. ALGORİTMALARAGENEL BAKIŞ

2.1. ALGORİTMANIN TANIMI

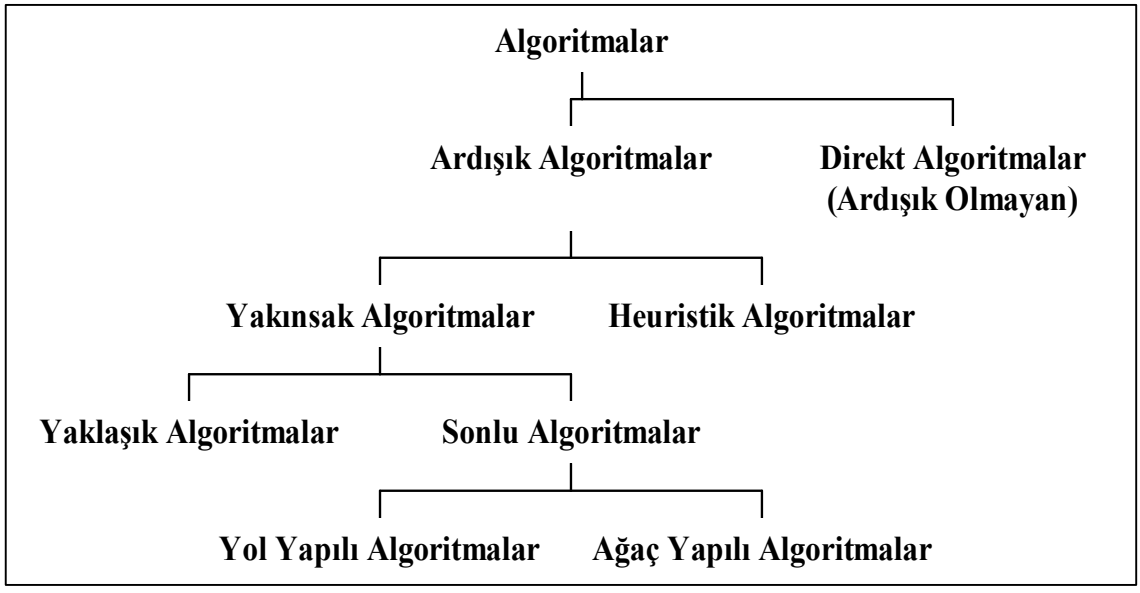
Bir algoritma için yaygın olarak kullanılan tanımlardan bazıları şunlardır:

- 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.

Bir algoritma, belirli bir problem için tatmin edici bir çözüme yakınsamayı sağlayan bir prosedürdür.

2.2. ALGORİTMA TİPLERİ

Algoritmalar, gerek yapıları gerekse çalışma durumlarına göre farklılıklar gösterirler ve şekildeki gibi ayrılırlar.



a) Direkt (Ardışık Olmayan) Algoritmalar

İterasyonlarda çalışmayan algoritmalar, direkt algoritmalar olarak adlandırılır. $y= ax^b$, $(a>0)$ polinomunun türevi $y'= abx^{b-1}$ bunun basit bir örneğidir.

b) Ardışık Algoritmalar

Belirli alt prosedürlerin pek çok kez tekrarlandığı algoritmalardır ve ardışık çalışırlar. Algoritmaların çoğu ardışık olarak çalışır.

c) Yakınsak Algoritmalar

Aranılan çözüme doğru yakınsayan ardışık algoritmalardır.

d) Yaklaşık Algoritmalar

Bazı yakınsak algoritmalar kesin çözümü elde edemezler, fakat bu çözüme yaklaşık bir değeri kesin çözüm alırlar. Yaklaşık algoritmalar sonlu değillerdir, fakat herbir ileri iterasyon onları kesin çözüme biraz daha yaklaştırır. Yaklaşık algoritmalara Değişken Kesen Metodu, Arama Teknikleri vb. çok bilinen birkaç örnek verilebilir.

e) Sonlu Algoritmalar

Bu algoritmalar, iterasyonların sonlu bir sayısında kesin çözümü garanti eden yakınsak algoritmalardır ve kendi arasında yol yapılı ve ağaç yapılı olmak üzere ikiye ayrılırlar.

1. Yol Yapılı Algoritmalar: Sonlu algoritmaların pekçoğu yol yapısına sahiptir; bu yol yapısında bir iterasyon bir önceki iterasyonu iterasyon dizilerinde farklı dallar üretmeksizin takip eder.

Böyle algoritmaların örnekleri, Tamsayı Programlamada Kesme Düzlem Algoritmalarının pekçoğu, Şebekelerde Maksimum Akış Algoritmaları, pekçok En Kısa Yol Algoritmaları ve Lineer Programlamanın Simpleks Algoritmaları ve herhangi pivot tekniği ile matris tersleridir.

2. Ağaç Yapılı Algoritmalar: Diğer sonlu algoritmalarda iterasyon dizileri, pekçok paralel dalları içeren bir ağaç şeklindedir. Bir çok ağaç arama algoritmaları bu sınıfa aittir. Bu arama algoritmalarının bazıları, Dinamik Programlama, Dal ve Sınır, Sınırlı Sayım, Kapalı Sayım, Dal ve Çıkarma, Dal ve Atma vb. olarak sıralanabilir.

Yaklaşık algoritmalar bu yol yapısına doğru yönelir. Diğer yandan heuristikler ise bir yol yapısı ile son bulabilen indirgenmiş bir ağaç yapısı olarak gözönüne alınabilir.

2.3. ALGORİTMALARIN YAPISI

Bir problemi çözmek için adım adım uygulanacak bir prosedür olarak tanımlanan algoritmanın tipik adımları;

I. Atama adımları, (Bir değişkene bazı değerlerin atanması gibi)
II. Aritmetik adımlar, (Toplama,bölme,çıkarma,çarpma gibi)
III. Mantıki adımlardır. (İki sayının karşılaştırılması gibi)

Genel yapısı bu şekilde kısaca özetlenen algoritmalar, belirli bir hata kabulü içinde yaklaşık cevap verirler; uygun programlama dili seçildiğinde genellikle hızlı çalışırlar.

Bir algoritmanın sahip olduğu adımların sayısı (ki bu, algoritmanın gereksinimi olan zamanı büyük ölçüde belirler) problemin bir örneğinden diğerine farklılık gösterir. Bu ilgili algoritmanın performansına bağlı bir faktördür.

2.4. ALGORİTMALARIN DİLİ

Kodlama :

1) Procedure = Bir algoritmanın kodlanmasına başlanan ilk ifadedir .Bu ifadede algoritmanın ad Örnek: Procedure max(L = list of integers)

Burada algoritmanın adı max iken tamsayıların L listesinin maksimumunu bulur.

2) Assignments=Atamalar ve ifadelerin diğer tipleri

Assigments ifadesi değişkenlere değer atamada kullanılır .Bu ifadede sol taraf değerin adını alırken sağ taraf ise prosedürlerle tanımlanan fonksiyonları , değerleri atanan değişkenleri , sabitleri içeren bir ifade yada deyimdir .Sağ tarafta ayrıca aritmetik işlemlerin herhangi biride bulunabilir.

: = atamalar için semboldür.

Böylece ;

Variable : = expression

Max: = a (a'nın değeri max değişkenine atanır.)

Ayrıca, x : = Largest integer in the list L şeklinde bir ifade de kullanılır.

x ' i L'de en büyük tamsayı olarak atar.

Güncel bir programlama diline bu ifadeyi dönüştürmek için birden fazla ifade kullanmalıyız . Bunun için ; interchange a and b kullanılır.

Karmaşık prosedürler için ifade blokları kullanılır . Bu begin ile başlar ve end ile sona erer.

```
Begin
Statement 1
Statement 2
.....
        Statement n
    end.
```

Şartlı Yapılar

1) if condition then statement

veya

if condition then

```
begin
  blok of statements
end
```

Eğer durum doğru ise verilen ifade gerçekleştirilir

2) if condition then statement1 else statement 2

Genel form: İf condition 1 then statement 1 else if condition 2 then statement 2 else if condition 3 then Statement 3……..

Else if condition n then statement n else statement n+1

Eğer durum 1 doğruysa ifade 1 gerçekleştirilir v e programdan çıkılır .Eğer, durum1 yanlışsa program durum2 'nin doğruluğunu kontrol eder .Eğer durum2 doğruysa ifade2
gerçekleştirilir . Böylece devam edilir. Sonuç olarak ,eğer durum1 ile durum – n'nin hiçbiri doğru değilse (n+1).ifade yerine getirilir.

Döngü Yapıları

1) **for**
2) **while**

1) for variable : = initial value to final value
statement
veya
for variable : = initial value to final value
begin
block of statements
end.

Eğer başlangıç değer sonuç değerden küçük yada eşitse değişken olarak başlangıç değer atanır ve sonuç değer değişken atanana kadar bu döngü gerçekleştirilir.Eğer, başlangıç değer sonuç değeri aşarsa döngü olmaz .Örneğin;

```
sum : = 0
for i : = 1 to  n
  sum : = sum+i
```

2) while condition statement
veya while condition
begin
block of statements
end şeklindedir. Burada durum doğruysa ifade gerçekleştirilir ve durum yanlış olana kadar bu döngü sürdürülür .

BÖLÜM 3

3. ALGORİTMALAR

3.1. ALGORİTMALARIN OLUŞTURULMASI

Sonlu bir tamsayı dizisinin en büyük elemanını bulmak için bir örnek algoritmayı ele alalım.

Örnek 3.1: Bir dizideki en büyük elemanı bulma problemi oldukça açık olmasına rağmen, algoritma kavramının anlaşılması açısından güzel bir örnek oluşturur. Ayrıca, sonlu bir tamsayı dizisindeki en büyük elemanı bulmayı gerektiren birçok algoritma örneği vardır. Örnek olarak bir üniversitede binlerce öğrencinin katıldığı rekabete dayalı bir yarışmada en yüksek skorun bulunmasına ihtiyaç duyulabilir veya bir spor organizasyonunda her ay yüksek reyting alan üyenin tanıtılması istenebilir. Biz sonlu bir tamsayı dizisindeki en büyük elemanı bulmakta kullanılacak bir algoritma geliştirmek istiyorlar.

Çözüm 3.1: Aşağıdaki adımları yerine getiriyoruz.

1. Maksimum değerini geçici olarak dizinin ilk elemanına eşitliyoruz. (Geçici maksimum değeri prosedürün herhangi bir adımında sınanmış en büyük tamsayıdır)
2. Dizinin bir sonraki elemanıyla geçici maksimum değerini karşılaştırıyoruz, eğer dizinin bir sonraki elemanı geçici maksimum değerinden büyükse, geçici maksimum değerine bu sayıyı atıyoruz.
3. Eğer dizide başka sayılar varsa önceki adımları tekrarlıyoruz.
4. Dizide karşılaştıracak tamsayı kalmadığında duruyoruz. Bu noktada geçici maksimum dizideki en büyük tamsayıdır.

Ayrıca bilgisayar dilini kullanarak algoritma tanımlanabilir. Böyle bir şey yapıldığında sadece o dildeki komutların kullanılmasına izin verilir. Bu da algoritma tanımını karmaşık ve anlaşılması zor bir hale getirir. Üstelik birçok programlama dilinin genel kullanımında, özel bir dil seçmek istenmeyen bir durumdur. Bu yüzden algoritmaları belirtirken özel bir bilgisayar dili yerine pseudocode kullanılır. Ayrıca bütün algoritmalar İngilizce tanımlanır.

Pseudocode bir algoritmanın İngilizce dil tanımı ve bu algoritmanın programlama diline dönüştürülmesi arasında bir ara basamak oluşturur. Programlama dilinde kullanılan komutlar bu noktada geçici maksimum değeri dizinin en büyük elemanıdır. İyi tanımlanmış işlemleri ve durumları içerir. Herhangi bir bilgisayar dilinde başlangıç noktasında pseudocode tanımlanarak bir bilgisayar programı üretilebilir.

3.2. ALGORİTMALARIN ÖZELLİKLERİ

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.

Sonlu bir tamsayı dizisindeki maksimum elemanı bulan algoritmanın özelliklerini şöyle ifade edebiliriz. Algoritmadaki girdi tamsayı dizisidir. Çıktı dizideki en büyük tamsayıdır. Algoritmadaki her adım tam olarak tanımlanmıştır, atamalar, sonlu döngü ve şarta bağlı durumlar yer almaktadır.

Algoritma sınırlı bir zamanda karşılaştırma ve atama adımlarının her birini gerçekleştirmektedir. Sonuç olarak algoritma geneldir ve herhangi bir sınırlı tamsayı dizisindeki maksimum sayıyı bulmak için kullanılabilir.

3.3. ARAMA ALGORİTMALARI

Sıralanmış bir listedeki bir elemanın yerini bulma problemi pek çok konuda karşımıza çıkar. Sıralanmış kelimelerin listesinin bulunduğu bir sözlükte, kelimelerin hecelenmesini kontrol eden bir program bunun en basit örneklerinden biridir. Bu tür problemler arama problemleri olarak adlandırılır. Bu bölümde arama için birkaç algoritma ele alınacaktır.

Genel olarak arama problemleri şöyle tanımlanır. Farklı a1,a2,....,an elemanlarının listesinden x' in yerini bulur ya da listede olmadığını belirler. Bu arama problemi için çözüm dizi içerisinde x' e eşit olan sayının yeridir. Eğer x=ai ise çözüm 1' dir. x=0 ise dizide yoktur. İlk algoritma sıralı arama algoritmadır. Sıralı arama algoritması x ve a1' i karşılaştırarak başlar. x=a1 ise çözüm a1' in yeridir. Eğer, x a1' e eşit olmazsa x, a2 ile karşılaştırılır. Çözüm, a2'nin yeridir. Eğer, x, a2' ye eşit olmazsa x, a3 ile karşılaştırılır. Bir eşleme oluncaya kadar listedeki her bir eleman ile x' i karşılaştırma işlemine devam edilir. Çözüm x' e eşit olan terimin yeridir. Eğer, tüm liste arandığında x' in yeri yoksa sonuç 0' dır.

3.3.1. SIRALI ARAMA (SEQUENTIAL SEARCH) ALGORİTMASI

Dizinin sıralı veya sırasız olması önemli değildir. Dizinin ilk elemanından başlayarak aranan eleman bulununcaya kadar işlem devam eder. Dizide 1000 elemanın olması durumunda ve aranan eleman 900. sırada ise bu durumda 900 adet test işleminin yapılması gerekir. Bu da zaman alıcı bir işlemdir. Şimdi bu sıralama yöntemi ile ilgili programı yazalım:

```
Program Sequential_Search;
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,yer, ara: integer;
Function SiraliAra(ara:integer; s:stip): integer;
Begin
        For i:=1 to n do
                if  s[i]=ara then begin
                        siraliara:=i;
                        exit;
                end;
        siraliara:=0;
end;
Begin
        Write('Aradığınız sayı : ');    readln(ara);
        Yer:=SiraliAra(ara,s);
        if yer=0 then writeln(ara,' kayıtlı değil')
        else writeln(ara,'sayısı dizinin ',yer,'. elemanı');
        Readln;
End.
```

Bu programda sabir olarak 10 adet sayı S dizi değişkenine aktarılmıştır. Program kullanıcıdan bir sayı istemekte ve bu sayının olup olmadığını araştırmak için SiraliAra isimli Function'a gitmektedir. Sayının bulunması durumunda sayının indis numarası, bulunmaması durumunda ise 0 değeri gönderilmektedir.

3.3.2. İKİLİ ARAMA (BINARY SEARCH) ALGORİTMASI

Bu arama yöntemi sıralı olan diziler üzerinde arama yapar. Sıralama işleminin küçükten büyüğe doğru olması durumu için işlem anlatılacaktır. Arama işlemine dizinin ortasındaki eleman ile başlanmakta ve bir test işlemi ile sayının ortasındaki elemandan önce veya sonra olduğu belirlenmektedir. Eğer aranan sayı ortadaki elemandan sonra ise, orta elemanın indis numarası alt değer, değilse üst değer olarak alınmaktadır. Bu üst ve alt değerlerin ortası bulunmakta ve buna göre karşılaştırma işlemine devam edilmektedir. Yani her defasında aralık ikiye bölünür. Buna göre bir alt ve üst değer belirlenir. Bu alt ve üst değer arasında kalan alanda arama yapılmaktadır. Dizinin diğer kısmı için arama yapılmasına gerek kalmamaktadır. Binary Search yöntemine göre arama yapan program aşağıda verilmiştir.

```
Program Binary_Search;
Uses crt;
Type
        Stip = Array[1..10] of  integer;
Const
        S:Stip = (3, 10, 24, 14, 37, 66, 63, 25, 80, 23);
        N=10;
Var
        yer, ara: integer;
Procedure Selection (var s:stip; N:integer);
Var
        i,j: integer;
```

```
Procedure Degis(var a,b : integer);
Var
        c:integer;
Begin
        c:=a;    a:=b;      b:=c;
end;
Begin    {selection}
        For i:=1 to n-1 do
                For j:=i+1 to n do
                        if  s[i] > s[j] then degis(s[i],s[j]);
end;
Function IkiliAra(ara:integer; s:stip): integer;
Var
        Orta,alt,ust:integer;
Begin
        Alt:=1;
        Ust:=n;
        İkiliara:=0;
        While ust>alt+1 do begin
                Orta:=(ust+alt) div 2;
                if  s[orta]=ara then begin
                        ikiliara:=orta;
                        exit;
                end;
                if ara<s[orta] then ust:=orta
                else alt:=orta;
        end;
        if ara=s[ust] then ikiliara.=ust
        else if ara=s[alt] then ikiliara:=alt
        else ikiliara:=0;
end;
Begin
        Selection(s,n);
        Write('Aradığınız sayı : ');    readln(ara);
        Yer:=ikiliAra(ara,s);
        if yer=0 then writeln(ara,' kayıtlı değil')
        else writeln(ara,'sayısı dizinin ',yer,'. elemanı');
        Readln;
End.
```

3.4. SIRALAMA ALGORİTMALARI

Bir kümedeki elemanların sıralanması problemi çok çeşitli konularda ortaya çıkar. Örneğin, telefon idaresi bir rehber bastırmak istese abonelerinin isimlerini alfabetik olarak sıralaması gerekir.

Bir kümedeki elemanların hepsinin sıralı olduğunu varsayalım. Sıralama, bu elemanların artan bir sırada bir liste içinde yeniden sıralaması olarak tanımlanır. Örneğin, 7, 2, 1, 4, 5, 9 elemanlarından oluşan bir listeden 1, 2, 4, 5, 7, 9 şeklinde sıralanmış yeni bir liste üreterbiliriz. Başka bir örnek olarak, d, h, k, m, n, a harflerinden

oluşan listeden alfabetik sıra gözönüne alınarak a, d, h, k, m, n şeklinde yeni bir liste elde ederiz.

Bu sıralamaları farklı şekillerde sağlayabiliriz. Kullanılan başlıca sıralama metodları aşağıda verilmiştir.

3.4.1. Bubble Sort

İki farklı sıralama şekli mevcuttur :

Bubble Sort-1 : Bu sıralama yönteminde işlem dizinin en son elemanından başlayarak 2. elemana kadar, her eleman kendisinden bir önceki elemanla test ediliyor. Verilen şarta uyan elemanlar bulunduğu takdirde elemanların dizi içerisindeki yerleri değiştiriliyor. Her bir turda geriye kalan sayıların en büyük veya en küçüğü bulunur. Bu işlem dizideki bütün elemanlar için tekrarlanır. Bu algoritmayı program haline getirelim.

```
Program Bubble_Sort1;
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 Bubble (var s:stip; N:integer);
Procedure Degis(var a,b: integer);
Var
        c:integer;
Begin
        c:=a;
        a:=b;
        b:=c;
end;
Begin
        For i:=2 to n do
                For j:=n downto i do
                        İf  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.
```

Bubble Sort-2 : Bu sıralama tekniğinde her eleman kendisinden bir sonraki elemanla test ediliyor. Verilen şartın küçük ya da büyük olma durumuna göre şarta uyan elemanların yerleri değiştiriliyor. Bu işlem dizi sonuna kadara devam eder. Eğer baştan

sona kadar hiçbir yer değiştirme işlemi yapılmamışsa dizi sıralanmış demektir. Aksi halde dizinin başından itibaren test işlemine devam edilir.

```
Program Bubble_Sort2;
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: byte;
Procedure Bubble2 (var s:stip; N:integer);
Var
        Tamam: boolean;
Procedure Degis(var a,b: integer);
Var
        c:integer;
Begin
        c:=a;
        a:=b;
        b:=c;
end;
Begin
        Repeat
                Tamam:=true;
                For i:=1 to n-1 do
                        İf  s[i] < s[i+1] then begin
                                degis (s[i], s[i+1]);
                                tamam:=false;
                        end;
        until tamam;
End;
Begin
        Bubble2(s,n);
        Clrscr;
        For i:=1 to n do writeln(s[i]);
        Readln;
End.
```

3.4.2. Shell Sort

Bubble Sort' da elemanları sıralı olmaması, yani dizi içerisinde elemanların çok karışık olarak bulunması durumunda sıralama çok zaman alır. Bunun için aşağıda programı verilen Shell Sort sıralama programı kullanılabilir.

```
Program Shell_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: integer;
Procedure Shell (var s:stip; N:integer);
Var
        Ort,k: integer;
Procedure Degis(var a,b: integer);
Var
        c:integer;
Begin
        c:=a;
        a:=b;
        b:=c;
end;
Begin
        Ort:=n div 2;
        While (ort > 0) do begin
                For i:=ort+1 to n do begin
                        j:=i-ort;
                        while (j>0) do begin
                                k:=j+ort;
                                İf  s[j] >= s[k] then c:=0
                                Else begin
                                        degis (s[j], s[k]);
                                        j:=j-ort;
                                end;
                        end;
                end;
                ort:=ort div 2;
        end;
end;
Begin
        Shell(s,n);
        Clrscr;
        For i:=1 to n do writeln(s[i]);
        Readln;
End.
```

3.4.3. Selection Sort

En çok kullanılan sıralama yöntemidir. Bu sıralama yönteminde sıralama işlemi dizideki ilk elemandan başlayarak, her eleman kendisinden sonrtaki elemanla test edilir (1. eleman 2., 3., 4.....n. elemanlarla, 2. eleman 3., 4.....n. eleman ile test edilir.). Verilen şartın sağlanması durumunda elemanların dizideki yerleri değiştirilir.

```
Program Selection_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 Selection (var s:stip; N:integer);
Var
        Tamam: boolean;
Procedure Degis(var a,b: integer);
Var
        c:integer;
Begin
        c:=a;
        a:=b;
        b:=c;
end;
Begin
        for i:=1 to n-1 do
                for j:=i+1 to n do
                        İf  s[i] < s[j] then degis (s[i], s[j]);
end;
Begin
        Selection(s,n);
        Clrscr;
        For i:=1 to n do writeln(s[i]);
        Readln;
End.
```

3.4.4. Quick Sort

Sıralama yöntemleri arasında en hızlı olanıdır. Aşağıdaki programda 1000 adet rastgele sayı üretilip bu sayılar bir diziye aktarılıyor. Daha sonra bu dizide sıralanıyor. 1000 adet sayıyı ne kadar kısa zamanda sıraladığına dikkat ediniz.

```
Program Quick_Sort;
Uses crt;
Const
        N=1000;
Type
        Stip = Array[1..n] of  integer;
Var
        s:stip;
        i: integer;
Procedure Quick (var s:stip; alt,ust:integer);
Procedure Qsort(L, R: integer);
Var
        i, j, x, y:integer;
Begin
```

```
        i:=1;           j:=r;
        x:=s[(1+r) div 2];
        repeat
                while s[i]<x do i:=i+1;
                while x<s[j] do j:=j-1;
                if i<=j then begin
                        y:=s[i];
                        s[i]:=s[j];
                        s[j]:=y;
                        i:=i+1;
                        j:=j-1;
                end;
        until i>j;
        if i<j then Qsort(i,j);
        if i<r then Qsort(i,r);
end;
Begin
        Qsort(alt,ust);
End;
Begin
        Write('Rastgele ',N,'adet sayı üretiliyor');
        Randomize;
        For i:=1 to n do s[i]:=random(30000);
        Writeln;
        Write('Şimdi Sıralama İşlemi Yapılıyor....');
        Quick(S,1,N);
        Writeln;
        For i:=1 to n do write(s[i]:8);
        Readln;
End.
```

3.5. ŞİFRELEME ALGORİTMALARI

Şifreleme tekniği, sizin okuduğunuz bilgiyi bir başkasının okuyamayacağı bir yapıya dönüştürmek için kullanılır. Bu yöntemde bilgi, alıcı dışında başka bir kişi tarafından okunmaması ya da değiştirilmemesi için kodlanır. Bilgi, transfer sırasında bir başkasının eline geçse bile şifrelenmiş olduğundan okunması güçtür. Şifreleme ve şifreyi çözmek için bir matematiksel algoritma ve bir anahtar gereklidir.

Anahtar, şifrelemek veya deşifre etmek için kullanılan sayısal karekterler dizisidir. Simetrik anahtar algoritmasında şifrelemek ve deşifre etmek için aynı anahtar; açık anahtar algoritmasında şifrelemek için açık anahtar, deşifre etmek için ise gizli anahtar kullanılır. Dijital imzalar açık anahtar algoritmasını kullanır. Dijital imza, imzanın sahibinin gizli anahtarı kullanılarak oluşturulur. Alıcı da imza sahibinin açık anahtarını kullanarak imzasını kontrol eder.

Günümüzde kullanılmakta olan modern ve güçlü şifreleme algoritmaları gizli değildir. Bu algoritmalar güvenliklerini kullandıkları farklı uzunluk ve yapılardaki anahtarlarla sağlarlar. Bütün modern algoritmalar şifrelemeyi ve şifre çözmeyi kontrol için anahtarları kullanır. Bir anahtar ile şifrelenmiş bilgi kullanılan algoritmaya bağlı

olarak, ilgili anahtar ile çözülebilir. Anahtar yapısı bakımından, şifreleme algoritmaları 2 grupta incelenebilir .

3.5.1. Simetrik (Gizli) Anahtar Algoritmaları

Simetrik Kripto-algoritmalar şifreleme ve çözmede aynı anahtarı kullanma prensibine dayalı olarak çalıştıklarından "simetrik" olarak nitelendirilirler. Bu tip algoritmalar gizli anahtar algoritması diye de adlandırılırlar. Başka bir deyişle şifreli iletinin çözülebilmesi için alıcının, kullanılan anahtarı daha önceden bilmesi gerekir.

3.5.2. Asimetrik Kripto-Algoritmaları

Whitfield Diffie ve Martin E. Hellman 1976' da iki farklı anahtara dayalı algoritma geliştirilebileceğini gösterdiler. Anahtarlardan biri açık (public), diğeri ise gizli (secret) olmalıdır. Bu yöntemde şifreleme için kullanılan anahtarlar ile şifre çözümünde kullanılan anahtarlar farklıdır. Bu nedenle bu algoritma tipinde şifreleme için kullanılan anahtar bilinse dahi (zaten bu anahtar açıktır ve herkes tarafından kolaylıkla öğrenilebilir), bu anahtarı kullanarak şifreyi çözmek için kullanılan anahtarı elde etmek olanaksızdır. Böylece ilk defa, gizli simetrik anahtar dağıtım problemi çözülmüş oldu ve herkes tarafından gerçeklenebilecek sayısal imza gibi yeni yöntemler mümkün oldu.

BÖLÜM 4

4. ALGORİTMA VE AKIŞ ŞEMALARI

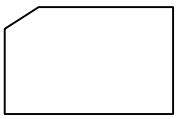
Bilgisayarlarda bir problemin çözümünde izlenecek yol genel tanımıyla **algoritma** olarak adlandırılır. Algoritmalar, bir problemin çözümündeki işlemlerin, kararların ve bunların icra edildiği sıranın oluşturduğu akış olarak düşünülebilir. Algoritma kurma, programlama aşamasının en önemli kısmıdır. Burada üretilen mantıksal akışlar, bir anlamda programlama olarak adlandırılan ve bilgisayarın anlayabileceği dilde kodlama sisteminin temelini oluşturmaktadır.

Programlamaya başlamadan önce problemin çözümüne ilişkin mantıksal akışları içeren algoritmanın oluşturulması, programlama aşamasında son derece kolaylık sağlayacaktır. Bir anlamda geriye sadece ilgili programlama dilinde işlemleri kısaltıcı fonksiyonların kullanılması ve kodlama işlemlerin yapılması kalacaktır.

Akış şemaları ise, algoritmaların içerdiği işlemlerin birer geometrik şekil ile ifade edilmesi olarak tanımlanabilir. Akış şemaları, programlama aşamasında programı kodlamak, kontrol etmek, açıklamak, gözden geçirmek ve gerekirse güncellemek açısından büyük kolaylıklar sağlamaktadır.

Akış şemalarında kullanılacak şekiller ve bunların anlamı aşağıda verilmiştir.

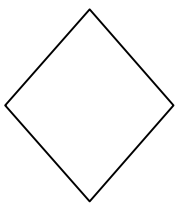
Başlama, Bitiş ve bağlantı İşlemleri



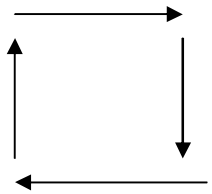
Giriş ve Okutma İşlemleri

Atama ve Hesaplama İşlemleri

Yazdırma İşlemleri



Karar ve Kontrol İşlemleri



Akış yönünü belirten işlemler

4.1. DEĞİŞKEN KAVRAMI

Bir problemin çözümünde tanımlanan bir bilgi alanı, farklı adımlarda farklı değerler alabiliyorsa bu bilgi alanına "**değişken"** adı verilir. Tanımlanacak bir değişken için aşağıdaki kurallar dikkat etmek gerekir.

1. Bir değişken adı A ile Z arasındaki alfabetik harfler ile başlamalıdır. Değişken adı bir kelime ya da arada bir boşluk olmama koşuluyla bir cümle olabilir.

A, TOPLAM, SAYI, SONUC, TOPLAM, SAYI, SONUC, ADSOYAD gibi tanımlanabilir.

2. Bir değişken adının ilk karakteri sayısal olamaz, yani 0 ile 9 arasında bir rakam ile başlayamaz. Ancak, ilk karakterden sonra istenilen bir sayı kullanılabilir.

A1, TOPLAM1, KIA37, B1589,..... şeklinde kullanılabilir.

3. Değişken adı algoritmanın kodlanacağı programlama diline ait bir komut ya da deyim olamaz.

PRINT, END, NO, READ,..... şeklinde kullanılamaz.

4. Algoritmada değişken adı verilirken Türkçe karakterler kullanmamaya dikkat edilmelidir.

SONUÇ, DEĞER, KOŞUL,.... gibi.

4.2. AKTARMA VE ATAMA İŞLEMLERİ

Bir aritmetik ifadenin ya da bir değerin herhangi bir değişkene tanımlanmasına **"aktarma"** ya da **"atama"** işlemleri adı verilir. Bu atama ve aktarma işlemleri sayısal ifadeler ve aritmetik ifadeler için;

<değişken adı>=<aritmetik ifade ya da değer> şeklindedir.

Eğer atanacak ifade sayısal ifade değilse bu durumda ifade " " ile birlikte atanmak zorundadır.

<değişken adı>="sayısal olmayan ifade" şeklindedir.
Örnek olarak, TOPLAM=A+B, A=3, ISIM="AHMET" verilebilir.

4.3. ARTIRIM İŞLEMLERİ

Bilgisayar mantığında matematik mantıktan farklı olarak bazı işlemler yapılabilmektedir. Herhangi bir değişkene kendisiyle birlikte bir değeri atamak mümkündür.

TOPLAM=TOPLAM+A tanımlaması yapılabilir.

Burada eşitliğin solunda tanımlanan yeni TOPLAM ifadesi, önceki TOPLAM ifadesine A ilave edilerek elde edilmiştir. Benzer şekilde;

X=X+1, SAYAC=SAYAC+1, SAYI=SAYI+A-1

Gibi tanımlamalar yapılabilir.

4.4. ARİTMETİK İŞLEMLER

Algoritma ve Akış Şemalarında kullanılacak olan işlem operatörleri aşağıdaki gibi tanımlamaktadır.

(+) 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

Matematiksel işlemlerdeki operatörlerin kullanım öncelikleri, algoritma ve akış şemalarında da aynen geçerlidir. Bu kullanım öncelikleri aşağıdaki gibi tanımlanmaktadır.

1. Öncelikli, varsa parantez içerisi ( )
2. Öncelikli, Üs alma işlemi ^
3. Öncelikli, Çarpma ve Bölme işlemi *, /
4. Öncelikli, Toplama ve Çıkarma işlemi +,-

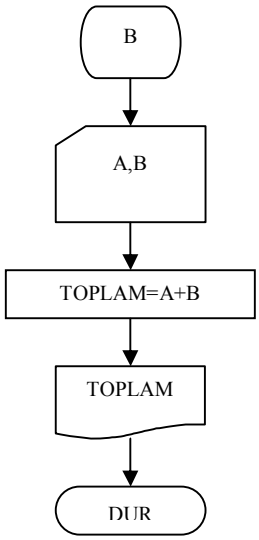
4.5. SAYILAR İLE İLGİLİ ALGORİTMALAR

Bu bölümde, sayıların değişik şekillerde kullanılmasına ilişkin algoritma ve akış şeması örnekleri yapılacaktır.

İşlemlerde bir sayının tam kısmını alırken TAM ifadesi, mutlak değerini alırken de MUTLAK ifadesi kullanılacaktır. Örneğin A sayısının tam kısmı olarak TAM(A) ve mutlak değeri için de MUTLAK(A) ifadesi kullanılacaktır.

Örnek 4.5.1. Girilen iki sayının toplamını bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. A ve B sayılarını gir/oku,
A3. TOPLAM=A+B al,
A4. TOPLAM' ı yaz,
A5. Dur.



Şekil 4.5.1. Girilen iki sayının toplamını bulan akış şeması

Yukarıdaki örneği adım adım incelemek gerekirse, A1. adım, işlemlere başlama anlamındadır. A2. adım, A ve B gibi herhangi iki sayının girilmesi ya da okunmasını belirtmektedir. A3. adım, girilen A ve B gibi iki sayının toplanarak TOPLAM değişkenine atanmasını sağlamaktadır. A4. adım, elde edilen TOPLAM değerinin yazdırılmasını sağlamaktadır. A5. adım, işlemlerin bittiğini belirtmektedir.

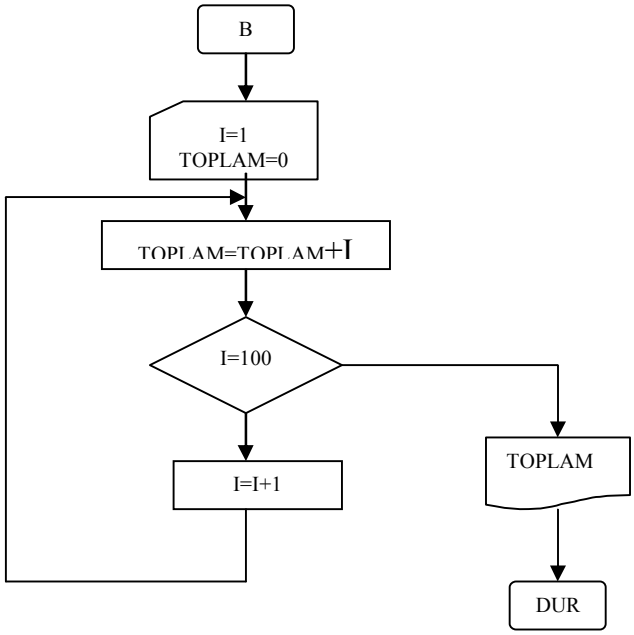
Bu algoritmayı bir sayısal örnekle değerlendirelim.
A=10, B=20 girilsin. Üçüncü adımda,
TOPLAM=10+20=30
olacaktır ve dördüncü adımda 30 değeri ekrana yazdırılacaktır.

Algoritmanın Pascal Dilindeki Karşılığı :

```
Program toplama;
Var
        a,b,c : integer;
Begin
        write('a sayısını giriniz : ');   readln(a);
        write('b sayısını gireiniz : '); readln(b);
        c:= a+b;
        writeln('Toplam : ',c);
End.
```

Örnek 4.5.2. 1’ den 100’ e kadar olan tamsayıların toplamını bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. I=1, TOPLAM=0 al,
A3. TOPLAM=TOPLAM+I al
A4. Eğer I=100 ise A6. adıma git,
A5. I=I+1 al ve A3. adıma geri dön,
A6. TOPLAM değerini yaz,
A7. Dur.



Şekil 4.5.2. 1 ile 100 arasındaki tamsayıların toplamını bulan akış şeması

Yukarıda tanımlanan algoritmada 1’ den 100’ e kadar artışı sağlayan bir I sayacı ve bu sayaçtaki değerleri toplayan bir TOPLAM değişkeni tanımlanmıştır. Başla komutundan sonra, I değerine 1 ve TOPLAM değişkene 0 değeri atanmıştır. Daha sonra yeni bir TOPLAM tanımlanarak ilk toplam değeri ve I değeri bu değişkene aktarılır. Bu aşamada, I değerinin TOPLAM değişkenindeki ilk değeri elde edilmiştir. Bu işlemler I değeri 100 olana kadar devam edeceğinden, bir kontrol işleminin tanımlanması gerekmektedir. A4. adımda I değeri kontrol edilmektedir. I değeri 100 oluncaya kadar A3. adımdaki toplama işlemi yapılmaktadır. I değeri 100 olunca elde edilen son TOPLAM değeri A7. adımda yazdırılarak işleme son verilmektedir.

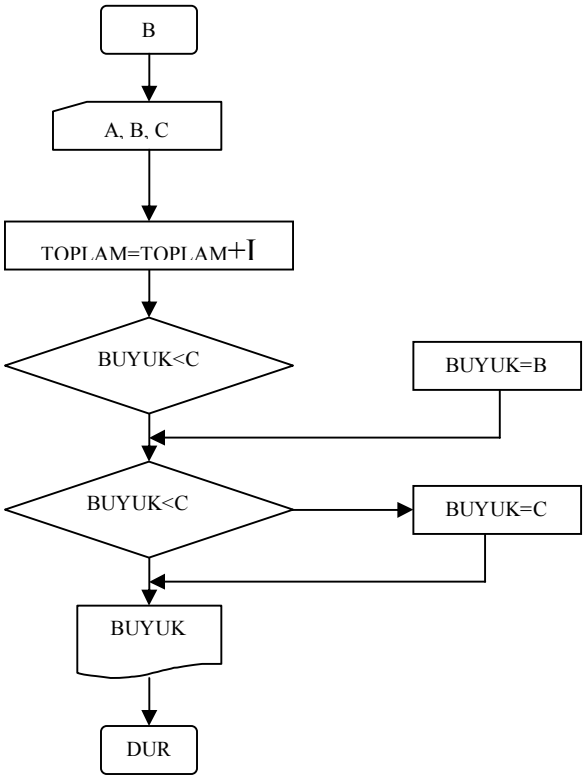
Algoritmanın Pascal Dilindeki Karşılığı :

```
Program 1_100_arasi_toplam;
Var
        i,TOPLAM: integer;
Begin
        TOPLAM=0;
```

```
For i:=1 to 100 do
        TOPLAM=TOPLAM+i;
        Writeln(‘ Toplam : ‘, TOPLAM);
End.
```

Örnek 4.5.3. Girilen üç tamsayıdan en büyüğünü bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. A,B ve C sayılarını gir,
A3. BUYUK=A al,
A4. Eğer BUYUK < B ise BUYUK=B al,
A5. Eğer BUYUK < C ise BUYUK=C al,
A6. BUYUK değerini yaz,
A7. Dur.



Şekil 4.5.3. Girilen üç sayıdan en büyüğünün bulunmasını sağlayan akış şeması

Problemin çözümünde özellikle A3. adıma dikkat edildiğinde BUYUK şeklinde bir değişken tanımlanarak, bu değişkene A değeri atanmıştır. Buradaki amaç, her bir sayıyı karşılıklı olarak birbirleri ile karşılaştırmak yerine girilen sayılardan bir tanesini bir değişkene atayarak diğer değerleri bununla karşılaştırmanın daha avantajlı olduğunu göstermektir. Eğer her bir sayı birbirleri ile karşılaştırılmış olsaydı, üç karşılaştırma gerekecekti. Sayı adedi dört olsaydı altı karşılaştırma, sayı adedi beş olsaydı on karşılaştırma gerekecekti. Oysa yukarıda tanımlanan algoritmada A3. adımdaki değişken yardımıyla üç sayı için iki karşılaştırma yapılmıştır. Benzer şekilde n sayı için

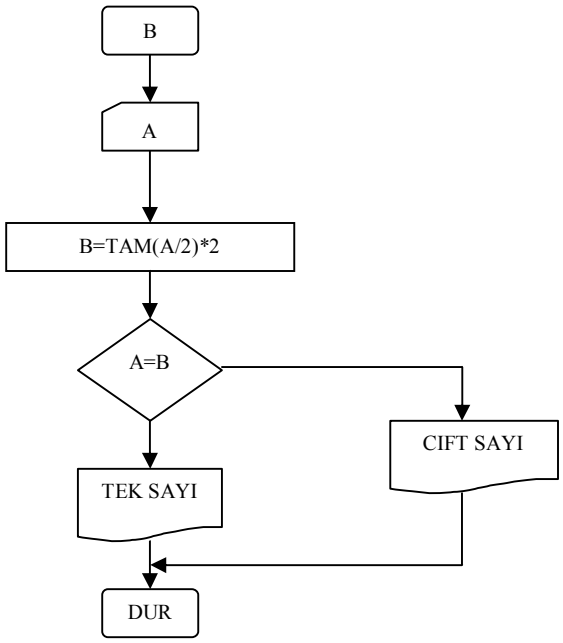
de (n-1) karşılaştırma ile en büyük sayı bulunacaktır. Algoritmaya devam edildiğinde, A4. adımda BUYUK olarak tanımlanan değer B ile karşılaştırılmaktadır. Eğer BUYUK, B' den küçük ise BUYUK yerine B değeri atanmaktadır. Benzer şekilde A5. adımda da bu ifade C ile karşılaştırılmaktadır. En son elde edilen BUYUK değeri yazdırılarak işleme son verilmektedir.

Algoritmanın Pascal Dilindeki Karşılığı :

```
Program en_buyuk;
Var
        a,b,c, BUYUK: integer;
Begin
        Write(' a, b ve c sayılarını giriniz : ');
        Readln(a,b,c);
        BUYUK=a;
        If  BUYUK<B then BUYUK:=B;
        If BUYUK<C then BUYUK:=C,
        Writeln('En büyük sayı : ',BUYUK);
End.
```

Örnek 4.5.4. Girilen bir tamsayının tek ya da çift olduğunu tespit eden algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. A sayısını gir,
A3. B=TAM+(A/2) al,
A4. Eğer A=B ise A6. adıma git,
A5. "Girilen sayı tek sayıdır" yaz ve A7. adıma git,
A6. "Girilen sayı çift sayıdır" yaz,
A7. Dur.



Şekil 4.5.4: Girilen bir tam sayının tek ya da çift olduğunu bulan akış şeması

Girilen bir sayının çift olabilmesi için 2 ile tam bölünebilmesi gerekir. Yukarıdaki algoritma bu temel doğrultusunda oluşturulmuştur. A2. adımda girilen bir A sayısı, A3. adımda 2' ye bölünüp tam kısmı alınarak tekrar 2 ile çarpılmakta ve elde edilen bu değer B olarak tanımlanan bir değişkene atanmaktadır. Eğer A sayısı B sayısı ile aynı değere sahipse, bu sayı çift olacaktır. Aksi halde bu sayı tek olacaktır. Bu durum A4. adımda verilen sart ile kontrol edilmektedir.

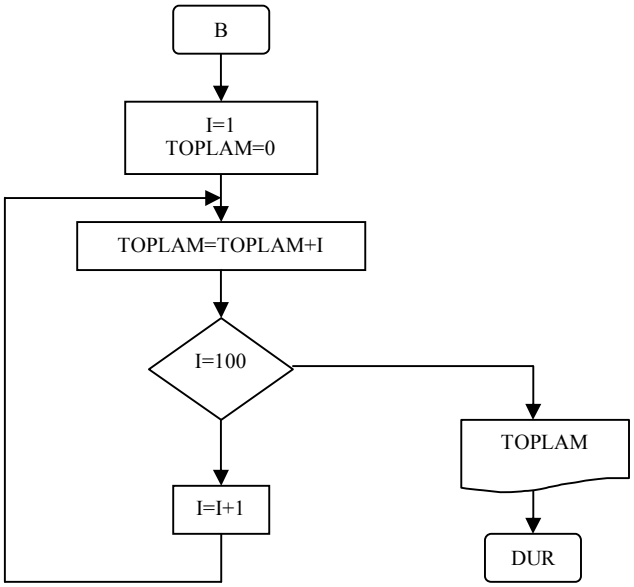
Örnek olarak,
A=75 alınırsa,
B=74 olacaktır.
A<>b olacağından 5. adım devreye girerek bu sayı tek sayıdır yazacaktır.

Algoritmanın Pascal Dilindeki Karşılığı:

```
Program tek_cift;
Var
        a,b: integer;
Begin
        Write('a sayısını gir :');
        Readln(a);
        b:=trunc(a/2);
        if (a=b) then
                Writeln('girilen sayı çift sayıdır ')
        Else
                Writeln('girilen sayı tek sayıdır ');
End.
```

Örnek 4.5.5. 1 ile 100arasındaki tam sayılardan tek ve çift olanların ayrı ayrı toplamını bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. SAYAC=1, TTEK=0,TCIFT=0 al,
A3. TTEK=TTEK+SAYAC al,
A4. TTEK=TCIFT+(SAYAC+1) al,
A5. Eğer SAYAC=99 ise A8. adıma git,
A6. SAYAC=SAYAC+2 al,
A7. A3. adıma geri dön,
A8. TTEK ve TCIFT değerlerini yaz,
A9. Dur.



Şekil 4.5.5. 1 ile 100 arasındaki tamsayılarının tek ve çift olanlarının toplamını bulan akış şeması

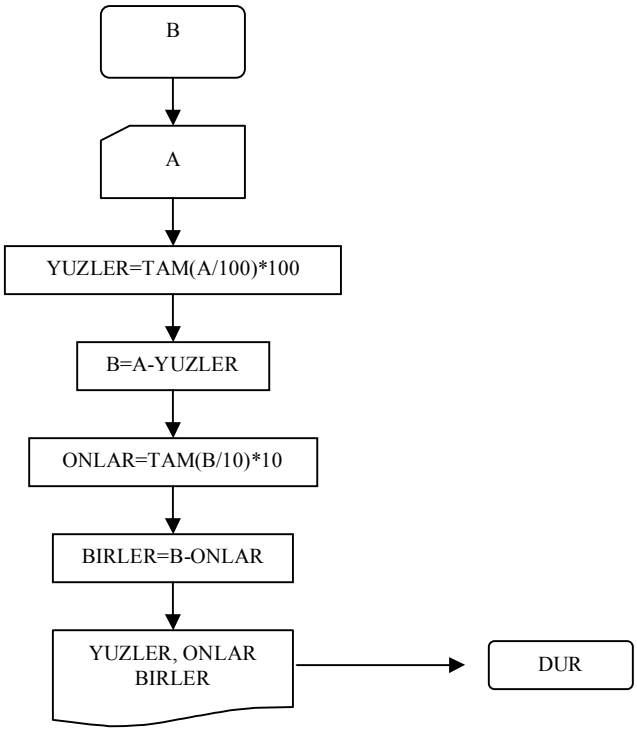
Yukarıdaki tanımlanan algoritmada 1' den 100' e kadar artışı sağlayan bir SAYAC değişkeni tanımlanmıştır ve ilk değer olarak 1 değeri atanmıştır. Tek ve çift sayıları toplamak üzere TTEK ve TCIFT değişkenleri tanımlanarak bunlara ilk değer olarak 0 değeri atanmıştır. Bunun nedeni, ilk aşamada herhangi bir toplamın söz konusu olmamasıdır. SAYAC değişkeninin ilk değeri tek sayı olduğundan bu değer A3. adımda TTEK değişkenine önceki değeri ile birlikte atanmıştır. A4. adımda SAYAC değişkenine 1 ilave edilerek ilk çift sayı tespit edilmiş ve bu değer TCIFT değişkenine önceki değeri ile birlikte atanmıştır. Bu işlem SAYAC değişkeninin alabileceği son değişken olan 99 ifadesine kadar devam edeceği için A5. adımda bu durum sorgulanarak elde edilen duruma göre işleme devam edilmektedir. A6. adıma dikkat edildiğinde SAYAC değişkenine önceki değeri ile birlikte 2 ilave edilmiştir. Bunun nedeni sonraki tek tamsayıyı elde etmektir. Çünkü SAYAC değerine 1 ilave edilerek çift sayılar TCIFT değişkeninde toplanmaktadır. Sonuçta en son tek sayı 99 olacağından kontrol işlemi 99'a kadar yapılmaktadır. Son olarak elde edilen TTEK ve TCIFT değerleri yazdırılarak işlemlere son verilmektedir.

Algoritmanın Pascal Dilindeki Karşılığı:

```
Program tek_cift_toplami;
Var
        Sayac,ttek,tcift: integer;
Begin
        Ttek:=0; tcift:=0;
        For sayac:=1 to 50 do
        Begin
                Ttek:=ttek+2*sayac-1;
                Tcift:=tcift+2*sayac;
        End;
        Writeln('tek sayı toplamı  =',ttek);
        Writeln('çift sayı toplamı  ='.tcift);
End.
```

Örnek 4.5.6. Üç haneli bir tamsayının birler, onlar ve yüzler hanesini bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. A sayısını gir { 3 haneli bir sayı },
A3. YUZLER=TAM(A/100)*100 al
A4. B=A-YUZLER al,
A5. ONLAR=TAM(B/10)*10 al,
A6. BIRLER=B-ONLAR al,
A7. YUZLER, ONLAR ve BIRLER değerlerini yaz,
A8. Dur.



Şekil 4.5.6. Üç haneli bir tamsayıyı birler, onlar ve yüzler hanesine ayıran akış şeması

Girilen bir A sayısının yüzler hanesindeki değerini bulmak için A3. adımda A sayısını 100‘ e bölünüp, tam kısmı alınarak tekrar 100 ile çarpılmıştır. Bu durumda girilen sayının yüzler hanesi YUZLER isimli bir değişkene aktarılmıştır. Geriye kalan değeri bulmak için, A4. adımda, A sayısından YUZLER olarak elde edilen değer çıkarılarak elde edilen değer B değişkenine aktarılmıştır. B değişkeni artık 2 haneli bir değişkendir. Benzer şekilde A5. adımda B değeri 10’ a bölünüp tam kısmı alınarak 10 ile çarpılmış ve bu değer ONLAR isimli bir değişkene aktarılmıştır. A6. adımda B değerinden ONLAR isimli değer çıkarılarak birler hanesi bulunmuştur ve bu değer de BIRLER isimli değişkene aktarılmıştır. Görüldüğü üzere YUZLER, ONLAR ve BIRLER isimli değişkenler, girilen üç basamaklı bir sayının hanelerini ifade etmektedir.

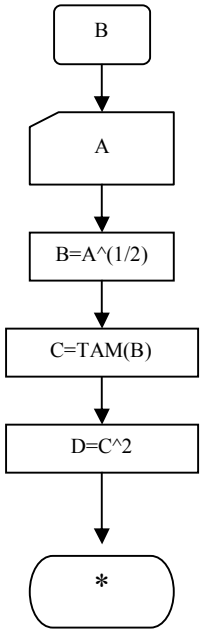
Örnek olarak A=543 alınırsa,
YUZLER=500 olacaktır. Sonraki adımda,
B=A-YUZLER=43 olacaktır. A5. adımda,
ONLAR=40 olarak elde edilecektir. A6. adımda,
BIRLER=B-ONLAR ifadesinden
BIRLER=43-40=3 elde edilecektir.

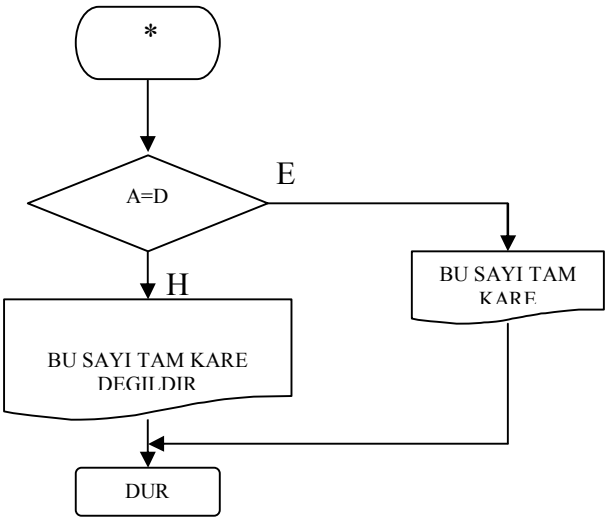
Algoritmanın Pascal Dilindeki Karşılığı :

```
Program hane_sayilari;
Var
        a, b, yuzler,onlar,birler :integer;
Begin
        Write('a sayısını giriniz : ');     readln(a);
        Yuzler:=trunc(a/100)*100;
        b:=a-yuzler;
        onlar:=trunc(b/10)*10;
        birler:=b-onlar;
        writeln(yuzler,' ',onlar,' ',birler);
End.
```

Örnek 4.5.7. Girilen bir tamsayının tam kare olup olmadığını araştıran algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. A sayısını gir,
A3. B=A^(1/2) al,
A4. C=TAM(B) al,
A5. D=C^2 al,
A6. Eğer A=D ise A9. adıma git,
A7. "tam kare değil" yaz,
A8. A10. adıma git,
A9. "tam kare"yaz,
A10. Dur.





Şekil 4.5.7. Girilen bir tamsayının tam kare olup olmadığını bulan akış şeması

Girilen bir A sayısının tam kare olabilmesi için, bu sayının karekökünün tam kısmı aşınarak elde edilen değerin tekrar karesi alındığında bu sayının ilk girilen sayıya eşit olması gerekir. Aksi halde girilen sayı tam kare olmayacaktır. Yukarıdaki algoritmada aynı mantıkla hareket edilerek A2. adımda bir A sayısı girilmiştir. A3. adımda bu sayının karekökü alınarak B gibi bir değişkene atanıp, A4. adımda bu değerin tam kısmı alınarak C gibi bir değişkene atanmıştır. A5. adımda c değerinin karesi alınarak elde edilen değer D olarak adlandırılan bir değişkene atanmıştır ve A6. adımda D değeri girilen A değeri ile karşılaştırılarak bu ilk değerin eşitliği durumunda A9. adıma gidilerek girilen sayının tam kare olduğu yazdırılarak işleme son verilmektedir. Aksi halde A7. adımda girilen sayının tam kare olmadığı yazdırılarak işleme son verilmektedir.

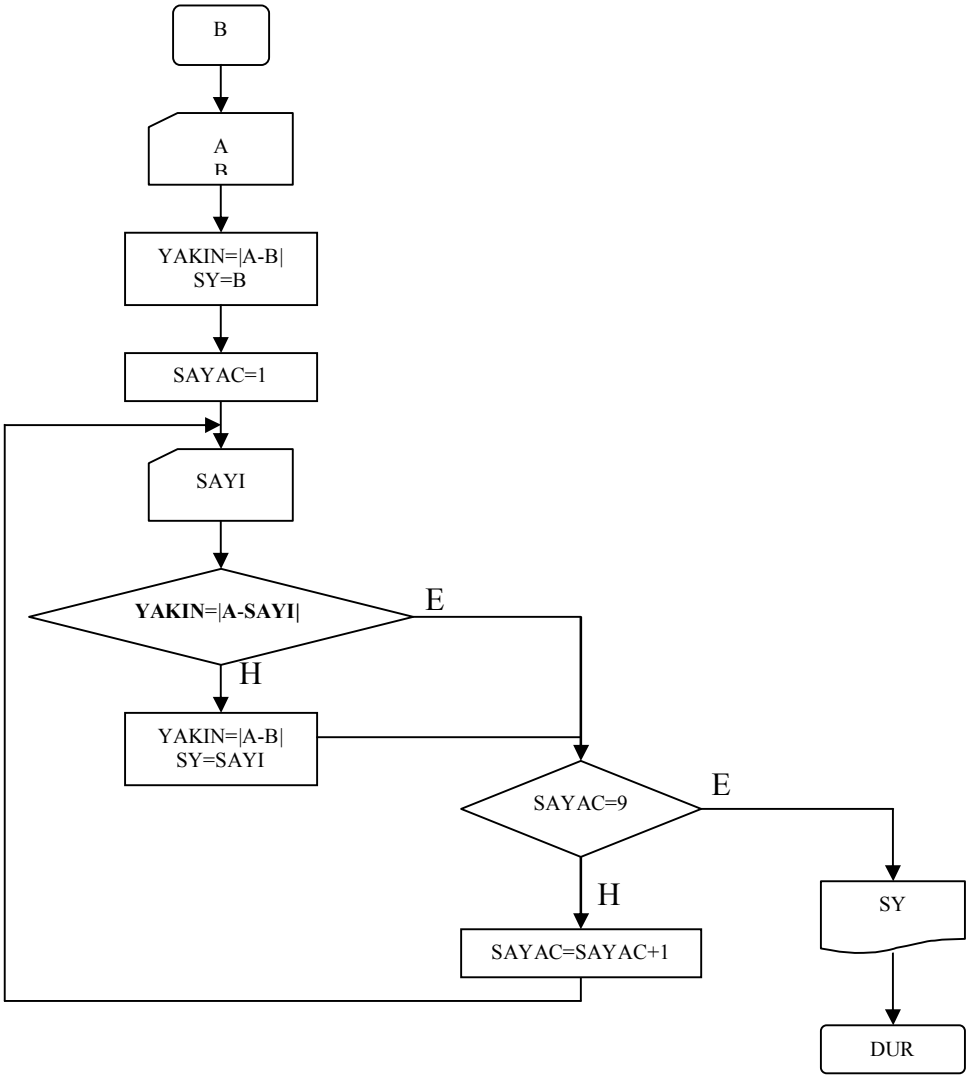
Örnek olarak A=5 alınırsa b=2,236.. şeklinde olacaktır. Buna göre C=2 ve D= 4 olacaktır. Dolayısıyla A<>D olacağından A7. adım devreye girerek bu sayının tam kare olmadığı yazılacaktır. Eğer A=9 olsaydı, B=3 ve D=9 olacağından A=D elde edilecekti ve durumda A9. adım devreye girerek sayının tam kare olduğu yazdırılacaktı.

