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

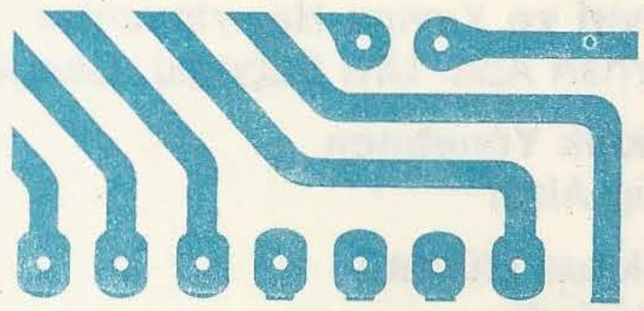
İletişim Yayınları

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

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

BİLGİSAYAR

ÇAĞDAŞ TEKNOLOJİ ANSİKLOPEDİSİ



Cilt 3

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
Serhan Ada, Ulvi Bağoğlu, Cengiz Onural

Teknik Yönetmen
Edip Alşar

Reklam Müdürü
Aslı Yada

Grafik/Pikaj
Saadettin Oktay

Dizgi
Serpil Şahin, Hüsnü Abbas

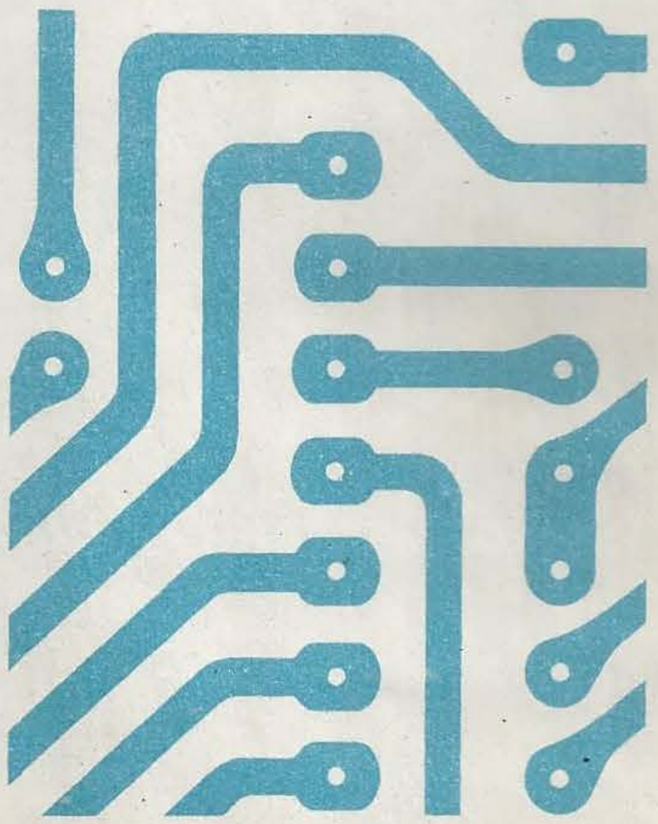
Montaj
H. Miray Yüzgeç

Baskı
Alaş Matbaası

© *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
Telefon: 520 14 53/54/55

© Bütün hakları İletişim Yayıncılık A.Ş.'ne aittir.



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

Bilgiyazarın Kullanımı

LPRINT komutu, daha önce öğrendiğimiz çeşitli şekilleriyle, verilerin belli sayıdaki yazı düzenlerine göre yazılmasını sağlar. Buna rağmen, kullanıcı programlarının yazılması sırasında ortaya çıkan gereklerin hepsini yerine getiremez.

Örneğin, bir karakterin büyüklüğünün veya iki satır arasındaki aralığın değiştirilmesi gerekebilir. Bu fonksiyonlar ancak, belli ASCII simgelerinden oluşan özel güdüm karakterlerinin bilgisayara verilmesiyle sağlanır. Bunlar bilgisayar için basılacak karakter değil, güdüm anlamını taşırlar.

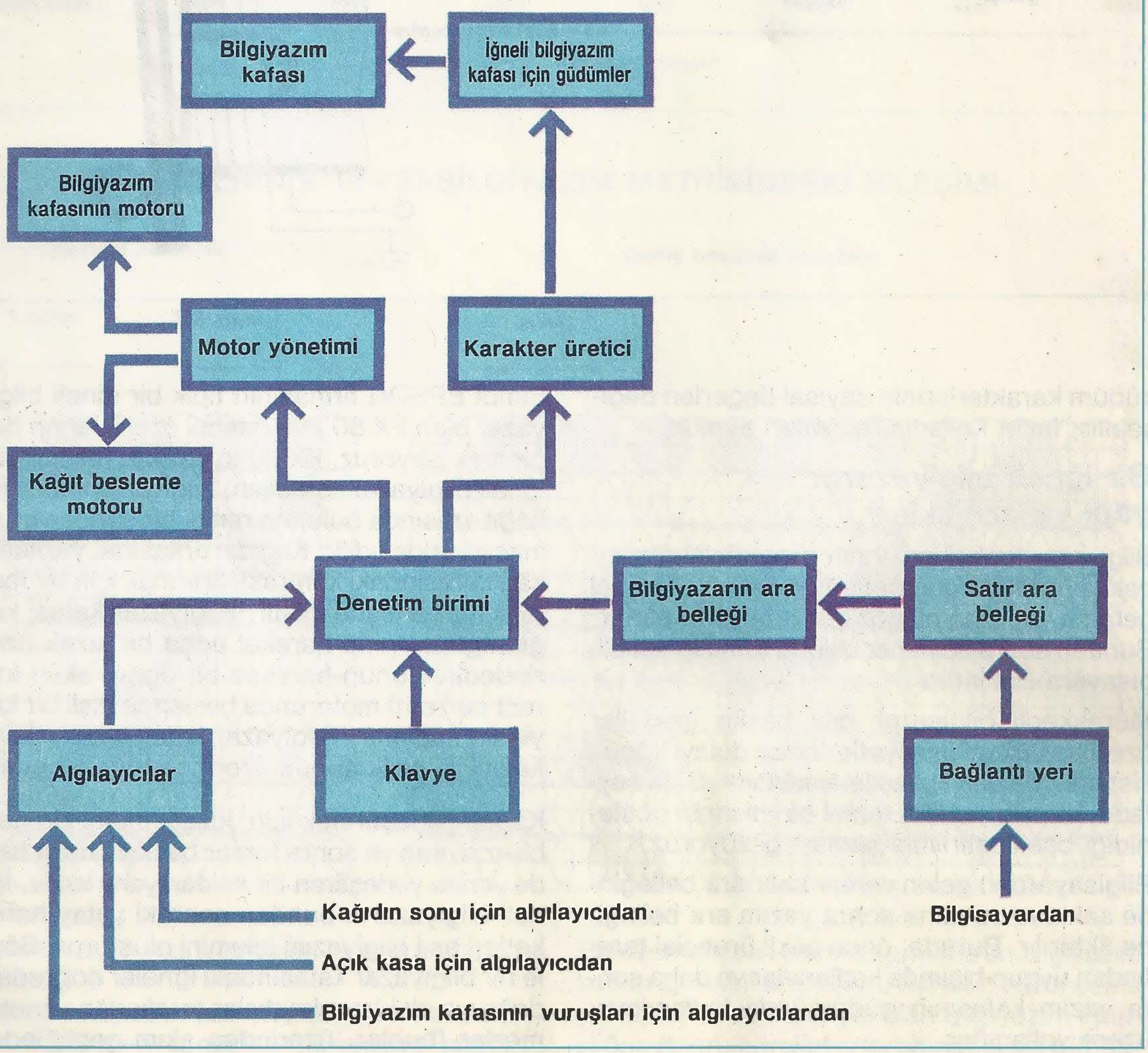
Bu güdüm karakterleri ile anlamları, bilgisayar imalatçısı tarafından belirlenir ve kullanım kılavuzunda onaltı veya on tabanlı sayılar olarak geçer. Bilgiyazara verilmeden önce bunlar, kullanıcı program tarafından ASCII karakterlerine çevrilmelidir. Belli bir (on tabanlı) sayısal değeri ASCII koduna çeviren komut CHR$(N)'dir. Örneğin 12 sayısı aşağıdaki komut yardımıyla ASCII koduna çevrildikten sonra bilgiyazara yollanır:

```
LPRINT CHR$(12)
```

Bilgisayar bu kodla karşılaşır karşılaşmaz, kodun işaret ettiği fonksiyonu yerine getirmeye başlar. Sayılar ve işaret ettikleri bilgiyazım işlevleri bilgiyazarın türüne göre değişir.

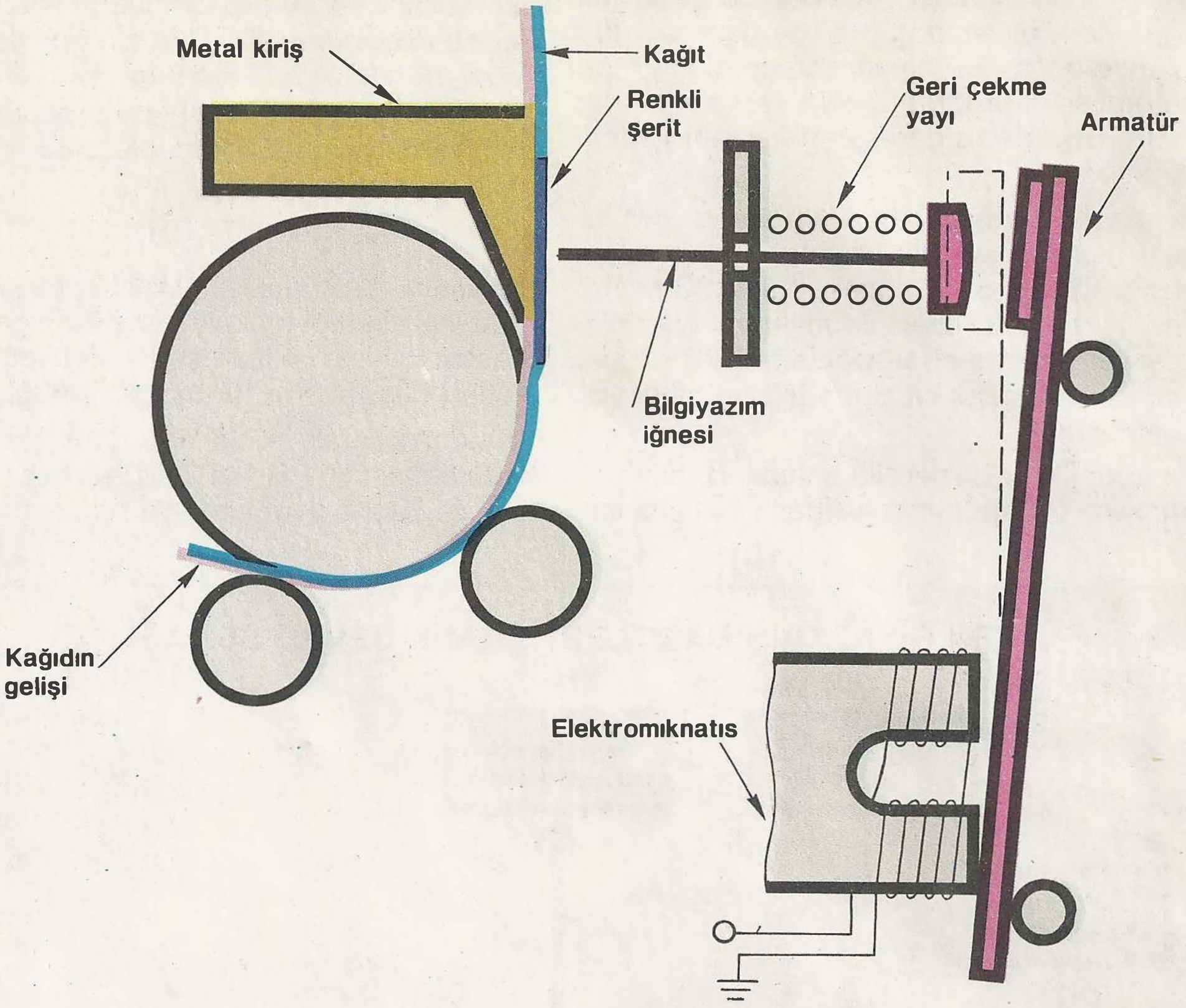
Bundan sonraki bölümlerde örnek olarak Olivetti firmasının PR 1472 bilgiyazarını kullanacağız. Başka uygulama ve firmalar için bazı

BİLGİYAZARIN BASİTLEŞTİRİLMİŞ DEVRE ŞEMASI



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

OLİVETTİ PR 1450 BİLGİYAZARIN BİLGİYAZIM MEKANİZMASI



güdüm karakterlerinin sayısal değerleri değişebilir, fakat kullanım biçimleri aynıdır.

Bir iğneli bilgiyazarın yapı ve özellikleri

Bilgiyazar fonksiyonlarının programlanmasındaki ayrıntılara geçmeden önce, bu çevresel gerecin yapısına bir göz atmakta yarar vardır. Bundan sonra söyleneceklerin tümü iğneli bilgisayara ilişkindir.

Mürekkepli bilgiyazar gibi başka modeller özellikle yüksek maliyetlerinden dolayı küçük sistemlerde pek kullanılmamaktadır. 579. sayfada, bir bilgiyazarın temel birimlerinin gösterildiği basitleştirilmiş şemayı görüyoruz.

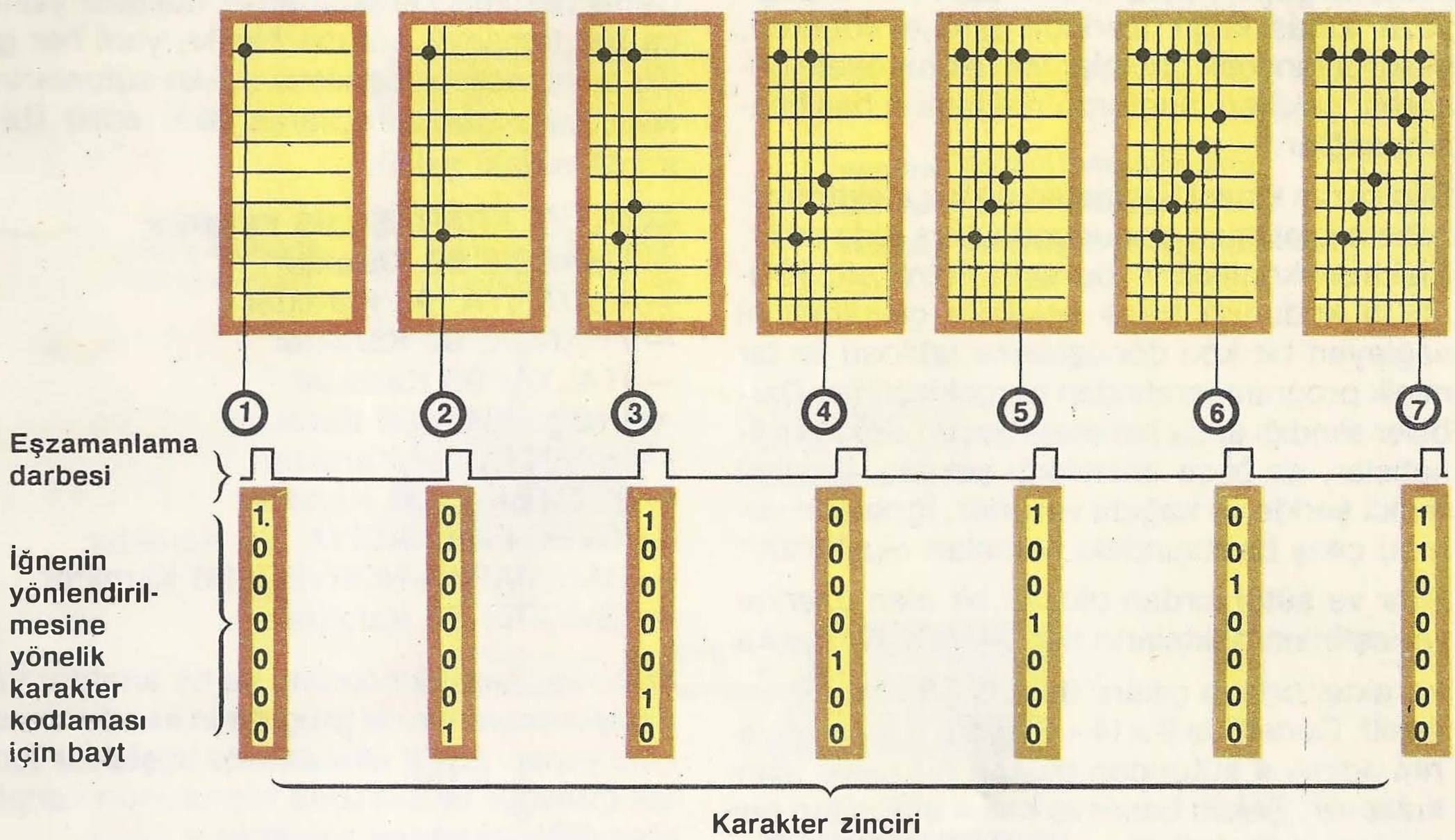
Bilgisayardan gelen veriler satır ara belleğinde saklanır ve daha sonra yazım ara belleğine aktarılır. Burada, önce şekil üreticisi tarafından uygun biçimde kodlanırlar ve daha sonra yazım kafasının güdümünde kullanılmak üzere yollanırlar.

Şimdi EPSON firmasının tipik bir iğneli bilgiyazar olan FX 80'inin önemli özelliklerine değinmek istiyoruz. FX 80'in yazıları (baskıları), iğneli bilgiyazar kafasının, bilgiyazar kafası ile kağıt arasında bulunan renkli bir şeride vurulmasıyla elde edilir. Kağıdın arkasına, yazılarla kağıt arasındaki kontrastı artırmak için bir metal kiriş yerleştirilmiştir. Bilgiyazar kafası kağıda göre enine hareket eden bir kızak üzerindedir. Bunun hareketi bir doğru akım (direct current) motorunca beslenen dişli bir kayış ile sağlanır. Bilgiyazar kafalı kızak, daire kesitli iki çelik çubuk üzerinde hareket eder.

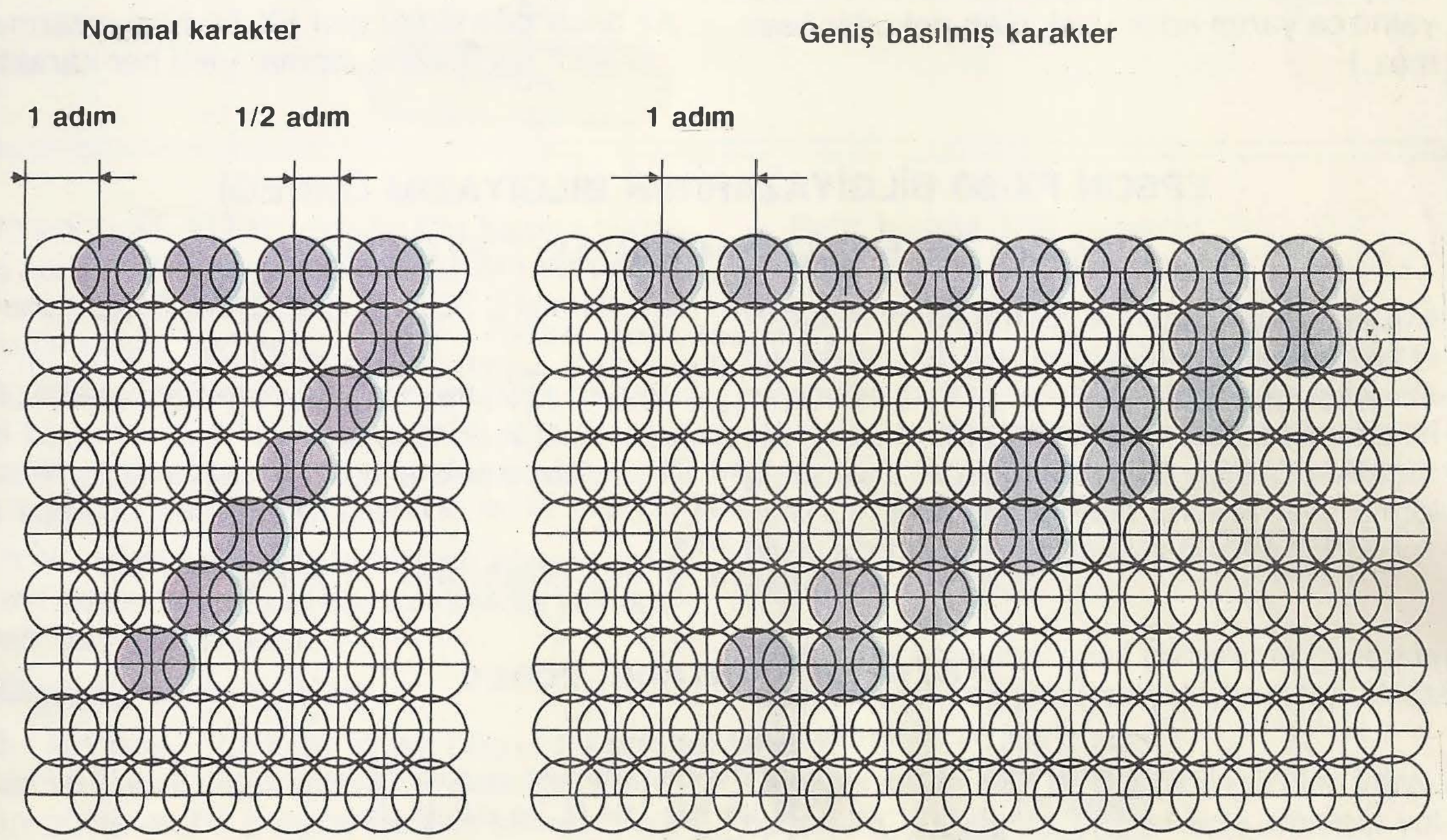
Kağıdı yerleştirmek için, kızağı metal kirişten biraz ayıran ve sonra tekrar baskıya hazır halde yerine yerleştiren bir koldan yararlanılır. İğneli bilgiyazarın bundan sonraki yatay hareketleri asıl bilgiyazım işlemini oluşturur. Böyle bir bilgiyazar kafasındaki iğneler doğrudan doğruya elektromıknatıslar tarafından yönetilmezler (bunlar, üzerinden akım geçtiğinde,

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

OLİVETTİ PR 1450 BİLGİYAZARLA 7 KARAKTERİNİN BİLGİYAZIMI



KARAKTERİN 9*(4 + 3)-BİLGİYAZIM MATRİSİNDEKİ BİLEŞİMİ



çevresinde bir manyetik alan oluşturan bobinlerdir).

Elektromanyetik bobinler, küçük bir armatürün bilgiyazım iğnesine vurmasını sağlar. Elektromanyetik akım bu armatürün bilgiyazım kafası iğnesini öne doğru hareket ettirmesine yol açar. Yeterli enerjisi olan bilgiyazım iğnesi öne doğru hareket eder ve renkli şerit ile ka-

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

ğıdı metal kirişe çarptırır. Bunun üzerine iğne geri gider ve bir yay yardımı ile başlangıç konumuna getirilir (bkz. s.580'deki şekil). Bilgiyazar kafası kağıt üzerinde hareket ederken, elektromanyetik bobinler tek tek harekete geçerek, şekilleri oluşturan noktaların basılmasını sağlar.

Bilgiyazım iğnesini yöneten dokuz elektromıknatısın işletilmesi sonucunda çıkış elde edilir. Elektromıknatısların bu sıralı hareketi, ISO-ASCII kodunun basılı şekillere çevrilmesini sağlayan bir kod dönüştürme tablosu ile bir minik program tarafından gerçekleştirilir. Darbeler alındığı anda harekete geçen elektromıknatıslar, az önce anlatıldığı şekilde, iğneleri renkli şeride ve kağıda vururlar. İğnelerin vuruşu çıkış baskısındaki noktaları oluşturur.

Satır ve sütunlardan oluşan bir alan üzerine yerleştirilen noktaların her bilesimi bir başka karakter ortaya çıkarır (bkz. s.580'de üstteki şekil). Genellikle 9 × (4 + 3), yani 9 satır ve yarım adımlı 4 sütundan oluşan bir baskı alanı kullanılır. Şeklin basılmış hali 4 sütundan her biri için en fazla 9 düşey noktadan oluşur. Ayrıca sütunların arasında kalan bölgelere de nokta basmak mümkündür (3, yarım adım).

Böylece yalnız 36'sı basılabilen 63 (9 × 7) olası konum elde edilir. (Aynı satırda birbirinden yalnızca yarım adım uzak olan noktalar basılmaz.)

FX 80 bilgiyazarında aşağıdaki karakter türleri öngörülmüştür.

Geniş basılmış karakterlerde, noktalar yalnızca her tamsayılı sütuna basılır, yani her geniş şekil, normal basılmış şeklin sütunlarının arası yarım adım açılarak elde edilir (bkz. s.581 alttaki şekil).

— ULUSLARARASI: 96 Karakter
— USASCII: 96 Karakter
— ALMANYA: 96 Karakter
— FRANSA: 96 Karakter
— İTALYA: 96 Karakter
— İNGİLTERE: 96 Karakter
— İSPANYA: 96 Karakter
— PORTEKİZ: 96 Karakter
— İSVEÇ-FİNLANDİYA: 96 Karakter
— DANİMARKA-NORVEÇ: 96 Karakter
— İSVİÇRE: 96 Karakter

Kullanıcı, karakter türünü ya bir anahtar yardımıyla seçer; ya da programın kendisi bu seçimi yapar. ASCII alfabesinde boşluklar (boşluk çubuğu) ve sıfırlama komutunun karşılığı olan DEL karakteri basılmaz.

Belli bir karakter türünde tasarlanmamış olan bir karakter basılmaya çalışıldığında, ya çizisel olarak ''III'' şeklinde gözükür ya da basılması engellenir.

Az önce belirtildiği gibi FX 80 bilgiyazarının, ''siyah'' bilgiyazım yapma, yani her karakte-

EPSON FX-80 BİLGİYAZARININ BILGİYAZIM ÖRNEĞİ

PRINTER CONTROL CODES

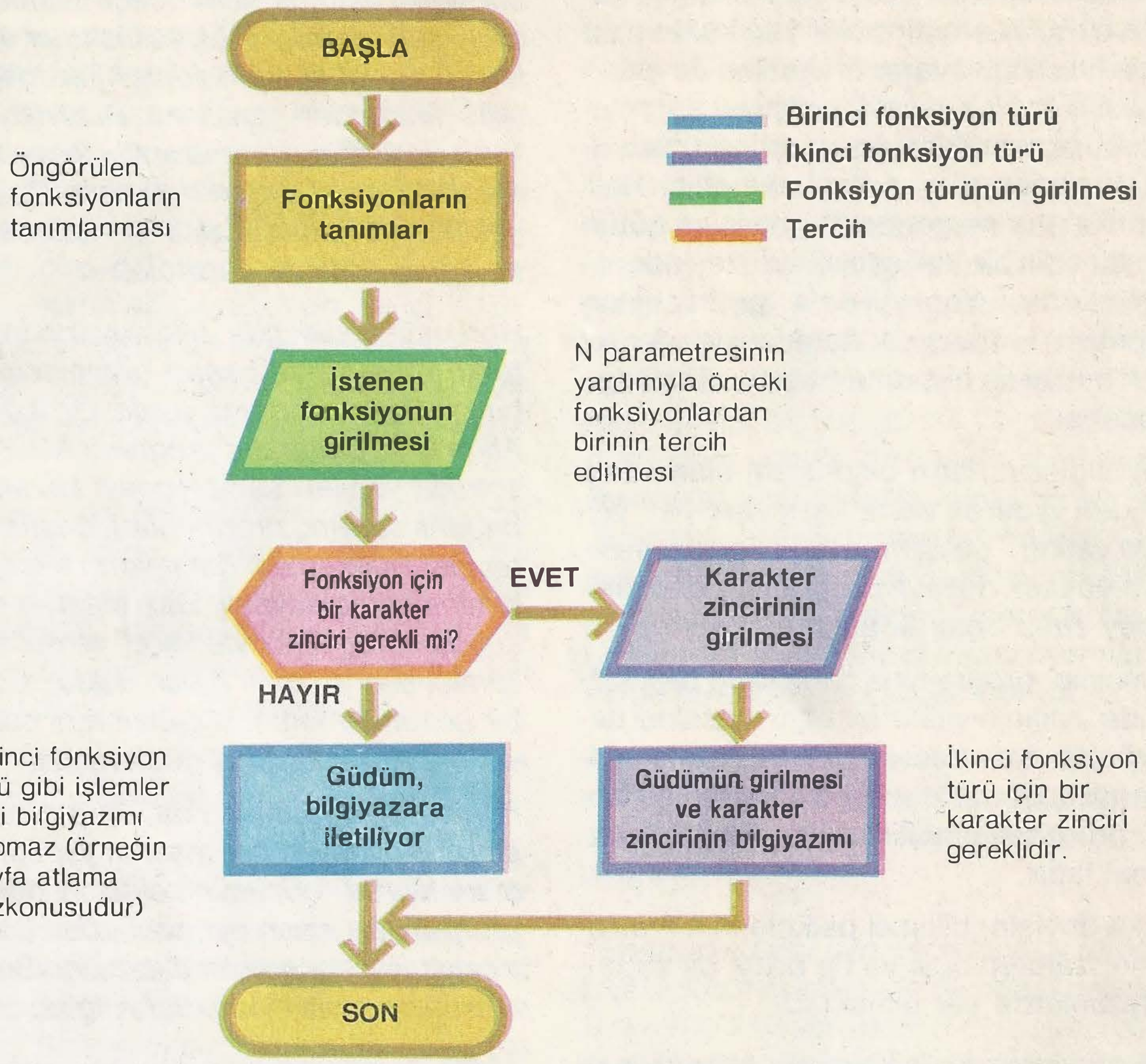
PRINT METHOD	Impact Dot Matrix
PRINTING SPEED	100 Char/Sec with fast carriage return
THROUGH PUT	50 lpm full line 86 lpm with 40 char
CHARACTERS COMPOSITION	9x7 (4+3) Dot Matrix
CHARACTERS PER LINE	1) 80 Char/line with 10 Char/inch 2) 132 Char/line with 16.6 Char/inch
LINE FEED	100 msec
COPIES	Up to 1 original and 2 copies

PRINTER CONTROL CODES

PRINT METHOD	Impact Dot Matrix
PRINTING SPEED	100 Char/Sec with fast carriage return
THROUGH PUT	50 lpm full line 86 lpm with 40 char
CHARACTERS COMPOSITION	9x7 (4+3) Dot Matrix
CHARACTERS PER LINE	1) 80 Char/line with 10 Char/inch 2) 132 Char/line with 16.6 Char/inch
LINE FEED	100 msec
COPIES	Up to 1 original and 2 copies

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

BİLGİYAZARLARDA GÜDÜM KARAKTERLERİNİN KULLANIMI



rin normalin iki katı yoğunlukta basma olanağı vardır. Bunu yandaki basılı örnekte görüyoruz. Siyah olarak basma, ESC 3 komutunun verilmesiyle sağlanır.

Bu işlem, bilgiyazım satırının istenilen herhangi bir konumundan başlatılabilir. Etki, komut verilir verilmez ve bu etkiyi tersine çeviren ESC 4 komutu gelene dek sürer.

FX 80 bilgiyazar çalıştırıldıktan sonra, program üzerinden uygun komut verilirse yalnız siyah baskı elde edilir.

Bilgiyazım örnekleri

Bu sayfada, tipik bir iğneli bilgiyazarın küçük ve kişisel bilgisayarlar için taşıdığı olanakları gösteren bir programın akış şeması verilmiştir.

Programda, Olivetti firmasının PR 1471 bilgiyazarı sözkonusu ediliyor.

— İki kat genişlik
— Satır başına 132 karakter
— Satır başına 159 karakter
— Satır başına 220 karakter

Programın tek amacı, kullanıcının klavye üzerinden verdiği uygun güdüm karakterleri ile tanımlanan farklı yazı düzenlerine karşılık olarak, aynı metnin farklı yazı düzenlerinde bilgiyazımını sağlamaktır. Yarı yarıya basitleştirmek için akış şemasında arka arkaya birçok baskı aşaması oluşmasını sağlayan döngü kaldırılmıştır.

Ancak bu şemanın nasıl programa çevrilebileceği makineden makineye büyük farklılıklar göstermektedir.

Örneğin sayfa 582'de aşağıda görülen karakterleri FX 80 bilgiyazarında elde etmenin yollarından biri şu komutu vermektir:

LPRINT CHR$(27); ''!''; CHR$(1)

Burada CHR$(27) bilgiyazarın genellikle ESC diye adlandırılan güdüm koduna işaret etmektedir. Bu kod, başka şekilde tanımlanmadığı

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

Bilgisayar ve Zeka

Bu makalede, Milano Politeknik Okulu'nda yapay zeka üzerine araştırmalar yapmakta olan B.Meltzer'in, bilgisayarın olanakları ile insan psikolojisine özgü bilişsel (cognitive) süreçler arasındaki benzerlikleri konu edinen bazı ilginç düşüncelerine yer verilmektedir. Belli fonksiyonları yerine getirmeye yarayan bütün programlar belli bir ''bilişsellik'' düzeyinde tanımlanmışlardır. Yani yerine getirecekleri fonksiyonların karmaşıklık derecesine göre az veya çok miktarda birbirine bağımlı olan bilgileri kapsarlar.

En yeni bilgisayarların bilgi-işlem alanındaki verimlilikleri giderek daha karmaşık ve ''öğrenmeye yatkın'' geliştirmeye imkan vermektedir. Bu yüksek düzeyli programlar genellikle ''yapay zeka'' olarak adlandırılırlar.

Programcının, programına daha fazla bilgi kattığı ölçüde (bilginin yalnız bir olgu meselesi değil, bir yöntem meselesi de olduğu hatırlanmalıdır), kendi düşünsel faaliyetinin artan bir bölümünü programlanabilir bir şekilde ifade etmiş olmaktadır.

Bir başka deyişle, bilişsel psikolojinin malzemeleri her zaman, açık ya da örtük bir biçimde programlarda yer almaktadır.

Yine de günümüzde psikolojiyle, felsefeyle ve diğer alanlarla ilgilenen birçok bilim adamı, çeşitli gerçeklerden ötürü bilgisayar programlarının insanınkilere benzeyen zihinsel başarıların üstesinden gelebilecek niteliklere sahip olmadığı düşüncesindedirler.

Ben bu gerekçelerin bir kısmından bahsetmek istiyorum.

En çok duyulan itirazlardan biri, insan zekasının sinir hücrelerinden oluşan biyolojik ortamlar temeli üzerinde çalıştığı ve bu noktada belirgin olarak transistörler, mikro devreler ve manyetik bantlardan ayrılmakta olduğudur. Bu nedenle bilişsel süreçlerin oluşum tarzı, bilgisayar programlarınınkinden tamamiyle farklı olmak durumundadır.

Herhalde bilgisayarlarla uğraşan insanların pek azı bu akıl yürütmelerden etkilenmektedir. Çünkü onlar için aynı programın, tamamiyle farklı yapı ve mekanizmaya sahip iki ayrı bilgisayarda, verilen görevleri esasta aynı yöntemlerle yerine getirmesi yeni bir olgu değildir.

Çok duyulan bir başka itiraz da, doğrudan doğruya makinelere yöneliktir. Yapay zekanın ''bilgi'' programlarının, düşünmeyi gerektiren sorunları aslında dolambaçlı aramalarla çözdüğü ileri sürülüyor. Oysa insanlar aynı sorunları çözerken kestirmelerden, sezgilerden, yaratıcı fikirlerden yararlanır. Dolayısıyla bilgisayarın dolambaçlı aramalarıyla insan düşüncesi arasında temel bir farklılık vardır. Bu görüş gerçeklikten o kadar uzaktır ki, sadece birkaç tane ilginç örnekle çürütülebilir.

Herkesin bildiği gibi, bilgisayarın insana karşı farklı düzeylerde başarı göstererek oynadığı birçok oyun programı vardır. Örneğin Slate ve Atkin'in 4, 5 satranç programı ABD'de Minnesota'da yapılan şampiyonayı kazanmıştır. Bu başarılı satranç programları, çeşitli zorluk derecelerindeki olası hamlelerin eksiksiz olarak sınanmasına dayanır. Bazı eleştirmenlerin düşündükleri gibi, böyle ''kör'' ve dolambaçlı çözümlemelerle hiçbir ilgileri yoktur. Çünkü böyle bir çözümlemenin, uygulamada çok kullanışsız olduğu kolaylıkla gösterilebilir.

Bilgisayarın yaptıkları da, usta olsun-olmasın satranç oynayan her insanın yaptığı gibi, oyun olanaklarını çözümlemektir. Çoğu satranç programları insan oyunculardan daha ayrıntılı çalışmayı becerebilmektedirler. Belki bu ilişki açısından tavla oyunu daha ilginç bir örnektir.

1979 Temmuzu'nda Montecarlo'da, Dünya Tavla Şampiyonu İtalyan Luigi Villa, Hans Berliner BKG 9.8 (Backgammon 9.8) programı tarafından 7:1 yenilgiye uğratıldı. Berliner, Carnegie Mellon Üniversitesi'nde yapay zeka alanında araştırmacı olarak görev yapıyor. Kendisi bu programı, ilk oyun programlarından biri olan Samuel'in dama oyununda kullanılan temel ilkenin ileri boyutlarda geliştirilmesi temeli üzerine kurmuştur. Bu geliştirilmiş biçimde program, sistematik aramaya çok az başvurmaktadır. Oyunun hamleleri, sistematik arama yerine, oyunu karakterize eden sınırlı sayıda elemanın yardımıyla durumun genel bir değerlendirmesinden türetilir. Berliner, Villa ile yapılan karşılaşmadaki oyun hamlelerini inceledikten sonra şu sonucu varmıştır: ''BKG 9.8'in iyi oynadığı kesindir. Villa hemen hemen tüm oyun boyunca teknik olarak doğru oynadığı halde, program 73 durumun yalnızca 8 tanesinde daha kötü oynamıştır. Bu hatalardan sadece biri programı zor durumda bırakmıştır. BKG 9.8 hayal gücü kullanımı gerektiren du-

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

rumlardan yüzünün akıyla çıkmıştır. Kimin siyahla, kimin beyazla oynadığını bilmeyen biri, insanın ne kadar mükemmel ve bilgisayarın ne kadar kötü oynadığını kolaylıkla düşünebilirdi.'' Dilin, zekanın penceresi olduğu söylenir. Birbirinden çok farklı yaklaşımlarla doğal dille yazılmış bir metni anlayan, yeniden yorumlayarak, yazan, tercüme eden ve özetleyen programların yazılması denenmiştir. Bu alanda bugüne kadar erişilenler hakkında kesin bir fikir verebilmek için, bir gazetenin haberlerini inceleyen, bunların arkasından en ilginç makaleleri çıkaran ve bunların İngilizce, İspanyolca veya Rusça gibi herhangi bir dilde kısa bir özetini veren bir programdan bahsetmek istiyorum.

Bu programın adı FRUMP'tur. (Fast Reading and Understanding Memory Program-Hızlı Okuma ve Anlama Bellek Programı) ve doğal dilin anlaşılması için çalışan, Roger Schank başkanlığındaki bir çalışma grubu tarafından Yale Üniversitesi'nde geliştirilmiştir.

FRUMP'un yeteneklerini aşağıda bir Amerikan gazetesinden alınan bir haber üzerine gösterelim:

''Dün akşam şiddetli bir deprem bütün Kuzey İtalya'yı sarstı. Deprem sırasında Venedik kentinin kuzeydoğusundaki Yugoslavya sınırına yakın bölgedeki kasabalar tümüyle yıkıldı. İtalyan İçişleri Bakanlığı'nın bildirdiğine göre en az 95 kişi öldü ve 1000 kişi yaralandı. Bir hükümet sözcüsü Udine kentinde yıkıntılar altında en az 200 ölünün gömülü olduğundan korkulduğunu bildirdi. Bu kent Roma ile Viyana arasındaki ana demiryolu hattı üzerindedir ve yaklaşık 90.000 nüfusa sahiptir. İtalyan polis örgütü Karabinieri'nin sözcüsü, Alplerin eteklerindeki yarım düzine kentten, bütün ailelerin evlerinin yıkıntıları altında gömülü olduğu, bölge merkezlerinde telekomünikasyon bağlantılarının henüz kopuk olduğu ve ağır kayıplar verildiği yolunda haberler alındığını açıkladı. Depremin şiddetinin, Richter ölçeğine göre 6.3 derece olduğu tespit edildi. Yerleşim bölgelerinde 4 şiddetinde bir deprem orta miktarda hasara neden olur. 6 derece, ağır hasar demektir ve 7 şiddetindeki bir deprem yokedici boyutlardadır.''

Ve işte FRUMP'un elde ettiği sonuç: ''İtalya'daki depremde 95 kişi öldü ve 1000 kişi yaralandı. Deprem Richter ölçeğine göre 6.3 şiddetindeydi.''

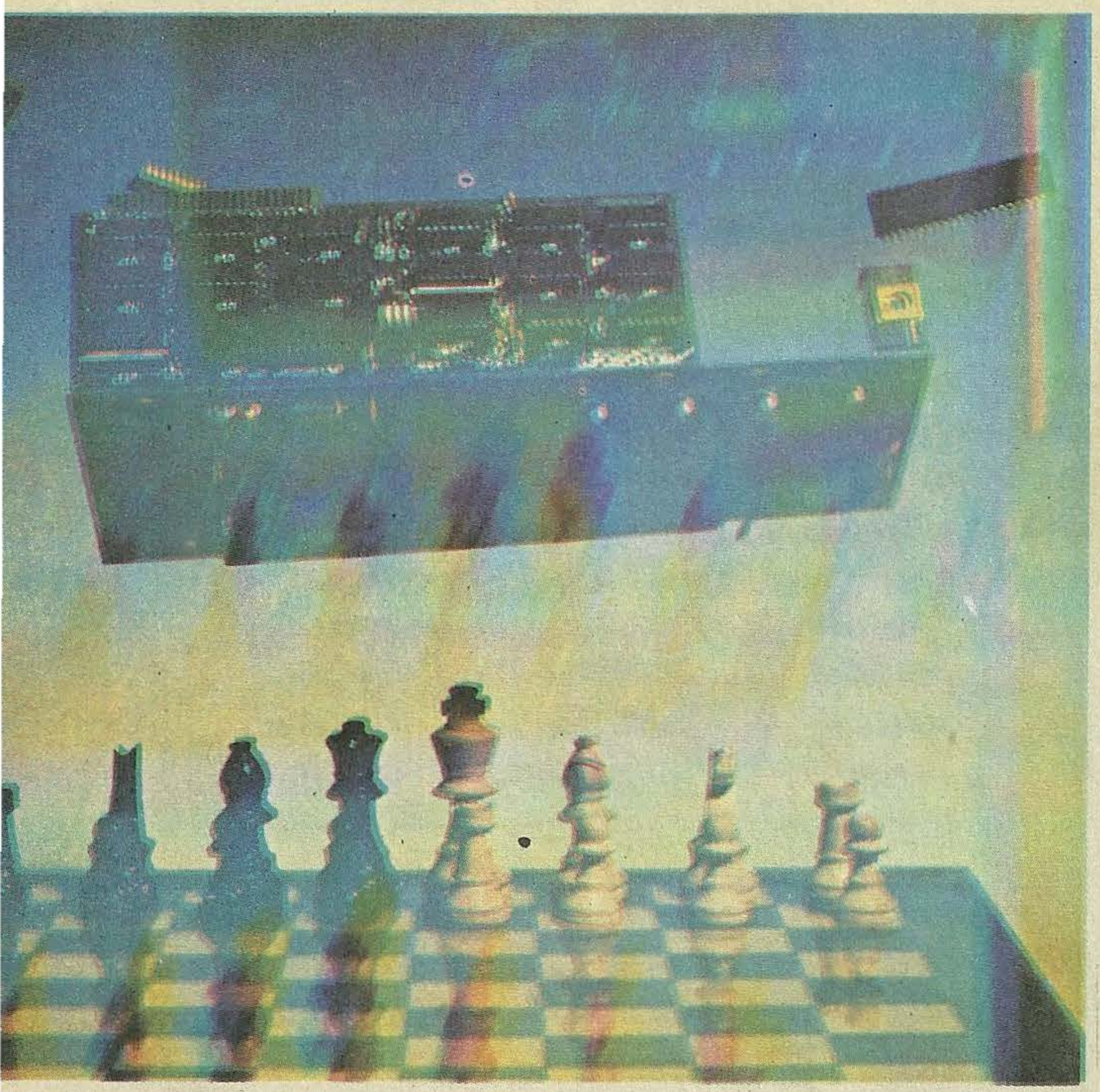
Schank'ın grubunun programları günlük yaşamın olaylarına yönelirken, şimdi anlatacağım bir diğer program da matematiksel kavramlarla uğraşıp, bunlardan, az veya çok ilginç sonuçlar üreten yeni kavramlar türetiyor. Program, genç Amerikalı Douglas Lenat tarafından, 1976 yılında Stanford Üniversitesi'nde yayımlanan doktora çalışmasının bir parçası olarak yazılmıştır. Lenat, ilginç matematiksel kavramlar ve teoremleri keşfetmeye yönelik modellerin araştırılmasıyla ilgilenmektedir. Bu yüzden program, çok sınırlı bir çıkarsama (deduction) yeteneğine sahiptir; hiç değilse teoremlerin kanıtlanması sözkonusu olduğunda. Programın fonksiyonu, bir dizi görevi, ''ilginçlik derecelerine'' göre sırayla yürütmek ve bunlara ilişkin kaynaklar veri işlemeye yetmediği anda hemen yapmakta olduğu görevi bırakmaktı. Doğal sayılar hakkında bir dizi keşif yapan AM adlı bir programa bunlar üzerinde hiçbir temel bilgi verilmemişti. (AM adlandırması herhalde Matematik-Automatik'e karşılık geliyor.)

Program, gerçek sayılar ve kesirler gibi kavramları bulacak durumda olmamasına rağmen, bunları tamamiyle kendiliğinden keşfetmiştir.

Programın bilgi için yararlandığı temel, o andaki koşullara göre ele alınması gereken yaklaşık 100 kavram ve 250 bulgusal (yeni bilgilere götüren) kuraldır. Bu kavramlar arasında, küme, liste (program listelere göre durum belirleyen LISP dilinde yazılmıştır), doğruluk tabloları, ilinti ve eşitlik gibi bağıntılar, evirtim, gruplandırma, derleme (kompozisyon) gibi işlemler bulunuyordu. Lenat, Piaget ve Copeland adlı psikologların araştırmalarını inceledikten sonra, derlediği bu kaynak kavramların dört yaşındaki çocukların düşündükleriyle aynı olduğu görüşüne vardı. Fakat ben bunun ne derece doğru olduğunu bilmiyorum. Program ampirik araştırmalar sonucunda çarçabuk, hiçbir sayının bölensiz olmadığını, yalnız birinin bir tek böleni olduğunu keşfetti. Fakat aynı zamanda yalnızca iki bölene sahip sayılar bulunduğunu (yani 1 ve sayının kendisi) ortaya çıkarttı. Böylece kavram dağarcığına yalnız iki bölene sahip sayı, yani asal sayı kavramını da ekledi.

Ayrıca AM'in, Lenat (ve de benim) için yeni olan, fakat yüzyılımızın ilk yıllarında genç bir deha olan Hintli Romanujan tarafından keşfe-

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



Satranç programları yapay zekanın doruk noktalarıdır.

dilen, sayılarla ilgili yeni kuramsal bir kavram bulabilmiş olması da ilginçtir.

Bu kavram adeta asal sayıların zıttı olan ''en fazla birleşik'' sayılar kavramıdır. Bunlar, kendisinden önce gelenlere göre daha çok sayıda böleni olan sayılardır. Buna göre ilk birkaç ''en fazla birleşik'' sayı, bir böleni olan 1, iki böleni olan 2, üç tane bölene sahip ilk sayı olan 4, dört böleni olan 6 vb.'dir. AM yalnızca doğal ve asal sayıları (birden fazla yöntemle) keşfetmekle kalmamış, her doğal sayının asal sayıların yalnız bir yolla çarpımıyla ifade edilebileceğini öne süren ''asal faktörler teoremi''ni de ortaya çıkarmıştır.

Daha sonra, Goldbach'ın (bugüne kadar kanıtlanmamış olan) her çift sayının iki asal sayının toplamı olduğu yolundaki önermesini de elde etmiştir.

Bu ölçüde dikkate değer başarılara, başarılardan aşağı kalmayan ilginç hatalar eşlik etmektedir. Programın, kesirleri algılayabilecek yetenekte olmadığını daha önce belirtmiştim. Bu nedenle, De Morgan gibi basit bağıntıları bulmasına rağmen, cümleler teorisinde büyük aşamalar yapamamıştır. İki asal sayının yalnız bir yolla toplamından elde edilen sayılar kavramıyla uğraşırken, sık sık anlamsız işlemler yürütmüştür. Lenat, doktora tezinde AM'in 25 iyi (''başarılı''), 200 kabul edilebilir ve 60 ''muğlak'' kavram keşfettiğine karar vermiştir.

Ben iddiamı kanıtlamak için nispeten yeni programlardan üç örnek (tavla oyunu, doğal dilin anlaşılması ve matematiksel keşifler) verdim. İddiam, biraz önce sözünü ettiğim yüksek seviyedeki programların -kelimenin olumsuz anlamıyla- mekanik değil, aksine insan ruhuna özgü yaratıcılık, sezgi, zeka gibi özellikleri taklit eden süreçler izleyen programlar olduğudur. Bunun ötesinde tıbbi teşhislerden, ''görsel algılama''dan, programların hassasiyet kontrolünden, IQ testlerinden, öğrenim süreçlerinden de inandırıcı örnekler verebilirdim.

