DİJİTAL KÜTÜPHANE KANALI

Keyifli Okumalar Dileriz

Binlerce kategoride milyonlarca PDF e-kitap, ders kitapları, dergiler ve romanlara platformumuz üzerinden ücretsiz ulaşabilirsiniz.

RESMİ WEB SİTEMİZ

https://rantey.org

Yedek ve Duyuru Kanallarımız
Sitemize erişim engeli gelmesi veya adres değişikliği durumunda aşağıdaki kanallardan güncel giriş adresine erişebilirsiniz:

• Telegram: https://t.me/birkitapbir

• WhatsApp Kanalımız : https://whatsapp.com/channel/0029VbDHv8uE50Us4IvMoc0Y

BİLGİSAYAR

ÇAĞDAŞ TEKNOLOJİ

ANSİKLOPEDİSİ



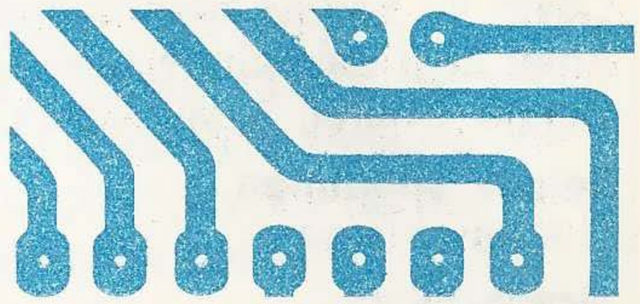
İletişim Yayınları

Ücretsiz PDF Kitap Arsivi - https://rantey.org

Ücretsiz PDF Kitap Arşivi - https://trankey.org

BİLGİSAYAR

ÇAĞDAŞ TEKNOLOJİ ANSİKLOPEDİSİ



Cilt.6

Ücretsiz PDF Kitap Arsivi - https://rantey.org

BİLGİSAYAR

ÇAĞDAŞ TEKNOLOJİ ANSİKLOPEDİSİ

İletişim Yayıncılık A.Ş. adına sahibi
Vedat Çakmak

Genel Yönetmen
Murat Belge

Yayın Yönetmeni
İskender Savaşır

Sorumlu Yazı İşleri Müdürü
Ahmet Ziya Bayman

Danışma Kurulu
Kemal İnan, Ayşe Şenatalar, Taha Parla

Yayın Kurulu
Vedat Çakmak, Bilge Karasu, İskender Savaşır

Çeviri ve Yayına Hazırlayanlar
Amet Ziya Bayman, Nilüfer Güngören,
Cengiz Onural, Ayşe Özberki

Teknik Yönetmen
H.Miray Yüzgeç

Grafik/Pikaj
Süleyman Hakan

Dizgi
Hüsnü Abbas, Nuri Abbas,
Hamit Balaban, Zuhal Yolgun

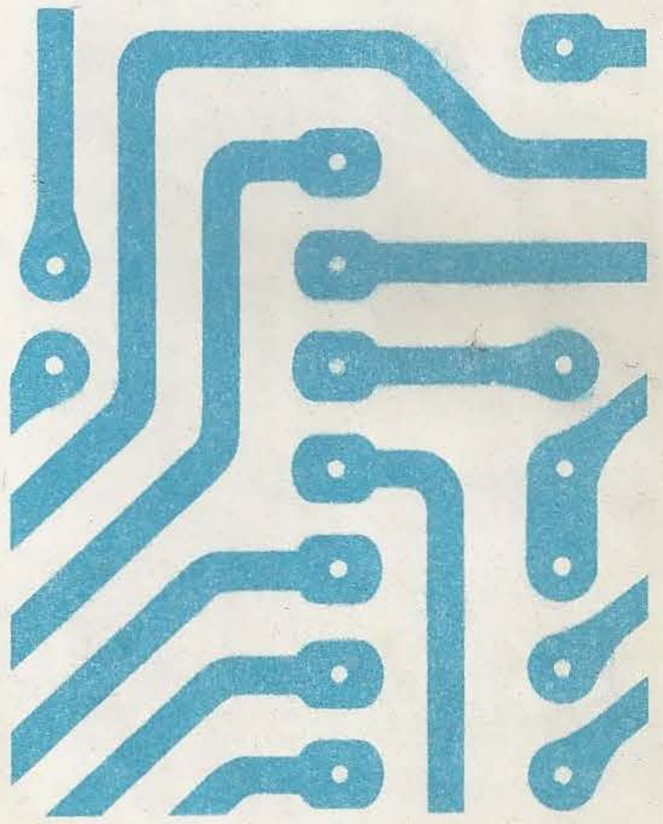
Baskı
Çiftay Matbaacılık Ltd. Şti.

© *Armando Curcio Editore, 1984*

Özgün adı: **"Basic, Enciclopedia dell'Informatica, dei Mini e Personal Computer"**
Sante Senni yönetiminde, E.G.S. Şirketi tarafından Giulia Abriani, Maria Albergo, Giovanni Aloisi, Vittoria Antinori, Alberto Berni, Carla Bertini, Ricardo Cantani, Maria Rita Ciucci, Gabriella Costarelli, Patrizia Fazzio, Renato Lazzarini, Marcella Mareaccini, Gianni Mazzoleni, Rolando Mazzoni, Piergiorgio Palma, Rosella Pozza, Laura Salvini, Roberto Sed, Ugo Spezice ve Graziella Tassi'nin katkılarıyla hazırlanmıştır.

İletişim Yayınları
Klodfarer Caddesi, İletişim Han, Cağaloğlu, İstanbul
Tel: 520 14 53/54/55

© Bütün hakları İletişim Yayıncılık A.Ş.'ye Aittir.



Ücretsiz PDF Kitap Arsivi - https://rantey.org

Programın başlangıç bölümü (10. ve 40. satırlar arası) ekranın parametrelerini tanımlar. N = 5, S = 153, E = 265, O = 15 değerleri kullanılan aların sınırlarını ekran noktası olarak belirler X0 = 140 veY 0 ÷ 80 (Siprel 2010) değerleri ise koordinatların başlangıç noktasını **(origin)** sunarlar. N = Kuzey, S = Güney, E = Batı, O = Doğu terimleri asal noktalara işaret eder ve grafik gösterim için kullanılan ''pencere''yi saptar. 140, 150 ve 160 komutları XA,YA ve XB,YB değerlerini ekran noktası olarak hesaplar.

X0 ve Y0'ın değerleri akış şemasındakinden farklıdır. Çünkü akış şeması genel bir durumu göstermektedir. Burada eğrinin her bir noktasının koordinatları X0 = 0 ve Y0 = 0 koordinatlı başlangıç noktası O'ya göre saptanmıştır. Buna karşılık dökümde ekranda merkezlenen bir gösterim elde etmek için başlangıç noktası 140, 80 olarak belirlenmiştir.

Y koordinatının, başlangıç noktası ile doğru denkleminde kullanılan değerler arasındaki farktan hareketle hesaplandığına dikkat edelim. Bu fark kullanılan makinede Y ekseninin ekranın sol üst köşesinden başlamasından kaynaklanmaktadır. M20 Olivetti makineleri gibi diğer makinelerde + işaretine gerek duyabilir.

Döngü 120. ve 190. satırlar arasında gerçekleştiği halde, görüntüleme adımı 110. satırda hesaplanmıştır. 160. satır ekran noktası şeklinde ifade edilen koordinatların değerlerinin 20. satırdaki en büyük değerleri aşıp aşmadığını kontrol eder. Döngünden 190. satırda yer darlığı nedeniyle NEXT I yerine NETX kısaltması kullanılmıştır. Aynı şekilde programı başa döndürmek için her makinede elverişli olmayan değişik bir sözdizimi kullanılmıştır. Bu durumda 190. satırı tamamlamak ve 230. satırı;

```
IF S$ = ''S'' THEN GOTO 50
```

şeklinde yazmak gerekir.

Bunun yanı sıra programı başka makinelere taşımak için aşağıda verilen komutları değiştirmek gerekmektedir:

HOME Ekranın silinmesi ve imlecin yukarıda solda konumlandırılması (satır 200)
TABn n. satırda duracak şekilde yapılan dikey tablo (satır 210)
HPLOT Koordinatları komutun içinde bulunan iki nokta arasındaki doğru parçasının çizimi (satır 1010)
HGR Grafik modun harekete geçirilmesi (satır 1000)

Sunulan program bir doğrunun çizimi için en rasyonel yol değildir. Genellikle her noktanın koordinatlarının ayrı ayrı hesaplanması gereksizdir. Demek ki X1,Y1 ve XF,YF koordinatlı noktalar arasında bir doğru parçası elde etmek için 100 ile 190 arasında bulunan komutların tümü tek bir komut ile ifade edilebilir.

Bu örnekte programı genelleştirmek için bir döngü ekledik. Bu şekilde doğrudan farklı olan herhangi bir fonksiyonun grafiğini elde etmek için 40. satırı değiştirmek yeterlidir.

Kartezyen Eksenlerin Görüntülenmesi

Bir fonksiyonun grafik gösterimi için bu fonksiyonun kartezyen eksenlerinin görüntülenmesi gerekir.

Akış şemasında gösterilen girişte temin edilmesi gereken parametreler, daha önceden bildiğimiz anlamlarını taşırlar. Alt-program, eksenleri tıpkı başlangıç noktası olan X0,Y0 noktasında kesişen iki dikey doğru parçasını çizermiş gibi çizer (bkz. sayfa 1444). Bu alt-program aynı zamanda uçlarda eksenlerin yönünü belirten okları da çizer, ayrıca da her bir eksen üzerindeki bölümleri belirtir.

''Ok'' simgesi sayfa 1444'teki şekilde gösterildiği üzere küçülen uzunlukta ve birbirine paralel doğru parçalarının çizimi ile elde edilir (dökümün 1030 ve 1040 komutları). Bu amaçla bir üçgene de başvurulabilir, ancak estetik açıdan sonuç daha çirkin olur. Yine de şekilde bu ikinci çözümde gösterilmiştir.

Y koordinatının en büyük pozitif değerine N derser, komutlar üç doğru parçasının çizimi ne indirgenmiş olur:

```
Doğru parçası 1:X0—1,N'den X0—1,K 'ye
Doğru parçası 1:X0+1,N'den X0,K—3 'ye
Doğru parçası 3:X0,N—3'den X0—1,K 'ye
```

Bu komutlar tek bir program satırı ile sunulabilir:

```
HPLOT X0—1,N TO X0+1,N TO X0,N—3
TO X0—1,N
```

Eksenlerin geriye kalan diğer 3 ucu (S,E,O) için gereken komutlar bu komutun aynısıdır.

Iki Nokta Yardımıyla Bir Doğrunun Çizimi

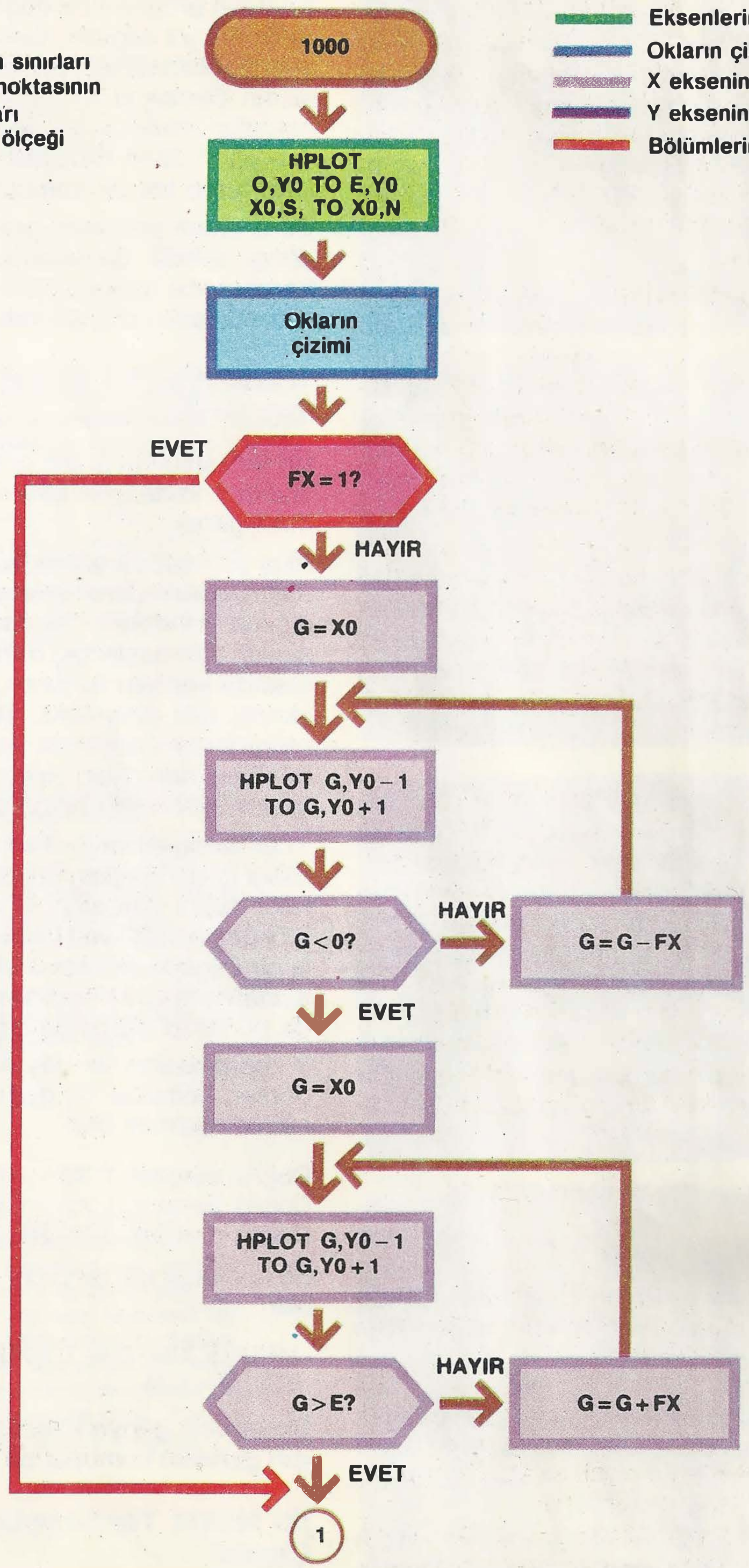
Birçok grafik uygulamada, elimizde koordinat-

Ücretsiz PDF Kitap Arsivi - https://rantey.org

EKSENLERİN ÇİZİMİNİN AKIŞ ŞEMASI

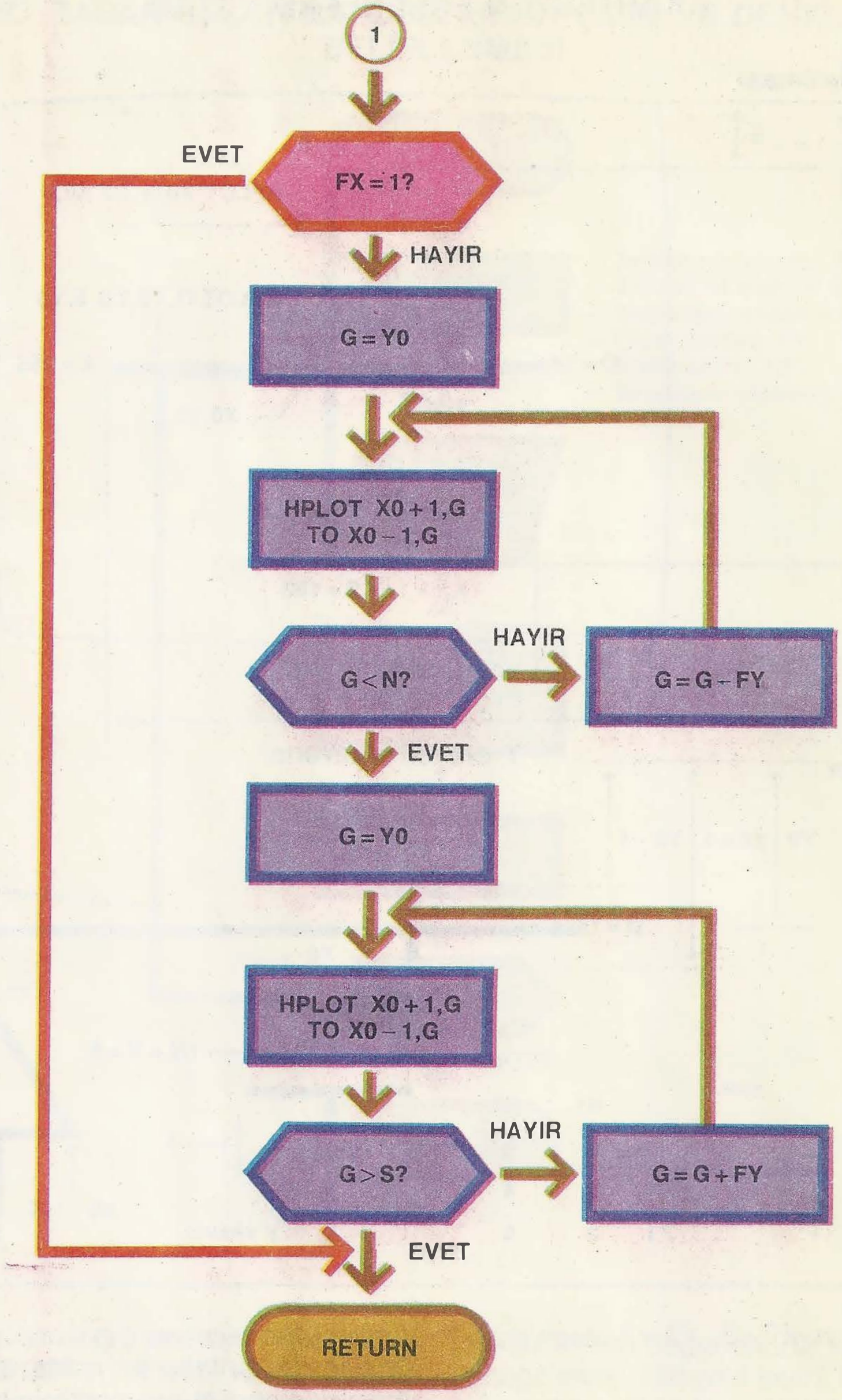
Girişler:

N, S, E, O	Pencerenin sınırları
X0,Y0	Başlangıç noktasının koordinatları
FX,FY	Eksenlerin ölçeği



Eksenlerin çizimi
Okların çizimi
X ekseninin bölümleri
Y ekseninin bölümleri
Bölümlerin atlanması

Ücretsiz PDF Kitap Arsivi - https://rantey.org



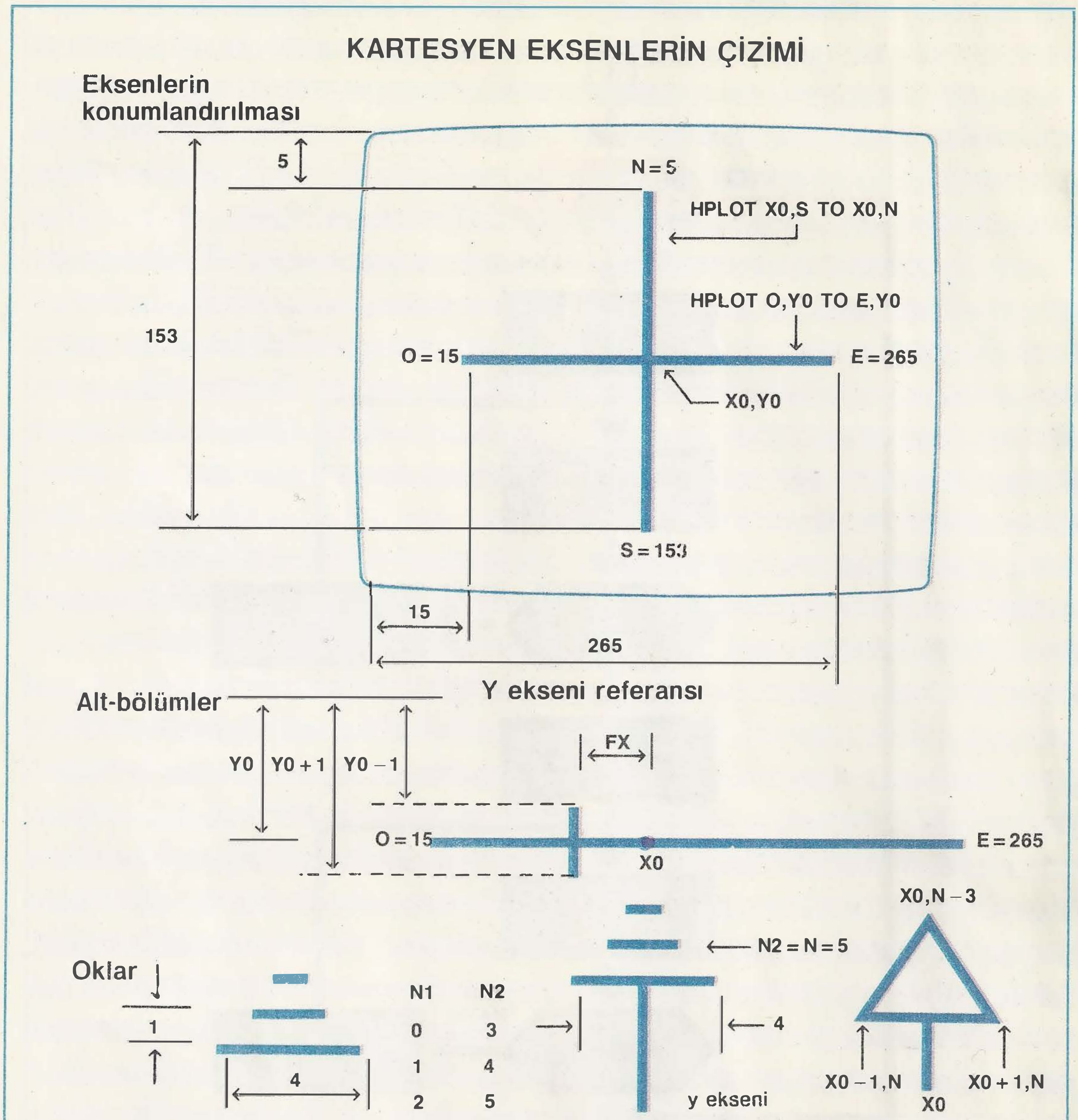
ları olan iki noktadan geçen bir doğrunun denklemini çıkarsamımızı sağlayacak bir alt-programın elimizin altında bulunması yararlı olur. Bir doğrunun genel denklemi $Y = A*X + B$ şeklinde yazıldığına göre X1,Y1 ve X2,Y2 noktalarından geçen doğrunun denklemini bulmak için;

$$A = \frac{Y2 - Y1}{X2 - X1}$$

$$B = \frac{(X1 * X2) - (Y2 * X1)}{X2 - X1}$$

formülleri uygulanır.

Ücretsiz PDF Kitap Arsivi - https://rantey.org



X1 = 2, Y1 = 2.5 ve X2 = 5, Y2 = 4 olsun o zaman;

$$A = \frac{4 - 2.5}{5 - 2} = \frac{1.5}{3} = 0.5$$

$$B = \frac{(2.5 * 5) - (4 * 2)}{5 - 2} = \frac{12.5 - 8}{3} = \frac{4.5}{3} = 1.5$$

olur.

Demek ki doğrunun denklemi;

$$Y = 0.5 * X + 1.5$$

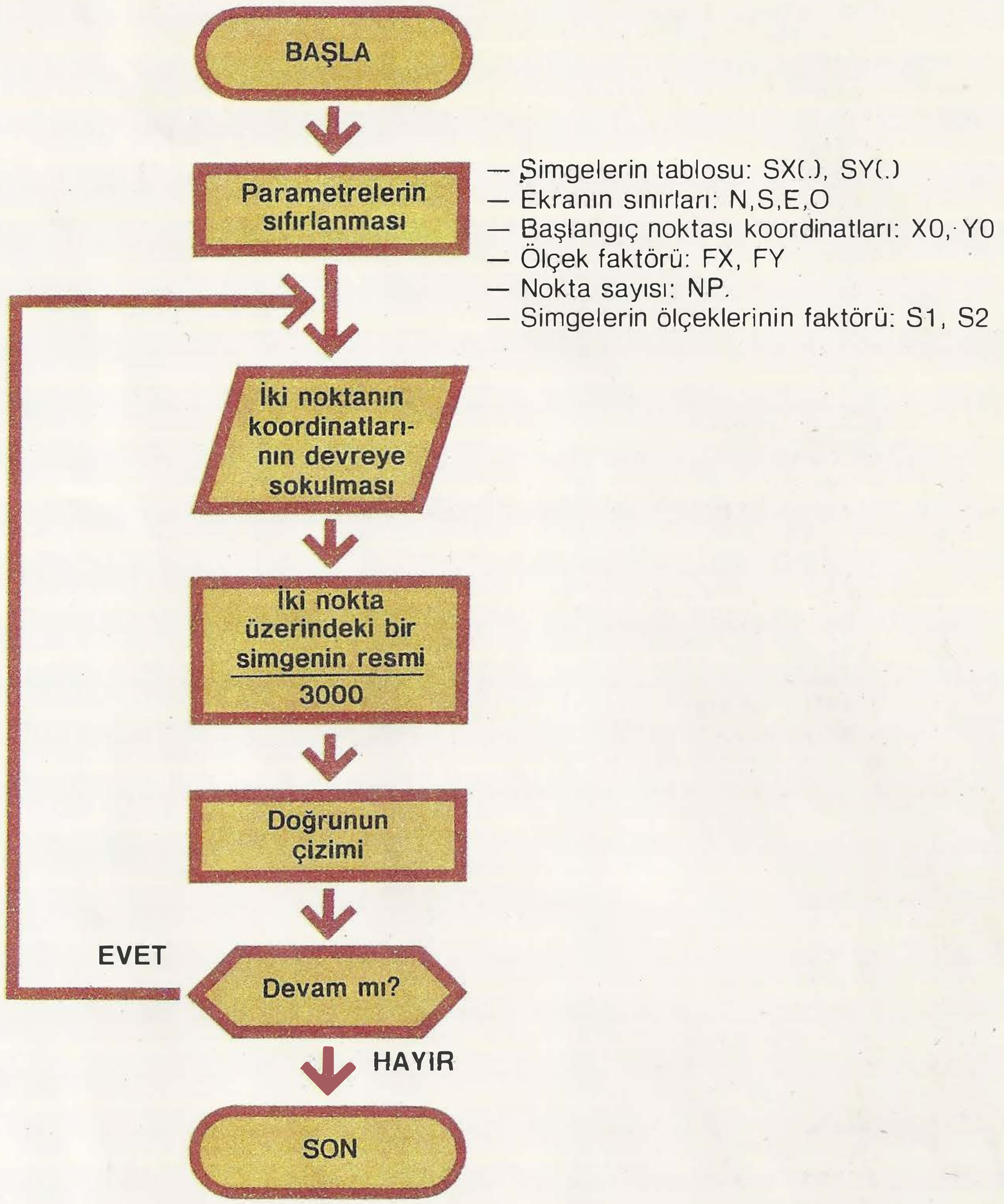
olacaktır.

Verilen formüller koordinatlar, örneğin klavye aracılığıyla girilmiş iki nokta arasındaki bir doğrunun görüntülenmesi için bir alt-programın yazılışına yarar. Program 80 ve 100. satırlar arasında birinci ve ikinci noktanın koordinatlarını ister ve iki simge saptar (burada simgeler iki dörtgendir alt-program 3000). Daha sonra program 120. ve 130. satırlar arasında A ve B değerlerini hesaplar ve doğruyu çizer (170. satırdan 240. satır arasındaki döngü).

Bu çözüm daha ilerde kullanacağımız bir yöntemi örneklemek amacıyla belirtilmiştir. Aynı sonucu elde etmek için eldeki iki noktayı bir-

Ücretsiz PDF Kitap Arsivi - https://rantey.org

İKİ NOKTADAN GEÇEN BİR DOĞRUNUN ÇİZİM VE DENKLEMİNİN BELİRLENMESİ



birine bağlayan bir doğru parçasının çizim komutu olarak HPLOT X1,Y1 TO X2,Y2 yeterli olurdu, fakat bunun için A ve B katsayılarının elimizde olması gerekir. A ve B katsayıları elimizde olmadığında doğru, X1,Y1, ve X2,Y2 noktaları tarafından sınırlanmış bir doğru parçası şeklinde gösterilir ve program, denklemin

Y = A * X + B şeklinde genel analitik ifadesinden haberdar olmaz.

Sayfa 1448 ve onu izleyen iki sayfadaki döküm, ekranın herhangi bir noktasında bir simge üretmek için bir yer değişim tablosunun nasıl kullanılacağını göstermektedir. Burada bir dörtgen olan simgenin kenarlarını belirleyen koordinatların değerleri 20. ve 30. satırlarda belleğe alınır. Simgeyi belirli bir noktada çizmek için bu değerleri sözkonusu değerlerinkine eklemek gerekir.

Bunun dışında 3000 alt-programı 50. satırda her bir doğru parçasının uzunluğunu değiştirmeye yarayan ve dörtgenler için iki adet gereken (S1 ve S2) katsayıları üretir.Simgenin ilk üç kenarı 3050-3090 döngüsü tarafından sonuncusu ise 3100. satırı tarafından görüntülenir.

Bu programı Siprel, Apple ve bunlara uyan makinelerden başka makinelere taşımak için,

Ücretsiz PDF Kitap Arsivi - https://rantey.org

İKİ NOKTADAN GEÇEN BİR DOĞRUNUN ÇİZİMİ

Apple versiyonu ve denklem

```
10      HOME
20      FOR J = 1 TO 4
      : READ SX(J), SY(J)
      : NEXT J
30      DATA -1,1,1,1,1,-1,-1,-1
40      N=5
      : S=153
      : E=265
      : O=15
50      XO=140
      : YO=80
      : FX=5
      : FY=5
      : NP=10
      : S1=2
      : S2=2
60      GOSUB 1000
70      DEF FN R(X) = A * X + B
80      VTAB 23
      : INPUT "BIRINCI NOKTANIN KOORDINATLARI: ";
        X1,Y1
90      XS = X1 * FX + XO
      : YS = YO - Y1 * FY
      : GOSUB 3000
100     HOME
110     VTAB 23
      : INPUT "IKINCI NOKTANIN KOORDINATLARI: "
        "; X2,Y2
120     A = (Y2-Y1)/(X2-X1)
130     B = ((Y1 * X2)- (Y2 * X1))/ (X2
            - X1)
140     XS = X2 * FX + XO
      : YS = YO - Y2 * FY
      : GOSUB 3000
150     P = (X2 - X1) / NP
160     X = X1
      : Y = FN R(X)
170     FOR XX = X1 + P TO X2 STEP P
180     YY = FN R(XX)
190     XA = XO + FX * X
      : YA = YO - FY * Y
200     XB = XO + FX * XX
      : YB = YO - FY * YY
210     IF XA > 265 OR XA < 15 OR XB > 26
          5 OR XB < 15 OR YA > 153 OR YA
            < 5 OR YB > 153 OR YB < 5
          THEN 230
220     HPLOT XA,YA TO XB,YB
230     X = XX
      : Y = YY
240     NEXT
250     HOME
260     VTAB 23
270     INPUT "DEVAM ? (E/H)";S$
280     IF S$ = "E" THEN RUN
290     END
997     REM ------------------------------------
998     REM  BICIMLENDIRME
999     REM ------------------------------------
1000    HGR
      : HCOLOR=3
1010    HPLOT O-15, N - 5 TO E + 14,N
          -5 TO E + 14, S + 3 TO O - 15, S
            + 3 TO O - 15,, N - 5
1020    HPLOT O, YO TO E, YO
      : HPLOT XO, S TO XO, N
1030    N2 = N
      : FOR N1 = 2 TO 0 STEP -1
```


Ücretsiz PDF Kitap Arsivi - https://rantey.org

```
      : HPLOT XO - N1, N2 TO XO + N1, N2
      : N2 = N2 - 1
      : NEXT N1
1040  N2 = E
      : FOR N1 = 2 TO 0 STEP -1
      : HPLOT N2, YO - N1 TO N2,YO + N1
      : N2 = N2 + 1
1050  HPLOT E + 6, YO - 2 TO E + 11, YO
        + 3
      : HPLOT E + 11, YO - 2 TO E + 11, YO
        + 3
1060  HPLOT XO - 10,N - 2 TO XO - 7,N
        + 1
      : HPLOT XO - 4,N - 2 TO XO - 10,N
        + 3
1070  IF FX = 1 THEN 1140
1080  FOR G = XO TO 0 STEP - FX
1090  HPLOT G, YO + 1 TO G,YO -1
1100  NEXT G
1140  IF FY = 1 THEN 1210
1150  FOR G = YO TO N STEP -FY
1160  HPLOT XO + 1, G TO XO - 1,G
1200  NEXT G
1210  RETURN
3000  REM ------------------------------------
3010  REM SIMGELERIN BICIMLENDIRILMESI
3020  REM ------------------------------------
3030  XA = SX(1) * S1 + XS
      : YA = SY(1) * S2 + YS
3040  XC = XA
      : YC = YA
3050  FOR I= 2 TO 4
3060  XB = SX(I) * S1 + XS
      : YA = SY(I) * S2 + YS
3070  HPLOT XA,YA TO XB,YB
3080  XA = XB
      : YA = YB
3090  NEXT I
3100  HPLOT XB,YB TO XC,YC
3110  RETURN
```

Olivetti M20 versiyonu

```
10    CLEAR
20    NP = 10
30    W2 = WINDOW(2,40)
40    DEF FN R(X) = A * X + B
50    CLS%2
60    LINE%2(0,0) - (511,255),3,B
70    CLS
80    SCALE%2, -50,50, - 50,50
90    GOSUB 270
100   INPUT "BIRINCI NOKTA";X1,Y1
110   XA = X1
      : YA = Y1
      : GOSUB 3000
120   INPUT "IKINCI NOKTA";X2,Y2
130   XA = X2
      : YA = Y2
      : GOSUB 3000
140   P = (X2 - X1) / NP
```


Ücretsiz PDF Kitap Arsivi - https://rantey.org

```
150     A = (Y2 - Y1) / (X2 - X1)
160     B = ((Y1 * X2) - (Y2 * X1)) / (X2
            - X1)

170     X = X1
    :   Y = FN R(X)
180     FOR XX = X1 + P TO X2 STEP P
190     YY = FN R(XX)
200     LINE%2(X,Y) - (XX,YY),2
210     X = XX
    :   Y = YY
220     NEXT
230     INPUT "DEVAM ? (E/H)";A$
240     IF A$= "E" THEN 10
250     CLEAR
    :   END
270     REM ***** BICIMLENDIRME **********
280     LINE%2(0, - 50) - (0, 50)
290     LINE%2(-50, 0) - (50, 0)
300     FOR Q = 0 TO 49 STEP 2
310     LINE%2(- .5, Q) - (.5, Q)
320     LINE%2(Q, -  .5) - (Q, .5)
330     NEXT Q
340     FOR Q = 0 TO -49 STEP -2
350     LINE%2(- .5, Q) - (.5, Q)
360     LINE%2(Q, - .5) - (Q, .5)
370     NEXT Q
380     RETURN
3000    REM  ******** SIMGELERI BICIMLENDIRMEK ******
3010    LINE%2(XA - 2,YA - 2) - (XA + 2,Y
            A + 2),2,B
3020    RETURN
```

VTAB, HPLOT, HCOLOR, HGR komutlarını ve 210. satırdaki kontrolleri kullanılmakta olan ekranın sayısal değerleri gözönünde bulundurularak değiştirmek gerekmektedir.

Örnek olsun diye Olivetti M20'ye uygun olan bir döküm verdik. Bu program daha sonra ayrıntılı olarak açıklayacağımız bir takım özel komutlara başvurmaktadır.

En Küçük Karelerin Doğrusal Regrasyonu

Bir olayın çözümlenmesi olayı ngelişimi sırasında çeşitli değişkenlerin aldığı farklı değerlerin gözlenmesinden ibarettir.

En basit durum biri bağımsız olan iki değişken tarafından belirlenir. Bu bağımsız değişkenin aldığı değerlerin kümesi gözlenen olayı grafik şekilde gösterir.

X ve Y değişkenleri için aşağıdaki değerleri vermiş olan 5 gözleme dayalı bir örnekten yola çıkalım.

Gözlemlenemeyen bir X değişkenine karşılık gelen bir Y değerinin hemen çıkarsanamayacağı açıktır. Bu bilgiler daha önce sunulmuş

Gözlem	X	Y
a	2.5	3
b	4	3.5
c	5	5
d	7	4.5
e	9	6

olan bir yöntem uyarınca elde edilir. Bu yöntem gözlenen her bir değerler çiftinin doğrusal olarak genelleştirilmesinden oluşmaktadır.

Böylelikle elde edilen sonuç a, b, c, d, e noktaları ile birleşen bir doğru parçaları dizisi oluşturur. Sonuç kaba taslak çıkarsanmış değerlerle elde edilen ve gerçeklikten çok uzak bir grafik olabilir (s.1451'e bakınız). Bu yöntemde olayı temsil eden denklemler bilinen noktaları ikişer ikişer birleştiren dört doğrunun denklemleridir.

Her bir denklem, yani olayın analitik bir biçimde betimlenişi, bir önceki paragrafta geliştirilmiş olan yöntemden (eldeki iki noktadan geçen doğru) kaynaklanır. ''En küçük karelerin doğrusal regrasyonu'' olarak adlandırılan ikinci bir yöntem vardır ki bu bazı koşullar altında daha güvenilir sonuçlar vermektedir.

Ücretsiz PDF Kitap Arsivi - https://rantey.org

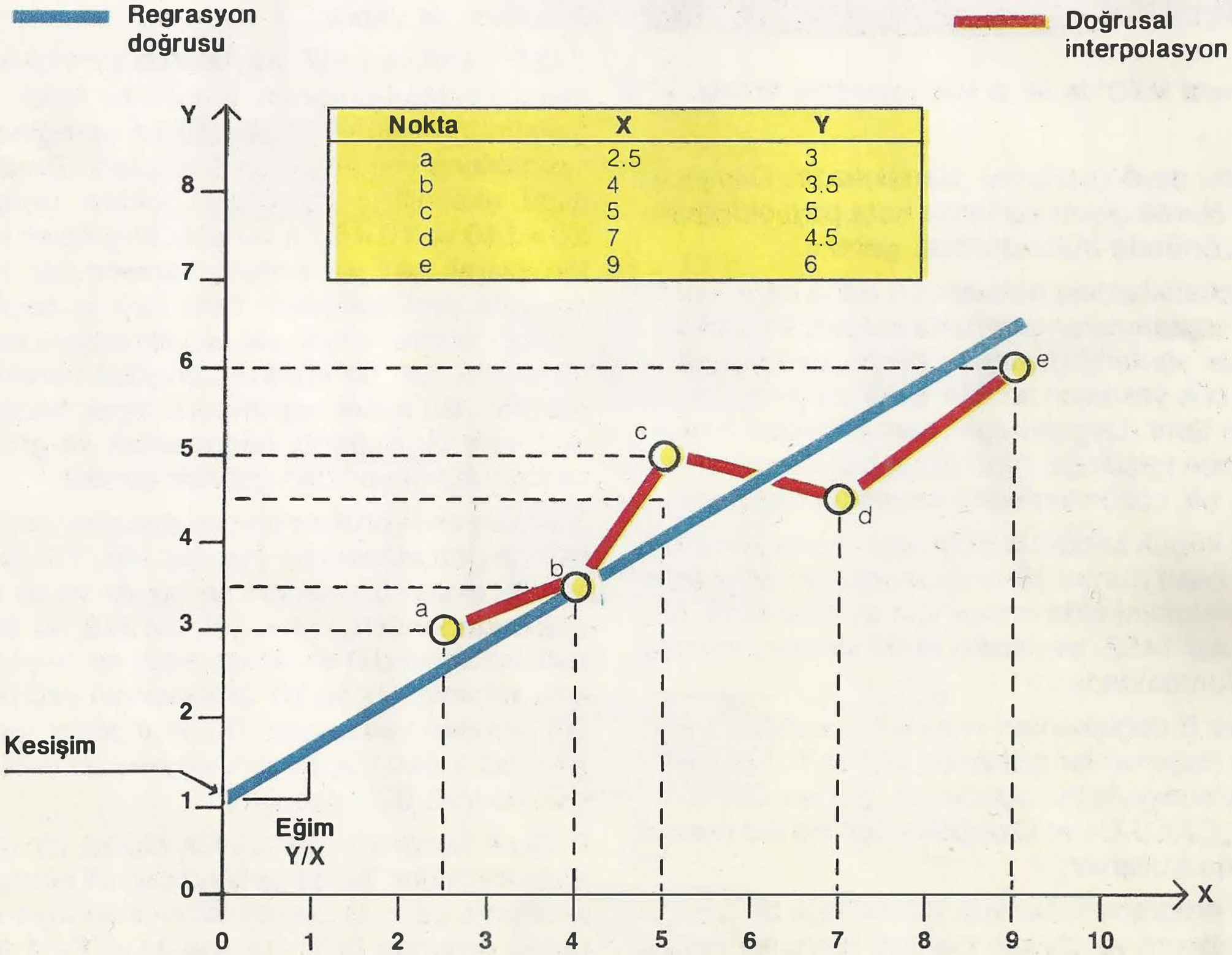
Şu an için olayın bir doğru ile simgeleneceğini varsayalım. Değişkenler üzerinde yapılan bir dizi gözlem sayesinde, hepsi doğru üzerinde yer almayan bazı değerler elde ederiz. Olayı temsil eden denklem bir doğru tarafından gösteriliyor olsa da, ölçülen a,b,c,d,e noktalarından (kırmızı çizgi) hiçbiri doğruyu vermez. Bu ''dağınıklık'' gözlenen olayın doğasına ilişkin bir çok etkene bağlıdır.

Fizik bir olayda hatalar aletlere veya raslantılara bağlı olabilir. Bir donanım ne kadar gelişmiş olursa olsun ancak bir birimin yaklaşık değerini verir.

Mekanik titreşimler sıcaklık veya rutubet değişiklikleri gibi dış faktörleri diğer hata nedenleri arasında sayabiliriz.

Bu hatalar birbirine eklenir ve birbirlerini ancak raslansal olarak ortadan kaldırır. Ayrıca ölçüm sırasında böyle bir nihai doğrulanma öngörülemez: o anda değerler kesin olsa da

5 GÖZLEMDEN OLUŞAN BİR OLAYIN EĞRİSİ



Nokta	X	Y
a	2.5	3
b	4	3.5
c	5	5
d	7	4.5
e	9	6

Ücretsiz PDF Kitap Arsivi - https://rantey.org



Olivetti MZO'de bir grafik uygulama örneği.

bunu da doğrulamak olanaksızdır. Demek ki ne olursa olsun verilerde hata payı olduğunu gözönünde bulundurmak gerekir.

İstatistiklerdeki hatalar çok daha karmaşıktır ve saptanmaları çok daha zordur. En ufak kareler yöntemi gözlenen olayın fonksiyonuna en çok yaklaşan analitik şekli araştırmaya imkan tanır. Uygulanacak olan formüller fonksiyonun başlangıç hipotezine bağlı olan karmaşık bir çözümlemeden kaynaklanmaktadır.

En küçük karelerin doğrusal regrasyonu için en basit durum bir doğrununkidir: Doğrunun denklemini elde etmek için uygulanacak formüller 1453. sayfadaki 8000 alt-programında bulunmaktadır.

A ve B değişkenleri denklemin katsayılarıdır. Bu değişkenler gözlenen X(.) ve Y(.) değerleri aracılığıyla hesaplanır. N, gözlem sayısıdır. C1,C2,C3,C4 ve D değişkenleri ise ara hesaplarda kullanılır.

Bu programı kullanmak için belleğe X(.) ve Y(.) ölçülerini ve gözlem sayısını içeren N değişkenini yerleştirmek gereklidir. Çıkışta olayı temsil ettiği varsayılan doğrunun A ve B katsayıları elde edilir (bakınız sayfa 1454'teki akış şeması).

İlk aşama kartezyen eksenlerin çizilmesinden oluşur. Sonra program N nokta sayısı ile gözlenen noktaların koordinatlarının değerlerini ister (sayaçlı döngü). Her bir koordinat çifti için 5900 alt-programı gözlenen değere karşılık olan noktanın üzerinde bir dörtgen çizer. Sunuş döngüsünden sonra A ve B katsayılarını hesaplayan 8000 alt-programı (yanda) çağrılır. Sonuç bir doğru olarak somutlaşır.

Aslında programın gelişimi özellikle ölçek faktörleri düzeyinde çok daha karmaşıktır.

Sunuş aşamasında ölçek benimsendikten sonra ekranın sınırlarından ''taşan'' X ve Y değerleri olabilir. Aynı şekilde sırasıyla X ekseni üzerinde en küçük ve en büyük değerler olan X=XN ve X=XM'den yola çıkarak çizilen doğrunun Y=B*X+A denklemini hesapladıktan sonra Y'ye karşılık olan değerlerin ekran sınırları dışına taştığı görülebilir.

X ve Y değerlerinin sınırlarıyla ilgili olan birinci kontrol her girişten sonra gerçekleşir, oysa doğru üzerindeki kontrol, dökümü sunmuş olduğumuz akış şemasına uymayan 8000 alt-programı ile yapılır.

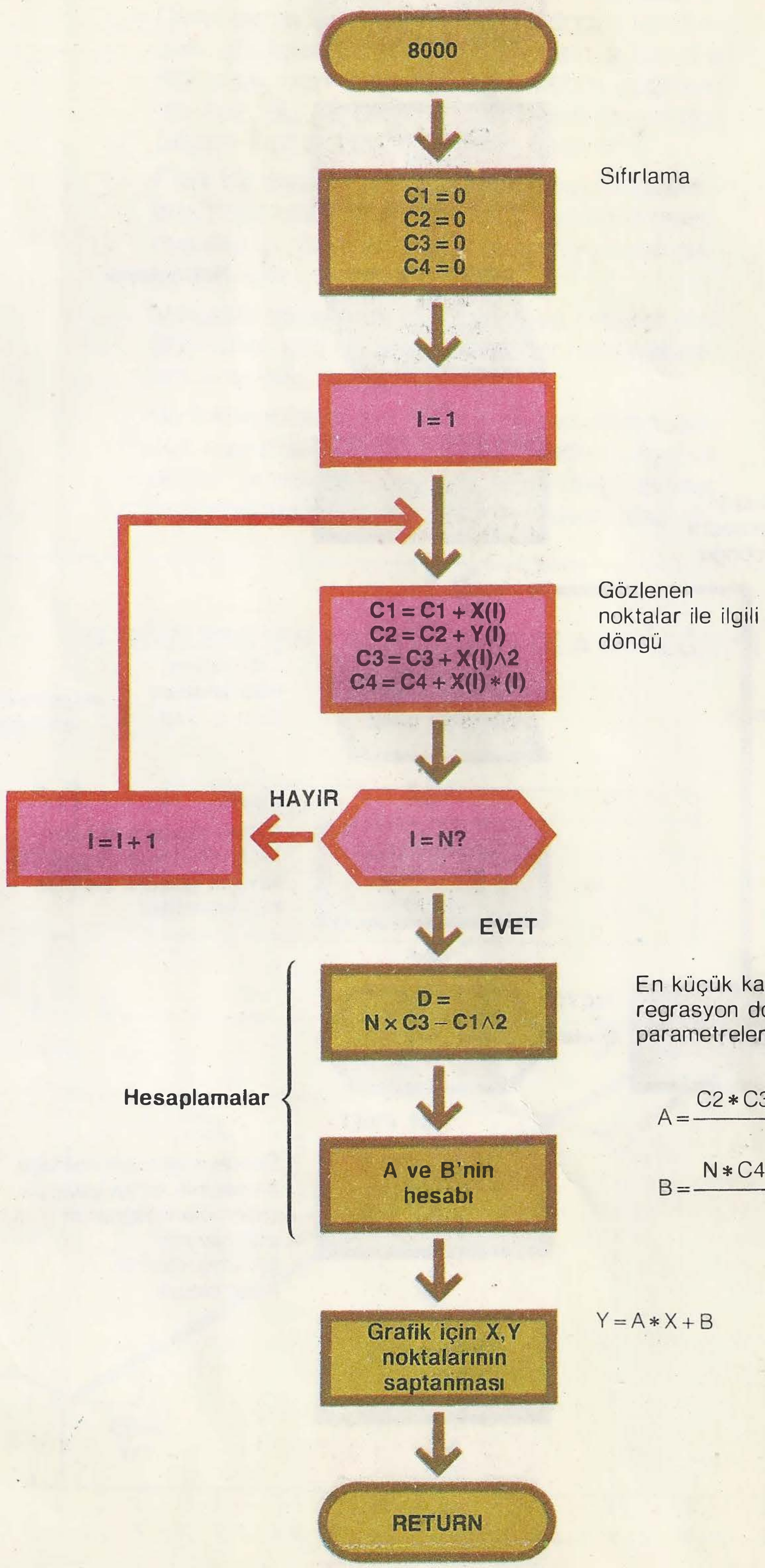
1455, 1456 ve 1457. sayfalarda ayrıntılı akış şeması verilen program, anında bir ölçek değişimini gözönünde bulundurma gereğinden kaynaklanan bir karmaşıklık gösterir. Başlangıçta eksenlerin başlangıç noktası **(origin)** X0=140 ve Y0=80'e yerleştirilmiş ölçek faktörü olarak ise F=0.7 olarak saptanmıştır. Her sunuşta yeni noktaların hata payı ile seçilen ölçeğin sınırları içinde yer alıp almadığını kontrol etmek için bir kontrol döngüsü harekete geçirilir. Bu işlem yapılmazsa ölçek faktörünün yeni bir değerini hesaplamak ve grafiği baştan aşağı yeniden çizmek gerekir.

Sunulan akıl yürütme bir dizi denetim ve çentiklerle gerçekleştirilir. Her bir X(I), Y(I) girişi için iki eksen üzerindeki en büyük ve en küçük değerler (XN,XM ve YN,YM 270. ve 360. satırlar arasında) ve aralarından en büyüğünün seçildiği DX ve DY aralıklarının uzunlukları yeniden hesaplanır. Bu yeni değer temel alınarak 1'den küçük olması gereken yeni ölçek faktörü (F) hesaplanır.

F ölçek faktörü 1'den büyük olursa bir hata sözkonusudur; bu da verilerin geçerli olmadığı anlamına gelir. O zaman işlem kesintiye uğrar ve program RUN komutu ile ve F<1 testinden sonra yeni baştan başlar. Ölçek faktörü belirlendiğine göre başlangıç noktasının

Ücretsiz PDF Kitap Arsivi - https://rantey.org

REGRASYON DOĞRUSUNUN A VE B ÇARPANLARININ HESABI



Sıfırlama

Gözlenen noktalar ile ilgili döngü

En küçük kareler regrasyon doğrusunun parametreleri

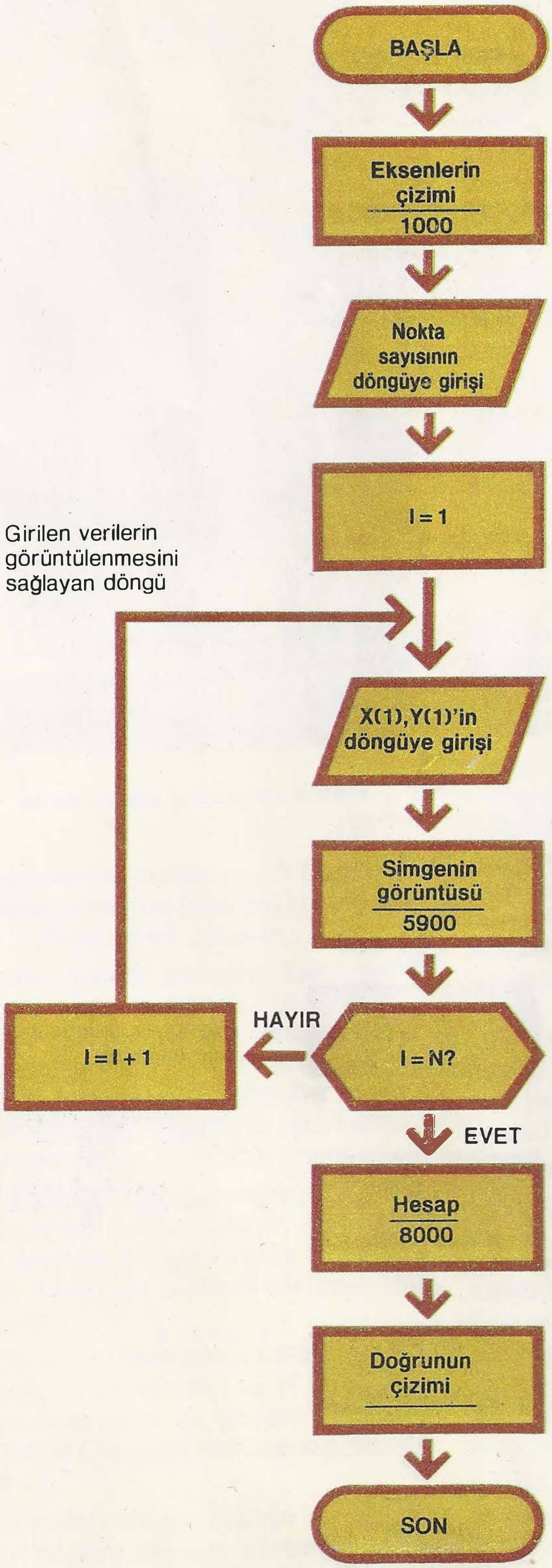
$$A = \frac{C2 * C3 - C1 * C4}{D}$$

$$B = \frac{N * C4 - C1 * C2}{D}$$

$$Y = A * X + B$$

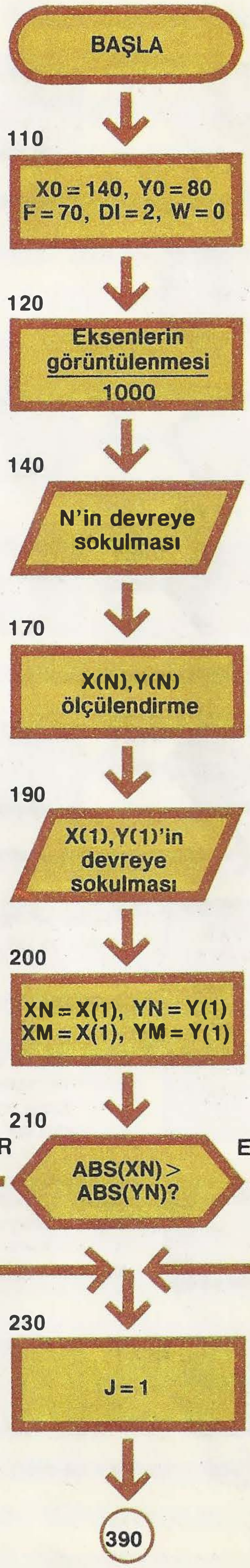
Ücretsiz PDF Kitap Arsivi - https://rantey.org

EN KÜÇÜK KARELERİN DOĞRUSAL REGRASYONU



Ücretsiz PDF Kitap Arsivi - https://rantey.org

REGRASYON DOĞRUSU



Kullanılan değişkenler:
X0,Y0 = Başlangıç noktasının koordinatları
F = Ölçek faktörü
DI = Alt-bölümlerin sayısı
XN,XM = X'in en küçük ve en büyük değerleri
YN,YM = Y'nin en küçük ve en büyük değerleri

Sıfırlama.
Başlangıç noktasının koordinatları, ölçek faktörü ve alt-bölümlerin sayısı kabul ettirilir.
W0, ölçek değişimleri kaçınılmaz hale geldiği zaman 1000 alt-programı tarafından verilen bir çentiktir.

N en az ikiye eşit olmak üzere sunulacak nokta sayısıdır

X ve Y dizeye ait koordinatları içerecek tablolardır.

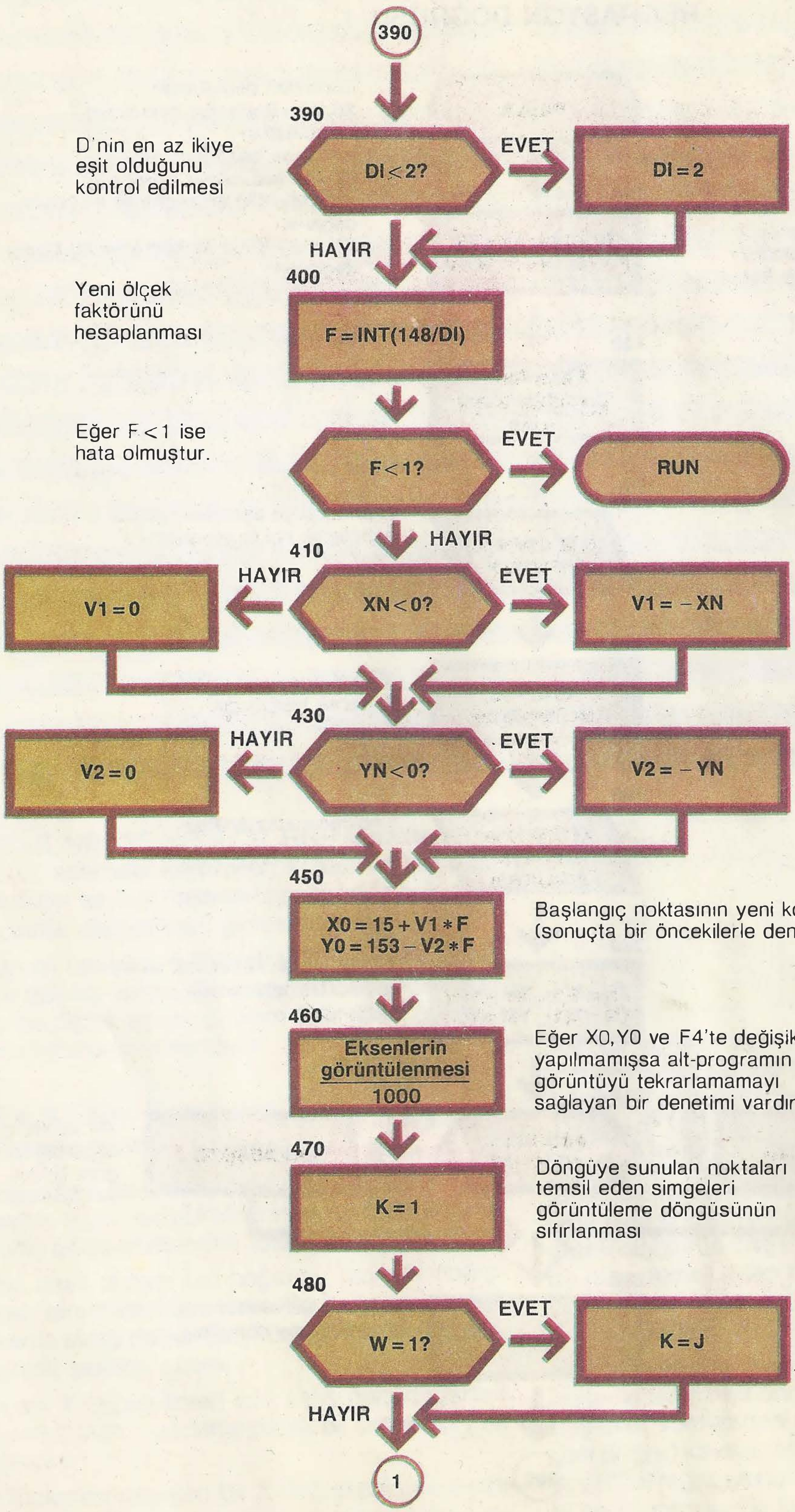
1.noktanın koordinatları

X ve Y'nin en küçük (N) ve en büyük (M) değerlerinin sıfırlanması

XN ve YN'in değerine göre DI'nın yenilenmesi

Girişlerin ve verilerin görüntülenme döngüsü

Ücretsiz PDF Kitap Arsivi - https://rantey.org



D'nin en az ikiye eşit olduğunu kontrol edilmesi

Yeni ölçek faktörünü hesaplanması

Eğer F<1 ise hata olmuştur.

Hata

Başlangıç noktasının koordinatlarının hesaplanmasına yarayan V1 ve V2 değerlerini vermek

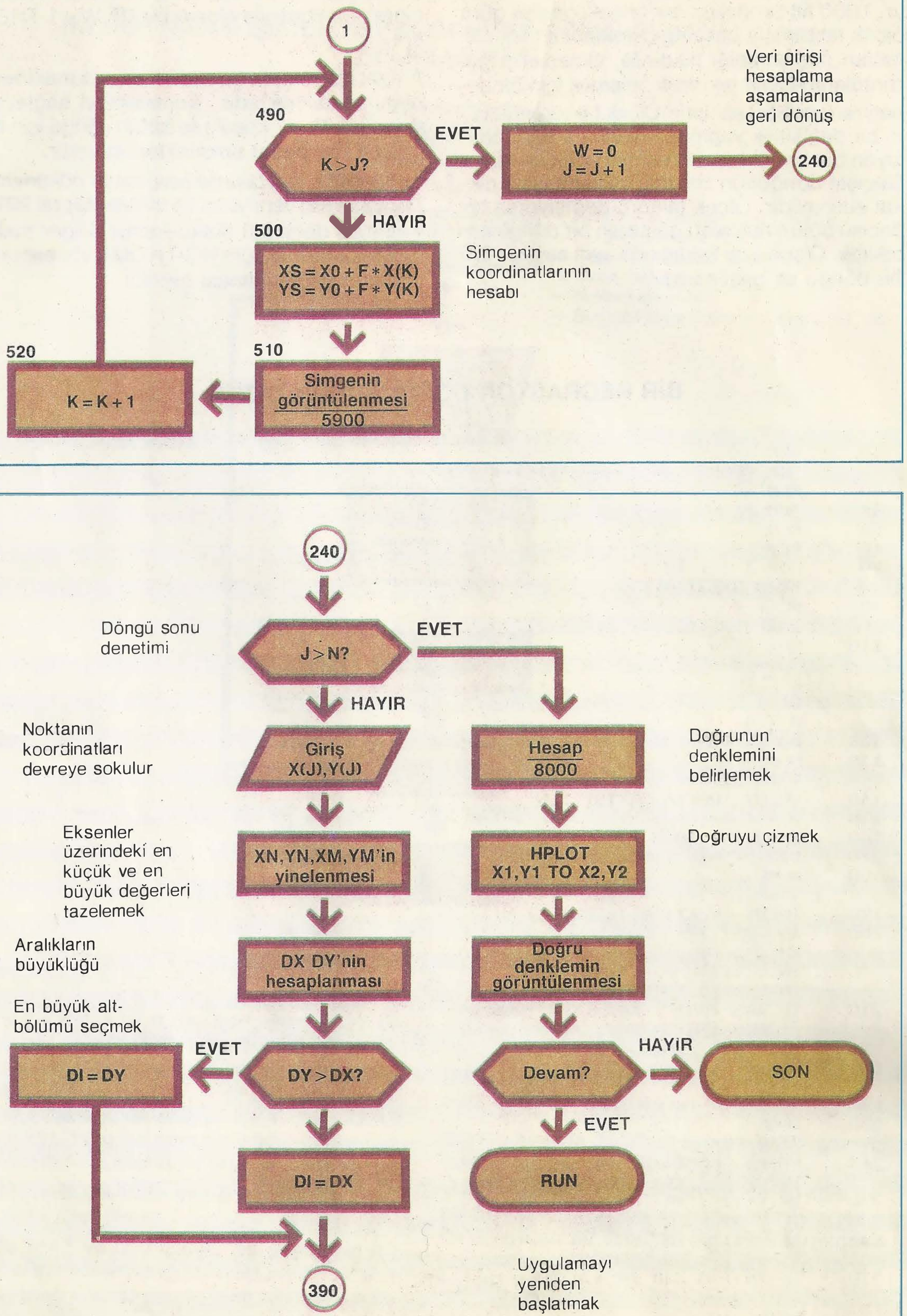
Başlangıç noktasının yeni koordinatları (sonuçta bir öncekilerle denk olacaklardır)

Eğer X0, Y0 ve F4'te değişiklik yapılmamışsa alt-programın görüntüyü tekrarlamamayı sağlayan bir denetimi vardır.

Döngüye sunulan noktaları temsil eden simgeleri görüntüleme döngüsünün sıfırlanması

W=1 olduğunda döngüden zorunlu çıkış

Ücretsiz PDF Kitap Arsivi - https://rantey.org



Ücretsiz PDF Kitap Arsivi - https://rantey.org

(XO,YO) yeni koordinatlarının hesabına geçilir. 1000 alt-programı, bir önceki geçişe göre ölçek faktörü ve başlangıç noktasının koordinatları değişmediği takdirde, eksenlerin görüntülenmesinin tekrarını önlemek için bir denetime (çentiğe) sahiptir. Ölçek faktöründe yeni bir değişiklik yapmak koşuluyla son fonksiyon bütün noktaların resmini yeniden çizer. Geçişte döngünün zorunlu çıkışına iyice dikkat edilmelidir. Ölçek faktörü değişiyorsa bir önceki bütün noktaları gösteren bir döngü gereklidir. Ölçek aynı kaldığında aynı sayaç 1 ise bu döngü es geçilmelidir.

Fonksiyon en üst uca döngü sayacının koşul olarak konmasıyla elde edilir (IF W = 1 THEN K = J).

Ancak bu yöntem çok sınırlı sayıda makineye uygulanabilmektedir. Genellemeyi sağlayan değişiklik W = 1 koşulu ile bütün döngü için bir atlama **(by pass)** sokmaktan ibarettir.

1458-1461. sayfalarda programın dökümünü Personal Kid versiyonu ile birlikte (Siprel 2010 Apple ve denkleri) bulacaksınız. Diğer makinelere aktarmak için HGR HOME vb. bazı komutların değiştirilmesi gerekir.

BİR REGRASYON DOĞRUSUNUN ÇİZİMİ

```
10      REM ---------------------------------------
20      REM REGRASYON DOGRULARI ********
30      REM ---------------------------------------
40      REM
50      REM
60      REM
70      FOR J = 1 TO 4
80      READ SX(J),SY(J)
90      NEXT J
100     DATA -1,1,1,1,1,-1,-1,-1
110     XO = 140
      : YO = 80
      : F = 70
      : DI = 2
      : W = 0
120     GOSUB 1000
130     HOME
      : VTAB 22
140     INPUT "NOKTA SAYISI?";N
150     N = INT (N)
160     IF N < 2 THEN 130
170     DIM  X(N), Y(N)
180     HOME
      : VTAB 22
190     INPUT "NOKTA N.1: ";X(1),Y(1)
200     XN = X(1)
      : XM = X(1)
      : YN = Y(1)
      : YM = Y(1)
210     IF ABS (XN) > ABS (YN) THEN DI
        = ABS (XN)
220     DI = ABS (YN)
230     J = 1
      : GOTO 390
240     IF J > N THEN 540
250     HOME
      : VTAB 22
260     PRINT "NOKTA N. "J": ";
      : INPUT ""; X(J),Y(J)
270     IF X(J) > XM THEN XM = X(J)
280     IF X(J) < XN THEN XN = X(J)
290     IF Y(J) > YM THEN YM = Y(J)
300     IF Y(J) < YN THEN YN = Y(J)
310     IF XM > 0 AND XN < 0 THEN DX = XM
          - XN
      : GOTO 340
```


Ücretsiz PDF Kitap Arsivi - https://rantey.org

```
320     IF XN > = 0 AND XM > 0 THEN DX
            = XM
        : GOTO 340
330     DX = - XN
340     IF YN < 0 AND YM > 0 THEN DY = YM
            - YN
        : GOTO 370
350     IF YN > = 0 AND  YM > 0 THEN DY
            = YM
        : GOTO 370
360     DY = - YN
370     IF DY > DX THEN DI = DY
        : GOTO 390
380     DI = DX
390     IF DI < 2 THEN DI = 2
400     F = INT (148 / DI)
        : IF F < 1 THEN RUN
410     IF XN < 0 THEN V1 = - XN
        : GOTO 430
420     V1 = 0
430     IF YN < 0 THEN V2 = - YN
        : GOTO 450
440     V2 = 0
450     X0 = V1 * F + 15
        : Y0 = 153 - V2 * F
460     GOSUB 1000
470     K = 1
480     IF W = 1 THEN K = J
490     IF K > J THEN W = 0
        : GOTO 530
500     XS = X(K) * F + X0
        : YS = Y0 - Y(K) * F
510     GOSUB 5900
        : REM SIMGELER
520     K = K + 1
        : GOTO 490
530     J = J + 1
        : GOTO 240
540     GOSUB 8000
        : REM HESAP
550     HPLOT X1, Y1 TO X2,Y2
560     S$ = "-"
570     IF A > 0 THEN S$ = "+"
580     HOME
        : VTAB 22
590     PRINT "EN YAKIN DOGRU E DIR:
        "
600     IF A = 0 AND B = 0 OR D = 0
            THEN INVERSE
        : PRINT "X =  ";XN
        : GOTO 630
610     IF B = 0 THEN INVERSE
        : PRINT "Y = " A
        : GOTO 630
620     INVERSE
        : PRINT "Y = "B" * X "S$" " ABS (A)

630     NORMAL
640     VTAB 24
650     PRINT "DEVAM MI? (E/H) ";
        : GET A$
660     IF A$ = "E" THEN RUN
670     TEXT
        : HOME
```


Ücretsiz PDF Kitap Arsivi - https://rantey.org

```
      : END
997   REM ------------------------------
998   REM EKSENLERI BICIMLEMEK
999   REM ------------------------------
1000  IF XO = XP AND YO = YP AND F = FF
        THEN W = 1
      : RETURN
1010  HGR
      : HCOLOR= 3
1020  HPLOT 0,0 TO 279,0 TO 279,159
        TO 0,159 TO 0,0
1030  HPLOT 15,YO TO 265,YO
      : HPLOT XO,153 TO XO,5
1040  N2 = 5
      : FOR N1 = 2 TO 0 STEP -1
      : HPLOT XO - N1,N2 TO XO + N1,N2
      : N2 = N2 - 1
      : NEXT N1
1050  N2 = 265
      : FOR N1 = 2 TO 0 STEP -1
      : HPLOT N2,YO - N1 TO N2,YO + N1
      : N2 = N2 + 1
      : NEXT N1
1060  HPLOT 271,YO - 2 TO 276,YO + 3
      : HPLOT 276,YO - 2 TO 271,YO + 3
1070  HPLOT XO - 10,3 TOXO - 7,6
      : HPLOT XO - 4,3 TO XO - 10,8
1080  IF F = 1 THEN 1220
1090  FOR G = XO TO 15 STEP - F
1100  HPLOT G,YO + 1 TO G,YO - 1
1110  NEXT G
1120  FOR G = XO TO 265 STEP F
1130  HPLOT G,YO + 1 TO G,YO - 1
1140  NEXT G
1150  FOR G = YO TO 5 STEP - F
1160  HPLOT XO + 1,G TO XO - 1,G
1170  NEXT G
1180  FOR G = YO TO 153 STEP F
1190  HPLOT XO + 1,G TO XO - 1,G
1200  NEXT G
1210  XP = XO
      : YP = YO
      : FF = F
1220  RETURN
5899  REM ------------------------------
5900  REM SIMGELER
5901  REM ------------------------------
5910  S = F / 4
      : IF S > 5 THEN S = 5
5920  IF S < 2 THEN S = 2
5930  XA = SX(1) * S + XS
      : YA = SY(1) * S + YS
5940  XC = XA
      : YC = YA
5950  FOR T = 2 TO 4
5960  XB = SX(T) * S + XS
      : YB = SY(T) * S + YS
5970  HPLOT XA,YA TO XB,YB
5980  XA = XB
      : YA = YB
5990  NEXT T
6000  HPLOT XB,YB TO XC,YC
6010  RETURN
```


Ücretsiz PDF Kitap Arsivi - https://rantey.org

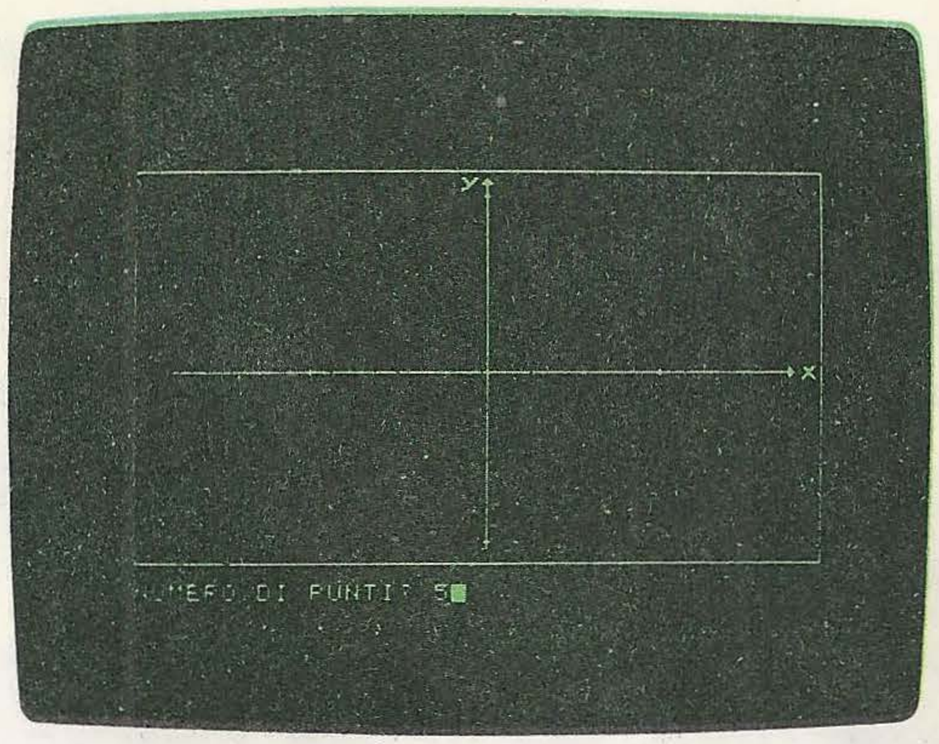
```
7999    REM ---------
8000    REM HESAP
8010    REM ---------
8020    C1 = 0
      : C2 = 0
      : C3 = 0
      : C4 = 0
8030    FOR I = 1 TO N
8040    C1 = C1 + X(I)
8050    C2 = C2 + Y(I)
8060    C3 = C3 + X(I) ^ 2
8070    C4 = C4 + X(I) * Y(I)
8080    NEXT I
8090    D = N * C3 - C1 ^ 2
8100    IF D = 0 THEN Y = YO - YN * F
      : Y2 = YO - YM * F
      : X1 = XO + XN * F
      : X2 = X1
      : RETURN
8110    A = (C2 * C3 - C1 * C4) / D
8120    B = ( N * C4 - C1 * C2) / D
8130    IF B = 0 THEN Y1 = YO - A * F
      : Y2 = Y1
      : X1 = XO + XN * F
      : X2 = XO + XM * F
      : RETURN
8140    IF A = 0 AND B = 0 THEN Y1 = YO
          - YN * F
      : Y2 = YO - YN * F
      : X1 = XO + XN * F
      : X2 = X1
      : RETURN
8150    WS = 0
8160    X = (YM - A) / B
      : Y = YM
8170    IF X > = XN AND X < = XM THEN
          GOSUB 8270
8180    X = (YN -A) / B
      : Y = YN
8190    IF X > = XN AND X < = YM THEN
          GOSUB 8270
8200    X = XM
      : Y = B * XM + A
8210    IF Y > = YN AND Y < = YM THEN
          GOSUB 8270
8220    X = XN
      : Y = B * XN + A
8230    IF Y > = YN AND Y < = YM THEN
          GOSUB 8270
8240    X1 = XO + X1 * F
      : X2 = XO + X2 * F
8250    Y1 = YO - Y1 * F
      : Y2 = YO - Y2 * F
8260    RETURN
8270    IF WS = 1 THEN 8290
8280    X1 = X
      : Y1 = Y
      : WS = 1
      : RETURN
8290    X2 = X
      : Y2 = Y
      : RETURN
```


Ücretsiz PDF Kitap Arsivi - https://rantey.org

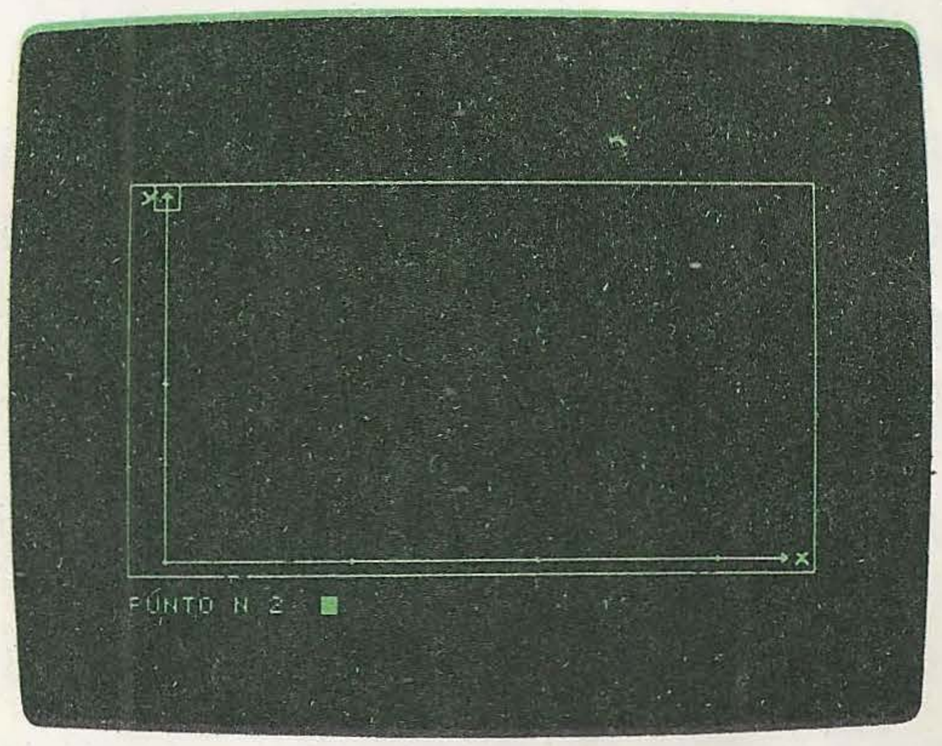
EN KÜÇÜK KARELERİN DOĞRUSAL REGRASYONU

Sistem iki kartezyen eksen sunmakta ve nokta sayısını sormaktadır.
yazılımın duyarlılığını ölçmek için Y = 3*X + 2 doğrusuna ait 5 nokta sunacağız.

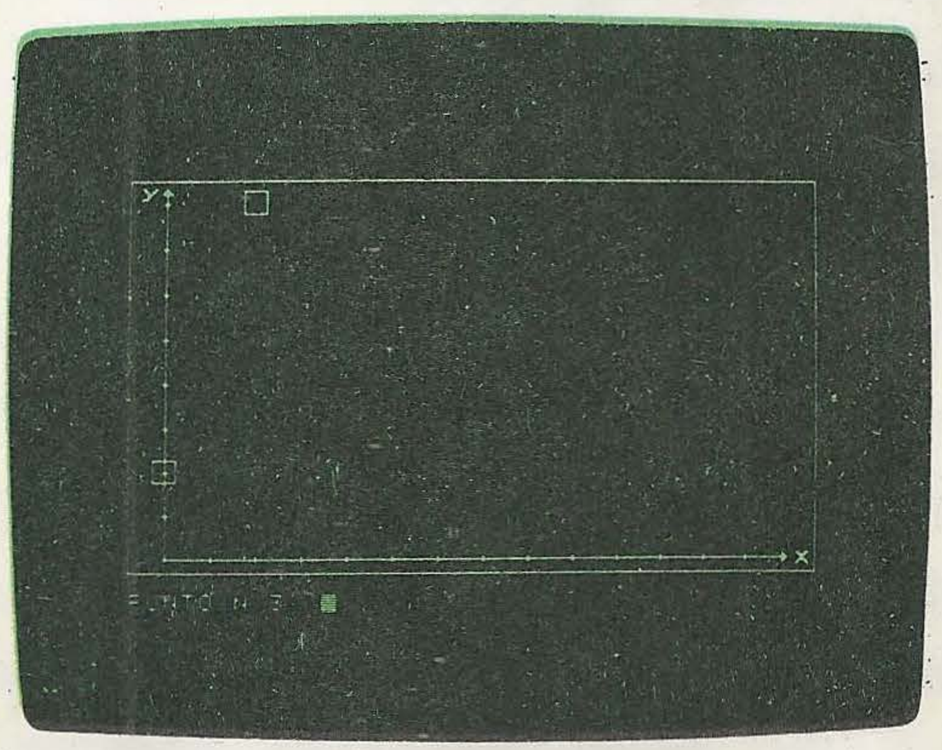
	X	Y
Nokta 1	0	2
Nokta 2	2	8
Nokta 3	3	11
Nokta 4	5	17
Nokta 5	10	32



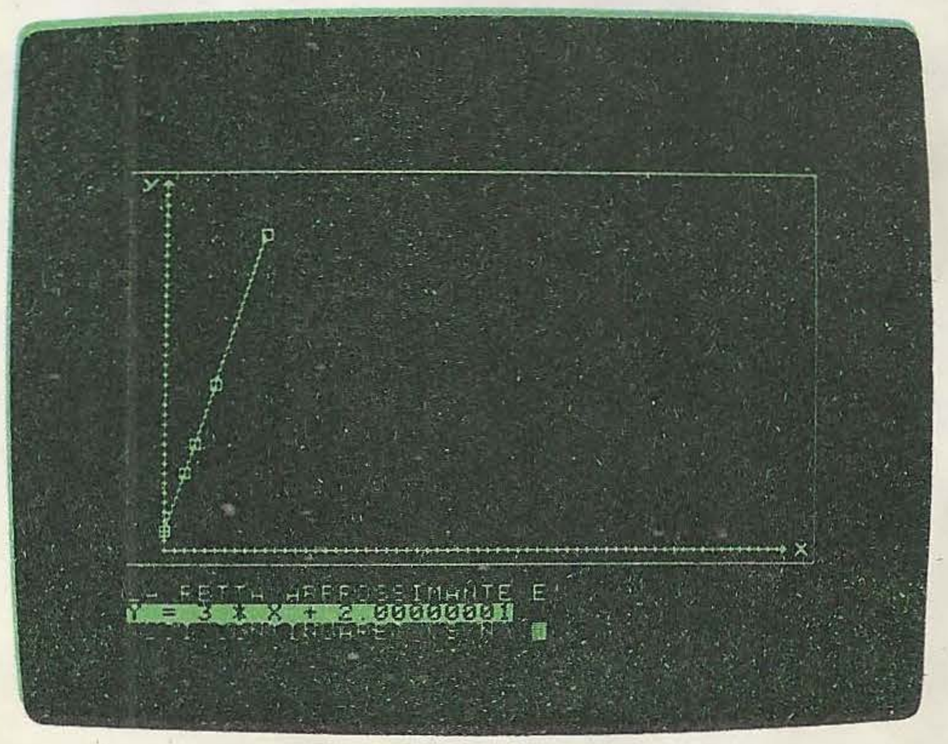
Sistemde 0.2 koordinatlı birinci nokta daha önce girilmişti.program kartezyen eksenlerin konumunu girilen değerlerin gösterimi için ekranın bütün yüzeyinden yararlanacak şekilde uyarlar.
Başlangıç noktası bir önceki ekrana göre şimdilik yalnızca 0,2 arasında (girilen değerlerde negatif ölçüler olmadığından) aşağı sola doğru kaydırılmıştır. Demek ki ekran tamamen referans sisteminin pozitif çerçevesine adanmıştır, dolayısıyla da Y'nin en büyük değeri girilen en büyük değere eşit olur.



2,8 koordinatlı 2. noktanın girilmesinden sonra program Y ekseninin ölçeğini Y = 8'e uyarlayacak şekilde küçültür. Koordinatları aynı kaldığı haldebir önceki nokta aşağı doğru yer değiştirmiş gibi olur. Gerçekten de bir önceki ekranda Y ekseni sadece yatay küçük çizgilerden oluşan iki bölümden oluşuyordu ve simge konum 2'de yani Y = 2 ordinatında yer alıyordu. Yeni durumda program ister istemez girişi yapılan Y = 8 yeni en büyük değere göre ayarlayarak ölçekte değişiklik yaptı.



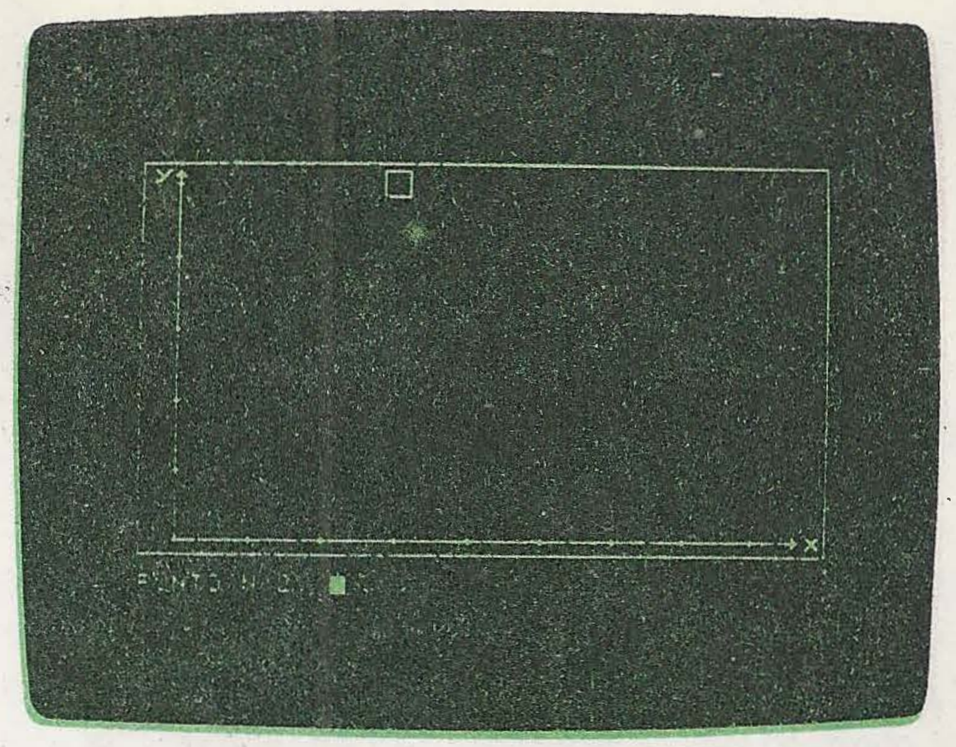
Sayıları başta açıklanmış olan verilerin girilmesinden sonra program noktaları ve regrasyonu içeren son grafiği sunar. Program ayrıca doğrunu denklemini de görüntüler. Hesaplanan katsayılar noktaları saptamak için kullanılan değerlere göre herhangi bir farklılık göstermezler. Virgülden sonraki bir dizi sıfırları izleyen 1 değeri katsayıların gösteriminde tamlama yapılmadığında ortaya çıkar. Pratik uygulamalarda sonuç 2. veya 3. ondalığa tamamlanır.



Ücretsiz PDF Kitap Arsivi - https://rantey.org

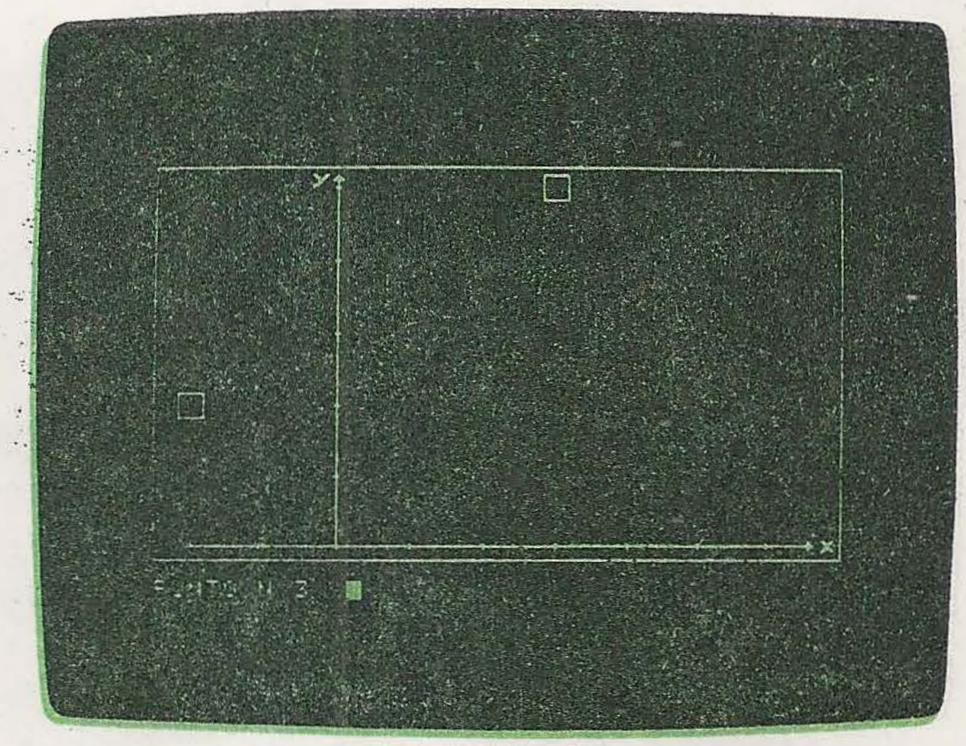
Birinci örneğin aksine ikinci örnek tam anlamıyla bir regrasyon hesabına ilişkindir. Bu hesap aşağıdaki şu değerlerle yapılır:

	X	Y
Nokta 1	3	5
Nokta 2	− 2	
Nokta 3	10	6
Nokta 4	− 5	3
Nokta 5	1	− 2

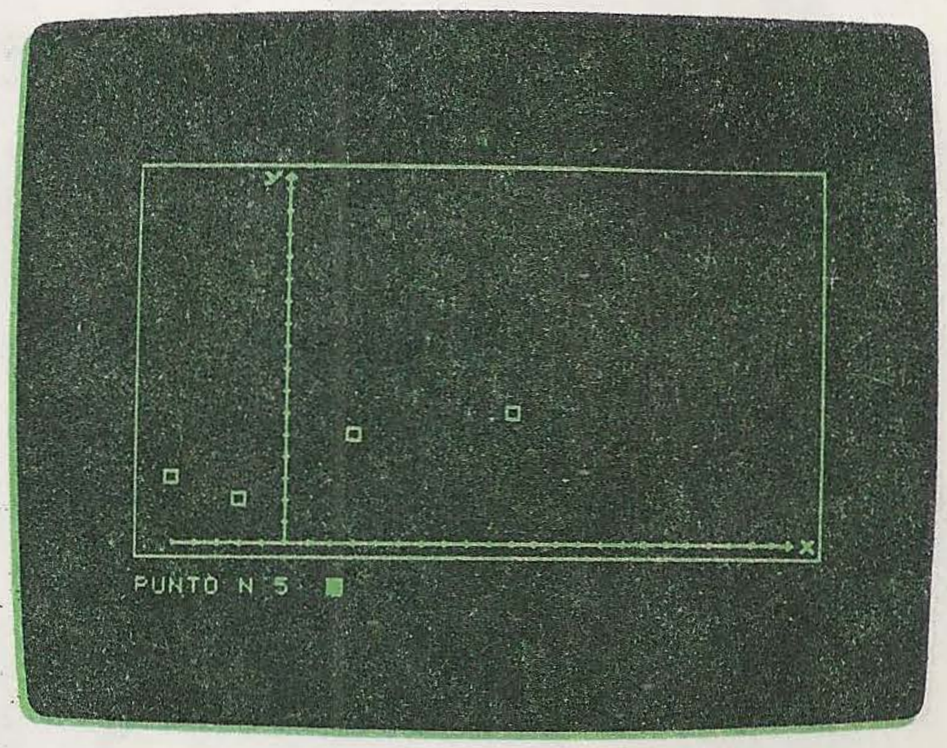


Birinci noktanın kordinatları girilmiştir ve program 2. veriyi ister.

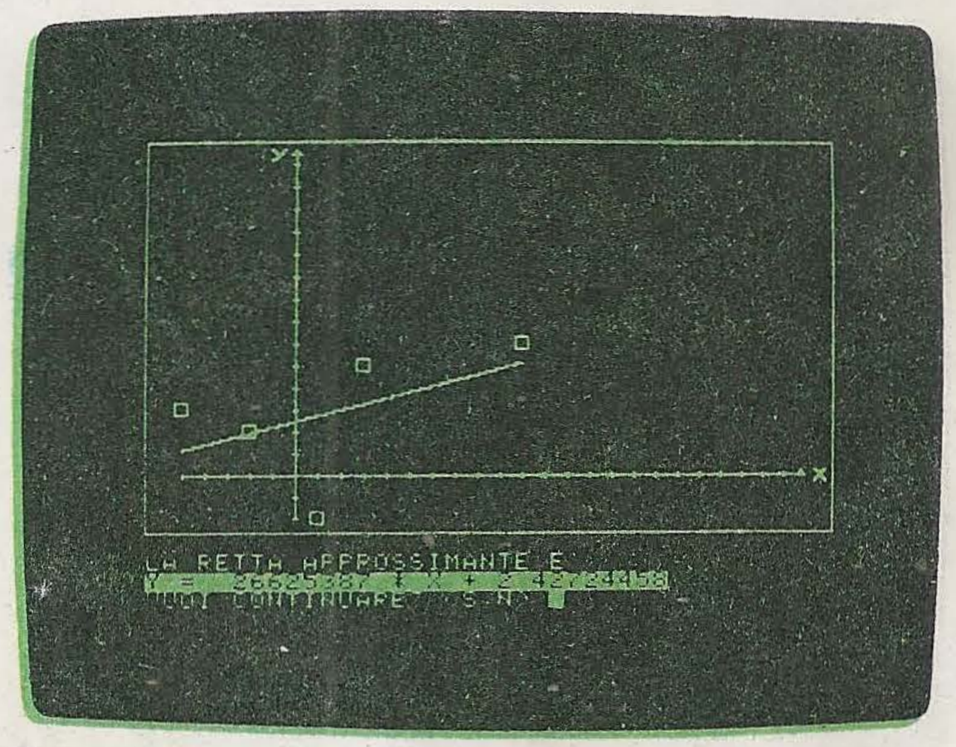
−2,2 koordinatlı nokta 2'de görülen ilk negatif değerin girilmesi ekseninin sağa doğru kaymasına neden olmakta, çünkü Y ekseninde (−2) değerinde bir negatif yere gereksinim olur



Daha sonraki girişler görüntü alanının ayarlanmasını yani başlangıç noktası ile ölçek faktörünün ayarlanmasını belirler.



Fotoğrafta dördüncü nokta az önce girilmiştir. Son noktanın (1,-2) girilmesi regrasyon doğrusunun hesabını ve görüntülenmesini başlatır.
Görüleceği gibi bu durumda değerler çok dağınıktır ve regrasyon doğrusu oldukça tatmin edici bir yaklaşık sunum vermektedir.



Ücretsiz PDF Kitap Arsivi - https://rantey.org

Herhangi Bir Fonksiyonun Grafiğinin Çizim Programı

Herhangi bir fonksiyonun grafik gösterimi, basit bir doğrununkine oranla çok daha fazla özen gerektirmektedir. Bunun için şu noktaları gözönünde bulundurmak gereklidir:

— Ölçek faktörünün hesaplanması ve normalizasyonu
— Hataların denetimi ve düzeltimi
— Kullanıcı tarafından simgelerin seçimi

Ölçek faktörünün hesaplanması çok basittir. Aynen iki eksene göre belirlenmiş ekran uzunluğu ile verilen değerin ekran üzerindeki nisbi uzunluğu hesaplanırken olduğu gibi hesaplanır. Bu şekilde Y ekseni üzerind e 200 noktalık bir ekranda, 20 ile 70 arasında bulunan değerler alan bir fonksiyonu göstermek için, ölçek faktörü 200/(70—20) = 200/50 = 4 olacaktır.

Bu değerden yola çıkarak Y(20)'nin en küçük değerine tam olarak uyması gereken başlangıç noktası Y0'ın koordinatına ek olarak eksenin ekran kenarlarına çok yakın çizilmesini önlemeye yarayacak bir fark hesaplanır.

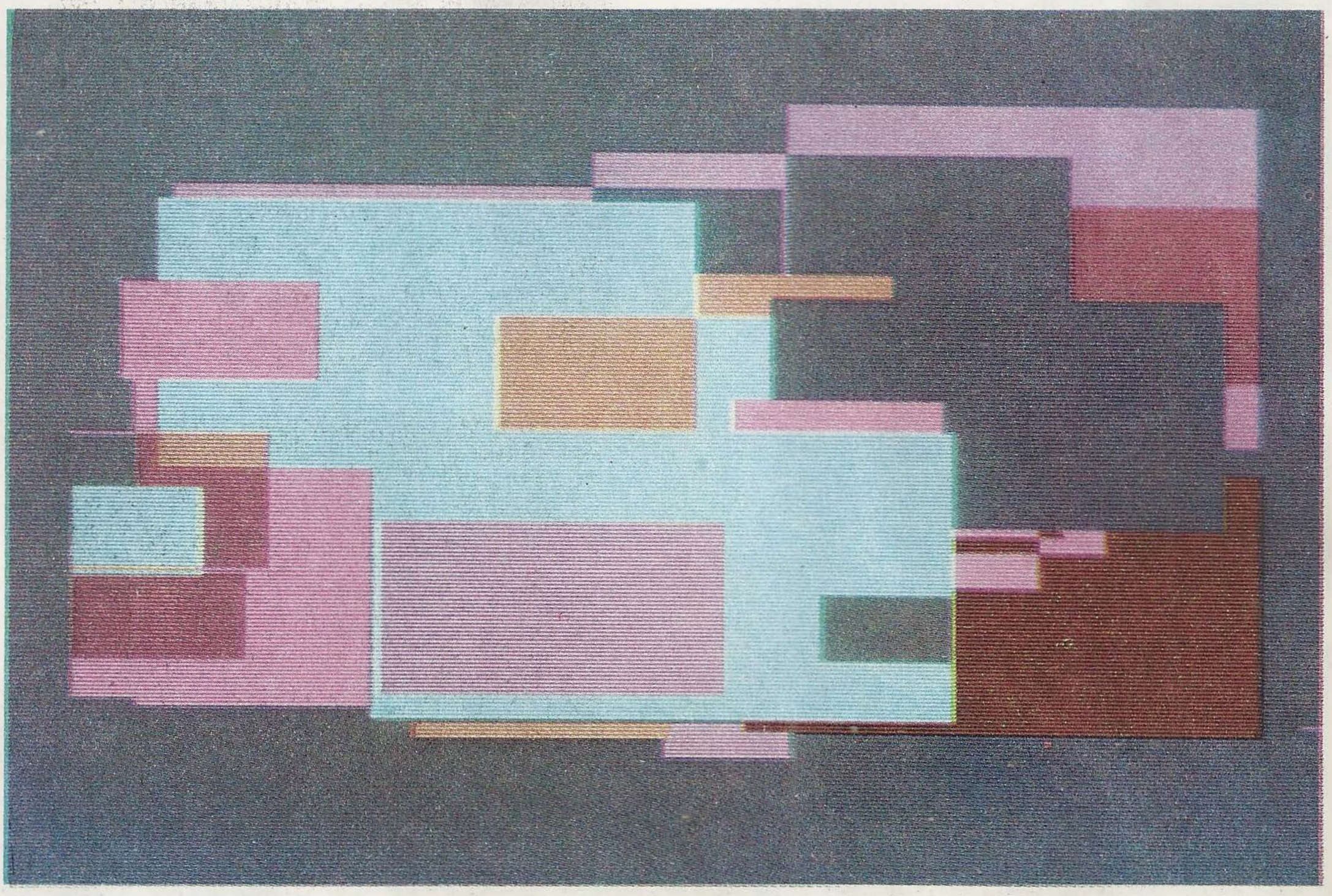
Bölümün sonucu genellikle bir tamsayı olmaz. Bir önceki örnekte fonksiyonun aldığı değerlerin 20 ila 71 arasında bulunması ölçek faktörünün ondalıklı bir değer alması dolayısıyla da zor yorumlanır olması için yeterlidir.

Bu sakıncanın önüne geçmek için kullanıcı tarafından belirlenen diğer değerler arasında en uygun olan değeri seçen bir normalizasyon öngörülür. Değerlerin 20 ve 71 olduğu bir önceki durumda 200/51 = 3.9 olan ölçek faktörü 3'e tamlanmalıdır. Sonuç ölçüleri daha küçük fakat yorumlanması daha kolay bir grafiktir. Bu durumda ekranın bütün alanı kullanılmış olur.

Fonksiyon ya negatif ya da pozitif değerler alir. Buna göre başlangıç noktası X0,Y0'ın konumunu, ekranı optimal (en verimli) şekilde değerlendirecek biçimde yerleştirmek uygun olur. Bunun yanı sıra fonksiyonun şekli çok küçük veya ekran sınırlarının dışına taşan çok büyük değerler veren bir şekil olabilir. Bu sorun 10 kuvvetine eşit bir çarpan faktörünün devreye sokulması ile çözülür.

Bir fonksiyonun grafiğinin 0.03 ile 0.07 arasında değiştiği durumda K = 102 faktörü devreye sokulur. Elbette ki kullanılan K değerini

Renkli bir ekranda üstüste bindirilmiş dikdörtgenlerin gösterimi.



Ücretsiz PDF Kitap Arsivi - https://rantey.org

PROGRAMDA KULLANILAN BAŞLICA DEĞİŞKENLER

V1	= Öngörülen en büyük ölçek faktörü
SX(4), SY(4)	= Simgeyi resimlemeye yarayan yer değişimleri
VN/10)	= Normalleştirme için öngörülen ölçek faktörleri
K$	= X ve Y ekseninin göstergesi
XS,YS	= Ekranda nokta sayısı olarak belirtilmiş koordinatlar
XM,XN	= X ekseninin en büyük ve en küçük değerleri
YM,YN	= Y ekseninin en büyük ve en küçük değerleri
FX,FY	= X ekseni ve Y ekseni ölçek faktörleri
DX	= X ekseni adımı
NP	= Gösterilecek olan nokta sayısı
LX,LY	= Eksenlerin uzunluğu
ER	= Ortaya çıkan en son hatanın kodu
LOC	= Hatanın gözden geçirildiği satırın numarası
K	= Ölçek faktöründe 10 ağırlığı
S1,S2	= Simgeler için ölçek faktörü
T$	= Grafik türleri (noktalar için 1, doğru parçaları için 2, simgeler için 3)
X0,Y0	= Eksenlerin başlangıç noktaları için koordinatlar

grafiğin üstünde belirtmek gereklidir hatta koordinatları hangi değer ile çarpmak gerektiğini belirtmek daha da iyi olur.

Hataların denetimi ve düzeltimi fonksiyonun belirsiz değerler alması durumunda gereklidir. Örneğin

$$Y = \frac{1}{X - 3}$$

fonksiyonu X = 3 noktasında sonsuz bir değer alacaktır. Düzgün bir program 3 değerine rasladığı anda bir aksaklık olduğu uyarısında bulunacaktır.

Bu denetimler olmasaydı ortaya programın durdurulmasıyla sonuçlanacak bir aşma (overflow) hatası çıkardı.

Kullanıcı tarafından belirlenen simgeler zorunlu olmamakla birlikte, daha hoş bir grafik gösterimi sağlamak açısından, yine de kullanışlı bir öge oluşmaktadır.

Bu sayfalarda sunduğumuz programda 3 tane görüntüleme sistemi öngördük:

— Noktalarla
— Doğru parçalarıyla
— Simgelerle

Birinci durumda fonksiyonun her bir noktası bir ekran noktası ile temsil edilir. Grafik ise birbirine az çok yakın olan bir noktalar silsilesi olarak gösterilir. İkinci yöntemde ise her nokta bir önceki nokta ile sonraki noktaya iki doğru parçası ile birleştirilir. O zaman eğri ortaya bir kırık çizgiler bütünü olarak çıkar. Son gösterim türü eğrinin her bir noktasının koordinatlarına karşılık olan ekran noktası üzerinde bir dörtgen çizerek elde edilir.

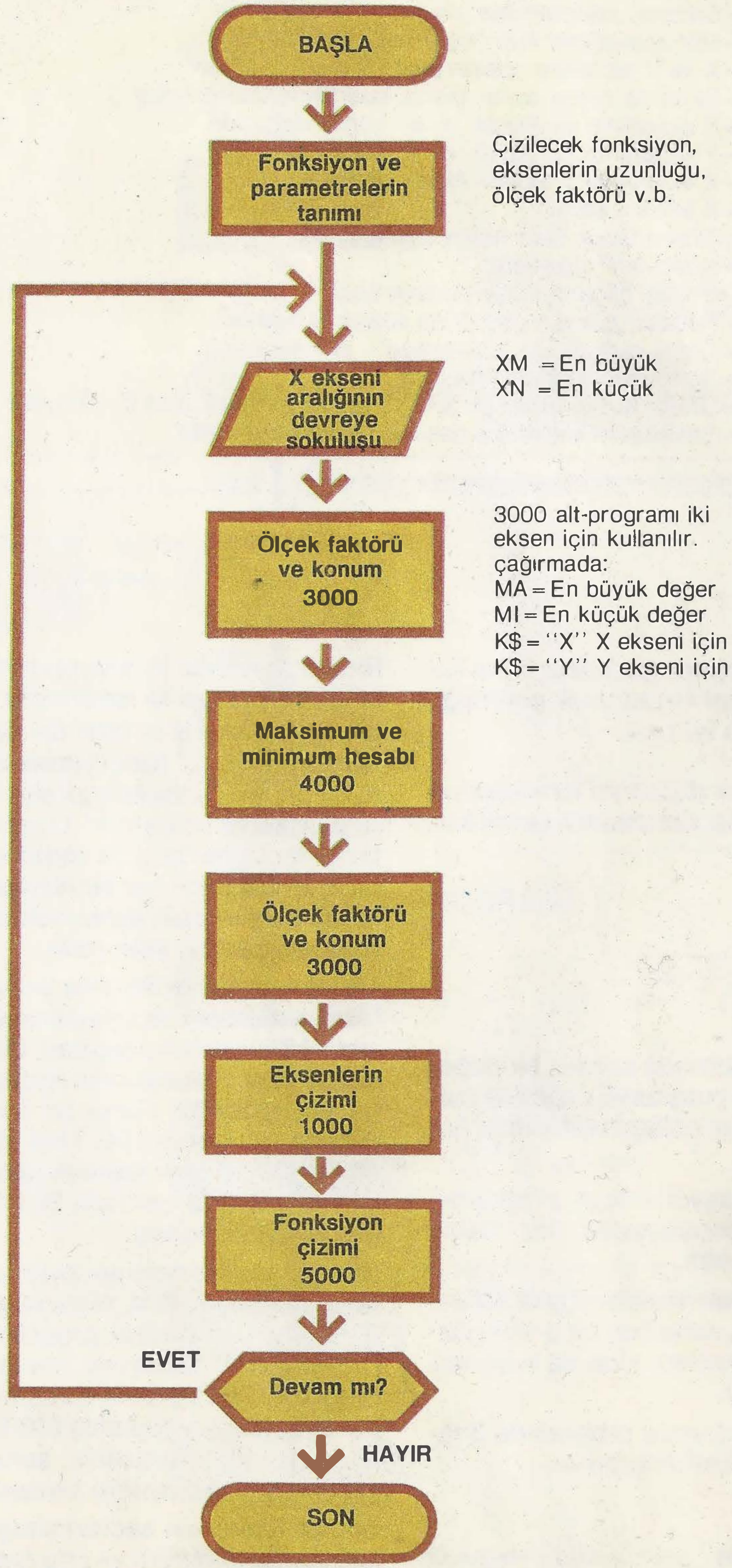
1464. sayfada verilen akış şemasındaki birinci öbek makinanın ve uygulamanın özgül parametrelerini (ekranın ölçüleri, ölçek faktörü) olduğu kadar çizilecek olan fonksiyonunkileri de tanımlamaktadır. Farklı bir fonksiyonun seçimi, tanımı içeren 100. satır DEF FN...'in yerine başka bir satır koyarak gerçekleştirilir. Bu işleme yalnızca çeviricili BASIC ile izin verildiği kaydedilmelidir.

Bundan sonra program bağımsız değişkenin aldığı değerlerin (XM,XN) aralığını ister. Bu değerlerden yola çıkarak program X ekseni üzerindeki ölçek faktörünü hesaplar (3000 altprogramı). XN'den XM'e kadarki değerler alanında bağımlı değişkenin (4000) en büyük ve en küçük değeri araştırılır, sonra da buna karşılık olan ölçek faktörü hesaplanır.

Son iki fonksiyon seçilen simgeler aracılığıyla eksenlerin (1000) ve grafiğin (5000) çizimine ilişkindirler.

Ücretsiz PDF Kitap Arsivi - https://rantey.org

BİR FONKSİYONUN GRAFİK GÖSTERİLİŞİ PROGRAMININ AKIŞ ŞEMASI



Ücretsiz PDF Kitap Arsivi - https://rantey.org

Ölçek faktörünün hesaplanması ve yuvarlatılması. Ölçek faktörünün hesaplanması alt-programı (1468. sayfadaki akış şemasına bakınız) üzerinde işlemin yapılacağı ekseni (X ya da Y) seçmek için K$ çentiğinden yararlanır. Çıkışta her eksenin ölçek faktörü (FX,FY) başlangıç noktası koordinatları X0,Y0 ve çarpım faktörü K elde edilir. Bu alt-programda maksimum ve minimum değerler aynı biçimde tamsayı olarak yuvarlatılır.

Yuvarlatma 6000 alt-programının çağrılmasıyla gerçekleştirilir.

Y'nin uç değerlerinin araştırılması (4000 alt-programı). Alt-programın akış şeması 1470. sayfada verilmektedir.

İşlem, her aralığın çeşitli noktaları için Y'nin değerlerinin hesaplanmasından ve her defasında en yüksek ve en düşük değerlerin belleğe alınmasından oluşur.

Grafiğin gösterilmesi (5000-7000 alt-programları). Öngörülen üç biçimden birine göre elde edilir (noktalar, doğru parçaları ve simgeler).

Noktalarla gösterim 7000 alt-programı tarafından gerçekleştirilir, doğru parçaları ya da simgeler biçiminde olanlar ise 5000 alt-programı tarafından yapılır (1471. sayfadaki akış şeması bu sonuncu çözümü örneklendirmektedir).

Simgeyi çizmek üzere satır 5900'ün çağrılması dışında (bkz. s.1472) 7000 alt-programı da aynıdır.

Hataların yönetimi (10000 alt-programı). 1473 ve 1474. sayfalardaki akış şeması hataların yönetimi alt-programını betimlemektedir. Makineye son derece bağlı, dolayısıyla da genelleştirilmesi güç olan tek bölüm budur.

Kullanılan mantık, Apple ve uyarlıları ile Personel Kid ve Siprel 2010'un mantığıdır. Diğer makineler için bazı hata kodlarını, bellek konumlarını değiştirmek ve kimi kez de başka bir mantık kullanmak gerekir. Bir hata yapıldığında buna denk düşen sayısal kod özel bir bellek konumuna yazılır (222), yanlış satırın numarası başka bir konumda bulunur.

Hata kodunu içeren değişkene ER, satır numarasını içerene LOC dersek, bu değerleri elde etme komutu

```
ER = PEEK (222)
LOC = PEEK (218) + PEEK (219)*256
```

olur.

İlki hemen kavranılabilir: PEEK komutu 222 numaralı bellek adresinin değerini çıkarır ve ER değişkenine yerleştirir.

İkinci satır ise birbirini izleyen iki bellek adresinden (218 ve 219) çıkarılan değerlerden yeni bir veri oluşturur. çünkü, her bellek adresi ancak 8 bit içerir, oysa içinde hatanın yer aldığı satır numarasının adreslenebilmesi için 16 bit gerekmektedir. Bu yüzden sistem, bu numarayı birbirini izleyen iki adrese yerleştirebileceği iki bölüme ayırır.

256 ile çarpma, adresin iki bölümünün birbirlerine göre doğru biçimde yerleştirilebilmesi için gereklidir (256 ile çarpma 8 bit ilerletme anlamına gelir).

Belirtilen makine için geçerli olan hata kodları şunlardır:

```
 55 = Doğru olmayan nicelik
133 = 0'a bölme
254 = Giriş yanlışı
 69 = Taşma (overflow)
191 = Çok karmaşık formül
```

Başka makinelerde, hata yönetimi yukarıda anlatılana yakınsa, kodların karşılıklı olarak doğrulanması ve gerekiyorsa değiştirilmesi yerinde olur.

Karakterlerin Grafik Biçimde Görselleştirmesi

Grafik uygulamalar da çoğu kez, alfabetik ya da sayısal karakterlerin ekrandaki görüntüye yerleştirilmesini gerektirir, örneğin bir grafiğe ilişkin açıklayıcı altyazıların konması. Yüksek kararlılıktaki **(high resolution)** monitör, ekranın her noktasını ayrı ayrı yönetecek biçimde kurulmuşsa, metin biçiminde kullanılan standart ASCII karakterlerini görselleştirmek artık mümkün değildir. Bu son biçimde ekran, artık karakter matrisleriyle yönetilir: Bu durumda bir satırla bir sütunun kesiştiği her noktada yer alan ''hane''ye bir karakter yerleştirmek mümkündür. Her hane, bir ekran noktaları matrisinden oluşur ve gösterilmesi istenen karaktere göre bu noktalardan bazıları sistem tarafından kendiliğinden harekete geçirilir.

Gösterilecek karakterin biçimi karakter ile ekran noktaları arasında bire bir denklik olmadığından bütün ekran noktaları teker teker yönetilebilir.

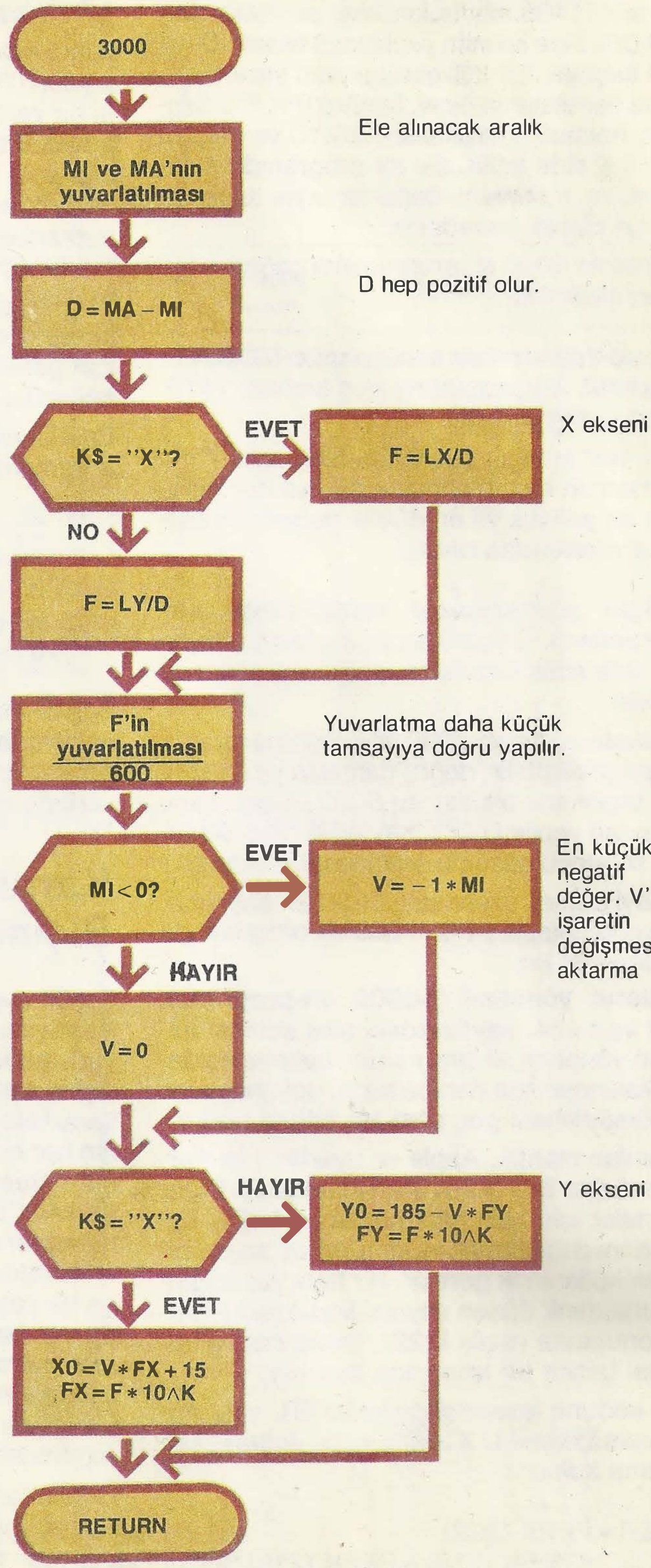
Bu durumda, ekran metinleri (metin modu) ya

Ücretsiz PDF Kitap Arsivi - https://rantey.org

ÖLÇEK FAKTÖRÜNÜN HESAPLANMASI VE YUVARLATILMASI

Girişler:
MA = Çizilecek en yüksek değer
MI = En düşük değer
K$ = X ekseni için ''X''
Y ekseni için ''Y''

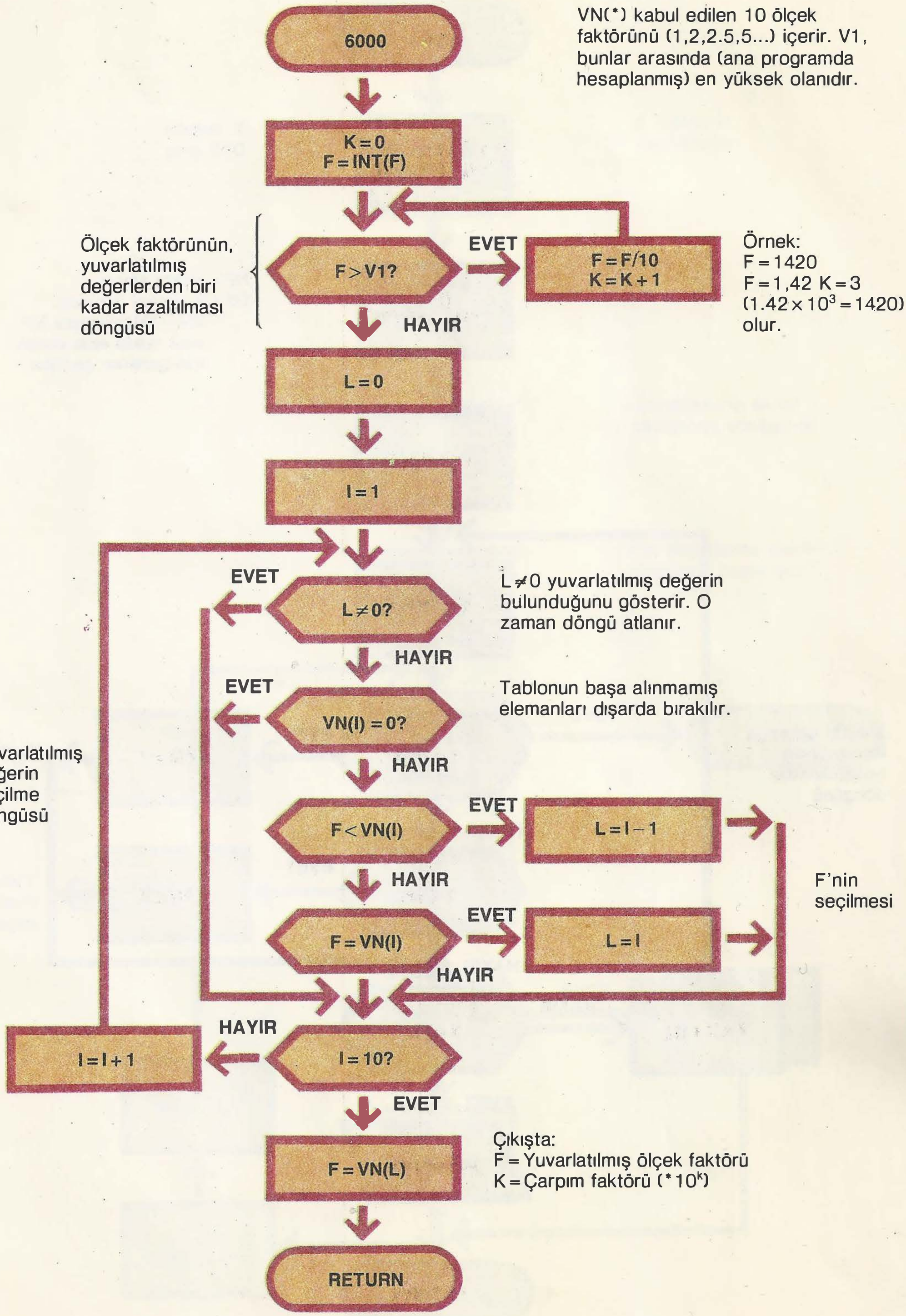
Çıkışlar:
F = Ölçek faktörü
X0,Y0 = Başlangıç noktası koordinatları



Bu alt-program her eksen için birer kez olmak üzere 2 kez kullanılır. Seçme K$ çentiğinin yardımıyla yapılır.

Ücretsiz PDF Kitap Arsivi - https://rantey.org

ÖLÇEK FAKTÖRÜNÜN YUVARLATILMASI



VN(*) kabul edilen 10 ölçek faktörünü (1,2,2.5,5...) içerir. V1, bunlar arasında (ana programda hesaplanmış) en yüksek olanıdır.

Ölçek faktörünün, yuvarlatılmış değerlerden biri kadar azaltılması döngüsü

Örnek:
F = 1420
F = 1,42 K = 3
($1.42 \times 10^3 = 1420$) olur.

L≠0 yuvarlatılmış değerin bulunduğunu gösterir. O zaman döngü atlanır.

Tablonun başa alınmamış elemanları dışarda bırakılır.

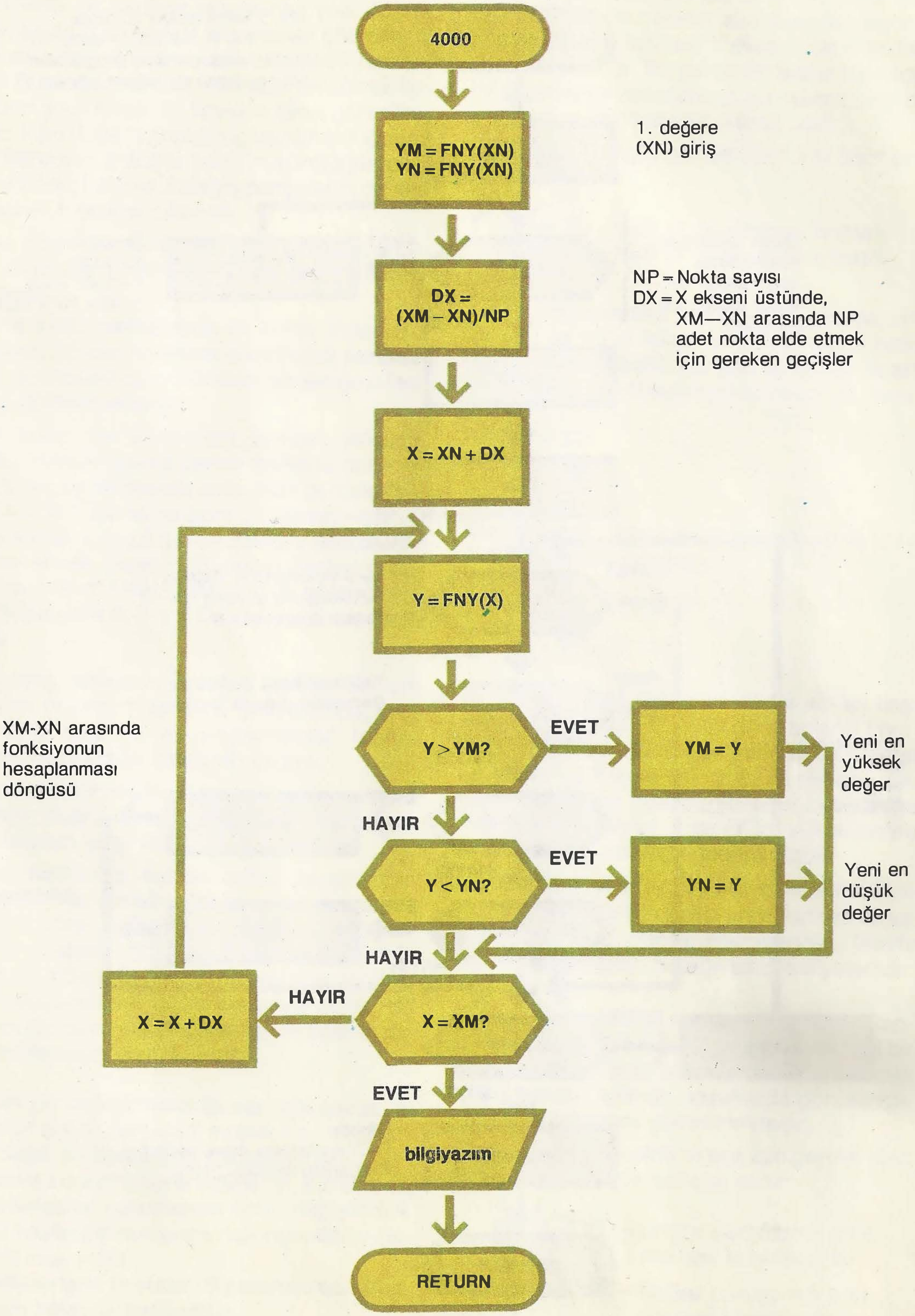
Yuvarlatılmış değerin seçilme döngüsü

F'nin seçilmesi

Çıkışta:
F = Yuvarlatılmış ölçek faktörü
K = Çarpım faktörü ($*10^k$)

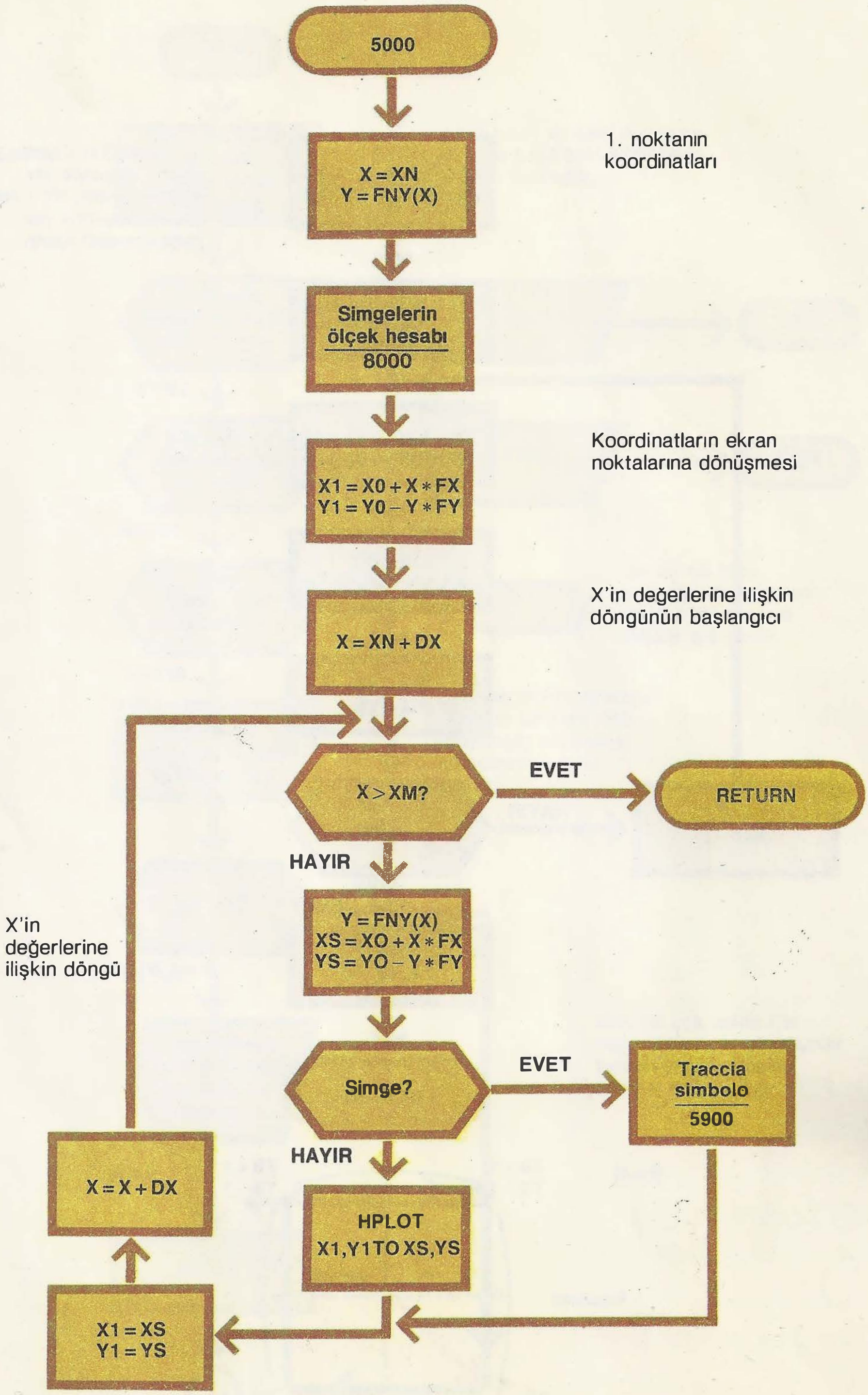
Ücretsiz PDF Kitap Arsivi - https://rantey.org

MAKSİMUM VE MİNİMUM'UN ARANMASI



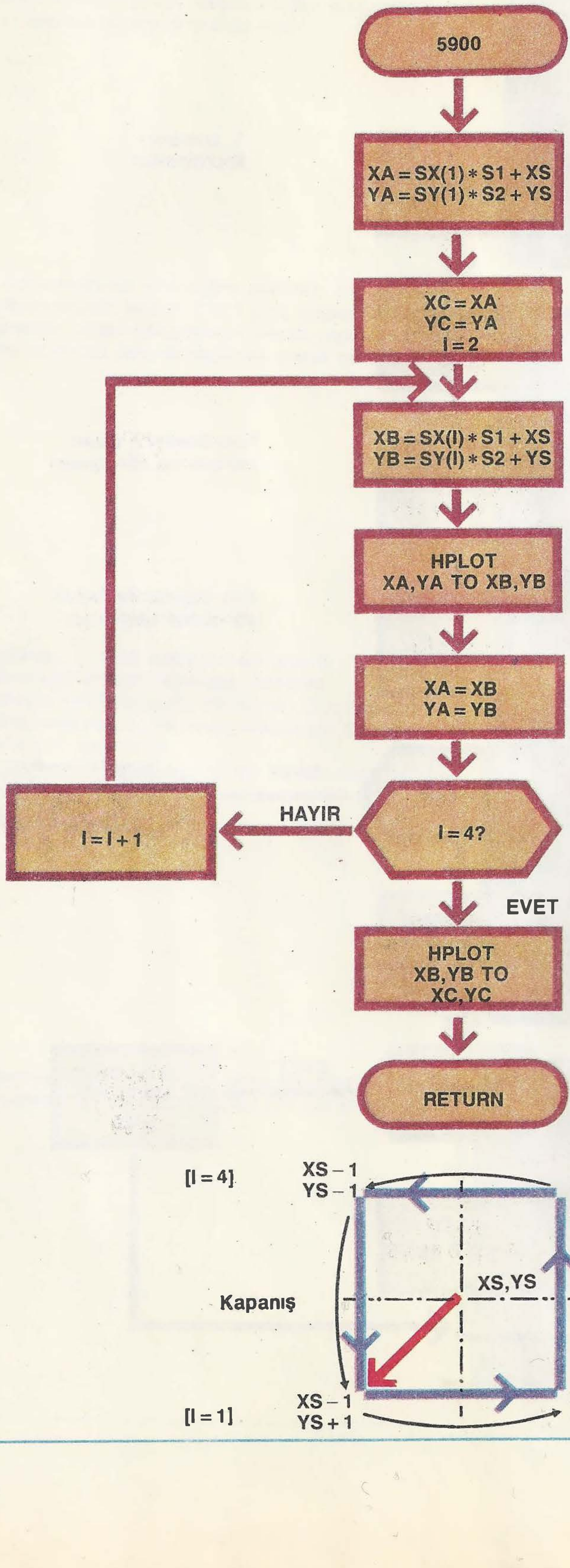
Ücretsiz PDF Kitap Arsivi - https://rantey.org

EĞRİNİN DOĞRU PARÇALARI YA DA SİMGELERLE GÖSTERİLMESİ



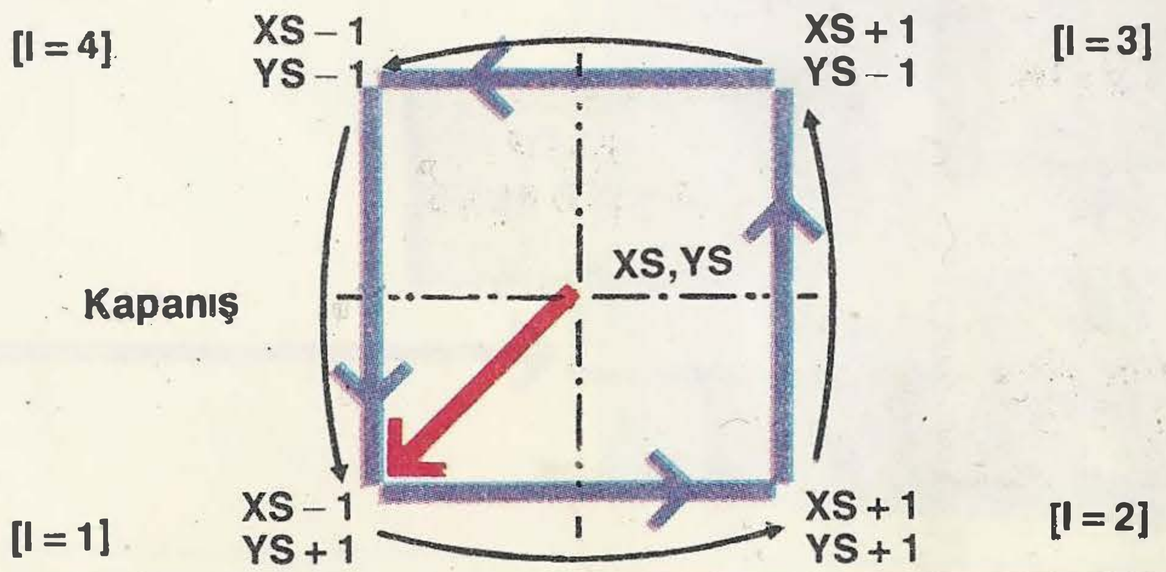
Ücretsiz PDF Kitap Arsivi - https://rantey.org

MERKEZİ XS, YS'DE BİR SİMGENİN ÇİZİMİ



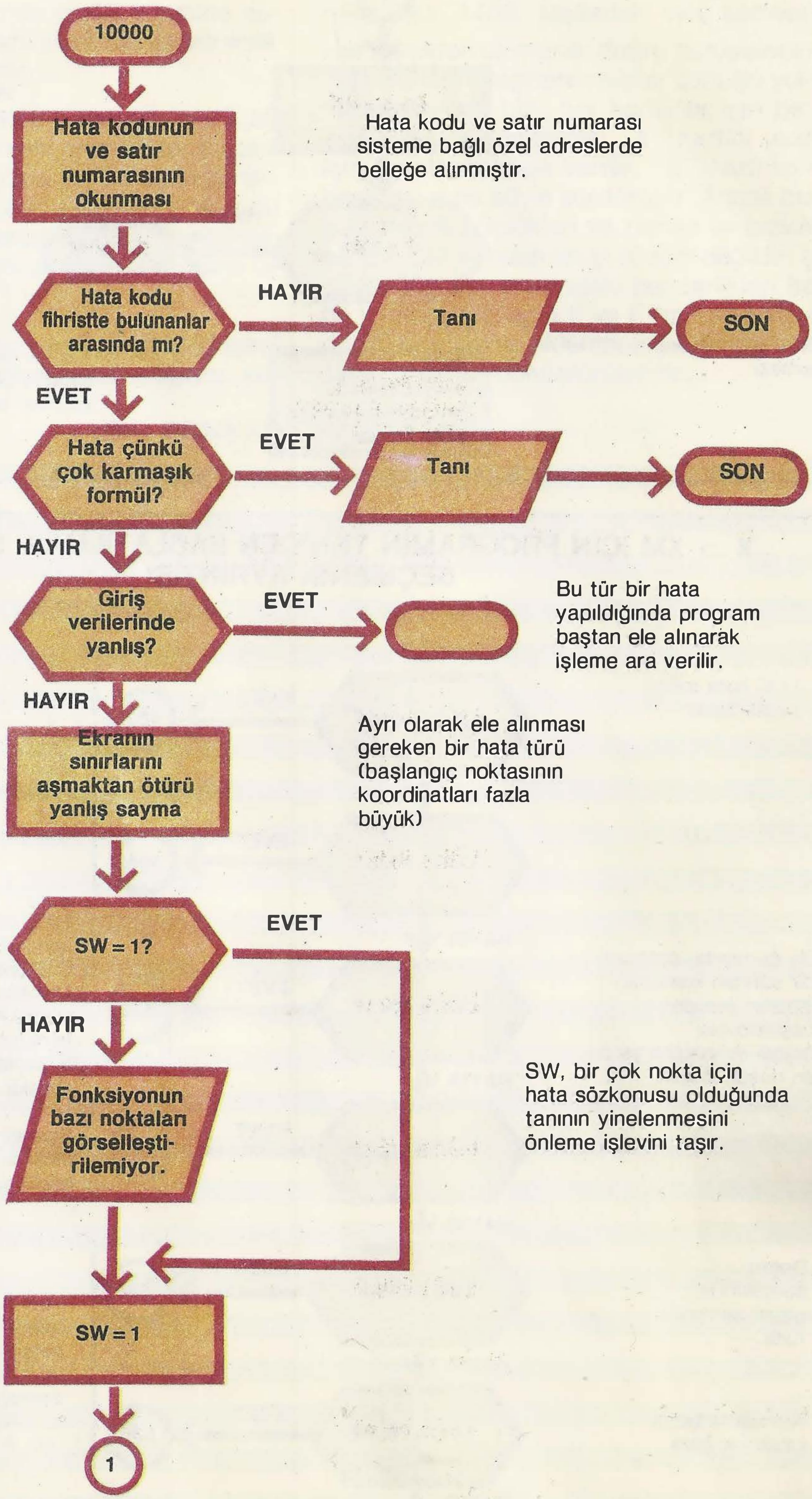
1. nokta. SX( ) tablosu X ekseni boyunca yer değiştirmeleri, SY ( )ise Y ekseni boyunca yer değiştirmeleri içerir.

Simgenin sona ermesi



Ücretsiz PDF Kitap Arsivi - https://rantey.org

HATA YÖNETİM ALT-PROGRAMI



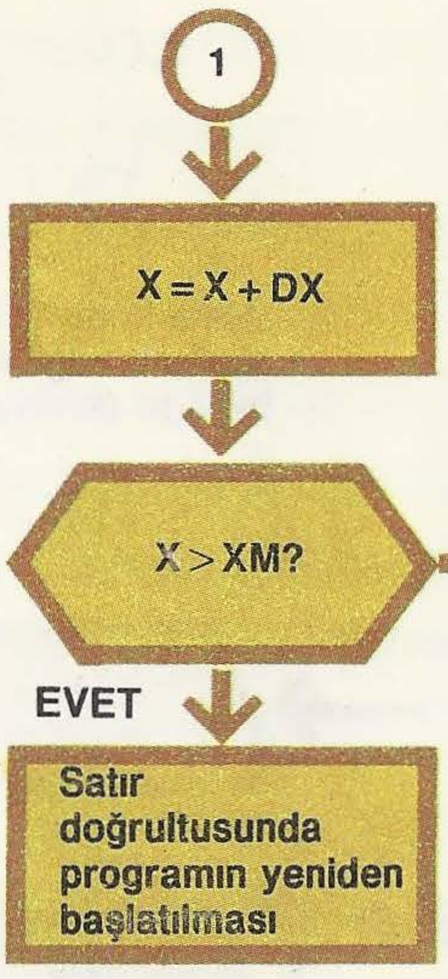
Hata kodu ve satır numarası sisteme bağlı özel adreslerde belleğe alınmıştır.

Bu tür bir hata yapıldığında program baştan ele alınarak işleme ara verilir.

Ayrı olarak ele alınması gereken bir hata türü (başlangıç noktasının koordinatları fazla büyük)

SW, bir çok nokta için hata sözkonusu olduğunda tanının yinelenmesini önleme işlevini taşır.

Ücretsiz PDF Kitap Arsivi - https://rantey.org



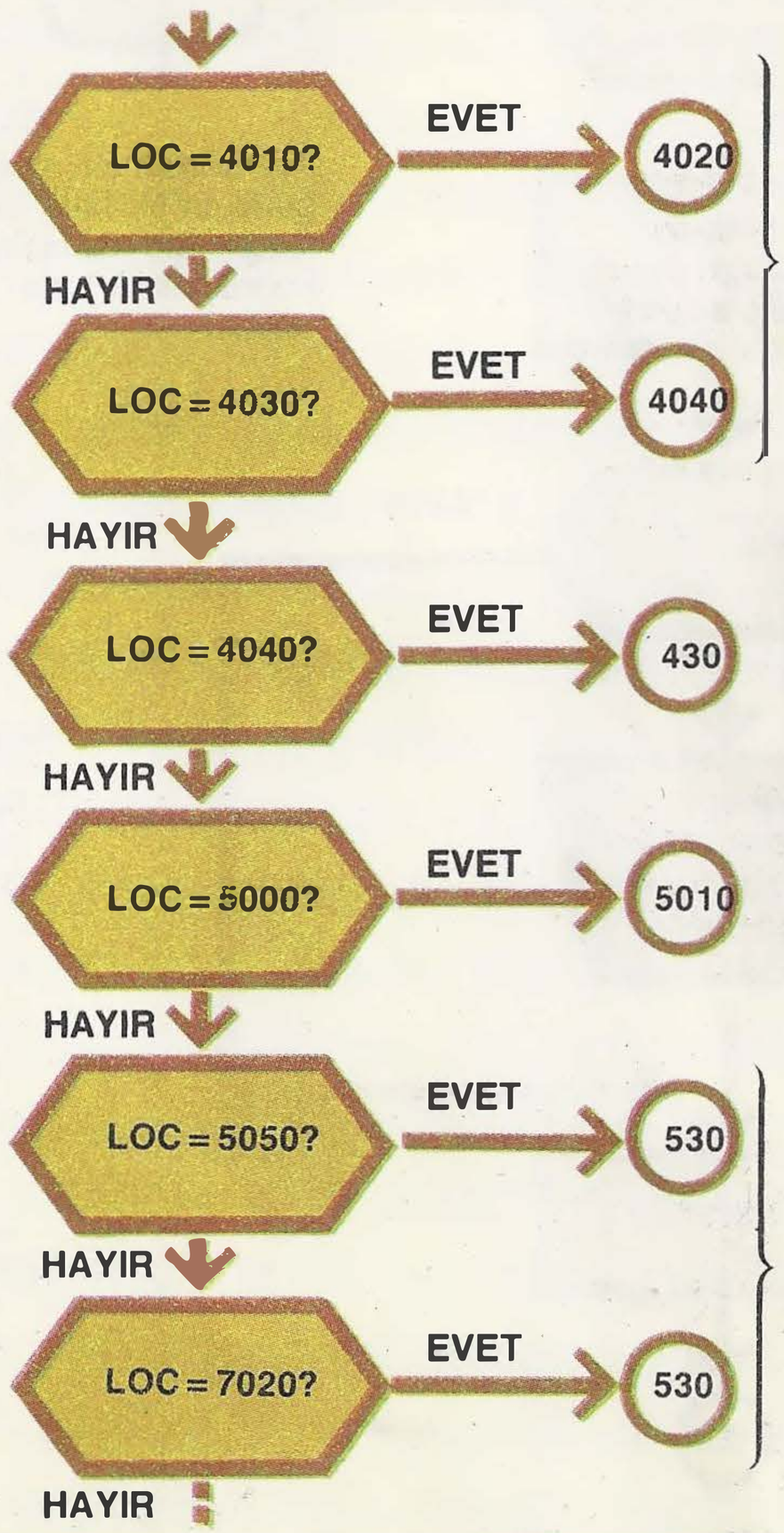
Fonksiyonun bu noktası çizilemeyeceğinden atlanan bir nokta

X ekseninin değeri, en yüksek değerden küçük, program yeniden başlatılabilir.

Bkz. ayrıntılı akış şeması

X > XM ise, program, hata satırına göre, farklı noktalarda yeniden başlatılabilir.

X > XM İÇİN PROGRAMIN YENİDEN BAŞLATILMASI SIRASINDA SEÇMENİN AYRINTİSI



LOC hata satırı numarasıdır

Sonraki satırlarda düzeltme

Bu durumda, program bir sonraki komuttan itibaren yeniden başlatılamaz (hatalı en yüksek ya da en düşük değer)

Hata döngünün içindedir (4030/4070). Maksimum ve minimumların yeniden hesaplanması gerekir.

Sonraki komuttan itibaren yeniden başlatma

Doğru parçalarının çizimi sırasında hata

Noktalarla çizim sırasında hata

Yeni bir fonksiyonun olası girişiyle ara verme ve sürdürmek için herhangi bir karakteri bekleme

Ücretsiz PDF Kitap Arsivi - https://rantey.org

da grafikleri (grafik modu) görselleştirebilir ama ikisinin birarada bulunması olanaksızdır. Bununla birlikte bazı makineler, metin moduna birkaç satır ayrılması (genellikle ekranın altında), geri kalanların da grafik biçimine sokulmasıyla karma kullanımlara olanak verecek biçimde düzenlenmiştir.

Alfabetik ya da sayısal bir karakterin grafik bir sayfada görselleştirilmesi, yazılım tarafından bu karakterin nokta nokta kurulmasını gerektirir. Kişisel ve küçük bilgisayarların çoğu bu işlem için uygun alt-programlarla donatılmamış, sorunun çözümü kullanıcının kendine bırakılmıştır.

Kurma tekniği, harflerin ve sayıların, düşsel bir kalemin temel hareketleri aracılığıyla, şematize edilebilir biçimler halinde görselleştirilmesine dayanır. Grafik modunda herhangi bir ''çizim'' elde etmek için, karşılık gelen programı harf harf ya da sayı sayı başlatmak gerekir (bkz. 1485. sayfadaki akış şeması).

Her karakteri oluşturan doğru parçalarının seçimi dışında programın hiçbir zorluğu yoktur. Yöntemlerden biri, her karakter için bir alt-programın yazılmasıdır: ''a'' harfini yazmak için bir alt-program vardır, ''b'' harfi için bir başkası ve bu böyle sürdürülür. Ancak bu çözüm teknik güçlükleri ve zaman ve bellek tasarrufu bakımından en iyi çözüm değildir. çünkü karakterler arasındaki benzerlikleri hesaba katmaz. Örneğin F ve E harfleri yatay bir doğru parçasının eklenmesi ya da kaldırılmasıyla birbirine dönüştürülebilir.

BİR FONKSİYONUN ÇİZİMİ

```
90      REM ------------------------------------
91      REM MATEMATIKSEL FONKSIYONLAR
92      REM ------------------------------------
93      REM
94      REM
95      REM
96      REM
97      REM ------------------------------------
98      REM BASA ALMA
99      REM ------------------------------------
100     DEF FN Y(X) = X * SIN (X)
110     LX = 110
      : LY = 180
      : SW = 0
120     FOR J = 1 TO 4
      : READ SX(J), SY(J)
      : NEXT J
130     DATA -1, 1,1,1,1,-1,-1,-1
140     ONERR GOTO 10000
150     V1 = 1
160     FOR J = 1 TO 10
      : READ VN(J)
      : IF VN(J) > V1 THEN V1 = VN(J)
170     NEXT J
180     DATA 1,2,3,4,5,6,7,8,9,0
190     REM ------------------------------------
200     REM MENU
210     REM ------------------------------------
220     TEXT
      : HOME
      : GOSUB 2000
230     HTAB 3
      : VTAB 5
      : INPUT "X EKSENININ GIRISLERI ?";XN,XM

      : HTAB 39
      : VTAB 5
      : PRINT "*"
240     IF XM < = XN THEN 220
250     IF XM - XN > 250 THEN 220
260     HTAB 3
      : VTAB 10
      : INPUT "NOKTA SAYISI?";NP
      : HTAB 39
      : VTAB 10
      : PRINT "*"
```


Ücretsiz PDF Kitap Arsivi - https://rantey.org

```
270     HTAB 3
      : VTAB 15
      : PRINT "GRAFIK TIPI:
        "
280     HTAB 3
      : PRINT "1)NOKTALARLA"
290     HTAB 3
      : PRINT "2)DOGRU PARCALARIYLA"
300     HTAB 3
      : PRINT "3)SIMGELERLE"
310     VTAB 15
      : HTAB 30
      : INPUT "";T$
      : VTAB 15
      : HTAB 39
      : PRINT "*"
320     GOSUB 9000
330     REM ----------------------
340     REM ANA PROGRAM
350     REM ----------------------
360     IF XN < 0 AND XM < = 0 THEN MI
            = XN
      : MA = 0
370     IF XN < 0 AND XM > 0 THEN MI = XN

      : MA = XM
380     IF XN > = 0 AND XM > 0 THEN MI
          = 0
      : MA = XM
390     IF MA > 250 THEN MA = 250
400     IF MI < - 250 THEN MI = - 250
410     K$ = "X"
      : GOSUB 3000
420     GOSUB 4000
430     IF YN < 0 AND YM < = 0 THEN MI
            = YN
      : MA = 0
440     IF YN < 0 AND YM > 0 THEN MI= YN

      : MA = YM
450     IF YN > = 0 AND YM > 0 THEN MI
              = 0
      : MA = YM
460     IF MA > 250 THEN MA = 250
470     IF MI < -250 THEN MI = - 250
480     K$ = "Y"
      : GOSUB 3000
490     HGR2
      : HCOLOR= 3
500     GOSUB 1000
510     IF T$ = "1" THEN GOSUB 7000
      : GOTO 530
520     GOSUB 5000
530     PRINT CHR$ (7)
      : GET A$
540     TEXT
      : HOME
      : VTAB 10
      : INPUT "FONKSIYON DEGISTIRMEK ISTERMISINIZ? (E/H)
          ";A$
550     IF LEFT$ (A$,1) = "N" THEN
            RUN
560     PRINT
570     INVERSE
      : PRINT "SATIR 100 DEKI FONKSIYONUN DEGISTIRILMESI

        VE RUN YAPILMASI "
      : NORMAL
580     LIST 100
      : END
997     REM ----------------------------------------
998     REM EKSENLERIN BICIMLENDIRILMESI
```


Ücretsiz PDF Kitap Arsivi - https://rantey.org

```
999     REM
1000    HPLOT 0,0 TO 279,0 TO 279,191
          TO 0,191 TO 0,0
1010    IF D > 191 THEN YO = 96
1020    HPLOT 15,YO TO 265,YO
      : HPLOT XO,185 TO XO,5
1030    N2 = 5
      : FOR N1 = 2 TO 0 STEP -1
      : HPLOT XO - N1,N2 TO XO + N1,N2
      : N2 = N2 -1
      : NEXT N1
1040    N2 = 265
      : FOR N1 = 2 TO 0 STEP -1
      : HPLOT N2,YO - N1 TO N2, YO + N1
      : N2 = N2 + 1
      : NEXT N1
1050    HPLOT 271,YO - 2 TO 276,YO + 3
      : HPLOT 276,YO - 2 TO 271,YO + 3
1060    HPLOT XO - 10,3 TO XO - 7,6
      : HPLOT XO -4,3 TO XO - 10,8
1070    IF FX = 1 THEN 1140
1080    FOR G = XO TO 15 STEP - FX
1090    HPLOT G,YO + 1 TO G,YO - 1
1100    NEXT G
1110    FOR G = XO TO 265 STEP FX
1120    HPLOT G,YO + 1 TO G,YO - 1
1130    NEXT G
1140    IF FY = 1 THEN 1210
1150    FOR G = YO TO 5 STEP - FY
1160    HPLOT XO + 1,G TO XO - 1,G
1170    NEXT G
1180    FOR G = YO TO 185 STEP FY
1190    HPLOT XO + 1,G TO XO - 1,G
1200    NEXT G
1210    RETURN
1997    REM ----------
1998    REM GOSTERME
1999    REM ----------
2000    A$ = "*"
2010    FOR J = 1 TO 23
2020    HTAB = 1
      : VTAB = J
      : PRINT A$
2030    NEXT J
2040    FOR J = 2 TO 39
2050    HTAB J
      : VTAB 23
      : PRINT A$
2060    NEXT J
2070    FOR J = 23 TO 1 STEP -1
2080    HTAB 39
      : VTAB J
      : PRINT A$
2090    NEXT J
2100    FOR J = 39 TO 2 STEP -1
2110    HTAB J
      : VTAB 1
      : PRINT A$
2120    NEXT J
2130    VTAB,1
      : HTAB 10
      : INVERSE
      : PRINT "MATEMATIKSEL FONKSIYONLAR "
      : NORMAL
2140    RETURN
2997    REM ----------
2998    REM YUVARLATMA
2999    REM ----------
3000    MA% = INT (MA)
      : MI% = INT (MI)
```


Ücretsiz PDF Kitap Arsivi - https://rantey.org

```
3010     IF MA > 0 AND MA% < > (MA)
            THEN MA = MA% + 1
         : GOTO 3030
3020     MA = MA%
3030     IF MI > 0 AND MI% < > MI THEN MI
               = MI% + 1
         : GOTO 3050
3040     MI = MI%
3047     REM ------------------------------------
3048     REM OLCEK FAKTORUNUN HESAPLANMASI
3049     REM ------------------------------------
3050     D = MA - MI
3060     IF K$ = "X" THEN F = LX / D
         : GOTO 6000
3070     F = LY / D
3080     GOSUB 6000
         : REM OLCEK FAKTORUNUN YUVARLATILMASI

3090     IF MI < 0 THEN V = - MI
         : GOTO 3110
3100     V = 0
3110     IF K$ = "X" THEN FX = F * 10  K
         : XO = V * FX + 15
         : RETURN
3120     FY = F * 10  K
         : YO = 185 - V * FY
         : RETURN
3997     REM ------------------------------
3998     REM  MAKSIMUM VE MINIMUM
3999     REM ------------------------------
4000     DX = (XM - XN) / NP
         : X = XN
4010     YN = FN Y(X)
4020     YM = FN Y(X)
4030     FOR X = XN + DX TO XM STEP DX
4040     Y = FN Y(X)
4050     IF Y > YM THEN YM = Y
4060     IF Y < YN THEN YN = Y
4070     NEXT X
4080     REM MAKSIMUM VE MINIMUMUN YAZILMASI
4900     RETURN
4997     REM ------------------------------------
4998     REM DOGRU PARCALARININ GRAFIGI
4999     REM ------------------------------------
5000     X = XN
         : Y = FN Y(X)
5010     GOSUB 8000
5020     X1 = XO + X * FX
5030     Y1 = YO - Y * FY
5040     FOR X = XN + DX TO XM STEP DX
5050     Y = FN Y(X)
5060     XS = XO + X * FX
5070     YS = YO - Y * FY
5080     IF T$ = "3" THEN GOSUB 5900
         : GOTO 5100
5090     HPLOT X1,Y1 TO XS,YS
5100     X1 = XS
         : Y1 = YS
5110     NEXT X
5120     RETURN
5897     REM ------------------------------------
5898     REM SIMGELERIN BICIMLENDIRILMESI
5899     REM ------------------------------------
5900     XA = SX(1) * S1 + XS
         : YA = SY(1) * S2 + YS
5910     XC = XA
         : YC = YA
5920     FOR I = 2 TO 4
5930     XB = SX(I) * S1 + XS
         : YB = SY(I) * S2 + YS
```


Ücretsiz PDF Kitap Arsivi - https://rantey.org

```
5940     HPLOT XA,YA TO XB,YX
5950     XA = XB
       : YA = YB
5960     NEXT I
5970     HPLOT XB,YX TO XC,YC
5990     RETURN
5997     REM ------------------------------
5998     REM YUVARLATMA
5999     REM ------------------------------
6000     K = 0
       : F = INT (F)
6010     IF F > V1 THEN F = F / 10
       : K = K + 1
       : GOTO 6010
6020     L = 0
6030     FOR I = 1 TO 10
6040     IF L < >  OR VN(I) = 0 THEN 6070

6050     IF F < VN(I) THEN L = I - 1
6060     IF F = VN(I) THEN L = I
6070     NEXT I
6080     F = VN(L)
6090     RETURN
6997     REM ------------------------------
6998     REM GRAFIK NOKTALAR
6999     REM ------------------------------
7000     GOSUB 8000
7010     FOR X = XN TO XM STEP DX
7020     Y = FN Y(X)
7030     XS = XO + X * FX
7040     YS = YO - Y * FY
7050     HPLOT XS,YS
7060     NEXT X
7070     RETURN
7997     REM ------------------------------
7998     REM SIMGELERIN OLCEKLERI
7999     REM ------------------------------
8000     S1 = INT (FX /.4)
       : S1 <  1 THEN S1 = 1
8010     S2 = INT (FY / 4)
       : IF S2 < 1 THEN S2 = 1
8020     IF S1 < S2 THEN S2 = S1
       : RETURN
8030     S1 = S2
8040     RETURN
8997     REM ------------------------------
8998     REM ARA
8999     REM ------------------------------
9000     HOME
       : HTAB 12
       : VTAB 12
       : FLASH
       : PRINT "                    "
       : NORMAL
       : RETURN
9997     REM ------------------------------
9998     REM HATA
9999     REM ------------------------------
10000    ER = PEEK (222)
       : LOC = PEEK (218) + PEEK (219)
           * 256
10010    IF ER < > 53 AND ER < > 133
           AND ER < > 254 AND ER < > 69
           AND ER < > 191 THEN HOME
       : PRINT " HATA N.";ER
       : END
10020    IF ER = 191 THEN HOME
       : HTAB 12
       : VTAB 12
       : INVERSE
```


Ücretsiz PDF Kitap Arsivi - https://rantey.org

```
      : PRINT "FORMUL COK KARMASIK"
      : NORMAL
      : STOP
10030 IF ER = 254 THEN RUN
10040 REM TANINMAYAN NICELIK
10050 IF XS > 279 THEN XS = 279
      : RESUME
10060 IF XS < 0 THEN XS = 0
      : RESUME
10070 IF YS > 191 THEN YS = 191
      : RESUME
10080 IF YS < 0 THEN YS = 0
      : RESUME
10090 IF XA > 279 THEN XA = 279
      : RESUME
10100 IF XA < 0 THEN XA = 0
      : RESUME
10110 IF YA > 191 THEN YA = 191
      : RESUME
10120 IF YA < 0 THEN  YA = 0
      : RESUME
10130 IF XB > 279 THEN XB = 279
      : RESUME
10140 IF XB < 0 THEN XB = 0
      : RESUME
10150 IF YB > 191 THEN YB = 191
      : RESUME
10160 IF YB < 0 THEN YB = 0
      : RESUME
10170 IF X1 > 279 THEN X1 = 279
      : RESUME
10180 IF X1 < 0 THEN X1 = 0
      : RESUME
10190 IF Y1 > 191 THEN Y1 = 191
      : RESUME
10200 IF Y1 < 0 THEN Y1 = 0
      : RESUME
10210 IF XC > 279 THEN XC = 279
      : RESUME
10220 IF XC < 0 THEN XC = 0
      : RESUME
10230 IF YC > 191 THEN YC = 191
      : RESUME
10240 IF YC < 0 THEN YC = 0
      : RESUME
10250 IF SW = 1 THEN 10300
10260 TEXT
      : HOME
      : VTAB 12
      : INVERSE
      : PRINT " FORMUL ARALIK ICINDE
          BELLI BIR NOKTA
        TANIMLAMIYOR "
      : NORMAL
10270 VTAB 23
      : PRINT "DEVAM ICIN BIR TUSA
        BASINIZ";
      : GET A$
10280 GOSUB 9000
10290 SW = 1
10300 X = X + DX
      : IF X > XM THEN 10320
10310 RESUME
10320 IF LOC = 4010 THEN 4020
10330 IF LOC = 4020 THEN 4030
10340 IF LOC = 4040 THEN 430
10350 IF LOC = 5000 THEN 5010
10360 IF LOC = 5050 THEN 530
10370 IF LOC = 7020 THEN 530
```

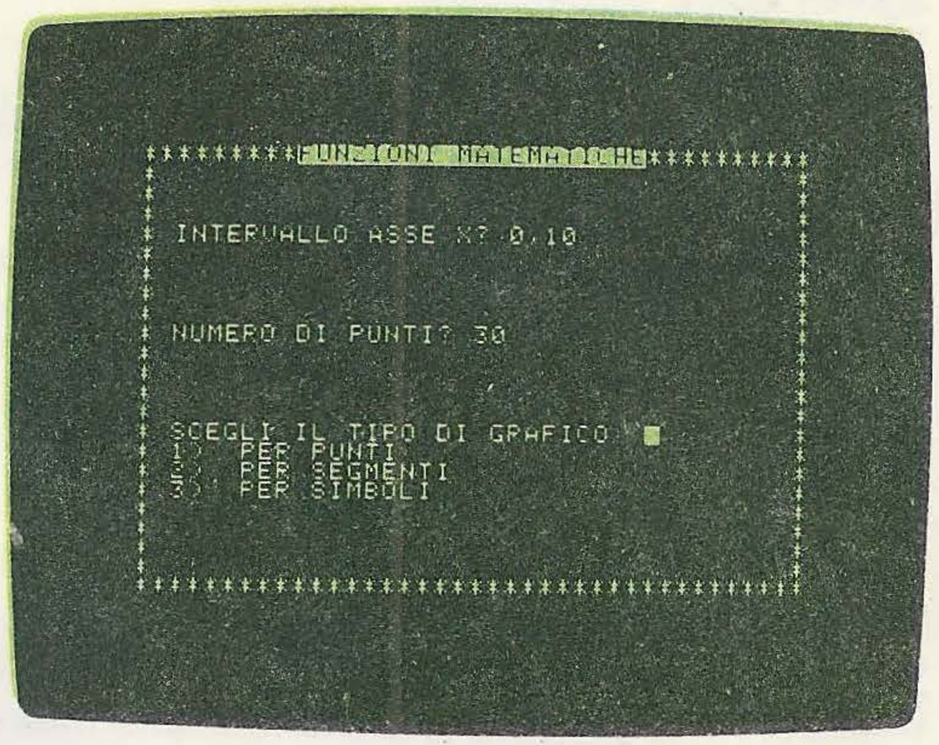

Ücretsiz PDF Kitap Arsivi - https://rantey.org

BİR FONKSİYONUN GRAFİĞİNİN GÖRSELLEŞTİRİLMESİ

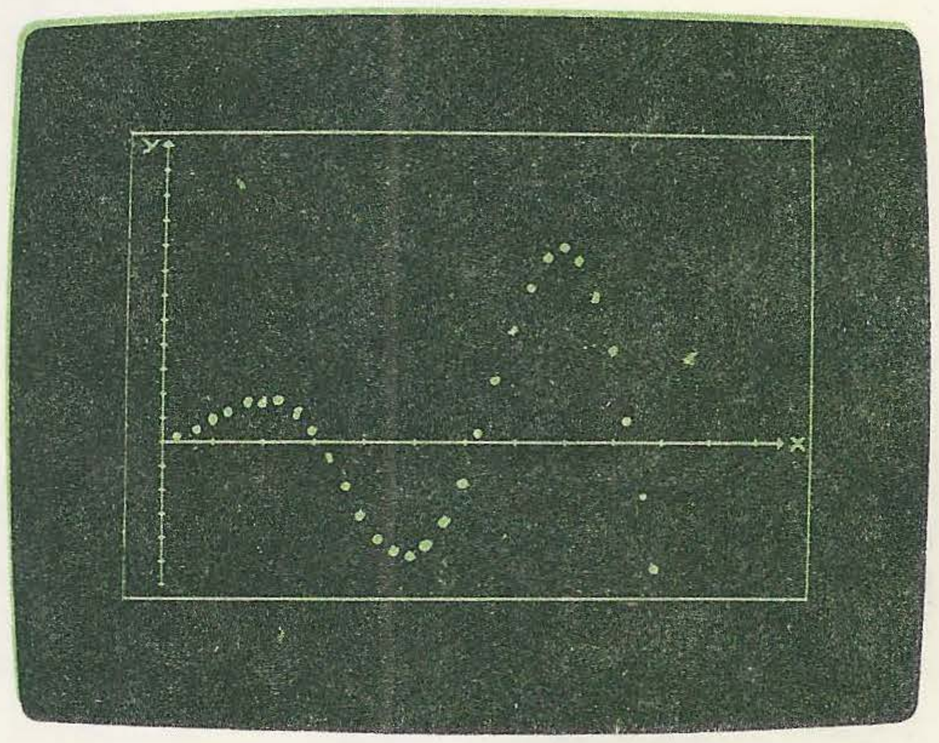
İstenen programın uygulaması ve başa alma bölümü (100'den 180.satıra) sona erdiğinde, alt-program 2000 (satır 220) çağrılır. Bu yıldızlarla çevrelenmiş bir çerçeve ve ters basılmış bir video başlığı sunar.

Alt-program 2000'in sonunda, önerilen değişik olasılıkları düzen içinde sunan ana programa geri dönülür. Fotoğraf, tüm sorulara yanıt verildikten sonra ve grafik türü seçilmeden önceki son durumu göstermektedir.

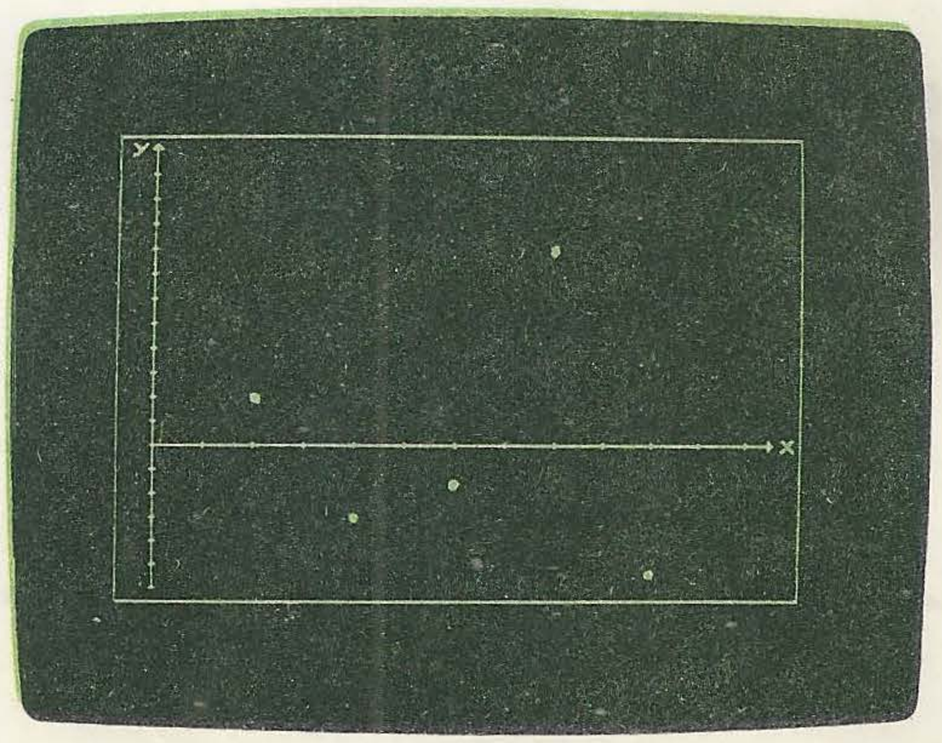
Giriş verileri X* SIN(X) fonksiyonuna (satır 100) ilişkindir. Sonraki fotoğraflar aynı fonksiyon için, NOKTA SAYISI ve GRAFİK TÜRÜ parametrelerine göre grafiğin farklı görünümlerini gösterir.