Algoritmanın Pascal Dilindeki Karşılığı:

```
Program  tamkare;
Var
        a,b,c,d:integer;
begin
        write('a sayısını gir :');   readln(a);
        b:=sqrt(a);
        c:=trunc(b);
        d:=sqr(c);
        if(a=b) then
                writeln('bu sayı tamkaredir')
        else
                writeln('bu sayı tamkare değildir');
end.
```

Örnek 4.5.8. Arka arkaya girilen 10 sayıdan istenilen sayıya en yakın olan sayının bulunmasını sağlayan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. A sayısını gir { istenilen sayının girilmesi },
A3. B sayısını gir {ilk sayı}
A4. YAKIN=MUTLAK(A-B),SY=B al,
A5. SAYAC=1 al,
A6. SAYI' yı gir al,
A7. Eğer YAKIN<MUTLAK(A-SAYI) ise A9. adıma git,
A8. YAKIN=MUTLAK(A-SAYI), sy=SAYI al,
A9. Eğer SAYAC=9 ise A12. adıma git,
A10. SAYAC=SAYAAC+1 al,
A!!. A6. adıma geri dön,
A12. SY' yi yaz,
A13. Dur.



Şekil 4.5.8. En yakın sayının bulunması

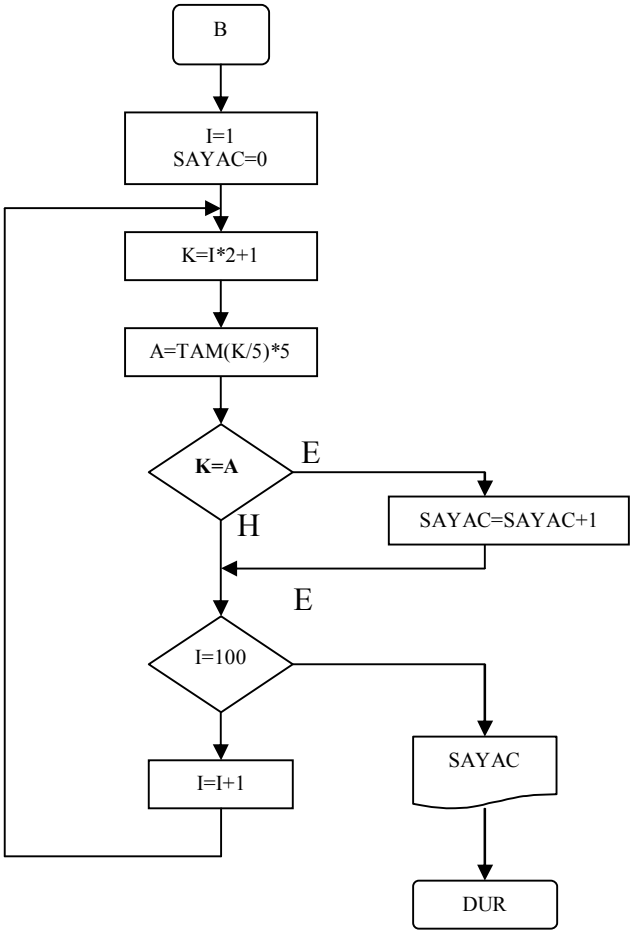
Algoritmaya bakıldığında A2. adımda rasgele bir A sayısının girilmesi istenmektedir. Bundan sonraki adımlarda girilen bu A sayısına en yakın sayının tespiti için gerekli mantıksal işlemler yapılmaktadır. A3. adımda B olarak tanımlanan ilk sayının girişi istenmektedir. Buradaki amaç, A sayısı ile girilen bu B sayısının farklı alınarak elde edilen bu değerin ilk yakın değer olarak kabul edilmesidir. A4. adımda bu fark YAKIN isimli bir değişkene atanmıştır. A5. adı A3. adımda B olarak tanımlanan ilk sayının girişi istenmektedir. Buradaki amaç, A sayısı ile girilen bu B sayısının farklı alınarak elde edilen bu değerin ilk yakın değer olarak kabul edilmesidir. A4. adımda bu fark YAKIN isimli bir değişkene atanmıştır. A5. adımda geriye kalan 9 sayının girişini yapabilmek için 1' den 9' a kadar değer alacak olan SAYAC tanımlanmıştır ve ilk olarak bu SAYAC' a 1 verilmiştir. A6. adımdan itibaren geriye kalan 9 sayının girişi başlamaktadır ve A4. adımda tanımlanan YAKIN değişkeni ile girilen SAYI değerinin A' dan farkı karşılaştırılarak işleme devam edilmektedir. A7. adımda YAKIN değişkeni A-SAYI değerinden büyük ise yeni bir YAKIN değişkeni tanımlanarak A-SAYI farkı bu değere atanmaktadır ve A8. adımda SAYAC için karşılaştırma yapılarak 9 değerine ulaştığında A11. adıma gidilerek en yakın sayının tespiti için A-YAKIN farkı alınmaktadır ve bu değer ENYAKIN isimli bir değişkene atanmaktadır. Daha sonra A12. adımda ENYAKIN değeri yazdırılarak işlemlere son verilmektedir. Eğer SAYAC 9 değerine ulaşmamış ise A6. adımdan itibaren işlemlere yeniden devam edilmektedir.

Algoritmanın Pascal Dilindeki Karşılığı:

```
Var
        a,b,yakin, sayac, sayi,sy:integer;
begin
        write('a sayısını gir :');  readln(a);
        write('b sayısını gir :'); readln(b);
        yakin:=trunc(a-b);
        for sayac:=1 to 9 do   begin
                write('sayıyı gir :'); readln(sayi);
        if(yakin>trunc(a-sayi)) then
        begin
                yakin:=trunc(a-sayi); sy:=sayi;
        end;
        end;
        printf('en yakin :',enyakin);
end.
```

Örnek 4.5.9. 1 ile 100 arasındaki tamsayının 2 katının bir fazlası 5 ile tam bölünen kaç sayı olduğunu bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. I=1, SAYAC=0 al,,
A3. K=I*2+1 al,
A4. A=TAM(K/5)*5 al,
A5. Eğer K=A ise SAYAC=SAYAC+1 al,
A6. Eğer I=100 ise A8. adıma git,
A7. I=I+1 al ve A3. adıma geri dön,
A8. SAYAC değerini yaz,
A9. Dur.



Şekil 4.5.9: 1 ile 100 arasındaki tam sayılardan 2 katının 1 fazlası 5 ile tam bölünen kaç sayı olduğunu bulan akış şeması

Oluşturulan algoritmada gerekli olan değişkenler A2. adımda tanımlanmaktadır. Bunlar sırasıyla, 1' den 100' e kadar değer alabilen bir I değişkeni ile istenilen şarta uyan sayı adedini bulan bir SAYAC değişkenidir. İlk aşamada bir SAYAC değişkeninin değeri 0 ve I' nın değeri de 1 olarak tanımlanmıştır. A3. adımda bir K değişkeni tanımlanarak buna I değişkeninin iki katının bir fazlası atanmıştır. A4. adımda, bir önceki adımda elde edilen K değişkeninin beş ile bölümünün tam kısmı alınarak tekrar beş ile çarpılarak A isimli bir değişkene atanmıştır. Buradaki amaç, eğer K değişkeni 5 ile tam olarak bölünebiliyorsa tam kısmı tekrar beş ile çarpıldığında yine K sayısı elde edilecektir ve dolayısıyla I değişkeni istenilen şarta uygun sayı olarak dikkate alınacaktır. Bu durum A5. adımda karşılaştırılmaktadır. Eğer şart gerçekleşiyorsa SAYAC değişkeni 1 artırılarak A6. adımdan itibaren işleme devam edilmektedir. A6. adımdan itibaren I değişkeninin değeri 100 olana kadar A3. adım ile A7. adım arasındaki işlemlerin tekrar yapılması sağlanmaktadır. Sonuçta I değişkeni 100 değerine ulaşınca A9. adıma geçilerek elde edilen SAYAC değeri yazılarak işleme son verilmektedir.

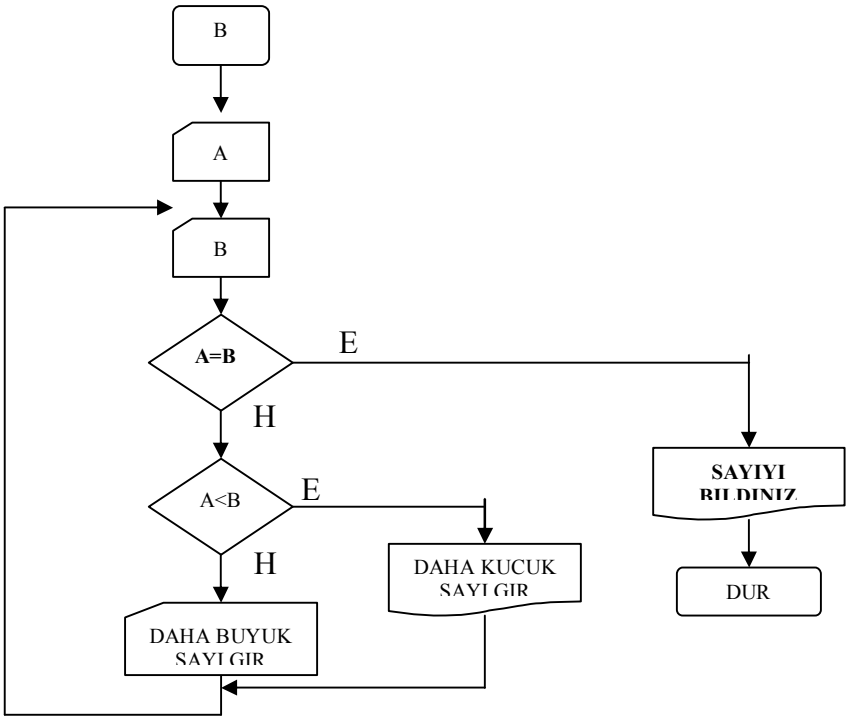
Algoritmanın Pascal Dilindeki Karşılığı:

```
Program bolme;
Var
        i,a,k,sayac:integer;
```

```
begin
        sayac:=0;
        for i:=1 to 100 do
        begin
                k:=2*i+1; a:=trunc(k/5)*5;
                if(k=a) then sayac:=sayac+1;
        end;
        writeln('bölünen sayı =',sayac);
end.
```

Örnek 4.5.10. (sayı sıkıştırma örneği) 1 ile 100 arasındaki tamsayılardan bir sayı okutularak bu sayıya ulaşmak için bu aralıkta sayılar girilerek okutulan sayıya ulaşmayı sağlayan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. A sayısını oku,
A3. Bir B sayısı gir,
A4. Eğer A=B ise A10. adıma git,
A5. Eğer A<B ise A8. adıma git,
A6. "Daha büyük bir sayı gir" yaz,
A7. A3. adıma geri dön,
A8. "Daha küçük bir sayı gir" yaz,
A9. A3. adıma geri dön,
A10. "Sayıyı buldunuz" yaz,
A11. Dur.



Şekil 4.5.10. Sayı sıkıştırmaya ilişkin akış şeması

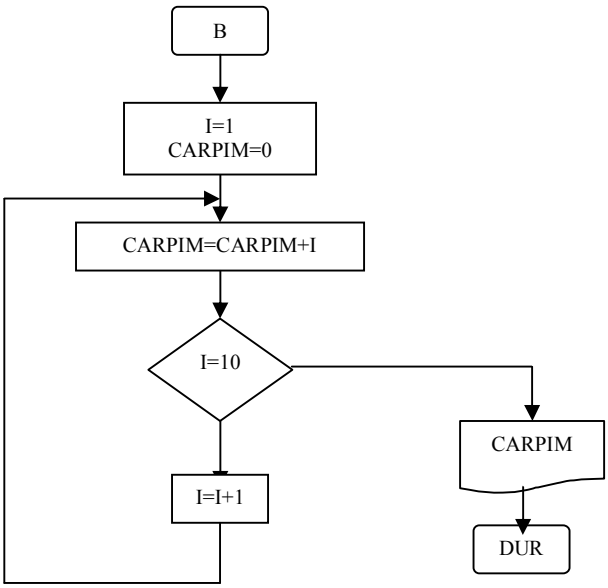
Verilen örnekte A2. adımda 1 ile 100 arasında rasgele bir A sayısının okutulması yapılmaktadır. A3. adımda ise bu sayıyı bulmayı sağlayan ve yine bu aralıkta bir B sayısının girilmesi sağlanmaktadır. A4. adımda girilen sayı A sayısına eşitse A10. adıma gidilerek sayıya ulaşıldığına dair mesaj yazdırılarak işleme son verilmektedir. Aksi halde, A5. adımda A sayısı ile B sayısı karşılaştırılmaktadır ve A<B ise A8. adıma gidilerek daha küçük bir sayı girilmesi istenip A3. adıma tekrar geri dönülmektedir. Eğer bu şart sağlanmıyorsa A>B olacağından A6. adımda daha büyük bir sayının girilmesi istenerek A3. adıma geri dönülmektedir.

Algoritmanın Pascal Dilindeki Karşılığı :

```
Program sayisikiştirma;
Var
        a,b : integer;
Begin
        Write('lütfen 1 ile 100 arasında değer giriniz....');
        Write('a sayısını gir : ');
        readln(a);
        Write('b sayısını gir : ');
        readln(b);
        while (a<>) do
        begin
                if (a<b) then
                writeln('daha küçük bir sayı gir');
                else
                writeln('daha büyük bir sayı giriniz');
         end;
         writeln('Sayıyı buldunuz');
end.
```

Örnek 4.5.11. 1 ile 10 arasındaki tamsayıların çarpımı bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. I=1, CARPIM=1 al,
A3. CARPIM=CARPIM*I al,
A4. Eğer I=10 ise A7. adıma git,
A5. I:=I+1 al,
A6. A3. adıma geri dön,
A7. CARPIM' ı yaz,
A8. Dur.



Şekil 4.5.11. 1 ile 10 arasındaki tamsayıların çarpımını bulan akış şeması

Verilen örnekte A2. adımda 1' den 10' a kadar artışı sağlayan bir I değişkeni ve bu değerleri çarpan bir CARPIM değişkeni tanımlanmıştır. A3. adımda CARPIM değişkenine önceki değeri ile birlikte I değişkeninin değeri çarpılarak atanmıştır. Bu durum A4. adımda verilen şartın gerçekleşmesine kadar devam etmektedir. Burada I değişkeninin değeri 10 olunca A7. adıma gidilerek CARPIM değeri yazdırılıp işlemlere son verilmektedir.

Bu örneği gerçek değerlerle yapmak istediğimizde, algoritmadaki adımlar aşağıdaki gibi çalışacaktır.

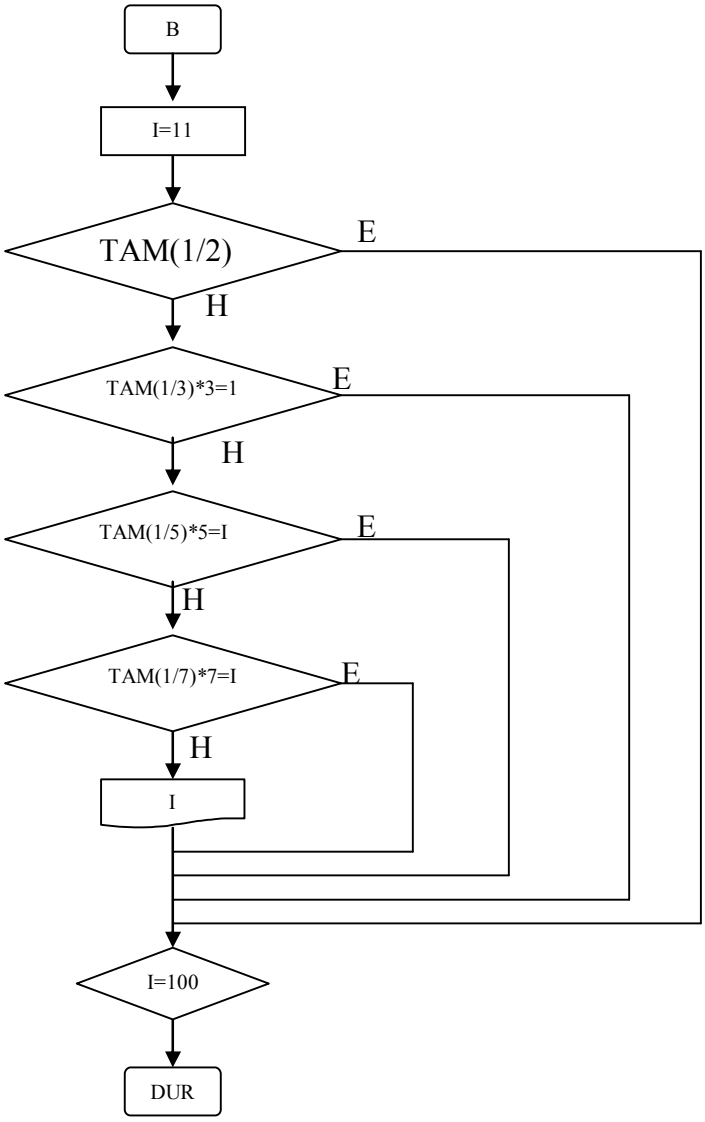
I=1 ve CARPIM=1 alınırsa yeni CARPIM=1*1=1 olacaktır. Sonra,
I=I+1=2 olup CARPIM=1*2=2 olacaktır. Benzer şekilde
I=3 için CARPIM=2*3=6,
I=4 için CARPIM=6*4=24,
I=5 için CARPIM=24*5=120,
I=6 için CARPIM=120*6=720,
I=7 için CARPIM=720*7=5 040,
I=8 için CARPIM=5040*8=40 320,
I=9 için CARPIM=40320*9=362 880,
I=10 için CARPIM=362 880*10=3 628 800 olarak bulunacaktır.

Algoritmanın Pascal Dilindeki Karşılığı :

```
Program faktoriyel,
Var
        I,carpim : integer;
Begin
        Carpim=1;
        For i:=1 to 10 do Carpim:=carpim*i;
        Writeln('Çarpım : ',carpim);
End.
```

Örnek 4.5.12. 10 ile 100 arasındaki tamsayılardan asal sayı olanların bulunmasını sağlayan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. I=11 al,
A3. Eğer TAM (I/2)*2=I ise A7. adıma git,
A4. Eğer TAM(I/3)*3=I ise A7. adıma git,
A5. Eğer TAM(I/5)*5=I ise A7. adıma git,
A6. Eğer TAM(I/7)*7=I ise A7. adıma git,
A7. I değerini yaz,
A8. Eğer I=99 ise A9. adıma git,
A9. I=I+2 al ve A3. adıma geri dön,
A10. Dur.



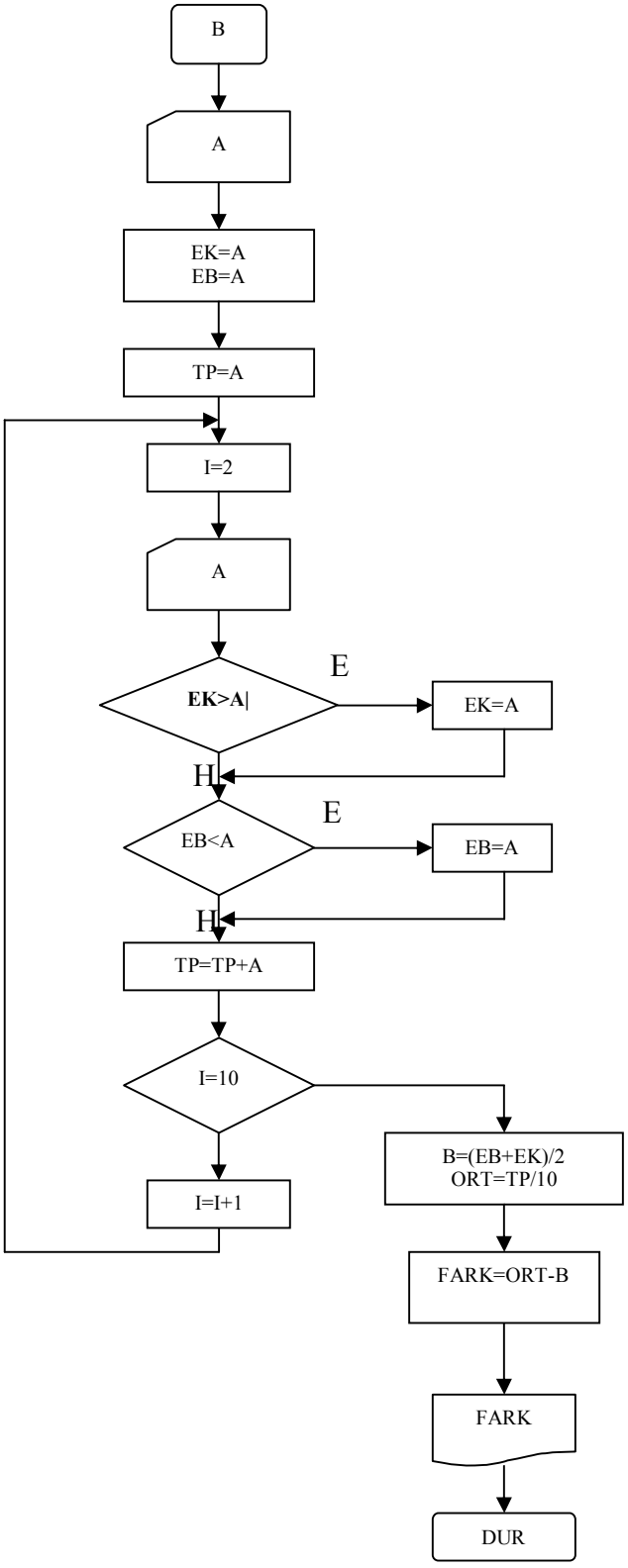
Şekil 4.5.12. 10 ile 100 arasındaki asal sayıları bulan akış şeması

Algoritmanın Pascal Dilindeki Karşılığı :

```
Program asalsayi;,
Var
        i:integer;
Begin
        For i:=11 to 100 do begin
                If (trunc(i/2)*2 <> i) and
                (trunc(i/3)*3 <> i ) and
                (trunc(i/5)*5 <> i) and
                (trunc(i/7)*7 <> i ) then
                Writeln(i);
        End;
End.
```

Örnek 4.5.13. Arka arkaya girilen rasgele 10 tamsayının ortalaması ile bu sayılardan en büyük ve en küçük olanının ortalamasını bularak elde edilen bu iki ortalamanın farkını alan bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. A' yı gir,
A3. EK =A al,
A4. EB=A al,
A5. TP=A al,,
A6. I=2 al,
A7. A' yı gir,
A8. Eğer EK>> ise EK=A al,
A9. Eğer EB<A ise EB=A al,
A10. TP=TP+A al,
A11. Eğer I=10 ise A13. adıma git,
A12. I=I+1 al ve A7. adıma gere dön,
A13. B=(EB+EK)/2 al,
A14. ORT=TP/10 al,
A15. FARK=ORT-B ,
A16. FARK' ı yaz,
A17. Dur.



Şekil 4.5.13. Girilen rasgele 10 sayıdan en büyüğü ile en küçüğünün ortalaması ile, tüm sayıların ortalamasının farkını alan akış şeması

Örnekte, A2. adımda girilen ilk A sayısı, hem küçük hem de en büyük sayı kabul edilmektedir. A3. ve A4. adımlarda, girilen A değeri EK ve EB değişkenlerine atanmaktadır. Burada EK değişkeni en küçük sayıyı belirlemek üzere tanımlanan değişkendir. A5. adımda girilen A sayısı toplam değişkeni olan TP' ye ilave edilmektedir. A6. adımdan itibaren geriye kalan 9 sayının girişi değerlendirilmesi işlemlerine başlanmaktadır. I sayacı 9 sayının girilmesini sağlamak üzere 2' den başlatılarak, A7. adımda girilen A sayısı A8. adımda EK ile A9. adımda da EB ile karşılaştırılmaktadır. Eğer EK>A ise EK yerine A atanmakta ve benzer şekilde EB<A ise EB yerine A atanmaktadır. Görüleceği üzere bu şartlardan sadece birisi gerçekleştirilebilmektedir. Girilen A sayısı yine TP değişkenine ilave edilerek işlemlere devam edilmektedir. Bu işlemler I sayacının değeri 10 oluncaya kadar devam etmektedir. I=10 olunca, A13. adıma gidilerek EB ile EK değerlerinin ortalaması B' ye atanmaktır. A14. adımda ORT olarak tanımlanan ifadeye TP/10 atanmaktadır. A15. adımda bu iki ortalamanın farkı hesaplanıp yazdırılarak işlemlere son verilmektedir.

Örnek olarak girilecek sayılar sırasıyla aşağıdaki gibi olsun.

3,-7,5,12,-3,10,8,-1,15,11

A=3 girildikten sonra,
EK=3 ve EB=3 alınarak,
TP=3 elde edilir,
I=2 için, A= -7 girildikten sonra,
EK> -7 olduğundan EK= -7 alınır,
TP = 3-7= -4 elde edilir,
I=3 için, A=5 girildikten sonra,
EB<5 olduğundan EB= 5 alınır,
TP= -4+5= 1 elde edilir,
I=4 için A=12 girildikten sonra,
EB<12 olduğundan EB=12 alınır,
TP=1+12=13 elde edilir,
I=5 için, A= -3 girildikten sonra,
Bu sayı hiçbir şarta uymadığından,
TP = 13-3= 10 elde edilir,
I=6 için, A=10 girildikten sonra,
Bu sayı hiçbir şarta uymadığından,
TP = 10+10= 20 elde edilir,
I=7 için, A=8 girildikten sonra,
Bu sayı hiçbir şarta uymadığından,
TP= 20+8=28 elde edilir,
I=8 için, A=-1 girildikten sonra,
Bu sayı hiçbir şarta uymadığından,
TP= 28-1=27 elde edilir,
I=9 için, A=15 girildikten sonra,
EB<15 olduğundan EB=15 alınır,
TP=27+15=42 elde edilir,
I=10 için, A=11 girildikten sonra,
Bu sayı hiçbir şarta uymadığından,
TP= 42+11=53 elde edilir.

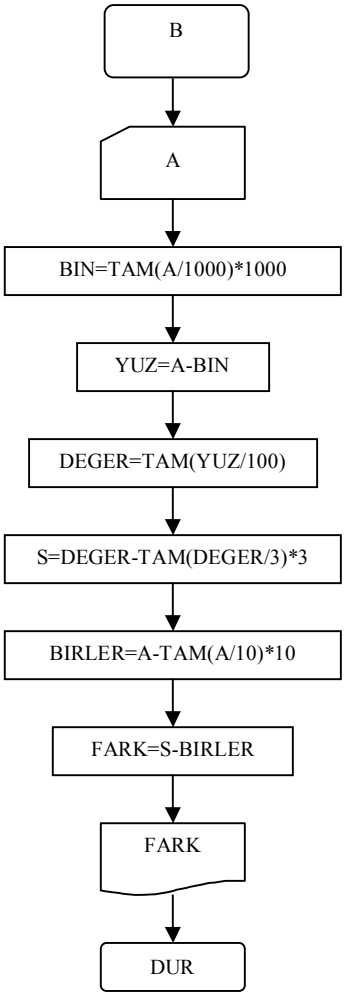
Buraya kadar girilen sayılardan en büyük, en küçük ve sayıların toplamı olan EB=15, EK= -7 ve TP=53 değerleri elde edilmiş olmaktadır.
B=(15-7)/2=4 elde edilir,
ORT=53/10=5.3 elde edilir,
FARK=5.3-4= 1.3 elde edilerek işlemlere son verilir.

Algoritmanın Pascal Dilindeki Karşılığı :

```
Var
        i,a,ek,eb,tp: integer;
        b,ort,fark: real;
begin
        write('a sayısını gir  :');  readln(a);
        ek:=a;eb:=a;tp:=a;
                for i:=2 to 10 do begin
                        write('a sayısının gir..');
                        readln (a);
                        if(ek>a) then ek:=a;
                        if(eb<a) then eb:=a;
                        tp:=tp+a;
                end;
                b:=(eb+ek)/2;
                ort:=tp/10;
                fark:= ort-b;
        writeln('fark=  ',fark);
end.
```

Örnek 4.5.14. Girilen 4 haneli tamsayının yüzler hanesindeki sayı değerinin 3 ile bölümünden elde edilen kalanın aynı sayının birler hanesindeki değer ile farkını bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. A sayısını gir,
A3. BIN=TAM(A/1000)*1000 al,
A4. YUZ=A-BIN al,
A5. DEGER=TAM(YUZ/100) al,
A6. S=DEGER- TAM(DEGER/3)*3 al,
A7. BIRLER=A-TAM(A/10)*10 al,
A8. FARK=S-BIRLER al,
A9. FARK' ı yaz,
A10. Dur.



Şekil 4.5.14. Girilen 4 haneli bir tamsayının yüzler hanesindeki sayı değerinin 3 ile bölümünden elde edilen kalanın aynı sayının birler hanesindeki değer ile farkını bulan akış şeması

Verilen örnekte 4 haneli olarak girilen bir A sayısının A3. adımda binler hanesindeki değeri bularak BIN isimli değişkene atanmaktadır. A sayısından BIN değeri çıkarılarak A4. adımda YUZ isimli değişkene atanmaktadır. Böylece girilen sayının binler hanesindeki değeri atılmaktadır. Üç haneli kalan sayının yüzler basamağındaki değeri A5. adımda DEGER isimli bir değişkene atanmaktadır. Bu değişken YUZ değerinin 100' e bölünerek tam kısmının alınmasıyla elde edilmektedir. Elde edilen DEGER isimli ifadenin 3 ile bölümünden kalanı bulmak için DEGER ifadesinin 3' e bölümünün tam kısmı alınarak tekrar üç ile çarpılıp elde edilen bu ifade DEGER isimli değişkenden çıkarılması gerekmektedir. Bu işlem A6. adımda yapılarak elde edilen sonuç S isimli bir değişkene atanmaktadır. Böylelikle girilen 4 haneli bir tamsayının yüzler hanesindeki değerinin 3 ile bölümünden elde edilen kalan bulunmaktadır. Girilen A sayısının birler hanesindeki değerini bulmak için A7. adımda A sayısından A sayısının 10' a bölümünün tam kısmının 10 ile çarpımı çıkarılarak elde edilen bu değer BIRLER isimli değişkene atanmaktadır. Böylelikle girilen sayının birler hanesindeki değeri de bulunmuş olmaktadır. Son olarak, FARK değişkenine S değeri ile BIRLER değerinin farkı yüklenerek bu ifade yazdırılıp işlemlere son verilmektedir.

Örnek olarak değeri 3764 olsun,
A3. adımda, BIN=TAM(3764/1000)1000=3000 elde edilir.
A4. adımda, YUZ=A-BIN=3764-3000=764 elde edilir.
A5. adımda, DEGER=TAM(YUZ/100)=7 elde edilir.
A6. adımda, S=DEGER- TAM(DEGER/3)*3=1 elde edilir.
A7. adımda, BIRLER=A-TAM(A/10)*10=3764-3760=4 bulunur.
A8. adımda, FARK=S-BIRLER =1-4= -3 elde edilir.

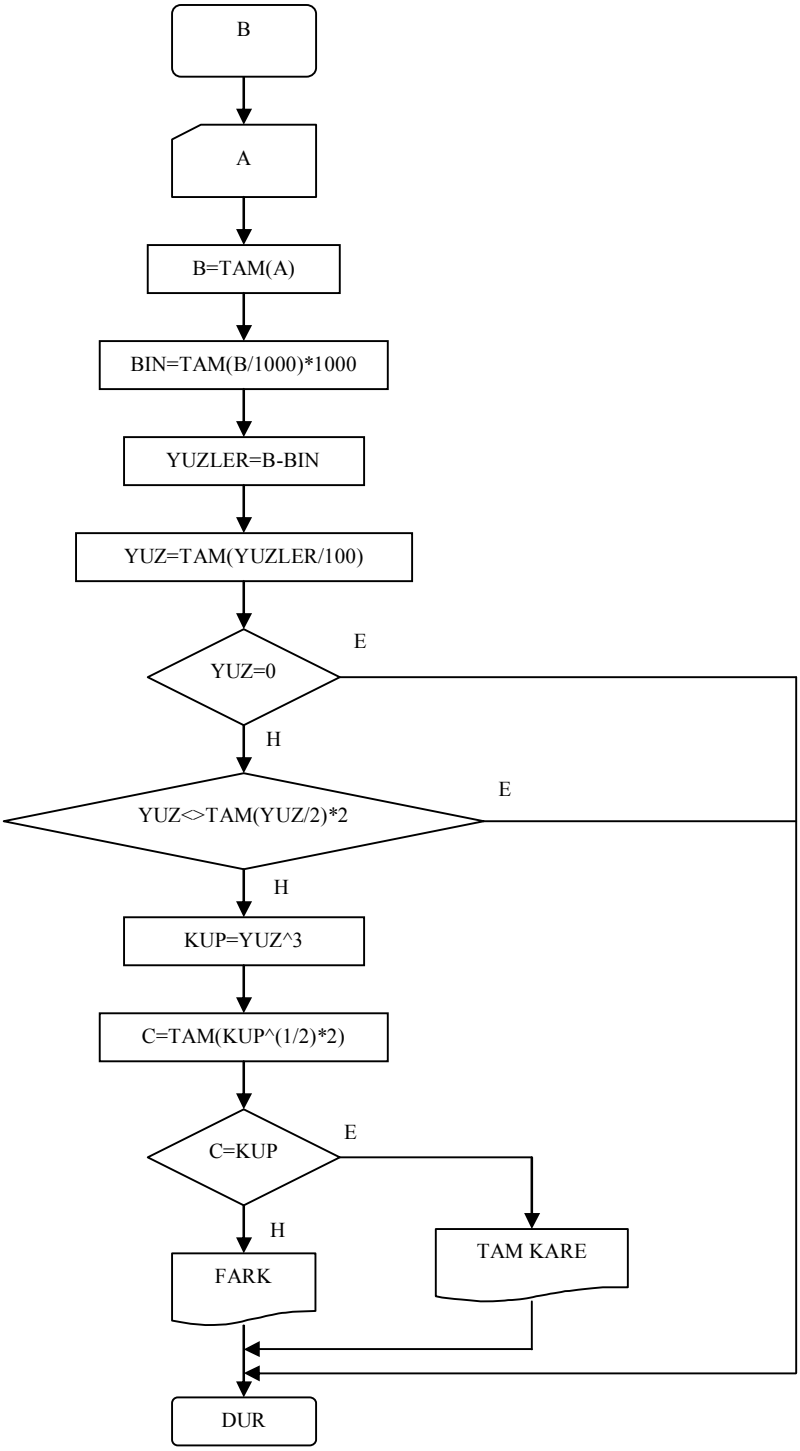
Sonuçta –3 yazdırılarak işlemlere son verilir.

Algoritmanın Pascal Dilindeki Karşılığı :

```
Var
        a,bin,yuz,deger,s,birler,fark: integer;
begin
        write('a sayısını gir  :'); readln(a);
        bin:=trunc(a/1000)*1000;
        yuz:=a-bin;
        deger:=trunc(yuz/100);
        s:=deger-trunc(deger/3)*3;
        birler:=a-trunc(A/10)*10;
        fark:=s-birler;
        writeln('fark :',fark);
        readln;
end.
```

Örnek 4.5.15. Tam kısmındaki değeri en fazla 4 haneli girilen bir rasyonel sayının tam kısmının yüzler hanesindeki değeri çift ise bu değerin küpünün bir tam kare olup olmadığını bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. A sayısını gir,
A3. B=TAM(A) al,
A4. BIN=TAM(B/1000)*1000 al,
A5. YUZLER=B-BIN al,
A6. YUZ=TAM(yuzler/100) al,
A7. Eğer YUZ=0 ise A14. adıma git,
A8. Eğer YUZ<>TAM(YUZ/2)*2 ise A14. adıma git,
A9. KUP=YUZ^3 al,
A10. C=TAM(KUB^(1/2))^2 al,
A11. Eğer C=KUB ise A13. adıma git,
A12. "Tam kare değil" yaz ve A14. adıma git,
A13. "Tam kare " yaz,
A14. Dur.



Şekil 4.5.15. Girilen bir rasyonel sayının tam kısmının yüzler hanesindeki değeri çift ise bu değerin küpünün bir tam kare olup olmadığını bulan akış şeması

Örnekte girilen bir A rasyonel sayısının tam kısmı a3. adımda B değişkenine atanmaktadır. Elde edilen B ifadesinin yüzler hanesindeki değerini bulabilmek için bu sayının binler hanesindeki değerinin bulunarak b sayısından çıkarılması gerekmektedir. Bu işlemler A4. adımda B sayısının binler hanesindeki değerinin bulunarak A5. adımda B sayısından çıkarılması ve elde edilen yüzler hanesinin YUZ isimli değişkene atanmasıyla sağlanmaktadır. YUZ isimli değişkenin değeri sıfır ya da çift ise bu sayı

dikkate alınmayacaktır. Bu sorgulama A6. ve A7. adımlarda sorulanarak karar verilmektedir. Aksi halde sayı sıfırdan farklı ve çift olacağından bu sayının kübünün bir tam kare olup olmadığının araştırılmasına geçilmektedir. Bunun için A8. adımda YUZ değerinin kübü alınarak KUB isimli değişkene atanmaktadır. A9. adımda elde edilen bu ifadenin karekökünün tam kısmının karesi alınarak C değişkenine atanmaktadır. Elde edilen C değeri ile KUB değeri eşit ise sayı tam kare olacaktır. Bu durum A10. adımda sorgulanarak gerekli sonuçlar yazdırılıp işlemlere son verilmektedir.

Bu algoritmanın değişik örneklerle gösterimi :

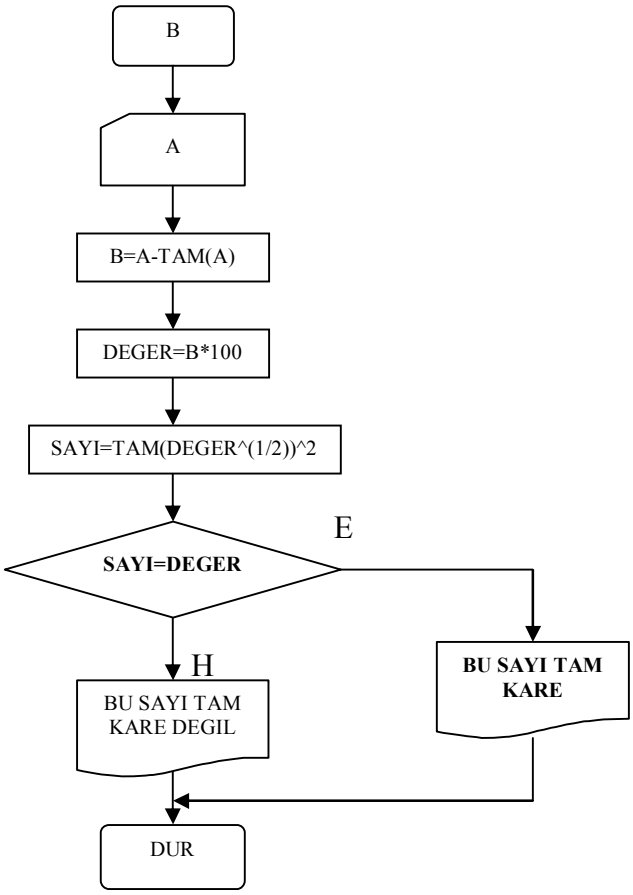
1. Girilen sayı A=23,456 olsun,
   B=TAM(A)=23 elde edilir.
   BIN=TAM(B/1000)*1000=0 elde edilir.
   YUZLER=B-BIN=23 elde edilir.
   YUZ=TAM(YUZLER/100)=0 olacağından, A13. adıma gidilerek işlemlere son verilmektedir.
2. Girilen sayı A=5924,645 olsun,
   B=TAM(A)=5924 elde edilir.
   BIN=TAM(B/1000)*1000=5000 elde edilir.
   YUZLER=B-BIN=924 elde edilir.
   YUZ=TAM(YUZLER/100)=9 elde edilir.
   9<>8 olduğundan (YUZ<>TAM(YUZ/2)*2) bu sayı çift olmayacaktır.
   Dolayısıyla A14. adıma gidilerek işlemlere son verilmektedir.
3. Girilen sayı A=2456,875 olsun,
   B=TAM(A)=2456 elde edilir.
   BIN=TAM(B/1000)*1000=2000 elde edilir.
   YUZLER=B-BIN=456 elde edilir.
   YUZ=TAM(YUZLER/100)=4 elde edilir.
   4=4 olduğundan (YUZ<>TAM(YUZ/2)*2),
   KUB=YUZ^3=64 elde edilir.
   C=TAM(64^(1/2))^2=64 elde edilir.
   KUB=C (64=64) olduğundan bu sayı istenilen şarta uygundur.

Algoritmanın Pascal Dilindeki Karşılığı:

```
Var
        A: real;
        B,bin,yuzler,yuz,kub,c: integer;
begin
        write('a sayısını gir  :'); readln(a);
        b:=trunc(a);
        bin:=trunc(a/1000)*1000;
        yuzler:=b-bin;
        yuz:=trunc(yuzler/100);
        if(yuz=trunc(yuz/2)*2) then begin
                kub:=yuz^3; c:=trunc(sqrt(kub))^2;
                if(c=kub) then writeln('tam kare')
                else writeln('tam kare değil');
        end;
end.
```

Örnek 4.5.16. Ondalıklı kısmı iki haneli girilen pozitif bir rasyonel sayının ondalıklı kısmının sayı değerinin bir tam kare olup olmadığını araştıran algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. A sayısını gir,
A3. B=A-TAM(A) al,
A4. DEGER=B*100 al,
A5. SAYI=TAM(DEGER^(1/2))^2 al,
A6. Êğer SAYI=DEGER ise A8. adıma git,
A7. "Bu sayı tam kare değil" yaz ve A9. adıma git,
A8. "Bu sayı tam karedir" yaz,
A9. Dur.



Şekil 4.5.16. Ondalıklı kısmı iki haneli girilen pozitif bir rasyonel sayının ondalıklı kısmının sayı değerinin bir tam kare olup olmadığını bulan akış şeması.

A2. adımda girilen bir A sayısının A3. adımda ondalıklı kısmı bulunarak B sayısına atanmaktadır. Bulunan değer A sayısından TAM kısmı çıkarılarak elde edilmektedir. Elde edilen B sayısı A4. adımda 100 ile çarpılarak tam sayıya çevrilmektedir. Bu değer DEGER isimli bir değişkene atanmaktadır. Bu aşamadan sonra bulunan DEGER ifadesinin bir tam kare olup olmadığını araştırmaktadır. Eğer DEGER ifadesinin karekökünün tam kısmının karesi yine DEGER ifadesine eşit ise bu ifade tam kare olacaktır. Aksi halde tam kare olmayacaktır. Bu durum A6. adımda sorgulanarak bu doğrultuda karar verilmektedir.

Bu algoritmanın bir örnekle gösterimi:

Eğer A=23,67 olsaydı,
B=23,67-TAM(23,67) =0,67 elde edilir.
DEGER=0,67*100=67 olarak bulunur.
SAYI=TAM(67^(1/2))^2=TAM(8,1853)^2=64 elde edilir.
SAYI<>DEGER olacağından,
"Bu sayı tam kare değildir" yazdırılarak işlemlere son verilir.

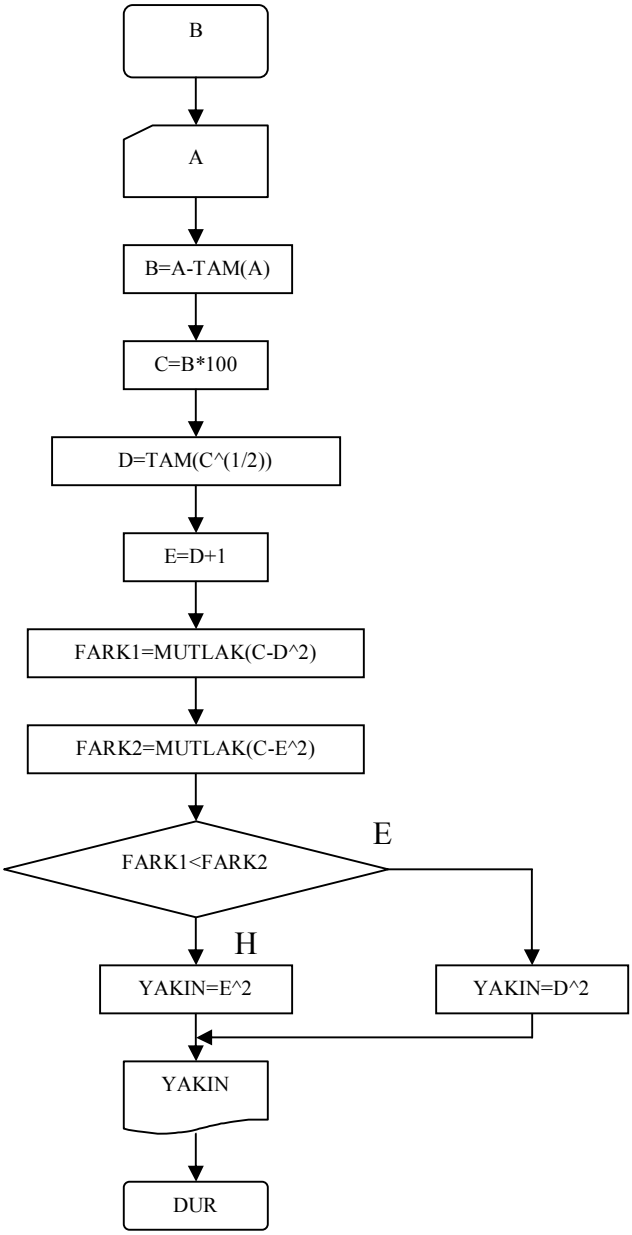
Eğer A=23,36 olsaydı,
B=23,36-TAM(23,36) =0,36 elde edilir.
DEGER=0,36*100=36 olarak bulunur.
SAYI=TAM(36^(1/2))^2=TAM(6)^2=36 elde edilir.
SAYI=DEGER olacağından,
"Bu sayı tam karedir" yazdırılarak işlemlere son verilir.

Algoritmanın Pascal Dilindeki Karşılığı:

```
Var
        a:real;
        b,deger,sayi: integer;
begin
        write('a sayısını gir : '); readln(a);
        b:=a-trunc(a);
        deger:=b*100;
        sayi:=sqr(trunc(deger));
        if(sayi=deger) then
                writeln('bu sayı tam karedir..')
        else
                writeln('bu sayı tam kare değildir..');
end.
```

Örnek 4.5.17. Ondalıklı kısmı iki haneli girilen pozitif bir rasyonel sayının ondalıklı kısmının tamsayı değerinin en yakın tam kare sayıya uzklığınıbulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. A sayısını gir,
A3. B=A-TAM(A) al,
A4. C=B*100 al,
A5. D=TAM(C^(1/2)) al,
A6. E=D+1 al,
A7. FARK1=MUTLAK(C-D^2) al,
A8. FARK2MUTLAK(C-E^2) al,
A9. Eğer FARK1<FARK2 ise A11. adıma git,
A10 YAKIN=E^2 al ve A12. adıma git,
A11. YAKIN=D^2 al,
A12. YAKIN değerini yaz,
A13. Dur.



Şekil 4.5.17. Ondalıklı kısmı iki haneli girilen bir rasyonel sayının ondalıklı kısmının sayı değerinin en yakın tam kare sayıya uzaklığını bulan akış şeması.

Girilen A rasyonel sayısının ondalıklı kısmı A3. adımda hesaplanarak B değişkenine atanmaktadır. A4. adımda elde edilen bu değer 100 ile çarpılarak ondalıklı kısmının tamsayı değeri hesaplanıp C değişkenine atanmaktadır. A5. adımda C değerinin karekökünün tam değeri alınarak D değişkenine atanmaktadır. C değerine en yakın tam kare sayı da D değerinin karesi olacaktır. Ya da D+1 değerinin karesi olacaktır. Bu durum için FARK1 ve FARK2 isimli iki değişken tanımlanarak bunlardan FARK1' e C değeri ile D^2 değerinin farkının mutlak değeri, FARK2' ye de C değeri ile D+1^2' ye karşılık gelen E^2 değerinin farkının mutlak değeri atanmaktadır. Bu aşamadan sonra FARK1 ve FARK2 değerlerinin karşılaştırılmasına geçilmektedir. Eğer FARK2<FARK1 ise D^2 yakın tamsayı olacağından YAKIN olarak D^2 alınmaktadır. Aksi halde E^2 yakın tam kare olacağından YAKIN yerine E^2 alınarak sonuç yazdırılıp işlemlere son verilmektedir.

Örnek olarak A=25,45 alınırsa,

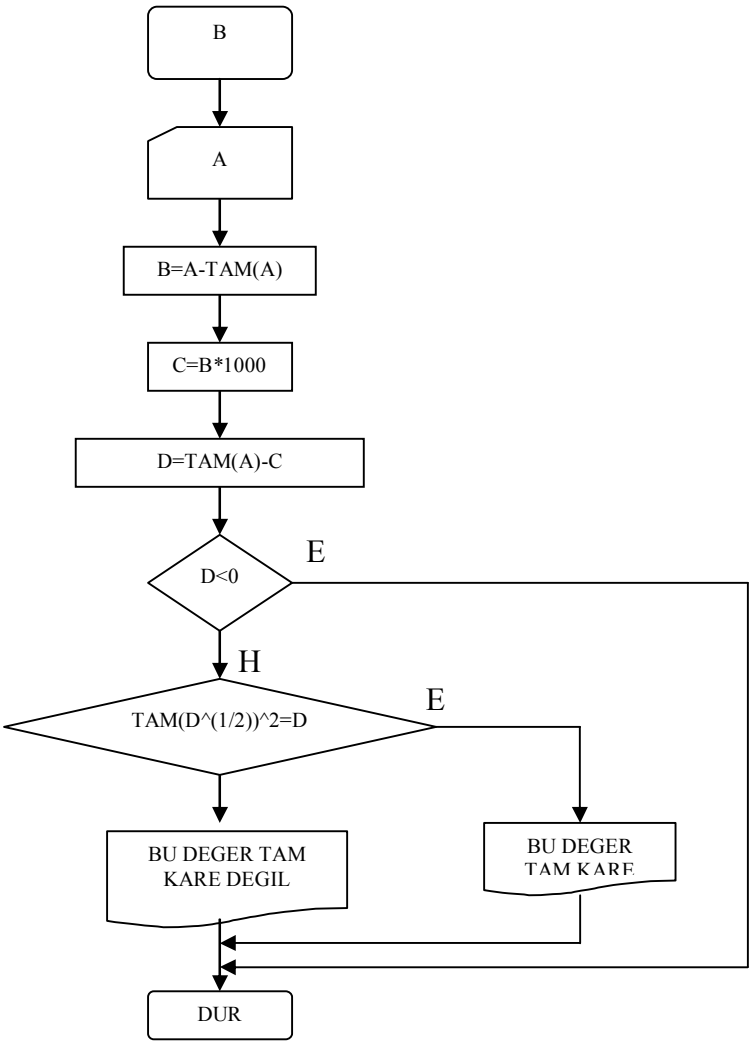
B=25,45-TAM(25,45)=0,45 elde edilir.
C=0,45*100=45 ifadesinden,
D=TAM(45^2(1/2))=6 elde edilir.
E=6+1=7 alınarak,
FARKI=MUTLAK(45-36)=9 bulunur.
FARK2=MUTLAK(45-49)=4 elde edilir.
9>4 olacağından,
FARK=7^2=49 elde edilir.

Algoritmanın Pascal Dilindeki Karşılığı:

```
Var
        a,k: real;
        b,c,d,e,fark1,fark2,yakin: integer;
begin
        readln(a);
        b:=trunc(a); k:=a-b;
        c:=trunc(k*100+0.5);
        d:=trunc(sqrt (c));
        e:=d+1;
        fark1:=trunc(c-d*d);
        fark2:=trunc(c-e*e);
                if(fark1<fark2) then
                        yakin:=d*d
                else
                        yakin:=e*e;
        writeln('en yakın tamkare sayı...',yakin);
end.
```

Örnek 4.5.18. Ondalıklı kısmı üç haneli girilen pozitif bir rasyonel sayının tam kısmı ile ondalıklı kısmının tamsayı değerinin farkı pozitif ise bu değerin bir tam kare olup olmadığını araştıran algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. A sayısını gir,
A3. B=A-TAM(A) al,
A4. C=B*100 al,
A5. D=TAM(A)-C al,
A6. Eğer D<0 ise A10. adıma git,
A7. Eğer TAM(D^(1/2))^2=D ise A9'a git,
A8. "Bu değer tam kare değil" yaz ve A10. adıma git,
A9. "Bu sayı tam kare" yaz,
A10. Dur.



Şekil 4.5.18. Ondalıklı kısmı üç haneli girilen rasyonel sayının tam kısmı ile ondalıklı kısmının tamsayı değerinin farkı pozitif ise bu değerin bir tam kare olup olmadığını araştıran akış şeması

Örnek algoritmada girilen A sayısının ondalıklı kısmının tamsayı değeri A4. adımda C değişkenine atanmaktadır. A5. adımda A sayısının tam kısmıyla ondalıklı kısmın tamsayı değerinin farkı alınarak D isimli bir değişkene atanmaktadır. Elde edilen bu değer sıfırdan küçük ise işlemlere son verilmektedir. Aksi halde bu değerin bir tam kare olup olmadığının araştırılmasına geçilmektedir. Bunun için D değerinin karekökünün tam kısmının karesi ile D değeri A7. adımda karşılaştırılmaktadır. Eğer bu iki ifade eşit ise bu sayı tam kare olacağından, tam kare olduğu yazdırılarak işlemlere son verilmektedir. Aksi halde bu değer tam kare olamayacağından bu durum yazdırılarak işlemlere son verilmektedir.

Örnek olarak eğer a=23,456 alınırsa,
B=23,456-23=0,456 elde edilir.
C=0,456*1000=456 ifadesinden,
D=23-456=433 elde edilecektir.
D<0 olacağından işlemlere son verilmektedir.

Eğer A=897,456 alınırsa,
B=897,456-897=0,456 elde edilir.
C=0,456*1000=456 ifadesinden,
D=897-456=852 elde edilecektir.
D>0 olacağından,
TAM(852^(1/2))^2=841 ifadesinden,
852<>841 olacağından, bu ifade tam kare olmayacaktır.

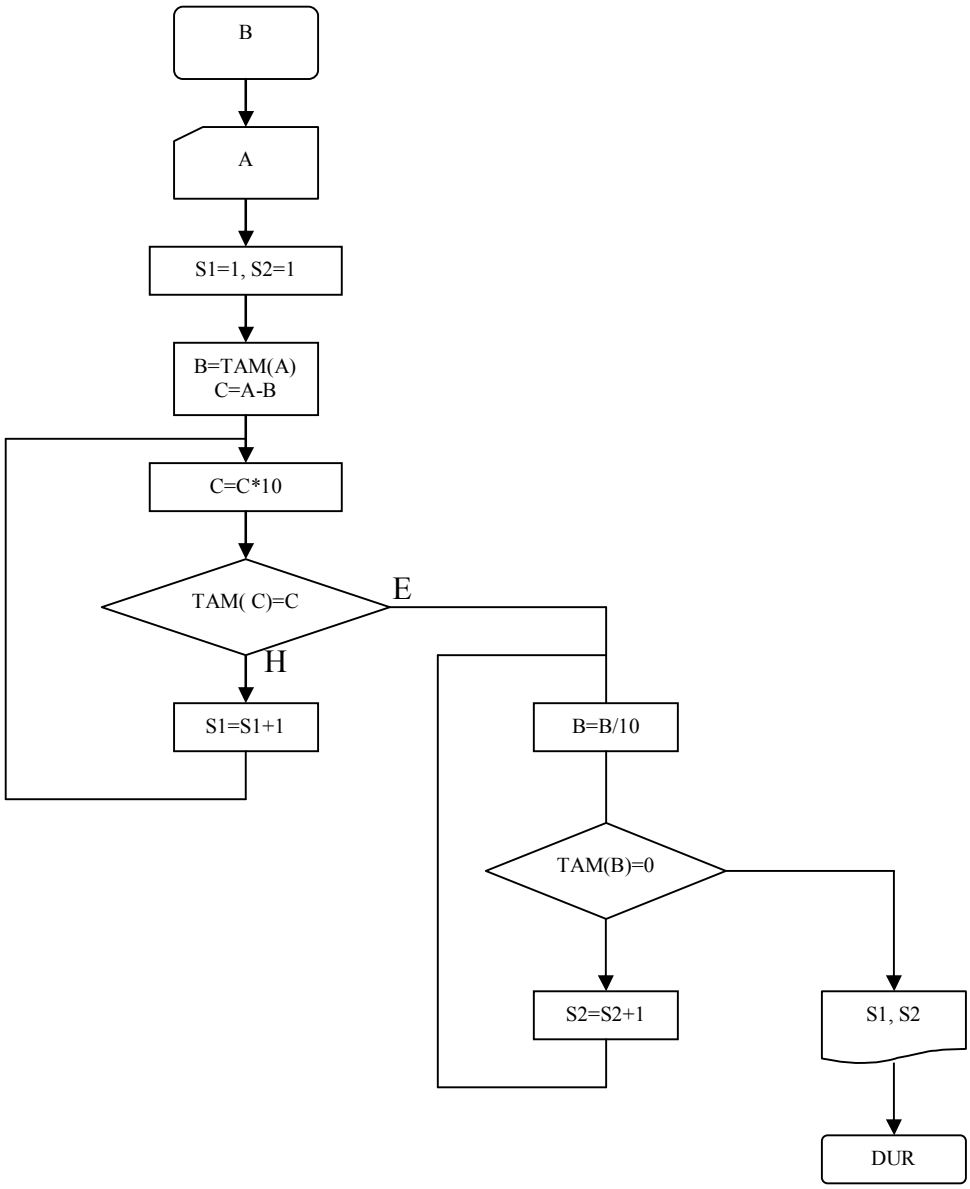
Eğer A=649,424 alınırsa,
B=649,424-649=0,424 elde edilir.
C=0,424*1000=424 ifadesinden,
D=649-424=225 elde edilecektir.
D>0 olacağından,
TAM(225^(1/2))^2=225 bulunur.
225=225 sonucundan bu ifade tam kare olacaktır.