Bilgisayar bilimi, bizleri, örneğin Assembler (çevirici) ve makine dilleri oluşturan ve geliştiren, fakat iletişim amacı taşımayan birtakım dillere yöneltmiştir. Son zamanlarda, Aaron Sloman'ın dikkat çekici bir felsefi çözümlemesi, gerçekte dilin asıl fonksiyonunun bu olduğu ve iletişim olanağının bunun bir ikinci ürünü olduğu fikrini doğurmuştur. Şunu belirtmek gerekir ki, hayvanlar ve uzak geçmişteki atalarımız birtakım şeyleri ifade etmek ve işlemek için dillere sahiptiler. Bu diller, daha sonra bulunan ses ve şekillerin yardımıyla, tabiri caizse belirgin hale geldi.

İnsan psikolojine dönecek olursak; kendini özellikle sinir sistemindeki olaylarla sınırlayan, işlemeye yönelik bir içsel dilin (ifadenin) varlığı reddedilmediği takdirde, içsel dillerle, benim ''dışsal'' diller diye adlandırdığım (örneğin İngilizce, Türkçe vb.) arasında net bir ayrım yapmak, şimdiye kadar bildiklerimizi daha emin ve daha birebirş ekilde yansıtmaz mı? Bence hayır. Gerek netlik nedenlerinden gerek bu tür bir ayrım, olgunun standart bir bakışla zımni açıklanma olanaklarını azaltacağından bu ayrımı reddediyorum.

Bu konuda, insanların öğrenme süreçlerinin özellikle daha çok sayıda formel simge kullanıldığında, nasıl geliştiği olgusundan hareket ediyorum. İçsel ve dışsal diller (ifadeler) arasındaki sıkı ilişki ve sürekliliğe örnek olarak, çocukluğumuzda temel aritmetiği öğrenirken geçirdiğimiz değişiklikleri ele alabiliriz. Bir toplamın ne olduğu bize ilk kez öğretildiğinde, bu işlemleri öğretmenimizin öğrettiği kavramlarla yaparız. Bu sırada bilinçli olarak doğal dilin

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

sözcüklerini ve cümlelerini kullanırız. Bunlara, ''iki kere üç altı eder'' veya ''elde var'' gibi örnekler verilebilir. Fakat zamanla biraz daha yatkınlık kazanıp, işlemleri otomatik olarak, yani ''düşünmeden'' çözmeyi başardığımızda, eskiden bellediğimiz kavramlardan ve deyimlerden eser kalmaz. Hiç kuşkusuz, bilinçsiz de olsa simgesel bir işlem yapmaktayız. Demek ki, aslında dışsal anlatım içsel anlatıma dönüşmüştür.

Bir anlatımdan ötekine geçişin ne kadar basit ve ''doğal'' olduğu görülüyor: Belli bir yatkınlığa eriştikten sonra, çok uzun ve zor bir toplamı yapmak sorunuyla karşılaştığımızda, bu olay bazen ters yönde işler. Bu kez anlatımın dışa yönelik olanına döneriz ve örneğin kendi kendimize şöyle konuşuruz: ''Şu sırayı topladıktan sonra, elde var 2 olduğunu unutmamalıyım.''

Bu nedenle şöyle bir varsayımı ileri sürmek istiyorum: ''Herhangi bir biyolojik sistemdeki zihinsel süreçler, o sisteme özgü içsel anlatım (dil) içinde oluşurlar. Bu anlatım her aşamada eylemselleştirilebilir (aktive edilebilir)''.

Bir veri işleme sisteminde aşamaların tasarımı genel olarak bilinmektedir. En üst aşamada, esas dilde yazılmış bir program işletileceği zaman, önce bilgisayarın Assembler diline çevrilmeli ve böylece değiştirilerek elde edilen sonuç, makine diline aktarılmalıdır. Programların işletilmesinde kullanılan bu normal yöntem, böyle hiyerarşik kademelere ayrılmış bir şemadan oluşur. Yine de birçok değişiklikler mümkündür. Örneğin bazı sistemlerde yüksek bir dille yazılmış programlara, (genelde verimliliği artırmak için) makine dili ile hazırlanmış bölümleri eklemek mümkündür.

Bu durumlar sözkonusu olduğunda veya yalnızca bunu yapmanın mümkün olabileceğinin düşünülmesi halinde bile, bilgisayarda farklı dil seviyeleri tasarımı anlamını kaybetmeyeb aşlar. O zaman programı bir tek dille yazılmış olarak kabul etmek daha faydalı olabilir (bu takdirde esas dil, yüksek düzeyli dil Assembler ve makine dili ve aralarındaki aktarımlar gibi tek tek bölümlerden oluşur).

Bu, benim iddia ettiğim gibi, biyolojik organizmaların tek tip diline tekabül edebilir. Bu dilin aktive edileceği (işlevselleştirileceği) ''aşama''lardan bahsettiğimde de, bu karmaşık dilbilimsel sistemdeki pozisyonları kastettim. Örneğin bir eşyayı görmediğimiz halde sadece adını duyduğumuzda gözümüzde canlandırabilmemiz olgusu bir açıklamayı gerektirir. Bu olgu, çeşitli duyu ve algılama olanaklarımızın çalışma biçimlerinin ortak bir dili doğurduğu ve bu yüzden aynı imgelerin farklı kaynaklardan da alınsa aynı dilbilimsel çerçeve içinde işlenebileceği varsayımıyla açıklanabilmektedir. Bu durum biraz önce bahsettiğim farklı aşamalardaki tek tip (standart) dilin nasıl aktive edileceğine dair bir örnek olabilir.

Bu bağlamda, şiir sanatına bir göz atmak ilginç olabilir. Çünkü şiirde sadece doğal dil kullanıldığı halde (özellikle nesir değil de nazım sözkonusu olduğundan) sanki, içsel dilin ''alt aşamalarını'' uyarmak amacı vardır. Bu yüzden şiir, normal dil kullanımının başaramadığı bir tepki uyandırabilmektedir. Müzik de konuşma dilini kullanamaz ve etkili olabilmek için de bu kanaldan geçmesi gerekmediğinden içsel dilimizi çoğu zaman daha etkin bir şekilde uyarabilmektedir.

Yapay zekanın yöntemlerinin, amaçları, beklentileri ve yorumları ifade etmesi için çok uygun olduğunu gördükten sonra, duyuların etkili modellerini saptamak çok güç olmasa gerek.

Freud'un, düş görme, şaka yapma ve dolaylama (bir kavramı tanımlama) sırasında olagelen bilinç dışı süreçlerin incelenmesindeki öncülüğü, bu konuda uzman olmayan yazarın kanısınca, bilimsel açıdan hakettiği oranda değerlendirilmemiştir. Düş görme ve şaka yapma sırasında olagelen, sözünü ettiğim değişimler, örneğin yorumlamada, derlemede ve bilgisayar tiplerinde kullanılan başka yöntemlerde olduğu gibi uzun zamandır bilinen bilişsel dönüşümler çerçevesine yerleştirilebilirler. Bu durumda, yeterli bir malzeme ile donatıldığında en parlak esprileri, şakaları üreten bir program yazmak hiç de güç olmamalı?

B.Meltzer'in ''Informatica e psicologiea'' adlı yazısından. QUADERNI DI INFORMATICA, 7. yıl, Nr.2, 1981, Honeywell Information Systems Italia.

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

zaman, yapılmakta olan işlemi durdurur (yani BREAK tuşunun işlevini görür). Ancak yukarıdaki komutta noktalı virgülden sonra gelen ''!'' karakteri; ESC kodunu yeniden tanımlamaktadır. Daha açık bir deyişle CHR$(27); ''!'' sırası bilgiyazara yazı düzeninin değiştirileceğini haber vermektedir (bkz. s.595). Bunu izleyen CHR$(1) ise hangi karakter biçiminin istendiğini belirtmektedir. Örneğin burada 1 yerine 0 kullanıldığı takdirde (CHR$(0)), bu sefer sayfa 582'deki şekilde yukarıda görülen karakter biçimi elde edilir.

Çubuk diyagramların çizimi

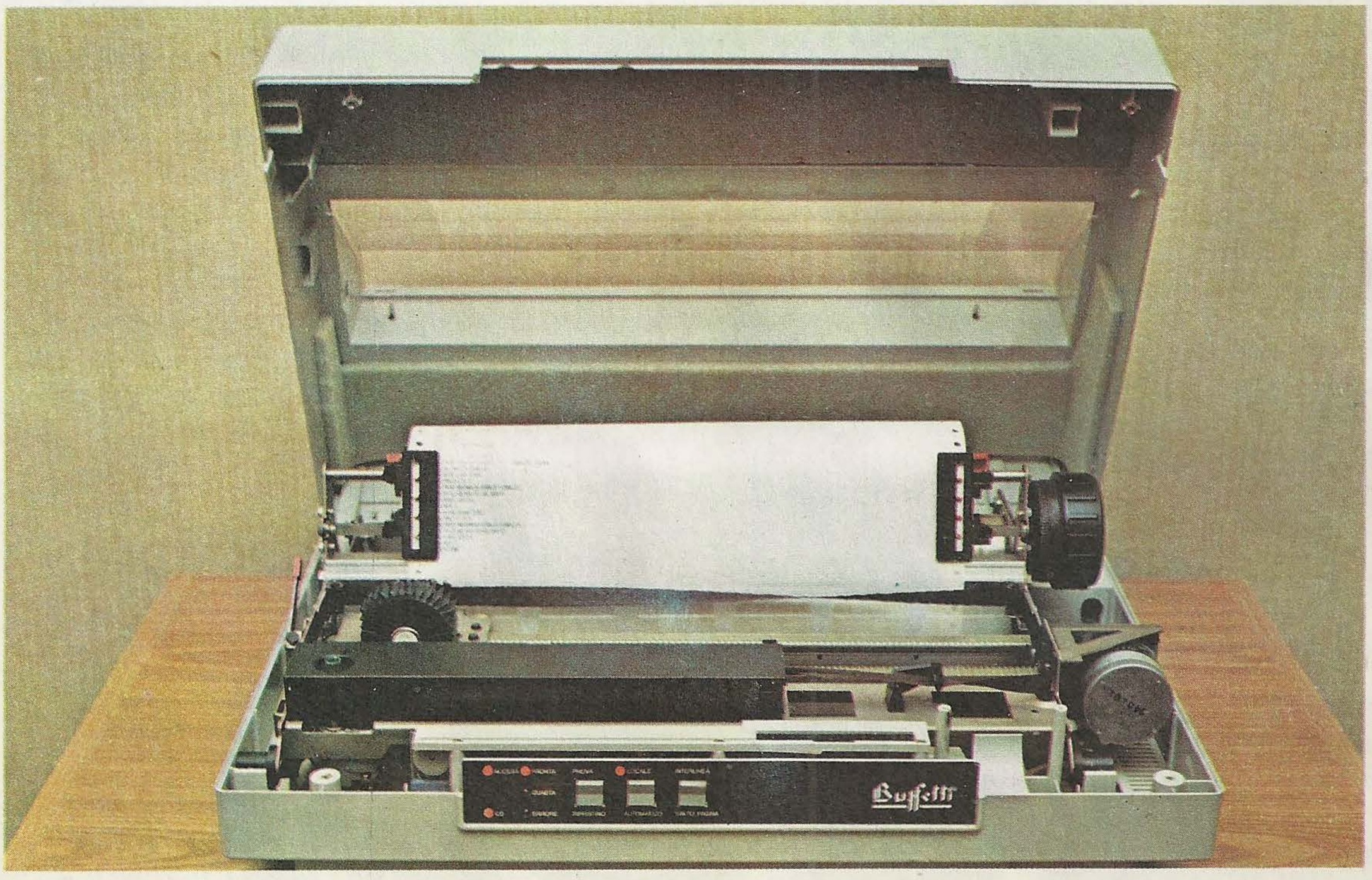
İster bilimsel ister ticari olsun bir işin planlanmasında verilerin çizisel birçok gösterimi gereklidir. Normal olarak bir oran veya bağıntının içeriğini ortaya koyan değerler, tekil çözümlemelerde kullanılabilmek için tablo olarak verilirler. Yine bir olgunun genel özelliklerinin belirlenmesinde veya hızlı bir genel değerlendirme için çizisel bir gösterime gereksinim duyulur. Bu konuya sonradan tekrar döneceğiz, fakat şimdi histogramların oluşturulmasında kullanılan bilgiyazarlarla ilgili yöntemleri göstermek istiyoruz.

Histogram, bir tablonun çizisel bir gösterimidir. Örneğin biz belli sayıda satıcının yaptığı satışlarla ilgili istatistiksel bir araştırma yapmak istiyorsak, aşağıdaki tabloyu elde edebiliriz:

SATICI	SATILAN MİKTAR
A	12
B	7
C	9
D	10
E	2
F	8

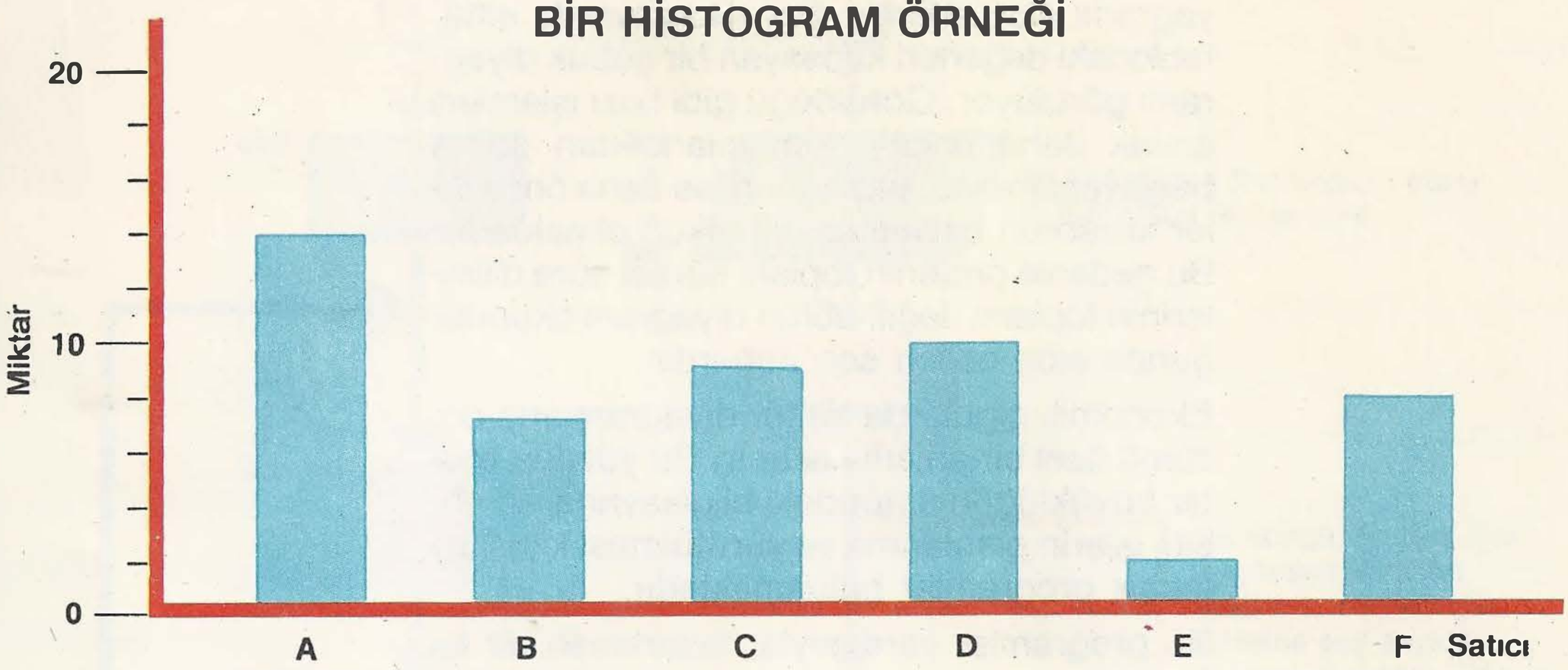
Her satıcının kendine ait bir satış miktarı vardır. Bu veriler sayfa 589'daki gibi çizisel bir şekilde tabloya aktarıldığı takdirde genel durum ortaya çıkar. Burada her bir harf bir satıcıya karşılık olmaktadır (A ve B gibi). Bunlar fazla sayıda değildir. Yani iki nokta arasında (A ile B gibi) başka değerler bulunmaz. Böyle durumlarda çizisel görünüm bir eğrinin çizisel görünümüne hiç benzemeyen bir histogram oluşturur. Bu iki çizisel görünüm türü arasındaki en temel fark, ikisinin tanım alanları arasındaki farktır. Bir eğride iki nokta arasında daha birçok noktalar bulunur (özel durum-

Buffetti marka bir bilgiyazar.

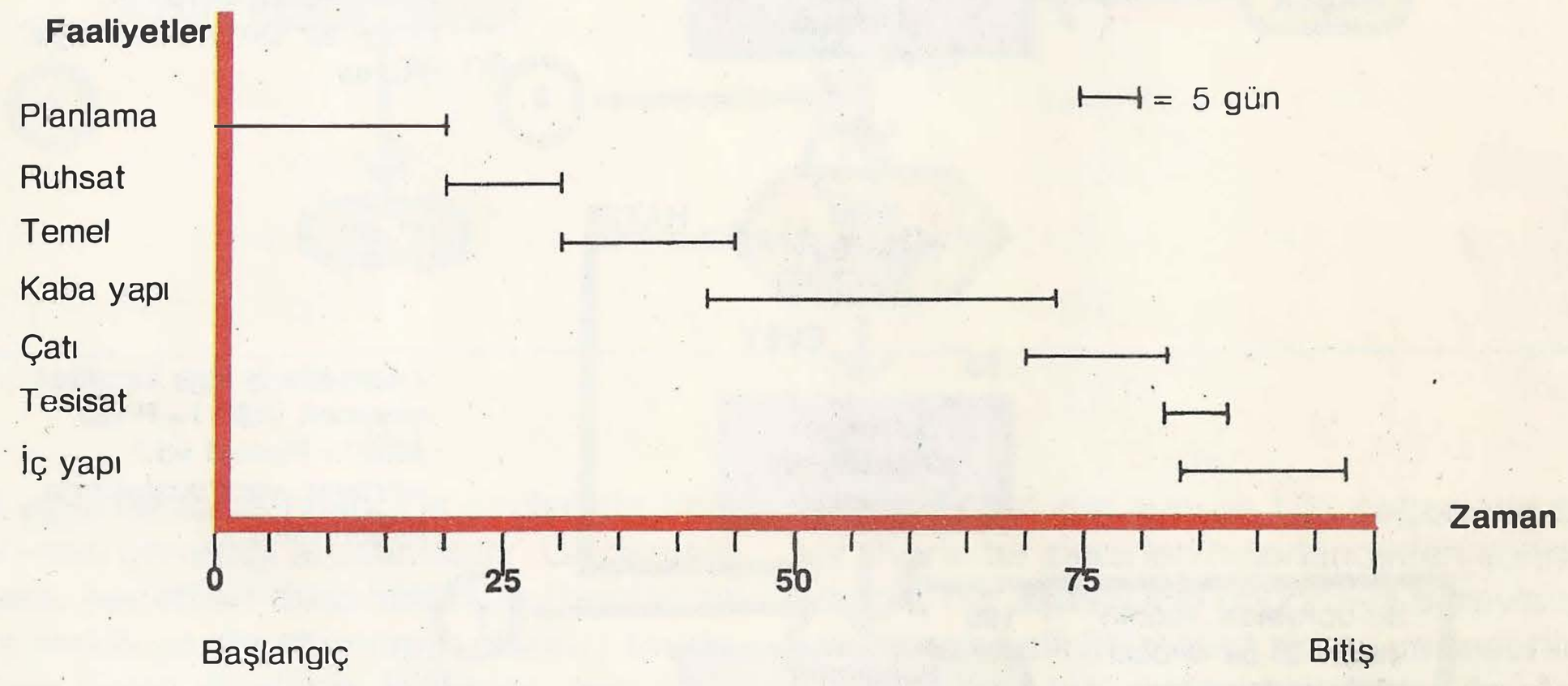


Ücretsiz PDF Kitap Arsivi - https://rantey.org

BİR HİSTOGRAM ÖRNEĞİ



BİR EVİN İNŞASINDA YAPILACAK İŞLER İÇİN ZAMAN DİYAGRAMI



lar dışında). Bir histogramda ise olası değerlerin miktarı sınırlıdır. Histogramlara benzeyen başka bir çizisel gösterim türü de çubuk diyagramlarıdır.

Her üretim planının başlangıcında uygulama zamanlarının bir çözümlemesi yapılmalıdır. Plan, birbirine bağımlı ve her biri için belirli bir süre ayrılmış olan bir dizi süreçten meydana gelir. Uygulama için gereken toplam süre, süreçlerin zaman bakımından uzunluklarına bağlıdır. Dolayısıyla plan aşamasında, her bir işin ne zaman yapılacağının ve ne kadar süreceğinin çizisel olarak gösterilmesi gerekir.

Örneğin bir evin inşası için ne kadar zaman gerektiğini tespit etmek istediğimizi düşünelim. Başlıca çalışma aşamaları aşağıdaki gibi sıralanabilir:

İŞ	SÜRE
Planlama	20 gün
Ruhsat	10 gün
Temel atma	15 gün
Kaba yapı	30 gün
Çatı	12 gün
Tesisat	8 gün
İç yapı	16 gün

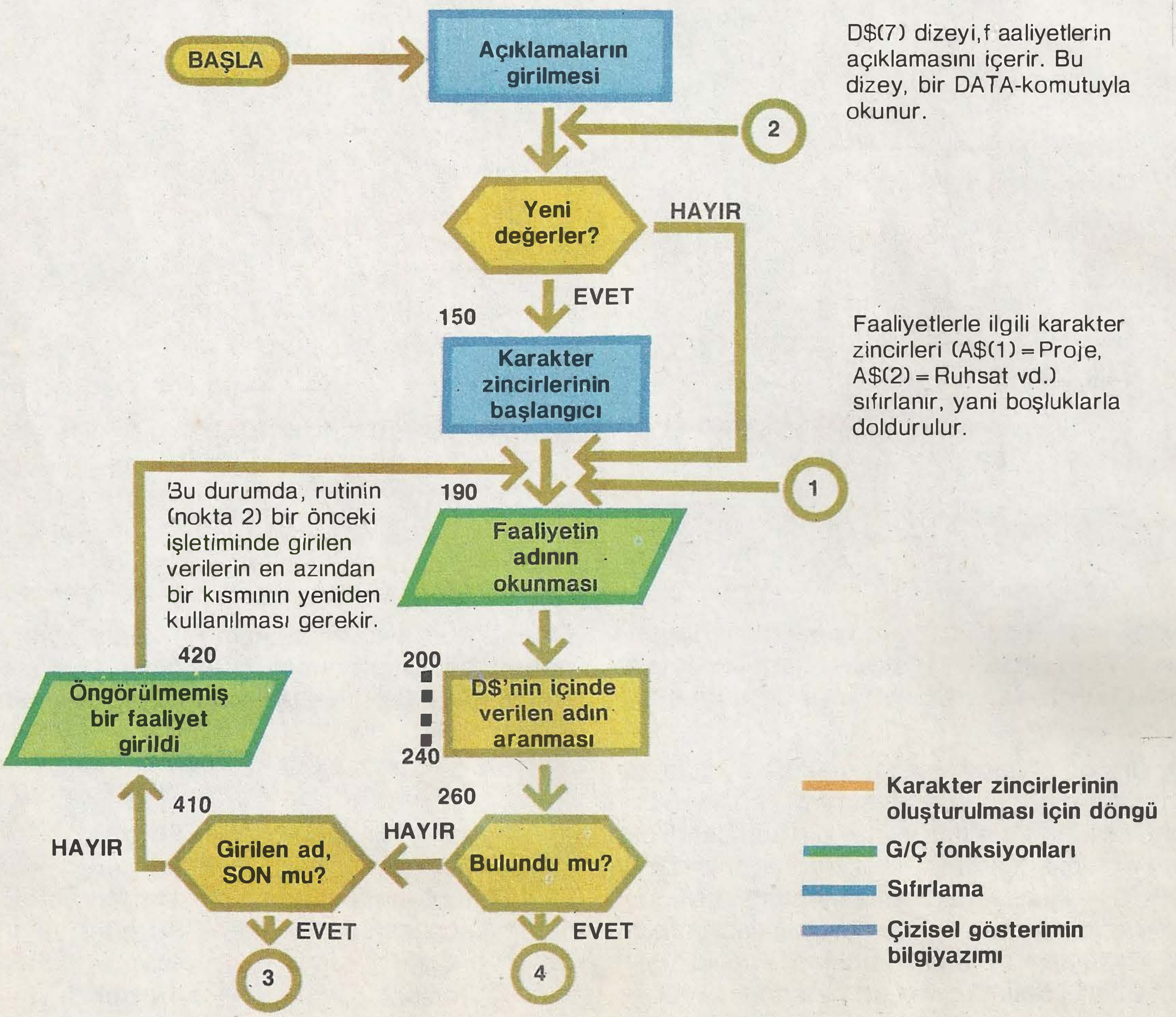
Ücretsiz PDF Kitap Arsivi - https://rantey.org

Farklı açıklamalar ve bunlara ait süreler bir çizisel gösterime aktarılırsa, projenin çubuk diyagramı elde edilmiş olur. Bu sayfada altta, tablodaki değerleri kapsayan bir çubuk diyagramı görülüyor. Görüldüğü gibi bazı işlemler, ancak daha önceki tamamlandıktan sonra başlayabilmekte, bazılarının ise daha önce işler sürerken başlaması mümkün olmaktadır. Bu nedenle projenin toplam süresi süre dilimlerinin toplamı değil, bütün diyagram okunduğunda elde edilen son değerdir.

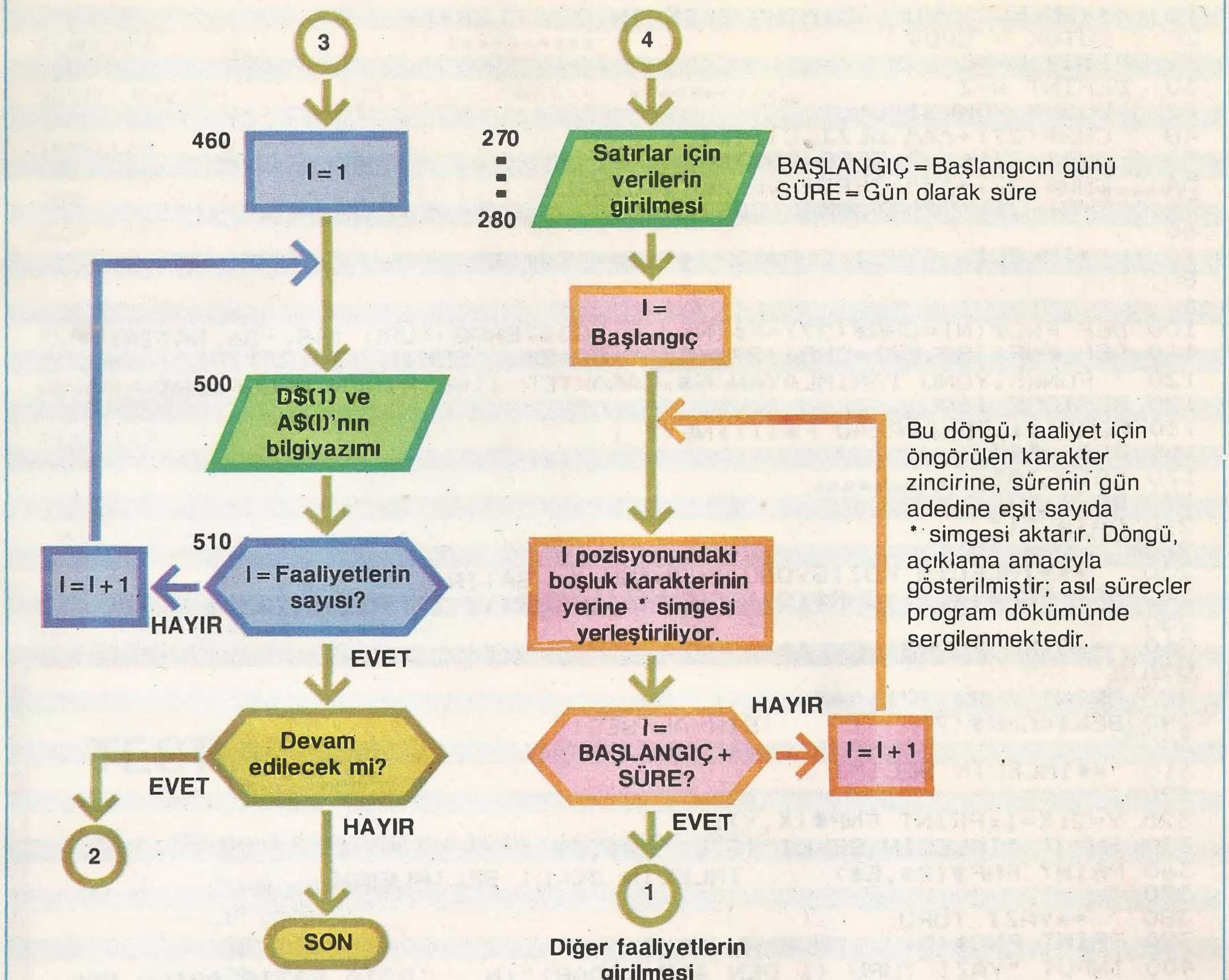
Ekonomik olgularda bu tür diyagramların çözümü özel bir anlam kazanır. Bu yüzden belli bir büyüklüğün üstündeki bilgisayarlarda çeşitli işlerin planlanma ve yürütülmesi için karmaşık programlar bulunmaktadır.

Bu programlar yardımıyla, tasarlanan bir işteki olası gecikmelerin tüm proje üzerindeki et-

ÇUBUK DİYAGRAMLARININ BİLGİYAZIMI İÇİN PROGRAMIN AKIŞ ŞEMASI



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



kileri veya ne zaman hangi iş güçlerinin hazır bulunması gerektiği araştırılabilir. Gecikmelerin olası nedenleri düşünülürken (bunlar zamana ilişkin ya da ekonomik olabilir) bu durumlara ilişkin düzeltme tedbirleri uygulanabilir.

Çubuk diyagramlarını bilgisayar kolaylıkla oluşturabilir. Bunun yöntemi, öngörülen her işlem için, uzunluğu en azından projenin süresi kadar olan birer karakter zincirinin basılmasıdır. Projenin süresini gösteren uzunluk uygun bir ölçek ile bölümlendirilmelidir. Başlangıçta karakter zinciri boşlukla doldurulur, program süreci içinde her işlem için öngörülen zaman dilimine istenen herhangi bir simge konur.

Örneğin sayfa 589'daki çizisel gösterim yedi tane işlemi gözönüne almaktadır. Yani yedi karakter zincirine ihtiyaç vardır. Başlangıç uzunluğu her biri için aynı ve 130 değerinde olabilir. Karakter zincirleri hazırlandıktan sonra her birine, her aşama için öngörülen süreyle orantılı olarak belli bir miktar simge yerleştirilmelidir. Faaliyet türü için zaman olarak öngörülen ölçüye uygun sayıda simgeyle, bu 130 karakterden bir kısmı doldurulmalıdır.

Aynı çizisel gösterimi daha netleştirirsek, ilk karakter zinciri (planlama) en başta bulunan 20 yıldızı içermelidir (20 gün süre). Buna karşılık sonuncusu (iç yapı), -örneğin- 82. pozisyondan itibaren 16 yıldız içermelidir (burada, iç yapı işleminin 82. günde başlayabileceği düşünülmektedir).

Sayfa 590 ve bu sayfanın üst kısmında bir çubuk diyagramını üreten ve sayfa 589'dakine benzeyen bir programın akış şeması verilmektedir. Buna tekabül eden döküm ve çıkış, sayfa 592-593'te verilmiştir.

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

İMLECİN KULLANIMINA İLİŞKİN ÖRNEKLER

```
10 '***IMLECIN KULLANIMINA ILISKIN ORNEKLER***
20 ' KUTUK = SVDV
25 OPTION BASE 1
30  DEFINT A-Z
42 ' GUDUM FONKSIYONU
48 ' CHR$(27)+"4" ALTI CIZILI
50 ' CHR$(27)+"0" TERS GOSTERIM
54 ' CHR$(27)+"5" TERSI VE ALTI CIZILI
56 ' CHR$(27)+"1" NORMAL GOSTERIM
60 '
70 ' **IMLECIN SEKLI:CHR$(27)+"c4"+"S"+"E"
80 '
90 '
100 DEF FNC$(N)=CHR$(27)+F$(N)      GOSTERGE TURU (48.-56.SATIR)
110 DEF FNF$(S$,E$)=CHR$(27)+"c4"+S$+E$   'SEKIL (70.SATIR)
120 ' FONKSIYONU TANIMLAYAN F$ KARAKTER ZINCIRININ DEGERLERI
125 RESTORE 160
130 FOR I=1 TO 4:READ F$(I):NEXT I
160 DATA "4"
170 DATA "0"
190 DATA "5"
210 DATA "1"
220 '
230 ' ***IMLECIN POZISYONU:Y=SUTUN X= SATIR
240 DEF FNP$(X,Y)=CHR$(27)+CHR$(61)+CHR$(Y+31)+CHR$(X+31)
250 '
260 '**OZEL FONKSIYONLAR**
270 '
280 PRINT CHR$(27);"*"
290 BEL$=CHR$(7)          'SINYAL SESI
300 '
310 '**IMLECIN SECIMI
320 '
325 Y=2:X=1:PRINT FNP$(X,Y);
330 INPUT "IMLECIN SEKLI (S,E)";S$,E$
360 PRINT FNF$(S$,E$)     'IMLECIN SEKLI BELIRLENDI
370 '
380 ' **YAZI TURU
390 PRINT FNC$(5)    'NORMAL YAZI
400 INPUT "YAZI TURU (1 DEN 4 E KADAR)";N    'DATA SATIRLARINA BAK
410 IF N>=1 AND N<=4 GOTO 520
420 '***HATA****
430 '
440 Y=35:X=12:PRINT FNP$(X,Y)    'IMLEC EKRANIN ORTASINDA
450 N=6:PRINT FNC$(N); "**HATA**"
460 FOR I=1 TO 800:NEXT I    'OKUMA ICIN BEKLEME DONGUSU
470 '****BILGIYAZIM KOMUTLARI****
480 LPRINT YOR$:LPRINT :LPRINT
490 FOR I=1 TO 7
500 LPRINT D$(I),A$(I)
510 NEXT I
511 '**ZAMAN CETVELI YAZILIYOR**
512 LPRINT SPACE$(14);
520 '
530 '
540 INPUT "DEVAM(EVET/HAYIR)";CEV$
550 IF CEV$="EVET" GOTO 115
570 END
```


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

```
CUBUK DIYAGRAMLARI DENEMESI

FLANLAMA          **********
DENEME              *****
HESAFLAMALAR              **********
HAFRIYAT                    *****
TEMELLER                      **********
KABA INSAAT                     *****
DAM                                 **********
```

```
                CUBUK DIYAGRAMLARI DENEMESI:2

FLANLAMA          **********.
DENEME              ********
HESAFLAMALAR              **********
HAFRIYAT                    ******
TEMELLER                      **********
KABA INSAAT
                          ******************
DAM
                                    ********
```

TEST 17

1 / Bir iğneli bilgiyazar aşağıdaki özelliklere sahiptir:

— İğneli yazım kafası
— Çift yönlü
— 132 sütun
— 120 cps
— 4 K'lık ara-bellek

Bahsedilen veriler ne anlama gelir?

---

2 / Aşağıdaki güdüm karakterleri bilgiyazara verildiğinde, hangi karakterlerin basılacağını tespit ediniz:

CHR$(33) CHR$(110) CHR$(27) CHR$(10)

---

3 / Aşağıdaki programdan elde edilecek çıkışlar nelerdir?

```
10 A$=''DENEME'':B=1270:C=10
20 PRINT A$, C
30 PRINT USING ''          ''; A$;B
```

---

4 / Sayısal A(5) dizeyinin bilgiyazımı aşağıdaki komutlarla yapılmaktadır:

```
10 FOR I=1 TO 5 : LPRINT A(I) : NEXT I
20 FOR I=2 TO 5 : LPRINT A(I) ; : NEXT I
30 FOR I=1 TO 5 : LPRINT A(I) , TAB (10); NEXT I
```

Buna göre veri çıkışları ne olur?

Test 17'nin çözümleri sayfa 599'dadır.

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

Ekran Terminalinin Kullanımı

Kişisel bilgisayarların standart kullanımlarında, giriş ve çıkışlar için mutlaka bir ekran terminali bulunur. Bu arada bu tür çevresel gereçlerin kullanımının rahatlık ve elverişliliği de herkesçe anlaşılmış ve küçük bilgisayarlar ile merkezî birimler de (main frames) ekran terminalleriyle donatılmıştır.

Ekran birimlerinin tüm imkanlarıyla kullanımları, insanla olan diyalogu kolaylaştırmak için ''konuşma'' özellikleri sağlamaları açısından, bir bilgi işlem sisteminin verimini gözle görülür ölçüde artırır. Ekran birimlerinin en verimli biçimde işletimi, kullandığı makinenin sunduğu denetim olanaklarına ve bu olanaklardan yeteneği ölçüsünde yararlanması gereken programcıya bağlıdır.

Ekran birimlerinin mükemmel kullanımı, tanım ve verilerin girişinde uygun alt-programlar kullanmak gerektiği sürece karmaşık programlama çalışmalarında pek değerli olanaklar sunmaktadır.

Bir ekran terminalinin yapısı ve özellikleri

Tipik bir ekran biriminin fonksiyon şeması, bu sayfada altta gösterilmiştir. Bundan sonra en önemli öbekler ve bunların işlevleri açıklanacaktır.

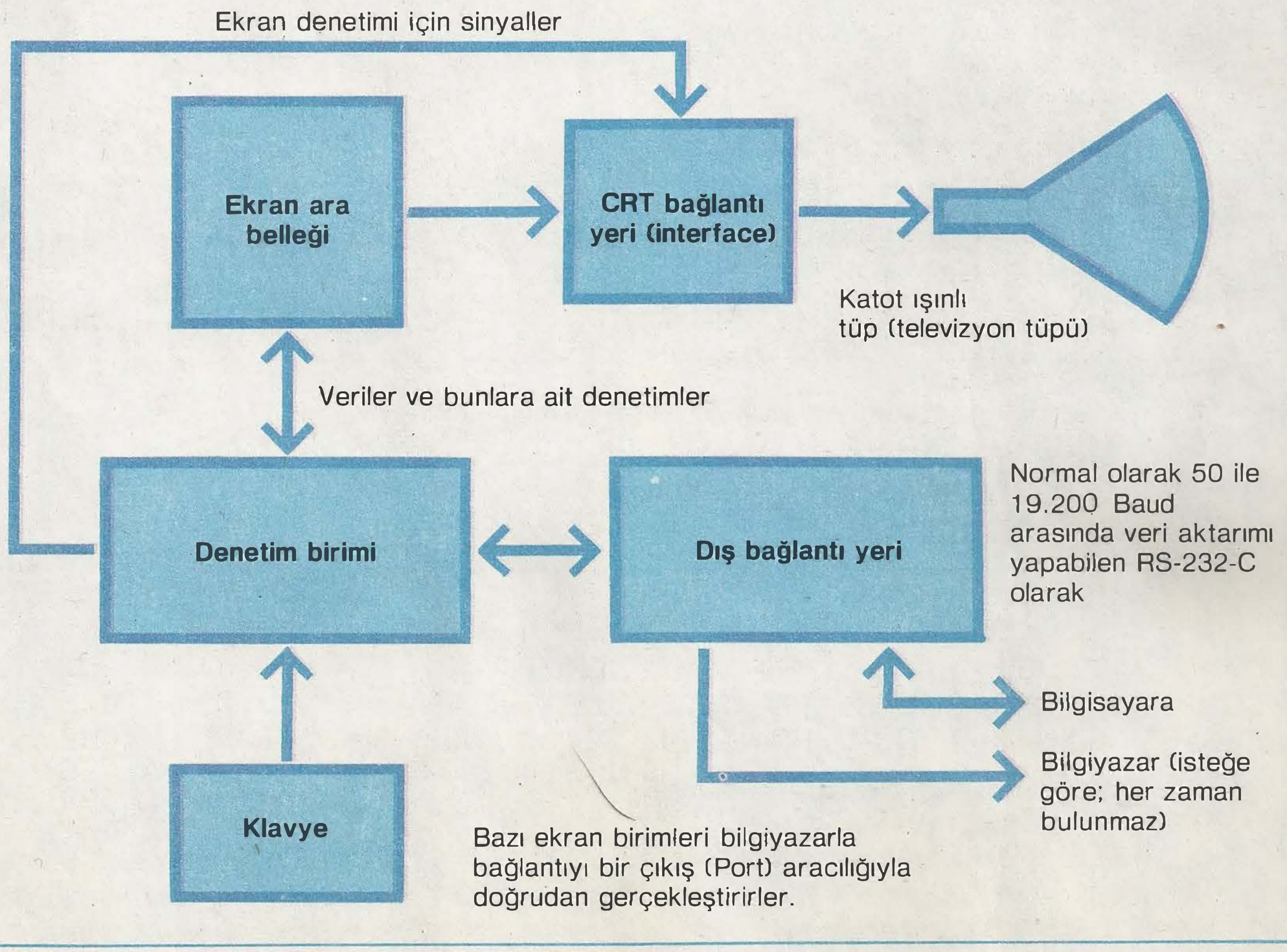
☐ **Klavye (keyboard):** Bu giriş organıdır. Bir tuşa basılarak denetim birimine aktarılacak olan uygun bir ASCII kodu üretilir.

☐ **Denetim Birimi:** Öbür birimleri denetler. Bilgisayarla olan ilişkiyi ve komutların yorumlanıp yerine getirilmesini sağlar. Görev erimi MİB'inkine benzer.

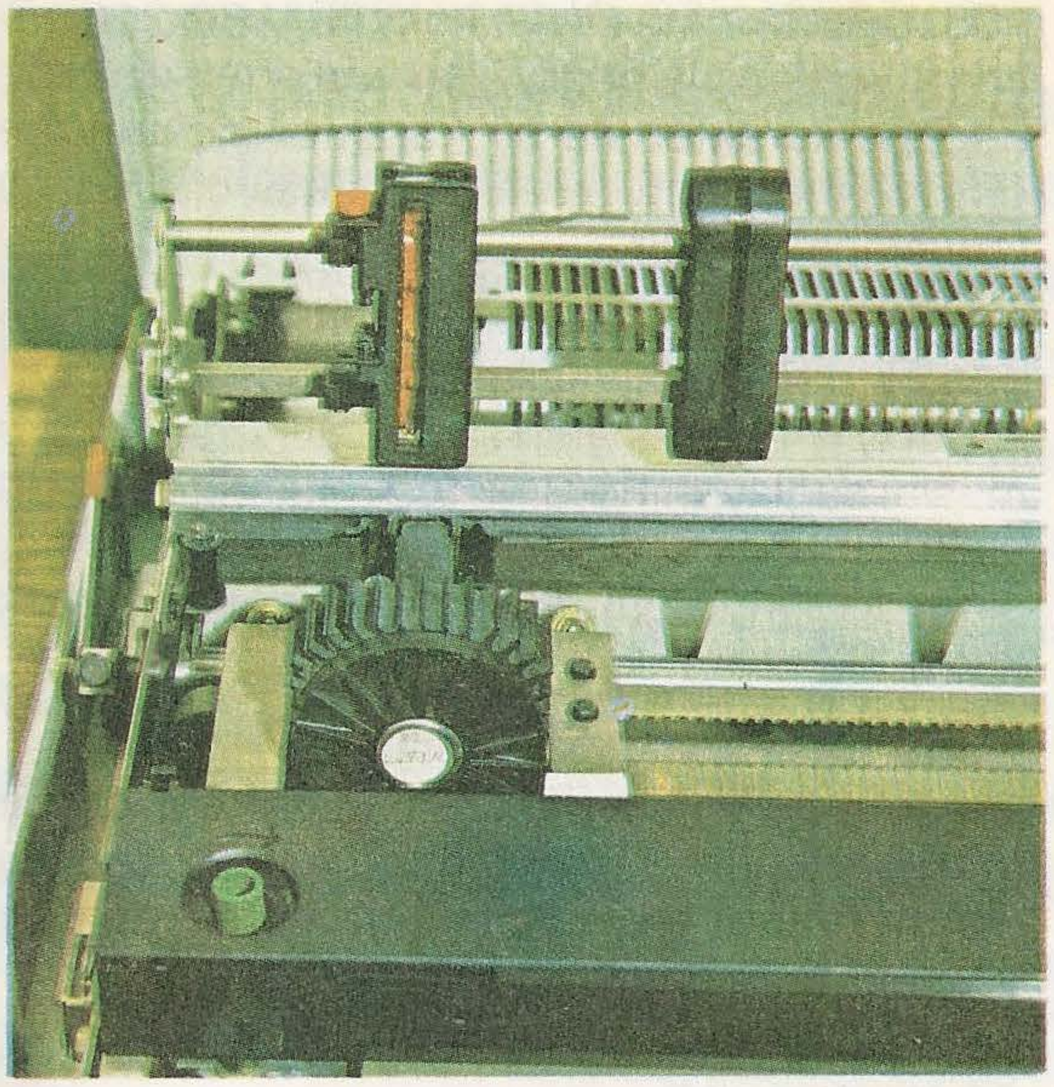
☐ **Dış Bağlantı:** Bilgisayarla, bir bilgiyazar arasındaki iletişimi denetler. Bazı ekran birimleri bilgiyazarla terminali doğrudan bağlayan ''çıkış''lar (portlar) üzerinden çalışırlar. Bunların yüksek vasıflı olanları ekranda bulunanların tümünü bilgisayarda herhangi bir işleme sokmadan bilgiyazara iletebilirler. Bu bilgiyazım tekniğinin ismi ''hard copy''dir.

☐ **Ekran Ara Belleği:** Burada görüntülenecek olan bütün bilgileri içeren bir bellek (RAM) sözkonusudur. Bu belleğin kapasitesi, ekran

EKRAN TERMİNALİNİN İŞLEYİŞ ŞEMASI



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



Buffetti marka bilgiyazarın içinin görünümü.

üzerinde bulunabilecek karakterlerin sayısı kadar olmalıdır. Örneğin, 80 sütun × 20 satırlı bir erkanı doldurmak için 1920 (80 × 24) karaktere gerek vardır, o halde böyle bir ekranın RAM-belleği 1920 pozisyon sunabilmelidir.

Belleğin yeteneklerini tanımlamak için burada ''pozisyon'' sözcüğü kullanılmıştır. Alışılmış olduğu üzere baytlardan bahsedilmemiştir, çünkü bu özel durumda bir bellek pozisyonu bir bayta tekabül etmemektedir. Ekranda gözükecek her karakter için ASCII kodu (8 bit) veya karakterin yazılacağı tarz (normal, ters, altı çizili vb.) hakkında açıklama yapan ek kodlar yani yazım nitelikleri saklanmalıdır.

Bu fonksiyon için normal olarak yine dört bit gereklidir. Dolu bir bellek alanının uzunluğu, bilgisayarın ve ekranın türüne bağlıdır. Yerleştirilen değerler en çok kullanılacak olanlardır. Bununla birlikte piyasada rastlanan terminaller arasında çok büyük farklar olabilir (burada sözü edilenler, grafik terminallere de maledilmemelidir, bunlar daha sonra ayrıca incelenecektir).

☐ **CTR-Ara Birim (Interface):** Karakterlerin görüntülenmesini sağlayacak olan ekran sinyallerini alır ve aynı anda bunları ekrana iletir. Ekran biriminin durumuna göre yapılan uygulamalara bağlı pek çok fonksiyon çeşidi vardır. Bunların en önemlileri aşağıda sunulmuştur:

CONVERSATIONAL: Her komut, merkezî bilgisayara (host computer) aktarılır. Bu işlem normal olarak RS-232 bağlantı yeri.aracılığıyla gerçekleşir. Bu yöntemde iki durum öngörülmüştür:

kısmi çift yönlü (half duplex): Terminal, klavye aracılığıyla verilen (ve hatla aktarılan) ve merkezî bilgisayarın yönetiminden gelen bütün komutları yürütür.

çift yönlü (full duplex): Klavye aracılığıyla verilen komutlar doğrudan yürütülmez, çıkış hattına verilir. Buna karşılık merkezî bilgisayardan gelen komutlar (yani hat üzerinde zıt yönde hareket edenler) yerine getirilir. Terminal tamamıyla bilgisayarın kontrolü altındadır.

SAYFA (PAGE): Ancak çok özel uygulamalarda kullanılan bu yöntemde, ekran için tasarlanan bütün karakterler yazılabilir, merkezî bilgisayara aktarma ancak bir sayfanın sonunda mümkün olur. BASIC'teki veri giriş fonksiyonları, bu işletim tarzının kullanımı için tasarlanmamıştır. PAGE konumunun kullanılması, ancak, bu özel aktarım çeşidinin yürütülmesini olanaklı kılan uygun alt-programların yazılması ile (genellikle Assembler dilinde) sağlanabilir.

SCROLL: Çalışma şekli bir öncekinde olduğu gibidir. Yalnız ufak bir fark vardır; dolu bir ekranda karakterler girmeye devam ederse ilk satırlar kaybolarak yukarıya doğru kayar.

(PAGE-konumunda ekran doluysa imleç ilerlemeye devam etmez.)