Seçilen biçim noktalardan oluşan grafik türüdür. Programda kartezyen eksenler ve 30 koordinat çiftine karşılık gelen noktalar bulunur. X'in değeri klavye ile girilmiş, Y ise program tarafından hesaplanmıştır.

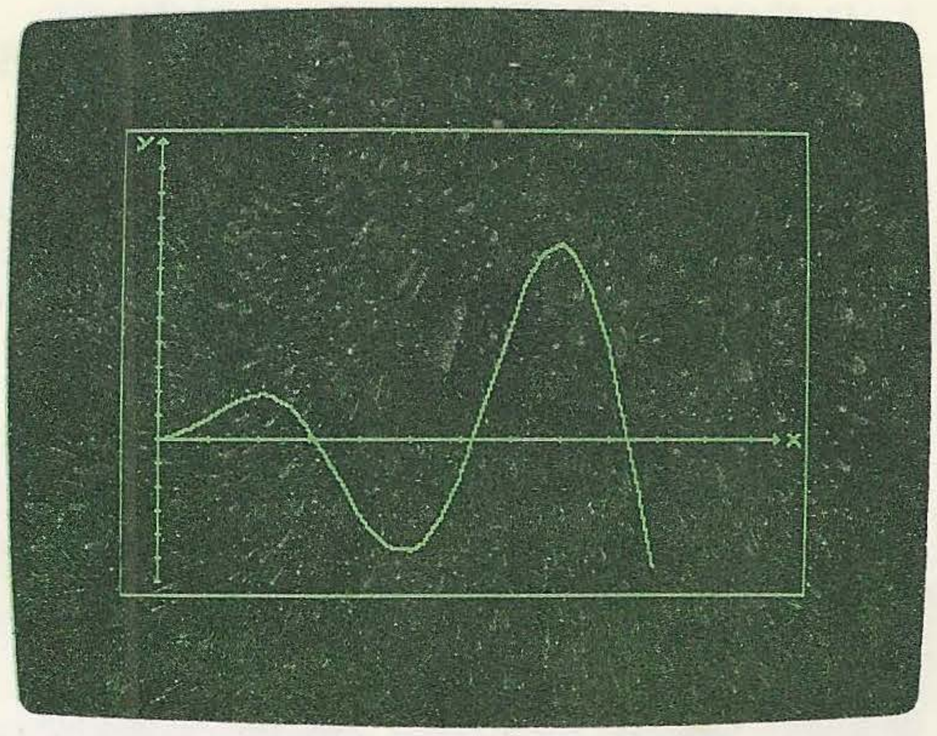


Bu durumda, kullanıcı daha az sayıda (5)X değerinden oluşan bir grafiğinin görselleştirilmesini istemiştir. Fonksiyonun eğrisi hep aynı, ancak daha az belirgindir.

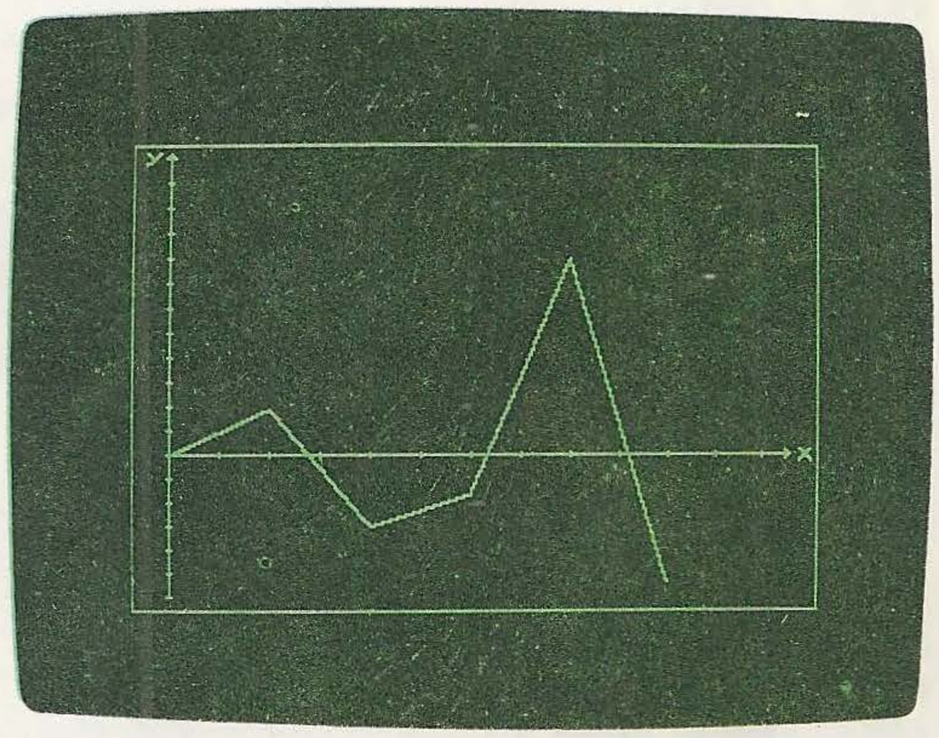


Ücretsiz PDF Kitap Arsivi - https://rantey.org

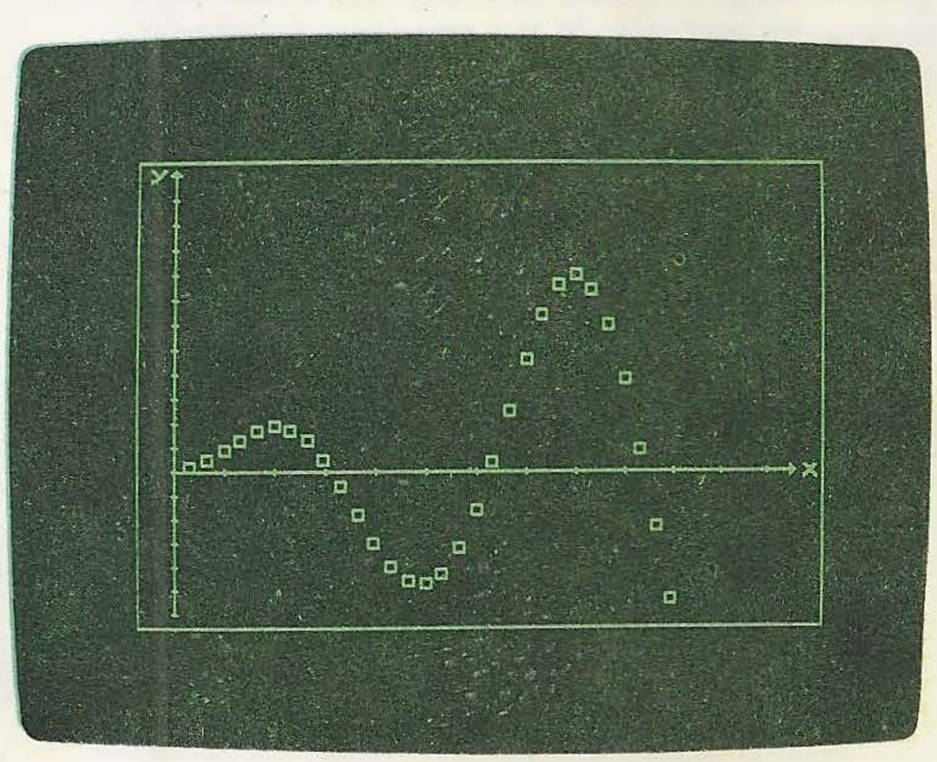
X değişkeninin 30 değerinin doğru parçalarıyla görselleştirilmesi. Verilerin çokluğundan ötürü eğrinin tanımı mükemmeldir. Program koordinatları bilinen noktaları ikişer ikişer birleştiren bir dizi doğru çizmiştir.



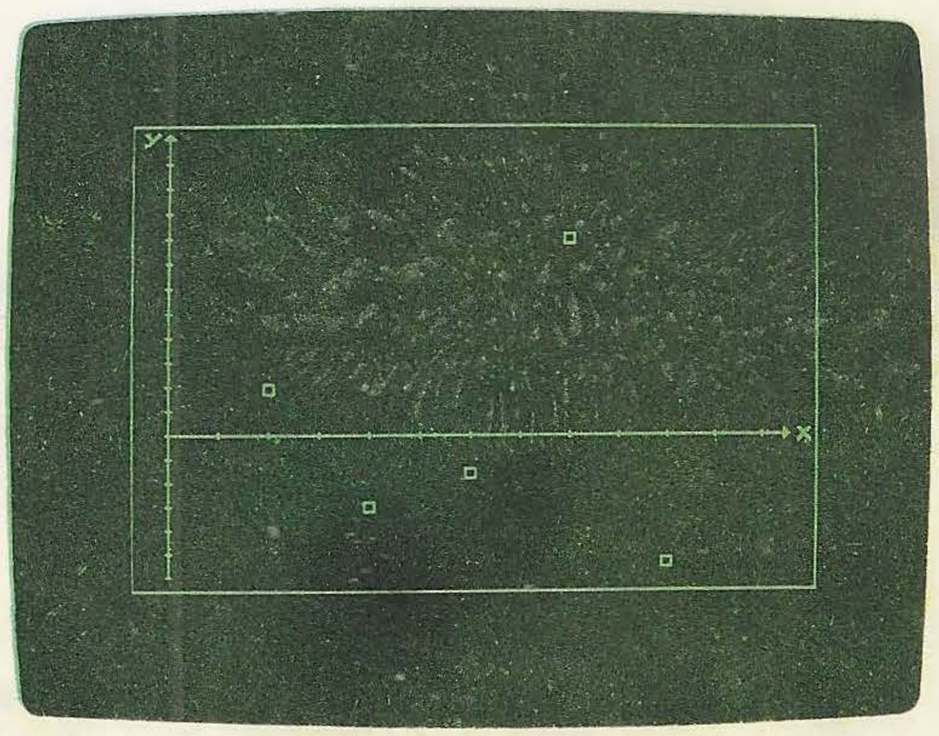
Görselleştirme yine doğru parçaları ile yapılmaktadır, ancak X'in 5 giriş değeri vardır. Grafiğin tanımı çok yaklaşık olarak yapılmıştır ve herbir doğru parçası bağımsız olarak görülür.



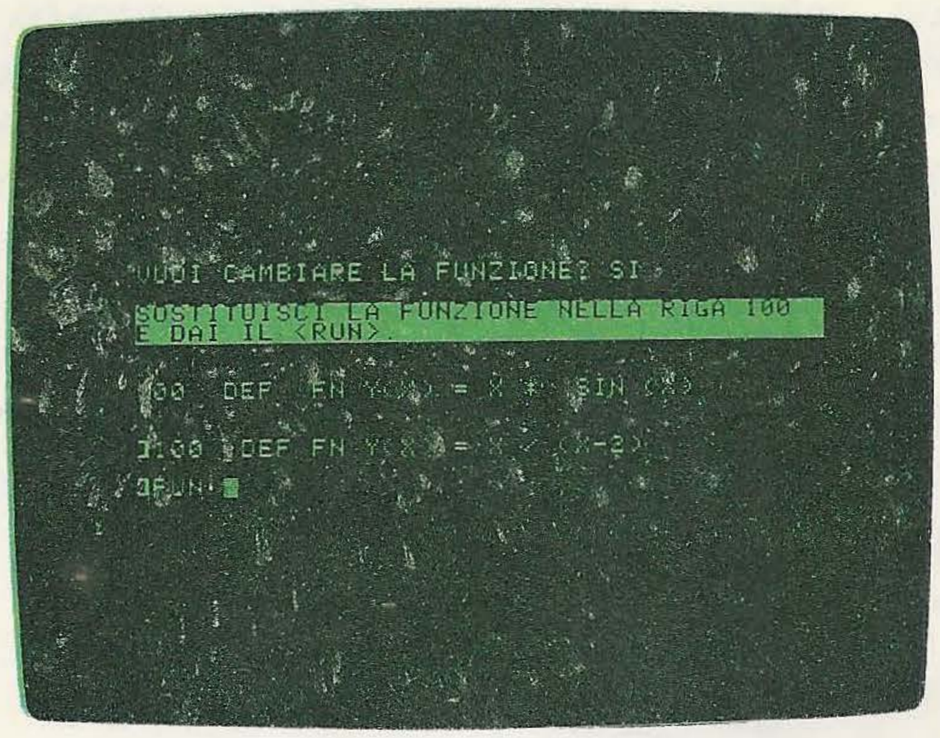
Kullanıcı, X'in 30 değerinin simgelerle görselleştirilmesini istemiştir. Kestirim yanlışlarının bulunduğu bir büyüklüğün biçimini görselleştirmek söz konusu olduğunda bu tasvir daha çok salık verilir.
Kullanılan simgenin büyüklüğü bu durumda bu yanlışları örtebilir.



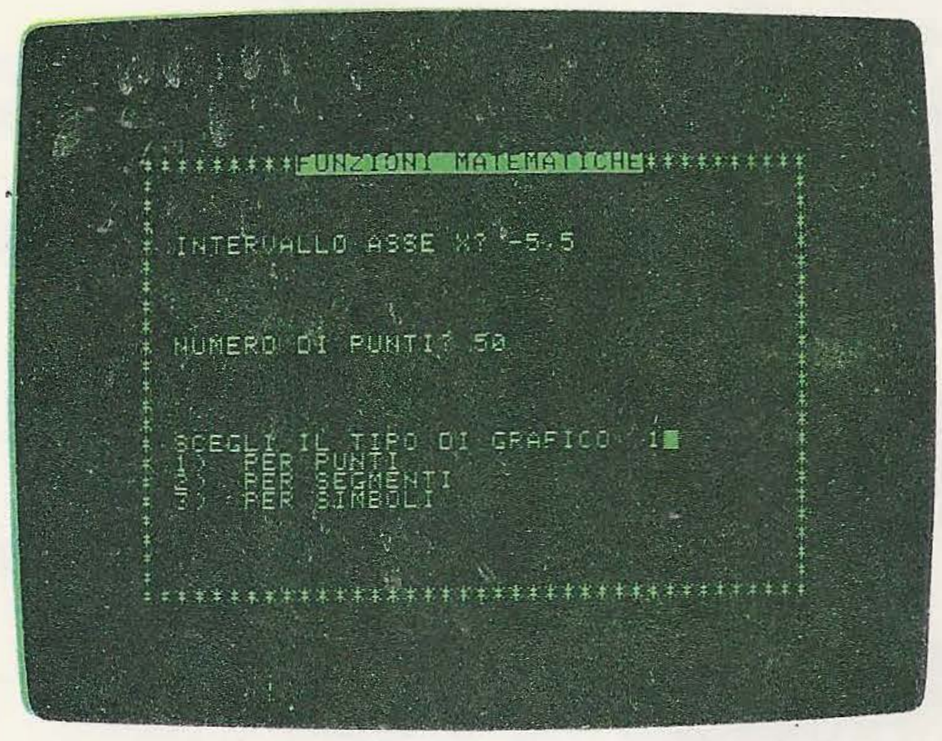
Bağımsız değişkenin 5 değerinin simgelerle görselleştirilmesi. Eğrinin gidişi çok zor seçilir.



Ücretsiz PDF Kitap Arsivi - https://rantey.org



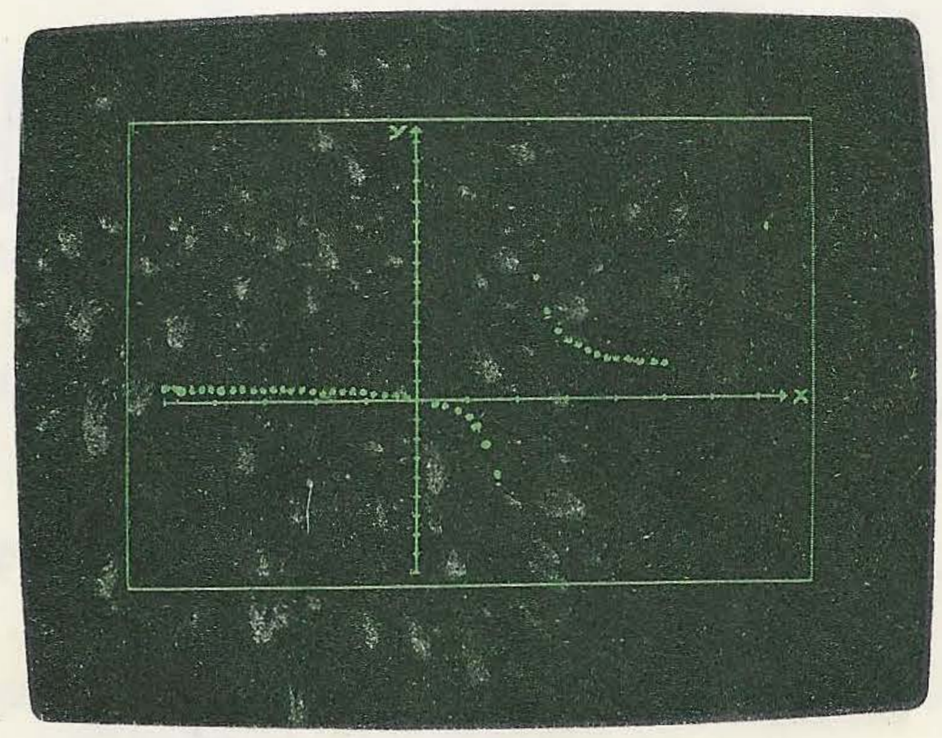
Grafik bir kez görselleştirildikten sonra, programın fonksiyonu harekete geçirmesi için herhangi bir tuşu harekete geçirmek yeter (satır 540). Satır 100'ün içeriğini göstermeye, ardından ele alınan fonksiyonu yeniden çağırmaya olanak veren satır 580'e dikkat. Satır 580'i programı sona erdiren END komutu izler. Bu komutu da işledikten sonra sistem kullanıcıdan komut kabul etmeye hazır hale gelir (bu durum klavuz olarak kullanan karakterinin görülmesiyle açığa çıkar) ve kullanıcı satır 100'ü yeniden yazarak fonksiyonu değiştirebilir. RUN komutu yeni X/(X-2) fonksiyonunun grafiğinin uygulamasını başlatır.



Program yeniden, kullanıcıdan uygulama için gerekli parametreleri sağlamasını istediği çerçeveyi gösterir.



Grafik türü seçildikten sonra program hesaplama evresine geçer. Burada fonksiyonun X=2 için bir süreksizlik noktası içerdiğini görür. Bu değer için elde Y=2/(2—2)=2/0 vardır. 2/0 kesiri grafikte gösterilemeyen bir sonsuz sayı verir. Program bu normal olmayan durumu bildirir (bu bir hata değil fonksiyonun bir özelliğidir). Ortaya çıkan tanı çok geneldir ve kimi kez matematik açısından doğru bile olmayabilir. Bu, bir çok durumu bir tek mesajda bir araya getiren yanlış bir alt-program yapısından kaynaklanır.



Kullanıcı herhangi bir tuşa basarak seçimini doğruladıktan sonra (satır 10250 ve sonrakiler) fonksiyon çizilir. X=2 için (fonksiyonun süreksizlik noktası) bir asimptotun bulunduğu belirlenmiştir.
Doğru parçalarıyla gösterilmesi istenerek eğrinin tanımı daha iyi hale getirilir.

Ücretsiz PDF Kitap Arsivi - https://rantey.org

Birbirine daha az benzeyen başka karakterler için, yalnızca doğru parçalarından oluşan bir şematizasyona başvurmak mümkündür. Bu yöntem 0'dan 9'a kadarki sayıların, her biri alfabenin bir harfiyle tanımlanan yedi doğruparçası (a,b,c,d,e,f,g) aracılığıyla görselleştirilmesinde çok kullanılır (sayfa 1485'teki şekil). Bu doğru parçalarının seçmeli olarak harekete geçirilmesi 10 sayıdan birini görselleştirir: Tüm doğru parçaları görünür olduğunda 8 elde edilir; g doğru parçası söndürülerek 0 elde edilir; f,a,b ve c doğru parçalarını da kaldırırsak 1 ortaya çıkar vb.

Sayı gösterme sistemlerindeki doğru parçaları elektrikle aydınlanan tüpler ya da sıvı kristallerdir.

Sayfa 1485'teki tabloda ilk kolon 1'den 0'a sayıları, ikincisi harekete geçirilecek parçalarını, üçüncüsü de görüntülenen simgeyi (sonuç) göstermektedir.

Son kolon ise ikincisinin aynıdır; yalnızca EVET/HAYIR işareti yerine bellekte daha az yer tutan ve seçmeleri daha hızlı gerçekleştiren ikili 0/1 işaretine göre düzenlenmiştir. 6 elde etmek için g,f,e,b ve a doğru parçalarını aktive etmek gerekir. Bunların herbiri programda 1 (EVET) ile, ötekiler 0 ile gösterilmiştir. Buna göre 6, 1 1 1 0 0 1 1 şeklinde ifade edilir.

Program, tablonun içeriğini konum konum gözden geçirir; 1 sayısına rasladığında, yedi harekete geçirme alt-programından (doğru parçlarına karşılık gelenini) çağırır.

Aslında yedi ayrı alt-programın bulunması gerekmez, doğru olarak parametrelenmiş bir tek alt-program yeterlidir.

Tablo 1'in ikinci kolonu doğru parçalarının herbirini elde etmek için başlangıç noktasına (X0,Y0) göre yapılması gereken yer değiştirmeleri gösterir. Böylece doğru parçası X0, Y0'ın X0,Y0 + S (S her parçanın uzunluğudur) doğrultusunda yer değiştirmesiyle çizilir. Bu sistem (S: birim yer değiştirme) iki önemli parametrelemeye olanak verir.

- Tüm yer değiştirmeler (buna bağlı olarak da bütün doğru parçaları) mutlak koordinatlarla değil, bir başlangıç noktasına göre belirlenmiş koordinatlarla belirlenir. Karakterin görselleşme konumunun yerini değiştirmek için başlangıç noktasının koordinatlarını değiştirmek yeter.
- Karakterlerin boyutları S parametresi sayesinde kolayca değişebilir.

Örneğin a doğru parçası sözkonusu olduğunda, ilk konum başlangıç noktasına göre hiç bir değişmeye uğramamıştır. Başka bir deyişle, artma Y gibi X ekseninde de 0'dır. Buna karşılık, doğru parçasının son koordinatlarının (X0 ve Y0 + S) artması Y ekseninde S kadardır. (X'de ise 0). Bu parametreleme tüm doğru parçalarını yönetebilen genelleştirilmiş bir programın yazılmasına olanak verir.

Eğer bir başlangıç noktası, (X1,Y1) öbür ucu da (X2,Y2) ise

$$X1 = X0 \pm N_X S \text{ (X ekseninde artma)}$$
$$X1 = X0 \pm N_Y S \text{ (Y ekseninde artma)}$$

olur.

X2,Y2 koordinatları için de aynı şey geçerlidir (X0 ve Y0'ın referans koordinatları olduğunu unutmayalım). Bu özel durumda ilk artmalar için N = 0 (başlangıç noktası)'dır, yani;

$$X1 = X0$$
$$Y1 = Y0$$

Buna karşılık, ikinci nokta için NX = 0 ve NY = 1'dir, buradan çıkarak;

$$X2 = X0$$
$$Y2 = Y0 \pm S$$

X1,Y1 ve X2,Y2 referans olarak alınan başlangıç noktasına (X0, Y0) göre uçların koordinatlarıdır. İkinci ifadede kullanılacak işaret (+ ya da —) Y ekseninin yönüne bağlıdır. Ekranın başlangıç noktasının altta bulunduğu terminallerde işaret + 'dır (Y'ler yukarı doğru artar); bunun tersi de geçerli olabilir.

Genelleştirilmiş bir program elde etmek için her doğru parçasına ilişkin artmaları ve her sayıyı oluşturan doğru parçalarının (sayfa 1486'daki tablolar) belleğe alınması yeterlidir.

Sayfa 1487'teki akış şeması, (X0, Y0 başlangıç noktası değiştirilerek) ekranın herhangi bir yerine yerleştirilmiş bir sayının (S parametresi sayesinde) istenen boyutlarda görselleştirilmesi programını göstermektedir.

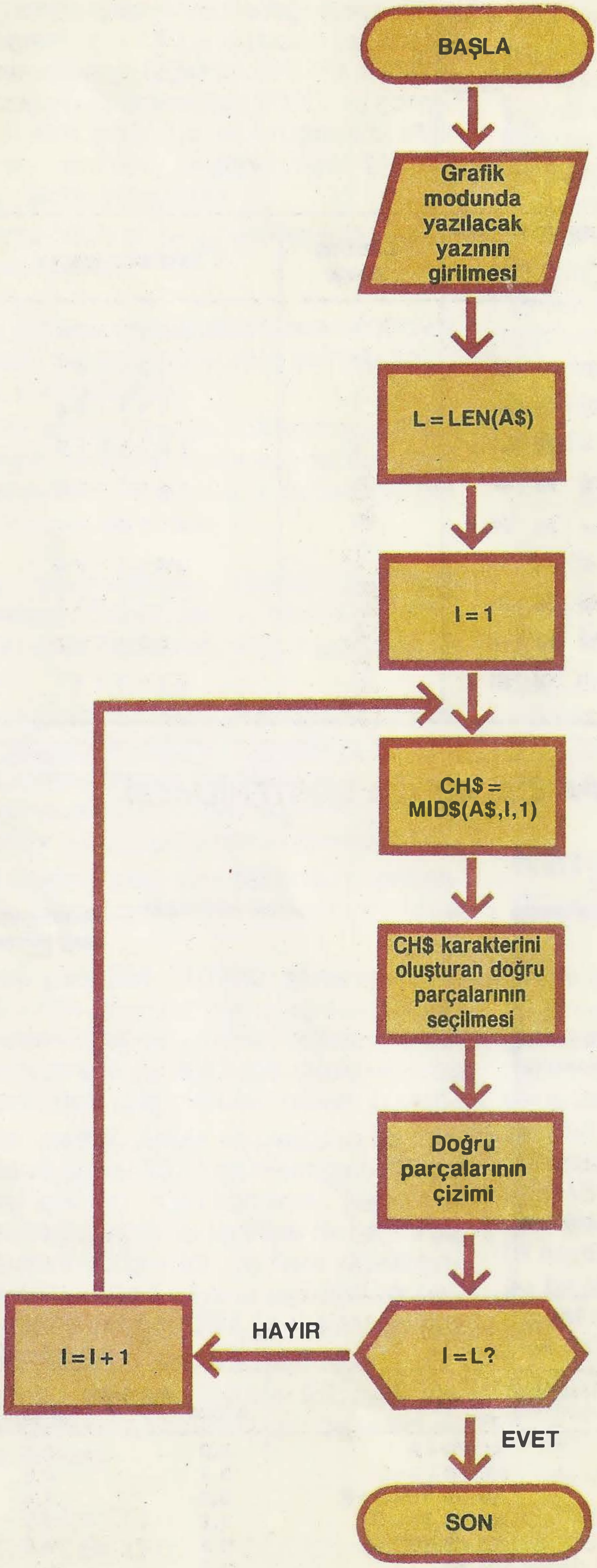
Bir doğru parçası elde etmek için gerekli olan artmalar 4 tabloda belleğe alınır:

XD(1),YD(1) = Doğru parçasının giriş noktası koordinatları

XA(1),YA(1) = Doğru parçasının bitiş noktası koordinatları

Ücretsiz PDF Kitap Arsivi - https://rantey.org

KARAKTERLERİN GRAFİK MODUNDA GÖSTERİLMESİ



Yazılacak karakterler girilmiş (klavyeyle) ve A$ zincirine yerleştirilmiştir.

Zincir uzunluğunun belirlenmesi (karaker sayısı olarak)

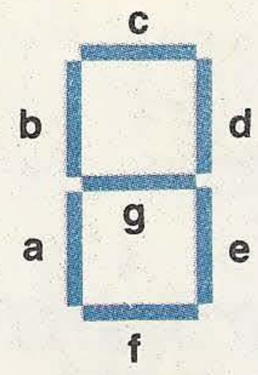
I konumundaki karakterin A$ zincirinden çıkarılması, I 1 ile A$'nin uzunluğu olan L arasında değişir.

CH$ karakterinin grafik modunda görselleştirilmesi

Zincirin sonunda kontrol

Ücretsiz PDF Kitap Arsivi - https://rantey.org

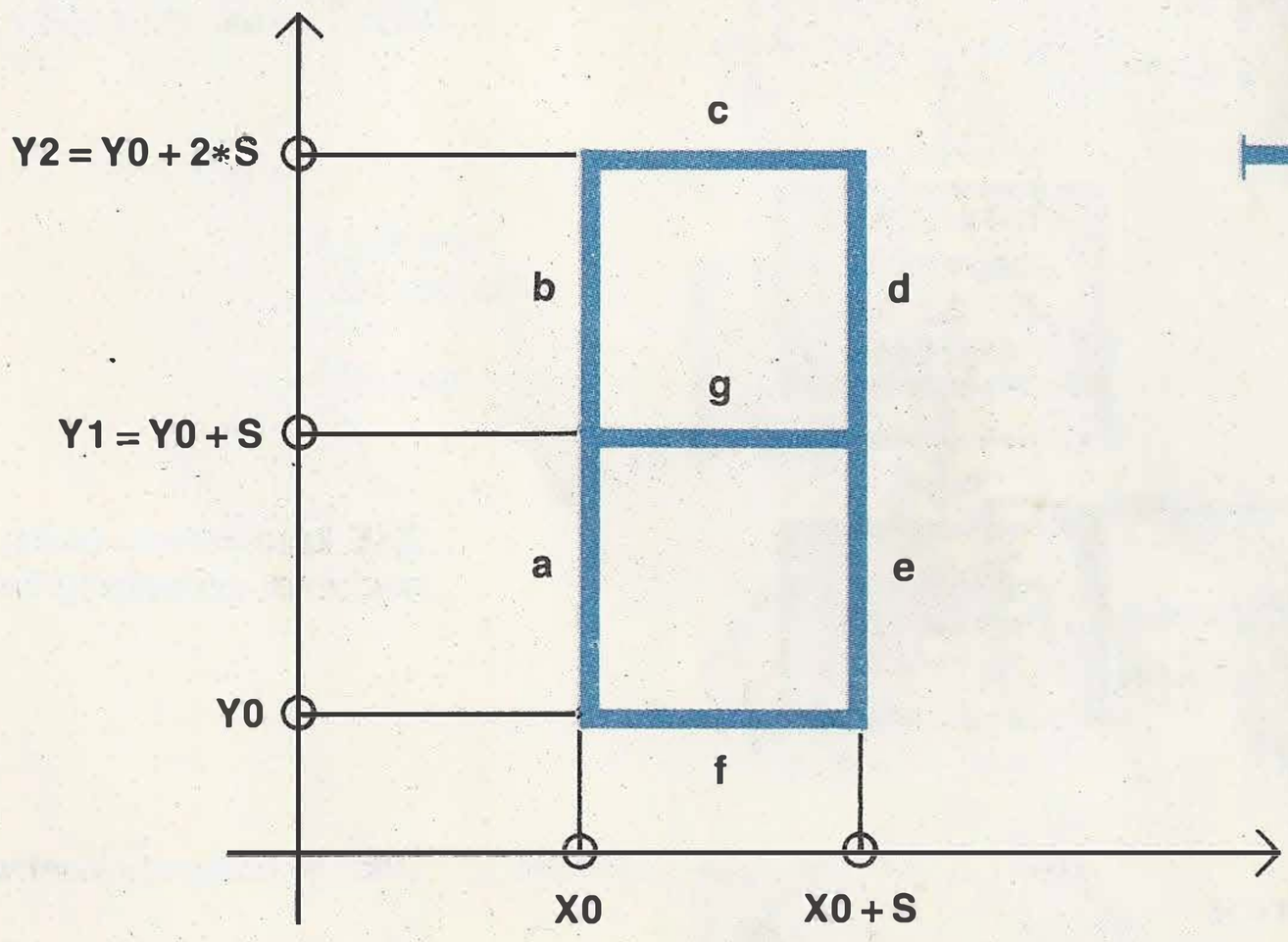
SAYILARIN DOĞRU PARÇALARIYLA GÖSTERİLMESİ



SI = Harekete geçirilmiş

Gösterilecek sayı	Harekete geçirilmiş doğru parçası 7 g	6 f	5 e	4 d	3 c	2 b	1 a	Çizilmiş simge	Tablonun içeriği
1	—	—	SI	SI	—	—	—	1	0 0 1 1 0 0 0
2	SI	SI	—	SI	SI	—	SI	2	1 1 0 1 1 0 1
3	SI	SI	SI	SI	SI	—	—	3	1 1 1 1 1 0 0
4	SI	—	SI	—	—	SI	—	4	1 0 1 0 0 1 0
5	SI	SI	SI	—	SI	SI	—	5	1 1 1 0 1 1 0
6	SI	SI	SI	—	—	SI	SI	6	1 1 1 0 0 1 1
7	—	—	SI	SI	SI	—	—	7	0 0 1 1 1 0 0
8	SI	SI	SI	SI	SI	SI	SI	8	1 1 1 1 1 1 1
9	SI	—	SI	SI	SI	SI	—	9	1 0 1 1 1 1 0
0	—	SI	SI	SI	SI	SI	SI	0	0 1 1 1 1 1 1

SAYININ DOĞRU PARÇALARIYLA GÖSTERİLMESİ

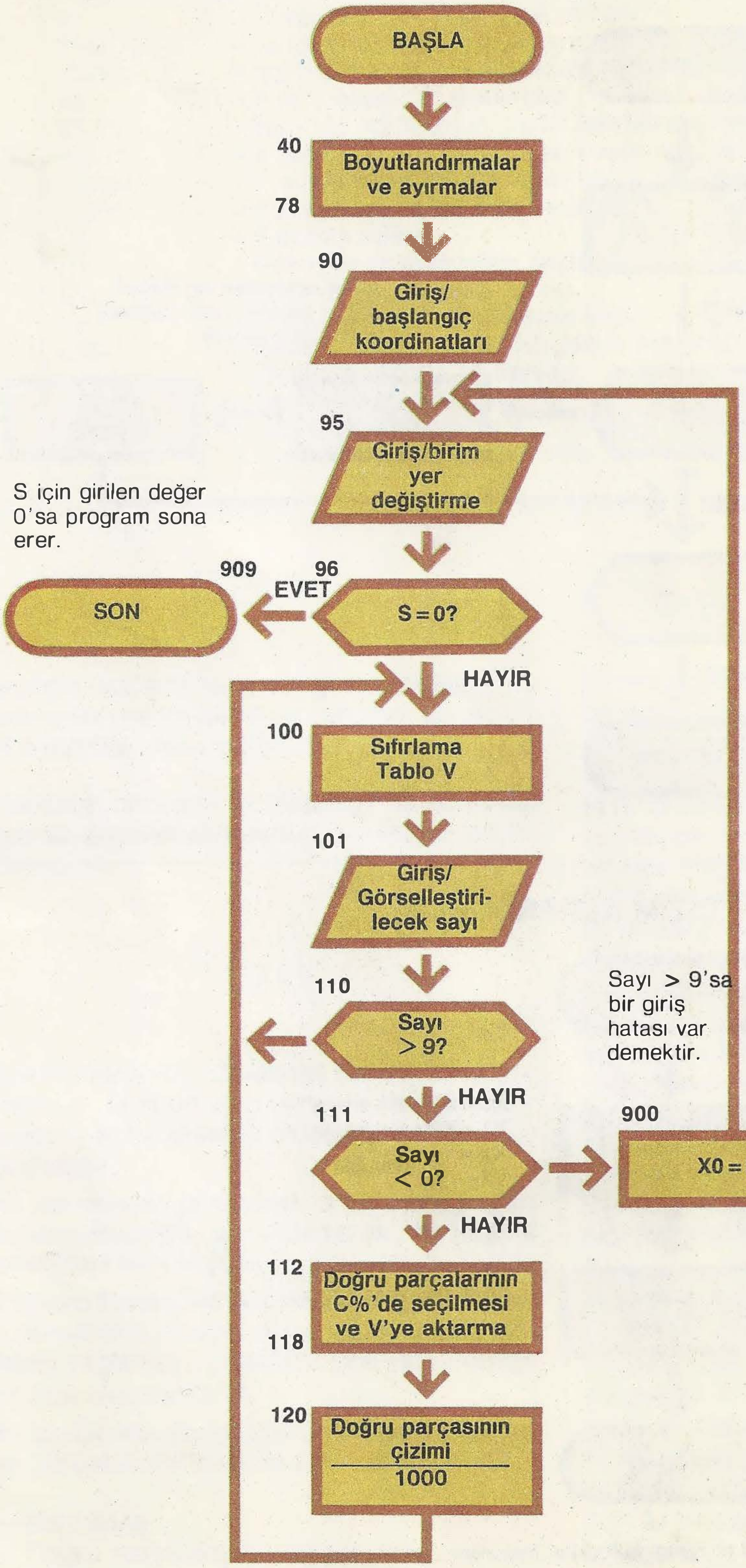


S = Birimi yer değiştirme

Doğru parçası	Koordinat a'dan	Koordinat b'ye	Artmalar a'dan	Artmalar b'ye
a	X0,Y0	X0,Y0 + S	0,0	0,1
b	X0,Y0 + S	X0,Y0 + 2 * S	0,1	0,2
c	X0,Y0 + 2 * S	X0 + S,Y0 + 2 * S	0,2	1,2
d	X0 + S,Y0 + 2 * S	X0 + S,Y0 + S	1,2	1,1
e	X0 + S,Y0 + S	X0 + S,Y0	1,1	1,0
f	X0 + S,Y0	X0,Y0	1,0	0,0
g	X0,Y0 + S	X0 + S,Y0 + S	0,1	1,1

Ücretsiz PDF Kitap Arsivi - https://rantey.org

BİR SAYININ GRAFİK MODUNUN GÖRSELLEŞTİRİLMESİ



S için girilen değer 0'sa program sona erer.

Sayı > 9'sa bir giriş hatası var demektir.

S = Birim yer değiştirme
C% = Sayılara göre yer değiştirme tablosu

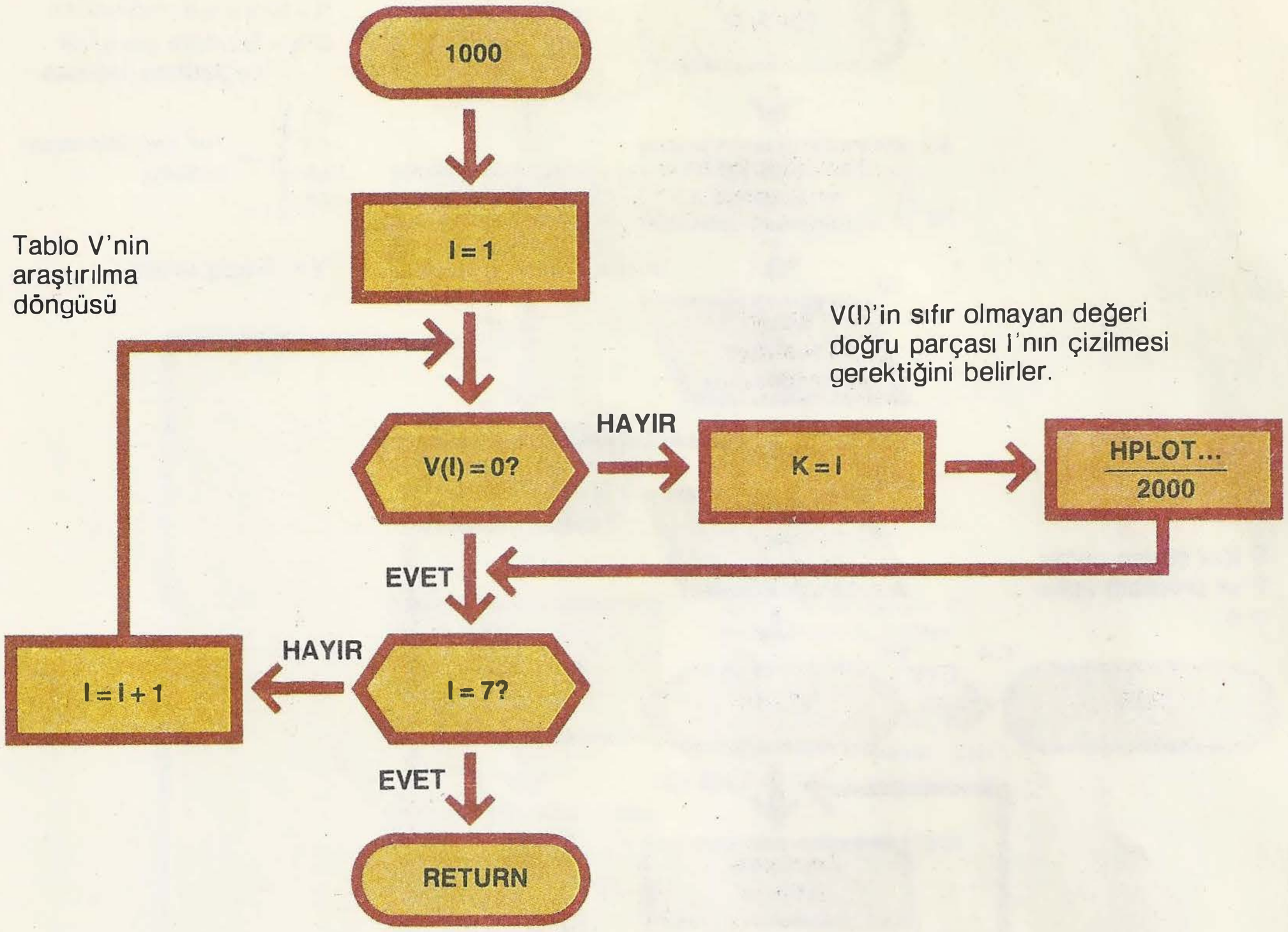
XD
YD
XA
YA
} = Yer değiştirmeler tablosu

V = Geçiş tablosu

Negatif bir sayının girişi, gösterme bölgesinin yerinin (örn. XO'yu değiştirerek) değiştirilmesinin gerektiği anlamına gelir.

Ücretsiz PDF Kitap Arsivi - https://rantey.org

BİR DOĞRU PARÇASININ GÖRSELLEŞTİRİLMESİ ALT-PROGRAMI



Tablo V'nin araştırılma döngüsü

V(I)'in sıfır olmayan değeri doğru parçası I'nın çizilmesi gerektiğini belirler.

ÇİZİM ALT-PROGRAMI



X1 = XO + S x artma
Y1 = YO ± S x artma
X2 = } aynı ilke
Y2 =

HPLOT X1,Y1 TO X2,Y2
LINE (X1,Y1)—(X2,Y2)

Bu alt-program kullanılan makinenin türüne bağlıdır.

Ücretsiz PDF Kitap Arsivi - https://rantey.org

Değerler 910-916. satırlardaki DATA komutlarının yardımıyla 60. satıra sokulur (verileri sayfa 1484'teki tabloyla karşılaştırınız).

On sayının görselleştirilmesi için gereken doğru parçaları, haneleri READ komutu (satır 76) ve DATA komutları (satır 918-927) aracılığıyla doldurulmuş olan iki boyutlu C% (10, 7) tablosuna (dizeyine) yerleştirilmiştir.

İlk boyut (10) gösterilebilir on sayıdan (0,1, 2...) birini belirtir, ikincisi ise (7) harekete geçirilecek doğru parçalarını gösterir.

Program, gösterilecek doğru parçalarına karşılık gelen C% elemanlarının (satır 116) aktarıldığı bir geçiş tablosundan (V, 7 boyutunda) yararlanır; bu durumda kontrol edilen de bu tablonun içeriğidir (alt-program 1000, 1488. sayfada alttaki şekil).

Alt-program 1000 (şekil 1, sayfa 1488), doğru parçasının başlangıç (X1,Y1) ve bitiş (X2, Y2) konumlarını hesaplayan ve ardından doğru parçasını çizen alt-program 2000'i (şekil 2) çağırır.

Apple ve Siprel 2010'dan farklı makineler için 3000 alt-programını (aşağıdaki program) değiştirmek yeterlidir. Olivetti M20 için ise diğer satırlar aynı kalmak üzere, 3010, 3030 ve 3040. satırlar değişerek;

```
3010 Y1 = Y0 + YD(K)*S
3030 Y2 = Y0 + YA(K)*S
3040 LINE (X1,Y1)—(X2,Y2)
```

olurlar.

Her harfi bir resim gibi gösteren bir alt-prog-

GRAFİK MOD İLE SAYILARIN GÖRÜNTÜLENMESİ

```
10      REM
20      REM   SAYILAR
30      REM
40      DIM XD(7),YD(7),XA(7),YA(7)
45      DIM V(7),C%(10,7)
46      HGR
50      FOR I = 1 TO 7
60      READ XD(I),YD(I),XA(I),YA(I)
70      NEXT I
72      FOR I = 1 TO 10
74      FOR J = 7 TO 1 STEP -1
76      READ C%(I,J)
      : NEXT J
78      NEXT I
90      PRINT "BASLANGIC KOORDINATLARI"
91      INPUT "XO = ?";XO
92      INPUT "YO = ?";YO
93      SX = XO
95      INPUT "BIRLESTIRICI YER DEGISTIRME? ";S

96      IF S = 0 GOTO 909
97      YO = YO + 3 * S
100     FOR I = 1 TO 7
      : V(I) = 0
      : NEXT I
101     INPUT " NUMARA ?";N
1.0     IF N > 9 GOTO 100
111     IF N < 0 GOTO 900
112     L% = N
      : IF L% = 0 THEN L% = 10
114     FOR I = 1 TO 7
116     V(I) = C%(L%,I)
118     NEXT I
120     GOSUB 1000
160     XO = YO + 2 * S
170     GOTO 100
900     XO = SX
902     GOTO 95
909     END
```

```
910   DATA 0,0,0,1
      : REM = a
911   DATA 0,1,0,2
      : REM = b
912   DATA 0,2,1,2
      : REM = c
913   DATA 1,2,1,1
      : REM = d
914   DATA 1,1,1,0
      : REM = e
915   DATA 1,0,0,0
      : REM = f
916   DATA 0,1,1,1
      : REM = g
918   DATA 0,0,1,1,0,0,0
      : REM = 1
919   DATA 1,1,0,1,1,0,1
      : REM = 2
920   DATA 1,1,1,1,1,0,0
      : REM = 3
921   DATA 1,0,1,0,0,1,0
      : REM = 4
922   DATA 1,1,1,0,1,1,0
      : REM = 5
923   DATA 1,1,1,0,0,1,1
      : REM = 6
924   DATA 0,0,1,1,1,0,0
      : REM = 7
925   DATA 1,1,1,1,1,1,1
      : REM = 8
926   DATA 1,0,1,1,1,1,0
      : REM = 9
927   DATA 0,1,1,1,1,1,1
      : REM = 0
1000  FOR I = 1 TO 7
1020  IF V(I) = 0 GOTO 1040
1025  K = I
1030  GOSUB 3000
1040  NEXT I
1050  RETURN
3000  X1 = XO + XD(K) * S
3010  Y1 = YO - YD(K) * S
3020  X2 = XO + XA(K) * S
3030  Y2 = YO - YA(K) * S
3040  HPLOT X1,Y1 TO X2,Y2
3050  RETURN
```

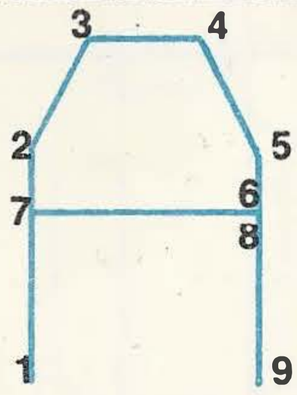
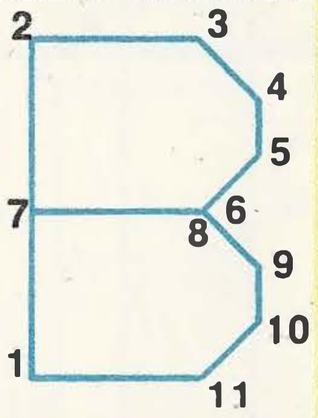
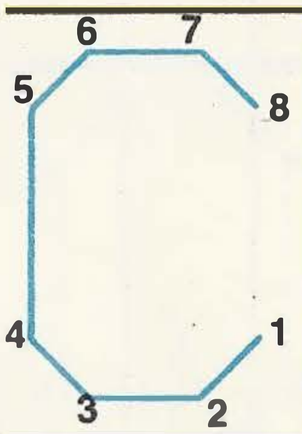
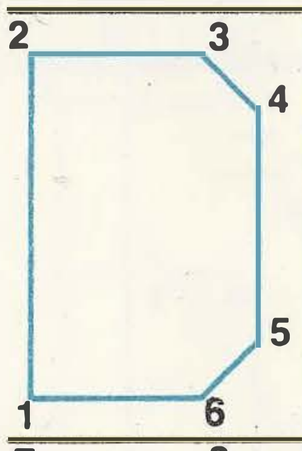
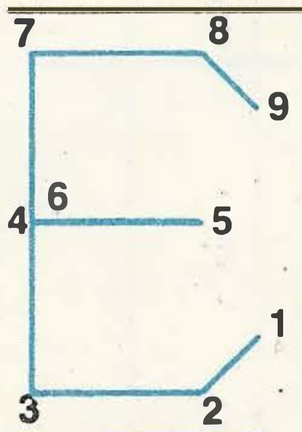
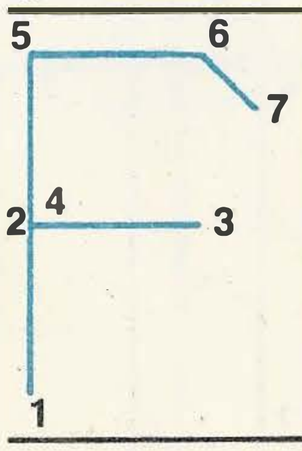
ram sayesinde, alfabe karakterlerinin görüntülenmesi sorunu da aynı şekilde kolaylıkla hallolur. Demek ki tıpkı rakamlarda yapıldığı gibi alfabe karakterlerinin de bir dizi doğru parçası ile şematize edilmesi sözkonusudur. Ancak bu benzerlik yalnızca şekle ilişkindir. Rakamlar için, farklı bileşimlerinin harekete geçirilmesiyle, bütün sayıları oluşturmaya yarayan 7 tane doğru parçası yeterlidir. Harfler için böyle bir işlem daha çok sayıda temel doğru parçasını gerektirecektir. Demek ki herhangi bir türden şekli tasarlamak için başka bir yöntem kullanılmalıdır.

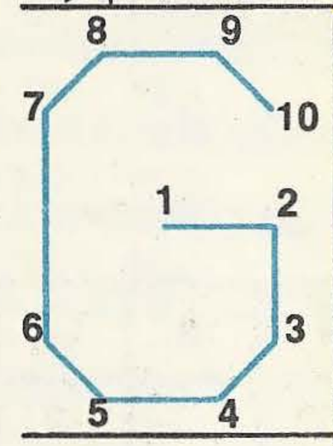
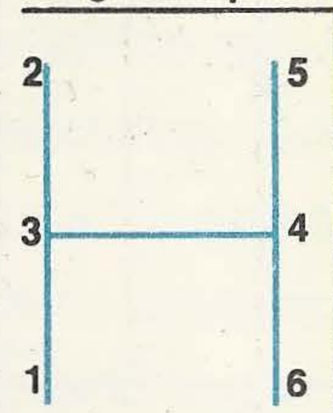
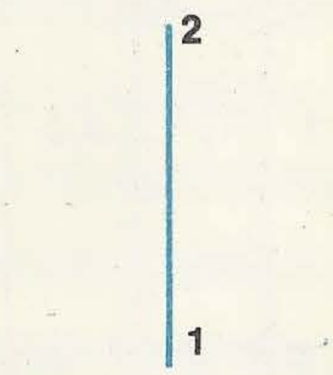
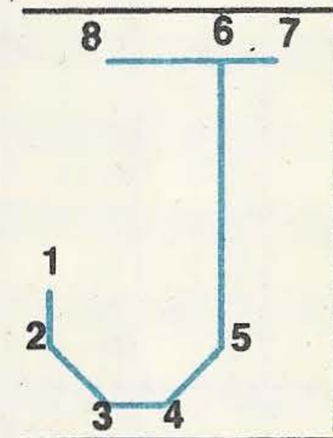
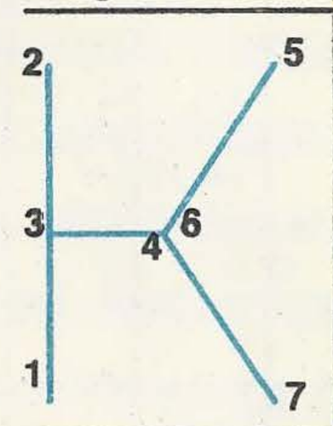
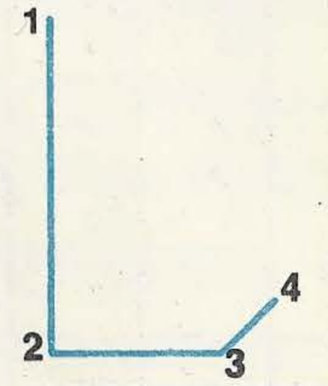
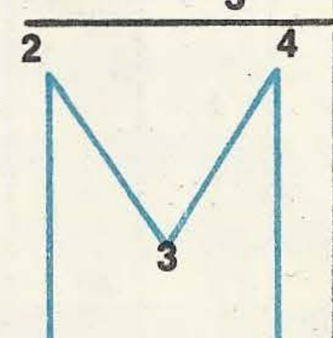
Her harfin bir dizi doğru parçası ile şematize edeceğini biliyoruz. Yapmakta olduğumuz uygulamada doğru parçası sayısı en fazla 11 olabilir. Harfin görüntülenmesi ise başlangıç yeri olarak seçilen noktanın üzerine elektronik fırçanın yerleştirilmesi, sonra da gereken doğru parçalarının çizilmesiyle sağlanır.

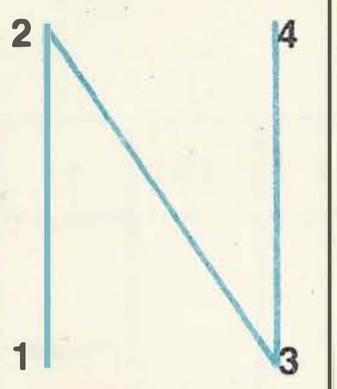
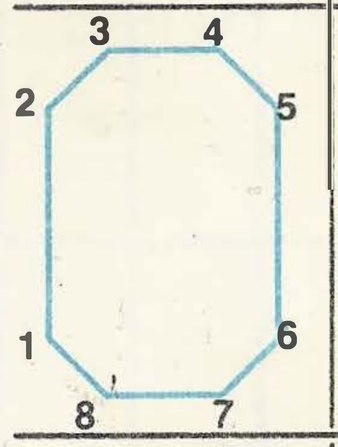
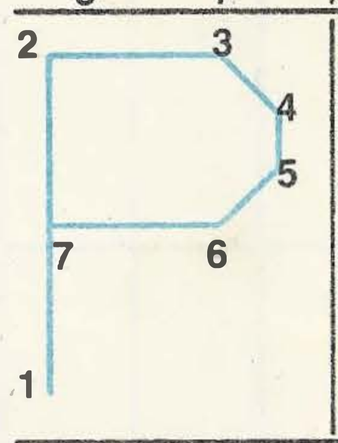
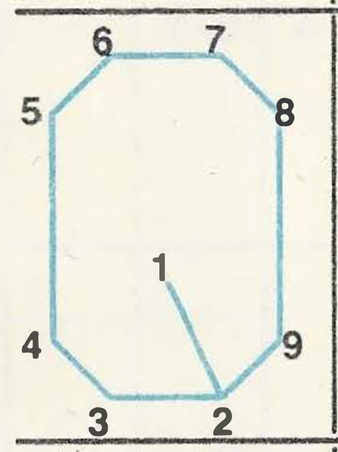
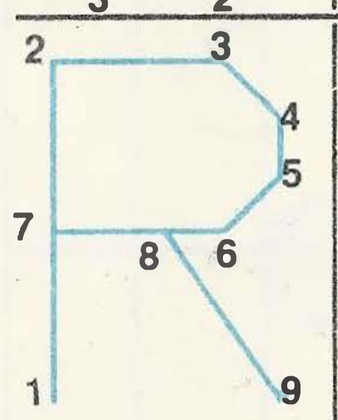
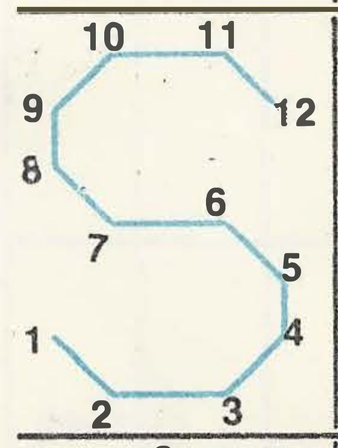
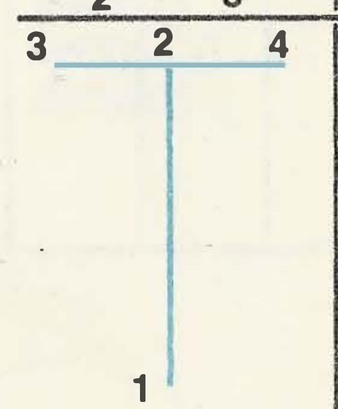
Sayfa 1491-1494'teki tablolarda bir alfabe karakterinin görüntülenmesini sağlayan yerdeğişimleri bulunmaktadır. Her harf, tabanı 4 yüksekliği 6 olan bir dikdörtgenin içindedir. Çeşitli konumlandırmalar için referans nokta-

GRAFİK MODDA ALFABETİK KARAKTERLERİN GÖRÜNTÜLENMESİ

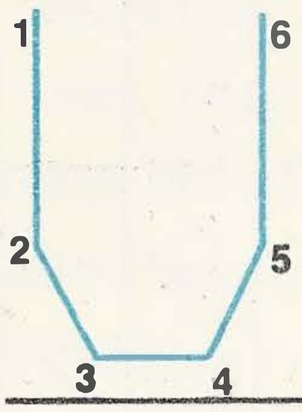
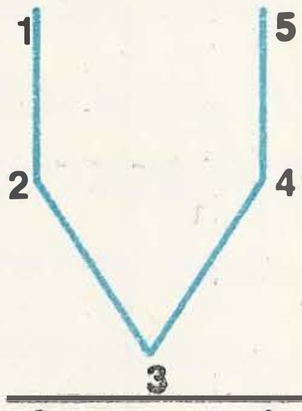
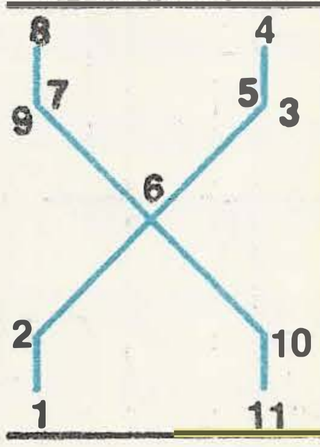
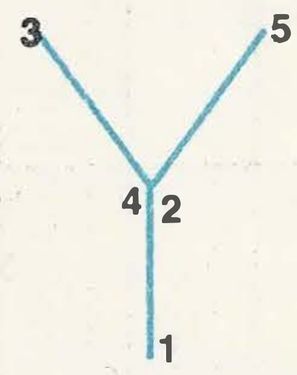
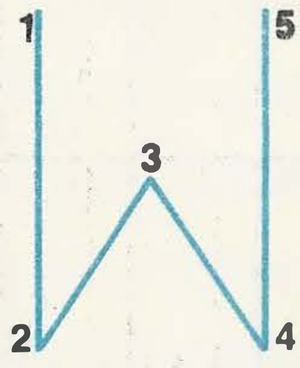
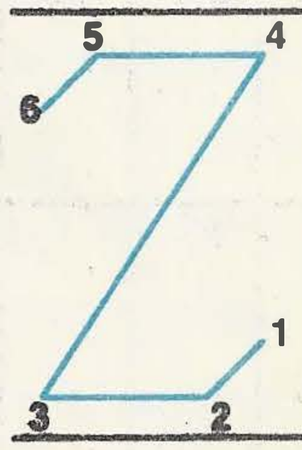


Harf	Konum	Yer değiştirmeler 1	2	3	4	5	6	7	8	9	10	11
	0,0	0,4	1,2	2,0	1,-2	0,-1	-4,0	4,0	0,-3	-	-	-
	0,0	0,6	3,0	1,-1	0,-1	-1,-1	-3,0	3,0	1,-1	0,-1	-1,-1	-3,0
	4,1	-1,-1	-2,0	-1,1	0,4	1,1	2,0	1,-1	-	-	-	-
	0,0	0,6	3,0	1,-1	0,-4	-1,-1	-3,0	-	-	-	-	-
	4,1	-1,-1	-3,0	0,+3	3,0	-3,0	0,3	3,0	1,-1	-	-	-
	0,0	0,3	3,0	-3,0	0,3	3,0	1,-1	-	-	-	-	-

	1	2	3	4	5	6	7	8	9	10	11
	2,3	2,0	0,-2	-1,-1	-2,0	-1,1	0,4	1,1	2,0	1,-1	-	-
	0,0	0,6	0,-3	4,0	0,3	0,-6	-	-	-	-	-	-
	2,0	0,6	-	-	-	-	-	-	-	-	-	-
	0,2	0,-1	1,-1	1,0	1,1	0,5	1,0	-3,0	-	-	-	-
	0,0	0,6	0,-3	2,0	2,3	-2,-3	2,-3	-	-	-	-	-
	0,6	0,-6	3,0	1,1	-	-	-	-	-	-	-	-
	0,0	0,6	2,-3	2,3	0,-6	-	-	-	-	-	-	-

	1	2	3	4	5	6	7	8	9	10	11
	0,0	0,6	4,-6	0,6	-	-	-	-	-	-	-	-
	0,1	0,4	1,1	2,0	1,-1	0,-4	-1,-1	-2,0	-1,1	-	-	-
	0,0	0,6	3,0	1,-1	0,-1	-1,-1	-3,0	-	-	-	-	-
	2,2	1,-2	-2,0	-1,1	0,4	1,1	2,0	1,-1	0,-4	-1,-1	-	-
	0,0	0,6	3,0	1,-1	0,-1	-1,-1	-3,0	2,0	2,-3	-	-	-
	0,1	1,-1	2,0	1,1	0,1	-1,1	-2,0	-1,1	0,1	1,1	2,0	1,-1
	2,0	0,6	-2,0	4,0	-	-	-	-	-	-	-	-



	1	2	3	4	5	6	7	8	9	10	11
	0,6	0,-4	1,-2	2,0	1,2	0,4	-	-	-	-	-	-
	0,6	0,-3	2,-3	2,3	0,3	-	-	-	-	-	-	-
	0,0	0,1	4,4	0,1	0,-1	-2,-2	-2,2	0,1	0,-1	4,-4	0,-1	-
	2,0	0,3	-2,3	2,-3	2,3	-		-	-	-	-	-
	0,6	0,-6	2,3	2,-3	0,6	-	-	-	-	-	-	-
	4,1	-1,-1	-3,0	4,6	-3,0	-1,-1	-	-	-	-	-	-

GRAFİK MODDA METİNLERİN GÖRÜNTÜLENMESİ İÇİN PROGRAMIN AKIŞ ŞEMASI



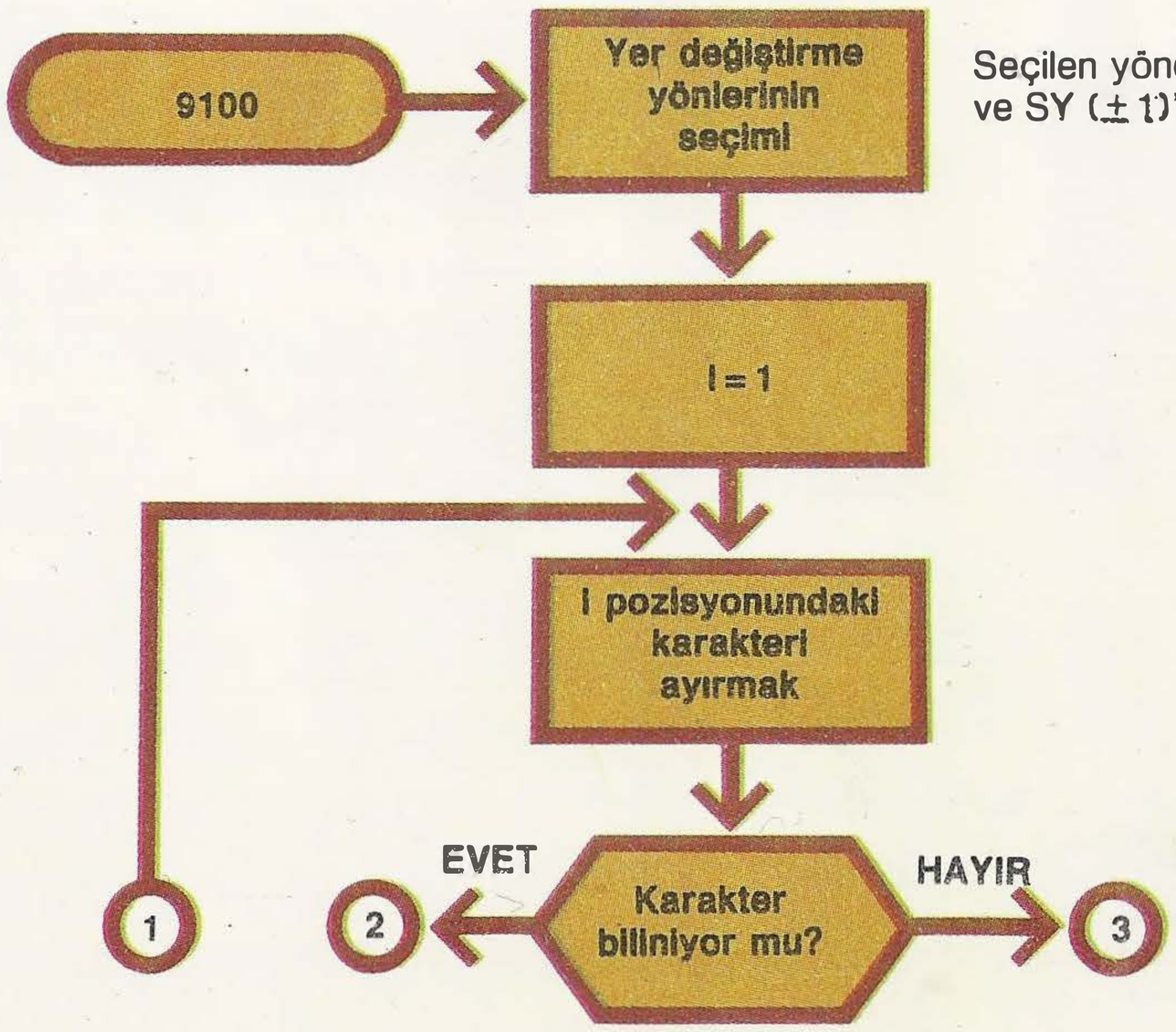
Yeni metin döngüsü

A$ grafik mod ile görüntülenecek olan metindir

X0,Y0 = Zincirin başlangıç noktaları
S = Ölçek
D = Alan
MOV = Yönlendirme

Yalnızca eğitici bir amaç taşıyan bu programda giriş birimi öngörülmemiştir.

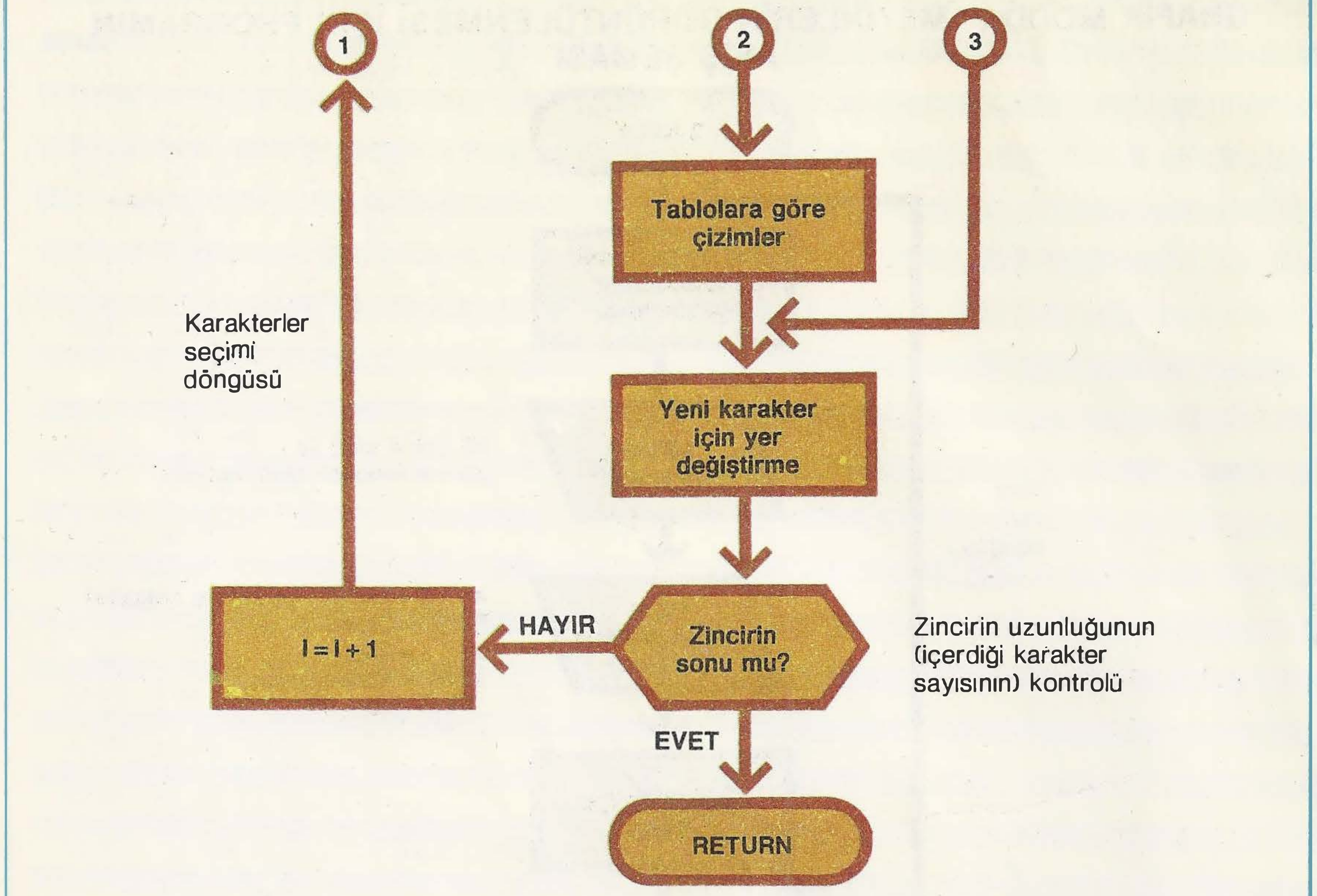
KARAKTERLERİN GÖRÜNTÜLENMESİ ALT-PROGRAMI



Seçilen yöne bağlı olarak SX ve SY (±1)'nin seçimi

Karakter seçimi döngüsü

ASCII değerlerine göre tabloda öngörülmemiş son karakterin atlanması



İŞARETLERİN GRAFİK MOD İLE GÖRÜNTÜLENMESİ

```
10      HGR
      : HCOLOR= 3
20      DIM PO%(26,2),CAR%(26,11,2)
30      GOSUB 350
40      HOME
50      VTAB 22
60      INPUT A$
70      INPUT XO,YO,S,D,MOV
80      GOSUB 9100
90      GOTO 40
320     REM ------------------------------
330     REM SIFIRLAMA
340     REM ------------------------------
350     FOR I = 1 TO 26
360     READ PO%(I,1),PO%(I,2)
370     FOR J = 1 TO 11
380     READ CAR%(I,J,1),CAR%(I,J,2)
390     NEXT
400     NEXT
410     RETURN
420     REM ------------------------------
430     REM VERILER
440     REM ------------------------------
450     REM    A
460     DATA 0,0
470     DATA 0,4,1,2,2,0,1,-2,0,-1,-4,0,
             4,0,0,-3,0,0,0,0,0,0
480     REM    B
490     DATA 0,0
500     DATA 0,6,1,0,1,-1,0,-1,-1,-1,-3,
             0,3,0,1,-1,0,-1,-1,-1,-3,0
510     REM    C
```

```
520     DATA 4,1
530     DATA -1,-1,-2,0,-1,1,0,4,1,1,2,0
           ,1,-1,0,0,0,0,0,0,0,0
540     REM  D
550     DATA 0,0
560     DATA 0,6,3,0,1,-1,0,-4,-1,-1,-3,
           0,0,0,0,0,0,0,0,0,0,0
570     REM  E
580     DATA 4,1
590     DATA -1,-1,-3,0,0,3,3;0,-3,0,0,3
             ,3,0,1,-1,0,0,0,0,0,0
600     REM  F
610     DATA 0,0
620     DATA 0,3,3,0,-3,0,0,3,3,0,1,-1,
           0,0,0,0,0,0,0,0,0,0
630     REM  G
640     DATA 2,3
650     DATA 2,0,0,-2,-1,-1,-2,0,-1,1,0,
              4,1,1,2,0,1,-1,0,0,0,0
660     REM  H
670     DATA 0,0
680     DATA 0,6,0,-3,4,0,0,3,0,-6,0,0,0
           ,0,0,0,0,0,0,0,0,0,0
690     REM  I
700     DATA 2,0
710     DATA 0,6,0,0,0,0,0,0,0,0,0,0,0,0,0
             ,0,0,0,0,0,0,0,0
720     REM  J
730     DATA 0,2
740     DATA 0,-1,1,-1,1,0,1,1,0,5,1,0,-
           3,0,0,0,0,0,0,0,0,0
750     REM  K
760     DATA 0,0
770     DATA 0,6,0,-3,2,0,2,3,-2,-3,2,-3
           ,0,0,0,0,0,0,0,0,0,0
780     REM  L
790     DATA 0,6
800     DATA 0,-6,3,0,1,1,0,0,0,0,0,0,0,
           0,0,0,0,0,0,0,0,0,0
810     REM  M
820     DATA 0,0
830     DATA 0,6,2,-3,2,3,0,-6,0,0,0,0,0
           ,0,0,0,0,0,0,0,0,0
840     REM  N
850     DATA 0,0
860     DATA 0,6,4,-6,0,6,0,0,0,0,0,0,0,
           0,0,0,0,0,0,0,0,0
870     REM  O
880     DATA 0,1
890     DATA 0,4,1,1,2,0,1,-1,0,-4,-1,-1
           ,-2,0,-1,1,0,0,0,0,0,0
900     REM  P
910     DATA 0,0
920     DATA 0,6,3,0,1,-1,0,-1,-1,-1,-3,,
           0,0,0,0,0,0,0,0,0,0,0
930     REM  Q
940     DATA 2,2
950     DATA 1,-2,-2,0,-1,1,0,4,1,1,2,0,
              1,-1,0,-4,-1,-1,0,0,0,0
960     REM  R
970     DATA 0,0
980     DATA 0,6,3,0,1,-1,0,-1,-1,-1,-3
           ,0,2,0,2,-3,0,0,0,0,0,0
990     REM  S
1000    DATA 0,1
1010    DATA 1,-1,2,0,1,1,0,1,-1,1,-2,0,
           -1,1,0,1,1,1,2,0,1,-1
1020    REM  T
1030    DATA 2,0
1040    DATA 0,6,-2,0,4,0,0,0,0,0,0,0,0,
```

```
1050  REM    U
1060  DATA 0,6
1070  DATA 0,-4,1,-2,2,0,1,2,0,4,0,0,0
        ,0,0,0,0,0,0,0,0,0
1080  REM    V
1090  DATA 0,6
1100  DATA 0,-3,2,-3,2,3,0,3,0,0,0,0,0
        ,0,0,0,0,0,0,0,0,0
1110  REM    W
1120  DATA 0,6
1130  DATA 0,-6,2,3,2,-3,0,6,0,0,0,0,0
        , 0,0,0,0,0,0,0,0,0
1140  REM    X
1150  DATA 0,0
1160  DATA 0,1,4,4,0,1,0,-1,-2,-2,-2,2
        ,0,1,0,-1,4,-4,0,-1,0,0
1170  REM    Y
1180  DATA 2,0
1190  DATA 0,3,-2,3,2,-3,2,3,0,0,0,0,0
        ,0,0,0,0,0,0,0,0,0
1200  REM    Z
1210  DATA  4,1
1220  DATA -1,-1,-3,0,4,6,-3,0,-1,-1,0
        ,0,0,0,0,0,0,0,0,0,0,0
9000  REM ------------------------------------------
9001  REM COZUMLEMEDEKI HARFLER
9002  REM ------------------------------------------
9023  REM ------------------------------------------
9100  D = D + 4
9110  IF MOV = 1 THEN SX = 1
      : SY = 1
      : M1 = 1
      : M2 = 2
9120  IF MOV = 2 THEN SX = 1
      : SY = -1
      : M1 = 2
      : M2 = 1
9130  IF MOV = 3 THEN SX = -1
      : SY = -1
      : M1 = 1
      : M2 = 2
9140  IF MOV = 4 THEN SX = -1
      : SY = 1
      : M1 = 2
      : M2 = 1
9150  LU = LEN (A$)
9160  FOR I = 1 TO LU
9170  LE$ = MID$ (A$,I,1)
9180  COD = ASC (LE$)
      : NL = COD - 64
9190  IF COD < 65 OR COD > 90 THEN 9260

      : REM  YENI ISARETLER ICIN
        DEGISTIRILECEK
9200  X1 = X0 + (S * P0%(NL,M1) * SX)
      : Y1 = Y0 - (S * P0%(NL,M2) * SY)
9210  FOR J = 1 TO 11
9220  X2 = X1 + (S * CAR%(NL,J,M1) * SX
      )
      : Y2 = Y1 - (S * CAR%(NL,J,M2) * SY
      )
9230  HPLOT X1,Y1 TO X2,Y2
9240  X1 = X2
      : Y1 = Y2
9250  NEXT
9260  IF MOV = 2 OR MOV = 4 THEN Y0
        = Y0 - (D * S * SY)
      : GOTO 9280
9270  X0 = X0 + (D * S * SX)
9280  NEXT
9290  RETURN
```

sı olarak ekranın sol aşağısındaki X köşe açısı alınır. Buna göre X ekseni için sağa ve Y ekseni için ise yukarıya doğru olmak üzere yerdeğişimlerin yönü pozitiftir. Tablodaki birinci sütun referans noktasına göre birinci noktanın koordinatlarını göstermektedir. Harfin ekrandaki yerinin kullanıcı tarafından belirlenmesi gerektiği için, bu koordinatlar gereklidir. Örneğin A harfi için çizim referans noktası olan Nokta 1'den başlar, demek ki referans noktasına göre ilk yerdeğiştirimi sıfırdır. C'yi çizmek için A örneğinin tersine başka bir noktadan başlamak ve bu uygulamada (4,1) olan bir ilk yerdeğişimi öngörmek gereklidir. Tablolarda 1,2,....., 11 şeklinde numaralanmış olan sütunlar her harfin üzerinde gösterilen yerdeğişimlerine eşit olmayan yerdeğişimlerini saptarlar. Bunlar izlenen güzergahı belirtirler. Örneğin A harfi için nokta 1'den nokta 2'ye giden birinci yerdeğişimi 0,4'dür, ikincisi ise 1,2'dir, v.b...

1495'ten 1498'e kadar program özellikle üç öbekten oluşur. 10. ve 90. satırlardaki ana öbek hafıza alanlarının tanımını (DIM) içerir ve bunları alt-programa çağırır. Bu alt-programlar arasında 350. satır bir dizi DATA komutu ile yerdeğişimlerinin değerlerini tanımlar, 9100. satır harfleri görüntüleyerek bu yerdeğişimleri gerçekleştirir.

Klavyeden yola çıkarak okunan X0,Y0,S,D, MOV parametreleri ve A$ tarafından gösterilen bir karakter zinciri örneği olsun. Programı genelleştirmek ve bunu dönüştürmek ve elde edilen veya başka bir yerde tanımlanmış

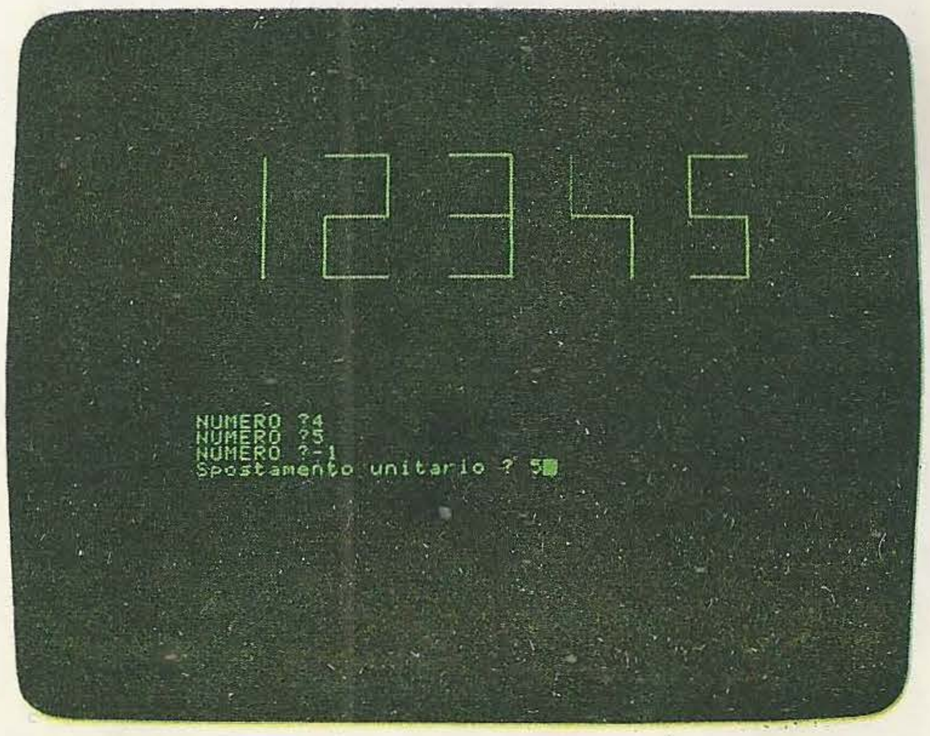
GRAFİK MODDA SAYISAL KARAKTERLERİN GÖSTERİLİŞİ

Bu sayfada alttaki ekran, sayısal karakterlerin grafik gösterilişini vermektedir.

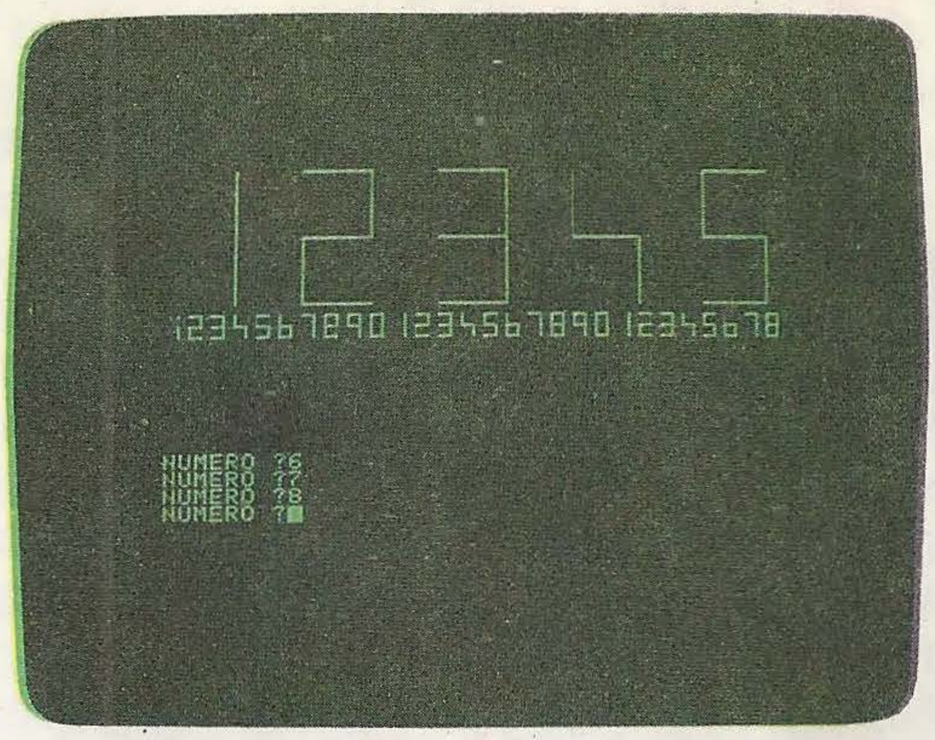
Yanda: Kullanıcı ile kurulan ilk etkileşim. Kullanıcı, başlangıç koordinatlarıyla birim yer değiştirmeyi giriyor.



Grafik mod harekete geçirildikten sonra da bir pencere kullanıcı ile etkileşimin sürdürülebilmesi için metin modunda bırakılır.
Kullanıcı, ekranda gösterilecek ve görüntülenecek bir sayılar dizisi giriyor.
Negatif bir sayının girilmesi birim yer değiştirme için yeni bir değerin verilmesini gerektiriyor.



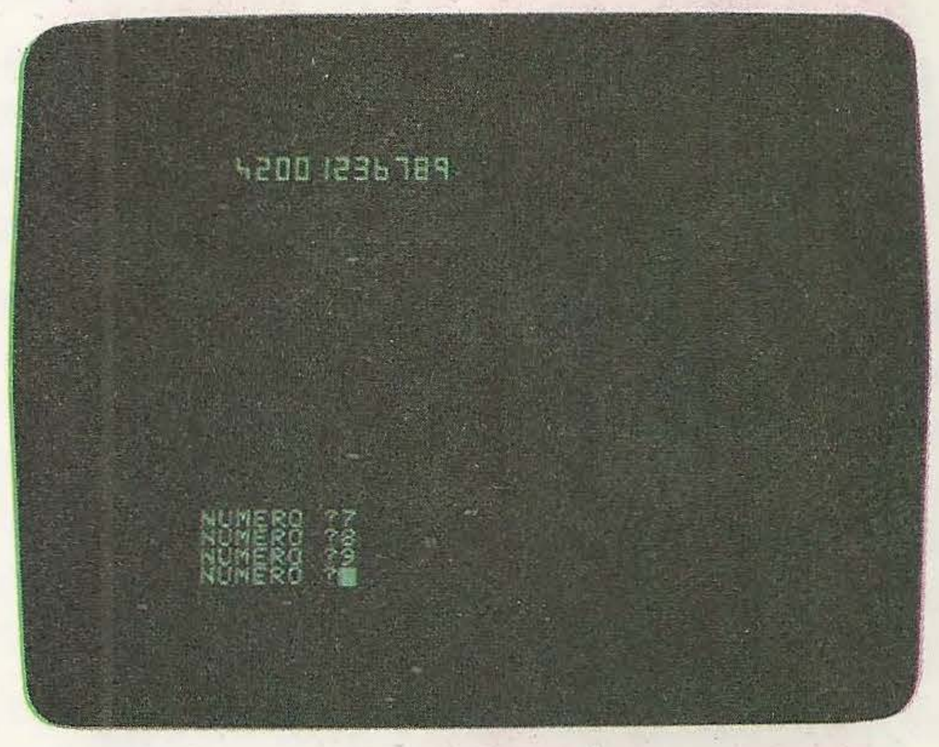
Daha küçük bir yer değişimi seçtikten sonra, sayıların büyüklüğü azalır. Ancak orantıların aynı kaldığını iyice kaydedelim.



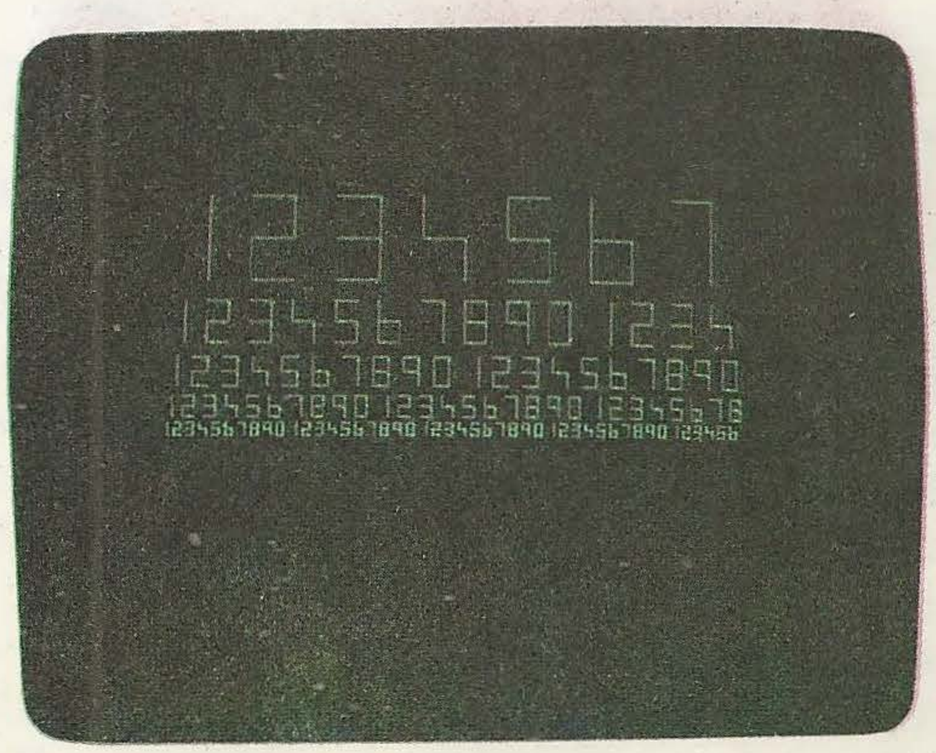
Bu durumda birim yer değişimi ile çıkış koordinatlarının değerleri değiştirildi.



Bir önceki adımla girilen yerdeğişiminin sonucunda, görüntülenen sayılar, kullanıcıyla etkileşim için ayrılan bölümde gördüğümüz standart sayılardan daha büyüktür.



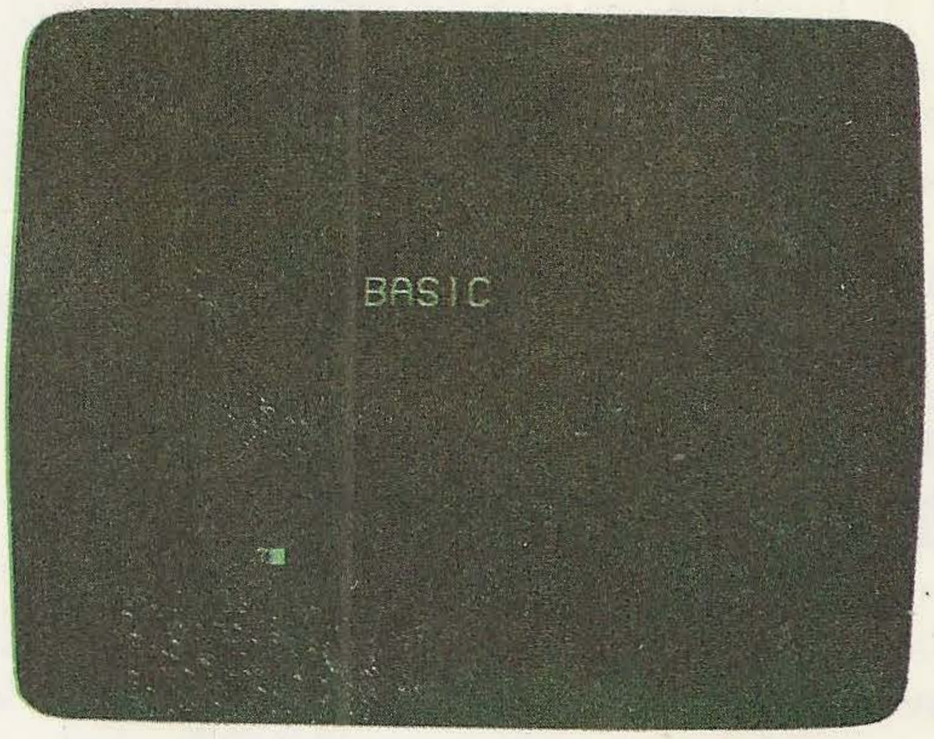
Sonuncu ekran, birim yerdeğişim değeri küçüldükçe adım adım elde edilen ölçek ufalmasını gösteriyor.



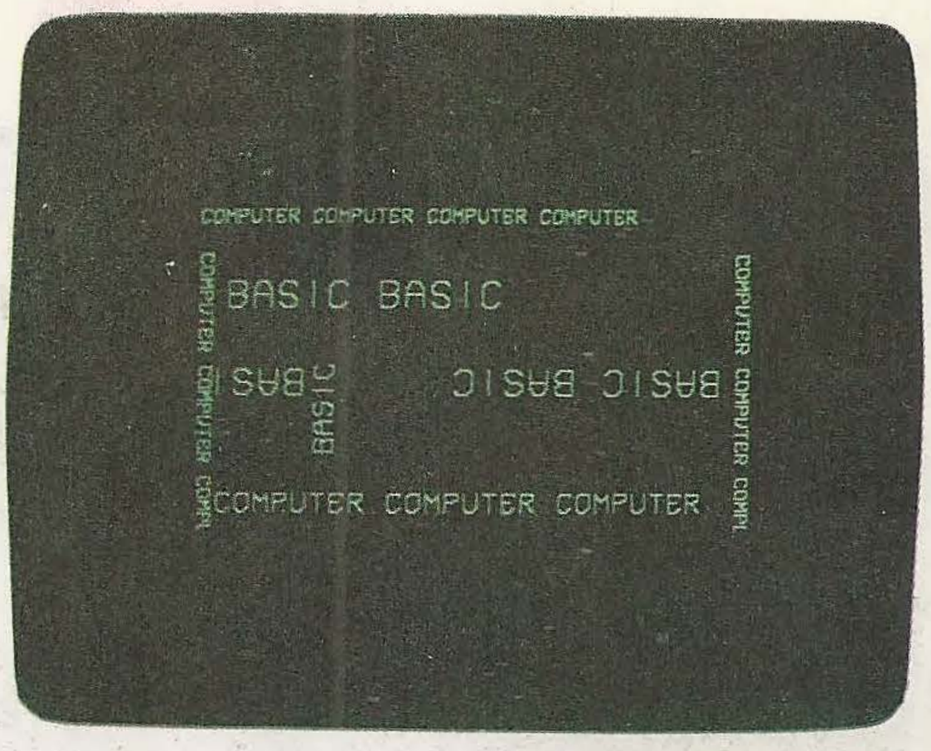
GRAFİK MODDA ALFABETİK KARAKTERLERİN GÖRÜNTÜLENMESİ

Grafik modda alfabetik karakterlerin kullanım örnekleri

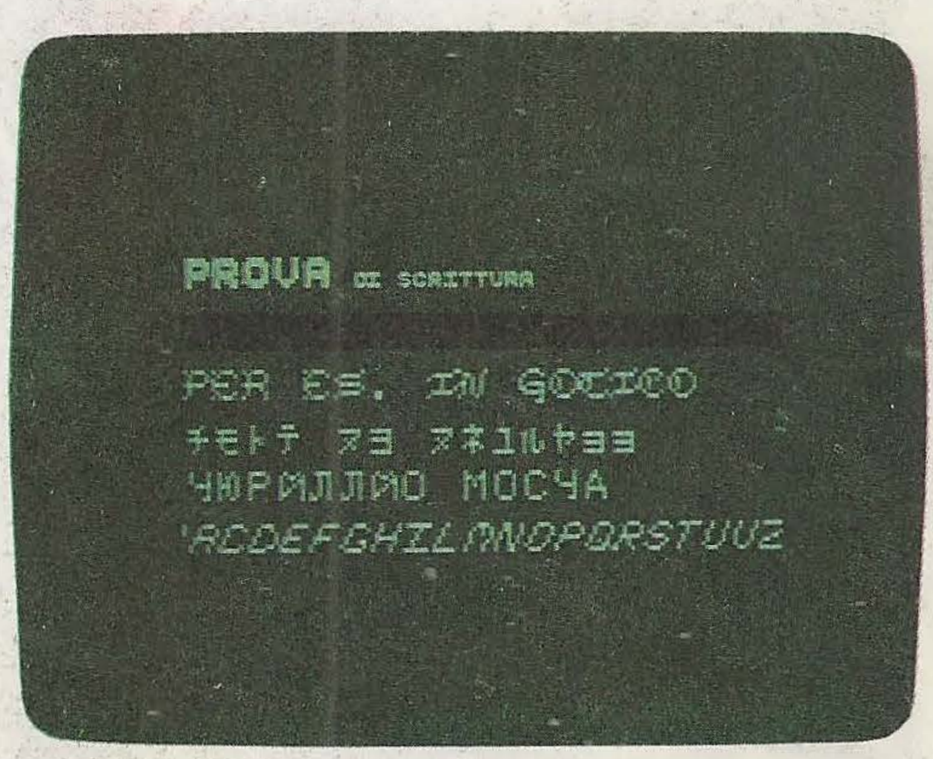
Yanda, programın uygulanmasının sonucu: Girilen karakter zinciri BASIC'tir.



İkinci ekran programın değişik olanaklarını gösteriyor. Metin arzu edilen herhangi bir yönde ve ekranın herhangi bir yerinde konumlandırılabilir.



Karakterlerin çizim planını daha çok işleyerek farklı tipografik karakterleri üretmek olanaklı. Yandaki ekran bu karakter türlerinin bir kaç örneğini sunmakta.



olan değerleri bu alt-programa vermek yeterlidir.

Kullanılan parametreler:

X0,Y0	Karakterlerin içinde yer alacağı dikdörtgenin referans noktasının koordinatları
S	Harflerin görüntüleme ölçülerini saptayan ölçek faktörü. Yerdeğişimlerini S ile çarparsak aynı harfi büyütülmüş olarak elde ederiz.
D	İki harf arasındaki uzaklığı tanımlar.
MOV	Metnin yönlendiriliişini gösterir.1'den 4'e kadarki numaralar ile kodlanan yönlendirilişler dökümde belirtilmiştir.

Histogramlar ve Dilimli Diyagramlar

Herhangi bir grafik gösterim, grafiği betimleyen bir matematik fonksiyonun varolması koşuluyla, bu fonksiyonun eğrisini çizerek elde edilebilir. Bazı olaylar bir fonksiyon aracılığıyla betimlenemez çünkü ya gösterilecek miktar sürekli değişen bir bağımsız değişkene bağlı değildir ya da bağımsız değişken sayısal türden bir değişken değildir.

Değişik satıcılar için bir ürünün satışı ile ilgili bir örneği daha önce vermiştik. Bu örnekte A, B,C... satıcılarını sayısal veriler olarak ele almak olanaksızdı. Çok sık karşılaşılan bir diğer örnek de bir yıl boyunca yapılan aylık bilançolardır. Bu durumda bağımsız değişken tartışmasız sayısal bir değer olmayan Ocak, Şubat, ... gibi ay isimleridir. Grafik gösterimin bu sorunlarını çözmek için çok yaygın iki yöntem vardır: histogramlar ile dilimli diyagramlar.

Histogramlarda sütunların Y ekseni üzerinde aldığı değerler bağımlı değişkenin değeriyle doğru orantılıdır; X ekseninde ise sütunların arasındaki mesafelerin büyüklüğü, bağımsız değişken X'in değerlerine orantılı olur. Genellikle olay düzenli aralıklarla gözlendiğinden bu adımlar sabittir. Bir bilanço örneğinde bağımsız değişken zaman, bağımlı değişken ise tutardır. X ekseni 12 bölüme bölündüğünden, bu değişkenler aylık bir süre olarak verilirler.

Oysa Y ekseni ister mutlak değer olarak olsun, ister toplamın yüzdesi olarak olsun, aylık rakamı gösterir. Bu son şekil en çok kullanılanıdır, çünkü hem olayın görüntüsünü anında verir hem de diyagramın yapılmasını kolaylaştırır.

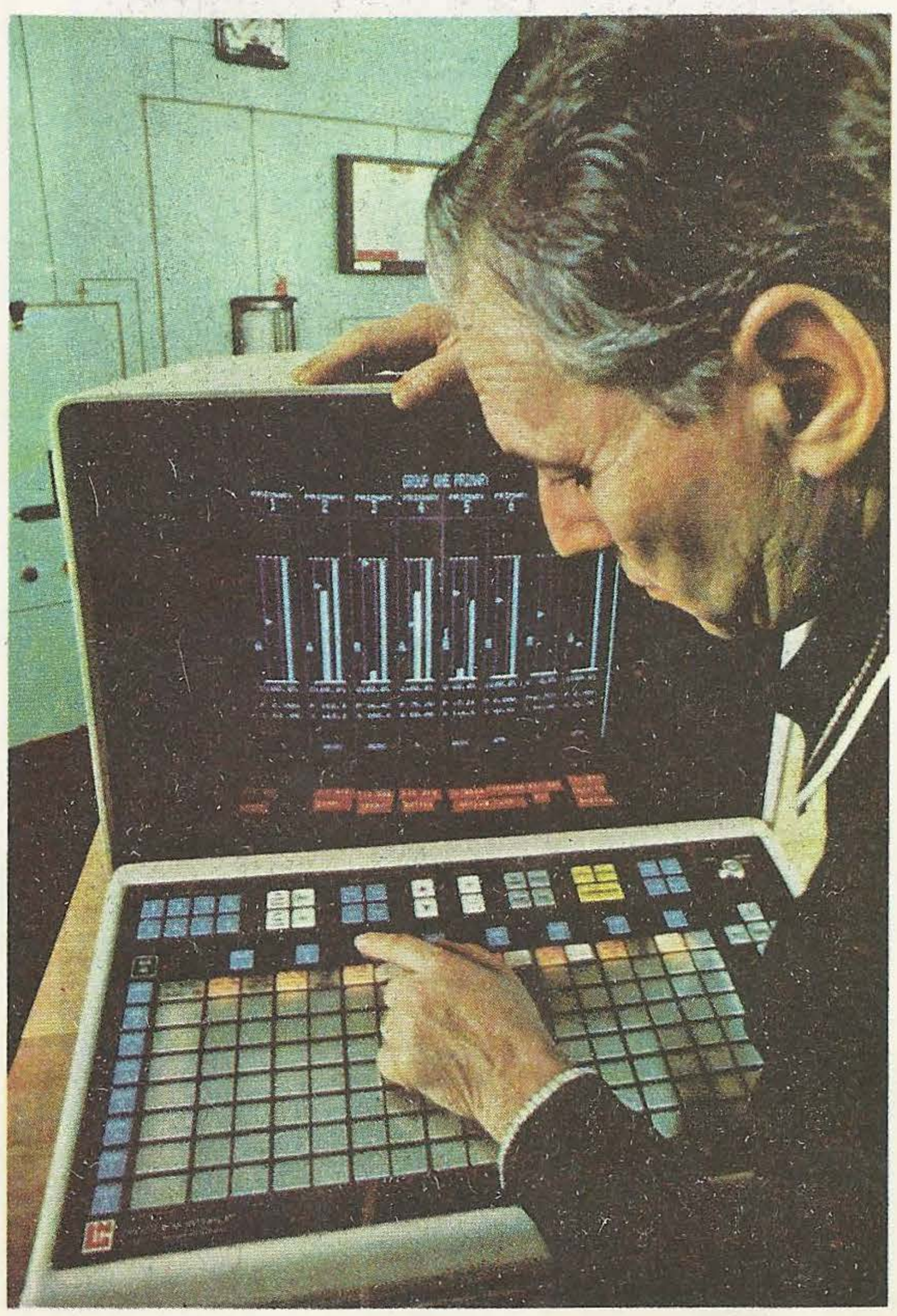
Dilimli diyagram histograma yakın bir gösterim biçimidir. Burada her bir değer, büyüklüğüyle doğru orantılı bir çember payına dönüştürülür.

Çemberin tümü toplam değeri gösterir, her bir dilim ise çemberin parçalarından birinin değerini verir. Örnek: Bir önceki bilançoyu bu şekilde göstermek için, her aya bir tane karşılık gelecek şekilde, bir çemberi 12 dilime bölmek gerekir. Dilimlerin büyüklüğü toplamyıllık tutara göre o ayki tutarın yüzdesine orantılıdır.

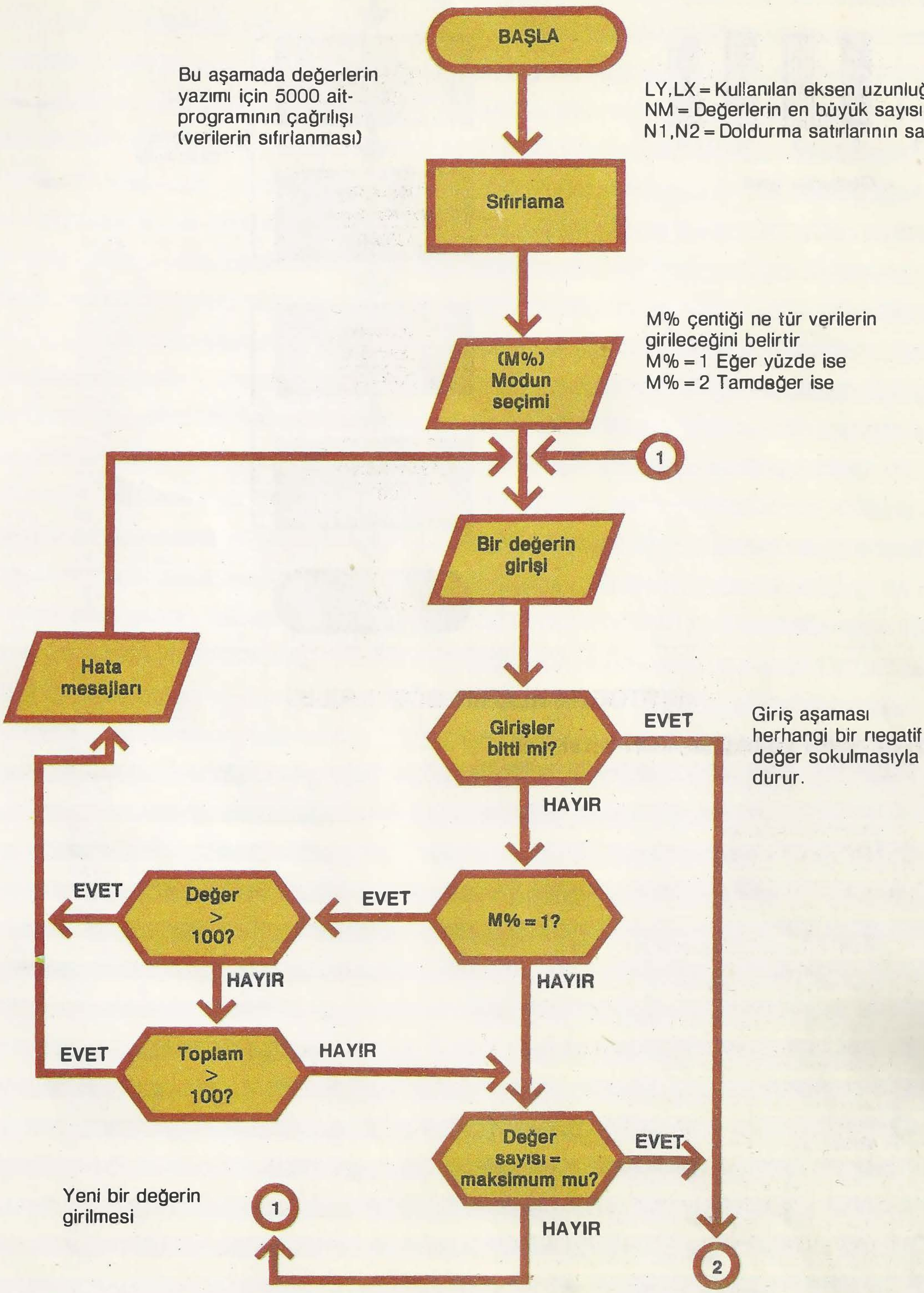
Demek ki histogram bilançonun zaman içindeki gelişimini göstermekte, oysa dilimli diyagram aylarla ifade edilen farklı parçalar arasında bilançonun bütününün nasıl bir dağılım gösterdiğini gözönüne koyar.

Demek ki birbirini tamamlayan iki gösterim biçimi sözkonusudur. 1503. ve 1504. sayfalarda histogramların gösterim programının akış şeması yapıldı (1504-1510. sayfalardaki dökümlere bakınız). Gösterilecek olan veriler ya mutlak değerler ya da yüzdeliklerdir. Birinci durumda program , verileri yüzdeliğe kendi çevirir. Kullanıcı tek renkli bir ekranda aynı grafiğin 5 ayrı gösterim şekli arasında seçim yapar.

Bir grafik işlem uygulaması.



HİSTOGRAMLARIN GRAFİK GÖSTERİLİŞİ



Bu aşamada değerlerin yazımı için 5000 ait-programının çağrılışı (verilerin sıfırlanması)

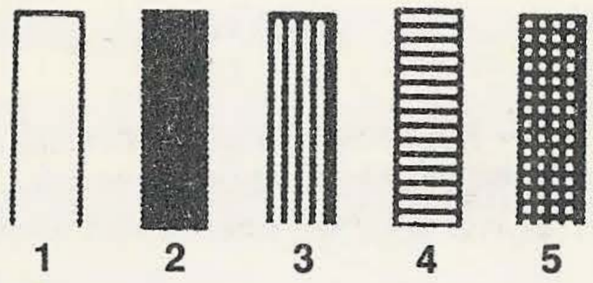
LY,LX = Kullanılan eksen uzunluğu
NM = Değerlerin en büyük sayısı
N1,N2 = Doldurma satırlarının sayısı

M% çentiği ne tür verilerin girileceğini belirtir
M% = 1 Eğer yüzde ise
M% = 2 Tamdeğer ise

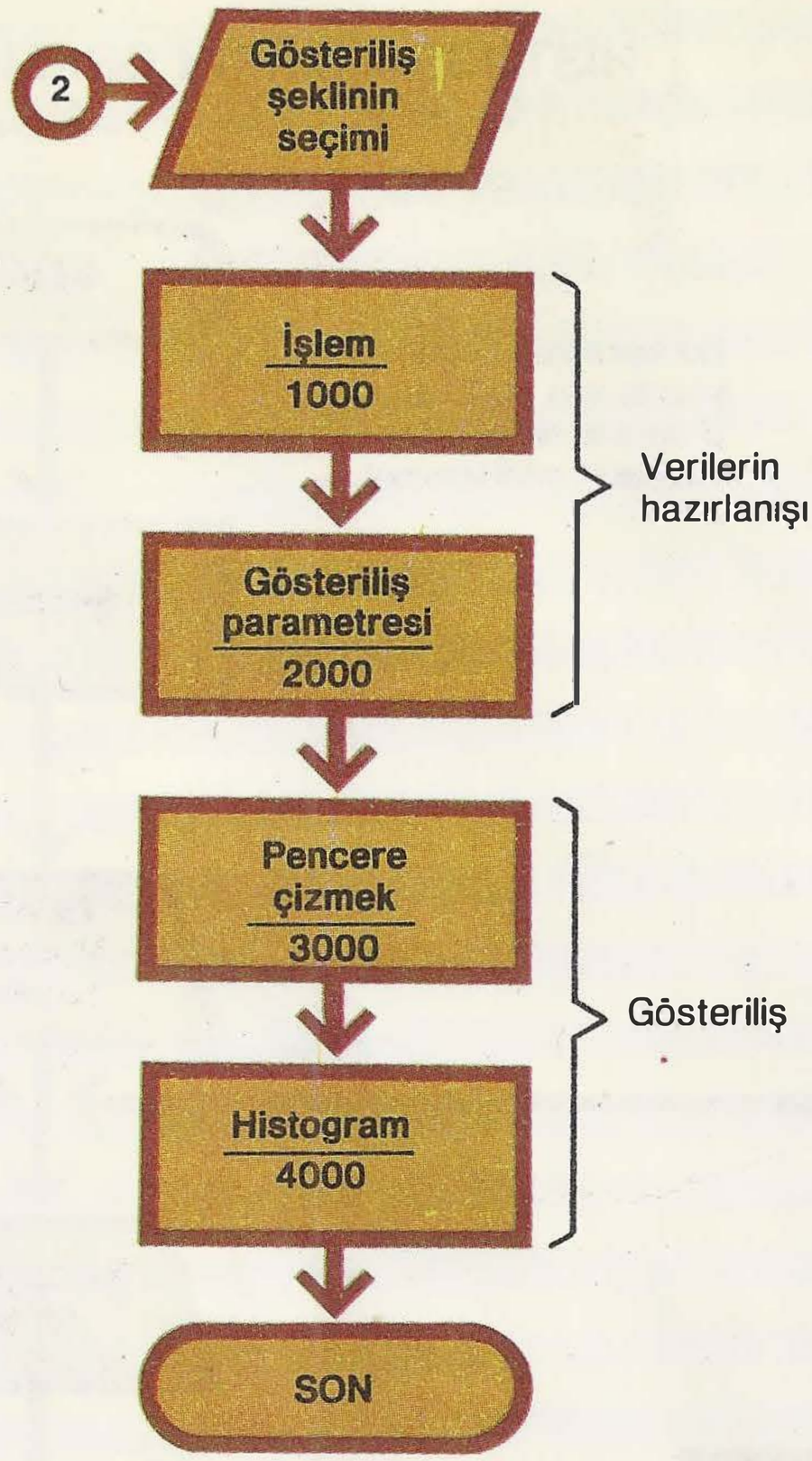
Giriş aşaması herhangi bir negatif değer sokulmasıyla durur.

Yeni bir değerin girilmesi

1 = Boş
2 = Dolu
3 = Dikey satırlar
4 = Yatay satırlar
5 = Kareli



Gösteriliş şekli



HİSTOGRAMLARIN GÖSTERİLİŞİ

Siprel, Apple ve bağdaşıkları versiyonu

```
4       REM ------------------------
5       REM HISTOGRAM
6       REM ------------------------
7
        :
8
        :
9       DIM PO%(26,2),CAR%(26,11,2)
10      LX = 230
        : LY = 150
20      NM = 20
30      P% = 0
50      DIM D(NM),P(NM),S(NM),E$(NM)
60      GOSUB 5000
65      GOSUB 9120
70      TEXT
        : HOME
80      VTAB 12
        : INPUT "VERILER YUZDE ILE MI
                 ? (E/H)";A$
90      IF A$ = "E" THEN M% = 1
        : P% = 1
        : GOTO 110
100     M% = 2
110     I  = 1
        : TT = 0
120     HOME
```

```
      : VTAB 12
130   PRINT "N VERISINI GIRMEK"I
140   INPUT "(BITIRMEK ICIN NEGATIF): ";D
        $
      : IF D$ = "" THEN 120
150   D = VAL (D$)
160   IF D > 99999 THEN 120
170   IF D < 0 THEN 260
180   D(I) = INT (D)
190   TT = TT + D(I)
200   IF M% = 2 THEN 220
210   IF D(I) > 100 OR TT > 100 THEN
          GOSUB 6000
      : HOME
      : GOTO 110
      : REM          HATA
220   VTAB 15
      : INPUT "ETIKET (MAK.3 ISARET
        ):";E$
230   E$(I) = LEFT$ (E$,3)
240   I = I + 1
      : IF I >NM THEN 260
250   GOTO 120
260   IF TT = 0 THEN RUN
270   NV = I - 1
280   HOME
290   VTAB 5
      : PRINT "GOSTERME MODLARI: "
300   VTAB 10
310   PRINT "1)  BOS"
320   PRINT "2)  DOLU"
330   PRINT "3)  DIKEY"
340   PRINT "4)  YATAY"
350   PRINT "5)      "
360   VTAB 22
      : INPUT "SECME :";MP$
      : IF MP$ = "" THEN 360
370   MP = VAL (MP$)
380   IF MP < 1 OR MP > 5 THEN 280
1000  REM ------------------------
1010  REM  ISLEMLER
1020  REM ------------------------
1030  IF M% = 1 GOTO 1140
1040  HOME
      : VTAB 12
1050  PRINT "YUZDE ILE MI GOSTERMEK
        ISTIYORSUNUZ ?";
1060  INPUT "(E/H) ";R$
1070  IF R$ = "E" THEN P% = 1
      : GOTO 1140
1080  MA = D(1)
1090  FOR K = 2 TO NV
1100  IF D(K) > (MA) THEN MA = D (K)
1120  NEXT K
1130  TT = MA
1140  FOR I = 1 TO NV
1150  P(I) = D(I) / TT * 100
1160  S(I) = INT (LY * P(I) / 100)
1170  NEXT I
1180  LA = INT (LX / (2 * NV))
2000  REM ------------------------
2001  REM GOSTERME PARAMETRESI
2002  REM ------------------------
2010  SX = 1
      : SY = 1
2040  ON MP GOTO 3000,3000,2060,2070,20
        60
2060  SX = 2
      : GOTO 3000
2070  SY = 2
      : GOTO 3000
```

```
2080  SY = 2
    : SX = 2
3000  REM ------------------------------
3010  REM PENCERELERIN BICIMI
3020  REM ------------------------------
3030  HGR2
    : HCOLOR = 3
3040  HPLOT 34,0 TO 279,0 TO 279,159
        TO 34,159 TO 34,0
3050  FOR I = 1 TO 10
3060  V = 159 - I * 15
3070  HPLOT 34,V TO 39,V
3075  HPLOT 274,V TO 278,V
3080  NEXT
3090  S = 3
    : YO = 159
3100  IF P% = 0 THEN 3170
3110  FOR NN = 0 TO 100 STEP 10
3120  XO = 9 * S
3130  GOSUB 5500
3140  YO = YO - 15
3150  NEXT
3160  GOTO 3250
3170  ST = MA / 10
    : IF ST < > INT (ST) THEN ST =
        INT (ST) + 1
3175  YO = 9
3180  FOR NN = MA TO 0 STEP -ST
3190  XO = 9 * S + 3
3200  GOSUB 5500
3210  YO = YO + 15
3220  NEXT
3230  XO = 9 * S + 3
3235  IF NN + ST = 0 THEN 3250
3240  NN = 0
    : GOSUB 5500
3250  REM
4000  REM ------------------------------
4001  REM HISTOGRAMLARIN BICIMLENDIRILMESI
4002  REM ------------------------------
4005  N = 0
4007  Y1 = 159
4010  FOR I = 1 TO NV
4025  X1 = 50 + N * LA
4027  X2 = X1 + LA
4030  HPLOT X1,Y1 TO X1,Y1 - S(I) TO X2
        ,Y1 - S(I) TO X2,Y1
4040  IF MP = 1 THEN 4900
4050  IF MP = 4 THEN 4300
4060  Y = Y1
    : X = X1 + SX
4070  IF X > X2 THEN 4200
4080  HPLOT X,Y TO X,Y - S(I)
4090  X = X + SX
4100  GOTO 4070
4200  IF MP = 5 THEN 4300
4210  GOTO 4900
4300  X = X1
    : Y = Y1 - SY
4310  IF Y < Y1 - S(I) THEN 4900
4320  HPLOT X1,Y TO X2,Y
4330  Y = Y - SY
4340  GOTO 4310
4900  N = N + 2
4905  A$ = E$ (I)
    : HO =X2
    : VO = 181
    : S = 1
    : D = 2
    : MOV = 4
    : GOSUB 10270
```

```
4910  NEXT I
4920  GET F$
4930  TEXT
      : HOME
      : VTAB 12
      : INPUT "BASKA BIR GRAFIK ISTIYORMUSUNUZ";R$
4940  IF LEFT$ (R$,1) = "E" THEN 70
4950  END
5000  REM ------------------------------
5010  REM SAYILARIN SIFIRLANMASI
5020  REM ------------------------------
5030  DIM XD(7),YD(7),XA(7),YA(7)
5040  DIM V(7),C%(10,7)
5050  FOR I = 1 TO 7
5060  READ XD(I),YD(I),XA(I),YA(I)
5070  NEXT I
5080  FOR I = 1 TO 10
5090  FOR J = 7 TO 1 STEP -1
5100  READ C%(I,J)
      : NEXT J
5110  NEXT I
5120  RETURN
5500  REM ------------------------------------
5510  REM SAYILARIN BICIMLENDIRILMESI
5520  REM ------------------------------------
5530  NN$ = STR$(NN)
5540  LN = LEN(NN$)
5550  FOR KK = LN TO 1 STEP -1
5560  N = VAL (MID$ (NN$,KK,1))
5570  XO = XO -2 * S
5580  FOR I = 1 TO 7
      : V(I) = 0
      : NEXT I
5590  L% = N
      : IF L% = 0 THEN L% = 10
5600  FOR I = 1 TO 7
      : V(I) = 0
      : NEXT I
5590  L% = N
      : IF L% = 0 THEN L% = 10
5600  FOR I = 1 TO 7
5610  V(I) = C%(L%,I)
5620  NEXT I
5630  FOR I = 1 TO 7
5640  IF V(I) = 0 GOTO 5710
5650  K = I
5660  X1 = XO + XD(K) * S
5670  Y1 = YO - YD(K) * S
5680  X2 = XO + XA(K) * S
5690  Y2 = YO - YA(K) * S
5700  HPLOT X1,Y1 TO X2,Y2
5710  NEXT I
5720  NEXT KK
5730  RETURN
6000  VTAB 20
      : PRINT "HATALI DEVREYE SOKULUS:":PRINT
6010  PRINT"VERILERIN DEGERI 100 DEGERINI GECIYOR"
6015  PRINT
6020  PRINT "YENIDEN BASLAMAK ICIN > TUSUNA BASIN";
6030  GET Q$
6040  RETURN
7945  REM ------------------
7950  REM VERILER
7955  REM ------------------
8000  DATA 0,0,0,1
      : REM = a
8010  DATA 0,1,0,2
      : REM = b
8020  DATA 0,2,1,2
      : REM = c
8030  DATA 1,2,1,1
      : REM = d
```

```
8040  DATA 1,1,1,0
      : REM =e
8050  DATA 1,0,0,0
      : REM=f
8060  DATA 0,1,1,1
      : REM=g
8070  DATA 0,0,1,1,0,0,0
      : REM = 1
8080  DATA 1,1,0,1,1,0,1
      : REM =2
8090  DATA 1,1,1,1,1,0,0
      : REM =3
8100  DATA 1,0,1,0,0,1,0
      : REM =4
8110  DATA 1,1,1,0,1,1,0
      : REM =5
8120  DATA 1,1,1,0,0,1,1
      : REM= 6
8130  DATA 0,0,1,1,1,0,0
      : REM =7
8140  DATA 1,1,1,1,1,1,1
      : REM =8
8150  DATA 1,0,1,1,1,1,0
      : REM =9
8160  DATA 0,1,1,1,1,1,1
      : REM =0
9090  REM ------------------------------
9100  REM SIFIRLAMA
9110  REM ------------------------------
9120  FOR I = 1 TO 26
9130  READ PO%(I,1),PO%(I,2)
9140  FOR J = 1 TO 11
9150  READ CAR%(I,J,1),CAR%(I,J,2)
9160  NEXT J
9170  NEXT I
9180  RETURN
9190  REM ------------------------------
9200  REM VERILER
9210  REM ------------------------------
9220  REM     A
9230  DATA 0,0
9240  DATA 0,4,1,2,2,0,1,-2,0,-1,-4,0,
           4,0,0,-3,0,0,0,0,0,0
9250  REM     B
9260  DATA 0,0
9270  DATA 0,6,3,0,1,-1,0,-1,-1,-1,-3,
           0,3,0,1,-1,0,-1,-1,-1,-3,0
9280  REM     C
9290  DATA 4,1
9300  DATA -1,-1,-2,0,-1,1,0,4,1,1,2,0
           ,1,-1,0,0,0,0,0,0,0,0
9310  REM     D
9320  DATA 0,0
9330  DATA 0,6,3,0,1,-1,0,-4,-1,-1,-3,
        0,0,0,0,0,0,0,0,0,0,0
9340  REM     E
9350  DATA 4,1
9360  DATA -1,-1,-3,0,0,3,3,0,-3,0,0,3
        ,3,0,1,-1,0,0,0,0,0,0
9370  REM     F
9380  DATA 0,0
9390  DATA 0,3,3,0,-3,0,0,3,3,0,1,-1,
        0,0,0,0,0,0,0,0,0,0
9400  REM     G
9410  DATA 2,3
9420  DATA 2,0,0,-2,-1,-1,-2,0,-1,1,0,
        4,1,1,2,0,1,-1,0,0,0,0
9430  REM     H
9440  DATA 0,0
9450  DATA 0,6,0,-3,4,0,0,3,0,-6,0,0,0
        ,0,0,0,0,0,0,0,0,0
```

```
9460  REM  I
9470  DATA 2,0
9480  DATA 0,6,0,0,0,0,0,0,0,0,0,0,0,0
        ,0,0,0,0,0,0,0,0
9490  REM  J
9500  DATA 0,2
9510  DATA 0,-1,1,-1,1,0,1,1,0,5,1,0,-
        3,0,0,0,0,0,0,0,0,0
9520  REM  K
9530  DATA 0,0
9540  DATA 0,6,0,-3,2,0,2,3,-2,-3,2,-3
        ,0,0,0,0,0,0,0,0,0,0
9550  REM  L
9560  DATA 0,6
9570  DATA 0,-6,3,0,1,1,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0
9580  REM  M
9590  DATA 0,0
9600  DATA 0,6,2,-3,2,3,0,-6,0,0,0,0,0
        ,0,0,0,0,0,0,0,0,0
9610  REM  N
9620  DATA 0,0
9630  DATA 0,6,4,-6,0,6,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0
9640  REM  O
9650  DATA 0,1
9660  DATA 0,4,1,1,2,0,1,-1,0,-4,-1,-1
        ,-2,0,-1,1,0,0,0,0,0,0
9670  REM  P
9680  DATA 0,0
9690  DATA 0,6,3,0,1,-1,0,-1,-1,-1,-3,
        0,0,0,0,0,0,0,0,0,0,0
9700  REM  Q
9710  DATA 2,2
9720  DATA 1,-2,-2,0,-1,1,0,4,1,1,2,0,
        1,-1,0,-4,-1,-1,0,0,0,0
9730  REM  R
9740  DATA 0,0
9750  DATA 0,6,3,0,1,-1,0,-1,-1,-1,-3
        ,0,2,0,2,-3,0,0,0,0,0,0
9760  REM  S
9770  DATA 0,1
9780  DATA 1,-1,2,0,1,1,0,1,-1,1,-2,0,
        -1,1,0,1,1,1,2,0,1,-1
9790  REM  T
9800  DATA 2,0
9810  DATA 0,6,-2,0,4,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0
9820  REM  U
9830  DATA 0,6
9840  DATA 0,-4,1,-2,2,0,1,2,0,4,0,0,0
        ,0,0,0,0,0,0,0,0,0
9850  REM  V
9860  DATA 0,6
9870  DATA 0,-3,2,-3,2,3,0,3,0,0,0,0,0
        ,0,0,0,0,0,0,0,0,0
9880  REM  W
9890  DATA 0,6
9900  DATA 0,-6,2,3,2,-3,0,6,0,0,0,0,0
        ,0,0,0,0,0,0,0,0,0
9910  REM  X
9920  DATA 0,0
9930  DATA 0,1,4,4,0,1,0,-1,-2,-2,-2,2
        ,0,1,0,-1,4,-4,0,-1,0,0
9940  REM  Y
9950  DATA 2,0
9960  DATA 0,3,-2,3,2,-3,2,3,0,0,0,0,0
        ,0,0,0,0,0,0,0,0,0
9970  REM  Z
9980  DATA 4,1
9990  DATA -1,-1,-3,0,4,6,-3,0,-1,-1,0
        ,0,0,0,0,0,0,0,0,0,0,0
```

```
10000 REM ----------------------------
10010 REM
10020 REM ----------------------------
10030 REM
10040 REM
10050 REM  GIRISTE
10060 REM
10070 REM  A$ - KARAKTER ZINCIRLERI
10080 REM         KANALI
10090 REM HO,VO - YAZIM BASI
10100 REM            KONUM
10110 REM
10120 REM  S - OLCEK ( > 0)
10130 REM  D - ARALANMA ( > 0)
10140 REM  MOV - YAZIM
10150 REM          TURU:
10160 REM
10170 REM     1 = YATAY
10180 REM     2 = DIKEY
10190 REM     3 = TERS
10200 REM     4 = CIKAN
10210 REM
10220 REM
10230 REM ----------------------------------------
10270 D = D + 4
10280 IF MOV = 1 THEN PX = 1
      : PY = 1
      : M1 = 1
      : M2 = 2
10290 IF MOV = 2 THEN PX = 1
      : PY = -1
      : M1 = 2
      : M2 = 1
10300 IF MOV = 3 THEN PX = -1
      : PY = -1
      : M1 = 1
      : M2 = 2
10310 IF MOV = 4 THEN PX = -1
      : PY = 1
      : M1 = 2
      : M2 = 1
10320 LU = LEN (A$)
10330 FOR II = 1 TO LU
10340 LE$ = MID$ (A$,II,1)
10345 IF LE$ = "" THEN II = LU
      : NEXT II
      : RETURN
10350 COD = ASC (LE$)
      : NL = COD - 64
10360 IF COD < 65 OR COD > 90 THEN 1043
         0
      : REM YENI KARAKTERLER ICIN
           DEGISTIRILECEK
10370 H1 = HO + (S * PO%(NL,M1) * PX)
      : V1 = VO - (S * PO%(NL,M2) * PY)
10380 FOR J = 1 TO 11
10390 H2 = H1 + (S * CAR%(NL,J,M1) * PX
        )
      : V2 = V1 - (S * CAR%(NL,J,M2) * PY
        )
10400 HPLOT H1,V1 TO H2,V2
10410 H1 = H2
      : V1 = V2
10420 NEXT J
10430 IF MOV = 2 OR MOV = 4 THEN VO
         = VO - (D * S * PY)
      : GOTO 10450
10440 HO = HO + (D * S * PX)
10450 NEXT II
10460 RETURN
```

Bilgisayar ve Radar İnsanları

Hepiniz hava trafiği denetçilerinden, yani radar operatörlerinden sözedildiğini duymuşsunuzdur. Ancak çok az insan bu denetçilerin işlerinin neo lduğunu bu işlerin sağlıklı bir şekilde yürütebilmesi için yeni teknolojilerin ne kadar gerekli olduğunu gerçek anlamda bilir. Hava trafiğinin olağan, gündelik kontrolünden bilgisayarlar arası veri alışverişine kadar tüm işlemlerin tamamen otomatik olarak yürütüldüğü hava trafiği örneklerine bugün dünyada rastlamak mümkündür.

Şu veya bu sistemin seçilmesi elbette ki otomatikleşme maliyeti ile bundan sağlanacak yararlar arasındaki ilişkiye bağlıdır. Radarla denetim durumunda maliyet, bakım ve yüksek derecede uzman bir personelin oluşturulması masraflardır. Öyle ki bu masraflar hala birçok ülkeyi böyle bir maceraya atılmaktan caydırmaktadır. Radarla denetimden sağlanacak yararlar arasında ise güvenliğin artmasını ve daha iyi çalışma koşullarını saymak gerekir. Elbette ki bütün bunlar denetlenecek trafiğin yoğunluğuna bağlıdır.

Hava trafiği kontrolü (HTK), yolcuların ve hava yolları şirketlerinin çıkarını kollamak üzere örgütlenmiş ve hızlı işleyen bir hava trafiği sağlamayı amaçlayan bir hizmettir. Hava trafiği kontrolünün karmaşık sorunlarını düzenleyen ve kontrol eden dünya çapındaki örgüt ICAO (International Civil Aviation Organization- Uluslararası Sivil Havacılık Örgütü)'dur.

Trafiği kolaylaştırmak için dünya hava sahası, coğrafi bakımdan ilgili uluslara denetimi emanet edilen FIR'lara (Flight Information Region- Uçuş Danışma Bölgesi) bölünmüştür.

Genişlikleri yaklaşık 10 mil ve yükseklik düzeyleri ise koşullara bağlı olarak farklı olabilen FIR'lar elverişli seyir yollarının göğe çizimini mümkün kılar. Sanılabileceğinin tam tersine denetim hizmeti bütün uçuş süresini kapsar. Denetim uçağın havalanmasından başlayarak her zaman için en nazik uygulama aşamaları olan piste yaklaşmak ve yere konmaya kadar devam eder.

Denetçinin görevi hava güzergâhları boyunca hem pilotların kendi isteklerini hem de yürürlükte olan trafiğin durumunu gözönünde bulundurarak uçakları radyo ile yönetmektir. Verilerin tümüne, yalnızca denetçi sahiptir,



İtalya hava sahasını üçe bölen FIR hatları.

çünkü pilotun görüş alanı biri hız olmak üzere (bir uçak dakikada yaklaşık olarak 10 kilometre kat eder) farklı etkenlerce sınırlanmış olur. Denetçinin işini kolaylaştırmak için pilotlar kalkıştan ön e bir uçuş planı yazmak zorundadırlar. Bu plan uçakla ilgili bütün bilgileri, uçağın teknik özelliklerini (tipini), gideceği yeri, tarifedeki saatlerin bir tahminini de kayd eder. U çuş planında katedilecek yol bir hava güzrgahları veya karekteristik noktalar (FIX) silsilesi şeklinde ifade edilir. Çoğu zaman bu rüzgarlar veya noktalar uluslararası haritalar üzerinde kararlaştırılan isimlerle anılırlar. Bu noktalara radyo cihazının gösterdiği uçuş pistindeki işaretler denk düşerler.

Uçuş planı gidilecek yerle ilgili bütün uçuş bölgelerine (FIR)'lara AFTN (Aeronautical Fixed Telecommunication Network- Saptanan Hava Tele İletişim Ağı) adı verilen özel bir teleks şebekesi sayesinde iletilir.

Her bölgede, radar operatörü yardımcıları her

bir uçuş planından kendi bölgeleri ile ilgili olan bilgileri alırlar. Gidilecek yerle ilgili her bir FIX için yardımcılar uçağın önemli verilerini, istenen yüksekliği, FIX'in ismi ve öngörülen fazla uçuş saatini bir teleks kuşağı aracılığıyla alırlar. Örneğin Roma FIR'ında uçuş başına 10-15 arası strip hazırlanır. Demek ki günde yaklaşık olarak 900 tanelik bir uçuş ortalaması için gündelik strip sayısı yaklaşık olarak 10000'e erişir. Denetçinin belirli bir FIX'e denk düşen trafiği her an bilmesini sağlayan geleneksel sistem, uçakların aynı düzeyde kesişmesini önlemek için o noktayla ilgili striplerin zamana göre sıraya dizilmesinden oluşur.

Yordam adı verilen bu tip bir denetim için iki uçak arasındaki gereken uzaklık aynı yükseklikte uçuyorlarsa genellikle 10 dakikalık bir uzaklıktır, eğer farklı yükseklikte seyir ediyorlarsa da bu 300 metre olmalıdır.

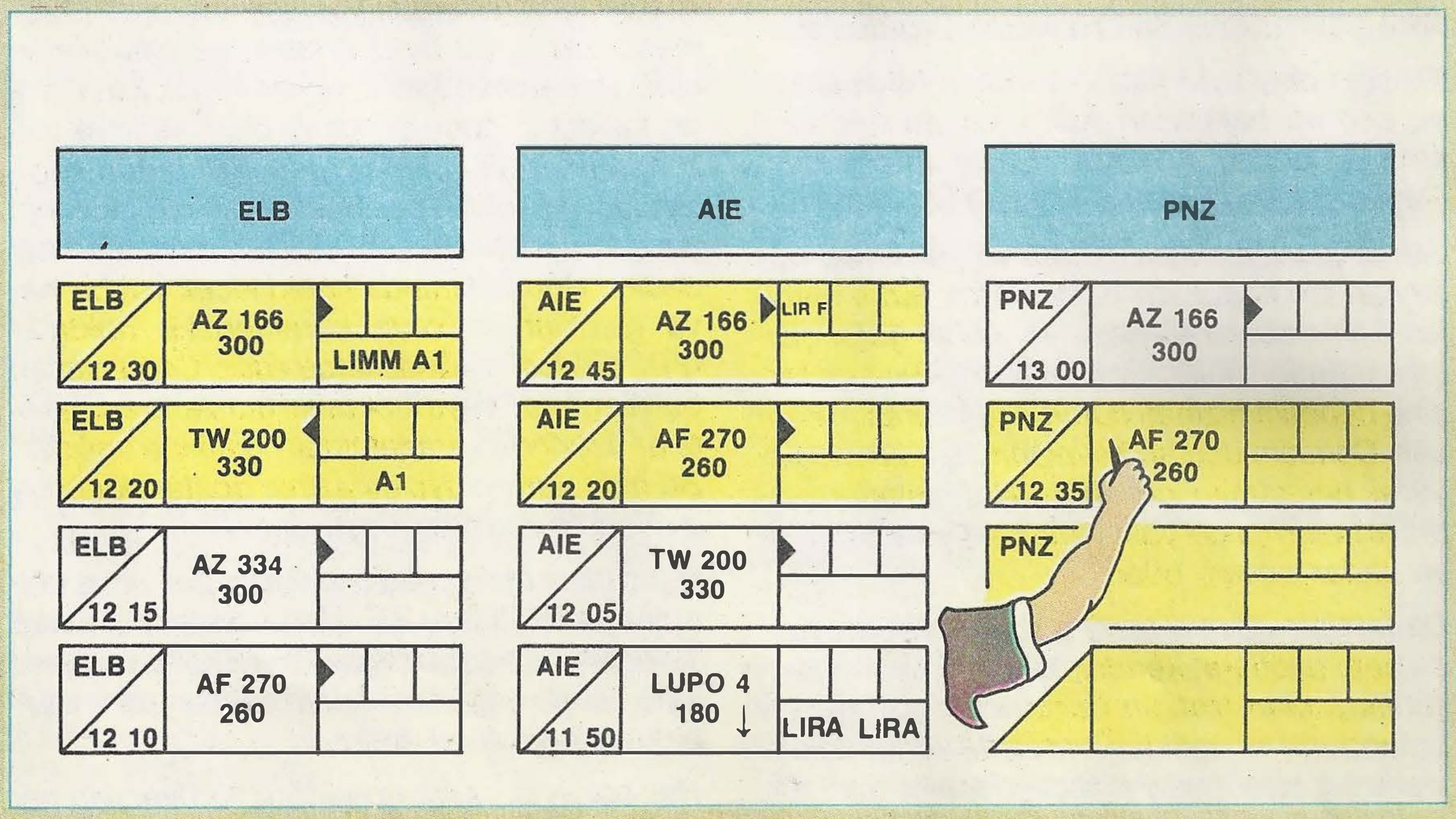
Bir uçağın iki uçuş bölgesi arasındaki sınırı aştığı an, uygulama açısından en nazikan dır. Denetçi uçağın öngörülen güzergâhı ile gerçek konumu arasında zaman ayarlamasını yapmak için striplerin tahmini saatlerine göre yapılması gereken düzeltmeyi haberleşme sayesinde kestirebilir. FIX'lerin üstünden uçuşları sırasında pilot ile yapılan değişik değişik haberleşmeler, öngörülene göre yer alabilecek olası gecikmeler veya erkencilikleri tahmin etmeyi mümkün kılar.

Hava trafik kontrolünde ilk önemli teknolojik uygulama 40'lı yıllarda olmuştur. Sözkonusu olan bu cihaz radardı. Bu uygulama radar yansımaları adı verilen bir dizi yanan noktaların görüntüsünü vererek o anda yürürlükte olan trafiğin görüntülenmesini sağladı. 1958 ile 1960 arasında SSR (Secondary Surveillance Radar- İkincil denetim radarı) denen bir ikinci radar kullanılmaya başlandı. Bu radar uçağa monte edilmiş ve transpondeur denilen bir alıcı-verici cihazı yoklar. İkincil denetim radarının transpondeur'dan aldığı sayısal yanıt, 64 bitlik bir kodla uçağın kimliğini ve uçuş yüksekliği ile ilgili bilgileri taşır.

Bir bilgisayar uçuşların ekranlar üzerindeki tesbitini yapmak için bu haberleşmeleri işler.

Otomatikleştirmenin birinci evresi etiketleme (labelling) denen bir yöntemdir. Labelling radardaki görüntüyle özellikle uçağın kodunu ve uçuş yüksekliğini bildiren bir etiketin eşleştirilmesi işlemine denir.

Üç FIX örneği.



Pilot ile ilk radyo kontrol bağlantısı yapıldıktan sonra, denetçi alıcı-verici cihaz aracılığıyla bilgisayara gönderilen imgeyi kimlik kodu, örneğin AZ 128 olan bir normal uçuş gösterimi ile değiştirir. Bu, uçağın radar bilgilerinden elde edilen konumu ile ilerleme fişleri (bir sonraki sayfadaki grafiğe bakınız) arasındakini karşılaştırmayı kolaylaştırır.

Bilgisayar sayesinde PPI (Konum planıg östericisi) ekrandaki, denetçiye gönderilecek imgeyi birçok fonksiyonlarla zenginleştirir. Örneğin bu fonksiyonlar şunlar olabilir:

- *Gökteki güzergahların şematik bir görünümü*

- *Hava sahasında sivil seyirlere yasak olan bölgeler, havaalanına yakın başlıca engebelerin yüksekliği gibi özel bölgelerin tümü ve belirleyici FIX'lerin tümü.*

- *Uçağın konacağı pistlere yakınlaşma güzergahlarının bir haritası.*

Ayrıca kullanıma hazır diğer fonksiyonları not edelim:

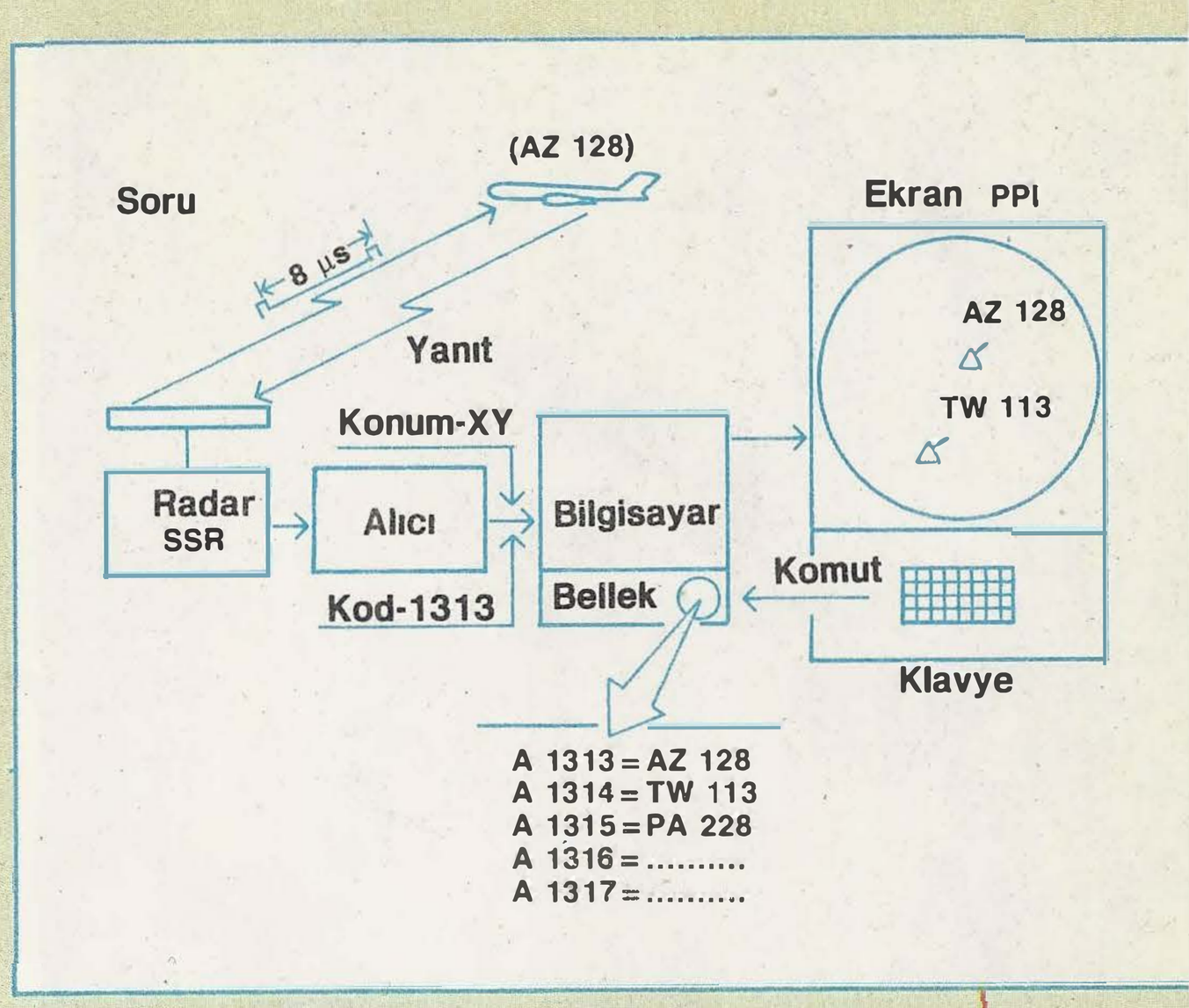
- *İki uçak arasındaki uzaklığın hesabı*
- *Ekrandaki gösterimin ölçek değişimi*

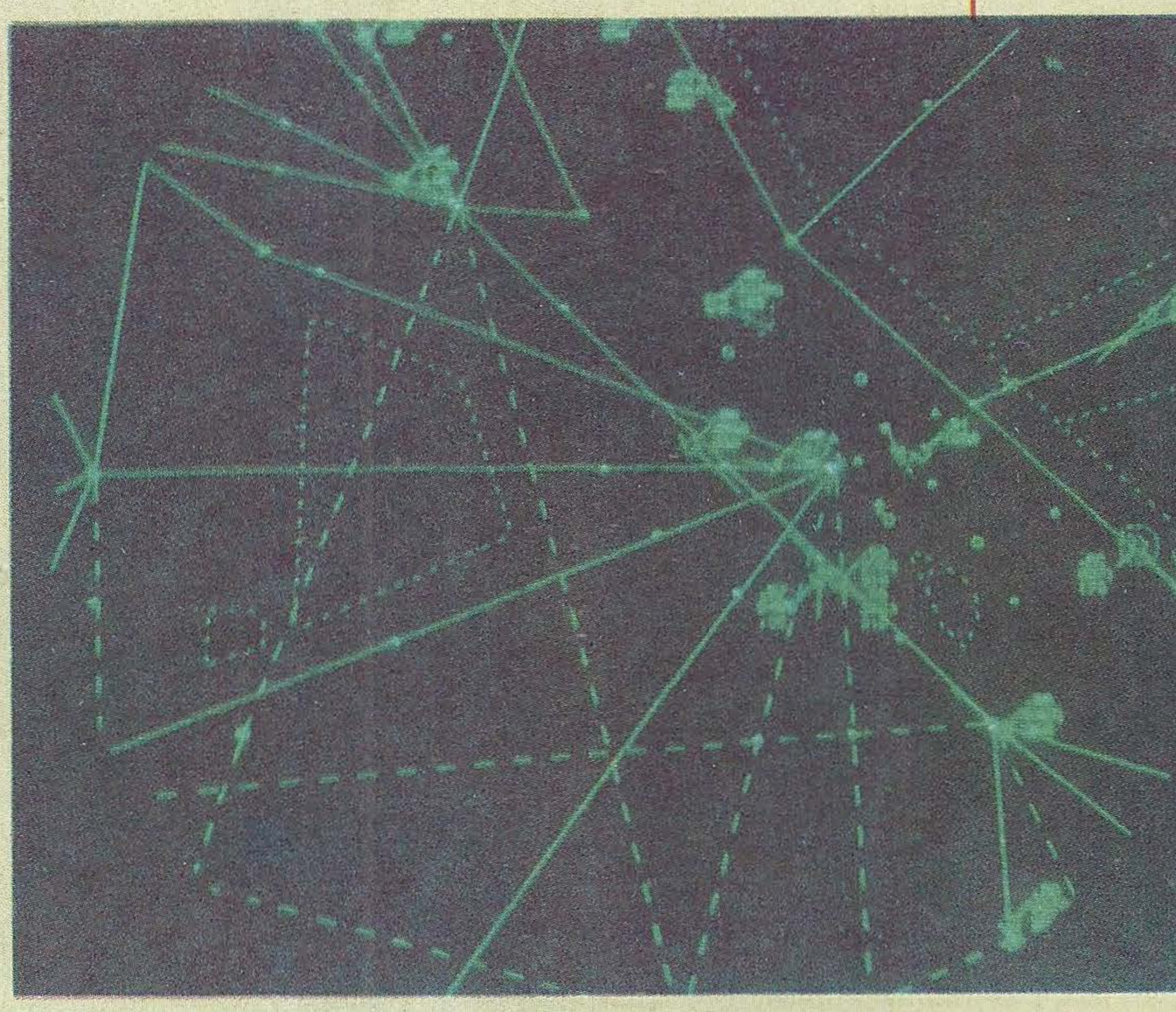
Radar bilgilerinin işlenmesini gerçekleştiren ***tracking*** *fonksiyonu otomatikleşme sürecinde yeni bir aşamadır. Gerçekten de genellikle yeni eksenler üzerinde bulunan veya aynı yönde dönen birincil ve ikincil radara yanıtlar ve transpondeur aracılığıyla iletilen bilgiler birinci radar için yalnızca yükseklik, ikinci radar için yalnızca uzaklığı verir.*

Tracking, bir hız değerini (şiddet, yön, taraf) çıkarsamak için bu verilerin zaman içinde korelasyonunu yapan bir fonksiyondur. Bu hız değeri daha sonra bir simge yardımıyla PPI'da gösterilir, ve ekranda bulunan uçağın kimlik kodu ile birleştirilir. Hava Trafik Kontrolünde rastlanan hataların büyük bir kısmı çok karmaşık bir fonksiyon olan tracking fonksiyonundan kaynaklanmaktadır.

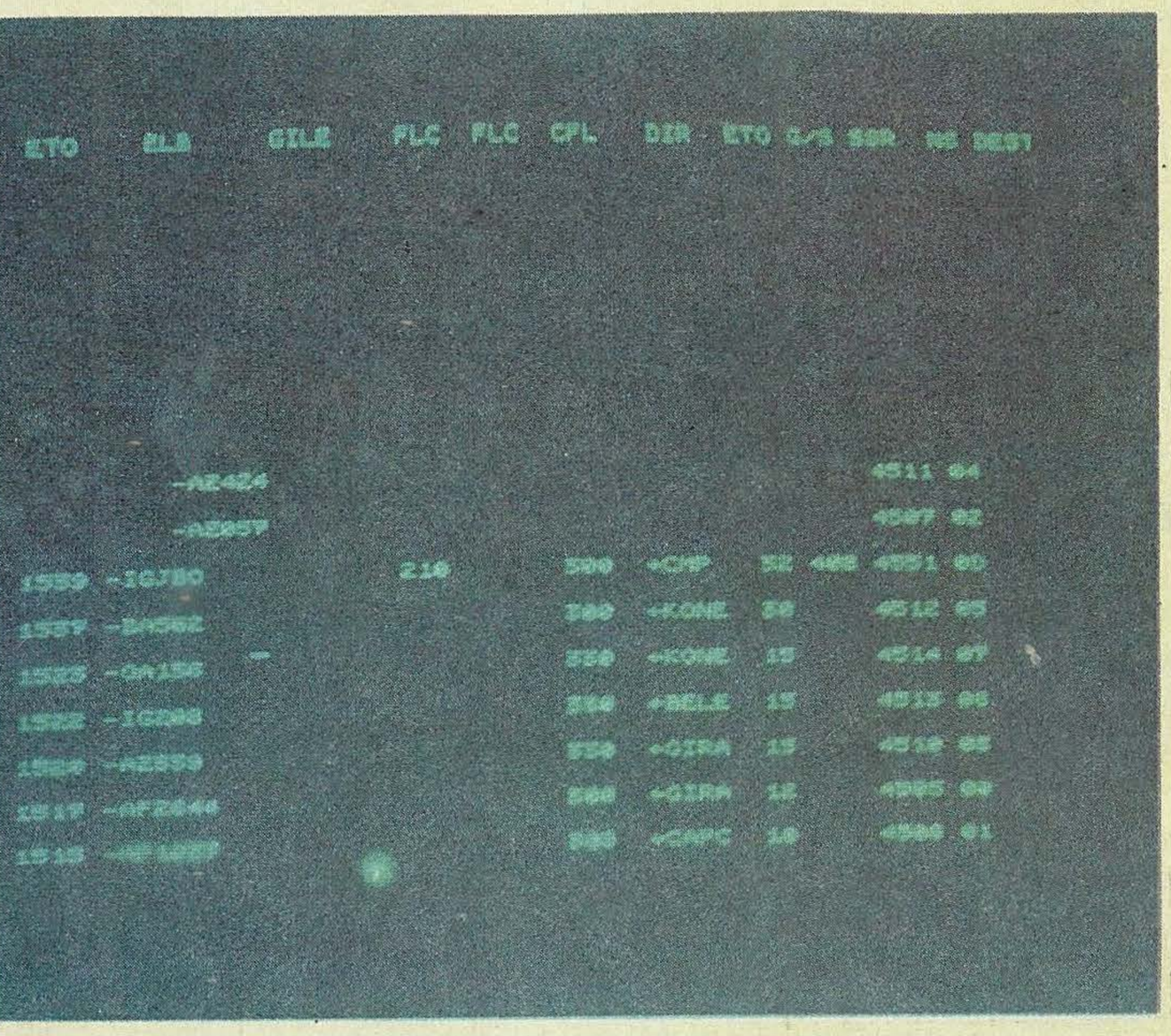
Ayrıca örneğin bir hava sahası uzak noktalara yerleştirilmiş bir çok radar tarafından tarandığında donanım ve programlarla ilgili hatalar da olabilmektedir. Eğer değişik radarlardan kaynaklanan bilgiler yanlış değilse, bir özümleme yapabilmek için geriye bu verileri zaman içinde aynı ölçeğe indirgeme zorluğu kalır.

Uçağın konumu ve hızı bir kez hesaplandı mı, çıkış verileri ya PPI üzerinde görüntülenir ya da çelişki çözümü adlı çok ilginç bir işlem için giriş olarak kullanılır.





Bir uçağın tespiti ve video üzerindeki tracking fonksiyonu.



Gerçekten de bir uçağın sinematik verilerinden yola çıkarak güzergahı ile ilgili, tahminler yürütmek olanaklıdır. Böylelikle başka bir uçakla aynı anda belirli bir noktada bulunması tehlikesi kestirilebilir. Sözkonusu tehlikeli durum, bir alarmın çalışmasına ve denetçinin acil müdahalesine yol açar.

Denetimin bu aşamasının makineleşmesi denetçilere çok büyük bir yardım sağlar ve bu sayede denetçiler güvenliği önemli ölçüde artırır.

Denetimlerden geçtikten sonra uçaklar 3 dakikadan az aralıklarla uçarlar. 3 dakikadan az olan aralıklar uçağın yere konması aşamasında trafiği hafifletmeye ve ***holding*** *adı verilen yükseklikte oluşan bekleme kuyruklarını ortadan kaldırmaya yarar. Havada oluşan bu kuyruklar havayolları şirketleri için pahalıya malolur, örneğin bir Boeing 747'nin havada bir saat bekleme tutarı 10.000.000 T.L'ye ulaşabilir.*

Hava denetçilerinin görevini hafifletmek amacıyla başka hizmetler de makine aracılığıyla yapılmaktadır. Bu durum ilerleme fişlerinin

Üstte belirli bir FIX'e göre belirlenmiş striplerin ekran üzerinde görüntülenmesi. Altta Roma-Ciampino'da bir kontrol konsolu.



baskısı ve özellikle uçuş planlarının yorumlanması için sözkonusudur. Demek ki sisteme bilgilerin şifresini çözmeyi kolaylaştıracak veri tabanları ve fonksiyonlar eklemek gereklidir.

Bellibaşlı veri tabanlarından bir tanesi hava trafiğinin değişik bölgelerdeki tam bir coğrafi betimlemesini içerir. Bu betimleme bütün hava yollarını, isimleri ve koordinatları ile belirtilmiş bütün FIX'leri, pistlere yaklaşma güzergahlarını olduğu kadar yerden kalkış güzergahlarını, pistlerin parametrelerini, yolun değişik bölümlerinde trafikle ilgili özellikleri, sınır üzerindeki uçuş bölgelerini biraraya getirir.

Getirilen yeni düzenlemeleri, hizmete sunulan yeni havayollarını, vb... hesaba katabilmek için bu bilgilerin kolaylıkla değiştirilebilir olmaları gerekir.

Bir başka veri tabanı, öngörülen yolu ve tarife saatlerini 6 ay boyunca kaydeden evraklardır (sözkonusu RPL, Repetitive Plan- Tekrarlanan Plan'dır). Bu tahminler denetlenecek olan günlük uçuşların % 50'sini oluştururlar. Uçuş bölgesine uçakların girişini saptayıp haber vermek için veri tabanı genellikle her saat olmak üzere düzenli aralıklarla taranır.

Geriye kalan uçuşların % 50'sinin, yani RPL'nin dışında kalanların planları, bilgisayara bağlanmış olan AFTN şebekesi aracılığıyla temin edilir. Sistem teleksleri bağlar, bunları yorumlar ve olası hataları düzeltir. Gereken değişikliklerin yapılmasını sağlamak için düzeltilmemiş olan mesajlar kullanıcıya ekran üzerinde sunulur (sayfa 1517'deki fotoğrafa bakınız).

RPL'ler ve AFTN uçuş planları ikinci bir fonkiyonun girişlerine hizmet ederler. Bu ikinci fonksiyon, rotanın otomatik olarak yeniden oluşturulmasıdır.

Bu fonksiyon, coğrafi verilerden çıkarak, uçuş planının yolunun tamamından bölgedeki denetçiyi ilgilendiren bölümü seçmek ve bu bölümü üzerinden uçulması gereken bir dizi FIX'e dönüştürmek zorundadır. Uçakların hızları ve özellikleri gibi daha az önem taşıyan diğer veri tabanları, uçuş planlarında gösterilen saatlerle ilgili bilgilerden hareketle her bir FIX için, bunun üzerinden yapılacağı bilinebilen uçuşların zamanını hesaplamayı sağlar. Bütün yeni veriler otomatik olarak çıkış birimlerine gönderilir, böylelikle ilerleme fişlerinin tüm bir gün boyunca üretimi yapılmış olur.



Roma-Ciampino'da Hava Trafik Kontrolü merkez işletim salonu.

Otomatikleşmenin üçüncü aşaması ise daha önce gördüğümüz iki makineleşme düzeyini tek bir sistemde birleştirmekle oluşur. Bilgiler aynı anda doğrudan doğruya radarla ve uçuş planından çıkarılan yolların aracılığıyla tahmini şekilde alınır. Tahmini bilgilerin elde edilmesi ancak çok büyük bir hızla hesaplar yapabilen büyük bir bilgisayar el altında bulunduğunda olanaklıdır.

Teorik ve gerçek FIX'lerin uçuş zamanı ve yol tahminleri gibi önemli bilgiler, denetçilere EDD (Electronic Data Display-Verilerin Elektronik Sunumu) adı verilen ve PPI'nın görüntülerini tamamlanması için radar ekranın yanına yerleştirilen özel bir ekranda sunulur (sayfa 1517'deki grafiğe bakınız).

Mevcut rotaları içeren veriler tabanını yeniden oluşturulan rotayla karşılaştırarak uçağın gerçek konumunu, bulunması gereken konum ile karşılaştırmak olanaklıdır. Bu da, dolayısıyla tutarsızlık durumunda bir alarmı harekete geçirmeye yarar.

Daha sonra tracking ve hızın girişleri, EDD'ler üzerinde uçuş planından çıkarılan verilerin yanında sayısal olarak ve bir bütün şeklinde görüntülenir ki, deneticiye verilen bilgiler en büyük sayıda olsun.

Makineleşmenin bu üçüncü aşamasında sis-

teme soru sormayı sağlayan bir komutlar dizisi çok geniş bir uygulama sahası yaratmaktadır. Böylelikle denetçi uçuş planlarında yapılan yenilikleri hesaba katmak, belirli yolların yeni bir hesabını istemek ve ilerleme fişlerini yeniden basmak için veri tabanında değişiklik yapabilir.

Makineleşmenin getirdiği güvenlik, bir çok ülkeyi CTA sistemini benimsemeye yöneltti.

Avrupa'da hava seyrinin denetimine yeni bir boyut kazandıran şey, değişik ülkelerin bilgisayarları arasında yapılan bilgi alışverişidir.

OACI veya ICA (International Civil Aviation Organization- Uluslararası Sivil Havacılık Örgütü) bilgisayarlar arasında mesajların alışverişini bir ilkeye bağlamak amacıyla şimdiden protokoller tasarlanmıştır. Amaç, eski AFTN'nin yerini alacak uzmanlaşmış bir bilgi-işlem ve haberleşme şebekesi kurmak.

Bu karşılıklı bağlantılar otomatik seyir düzenleme kavramını Avrupa Trafiğinin tümüne yaygınlaştırabilir. Bunu da artık tek bir FIX içinde çözüm aramakla (ki bu özellikle hava durumundan kaynaklanan nedenlerle çoğu zaman pek kolay olmamaktadır) yetinmeyip olası yol değişikliklerin de başta öngörülmüş olan rotaların dışındaki rotalara başvurarak gerçekleştirirler.

Bu iddialı proje, şimdiye kadar, yalnızca Amerika Birleşik Devletleri'nde gerçekleştirilebildi. Ülkedeki Federal Aviation Administration adında tek örgüt 51 eyaletin denetiminin eşgüdümünü sağlamaktadır.

Sistemin resimlendirilmesi bakımından radar otomatikleşmesi, büyük hacimli yığın belleği olmayan küçük bilgisayarlar (16 bitlikler yeterli olabilir) gerektirmektedir.

Radar verileri daha önceden, bir kez uzmanlaşmış donanımla hesaplandığından, bilgisayarın yaptığı iş gerçekten de kısıtlı sayıda verilerin hesaplamalarıyla sınırlıdır. Uçuş planlarının ve tekrarlanan planların (RPL) işlenmesi de büyük bilgisayarlar gerektirmez, ancak bunlar coğrafi bilgileri, uçakların özelliklerini ve uçuş planlarını stoklamak için önemli veri tabanlarına ihtiyaç duyarlar.

Eğer RPL'lerin çözümlenmesi ile ilerleme fişlerinin bilgiyazımı radar verilerinin işlenmesinden ayrı olarak yapılırsa, verilerin tümünü bir diskte stoklamak olanaklıdır, çünkü G/Ç'in zamanlaması hayati bir öneme sahip değildir.

Buna karşılık biraraya getirilen denetim düzeyleri, hesaplama hızını ve bellek kapasitesini arttırmak için daha karmaşık sistemlere gerek göstermektedir.

Bu sistemlerin çizimleri devirli olarak ele almaları ve bunları radarların gidiş geliş zamanları ile ayarlı şekilde (gidiş geliş başına 10 saniyeden az) işlemeleri gerektiğine göre, gerçek zamanda veri işlemenin tipik bir uygulamasına gerek duyarlar.

Bunun gerçekleşmesi için bilgisayarların verimliliklerinin 1 Mips'e (saniyede bir milyon komut) erişmesi gerekiyor.

G/Ç'lerin ölü zamanlarını azaltmak için veri tabanları, sistemin harekete geçişinde doğrudan doğruya belleğe yüklenir; bu merkezi bellekte 1'den 2 bayta kadar bir yer kaplasa da yine yapılır. Geçmişteki on yıllık dönemler süresince bu sistemlerin gelişiiminde gözlenen gecikmeyi de zaten bellekler için harcanan gider ile açıklanabilir.

Bugün trafik üzerine dinamik verilerle, denetçi ile tracking'in diyalogunu içeren veri tabanın merkezi bir bilgisayar tarafından yönetilmesine doğru yöneliniyor.

Uçuş planlarının, ilerleme fişlerinin ve PPI'lar ile EDD'ler üzerindeki görüntülemenin yönetimini bir küçük bilgisayarlar şebekesi hazırlamaktadır. Herzaman olasılık dahilinde olan, makinenin durma tehlikesini önlemek için sistemin ögeleri iki katına çıkartılmıştır. Sistemin hata türleri üç çeşittir; kaynaktan gelen hatalar, malzemeden kaynaklanan hatalar ve mantık hataları.

Birinci türden hatalar elektrik kesintilerini kapsar. Bu durumda yedek bataryalar ile takviye edilmiş jeneratörlere başvurulur ki, bunlar da çifter çifter kullanılmaktadır. Yedek bataryalar kaynak sektöründeki kesinti ile jeneratörlerin harekete geçmeleri için aradan geçen ölü zamanı doldururlar.

Gerçekten de saniyenin onda biri süren bir kesinti, bellekteki bilgilerin kaybedilmesine yeterli olmaktadır.

İkinci türden hata ise bilgisayarla ilgilidir. Yeniden harekete geçiş genellikle daha uzun bir zaman alır (yaklaşık olarak bir dakika).

Denetçiler tarafından girişi yapılan herhangi bir bilginin kaybolmaması için, her komutun gönderilişi sırasında merkezi bellekte bulunan

veri tabanlarının disket üzerinde kopyası yapılır.

Birincil bir bilgisayar arızalandığı zaman, aynı disketlere ulaşabilen ikincil bir bilgisayar hata durumundan haberdar edilir ve harekete geçişi sırasında yürürlükte olan trafikle ilgili tüm bilgileri yükler. Hatalarla ilgili bilgiler de aynı şekilde bir diskete kaydedilir.

Makine ile hava denetimi sisteminin İtalya'da oturması 60'lı yılların sonunda işlerlik kazandı. Bu tarih, ordunun, Selenia ile IBM'in işbirliği sonucu ortaya çıkan ATCAS sistemini (Air Trafic Control Automatic System-Otomatik Hava Trafiği Denetim Sistemi) ısmarlanmasına rastlar. Bu sistem Roma Hava sahasının denetimini sağlayacaktır.