Algoritmanın Pascal Dilindeki Karşılığı:

```
Var
        a,k: real;
        b,c,d,e: integer;
begin
        readln(a);
        b:=trunc(a);
        k:=a-b;
        c:=trunc(k*1000+0.5);
        d:=b-c;
        e:=trunc(sqrt(d));
        if(d>=0 and e*e=d) then
                writeln('bu sayı tam karedir...')
        else  writeln('bu tamkare değildir..');
end.
```

Örnek 4.5.19. Rasgele girilen bir rasyonel sayısın ondalıklı kısmının ve tam kısmının hane sayısını bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. A sayısını gir,
A3. S1=1,S2=1 al,
A4. B=TAM(A) al,
A5. C=A-B al,
A6. C=C*10 al,
A7. Eğer TAM(C) =C ise A9'a git,
A8. S1=S1+1 al A6'a geri dön,
A9. B=B/10 al,
A10. Eğer TAM(B)=0 ise A12'ye git,
A11. S2=S2+1 al ve A9'a geri dön,
A12. S1 ve S2' yi yaz,
A13. Dur.



Şekil 4.5.19. Rasgele girilen bir rasyonel sayısın ondalıklı kısmının ve tam kısmının sayı değerinin hane sayısını bulan akış şeması.

Örnek olarak A=765,345 alınırsa,
B=765 ve C=0,345 elde edilir.
A3. adımda S1=1 ve S2=1 alınarak,
A6. adımda yeni C=3,45 elde edilir.
TAM(C)=3 olacağından eşitlik elde edilemez ve S1=2 olur.
Bu kez yeni C=34,5 elde edilir,
TAM(C)=34 olacağından TAM(C)=C bulunur.
Bu aşamada ondalıklı kısmın hane sayısı bulunmuş olur S1=3 tür.

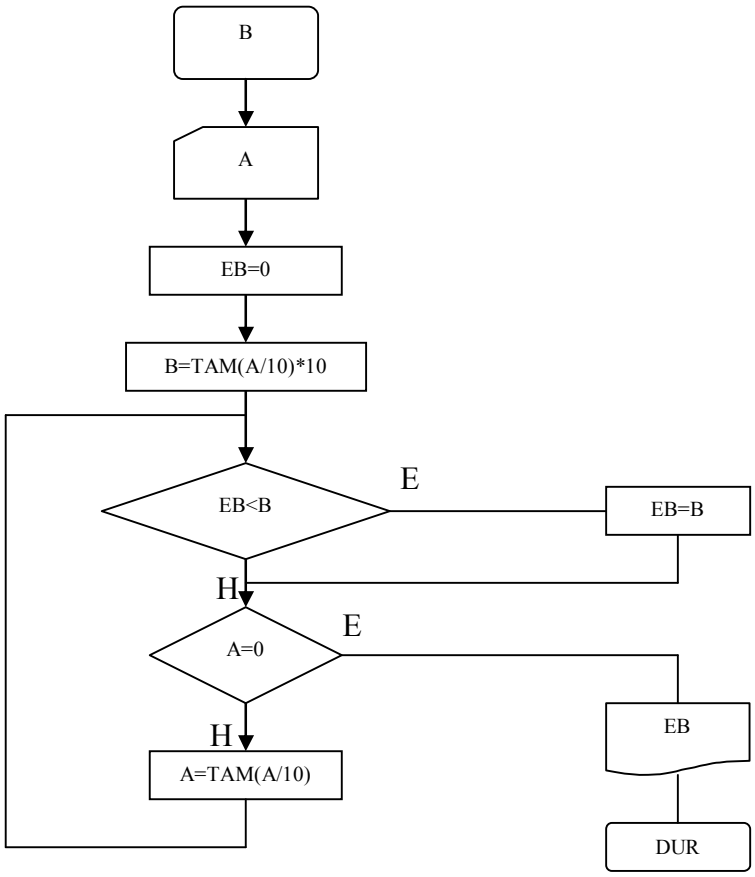
Benzer şekilde,
A9. adımda yeni B=76,5 elde edilir.
TAM(B)=76 olacağından S2=2 elde edilir..
Bu kez yeni B=7,65 elde edilir.
TAM(B)=7 olacağından S2=3 elde edilir..
Bu kez yeni B=0,765 elde edilir.
TAM(B)=0 olacağından girilen sayının tam sayının hane sayısı bulunmuş olur ve bu değer S2=3 tür

Algoritmanın Pascal Dilindeki Karşılığı:

```
Var
        a,b,d,e: real;
        b,s1,s2,k: integer;
begin
        s1:=1; s2:=0; readln(a);
        b:=trunc(a);
        c:=a-b;
        c:=c*10;
        k:=trunc(c);
                while(k<>trunc(c)) do begin
                        c:=c*10;
                        k:=trunc(c);
                        s1:=s1+1;
                end;
                while(trunc(b)<>0) do begin
                        b:=b/10;
                        s2:=s2+1;
                end;
        writeln(s1,'',s2);
end.
```

Örnek 4.5.20 Girilen bir tam sayının hanelerindeki en büyük sayıyı bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. A sayısını gir,
A3. EB=0 al,
A4. B=A-TAM(A/10)*10 al,
A5. Eğer EB<B ise EB=B al,
A6. Eğer A=0 ise A9. adıma git,
A7. A=TAM(A/10) al,
A8. A4. adıma geri dön,
A9. EB' yi yaz,
A10. Dur.



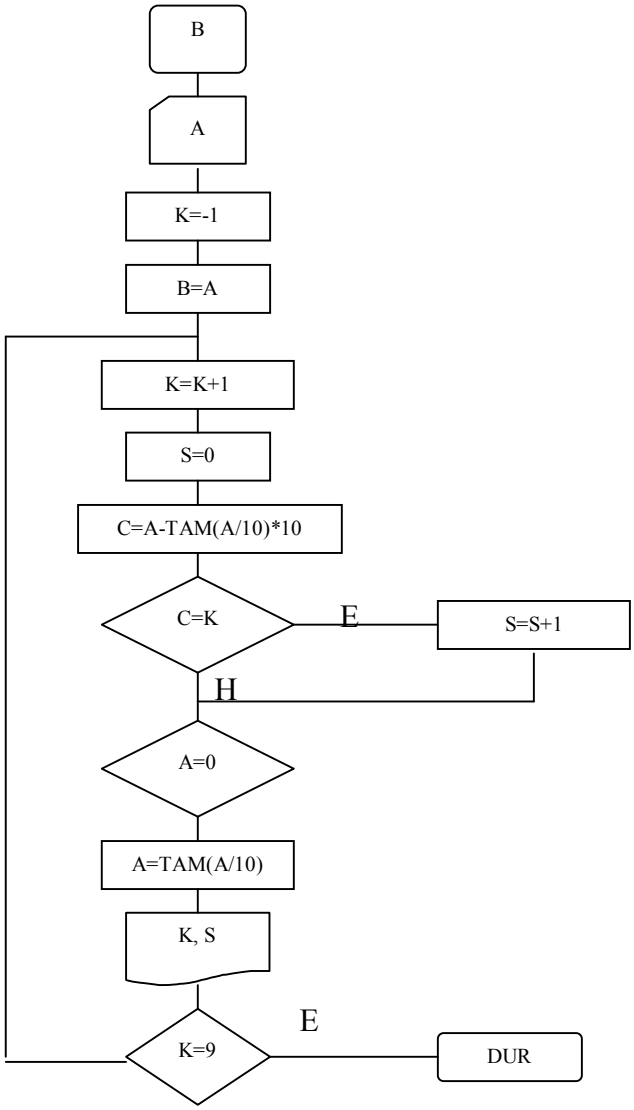
Şekil 4.5.20. Girilen bir tam sayının hanelerindeki en büyük sayıyı bulan akış şeması.

Algoritmanın Pascal Dilindeki Karşılığı:

```
Program tamsayılar;
Var
        a,eb,b: integer;
begin
        eb:=0;
        write(‘ a sayısını giriniz :’);
        readln(a);
        while(a<>0) do
        begin
                b:=a-trunc(a/10)*10;
                if(eb<b) then eb:=b;
                a:=trunc(a/10);
        end;
        writeln(‘en büyük  :’,eb);
end.
```

Örnek 4.5.21. Girilen bir tamsayının hanelerindeki tekrar eden sayısıları bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. A sayısını gir,
A3. K=-1 al,
A4. B=A al,
A5. K=K+1 al,
A6. S=0 al,
A7. C=A-TAM(A/10)*10 al,
A8. Eğer C=K ise S=S+1 al,
A9. Eğer A=0 ise A11. adıma git,
A10. A=TAM(A/10) al ve A7. adıma git,
A11. K ve S' yi yaz,
A12. Eğer K=9 ise A14. adıma git,
A13. A=B al ve A5. adıma geri dön,
A14. Dur.



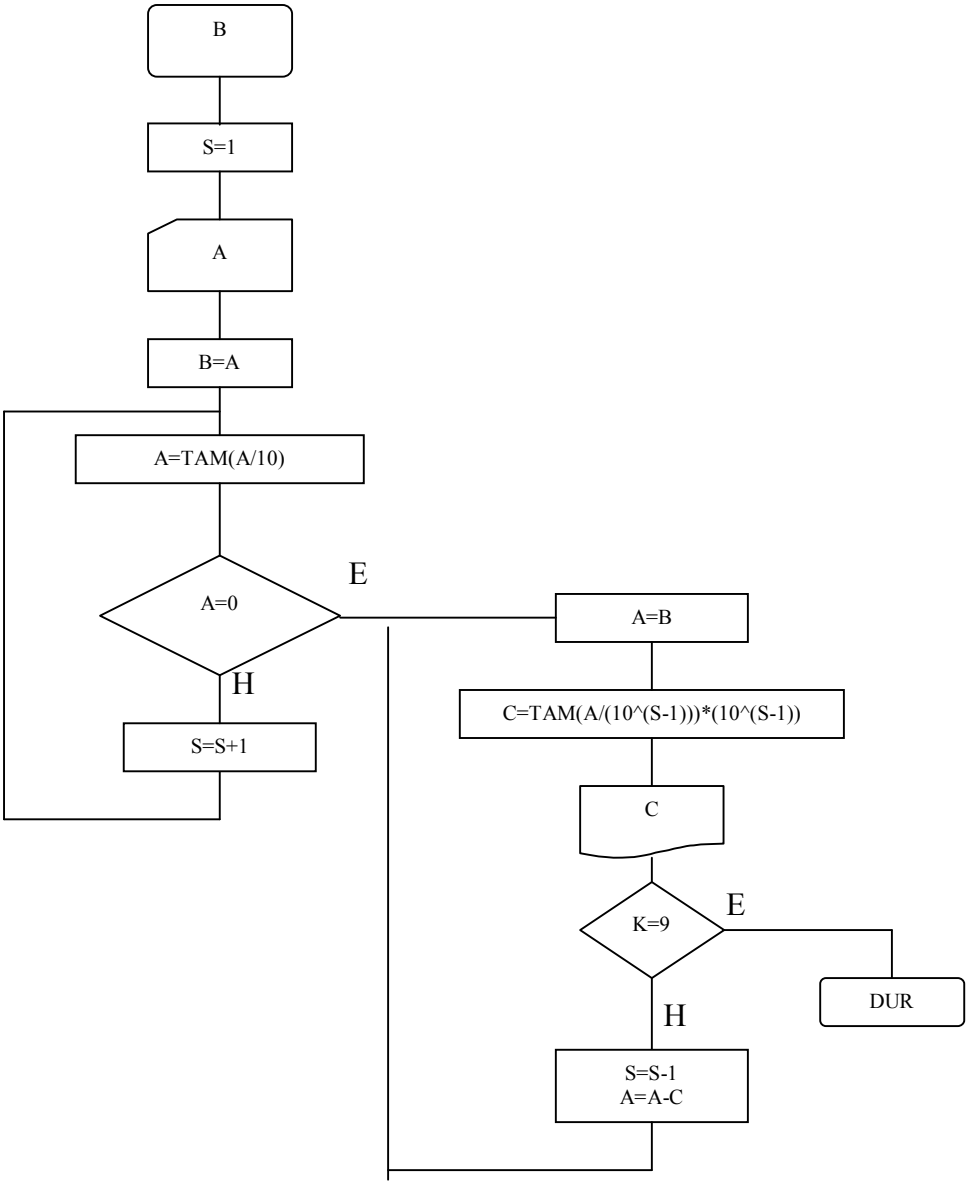
Şekil 4.5.21. Girilen bir tamsayının hanelerindeki tekrar eden sayıları bulan akış şeması.

Algoritmanın Pascal Dilindeki Karşılığı:

```
Program tekrarlisayi;
Var
        a,b,c,k,s: integer;
begin
        write('a sayısını giriniz :');
        readln(a);
        b:=a;
        for k:=0 to 9 do
        begin
                s:=0;
                while(a<>0) do
                begin
                        c:=a-trunc(a/10)*10;
                        if(c=k) then
                                s:=s+1;
                        a:=trunc(a/10);
                end;
                writeln(k,' sayısı  ',s,' kere tekrar edilmiştir.');
        a=b;
        end;
end.
```

Örnek 4.5.22. Girilen herhangi bir tamsayının hanelerine ayrılmasını sağlayan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. S=1 al,
A3. A sayısını gir,
A4. B=A al,
A5. A=TAM(A/10) al,
A6. Eğer A=0 ise A9. adıma git,
A7. S=S+1 al,
A8. A5. adıma geri dön,
A9. A=B al,
A10. C=TAM(A/(10^(S-1)))*(10^(S-1)) al,
A11. C' yi yaz,
A12. Eğer S=1 ise A13. adıma git,
A13. A=A-C, S=S-1 al ve A8. adıma geri dön,
A14. Dur.



Şekil 4.5.22. Girilen bir tamsayının hanelerine ayrılmasını sağlayan akış şeması.

Algoritmanın Pascal Dilindeki Karşılığı:

```
Program hanelere_ayırma;
Var
        a,b,c,d: integer;
begin
        s:=1;
        write('a sayısını giriniz :'); readln(a);
        b:=a;
        while(a<>0) do begin
                a:=trunc(a/10);
                s:=s+1;
        end;
        a:=b;
```

```
        while(s>0) do begin
                d:=exp(10*in(s-1));
                c:=trunc(a/d)*d;
                writeln(c);
                s:=s; a:=a-c;
        end;
        readln;
end.
```

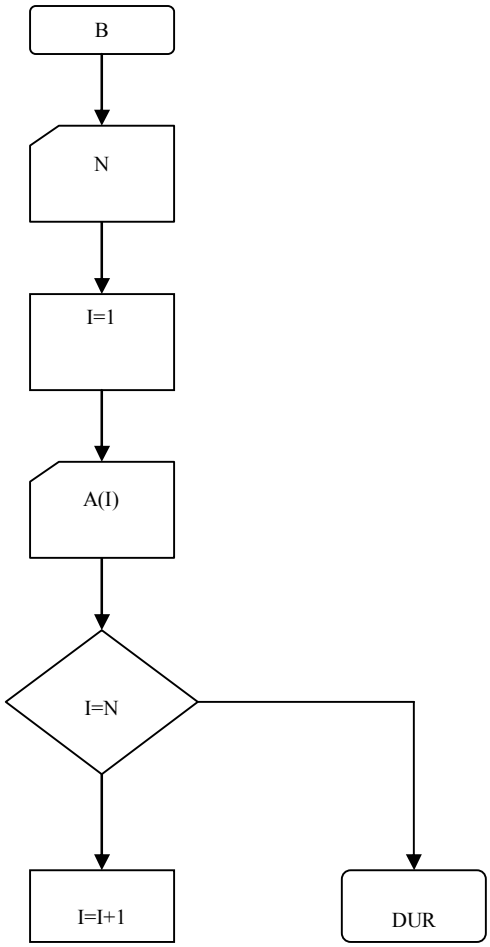
4.6. DİZİLER VE DİZİLERE İLİŞKİN ALGORİTMA VE AKIŞ ŞEMALARI

4.6.1. Dizi Kavramı ve Örnekler

Değişken isimlendirilirken her bir değişken için bir ad kullanılmak zorundadır. Bu ise değişken sayısının çok fazla olması durumunda uygun bir yöntem değildir. Bu durumda benzer özelliklere sahip elemanları bir küme gibi düşünerek bu doğrultuda işlem yapmak daha uygun olacaktır. Belli özelliklere sahip bu elemanların oluşturdukları kümeleri dizi olarak adlandıracağız. Diziler indisli değişkenler kullanılarak isimlendirilirler. Örneğin N elamanlı bir dizi için N1 dizininin birinci elemanını ve N' de dizinin son elemanını tanımlamaktadır. Dizi adı herhangi bir değişken adı olabilir; ancak indis parantez içerisine alınmalıdır. Örneğin 10 elemanlı bir A dizisinin elemanını sırasıyla A(1),A(2),A(3),....,A(10) şeklinde ifade etmek gerekir.

Örnek 4.6.1. N elemanlı bir sayı dizisinin girişini yapan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. N' yi giriniz,
A3. I=1 al,
A4. A(I)' ı gir,
A5. Eğer I=N ise A7. adıma git,
A6. I=I+1 al ve A4 adıma geri dön,
A7. Dur.



Şekil 4.6.1. N elemanlı bir sayı dizisinin girişini yapan akış şeması

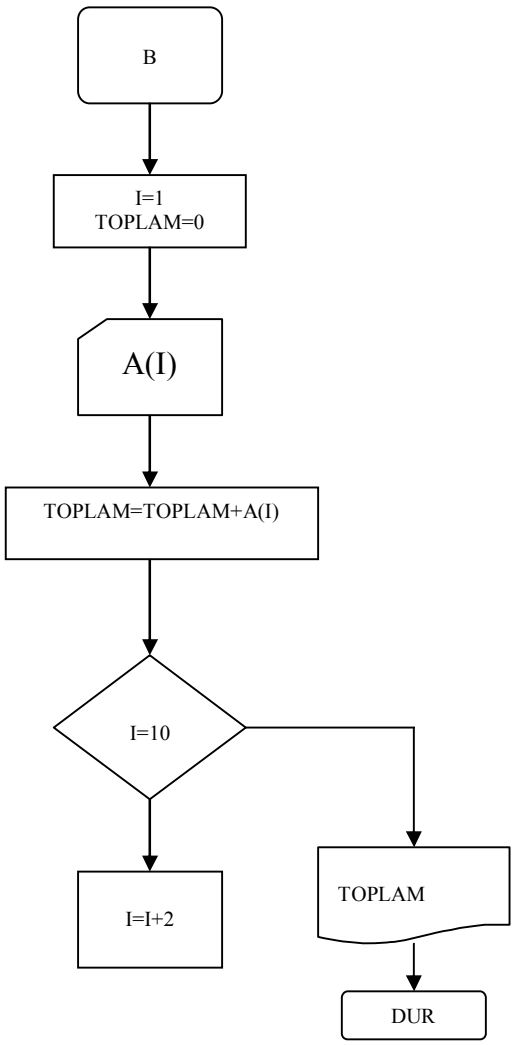
Örnekte N tane dizi elemanın girişinin yapılabilmesi için A2. adımda N değerinin girilmesi istenmektedir. 1' den N' ye kadar artışı sağlayacak bir I indisi tanımlanarak A3. adımda buna ilk değe olarak I atanmıştır. Sonraki adım olan A4. adımda I indisinin aldığı değerlere göre A dizisinin elemanlarının girişi yapılmaktadır. Bu işlemler I indisinin değeri N oluncaya kadar devam etmektedir. Bu sorgulama A5. adımda yapılarak istenilen şartın gerçekleşmesi durumunda işlemlere son verilmektedir. Aksi halde I indisinin değeri I artırılarak A4. adımdan itibaren işlemlere devam edilmektedir.

Algoritmanın Pascal Dilindeki Karşılığı:

```
Program_dizigirisi;
Var
        n,i:integer;
        a:array[1..100] of integer;
begin
        write('n sayısını giriniz  :'); redaln(n);
        for i:=1 to n do begin
                write('dizi elemanlarını gir  :'); readln(a[i]);
        end;
end.
```

Örnek 4.6.2. 10 elemanlı bir sayı dizisinin elemanlarının toplamını bulan algoritma ve akış şemasının oluşturulması.

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.



Şekil 4.6.2. 10 elemanlı bir sayı dizisinin elemanlarının toplamını bulan akış şeması.

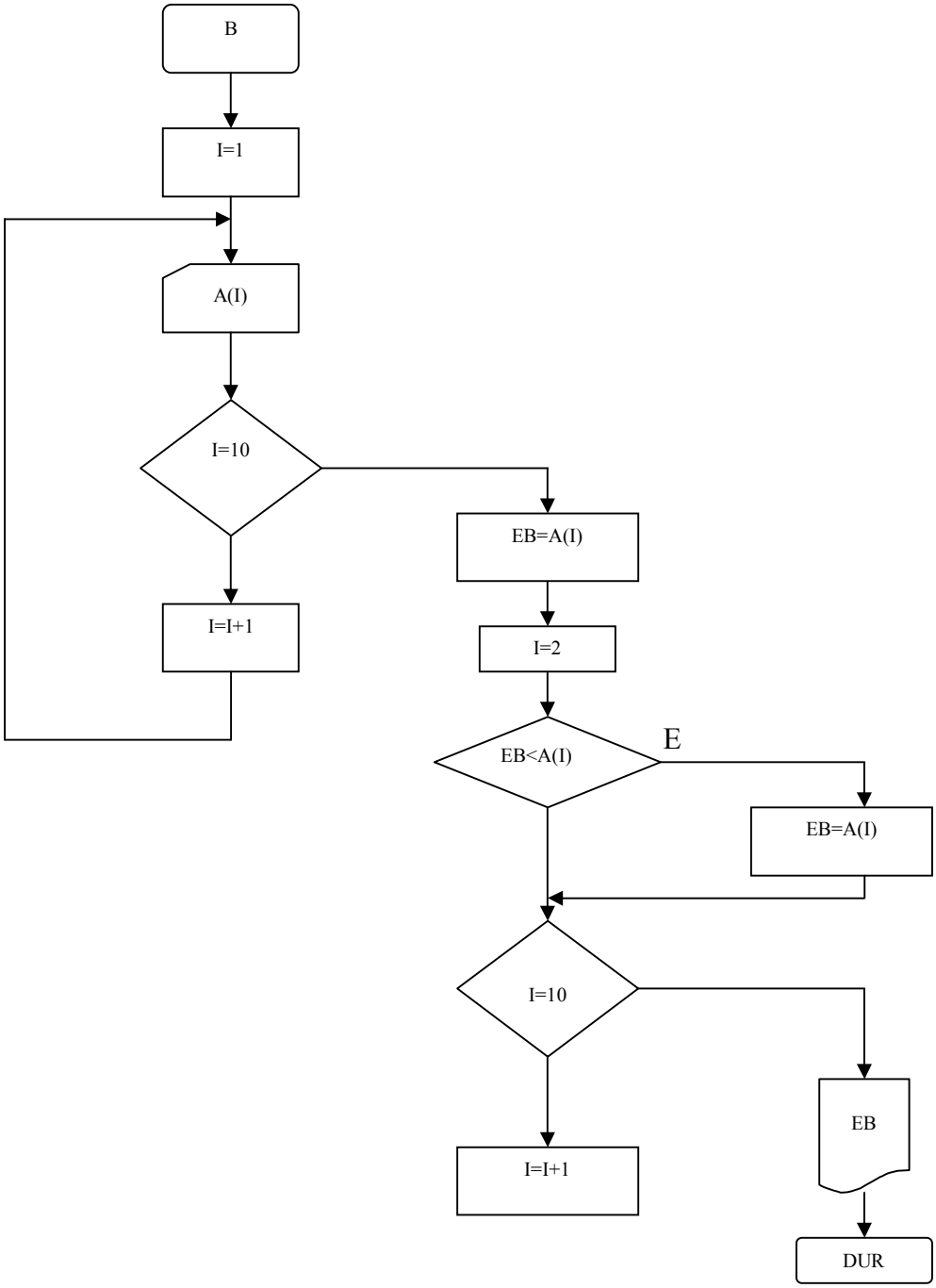
Verilen örnekte A2. adımda, bir I indisi ve TOPLAM değişkeni tanımlamıştır. Burada tanımlanan I indisi 1' den 10' a kadar artırılarak A dizisinin elemanlarının girişi yapılmaktadır ve girilen her eleman TOPLAM değişkene ilave edilerek toplatılmaktadır. A3. adımda A dizisinin I' yıncı elemanı girilerek 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.

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  :');
                redaln(a[i]);
                toplam:=toplam+a[i];
        end;
        writeln('toplam = ',toplam);
end.
```

Örnek 4.6.3. 10 elemanlı bir sayı dizisinin en büyük elemanını bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. I=1 al,
A3. A(I)' yı gir,
A4. Eğer I=10 ise A6. adıma git,
A5. I=I+1 al ve A3. adıma geri dön,
A6. EB=A(I) al, I=2 al,
A7. Eğer EB<A(I) ise EB=A(I) al,
A8. Eğer I=10 ise A10. adıma git,
A9. I=I+1 al ve A7. adıma geri dön,
A10. EB' yi yaz,
A11. Dur.



Şekil 4.6.3. 10 elemanlı bir sayı dizisinin en büyük elemanını bulan akış şeması.

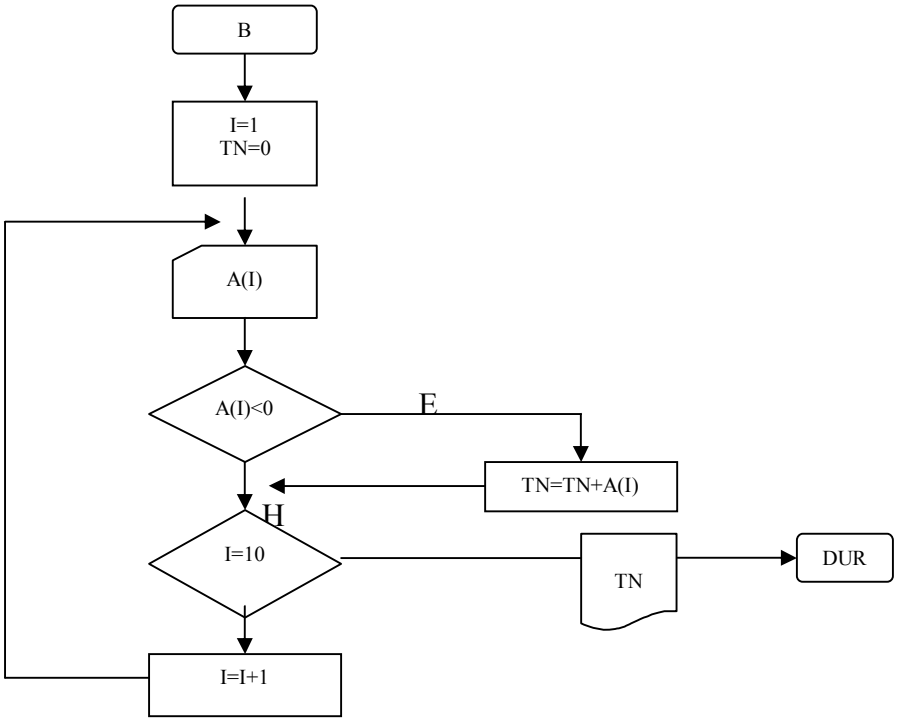
İlk 5 adımda 10 elemanlı sayı dizisinin giriş işlemleri yapılmaktadır. Bunun için bir I indisi kullanılarak bunun değeri 10 oluncaya kadar dizi elemanlarının girişine devam edilmektedir. A6. adımda EB isimli bir değişken tanımlanarak A dizisinin I. Elemanı olan A(I) bu değişkene atanmıştır. Buradaki amaç, dizinin elemanlarını birbirleriyle karşılaştırmak yerine EB olarak adlandırılan değişkenle karşılaştırmaktır. A8. adımda EB ile A dizisinin diğer elemanları karşılaştırılmaktadır ve eğer EB değişkeni bu dizi elemanlarından daha küçük ise EB yerine karşılaştırılan dizi elemanı atanmaktadır. Bu işlem I sayacı 10 oluncaya kadar devam etmektedir. A11. adımda en son elde edilen EB değeri en büyük olarak yazılmaktadır.

Algoritmanın Pascal Dilindeki Karşılığı:

```
Program en_buyuk;
Var
        i,eb:integer;
        a:array[1..10] of integer;
begin
        toplam:=0;
        for İ:=1 to 10 do
                readln(a[i]);
        eb:=a[i];
        for i:=2 to 10 do begin
                if(eb<a[i]) then eb:=a[i];
        end;
        writeln('en büyük sayı=', eb);
end.
```

Örnek 4.6.4. 10 elemanlı bir sayı dizisinde negatif elemanların toplamını bulan algoritma ve akış şemasının oluşturulması.

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



Şekil 4.6.4. 10 elemanlı bir sayı dizisinin negatif elemanların toplamını bulan akış şeması.

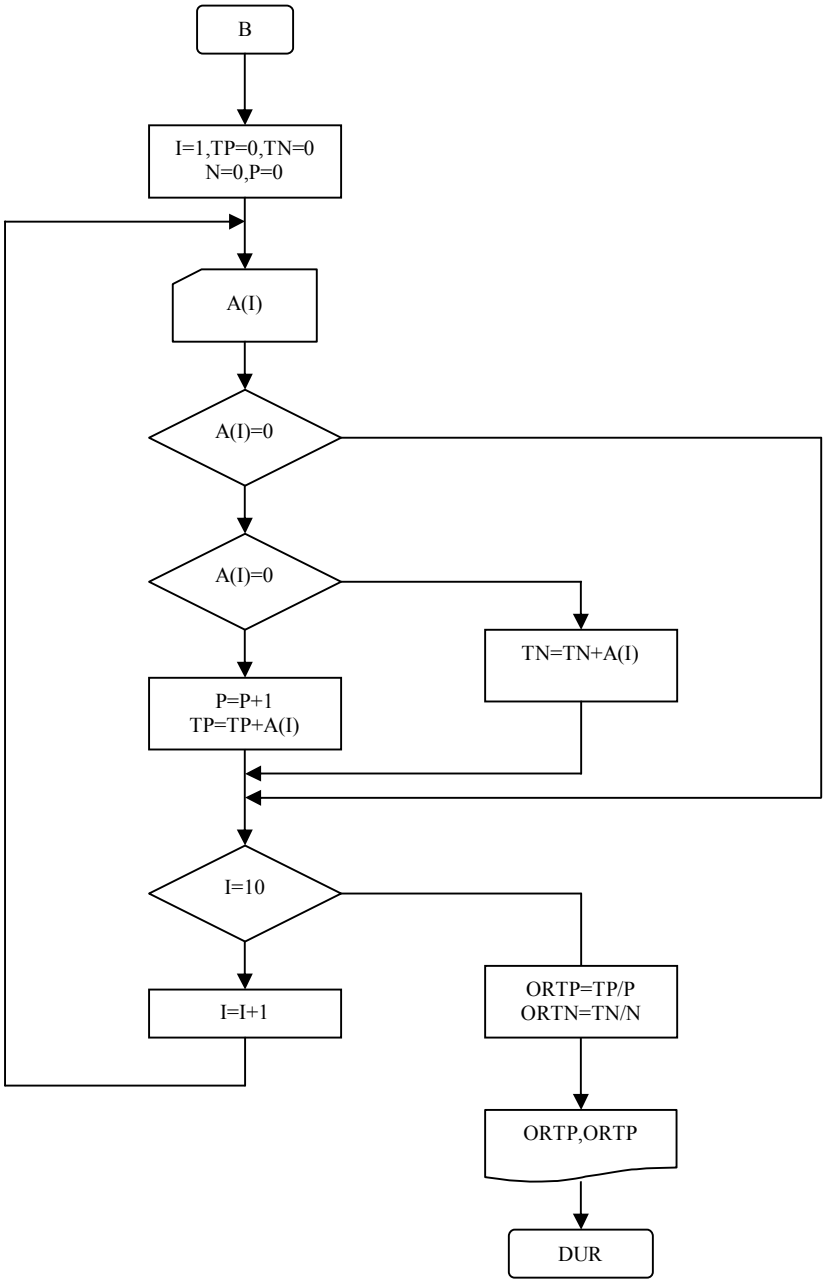
Örnekte A3. adımda girilen dizi elemanının değeri A4. adıma gelindiğinde 0' dan küçük ise bu değer TN olarak tanımlanan ve negatif elemanların toplamını ifade eden değişkene eklenmektedir. Bu işlem I sayacının değeri 10 oluncaya kadar devam eder etmektedir. Elde edilen negatif sayıların toplamı A7. adımda yazdırılmaktadır.

Algoritmanın Pascal Dilindeki Karşılığı:

```
Program negatif_toplamı;
Var
        i,tn,n:integer;
        a:array[1..10] of integer;
begin
        tn:=0; n:=0;
        for i:=1 to 10 do
        begin
                readln(a[i]);
                if (a[i]<0) then
                        tn:=tn+a[i];
        end;
        writeln('negatif toplamı =',tn);
end.
```

Örnek 4.6.5. 10 elemanlı bir sayı dizisinde negatif ve pozitif elemanların ayrı ayrı ortalamasını bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. I=1, TN=0, TP=0, N=0, P=0, ORTP=0, ORTN=0 al,
A3. A(I)' yı gir,
A4. Eğer A(I)=0 ise A7. adıma git,
A5. Eğer A(I)<0 ise TN=TN+A(I), N=N+1 al ve A7. adıma git,
A6. TP=TP+A(I), P=P+1 al
A7. Eğer I=0 ise A9. adıma git,
A8. I=I+1 al ve A3. adıma geri dön,
A9. Eğer P=0 ise A11. adıma git,
A10. ORTP=TP/P
A11. Eğer N=0 ise A13. adıma git,
A12. ORTN=TN/N al,
A13. ORTP ve ORTN' yi yaz,
A14. Dur.



Şekil 4.6.5. 10 elemanlı bir sayı dizisinin negatif ve pozitif elemanların ortalamasını bulan akış şeması.

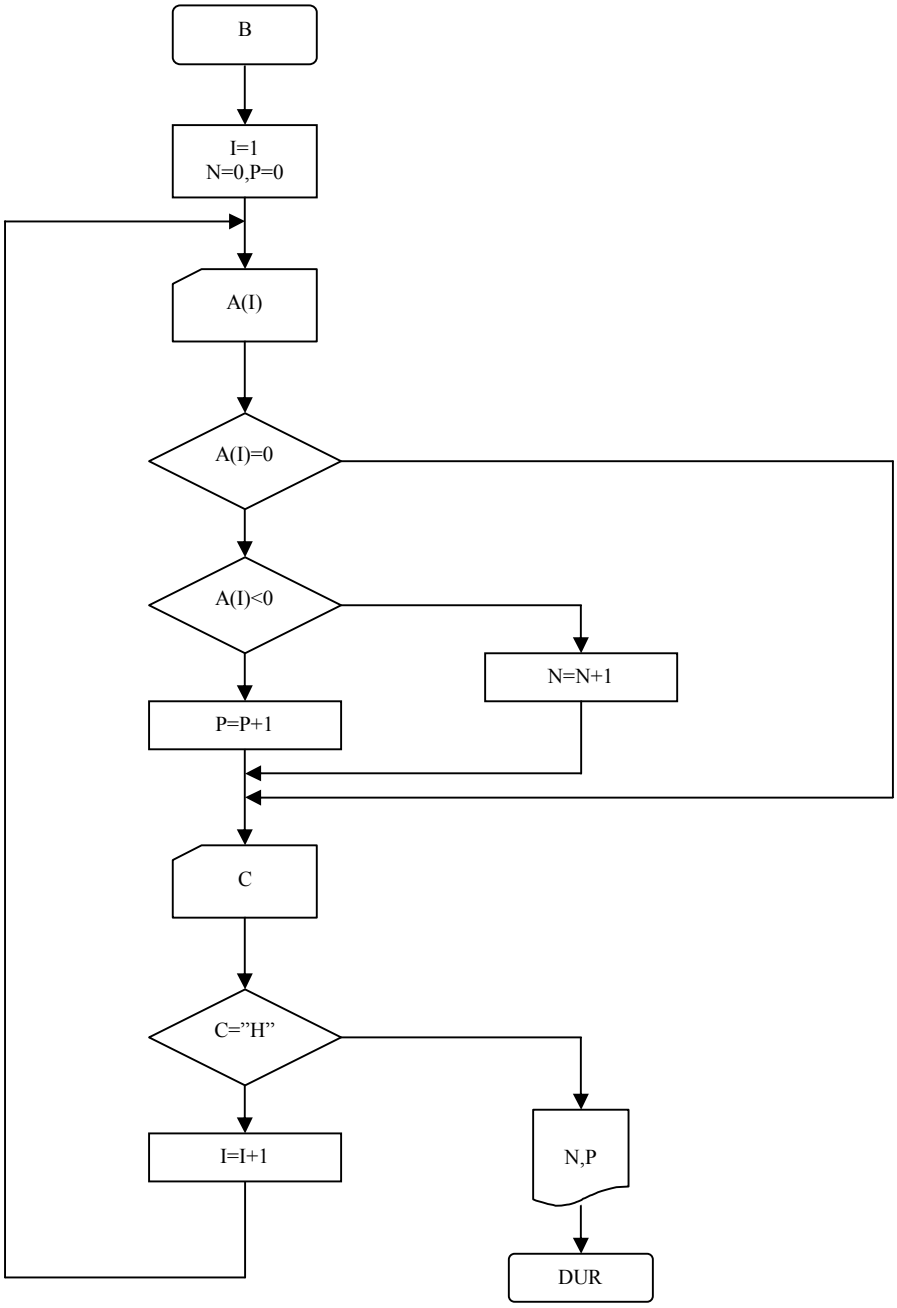
Verilen örnekte negatif sayıların toplamını tanımlamak üzere bir TN değişkeni ve pozitif sayıların toplamını tanımlamak üzere de bir TP değişkeni tanımlanmıştır. A3. adımda girilen dizi elemanı A4. adımda sıfır ile karşılaştırılmaktadır. Buradaki amaç, eğer dizi elemanının değeri 0 ise herhangi bir toplama ilave edilmemesidir. A5. adımda dizi elemanının değerinin 0' dan küçük olup olmadığı araştırılmaktadır. Eğer bu şart sağlanıyorsa dizi elemanın değeri negatif olacağından bu değer negatif toplamı tanımlayan TN değişkenine ilave edilmektedir. Aksi halde diz değeri pozitif olacağından bu değer pozitif toplamı tanımlayan TP değişkenine ilave edilmektedir. Bu işlemler I' nın değeri 10 oluncaya kadar devam etmektedir ve elde edilen TP ve TN değerleri A9. adımda yazdırılarak işleme son verilmektedir.

Algoritmanın Pascal Dilindeki Karşılığı:

```
Program negatif_pozitif_ortalama;
Var
        i,tn,n,tp,p:integer;
        ortp,ortn:real;
        a:=array[1..10] of integer;
begin
        tn:=0; n:=0;
        p:=0; p:=0;
        ortp:=0; ortn:=0;
        for i:=1 to 10 do begin
                readln(a[i]);
                if(a[i]<0) then begin
                        tn:=tn+a[i];
                        n:=n+1;
                end;
                if (a[i]>0) then begin
                        tp:=tp+a[i];
                        p:=p+1;
                end;
        end;
        if(p=0) then ortp:=0
        else
                ortn:=tp/p;
        if (n=0) then ortn:=0
        else
                ortn:=tn/n;
        writeln('negatif ortalama  =',ortn);
        writeln('pozitif ortalama =',ortp);
end.
```

Örnek 4.6.6. İstenildiği kadar elemandan oluşan bir sayı dizisinde negatif ve pozitif elemanların sayısını bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. I=1,N=0,P=0 al,
A3. A(I)' yı gir,
A4. Eğer A(I)=0 ise A7. adıma git,
A5. Eğer A(I)<0 ise N=N+1 al ve A7. adıma git,
A6. P=P+1 al,
A7. C' yi gir{burada C tekrar sayısı girişini kontrol et},
A8. Eğer C="H" ise A10. adıma git,
A9. I=I+1 al ve A3. adıma git,
A10. P' yi ve N' yi yaz,
A11. Dur.



Şekil 4.6.6. İstenildiği kadar elemandan oluşan bir dizide negatif ve pozitif elemanların sayısını bulan akış şeması.

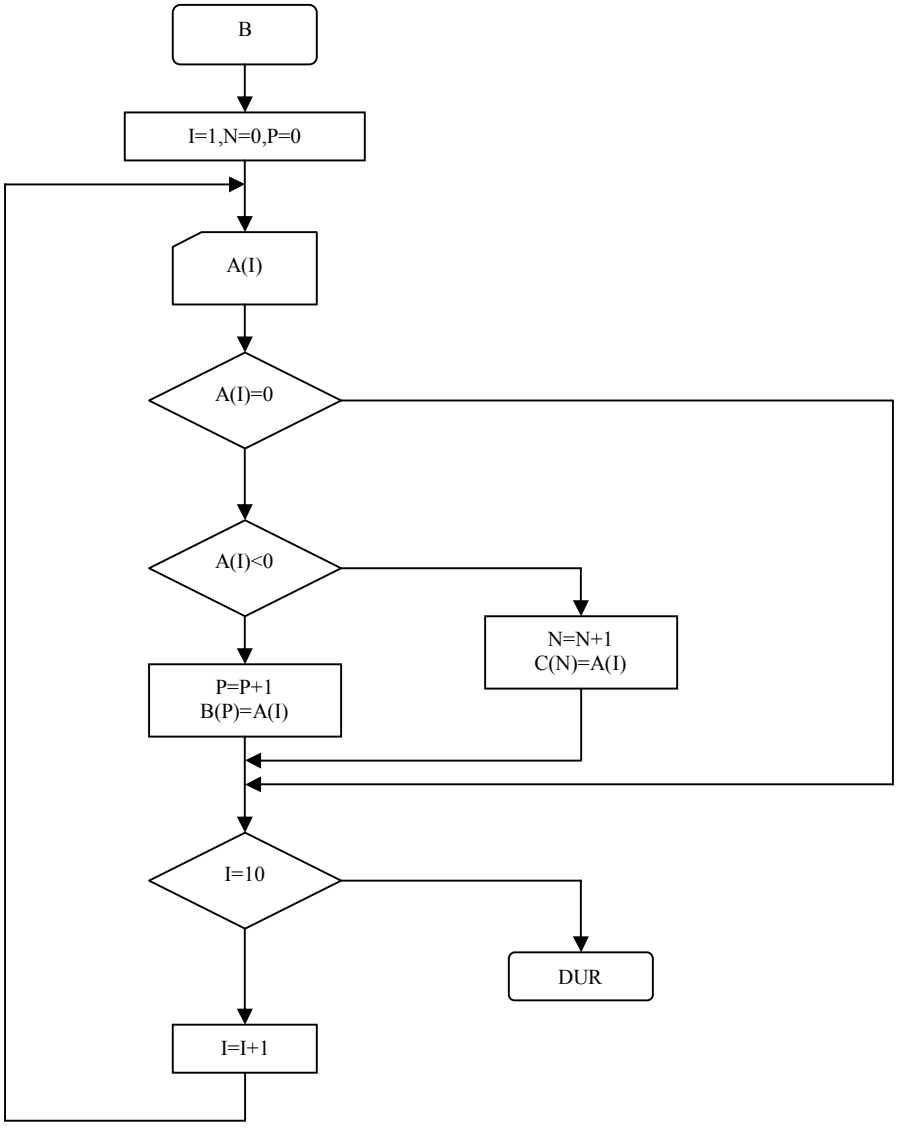
Örnekte P ve N gibi iki değişken tanımlanmıştır. Bunlardan P, pozitif sayıların, N' de negatif sayıların sayısını belirlemek için kullanılmaktadır. A3. adımda girilen dizi elemanı A4. adımda sıfır ile karşılaştırılmaktadır. Eğer dizi elemanının değeri sıfır ise işleme tabi tutulmayarak A7. adıma gönderilmektedir. Bir sonraki adımda dizi elemanının negatif olup olmadığı araştırılmaktadır. Eğer bu sayı negatif ise N değişkeni bir artırılarak A7. adıma gönderilmektedir. Aksi halde bu değer pozitif olacağından A6. adım devreye girerek P değişkeni bir artırılmaktadır. A7. adımda tekrar bir dizi elemanının girilip girilmeyeceği sorgulanmaktadır. Bunun için bir C değişkeni kullanılmaktadır. Girilen C değeri hayır anlamına gelen "H " ise A10. adıma gidilerek P ve N değerleri yazdırılıp işlemlere son verilmektedir. Aksi halde I sayacı bir rtırılarak A3. adıma gönderilerek yeniden bir dizi elemanının girilmesi sağlanmaktadır.

Algoritmanın Pascal Dilindeki Karşılığı:

```
Program negatif_pozitif;
Var
        i,n,p,tp,tn:integer;
        devam:char;
        a:array[1..100] of integer;
begin
        n:=0;
        p:=0;
        i:=0;
        tp:=0;
        tn:=0;
        devam:='e';
        while (devam<>'h') do
        begin
                readln(a[i]);
                if(a[i]<0) then
                        tn:=tn+a[i];
                if(a[i]>0) then
                        tp:=tp+a[i]; i:=i+1;
                write('tekrar sayısını girilecek mi...');
                readln(devam);
        end;
        writeln('negatif toplamı=',tn);
        writeln('pozitif toplamı=',tp);
end.
```

Örnek 4.6.7. 10 elemandan oluşan bir A dizisinde negatif elemanların ayrı bir diziye, pozitif elemanların ayrı bir diziye yüklenmesini sağlayan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. I=1,N=0,P=0 al,
A3. A(I)' yı gir,
A4. Eğer A(I)=0 ise A8. adıma git,
A5. Eğer A(I)<0 ise A7. adıma git,
A6. P=P+1, B(P)=A(I) al ve A8. adıma git,
A7. N=N+1, C(N)=A(I) al,
A8. Eğer I=10 ise A10. adıma git,
A9. I=I+1 al ve A3. adıma git,
A10. Dur.



Şekil 4.6.7. 10 elemanlı bir dizide negatif ve pozitif elemanların ayrı dizilere yüklryen akış şeması

Verilen örnekte A2. adımda A dizisindeki negatif elemanları belirlemek üzere bir N değişkeni ve pozitif elemanları belirlemek üzere bir P değişkeni tanımlanmıştır. A3. adımda girilen dizi elemanı A4. adımda sıfır ile karşılaştırılmaktadır. Eğer dizi elemanının değeri sıfır ise dikkate alınmayarak A8. adıma gönderilmektedir. Aksi halde A5. adım devreye girerek girilen sayının negatif olup olmadığı sorgulanmaktadır. Eğer sayı negatif ise A7. adıma geçilerek N değişkeni bir artırılıp C dizisinin N elemanına A dizisinin I. elemanı atanmaktadır. Aksi halde dizi elemanı pozitif olacağından A6. adım dikkate alınarak P değişkeni bir artırılıp P. Elemanı yerine A dizisinin I. elemanı atanmaktadır. Bu işlemler I sayacının değeri 10 oluncaya kadar etmektedir.

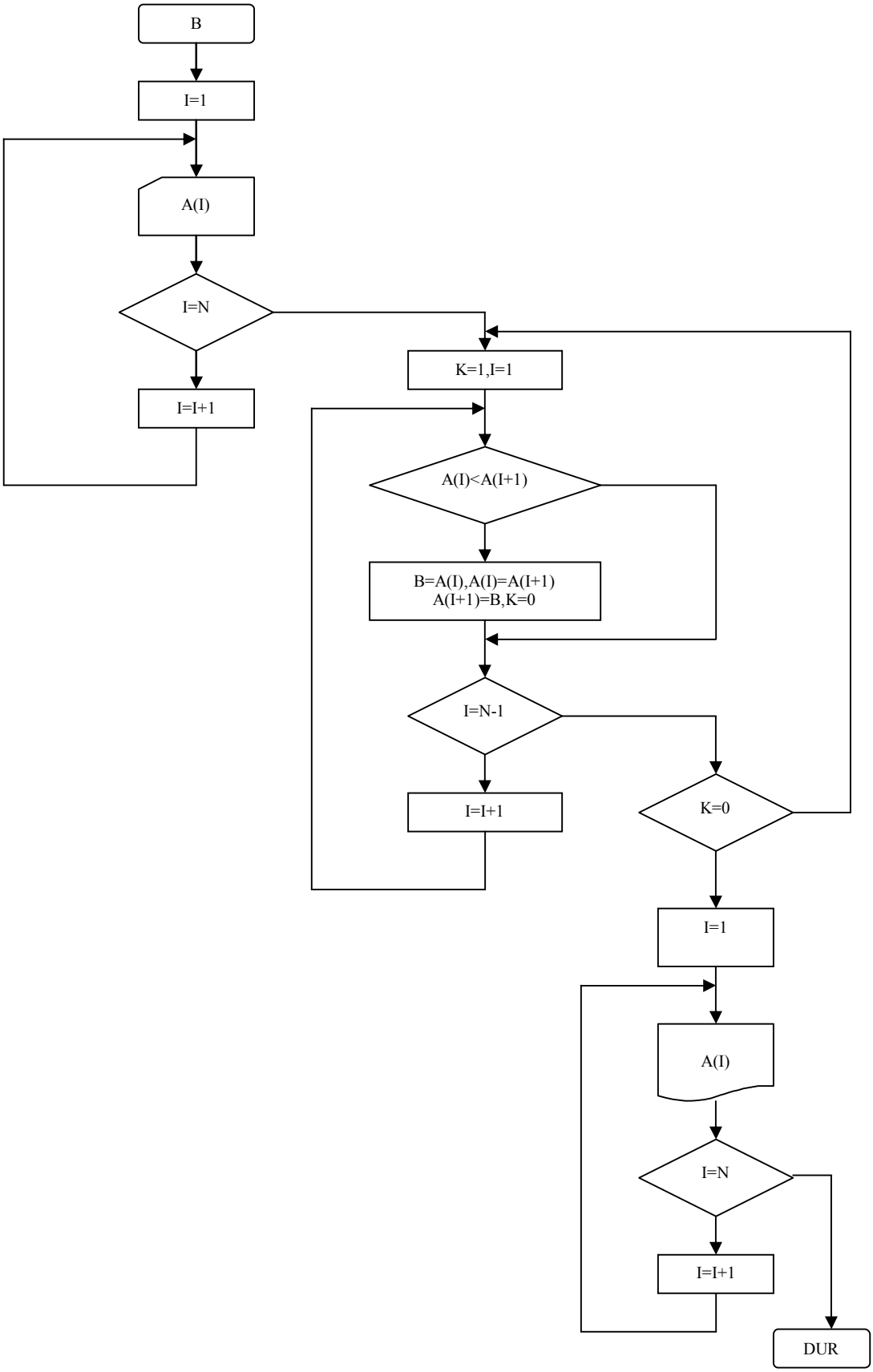
Algoritmanın Pascal Dilindeki Karşılığı:

```
Program dizi_aktarımı;
Var
        i,n,p:integer;
        a,b,c:array[1..10] of integer;
```

```
begin
        n:=0;
        p:=0;
        for i:=1 to 10 do
        begin
                readln(a[i]);
                if (a[i]<0)  then
                begin
                        n:=n+1;
                        b[n]:=a[i];
                end;
                if (a[i]>0)  then
                begin
                        p:=p+1;
                        c[p]:=a[i];
                end;
                for i:=1 to n do
                        writeln(b[i]);
                for i:=1 to p do
                        writeln(c[i]);
end.
```

Örnek 4.6.8. N elemandan oluşan bir A sayı dizisinin küçükten büyüğe doğru sıralamasını yapan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. I=1 al,
A3. A(I)' yı gir,
A4. Eğer I=N ise A6. adıma git,
A5. I=I+1 al ve A3. adıma git,
A6. K=1 al, I=1 al,
A7. Eğer A(I)<A(I+1) ise A9. adıma git,
A8. B=A(I),A(I)=A(I+1), A(I+1)=B, K=0 al,
A9. Eğer I=N-1 A11. adıma git,
A10. I=I+1 al ve A7. adıma geri dön,
A11. Eğer K=0 ise A6. adıma geri dön,
A12. I=1 al,
A13. A(I)' yı yaz,
A14. Eğer I=N ise DUR,
A15. I=I+1 al ve A14.adıma geri dön.



Şekil 4.6.8. N elemanlı bir sayı dizisinin küçükten büyüğe doğru sıralanmasını bulan akış şeması.

Örnekte N elemanlı bir sayı dizisinin küçükten büyüğe doğru sıralaması yapılmaktadır. Bunun için ilk aşamada dizi elemanlarının girişi gerçekleştirilmektedir. A6. adımda tanımlanan K değişkeni dizinin sıralanıp sıralanmadığını kontrol etmek için tanımlanmıştır. A7. adımda I indisine ilk değeri olan 1 verilmiştir. A8. adımda A dizisinin I. elemanı ile I+1 elemanı karşılaştırılmaktadır. Eğer I. eleman I+1. elemandan küçük ise A10. adıma gidilmekte, aksi halde ise bu dizi elemanları yer değiştirilmektedir. Bu elemanların yerlerinin değiştirilebilmesi için bir B değişkeni aracı olarak kullanılmaktadır. Aynı adımda dikkat edildiğinde K değişkeninin değeri 0 olarak tanımlanmaktadır. Bunu anlamı dizi elemanları arasında yer değişikliğinin olduğunu belirtmektir. Bu durumda dizi henüz sıralanmamış anlamındadır. Bu karşılaştırma işlemi A10. adımda sorgulanan I indisinin alacağı son değer olan N-1 değerine kadar tekrarlanmaktadır. I indisinin değeri N-1 olunca A12. adım devreye girerek K değişkeninin değeri sorgulanmaktadır. Eğer K değişkeninin değeri 0 ise dizi elemanları arasında değişiklik yapıldığından henüz sıralama gerçekleşmemiş anlamındadır. Dolayısıyla A6. adıma geri dönülerek aynı işlemlerin tekrar yapılması sağlanmaktadır. Aksi halde, yani K değişkeninin değeri 1 ise A9. adım hiç kullanılmamış olacağından dizi elemanları arasında bir yer değişikliği söz konusu olamayacaktır. Bu durum ise dizinin sıralandığınıeddc3 göstermektedir. Bu aşamadan sonra dizi elemanlarının yazılması yapılmaktadır.

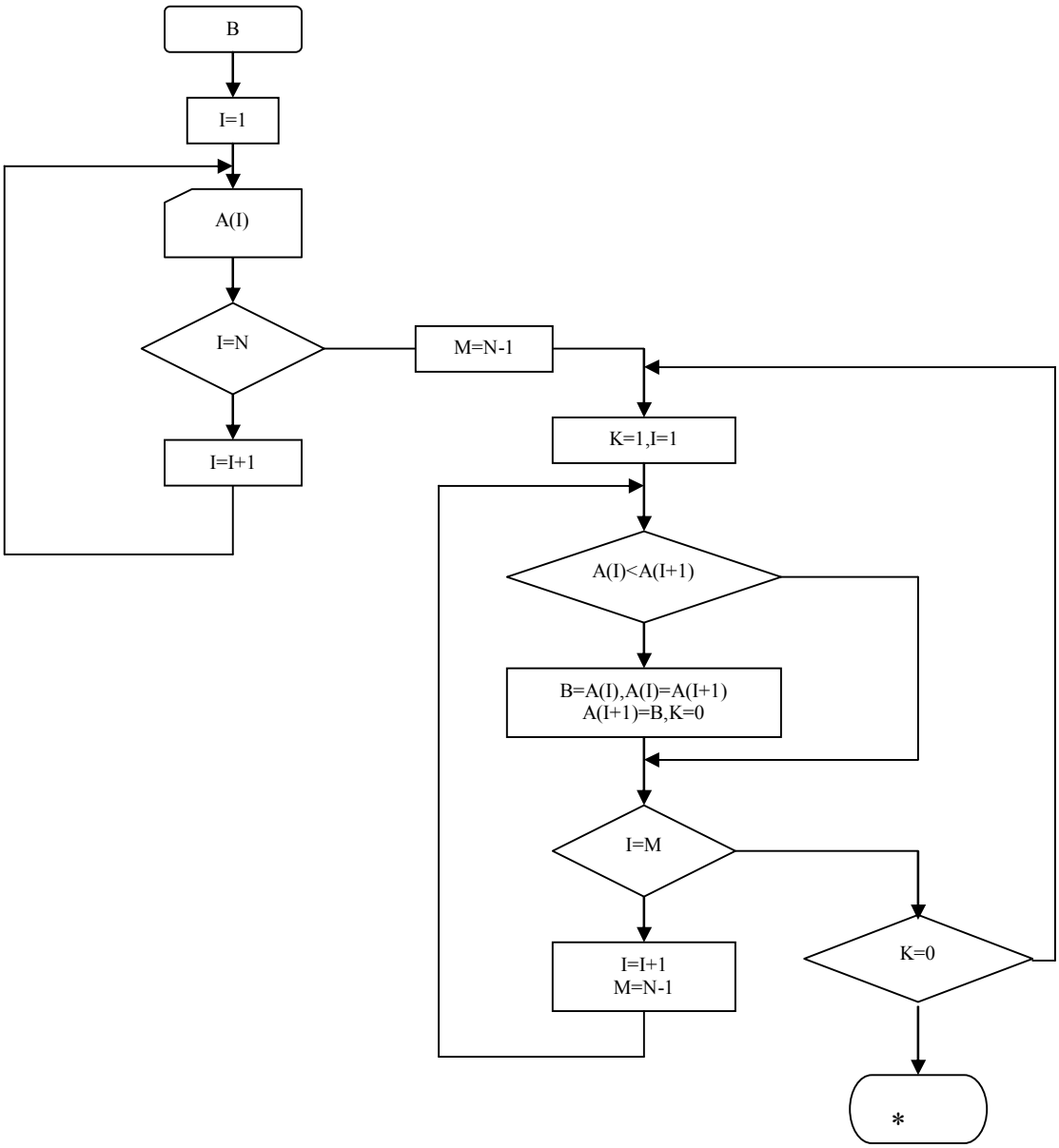
Algoritmanın Pascal Dilindeki Karşılığı:

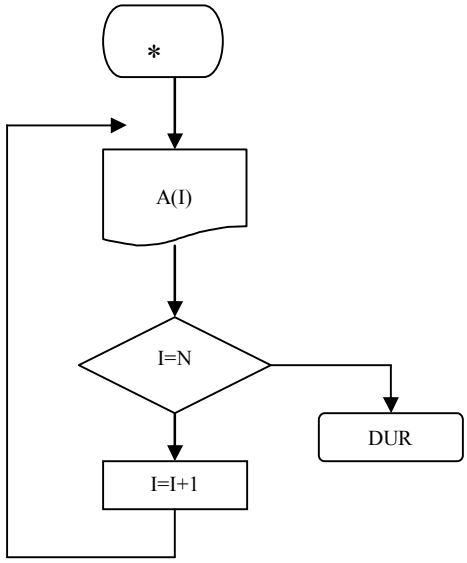
```
Program siralama;
Var
        i,b,k,n:integer;
        a:array[1..100] of integer;
begin
        write('n sayısını giriniz: ');    readln(n);
        for i:=1 to n do readln(a[i]);
        k:=1;
        while(k<>0) do begin
                k:=0;
                for i:=1 to n-1 do begin
                        if (a[i]>a[i+1]) then begin
                                b:=a[i];  a[i]:=a[i+1];
                                a[i+1]:=b; k:=1;
                        end;
                end;
        end;
        for i:=1  to n do
                writeln(a[i]);
end.
```

2. Yöntem :

A1. Başla,
A2. I=1 al,
A3. A(I)' yı gir,
A4. Eğer I=N ise A6. adıma git,

A5. I=I+1 al ve A3. adıma git,
A6. M=N-1 al,
A7. K=1 al,
A8. I=1 al,
A9. Eğer A(I)<A(I+1) ise A10. adıma git,
A10. B=A(I), A(I)=A(I+1), A(I+1)=B, K=0 al,
A11. Eğer I=M ise A12. adıma git,
A12. I=I+1 ve M=M-I al ve A8. adıma geri dön,
A13. Eğer K=0 ise A6. adıma geri dön,
A14. I=1 al,
A15. A(I)' yı yaz,
A16. Eğer I=N ise A17. adıma git,
A17. I=I+1 al ve A14.adıma geri dön.
A18. Dur.





Şekil 4.6.8.2. N elemanlı bir sayı dizisinin küçükten büyüğe doğru sıralanmasını bulan akış şeması.

Bu yöntemde diğer yöntemden farklı olarak A6. adımda bir M değişkeninin tanımlanması yapılmıştır. Bu değişkene ilk değer olarak N-1 atanmıştır. Bunun anlamı ilk iterasyonda dizi elemanlarının karşılaştırılması N-1. elemana kadar yapılacağından A11. adımda sorgulama bu değere karşılık gelen M değerine kadar yapılmaktadır. Dizinin sıralanması aşamasındaki her iterasyon sonunda dizinin en büyük değeri en sona atanmaktadır. Bir anlamda dizi tersten sıralanmaktadır. Bu nedenle her iterasyonda N-1 elemanının karşılaştırılmasının yapılmasına gerek yoktur. İterasyon sonunda dizi sıralanmamış ise M değeri bir azaltılarak işleme devam etmek daha mantıklı olacağından algoritmanın A12. adımında bu işlem dikkate alınmaktadır.

Algoritmanın Pascal Dilindeki Karşılığı:

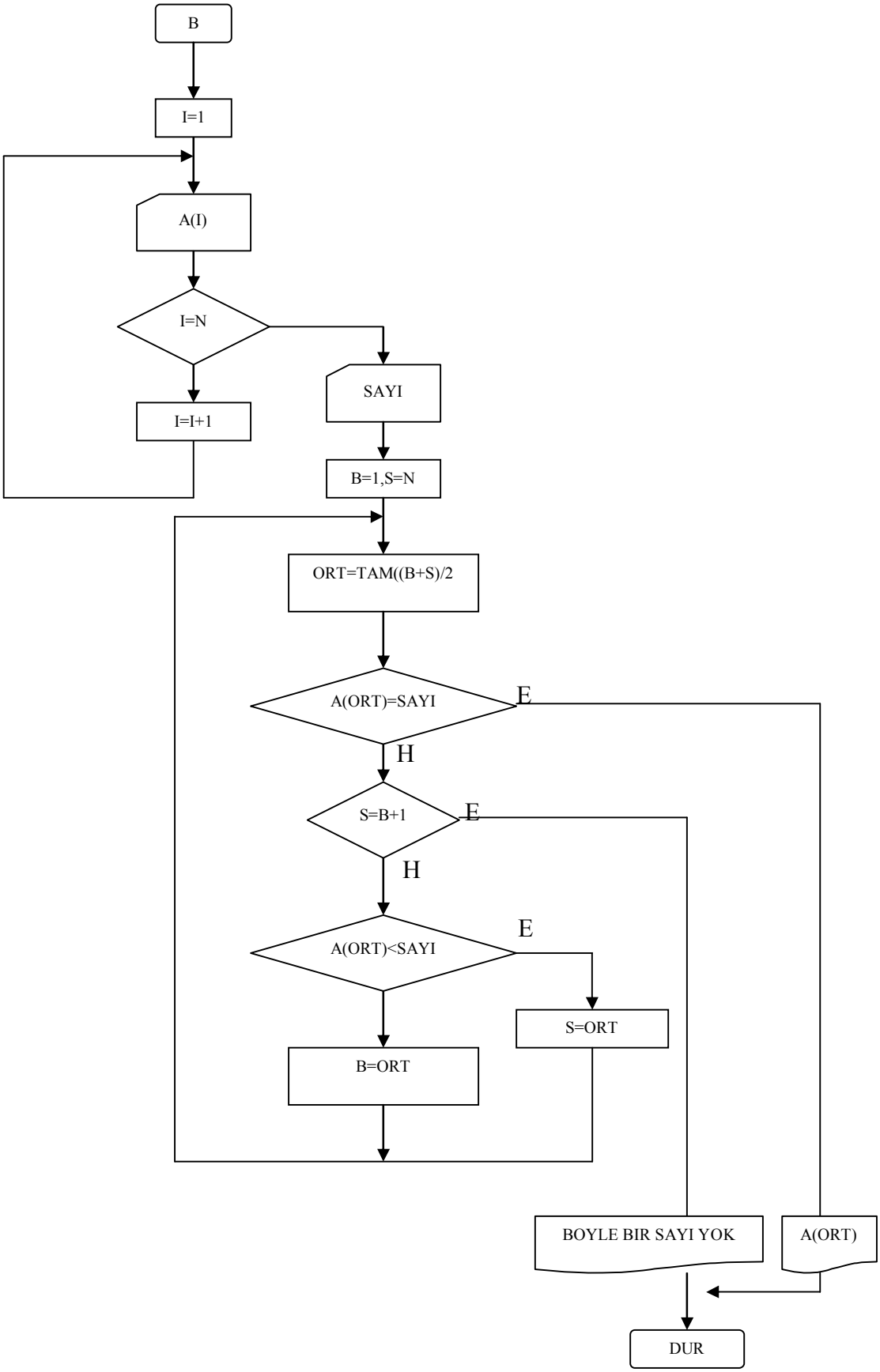
```
Program siralama;
Var
        i,b,k,n,m:integer;
        a:array[1..100] of integer;
begin
        write('n sayısını giriniz: ');
        readln(n);
        for i:=1 to n do
                readln(a[i]);
        k:=1;
        m:=1;
        while(k<>0) do
        begin
                k:=0;
                for i:=1 to m do
                begin
                        if (a[i]>a[i+1]) then
```