PROTECT: Karakterlerin korunması özel bir kod ile sağlanır. Bu karakterlerin kullanıcı tarafından değiştirilmesi mümkün değildir. (İmleç, korunmuş olarak tanımlanan karakterlerde durmaz.) Ekranın fonksiyon çeşidi uygun kodların girişi ile program aracılığıyla değiştirilebilir. Bir kod iki veya daha fazla karakterden oluşur.

Bunlardan ilki, çevresel gereçlere kendisini izleyen karakterlerin yazılacak veriler olarak değil, komutlar olarak anlaşılması gerektiğini belirten ESC (escape = CHR$(27))'dir. Örneğin, bazı ekran istasyonlarında, diyalog işlemine geçiş için kullanılan kod C'dir. Bu karakter, yürütülmesi gereken bir komut ile beraber kullanılırsa, CHR$(27) + ''C'' dizimi (zincirli) verilmelidir.

İlk veri (CHR$(27)); ASCII kodunda 27'ye karşılık olan karakteri üretir ve bunu izleyen karakterin (C harfi) komut olarak okunmasını sağlar.

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

İmleç'in kullanımı

İmleçin şekli ve yazının türü de uygun komutlarla değiştirilebilir. Az sonra verilecek kodlar, belirli bir makine (Seletron 3410 terminalli bir Bufetti B801) için geçerlidir; başka bilgisayar tipleri için farklılıklar olabilir; fakat çoğu durumlarda kullanma şekli ve mantığı değişmez. Yalnızca (burada açıklanacak olan) bazı özel durumlarda, belli anlamlara sahip farklılıklara rastlanır, ama sonuç olarak bunlar da biçimsel farklılıklardır (komutun yapısı gibi).

İmleç, ekran çalıştırıldığında beliren 6 yatay ve 11 düşey noktanın oluşturduğu ışıklı bir dikdörtgen alandır. İmleci denetleyen komutlar, hem şeklini hem de konumunu değiştirmeyi mümkün kılar. Olası durumlar şunlardır:

— Işıltılı imleç
— Görünmeyen imleç
— Kırpışan imleç
(bazı makinelerde ışıldama frekansı değiştirilebilir)

İmleçin şeklinin değiştirilmesi toplam 6 satır ile 11 sütunun hangilerinin ışıldaması gerektiği saptanarak sağlanır. Komutların biçimi:

ESC + ''Parametre''

olmalıdır. Burada ESC; ESCAPE (CHR$(27)) karakteri ve ''parametre'' istenen fonksiyona tekabül eden kodların yazıldığı bir karakter zinciridir. Örneğin, burada incelenen makinede ışıltılı imlecin durumu ''c0'' koduna uygundur; imlecin yanıp sönmesi isteniyorsa ''c3'' kodu verilmelidir.

```
PRINT CHR$(27) + ''c0''
```

komutunun girişiyle bu komutu izleyen bütün yazılar için,

```
PRINT CHR$(27) + ''c3''
```

komutu verilene dek ışıltılı imleç elde edilir. Bu ikinci komutun verilmesinden yeni bir denetim güdümü girişine kadar imleç yanıp söner.

İmlecin şekli,

```
PRINT CHR$(27) + ''c4SE''
```

koduyla değiştirilebilir. Burada S, alanın hangi satırının bunun için ayrılmış noktalarca aydınlatılacağını, E ise alanın sütunlarını belirleyen karakterlerdir.

Yandaki sayfada birkaç çizisel fonksiyonu gösteren örnek bir program dökümü verilmiştir. Çizisel fonksiyonlar ESC(CHR$(27)) karakteri ve bir veya daha çok kontrol karakteriyle yönetilmektedir (yan sayfadaki programın 48-56. satırları ile karşılaştırınız).

100. satırda içine DATA komutu (160-210 satırlar) yardımıyla öngörülen kodların yazıldığı, Escape karakteri ile bir karakter zincirinden (F$(N)) meydana gelen bir fonksiyon (FNC$) tanımlanmaktadır.

İstenilen çizisel fonksiyon, FNC$(N)'e N'in uygun bir değerinin yazılmasıyla elde edilir. Örneğin N = 4 olarak alınırsa, bir alt çizginin oluşmasını sağlayan F$(3) = ''G1'' karakter zinciri kullanılır. Buna göre, yazıların altının çizilmesini sağlayan komutlar şöyledir:

```
N = 4
PRINT FNC$(N) = ''Deneme satırı''
```

Programda imlecin adreslenmesi için de bir fonksiyon gösterilmiş (satır 240). Bu fonksiyon herhangi bir yazının ekranın istenen bir yerinde belirmesini sağlar. Örneğin:

```
10 A$ = ''deneme''
20 X = 5:Y = 10
30 PRINT FNP$(X,Y); A$
```

komutları yazılmış olan ''deneme'' sözcüğünün 10. satırın (y = 10) 5. sütunundan başlayarak yerleştirilmesini sağlar.

Ekran maskeleri

Bir ekran birimi mevcutsa, verilerin girişinde kullanılan ve ''ekran maskeleri'' olarak adlandırılan belli programlar oluşturulabilir.

Ekran maskeleri kullanıldığı zaman, başka durumlarda gerçekleştirilmesi mümkün olmayan bir dizi denetim uygulanabilir. Her şeyden önce, kullanıcı istediği anda, daha önce girişi yapılmış veriler üzerinde de düzeltmeler yapabilir.

Normal olarak INPUT komutu kullanıldığında bu sağlanamaz, çünkü veriler, girişleri yapıldığı anda alınır ve programa aktarılır. Ekran maskeleri için program yazma, beraberinde birtakım zorluklar da getirir. Bu nedenle, örneklerin gittikçe zorlaştığı bu konuyu aşama aşama incelemek istiyoruz. Başlangıçta, bir tarihin girişini ve sınanmasını öngören bir altprogram inceleyeceğiz. Birçok kullanıcı programlarında tarih, tablolar için basit bir dayanak olarak veya ekonomik sebeplerle (muhasebe belgelerinin çıkışı), her defasında yeniden verilmek zorundadır. Her iki durumda da, girilen tarihin geçerliği sınanmalıdır. Bizim amaçlarımız açısından bu sınamanın bir ''ay''ın bulunduğunun ve gün için verilen değerin bu ay için mümkün en üst değeri aşmadığının denetlenmesine kadar indirgenmesini kabul edebiliriz. Bir sonraki sayfada sınama altprogramının akış şeması verilmiştir.

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

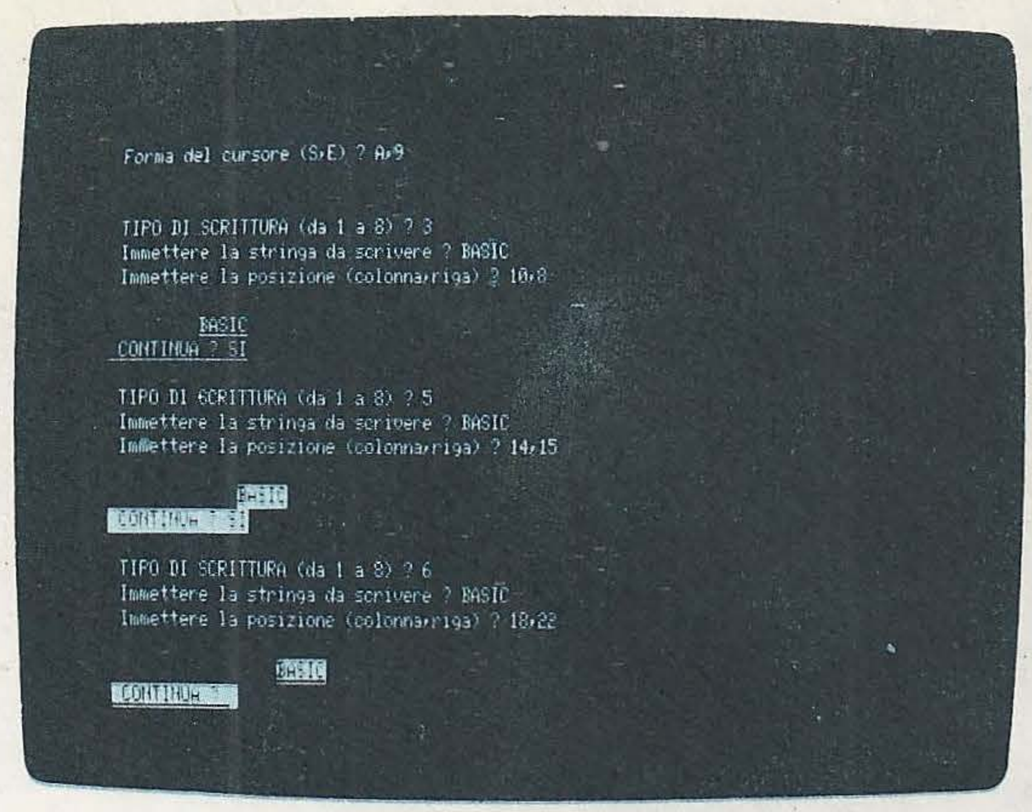
ÇUBUK DİYAGRAMLARININ BİLGİYAZIMI İÇİN PROGRAM

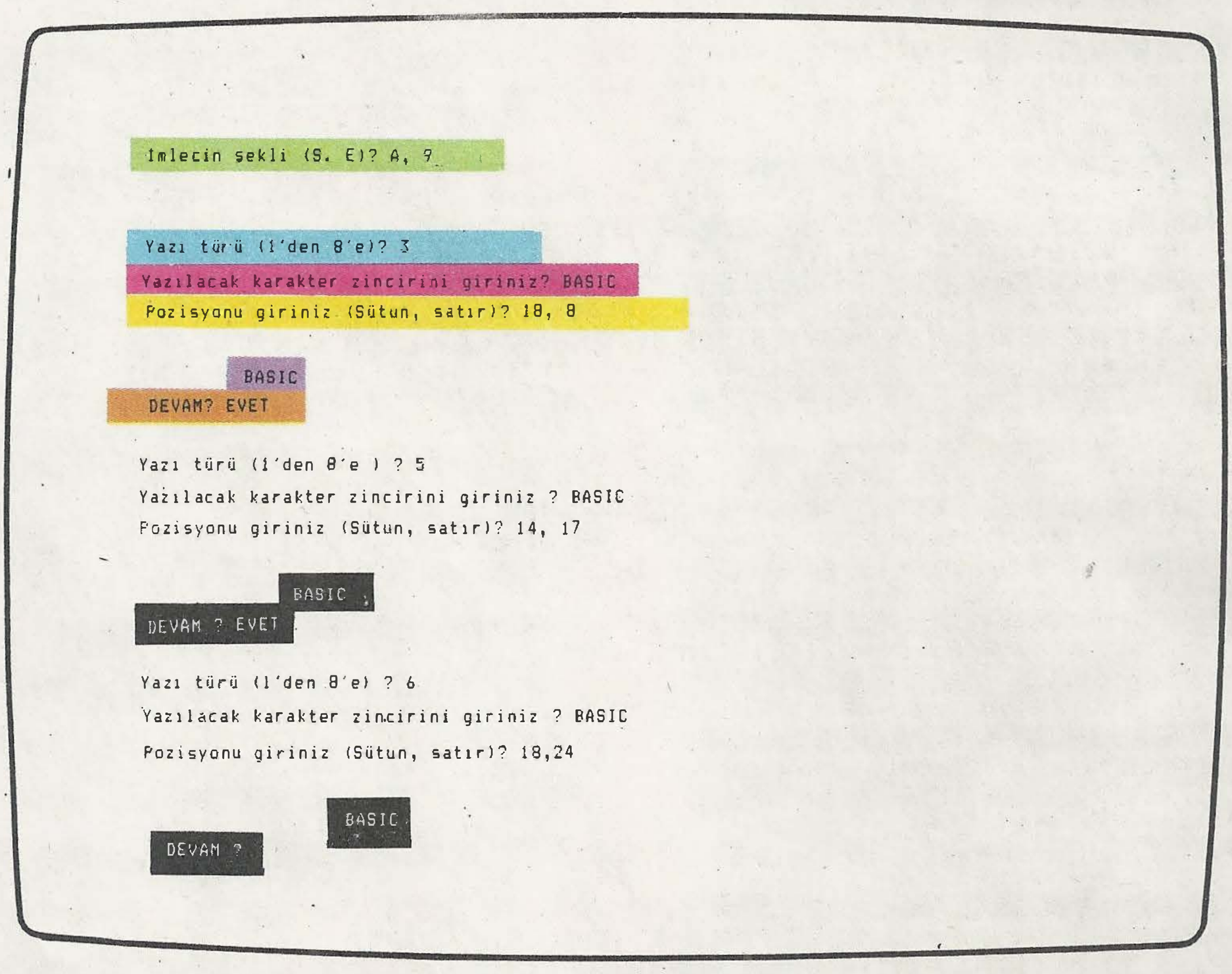
```
10  '  ***CUBUK DIYAGRAMLARININ BILGIYAZIMI ICIN PROGRAM
20 'KUTUK=
25 ' OPTION BASE 1
30 DIM D$(7),A$(7)
35 DEFINT I-N
40 FOR I=1 TO 7:READ D$(I):NEXT I
50 DATA "PLANLAMA"
60 DATA "DENEME"
70 DATA "HESAPLAMALAR"
80 DATA "HAFRIYAT"
90 DATA "TEMELLER"
100 DATA "KABA INSAAT"
110 DATA "DAM"
115 INPUT "YORUM GIRINIZ";YOR$
120 INPUT "EVET/HAYIR";CEV$
140 IF CEV$<>"EVET" GOTO 190
150 '****KARAKTER ZINCIRLERININ SINYALLENMESI
160 FOR I=1 TO 7
170 A$=""    'SIFIR KARAKTER ZINCIRI (BOS ZINCIR)
180 NEXT I
190 INPUT "FAALIYETIN ADINI GIRINIZ";C$
200 '***ONGORULENLER (50.-100. SATIR ARASINDAN GIRILEN FAALIYETIN ARANMASI
201 '
202 '
203 '
204 C$="" +C$   'VERILER BIR BOS HANE ILE BASLAR (DATA SATIRLARİYLA KARSILASTIR
205 N=14-LEN(C$)
207 IF N=0 GOTO 210    'FAALIYET 14 KARAKTERE SAHIP
208 A$=C$+SPACE$(N)
210 K=0        'K=CENTIK;DEGER SIFIRSA,ONGORULMEMIS BIR FAALIYET SOZKONUSUDUR
220 FOR I=1 TO 7    'YEDI FAALIYETTEN BIRINI TERCIH ICIN DONGU
230 IF C$=D$(I) THEN K=I   'GIRILEN FAALIYET(C$) D$ DE ICERILMEKTEDIR
240 NEXT I
250 '
260 IF K=0 GOTO 400    'HAYIR**HATA*YENI GIRIS
270 INPUT "BASLAMA GUNU (ISIN BASINDAN ITIBAREN)";BASLAMA
280 INPUT "FAALIYET SURESI (GUN SAYIOLARAK)";SURE
290 S$="*"    '*=DIYAGRAMLARDA KULLANILACAK SIMGE
295 ' (YERINE HERHANGI BIR BASKA SIMGEDE KULLANILABILIR)
300 N=SURE
310 T$=STRING$(N,S$)    'SECILEN SIMGEDEN N SAYIDA
320 ' ALINARAK T$ KARAKTER ZINCIRI URETILIR
330 '
340 ' ** T$ BILGIYAZIMI YAPILACAK A$(I)KARAKTER ZINCIRINE EKLENIR
350 I=K     'SECILEN FAALIYET (210.-240. SATIR)
360 M=BASLAMA-1    'ZAMAN OLARAK BASLAMA POZISYONU
370 '
380 A$(I)=SPACE$(M)+T$
390 GOTO 190
400 '
410 IF C$="SON" GOTO 460
420 PRINT "**HATA ONGORULMEMIS FAALIYET TURU GIRILDI**"
430 PRINT "DEVAM ETMEK ICIN BIR KARAKTER GIRINIZ"
440 S$=INPUT$(1)
450 GOTO 190
460 '
470 '
480 GOTO 260
510 '
520 INPUT "YAZILACAK KARAKTER ZINCIRI GIRILDI";A$
530 INPUT "POZISYON(SUTUN, SATIR) GIRILDI";X,Y
540 K=LEN(A$)    'YAZILACAK KARAKTER ZINCIRININ UZUNLUGU
550 M=80-K    'KARAKTER ZINCIRININ EKRANA SIGMASININ SINANMASI
560 IF X<0 OR X>M THEN PRINT "**HATA**":GOTO 520
570 IF Y<0 OR Y>24 THEN PRINT "**HATA**":GOTO 520
580 '
590 ' *****YAZMA*****
610 PRINT FNP$(X,Y);FNC$(N);A$
640 INPUT "DEVAM";CEV$
650 IF CEV$="EVET" GOTO 390
660 END
```


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

EKRANIN İŞLETİMİNE ÖRNEK

Bu sayfada, bir önceki sayfada bulunan programın işletilmesiyle elde edilen bazı çıktılar gösterilmektedir.





İmlecin şeklini belirleyen verilerin sorulması, 330 satırının yerine getirilmesiyle gerçekleşir. Kullanıcı onaltı tabanlı sayılar olan A ve 9'u girer. Bu değerler imlecin dikdörtgen şeklinde olmasını sağlarlar.

400 satırında program, yazı türünü belirleyen kodun girilmesini ister. Verilen değer (3), "altı çizili" anlamına gelmektedir (satır 48).

Programın 520. satırı ekrana yazılacak olan karakter zincirinin girilmesini ister. BASIC karakter zinciri, A$'nin değişkenlerine aktarılır.

Bu aşamada (satır 530) program kullanıcıya, A$ karakter zincirinin başlangıcını oluşturacak noktanın ekran üzerindeki koordinatlarını sorar. Kullanıcı, 10. sütun ile 8 satırın kesiştiği noktayı seçer.

610 satırında bulunan PRINT komutu yerine getirilir. Bunu sonucunda daha önce verilmiş olan karakter zinciri (BASIC), seçilen türde yazılır. (3 = altı çizili).

Programın sorusu (satır 640) üzerine, kullanıcı devam etmek istediğini belirtir. Denetim 390 satırına geri döner.

Bundan sonraki iki aşama, aynı karakter zinciriyle gerçekleştirilen iki yazı örneğini göstermektedir. Örneklerin ilki ters gösterim (açık renk ekrana koyu yazı), ikincisi ise tersi ve altı çizili gösterimdir.

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

Bu alt-programın kullanımı, onu çağıran ana program için, biri aylar için bir kısaltma, öbürü de her ay için gün sayısını içeren iki tane tek boyutlu dizey oluşturur.

Sınama mantığı aşağıdaki gibidir:

Alt-program çağırıldığında şu ifadeleri içeren üç karakter zinciri gözden geçirilmelidir; GUN$- iki karakterli, AY$- iki karakterli, YIL$- iki karakterli.

Alt-programda AY$ değişkeninde içerilen kısaltma, öngörülenlerle karşılaştırılır; bulunamazsa, hata ''ay''dadır. Kısaltma bulunabilirse, konumu ayın sayısını gösterir (Ocak-1, Şubat-2 vb.). Günün geçerliğinin sınanması, verilen değerin (ASCII tarafından sayıya dönüştürülmüştür) ilişkili aydaki gün sayısından küçük veya o sayıya eşit olup olmadığı açısından yapılır (ay, daha önce geçerli olarak kabul edilmiş olmalıdır). Bu süreçte karşılaşılan tek sakınca 29 Şubat ile ilgilidir. Burada bir artık yılın sözkonusu olup olmadığı kontrol edilmelidir.

TEST 17'nin Çözümleri

1 / Bilgiyazar noktalardan oluşmuş bir matris kullanarak her şekli basar. Bilgiyazım, yazı kafasını taşıyan kızak iki yönde hareket ederken, bir ''darbe'' ile oluşur.

Bir satıra, saniyede 120 karakter (yaklaşık saniyede 1 satır) süratiyle 132 karakter basılabilir. Bilgiyazarın, bilgisayardan gelen bilgilerin geçici olarak saklandığı 4096 baytlık bir arabelleği vardır.

---

2 / CHR$(33) ve CHR$(110) kodları bir defa ! karakterini, bir defa da n karakterini meydana getirirler. Buna karşın CHR$(27) ve CHR$(10) özel kodlardır. CHR$(27), bilgiyazarı güdümleri kabul etmesi için hazır hale getiren ESCAPE kodudur. CHR$(10) ise LINE FEED olarak adlandırılan ve satır kaydırmaya yarayan bir koddur.

---

3 / 20. satır, A$ karakter zincirini C değişkenine yazar.30.satır bir hata içerir. çünkü bir karakter zincirinin bilgiyazımı istendiği halde, kullanılan yazı düzeni yalnızca sayılara yöneliktir. Komutun doğru şekli şu olmalıdır:

30 PRINT A$; : PRINT USING '' # # # # . # '' ; B

---

4 / 10. satır: Beş değer; her biri bir satıra gelmek üzere yazılır.

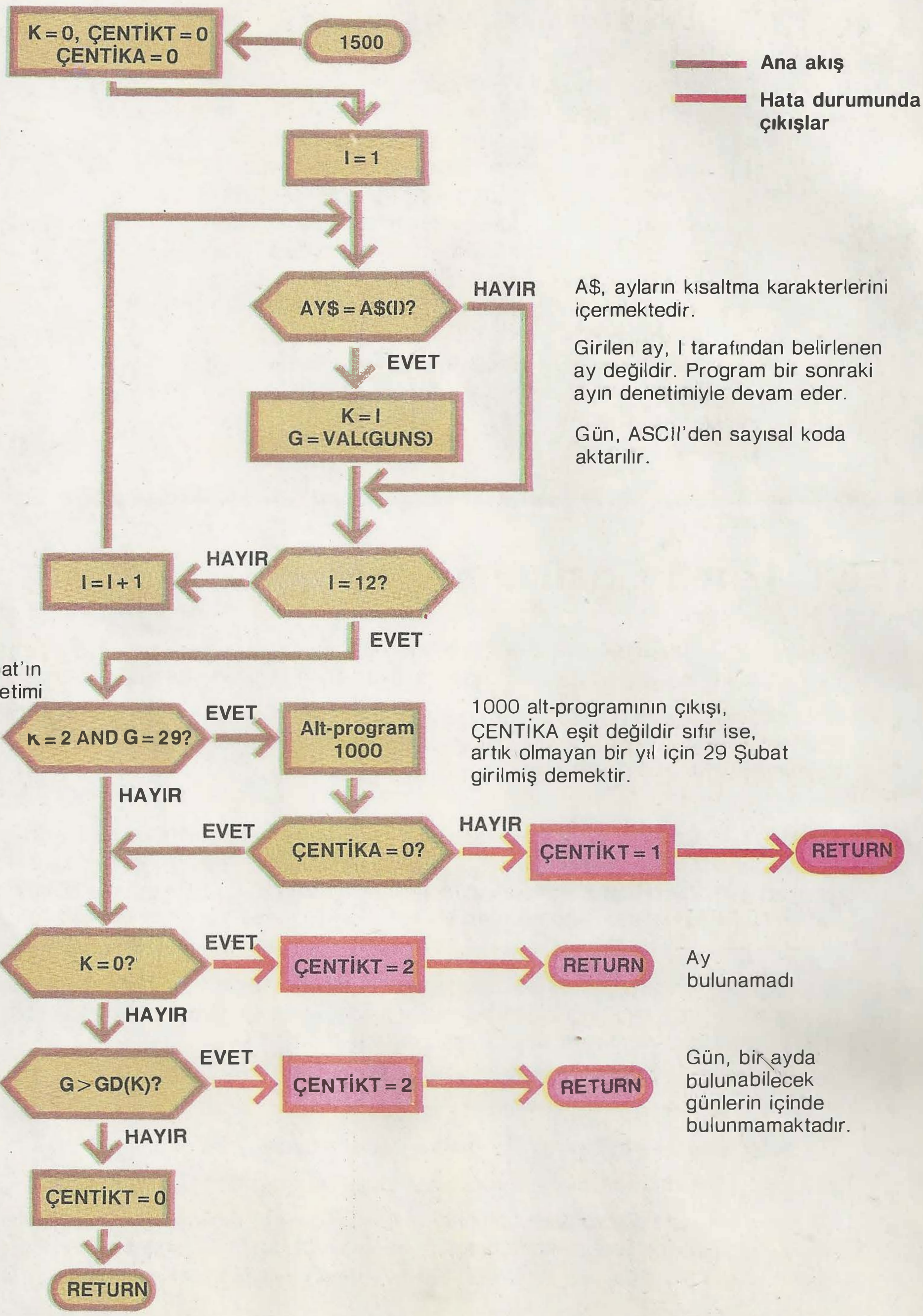
20. satır: Beş değerin hepsi aynı satıra yazılır (simge ;).

30. satır: Bir hata sözkonusudur, çünkü TAB(10), eğer ardından ''; '' simgesi geliyorsa, beş değerin yazımının aynı satıra ve aynı sütuna (10) yapılmasını sağlar. Bu nedenle bu durumda beş tane üstüste basılmış çıkış elde edilecektir.

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

AY/GÜN DENETİMİ İÇİN ALT-PROGRAM

Girişler: GUN$ = Günü içeren karakter zinciri (1'den 31'e kadar sayısal karakter)
AY$ = Ayı içeren karakter zinciri (iki rakamlı kısaltma)
YIL$ = Yılı içeren karakter zinciri (Artık yıl denetimi)
Çıkışlar: Hatalarda ÇENTİKT > 0, doğru tarihlerde ÇENTİKT = 0



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

Sınama işlemi özel bir alt-program ile gerçekleştirilir (alttaki şema).

Yüzyılların başlangıç yıllarının (1900, 2000 vb.) 400'e bölünebilir olmaması durumunda, artık yıl olmadıklarına dikkat edilmelidir. Bu alt-programda her bir hata türü için farklı çentikler öngörülmüştür. Sayfa 602'de ekranda bir tarihi okuyan ve daha önce adı geçen alt-programları kullanarak denetleyen bir alt-programın akış şeması verilmiştir. Bu alt-program, ekran maskesi ile yapılan bir tarih girişi örneği veriyor. Programa verileri girmek için en kolay ve en çabuk yöntem INPUT komutunun kullanılmasıdır. Örneğin;

INPUT GÜN$, AY$, YIL$

yazıldığı takdirde, tarihi oluşturacak olan üç değer karakter zinciri olarak kaydedilmiş olur.

Bu giriş şeklinin iki önemli kusuru vardır. Öncelikle kullanıcının veriler için seçilmiş yazı düzenine doğrudan bir etkisi olamaz, örneğin ayın kaç karakterle verilmesi gerektiğini bilemez. Ayrıca giriş satırı sistem tarafından belirlenmiştir ve daha sonraki kayıtlarda silinebilir. Şekilde gösterilen yöntem kullanıldığı takdirde (bkz. sayfa 604-605'teki program dökümü) ekranda

. . / . . / . .

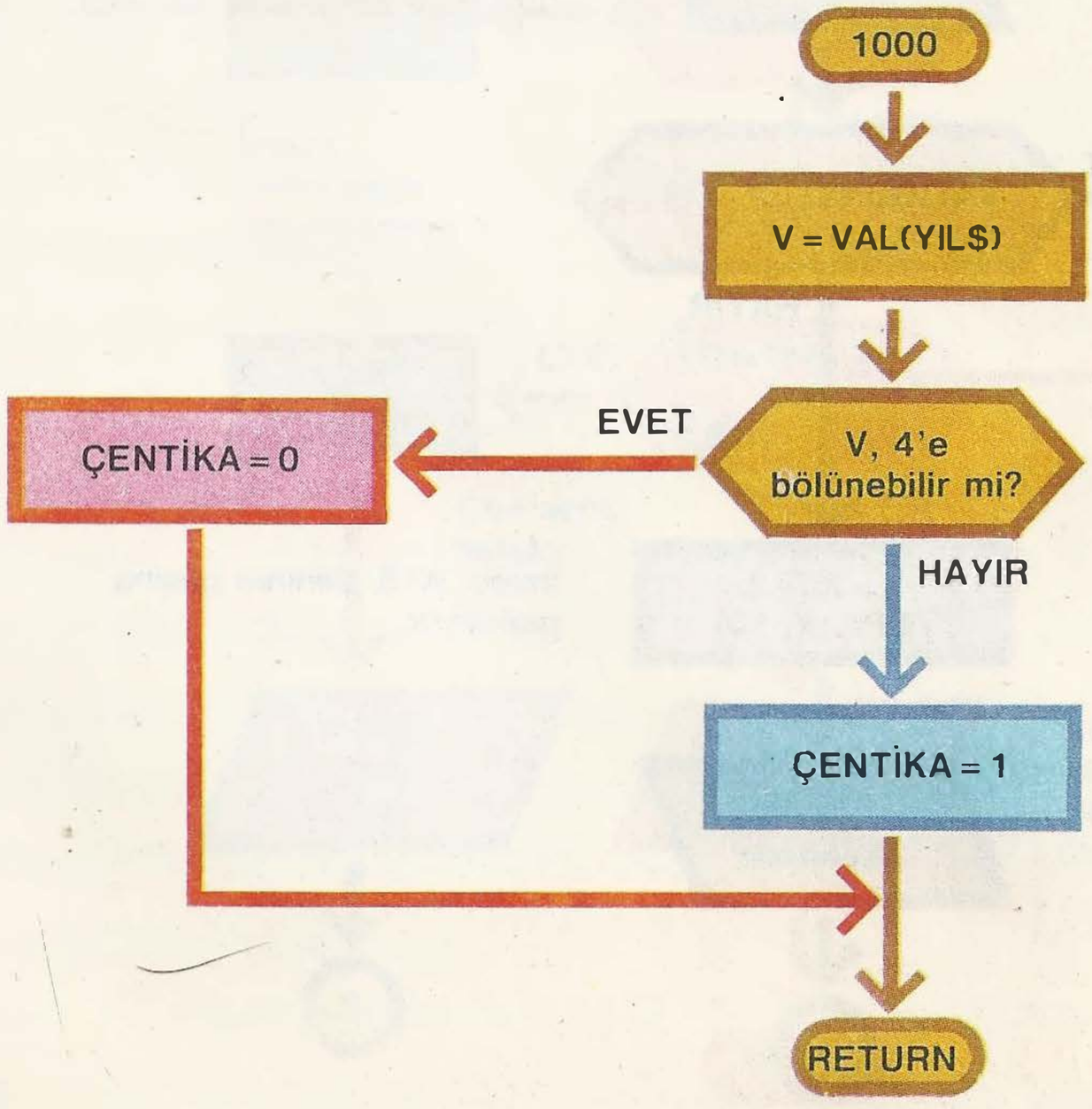
görüntüsü belirir. Burada imleç ilk alanın başında bulunmaktadır. Veri girişi tümüyle program tarafından yönlendirilmektedir. Kullanıcı alanların uzunluğunu görebilir ve farklı veriler arasındaki ayırıcılar (/ simgesi) her zaman hazırdır. Yazı program tarafından verilen uygun bir komutla tekrar silinene kadar, ekranın istenen bir noktasında kalabilir. Gereken diğer veri girişleri, bir önceki veri kaydını silmeden ekranın başka bölgelerinde belirirler.

Alt-program, olası hatalı değerlerini ekran üzerine ikinci bir defa ve ters olarak (açık renk üzerine siyah) yazmayı ve böylece kullanıcının dikkatini çekmeyi sağlayan bazı komutlarla donatılmıştır.

Hatalı veri girişlerinin ters görüntülenmesi yöntemi çok fazla verinin girilmesi gereken programlarda çok gereklidir. Böylece hata hemen görülebilir.

ARTIK YILLAR İÇİN ALT-PROGRAM

Girdi : YIL$ = Yılı gösteren karakter zinciri
Çıktı : Artık yılsa ÇENTİKA = 0, başka bir durum için ÇENTİKA = 1
Kullanılan algoritmalar : Yıl'ı gösteren rakam 4'e bölünebiliyorsa, sözkonusu yıl artık yıldır



4'e bölünebiliyorsa, bir artık yıl sözkonusudur. 1900, 2000 gibi yüzyıl başlarının 400'e bölünemez olmaları halinde, artık yıl olmadıklarına dikkat edilmelidir.

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

TARİH GİRİŞİ VE DENETİMİ İÇİN ALT-PROGRAM

Alt-programın, DATA komutlarıyla donatılmış iki dizeye ihtiyacı vardır: Birinci dizey (MS.(12)) sırayla ayların numaralarını içermelidir (1, 2, 3 vb). İkinci dizey (AG (12)) her ayda bulunabilecek gün sayılarını içerir (31, 28, 31 vb).

Girdiler: XO, YO = Ekran üzerinde tarihin başlaması gereken noktanın koordinatları.

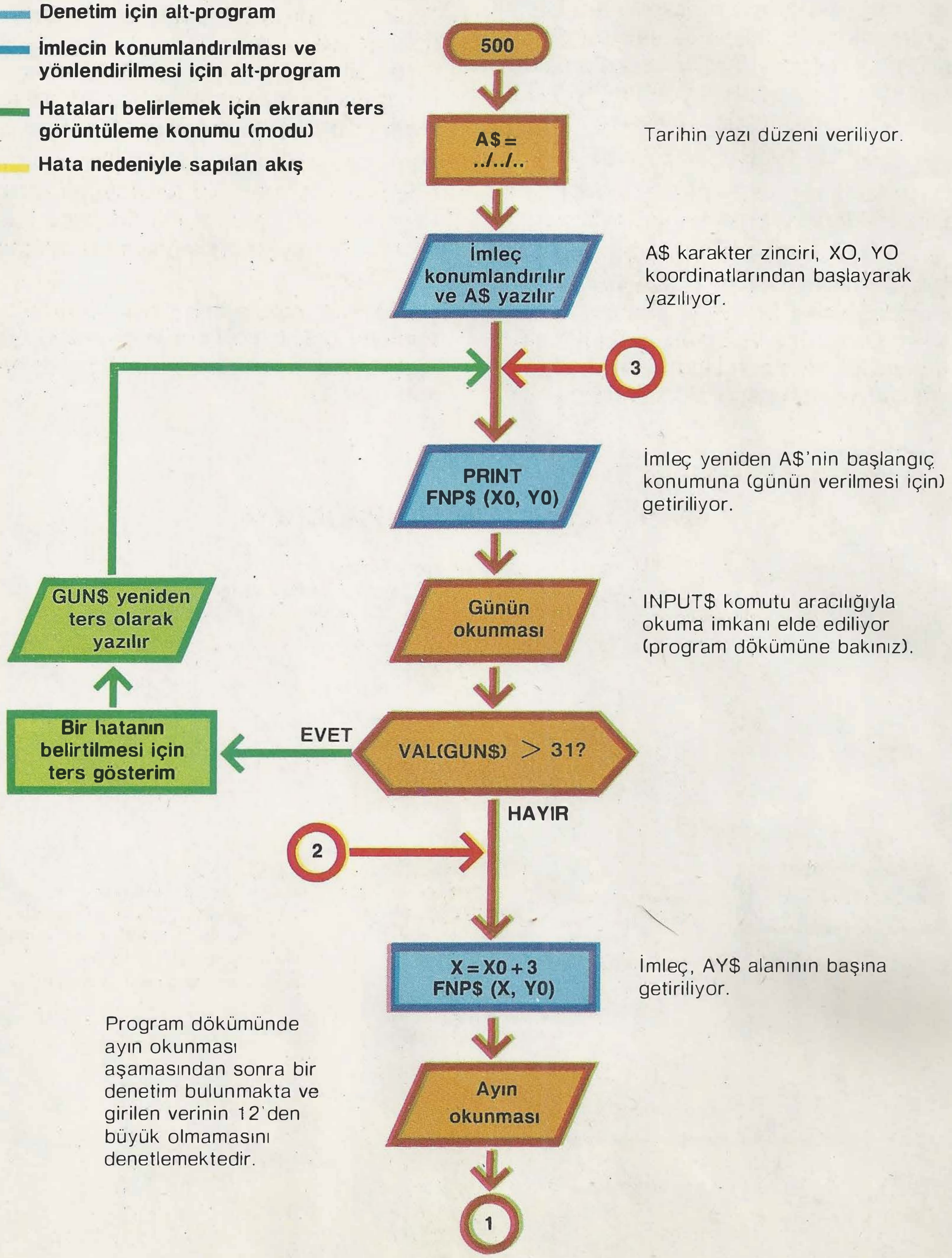
Çıktılar: TAR$ = gün, ay (sayısal) ve yılı (son iki rakam) içeren karakter zinciri.

Denetim için alt-program

İmlecin konumlandırılması ve yönlendirilmesi için alt-program

Hataları belirlemek için ekranın ters görüntüleme konumu (modu)

Hata nedeniyle sapılan akış



Tarihin yazı düzeni veriliyor.

A$ karakter zinciri, XO, YO koordinatlarından başlayarak yazılıyor.

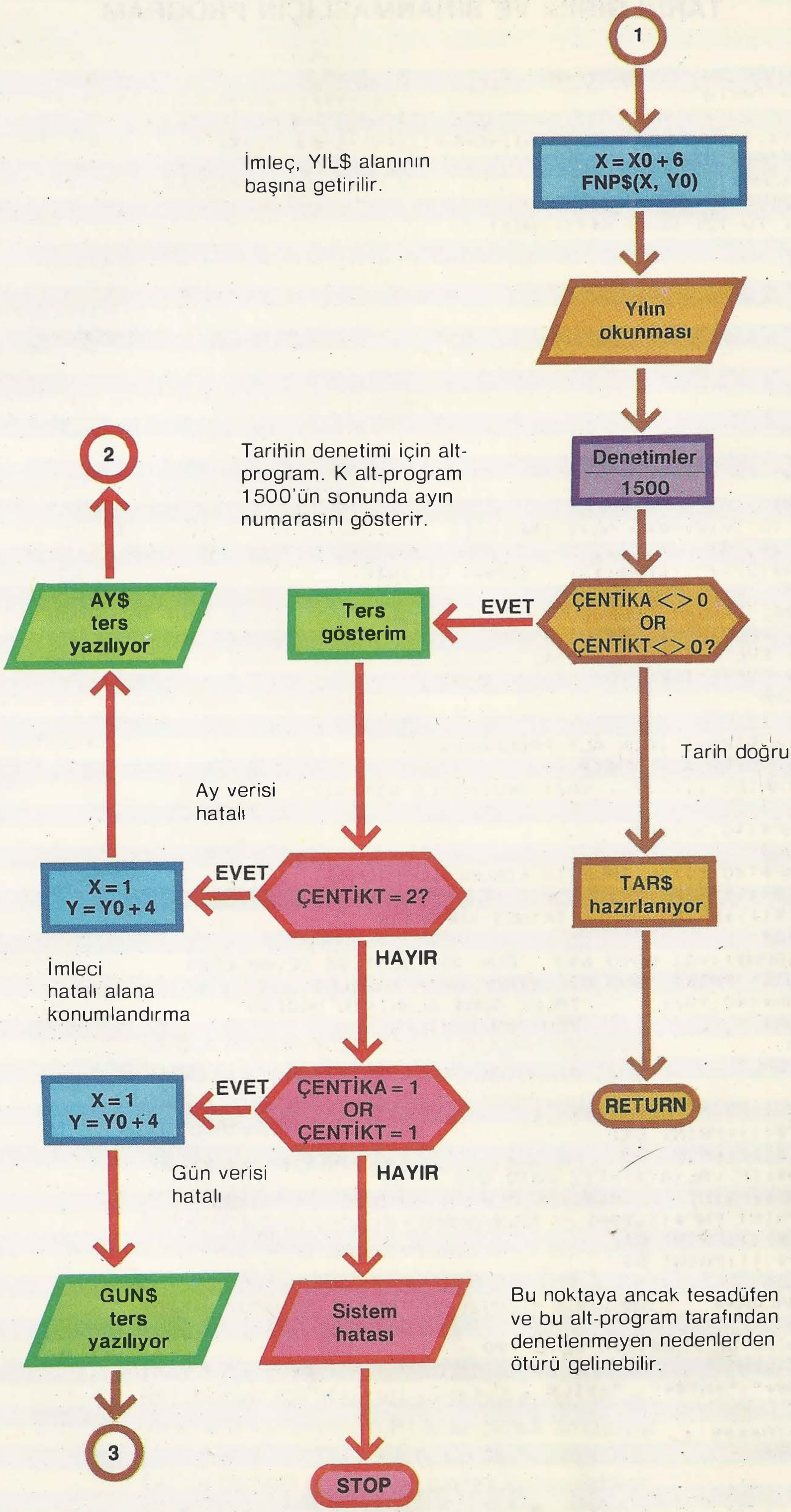
İmleç yeniden A$'nin başlangıç konumuna (günün verilmesi için) getiriliyor.

INPUT$ komutu aracılığıyla okuma imkanı elde ediliyor (program dökümüne bakınız).

İmleç, AY$ alanının başına getiriliyor.

Program dökümünde ayın okunması aşamasından sonra bir denetim bulunmakta ve girilen verinin 12'den büyük olmamasını denetlemektedir.

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



İmleç, YIL$ alanının başına getirilir.

Tarihin denetimi için alt-program. K alt-program 1500'ün sonunda ayın numarasını gösterir.

Ay verisi hatalı

İmleci hatalı alana konumlandırma

Gün verisi hatalı

Bu noktaya ancak tesadüfen ve bu alt-program tarafından denetlenmeyen nedenlerden ötürü gelinebilir.

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

TARİH GİRİŞİ VE SINANMASI İÇİN PROGRAM

```
10  ' *** DENEME ANA PROGRAMI***
15  ' KUTUK = TARIH
20 DEFINT A - Z
30 DEF FNP$(X,Y)=CHR$(27)+CHR$(61)+CHR$(31+Y)+CHR$(31+X)
40 DEF FNN$=CHR$(27)+"O"
50 DEF FNR$=CHR$(27)+"1"
60 DIM M$(12), AG(12)
62 FOR I = 1 TO 12: READ M$(I):NEXT I
64 DATA "01"
66 DATA "02"
68 DATA "03"
70 DATA "04"
72 DATA "05"
74 DATA "06"
76 DATA "07"
78 DATA "08"
80 DATA "09"
82 DATA "10"
84 DATA "11"
86 DATA "12"
87 XO=1: YO=10
88 FOR I =1 TO I=12:READ AG(I):NEXT I
90 DATA 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31
91 PRINT CHR$(27)+"+"CHR$(12)  'EKRAN SILINIR
92 GOSUB 500
94 PRINT TAR$,"TARIHI DOGRUDUR"
100 INPUT "DEVAM?(EVET/HAYIR)";CEV$
105 IF CEV$="EVET" THEN YO=YO+1
110 IF CEV$="EVET" THEN YO=YO+5:GOTO 92
115 PRINT FNR$
120 END
500 ' **TARIH SIRASI ICIN ALT-PROGRAM**
501 X=XO:Y=YO-2:PRINT FNP$(X,Y);
505 PRINT "TARIHI ../../.. YAZI DUZENIYLE GIRINIZ"
510 A$="../../.."
520 PRINT FNP$(XO,YO);A$
530 PRINT FNN$
540 PRINT FNP$(XO,YO); 'IMLECIN KONUMLANDIRILMASI
550 C$=INPUT$(1):PRINT C$;   'BIRINCI KARAKTER
560 D$=INPUT$(1):PRINT D$    'IKINCI KARAKTER
570 GUN$=C$+D$
580 IF VAL (GUN$)<=31 GOTO 640  'GUN >31 DEGILSE DEVAM EDER
590 PRINT FNR$: PRINT "GUN GIRISINDE HATA YAPILDI"
610 PRINT FNP$(XO,YO);      'IMLEC GUN$ ALANININ BASINA
620 PRINT GUN$              'YENIDEN YAZILIR
630 GOTO 530
640 'GUN GECERLI
650 X=XO+3
655 PRINT FNN$:PRINT FNP$(X,YO);
660 C$=INPUT$(1):PRINT C$;
670 D$=INPUT$(1):PRINT D$
680 AY$=C$+D$: IF VAL(AY$)=<12 GOTO 690
685 PRINT FNR$:PRINT "AY GIRISINDE HATA YAPILDI":GOTO 655
690 X=XO+6:PRINT FNP$(X,YO);  'YIL$ ALANI
700 C$=INPUT$(1):PRINT C$;
710 D$=INPUT$(1):PRINT D$
720 YIL$=C$+D$
730 '**SINAMALAR**
740 GOSUB 1500
750 IF FLAGA<>0 OR FLAGT<>0 GOTO 790
760 'DOGRU TARIH
770 TAR$=GUN$+" "+AY$+"  "+YIL$
780 RETURN
790 '*****HATA****
800 PRINT FNR$
```


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

```
810 IF FLAGT=2 THEN X=1:Y=YO+2:PRINT FNP$(X,Y);:PRINT AY$:PRINT FNN$:GOTO 650
820 IF FLAGT=1 OR FLAGA=1 GOTO 880
830 '
840 '****CENTIK DEGERLERINDE HATA****
850 '
860 PRINT "********SISTEM HATASI********"
870 STOP
880 ' **DUZELTMELER ICIN HAZIRLANMA****
890 '
900 Y=YO+2:X=1:PRINT FNR$:PRINT FNP$(X,Y);
910 PRINT FNN$:PRINT GUN$
920 GOTO 530
930 '
940 '***
1000 ' **ARTIK YIL OLUP OLMADIGININ SINANMASI
1010 ' GIRIS=YIL$
1020 ' CIKIS=EGER ARTIK YILSA FLAGA=0
1030 '
1040 V=VAL(YIL$)
1050 FLAGA=V MOD 4:IF FLAGA>0 THEN FLAGA=1
1060 RETURN
1070 '
1500 '
1510 ' **GUN/AY UYUSMASININ SINANMASI
1515 K=0:FLAGT=0:FLAGA=0
1520 FOR I=1 TO 12
1530 IF AY$=M$(I) THEN  K=I
1540 G=VAL(GUN$)
1545 NEXT I
1550 IF K=2 AND G=29 THEN GOSUB 1000  'ARTIK YIL OLUP OLMADIGININ SINANMASI
1560 IF K=2 AND G=29 AND FLAGA<>0 GOTO 1610  '**HATA**
1585 IF K=2 AND G=29 GOTO 1600  'ARTIK YIL GUN=29 GECERLI
1590 IF K=0 THEN FLAGT=2: RETURN
1592 IF G>AG(K)   THEN FLAGT=2:RETURN
1600 FLAGT=0:RETURN
1610 '***HATA***
1620 FLAGT=1
1630 RETURN
```

Fonksiyon Tuşları

Birçok bilgisayarda, kodları programlanabilen tuşlar vardır. Bunlara **fonksiyon tuşları** denir; çünkü programlanmış fonksiyonları yürütmekte kullanılırlar.

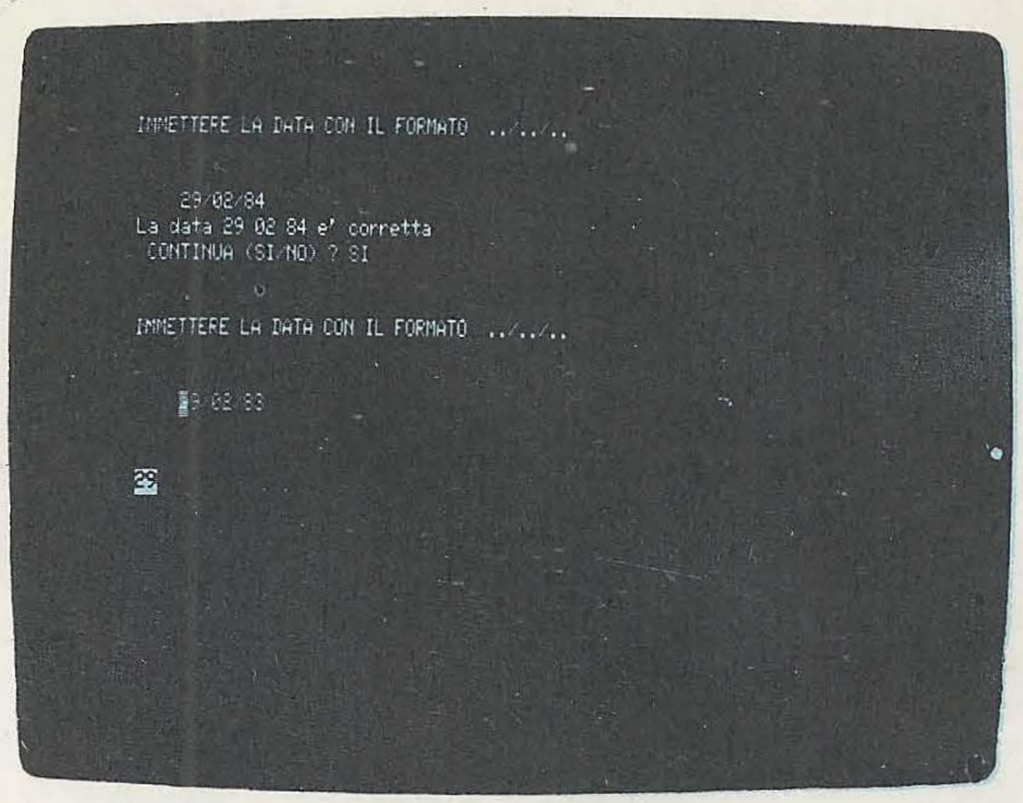
Klavyenin istenen her tuşu, bu tuşun karşılığı olan ASCII kodunu üretir: Üzerinde B harfi bulunan tuşun karşılığı 66 değeridir vb. Fonksiyon tuşlarına önceden herhangi bir değer verilmemiştir, programcı bunların her birine uygun komutlar yardımıyla istediği kodu verebilir.