Gelişim programı 1975-85 yılları arasındaki 10 yıllık süre içinde Hava Trafiği Kontrolünün gerek görülen her düzeyde otomatikleşmesini sağlamasını öngörüyordu. 1980 yılından sonra Roma 'Datamat Ingegneria dei Sistemi' yeni otomatikleşme düzeylerini tasarlamak amacı ile ortaklığa katıldı.

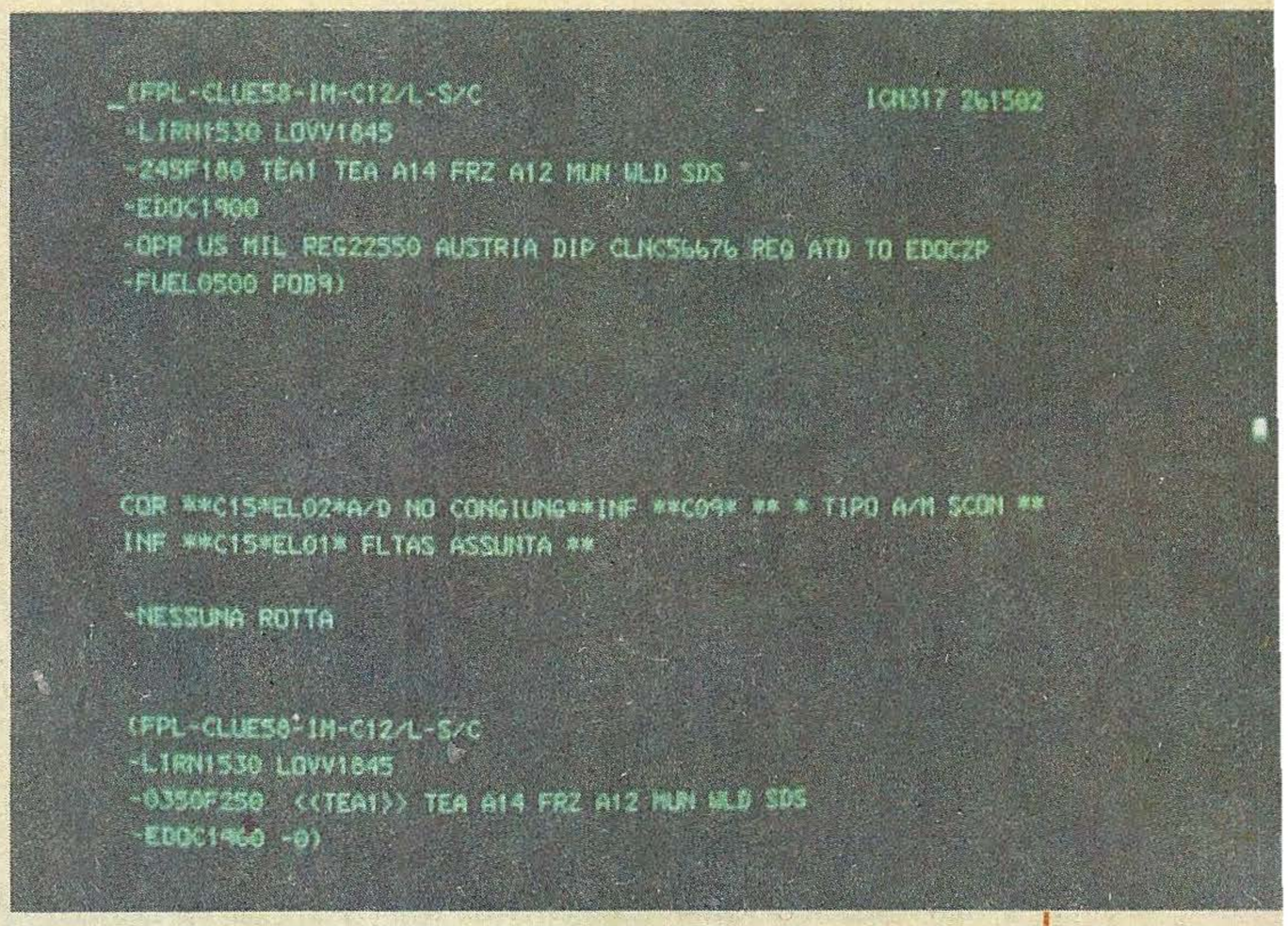
Son şekliyle sistem, IBM 3033 türünden büyük bir merkezi bilgisayarın kullanımını öngörüyor. IBM 3033'in işlem hızı ise 4,5 mips olup iki GP-16, bir CDG (Selenia) ve ''köle'' görevi gören bir ikinci IBM 3033'e bağlıdır (yandaki grafiğe bakınız).

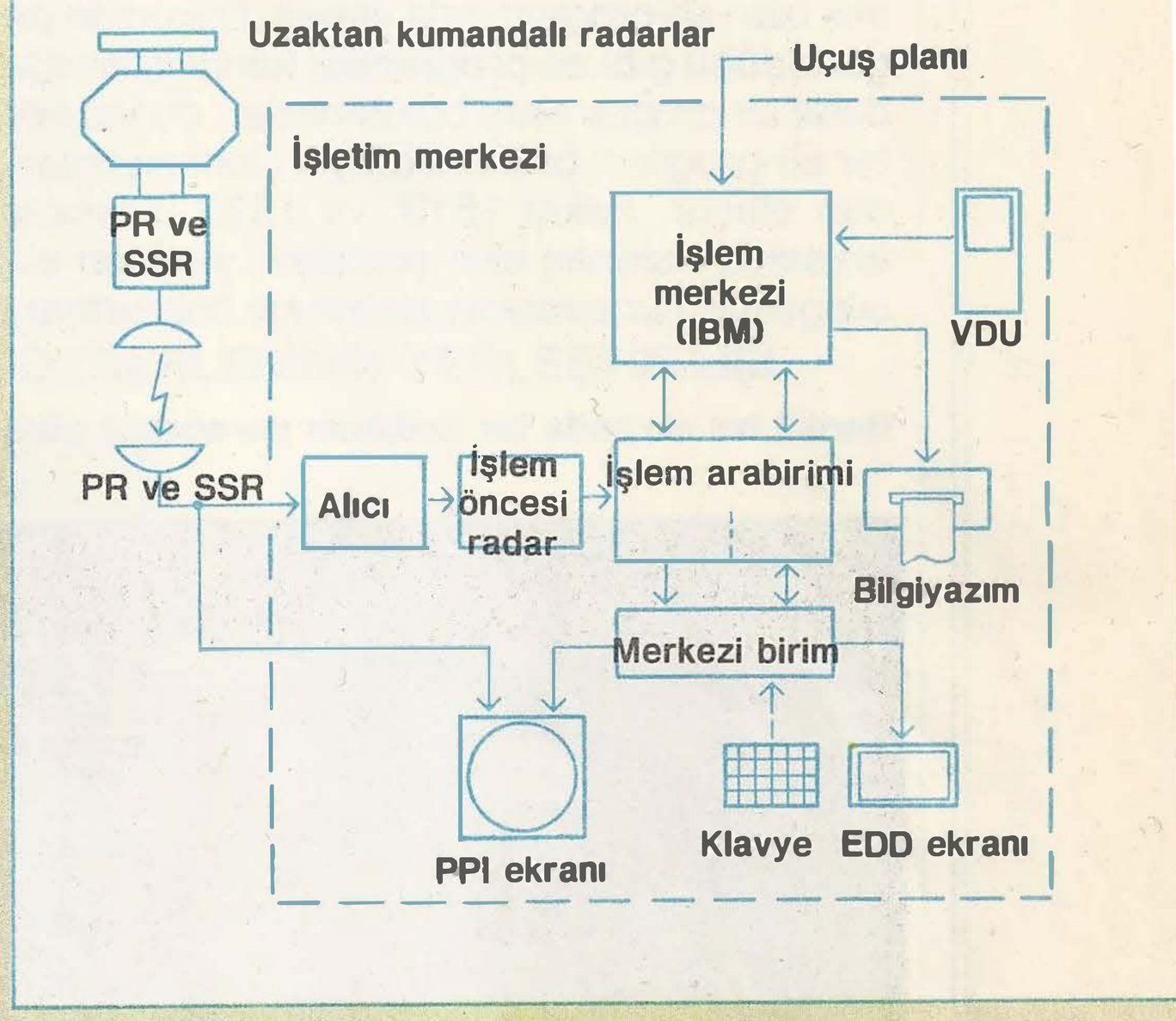
Merkezi bilgisayar tarafından yönetilen başlıca çevresel gereçler şunlardır:

- Her biri gelmekte olan durum ile ''tekrarlanan planların'' kaydedilmesine yarayan 400 baytlık disk birimleri
- Çeşitli denetçiler ile sistem arasında yer alan her türlü haberleşmenin sayısal kaydına yarayan manyetik şerit sürücüleri
- AFTN şebekesinden alınan telekslerin bir kopyasını saklamaya ve ilerleme fişlerini basmayı sağlayan bir çıkış gereci. - Teleks mesajlarının düzeltimi ve uçuş planlarının veya tekrarlanan planların girişlerinin video birimleri

Ayrıca uçuş planları ile ilgili mesajların tümünü almak amacıyla bilgisayarın doğrudan doğruya AFTN şebekesine bağlantısı vardır.

Denetçilerin yararlandığı PPI/EDD görüntüleme birimlerinin yönetimi işlenmiş verileri merkezi bilgisayardan alan GP-16'lar tarafından yürütülüyor. Küçük bilgisayarlar normal klavyeleri veya hızlı giriş tertibatlarını yöneterek





AFTN üzerinden alınan bir mesajın ekran üzerinde gösterimi. Altta mesajın alınmış şeması.

makineyi kullananlar ile baş makinist arasında arabirim işlevi de görürler.

Makineden kaynaklansın, mantıki olsun herhangi bir hata durumunda, ikinci IBM 3033'ün harekete geçmesi için gereken bir dakika içersinde verilerin etiketlenmesi işleminden CD-c7- sorumludur.

Bu olgu gösterim alt-programını karmaşıklaştırır, çünkü bu histogramın sütunlarının gösterilişi için farklı biçimlerin öngörülmesini zorunlu kılar. Renkli bir ekranda her bir gösterme biçimi farklı bir renkle bağdaştırılabilir ve farklı histogramlar böylelikle renklendirilebilir.

Renkli ekran farklı görüntüleme modlarını farklı renklerle ifade etmeyi sağlar. Histogramların tümü, içi dolu sütunlar şeklinde çizilir. Farklılık yalnızca sütunların renklerindedir.

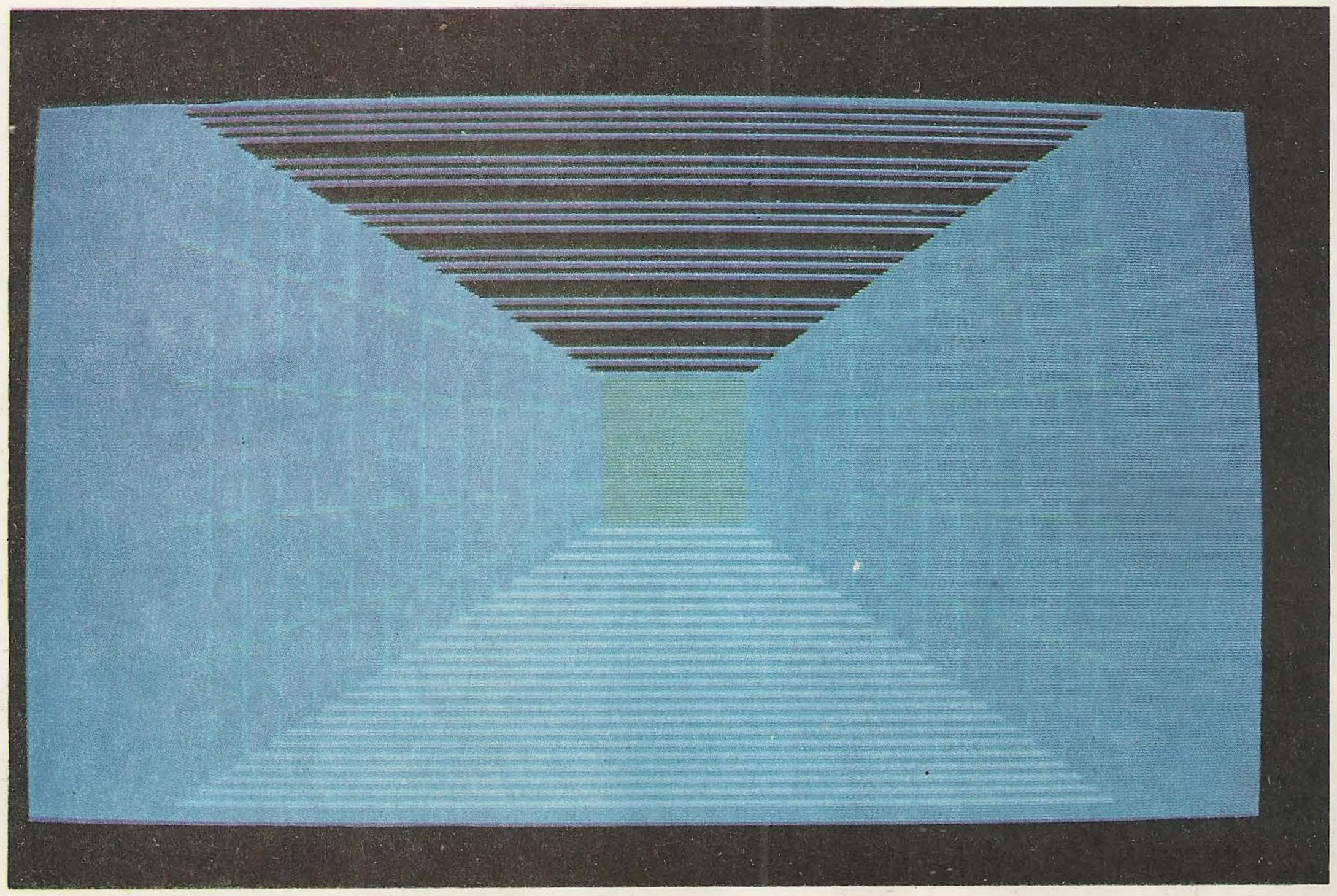
Tek renkli ekranlarda ise sütunların içini farklı şekillerle doldurmak gerekir.

Program, birincisi verilerin girilmesine, ikincisi ise bunların gösterimine ilişkin olmak üzere iki bölüme ayrılmıştır. Tüm uyarlamalarda kullanılabilmesini sağlamak için veri giriş bölümü ana programın içine oturtulmuştur.

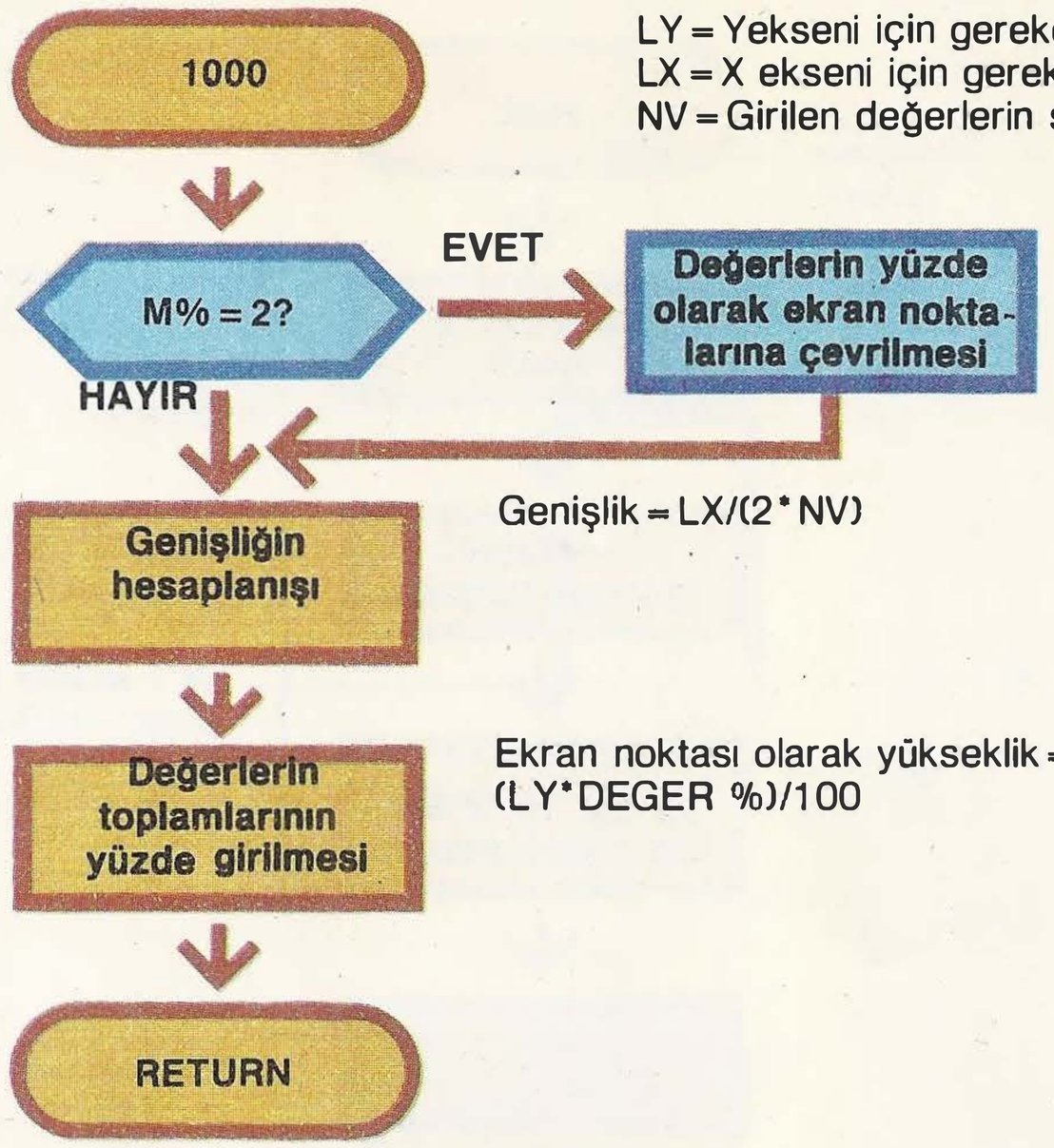
Gösterim, her biri özel bir görevde uzmanlaşmış olan alt-programlarla yapılır. Dökümde de görüldüğü gibi alt-programlar kendi içlerinde daha alt-programlara bölünmezler, çünkü her bir alt-program birbirini izleyen fonksiyonlardan oluşur. Fakat 1519. ve 1522. sayfalar arasında bulunan akış şemaları, yeniden alışılagelmiş numaralama sistemine başvurmaktadır. Programa daha yapısal bir şekil vermek için 1000 alt-programının uygulamasından önce ana programı kesmek ve buna çağrıları katarak, her bir öbeği (alt-programı) bir RETURN komutu ile tamamlamak gerekmektedir. Başka bir deyişle 380. satırdan sonra GOSUB 1000, GOSUB 2000.... komutları yer almakta, oysa 1180. satır (aynı şekilde 2085. satır v.d) RETURN ile devam etmektedir. Bu yüzden her bir öbek bir alt-program olarak kabul edilebilir.

Program, bunun yanısıra alfabetik ve sayısal karakterlerin gösterimi için bir alt-programa sahiptir. Bunlarla ilgili veriler 60. ve 65. satırlarda başlatılır. Bundan sonra verilerin girilmesi aşaması gelir. Bu aşama sırasında bir geçerlilik denetimi yapılır. Değerlerin yüzde olarak girilmesi yolu seçilmişse, girilen değerlerin toplamı 100'ü (% 100) geçemez (210. satır; 6000 alt-programının sadece hata mesajının görünür kılınması işine yaradığını kaydediniz). Programdan çıkış, öngörülmüş olan en büyük veri sayısına (NM = 20, 20. satır) eşit bir sayının veya negatif bir değerin girilmesi ile gerçekleştirilir. 20 verilik azami sınır, X ekseni üzerinde dağılımı yapılması gereken 20

Renkli bir ekranda bir mekânın perspektif görüntüsü.



BİR HİSTOGRAM İÇİN VERİLERİN İŞLENİŞİ

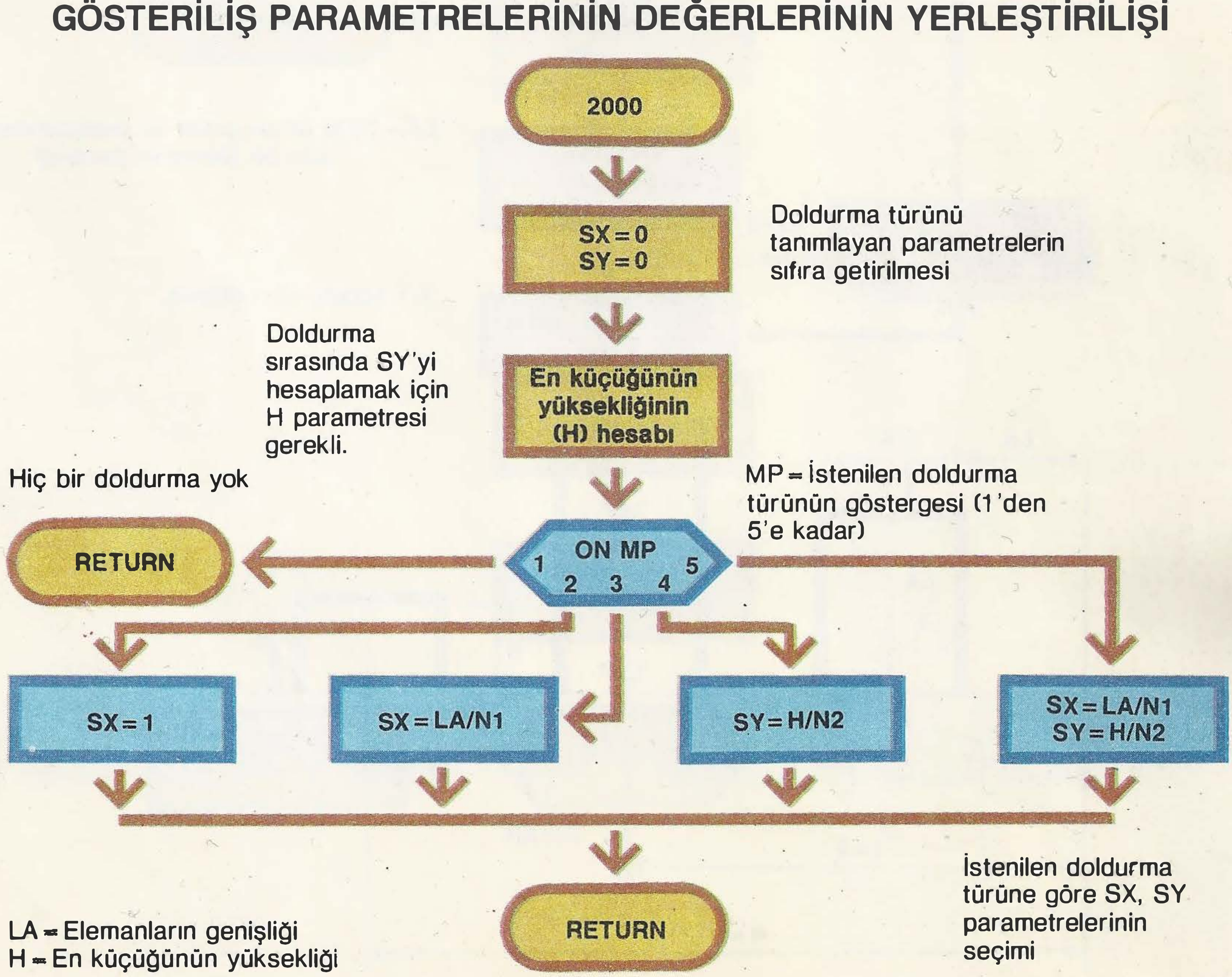


LY = Yekseni için gereken uzunluk
LX = X ekseni için gereken uzunluk
NV = Girilen değerlerin sayısı

Genişlik = LX/(2 * NV)

Ekran noktası olarak yükseklik = (LY*DEGER %)/100

GÖSTERİLİŞ PARAMETRELERİNİN DEĞERLERİNİN YERLEŞTİRİLİŞİ



Doldurma türünü tanımlayan parametrelerin sıfıra getirilmesi

Doldurma sırasında SY'yi hesaplamak için H parametresi gerekli.

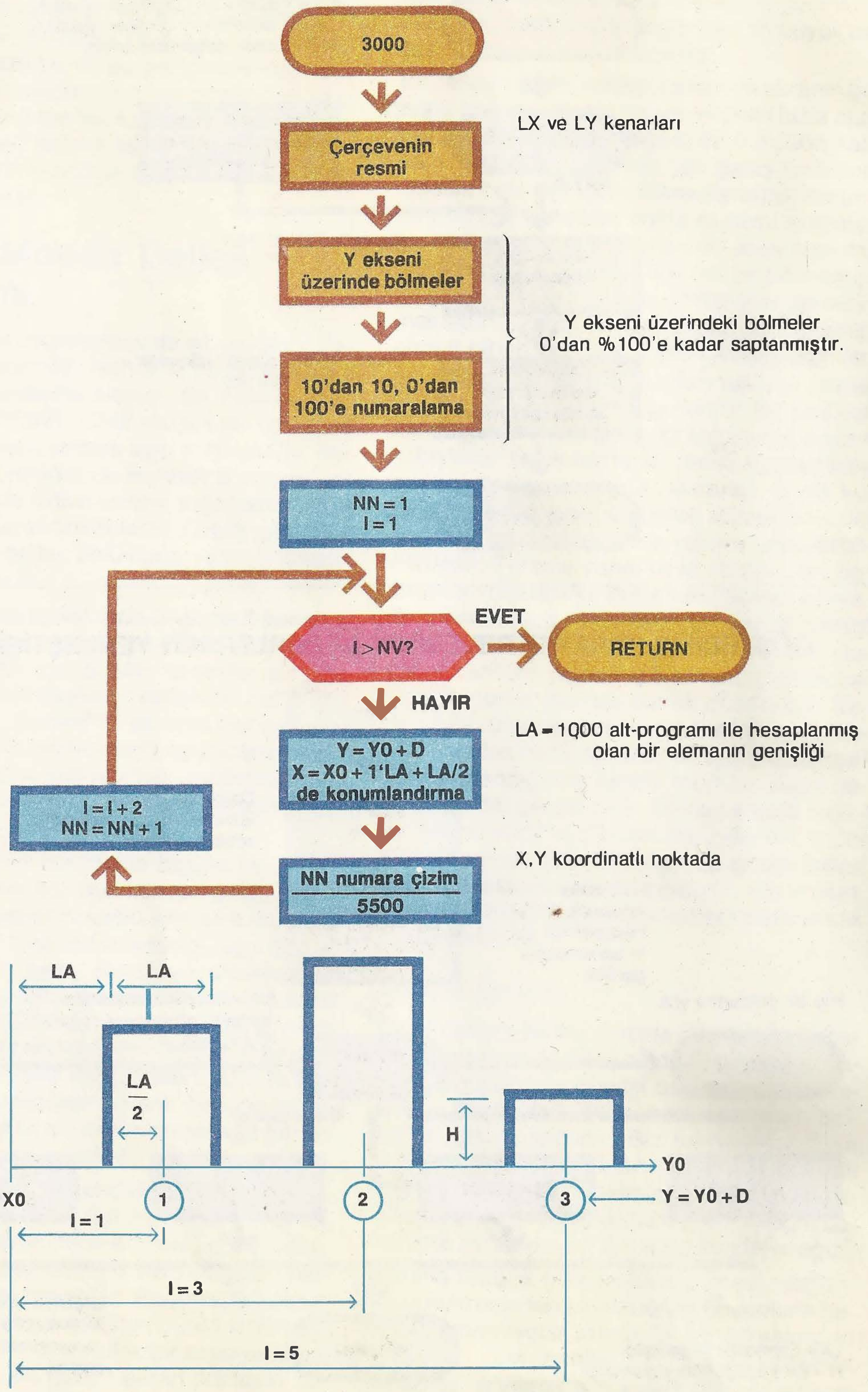
Hiç bir doldurma yok

MP = İstenilen doldurma türünün göstergesi (1'den 5'e kadar)

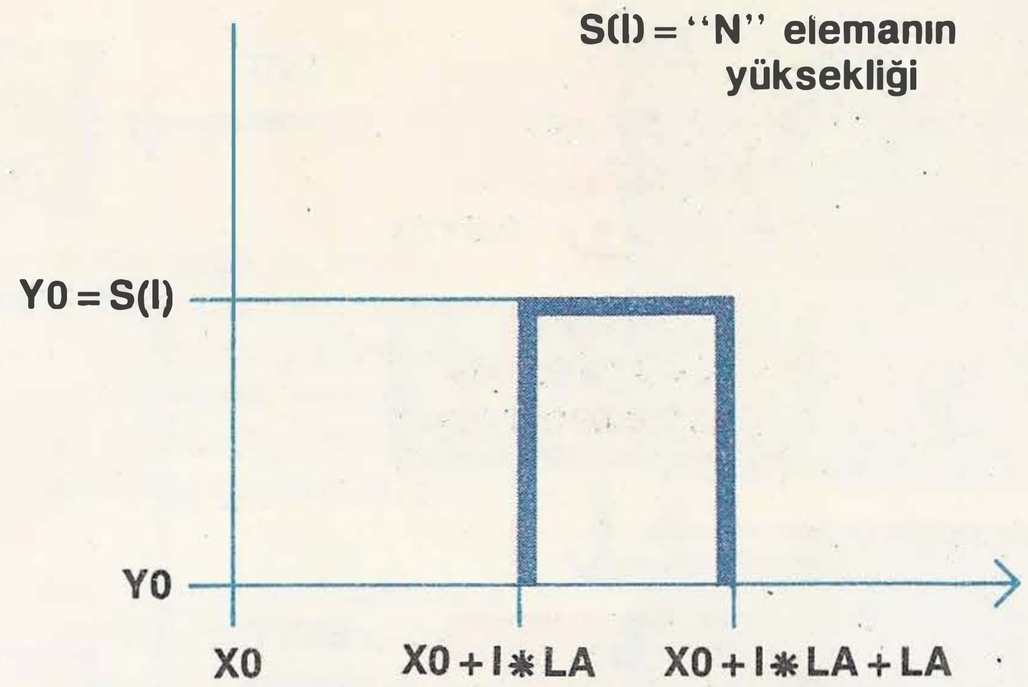
İstenilen doldurma türüne göre SX, SY parametrelerinin seçimi

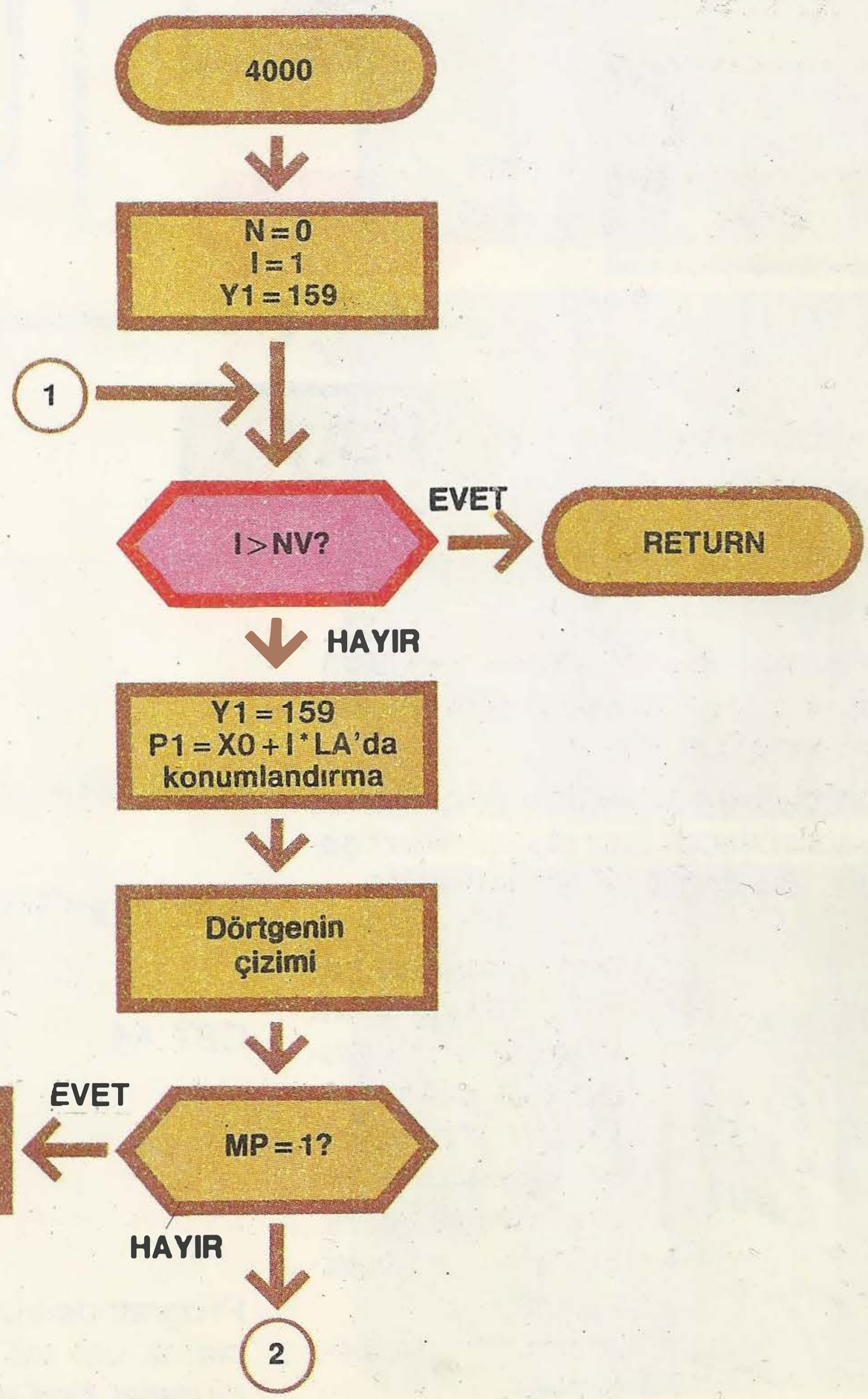
LA = Elemanların genişliği
H = En küçüğünün yüksekliği

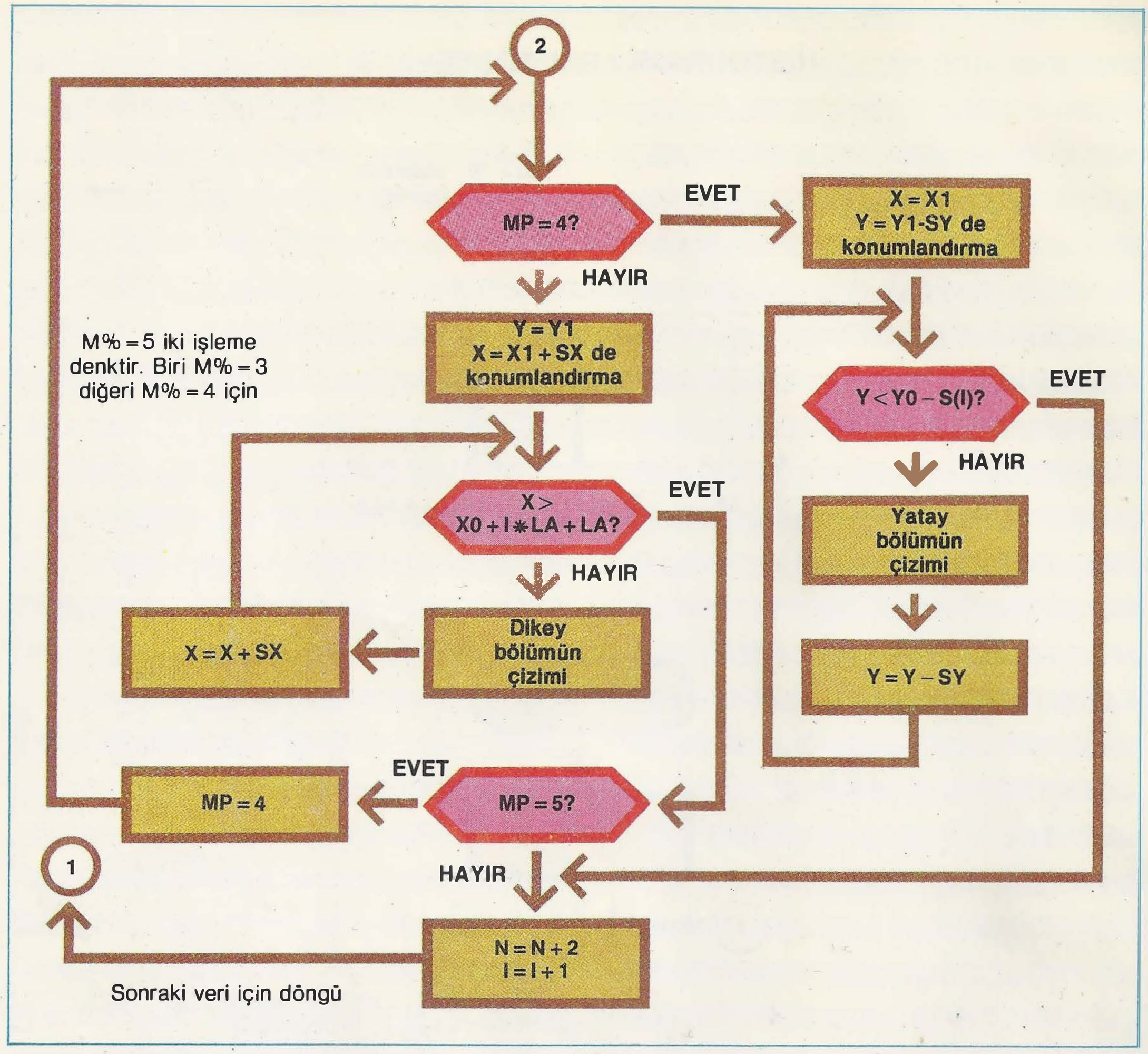
PENCERENİN GÖSTERİLİŞİ



HİSTOGRAMLARIN GÖSTERİLİŞİ







sütuna karşılık gelir. Bu sayı, kullanılan ekranın ölçülerine ve histogramların belirli şekillerde doldurulmasına bağlıdır.

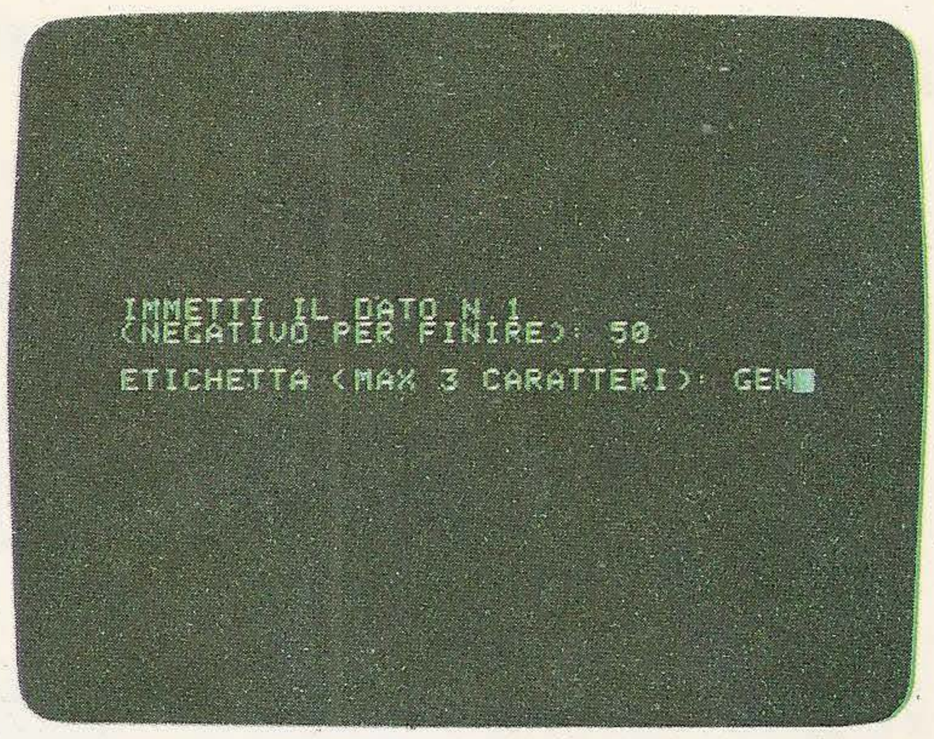
Bu programın başka makinelere geçirilmesi, komutlarda yapılabilecek bazı değişiklikleri gerektirmektedir. Bu değişiklikler şunlardır:

TEXT	Metin modunu belirtir. Programın başında grafik modu iptal eder.
HOME	Ekranın içeriğini siler ve imleci yukarıda sola yerleştirir.
VTAB n	İmleci n. satıra yerleştirir.
HTAB m	İmleci m. sütuna yerleştirir.
HGR2	Grafik modu yüksek kararlılıkta **(high resolution)** harekete geçirir.
HCOLOR = n	n rengindeki ''kalem''i harekete geçirir.
HPLOT X1,Y1 TO X2,Y2	X1,Y1 ve X2,Y2 koordinatlı noktaları birleştirir.
GET A$	Klavyeden girilmiş olan bir karakteri okur. INPUT$(1) komutunun dengidir.

Programda bütün konumlandırmalar referans olarak üst sol köşeyi alırlar. Değişik yönlendirmeler elde etmek için yerdeğişimlerinin işaretinde bir değişiklik yapmak gerekebilir.

HİSTOGRAMLARIN GÖSTERİLİŞİ

Histogramların gösteriliş programının işleyişinin bazı aşamaları

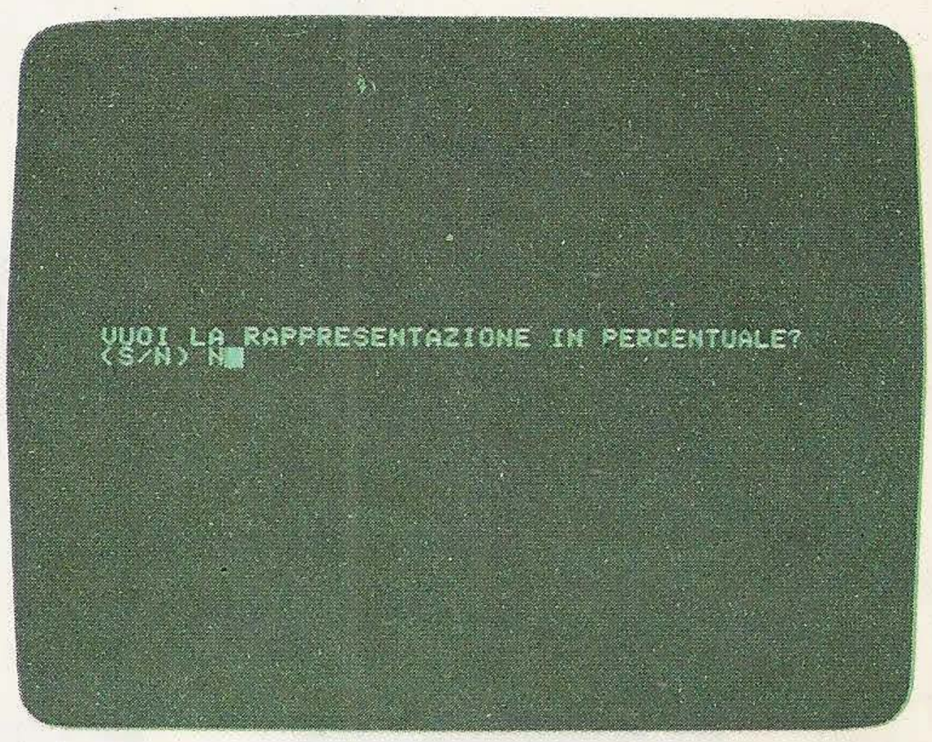


Yüzdeliklerle ifade edilmiş veriler değil de mutlak veriler gireceğini belirterek kullanıcının yanıtladığı 80. satırdaki sorudan sonra, program, gösterilecek olan verilerin girilmesi aşamasına geçti. Burada kullanıcı birinci veri olarak 50 değerini girdi. Verinin JAN etiketi ile işaretlenmesi gerekmektedir.

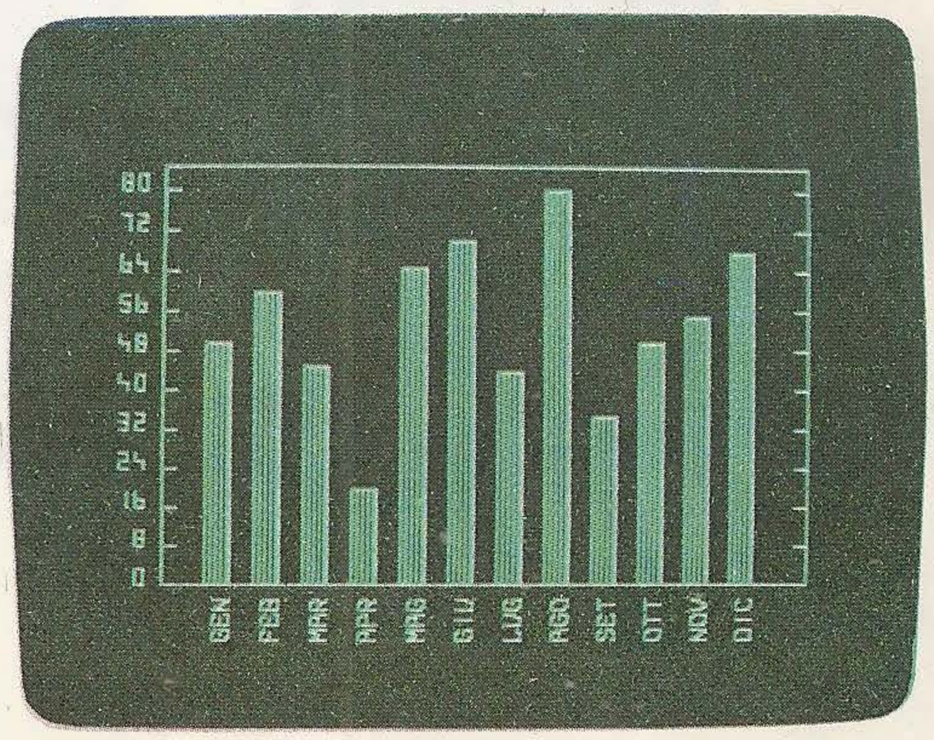
310 ila 360. satırlar, veri girişi aşamasının sonunda yandaki menünün gösterilmesini sağlar. Kullanıcı değerlerin dikey sütunlar halinde gösterilmesini seçmiş.



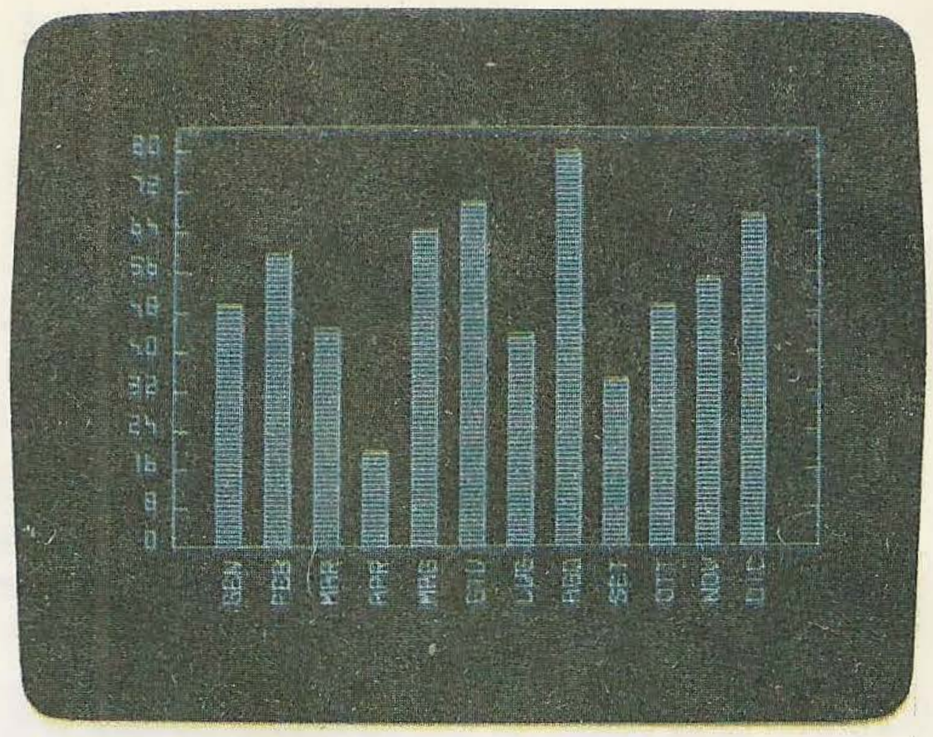
Bundan sonra program kullanıcıya verileri yüzdelik olarak göstermek isteyip istemediğini sorar.



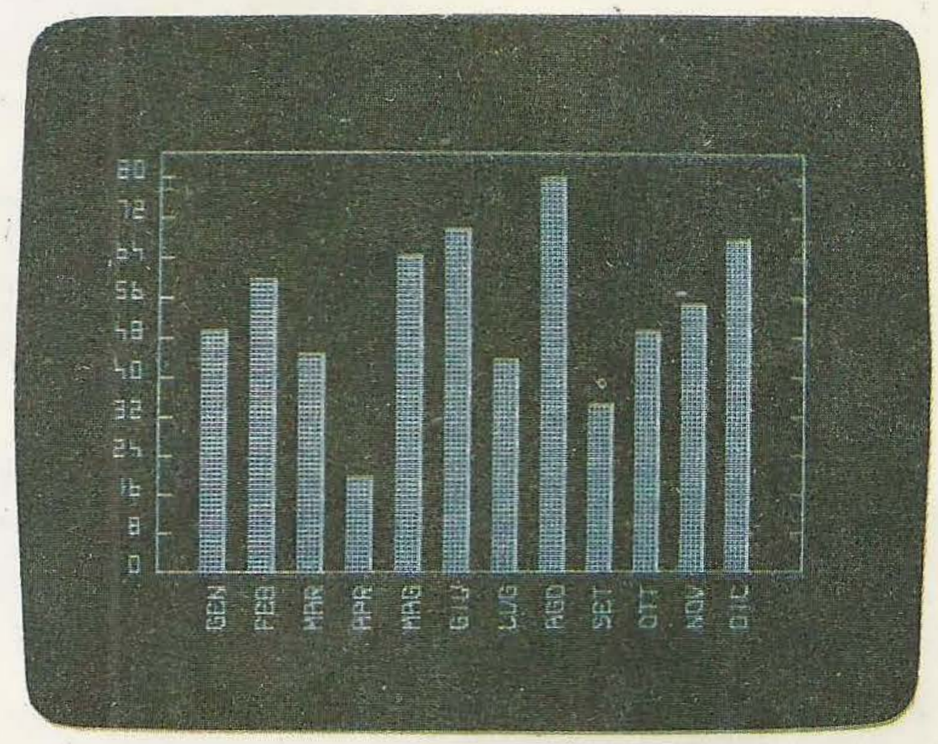
Program girilen mutlak değerlerin dikey satırlar halinde gösterimini canlandırdı.



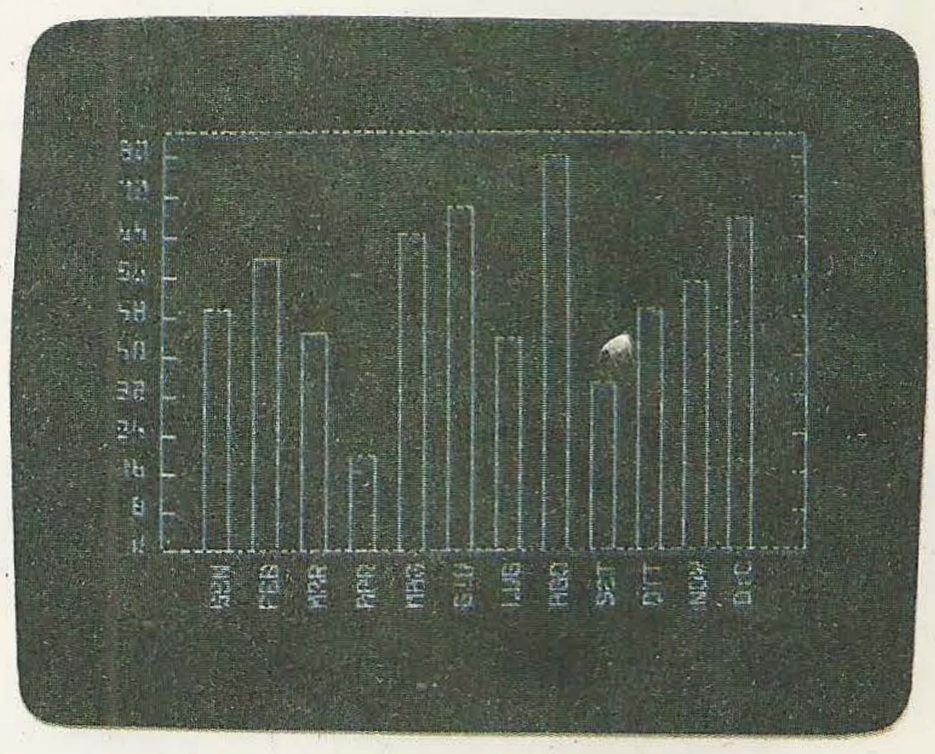
Yandaki ekran ve alttaki üç ekran farklı görüntüleme şekillerini gösteriyor.
Yatay satırlar



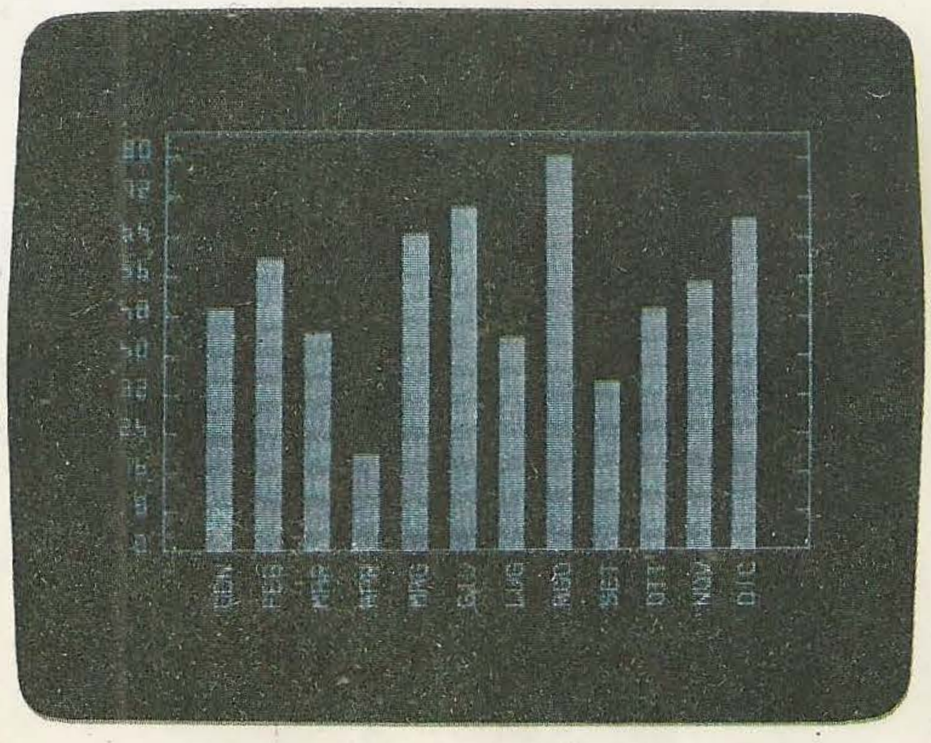
Kareli



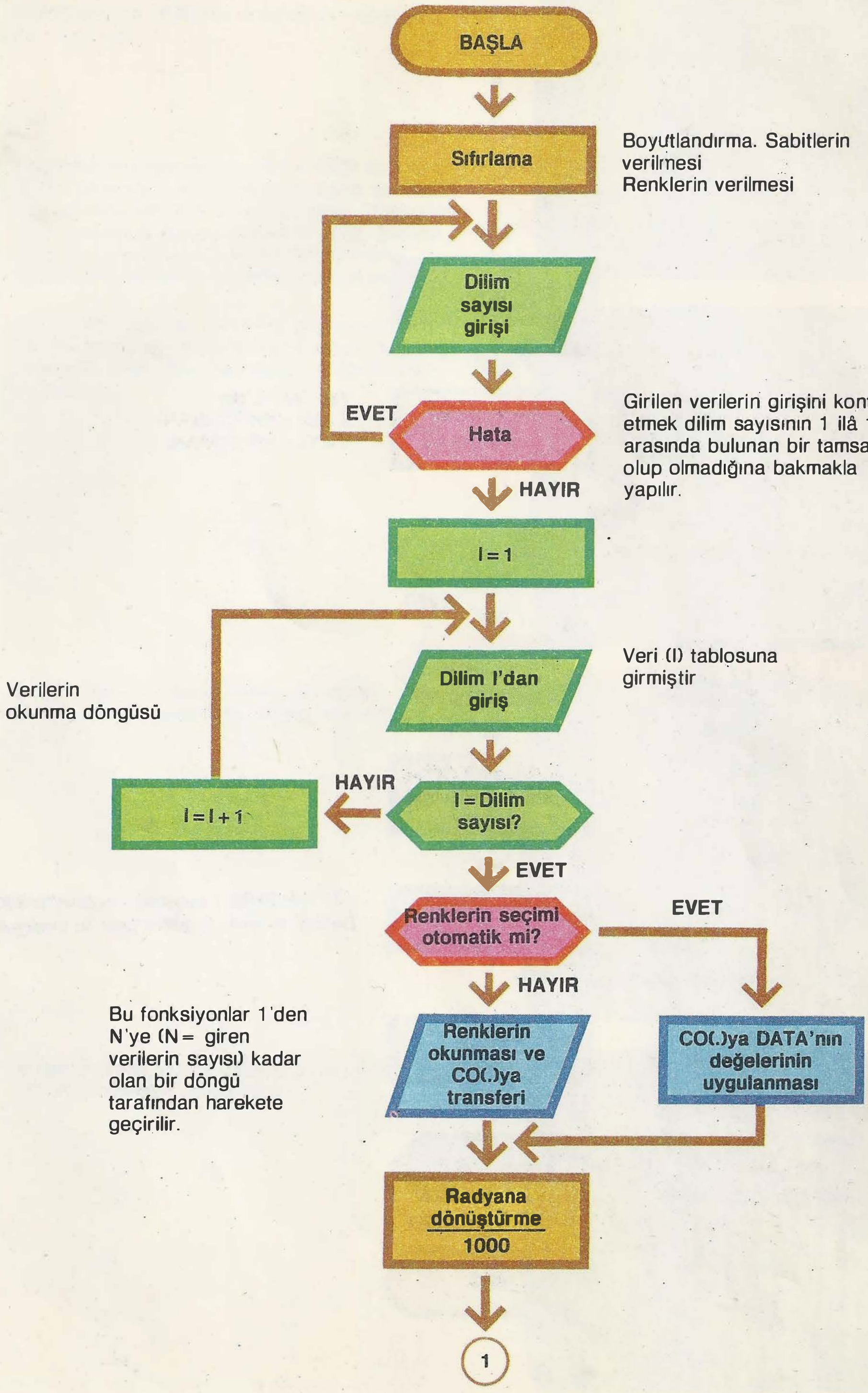
Boş sütunlar

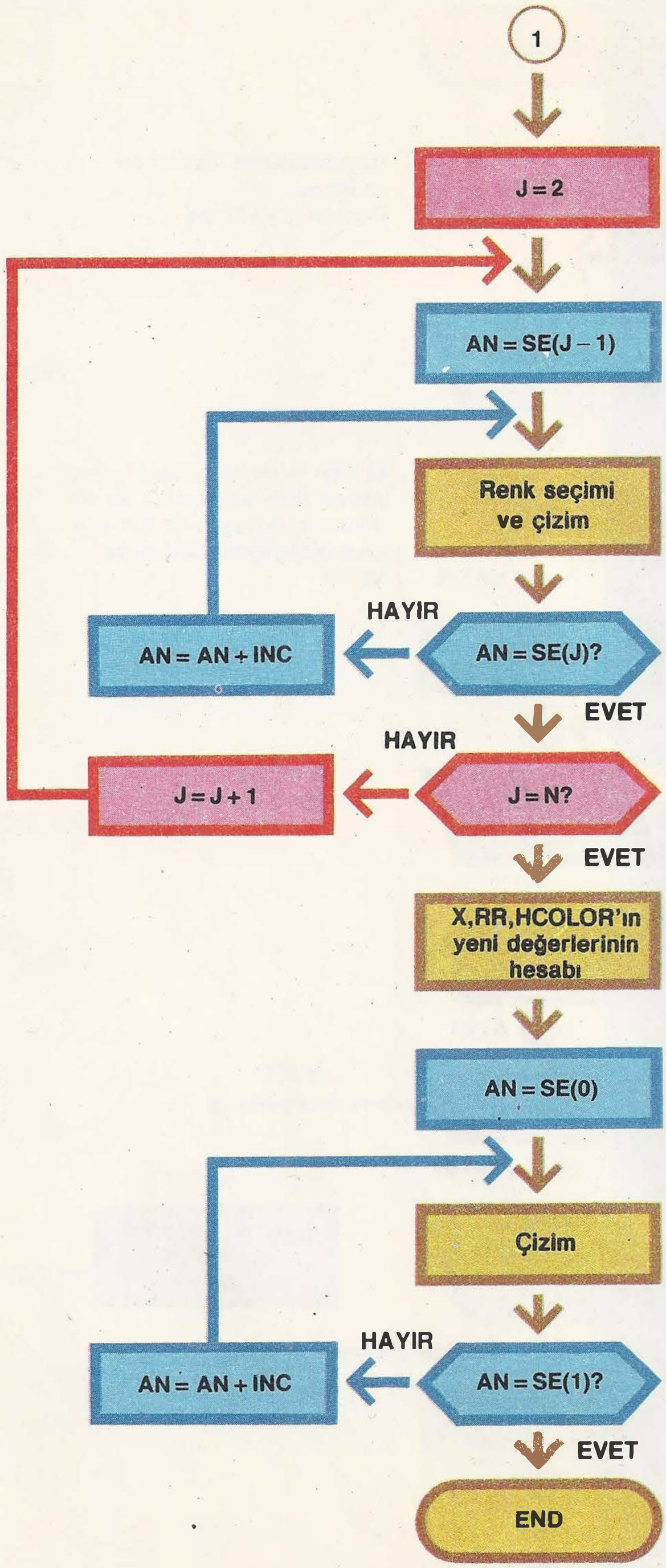


Dolu sütunlar



DİLİMLİ BİR DİYAGRAMIN ÇİZİMİ İÇİN AKIŞ ŞEMASI





Yay, Xc,Yc'den
X = Xc + RR*COS(AN)
Y = Yc—RR*SIN(AN)

OPTION BASE 1 seçeneği öngörülmemiştir.
Demek ki tablo 0 göstergesi ile başlıyor.

DİLİMLİ ALT-PROGRAMLARIN GÖSTERİMİ

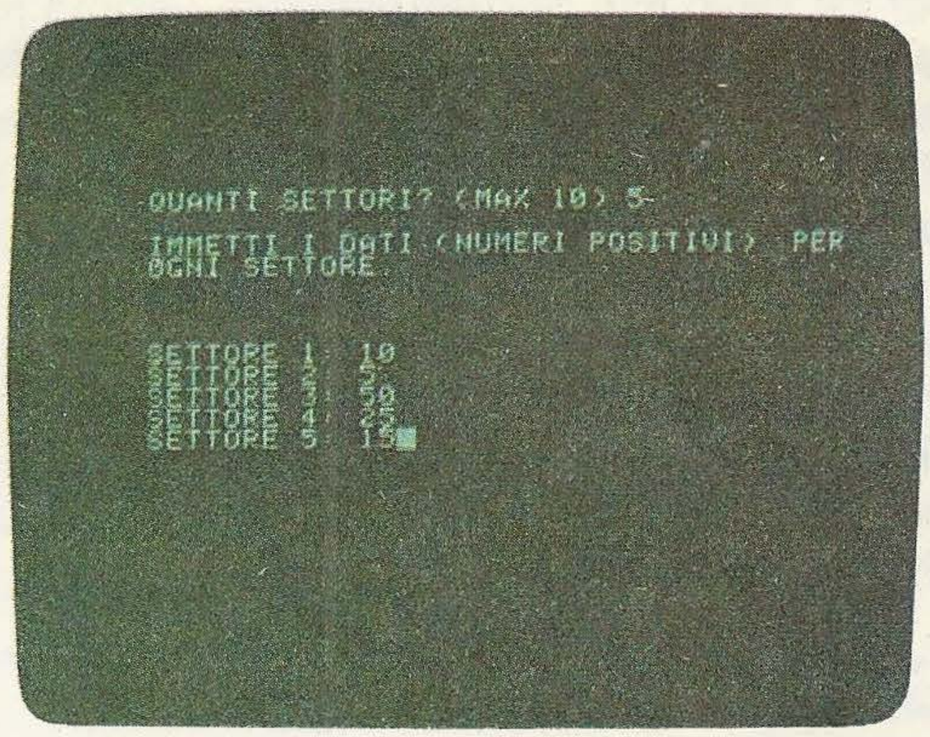
```
10    REM ----------
20    REM BOLUMLU ALT-PROGRAMLAR
30    REM ----------
40    REM
50    REM
60    XC=130
    : YC=90
    : R=90
    : PI=3.1415926
    : INC=PI/300
70    DIM SE(10),C(10),CO(10),S(10)
80    FOR I = 1 TO 10
    : READ C(I)
    : NEXT
90    DATA 5,6,7,1,2,3,5,6,7,1
100   REM
110   REM ----------
120   REM VERILERIN GIRISI
130   REM ----------
140   REM
150   HOME
160   VTAB 4
    : INPUT "SEKTOR SAYISI (MAK.10)"
    ; A$
170   A = VAL(A$)
180   IF A < 1 OR A > 10 OR INT(A) < > (A) THEN 150
190   VTAB 6
    : PRINT "HER BIR SEKTORUN VERILERININ
      (POZITIF SAYILAR) GIRISI
200   N=A
    : TT=0
210   FOR I = 1 TO N
220   VTAB (10+I)
    : PRINT "SEKTOR"I;
    : INPUT ": ";A$
230   S(I)=VAL(A$)
    : IF S(I)<0 THEN 220
240   TT=TT+S(I)
    : S(I)=TT
250   NEXT
260   HOME
270   VTAB 4
    : PRINT "ALT-PROGRAMDA 'N'
      SEKTORU VAR"
280   REM
290   REM ----------
300   REM RENKLERIN SECIMI
310   REM ----------
320   REM
330   VTAB 6
    : INPUT "RENKLERI KENDINIZ MI SECMEK ISTERINIZ YOKSA
      BUNU BILGISAYAR MI USTLENSIN
      (1/2) ";A$
340   IF A$ < > "1" THEN 390
360   FOR I = 1 TO N
    : CO(I) = C(I)
    : NEXT
370   IF CO(N) = 5 THEN CO(N) = 1
380   GOTO 550
390   HOME
400   VTAB 4
    : PRINT "SECILEN RENGE BAGLI OLARAK
      SU TUSLARDAN BIRINE BASINIZ
      0 <V>,<R>,<A>,<B>,0 <W>
410   FOR I = 1 TO N
420   VTAB (6 + I)
    : PRINT "BOLUMUN  RENGI"I;
    : INPUT ": ";A$
430   IF A$ = "V" THEN CO(I) = 1
    : GOTO 490
```

```
440    IF A$ = "R" THEN CO(I) = 2
     : GOTO 490
450    IF A$ = "A" THEN CO(I) = 3
     : GOTO 490
460    IF A$ = "B" THEN CO(I) = 4
     : GOTO 490
470    IF A$ = "W" THEN CO(I) = 5
     : GOTO 490
480    GOTO 420
490    NEXT
500    REM
510    REM ----------------------------------------
520    REM BOLUMUN GOSTERILISI
530    REM ----------------------------------------
540    REM
550    HOME
560    VTAB 4
     : INPUT "BOLUMU YENIDEN GOSTERMEK
       ISTIYORMUSUNUZ ? (E/H) "; A$
570    FL = 0
     : IF LEFT$ (A$,1) = "E" THEN FL
       = 1
580    GOSUB 1000
590    REM
600    REM ----------------------------------------
610    REM DIYAGRAMIN RESMI
620    REM ----------------------------------------
630    REM
640    HGR2
650    FOR J = 2 TO N
660    FOR AN = SE(J - 1) TO SE(J) STEP INC
670    HCOLOR= CO(J)
680    HPLOT XC,YC TO XC + R * COS (AN),
       YC - R * SIN (AN)
690    NEXT AN,J
700    X = XC + 15 * FL
     : HCOLOR = CO(1)
     : RR = R - 8 * FL
710    IF SE(1) > .5 THEN RR = R - 3 * FL
     : X = XC + 12  * FL
     : IF SE(1) > 1 THEN RR = R
     : XC = XC+ 9 * FL
720    FOR AN = SE(0) TO SE(1) STEP INC
730    HPLOT X,YC TO X + RR * COS (AN),Y
       C - RR * SIN (AN)
740    NEXT
750    END
1000   REM
1010   REM ----------------------------------
1020   REM RADYANA DONUSTURME
1030   REM ----------------------------------
1040   REM
1050   IF TT = 360 THEN 1090
1060   FOR I = 1 TO N
1070   S(I) = S(I) - TT / * 360
1080   NEXT
1090   FOR I = 1 TO N - 1
1100   SE(I) = S(I) - S(1) / 2
1110   NEXT
1120   SE(0) = - S(1) / 2
     : SE(N) = 360 + SE(0)
1130   FOR I = 0 TO N
1140   SE(I) = SE(I) / 180 * PI
1150   NEXT
1160   RETURN
```

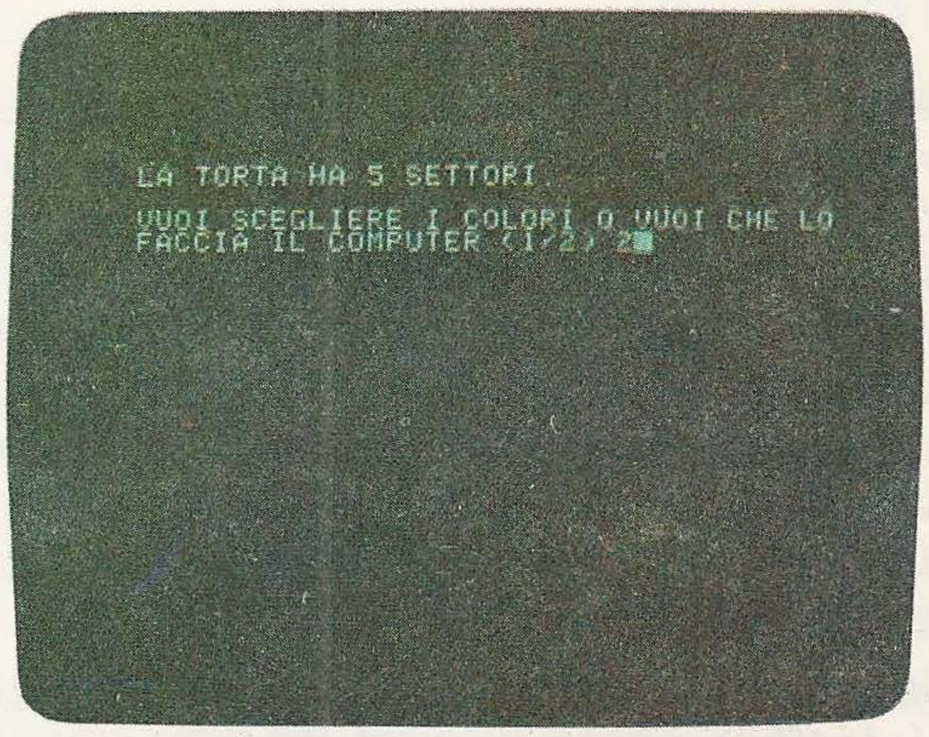
1525, ve 1526. sayfalarda dilimli bir diyagramın görüntüleme programının akış şemasını bulacaksınız. Bu program, tasarlama açısından, histogramlarınkinden farklı değildir. Fakat programın, verilerin değerleri ve toplama göre bu değerlerin yüzdeleri için, bir dizi dikdörtgen yerine, gösterilecek olan verilerin sayısına eşit sayıda dilimlere bölünmüş bir dai-

DİLİMLİ DİYAGRAM

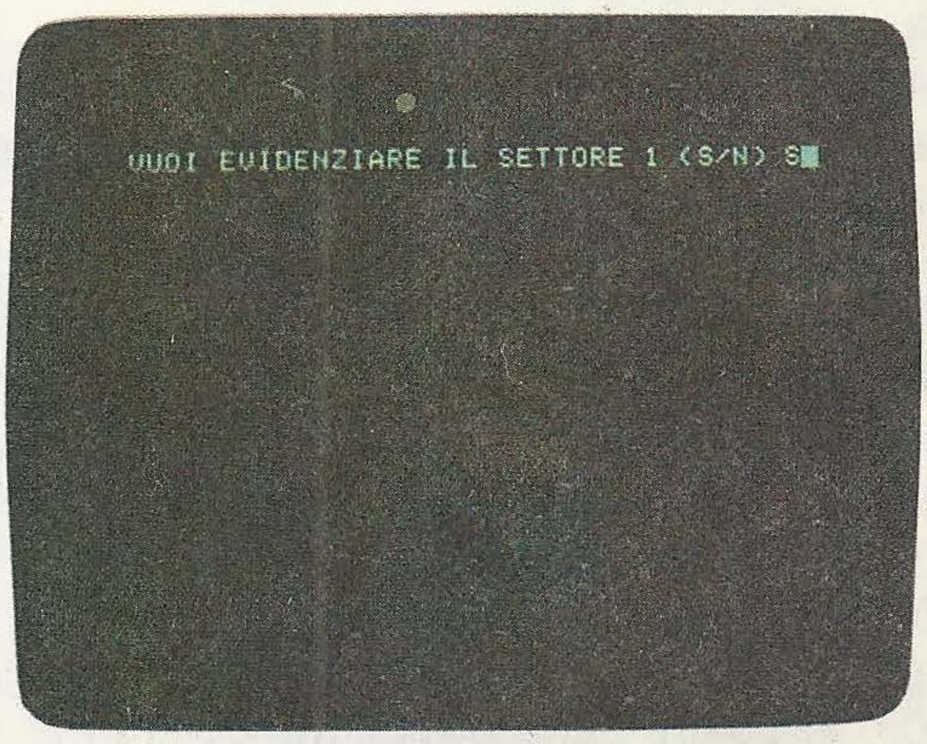
Program (satır 160) ekrana kaç tane dilim (veri sektörü) gösterilmesi gerektiğini sorar. Girilen dilim sayısını (burada 5) verilerin istenmesi aşaması (satır 190'dan 250) izler.



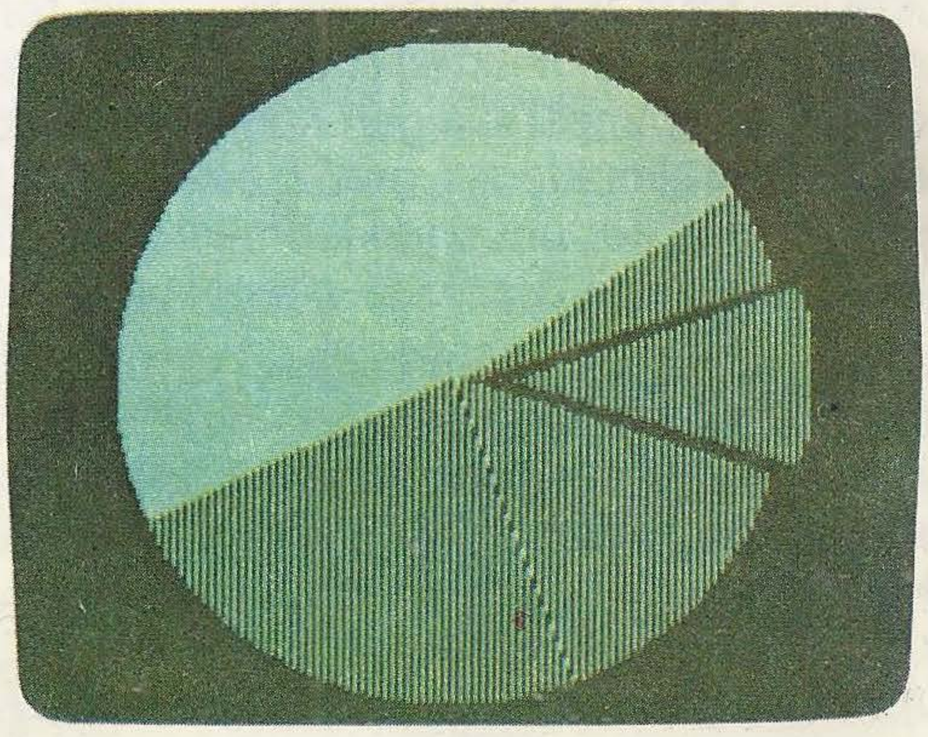
Verilerin girişi bittiğinde program ekrana giren veri sayısını bildirir (satır 270) ve renklerin seçimi ile ilgili ayrıntıları sorar. (satır 330)



Kullanıcı renklerin otomatik seçimine karar verdiğinden, kontrol birinci dilimin değerlendirilip değerlendirilmeyeceğini soran satır 560'a geçer.



Dilimli diyagramın gösterilişi. Birincisi "kesik" olan dilim. Diğerleri bir saatin yelkovanın döndüğü yönün tersini izlerler.



reyi görüntülemesi gerekmektedir. Program, her bir dilim için görüntülenecek verinin miktarına, gösterilen veriye orantılı bir alan tanımaktadır. Program (1527. ve 1528. sayfadaki dökümler) renkli bir ekranın kullanımını öngörmektedir. Kullanıcı tarafından ayrı bir komut verilmedikçe, diyagram dilimlerinin renkleri otomatik olarak program tarafından seçilir. Tek önemli özellik, kullanıcı tarafından girilen değerleri radyan ile ifade edilmiş değerlere (dilimlerin büyüklükleri) dönüştüren 1000 alt-programıdır.

Grafik Modda Bellek Yönetimi

Ekran, grafik uygulamalarda en yaygın kullanılan çıkış birimidir. Ne var ki bu birim, çıkan görüntüleri bellekte tutmaz. Bir metnin veya bir resmin (PRINT, LINE vd) tek bir komut ile görüntülenmesi, aslında aynı görüntünün, her bir noktasını yeniden oluşturmak amacıyla çok kısa aralıklarla aktive etmeyi sağlayan sürekli bir faaliyet gerektirmektedir. Grafik görüntünün özel bir bellek bölümüne yerleştirilmesinin nedeni budur.

Sistemin programları grafiği sürekli yeniden görüntüleyerek ekrana aktarmak için, bu bellek bölümünün içeriğinden örnekler alır. Bu BASIC veya bambaşka bir dildeki bir metin için de geçerlidir. Demek ki, ekranın her bir noktasının durumu hakkındaki bilgiyi içerecek bir bellek bölümüne sahip olmak gerekmektedir. Sözkonusu belleğin büyüklüğü ekran türüne ve kullanılan sisteme bağlıdır.

Ekran tek renkli ise, nokta başına bir bit yeterlidir. Belli bir noktaya karşılık gelen bellek bölümünün içeriği, örneğin sıfır ise o noktanın gözükmediği, 1 ise aktive edildiği, yani gözüktüğü anlamına gelir. Renkli ekranlarda belleğin kapsamı daha büyüktür, çünkü her bir nokta için ON/OFF durumuna ek olarak rengi tanımlamak gerekmektedir. Demek ki bir resmin görüntülenmesi için gereken işlem iki aşamada yürütülmektedir: grafik belleğe iki tabanlı görüntü kodunun yüklenmesi ve görüntüleme sürecinin harekete geçirilmesi.

Örneğin iki nokta arasındaki bir doğru parçasının çizim komutları (LINE, HPLOT, vb) ekran noktası olarak ifade edilmiş koordinatlardan yola çıkarak, bunlara karşılık gelen bellek konumlarını hesaplar. Bu amaçla komutlar bu doğru parçasıyla ilgili bütün bitleri ON durumuna getirirler. 8 bitlik bir makinede her bellek konumu en fazla 1 ekran noktasını denetler. Eğer bellek konumlarından birine 1 değeri yazılmışsa, denetlenen noktalardan ilki aktive edilmiştir. Buna karşılık 3 değeri ile, bitişik olan iki nokta (1 + 2) harekete geçirilmiş olacaktır; bu, işlemde öngörülen en büyük değere kadar devam edecektir.

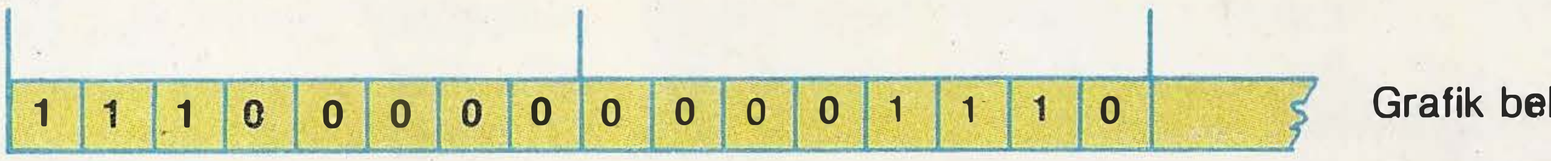
Bir resmin grafik kalitesi tamamen ekranın gücüne bağlıdır. Nokta sayısı ne denli fazla olursa, gereken bellek bölgesi de o ölçüde kapsamlı olmaktadır. Ayrıca çok geniş bir alanın işgalini önlemek için yüksek kararlılık durumlarında bile yönetilen nokta sayısını azaltmak gerekmektedir. 280 yatay, 190 dikey konumu olan bir ekran, yaklaşık her biri bir bit'e karşılık gelen 53.000 durum göstergesi gerektirmektedir. Demek ki, her konum için, elverişli olan 8 bitten yalnızca 7'sinin kullanıldığı varsayıldığında, bellek konumları yaklaşık olarak 7.5 kilobaytlık bir hacim gerektirir. Bu özel bellek bölgesinin varlığı, programlamada önemli kolaylıklar sağlamaktadır. Belirli karmaşıklıktaki şekilleri yapmak için, kullanıcı, grafik komutlarla sınırlanmadan bellek adreslerinin değerlerini doğrudan girme yoluna başvurabilmekedir. Tersine yanık veya sönük olan her bir ekran noktasının durumunu bilmek ve resmi işlemek veya saklamak için belleğin içeriği kolaylıkla okunabilir. Bu yöntemde grafik, üstünde her türlü sayısal işlemin yapıldığı bir sayısal değerler tablosu olarak düşünülür. Ekranda görüntülenmesi ya otomatiktir (yani tabloda yapılan her değişiklik ekrana yapıldıkça) ya da istendiğinde (grafik mod harekete geçirildiğinde) gerçekleşir. Böyle olmakla birlikte, işin bu yürütülüş biçimi makinelerin tümünde geçerli değildir. Bazılarında grafik bellek yalnızca sistemin denetimindedir, böyle olunca da yüksek düzeyde komutlara başvurmak gerekmektedir.

Grafik Bellek

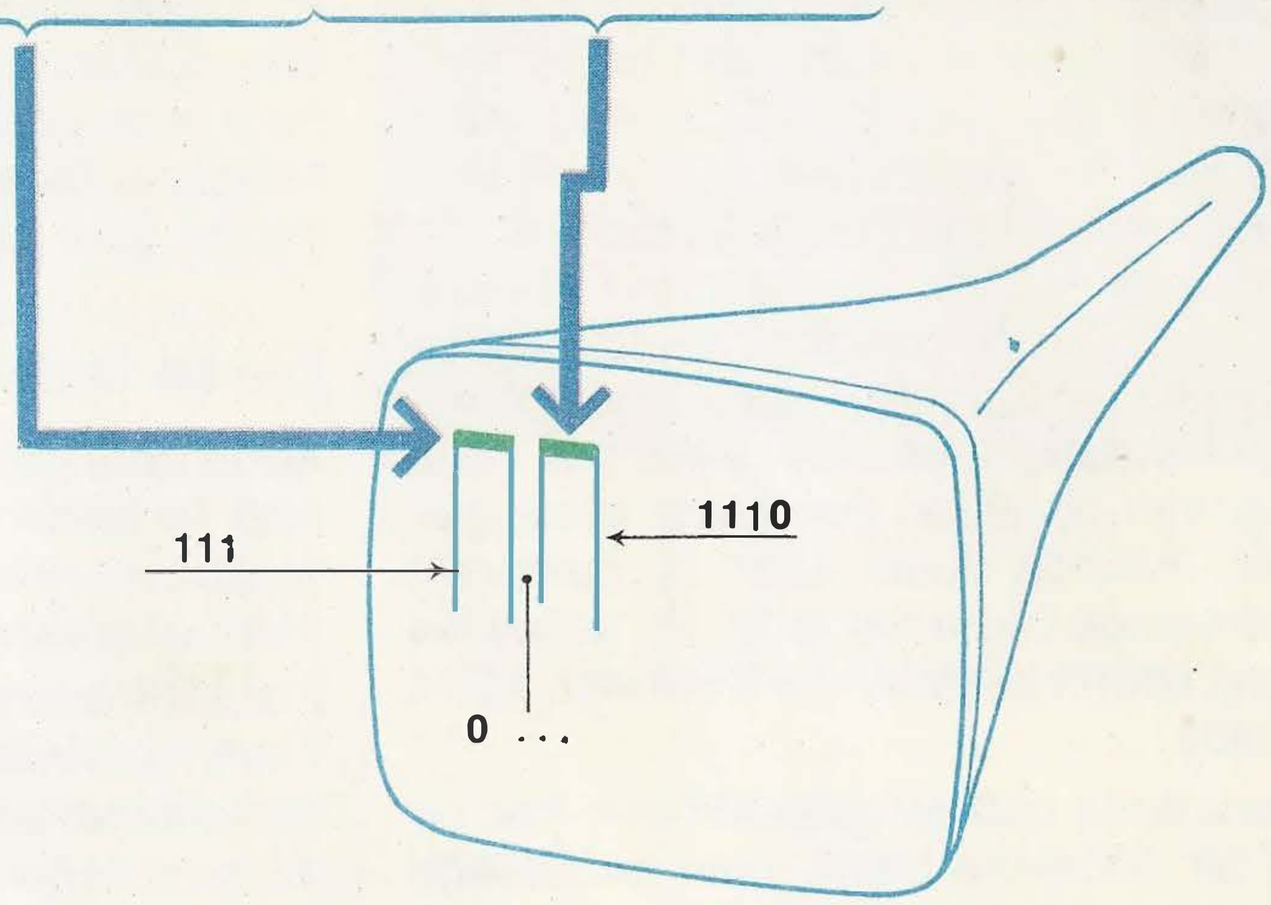
Bir ekran noktasını, o noktayı bellek adresine bağlayan ilişki ne basit ne de doğrusaldır. Ekranın yönetiminin işleyişi değişik parametrelere ve bazen de kullanılan merkezi işlem biriminin türüne bağlıdır. Biz burada Apple ve uyarlı makine türlerinde bulunan Rockwell 6502 merkezi işlem birimine sahip kişisel bilgisayarlarda çoğunlukla kullanılan belleğin bölümlere ayrılışının bir örneğini inceleyeceğiz.

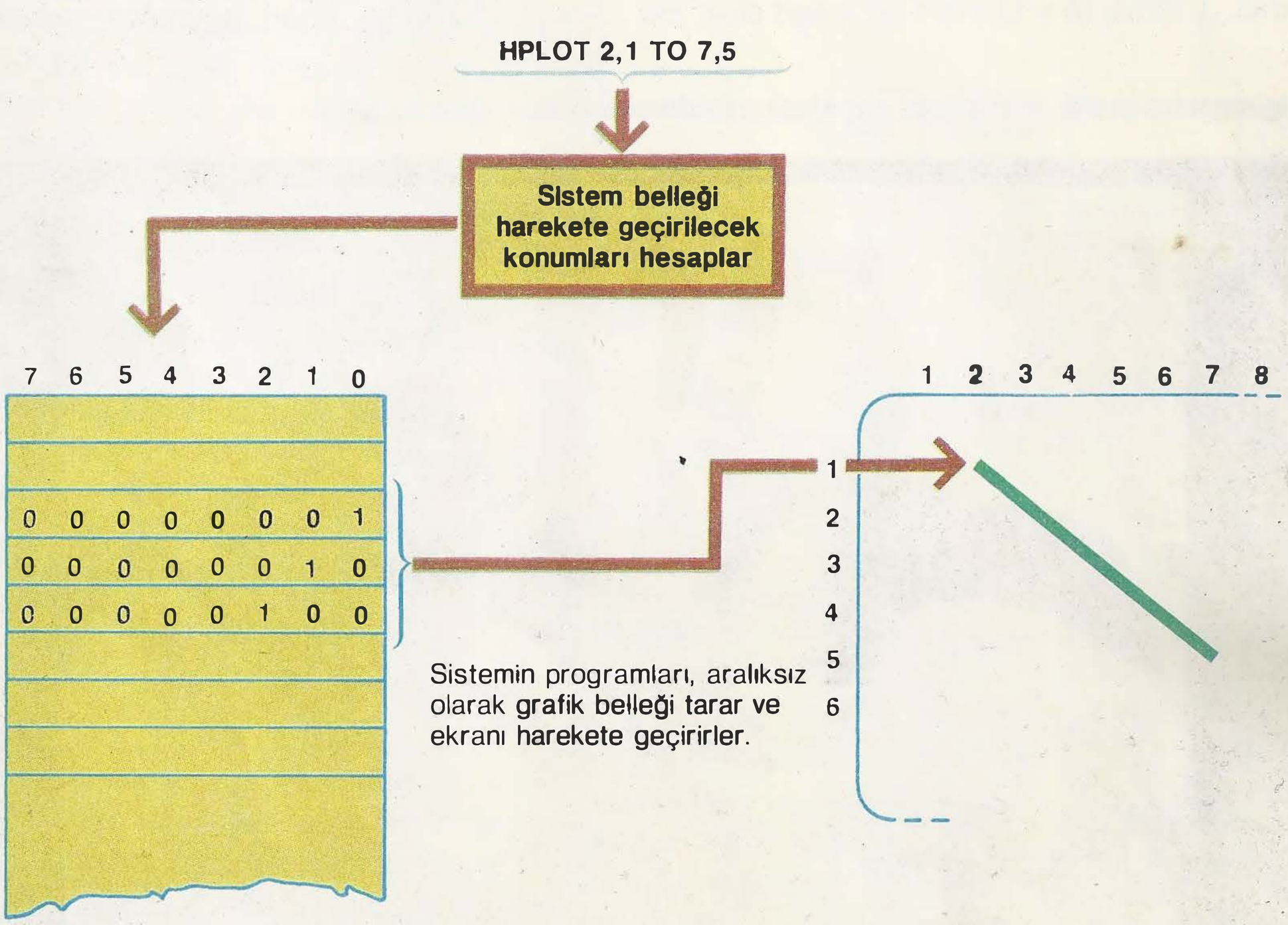
Ekrana karşılık gelen bellek bölgesi, çoğunlukla malzemeden kaynaklanan nedenlerle hemen hemen hiçbir zaman bitişik adreslerden oluşmaz. Yani örneğin 100 konum uzaklıkta iki bölge, aralarında 100 fark olan iki adrese

EKRANDA GRAFİK MODUN YÖNETİMİNİN BASİTLEŞTİRİLMİŞ ŞEMASI



Her konum belirli sayıda yanyana bulunan noktaların durumunu belirtir.





Sistemin programları, aralıksız olarak grafik belleği tarar ve ekranı harekete geçirirler.

karşılık gelmezler. 1533. sayfadaki şema Rockwell 6502'ye dayanan sistemlerde genellikle kullanılan iki ekran sayfasından birini göstermektedir. Ekranın birinci bölgesi (7 nokta) görüntü belleği olarak 8192 numaralı konuma sahiptir. Aynı satırda bulunan sonraki bölgeler bitişik bellek adreslerine sahiptirler. Aslında aynı satır üzerinde bulunan değişik bölgeleri ayırmak için, çıkıştaki adrese 1 eklemek yeterlidir. Ekran genellikle 0'dan 39'a kadar numaralanan 40 sütundan oluşurken, 0 ile numaralanan 1. satır 8192 ve 8192 + 39 = 8231 adresleri arasında bulunan bellek adresleri tarafından denetlenir. 2. satıra (1 numaralı) geçerken buna karşılık gelen belleğin 8232'de, yani bir önceki satır tarafından işgal edilen adresten sonraki boş adresten başlaması beklenebilir. Fakat bu bellek çok daha altta bulunan bir ekran noktasına denk gelmektedir. Aslında ikinci satır (1 numaralı) 9216 adresinde başlamaktadır. Bu adres ise bir önceki satırın başlangıç adresinden 1024 uzaklıktadır.

Diğer satırlarda olduğu gibi, ekranın her bir bölgesi bir öncekine bitişik olan bir belleğe karşılık gelir. 1024'lük artış 7. satır 1. numaradan 7 numara için doğru olurken; 8. satırınki, 1'incisi (n + 1) artı 128'e eşit olur. Bu işlem 7 kez tekrarlanır, bu defa 8 satırdan oluşan 8 tane öbek oluşturur. Ekranın ikinci bölümüne karşılık gelen bellek yine 8 satırlı 8 öbeğe bölünür. Bu özel örnekte 8232 sözkonusu olduğuna göre, yalnızca birinci adres değişmektedir. 1533. sayfadaki şema ekranın belleğinin bütününün bölümlenişini göstermektedir. İlk yarısı bütünüyle görüntülenmiştir, ikinci yarısı ise bu şemadan çıkarsanabilir. Adreslerin numaralanması işlemi aynı ilkeyi izler. 0,0 koordinatları tarafından belirlenen birinci ekran noktası 0 bitinde 0'a karşılık gelmektedir. Diğer noktaların adresleri 8192 değerine göre hesaplanır.

Ekran Belleğinin Adreslenmesi

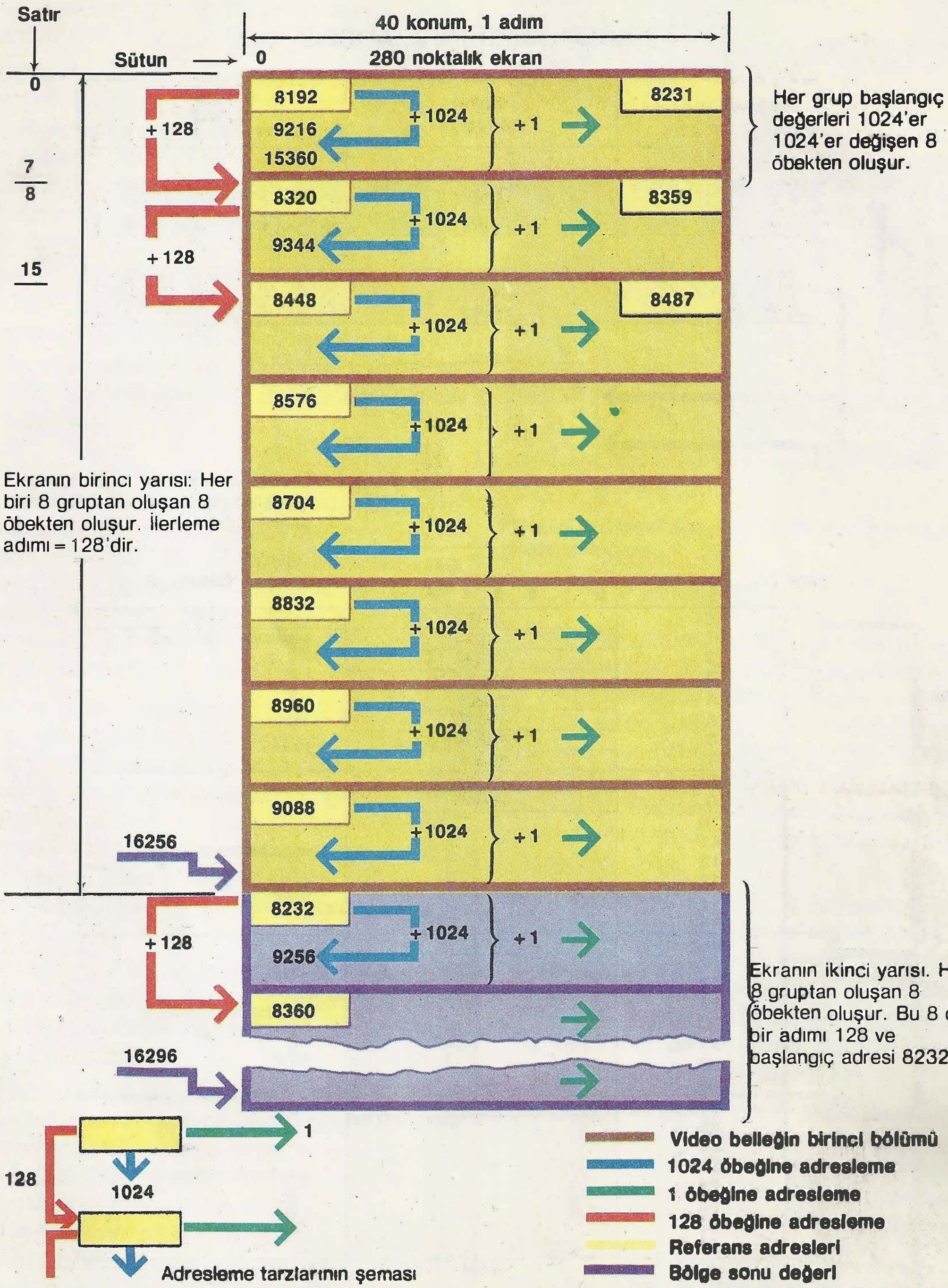
Adresleme işlemini anlatmak için bir hesap örneği ile bellek konumlarının ekran noktasına ve ekran noktalarının bellek konumlarına dönüştürülmesi alt-programlarını ele alacağız.

Y ekseni üzerinden adresleme. Herhangi bir Y satırı üzerinde sıfır sütununda (X=0) bulunan bir noktanın adreslemesi sözkonusu olduğunda, giriş değeri ordinattır (Y); bu değişkenden yola çıkarak buna karşılık gelen bellek konumunu bulmak gerekmektedir. Y=7 için konum 15360'tır. 1534. sayfadaki şekil bir ön-

Bilgisayarın grafik olanakları sanatsal uygulamaları da mümkün kılar.



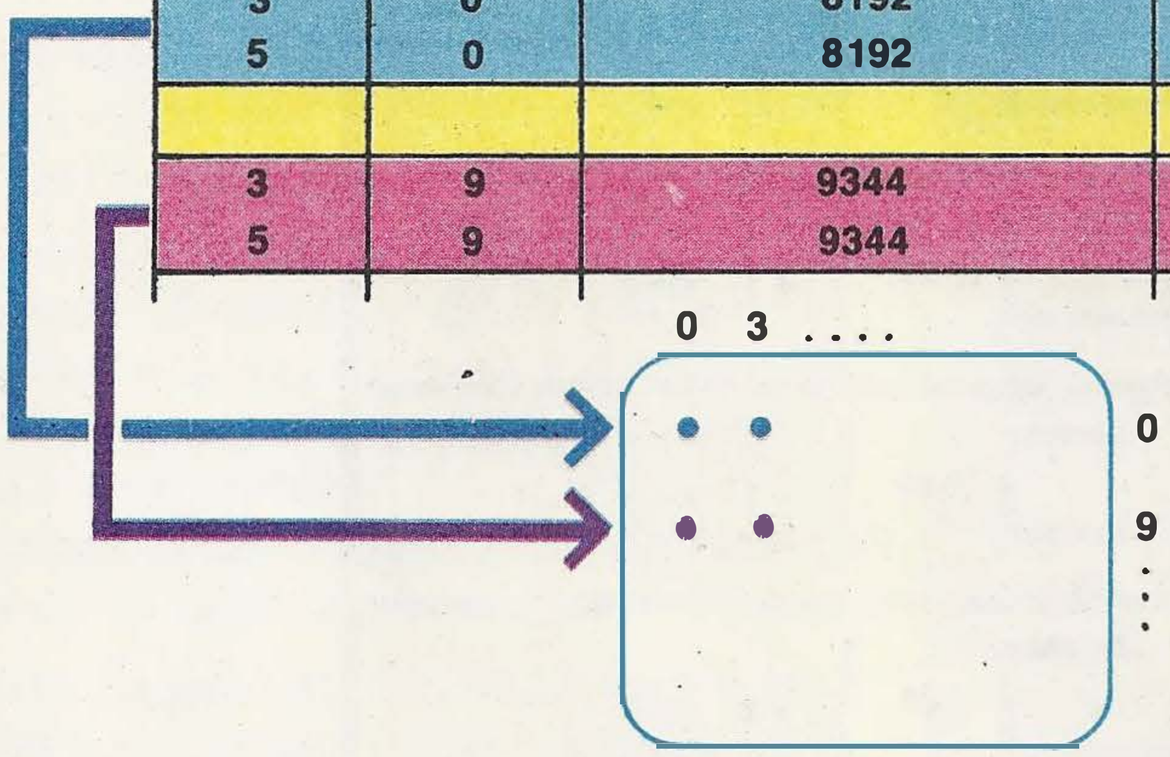
EKRAN BELLEĞİNİN TOPOGRAFYASI: GEEL ŞEMA



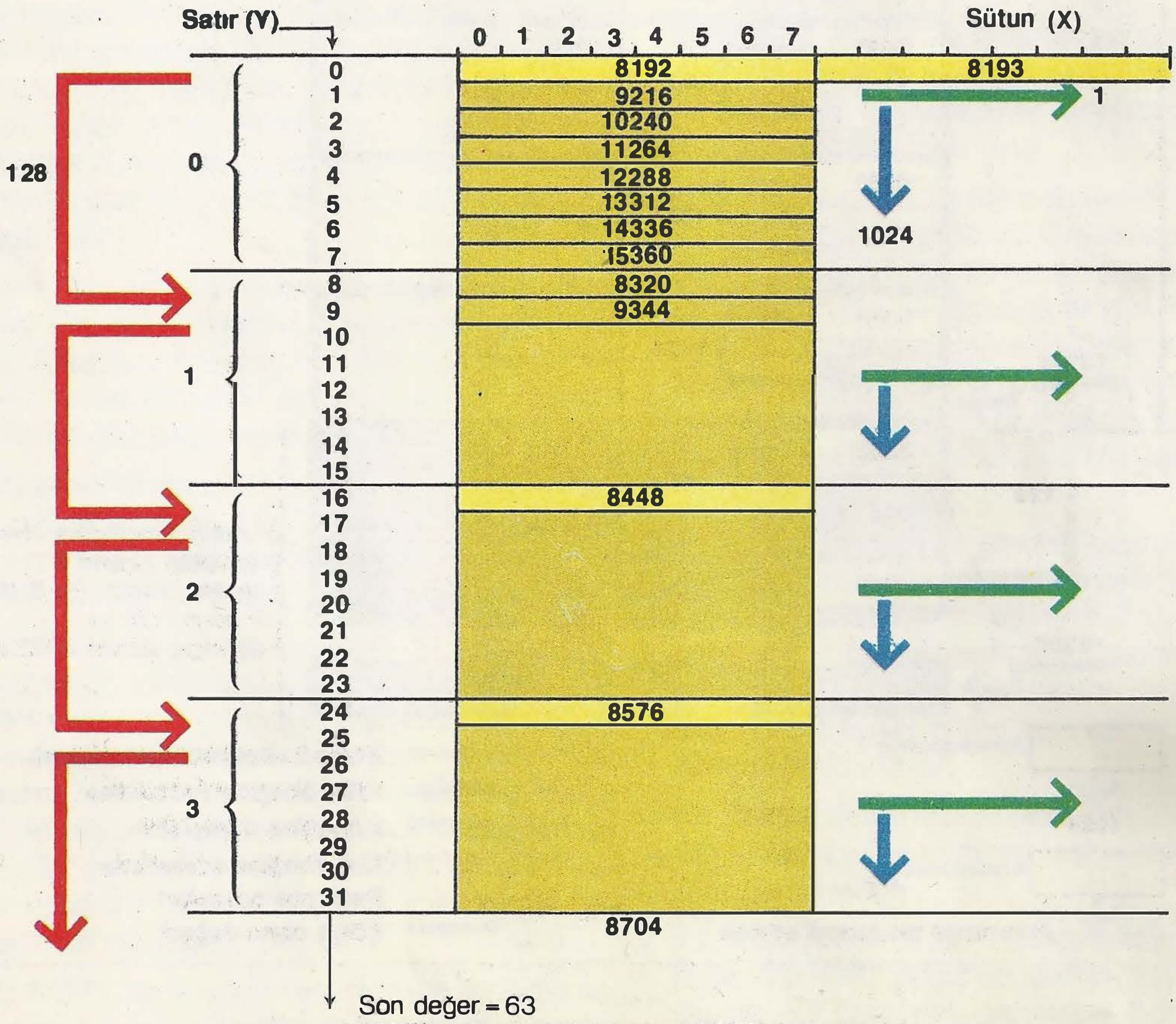
.ceki şekilde X = 0 olan noktalarla ilgili bölümü yeniden ele almaktadır. Yer sorunu nedeniyle yalnızca ilk 4 öbek gösterilmiştir (Y,0 ile 31 arasında bulunuyor). Diğerleri için ise işleyiş

Y EKSENİ ÜZERİNDE ADRESLEME

Koordinatlar X	Koordinatlar Y	Bellek konumları	Belleğin içeriği 0	1	2	3	4	5	6	7
3	0	8192	0	0	0	1	0	0	0	0
5	0	8192	0	0	0	0	0	1	0	0
3	9	9344	0	0	0	1	0	0	0	0
5	9	9344	0	0	0	0	0	1	0	0



Belleğe adresleme, bellek içeriği ve ekran noktası arasındaki bağlantı örneği



Son değer = 63
128'den sonra birinci adres 8232'dir.

mekanizması belirtilmektedir.
Verilen bir Y değerine karşılık gelen başlangıç noktasını saptamak için şunlar gereklidir:

A - Başlangıç noktasının hangi G öbeğine (8 satırlık grup) ait olduğunu saptamak. Her grupta Y'den 8 değer bulunduğuna göre; G, Y'yi 8'e bölerek ve sonucun tam olan bölümünü alıkoyarak elde edilir.

B - Y'nin ait olduğu G grubun başlangıç adresini saptamak. Her grubun başlangıç konumları 128'den 128'e değişirken, G'nin adresi 8192+G*128 formülü tarafından verilir.

C - Y'nin G'deki konumunu saptamak. Her grup Y'nin belirli bir değeri (0,8,16 vd.) ile başlar ve her grup 0'dan 7'ye kadar 8 satıra sahip olduğuna göre +7'lik bir değer ile biter. Herhangi bir Y satırı öbekte Y+G*8 konumunu işgal eder.

Gerçekten de G, Y'nin ait olduğu gruptur, G*8 çarpımı ise Y satırından önce gelen satır sayısı ile başka öbeklere ait olan satırları belirtir. Y ile bu son değer arasındaki fark, Y satırının G öbeğindeki konumunu verir.

Öbekte ilerleme adımının 1024 olduğu bilindiğinde, bellek konumunu bulmak için Y'nin G öbeğinin içindeki yeri olan bir önceki değeri 1024 ile çarpıp sora da buna G noktasında hesaplanmış olan G öbeğinin başlangıç adresini eklemek erekmektedir. Belleği 8232 adresinde başlayan ekranın ikinci bölümü, aynı şekilde ele alınabilir; elbette ki, bu işlem, G noktasında kullanılmış olan başlangıç değeri yerine 8232 değerini koyarak yapılır.

Pratik nedenlerle 1535. sayfada her biri bir

Y EKSENİ ÜZERİNDEKİ REFERANS KONUMLARININ TABLOLARI

Birinci grup — Başlangıç 8192- ordinat (Y) değeri 0 ila 63

Öbek	Y'nin değerleri da	a	Öbeğin ilk adresi
0	0	7	8192
1	8	15	8320
2	16	23	8448
3	24	31	8576
4	32	39	8704
5	40	47	8832
6	48	55	8960
7	56	63	9088

Bellek adresi = 8192 + 128*Öbek
Öbek = INT (Y/8)

İkinci grup

Öbek	Y'nin değerleri da	a	Öbeğin ilk adresi
8	64	71	8232
9	72	79	8360
10	80	87	8488
11	88	95	8616
12	96	103	8744
13	104	111	8872
14	112	119	9000
15	120	127	9128

Bellek adresi = 8232 + 128 *(Blok-8)
Öbek = INT (Y/8)

bölgeye ilişkin olan iki tablo verilmektedir. Bu tablolardan her biri, Y'nin değerine bağlı olarak ait olunan öbek ile bu öbeğin başlangıç konumunu vermektedir. Örneğin Y = 35,8704 konumunda başlayan öbek 4'e aittir. Bu öbek:

8704 + 1204 * (35—4 * 8)

= 8704 + 1204 * 3 = 11776

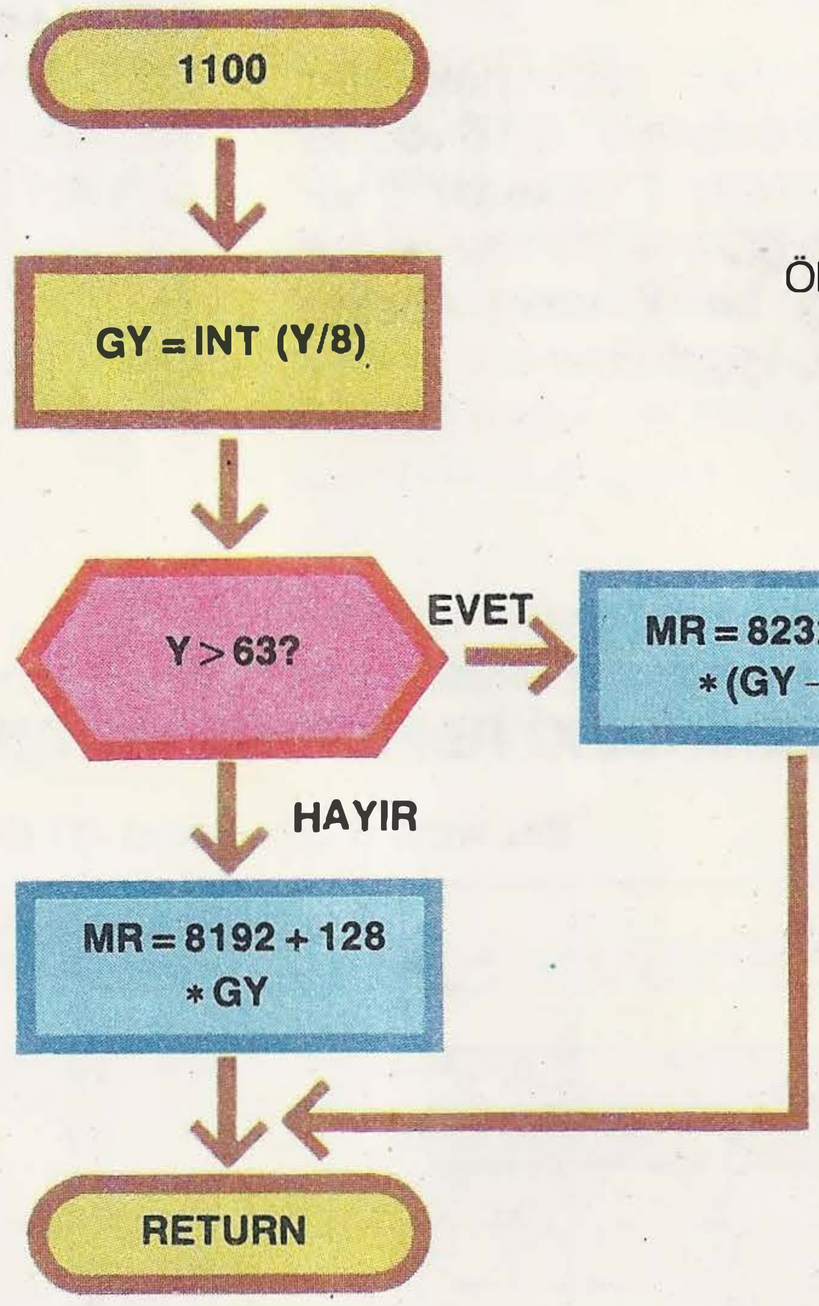
konumuna karşılık gelir.

Aşağıdaki akış şemaları bir Y değerinden yola çıkarak hep X = 0 olmak üzere Y'ye karşılık gelen adresi hesaplayan iki alt-programı göstermektedir.

Y EKSENİ ÜZERİNDE ADRESİN HESAPLAMASI

Referansı Y ekseni alarak adresin hesaplanması

Giriş Y = Ordinat değeri
Çıkış MR = Öbeğin başlangıç adresi
GY = Öbek

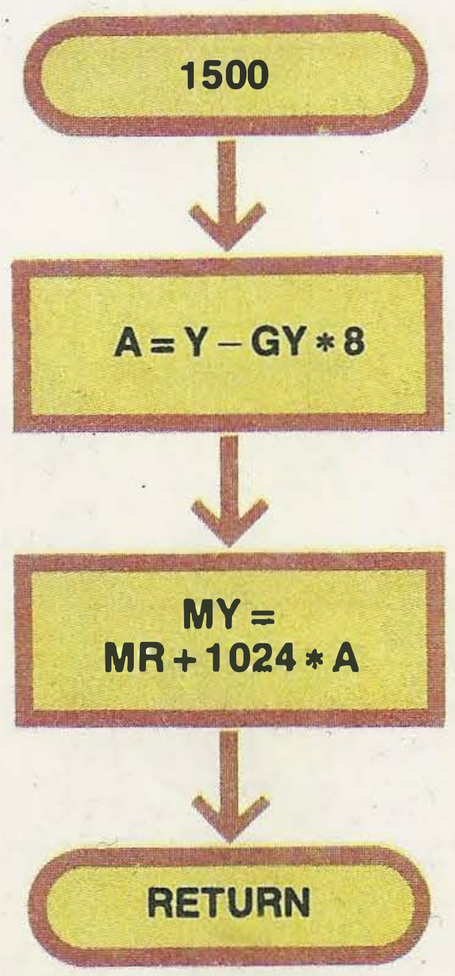


Öbeğin hesaplanması

Adım 128

Ordinata karşılık gelen adresin hesaplanması

Giriş: Y = Ordinat
MR = Öbeğin başlangıç adresi
GY = Öbek
Çıkış: MY = Y karşılık gelen bellek adresi



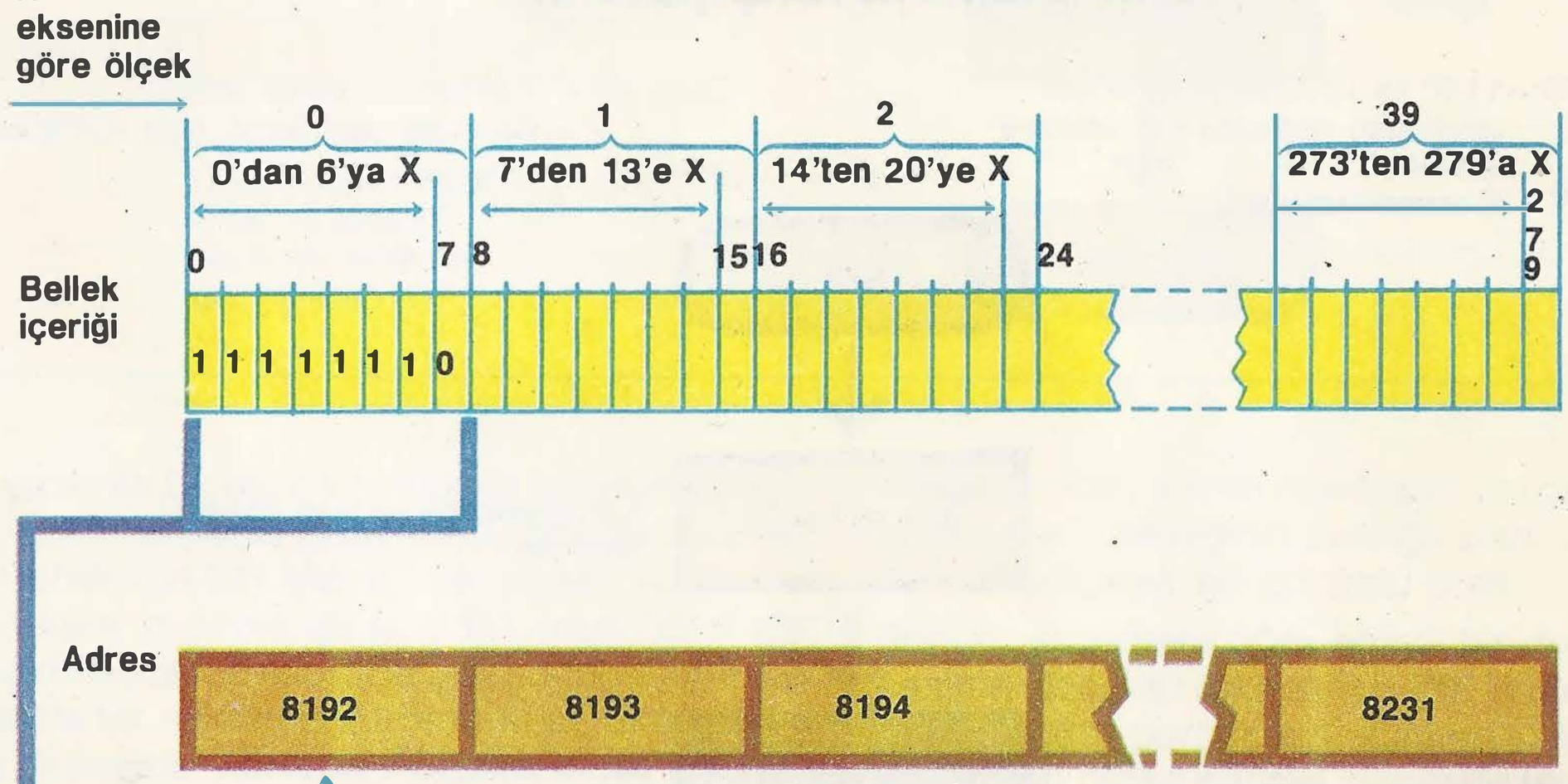
A değişkeni Y'nin konumunu öbek başlangıcına göre belirler.

İlk alt-programda (1100), referans değeri (8192 ya da 8232), Y'nin değeri doğrultusunda yeniden belirlenir. Bellekten adrese erişim basittir, ancak bu teknik en dolaysız olanı değildir. Kullanıcının, sayısal değerleri değiştirme yoluyla müdahale edebilmesine olanak vermek üzere, çok bölümlü bir yöntem seçilmiştir.

X eksenine göre adresleme: Her satırda (Y'nin değeri),sütun değerleri (0'dan 39'a) bitişik bellek adreslerine karşılık gelir. Bir Y satırına ve X sütununa (ya da daha iyisi X = 0 eksenine) karşılık gelen başlangıç değeri bilindiğinde, 1 eklenerek diğer adresler sıralı biçimde elde edilir.

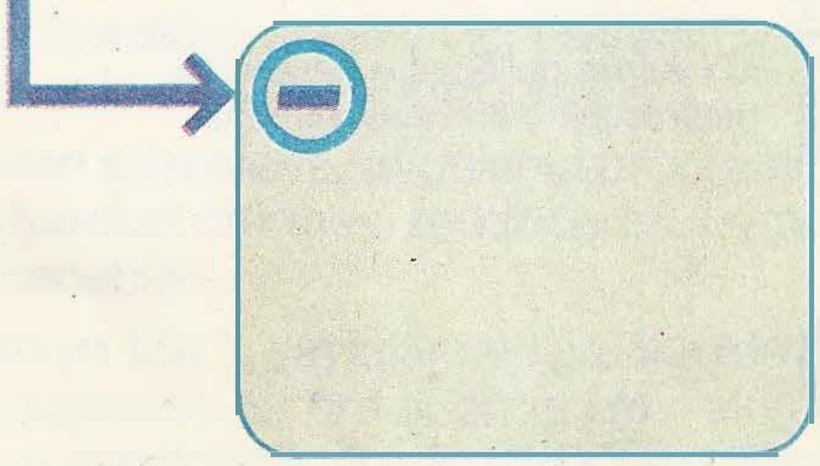
8 bit içeren her bellek adresi, 8 ekran noktasının durumunu belirtir. 0 biti, X = 0 noktasına, 1 biti, X = 1'e karşılık gelir ve bu böylece sürer. Makinemizde, 7 biti (numaralandırma 0'dan başlar) kullanılmamıştır; böylece her bellek adresi 7 noktanın (0'dan 6'ya kadarki

X EKSENİNİN ADRESLENMESİ



X ekseni üzerindeki referans belleği, Y(MY) koordinatının adreslenme belleğidir.

Adreslerin durumları için Y = 0 durumunu örnek olarak ele alıyoruz.



Öbek = INT (X/7)
Bellek = Referans belleği + öbek (artma adımı 1'dir)
Bit = X—Öbek*7

bitler) durumunu (ON/OFF) içerir. 1537. sayfadaki grafikte, Y = 0 satırına karşılık gelen ekran noktasının adresleri bulunmaktadır. Satırın çıkış adresi 8192'dir. Sonrakiler en çok öngörülen satır sayısı olan 39'a kadar (40 adet sütun 0'dan 39'a kadar numaralandırılmıştır), 1 eklenerek elde edilir. Her sütun 7 ekran noktasına bölünmüştür. Tüm sütunların toplamı 7 × 40 = 280 noktadır.

X ekseni üzerinde herhangi bir noktayı adreslemek için önce hangi öbekte olduğunu (X = 0'dan Y = 279'a) belirlemek gerekir. Bu değerden yola çıkarak bellek adresi, daha sonra da bu adresteki hangi bit tarafından belirlendiğini hesaplamak gerekir. Örneğin X ekseninindeki 16 noktası, 2.öbeğe aittir. Gerçekten de:

Öbek = INT(X/7) = 2

8192 satırına karşılık gelen adresi artı 2'de:

Bellek = INT(X/7) + 2 = 8194 olur.
olur.

Böylece belirlenen yerde, ilk bit (0 numara) X = 14 ekran noktasını temsil eder: X = 16 olduğunda incelenen nokta üçüncü bit ile, yani 2 numara ile gösterilecektir (0 biti 14 noktası, 1 biti 15 noktası, 2 biti 16 noktası). Genellikle bitin hesaplanması için,

Bit numarası = X—öbek × 7

olur.

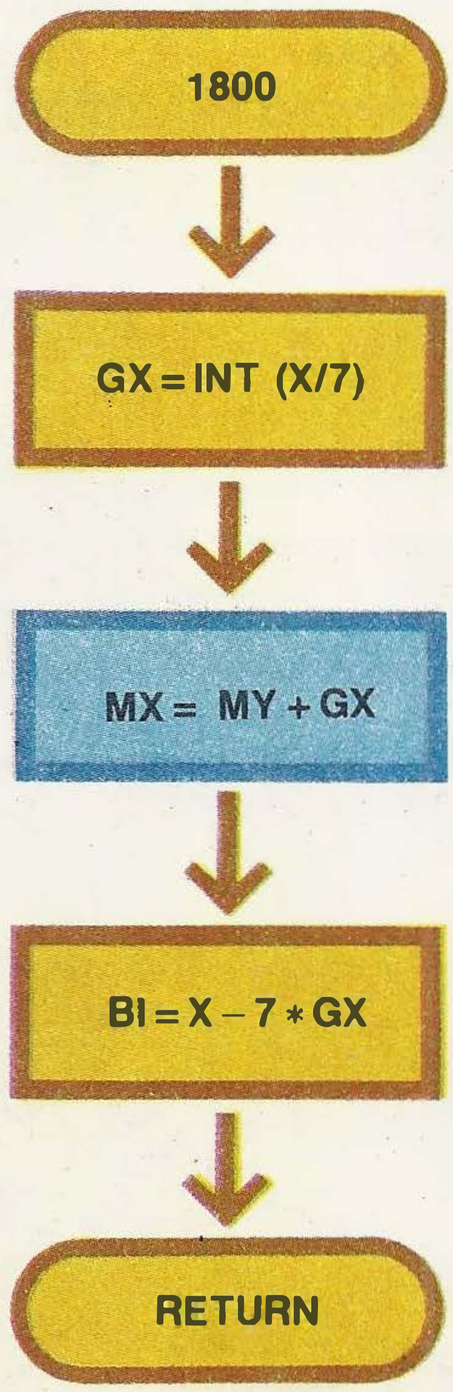
1538. ve ileriki sayfalardaki akış şemaları, bu işlemleri gerçekleştiren iki alt-programı göstermektedir. İlkinde (1800), satırın, Y'nin değerleriyle belirlenmiş olan değerini ve X ekseninin değerini vermek gerekir. Çıkışta ekran noktasının bellek adresi içindeki durumunu (X, Y koordinatları) ve konumunu (bitin numarası) içeren bellek adresi MX elde edilir.

İkinci alt-program (1900), 1 değerini uygun bi-

ADRESLEMENİN AKIŞ ŞEMASI

Girişler: MY = 1100 ve 1500 alt-programları tarafından hesaplanmış referans adresleri
X = X ekseni değerleri

Çıkış: MX = X ekseninin bellek adresi
BI = Harekete geçirilecek bitin numarası



X koordinatına ait olan öbeğin seçilmesi

Artma adımı = 1
MY referansından itibaren

MX adres belleğinde harekete geçirilecek bitin hesaplanması

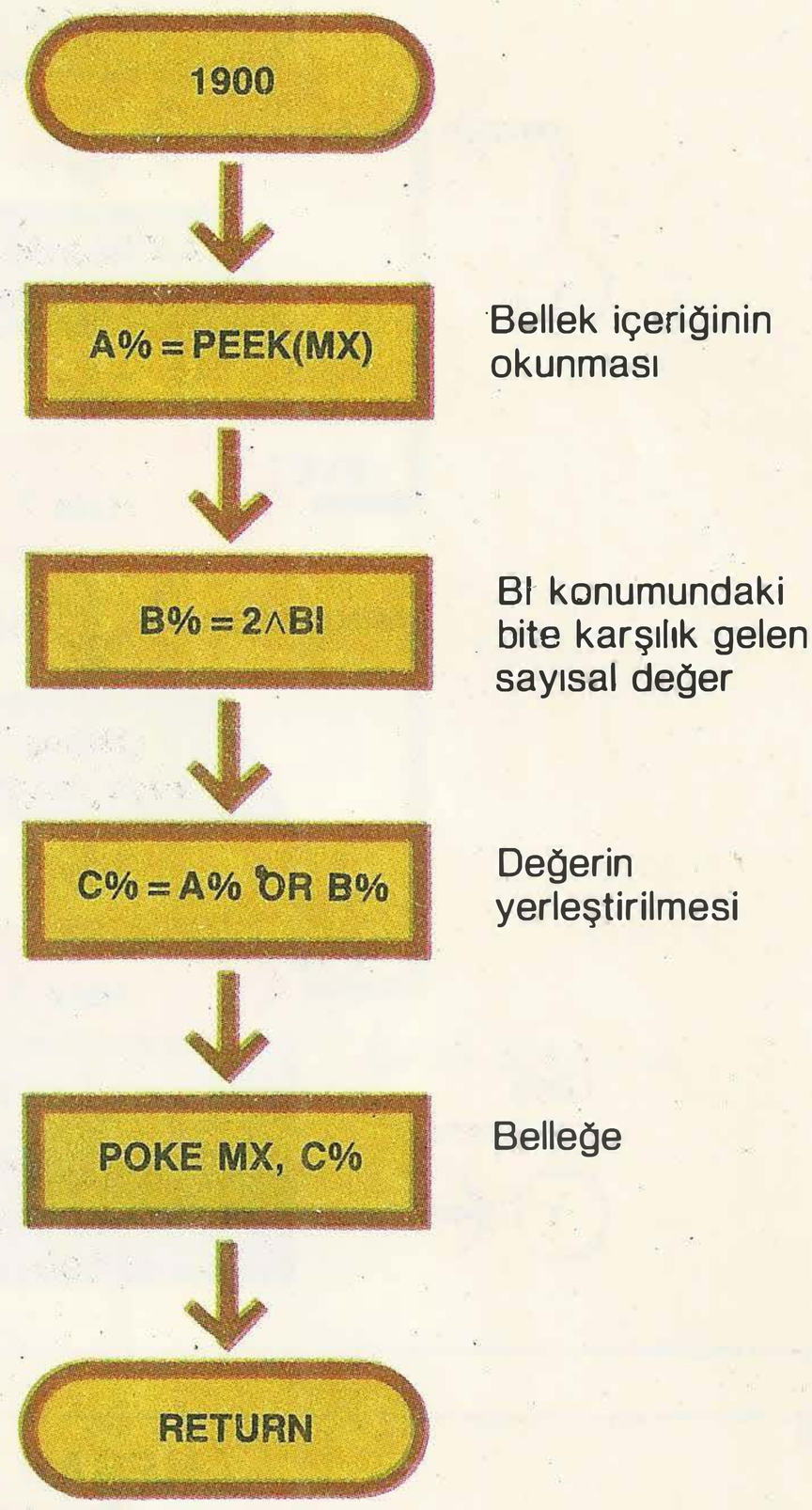
BİR BİTİN HAREKETE GEÇİRİLMESİNİN AKIŞ ŞEMASI

Girişler: MX = Bellek adresi
BI = Bit numarası

Örnek: BI = 3

	0	1	2	3	4	5	6	7	
A% =	0	1	1	0	1	1	0	X	Önceki içerik
B% =	0	0	0	1	0	0	0	X	2 ∧ BI = 8 ondalık
C% =	0	1	1	1	1	1	0	X	

Önceki bütün diğer bitler ve ayrıca 3 sayılı bit harekete geçirilmiştir.



te yazarak bir ekran noktasını harekete geçirir. Bitin harekete geçirilmesi, önceki durumu korumak için OR işlemcisi ile yapılır, yoksa bu harekete geçirme işlemi daha önce harekete geçirilmiş bitleri siler. Bu alt-program aynı zamanda bir noktanın durumunu da okuyabilir. Bu durumda yalnızca PEEK komutu kullanılır.

Bu teknik resim canlandırma (animasyon) programlarında kullanılır. Bir resmi hareket ettirmeden önce, daha önce harekete geçirilip geçirilmediğini anlamak için ekranın varış noktasındaki durumu kontrol edilir.

Bu veri sayesinde, uygulama programı, örneğin hareket ettirmeyi reddederek, uygun işlemi seçebilir.

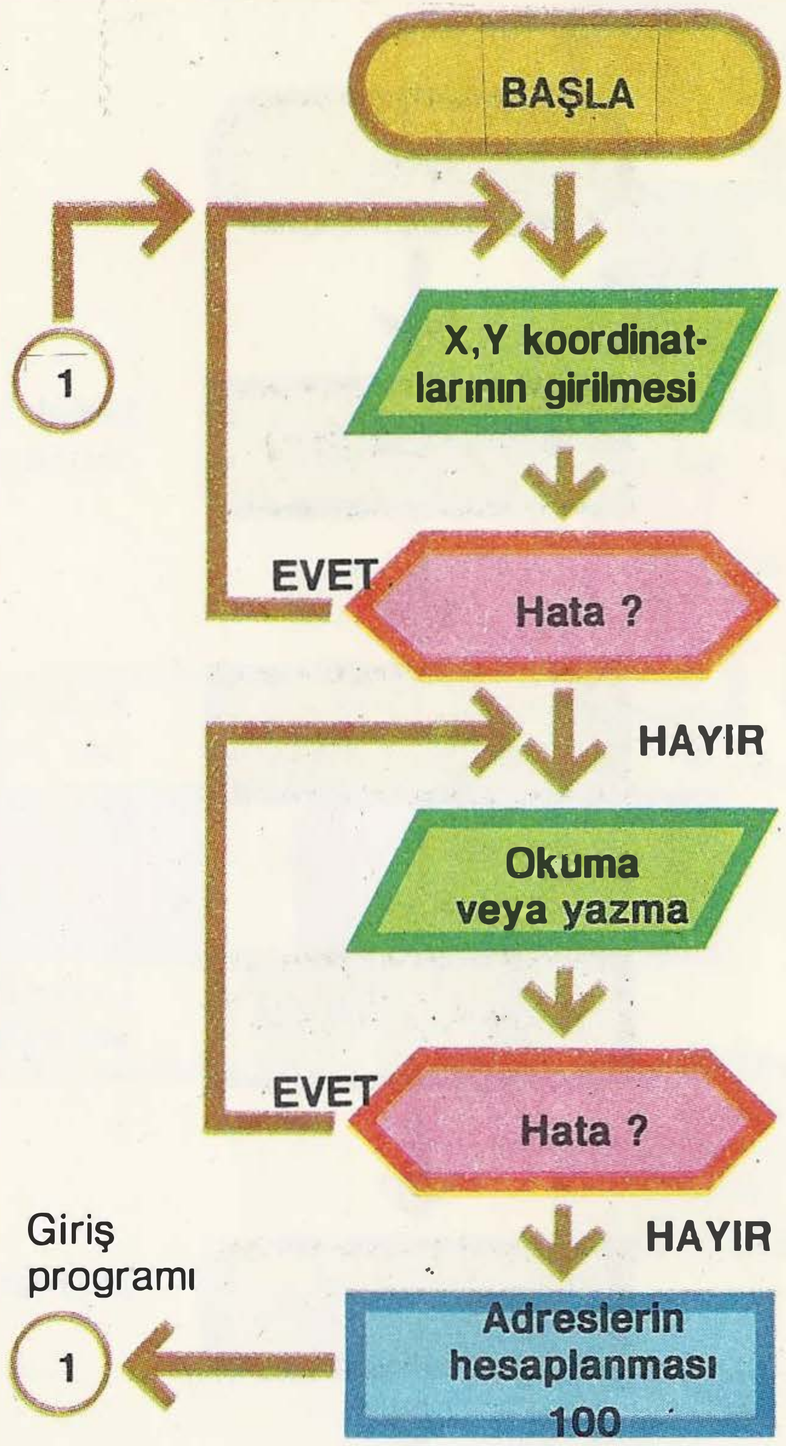
1540 ve 1541. sayfalardaki grafiklerde bu tekniği kullanan bir örnek programın akış şeması bulunmaktadır. Sayfa 1542'deki dökümde, daha önce sunulmuş olan alt-programları dışarda bıraktık. Ekran noktalarından oluşan yapı (''raster'' tekniğinin özelliği) bazı durumlarda çok yaklaşık bir görüntü üretir.

Sonuç, eksenlere göre eğimli bir doğru parçasının çizimidir (bkz. s. 1543). Bu, eğimden ötürü, doğru parçasının eğimleri (1 ve 3) eksenlerden birine paralel çizgilerle temsil edilmiştir: bu çizgiler eğime göre yatay veya dikey olabilir. Eğimin çok yüksek veya düşük olduğu durumlarda sütun veya satır değerini belirleyen bitler, birden çok nokta için aynı kalabilir. Oysa eğimin 45 derece civarında olduğu durumlarda, her nokta için farklı bir sütun ve satır belirlemek gerekir.

Belleğin Doğrudan Adreslenmesi

Bir şeklin hazırlanmasında (ya da bulunmasında) ekran belleğin adreslenmesinden yararla-

BİR GRAFİK BELLEĞİN ADRESLENMESİNİN GÖSTERİM PROGRAMI

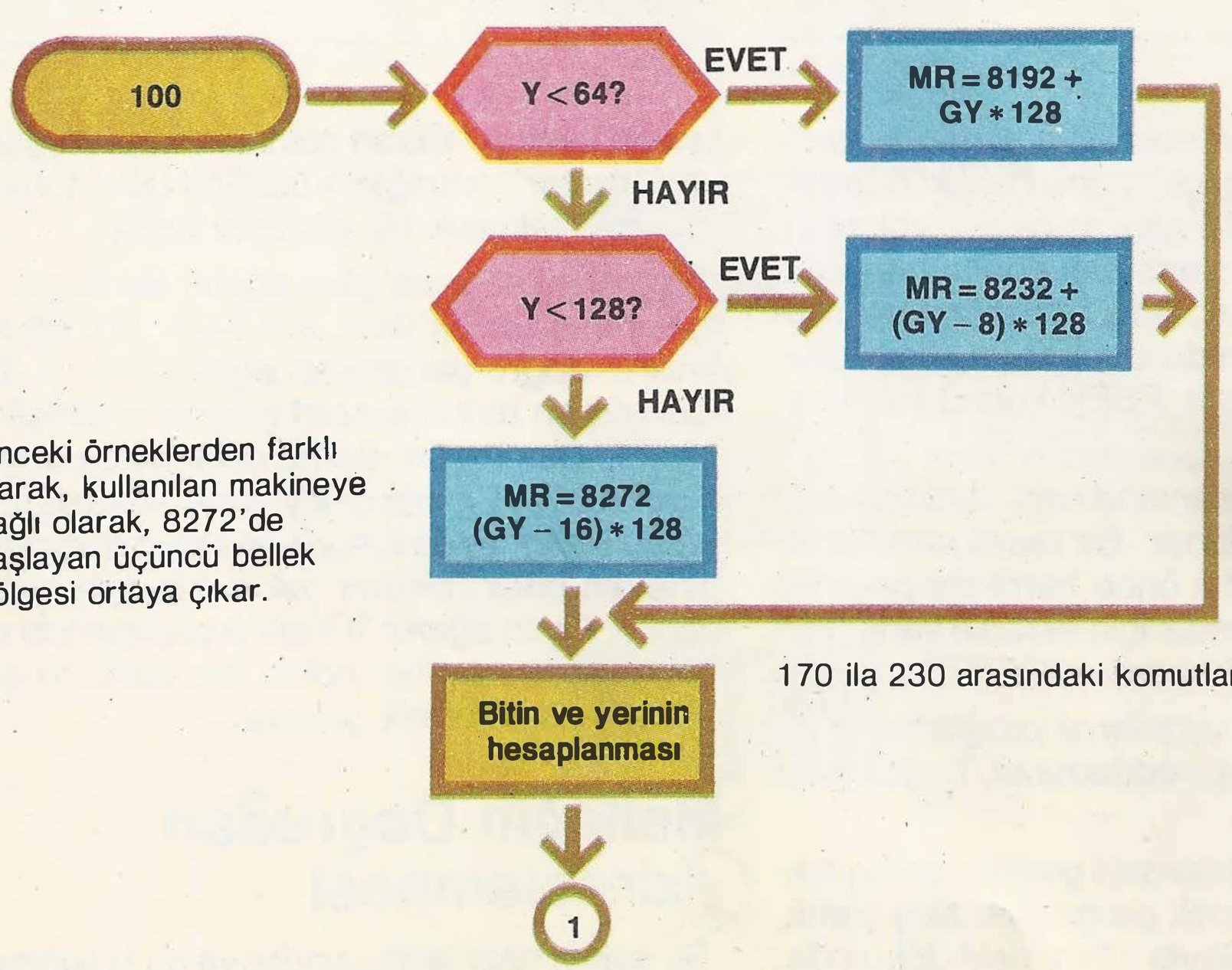


Ekran sınırları içindeki koordinatlar (0<X<279 ve 0<Y<159)

L (okuma) veya S (yazma) cevabı. Öbür karakterler ayrılmıştır.

100 alt-programı, iki ayrı alt-programda (1000 veya 1500) daha önce anlatılmış olan iki hesabı (Y ve X eksenleri) içerir (bkz. ayrıntılı akış şeması).

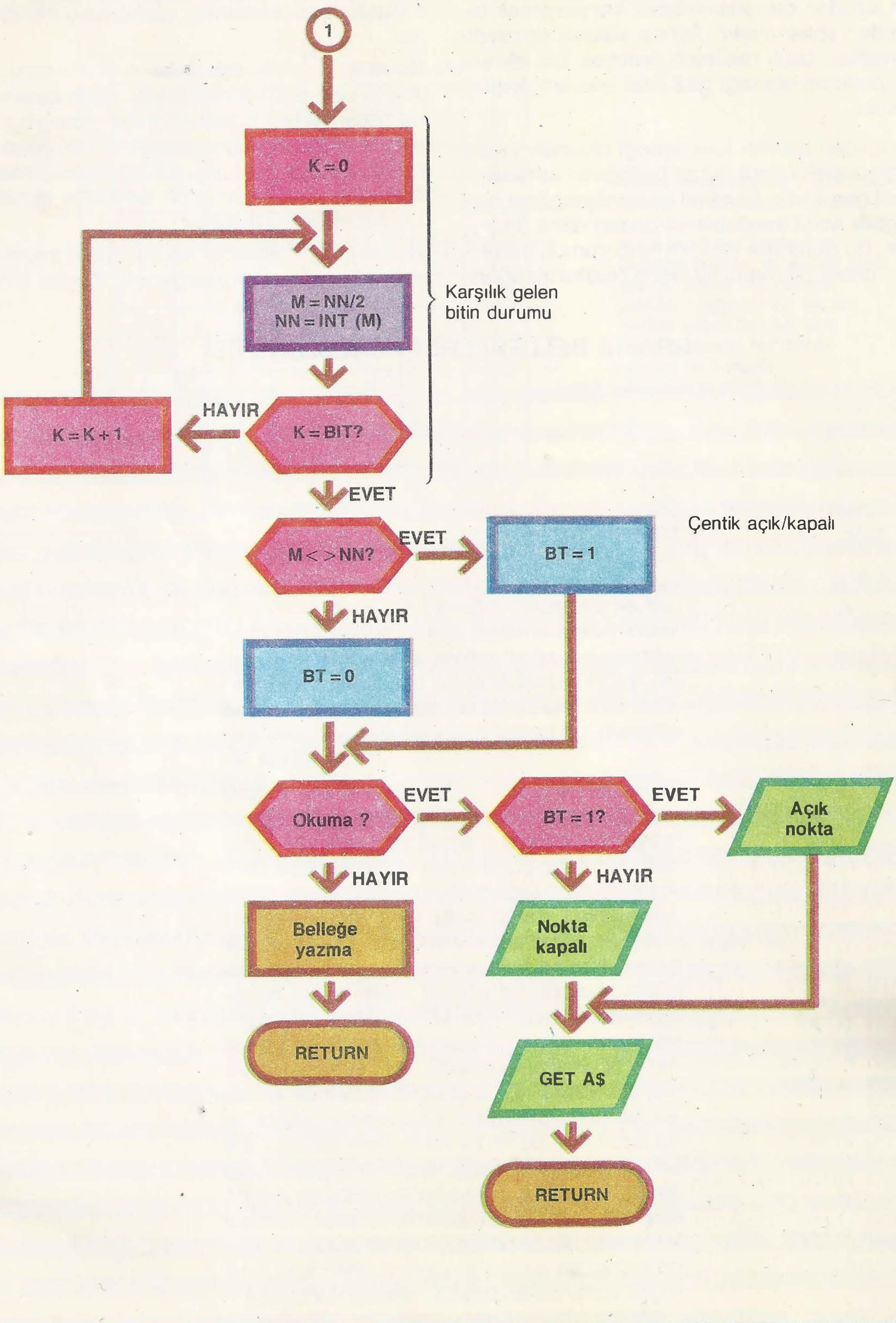
HESAP ALT-PROGRAMI



Bu ikisi, 1100'ün hesabıdır

Önceki örneklerden farklı olarak, kullanılan makineye bağlı olarak, 8272'de başlayan üçüncü bellek bölgesi ortaya çıkar.

170 ila 230 arasındaki komutlar



Karşılık gelen bitin durumu

Çentik açık/kapalı

nılması yüksek düzeyde komutların kullanılmasından çok daha fazla olanak sağlar. Bu komutlar, resim üretmek isteyen bir kullanıcının sıradan gereksinimlerini karşılayacak biçimde yapılanmıştır. Ayrıca yüksek düzeyde komutlar, olası resimleri üretmek için ekranı çözümleme olanağı gibi özel işlevleri öngörmezler.

Bu türden işlevler için; içeriği okumak ya da değiştirmek üzere ekran belleğinin adreslenmesi gereklidir. Ekranın çözümlenebilme olanağı ilk anda sanılabileceğinden daha önemlidir. Bu gereklilik kendini iki durumda hissettirir; grafik bir menü ile resim oluşturulduğunda ve başka aletler tarafından oluşturulmuş bir resim ele alınacağında (örneğin bir fotoğraf makinesi tarafından çekilmiş bir resmin sayısallaştırılarak işlenmesi sözkonusu olduğunda).

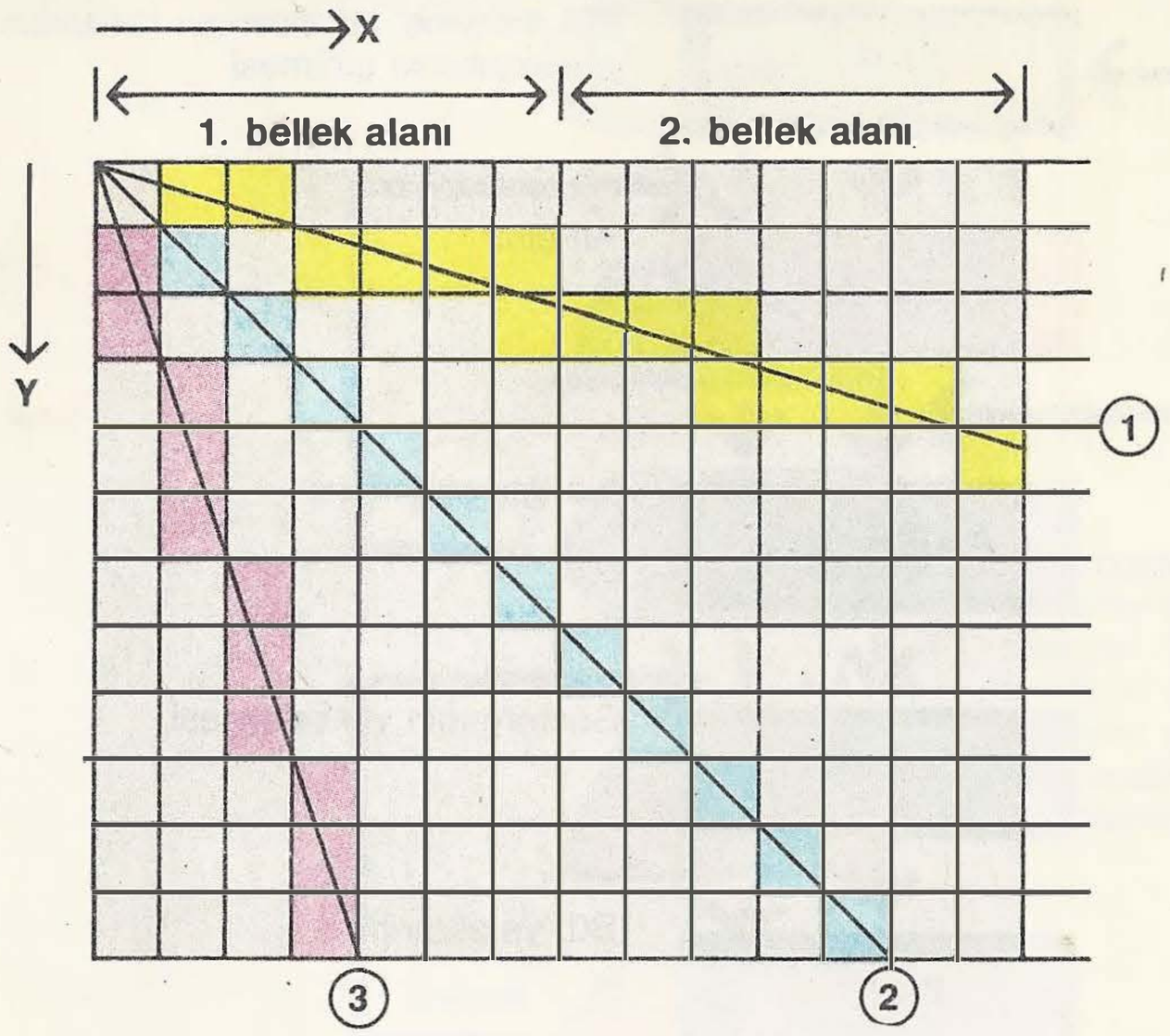
Bir resmin grafik menü ile oluşturulması, bir imlecin ekran üzerinde tuşlar ya da başka giriş sistemleriyle (paddle, grafik tablo vb.) hareket ettirilmesiyle gerçekleştirilir. Böylece ekranın boyutları ve kararlılık derecesi **(resolution)** dışında herhangi bir sınırlama olmaksızın resimler yaratılabilir.

Resim oluşturulduktan sonra, çeşitli tekniklerle belleğe yerleştirilmesi gerekir. Bunun için en

GRAFİK BELLEKLERİN ADRESLENMESİ

```
10 HGR
20 HOME
30 VTAB 23
40 INPUT "KOORDINATLARI GIRINIZ: ";X,Y
50 IF X < 0 OR X > 279 OR Y < 0 OR Y > 159 THEN 20
60 INPUT "OKUMA MI YAZMA MI? (O/Y)";MD$
70 IF MD$<>"O" AND MD$<>"Y" THEN 20
80 GOSUB 100
90 GOTO 20
100 REM ----------------------------------
110 REM ADRESLERIN HESAPLANMASI
120 REM ----------------------------------
130 GY = INT(Y/8)
140 IF Y < 64 THEN MR = 8192+GY*128:GOTO 170
160 MR = 8272+(GY-16)*128
170 A=Y-GY*8
180 MY=MR+A*1024
190 GX=INT(X/7)
200 MX=MY+GX
210 BIT = X-7*GX
220 A%=PEEK(MX)
230 B%=2BIT
240 NN=A%
250 FOR K=0 TO BIT
260 M=NN/2
270 NN=INT(M)
280 NEXT
290 IF M<>NN THEN BT=1:GOTO 310:REM
300 BT = 0:                          REM
310 IF MD$="O" THEN 370
320                                  REM YAZIM
330 IF BT = 1 THEN 360
340 C%=A%+D%
350 POKE MX,C%
360 RETURN
370                                  REM OKUMA
380 HOME
390 VTAB 23
400 IF BT=1 THEN PRINT "NOKTA HAREKETE GECIRILDI":GOTO 420
410 PRINT "NOKTAYA ULASILDI"
420 VTAB 24
430 PRINT "DEVAM ETMEK ICIN HERHANGI BIR TUSA BASINIZ":GET A$
440 RETURN
```

EKSENLERE PARALEL OLMAYAN DOĞRU PARÇALARININ GÖSTERİLMESİNDE EKRAN KARARLILIĞININ (RESOLUTION) ETKİSİ

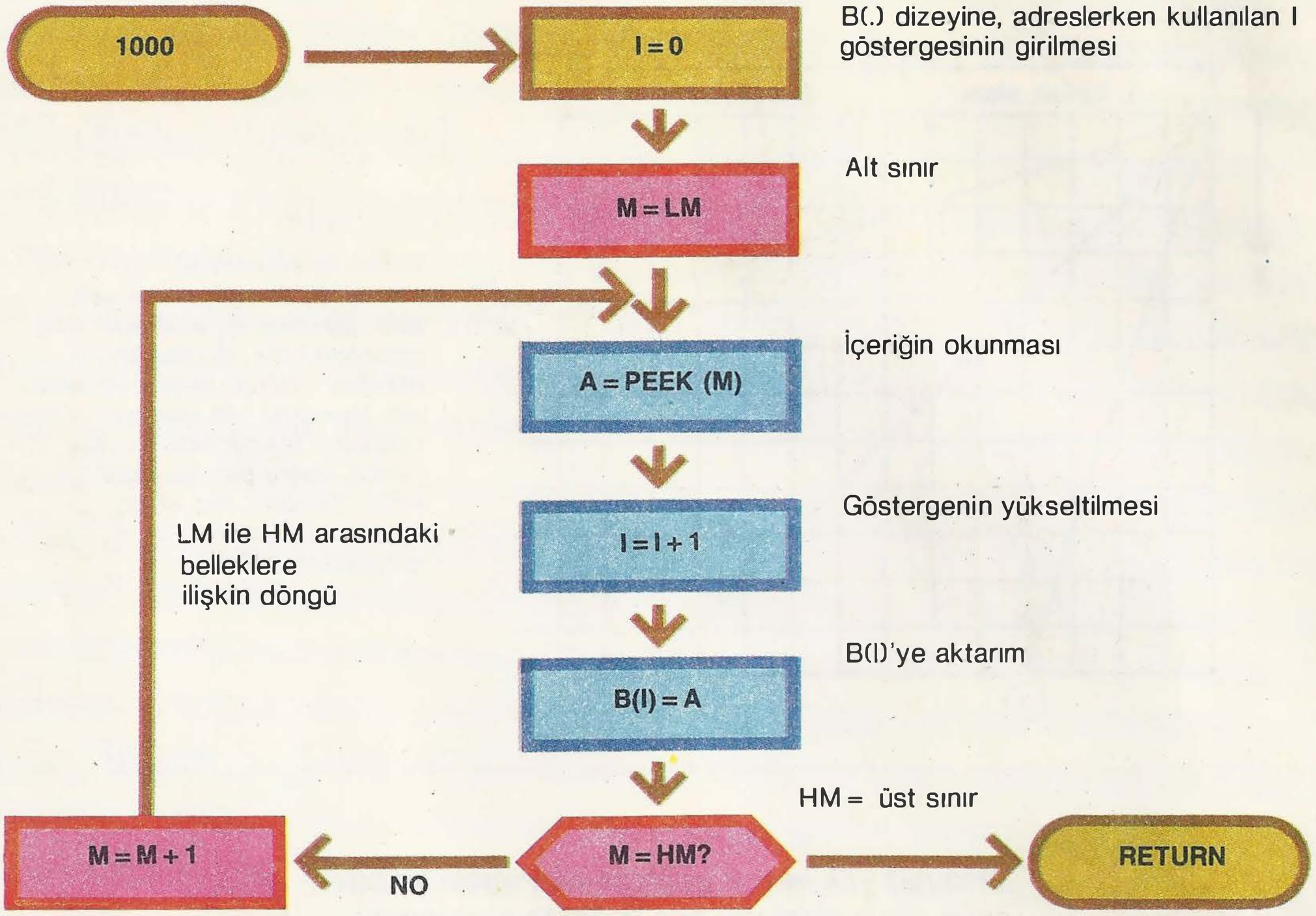


Minimum adresleme 1 bittir, bu bite bir ekran noktası karşılık gelir. Şemada üç adet eğik çizgi bulunmaktadır. Bunlardan yalnızca birinde, ekrandaki şekil tam bir eğime (45 derece) yaklaşır. Diğerlerinde bir dizi paralel doğru parçası elde edilir. Şemada, her ekran noktası bir kareyle simgelenmiştir.

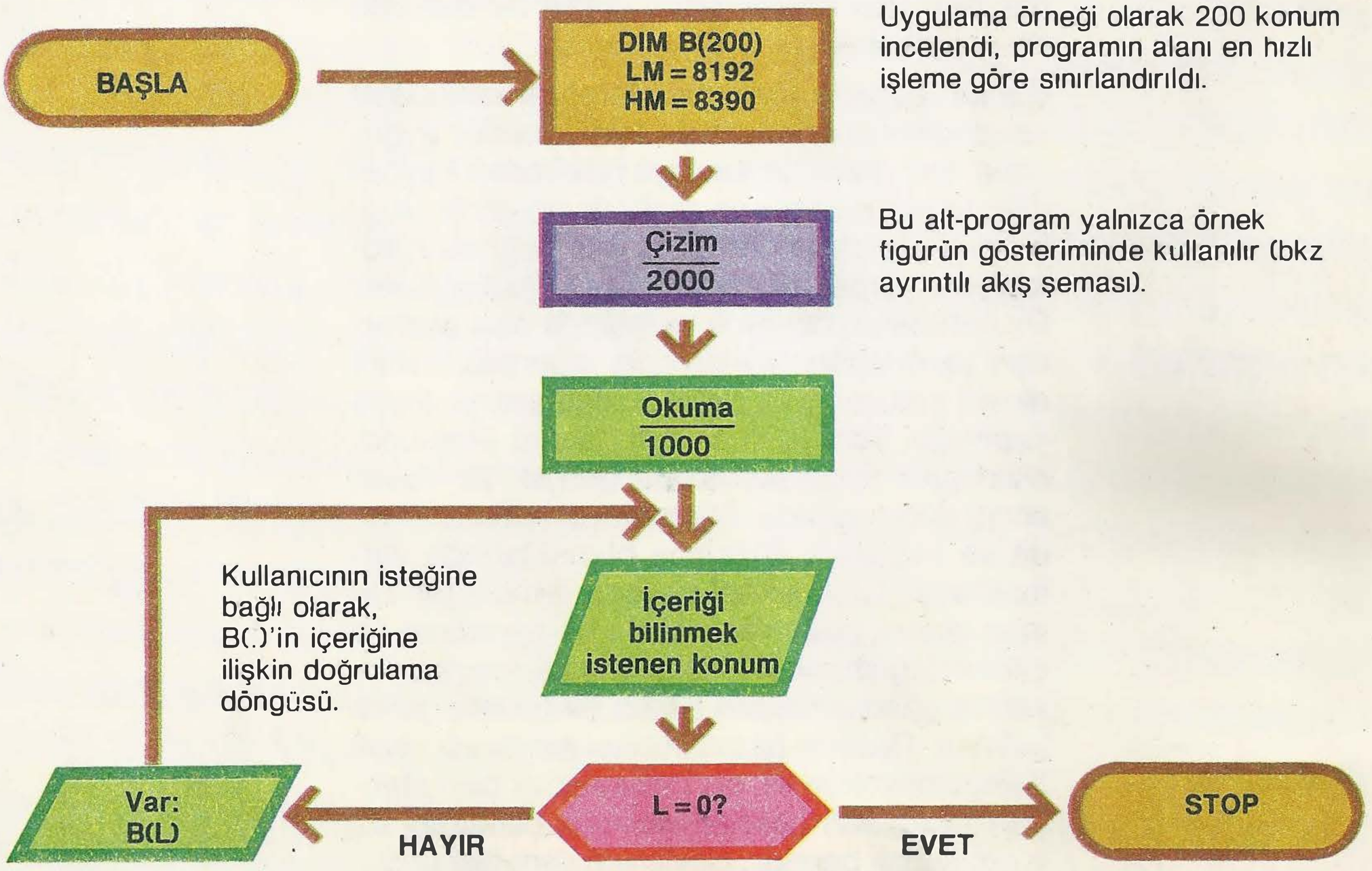
dolaysız yol, ekran belleğin okunması ve burada yazılı olanların ana bellekte saklanmasıdır. Böylelikle resim hakkında başka işlemlere tabi tutulabilecek sayısallaşmış bilgiler örneğin, ölçek ve rotasyon faktörleri; elde edilmiş olur. Bu bilgiler daha sonra bir diskette de saklanabilir.

Başka aletlerle (örneğin, fotoğraf makinesi) kaydedilmiş resimlere de aynı işlemler uygulanır. Asıl resim bir kamera tarafından kaydedilir, sayısallaştırılır ve ekranda gösterilir. Kullanıcı, bunu değiştirerek, daima doğrudan adresleme yöntemiyle olmak üzere, belleğe alır. Bu olanaklar, özellikle yaratımda özel aletlerden yararlanılan reklamcılık alanında, renkli ekran kullanımıyla birlikte bilgisayarla resim yapmada yeni ufuklar açtı. Sanat alanında, elektronik fırça sayesinde gerçek bir resim ekolü doğmuş oldu. Bilgisayarın katkısı, hızında ve sağladığı düzeltme olanaklarında yatmaktadır. Geleneksel yollarla önemli bir zaman alacak olan, renk ya da bir ayrıntının biçimini değiştirme gibi işler için bilgisayara bir komut göndermekten başka bir işleme gerek kalmaz. Resmin hazırlanması sırasında, son kompozisyonda sık sık kullanılacak bazı standart biçimlerin belleğe alınması olanağını da vurgulamak gerekir. Böylelikle sanatsal uygulamalar için bir grafik menü yaratılmış olur.

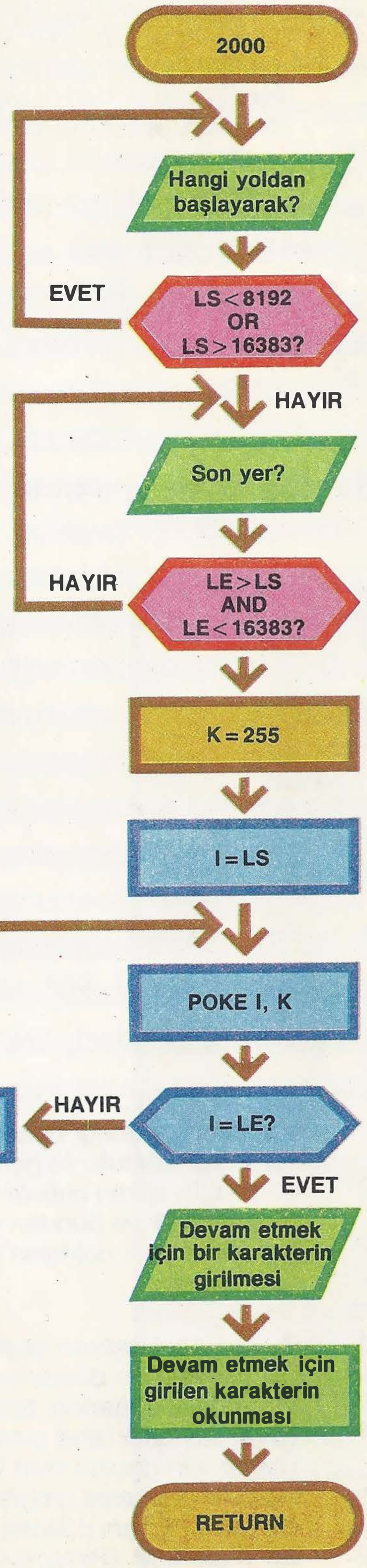
EKRAN BELLEĞİNİN OKUNMASI



Ana deneme programı



DENEME OLARAK DOĞRU PARÇASININ HAZIRLANMASI



Alt-program yalnızca eğitici amaç taşımaktadır ve yatay doğru parçalarının gösterimiyle sınırlıdır. Dolayısıyla iki bellek yerinin aynı satıra ait olması gerekir. Çünkü aynı satıra ait olmayan adreslerin girilmesiyle, bir alt ve üst sınır arasındaki bütün ekran noktaları harekete geçirilebilir.

0'dan 7'ye kadarki bitlerin hareketi

Seçilen grafik belleklere 255 değerini yazan döngü

Siprel 2010, Apple denkleri versiyonu

```
 50 HGR
 100 DIM B(200):LM=8192:HM=8390
 110 GOSUB 2000
 120 GOSUB 1000
 130 HOME:VTAB 22:INPUT "BELLEK BOLGESINI GIRIN ";L
 140 IF L=0 THEN TEXT:END
 150 HOME:VTAB 22: PRINT "DEVAM = "B(L)
 155 PRINT "DEVAM ICIN BIR TUSA BASIN ";GET Q$
 160 GOTO 130
 1000 REM ==================
 1001 REM * OKUMA
 1002 REM ==================
 1003 :
 1010 I = 0
 1015 FOR M = LM TO HM
 1020 A=PEEK(M):I=I+1
 1030 B(I)=A
 1040 NEXT M
 1050 RETURN
 2000 REM ==================
 2001 REM * CIZIM *
 2002 REM ==================
 2003 REM
 2010 HOME:VTAB 22:INPUT "ILK BOLGE ";LS
 2015 IF LS<8192 OR LS>16383 THEN 2010
 2020 INPUT "SON BOLGE ";LE
 2025 IF LE > LS AND LE<16383 THEN 2040
 2030 GOTO 2020
 2040 K=255
 2045 FOR I = LS TO LE
 2050 POKE I,K
 2060 NEXT I
 2070 HOME:VTAB 22: PRINT "BIR TUSA BASINIZ ":GET Q$
 2080 RETURN
```

Grafik İmgelerin Saklanması

Bir grafik imgenin ilk adımı, istenen imgenin ekranda elde edilmesidir.

Bunun için bazı makinelerde özelleşmiş komutlar bulunmaktadır. Olivetti M20'de POINT (X,Y) komutu, X,Y koordinatlarındaki noktanın rengini ve durumunu belirler. Aynı makinede ayrıca tüm ekran sayfasının belleğe alması ve gösterilmesi için komutlar vardır. Öbür makinelerde alt-programların, kullanıcının kendisi tarafından yazılması gerekir.

En basit yöntem, PEEK komutu ile, ekran belleğindeki her yerin içeriğinin elde edilmesi ve kullanıcının belirlediği bir dizeye (tabloya) aktarılmasıdır. Dolaysız olan yol dışında başka adresleme olanağı bulunmadığında, bu okuma PEEK komutu ile gerçekleştirilir. Bir başka deyişle, ekran belleğe hiçbir sembolik isim verilmemiştir ve bundan yararlanamayan kullanıcının da her noktanın adresini ayrı ayrı vermesi gerekir.

Bu işi gerçekleştiren alt-program, yalnızca ekran belleğin en düşük ve en yüksek sınırları arasındaki değerleri tarayan bir döngüdür. 1544. sayfadaki akış şeması bunun çeşitli makineler için geçerli olan bir örneğini vermektedir. Ana deneme programında (sayfa 1546'daki alt-program dökümüne bakınız), kullanılan makineye (Personel Kid, Siprel, Siprel 2010) uygun olan adreslere, 8192 ve 16383'e,

başvurulur. Diğer bilgisayarlar için, kullanım klavuzunda verilen bilgilere göre değerlerin değiştirilmesi gerekir. Dökümde, makineyle doğrudan ilişkili iki komut görülebilir: HIMEM ve LOMEM. Bu komut türü, programın kapladığı alanı gösterir. Yani bu alan sistemin donanımı ve mantığına göre HIMEM (en yüksek bellek adresi) ve LOMEM (en düşük bellek adresi) ile sınırlıdır.

Sistemin RAM'ı dört işleve göre bölünmüştür, özetle:

- İşletim sistemi, örneğin DOS
- Özel işlevler (örneğin metin bellek ekranı)
- Grafik bellek
- Kullanım alanı

Kullanılan sistemde, (0'dan 1023'e) ilk 1024 adres sistem tarafından kulanılır (noktalayıcılar, arabellekler.....), ardından (1024'den 3071'e) metnin ilk iki sayfası, (3072'den 8191'e) serbest alan ve son olarak (8192'den 16383'e) grafiğe ayrılmış olan alan gelir.

Kalan bellek alanı DOS ile kullanıcı programları arasında paylaştırılmalıdır. DOS her zaman en üstte kalan alana yerleştirilir; böylece, kullanıcı programına, grafik sayfanın sonuyla, DOS'un başı arasındaki alan kalır (bu sayfadaki şema).