```
                    begin
                        b:=a[i];
                        a[i]:=a[i+1];
                        a[i+1]:=b;
                        k:=1;
                    end;
                end;
                m:=m-1;
        end;
        for i:=1  to n do
                writeln(a[i]);
end.
```

Örnek 4.6.9. Küçükten büyüğe doğru sıralı bir şekilde girilen n elemanlı bir sayı dizisinde istenilen bir sayıya yarılama işlemleriyle ulaşmayı sağlayan algoritma ve akıs semasının olusturulması.

Yarılama işlemi:Sıralı bir dizide istenilen bir sayıya ulaşmak için dizinin yarılanarak ortasındaki elemanın sorgulanması metodu olarak kabul edilir. Yarılama işlemleri ortadaki elemanın sorgulanmasıyla dizinin daraltılması gerçekleştirilir ve bu işlemlere istenilen elemana ulaşılıncaya kadar devam eder.

A1. Başla,
A2. I=1 al,
A3. A(I)' yı gir,
A4. Eğer I=N ise A6. adıma git,
A5. I=I+1 al ve A3. adıma git,
A6. SAYI' yı gir, {SAYI değişkeni aranılan sayıyı belirtmektedir.}
A7. B=1, S=N+1 al,
A8. ORT=TAM((B+S)/2) al,
A9. Eğer A(ORT)=SAYI ise A14. adıma git,
A10. S=B+1 ise A15. adıma git,
A11. Eğer A(ORT)<SAYI ise A13. adıma git,
A12. B=ORT al ve A8. adıma git,
A13. S=ORT al ve A8. adıma git,
A14. A(ORT) değerini yaz ve A16. adıma git,
A15. "Böyle bir sayı yok" yaz,
A16. Dur.



Şekil 4.6.9. Yarılama metodu ile sıralı bir dizide istenilen elemana ulaşmayı sağlayan akış şeması.

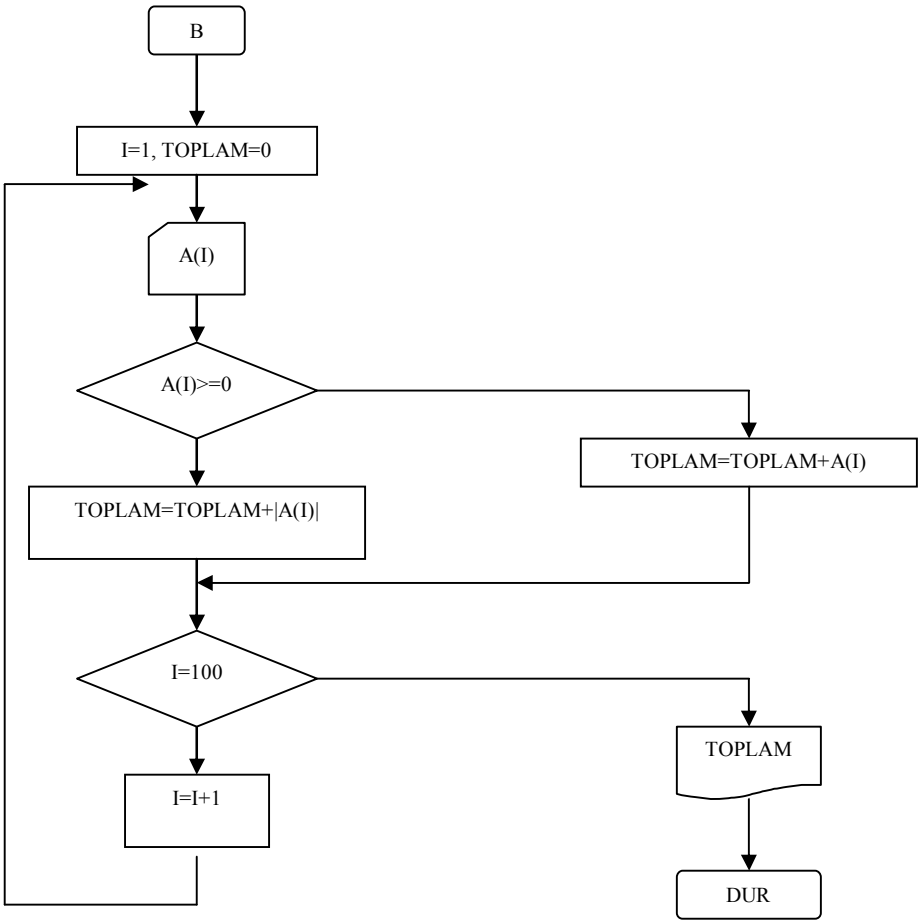
Örnekte ilk beş adım sıralı dizinin girilmesini gerçekleştirmektedir. A6. adımda girilen SAYI değişkeni ile aranan sayı tanımlanmaktadır. A7. adımda tanımlanan B ve S değişkenleri dizinin başlama ve bitiş sınırını tanımlamaktadır. İlk değer olarak B' ye 1 ve S' ye de N değeri verilmiştir. A8. adımda ORT olarak tanımlanan S ve B değerlerinin toplamlarının yarısının tam kısmı atanmıştır. Buradaki amaç, diziyi yarılayarak buradan elde edilen değere karşılık gelen dizi değişkenini sorgulamaktır. A9. adımda burada elde edilen değere karşılık gelen dizi elemanı ile SAYI değeri karşılaştırılmaktadır. Eğer eşitlik söz konusu ise istenilen sayıya ulaşılmıştır. Aksi halde aranılan sayı ORT değişkenine karşılık gelen dizi elemanından ya küçük ya da büyük olacaktır. Bu durumlardan küçük olma koşulu A11. adımda sorgulanmaktadır. Eğer buradaki şart sağlanıyorsa üst sınır değeri olan S, ORT değerine çekilerek dizi sağdan daraltılmaktadır. Bu işlemler sonucunda tekrar A8. adıma dönülerek yeni bir ORT değeri tanımlanarak aynı işlemlere devam edilmektedir. Bu işlemler ORT değerine karşılık gelen dizi elemanının aranılan sayıya eşit olmasına kadar devam edecektir. Ancak, eğer dizi içerisinde aranan eleman yoksa A15. adımın devreye girmesi gerekmektedir. Bunun için, A10. adımda bir sorgulama yapılmaktadır. Üst sınır değeri olan S, alt sınırı değeri olan B' den 1 fazla olduğunda ORT olarak adlandırılan değer hep aynı kalacağından istenilen sayı bulunamayacaktır ve dolayısıyla dizi içerisinde böyle bir sayı mevcut olmadığından A15. adıma gidilerek böyle bir sayı yok mesajı verilecektir.

Algoritmanın Pascal Dilindeki Karşılığı:

```
Program sayi_bulma;
Var
        i,n,sayi,ort,b,s,t:integer;
        a:array[1..100] of integer;
begin
        write('n sayısını giriniz: ');
        readln(n);
        for i:=1 to n do
                readln(a[i]);
        b:=1;
        s:=n+1;
        write ('sayıyı giriniz...');
        readln(sayi);
        t:=1;
        while(t<>0) do begin
                ort:=trunc((b+s)/2);
                if (a[ort]=sayi]) then begin
                        writeln('aranan sayı  :',ort);
                        t:=0;
                end;
                if(s=b+1) then begin
                        writeln('böyle sayı yok...');
                end;
                if(a[ort]<sayi) then s:=ort
                else
                b:=ort;
        end;
end.
```

Örnek 4.6.10. Rasgele 100 elemandan oluşan bir A dizisinde negatif olan elemanların mutlak değerlerini alarak dizi elemanlarının toplamını bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. I=1,TOPLAM=0 al,
A3. A(I)' yı gir,
A4. Eğer A(I)>=0 ise A6. adıma git,
A5. TOPLAM=TOPLAM+|a(I)| al ve A7. adıma git,
A6. TOPLAM=TOPLAM+A(I) al,
A7. Eğer I=100 ise A9. adıma git,
A8. I=I+1 al ve A3. adıma geri dön,
A9. TOPLAM değerini yaz,
A10. Dur.



Şekil 4.6.10. Yüz elemanlı bir dizide negatif elemanların mutlak değerleri ile birlikte dizi elemanlarını toplayan akış şeması

Örnekte A3. adımda girilen A dizisinin eleman A4. adımda sorgulanmaktadır. Eğer bu dizi elemanı sıfırdan büyük ya da eşit ise A6. adıma gönderilerek TOPLAM değişkenine doğrudan ilave edilmektedir. Aksi halde A5. adım devreye girerek dizi elemanın mutlak değeri TOPLAM değişkenine ilave edilmektedir. Bu işlemler A7.

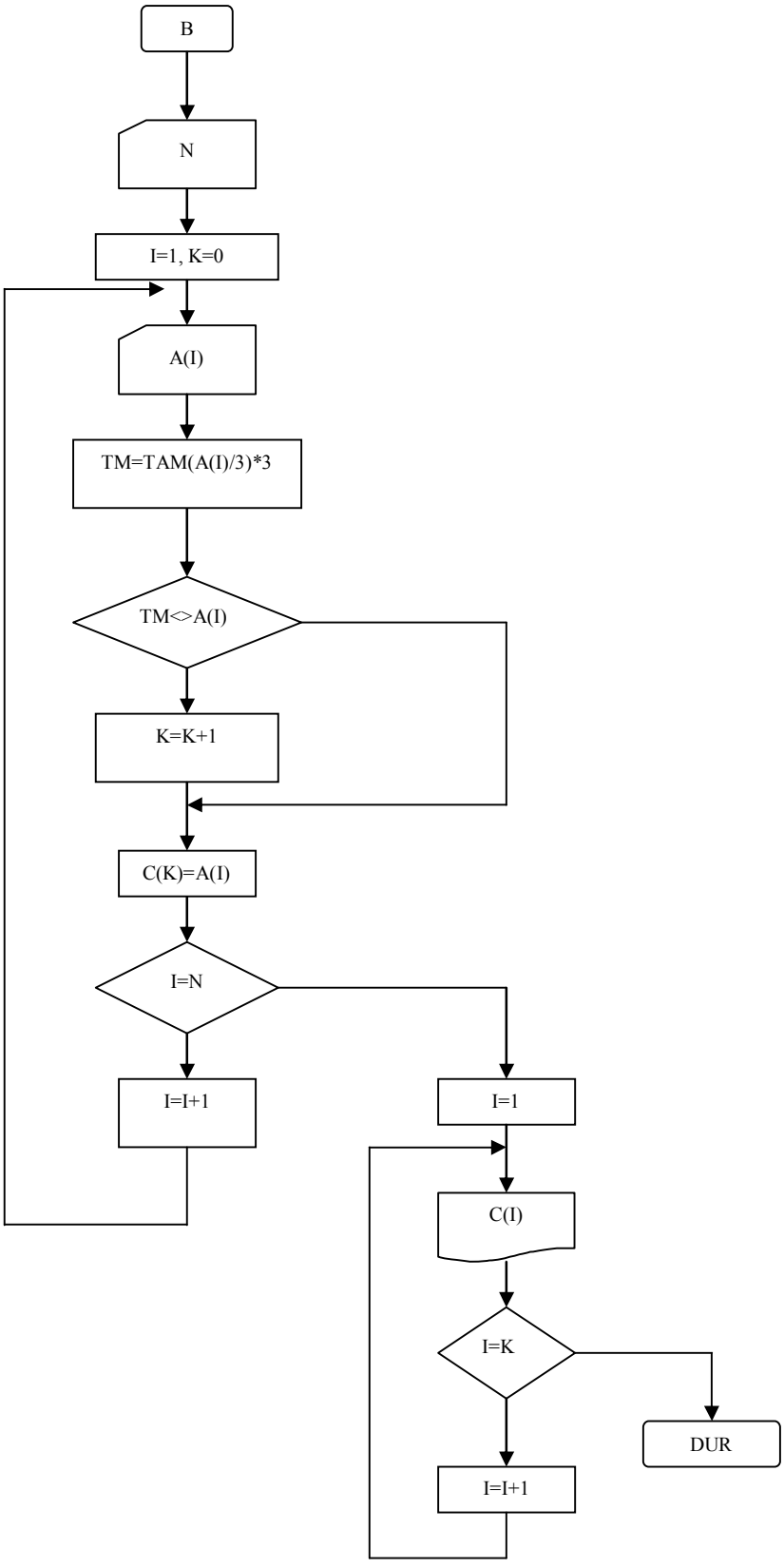
adımda verilen I sayacının değeri 100 oluncaya kadar devam etmektedir ve istenilen şart gerçekleşince A9. adımda elde edilen TOPLAM ifadesi yazdırılarak işleme son verilmektedir.

Algoritmanın Pascal Dilindeki Karşılığı:

```
Program toplama;
Var
        i,toplam:integer;
begin
        toplam:=0;
        for i:=1 to 10 do
        begin
                readlan(a[i]);
                if(a[i]<0) then
                        toplam:=toplam+abs(a[i])
                else
                        toplam:=toplam+a[i];
        end;
        writeln('toplam =',toplam);
end.
```

Örnek 4.6.11. Rasgele N elemandan oluşan bir tam sayı dizisinde 3 ile tam bölünebilen sayıların başka bir diziye yüklenmesini sağlayan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. N' yi gir,
A3. I=1, K=0 al,
A4. A(I)' yı gir,
A5. TM=TAM(A(I)/3)*3 al,
A6. Eğer TM<>A(I) ise A9. adıma git,
A7. K=K+1 al,
A8. C(K)=A(I) al,
A9. Eğer I=N ise A11. adıma git,
A10. I=I+1 al ve A4. adıma geri dön,
A11. I=1 al,
A12. C(I)' yı yaz,
A13. Eğer I=K ise A15. adıma git,
A14. I=I+1 al ve A12. adıma geri dön,
A15. Dur.



Şekil 4.6.11. Bir tam sayı dizisinde 3 ile tam bölünebilen sayıları başka bir diziye yükleyen akış şeması.

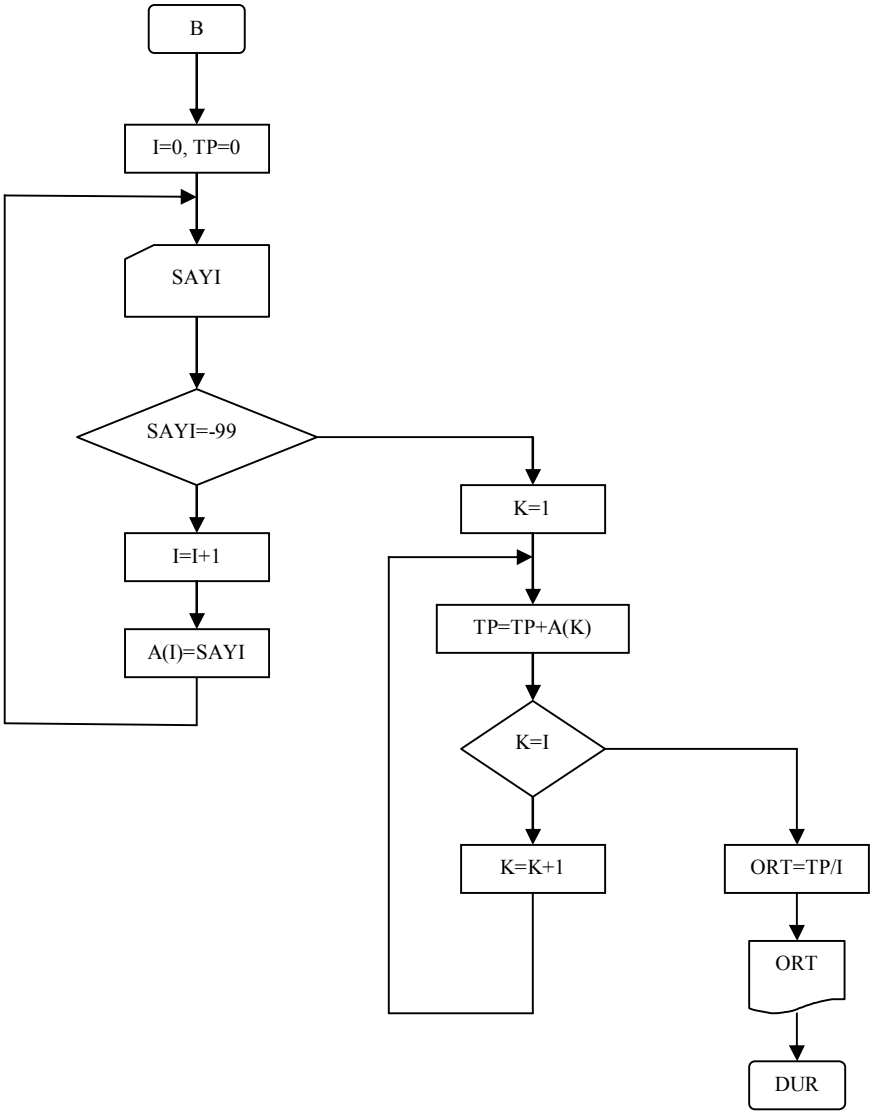
Örnekte A4. adımda girilen dizi elemanı A5. adımda 3 ile bölünüp tam kısmı alınarak tekrar 3 ile çarpılmaktadır. Elde edilen bu ifade TM değişkenine aktarılmaktadır. A6. adımda, elde edilen bu değer A dizisinin ilgili elemanı ile karşılaştırılmaktadır. Eğer dizi elemanı ile TM değişkeni aynı değere sahip ise ilgili eleman 3 ile tam bölünüyor anlamındadır. Bu durumda A7. adımda K değişkeni I artırılarak A8. adımda C(K) yerine A dizisinin I. elemanı atanmaktadır ve A9. adımdan itibaren işlemlere devam edilmektedir. Aksi halde de yine A9. adımdan itibaren işlemlere devam edilmektedir. I indisinin değeri N oluncaya kadar A4. adımdan itibaren aynı işlemler devam etmektedir. I sayacının değeri N olunca A11. adımdan itibaren C dizisinin elemanları yazdırılarak işlemlere son verilmektedir.

Algoritmanın Pascal Dilindeki Karşılığı:

```
Var
        i,k,n,t:integer;
        a,c:array[1..10] of integer;
begin
        k:=0;
        for i:=0 to 9 do begin
                readlan(a[i]);
                t:=trunc(a[i]/3)*3;
                        if(a[i]=t) then  begin
                                k:=k+1;c[k]:=a[i];
                        end;
        end;
        for i:=0 to 9 do
                writeln(c[i]);
end.
```

Örnek 4.6.12. İstenilen değer verilinceye kadar girilen sayıları diziye yükleyerek aritmetik ortalamasını bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. I=0, TP=0 al,
A3. SAYI' yı gir,
A4. Eğer SAYI=-99 ise A7. adıma git,
A5. I=I+1 al,
A6. A(I)=SAYI al ve A3. adıma geri dön,
A7. K=1 al,
A8. TP=TP+A(K) al,
A9. Eğer K=1 ise A11. adıma git,
A10. K=K+1 al ve A8. adıma geri dön,
A11. ORT=TP/I al,
A12. ORT değerini yaz,
A13. Dur.



Şekil 4.6.12. İstenilen elemana ulaşıncaya kadar girilen sayıları diziye yükleyerek aritmetik ortalamalarını bulan akış şeması.

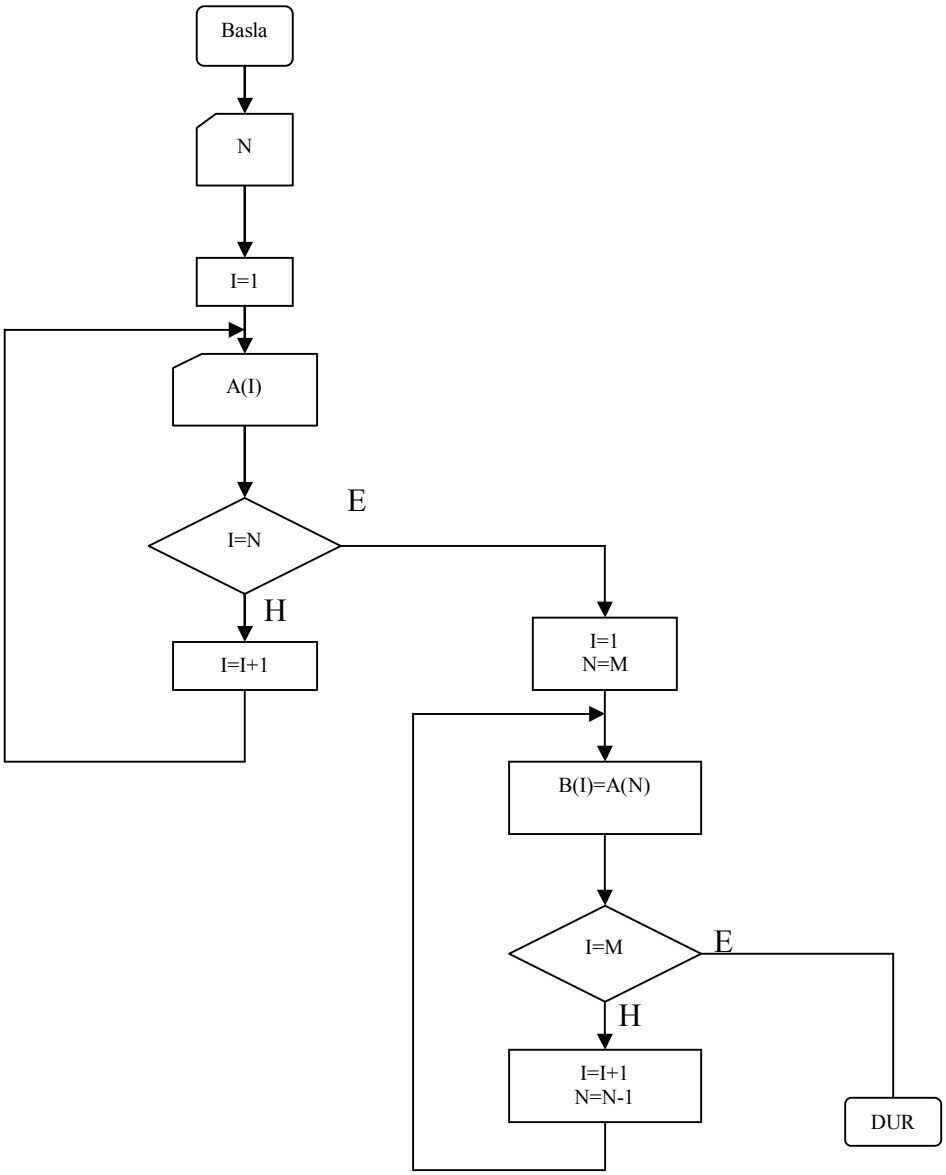
Verilen örnekte girilen SAYI değişkeni –99 oluncaya kadar dizi değişkenlerine atanması gerçekleşmektedir. A3. adımda girilen SAYI değişkeni A4. adımda sorgulanmaktadır. Eğer –99 değerini almamış ise I sayacı 1 artırılarak A dizisinin I. elemanı yerine SAYI değişkeni atanmaktadır. "SAYI=A(I)" ve adıma dönülerek SAYI girişi tekrar istenmektedir. Eğer SAYI değeri –99 olmuş ise A7. adıma gidilerek dizi elemanlarının toplamı gerçekleştirilmektedir. Bu işlem K değişkeni ile kontrol edilerek K değeri I oluncaya kadar dizi elemanlarının toplamı TP değişkenine ilave edilmektedir. Elde edilen toplam değeri olan TP değişkeni A11. adımda I değerine bölünerek ORT olarak adlandırılan ortalama dizi değeri hesaplanıp yazdırılarak işleme son verilmektedir.

Algoritmanın Pascal Dilindeki Karşılığı:

```
Var
        i,tp,sayi,k:integer;
        a:array[1..10] of integer;
        ort:real;
begin
        sayi:=0;
        i:=-1;
        tp:=0;
        while(sayi<>-99) do
        begin
                write('sayıyı giriniz..');
                readln(sayi);
                i:=i+1;
                a[i]:=sayi;
                tp:=tp+a[i];
        end;
        ort:=tp/i;
        writeln('ortalama  =',ort);
end.
```

Örnek 4.6.13. N elemandan oluşan ve küçükten büyüğe doğru sıralanmış bir sayı dizisinin büyüten küçüğe doğru sıralamasını değiştiren algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. N'yi gir,
A3. I=1 al,
A4. A(I)' yı gir,
A5. Eğer I=N ise A7. adıma git,
A6. I=I+1 al ve A4. adıma geri dön,
A7. I=1,M=N al,
A8. B(I)=A(N) al,
A9. Eğer I=M ise A11. adıma git,
A10. I=I+1, N=N-1 al ve A8. adıma geri dön,
A11. Dur.



Şekil 4.6.13. Küçükten büyüğe doğru sıralanmış bir diziyi, büyükten küçüğe doğru değiştiren akış şeması.

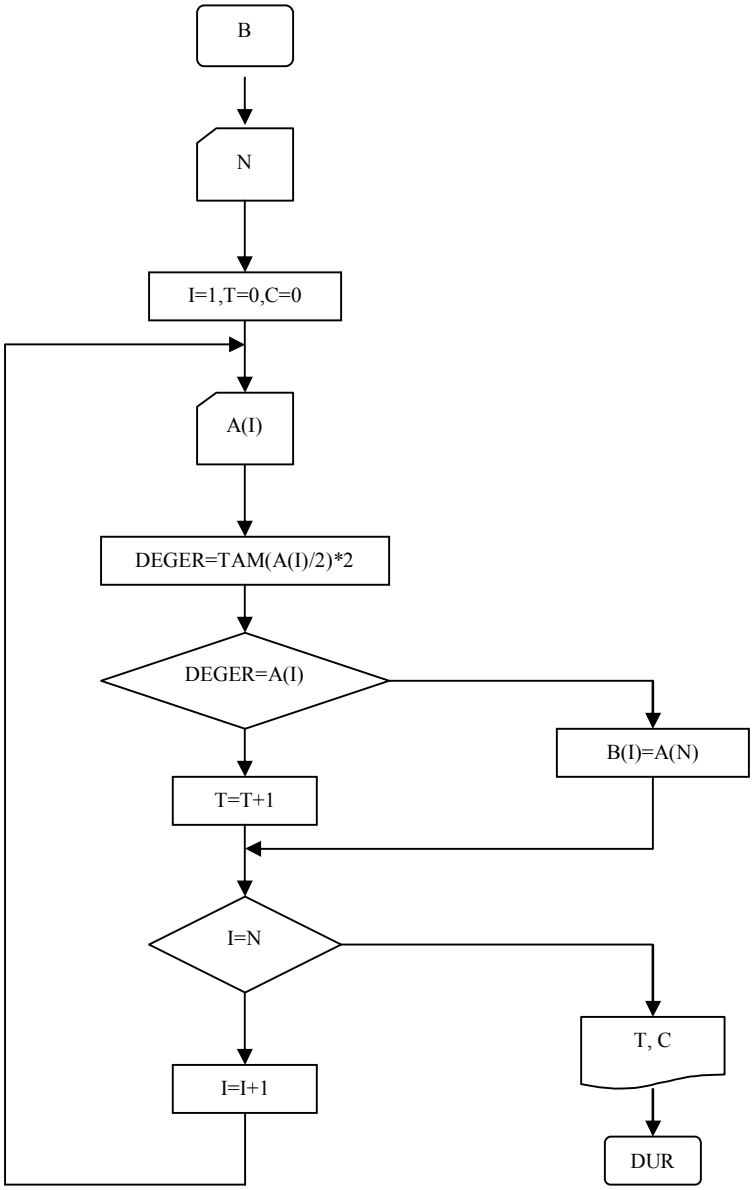
Örnekte ilk 6 adımda küçükten büyüğe doğru sıralı dizi elemanlarının girişi gerçekleştirilmektedir. A7. adımdan itibaren dizinin büyükten küçüğe doğru çevrilmesi gerçekleşmektedir. Dizi elemanları küçükten büyüğe doğru sıralı olduklarından, bu işlemin tersini yapmak yeterli olacaktır. Bu doğrultuda A dizisinin N. elemanı, yeni tanımlanan bir B dizisinin I. elemanı olarak alınmaktadır. Dolayısıyla B dizisinin ilk elemanı en büyük olacaktır. Bunun için A7. adımda I alınarak A8. adımda B(I)=A(N) alınmak suretiyle işlemlere başlanmaktadır. A9. adımdan I' nın değeri N ile karşılaştırılmaktadır ve eğer I' nın değeri N' den küçük ise I bir artırılıp N değeri bir azaltılarak A8. adımdan itibaren işlemlere devam edilmektedir. Bu işlemler A9. adımdaki A9. adımdaki I=N karşılaştırılması gerçekleşene kadar devam etmektedir.

Algoritmanın Pascal Dilindeki Karşılığı:

```
Var
        i,n,m:integer;
        a,b:array[1..100] of integer;
begin
        write('n sayısını gir');
        readln(n);
        for i:=1 to n do
                readln(a[i]);
        m:=n;
        for i:=1 to m do
        begin
                b[i]:=a[n];
                n:=n-1;
        end;
        for i:=1 to m do
                writeln(b[i]);
end.
```

Örnek 4.6.14. N elemanlı bir tamsayı dizisinde tek ve çift olan elemanların sayısını bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. N değerini gir,
A3. I=1, T=0, C=0 al,
A4. A(I) değerini gir,
A5. DEĞER=TAM(A(I)/2)*2 al,
A6. Eğer DEĞER=A(I) ise A8. adıma git,
A7. T=T+1 al ve A9. adıma git,
A8. C=C+1 al,
A9. Eğer I=N ise A11. adıma git,
A10. I=I+1 al ve A4. adıma geri dön,
A11. T ve C' yi yaz,
A12. Dur.



Şekil 4.6.14. N elemanlı bir dizide tek ve çift olan elemanların sayısını bulan akış şeması.

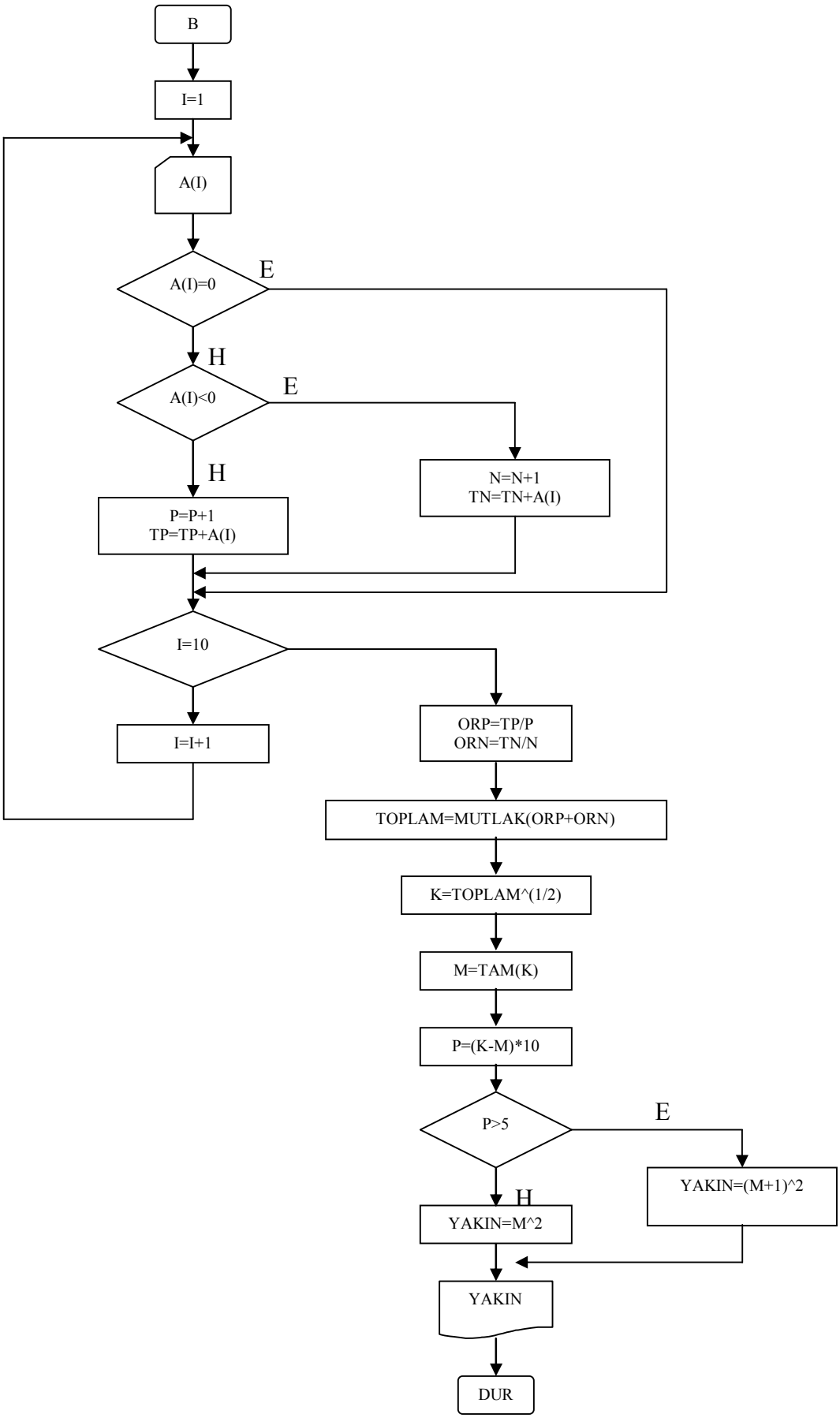
Örnekte A3. adımda T ve C gibi iki değişken tanımlanmıştır. Bunlardan T dizi elemanları içerisindeki tek sayıları belirlemek amacıyla ve C de çift sayıları belirlemek amacıyla tanımlanmıştır. A4. adımda girilen dizi elemanının A5. adımda yarısının tam kısmı alınarak elde edilen değer tekrar iki ile çarpılıp DEGER isimli değişkene yüklenmiştir. Buradaki amaç, A(I) dizi elemanının bu işlemler sonunda değerini koruyup korumadığının tespit edilmesidir. Eğer DEGER olarak adlandırılan değişken ile A(I) dizi değişkeni aynı değere sahip ise dizi değişkeninin değeri çift olacaktır, aksi taktirde değeri tek olacaktır. Bu durum A6. adımda sorgulanmaktadır. Eğer DEGER=A(I) ise çift sayı değişkeni olan C bir artırılmakta, aksi halde tek sayı değişkeni olan T bir artırılarak işlemlere devam edilmektedir. Bu işlemler A9. adımda verilen I=N şartının gerçekleşmesine kadar A4. adımdan itibaren devam etmektedir ve sonuçta elde edilen T ve C değeri yazdırılarak işlemlere son verilmektedir.

Algoritmanın Pascal Dilindeki Karşılığı:

```
Var
        i,n,t,c,deger:integer;
        a:array[1..100] of integer;
begin
        write('n sayısını gir');
        readln(a[i]);
        for i:=1 to n do
        begin
                readln(a[i]);
                deger:=trunc(a[i]/2)*2;
                if (deger=a[i]) then
                        c:=c+1
                else
                        t:=t+1;
        end;
        writeln('çift sayı sayısı=',c);
        writeln('tek  sayı sayısı=',t);
end.
```

Örnek 4.6.15. 10 elemanlı bir rasyonel sayı dizisinde negatif ve pozitif elemanların ayrı ayrı ortalamalarının mutlak değerleri toplamının en yakın tam kare sayıya uzaklığını bulan algoritma ve akış semasının oluşturulması.

A1. Başla,
A2. I=1, P=0, N=0, TP=0, TN=0 al,
A3. A(I) değerini gir,
A4. Eğer A(I)=0 ise A8. adıma git,
A5. Eğer A(I)<0 ise A7. adıma git,
A6. P=P+1, TP=TP+A(I) al,
A7. N=N+1, TN=TN+A(I) al,
A8. Eğer I=10 ise A10. adıma git,
A9. I=I+1 al ve A3. adıma geri dön,
A10. ORP=TP/P ve ORN=TN/N al,
A11. TOPLAM=MUTLAK(ORP+ORN) al
A12. K=TOPLAM^(1/2) al,
A13. M=TAM(K) al,
A14. P=(K-M)*10 al,
A15. Eğer P>5 ise A17. adıma git,
A16. YAKIN=M^2 al ve A18. adıma git,,
A17. YAKIN=(M+1)^2 al,
A18. YAKIN' ı yaz,
A19. Dur.



Şekil 4.6.15. 10 elemanlı bir rasyonel sayı dizisinde negatif ve pozitif elemanların ayrı ayrı ortalamalarının mutlak değerleri toplamının en yakın tam kare sayıya uzaklığını bulan akış şeması.

Örnekte A3. adımda girilen dizi elemanlarının A4. adımda sıfır olup olmadığı sorgulanmaktadır. Eğer bu şart gerçekleşiyorsa A8. adıma gidilerek diğer elemanının girilebilmesine ilişkin sorgulama yapılmaktadır. Aksi halde dizi elemanlarının değeri sıfırdan büyük ya da sıfırdan küçük olacağından A5. adımda sıfırdan küçük olma durumu sorgulanmaktadır. Eğer bu şart gerçekleşiyorsa A7. adıma gidilerek negatif sayaç olan N değeri bir artırılıp negatif toplama dizi elemanı aktarılmaktadır. Eğer dizi elemanı sıfırdan büyük ise A6. adım devreye girerek pozitif sayaç olan P değişkeni bir artırılarak dizi elemanı pozitif toplama aktarılmaktadır. Bu işlemler I sayacının değeri 10 oluncaya kadar devam etmektedir. Bütün elemanların girişi ve sorgulaması yapıldıktan sonra A10. adımda pozitif toplam olan TP değeri yine pozitif eleman sayısını veren P değerine bölünerek pozitif elemanların ortalaması hesaplanarak ORP değişkenine atanmaktadır. Benzer şekilde negatif toplam olan TN değişkeni de negatif sayıları tanımlayan N değişkenine bölünerek ortalama negatif değeri hesaplayarak ORN değişkenine atanmaktadır. A11. adımda negatif ve pozitif ortalamalar toplatılarak bunların mutlak değerleri TOPLAM değişkenine atanmaktadır.

Bu aşamadan sonra elde edilen bu değerin en yakın tam sayıya uzaklığının bulanması işlemlerine geçilmektedir. Bunun için A12. adımda K değişkenine TOPLAM değerinin karekökü alınarak, A13. adımda bu ifadenin tam kısmı M değişkenine atanmaktadır. K ile M' nin farkı K değerinin ondalıklı kısmını vereceğinden A14. adımda bu fark alınarak elde edilen değer 10 ile çarpılarak P değişkenine aktarılmaktadır. Elde edilen bu değer 5' den büyük ise TOPLAM değeri M değerinin bir fazlası olan sayının karesine yakın olacağından bu işlem A17. adımda yapılmaktadır. Aksi halde TOPLAM değeri M değerinin karesine en yakın olacağından A16. adımda bu işlem yapılmaktadır. Her iki halde de elde edilen sonuç YAKIN değişkenine atanarak A18. adımda bu ifade yazdırılarak işlemlere son verilmektedir.

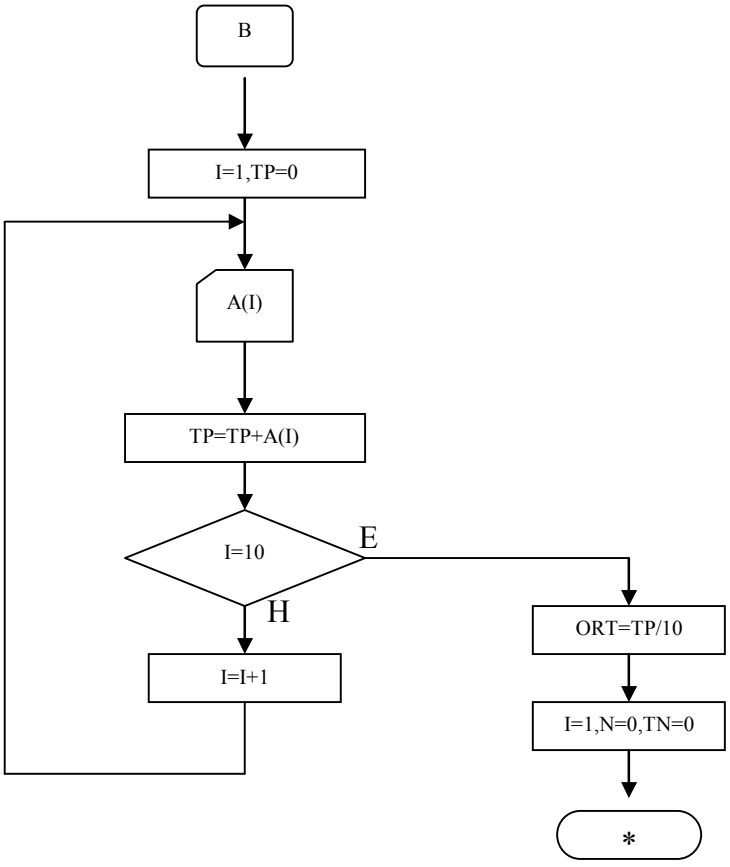
Algoritmanın Pascal Dilindeki Karşılığı:

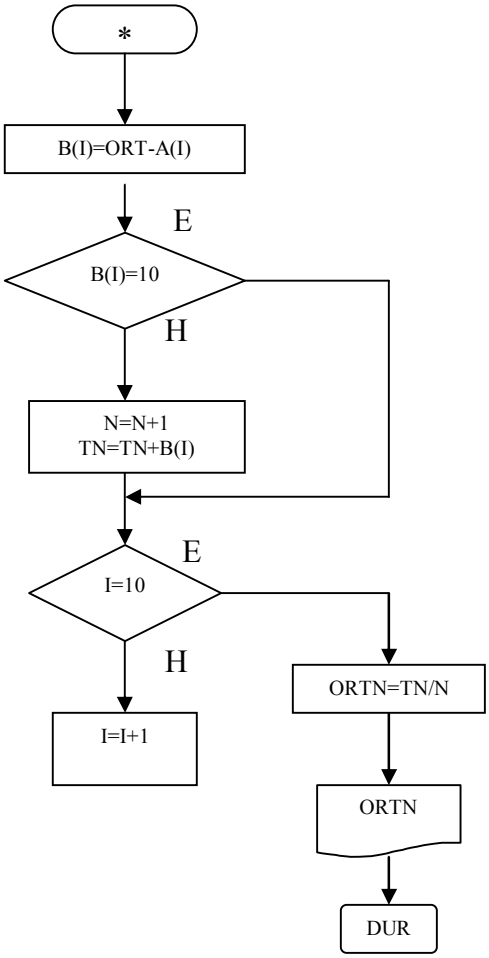
```
Var
        i,p,n,tp,tn,toplam,m,s,yakin:integer;
        a:array[1..100] of integer;
        orp,orn,k:real;
begin
        p:=0; n:=0; tp:=0; tn:=0;
        for i:=1 to 10 do begin
                readln(a[i]);
                if(a[i]<0) then begin
                        n:=n+1; tp:=tp+a[i];
                end;
                if (a[i]>0) then begin
                        p:=p+1; tp:=tp+a[i];
                end;
        end;
        orp:=tp/p; orn:=tn/n;
        toplam:=abs(orn+orp);
        k:=sqrt(toplam);
        m:=trunc(k);
        s:=(k-m)*10;
```

```
        if(s>5) then yakin:=sqr(m+1)
                else sqr(m);
        writeln('yakin=',yakin);
end.
```

Örnek 4.6.16. 10 elemanlı bir sayı dizisinin ortalaması ile her bir elemanın farkını bir diziye yükleyerek bu dizideki negatif elemanların ortalamasını bulan algoritma ve akıs semasının olusturulması.

A1. Başla,
A2. I=1, TP=0 al,
A3. A(I) değerini gir,
A4. TP=TP+A(I) al,
A5. Eğer I=10 ise A7. adıma git,
A6. I=I+1 al ve A3. adıma geri dön,
A7. ORT=TP/10 al,
A8. I=1, N=0, TN=0 al,
A9. B(I)ORT-A(I) al,
A10. Eğer B(I)>=0 ise A12. adıma git,
A11. N=N+1, TN=TN+B(I) al,
A12. Eğer I=10 ise A14. adıma git,
A13. I=I+1 al ve A9. adıma git,
A14. ORTN=TN / N al,
A15. ORTN' yi yaz,
A16. Dur.





Şekil 4.6.16. 10 elemanlı bir sayı dizisinin ortalaması ile her bir elemanının farkını bir diziye yükleyerek bu dizideki negatif elemanların ortalamasını bulan akış şeması.

Algoritmanın ilk adımında dizi elemanlarının girilmesi ve girilen bu elemanların toplanması işlemleri yapılmaktadır. A7. adımda toplanan dizi elemanlarının ortalaması hesaplanarak ORT isimli değişkene atanmaktadır. Bu aşamadan sonra ORT değeri ile dizi elemanlarının her birinin farkı alınarak elde edilen bu değerlerin başka bir dizi elemanına atanması işlemine geçilmektedir. Bu aşamada yeni dizinin elemanlarının sorgulanması yapılarak negatif olanların toplanması gerçekleştirilmektedir. Bu işlemler A8. adım ile A13. adımlar arasındaki işlemlerle yapılmaktadır. Elde edilen negatif sayı toplamı A14. adında negatif eleman sayısına bölünerek negatif elemanların ortalaması bulunarak ORTN isimli değişkene aktarılmaktadır. Elde edilen bu değer yazdırılarak işlemlere son verilmektedir.

Algoritmanın Pascal Dilindeki Karşılığı:

```
Program  islem;
Var
        i:integer;
        a,b:array[1..100] of integer;
        ort,orn,tn,tp,n:real;
```

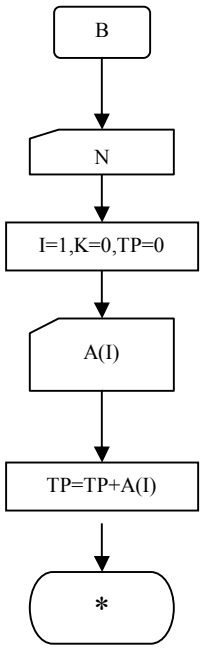
```
begin
        tp:=0;
        for i:=1 to 10 do begin
                readln(a[i]);
                tp:=tp+a[i];
        end;
        ort:=tp/10;
        n:=0;tn:=0;
        for i:=1 to 10 do
        begin
                b[i]:=ort-a[i];
                if (b[i]<0) then
                begin
                        n:=n+1;
                        tn:=tn+b[i];
                end;
        end;
        ortn:=tn/n;
        writeln('ortalama=',ortn);
end.
```

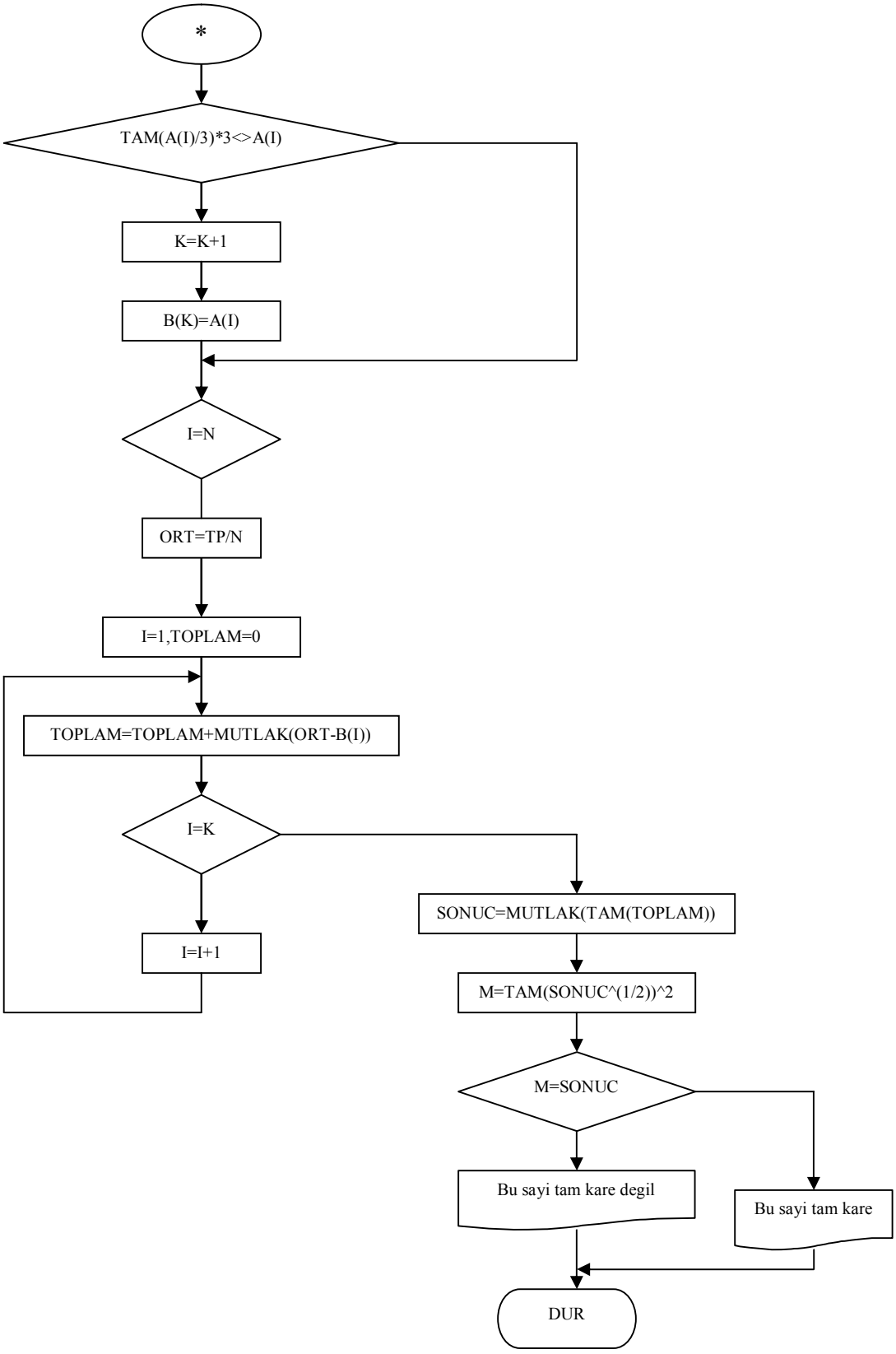
Örnek 4.6.17. N elemandan oluşan bir tam sayı dizisinin ortalaması ile bu dizide üç ile tam bölünebilen sayıların mutlak değerinin tam kısmının bir tam kare olup olmadığını araştıran algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. N' yi gir,
A3. I=1, K=0, TP=0 al,
A4. A(I)' yı gir,
A5. TP=TP+A(I) al,
A6. Eğer TAM(A(I)/3)*3<>A(I) ise A9. adıma git,
A7. K=K+1 al,
A8. B(K)=A(I) al,
A9. Eğer I=N ise A11. adıma git,
A10. I=I+1 al ve A4. adıma geri dön,
A11. ORT=TP/N al,
A12. I=1, TOPLAM=0 al,
A13. TOPLAM=TOPLAM+MUTLAK(ORT-B(I)) al,
A14. Eğer I=K ise A16. adıma git,
A15. I=I+1 al ve A13. adıma git,,
A16. SONUC=MUTLAK(TAM(TOPLAM)) al,
A17. M=TAM(SONUC^(1/2))^2 al,
A18. Eğer M=SONUC ise A20. adıma git,
A19. "Bu sayı tam kare değildir. " yaz ve A21. adıma git,
A20. "Bu sayı tam kare " yaz,
A21. Dur.

Algoritmanın Pascal Dilindeki Karşılığı:

```
Program  islem;
Var
        i,n,k,sonuc,m1,m:integer;
        a,b:array[1..100] of integer;
        ort,tp,toplam:real;
begin
        tp:=0;k:=0;
        for i:=1 to 10 do begin
                readln(a[i]);
                tp:=tp+a[i];
                if(abs(a[i]/3)*3=a[i]) then begin
                        k:=k+1;
                        b[k]:=a[i];
                end;
        end;
        ort:=tp/10;
        toplam:=0;
        for i:=1 to k do
                toplam:=toplam+abs(ort-b[i]);
        sonuc:=abs(trunc(toplam));
        m1:=trunc(sqrt(sonuc));
        m:=m*m;
        if (m=sonuc) then
                writeln('Bu sayı çift sayıdır..')
        else
                writeln('Bu sayı tam kare değildir..');
end.
```





Şekil 4.6.17. N elemanlı bir sayı dizisinin ortalaması ile bu dizide üç ile tam bölünen sayıların farkının toplamının mutlak değerinin tam kısmının tam kare olup olmadığını araştıran akış şeması.

4.7. MATRİSLER VE MATRİSLERE İLİŞKİN ALGORİTMA VE AKIŞ ŞEMALARI

4.7.1. Matris Kavramı Ve Örnekler

Bazı hallerde büyük miktarlarda veri dizileri kullanılır. Bu durumda matrislere gerek duyulabilir. Matrisler, birçok veriyi bünyesinde taşıyan ve her bir satırı birer dizi olarak düşünülen geniş bilgi alanları olarak tanımlanabilir. Matrislerde de dizilerde olduğu gibi indisler kullanılmaktadır. Bu indisler satır ve sütunları belirtmek üzere tanımlanırlar. Örneğin, 10 satır ve 10 sütundan meydana gelen bir A matrisi A(10,10) şeklinde tanımlanır. I satırları ve J de sütunları belirtmek üzere tanımlanırsa matrisin elemanları A(I, J)' ler olarak belirtilir. Burada I ve J indislerinin değerleri en fazla 10 olabilmektedir.

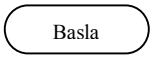
Matrisler tanımlanırken genel olarak boyutları ile ifade edilirler. Örneğin NxM' lik bir matris denildiğinde N satır ve M sütundan meydana gelen ve NxM elemandan oluşan bir matris tanımlanmış anlamındadır.

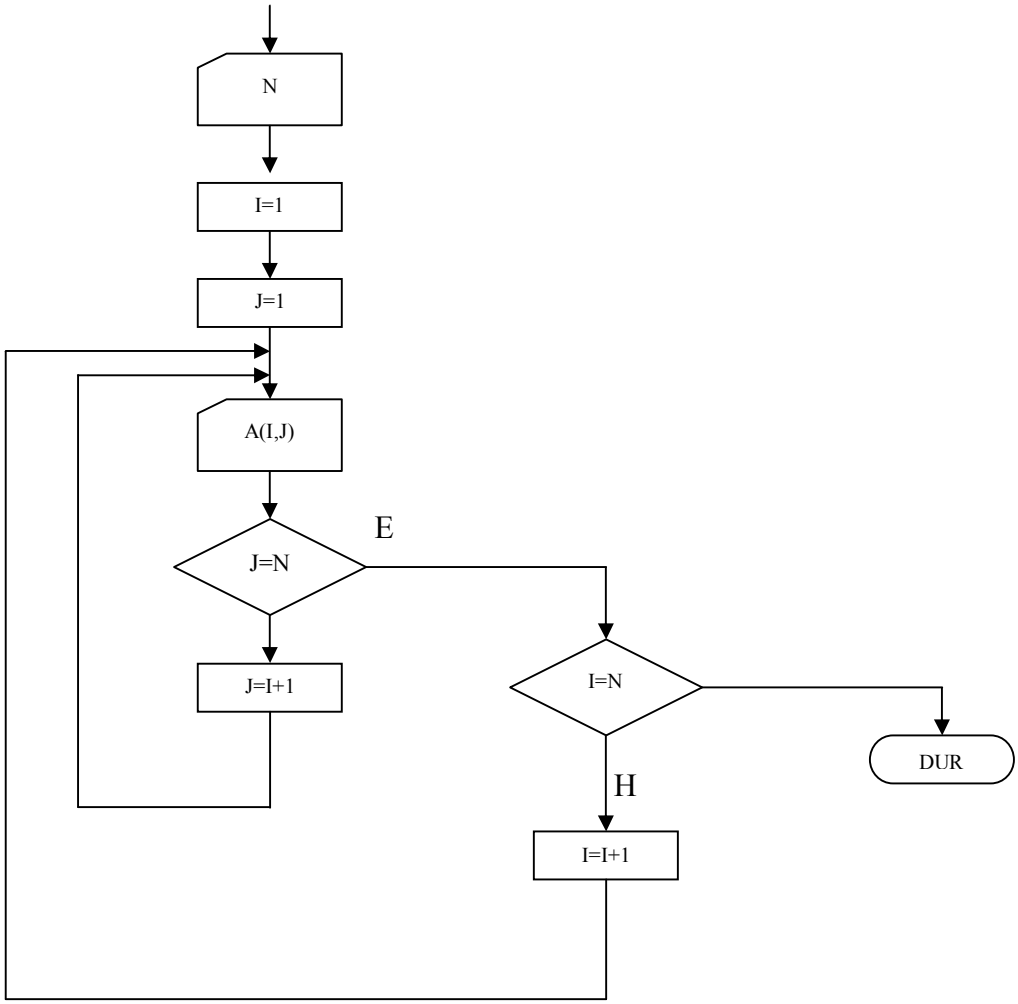
Örnek 4.7.1. NxN lik bir matrisin elemanlarının girişini yapan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. N' yi gir,
A3. I=1,
A4. J=1 al,
A5. A(I,J)' yi gir,
A6. Eğer J=N ise A8. adıma git,
A7. J=J+1 al ve A5. adıma geri dön,
A8. Eğer I=N ise A10. adıma git,
A9. I=I+1 al ve A4. adıma geri dön,
A10. Dur.

Algoritmanın Pascal Dilindeki Yazılımı :

```
var
        n,i,j : integer;
        a: array[1..100,1..100] of integer;
begin
        write ('n sayısını gir : ');
        readln(n);
        for i:=1 to n do begin
                for j:=1 to n do readln(a[i,j]);
        end;
end.
```





Şekil 4.7.1. NxN' lik bir matrisin girişini yapan akış şeması

Verilen örnekte NxN' lik bir matrisin elemanlarının klavyeden girişini sağlayan algoritma ve akış şeması istenmektedir. Bunun için A2. adımda N değerinin girişi yapıldıktan sonra, A3 ve A4. adımlarda matrisin satırını ve sütununu tanımlayan I ve J değişkenleri tanımlanmıştır. Burada I değişkeni satırı ve J değişkeni sütunu tanımlamaktadır. A5. adımda matrisin A(I,J)' inci elemanının girişi istenmektedir. I ve J' nin değerleri 1 olduğundan A(I,J)=A(1,1) matrisin ilk elemanını tanımlamaktadır. A6. adımda J değişkeni N ile karşılaştırılmaktadır, eğer J değişkeni N değerine ulaşmış ise matrisin ilk satırının girişi tamamlanmış olacaktır. Aksi halde J değişkeninin değeri 1 arttırılarak A5. adıma geri gönderilmektedir. J değişkeninin değeri N olunca A8. adım devreye girerek I değişkenin değeri N ile karşılaştırılmaktadır. Eğer I' nın değeri N olmuş ise matrisin girişi tamamlanmış olacaktır. Aksi halde A9. adımda I değişkeninin değeri 1 arttırılarak A4. adıma geri gönderilmektedir.

Örnek olarak N=3 alınırsa yukarıdaki algoritmanın akışı aşağıdaki gibi olacaktır.

I=1, J=1 iken A(I,J) = A(1,1) = 2 girilsin,
J=J+1=2 iken A(I,J) = A(1,2) = 3 girilsin,
J=J+1=3 iken A(1,3) = A(1,3) = 1 girisin,

Bu aşamada J değeri 3 olduğundan I değişkeninin değeri 1 arttırılarak J değişkeninin değeri tekrar 1 alınacaktır.

I=I+1=2 ve J=1 iken A(I,J) = A(2,1) = -2 girilsin,
J=J+1=2 iken A(I,J) = A(2,2) = 6 girilsin,
J=J+1=3 iken A(I,J) = A(2,3) = -3 girilsin,

J değişkeninin değeri tekrar 3 olduğundan I değişkeninin değeri 1 arttırılarak J değişkeninin değeri tekrar 1 alınacaktır.

I=I+1=3 ve J=1 iken A(I,J) = A(3,1) = 9 girilsin,
J=J+1=2 iken A(I,J) = A(3,2) = 0 girilsin,
J=J+1=3 iken A(I,J) = A(3,3) = 7 girilsin,

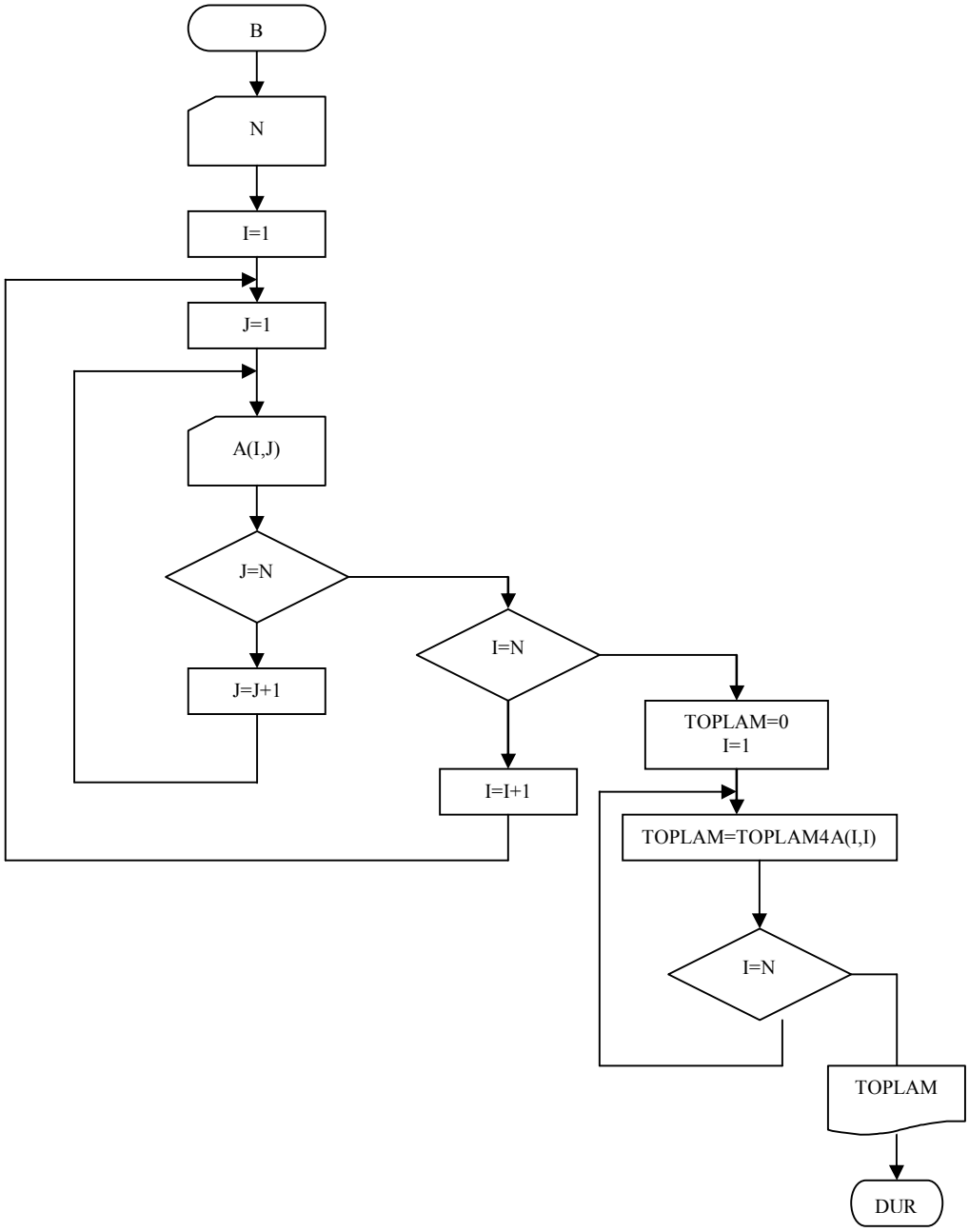
Bu aşamada J değişkeninin ve I değişkeninin değeri 3 olacağından matrisin elemanlarının girişi tamamlanmış olacaktır.

Örnek 4.7.2. NxN lik bir matrisin esas köşegeni üzerindeki elemanlarının toplamını bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. N' yi oku,
A3. I=1,
A4. J=1 al,
A5. A(I,J)' yi gir,
A6. Eğer J=N ise A8. adıma git,
A7. J=J+1 al ve A5. adıma geri dön,
A8. Eğer I=N ise A10. adıma git,
A9. I=I+1 al ve A4. adıma geri dön,
A10. TOPLAM=0, I=1 al,
A11. TOPLAM=TOPLAM+A(I,J) al,
A12. Eğer I=N ise A14. adıma git,
A13. I=I+1 al ve A11. adıma geri dön,
A14. TOPLAM değerini yaz,
A15. Dur.

Algoritmanın Pascal Dilindeki Yazılımı :

```
var
        i,j,n,z,k,toplam : integer;
        a: array[1..10,1..10] of integer;
begin
        write (‘n sayısını gir : ‘);   readln(n);
        for i:=1 to n do begin
                for j:=1 to n do readln(a[i,j]);
        end;
        toplam:=0;
        for i:=1 to 10 do
                toplam:=toplam+a[i,j];
        writeln(‘toplam : ‘,toplam);
end.
```



Şekil 4.7.2. NxN' lik bir matrisin esas köşegeni üzerindeki elemanlarının toplamını bulan akış şeması

Verilen örnekte ilk 9 adım bir önceki örnekteki gibi matrisin elemanlarının girişinin yapılmasını sağlamaktadır. A10. adımda TOPLAM adında bir değişken tanımlanmıştır. Bu değişkenin görevi matrisin esas köşegeni üzerindeki elemanlarını toplamaktır. İlk toplam değeri 0 olduğundan TOPLAM değişkenine ilk değer olarak 0 değeri atanmıştır. Matrisin esas köşegeni üzerindeki elemanların satır ve sütunları eşit olacağından bu elemanları tek bir değişken ile kontrol etmek mümkün olabilecektir. Bu doğrultuda A10. adımda I değişkenine 1 değeri verilerek matrisin esas köşegeni üzerindeki elemanları bu değişkenle kontrol edilmektedir. A11. adımda TOPLAM değişkenine ilk değeri ile birlikte A(I,I) elemanı atanmıştır. A12. adımda I değişkeni N

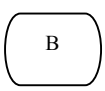
ile karşılaştırılmaktadır. Eğer I değişkeni N değerine eşit ise esas köşegen üzerindeki elemanların toplanması işlemleri gerçekleşmiş olacağından A14. adıma gidilerek TOPLAM değeri yazdırılıp işlemlere son verilmektedir. Aksi halde I değişkeninin değeri 1 arttırılarak A11. adımdan itibaren toplama işlemine devam edilmektedir.

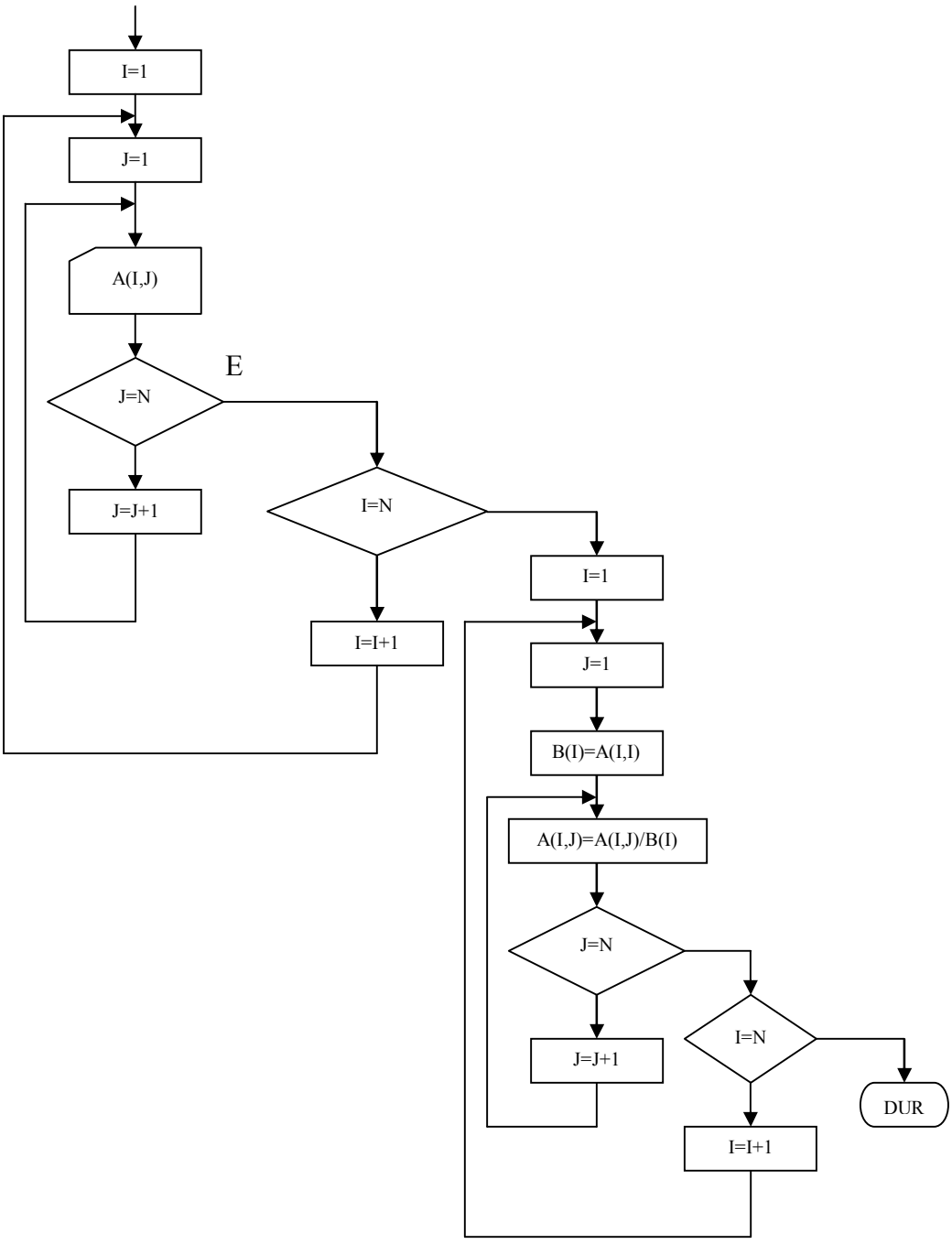
Örnek 4.7.3. Sıfırdan farklı NxN lik bir matrisin esas köşegeni üzerindeki elemanlarını 1 (bir) yapan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. N' yi oku,
A3. I=1 al,
A4. J=1 al,
A5. A(I,J)' yi gir,
A6. Eğer J=N ise A8. adıma git,
A7. J=J+1 al ve A5. adıma geri dön,
A8. Eğer I=N ise A10. adıma git,
A9. I=I+1 al ve A4. adıma geri dön,
A10. I=1,
A11. J=1,
A12. B(I)=A(I,I) al,
A13. A(I,J)=A(I,J)/B(I) al,
A14. Eğer J=N ise A16. adıma git,,
A15. J=J+1 al ve A13. adıma geri dön,.
A16. Eğer I=N ise Dur.
A17. I=I+1 ve A11. adıma geri dön,
A18. Dur.

Algoritmanın Pascal Dilindeki Yazılımı :

```
var
        i,j,n : integer;
        a: array[1..10,1..10] of integer;
        b. array[1..10] of real;
begin
        write (‘n sayısını gir : ‘);
        readln(n);
        for i:=1 to 10 do begin
                for j:=1 to 10 do readln(a[i,j]);
        end;
        for i:=1 to 10 do
                b[i]:=a[i,i];
                for j:=1 to 10 do begin
                        a[i,j]:=a[i,j]/b[i];
                        writeln(a[i,j]);
                end;
                writeln;
        end;
end.
```





Şekil 4.7.3. NxN' lik bir matrisin esas köşegeni üzerindeki elemanları bir yapan akış şeması

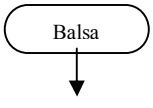
Verilen örnekte ilk 9 adım matrisin elemanlarının girişini gerçekleştirmektedir. A10 ve A11. adımlarda I ve J değişkenleri 1 değerini alarak yeni bir döngü oluşturulmaktadır. A12. adımda B isimi bir dizi tanımlanarak A matrisinin esas köşegeni üzerindeki ilk elemanı bu diziye atanmaktadır. Buradaki amaç, A matrisinin esas köşegeni üzerindeki elemanlarını bu diziye yükleyerek ilgili satırdaki tüm elemanların bu dizi elemanına bölünebilmesini sağlamaktır. Bu işlemler sonunda da esas köşegen üzerindeki elemanlar 1 olacaktır.

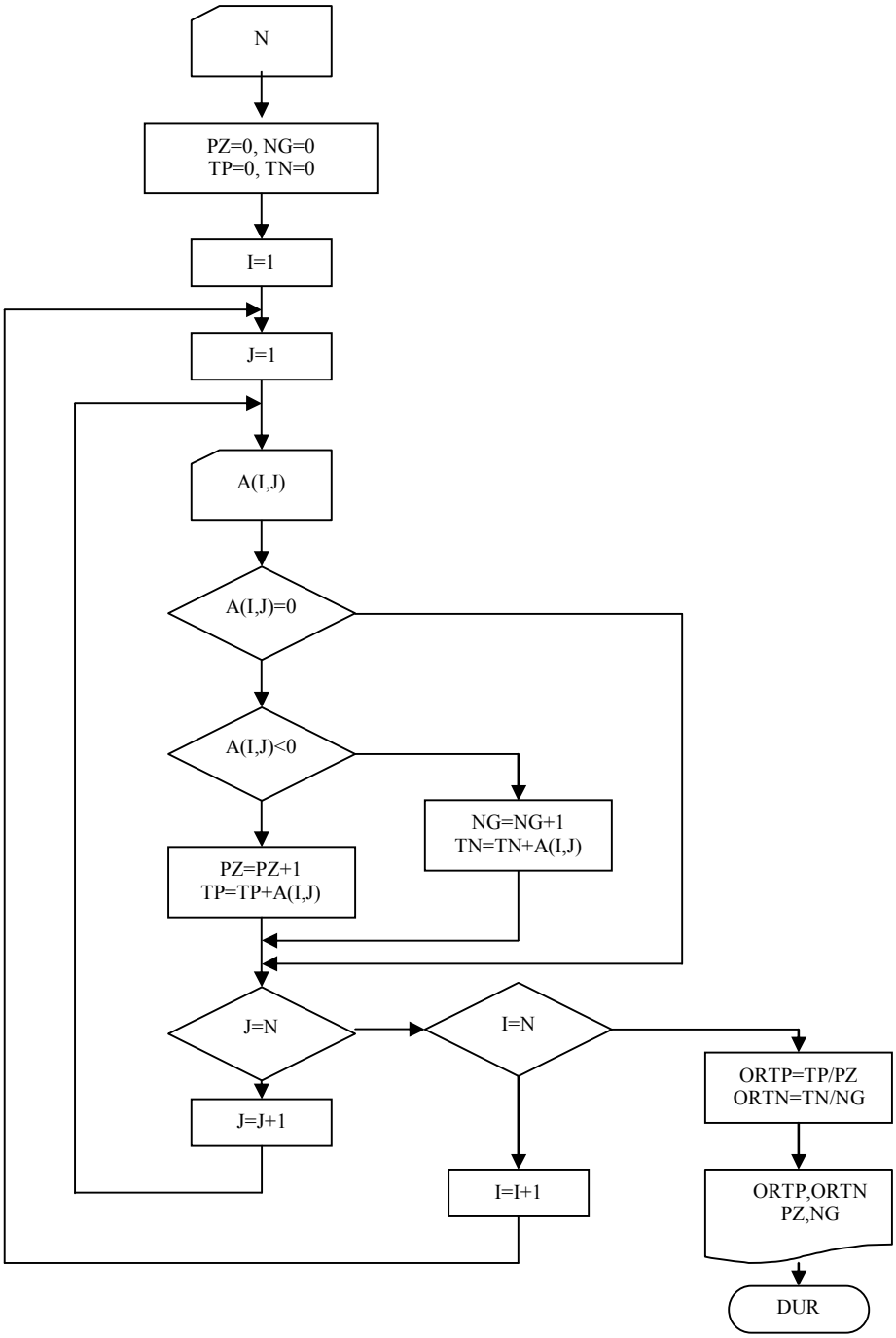
Örnek 4.7.4. NxN lik bir matriste negatif ve pozitif elemanların sayısını ve ortalamalarını bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. N' yi oku,
A3. PZ=0, NG=0, TP=0, TN=0 al,
A4. I=1 al,
A5. J=1 al,
A6. A(I,J)' yi gir,
A7. Eğer A(I,J)=0 ise A11. adıma git,
A8. Eğer A(I,J)=<0 ise A10. adıma git,
A9. PZ=PZ+1, TP=TP+A(I,J) al ve A11. adıma git,
A10. NG=NG+1, TN=TN+A(I,J) al ,
A11. Eğer J=N ise A13. adıma git,
A12. J=J+1 al ve A6. adıma geri dön,
A13. Eğer I=N ise A15. adıma git,
A14. I=I+1 al ve A5. adıma geri dön,
A15. ORTP=TP/PZ, ORTN=TN/NG al,
A16. ORTP, ORTN, PZ ve NG' yi yaz ve Dur.

Algoritmanın Pascal Dilindeki Yazılımı :

```
var
        i,j,pz,ng : integer;
        tp, tn, ortp, ortn : real;
        a: array[1..20,1..20] of real;
begin
        write (‘n sayısını gir : ‘);
        readln(n);
        pz:=0;  ng:=0;   tp:=0;     tn:=0;
        for i:=1 to n do begin
                for j:=1 to n do begin
                        readln(a[i,j]);
                        if (a[i,j]<0) then begin
                                ng:=ng+1;      tn:=tn+a[i,j];
                        end;
                        if (a[i,j]>0) then begin
                                pz:=pz+1;
                                tp:=tp+a[i,j];
                        end;
                end;
        end;
        ortp:=tp/pz;
        ortn:=tn/ng;
        writeln(‘pozitif sayı : ‘,pz);
        writeln(‘pozitif ortalama : ‘,ortp);
        writeln(‘negatif sayı : ‘,ng);
        writeln(‘negatif ortalama : ‘,ortn);
end.
```





Şekil 4.7.4. NxN' lik bir matrisin negatif ve pozitif elemanlarının sayısını ve ortalamasını bulan akış şeması

A3. adımda PZ ve NG gibi iki değişken tanımlanmıştır. BU değişkenlerin görevleri A matrisi içerisindeki pozitif ve negatif elemanların sayısını bulmaktır. PZ pozitif ve NG negatif elemanların sayısını tanımlamaktadır. Yine aynı adımda TP ve TN gibi iki değişken tanımlanmıştır. BU değişkenlerin görevleri pozitif ve negatif elemanların toplanmasını sağlamaktır. TP pozitif ve TN negatif elemanların toplanmasını sağlamaktadır. A6. adımda girilen eleman A7. adımda sorgulanmaktadır. Eğer bu elemanın değeri 0 ise işleme alınmamaktadır. Benzer şekilde A8. adımda girilen elemanın değeri sıfırdan küçük ise A10. adımda negatif eleman sayacı NG bir

arttırılarak bu eleman TN isimli toplam değişkenine ilave edilmektedir. Aksi halde girilen eleman pozitif olacağından A9. adım devreye girerek pozitif eleman sayacı olan PZ bir arttırılıp bu eleman TP isimli toplam değişkenine ilave edilmektedir. Bu işlemler A matrisinin tüm elemanlarının girilmesine kadar devam etmektedir. Giriş tamamlandıktan sonra , A15. adımda pozitif ve negatif elemanların ortalaması bulunmaktadır. ORTP olarak adlandırılan pozitif elemanların ortalaması TP değerinin PZ değerine bölünmesiyle elde edilmektedir. Benzer şekilde ORTN ile negatif elemanların ortalaması da TN değerinin NG değerine bölünmesiyle elde edilmektedir.

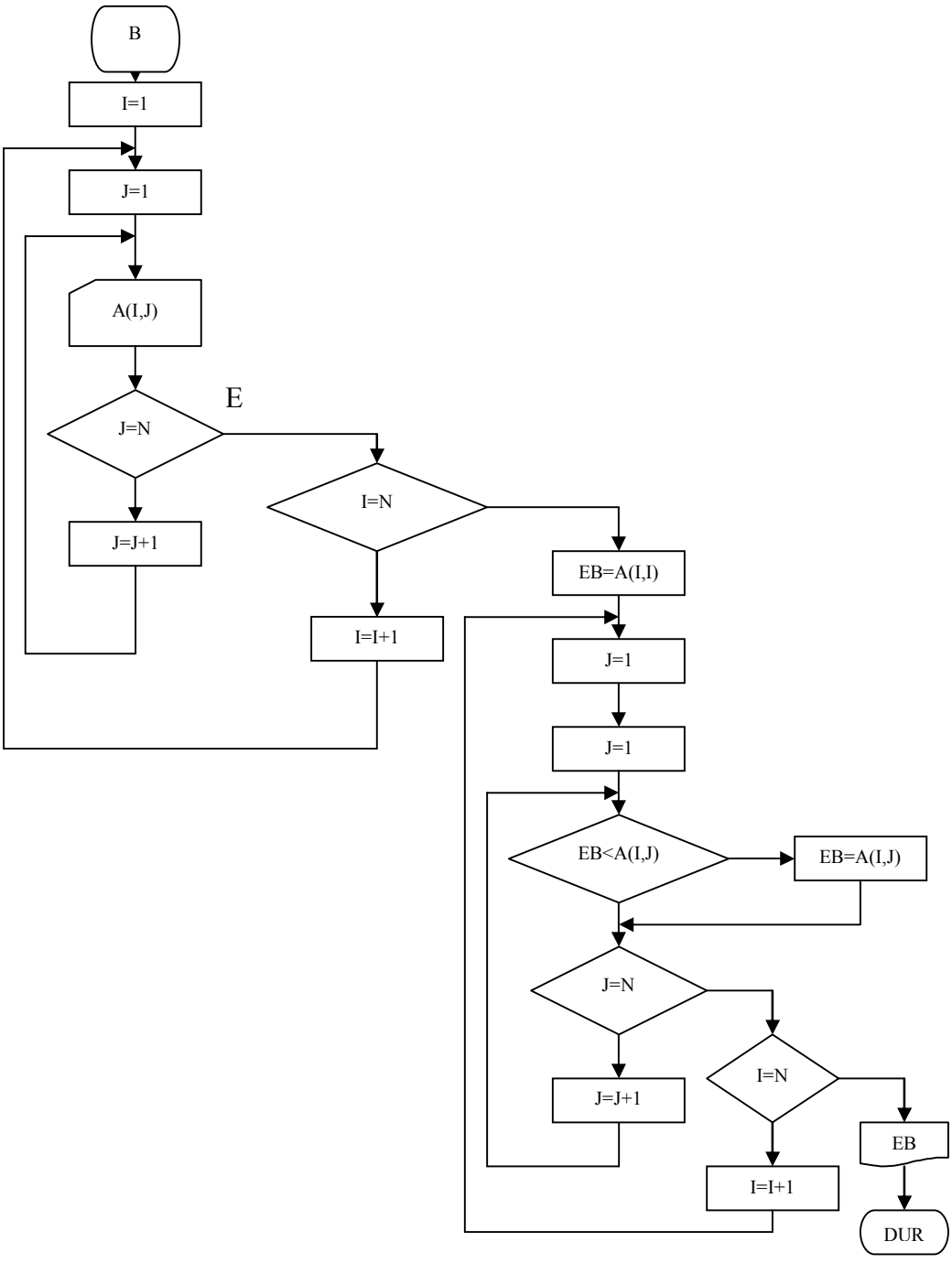
Örnek 4.7.5. NxM lik bir matrisin en büyük elemanını bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. N ve M' yi oku,
A3. I=1 al,
A4. J=1 al,
A5. A(I,J)' yi gir,
A6. Eğer J=M ise A8. adıma git,
A7. J=J+1 al ve A5. adıma geri dön,
A8. Eğer I=N ise A10. adıma git,
A9. I=I+1 al ve A4. adıma geri dön,
A10. EB=A(1,1) al,
A11. I=1 al,
A12. J=1 al,
A13. Eğer EB<A(I,J) ise EB=A(I,J) al,
A14. Eğer J=M ise A16. adıma git,,
A15. J=J+1 al ve A13. adıma geri dön,.
A16. Eğer I=N ise A18. adıma git,
A17. I=I+1 al ve A12. adıma geri dön,
A18. EB' yi yaz,
A19. Dur.

Algoritmanın Pascal Dilindeki Yazılımı :

```
var
        i,j,n,m,eb : integer;
        a: array[1..20,1..20] of integer;
begin
        write (‘n sayısını gir : ‘);    readln(n);
        write(‘m sayısını gir : ‘);    readln(m);
        for i:=1 to n do begin
                for j:=1 to m do readln(a[i,j]);
        end;
        eb:=a[1,1];
        for i:=1 to n do begin
                for j:=1 to n do
                        if (eb<a[i,j]) then eb:=a[i,j];
        end;
```

writeln(‘en büyük : ‘,eb);
end.



Şekil 4.7.5. NxM’ lik bir matrisin en büyük elemanını bulan akış şeması

İlk 8 adımda NxM’ lik matrisin elemanlarının girişi yapılmıştır. A9. adımda EB isimli bir değişken tanımlanarak, bu değişkene A matrisinin ilk elemanı atanmıştır. Buradaki EB değişkeninin görevi matristeki en büyük elemanı taşımaktır. Matrislerde en büyük elemanı bulma işlemleri, daha önce dizilerde yapılan en büyük elemanı bulma işlemlerine benzemektedir. Sadece burada işlemler N satır için yapılmaktadır. A10. ve A11. adımlarda tanımlanan I ve J değişkenleri yardımıyla A12. adımda EB değişkeni matrisin A(I,J) elemanıyla karşılaştırılmaktadır. Eğer EB değişkeni A(I,J) elemanından

küçük ise EB değişkenine A(I,J) elemanı atanmaktadır ve işlemler devam edilmektedir. Aksi halde bu atama işlemi yapılmadan bir sonraki adıma geçilerek buradan itibaren işlemlere devam edilmektedir. Bundan sonraki işlemler A11 ve A16. adımlar arasında sürmektedir.Matrisin tüm elemanlarının karşılaştırması yapıldıktan sonra A17. adımda en son elde edilen EB değişkeni yazdırılarak işlemlere son verilmektedir. Burada elde edilen en son EB değeri matrisin en büyük elemanı olacaktır.

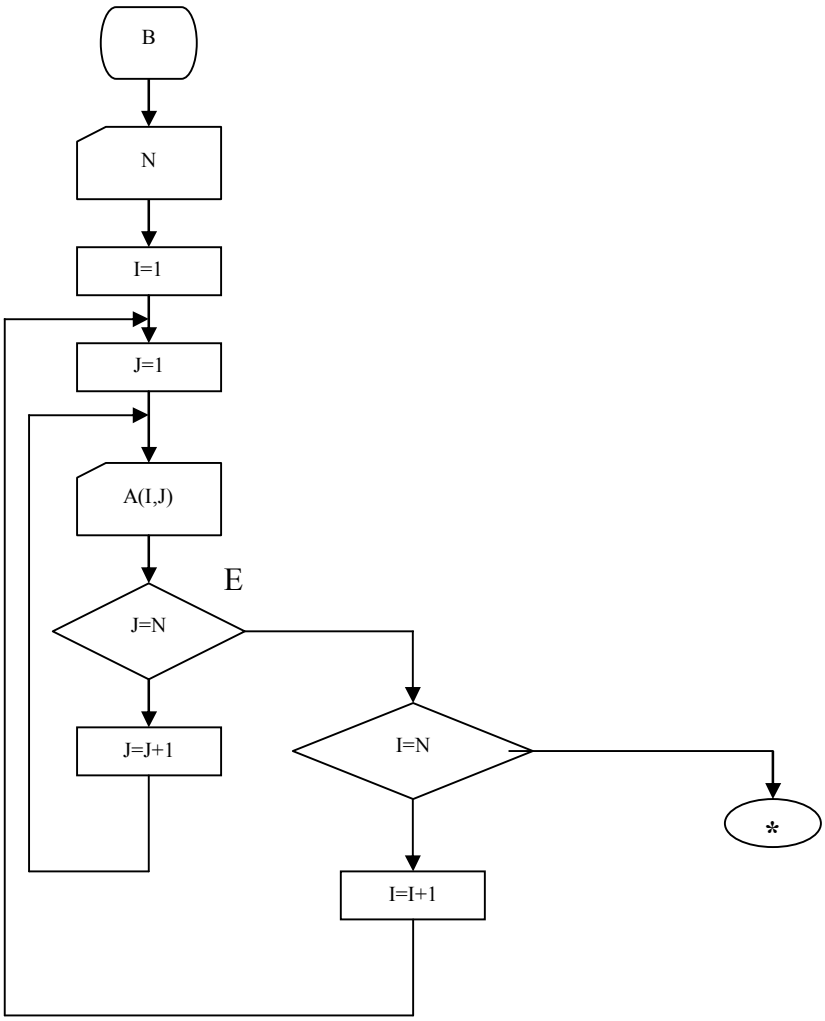
Örnek 4.7.6. NxN lik bir matrisin satır ve sütun toplamlarını ayrı ayrı bulan algoritma ve akış şemasının oluşturulması.

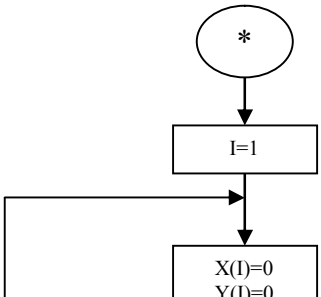
A1. Başla,
A2. N' yi oku,
A3. I=1 al,
A4. J=1 al,
A5. A(I,J)' yi gir,
A6. Eğer J=N ise A8. adıma git,
A7. J=J+1 al ve A5. adıma geri dön,
A8. Eğer I=N ise A10. adıma git,
A9. I=I+1 al ve A4. adıma geri dön,
A10. I=1 al,
A11. X(I)=0, Y(I)=0 al,
A12. J=1 al,
A13. X(I)=X(I)+A(I,J) al,
A14. Y(I)=Y(I)+A(J,I) al,
A15. Eğer J=N ise A17. adıma git,.
A16. J=J+1 al ve A13. adıma geri dön,
A17. Eğer I=N ise A19. adıma git,
A18. I=I+1 al ve A11. adıma geri dön,
A19. K=1 al,
A20. X(K) ve Y(K)' yı yaz,
A21. Eğer K=N ise A23. adıma git,
A22. K=K+1 al ve A20. adıma geri dön,
A23. Dur.

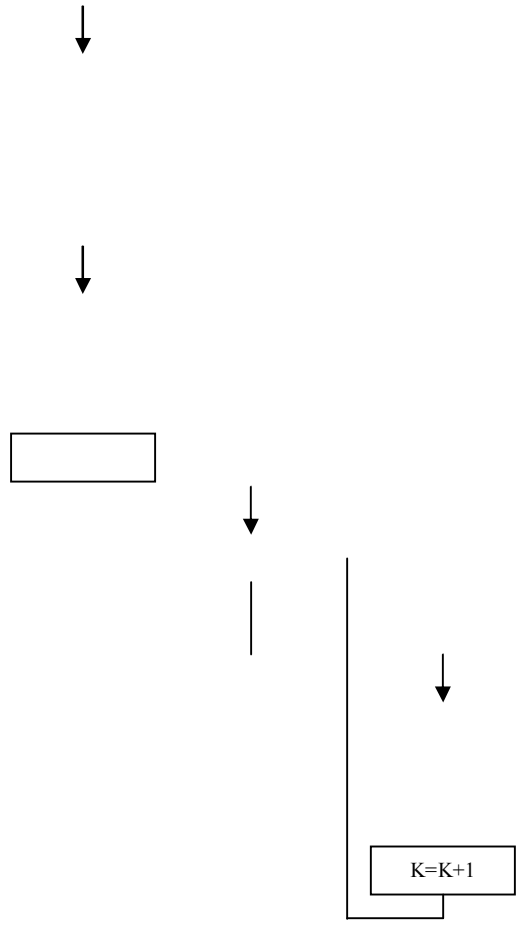
Algoritmanın Pascal Dilindeki Yazılımı :

```
var
        i,j,n,k : integer;
        a: array[1..20,1..20] of integer;
        x,y: array[1..20] of integer;
begin
        write ('n sayısını gir : ');
        readln(n);
        for i:=1 to n do
        begin

                for j:=1 to n do
                        readln(a[i,j]);
        end;
```

```
        for i:=1 to n do
        begin
                x[i]:=0;
                y[i]:=0;
                for j:=1 to n do begin
                        x[i]:=x[i]+a[i,j];
                        y[i]:=y[i]+a[j,i];
                end;
        end;
        for k:=1 to n do begin
                writeln(x[k],' ',y[k]);
end.
```







Şekil 4.7.6. NxM' lik bir matrisin satır ve sütun toplamlarını ayrı ayrı bulan akış şeması

İlk 9 adım matrisin elemanlarının girişini sağlamaktadır. Matrisin satır ve sütunlarının toplanması işlemlerine A10. adımdan itibaren geçilmektedir. Bu aşamada X ve Y gibi iki dizi tanımlanmaktadır. Bunlardan X dizisi matrisin satır elemanlarının toplamını, Y dizisi de sütun elemanlarının toplamını tanımlamak üzere oluşturulmuştur. A10. adımda I değişkenine ilk değer olarak 1 atandıktan sonra A11. adımda X ve Y dizilerinin I. (birinci) elemanlarına ilk değer olarak 0 değeri atanmıştır. Bunun anlamı ilk aşamada herhangi bir toplamın olmadığıdır. A12. adımda J değişkenine 1 değeri atandıktan sonra A13. adımda toplama işlemlerine geçilmektedir. Bu adımda X dizisine önceki değeriyle birlikte A dizisinin I ve J' ye karşılık gelen A(I,J) elemanı atanmaktadır. A14. adımda ise sütunları toplayan Y dizisine önceki değeriyle birlikte A matrisinin J. satır ve I. sütundaki elemanı atanmıştır. Burada dikkat edildiğinde, (I,J) ile (J,I) farklı elemanları tanımlamaktadır. Burada J değişkeni hızlı döndüğünden birincisi satır elemanını tanımlarken diğeri sütun elemanını tanımlamaktadır. Bu işlemler I değişkeninin değeri N oluncaya kadar devam etmektedir. I=N olunca A19. adımdan itibaren X ve Y dizileri yazdırılarak işlemlere son verilmektedir.

Örnek olarak N=3 alınarak matrisin elemanlarının

1	2	4
5	1	3
1	1	2

şeklinde girildiği varsayıldığında algoritmanın akışı aşağıdaki gibi olacaktır.

I=1 için; X(I)=X(1)=0 ve Y(I)=Y(1)=0 olacaktır,
J=1 için; X(I)=X(I)+A(I,J)=0+1=1 ve Y(I)=Y(I)+A(J,I)=0+1=1 olacaktır,
J=J+1=2 için; X(I)=X(2)=1+2=3 ve Y(I)=Y(2)=1+5=6 olacaktır,
J=J+1=3 için; X(I)=X(3)=3+4=7 ve Y(I)=Y(3)=6+1=7 olacaktır,

I=I+1=2 için; X(I)=X(2)=0 ve Y(I)=Y(2)=0 olacaktır,
J=1 için; X(I)=X(2)=0+5=5 ve Y(I)=Y(2)=0+2=2 olacaktır,
J=J+1=2 için; X(I)=X(2)=5+1=6 ve Y(I)=Y(2)=2+1=3 olacaktır,
J=J+1=3 için; X(I)=X(2)=6+3=9 ve Y(I)=Y(2)=3+1=4 olacaktır,

I=I+1=3 için; X(I)=X(3)=0 ve Y(I)=Y(3)=0 olacaktır,
J=1 için; X(I)=X(3)=0+1=1 ve Y(I)=Y(3)=0+4=4 olacaktır,
J=J+1=2 için; X(I)=X(3)=1+1=2 ve Y(I)=Y(3)=4+3=7 olacaktır,
J=J+1=3 için; X(I)=X(3)=2+2=4 ve Y(I)=Y(3)=7+2=9 olacaktır.

Elde edilen bu değerler aşağıdaki gibi olacaktır.

X(1)=7 ve Y(1)=7
X(2)=9 ve Y(2)=4
X(3)=4 ve Y(3)=9

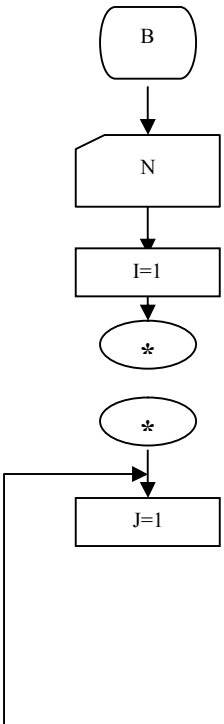
Örnek 4.7.7. Elemanter satır işlemleri yardımıyla NxN lik bir matrisin esas köşegeni üzerindeki elemanlarını 1 ve esas köşegeninin dışında kalan elemanlarını 0 yapan algoritma ve akış şemasının oluşturulması.

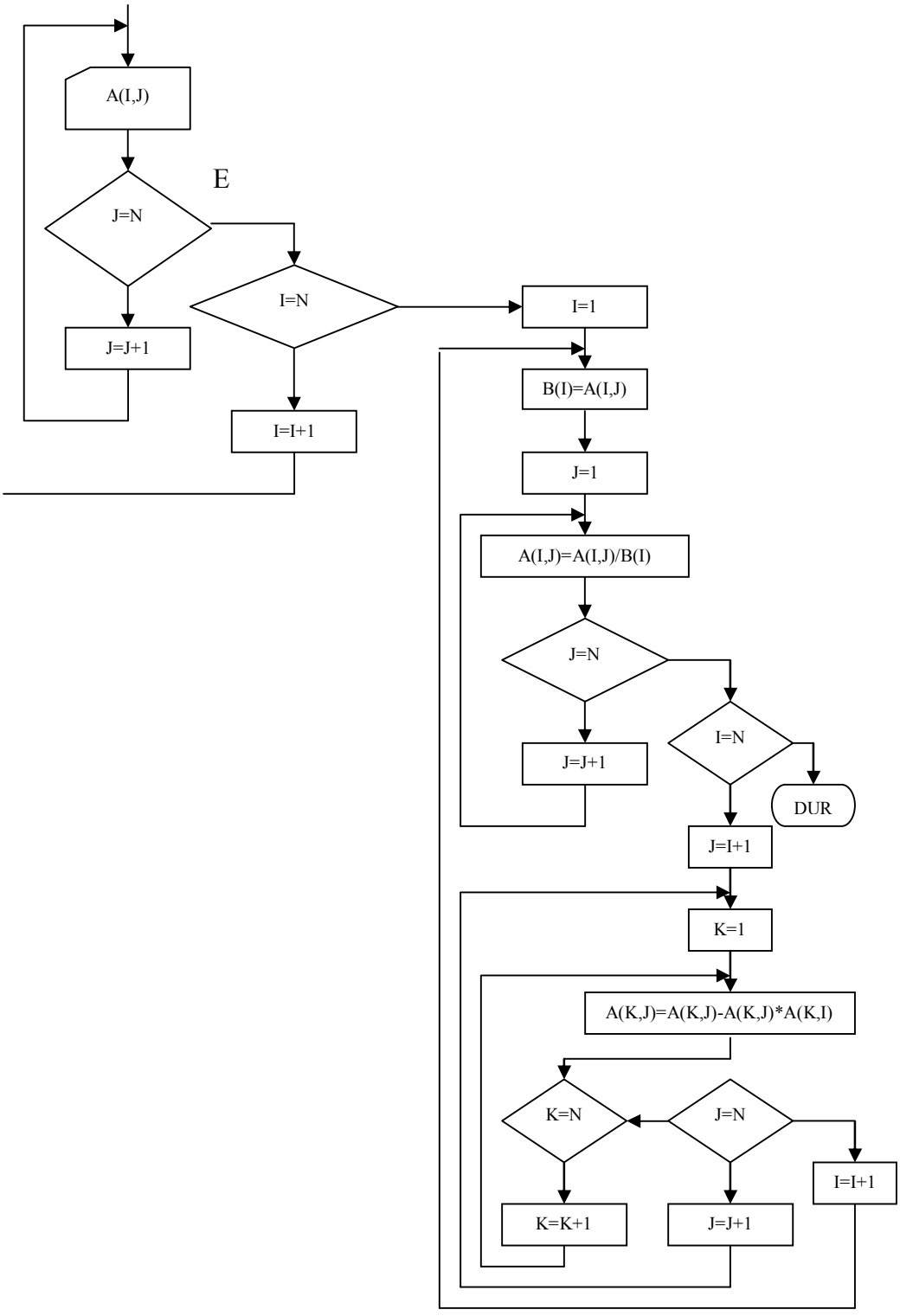
A1. Başla,
A2. N' yi oku,
A3. I=1 al,
A4. J=1 al,
A5. A(I,J)' yi gir,
A6. Eğer J=N ise A8. adıma git,
A7. J=J+1 al ve A5. adıma geri dön,
A8. Eğer I=N ise A10. adıma git,
A9. I=I+1 al ve A4. adıma geri dön,
A10. I=1 al,
A11. B(I)=A(I,I) al,
A12. J=1 al,
A13. A(I,J)=A(I,J)/B(I) al,
A14. Eğer J=N ise A16. adıma git,.
A15. J=J+1 al ve A13. adıma geri dön,
A16. Eğer I=N ise A25. adıma git,
A17. J=I+1 al,
A18. K=1 al,
A19. A(K,J)=A(K,J)-A(K,J)*A(K,J) al,

A20. Eğer K=N ise A22. adıma git,
A21. K=K+1 al ve A19. adıma geri dön,
A22. Eğer J=N ise A24. adıma git,
A23. J=J+1 al ve A18. adıma geri dön,
A24. I=I+1 al ve A11. adıma geri dön,
A25. Dur.

Algoritmanın Pascal Dilindeki Yazılımı :

```
program matris_islemleri;
var
        i,j,n,k : integer;
        a: array[1..20,1..20] of real;
        b: array[1..20] of real;
begin
        write (‘n sayısını gir : ‘);
        readln(n);
        for i:=0 to n-1 do begin
                for j:=0 to n-1 do readln(a[i,j]);
        end;
        for i:=0 to n-1 do begin
                b[i]:=a[i,j];
                for j:=0 to n-1 do
                        a[i,j]:=a[i,j]/b[i];
                for j:=i+1 to n-1 do begin
                        for k:=0 to n-1 do
                                a[k,j]:=a[k,j]-a[k,j]*a[k,i];
                end;
        end;
        for i=0 to n-1 do begin
                for j:=0 to n-1
                        writeln(a[i,j]);
        end;
end.
```





Şekil 4.7.7. Elemanter satır işlemleri ile bir kare matrisin esas köşegeni üzerindeki elemanlarını 1 ve diğer elemanlarını 0 yapan akış şeması

Verilen örnekte matrisin elemanlarının girişi yapıldıktan sonra A10. adımda I değişkeni 1 değerini almıştır ve A11. adımda B isimli bir dizi tanımlanarak bu dizinin I. elemanına A matrisinin A(I,I)'ıncı elemanı atanmıştır. Buradaki amaç, matrisin esas

köşegeni üzerindeki elemanların 1 değerine ulaşabilmesini sağlamak için her bir satırın B dizisinde tanımlanan elemanlara bölünmesini sağlamaktır. Matrisin (I,I). elemanı zaten esas köşegen üzerindeki eleman olacağından ilgili satırları bu elemana bölmek yeterli olacaktır. A12. adımda J değişkeni 1 değerini alarak, A13. adımda biraz önce bahsedilen bölme işlemlerine geçilmiştir. Bilindiği üzere bir matrisin herhangi bir satırını herhangi bir elemana bölmek ya da bu elemanla çarpmak bu matrisin değerini değiştirmemektedir. Aynı şekilde elemanter satır ve sütun işlemleri olarak tanımlanan işlemlerde de matrisin değeri değişmemektedir. Herhangi bir satır ya da sütun bir değerle çarpılıp diğer bir sütuna veya satıra ilave edildiğinde matrisin değeri değişmemektedir. Bu doğrultuda A13. adımda A matrisinin (I,J). elemana karşılık gelen değeri B dizisinin I. elemanına karşılık gelen değerine bölünerek elde edilen değer tekrar A matrisinin (I,J). elemanına aktarılmaktadır. Burada dikkat edildiğinde bu eleman ilk eleman olduğundan esas köşegen üzerindeki elemandır ve dolayısıyla bu bölme işlemi sonucunda değeri 1 olmaktadır. Benzer şekilde matrisin değerinin değişmemesi için bu satırdaki her bir eleman B(I) değerine bölünmektedir. Bu işlem J değişkeninin değerinin N olmasına kadar devam etmektedir. J değişkeni N değerine ulaşınca, I değişkeninin değeri sorgulanmaktadır. Eğer I değişkeni N değerine ulaşmışsa A17. adımda J değişkenine I+1 değeri verilerek A18. adıma geçilmiştir ve burada bir K değişkeni tanımlanarak ilk değer olarak 1 atanmıştır. Buradaki amaç matrisin ilgili satırındaki esas köşegen üzerindeki elemanların değerini 0 yapmaktır. A16. adımda bu işlemlere geçilmiştir. Dikkat edildiğinde matrisin esas köşegeni üzerindeki elemandan sonra gelen elemanları sıfırlamak için ilgili elemandan kendisinin esas köşegeni üzerindeki elemanının çarpımı çıkarılmaktadır. Bu işlemler matrisin değerinin bozulmaması için her bir sütun için de yapılmaktadır. Bu kontrol K ve J değişkenleri vasıtasıyla yapılmaktadır. K değişkeninin değeri N olunca A22. adımda J değişkeninin değeri 1 arttırılarak A18. adıma gönderilip işlemlere devam edilmektedir. J değişkeni N değerine ulaşınca A24. adıma gidilerek I değişkenin değeri 1 arttırılıp diğer satırın esas köşegeni üzerindeki elemanın 1 yapılması için A11. adıma geri dönülmektedir. Bu işlemler I değişkeninin değeri N oluncaya kadar sürmektedir. I=N olunca istenilen sonuca ulaşıldığı için işlemlere son verilmektedir.

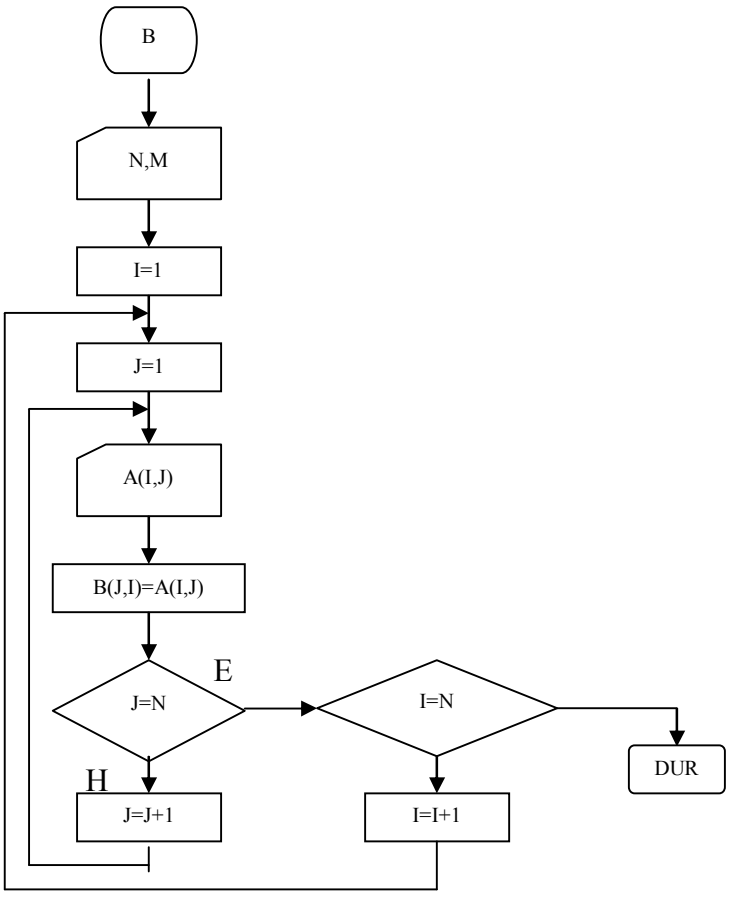
Örnek 4.7.8. NxN lik bir matrisin transpozesinin bulunmasını sağlayan algoritma ve akış şemasının oluşturulması.

Bir matrisin transpozesi olarak adlandırılan matris, mevcut matrisin satırları ile sütunlarının yer değiştirilmesiyle elde edilen matris olarak adlandırılmaktadır.

A1. Başla,
A2. N ve M' yi oku,
A3. I=1 al,
A4. J=1 al,
A5. A(I,J)' yi gir,
A6. B(J,I)=A(I,J) al,
A7. Eğer J=N ise A9. adıma git,
A8. J=J+1 al ve A5. adıma geri dön,
A9. Eğer I=N ise A11. adıma git,
A10. I=I+1 al ve A4. adıma geri dön,
A11. Dur.

Algoritmanın Pascal Dilindeki Yazılımı :

```
program ilsem;
var
        i,j,n : integer;
        a,b: array[1..20,1..20] of integer;
begin
        write (‘n sayısını gir : ‘); readln(n);
        for i:=0 to n-1 do begin
                for j:=0 to n-1 do readln(a[i,j]);
        end;
        for i:=0 to n-1 do begin
                for j:=0 to n-1 do
                        b[i,j]:=a[j,i];
        end;
        for i:=0 to n-1 do begin
                for j:=0 to n-1 do
                        writeln(b[i,j]);
        end;
end.
```



şekil 4.7.8. Bir matrisin transpozesini bulan akış şeması

Verilen örnekte A5. adımda girilen A matrisinin transpozesi A6. adımda bir B matrisine atanmaktadır. A(I,J) olarak girilen matris elamanı B matrisine B(I,J) olarak atanmaktadır. Burada önemli olan kısım (I,J)’ ye karşılık gelen elemanın B matrisinde

(J,I). eleman olarak dikkate alınmasıdır. Bunun amacı da I. satır ve J. sütundaki elemanın transpoze matrisinde J. satır I. sütuna karşılık gelmesidir. Bu işlemler A matrisinin girişi bitinceye kadar devam etmektedir.

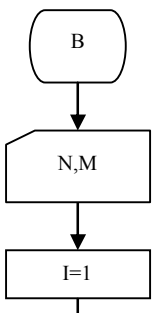
Örnek 4.7.9. Girilen iki matrisin toplamını bulan algoritma ve akış şemasının oluşturulması.

İki matrisin toplanabilmesi için bu matrisin satır ve sütun sayılarının aynı olması gerekir.

A1. Başla,
A2. N ve M' yi oku,
A3. I=1 al,
A4. J=1 al,
A5. A(I,J)' yi gir,
A6. B(I,J)' yi gir,
A7. T(J,I)=A(I,J)+B(I,J) al,
A8. Eğer J=M ise A10. adıma git,
A9. J=J+1 al ve A5. adıma geri dön,
A10. Eğer I=N ise A12. adıma git,
A11. I=I+1 al ve A4. adıma geri dön,
A12. Dur.

Algoritmanın Pascal Dilindeki Yazılımı :