Fonksiyon tuşları komut anlamını taşıyan kodları (bunlar ASCII tablosunda bulunmazlar veya orada kullanılmazlar) elde etmek amacıyla kullanılır. Örneğin bir fonksiyon tuşuna, ASCII'de ne harf ne de sayı olarak kullanılabilen 18 verilirse, program özel bir kodun sözkonusu olduğunu anlar ve bu yeni komutu yerine

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

BİR TARİHİN GİRİLMESİ VE DENETİMİ

Bu sayfada, sayfa 604'te gösterilen programın bazı çıktılarının ekran görüntüsü sergilenmektedir.




```
TARIHI../../.. YAZI DUZENIYLE GIRINIZ

29/02/84
29/02/84 tarihi doğrudur.
DEVAM(EVET/HAYIR)?EVET

TARIHI../../.. YAZI DUZENIYLE GIRINIZ.

29/02/83
```

Başlığın bilgiyazımı 505. satır aracılığıyla gerçekleşir ve yalnızca kullanıcıya yardım amacını taşır.

520. satırın yerine getirilmesi, ../../.. ekran maskesinin bilgiyazımını sağlar. Yukarıdaki fotoğraf, kullanıcının tarihi maskeye yazmasından sonra çekilmiştir.

Veri girişinden sonra, girilen verilerin geçerliliğinin denetlenmesi aşamasına geçilir; bu işlem bilgisayar tarafından yapılır. 29/02/84 tarihi doğru olarak kabul edilmiştir (1984 bir artık yıldır) ve bu bilgi program tarafından kullanıcıya iletilmiştir.

Programın sorusu üzerine (100. satır), kullanıcı, veri girişine devam etme isteğini bildirir.

Bu defa, yanlış bir tarih verilmiştir: 29 Şubat 1983 şeklinde bir tarih olamaz, çünkü 1983 bir artık yıl değildir.

Doğruluk denetimi hatayı tespit eder, günü ekrana ters olarak yazar ve imleci gün alanının başına yerleştirir: (500, 610, 620. satırlar).

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

getirmek için o anda yürüttüğü fonksiyonu durdurur. 18 koduna diskette saklama fonksiyonu, 19 koduna bilgiyazım vb. yerleştirilebilir. Sayfanın altında bu mantığın kullanımını gösteren bir akış şeması verilmiştir. Bu örnekte 18, 19 ve 20 değerlerine sahip üç adet tuş bulunmaktadır.

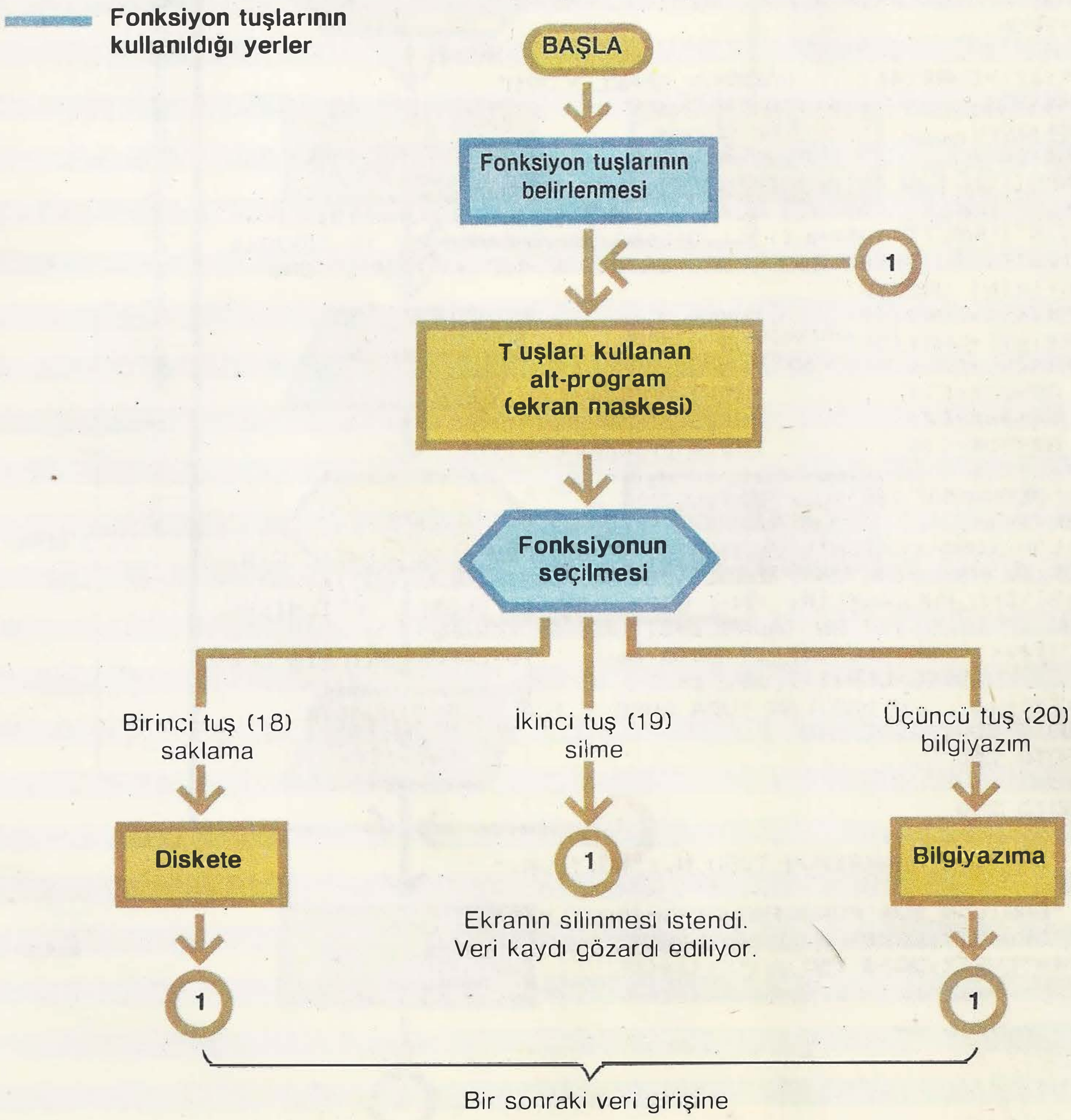
Her bir tuş üzerine bir sayının (kodun) yerleştirilmesi, tuşların program tarafından kullanılmasından önce gerçekleştirilmelidir. Bu nedenle fonksiyon tuşları kullanan her programda, genel olarak kodun sıfırlanması (hazırlanması-initialization) için bir alt-program bulunmalıdır. Bir tuşa yerleştirilmiş bir sayı, bilgisayar kapatılana veya aynı tuşa başka bir değer verilene kadar geçerli kalır. Fonksiyon tuşlarının tanımlanması, farklı makinelerde tamamen farklı bakış açılarına göre yapılabilir. Bu bölümde ilgilendiğimiz makinede (Bufetti B801) tuşlar küçük harflerle belirtilmiştir.(*)

Bu makinede hazırlama komutu aşağıdaki karakter sırasından oluşur:

— Escape kodu = CHR$(27)
— Hazırlama fonksiyonunu gösteren kod = CHR$(46)
— Tuşun tanımlanması (örneğin c; ASCII'de C = 99) = CHR$(99)

(*) Bir fonksiyon tuşunu adlandıran bir harf, klavyenin elektriksel bağlantılarının düzenlenişine göre de değişebilir.

FONKSİYON TUŞLARININ KULLANIM MANTIĞI



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

— Tuşa göre yerleştirilecek kodun rakam bileşenlerine ayrılmış hali

Örneğin, fonksiyon tuşu c'ye A değeri verilmek isteniyorsa, A harfinin ASCII kodunun (onaltı tabanlı 41) karakterlerini oluşturan 4 ve 1'in ASCII koduna çevrilmiş hali verilmelidir.

4'ün ASCII kodu 52 ve 1'inki 49'dur, bu nedenle A harfi CHR$(52) ve CHR$(49) ile oluşturulur.

Eksiksiz bir hazırlama komutu aşağıdaki karakter zinciriyle verilir:

```
A$ = CHR$(27) + CHR$(46) + CHR$(99) +
     Escape     Formüllendirme  (c) tuşunun tanımlanması

     CHR$(52) + CHR$(49)
     Yerleştirilecek olan kodu (A) oluşturan karakterler
```

Tuşun programlanması, A$ karakter zincirinin PRINT A$ komutu ile yazılmasıyla sağlanır.

Bu sayfada 17, 18 ve 19 ondalık değerlerini fonksiyon tuşlarına yerleştiren bir hazırlama alt-program örneği verilmiştir.

Verilerin ekran maskeleriyle verildiği programlarda, fonksiyon tuşlarının yararlı bir uygulamasını görmek mümkündür. Sayfa 609 ve 610'da veri girişi için genelleştirilmiş bir alt-programın akış şemasını görüyoruz.

Bu alt-program daha ayrıntılı olarak incelenecektir. Burada bizi ilgilendiren gelişmeler aşa-

FONKSİYON TUŞLARININ KULLANIMINA ÖRNEK

```
10 '**FONKSIYON TUSLARININ KULLANIMINA ORNEK
15 'KUTUK=TFON
30 DEFINT A - Z
40 A$=CHR$(27)+CHR$(46)     'GUDUMUN GENEL KISMI
50 B3$=CHR$(99)             'TUS N.2=c
60 B4$=CHR$(100)            'TUS N.3=d
70 B5$=CHR$(101)            'TUS N.4=e
71 ' **FONKSIYON TUSLARINA DEGERLERIN VERILMESI**
72 ' DEGERLER ONALTI TABANLI OLARAK IFADE EDILMIS OLMALIDIR
73 ' ORNEGIN (ONALTI TABANLI) 11 DEGERI ON TABANLI 17 YE TEKABUL
74 'ETMEKTEDIR. BU DEGER BIR TUSA VERILIRSE ASC(*) FONKSIYONU
75 ' 17 SAYISINI URETIR
80 C1$=CHR$(49)+CHR$(49)    ' TUS N.2
90 C2$=CHR$(49)+CHR$(50)    ' TUS N.3
92 C3$=CHR$(49)+CHR$(51)    ' TUS N.4
110 PRINT A$+B3$+C1$        ' TUS 2 PROGRAMLANIYOR
120 PRINT A$+B4$+C2$        ' TUS 3 PROGRAMLANIYOR
130 PRINT A$+B5$+C3$        ' TUS 4 PROGRAMLANIYOR
140 '
150 PRINT "HERHANGI BIR TUSA BASINIZ"
160 I$=INPUT$(1)
162 ' BASILAN TUSA VERILMIS OLAN DEGERI YENIDEN ELDE ETMEK ICIN
163 ' SONUC OLARAK YINE ON TABANLI DEGERE SAHIP OLAN ASC(*)
164 ' FONKSIYONU KULLANILIR; YANI EGER TUSA 11 DEGERI VERILMISSE,
165 ' SONUC 17 OLUR (17 ON TABANLI=11 ONALTI TABANLI)
170 IF ASC(I$)<17 OR ASC(I$)>19 GOTO 260  ' TUSLARA VERILMIS OLAN
172 '              DEGERLER:17, 18, 19
180 K=ASC(I$)-16    'K BASILAN TUSA GORE 1 ILA 3 ARASINDADIR
190 ON K GOTO 200, 220, 240
200 NU=2:GOTO 250
210 GOTO 150
220 NU=3:GOTO 250
240 NU=4
250 PRINT "BASILAN FONKSIYON TUSU N.:"NU,"DUR:"
255 GOTO 150
260 PRINT "BASILAN TUS FONKSIYON TUSU DEGILDIR"
270 INPUT "DEVAM?";CEV$
280 IF CEV$="EVET" GOTO 150
300 END
```


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

VERİ GİRİŞİ İÇİN ALT-PROGRAM GENEL AKIŞ ŞEMASI

Bu alt-programda veri sıralaması eksiktir (3. nokta); sıralama ayrıca ele alınmalıdır.

Ekranın denetimi için fonksiyonlar

Karakterlerin denetimi için fonksiyonlar

Karakterin kabulü



Giriş, alanın başından itibaren istenmektedir.

Yazılacak olan ara bellek, alanın uzunluğundan daha kısa olabilir. Bu durumda alan tümüyle doldurulmalıdır.

Alanla ilgili açıklama XO sütunu ve YO satırından başlamalıdır.

Ara-belleğin ters gösterimle yazılması seçeneği

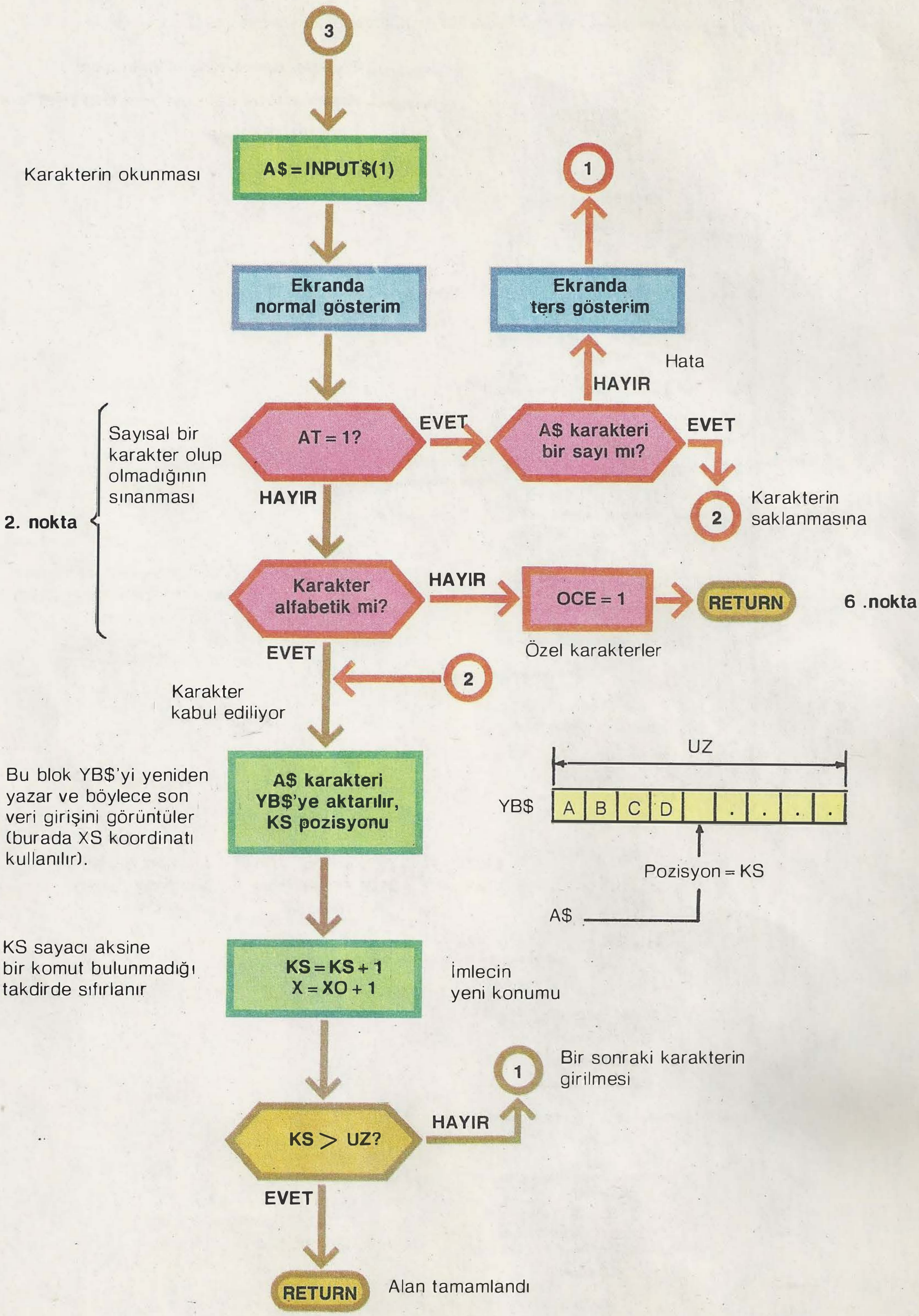
U = Açıklamaların uzunluğu; girişin başlangıç pozisyonu XO + U + KN'dir. Bu değer XS'de ilerde kullanılmak üzere saklanır.

4. nokta

1. nokta — KN <> 1, açıklamanın yeniden yazılmaması gerektiğini gösterir

5. nokta

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



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

ğıdaki fonksiyonları yerine getiren bir alt-program oluşturulmasıdır.

1 / Girişi yapılacak alan uzunluğu, ekranın istenen bir yerinde sahip olduğu karakter adedi kadar nokta ile uzunluğu belirtilerek yapılabilmeli.

2 / Her verilen karakterin uygunluğunun sınanması (sayısal, harfsayısal, özel). Bir hata durumunda imleç ilerlemez, hatalı karakter tekrar ters olarak yazılır ve program düzeltmeyi bekler.

3 / Alanın tamamiyle dolmaması durumunda veri kaydı sayısal ise sağa, harfsayısal ise sola yeniden dizilmelidir.

4 / Alt-program, girişi yapılan verilerin içinde saklanabileceği bir ara-belleğin yardımıyla çalışmalıdır ve veri yazılacağını bir çentiğin aldığı belirli bir değere göre öngörmelidir. Rutin çağırıldığı takdirde bu yolla hem veri kaydını tam olarak yazabilmek, hem de yalnızca tam bir veri kaydını gösteren noktalamayı görüntülemek mümkündür.

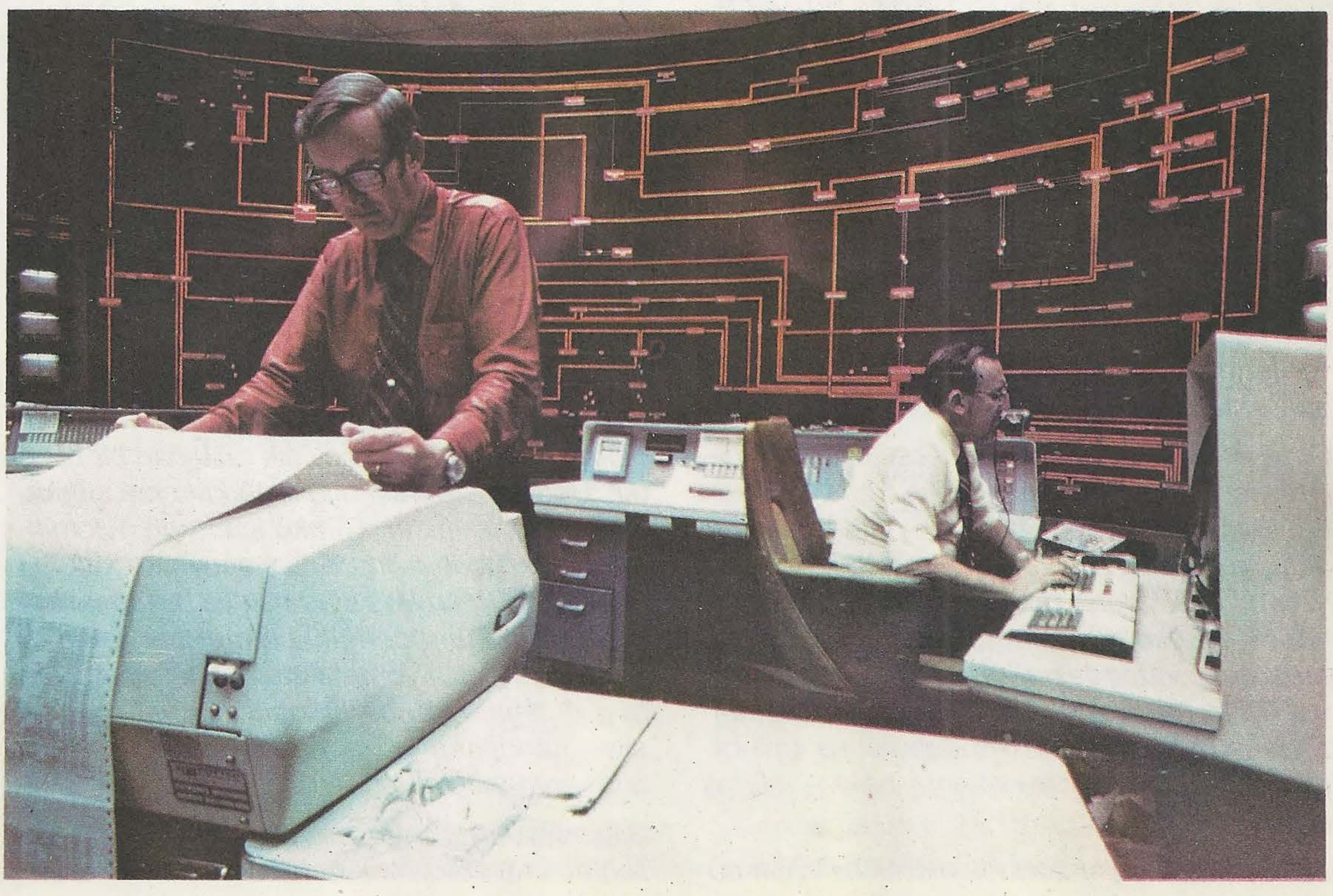
5 / İmlecin başlangıç pozisyonu (veri girişinin başladığı yer) program tarafından öyle belirlenmelidir ki, rutin aynı zamanda kısmi bir veri girişi için de kulanılabilsin.

6 / Yabancı bir karakter verildiği anda (ne sayı ne de harf olan) bu karakterin sayısal karşılığı, sayısal bir değişkene aktarılır ve rutin terkedilir. Bu fonksiyon, programlanabilen tuşların düzenlenmesinde doğrudan kullanılır.

4. ve 5. noktalarda belirtilen işlevlerin gerekliliği, ekran maskeleriyle ilgili ayrıntılardan bahsedilirken, daha belirgin olarak anlaşılacaktır. Programda kullanılan değişkenler şunlardır:

YB$ = Girişi yapılan verilerin saklanması için yardımcı ara-bellek
UZ = Alanın uzunluğu (karakterlerin sayısı)
AT = Alan cinsi (1 = sayısal, 2 = harfsayısal)
AZ$ = Alanın açıklamasını içeren karakter zinciri
XO,YO = Alanın ekran üzerindeki yerinin koordinatları
KN = Veri girişinin başlayacağı karakterin numarası
KS = Girilen karakterlerin sayıcısı
OCE = Çok özel bir çentik, özel bir karakterin girişi yapıldığında OCE = 1 olur.

Modern bir enerji santralında denetim odası.



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

Verilerin Korunması

Bilgisayar dünyasında kullanılagelen güncel deyimler arasında ''Veri Bankası'' deyimine çok sık rastlanır. Günümüzde Batı'da veri bankaları sosyal, özel ve kamu kuruluşlarının yönetim kademelerinin önemli bir bölümünü oluşturur ve bütün gelişmiş sanayi ülkelerinde her vatandaşın en azından bu veri bankalarından birinde, bir bilgisayar kütüğünde kaydı vardır. Bu tür bir kayıt sisteminin zorunlu olarak bazı olumsuz sonuçları beraberinde getirmesi şart değildir; modern devlet yönetimleri bu sistem olmaksızın yeterince etkin çalışamazlardı, sağlık hizmetleri gereğince sunulmaz, trafik ve motorizasyon kurallarına uyulmasına yönelik bir çözüm bulunamazdı. Nüfus sayımı yapılamaması durumunda vergi reformları, sivil kuruluşların oluşturulması ve kentlerin gelişmesi yalnızca kağıt üzerinde kalırdı. Fakat bazı tedirgin edici yan arazlar da kendini göstermektedir. ABD'de bu şekilde birçok mahrem bilgiyi kaydetmiş olan iki binden fazla kredi enstitüsü bulunmaktadır. Örnek vermek gerekirse, bu şirketlerden birinin arşivinde 100 milyon belge vardır.

Ama asıl çarpıcı olan toplanan bilgilerin miktarı değil, daha çok bu bilgilerin kaydedilmesinde yararlanılan yöntemlerdir. Bu veriler uzun zamandır geleneksel arşivlerin sayısız raflarında yığılmış olarak bulunuyordu. Bugün bilgisayarların bellek kapasitelerinin çok artmış olması, terminaller aracılığıyla dünyanın öbür ucundan bile veri aktarılabilmesi, yani erişim olanaklarının olağanüstü kolaylaşmış olması, veri toplama yöntemlerini çok değiştirmiştir. Bir örnek olarak, kısa zaman öncesine kadar Kanada Sosyal Yardım Örgütü'nün verilerinin Güney Kore'de bulunan bir bürodan aktarılıyor olması verilebilir.

Giderek yaygınlaştırılmaya çalışılan etkili uluslararası kontroller olmaksızın, bireyin, veri bankalarında kendisi hakkında nelerin kayıtlı olduğunu ve bu bilgilerin kimlere verildiğini öğrenebilmesi pratik olarak imkansızdır. Diğer birçok gruplar gibi vatandaş özgürlüğü taraftarları da, vatandaşın kendisiyle ilgili bilgiler üzerinde hakkı olduğunu savunmaktadırlar. Bu düşünce ''her bireyin kendisi ile ilgili bilgilerin yayılmasını denetleme hakkı'' olarak adlandırılmaktadır.

Bütün vatandaşlar zorunlu olarak birtakım büyük makamlara, bazı kişisel bilgiler vermeye mecbur tutulmuşlardır. Bu bilgiler ilgili makamların veri bankalarına aktarılır. Bu yöntem olmaksızın bir banka hesabı açmak veya bir taşıt aracı kullanmak mümkün olmazdı. Gerek veri bankalarında saklanan bilgilerin miktarına, gerek bunlara erişim imkanlarına yönelik birtakım sınırlamalar olmalıdır. Bilgisayar çağında, kişisel bilgilerin, vatandaşların çoğunun istediği hassasiyette korunmuyor olması çok ciddi bir sorundur. Çünkü bilgiler, yanlış, eksik, zaman aşımına uğramış veya gereksiz olabilir ve daha da kötüsü, belli bir amaç için verilmiş olan bilgi başka bir devlet dairesine iletilip tamamiyle değişik bir amaç için kullanılabilir. Ayrıca bilgilerin el altından veya yasadışı yollarla, ilgili kişinin haberi olmaksızın başkalarına aktarılması da mümkündür.

Halen polis örgütünün veri bankalarında saklanan bilgiler şaşılacak ölçüde büyük boyutlara ulaşmıştır. Bir devlet ne kadar demokratik olursa olsun, polis örgütü her halükarda, taşıt sahiplerini, çalınan araba ve dikkatsiz sürücü listelerini, kayıp ve aranan şahıslar hakkındaki verileri, suçluların fiziksel özelliklerini ve parmak izlerini içeren geniş bir bilgi yığınına sahiptir. Ayrıca terörist olduğundan veya siyasal yeraltı faaliyetlerinde bulunduğundan şüphe edilen kişilerle ilgili veriler de toplanabilir. Örneğin İngiltere'de taşıt sahibi olmak isteyen her vatandaş, kendine ilişkin birtakım verileri Swansea'deki ehliyet bürosuna kaydettirmek zorundadır. Bu bilgiler, bir merkezî polis bilgisayarına aktarılır. Bunun sonucunda polis, çoğu sabıkasız olan 26 milyon vatandaşın her biri hakkında en azından birkaç bilgiye sahip olur.

Demek ki bir vatandaş olarak bir devlet dairesine verilen bilgilerin, bunları tamamıyla farklı amaçlarla kullanacak olan bir başka devlet dairesine aktarılması çok sık raslanan bir olaydır. Üstelik polisteki bilgilerin kolayca çalınabildiği gözlenmiştir. Çünkü içlerinde çok miktarda eski polis memuru çalışan özel dedektif büroları, bilgisayara ulaşmak ve buradaki belirli bir kütüğün içeriğini elde etmek için her fırsatta yasadışı yöntemlere başvururlar. Hatta gizli telefon numaralarını bilen yasadışı kişiler, kendilerini polis memuru diye tanıtarak bilgi edinebilmektedirler.

Bilgisayarlarda saklanan veri bankalarının gizliliğini sağlayabilmek çok zordur. Eski arşiv-

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

leme sistemlerinde, istenen bilginin alınması için kişinin arşive bizzat girmesi gerekirdi. Ayrıca o dönemde arşive yönelik birtakım güvenlik tedbirleri almak mümkündü. Herhangi bir kişi belgelere izinsiz ulaşmayı başarabilse bile, bu kişinin kopya edebileceği, fotoğrafını çekebileceği ve hatta alıp götürebileceği bilgilerin miktarıyla ilgili belirli kısıtlamalar vardır.

Gerçi bilgisayar kütükleri verilerle ilgilenen kişilerin, gerekli yetkiye sahip olmadan istedikleri bilgiyi almalarını zorlaştırmıştır. Çünkü bilgisayar programlarında sadece yetkili kişilerin bilgi edinebilmesine izin veren ''erişim kodları'' (parolalar) bulunmaktadır. Bugün bilgisayar endüstrisinde çalışan elemanlar dışında, bu kodları çözebilecek bilgiye sahip çok az kişi vardır. Fakat kararlı ve uzman bir hırsız, bir veri bankasının içeriğini okumak isterse onu bu konuda engelleyebilecek hiçbir bilgisayar yoktur, üstelik böyle bir hırsız bu yolla bir anda çok büyük miktarda bilgiyi elde edebilir.

Şimdiye kadar çok az sayıda ''bilgisayar korsanlığı'' saptanmış olmasına rağmen, bu olayların her biri çok büyük yankılar uyandırmıştır. Massachusetts Institute of Technology'den birkaç öğrenci ulusal savunma açısından çok önemli birtakım bilgilerin saklandığı bir bilgisayara erişmeyi başarmışlardır. Michigan Üniversitesi öğrencileri kendi üniversitelerinin bilgisayar kodlarını çözmüş ve bir miktar bilgiyi silmişlerdir. Bilgisayar teknolojisinin koyu bir taraftarı ise FBI'ın kütüklerinden şüpheli ve tutuklu kişiler ile ilgili bazı bilgileri almayı başarmış, fakat sonunda soluğu hapishanede almıştır. Bu yasadışı büyücüler çok makbul kişiler olmaktadırlar. Sonunda ele geçirilen bu az sayıdaki profesyonel bilgisayar korsanı, bu işle ilgilenen bazı kuruluşlar tarafından danışman olarak işe alınmakta ve çok iyi para kazanmaktadırlar.

Bilgisayarlar, uzakta bulunan terminaller tarafından yönetildikleri veya bunları yönettikleri için olası hataların oranı artmaktadır. Örneğin, telefon hatlarındaki girişlerden kaynaklanan rastlansal bir olay, bilgilerin gizliliğini zedeleyebilir, hatta silinmesine neden olabilir.

Bu durumda insan kendine şu soruyu sorabilir: Vatandaşın refahı için gerekli bazı bilgilerin hükümet ve diğer devlet daireleri tarafından kaydedilmesi gereksinimi dikkate alınarak, vatandaşın özel hayatı nasıl korunabilir? Örneğin polis örgütü, vatandaşın sahip olduğu taşıtın özellikleriyle ilgili bilgilerden farklı olarak, vatandaş hakkında daha önceden edinilmiş bilgilere sahiptir. İngiltere'de anti-terörist mücadele ile ilgili olarak tutuklanan şüphelilerin otomatik olarak fotoğrafları çekilir ve parmak izleri alınır. Kendilerine dava açılmasa bile (ki çoğunlukla böyle olur) bu parmak izleri arşivde kalır. Yasadışı bir faaliyet hazırlığı içinde oldukları şüphesiyle gözaltında tutulan kişiler ve tanınmış sabıkalılarla çeşitli şekillerde ilişkisi olan kişiler kütüklere kaydedilirler. Bütün bunlar polisin yasadışı hareketleri daha gerçekleşmeden engellemek veya bir suç işlenir işlenmez hemen tutuklama yapabilmek için aldığı meşru önlemlerdir. Bugünkü teknoloji dev miktarda bilgileri bir araya getirme, karşılaştırma ve iletme imkanlarını sağlamaktadır. Ortalama vatandaş ise bu bilgilerin tamamıyla özel nitelikte olduğunu sanmaya devam edebilir.

Belli bazı uzman kuruluşlar kredi kayıtları tutmakta ve bunları potansiyel müşterilerinin mali durumları hakkında bilgi isteyen şirketlerin emrine sunmaktadır. Dünyanın en büyük özel veri bankası, 50 milyondan fazla Amerikan vatandaşı hakkında özel bilgiye sahip olan TRW-Credit Data adlı kuruluşa aittir. İsveç'te önde gelen iskonto bankaları, kısa bir süre önce Kredi Enformasyon Acentası adını taşıyan bir enstitü kurmuşlardır. Bu kuruluş, on altı yaşını geçmiş her İsveç vatandaşının adını, adresini, kazancını, medeni halini, vergiye tâbi işlerini, kira ve alım satım anlaşmalarını ve eğer mevcutsa sabıkalarını bildirebilir. Enstitü tarafından üçüncü şahsa verilen bilginin bir kopyası, hakkında bilgi istenen kişiye mutlaka iletilir.

Genellikle kredi ajansları elde ettikleri bilgilerin doğruluğunu kontrol etmezler. Görevleri yalnızca borçlar ve ödenmeyen kredilerle ilgili bilgileri toplamak ve aktarmaktır. Bu bilgileri isteyen kuruluşlar, müşteriye kredi açmadan önce kendi imkanlarıyla bu konudaki araştırmalarını genişletebilirler.

Her ne kadar bu kredi kayıtları tehlikeli borçlanmalara karşı faydalı ve yasal bir önlem oluşturuyorlarsa da, büyük hatalar yapma ihtimalini de beraberinde getiriyorlar. Bununla ilgili şöyle bir olay anlatılmaktadır: İyi iş yapan bir şirketin başında bulunan bir işadamı, iş olanaklarını genişletmek için kredi isteğine kar-

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

şılık red cevabı alır. Bunun üzerine şaşkına dönen işadamı, şirketin n durumu hakkında araştırma yapar ve kendisine bu şirketle kesinlikle iş yapmaması tavsiye edilir! Araştırmalarını derinleştirdiğinde, kendisiyle aynı ismi taşıyan bir kişinin kısa bir süre önce mahkemeye çıkarıldığını öğrenir.

Toplum için çok önemli olan sosyal hizmetler, kişisel özgürlükleri ihlal etmeye en uygun ortamı yaratır. Sosyal hizmetlerde çalışanlar, bazen hizmet ettikleri kişilerin sıkıntılı anlarında kendilerine güvenerek açıkladıkları çok gizli bilgilere sahip olurlar.

Bir veri bankasında toplanan bu bilgiler, bazen bir memurun tamamen kişisel görüşüne dayanabildiği halde bilimsellik görünümünde olan bir dille ifade edildiğinden, inandırıcı olur. ''Paronoid'' ve ''nörotik'' gibi kavramlar sosyal hizmet kuruluşlarında çalışanların, yardım isteğinde bulunan kişiler hakkında sıkça kullandıkları sözlerdir. Gerçekte bu kişilerin pek çoğu böyle bir sinirsel gerilim içinde olmasalar, zaten yardım istemezlerdi. Fakat böyle bir bilgi kodlanıp bilgisayara verilirse, sadece kayda geçmiş olması nedeniyle bile resmiyet kazanabilmektedir.

Mahkemeler de genel olarak sosyal hizmet kuruluşlarında çalışanların raporlarına derin bilgiye sahip tanığın ifadesiymiş gibi değer vermektedir. Bu nedenle rastlantısal bir ifade kolayca insana ömür boyu yapışıp kalan bire tiket haline dönüşebilir.

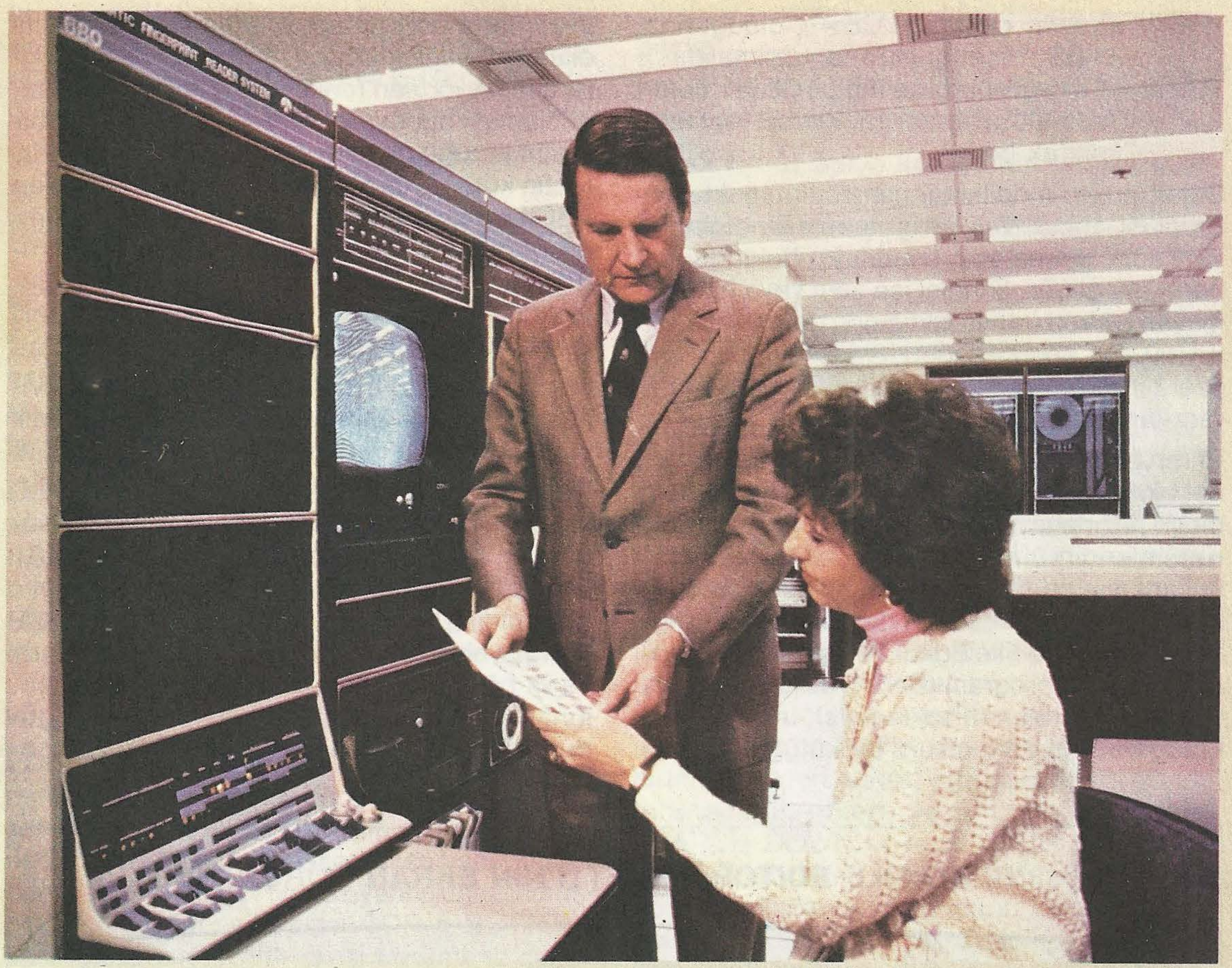
Yerel sosyal hizmet büroları tarafından vatandaşlar hakkında toplanan bilgiler konularına göre sınıflandırılırlar: Eğitim, konut, iş, mali durum, sosyal hizmetler gibi... Mali durum hakkındaki bilgiler, çoğunlukla vergi kayıtlarını saklayan dairelerin arşivlerinde bulunur.

Yerel bürolar ayrıca tek tek bireyler hakkında ekonomik bilgiler de toplar. Bunun amacı, sözkonusu olabilecek 40 ayrı yardım türü arasından hangisinin en doğrusu olacağını belirlemektir.

Bugün yerel sosyal hizmet büroları tarafından toplanan bütün bilgileri bilgisayara verme eğilimi giderek ar maktadır. Her ne kadar her büronun yalnızca kendi alanına ilişkin bilgileri alabilmesini sağlayacak bir kodlama yapmak teorik olarak m ümkün ise de, farklı bürolar, hatta bunların dışında kalan polis örgütü gibi kuruluşlar arasında bile, bilgi değiş tokuşu yapmayı yasaklayan bir kanun yoktur. Gerçekte durum şöyledir: Sosyal yardımdan faydalanmak isteyen bir kişinin dosyası ilk önce bir sürü bürodan geçer. Örneğin, sosyal yardı m organizasyonu; aile doktoru, hastane personeli, öğretmenler, bazı polis ve adliye memurları veya akli kısıtlılık durumu varsa vasisi gibi kimselerin fikrini almadan, bir kişiye yardım edebilme durumunda değildir. Haklarında bilgi toplanan kişiler, kendi kişisel dosyalarını gözden geçirebilme yetkisine sahip olmadıklarından, kendileriyle ilgili, ister doğru, ister yanlış, fakat mutlaka tartışılabilir olan bilgileri öğrenemezler. Dolayısıyla en olmayacak zamanda ortaya çıktığında, muhtemel yanlışlıkları düzeltebilmek veya sağlam olmayan hükümleri belirli bir temele oturtacak kanıtları isteyebilmek durumunda değildirler. Öyle durumlar olmuştur ki, sosyal yardım kuruluşunda çalışan bir memurun yardım talebinde bulunan bir kişi hakkında yazmış olduğu rapor, tam sosyal yardım izninin açıklandığı celsede yüksek sesle okunmuştur. Üstelik bu raporda raporu yazan kişinin, yardım isteyen kişinin birtakım zaafları hakkında kişisel fikri açıklanıyordu. Bir başka olayda da, bir yurtta kalmakta olan bir adam, yapılan bir toplantıda kendisi hakkında sosyal yardım uzmanı tarafından yüksek sesle okunan bir raporu dinlemek zorunda kalmıştı. Bu raporda, doktorunu öldürmek istemekle suçlanmaktaydı. Daha sonra ise raporun içeriğinin gerçeğe uymadığı ortaya çıkmıştır.

Birçok ülkede, özel yaşamın mahremiyetinin korunmasını sağlayan yasaların çıkarılması gerektiğinin bilincine varılmıştır. Bazı ülkeler bunun gereklerini yerine getirmeye başlamışlardır. ABD'de, eyalet dairelerinde saklanmakta olan kişisel dosyalara serbestçe erişebilme olanağı sağlanmakla beraber, bunlardaki bilgilere itiraz etmek veya düzeltmek hakkını tanıyan çok sayıda yasa vardır. Kişisel verileri k aydetmeye yönelik, merkezileştirilmiş ve ileri derecede otomatikleştirilmiş bir sisteme sahip olan İsveç'te de, özel bilgileri elde etmeyle ilgili belli bir gelenek sürdürülmektedir. Ayrıca, veri bankası kurmak isteyen her kuruluşun denetlenmesini ve lisans alma zorunluluğunu öngören bir yasa hükmü vardır. Federal Almanya'daki yasalara göre, bilgisayarlara kaydedilen kişisel veriler beş yıl sonra silinmek veya tamamen bir kenara atılmak zorundadır. Fransa ve Hollanda'da gerekli yet-

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



FBI'ın veri bankaları bilgisayara bağlanmıştır. Fotoğrafta: Parmak izi kütüğünün bir bölümü.

ki olmaksızın veri toplanmasını yasaklayan yasalar vardır. Fransa'da ayrıca herhangi bir şahsın etnik kökenini, politik veya dinsel eğilimini içeren bilgilerin bilgisayara kaydedilmesini yasaklayan bir yasa da bulunmaktadır.

Danimarka'da yürürlükte olan bir yasa, vatandaşların özel hayatına ilişkin bilgilerin yetki alınmadan yaygınlaştırılmasını yasaklamaktadır. Bilgilerin elde edilme yollarına ilişkin yasal denetimlerin gerekliliğini savunan fikirler yaygınlık kazanmaktadır. Bu yolların her halükarda en aza indirgenmesi ve yetki sınırları içine çekilmesi gerekmektedir. Ayrıca, belli bir amaca yönelik olarak toplanan bilgiler başka hiçbir amaç için kullanılmamalı veya bunun için mutlaka ilgili kişinin izni alınmalıdır. Kişisel verilerin kayıt sistemleri kendi aralarında birleştirilmemeli ve bu bilgileri toplamakla görevli kişiler kendilerinin üstünde bulunan bir makamın denetimi altında, belirli davranış kurallarına uymalıdırlar.

İngiltere'de bu alanda karşılaşılan bazı eksiklikler, ''Consumer Credit Act''ın 1974'te yürürlüğe girmesiyle giderilmiştir. Bugün İtalya'da bir kredi talebinde bulunan kişi, bilgilendirme daireleri tarafından kendisi hakkında bilgi aktarıldığı konusunda haberdar edilir.

Ama diğer yandan edinilmesi yerinde olmayan ve araştırmaya dayanan bilgileri (örneğin kişilerin politik görüşleri ve cinsel yaşamları ile ilgili bilgiler) toplayan ve satan ajanslara yönelik kısıtlamalar konulmamıştır. Ayrıca bilgilerin ne kadar zaman saklanabileceği de yasal olarak saptanmamıştır. Örneğin bu bilgiler çoktan ödenmiş borçlara ilişkin de olabilir.

Verilerin daha iyi korunabilmesi ile ilgili tasarımlar birbirine birçok bakımdan benzemekte ise de, bunların nasıl gerçekleştirilebileceğini şimdiden tahmin etmek zordur.

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

YCE = Yazım çentiği; YCE = 1 olduğunda YB$ içeriği ekrana yazılmalıdır.
TCE = Ters görüntü çentiği; TCE = 1 bulunan alt-program çağırılırsa yazı ters olarak belirir.

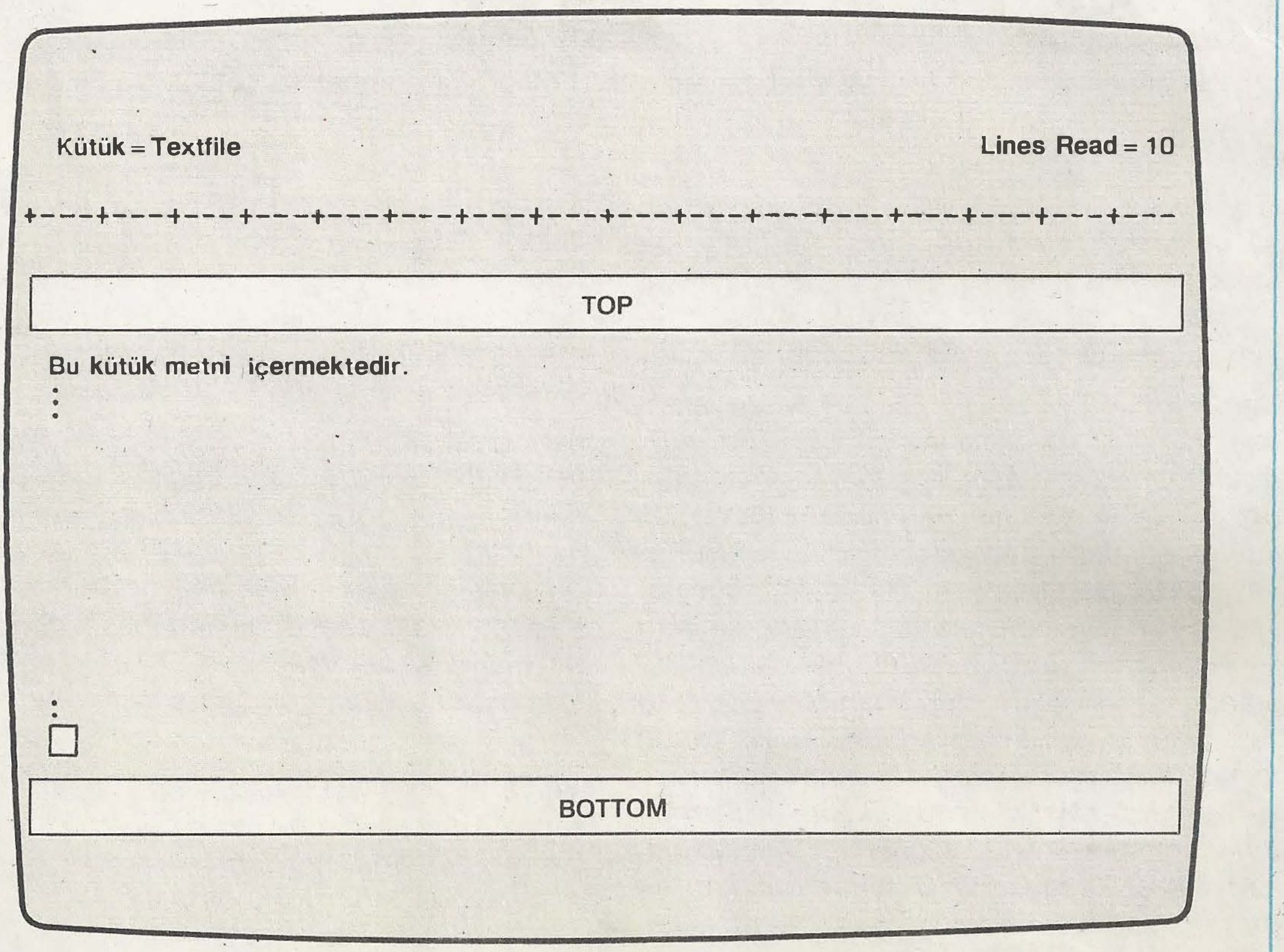
Bu alt-programda iki tür çıkış bulunmaktadır. Birincisi ancak veri girişi tamamlandığında, yani sözü geçen alan tamamıyla dolduğunda harekete geçer. İkincisi ise sayı veya harf olmayan bir kod ortaya çıktığında gerçekleşir (6. nokta). Bu kodun değeri, bu işi yapmak için özel bir fonksiyon kullanan bir çağırıcı program tarafından çağırılabilir.