Bu sistemde ayrıca, birinciden bağımsız olarak yönetilen ve gösterilen ikinci bir grafik sayfa vardır (16384'den 24575'e kadar olan yer-

SİPREL 2010'DA BELLEĞİN BÖLÜMLERİ



Kişisel Bilgisayarların Anatomisi

Kişisel bilgisayar kullanan biri programlama dilleri konusunda bir şey bilir. Buna karşılık çoğunlukla makinenin yapısını tanımaz. Çünkü, bilgisayar şasisi çoğu kez aşılamaz bir engel olarak görülür.

Ancak makine ve parçaları karşısındaki bu bilgi eksikliği can sıkıcı sonuçlara yol açabilir, çünkü bozukluklar ya da en beklenmedik anlarda sistemin işleyişindeki terslikler çıkabilir.

Bilgisayarındaki maddi sorunu giderme gerekliliği ile karşı karşıya kalan biri onarım merkezlerinde bozukluğun ne özensizce teşhis edildiğini ve onarım süresinin ne uzun olduğunu bilir.

Burada kişisel bilgisayarın iç yapısının bir televizyon ya da radyo devresinden bile daha basit olduğunu hatırlatmakta yarar var. Çünkü televizyon ve radyo gibi aletler sayısal değil, benzeşik sinyalleri işleme sokar.

Bir gün devreleri örten kapağı açmaya karar veren cesur bilgisayarcı, elemanların azlığı ve sistemin çapı karşısında şaşkınlığa düşer. Her fonksiyonun birbirinden iyice ayrılmış, kolay ayırt edilebilir ve ilk bakışta temel bağlantılarla birleşen parçalar (kartlar) tarafından gerçekleştirildiğini görür.

Her kartta entegre edilmiş olan parçalar lehimle değil, bu iş için hazırlanmış yarıklara basınçla sokularak tutturulmuştur.

Bu koşullarda onarımda, teşhis için en çok 15 dakika, birkaç bin liradan fazla tutmayan sözkonusu entegre devrenin değiştirilmesi için de 1 dakikadan az zaman yeterlidir.

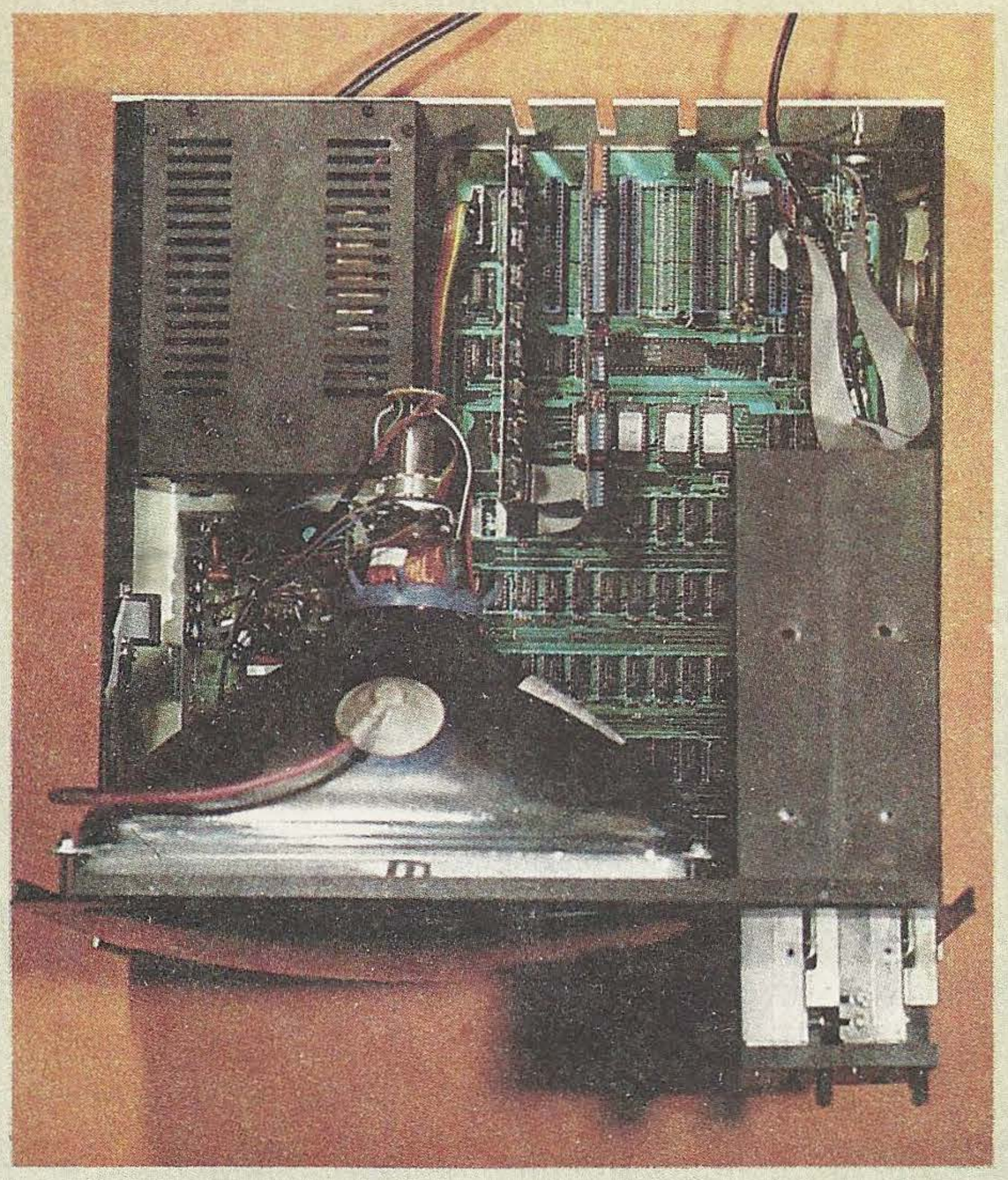
Söylediğimizin doğruluğunu görmek için bilgisayarımızın iç yapısına bir göz atmaya değer.

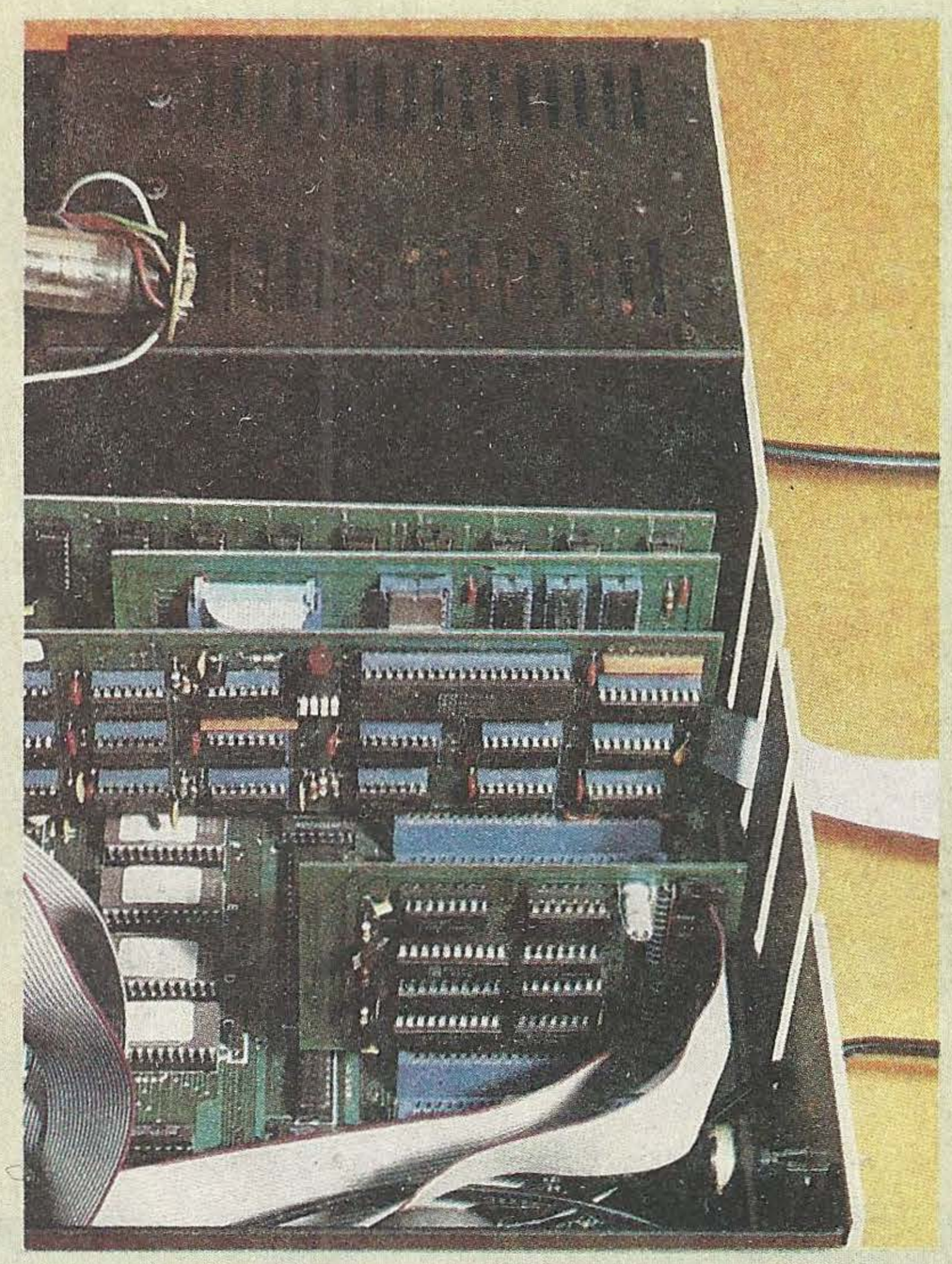
Sip rel 2040 sistemi (Personal Kid), kişisel bilgisayarlar mimarisinin genel incelemesine çok uygundur. Ekranın, sürücülerin ve kartların aynı yerde bulunduğu, dış bir klavyeye bağlanan bir makinedir bu.

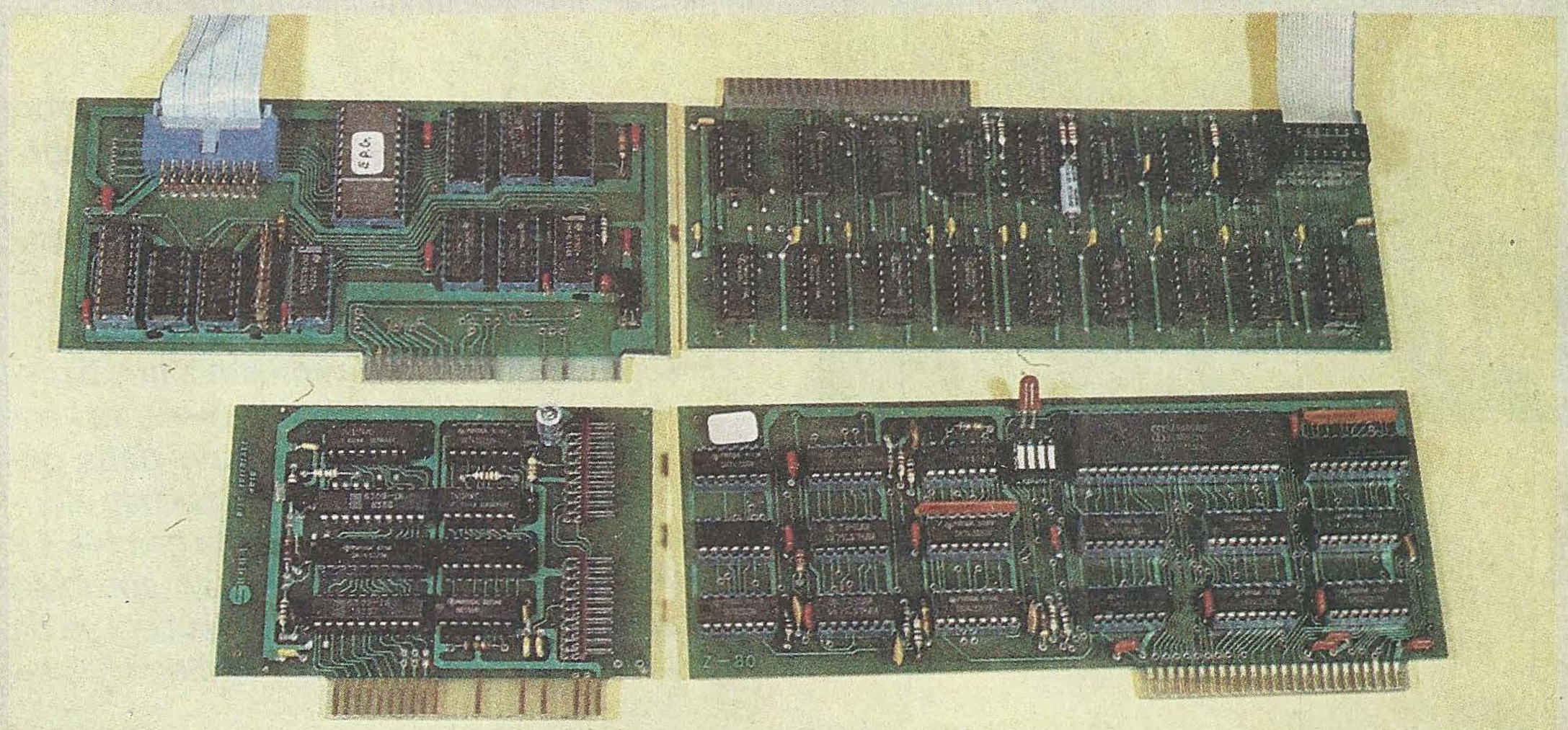
Bu sistemin birimlerin ayrı ayrı olarak bulunduğu varyasyonları varsa da, biz ilk bakışta ulaşılmaz görünen bu biçimini inceleyeceğiz.

Aşağıdaki ilk fotoğrafta, iki sürücüyü çıkarmak için şasi kapağını kaldırdık: Bunun için 8 vidayı sökmek yetti. İkincisi (s. 1549'un yukarısı) sistemi yukarıdan gösteriyor. "Slim" türü sürücülerden başka, burada, katotlu tüp ve yukarıda transformatörün havalandırılması için üzeri yarıklı bakım kutusu iyice görülüyor. Geride ana fiş ile çeşitli fonksiyonel kartları (fotoğrafın üst bölümünde görülebilir) bağlama yarıkları (slot) seçiliyor. Üçüncü fotoğraf monte edilmiş kartlarla, ön planda, iç yüzey fişiyle birlikte sürücüleri ve buradaki her disk birimi için bir adet olmak üzere iki yassı kabloyu gösteriyor. İkinci planda merkezi birimi (Zilog Z80) içeren kartın bir bölümü ve iç yüzey kartı ile ekran ve RAM bellek genişleme kartı ayırt ediliyor. Aynı kartlar 4. fotoğrafta da var: solda yukarıda 80 sütunlu ekran yönetim kartı, yanında bellek genişleme, aşağıda solda sürücülerle iç yüzey kartı ve yanında Z80 geliştiricili kart görülüyor. 5. fotoğraf (s. 1550), fonksiyonel kartlarla ana fişin ayrılmasından





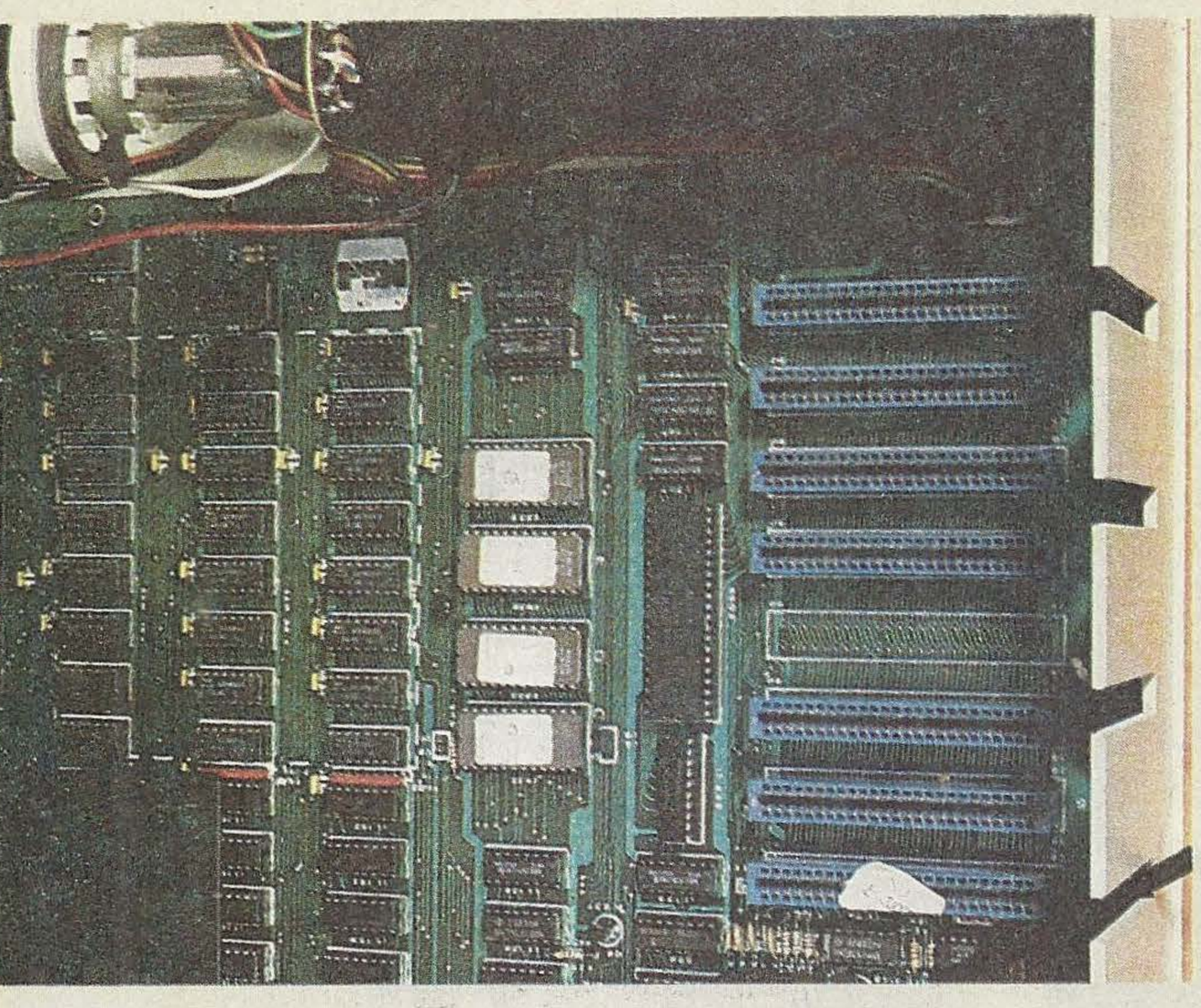


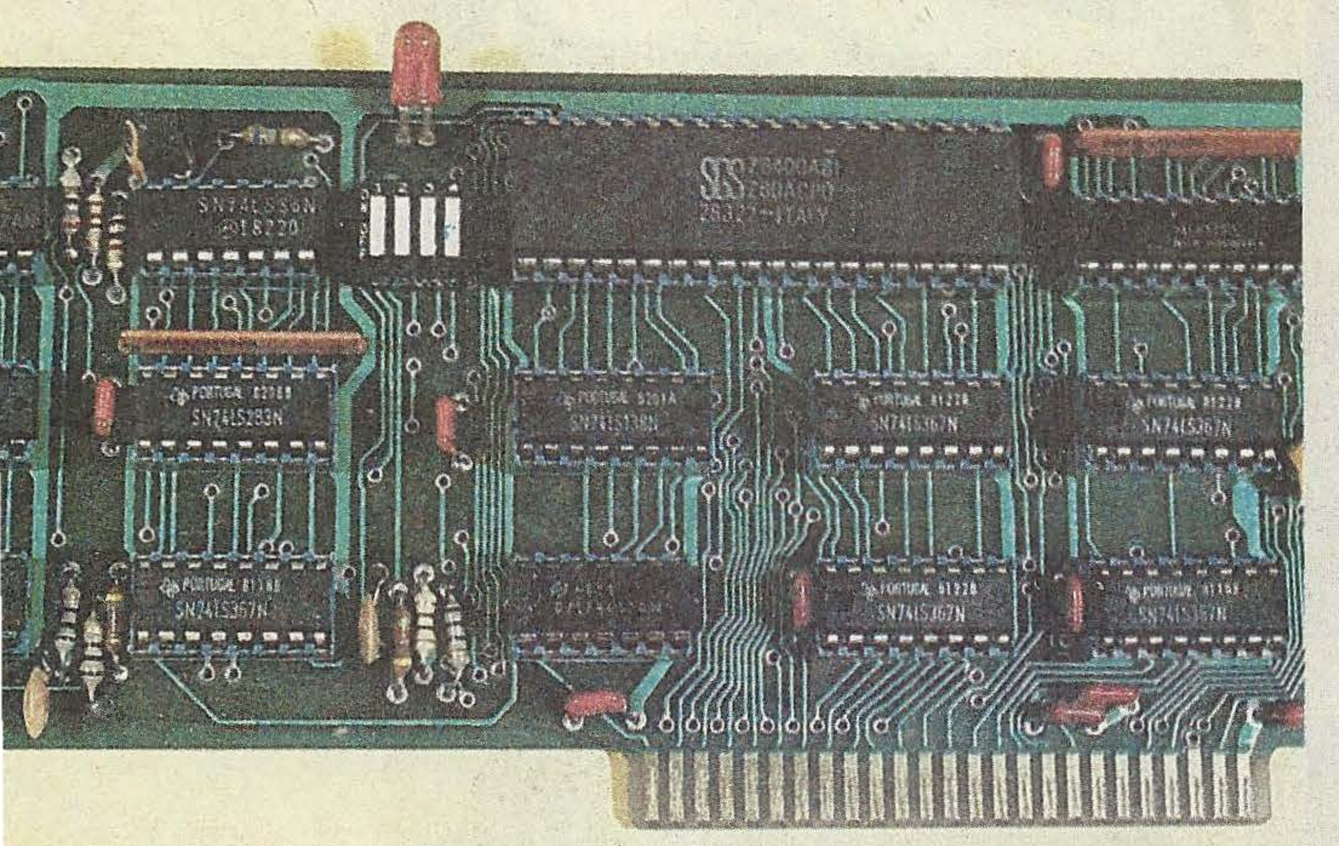


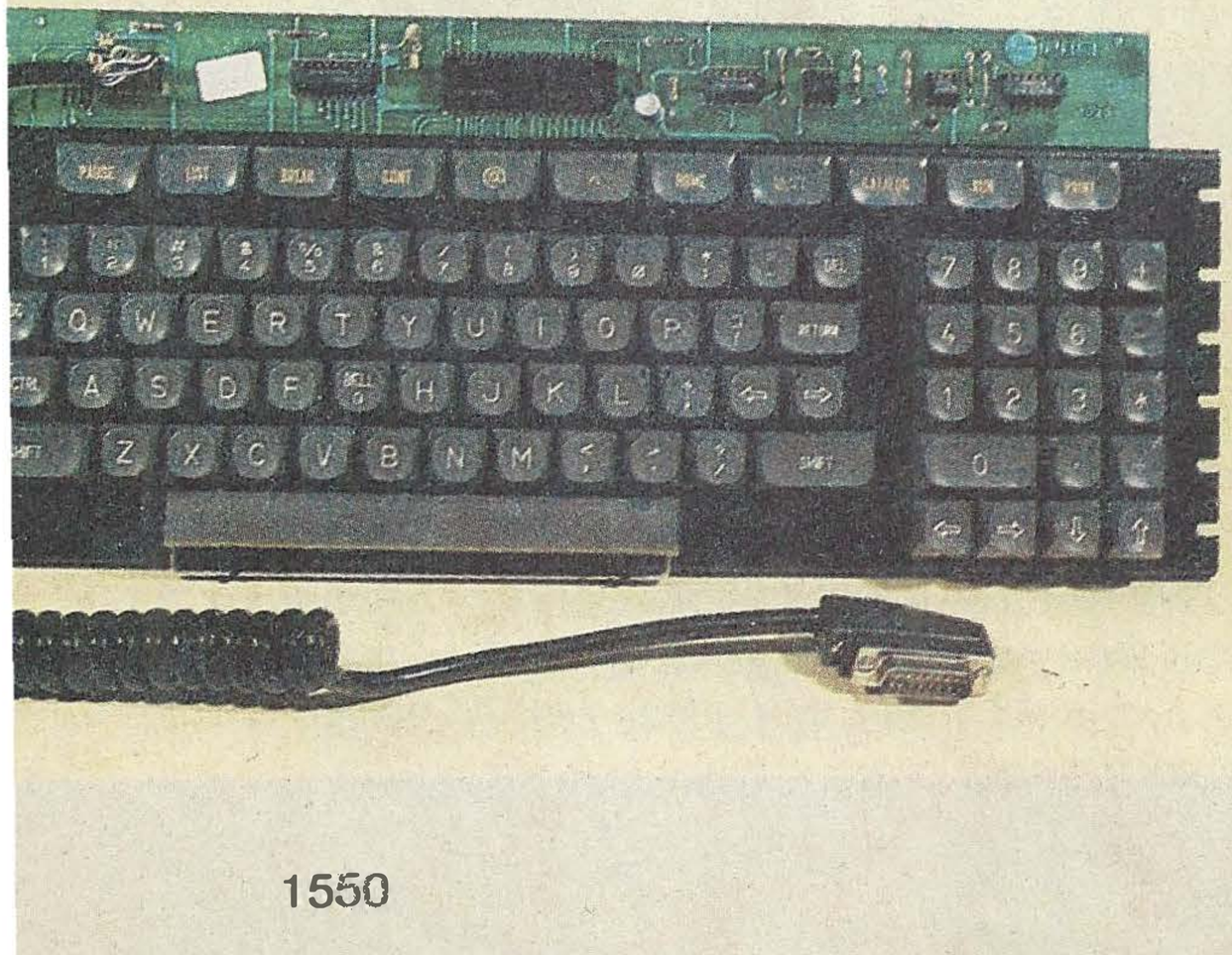
sonraki bir iça yrıntıyı gösteriyor. Şimdi 7 yarık iyice ortada, aşağıda ise çevresel gereçlerin (joystick, grafik tablet, paddle, vs.) bağlantı topacı seçilebilir. Örneğimizde topaca bir grafik tablet bağlanmış: bağlantı tablosu fotoğrafın sol yanından çıkıyor.

Ötekilerin ters yönünde bulunan büyük boyda entegre devre MİB (Rockwell 6502), resmin sol yanındaki daha küçük entegre devreler ise RAM bellekleridir.

İçersinde işletim sistemi bulunan, etiketlenmiş dört entegre devre ise, programlanabilip silinebilen ve yalnızca okumaya yönelik bellekler olan EPROM'dur. Pencereleri kaplayan etiketler, mor-ötesi ışınlarla belleklerin içeriğinin silinmesine olanak verir ve uzun süre gün ışığına açık kaldığında yanlışlıkla olabilecek silinmelere engel olur.







Siprel sistemi temel olarak DOS'ta çalışan ve normal olarak CP/M altında çalışan bir Zilog işleyicisiyle tamamlanabilen Rockwell 6502 merkez biriminden (Apple ve Commodore 64'-ün aynı) yararlanır (ortadaki foto). 6502 işlemcisi, üstünde Z80 içeren ikinci bir kartın sokulabildiği ana kartın içinde bulunur. İsteğe bağlı olarak eklenebilen bu kartın sokulmasıyla bütün fonksiyonel birimler (bellekler, G/Ç tertibatı, vb.) yeni merkez birimler tarafından çoğaltılabilir ve böylelikle de CP/M altında çalışan bir sistem elde edilir.

Bir işleyiş biçiminden ötekine geçiş otomatiktir. Çalıştırmaya başlarken sistem, 6502 işlemcisi ile, DOS altında işlemeye hazırdır. CP/M disketinin sürücüye sokulması ve klavyeyle (IN = 6) değişme komutunun verilmesiyle, DOS'un yerini CP/M alır ve sistem Z80 biriminin kontrolüne girer. Bu ikililik makinenin, en çok kullanılanlardan şu ya da bu işletim sistemiyle yazılmış bütün programları yerine getirmesine olanak verdiği ölçüde pratiktir.

Son fotoğraf, sistemin klavyesini gösteriyor. Ortada tertibatın yönetimine ilişkin entegre devre görülüyor. Kablo bağının yanındaki daha küçük entegre devre, temel hat devresini yalıtmaya yarar.

Böylece bilgisayarcılara iyi düşünülmüş bir işlem sisteminin yalınlığını gösterebilmiş ve kendi paylarına gereçlerin anatomisini daha derinden inceleme isteğini uyandırabilmiş olduğumuzu umuyoruz. Buradan yola çıkarak bir bilgisayarda karşılaşılması olağan sorunları herkesin tek başına pekala çözebileceği ya da hiç değilse, olası bir onarımı yönlendirebileceği anlaşılabilir.

ler). Böylece kullanıcı programı 24576'da başlayıp DOS'un başlangıcından önce (40960. yer) sona erer.

HIMEM ve LOMEM parametreleri, bu sınırları, grafiğe ayrılan alana girilmesini önlemek için koyar.

Program çok az dolu olduğunda bu iki parametreye gerek duyulmaz, bunlardan sözetmemizin nedeni, grafik uygulamalarda özellikle önem taşıyan, belleğin dolması sorununa dikkati çekmek içindir. Bu arada programımız, belleğe alma ve grafik sayfaların çağrılması için bazı ek fonksiyonları da gereksinir.

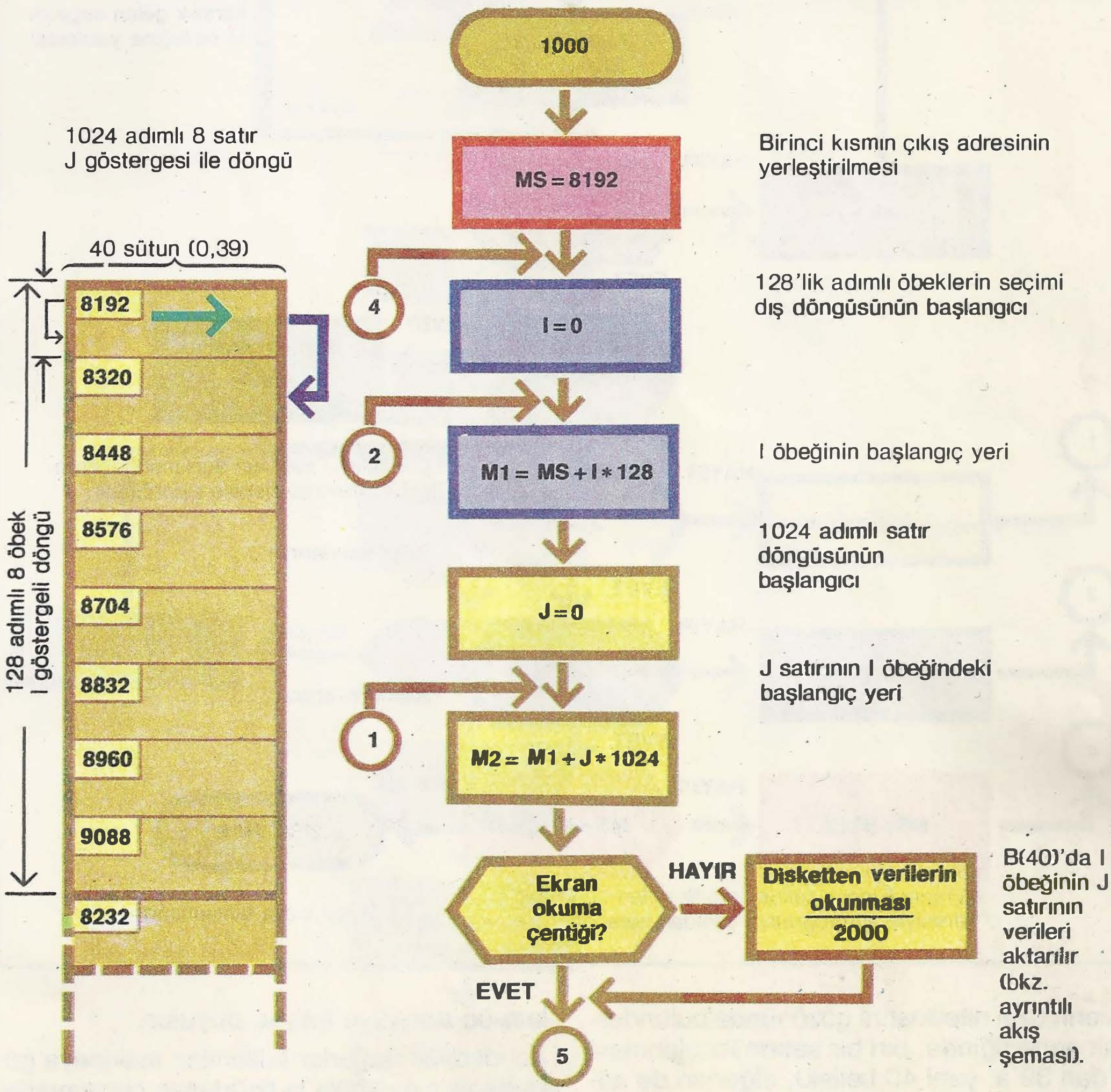
En avantajlı yöntem, her defasında ancak 40 bellek bölgelik bir satır içeren bir ara belleğin kullanılmasıdır her yeni satırın içeriği diskete aktarılır ve böylece ara bellek, sonraki satırı içine alacak biçimde hep boş bırakılır.

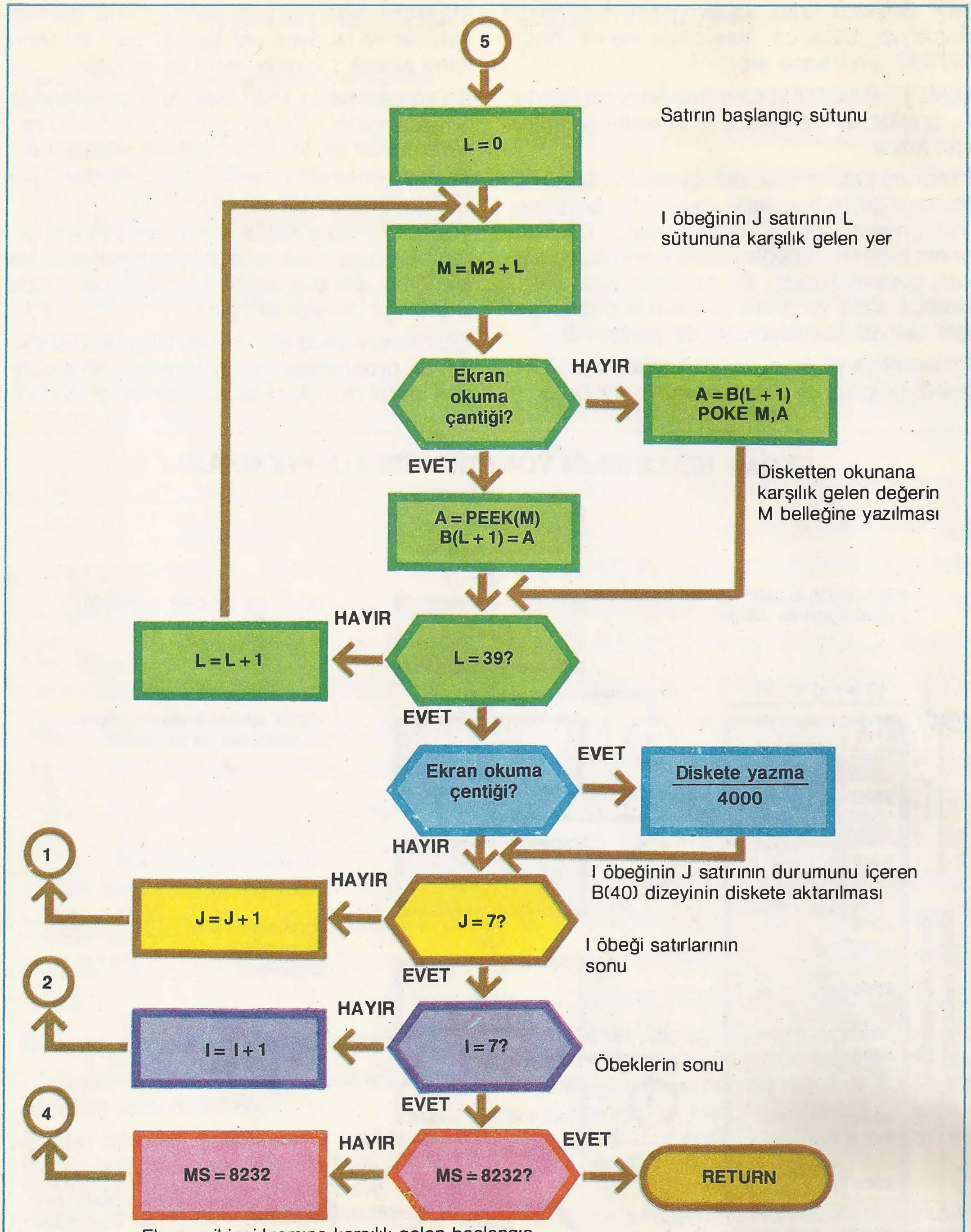
Bu yöntem sayfa 1551'deki akış şemalarında örneklendirilmiştir. Programın sınanması için ekranda bir resim olması gerektiğinden (noktaları alt-programda verilir), döküme resim yaratma alt-programı da eklenmiştir.

1000 alt-programında kullanılan yöntem ekran belleğinin satır satır incelenmesi ve B'nin içeriğinin stoklanmasıdır: her satırın sonunda B tablosu diskete aktarılır.

Eğer taranmakta olan bellek bölgeleri bitişikse, alt-programlar tek bir tarama döngüsünden ibaret olur. Ama bunun yerine, grafik bel-

EKRAN BELLEĞİNİN YÖNETİLMESİ ALT-PROGRAMI



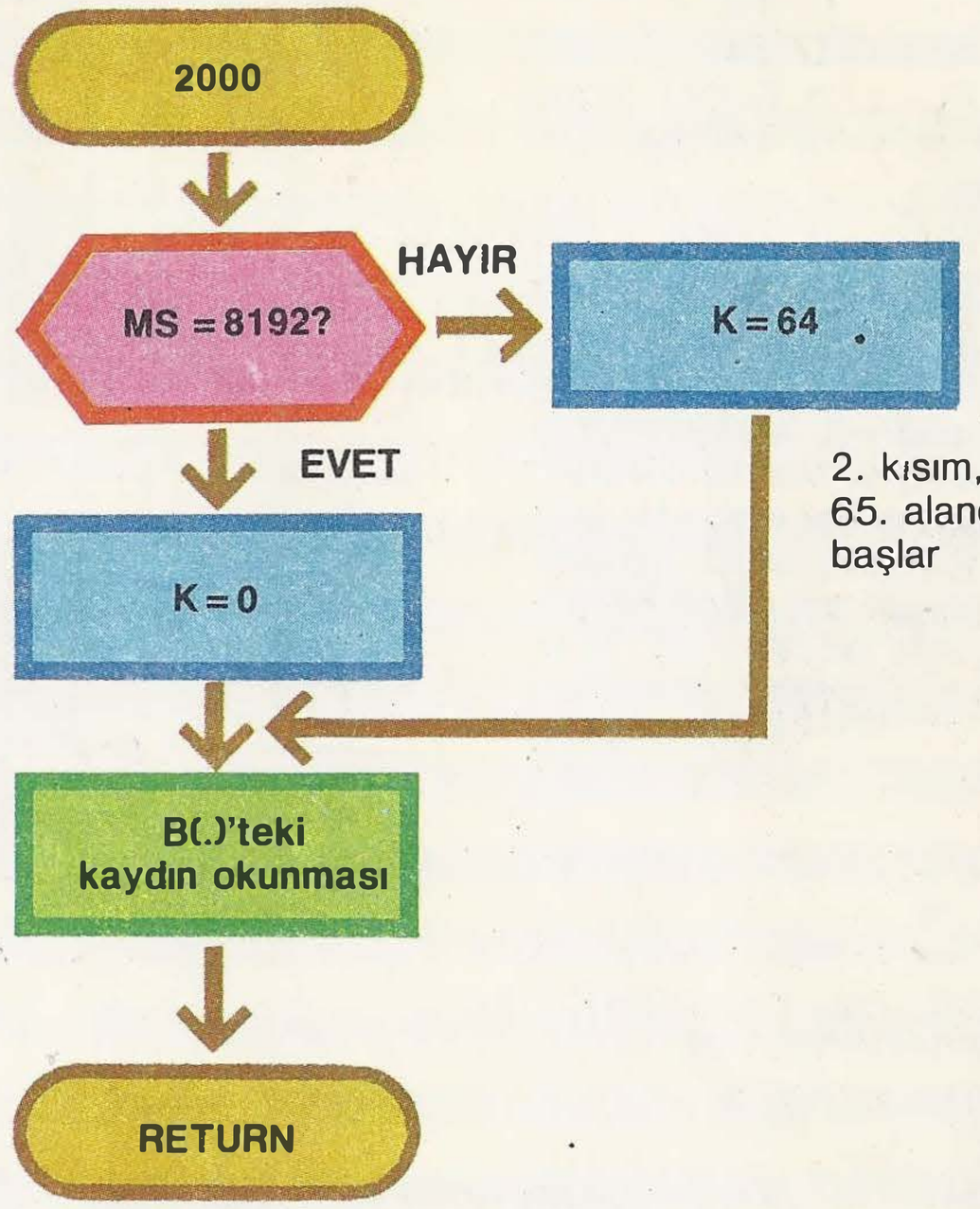


leklerin özel niteliklerini gözönünde bulundurmak gerektiğinde, biri bir satırın incelenmesi (0'dan 39'a, yani 40 bellek), diğerleri de adreslerin hesaplanması için olmak üzere toplam üç döngüye ihtiyaç duyulur.

Kaydedilen değerler kullanılan makineye göre değişir ve Apple'la bağdaşan bilgisayarların yaklaşık hepsinde aynıdır; bu nedenle bu

alt-program hiçbir değişme gerektirmeksizin bunların hepsinde uygulanır. İzlenen mantık, belleğin ilki 8192. öbürü de 8232. yerinde başlayan her iki bölümünün de taranmasından ibarettir. Alt-program çağrıldığı zaman, iki yerden başlayarak iki kez çalışır. Her bölüm sı-

DİSKET OKUMA



Yazma alt-programı da aynıdır; yalnızca G/Ç fonksiyonu değişir.

GRAFİK FİGÜRLERİN DİSKETTE SAKLANMASI

```
1   REM ========================
2   REM * CIZIM           *
3   REM * VE DISKETTE     *
4   REM * SAKLAMA ICIN    *
5   REM * PROGRAM         *
6   REM ========================
7   :
30  :
40   HIMEM: 8191
45  :
50   DIM B(40):Y = 90: X = 140: S =
      5
55   D$ = CHR$(4)
60   PRINT D$ "CIZIMI ACIN"
65   HCOLOR = 3
100 HOME: VTAB 22: INPUT "OKUMA/YAZMA
                  (O/Y)?";A$
110 IF A$ = "O" THEN FL = 0:HGR
      :GOTO 150
120 IF A$= "Y" THEN FL = 1:HGR
      :GOTO 140
130 IF A$= " " THEN 200
135 GOTO 100
140 GOSUB 3000
150 GOSUB 1000: GOTO 100
200 PRINT D$"CIZIMI KAPA":TEXT
      :END
999 :
1000 REM ========================
```

```
1001 REM * EKRAN BELLEGI
1002 REM ====================
1003 :
1010 MS=8192
1020 FOR I = 0 TO 7
1030 MI = MS + I * 128
1040 FOR J = 0 TO 7
1050 M2=MI+J*1024
1060 IF FL = 0 GOSUB 2000
1070 FOR L = 0 TO 39
1080 M=M2+L
1090 IF FL = 0 THEN A=B(L+1):POKE M,A:GOTO 1110
1100 A=PEEK(M):B(L+1)=A
1110 NEXT L
1120 IF FL=1 THEN GOSUB 4000
1130 NEXT J
1135 NEXT I
1140 IF MS = 8232 THEN RETURN
1150 MS=8232:GOTO 1020
1160 :
2000 REM ====================
2001 REM * OKUMA
2002 REM ====================
2003 :
2010 IF MS<>8192 THEN K=64:GOTO 2030
2020 K = 0
2030 :
2050 PRINT D$ " CIZIMI OKU"
2060 FOR P=1 TO 40
2070 INPUT B(P)
2080 NEXT P
2150 PRINT D$";:PRINT";:RETURN
3000 REM ====================
3001 REM * CIZIM *
3002 REM ====================
3003 :
3005 HPLOT 1,1, TO 278,1 TO 278,159 TO 1,159 TO 1,1
3010 HCOLOR=3
3015 HPOT X,Y
3020 HOME:VTAB 22:PRINT "I=YUKSEK/M=ALCAK/K=SAG/J=SOL"
3025 PRINT "S=SAKLA VE KORU "
3030 VTAB 1:GET Q$:PRINT
3040 IF Q$="I" THEN Y=Y-S:GOTO 3200
3050 IF Q$="M" THEN Y=Y+S:GOTO 3200
3060 IF Q$="K" THEN X=X+S:GOTO 3200
3070 IF Q$="J" THEN X=X-S:GOTO 3200
3080 IF Q$="C" THEN 3010
3090 IF Q$="S" THEN PRINT:RETURN
3100 GOTO 3030
3200 HPLOT TO X,Y:GOTO 3030
4000 REM ====================
4001 REM *YAZIM *
4002 REM ====================
4003 :
4010 IF MS<>8192 THEN K=64:GOTO 4030
4020 K=0
4030 :
4050 PRINT D$"CIZIMI YAZ"
4060 FOR P=1 TO 40
4070 PRINT B(P)
4080 NEXT P
4150 PRINT D$:PRINT:RETURN
```

rasıyla 128 ve 1024'lük bir artma adımı ile, 8 satırlık 8 bölgeye ayrılır (sayacı 0'dan 7'ye artan iki döngünün bulunması da bundan ötürü gereklidir). Birbirini izleyen bölgelerin işaretlenmesi döngü sayacının göstergesinin her geçişte artırılmasıyla elde edilir. Disket üzerinde belleğe alınmış verilerden yola çıkılarak resmin yeniden oluşturulmasında izlenen mantık, fiş yerleştirmede kullanılana benzer. Dolayısıyla aynı alt-programdan iki farklı amaç için yararlanılabilir: alt-programın hangi amaç için kullanılacağı, çağırma sırasında hareketе geçirilen çentiğin değeriyle belirlenir.

Bu alt-programın başka makinelerde kullanılması kimi kez önemli değişiklikleri gerektirir; örneğin bir noktanın (ya da rengin) durumunu okuyabilen bir komut varsa, dış döngüyü eleyerek, biri satırlar öbürü sütunlar için iki döngüyle yetinmek mümkün olur.

Alan ve Çevrelerin Hesaplanması

Bir noktanın durumunun, daha doğrusu bunu kontrol eden bellek adresinin içeriğinin çeşitli uygulamalar, örneğin video oyunları programları için zorunlu olmaktadır. Bu programlarda, ekranda gösterilen başka figürlerle olası çarpışmaları denetlemek üzere sık sık figürün yeri değiştirilir.

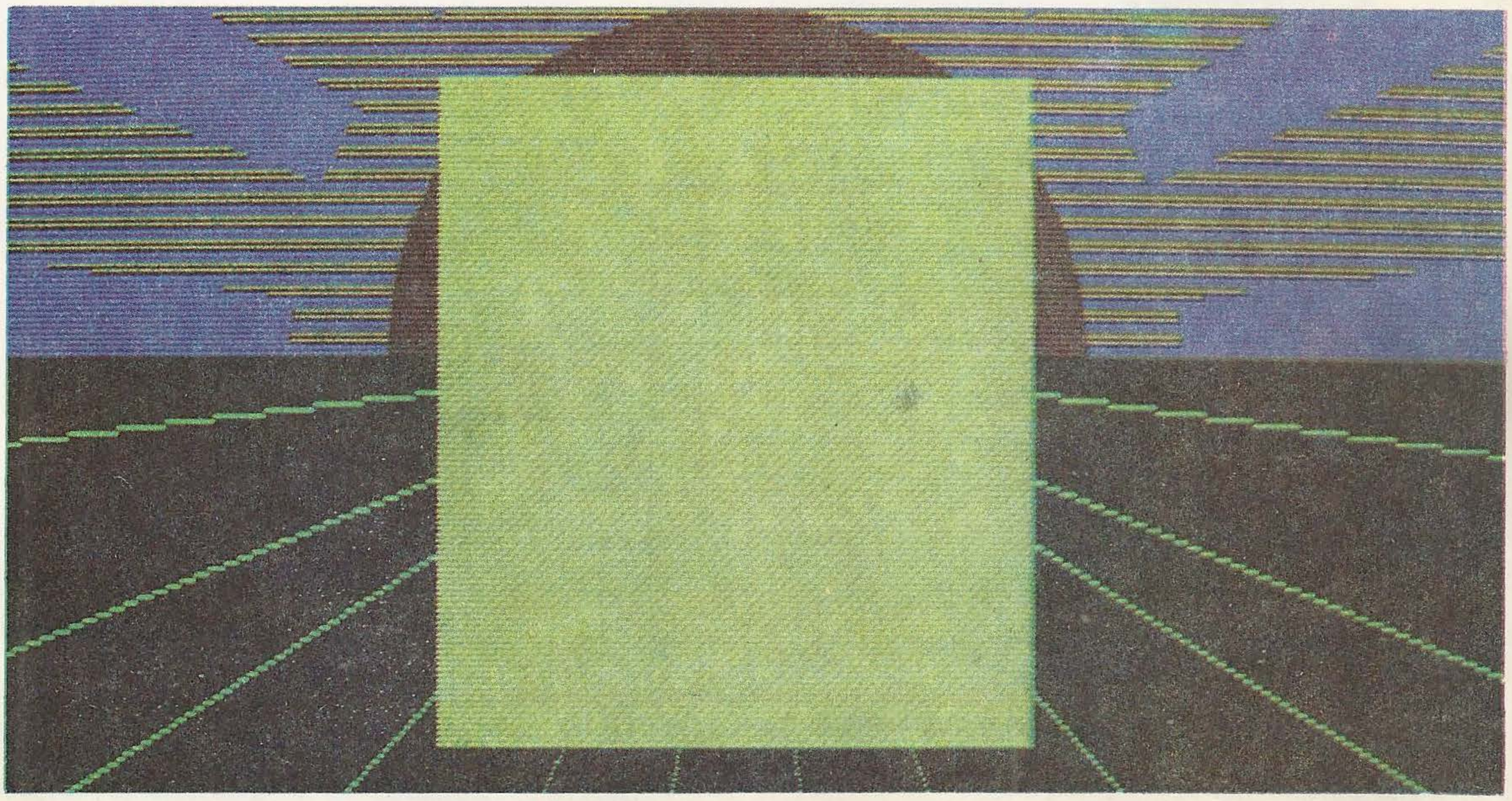
Bazı makinelerde bu kontrol, çok daha yaygın kullanılan kişisel bilgisayarlarda az raslanan bir BASIC komutuyla sağlanır. Bu durumda yer değiştirmeden sonra kullanılacak alanın boş mu yoksa başka bir imge içeriyor mu olduğunu belirlemek için ekran belleğinin doğrudan okunması yöntemine başvurulur. Bir başka deyişle, eğer varış alanındaki değerlerin hepsi 0 değilse, haraket halindeki figürle daha önce orada bulunan arasında çarpışma olacaktır. Görünüşte bu yöntemin hiçbir güçlüğü yoktur.

Betimlenen işlevleri gerçekleştirebilen bir alt-program gerçekte daha karmaşık olabilir. Bu konu, daha sonra bu işlevin özel bir yazılımla gerçekleştirilmesi ile özel tertibatlı makinelerde gerçekleştirilmesinde elde edilen sonuçlar arasındaki farklar anlatılırken ele alınacaktır.

Çarpışma sorununa benzeyen, ancak daha basit diğer bir sorun da alan ve çevrelerin hesaplanmasıdır. Birçok grafik uygulamada, resim yapıldıktan sonra, görüntülenen nesnenin boyutlarını ölçmek için bir işlem evresine geçmek gerekir. Çizim sona erdiğinde artık tüm boyutlar grafik bellekte bulunmaktadır, dolayısıyla, geometrik özelliklerin hesaplanması için birkaç alt-program tasarlamak güç değildir.

Sorun, resmin kavranma biçimine göre iki türlü ele alınabilir. İlk yöntemde kullanıcı tarafından bir grafik menü ile yaratılan resmin daha önceden bellekte bulunuyor olması gerekir. Bu durumda resmin kenarlarını ve yüzeyini nokta nokta ele alarak, tüm bellek ekranını çözümlemek gerekir.

Bir ekran penceresinin bellekteki bir imgeyle üstüste getirilmesi



İkinci yöntemde ise resmin hesaplanması, çizimiyle aynı anda gerçekleşir: örneğin grafik levha aracılığıyla yapılan girişlerde bu durum geçerlidir.

Daha önce görüldüğü gibi, bu çevresel gereç, bilgisayara sürekli olarak bağlantı elemanının koordinatlarını veren bir konum transdüktörüdür. Resim, bu referansın figürün dış hatları boyunca ilerletilmesiyle elde edilir.

Bu modda, çevre doğrudan doğruya hesaplanır. Her yer değiştirmenin bir doğru parçasıyla şematize edilmesiyle, çeşitli noktalar arasındaki uzaklıklar ölçülür. Dolayısıyla, yer değiştirmeler ne denli küçükse hesap o ölçüde kesindir.

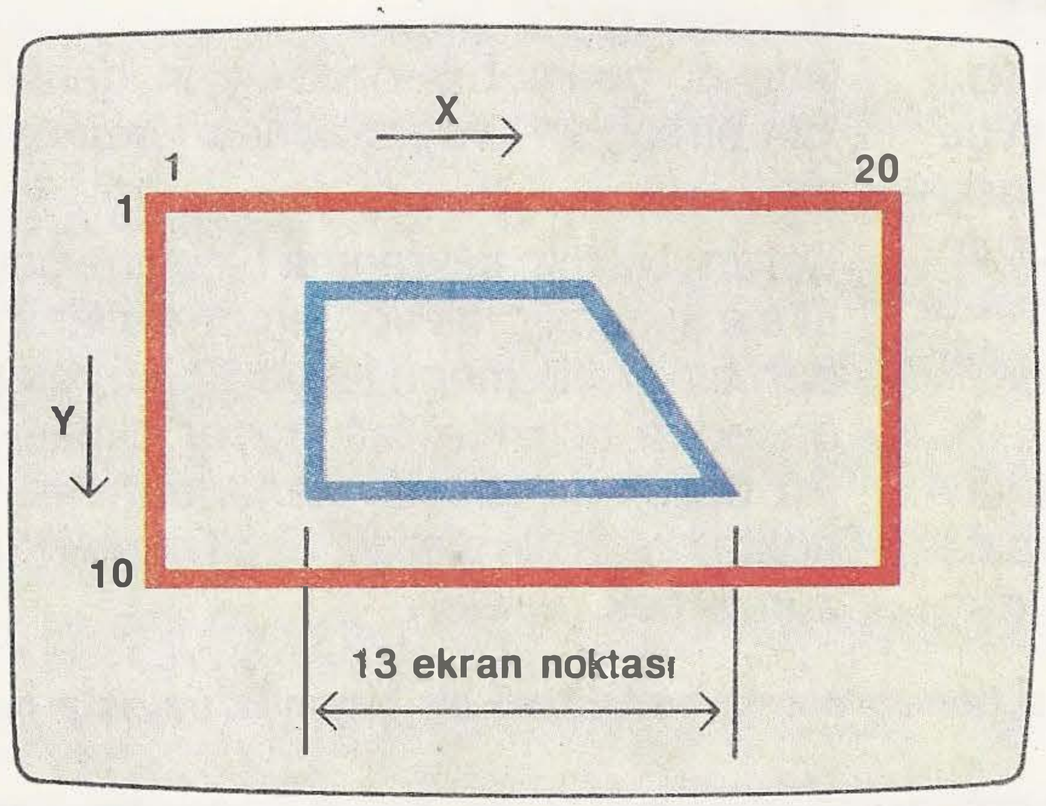
Gerçekte kullanıcı, dairenin çevresinin bu şekilde doğru parçalarına bölünmesi işleminin farkına varmaz; bu süreç ona kesintisiz görünür. Programın akışını çok yavaşlatmadan, kenarı yeterince küçük elemanlara bölerek tam bir kesinlik sağlamak yazılımın görevidir.

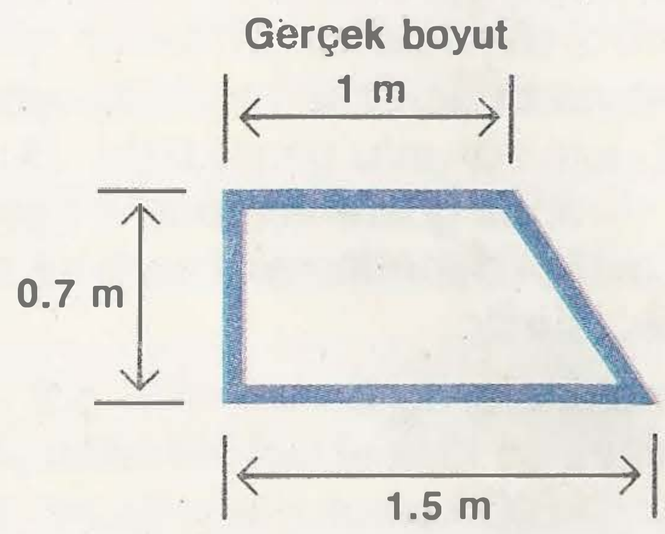
Resim sona erdiğinde yüzey hesaplanır. Bu durumda en kolay yöntem, ekran belleğinin incelenerek hesabın bellekte saklanan imge üzerine yapılmasıdır (bkz. bu sayfadaki şema).

Yöntemi örneklendirmek için, matematiksel olarak hesaplanmış yüzeyi 0.875 m tutan bir yamuk seçilmiştir (boyutlar metre olarak verilmiştir). Büyük tabanı (13 ekran noktası) ile gösterilen çizilmiş bir imgeyi ele alalım. Ölçek faktörü ise 1.5/13 olsun; bu durumda her ekran noktası 0.11...m. olacaktır. Yüksekliği ise 6.06...m.'dir (ekran noktası olarak yükseklik = 13/1.5 × 0.7).

Grafik nedenlerle seçilen bu boyutlandırma

EKRANDA ELDE EDİLEN İMGE

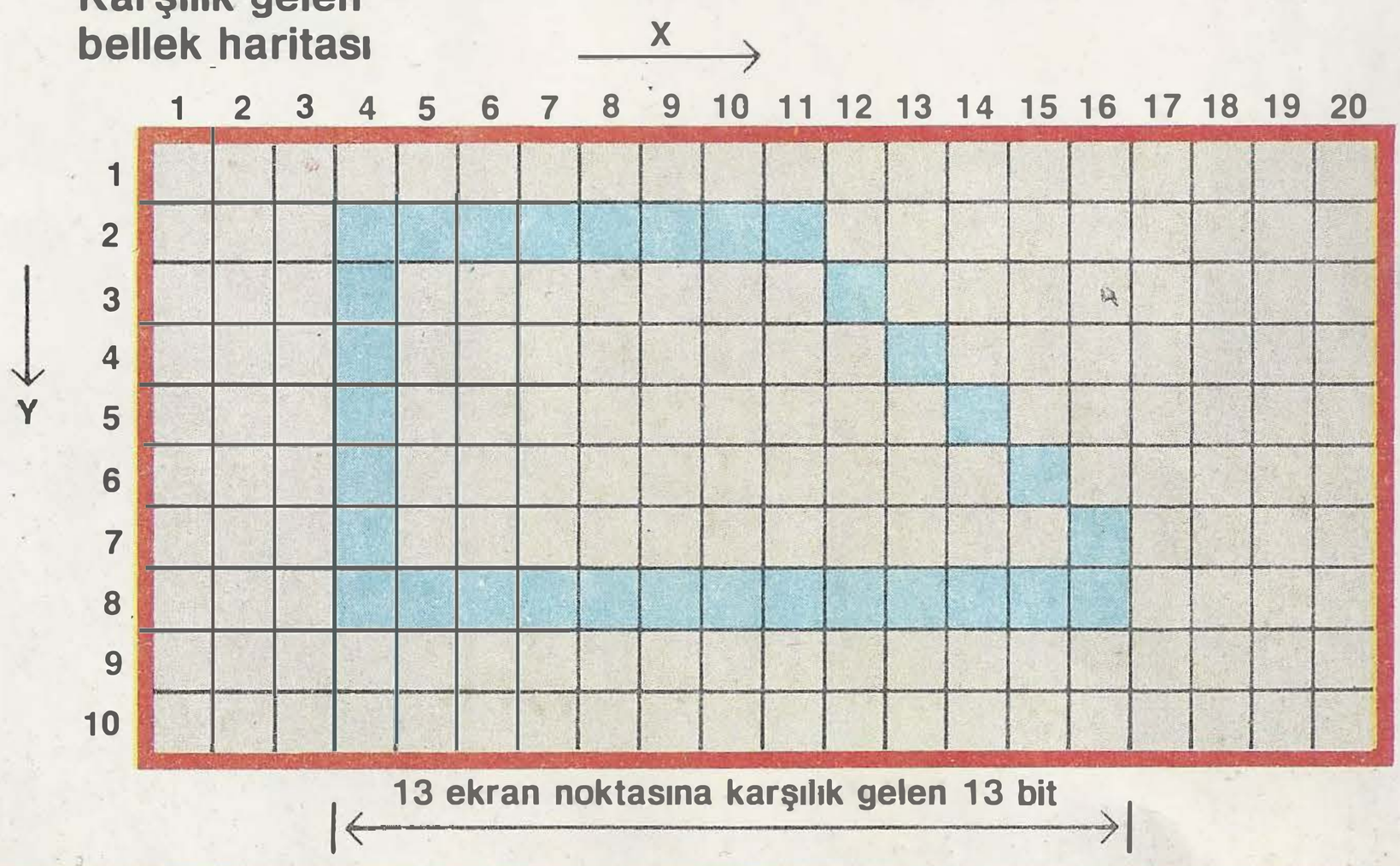


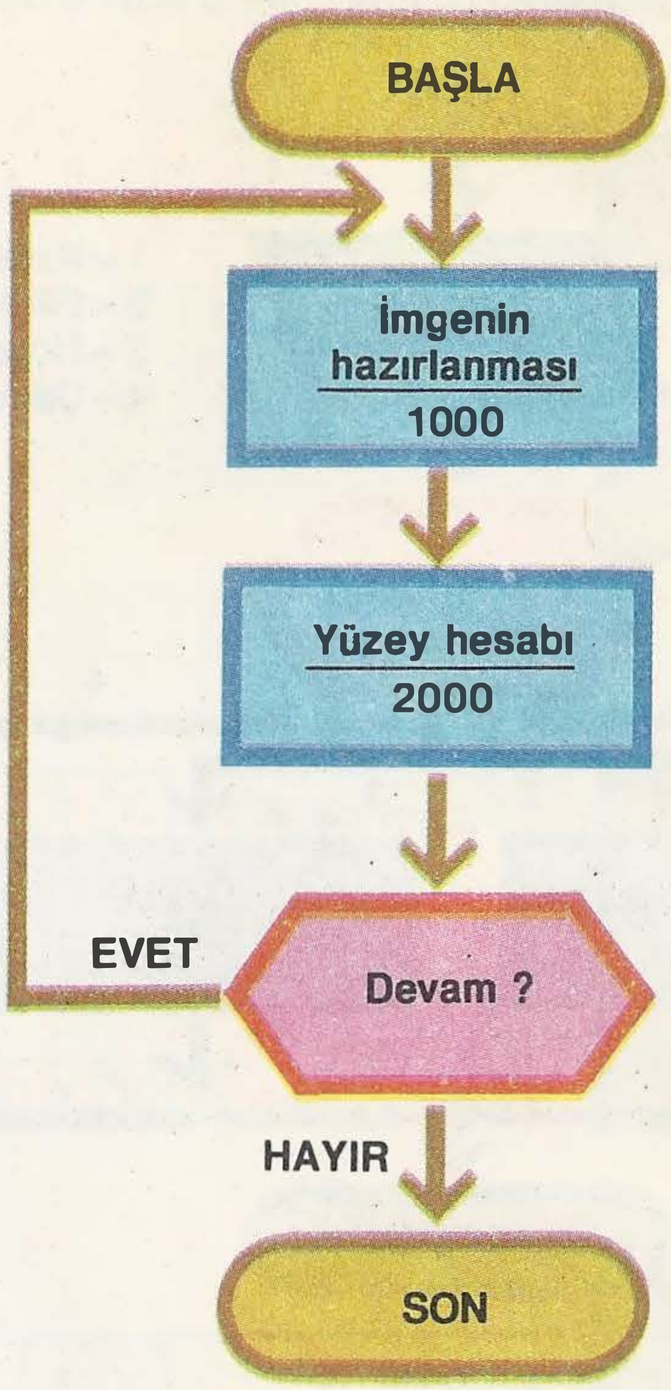


$$S = \frac{2.5}{2}\ 0.7 = 0.875\ m^2$$

Ölçek faktörü = 1.5/13

Karşılık gelen bellek haritası





yetersizdir. Gerçekte resmi göstermek için en uygun ölçek 100'dür; o zaman taban 1.5 × 100 = 150 ekran noktası, yükseklik de 70 olur ve daha yüksek bir hassasiyet elde edilir.

Şemanın alt bölümünde imgeye karşılık gelen bir bellek kartı gösterilmektedir. Numaralar ekran numaraları değil, ekranda figürün çevresine rasgele verilmiş numaralardır; ancak yöntemin gösterilmesi bakımından anlamlıdır. Yüzey hesabı sözkonusu belleğin alanının incelenmesi ve her noktanın figürün içinde yer alıp almadığının kontrol edilmesidir. İç noktaların toplamı, ölçeğe göre yüzeyi verir.

Belleğin incelenmesi, iki döngüyle gerçekleştirilir. Bunlardan en dıştaki Y eksenine (örnekte 1'den 10'a), en içteki de X eksenine (1'den 20'ye) göredir.

Y = 1'de bu satıra ait bütün bellek yerleri incelenir ve hiçbir aktif bit bulunmaz, demek Y = 1 satırında çizilmiş nokta yoktur. Y = 2 olduğunda sütun 4'te ilk bitin aktif olduğu görülür, çünkü figürün içindeki bir satırın başlangıcı olarak (X1 değişkeninde) belleğe ilk alınan budur. Satırın incelenmesi sürdürülerek, (1'den sonra) her aktif bite raslandığında bunun konumu XF değişkenine yerleştirilir. İç döngünün sonunda (X1'den 20'ye) XF = 11 olur ve XF-X1 farkı da ilk iç satırın uzunluğunu verir (özellikle Y = 2'de: bu satır incelenen yamuğun kısa tabanıdır).

Resmin ölçeği 1.5/13 olduğunda, doğru parçasının uzunluğu, gerçek değerlerle (1.5/13) × (XF—X1) = (1.5/13) × 8 = 0.92.... m'dir.

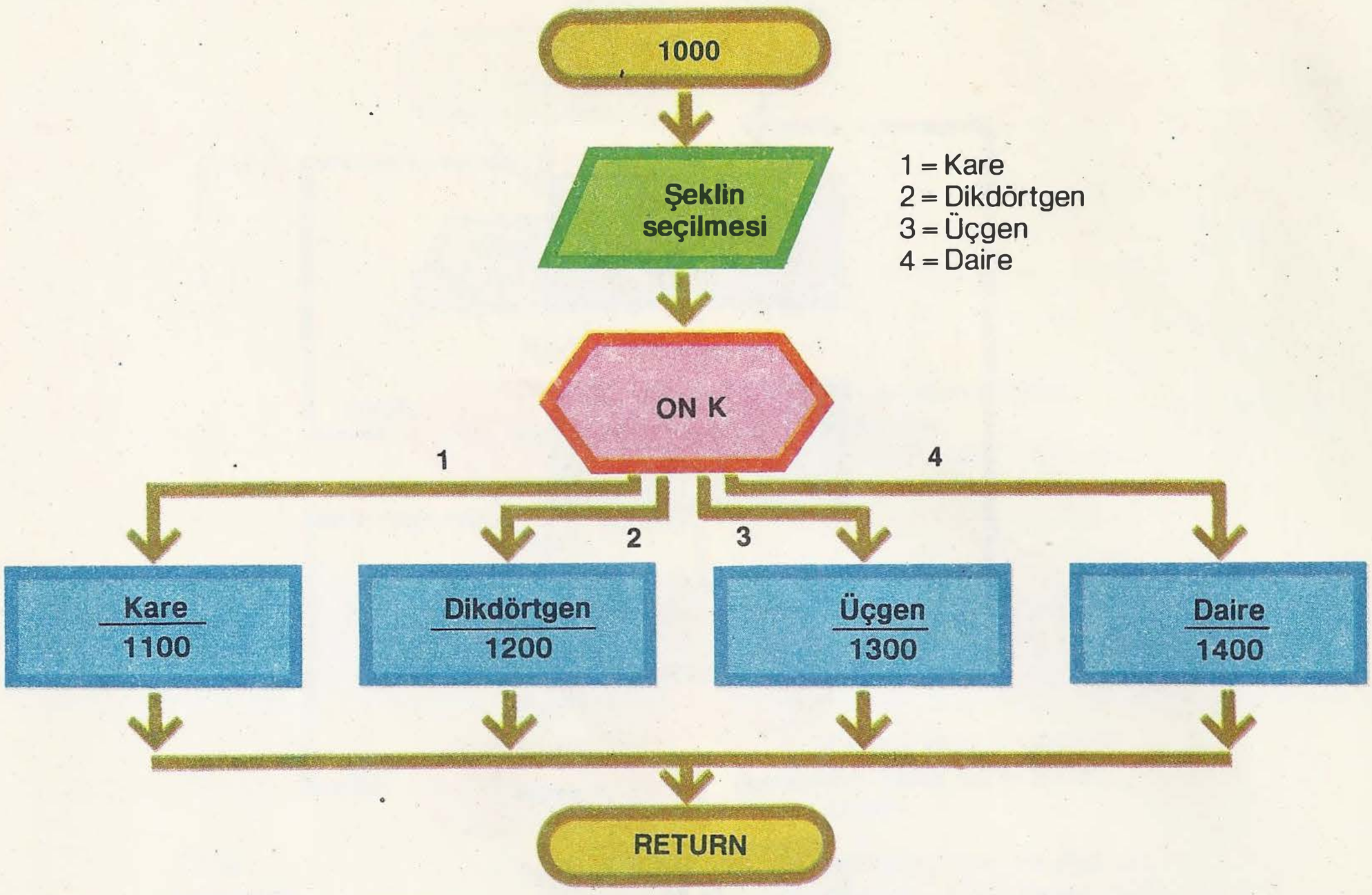
Kısa tabanın gerçek ölçüsü ile uygulanan ölçek faktöründen kaynaklanan farka (1 m) dikkat etmek gerekir. 1.5/13 değeri ile, kısa tabanın 8.66 ekran noktası uzunluğunda olması gerekirdi, ancak hata payı ile birlikte 8 olarak alınması gerekir(*).

Daha uygun hale getirilmiş bir ölçekle bu hata önemsizleştirilir. Böylece ölçülen doğru parçasının (uzunluk = 0.92) boyu, Y eksenine göre bir ekran noktasına eşit, yani 1.5/13 m.dir. Yüzeyin ilk elemanı bu niceliğin uzunlukla çarpılmasıyla elde edilir.

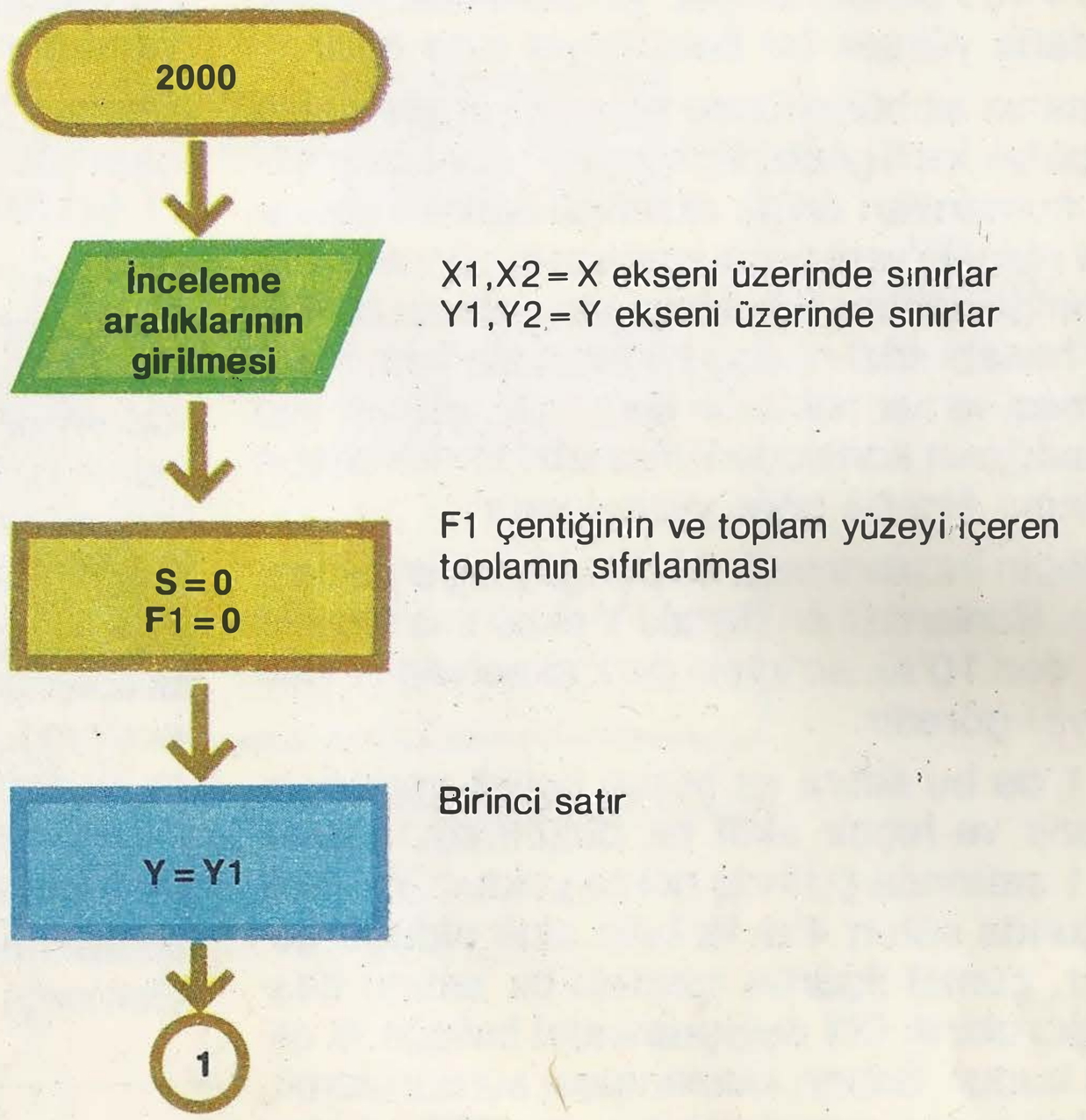
İşlem bütün satırlar için bu yolla bulunan her elemanın eklenmesiyle sürdürülür.

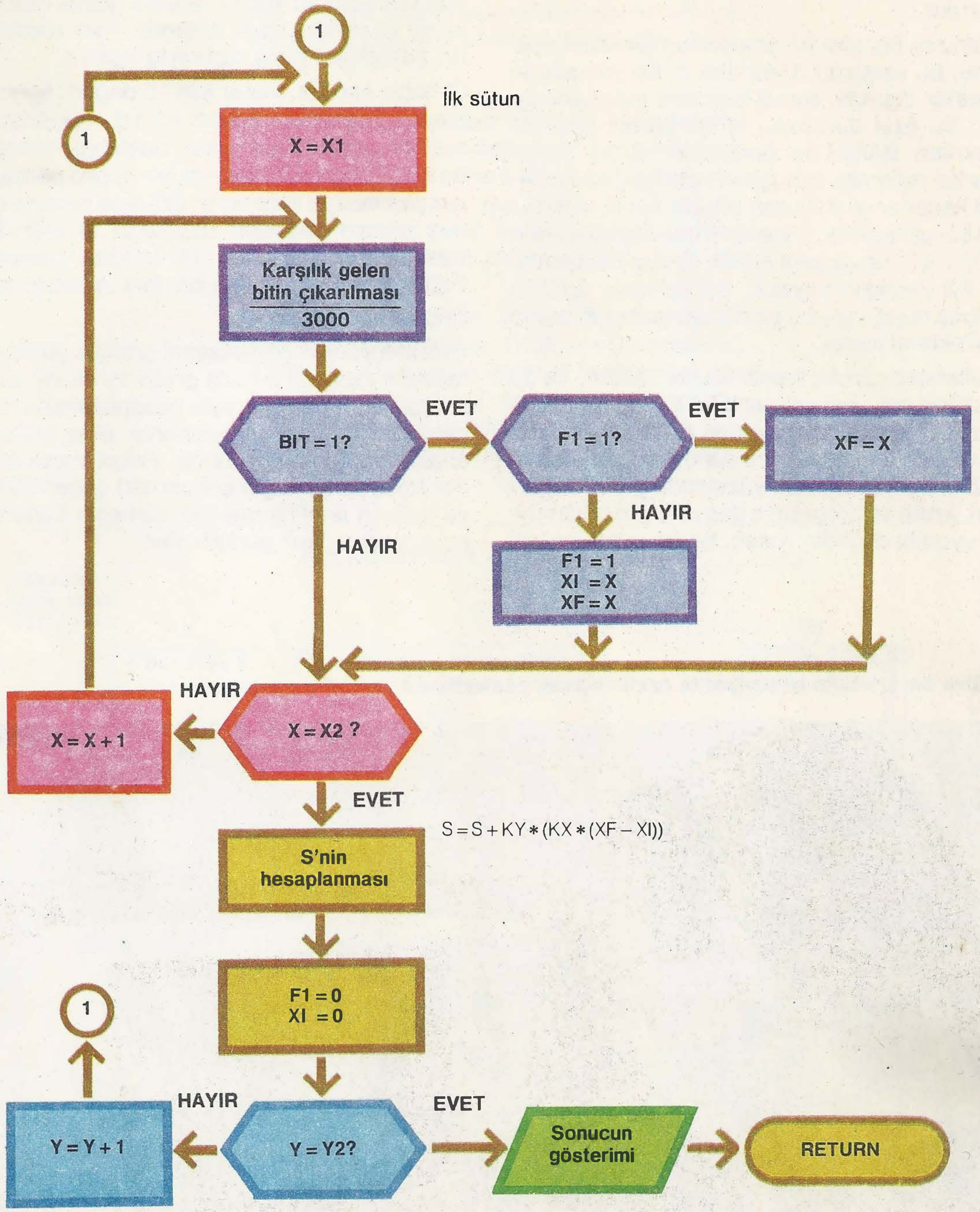
(*) Ekran belleği ve ekran, gerçek değerlerle adreslenemez.

DENEME İÇİN BİR İMGENİN HAZIRLANMASI ALT-PROGRAMI



YÜZEY HESABI ALT-PROGRAMI




1
1
X = X1
İlk sütun
Karşılık gelen
bitin çıkarılması
3000
BIT = 1?
EVET
F1 = 1?
EVET
XF = X
HAYIR
F1 = 1
XI = X
XF = X
HAYIR
HAYIR
X = X + 1
X = X2 ?
EVET
S'nin
hesaplanması
S = S + KY * (KX * (XF – XI))
F1 = 0
XI = 0
1
HAYIR
Y = Y + 1
Y = Y2?
EVET
Sonucun
gösterimi
RETURN

Sorunumuz 2000 alt-programı tarafından ele alınır, diğerleri ise bazı geometrik figürlerin ekranda hazırlanmasını üstlenmiştir. Programda, bellek yerlerinin hesaplanmasında, daha önceden sergilenmiş olan alt-programa başvurulur.

Yalnızca öğretici bir amacı olan öncekinin yerine, bu versiyon 3142'den 3150'ye kadarki satırlar dışında, temel biçimine indirgenmiştir. Bu özel durumda, işlem süresi özellikle çevrilen BASIC'de önemlidir. Uzun beklemeleri önlemek için işlem ilerledikçe imgelerin kenarlarını dolduran bu satırları (3142'den 3150'ye) soktuk. Yine aynı mantığı izleyerek, 2071 için de geçerli olmak üzere, kaldırılabilir F2 çentiğini koyduk. Koordinatlar figürün dışına taştığında bu çentik ekran noktalarının hareketini keser.

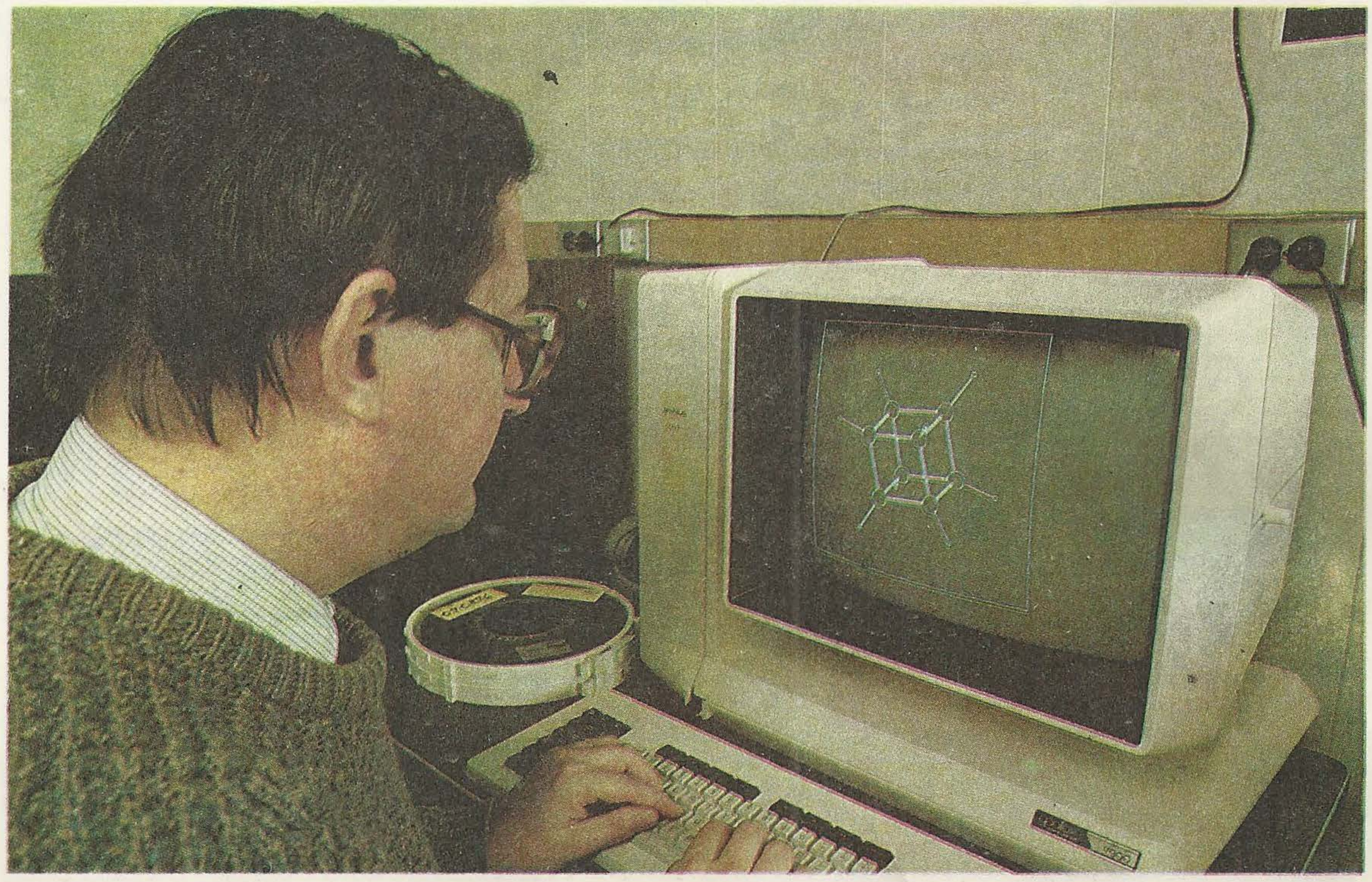
Kullanılan mantık kasıtlı olarak basittir, ve bu yüzden çok işlevsel değildir. Doldurma, figürün 1. satırı düşey çizgiden çok uzak olmadığı ölçüde tamdır. Ayrıca sunulan geometrik figürler ve hesaplanan yüzeylerin göreli değerleri, kesin ve programın geriye kalan bölümüyle uyumlu değildir. Amaç, başka noktalardaki karmaşıklıklardan kaçınarak yalnızca alt-program 2000'den nasıl yararlanılacağını göstermektedir.

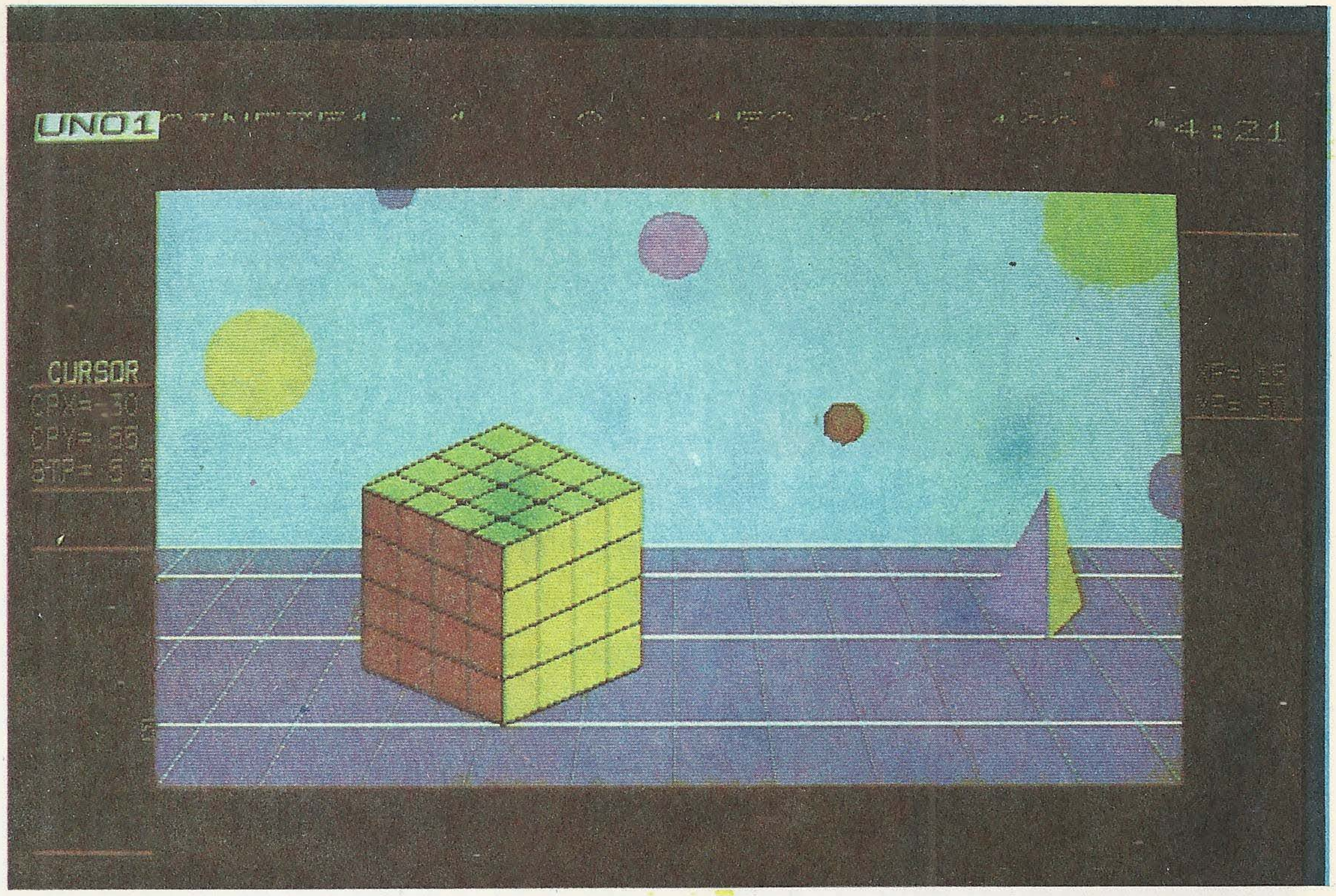
Özellikle, figürlerin çizimi ve program tarafından hesaplanan yüzey, analitik olarak hesaplanan yüzeylere denk düşmez. Fark resimlerin gerçekleştirilme biçimiyle ilgilidir.

Örneğin karede, kenar için 10 değeri, figürün başlangıç noktası olarak 10,10 koordinatları girilir ve ilk düşey kenar, başlangıç noktası ile 10,20 noktası arasında bir doğru parçasının çizilmesiyle elde edilir. Böylece kenarın ekran noktası cinsinden uzunluğu 11 olur: Bu hesaplanan değerden 10 farklılık gösterir. Figür doğru çizilmişse bu fark ortadan kalkar.

Sunulan yöntem, hesaplama grafiğin çizimi sırasında yapıldığı ölçüde grafik türde bir yöntemdir. Sonucun (yüzeyin hesaplanması) matematik formüller uygulanarak elde edildiği analitik yöntemler de vardır. Fakat ancak özel durumlarda (örneğin çokgenler) geçerlidirler ve figürün analitik tasvirini (örneğin kenarların koordinatları) gerektirirler.

Kübik bir kristalin bilgisayarda grafik olarak gösterilmesi





Renkli ekranda üç boyutlu grafik görüntü

YÜZEYLERİN HESAPLANMASI PROGRAMI

Siprel 2010 versiyonu, Apple ve denkleri

```
10 REM
20 REM  : KUTUK = YUZEYLER
30 REM
40 KY = 1:KX = 1
50 GOSUB 1000
60 GOSUB 2000
70 REM
80 INPUT"DEVAM ":A$
90 IF A$ = "E" GOTO 50
92 END
1000 REM : DENEME FIGURU
1005 REM
1010 HGR
1015 PRINT "SECME (1.2.3.4
       ? "
1020 GET R
1025 IF R< 1 OR R > 4 GOTO 1015

1030 ON R GOSUB 1100, 1200, 1300,1
      400
1040 RETURN
1050 REM
1052 REM *********
1054 REM
1100 REM : KARE
1105 REM
1110 INPUT "KENAR ":L
1115 INPUT "KONUM (X,Y) ":X
       0,YO
1120 HPLOT XO,YO TO XO + L,YO TO
     XO + L,YO - L TO XO,YO - L TO
      XO,YO
1121 A = L * L
1122 PRINT "YUZEYLER = ":A
1125 RETURN
1130 REM
1132 REM *********
1134 REM
1200 REM DIKDORTGEN
1210 INPUT "    "LX,LY
1215 INPUT " KONUM (X,Y) ";XO
       ,YO
1220 HPLOT XO,YO TO XO + LX,YO TO
     XO + LX,YO - LY TO XO,YO - L
      Y TO XO,YO
1221 A = LX * LY
1222 PRINT "YUZEY = ":A
1225 RETURN
1230 REM
1232 REM ****
1234 REM
1300 REM : UCGEN
1305 REM
1310 INPUT "TABAN VE YUKSEKLIK ":B,
       H
1315 INPUT "KONUM (X,Y) ":X
       0,YO
1320 HPLOT XO + B,YO TO XO,YO TO
       XO,YO + H TO XO + B,YO
1321 A = B * H /2
1322 PRINT "YUZEY = ":A
1325 RETURN
1330 REM
1332 REM **********
```

```
1334 REM
1400 REM : DAIRE
1405 REM
1410 INPUT "YARICAP ";R
1415 INPUT "MERKEZ (X,Y) ";XO,Y
     O
1520 X1 = XO + R:Y1 = YO
1525 FOR A = -0.05 TO 6.3 STEP
     0.05
1530 X2 = XO + R * COS (A)
1535 Y2 = YO + R * SIN (A)
1540 HPLOT X1,Y1 TO X2,Y2
1545 X1 = X2:Y1 = Y2
1550 NEXT A
1555 A = 3.14 * R ** 2
1556 PRINT "YUZEY= ";A
1560 RETURN
2000 REM
2005 REM : YUZEYLER
2010 REM
2015 INPUT "X EKSENI SINIRLARI ";X1,
     X2
2020 INPUT "Y EKSENI SINIRLARI ";Y1,
     Y2
2025 S = 0: F1 = 0:XI = 0:XF = 0
2030 FOR Y = Y1 TO Y2
2035 FOR X = XI TO X2
2040 GOSUB 3000
2045 IF BIT = 0 GOTO 2060
2050 IF F1 = 1 THEN XF = X:GOTO
     2060
2055 XI = X:XF = X:F1 = 1
2060 NEXT X
2062 PRINT "Y= ":Y,"XI= ";XI,"XF
     = ":XF
2065 S = S + KY * (KX * (XF - XI)
     )
2070 F1 = 0:XI = 0: XF = 0
2071 F2 = 0
2075 NEXT Y
2080 PRINT "YUZEY = ":S
2082 GET D$
2085 RETURN
2090 REM
2092 REM *****
2094 REM
3000 REM : BITIN CIKARILMASI
3005 REM
3010 REM   GIRISLER
3015 REM                X,Y
3016 :
3020 REM
3025 REM        CIKISLAR :
3030 REM                      BIT=0.1
3035 REM
3036 :
3040 GY = INT (Y / 8)
3045 IF Y < 64 THEN MR = 8192 +
     GY * 128: GOTO 3070
3050 IF Y < 128 THEN MR = 8132 +
     (GY - 8) * 128: GOTO 3070
3060 MR = 8272 + (GY -16) * 128
3070 A = Y - GY * 8
3080 MY = MR + A * 1024
3090 GX = INT (X / 7)
3100 MX = MY + GX:BIT = X - 7 * G
     X
3105 A% = PEEK (MX):B% = 2  BIT
3110 NN = A%
3115 FOR K = 0 TO BIT
3120 M = NN / 2
3125 NN = INT (M)
3130 NEXT K
3135 BIT = 0
3136 IF M < > NN THEN BIT = 1
3142 IF F1 = 0 THEN RETURN
3143 C% = A% + B%
3145 IF F1 = 1 AND BIT = 1 THEN
     F2 = 1
3146 IF F2 = 1 THEN RETURN
3150 POKE MX,C%
3152 RETURN
3160 REM
3162 REM ************
3164 REM
```

Analitik yöntemler, bazı bilimsel bilgileri gerektirmeleri nedeniyle grafik programlamadan çok bilimsel programlamayı ilgilendirir.

Ama şimdi sunulacak yöntem hiçbir formülden yararlanmaz ve her düzlemsel figüre uygulanabilir. Bu yöntemde grafik sayfanın kontrol döngülerinin işlenmesi hatırı sayılır bir zaman gerektirir. Bunun yol açtığı bekleme süresini kısaltmak için;

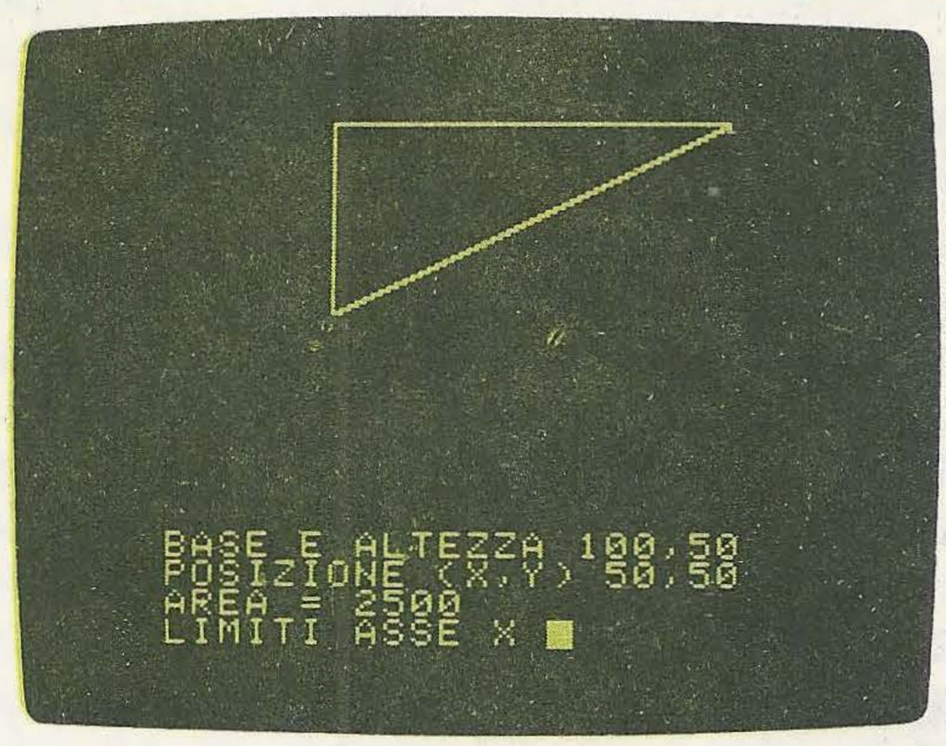
1) programın derleme versiyonundan yararlanılabilinir
2) ve/veya figürün içinde incelenecek bellek bölgesi sınırlandırılır.

Bu nedenle 2000 alt-programı ekranın iki eksen boyunca incelenmesine konacak sınırları arar, böylece figürün dışında kalan alanlar çözümlemenin de dışında kalır. İncelenecek alanın bu şekilde sınırlandırılmasının bir diğer avantajı da, ekranda aynı anda bulunan figürlerin çözümlenmesine olanak vermesidir. Bunun için kaç figür varsa, herbirine özgü inceleme sınırlarını koyarak, alt-programı aynı sayı kadar çağırmak yeterlidir. Dikkat: Verilerin girişine özel bir dikkat göstermek gerekir çünkü yanlış bir hesap büyük hatalara yol açar. Burada da bir dizi denetime gerek duyulur. En basit işlem, ekranda koordinatları belirlemek üzere bir kafesin çizilmesidir. Bunun için ekranın kenarlarını izleyerek ve örneğin her on noktada bir referans köşelerini işaretleyerek bir dikdörtgen çizmek yeterlidir. Bu çizimi gerçekleştiren programla, kartezyen eksenlerin ya da histogramların çerçevesini çizen program aynıdır.

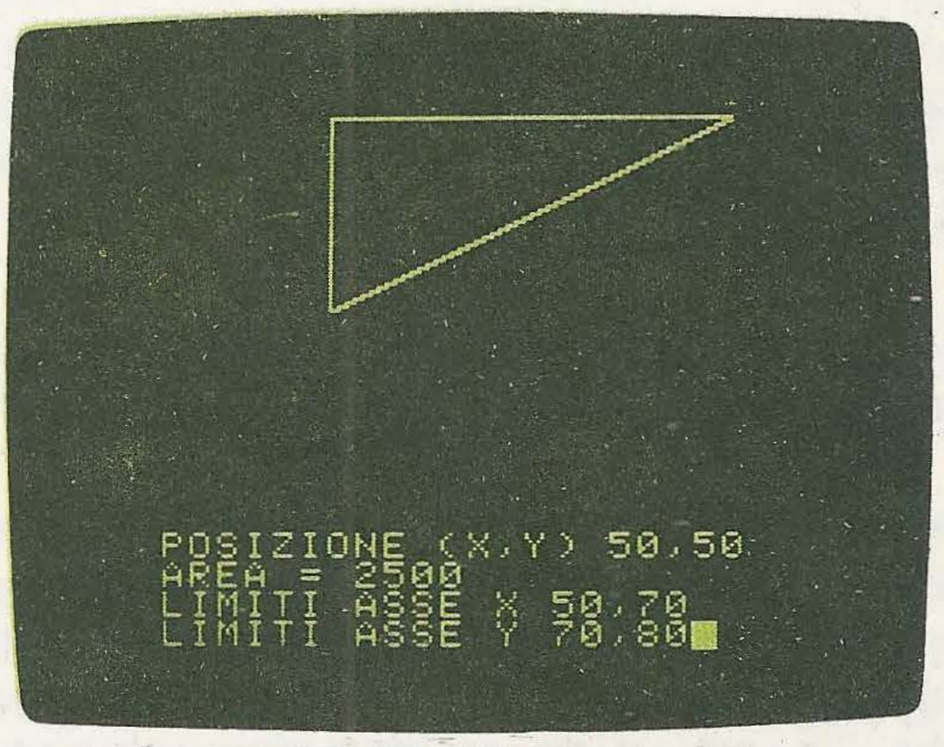
Bir başka (analitik) teknik de, resim ilerledikçe asgari ve azami koordinatların belleğe alınmasından oluşmaktadır. Bu şekilde sonunda elde edilen asgari ve azami değerler dikdört-

YÜZEYLERİN HESAPLANMASI PROGRAMI

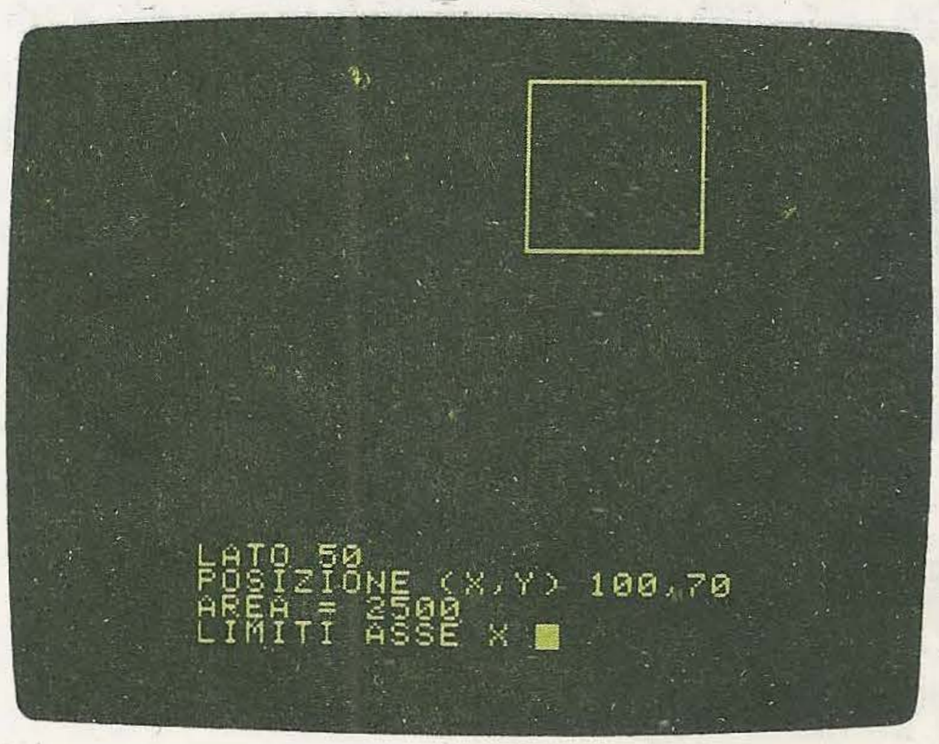
1561 ve 1562. sayfalarda verilmiş olan programın bazı uygulama örneklerini inceleyelim. Ekranda, 1015 satırına yanıt olarak 3 değerinin girilmesinden sonra beliren görüntü görülmektedir. Denetim, üçgenin tabanını, yüksekliğini ve konumunu soran ve yüzeyin analitik değerini (1321, 1322) veren 1300 alt-programına geçer.



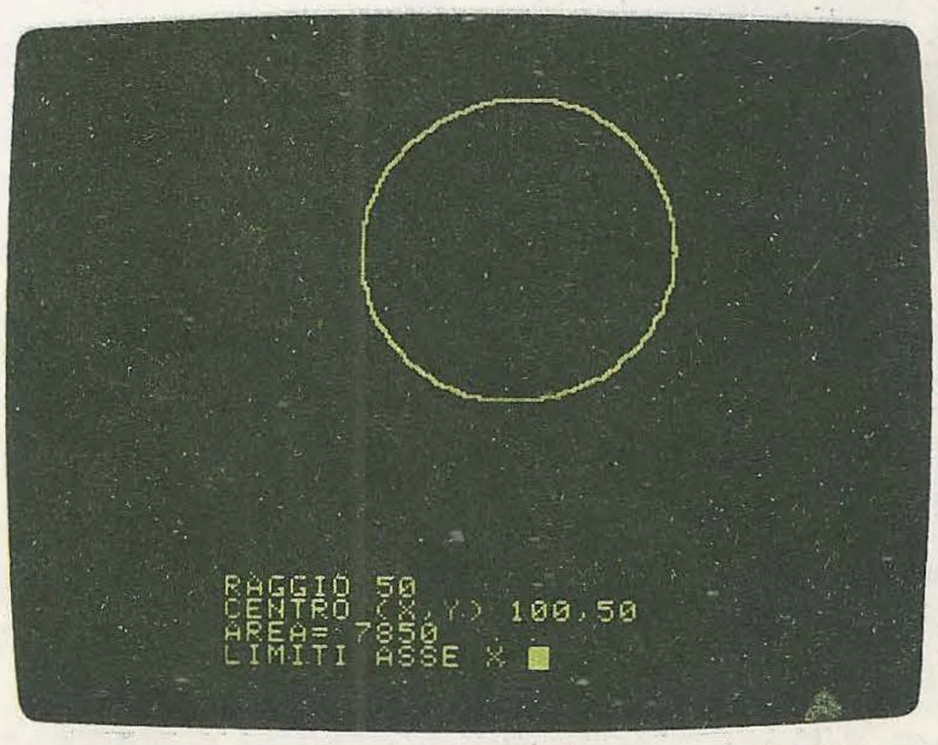
Daha sonra ekran belleği taranarak yüzeyi hesaplayan 2000 alt-programına geçilir. Bu alt-program X ve Y eksenleri boyunca ekranın taranmasını ister. Bu ve önceki ekranlar bu sınırların girilmesi evresini gösterir: X ekseni 50'den 70., Y ekseni ise 70'den 80. konuma kadar incelenir. Sonunda, taramayla belirlenen yüzey değeri görülür.



3. ekran, 100,70'te konumlandırılmış, bir kenarı 50 olan karenin aynı durumunu gösterir. Analitik olarak hesaplanan yüzey 2500 değerindedir; programın belirlediği değer, ekranın analiz sınırları girildiğinde görülecektir.



Son ekran (daire). Sistem, grafik modda, kullanıcıyla diyaloga girebilmek için 4 alt satırı ayrı (metin modunda) tutabilir.



gen yüzeyin sınırlarını verecektir. Tabii birden çok figür olduğunda, kullanıcının, herbir figür sona erdiğinde bunu programa bildirmesi gerekir.

İncelenecek alanın sınırlandırılması bir başka hata türünün de önüne geçer. Kenarları kötü çizilmiş bir resimde bu hata türü özellikle sözkonusu olur. Resim kapalı olmadığından, incelenmesi ve sonraki hesaplama, program ekranın kıyısına varana kadar devam eder. Sonuç olarak bu bölüm için gerçektekinden daha büyük bir alan elde edilir.

1400 alt-programında, çember çizmekte yararlanılan döngü bu nedenle hazırlanmıştır. Dairenin o açıdan başlayıp $2*\pi=6.28$ açısına kadar (bu da tam bir daireye eşittir) çizilmesi gerekirken, figürün bütünüyle kapanmasını sağlamak için -0.05 ve 6.3 sınırlarından yararlanılır.

Figürlerin bellekte saklanması için BASIC komutları:

Bir figürü belleğe almak için uygulanan yöntem hiç bir özel komut gerektirmez. Bu komutlar bellek adresleri bilinen her makineye uygulanabilir. Bazı durumlarda, çok basit ve belleğe alınacak ya da yeniden çizilecek alanın grafik koordinatları dışında hiç bir adresleme gerektirmeyen yüksek düzeyli komutlar bulunur. Bu komut türü, makinenin yapısıyla yakından ilintilidir. Bu nedenle, sözdizimleri hatta kimi kez gerçekleştirilen fonksiyonlar makineye özgüdür... Örneğin Olivetti M20'de bulunan komutlar şunlardır:

POINT: Noktanın rengini kaydeder.
GET : Ekran içeriğini belleğe alır.
PUT : GET komutuyla kaydedilmiş imajı görselleştirir.

Burada kullanılan GET ve PUT komutları özdizimleri ve amaçları bakımından G/Ç fonksiyonlarıyla kullanılanlardan farklıdır.

POINT. Sözdizimi A% = POINT (X,Y)'dir.

Program tamsayı A% değişkenine işlem sırasında X ve Y koordinatlarında bulunan noktanın rengine göre bir sayısal değer verir. Tek renkli bir ekranda bu değer 0 ya da 1'dir (0 = kapalı, 1 = açık) dört renkli ekranda bu değer 0'la 3 arasında sekiz renkli ekranda ise 0'la 7 arasında yer alır.

X ve Y koordinatlarının değerleri kullanıcı tarafından belirlenmeyip, gerece yani kulanılmakta olan ekranın noktalarına bağlıdır. Bazı makinelerdeyse ekran noktalarını belirleyeninkinden farklı bir referans sistemine başvurulabilir: Bu yeni sistem **kullanıcı referans sistemi** diye adlandırılır.

POINT komutu, biri metin diğeri grafikle ilgili iki imlecin elde edilmesi için bir başka grafik komutla birlikte, ancak farklı bir biçimde kullanılabilir. Komutlar biçimsel bakımdan farklı olmakla birlikte, metinle ilgili olanının işletimi diğer makinelerdekinin aynıdır. Oysa grafik imlecin POINT komutuyla işletimine diğer makinelerde rastlanmaz.

Bu komutun en basit sözdizimi:

CURSOR POINT (X,Y)N,M'dir.

N ve M görünen (N = 1) ya da görünmeyen (N = 0) bir noktayı veya imlecin yanıp sönme düzenini belirtir (M = 0'da yanıp sönmez).

GET. Basitleştirilmiş sözdizimi;

GET (X_1,Y_1)-(X_2,Y_2),A%(0)'dır.

X_1,Y_1 ve X_2,Y_2 belleğe alınacak dikdörtgen ekran parçasının köşelerini gösterir, A%(0) ise değerlerin yerleştirileceği tablonun ilk elemanıdır (BASIC'te bir tablonun ilk göstergesinin 0 olduğunu unutmayalım).

Komut uygulamaya konulduğunda, koordinatlarla belirlenen ekran kesitine karşılık gelen tüm bitler A%(.)'da toplanır. A%(.)'da bitlerin herbirinin durumlarına ek olarak şu bilgiler bulunur:

A%(0): X_1, Y_1 ve X_2, Y_2 ile belirlenen dikdörtgenin tabanı
A%(1): Dikdörtgenin yüksekliği
A%(2): Tek ya da çok renkli ekran.

A%(3)'den itibaren, adresler betimlenen grafik alanın bitlerini stok etmek için kullanılır.

A%(.) matrisinin boyutlandırılması belleğe alınacak alanın yayılımına bağlıdır ve şöyle hesaplanır:

Boyut = [(taban/16 *'yükseklik)*K] + 3

K tek renkli ekranda 1, dört renklide 2, sekiz renklide 3 değerindedir. Taban/16 kesirinin ise çoğunlukla (yukarı doğru) yuvarlatılması gerekir. M20 bilgisayarı, ekranı birbirinden bağımsız pencerelere böler. Pencerelerden her-

biri, numarasının belirtilmesiyle bağımsız olarak kullanılabilir. GET komutunun tam sözdizimi:

GET W% (X_1,Y_1)-(X_2,Y_2),A%(0)'dır.

W%, pencerenin sayısal değeridir (bu konu ilerde ele alınacak).

PUT. Tam sözdizimi

PUT W%(X_1,Y_1)-(X_2,Y_2),A%(0),V'dir.

Burada da W% (pencere) parametresi seçilmelidir. PUT, önceki komuttakilere ek olarak V parametresini de içerir. V parametresi A%(.) matrisinde bulunan renk numaraları ile ekranda tanımlanmış dikdörtgende bulunan renklerin numaralarına uygulanmış mantıksal bir işlemin sonucunu gösterir.

V hesaba katılmadığında, A%(.) matrisi belleğe alındığı biçimiyle görselleşir. V belirlenmişse, görselleşme, önceden belirlenmiş renklerle A%(.)'ninkiler arasında V ile gösterilen mantıksal işlemin sonucunda gerçekleşir. V seçeneği AND, OR, XOR, NOT, PSET ve PRESET değerlerini alır. AND, OR ve XOR alışılmış anlamlarında kullanılır. NOT ekranda bulunan renklerin numaralarının tümleyicisini verir.

Ekran Pencereleri

Uygulamaların çoğunda bir ekranın birbirinden ayrı olarak yönetilen alanlara bölünebilmesine gerek vardır. **Pencere** adı verilen bu alanlar, uygulama programı açısından, ayrı küçük ekranlar gibidirler. çoğu durumda bir resmin bütünüyle, aynı resmin ayrıntılarını aynı anda vermede kullanılırlar.

Örneğin bir fonksiyonun grafiğini çizmek için ekran, biri -genellikle en büyük alanı- grafiğin tümüne, öbürü de grafiğin küçük parçalarının ayrıntılı çözümlemesine ayrılan iki ya da üç pencereye bölünebilir. Pencere oluşturulabilen sistemlerde işlem genellikle iki evrede gelişir. Birinci evre, pencereleri belirleyen parametrelere (boyutlar, konum, vb.) değerlerin verilmesiyle başlar, ardından daha önce verilmiş sayı ya da simgenin yardımıyla adreslenmiş pencere üzerinde çalışma evresine geçilir.

Pencerelerin yönetilmesinin sağladığı olanaklar bir makineden ötekine değişir. Bununla birlikte üç büyük kategori vardır:

- Pencerelerin öngörülmediği makineler
- Biri metin öbürü grafik için ancak iki pencerenin oluşturulmasına olanak veren bilgisayarlar.
- Hem sayıları hem de modları (grafik ya da metin) açısından pencere yönetimine olanak veren bilgisayarlar.

İlk iki kategori hemen hemen birbirinin aynıdır. Bir metin penceresinin genellikle birkaç satırla sınırlı görselleştirilmesi, kullanıcıyla diyalogun ve grafik sayfanın görüntülenmesinin aynı zamana denk düşmesi istendiğinde gerekli olur. Bu duruma örneğin 4 satır yüksekliğinde bir metin penceresi bulunduran Apple bilgisayarlarında raslanır (HGR komutu). Dolayısıyla ekranı birbirinden bağımsız pencerelere bölme sözkonusu değildir; ancak nisbeten basit programlarla kullanıcı bunu kendisi gerçekleştirebilir.

Pencere Oluşturma Programı

Pencere bir sayı ya da simgeyle belirtilen, çoğunlukla kare ya da dikdörtgen biçimde bir ekran alanıdır. Bunu kullanabilmek için, grafik programın parametrelenmiş olması, yani pencerenin parametreleri doğrultusunda bütün konumlandırma ve yer değiştirmeleri yönetmesi gerekir.
Bu parametreleme, elbette ancak ekranın bölümlenmesi kullanıcı tarafından yazılan programlarla elde edildiği zaman gereklidir. Sistem düzeyinde pencereleri öngören makinelerde, parametreleme kullanıcıya gözükmez; kullanıcının çalışma penceresini çağırması yeterlidir.

Sayfa 1566'da hem temel sisteme hem de bir pencerenin iki kenarıyla çalışan bir referans sistemine göndermeyle çizilmiş bir dairenin resmi bulunmaktadır. Pencere içindeki bir resim sözkonusu olduğunda, resmin koordinatlarının referans sisteminin koordinatlarına göre belirtilmesi gerekir. R yarıçapında bir daire, her noktanın koordinatlarını aşağıdaki formüllerle hesaplayan bir döngünün yardımıyla gösterilir.

```
X = X0 + R*COS(A)
Y = Y0 + R*SIN(A)
```

X0 ve Y0 merkezin koordinatlarıdır. Daireyi çerçeveye aktarmak için, merkezin koordinatlarının, pencerenin koordinatlarına oranla belirtilmesi yeterlidir. Kullanılan simgeler s.1567'deki çizimde görülmektedir. Dairenin, pencereye göre herhangi bir noktasınınX,Y

koordinatları, temel sisteme göre XV,YV koordinatları olarak elde edilmelidir. Bu, bilgisayar tarafından, elektronik kalemin yer değiştirmelerini adreslemekte kullanılır. Yani;

XV = XR + X
YV = YR + X

Aynı biçimde, temel sisteme göre alındığında merkez koordinatları;

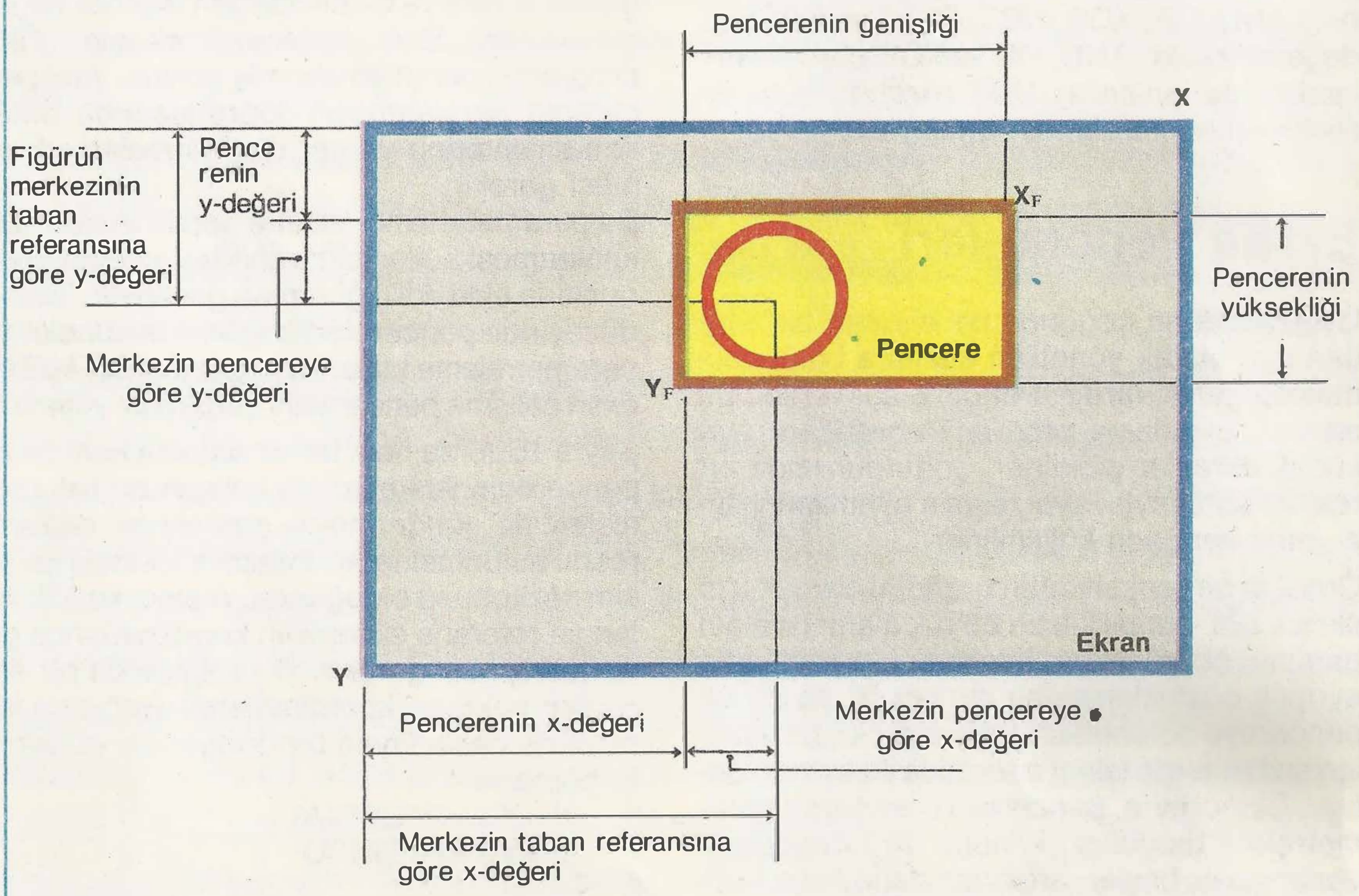
XOV = XR + XO
YOV = YR + YO haline gelir.

Daireyi pencerenin içine yerleştirmek için, her noktanın koordinatlarına pencerenin başlangıç noktasının koordinatlarını eklemek yeterlidir. Bu özel durumda figürün her noktasının koordinatlan merkezin koordinatlanna aktarılır ve toplam da, her noktada merkezdeki kadar doğru olur.

Başka deyişle, bir figürün pencerenin içine yerleştirilmesi yalnızca referans sisteminin aktarılması haline gelir. Bu iki işlem biçimi (ya her noktanın ya da tek bir X referans noktasının koordinatlarının aktarılması), her noktanın koordinatlarının kenar olarak belirli bir noktaya gönderme yapması koşuluyla, merkezi olmayan figürler için de geçerlidir.

1568. sayfadaki akış şeması aynı anda birçok pencereyi kullanan bir örneklendirme programıdır. Figür herbiri pencereye, o pencereye ait koordinatlar sistemine gönderme yaparak

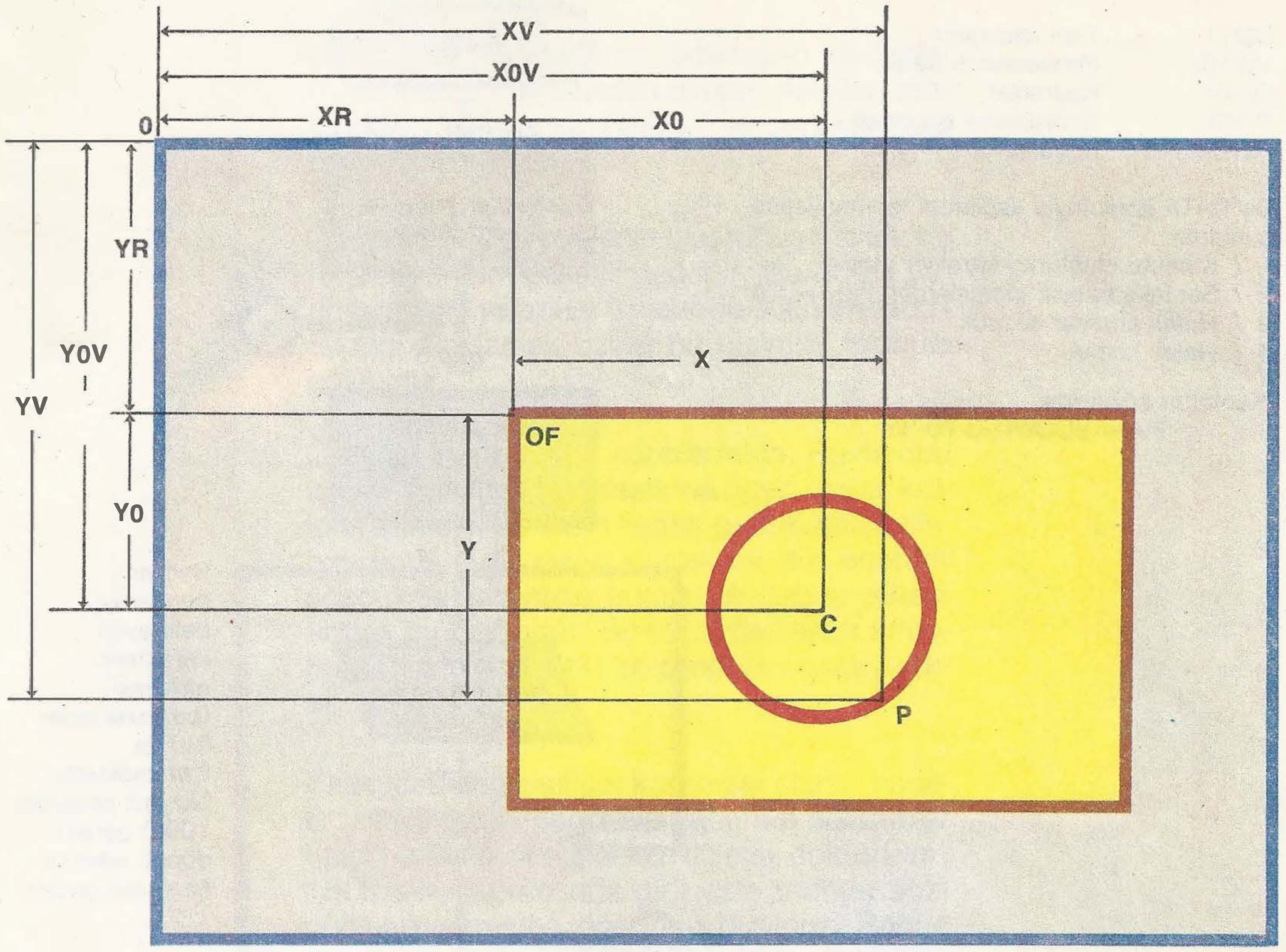
EKRAN PENCERESİ İÇİNDE FİGÜRÜN YÖNETİLMESİ



Kullanılan ekran alanı ve taban referansı

Pencere ve alanın referans sistemi

PENCERELERİN YÖNETİMİNDE KULLANILAN SİMGELER



çizilebilir. Böylece program varolan pencere sayısı kadar ayrı ekran varmış gibi hareket eder.

Pencereler zincir biçiminde tanımlandığına göre, bu program bir çevirici gibi yapılandırılmıştır. 1000 alt-programı parametreleri çıkararak bunları yorumlar. Kullanılan biçim bu tür uygulamalar öngören sistemlerde pencerelerin oluşturulmasında en çok kullanılan BASIC komutuna denk düşmesi gözetilerek seçilmiştir. Taklit etmek istediğimiz sözdizimi şudur:

- herhangi bir ilk harf, oluşmakta olan çerçeveyle özdeşlenir
- = simgesi sisteme sonraki sözcüğün anahtar sözcük olduğunu gösterir
- anahtar sözcük (WINDOW) istenen fonksiyonu tanımlar (pencerenin yaratılması)
- parantez içirideki bir dizi parametre pencerenin geometrik yapısını çizer.

Parametreler şunlardır:

q	Ekran kadranını belirtir (1,2,3,4)
p	Kadranın içindeki konumu (1'-den 4'e) verir
nnn, mmm	Her biri en çok üç sayıdan oluşan ve ekran noktası olarak sırasıyla pencerenin genişliğini ve yüksekliğini veren sayısal değerler.

Elde edilecek zincirin biçimi şöyle olur:

V = WINDOW(1—4—50—20)

Bu formül V harfiyle gösterilen, 4. konumda 1. kadranı dolduran, 50 × 20 boyutlarında bir pencere tanımlar. Parametreleri çıkaran alt-programlar komut zincirinin değişik bölümleri arasındaki olası boşlukları ortadan kaldıracak biçimde yapılandırılmıştır: Örneğin, WINDOW anahtar sözcüğü = simgesine oldukça bitişik olabilir.

Tek sınırlandırma, parametrelerin tanımını içeren parantezlerin içinde boşlukların bulunma-

PENCERELERİN OLUŞTURULMASI

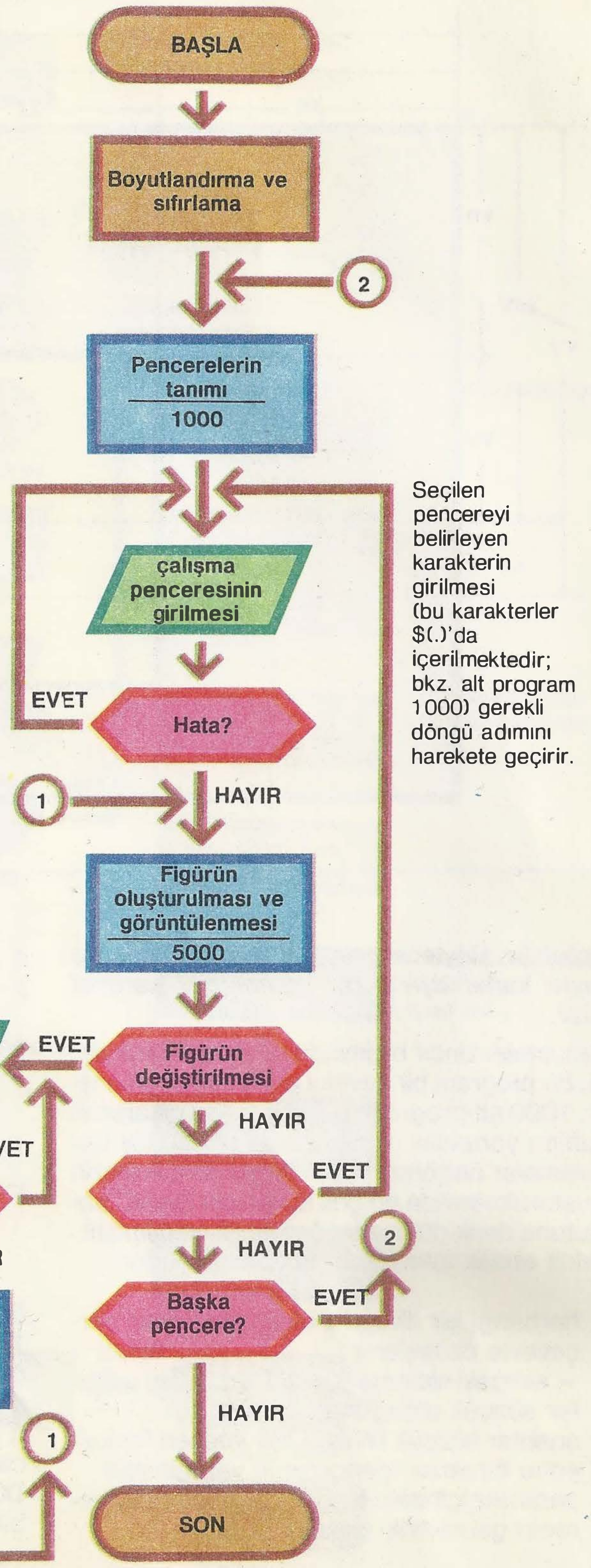
D$(4)	Tanı mesajları
V$(16)	Pencerelerin adları
Q(16)	Kadranlar
P(16)	Kadranların boyutları
H(16),L(16)	Boyutlar

Bir DATA komutuyla sağlanan tanı mesajları şunlardır:

1 / Komutu oluşturan karakter zinciri
2 / Sentaks hatası, simgelerden biri (=,(.))
3 / Hatalı anahtar sözcük
4 / Hatalı komut

Komutun sözdizimi

F=WINDOW(1-3-70-35)



Seçilen pencereyi belirleyen karakterin girilmesi (bu karakterler $(.)'da içerilmektedir; bkz. alt program 1000) gerekli döngü adımını harekete geçirir.

NF seçilen pencerenin numarasını içeriyor

Programın bu bölümünde figürün kenar sayısını belirleyen ve 200 alt program tarafından sağlanan K değeri değiştirilemez.

masıdır (aksi takdirde örneğimiz fazlasıyla karmaşıklaşırdı).

Pencerenin boyutlarını belirleyen değerler (nnn, mmm) 1-3 sayıdan oluşur. Alanlar, aralarındaki ayırma simgesi (-) ve parantezlerle belirlenir.

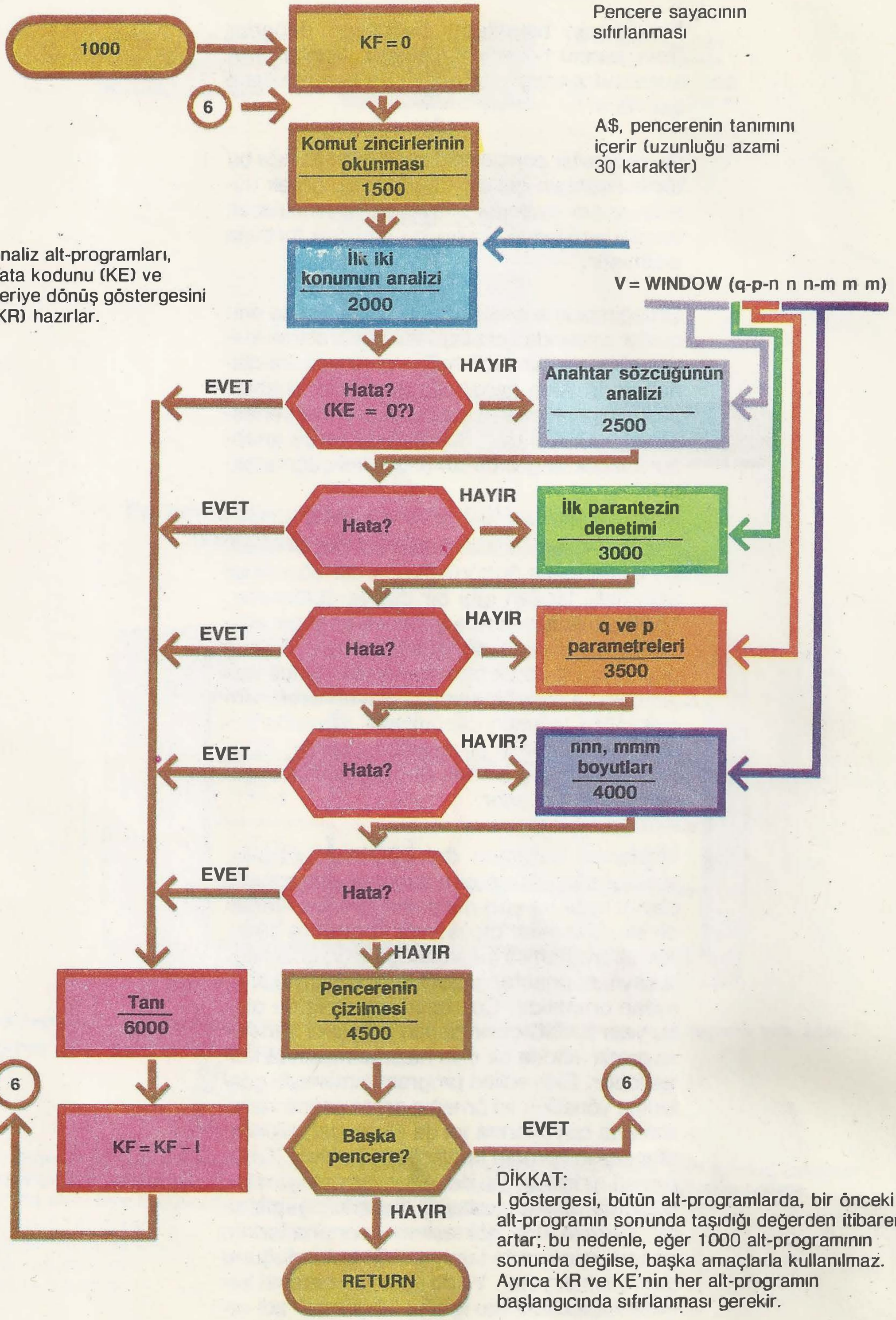
Gerçekte, bir pencerenin oluşturulmasında bu tür bir yöntem gerekli değildir. Bu örnek uygulayıcının yazması sırasında başvurulacak tekniği açıklamak üzere bu biçimde formüle edilmiştir.

Örneğimizde anahtar sözcük tektir. Bu da onu tanılar sırasında karşılaştırma ögesi olarak kullanılabilecek bir sabit haline getirmemize olanak verir. Ama genel olarak bir dizi anahtar sözcük tanımlayarak bunları bir dizeye yerleştirmek mümkündür. But akdirde herbir anahtar sözcük başka bir alt-programı çağıracaktır.

Yapılacak değişiklikler kuramsal olarak basittir: başlangıçta öngörülen anahtar sözcükler tıpkı hata tanıları gibi bir dizeye doldurulur. 1500 alt-programında girilecek anahtar sözcüğün tanımlama döngüsü bulunur. Sonuç olumluysa (sözcük tanımlanmışsa), zincir kullanıcıya ait dilin bir komutu gibi alınarak, zincirin öbür terimleri çözümlenir. Bununla birlikte, bu yöntemin bir sakıncası vardır: kontrol ve parametre üretme alt-programlarının çoğalmasına yol açar.

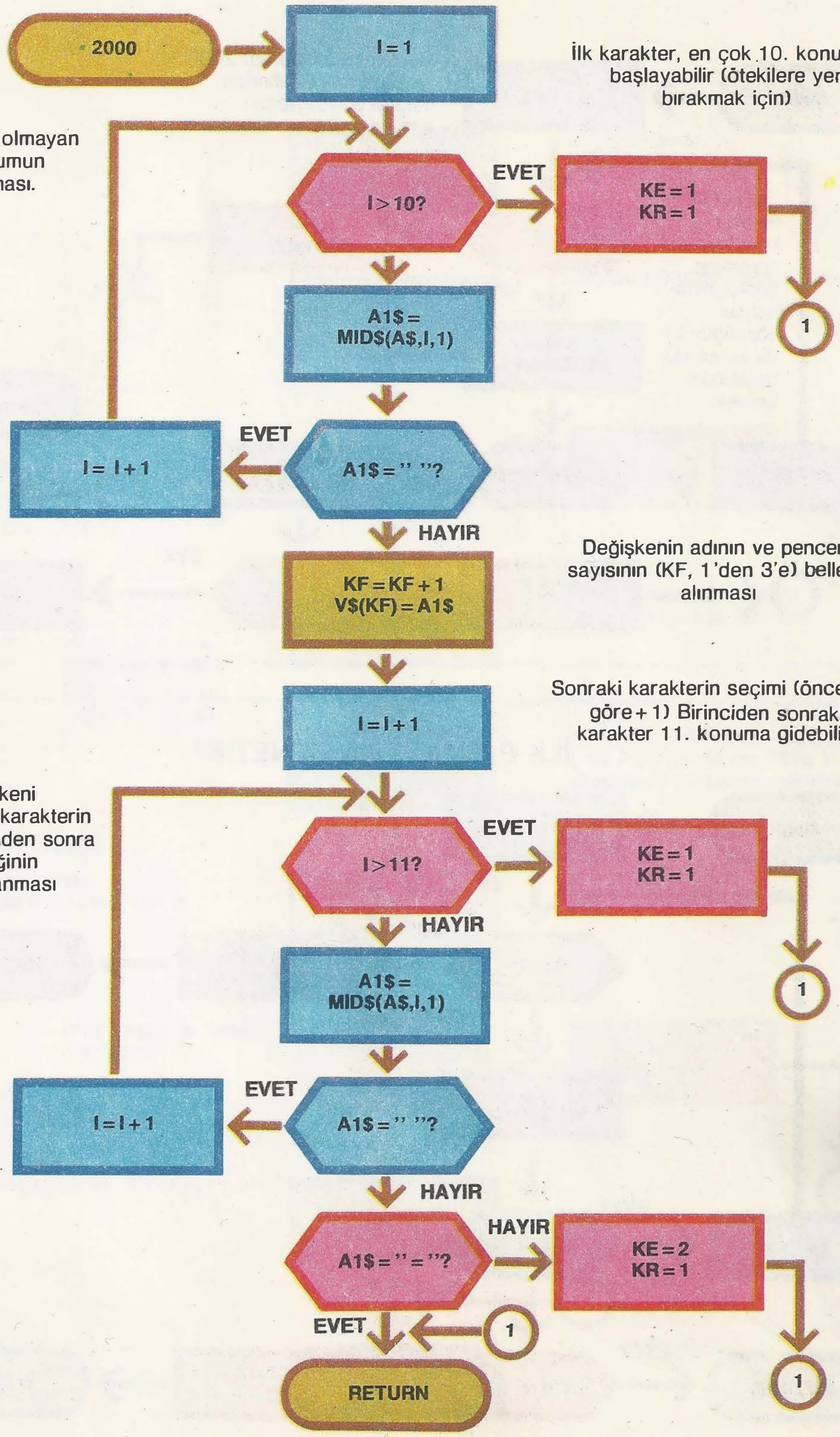
İncelenen durumda (tek anahtar sözcük), anahtar sözcüğü izleyen parametreler zorunlu olarak hazır bir yazı düzenine girerler. Genelde ise, olasılıklar ölçüsünde kontrol ve çıkarma alt-programını çağırmak gerekir; dolayısıyla sayıları, anahtar sözcüklerin sayısıyla doğrudan orantılıdır. Çok basitleştirilmiş de olsa bu yapı BASIC çeviricisinin işleyişini yansıtır ve grafik mödda bir dilin hazırlanmasında kullanılabilir. Elde edilen program tamamıyla gösterime yöneliktir ve örneğin çerçevelerin ve resimlerin boyutlarına ya da ölçek değişikliklerine ilişkin zorunlu kontrolleri içermez. Kullanıcının girişte doğru değerler vermesi gerekir. Bununla birlikte, kolayca değişiklik yapılabilir: Resmin her noktasının koordinatlarının pencerenin içinde tam yerinde bulunduğunu doğrulamak yeter. Ya da figür pencerenin kenarlarında kesilir (bu işleme ''budama'' adı verilir).

PENCERELERİN TANIMLANMASI ALT-PROGRAMI

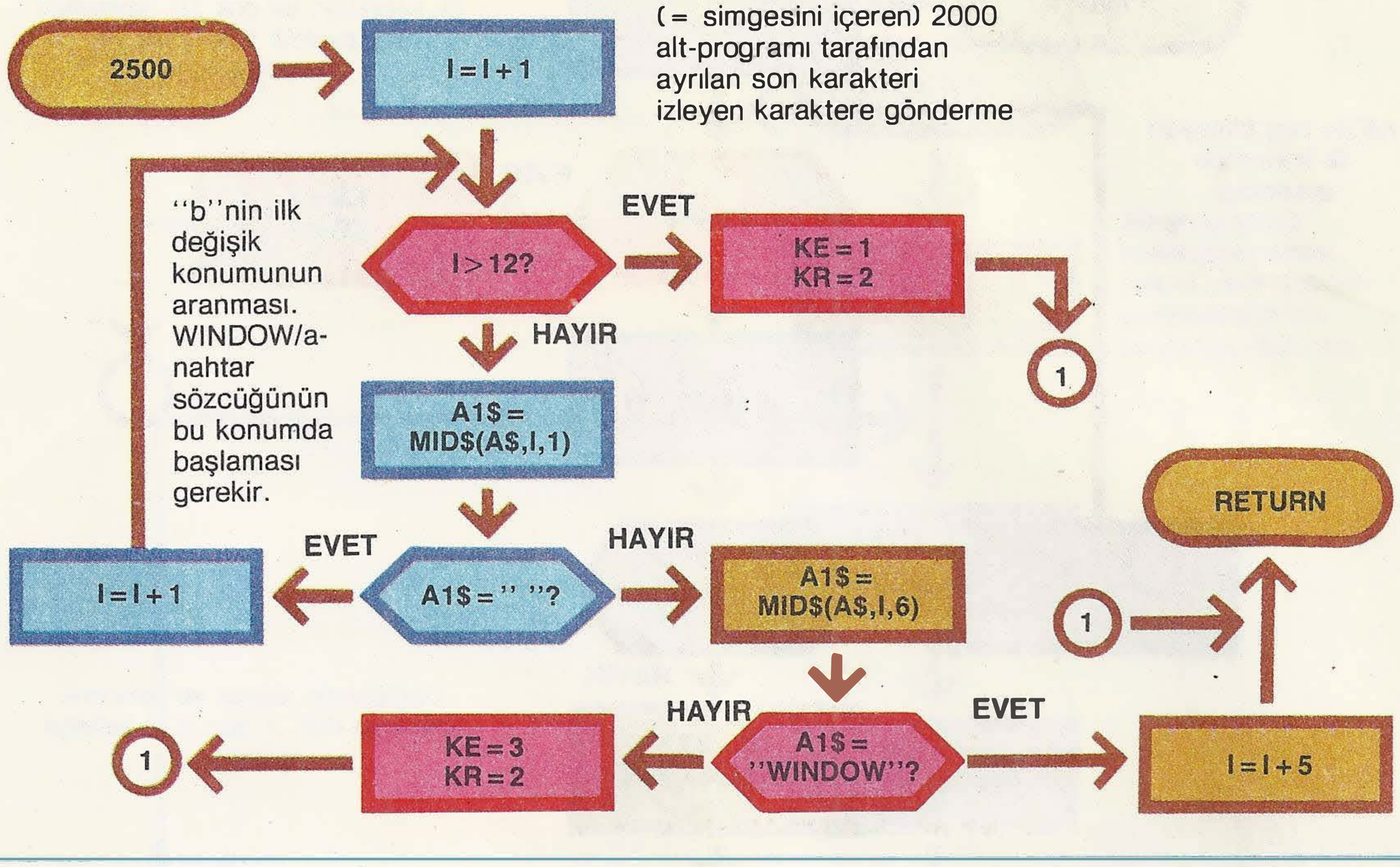


DİKKAT:
I göstergesi, bütün alt-programlarda, bir önceki alt-programın sonunda taşıdığı değerden itibaren artar; bu nedenle, eğer 1000 alt-programının sonunda değilse, başka amaçlarla kullanılmaz. Ayrıca KR ve KE'nin her alt-programın başlangıcında sıfırlanması gerekir.

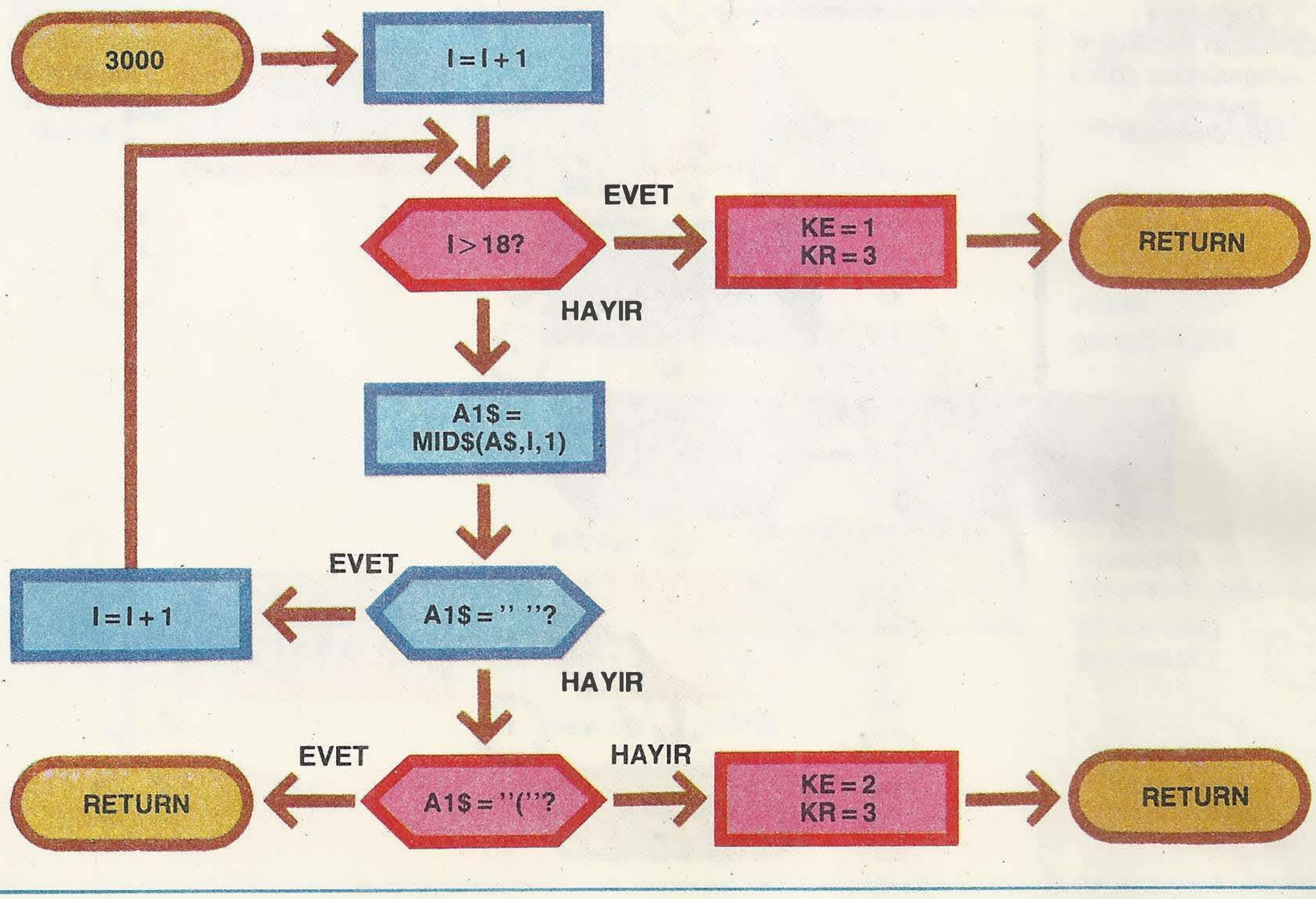
İLK İKİ KARAKTERİN ANALİZİ ALT-PROGRAMI



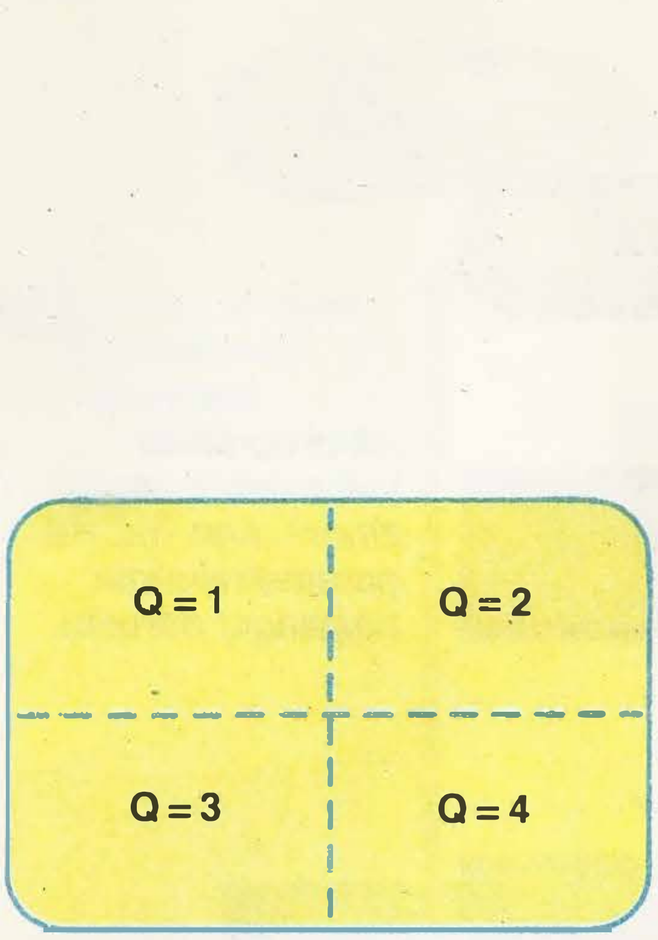
ANAHTAR SÖZCÜĞÜN ANALİZİ



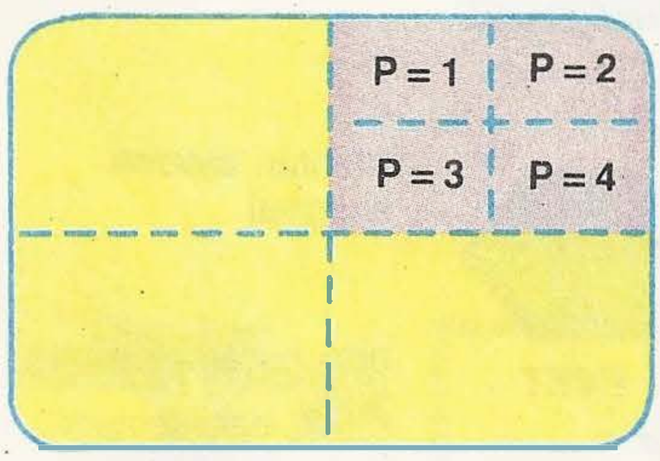
İLK PARANTEZİN DENETİMİ



Q VE P (KARE VE KONUM) PARAMETRELERİNİN DENETİMİ

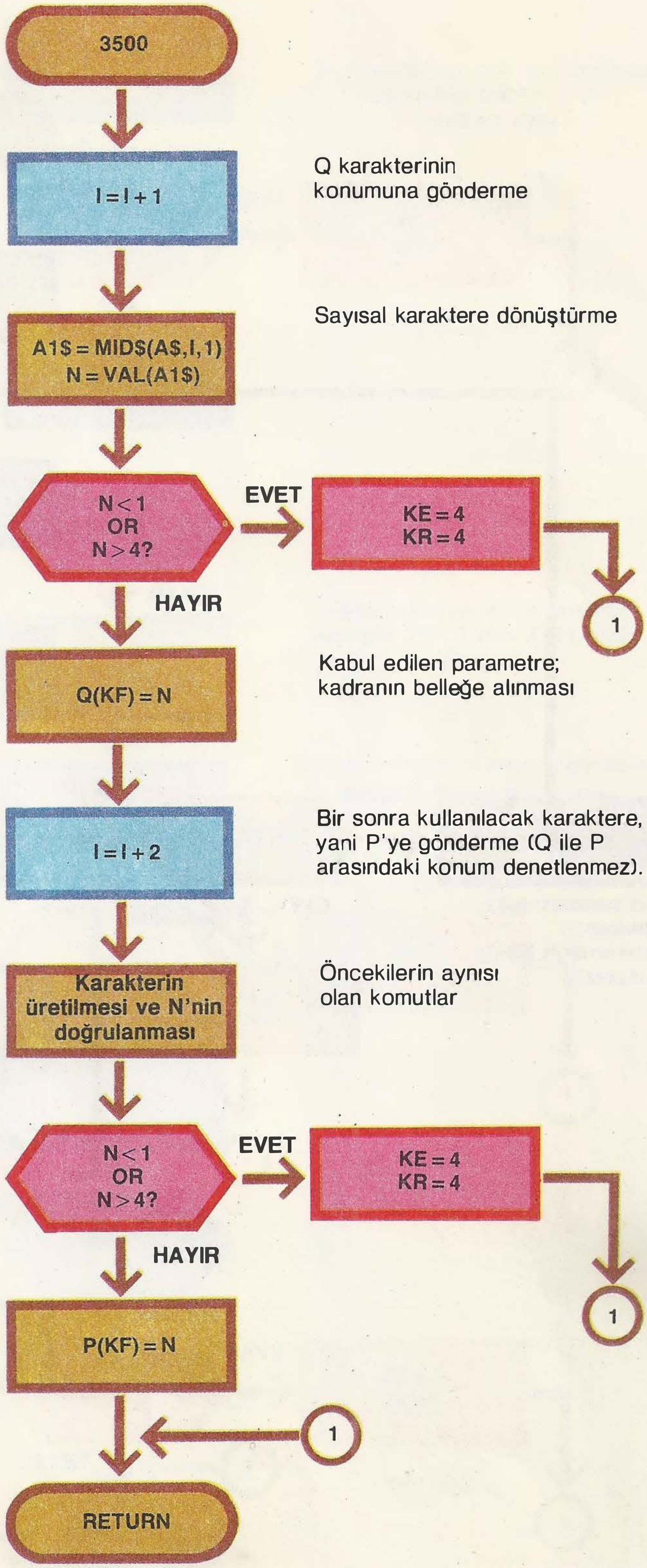


Pencerenin Q'ya göre konumu



Verili bir kadranda, pencerenin P'ye göre konumu

P'nin doğru değerleri 1,2,3,4'tür.



Q karakterinin konumuna gönderme

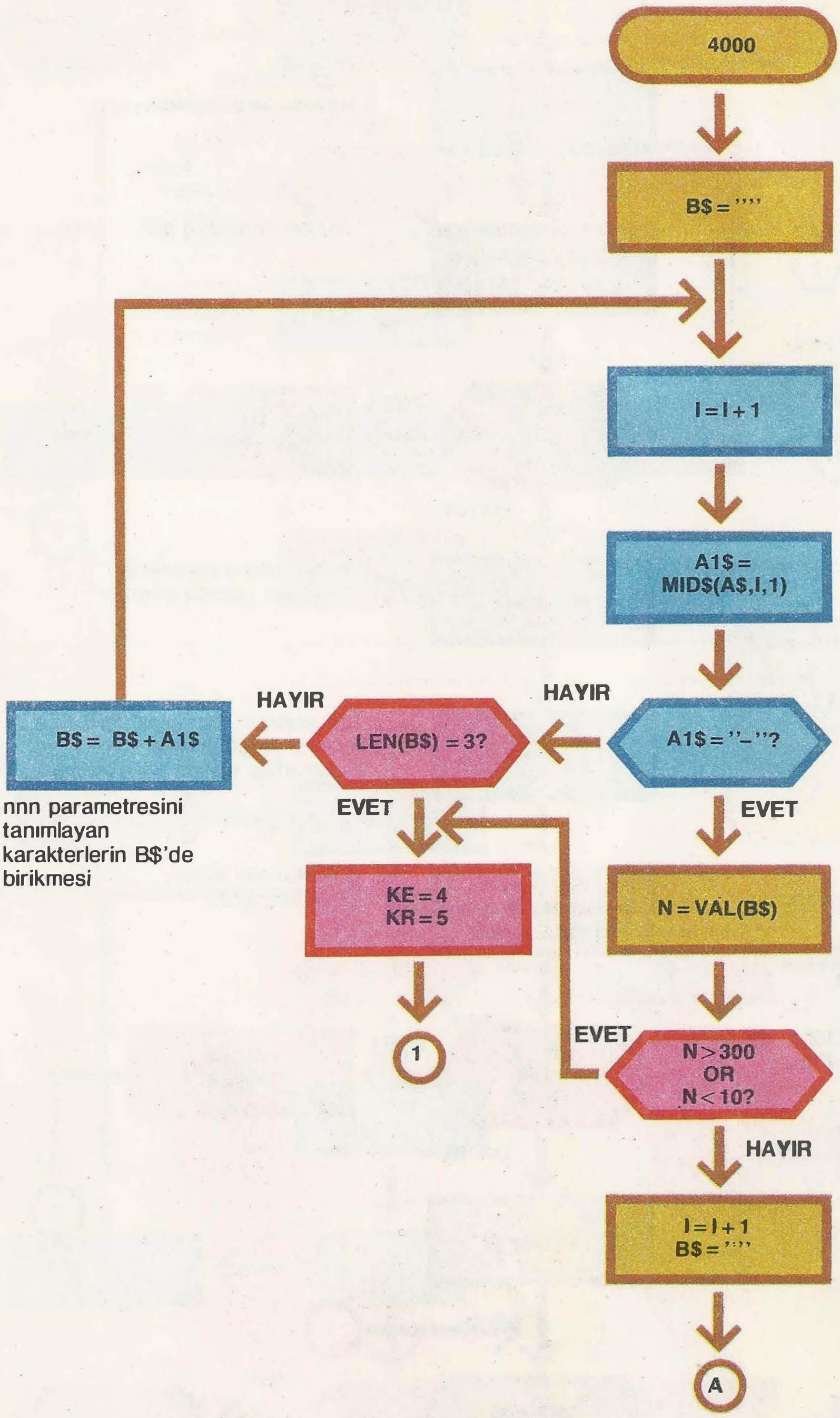
Sayısal karaktere dönüştürme

Kabul edilen parametre; kadranın belleğe alınması

Bir sonra kullanılacak karaktere, yani P'ye gönderme (Q ile P arasındaki konum denetlenmez).

Öncekilerin aynısı olan komutlar

BOYUTLARIN DENETİMİ (nnn, mmm PARAMETRELERİ)



Alt-programda kullanılan destek zinciri. nnn (taban) parametrelerinin başlangıç durumu.