```
program islem;
var
        i,j,n : integer;
        a,b,t: array[1..20,1..20] of integer;
begin
        write ('n sayısını gir : ');
        readln(n);
        for i:=0 to n-1 do begin
                for j:=0 to n-1 do begin
                        readln(a[i,j]);
                        readln(b[i,j]);
                        t[i,j]:=a[i,j]+b[i,j];
                end;
        end;
        for i:=0 to n-1 do begin
                for j:=0 to n-1 do
                        write(' ',t[i,j]);
                writeln;
        end;
end.
```



E

H E

H

Şekil 4.7.9. İki matrisin toplamını bulan akış şeması

Örnekte A5 ve A6. adımlara girilen matris elemanları toplanarak, A7. adımda T olarak adlandırılan bir toplam matrisine atanmaktadır. Bu işlemler A ve B olarak adlandırılan matrislerin tüm elemanlarının girilmesine kadar devam etmektedir.

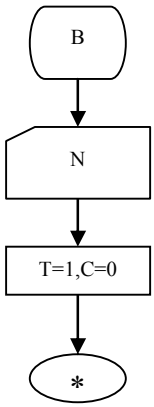
Örnek 4.7.10. Rasgele elemanlardan oluşan NxN lik bir matrisin tek ve çift elemanlarının sayısını bulan algoritma ve akış şemasının oluşturulması.

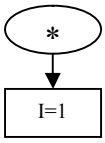
A1. Başla,
A2. N' yi oku,
A3. T=0, C=0 al,
A4. I=1 al,
A5. J=1 al,
A6. A(I,J)' yi gir,
A7. B=TAM(A(I,J)/2)*2 al,
A8. Eğer B=A(I,J) ise A10. adıma git,
A9. T=T+1 al ve A11. adıma git,
A10. C=C+1 al,
A11. Eğer J=N ise A13. adıma git,
A12. J=J+1 al ve A6. adıma geri dön,
A13. Eğer I=N ise A15. adıma git,
A14. I=I+1 al ve A55. adıma geri dön,
A15. T ve C' yi yaz,.

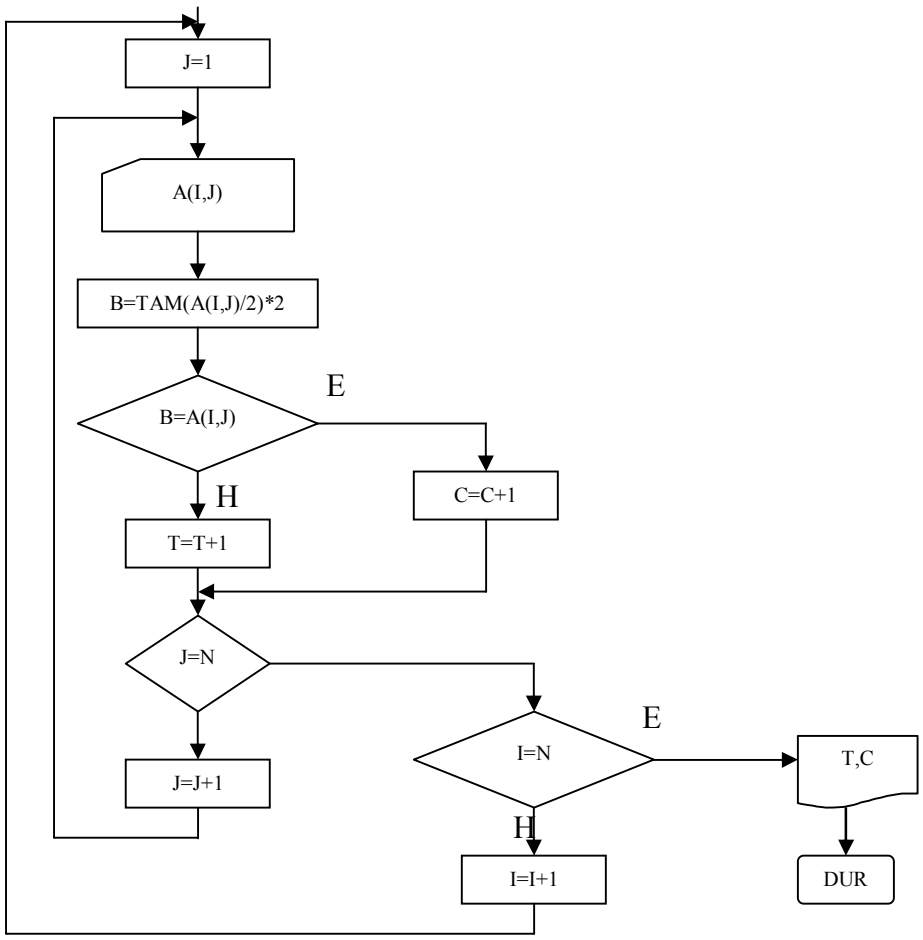
A16. Dur.

Algoritmanın Pascal Dilindeki Yazılımı :

```
program tek_cift_sayi;
var
        i,j,n,b,c,t : integer;
        a: array[1..20,1..20] of integer;
begin
        t:=0;
        c:=0;
        write (‘n sayısını gir : ‘);
        readln(n);
        for i:=1 to n do
        begin
                for j:=1 to n do
                begin
                        readln(a[i,j]);
                        b:=trunc(a[i,j]/2)*2;
                        if (b=a[i,j]) then
                                c:=c+1
                        else
                                t:=t+1;
                end;
        end;
        writeln(‘tek sayı sayısı : ‘, t);
        writeln(‘çift sayı sayısı : ‘,c);
end.
```







Şekil 4.7.10. Bir matrisin tek ve çift elemanlarının sayısını bulan akış şeması

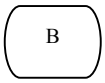
Rasgele girilen bir elemanın çift ya da tek olduğunu araştıkrmak için daha önce dizilerde yapılan işlemlerin aynısı burada da yapılmaktadır. Bunun için girilen elemanın yarısının tam kısmı alınarak tekrar 2 ile çarpımı gerçekleştirilmektedir. Elde edilen bu değer önceki değer ile aynı ise bu sayı iki ile tam bölünüyor anlamındadır. Bu durumda bu sayının çift sayı olması gerekmektedir. Aksi eşitlik olmayacağından girilen sayı tek sayı olacaktır. Örnekte A6. adımda girilen matris elemanı A7. adımda bu işlemlere tabi tutularak A8ş. adımda sorgulanmaktadır. Eğer şart gerçekleşiyorsa çift elemanları sayan C sayacı A10. adımda bir arttırılmaktadır. Aksi halde girilen eleman tek sayı olacağından A9. adımda tek sayılara sayan T sayacı bir arttırılmaktadır. Bu işlemler matrisin elemanlarının tümünün girişi yapılana kadar devam ederek giriş işlemleri tamamlandıktan sonra A15. adımda yazdırılıp işlemlere son verilmektedir.

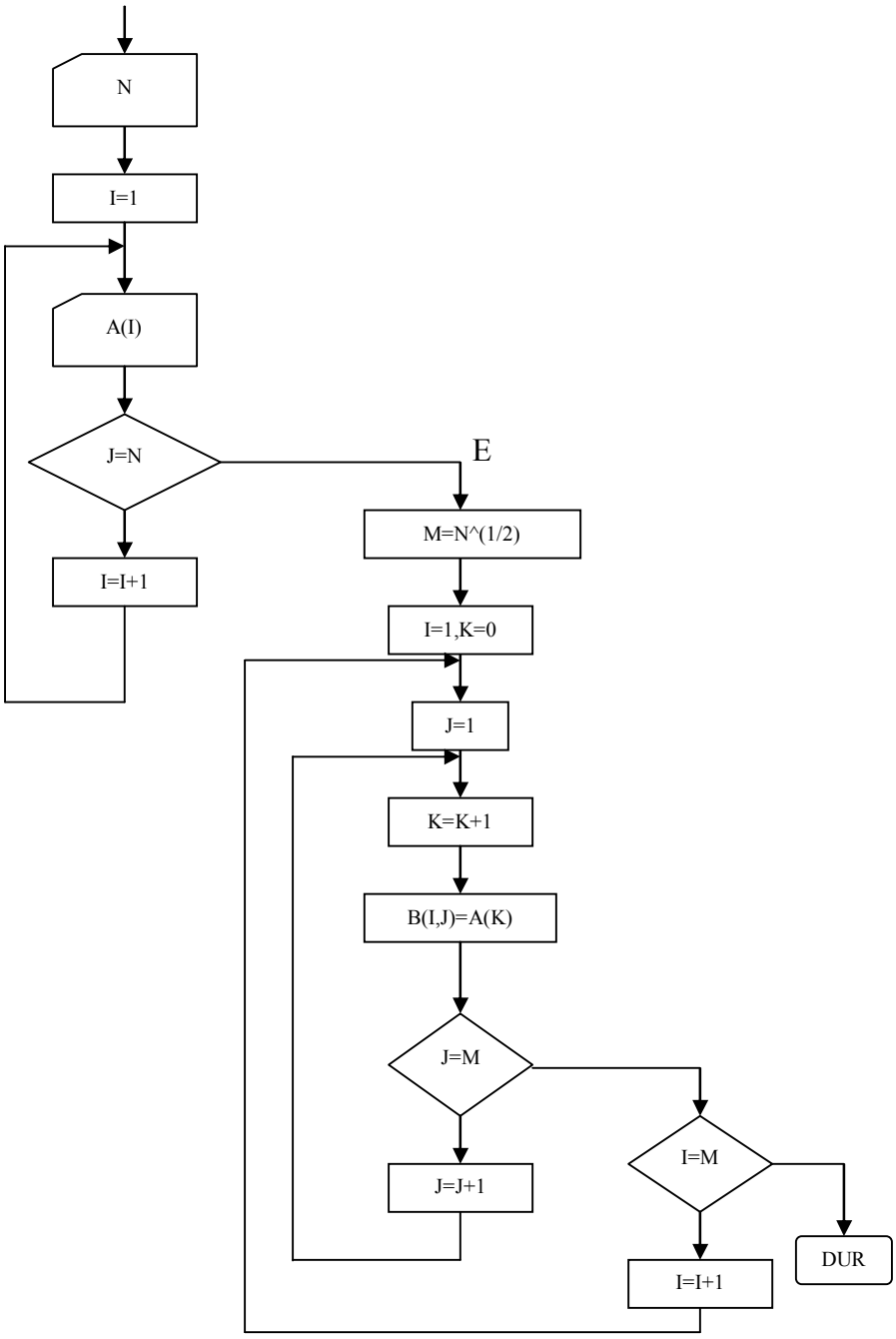
Örnek 4.7.11. N tam kare bir sayı olmak üzere N elemandan meydana gelen bir A dizisinin kare matris şekline dönüştürülmesini sağlayan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. N' yi oku,
A3. I=1 al,
A4. A(I)' yı gir,
A5. Eğer I=N ise A7. adıma git,
A6. I=I+1 al ve A4. adıma geri dön,
A7. M=N^(1/2) al,
A8. I=1 ve K=0 al,
A9. J=1 al,
A10. K=K+1 al,
A11. B(I,J)=A(K) al,
A12. Eğer J=M ise A14. adıma git,.
A13. J=J+1 al ve A11. adıma geri dön,
A14. Eğer I=M ise A16. adıma git,
A15. I=I+1 al ve A9. adıma geri dön,
A16. Dur.

Algoritmanın Pascal Dilindeki Yazılımı :

```
program diziye_yukleme;
var
        i,j,n,m,k : integer;
        b: array[1..20,1..20] of integer;
        a: array[1..20] of integer;
begin
        write ('n sayısını gir : ');
        readln(n);
        for i:=1 to n do
                readln(a[i]);
        m:=trunc(sqrt(n));
        k:=1;
        for i:=1 to m do
        begin
                for j:=1 to m do
                begin
                        b[i,j]:=a[k];
                        k:=k+1;
                        writeln(' ',b[i,j]);
                end;
                writeln;
        end;
end.
```





Şekil 4.7.11. Bir dizinin bir matrise dönüştürülmesini sağlayan akış şeması

İlk 6 adımda N elemanlı A dizisinin girişi yapıldıktan sonra, A7. adımda M değişkenine N sayısının karekökü yüklenmektedir. N sayısı bir tam kare sayı olduğundan M değeri de bir tam sayı olacaktır. N elemandan meydana gelen dizi sonuçta MxM' lik bir matrise dönüşecektir. Bunun için A8. adımda I ve K değişkenleri tanımlanarak ilk değerleri atanmıştır. A9. adımda J değişkeni tanımlanarak ilk değer olarak 1 atanmıştır. A10. adımda K değişkeninin değeri bir arttırılarak A11. adımda B olarak tanımlanan matrisin (I,J). elemanı yerine A dizisinin K. elemanı atanmıştır. J değişkeninin değeri M olunca I değişkeni arttırılarak aynı işlemlere devam edilmektedir. Sonuçta I değişkeninin değeri M sayısına ulaşınca atama işlemleri yapılmış olacağından işlemlere son verilmektedir.

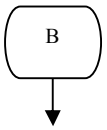
Örnek 4.7.12. NxN lik bir A matrisinin simetrik olup olmadığını araştıran algoritma ve akış şemasının oluşturulması.

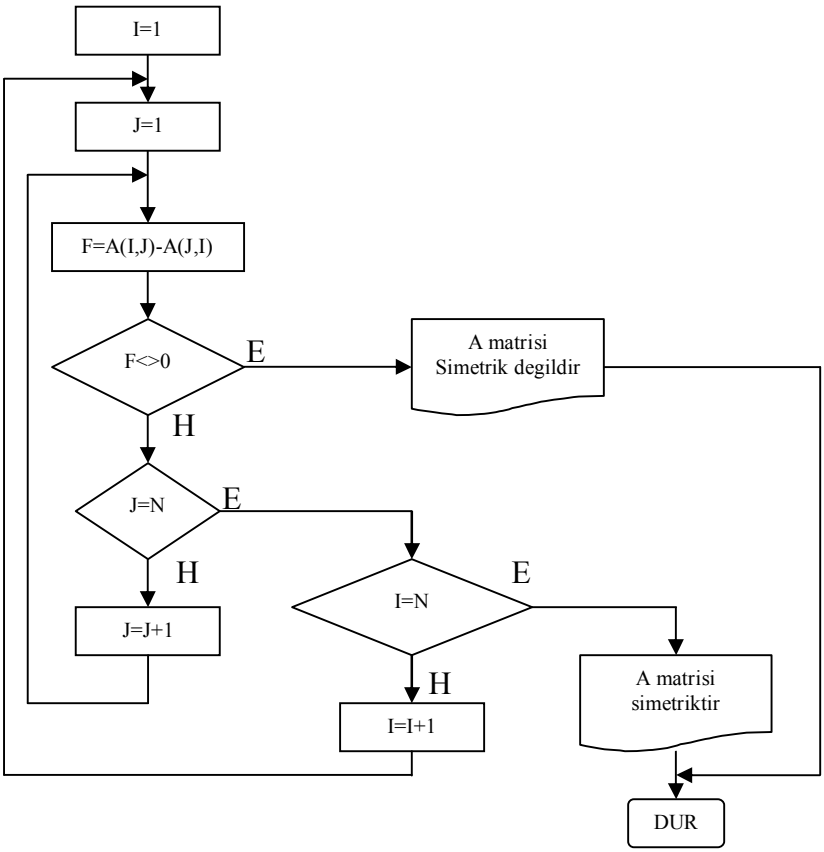
Bir matrisin simetrik olma koşulu, bütün I ve J' ler için A(I,J)=A(J,I) olmasıdır.

A1. Başla,
A2. I=1 al,
A3. J=1 al,
A4. F=A(I,J)-A(J,I) al,
A5. Eğer F<>0 ise A12. adıma git,
A6. Eğer J=N ise A8. adıma git,
A7. J=J+1 al ve A4. adıma geri dön,
A8. Eğer I=N iseA10. adıma git,
A9. I=I+1 al ve A3. adıma geri dön,
A10. A matrisi simetriktir yaz,
A11. A13. adıma git,
A12. A matrisi simetrik değildir yaz,
A13. Dur.

Algoritmanın Pascal Dilindeki Yazılımı :

```
program simetri;
var
        i,j,n,f : integer;
        a: array[1..20,1..20] of integer;
        s: string;
begin
        write (‘n sayısını gir : ‘);
        readln(n);
        for i:=1 to n do
        begin
                for j:=1 to n do
                        readln(a[i,j]);
        end;
        s:=’simetri’;
        for i:=1 to n do
        begin
                for j:=1 to n do
                begin
                        f:=a[i,j]-a[j,i];
                        if (f<>0) then s:=’simetri değil’;
                end;
        end;
        writeln(s);
end.
```





Şekil 4.7.12. NxN' lik bir matrisin simetrik olup olmadığını araştıran akış şeması

Daha önceden elemanlarının girişi yapılmış olan bir A matrisinin simetrik olup olmadığı araştırılmaktadır. Bunun için A4. adımda F olarak adlandırılan bir değişken tanımlanarak bu değişkene A matrisinin (I,J). elemanı ile (J,I). elemanı farkı yüklenmektedir. A5. adımda F değişkeni sorgulanmaktadır. Eğer F değişkeninin değeri sıfırdan farklı ise matris simetrik olmayacaktır. Bu durumda A12. adıma gidilerek matrisin simetrik olmadığı yazdırılıp işlemlere son verilmektedir. Aksi halde J değişkeninin değeri arttırılarak yeniden A4. adıma dönülerek F değeri hesaplanmaktadır. Bu işlemler A matrisinin tüm elemanlarının karşılaştırılması yapılana kadar devam etmektedir. Sonuçta her I,J için A(I,J)=A(J,I) şartı gerçekleşiyorsa bu durumda A10. adım devreye girerek A matrisinin simetrik olduğu yazdırılıp işlemlere son verilmektedir.

Örnek 4.7.13. NxM ve MxK' lik iki matrisin çarpımını bulan algoritma ve akış şemasının oluşturulması.

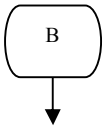
İki matrisin çarpılabilmesi için gerekli kural birinci matrisin sütun sayısı ile ikinci matrisin satır sayısının eşit olmasıdır.

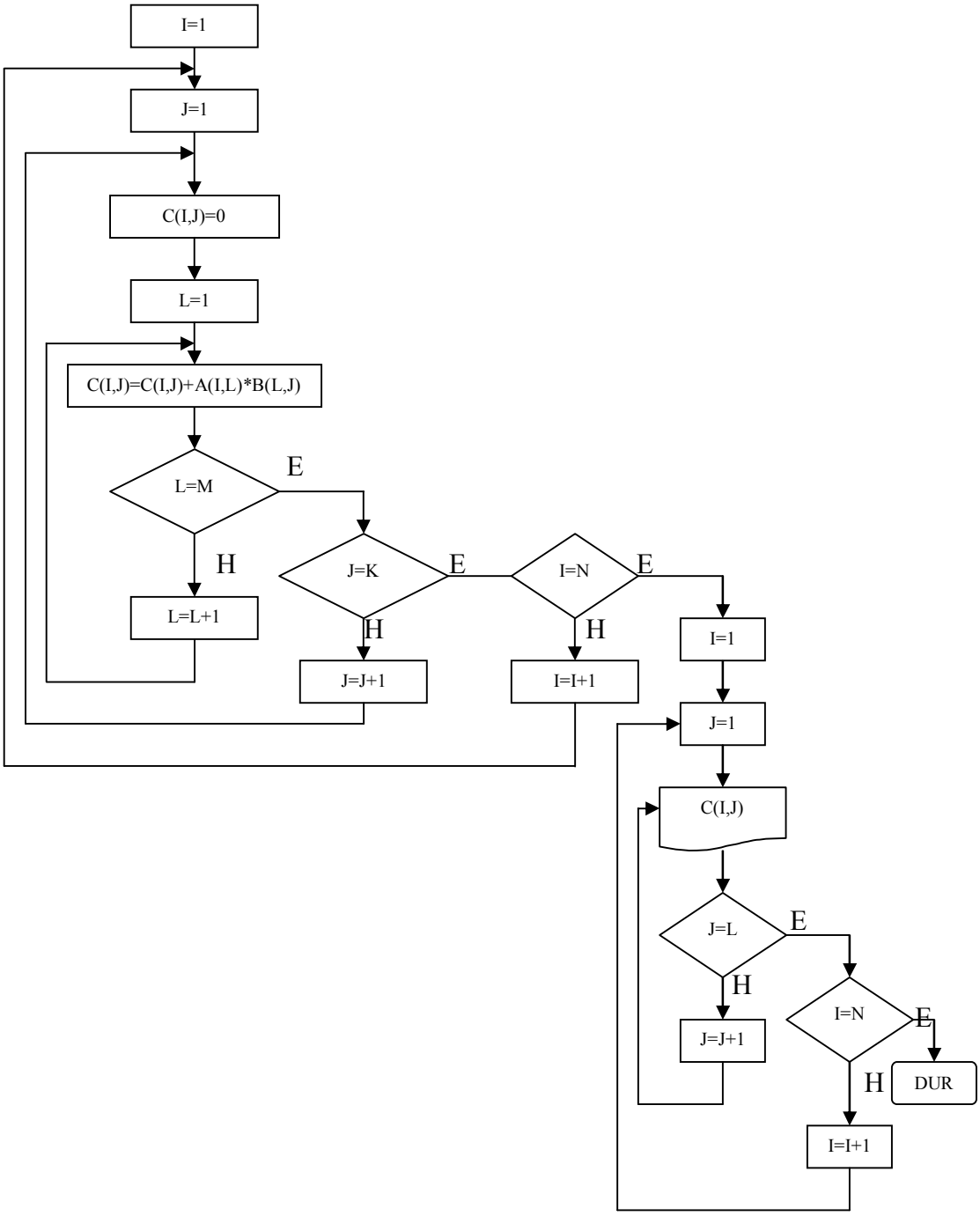
A1. Başla,
A2. I=1 al,
A3. J=1 al,

A4. C(I,J)=0 al,
A5. L=1 al,
A6. C(I,J)=C(I,J)+A(I,L)*B(L,J) al,
A7. Eğer L=M ise A9. adıma git,
A8. L=L+1 al ve A6. adıma geri dön,
A9. Eğer J=K ise A11. adıma git,
A10. J=J+1 al ve A4. adıma geri dön,
A11. Eğer I=N ise A13. adıma git,
A12. I=I+1 al ve A3. adıma geri dön,
A13. I=1 al,
A14. J=1 al,
A15. C(I,J)' yi yaz,
A16. Eğer J=L ise A18. adıma git,
A17. J=J+1 al ve A15. adıma geri dön,
A18. Eğer I=N ise A20. adıma git,
A19. I=I+1 al ve A14. adıma geri dön,
A20. Dur.

Algoritmanın Pascal Dilindeki Yazılımı :

```
program matris_carpim;
var
        i,j,k,l,m,n : integer;
        a,b,c: array[1..20,1..20] of integer;
begin
        write ('n sayısını gir : ');       readln(n);
        write('m sayısını gir : ');        readln(m);
        write('k sayısını gir : ');        readln(k);
        for i:=1 to n do begin
                for j:=1 to m do readln(a[i,j]);
        end;
        for i:=1 to m do begin
                for j:=1 to k do readln(b[i,j]);
        end;
        for i:=1 to n do begin
                for j:=1 to k do begin
                        c[i,j]:=0;
                        for l:=1 to m do
                                c[i,j]:=c[i,j]+a[i,l]*b[l,j];
                end;
        end;
        for i:=1 to n do begin
                for j:=1 to k do  write(' ',c[i,j]);
                writeln;
        end;
        readln;
end.
```





Şekil 4.7.13. İki matrisin çarpımını bulan akış şeması

İki matrisin çarpımında ilk aşamada birinci matrisin ilk satırının elemanları ile ikinci matrisin ilk sütununun elemanları karşılıklı olarak çarpılarak toplanır. Böylece çarpım matrisinin ilk elemanı elde edilmiş olur. Benzer şekilde ilk matrisin ilk satırı ile ikinci matrisin diğer satırları da çarpılıp toplanarak çarpım matrisinin ilk satır elemanlarının bulunması sağlanır. Aynı mantıkla birinci matrisin diğer satır elemanları ikinci matrisin sütun elemanlarıyla çarpılıp toplanarak çarpım matrisi elde edilmiş olur. Örnekte bu mantıkta hareket edilerek matrisin çarpımları gerçekleştirilmektedir. Bunun için, A2 ve A3. adımlarda tanımlanan I ve J değişkenlerine ilk değer olarak 1 değeri verilerek işlemlere başlanmıştır. A4. adımda çarpım matrisinin I ve J elemanına karşılık

gelen değerine 0 atanmıştır. Bunun anlamı, çarpım matrisi olan C matrisinin I,J ye karşılık gelen elemanının bulunması için birinci matrisin ilk satır ile ikinci matrisin ilk sütununun karşılıklı elemanlarının çarpımlarının toplamı C matrisinin I,J ye karşılık gelen bir önceki değeriyle birlikte toplanarak çarpım matrisinin ilk elemanını elde etmek içindir. A5. adımda bir L değişkeni tanımlanarak ilk değer olarak 1 atanmıştır. A6. adımda C(I,J)'ye ilk çarpım işlemi atanmıştır. Bundan sonraki aşamalarda J, L ve I değişkenleri sorgulanarak alacakları son değerlere kadar bu işlemlere devam edilerek çarpım matrisi elde edilmiştir. Son olarak da elde edilen çarpım matrisi yazdırılarak işlemlere son verilmiştir.

Yukarıdaki algoritmayı bir örnekle değerlendirelim.

$$A=\begin{pmatrix} 1 & 2 & -1 \\ 3 & 1 & 0 \\ 2 & 4 & 5 \end{pmatrix} \quad \text{ve} \quad B=\begin{pmatrix} 2 & 1 \\ 0 & 3 \\ 4 & 2 \end{pmatrix} \quad \text{olsun}$$

I=1, j=1 alınarak, C(I,J)=C(1,1)=0 alınır,
L=1 alınarak, C(1,1)=C(1,1)+A(1,1)*b(1,1)=0+1*2=2 elde edilir,
L=L+1=2 alınarak, C(1,1)=2+2*0=2 elde edilir,
L=3 için, C(1,1)=2+(-1)*4=-2 elde edilir,
L=3 olduğundan, J=J+1=2 alınarak, C(1,2)=0 ve L=1 alınır,
C(1,2)=0+1*1=1 elde edilir,
L=L+1=2 alınarak, C(1,2)=1+2*3=7 elde edilir,
L=3 için, C(1,2)=7+(-1)*2=5 elde edilir,
L=3, J=2 olduğundan I=I+1=2 alınır, J=1 ve C(2,1)=0 alınarak,
L=1 değeri için C(2,1)=0+3*2=6 elde edilir,
L=L+1=2 için, C=(2,1)=6+1*0=6 elde edilir,
L=3 için, C(2,1)=6+0*4=6 bulunur,
L=3 olduğundan, J=2 alınarak, C(2,2)=0 ve L=1 için C=(2,2)=0+3*1=3 elde edilir,
L=2 için, C(2,2)=3+1*3=6 bulunur,
L=3 için, C(2,2)=6+0*2=6 elde edilir.
J=2 olduğundan, I=I+1=3 alınarak, J=1 ve C(3,1)=0 alınarak,
L=1 için, C=(3,1)=0+2*2=4 bulunur,
L=2 için, C(3,1)=4+4*0=4 elde edilir,
L=3 için, C(3,1)=4+5*4=24 elde edilir,
L=3 olduğundan, J=JJ+1=2 alınarak, C(3,2)=0 ve L=1 alınarak, C(3,2=)0+2*1 elde edilir,
L=2 için, C(3,2)=2+4*3=14 elde edilir,
L=3 için, C(3,2)=14+5*2=24 bulunur.

Sonuç olarak, elde edilen çarpım matrisi,

$$C= \begin{pmatrix} -2 & 5 \\ 6 & 6 \\ 24 & 24 \end{pmatrix}$$

şeklinde elde edilmiş olur.

Örnek 4.7.14. NxN lik bir kare matrisin tersinin bulunmasını sağlayan algoritma ve akış şemasının oluşturulması.

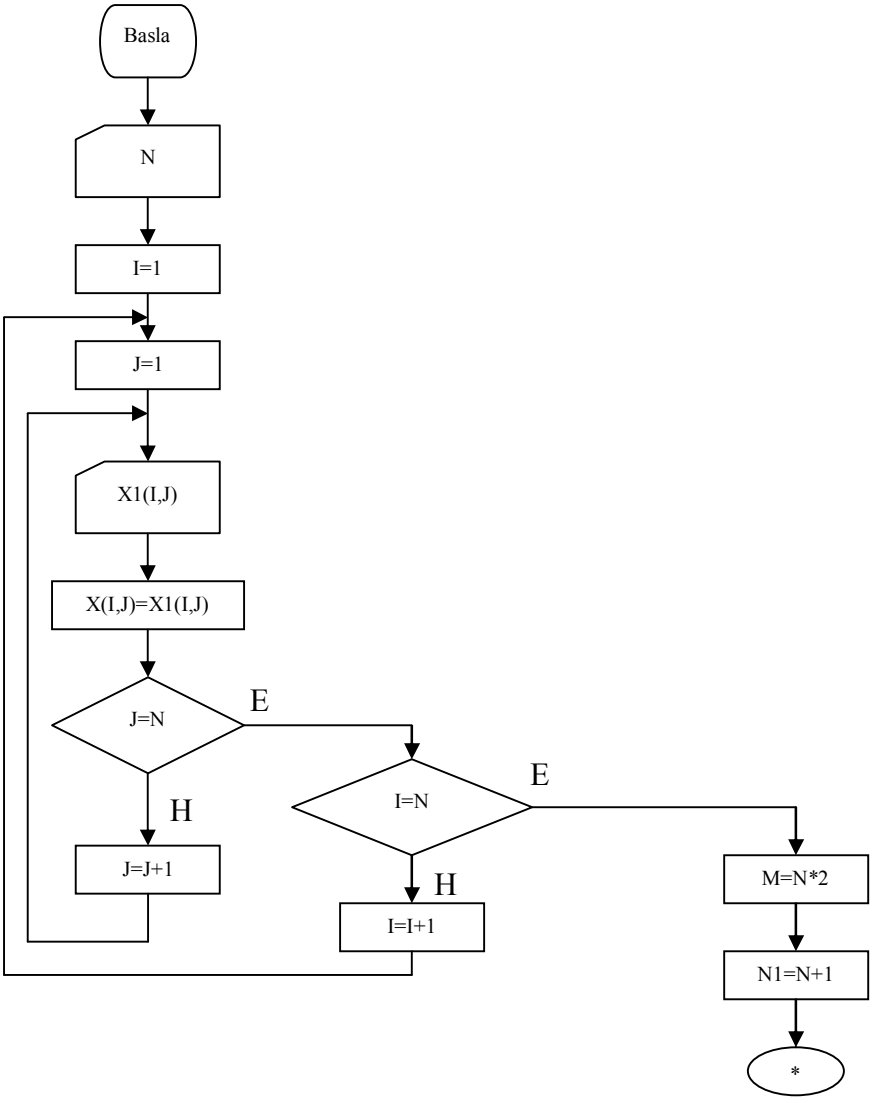
A1. Başla,
A2. N' yi gir,
A3. I=1 al,
A4. J=1 al,
A5. X(I,J)' yi gir,
A7. Eğer J=N ise A9. adıma git,
A8. J=J+1 al ve A5. adıma geri dön,
A9. Eğer I=N ise A11. adıma git,
A10. I=I+1 al ve A4. adıma geri dön,
A11 . M=N*2 al,
A12. N1=N+1 al,
A13. I=1 al,
A14. J=N1 al,
A15. J1=J-I al,
A16. Eğer J1-N<0 ise X(I,J)=0 al,
A17. Eğer J1-N=0 ise X(I,J)=1 al,
A18. Eğer J=M ise A20. adıma git,
A19. J=J+1 al ve A15. adıma geri dön,
A20. Eğer I=N ise A22. adıma git,
A21. I=I+1 al ve A14. adıma geri dön,
A22. K=1 al,
A23. K1=K+1 al,
A24. K3=K+N-1 al,
A25. Eğer X(K,K)<>0 ise A30. adıma git,
A26. I1=K+1 al,
A27. Eğer X(I1,K)<>0 ise A34. adıma git,
A28. I1=I1+1 al,
A29. A25. adıma geri dön,
A30. J=1 al,
A31. X(K,J)=X(K,J)+X(I1,J) al,
A32. Eğer J=M ise A34. adıma git,
A33. J=J+1 al ve A31. adıma geri dön,
A34. J=1 al,
A35. X(K+N,J)=X(K,J)/X(K,K) al,

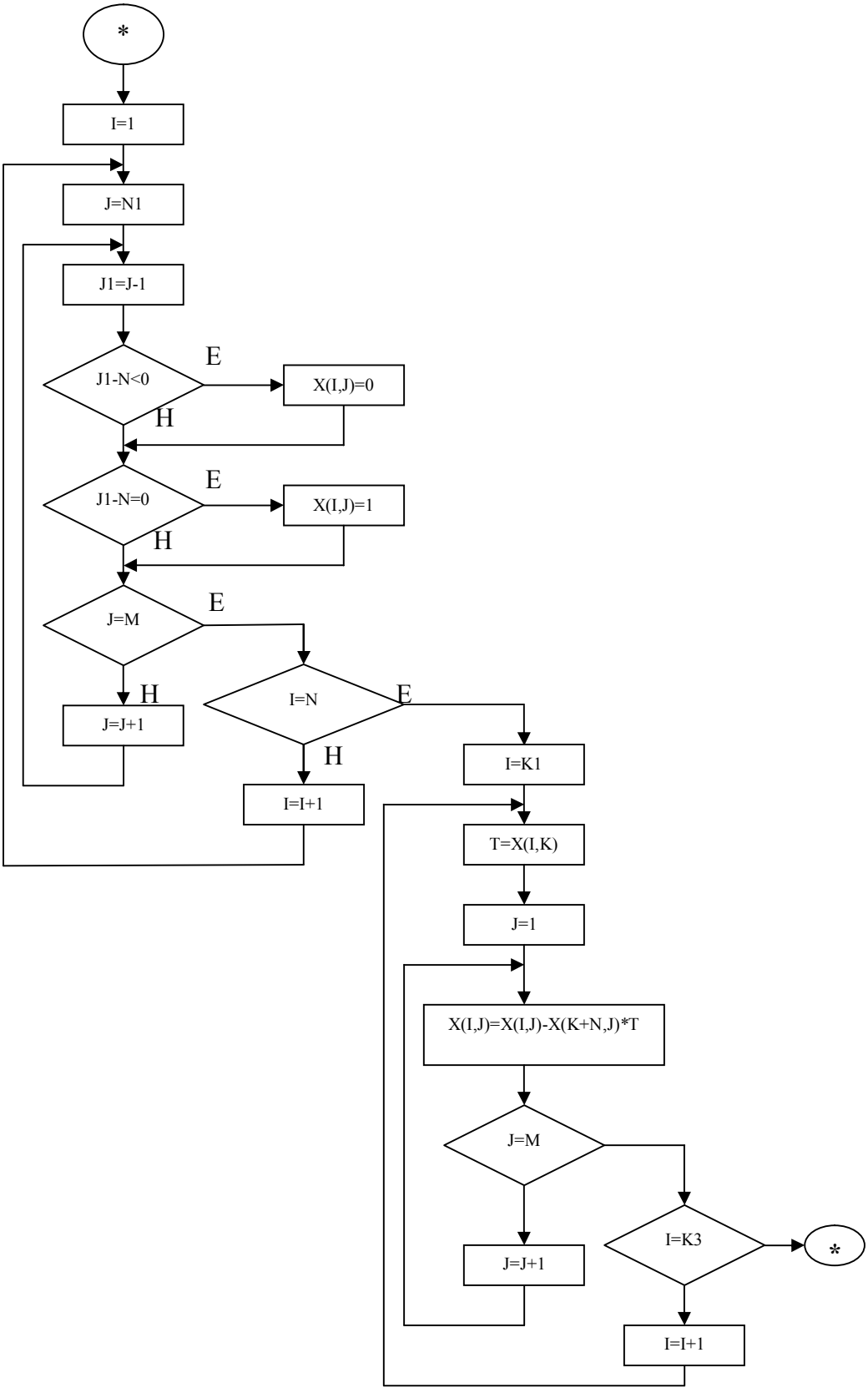
A36. Eğer J=M ise A38. adıma git,
A37. J=J+1 al ve A35. adıma geri dön,
A38. I=K1 al,
A39. T=X(I,K) al,
A40. J=1 al,
A41. X(I,J)=X(I,J)-X(K+N,J)*T al,
A42. Eğer J=M ise A44. adıma git,
A43. J=J+1 al ve A41. adıma geri dön,
A44. Eğer I=K3 ise A46. adıma git,
A45. I=I+1 al ve A39. adıma geri dön,
A46. I=1 al,
A47. J=1 al,
A48. X(I,J)' yi yaz,
A49. Eğer J=oN ise A51. adıma git,
A50. J=J+1 al ve A48. adıma geri dön,
A51. Eğer I=N ise Dur.
A52. I=I+1 al ve A47. adıma geri dön.

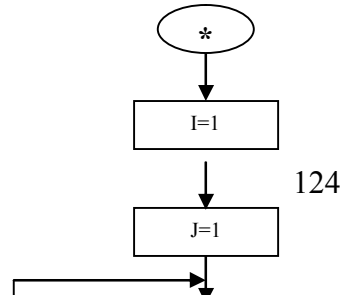
Algoritmanın Pascal Dilindeki Yazılımı :

```
program matris_tersi;
var
        n1,k1,k3,i,i1,j1,j,k,m,n : integer;
        x,x1: array[1..40,1..40] of real;
        t:real;
label 10,20,30;
begin
        write ('n sayısını gir : ');       readln(n);
        for i:=1 to n do begin
                for j:=1 to n do begin
                        readln(x1[i,j]);
                        x[i,j]:=x1[i,j];
                end;
        end;
        m:=n*2;
        n1:=n+1;
        for i:=1 to n do begin
                for j:=n1 to m do begin
                        j1:=j-i;
                        if (j1-n<0) then x[i,j]:=0;
                        if (j1-n=0) then x[i,j]:=1;
                end;
        end;
        for k:=1 to n do begin
                k1:=k+1;
                k3:=k+n-1;
                10:  if (x[k,k]<>0) then goto 20;
                        i1:=k+1;
                        if (x[i1,k] <>0=then goto 30;
                        i1:=i1+1;
```

```
                goto 10;
        20: for j:=1 to m do
                x[k,j]:=x[k,j]+x[i1,j];
        30: for j:=1 to m do
                x[k+n,j]:=x[k,j]/x[k,k];
        for i:=k1 to k3 do begin
                t:=x[i,k];
                for j:=1 to m do
                        x[i,j]:=x[i,j]-x[k+n,j]*t;
        end;
end;
for i:=1 to n do begin
        for j:=1 to n do
                write(' ',x(i+n,j+n]:3:2);
        writeln;
end;
readln;
end.
```




*
I=1
J=N1
J1=J-1
J1-N<0
E
X(I,J)=0
H
J1-N=0
E
X(I,J)=1
H
J=M
E
H
J=J+1
I=N
E
H
I=I+1
I=K1
T=X(I,K)
J=1
X(I,J)=X(I,J)-X(K+N,J)*T
J=M
J=J+1
I=K3
*
I=I+1


*
I=1
J=1

Şekil 4.7.14. Bir kare matrisin tersini bulan akış şeması

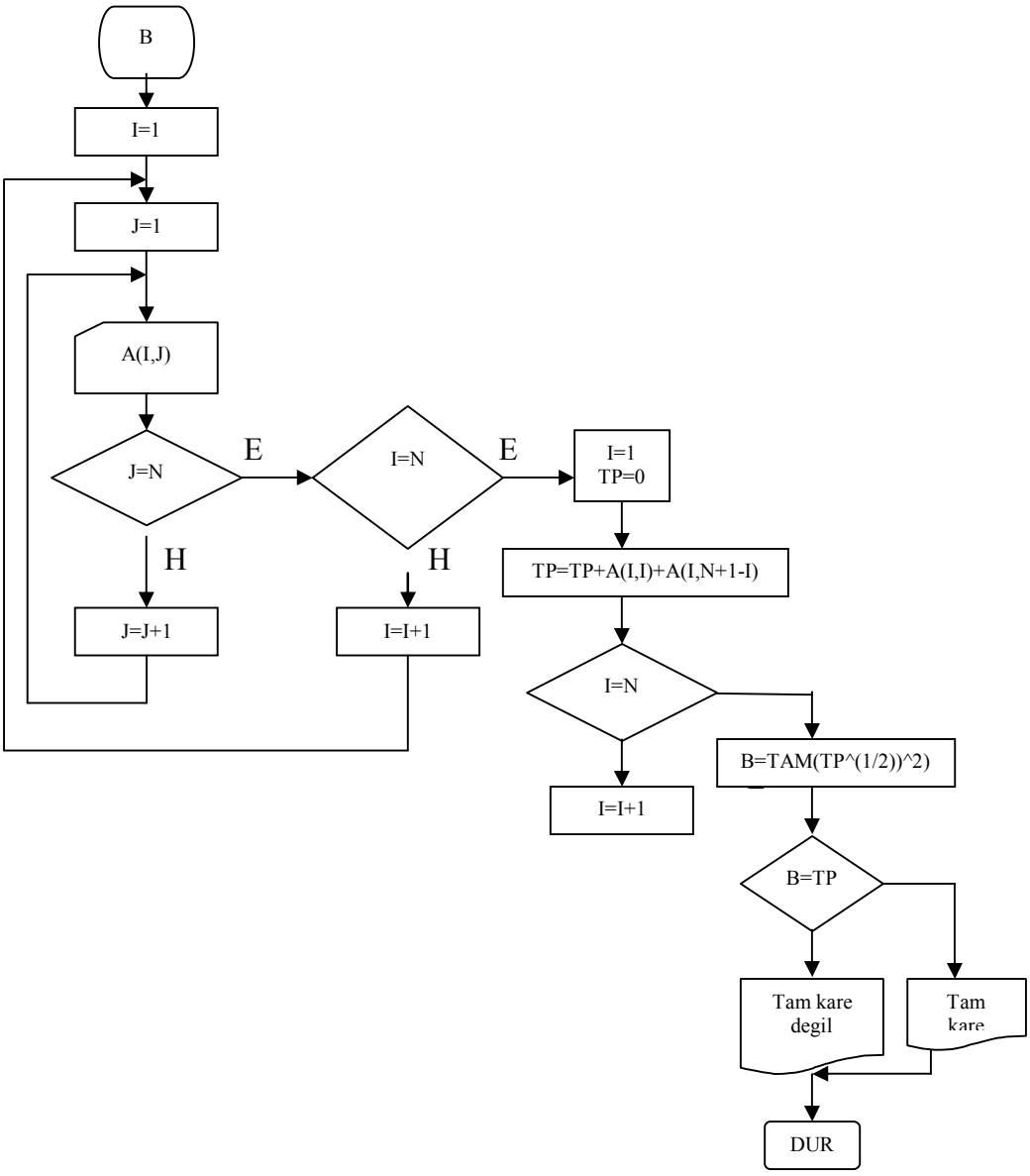
Örnek 4.7.15. NxN lik bir tamsayı matrisinin esas köşegeni ile yedek köşegeni üzerindeki elemanlarının toplamının bir tam kare sayı olup olmadığını araştıran algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. I=1 al,
A3. J=1 al,
A4. A(I,J)' yi gir,
A5. Eğer J=N ise A7. adıma git,
A6. J=J+1 al ve A4. adıma geri dön,
A7. Eğer I=N ise A9. adıma git,
A8. I=I+1 al ve A3. adıma geri dön,
A9. I=1, TP=0 al,
A10. TP=TP+A(I,I)+A(I,N+1-I) al,
A11. Eğer I=N ise A13. adıma git,
A12. I=I+1 al ve A10. adıma geri dön,
A13. B=TAM(TP^(1/2))^2 al,
A14. Eğer B=TP ise A16. adıma git,.
A15. "tam kare değil" yaz ve A17. adıma git,
A16. "tam kare" yaz,
A17. Dur.

Algoritmanın Pascal Dilindeki Yazılımı :

```
program kosegen;
var

        i,j,b,c,tp,n : integer;
        a: array[1..20,1..20] of integer;
```

```
begin
        write (‘n sayısını gir : ‘);readln(n);
        for i:=1 to n do begin
                for j:=1 to n do readln(a[i,j]);
        end;
        tp:=0;
        for i:=1 to n do
                tp:=tp+a[i,i]+a[i,n+1-i];
                b:=trunc(sqrt(tp));      c:=b*b;
        if (c=tp) then writeln(‘tamkare’)
        else  writeln(‘tamkare değil’);readln;
end.
```



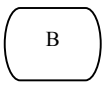
Şekil 4.7.15. NxN’ lik bir tamsayı matrisinin esas köşegeni ile yedek köşegeni üzerindeki elemanlarının toplamının bir tam kare sayı olup olmadığını araştıran akış şeması

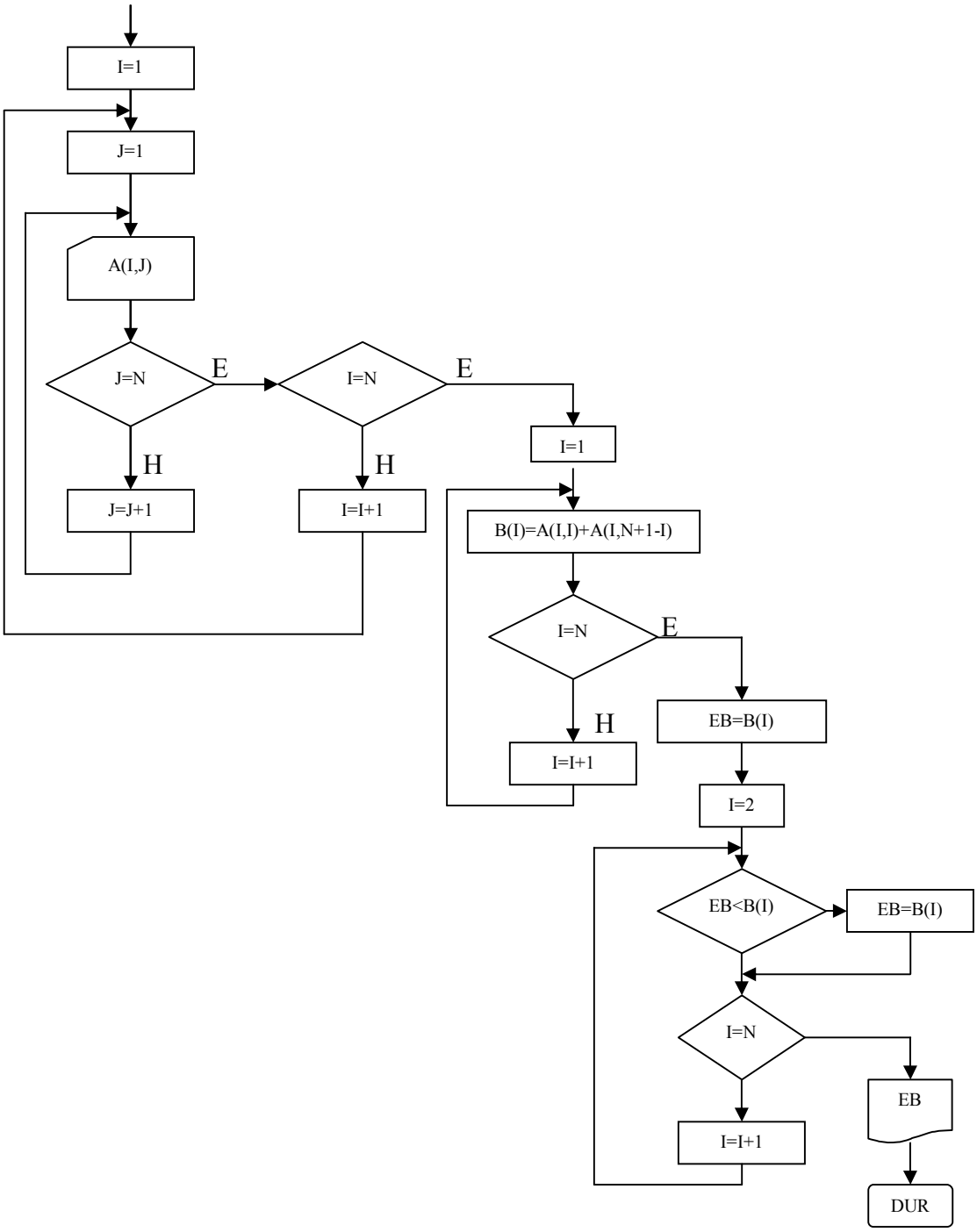
Örnek 4.7.16. NxN lik bir sayı matrisinin esas köşegeni ile yedek köşegeni üzerindeki karşılıklı elemanların çarpımlarını bir diziye yükleyerek bu dizinin en büyük elemanını bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. I=1 al,
A3. J=1 al,
A4. A(I,J)' yi gir,
A5. Eğer J=N ise A7. adıma git,
A6. J=J+1 al ve A4. adıma geri dön,
A7. Eğer I=N ise A9. adıma git,
A8. I=I+1 al ve A3. adıma geri dön,
A9. I=1 al,
A10. B(I)=A(I,I)*A(I,N+1-I) al,
A11. Eğer I=N ise A13. adıma git,
A12. I=I+1 al ve A10. adıma geri dön,
A13. EB=B(1) al,
A14. I=2 al,.
A15. Eğer EB<B(I) ise EB=B(I) al,
A16. Eğer I=N ise A18. adıma git,
A17. I=I+1 al ve A15. adıma geri dön,
A18. EB'yi yaz,
A19. Dur.

Algoritmanın Pascal Dilindeki Yazılımı :

```
program enbuyuk;
var
        i,j,eb,n : integer;
        a: array[1..20,1..20] of integer;
        b: array[1..20] of integer;
begin
        write (‘n sayısını gir : ‘);
        readln(n);
        for i:=1 to n do
        begin
                for j:=1 to n do
                        readln(a[i,j]);
        end;
        for i:=1 to n do
                b[i]:=a[i,i]*a[i,n+1-i];
                eb:=b[1];
                for i:=2 to n do
                 begin
                        if (eb<b[i]) then
                                eb:=b[i];
                end;
        writeln(’en büyük eleman ‘,eb);
        readln;
end.
```





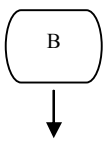
Şekil 4.7.16. NxN' lik bir sayı matrisinin esas köşegeni ile yedek köşegeni üzerindeki karşılıklı elemanların çarpımlarını bir diziye yükleyerek bu dizinin en büyük elemanını bulan akış şeması

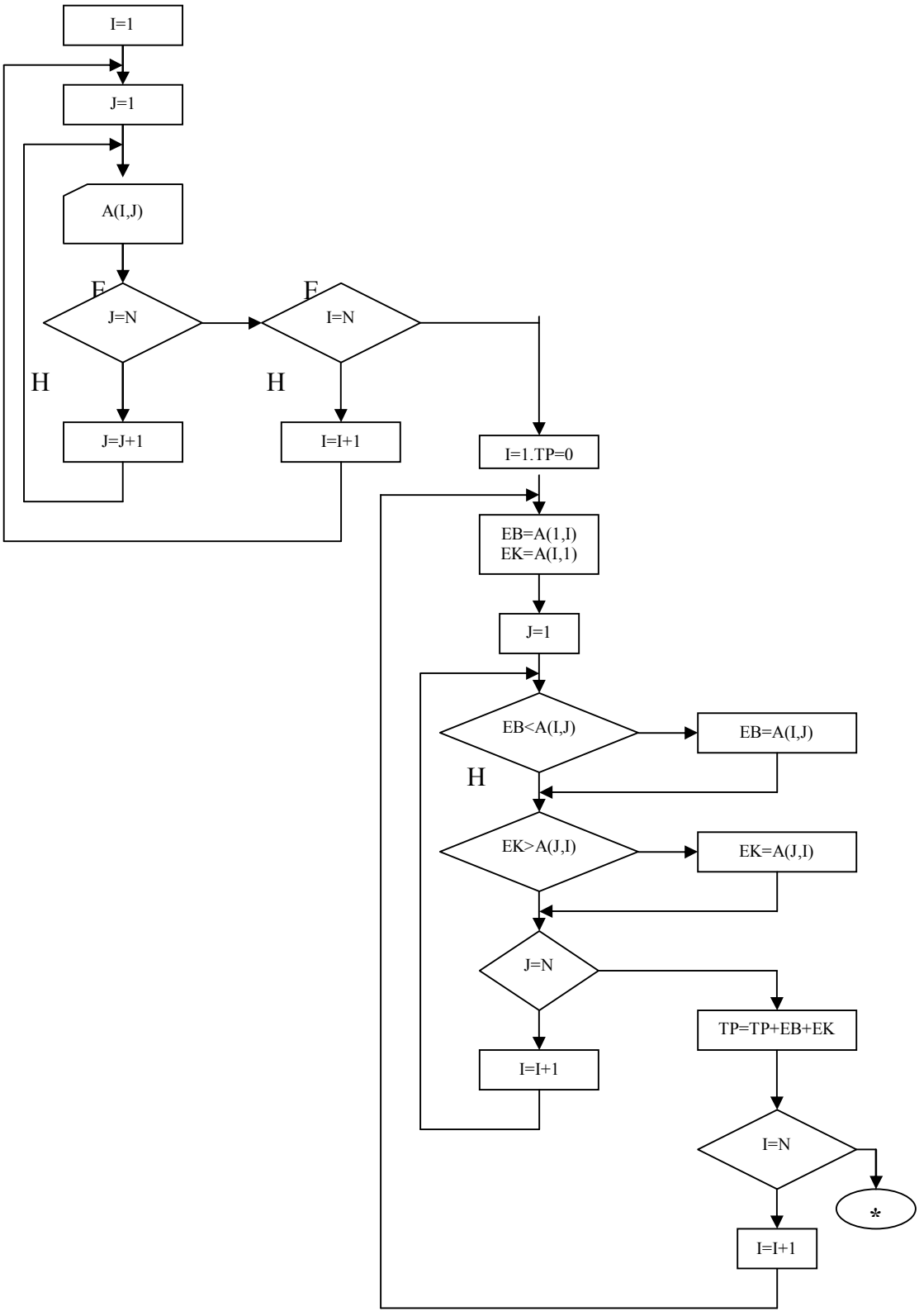
Örnek 4.7.17. NxN lik bir sayı matrisinin her satırının en büyük elemanları ile her sütununun en küçük elemanlarının karşılıklı toplamlarının bir tamkare sayı olup olmadığını araştıran algoritma ve akış şemasının oluşturulması.

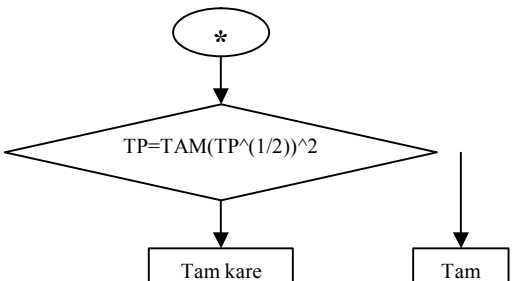
A1. Başla,
A2. I=1 al,
A3. J=1 al,
A4. A(I,J)' yi gir,
A5. Eğer J=N ise A7. adıma git,
A6. J=J+1 al ve A4. adıma geri dön,
A7. Eğer I=N ise A9. adıma git,
A8. I=I+1 al ve A3. adıma geri dön,
A9. I=1 ve TP=0 al,
A10. EB=A(I,I) ve EK=A(I,1) al,
A11. J=1 al,
A12. Eğer EB<A(I,J) ise EB=A(I,J) al,
A13. Eğer EK>A(J,I) ise EK=A(J,I) al,
A14. Eğer J=N ise A16. adıma git,
A15. J=J+1 al ve A12. adıma geri dön,
A16. TP=TP+EB+EK al,
A17. Eğer I=N ise A19. adıma git,
A18. I=I+1 al ve A10. adıma geri dön,
A19. Eğer TP=TAM(TP^(1/2))^2 ise A21. adıma git,
A20. "tam kare değil" yaz ve A22. adıma git,
A21. "tam kare" yaz,
A22. Dur.

Algoritmanın Pascal Dilindeki Yazılımı :