Ekran biriminin bir bütün olarak işletilmesi, genel bir bakış vermeyi ve bilgisayar yardımıyla metin hazırlanması veya verilerin grafiksel görünümlerinin üretilmesi için özel programlar kullanmayı mümkün kılar. İlgileneceğimiz ilk örnek, Olivetti firmasının bir uygulayıcı programı olan Video File Editor ile ilgilidir. Bu program, ister bir program (BASIC dilinde yazılmış ve ASCII formatında saklanmış), ister bir veri kütüğü sözkonusu olsun, bir kütüğün yazımı veya hatta değiştirilmesi aşamasında, çeşitli olanaklar sağlamaktadır. Açıklamak için, programda öngörülen fonksiyonları nitel olarak tanımlamayı deneyelim. Video File Editor ile bir **metin kütüğü** oluşturulabilir ve değiştirilebilir. Metin kütüğü kastedilen bilgiyazımı yapabilen ASCII karakterleriyle oluşturulan ve veri cümlelerinden meydana gelmiş bir kütüktür. Kütüğü oluşturan cümleler RETURN/LF (Return/Line Feed) veya RS (Record Separator) karakterleriyle biribirinden ayrılır.

Eğer (PCOS'un EDIT güdümü yardımıyla) Video File Editor'e girilirse ve daha önce belli bir kütük tanımlanmamış ise yeni bir kütük oluşturulur ve disketten belleğe çağırılabilir.

Video File Editor devamlı 24 satır ve 80 sütundan oluşan bir ''pencere'' oluşturur (21 satır kütük için ayrılmıştır, diğerleri ''hizmet'' satırlarıdır). Kütüğün istenen bir bölümünü görüntüleyebilen bu pencere, başlangıçta yalnızca ilk bölümü gösterir. Kullanıcı, bellekte bulunan Video File Editor'un fonksiyon ve güdümleri yardımıyla görüntülenen kütüğün sa-

VIDEO FILE EDITOR OLİVETTİ'NİN EKRAN GÖRÜNTÜSÜ



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

tırlarını değiştirebilir. Kütükte yapılan değişiklikler Video File Editor'un çıkışında disket üzerine kaydedilebilir. Video File Editor çağırıldığında s.616'da gördüğümüz görüntü belirir.

İlk satır kütüğün adını ve o andaki bilgiyi gösterir.

İkinci satır başlangıçta henüz boştur. Bu satır daha sonra hem karakter zincirleri aramakta, hem de güdüm girişlerinde kullanılabilir.

Üçüncü satır ise ''cetvel'' olarak adlandırılan satırdır. Sütunların pozisyonlarını ve ekran alanlarını gösterir. Ekran, her biri dört karakterden meydana gelen alanlara ayrılmıştır.

Dördüncüden, yirmidördüncüye kadarki 21 satır metin penceresini (text window) oluştururlar. Yirmibeşinci satır kullanılmaz. Kütüğün başlangıç ve bitimi iki ''gizli'' satır ile (TOP ve BOTTOM) belirtilir. Bunlar metne dahil değildir, yalnızca nirengi noktaları olarak işgörür ve açık fon üzerine koyu olarak yazılmış TOP ve BOTTOM kelimelerini içerirler.

EDIT komutu yardımıyla Video File Editor'a girişte kütüğün ilk satırından hemen önce TOP satırı belirir ve imleç bu satır üzerinde bulunur. BOTTOM satırı ise metnin son satırından hemen sonra belirir. Henüz yeni oluşturulmuş bir kütük boştur ve yalnızca TOP ve BOTTOM satırlarını gösterir. Veri girişi konumuna geçildiğinde, imleç derhal şeklini değiştirir. Karakterin altındaki küçük çizgi, karakterin solunda küçük bir üçgene dönüşür. Her iki durumda da imleç yanıp söner. Video File Editor'a giriş sırasında klavye, çeşitli çıkış hazırlığı (editing) fonksiyonlarının uygun şekilde programlanmış bir dizi fonksiyon tuşu tarafından harekete geçirilebilmelerini sağlayabilecek farklı bir işlev görür.

Sayfa 618'de klavyenin bir görünümünü ve sayfa 619'dan sayfa 623'e kadar Video File Editor'un önemli fonksiyonlarını görmekteyiz.

Ekranın cetvel satırının ikinci satırına **''güdüm satırı''** denir. Karakter zinciri aramalarında ve yüksek düzeyde güdümlerin girişinde kullanılır. Bu satır üzerinde karakterleri görünür hale getirmek için, kullanıcı önce imleci satır başına getiren COMMAND MODE (COMMAND + 1) fonksiyonunu vermelidir. INSERT MODE, BACK SPACE, DELETE CHAR,. RECALL LINE... vb. bütün çıkış hazırlığı işlemleri güdüm satırının bu noktasından itibaren kullanılır. Güdüm durumunda kullanılan RECALL LINE, güdüm satırının o ana kadarki içeriğini tekrar verir. Oysa bütün diğer fonksiyonlar yalnızca metin kütüğünde uygulanabilirler.

Komut durumunda ↵ tuşunun işlevi, INSERT LINE değil, EXECUTE COMMAND'dır.

İmleci tekrar metin kütüğünün başına getirmek ve çıkış hazırlığı işlemlerini tekrar kütüğe geçirmek için, yeniden COMMAND MODE komutu verilmelidir.

Biçimleme (Editing) ve Karakter Zinciri Araması İçin Fonksiyonlar

Sayfa 619'dan 623'e kadar Video File Editor'un biçimleme güdümlerinin çalışma şekli, farklı fonksiyonların herhangi bir metin üzerindeki etkileri ile birlikte gösterilmiştir. Karakter zinciri aramasının kullanımı bütün bir kütüğü belli karakter zincirine göre taramayı mümkün kılar. Kullanıcı, aramayı başlatmak için öncelikle COMMAND MODE fonksiyonunu ve daha sonra aranan karakter zincirini vermelidir; bunun sonucunda karakter zinciri ekranın ikinci satırında belirir. Kullanıcı karakter zincirini verdikten sonra, kütüğün hangi yönde taranacağını (ileri veya geri) belirleyen SEARCH DOWN veya SEARCH UP fonksiyonlarından birini verir. Aramanın başlangıç noktası imlecin pozisyonudur. Daha önce de belirtildiği gibi Video File Editore'a giriş EDIT komutuyla sağlanır. Böylelikle PCOS (işletim sistemi) terkedilmiş olur. Kullanıcı PCOS'a tekrar dönebilmek için, EXIT AND SAVE (CTRL + 6)'i vermelidir; böylece en son haline getirilmiş metin kütüğü de diskete kaydedilir. Bundan sonra ya RETURN ya da ABORT (COMMAND + 6) ve RETURN verilmelidir.

Eğer kullanıcı SAVE TEXT (CTRL + 5) verirse, halihazırdaki metin kütüğü diskete kaydedilir, fakat Video File Editor'dan çıkılmamış olur.

Menüler

Her yöntemde, hatta en basitinde bile, özgün birimler oluşturan bir miktar işlevsel öbek saptanabilir. Farklı öbekler, daima aynı verilerle çalışır ve aynı problemin çözümü için birbirleriyle yarışırlar. Fakat her biri, işlenmekte olan bütünün bir özelliğini veya bir yönünü ele alır.

Yöntemin kullanımı, bu yöntemi oluşturan farklı bölümlerin ya işin gereklerine ya da program tarafından belirlenmiş zamansal önceliğe göre yürütülmesinden ibarettir. İlk durumda diğer bölümlerin yürütülmesi kullanıcıya tabi kılınmıştır ve kullanıcı belli bir sırayı iz-

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

VİDEO FILE EDITOR PROGRAMININ FONKSİYON TUŞLARI



Bu tuşların kullanımı için, tuşla birlikte aynı anda COMMAND tuşuna basmak gereklidir.

Command Mode (1)	Delete Line (2)	Split Line (3)	Start of Line (4)	Restore Lines (5)	Abort (6)	Insert Mark (7)	Search Up (8)	Line Up (9)	Half Screen Up (0)	Full Screen Up (−)	Top (^)
Insert Mode (1)	Delete Char (2)	Join Lines (3)	End of Line (4)	Save Text (5)	Exit and Save (6)	Reverse Tab (7)	Search Down (8)	Line Down (9)	Half Screen Down (0)	Full Screen Down (−)	Bottom (^)

Bu tuşların kullanılması için, tuşla birlikte aynı anda CTRL tuşuna basmak gereklidir.

COMMAND

CTRL

SHIFT

H - Backspace
K - Erase to End
L - Refresh Display
R - Recall Line

Bu tuşların kullanılması için, tuşla birlikte aynı anda CTRL tuşuna basmak gereklidir.

Tab (S2)

Backspace (S1)

Insert Line or Execute Command

(7)	Cursor Up (8)	(9)	(+)
Cursor Left (4)	Goto Mark (5)	Cursor Right (6)	(−)
Next Line (1)	Cursor Down (2)	Escape (3)	(*)
(/)	(0)	(00)	(\)

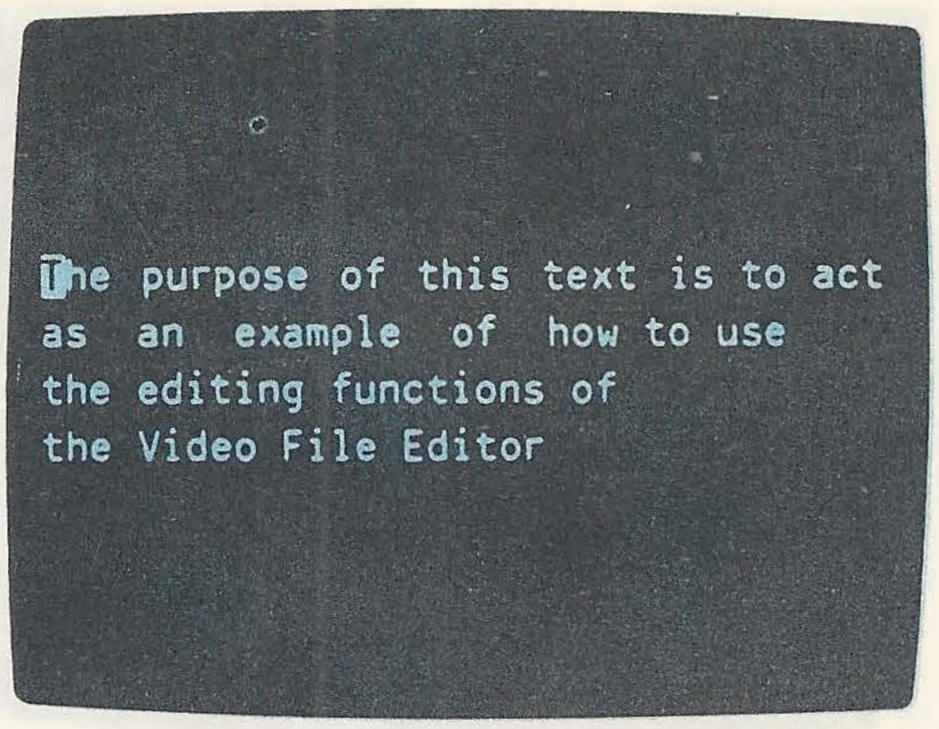
Bu tuşların kullanılması için, tuşla birlikte aynı anda SHIFT tuşuna basmak gereklidir.

Parantez içindeki simgeler USA-ASCII klavyesinin tuşları üzerindeki kısaltmalardır.

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

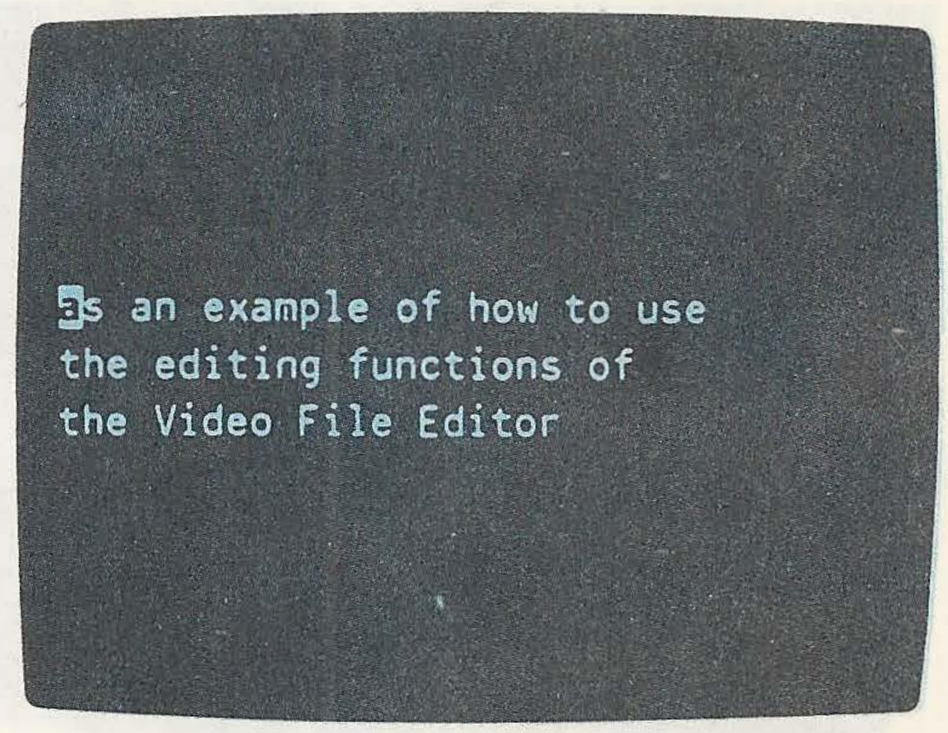
VİDEO FILE EDITOR OLİVETTİ PROGRAMININ BİÇİMLEME VE KARAKTER ZİNCİRİ ARAMA FONKSİYONLARI

Bu ve bunu takip eden sayfalarda, Video File Editor programının biçimleme (editing) ve karakter zinciri arama fonksiyonları için bazı örnekler verilmektedir. Sağda değiştirilecek metin kütüğünü içeren ekran görüntüsü bulunmaktadır. Altta sol kısımda girilen fonksiyonun açıklamaları aynı hizada sağda ise metin üzerindeki sonuçları verilmektedir.



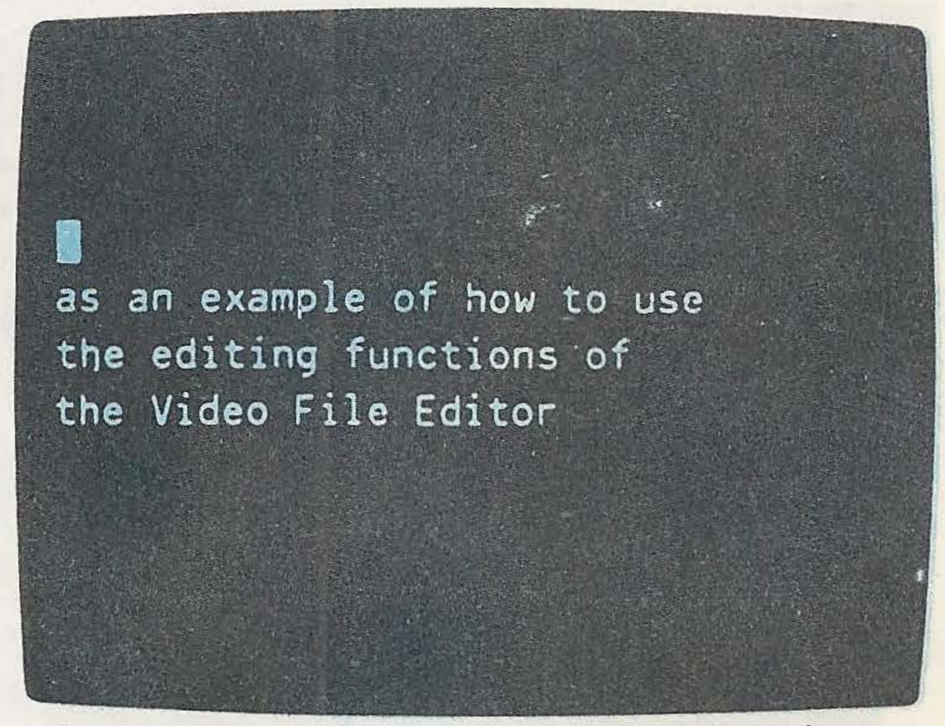


DELETE LINE
İmlecin bulunduğu satır silinir.



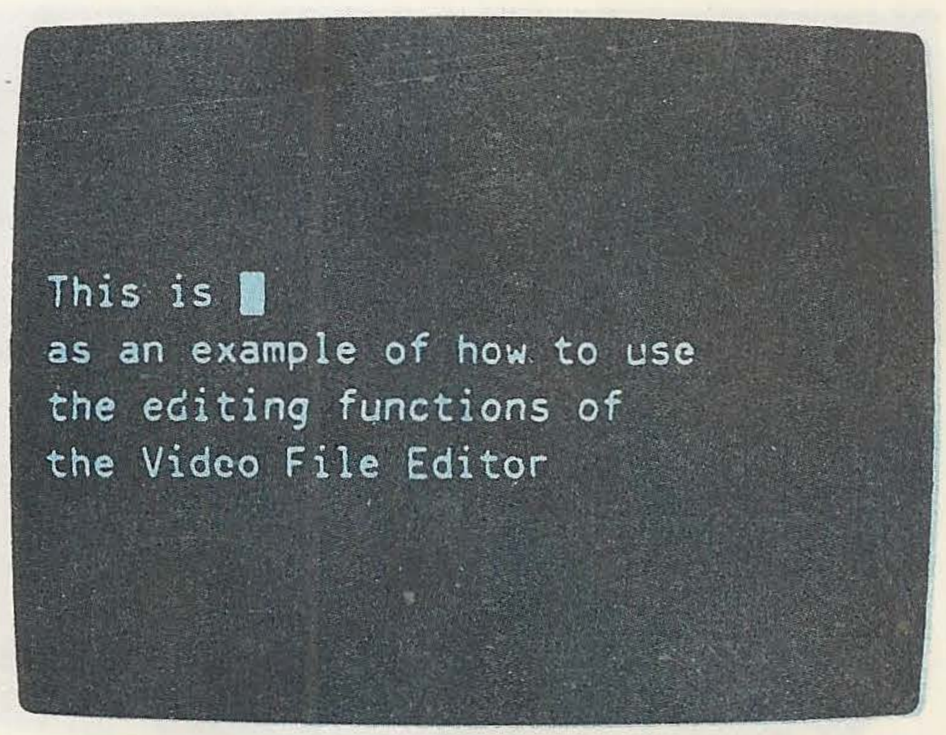


CURSOR UP
İmleç bir üstteki satıra çıkar.





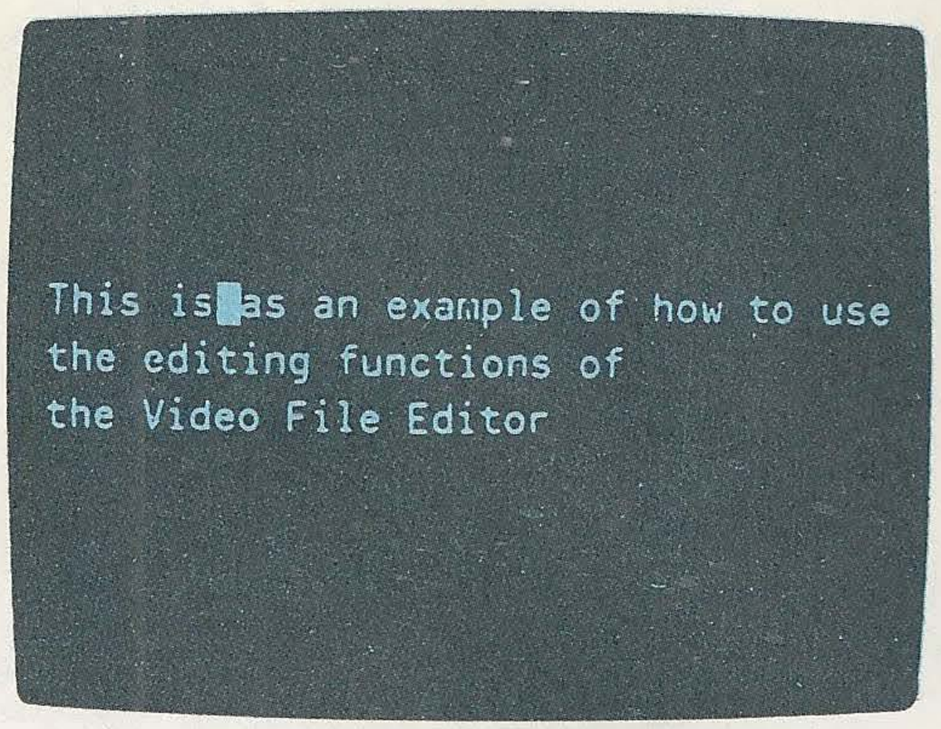
INSERT MODE
İmlecin bulunduğu satıra **This is** karakter zinciri girilir.



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

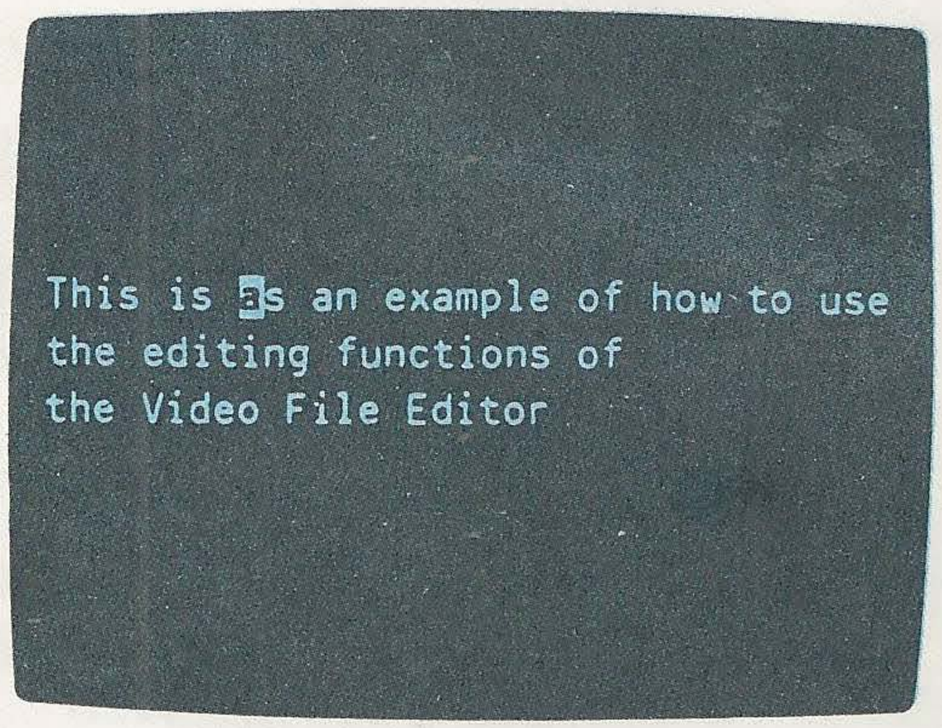


JOIN LINES
İmlecin bulunduğu satır, takip eden satır ile birleştirilir.



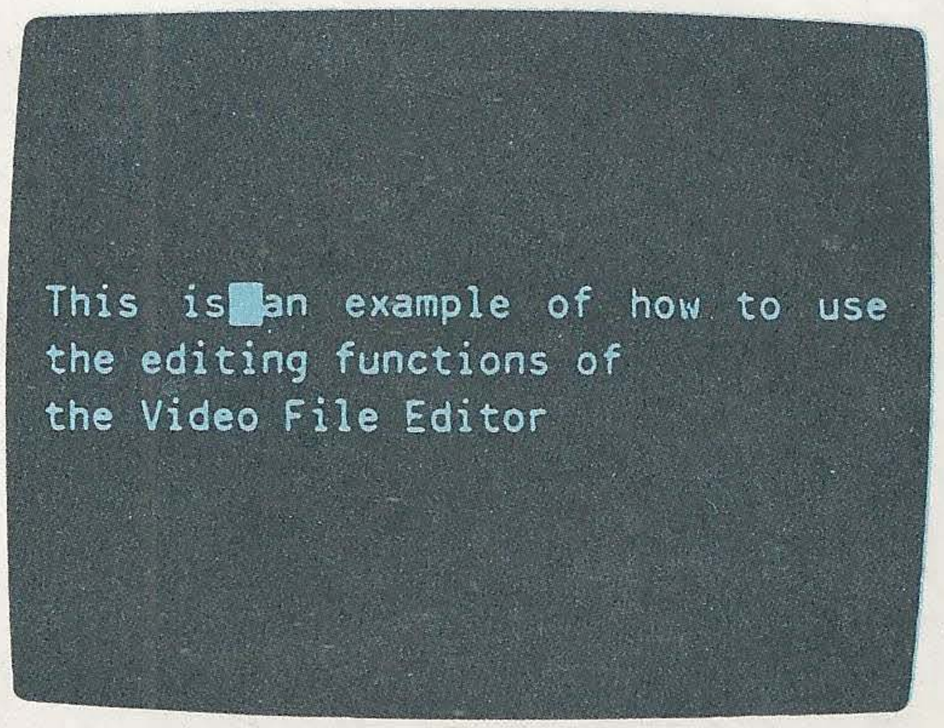


CURSOR RIGHT
İmleç, bir pozisyon sağa kaydırılır.



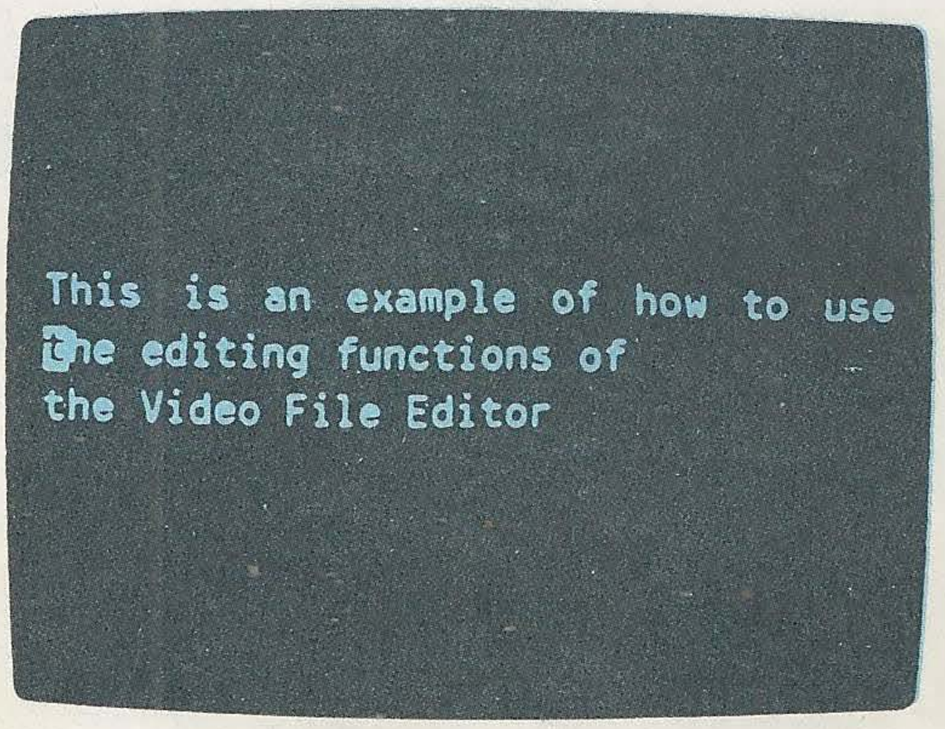


DELETE CHAR
İmlecin bulunduğu pozisyondaki karakter silinir ve o satırda bulunan sonraki karakterler birer pozisyon sağa kayar. Burada işlem üç kez tekrarlanmıştır.





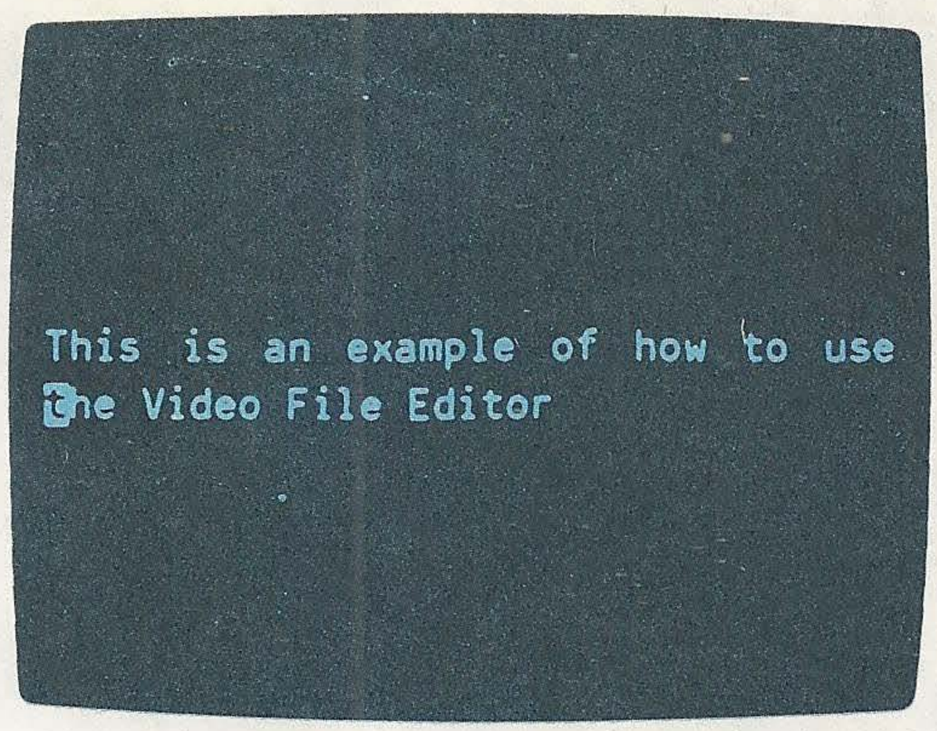
NEXT LINE
Metin, bir satır yukarı kaydırılır. İmleç, daha önce bulunduğu satırı takip eden satırın başına kayar.



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

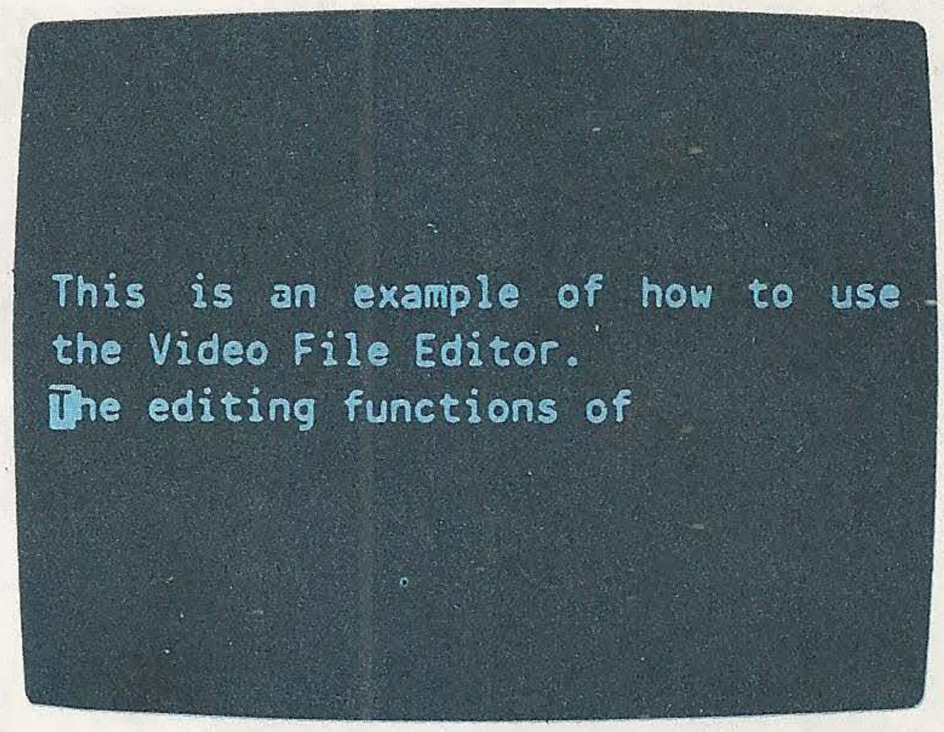


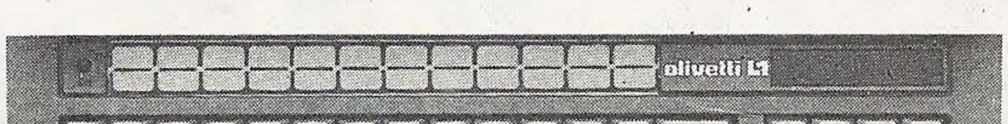
DELETE LINE
İmlecin bulunduğu satır silinir.



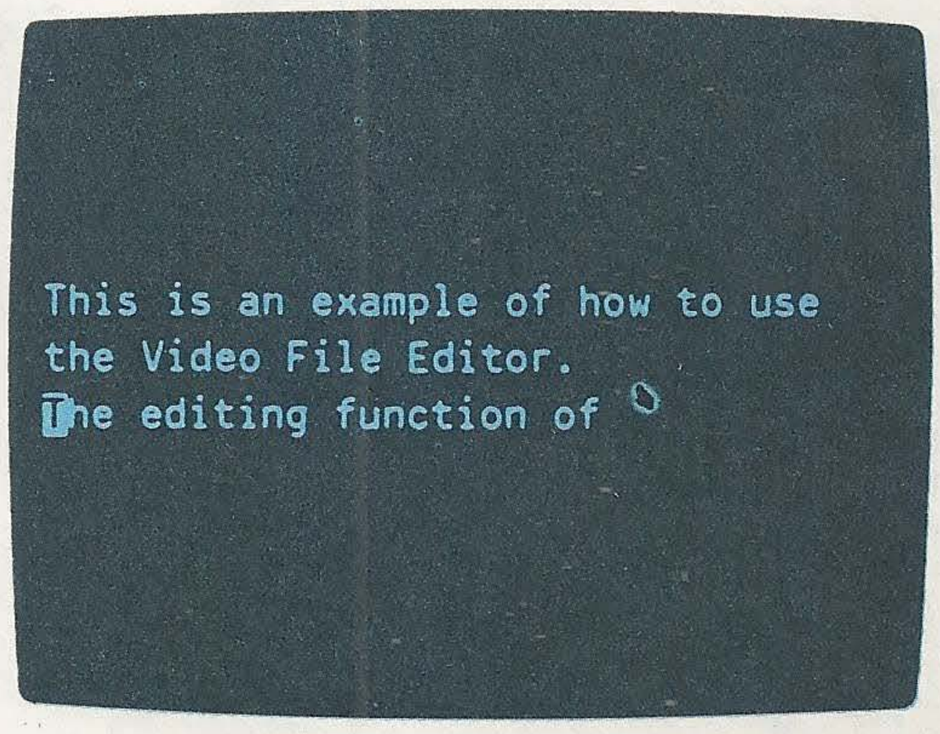


RESTORE LINES
NEXT LINE
Metin kütüğünün bir sonraki satırı ekrana alınır.



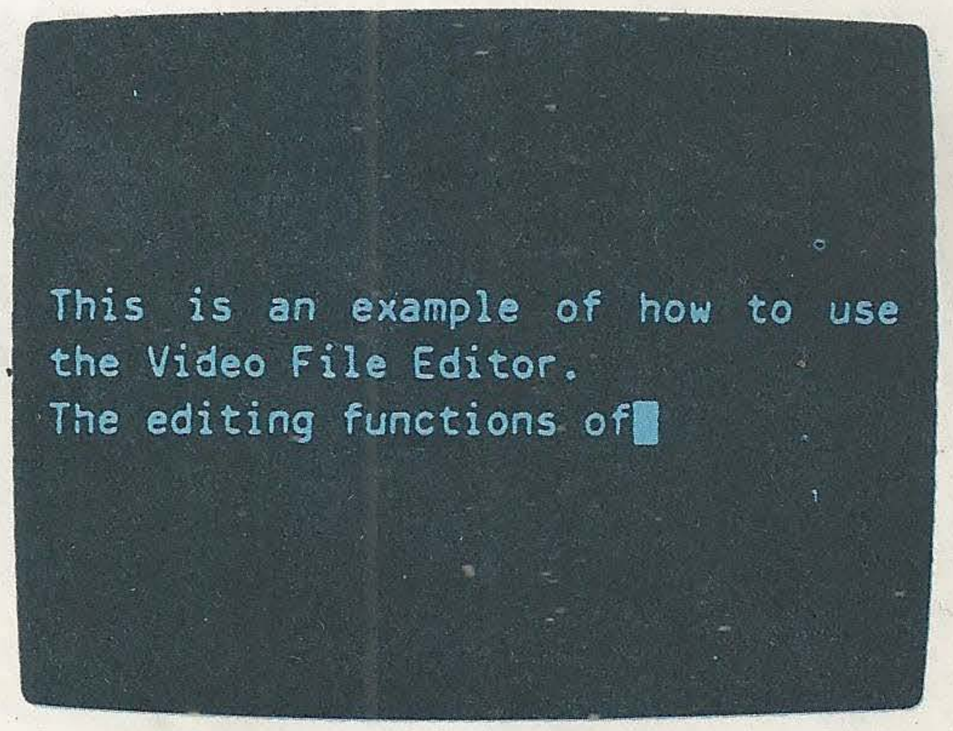


INSERT MODE
Değiştirme durumundan veri girişi durumuna geçiş.





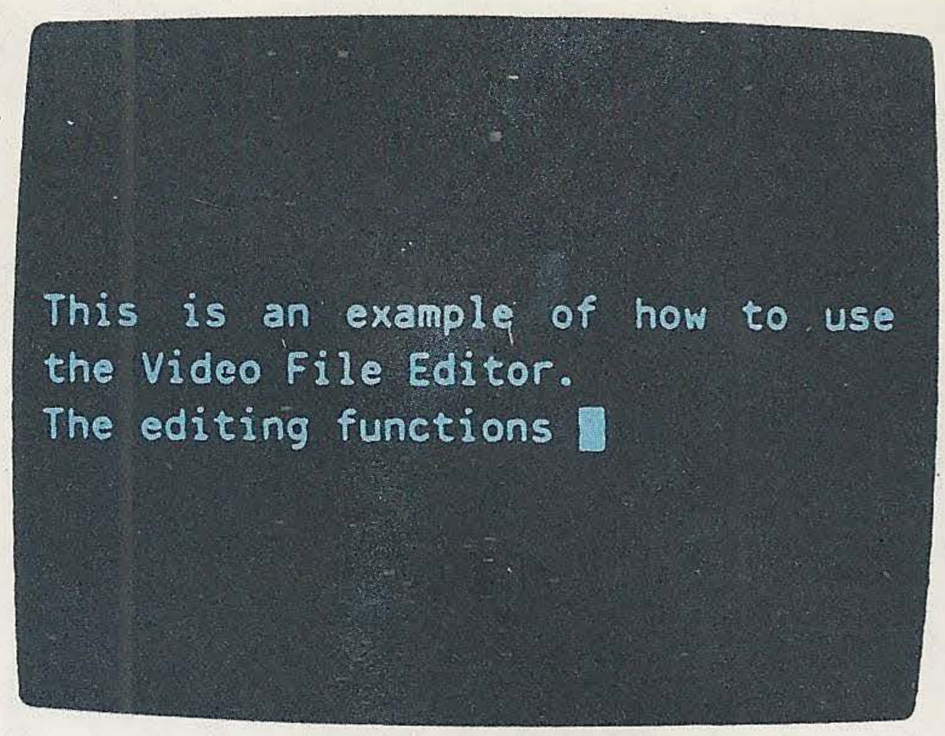
END OF LINE
İmleç satırın son karakterini (boşluk karakteri hariç) takip eden pozisyona getirilir.



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

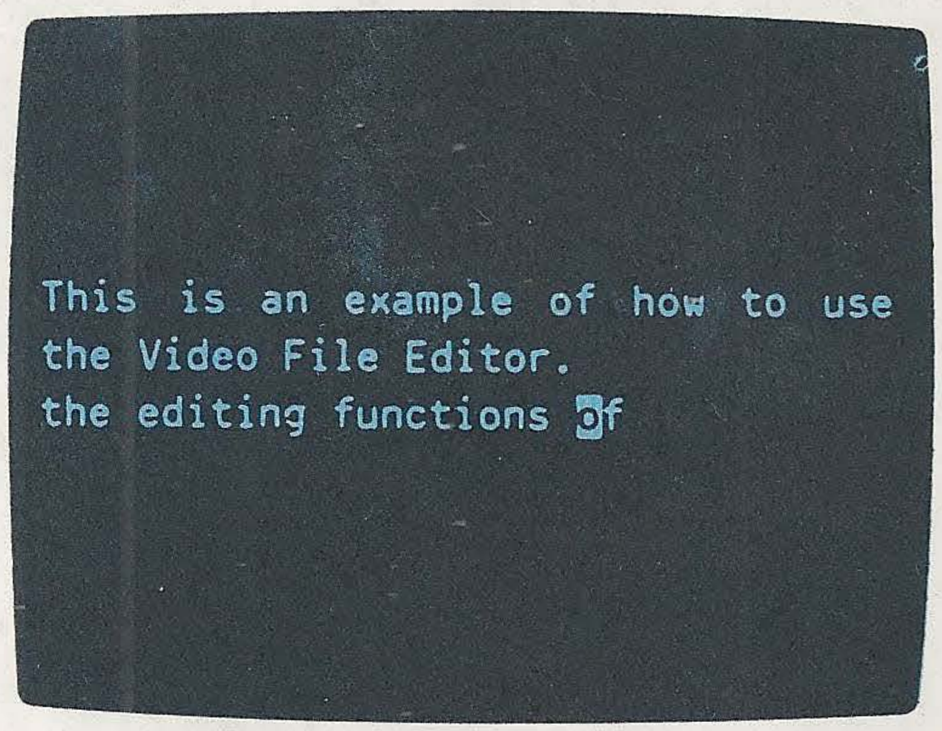


BACKSPACE
İmleç bir pozisyon sola kayar ve imlecin kaydığı yerdeki karakter silinir. İşlem burada iki kere tekrarlanmıştır.



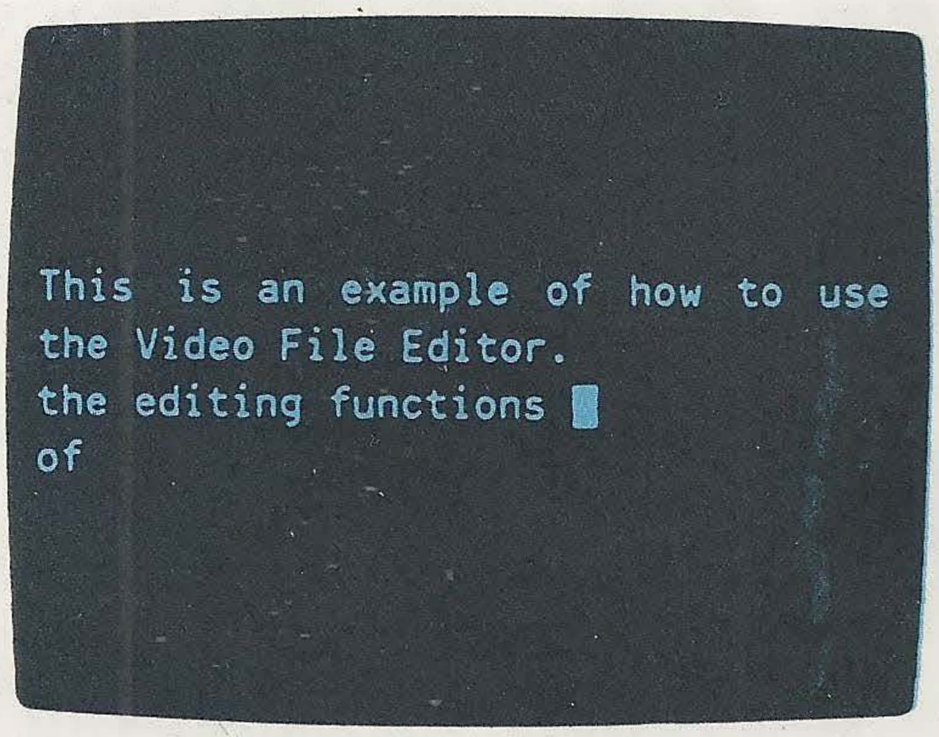


RECALL LINE
Satırın daha önceki içeriği yeniden oluşturulur. İmleç daha önce bulunduğu noktada kalır.



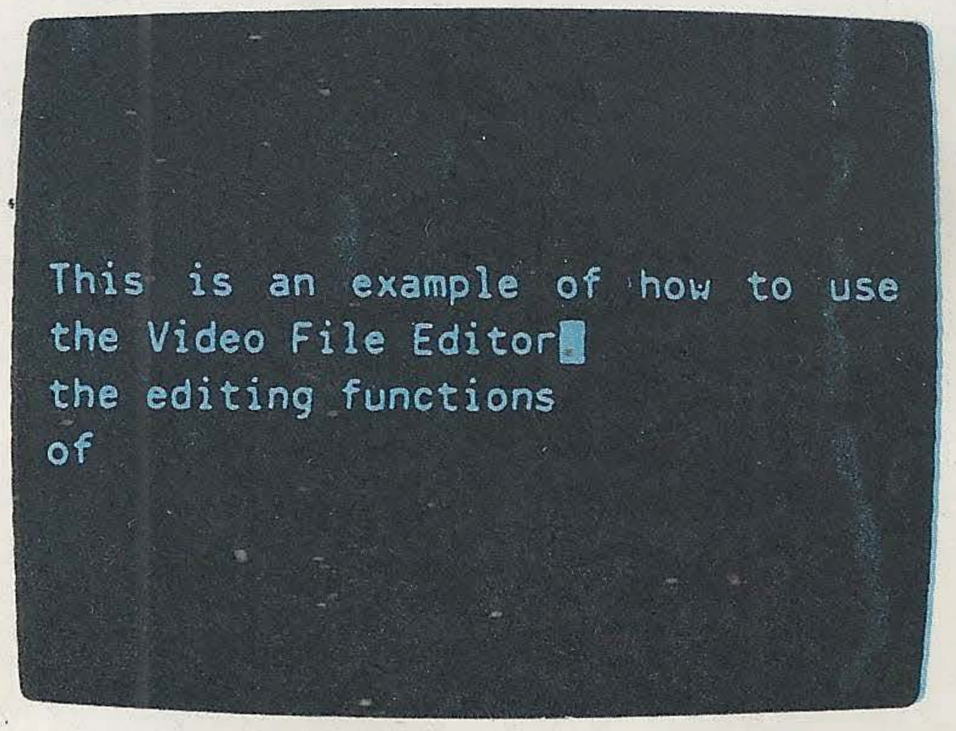


SPLIT LINE
Bulunduğu satırda imlecin sağında kalan kısım (üzerinde bulunduğu karakterden itibaren), bir sonraki satırın başına yerleştirilir.





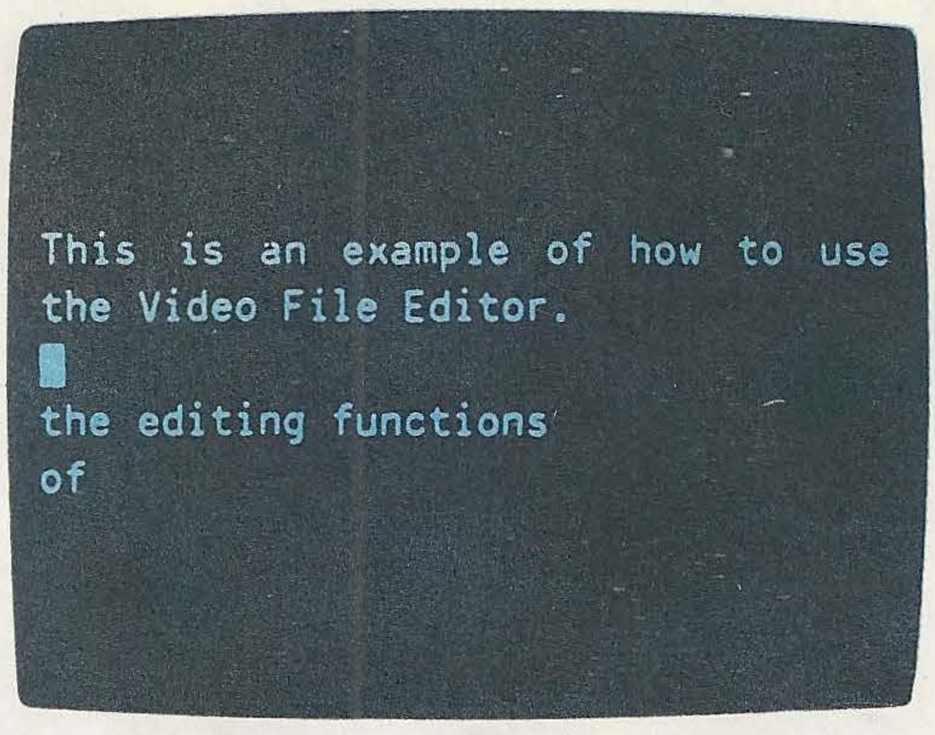
CURSOR UP
İmleç bir üstteki satıra çıkar.