Alanları ayırma simgesi

nnn parametresinin sonu

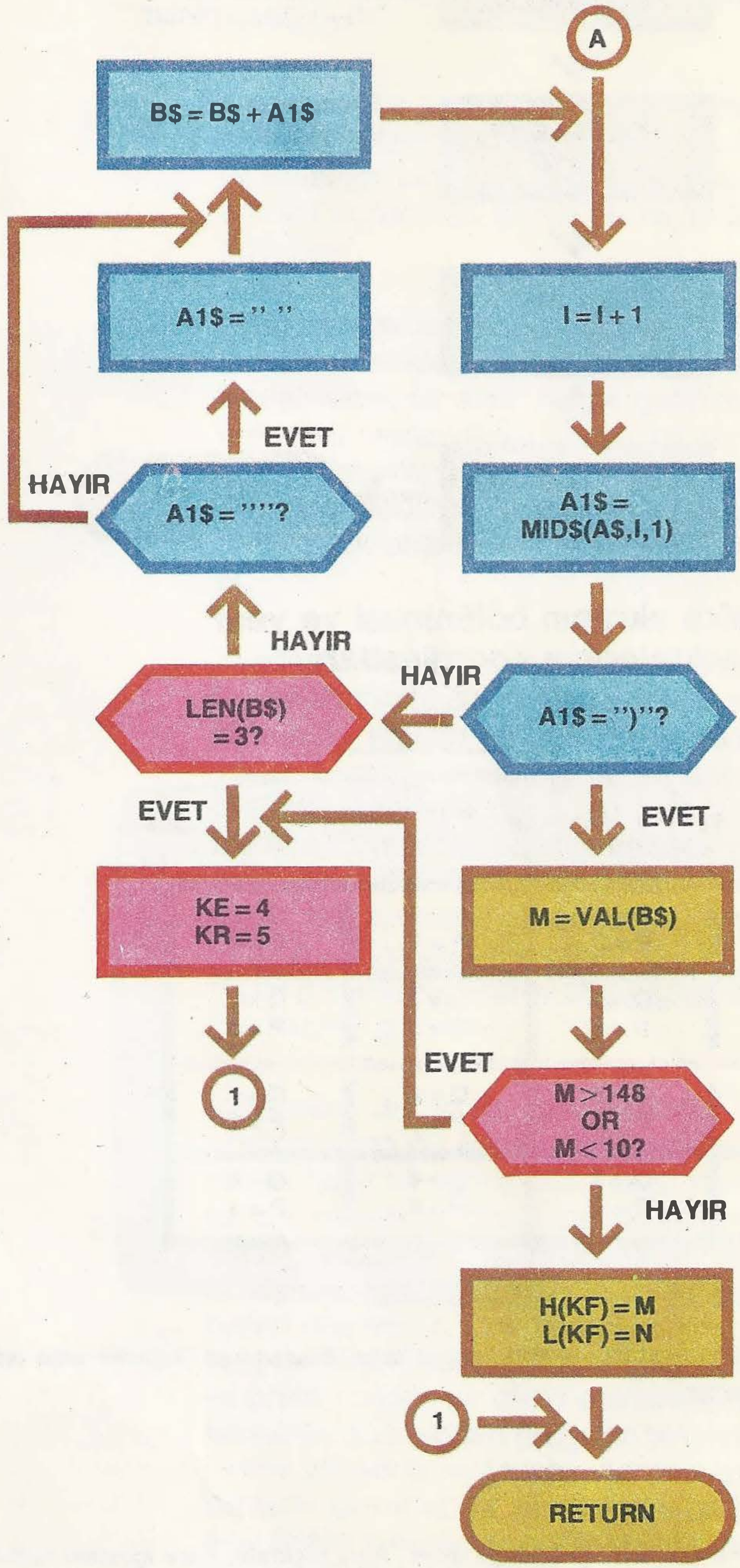
nnn parametresini tanımlayan karakterlerin B$'de birikmesi

Sayıyı dönüştürme

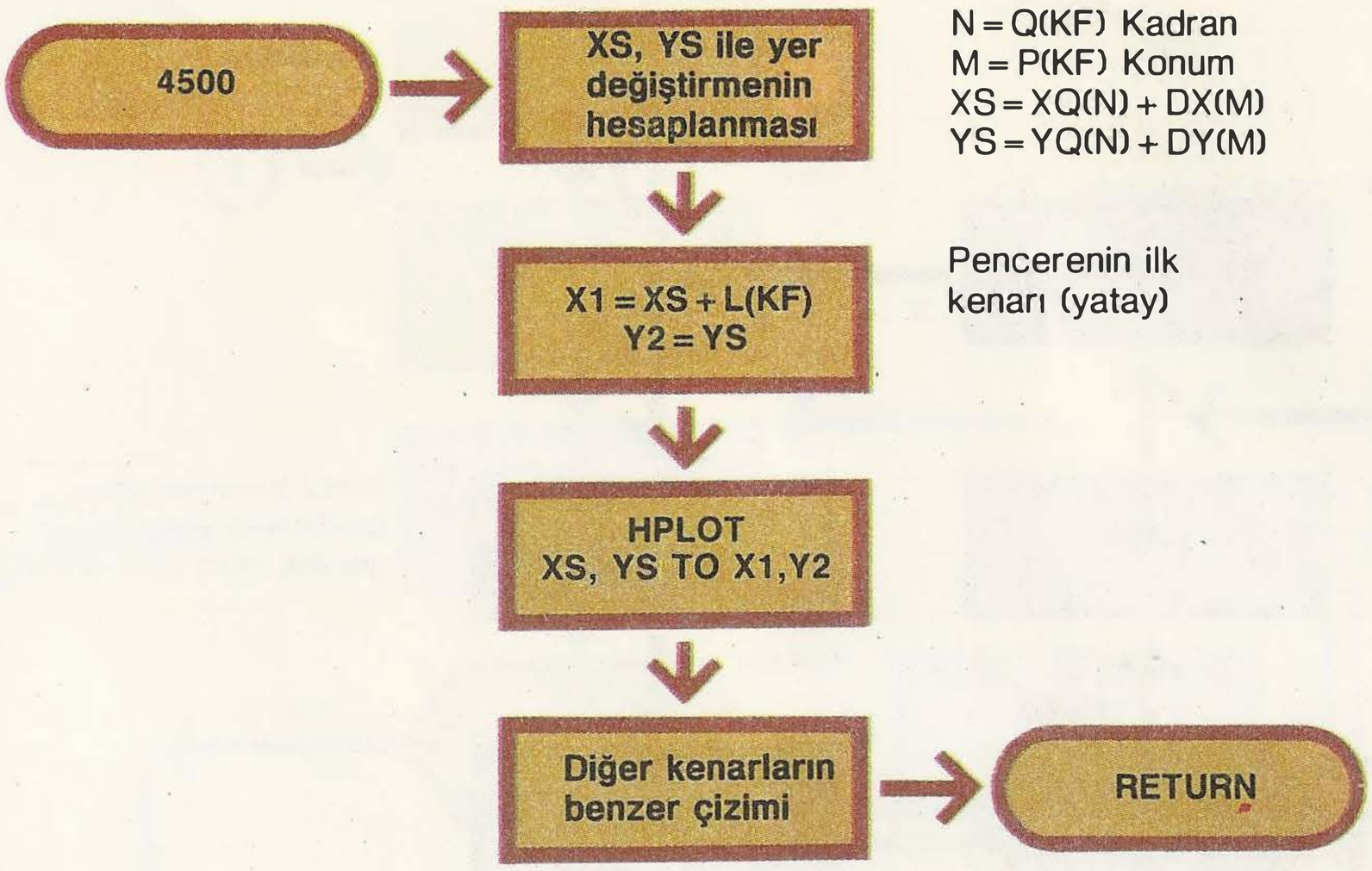
Azami genişlik = 300

İki parametre arasındaki — simgesinin atlanması

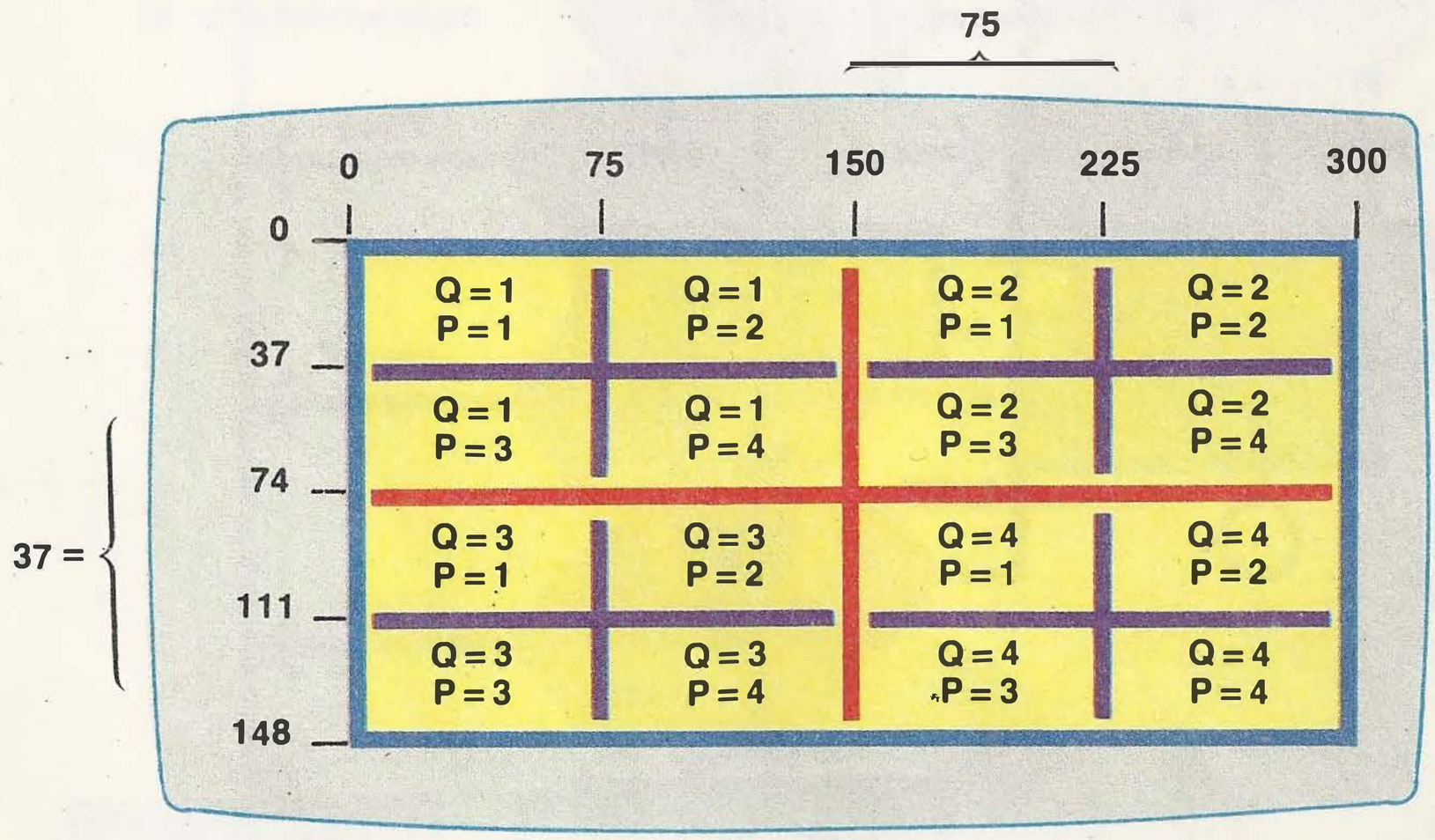
B$'nin sıfıra getirilmesi


A
B$ = B$ + A1$
I = I + 1
A1$ = " "
İkinci parametrenin üretilmesi döngüsü için gerekli olan yeni artma
EVET
HAYIR
A1$ = ""?
A1$ = MID$(A$,I,1)
HAYIR
HAYIR
LEN(B$) = 3?
A1$ = ")"?
mmm (yükseklik) parametresinin sonu
EVET
EVET
KE = 4
KR = 5
M = VAL(B$)
1
EVET
M > 148 OR M < 10?
HAYIR
H(KF) = M
L(KF) = N
1
RETURN

PENCERENİN GÖRÜNTÜLENMESİ ALT-PROGRAMI



Parametrelere göre ekranın bölünmesi ve yeni başlangıç noktalarının koordinatları



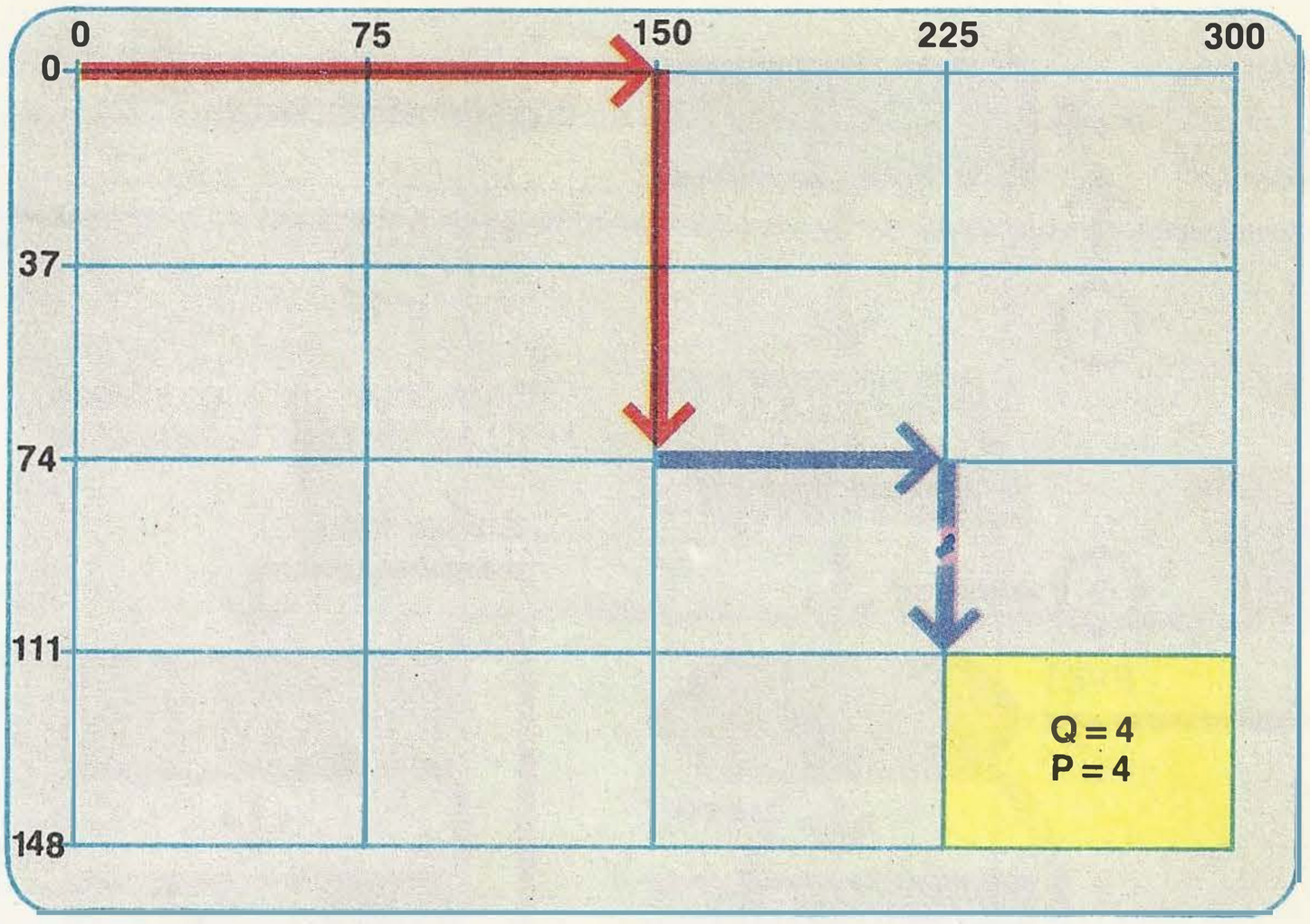
Her karenin (kırmızı alt-bölümler) referans noktası yukarıda sol uçtadır. Böylece şu değerler elde edilir:

Kadran	Dikey Koordinatlar
1	0,0
2	150,0
3	0,74
4	150,74

Bu değerler (bir DATA komutuyla) XQ(4) ve YQ(4)'te belleğe alınır. Aynı biçimde, kare içindeki konumun başlangıç noktası, karenin koordinatlarına göre yukarıda solda bulunan (DX(4) ve DY(4)'te belleğe alınmış olan) tepe koordinatları tarafından belirlenir. Değerler şunlardır:

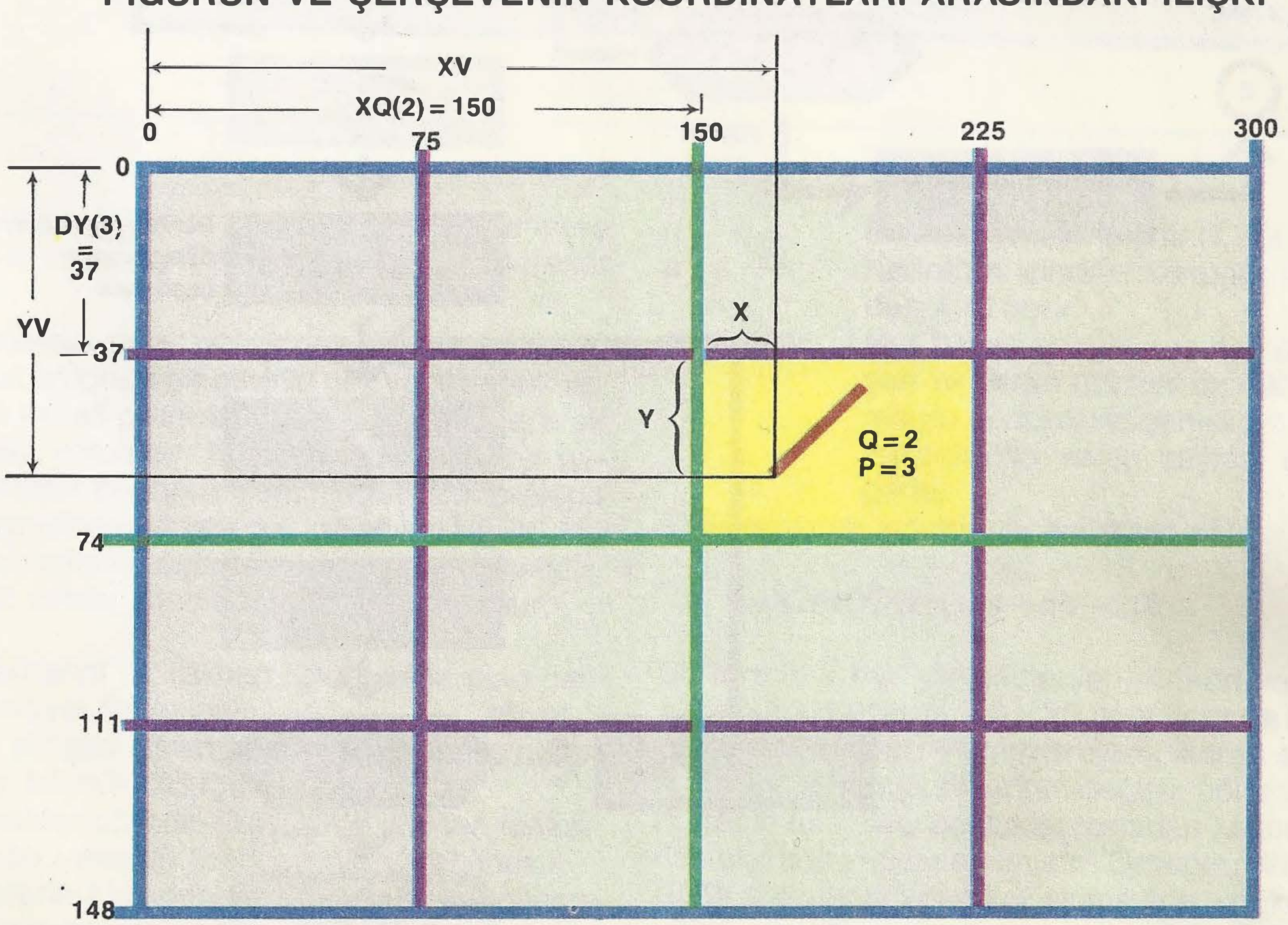
XQ(.) = 0,150,0,150
YQ(.) = 0,0,74,74
DX(.) = 0,75,0,75
DY(.) = 0,0,37,37

KARE VE KONUM KOORDİNATLARININ KULLANILMASI

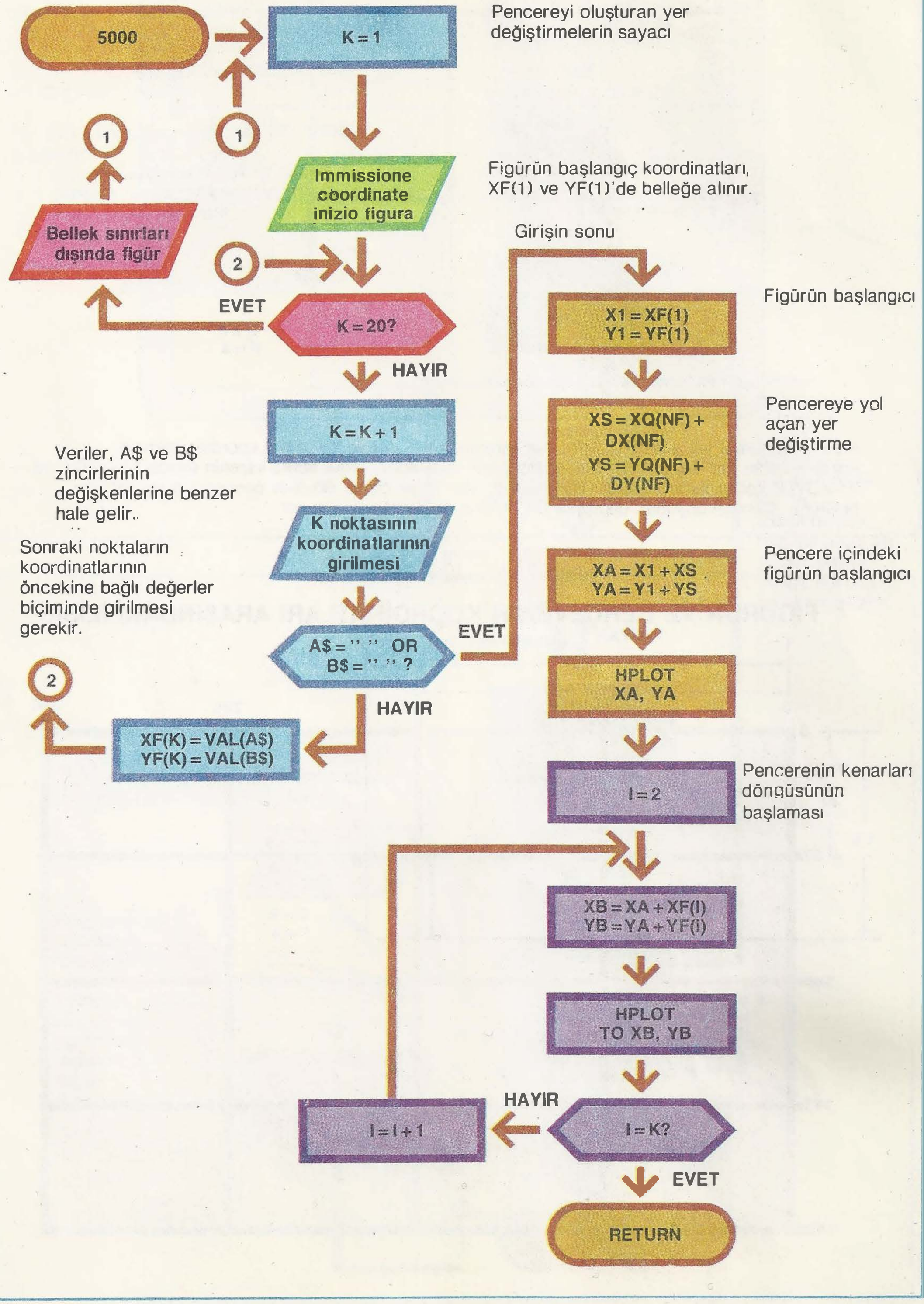


Kare 4, konum 4'teki pencere, öncelikle karenin, X = XQ(4) ve Y = YQ(4) koordinatlarındaki noktaya kadar yer değiştirmesiyle (kırmızı çizgi) elde edilir. Daha sonra karenin içinde X1 = DX(4) ve Y1 = DY(4) koordinatlarına kadar ilerlemek gerekir (mavi çizgi). Böylece pencerenin sol üst ucu belirlenir. Ekranın boyutları değişirse DATA'ların da değişmesi gerekir.

FİGÜRÜN VE ÇERÇEVENİN KOORDİNATLARI ARASINDAKİ İLİŞKİ



FİGÜRÜN SAKLANMASI VE GÖSTERİLMESİ



ÇERÇEVELERİN YARATILMASI VE YÖNETİMİ

```
1       REM ====================
2       REM * SAYFA HGR2'DE
3       REM * CERCEVELERIN
4       REM * OLUSTURULMASI
5       REM * PROGRAMI
6       REM ====================
7
    :
8
    :
10      HOME
100     GOSUB 10000
110     GOSUB 500
115
    :
120     REM ====================
130     REM * CAGIRMA CERCEVE
140     REM ====================
145
    :
147     TEXT
150     HOME
    : PRINT "CERCEVELER HAZIR :"
160     VTAB 3
    : PRINT I$
170     FOR J = 1 TO KF
180     PRINT V$(J);
    : HTAB 11
    : PRINT Q(J);
    : HTAB 23
    : PRINT P(J);
    : HTAB 31
    : PRINT H(J)"*"L(J)
200     NEXT J
210     VTAB 22
    : INPUT "CERCEVELERIN ADINI
      GIRIN : ";NM$
220     FOR J = 1 TO KF
230     IF NM$ = V$(J) THEN NF = J
    : J = KF
    : NEXT J
    : GOTO 250
240     NEXT J
245     GOTO 210
250     RETURN
499
    :
500     REM ======================
501     REM CAGIRMA ALT-PROGRAMLAR
502     REM ======================
503
    :
505     TEXT
510     GOSUB 1000
520     GOSUB 120
530     GOSUB 5000
540     TEXT
    : HOME
    : VTAB 10
    : INPUT "FIGURUN YERINI DEGISTIRMEK
              ISTIYORMUSUNUZ ?";RS$
550     IF LEFT$ (RS$,1) = "E" THEN 640
560     VTAB 10
    : INPUT "RESMI DEGISTIRMEK
              ISTIYORMUSUNUZ ? ";RS$
570     IF LEFT$ (RS$,1) = "E" THEN 520
580     VTAB 10
    : INPUT "CERCEVELERI DEGISTIRMEK
              ISTIYORMUSUNUZ ? ";RS$
590     IF LEFT$ (RS$,1) = "E" THEN 500
600     HOME
    : VTAB 22
    : END
640     GOSUB 120
    : GOSUB 7000
645     IF Q$ = " " THEN 655
650     GOSUB 5125
655     GOTO 540
1000
    :
1001    REM ====================
1002    REM CERCEVELERIN TANIMI
1003    REM ====================
1004
    :
1010    KF = 0
    : HGR2
1020    GOSUB 1500
1030    GOSUB 2000
1040    IF KE < > 0 THEN 6000
1050    GOSUB 2500
1060    IF KE < > 0 THEN 6000
1070    GOSUB 3000
1080    IF KE < > 0 THEN 6000
1090    GOSUB 3500
1100    IF KE < > 0 THEN 6000
1110    GOSUB 4000
1120    IF KE < > 0 THEN 6000
1130    GOSUB 4500
1140    TEXT
1150    HOME
    : VTAB 10
    : INPUT "BASKA CERCEVE ?";RS$
1160    IF LEFT$(RS$,1)="E"THEN 1020
1170    RETURN
1499
    :
1500    REM ====================
1501    REM CERCEVE TASVIRI
1502    REM ====================
1503
    :
1505    TEXT
1510    HOME
    : VTAB 10
    : INPUT "CERCEVE: ";A$
1520    IF LEN (A$) > 30 THEN 1510
1530    RETURN
2000
    :
2001    REM ====================
2002    REM CTRL KONUM 1-2
2003    REM ====================
2004
    :
2010    KR = 0
    : KE = 0
    : I = 1
2020    IF I > 10 THEN KE = 1
    : KR = 1
    : RETURN
2030    A1$ = MID$ (A$,I,1)
2040    IF A1$ = " " THEN I = I + 1
    : GOTO 2020
2050    KF = KF + 1
    : V$(KF) = A1$
2060    I = I + 1
2070    IF I > 11 THEN KE = 1
    : KR = 1
    : RETURN
```

```
2080    A1$ = MID$ (A$,I,1)
2090    IF A1$ = " " THEN I = I + 1
      : GOTO 2070
2100    IF A1$ = "=" THEN RETURN
2110    KE = 2
      : KR = 1
      : RETURN
2499
      :
2500    REM ==================
2501    REM CTRL WINDOW
2502    REM ==================
2503
      :
2510    KR = 0
      : KE = 0
      : I = I + 1
2520    IF I > 12 THEN KE = 1
      : KR = 2
      : RETURN
2530    A1$ = MID$ (A$,I,1)
2540    IF A1$ = "" THEN I = I + 1
      : GOTO 2520
2560    IF A1$="WINDOW" THEN I = I + 5
      : RETURN
2570    KE = 3
      : KR = 2
      : RETURN
3000
      :
3001    REM ==================
3002    REM CTRL "("
3003    REM ==================
3004
      :
3010    KE = 0
      : KR = 0
      : I = I + 1
3020    IF I > 18 THEN KE = 1
      : KR = 3
      : RETURN
3030    A1$ = MID$ (A$,I,1)
3040    IF A1$ = " " THEN I = I + 1
      : GOTO 3020
3050    IF A1$ = "(" THEN RETURN
3060    KE = 2
      : KR = 5
      : RETURN
3500
      :
3501    REM ==================
3502    REM CTRL O/P
3503    REM ==================
3504
      :
3510    KR = 0
      : KE = 0
      : I = I + 1
3520    A1$ = MID$ (A$,I,1)
3530    N = VAL (A1$)
3540    IF N < 1 OR N > 4 THEN KE = 4
      : KR = 4
      : RETURN
3550    O(KF) = N
      : I = I + 2
3560    A1$ = MID$ (A$,I,1)
      : N = VAL (A1$)
3570    IF N < 1 OR N > 4 THEN KE = 4
      : KR = 4
      : RETURN
3580    P(KF) = N
      : RETURN
```

```
4000
      :
4001    REM ================
4002    REM CTRL.PARAMETRELER
4003    REM ================
4004
      :
4010    KE = 0
      : KR = 0
      : B$ = ""
      : I = I + 1
4020    I = I + 1
      : A1$ = MID$ (A$,I,1)
4030    IF A1$ = "-" THEN 4070
4040    IF LEN(B$) = 3 THEN KE = 4
      : KR = 5
      : RETURN
4050    B$ = B$ + A1$
      : GOTO 4020
4070    N = VAL (B$)
      : IF N > 300 OR N < 10 THEN KE=4
      : KR = 5
      : RETURN
4090    I = I + 1
      : A1$ = MID$ (A$,I,1)
4100    IF A1$ = ")" THEN 4130
4110    IF LEN (B$) = 3 THEN K = 4
      : KR = 5
      : RETURN
4115    IF A1$ = "" THEN A1$ = " "
4120    B$ = B$ + A1$
      : GOTO 4090
4130    M = VAL (B$)
      : IF M > 140 OR M < 10 THEN KE=4
      : KR = 5
      : RETURN
4140    H(KF) = M
      : L(KF) = N
      : RETURN
4500
      :
4501    REM ====================
4502    REM  DISP. CER
4503    REM ====================
4504
      :
4505    POKE - 16304,0
      : POKE - 16297,0
      : POKE - 16299,0
4507    HCOLOR= 3
4510    N = O(KF)
      : M = P(KF)
4520    XS = XO(N) + DX(M)
      : YS = YO(N) + DY(M)
4530    X1 = XS + L(KF)
      : Y2 + YS
4535    IF X1 > 279 THEN X1 = 279
4537    IF Y2 > 191 THEN Y2 = 191
4540    HPLOT XS,YS TO X1,Y2
4550    Y2 = YS + H(KF)
4555    IF Y2 > 191 THEN Y2 = 191
4560    HPLOT TO X1,Y2
4570    X1 = XS
4575    IF X1 > 279 THEN X1 = 279
4577    HPLOT TO X1,Y2
4580    Y2 = YS
4585    IF Y2 > 191 TJHEN Y2 = 191
4587    HPLOT TO X1,Y2
4600    GET Q$
      : RETURN
5000
      :
```

```
5001  REM ======================
5002  REM * FIGURLER
5003  REM ======================
5004
5005  TEXT
5010  K = 1
5020  HOME
    : VTAB 10
    : PRINT "BASLANGIC KOORDINATLARINI
      SOKUN"
5030  VTAB 10
    : HTAB 30
    : PRINT  "X / Y"
5040  VTAB 11
    : HTAB 31
    : INPUT "";A$
5050  VTAB 11
    : HTAB 35
    : INPUT "";B$
5052  IF A$ = "" OR B$ = "" THEN 5020
5055  GOTO 5110
5060  IF K = 20 THEN 5700
5070  K = K + 1
5080  VTAB 10
    : HTAB 21
    : PRINT "NOKTA " K
5090  VTAB 11
    : HTAB 31
    : INPUT "";A$
5100  VTAB 11
    : HTAB 35
    : INPUT "";B$
5110  IF A$ = "" OR B$="" THEN K=K
        - 1
    : GOTO 5126
5120  XF(K) = VAL (A$)
    : YF(K) = VAL (B$)
    : IF XF(K) > 279 OR XF(K) < 1
        THEN 5000
5121  IF YF(K) > 191 OR YF(K) < 1
        THEN 5000
5122  GOTO 5060
5123
    :
5124  REM  >> BICIMLENDIRME <<
5125
    :
5126  HCOLOR = 3
5127  POKE - 16304,0
    : POKE - 16297,0
    : POKE - 16299,0
5130  X1 = XF(1)
    : Y1 = YF(1)
5140  XS = XQ(Q(NF)) + DX(P(NF))
    : YS = YQ(Q(NF)) + DY(P(NF))
5145  CF = NF
5150  XA = X1 + XS
    : YA = Y1 + YS
    : I = 2
5155  HPLOT XA,YA
5160  XB = XF(I) + XS
    : YB = YF(I) + YS
5170  REM
5175  IF XB >= XS+L(NF) THEN XB=XS
        + L(NF) - 1
5176  IF YB > = YS + H(NF) THEN YB = YS
        + H(NF) - 1
5180  HPLOT TO XB,YB
5190  IF I = K THEN GET Q$
    : RETURN
5200  I = I + 1
    : GOTO 5160
```

```
5700  TEXT
    : HOME
    : VTAB 10
5710  PRINT "EN FAZLA 10 NOKTA ";
    : GET Q$
6000
    :
6001  REM ======================
6002  REM * TANI
6003  REM ======================
6004
    :
6010  PRINT CHR$(7)
6020  VTAB 20
    : PRINT D$(KE);
    : GET Q$
6030  VTAB 20
    : PRINT BK$
6040  KF = KF -1
    : GOTO 1020
6999
    :
7000  REM ======================
7001  REM *FIGURLERIN SILINMESI
7002  REM ======================
7003
    :
7010  MF = NF
    : NF = CF
7020  HCOLOR = 0
    : GOSUB 5127
7030  NF = MF
    : RETURN
10000
    :
10001 REM ==================
10002 REM *HARFLENDIRME
10003 REM ==================
10004
    :
10005 HIMEM: 16383
    : HGR2
    : TEXT
10010 DIM D$(4),V$(16),Q(16),P(16),
        H(16),L(16),XF(20),YF(20)
10020 I$ = "AD/KADRAN/POZISYON/
        BOYUT"
10025 BK$ = ""
    : FOR J = 1 TO 30
    : BK$ = BK$ + " "
    : NEXT J
10030 FOR J = 1 TO 4
    : READ XQ(J),YQ(J),DX(J),DY(J)
    : NEXT J
10040 FOR J = 1 TO 4
    : READ D$(J)
    : NEXT J
10050 RETURN
10100 DATA 0,0,0,0,150,0,75,0,0,74,0,
        37,150,74,75,37
10110 DATA "YETERSIZ VERILER"
        ,"SENTAKS HATASI"
        ,"YANLIS ANAHTAR SOZCUK","
        PARAMETRELERE HATA"
```

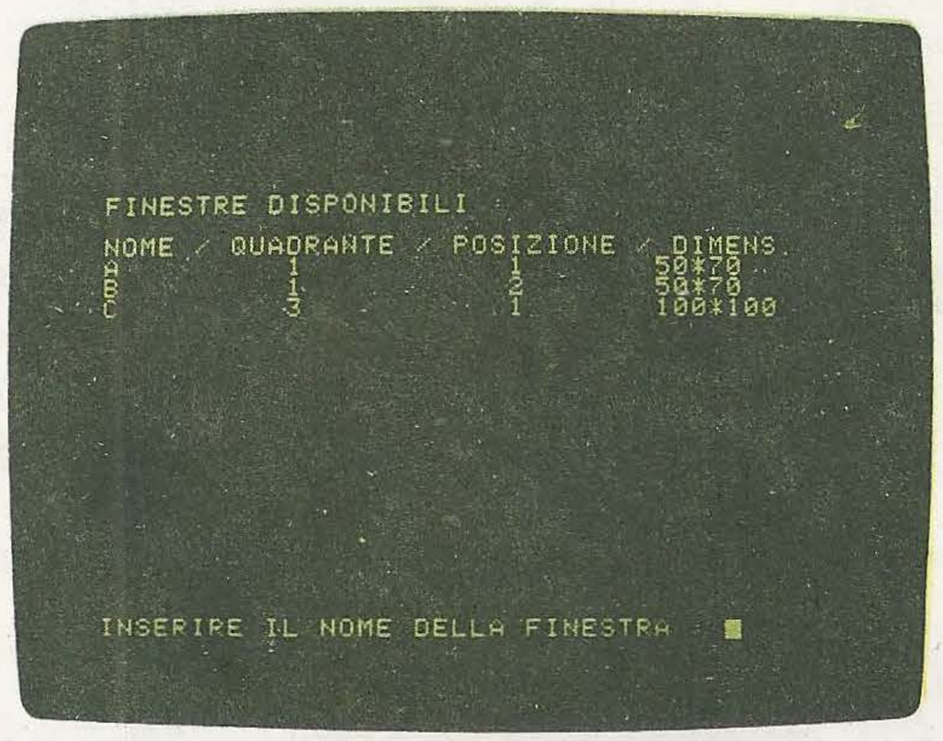
EKRAN PENCERELERİNİN OLUŞTURULMASI VE YÖNETİMİ

Burada, 1579 ve 1581. sayfalar arasında sunulan programın işleyişine ilişkin örnekler verilmektedir. Yandaki ekranda, 1,1 noktasında (üstte solda) yer alan, 50*70 noktalık bir A penceresinin tanımlanma evresi görülmektedir. Tersine bir sırayla yer alan bu parametrelerden önce, WINDOW anahtar sözcüğü bulunmaktadır.

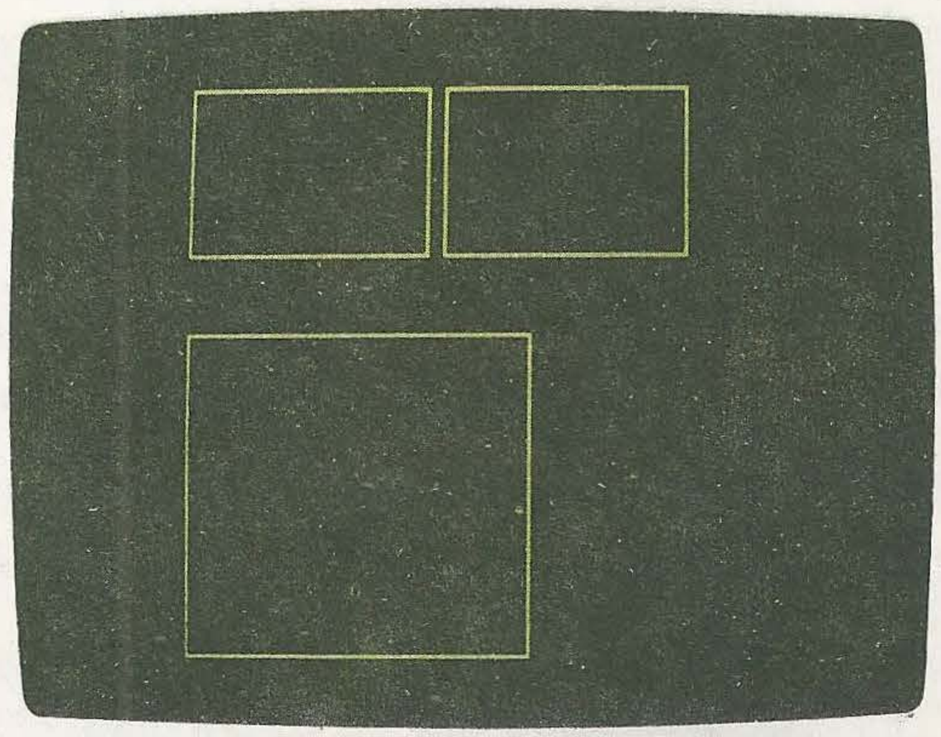


İki tanesinin daha tanımlanmasından sonra, program pencerelerin listesini verir.

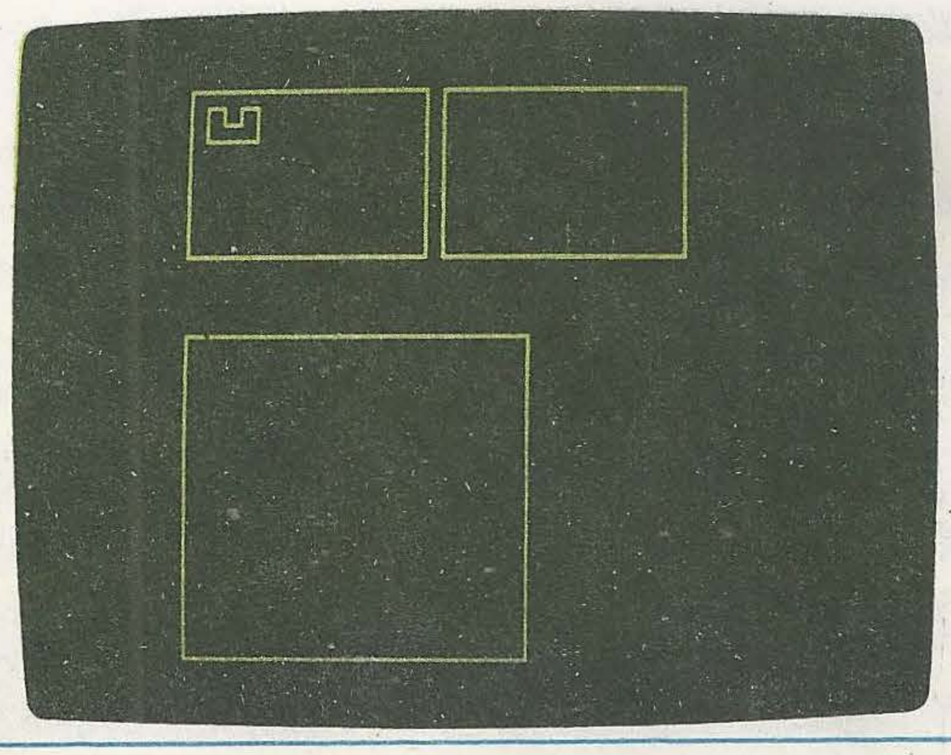
Pencere, kendisini tanımlayan adın girilmesiyle harekete geçirilir.



Kullanıcının isteği üzerine, başta tanımlanan üç çerçeve görüntülenir.



Grafik figür çizilmiştir. Figür konumlandırma komutları yardımıyla, harekete geçirilmiş üç çerçeveden herhangi birinin içine konur.



Bilgisayar ve Yelkenli

22 Ağustos 1851'de Amerikan yelkenlisi America'nın kazandığı ünlü Amerika kupası dünyada bütün deniz sporlarını heveslendiren bir kupadır.

1851'de kazanılan kupa Yeni Kıta'nın yelkenli dünyasında Eski'sine karşı ilk zaferiydi. O sıralar Yüz Metelik Kupası adını taşıyan kupa, Amerika Kupası olarak yeniden adlandırıldı ve ününü pekiştiren kahramanca çabalara rağmen, aralıksız 132 yıl boyunca Amerikalıların elinde kaldı.

1983'deki yarışla America Kupası basit bir ödül olmaktan öte bir anlam kazandı. O yıl yarışmanın tarihinde bir dönüm noktası oldu. Öncelikle, Australia II'nin Liberty karşısında, Amerikalıların yenilmezliğini bir ölçüde sarsan zaferi gerçekleşti.

Ardından, yarışla doğrudan ilişkili olmayan bir kitlenin artan ilgisinden sözetmek gerek. Yarışmanın hızındaki değişikliğe büyük ölçüde katkıda bulunan bir başka etken de elektronik ve bilgisayardan gitgide daha çok yararlanılmasıydı.

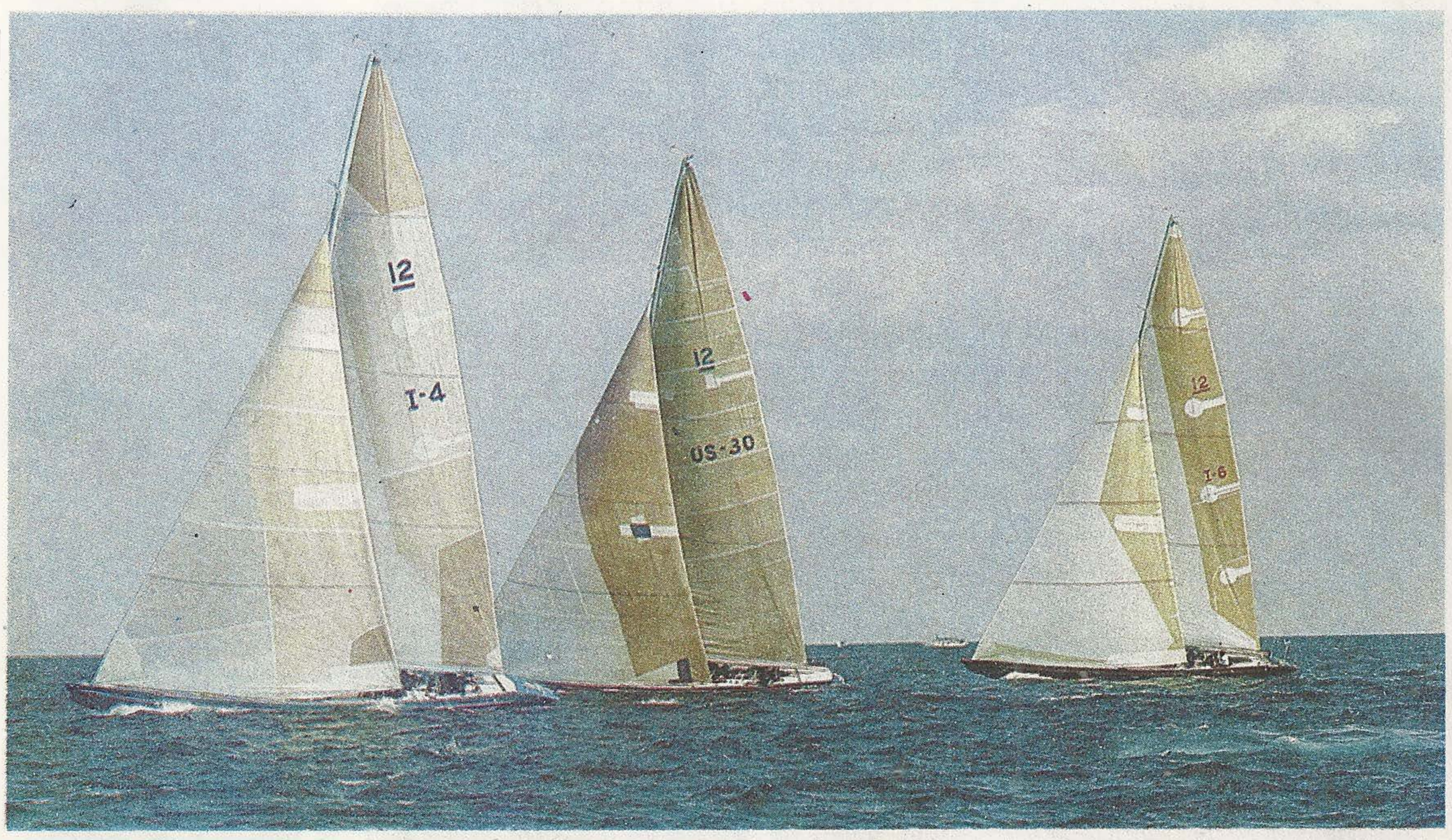
Önceleri yelkenlilerin yapımı ve tasarlanmasında yararlanılan yardımcı aletler ya da basit hesap aletleri olarak kullanılan bilgisayarlar, borda verilerini toplamak ve işlemek üzere su üstündeki denemeler ve yarışlarda ''bordaya çıktılar''. Çok daha incelikli yelkenlilerin yapımıyla yarışma yepyeni bir görünüm kazandı.

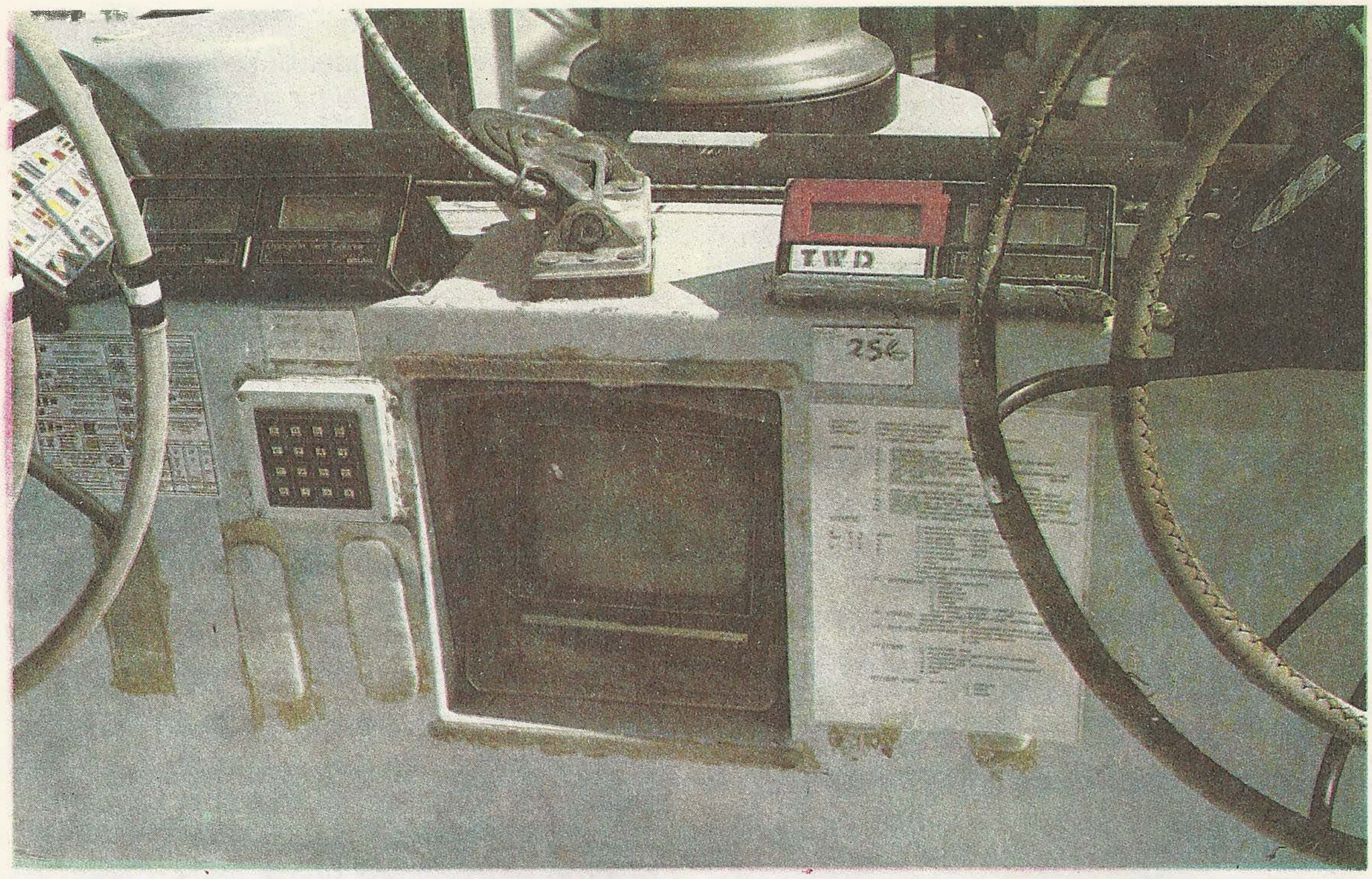
Bilgisayar America Kupasında ilk kez tam olarak 1963'de gerçek rüzgarın açısını hesaplayan bir aletin tekneye yerleştirilmesiyle kendini gösterdi. O zamanların aletleri ''görünen rüzgar'' (teknenin kalkış hızıyla oluşan rüzgarla birleşen gerçek rüzgar) üstüne bilgiler - özellikle rüzgarın yönü ve hızıyla ilgili bilgiler- sağlamaktan öteye gidemiyordu. Oysa gerçek rüzgara ilişkin veriler vektör hesapları sonucu elde edilmektedir. Bu nedenle, bilgisayarların gemicilik sorunlarını çözümleyen basit hesap aletleri olarak kullanılmasından, çeşitli sonuçlara yönelik parametrelerin belirlenmesi için verilerin toplandığı, işlendiği ve kontrol edildiği karmaşık sistemlerde yararlanılmasına geçildi.

Elektronik alanında gerçekleştirilen teknolojik gelişmeler -özellikle bilgisayarın küçülmesi, hesap gücü ve fiyatı- bilgisayarın 12 m.lik teknelerde vazgeçilmez borda aletleri arasında yerini almasını çok hızlandırdı.

1980'de Newport'da yapılan yarışmada bilgisayar bu kez ilk olarak karaya yerleştirilip radyoyla birbirine bağlanan güçlü hesap sistemlerinden oluşan bir dış donanım şeklinde görüldü. Antremanlar ve yarışlar sırasında bor-

Amerika kupasının çıkışı





Victory'nin kaptan köşkü. Ön planda dümen mahalli. Dümenler arasında, borda bilgisayarının ekranı, kumanda klavyesi ve verilerin görselleştirilmesini sağlayan sıvı kristalli göstergeler görülüyor.

da alıcıları tarafından elde edilen bütün dikkate değer veriler - yelkenlinin hızı ve yönü, rüzgarın hızı ve yönü, vb.- teknedekilerin hemen yararlanabilmesi için önce borda aletlerinin vizörlerinde gösterilip, ardından borda hesap makinesi tarafından alınarak işleniyor, son olarak da tamamı diske ya da manyetik şeride kaydediliyordu. Günün sonunda veriler karaya aktarılıyor, teknenin elde ettiği sonuçlar hakkında olabilecek en fazla bilgiyi edinmek için çözümlenmek üzere merkez bilgisayara sokuluyordu.

Yine 1980'de bir başka yenilik oldu: Yelkenlinin verilerinin karadaki işlem merkezine doğrudan aktarılmasını sağlayan telematiğin ortaya çıkışı. Defender Clipper'in (koruyucu anlamına gelen defender eldeki kupayı koruyacak 12 m.lik Amerikan teknelerine verilen addır) sahibi Newyork Yelken Klübünü verilerin en hızlı ve etkili biçimde işlenebilmesi için bu yeni yöntemden yararlandı.

Toplanan verilerden yararlanabilmek için artık teknenin üsse dönmesini beklemeye gerek kalmamıştı, bunlar teknede doğrudan işlenebiliyordu.

Veriler borda bilgisayarlarından karadaki merkeze sürekli olarak radyoyla iletiliyordu. Böylelikle hemen hemen anında işlenen veriler, tekneye geri gönderildiklerinde, hesap makinelerinde çözümlenmiş olarak kullanılabiliyorlardı.

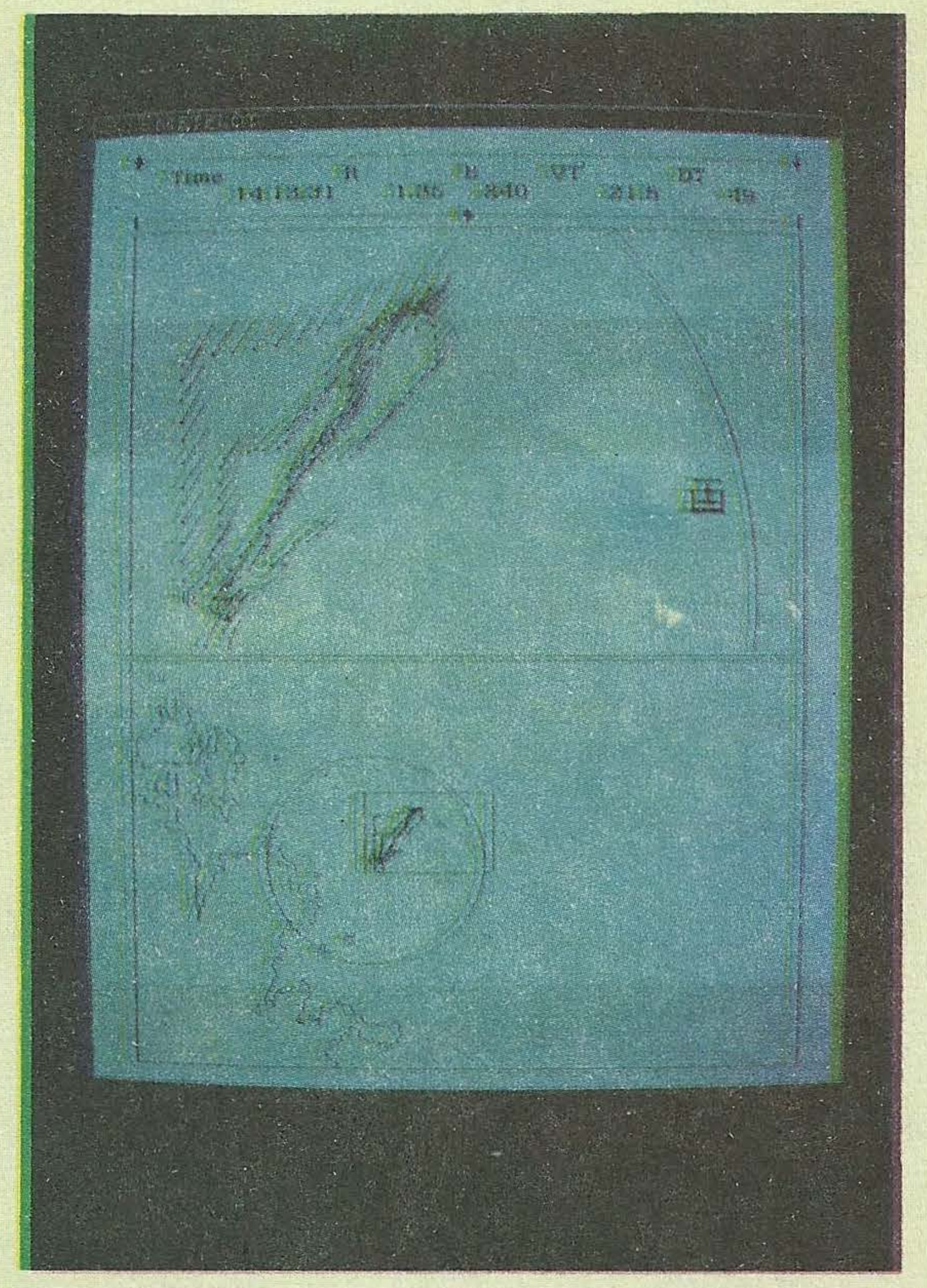
Özellikle yeni teknelerin hizmete sokulmasında, bu yeniliğin büyük yardımı oldu. Tayfanın yelkenliyle denizde elde edilebilecek sonuçları yarıştan önce taşıması gerekiyordu: Bilgisayar, öğrenme evresini en üst düzeyde hızlandıran araç ve en büyük yardımcıydı.

1983 yarışı, yalnız tarihsel ve teknik bakımdan bilgisayardan payını en çok alan American Defenderlerinde değil, yaklaşık 12 m.liklerde, borda bilgisayarlarına ve karada yönetilen işlem sistemlerine gitgide daha çok başvurulduğunu gösterdi. Ayrıca, bu yarışta hiç kullanılmamış olan en incelikli sistemlerden birinin de İngiliz teknesi Victorya 83'ün bordasına yerleştirildiğini söylemek gerek.

Bu örnekle 12 m.lik bir teknenin bordasında bilgisayarın nasıl kullanıldığını ayrıntısıyla gö-

Victory'nin bordasındaki bilgisayar verilerin alınması, işlenmesi ve belleğe alınması için yeterlidir.





Yarış alanının ve Victory'nin rotasının ekran üzerinde gösterimi. Yukarıda yarış alanının büyütülmüş şekli.

receğiz. Sistem birçok ögeden oluşur:

Verilerin çıkarılması, hazırlanması ve görselleştirilmesi sistemi (transdüktörler ve alıcılar): *Teknenin yönetimine ilişkin verileri ölçer (ör. rüzgarın hızı ve yönü, tenenin hızı ve yolu, dümen açısı, konum vb...)*

Buna, işlevi brüt verileri hazırlayıp basit hesaplar yapmak olan minik işlemci gibi yüzey birimlerini de eklemek gerekir. Sonuçlar sıvı kristalli ekranlarda (LCD) görülür. Dolayısıyla tayfa tarafından anında kullanılabilirler; ancak her durumda, işlem için merkezi birime de iletilmiş olurlar.

Verilerin alınıp işlendiği sistem: *Veriler arasındaki ilişki, alıcıların aktardığı verilerden yola çıkılarak yeni sonuçların üretilmesi, görselleştirme ve verilerle sonuçların denemeler ya da önceki yarışlar sırasında belleğe alınmış olanlarla karşılaştırılması gibi özel işlemleri yerine getiren bir borda bilgisayarından oluşur.*

Teknik olmaktan çok tarihsel nedenlerden ötürü borda sisteminin gücü o zamana kadar sınırlıydı. İstatistik analiz, fonksiyon üretimi, belleğe alma gibi işler daha çok karadaki bir sistem tarafından gerçekleştirliyordu. Bu durumda veriler teknede toplanmıyor, radyoyla sisteme aktarılıyordu. Ancak Victory'nin durumunda olduğu gibi güçlü bir bilgisayarın bordada kulanılması artık 12 m.likler dünyasında kabul edildi.

Sistemin çekirdeğinde bilgisayar dışında bir diskb irimi, video ekranı, bir grafik bilgiyazar ve bir klavye bulunur. Sistemin teknenin hizmetine açık bir veri bankası vardır: teknedeki aletler sayesinde tayfa bundan teknedeki başarı gücünü etkin biçimde kontrol etmekte yararlanır.

Borda alıcıları tarafında çıkarılan veriler sürekli olarak kaydedilip, gereğinde gerçek sürede işlenir. Böylece karadaki operatör de olayları ''yönetme'', örneğin radyoyla teknedekileri olası tersliklerden haberdar etme ve gerekli görüyorsa verileri kaydetme olanağına sahip olur. Ancak bu yalnızca denemelerde geçerlidir çünkü yarış sırasında tekneyle haberleşme izni yoktur.

Gelecek Amerika Kupası Avustralya'da Perth'de yapılacak. Burada yarışacak bir Avrupa ülkesini ele alalım. Örneğin İtalya 1987 yılı için şimdiden üç hazırlık yarışması açtı.

İlki Consorzio Azzurra'nın desteğiyle Costa Smeralda Yelken Klübü tarafından açılan Sfida Italian America's Cup. İkincisi Consorzio Italia ile birlikte bir konsorsiyum tarafından tasarlanıp hazırlanacak yeni 12 m.liğin ''yedek''i olarak Victory 83'ü satın alan İtalyan Yelken kulübününki (yedek- tekne öbür teknenin antremanında kullanılır). Kupa yönetmeliği teknenin ve donanımın temsil ettiği ülke tarafından tasarlanmış ve gerçekleştirilmiş olmasını şart koşuyor.

Sonuncusu ise Club Nautico Marina di Carrata'nın Consorzio Futura tarafından desteklenen yarışması. Büyük gizlilik altında tutulan yeni buluş ve projeler gün ışığına çıkacak artık. Hiç değilse, 12 m.liklere yerleştirilen borda sistemlerinin kapasitesinin artacağı ve yarış alanındaki meteorolojik araştırmaların (meteostat türü meteorolojik uydular yardımıyla rüzgar ve başka verilerin istatistik analizi) daha iyileşeceği kesin.

Ekran Çerçevelerinin Yönetiminde BASIC Komutları

Bazı makinelerde bulunan standart BASIC, ekran çerçevelerinin yönetiminde gerekli olan komutları sağlar. Bu yönetim iki aşamadan oluşur:

- Boyutlarını ve konumunu belirleyerek çerçeveyi oluşturma
- Çerçeveye verilen sembolik adın çağrılmasıyla grafiğin ya da metnin adreslenmesi

Çerçeve önceki WINDOW komutuna benzer komutlarla tanımlanır. Komutun çözülmesinde ve işleme sokulmasında gerekli olan bütün alt-programlar dili bütünleyen parçalardır ve kullanıcı bunları ayırdedemez.

Örneğimizde kod çözme alt-programı BASIC'le yazılmıştır ve yine BASIC'le başka alt-programlara çağrı yapar. Bunlar işleme sokulmadan önce çevirici tarafından tercüme edilir. Gerekli komutlar dilin bir parçası olduğunda çözme işlemi, komutlara doğrudan doğruya son şekli verir. Buna karşılık bu uygulamayı kullanıcının kendisi yaptığında işlem süresi daha uzun olur (programın derlenmiş olmadığı durumda). Şimdi Olivetti M20'de çerçevelerin yönetimi komutlarının sözdizimini göreceğiz.

Bazı makinelerde video ekranı iki görselleştirme biçimi sunar: sistemin belirme anında seçilecek olan 512 x 256 ekran noktası ya da 480 x 256 ekran noktası. Ekranın noktalarından her biri (pixel), kaynağı aşağıda solda bulunan taban koordinatları yardımıyla adlandırılır. Bu referans sistemi kullanıcı tarafından belirlenebilir. Bu sonuncu durumda bir noktanın koordinatları makine düzeyinde beirlenmiş ekran noktalarından hiçbirine denk düşmüyor olabilir (her pixel'in konumu makine düzeyinde belirlenir). O zaman sistem belirtilen koordinatlara en yakın ekran noktasını harekete geçirir. Bu işleyiş biçimi ekran noktası referanslarına başvurmayı gerektirmez, ancak grafik imgelerin gösterilmesinde bazen önemli belirsizliklere yol açabilir. Bir çerçeveyi;

A = WINDOW (q,p,v,h)

biçiminde tanımlayan komut şu ögelerden oluşur:

A Çerçeveyi gösteren (tamsayı) değerdir. Sistem 2 ilâ 16 arasında bir değer verebilir: 16 aynı anda açılabilecek en fazla çerçeve sayısı, 1 ise ekranın bütünüdür. Artan numaralandırma ilk değerden başlar ve olası kapatmalarla boş kalan çerçeveler de yeniden kullanılarak, çerçeveler çoğaldıkça yükselir. Örneğin 2,3 ve 4.çerçeveler oluşturulmuşsa, öncekilerden biri silinmediği (kapatılmadığı) takdirde, sonraki 5 olur. Yoksa sonraki 5'i değil serbest kalan değeri alır.

q ''Ana'' çerçeve içindeki oluşturma sırasında çerçevenin aldığı konumu gösterir. Çerçeve ancak ekranın bütününe göre konumlandırılabildiğine göre, bu fonksiyon önceki örnektekinden farklıdır. Bu makinede, tersine çerçeve bir ekran olarak kabul edilir (ekranın bütünü çerçeve 1'dir). Parametreye verilen değerler şunlardır:

0 yeni çerçevenin ana çerçevenin üst bölümünde konum alması
1 alt bölümde konum alma
2 solda konum alma
3 sağda konum alma

p Yeni çerçevenin yaratılması için ana çerçevenin (ya da ekranın bütününün) alt-bölümlenme konumunu gösterir.

p Parametresine verilecek değer q'nunkine göre belirlenir. Eğer q, 0 ya da 1 değerindeyse (ana çerçevenin alt ya da üst bölümündeki çerçeve), çerçeve yaratabilmek için p'nin değeri 1'le 255 arasında değişir, yani yeni çerçevenin taranacak satırlarının sayısını gösterir.(*) Aşağıda bir alt-bölünme örneği verilmiştir. q parametresi (kadran) 2 ya da 3 değerindeyse bölünme dikey olur. O zaman p parametresi (konum) 1'le, ana çerçevenin uzunluğu eksi 1 arasında değişir.

v Her metnin satırındaki taranacak satırları gösteren opsiyonel parametredir. 1'le 16 arasında değişir. Gıyabi değeri (per default) ana çerçevenin değerinin aynıdır.

h Metnin iki karakteri arasındaki uzaklığı ekran noktası olarak göstermeye yarar. 6 ya da 8 değerini alabilir. Birinci durumda her satır, seksen, ikinci durumdaysa altmışdört karakter içerir. Opsiyonel parametredir. Gıyabi değeri ana çerçe-

(*) ''Tarama satırı'' terimi grafik ekranın bir satırını belirtir. Metin biçimine ilişkin olan ''sıra'' ya da ''satır'' terimleriyle karıştırılmaması gerekir. Bu durumda bir satır ötekilerden en az bir karakterin yüksekliği kadar ayrıdır ve genellikle 7'den 17'ye birçok tarama satırından oluşur.

venin değeriyle aynıdır.

Yandaki sayfa v ve h'deki değişikliklere bağlı olarak WINDOW komutunun işlevinde meydana gelen değişiklikleri gösteriyor. Öbür parametreler hesaba katılmamıştır, bu da sistemin ana çerçevenin değerlerini esas aldığı anlamına gelir. v ve h parametreleri aynı WINDOW komutuyla (çerçeve açılmışsa bile) her an değiştirilebilir. Bu durumda;

```
A = WINDOW(O,O,v,h)
```

biçimini alır.

Demek A saymaca bir değerdir, çünkü sistem buna kendiliğinden, seçilmiş olan çerçevenin değerini verir. Çerçevenin seçimi WINDOW komutunu yeniden

```
WINDOW %N
```

biçiminde ele alır. Seçilen çerçeve tamsayı N değişkeninin sayısal değerine denk düşer.

Örneğin:

```
N = 2
WINDOW %N
```

komutları 2 numaralı çerçevelerin seçimine yol açar. Sayısal değer doğrudan doğruya elde edilebilir; o zaman komutlar; WINDOW %2'de toplanır.

WINDOW artık gereksiz olan çerçeveyi kapatıp bu alanı başka uygulamalara da açar. O zaman komut;

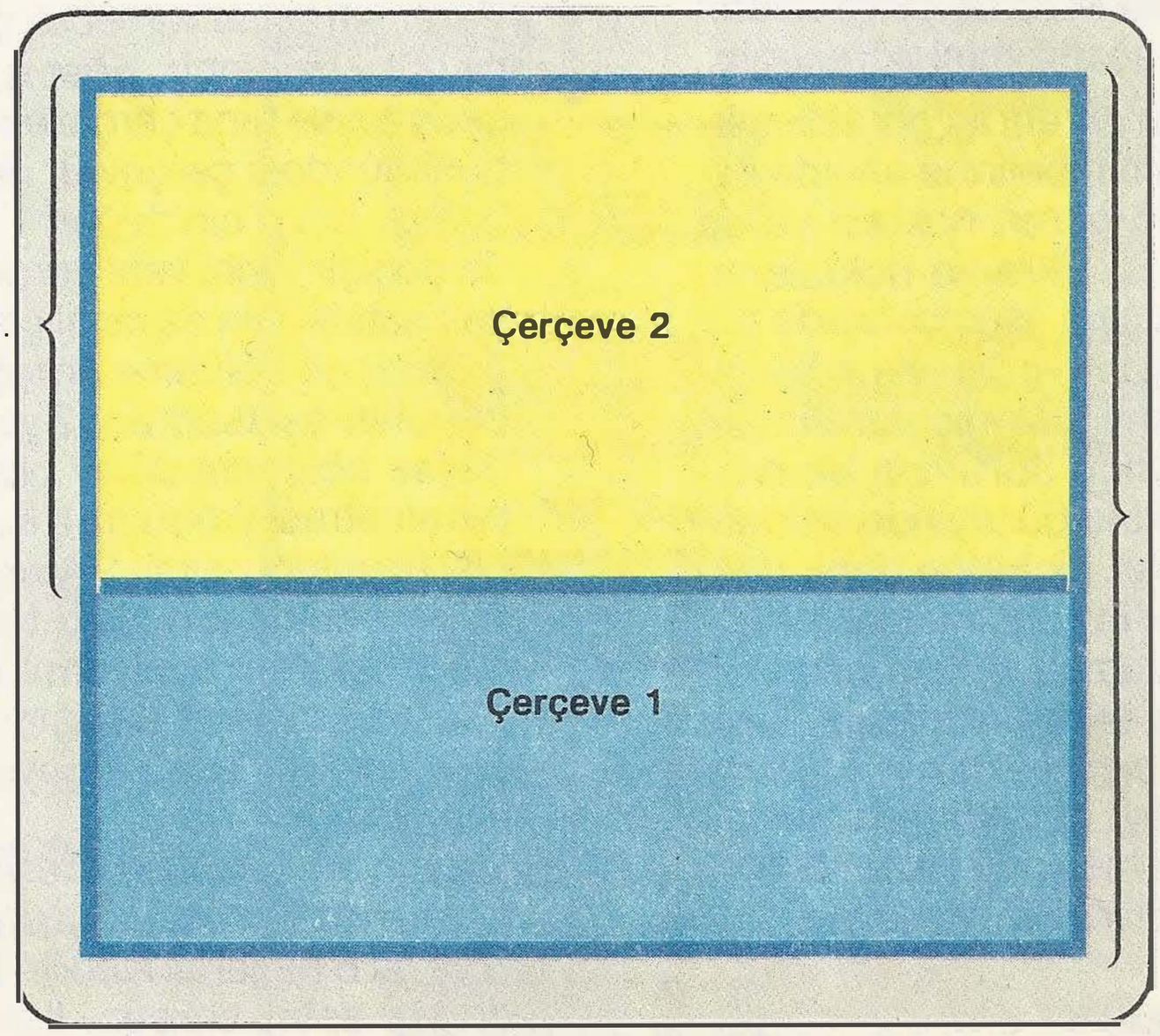
```
CLOSE WINDOW %A
```

biçimini alır. A silinecek çerçeveyi gösterir. A hesaba katılmadığında ekranda gösterilen bütün çerçeveler kapanır.

Grafik Vektörler ve Figür Tabloları

Her resim bir dizi temel doğrusal çizgiye ayrı-

EKRANIN P VE Q PARAMETRELERİ DOĞRULTUSUNDA BÖLÜNMESİNİN ÖRNEĞİ

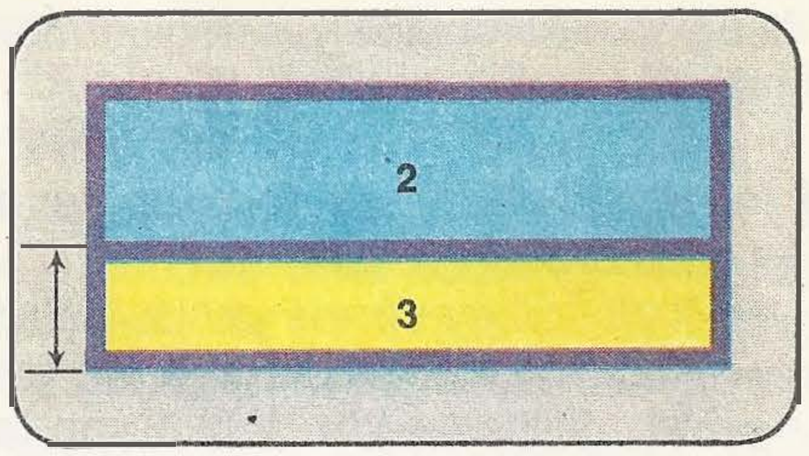


Bir çerçeve (ör.no 2) aşağıdaki değerlere göre açılmışsa

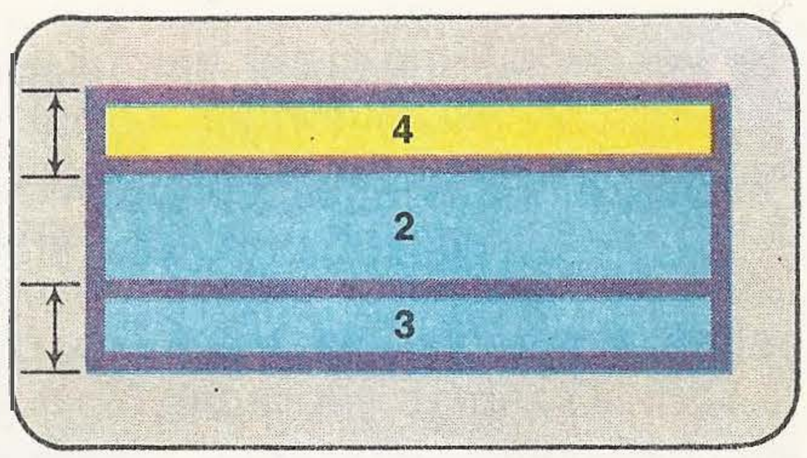
q = 0 çerçeve ana çerçevenin (burada bütün ekran) üst bölümünde yeralır ve bölünme yataydır.

p = 180 çerçevenin yüksekliği üst sınırdan itibaren 180 satırdır.

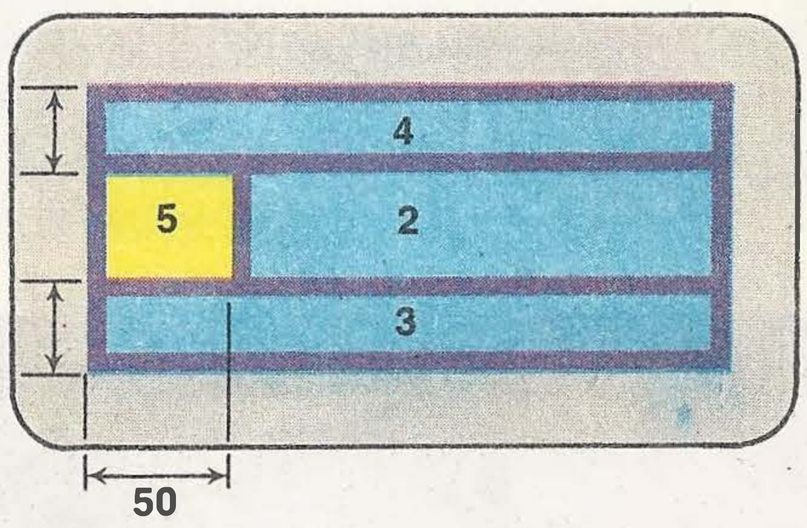
WINDOW KOMUTU UYGULAMA ÖRNEĞİ



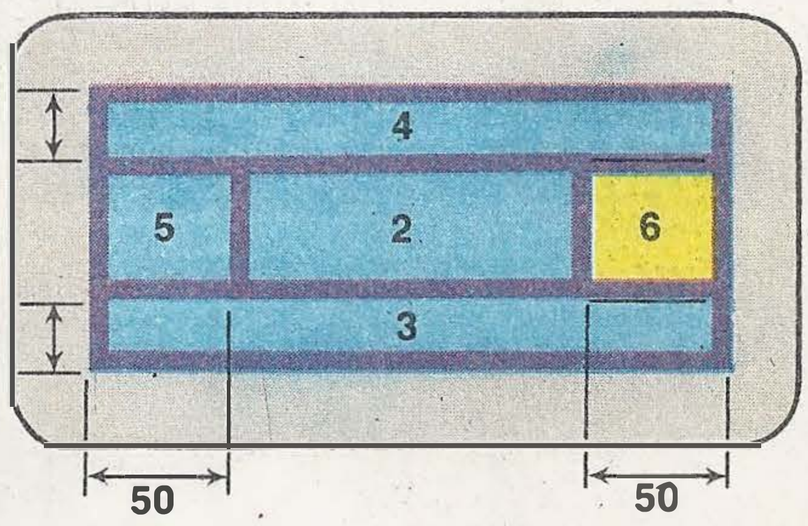
WINDOW (1,60) alt-bölümün 160 tarama satırı oluşturacak şekilde ayrılması (q = 1)



WINDOW (0, 100) üst bölümün 100 satır oluşturacak şekilde ayrılması



WINDOW (2, 50) Sol bölümde bölünme. Bu durumda p parametresinin değeri (50) çerçevenin genişliğini gösterir.



WINDOW (3,50). Ana çerçevenin sağında (q = 3) gerçekleşen benzer bir bölünme.

şır. Bu teknik bazı sistemlerde figürü ortaya çıkaran iğnenin temel hareketlerinin bir bütün halinde belirlenmesine olanak verir. Bütünler genellikle sayı ya da harflerin oluşturulmasına yarayan yer değiştirmelerin yansıdığı figür dizeyleriyle (tablolarıyla) gösterilir.

Buna dayalı sistemlerde tablolar belli bir değer tablosu kullanılarak yaratılır. Bu ise ölçek

ya da rotasyon değişmelerine ilişkin komutları içerir.

Aynı fonksiyonları bu komutların başlangıçta öngörülmediği makinelerde de gerçekleştirerek çok basit bir mantık yaratılabilir.

Yer Değiştirme Vektörlerinin Kodlanması

Bir figür tablosu hazırlamak için, resmin çiziminde gereken bütün yer değiştirmeleri seçilen ölçekte belleğe almak yeter. 4 yer değiştirme türü vardır (yukarı, aşağı, sağa, sola) Eğimli çizgiler ya da eğriler, silikleşen kesikli çizgilerle gösterilir.

Tablodaki figür gerçek boyutlarında değil, en kısa kenar boyunca yapılan yer değiştirmeye dayalı bir sisteme göre belleğe alınır. Böylece, yüksekliği tabanın iki katı olan bir dikdörtgeni belleğe almak için aşağıdaki tabloyu izlemek gerekir (makinemizde eksenlerin başlangıç noktası ekranın yukarısında solda bulunur):

1 / Yatay yer değiştirme (taban)
2 / İki dikey yer değiştirme (yükseklik = 2 × taban)
3 / Yatay yer değiştirme (öncekinin tersi yönde)
4 / İki dikey yer değiştirme (öncekilerin tersi yönde)

Aşağıdaki semboller yer değiştirmeleri göstermek için kullanılırsa:

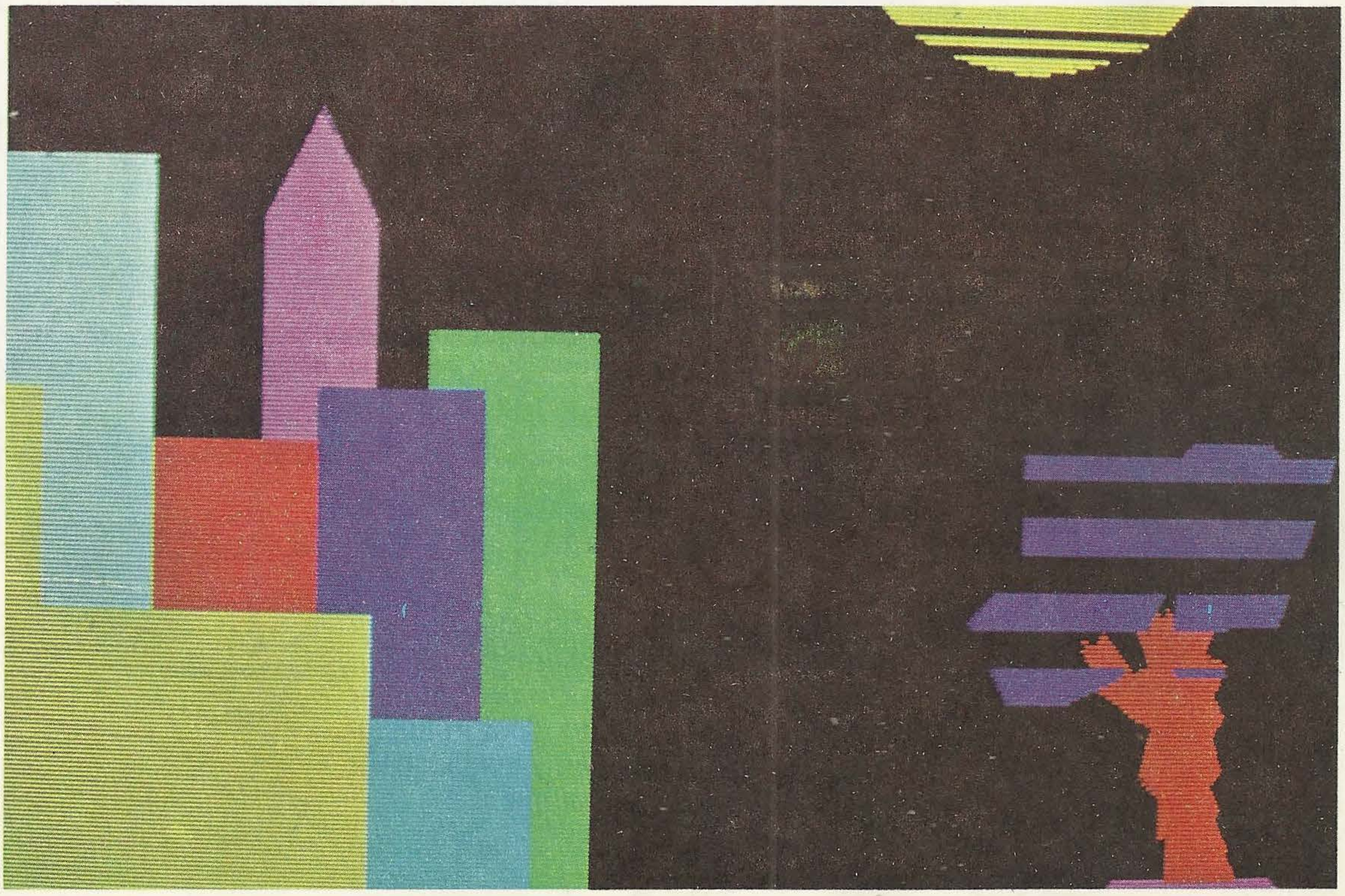
↑ = Negatif dikey yönde birim yer değiştirme (yukarı)
↓ = Pozitif dikey yönde yer değiştirme (aşağı)
→ = Pozitif yatay yer değiştirme (sağa)
← = Negatif yatay yer değiştirme (sola)

Bir dikdörtgen, sembollerin şu dizilişleriyle tanımlanır:

→ ↑ ↑ ← ↓ ↓

Her yer değiştirme **grafik vektör** adını alır: Bu durumda önceki dikdörtgen altı vektörden oluşacaktır. Resmin yeniden oluşturulması sırasında bunlardan her biri, bir kenarı çizmek üzere yapılacak yer değiştirmelerin yönünü belirtir; yer değiştirme eşiğini ise ölçek faktörü verir. O zaman yapılacak ölçek değişiklikleri çok basittir; aynı çizimi, nicelikleri yeni ölçeğe göre artırarak izlemek yeter. Her figür dört temel yer değiştirmenin bir türlü biraraya gelişi olarak şematize edilirse, belleğe almada kazanılan yer önemli bir miktar tutar. Her şekilde mümkün seçmeler 4'le sınırlıdır (↑, →, ←, ↓) ve iki bitle gösterilir. Örneğin yan

Grafik modda resmedilmiş bir şehir görüntüsü.



sayfadaki tabloda gösterildiği gibi.

Gösterilen kodlama bu tekniği kullanan makinelerde (Apple ve uyarlı olanlar) en çok raslanandır. Bütün bu durumlarda, vektörler hep iki bitle gösterilebilir.

Tablo 1, grafik vektörlerle yapılabilecek bir kodlamayı gösterir ama kodların niteliklerini belirlemez. Vektörle gösterilen yer değiştirmenin yalnızca bir konumlandırma mı (ekranda hiç bir şey gözükmeksizin) yoksa bir doğru parçasının görselleştirilmesi mi olduğunu da belirtmek gerekir. Birinci durumda vektörün tanımladığı yer değiştirmenin arka planla aynı renkte çizilmesi gereklidir; ikinci de ise (görselleştirme) renk görünmelidir. Demek ki, bu bilgi doğru parçasının görünür olup olmadığını belirten bir üçüncü bit gerektirir. Yalnızca yer değiştirme olduğunu göstermek için 0, çizginin görünür olduğunu belirtmek için 1 değeri verilirse, sayfanın altındaki tablo 2'de sunulan denklikler elde edilir. Görüldüğü gibi, görünmez bir yer değiştirmeden, aynısının görünenine geçmek için, ilk değere 4 eklemek yeter. Örneğin pozitif yatay yer değiştirmenin (→) ondalık kodu görünmez olduğunda 1, tersi durumda 5'tir. Kodu böyle çözdüğümüzde önceki dikdörtgen 5 (→), 4 (↑), 4 (↑), 7 (←), 6 (↓), 6 (↓), ondalık değerleri ve bunlara denk düşen ikili değerler;

101 (5, →)
100 (4, ↑)
100 (4, ↑)
111 (7, ←)
110 (6, ↓)
110 (6, ↓)

ile gösterilir. Bu tablonun belleğe alınması, biri yer değiştirme için olmak üzere 6 bellek alanı gerektirir. Bellek alanı boş yere doldurulmuş olur çünkü her vektör için 8 bit ayrılmışken, bir vektör üç tanesini kullanır. Bu durumda bir çok vektörü aynı adreste belleğe almaya olanak veren bir yoğunlaştırma biçiminin aranması olağandır.

En geçerlisi (bu sayfanın ilk şekli) bir bellek adresinin 8 bitinden, 6'sını 3,4 ve 5. bitleri birincisi, 0,1 ve 2. bitleri ikincisi olmak üzere birbirini izleyen 2 vektörle doldurmaktır. O zaman adresin 6 ve 7. bitleri hep 0'da, 5 ve 2.bitleri 1'dedir (bunları görünen yer değiştirmeyi belirtirler). Makineye tablonun sona erdiğini bildirmek için, bütün bir adres 0'a getirilir (konum 4).

Figürün tablosu belleğe alındığında, çizimi başlatmak için spesifik bir alt-program çağrılır. Bu her adresin içeriğini çözümleyip 3,4 ve 0 bitleriyle gösterilen yönde 5 ve 2. bitlerin gösterdiği renkte bir doğru parçası çizer.

Belleğin içeriği 0'a geldiğinde (tablonun sonunu bildirmek için kararlaştırılan değer) döngü sona erer. 1593-1596. sayfalarda örnek programının akış şemaları ile bunun dökümünü göreceğiz. Bu örnekte DATA komutuyla girilen figür tablosu (dikdörtgen) sabittir, oysa gösterme evresinden (alt-program 1000) genel kullanımlarda da yararlanılabilir.

Program ancak niteliksel açıdan değerlidir. Aynı tabloda birçok figürün bulunmasına, oluşturma alt-programlarına ve figürlerin belleğe alınmasına olanak vermez.

Uygulamada, eldeki bir dizi sembolle karmaşık bir resim çizmek çok güçtür. En iyi yaklaşım resmi basit figürlere ayırıp, bunları tek bir figürler tablosunda belleğe almaktır. O zaman

Yer değiştirme	Ondalık değer	İkili değer
↑	0	0 0
→	1	0 1
↓	2	1 0
←	3	1 1

YER DEĞİŞTİRME VEKTÖRLERİNİN KODLANMASI

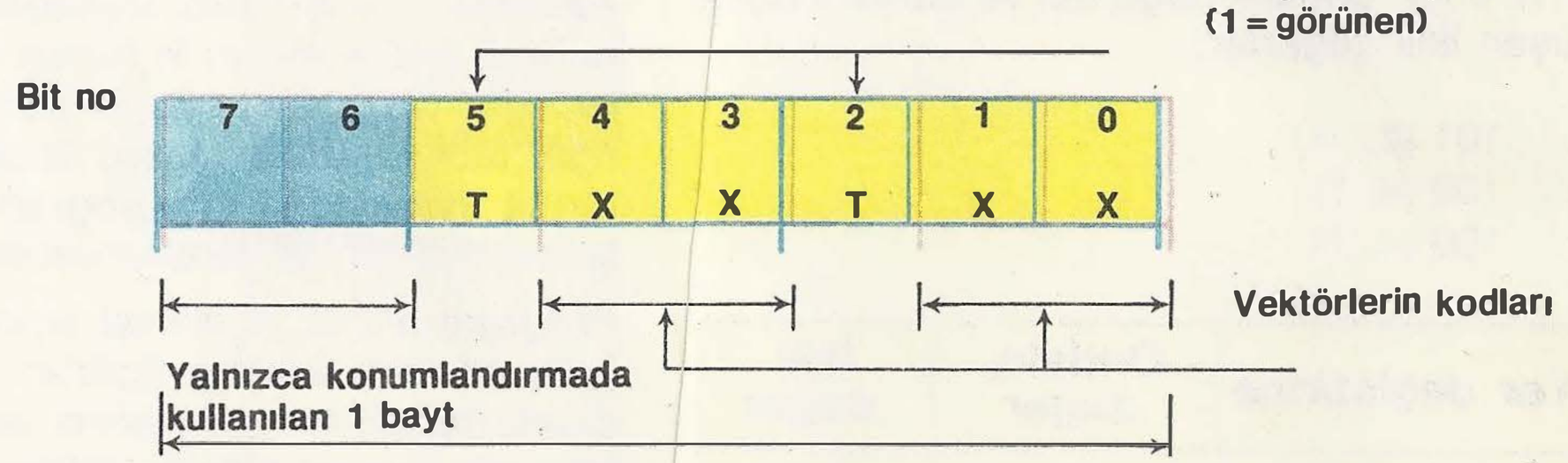
Yer değiştirme (vektör)	Görünen			Görünmeyen			Görünen ondalık değer	Görünmeyen ondalık değer
	2	1	0	2	1	0		
↑	1	0	0	0	0	0	4	0
→	1	0	1	0	0	1	5	1
↓	1	1	0	0	1	0	6	2
←	1	1	1	0	1	1	7	3

tablonun daha bütünlüklü bir yapısı olur.

1597. sayfadaki şekil böylesi bir mantıkla çalışan makinelerde kullanılan tipik yapıyı gösterir. İlk adreste (bayt 0) tabloda bulunan figürlerin sayısı yer alır, ikincisi hep boştur. Ötekiler ikiye ayrılır:

Adres alanı: Her figürün başlangıcını belirten ve tablonun repertuarını oluşturan noktaları içerir. Tablonun figürlerinden herbirini temsil eden grafik **vektörler alanı:** Her figürün sonu, içeriği 0 olan bir bitle belirtilir. Sistem bayt 0'ın (ilk adres) içeriğini inceleyerek varolan figürlerin sayısını öğrenir (örnekte 3). İkinci bayt hep sıfırda olduğundan atlanmalıdır. 3 farklı figürün adresi, adres 2'den çıkarılarak, 2. adımda bulunur. Başka bir deyişle 2. 3. adresler birinci, 4. ve 5. adresler ikinci, 6. ve 7. adresler üçüncü figürün başlangıcını belirleyen göstergeyi içerir: Resmin boyutlarının ve karmaşıklığının artması durumunda göstergelerin girebilmesi için bir adres genellikle 0'da bırakılır.

GRAFİK VEKTÖRLERE KARŞI GELEN KODLARIN YOĞUNLAŞTIRILMASI



ÜÇGENİN BELLEĞE ALINMASI

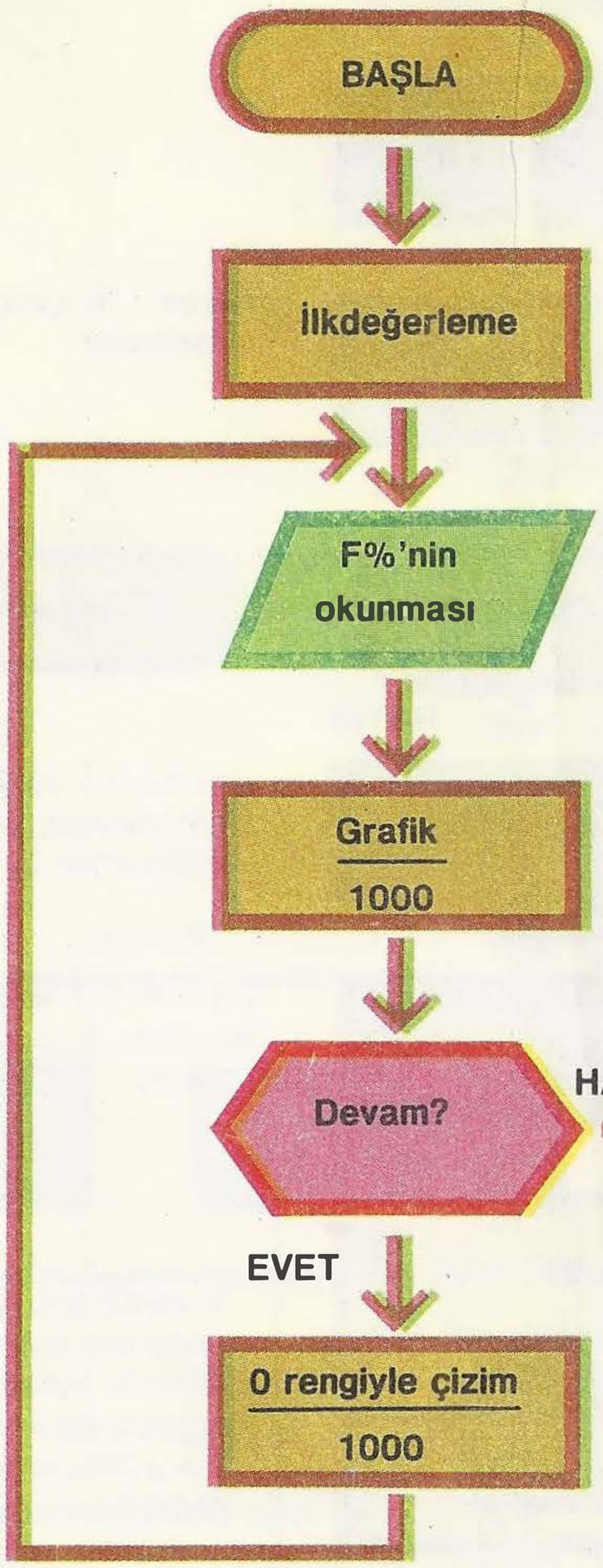
Bellek konumu	7	6	32 5 T	16 4	8 3	4 2 T	2 1	1 0	Ondalık değer
1	0	0	1	0	1	1	0	0	44
2	0	0	1	0	0	1	1	1	39
3	0	0	1	0	1	1	0	1	45
4	0	0	0	0	0	0	0	0	0

5(→) + 4(↑)

4(↑) + 7(←)

6(↓) + 6(↓)

BİR FİGÜR TABLOSUNU GÖSTERME AKIŞ ŞEMASI



Grafik tablosu A%(4) şu değerlerle ilkdeğerlenir.
A%(1) = 44
A%(2) = 39
A%(3) = 45
A%(4) = 0

F% ekran noktası olarak yer değiştirme vektörüne denk düşen doğru parçası

Ayrıntılı akış şemasının görselleştirilmesi

0 rengiyle (ya da kullanılan sistemde buna denk düşen başka bir renkle) resim çizmek, resmi silmek anlamına gelir.

Figür Tabloları Yönetim Programı

Bir figür tablosu yaratmak için iki yöntem vardır: İlki (ve en basiti), yapılacak yer değiştirmeleri doğrudan belirtecek olan kullanıcıyla diyalogu öngörür. İkincisi (programlama açısından daha karmaşık ama gerekli olan) 4 ana yer değiştirmeye 4 tuş ayırır, ayrıca yer değiştirmelerin herbiri harekete geçirildikçe görselleştirilir ve belleğe alınır. Figür tablosunun yapımı bakımından sonuç aynıdır ancak kullanılan resmin aşama aşama oluşturulmasını öngörür.

Bu fonksiyonların bulunmadığı bir sistemde kullanılabilecek bir figür tablosu oluşturma ve kullanma programını inceleyelim. Aynı program gerekli değişikliklerle yer değiştirme tablolarının yönetimini öngören sistemlerde de kullanılabilir. Bu tür makinelerde ölçek değiştirme ve figürün nakledilmesi gibi yan fonksiyonlar özel komutlarla elde edilir.

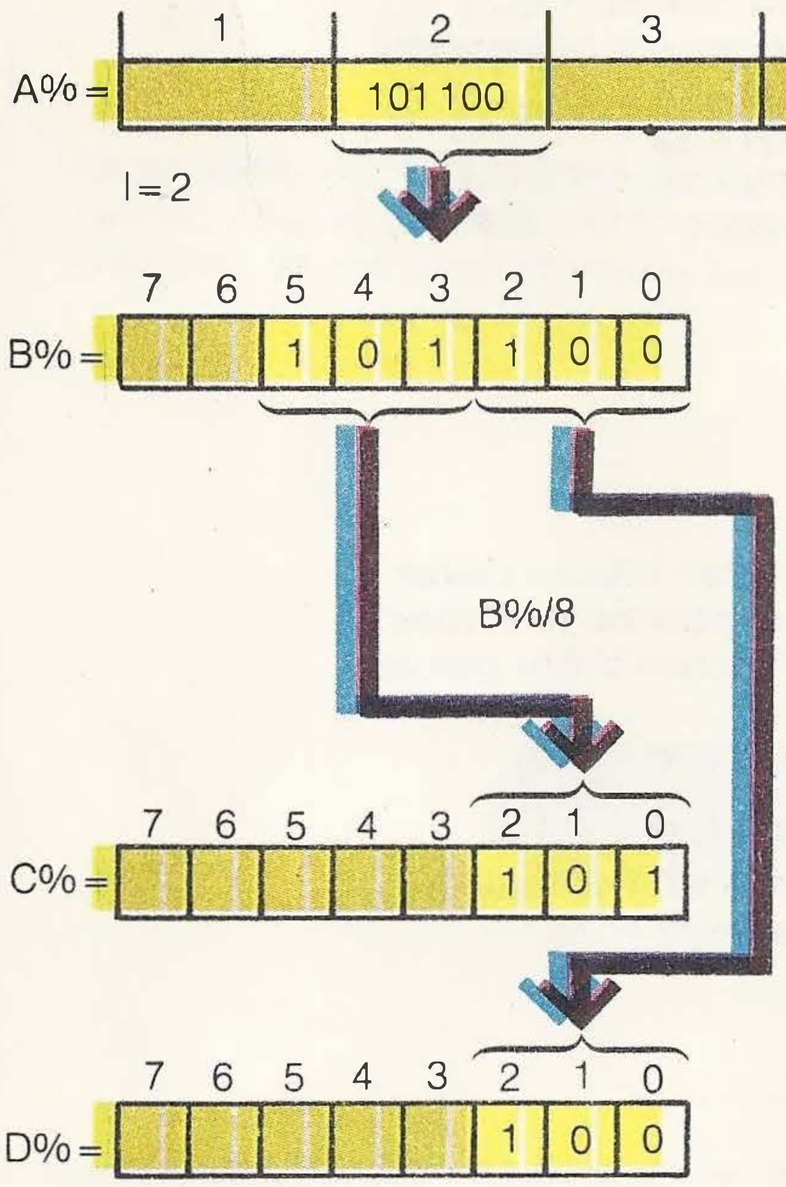
Bu programın yerine getirdiği başlıca fonksiyonlar şunlardır:

- Resmi oluşturan figürlerin kavranması, yer değiştirmeler halinde belleğe alınması ve görselleştirilmesi
- Sonradan değişiklik yapmak için yeniden okuma olanağı saklı tutulmak kaydıyla, değiştirme tablosunun diskete kaydedilmesi
- Ölçek değiştirme, her türlü figürü yerinden oynatma ve çevirme olanağı.

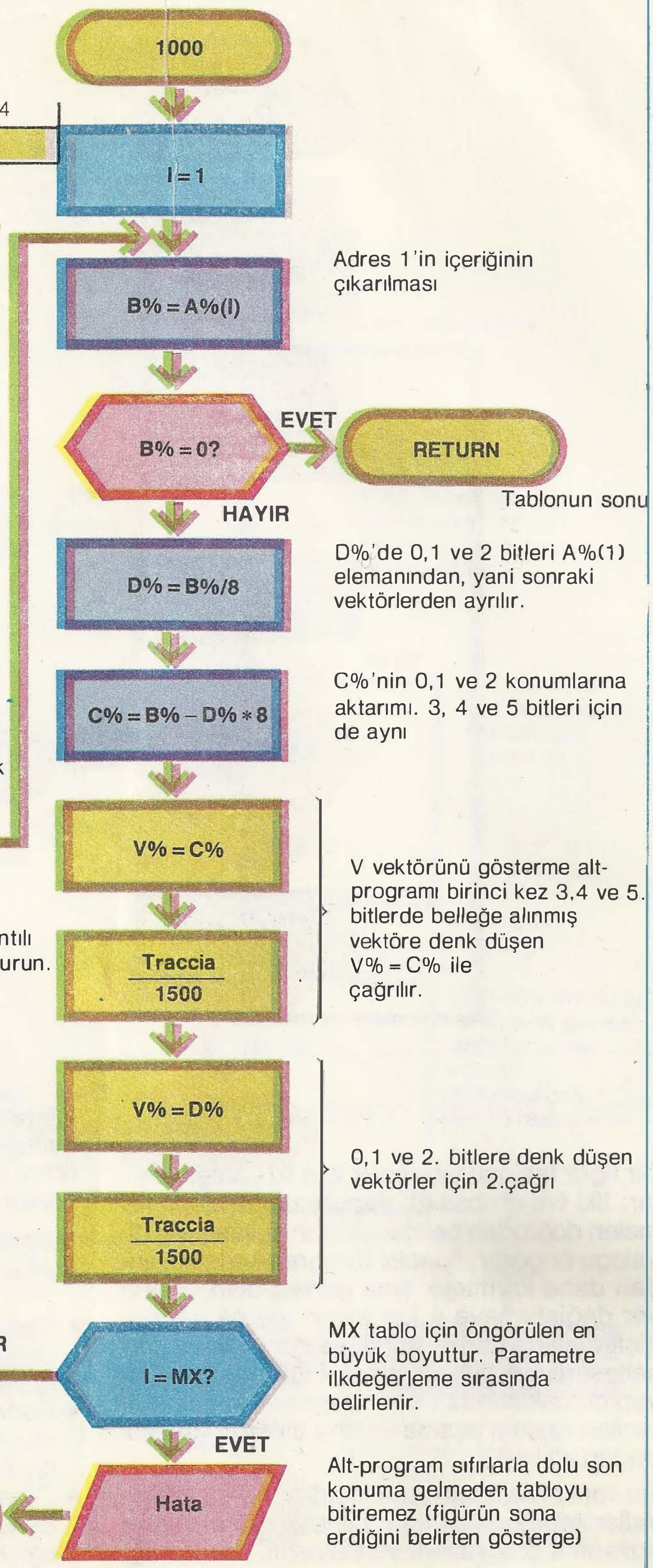
Değişkenleri sayfa 1599'daki tabloda sıralanan ana programın akış şeması (s.1598) 4 te-

GÖSTERME ALT-PROGRAMI

Örnek

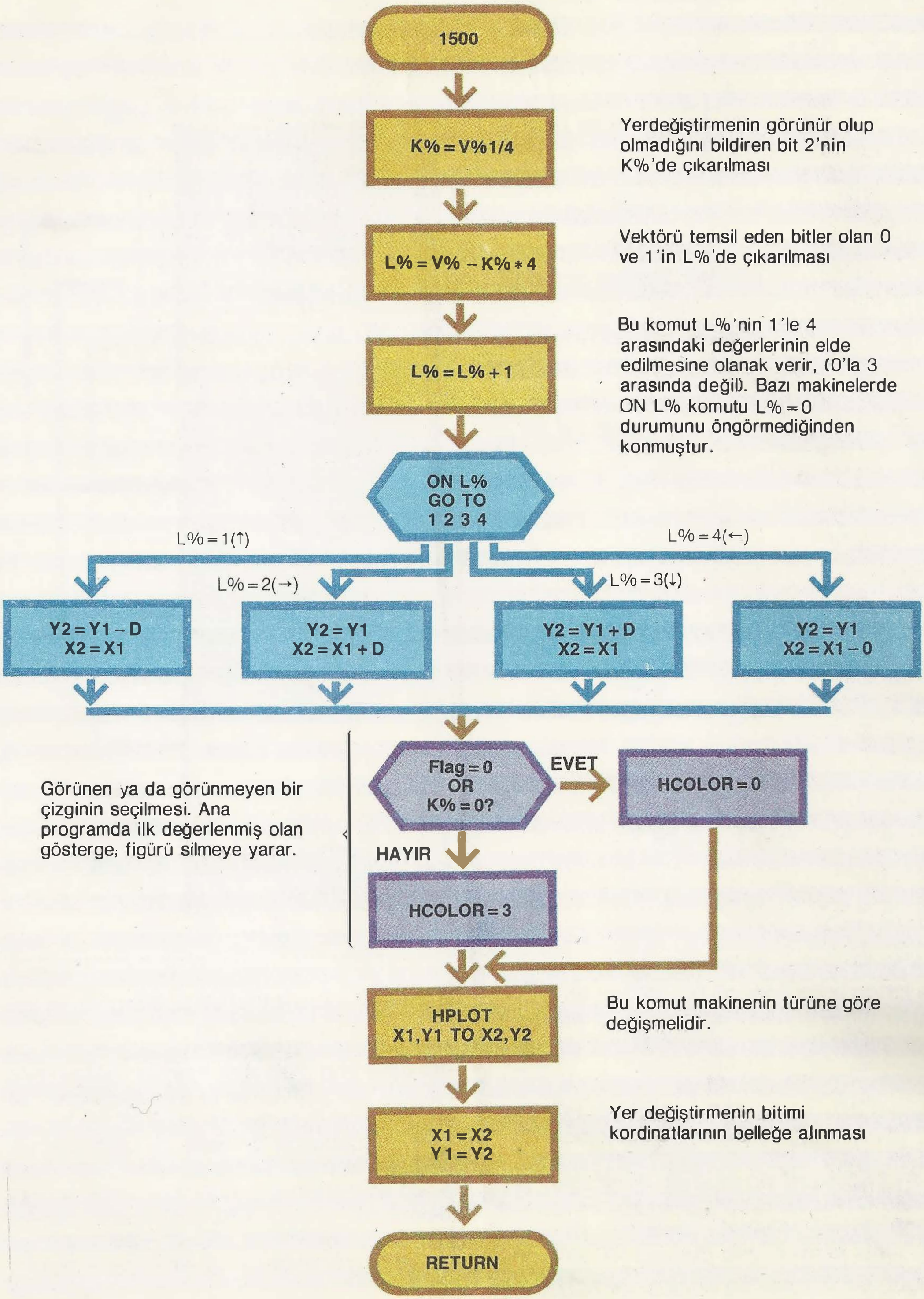


Tablonun doldurduğu bellek konumları döngüsü



Çizim 1500 için ayrıntılı akış şemasına başvurun.

ÇİZİM ALT-PROGRAMI



Yerdeğiştirmenin görünür olup olmadığını bildiren bit 2'nin K%'de çıkarılması

Vektörü temsil eden bitler olan 0 ve 1'in L%'de çıkarılması

Bu komut L%'nin 1'le 4 arasındaki değerlerinin elde edilmesine olanak verir, (0'la 3 arasında değil). Bazı makinelerde ON L% komutu L% = 0 durumunu öngörmediğinden konmuştur.

Görünen ya da görünmeyen bir çizginin seçilmesi. Ana programda ilk değerlenmiş olan gösterge, figürü silmeye yarar.

Bu komut makinenin türüne göre değişmelidir.

Yer değiştirmenin bitimi kordinatlarının belleğe alınması

FİGÜR TABLOSUNUN YÖNETİMİ

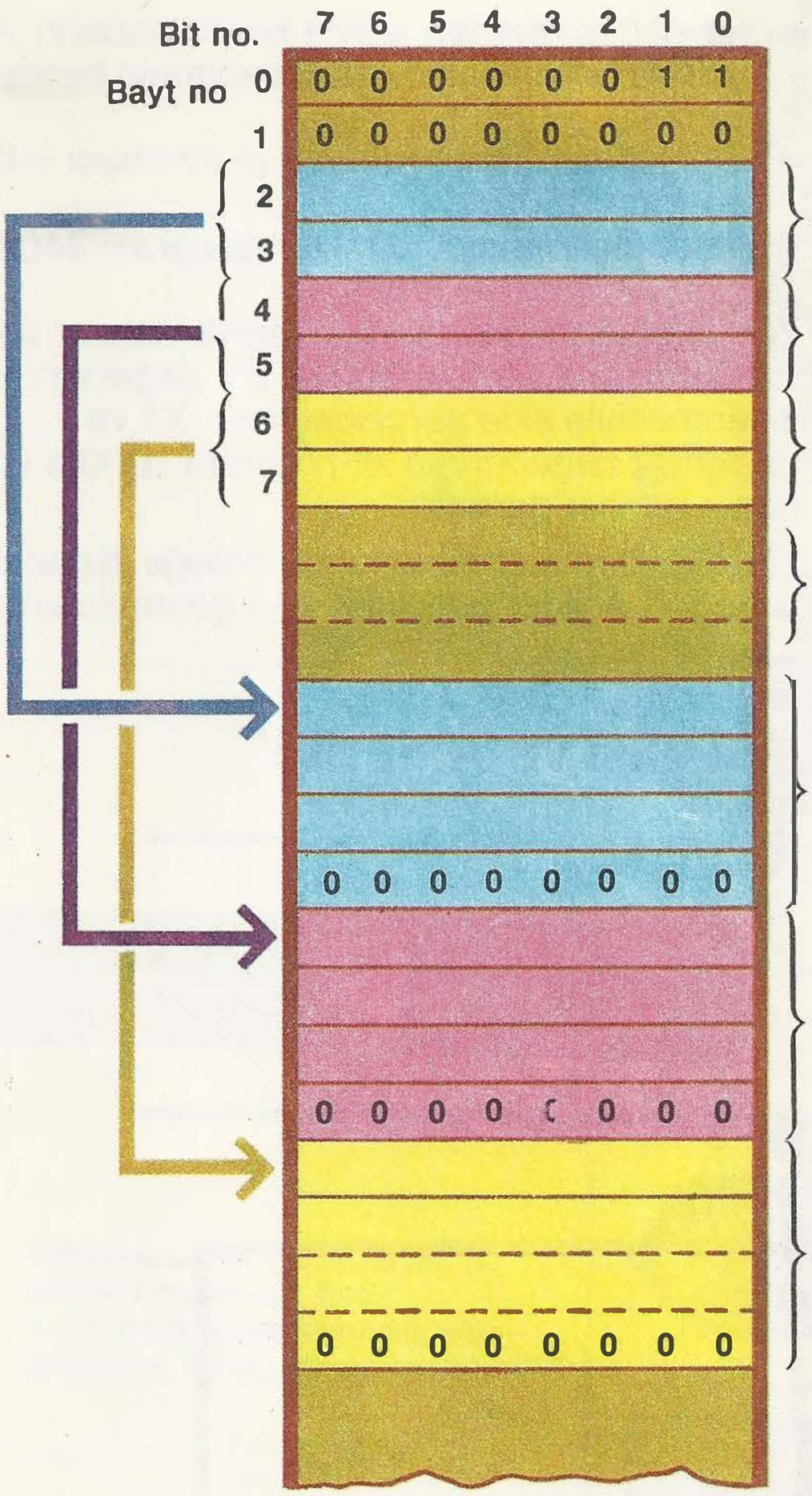
Siprel, Apple ve Bağdaşanlar için

```
1 REM ----------
2 REM * TABLO         *
3 REM * FIGURLER     *
4 REM **********
5 :
6 :
90 HGR
100 DIM A% (20):MX=20
105 :
106 REM
107 REM : VEKTORLERIN YUKLENMESI
108 :
110 A%(1) = 36:A% (2) = 55:A%(3) =
    16
112 REM ----------
115 :
120 X1 = 100:Y1 = 50:X2 = X1:Y2 =
    Y1
125 FL = 1
150 HOME :VTAB 22:INPUT "DOGRU
    PARCASININ UZUNLUGU :";I %
155 D = I%
160 GOSUB 1000
170 HOME: VTAB 22: INPUT "DEVAM
    ? ";A$
180 IF LEFT$ (A$,1) = "S" THEN
    200
190 TEXT : HOME : END
200 FL = 0:X1 = 100:Y1 = 50:X2 =
    X1:Y2 = Y1
210 GOSUB 1000
220 GOTO 125
999 :
1000 REM ==========
1001 REM * GRAFIKLER  *
1002 REM ==========
1003 :
1010 FOR I = 1 TO MX
1020 B% = A%(I)
1030 IF B% = 0 THEN I = MX:NEXT
     I: RETURN
1040 D% = B% / 8:C% = B%   D% * 8

1050 V% = C%:GOSUB 1500
1060 V% = D%:GOSUB 1500
1070 NEXT I
1080 HOME : VTAB 22: PRINT " HATA
     !":RETURN
1499 :
1500 REM ==========
1501 REM * CIZIM      *
1502 REM ==========
1503 :
1510 K% = V% / 4:L% = V%   K% * 4

1520 L% = L% + 1
1530 ON L% GOTO 1550, 1560, 1570, 1
     580
1550 Y2 = Y1   D:X2 = X1:GOTO 16
     00
1560 Y2 = Y1:X2 = X1 + D:GOTO 16
     00
1570 Y2 = Y1 + D:X2 = X1: GOTO 16
     00
1580 Y2 = Y1:X2 = X1   D
1600 IF FL = 0 OR K% = 0 THEN HCOLOR = 0:GOTO 1620
1610 HCOLOR = 3
1620 HPLOT X1,Y1 TO X2,Y2
1630 X1 = X2:Y1 = Y2: RETURN
```

FİGÜR TABLOSUNUN YAPISI



İlk adres varolan figürlerin sayısını içerir (burada 3 = 11 bit). Her zaman sıfır

1. Figürün başlangıç göstergesi

2. Figürün başlangıç göstergesi

3. Figürün başlangıç göstergesi

4. Gelecekteki uygulamalar için boş alan

1. figür

2. figür

3. figür

mel alt-program ya da fonksiyonel blok içerir.

İlki (1000) değişkenleri, DATA komutlarını ve göstergeleri sıfırlar. Ardından, ana programda gerçekleştirilen ve INPUT$(1) (CP/M altında) ya da GET (ya da benzeri) komutundan yararlanması gereken A$ karakterinin okunması gelir. Normal INPUT komutlarındakinin aksine, ekrana bir yankı göndermeyen bu komutlar, arada RETURN tuşuna basılmasını gerektirmeksizin (yani kaydın kapanmasını gerektirmeksizin) tek bir karakterin girilmesine izin verirler.

Bu biçimde girilen karakterler, sırasıyla K, TS ve TP'de içerilen üç değerle yanıt veren alt-program 2000'de çözümlenir. K'nın değeri basılmış tuşu gösterir.

1- Öngörülmemiş bir tuş
2- Yer değiştirme tuşu
3- Komutları girme tuşu

Yer değiştirme (figürün çizilmesi için) ya da komut (belleğe alma, ölçek değiştirme vb.) durumunda verilerin değerlerini TS (yer değiştirme) ve TP (komut) değişkenleri içerir.

TS değişkeni istenen yer değiştirme doğrultusunda (↑, →, ↓, ←), 1'den 4'e değer alır;

TP'nin değeri 1'le 9 arasındadır:

1 Bu komutu izleyen yer değiştirmeler görünmez.
2 Yer değiştirmelerin görselleştirilmesi
3 Figürün hemen sonra basılan tuşun gösterdiği yönde yer değiştirmesi
4 Figürün tuşun gösterdiği yöne dönmesi
5 Figürün belleğe alınması
6 Belleğe alınmış figürün çağrılması
7 Silme
8 Büyütme
9 İndirgeme

Alt-program 2000'in ayrıntılı akış şeması sayfa 1600'dedir.

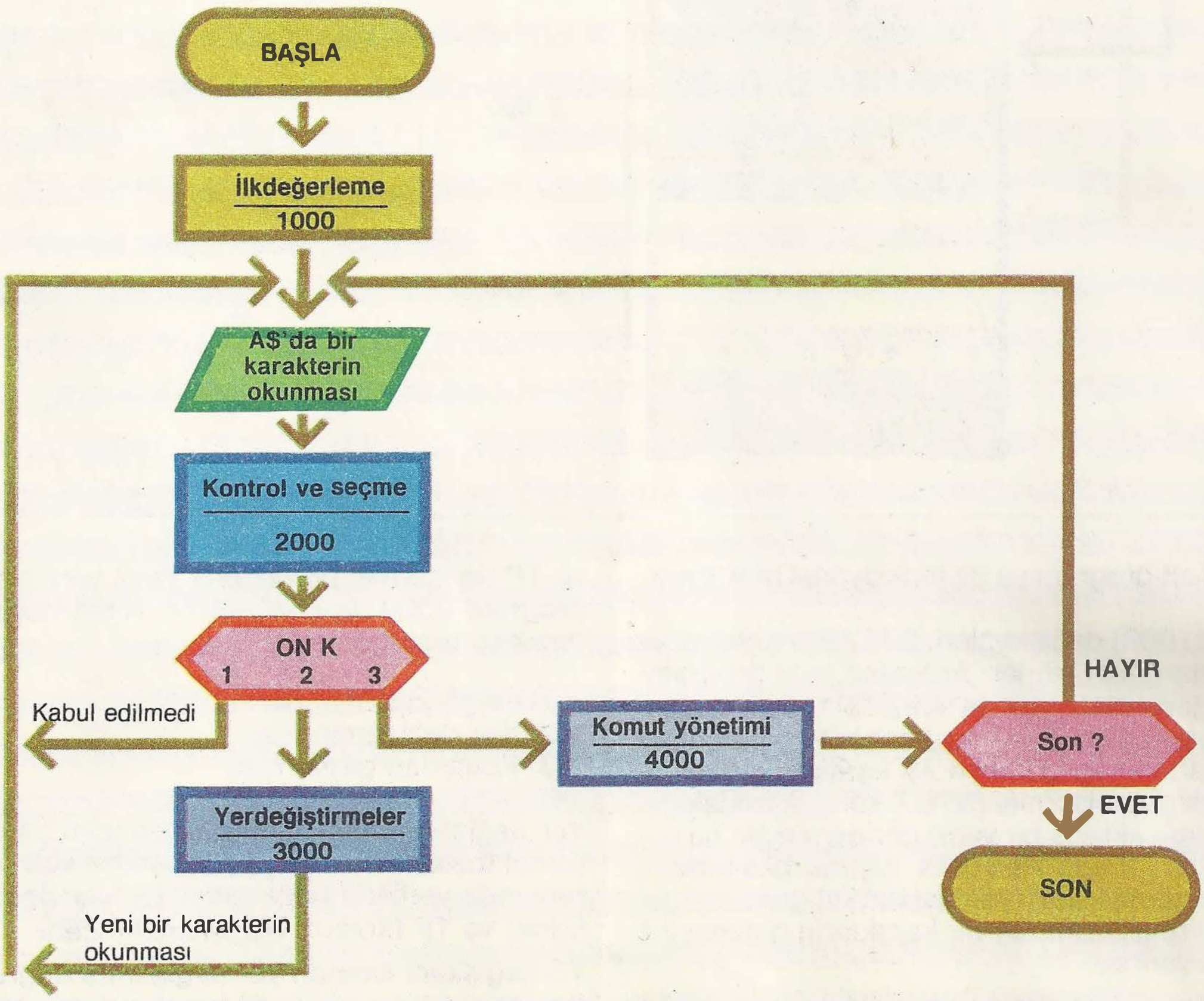
Giriş parametresi A$'in içerdiği karakterdir. Alt-program bunun yer değiştirmeler (tablo S'de) ya da komutlar (tablo 0'da) için öngörülen karakterlerden biri olup olmadığını bulur. Yanıt olumluysa, karakterin başlangıç konumunu, denk düşen değişkene (TS ya da TP) aktarır. Bu değer ayrıca gerçekleştirilecek yer değiştirme ya da komut türünü gösterir. Alt-program 3000 (sayfa 1601'deki akış şeması) yer değiştirmelerin yönetilmesiyle ilgilidir. Gerçekleştirilen işlevler şunlardır:

- Yer değiştirmeden sonra koordinatların ve tabloda belleğe alınacak V değerinin hesaplanması
- Yer değiştirmenin tabloya aktarılması (alt-program 3500)
- Yer değiştirmenin çizimi (alt-program 3400)

Yer değiştirmeden sonra koordinatların hesaplanması X1,Y1'e, DX ve DY değerlerinin eklenmesiyle elde gerçekleştirilir. X1 ve Y1'in başlangıç değerlerinin alt-program 1000'e verilmiş olması gerekir.

Yer değiştirmeler tablosunda belleğe alınacak V değeri açıklık getirmek için gösterilmiştir.

FİGÜR TABLOLARININ OLUŞTURULMASI VE YÖNETİMİ



PROGRAMLAMADA KULLANILAN DEĞİŞKENLER VE SEMBOLLER

A$ = Okunan karakter. Bir karakteri ya da komutu gösterir.
A1$ = Komutun gerçekleştirilmesinde ikinci karakter
NM$ = Belleğe alınmış kaydın adı ya da figürün yeniden çağrılması
K = Harekete geçirilmiş komut türü göstergesi (geçersiz, yerdeğiştirme, komut
TS = İstenen yerdeğiştirme türü
TP = Komut türü
S(10) = Yerdeğiştirme tuşlarının ASCII kodlarını içeren tablo
O(10) = Komut tuşlarının ASCII kodlarını içeren tablo
X1,Y1 = Yerdeğiştirmeden önce koordinatlar
X2,Y2 = Yerdeğiştirmeden sonra koordinatlar
DX,DY = Birim yerdeğiştirme değerleri (ekran noktası olarak)
HC = Görünen ya da görünmeyen yerdeğiştirme göstergesi. TP = 1 ya da 2 tarafından harekete geçirilir.
B(100) = Figürü oluşturan yerdeğiştirmelerin tablosu
IN = B(100)'de doldurulan son konum göstergesi
X0,Y0 = Çizimin başlangıç koordinatları
FL = Yerdeğiştirme göstergesi
FR = Rotasyon göstergesi
FS = Ölçek göstergesi

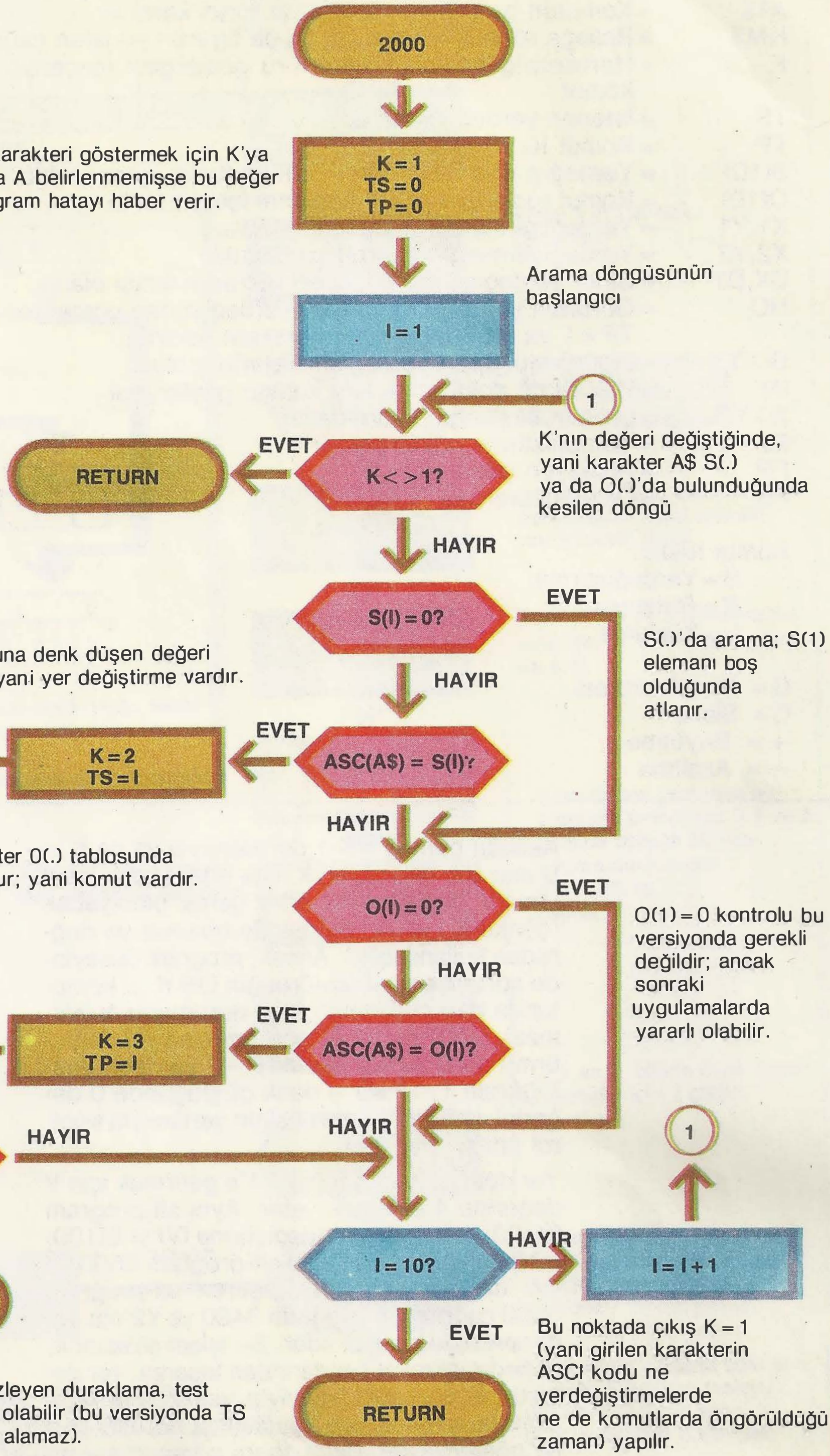
Komut tuşları
S = Yerdeğiştirme
R = Rotasyon
L = Yükleme

D = Disk (koruma)
C = Silme
+ = Büyütme
— = Azaltma

Aslında bunu V = K-1 denklemiyle ya da K'yı 0'dan 3'e değiştirerek elde edebilirdik. Yani yeni V değişkenine artık gerek olmayacak (çünkü K bellek bölgesinde bulunur ve doğrudan kullanılabilir). Ancak, program düzeyinde sorunlar çıkabilir (örneğin ON K ... komutunda bazı makineler K = 0 durumunu öngörmez). Görünen ya da görünmeyen yer değiştirme durumu, karakter TP = 2'ye denk düştüğünde 1, TP = 1'e denk düştüğünde 0 değerini alan HC göstergesinin yardımıyla kontrol edilir.

Yer değiştirmenin 3.bitini 1'e getirmek için V değerine 4 eklemek yeter. Aynı alt-program (3000) sırasında, yer değiştirme (V)'yi B(100) tablosunda belleğe alan alt-program 3500 ve yer değiştirmeyi görseleştiren alt-program 3400 çağrılır. Alt-program 3400 ve Y2'nin değerlerini de kontrol eder. Bu işlem sırasında, değerler ekranın sınırlarından taşarsa, yer değiştirme görselleştirilmeyip yalnızca belleğe alınır (kuşkusuz akış şemasında görülmeyen bir göstergeyle). Daha sonra göreceğimiz gibi, bu program ölçek değiştirme olanağı ver-

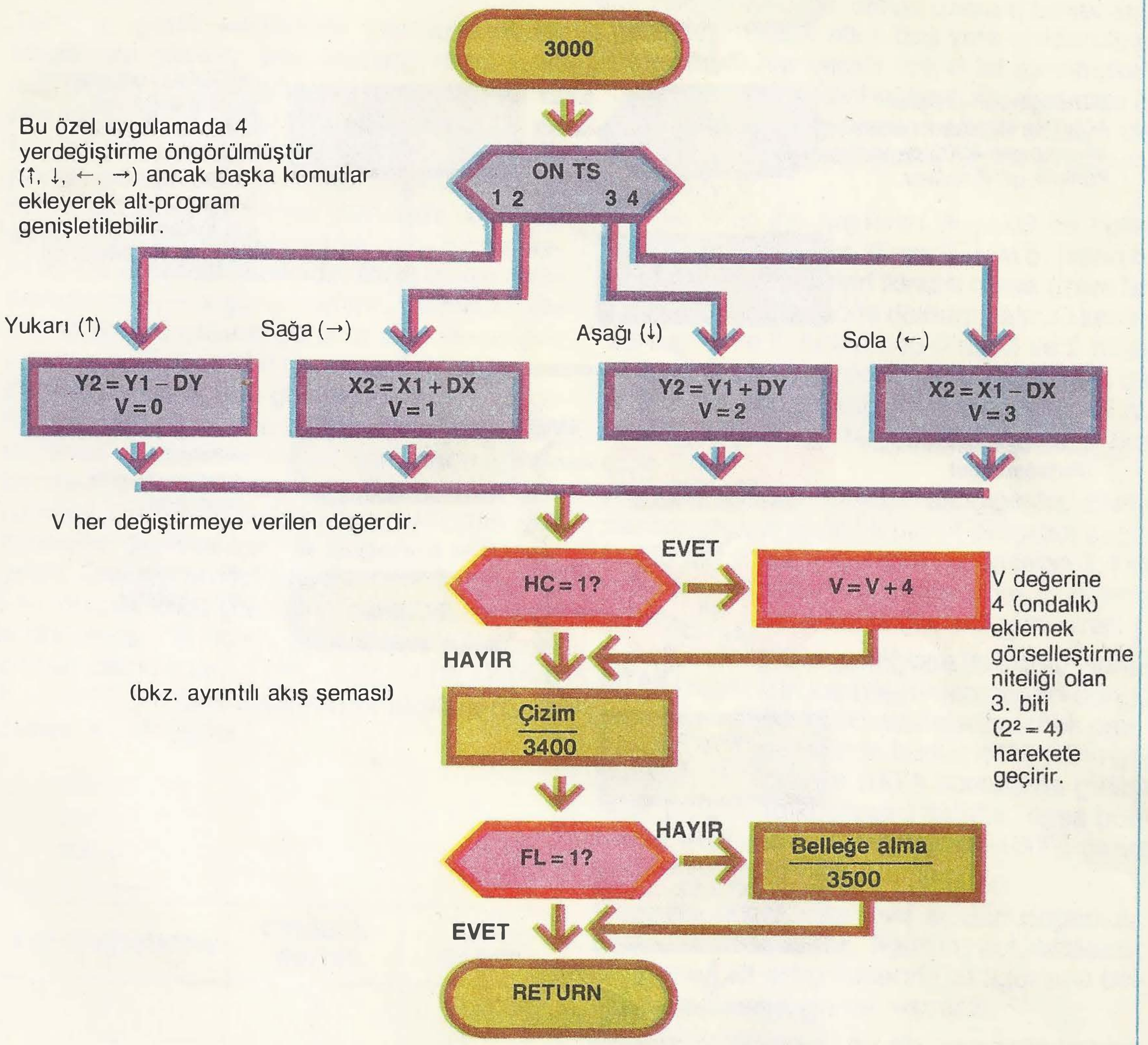
GİRİLMİŞ KARAKTERİN KONTROLU ALT-PROGRAMI



Bu kontrol ve onu izleyen duraklama, test aşamasında gerekli olabilir (bu versiyonda TS 4'ten yüksek değer alamaz).

Bu noktada çıkış K=1 (yani girilen karakterin ASCI kodu ne yerdeğiştirmelerde ne de komutlarda öngörüldüğü zaman) yapılır.

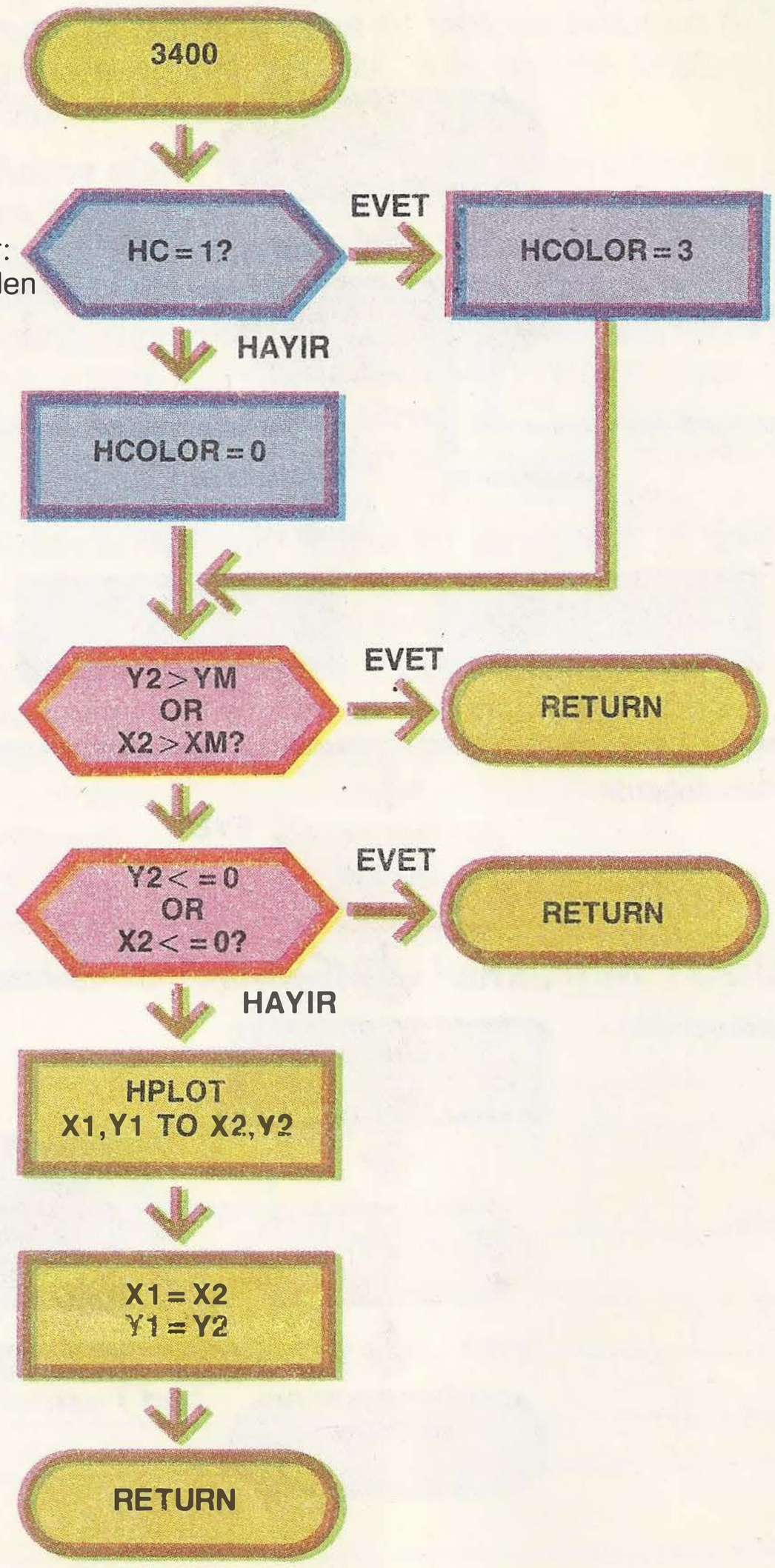
YERDEĞİŞTİRMELERİN YÖNETİMİ ALT-PROGRAMININ AKIŞ ŞEMASI



diğinden bu yöntem seçilmiştir. Bir yer değiştirme ekranın sınırları dışına çıkarsa, komut durumunu harekete geçirip, sonra yeniden düzeltmek üzere, gösterge ölçeğini değiştirmek yeter.

Alt-program 3500 ise daha karmaşıktır, çünkü V değişkeninin aşama aşama aldığı değerlerin yarattığı ''tıkanma''yı da gözönünde bulundurur (aşağıdaki akış şemasına bakınız). IN göstergesi, B(100) tablosunda ele alınan en son konumu içerir. Başlangıç değeri 0'dır. Buyüzden yer değiştirmede alt-program ara komutları atlayıp IN'de yani B(1)'in sağ

TEMEL BİR YERDEĞİŞTİRMENİN ÇİZİMİ ALT-PROGRAMI



HC değeri alt-program 1000'de ilkdeğerlenmiştir: alt-program 4000'de verilen komuta göre değişir.

HCOLOR kullanılan sayısal değerleri kullanılan kişisel bilgisayarın fonksiyonlarıdır. Başka makineler için kullanma klavuzlarına başvurulur.

XM ve YM alt-program 1000'de tanımlanmıştır (ilkdeğerleme)

Yerdeğiştirme ekranın üst sınırları dışında sona erer.

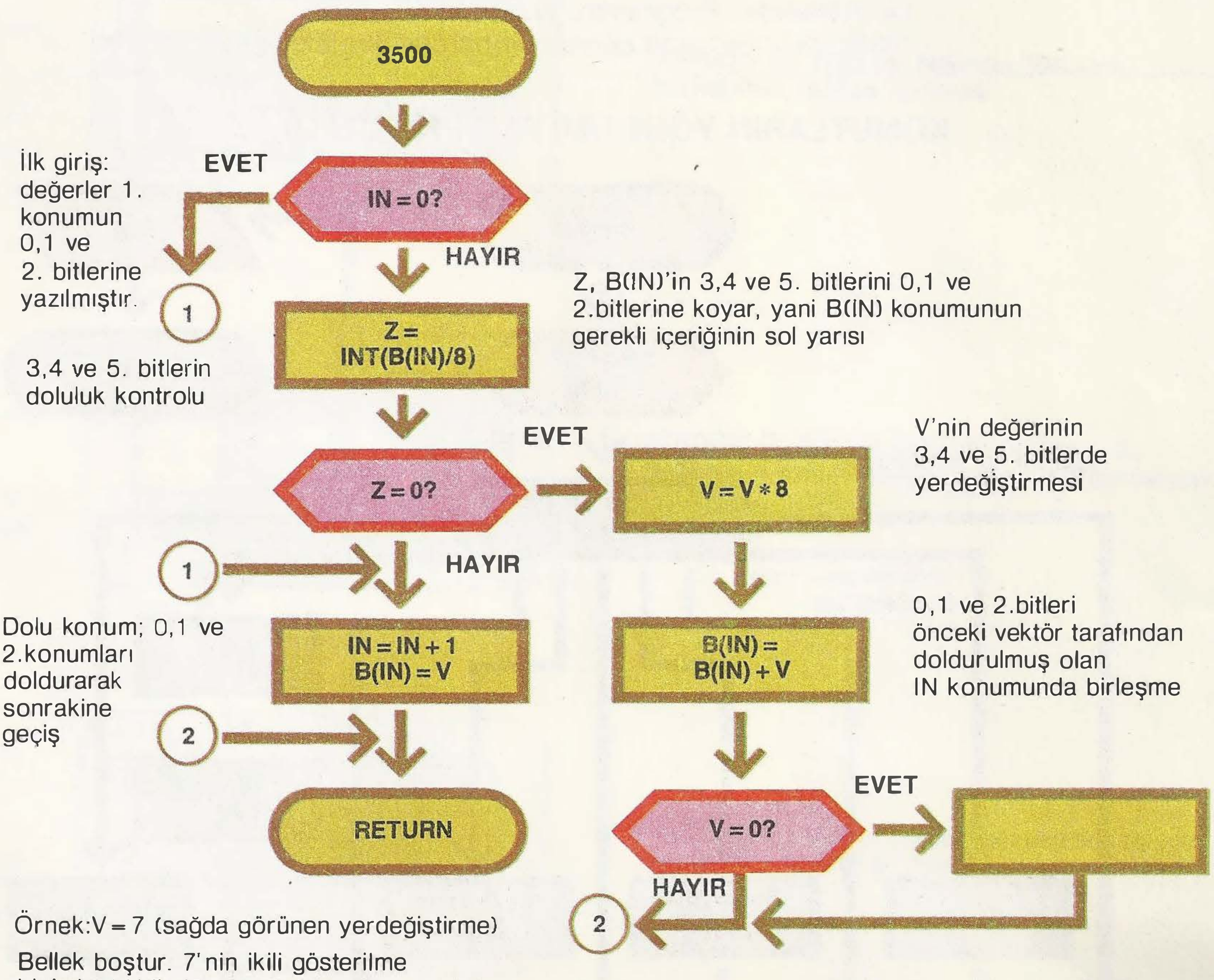
Alt sınırlar için aynı kontrol

bölümünde +1'i belleğe alır.
B(1) = V dersek, V değeri 0'dan 2'ye bitlerde, buna denk düşen konumları alır. İkinci çağırma sırasında IN artık 0 değildir, merkez bölüm tarafından harekete geçirilir. Bu veriler 8'e bölünen B(IN)'in içeriğini çözümler. B(IN)'de içerilen veri yalnızca 0, 1 ve 2 bitlerini doldurıyorsa, bölmenin tamsayı değeri 0'dır, öbür durumlarda 0'dan farklıdır.
Bölmenin sonucu 0'sa, sağ bölüm (0'dan 2'ye bitler) önceden dolu demektir (belleğin yalnızca 0 içermesi olanaksızdır çünkü IN yazmadan önce artmıştır). O zaman V'yi 3,4 ve 5 bitlerinde belleğe almak gerekir; bu da değerinin 8'le çarpılıp sonucun önceki içeriğe eklenmesiyle elde edilir. Bu nedenle bir adrese ilk girilen yer değiştirmenin sağda, sonrakinin ise solda bulunduğunu belirtelim. S.1604'te yalnız ca, herbiri 9 olası komuttan birine ayrılmış başka alt-programları çağırmak üzere oluşturulmuş alt-program 4000'in (komutların yönetimi) akış şeması görülmektedir. Adreslenmiş çağırmalara (ON TP...) yeni komutlar eklenerek yeni fonksiyonlar ve bunların gerçekleştirilmesinde gerekli olan alt-programlar eklenebilir. Görünen ya da görünmeyen yer değiş-

tirme fonksiyonları yalnızca HC göstergesinde 0 ya da 1 değerinin verilmesini gerektirir ve ana alt-programda yerine getirilirler.

Aşırı ucuna çekilmiş bu alt bölünme yöntemi ilginçtir. Alt-program 4000 öbür komutların tümünü içerip, sonraki alt-programlara geçirilecek göstergeleri belirtme gereğini ortadan kaldırıyor olabilirdi. O zaman bu alt-program çok daha karmaşık ve değiştirilmesi güç olabilirdi. Ayrıca belirli bir işe ayrılmış alt-programlar, aynı alt-programın, programın bir çok noktasında yeniden kullanılmasıyla, komutların sa-

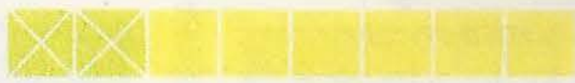
FİGÜR TABLOSU BİÇİMİNDE BELLEĞE ALMA ALT-PROGRAMI



Örnek:V=7 (sağda görünen yerdeğiştirme)

Bellek boştur. 7'nin ikili gösterilme biçimi 111'dir (0,1 ve 2. bitlerde yazılıdır).

V=4(yukarı doğru görünen yerdeğiştirme)



Kullanılmamış olanlar Gerekli bellek

Gerekli bellek sağ bölümünde doludur (INT(B(.)/8=INT(7/8)=0). O zaman 5%V'nin 3 bit ilerletilmesi gerekir: V=V*8= 32=1 0 0 0 0 0 (32=2∧5) Belleğin içeriği çizime denk düşer.



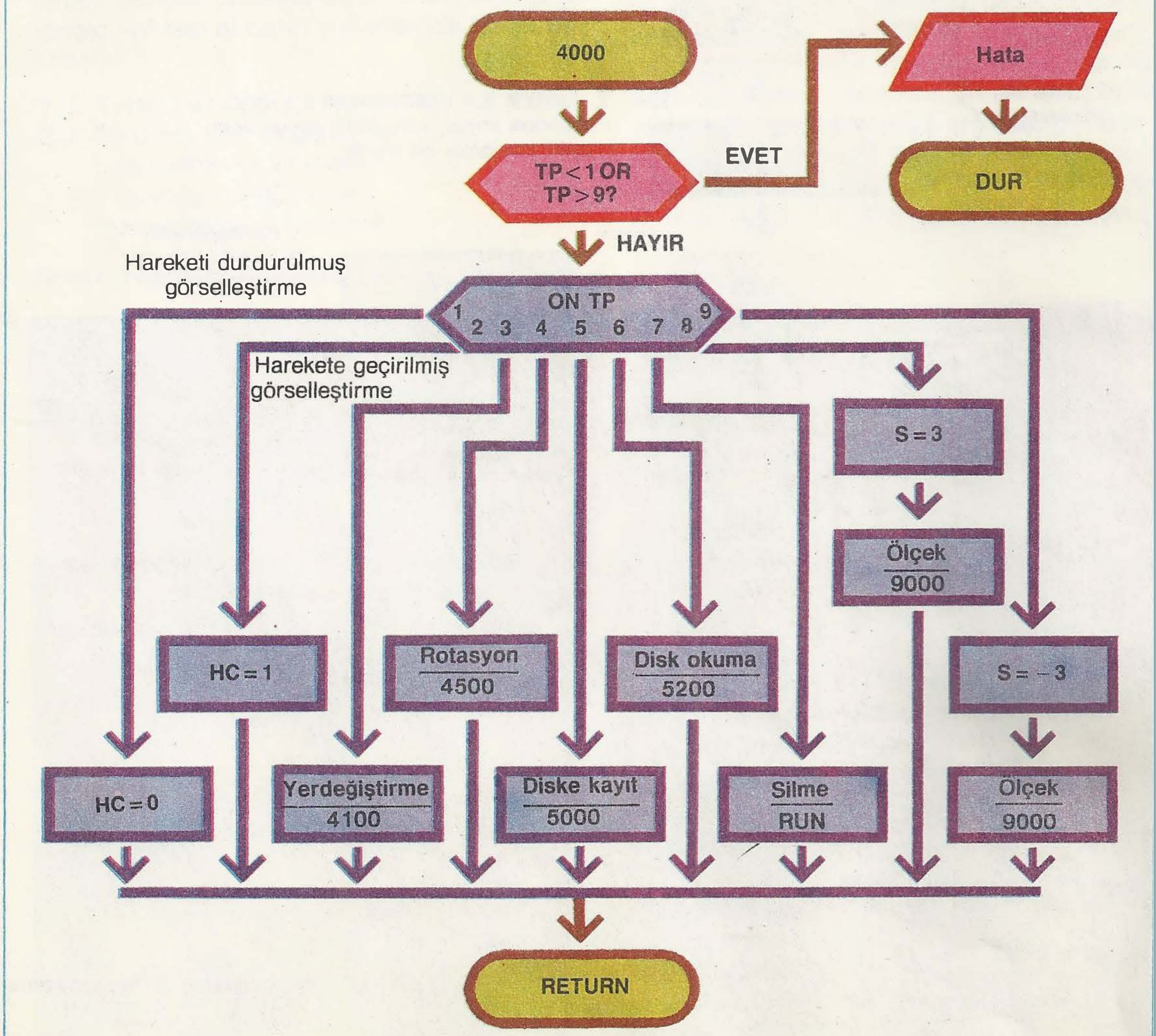
Yerdeğiştirme N+1 Yerdeğiştirme N

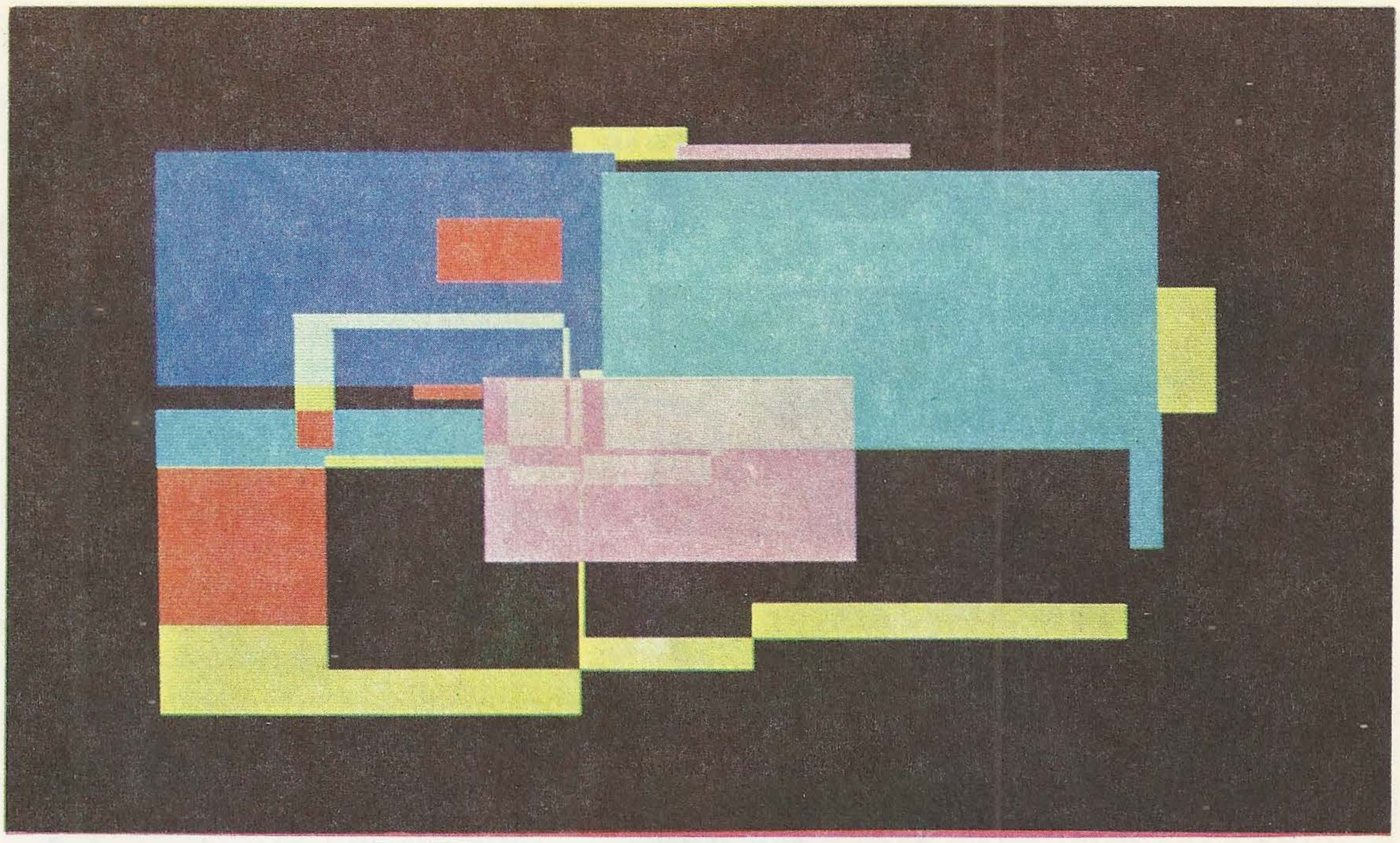
yısının azaltılmasına olanak verir.

Bütünün Bölümlerinin Yönetimi

Bu alt-programın işlevi figürün bütününün ekra ın herhangi bir alanının içinde hareket ettirmektir. (Resmin oluşturulması sırasında yer değiştirmeleri yöneten alt-program 3000'le karıştırmamak gerekir). Resmin naklinden önce yer değiştirme tablosu B(100)'ün korunmasını sağlar. Özgün tablonun değişmemesi için dengi bir tablonun kullanılması yeterlidir. Sonuçlar doyurucu değilse önceki duruma geri dönmek de mümkündür. Gerçekleştirilecek ilk fonksiyon istenen yer değiştirme türünün okunmasıdır. Programın, tablonun oluşturulmasında kullanılana benzer bir parçasıdır; tek

KOMUTLARIN YÖNETİMİ ALT-PROGRAMI





Basit yüksek düzeyli komutlarla oluşturulmuş bir grafik kompozisyon.

değişen naklin sona erdiğini gösteren 0 değerinin belirlenmesidir. Resmin yer değiştirmelerini harekete geçiren tuşlar, figürü oluşturan yer değiştirmelerin tablosunun çıkarılmasında kullanılanlarla aynıdır; bu nedenle onlar da alt-program 2000 tarafından kontrol edilir.

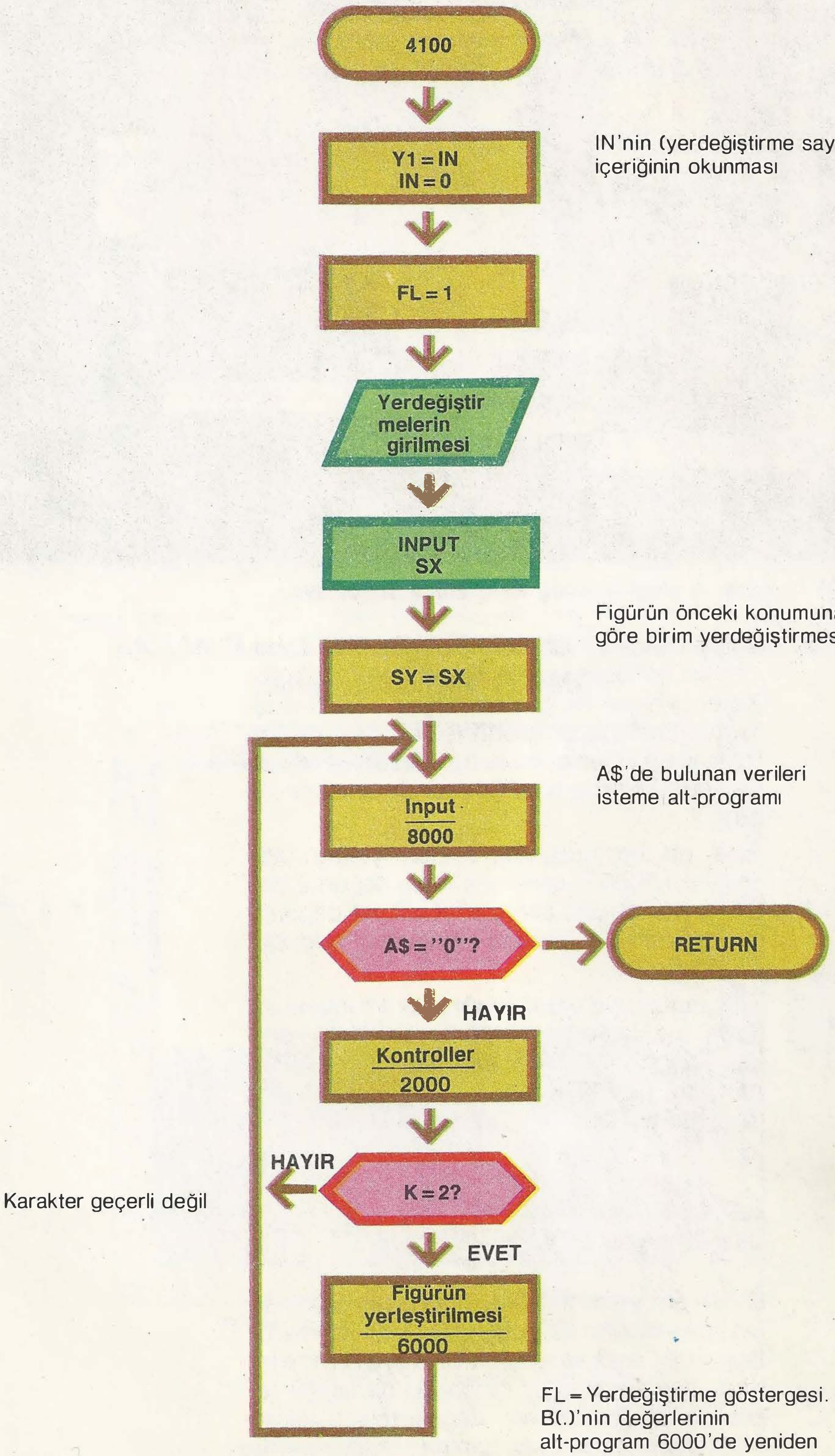
Yine de bu durumda komut yoktur: Alt-program 2000'in çıkışında K'nın değeri 2'den başkaysa dikkate alınmaz (K = 2 yer değiştirmeye ayrılmış tuşlardan birinin basılı olduğunu gösterir).

Tuş saptanırsa (yani öngörülen 4 tuştan biri) figürü hareket ettiren alt-program 6000 harekete geçer. Alt-program 6000'in başında, program yeni bir yer değiştirmeyi ya da sıfır değerini bekler.

Figürün Yer Değiştirmesi (Alt-program 6000).

Bu alt-program B(100)de figürü ve koordinatları istenen nakil türü doğrultusunda, X0 ve Y0 başlangıç noktasına göre SX veya SY miktarında değiştirilmiş bir noktadan hareketle resimleri oluşturan yer değiştirmeleri okur. Fonksiyon, kodun niteliğinin (görünen ya da görünmeyen) ve yer değiştirme türünün (0,1,2,3) yer değiştirmeler tablosunda hesap-

BÜTÜNÜN BÖLÜMLERİNİN YÖNETİMİ



IN'nin (yerdeğiştirme sayacı) içeriğinin okunması

Figürün önceki konumuna göre birim yerdeğiştirmesi

A$'de bulunan verileri isteme alt-programı

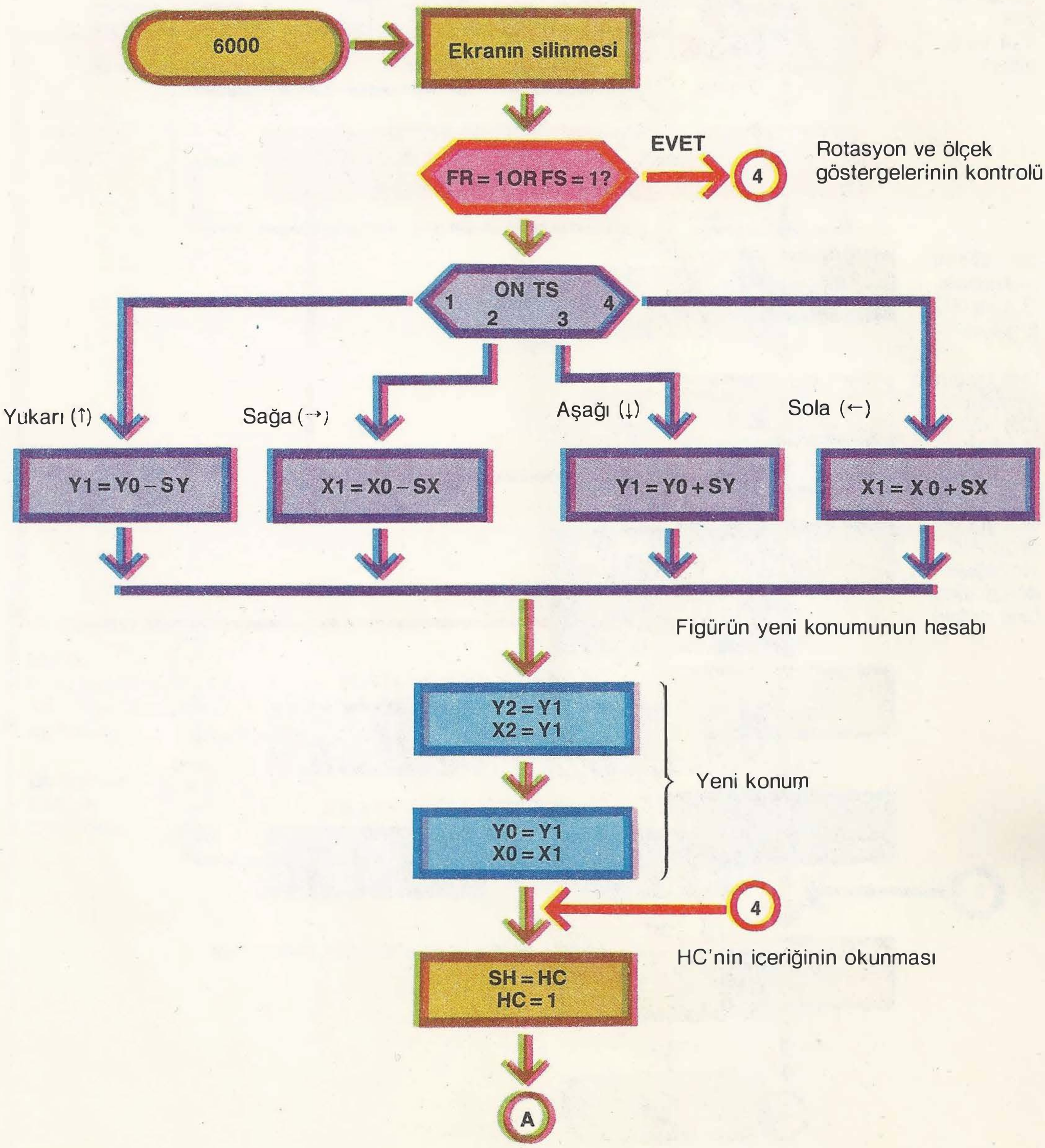
Karakter geçerli değil

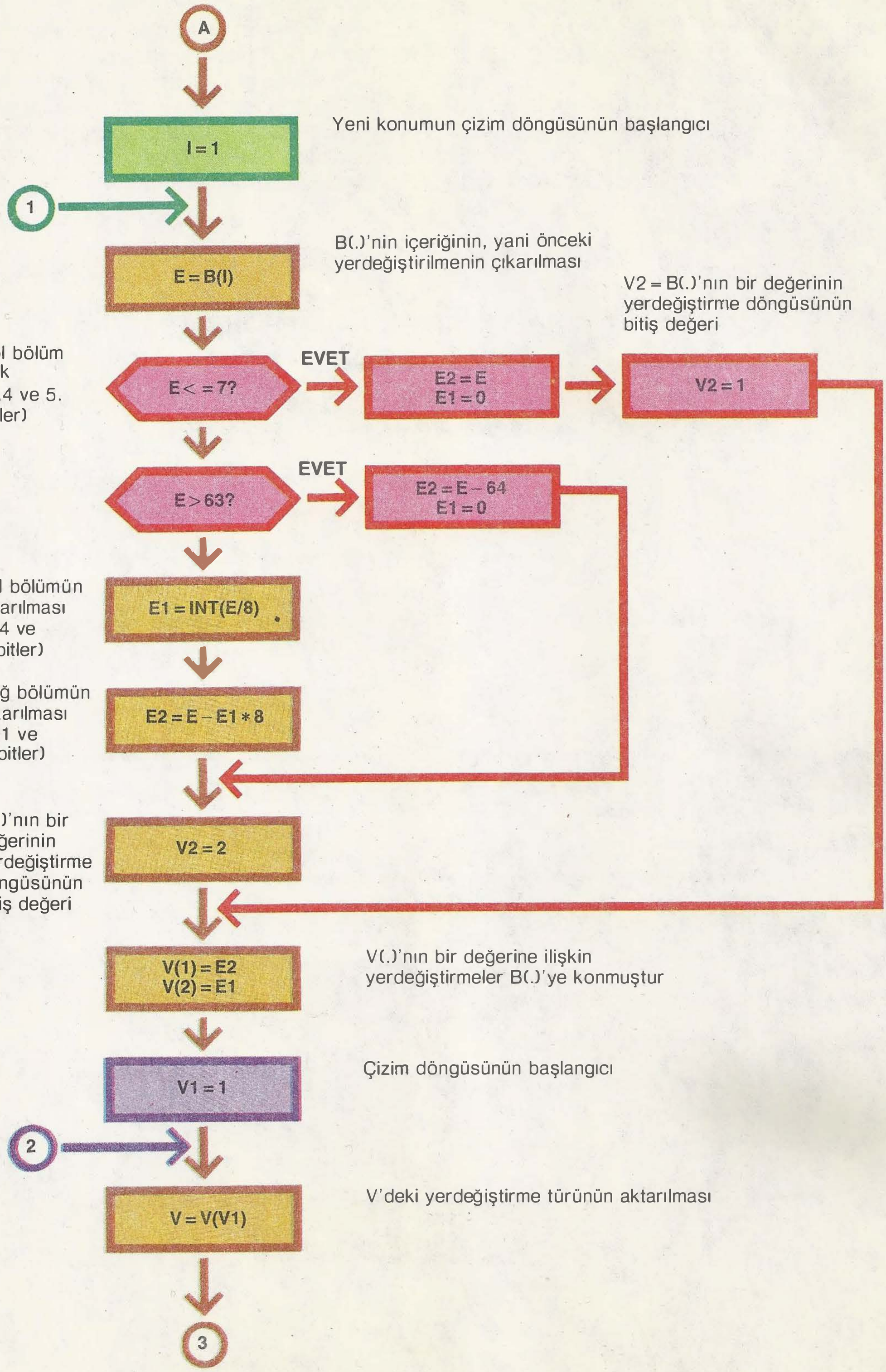
FL = Yerdeğiştirme göstergesi. B(.)'nin değerlerinin alt-program 6000'de yeniden hesaplanmasını önler.

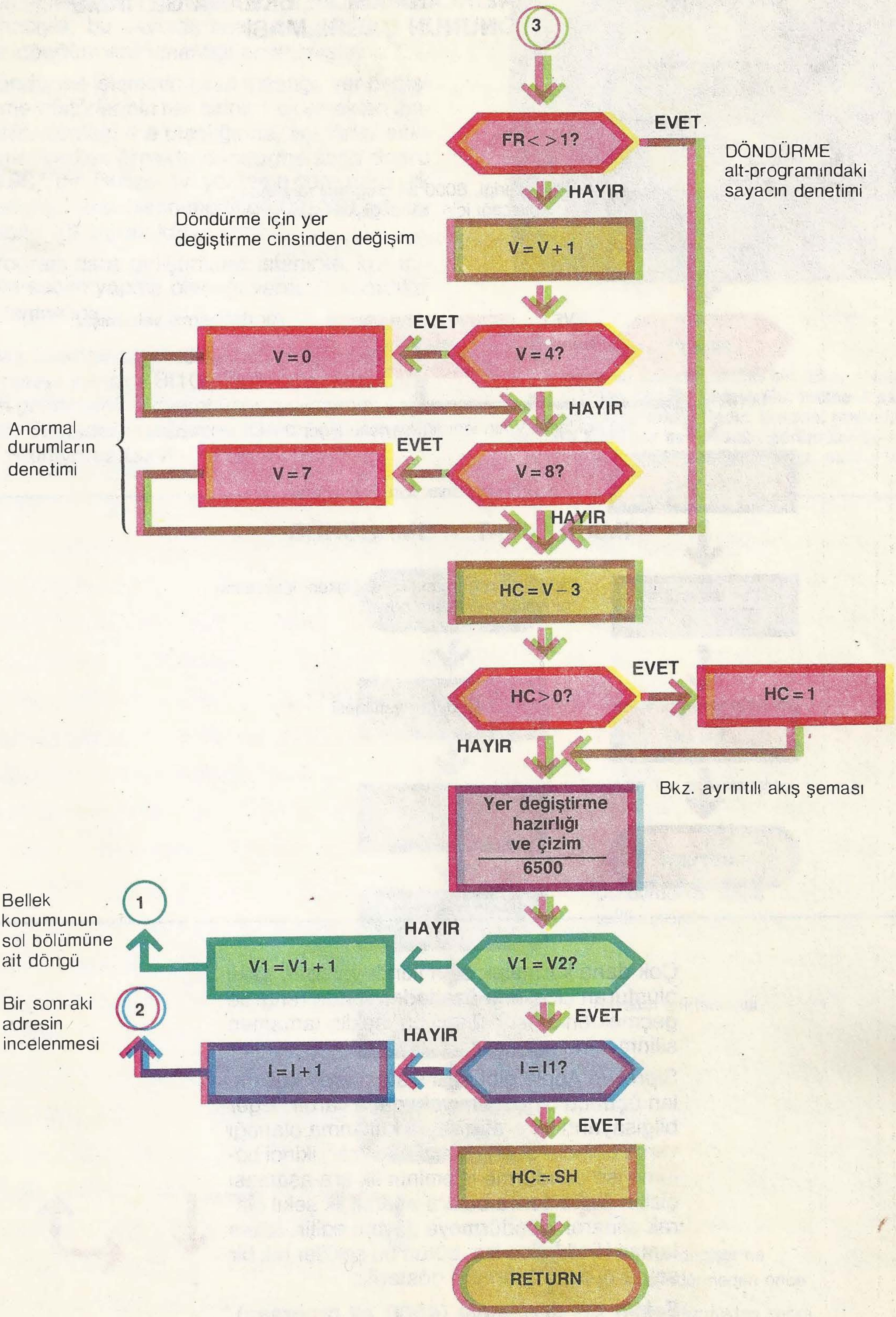
Yeni komut

lanmasıyla ve tablo oluşturma alt-programlarının kullanılmasıyla elde edilir. Aslında bu iki değer zamanlı olarak da alınır.
Akış şemasını ve dökümü fazla doldurmamak için önceki yer değiştirmenin silinmesi hesaba katılmamıştır. Bir figürün nakli harekete geçirildiğinde, aynı figür başka bir noktadan tekrarlanır. Son konum (örneğin 10 ara geçişten sonra) elde edilmemişse, figürün o ölçümü de tekrarlanır. Bunlardan her biri öncekine göre son değiştirmenin 1/10'i kadar yer değiştirir. İki yöntemle de bu terslik önlenir. İlki yer değiştirmeden önce bütün grafik sayfanın silinmesidir (ama bu takdirde başlangıç figürü de silinir). Bunu önlemek için alt-program 6000'i çağırmadan önce ekran belleğini silme komutunu sokmak yeter. Boşaltılacak başka ekran belleği yoksa, alt-program 6000'in sonunda figür özgün konumunda yeniden oluşturulabilir.