```
var
        i,j,eb,tp,n,ek : integer;
        a: array[1..20,1..20] of integer;
begin
        write ('n sayısını gir : ');       readln(n);
        for i:=1 to n do begin
                for j:=1 to n do readln(a[i,j]);
        end;
        tp:=0;
        for i:=1 to n do begin
                eb:=a[1,i];      ek:=a[i,1];
                for j:=1 to n do begin
                        if (eb<a[i,j]) then eb:=a[i,j];
                        if (ek>a[j,i]) then ek:=a[j,i];
                end;
                tp:=tp+eb+ek;
        end;
        if (tp=sqr(trunc(sqrt(tp)))) then writeln('tam kare ')
        else writeln('tam kare değil');
        readln;
end.
```




I=1
J=1
A(I,J)
J=N
F
H
J=J+1
I=N
F
H
I=I+1
I=1.TP=0
EB=A(1,I)
EK=A(I,1)
J=1
EB<A(I,J)
EB=A(I,J)
H
EK>A(J,I)
EK=A(J,I)
J=N
I=I+1
TP=TP+EB+EK
I=N
*
I=I+1


*
TP=TAM(TP^(1/2))^2
Tam kare
Tam

Şekil 4.7.17. Matriste tamkare algoritmasına ilişkin akış şeması

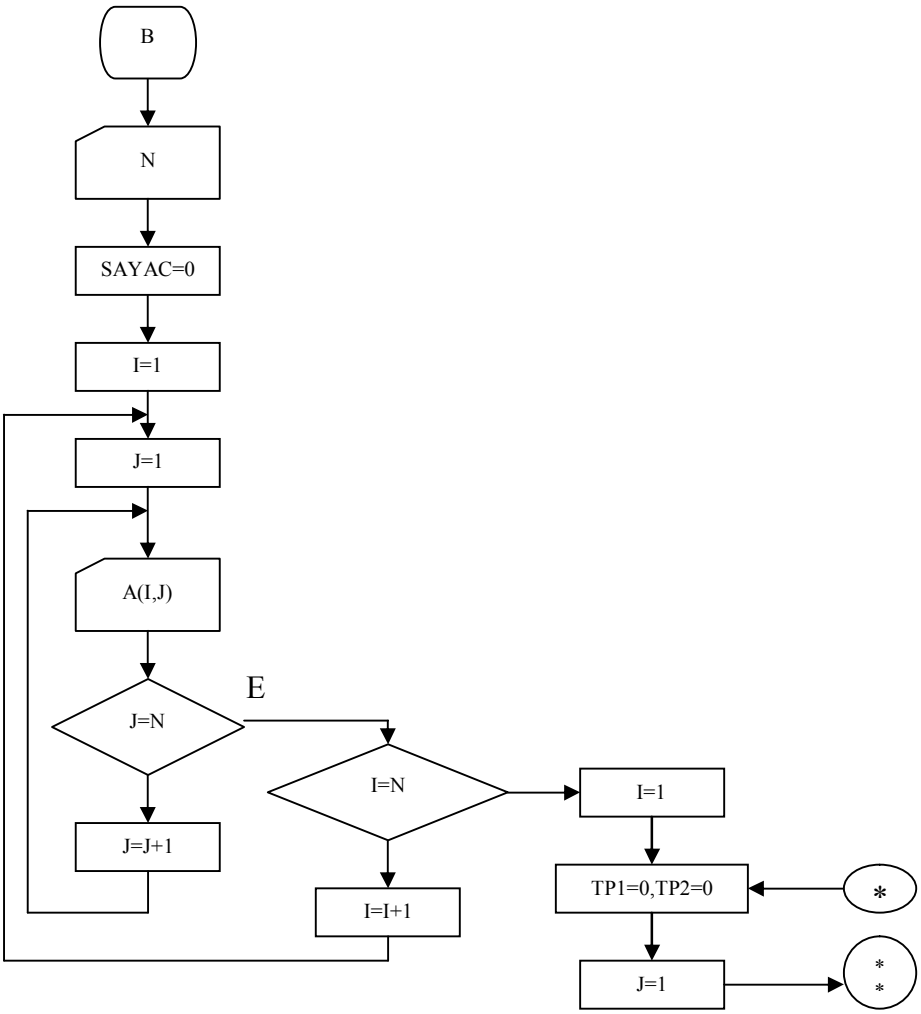
Örnek 4.7.18. NxN' lik bir sayı matrisinin her bir satırının ayrı ayrı ortalamaları ile bu matrisin transpozunun her bir satırının ayrı ayrı ortalamalarından karşılıklı olarak eşit olanların sayısını bulan algoritma ve akış şeması

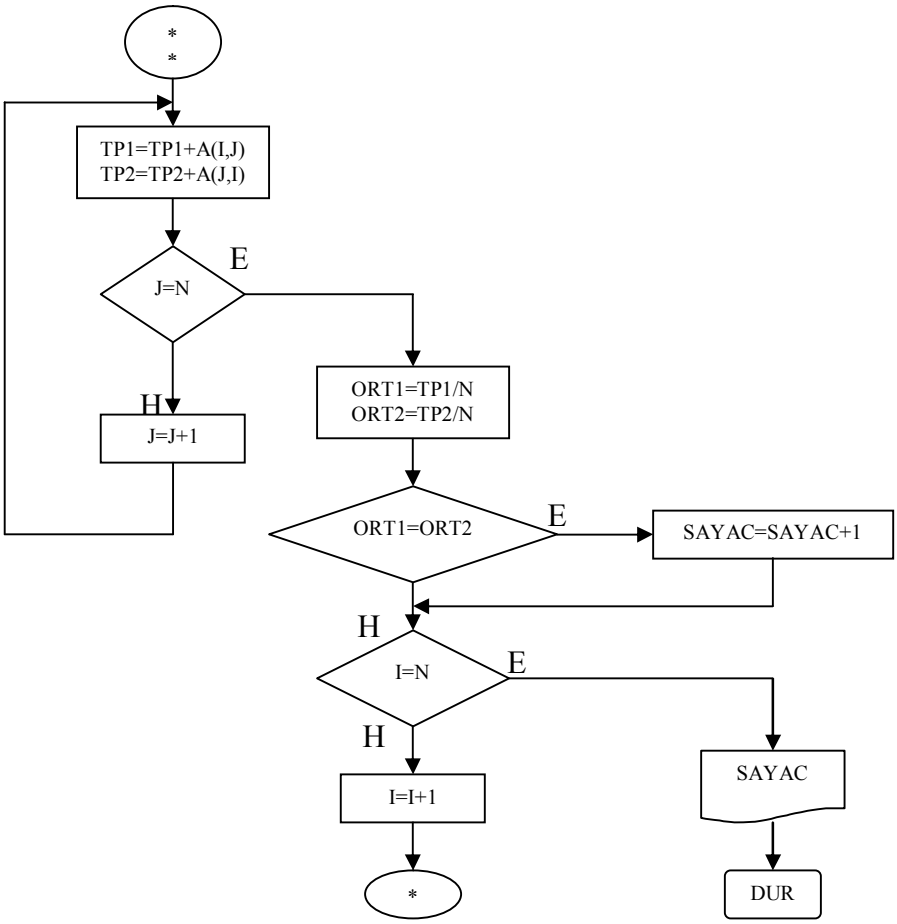
A1. Başla,
A2. N' yi gir,
A3. SAYAC=0 al,
A4. I=1 al,
A5. J=1 al,
A6. A(I,J)' yi gir,
A7. Eğer J=N ise A9. adıma git,
A8. J=J+1 al ve A6. adıma geri dön,
A9. Eğer I=N ise A11. adıma git,
A10. I=I+1 al ve A5. adıma geri dön,
A11. I=1 al,
A12. TP1=0, TP2=0 al,
A13. J=1 al,
A14. TP1=TP1+A(I,J) al,
A15. TP2=TP2+A(J,I) al,
A16. Eğer J=N ise A18. adıma git,
A17. J=J+1 al ve A14. adıma geri dön,
A18. ORT1=TP1/N al,
A19. ORT2=TP2/N al,
A20. Eğer ORT1=ORT2 ise SAYAC=SAYAC+1 al,
A21. Eğer I0N ise A23. adıma git,
A22. I=I+1 al ve A12. adıma geri dön,
A23. SAYAC' ı yaz,
A24. Dur.

Algoritmanın Pascal Dilindeki Yazılımı :

```
program islemler;
var
        i,j,,tp1,tp2,sayac,n,z : integer;
        ort1,ort2:real;
        a: array[1..20,1..20] of integer;
begin
```

```
        write (‘n sayısını gir : ‘);
        readln(n);
        for i:=1 to n do begin
                for j:=1 to n do readln(a[i,j]);
        end;
        sayac:=0;
        for i:=1 to n do begin
                tp1:=0;  tp2:=0;
                for j:=1 to n do begin
                        tp1:=tp1+a[i,j];
                        tp2:=tp2+a[j,i];
                end;
                        ort1:=tp1/n;
                        ort2:=tp2/n;
                if (ort1=ort2) then sayac:=sayac+1;
        end;
        writeln(sayac);
        readln;
end.
```





Şekil 4.7.18. NxN' lik bir sayı matrisinin her bir satırının ortalamaları ile bu matrisin transpozunun her bir satırının ortalamalarından karşılıklı olarak eşit olanların sayısını bulan akış şeması

Örnek 4.7.19. N bilinmeyenli N denklemden meydana gelen lineer denklem sisteminin çözümünü bulan algoritma ve akış şemasının oluşturulması.

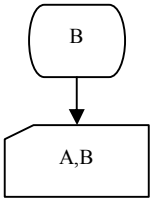
A1. Başla,
A2. A, B' yi gir,
A3. I=1 al,
A4. J=1 al,
A5. D(I,J)' yi gir,
A6. Eğer J=B ise A8. adıma git,
A7. J=J+1 al ve A5. adıma geri dön,
A8. Eğer I=A ise A10. adıma git,
A9. I=I+1 al ve A4. adıma geri dön,
A10. M=0, L=0, S=1 ve P=0 al,
A11. T=1 al,
A12. G=D(T,T) al,
A13. I=1 al,
A14. D(T,I)=D(T,I)/G al,
A15. Eğer I=B ise A17. adıma git,
A16. I=I+1 al ve A14. adıma geri dön,
A17. M=M+1, S=S+1, P=P+1 ve L=L+1 al,
A18. K=S al,
A19. C=D(K,M) al,
A20. J=P al,
A21. D(K,J)=D(L,J)*C-D(K,J) al,

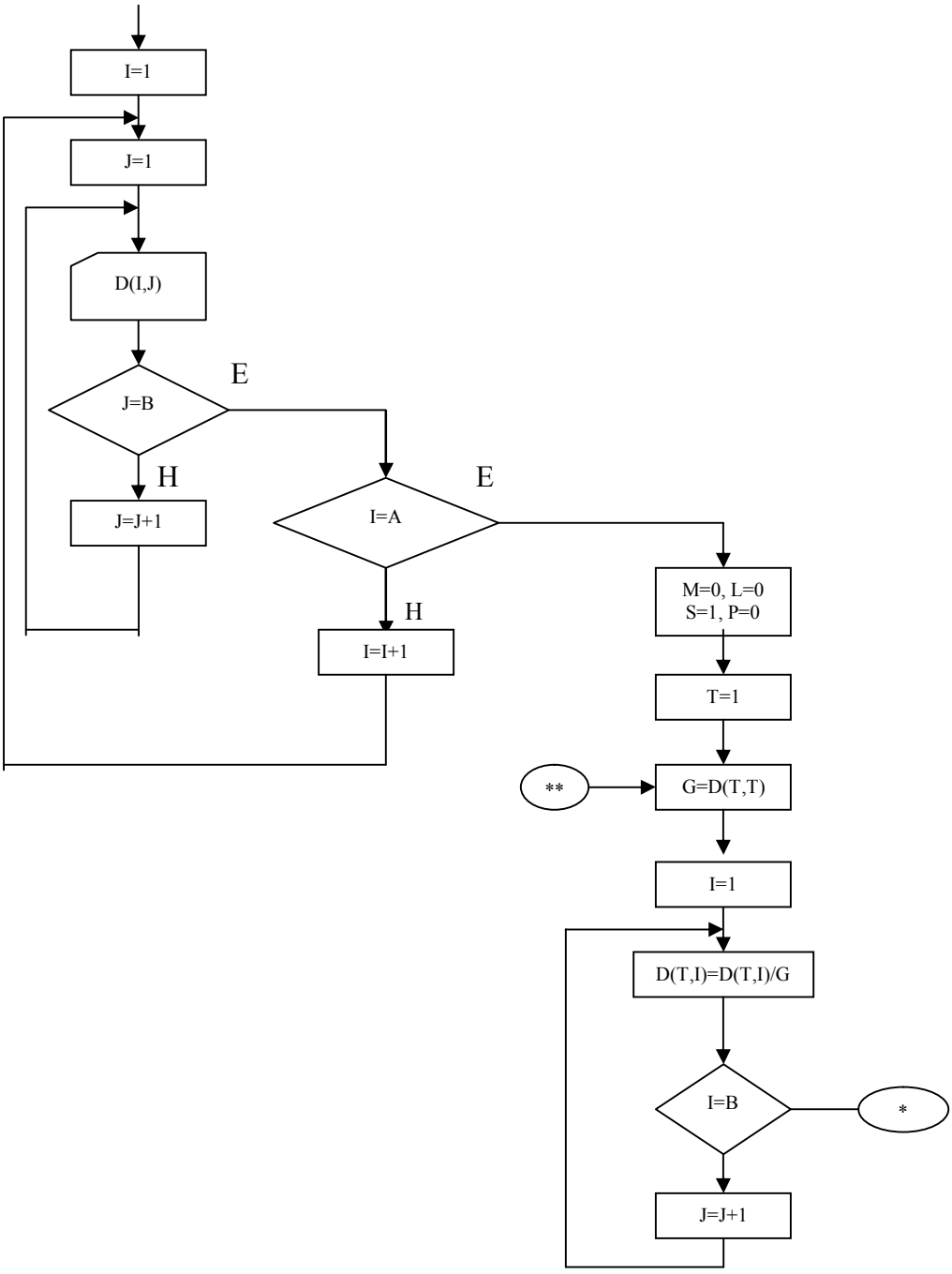
A22. Eğer J=B ise A24.adıma git,
A23. J=J+1 al ve A21. adıma geri dön,
A24. Eğer K=A ise A26. adıma git,
A25. K=K+1 al ve A19. adıma geri dön,
A26. Eğer T=A-1 ise A28. adıma git,
A27. T=T+1 al ve A12. adıma geri dön,
A28. T=A, G=D(T,T) al,
A29. I=1 al,
A30. D(T,I)=D(T,I)/G al,
A31. Eğer I=B ise A33. adıma git,
A32. I=I+1 al ve A30. adıma geri dön,
A33. X(A)=D(A,B) al,
A34. K=A-1 al,
A35. J=K+1 al,
A36. TP=TP+D(K,J)*X(J) al,
A37. Eğer J=A ise A39. adıma git,
A38. J=J+1 al ve A36. adıma geri dön,
A39. X(K)=D(K,B)-TP al,
A40. TP=0 al,
A41. Eğer K=1 ise A43. adıma git,
A42. K=K-1 al ve A35. adıma geri dön,
A43. I=1 al,
A44. X(I)' yı yaz,
A45. Eğer I=A ise A47. adıma git,
A46. I=I+1 al ve A44. adıma geri dön,
A47. Dur.

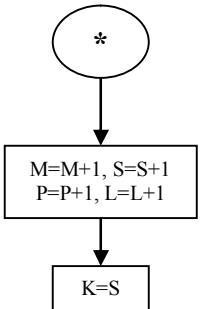
Algoritmanın Pascal Dilindeki Yazılımı :

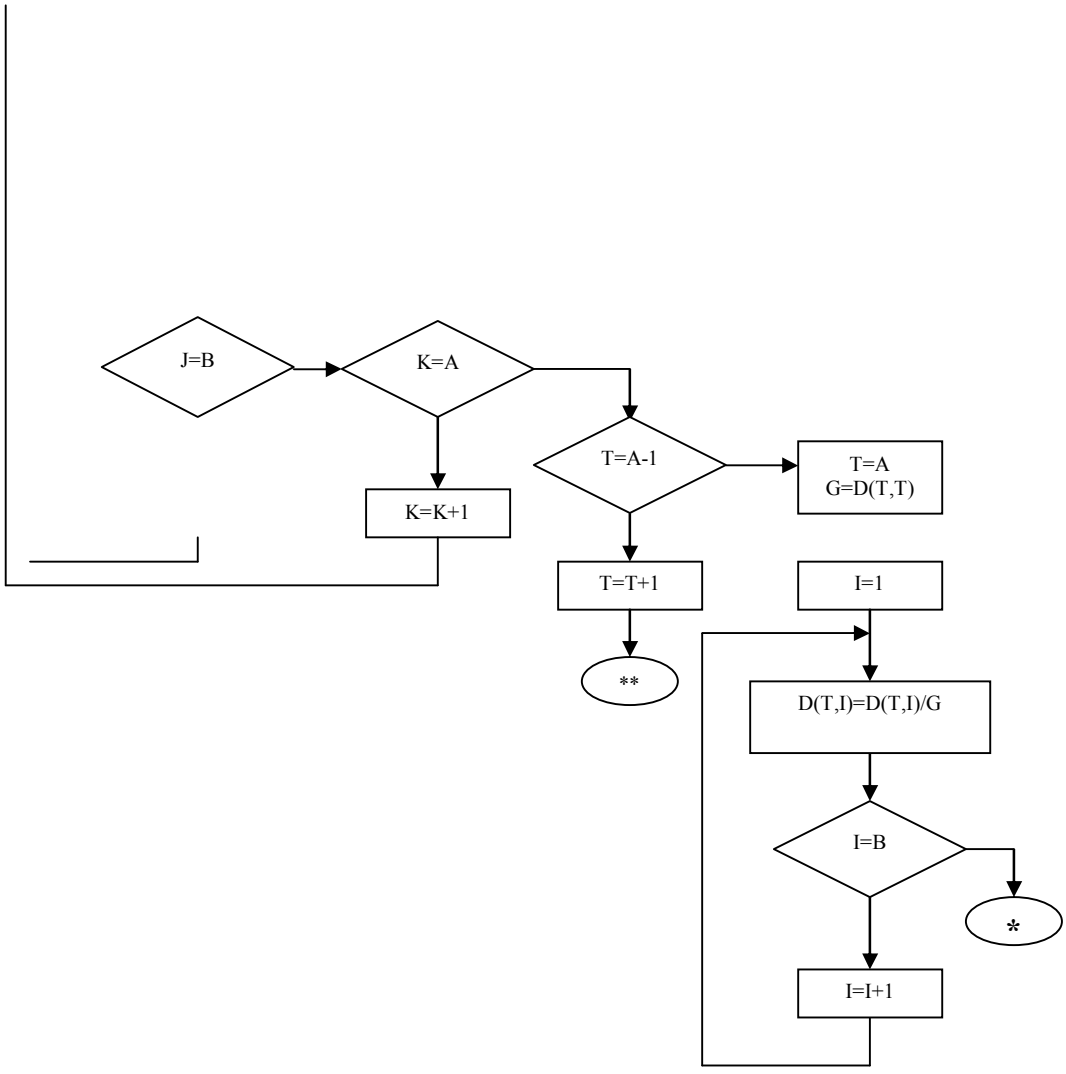
```
program denklem;
var
        i,j,a,b,k,s,m,p,l : integer;
        d: array[1..20,1..20] of real;
        x: array[1..20] of real;
        c,tp,g:real;
procedure islem1;
begin
        for i:=1 to b do begin
                d[t,i]:=d[t,i]/g;
        end;
end;

procedure islem2;
begin
        for k:=s to a do begin
                c:=d[k,m];
                for j:=p to b do
                begin
                        d[k,j]:=d[l,j]*c-d[k,j];
                end;
```

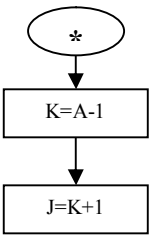
```
        end;
end;

begin
        write (‘a değerini giriniz  : ‘);
        readln(a);
        write(‘b değerini giriniz . ‘);
        readln(b);
        for i:=1 to a do
        begin
                for j:=1 to b do
                        readln(d[i,j]);
        end;
        m:=0; l:=0;   s:=1;   p:=0;
        for t:=1 to a-1 do
        begin
                g:=d[t,t];
                islem1;
                m:=m+1;   s:=s+1;
                p:=p+1;    l:=l+1;
                islem2;
        end;
        t:=a;    g:=d[t,t];
        islem1;
        x[a]:=d[a,b];
        k:=a-1;
        while (k>=1) do
        begin
                for j:=k+1 to a do
                begin
                        tp:=tp+d[k,j]*x[j];
                end;
                x[k]:=d[k,b]-tp;
                tp:=0;
                k:=k-1;
        end;
        writeln(’denklemin kökleri’);
        writeln;
        for i:=1 to a do
        begin
                writeln(‘x(‘,i,’)=’,x[i]:5:2);
        end;
        readln;
end.
```

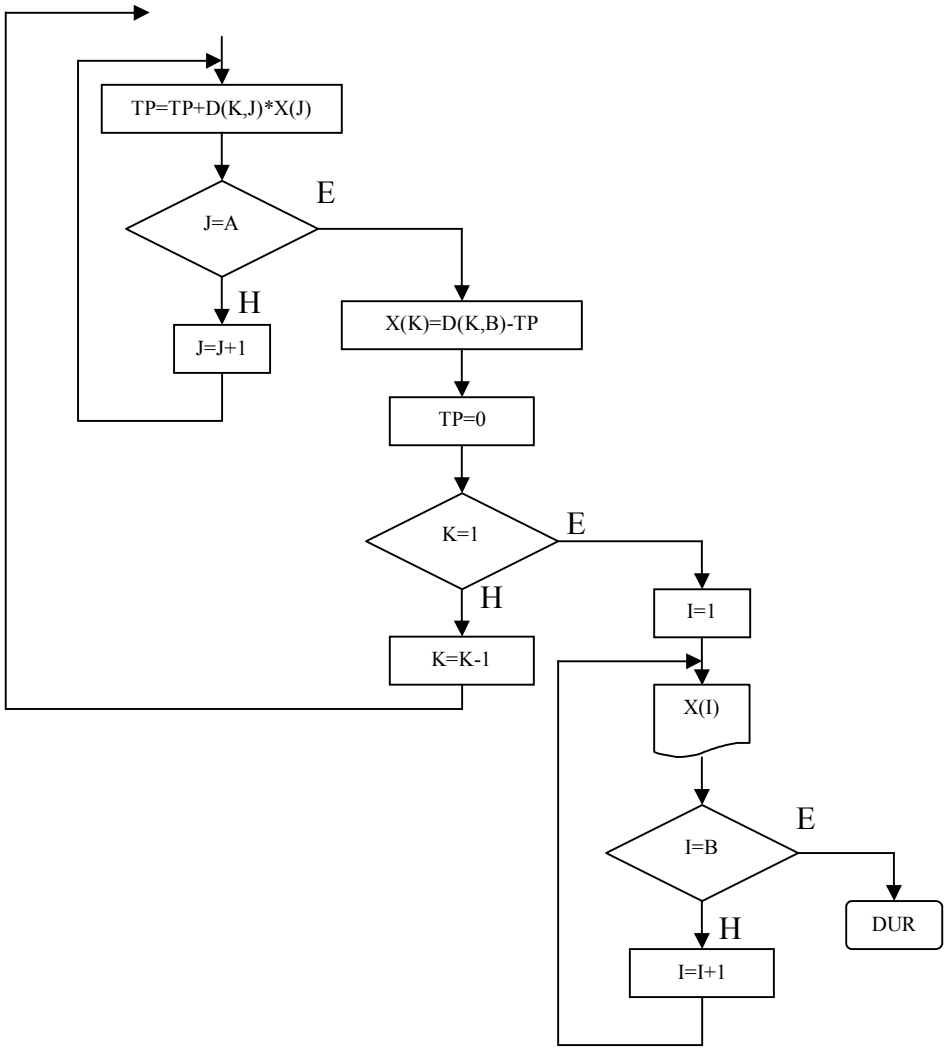



I=1
J=1
D(I,J)
J=B
E
H
J=J+1
I=A
E
H
I=I+1
M=0, L=0
S=1, P=0
T=1
**
G=D(T,T)
I=1
D(T,I)=D(T,I)/G
I=B
*
J=J+1


*
M=M+1, S=S+1
P=P+1, L=L+1
K=S


J=B
K=A
K=K+1
T=A-1
T=A
G=D(T,T)
T=T+1
**
I=1
D(T,I)=D(T,I)/G
I=B
*
I=I+1


*
K=A-1
J=K+1



Şekil 4.7.19. N bilinmeyenli lineer denklem sistemini çözen akış şeması

BÖLÜM 5

5. DOSYALAMA SİSTEMLERİ

5.1. 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.

Dosyalama sistemlerini iki ana grupta incelemek gerekir. Bunlardan birincisi sırasal erişimli dosyalar, diğeri ise doğrudan erişimli dosyalardır.

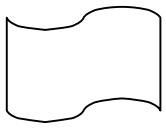
5.1.1. Sırasal Erişimli Dosyalar

Sırasal erişimli dosyalarda bilgiler kaydediliş sırasına göre dosya içerisinde yer alırlar. Yani birbiri ardına kaydedilirler. Dolayısıyla istenilen bir bilgiye ulaşmakta yine sırasallık gerektirmektedir. Ulaşılmak istenilen kayda ilişkin herhangi bir bilgi verilerek dosya baştan itibaren taranmak suretiyle istenilen bilgiye ulaşmak mümkün olmaktadır. Bir anlamda, n. bilgiye ulaşmak için n-1 bilginin sırasıyla ana belleğe taşınarak kontrol edilmesi gerekir.

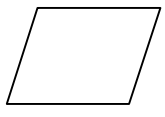
5.1.2. Doğrudan Erişimli Dosyalar

Doğrudan erişimli dosyalarda girilen bilgiler birer kayıt numarası ile dosya içerisinde saklanmaktadır. Bu kayıt numaraları bir anlamda bilgilerin adreslerini tanımlayan sistemler olarak adlandırılabilirler. İstenilen bir kayda ulaşmak için o kayda ilişkin kayıt numarasının girilmesi ile mümkün olabilmektedir. Bu tür dosyalama sistemleri sırasal erişimli dosyalara göre daha kullanışlıdır.

Dosyalama sistemlerinde, diğer işlemlerde kullanılan akış şeması şekillerine ilave olarak dosyaya bilgi yazdırma ve dosyadan bilgi okuma şekilleri tanımlanacaktır. Ayrıca diğer işlemlerde kullanılan bazı şekiller, dosyalama sisteminde farklı amaç için de kullanılabilmektedir. Örneğin; dosyanın tanımlanması, dosyanın açılması, dosyanın kapanması, herhangi bir kayda konumlanma gibi işlemler önceden bilinen atama ve işlem şekli ile tanımlanabilecektir.



Dosyaya bilgi yazdırma



Dosyadan bilgi okuma

5.2. SIRASAL ERİŞİMLİ DOSYALARA İLİŞKİN ALGORİTMA VE AKIŞ ŞEMALARI

Bu kısımda sırasal erişimli dosyalara ilişkin tüm işlemleri içeren algoritma ve akış şemaları verilmiştir. Sırasıyla bu tür dosyalara kayıt işlemleri, istenilen kayıtlara ulaşılmasını sağlayan işlemler, istenilen bir kaydın silinmesi ya da bu kayıttaki bazı

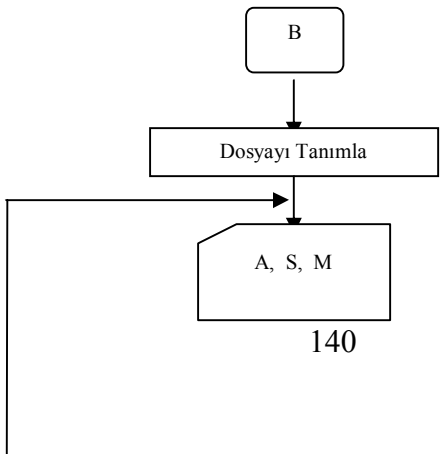
bilgilerin değiştirilmesine yönelik işlemleri içeren algoritma ve akış şemalarının oluşturulması sağlanmıştır.

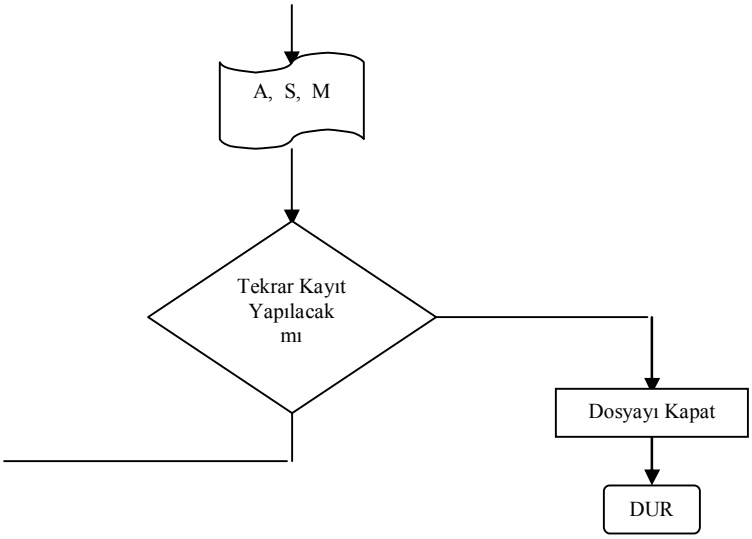
Örnek 5.2.1: Sırasal erişimli bir dosyaya çalışanlara ait ad-soyad, sicil numarası ve maaş bilgilerini kaydeden algoritma ve akış şemasının oluşturulması.

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.

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.
```





Şekil 5.2.1:Sırasal erişimli bir dosyaya kayıt yapılmasına ilişkin akış şeması

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

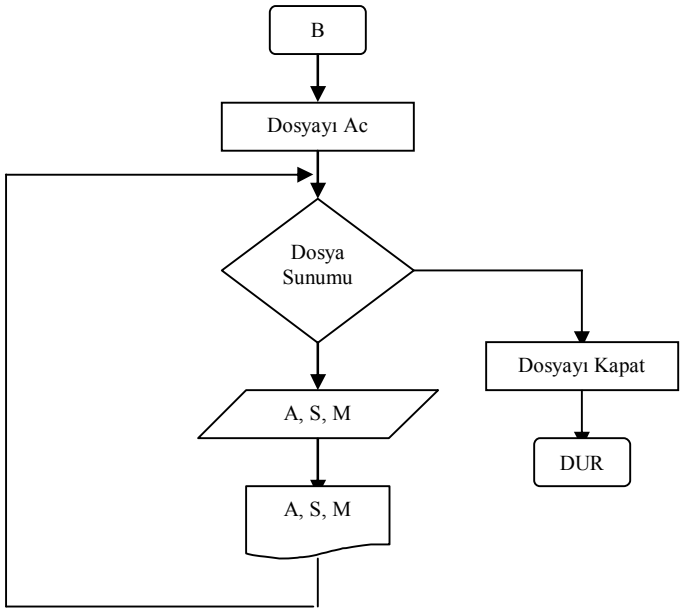
Örnek 5.2.2: Yukarıdaki örnekte verilen dosyadan bilgileri okuyarak ekrana yazdıran algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. Dosyayı aç,
A3. Eğer dosya sonu ise A7. adıma git,
A4. A,S ve M' yi dosyadan oku,
A5. A,S ve M' yi ekrana yaz,
A6. A3. adıma geri dön,
A7. Dosyayı kapat ve dur,

Algoritmanın Pascal dilindeki yazılımı:

```
Program listeleme;
var
        a:string[20];
        s:integer;
```

```
        m:longint;
        c:char;
        t:text;
begin
        assign(t,'bilgi'); reset(t);
        while(not eof(t)) do begin
                readln(t,a);
                readln(t,s);
                readln(t,m);
                writeln(a,'',s,'',m);
        end;
        close(t);
        readln;
end.
```



Şekil 5.2.2: Sırasal erişimli dosyadan bilgi okunmasına ilşkin akış şeması

Bu örnekte, bir önceki örnekte oluşturulan dosyadaki bilgilerin ekrana yazdırılması sağlanmaktadır. A2. adımda dosya açıldıktan sonra, A3. adımda dosya sonu kontrolü yapılmaktadır. Buradaki amaç dosyadaki bilgilerin tümünü tarayarak bunların ekrana yazdırılmasını sağlamaktır. Sırasal erişimli dosyalarda bilgiler okunurken tüm dosyanın taranması gerektiğinden okutma işlemi dosya sonu olana kadar yapılmaktadır. Eğer dosya sonuna gelinmiş ise; tüm bilgiler taranmış olacağından A7. adıma gidilerek dosya kapatılıp işlemlere son verilmektedir. Aksi halde, A4. adımdan itibaren işlemlere devam edilmektedir.

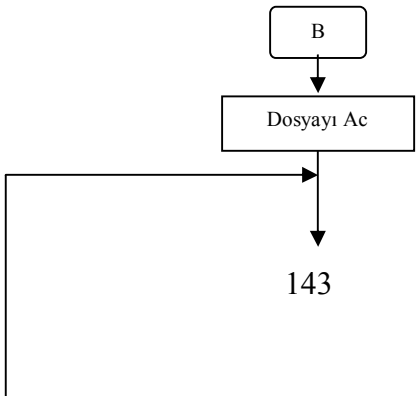
Örnek 5.2.3: Bir dosyada öğrencilerin adı-soyadı, numarası ve bilgisayar dersinden almış oldukları vize ve final notları kayıtlıdır. Bu dosyayı okutarak başarılı olan öğrencilerin ekrana yazdırılmasını sağlayan algoritma ve akış şemasının

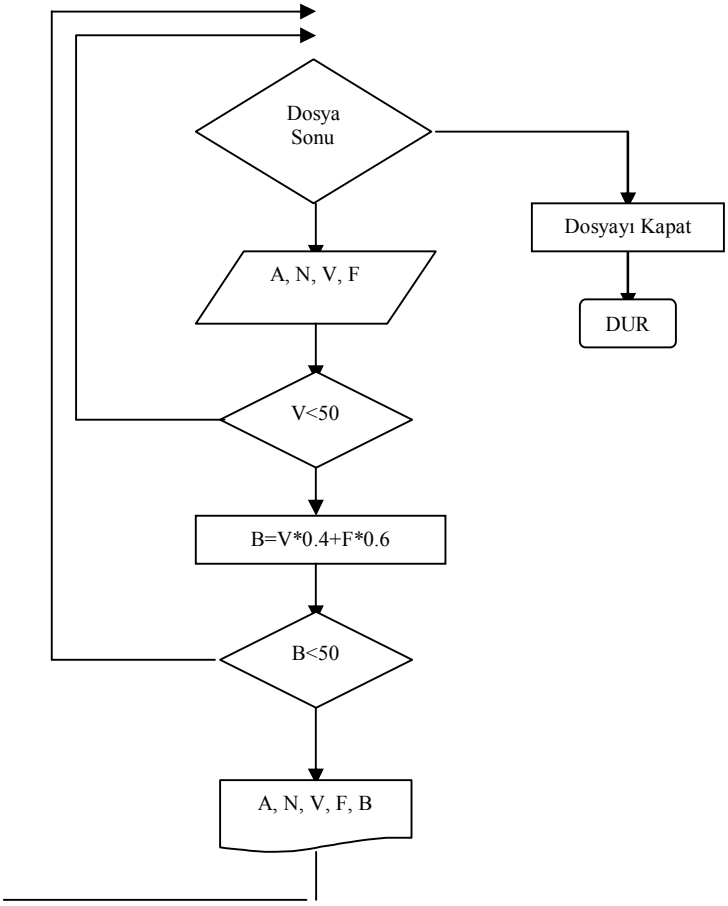
Başarı notu hesaplanırken öncellikle öğrencinin final notunun 50 ya da 50'nin üzerinde olması gerekir. Bu şart sağlanıyorsa vize notunun %40' ı ile final notunun %60' ı toplanarak başarı notunu oluşturmaktadır.

A1. Başla,
A2. Dosyayı aç,
A3. Eğer dosya sonu ise A9. adıma git,
A4. A,N,V ve F' yi dosyadan oku,
A5. Eğer F<50 ise A3. adıma geri dön,
A6. B=V*0.4+f*0.6 al.
A7. Eğer B<50 ise A3. adıma geri dön,
A8. A,N,V,,F ve B' yi ekrana yaz.
A9. Dosyayı kapat,
A10. Dur.

Algoritmanın Pascal dilindeki yazılımı:

```
Program kayit;
var
        a:string[20];
        n,v,f:integer;
        b:real;
        c:char;
        t:text;
begin
        assign(t,'ogrenci'); reset(t);
        while(not eof(t)) do begin
                readln(t,a);
                readln(t,n);
                readln(t,v);
                readln(t,f);
                if (f>=50) then begin
                        b:=v*0.4+f*0.6;
                        if (b>=509 then
                        writeln(a,'',n,'',v,'',f,'',b);
                end;
        end;
        close(t);
        readln;
end.
```





Şekil 5.2.3: Başarılı öğrencilerin yazdırılmasına ilişkin akış şeması.

Dosyanın açılması ve dosya sonu kontrolü yapıldıktan sonra, A4. adımda dosyadaki bilgiler okutularak A5. adımda final notunun yeterli olup olmadığı karşılaştırılmaktadır. Eğer final notu olan F değeri 50' den küçük ise A3. adıma geri dönülerek diğer kayıt okutulmaktadır. Aksi halde A6.adımda B olarak adlandırılan başarı notu hesaplanarak A7. adımda bu not 50 değeri ile karşılaştırılmaktadır. Eğer B değeri 50 ya da 50'den büyük ise öğrenci başarılı olacağından A8. adımda bu bilgiler ekrana yazdırılmaktadır. Aksi halde A3. adıma geri dönülmektedir. Dosyada kayıtlı tüm bilgiler okutulduktan sonra dosya kapatılarak işlemlere son verilmektedir.

Örnek 5.2.4: 3. Örnekte verilen ve öğrencilerin notlarına ilişkin bilgileri taşıyan sırasal erişimli dosyadan istenilen bir kaydın silinmesini sağlayan algoritma ve akış şemasının oluşturulması.

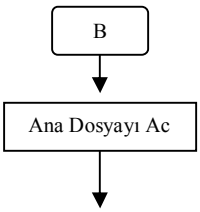
Sırasal erişimli dosyalarda istenilen bir kaydın silinebilmesi için geçici bir dosyanın oluşturulması gerekir. Silinecek kişiye ait herhangi bir bilgi girildikten sonra bu bilgiye göre dosya taranarak silinmesi istenilen kaydın dışındaki diğer kayıtlar geçici olarak adlandırılan dosyaya yazılır. Daha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

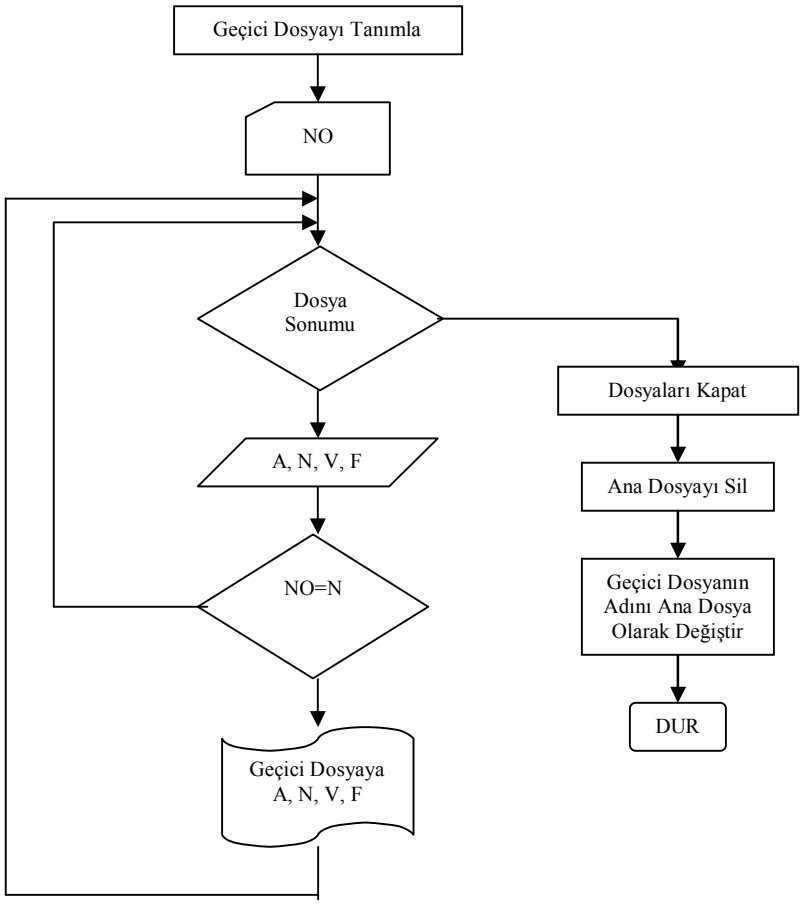
A1. Başla,
A2. Ana dosyayı aç,

A3. Geçici dosyayı tanımla,
A4. NO' yu gir {Silinecek öğrencinin numarası}
A5. Eğer ana dosyanın sonu ise A10. adıma git,
A6. A,N,v ve F' yi ana dosyadan oku,
A7. Eğer NO=N ise A5. adıma git,
A8. A,N,V ve F' yi geçici dosyaya yaz,
A9. A5. adıma geri dön,
A10. Ana dosyayı ve geçici dosyayı kapat,
A11. Ana dosyayı sil,
A12. Geçici dosyanın adını ana dosyanın adı olarak tanımla ve dur,

Algoritmanın Pascal dilindeki yazılımı:

```
Program iptal;
var
        a:string[20];
        n,v,f,nm:integer;
        c:char;
        t,k:text;
begin
        assign(t,'ogrenci');
        reset(t);
        assign(k,'gecici');
        rewrite(k);
        write('Silinecek kişinin numarasını giriniz..:');
        readln(nm);
        while(not eof(t)) do begin
                readln(t,a);
                readln(t,n);
                readln(t,v);
                readln(t,f);
                if (nm<>n) then begin
                        writeln(k,a);
                        writeln(k,n);
                        writeln(k,v);
                        writeln(k,f);
                end;
        end;
        close(t);
        close(k);
        erase(t);
        rename(k,'ogrenci');
        readln;
end.
```





Şekil 5.2.4: Sırasal erişimli dosyadan bir kaydın iptal edilmesine ilişkin akış şeması

Verilen örnekte ana dosya bilgi okumak amacıyla, geçici olarak adlandırılan dosya da bilgi yazmak amacıyla açılmıştır. A4. adımda silinmesi istenilen öğrencinin numarası girilerek bu bilgisine göre taranması sağlanmıştır. A5. adımda ana dosya sonu kontrolü yapılarak, eğer dosya sonuna gelinmişse A6. adımda dosyadan bilgi okutularak A7. adımda aranan numara ile dosyadan okutulan numara karşılaştırılmaktadır. Eğer bu iki bilgi eşit ise silinmesi gereken kişiye ulaşılmış olacağından bu bilgi geçici dosyaya yazdırılmayacaktır. Bu nedenle A5. adıma geri dönülerek diğer bilginin okutulması sağlanmaktadır. Eğer dosyadan okutulan numara ile klavyeden girilen numara eşit değilse dosyadan okutulan bu kayıt geçici dosyaya yazdırılmak üzere A8. adıma gidilmektedir.

Bu işlemler ana dosyadaki tüm kayıtların okunmasına kadar devam etmektedir. Ana dosyadaki bilgilerin sonuna gelindiğinde, her iki dosya da A10. adımda kapatılarak A11. adımda ana dosya silinmektedir. A12. adımda geçici dosyanın adı olarak değiştirilerek işlemlere son verilmektedir.

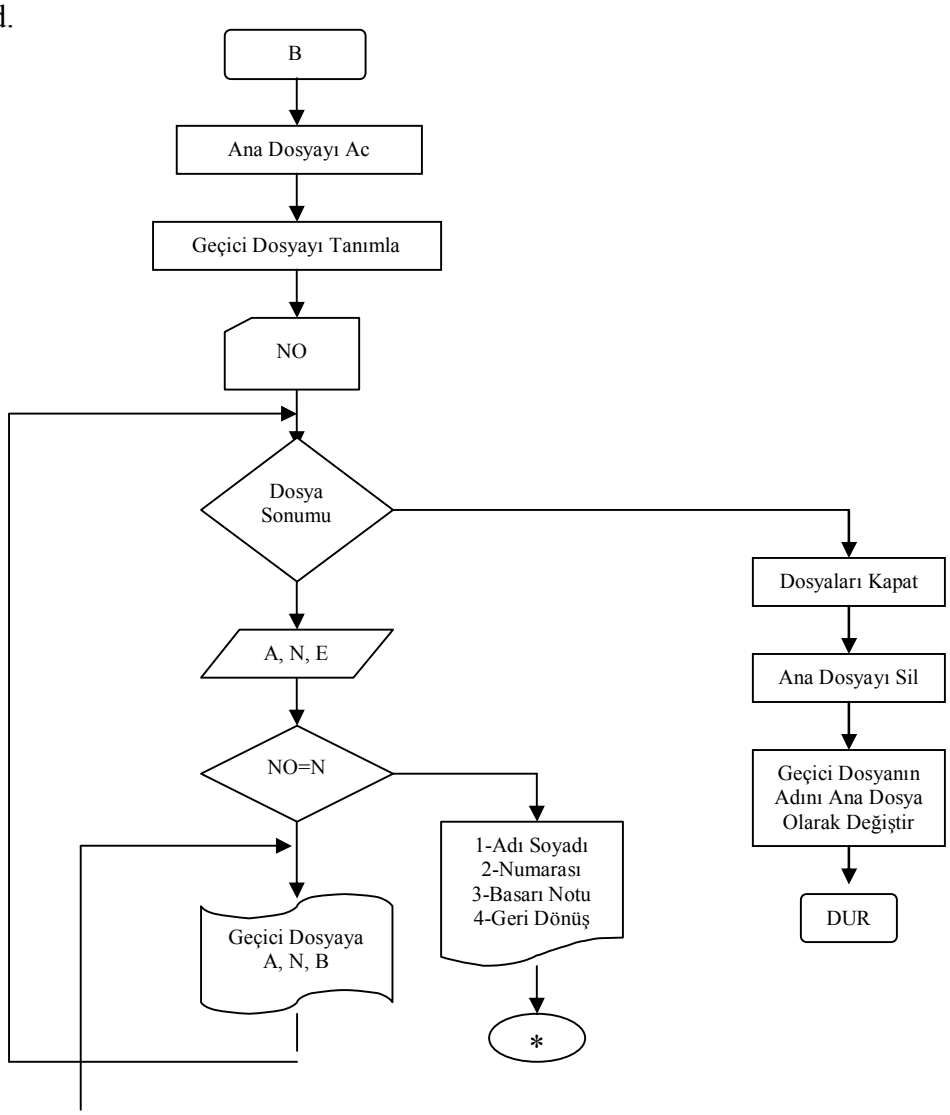
Örnek 5.2.5: Öğrencilerin adı-soyadı, numarası ve bilgisayar dersinden almış oldukları başarı notunun kayıtlı olduğu bir sırasal erişimli dosyadan, numarasına göre istenilen bir öğrenciye ulaşılarak bu öğrenciye ait bilgilerden istenilenlerin değiştirilmesini sağlayan algoritma ve akış şemasının oluşturulması.

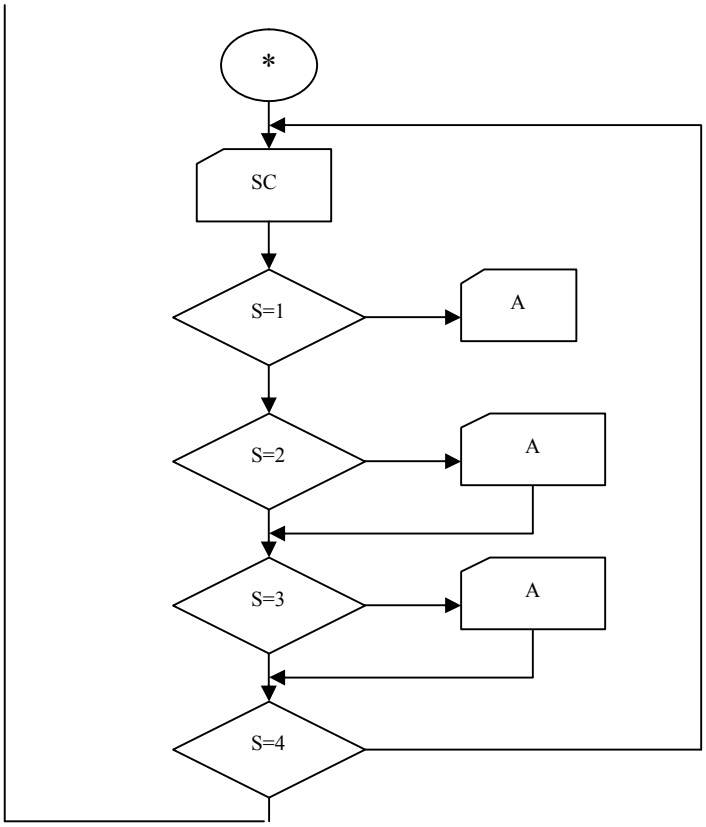
A1. Başla,
A2. Ana dosyayı aç,
A3. Geçici dosyayı tanımla,
A4. NO' yu gir{Aranan öğrencinin numarası},
A5. Eğer ana dosyanın sonu iseA10. adıma git,
A6. A, N ve B' yi ana dosyadan oku,
A7. Eğer NO=N ise A10. adıma git,
A8. A,N ve B' yi geçici dosyaya yaz,
A9. A5. adıma geri dön,
A10. "1-Adı soyadı, 2-Numarası, 3-Başarı notu, 4-Çıkış" yaz
A11. "Seçininiz nedir" diye sor,
A12. SC' yi gir {Burada SC değeri 1,2,3 ya da 4 olacaktır},
A13. Eğer SC=1 ise A' yı gir,
A14. Eğer SC=2 ise N' yi gir,
A15. Eğer SC=3 ise B' yi gir,
A16. Eğer SC=4 ise A8.adıma geri dön,
A17. A5. adıma geri dön,
A18. Ana dosyayı ve geçici dosyayı kapat,
A19. Ana dosyayı sil,
A20. Geçici dosyanın adın ana dosyanın adı olarak tanımla,
A21. Dur.

Algoritmanın Pascal dilindeki yazılımı:

```
Program guncelleme;
var
        a:string[20];
        n,b,nm,sc:integer;
        c:char;
        t,k:text;
begin
        assign(t,'basari'); reset(t);
        assign(k,'gecici'); rewrite(k);
        write('guncellenecek kişinin numarasını giriniz..:');readln(nm);
        while(not eof(t)) do begin
                readln(t,a);
                readln(t,n);
                readln(t,b);
                if (nm=n) then begin
                        writeln('1- adi soyadi   2- numarasi   3-basari notu   4-cikis');
                        writeln(a,'   ',n,'   ',,b);
                        write('Seciminiz...:');readln(sc);
                        if(sc=1) then begin
                                write('Yeni ad soyad..:');
                                readln(a);
```

```
                    end;
                    if(sc=2) then begin
                            write('Yeni numara..:');
                            readln(n);
                    end;
                    if(sc=3) then begin
                            write('Basari notu..:');
                            readln(b);
                    end;
            end;
            writeln(k,a);
            writeln(k,n);
            writeln(k,b);
            end;
    close(t);
    close(k);
    erase(t);
    rename(k,'basari');
    readln;
end.
```





Şekil 5.2.5: Sırasal erişimli bir dosyada kayıtların güncelleşmesine ilişkin akış şeması

Sırasal erişimli bir dosyada istenilen bir kayda ulaşmak için daha önce verilen örnekte olduğu gibi bu kayda ilişkin herhangi bir bilgi girilerek buna göre tarama yapılıp aranan kayda ulaşmak mümkün olabilmektedir. Bu aşamada bir önceki örnekte olduğu gibi geçici bir dosyanın kullanılması gerekmektedir. Aranan kaydın dışındaki kayıtlar ve aranan kaydın değiştirilmesinden sonraki hali geçici dosyaya kaydedilmektedir. Yukarıdaki örnekte de bu yapı esas alınarak gerekli işlemlerin yapılması gerçekleştirilmektedir.

Örnekte ana dosya bilgi okumak amacıyla, geçici olarak adlandırılan dosya da bilgi yazmak amacıyla açılmıştır. A4. adımda aranan öğrencinin numarası girilerek bu bilgiye göre taranması sağlanmıştır. A5. adımda ana dosya sonu kontrolü yapılarak, eğer dosya sonuna gelinmemiş ise A6. adımda dosyadan bilgi okutularak A7. adımda aranan öğrencinin numarası ile dosyadan okutulan numara karşılaştırılmaktadır. Eğer bu iki bilgi eşit ise aranan kişiye ulaşılmış olacağından A10. adıma gidilerek öğrenciye ilişkin bilgilere birer numara verilerek ekrana yazdırılması sağlanmaktadır. Daha sonra, hangi bilginin değiştirileceğine ilişkin bilgi girişi A12. adımda yapılarak buna göre değiştirilecek bilgiye ilişkin karşılaştırmalar A13. adımdan itibaren yapılmaktadır. Değişiklikleri içeren ilgili atamalar yapıldıktan sonra bu bilgilerin geçici dosyaya yazdırılması için tekrar A8. adıma dönülmektedir. Eğer aranan kişiye ilişkin numara ile dosyadan okutulan numara ile aynı değilse bir sonraki adıma gidilerek bu bilgiler geçici dosyaya yazdırılmaktadır. Bu aşamadan sonra ana dosyadaki tüm bilgilerin teker teker okutularak geçici dosyaya aktarımı yapıldıktan sonra, bir önceki örnekte olduğu gibi dosyalar kapatılarak ana dosyanın silinmesi sağlanmaktadır. Son olarak da geçici dosyanın ismini ana dosyanın ismi olarak değiştirerek işlemlere son verilmektedir.

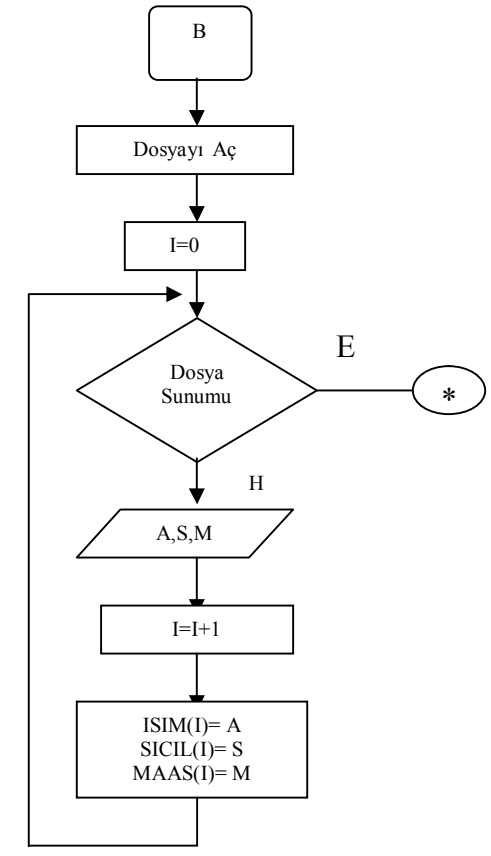
Örnek 5.2.6: Çalışanlara ait isim, sicil numarası ve maaş bilgilerinin kayıtlı olduğu bir sırasal erişimli dosyadan bu bilgilerin okutularak isimlere göre alfabetik olarak sıralanmasını sağlayan algoritma ve akış şemasının oluşturulması.

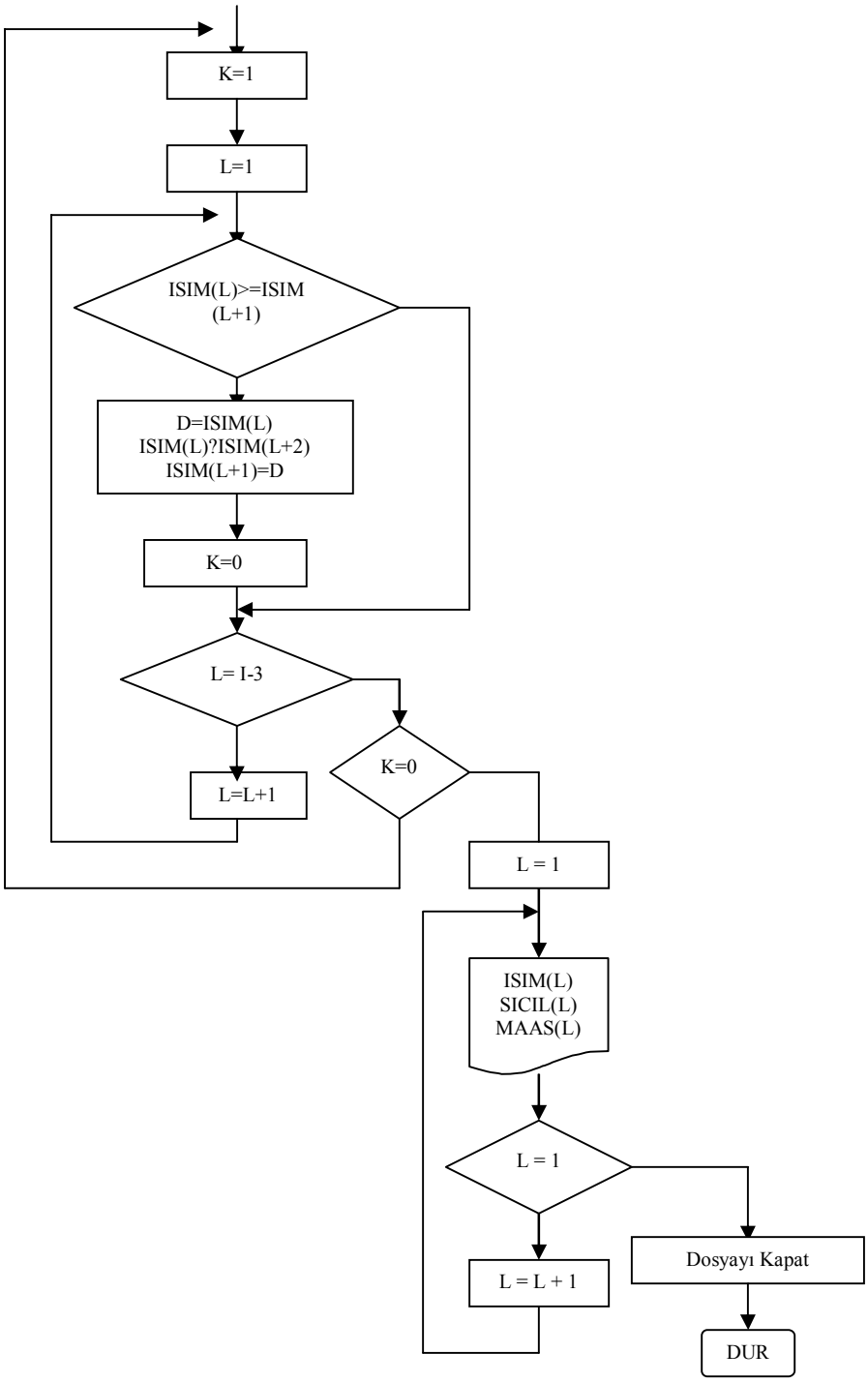
A1. Başla,
A2. Dosyayı aç,
A3. I=0 al,
A4. Eğer ana dosyanın sonu ise A9. adıma giit,
A5. A,S ve M' yi ana dosyadan oku,
A6. I=I+1 al,
A7. ISIM(I)=A, SICIL(I)=S ve MAAS(I)=M al,
A8. A3. Adıma geri dön,
A9. K=1 al,
A10. L=1 al,
A11. Eğer ISIM(L)>=ISIM(L+1) ise A16. Adıma git,
A12. D=ISIM(L), ISIM(L)=ISIM(L+1), ISIM(L+1)=D al,
A13. E=SICIL(L), SICIL(L)=SICIL(L+1), SICIL(L+1)=E al,
A14. F=MAAS(L), MAAS(L)=MAAS(L+1), MAAS(L+1)=F al,
A15. K=0 al,
A16. Eğer L=I-1 ise A18. adıma git,
A17. L=L+1 al ve A11. adıma geri dön,,
A18. Eğer K=0 ise A9. adıma geri dön,
A19. L=1 al,
A20. ISIM(L), SICIL(L) ve MAAS(L) ' yi yaz,
A21. Eğer L=1 ise A23. adıma git,
A22. L=L+1 al ve A20. adıma geri dön,
A23. Dosyayı kapat,
A24. Dur.