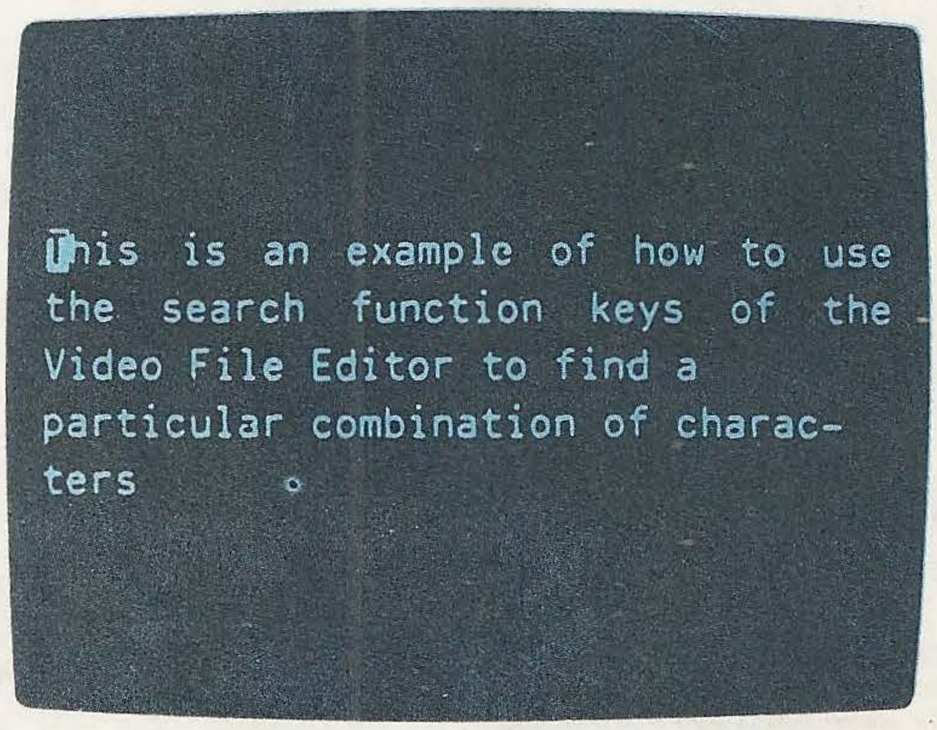
Ücretsiz PDF Kitap Arsivi - https://rantey.org



INSERT LINE
İmlecin bulunduğu satırın hemen altında boş bir satır oluşur ve imleç bu boş satırın başına gider.

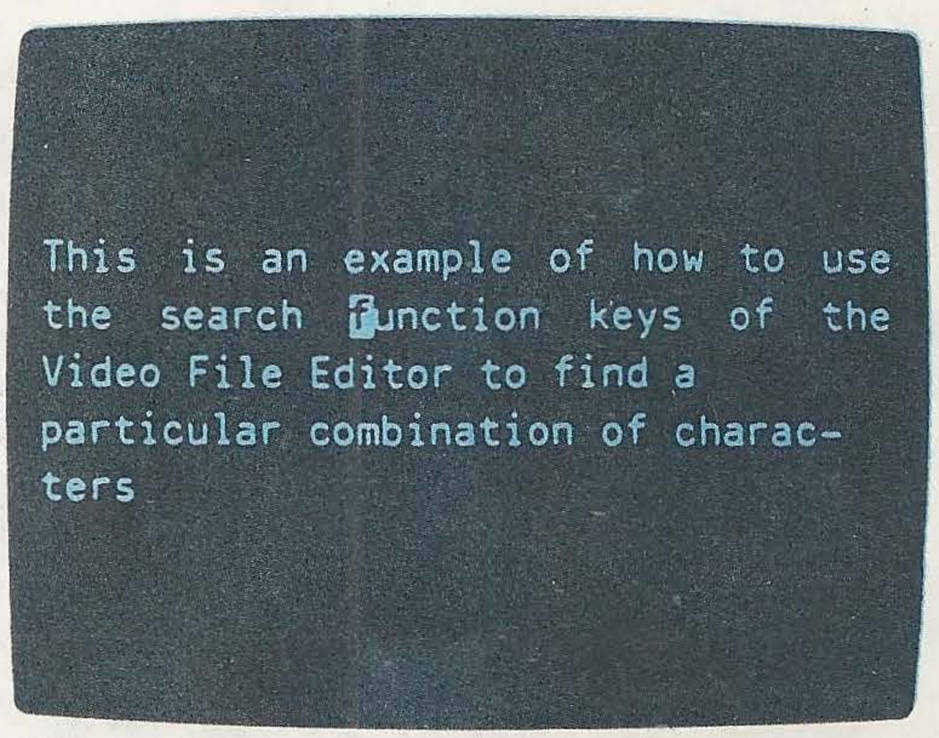


Sonraki iki örnekte karakter zinciri aramalarına iki örnek verilmiştir. Sağ tarafta bulunan metinde önce ''func'', ardından ''e of'' karakter zinciri aranacaktır.



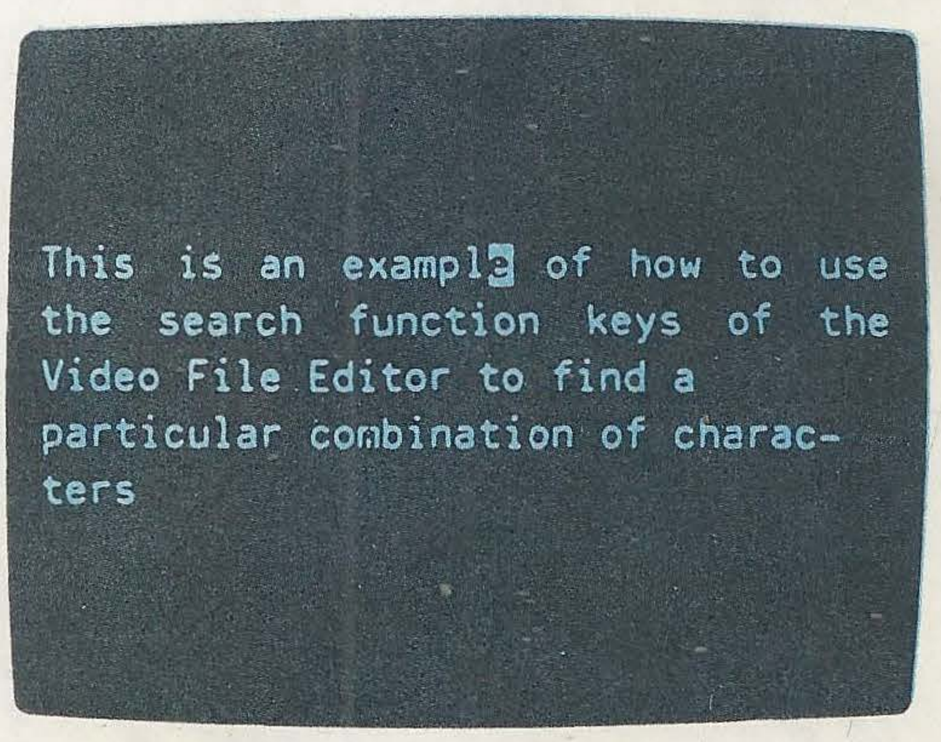


Ekranın ikinci satırına ''func'' yazılarak SEARCH DOWN verilince, Video File Editor imlecin bulunduğu pozisyondan başlayarak metnin sonu yönünde ''func'' karakter zincirini arar.





Ekranın ikinci satırına ''e of'' yazılarak SEARCH UP verilince, Video File Editor ''e of'' karakter zincirini imlecin bulunduğu pozisyondan başlayarak metnin başlangıcı yönünde arar.



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

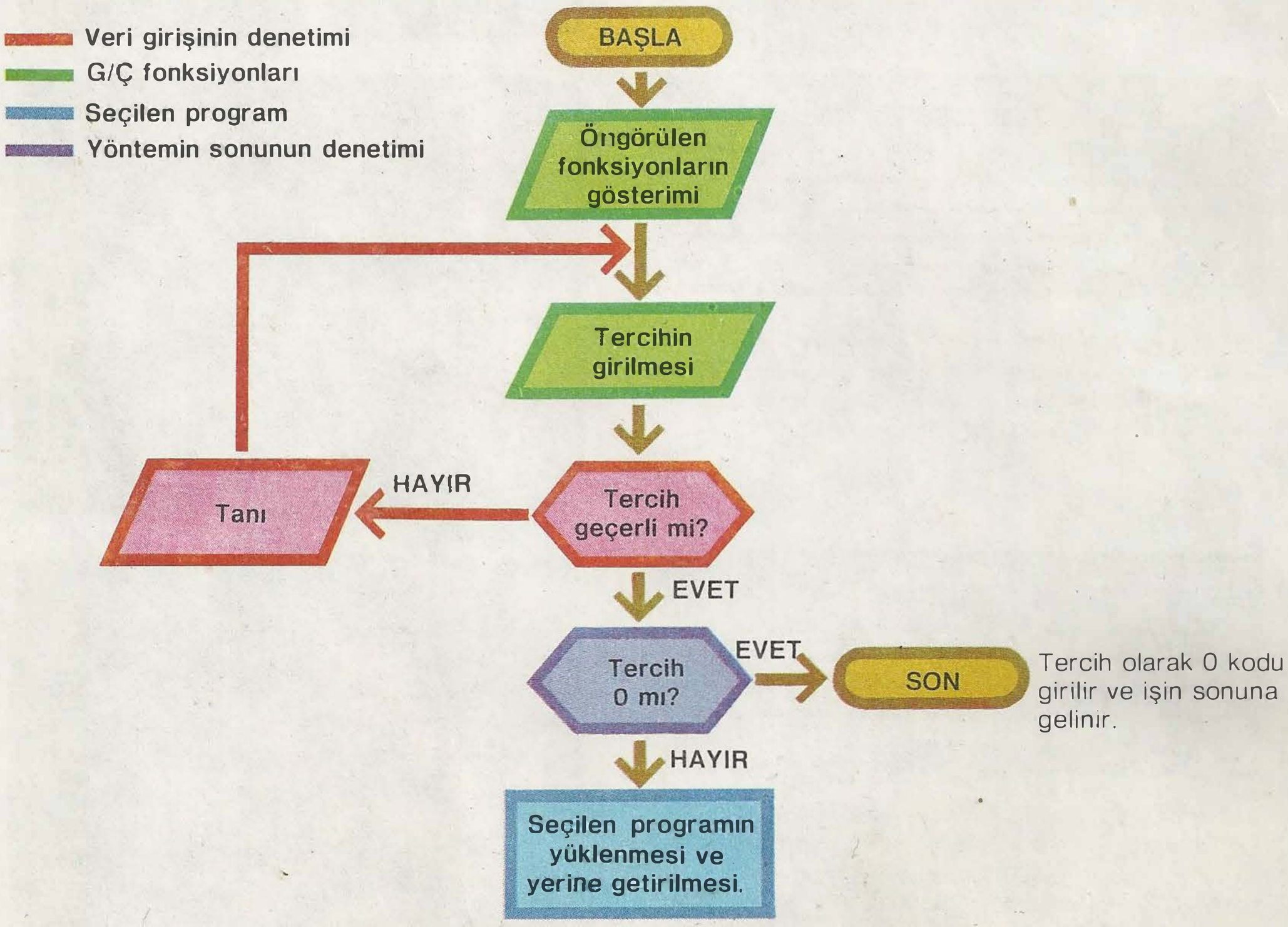
lemek zorunda olmaksızın veri kaydına serbestçe müdahale edebilir. İkinci durumda ise veri işlemlerinin özel gereksinimleri, bir sıra izleyen belirli kuralları dikkate almayı gerektirir. Bu nedenle fonksiyonların seçimi program tarafından denetlenmelidir. Her durumda kullanıcının aşağıdakileri yerine getirebilecek yardımcı malzemeye ihtiyacı vardır:

- Yöntemde öngörülen fonksiyonların liste halinde gösterilmesi.
- Kullanıcı tarafından yapılan seçimin kabul edilip denetlenmesi.
- Bir soru karşısında yöntemin kendisinin veya yöntemi oluşturan programların özelliklerinin ve fonksiyon çeşitlerinin gösterilmesi.

Benzeri bir yapı, programcı için de yararlıdır, çünkü böylece ana programın küçük bölümler halinde gruplandırılması ve bunların yazılması kolaylaşır; bellek yerleşiminde ortaya çıkabilecek sorunlar da, azaltılmış olur. Yukarıda belirtilen gereksinimleri karşılamak üzere, çeşitli fonksiyonların seçimini doğru bir şekilde denetleyerek yürüten programa "Menü" denir. Yapı olarak bir menü çok basittir. Olası fonksiyonların tanımı ve o anda istenen fonksiyon kodunun girişinin sorulmasından ibarettir. Menü programı, verilen cevaba göre, istenen program kısmının yüklenmesini ve yürütülmesini sağlar.

Bu sayfada, aşağıda genel bir menünün akış şemasını görüyoruz. Kullanıcı tarafından yapılan seçime göre, menünün yüklendiği bellek alanını da işgal eden uygun program yüklenir.

GENEL BİR MENÜNÜN AKIŞ ŞEMASI



Sonda çağırılan program, bir sonraki tercih için menüyü yeniden yüklemelidir.

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

Öngörülen fonksiyonların sonunda yeni bir seçim yapılabilmesi için, çağrılan program menüyü yeniden belleğe yüklemelidir. Bunu sağlayan komut şöyledir:

RUN ''X:ISIM''

X, içinde çağrılan programın bulunduğu (A/B ya da 0/1) disket birimini belirtir; ISIM ise yürütülecek programın disket üzerinde tanımlandığı isimdir. Tercih olasılıklarını artırmak ve istenen fonksiyonun aranmasını kolaylaştırmak için, menüler sırayla çağrılabilir.

Örneğin bir ana menü, her biri belli fonksiyon kategorilerini kapsayan belirli sayıda ikincil menüyü bu ikincil menüler de özel menüleri çağırabilir. Her özel menü, sonunda kendisini çağırmış olan ikincil menüyü ve her ikincil menü de ana programı çağırmalıdır. Böylece iki yönde seçim yapılabilmektedir: Genelden detaylara doğru ve ters yönde özel bir uygulamadan genele. Bu ve sonraki sayfada diğer menüleri çağıran bir ana menünün dökümü verilmiştir.

Bu yöntem, kullanıldığı sürecin tümü için bir başlangıç noktası oluşturduğundan, içersinde programların hepsi için geçerli olan fonksiyonlar öngörülmüş olmalıdır: Örneğin, fonksiyon tuşlarının sıfırlanması veya imlece farklı bir şekil verilmesi. Burada, fonksiyon tuşlarının yalnızca sıfırlanmakta olduğuna, yani kendilerine sayısal bir kod verilmekte olduğuna dikkat edilmelidir. Daha sonra bu kod altında yürütülecek fonksiyon, programın kendisi tarafından belirlenir. Örneğin ana menüde bir tuşa 2 sayısı verilmiş olabilir, bu kod bir programda bilgiyazım fonksiyonlarını yönetirken bir başkasında işlemi terketme için bir anahtar görevi yapabilir vb.

625-626. sayfalarda bulunan program dökümü, bir makinede imlecin ve fonksiyon tuşlarının işletimi için gerekli komutları içermektedir. Başka bir sistem kullanıldığında komutlar değiştirilmelidir, çünkü daha önce de belirtildiği gibi, kullanılacak kodlar her makinede farklıdır.

ANA MENÜ

```
4540 ' KUTUK=MENU
4541 '*******KUTUKLER********
4542 '
4543 '  AD          DB    ICERIK                BAYT      ALANLAR
4544 '
4545 ' MULI         A     MUSTERI LISTESI       128         7
4546 ' TOLI         A     TOPTANCI LISTESI      128         7
4547 ' HAST         A     HAFTALIK STOK          91         9
4548 ' YIST         A     ONCEKI YILDAN STOK    100         10
4549 ' HESA         B     HESAPLAR               70         9
4550 ' SUGE         B     SUREKLI GELIRLER       70         9
4551 ' SUGI         B     SUREKLI GIDERLER       70         9
4552 ' KRGE         B     KREDI SENETLERI        70         9
4553 ' KDVE         A     KDV-PAYI              128         7
4554 ' KRPA         A     KREDI-PAYI            128         7
4555 '
4556 LPRINT CHR$(27)+CHR$(62)      '220
4560 GOSUB 6040   'FONKSIYON TUSLARININ BELIRLENMESI
4580 BI$=CHR$(27)+"+"+CHR$(7)
4585 PRINT CHR$(27);"*"
4600 GOSUB 4960
4620 ON V GOTO 4640, 4660, 4680, 4700, 4720, 4740, 4760, 4780, 4800, 4820
4640 RUN "A:MENU1"              ' STOK
4660 RUN "A:MENU2"
4680 RUN "A:MENU3"              ' GELIRLER
4700 RUN "A:MENU4"              ' TAKAS
4720 RUN "A:MENU5"              ' MAL CIKISI
4740 RUN "A:MENU6"              ' KREDI SENETLERI.
4760 RUN "A:IMMU"               ' DIGER MUSTERILER
4780 RUN "A:IMTO"               ' DIGER TOPTANCILAR
4800 RUN "A:HLP1"
4820 PRINT BI$
4840 CK=5:  X=30:Y=10:GOSUB 5700
4860 PRINT CHR$(27)+"G1"+"ISLEMIN SONU"
4880 PRINT CHR$(27)+"G0"
4900 STOP
4920 END
```

```
4940 '*****************************
4950 PRINT CHR$(27);"*"
4960 PRINT BI$
4980 CK=5:X=2:Y=1:GOSUB 5700:PRINT "E.G.S GENEL MUHASEBE"
5000 CK=5:X=15:Y=6:GOSUB 5700
5020 PRINT "1-STOK"
5040 Y=Y+1:GOSUB 5700
5060 PRINT "2-MUSTERI VE TOPTANCI LISTELERI"
5080 Y=Y+1:GOSUB 5700
5100 PRINT "3-GELIRLER VE ODEMELER"
5120 Y=Y+1:GOSUB 5700
5140 PRINT "4-TAKAS"
5160 Y=Y+1:GOSUB 5700
5180 PRINT "5-MAL GIRISI"
5200 Y=Y+1:GOSUB 5700
5220 PRINT "6-KREDI-SENETLERI"
5240 Y=Y+1:GOSUB 5700
5260 PRINT "7-DIGER MUSTERILER"
5280 Y=Y+1:GOSUB 5700
5300 PRINT "8-DIGER TOPTANCILAR"
5320 Y=Y+1:GOSUB 5700
5340 PRINT "9-PROGRAMIN KULLANIMINA ILISKIN ACIKLAMALAR"
5360 Y=Y+1:GOSUB 5700
5380 PRINT "10-ISIN SONU"
5400 X=1:Y=Y+2:GOSUB 5700
5420 PRINT "ISTENEN SECENEGI GIRINIZ"
5440 X=26:GOSUB 5700:A$=INPUT$(1):PRINT A$
5460 V=VAL(A$)
5480 IF V>9 GOTO 5600
5500 PRINT BI$
5510 PRINT CHR$(27);"*"
5520 X=5:Y=6:GOSUB 5700
5540 PRINT "PROGRAMIN YUKLENMESINI BEKLEYINIZ"
5560 IF V=0 THEN V=10
5580 RETURN
5600 PRINT CHR$(7),CHR$(7)
5620 X=42:GOSUB 5700
5640 PRINT "GECERSIZ"
5660 GOTO 5440
5680 '
5700 'IMLECIN YERLESTIRILMESI
5720 ' KUTUK IMLE-X/Y
5740 ON CK GOTO 5760, 5800, 5920, 5960
5760 PRINT CHR$(27)+CHR$(61)+CHR$(31+Y)+CHR$(31+X);
5780 RETURN
5800 PRINT CHR$(27)+"c"+"1"
5820 PRINT CHR$(27)+CHR$(61)+CHR$(31+Y)+CHR$(31+X);
5840 PRINT CHR$(27)+"c"+"3"
5860 RETURN
5880 PRINT CHR$(27)+CHR$(61)+CHR$(L+31)+CHR$(71)
5940 RETURN
5960 PRINT CHR$(27)+CHR$(61)+CHR$(Y+31)+CHR$(X+31)
5980 RETURN
6000 '******
6020 ' ******
6040 'FONKSIYON TUSLARI
6060 '
6080 ' KUTUK=TUSLAR
6100 PRINT CHR$(27)+"c4AA"   'IMLECIN SEKLI
6120 A$=CHR$(27)+CHR$(46)
6140 B=98:I1=49:I2=48
6160 FOR I=1 TO 8
6180 PRINT A$+CHR$(B)+CHR$(I1)+CHR$(I2);
6200 K=(I1-48)*16+(I2-48):N=B-97
6220 B=B+1:I2=I2+1
6240 NEXT I
6260 B=106:I1=48:I2=52
6280 FOR I=1 TO 4
6300 PRINT A$+CHR$(B)+CHR$(I1)+CHR$(I2);
6320 K=(I1-48)*16+(I2-48):N=B-97
6340 B=B+1:I2=I2+1
6360 NEXT I
6380 RETURN
```

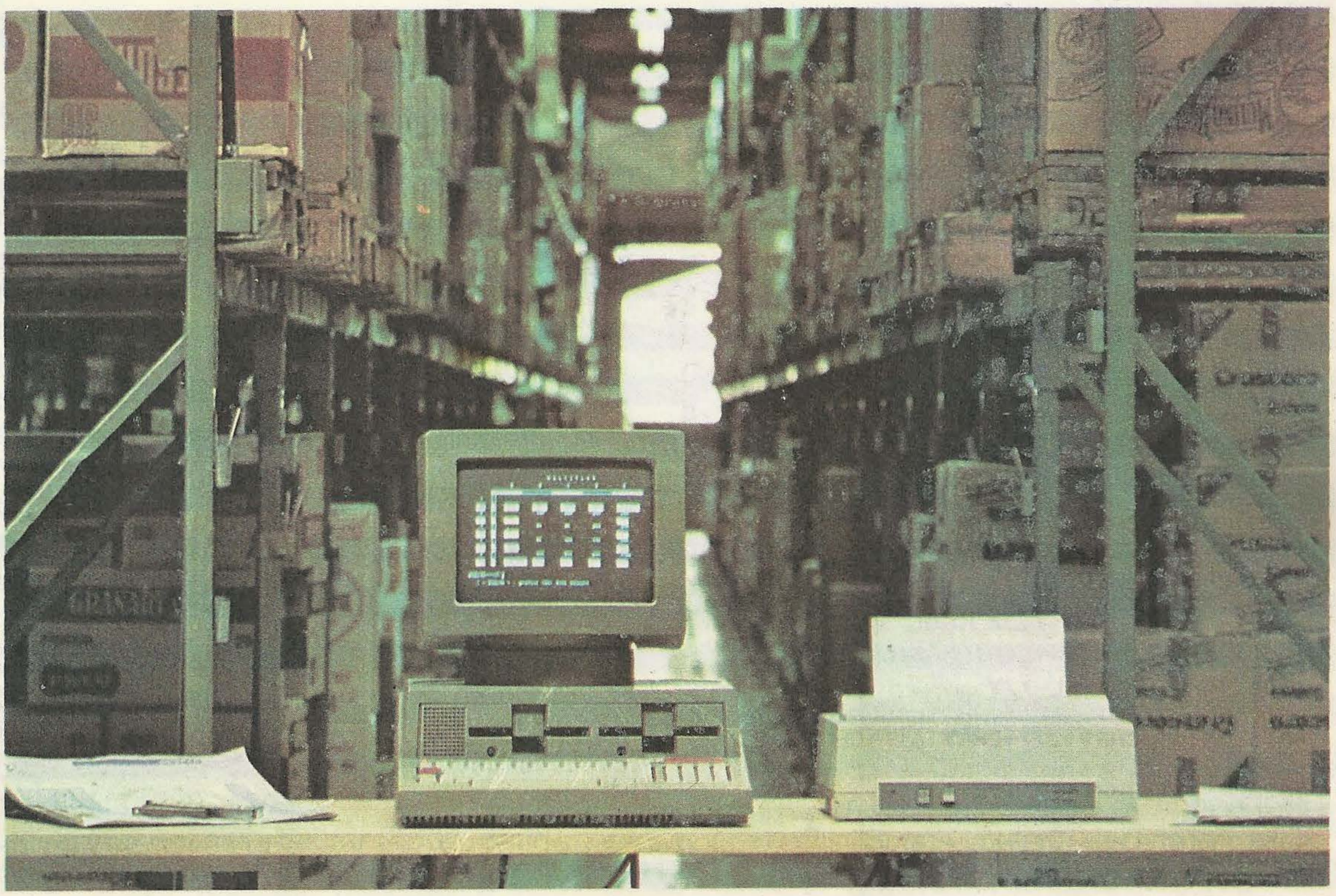
Gösterilen programın uyarlanabilmesini kolaylaştırmak için, sözkonusu makineye özgü özel fonksiyonlar, alt-programlarda tek tek tanımlanmışlardır. Sayfa 625'teki döküme (ana menü) dönersek, süreç içinde kullanılan veri kütüklerini tanımlayan bir başlangıç bölümüyle karşılaşırız (4545.-4555. satırlar). Burada kütüklerin isimleri ve kütükleri içeren disketlerin yerleştirilmesi gereken disket birimleri gösterilmektedir (DB sütunu). Ayrıca içerik tanımları, veri kaydının uzunluğu (Bayt olarak) ve veri kaydının bölündüğü alanların sayısı da aynı yerde bulunmaktadır. Bu tanımlayıcı başlangıç bölümünün amacı, programın belgelenmesidir. Bu yüzden, program üzerinde yapılan değişiklikleri kullanıcı ya da programcıya hatırlatılması bakımından çok yararlıdır. 4560. satır, fonksiyon tuşlarını sıfırlayan ve imlecin şeklini saptayan 6040 alt-programını çağırır. BI$ karakter zinciri (4580. satır) escape [CHR$ (27)] karakterinin yanısıra + ve CHR$(7) karakterlerini içermektedir. Birincisi, bu durumda kullanılan ekran birimi için, tüm ekranın içeriğini silme fonksiyonuna sahiptir. İkincisi, kullanıcının dikkatli olması için bir ses üretir. Bu ses ''bip'' şeklindedir ve bazı aygıtlarda, belli bir komutla yönlendirilir (BEEP). Görüldüğü gibi bu sesin üretilmesini sağlayan CHR$(7) kodu, ASCII tablolarında içerilmektedir ve bu nedenle hemen hemen her aygıtta mevcuttur.

Kullanılmak üzere hazır duran birçok ifadenin gösterimi, ekranın silinmesi (PRINT BI$) ve başlığın bulunduğu 4980. satırın gösterimi ile başlayan 4960 alt-programı sayesinde gerçekleşmektedir. Ekranda yazılan farklı satırların konumları parametrelendirilmiştir.

4980 satırı daha dikkatli incelendiğinde, burada üç parametre (CK, X ve Y) bulunduğu görülür. CK parametresi kullanılmamakta ve ilerdeki uygulamalar için saklanmaktadır. X değeri, yazımın başlayacağı sütunu, Y değeri ise satırı gösterir. Yazının X, Y koordinatlarında (sütun, satır) başlatılması için, imleç yazının ekran çıkışının alınmasından önce bu pozisyona getirilmelidir. Bu yerleştirme 5700 alt-programında gerçekleştirilir. Yazının çıkışı, imlecin kaydırılması için yararlanılan alt-programın çağrılmasını takip eden PRINT komutu ile elde edilir.

İmleç, ekrana güdüm karakter zincirinin (5760. satır) yazılması ve satır başına dönülmesini önleyen ; simgesinin en sona konması ile kolayca kaydırılabilir.

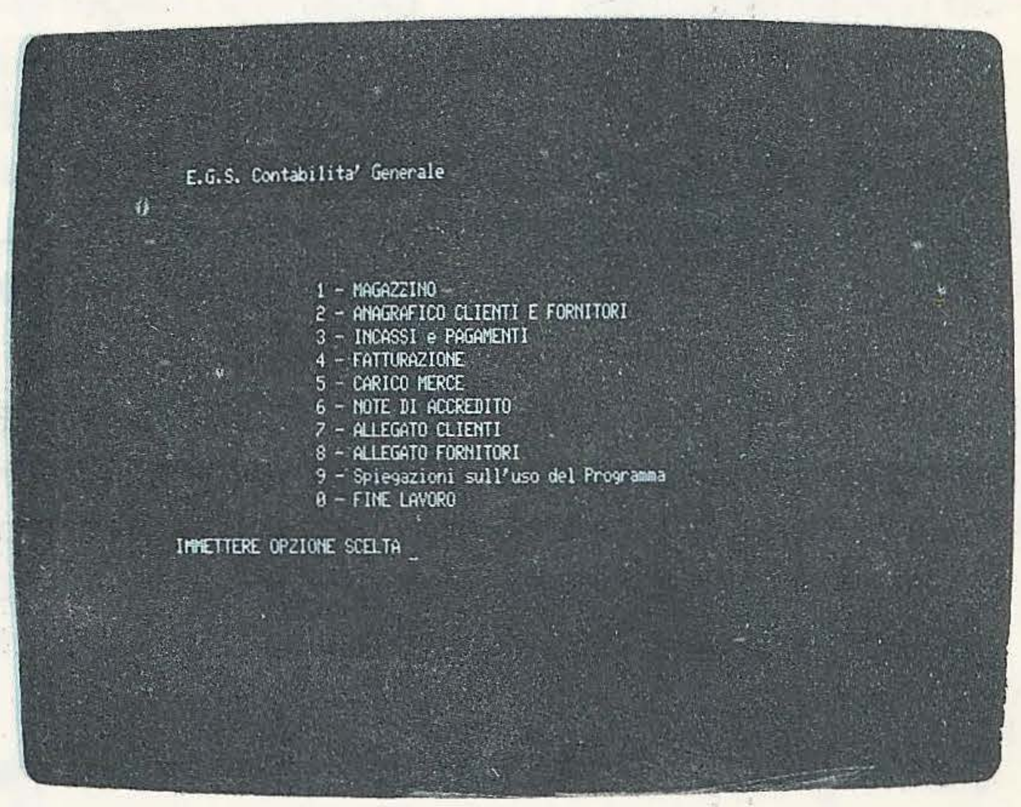
Kişisel bilgisayar, stok işlemlerini oldukça basitleştirir.



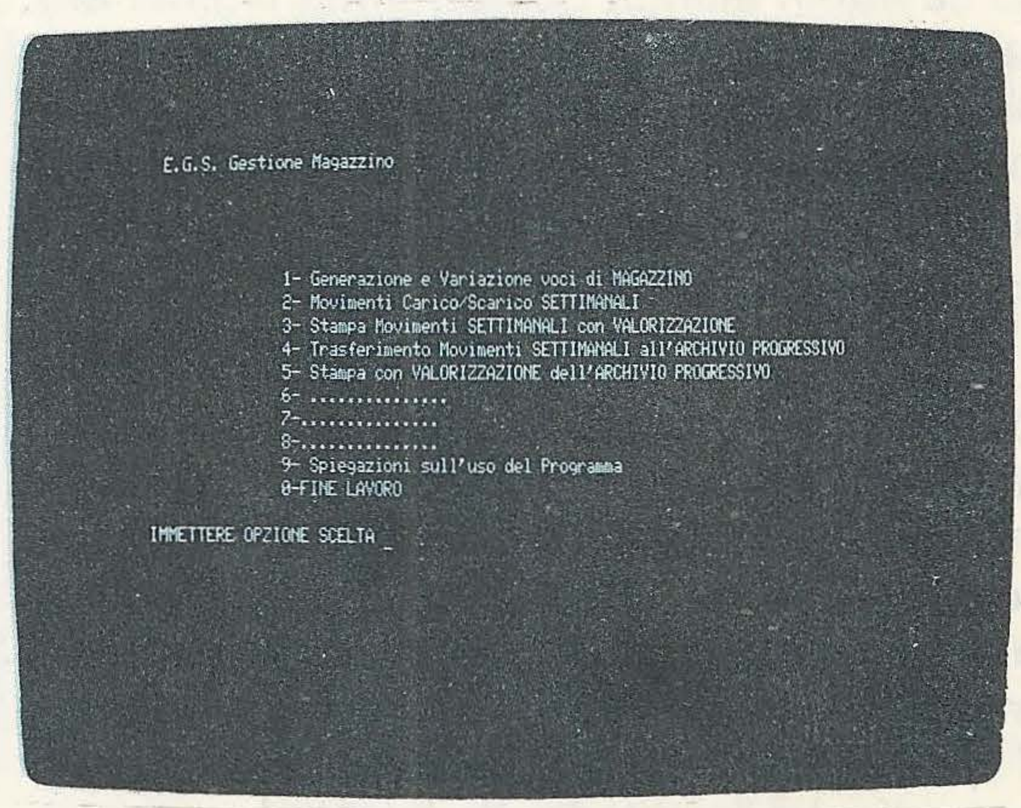
STOK HESABI MENÜSÜ

```
4500 '
4505 ' *** STOK HESABI MENUSU***
4510 '     KUTUK = MENU1
4512 '
4514 '    TARIH 2/12/83
4515 ' FONKSIYON TUSLARI ANA MENU TARAFINDAN BELIRLENIR--KUTUK:MENU
4520 BI$=CHR$(1'2)+"+"+CHR$(7)
4525 GOSUB 4615
4530 ON V GOTO 4535,4540,4545,4550,4555,4560,4565,4570,4575,4580
4535 RUN "A:ADOL"    'ADLARIN OLUSTURULMASI
4540 RUN "A:HADE"    'HAFTALIK DEVIR
4545 RUN "A:HDEB"    'HAFTALIK DEVRIN BILGIYAZIMI
4550 RUN "A:ARAK"    'ARSIVE AKTARMA
4555 RUN "A:ARBI"    'ARSIVIN BILGIYAZIMI
4560 GOTO 4510
4565 GOTO 4510
4570 GOTO 4510
4575 RUN "A:HLF2"
4580 PRINT BI$
4585 CK=5:X=30:Y=10:GOSUB 4800
4590 PRINT CHR$(27)+"O"
4595 PRINT CHR$(27)+"1"
4600 RUN "A:MENU"
4605 END
4610 '************************
4615 PRINT BI$
4620 CK=5:X=2:Y=1:GOSUB 4800:PRINT "E.G.S.STOK HESABI"
4625 CK=5:X=15:Y=6:GOSUB 4800
4630 PRINT "1-STOK ADLANDIRMALARININ YAPILMASI VE DEGISTIRILMESI"
4635 Y=Y+1:GOSUB 4800
4640 PRINT "2-HAFTALIK GIRIS/CIKIS AKISI"
4645 Y=Y+1:GOSUB 4800
4650 PRINT "3-HAFTALIK AKISIN DEGISIKLIKLERIYLE BILGIYAZIMI"
4655 Y=Y+1:GOSUB 4800
4660 PRINT "4-HAFTALIK AKISIN SUREKLI ARSIVE AKTARILMASI"
4665 Y=Y+1:GOSUB 4800
4670 PRINT "5-SUREKLI ARSIVIN DEGISIKLIKLERIYLE BILGIYAZIMI"
4675 Y=Y+1:GOSUB 4800
4680 PRINT "6-....................."
4685 Y=Y+1:GOSUB 4800
4690 PRINT "7-....................."
4695 Y=Y+1:GOSUB 4800
4700 PRINT "8-....................."
4705 Y=Y+1:GOSUB 4800
4710 PRINT "9-PROGRAMIN KULLANILMASINA ILISKIN ACIKLAMALAR"
4715 Y=Y+1:GOSUB 4800
4720 PRINT "0-ISIN SONU"
4725 X=1:Y=Y+2:GOSUB 4800
4730 PRINT "ISTENEN SECENEGI GIRINIZ"
4735 X=26:GOSUB 4800:A$=INPUT$(1):PRINT A$
4740 V=VAL(A$)
4745 IF V>9 GOTO 4775
4750 PRINT BI$
4755 X=5:Y=6:GOSUB 4800
4760 PRINT "PROGRAMIN YUKLENMESINI BEKLEYINIZ"
4765 IF V=0 THEN V=10
4770 RETURN
4775 PRINT CHR$(7),CHR$(7)
4780 X=42:GOSUB 4800
4785 PRINT "GECERSIZ"
4790 GOTO 4735
4795 '
4800 ' IMLECIN KAYDIRILMASI
4805 ' KUTUK IMLE-X/Y
4810 ON CK GOTO 4815,4825,4855,4865
4815 PRINT CHR$(27)+CHR$(61)+CHR$(31+Y)+CHR$(31+X);
4820 RETURN
4825 PRINT CHR$(27)+"c"+"1"
4830 PRINT CHR$(27)+CHR$(61)+CHR$(31+Y)+CHR$(31+X);
4835 PRINT CHR$(27)+"c"+"3"
```

```
4840 RETURN
4845 PRINT CHR$(27)+CHR$(61)+CHR$(31+Y)+CHR$(31+X);
4850 RETURN
4855 PRINT CHR$(27)+CHR$(61)+CHR$(L+31)+CHR$(71);
4860 RETURN
4865 PRINT CHR$(27)+CHR$(61)+CHR$(Y+31)+CHR$(X+31);
4870 RETURN
4875 '**********
4880 '***********
```



Ana menü: Seçilen numaraya göre ikincil menülerden biri yüklenir ve ekranda gösterilir.



1 numara (STOK) seçilmiştir. Bundan sonraki tercih karşılık gelen işleme erişilmesini sağlayacaktır.

5440. satırda, tek bir karakterin okunmasını sağlayan A$ = INPUT$(1) komutu bulunmaktadır (yalnızca 0 ile 9 arasında bir tercih yapılabilir). Bu şekilde, ele alınan karakterin sayısal değeri elde edilir (5460. satır) ve sınanır (5480. satır). Değer, öngörülen değerler arasında mevcutsa, ekran silinir (5500. satır) ve kullanıcı, beklemesi gerektiğini anlar (5540. satır), bundan sonra program, bu alt-programın çağrılmasını takip eden ana programın komutlarının yürütülmesine geri döner (4620. satır).

Bu aşamada kullanıcı tarafından yapılan tercihe göre, yüklenmesi gereken program aranır.

Örneğin, yanıt 1 olduğunda (V = 1) MENU 1 programı yüklenir ve işletilir. Dökümü 628. ve 629. sayfalarda görülen bu program, daha başka tercihler yapılmasını da mümkün kılmaktadır (4530.-4555. satırlar) ve sonuna geldiğinde bir öncekini çağırır (4600. satırı). Yani ikincil bir menü sözkonusudur.

Bu ikincil menüde, imlecin belli bir şeklinin elde edilmesine ve fonksiyon tuşlarının sıfırlanmasına (bu işlem daha önce ana menüde yerine getirilmiştir) yönelik komutlar bulunmamaktadır, ancak daima imlecin kaydırılması için alt-program (4800.-4840. satırlar) mevcut olmak zorundadır.

Gerek ana menünün, gerekse ikincil menünün aynı numaralamaya sahip olduklarına dikkat edilmelidir, çünkü biri diğerine aktarma yapmaktadır ve farklı numaraların kullanılmasını gerektiren hiçbir neden yoktur. Aksine bütün menüler için aynı satır sayısından yola çıkıldığında, hata aranması ve programda değişiklikler yapılması daha kolaydır.

Histogramların Üretimi

İmlecin adresleme fonksiyonu kullanıldığında, histogramlar çizisel olmayan ekranlarda da gösterilebilir. Bunun yöntemi, ekrana önceden saptanan belli bir gösterge katsayısına göre,

gösterilecek büyüklüğe karşılık gelen sayıda simgenin yazılmasından ibarettir. Örneğin, 150 değerinin çizisel olarak gösterilebilmesi için, gösterge katsayısı 10 olarak saptanabilir ve 150 sayısı 15 simgelik (150/10) bir sütun ile gösterilir. Ekranda gösterimin elde edilebilmesi için aynı simgenin 15 defa yazılması gerekir. Bu arada, her defasında satır sayısının bir birim artırılması ve sütun sayısının sabit tutulması gerekmektedir. Böylece, yüksekliği, çizisel olarak gösterilmek istenen sayıyla orantılı uzunlukta bir sütun elde edilir.

Bu sayfanın altında gösterim mantığının ana şeması gösterilmiştir. İmlecin konumlandırılması adresleme kodlarını içermesi gereken bir fonksiyonun tanımlanması ile sağlanır. Örneğin,

```
FNP$(A,B) = CHR$(27) + CHR$(61) +
            CHR$(31 + B) + CHR$(31 + A)
```

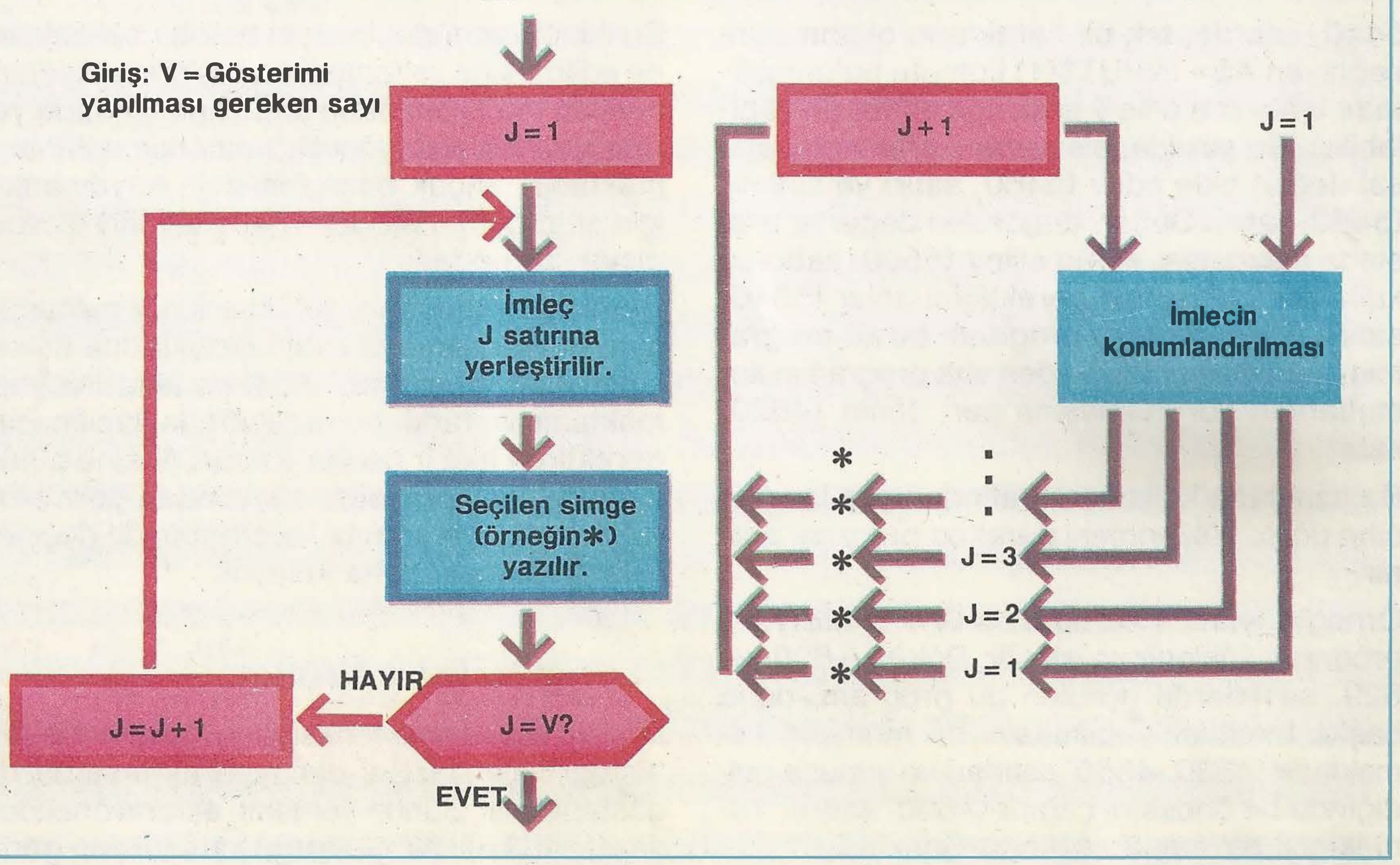
fonksiyonu, imleci B satırına ve A sütununa yerleştirir. Ekranda normal olarak 80 sütun ve 24 satır bulunduğundan A'nın alabileceği en büyük değer 80, B'ninki ise 24'tür. Bununla birlikte I olarak tanımlanan X-ekseni üzerinde en çok bir karakterlik ve J olarak tanımlanan Y ekseni üzerinde en çok bir satırlık kaydırmalar yapılabilmektedir. Bir başka deyişle simgelerin, iki satır (y-ekseni) veya iki karakter arasındaki (x-ekseni) mesafenin izin verdiğinden daha yakın durması halinde bir diyagram elde edilemez, çünkü **çizisel olmayan bir ekran için adresleme yalnızca tam satır ve sütun (karakter) sayıları ile mümkündür; ara konumlar yoktur.**

Sayfa 631'deki şemaya bağlı olarak, histogramın 150 değerini gösteren sütunu elde edilir. Bunu yaparken, imleç birbirini takip eden 15 satıra (1, 2, 3, vb) adreslenir ve istenilen simge yazılır. Sütunun değeri (I) ise değişmeksizin kalır. Öncekinin yanında yeni bir değerin gösterilebilmesi için X ekseninin değerinin (sütun, I) değiştirilmesi ve satırlara (J) ilişkin döngünün tekrarlanması gerekir. Her iki döngü biraraya getirildiğinde, yani biri diğerinin içine sokulduğunda, her biri ekranın bir sütununu gösteren en fazla 80 sütun elde edilebilir.

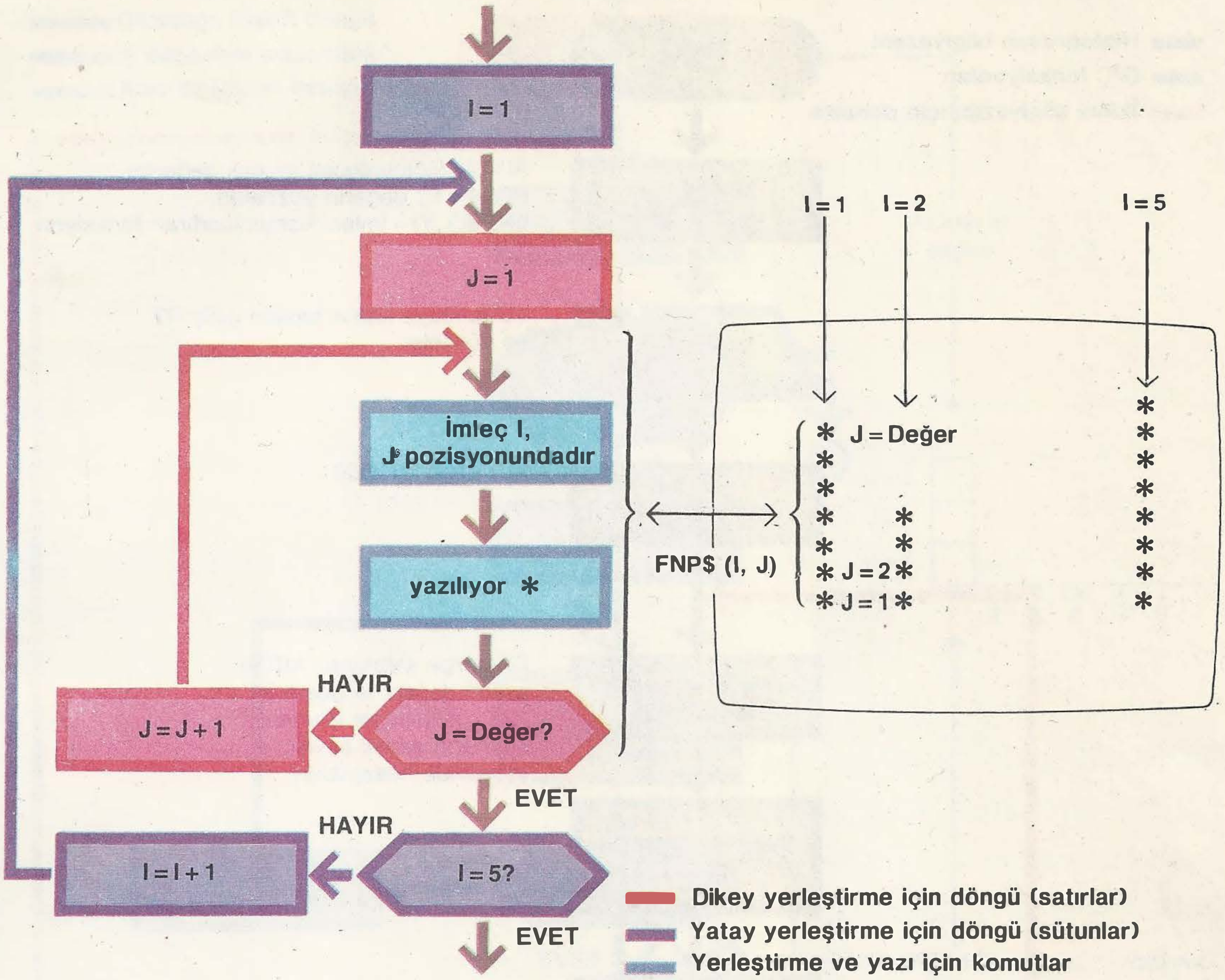
Şemada, satır ve sütun değerlerinin tekil olarak doğrudan ekranda adreslenmesinin, koordinatlarla adresleme olarak adlandırılan genel yöntemden ayırdedilebilmesi için X ve Y koordinatları, I ve J simgeleri ile adlandırılmıştır.