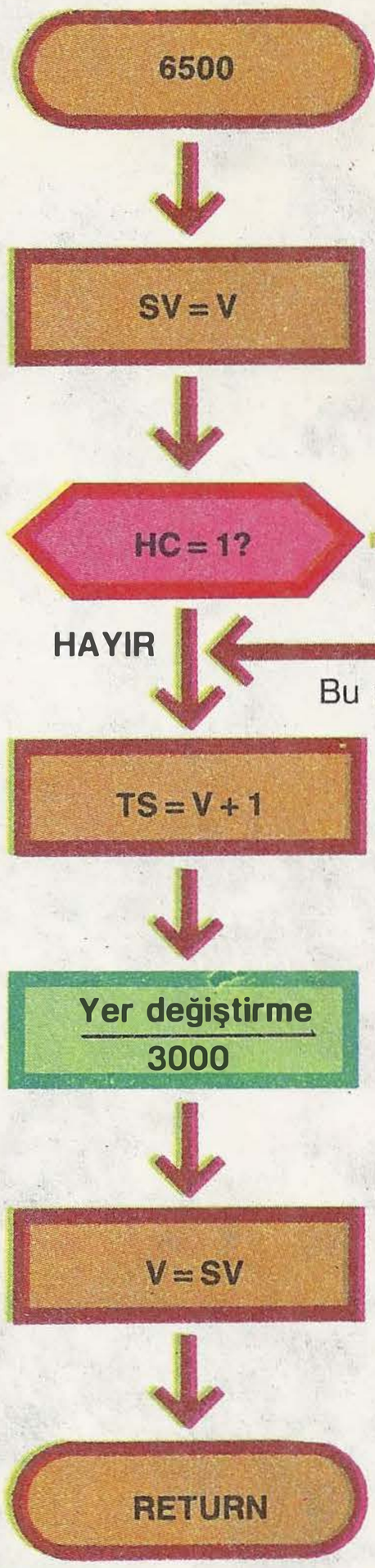
FİGÜRÜN YERDEĞİŞTİRMESİ ALT-PROGRAMI




A
I=1
1
E=B(I)
Yeni konumun çizim döngüsünün başlangıcı
B(.)'nin içeriğinin, yani önceki yerdeğiştirilmenin çıkarılması
V2=B(.)'nın bir değerinin yerdeğiştirme döngüsünün bitiş değeri
Sol bölüm yok (3,4 ve 5. bitler)
E<=7?
EVET
E2=E
E1=0
V2=1
E>63?
EVET
E2=E-64
E1=0
Sol bölümün çıkarılması (3,4 ve 5.bitler)
E1=INT(E/8)
Sağ bölümün çıkarılması (0,1 ve 2.bitler)
E2=E-E1*8
B(.)'nın bir değerinin yerdeğiştirme döngüsünün bitiş değeri
V2=2
V(1)=E2
V(2)=E1
V(.)'nın bir değerine ilişkin yerdeğiştirmeler B(.)'ye konmuştur
V1=1
Çizim döngüsünün başlangıcı
2
V=V(V1)
V'deki yerdeğiştirme türünün aktarılması
3


3
FR< >1?
EVET
HAYIR
DÖNDÜRME alt-programındaki sayacın denetimi
Döndürme için yer değiştirme cinsinden değişim
V=V+1
EVET
V=4?
V=0
HAYIR
Anormal durumların denetimi
EVET
V=8?
V=7
HAYIR
HC=V-3
HC>0?
EVET
HC=1
HAYIR
Bkz. ayrıntılı akış şeması
Yer değiştirme hazırlığı ve çizim
6500
Bellek konumunun sol bölümüne ait döngü
1
HAYIR
V1=V2?
V1=V1+1
Bir sonraki adresin incelenmesi
2
EVET
HAYIR
I=I1?
I=I+1
EVET
HC=SH
RETURN

YER DEĞİŞTİRMENİN HAZIRLANMASI VE EKRANA GETİRME FONKSİYONUNUN ÇAĞRILMASI



V'nin içeriği, 6000 alt-programı tarafından kullanılacağı için, SV değişkeninde toplanır.

Yer değiştirme yalıtılmıştır.

Bu noktada, V'nin içerdiği sayısal değer 0 ila 2 arasındadır.

TS'yi hesaplamak için 1 eklenmesi

3000 alt-programı, TS değişkeni klavyeden girilmiş gibi devam eder.

V'ye yeniden değer verilmesi

Çok daha karmaşık olan ikinci yöntem, şekli oluşturan çizgilerin üzerinden fonun rengi ile geçmekten oluşur. Tabii bu, şeklin tamamen silinmesine yol açar.

Siprel ve Apple gibi bazı sistemlerde kullanılan üçüncü bir çözüm yolu daha vardır. Eğer bilgisayarın iki grafik sayfa kullanma olanağı varsa birincisi şeklin ilk haline ayrılır, ikinci bölüme ise döndürme işleminin ilk ara-aşaması çizilir. Daha sonra bu ara-aşama ilk şekil olarak alınarak döndürmeye devam edilir. İşlem tamamlandığında ise, bütün bu şekiller tek bir sayfa düzeni üzerinde gösterilir.

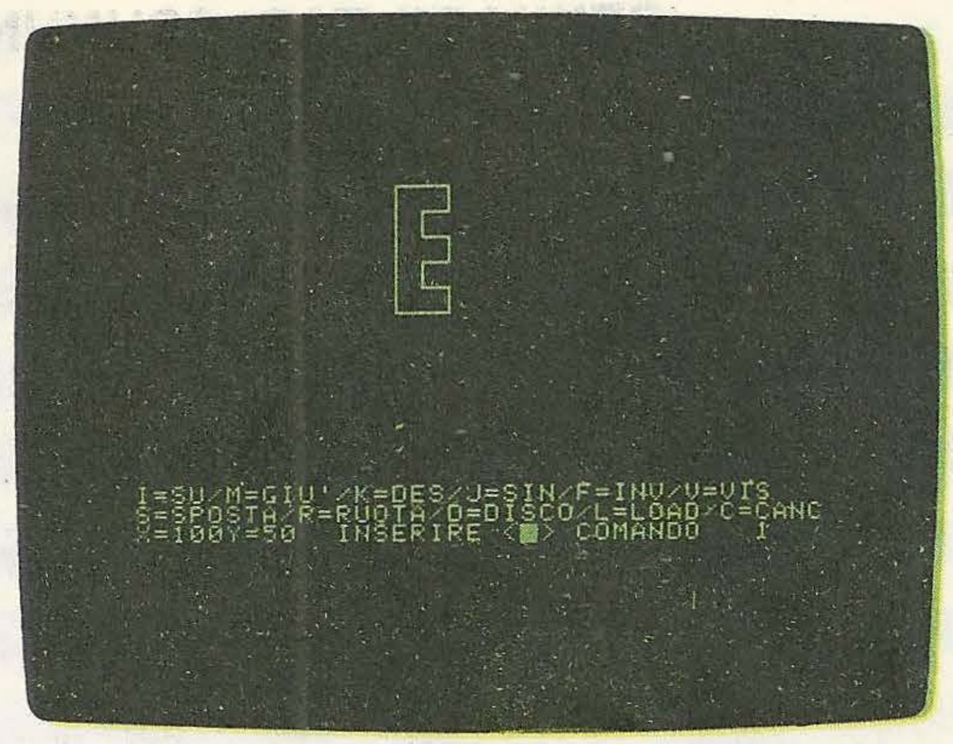
Şeklin döndürülmesi (4500 alt-programı). Bu fonksiyon dört yöne yer değiştirme için faz-

la elverişli değildir: Program, eğimli çizgiler çizememektedir. Yalnızca örnek oluşturmak amacıyla, bu sayfada saat yönünde 90°'lık bir döndürmenin mantığı anlatılmıştır.

Döndürme işleminin basit mantığı, yer değiştirme vektörlerinin her birine 1 eklemekten ibarettir. Toplam 4'e ulaştığında, vektörler sıfırlanır. Verilen örnekte döndürme sağa doğru ve 90°'dır. Benzer bir yönteme göre şeklin ilk konumu, aksi bildirilmedikçe (X0,Y0) artı bir sabit (10) olarak kabul edilir.

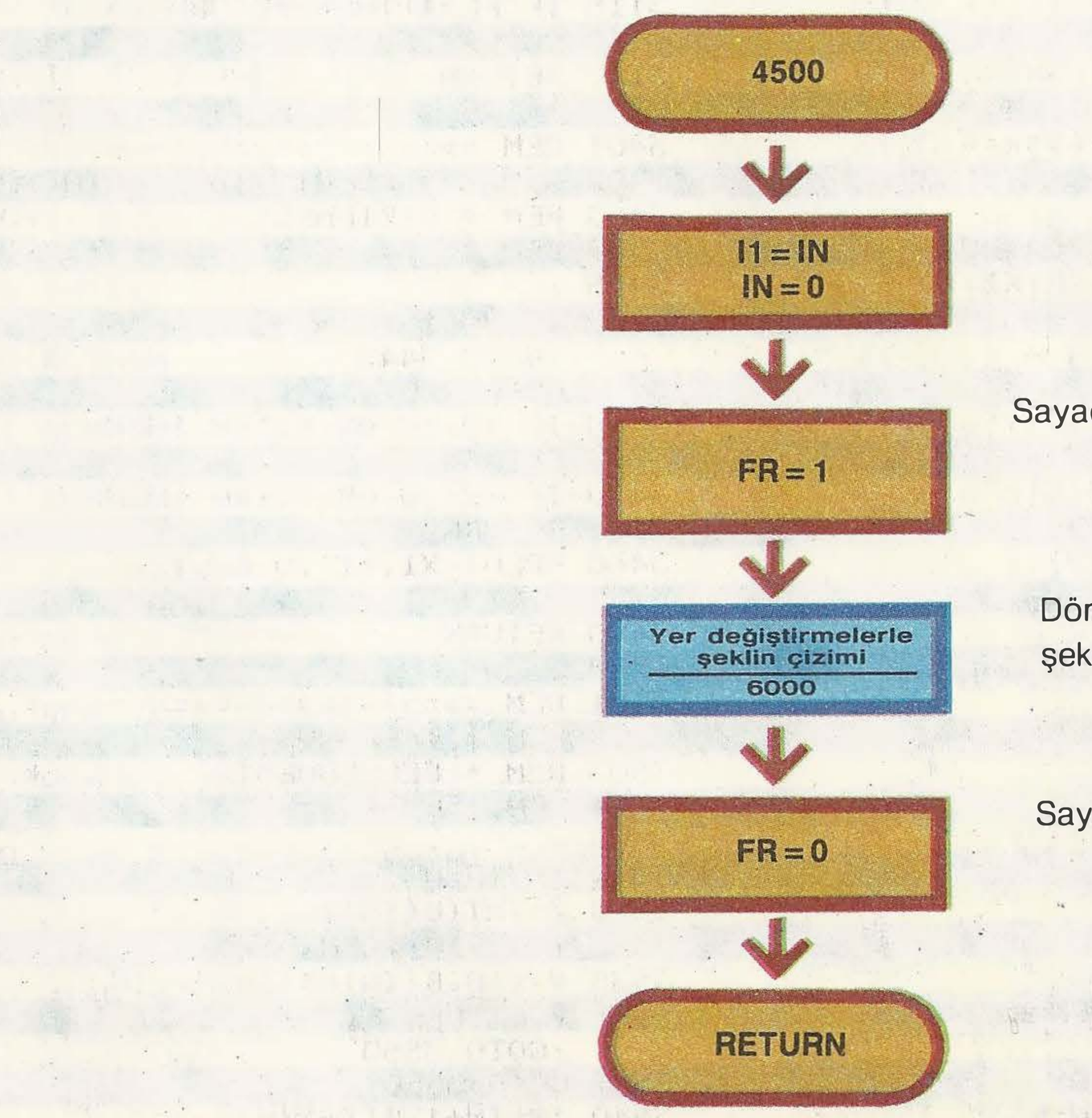
Program daha geliştirilmek istenirse, kullanıcıya seçim yapma olanağı verecek komutlar eklenmelidir.

Disk üzerine kayıt (5000 alt-programı). Alt-program yalnızca B(100)'de içerilen kütük adının girilmesini ve disket üzerine yazımını kapsar. Bu nedenle yalnızca dökümünü vermekle yetiniyoruz (sayfa 1612 ve 1614).



Programın aşamalarından biri (bkz. döküm, s.1612'den 1614'e kadar). Bu, metne 4 satır ayıran 1. grafik sayfadır. Burada, metindeki pencere komutlar menüsünün görüntülenmesini sağlıyor (bkz. satır 118).

DÖNDÜRME ALT-PROGRAMI



ŞEKİLLER TABLOSUNUN YARATILMASI VE KULLANIMI

```
1 REM ==========================
2 REM *SEKILLER TABLOSUNUN      *
3 REM * YARATILMASI             *
4 REM * VE                      *
5 REM * KULLANIMI               *
6 REM ==========================
7 :
8 :
100 GOSUB 1000
110 :
111 REM ========================
112 REM * KARAKTERLERIN         *
113 REM *    OKUNMASI           *
114 REM ========================
115 :
117 HGR:HOME
118 VTAB 21:PRINT "I=YUKARI/M=ASAGI
    /K=SAGA/J=SOLA/F=TERS/V=GOR":
     PRINT "S=YERD/R=DOND/D=DISK/
     L=YUKLE/C=SIL"
120 GOSUB 8000
140 GOSUB 2000
150 ON K GOTO 160,170,190
160 GOTO 120
170 GOSUB 3000
180 GOTO 120
190 :
192 GOSUB 4000
200 GOTO 120
300 STOP
1000 :
1001 REM ======================
1002 REM * SIFIRLAMA          *
1003 REM ======================
1004 :
1010 DIM B(100)
1020 DX=10:DY=10:Y0=50:X0=100
     :XM=278:YM=159:Y1=Y0:X1=X0

1025 X2=X1:Y2=Y1:S=1
1030 S(1)=73:S(2)=75:S(3)=77
     :S(4)=74
1035 O(1)=70:O(2)=86:O(3)=83
     :O(4)=82:O(5)=68:O(6)=76

1040 O(7)=67:O(8)=43:O(9)=4
      5
1045 D$=CHR$(4)
1100 RETURN
2000 :
2001 REM ======================
2002 REM * KARAKTER           *
2003 REM * KONTROLU           *
2004 REM ======================
2005 :
2010 K=1:TS=0:TP=0:I=1
2020 IF K<>1 THEN RETURN
2030 IF S(I)=0 THEN 2050
2040 IF ASC(A$)=S(I) THEN K=2
     :TS=I:GOTO 2200
2050 IF O(I)=0 THEN 2070
2060 IFASC(A$)=O(I) THEN K=3
     :TP=I:GOTO 2200
2070 IF I=10 THEN RETURN
2080 I=I+1:GOTO 2020
2100 :
2200 :
2210 GOTO 2070
3000 :
3001 REM ======================
3002 REM * YERDEGISTIRMELERIN *
3003 REM * YONETIMI           *
3004 REM ======================
3005 :
3010 ON TS GOTO 3020,3030,3040
      ,3050
3011 :
3012 REM <YUKARI>
3013 :
3020 Y2=Y1-DY:V=0:GOTO 3100

3021 :
3022 REM <SAGA>
3023 :
3030 X2=X1+DX:V=1:GOTO 3100

3031 :
3032 REM <ASAGI>
3033 :
3040 Y2=Y1+DY:V=2:GOTO 3100

3041 :
3042 REM <SOLA>
3043 :
3050 X2=X1-DX:V=3
3060 :
3100 :
3105 IF HC=1 THEN V=V+4
3110 GOSUB 3400
3115 IF FL=1 THEN RETURN
3150 GOSUB 3500
3160 RETURN
3400 :
3401 REM ======================
3402 REM * YERDEGISTIRMELERIN *
3403 REM * CIZILMESI          *
3404 REM ======================
3405 :
3420 IF HC=1 THEN HCOLOR=3
     :GOTO 3440
3430 HCOLOR=0
3440 IF Y2>YM OR X2>XM THEN
     RETURN
3450 IF Y2<=0 OR X2<=0 THEN
     RETURN
3460 HPLOT X1,Y1 TO X2,Y2
3470 X1=X2:Y1=Y2
3480 RETURN
3500 :
3501 REM ======================
3502 REM * TABLONUN           *
3503 REM * BELLENMESI         *
3504 REM ======================
3505 :
3510 IF IN=0 THEN 3540
3520 Z=INT(B(IN)/8)
3530 IF Z<>0 THEN 3540
3535 V=V*8:B(IN)=B(IN)+V:IF
     V=0 THEN B(IN)=B(IN)+64
     :GOTO 3550
3537 GOTO 3550
3540 IN=IN+1:B(IN)=V
3550 RETURN
4000 :
4001 REM ======================
4002 REM * SIRALAMANIN         *
4003 REM * YAPILMASI          *
4004 REM ======================
4005 :
4010 IF TP<1 OR TP>9 THEN STOP
```

```
4011 :
4020 ON TP GOTO 4030,4040,4050
     ,4060,4070,4080,4090
     ,4095
4030 HC=0:RETURN
4040 HC=1:RETURN
4050 GOSUB 4100:RETURN
4060 GOSUB 4500:RETURN
4070 GOSUB 5000:RETURN
4080 GOSUB 5200:RETURN
4085 RUN
4090 S=3:GOSUB 9000:RETURN
4095 S=-3:GOSUB 9000:RETURN

4100 :
4101 REM ========================
4102 REM * YERDEGISTIRMELERIN *
4103 REM *  YONETIMI           *
4104 REM ========================
4105 :
4106 FL=1
4140 I1=IN
4145 VTAB 23:HTAB 12:INPUT
     "YERDEGISTIRME KATSAYISI? ";SX

4147 SY=SX
4148 VTAB 23:HTAB 22:PRINT "
4150 GOSUB 8000
4160 IF A$="0" THEN 4300
4170 GOSUB 2000
4180 IF K<>2 THEN 4150
4190 GOSUB 6000
4200 GOTO 4150
4300 :
4310 :
4320 :
4325 IN=I1
4327 FL=0
4330 RETURN
4500 :
4501 REM ========================
4502 REM * DONDURME             *
4503 REM ========================
4504 :
4510 I1=IN:IN=0
4520 FR=1:GOSUB 6000
4530 FR=0
4540 RETURN
5000 REM ========================
5001 REM *    DISKE KAYIT       *
5002 REM ========================
5003 :
5010 VTAB 23:HTAB 12:PRINT "ADI
     :"
5020 VTAB 23:HTAB 19:INPUT " "
     ;NM$
5030 VTAB 1:PRINT D$ "AC"NM$
5040 PRINT D$"YAZ"NM$
5045 PRINT IN
5050 FOR I=1 TO IN:PRINT B(I)
     : NEXT I
5060 PRINT D$"KAPA"NM$:PRINT
     D$
5070 RETURN
5200 REM ========================
5201 REM * YUKLEME              *
5202 REM ========================
5203 :
5210 VTAB 23:HTAB 12:PRINT "ADI
     "
     ;NM$
5230 VTAB 1:PRINT D$"AC"NM$
5240 PRINT D$"OKU"NM$
5250 INPUT I1
5255 IN=I1
5260 FOR I=1 TO I1:INPUT B(I)
     :NEXT I
5270 PRINT D$"KAPA"NM$:PRINT
     D$
5273 X0=100:Y0=50:X1=X0:Y1=Y0:X2
     =X1:Y2=Y1:FL=1
5275 GOSUB 6105:FL=0
5200 RETURN
6000 :
6001 REM ========================
6002 REM * SEKLIN               *
6003 REM * YER DEGISTIRMESI     *
6004 REM ========================
6005 :
6007 HGR
6008 IF FR=1 OR FS=1 THEN 6105

6010 ON TS GOTO 6020,6030,6040
     ,6050
6020 Y1=Y0-SY:GOTO 6100
6030 X1=X0+SX:GOTO 6100
6040 Y1=Y0+SY:GOTO 6100
6050 X1=X0-SX
6100 X2=X1:Y2=Y1:Y0=Y1:X0=X1

6105 SH=HC:HC=1
6110 FOR I=1 TO I1
6120 E=B(I)
6130 IF E<=7 THEN E1=0:E2=E:V2=1
     :GOTO 6155
6135 IF E>63 THEN E2=E-64:E1=0
     :GOTO 6153
6140 E1=INT(E/8)
6150 E2=E-E1*8
6153 V2=2
6155 V(1)=E2:V(2)=E1
6160 V1=1
6165 V=V(V1)
6167 IF FR<>1 THEN 6175
6169 V=V+1
6170 IF V=4 THEN V=0
6172 IF V=8 THEN V=4
6175 HC=V-3:IF HC>0 THEN HC=1

6180 GOSUB 6500
6185 VTAB 23:HTAB 35:PRINT TS
6190 :
6200 IF V1<>V2 THEN V1 =V1+1
     :GOTO 6165
6210 NEXT I
6220 IF I<I1 THEN I=I1:NEXT I

6230 HC=SH:RETURN
6500 :
6501 REM ========================
6502 REM * YERDEGISTIRMENIN *
6503 REM * HAZIRLANMASI      *
6504 REM ========================
6505 :
6510 SV=V:IF HC=1 THEN V=V-4
6520 TS=V+1:GOSUB 3000
6530 V=SV:RETURN
8000 :
8001 REM =========
8002 REM * INPUT *
```

```
8003 REM ==================                    :HOME:END
8004 :                                    8030 RETURN
8010 VTAB 23:HTAB 12:PRINT                9000 REM ==================
     "KOMUTU (?) GIRINIZ"                 9001 REM *  OLCEK  .   *
8020 VTAB 23:HTAB 22:GET A$               9002 REM ==================
8021 VTAB 23:HTAB 35:INVERSE              9003 :
       :PRINT A$:NORMAL                   9010 FS=1
8022 VTAB 23: HTAB 1:PRINT "              9015 DY=DY+S
                "                         9020 DX=DX+S:X1=X:N=N+1:IF N=0
8024 VTAB 23:PRINT "X="X1"Y="Y1           9030 GOSUB 6000
                                          9040 FS=0:RETURN
8025 IF ASC(A$)=13 THN TEXT
```

Disketin okunması (5200 alt-programı). Bu alt-program yukardakilerin tam tersi işlevleri yerine getirir: Görüntüye getirilecek çizime ait verileri içeren kütüğün adı girilir girilmez, bütün içerik B(100)'e aktarılır ve 6000 gösterim alt-programı çağrılır (bkz. 1612-1614 sayfalarındaki döküm).

Grafik Figürlerin Yaratılması İçin Hizmet Programları

Burada inceleyeceğimiz yer değiştirmeleri döken bir tablonun oluşturulması örneği, özel komutlar içermeyen sistemlerde grafik figürlerin yaratılması yöntemini sergiliyor.

Çeşitli biçimleri bulunan kişisel bilgisayarlar iki tür komut kullanırlar:

- Birincisi (Apple ailesinin 6502 merkez birimine dayalı ve bunlarla uyumlu diğer bilgisayarlarda) örneğimizde yer alan birim yerdeğiştirmeleri, simgeleri ve uzlaşmaları (konvansiyonları) kullanır.
- İkinci tür ise, içinde yer değiştirmelerin birbiri ardından izlendiği bir dizi komut kullanır.

Birinci tür komutlarla bir figürler tablosu yaratmak için, en iyi çözüm POKE komutu yardımıyla yer değiştirme vektörlerini doğrudan belleğe yazmaktır. Öyleyse tabloyu önce bir kağıda çizerek birim yer değiştirmeleri üzerinde belirlemek, daha sonra ise bunu daha önce anlattığımız biçimde belleğe yüklemek doğru olacaktır.

Bundan başka karmaşık şekillerin yönetimi ve görüntülenmesi için gereken yüksek düzeyli komutların yer almadığı makinelerde bile, aynı işlemleri gerçekleştirmek olanaklıdır.

Bu noktada çok pratik iki örnek verelim:

- Çok yaygın bir terminolojiye sadık kalarak ''Shape Editor'' (şekil düzenleyici) adını verdiğimiz birinci program, bazen çok karmaşık şekillerin de ekran üzerinde doğrudan oluşturulabilmesine, sonrada bunların yer değiştirmeler cinsinden, bir tablo ile uzlaşılmış (konvansiyonel) bir ad altında diskette saklanmasına olanak vermektedir.
- ''Shape Loader'' (şekil yükleyicisi) adlı ikinci program ise, bunun tersini, yani daha önce oluşturulmuş ve diskte saklanmış bir şeklin geri yüklenmesini ve görüntülenmesini sağlamaktadır.

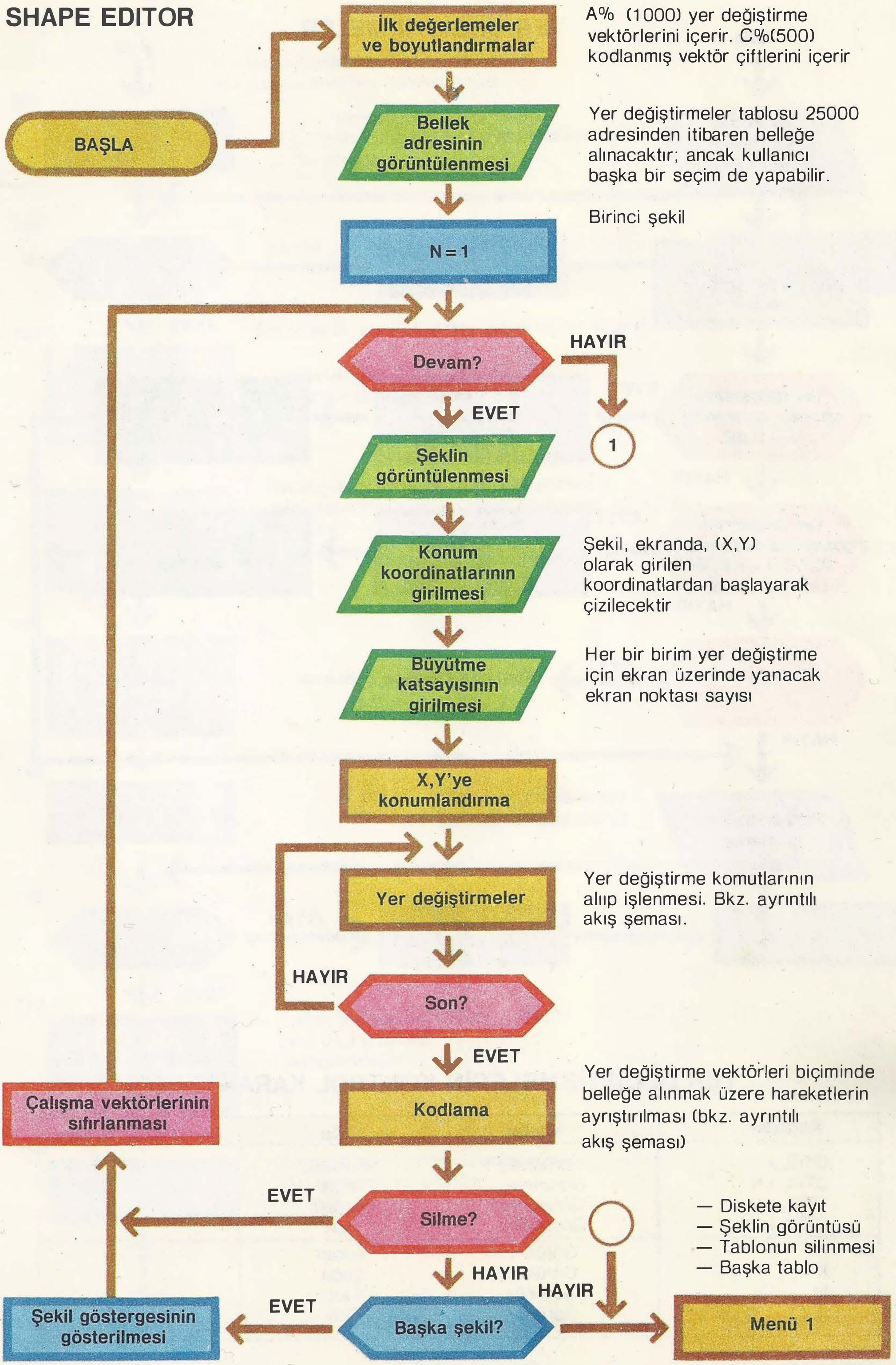
Şekil Düzenleyicisi. Shape Editor (akış şeması için bkz. yan sayfa) tuşlar yardımıyla ekran üzerinde bir şekil oluşturma olanağı veriyor. Bir noktadan hareket ederek, hayali bir çizgi üzerinde, görünür veya görünmez yer değiştirmeler yapılıyor. 1. Öbek yer değiştirme komutlarının alınmasını ve yürütülmesini ayrıntılarıyla veriyor (s. 1616).

Grafik çizim tamamlanınca, kodlama öbeği olan 3. öbeğe (s.1617) geçiliyor: A% dizeyinde kayıtlı bulunan noktaların yer değiştirmeleri, bir dizi çizim vektörüne çevriliyor ve C% dizeyine aktarılıyor. Kodlama bittiğinde, C%'deki vektörler belleğe yükleniyor Bu aşamada, kullanıcı bir şekil daha oluşturmak isteyebilir. Bu da, program tarafından yine aynı biçimde ele alınıp işlenir.

İstenilen şekillerin çizimi bitince, menüye dönülerek bunların şekiller tablosunu simgeleyen bir ad altında diskete kaydedilmesi sağlanır. Her tabloda şekiller artan bir sırayla numaralandırılır.

Bu programın, Siprel, Apple ve bunlara uyum-

SHAPE EDITOR



A% (1000) yer değiştirme vektörlerini içerir. C%(500) kodlanmış vektör çiftlerini içerir

Yer değiştirmeler tablosu 25000 adresinden itibaren belleğe alınacaktır; ancak kullanıcı başka bir seçim de yapabilir.

Birinci şekil

Şekil, ekranda, (X,Y) olarak girilen koordinatlardan başlayarak çizilecektir

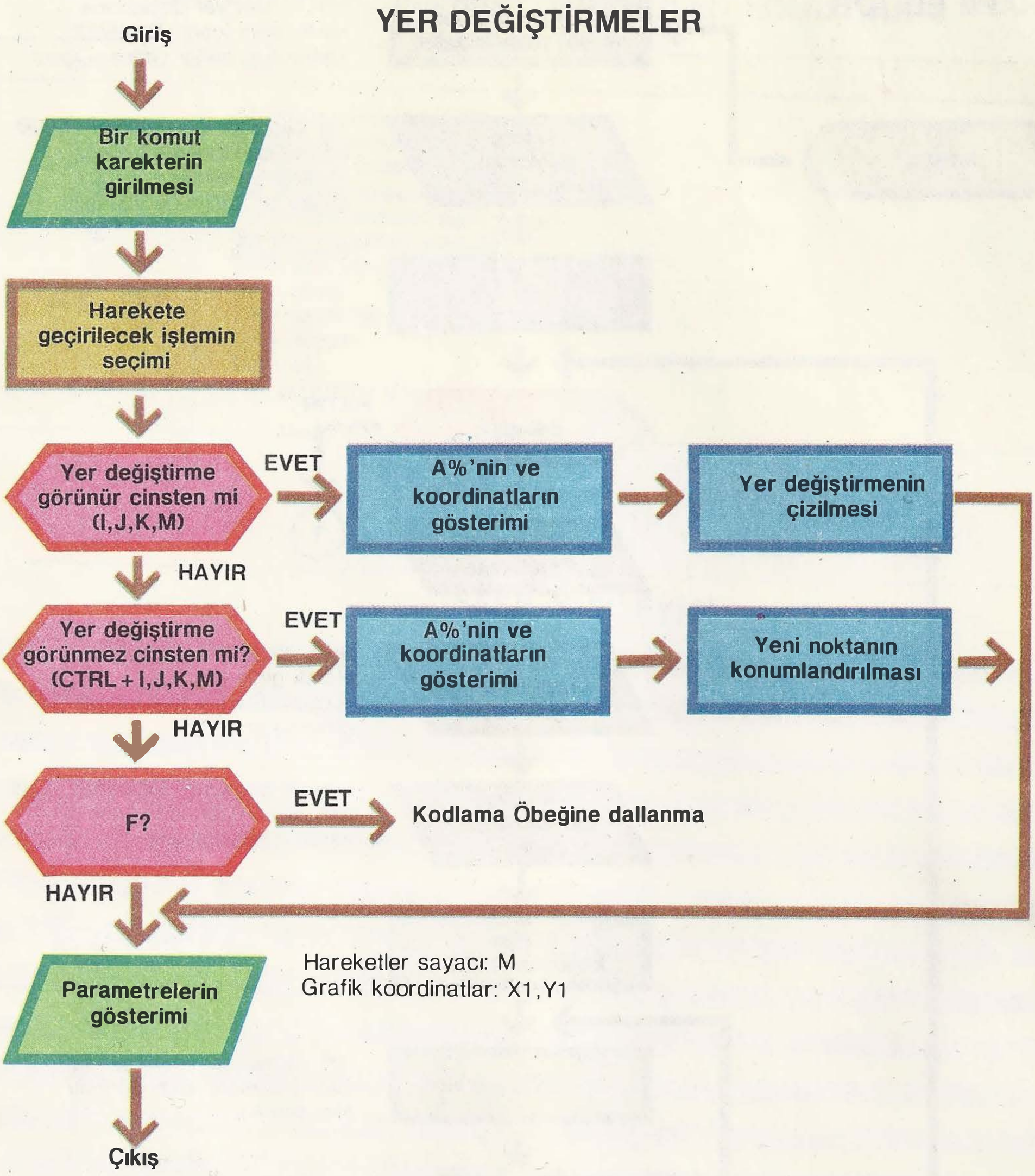
Her bir birim yer değiştirme için ekran üzerinde yanacak ekran noktası sayısı

Yer değiştirme komutlarının alıp işlenmesi. Bkz. ayrıntılı akış şeması.

Yer değiştirme vektörleri biçiminde belleğe alınmak üzere hareketlerin ayrıştırılması (bkz. ayrıntılı akış şeması)

— Diskete kayıt
— Şeklin görüntüsü
— Tablonun silinmesi
— Başka tablo

YER DEĞİŞTİRMELER

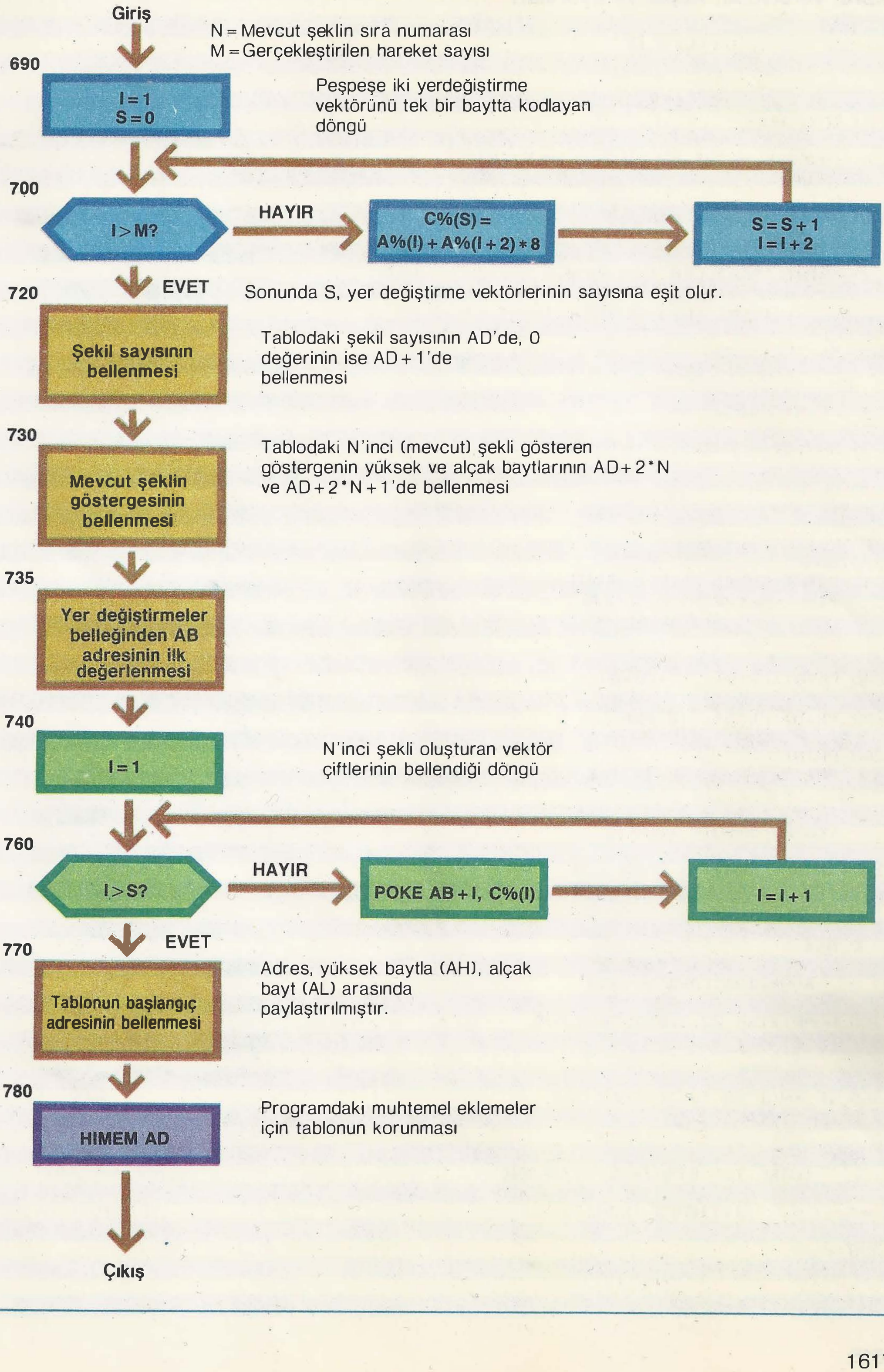


Hareketler sayacı: M
Grafik koordinatlar: X1,Y1

YER DEĞİŞTİRMELERİN KONTROL KARAKTERLERİ

Karakter	Nitelik	Yön	A%'deki kodlanması
CTRL + 1	Görünmez	Yukarı	0
CTRL + K	Görünmez	Sağa	1
CTRL + M	Görünmez	Aşağıya	2
CTRL + J	Görünmez	Sola	3
I	Görünür	Yukarı	4
K	Görünür	Sağa	5
M	Görünür	Aşağı	6
J	Görünür	Sola	7

KODLAMA


Giriş
N = Mevcut şeklin sıra numarası
M = Gerçekleştirilen hareket sayısı
690
I = 1
S = 0
Peşpeşe iki yerdeğiştirme vektörünü tek bir baytta kodlayan döngü
700
I > M?
HAYIR
C%(S) = A%(I) + A%(I + 2) * 8
S = S + 1
I = I + 2
720
EVET
Sonunda S, yer değiştirme vektörlerinin sayısına eşit olur.
Şekil sayısının bellenmesi
Tablodaki şekil sayısının AD'de, 0 değerinin ise AD + 1'de bellenmesi
730
Mevcut şeklin göstergesinin bellenmesi
Tablodaki N'inci (mevcut) şekli gösteren göstergenin yüksek ve alçak baytlarının AD + 2*N ve AD + 2*N + 1'de bellenmesi
735
Yer değiştirmeler belleğinden AB adresinin ilk değerlenmesi
740
I = 1
N'inci şekli oluşturan vektör çiftlerinin bellendiği döngü
760
I > S?
HAYIR
POKE AB + I, C%(I)
I = I + 1
770
EVET
Tablonun başlangıç adresinin bellenmesi
Adres, yüksek baytla (AH), alçak bayt (AL) arasında paylaştırılmıştır.
780
HIMEM AD
Programdaki muhtemel eklemeler için tablonun korunması
Çıkış

SHAPE EDITOR

Siprel versiyonu, Apple ve uyarlıları

```
 70 REM 30 SEKLE
 80 REM KADAR ICEREBILIR
 90 :

100 DIM A%(1000),C%(500)
110 TEXT:HOME

120 INVERSE:PRINT "SEKIL BICIMLEYICI"
    :NORMAL

130 PRINT:PRINT

150 ADD=25000
160 PRINT "SEKILLER TABLOSUNUN
        ADRESI=25000":PRINT "DEGISTIRMEK
        ISTERMISINIZ (E/H)?;:INPUT Z$

170 IF LEFT$(Z$,1)="O" THEN PRINT
        :INPUT "YENI DEGER ?";AD

180 PRINT:PRINT "OK!";AD

190 AH=INT(AD/256):AL=AD-AH*256

200 BL=62:BH=0:BB=62


210 N=1
220 HOME
230 PRINT:PRINT "SEKIL NUMARASI?"
    :INVERSE:PRINT N:NORMAL


240 PRINT:PRINT "<1>DEVAM"

250 PRINT:PRINT "<2>DUR"

260 PRINT
270 PRINT:INPUT "SECIM";A$

280 IF VAL(A$)=2 THEN N=N-1
    :BB=BB-5:GOTO 1000

290 IF VAL (A$)<>1 THEN 220
300 HGR
310 HOME
320 S=1
330 M=1
340 VTAB 22:INPUT "BASLANGIC
      KOORDINATLARI ?";X,Y

350 HOME:VTAB 22:INPUT "BUYUTME
      KATSAYISI (0-10) ?";IN


360 IF IN<1 OR IN>10 THEN 350
370 HOME:VTAB 21

380 PRINT "<I>,<J>,<K>,<M> CIZIM ICIN
      (CTRL ILE BIRLIKTE CIZMEDEN
          YER DEGISTIRMEK ICIN),(F)
            BITIRMEK ICIN.";

390 :
400 REM BASLANGIC NOKTASI
```

```
420 HCOLOR=3
430 HPLOT X,Y
440 X1=X:Y1=Y

450 :

460 REM HAREKETLER
470 :

480 HTAB 1:VTAB 24:PRINT "HAREKETLER"
    :INVERSE:PRINT M:NORMAL



500 IF A$="I" THEN A%(M)=4
    :Y=Y-IN:GOTO 600

510 IF A$="M" THEN A%(M)=6
    :Y=Y+IN:GOTO 600

520 IF A$="J" THEN A%(M)=7
    :X=X-IN:GOTO 600

530 IF A$="K" THEN A%(M)=5
    :X=X+IN:GOTO 600

540 IF A$=CHR$(9) THEN A%(M)=0
    :Y=Y-IN:GOTO 610

550 IF A$=CHR$(13) THEN A%(M)=2
    :Y=Y+IN:GOTO 610

560 IF A$=CHR$(10) THEN A%(M)=3
    :X=X-IN:GOTO 610

570 IF A$=CHR$(11) THEN A%(M)=1
    :X=X+IN:GOTO 610

580 IF A$="F" THEN 670
590 GOTO 430
600 HPLOT X1,Y1 TO X,Y:GOTO 630

610 IF IN=1 THEN 630
620 HPLOT X,Y
630 M=M+1
640 X1=X:Y1=Y

650 GOTO 460
660 :

670 REM KODLAMA
680 :

690 FOR I=1 TO M STEP 2
700 C%(S)=A%(I)+A%(I+1)*8
    :S=S+1
710 NEXT
720 POKE AD,N:POKE AD+1,0

730 POKE AD+2*N,BL
    :POKE AD+2*N+1,BH
735 AB=AD+BB-1
740 FOR I=1 TO S
```

```
750 POKE AB+I,C%(I)
760 NEXT
770 POKE 232,AL:POKE 233,AH

780 HIMEM AD
790 :

800 REM MENU 2
810 :

820 HOME:VTAB 21

830 PRINT "SEKIL NUMARASI"
    : PRINT
840 PRINT "(1)SILIYORUM, (2)
      DIGER, (3)DUR"
850 PRINT
860 PRINT SECIM":GET Z$

870 Z=VAL(Z$)
    :IF Z(1 OR Z)3 THEN 900
880 ON Z GOTO 900,910,1000
890 GOTO 910
900 N=N-1:BB=BB-S

910 N=N+1:BB=BB+S

920 BH=INT(BB/256):BL=BB-BH*256

930 FOR I=1 TO S:C%(I)=0:NEXT


940 FOR I= 1 TO M:A%(I)=0:NEXT


950 TEXT:HOME:PRINT "SU NUMARALI
      SEKLI YAP :";N

960 VTAB 5:GOTO 230

970 :

980 REM MENU 1
990 :

1000 TEXT:HOME:VTAB 5


1010 FI$="E": IF N=1 THEN FI$="A"

1020 PRINT:PRINT "SEKLI CIZDINIZ"
     :N:"SEKIL":FI$

1030 PRINT:PRINT "TABLONUN
       ADRESI:";AD

1040 BT=BB+S
1050 IF N=0 THEN BT=0
1060 PRINT "UZUNLUK :";BT
1070 PRINT:PRINT

1080 PRINT "(1) TABLONUN DISKTE
       SAKLANMASI"
1090 PRINT "(2)SIRAYLA
     GORUNTULEME"
1100 PRINT "(3)TABLONUN
     SILINMESI"
1110 PRINT "(4)BASKA TABLO"
1120 PRINT "(5)SON"
1130 PRINT
1140 PRINT "SECINIZ:"GET Z$
1150 Z=VAL(Z$)
```

```
1160 IF Z<1 OR Z>5 THEN 1000
1170 ON Z GOTO 1210,1260,1490,1190
     ,1160
1180 HOME:END

1190 CLEAR:GOTO 100

1200 :

1210 REM DISKTE SAKLAMA
1220 :

1230 IF N=0 THEN HOME:VTAB 12:HTAB 10
     :FLASH:PRINT "SEKIL YOK":NORMAL:GET Z$:GOTO 1000



1240 PRINT:INPUT "HANGI DLA";A$

1250 PRINT CHR$(4)"BSAVE";A$",A";AD;
     ",L";BB+S
1260 GOTO 1000
1270 :

1280 REM TABLONUN GOSTERILMESI
1290 :

1300 FOR I=1 TO N
1310 RO=0:SC=1

1320 HGR:HCOLOR=3:ROT=RO:SCALE=SC



1330 HOME
1340 VTAB 22:PRINT "SEKIL NO"
     ;I;"OLCEK=";SC;"ROT=";RO

1350 VTAB 23:PRINT "DEGISTIRMEK ICIN:<S>
      OLCEK,<R> DON,<F> SEKIL,
     .(<CTRL><TUS> ILE GERI
      DONULUR)";
1360 DRAW I AT 140,80
1370 GET A$
1380 IF A$="S" THEN SC=SC+1
     : IF SC>255 THEN SC=255
1390 IF A$="R" THEN RO=RO+1
     :IF RO>255 THEN RO=255
1400 IF A$=CHR$(18) THEN RO=RO-1
     :IF RO<0 THEN RO=0

1410 IF A$=CHR$(19) THEN SC=SC-1
     :IF SC<1 THEN SC=1

1420 IF A$="F" THEN 1440
1430 GOTO 1320
1440 NEXT
1450 GOTO 1000
1460 :

1470 REM TABLONUN SILINMESI
1480 :

1490 PRINT:PRINT "GERCEK MI (E/H) ?"
     ;GET Z$

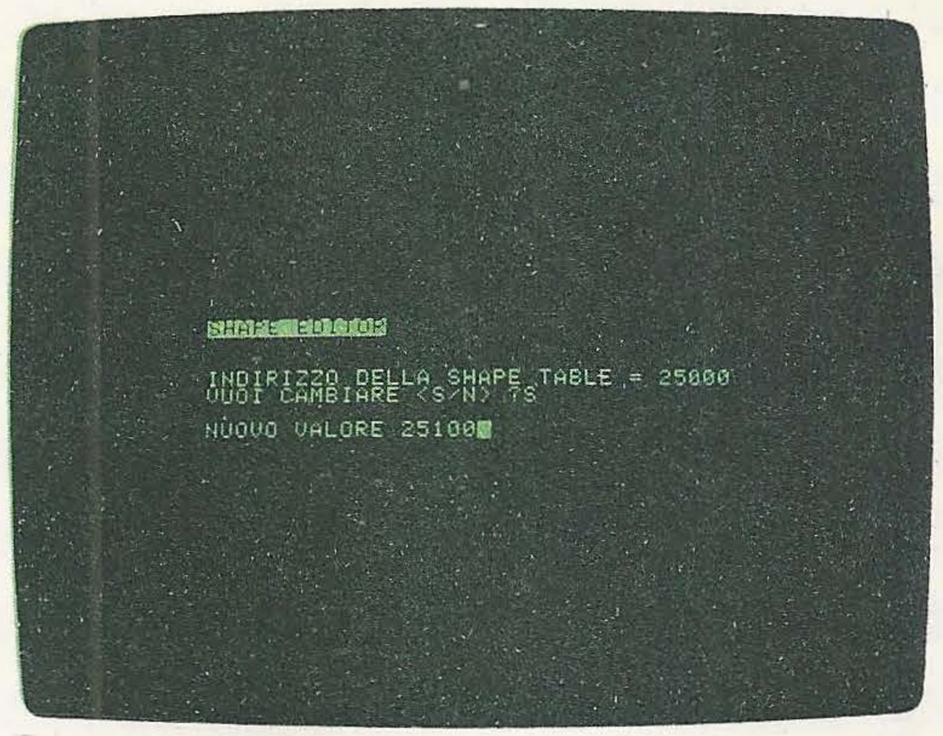

1500 IF Z$="O" THEN CLEAR:GOTO 100

1510 GOTO 1000
```

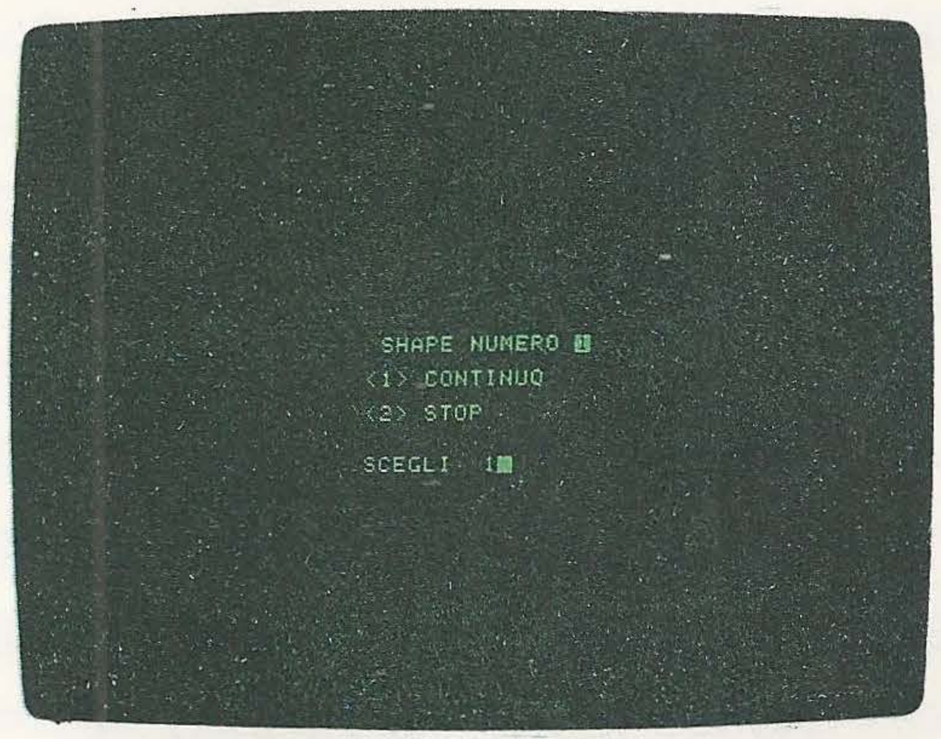
SHAPE EDITOR

Bu ekranlar, SHAPE EDITOR programının çalışması sırasında görünür

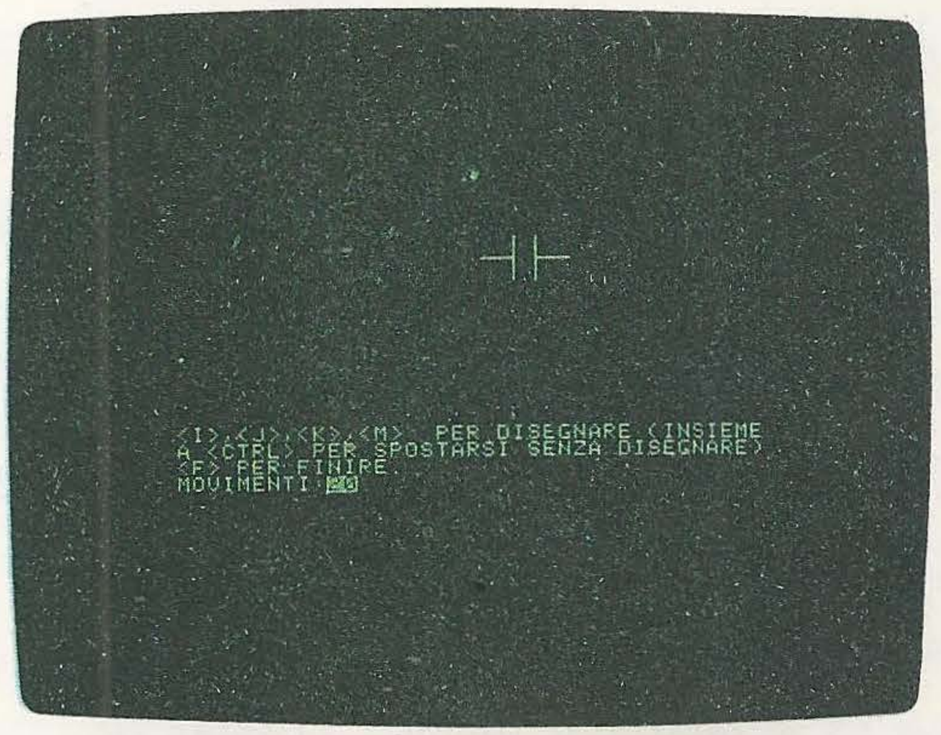
Program yaratılacak tablonun, bellekteki başlangıç adresini hesaplamakla işe başlar. Kullanıcı, yine de, bu adresi değiştirebilir.



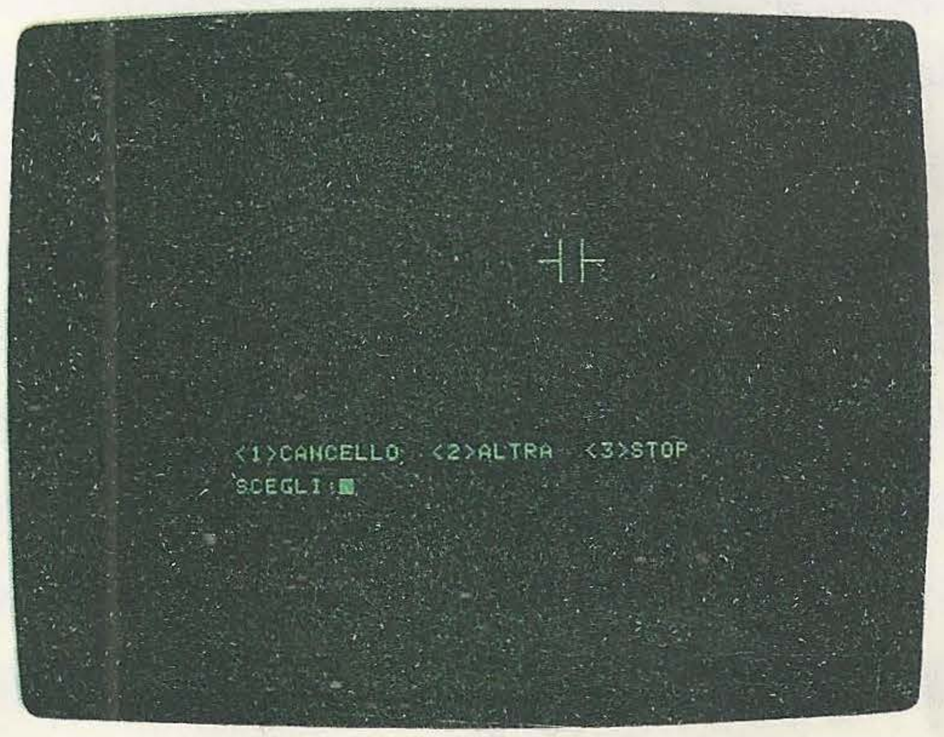
Başlangıçta tablo boştur. Program buraya, işlediği şeklin numarasını yazar.



Parametreler (başlangıç koordinatları, büyütme katsayısı) girildiğinde, program çizim moduna geçerek, ekranda komutlar menüsünü görüntüler. Burada bir kondansatör simgesi çizilmiştir.



Şekil bittiğinden, bir yenisine geçilebilir. Tablo tamamlanınca, disk üzerine kaydedilir.



SHAPE EDITOR PROGRAMININ ANA DEĞİŞKENLERİ

A%	Kodlanmış yer değiştirmeleri içeren dizey
C%	Kodlanmış çizim vektör çiftlerini içeren dizey
AD	Şekiller tablosunun bellekteki başlangıç adresi (alçak kısım=AL, yüksek kısım= AH)
BB	Yeni şeklin başlangıç adresi (alçak kısım= BL, yüksek kısım=BH)
IN	Büyütme katsayısı
M	Şekli oluşturan hareket sayısı
N	Şekil numarası
RO	Görüntülenen bir şeklin döndürülmüş olduğunu belirten simge
SC	Görüntülenen bir şeklin ölçeğini belirten simge
X,Y	
X1, Y1	Çizim koordinatları

lu bilgisayarlar için düzenlenmiş versiyonu, 1618 ve 1621. sayfalarda yer almaktadır. Kullanılan değişkenler, bu sayfada yer alan tabloda gruplandırılmıştır.

Şekil Yükleyicisi. Şekiller tablosunun disket üzerinde saklanmasının, ancak onu tekrar alıp yararlanacak bir program varsa, anlamı olabilir.

Bir tablo içinde gruplanmış belli sayıda şeklin tanımlanması, gerçekte bir simgeler menüsüne sahip olma anlamına gelir. Bu simgeler değişik biçimlerde biraraya getirilerek çok karmaşık şekiller oluşturulabilir. Öyleyse, Shape Editor programının işlevi, şekilleri tablodan okuyarak, bu elemanları ekranda birleştirme işlemlerini yürütebilecek bir ikinci programla tamamlanmış olur (bkz. bir sonraki sayfadaki akış şeması).

Kullanıcı bir başlangıç menüsünden seçimini yapar. Cevap olarak programın çağrılan kısmı harekete geçer. 300 alt-programı (akış şeması 2, bir sonraki sayfa), disket üzerindeki tabloyu belleğe yükler. Bunun sonucunda ekranda menü görüntülenir.

900 alt-programıyla (akış şeması 3, s.1625), tablodaki şekillerden biri ekrana gelir. Bu işlemlerden sonra, program menüyü yeniden görüntüler. Bundan sonra, 3 seçeneği ile bir şekil yaratılması aşamasına girmek mümkündür (500 alt-programı, s.1625). 500 alt-programı (akış şeması 4), çizim modunu seçer ve çizilecek şeklin ne olacağıyla ekranın hangi noktasından başlayacağını sorar.

Bu parametreler girildiğinde istenilen şekil ekrana gelir ve program 580 giriş öbeğine ve şekil yaratma komutlarının yürütülmesine geçer (bkz. tablo, s. 1626).

Programın tamamlayıcı iki işlevi de, disket üzerindeki dağarcığın (fihrist) öğrenilmesi ve yaratılan şeklin saklanmasıdır.

Şekil değiştirme programlarının dökümü s.1626 ve 1630'da verilmiştir (kullanılan değişkenler, s. 1630'daki tablo 4'de özetlenmiştir). Bu arada dökümü verilen iki programın Siprel bilgisayarı üzerinde geliştirildiğini belirtelim. Bu biçimiyle yalnızca Apple ailesi ve buna uyumlu bilgisayarlar üzerinde çalışabilirler. Bunları diğer makinelere aktarabilmek için, daha önce anlatılan aktarma kurallarına başvurmak gerekir.

Grafik Figürlerin Yönetimi İçin BASIC Komutları

Bellekte bulunan bir şekil, bazı makinelerde bir veri kümesi olarak, yüksek düzeyli komutlar aracılığıyla işlenebilir. Örneğin:

DRAW	: Şeklin çizilmesi
XDRAW	: Silinmesi
ROT	: Döndürme
SCALE	: Ölçek katsayısının değiştirilmesi

DRAW. Bu komutun sözdizimi şöyledir:

DRAW N AT X,Y

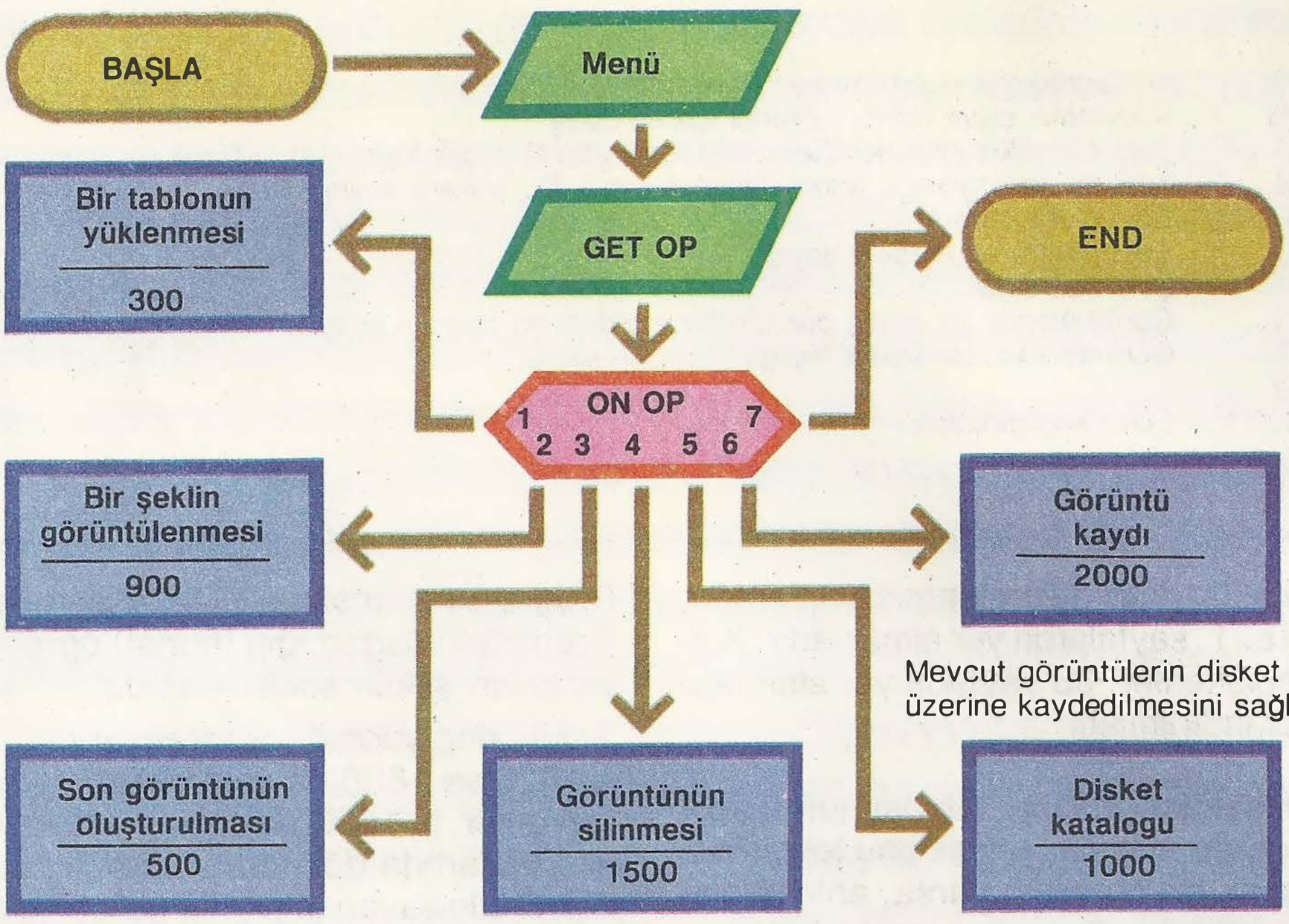
Burada,

N : Görüntülenecek şeklin numarası,
X,Y : Çizime başlanacak noktanın koordinatlarıdır.

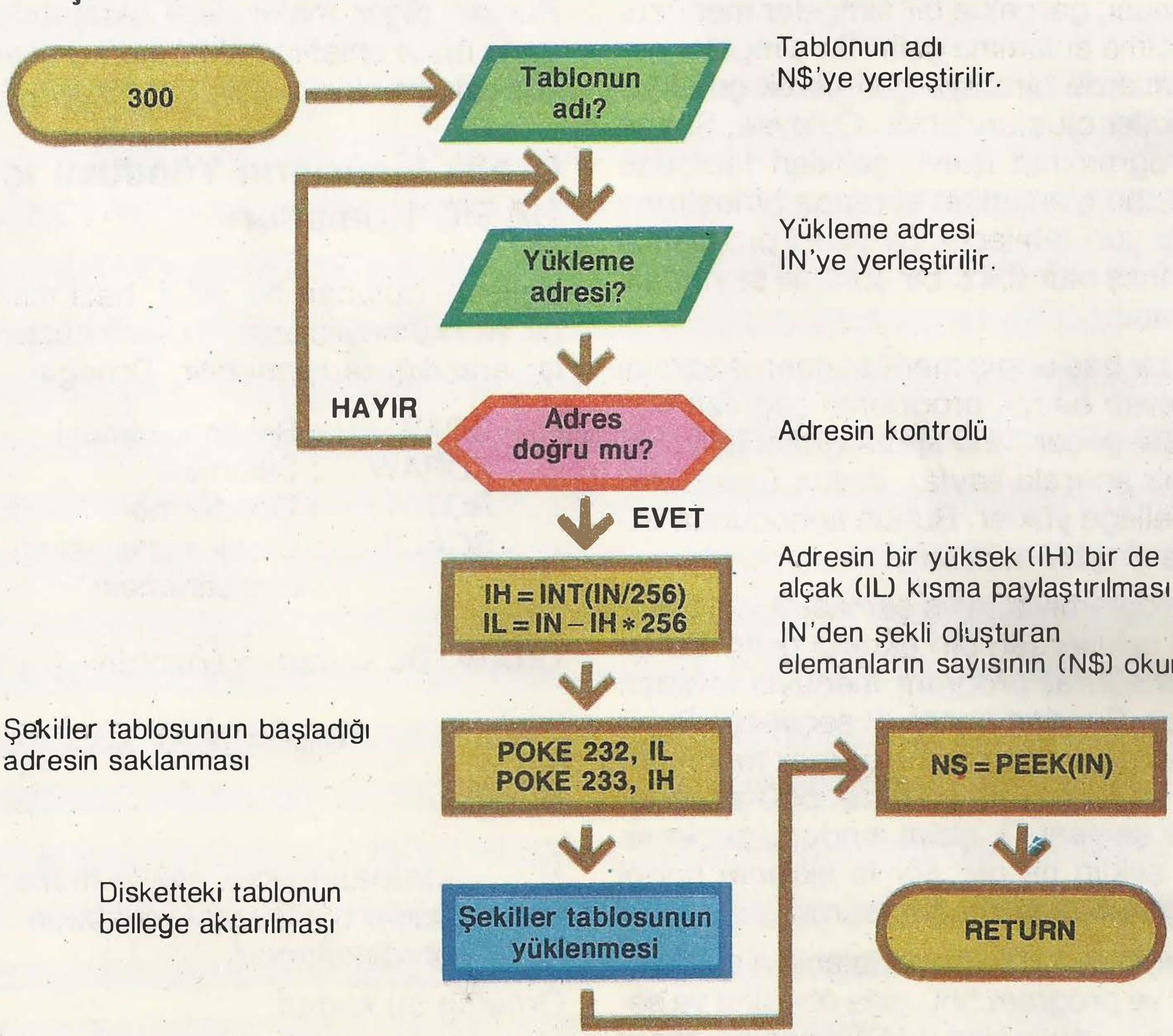
Örneğin şu komut:

DRAW 2 AT 30,50

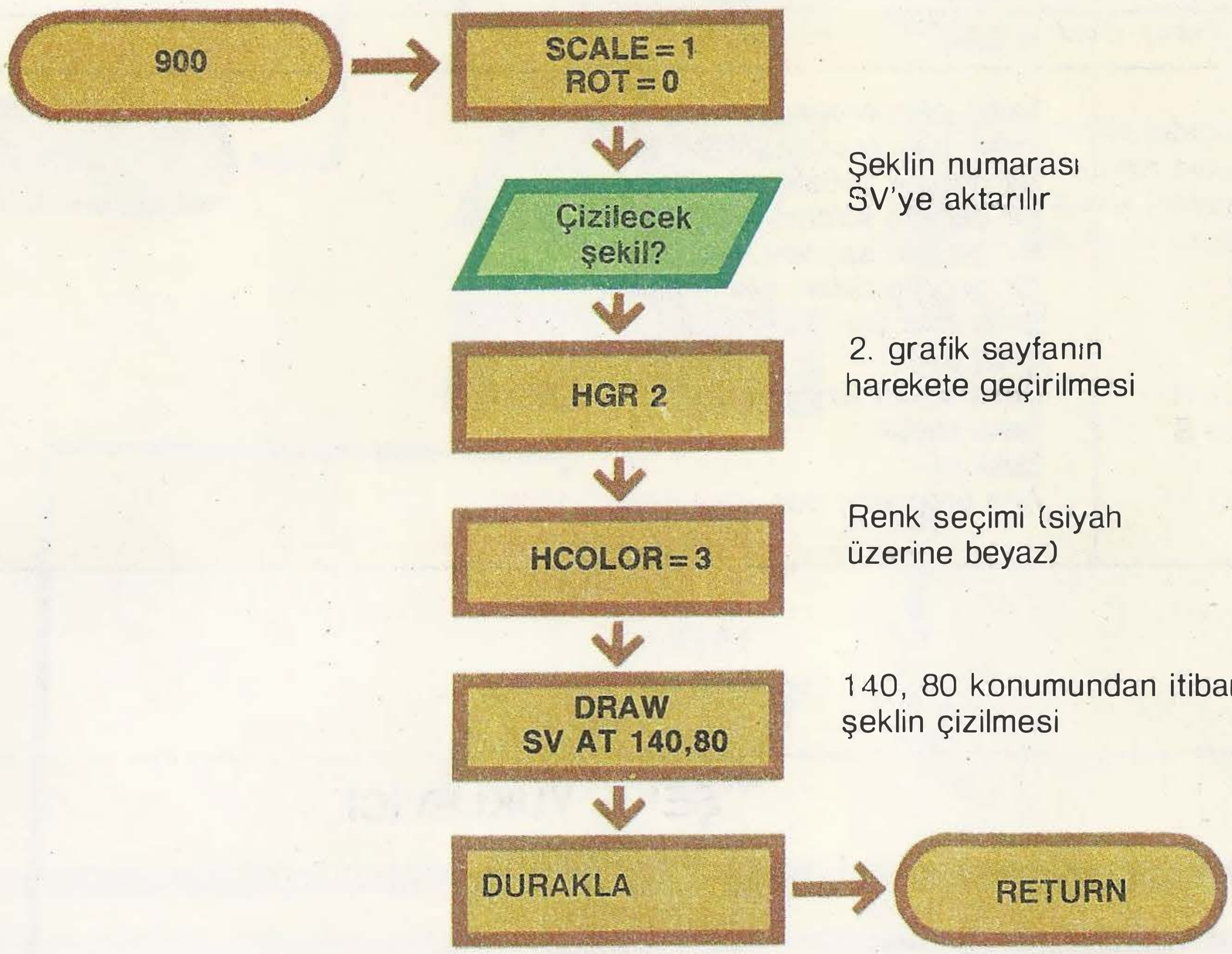
SHAPE LOADER (ŞEKİL YÜKLEYİCİ)



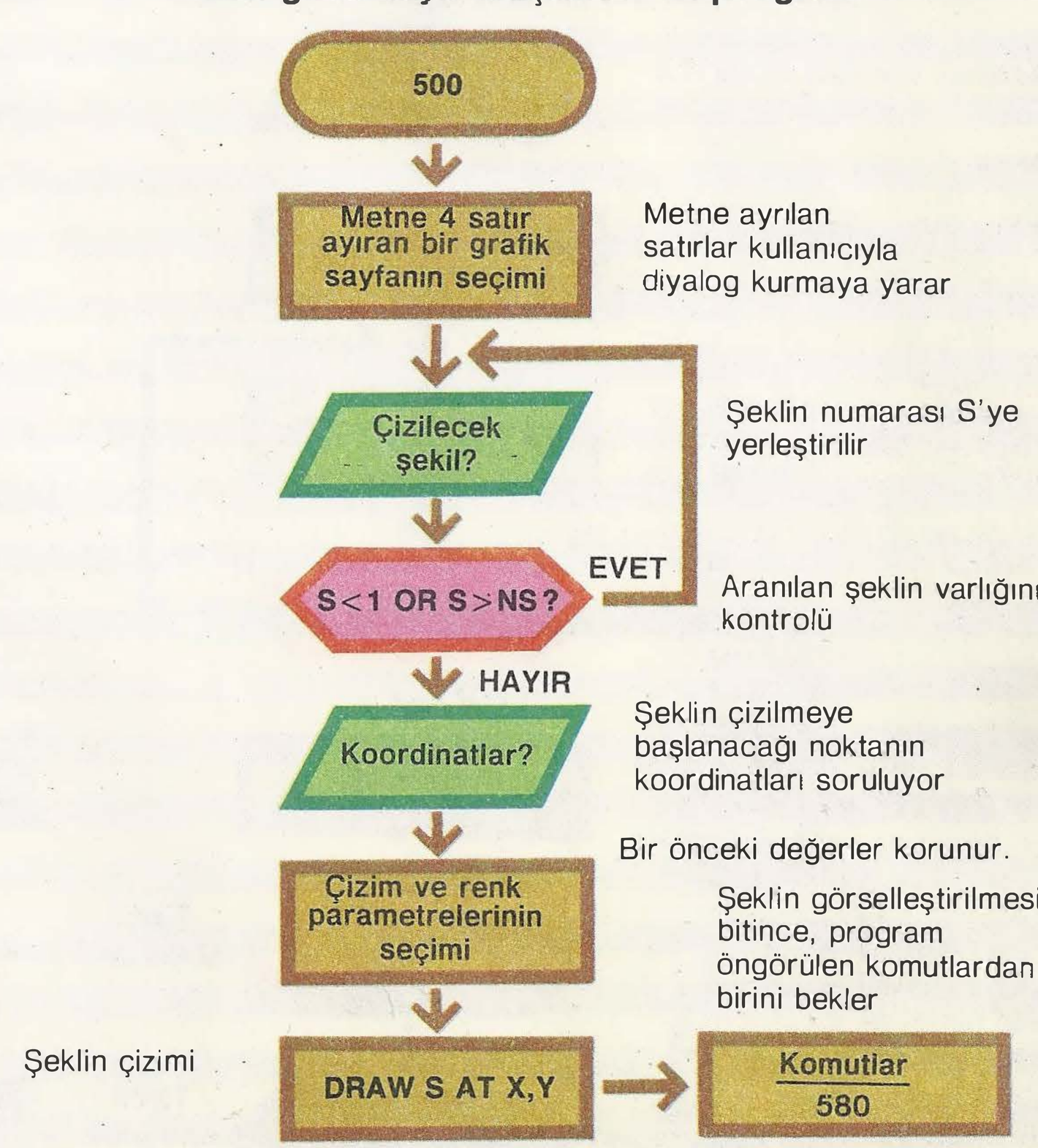
ŞEKİLLER TABLOSUNU YÜKLEYEN ALT-PROGRAM



ŞEKLİ GÖRSELLEŞTİREN ALT-PROGRAM



Son görüntüyü oluşturan alt-program



580 ALT-PROGRAMINDA ÖNGÖRÜLEN KOMUTLAR

Komut	Anlamı
/	Metin penceresinden vazgeç
?	Metin penceresini görselleştir
I	Bir geçişte üstteki şekli değiştir
J	Bir geçişte soldaki şekli değiştir
K	Bir geçişte sağdaki şekli değiştir
M	Bir geçişte alttaki şekli değiştir
R	Şekli saat yönünde 90 döndür.
S	Şekli büyüt
CTRL+R	Şekli saatin tersi yönde 90 döndür
CTRL+S	Şekli küçült
C	Şekli sil
ESC	Ana programa dön

ŞEKİL YÜKLEYİCİ

```
10 REM ************************
20 REM *                      *
30 REM * SEKIL YUKLEYICI      *
40 REM *                      *
50 REM ************************
60 :

70 REM MENU
80 :

90 TEXT:HOME

100 HI=PEEK(135)+PEEK(116)*256

110 HTAB 18:INVERSE:PRINT "MENU"

120 NORMAL:VTAB 6

130 PRINT "BIR SEKILLER TABLOSUNUN
    YUKLENMESI.....1":PRINT

140 PRINT "BIR SEKLIN GORUNTULENMESI
    ..................2":PRINT

150 PRINT "SON GORUNTUNUN OLUSTURULMASI
    ..................3":PRINT
160 PRINT "GORUNTUNUN SILINMESI
    ..................4":PRINT
170 PRINT "DISK KATALOGU
    ..................5":PRINT

180 PRINT "GORUNTUNUN KAYDI
    ..................6":PRINT

190 PRINT "SON ..............
    ..................7":PRINT
200 VTAB 21

210 INPUT "HANGI OPSIYONU
    SECIYORSUNUZ";OP$
```

```
220 OP=VAL(OP$):IF OP<1 OR OP>7
    THEN 90
230 ON OP GOSUB 300,900,500,1500,1000
    ,2000
240 IF OP<>7 THEN 90
250 HOME:END

300 :

310 REM BIR SEKILLER TABLOSUNUN
        YUKLENMESI
320 :
330 TEXT:HOME

340 INPUT "SEKILLER TABLOSUNUN ADI
    ?";N$
345 IF ASC(LEFT$(N$,1))<65 OR
    ASC(LEFT$(N$,1))>90 THEN 330
350 PRINT:PRINT

360 PRINT "YUKLEME ADRESI :"
    :PRINT "(24576-";HI;")" 370 VTAB
     4:HTAB 26:PRINT SPC(14)


380 VTAB 4:HTAB 26:INPUT " ";IN$

390 IN=VAL(IN$)
400 IF IN<24576 OR IN>HI THEN 370

410 IH=INT(IN/256):IL=IN-IH*256

420 POKE 232,IL:POKE 233,IH

425 VTAB 16:INVERSE:PRINT "DISKI YUKLEYIN VE
     <RETURN> YAZIN":NORMAL
    :GET R$


427 PRINT CHR$(4)
430 PRINT CHR$(4);"BLOAD";N$;",A";IN

435 NS=PEEK(IN)
440 RETURN
500 :

505 REM GORUNTUNUN OLUSTURULMASI
510 :

515 IF NS=0 THEN HOME:VTAB 12:FLASH
    :PRINT "BELLEKTE TABLO
    YOK":NORMAL:GET A$:RETURN


520 POKE-16297,0:POKE-16300,0
    :POKE-16301,0:POKE-16304,0
    :POKE 230,32


525 HOME:VTAB 21

530 PRINT "GORUNTULENECEK SEKIL(1-"
    ;NS;"):INPUT "?";S$
535 IF S$="" THEN RETURN

540 S=VAL(S$)
545 IF S<1 OR S>NS THEN 525
```

```
546 VTAB 23:INPUT "KOORDINATLAR (X,Y)"
    ;X$,Y$:X=VAL(X$):Y=VAL(Y$)


547 IF X<0 OR X>279 OR Y<0 OR Y>191
    THEN 546
550 RO=0:SC=1

555 X1=X:Y1=Y:R1=RO:S1=SC


557 HCOLOR=3
560 SCALE=SC:TOR=RO

565 DRAW S AT X,Y
570 HOME:VTAB 23

575 INVERSE:PRINT "KOMUTLAR:I,J,K,
     M,R,R S, S/,? C ESC";
    :NORMAL

580 HTAB 1:VTAB 21:PRINT SPC(79)

585 HTAB 1:VTAB 21

590 PRINT "SEKIL=";:INVERSE:PRINT S
    :NORMAL

595 PRINT "OLCEK=";:INVERSE:PRINT
    SC;:NORMAL

600 PRINT "DON=";:INVERSE:PRINT RO;
    :NORMAL

605 PRINT "X= ";:INVERSE:PRINT X
    :NORMAL

610 PRINT "Y= ";:INVERSE:PRINT Y;
    :NORMAL

615 VTAB 1:HTAB 1:GET T$

620 IF T$="/" THEN POKE -16302,0
625 IF T$="?" THEN POKE -16301,0
630 IF T$="I" THEN Y=Y-1
    : IF Y<0 THEN Y=191
635 IF T$="J" THEN X=X-1
    : IF X<0 THEN X=279
640 IF T$="K" THEN X=X+1
    : IF X>279 THEN X=0
645 IF T$="M" THEN Y=Y+1
    : IF Y>191 THEN Y=0
650 IF T$="R" THEN RO=RO+1
    : IF RO>255 THEN RO=255
655 IF T$=CHR$(18) THEN RO=RO-1
    : IF RO<0 THEN RO=0

660 IF T$="S" THEN SC=SC+1
    :IF SC>255 THEN SC=255
665 IF T$="C" THEN GOSUB 1800
670 IF T$=CHR$(19) THEN SC=SC-1
```

```
675 IF T$=CHR$(27) THEN RETURN
677 HCOLOR=0
680 SCALE=S1:ROT=R1
685 DRAW S AT X1,Y1
687 HCOLOR=3
690 SCALE=SC:ROT=R0:DRAW S AT X,Y
695 S1=SC:R1=R0:X1=X:Y1=Y
700 GOTO 580

900 :
905 REM BIR SEKLIN CIZILMESI
910 :
915 IF N$=0 THEN HOME:VTAB 12:FLASH
    :PRINT "TABLO BELLEKTE
     YOK":NORMAL:GET A$:RETURN

920 TEXT:HOME
925 SCALE=1:ROT=0
930 VTAB 12:HTAB 1
935 PRINT "GORUNTULENECEK SEKIL (1-"
    ;N$;")";INPUT SV$:SV=VAL(SV$)
940 IF SV<1 OR SV>NS THEN 920
945 HGR2:HCOLOR=3
950 DRAW SV AT 140,80
955 GET R$
960 RETURN

1000 :
1010 REM KATLOG
1020 :
1030 HOME:PRINT CHR$(4);"KATALOG"
1040 GET A$:RETURN
1075 PRINT  CHR$(4)

1500 :
1510 REM SILME
1520 :
1530 HOME:VTAB 12:INPUT "EMINMISINIZ ?(E/H)
     ?";R$
1540 IF R$<>"O" AND R$<>"N" THEN 1530
1550 IF R$="O" THEN HGR
1560 RETURN

1600 :
1810 REM
1820 :
1830 XO=X:YO=Y:X=140:Y=80
```

```
1835 VTAB 23:HTAB 36:PRINT "O"

1840 HCOLOR=3:HPLOT X,Y

1850 VTAB 1:HTAB 1:GET A$

1855 HCOLOR=0:HPLOT X,Y

1860 IF A$="I" THEN Y=Y-1
     :IF Y<0 THEN Y=191
1870 IF A$="M" THEN Y=Y+1
     : IF Y>191 THEN Y=0
1880 IF A$="J" THEN X=X-1
     :IF X<0 THEN X=279
1890 IF A$="K" THEN X=X+1
     :IF X>279 THEN X=0
1900 IF A$="C" THEN 1920
1910 GOTO 1840
1920 HCOLOR=3:X=XO:Y=YO

1930 VTAB 23:INVERSE

1940 PRINT "KOMUTLAR: I,J,K,M,R R
     S,  S/,? C ESC";NORMAL

1950 RETURN
2000 :

2010 REM SEKLIN SAKLANMASI
2020 :

2030 HOME:VTAB 12

2040 INPUT "HANGI AD ILE ";NM$
2050 IF ASC(LEFT$(NM$,1))<65 OR
     .ASC(LEFT$(NM$,1))>90 THEN 2030

2060 VTAB 14:PRINT "GUZEL !";NM$
     ",PTC"
2070 VTAB 16:INVERSE:PRINT "BIR DISKETI
       YUKLEYIN VE <RETURN> YAZIN":NORMAL
     :GET R$


2080 PRINT CHR$(4)
2090 PRINT CHR$(4);"BSAVE";NM$;",PTC
     ,A$2000,L$1FFF"
2100 PRINT CHR$(4)
2110 RETURN
```

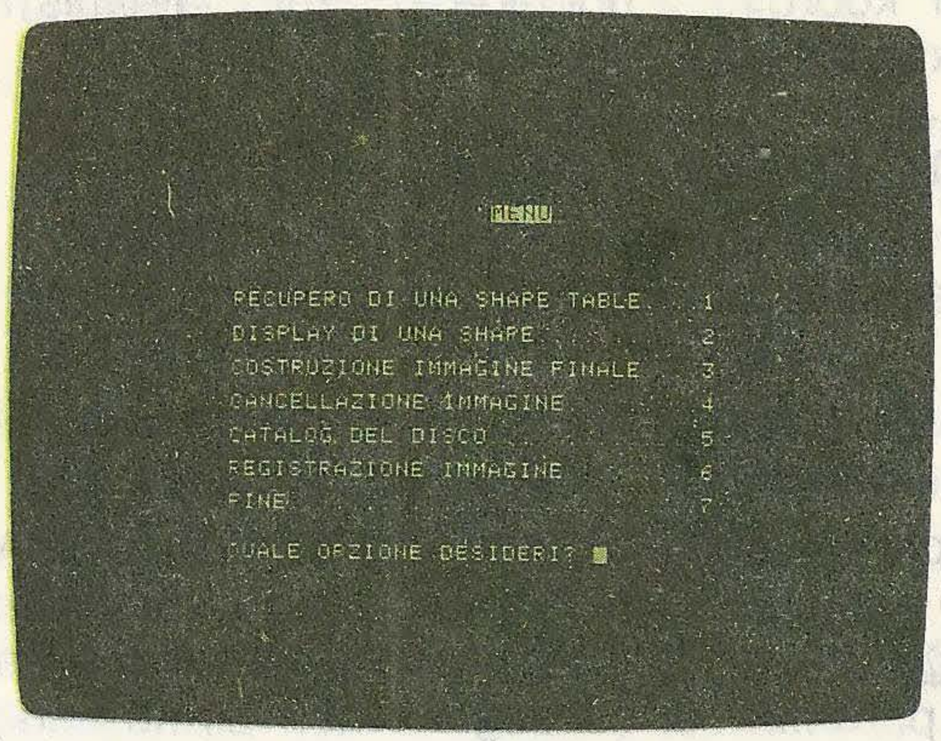
SHAPE LOADER PROGRAMINDA KULLANILAN DEĞİŞKENLER

HI	= Belleğin grafiğe ayrılan bölümünün sınırı
IN	= Şekiller tablosunun başlangıç adresi
N$,NM$	= Şekiller tablosunun adı
NS	= Şeklin tablodaki numarası
R1,RO =	= ROT (döndürme değerleri)
S$	= Mevcut şekil
S1,SC	= SCALE (ölçek) değerleri
X,Y,X0,Y0,X1,Y1	= Ekran koordinatları

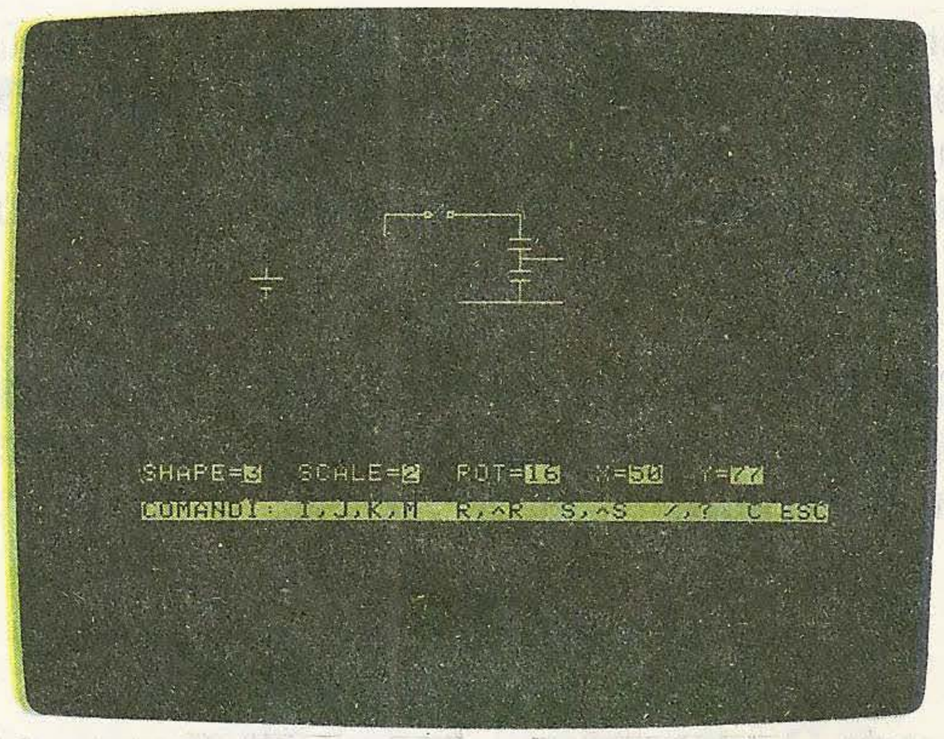
SHAPE LOADER PROGRAMININ UYGULAMA ÖRNEĞİ

Bu ekranlar, Shape Editor yardımıyla yaratılan ve saklanan basit şekillerden hareket ederek daha karmaşık bir şeklin oluşturulmasını sergiliyor.

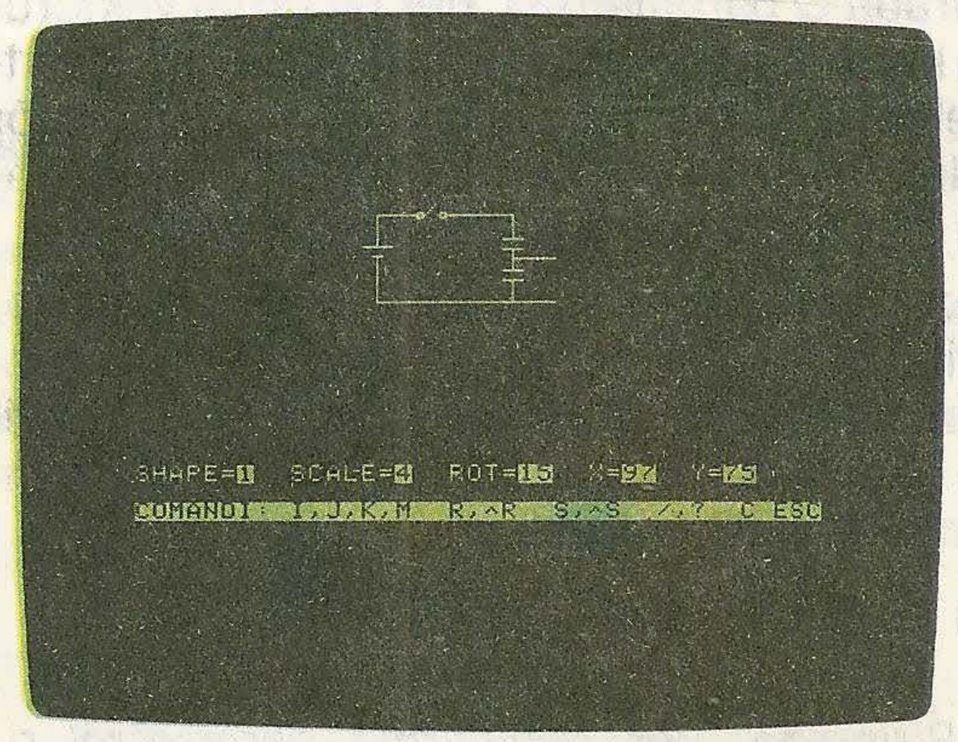
Shape Loader programı menüyü sunuyor. Kullanıcı, elektriksel simgeleri içeren bir şekiller tablosunun disketten alınıp belleğe yüklenmesini istiyor.



Son görüntüyü oluşturan alt-program seçilmiş ve yürütülmüştür. Ekranın altında sorular, cevaplar ve komutlar menüsü görünüyor. Ekranın ortasında oluşturulmakta olan çizim yer alıyor. Soldaki eleman ise, konumlandırılacak olan şekil.



Oluşturulan şekil tamamlanmış durumda. Kullanıcı, buna, istediği zaman yeniden çağırabilmek için özel bir ad verip, diskette saklamalıdır.



2 numaralı şeklin çizimine X = 30 ve Y = 50 noktasından başlanır. Eğer koordinatlar verilmemişse çizim son çağrılan noktadan başlar. Bu komutun DRAW N biçiminde kulanılabileceği ve bu durumda N şeklinin çiziminin DRAW veya HPLOT (konumlandırma için) komutunda kullanılan koordinatlardan başlayacağı anlamına gelir.

XDRAW. Bu komutun da biçimi aynıdır:

XDRAW N AT X,Y

X,Y koordinatlı noktada konumlandırılmış N numaralı şekli siler.

ROT. Sözdizimi çok basittir:

ROT M

Burada M, döndürme açısını belirten, genelde derece cinsinden bir sayısal değerdir.

M parametresinin geçerliliği, daha önce SCALE komutuyla tanımlanan ölçek katsayısının değerine bağlıdır. Eğer ölçek katsayısı 1 ise, M parametresi şu değerlerden birine sahip olur: 0 = 0°, 16 = 90°, 32 = 180°, 48 = 270°.

Buna karşılık, eğer ölçek katsayısı 4'den büyükse, eşit aralıklarla 64 değişik döndürme olanağı vardır (bu değerler kullanılan bilgisayara yakından bağlı olduğundan, kesin bir kural oluşturmuyorlar).

SCALE. Bu komut çizimin ölçeğini tanımladığından, bu gruptaki diğer komutların hepsinden önicek ullanılmalıdır. Sözdizimi şöyledir:

SCALE = Q

Burada Q parametresi, her bir çizim vektörüne karşılık gelen birim yer değiştirmeyi belirtmektedir. Örneğin, SCALE = 1 ile her vektör yalnız bir kez çizilirken, SCALE = 2 ile yanyana iki birim uzunluğunda çizilecektir.

Bazı makinelerde, bir dizi komut yardımıyla, birim yerdeğiştirmelerden hareketle çizim yapmak olanaklıdır. Anahtar sözcük genelde yine DRAW'dur. Sözdizimi ise yukardakine benzemekle birlikte farklıdır.

Bu komut iki biçim alabilir:

- Birincide, yer değiştirme parametreleri, DRAW komutuna bağlı bir zincir olarak içerilir.
- İkincide ise, bunlar bellekte kalıcı bir tabloda saklanır. Bu durumda komut, parametre olarak yalnızca istenilen şekle karşılık gelen numarayı içerir:

DRAW : ''Komutlar zinciri''
DRAW : N (N = Şeklin numarası)

Parametreler zincirinde, kullanılan yerine getirlecek birim hareketleri temsil eden bazı simgeleri açıklamak gerekir. Bu simgeler şu anlama gelir.

M DX,DY	İmlecin X,Y noktasından hareketle X + DX ve Y + DY noktalarına getirilmesi
J,X,Y	X,Y noktasına hareket
U DY	Yukarı doğru DY kadar hareket
L DX	Sola doğru DX kadar hareket
R DX	Sağa doğru DX kadar hareket
C	Çizgi renklerinin belirtilmesi. Eğer yoksa sistem kendiliğinden son DRAW komutunda kulanılan rengi alır.

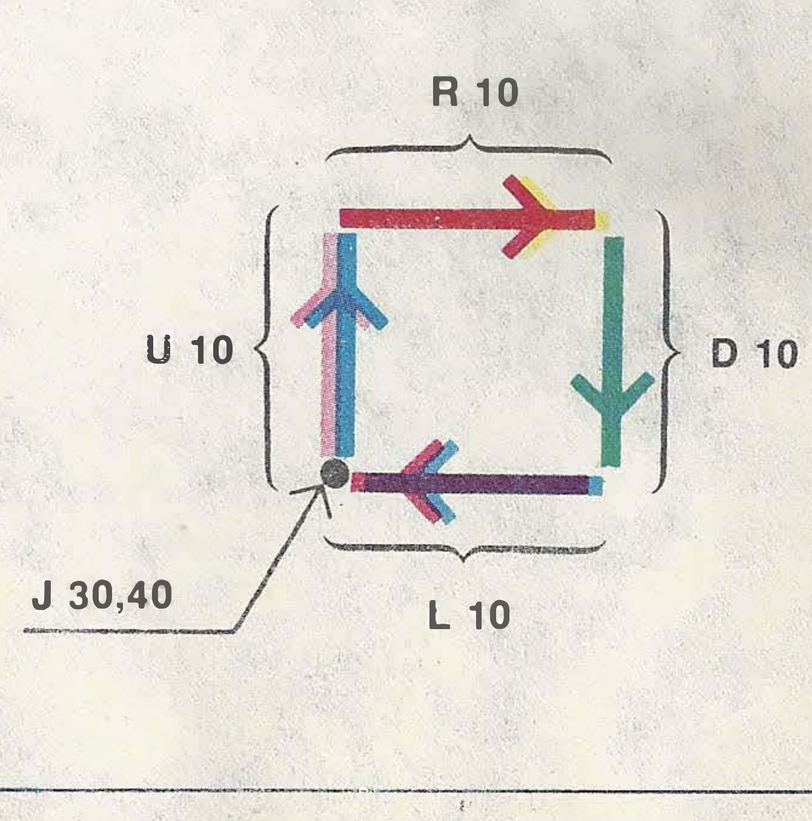
Yukarıdaki parametreler, Olivetti M20 sistemine tekabül ediyorlar. Başka makinelerde farklı olabilirler. Ancak eğer DRAW komutunun bu biçimi mevcutsa, mantık da aynıdır.

Aşağıdaki şekil, alt sol köşesi 30,40 noktasında bulunan ve bir kenarı 10 uzunluğundaki bir kareyi çizmek için gereken parametreler dizisini sergiliyor.

J 30,40 komutu görünür modda yürütüldüğü için, imlecin ilk bulunduğu nokta ile karenin sol alt köşesi arasında istenmeyen bir çizgi oluşacaktır. İstenmeyen bu çizginin görüntüden kaybolmasını sağlamak için parametreler dizisinin başına B harfini eklemek gerekir. Öyleyse bu komutun tam doğru biçiminin şöyle olması gerekir:

''BJ30, 40 U10 R10 L10''

'' J 30,40 U10 R10 D10 L10''



Komutların her biriyle birlikte, aşağıdaki seçeneklerden birini kullanmak mümkündür:

AND	Elde edilen renk belirtilen iki rengın mantıksal kesişimidir.
XOR.OR	Aynı ilke geçerli, ancak bu defa sözkonusu işlem, mantıksal bileşimdir (dışlayıcı veya rıormal VEYA).
NOT	Şekil için kullanılan rengin, fon için kullanılanın tümleşiği

olduğunu belirtir.

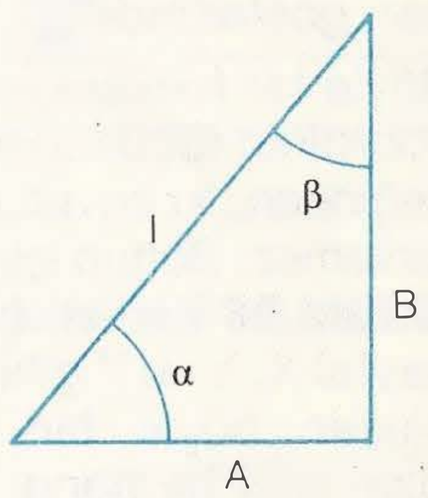
Üçboyutlu grafik (3D)

Üç boyutlu grafik (3D), bazıları çok karmaşık olan matematik formüllerine başvurmayı gerektirir. Bunlar ileride anlatılacak. Aralarından en çok kullanılanları dik üçgen problemlerine uygulanan trigonometri formülleridir.

Aşağıdaki şeklin ortasında yer alan tabloda, bir dik üçgenin bilinmeyen ögelerinin, bilinenlerden hareketle nasıl hesaplanabileceği gösteriliyor.

Kullanılan trigonometrik fonksiyonlar (mate-

DİK ÜÇGENLERİN ÇİZİLMESİ



$$B = l * sen(\alpha)$$
$$A = l * cos(\alpha)$$

$$\frac{B}{A} = \frac{sen(\alpha)}{cos(\alpha)} = tg(\alpha)$$

Bilinen elemanlar	Diğerlerinin hesaplanması
l, α	Si Aşağıdaki formüller uygulanır
A, B	$\alpha = arctg\ (B/A)$ $l = A/cos(\alpha) = B/sen(\alpha)$
A, α	$l = A/cos(\alpha)$ $B = l \cdot sen(\alpha) = A \cdot \frac{sen(\alpha)}{cos(\alpha)} = A \cdot tg(\alpha)$
B, α	$l = B/sen(\alpha)$ $A = l \cdot cos(\alpha) = B \cdot \frac{cos(\alpha)}{sen(\alpha)} = \frac{B}{tg(\alpha)}$
l, A	$cos(\alpha) = A/l$ $sen(\alpha) = \sqrt{1 - cos^2(\alpha)}$ $B = l \cdot sen(\alpha)$
l, B	$sen(\alpha) = B/l$ $cos(\alpha) = \sqrt{1 - sen^2(\alpha)}$ $A = l \cdot cos(\alpha)$

Formüllerin tümü, bir programda kullanılmak üzere simgesel bir dile çevrilebilir.
Ancak şunlar unutulmamalıdır:
— açısı radyan cinsinden ifade edilmeli
— fonksiyon adları genelde şöyledir:

$sen(\alpha) = SIN(ALFA)$
$cos(\alpha) = COS(ALFA)$
$tg(\alpha) = SIN(ALFA)/COS(ALFA)$
$\sqrt{\ldots} = SQR(\ldots)$
$arctg(B/A) = ATN(B/A)$

α açısı yerine β açısının bilindiği durum gözönüne alınmıyor $\alpha = 90 - \beta$ eşitliği bilindiği için bu durum kolaylıkla çözülür.

matik yazılışlarıyla) şunlardır:

Sinüs = Sin (α) Tanjant = tg(α)
Kosinüs = Cos (α) Kotanjant = Cotg(α)

I hipotenüsü ve açısından hareketle, dikaçının iki kenarını hesaplamak için şu formül kullanılır:

A = I * Cos(α)
B = I * Sin(α)

Başlıca trigonometrik fonksiyonlar FORTRAN ve BASIC gibi programlama dillerinde zaten yeralmaktadır. Örneğin yukarıdaki işlemleri gerçekleştirmek için gereken komutlar şunlardır:

```
A = I * COS(ALFA)
B = I * SIN(ALFA)
```

Ancak bu formüllerde açı, radyan cinsinden belirtilmelidir. Bunun için de aşağıdaki çevirme formülü kullanılır:

$$\text{Radyan cinsinden açı} = \frac{\pi * \text{derece cinsinden açı}}{180}$$

$$= \frac{3.143 \text{derece cinsinden açı}}{180}$$

Bazı diller tg(α) formülünü içermezler. Bunun için aşağıdaki formüle başvurulur:

TG(ALFA) = SIN (ALFA)/COS(ALFA)

Bu durumda kullanıcı tanjantı bir fonksiyon olarak (DEF FN komutu yardımıyla) bizzat kendisi de tanımlamayıp kullanabilir.

Eğer diküçgenin yalnızca kenarları biliniyor, açıları bulunmak isteniyorsa, çözüm daha zordur. Bu durumda Pisagor Teoremine veya ters trigonometrik fonksiyonlara başvurulabilir. Örneğin, dik açının ikikenarı biliniyorsa, α açısı kotanjant (ATN) fonksiyonunda yararlanılarak hesaplanabilir:

ALFA = ATN(B/A)

burada A ve B dik açının iki kenarıdır.

Kotanjant fonksiyonu da, genellikle programlama dillerinde bulunur.

İki değişkenli fonksiyonlar

Buraya kadar karşılaşılan fonksiyonlar Y = f(X) biçimindeydi, ve bağımlı değişken (Y),X gibi tek bir bağımsız değişkenin fonksiyonu olarak ifade ediliyordu. Bu tür fonksiyonlar, kartezyen bir düzlem üzerinde gösterilebilirler. Bunun yolu da her bir noktayı iki değerle (koordinatlar) konumlandırmaktır. Eğriler de, referans eksenleri ekranın yüzeyiyle aynı olduğundan, kolaylıkla gösterilebilirler.

Ancak birden fazla değişkene bağlı fonksiyonlar da vardır. En çok rastlanılanları da iki bağımsız değişkenli, Y = f(X,Z) veya Z = f(X,Y) biçimindeki fonksiyonlardır.

Kullanageldiğimiz simgeler yine geçerli: f, yaratılan bir fonksiyonu, yani bağımlı değişkenlerin değerlerini bağımsızlarınkine bağlayan bir kuralı gösteriyor. Bu tür bağıntıların Y = f(Y,Z) biçiminde gösterimi, en çok rastlanan gösterimdir.

Böyle bir fonksiyon, noktaların herbiri için 3 koordinat (üçüncüsü Z olmak üzere) gereke-ceğinden, iki boyutlu bir düzlem üzerinde gösterilemez. Bunun için 3 referans doğrusundan oluşan bir sistem gerekir. Şekil 1 (bkz. yan sayfa) X, Y ve Z gibi 3 referans doğrusundan oluşan böyle bir sistemi göstermektedir. Uzaydaki herhangi bir nokta, bu doğrulara olan uzaklıklarıyla tanımlanabilir.

Tabiî, bu üçüncü boyutun varlığı, iki değişkenli bir fonksiyonu bir düzlem üzerinde (örneğin ekran veya çizim kağıdı) gösterme olanağını ortadan kaldırmaktadır. Bu olanaksızlığı yenmek amacıyla, tıpkı ressamların kullandığı perspektif gibi, derinlik izlenimi verecek algoritmaların yazılması düşünülmüştür.

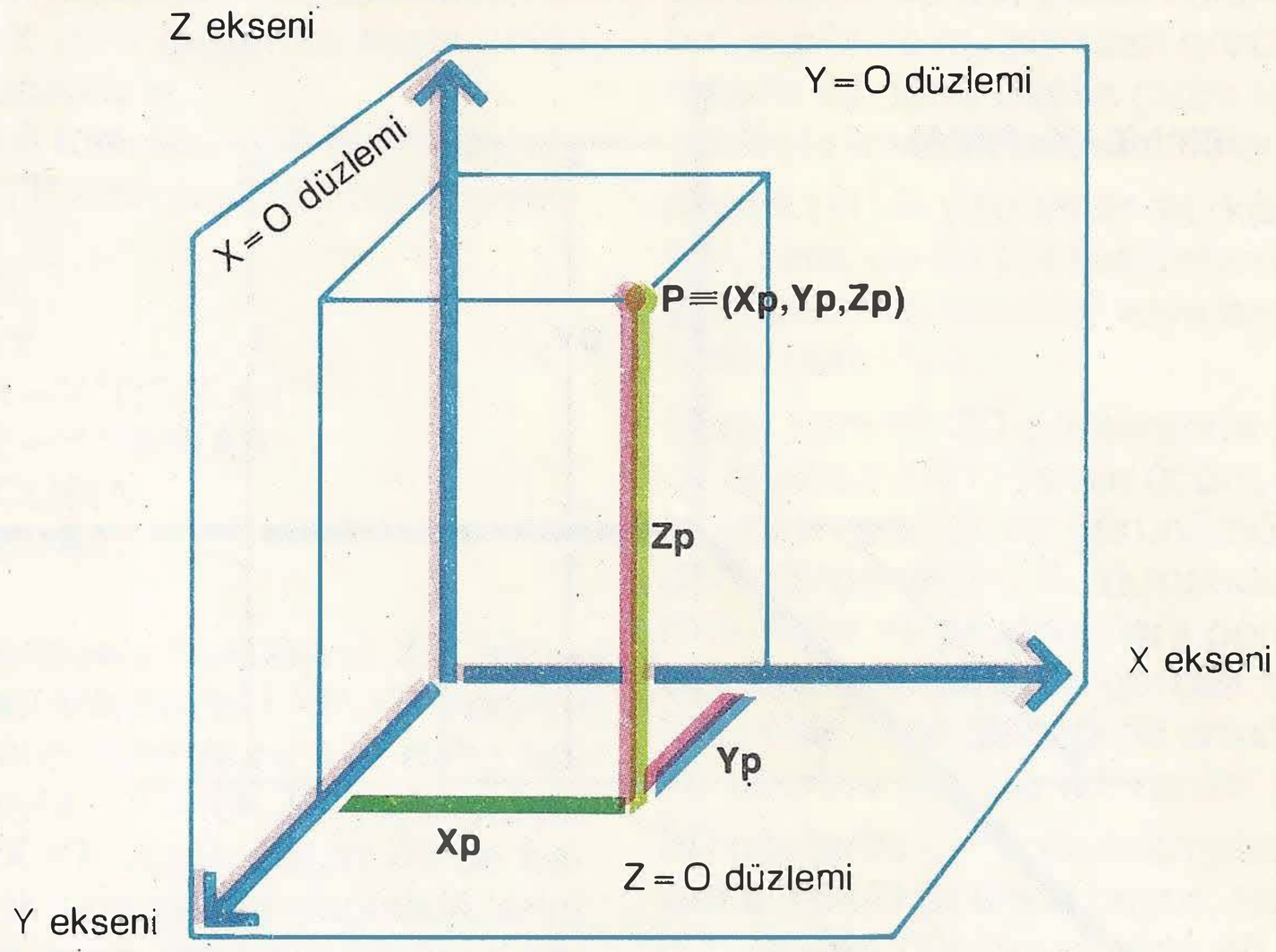
Ancak burada sergilenen yöntemler, gerçek üç boyutlu grafik uygulamalarda kullanılanlar değildir. Bunun yerine, çeşitli kullanım olanaklarına yatkın olmakla birlikte, eldeki donanımla sınırlanan örnekler veriliyor. Gerçek bir 3D yazılımı ortaya koyabilmek için çok güçlü makineler kullanmak gerekir.

3 D gösterimi. Uzaydaki herhangi bir P noktasını ekranda göstermek için kullanılan yordam, şekil 2'de gösterilmiştir.

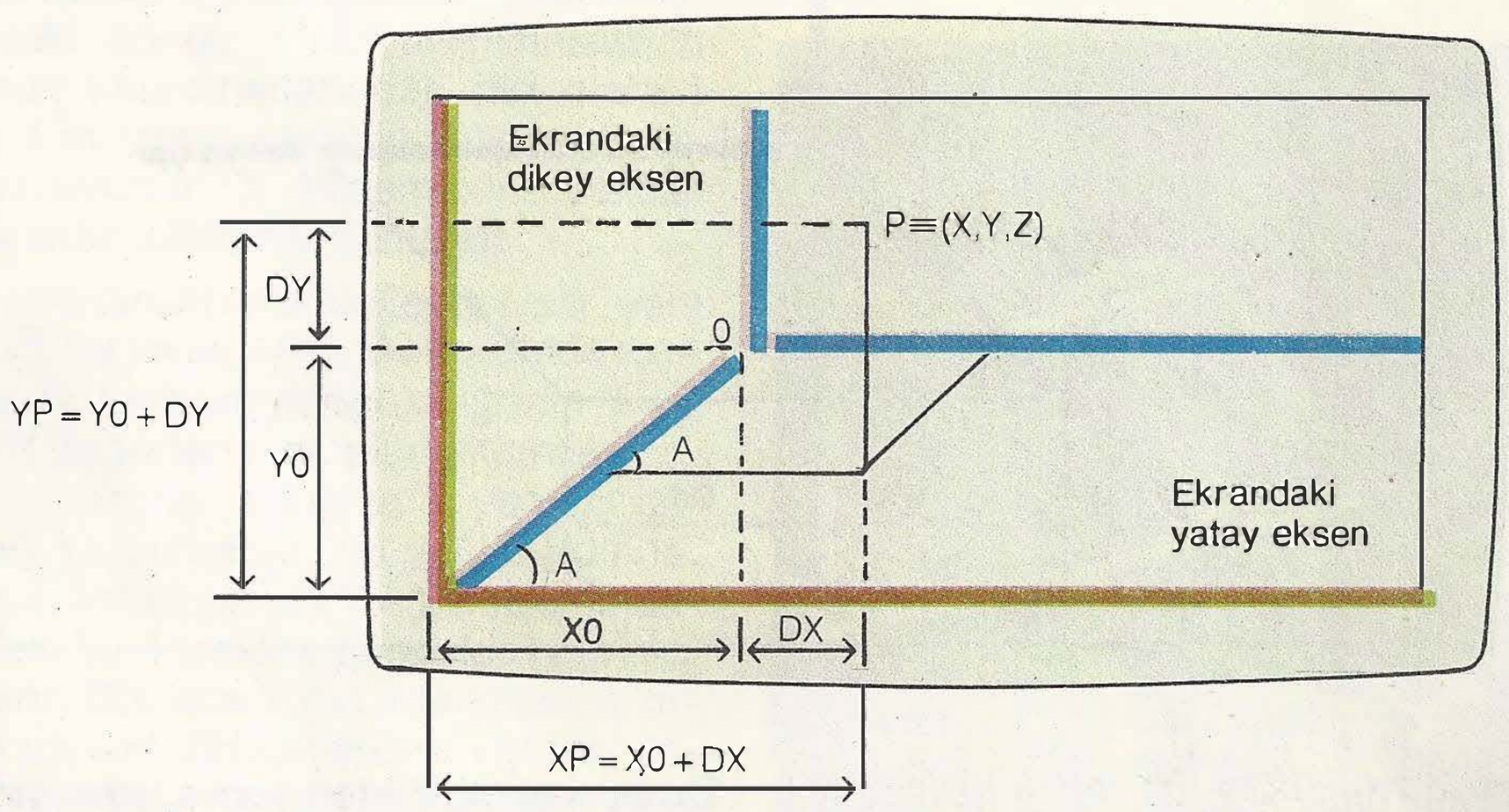
Referans sisteminin O harfiyle belirtilen merkezi, ekrandaki bir noktaya konumlandırılmıştır. Örneğimizi basitleştirmek amacıyla, bu merkezin ekranın sol alt kısmında yer aldığını kabul edelim (bazı durumlarda da bu merkezin sol üstte bulunması daha elverişli olabilir).

P noktasının (bu nokta X,Y, koordinatlarıyla

UZAYDA REFERANS SİSTEMİ



UZAYDA BİR REFERANS SİSTEMİNİN EKRANDA GÖSTERİMİ

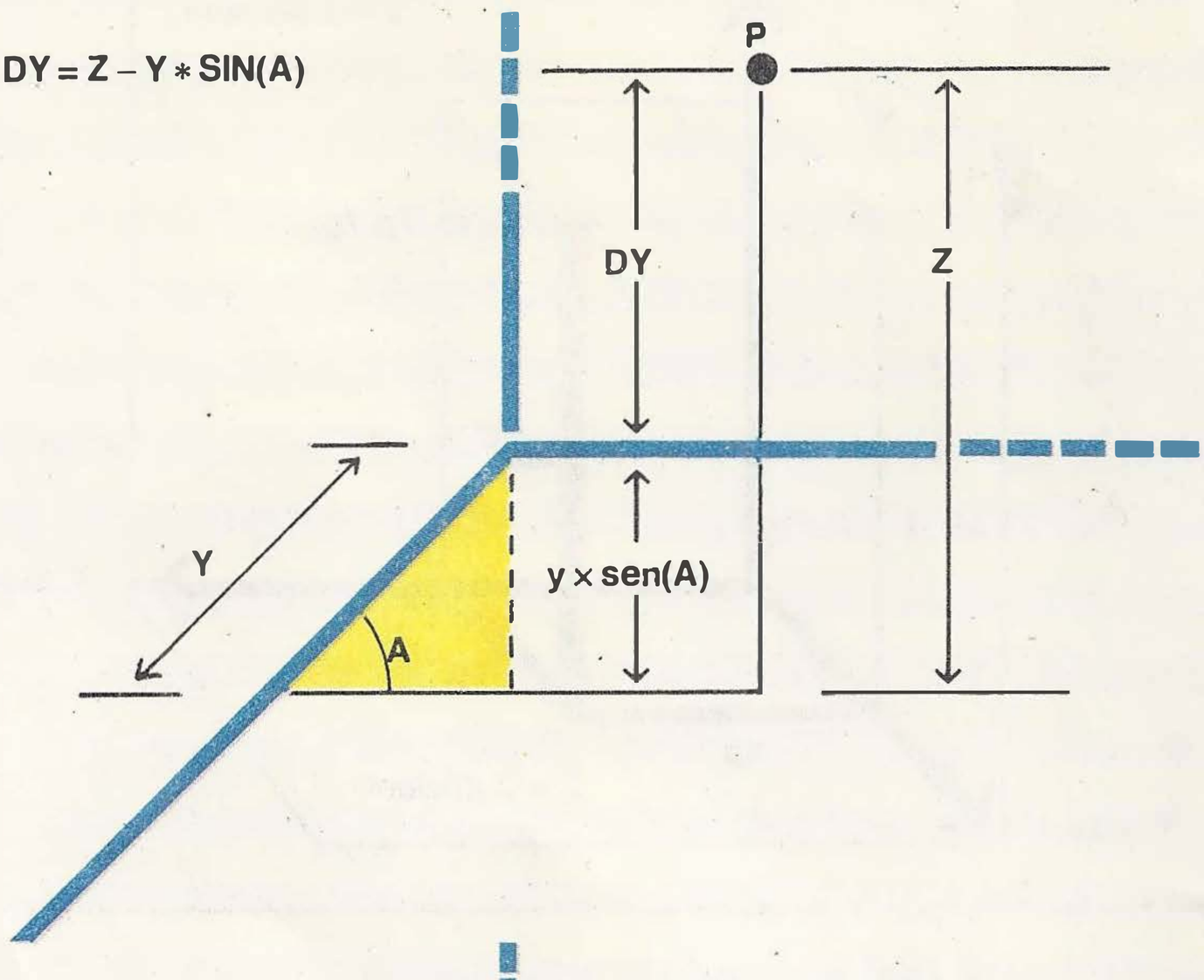


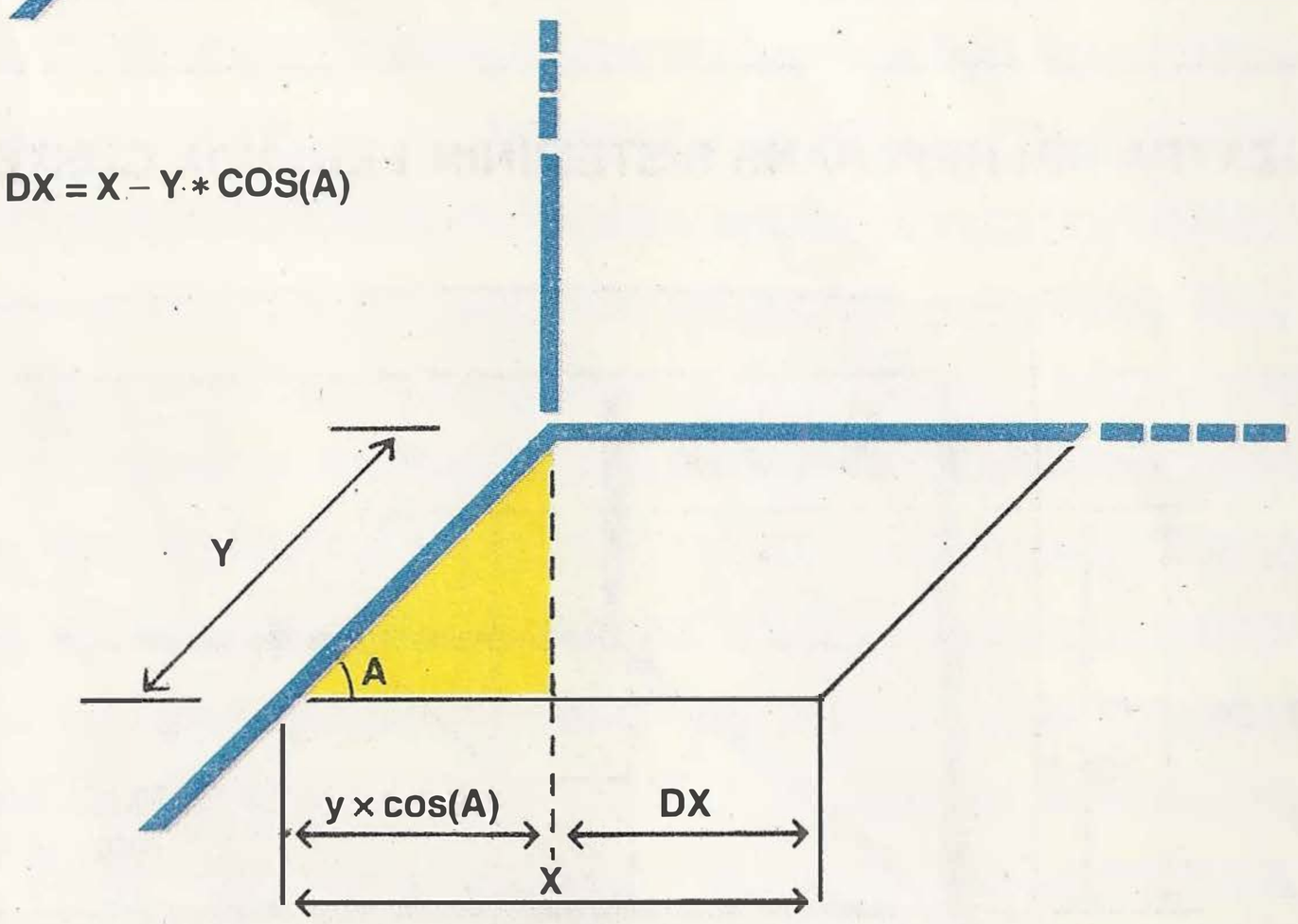
tanımlıdır), ekrana göre koordinatları XP ve YP'dir. Bu koordinatlar, referans sisteminin koordinatlarına DX ve DY değerleri eklenerek elde edilir, yani:

$$XP = X0 + DX$$
$$YP = Y0 + DY$$

DX ve DY değerleri bir kez saptandıktan sonra, 3D fonksiyonunun tüm noktaları için ekran koordinatları hesaplanır. Yani, son çözüm-

EKRAN KOORDİNATLARININ HESAPLANMASI





lemede, problem, DX ve DY miktarlarının hesaplanmasına indirgenebilmektedir.

İlk aşama, bilinen trigonometri formüllerinden yararlanarak A açısının hesaplanmasından oluşur (pratik nedenlerle, formüllerde ALFA olarak geçen açı, burada A ile belirtilmektedir). A'nın değeri X0 ve Y0 koordinatlarıyla belirlenir. Bu koordinatlar, hipotenüsü 0 noktasıyla (ekrandaki) merkezi birleştiren doğru parçası olan bir dik üçgenin diğer iki kenarını oluştururlar. Sonuç şöyledir:

A = ATN(Y0/X0)

Bu açı belirlendikten sonra, sıra, yine dik üçgenin özelliklerinden yararlanılarak DX ve DY'nin hesaplanmasına gelir. Bu özellikler kullanıldığında, şunlar elde edilir:

DY = Z-Y*SIN(A)
DX = X-Y*COS(A)

Bu formülleri anlayabilmek için, yukarıdaki şekiller, referans sisteminin 3D olduğu dikkate alınmadan incelenmelidir.

Şimdi üç boyutlu bir fonksiyonun nasıl çizildiğini görelim:

1 / Önce X ve Y bağımsız değişkenlerine rasgele değerler verilir.
2 / Fonksiyonun matematik ifadesinden hareketle her X ve Y değerine karşı gelen Y değeri hesaplanır.
3 / Aşağıdaki ikili formüllerden biriyle ekrandaki XP ve YP koordinatları hesaplanır.

```
YP = X0 + DX
YP = YO + DY
XP = XO + (X—Y*COS(A))
YP = YO + (Z—Y*SIN(A))
DX = X—Y*COS(A)
DY = Z—Y*SIN(A)
```

Burada, eğri üzerindeki noktaların X,Y koordinatlarıyla, ekran üzerindeki XP,YP koordinatlarının aynı olmadıklarını bir kez daha belirtelim. Dolayısıyla, Y = f(X,Z) biçimindeki fonksiyon, Z = f(X,Y) olarak değiştirilirse hesaplarda da gerekli koordinat değişikliklerini yapmak koşuluyla, formüller geçerliliklerini korur. Bu durumda bağımsız değişkenler X ve Z değil, X ve Y'dir. Fonksiyon Y yerine Z'yi hesaplamaktadır. Z = f(X,Y) en çok rastlanılan biçimdir. Basitleştirmek amacıyla, plan üzerinde fonksiyon çizdirme programında kullanılan tüm kontrol ve ölçek değiştirmeler çıkarılmıştır. Dışardaki döngü Y'nin artırılmasından (fonksiyonun koordinatlarından biri olarak) içereki ise X'in artırılmasından oluşmaktadır. Bu da, fonksiyonun, X düzlemlerine paralel kesitler alınarak çizilmesi demektir.

Örneğin Y koordinatının ilk değerinin sıfır, yani Y1 = O olduğunu varsayalım. Bu durumda içerdeki döngü (X'i artıran döngü), eğrinin Y = 0 için bütün X değerleri hesaplandıktan sonra, dıştaki döngü artırılır. Böylece Y, yeni bir değere, örneğin L'ye geçer. Bu yeni durumda, içteki döngü, fonksiyonun Y = 1 düzlemindeki (bu düzlem bir öncekine paraleldir) eğrisini ortaya çıkarır. Böylece Y'nin artırılmasıyla ortaya çıkan paralel düzlemlerdeki eğriler, üstüste bindirilerek fonksiyonun 3D çizimi elde edilir. Paralel düzlemlerde ortaya çıkan bu eğriler, aslında üç boyutlu bir cismin kesitlerinden başka bir şey değildir. Burada anlatılan mekanizma, sayfa 1640'da şematik bir biçimde gösterilmiş ardından da programın dökümü verilmiştir.

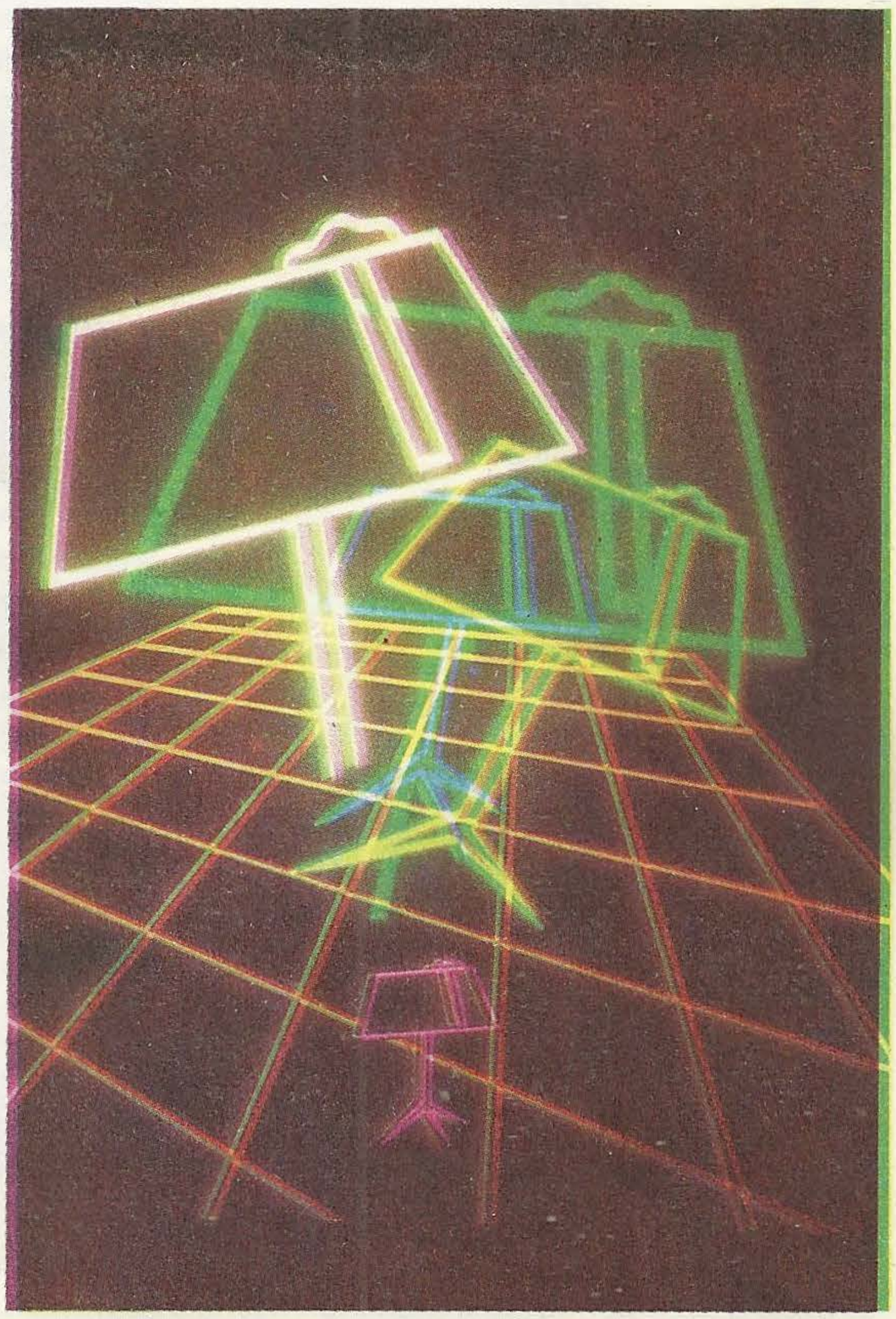
3D grafik problemleri. Gerçek 3D grafikle, basit üç boyutlu gösterim, aynı şeyler değillerdir. Gerçekte, gerek kullanılan araçlar ve gerekse programlar ve fiyatları konusunda aralarında önemli farklılıklar vardır. 3D grafik gösterilecek cismi oluşturan uzaydaki noktaların konumlarının bellekte bulundurulmasını gerektirir. Bu da, cismin çeşitli yansıtmalarının yapılmasını, ara kesit görünümlerinin alınmasını ve daha başka çizim işlemlerinin gerçekleştirilmesini olanaklı kılar.

Ancak bu tür yazılımlar da, karmaşık makineleri, özellikle de yüksek bellek kapasitelerini ve yine gelişmiş terminal veya başka çıkış gereçlerini gerektirir.

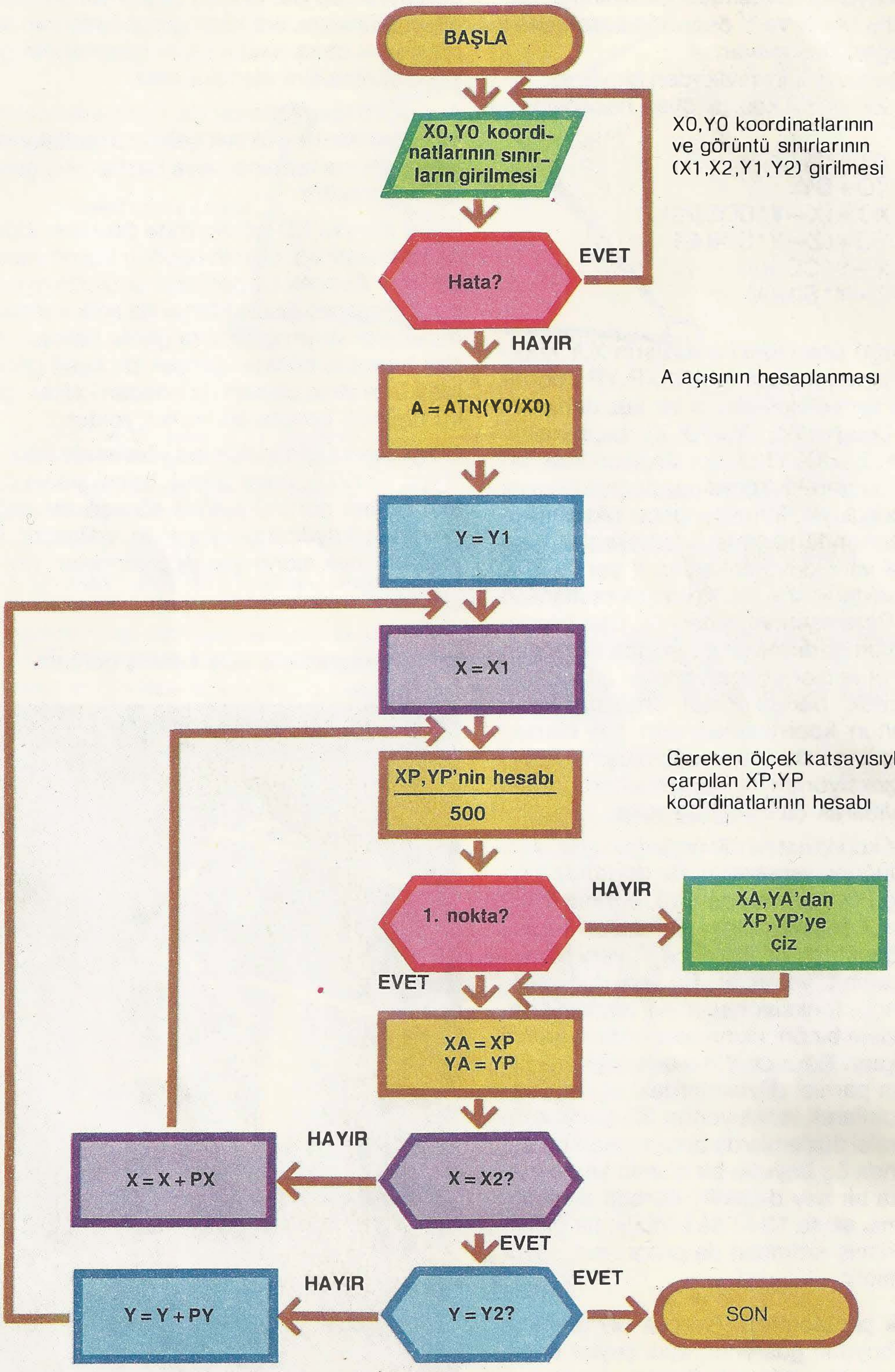
Buna karşılık 3D gösterimde problem oldukça basitleştirilip, cismin üçüncü boyutunun izlenimini verecek bir görünümünü çizmeye kadar indirgenebilir. Bu durumda artık karmaşık makineler ve programlara gerek kalmaz. Ancak bununla birlikte, gerçek bir kesit görüntüsü elde etme olanağı da ortadan kalkar, çünkü üçüncü boyuta ait veriler yoktur.

3D gösterim için kullanılan yöntemde (bkz, çizim s.1642) üçüncü boyut, temel şeklin (cismin önden görünüşünün) 45 açılı bir doğru boyunca kaydırılmasıyla elde edilmiştir (bu yüzden, noktaların yer değiştirmeleri her iki

Bir 3D programıyla elde edilmiş görüntü.



2 DEĞİŞKENLİ BİR FONKSİYONUN GÖRÜNTÜLENMESİ

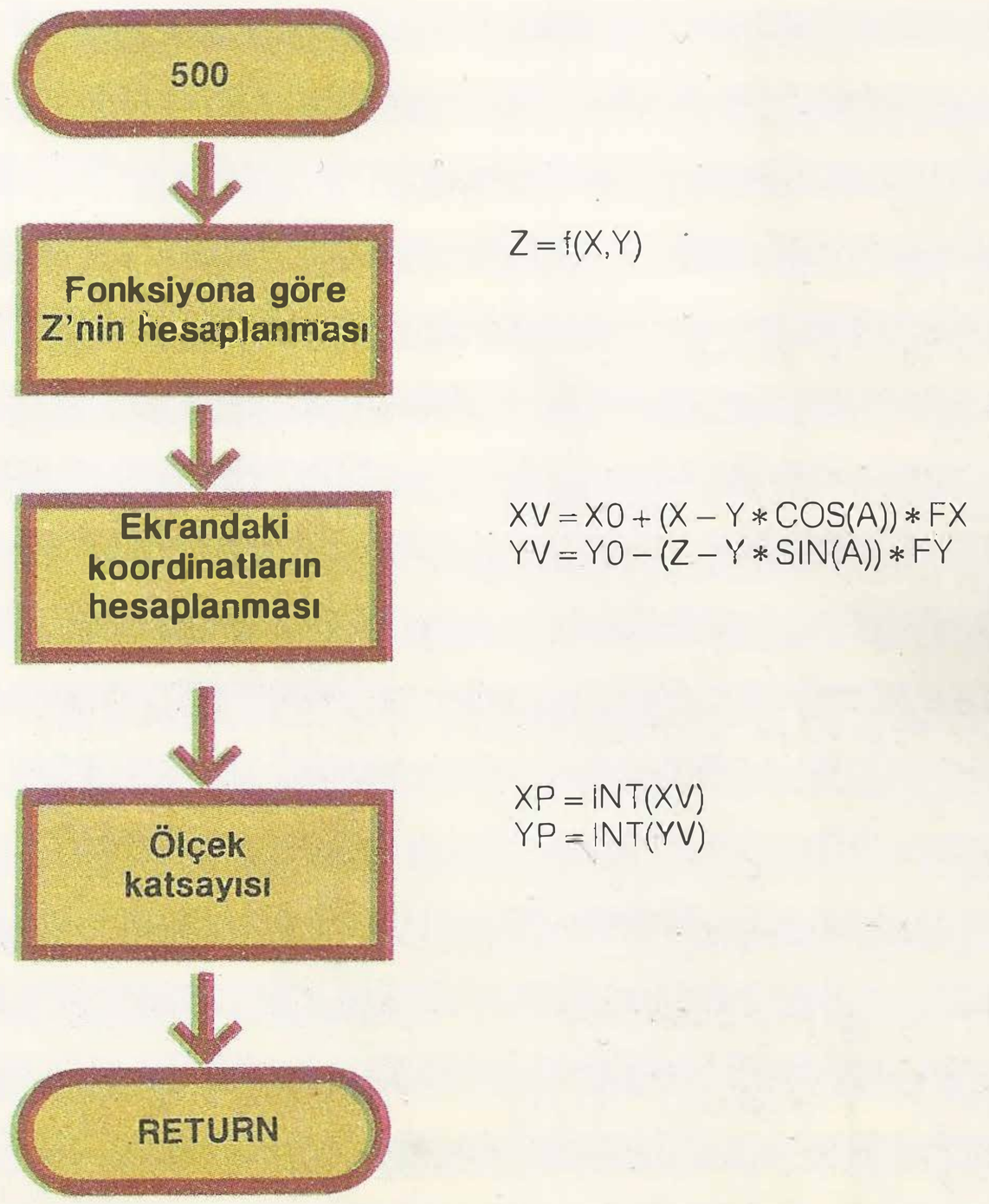


eksen üzerinde de aynıdır). Temel şeklin köşeleri 1'den 6'ya kadar numaralandırılmıştır. Derinlik izlenimi vermek için, bu köşelerin koordinatları S kadar artırılarak şekil yeniden çizilir. Bu yeniden çizimden sonra da aynı numaraları taşıyan köşeler birer doğruyla birleştirilir. Ancak yeniden çizim işlemi sırasında, temel şeklin ardında kalan çizgilerin silinmesi (daha iyisi hiç çizilmemesi) gerekmektedir. Bu üçboyutlu çizimin en karmaşık sorunlarından biridir: Çizilmemesi gereken çizgiler nasıl ayıklanacaktır. Aynı sorun, gerçek bir cisimde görünmeyen yüzeylerin bulunması biçiminde somutlaştırılabilir.

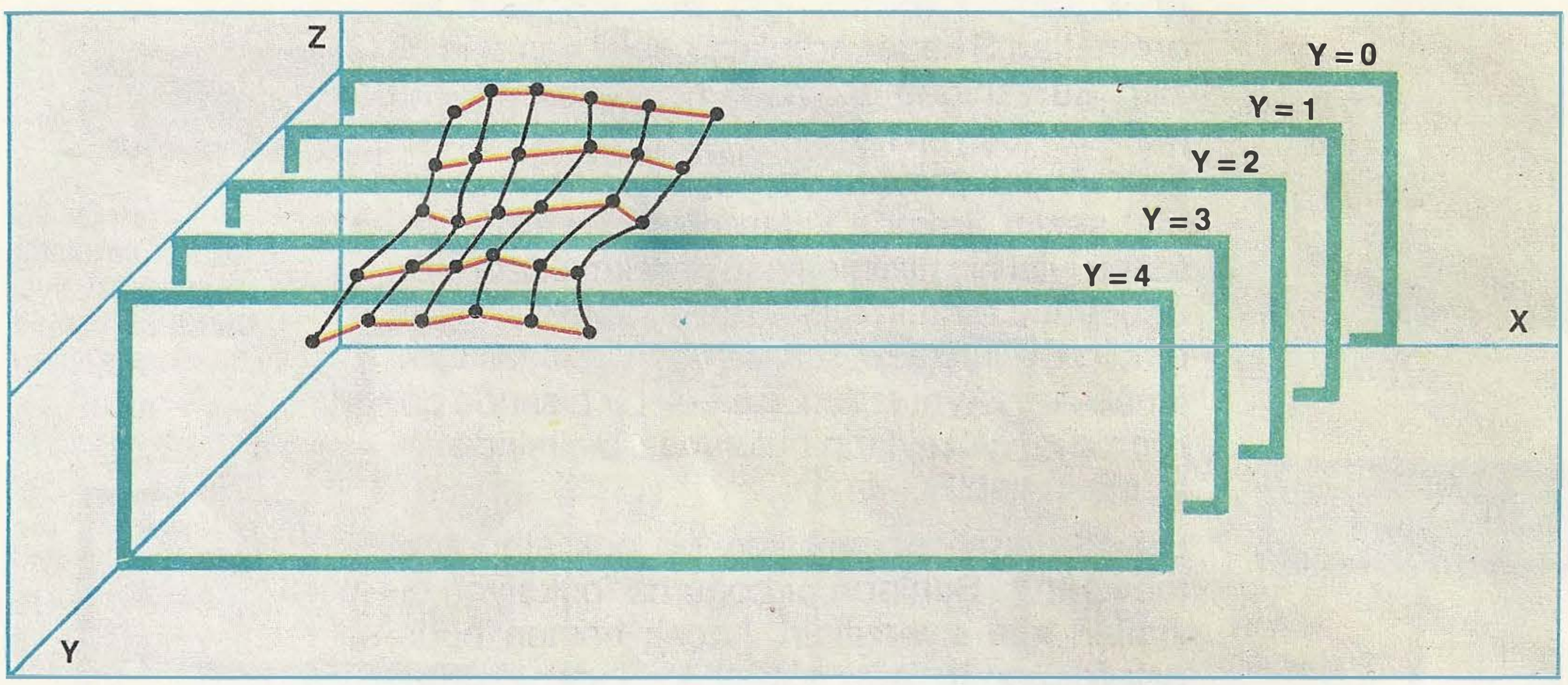
Bu sorunun çözümü için bir çok algoritma mevcuttur. Bunların birçoğu da fonksiyon çizimleri için elverişlidir. Ancak hemen hepsi çok özgün çözümler getirebildiğinden, bunları matematik fonksiyonların çizimleri dışında kalan üç boyutlu çizimler için genelleştirmek çok zordur.

Öyleyse sorunu şöyle koymak yerindedir: Öyle bir yazılım gerekmektedir ki, arka planda kalan bütün çizgileri ayıklayabilsin.

EKRANDAKİ KOORDİNATLARI HESAPLAYAN ALT-PROGRAM



GÖSTERİM YÖNTEMİNİN ŞEMASI



İKİ DEĞİŞKENLİ BİR TABLONUN ÇİZİMİ

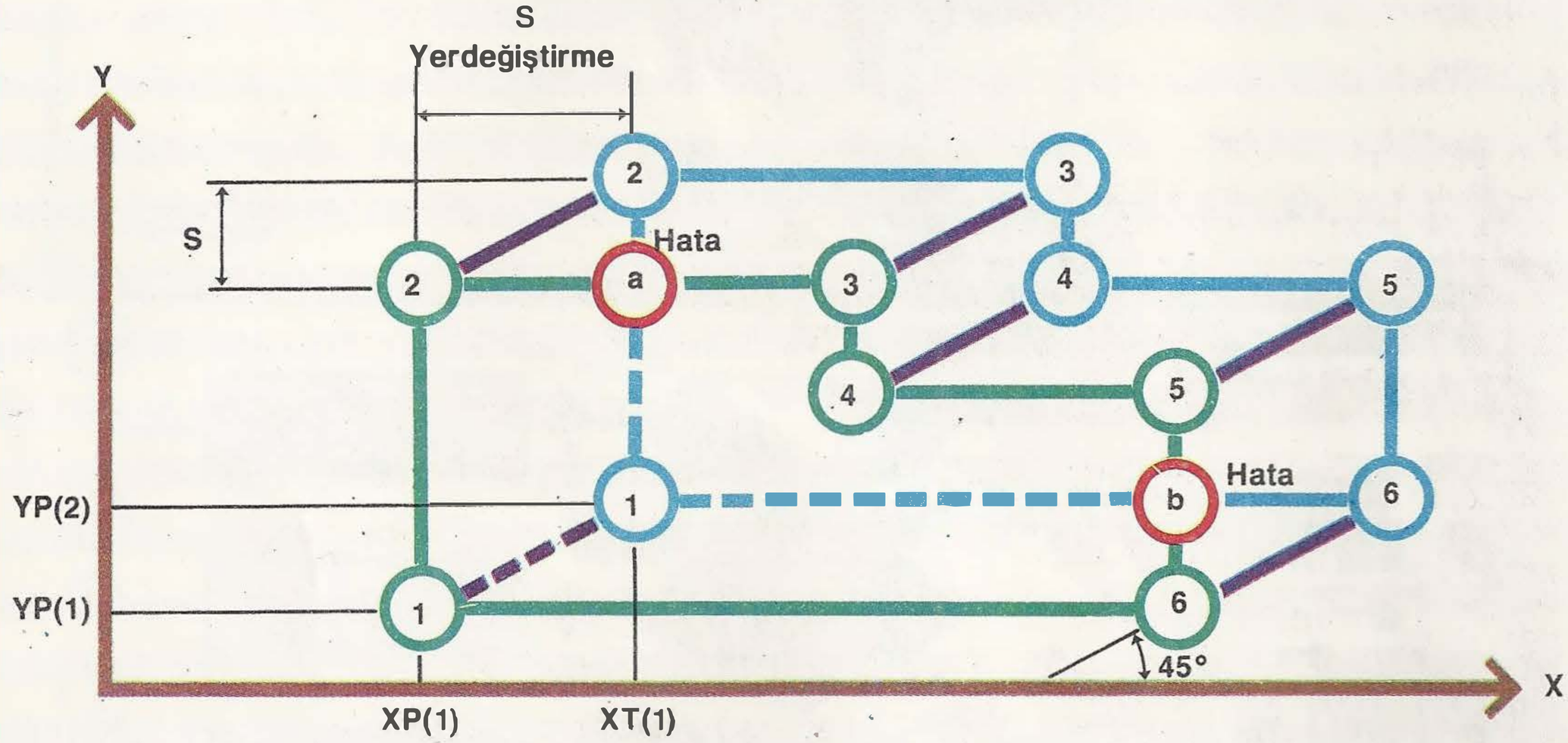
```
10 REM ************************
20 REM *                      *
30 REM *                      *
40 REM * UCBOYUTLU CIZIM      *
50 REM *                      *
60 REM *                      *
70 REM ************************
80
   :
90.
   :
100 REM ------------------------------
110 REM EKRAN SINIRLARI
120 REM ------------------------------
130 XN=0
   :XM=279
   :YN=0
   :YM=191
140 REM ------------------------------
150 REM OLCEK KATSAYISI
160 REM ------------------------------
170 FX=5
   :FY=10
180 REM ------------------------------
190 REM NOKTALARIN YOGUNLUGU
200 REM ------------------------------
210 PX=.2
   :PY=.2
220 REM ------------------------------
230 REM MERKEZIN KOORDINATLARI
240 REM ------------------------------
250 XO=129:
    YO=61
270 REM ------------------------------
280 REM VERILERIN GIRILMESI
290 REM ------------------------------
300 TEXT
       :HOME
310 VTAB 12
320 INPUT "X DOGRUSU";X1,X2

330 VTAB 14
340 INPUT "Y DOGRUSU";Y1,Y2
```

```
350 A=ATN(Y0/X0)
    :REM
360 HGR2
    :HCOLOR=3
370 GOSUB 2000
380
    :
390
    :
400 REM **********************
410 REM **                  **
420 REM ** ANA DONGU        **
430 REM **                  **
440 REM **********************
450
    :
460
    :
470 FLAG=1
480 FOR Y=Y1 TO Y2 STEP PY
490 FOR X=X1 TO X2 STEP PX
500 REM ------------------------------
510 REM XP VE YP'NIN HESAPLANMASI
520 REM ------------------------------
530
    :
540
    :
550 Z=(SIN(X)/X)*(5*SIN(Y))
560
    :
570
    :
580 DX=X-YCOS(A)
590 DY=Z-Y*SIN(A)
600 XV=X0+DX*FX
610 YV=Y0-DY*FY
620 XP=INT(XV)
630 YP=INT(YV)
640 REM ------------------------------
650 REM GRAFIK CIZIMI
660 REM ------------------------------
670 IF FLAG THEN FLAG=0
    :GOTO 690
680 HPLOT XA,YA TO XP,YP
690 XA=XP:
    YA=YP
700 NEXT X
710 FLAG=1
720 NEXT Y
730
    :
740
    :
750 REM **********************
760 REM ** DONGUNUN SONU **
770 REM **********************
780
    :
790
    :
800 GET D$
810 TEXT
    :HOME
820 END
2000 REM ------------------------
2010 REM KARTEZYEN EKSENLER
2020 REM ------------------------
2030 HPLOT XN,YN TO XM,YN TO XM,YM
     TO XN,YM TO XN,YN
```

```
2040 HPLOT XO,YO TO XO,YN
2050 HPLOT XN,YO TO XM,YO
2060 HPLOT XO,YO TO XN,YM
2065 HCOLOR=0
2070 FOR I=XO-3 TO 0 STEP -3
2080 HPLOT I,YO
2090 NEXT
2100 HCOLOR=3
2110 RETURN
```

3D GÖSTERİM İÇİN KULLANILAN YÖNTEM



Program, gerekli bütün fonksiyonlara sahip olmadığından, yanında Hata yazılı çizgiler gerekmediği halde çizilmiştir.

Referans şekli (ilk çizilen şekil budur)

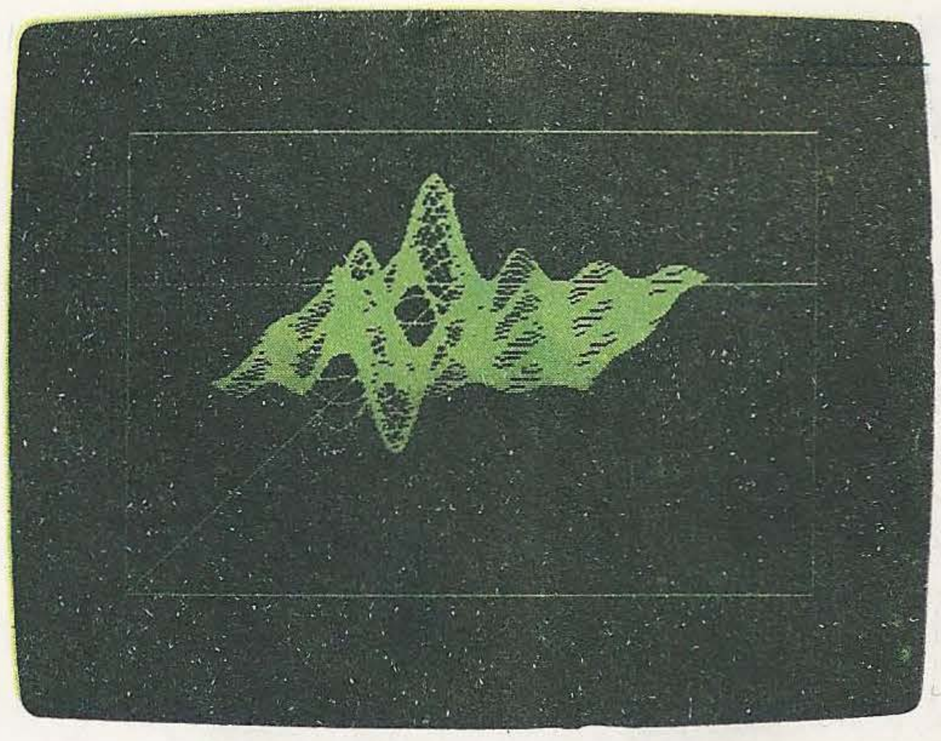
Yerdeğiştiren şekil

İki şekli birbirine bağlayan ara çizgiler (derinlik izlenimi)

Referans şeklinin köşeleri

Yerdeğiştirmiş şeklin köşeleri

Kesişim noktaları



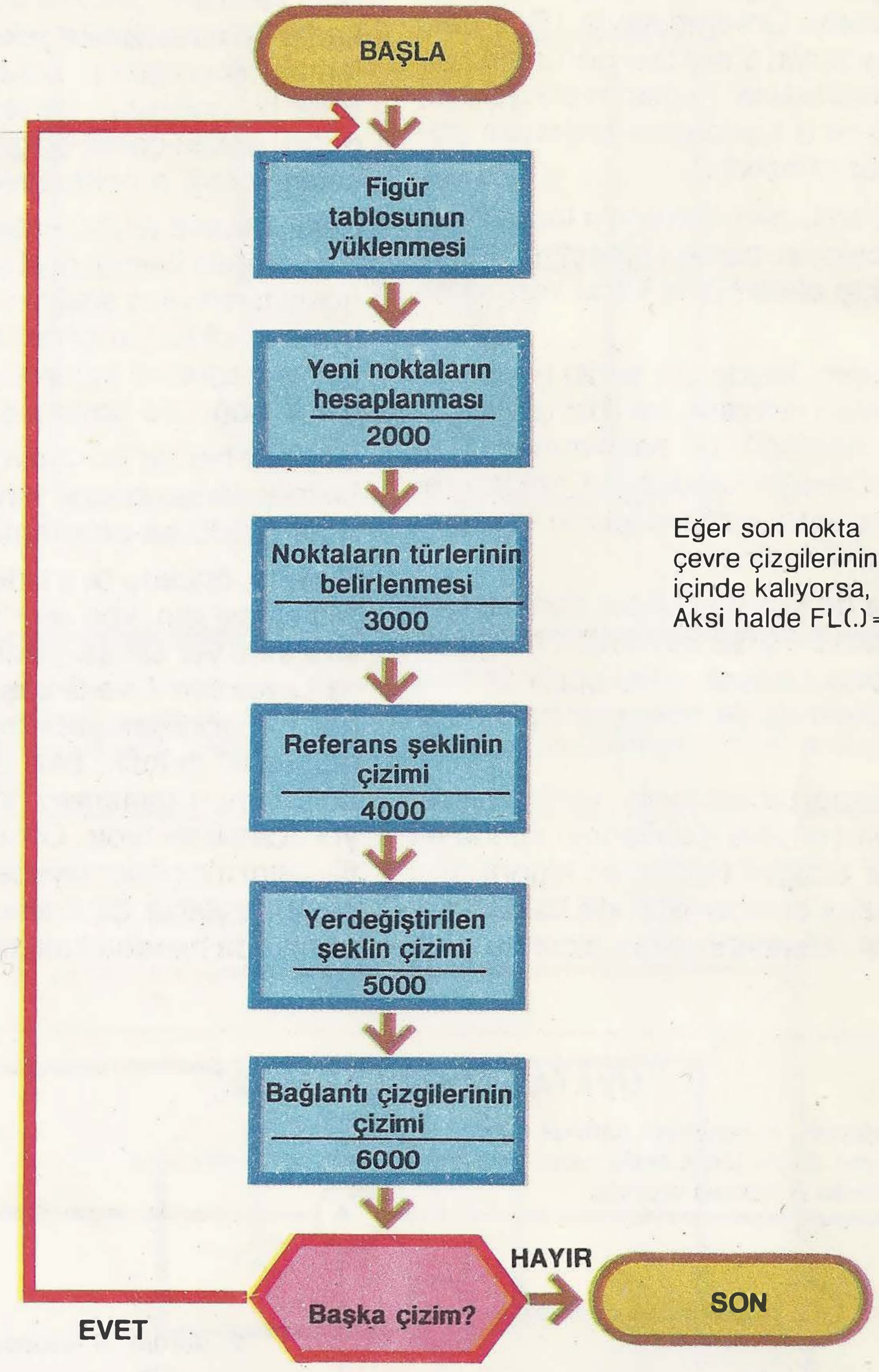
İki değişkenli bir fonksiyonun nihaî gösterimi.

Burada verdiğimiz programda (bkz. karşı sayfadaki akış şeması) kullanılan yöntem, yalnızca referans şeklinin çözümlenmesine dayanmakta ve bazı durumlarda hatalı sonuçlar verebilmektedir. Bu nedenle kullanılan yöntem genelde doğru değildir. Ancak, yine de soruna yaklaşımda iyi bir örnek olarak alınmalıdır.

Kullanılan fonksiyonlar şunlardır:

- Kaydırılarak üçüncü boyutu elde edilen temel şeklin köşelerinin koordinatlarının belirlenmesi (1000 alt-programı).
- Kaydırılan noktanın koordinatlarının hesaplanması (2000 alt-programı). Kullanılan teknik, yalnızca bütün koordinatların S kadar artırılmasından oluşuyor.
- Kaydırılan şeklin her noktasının türünün be-

3D BENZETİM PROGRAMININ AKIŞ ŞEMASI



Eğer son nokta
çevre çizgilerinin
içinde kalıyorsa, FL(.) = 1.
Aksi halde FL(.) = 0

Program dökümüne oranla, verilen akış şemalarından bazıları basitleştirilmiştir.

XP(21), YP(21)	Referans şeklinde yeralan noktaların koordinatları
XT(21), YT(21)	Kaydırılan şeklin koordinatları
FL(20)	Konum göstergesi (nokta içerdeyse 1, dışardaysa 0).
N1	Girilen nokta sayısı
FS	Ölçek katsayısı
On	n ekseninin yön göstergesi (OY = -1 merkezin sol üstte olduğunu gösterir.
S(2,20)	Yatay analizde sağ/sol göstergesi
T(2,20)	Düşey analizde yukarı/aşağı göstergesi

lirlenmesi. Köşelerin koordinatları hesaplandıktan sonra, aynı numaralı köşeleri birleştiren doğruların görünür olup olmadığının belirlenmesi. Örneğin sayfa 1642'deki örnekte, kaydırılan 1 noktası görünmez bir çizgiyle 2 noktasına bağlanmıştır. Oysa, kaydırılan 2 ve 3 noktalarını bağlayan çizgiler görünür olmalıdır.

3000 alt-programı, aynı numarayı taşıyan köşe çiftlerini inceleyip, bunları birleştiren kenarın görünür olup olmadığına karar vermektedir.

Kullanılan yöntem, kaydırılan şeklin çevre çizgilerinin ne zaman refarans şeklinin çevre çizgilerinin içine düştüğünün belirlenmesinden oluşmaktadır. Örneğin kaydırılan 1 noktası referans şeklinin çevre çizgileri içinde kaldığından görünmemektedir.

Bu teknik yine de sınırlıdır. Eğer derinlik (yani S miktarı) fazla olursa kaydırılan nokta temel şeklin dışına çıkacak, yani görünür hale gelecek bu durumda da sonuçlar hatalı olacaktır.

Aşağıda, herhangi bir noktanın, verilen bir referans şeklinin çerçeve çizgilerinin içinde mi dışında mı yer aldığını belirleyen algoritmayı veriyoruz. Başka amaçlar için de kullanılabilen bu yöntem, köşelerin sıralı biçimde girildiği varsayımına dayanmaktadır.

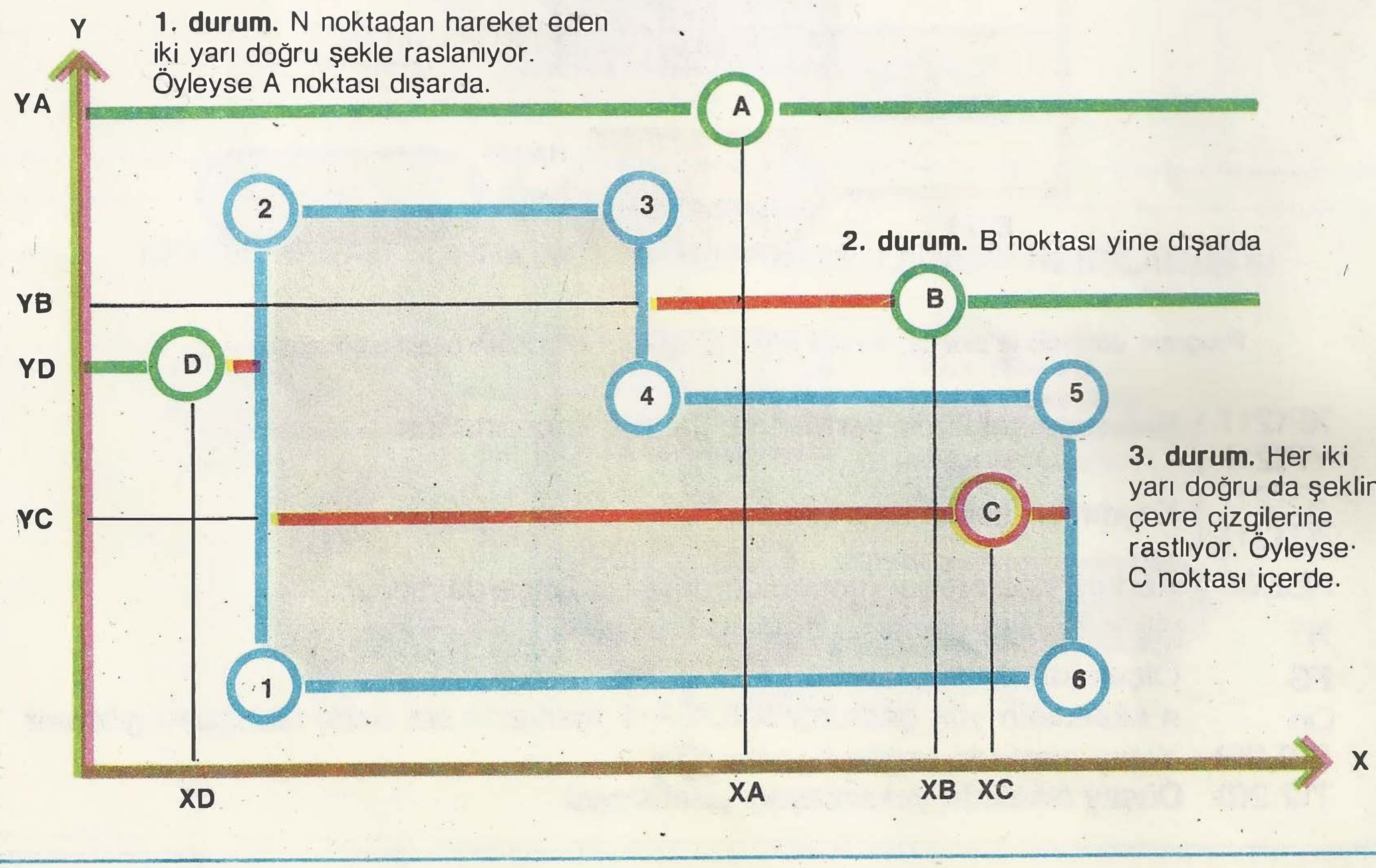
Gösterimimizde, 1'den 6'ya kadar olan numaralar referans şeklinin köşelerini, A'dan O'ya kadar olan harfler de şekle göre konumu belirlemek istediğimiz noktaları belirtmektedir. Eğer bir noktadan hareketle çizilen bir yarım doğru şeklin çevre çizgilerinin ikisiyle birden kesişiyorsa, o nokta şeklin içinde demektir.

Şekil 1 (karşı sayfa) yatay doğru incelememizin yetersiz kaldığı özel bir şekli gösteriyor. P noktası referans şeklinin dışında yer aldığı halde, bu noktadan çıkan iki doğrunun ikisi de çevre çizgilerini kesiyor. Öyleyse aynı inceleme Y doğrusu boyunca da yapılmalı.

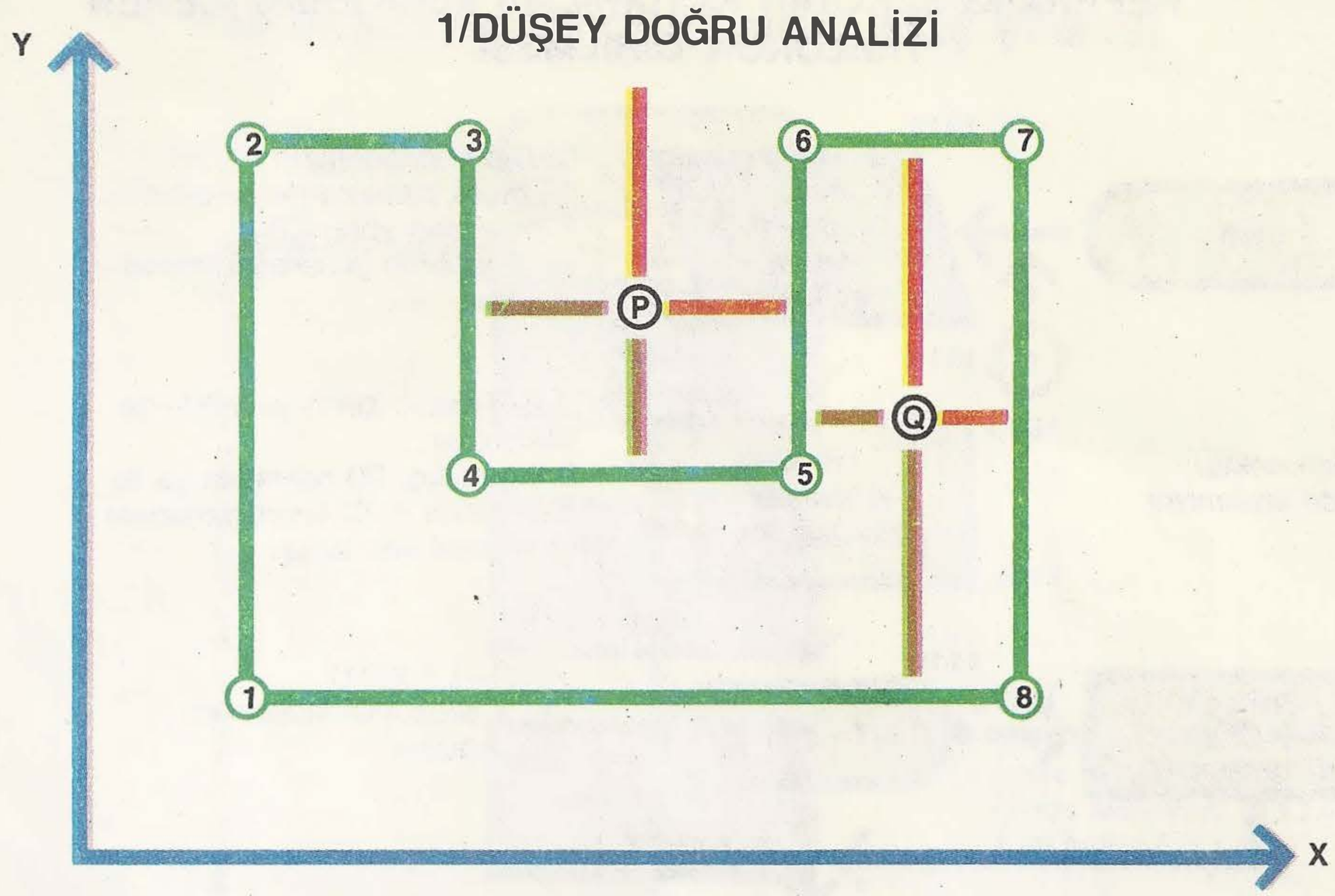
Böylece her bir noktanın türü (içerde/dışarda) belirlendikten sonra, kaydırılan çevre çizgilerinin (5000 alt-programı) ve derinliği.