Algoritmanın Pascal dilindeki yazılımı:

```
program siralama;
var
        a,d:string[20];
        i,k,l:integer;
        s,m,e,f:longint;
        isim:array[1..20] of longint;
        c:char;
        t:text;
begin
        assign(t,'maaslar'); reset(t);
        i:=1;
        while(not eof(t)) do begin
                readln(t,a);
                readln(t,s);
                readln(t,m);
                isim[i]:=a;
                sicil[i]:=s;
```

```
            maas[i]:=m;
            i:=i+1;
        end;
        k:=1;
        while(k<>0) do begin
            k:=0;
            for l:=1 to i-1 do begin
                if(isim[l]>isim[l+1] then begin
                    d:=isim[l]; isim[l]:=isim[l+1]; isim[l+1]:=d;
                    e:=sicil[l]; sicil[l]:=sicil[l+1]; sicil[l+1]:=e;
                    f:=maas[l]; maas[l]:=maas[l+1]; maas[l+1]:=f; k:=1;
                end;
            end;
        end;
        for l:=1 to i do
        writeln(isim[l],'',sicil[l],'',maas[l]);
        close(t);
        readln;
end.
```





Şekil 5.2.6: Sırasal erişimli bir dosyada sıralama işlemini yapan akış şeması

Örnekte, öncelikle dosyadan okutulan bilgilerin sıralanabilmesi için dizi elemanlarına aktarımı sağlanmaktadır. Bunun için A3. Adımda bir l indisi tanımlanarak ilk değer olarak 0 değeri atanmıştır. Sonraki adımda dosya sonu kontrolü yapılmaktadır. Eğer dosya sonuna gelinmişse dosyadaki kayıt okutularak bu bilgiler dizinlere atanmaktadır. İsim bilgisini atamak üzere ISIM adlı bir dizi, sicil numarasını atamak üzere SICIL isimli bir dizi ve maaş bilgilerini atamak üzere MAAS isimli bir dizi tanımlanmıştır. A6. adımda l indisi bir arttırılarak okutulan bilgiler dizinlerin l elemanı

olarak atanmaktadır. Bu işlemler dosya sonuna gelininceye kadar devam etmektedir. Dosya sonuna gelindiğinde dizilere atama işlemi tamamlanmış olacağından dizilerin sıralanması işlemlerine geçilmektedir. Burada dikkat edildiğinde dosyada kayıtlı eleman sayısı l kadardır. Dolayısıyla dizi elemanlarının sayısı da l kadar olacaktır. Bu aşamadan sonraki işlemler dizilerde sıralama işlemleri ile aynıdır. Ancak, birden fazla dizinin olması nedeniyle dizi elemanlarının yer değişimleri de bu oranda işleme tabi tutulmaktadır. Sıralamada esas alınan bilgi isim bilgisi olduğundan ISIM dizisi sıralamada ön plandadır ve bu dizideki yer değişikliği söz konusu olduğunda diğer dizi elemanlarının da yer değiştirilmesi gerekmektedir. K olarak tanımlanan değişken de dizilerde sıralama işlemlerinden hatırlanacağı üzere sıralama işlemi gerçekleştiği anda döngüden çıkmayı sağlamak için kullanılmaktadır.

Örnek 5.2.7: Bir önceki örnekte tanımlanan dosyadan isim referans alınarak yarılama metodu ile istenilen bir kayda ulaşmayı sağlayan algoritma ve akış şemasının oluşturulması.

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 yok açabilmektedir. Bu nedenle dosyadaki bilgilerin bir önceki örnekte olduğu gibi dizilere aktarılarak sıralanması sağlanarak yarılama metoduyla istenilen kayıta ulaşılması daha kolay olacaktır.

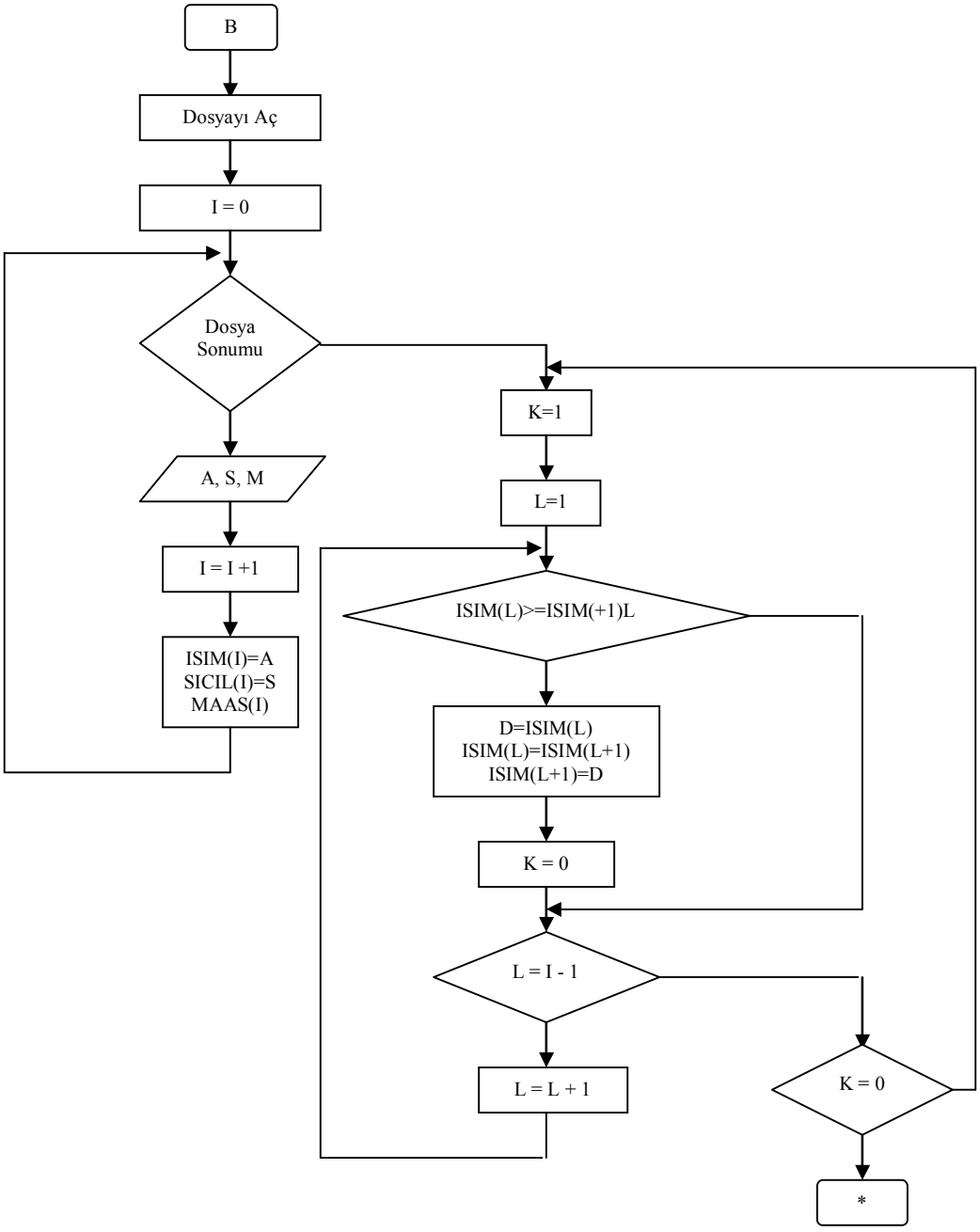
A1. Başla,,
A2. Dosyayı aç,
A3. I=0 al,
A4. Eğer ana dosyanın sonu ise A9. adıma git,
A5. A,S ve M' yi ana dosyadan oku,
A6. I=I+1,
A7. ISIM(I)=A, SICIL(I)=S, MAAS(I)=M al
A8. A3. adıma geri dön,
A9. K=1 al,
A10. L=1 al,
A11. Eğer ISIM(L)>=ISIM(L+1) ise A16. adıma git,
A12. D=ISIM(L), ISIM(L)=ISIM(L+1), ISIM(L+1)=D al,
A13. E=SICIL(L), SICIL(L)=SICIL(L+1), SICIL(L+1)=E al,
A14. F=MAAS(L), MAAS(L)=MAAS(L+1), MAAS(L+1)=F al,
A15. K=0 al,
A16. Eğer L=I-1 ise A18. adıma git,
A17. L=L+1 al ve A11. adıma geri dön,,
A18. Eğer K=0 ise A9. adıma geri dön,
A19 ISM' yi gir {aranan kişinin ismi},
A20. BS=1, BT=1 al,
A21. ORT=TAM((BS+BT)/2) al,
A22. Eğer ISIM(ORT)=ISM ise A26. adıma git,
A23. Eğer ISIM(ORT)<ISM ise A25. adıma git,
A24. BT=ORT al A21. adıma geri dön,
A25. BS=ORT al A21. adıma geri dön,

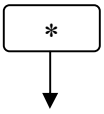
A26. ISIM(ORT), SICIL(ORT) VE MAAS(ORT) bilgilerini yaz,
A27. Dosyayı kapat,
A28. Dur.

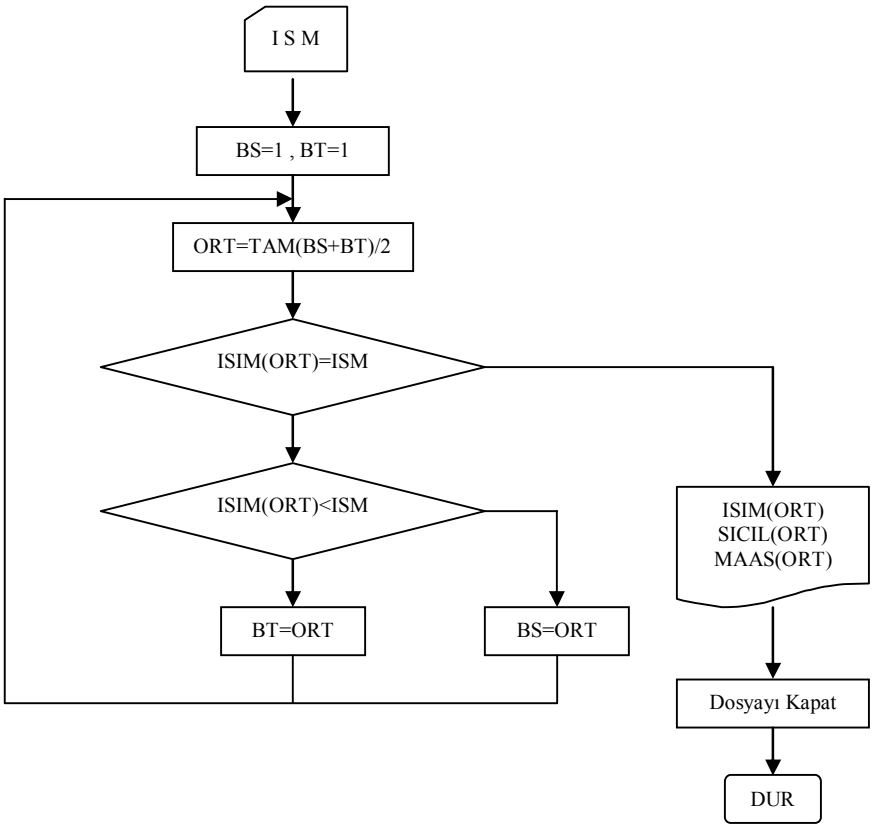
Algoritmanın Pascal dilindeki yazılımı:

```
program yarilama;
var
        a,d,ism:string[20];
        i,k,l,bs,bt,ort:integer;
        s,m,e,f:longint;
        isim:array[1..20] of string[20];
        sicil,maas:array[1..20] of longint;
        c:char;
        t:text;
begin
        assign(t,'maaslar'); reset(t);
        i:=1;
        while(not eof(t)) do begin
                readln(t,a);
                readln(t,s);
                readln(t,m);
                isim[i]:=a;
                sicil[i]:=s;
                maas[i]:=m;
                i:=i+1;
        end;
        k:=1;
        while(k<>0) do begin
                k:=0;
                for l:=1 to i-1 do
                begin
                        if(isim[l]>isim[l+1] then
                        begin
                                d:=isim[l]; isim[l]:=isim[l+1]; isim[l+1]:=d;
                                e:=sicil[l]; sicil[l]:=sicil[l+1]; sicil[l+1]:=e;
                                f:=maas[l]; maas[l]:=maas[l+1]; maas[l+1]:=f; k:=1;
                        end;
                end;
        end;
        write('Aranan kisinin ismini giriniz..:');
        readln(ism);
        c:='e';
        bs:=1;
        bt:=1;
        while(c<>'h') do
        begin
                ort:=trunc((bs+bt)/2);
                if(ism=isim[ort] then begin
```

```
                writeln(isim[ort],'',sicil[ort],'',maas[ort]);
                c:='h';
            end;
            if(ism<isim[ort]) then bt:=ort;
            if(ism>isim[ort]) then bs:=ort;
        end;
        close(t);
        readln;
end.
```







Şekil 5.2.7: Sırasal erişimli bir dosyadan yarılama metodu ile istenilen kayda ulaşılmasına ilişkin akış şeması

Bir önceki örnekte olduğu gibi ilk 18 adımda dosyadaki bilgilerin dizilere aktarımı ve sıralanması yapılmaktadır. Bundan sonraki aşamalarda yarılama metodu kullanılarak istenilen bir kayda ulaşım için gerekli işlemlere geçilmektedir. Dizilerle ilgili örneklerde de belirtildiği gibi yarılama işlemi sıralı dizinin ortasındaki bilginin aranan eleman olup olmadığının kontrolü şeklinde yapılmaktadır. Bunun için öncelikle dizinin sınır değerleri birer değişkene yüklenmektedir. Bu değerin tam yarısının tam kısmı alınarak dizinin ortasındaki elemana ulaşım sağlanmaktadır. Bu değere karşılık gelen dizi elemanı ile aranan elemanın karşılaştırılması yapılarak istenilen bilgiye ulaşabilmek mümkün olabilmektedir. Bunun için, A19. adımda aranan kişinin ismi ISM değişkeni olarak girilmekte ve dizilerin sınır değerleri A20. adımda BS ve BT değişkenlerine atanmaktadır. A21. adımda ORT değişkenine BS ve BT değişkenlerinin yarısının tam kısmı atanarak dizinin ortasındaki değere ulaşım sağlanmaktadır. A22. adımda ORT değerine karşılık gelen isim dizisinin elemanı ile ISM değeri karşılaştırılmaktadır. Eğer bu iki değer birbirine eşit ise aranan eleman bulunmuş olacağından A26. adıma gidilerek dizi değerleri yazdırılıp işlemlere son verilmektedir. Aksi halde, aranan eleman ORT değerine karşılık gelen dizi elemanından ya küçük ya da büyük olacaktır. Bu durum A23. adımda karşılaştırılmaktadır. Eğer ORT elemanına karşılık gelen dizi elemanı ISM değerinden küçük ise BT olarak tanımlanan dizinin üst sınırı öne çekilmektedir. Dolayısıyla BT değişkeninin değeri ORT olarak alınmaktadır ve yeniden A21. adıma dönülerek işlemlere devam edilmektedir. Eğer ORT değerine karşılık gelen dizi elemanı ISM değerinden büyük ise dizinin başlangıç sınır değeri ORT değişkenine çekilerek BS değişkeninin değeri ORT olarak değiştirilmektedir. Yine

aynı şekilde A21. adıma dönülerek işlemlere devam edilmektedir. Bu işlemler istenilen elemana ulaşıncaya kadar devam etmektedir.

Bu örnekte dosyadan aranılan kişinin kayıtlı olduğu varsayılmıştır. Aksi halde A21 ve A25. adımlar arasında algoritma sürekli çalışacağından istenilen sonuca ulaşılmaktadır.

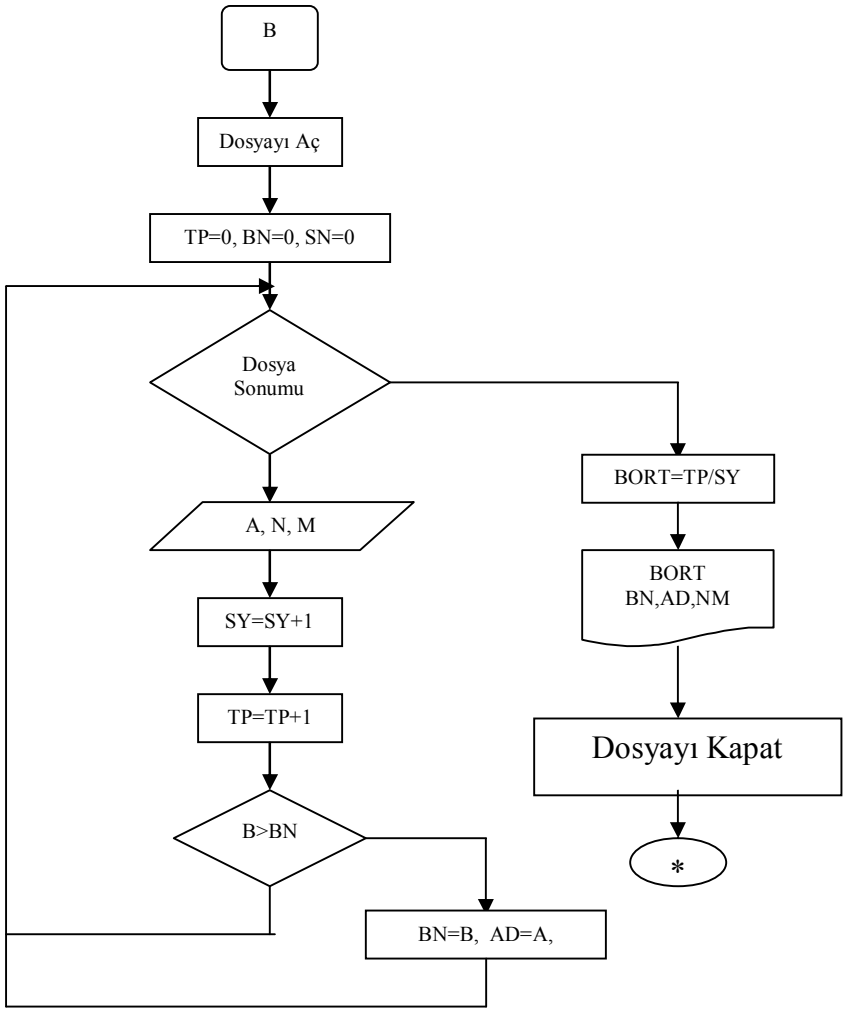
Örnek 5.2.8: Öğrencilerin adı soyadı, numarası ve bilgisayar derslerinden almış oldukları başarı notları sırasıyla erişimli bir dosyada kayıtlıdır. Bu dosyayı okutarak sınıf başarı ortalamasını ve en başarılı öğrenciyi bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. Dosyayı aç,
A3. TP=0, BN=0 ve SY=0 al,
A4. Eğer dosya sonu ise A10. adıma git,
A5. A,N ve B' yi dosyadan oku,
A6. SY=SY+1 al,
A7. TP=TP+B al,
A8. Eğer B>BN ise BN=B, AD=A ve NM=N al,
A9. A4. adıma geri dön,
A10. BORT=TP/SY al,
A11. "Sınıf ortalaması olan" BORT değerini yaz,
A12. BN, AD ve NM' yi yaz,
A13. Dosyayı kapat ve dur.

Algoritmanın Pascal dilindeki yazılımı:

```
program basari;
var
        n,b,k,tp,bn,sy:integer;
        bort:real;
        a:char[20];
        ad:char;
        dosya:text;
begin
        sy:=0;
        tp:=0;
        bn:=0;
        assign(dosya,'notlar.dat'); reset(dosya);
        while(not eof(dosya)) do begin
                read(dosya,a);
                read(dosya,n);
                read(dosya,b);
                sy:=sy+1;       tp:=tp+b;
                if(b>bn) then begin
                        bn:=b;
                        ad:=a;
                        nm:=n;
```

```
            end;
        end;
        bort:=tp/sy;
        writeln('Sinif ortalamasi:',bort);
        writeln(a,'',n,'',bn);
        close(dosya);
        readln;
end.
```



Şekil 5.2.8: Sırasal erişimli bir dosyadan sınıf ortalamasının ve en başarılı öğrencinin bulunmasına ilişkin akış şeması

A3. Adımda tanımlanan TP değişkeni, dosyadaki başarı notlarının toplanmasını sağlamak üzere tanımlanmıştır. Benzer şekilde SY değişkeni de dosyadaki kayıtlı olan bilgi sayısını belirlemek üzere tanımlanmıştır. BN değişkeni ise en başarılı öğrenciyi bulmak amacıyla bir karşılaştırmanın yapılabilmesi için tanımlanan bir değişkendir. Bu değişkene ilk değer olarak sıfır atanmıştır. Karşılaştırmalar sonucunda bu değişken en büyük başarı notu değerini alacaktır. Buna bağlı olarak da Ad değişkeni bu nota karşılık gelen ismi ve NM değişkeni de buna karşılık gelen öğrencinin numarasını taşıyacaktır. Böylelikle en yüksek nota sahip olan öğrencinin bilgileri bu değişkenlerde tutulacaktır. A4. adımda dosya sonu kontrolü yapıldıktan sonra, eğer dosya sonuna gelinmiş ise bilgiler okutulmaktadır ve SY değişkeni bir artırılarak burada okunan başarı notu TP

değişkenine ilave edilmektedir. A8. Adımda dosyadan okutulan B değeri BN değerinden büyük ise (ilk aşamada büyük olacaktır, çünkü BN değişkeninin değeri ilk aşamada 0 dır) BN değişkenine B değeri ve benzer olarak da AD değişkenine A değeri ve NM değişkenine de N değeri atanmaktadır. Bu işlemler dosya sonuna gelinceye kadar devam etmektedir. Dosya sonuna gelindiğinde TP değişkeni ile tüm başarı notlarının toplamı elde edilmiş olacaktır. Benzer şekilde, SY değişkeni ile de dosyada kayıtlı eleman sayısı elde edilmiş olacağından, A10. adımda TP değeri SY değerine bölünerek sınıf ortalaması hesaplanmış olmaktadır. Bu değerle birlikte en son elde edilen BN değeri en yüksek nota sahip öğrenciyi tanımlayacağından BN, AD ve NM değerleri yazdırılıp işlemlere son verilmektedir.

5.3. DOĞRUDAN ERİŞİMLİ DOSYALARDA ALGORİTMA VE AKIŞ ŞEMALARI

Bu kısımda doğrudan erişimli dosyalara ilişkin tüm işlemleri içeren algoritma ve akış şemaları verilmiştir. Sırasıyla bu tür dosyalarda kayıt işlemleri, istenilen kayıtlara ulaşılmasını sağlayan işlemler, istenilen bir kaydın silinmesi ya da bu kayıttaki bazı bilgilerin değiştirilmesine yönelik işlemleri içeren algoritma ve akış şemalarının oluşturulması sağlanmıştır.

Doğrudan erişimli dosyalara kayıt yapılırken belirli bir kayıt sırası yoktur. Yani kayıt numaraları arka arkaya verilecek diye bir kural yoktur. Örneğin önce 1 numaralı kayıt yapıldıktan sonra 20 numaralı kayıt arkasından yapılabilir ve isteğe göre 7 numaralı kayıt bunların arkasından yapılabilir. Tüm bunlara rağmen doğrudan erişimli dosyalarda eğer gerekli güvenlik önlemleri alınmaz ise aynı kayıt numarasına başka bilgi kaydetmek mümkün olabilmektedir. Oysa sırasal erişimli dosyalarda bu tehlike yoktur. Çünkü bilgiler arka arkaya kaydedilmektedir. Bu durumu engellemek için en güvenilir yol dosyayı belirli bir kayıt uzunluğunda hazırlamaktır. Bunun anlamı dosya içerisine kaydedilecek olan bilgilerden herhangi birisinin yazım aşamasında kullanılmayan bir karakterle yüklenmesidir. Böyle bir sistemle istenilen kayda erişildiğinde bu kayıttaki ilgili bilgi kontrol edilerek başlangıçta verilen işaretin olup olmadığı sorgulanabilir ve bu doğrultuda ilgili kaydın boş olup olmadığı anlaşılabilir.

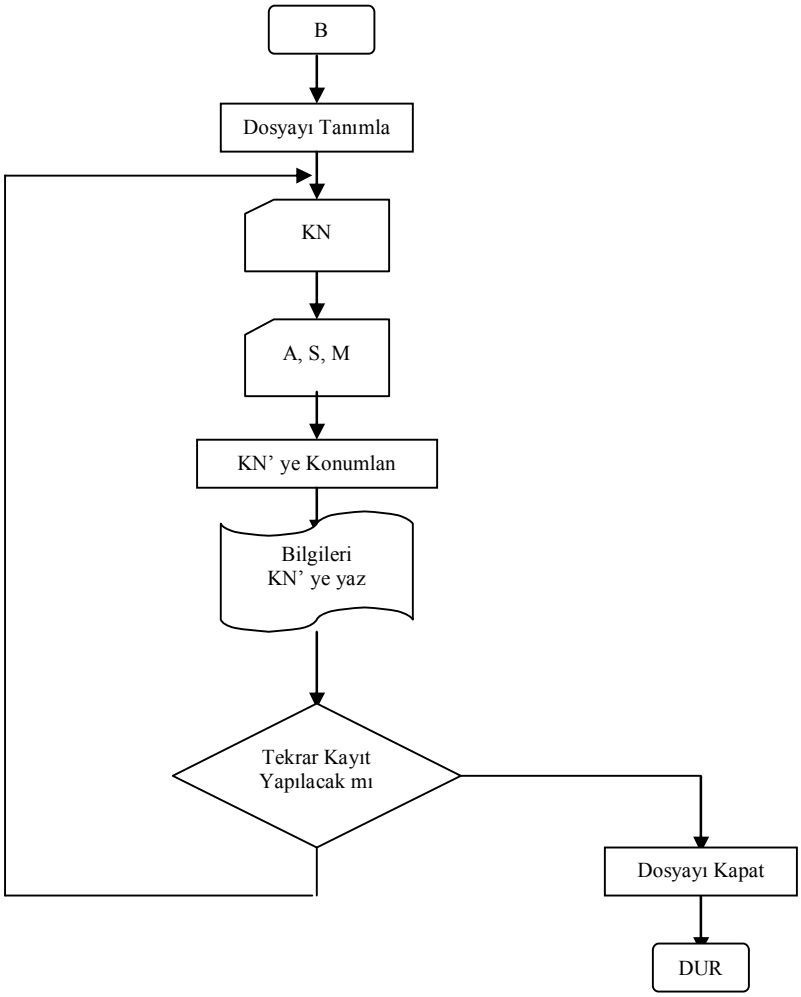
Örnek 5.3.1: doğrudan erişimli bir dosyada ad soyad, sicil numarası ve maaş bilgilerini kaydeden algoritma ve akış şemasının oluşturulması.

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.

Algoritmanın Pascal dilindeki yazılımı:

```
program kayit;
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 kayit yapilacak mi(e/h)..?');
                readln(c);
        end;
        close(dosya);
end.
```



Şekil 5.3.1: Doğrudan erişimli dosyada kayda ilişkin akış şeması

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.

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.

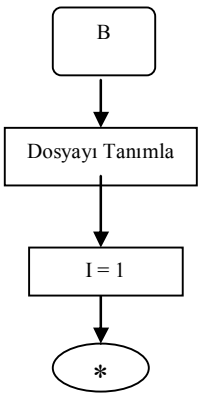
Örnek 5.3.2: Doğrudan erişimli dosyaların güvenilir kullanılabilmesi için hazırlanmasını sağlayan algoritma ve akış şemasının oluşturulması.

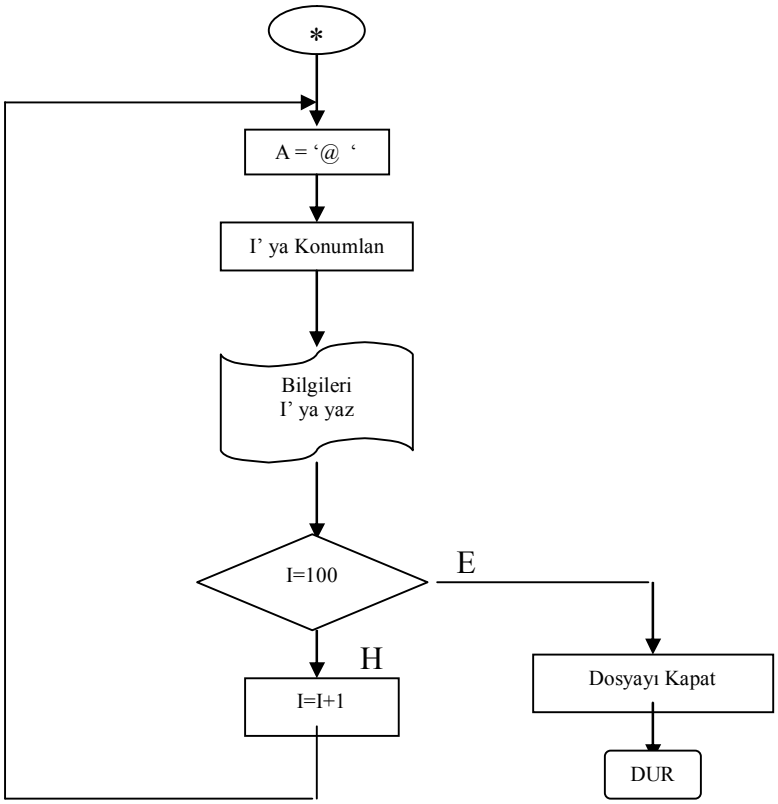
Örnekteki dosyaya en fazla 100 kişinin adı soyadı, sicil numarası ve maaşı bilgilerinin kaydedileceği düşünülerek dosya buna göre hazırlanacaktır.

A1. Başla,
A2. Dosyayı tanımla,
A3. A=’@’ al,
A4. I’ ya konumlan,
A5. Dosyaya yaz,
A6. Eğer I=100 ise a10. adıma git,
A7. I=I+1 al,
A8. A4. adıma geri dön,
A9. Dosyayı kapat,
A10. Dur.

Algoritmanın Pascal dilindeki yazılımı:

```
program hazirlama;
type
        bilgi=record
                a:string[20];
                s,m:longint;
end;
var
        d:file of bilgi;
        kay:bilgi;
        kn,i:integer;
        c:char;
begin
        assign(d,’maas.dat’);
        {$I-}; reset(d); {$I+};
        if(IOResult<>0) then rewrite(d);
        kay.a:=’@’;
        for i:=1 to 100 do begin
                seek(d,i);
                write(d,kay);
        end;
        readln;
        close(d);
end.
```





Şekil 5.3.2: Dosya hazırlayan akış şeması

Ad soyad, sicil numarası ve maaş bilgilerinin kaydedileceği doğrudan erişimli dosyanın en fazla 100 kişi kaydedilecek şekilde düzenlenmesi için öncelikle 12 den 100' e kadar artışı belirlemek üzere I sayacı tanımlanarak ilk değer olarak 1 atanmıştır. Ad soyad belirtilen A değişkenine A4. adımda @ değeri olarak atanmıştır. Burada @ karakterinin seçilmesindeki amaç, isimlerde hiçbir zaman bu karakterin kullanılmamasıdır. Bir sonraki adımda 1. kayda konumlanarak bilgiler dosyaya yazdırılmaktadır. Bu işlem I değişkeninin değeri 100 oluncaya kadar devam edecektir. Bu sayıya ulaşılınca dosya kapatılarak işlemlere son verilmektedir.

Bu işlemler sonucunda kişilerin adı soyadı, sicil numarası ve maaşı bilgilerinin girileceği doğrudan erişimli dosya kayıt için hazırlanmış olmaktadır. Bu aşamadan sonra daha güvenilir bilgi kaydı ve bilgi taraması yapılabilecektir.

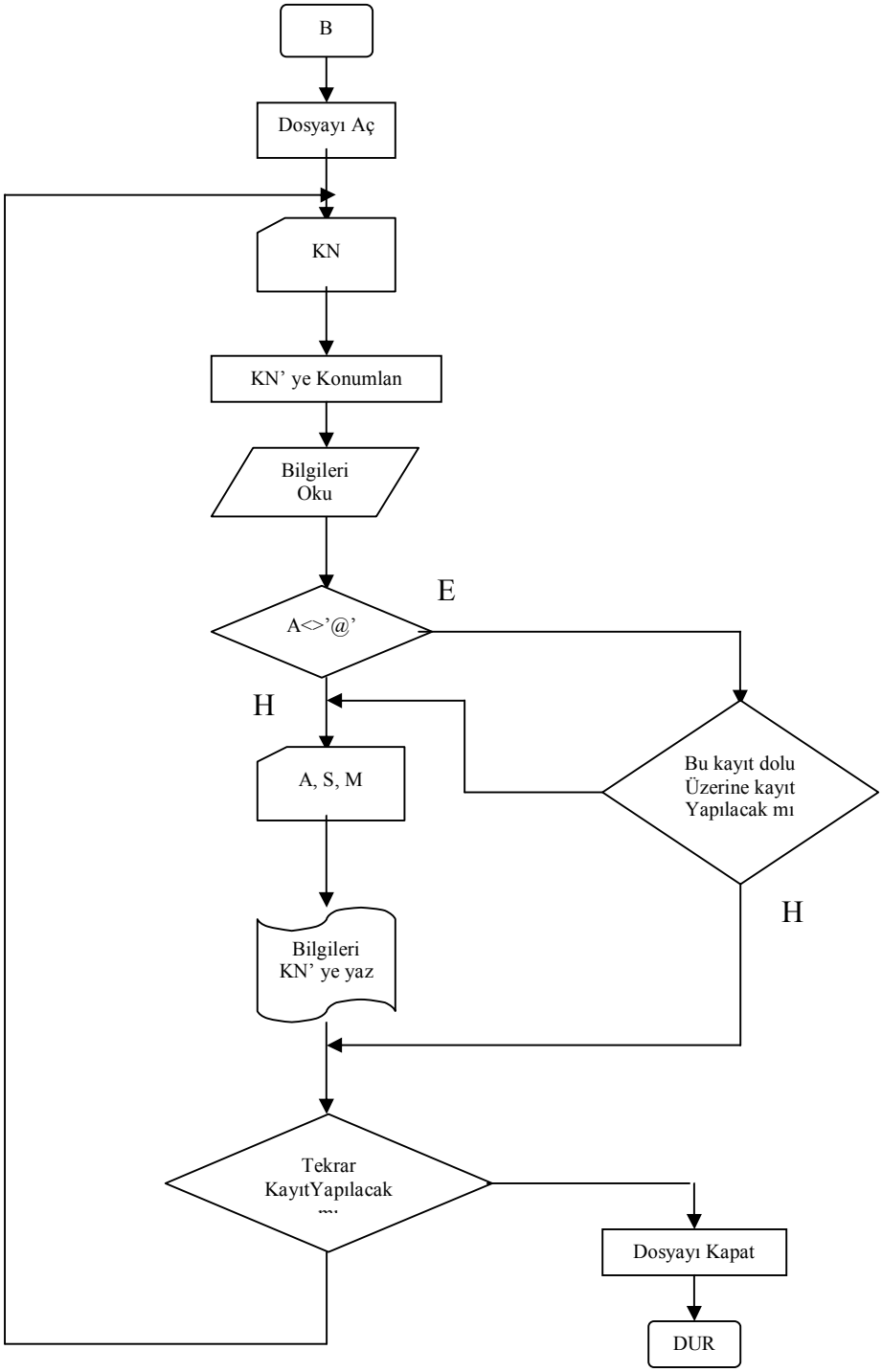
Örnek 5.3.3: Bir önceki örnekte 100 kişi için hazırlanan dosyaya kayıt yapan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. Dosyayı aç,
A3. KN' yi gir,
A4. Dosyaya konumlan,
A5. Dosyadan oku,
A6. Eğer A<>'@' ise A12. adıma git,
A7. A' yı gir,{Adı soyadı}

A8. S' yi gir,{Sicil numarası}
A9. M' yi gir,{Maaş}
A10. Dosyaya yaz,
A11. A14. adıma git,
A12. 'Bu kayıt dolu, üzerine kayıt yapılacak mı2 diye sor,
A13. Eğer 'E' ise A7. adıma geri dön,
A14. Tekrar kayıt yapılacak mı? diye sor,
A15. Eğer 'E' ise A13. adıma geri dön,
A16. Dosyayı kapat,
A17. Dur.

Algoritmanın Pascal dilindeki yazılımı:

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



Şekil 5.3.3: Kayıt için hazırlanmış bir dosyaya kayıt yapılmasına ilişkin akış şeması

Örnekte daha önceden kayıt amacıyla hazırlanmış bir doğrudan erişimli dosyaya kayıt yapılmaktadır. Bunun için A3. adımda girilen kayıt numarasına A4. adımda konumlanarak dosyadan ilgili kayıt numarasına ilişkin bilgiler okutulmaktadır. Eğer okutulan kaydın isim yerinde @ karakteri var ise bu kayıt boş olacağından buraya kayıt yapılabilecektir. Aksi halde burada @ karakterinin dışında bir karakter bulunacağından bu kayıt dolu olacaktır. Bu durum, A6. Adımda sorgulanmaktadır. Eğer isim bilgisi olan A değeri @ karakteri ile aynı değerde ise A7. adımdan itibaren bilgi girişleri yapılarak

bu bilgiler dosyada ilgili kayıt numarasının adresine yazdırılmaktadır. Aksi halde bu kayıt dolu olacağından A12. Adıma gidilerek üzerine kayıt yapılıp yapılmamasına ilişkin sorgulama yapılmaktadır. Eğer üzerine yeni kayıt yapılacaksa A7. adımdan itibaren işlemlere devam edilmektedir. Aksi halde A14. adıma geçilerek başka kaydın yapılıp yapılmayacağı sorgulanmaktadır. Benzer şekilde normal kayıt yapıldıktan sonra da yine A14. adıma gidilerek tekrar kayıt için sorgulama yapılmaktadır. Eğer yeni kayıt yapılacaksa A3. adımdan itibaren işlemlere devam edilmektedir. Aksi halde dosya kapatılarak işlemlere son verilmektedir.

Örnek 5.3.4: Bir önceki örnekte kayıt yapılan doğrudan erişimli dosyadan istenilen bir kaydın silinmesini sağlayan algoritma ve akış şemasının oluşturulması.

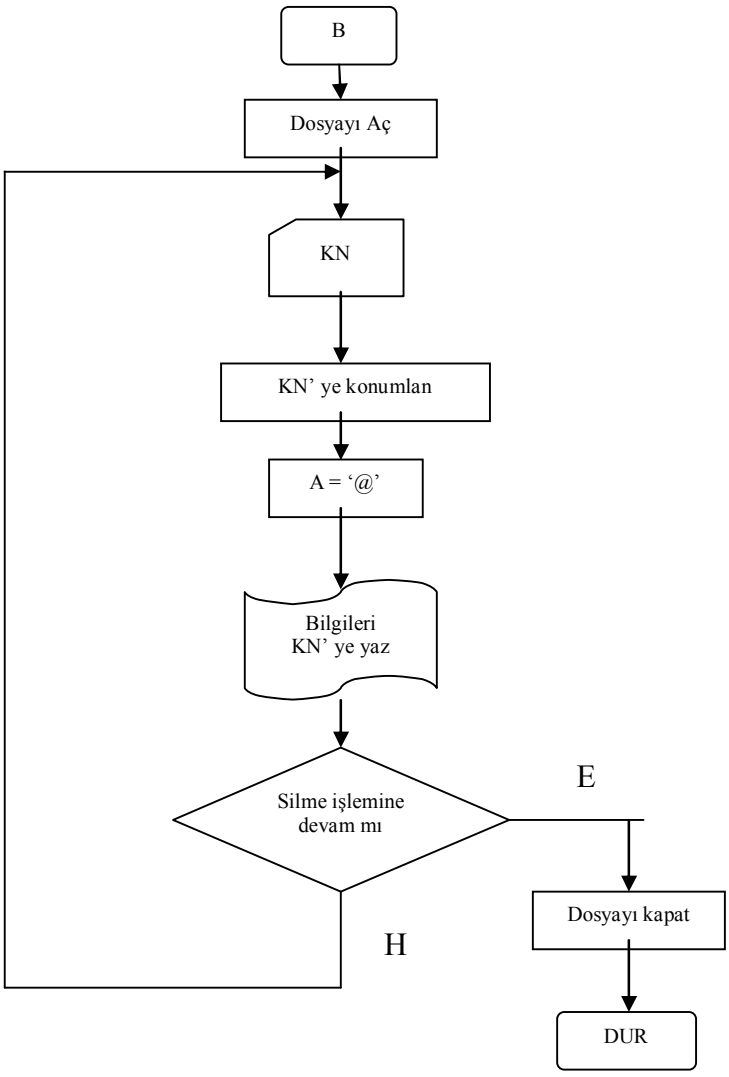
Daha önceden hazırlanarak kayıt yapılan doğrudan erişimli dosyadan istenilen bir kaydın silinmesi için yapılacak işlemler son derece kolaydır. Sırasal erişimli dosyalarda olduğu gibi ikinci bir dosya, buradaki ilgili bilgi yerine hazırlık aşamasında kullanılan karakter yüklenerek bilgi iptal edilmiş olur.

A1. Başla,
A2. Dosyayı aç,
A3. KN' yi gir {Silinecek kişinin kayıt numarası},
A4. KN' ye konumlan,
A5. A='@' al,
A6. Dosyaya yaz,
A7. Silinecek başka kayıt var mı?
A8. Eğer 'E' ise A3. adıma git,
A9. Dosyayı kapat,
A10. Dur.

Algoritmanın Pascal dilindeki yazılımı:

```
program iptal;
type
        bilgi=record
                a:string[20];
                s,m:longint;
end;
var
        d:file of bilgi;
        kay:bilgi;
        kn:integer;
        c:char;
begin
        assign(d,'maas.dat');
        {$I-}; reset(d); {$I+};
        if(IOResult<>0) then rewrite(d);
        c:='e';
        while(c<>'h') do begin
                write('Silinecek kayit numarasini giriniz..:');readln(kn);
                seek(d,kn);
```

```
            read(d,kay);
            if(kay.a='@') then begin
                  kay.a:='';
                  seek(d,kn);
                  write(d,kay);
            end;
            write('Tekrar kayit yapilacak mi(e/h)..?');readln(c);
      end;
      close(d);
end.
```



Şekil 5.3.4: Hazırlanarak kayıt yapılan bir dosyadan kayıt iptal edilmesine ilişkin akış şeması

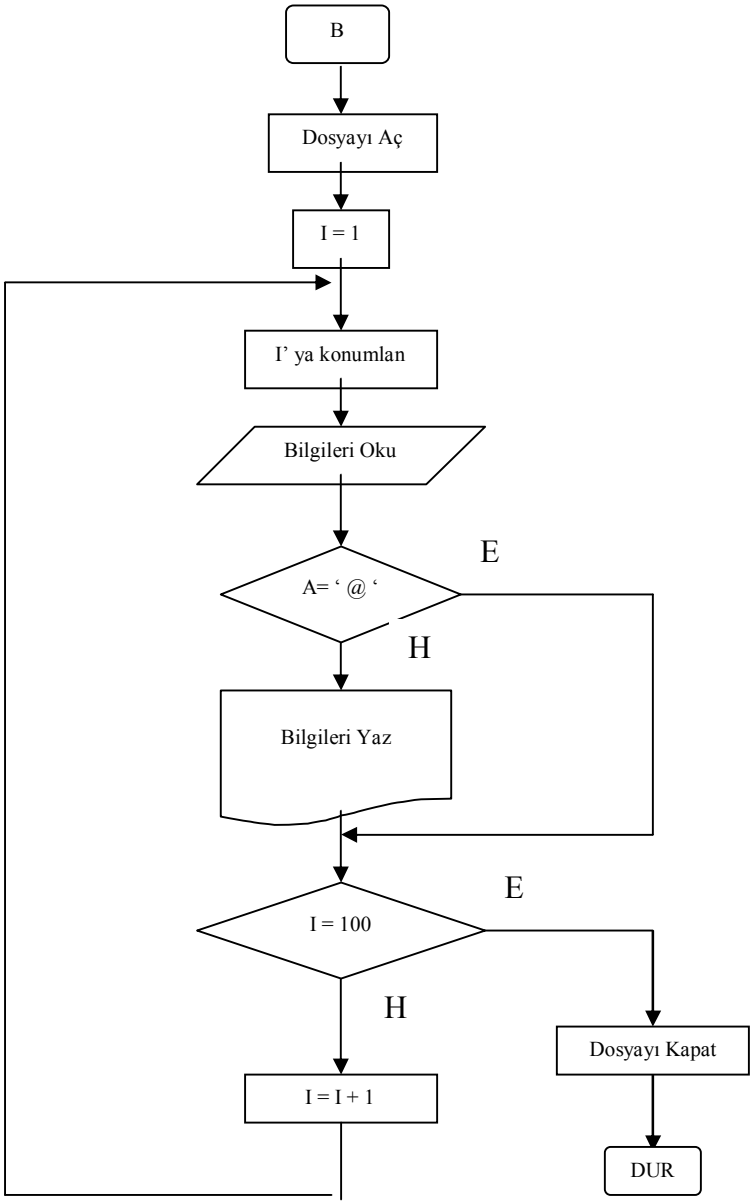
A3. adımda silinecek kişiye ilişkin kayıt numarası girildikten sonra A4. adımda bu kayıt numarasına konumlanılarak bir sonraki adımda A olarak tanımlanan isim değişkenine @ karakteri atanmaktadır. Bu değer A6. adımda dosyaya yazdırılarak ilgili kayıt iptal edilmiş olmaktadır. Daha sonra tekrar iptal edilecek kaydın olup olmadığı sorgulanarak bu doğrultuda işlemlere devam edilmektedir.

Örnek 5.3.5: Örnek 5.2.4'te ad soyad, sicil numarası ve maaş bilgileri kaydedilen kişilerin bilgilerini okuyarak ekrana yazdırılmasını sağlayan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. Dosyayı aç,
A3. I=1 al,
A4. I' ya konumlan,
A5. Dosyadan oku,
A6. Eğer A='@' ise A8. adıma git,
A7. Bilgileri ekrana yaz,
A8. Eğer I=100 ise A11. adıma git,
A9. I=I+1 al,
A10. A4. Adıma geri dön,
A11. Dosyayı kapat,
A12. Dur.

Algoritmanın Pascal dilindeki yazılımı:

```
program raporlama;
type
        bilgi=record
                a:string[20];
                s,m:longint;
end;
var
        d:file of bilgi;
        kay:bilgi;
        kn,i:integer;
        scl:longint;
        c:char;
begin
        assign(d,'maas.dat');
        {$I-}; reset(d); {$I+};
        if(IOResult <>0) then rewrite(d);
        for i:=1 to 100 do
        begin
                seek(d,i);
                read(d,kay);
                if(kay.a='@') then
                        writeln(kay.a,'',kay.s,'',kay.m);
        end;
        readln;
        close(d);
end.
```



Şekil 5.3.5: Hazırlanarak kayıt yapılan bir dosyadan bilgilerin ekrana yazdırılmasına ilişkin akış şeması

Verilen dosya 100 kayıt için daha önceden hazırlanarak kayıtların yapıldığı bilinmektedir. Bu doğrultuda A3. Adımda I değişkeni 1 değerini alarak işlemlere başlanmaktadır. A4. Adımda 1. Kayda konumlanarak bu kaydın boş olup olmadığı sorgulanmaktadır. Eğer buradaki A değeri @ karakterine eşit ise bu kayıt boş olacağından ekrana yazdırılmaktadır. Bu durum A6. Adımda sorgulanmaktadır. A değeri @ karakterinden farklı ise A7. Adım devreye girerek bu bilgiler ekrana yazdırılmaktadır. Bu işlemler I değişkeninin değeri 100 oluncaya kadar devam etmektedir. Sonuçta dosya kapatılarak işlemlere son veril mektedir.

EK

ÇEŞİTLİ KONULARDA ALGORİTMA ÖRNEKLERİ

Bu bölümde değişik problemlere ilişkin Pascal dili ile yazılmış program örnekleri verilmektedir. Daha önceden anlatılan konuları kendinize örnek alarak akış diyagramlarını çizebilirsiniz.

Örnek 6.1 : Girilen iki sayının OBEB ve OKEK'ini bulan pogramın yazılması.

```
Uses crt;

Var
        a,b,t,k:byte;
Begin
        Clrscr;
        Write('1. sayıyı giriniz : ');    readln(a);
        Write('2. sayıyı giriniz : ');    readln(b);
        t:=a*b;
        İf (a<b) then begin
                k:=a;
                a:=b;
                b:=k;
        end;
        repeat
                k:=b;
                b:=a mod b;
                a:=k;
        until b=0;
        write('Obeb : ',a);
        write('Okek : ',t/a:4:2);
        readln;
end.
```

Örnek 6.2 : Maksimum 20 adet rastgele oluşturulan sayıları Buble Sort (Bublesiralama) ve Quick Sort (Hızlı sıralama) yöntemleri ile sıralayan programın yazılması.

```
Uses dos,crt;
type datatp=integer;
const max=20;
        MAX_N=max;

var
        a:array[1..max] of datatp;
        t:datatp;
```

```
        iter:longint;

procedure hizli_siralama(l,r,v:integer);
var
        i,j:integer;
begin
        inc(iter);
        if r>1 then begin
                v:=a[r];
                i:=l-1;
                j:=r;
                repeat
                        repeat
                                inc(i);
                        until (a[i]>=v) or (i=r);
                        repeat
                                dec(j);
                        until (a[j]<=v) or (j=l);
                        t:=a[i];
                        a[i]:=a[j];
                        a[j]:=t;
                until j<=i;
                a[j]:=a[i];
                a[i]:=a[r];
                a[r]:=t;
                hizli_siralama(l,i-1,v div 2);
                hizli_siralama(i+1,r,v+(v div 2));
        end;
end;

procedure BubleSiralama(n1,n2:integer);
var
        t:datatp;
        i,j:integer;
begin
        for j:=n1+1 to n2 do begin
                for i:=n1 to n2-1 do begin
                        inc(iter);
                        if (a[i]>a[j]) then begin
                                t:=a[i];
                                a[i]:=a[j];
                                a[j]:=t;
                        end;
                    end;
          end;
end;
var
        h,m,s,ss:word;
        e:integer;
begin
```

```
clrscr;
{BubleSıralama bölümü}

writeln(max,'tane rastgele sayi olusturuluyor...');
randomize;
for e:=1 to max do a[e]:=round(random*MAX_N);
writeln('....olusturuldu.');
write('Girmek icin Entere basin');  readln;
for e:=1 to max do write(a[e]:3,' '); writeln;
writeln('"Buble Sıralama" ile siralanacak');
write('Entere basiniz'); readln;
gettime(h,m,s,ss);
writeln('** Zaman:',h,':',m,':',s,':',ss);
writeln(max,'tane kayit,"Buble S   ralama" ile siralaniyor...');
iter:=0;
bublesiralama(1,max);
writeln(iter,'adimda siralandi');
gettime(h,m,s,ss);
writeln('** Zaman:',h,':',m,':',s,':',ss);
write('Girmek icin Entere basin'); readln;
for e:=1 to max do write(a[e]:3,' ');  writeln;
write('Entere basiniz'); readln;

{Hızlı sıralama bölümü}

writeln(max,'tane rastgele sayi olusturuluyor...');
randomize;
for e:=1 to max do a[e]:=round(random*MAX_N);
writeln('....olusturuldu.');
write('Girmek icin Entere basin');
readln;
for e:=1 to max do write(a[e]:3,' ');
writeln;
writeln('"Hızlı sıralama" ile sıralanacak');
write('Entere basiniz');
readln;
gettime(h,m,s,ss);
writeln('** Zaman:',h,':',m,':',s,':',ss);
writeln(max,'tane kayit,"hızlı sıralama" ile siralaniyor...');
iter:=0;
cabuk_siralama(1,max,(MAX_N div 2));
writeln(iter,'adimda siralandi');
gettime(h,m,s,ss);
writeln('** Zaman:',h,':',m,':',s,':',ss);
write('Girmek icin Entere basin');
readln;
for e:=1 to max do write(a[e]:3,' ');
writeln;
write('Entere basiniz'); readln;
end.
```

Örnek 6.3 : a(x**2)+bx+c=0 denkleminin (a,b,c değerleri dışarıdan girilecek) köklerini bulan programın yazılması.

```
program kok_bul;
uses crt;
var
        a,b,c:integer;
        d,x1,x2:real;
begin
        clrscr;
        writeln('"a(x**2)+bx+c=0" denkleminin a,b,c degerlerini giriniz :');
        readln(a,b,c);
        d:=b*b-4*a*c;
        if d>0 then begin
                x1:=(-b+sqr(d))/2*a;
                x2:=(-b-sqr(d))/2*a;
                writeln('x1= ',x1:4:2,'x2= ',x2:4:2);
        end
        else if d=0 then begin
                x1:=-b/2*a;
                writeln('x1=x2=',x1:4:2);
        end
        else if d<0 then writeln('Kök yok');
        readln;
end.
```

Örnek 6.4 : Klavyeden girilen iki sayı arasındaki çift ve tek sayıların toplamını bulan programın yazılması.

```
uses crt;
var
        a,b,c,d,e,f:integer;
begin
        clrscr;
        writeln('1. sayıyı gir');     readln(a);
        writeln('2.sayıyı gir');     readln(b);
        d:=0;   e:=0;    f:=0;
        for c:=a to b do begin
                write(c:2);
                d:=d+c;
                if (c mod 2) = 0 then e:=e+c;
                if (c mod 2) = 1 then f:=f+c;
        end;
        writeln('a dan b ye kadar olan sayıların toplamı',d);
        writeln('cift sayıların toplamı   ',e);
        writeln('tek sayıların toplamı    ',f);
        readln;
```

end.

Örnek 6.5 : Klavyeden girilen bir dizi tamsayı içerisinde istenilen sayıyı (dizinin kaçıncı elemanı olduğunu) bulan programın yazılması.

```
Uses crt;
var
        no:array[1..5] of integer;
        ara,i,j,g:integer;
begin
        clrscr;
        for i:=1 to 5 do begin
                write('sayi');     readln(no[i]);
        end;
        for i:=1 to 5 do
                for j:=i+1 to 5 do begin
                        if no[i] < no[j] then begin
                                g:=no[i];
                                no[i]:=no[j];
                                no[j]:=g;
                        end;
                end;
        for i:=1 to 5 do   begin
                writeln(no[i]);
        end;
        writeln('aranılacak sayıyı gir');      readln(ara);
        if no[i]=ara then begin
                writeln(i ,'   eleman',ara);
        end
        else writeln('dizide yok');
        readln;
end.
```

Örnek 6.6 : Klavyeden girilen bir dizi tamsayı eleman içerisindeki en büyük ve en küçük elemanı bulan programın yazılması.

```
uses crt;
var
        sayi,enb,enk:longint;
        i:byte;
begin
        clrscr;
        i:=0;
        while i<5 do begin
                i:=i+1;
                write('sayiyi gir');       readln(sayi);
                if i=1 then begin
                        enb:=sayi; enk:=sayi;
                end;
```

```
            if sayi<enk then enk:=sayi;
            if sayi > enb then enb:=sayi;
        end;
        writeln('en buyuk sayi',enb);
        writeln('en kucuk sayi',enk);
        readln;
end.
```

Örnek 6.7 : Klavyeden girilen bir karakterin cinsini (büyük harf, küçük harf, sayı) bulan programın yazılması.

```
uses crt;
var
        a:char;
begin
        clrscr;
        writeln('bir karekter giriniz');
        readln(a);
        case  a of
            '0'..'9': writeln('sayı girdiniz');
            'a'..'z':writeln('küçük harf ');
            'A'..'Z':writeln('büyük harf');
        else
            writeln('Özel karekter girdiniz');
         end;
         readln;
end.
```

SONUÇ ve ÖNERİLER

Bilgisayar teknolojisinde meydana gelen hızlı gelişmeyle birlikte bilginin, teknolojinin yeri oldukça önem kazanmıştır. Bilgisayarların istenilen amaçlara cevap verebilmesi için doğru programlanması ve problemleri çözmek için etkili algoritmaların geliştirilmesi gerekmektedir. Dolayısıyla bu tip problemlerin çözümü için algoritmaların iyi analiz edilmesi gerekmektedir.

Özellikle son yıllarda her sektördeki güvenlik sistemleri ve bilgi güvenliğinin önemi artmıştır. Bu nedenle; bilgi güvenliğini sağlayacak algoritmalar (şifreleme, sıralama, arama algoritmaları,yapay zeka algoritmaları, vb.) özellikle matematik, bilgisayar, elektronik ve haberleşme ana bilim dallarında sıkça kullanılmaktadır

Bu gelişmelere ayak uydurmak için, algoritmalar hakkında yaptığımız araştırmalardan özet bir proje çalışması hazırladık. Bizi bu çalışmayı hazırlamaya iten sebep, Türkiye' de algoritmalar hakkında sınırlı sayıda Türkçe kaynak olmasıdır. Amacımız, bu boşluğu doldurucu nitelikte bir çalışma hazırlamaktır. Böyle bir kitabın bilgisayar programcılığında okuyan öğrenciler için çok gerekli ve yararlı olacağı düşüncesindeyiz.

Projemizde özellikle Algoritmaların genel yapısı üzerinde detaylı olarak durulmuş ve Pascal dilinde verilen örneklerle de desteklenmiştir. Umarız bu kaynak, bizden sonra okuyacak olan tüm öğrenci arkadaşlarımıza yardımcı nitelikte olur.

Son olarak bize emeği geçen tüm öğretmenlerimize teşekkürü bir borç biliriz.

Saygılarımızla...

KAYNAKLAR

ÇELİKKOL, Soner, “Programlamaya Giriş ve Algoritmalar”, Rize, 2001

AKGÖBEK Ömer, “Turbo Pascal ve Programlama Sanatı (4.0 – 5.0 – 6.0 – 7.0)”, Beta Basım Yayım Dağıtım A.Ş., İstanbul,1996.

ALTAN Naci, “Turbo Pascal 7.0”, Alfa Basım Yayım Dağıtım A.Ş.,İstanbul,1993.

BORLAND, Turbo Pascal, User’s Guide.

TEKTAŞ Mehmet ve TOPUZ Vedat, ”Algoritmalar ve Güvenlik Sistemleri”, İstanbul, 2001.

GÜVEN Zülkif ve ERVERDİ A. Taylan, “Pascal ve Program Geliştirme”, Sürat A.Ş, İzmir, 2000.