Çizisel olmayan ekranlarda, koordinatlar yalnızca tamsayılarla ifade edilebilir, çünkü böylece fiilen ekranda mevcut olan satırlar ve sü-

TAMSAYILI BİR SAYININ ÇİZİSEL GÖSTERİMİNİN MANTIĞI



BİRKAÇ SÜTUNA YAYILAN ÇİZİSEL GÖSTERİMİN MANTIĞI

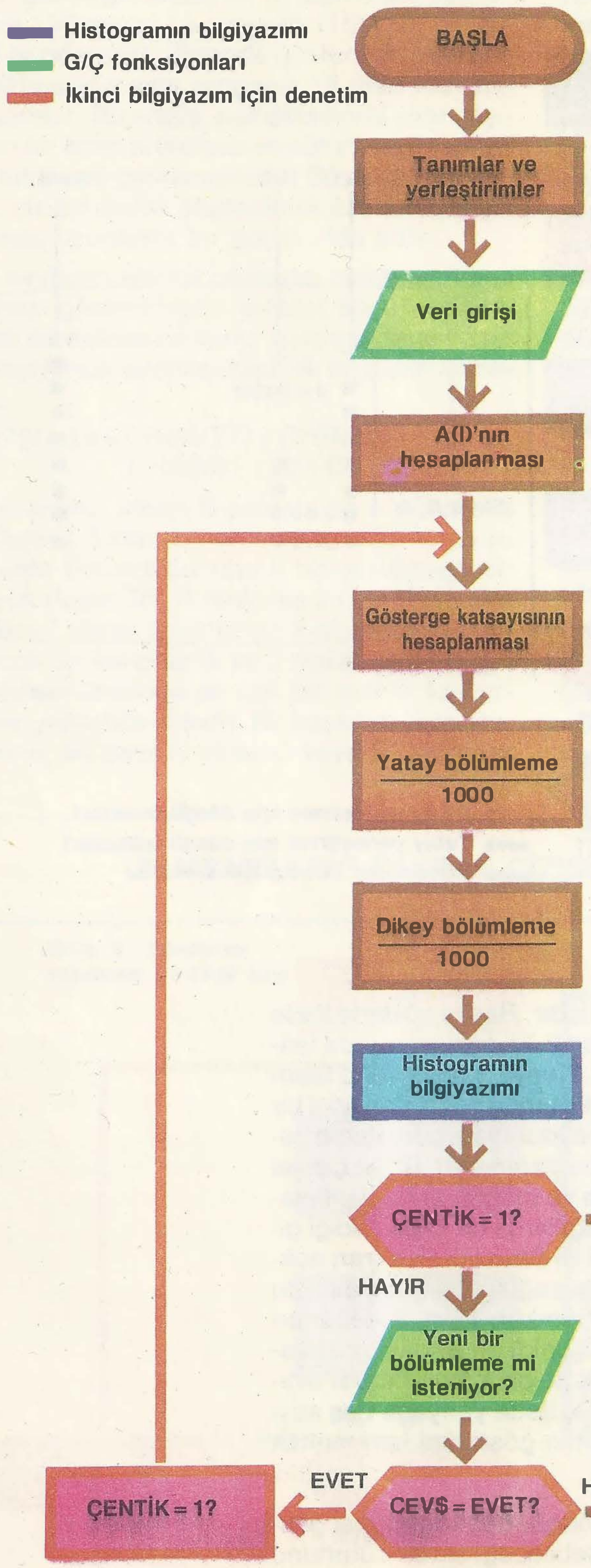


tunlar tanımlanmaktadır. Başka sözlerle ifade edilecek olursa imleç 2.75 pozisyonunda (satırında) bulunamaz. 2 veya 3 pozisyonu (satırı) seçilmelidir. Bu durumda doğrudan tekil bir adresleme sözkonusudur (yani adresleme belirli sayıda değere bağlanmıştır). Oysa çizisel ekranlarda satırlara ve sütunlara bağlanılmamakta, milimetrik kağıtlarda da yapılabildiği gibi, bir koordinat çifti ile ifade edilen **ekran noktalarına** yerleştirme sağlanabilmektedir. Bu durumda da tüm mümkün X ve Y değerleri gösterilemeyebilir, çünkü ekrandaki noktaların sayısı çok büyük de olsa, belli bir üst sınırı vardır. Bu sayfada, üstte yanyana beş sayı içeren bir histogramın gösterimi için mantık şeması verilmiştir.

En dıştaki birinci döngü, değerin çizisel gösteriminin gerçekleşebileceği ekran sütununu belirlemektedir.

HİSTOGRAMLARIN BİLGİYAZIMI İÇİN PROGRAM

Histogramın bilgiyazımı
G/Ç fonksiyonları
İkinci bilgiyazım için denetim



A(12) = Gösterilecek mutlak değerler
P(12) = 12 değerin yüzdeleri
FNP$(X,Y) = İmleci konumlandıran fonksiyon

12 noktaya ilişkin toplam giriş (T) ve yüzdeler

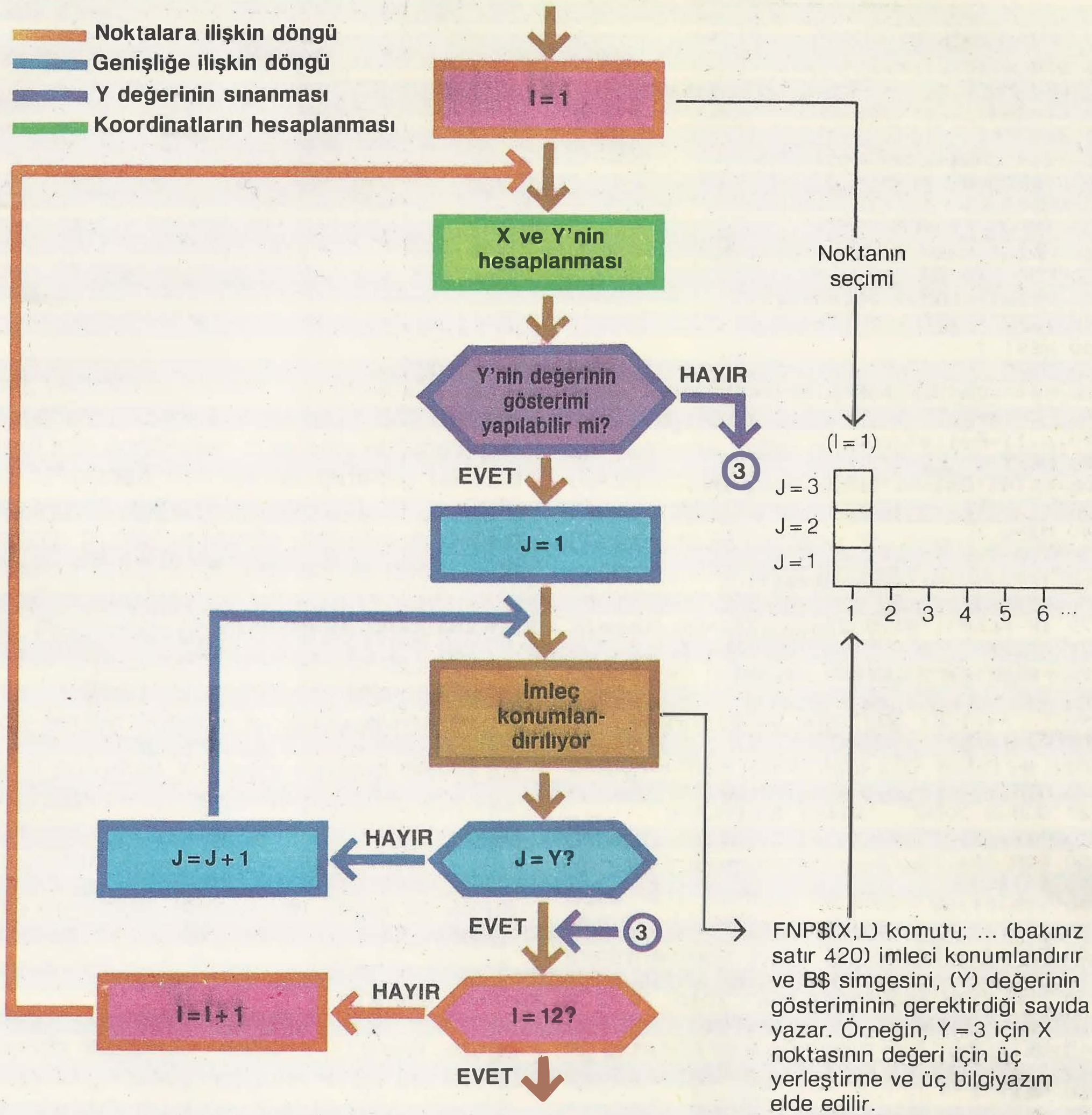
A(I) = P(I) * T/1000

Gösterge katsayısı, A(I)'in azami değeri ve gösterim için kullanılması istenen satırların azami sayısı temelinde hesaplanır.

Geliştirme akış şemasına bakınız.

Histogramın bilgiyazımı tamamlandığında ÇENTİK'in değeri 1 olur.

"HİSTOGRAMIN BİLGİYAZIMI" KUTUSUNUN AÇIMLANMASI



En içteki döngü, beş sütunun her birinin yüksekliğini saptamaktadır. Bunun yanısıra gösterilecek değerle orantılı olan satır sayısını ve her satır için seçilen simgeyi belirlemektedir (örneğimizde * simgesi). 632. sayfada ve bu sayfada üstte, 12 sayılı histogramın bilgiyazımını sağlayan bir programın akış şeması gösterilmiştir. Örnek, genelleştirilerek sunulmuştur ve pratik durumlarda, rastlansal büyüklükte 12 değerin [örneğin, öngörülen masraflar (bütçeler), çalışma saatleri vb] dağılımının çizisel gösterimi için kullanılabilir. Noktaların sayısı (12) bir yıllık akışın aylık gösterimini ortaya koymayı mümkün kılmaktadır. 634. ve 635. sayfada üstte gösterilen prog-

HİSTOGRAMLARIN BİLGİYAZIMINA ÖRNEK

```
30 'KUTUK:HISTO
40 DIM A(12), F(12)
50 DEF FNP$(X,Y)=CHR$(27)+CHR$(61)+CHR$(31+Y)+CHR$(31+X)
60 C$=CHR$(12)+CHR$(7)
70 B$= "*"
80 FI$= CHR$(27)+"O"
90 FV$=CHR$(27)+"1"
95 PRINT C$
100 INPUT "TOPLAM SAYI"; T
110 PRINT "HER ZAMAN DILIMI ICIN YUZDELERI GIRINIZ"
120 FOR I=1 TO 12
130 PRINT "ZAMAN DILIMI";I
140 INPUT "YUZDE";P(I)
150 NEXT I
160 SO=5    'YATAY GOSTERGE KATSAYISI 1 NOKTA=5 POZISYON
170 '** TEK TEK NOKTA MIKTARLARININ HESAPLANMASI
180 FOR I=1 TO 12
190 A(I)=P(I)*T/100
200 NEXT I
205 PRINT C$     'EKRAN SILINIR
220 ' **A(I) DAN AZAMI DEGERIN ARANMASI
240 MX=0
250 FOR I=1 TO 12
260 IF A(I)>MX THEN MX=A(I)
270 NEXT I
272 IF FLAG=1 GOTO 276    'IKINCI DONUSTE BOLUMLEME TERCIHI ATLANIR
275 GOSUB 3000  'DIKEY BOLUMLEME TURUNUN SECILMESI
276 PRINT C$    'EKRAN SILINIR
280 '
290 '   **DIKEY GOSTERGE KATSAYISI=SATIR SAYISI/MAKSIMUM
310 SV=16/MX
320 '
325 GOSUB 1000    'YATAY BOLUMLEME
327 GOSUB 2000    'DIKEY BOLUMLEME
330 '****HISTOGRAMIN BILGIYAZIMI ICIN DONGU***
350 FOR I=1 TO 12
360 X=I*SO
370 Y=A(I)*SV
375 IF Y<1 GOTO 460    'DEGER YENIDEN VERILEMIYORSA, BILGIYAZIM ATLANIR
380 FOR J=1 TO Y
390 L=18-J'*****Y=1 POZISYONU USTTE SOLDADIR
400 ' ALTTAN BASLAYABILMEK  ICIN, 18-J YE KAYMASI GEREKIR
410 PRINT FNP$(X,L);FI$;B$;FV$
415 NEXT J
420 X1=X-1:PRINT FNP$(X1,L);P(I)
460 NEXT I
470 X=40:Y=1:PRINT FNP$(X,Y);     ' IMLEC 40,1 E YERLESTIRILIR
472 IF CEV$="HAYIR" GOTO 505
475 PRINT "DEVAM ICIN BIR KARAKTER GIRINIZ":S$=INPUT$(1)
480 PRINT C$
485 IF FLAG=1 GOTO 510      'BUTUNLENDIGINDE IKINCI DONGU SONA ERER
490 INPUT "GORELI BOLUMLEMELI BILGIYAZIM DA ISTENIYOR MU?(EVET/HAYIR)";CEV$
500 IF CEV$="EVET" THEN FLAG=1:MX=T:GOTO 310
505 X=-31:Y=1:PRINT FNP$(X,Y)
510 END
1000 '**12 NOKTADAN OLUSAN YATAY BOLUMLEMENIN YAZILMASI
1005 NL=SO*12+1 'YATAY EKSENIN UZUNLUGU
1006 G$=STRING$(NL,"-")
1007 X=1:Y=20:PRINT FNP$(X,Y);G$
1008 Y=23
1010 FOR I=1 TO 12
1020 X=1*SO
1030 G$="   "
1040 PRINT FNP$(X,Y);G$;I;
1050 NEXT I
1055 PRINT          ' IMLEC POZISYONUNA DONER
1060 RETURN
2000 '  **** DIKEY BOLUMLEMENIN YAZILMASI
2005 X=1
```

```
2010 FOR I=1 TO 20          20 HISTOGRAMIN AZAMI YUKSEKLIGIDIR
2020 L=18-I
2030 PRINT FNP$(X,L);"I"
2040 NEXT I
2045 PRINT FNP$(X,1);"TOPLAM=";T
2050 RETURN
3000 ' **** DIKEY BOLUMLEME***
3010 '
3020 '
3030 INPUT "BOLUMLEME DEGERI OLARAK MAKSIMUM MU ISTENIYOR (EVET/HAYIR)";CEV$
3040 IF CEV$="EVET" THEN RETURN
3060 MX=T:GOTO 310
3070 IF CEV2$="EVET" THEN RETURN
3080 RETURN
```

rama, veri olarak toplam sayı (masrafların tutarı, çalışma saatleri vb) ve her nokta (zaman birimi) için öngörülen oranların verilmesi gerekir. Program her zaman birimine tekabül eden mutlak değeri hesaplamakta (170.-200. satırlar), bunlar arasında maksimum değeri aramakta (240.-270. satırlar) ve buna bağlı olarak dikey gösterge katsayısını saptamaktadır. Programda üç alt-program öngörülmüştür. Farklı bölümlemelerin en açıklayıcı örneği olan 300 alt-programı iki gösterim şekli arasında tercihi mümkün kılmaktadır.

Birinci durumda Y-ekseni (ordinat-dikey) tümüyle maksimum değerin gösterimi için kullanılır, yani genişletilmiş bir bölümleme sözkonusudur.

İkinci durumda, maksimum değer ile sütun yüksekliğinin yalnızca bir yüzde miktarını dolduran bir dikey bölümleme kullanılmaktadır.

Örneğin; 12 zaman diliminin bütün değerleri arasında maksimum değerin 10'a eşit olduğu varsayılırsa (toplam değerin % 20'si), birinci bölümlemede bu değer Y-ekseninin yüksekliğine sahip bir dikey sütunla ve diğer değerler ise buna göre hesaplanmış oranlarla gösterilir. Örneğin, mutlak 5 değeri yarım yükseklik ile gösterilir.

İkinci durumda (yüzdelik bölümleme) 10 değeri, kullanılacak sütunun 2/10'luk yüksekliğine, 5 mutlak değeri ise sütunun 1/10'luk yüksekliğine sahiptir.

Programda, sırayla her iki gösterim şeklinin görüntülenmesini sağlayan bir çentik kullanılmıştır. Önce yüzdelik bölümlemeli şekil görüntülenir. 470. ve 475. satırlar yanıt beklenilen bir ileti için tipik bir örnek oluşturmaktadır.

Bir çıkışın sonunda veri işlemeye devam edilebilmesi için, kullanıcının, ekranda gösterilen sonuçları okuması beklenmelidir. Genel uygulama, kullanıcı ile iletişim amacıyla iki veya üç satırın ayrılması şeklindedir. Bunun üzerine bir bekleme iletisi görünür (475. satır). Bu bekleme durumu, program akışını herhangi bir karakter girişine kadar bekleten S$ = INPUT$ (1) komutu ile sağlanır. Bu komut yankıyı engeller, yani karakter ekranda gözükmez ve S$ karakter zinciri yalnızca bekleme durumunu sağlamakla görevlidir.

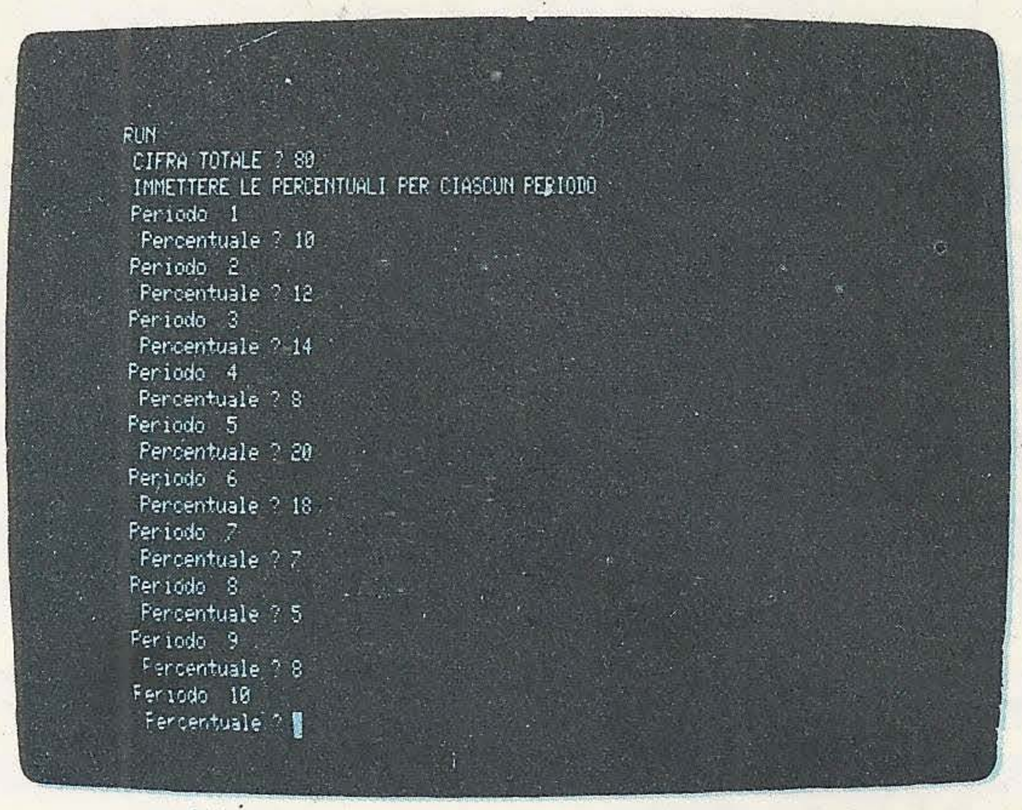
Gösterilen program, histogramlar için simge olarak ters bir boş yer kullanmaktadır (ışıklandırılmış bir boş dikdörtgen gözükür), simge B$ değişkeninde içerilmiştir (70. satır) ve değiştirilmesi için yerleştirimin değiştirilmesi yeterlidir.

İmlecin konumlandırılması ve simgenin yazılması, 410 komutu ile sağlanır. Bu komutta FNP$(X,L), imleci X sütununa, L satırına yerleştirmektedir. Satır, histogramın alt ucunun, ekranın alt bölümünde 18. satırdan başlaması için 18-Y olarak verilmektedir. FI$ değişkeni, ekranda ters gösterim için güdüm kodunu içermektedir (80. satır); B$, yazılacak olan simgedir ve FV$, ekranı yeniden normal yazıya döndüren kodları içermektedir. Belli bir değeri gösteren sütunun sonunda, gereken yüzde miktarı P(I) yazılmaktadır (420. satır). Bu değer imlecin yeniden, döngüden çıkışın L satırına, fakat bir sütun soluna (X1 = X—1) yerleştirilmesiyle elde edilir. Böylece sayının birinci rakamı, histogramın sütununa göre ortalanmış olarak elde edilir. Belirtilen kodlar özel bir bilgisayara ilişkindir, diğer modellerde, 50, 60, 80 ve 90. satırlara kullanım kılavuzunda tanımlara uygun değerler yerleştirilmelidir.

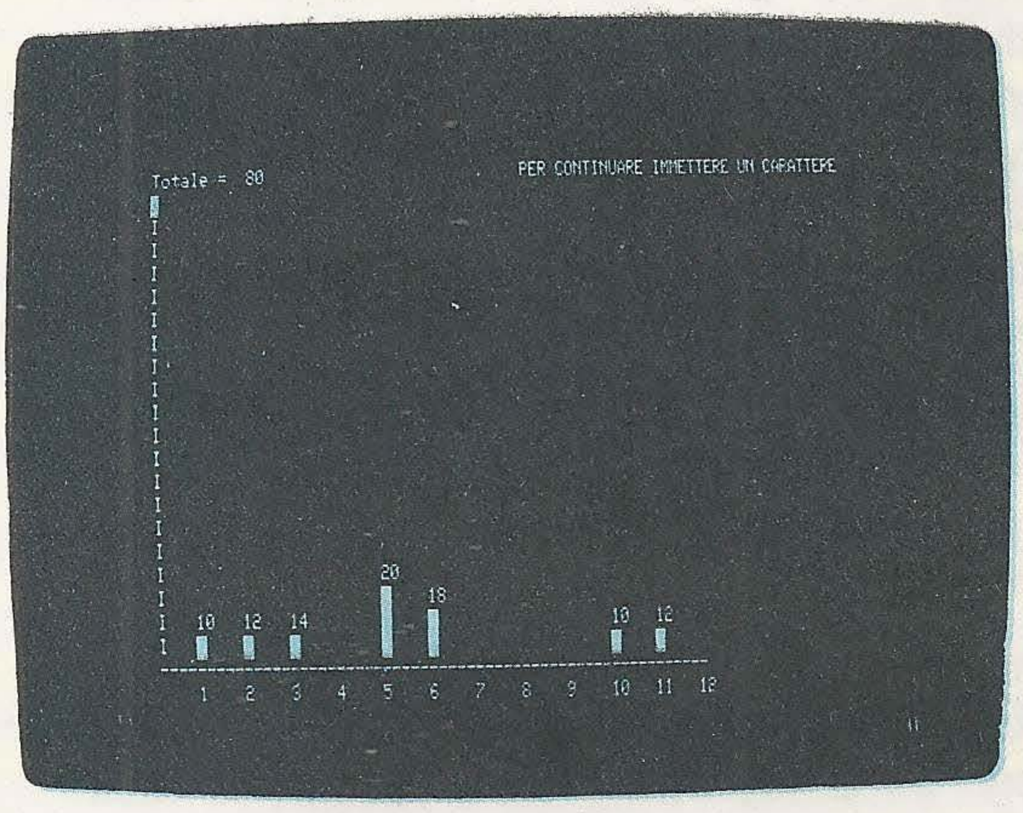
1000 ve 2000 alt-programları yatay ve dikey bölüm çizgilerinin çekilmesini sağlarlar. 1006 satırında yalnızca — simgesini içeren N uzunluğundaki G$ karakter zincirinin üretilmesi için STRING$(NL,''—'') komutu kullanılmış-

HİSTOGRAMLARIN EKRANDA OLUŞTURULMASI

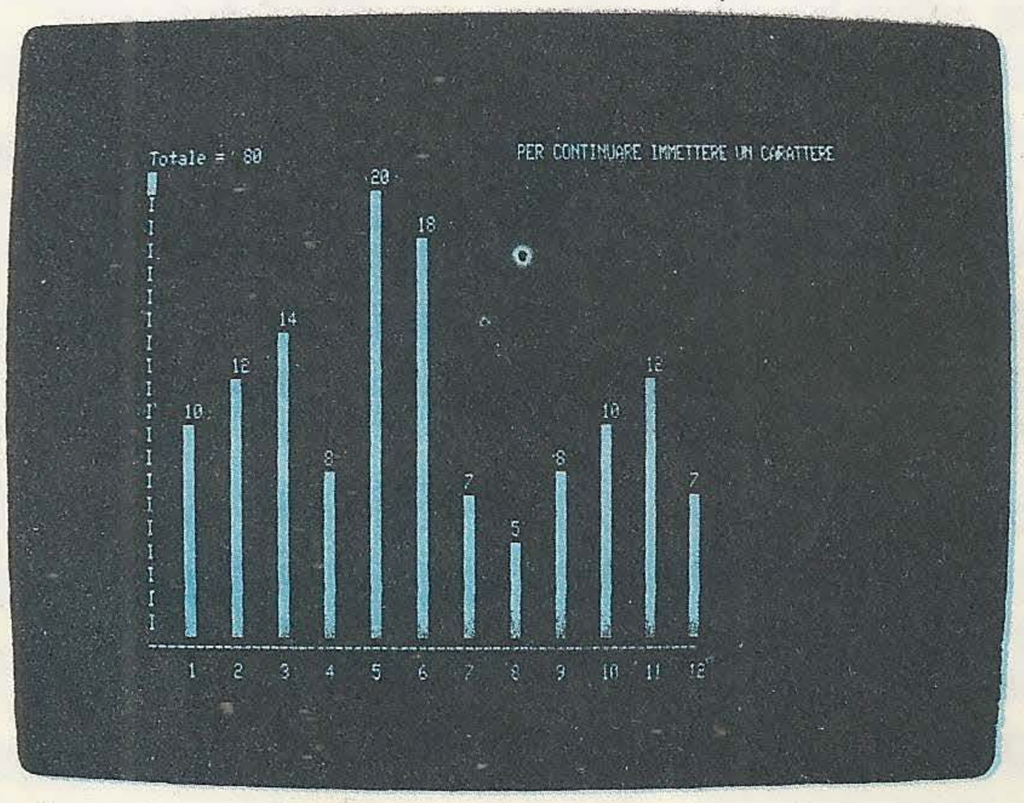
Bu sayfada, veri girişi aşamasında (bu hizada sağda) ve histogramların bilgiyazımı sırasında altta oluşan ekran görüntüleri verilmektedir. Burada 634. sayfadaki programdan yararlanılmaktadır. Veri girişi aşaması gerekli verileri sırayla talep eden program tarafından yönlendirilmektedir (100.-150. satırlar). Hesaplama aşamasından sonra denetim, kullanılacak dikey bölümlemeyi belirleyen 3000 alt-programına geçer.



Birinci durumda göreceli bir bölümleme seçilmiştir: Her sayıda veri girişi bir sütun olarak görüntülenir. Bu sütunun yüksekliği bizzat veri kaydı tarafından tanımlanan olası azami yüksekliğin yüzdesine eşittir.



İkinci durumda dolu bölümleme seçilmiştir: Bu kez verilen azami değer, yüksekliği olası azami yüksekliğe eşit olan bir sütunla gösterilmektedir. Diğer veriler bu sütunla orantılı olarak görüntülenir.



TEST 18

1 / Bir karakterin yüklemleri nelerdir?

2 / Aşağıdaki komutlardan hangileri yanlıştır?

```
10 PRINT A, B, C
20 FOR I=1 TO 20:PRINT ''DENEME'';:NEXT I
30 PRINT A, TAB(60),B,TAB(100),C
```

3 / Aşağıdaki komutların etkileri nelerdir?

```
10 FOR I=1 TO 30
20 PRINT I,''VERİLER'',A$
30 NEXT I
```

4 / FNP$(X,Y) fonksiyonunun, imleci X sütunu ve Y satırına getirdiği varsayılırsa, hangi komutlar ekranın istenilen bir yerine herhangi bir karakter zincirinin yazılmasını mümkün kılar? Koordinatlar (sütun/satır) ve karakter zinciri klavye aracılığıyla verilmelidir ve sütun=0 veya satır=0 verildiğinde program sona ermelidir.

5 / INV$ karakter zincirinin ekranda yazının üretimi için kodları içerdiği ve NOR$'ın normal fonksiyon tarzı için kodları içerdiği varsayılırsa, önceki soruda yazılış yöntemi, karakter zincirinin bilgiyazımının ters olarak elde edilmesi için nasıl değiştirilmelidir?

Testin çözümleri 644-645. sayfalarda bulunmaktadır.

tır. Bu karakter zinciri yazıldığında yatay çekilmiş bir doğru elde edilir. Yatay bölümde ayrıca 1010 ve 1050 komutları ile aralık değerleri (1'den 12'ye kadar) gösterilmiştir.

Kütük Yönetimi

BASIC'in tanıdığı iki kütük türü vardır: Sıralı ve doğrudan erişimli (random).

Sıralı kütüklerde, veriler birbiri ardına yazılır (ve okunur); diğer bir deyişle, öncesindeki tüm veri kayıtları sırayla gözden geçirilmeden, bir veri kaydına erişilemez. Bu kütüklerin tipik kullanımı klavye aracılığıyla girilen verilerin saklanmasıdır. Buna karşın, doğrudan erişilebilen kütükler (random files), doğrudan yalnızca sözkonusu veri kaydının adreslenmesiyle, verilerin okunmasını ve yazılmasını mümkün kılarlar. Bir kütükteki verilere erişmenin man-

tığı her iki durumda da aynıdır ve aşağıdaki fonksiyon sıralarını içerir:

- Kütüğün açılması
- Verilerin okunması ve/veya yazılması
- Kütüğün kapatılması (kullanımının sonunda)

Her fonksiyon, sıralı veya doğrudan erişimli kütüklerin sözkonusu olmasına bağlı olarak farklı bir sözdizimine sahiptir. Bu nedenle, yukarıda bahsedilen iki durumla bağlantılı olarak ayrı ayrı incelenecekleridir.

Sıralı Kütüklere Erişim İçin Fonksiyonlar

İki çeşit sıralı kütük vardır: ''O'' (Output) kısaltması ile gösterilen çıkış kütükleri veya ''I'' (Input) kısaltması ile gösterilen giriş kütükleri. Kütüğün türü (O/I) açılış komutunda kesin olarak saptanmış olmalıdır. Verilmesi gereken komut, verilere erişme denemesi yapılmadan önce,

OPEN ''TUR'', N, ''ISIM''

şeklindedir. Burada,

TUR G/Ç değerlerinden birini alabilir.
N Kütüğe verilmiş bir sayıdır. Kütük, takip eden verilere erişme işlemlerinde bu sayıyla tanımlanmalıdır.
ISIM Kütüğün üretildiği isimdir.

Örneğin;

OPEN ''I'', 2, ''A:DENEME''

satırı, okunması için [INPUT (I)], DENEME isimli, birinci disket biriminde (A:) bulunan bir kütüğü açar. Bazı sistemlerde kütük numarası # simgesinden sonra gelmelidir. Bu durumlarda açılış komutu aşağıdaki gibi olur:

OPEN ''I'',#2, ''A:DENEME''

Aşağıda ise OPEN komutunda # simgesi kullanılmayacaktır, fakat BASIC türüne göre kullanılması zorunlu olabilir. Aynı kütüğün Output şartı altındaki açılımı;

OPEN ''O'', 2, ''A:DENEME''

satırı ile elde edilir.

Kütüğün açılmasında gerekli olan şart, kullanılmak istenen ve kendinden sonra gelen fonksiyona bağlıdır.

Kütükten veriler alınmak isteniyorsa, kütük INPUT şartı ile açılmalıdırlar, yani aslında programa bir veri girişi sözkonusudur.

Buna karşılık kütüğe yazılmak isteniyorsa, Output şartı ile açılmalıdır, çünkü program açısından kütük, bir çıkış hazırlığına (mekanizmasına) tekabül eder.

Verilere erişmek için yapılan işlemlerin sonunda kütük kapatılmalıdır. Bunun için, CLOSE N komutu verilmelidir. N, kütüğün açılış aşamasında ayrıldığı kanalı göstermektedir. Örneğin, yukarıdaki kütüğün kapatılması için verilecek komut CLOSE 2 şeklinde olmalıdır.

OPEN ve CLOSE komutları arasında, verilere erişmek için veya kütüğün durumunun kontrolü için öngörülen tüm fonksiyonlar kullanılabilir. Aşağıdaki komutlar sıralı kütüklerde kullanılabilir:

INPUT, LINE INPUT, WRITE, PRINT, PRINT USING, EOF, LOC, KILL, NAME.

Aşağıda her komut açıklanıp yorumlanacaktır:

INPUT. INPUT # N, DEGISKEN söz dizinine sahip olan bu fonksiyon, N numaralı kütükteki verileri alır ve verilen değişkene aktarır. Örneğin;

INPUT # 1, B$

satırı verildiğinde, kütük 1'den bir veri kaydının içeriği (güncel veri kaydı = sıralı kütüklerde okuma ve yazmanın daima 1. veri kaydından başlayıp seri halinde devam ettiği düşünülmelidir) okunur ve B$'a aktarılır. Bu komutta verilere erişmek için yararlanılan diğer komutlardaki gibi, # simgesi unutulmamalıdır.

LINE INPUT. Bu komutun noksansız sözdizimi,

LINE INPUT # N, DEGISKEN

şeklindedir. Burada, klavyede kullanılan LINE INPUT komutunda olduğu gibi, karakterlerden (254'e kadar) oluşan tüm bir satır alınır.

Bu komutun özel bir kullanımı, diskette ASCII yazı düzeninde saklanan bir programın, başka bir program aracılığıyla okunmasında ortaya çıkar. İkinci program, birincisini diskette bulunanı) normal bir kütük (veri kütüğü) gibi görür ve bu nedenle okuyabilir. Tek sakıncası (mantıksal) veri kayıtlarının uzunluklarının değişken olmasıdır.

ASCII'de saklanan bir programın her satırı, içinde yazılmış olan komuta bağlı olan sayıda karakter içerir. Satırın nerede bittiğini öğrenmenin tek yolu veri kaydının BASIC satırının sonunda bulunan RETURN koduna raslanıncaya kadar sıralı olarak okunmasıdır.

WRITE.

WRITE # N, DEGISKEN

sözdizimine sahip olan bu fonksiyon, N sayısı ile adlandırılan kütüğe, verilen değişkende içerilen değeri yazar. Bu komutun kullanılabilme-

si için kütüğün ''O'' şartı altında açılması gerekir (programın bir çıktısı sözkonusudur).

WRITE komutu, aktarılacak birden fazla değişken sözkonusu olduğunda otomatik olarak değerler arasına bir virgül yerleştirir.

PRINT ve PRINT USING. Bu komutun sözdizimi aşağıdaki şekildedir:

```
PRINT # N, DEGISKEN
PRINT # N, USING...
```

Bunlar daha önce bilgiyazım fonksiyonları için öğrendiklerimizle aynı anlamlara sahiptirler. Bu komutlarla N kütüğü, herhangi bir çevresel gereçmiş gibi ele alınır.

EOF(N). Bu, kütüğün okunması sırasında kütüğün sonunu kendiliğinden gösteren belli bir koda rastlanıldığında, —1 (gerçel) değerini tekrarlayan (sırasal) N kütüğü ile bağıntılı bir fonksiyondur. EOF kodu End of File (kütüğün sonu) sözcüğünün başlangıç harflerinden oluşmaktadır.

Herhangi bir sıralı kütüğün okuma döngüsünde kullanılacak olan bu fonksiyon, önceden kütüğün uzunluğu bilinmeksizin, tüm yazılı değerlerin kabulünü sağlayan bir döngünün işletilmesine yarar. Örneğin;

```
10 OPEN ''I'', 1, ''A:DENEME''
20 INPUT # 1, A$, B$, C$
30 IF EOF(1) GOTO 100
40 GOTO 20
100 PRINT ''Kütüğün okunmasının sonu''
110 END
```

programı, 20, 30 ve 40 komutlarından oluşan bir döngüyü, End of File koduna rastlanıncaya kadar işletmektedir. Bu durumda EOF(1) fonksiyonu geçerlidir [—1 = gerçel(*) değerini vermektedir] ve program 100 komutuna atlamaktadır. 30. satır olmasaydı hiçbir denetim kalmazdı ve kütükte içerilenden daha fazla sayıda veri kaydının okunmasına çalışılabilirdi; sonuçta bu bir hataya yol açardı ve program dururdu.

LOC(N). N kütüğünde okunan veya yazılan kesimlerin sayısı, açılışlarından itibaren verilir (bir kesiminin 128 veya 256 bayttan oluştuğunu hatırlayınız).

KILL. KILL ''ISIM'' satırı ISIM kütüğünü siler. Silinecek kütük tedbir olarak kapatılmalıdır.

KILL komutu her türlü kütük (yani sıralı, doğrudan erişimli veya program kütüğü için) kullanılabilir.

NAME. Bu fonksiyon bir kütüğün isminin değiştirilmesi için kullanılır.

```
NAME ''ISIM 1'' AS ''ISIM 2''
```

genel satırı eski isim olan ISIM 1 yerine yeni isim olan ISIM 2'yi yerleştirir. Örneğin bu yöntemle,

```
NAME ''ESKIKU'' AS ''YENI''
```

satırı ESKIKU ismi yerine YENI ismini yerleştirir. Bunun gerçekleşmesi için seçilen disket-te, YENI ismini taşıyan bir kütük henüz bulunmamalı, ESKIKU isimli bir kütük ise önceden varolmalıdır. Sayfa 640'ta, klavye aracılığıyla girilen bir dizi verinin bir sıralı kütüğe yazılması için, burada açıklanan komutları kullanan bir programın akış şeması gösterilmiştir. Veri girişinin sonunda program, kütüğün uzunluğunu (kesim olarak) verir ve girilen verilerin yeniden okunması ve bilgiyazımına devam eder.

Akış şeması, şimdiye kadar verilmiş olanlardan daha dar kapsamlıdır ve pratikte normal olarak kullanılan şekli yansıtmaktadır.

Bir programın akış şemaları, sonucun elde edildiği ayrıntılara inmeksizin, programın yerine getirmesi gereken tüm iş akışlarını biraraya getirmelidir. Bu ayrıntıların düzeyi komutların yazılması aşamasında belirlenebilir.

Örneğin sayfa 640'taki akış şemasında 500. kutusunda, girilen karakter zincirinin ilk üç karakteri üzerinde bir denetim öngörülmüştür.

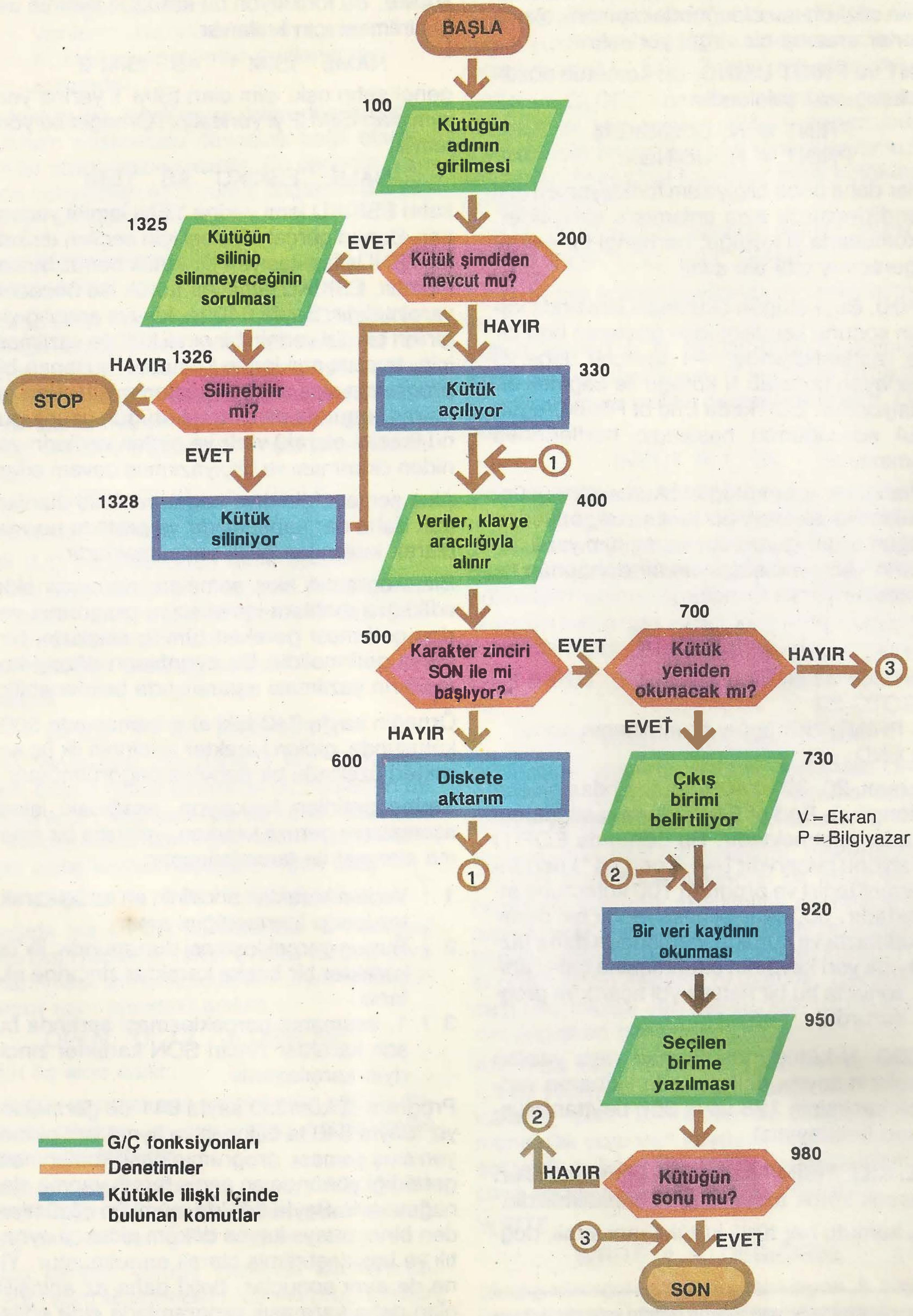
Yerine getirilen fonksiyon, aşağıdaki işlem aşamalarını gerekli kılarken, yalnızca bir sınama simgesi ile tanımlanmıştır:

1 / Verilen karakter zincirinin en az üç karakter içerip içermediğini sınar.
2 / Bunun gerçekleşmesi durumunda, ilk üç karakter bir başka karakter zincirine aktarılır.
3 / 1. aşamanın gerçekleşmesi şartında bu son karakter zinciri SON karakter zinciriyle karşılaştırılır.

Program dökümünü sayfa 641'de görmekteyiz. Sayfa 640'ta bulunan ve ayrıntılara girmeyen akış şeması, programın nasıl kodlanması gerektiği yönünde en geniş tercih yapma olanağını verir. Sayfa 641'de mümkün çözümlerden birini ortaya koyan döküm oldukça ayrıntılı ve basitleştirilmiş olarak sunulmuştur. Yine de aynı sonuçlar, belki daha az anlaşılır olan daha karmaşık programlarla elde edilebilir.

(*) Bazı araçlarda bu şart ters ön işarete sahiptir: —1 yerine +1.

SIRALI BİR KÜTÜĞÜN OKUNMASI VE YAZILMASI İÇİN ÖRNEK



SIRALI BİR KÜTÜĞÜN OKUNMASI VE YAZILMASINA ÖRNEK

```
10 ' **SIRALI BIR KUTUGUN OKUNMASI VE YAZILMASINA ORNEK
20 ' KUTUK=SIRA
30 ON ERROR GOTO 1000
100 INPUT "KUTUGUN ADI:",KA$
200 ' ** MEVCUDIYET SINAMASI
205 OPEN "I",1,KA$    'KUTUGE 1 SAYISI VERILDI
210 ' **EGER KUTUK MEVCUT DEGILSE BIR ONCEKI KOMUT HATA URETIR
215 ' BU HATA DENETIMI 1000 KOMUTUNA ILETILIR (BKZ.ON ERROR..)
300 '
305 '
310 '
320 '
322 ' KUTUK MEVCUTTUR
324 PRINT "KUTUK DISKETTE MEVCUTTUR"
325 INPUT "KUTUK SILINMELI MI (EVET/HAYIR)";CEV$
326 IF CEV$<>"EVET" GOTO 2030
328 CLOSE 1: KILL KA$
329 ' ALT-PROGRAM 1000 DEN DONUSTE VARIS YERI
330 OPEN "O", 1, KA$ 'KUTUK CIKIS OLARAK ACIKLANIYOR
331 PRINT
332 PRINT "GIRISLER (SON) SOZCUGUYLE SONA ERIYOR"
335 PRINT "(O)OUTPUT TURU BIR KUTUK URETILDI"
338 PRINT
400 INPUT "VERILER(4 SAYISAL VE 1 KARAKTER ZINCIRI):",N1,N2,N3,N4,A$
500 IF A$="SON" GOTO 700
600 WRITE #1,N1,N2,N3,N4,A$
610 GOTO 400
700 ' **GIRISLERIN SONU
710 INPUT "KUTUK YENIDEN OKUNMALI MI (EVET/HAYIR)";CEV$
720 IF CEV$="HAYIR" GOTO 2030
730 PRINT "KUTUGUN DOKUM BILGIYAZIMI HANGI GEREGLE YAPILMALIDIR?"
740 INPUT "E=EKRAN  B=BILGIYAZAR";F$
750 IF F$<>"E" AND F$<>"B" THEN PRINT "***HATA**":GOTO 740
900 CLOSE 1:OPEN "I",1,KA$    'KUTUK KAPATILDI VE (I) TURUNDE YENIDEN ACILDI
910 K=0     'VERI KAYITLARININ SAYACI SIFIRLANIR
920 INPUT #1,N1,N2,N3,N4,A$    ' BIR VERI KAYDININ OKUNMASI
930 K=K+1      ' VERI KAYITLARININ SAYACI
940 PRINT "VERI KAYDI N.:";K
950 IF F$="E" THEN PRINT N1,N2,N3,N4,A$
960 IF F$="B" THEN LPRINT "VERI KAYDI N.:";K:LPRINT N1,N2,N3,N4,A$
980 IF EOF(1) THEN 2000    'THEN 2000 (GOTO 2000) E KARSILIK GELIR
995 GOTO 920
1000 ' ********** HATA DENETIMI **********
1010 PRINT "KUTUK MEVCUT DEGIL, YENI URETILMELI"
1020 RESUME 330
2000 '
2020 PRINT "OKUNAN :";K;"VERI KAYDI"
2030 PRINT " ********** RUN END*********"
2040 END
```

```
VERI KAYDI N.: 1
 12              23              43              54              ESRA
VERI KAYDI N.: 2
 67              25              48              1               AYSE
VERI KAYDI N.: 3
 26              27              29              33              DERYA
VERI KAYDI N.: 4
 123             56              78              5               HASAN
```

Verilerin Sıralı Bir Kütüğe Eklenmesi

Sıralı bir kütüğün veri kayıtları adreslenemezler, bu nedenle veriler, doğrudan kütüğün sonuna yazılarak eklenemezler. Bunun yerine, değiştirilecek kütüğün bir kopyası yapılmalı ve bu yapılırken yazma aşamasında yeni veri kayıtları eklenmelidir. Takip edilecek yöntemin ayrıntıları aşağıda sıralanmıştır:

1 / İlk kütük ''I'' şartı altında açılmalı (program için bu bir giriş anlamını taşır).
2 / ''O'' şartı altında ikinci bir kütük açılmalı; ikinci kütük birincisinin verileri artı eklenecek veri kayıtlarını içermelidir.
3 / Birinci kütük ikinci kütüğe aktarılmalı; bu yolla ikinci kütükte birincisinin verilerinin bir kopyası üretilir.
4 / Yeni veri kayıtları ikinci kütüğe eklenmeli.
5 / Birinci kütük artık gerekli değildir ve önce kapatılıp sonra silinir.
6 / Son adım, birinci kütüğün adının ikincisine verilmesinden oluşur; böylece ilk kütükle aynı isme sahip, ancak daha uzun olan bir kütük elde edilir.

Doğrudan Erişimli Kütüklere (Random Files) Erişim İçin Fonksiyonlar

Doğrudan erişimli kütükler en çok kullanılan kütüklerdir, çünkü istenen veri kaydına doğrudan erişimi mümkün kılarlar. Ayrıca disket üzerinde iki tabanlı sayı sistemiyle saklanmaları nedeniyle, ASCII'de saklanan sıralı kütüklerden daha az yer kaplarlar. Doğrudan erişimli kütükler için tür belirtilmelidir (''I'', ''O''), çünkü bu kütükler hem okuma için, hem yazma için kullanılabilir.

İşletim sistemi bu kütüklerle veri değişimi için bir ara bellek kullanır (bu kavramla, özel fonksiyonların akışı sırasında,ara saklama için hizmet gören bir bellek alanının anlaşıldığını hatırlayınız; örneğin çevresel gereçlerle veri değişiminde de bu durum olabilir). Diskete yazma işleminin yerine getirilmesi gerektiğinde, daha sonra diskete yüklenecek olan değerler ara belleğe verilmiş olmalıdır.

Bunun tersi olarak, disketten okumada veriler alınır ve ara belleğe verilir; buradan da uygun komutlarla kendilerini talep eden programa iletilirler.

İsmi kullanıcı programında tanımlanan ara belleğin yalnızca disketle veri değişiminde kullanılabileceğine dikkat edilmesi gerekir. Başka herhangi bir kullanımında burada içerilen veriler önce başka bir isimle tanımlanan farklı bir bellek alanına aktarılmalıdır.

Doğrudan erişimli kütüklerin işletiminde kullanılan fonksiyonlar ve komutlar aşağıda verilmiştir:

OPEN, FIELD, CLOSE, PUT, GET ,LOC, KILL, NAME.

Bunlar, verilere doğru yazı düzenlerinin verilmesinde gerekli olan tüm karakter zinciri fonksiyonlarına eklenirler. Şimdi her fonksiyonun sözdizimini ve işlevlerini daha dikkatle gözden geçirelim. KILL ve NAME için, daha önce sıralı kütüklerde anlatılanlara bakınız.