Burada, öncede belirtildiği gibi, bir kenarın çizilmemesi için, köşelerinden birinin temel şeklin içinde yer alması yeterlidir. Bu yönteme göre kaydırılan 1 ve 2 köşelerini birleştiren kenar hiç görünmeyecektir. Oysa, bu yöntemi oluşturan mantık, bazı durumlarda şeklin bir bölümünün tamamen ortadan kaybolmasına yol açabilmektedir. Örneğin şekil 1'de, girintili kısım hiç çizilemeyecektir. Öyleyse bazı kenarların yalnız bir kısmının çizilebileceği durumlar da hesaba katılmalıdır (5500 alt-prog-

1/YATAY DOĞRU ANALİZİ

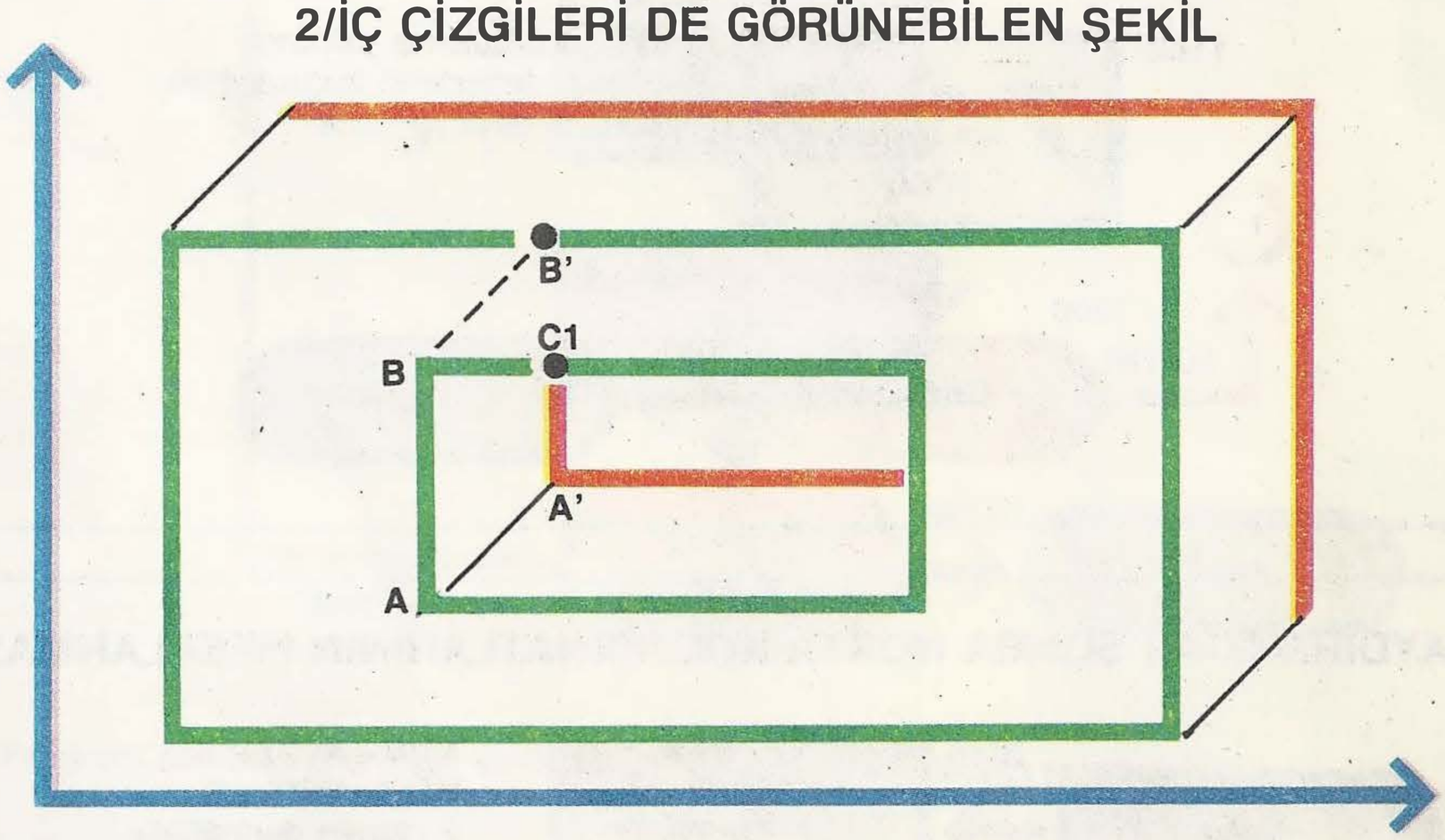


1/DÜŞEY DOĞRU ANALİZİ



Burada, Y ekseni boyunca bir analizi daha gerektiren normal dışı bir durum sözkonusudur. Yalnızca deminki analizle, P noktası içerde gözükecekti (1. durum). Ancak Y ekseni boyunca ikinci bir analizden sonra noktanın dışarda olduğu anlaşılıyor. Buna karşın Q noktası, her iki analizden sonra 1. durumdan ayrılıyor.

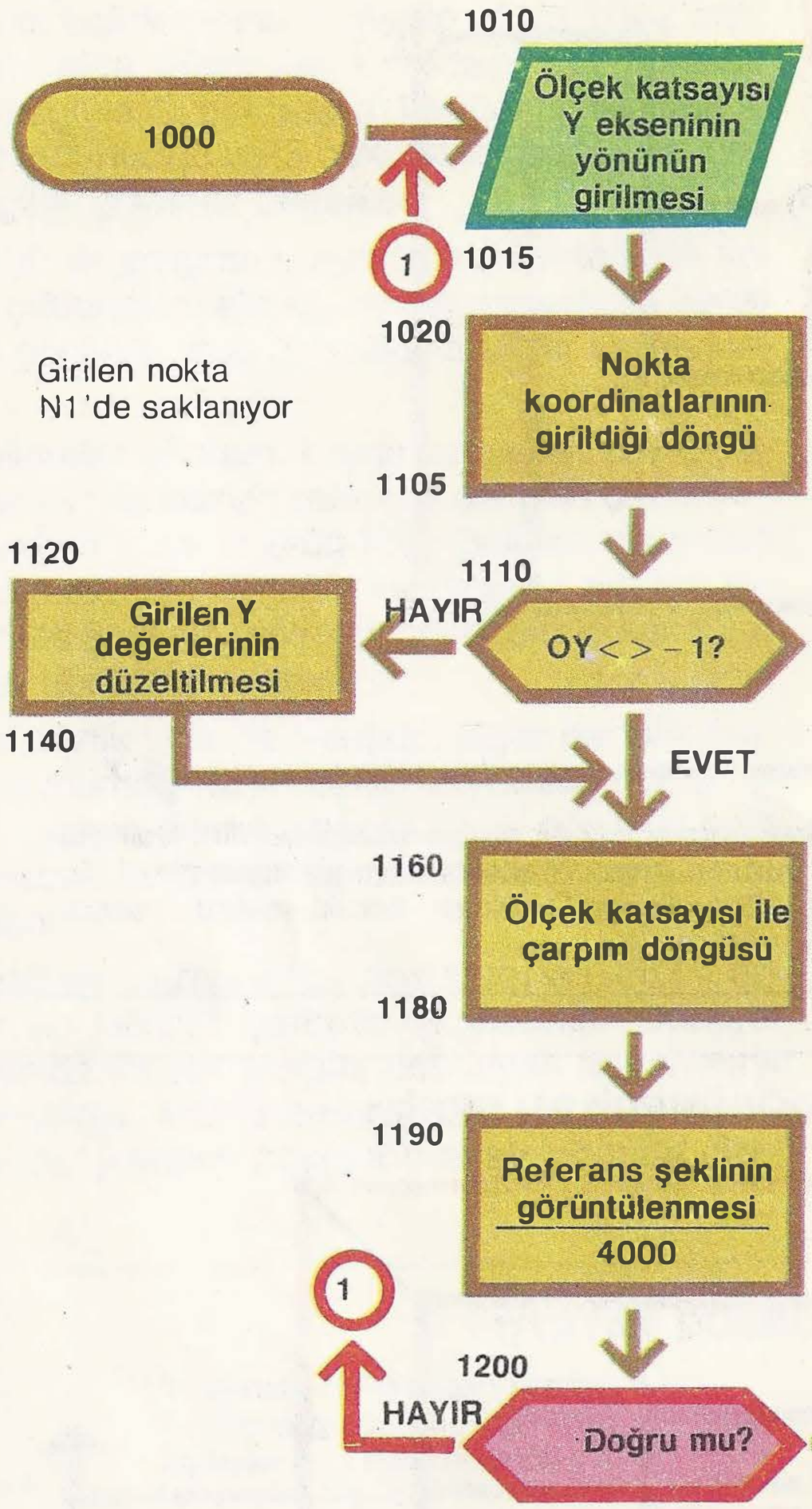
2/İÇ ÇİZGİLERİ DE GÖRÜNEBİLEN ŞEKİL



ramı). Bu durumda bile, şeklin bazı kısımlarında hata olabilmektedir (sayfa 1642'deki şeklin 2-a ve 6-b kısımları). Hata, kaydırlan noktaların, yalnızca temel şekle göre olan konumlarının incelenmesinden kaynaklanmaktadır.

5500 alt-programı, pencereleri içinde kalan noktaların belirlenmesinde kullanılan bir yöntemi sergilemektdir. 1646 ve 1657. sayfalarda, bütün sürecin akış şemaları ve program dökümleri verilmiştir.

REFERANS ŞEKLİNİN KAYDIRILMA VERİLERİNİ İÇEREN TABLONUN GİRİLMESİ



Girilen koordinatların herbiri FS ölçek katsayısıyla çarpılacak. Y ekseninin yönü (OY) alt-programın genelleştirilmesini sağlıyor.

Koordinatlar XP(1) ve YP(1)'de saklanırlar.
Döngü çıkışı 20 noktadan ya da daha örıce (0.0) koordinatlarının girmesiyle son bulur.

YP(1) = LY-YP(1)
LY, Y ekseninin kullanılan uzunluğudur.

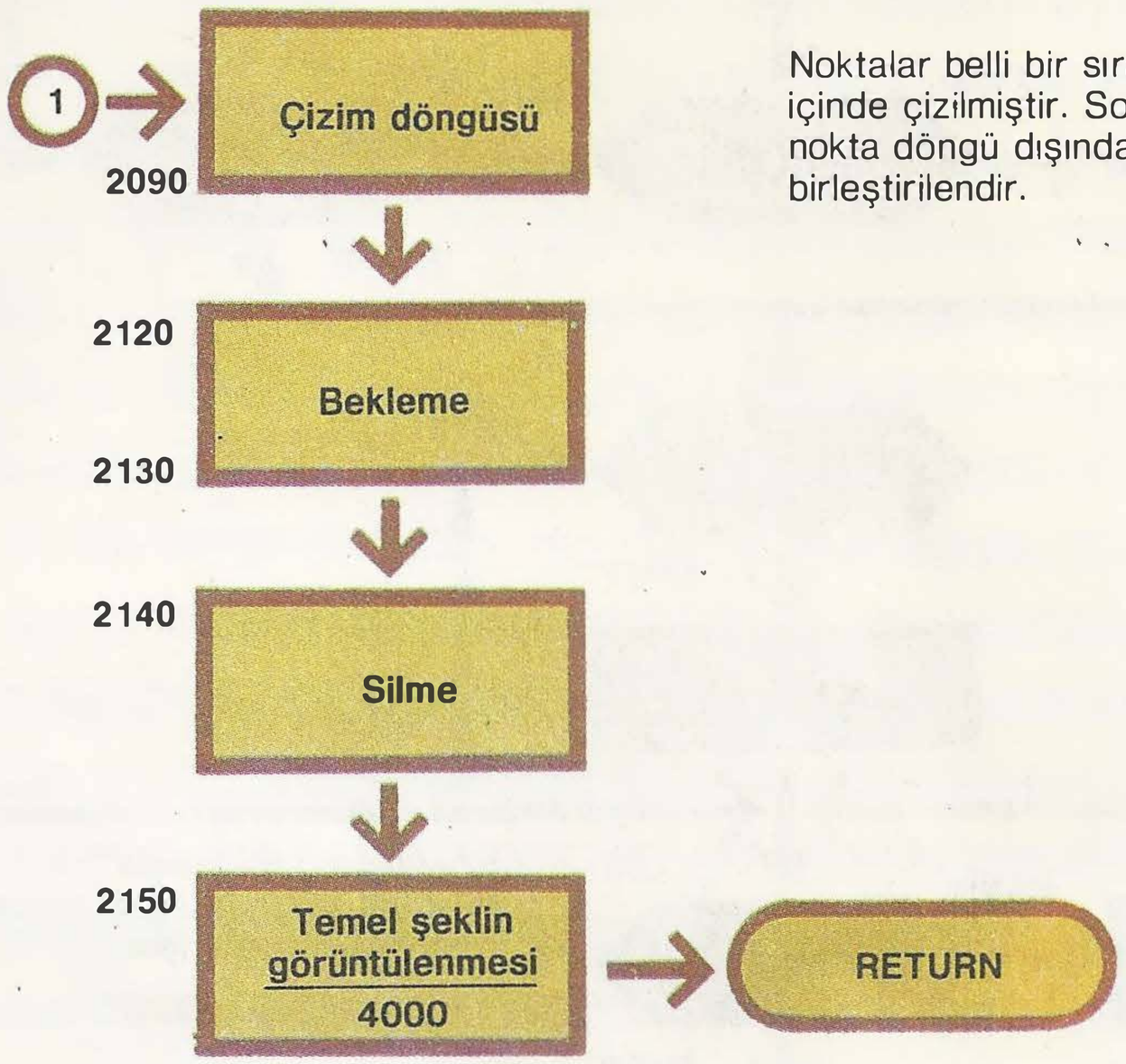
XP(I) = FS * XP(I)
YP(I) = FS * YP(I)

Bu görüntüleme yalnızca girilen değerlerin doğruluğunu kontrol etmeye yarar.

KAYDIRMADAN SONRA NOKTA KOORDİNATLARININ HESPLANMASI



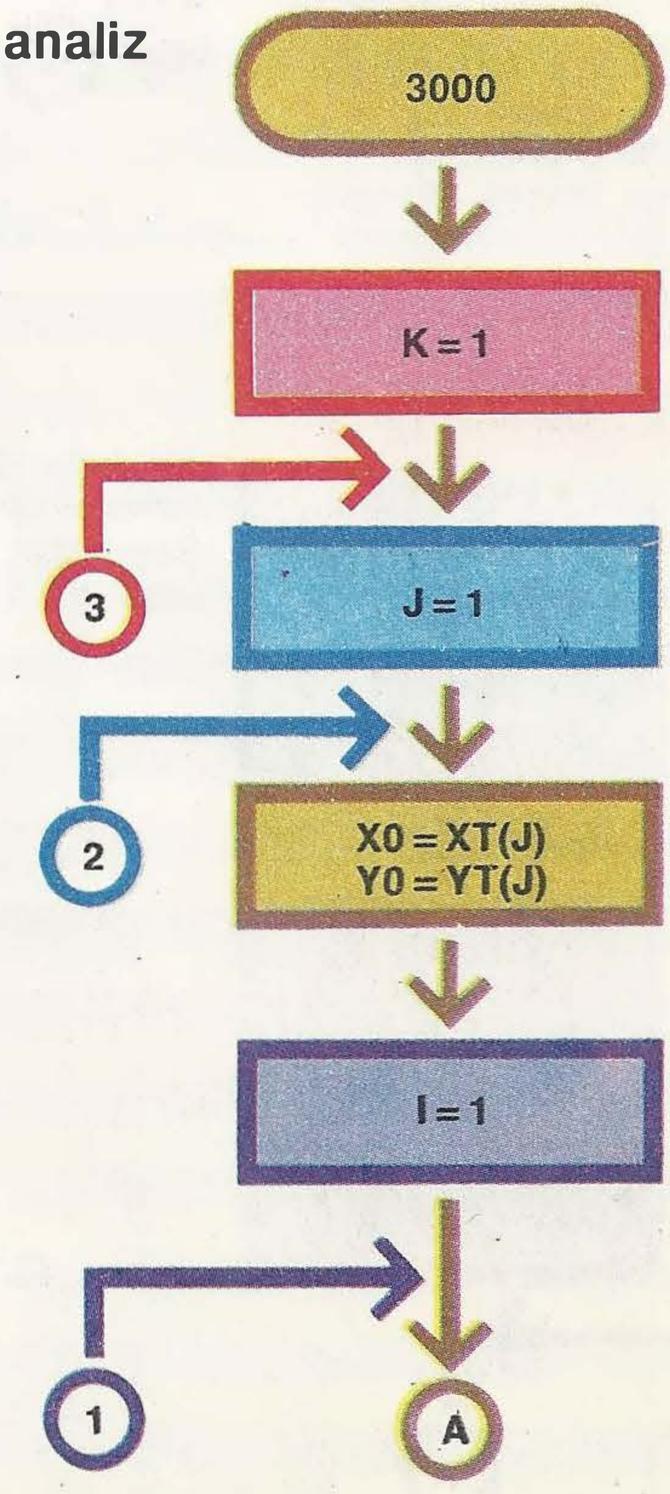
XT(I) = XP(I) + S
YT(I) = YP(I) + S
S cismin derinliğidir.
20 ekran noktasına eşit olduğu kabul edilir.



Noktalar belli bir sıra içinde çizilmiştir. Sonuncu nokta döngü dışında ilk birleştirilendir.

NOKTA TÜRÜNÜN BELİRLENMESİ

Yatay çizgi üzerinde analiz

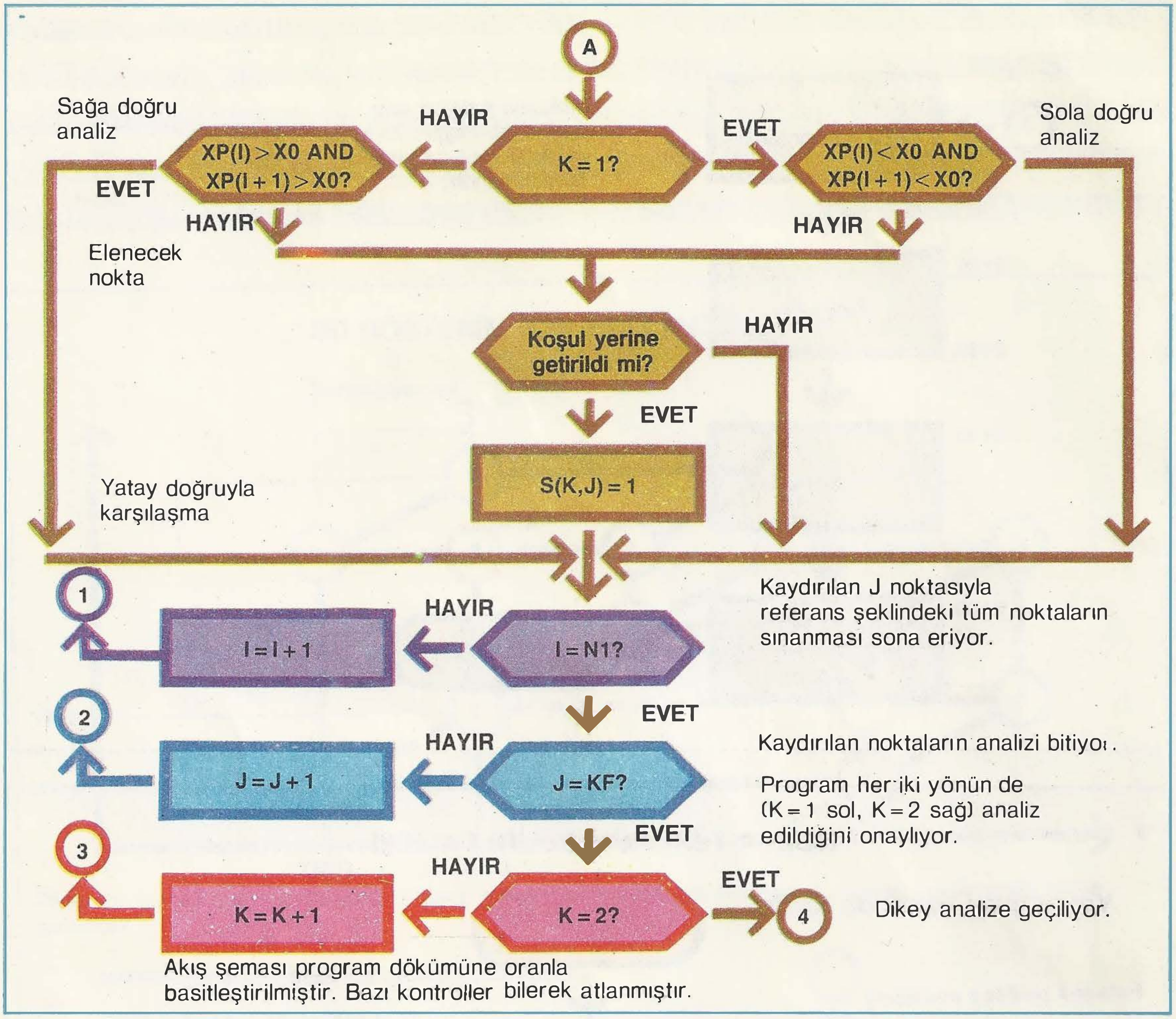


K, analizin bölgesini gösteren bir göstergedir.
K=1 sola, K=2 sağa

Analiz edilecek kaydırılmış noktaları seçen döngü

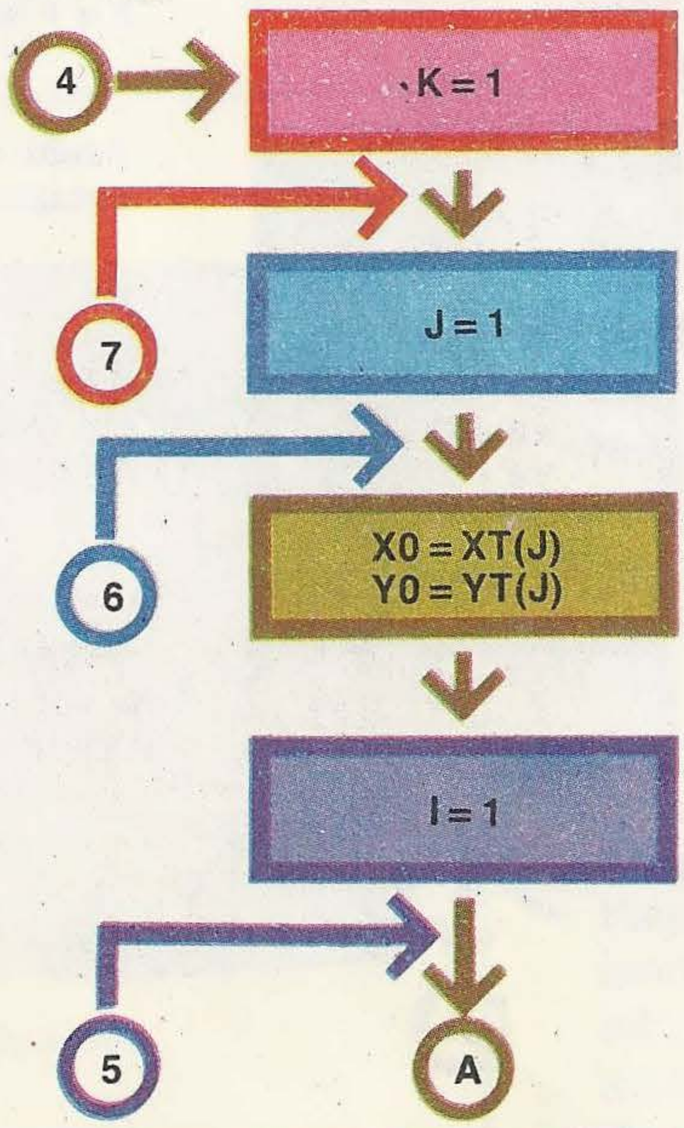
Analiz için kaydırılmış noktaların koordinatlarının çıkarılması

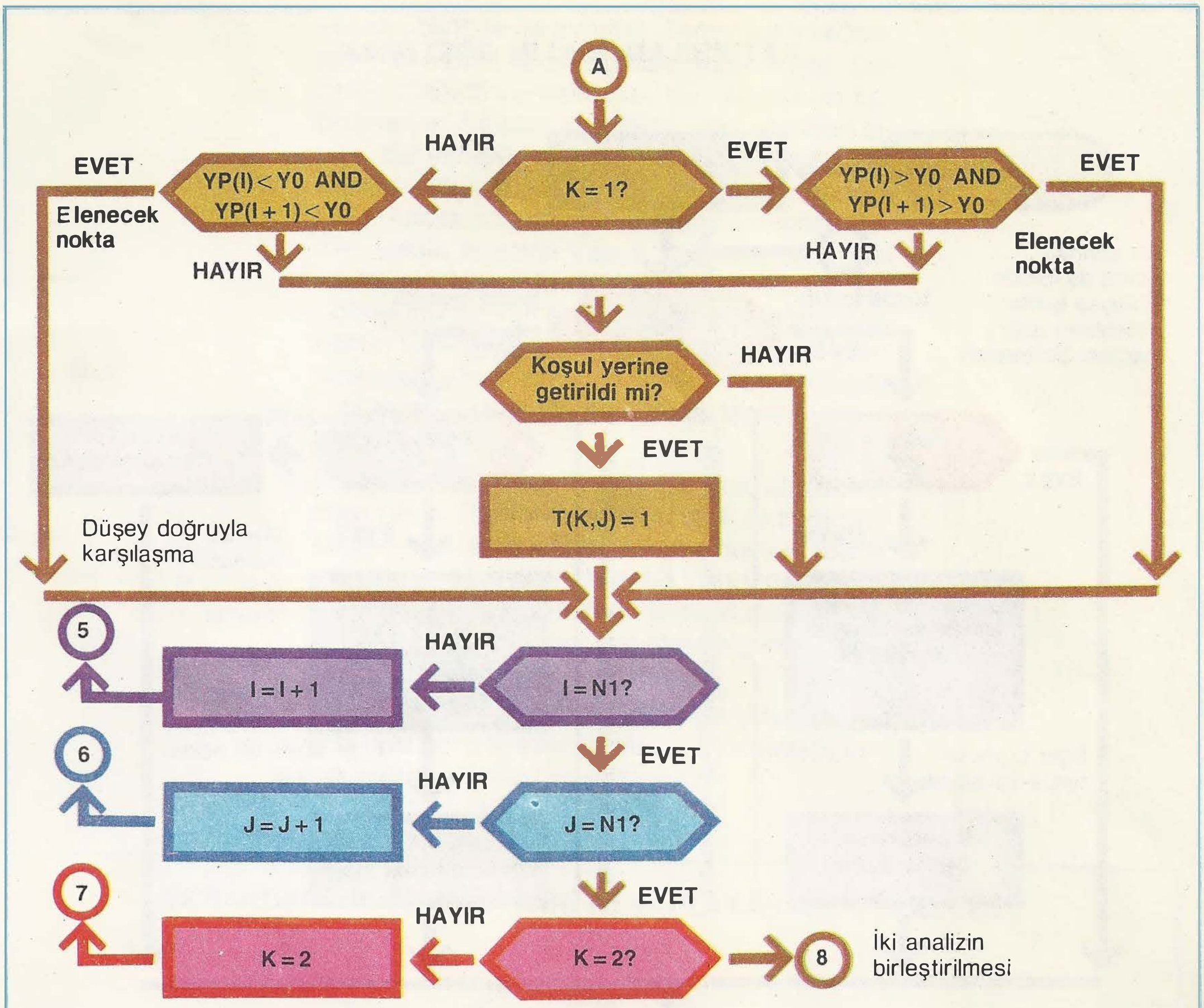
Referans şeklinin noktalarını ele alan döngünün başlangıcı



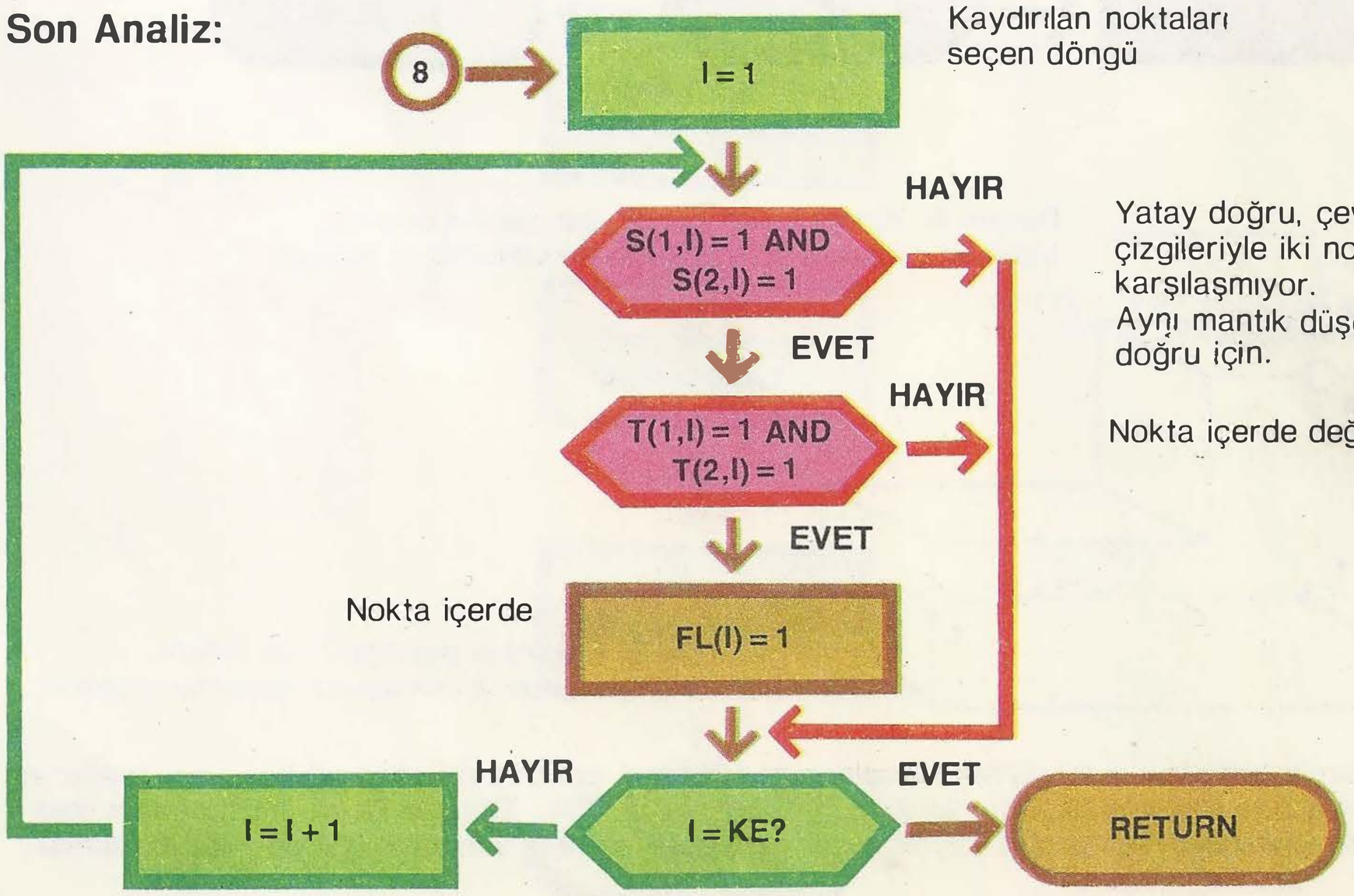
Akış şeması program dökümüne oranla basitleştirilmiştir. Bazı kontroller bilerek atlanmıştır.

Düşey doğru üzerinde analiz

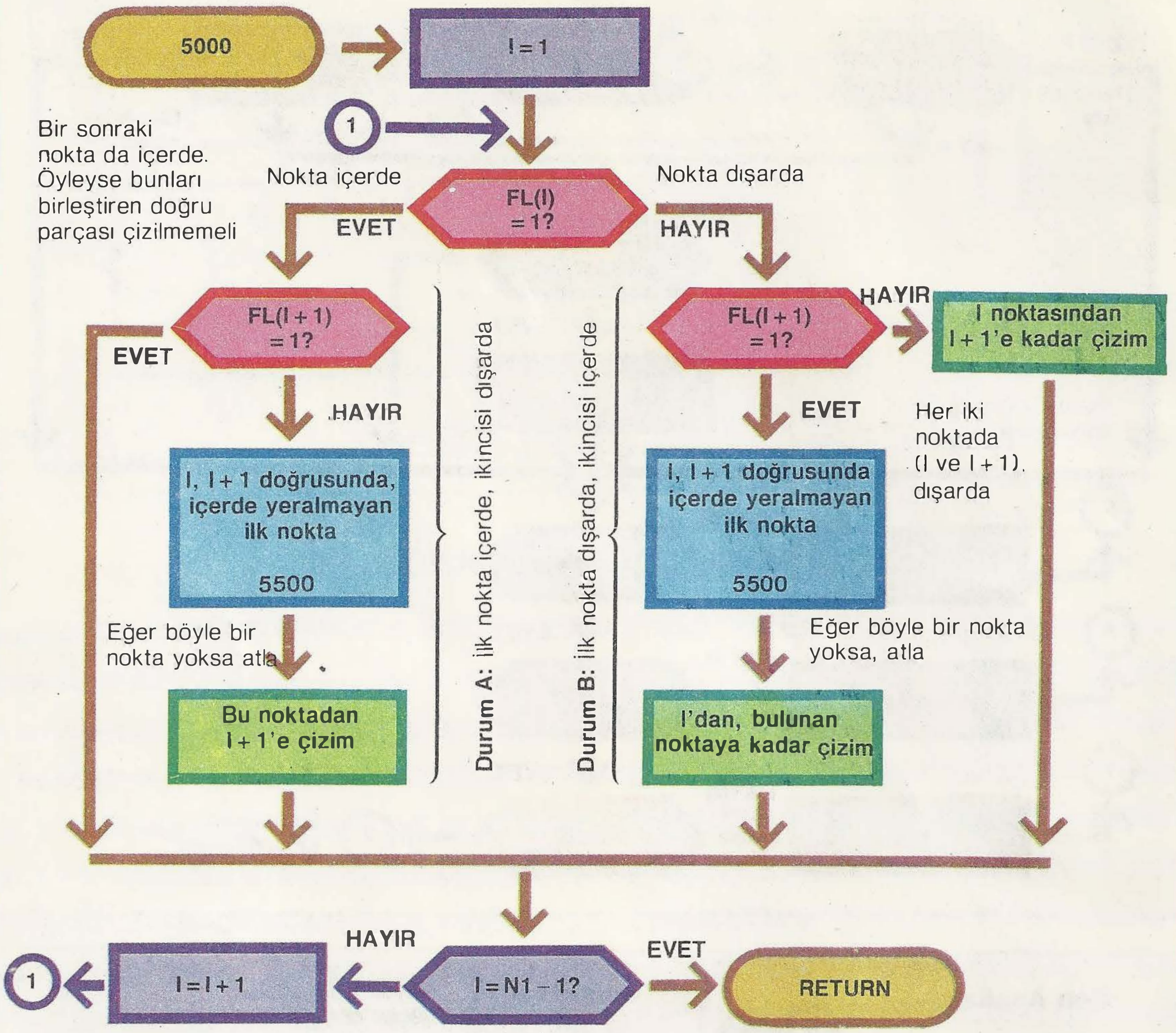


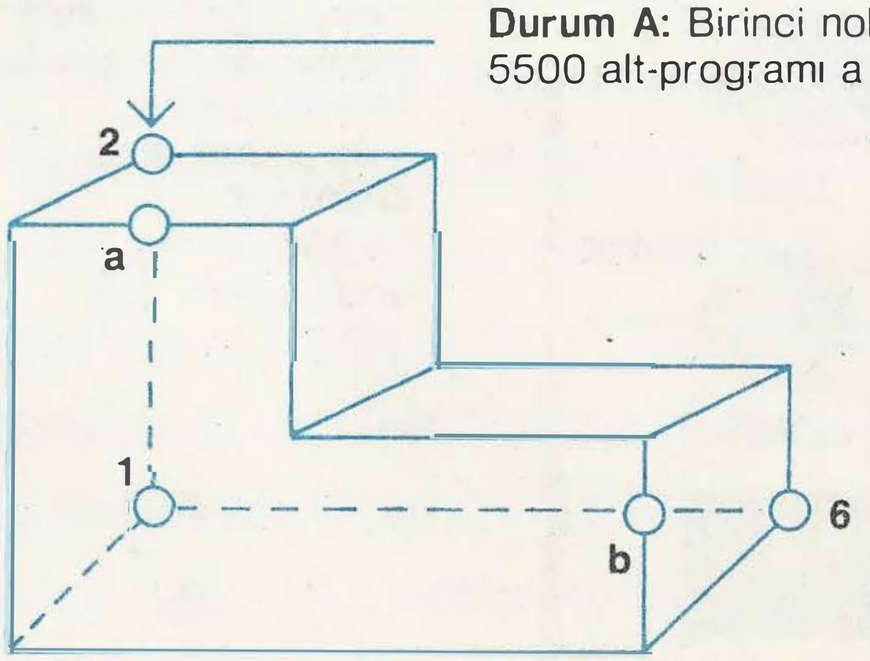

A
K=1?
HAYIR
EVET
YP(I)<Y0 AND YP(I+1)<Y0
EVET
Elenecek nokta
HAYIR
YP(I)>Y0 AND YP(I+1)>Y0
EVET
Elenecek nokta
HAYIR
Koşul yerine getirildi mi?
HAYIR
EVET
T(K,J)=1
Düşey doğruyla karşılaşma
5
I=I+1
HAYIR
I=N1?
6
J=J+1
HAYIR
EVET
J=N1?
7
K=2
HAYIR
EVET
K=2?
8
İki analizin birleştirilmesi

Son Analiz:


8
I=1
Kaydırılan noktaları seçen döngü
HAYIR
S(1,I)=1 AND S(2,I)=1
EVET
HAYIR
T(1,I)=1 AND T(2,I)=1
EVET
Yatay doğru, çevre çizgileriyle iki noktada karşılaşmıyor.
Aynı mantık düşey doğru için.
Nokta içerde değil.
Nokta içerde
FL(I)=1
HAYIR
I=I+1
I=KE?
EVET
RETURN

KAYDIRILAN ŞEKLİN GÖSTERİMİ



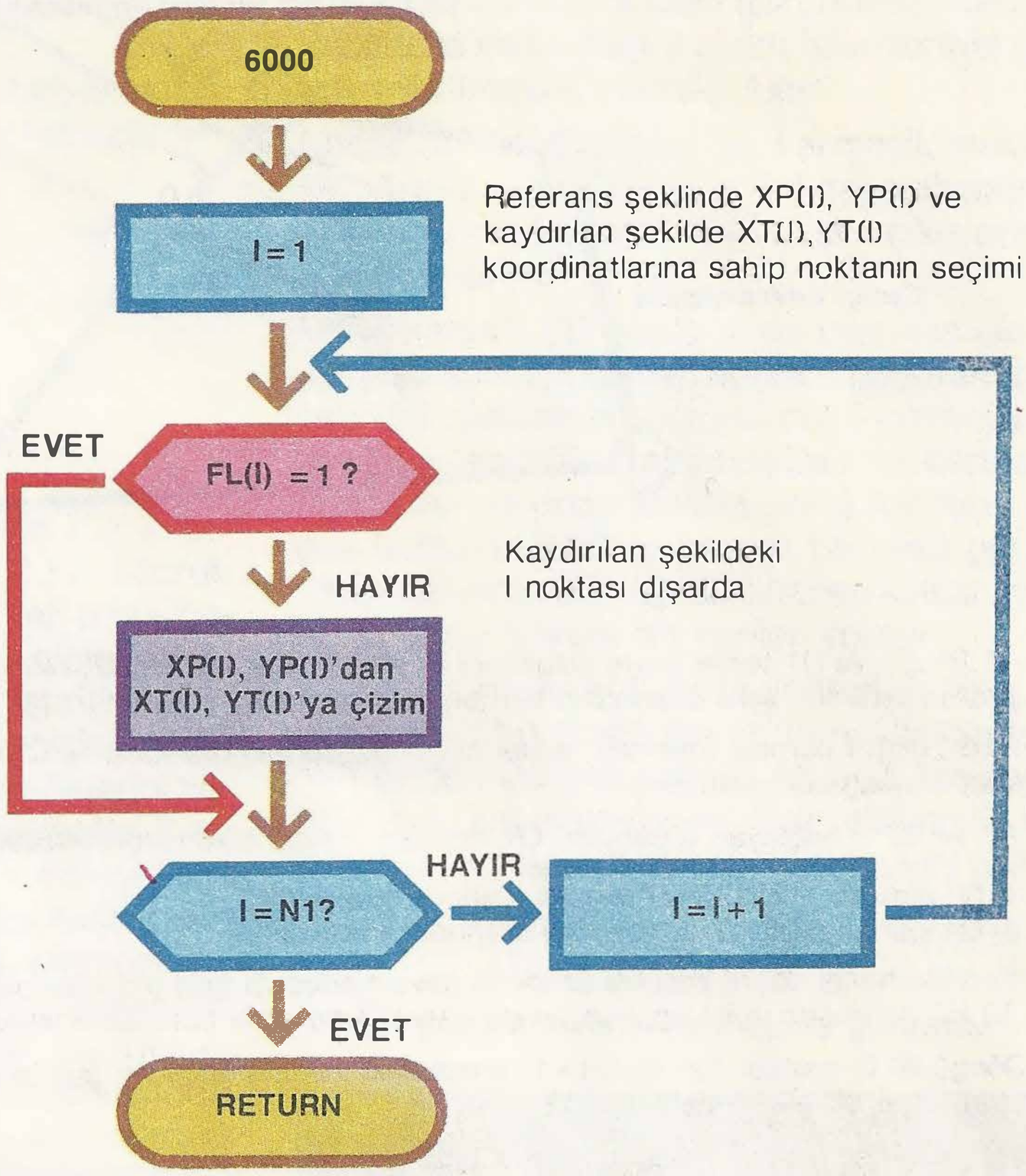


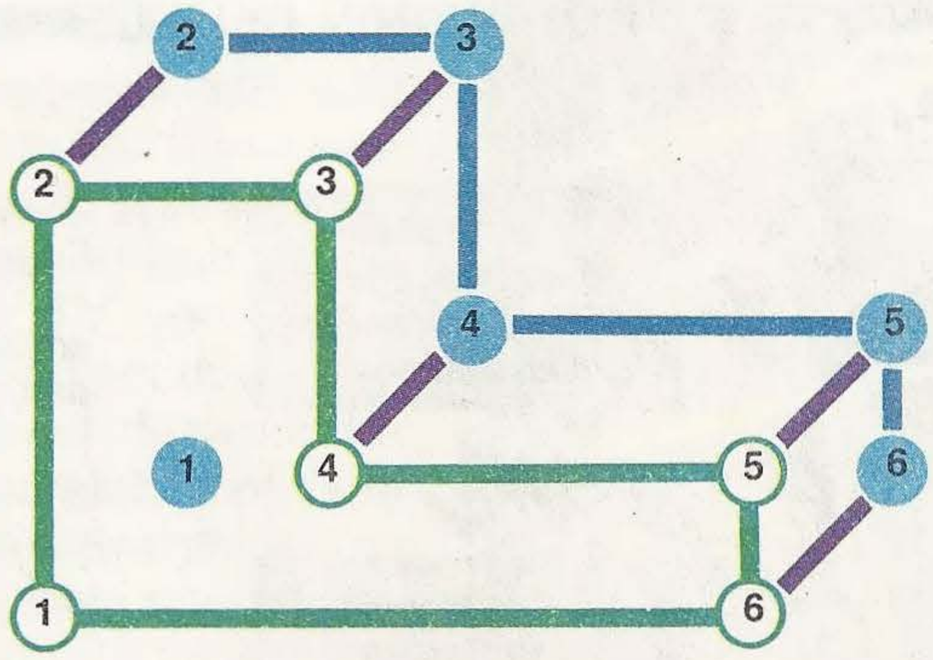
Durum A: Birinci nokta içerde, ikincisi ise dışarda.
5500 alt-programı a noktasının koordinatlarını belirler.

Durum B. 6 noktası dışarda, 1 ise içerde.
5500 alt-programı b noktasının konumunu belirler.

Burada sergilenen yöntem kaydırılan noktaların konumlarını yalnızca referans şekline göre inceler ve bu yüzden a,2 ve 6,b parçalarında olduğu gibi hataya yolaçabilir. Tümüyle doğru bir görünüm elde etmek için, kaydırılan her noktayı, kaydırmanın yolaçtığı yüzeylere yani derinliğe göre de incelemek gerekir.

BAĞLANTI ÇİZGİLERİ





Referans şekli

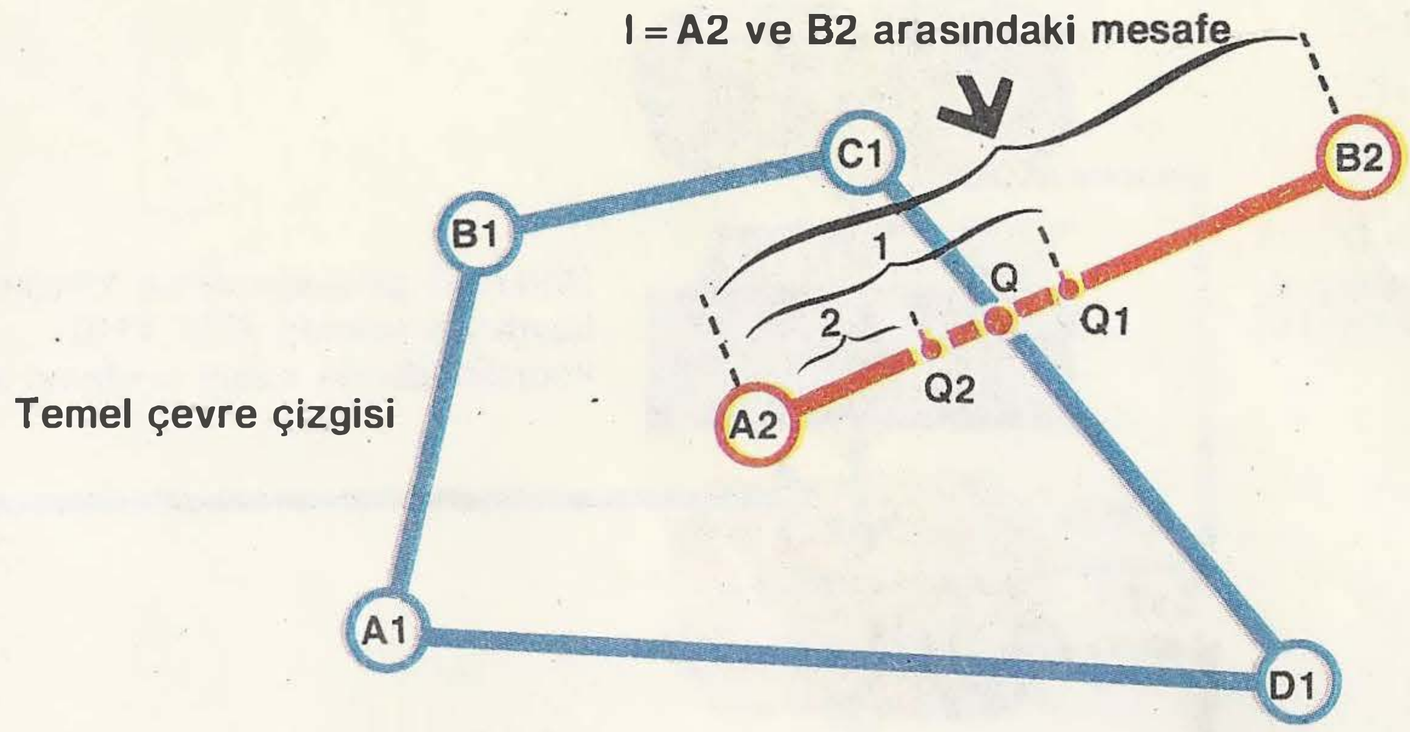
Kaydırılan noktalar XT(I), YT(I)

Kaydırılan şeklin çevre çizgilerinin 5000 alt-programı tarafından gösterimi (5500 alt-programının elemelerinden sonra)

Bu alt-programca çizilen bağlantı çizgileri

Her iki 1 noktası birbirine bağlanmamıştır, çünkü bu noktaların kaydırılan şekle ait olanı, temel çevre çizgilerinin içinde kalıyor.

5500 ALT-PROGRAMI TARAFINDAN KULLANILAN ALGORİTMA



A1,B1,C1 ve D1 temel çevre çizgilerini oluşturuyorlar. A2 ve B2, biri içerde, diğeri dışarda olmak üzere kaydırılan şekil üzerindeki herhangi bir noktadır.

A2B2 doğru parçası üzerinde, temel çevre çizgilerinin dışındaki ilk Q noktasının koordinatları bulunmalıdır.

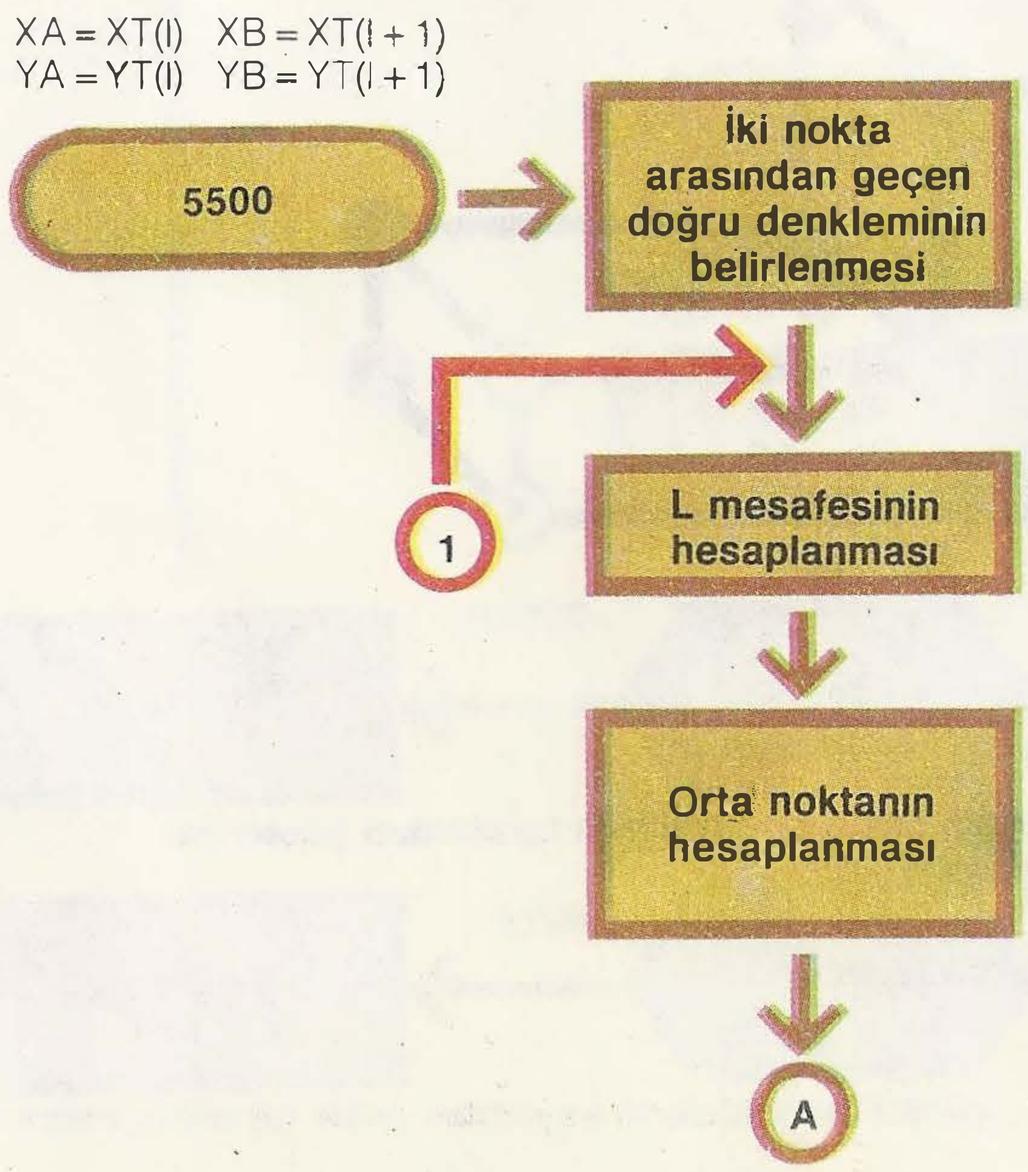
A2 ve B2'yi bağlayan L parçası, Q1 noktasıyla ikiye bölünüyor. Burada iki olasılık sözkonusu:

1- Q1 dışarda; analiz A2 Q1 parçası üzerinde devam eder.
2- Q1 içerde: analiz Q1 B2 parçası üzerinde devam eder.

Analizin hangi doğru parçası üzerinde devam edeceği belli olduktan sonra, program ilk adına dönerek A2,B2 doğrusunun iki ucunun yerine önceki adımlarda bulunan noktaları koyar.

Döngü iki Q noktası (Qn ve Qn + 1) arasındaki mesafe önceden belirlenen bir değerin altına indiğinde (yani istenilen hassasiyette) sona erer.

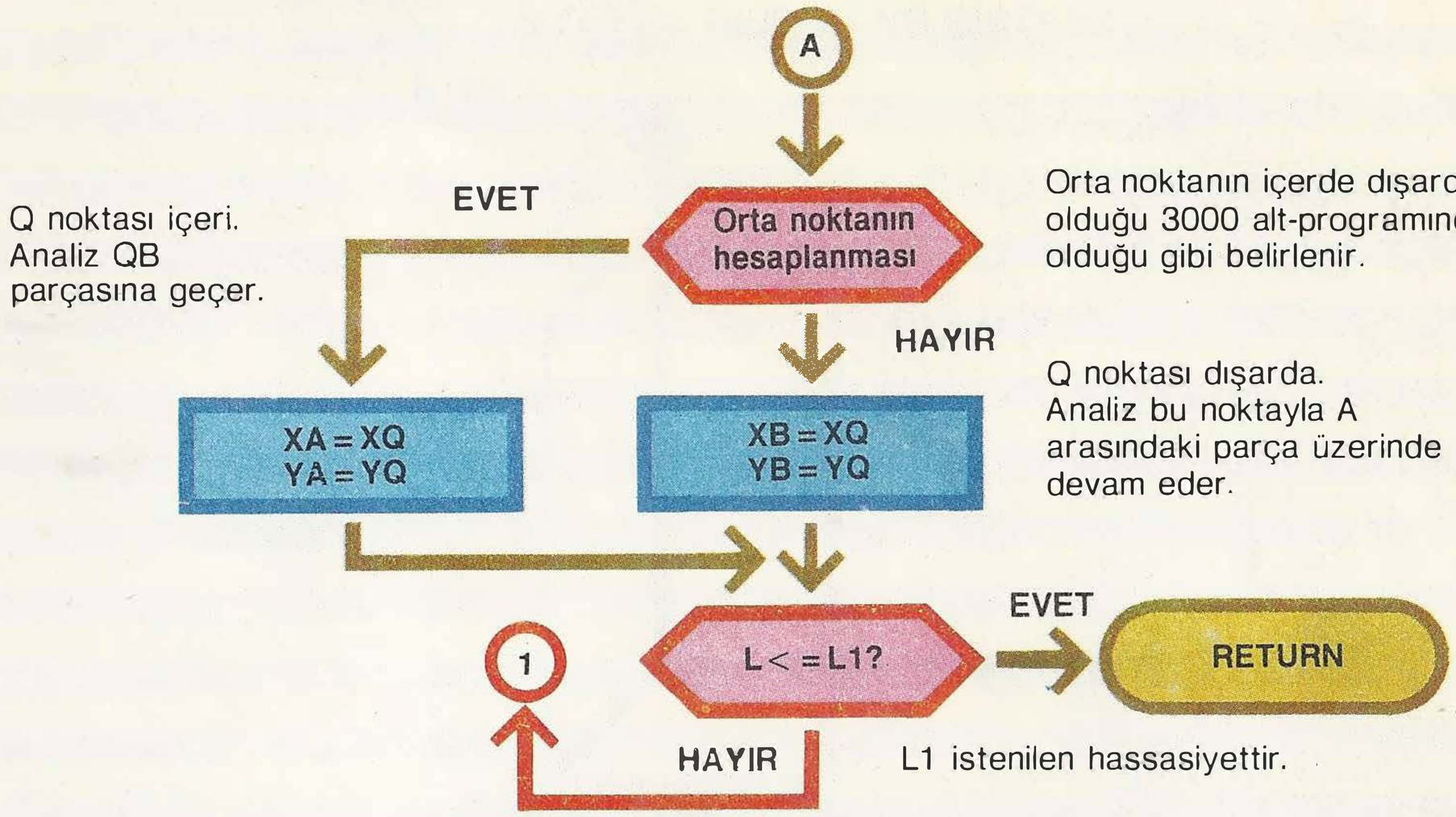
İÇ/ DIŞ GEÇİŞ NOKTASININ BELİRLENMESİ



Y = M * X + N
Çağırı sırasında XA,YA ve XB,YB değişkenleri analiz edilecek doğru parçasının iki ucunun koordinatlarını taşır.

L = SQR((XA – XB)∧2 + (YA – YB)∧2)

XQ = (XA + XB)/2
YQ = M * XQ + N



Q noktası içeri. Analiz QB parçasına geçer.

Orta noktanın içerde dışarda olduğu 3000 alt-programında olduğu gibi belirlenir.

Q noktası dışarda. Analiz bu noktayla A arasındaki parça üzerinde devam eder.

L1 istenilen hassasiyettir.

Burada sergilenen yöntem yaklaşık sonuçlar verir. Aslında, çizim açısından yeterli çözüm sağlanamadan istenilen hassasiyet (L1) elde edilip araştırma durabilir. Daha iyi bir çözüm yolu, ekran belleğinin okunmasından oluşan bir sisteme dayanır.

Burada, kısaca, A2,B2 parçası boyunca her noktanın ekran durumu belirlenmekte ve Q noktasının koordinatları, ekran noktasının aktif olduğu bir noktaya raslatılmaktadır. Bu yöntem çizim açısından en iyi sonucu verdiği halde daha fazla zaman gerektirmektedir. Çünkü gerçekte, A2,B2 doğrusu boyunca bütün noktalara karşılık gelen adreslerin içerikleri incelenmektedir. Ancak bu iki yöntemi optimum bir biçimde birleştirmek de mümkündür. Böylece ilk yöntem Q noktasının bulunduğu küçük aralığı belirledikten sonra, ikincisi ekran belleğinden hareketle kesin tespiti yapmaktır.

5500 alt-programı şunla değiştirilebilir: 5500 x Q = XB: YQ = YB: RETURN

Böylece, program yalnızca kısmi olarak çalıştırılır, buna karşın hata giderilmiş olur.

GİZLİ ÇİZGİLERİ SİLEN GRAFİK

```
1 REM ==========================
2 REM * 3D SEKILLERI     *
3 REM * ARKA PLANDAKI    *
4 REM * CIZGILERI        *
5 REM * SILEREK CIZEN    *
6 REM * PROGRAM          *
7 REM ==========================
8 :
9 :
50 S=20::L1=5:LY=191
55 DIM YP(20),XP(20),YT(20),XT(20)
      ,FL(20),S(2,20),T(2,20)
60 ONERR GOTO 10000
65 CH=1
100 GOSUB 1000
110 GOSUB 2000
120 GOSUB 3000
130 GOSUB 4000
140 GOSUB 5000
150 GOSUB 6000
200 HOME:VTAB 21:INPUT "BASKA CIZIM
    (E/H)";RS$
210 IF LEFT$(RS$,1)="E" THEN RUN
```

```
220 TEXT:HOME:VTAB 22:END
999 :
1000 REM ********************
1001 REM TEMEL SEKLIN GIRILMESI
1002 HGR
1003 :
1005 TEXT
1010 HOME:VTAB 10:INPUT "OLCEK KATSAYISINI
       GIRINIZ ";F$
1015 VTAB 10:HTAB 10:INPUT
     "GIRIS KOMUTLARI"
     ;OY
1020 I = 1
1030 VTAB 10:HTAB 10: PRINT
     "I NOKTASININ  KOORDINATLARI
1040 VTAB 10:HTAB 31:INPUT "X=
     ";XP(I)
1050 XP(I)*FS>279 THEN 1040

1060 VTAB 12:HTAB 31:INPUT "Y=
     ";YP(I)
1070 IF YP(I)*FS>191 THEN 1200

1080 IF YP(I)=0 OR XP(I)=0 THEN 1100

1090 IF I=20 THEN 1100
1095 I=I+1:GOTO 1030
1100 N1=I-1
1105 XP(N1+1)=XP(N1):YP(N1+
     1)=YP(N1)
1110 IF OY<>1 THEN 1160
1120 FOR I=1 TO N1
1130 YP(I)=LY-YP(I)
1140 NEXT I
1150 FOR I=1 TO N1
1170 XP(I)=FS*XP(I):YP(I)=F
     S*YP(I)
1180 NEXT I
1190 GOSUB 4000
1200 VTAB 22:INPUT "CIZIM DOGRU MU?
     (E/H)";RS$
1210 IF LEFT$(RS$)="E" THEN
     RETURN
1220 GOTO 1000
2000 :
2001 REM ==========================
2002 REM * KAYDIRILAN NOKTALARIN *
2003 REM *    HESAPLANMASI        *
2004 REM ==========================
2005 :
2010 FOR I=1 TO N1
2020 XT(I)=S+XP(I):YT(I)=YP
     (I)-S
2030 NEXT I
2035 XT(I)=XT(N1)+S:YT(I)=Y
     T(N1)-S
2040 HOME:VTAB 22:INPUT "KAYDIRILAN SEKIL
       GOSTERILSIN MI? (E/H)";RS$

2050 IF LEFT$(RS$,1)="E" THEN
     2070
2060 RETURN
2070 :
2090 FOR I=1 TO N1-1
```

```
2100 HPLOT XT(I),YT(I) TO XT(I+1)
     ,YT(I+1)
2110 NEXT I
2120 HPLOT XT(N1-1),YT(N1-1) TO
      XT(N1),YT(N1)
2130 VTAB 22:PRINT  SILMEK ICIN HERHANGI
      TUSA BASINIZ";GET Q$

2140 HGR
2150 GOSUB 4000
2160 RETURN
2999 :
3000 REM ----------------------
3001 REM NOKTA TURUNUN
3002 REM BELIRLENMESI
3003 REM ----------------------
3004 :
3005 KE=N1:HOME:VTAB 22:HTAB 10
     :PRINT "BIR DAKIKA LUTFEN, SVP"
3006 REM ----------------------
3007 REM : YATAY DOGRU        :
3008 REM : INCELEMESI         :
3009 REM ----------------------
3010 FOR K=1 TO 2
3020 FOR J=1 TO KE
3025 IF KE=1 THEN 3040
3030 XO=XT(J):YO=YT(J)
3040
     IF K=1 THEN 3070
3060 IF XP(I)>=XO AND XP(I+1)<=XO
      THEN 3100
3065 GOTO 3080
3070 IF XP(I) = < XO AND XP(I +
     1) < = XO THEN 3100
3080 IF (YP(I)<=YO AND YP(I+1)>=YO)
      OR (YP(I)>=YO AND YP(I+1)<=TO)
      THEN S(K,J)=1
3100 NEXT I,J,K
3105 :
3106 REM ----------------------
3107 REM : DUSEY DOGRU        :
3108 REM : INCELEMESI         :
3109 REM ----------------------
3110 FOR K=1 TO 2
3120 FOR J=1 TO KE
3125 IF KE=1 THEN 3140
3130 XO=XT(J):YO=YT(J)
3140 FOR I=1 TO N1
3150 IF K=1 THEN 3170
3160 IF YP(I)>=YO AND YP(I+1)>=0
      THEN 3200
3165 GOTO 3180
3170 IF YP(I)<=YO AND YP(I+1)<=YO
      THEN 3200
3180 IF (XP(I)<=XO AND XP(I+1)>=XO)
      OR (XP(I)>=XO AND XP(I+1)<=XO)
      THEN T(K,J)=1
3200 NEXT I,J,K
3205 :
3206 REM ----------------------
3207 REM :  SON COZUMLEME      :
3208 REM ----------------------
3210 FOR I=1 TO KE
3220 IF S(1,I)=1 AND S(2,I)=1 THEN
       3240
```

```
3230 GOTO 3250
3240 IF T(1,I)=1 AND T(2,I)=1 THEN
       FL(I)=1
3250 NEXT I
3260 RETURN
3999 :
4000 REM ==========================
4001 REM * TEMEL SEKLIN          *
4002 REM *     GOSTERIMI          *
4003 REM ==========================
4004 :
4005 HGR:HCOLOR=3
4010 FOR I=1 TO N1-1
4020 HPLOT XP(I),YP(I) TO XP(I+1)
     ,YP(I+1)
4030 NEXT I
4035 GOTO 4060
4040 HPLOT XP(N1-1),YP(N1-1) TO
      XP(N1),YP(N1)
4060 RETURN
4999 :
5000 REM ==========================
5001 REM * KAYDIRILAN SEKLIN    *
5002 REM *   GOSTERIMI          *
5003 REM ==========================
5004 :
5010 FOR P=1 TO N1-1
5020 IF FL(P)=CN THEN 5200
3030 IF FL(P+1)=CN THEN 5050

5040 HPLOT XT(P),YT(P) TO XT(P+1)
     ,YT(P+1)
5045 GOTO 5300
5050 XB=XT(P):XA=XT(P+1):YB=YT(P)
     :YA=YT(P+1)
5055 GOSUB 5500
5060 HPLOT XT(P),YT(P) TO XQ,YQ
5070 GOTO 5300
5100 :
5200 IF FL(P+1)=CN THEN 5300

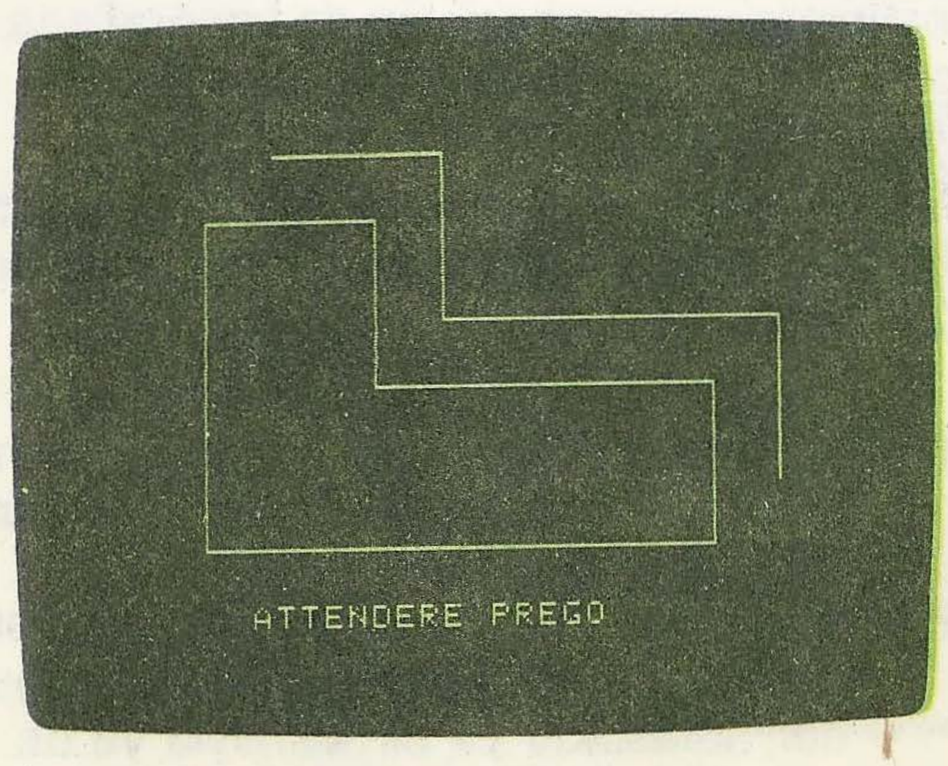
5210 XB=XT(P):XA=XT(P+1)
5211 YA=YT(P):YB=YT(P+1)
5215 GOSUB 5500
5220 HPLOT XQ,YQ TO XT(P+1),YT
     (P+1)
5300 NEXT P
5310 RETURN
5499 :
5500 REM ==========================
5501 REM * IC/DIS                 *
5502 REM * GECIS NOKTALARI       *
5503 REM ==========================
5504 :
5505 IF XA=XB THEN X=1:GOTO
     5515
5506 REM ----------------------
5507 REM : YQ=XQ*M+N      :
5508 REM ----------------------
5510 X=XB-XA
5515 M=(YB-YA)/X:N=INT(
     YA-(XA*M))
5520 L=INT(SQR((XA-XB)**2+(YA-YB)**2))
```

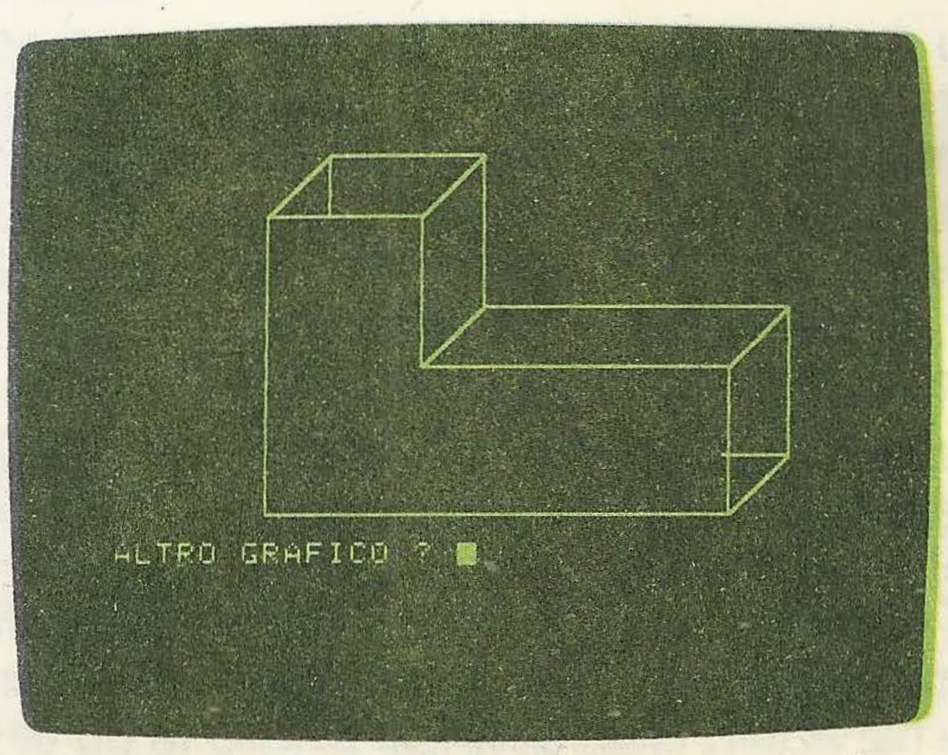
```
5525 IF X = 1 THEN XO = XA:YO =
     INT ((YA + YB) / 2): GOTO 5
     540
5530 XO = INT ((XA + XB ) / 2):YO
     = INT (M * XO + N)
5540 XO = XO:YO = YO
5545 S(1,1) = 0:S(2,1) = 0:T(1,1)
     = 0:T(2,1) = 0
5550 CF = FL(1):KE = 1:FL(1) = M:
     GOSUB 30100
5580 IF FL(1) = CN THEN XA = XO:
     YA = YO:GOTO 5600
5590 XB = XO:YB = YO
5600 FL(1) = CF
5605 IF L < = L1 THEN SW = 0:RETURN

5610 GOTO 5520

59999 :
60000  REM ==========
60001  REM * FONKSIYON *
60002  REM * SATIRLARI *
60003  REM ==========
60004  :
60010  FOR I = 1 TO N1
60020  IF FL(I) = CN THEN 60040
60030  HPLOT XP(I),YP(I) TO X1(I),
       Y1(I)
60040  NEXT I
60050  RETURN
100000 :
100001  REM ========
100002  REM * HATA  *
100003  REM ========
100004 :
100005  TEXT: HOME
100010 ER = PEEK (222)
100020 IF ER = 55 THEN VTAB 10:PRINT
       "HATALI KOORDINATLAR ":GOTO 1
       00050
100030 VTAB 10:PRINT "HATALI VERILER
       "
100050 VTAB 22: HTAB 30:PRINT "
       >";
100060 GET G$: RUN
```

Verilen programın işleyişini gösteren bir örnek. Solda, program referans şeklinin koordinatlarını incelemiş, kaydırmayı yapmış ve görünen bütün çizgileri görüntülemiştir. Sağda çizim tamamlanmıştır. Hatalı iki doğru parçasına dikkat edelim.





Bilgisayarla Resim Canlandırma (Animasyon)

Resim canlandırmanın bilgi-işlemin çizime yönelik alanında özel bir yeri vardır. Canlandırma (animasyon) teknikleri bilgisayar oyun programlarında zaten yaygın olarak kullanılmaktadır. Ayrıca resim canlandırmanın bilimsel bir problem çözümü için gerektiği örneklere çok raslanır. Bir kuvvetin bir cisme yaptığı etkiyi göstermek veya bir mekanizmanın hareketlerini incelemek bu örnekler arasında sayılabilir.

Hareketli Cisimler

Daha önce ekranda bir çizım gerçekleştirmenin kolay bir iş olmadığını gördük. Bu tür bir görüntü yaratabilmek için çok sayıda doğru parçalarına gerek duyulmaktadır. Dolayısıyla bir cismi hareket ettirmek için bu parçaların her defasında silinmesi ve yeni konumlarına göre yeniden çizilmesi gerekir.

''Hareketli cisim'' (İngilizcede **sprite** yani **"ekran cini"** denir) ise tersine biçimi ve ekrandaki çeşitli konumları seçildikten sonra program tarafından olduğu gibi yönetilen bir şekildir. Bir televizyon ekranının boyutlarıyla sınırlanan hareketli cisimlerin üç ana özelliği vardır:

— Kullanıcı tarafından tümüyle programlanabilir
— Bir plan içinde her dört yöne de hareket ettirilebilir
— Ekran üzerindeki diğer hareketli cisimler veya başka cisimlerle olan çarpışmaları hatırlarlar.

Bu özelliklerden her biri ayrı bir alt-program tarafından kazandırılır. Aynı alt-programlar resim canlandırmaları öngörülmemiş sistemlerde de kullanılabilir. Ancak özel hareket ettirme, çarpıştırma vb. komutları içeren makineler de vardır. Bunlar standart BASIC'te bulunmayan bazı özelliklerle güçlendirilmişlerdir.

Hareketli cisimler genellikle ilgili donanım bileşenleriyle yönetilir ve yönlendirilir. Daha ilkel makineler için ise bu bileşenlerin işlevlerini yüklenecek programlar yazmak gerekir. Ama bu takdirde hem yürütülecek işlevler hem de hız azalır. Hele çeviricili BASIC'te bu ikisi iyice düşer.

Hareketli Cisimlerin Yazılımla Yönetilmesi

Hareketli cisimlerin kullanımı büyük ölçüde hareketli çizgilerin üretimine bağlıdır. Programla hareketli cismin yönlendirilmesi sırasında karşılaşılan zorluklardan bir kısmı doğrudan bu soruna bağlanabilir. Hareketli bir cismin hareketi, bir şekiller tablosu yaratılıp sonra tablonun çizim merkezi değiştirilerek sağlanabilir. Çok yaygın olan bu teknik, hareket edecek cismi merkeze göre bir dizi uzaklık parametresiyle oluşturmaya dayanır.

Referans olarak seçilen noktanın koordinatları değiştirildiğinde cisim de yer değiştirmiş olur (tabii daha önceki görüntüyü silmeyi unutmamak gerekir).

Yandaki sayfada klavyeden girilen komutlara uyarak ekran üzerindeki bir cismin yerini değiştiren bir programın akış şeması görülüyor. Programın çeşitli aşamaları anında görüntüleniyor. A alt-programının hem silme hem de yeniden görüntüleme işlevlerini birlikte yürüttüğüne dikkat etmek gerekir. Çalıştırmak için bu iki işlevden hangisinin istendiğini belirtir bir işaret verilmesi yetiyor.

Akış şemasında yine yalnızca bu uygulamaya özgü iki öbek (zemin ve çarpışma) yer alıyor.

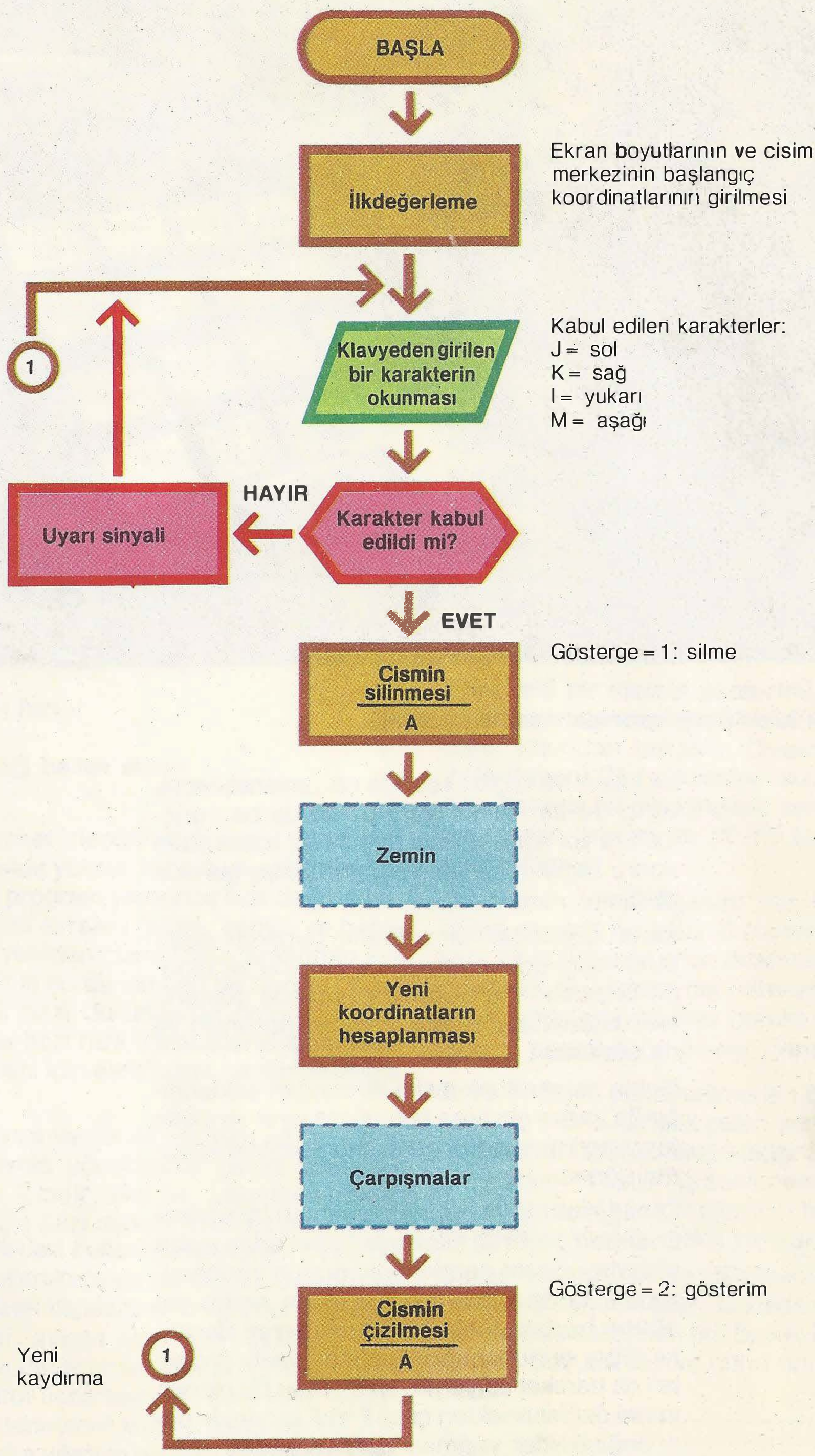
Resim canlandırma. Bilgisayarla canlandırma tekniği sinemada kullanılana çok yakındır. Sahne, hareketli cismin aralarında dolaştığı sabit cisimlerden (zemin) oluşmaktadır.

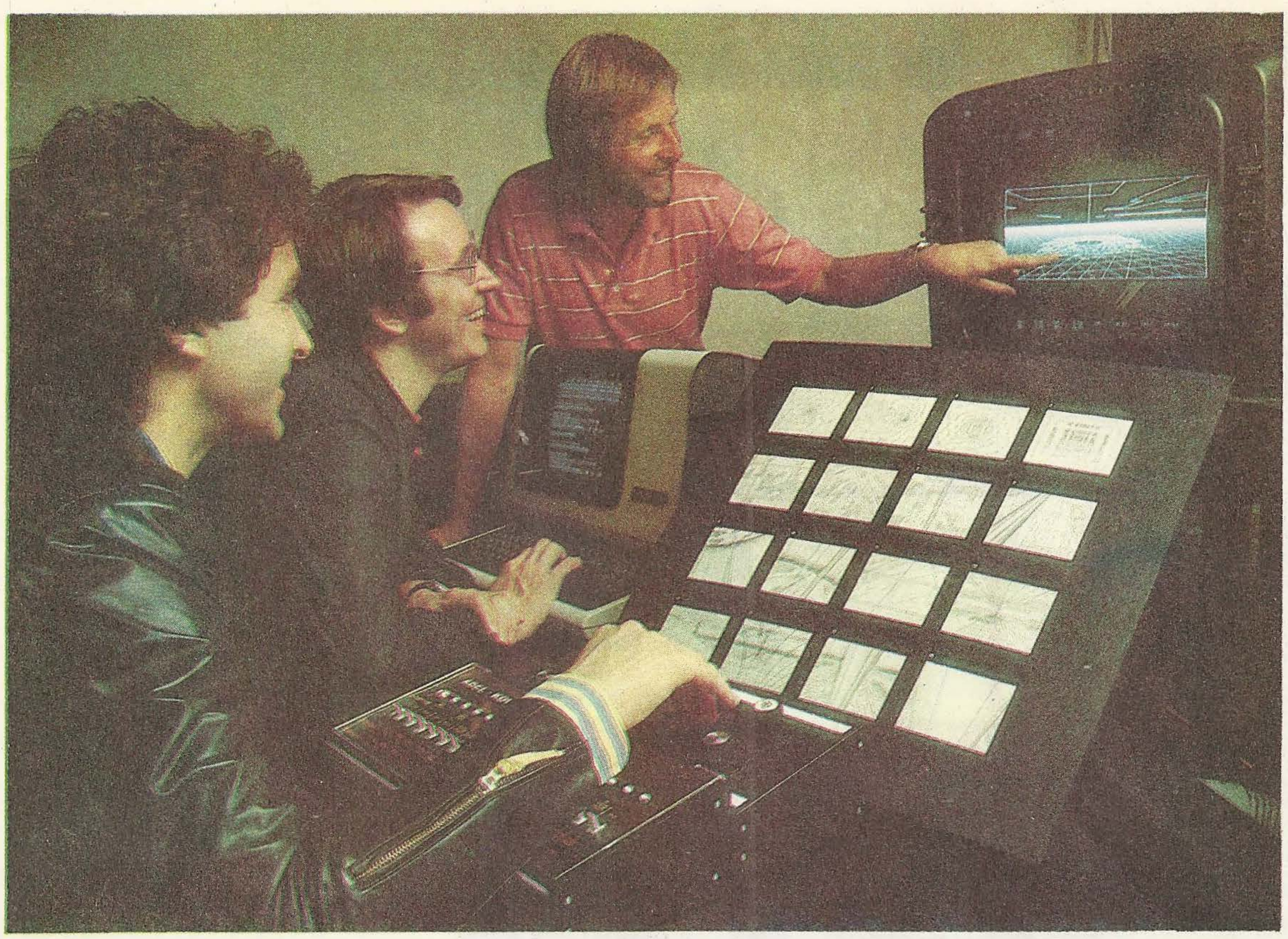
Sinemada kullanılan teknik saydam bir ortamın üzerine zeminin mevcut kare sayısı kadar basılmasından (veya çizilmesinden) oluşur. Bu karelerin her birinde hareket eden cisim bir öncekinden ve bir sonrakinden biraz farklı olarak yer alır. Çeşitli tablolar zemin üzerine yerleştirilerek ve ara geçişleri de fotoğraflanarak projeksiyon sırasında gerçek bir hareket zemini elde edilir.

Aslında sorun sanıldığı kadar kolay değildir. Burada hareketli cismin iki şekilde yönetileceğini belirtelim:

1 / Zemine göre hareket sırasında cismin kendisi sabit kalır; bu durumda yalnızca konumunu değiştirmek yeterlidir.
2 / Cisim hem zemine göre, hem kendisine göre hareketli olabilir: Bu durumda cismi her defasında ya bir kısmıyla ya da tümüyle yeniden çizmek gerekir.

BİR CİSMİN (SPRITE) KAYDIRILMASI





TRON filminin hazırlanma aşamalarından bir görüntü.

Bir kalem ve kağıtla kolayca çözümlenebilecek ikinci durum, bilgisayar düzeyinde bazı güçlüklere yol açar. Birinci güçlük bir hareketten sonra zeminin yeniden oluşturulması gerekliliğidir.

Ekran üzerindeki bir noktanın hareketli cisim tarafından dokunulmadıkça zemine ait olduğu farzedilir. Bu durumda zeminde gösterilmek istenene göre nokta yanık (görünür) ya da sönüktür.

Nokta hareketli bir cisim tarafından saklandığında, cismi çizdiren komutlara uyar ve cisim tamamen üzerinden geçtiğinde baştaki konuma döner.

Örneğin ekranda gezinen bir insan bir ağacın önünden geçerse bilgisayar adım adım insan silüetini ağacın gerekli bölümünün yerine çizecek, insan tamamen geçtikten sonra ise ağacın baştaki bütün görüntüsü yerini alacaktır. Eğer bunu yanında insanın çeşitli organları da hareket ediyorsa, artık sorun bir resmin yerini değiştirmekten çıkıp kendi içinde de bazı değişiklikler yapması gerekecektir.

Canlandırma yapabilen sistemlerde zemin ma-

kine tarafından işlenir. Hareketli cisim de zemine göre yerleştirilebilir.

Buraya kadar elde edilen durağan bir cismin yer değiştirmesidir. Ancak daha gerçekçi bir yerleştirme için şeklin kendisinde de bazı düzenlemeler yapmak gerekir.

Hareketli cisim tekniğinde iki olanak vardır: Yer değiştirmeden önce hareketli cisimde değişiklik yapmak veya hareketli cismin muhtemel bütün konumlarını yarattıktan sonra bunları birbiri ardından görüntülemek. Bu durumda farklı şekillerden her biri ayrı bir cisim gibi ele alınırlar. Yani bu sistem geleneksel canlandırma tekniğine tıpatıp benzemektedir. Kullanıcı tarafından yazılan programlarla resim canlandırmada da aynı sorunlar mevcuttur. Bir yandan zemini yeniden oluşturan bir alt-program öngörülmeli diğer yandan da çarpışmalar kontrol edilmelidir. Bu tür kullanıma yönelik makinelerde bu sonuncu otomatik olarak halledilir.

Zemin yönetimi. Resim canlandırmada zemin yönetimi değişik yöntemlerle gerçekleştirilebilir. Bu yöntemlerin gelişmişlik derecesi temelde iki parametreyle ölçülebilir:

- Yürütme hızı
- İşgal ettiği bellek alanı

Sürekli hareket izlenimi verebilmek için resim canlandırmada yüksek hızlara ulaşabilmek gerekir. Eğer program yeterince hızlı değilse hareketin çeşitli evreleri (silme, zemini ve hareketli cismi yeniden çizme) göz tarafından ayrımsanabilir ki bu da istenen bir durum değildir. Üstelik bazı durumlarda örneğin video oyunlarında bazı hızlı hareketlerin anlamlarını yitirmemesi için aynı hızda canlandırılması gerekir.

Bundan sonra verilecek yöntem ve örnekler yalnızca zemin yönetiminin çeşitli yönlerini sergilemek içindir. Bunlar, yeterince genel olabilmek için bazı makinelerin donanımlarıyla ilgili özellikleri kullanmamaktadır. Oysa bu özellikler kullanıldığında gerek yürütme hızı ve gerekse bellek kapasitesiyle ilgili sorunlar çözümlenebilir. Ayrıca yeterince yüksek bir çalışma hızına uaşabilmek için programın yazılım ve kontrol aşamalarında çeviriciyi, uygulamada da derleyiciyi kullanmak pratik bir yoldur. Bir başka yöntem de bazı alt-programları Assembler dilinde yazmaktır.

Zemin ekranın tümünü kaplayan bir resim olarak düşünülebilir. Bu durumda başvurulabilecek en kolay yol, bu resmi belleğin belirli bir bölgesine kaydedip daha sonra çizim sırasında da buradan çağırmaktır.

Bir cismi canlandırmanın gerektirdiği işlevler şunlardır:

1 / Zemini belleğin ekranla ilgili bölümüne aktarma (görüntüleme)
2 / Cismin başlangıç konumunda gösterimi
3 / Zeminin gösterimi (cismi silerek)
4 / Cismin bir sonraki konumunda gösterimi

İşte cismin hareketi bu dört işlev peşpeşe ve hızla yinelenerek ve süreklilik izlenimi veren bir dizi küçük hareket yaratılarak oluşur. Yine de bu yöntemin sonuçları en iyi animasyon örnekleri arasında değildir. Çünkü hemen bütün BASIC çeviricilerinin ortak özelliği olan düşük yürütme hızı, işlevleri gözle görülür biçimde yavaşlatmaktadır.

Aslında genelleştirilebilecek bir yazılım çözümü de yoktur çünkü iyi bir canlandırma için mutlaka donanımla ilgili bazı ögeler kullanılmalıdır.

Hareketli bir cismin gösterimi. Şimdi makine sorununun nasıl çözülebileceği konusuna daha yakından bakalım. Diyelim ki ekranda 24'e (taban) 21 (yükseklik) ekran noktası(*) boyutlarında bir dikdörtgeni, zemini oluşturan bir resmin üzerine ve X0,Y0 konumuna çizmek istiyoruz.

Bu cismin (dikdörtgenin) her hareketinden sonra zeminin bir bölümü cismin dışına çıkarken, bir bölümü de onun tarafından maskelenmektedir. Bu yüzden her hareketten sonra zeminin yeniden çizilmesi gerekmektedir.

Süreç, peşpeşe üç evreyi içerir:

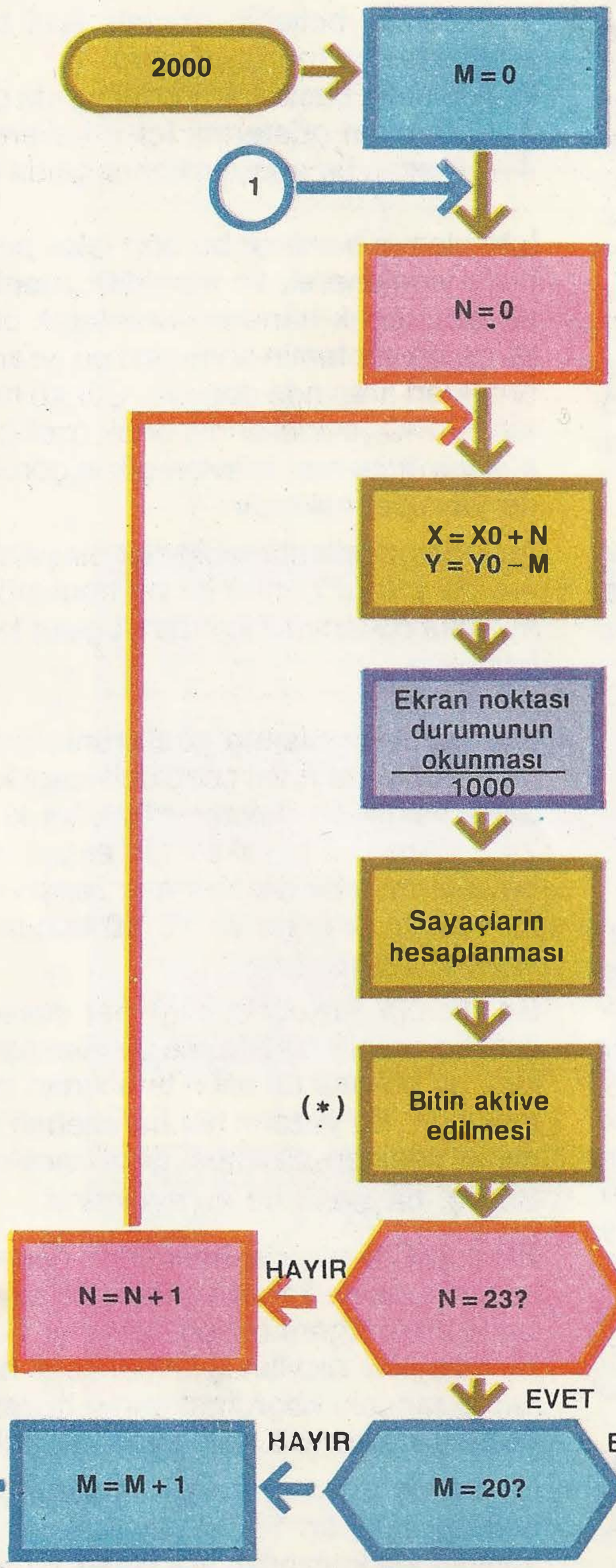
Evre 1. Cisim yerleştirilmeden önce ekranda zeminin cisme karşılık gelen yeri (aynı boyutlarda dikdörtgen) belleğe aktarılır (iki boyutlu bir dizeyle). Yapılan iş saklanacak her noktanın ekrandaki koordinatlarının hesaplanması ve bellekte karşılık gelen yere aktarılmasıdır.

Bu bellek alanının içeriği okundukça bir ara belleğe aktarılır. 1662. sayfada yer alan akış şemasında kullanılan ara bellek B1% (21,3)'-tür (bu boyutlandırma daha sonra açıklanacaktır).

(*)Bu değerler donanımı kullanarak canlandırma yapan bir sisteme elverişli olduğu için seçilmiştir.

Evre 2. Cisim X0,Y0 noktasından itibaren girilir. Buna karşılık gelen görüntü (ki bu da cismin özel bir çizimidir) S%(21,3) dizeyine aktarılır.

1. CİSMİ İÇEREN PENCERELERİN BELLEĞE ALINMASI



1. sayaç:
Noktanın X0,Y0 referansına göre dikey uzaklığı

2. sayaç:
Yatay uzaklık

Noktanın koordinatları

MD$="L" koşulu 2000 alt-programın başında, döngülerin dışında girilir.
Noktanın durumu BT ile belirtilir.

I=21-M (1. sayaç)
J= INT(N/8)+1 (2. sayaç)
K=J*8-N8-1 (n. bit)

B1((IN) elemanının K nolu biti BT'ye eşitlenir

N=23 (şeklin sonu)

Dikdörtgenin sonu

• Bu komut bloğu bir ekran noktasını aktive etmek için 1000 alt-programında kullanılanla aynıdır. Bu yöntem bayt içinde maskeleme yaparken mantıksal işlemci kabul etmeyen makineler için de geçerlidir. Böyle bir olasılık varsa (örneğin CP/M'de vardır) gerekli mantıksal işlemciyi eklemek yeter.

Evre 3. Cismin hareketi zeminin yeniden oluşturulması, yani B1%(21.3)'de saklanan bölümün yeniden eski yerine konması ve X0,Y0 değerleri yeniden düzenlenerek çevrime başlanmasıyla gerçekleştirilir.

2 ve 3 evreleri aynı akış şemasında gösterilebilirler. Karşılık gelen parametreli alt-program X0,Y0 koordinatlarından itibaren ya cismi S%(21,3) ya da zemini B1%(21,3) gösterir.

Ekran belleğinden veya ona doğru olan veri aktarımı, karşılık gelen bellek konumundan ekran üzerindeki koordinatların hesaplanmasını gerektirir. Burada daha önce verilen 100 alt-programını uyguladık. Bu alt-program girişte ekrandaki noktanın X ve Y koordinatlarını ve MD$'ye verilen değeri soruyor (MD$ = ''L'', yanık/sönük durumunun okunması, MD$ = ''S''dey azılması için). Çıkışta da nokta yanıksa BT = 1 değilse, BT = 0 veriyor.

Bir sonraki sayfada bu alt-program, B1% (21,3) dizeyi ile sergilenmektedir. Her bir elemanı 8 bitten oluşan bu dizey, ekrandaki nokta sayısı kadar durum içerebilmektedir. Dolayısıyla yatay 24 nokta yalnızca 3 bellek biriminde saklanabilmektedir. Aynı mantık canlandırma özelliğine sahip bazı makinelerde de kullanılmaktadır.

F göstergesinin aldığı değere (F = 1 zemin F = 2 cisim) göre zemini veya cismi oluşturan akış şeması 1665. sayfada yer alıyor. Kullanılan fonksiyonların tümü 2000 alt-programınınkilerle aynıdır, şu farkla ki burada F'nin değerine bağlı olarak iki farklı dizey kullanılmaktadır (F = 1 ise B1%, F = 2 ise S%).

Son olarak sayfa 1686'da yer alan akış şeması, içinde verileri hazırlamaya yarayan 5000 ve 6000 alt-programlarının bulunduğu bir programa aittir (bkz. s. 1667'den 1669'a kadar olan döküm).

Canlandırmayla ilgili sorunlar. 1666. sayfada yer alan programda bir dikdörtgene indirgenmiş olan cismi temsil eden 2 tabanlı değerler, S% dizeyinde saklanmakadır. Bu yöntemin iki eksikliği vardır:

1 / Çevre çizgilerini tanımaya olanak vermemektedir
2 / Cismin kendi içinde canlandırma yapmamaktadır.

Birinci eksiğin giderilmesi cisme ayrılan alanın tümünün doldurulmasını gerektirmektedir. Aslında cisme ayrılan alanın bütünüyle, cismin kapsamakta olduğu alan arasında bir fark olması sağlanabilseydi, hep sönük kalan noktalar bir çevre çizgisini oluştururlardı. Bu eksiklik daha ileri bir gösterim tekniği kullanan bir alt-program tarafından giderilebilir. Burada özetle cismi oluşturan dikdörtgenle ekranın kesiştiği noktalar yeniden oluşturulmaktadır. Bir başka deyişle cismin üzerindeki bir nokta aktifse (yanıksa) ekranda gösterilir, aktif değilse (sönük) ekran belleğine dokunulmaz. Bu, cismin şeffaf bir dikdörtgen üzerine çizilmesine benzetilebilir.

Bu işlem cisme ait görüntünün ekran belleğine aktarımı sırasında bir IF yardımıyla kolayca gerçekleştirilebilir. Eğer sözkonusu nokta aktif değilse aktarım olmaz ve zemin aynen kalır (bkz. şema s.1671).

İkinci eksikliğin ise (cismin içinde canlandırma yapılamaması) giderilmesi çok daha zordur. Bu sırf bu iş için tasarlanan makineler için de sözkonusudur. Burada ele aldığımız dikdörtgen hareket ettiği anda cismin kendisinin de biçim değiştirmesi sözkonusudur.

Bu hareket iki şekilde gerçekleştirilebilir:

1 / Bazı ara konumların bellekte saklanmasıyla.
2 / Hareketlerin matematiksel kurallarla şematize edilmesiyle.

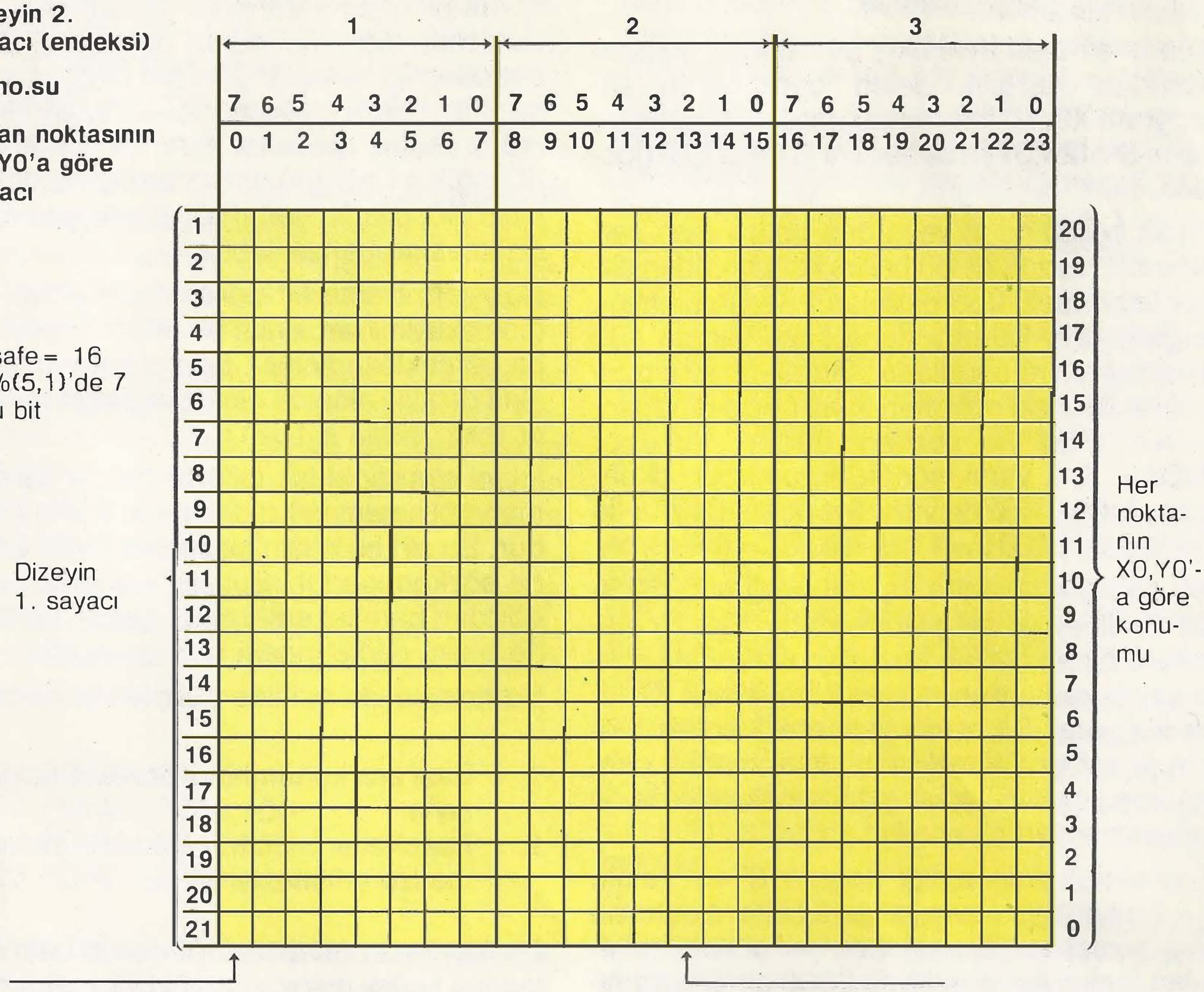
Bu ikinci çözüm hem yürütme hızı hem de kullanılan bellek alanının azlığı bakımından daha kullanışlıdır. Ancak bir şeklin hareketleri her zaman matematiksel formüllerle şematize edilmediğinden bu çözüm her zaman kullanılamaz.

Birinci çözüme gelince bir sayaç ve kullanılmak istenen bütün ara konumları kapsayan bir bellek alanı gerektirir. Örneğin T%(100,21,3) dizeyine 100 ara konum yerleştirilirse bunları sayaçla birbiri ardından S%(21,3)'e aktarmak ve böylece cismin kendi içinde bir hareket sağlamak mümkün olur.

Son olarak da yalnızca ilk ve son konumları ve eğer bu ikisi arasında çok fark varsa bir iki de ara konumu çizmek ve gerisini bilgisayara bırakmaktır. Çok karışık yönleri olan bu teknik halen incelenme aşamasında olmakla birlikte sinemada gittikçe daha fazla kullanılacaktır.

Sayfa 1666'da verilen akış şemasında cismin kendi içinde de canlandırma yapabilecek biçimde yeniden düzenlenmiş şekli yer alıyor. Bu canlandırma ilerde verilen bir düzine ka-

CİSMİN BELLEKTE ADRESLENMESİ



Tarama son satırdan başladığından (1.sayaç elemanı = 21) 21 satırının ekran noktaları 3 gruba ayrlabilir.

— İlki (2.sayaç = 1), B1%(21,1) elemanının 0-7 bitleri arasında yer alır.
— İkincisi B1%(21,2)'nin 0-7 bitleri arasındadır.
— Üçüncüsü B1%(21,3)'tedir.

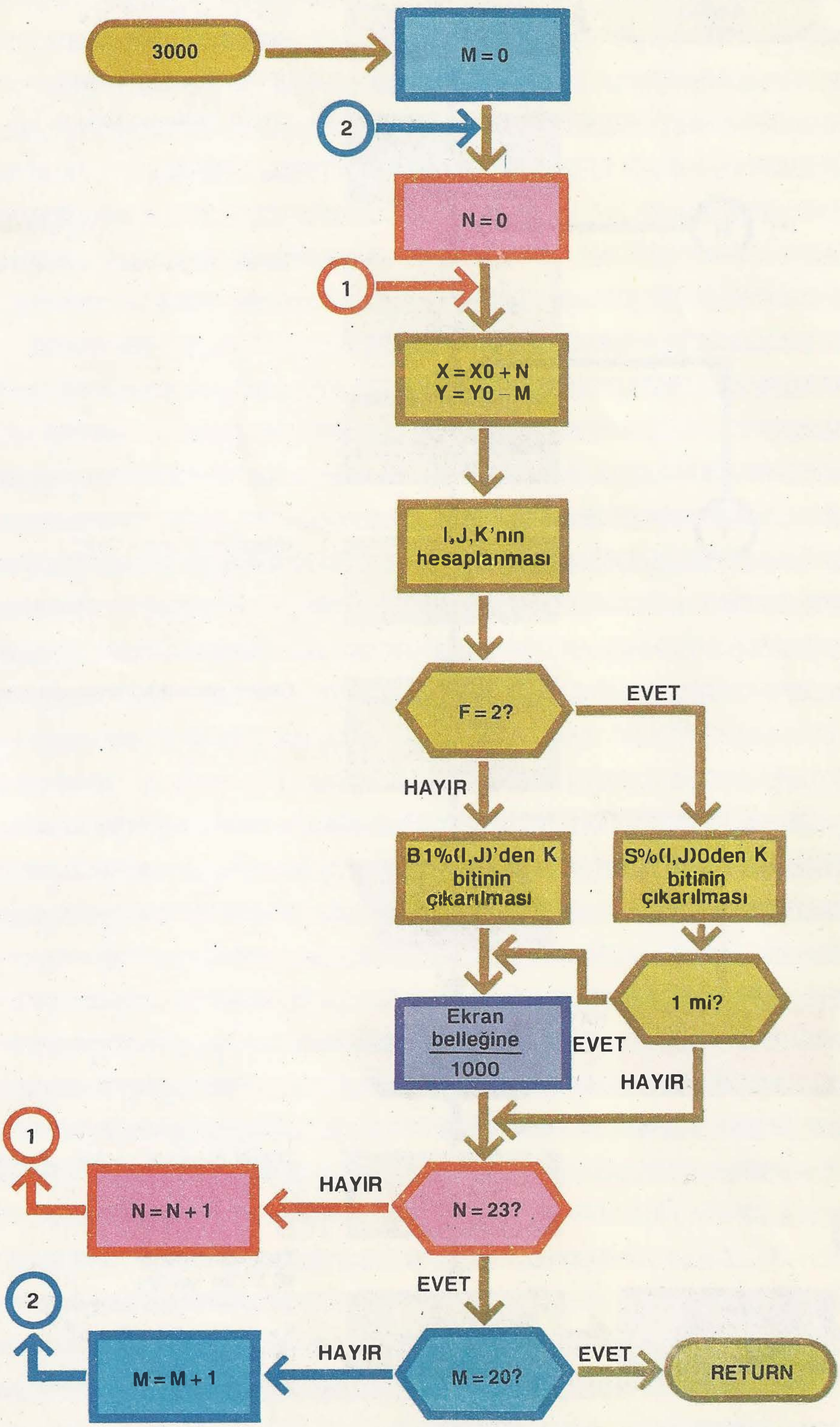
Aynı süreçle diğer bütün noktaların adresleri elde edilebilir (sayaçlar ve bitler). Örneğin N = 12 M = 13 noktası B1%(8,2) elemanının 3. bitinde bulunur. Kural şöyledir:

1. sayaç = 21-m
2. sayaç = INT(N/8) + 1
3. bit = ikinci sayaç*8-N-1

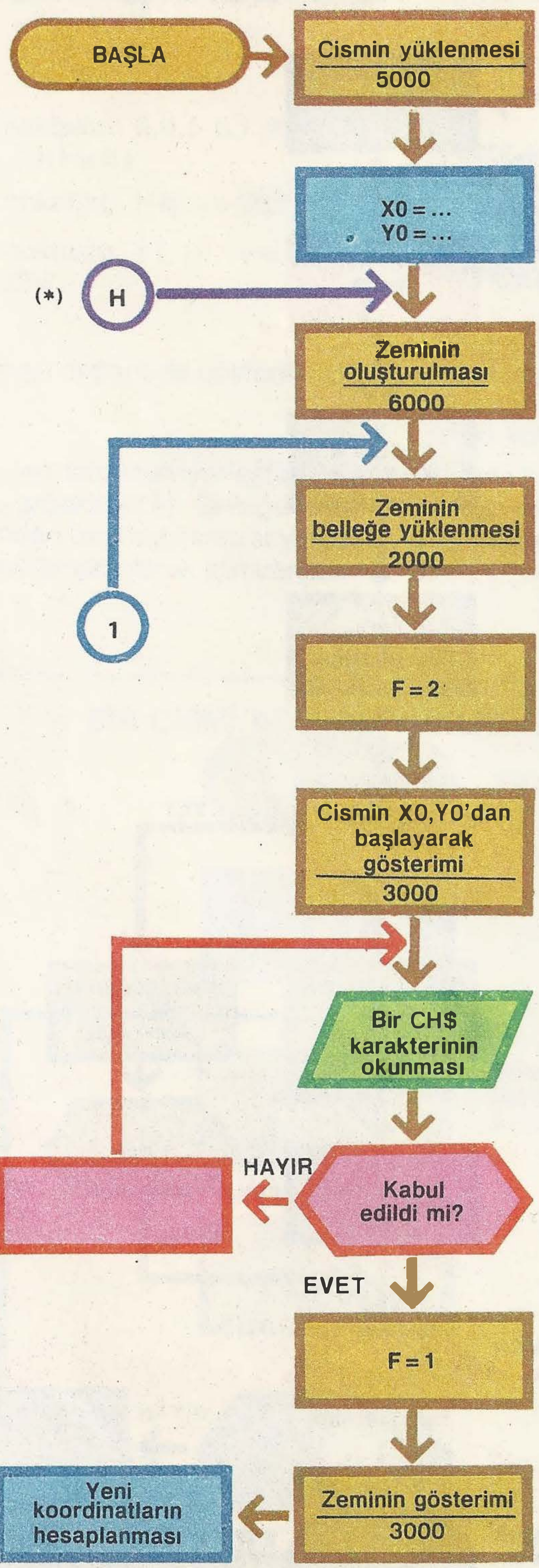
Burada referans noktasının da (X0,Y0 noktası kendisinden N = 0,M = 0 kadar uzakta olduğuna dikkat edelim. Öyleyse:

1. sayaç = 21-0 = 21
2. sayaç = 1
Bit no su = 7

ZEMİN YA DA CİSMİN GÖSTERİMİ



HAREKETLİ CİSMİN YÖNETİMİ ÖRNEĞİ



Cisim bir DATA komutuyla S%(21,3)'e yüklenir

Cismin başlangıç konumu

Bu alt-program zemini oluşturur

Gösterim (Cisim = ON)

Çizilen resmin konumlandırılması

Zemini harekete geçiren gösterge

Yeni koordinatlar
X0,Y0'da saklanır.
J = sol : X0 = X0-ST
K = sağ : X0 = X0 + ST
I = yukarı : Y0-Y0-ST
M = aşağı : Y0 = Y0 + ST

H ve L bağlantılarına daha ilerde gönderme yapılacaktır.

BELLEKLE YÖNLENDİRİLEN NESNELERİN HAREKETİNE ÖRNEK

```
 10  REM ------------
 20  REM     MAIN
 30  REM ------------
 40  DIM S%(21,3),B1%(21,3)
 50  ST = 5
 60  GOSUB 5000
     : REM
 70  XO = 10
     :YO = 180
 80  GOSUB 6000
     : REM ZEMININ GOSTERIMI
 90  GOSUB 2000
     : REM PENCERENIN OKUNMASI
100 F = 2
110 GOSUB 3000
    :REM HAREKETLI CISMIN YAZILMASI
120 GET CH$
130 IF CH$ < > "I" AND CH$ < > "J"
       AND CH$ < > "M" AND CH$ < > "K"
       AND CH$ < > CHR$ (27) THEN 120
140 F = 1
150 GOSUB 3000
    :REM PENCERENIN YAZIMI
160 IF CH$ = "I" THEN YO = YO - ST
170 IF CH$ = "M" THEN YO = YO + ST
180 IF CH$ = "J" THEN XO = XO - ST
190 IF CH$ = "K" THEN XO = XO + ST
200 IF CH$ = CHR$ (27) THEN END
210 GOTO 90
1000 REM ------------------------------------
1010 REM * ADRESLERIN HESAPLANMASI *
1020 REM ------------------------------------
1030 GY = INT (Y / 8)
1040 IF Y < 64 THEN MR = 8192 + GY
       * 128
     :GOTO 1070
1050 IF Y < 128 THEN MR = 8232 + (GY
       - 8) * 128
     :GOTO 1070
1060 MR = 8272 + (GY - 16) * 128
1070 A = Y - GY * 8
1080 MY = MR + A * 1024
1090 GX = INT (X / 7)
1100 MX = MY + GX
1110 BIT = X - 7 * GX
1120 NN = PEEK (MX)
1130 BY = NN
1140 FOR KK = 0 TO BIT
1150 MS = NN / 2
1160 NN = INT (MS)
1170 NEXT
1180 IF MS < > NN THEN BT = 1
     :GOTO 1200
     :REM EKRAN NOKTASI YANIK
1190 BT = 0
     :REM EKRAN NOKTASI SONUK
1200 RETURN
2000 REM ------------------------------
2010 REM * PENCERENIN OKUNMASI
2020 REM ------------------------------
2030 MD$ = "L"
2040 FOR M = 0 TO 20
2050 FOR N = 0 TO 23
2060 X = XO + N
     :Y = YO - M
```

```
2070  GOSUB 1000
2080  I = 21 - M
    : REM BIRINCI SAYAC
2090  J = INT (N / 8) + 1
    : REM IKINCI SAYAC
2100  K = J * 8 - N - 1
    : REM EKRAN NOKTALARININ OKUNMASI
2110  B1%(I,J) = B1%(I,J) + BT * 2  K
2120  NEXT N
2130  NEXT M
2140  RETURN
3000  REM ------------------------------
3010  REM * PENCERENIN YAZIMI *
3020  REM ------------------------------
3030  MD$ = "S"
3040  FOR M = 0 TO 20
3050  FOR N = 0 TO 23
3060  X = XO + N
    : Y = YO - M
3070  I = 21 - M
    : REM
3080  J = INT (N / 8) + 1
    : REM CF.2080
3090  K = J * 8 - N - 1
    : REM CF.2100
3100  IF F = 2 THEN NO = S%(I,J)
    : GOTO 3120
3110  NO = B1%(I,J)
3120  FOR N1 = 0 TO K
3130  N2 = NO / 2
3140  NO = INT (N2)
3150  NEXT N1
3160  IF N2 = NO THEN BB = 0
    : GOTO 3180
3170  BB = 1
3180  GOSUB 1000
3190  IF BT = BB THEN 3220
3200  IF BT > BB THEN POKE MX,BY - 2
        BIT
3210  POKE MX,BY + 2  BIT
3220  NEXT N
3230  NEXT M
3240  RETURN
5000  REM ------------------------------------
5010  REM * HAREKETLI CISMIN YUKLENMESI
5020  REM ------------------------------------
5030  FOR I = 1 TO 21
5040  FOR J = 1 To 3
5050  READ S%(I,J)
5060  NEXT J,I
5070  DATA 0,0,0
5080  DATA 0,0,0
5090  DATA 0,0,0
5100  DATA 0,0,0
5110  DATA 0,0,0
5120  DATA 0,0,0
5130  DATA 0,0,0
5140  DATA 28,0,248
5150  DATA 62,1,252
5160  DATA 103,3,254
5170  DATA 199,7,158
5180  DATA 143,15,28
5190  DATA 30,15,28
5200  DATA 60,30,60
5210  DATA 60,30,108
5220  DATA 31,252,248
5230  DATA 15,248,144
```

```
5240 DATA 7,240,48
5250 DATA 0,0,0
5260 DATA 0,0,0
5270 DATA 0,0,0
5280 RETURN
6000 REM ------------------------------
6010 REM * ZEMININ YUKLENMESI
6020 REM ------------------------------
6030 HOME
6040 INPUT "ZEMININ ADI ";NS$
6050 HGR
     :POKE - 16302,0
     :HCOLOR = 3
6060 PRINT CHR$(4);"    ";NS$",A$20
     00"
6070 RETURN
```

dar şeklin hazırlanmasıyla elde edilmektedir. Cismin hareketi tuşlarla yönlendirildiği için canlandırma giriş aşamasında oldukça yavaştır. Daha gerçekçi bir sonuç elde edebilmek için otomatik olarak yürütülen bir parkur tanımlanır. Bu tanımlamayı yapmak için kullanılan en basit yöntem hareket tuşlarına karşılık gelen kodların DATA komutu aracılığıyla girildiği 10 değerli bir dizey oluşturmaktır. Bu durumda okuma komutu CH$ = SP(P) komutuyla harekete geçirilir. Burada SP hareket kodlarını içeren bir dizeydir.

P'nin 1'den 10'a kadar değiştiği bir döngü hareket tuşuna 10 kere peşpeşe basılmış gibi çalışacak, öte yandan DATA komutuyla tanımlanan parkur otomatik olarak gerçekleştirilecektir.

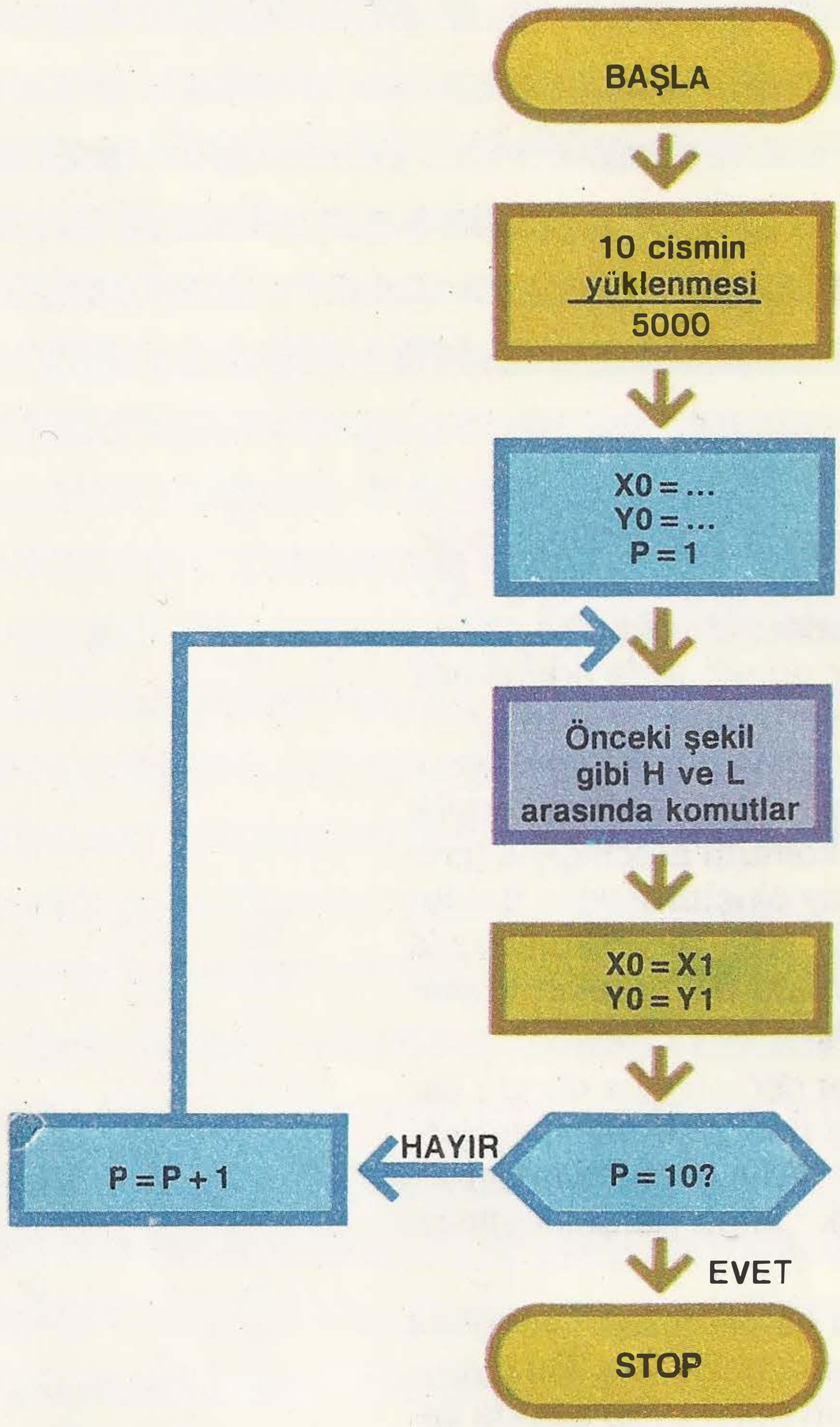
Bu tür uygulamalarda çarpışmaların yönetimi kaçınılmaz bir uygulama halini alır. Bu uygulamada hareketli cismin hareketi sırasında zeminin herhangi bir özel alanıyla karşılaşıp karşılaşmadığı araştırılır. Bu kontrol cisim tarafından doldurulan bellek alanının okunmasıyla yapılır; eğer alan aktifse çarpışma var demektir.

Hareketli Cisimlerin Donanımla Yönetimi

Çoğu kişisel bilgisayar, tabii özellikle de profesyonel uygulamalardan çok video oyunlarına yatkın olanlar, hareketli cisimlerin yönetimine yönelik donanımlara sahiptirler.

Donanıma ait bu ögeler aynı anda birçok cismi belleğe alma, canlandırma ve kontrol etme özelliklerine sahiptirler. Bu durumda kullanıcıya da bu cisimleri ve zemini yaratıp sonra da silme ve yeniden çizme ile uğraşmadan cisimlere hareket komutları vermek kalmaktadır. Bu tür makinelerde her hareketli cisim yatay

KENDİ İÇİNDE DE HAREKET EDEN CİSİMLERİN YÖNETİMİ



Hazırlanan cisimler (10 adet) aynı şeklin çeşitli biçimleridir. Birbiri ardından gösterildiklerinde dikdörtgenin içinin de hareketini sağlarlar. Bu 10 biçim, bir DATA komutuyla T(10,21,3)'de saklanır.

P cismin 10 şeklinin birbirini izlemesini düzenleyen bir sayaçtır.

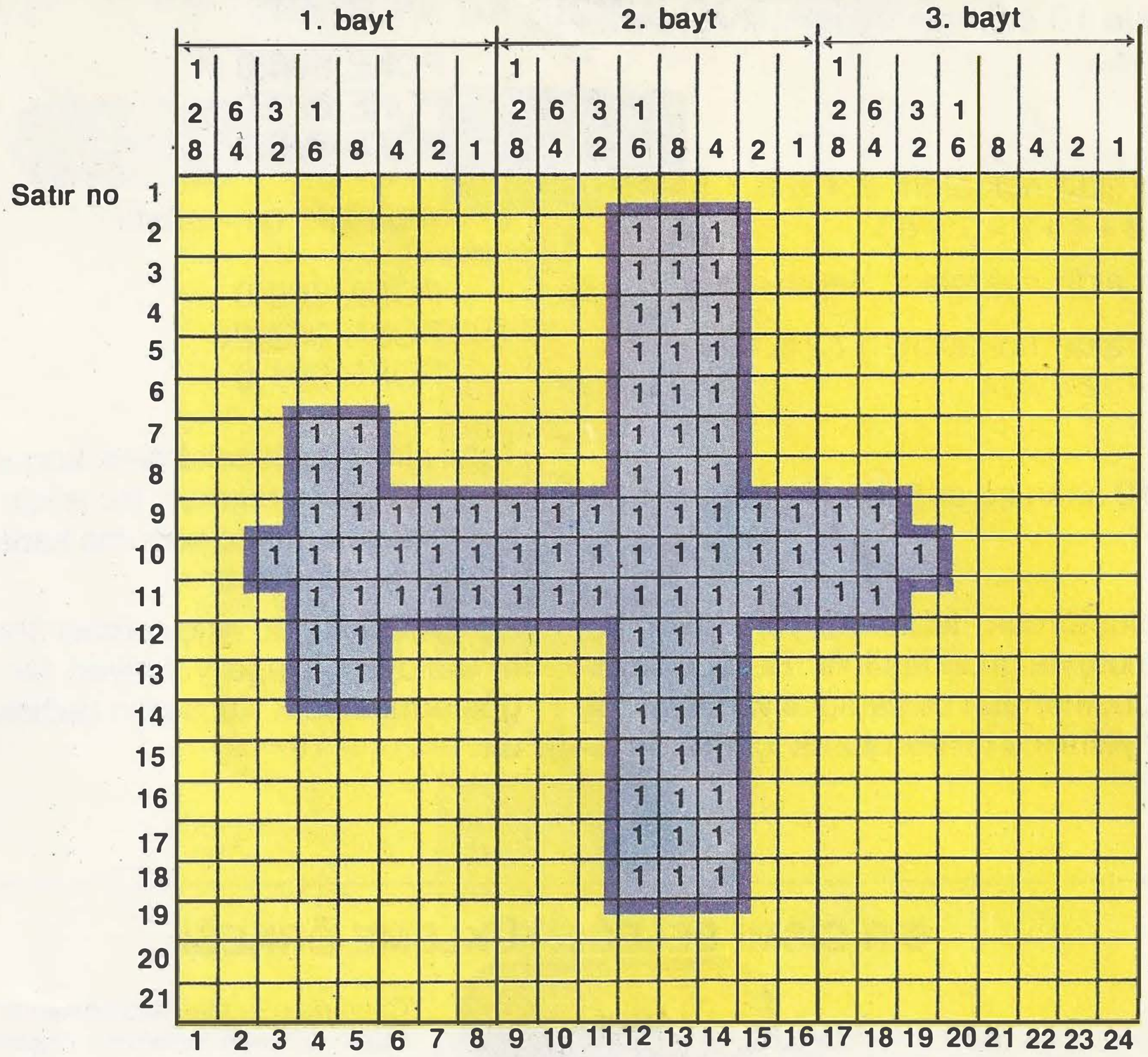
Tek fark burada 3000 alt-programının zemini, diğerinin (3700) ise cismi yönetmesidir.

ve dikey olarak ele alınan bir noktalar kümesi olarak temsil edilir. Bunlardan bazıları yakılıp bazıları söndürülerek istenilen şekil elde edilebilir.

Hareketli cismin bellekte saklanması. Her cisim için yüzlerce ekran noktası sözkonusudur. Örneğin Commodore 64'te kullanılan değer 21 satır ve 24 kolondan oluşan 504 noktadır. Dolayısıyla bellekte kapladığı yer 504 bit yani 63 bayttır. Böylece bir cismin oluşturulması için 63 bellek biriminde 504 bitin her birinin yanık/sönük durumları yer alır. Bir noktayı yakmak için karşılık gelen bayta bitlerden birini 1 yapacak belli bir ondalık sayı yazılır. Örneğin birinci satırla birinci kolonun kesiştiği yerde bulunan noktayı yakmak için 128, dördüncü kolon için ise 16 yazmak gerekir (bitin durumuna karşılık gelen ondalık sayıların 1,2,4,8,16,32,64,128 olduğunu unutmayalım). 7 rakamı yazılarak bu değerler toplanabildiğinden 1,2 ve 3 konumlarında bulunan bitler aynı anda aktif hale getirilir.

Karşı sayfada yer alan şemada birinci satır hiç aktif nokta içermiyor; öyleyse buna karşılık gelen noktalar 0,0,0'dır (sırasıyla 1,2 ve 3 baytları). 2. satırda ise aktif olan 12,13 ve 14 nok-

BİR CİSMİN BELLEKTE YERLEŞTİRİLMESİNİN ŞEMASI



3 bayt * 8 Bit = 24 kolon

Satır no	Değerler		
	1. bayt	2. bayt	3. bayt
1	0	0	0
2	0	28	0
3	0	28	0
4	0	28	0
5	0	28	0
6	0	28	0
7	24	28	0
8	24	28	0
9	31	255	192
10	63	255	224
11	31	255	192
12	24	28	0
13	24	28	0
14	0	28	0
15	0	28	0
16	0	28	0
17	0	28	0
18	0	28	0
19	0	0	0
20	0	0	0
21	0	0	0

taları 2 baytına ait ve 16 + 8 + 4 = 28 değerindedirler yani 0,28,0 grubuyla gösteriliyorlar.

Aynı şekilde 10 satırının değerlerinin hesabı için:

1 baytının aktif noktaları: 3,4,5,6,7,8 = 32 + 16 + 8 + 4 + 2 + 1 = 63

2 baytının aktif noktaları: Hepsi = 255

3 baytının aktif noktaları: 17,18,19 = 128 + 64 + 32 = 224

Öyleyse 10 satırı şu değerlerle gösterilir: 63, 255, 224.

Böylece oluşturulan tablonun yerleştirilmesi POKE komutuyla grçekleştirilir. Belleğe almanın 896 konumundan başladığını varsayarsak (bu değer yalnızca örnek olarak için verilmistir) ilk satırı belleğe almak için 3 komut gerekecektir (her bayt için bir komut):

```
POKE 896,0      1. bayt
POKE 897,0      2. bayt
POKE 898,0      3. bayt
```

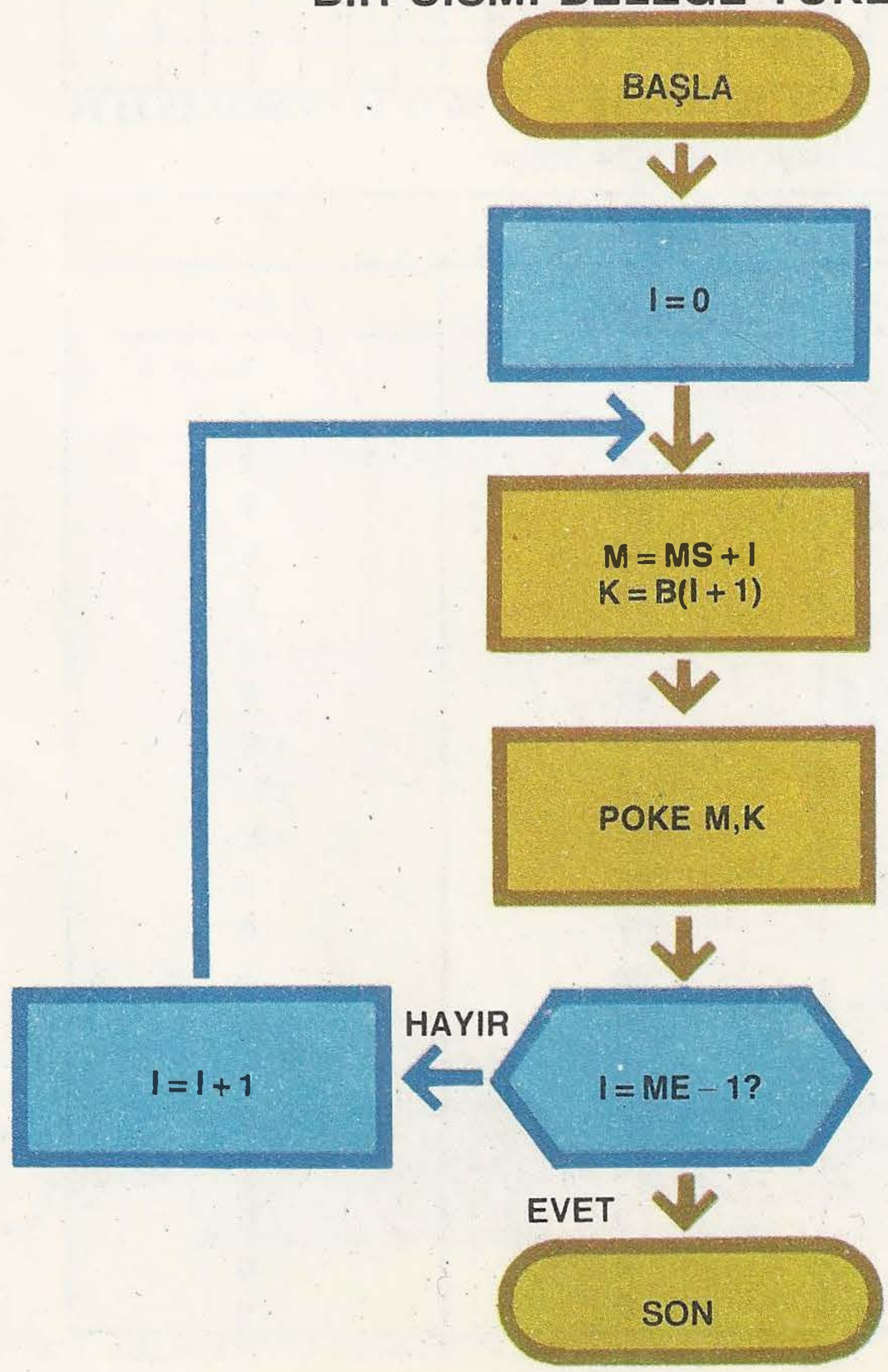
2. satır şöyle girilecektir:

```
POKE 899,0
POKE 900,28
POKE 901,0
```

Tabii aktive edilecek bellek konumu kadar komut yazmak gerekmez. Bu iş için en iyi çözüm birinciden sonuncu konuma kadar değişen bir döngü kullanmaktır.

Aşağıda bulunan akış şeması önceki sayfadaki tabloyu belleğe yükleyen bir alt-programı göstermektedir. Kullanılan değişkenler şunlardır:

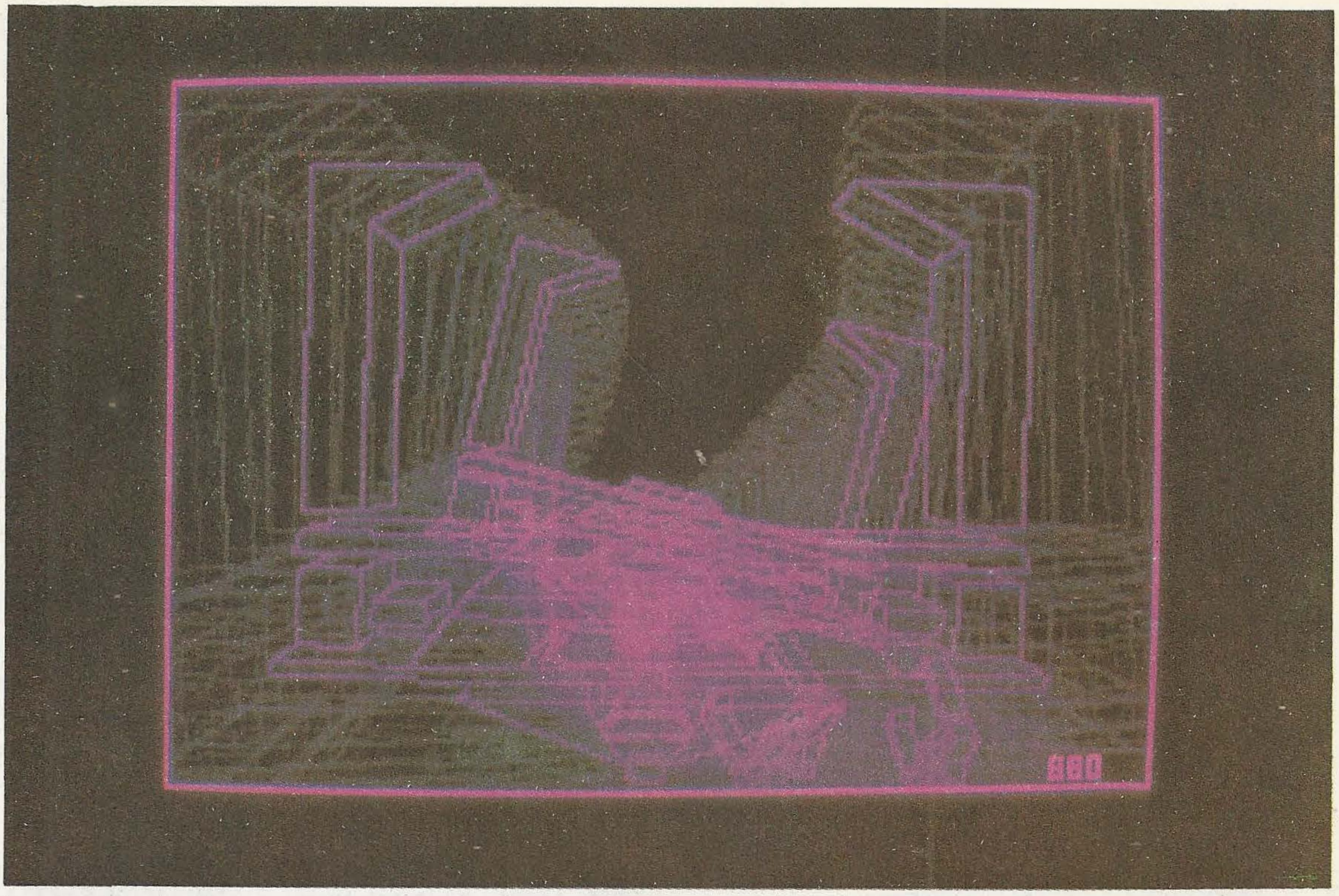
BİR CİSMİ BELEĞE YÜKLEME ÖRNEĞİ



Çağırırken şunlar sağlanmalıdır:
MS = Cismın bellekteki başlangıç adresi
ME = İlgili bellek konumu sayısı
B(*)= Değerler dizeyi

Adres (M) başlangıç adresi (MS) artı I sayacına eşittir. İçeriği de B dizeyinde karşılık gelen değerdir.

Döngü I = ME-1'de tamamlanır.



B(63) = Her satırın 3 baytını içerir (3 bayt × 21 satır = 63 bayt)
MS = İlk bellek konumu
ME = Aktive edilecek bellek konumu sayısı. Tablonun tümü için ME = 63 girmek gerekir (eğer tanımı tamamlayan son bayt da hesaba katılırsa 64).

Tanımlar tablosu yazılabilir beleğin (RAM) herhangi bir yerinde yer alır, mutlaka diğer cisimlerin tanımlarına yakın bir yerde olması gerekmez. 8 farklı cisim içerebilen Commodore 64'te tanımlanan cisimlerin konum göstergeleri 2040 ilâ 2047 adreslerini kaplar.
Uygulamada cisimlerin tanımlandığı tablolara yönetilecek cisim sayısı kadar 64 baytlık bloklara bölünmüş tek bir alan ayırmak kullanışlı olur.

Cisimlere ayrılan bellek alanının 832 konumundan başladığını varsayarsak demin tanımlanan cisim (896 konumuna yüklenen) 2 bloğunda yer alır. Aktive edildikte sonra adreslemeye ayrılan konuma (2040) 14 değerini (12 + 2) yazmak gerekir. Sistem böylece 14 × 64 çarpımını yaparak yerini bulur.

Hareketli cismin görüntülenmesi. Tablo belleğe yüklendikten sonra cismi görüntüleyen mantık harekete geçirilir. Cisimlerin yönetimi (çizimi, hareketleri vb.) genelde belleğin ön-

ceden programlanabilir bazı özel bölgelerini harekete geçirerek veya durdurarak yapılır.

Sisteme bir çok cisim yüklenebildiğinden hangisinin üzerinde çalışılmak istendiği belirtilmelidir. Bu da genelde bellekteki özel bir konuma sahip bir bitin 1 durumuna getirilmesiyle olur. Örneğin bu adrese 10 tabanlı 3 (yani 2 tabanlı 11) yazılırsa, 0 ve 1 numaralı cisimler aktive edilir.

Commodore 64'te cisimleri görüntüleyen kontrol kaydı 53269 adresinde bulunur. Eğer bu kaydın 0 numaralı biti 1 yapılırsa, 0 numaralı cisim görüntüleniyor demektir. Bu yaklaşım diğer bütün cisimler için de aynıdır.

Yine de cismin bulunduğu ekran noktası belirlenmedikçe görüntüleme gerçekleşmez. Bunun için de her cisim içinde istenilen ekran noktasının koordinatları bulunan iki kayıtla kontrol edilir. Örneğin sıfır cismi 53248 ve 53249 kayıtlarıyla kontrol edilir.

Bunlardan ilkine X koordinatı, ikincisine ise Y koordinatı yüklenir (POKE). 0 cisminin 100, 150 noktasında görüntülenmesi (bellekte 2 bloğunda saklıdır) şu komutlarla elde edilir:

```
30 POKE 2040,2
40 POKE 53248,100
50 POKE 53249,150
60 POKE 53269,1
```

Cismin hareketi. Görüntülenen cismi hareket ettirmek için bir döngü yardımıyla iki kontrol kaydında bulunan konum koordinatlarını değiştirmek gerekir. Bu durumda cisminin yatay hareketi şöyle elde edilir:

```
100 POKE 53249,100
110 FOR I=50 TO 150
120 POKE 53248,I
130 NEXT I
```

100 satırı Y değerini sabitlerken, döngüyle X koordinatına 50'den 150'ye kadar değerler verilmektedir. Genelde cismin hareketi sırasında bir de çarpışma kontrolü yapmak gerekir. Burada da içeriği çarpışma olup olmadığını belirten özel bellek konumları kullanılır.

Commodore 64'te iki cismin çarpışması 53278 kaydı ile cisim ile, zeminin bir parçasının çarpışması ise 53279 kaydı ile kontrol edilir (bkz. karşı sayfadaki akış şeması).

Hareketli cisimlerin yönetiminde bellek alanının kullanımı. Commodore 64'ün hareketli cisimlerin yönetiminden sorumlu bellek alanı sayfa 1676'da şematize edilmiştir; 53248 konumundan başlar.

İlk 16 birimde makinenin yönetebileceği 8 cismin konum koordinatlarının kayıtları bulunur. Ardından X koordinatlarının en belirleyici olan bitlerini (her cisme 1 bit) içeren kayıt (53264) gelir. Sonraki 4 bayt hareketli cisimlerle ilgili değildir. Harekete geçirme/durdurma kaydı ise 53269 adresinde yer alır (göreceli 21 adresi). Bu kaydın bitlerinden birine 1 vererek karşılık gelen cisim harekete geçirilir.

53271 ve 53277 adreslerinde cismin boyutlarını iki katına çıkarmaya olanak veren iki kayıt bulunur. Bunlardan birincisi X, ikincisi ise Y ekseni içindedir. Hangisi büyütülmek isteniyorsa o bite 1 verilir (0 cismi için 0,1 cismi için 1 vb.).

53275 kaydı bir cismin zemine göre önceliğini sağlar. Örneğin bu kaydın 3 numaralı biti 0 değerine sahipse, 3 cisminin zemine göre önceliği var (yani zemini kaplayacak) demektir. Eğer aynı bite 1 verilirse tam tersi olur.

53278 ve 53279 konumları cisim-cisim ve cisim-zemin çarpışmalarının göstergelerini içerir. Eğer 2 cismi 3 cismiyle çarpışıyorsa, 53278 kaydının 2. ve 3. bitleri 1 değerini alır. Zeminle çarpışma halinde de aynı mantık geçerlidir.

Son olarak da 53287'den 53294'e kadar olan kayıtlar cisimlerin renklerini seçmeye olanak verir. Bunun için istenilen rengin sayısal kodu verilir (0 = beyaz 1 = siyah vb.).

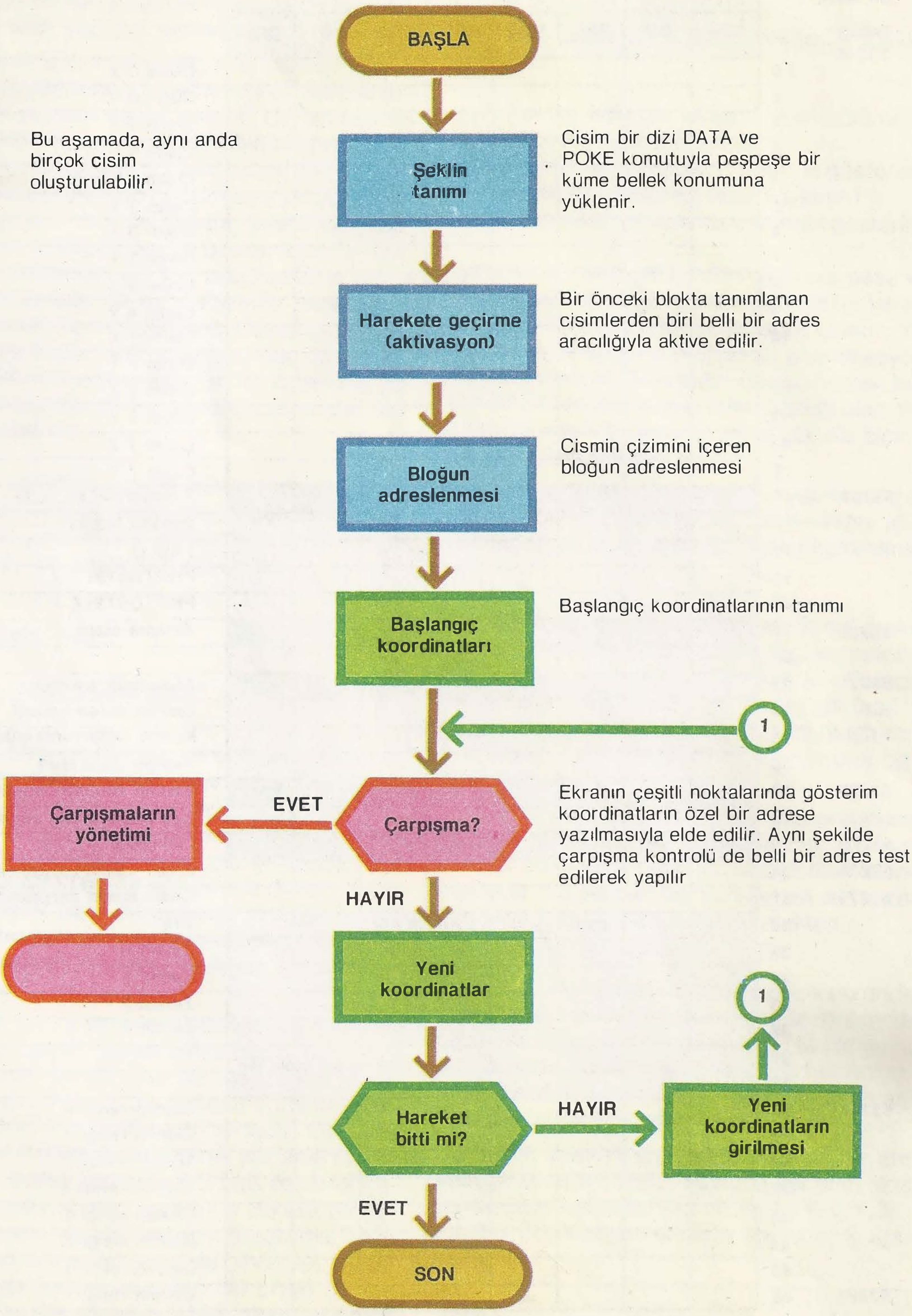
Diğer kayıtlar özel bazı fonksiyonları yürütmek için kullanılır. Bunların sayıları oldukça kabarık olduğundan burada üzerinde durmayacağız. Bir cismin tanımının peşpeşe 63 + 1 bayttan (sonuncusu hep 0 değerini alır) oluştuğunu ve bellekte belirli bir başlangıç adresinden itibaren saklandığını söylemiştik. Daha sonra bu konumun adresi sözkonusu cismi gösteren bir gösterge kaydında saklanır.

Aslında belleğe yüklenen, bir bellek adresi değil 64 bayttan oluşan bir bellek bloğunun sıra sayısıdır. Bu değeri 64 ile çarparak sistem cisim tanımının başladığı gerçek adresi bulur.

Bu mantık sayfa 1677'de daha ayrıntılı olarak açıklanmıştır.

Uygulama örneği. Sayfa 1678 ve 1680'de 4 çok renkli hareketli cisimler kullanan bir oyunun akış şeması bulunmaktdır. Oyun bir paraşüt atlayıcısını yatay hareketlerle ansızın be-

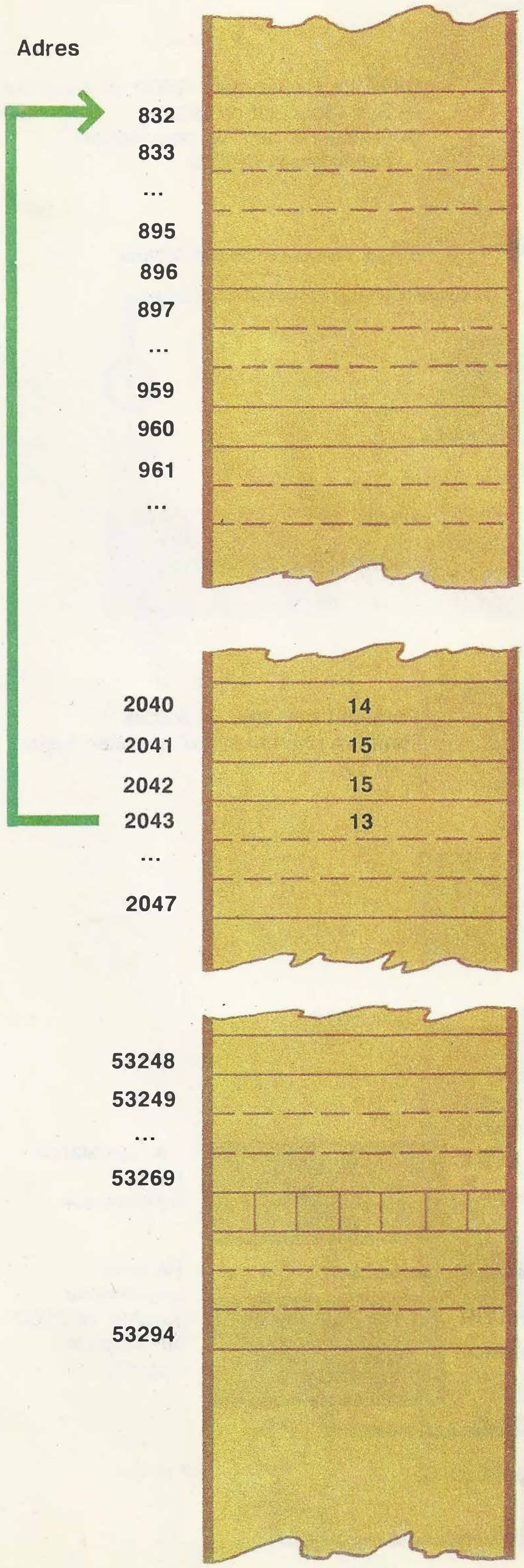
HAREKETLİ CİSİMLERİN YÖNETİMİNİN MANTIKSAL ŞEMASI



COMMODORE 64'DE CİSİM KAYITLARI İÇİN KULLANILAN SİCİLİN DÜZENİ

Bit nosu		7	6	5	4	3	2	1	0	
Taban = 53248		DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
	0									Cisim 0 X
	1									Cisim 0 Y
	2									Cisim 1 X
	3									Cisim 1 Y
	4									Cisim 2 X
	5									Cisim 2 Y
	6									Cisim 3 X
	7									Cisim 3 Y
	8									Cisim 4 X
	9									Cisim 4 Y
	10									Cisim 5 X
	11									Cisim 5 Y
	12									Cisim 6 X
	13									Cisim 6 Y
	14									Cisim 7 X
	15									Cisim 7 Y
53264	16									X değerinin en fazla
	17									ağırlıklı bitleri
	18									TABLO
	19									PHOTOSTYLE X
	20									PHOTOSTYLE Y
53269	21									Sınama cismi
	22									
53271	23									Cisimdeki Y artışı
	24									Cismin ekran belleği
	25									Kesme (interrupt) istekleri
	26									Kesme maskeleri
53275	27									Cismin zemine önceliği
	28									Çok renkli cisim seçimi
53277	29									Cisimdeki X artışı
53278	30									Cisim -cisim çarpışması
53279	31									Cisim-zemin çarpışması
	32	RENKLE İLGİLİ BİLGİLER								Ara
	33									Zemin 0
	34									Zemin 1
	35									Zemin 2
	36									Zemin 3
	37	Çok renkli cisimler								
	38									
53287	39									Cismin rengi 0
	40									Cismin rengi 1
	41									Cismin rengi 2
	42									Cismin rengi 3
	43									Cismin rengi 4
	44									Cismin rengi 5
	45									Cismin rengi 6
53294	46									Cismin rengi 7

HAREKETLİ CİSİMLERİN YÖNETİMİNDE BELLEK KULLANIMI



3 nolu cismin DATA başlangıcı (blok 13)

0 nolu cismin DATA başlangıcı (blok 14)

1 ve 2 nolu cisimlerin DATA başlangıcı (blok 15)

Cisim tanımlarının adresleme tablosu

2040 adresi 14 değerini içeriyor;
öyleyse 0 nolu cisim 14 bloğunda tanımlanmış

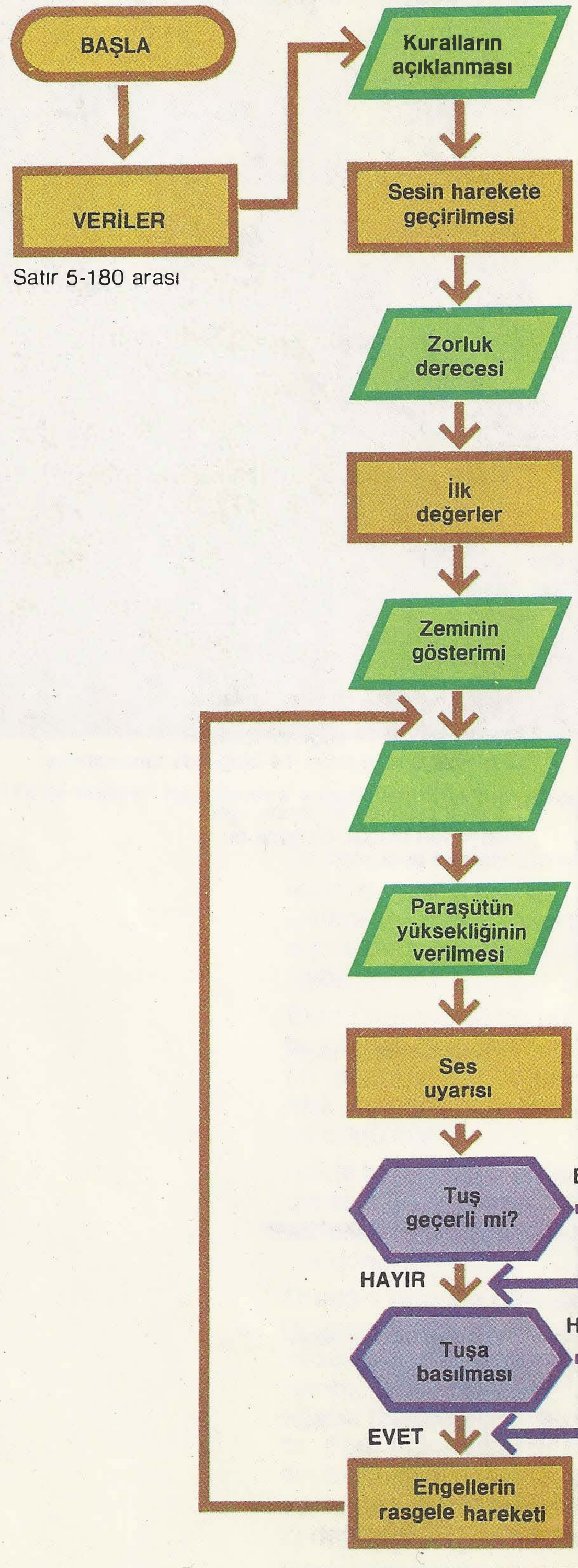
1 ve 2 cisimlerinin tanımı aynı
(aynı veri bloğu). Öyleyse bu
cisimler aynı.

Yönetim tablosu başı

Cisim hareket göstergesi

Yönetim tablosu sonu

PARAŞÜTLER PROGRAMININ AKIŞ ŞEMASI



Satır 5-180 arası

DATA'ların okunması (1000 alt-programı)
— Ses (3000 alt-programı)
— Cisimlerin renk ve boyutlarının belirlenmesi (2000)

Engel sayısının değiştirilmesi

5002 ve 5003 satırları

Paraşütü harekete geçirecek komutun okunması (kabul edilen tuşlar: X ve Z)

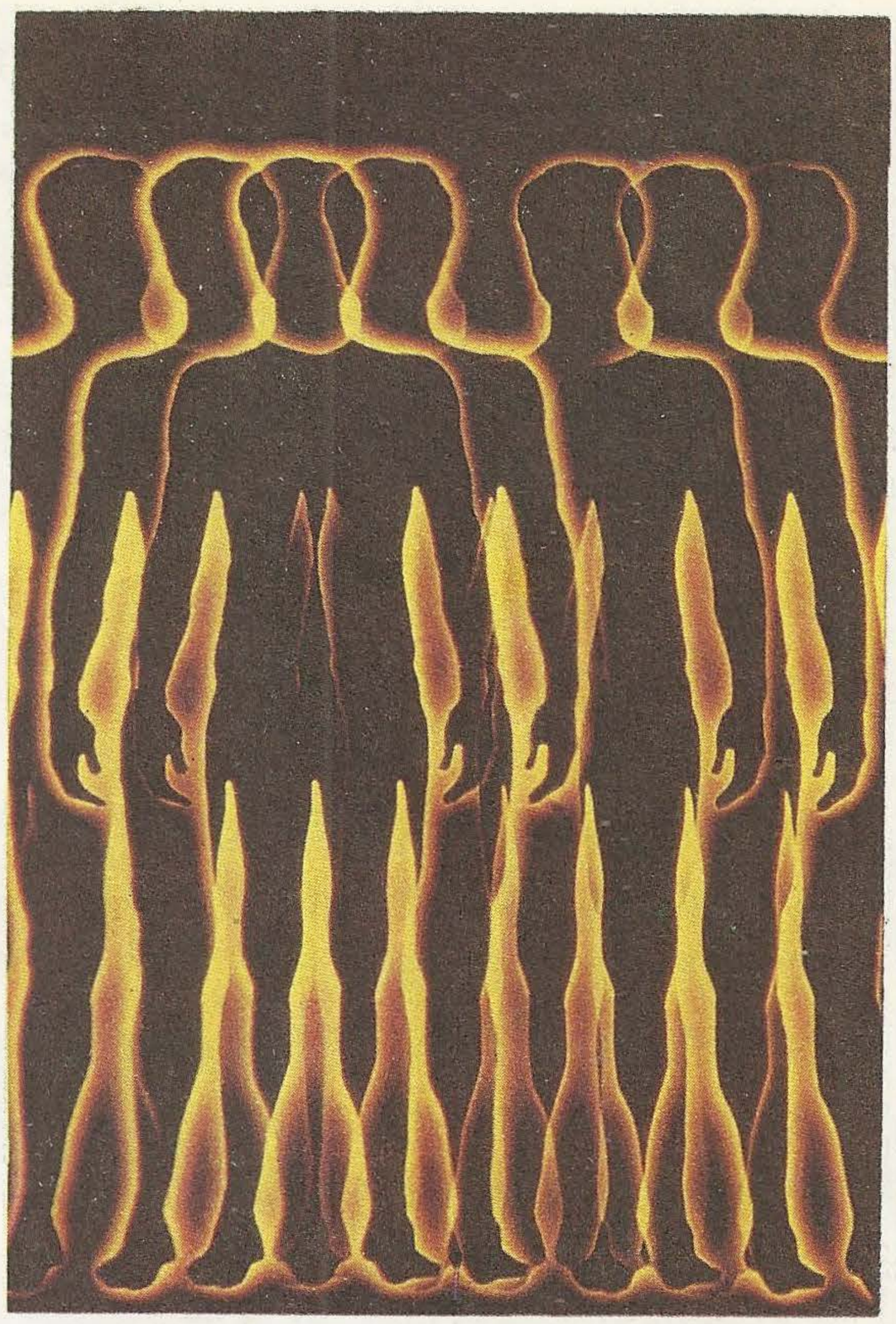
Satır 5015

Çarpışmalarda kontrol edilmektedir.

Rasgele koordinatlar yaratılır ve 21000 alt-programı çağrılır.

liren cisimlerle çarpışmaktan kurtarmaktan oluşmaktadır. Eğer paraşütçü yere inmeyi başarırsa oyun kazanılmaktadır.

İnişini biraz zorlaştırmak ve oyuncunun kontrolünü bozmak için de yandan gelen bir rüzgar oyuncuyu soldan sağa doğru itecektir.



Hareketli Cisimlerin Gelişmiş Yönetimi.

Hareketli cisimler video oyunlarından başka örneğin daha karmaşık çizimlerin oluşturulması için birim simgeler hazırlanması gibi alanlarda da kullanılırlar. Bu tür özel kullanımlar her zaman bellek adreslemesi ya da tablo oluşturulması gibi karmaşık sorunlarla uğraşmadan doğrudan BASIC dili ile çağrılabilecek fonksiyonlar gerektirir.

Özellikle kişisel bilgisayar sınıfına dahil modern makineler, çizim komutlarının yanında hareketli cisim yönetimini de içeren ileri BASIC versiyonları kullanırlar. Kullanılan başlıca fonksiyonlar şunlarır:

— Hareketli cisimlerin değişken gibi tanımlanması
— Gösterim
— Çarpışma kontrolü

Hareketli cisim tanımı (Sprite definition). Yöntem buraya kadar anlatılanla aynıdır ve cismin iki tabanlı görüntüsünün aktarıldığı bir bellek alanı ayırmaktan oluşur. Ancak deminkilerin aksine her cisim bir zincir değişken olarak tanımlanır ve dolayısıyla adres tablolarının yazılmasına gerek kalmadan tek bir DATA komutuyla yüklenir.

Komutun en çok kullanılan biçimi şudur:

```
SPRITE$(N) = B$
```

Burada N cismi BS zincir değişkeninin içeriğine eşit olarak tanımlanmaktadır. BS'nin kendisi de bir DATA komutuyla yüklenmiştir.

Bu komut da bundan sonrakiler gibi Philips VG8000 sistemine özgü olmakla birlikte, Standart MSX BASIC kullanan diğer bilgisayarlarda da kullanılabilmektedir. BASIC'in bu biçimi CP/M'in BASIC 80'ine oldukça yakın olmakla birlikte özel komutlar açısından daha zengindir. BASIC MSX daha yeni olmakla birlikte kullanım kolaylıklarından dolayı (özellikle çizmle ilgili olanlar) Z80 minik işlemcisine dayalı makinelerin çoğu için uyarlanmıştır.

Görüntüleme. Bu komutun en basit biçimi şöyledir:

```
PUT SPRITE P, (X,Y), C, N
```

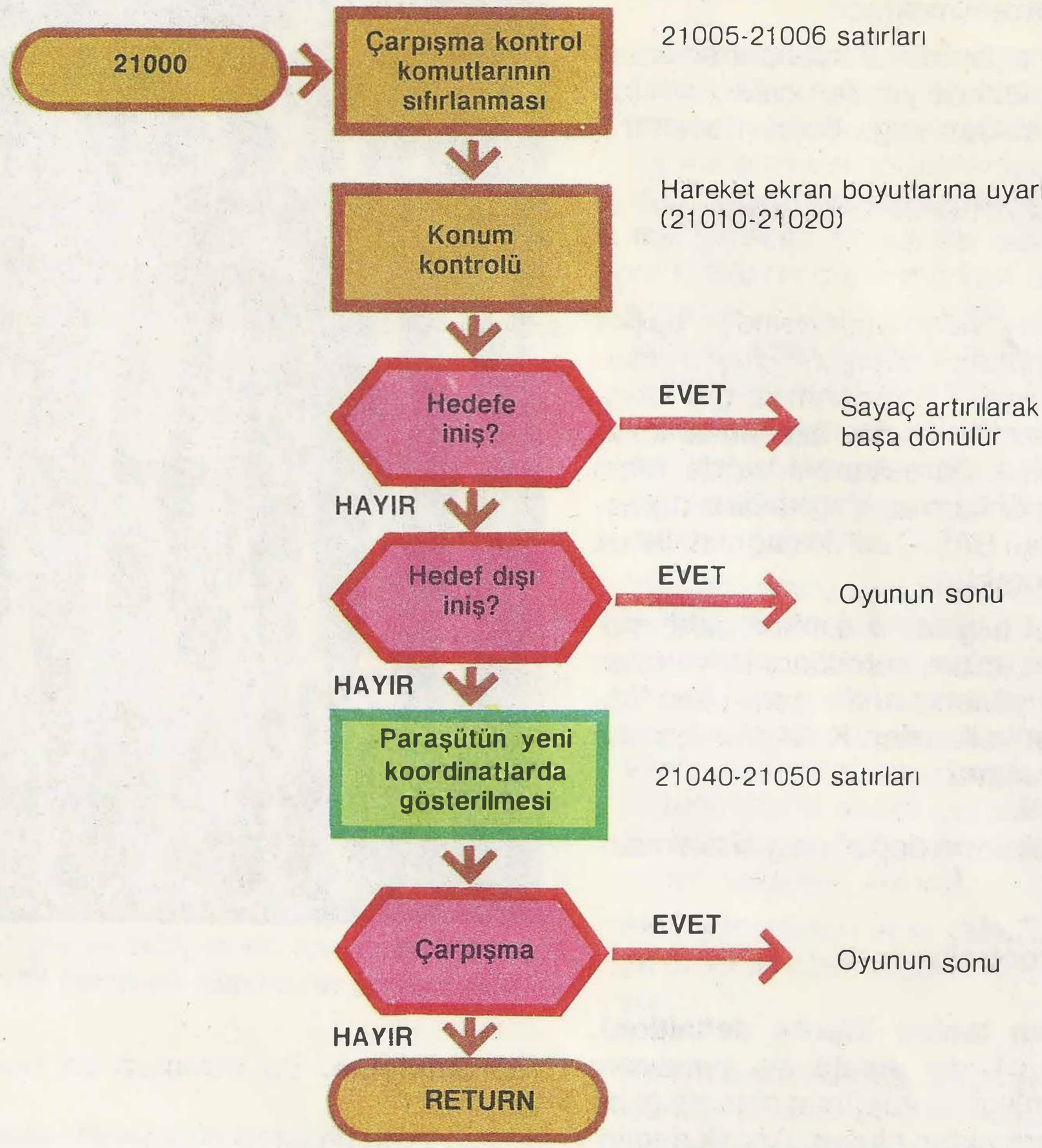
Burada;

P : Cismin önceliğini belirler (iki cismin ekranın aynı noktalarında gösteriminde çözümsüzlükten kurtulmak için)
X,Y :Koordinatlardır
C :Seçilen rengi belirten sayıdır ve 0 ile 15 arasındadır
N :SPRITE$(N) komutuyla tanımlanan cisim sayısıdır.

Burada da kullanılan sözdizimi Philips'e özgü olmakla birlikte BASIC MSX'i kabul eden bilgisayarlarda da kullanılabilir. Ancak bu komut cismin boyutlarını oluşturmaz. Bunun için daha önce SCREEN komutunu kullanmak gerekmektedir:

```
SCREEN A, B, C, D
```

HAREKET VE KONTROL ALT-PROGRAMI



PARAŞÜT PROGRAMININ CMB 64 VERSİYONUNDA KULLANILAN ÖZEL KARAKTERLER

CHR $	İşlevi	CHR $	İşlevi
147	Ekranı boşaltmak	144	Siyah yazım
18	Ters ON	31	Mavi yazım
146	Ters OFF	5	Beyaz yazım
159	Açık mavi yazım	17	İmleci aşağı hareket ettirmek
28	Kırmızı yazım		

Bellek konumları

650,128	Kendiliğinden tekrar
53248 + 28,8	Çok renkli cisim tanımı
53248 + 42 53248 + 37 53248 + 38	Cismin renkleri
53248 + 29,1	0 nolu cismin (10 tabanlı 1) X ekseni üzerinde iki kat büyütülmesi
53278 53279	Çarpışmalar

N cismini harekete geçiren komut. "n":

POKE S + 21, PEEK (S + 21) AND "n"

Yapay Zekâ

Öteden, Kadim çağlardan beri, insanı diğer bütün yaratıklardan ayıran şeyin ''zekâ'' ya da ''düşünmek'' olduğu söylenir. Ama ''zekâ'' ya da ''düşünce'' denen bu ayırdedici özelliğin nasıl bir şey olduğu, diğer nesnelere ve yaratıklara özgü hareketlilik biçimlerinde nasıl, hangi bakımlardan ayrıldığı, ancak 17. yüzyıldan itibaren merkezî bir önem kazanmıştır. Gerçi daha önce de, özellikle filozoflar matık başlığı altında ''düşünmek''ten ve kurallarından sözediyorlardı ama o zamanlar mantık daha çok bir **organon**, yani bir araç, olarak görülürdü. Dolayısıyla da, mantık başlığı altında, insanların düşünürken ne yaptıklarına dair değil, belli bazı şeyleri gerçekleştirmek, örneğin doğru olduğu bilinen önermelerden hareket ederek, doğru bir sonuç çıkarabilmek için ne yapmaları gerektiğine dair kurallar incelenirdi.

Oluşturucu Kurallar Düzenleyici Kurallar

Daha fazla ilerlemeden ''kural'' sözcüğünün çift anlamlılığı üzerinde duralım. Çünkü kural kavramı, yapay zekâ **(artificial intelligence)** araştırmalarının temelinde yatan bir kavram olduğundan ilerde sık sık başvurmak zorunda kalacağız. Bu bağlamda kural teriminin anlamı, gündelik dilde kasdettiğimiz anlama benzemekle birlikte, hem sınırları daha açıkça tanımlanmıştır hem de kendine özgü bir nüans kazanmıştır.

Yukarıda bir şeyin nasıl yapılmakta olduğuna dair olan kurallarla, nasıl yapılması gerektiğine dair olan kurallar arasındaki farktan sözettik. Bunlardan birincisine yani birşeyin nasıl yapılmakta olduğuna dair olan kurallara ''**oluşturucu (constitutive) kurallar**'', nasıl yapılması gerektiğine dair kuralara ise ''**düzenleyici (regulative) kurallar**'' denir.

Oluşturucu kurallar, herhangi bir faaliyeti, daha doğrusu kuralı oldukları faaliyeti mümkün kılan kurallardır; kendilerinden bağımsız olarak varolan bir faaliyeti düzenlemezler. Örneğin satranç oyununda, ilk hamle olarak şahın önündeki piyonu açmak oyunun başında belli bir üstünlük sağlar; dolayısıyla bu iyi bir kural olabilir. Ancak ilk hamle olarak başka bir taşı örneğin atı da, oynamak mümkündür. Yani insan bu kurala uymadan da satranç oynayabilir. Dolayısıyla, ''oyuna şahın önündeki piyonu açarak başlanır'' kuralı düzenleyici bir kuraldır.

Karşıt bir örnek olarak ''filler çapraz ilerler'' kuralını ele alalım. Bu kurala uyulmuyorsa, artık oynanan oyun satranç değil demektir. Dolayısıyla bu kural oluşturucu bir kural, satranç oyununu tanımlayan, oluşturan kurallardan biridir.

İşte yapay zekâ iddiaları, düşünce gerektiren bütün faaliyetlerin aslında bu tür oluşturucu kuralların bilgisine sahip olmak ve bu kuralları uygulamaktan ibaret olduğu kabulünden kaynaklanır. Bilgisayarların matematik problemlerinin çözümünde gösterdikleri başarının temelinde, matematik dediğimiz insan faaliyetinin oluşturucu kurallarının keşfedilerek formüle edilebilmiş olması yatmaktadır. Bu sayede bugün bilgisayarlar matematik becerisi konusunda, insanları çoktan aşmıştır. Zekâ dediğimiz şey de aslında nitelik olarak, matematik becerisinden farklı değildir. Çeşitli düşünce biçimlerinin temelinde yatan oluşturucu kurallar keşfedilip formüle edildikçe, bilgisayarların da bu düşünce biçimlerini taklit etmesinin önünde hiçbir engel kalmayacaktır.

Felsefe ve Yapay Zekâ

Yapay zekâ araştırmalarının birbirinden nisbeten bağımsız, iki hedefi olduğunu söyleyebiliriz. Bunlardan birincisi bütün bilgi işlem araştırmalarının ve genel olarak bilgisayar teknolojisinin ortak hedefi: İnsanların mevcut becerileriyle yapamadıkları ya da yeterli güvenirlik yada hassasiyet düzeyinde yapamadıkları ya da yapmak istemedikleri faaliyetleri makinalara yaptırmak. İkinci hedef ise daha yapay zekâ araştırmalarına özgü bir hedef: Bilgisayar modellerinden hareketle insan zekâsının özelliklerini araştırmak.

Bu iki hedefin varlığına bağlı olarak yapay zekâ iddiaları da ikiye ayrışmıştır. Bunlardan birinci gruba **teknik** ya da ılımlı yapay zekâ iddiaları diyebiliriz. Bu tür iddiaları savunanlar bilgisayarların insanlar gibi düşündüğünü, insanlarınki gibi bir zekâya sahip olduğunu iddia etmezler. Zekâ gerektiren işlerin çoğunu bilgisayarların da -insanlarınkinden farklı yöntemler kullanarak da olsa- yapabileceğini öne sürerler.

Felsefi ya da aşırı yapay zekâ iddiaları ise bilgisayarların insanların yaptığı herşeyi yapabileceğini, çünkü insanların zaten düşünürken, karar verirke bilgisayar gibi davrandıkları gö-

rüşünden ibarettir. Hatta daha da ileri giderek, insanların zaten birer bilgisayar, daha da doğrusu birer bilgi işlem sistemi **(information processing system)** olduklarını söyleyebiliriz.

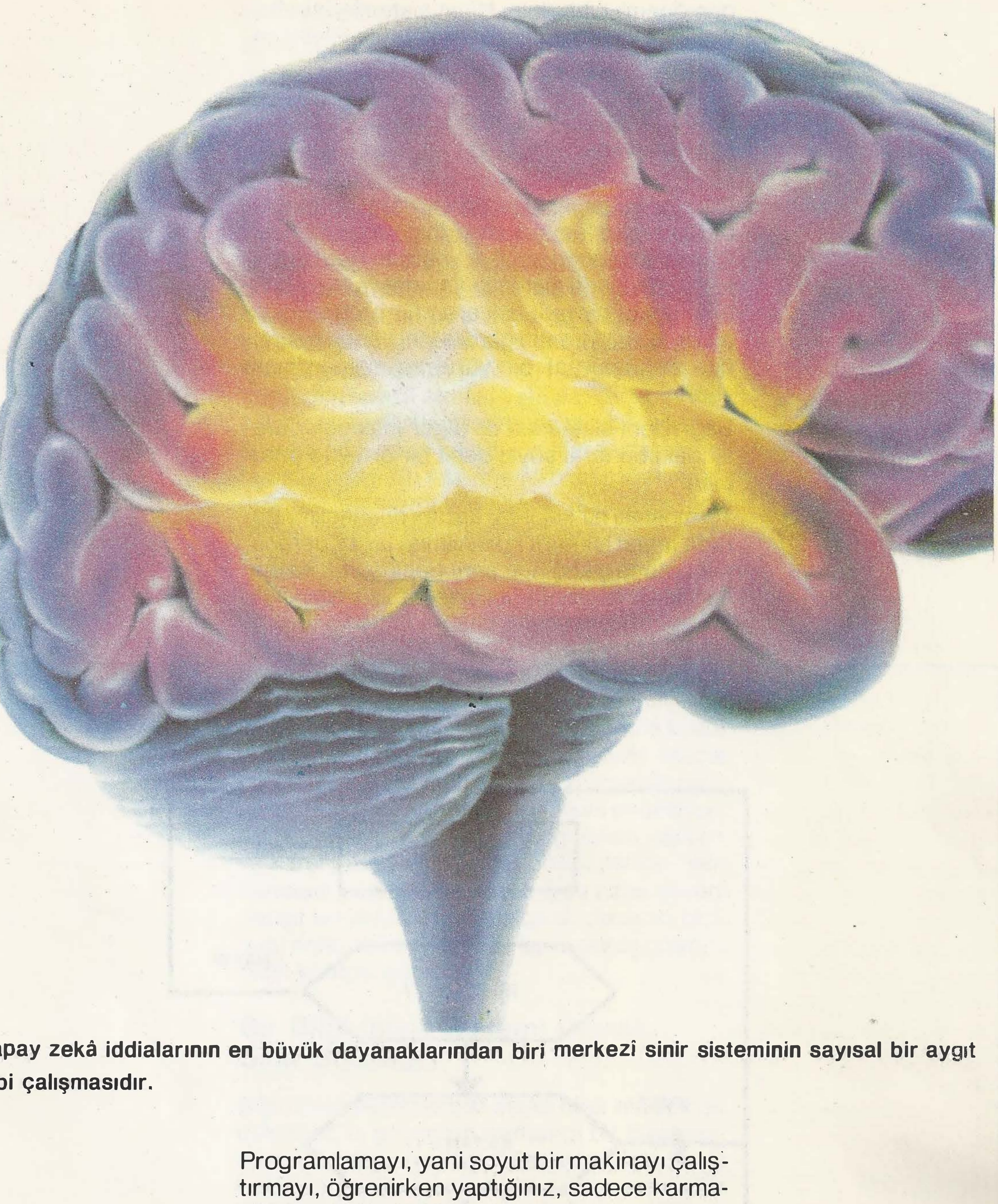
Soyut Bir Makine Olarak Bilgi İşlem Sistemleri

Bilgisayarlar hakkında söylenebilecek ilk şeylerden biri bunların birer makine olduklarıdır. Başka bir deyişle istediğimiz herhangi bir sonucu elde etmek için birbirinden bağımsız olarak tanımlanmış bir dizi işlemi koordine bir biçimde yaptırdığımız aygıtlardır. Ancak sayısal bilgisayarların ''düşünebileceği'', bu aygıtlar elektronik birer makine oldukları için iddia edilmiyor; bu makineler bir bilgi işlem sistemini (information processing system) örnekledikleri için ''düşünebilecekleri'' olasılığı üzerinde duruluyor.

Bilgi işlem sistemleri aslında maddi/fiziki varlıklarından bağımsız olarak tarif edilebilirler. Şöyle ki, herhangi bir bilgi işlem sisteminde, işlemcinin neden yapılmış olduğu, elektriksel, mekanik ya da optik enerjiyle çalışıyor olması sistemin yapabileceği işler bakımından önemli değildir. Önemli olan bilgilerin, sistemin kullandığı enerji cinsinden formel bir simgeler sistemine tercüme edilebilmesi ve bu bilgilerin, eninde sonunda birtakım çıktılara (işe) çevrilebilmesidir. Dolayısıyla yapabilecekleri işler bakımından bütün sayısal bilgi işlem sistemleri birbirlerine eşdeğerdirler; aralarında kullanılan enerjinin türünden kaynaklanan hız farkları bulunabilir ama herhangi bir bilgi işlem sisteminde çözülebilen bir problem ilkece, diğer bütün bilgi işlem sistemleri tarafında da çözülebilir.(*)

Buraya kadar söylediklerimizi şöyle de özetleyebiliriz: Bilgi işlem sistemleri aslında soyut birer makinedirler. Elektronik bilgisayarlar, kendisi soyut bir makine olan bilgi işlem sistemlerinin olası somutlaşmalarından yanlızca biridir.

(*) Burada bilgi işlem sistemlerini özellikle işlemcileri (processors) açısından karşılaştırmakta olduğumuzu vurgulayalım. Her bilgi işlem sisteminin, işlemciye ek olarak işlenmekte olan veya işlenecek olan bilgilerin depolandığı bir de belleği olması gerekir ve kuşkusuz, bellek kapasiteleri bakımından bilgi işlem sistemleri arasında hatırı sayılır farklar olabilir. Öyle ki, bellek kapasitelerinin sınırlılığı bazı bilgi işlem sistemlerini bazı problemleri çözmek için tamamiyle elverişsiz kılabilir. Ama bellekler, belli bir anlamda, bilgi işlem sistemine dışsaldır: Belli bir işlemcinin erişebileceği bellek alanı daima genişletilebilir.



Yapay zekâ iddialarının en büvük dayanaklarından biri merkezî sinir sisteminin sayısal bir aygıt gibi çalışmasıdır.

Programlamayı, yani soyut bir makinayı çalıştırmayı, öğrenirken yaptığınız, sadece karmaşık bir problemi birbirinden bağımsız olarak tanımlanmış, ayrık (discrete) simgeler üzerinde yapılan ve yine birbirinden bağımsız olarak tanımlanmış ve ayrık işler halinde ifade etmekti. Bu yüzden programlama üzerinde düşünürken devrelerin fiziki ve elektronik özellikleri üzerinde düşünmeniz gerekmedi.

Kuşkusuz her bilgi işlem sisteminin aynı zamanda fiziki bir ifadesi, başka bir deyişle bu soyut makinenin işleyişinin somut fiziki bir tekabülü olması gerekir. Ama konumuz açısın-

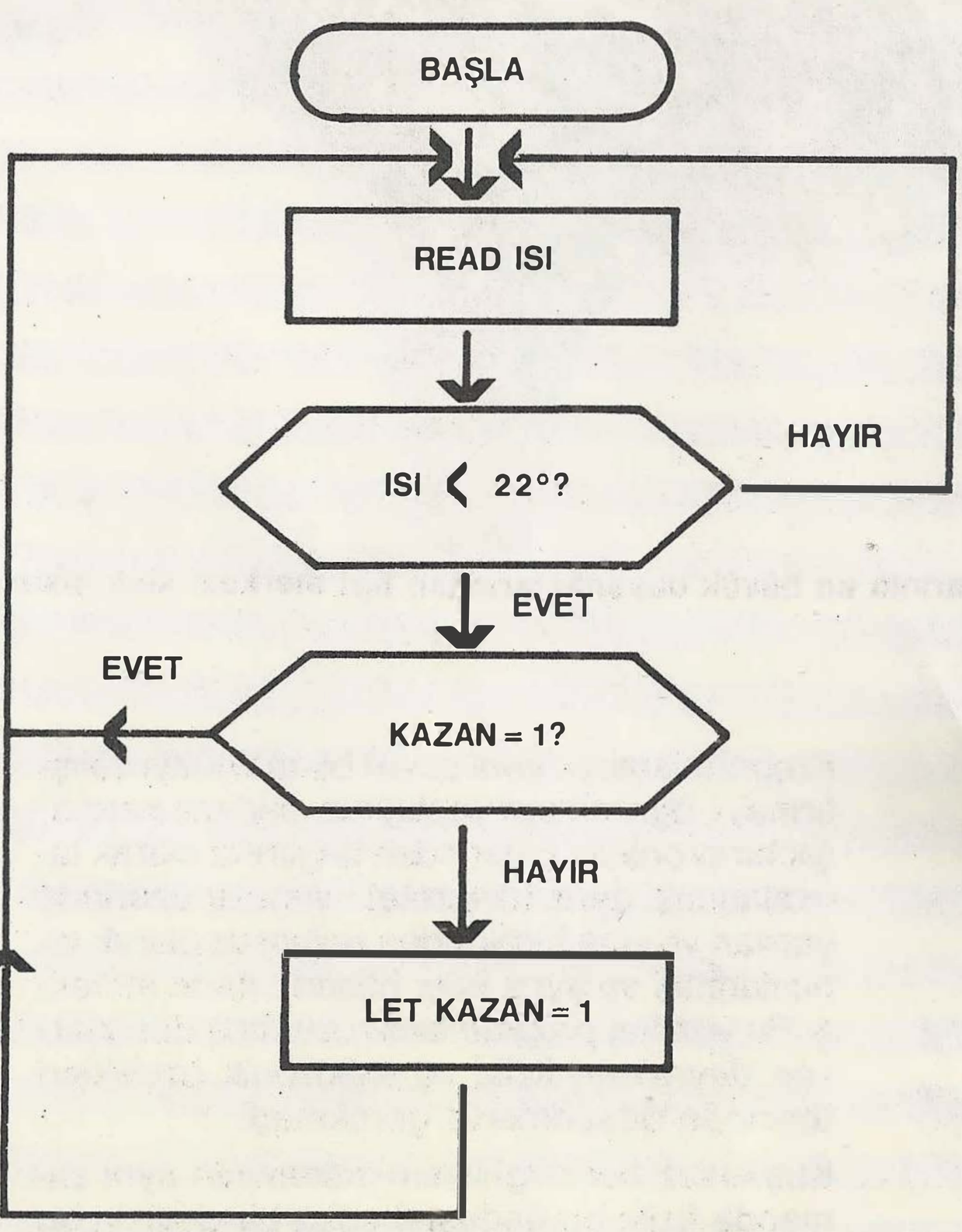
dan önemli olan, bilgi işlem sistemlerinin fiziki yapısından bağımsız olarak tarif edilebilmesidir.

Bilgi İşlem Sistemleri ve Diğer Makineler

Buraya kadar söylediklerimizden hareketle, bütün bilgi işlem sistemlerinin şu genel özelliğe sahip olduğunu söyleyebiliriz: Her bilgi işlem sisteminde sistemin yapması gereken işlerin soyut bir tarifini içeren bir program kısmı ile, bu programın gereklerini yerine getiren (işe dönüştüren) bir yürütme mekanizması vardır.

Bilgi işlem sistemleri ile diğer makineler arasındaki bu farkı şöyle basit bir örnekte görebiliriz.

Termostatlı bir kalorifer kazanı düşünelim: Burada metal bir şerit kullanılmış. Isı 22 derecenin üstüne çıktığında şerit genleşerek kazanı çalıştıran devrenin açılmasına ve dolayısıyla



kazanın çalışmasının durmasına yol açıyor.

Burada bir bilgi işlem sisteminden sözetmemize olanak yok. Çünkü istenen sonuç, yani kazanın ısı 22 derecenin altına düştüğünde sönmesi, sadece sözkonusu mekanizmanın fizik yasalara uygun olarak hareket etmesinin bir sonucu olarak gerçekleşmektedir. Başka bir deyişle, mekanizma, hangi koşullarda kazanın söneceğinin bilgisini içermekte uzmanlaşmış özel bir bileşen içermemektedir. Ancak bu mekanizma, mekanizmanın hangi koşullarda çalıştığını temsil eden bir bileşen içerdiği takdirde bir bilgi işlem sistemi sayılabilir. Kuşkusuz, bu bileşenin kazanın hangi koşullarda çalıştığını temsil etmesi yetmez, örneğin mekanizmanın içine yerleştireceğimiz bir kullanım kılavuzu da, onun hangi koşullarda çalıştığını temsil edebilirdi. Asıl önemli olan çalışma koşullarını temsil eden bileşenin, aynı zamanda mekanizmanın bilfiil işleyişini de kontrol etmesidir.

Bu tür bir gerçek bilgi işlem sistemine basit bir örnek olarak I. ciltte anlatmış olduğumuz (bkz. s. 25-6) Jacquard'ın dokuma tezgâhları için önerdiği mekanizmayı hatırlayabiliriz. Bu makinede dokuma tezgahının şişleri delikli bir kart tarafından yönlendiriliyordu; şöyle ki dokuma mekiğinin her geçişinde şişlerin önüne bir delikli karton çıkıyor ve bazı şişlerin geçmesine izin verirken bazılarınınkini engelliyordu. Burada gerçekten bir bilgi işlem sistemi ile karşı karşıyayız çünkü delikli karton hem kumaşın üstünde belirecek olan nihaî deseni temsil ediyor (kartona bakarak desenin biçimini anlayabiliriz) hem de bu desenin çizimini bilfiil kontrol ediyor.

Bir Bilgi İşlem Sistemi Olarak Sinir Sistemi

Bilgi işlem sistemlerinin soyut birer makine oldukları, iş görebilen herhangi bir mekanizma tarafından örneklenebileceğini söyledik. O halde, organik temelli mekanizmaların da bir bilgi işlem sistemini örneklemelerinin önünde ilkece hiçbir engel yoktur. Daha açıkça söyleyecek olursak, insan gövdesini de nörolojik bir bilgi işlem sistemi olarak düşünmememiz için hiçbir teorik gerekçe yoktur.

Tabii, insan gövdesini nörolojik bir bilgi işlem sistemi olarak görebilmemiz, onun gerçekte böyle bir sistem olduğu anlamına gelmez. Deminki termostatlı kazan örneğini hatırlayalım: Bu mekanizmaya dışardan baktığımız takdirde, mekanizmanın yan sayfada akış şeması ve-

rilen bir programa göre çalıştığını sanabilirdik. Dahası aynı bu termostatlı kazanın yaptığı işleri yapabilecek bir bilgi işlem sistemi tasarlayabilmek, bize baktığımız mekanizmanın da bir bilgi işlem sistemi olduğunu düşündürebilirdi.

O halde benzer bir noktayı da, yapay zekâ iddiaları ile ilgili olarak düşünmek zorundayız: Sinir sisteminin bir bilgi işlem sistemi imişcesine tarif edilebilmesi ve sinir sisteminin yapabildiği bazı işleri bir bilgi işlem sisteminin de yapabilmesi sinir sisteminin (ve dolayısıyla insan zekâsının) gerçekte bir bilgi işlem sistemi olduğunu göstermeye yeterli değildir. Sinir sisteminin bir bilgi işlem sistemi dolayısıyla insan düşüncesinin bir bilgi işlem süreci olduğuna dair elimizde başka kanıt var mı?

Bu konuda, kesin bir delil teşkil etmemekle birlikte, yapay zekâ iddialarını inandırıcı kılan en önemli şeylerden biri, sinir sistemi ve özellikle de sinir hücreleri (nöronlar) hakkındaki (yetersiz) bilgilerimiz. Her ne kadar, bir bilgi işlem sisteminin maddi/fizik yapısının onun doğası bakımından önemsiz olduğunu söylüyorsak da, sinir hücrelerinin sayısal ilkelere uygun bir biçimde, daha da somut olarak söylemek gerekirse transistörlere benzer bir biçimde,çalışıyormuş gibi görünmesi, insan zekâsının zaten aslında bir bilgi işlem sistemi olduğu iddialarına önemli bir dayanak sağlıyor.

Ne kastediyoruz sinir hücrelerinin sayısal ilkelere uygun bir biçimde çalıştığı iddiasıyla? Herhangi bir sinir hücresi, gelen uyarıcının şiddetiyle doğru orantılı bir biçimde tepki vermez; bu anlamda benzeşik (analojik) bir mekanizma değildir. Aksine, her hücrenin belli bir eşiği vardır; gelen uyarıcının şiddeti bu eşiği aştığında, hücre kadir olduğu en yüksek şiddette tepki verir. Dolayısıyla her sinir hücresinin durumu iki tabanlı sayı sistemi ile temsil edilebilir: 1 = nöron uyarılmış; 0 = nöron uyarılmamış.

Bilgi ve Davranış

Şimdiye kadar sık sık zekâ gerektiren faaliyetlerden sözettik. Ancak zekâ ve düşünce sadece birtakım faaliyetlerle uğraşabilme,birtakım işleri yapabilme becerisinden ibaret değildir, aynı zamanda bu iş ve faaliyetleri temsil edebilme yeteneğidir de. Bunu bir örnek üzerinde görelim.

Birisi için bir dili bildiğini söylediğimiz zaman kasdettiğimiz sadece o dili konuşabildiği değildir; yalnızca o dilde konuşmak değil, o dilde birşeyler yazabileceğini, o dilde yazılmış bir metni okuyabileceğini de farzederiz. Başka bir deyişle, dil bilgisi, bütün bu davranışları mümkün kılmakla birlikte, bu davranışların hiçbirine indirgenemeyecek birşeydir. İşte bilgisayarların insanlarınkine benzer bir zekâya sahip olabileceği (çünkü insanların zekâsının bilgisayarlarınkinden nitel olarak farklı olmadığı) iddialarına göre, dil bilgisinin bu kadar farklı biçimlerde ifade edilebilmesi aslında bu bilginin çeşitli davranışları temsil eden soyut (bu davranışların fiziki özelliklerinden bağımsız) bir program olmasından ötürüdür.

Yalnız dilbilgisi değil, en karmaşığından en yalınına kadar zekâ gerektiren faaliyetlerin tümü, bu iddialara göre, aslında sözkonusuf aaliyetleri temsil eden bir programın kontrolü altındadır. Daha yalın bir örnek verelim: Dans etmeyi bilmek öncelikle belli bir dans adımını oluşturan kurallara vâkıf olmak demektir. Bu kuralların bilgisini bir programa benzetebiliriz. Bir kez bu kuralların bilgisine vâkıf olunduktan sonra, bu bilginin nasıl ifade edileceği (nasıl ise dönüştürüleceği) programın yürütülmesine ilişkin, nisbeten arızi bir sorundur.

Dil bilgisi ile karşılaştırıldığından dans örneğinin önemi, dansederken kullanmakta olduğumuz kuralları, gerektiğinde (tam olarak formel bir dilde olmasa da) dile getirebilmemizdedir (iki adım sağa bir adım sola, gibi). Başka bir deyişle, dans kurallarının bilincindeyiz, oysa çoğumuz için aynı şeyi dilbilgisi kuralları için söylemek güç. Bilinç sorununun yapay zekâ iddiaları için önemine ilerde tekrar döneceğiz.

Şimdi yapay zekâ iddialarının daha somut biçimlerini gözden geçirmeden ve bu iddiaların geçerlilik derecesini değerlendirmeden önce, buraya kadar söylediklerimizi dört ana tez halinde özetleyelim:

— Bir bilgi işlem sistemi soyut bir makinedir.
— İnsan zekâsı sayısal bir bilgi işlem sistemidir.
— Zekâ gerektiren faaliyetler bir oluşturucu kurallar kümesinin yürütülmesinden ibarettir. Dolayısıyla zekâ bu oluşturucu kuralların bilgisidir.
— Bu oluşturucu kuralların bilgisi, zaten bir program şeklinde olduğundan, aynı bilgi bir bilgisayara da programlanabilir.

Kuralların Anatomisi

Bir örnek: M I U Sistemi

Yapay zekâ iddialarının temelinde, zekâ gerektiren faaliyetlerin gerçekten oluşturucu kuralların bilgisi olup olmadığı sorusu yatıyorsa, ''kural'' terimiyle ne kasdettiğimizi gözden geçirmemiz gerekir. Bunun için, Douglas Hofstadter'ın çeşitli alanlardaki yapay zekâ iddialarını gözden geçirdiği **Gödel Escher Bach** adlı kitabındaki formel kurallar sistemini kendi amaçlarımıza uyarlayacağız:

Bu sistem üç harf üzerine çeşitli işlemleri uygulayan dört kuraldan oluşuyor. Bu kurallar sayesinde şöyle diziler elde edebiliyoruz:

M U I
M U U I
M M U I
U M I
U M U M U M M I

Ancak bütün bu diziler mümkün diziler olmakla birlikte alıştırmamızın başında henüz bu dizilere ''sahip değiliz''. Alıştırmamızın amacı, 'sahip olduğumuz'' tek dizi olan M I'den hareketle ve kuralları kullanarak diğer dizileri elde etmek.

Kurallarımız şunlar:

1 / Elimizde son harfi I olan bir dizi varsa, bu harfin önüne bir U ekleyebiliriz. Yani:
M I = M U I
2 / Elimizde U M gibi bir dizi varsa, bu diziden U U M gibi bir dizi elde edebiliriz:
U M = U U M = U U U M
3 / Elimizde M I I gibi bir dizi varsa buradan M'yi atabiliriz.
4 / Elimizde iki U varsa bunların yerine M koyabiliriz.

Aksiyonlar Çıkarım Kuralları ve Türetmeler

Formel sistemlerde kurallardan bağımsız olarak verilmiş diziye yani bizim örneğimizdeki M I'ye, **aksiyom** adı verilir. Kullanılan simgelerin yerlerinin değiştirilmesini mümkün kılan kurallara üretim ya da **çıkarım kuralları** (ru-

les of production ya da inference), bu kurallardan hareketle yeni diziler elde etmeye ise **türetme** (derivation) denir.

Bu örnekte elimizdeki aksiyomdan hareketle şu dizileri türetebiliriz:

1 / M I (aksiyom)
2 / M U I (1'den kural 2'yle)
3 / M U U I (2'den kural 2'yle)
4 / M M I (3'ten kural 4'le)
5 / M M U I (4'ten kural 1'le)

Aksiyom, türetme gibi terimlerin örneklerini daha önce de görmüştük. Ansiklopedi'nin başında incelediğimiz ve bütün programlarda sık sık kullandığımız Boole cebiri de aslında bir formel kurallar bütünüdür. Bu sistemde aksiyomlar şunlardır:

A AND B
A OR B
NOT A
A XOR B

Burada A ve B gibi terimlerin basit ya da bileşik herhangi bir dizi için kullanıldığını gözönünde bulundurursak, şu türden dizilerin de aksiyomatik (yani çıkarım kurallarından bağımsız olarak) verildiğini görürüz:

A AND B AND C AND (D OR E) ...
A OR B OR C OR (NOT A) ...
NOT (NOT A) XOR B ...

Ancak bu sistemi incelerken aşağıda verilen ve ''de Morgan kuralları'' diye anılan türde çıkarım kuralları üzerinde fazla durmamıştık:

NOT (A AND B) = NOT A OR NOT B
NOT (A OR B) = NOT A AND NOT B

Boole cebirini nasıl olup da bu tür çıkarım kurallarına başvurmadan kullanabildiğimiz konusunu daha sonra işleyeceğiz.

Karar Yordamları

MIU sisteminde, türetmeleri kendi kendinize sürdürürseniz, büyük bir olasılıkla bir süre sonra, türettiğiniz dizilerin birtakım ortak özelliklerinin farkına varmaya başlarsınız. Hatta sadece kuralları incelediğiniz takdirde bile, türetilebilecek dizilerin genel bir özelliğinin farkına varabilirsiniz: Bazı kurallar M'lerin ve U'ların sayısını arttırıp (1 ve 2) bazılar eksiltir-



ken (3 ve 4), kurallardan hiçbirisi I'ya ilişmemektedir. Bu yüzden bütün türetmeler boyunca I aksiyomdaki varlığını sürdürmekte, sabit kalmaktadır. Başka bir deyişle, bu sistem için I, mümkün bütün dizilerde bulunması gereken zorunlu bir simgedir.

Dizilerin ikinci bir ortak özelliği ise kısmen bu özelliğe bağlı: I dizilerin sadece sonunda yer alabiliyor. 1. 2. ve 4. kuralların birlikte işleyişi U'nun ve M'nin birbirlerine göre yerlerinin değişmesi sonucunu doğurabiliyor. Oysa hiçbir kural bu simgelerin I'ya göre yerlerini değiştirmiyor. (Dolayısıyla sistemin adı olarak kullandığımız MIU bu sistemde türetilebilecek bir dizi değildir).

Dolayısıyla herhangi bir dizinin, bu formel ku-

rallar sistemi tarafından türetilebilecek bir dizi olup olmadığına karar vermeye çalışırken, eğer sözkonusu dizinin bir I harfi içermediğini ya da harfinin dizini sonunda yer almadığını farkedersek, hemen bu dizinin bizim sistemimiz içersinde mümkün olmayan bir dizi olduğuna karar verebiliriz. İşte herhangi bir dizinin mümkün bir dizi olup olmadığına karar verirken münferit dizilerin özelliklerinden hareket eden ve bizi uzun türetme işlemlerinden kurtaran bu tür yordamlara **karar yordamı** (decision procedure) diyoruz.

Şimdi tekrar Boole cebiri örneğimize geri dönebiliriz: Boole cebirini incelerken çıkarım kuralları üzerinde fazla durmamız gerekmedi çünkü herhangi bir dizinin mümkün (diğer bir diziden çıkarılabilir) bir dizi olup olmadığına karar verirken kullanabileceğimiz ve çok kesin sonuçlar veren bir karar yordamımız vardı: Doğruluk değerleri tablosu.

Kuşkusuz, doğruluk değerleri tablosu ile MIU sistemine ilişkin olarak keşfettiğimiz karar yordamları arasında önemli farklar var. Bir kere doğruluk değerleri tablosu, bizim keşfettiğimiz yordama göre çok daha kesin sonuçlar veren bir yordam. İki dizi verildiğinde, birinin diğerinden çıkarılabilip çıkarılamıyacağını kesin olarak tesbit ediyor. Oysa bizim yordamımız sadece bazı dizilerin, (sonu I ile bitmeyen dizilerin) sisteme dahil olamayacağını gösteriyor. Ama sonu I ile biten bir dizi için yine de aksiyomdan hareketle ve kuralları uygulayarak sözkonusu diziyi türetmeye çalışmak zorundayız.

İkisi arasındaki daha önemli ikinci bir fark, bizim yordamımızın sadece dizilerin biçimsel özelliklerini gözönünde bulundurmasından kaynaklanıyor. Oysa doğruluk değerleri tablosu yordamını kullanırken, ilgilendiğimiz dizileri, belli bir biçimde ve sistematik olarak yorumluyoruz. Yani her iki dizideki bütün elemanlara belli bazı değerler vererek dizilerin bu şekilde yorumlandıklarında kurdukları ilişkileri inceliyoruz.

Ama bu noktada bizi daha çok ilgilendiren, farklardan çok iki yordam arasındaki benzerlikler. Her iki yordam da, uygulanmakta olan kurallar sistemine belli bir anlamda dışardan bakmayı içeriyor. MIU sistemi içinde bütün dizilerin I ile bitmekte olduğu, herhangi bir insan gözünün hiç çaba sarfetmeden farkedebileceği bir özellik. Ama bu, sözkonusu özelliğin çıkarım kurallarıyla temsil ettiğimiz işlemler tarafından keşfedilemiyeceği gerçeğinin gözden kaçırılmasına yolaçmamalı.



Doğruluk değerleri tablosu örneğinde sisteme dışardan bakmak (ve hatta müdahale etmek) gerekliliği çok daha bariz. Ama her iki sistemde de kuralların işlevi, simgeleri birbirine dönüştürmekle ya da yerlerini değiştirmekle sınırlı. Bu değişikliklerin sonuçlarını kaydetmek ve değerlendirmek, çıkarım kurallarının işlevlerinden değil.

Yapay Zekânın İki Kuşağı

Yapay Zekâ (YZ) tartışmaları popüler bilim kurgu yazınının spekülatif alanından çıkıp bilimsel araştırma programlarının bir konusu durumuna geldiğinde olay pek karmaşık görünmüyordu. Sorun sadece bir teknoloji sorunu olarak görüldüğünden herşey üç beş yeni keşfe bağlı sanılıyordu. İnsan düşüncesinin kurallarını keşfetmek, beynin çalışmasının elektronik benzetimini ***(simulation)*** *kurmak, yapay zekâyı gerçekleştirmeye yetecekti. Yüzyıl başından beri geliştirilmekte olan simgesel mantığın düşüncenin kurallarını temsil edebilemeye hayli yaklaşmış olduğuna zaten inanılıyordu. Artık bütün iş, insan zihninde depolanmış bilgiyi, simgesel ve formel bir dile dönüştürerek ''uzman sistemler'' adı verilen programları yazmaktan ibaretti. Bu yaklaşıma ''birinci kuşak yapay zekâ'' adını verebiliriz. Açık seçik simgelerle kodlanmış bir veri kümesini, kıyaslama ve hesaplamaya dayalı bir kurallar sistemiyle işleyen sözkonusu programlar oldukça başarılı satranç oyunları çıkarıp Japonca'dan İngilizce'ye teknik çeviriler yapıyor; belirli semptomları değerlendirip teşhis koyabiliyordu.*

Ama bir süre sonra yolun sonuna gelindi; uzman sistemler düşünemiyorlar, hele yaratıcı hiç olamıyorlardı. En belirgin çıkmaz, kurallara dayalı programların korkunç ayrıntılar içinde boğulmasıydı. Laboratuar koşullarında bile bir robot-el'e kibritle sigara yaktırmak neredeyse olanaksız düzeyde karmaşık kurallar yığını gerektiriyordu. Çok geniş bir beklenmedik olaylar ortamında zeki davranışlar geliştirecek bir program pratik olarak olanaksız hale geliyordu.

Üstelik yaklaşımın temel kabullerinin tutarlılığından da kuşku duyulmaya başlandı. Soyut simgeler kullanan YZ'nın en önemli kabullerinin insan yapısına uyduğu hakkında kuşkular beslenmeye başlandı.

Bu kuşkular özellikle insan düşüncesinin gövdeden bağımsız olduğu ve dolasıyla bu düşüncenin formel simgelerle taklit edilebileceği kabulleri çevresinde yoğunlaşıyordu. Örneğin renklerin algılanması üzerine yürütülen araştırmalar, insanların renkleri yalnız dalga boylarına göre değil, insan nöropsikolojisindeki farklılıklar nedeniyle, çok daha karmaşık ve çok yönlü etkileşimler altında sınıflandırdığını ortaya koyuyordu. Örneğin, ''mavi'' sözcüğünün yalnızca kapsadığı renk alanının dalgaboyunun sınırları değil standart çağrışımları da bir kültürden diğerine önemli farklılıklar gösterebiliyor.

Klasik yapay zekâ yaklaşımı için temel dayanaklardan biri nöronlarla transistörler arasında kurulan paralellikti. Ancak bu paralellik veri kabul edildiğinde, nasıl olup da beyindeki nöron sayısına eşit ya da hatta daha fazla sayıda transistör içeren bilgisayarların, bazı konularda beynin kapasitesine erişemediği bir muamma oluyordu.

Bir nöronda bir elektrik sinyalinin üretimi, herhangi bir minik işlemcinin bir transistöründe bir bit bilginin işlenmesinden yaklaşık 100 kere daha uzun zaman alıyor. Elektronik devrelerin bu üstün hızına karşın beynin henüz erişilememiş gücü nereden kaynaklanıyor. Araştırmacılar henüz kesin olarak kanıtlayamasalar da, bu soruya şöyle bir cevap öneriyorlar: Belli bir bilgiyi işlemekte olan elektronik devre tesbih benzeri bir dizisel bağlantı içersinde yeralır. Oysa her nöron 10.000 diğer nöronla bağlantılı yoğun bir ağ içersinde yeralmaktadır. Böylece beyin aynı anda çok daha büyük miktarlarda veriyi işleyebilme yeteneği olan bir paralel işlemci gibi çalışarak hız eksikliğini telafi edebiliyor.

Beyni böyle de görebilme olanağı, 1980 başlarında bir grup YZ araştırmacısının, uzman sistemler amacını güden ve düşünceyi bir kurallar manzumesinden ibaret gören klasik yapay zekâ yaklaşımını terketmesine yolaçtı. Kendilerini yeni ''bağlantıcılar'' diye adlandıran bu genç araştırmacılar aslında Gestalt psiokolojisini yeniden keşfediyorlardı çünkü zekânın,

birtakım kuralların sistematik olarak uygulanması değil, akli olarak temellendirilemese de, verili bazı bütünlüklerin, biçimlerin (Gestaltlerin) keşfi olduğu kabulünden hareket etmeye karar verdiler. Düşünmenin üstüste kurulan bağlantı biçimleri arasında hedeflenen biçimin, kurallara bağlı değil, neredeyse ''tesadüfi'' bir şekilde belirdiği bir süreç olduğu iddiasını araştırmaya başladılar. Pullitzer ödülü sahibi Dr. Hofstadter'ın örneği, bu sürecin basite indirgenmiş de olsa bir örneğini veriyor:

Pazar günü gazetenin eğlence sayfasında bulmacalara bakıyorusunuz: Anagram biçiminde bir bulmaca var: İRGLİASBYAA. İstenen bu anlamsız harfler dizisinden bir sözcük çıkarabilmeniz (...) Pek aritmetiğe benzemiyor bu, istenen biçimin nasıl keşfedileceğine dair hiçbir yol gösterici kural yok. SRGAYİİLBYA ya da GASYRBLİLAİ gibi anlamsız bileşimler geliyor akla (...) Dr. Hofstadter bu tür bilmeceleri çözecek bir program yazmaya çalışıyor. Gerçi mümkün bütün harf bileşimlerini yazıp, bu bileşimleri verili sözcük dağarcığındaki sözcüklerle karşılaştıracak bir program yazmak mesele değil; ama Hofstadter'ın amacı bu değil. Bu tür bir problemi çözmek için, böylesi çıplak bir hesap kitap gücüne başvurmak, bir ceviz kırmak için atom bombası kullanmak gibi birşey olurdu. Çünkü bu tür bir program SİYLBAAİGR gibi bir seçeneği değerlendirmek için bile ciddi ciddi düşünecektir. Oysa Dr. Hofstadter bilincin ve mantığın ötesindeki akışkanlığı arıyor. Bu inancına bağlı olarak Hofstadter, kurallara bağlı olarak akıl yürütmenin, düşüncenin ilk değil son adımı, buzdağının su üzerinde görünen kısmı olduğuna inanıyor. Bu yüzden genel bir algoritma ya da kurallar kümesi değil birbirinde bağımsız olarak hareket eden bir sürü ''kod''cukla başlamayı tercih ediyor. ''Yapılan iş mini minnacık yapılar için bir sürü algoritma yazmaktan ibaret. Sonra bırakıyorsunuz onları birbirleriyle etkileşsinler. Gerçi onların nasıl etkileşiceğine dair algoritmayı da kendiniz yazıyorsunuz ama, bir anlamda bu algoritma onların yüzer gezer bir halde birbirleriyle etkileşmesini sağlıyor. Aslında yaptığınız onların determinist olmayan bir biçimde birbirleriyle etkileşmesine izin vermek ve zekâ dediğimiz şey de birbirleriyle birlikte çalışma derecelerinin bir ürünü olarak ortaya çıkıyor zaten.''

Bunlardan da anlaşılabileceği gibi, yeni ''bağlantıcı''lar yapay zekâ sorunun çözümü için en iyi yolun hesaplamalara ilişkin kurallar yazma yerine, sistemin kendi başına ve yönlendirilmeksizin farklı bağlantıları deneyip, doğru çıktılara uyan bağlantı konfigürasyonları belirlemesi olduğuna inanıyorlar. Bu yaklaşım ikinci kuşağı birinci kuşak yapay zekâcılardan birkaç noktada ayırıyor:

- *Birincisi, bağlantıcılar donanıma özel bir önem veriyorlar. Çünkü yüksek düzeyli simgesel işlem, bu işlemin yapıldığı donanımdan soyutlanamaz ve dolayısıyla bu tür işleme uygun devreler gerektirir.*
- *İkincisi, bağlantıcılar, dizisel yerine paralel yapılarla çalışmaya kuramsal bir önem atfediyorlar.*
- *Üçüncüsü, bağlantıcıların tercih ettiği makineler çok daha ''yaygın'' bir veri işlem sistemi kullanıyorlar: Bu sistemlerde bellek ve işlemler tüm şebekeye dağılmış durumdadır ve bunların üzerindeki merkezi kontrol çok zayıftır.*
- *Son olarak, bağlantıcı makinelerin çoğu ancak asgari ölçüde programlanmıştır. Yanlızca genel komutların verildiği bu sistemlerde şebeke, ayrıntılı algoritmalara başvurmaksızın denge durumlarını neredeyse deneme yanılma yöntemini kullanarak kendi bulur.*

Ama birinci kuşak ve ikinci kuşak yapay zekâcılar yine de aynı aileye mensupturlar. Donanıma verdikleri onca öneme karşı bağlantıcılar zekâyı, fiziki, maddi ve biyolojik yapısından bağımsız bir şey olarak görmeye devam ederler. Örneğin, zekâyı Gestaltler, bütünlükler, 'desen'ler (patterns) bulmak olarak tanımlayan Hofstadter, zekânın kendisini de bir desen olarak görmektedir: ''Desenin nöronların ya da silikon yongaların (chips) sinyalleri ya da hatta bir karınca kolonisinin yürüyüşü tarafından örnekleniyor olması önemli değildir.''

Şöyle de söylenebilir: çıkarım kuralları objesi M, I, U gibi harfler ya da A, B gibi simgelerle simgelenen önermeler olan bir düşünce tarzının formelleştirilmiş biçimi. Oysa bir karar yordamı keşfederken kullandığımız düşünce, kurallarının kendisini ve işleyişlerini obje edinen bir düşünce tarzı.

Kuşkusuz bir kez keşfedildikten ve dile getirildikten sonra karar yordamının kendisini de kurallara ekleyebiliriz. Ama yapay zekâ iddiaları açısından bizi ilgilendiren, karar yordamının temelinde yatan düşüncenin kurallaştırılabilip kurallaştırılamayacağı. Çünkü yapay zekâ iddialarının temelinde insan düşüncesinin birtakım kuralların uygulanmasından ibaret olduğu görüşünün yattığını söylemiştik. Oysa bu örnekler insan düşüncesinin çıkarım kuralları ile temsil edilemeyen bir yönü olduğunu gösteriyor. Öyle görünüyor ki, insanlar sadece kuralların uygulanmasından ibaret olan faaliyetlerle uğraşırken bile, bir yandan da bu kuralları uygulamalarını izliyor, kurallardan çıkan sonuçları kaydediyor, yorumluyor ve bu sonuç ve yorumlarından yeni genellemelere gidiyorlar.

Bu, yapay zeka iddiaları için çok ciddi bir engel teşkil edebilecek bir olgu. Çünkü böyle bir olgudan çıkarılabilmesi mümkün olan sonuçlardan biri şu: Her ne kadar insanlar birtakım kuralları mekanik (yani bir bilgisayarın da yapabileceği) bir biçimde uygulamaya muktedir ise de, asıl zeka dediğimiz şey bu kuralların uygulanmasından ibaret değildir. Aksine kuralları sezgisel, algısal bir şekilde görebilmesi, sonuçları mekanik olmayan bir tarzda keşfedebilmesidir. Asıl zeka dediğimiz şey de, bu ikinci, mekanik olmayan düşünce tarzıdır.

Çünkü, insan düşüncesinin birtakım kuralların uygulanmasından ibaret olduğunu iddia eden birine daima şu soruyu sorma hakkımız vardır: Kuralların nasıl yorumlanacağı, daha da önemlisi nasıl uygulanacağı nasıl belirlenecektir?

Bu bizi zekânın ancak farklı düzeylerdeki kurallarla temsil edilebileceği sonucuna götürür. Yalnızca bazı işleri yapmamızı mümkün kılan kuralların bulunması yetmez; kendisine bu kuralları obje alan "üst-kurallar", kısacası bir kurallar hiyerarşisi bulunmalı.

Ancak bu takdirde de bir sonsuz döngünün içine girmiş oluyoruz. Birinci düzeydeki "sıradan" kuralların nasıl yorumlanıp uygulanacağını belirleyen üst-kuralların bulunduğunu söylemek, bu kez de o üst-kuralların nasıl yorumlanıp uygulanacağı sorusuna yol açacaktır. Bugünkü yapay zeka tartışmalarının merkezinde yatan sorun da budur zaten. Yapay zeka imkanını savunanlar, en üst bir düzeyin bulunmadığını, zekanın alt düzeylerin de üst düzeyleri etkileyebildiği, kendi üzerine dönen (hiyerarşik değil heterarşik) bir yapı olduğunu söylüyorlar. Eleştirmenler ise insanların kuralları yorumlayabilme ve uygulayabilme yeteneklerinin kendisinin, kurallar tarafından temsil edilemeyeceğini iddia ediyorlar herhalükârda buraya kadar geliştirdiğimiz biçimiyle formel bir kurallar sistemi kavramı, insan düşüncesinin aynı zamanda kurallar üzerine de düşünmek olduğu gerçeğini yansıtacak esneklikte bir kavram değil. Bu sorunun yapay zekâ araştırmalarında nasıl kavramsallaştırıldığını ve çözülmeye çalışıldığını daha sonra anlatacağız. Şimdi formel kurallar sistemlerinin bir başka sorunlu yönünü gözden geçirelim.

Sınıflandırma Sorunları

Buraya kadar gözden geçirdiğimiz sistem, formel kurallar sistemi ile ilgili önemli bir sorunu sergilemiyor. MIU sisteminde kuralların, hakkında oldukları simgeleri (yani M I ve U harflerini) tanımalarının bir sorun yaratmayacağını farzettik. Basılı harfler gibi, bir kullanılışından diğerine değişmeyen anlamsız simgeler sözkonusu olduğunda, böyle bir kabulde bulunmak sakıncalı olmayabilir. Ancak insan düşüncesi nadiren böylesi anlamsız ve sadece biçimsel düzeyde tanımlanmış simgelerle uğraşır.

Sorunu özetleyebilmek için önce MIU sistemi hakkında bir ipucu verelim: Hofstadter'ın sistemini uyarlarken kısmen Türkçe'ye benzeyen ancak çok daha ilkel ve sınırlı bir dil düşündük. İçinde şu türden (ve ancak şu türden) cümlelerin kullanılabileceği bir dil:

Tren iyi gitti.

İyi tren gitti.

Güzel yakışıklı adam geldi.

Kadın rahat iyi uyudu.

Başka bir deyişle, biz kuralları yazarken M I ve U gibi simgeler değil, isim (= M), sıfat (= U)

ve fiil (=I) gibi sözcük sınıfları düşünüyorduk. Ancak kuralları M, I, U gibi anlamsız simgelerle değil de, sözcük sınıflarıyla ilgili olarak yazsaydık, o zaman daha ilk aşamada neyin isim, neyin sıfat, neyin fiil olduğuna bilgisayara nasıl bildireceğimiz sorunu ile karşı karşıya kalacaktık.

Daha genel bir deyişle söyleyecek olursak, kuralların objeleri, salt biçimsel düzeyde tanımlanmış ve çeşitli kullanılışları arasında bir fark olmayan şeyler olmaktan çıkıp anlamlı birimleri biraraya getiren sınıflar olmaya başladığı andan itibaren, formel sistemler yeni bir bileşenin eklenmesini gerektirirler: çıkarım kurallarının yanında sınıflandırma (categorization) kuralları (rules) ya da ölçütleri (features)...

Formellik gerekliliği, çıkarım kuralları için olduğu kadar sınıflandırma ölçütleri için de geçerlidir. Yani belli bir sınıfın ölçütü olan özellik her durumda, karşılaşılan belli bir nesnenin sözkonusu sınıfa dahil olup olmadığını tayin etmemizi sağlayacak açıklıkta dile getirilmiş olmalıdır.

Bu açıklık gerekliliğinin nasıl sağlandığını bir örnek üzerinde görelim. Ama artık örneğimizde MIU sistemini kullanamayacağız. Çünkü bu sistemde, M, I ve U gibi harflerin sözcük sınıflarına işaret ettiğini söylemiştik. Biz herhangi bir sözcüğün isim mi sıfat mı, fiil mi olduğunu söyleyebiliriz. Ama bu bilgimizin temelinde açık seçik ölçütlerin varolup olmadığı, böyle ölçütler varsa bunların neler olduğu dilbilimcileri en çok uğraştıran konulardan biridir.

Bu yüzden biz örneğimizi çok daha tanımlı ve dar bir alandan seçeceğiz. Amacımız yapay zekâ iddiası ile kasdedilen işlem tarzını olabildiğince basit örnekler üzerinde göstermek.

Bir Örnek:

Pratisyen Bir Bilgisayar

Hastalık tedavisini bir bilgi işlem süreci olarak düşüneceksek, sürecin bütününü iki evreye ayırabiliriz: Teşhis ve tedavinin kendisi. Tedavinin kendisini bir koşullu komutlar dizisinden oluşan bir program olarak düşünmek mümkün. Ama hem daha çetrefil olduğundan hem de konumuz sınıflandırma sorunu olduğundan, bizi burada ilgilendiren teşhis evresi.

Sorumuz şu: Bir doktorun hastasının rahatsız-

lığını teşhis ederken kullandığı düşünce sürecini bir bilgisayara da programlayabilir miyiz? Bilgisayar teşhis yapabilir mi? Daha da önemlisi, teşhis yapmakta olan gerçek bir doktor gibi düşünebilir mi?

Öğretici amaçlarla geliştireceğimiz program yalnızca beş hastalık ve bu hastalıklarla ilgili sekiz semptom tanıyacak. Bu verileri, doktorun eğitimi sırasında edinmiş olduğu bilgilere benzetebiliriz:

	Ateşi var mı?	Öksürüyor mu?	Burnu akıyor mu?	Boynunda sertlik var mı?	Boğazı ağrıyor mu?	Başı ağrıyor mu?	Lenflerinde şişlik var mı?	Vücudunda lekeler var mı?
Nezle	0	0	1	0	0	0	0	0
Grip	1	0	0	0	0	1	0	0
Faranjit	1	1	0	0	1	0	0	0
Kızıl	1	0	0	0	0	0	1	1
Menenjit	1	0	0	1	0	1	1	0

O halde ilk yapmamız gereken şey, bu hastalıkları onlara ilişkin semptomlarla birlikte depolamak.

Bunu gerçekleştiren dökümü yan sayfada verilen programın 10000 alt-programında verilmiştir. HAS$(5) hastalık adlarını, iki boyutlu SEM(5,8) dizeyi ise semptomları içermektedir (1 semptomun sözkonusu hastalıkta görüldüğünü 0 ise görülmediğini gösteriyor). PU(5) dizeyinin ne işe yaradığını ise daha sonra göreceğiz. Programın kendisi oldukça basit. Şikayeti olan hastaya sözkonusu sekiz semptomun kendisinden bulunup bulunmadığının (100-250. satırlarda) sorulmasından ve 500. satırda girilen cevabın SEM dizeyindeki değerlerle karşılaştırılmasında ibaret. 510. satırdaki CE değişkeni girilmiş olan cevabı (CEV$) kodluyor (Evetse 1, Hayırsa 0). 520-550. satırlar arasındaki döngü ise karşılaştırmayı yapıyor. Burada PU(.) dizyeinin işlevini de görüyoruz: Her satırda (N = satır sayacı) kodlanmış cevap, o satırın M. sütünundaki (M = sütun sayacı) değerle uyuşuyorsa PU(N) yani puan dizeyinde o satıra tekabüle eden satırın değeri 1 arttırılıyor.

Ayrıca programın nasıl çalıştığının daha iyi kavranmasını sağlamak için, 530. satır her sorudan sonra hastanın verdiği cevaba göre semptomun görüldüğü ya da görülmediği hastalıkların tüm bir listesini veriyor.

```
10 GOSUB 10000
110 PRINT "ATESI VAR MI?"
120 GOSUB 500
130 PRINT "OKSURUYOR MU?"
140 GOSUB 500
150 PRINT "BURNU AKIYOR MU?"
160 GOSUB 500
170 PRINT "BOYNUNDA SERTLIK VAR MI?"
180 GOSUB 500
190 PRINT "BOGAZI AGRIYOR MU?"
200 GOSUB 500
210 PRINT "BASI AGRIYOR MU?"
220 GOSUB 500
250 PRINT "LENFLERINDE SISLIK VAR MI?"
240 GOSUB 500
250 PRINT "VUCUDUNDA LEKELER VAR MI?"
260 GOSUB 500
270 PRINT:PRINT "PUAN"
280 PRINT
300 FOR N= 1 TO 5
310 PRINT AR$(N),PU(N)
320 NEXT N
400 PRINT
410 RUN
420 RUN
500 INPUT CEV$
510 M=M+1:CE=1:IF CEV$="H" THEN CE=0
520 FOR N=1 TO 5
530 IF SEM(N,M)=CE THEN PRINT HAS$(N):PU(N)=PU(N)+1
540 IF N=1 AND M=1 AND CE=1 THEN 550:M=0
550 NEXT N
560 RETURN
10000 DIM HAS$(5), SEM(5,8),PU(5)
11000 DATA NEZLE, 0,0,1,0,0,0,0,0
12000 DATA GRIP,1,0,0,0,0,1,0,0
13000 DATA PARANJIT,1,1,0,0,1,0,0,0
14000 DATA KIZIL,1,0,0,0,0,0,1,1
15000 DATA MENENJIT,1,0,0,1,0,1,1,0
16000 FOR N=1 TO 5
17000 READ HAS$(N)
18000 FOR M=1 TO 8
19000 READ SEM(N,M)
20000 NEXT N,N:M=0
21000 RETURN
```

Yan sayfalarda programın iki çalışma örneğini görüyorsunuz. Önce birinci örnek üzerinde duralım: Her sorudan sonra program sözkonusu semptomun kendi tanıdığı hastalıklar içersinden hangilerinde görüldüğünü (ya da verile cevaba göre görülmediğini) bildiriyor ve bütün sorular sorulduktan sonra sonra bir puan cetveli yazıyor. Birinci hastada faranjitin bütün semptomları görüldüğünden, faranjit 8 puan alıyor ve teşhiste bir sorun çıkmıyor.

Ancak gördüğünüz gibi diğer hastalıkların da puanı 0 değil. Bu puanları çeşitli hastalıklar arasındaki semptomlar açısından benzerliğin bir ifadesi olarak yorumlayabiliriz.

Ancak ikinci örneğin durumu bir anlamda daha ilginç: Formel sınıflandırmalarla, gündelik hayat ve hatta tıp gibi oldukça formelleştirilmiş disiplinlerde bile kullanılan sınıflandırma ölçütleri arasındaki farkı göstermesi bakımından...

Gerek mevcut bilgileri depolarken başvurduğu sınıflandırma ölçütleri, gerekse hazır bilgilerden yeni sonuçlara varmada kullandığı çıkarım kuralları açısından, bütün sayısal bilgi işlem sistemleri formellik gereğine uymak zorundadırlar. Başka bir deyişle, herhangi bir objenin bir A sınıfına dahil olup olmadığının, daima üçüncü bir şıkka yer vermeyecek şekilde ve kesin olarak belirlenebilmesi gerekir. Bu yüzden herhangi bır sınıfa dahil olan objeler arasında, o sınıfa dahil olmak bakımından fark gözetilemez. Örneğin bir an için ASLAN kavramının formel bir sınıf olduğunu farzedelim. Bu takdirde, değişik aslanları, aslan olmaklıkları bakımından kıyaslayamaz, tipik olan ve olmayan aslanlardan söz edemeyiz.

Oysa, gündelik dilde kullandığımız kavramların çoğu bu tür yargılara olanak verir. Örneğin, ASLAN formel bir sınıf olarak yorumlandığı takdirde hiçbir anlam ifade etmeyen ''Bu aslan pek aslana benzemiyor'' cümlesi bizim için pekala anlaşabilir bir cümledir. Bu cümlenin bizim için anlaşılabilirliğini temin eden ise kavramlarımızın (gündelik dilimize kaynaklık eden sınıflandırmaların) yapısıdır.

Gündelik kavramların yapısı üzerinde sonra tekrar duracağız. Şimdilik şu kadarını söylemekle yetinelim: Formel kavramların aksine gündelik kavramlarımız homojen bir yapıya sahip değildirler, içerdikleri objelerin hepsi eşdeğer değildir. Çoğunlukla, prototipik bir örnek çevresinde örgütlenmiş objelerden oluşurlar.

Ateşi var mı? E
Faranjit
Kızıl
Menenjit

Boğazı ağrıyor mu? E
Faranjit

Öksürüyor mu? E
Faranjit

Başı ağrıyor mu? H
Nezle
Faranjit
Kızıl

Burnu akıyor mu? H
Grip
Faranjit
Kızıl
Menenjit

Lenflerde şişlik var mı? H
Nezle
Grip
Faranjit

Boynunda sertlik var mı? H
Nezle
Grip
Faranjit
Kızıl

Vücudunda lekeler var mı? H
Nezle
Grip
Faranjit
Menenjit

Puan

Nezle	4
Grip	5
Faranjit	8
Kızıl	4
Menenjit	3

Ateşi var mı? E
Faranjit
Kızıl
Menenjit

Boğazı ağrıyor mu? H
Nezle
Grip
Kızıl
Menenjit

Öksürüyor mu? H
Nezle
Grip
Kızıl
Menenjit

Başı ağrıyor mu? H
Nezle
Faranjit
Kızıl

Burnu akıyor mu? H
Grip
Faranjit
Kızıl
Menenjit

Lenflerde şişlik var mı? E
Kızıl
Menenjit

Boynunda sertlik var mı? H
Nezle
Grip
Faranjit
Kızıl

Vücudunda lekeler var mı? H
Nezle
Grip
Faranjit
Menenjit

Puan

Nezle	5
Grip	6
Faranjit	5
Kızıl	7
Menenjit	6

''Prototip'' kavramıyla ne kasdettiğimiz tekrar ''otomatik pratisyen''imizin çalışması örneği üzerinde anlatalım. Birinci uygulama örneğimizde, hastada faranjitin bütün semptomlarının birden görüldüğünü görmüştük. Bu yüzden bu hastanın prototipik bir faranjitten şikayetçi olduğunu söyleyebiliriz. Oysa, ikinci örnekte hastalık kızıla benzemekle birlikte, bütün semptomlar görülmemektedir. Bu yine de hastalığı kızıl saymamamız için yeterli bir neden değildir. Böyle bir durumda, gerçek bir doktorun ne yapacağını düşünelim. Büyük bir olasılıkla, hastanın ne kadar zamandır ateşi olduğunu soracak, başlangıç safhalarında bir kızılın mı sözkonusu olduğunu tespit etmeye çalışacaktır. Ama herhalükarda, şu anda görülen hastalık, prototipik olmamakla birlikte kızıl olan bir hastalıktır.

Bizim programımızın yaptığı, kendisi formel olarak tanımlanmış bir dizi karşılaştırmayla, gündelik kavramlarımızın heterojen ve sınırları belirsiz yapısını yansıtmaya çalışmak.

Otomatik Pratisyenin Kısıtlılıkları

Bilgilerin Temsil Edilişi ile İlgili Sorunlar

Kuşkusuz yazdığımız programın zeki, hele yaratıcı olduğu söylenemez. Programın kullandığı işlemlerde zekaya özgü olduğunu söyleyebileceğimiz tek bir işlem bile yok; sadece mümkün en basit karşılaştırmaları kullanmakla yetindik. Ancak, neden programımız için zeki denemeyeceğini, programın yaptıkları ile gerçek zekanın farklarının neler olduğunu doğru olarak tesbit etmemez gerekiyor.

Örneğin, programımızın sadece beş hastalık ve sekiz semptom tanıdığı, oysa en cahil pratisyen hekimin bilgi dağarcığında bile bundan kat kat daha fazla hastalık ve semptom bulunduğu söylenebilir. Ancak bu, bir bilgi işlem sistemi olarak programın esasına (mantığına) değil, elimizdeki fiziksel olanaklara (bellek hacmi, zaman vb.) bağlı bir kısıtlılıktır.

Programımıza yöneltilebilecek ikinci bir eleştiri ise bilgilere erişmekte kullandığı yöntem ve buna bağlı olarak da bilgilerin temsil ediliş tarzı ile ilgili bir eleştiri. Bu eleştirinin alabileceği biçimi bir örnek üzerinde gösterelim.

Herhangi bir pratisyen doktor, hastasının yüksek ateşi olduğunu öğrendiği takdirde, artık nezle olasılığını gözönünde bulundurmaya de-

```
 0 GOSUB 10000
100 X=1
110 PRINT "ATESI VAR MI?"
120 GOSUB 500
130 PRINT "OKSURUYOR MU?"
140 GOSUB 500
150 PRINT "BURNU AKIYOR MU?"
160 GOSUB 500
170 PRINT "BOYNUNDA SERTLIK VAR MI?"
180 GOSUB 500
190 PRINT "BOGAZI AGRIYOR MU?"
200 GOSUB 500
210 PRINT "BASI AGRIYOR MU?"
220 GOSUB 500
230 PRINT "LENFLERINDE SISLIK VAR MI?"
240 GOSUB 500
250 PRINT "VUCUDUNDA LEKELER VAR MI?"
260 GOSUB 500
270 PRINT:PRINT "PUAN"
280 PRINT
300 FOR N= 1 TO 5
310 PRINT AR$(N),PU(N)
320 NEXT N
400 PRINT
410 RUN
420 %UN
500 INPUT CEV$
510 M=M+1:CE=1:IF CEV$="H" THEN CE=0
520 FOR N=X TO 5
530 IF SEM(N,M)=CE THEN PRINT HAS$(N):PU(N)=PU(N)+1
540 IF N=1 AND M=1 AND CE=1 THEN X=2
550 NEXT N
560 RETURN
10000 DIM HAS$(5), SEM(5,8),PU(5)
11000 DATA NEZLE, 0,0,1,0,0,0,0,0
12000 DATA GRIP,1,0,0,0,0,1,0,0
13000 DATA FARANJIT,1,1,0,0,1,0,0,0
14000 DATA KIZIL,1,0,0,0,0,0,1,1
15000 DATA MENENJIT,1,0,0,1,0,1,1,0
16000 FOR N=1 TO 5
17000 READ HAS$(N)
18000 FOR M=1 TO 8
19000 READ SEM(N,M)
20000 NEXT M,N:N=0
21000 RETURN
```

vam etmez. Başka bir deyişle, gerçek bir pratisyen doktor, ateşi olduğunu bildiği bir hastanın semptomlarını nezlenin semptomları ile karşılaştırmaya devam etmez. Oysa bizim programımız tam anlamıyla mekanik bir biçimde davranmakta, yaptığı karşılaştırmaların hiçbiri daha sonra yapacağı karşılaştırmaları etkilememektedir.

Kuşkusuz bu sorun çözümsüz değildir, ancak böyle bir programda bulacağımız çözümün kendisi de mekanik olacaktır. Örneğin mümkün çözümlerden biri 520. satırda başlayan döngünün sayacını bir sabit değil bir değişken olarak tanımlamaktır. Bu değişken 1'e eşitlenir (satır 100) ama daha sonra eğer hastanın ateşi olduğu tesbit edilirse, bu değişken 2'ye eşitlenir ve böylelikle bundan sonraki karşılaştırmaların N = 2'den başlaması, yani nezlenin gözden geçirilmemesi sonucu sağlanmış olur.

Ancak böyle bir çözümün kendisi de çok ciddi yeni sorunlar yaratır. Gerçek bir pratisyen doktorun, artık nezleyi gözönünde bulundurmamaya karar vermesi, herhalde işini kolaylaştırma ve harcadığı zihinsel çabayı azaltma anlamına geliyor. Oysa bizim programımızda tek bir semptoma bağlı olarak bile karşılaştırmaları azaltma kararımız, program yapısının hatırı sayılır bir biçimde karmaşıklaşmasına yolaçıyor. Bellekte sakladığımız bilgiler 5 × 8'lik bir dizeyi aştığında ve yine bazı karşılaştırmaların doğrulanmasına bağlı olarak başka bazı karşılaştırmalar önlenmek istendiği takdirde bu karmaşıklık kat kat artacaktır.

Sorun semptomların birbirlerinden bağımsız ölçütlermişcesine bir liste halinde saklanmış olmasından kaynaklanmaktadır. Başka bir deyişle, kullanmakta olduğumuz temsil biçimi, yalnızca bazı semptomlarla bazı hastalıkların birarada göründüğünü saptayabilmekte, onların arasındaki iç ilişkileri temsil edememektedir. Oysa pratisyen hekim bir hasta ile karşılaştığında, ona tanıdığı bütün semptomların kendisinde bulunup bulunmadığını sormaz; çünkü tanıdığı semptomlar belleğinde belirli bazı mantıksal ilişkiler içerisinde depolanmıştır. Beş hastalık ve sekiz semptomdan oluşan evrenimiz hakkındaki bilgilerimizin bir kısmı sayfa 1704'teki gibi hiyerarşik bir şekilde temsil edilmiş olsaydı, nezleyle ilgili sorunumuz ç özülmüş olabilirdi:

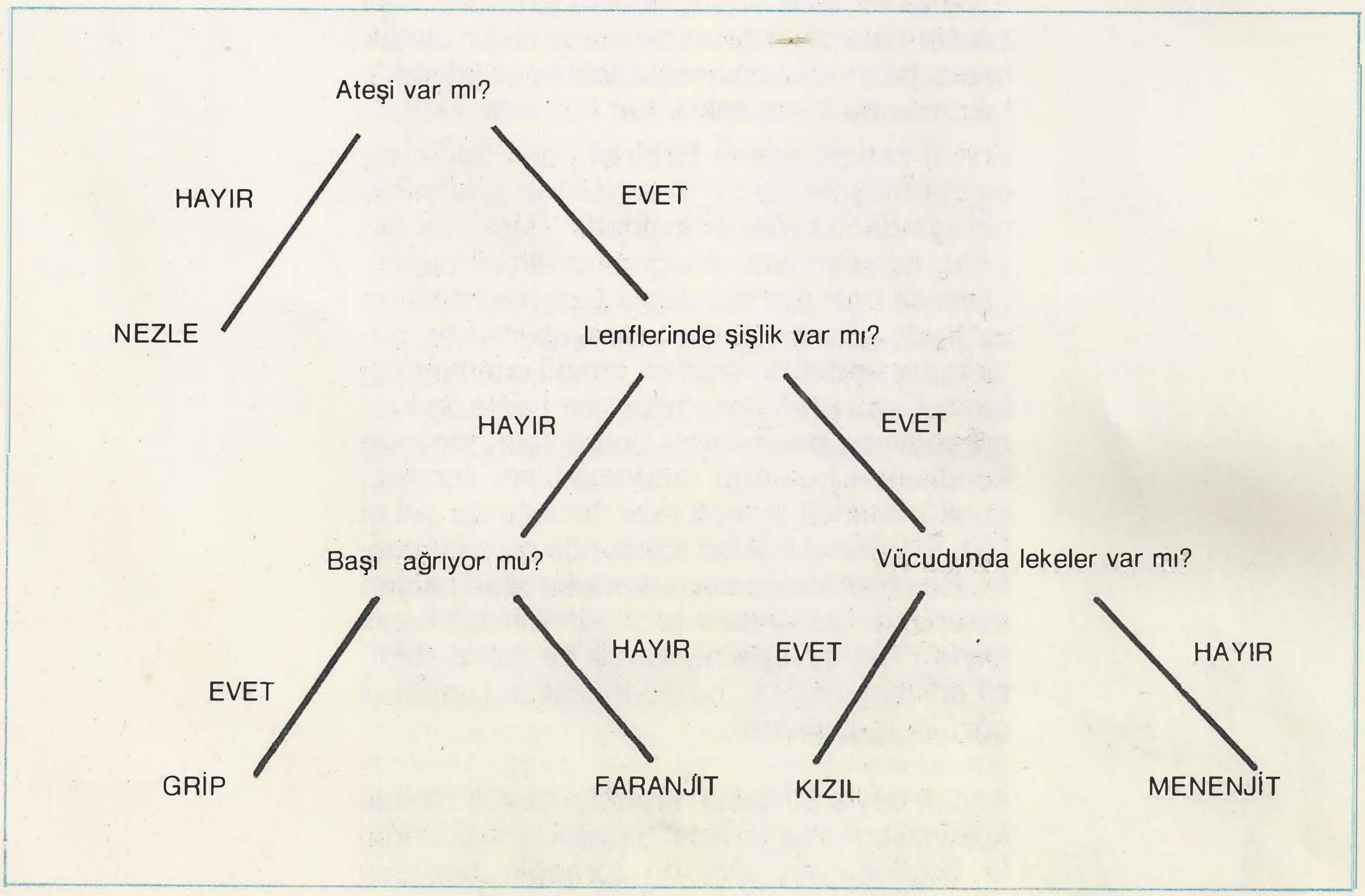
Ancak böyle bir tablo, aramayı büyük ölçüde kolaylaştırmakla birlikte, hastalıklar hakkındaki bilgilerimizin tümünü (örneğin nezlenin

semptomlarından birinin hastanın burnunun akması olduğunu, faranjitinkilerden birinin ise öksürük olduğunu) temsil edemez.

Bu tür, mevcut bilgilerimizin temsili ile ilgili sorunlar araştırmacıları, özellikle yapay zeka araştırmaları için tasarlanmış yeni bilgisayar dilleri geliştirmeye yöneltmiştir. LISP ve PROLOG gibi bu dillerin özelliklerini ve bu problemin çözümünde hangi ölçüde başarılı olduklarını daha ilerde inceleyeceğiz.

Öğrenme Sorunu

Yazmış olduğumuz otomatik pratisyen programının zeki bir program olarak kabul edilmesinin önünde duran ikinci büyük engel, programımızın hiçbir şey öğrenememesi. Gerçi biz problemi bir teşhis problemi olarak sınırlamış, bilgilerin bir zamanlar öğrenilmiş olduğunu veri kabul etmiştik. Yine de gerçek bir pratisyen doktor, bir teşhis sürecinde yeni hiçbir hastalık ya da semptom keşfetmese, yani mevcut bilgi dağarcığına birşey eklemese bile, tecrübe kazanır. Daha önce yapmış olduğu teşhislerle daha sonrakiler arasında sırf bu tecrübe sayesinde bir güvenirlilik farkı doğar. Öyle ki, bir deneyimin sonuçlarının daha sonraki deneyimler (bizim örneğimizde, teşhisler) için bir girdi oluşturmasının, zekanın ayırdedici özelliği olduğu söylenebilir.



Otomatik Pratisyenin Öğrenmesi

O halde, teşhis yapan bilgisayarımızın gerçekten düşünebildiği iddiasına biraz olsun inandırıcılık kazandırabilemk için programımızı her muayeneden, daha sonra yararlanabileceği sonuçlar çıkarmasını sağlayacak şekilde geliştirmemiz gerekir.

Bu amaçla kendi yanlışlarından öğrenen ve bu temelden hareketle hangi semptomların hangi hastalığa ait olduğuna karar vermesini sağlayacak bir ''karar kuralı'' geliştiren bir program yazacağız. Başka bir deyişle, program, çalışmaya başladığı noktada hangi semptomların hangi hastalıklarda görüldüğünü bilmiyor olacak; ancak çeşitli teşhislerden sonra bunu öğrenecek. Tabii bunun için programa hangi teşhislerinin doğru, hangilerinin yanılmış olduğunu bizim bildirmemiz gerekiyor.

Başlangıçta bir önceki örneğimizdeki 10000 alt-programına çok benzer bir alt program kullanarak hastalık adlarını ve semptomları bilgisayara veriyoruz. Ancak bu sefer, semptom adlarının doğrudan doğruya bir dizeye okuyarak (bkz. s.1709, 10020-10040 satırları arasındaki döngü), sekiz ayrı PRINT komutu yazmaktan kurtuluyoruz. Ayrıca semptomlarla hastalık adlarının eşleştirilmesini okuma aşamasında sağlamıyoruz. Çünkü bu sefer istediğimiz, bir önceki örnekte bilgisayara doğrudan doğruya girdiğimiz, bu sayfada yeralan tabloyu, programın kendisinin oluşturması.

SEM(.) dizeyi ise SEM$(.) dizeyindeki semptomların, belli bir hastada görülüp görülmediğini kodlayan değerleri içeriyor. (SEM(.)'in işlevi bu bakımdan, bir önceki örneğimizdeki CEV$ cevabını kodlayan CE değişkeninin işlevine benziyor). KU(.) dizeyi ise her semptom için bir karar kuralı içerecek. Ancak bu kuralın nasıl kullanıldığını görmek için, önce başka bir değişkenden, hastalık kodundan (KOD) sözetmemiz gerekiyor.

Kullandığımız mantık içersinde, her hastalık, semptomların belli bir dağılımı tarafından nitelendirilmektedir. KOD değişkeninin işlevi, mümkün her dağılımı belli bir hastalık adıyla eşleştirmek üzere, her dağılıma bir sayısal değer vermektir. Bunu yapmanın en görünür ve kolay yolu ise, sekiz semptomdan hangilerinin görüldüğünün (yani, SEM(.) değişkeninin hangi durumlarda 1 değeri aldığının) tesbit edilmesi ve 1 değeri alan SEM(.)'lerin KOD'da toplanmasıdır. Yani;

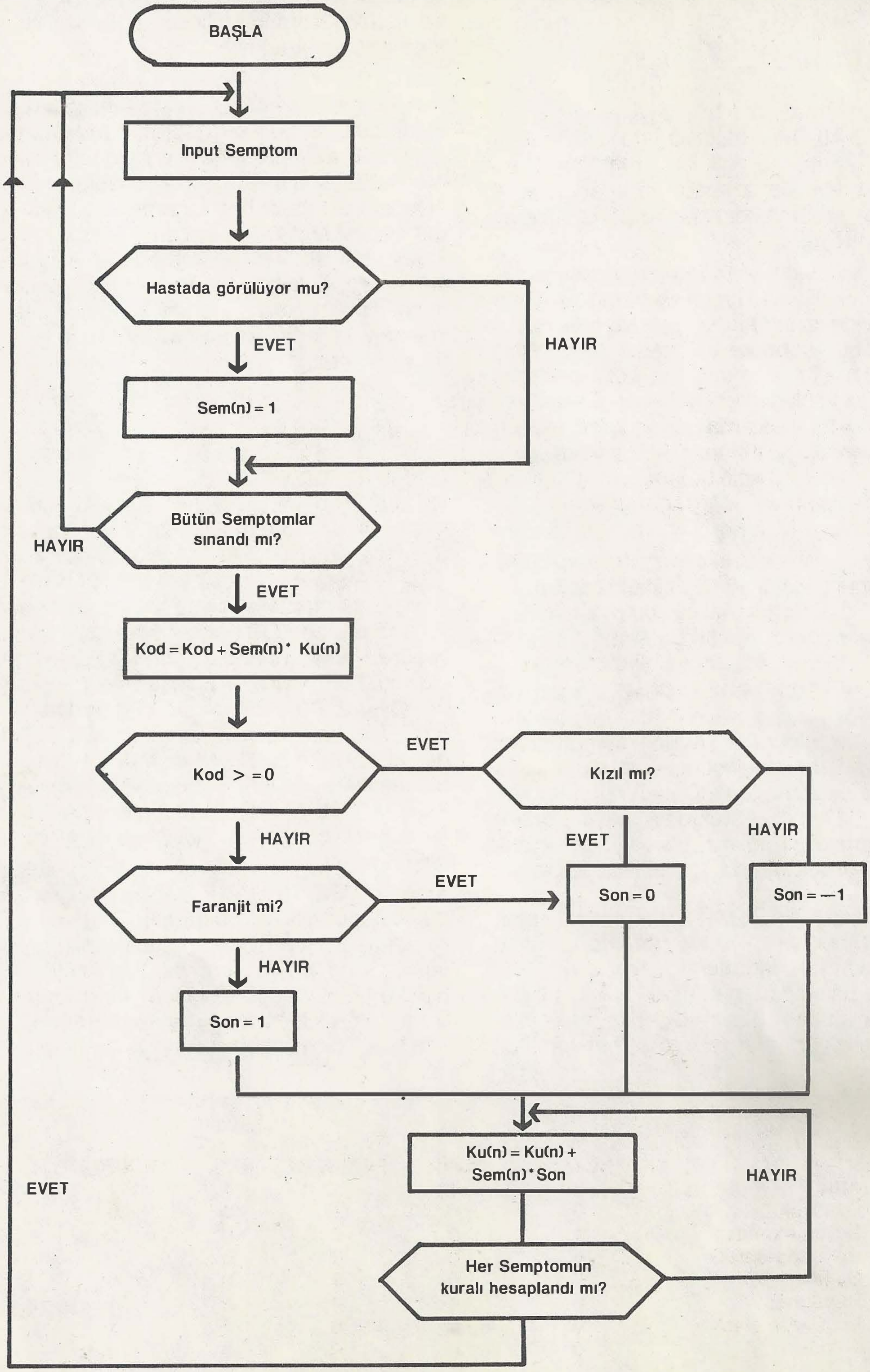
KOD = KOD + SEM(.)

NEZLE örneğini ele alalım. Nezlesi olan bir hasta muayeneye geldiğinde, onda yanlızca SEM$(.) = BURUN AKMASI değişkeni için SEM(.) 1 değerini alacaktır (yani SEM(3) = 1). Buna bağlı olarak da nezle hastalığının tanıtımı olacak olan KOD değeri, yukardaki formüle göre 1 değerini alacak, bundan böyle program NEZLE'yi KOD = 1 değeri ile tanıyacaktır. Oysa GRİP sözkonusu olduğunda hem ATEŞ için hem de BAŞAĞRISI için SEM(.) değişkeni (yani SEM(1) ve SEM(5)) 1 değerini alacağından KOD = 2 olacaktır. Böylelikle KOD'un 1 ya da 2 olmasına bağlı olarak, hastanın nezle mi, grip mi olduğuna karar verilecektir.

Ama FARANJİT'le KIZIL'ı karşılaştırdığımızda, bu çözümün yetersiz olduğunun farkına varırız. Çünkü yukarıdaki formülle yetinildiği takdirde, hem KIZIL hem de FARANJİT için KOD'un aldığı değer 3 olacak ve dolayısıyla bu iki hastalık birbirinden ayırdedilemiyecekir. İşte KU(.) dizeyini bu sorunu aşmak için dev-

	NEZLE	GRİP	FARANJİT	KIZIL	MENENJİT
Ateş	0	1	1	1	1
Öksürük	0	0	1	0	0
Burun akması	1	0	0	0	0
Boyunda sertlik	0	0	0	0	1
Boğaz	0	0	1	0	0
Başağrısı	0	1	0	0	1
Lenflerde şişlik	0	0	0	1	1
Lekeler	0	0	0	1	0

İKİ HASTALIĞI AYIRDETMEYİ ÖĞRENEN PROGRAM İÇİN AKIŞ ŞEMASI



reye sokuyor ve formülümüzü şu şekilde uyarlıyoruz:

KOD = KOD· + SEM(.)'KU(.)

Bu formülde KU(.)'nun alacağı değerin işlevi, iki hastalık arasında ayırdedici olan semptomların ayıklanmasını sağlamak olacak. Ancak programımızın öğrenme yeteneğine sahip olmasını istediğimizden, bu dizeyin değerlerini kendimizin belirlelemesi sözkonusu olamaz. Programın her tek uygulamada çıkardığı sonuçlara göre, bu değerleri belirleyebilmesi gerekir. Bunun için de, her teşhis deneyiminin sonuçlarının program tarafından kaydedilmesi gerekir. Programımızdaki SON değişkeni, verilen kararın doğru olup olmadığını tesbit etmek üzere kullanılacaktır.

Bu mantığı açıklamak için önce sadece iki hastalığın, FARANJİT ve KIZIL'ın birbirlerinden ayırdedilmeye çalışıldığı bir durumu inceleyelim (bkz. sayfa 1706'daki akış şeması ve sayfa 1708'deki döküm). Sadece iki hastalık sözkonusu olduğundan, KOD değişkeni ile ilgili tek bir karşılaştırma yetecektir (satır 180).

İlk muayenede KU(.) dizeyinin bütün değerleri 0 olduğundan, KOD = KOD + SEM(.)'KU(.) formülü (150. satır) daima 0 sonucunu verecek ve dolayısıyla ilk muayenede elde edilen HASTALIK KODU, zorunlu olarak 0 olacaktır. Bu ise programın, hasta hangi semptomlarla gelmiş olursa olsun, KIZIL MI? sorusunu sorması anlamına gelecektir.

Eğer tesadüfen, gelen hasta gerçekten kızıldan şikayetçi ise, SON değişkeni 0 değerini alacak ve dolayısıyla KU(.) = KU(.) + SEM(N) * SON formülü de (250. satır) sonucunu verecek ve KU(.)'nun içerdiği değerler değişmeden kalacaktır. Bu ise programın hiçbirşey öğrenmediği anlamına geliyor. Yani program ancak hatalarından bir şey öğrenebilmektedir.

Bu öğrenmenin aldığı somut biçimi görebilmek için, ilk muayeneye gelen hastamızın şikayetinin faranjit olduğunu farzedelim. Bu takdirde, yine muayene sonucunda program KIZIL MI? sorusunu soracaktı. Ancak cevabın yanlış olduğu bildirildiğinde bu kez SON değişkeni —1 değerini alacak ve 250. satırdaki formülde, kızılda görülen semptomlara (yani SEM (N) = 1 durumuna) tekabül eden KU(N) değişkenleri de, —1 değerini alacaktı. Başka bir deyişle, KU(.) dizeyi, ATEŞ, ÖKSÜRÜK ve BOĞAZ değişkenlerini KIZIL'ın ayırdedici değişkenleri olarak kodlamış olacaktı.

Bundan sonra bir faranjitli hasta daha geldiği

İKİ HASTALIĞI AYIRDETMEYİ ÖĞRENEN PROGRAM

```
10 GOSUB 10000
60 FOR N= 1 TO M
70 SEM(N)=0
80 PRINT SEMO(N);" "
90 INPUT CEVO
110 IF CEVO="E" THEN SEM(N)=1
120 NEXT N
125 KOD=0
130 FOR N= 1 TO M
150 KOD=KOD+SEM(N)*KU(N)
160 NEXT N
170 PRINT:PRINT "HASTALIK KODU=";KOD
180 IF KOD>= 0 THEN PRINT "KIZIL MI";:INPUT CEVO:GOTO 240
190 IF KOD< 0 THEN PRINT "FARANJIT MI";:INPUT CEVO:GOTO 240
200 IF CEVO="E" THEN SON=0:GOTO 240
210 SON=-1:GOTO 240
220 IF CEVO="L" THEN SON=1:GOTO 240
230 SON=1
240 FOR N=1 TO M
250 KU(N)=KU(N)+SEM(N)*SON
260 PRINT KU(N)
270 NEXT N
280 PRINT
290 GOTO 60
10000 M=8
10010 DIM SEMO(M), SEM(M), KU(M)
10020 FOR N=1 TO M
10030 READ SEMO(N)
10040 NEXT N
11000 DATA ATES,OFKURUK,BURUN AKINTISI,BOYUN SERTLESMESI,EGUZAL,FARANJIT,LEKE
SIS, LEKELER
12000 RETURN
```

takdirde, 150. satırdaki KOD = KOD + SEM(N) 'KU(N) formülü bu değişkenler için —1 sonucunu verecek, bu yüzden de döngünün sonunda KOD değişkeni —1 değerini alacaktır. —3< 0 olduğundan (bkz. 180. satırdaki karşılaştırma) program bu kez doğru olarak FARANJIT MI? sorusunu soracaktır.

Ancak henüz bu noktada program ulaşabileceği yetkinlik düzeyine ulaşmış durumda değildir. Dördüncü muayenede tekrar kızılı olan bir hastanın geldiğini farzedelim. Bu muayenede 150. satırdaki formül, ATEŞ semptomu (yani SEM(1)) için —1, diğer bütün semptomlar içinse ya SEM(N) = 0 ya da KU(N) = 0 olduğundan 0 sonucunu verecektir. Dolayısıyla da döngünün sonunda KOD = -1 olacaktır. —1 < 0 olduğundan, program yine FARANJIT MI? sorusunu soracaktır. Bu süreci kendi terimlerimizle şu şekilde yorumlayabiliriz. İkinci muayenenin sonunda program üç adet ayrıcalıklı semptom tesbit etmiş durumdadır; ama daha bu ayrıcalıklı semptomların ayırdediciliğini öğrenmiş değildir. Herhangi bir muayenede yanlızca ayrıcalıklı semptomları saymakla (yani, KOD değişkeninde biriktirmekle) yetindiğinden, kızıl muayenesinde de, yanlızca kızılın faranjitle paylaştığı tek semptom olan ATEŞ'i kaydeder. Ancak sınama koşulu 1 > = 0 olduğundan bu ayrıcalıklı semptomun varlığı, yine FARANJIT MI sorusuna yolaçar. Şimdi bu muayenenin sonunda ne olduğunu görelim:

Faranjit teşhisinin yanlış olduğu bildirilince, SON = 1 olur (bkz. 190. ve 230. satırlar). Bunu üzerine 250. satırdaki KU(N) = KU(N) + SEM(N) * SON formülü N = 1 için (yani ATEŞ semptomu için) 0 sonucunu verir (KU(1) = —1 + 1 * 1). Oysa ikinci değişken olan ÖKSÜRÜK için, bu kez karar dizeyindeki (KU(.)) değeri 1 olur (KU(2) = 0 + 1 * 1). Bu hesaplar so-

BEŞ HASTALIĞI AYIRDETMEYİ ÖĞRENEN PROGRAM

```
10 GOSUB 10000
20 KOD=0
30 FOR N=1 TO I
40 KOD(N)=0
50 NEXT N
60 FOR N= 1 TO M
70 SEM(N)=0
80 PRINT SEM$(N); "   ";
90 INPUT CEV$
110 IF CEV$="E" THEN SEM(N)=1
120 NEXT N
125 KOD=0
130 FOR N= 1 TO 8
140 FOR J= 1 TO 5
150 KOD(J)=KOD(J)+SEM(N)*KU(N,J):PRINT KOD(J);
160 NEXT J,N
170 FOR N= 1 TO I
180 IF KOD(N)>=KOD THEN KOD=KOD(N):HAS=N
190 NEXT N
200 PRINT:PRINT HAS$(HAS);" MIYDI"; " ";
210 INPUT CEV$
220 IF CEV$="E" THEN 20
230 FOR N= 1 TO I
240 PRINT N, HAS$(N)
250 NEXT N
260 PRINT "PEKI HANGISIYDI":INPUT CEV
280 FOR N=1 TO I
290 PRINT KOD(N),HAS$(N)
300 IF KOD(N)>=KOD AND N<>CEV THEN FOR J=1 TO M:KU(J,N)=KU(J,N)-SEM(J):NEXT J
310 NEXT N
320 FOR N= 1 TO 8
330 KU(N,CEV)=KU(N,CEV)+SEM(N)
340 NEXT N
350 FOR N=1 TO I
355 PRINT HAS$(N),
360 FOR J= 1 TO M
370 PRINT KU(J,N) ;
380 NEXT J
390 PRINT:PRINT
400 NEXT N
410 GOTO 20
10000 M=8:I=5
10010 DIM SEM$(M), SEM(M), KU(M,I),HAS$(I),KOD(I)
10020 FOR N=1 TO M
10030 READ SEM$(N)
10040 NEXT N
10050 FOR N=1 TO I
10060 READ HAS$(N)
10070 NEXT N
11000 DATA ATES,OKSURUK,BURUN AKINTISI,BOYUN SERTLIGI,BOGAZ,BASAGRISI,LIMFLERDE
SIS, LEKELER
11110 DATA MEZLE,GRIP,FARANJIT,KIZIL,MENENJIT
12000 RETURN
```

nuna kadar yapıldığında, yanlızca kızılla faranjiti birbirinden ayıran dört değişkenin (ÖKSÜRÜK BOĞAZ, LENFLERDE ŞİŞLİK, LEKELER) 0'dan farklı değerler aldığı görülür. Bunlardan yanlızca kızılda görülenler +1, yanlızca farnajitte görülenler ise —1 değerini almıştır. Dolayısıyla bundan böyle KIZIL'ın hastalık kodu 2, faranjitinki ise —2 olacaktır. Sistem istikrar kazanmış, yani program dört semptom temelinde iki hastalığı birbirinde ayırdedecek kuralı, çeşitli muayene deneyimlerinden çıkardığı sonuçlara dayanarak geliştirmiştir.

Daha Gelişkin Bir Program

Ancak amacımız iki hastalığı değil beş hastalığı (ve ideal olarak çok daha fazla sayıda hastalığı) birbirinde ayırdetmeyi öğrenebilecek bir program geliştirmekti. Örneği geliştirme biçimimizden de görülebileceği gibi, iki hastalıkla beş hastalık arasında programın yapı ve mantığı açısından bir fark olmakla birlikte, beş hastalıkla n hastalık arasında böyle bir fark bulunması gerekmiyor. İkiden fazla hastalığı birbirinden ayırdedebilecek bir program geliştirdiğimiz takdirde, artık bunu elimizdeki bellek hacmına bağlı olarak istediğimiz kadar çok sayıda hastalığa genelleyebiliriz.

Önce 10000 alt-programında, hastalık adlarını saklayabileceğimiz yeni bir dizey tanımlamamız gerekiyor (HAS$(.)). Ayrıca, artık sadece iki hastalık sözkonusu olmadığına göre, KOD değişkeniyle ilgili tek bir karşılaştırmayla yetinemeyeceğimizden, hastalık kodlarını içerecek KOD(.) adlı bir dizey tanımlamamız ve yine aynı nedenden ötürü, KU(.) dizeyini, KU(8,5) şeklinde iki boyutlu olarak tanımlamamız gerekiyor.

Ana programdaki bazı değişiklikler (bkz. sayfa 1709'daki döküm) oldukça yüzeysel, gerekçesi ilk bakışta kavranacak değişiklikler. Örneğin, 20-50. satırlar arasındaki döngü her muayenede hastada görülen semptomlardan hareketle hesaplanacak kodların içerileceği KOD(.) dizeyini sıfırlıyor. 150. satırda ise formül, KU(.,.) dieyinin iki boyutlu olarak tanımlandığını hesaba katacak şekilde yeniden yazılmış.

Ana programdaki ilk önemli değişiklik 170-190. satırlar arasındaki döngünün varlığı. Dikkat edileceği gibi burada KOD adıyla hem bir dizey hem de bir değişken kullanılıyor. Bunun nedeni her hastalık için farklı bir sayısal değere ihtiyacımız olmasından. Bu sayısal değerler 130-160. satırlar arasındaki döngüde, KOD(.) dizeyinde biriktiriliyor. Yani her muayenede hastanın semptomları NEZLE'den başlamak üzere her hastalığın ayırdedici semptomları ile karşılaştırılıyor ve bu karşılaştırmadan hareketle her hastalığa bir kod veriliyor (130-160. satırlar arasındaki döngü). KOD değişkeninin işlevi ise en yüksek KOD(.)'u içermek. Örneğin GRİP'in kodu en yüksekse, KOD(2) = KOD oluyor. Dizeydeki sıralamada GRİP'in sayısı olan 2 ise HAS değişkenine aktarılıyor. Bu programın tahmini için temel oluşturuyor. 200. satırda HAS$(.) dizeyinde HAS'ıncı hastalık adının bilgiyazımı yapılıyor (bizim örneğimizde HAS$(2)) ve teşhisin doğru olup olmadığı soruluyor. Eğer teşhis doğruysa başa dönülüyor (programın yalnızca hatalarından öğrendiğini hatırlayın). Programın bundan sonraki formüllerle, karar kurallarını geliştirmesinin ardında yatan mantık sayfa 1711-1714 arasında bir uygulama örneği üzerinde gösterilmiştir.

Öğrenme Programı Zeki mi?

Öğrenmenin zekanın ayırdedici özelliklerinden biri olduğunu söylemiştik. Geliştirdiğimiz program çeşitli denemeler sonucunda beş hastalığı birbirinden ayırdedemez bir durumdan, ayırdedebilir bir duruma geliyor. Bu süreci bir öğrenme süreci diye adlandırabilir ve buradan hareketle bu programla çalışan bir makinenin, bir insanınki kadar güçlü değilse de, nitelik bakımından insanınkinden farksız bir zekaya sahip olduğunu söyleyebilir miyiz?

Bu soruya olumlu bir cevap vermenin güçlüğü, kısmen programın geçirdiği bu süreci yorumlamanın, tarif etmenin güçlüğünden ileri geliyor. Bu, özellikle de programın sonunda geliştirdiği ve doğru teşhis yapmasını mümkün kılan karar verme kuralları tablosu için geçerli.

Kuşkusuz, tabloyu yaptığı iş bakımından tarif edebiliyoruz: Beş hastalığın birbirinden ayırdedilmesini mümkün kılıyor. Ancak bu tabloda temsil edilen bilgiyi, bir pratisyen hekimin hastalıklar hakkındaki bilgisini normal olarak ifade edeceği gündelik dile çevirmek olanaksız.

Sadece iki hastalığı birbirinden ayırdetmek için geliştirdiğimiz programda, bu, böylesine ciddi bir sorun olarak ortaya çıkmamıştı. çünkü, hernekadar her iki programın yaptığı işlemler birbirlerine benziyorsa da, birinci programın karar kuralları ile yaptığı semptomlar sınıflandırmasını, ''ayrıcalıklı semptomlar'',

```
ATES                        E
OKSURUK                     E
BURUN AKINTISI              H
BOYUN SERTLIGI              H
BOGAZ                       E
BASAGRISI                   H
LENFLERDE SIS               H
LEKELER                     H
MENENJIT MI?                H
PEKI HANGISIYDI?

1 NEZLE
2 GRIP
3 FARANJIT
4 KIZIL
5 MENENJIT
?                           3

0 NEZLE
0 GRIP
0 FRANJIT
0 KIZIL
0 MENENJIT

NEZLE     -1 -1  0  0 -1  0  0  0
GRIP      -1 -1  0  0 -1  0  0  0
FRANJIT    1  1  0  0  1  0  0  0
KIZIL     -1 -1  0  0 -1  0  0  0
MENENJIT  -1 -1  0  0 -1  0  0  0
```

KU(.) dizeyi henüz boş olduğundan, bu noktada KOD(.) dizeyinin bütün değerleri 0'a eşittir. Bu yüzden KOD, sonuncu KOD satırı olan KOD(5)'e HAS ise 5'e eşitlenir ve program MENENJIT MI sorusunu sorar.

Görüldüğü gibi, program yalnızca hastada görülen semptomlara ilişkin KU(.) değerleri üzerinde işlem yapıyor. Doğru olduğu bildirilen hastalık için bu semptoma tekabül eden KU(.) değerleri 1 artırılırken, diğer hastalıklarda 1 eksiltiliyor.

```
ATES                        E
OKSURUK                     H
BURUN AKINTISI              H
BOYUN SERTLIGI              H
BOGAZ                       H
BASAGRISI                   H
LENFLERDE SIS               E
LEKELER                     E
FARANJIT MI?                H
PEKI HANGISIYDI?
1 NEZLE
2 GRIP
3 FARANJIT
4 KIZIL
5 MENENJIT
?                           4

NEZLE
GRIP      -1 -1  0  0 -1  0  0  0
FARANJIT   0  1  0  0  1  0  0  0
KIZIL      0 -1  0  0 -1  0  1  1
MENENJIT  -1 -1  0  0 -1  0  0  0
```

Bir önceki muayenede KU(.)'nun yalnızca FARANJIT satırında 0'dan büyük değerler yer aldığından, azami KOD değeri KOD(3)'e eşit oldu.

Yine aynı formüllerin uygulanmasıyla, bu kez KIZIL'a özgü olan semptomlar, kızıl için 1 artırılmış, diğerleri içinse 1 eksiltilmiştir. Bu yüzden KIZIL'ın FARANJIT'le paylaştığı semptom 0'a eşitlenmiştir.

ATES
OKSURUK
BURUN AKINTISI
BOYUN SERTLIGI
BOGAZ
BASAGRISI
LENFLERDE SIS
LEKELER
KIZIL MI
PEKI HANGISIYDI?

1 NEZLE
2 GRIP
3 FARANJIT
4 KIZIL
5 MENENJIT
?

E H H H E H E H H H

2

—1 NEZLE
—1 GRIP
  0 FARANJIT
  0 KIZIL
—1 MENENJIT

NEZLE	—1	—1	0	0	—1	0	0	0
GRIP	0	—1	0	0	—1	1	0	0
FARANJIT	—1	1	0	0	1	—1	—1	—1
MENENJIT	—1	—1	0	0	—1	0	0	0

Görüldüğü gibi bu noktada KOD(.) dizeyinin aldığı değerler henüz ayırdedici değildir.

Bu noktadan itibaren KU(.) dizeyinin içerdiği değerler yalnızca matematiksel bir anlam kazanmaya başlamış, yorumlanamaz hale gelmiştir.

ATES
OKSURUK
BURUN AKINTISI
BOYUN SERTLIGI
BOGAZ
BASAGRISI
LENFLERDE SIS
LEKELER
GRIP MI?
PEKI HANGISIYDI?

E H H E H E H E H H

1

1— NEZLE
2 GRIP
3 FARANJIT
4 KIZIL
5 MENENJIT
?

5

—1 NEZLE
  1 GRIP
—3 FARANJIT
—1 KIZIL
—1 MENENJIT

*** Zaten yalnızca formülün işleyişini örneklemek amacıyla vermiş olduğumuz KU(.) dizeyinin değerlerini, yorumlanamaz hale geldiklerinden artık vermiyoruz. Ancak bir sonraki muayenede vereceğiz.**

```
ATES
OKSURUK
BURUN AKINTISI
BOYUN SERTLIGI
BOYUN SERTLIGI
BOGAZ
BASAGRISI
LENFLERDE SIS
LEKELER
MENENJIT MI?
PEKI HANGISIYDI?

1 NEZLE
2 GRIP
3 FARANJIT
4 KIZIL
5 MENENJIT
? 1

0 NEZLE
0 GRIP
0 FARANJIT
0 KIZIL
0 MENENJIT
```

NEZLE	−1	−1	1	0	−1	0	0	0
GRIP	−1	−1	−1	−1	−1	0	−1	0
FARANJIT	−1	1	−1	0	1	−1	−1	−1
KIZIL	−1	−1	−1	0	−1	0	1	1
MENENJIT	0	−1	−1	1	−1	1	1	0

Bu noktada KOD(.) ve KU(.) dizeylerinin karşılaştırılması, çok ilginç bir matematiksel "kazaya" işaret etmektedir. Her ne kadar KU(.) dizeylndeki değerlerin dağılımı, bütün hastalıkları birbirinden ayırdetmekle birlikte, KOD(.) dizeylnin bütün değişkenleri 0 değerini almaktadır. Bu sadece 150. satırdaki formülün, beklenmedik "tesadüfi" bir sonucudur.

```
ATES
OKSURUK
BURUN AKINTISI
BOYUN SERTLIGI
BOGAZ
BASAGRISI
LENFLERDE SIS
LEKELER
NEZLE MI?
```

Bir önceki muayenede ortaya çıkan matematiksel "tuhaf"lığı telafi eden program ilk doğru teşhisini yapıyor.

```
ATES
OKSURUK
BURUN AKINTISI
BOYUN SERTLIGI
BOGAZ
BASAGRISI
LENFLERDE SIS
LEKELER
FARANJIT MI?
```

İkinci doğru teşhis. Ancak, bir sonraki örnekten de anlaşılacağı gibi, öğrenme sürecini bu noktada kesmek hatalı olur.

ATES
OKSURUK
BURUN AKINTISI
BOYUN SERTLIGI
BOGAZ
BASAGRISI
LENFLERDE SIS
LEKELER
MENENJIT MI?
PEKI HANGISIYDI?

1 NEZLE
2 GRIP
3 FARANJIT
4 KIZIL
5 MENENJIT

—1 NEZLE
—1 GRIP
—2 FARANJIT
—2 KIZIL
1 MENENJIT

Bu, KU(.) dizeyinin alacağı nihai biçimdir. Ancak bu noktaya kadar örnek olarak, semptomları tanımladığımız hastalıklara tam tamına uyan, yani prototipik hastalıklar kullandığımıza dikkatinizi çekelim. Prototipik olmayan hastaların varlığı, böyle bir programın öğrenme sürecini çok daha uzatır, hatta nihai bir sonuca ulaşmasını olanaksız bile kılabilir.

NEZLE	—1	—1	1	0	—1	0	0	0
GRIP	—0	—1	—1	—1	—1	1	1	0
FARANJIT	—1	1	—1	0	1	—1	—1	—1
KIZIL	—1	—1	—1	0	—1	—1	1	0
MENENJIT	—1	—1	—1	1	—1	0	1	0

''ayırdedici semptomlar'' gibi bizim de anlamlandırabileceğimiz terimler çerçevesinde yorumlamak mümkündü.

Ancak, hastalık sayısı arttığında programın oluşturduğu soyut ve matematiksel düzlem artık gündelik dil tarafından anlamlandırılabilme yeteneğini yitiyor. Bu düzlemde program her hastalığa karşılık gelmek üzere ve birbirlerinden yanlızca biçimsel düzeyde ayrılan bir dizi semptom dağılımı tanımlıyor. Somut bir örnek verelim: Bu sayfadaki tabloda KU(2,1), yani GRİP hastalığı ve ATEŞ semptomuna karşılık gelen değişkenin değerinin 0 olduğunu görüyoruz. Bu değerin bütün işlevi, GRİP hastalığının tablonun bütününde diğer hastalıklardan ayırdedilmesine katkıda bulunmaktır. Yani, kendi başına, grip hakkında bildiklerimizin herhangi bir yönünü yansıtmamaktadır. Başka bir deyişle, bu değer, ancak programın kullandığı diğer matematiksel işlemler (örneğin 150.

satırdaki formül) ve tablonun bütünü bağlamında anlam kazanmaktadır.

Şöyle de söylenebilir: Bizim grip hakkındaki en basit bilgilerimiz bile, örneğin gripte hastanın ateşinin yükseldiği, bu programın hiçbir yerinde temsil edilmemektedir. Program yanlızca bizim bu bilgilerimiz ile tutarlı sonuçlar üretmektedir.

Önermeler Mantığının Sınırları

Sorunun fazlasıyla matematiksel (hesaba dayalı) bir yol izlemiş olmamızdan kaynaklandığı düşünülebilir. Gerçekten de herhangi bir pratisyen hekim ya da tıp öğrencisinin benzer bir durumda bizim programda kullandığımız türden formüller kullanacağını düşünmek abes olurdu.

Bir muayene sırasında bir hekimin düşünce tarzının, yaklaşık olarak şöyle bir seyir izleyeceğini farzedebiliriz: ''Hastanın ateşi varsa belki grip olabilir'' ya da ''Hastanın ateşi varsa ya griptir ya kızıldır, ya faranjittir ya menenjittir'' (evrenimizi beş hastalıkla sınırlı tutmaya devam ediyoruz). ''Hastanın hem ateşi var hem de boğazı ağrıyorsa, faranjittir'' ''Hastanın boğazı ağrımıyor'', ''O halde demekki hasta faranjit değil'' ilh.

Görüldüğü gibi bu düşünce tarzı matematiksel olmaktan ziyade mantıki bir akılyürütmeyi andırmaktadır. Dolayısıyla daha önce gözden geçirmiş olduğumuz Boole cebiri (ya da daha sınırlı olarak ''önermeler mantığı'') kuralları kullanılarak şu şekilde formelleştirilebilir:

IF P THEN Q OR R OR S OR T
IF O AND P THEN S
NOT O

NOT S

P = Hastanın ateşi var.
Q = Hasta Grip
R = Hasta Kızıl
S = Hasta Faranjit
T = Hasta Menenjit
O = Hastanın boğazı ağrıyor.

Bu formelleştirme, yukarda andığımız cümlelerde dile gelen bilgileri yansıtmak açısından hangi ölçüde yeterli? Bu soruyu yanıtlamaya çalışırken yine daha önce değinmiş olduğumuz çeşitli önermeler arasındaki içsel bağlantıların nasıl temsil edileceği sorusu ile karşı karşıya kalıyoruz. IF P THEN Q önermesi herhangi iki cümle arasındaki ilişkiyi, anlamlarından tamamen bağımsız olarak formelleştirmektedir. Örneğin, ''Türkiye'nin başkenti Ankaraysa, 1986 yılındayız'' cümlesi ile yukarda andığımız cümleler ya da ''Hastanın vücudunda lekeler varsa kızıldır'' cümlesi arasında, önermeler mantığındaki formalizasyonları açısından hiçbir fark yoktur.

Oysa, ''Hastanın ateşi varsa, ya grip ya kızıl ya faranjit ya menenjittir'' cümlesinin doğruluğunun kaynağı, grip, kızıl, faranjit ve menenjitin ateşli hastalıklar olduğunun bilinmesidir. Başka bir deyişle, bu cümle doğruluğunu büyük ölçüde öncüldeki önerme ile sonuç arasındaki anlamsal ilişkiden almaktadır. Cümleyi önermeler mantığı içersinde formelleştirdiğimizde ise bu anlamsal ilişkiyi tamamen yitirmiş oluyoruz.

Başka bir deyişle, önermeler mantığı ile ifade edilen kural, bizim istediğimizden çok daha genel ve dolayısıyla bir o kadar da içeriksizdir. Bu yüzden, pratisyen bir hekim gibi düşünmesini istediğimiz bir programda kuralları önermeler mantığı ile yazdığımız takdirde, başka ve büyük bir olasılıkla, zorlama ve *ad hoc* olacak yollarla, bu kuralların sınırlandırılıp özgülleştirilmesi gerekecektir.

Ancak önermeler mantığının daha da önemli bir sınırlılığı, bu mantıkla öğrenme sürecini programlamaya çalıştığımızda ortaya çıkacaktır.

Bu sorunların ne olabileceğini görmek için, bu konuyu öğrenmeye çalışan bir insanın düşünce tarzını yine yaklaşık olarak nitelemeye çalışalım:

''Hastanın ateşi varsa grip olabilir'' ''Hastanın ateşi var ama grip değil'', ''O halde demek ki, bütün ateşli olan hastalar grip değildir''.

Programın bir insanın öğrenme sürecini taklit edebilmesi için bu türden bir akılyürütme yapabilmesi şarttır. Şimdi bu akılyürütmeyi önermeler mantığının çerçevesi içersinde formelleştirmeye çalışalım:

IF P THEN Q
P AND NOT Q

Boole cebiri kuralları içersinde bu iki önermeni kesişimi (toplamı) bir çelişki (**contradiction**) oluşturur. Dolayısıyla yine aynı kuralların sınırları içersinde yapabileceğimiz tek şey, iki

önermeden birini değillemek. Hastanın semptomunu reddedemeyeceğimize göre, değilleyeceğimiz birinci (koşullu) önerme olacaktır:

NOT (IF P THEN Q)

Bu önermenin yorumu ise, olsa olsa ''Hastanın ateşi varsa grip olduğu doğru değildir'' ya da ''Ateşi olanlar grip olanlar değildir'' olabilir. Oysa bizim istediğimiz önermenin tümünü değil yüklemin (ateşi olmak) kendisi için geçerli olduğu kümenin (hastalar) evrenselliğini değillemek; yani önermenin kapsamını sınırlandırmak. Bu ise NOT işlemcisini önermenin tümü ile değil de, ancak bir bölümü ile ilişkilendirmekle mümkün. Yani, daha genel bir deyişle söyleyecek olursak, önermeler mantığı çerçevesi içersinde çözümlenemez bir birim olarak simgeleyegeldiğimiz önermelerin iç yapısını çözümleyebilmemiz gerekiyor.

Bize bu olanağı tanıyan formel sistem, genel olarak yüklemler mantığı (**predicate calculus**) olarak anılır. Buraya kadar saydığımız gerekçelerden ötürü, yapay zeka araştırmalarında kullanılan bilgisayar dillerinin çoğu, yüklemler mantığından türemiş ya da esinlenmişlerdir.

Yüklemler Mantığı

Yüklemler mantığı çerçevesi içersinde her cümle bir (ya da bir dizi) yüklem (**predicate**) ve bu yüklemlerin tümleç ya da argümanları şeklinde simgelenir:

Deniz erkektir:

ERKEK(Deniz)

Deniz kadındır:

KADIN(Deniz)

A ile B'nin çarpımı C'dir:

ÇARPIM(A,B,C)

Bandırma bir il merkezi değildir:

NOT İLMERKEZİ(Bandırma)

Bu notasyona çevrilmiş bir önermeye tümce (**clause**) diyeceğiz. Argümanlarıyla birlikte yükleme ise kök (**literal**) denir. Son örnek bir mantık işlemcisi ile bir kökten oluşuyor. Bu örnekten de anlaşılacağı gibi kökler (ve tümceler üzerinde) tanıdığımız bütün mantık işlemcilerini uygulamak mümkündür.

Deniz ya erkektir ya kadındır.
ERKEK(Deniz) OR KADIN(Deniz)

Deniz kadındır ve evli değildir.
KADIN(Deniz) AND NOT EVLİ(Deniz)

Ancak çoğunlukla alt alta yazılmış kök ve tümcelerin AND bağlantısı ile bağlı olduğu varsayıldığından, AND kelimesi yazılmıyabilir. Dolayısıyla son örnek şu şekilde de yazılabilir:

KADIN(Deniz)
NOT EVLİ(Deniz)

Nitelikler ve İlişkiler

Yüklemler mantığında önermelerin yüklemler ve argümanlar (ya da tümleçler) şeklinde ifade edildiğini gördük. Yüklemler sahip oldukları argüman sayısına göre sınıflandırılabilir. Bir argümanı olan bir yüklem, bir konumlu yüklem **(one place predicate)** ya da birinci dereceden bir yüklem **(first degree predicate),** iki argümanı olan ikinci dereceden, n argümanı olan ise n. dereceden yüklem olarak anılır.

Yüklem, argümanlarının bir niteliğini ya da argümanlar arasındaki bir ilişkiyi betimler. Genellikle bir konumlu yüklemler argümanlarının bir niteliğini, birden çok konumlu olan yüklemler ise argümanları arasındaki bir ilişkiyi betimler. Örneğin, şu iki cümleyi ele alalım: ''Ahmet evlidir'' ve ''Ahmet Ayşe'yle evlidir''. Bunlardan birincisini EVLİ(Ahmet), ikincisini ise EVLİ(Ahmet Ayşe) diye gösterebiliriz. Çünkü birincisi Ahmet'i bir özelliğini, ikincisi ise Ahmet'le Ayşe arasındaki bir ilişkiyi betimlemektedir.

Bu noktada, yüklemler mantığının çok önemli bir genel kuralını belirtelim. Herhangi bir dizide, bir yüklem daima aynı dereceden olarak kullanılmak zorundadır. Bir kere n. dereceden olarak kullanılan bir yüklem, daha sonra da aynı dizi içersinde daima n. dereceden kullanılmak zorundadır. Bu yüzden, örneğin ''Ahmet Ayşeyle evliyse Ahmet evlidir'' gibi bir cümleyi, yüklemler mantığına şu şekilde çeviremeyiz:

IF EVLİ(Ahmet, Ayşe) THEN EVLİ(Ahmet)

Çünkü burada aynı yüklem (EVLİ) önce ikinci dereceden, sonra birinci dereceden olarak kullanılmaktadır.

IF-THEN İşlemcisi

Pratisyen bir hekimin akılyürütme tarzını temsil etmeye çalışırken, sık sık ''Ahmet'in ateşi varsa griptir'' türünden koşullu cümlelere başvurduk. Böyle bir cümleyi yüklemler mantığı içersinde şöyle formüle edebiliriz.

IF ATEŞLİ(Ahmet) THEN GRİP(Ahmet)

Bu tümceyi şimdiye kadar kullanageldiğimiz AND, OR ve NOT işlemcileri ile yeniden yazabilmek, formel sistemimizin yalınlığı ve okunabilirliği açısından yararlı olacaktır. Ayrıca daha da önemlisi, bugün kullanılmakta olan 'zeki' sistemlerin birçoğu IF-THEN'le yazılmış tümceleri değil, diğer işlemcilerle yazılmış tümceleri kabul edebilmektedir.

Şöyle bir önerme ele alalım: ''Deniz ya kadındır ya da erkek''. Bu önermeden şu önermeyi çıkarsayabiliriz: ''Deniz kadın değilse erkektir.'' Yani, P OR Q doğru ise IF NOT P THEN Q da doğrudur. Bu kuralın genelgeçerliğini doğruluk tablosu ile sınayalım:

P	Q	IF P OR Q	NOT P	IF NOT P THEN Q
D	D	D	Y	D
D	Y	D	Y	D
Y	D	D	D	D
Y	Y	Y	D	Y

Benzer bir biçimde, IF P THEN Q tümcesi de NOT P OR Q'ya çevrilebilir. ''Yağmur yağıyorsa sokaklar ıslaktır'' önermesi ile ''Ya yağmur yağmıyor ya da sokaklar ıslaktır'' önermesi ile eşdeğerdir. Bunu da doğruluk değerleri tablosuyla sınayalım:

P	Q	IF P THEN Q	NOT P	NOT P OR Q
D	D	D	Y	D
D	Y	Y	Y	Y
Y	D	D	D	D
Y	Y	D	D	D

Ancak buraya kadar söylediklerimiz yanlızca yalın, yani öncülü bileşik olmaya bir önerme içeren koşullu önermeler için geçerlidir. Öncülün bileşik olduğu durumlar için örneğin ''yağmur yağıyorsa ve şemsiyen yoksa, ıslanırsın'' gibi bir önerme, buraya kadar anlattığımız çıkarım kuralları ile ''veya'' ile bağlanmış bir dizi önermeye çevrilemez.

Bunun için önce daha önce gözden geçirmiş olduğumuz de Morgan kurallarını hatırlamak gerekiyor. Bu kurallardan hatırlanacağı gibi, NOT (P AND Q) ile NOT P OR NOT Q eşdeğerdir. Şimdi ''Yağmur yağıyorsa ve şemsiyen yoksa, ıslanırsın'' gibi bir önermeyi şu şekilde gösterebiliriz:

IF (P AND Q) THEN R

Parantezler hem P hem de Q önermelerinin IF'in kapsamı içersinde olduğunu gösteriyor. Dolayısıyla yukarda gördüğümüz çıkarım kuralı gereğince bu tümceyi şu şekle çevirebiliriz:

NOT (P AND Q) OR R

De Morgan kuralını uygulayarak da,

NOT P OR NOT Q OR R

açılımını elde ederiz.

Bunu yine yüklemler mantığında ifade edilmş bir önerme örneği üzerinde uygulayalım: ''Ahmet'in ateşi varsa ve Ahmet öksürüyorsa, Ahmet faranjittir'' önermesini önce şu şekilde yazabiliriz.

IF ATEŞLİ(Ahmet) AND ÖKSÜRÜYOR(Ahmet)
THEN FARANJİT(Ahmet)

Yukardaki çıkarım kuralları gereğince ise bu, şu şekilde OR'larla bağlanmış bir dizi tümceye çevrilebilir:

NOT ATEŞLİ(Ahmet) OR NOT ÖKSÜRÜYOR
(Ahmet) OR FARANJIT(Ahmet)

''Ahmet ya ateşli değildir ya öksürmüyordur ya da faranjittir''.

Son olarak karmaşık AND ve OR önermeleri arasındaki ilişkiyi ifade eden çıkarım kurallarına ihtiyacımız var. Bu kurallara OR'un ve AND'in birbirleri üzerinde dağıtılabilirliği **(distributivity)** denir. Bu kurallara, örneğin şu türden önermeleri formelleştirebilmek için ihtiyacımız var: ''Eğer Deniz anneanneyse hem çocuğu hem de torunu var demektir''. Bu cümleyi önce (yaklaşık olarak) şu şekilde yazabiliriz:

IF ANNEANNE(Deniz) THEN ÇOCUKLU
(Deniz) AND TORUNLU(Deniz)

IF'i OR cinsinden ifade den kural gereğince, yukardaki tümce şu şekli alır. (Önermeler man-

tığında, yukardaki cümlenin IF P THEN Q AND R şeklinde yazılacağını hatırlayın).

NOT ANNEANNE(Deniz) OR (COCUKLU (Deniz) TORUNLU(Deniz)

Bu ise, OR'un AND üzerinde dağıtılabilirliği (distributivity) kuralı gereğince şu şekilde yazılabilir:

(NOT ANNEANNE (Deniz) OR ÇOCUKLU (Deniz) (AND(NOT ANNEANNE (Deniz) OR TORUNLU (Deniz))

Bu kuralın önermeler mantığında ifadesi şöyledir:

P OR (Q AND R) = (P OR Q) AND (P OR R)

Bu kuralın simetriği denebilecek kural ise, AND'in OR üzerinde dağıtılabilirliği kuralıdır. Kısaca bu kural şu şekilde ifade edilebilir: P AND (Q OR R) ile (P AND Q) OR (P AND R) eşdeğerdir.

Sabitler ve Değişkenler

Buraya kadar verdiğimiz kurallar, herhangi bir tekil cümleyi karmaşık bir dizi tümceler şeklinde ifade etmenize yetecektir. Ancak bizim pratisyen programızın ihtiyacı olan bilgiler, tekil değil genel ya da tümel önermeler şeklinde ifade edilen bilgilerdi. Örneğin bizim programımıza yüklemek istediğimiz bilgi ''Ahmet'in ateşi varsa Ahmet griptir'' önermesi değil, ''Hastanın ateş varsa, hasta griptir'' önermesiydi. İkinci önermenin birinci önermeden farkını şu şekilde ifade edebiliriz: İkinci cümlede herhangi bir insanın ateşi varsa, o insanın grip olabileceği ifade edilmektedir. Başka bir deyişle, cümlenin öznesi farklı değerler alabilecek (Ahmet, Mehmet, Ayşe vs. olabilecek) bir değişkendir. Dolayısıyla, bu cümleyi ifade edebilecek formel sistemin de değişkenler içermesi gerekmektedir.

''Ahmet'i ateşi varsa Ahmet griptir'' önermesini şu şekilde formelleştirmiştik:

NOT ATEŞLİ(Ahmet) OR GRİP(Ahmet)

''Hastanın ateşi varsa hasta griptir'' önermesini ise şu şekilde formelleştirebiliriz:

NOT ATEŞLİ(x) OR GRIP(x)

Burada x değişik değerler alabilen bir değişkeni göstermektedir.

Niceleyenler

Son örneğimiz ''Hastanın ateşi varsa griptir'' cümlesinin (ya da daha doğrusu ''Herkes ya ateşli değildir ya da griptir'') cümlesinin yüklemler mantığına çevirisidir. Dolayısıyla bu cümlede yeralan x değişkeni, ''bütün x'ler'' şeklinde yorumlanmaktadır.Öyle ki, formel dizinin gündelik dile kelimesi kelimesine uyan bir karşılığını verecek olursak, bunun, ''Bütün x'ler ya ateşli değildir ya da griptir'' şeklinde olması gerekir.

Bunu, mantık dilinde x'in tümel bir niceleyenin **(universal quantifier)** kapsamı **(scope)** içersinde olduğu şeklinde ifade ederiz.

Ancak, değişkenlere ihtiyaç duyduğumuz durumlar, sadece yukarıda örneklediğimiz ve ''bütün x'ler'' şeklinde yorumlanan durumlarla karşılaştırılabilecek durumlar değildir. Tümel niceleyenin kapsamı dışında kalan değişkenlere ihtiyacımız olduğu durumlara en basit örnek olarak şöyle bir cümleyi verebiliriz: ''Biri hasta olmuş''. Argüman yerinde bir değişken kullanıp, bu önermeyi şu şekilde yazdığımız takdirde,

HASTA(x)

Bu tümce ''Bütün x'ler için, x hastadır'' ya da ''Her x hastadır'' diye yorumlanacaktı. İşte, ''Herkes hastadır'' ile ''Biri hastadır'' arasındaki farkı ifade edebilmek için yüklemler mantığı içersinde tümel niceleyenin yanında, varoluşsal niceleyen **(existential quantifier)** adlı bir niceleyen tanımlanmıştır. Bu niceleyeni kullanarak, yukardaki tümceyi şöyle yeniden yazabiliriz:

(∃x)HASTA(X)

Bu tümcenin yorumu şöyledir: Hiç değilse öyle bir x vardır ki, o hastadır. Başka bir deyişle varoluşsal niceleyen, kapsamı altında olan değişkenin hiç değilse bir değerinin varolduğunu bildirir.

Buraya kadar söylediklerimiz, varoluşsal niceleyenin işlevini çok fazla açıklayamamış olabilir. Çünkü, ''Biri hastadır'' cümlesini, enin-de sonunda, çok fazla anlam kaybına uğramadan bir sabit kullanarak da yazabiliriz:

HASTA(biri)

Ama ''bazı'' gibi terimlerin de formelleştirilmesi gerektiği gözönünde bulundurulursa, varoluşsal niceleyenin kaçınılmazlığı daha iyi anlaşılır. Başka bir deyişle, varoluşsal niceleyenin kapsamı altındaki değişken tek bir değer aldığı sürece varoluşsal niceleyenle nicelenmiş bir değişkenle, bir sabit arasındaki fark çok önemli olmayabilir. Ama bir değişkenle belirsiz sayıdaki bireylere işaret etmek istiyorsak, o zaman bunu yüklemler mantığında varoluşsal niceleyen dışında bir yolla yapmamız mümkün değildir. Örneğin ''Bazı insanlar akıllıdır''ı yüklemler mantığında şu şekilde yazabiliriz:

∃(x)AKILLI(x)

Yorumu: Hiç değilse bir x için x akıllıdır.

Aslında varoluşsal niceleyeni, tümel niceleyeni ve NOT işlemcisini kullanarak tanımlayabiliriz. P herhangi bir yüklem olsun. O takdirde,

∃(x)P(x) = (tanım) NOT(x) NOT P(x)

Bu tanımı şöyle yorumlayabiliriz: Hiç değilse bir x için P'nin doğru olması demek, bütün x'ler için P'nin doğru olmamasının doğru olmadığı demektir.

Kapsam Sorunları

Gündelik dildeki cümleleri yüklemler mantığının dilindeki tümcelere çevirirken özellikle dikkat edilmesi gereken hususlardan biri, hangi değişkenin hangi niceleyenin kapsamı **(scope)** içersinde olduğudur.

Bu sorunun önemini şöyle bir örnekle gösterebiliriz: ''Her erkek bir kadını sever'' gibi bir cümle çift anlamlıdır; başka bir deyişle, iki ayrı önerme içermektedir. Bunlardan birincisine göre öyle bir kadın vardır ki (örneğin Brigitte Bardot ya da Marilyn Monroe) bütün erkekler tarafından sevilmektedir. Diğer yorumunda ise cümle, bütün erkeklerin sevdiği tek bir kadının varolduğunu iddia etmemekte, aksine her erkek için, o erkeğe tekabül eden ve o erkeğin sevdiği bir kadın olduğunu söylemektedir. Şöyle de söylenebilir: Birinci önerme ile ifade edilen bilgiyi bir bilgisayara yüklediğimiz takdirde bilgisayarın ''Kim?'' sorusunu sormasını isteyebiliriz. Oysa ikinci önerme böyle bir soruya izin vermemekte, ancak her tek erkek için o erkeğin sevdiği bir kadın bulunması gerektiği bilgisinin depolanmasını sağlamaktadır. Örneğin eş bulma servislerinde kullanılan bilgisayar programlarına temel bir aksiyom olarak böyle bir önermenin verilmesi gerektiğini düşünebiliriz.

Burada aynı cümleyle birbirinden tamamen farklı iki bilgi ifade edilebildiğine göre, yüklemler mantığını kullanarak bu iki bilgiyi birbirinden ayrıştırabilmemiz gerekir. Bu ayrıştırmayı yaparken de özellikle dikkat etmemiz gereken ''Her'' ve ''Bir'' gibi gündelik kelimelerle ifade edilen niceleyenlerin kapsamlarını doğru tayin edebilmektir. Cümlenin birinci anlamını şu şekilde formelleştirebiliriz(*):

∃(y)SEVER(x,y)

Tümcenin en solundaki niceleyen kapsamı en geniş olan niceleyendir. Dolayısıyla yukarıdaki tümcede ilk ifade edilen şey (hiç değilse) bir kadının varolduğudur. Oysa tümceyi şu şekilde yazsaydık:

(x)∃(y)SEVER(x,y)

''hiç değilse bir kadın'' diye yorumladığımız değişken ''bütün erkekler'' diye yorumlanan x değişkeninin kapsamı altına girmiş olurdu. Çünkü bu durumda tümcenin en solunda tümel bir niceleyene işaret eden x karakteri bulunmaktadır.**

Yani ''hiç değilse bir kadın'' şeklinde yorumlanan ∃(y) değişkeni ''bütün erkekler''in kapsamına alınmıştır. Dolayısıyla tümceyi şu şekilde yorumlayabiliriz: ''Bütün erkekler için hiç değilse bir kadın vardır öyle ki, her erkek o kadını sever.''

Kısaca özetlemek gerekirse, mantık işlemciler ve niceleyenlerin kapsamı daima en soldan başlayarak hesaplanır ve her niceleyenin kendi sağındaki bütün diğer niceleyen ve değişkenleri kapsamına aldığı söylenebilir.

(*) Bu örnekte x'in erkeklere, y'nin ise kadınlara işaret ettiğini varsayacağız. Normal bir bilgi işlem uygulamasında böyle bir varsayımda bulunamayız. x'in erkeklere ve y'nin kadınlara işaret ettiğini bilgisayara açık olarak bildirmemiz gerekir. Bunun nasıl yapılabileceğini bir sonraki bölümde göreceğiz.

Ancak biz kullandığımız notasyonda değişkenin tümel niceleyenin kapsamında olduğunu göstermek için yanlızca değişkeni gösteren harfi kullanıyoruz.

(**) Bazen tümel niceleyen ∀ harfi ile gösterilir. O takdirde yukarıdaki ifade şöyle yazılırdı:

∀(x)∃(y)SEVER(x,y)

Yüklemler mantığında kullanabileceğimiz terimler yüklem ya da argüman olabilir. Yüklemler, argümalarını niteliklerini ya da argümanları arasındaki ilişkileri betimleyen terimlerdir.

Argümanlar ise değişken veya sabit olabilir. Sabitler tek bir bireye işaret eder. Değişkenler ise farklı değerler alabilir, yanif arklı bireyler tarafından tatmin edilebilen argümanlardır. Dolayısıyla bir bireyler kümesine işaret ederler.

Aksiyomlar

Herhangi bir bilgi İşlem sistemine işlemeye başlamadan önce verdiğimiz bilgiler -olgular, tanımlar, mevcut nitelik ve ilişkiler- bütününe aksiyomlar (ya da sayıltılar **-assumptions-**) denir. Başka bir deyişle, aksiyomlar bilgisayarın kendisinin çıkarsamak zorunda almadan bir-diği bilgilerdir.

Dolayısıyla kullanmakta olduğumuz program 'zeki' bir program da olsa çalışmasının başında da bizim ona bildirdiklerimizin dışında hiçbirşey bilmez. Dahası bizim ona bildirdiklerimizin (aksiyomların) doğruluğunu, dış dünya ile uygunluğunu da sınama olanaklarından yoksundur. Kendisine yanlış bilgiler verildiği takdirde bunlardan türeteceği sonuçlar da yanlış olacaktır.

Aksiyomları kabaca iki gruba ayırmak mümkündür:

1 / Yüklemler arasındaki ilişkileri betimleyen genel önermeler.

2 / Belli bir münferit durumu betimleyen tekil önermeler.

Ancak özellikle birinci türden aksiyomları yazabilmek için önce yüklemler mantığında bireylerin nasıl temsil edildiği üzerinde biraz daha durmamız gerekiyor.

Bireyler

Bir önceki bölümde ''her erkek bir kadını sever'' cümlesini formelleştirirken kullandığımız x ve y değişkenlerinin otomatik olarak ''erkekler'' ve ''kadınlar'' olarak yorumlanacağını varsaydık. Ancak biz bunu bilgisayara bildirmedikçe bilgisayarın böyle br yorum yapması için hiçbir gerekçe yoktur.

Daha basit bir örnek verelim:

''Bütün insanlar akıllıdır'' ya da ''Bazı insanlar akıllıdır'' cümlelerini nasıl formelleştireceğiz? Şimdiye kadar kullandığımız yöntemle bu cümleleri şöyle yazabiliriz:

(x)AKILLI(x)
∃(x)AKILLI(x)

Ancak her iki örnekte de x'in ''insan'' olarak yorumlanması gerektiğine dair herhangi bir ibare yoktur.

Soruyu genel olarak şöyle ifade edebiliriz: Herhangi bir tümcede bir değişken kullandığımızda, bu değişkenin işaret ettiği bireyler kümesini nasıl sınırlandıracağız? Cevabımız kısaca: Gündelik dilde değişken argümanlara işaret etmek için kullanılan terimlerin kendisini de bir yükleme çevirerek.

Bunu bir örnek üzerinde görelim: ''Bütün insanlar akıllıdır'' cümlesinde ''insan'' nitelemesinin kendisini de bir yüklem olarak yorumlarsak, o zaman cümlede ifade edilen önermeyi şöyle de ifade edebiliriz: ''Eğer herhangi bir şey insansa o şey akıllıdır''. Bu ikinci cümleyi ise kullanmakta olduğumuz notasyonla şöyle yazabiliriz:

(x)(IF İNSAN(x) THEN AKILLI(x))

Yorumu: Bütün x'ler için eğer x insansa o zaman x akıllıdır. Görüldüğü gibi artık tümel niceleyeni göstermek için ilk başta kullanılan x değişkenini sınırlandırmak gerekmiyor çünkü İNSAN yükleminin kendisi bu sınırlandırmayı sağlıyor.

IF-THEN tümcelerini OR cinsinden ifade etmemizi sağlayan kuralları da hatırlarsak yukarıdaki tümceyi şu şekilde yazabiliriz:

(x)(NOT İNSAN(x) OR AKILLI(x))

Yorumu: Bütün x'ler için x, ya insan değildir ya da akıllıdır.

Aynı şekilde ''Bazı insanlar akıllıdır'' cümlesini de şu şekilde formelleştirebiliriz:

∃(x)(İNSAN(x) AND AKILLI(x))

Yorumu: Hiç değilse bir x vardır, öyle ki o x insandır ve akıllıdır.

Şimdi bu ilkeler ışığında daha önce gözden geçirmiş olduğumuz ''Her erkek bir kadını sever'' cümlesine geri dönebiliriz. Bu cümlenin birinci yorumunu yani bütün erkeklerin sevdiği tek bir kadının varolduğu iddiasını bilgisayarın anlayamayacağı kabullerde bulunmadan, şu şekilde formelleştirebiliriz:

TEST 24

1 / Aşağıdaki cümleleri veya (OR) bağlantısı ile ifade edin.

a. Eğer Ahmet evdeyse, Ayşe mutludur.

b. Eğer Ahmet evde değilse, Ayşe çalışıyordur.

---

2 / Aşağıdaki cümleleri yüklemler mantığının tümceleri cinsinden ifade edin:

a. Ahmet Ali'den yaşlıdır veya Cengiz'den zengindir.

b. Ahmet Ayşe'yi veya Zeynep'i seviyor.
(Bağlantı dar veya(XOR) bağlantısı değildir).

c. Ahmet hem Ayşe'yi hem Zeynep'i seviyor.

d. Ahmet mutlu bir babadır.

---

3 / Aşağıdaki cümleleri yüklemler mantığının tümceleri cinsinden ifade edin:

a. Bütün hakemler inatçıdır.

b. Bütün tamsayılar ya tektir ya da çifttir.

(Bu soruyu cevaplandırırken XOR işlemcisini kullanmadan ''dar veya'' bağlantısını kurun. Yani ''Bütün sayılar ya tektir ya çifttir ama her ikisi birden değil.)

c. Bütün yaşlılar hastalandı.

d. Sadece yaşlılar hastalandı.

e. Erkeklerin hiçbiri kurtulamadı.

---

4 / Aşağıdaki cümleleri yüklemler mantığının tümceleri cinsinden ifade edin:

a. Her insanın bir dedesi vardır.

b. Herhangi iki tamsayının toplamı da bir tamsayıdır.

---

5 / Aşağıdaki cümleleri yüklemler mantığının tümceleri cinsinden ifade edin:

a. Eğer Ahmet işe Pazartesi giderse ve Ayşe Salı giderse ben de çarşamba giderim.

b. Eğer bir futbolcu hem güçlü hem de hızlı ise o değerli bir futbolcudur demektir.

c. Eğer Ahmet Pazartesi işe giderse ben de Salı ve çarşamba giderim.

d. Eğer Ahmet işe Pazartesi Ayşe Salı giderse ben de hem çarşamba hem de Perşembe giderim.

e. Eğer bir güreşçi yorgun ya da güçsüzse karşılaşmayı kaybedecektir.

∃(y)(KADIN(y) AND (x)(IF ERKEK(x) THEN SEVER(x,y)))

Yorum: Öyle bir y vardır ki y kadındır ve bütün x'ler eğer erkekseler o kadını severler.

İstendiği takdirde, tümel niceleyen ∃(y)'den sonra gelmek kaydıyla, tümcenin başına alınabilir:

∃(y)(x)(KADIN(y) AND (IF ERKEK(x) THEN SEVER(x,y)))

Çünkü asıl önemli olan, (x)'in ∃(y)'nin kapsamı içinde olmasıdır.

IF-THEN'i OR'a çeviren kuralı da uygulayarak,

∃(y)(x)(KADIN(y) AND (NOT ERKEK(x) OR SEVER(x,y)))

elde ederiz.

Yorum: Öyle bir y vardır ki y kadındır ve bütün x'ler için her x ya erkek değildir ya da o kadını sever.

Cümlenin ikinci yorumuna, yani her erkeğin sevdiği bir (fakat aynı değil) kadın olduğu yorumuna gelince, onu da şu şekilde formelleştirebiliriz:

(x)(IF ERKEK(x) THEN ∃(y)(KADIN(y) AND SEVER(x,y)))

Yorum: Bütün x'ler için eğer x erkekse öyle bir kadın vardır ki erkek onu sever.

Yine IF-THEN'i OR'a çeviren kural gereğince,

(x)∃(y)(NOT ERKEK(x) OR (KADIN(y) AND SEVER(x,y)))

Yorum: Bütün x'ler için ya x erkek değildir ya da öyle bir kadın vardır ki, erkek onu sever.

OR'un AND üzerinde dağıtılabilirliği kuralı gereğince ise

(x)∃(y)((NOT ERKEK(x) OR KADIN(y)) AND (NOT ERKEK(x) OR SEVER(x,y)))

Yorum: Bütün x'ler için ya x erkek değildir ve öyle bir y vardır ki y bir kadındır ve ya x erkek değildir ya da x y'yi sever.

Bir Örnek: Soyağacı ve Soyadları

Yüklemler mantığı içersinde bazı aksiyomların nasıl verildiğini ve bu aksiyomlardan hareketle nasıl sonuçların çıkarılabileceğini örneklemek üzere belli bir problem üzerinde çalışacağız. Problemimiz şu: Elimizde belli veriler (olgular) var. Bu olgulardan harektle belli bir iddianın doğru olduğunu kanıtlamak istiyoruz.

Elimizdeki olgular bazı insanların akrabalık ilişkileri ile ilgili. Bu ilişkilerden hareketle bu insanların soyadları ile ilgili bazı sonuçlar çıkarmak istiyoruz. Bildiklerimiz şunlar:

1. veri: Ahmet Ali'nin babasıdır.

2. veri: Ahmet Ayşe'nin abisidir.

3. veri: Ayşe hiç evlenmemiş.

Kanıtlanmasını istediğimiz iddia ise Ayşe ile Ahmet'in soyadlarının aynı olduğu.

''Zeki'' bir akılyürütme programından böylesi bir problemi çözmesi istendiğinde, sık sık başvurulan yol, bir çelişki **(contradiction)** bulunmasını istemektir. Bunun için program kanıtlanması istenen tümcenin değillenmişini kendiliğinden aksiyomlarına ekler ve sonra aksiyomlardan ve verilerden hareketle ve çıkarım kurallarını kullanarak, yeni tümceler türetmeye başlar. Eğer kanıtlanması istenen iddia gerçekten doğru ise, bu tümcenin değillenmişi aksiyomlara dahil edilmiş olduğundan, program eninde sonunda bir çelişki ile karşılaşacaktır. Bir çelişki ile karşılaşılması ise kanıtlanması istenen önermenin doğru (kanıtlanmış) olduğunu gösterecektir.

Örneğimize dönüp Ahmet'le Ayşe'nin soyadının aynı olduğunu kanıtlamaya çalışalım. Önce iddianın doğru olmadığını varsayalım. Bunu aksiyomlarımıza dahil edeceğiz. Eğer iddiamız aslında doğruysa program, bilinenlerden hareketle yeni verilere ulaştıkça, eninde sonunda bir çelişkiyle karşılaşacaktır.

Önce verili bilgileri yüklemler mantığının tümcelerine çevirelim:

(1) BABASI(Ali, Ahmet)
(2) KIZKARDEŞİ(Ayşe,Ahmet)
(3) EVLENMEMİŞ(Ayşe)

Genellikle gerek aksiyomların birbirleriyle, ge-

rekse türetilen yeni tümcelerin birbirleriyle ve aksiyomlarla AND bağlantısı ile bağlanmış olduğu varsayılır. Biz de bu uzlaşıma uyacağız; dolayısıyla yukardaki üç tümce aşağıdaki tümceye eşdeğerdir:

BABASI(Ali,Ahmet) AND KIZKARDEŞİ
(Ayşe,Ahmet) AND EVLENMEMİŞ(Ayşe)

Dikkat ederseniz 3. aksiyomumuz için EVLENMEMİŞ adlı birinci dereceden tek bir yüklem tanımladık. Eğer bu bilgileri daha karmaşık problemlerin de çözümünde kullanacak olsaydık, o zaman EVLENMİŞ diye bir yüklem tanımlamak ve verimizi NOT EVLENMİŞ(Ayşe) şeklinde yazmak elverişli olabilirdi. Ama bu problemde ''evlenmelerle'' ilgili çıkarımlara çok fazla ihtiyacımız olmadığından aksiyomlarımızı daha yalın tutmayı tercih ettik.

Ancak bu tümceler herhangi bir akılyürütme için yeterli olmayan önermeler çünkü hepsi tekil önermeler ve ayrıca henüz kullanmış olduğumuz yüklemlerin kendi aralarındaki ilişkiler tanımlanmamış durumda. Bunun için aksiyomlarımıza yüklemlerin kendi aralarındaki ilişkileri tarif eden yeni aksiyomlar eklemeliyiz:

(4) NOT EVLENMEMİŞ(x) OR NOT
BABASI(y,x) OR AYNISOYAD(y,x)
(5) NOT KIZKARDEŞİ(x,y) OR NOT
BABASI(z,y) OR BABASI(z,x)
(6) NOT AYNISOYAD(x,y) OR NOT
AYNISOYAD(y,z) OR AYNISOYAD(x,z)
(7) NOT BABASI(x,y) OR NOT ERKEK(y) OR
OĞLU(y,x)
(8) NOT OĞLU(x,y) OR AYNISOYAD(x,y)
(9) ERKEK(Ahmet)

Bu aksiyomlar akrabalık ilişkilerinin birbirleriyle ilişkilerini (örneğin 5) betimlediği gibi soyadları ile akrabalık dereceleri arasıdaki ilişkileri de betimliyor.

Son olarak kanıtlanmasını istediğimiz iddianın değillenmişini de aksiyomlara ekliyoruz:

(10) NOT AYNISOYAD(Ahmet,Ayşe)

Programın ilk ağızda çıkarabileceği sonuçlardan biri Ali'nin Ayşe'nin babası olduğu olabilir. Bu sonuç. Ayşe'nin Ahmet'in kızkardeşi olması, Ali'nin Ahmet'in babası olması olgularından ve kardeşlerin aynı babaya sahip olduğu hakkındaki genel önermeden (5) çıkıyor. Çıkarımı biraz daha yakından inceleyelim. 5. aksiyomumuzu şöyle yorumluyabiliriz:

''Bütün x, y ve z'ler için eğer x y'nin kızkardeşi ise ve z y'nin babası ise, o zaman z x'in de babasıdır.

Tümcedeki değişkenler tümel niceleyenin kapsamında olduğuna göre x, y ve z'nin yerine Ayşe Ahmet ve Ali değerlerini koyabiliriz. Bu takdirde tümce şu hali alır:

Eğer Ayşe Ahmet'in kızkardeşi ise ve Ali Ahmet'in babası ise o zaman Ali Ayşe'nin de babasıdır.

1. ve 2. tümceler tam da yukardaki koşullu önermenin öncülüğünü doğruladığına yani Ayşe'nin Ahmet'in kızkardeşi, Ali'nin de Ahmet'in babası olduğunu söylediğine göre koşullu önermenin doğru olduğuna, yani Ali'nin Ayşe'nin babası olduğuna karar verebiliriz:

1., 2. ve 5. tümcelerden
(11) BABASI(Ali, Ayşe)

Dikkat ederseniz 5. tümcenin kökü **(literal)** ile 1. ve 2. tümcelerin kökü birbirlerine denk düşmektedir. Denklikler şöyledir:

2. tümcenin kökü: KIZKARDEŞİ(Ayşe, Ahmet)
5. tümcenin kökü: NOT KIZKARDEŞİ(x,y)
1. tümcenin kökü: BABASI(Ali, Ahmet)
5. tümcenin kökü: NOT BABASI(x,y)

5. tümcedeki değişkenlere ''uygun'' değerler yerleştirildiğinde kökler başlarındaki NOT işlemcisi hariç aynı hale geliyor. Bu eşleştirme işlemine birleştirme **(unification)** denir. Bir tümcedeki köklerden biri, başka bir tümcedeki bir kökle eşleştirilirse ilk tümceden çıkarılır. 5. tümcedeki ilk iki öncül 1. ve 2. tümcelerle birleştirildiğinden 5. tümceden çıkarıldı ve böylelikle (değişkenlere uygun değerleri de vererek) 11. tümceyi elde ettik. Bir akıl yürütme programının başlıca işlevlerinden biri, bu şekilde birleştirilebilecek kökler aramak ve birleşmeyi mümkün kılacak değişken değerlerinin hangileri olduğuna karar vermektir.

Bundan sonra program Ali ile Ayşe'nin soyadlarının aynı olduğuna karar verebilir. Bu sonuç Ayşe'nin evlenmemiş olduğu hakkındaki tekil bilgiden (3), Ali'nin Ayşe'nin babası olduğu hakkındaki çıkarsadığımız bilgiden (11) ve evlenmemiş bir insanın babasıyla aynı soyadı taşıdığı hakkındaki genel bilgiden (4) çıkarsanmaktadır.

3., 11. ve 4. tümcelerden (12) AYNISOYAD(Ali, Ayşe)

Ali Ahmet'in babası olduğuna ve Ahmet erkek olduğuna göre Ahmet Ali'nin oğludur:

1., 9. ve 7. tümcelerden (13) OĞLU(Ahmet, Ali)

Demek ki Ahmet'le Ali'nin soyadı aynıdır:

13. ve 8. tümcelerden (14) AYNISOYAD(Ahmet,Ali)

Son olarak Ahmet'le Ali aynı soyadına sahip olduğunag öre (14) ve programa Ahmet'le Ayşe'nin soyadının aynı olmadığı bildirildiğine göre (10), program Ali'ye Ayşe'nin soyadının aynı olmadığına karar verir:

14., 10. ve 8. tümcelerden (15) NOT AYNISOYAD(Ali, Ayşe)

Bu önerme ise programın kendi çıkarsamış olduğu 12. önermenin değillenmişidir. Dolayısıyla aramakta olduğumuz çelişki bulunmuş olduğundan, program 10. önermenin (yani bizim kanıtlanmasını istediğimiz önermenin değillenmişinin) yanlış olduğuna, yani kanıtlanmasını istediğimiz önermenin doğru olduğuna karar verebilir.

Fonksiyonlar

Yukarıda verdiğimiz örnekle ilgili olarak, fazlasıyla basit bir problemi çözmek için problemin mantıki yapısını neredeyse gizleyecek karmaşık bir notasyon ve akılyürütme tarzı kullanıldığı düşünülebilir. Ama bu problemi incelemekteki başlıca amacımız çok daha karmaşık uzman sistemlerde de çalışacak bir akılyürütme programının çalışma tarz ve ilkelerini örneklemekti.

Örnek problemimizde bilgiler sorular ve iddialar herhangi bir programlama dili ile değil doğrudan doğrudan doğruya yüklemler mantığının kendi dili ile ifade edilmişti. Ama bu bağlamda elektronik aksamla ilgili bazı mülahazalar, doğrudan doğruya yüklemler mantığının kendi dilini değil de uyarlanmış bir versiyonunu kullanmamızı daha iktisadi kılabilir. Bu gerekçeyle, bizim bundan sonra gözden geçireceğimiz gerçek uzman sistemlerde, yüklemler mantığında varoluşsal niceleyenle karşılanan işlev fonksiyonlarla karşılanmaktadır. Bu yüzden gerçekten kullanılmakta olan bazı uzman sistemleri incelemeye geçmeden önce, bu sistemlerin dilini (PROLOG) yüklemler mantığından ayıran en önemli fark olan fonksiyonlar üzerinde durmamız gerekiyor.

Tıpkı varoluşsal niceleyen gibi fonksiyonların da işlevi ''öyle bir şey vardır ki'' ya da ''hiç değilse bir şey vardır öyle ki'' kavramlarını ifade etmektir. Nasıl yüklemler mantığında ''Her erkek bir kadını sever cümlesinin ikinci anlamını ifade ederken tümel niceleyenin kapsamı içersinde bir varoluşsal niceleyen kullandıysak aynı işlevi gerçekleştirmek için bir fonksiyon da kullanabiliriz:

(x)(NOT ERKEK(x) OR VARDIR(x, sevdiği(x))

Burada küçük harflerle yazdığımız ''sevdiği'' bir fonksiyondur ve işlevi (aynen matematiksel fonksiyonlarda olduğu gibi) her x değeri için o x değerine ''sevdiği'' ilişkisi ile bağlı olan tek bir değer üretmektir. Başka bir deyişle bu notasyonda ''her erkeğin sevdiği bir kadın'' deyimini çevirirken ''bir kadın''ın varlığını her x değeri için o x'ten hareketle ''sevdiği'' adlı bir fonksiyon aracılığıyla türetiyoruz.

Bu yüzden fonksiyonlar bireylerin karşılıklı ilişkiler içersinde bulundukları ve birbirleriyle olan bu karşılıklı ilişki aracılığıyla tanımlandıkları durumları göstermek bakımından çok elverişlidir. Örneğin evlilik ilişkisinde bireylerin durumu böyledir ve dolayısıyla ''karısı'' ve ''kocası'' terimlerinin bir yüklem olarak değil de bir fonksiyon olarak tanımlanması daha elverişlidir. Oysa ''oğul'' terimi, ebeveynler bilindiği zaman bile, tekil bir karşılıklı ilişkiye işaret etmemektedir dolayısıyla çoğu durumda (bizim örnek problemimizde olduğu gibi) bir yüklemle ifade edilmesi daha yerinde olur.

Birkaç başka örnek daha verelim: Matematiksel işlemler çok doğal olarak fonksiyonlarla gösterilir çünkü her matematiksel işlem, o işleme tekabül eden tek bir değer (yani sonucunu) üretir:

EŞİTTİR(kere(iki, üç),altı)

Bu ifade daha önce görmüş olduğumuz ve yüklem olarak ifade edilmiş şu ifadenin aynıdır:

ÇARPIM(iki, üç, altı)

Fonksiyonlar ve Değişkenler

Biraz önce ''evlilik'' ilişkisinin yüklemlerden ziyade fonsiyonlarla ifade edilmeye yatkın olduğunu söylemiştik. Aradaki farkı görelim:

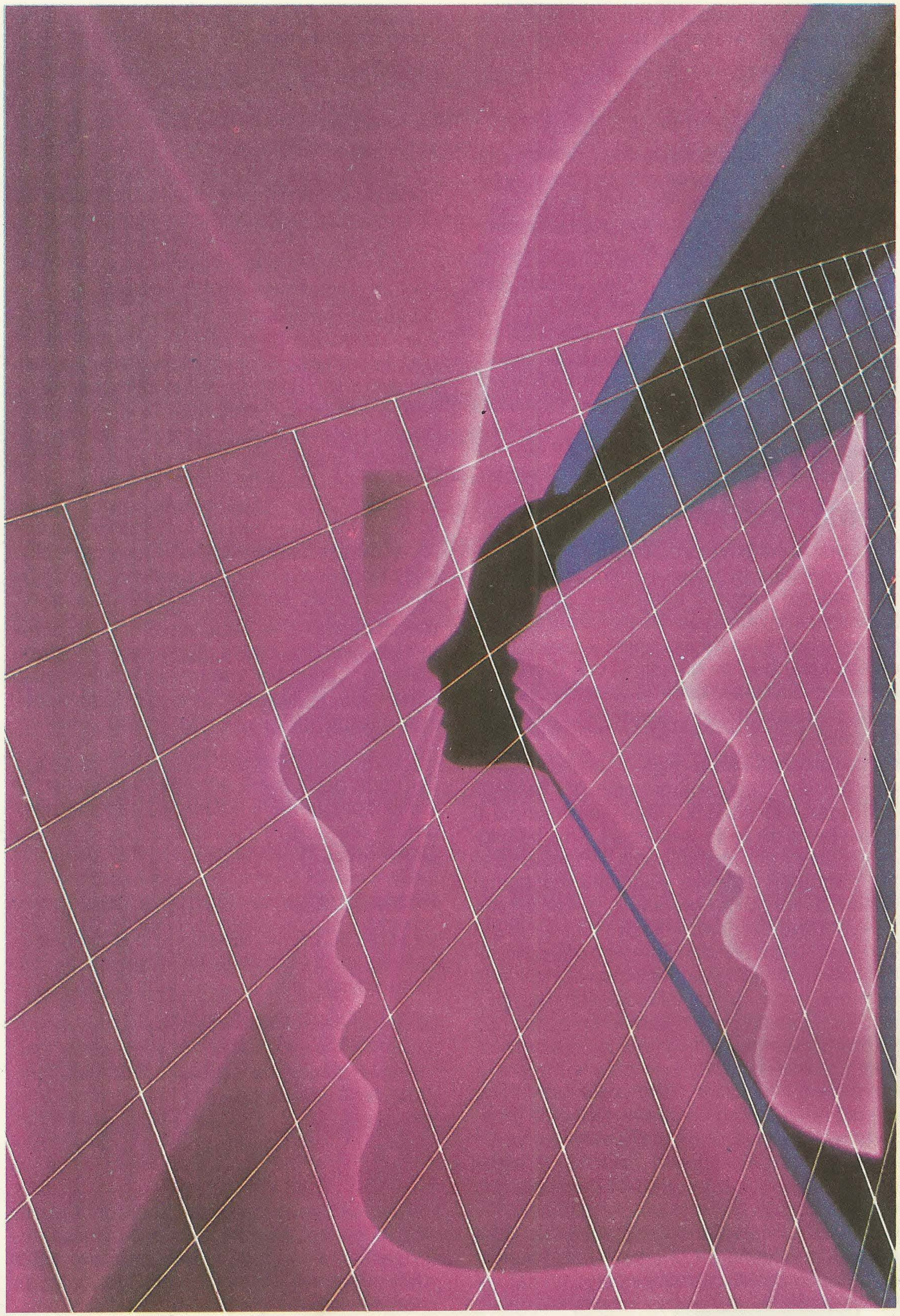
NOT EVLİ(x,y) OR KARISI(x,y) OR
KOCASI(x,y)

Yorumu: Eğer x y'yle evliyse x y'nin karısıdır ya da x y'nin kocasıdır.

Aynı ilişkinin fonksiyonlarla ifadesi:

NOT EVLİ(x,y) OR DIR(x, karısı(y))
OR DIR(x, kocası(y))

Burada DIR yanlızca kendi iki argümanı arasındaki ayniyet ilişkisini belirten bir yüklemdir. Bütün ifadenin yorumu şöyle yapılabilir: Eğer x'le y evliyse ya x'le y'nin karısı aynıdır ya da x'le y'nin kocası aynıdır.



1726

Uzman Sistemler

Satranç ya da briç oynamakla, tıp hukuk ya jeoloji eğitim yapmak arasındaki ortak özellik nedir? Herhangi birini yapıyorsanız ve sonunda bu konulardan herhangi birinde hatırı sayılır bir hüner kazandıysanız sizin için bir ''uzman'' denecektir. Bir uzmandan akılyürütmesi, enine boyuna ve derin konularda akılyürütmesi beklenir.

Dolayısıyla yapay zekâ iddiaları için bir bilgisayar programının bir uzmanın yaptıklarını yapıp yapamayacağı merkezi bir önem taşır. Aslında yapay zeka konusunun başında yazdığımız otomatik pratisyen programını çok ilkel de olsa bir uzman sistem olarak düşünmek mümkün. Öyleyse bir uzmanı, belli bir konu için gerekli kalifikasyonlara sahip kişi, bir uzman sistemi de bir uzman gibi iş gören bir bilgisayar programı olarak tanımlayabiliriz.

İki matrisi birbirleriyle çarpmak gibi bir işin güçlüğü entellektüel olmaktan ziyade mekanik bir güçlüktür. Dolayısıyla bu tür faaliyetler uzman sistemler başlığı altında incelenmez. Bu başlık altında işlenen konular daha çok mesleki bir eğitim ve kalifikasyon gerektiren işlerdir. Aşağıdaki örnekler mekanik ve entellektüel güçlük arasındaki farklar hakkında bir fikir verebilir.

1 / Satranç, briç, dama, tavla gibi bir oyun oynayabilen bir program genellikle uzman bir sistem sayılır.

2 / Tıbbi ya da mekanik bir kusur teşhis edebilen bir program uzman bir sistem sayılır.

3 / Bunların tersine karmaşık istatistiki formüller uygulayıp problem çözebilen bir program uzman bir sitem sayılmaz.

Uzman sistemlerin avantajları arasında ise şunları sayabiliriz:

1 / Bir uzman sistem ölmez. Böyle bir sistemin verimi sürekli olarak daha iyileşmese de, başlangıçtaki kalitesini korur. Ama maalesef aynı şeyi insanlar için söyleyemiyoruz. Kendini uzun yıllar boyunca geliştiren bir uzman bir süre sonra gerekli becerilerini yitirmeye başlayacaktır.

2 / Bir uzman sistem çok ucuza çoğaltılabilir. Örneğin tıbbi tavsiyelerde bulunabilen bir programın yazılabildiğini düşünün; bunun binlerce kopyasının yapılması hem çok ucuz hem de yararlı olurdu.

3 / Her biri farklı farklı hizmetler veren bir dizi uzman sistem geliştirmek mümkünse, bütün bu hizmetleri kendinde birleştiren bir uzman sistem yaratmak da mümkündür. Bu takdirde belki tümleşik **(integrated)** zekadan da sözetmek mümkün olabilir.

Oyun Oynayan Uzman Sistemler

Daha önceki bölümlerde sık sık oyun oynayan programlardan sözettik. Bu yüzden burada, bu konu üzerinde çok kısaca duracağız.

İlk stratejik oyun programlarından biri bir dama programıydı. Ancak dama için geliştirilen programlama teknikleri başarılı bir satranç programı yazılmasını mümkün kılmadı. Yine de bugün bir dizi iyi satranç programı bulunmaktadır. Bu programlar birçok sıradan oyuncudan daha iyi oynamakla birlikte henüz ustaların düzeyini aşamamış durumdalar. İyi satranç programları özel olarak satranç için tasarlanmış algoritmalar kullanıyorlar ve bu yüzden de bu algoritmalar başka oyunlara genellenemiyor.

İyi bir tavla programı da geliştirilmiştir. Bu program genellikler ustaları yenememekle birlikte bir kısa tavla maçında dünya şampiyonunu yenmeyi başarmıştır.

Son olarak da go oyununa geliyoruz: Japonlar oldukça zor bir strateji oyunu olan go'yu oynayabilecek bir program yazabilmek için geniş araştırmalar yapıyorlar. Ama bu alanda sıradan iyi oyuncuları bile yenecek bir programın geliştirilmesi daha bir süre alacağa benziyor.

Oyun oynayan uzman sitemlerle ilgili araştırmalar birkaç bakımdan ötürü ilginçtir. Birincisi, bugüne kadar birden fazla oyuna kaynaklık edebilecek bir algoritmanın bulunamamış olması. Sonunda belki böyle bir genel algoritma bulunabilir ama bugün için iyi stratejik oyun programlarının hiçbiri bizim burada örneklediğimiz türden genel algortimalar kullanmıyor. Dolayısıyla diğer uzman sistemler birkaç alana birden genellenebilecek akılyürütme tarzlarını algoritmalarla formelleştirmeye çalışırken, oyun programları da zekanın daha özgül, belli bir faaliyet alanı tarafından belirlenen yönlerine ışık tutabilir.

Diğer Uzman Sistemler

Araştırmacılar tıbbi teşhis kimyasal analiz ve coğrafi keşif gibi alanlarda kullanılabilen uzman sistemleri başarıyla üretiyorlar. Üstelik bu sistemlerin temelinde genellenebilir teknikler yatıyor. Bir oyun programının tersine, tıbbi teşhis sistemlerinden biri olan MYCIN'in üretmede kullanılan teknikler birçok alanda uzman sistem üretmek için kullanılabilir. Hatta MYCIN'in üretilmesi yeni uzman sistemlerin üretilmesinde kullanılabilecek EMYCIN adı verilen bir dizi yazılım aracının tasarımı ve uygulanmasına yolaçmıştır.

Bugün muhtemelen en çok tanınan başlıca uzman sistemler şunlardır:

1 / **MYCIN:** Tıbbın sınırlı bir alanında konsültasyon vermek için geliştirilmiştir.
2 / **DENDRAL:** Organik kimyasal bileşenlerin analizine yardımcı olmak için geliştirilmiştir.
3 / **PROSPECTOR:** Maden araştırma alanlarının seçimine yardımcı olmak için geliştirilmiştir.

Tipik bir uzman sistemde kullanılan akılyürütme bileşeni aslında çok fazla güçlü değildir. Bu sistemlerin canalıcı bileşeni kullanıcıyla iletişimi sağlayan arabirimdir.

Uzman sistemlerin gelişmesindeki gecikmenin ana nedeni, yeterince güçlü bir akıl yürütme bileşeninin eksikliği değil diğer bazı özelliklerin, özellikle de kullanıcının uzman sistemden kolaylıkla yararlanmasını sağlayacak ara birimlerin eksikliğiydi.

MYCIN gibi bir uzman sistem üç bileşenden oluşur:

1 / İlki sistemle kullanıcısı arasındaki iletişimi sürdürürken aynı zamanda da gerekli çıkarımları yapan **akılyürütme bileşeni**dir.
2 / İkincisi gerçek bir uzmanın bilgisini edinen bileşendir. Bileşen öğretici konumunda olan gerçek insanın programlama ya da bilgi işlem hakkında çok az bilgisi olduğu durumlarda bile kullanılabilecek şekilde tasarlanmıştır. Amaç uzmanın bildiği verileri ve akılyürütmede kullandığı kuralları bilgisayara aktararak, aleti uzmanlık yapabilecek bir düzeye getirmektir. Bu bileşenin adı **bilgi edinme bileşeni**dir.
3 / Üçüncüsü ise bu işin uzmanı olmayan bir kullanıcın bile programdan yararlanabilmesini sağlayan **kullanıcı arabirim**dir.

Akılyürütme Bileşeni

Birçok uzman sistemin akılyürütme bileşeni üretim (ya da çıkarım -**production**-) kuralları na dayalıdır. Üretim kurallarını MYCIN'de aldığı biçime bir örnek verelim:

Eğer (IF)

1 / Tedavi gerektiren enfeksiyon menenjitse, ve
2 / Hastada ciddi deri ya da doku enfeksiyonu belirtileri varsa, ve
3 / Hastanın kültüründe organizmalara rastlanmadıysa, ve
4 / Enfeksiyon bakteriyal türdeyse

O zaman (THEN)

Enfeksiyona neden olan organizmanın (kültürde ya da doku sıvısında görülen hariç) staphylococcus-coag-pos.(75) streptococcus-a grubu (.5) olma ihtimali vardır.

Verilen kuralın dikkate değer birkaç noktası var. Önce kural kolaylıkla iki tümceye indirgenebilir (ya da tümce halinde kodlanabilir). Öncüldeki koşullar doğrulandığında iki farklı sonuçtan birine varılabileceği için, iki tümce (clause) gereklidir. Çünkü verildiği haliyle kuralın sonuç kısmında örtük bir 've' (AND) bulunmaktadır. İkincisi öncüldeki sonuçlardan her ikisi ile de bir güvenilirlik faktörü (**confidence factor**) eşleştirilmiştir. Değişik uzman sistemler böyle bir güvenilirlik faktörü kullanılmasını gerektiren belirsizlik durumları ile farklı farklı şekillerde başa çıkarlar- bazıları koşullu olasılık ve Bayes teoremi kullanırken diğerleri daha olaya özgü *(ad hoc)* teknikler kullanır. Üçüncüsü hernekadar tümcelerin makine içersindeki saklandıkları yazı düzeni normal olarak kuralın kendisi kadar karmaşıksa da, kullanıcıya sunulan format şaşırtıcı ölçüde okunaklıdır.

İnsanları en çok şaşırtan üretim kurallarının kolaylıkla tümcelere çevrilebilmesi ve hatta çoğunun tek bir tümceye indirgenebilmesidir.

Kullanılan yazılım paketi olan EMYCIN çıkarım mekanizmalarının çoğunu üretim kuralları kullanarak geriye-zincirlemeye **(backchaining)** dayandırır. EMYCIN'de bu tür çıkarım mekanizmalarına sonuç **(consequent)** kuralları denir. Bu kuralların sırası sonuçların çıkarılma sırasını etkilemez. Ayrıca sonuç kurallarının yanısıra öncül **(antecedent)** kuralları da kullanılır.

Hones