OPEN. Sözdizimi aşağıdaki şekildedir:

OPEN ''R'', # N,''ISIM'',L

Burada:

R doğrudan erişimli bir kütük olduğunu,
N kütüğün program içinde gösterildiği sayıyı (bu komutta # sembolü her zaman gerekli değildir)
ISIM kütüğün adını gösterir. Buraya aynı zamanda kütüğün üzerinde bulunduğu disket birimi de eklenebilir (örneğin A:ISIM). Disket birimi tam olarak belirtilmediğinde, sistem son kullanılmış olan disket birimini (drive) kullanır.
L Mantıksal veri kaydının uzunluğunu bayt olarak ifade eder.

Örneğin bu yöntemle;

OPEN''R'',3,''B:TEST'',64

komutu B disket biriminde TEST isimli bir doğrudan erişimli kütük açar. Kütük 3. kanala verilir ve veri kayıtlarının uzunluğu 64 bayttır.

OPEN komutu **kütüklerin düzenlenmesinde** de kullanılabilir.

Yukarıdaki örnekte TEST kütüğü mevcut ise açılır ve içeriği erişilebilir hale gelir. Kütük yoksa, sistem aynı komutla bu kütüğün yaratılmasını ve daha sonra açılmasını sağlar.

Bu mantık beraberinde bazı zorluklar getirebilir. Karmaşık programlar sık sık farklı disketlerde bulunan kütüklerle çalışmasını gerekli kılarlar.

Kullanıcı, disketi değiştirmeyi unutursa doğru diskette yalnızca kütüğü açan OPEN komutu, gerçekte yanlış diskette aynı isimde yeni bir kütük açar. Bundan kaçınmak için, kütük INPUT şartı altında (Tür ''I'') sıralı kütük olarak açılmalıdır.

Çünkü kütük mevcut olmadığında, bu komut bir hata iletisine yol açar ve program akışını durdurur. Sistem ancak kütük önceden mevcut olduğunda hiçbir hata göstermez. Daha sonra kütük, sıralı yöntemle yalnızca sınama için yapılmış olan açma işlemi tersine çevrilerek kapatılır ve ''R'' (random) şartı altında bir daha açılır.

Sunulan tekniği belirginleştirici bir program aşağıdaki şekilde yazılabilir:

```
10 'Doğrudan erişimli bir kütüğün açılışı
20 ON ERROR GOTO 100
30 OPEN ''I'',1,''DENEME''
40 CLOSE 1
50 PRINT ''Kütük çalışıyor''
60 OPEN ''R'',1,''DENEME'',64
70 '
80 'Kullanıcı programı sırada
90 '
100 'HATA
110 PRINT ''Kütük çalışmıyor''
120 INPUT ''Kütük düzenlenmeli'';A$
130 IF A$ <>''EVET'' THEN STOP
140 RESUME 60
```

Programın yalnızca göstermelik bir anlamı vardır. Çünkü 20. satırdaki ON ERROR komutunun kullanımı, hata kaynağı ne olursa olsun, kütüğün düzenlenmesini engeller. Kullanıcı programlarında kütüğe ait komutlar (110. 120. 130. 140. satırlar) hata kaynağından bağımsız hale getirilmelidir. Bir sistem hatası ortaya çıktığında ERR değişkeninde hatanın kodu ve ERL'de ise hataya neden olan satırın numarası yazılır (ERR ve ERL sistem için ayrılan isimlerdir). Bu değerlerin denetlenmesiyle hata kaynağının nedeninin kütük yokluğunda mı yattığı, yoksa başka bir yerde mi aranması gerektiği saptanabilir. Örneğin işletim sisteminin ERR=53 (CP/M'deki durum) ile yanıt verdiğini varsayalım, bu durumda yukarıdaki program aşağıdaki şekilde değiştirilmelidir:

```
100 'HATA
110 IF ERR=53 AND ERL=30 GOTO 130
120 ON ERROR GOTO 0
130 'programın 110, 120, 130, 140. satırla-
      rındaki
140 'önceden gösterilen komutlar
150 'aynen takip eder
```

110. satır hatanın cinsini ve kaynağını araştırır; eğer değerlerden hatanın kütüğün yokluğundan meydana geldiği çıkarılamıyorsa, denetim, sistemin hata arama mekanizmasını tekrar harekete geçiren 120. satıra geçer; normal hata arama iletisini verir ve program durur.

FIELD. Daha önceden açılmış olması gereken bir kütük, okuma ve yazma işlemlerinde kullanılacak olan bir ara belleğe iletilir. Sözdizimi,

FIELD 1, 64 AS B$

şeklindedir. Satır tüm veri kaydına (64 bayt) B$ ara belleğini ayırır. Bu şekilde tanımlanan ara bellek, disketle veri değişimi için bir ''taşıyıcı''dır; başka amaçlar için kullanılamaz, özellikle muhtemel INPUT ve düzenleme komutlarına eklenemez. Bunu takip eden, ara bellekten programa veri aktarımı LSET veya RSET komutları ile sağlanır. Veri kaydının tümü, her birinde uzunluğun bayt olarak saptanması gereken birden fazla ara belleğe bölünebilir.

Örneğin daha önce gözden geçirilen düzen (64 AS B$) dört ara belleğe aşağıdaki şekilde bölünebilir:

FIELD 1, 12 AS N$, 30 AS C$, 20 AS V$,
2 AS F$

64 karakterlik veri kaydı dört ara belleğe bölünmüştür: 12 karakterlik N$, 30 karakterlik C$, 20 karakterlik V$ ve 2 karakterlik F$.

CLOSE. İşlevi sıralı kütüklerdekinin aynısıdır. CLOSE # N satırı verildiğinde kütük numarası N'in (önceden OPEN ile tanımlanmıştı) kapatılmış olduğu tanımlanır ve aynı kütük bir daha açılıncaya kadar içeriğine ulaşılamaz. CLOSE komutu kapatılan kütüğe ayrılmış yardımcı ara belleği serbest bırakır, böylece bu ara bellek başka bir kütük için kullanılabilir. Doğal olarak bu kütük öncekinin kapanmasından sonra açılmalıdır.

PUT. Bu komutla ara belleğin (veya ara bellekler in) içeriği diskete aktarılır. Komutta, hangi kütüğe yazılmak istenildiği belirtilmelidir.

Bunu yapmak için, OPEN komutunda tanımlanmış olan kanal ve pozisyon (konum), yani mantıksal veri kaydının numarası verilmelidir.

Örneğin PUT # 1,75 komutu, 1 kütüğünün 75. veri kaydını yazar (bazı durumlarda # simgesi ihmal edilebilir).

Doğrudan erişimli bir kütüğün uzunluğu önceden tanımlanmamalıdır. Yeni veri cümleleri gerekli olduğunda, sistem bunu dikkate alır ve

TEST 18'in Çözümleri

1 / Yüklemler, karakterlerle birlikte saklanan ve onların özelliklerini tanımlayan göstergelerdir. Bundan dolayı ekranın belleği, yalnızca karakterin ASCII kodunu saklamak için gerekli olandan daha fazla pozisyona sahiptir.

---

2 / 20. ve 30. satırlar. 20. satır altı karakterlik bir karakter zincirini (DENEME) ekranın bir satırına yirmi defa yazmaktadır, böylece toplam olarak 120 karakter elde edilir. Normal olarak tek bir ekran satırı yalnızca 80 karakter alabilmektedir. 30. satır aynı tür bir hatayı içermektedir: TAB(100), C değişkenini ekranda varolmayan 100. sütuna yazdırır (bu komut PRINT yerine LPRINT yazılarak bilgiyazar için geçerli hale getirilebilir).

---

3 / 30. satıra kadar olan döngü. Ekranın her satırına bir dizi veri yazar. Normal olarak ekranda 24 satır olduğundan, ilk değerler kaybolur, çünkü daha sonrakilere yer açmak için yukarı doğru kaymaları gerekir.

---

4 / Programın dökümü şöyledir:

```
10 '4. SORUNUN ÇÖZÜMÜ
20 'EKRANDAN VERİLERİN OKUNMASI
30 INPUT ''Yazılacak karakter zinciri'';A$
40 INPUT ''Hangi sütuna'';X
50 IF X=0 GOTO 160
60 INPUT ''Hangi satıra'';Y
70 IF Y=0 GOTO 160
80 '
90 'SINAMALAR
100 '
110 K=LEN(A$)'Yazılacak karakter zincirinin uzunluğu
120 IF (K+X) > 80 THEN PRINT ''Hata'': GOTO 40
130 '
140 PRINT FNP$(X,Y);A$
150 GOTO 30
160 END
```

---

5 / 140 komutu;

140 PRINT	FNP$(X,Y) (pozisyon)	INV$ (ters gösterim)	A$ (veri girişi)	NOR$ (normal gösterim)

şeklinde olur. Bu komut daha fazla satıra da yazılabilir; fakat başlangıca dönmeyi engelleyen ; simgesinin kullanılması gereklidir. Örneğin yukarıdaki şekil aşağıdaki şekle dönüştürülebilir:

```
140 PRINT FNP$(X,Y);'POZİSYON
142 PRINT INV$;      'TERS GÖSTERİM
144 PRINT A$;        'YAZILACAK KARAKTER ZİNCİRİ (VERİ GİRİŞİ)
146 PRINT NOR$       'NORMAL GÖSTERİM
```

Böylece bazı bilgiyazım satırları bir çentiğin değerine bağımlı kılınabilir ve çentiğin değerine göre farklı gösterim şekilleri elde edilebilir. Örneğin, aşağıdaki komutlar, yalnızca ÇENTİK 1'e eşit olduğunda ekranda bilgiyazım sağlarlar.

```
140 PRINT FNP$(X,Y)
142 IF FLAG=1 THEN PRINT INV$;
144 PRINT A$;
146 IF FLAG=1 THEN PRINT NOR$
148 IF FLAG <> 1 THEN PRINT
```

146. satır ekranı yeniden normal işlev tarzına getirir ve ÇENTİK=1 durumu için başlangıca geri dönmeyi sağlar (RETURN). Bunun tersi durumunda (ÇENTİK <>1) 144. satırda belirtilmemiş olan RETURN'un yerine getirilebilmesi için yalnızca PRINT komutu bulunan bir satır gereklidir.

kütüğün sayacını değiştirir (sayaç ve diğer bilgiler içerik dizininde bulunur).

Genellikle bu yöntem pek sakınca yaratmaz, yalnızca birçok veri uzunluğunun sık sık değiştirilmesi durumu gerekebilir. Örneğin başlangıçta üç kütük açıktır ve fiziksel olarak disket üzerinde birbirini takip eden üç alan kaplamaktadırlar. Birincisinin belli bir sınırın üzerine uzatılması gerektiğinde, yeni mantıksal veri kayıtları öncekilerle fiziksel olarak yanyana gelecek şekilde eklenemezler. Çünkü er veya geç ikinci kütüğün veri kayıtlarına rastlanacaktır. Bu sebepten dolayı birinci kütüğün genişletilmesi ayrı tutulur. Bu durum benzer şekilde ikinci veya üçüncü kütükte de meydana gelebilir (örneğin sonuncusu, önceki kütüklerin genişletilmesi sebebiyle, yerini doldurulmuş bulabilir).

Kütüğün farklı bölümleri yalnızca karmaşık bir adres tablosunun [kaplam (extend) tablosu] yardımı ile yeniden bulunabilir. Bu tamamen işletim sistemine bağlı olsa bile, programın yürütülmesi karmaşıklaşır ve yavaşlar. İki çözüm olabilir:

1 / Kütükleri içeren disketten belirli zaman aralıklarıyla yeni bir diskete kopya edilir. Aktarım sırasında sistem, tüm kütüklerin ve genişlemelerinin fiziksel olarak yanyana kalmalarına dikkat eder.

2 / Başlangıçtan itibaren kütükler öngörülen maksimum uzunluklarda üretilirler. Bu işlemde, öngörülen son veri kaydına istenilen bir değer yazılır.

Basılmış bir entegre devrenin planlanması.



Oyun İçin Birkaç KB

70'li yılların başlarında uçan daireler psikozunun hızla yayılıp, insanların, küçük yeşil Marslıların dünyayı istila edeceğinden bahsettikleri sıralarda, yeryüzünde ilk gerçek yabancı ortaya çıktı.

Bu yabancı, ismi ''PONG'' olan bir video oyunuydu. Küçük bir bilgisayarın kullanılmasıyla, televizyon ekranına çok benzeyen bir ekranda ilk defa olarak bir ping pong oyununu canlandırılıyordu.

Gerçi Pong'un dış görünüşü bir insanınkine benzemiyordu, ama yine de gezegenimizin sakinleriyle hemen uyum sağlayıp, barış içinde birarada yaşamak üzere anlaşmayı başardı. İlk olarak oyun salonlarında para otomatlarını, bozuk paralarıyla devamlı olarak besleyen insanların dikkatlerini üzerinde topladı. Bu inanılmayacak başarının etkileri, Amerika'dan tüm yer küreye yayıldı. Bu olayın bomba gibi yayılışını ilk gözlemleyen Pong'un bulucusu olan Nolan Bushnell'di. Hiç zaman kaybetmeden patent hakkını alarak, ATARİ firmasını kurdu. Bu firma endüstri alanında video üretimi için kurulan, ilk büyük girişimci firmadır.

Oyun salonlarında, parayla çalışan elektronik oyunların bu beklenmedik başarısından sonra ATARİ firmasının yönetim kurulu, PONG'un evlere de girebilecek yeni bir çeşidini piyasaya sürmeye karar verdi. Ekran olarak, evlerde son derece yaygın olan televizyon ekranından yararlanıldı. Pong ilk olarak 1975 yılında evlerde de kullanılabilen bir video oyunu haline geldi.

Bu yeni ev oyunu doğrudan doğruya, seyirci ile televizyon arasındaki ilişkiyi etkiledi. Böylelikle ilk defa olarak,artık ekranda sadece, seyirciye tek taraflı olarak sunulan görüntülerin dışında da birtakım şeylerin görülebileceği ortaya çıktı. Bu andan itibaren her televizyon aygıtını istasyonuna bağladığı farzedilen ve dokunulmazlığı olan egemenliği de bitmiş oldu.

İletişim tekelleri, seyirci ile televizyon arasında tek yönlü olarak kurdukları bu ilişkinin kurbanı oldular. Video oyunlarının kullanılması sayesinde ekran yepyeni bir ekrana oturarak, yeni ufuklara açıldı. Oyuncu, video oyunu bilgisayarı ile alternatif bir ilişkiye girecek, sürekli bilgi alışverişinde bulunuyor ve sadece ekrandaki çizimin güzelliği nedeniyle de olsa yeni ve gerçek birtakım duyarlıklar ediniyordu. O artık evindeki televizyonda başrol oynayan bir kahramandı.

Bugün çağımızın en tipik iki tüketim aracı arasında işlevsel bir ilişki kurulmuştur. Milyonlarca aile tarafından halen iyi bir alışkanlık olarak benimsenen televizyon ile kişisel bilgisayar arasında. Oysa bu ilk yıllarda değişik kullanımlarının zengin bir gelişme göstereceği umulmakla birlikte, henüz yeni ve keşfedilmemiş olan bir pazara atılan, ürkek ilk adımlardı. Video oyunları bu geniş alanın uygulama olanaklarından biriydi.

Bu aşamada video ve oyun sözcüklerinin İngilizce karşılıklarından ''video game'' sözcüğü türetildi. Bu sözcük, belirli bir oyunun, kurallarını akışını ekranda görüntüleyebilen elektronik bir aleti betimliyordu. Buradaki temel düşünce, makinenin, kullanıcı ile giriştiği söyleşi ve komut alış-verişi sayesinde gerçek bir oyunun olaylarını taklit edebilmesi idi.

Bu yüzden video oyunlarında, etkileşimli bilgisayarların bütün tipik işlevsel özellikleriyle, gerçeğe uygun bir benzerini bulabiliriz.

Bu oyun makinelerinin fiziksel yapısı bir klavye, bir veya iki oyun kolu ve normal bir televizyondan oluşur. Bilgisayar ile oyuncunun devamlı olarak karşılıklı bilgi alış-verişinde bulunduğunu biliyoruz. Bu alış-veriş mantığının esasından hareketle, video oyunlarının değişik bileşenlerinin yapısal ve işlevsel özelliklerini tanımlayabiliriz. Elektrik bağlantıları kurulduktan sonra, klavye ilk bilgiyi oyuncuya iletir. Bu ilk ileti bir ölçüde, çizgi film hilelerini andırır. Klavye, televizyon ekranında, oyundaki aktif şahısların veya cisimlerin üzerinde hareket edebilecekleri bir dekor meydana getirir. Dekor ve oyunun aktif şahısları televizyon görüntüsünü oluştururlar.

Oyuncu bu görüntülere tepki gösterir ve kendi tercihini belirtir. Karar verme hızı, oyunun gerektirdiği dinamiğe göre değişebilir. Kararlar bilgisayara kontrol mekanizmaları tarafından iletilir. Tıpkı bir şoförün viraja girerken direksiyonu çevirmesi veya durmak ya da hızlanmak için ilgili pedala basması gibi, kullanıcı, oyun sırasında kararlarını oyun kolu ya da bir pedal aracılığıyla bilgisayara iletir. Oyuncunun mekanik hareketlerle verdiği bu iletileri, komut alıcı (oyun kolu ya da pedal) bilgisayarın kolayca yorumlayabileceği elektrik sinyallerine dönüştürür.

Bir saniyeden çok daha kısa bir süre içersinde bu komutlar bilgisayara ulaşır ve oyunun doğru akışına uyup uymadıkları denetlenir. Denetimden sonra kontrol programında saklanan bu komutun gerekleri doğrultusunda oyunun yeni bir evresi televizyon görüntüsü olarak oyuncuya sunulur. Etki ve tepkinin devamlı değişmesiyle, taklidin daha etkileyici olmasını sağlamak için, video oyunu aktörlerin pozisyonlarını değiştirir, yeni dekorlar ve manzaralar çizer, renkleri değiştirip müzik eşliği sağlar. Sunulan televizyon görüntüsünün çeşidine göre iki farklı oyun kategorisi vardır. Birincisi, oto sürücüsü örneğinde olduğu gibi, kontrol mekanizmasının hiçbir aktörü hareket ettirmeden, sanki oyuncu bir arabanın ya da uçağın içindeymişcesine bütün görüntüyü genel olarak değiştirdiği, ''öznel'' olarak tanımlanan oyunlardır. İkinci kategori olan ''nesnel'' oyunlarda ise, oyuncu aktör olarak ekranda temsil edilir ve bu aktörü kendi komutlarıyla hareket ettirir.

Video oyunları için hazırlanmış her bilgisayar, her iki oyun türünü de uygulayabilir. Bunun gerçekleştirilmesi için sadece oyun programının saklanmış olduğu oyun kasetini (''cartridge-kartuş'') değiştirmek yeterlidir. Kasetler, video oyunları pazarının, ticari karın asıl büyük bölümünü getiren en etkin parçalarıdır. Oyun değiştirilmek istendiğinde, araçları değiştirmek zorunda kalmaktan kasetler sayesinde kurtulunmuştur. Pong oyununun bütün verileri bir belleğe yüklenmiştir. Daha ileri kuşak video oyunlarında (örneğin ATARİ firmasının VCS 2600) aynı program, araçtan ayrılabilen bir belleğe yazılır. Bir ara bağlantı kablosu (slot) yardımıyla başka kasetlerle ikame edilebilir ve bu yöntemle farklı oyunlar üretilebilir. Teypler ve kasetler, plaklar ve manyetik bantlar (şerit) arasındaki çok yönlü benzerlik, yeni bir pazar açmış ve bilgisayar üretmeyen birtakım firmaların da video oyunları için kaset üretmeye başlamasını sağlamıştır. Gelişmekte olan bu yeni endüstriler yılda yüzlerce milyon dolarlık karlara ulaşmaktadırlar. Ayrıca bu yeni pazar, ''video oyunu tasarımcılığı'' olarak adlandırılan yeni bir meslek dalının da ortaya çıkmasına yol açmıştır.

Nasıl sanat ve müzik adamları kendi eserlerine imzalarını atıyorlarsa, oyun tasarımcılarının da satış kârından payları, izleyicileri ve hatta taraftarları vardır. Bu kişiler aslında 80'li yılların yeni yıldızlarıdır.

Çok umut vaat eden bir konuyu video oyunu şekline dönüştürmek, fevkalade kâr getirici bir ticaret olabilir. Üretici firma için her yeni kaset milyonlarca dolarlık bir kâr olanağı, yeni bir ELDORADO, anlamına gelebilir.

Ancak 1984 yılında bu efsanevi ülkeye giden yollar gittikçe daha fazla kalabalıklaşmaktadır.

Video oyunlarından umulan milyarlardan küçük bir diliminin bile hayali, küçüklü büyüklü bir çok işletmenin bu maceraya atılmasına yol açmıştır.

Bu ise kaçınılmaz olarak bir rekabete yol açmıştır. Bir pazarın ele geçirilmesi gibi delice bir yarışa kalkışan birçok girişimci, basit ve çok şeyler vaat eden bu pazarın hiç beklemedikleri güçlüklerle dolu olduğunu keşfettiler. Örneğin pazarda en çok iş yapan en ''iyi'' oyunun 'war games' (savaş oyunları) olduğu manzarasıyla karşılaştılar.

1983 yılının başlarında üçüncü kuşak video oyunlarının ortaya çıkması pazar paylarını etkili şekilde değiştirdi ve teknolojik açıdan yeni ürünlerin sunduğu yeni koşullara sahip olmayan bir çok firmanın yaşama şanslarını yok yetti.

Buna paralel olarak video oyunlarının psikososyal açıdan tehlikeleri de bir polemiğe yol açtı. Bir çok kitle iletişim aracının akrabaları sayılabilecek bu yeni teknolojiye karşı acımasızca açtığı yoğun kampanya, henüz kısa bir süre önce herkesin dikkatini üzerinde toplamış olan bu modern ilahları gözden düşürdü.

1983 yılında video oyunları sahnesinden birçok oyuncu çekilmek zorunda kaldı. Ancak oyun daha bitmemişti. Kasetlerin ve araçların satışının düşmesi pazarlama çabalarına daha fazla özen gösterilmesine yol açtı. Zaten kamuoyu oyunları toptan terketmemişti. ABD'de oyun salonlarının kazancı 5 milyar doları aştı. Bu sinema ve plak endüstrisinin toplam kazancından fazlaydı. Dünya sakinlerinin yarısının alışkanlığı haline gelen bir mucize ile karşı karşıyayız: Duyguların yerini alabilecek ucuz bir ürün.

Pratik olarak oyuncuya neredeyse her zevke uygun sınırsız macera olanakları sunulabiliyordu. İstatistikçilere göre ABD'de kasetler ve araçlar arasındaki satış oranı 1'e 7'dir. Yani oyuncu başına 7 ayrı video oyunu satılmaktadır. Kişinin seçenekleri arasında, yıldız savaşlarında uçuş (''vurmak ve tehlikeden kur-

tulmak'') fikrinintipik bir kullanımı karışık, kafa yorucu labirentlerden kurtulma, vahşi hayvanlardan, hayaletlerden kaçıp kurtulma veya dünyayı başka gezegenlerden gelen istilacılardan kurtarma gibi oyunlar vardır. Oyuncu evindeki rahat koltuğunda otururken, bir B 17'nin kumandak abininden nasyonel sosyalist Almanya'yı bombalayabilir. Aslında, pek rahat olmayan bu kabinde oyuncu bir benzetici yardımıyla bir ses üretici sayesinde insan sesleri üreterek korkuları ve duyguları canlıb ir şekilde taklit ederek, kendisini gerçek bir bombacı gibi hissedebilir. Canhıraş çığlıklar, yardım istemeler ve emirler değişik ritmlerde canlandırılır. Bu ortam içinde oyuncu pilot aniden kararlar vermek zorunda kalır. Aksi halde, vurulma riskine girer ve ölüsünü ancak Manş Denizi'nin sularında bulabilir. Bu şekildeki bir kişinin, kendisini, RAF (Royal Air Force-Kraliyet Hava Yolları)'nın ileri gelenlerinin rolünde görmesi oldukça komik bir manzaradır.

Ama yeni video oyunlarında daha şiddetli duygusal ve fiziksel olaylar araştırılıp denenmek-

Milano'da, AICA tarafından düzenlenen "Computer Play 83" sergisinden bir görünüm.



tedir. Günümüzde mikro elektronik ve lazer tekniğinin kullanılmasıyla, televizyon görüntülerinin manipülasyon ve taklitlerini yapabilecek karmaşık teknolojiler, ticari ürünlere dönüşmüştür. Bu ürünlerde özel yapısal elemanlarla (minik işlemciler vb.) zenginleştirilmiş geniş elektronik araçlar televizyon görüntüsünü sayısallaştırarak bilgisayarın anlayabileceği h ale getirir. (Bu sayısal ileti lazer ışınlarıyla metal ve plastik plakalara aktarılır.) Plakanın bir yüzüne kullanım için 250.000 farklı televizyon görüntüsü yerleştirilebilmektedir. Bu sayede, resimlerin sırasal bir düzenle yerleştirildiği bantlarından farklı olarak, plakanın yüzeyindeki, farklı ve fiziksel olarak birbirinden uzak iki resmin en kısa sürede ''okunması'' mümkün olmuştur.

Bombardıman uçağı örneğimize geri dönelim. Video oyunları tarafından üretilen resimler çok kaba olduklarından; doğal bir görüntü elde etm ek için, üzerinde bir veya birkaç bombardıman hedefi olan film görüntülerini ekrana yansıtmayı düşünebiliriz. Eğer video oyunlarının komutlarını sıklaştırırsak, bir B 17'nin güvertesinde olduğumuz hissine kapılabiliriz. Ancak komutlarımızı yönetemeyiz, çünkü komut ile film arasında neden sonuç ilişkisi veya daha kesin söylesek, hiçbir komut-hareket ilişkisi kurulamamaktadır.

Film hiçbir şeyden etkilenmeden devam etmektedir.

Oysa video plakları kullanılırsa, minik işlemci, birçok resim arasından, oyuncu pilotun manevra ve kararları ile en çok uyuşan resmi arayacaktır. Böylece gerçek bir etki-tepki ilkesine dayanan mantıksal bir akış oluşturur.

Tekrar B 17 kabinine dönüp, dümen manivelasını hareket ettirdiğimde, dümen pozisyonuna bağlı olarak bir dizi televizyon görüntüsü elde edeceğim; eğer manivelayı sola çevirirsem bir film netliğinde temsil edilen yeryüzünün, evlerin ve ufuk çizgisinin bana göre sağa doğru hareket ettiklerini görürüm. Düşmanlarımın üzerine ateş açar ve mermilerimin gidiş yönlerini izleyerek nişan alabilirim, böylece düşmanlarımın nasıl yok olduklarını görebilirim. Kendimi bir savaş uçağının kumandanı veya bir filmin başrol oyuncusu olarak hissedebilirim.

Böylece uçuş öğretmenlerinin rüyası olan, kusursuz bir temsil gerçekleşmiş oluyordu.



Bilgisayarla oyun oynamak, aynı zamanda eğitici de olabilmektedir.

Bu ise video oyunlarının geleceğine yeni bir boyut kazandırıyordu. Bu çeşidin bazı örnekleri ABD'de halen kullanılmaktadır.

"Dragon's Lair-Ejderhanın İni" bu yeni tür video oyunlarını belirtmek için kullanılan bir isimdir. Bu oyun, Walt Disney stilindeki bir miki filminin fotogramlarından oluşur ve oyuncunun kullanabileceği, içinde olayların geçtiği 42 tane son derece güzel manzara grafiğini içerir. Bu dekorlar içerisinde en güzel ortaçağ epikleri cereyan eder. Bir ejderhanın elinde bulunan güzel prensesin kurtarılması için, 200 tane kararın doğru verilmesi gerekir. Aksi halde prenses ölecektir.

Amerikan tıp öğrencilerinin tercih ettikleri konu ise daha az şövalyece ve dramatiktir.

Güzel prenses, karın zarı iltihabı olan, şişmanlama eğiliminden şikayetçi ve hiç vakit kaybetmeden ameliyat edilmesi gereken bir hastayla yer değiştirmiştir. Bu gerçek dekor, büyük dikkat ve hızla karar vermek gerektirdiği için, işe yeni başlayan doktorlara hiç kan dökmeden pratik bir deneyim kazandırabilir. Bu yeni video oyununun ilk örneği daha çok öğretici bir mahiyettedir, ama çok kısa bir süre içersinde evdeki ekranlarımızda da kullanabileceğimiz kesindir.

*Yayınevleri de video oyunlarına ilgi göstermektedir, örneğin 1983 yılında İtalya'da bu sektöre giren girişimcilerin arttığı gözlemlenmiştir. "**Video giochi**" isminde yeni bir derginin çıkmasıyla, bu olgulardan sözeden özel sütunlar ve çok sayıda başlıklar ortaya çıkmıştır. Bu ise bazı İtalyanlar tarafından oyunlar dünyasına ve kendi kültürlerine karşı savaşçı olmasa da bir istilâ olarak algılanmaktadır. italyanların çok eskibir oyun geleneği olduğu düşünülürse, bu çağdaş oyun furyasının Atlantiğin öbür tarafından başlatılmış olmasının yazık olduğu düşünülüyor. İtalyanlar'ın oyuna olan bağlılıkları yıllar boyu devam etmiş ve bugün birçok girişimci bundan yararlanarak bu kârlı meşguliyet sayesinde yerlerini sağlamlaştırmışlardır. Bu ise Akdeniz kültürünün tarihi köklerinden birinin canlı kalmasını sağlamıştır. Bugün İtalyanlar oyun pazarındaki Amerikan istilâsının karşısında bu geleneğin basit*

bir ticaret olarak kabul edilmesinden korkuyorlar. Bu takdirde bu oyun geleneği mirasının düşünülmeden kullanılması oyunun pedagojik değerini zedeleyecektir.

1984 yılında cesaretli bir seçim yapılmak zorunda kalındı ve gelecek yıllarda boş vakitlerimiz üzerinde daha belirleyici olacak bir olgunun temelleri ve karakteristik özellikleri tespit edildi. Günümüz toplumunun tipik bir özelliği olan, video oyunlarının zihinsel tasarımlarından bağımsız olduğu yanılgısına düşülmemelidir. Video oyunları gerçeğe uygun yeniden üretimlerdir. Kültürel ortam, oyunları ve video oyunlarını üretir. İnsanların bu oyunları sadece saldırı aracı ve savaşın temsilcisi olarak görmeleri, dünyanın kendi kendisini yok etmesi tehlikesini doğuracaktır. Günümüzde bile yaşamanın ne kadar zor olabileceğini kavramak için Afrika'ya yapılacak çok kısa bir yolculuk yeterli olacaktır. Afrika'da yaşayan insanların duygularını tatmin etmeleri için video oyunlarının benzetimlerine gereksinimleri yoktur! Bu yüzden düşmanlık duygularını temsil eden video oyunlarının (örneğin Yıldızlar savaşı) kabaca yayılması yerine, ahlaki ilkelere dayanan yeni bir oyun felsefesinin aranması gerekmektedir. Video oyunlarının, düşüncelerin katili olması tehlikesi ortaya çıkmıştır. Bu herhangi bir eleştirellikten uzak oyunlar, cazip yönleriyle (renk, müzik, ses) belleğimizi olumsuz yönde etkileyen zihinsel tasarımların yaygınlaşmasına yol açıyor. Üstelik bu eğlence aletinin en istekli kullanıcılarının çocuklar ve gençler olduğunu unutmamak gerekir. Bu oyunların gençlerin karakterlerini tehlikeli bir şekilde etkileyeceği gerçeği kasıtlı bir şekilde gizlenmiş olabilir. Anglosakson ülkelerde oyunlar çocukların ve gençlerin yaş gruplarına göre ayarlanmaktadır. Bu sosyal açıdan gelişmekte olan ülkelerde son derece yaygın bir zayıfları kollama ve koruma şeklidir. Benzer bir korumanın video oyunları için de uygulanarak, hizmet sundukları müşterilerin pedagojik içeriğine ve çeşidine göre sınıflandırılmalıdır.

Video oyununun varlığıyla tartışmak artık yararsızdır. Bu oyunlar bir ihtiyaca cevap verdikleri için var ama eğer oyuncuya, kendi yaratıcılığını destekleyecek biçimde yardımcı olabilirlerse, ancak o zaman olumlu bir toplumsal işlev gördükleri kabul edilebilir.

Psikolog Jean Piaget, ''Bir şeyi keşfetmek zorunda kaldığımız durumlardan daha çok şey öğreniriz'' der. Bu mantıkla düşünürsek, gerek video oyunlarının, gerek ev bilgisayarlarının değerli bir eğitim aracı olduğunu görmek kolaylaşır.

Aslında ikisinin arasındaki fark pek o kadar büyük de değildir: Ev bilgisayarları, video oyunlarının olanaklarını tanıdık bazı alanlarla ilgili bilgi-işlem kapasitesi ekleyerek zenginleştirir. Ev bilgisayarlarının temel yapısı, başka bazı çevresel gereçlere eklenerek genişletilebilen video oyunlarının yapısıyla aynıdır. Klavye, büyük bellek kapasitesi ve program kasetleri için kayıt cihazı, kullanıcıdan çok ciddi özel bazı beceriler istemeden, çok yönlü kullanım olanakları sunmaktadır. Aynı zamanda ev bilgisayarı televizyon ekranına basitçe bağlanabilir ve birçok çevresel gereç (örneğin, disk birimi, bilgiyazar, telefon hatlarının eklenmesi için bağlantı yerleri, optik yazıcı -ışıklı kalem-) ile donatılabilir.

1983 yılı, İtalya için gerçek bir ev bilgisayarları istilasının gerçekleştiği yıldı. Güvenilir kaynaklara göre, yalnızca yılbaşında 120.000 bilgisayar satıldı; bu, o güne kadar işitilmemiş bir rakamdı. Ev programlarında kullanılmak üzere alındığı tahmin edilen programların bir listesini yaptığımızda şöyle bir durumla karşılaşıyoruz: İlk sırayı hep oyun programları almaktadır. Boş zaman değerlendirme oyunlarıyla eğitsel oyunlar birlikte toplam program satışlarının %54'ünü oluşturuyor. Ev idaresi programları toplam satışın %17'sini, işletme programları %17'sini tutuyorlar. Metin işlem programlarının satışı ise toplam satışın ancak %3.5'una ulaşmıştır.

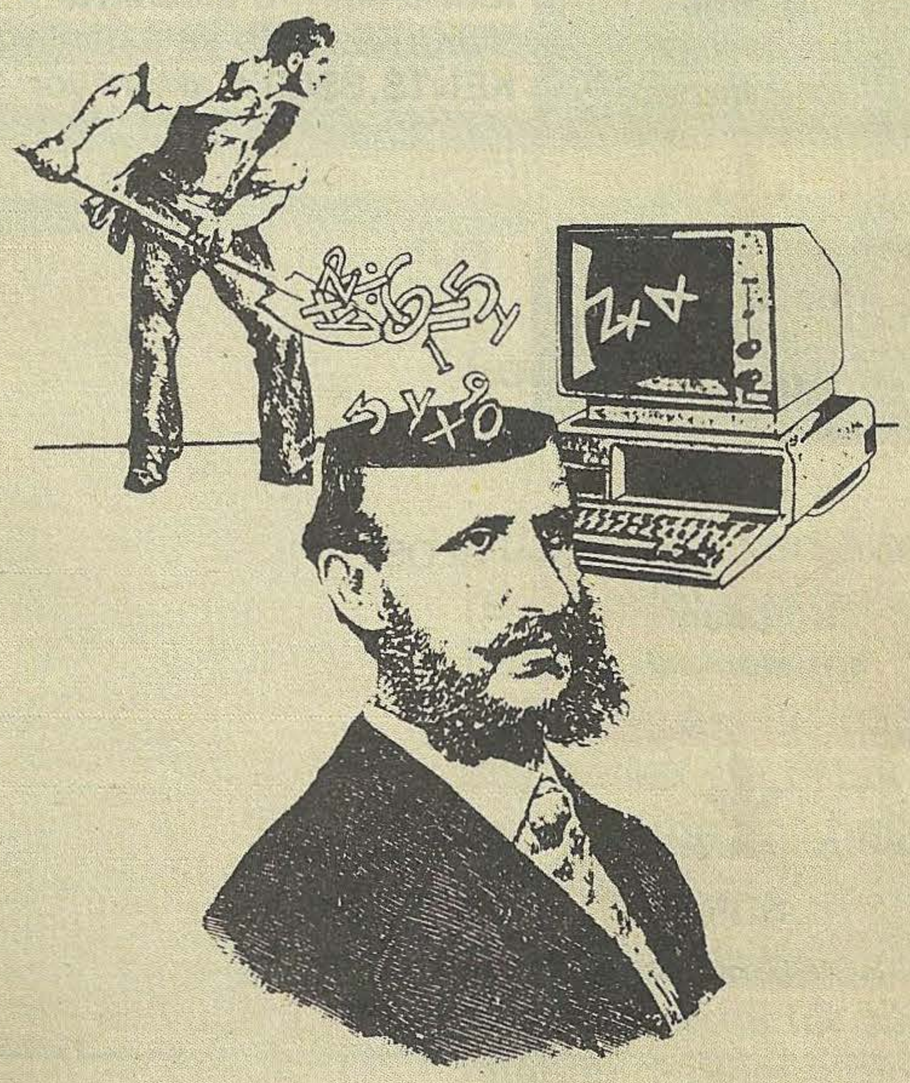
Bu olgunun ne anlama geldiğini kestirmek güç; ama şu kadarı söylenebilir: Kullanıcı kendi program üretebildiği halde hazır oyun programlarını tercih etmektedir. Bu ise insanların çoğunun,tatmin edemedikleri duygu ve beklentileri tatmin eden bir alet karşısından edilgin bir konumda kalmayı tercih ettiklerini gösteriyor.

Bu konunun daha ayrıntılı olarak incelenmesi için 2-3 Aralık 1983'te Milano'da bilgisayar oyunları konulu ''COMPUTER PLAY 83'' adlı sergiyle birlikte bir de kongre yapıldı. Kongre fikri ilk kez AICI (''Associazone Italiana per l'Informatica e il Calcolo Automatica'' = İtalyan Bilgi İşlem ve Otomatik Hesap Kurumu) öğrenci kesiminin düzenlediği ''Milano'da Bilgi İş-

lemin Gelişmesine Katkı'' adlı derste doğmuştu; amacı da herkesi (doçentleri, araştırmacıları, öğrencileri) oyun kültürü ve bu kültürün taşıdığı potansiyeller hakkında bilgilendirmekti.

Bu toplantıda bilgisayar oyunlarının yaygınlaşmasının doğurduğu sorunlar, bilgi-işlem alanında uzman elemanlara duyulan ihtiyaçla birlikte tartışılmıştı. Bu tartışmalarda bir dizi temel soru dile getirildi. Bunlardan bazıları şunlardır: Oyun nedir? Bir oyuntü rü nasıl gelişir? Oyunlar nasıl değerlendirilmeli? Bilgisayar oyunlarının eğitim üzerindeki etkileri nelerdir? Bilgisayar oyunlarının pazarı nedir, kimlerden oluşur? Bilgisayar oyunları nasıl öğretici kılınabilir? Hangi oyunlar hangi yaş gruplarına uygundur? Bütün bu sorular, konuşmacılar tarafından değişik açılardan ve farklı yaklaşımlarla çözümlenmiştir. Tartışmalar sırasında doğan genel eğilim, oyunların eğitim alanında daha yaygın olması gerektiği ve oyun hazırlamada didaktik ölçütlerin ön planda tutulması gerektiği yolundaydı.

Toplantıdaki önerilerden biri de, bilgisayar oyunları alanını bu doğrultuda yönlendirmek için özel sektörle işbirliğine gitmekti. Halen çeşitli ülkelerde olduğu gibi İtalya'da da birçok bağımsız programcı oyun programları yazmaktadır. Bu programcıların faaliyetlerini desteklemek e onların programlarından farklıb ir bilgisayar oyun anlayışı üretmek için, OYUNLARA ÇAĞRI sloganıyla bir yarışma düzenlenmiş ve katılan yüzden fazla yarışmacının oyunları bir uzmanlar jürisi tarafından değerlendirilmiştir.





En gelişkin kişisel bilgisayarlar bile, oyun oynamak için kullanılabilmektedir.

Kuşkusuz böyle bir ÇAĞRI, o yunların kalitesinin artmasına yol açacağı kadar, İtalyanların kendi oyun geleneklerinin Amerikan oyunlarının istilası karşısında ticarileşerek yok olacağı endişesini de bir ölçüde hafifletecektir. Çünkü programcılar oyun programı yazarken, kendi oyun gelenekleriyle beslenmiş olan duyarlıklarını yeni bir ortam içerisinde ifade etmeye çalışacaklardır. Böyle b ir etkileşimin ürünü olan oyunların da, oyuncular üzerindeki etkisi kuşkusuzb ugüne kadar alşıl mış olan, saldırganlık ve çatışma temelli oyunlardan çok farklı olacaktır.

(Antonio Varga, ''Associazone Italiana per l'Informatica e il Calcolo Automatico'' = İtalyan Bilgi İşlem ve Otomatik Hesap Kurumu)

GET: Bu komutun sözdizimi aşağıdaki şekildedir:

GET # N,R

Bu komutla N kütüğünün, R veri kaydındaki veriler okunur ve önceden N kütüğüne ayrılan ara belleğe aktarılır. örneğin;

GET # 1,75

komutu 1 kütüğünün 75. veri kaydında içerilen verileri okur.

LOC(N). N, ilgili kütük numarasına eşit olduğunda LOC(N) fonksiyonu, veri kaydı göstergesinin ''güncel'' değerini verir. Güncel değer, bir PUT veya GET komutunda son kullanılan veri kaydı numarasına 1 eklenmesiyle elde edilir. Örneğin;

10 GET 1,7
20 PUT 1,9
30 A = LOC(1)

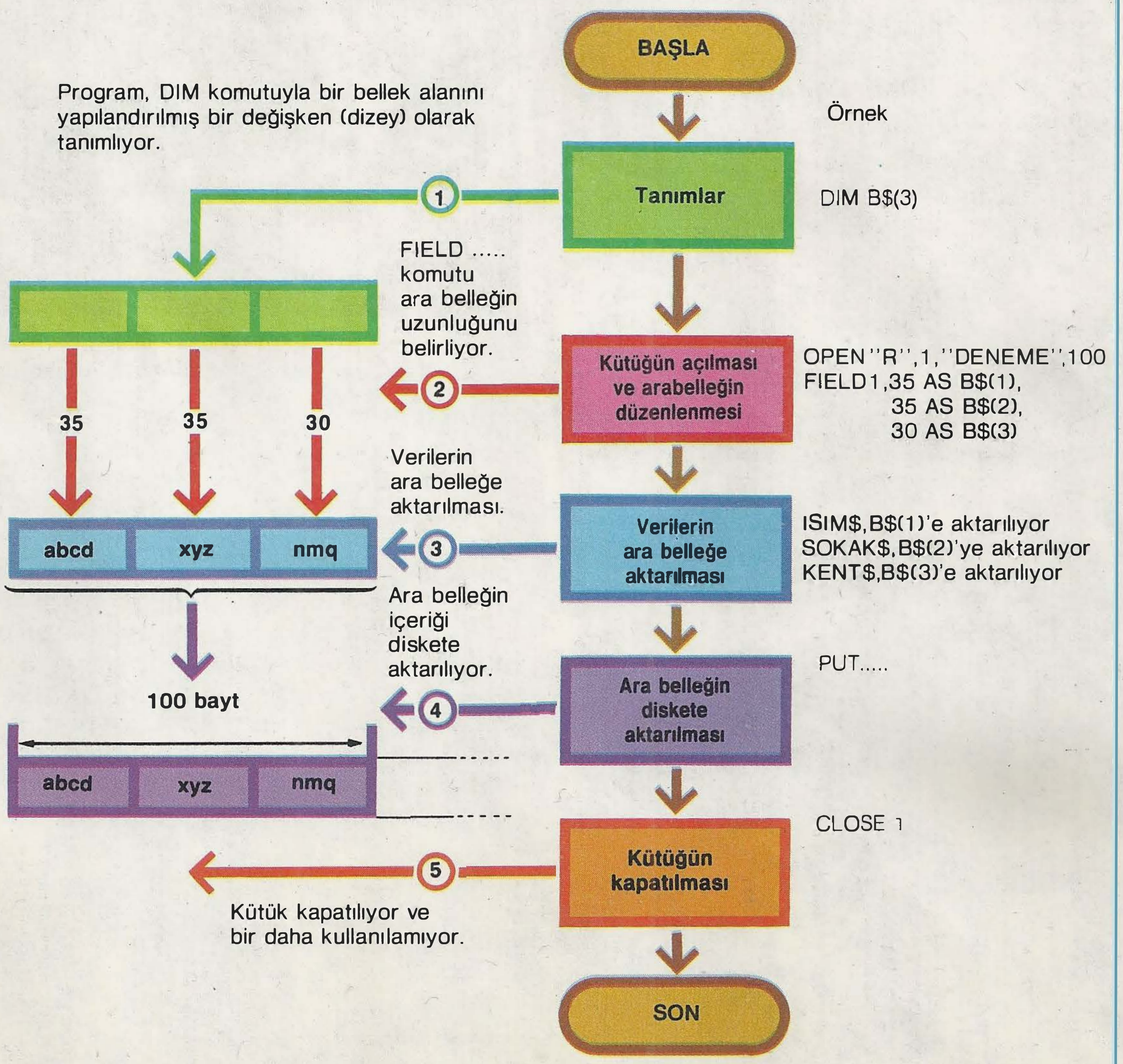
komutları A = 10 değerini verir, çünkü kullanılmış bir veri kaydının son numarası 9'dur (komut 20).

Bu sayfada altta, bir adres listesinin saklanması için diskete veri aktarılmasını gösteren bir şema verilmiştir.

Öngörülen alanlar şunlardır:

Soyadı ve isim = 20 + 15 karakter

DİSKETLE VERİ ALIŞVERİŞİNİN ŞEMASI



SIRALI BİR KÜTÜĞE YAZAN PROGRAM

```
10 '*** SIRALI BIR KUTUK YAZAN PROGRAM****
20 ' KUTUK = SIRKUT
30 INPUT "KUTUGUN ADI";AD$
40 AB$="A:"+AD$        'AB$ DEGISKENINDE, (AD$) ADLI KUTUGUN BULUNDUGU
50 '      DISKET BIRIMI (A:) SAKLANMISTIR
60 OPEN "O",1, AD$   '(OUTPUT) TURUNDE (1) NUMARALI BIR KUTUK ACILIYOR
70 '                  ' KUTUGUN ADI (AD$) DEGISKENINDE BULUNMAKTADIR
80 PRINT "VERI GIRISI (0)IN GIRILMESIYLE SONA ERER"
90 PRINT
100 INPUT "YAZILACAK SAYISAL DEGER";V
105 IF V=0 GOTO 150
110 INPUT "YAZILACAK KARAKTER ZINCIRI";A$
120 PRINT #1,V,A$       ' 1 KUTUGUNE A$ VE V DEGISKENLERI YAZILIYOR
130 GOTO 80             ' GIRISLER ICIN YENI BR ISLETIM BASLIYOR
140 '
150 PRINT "****** GIRISIN SONU **********"
160 CLOSE 1             ' 1 KUTUGU KAPATILIYOR. BIR SONRAKI SATIR INPUT
170 OPEN "I",1,AD$      ' KOSULU ALTINDA YENIDEN ACILIYOR (PROGRAM
180 '                     OKUMAYA DEVAM EDEBILDIGI SURECE)
190 INPUT #1,V,A$       ' VERILER KUTUKTEN OKUNUR VE
200 '                     V VE S$ DEGISKENLERINE AKTARILIR
208 LPRINT
210 LPRINT "OKUNAN SAYISAL DEGER=";V
212 LPRINT
220 LPRINT "OKUNAN KARAKTER ZINCIRI=";A$
230 IF EOF(1) GOTO 300      ' KUTUK SONA ERDIGINDE 300 SATIRINA ATLANIR
240 GOTO 190                ' AKSI DURUMUNDA OKUMA ICIN YENI BIR GECIS BASLAR
250 '
260 '
300 PRINT " ********* RUN END **********"
310 END
```

```
OKUNAN SAYISAL DEGER= 1234

OKUNAN KARAKTER ZINCIRI=DERYA

OKUNAN SAYISAL DEGER= 1567

OKUNAN KARAKTER ZINCIRI=HASAN

OKUNAN SAYISAL DEGER= 7890

OKUNAN KARAKTER ZINCIRI=TALAT
```

Sokak ve ev no = 30 + 15 karakter
Şehir ve kod no = 30 karakter

Burada her veri alanı için bir tane olmak üzere üç adet ara bellek gerekmektedir. Bu ara bellekler karakter zincirlerinden oluşan herhangi üç isimle çağırılabilirler. Ya da her üç ara bellek de ortak bir isimle adlandırılabilir; bu takdirde ara bellekler daha sonra bir dizin yardımıyla ayırdedilebilir.

Başka bir deyişle, dizinlenmiş bir karakter zinciri değişkeni (örneğimizde üç değerli) kullanılabilir. Böylece giriş ve çıkış işlemlerinde ara

belleğin işleri için ayrılmış olan yalnızca bir tek isim olur.

Verilerin yazı düzenleri ve kodlanmaları

Sıralı kütüklerde veriler ASCII yazı düzeninde saklanırlar, fakat alt-programlar ve programlar tarafından sayı olarak da kabul edilebilirler. Sayfa 653'te üstte, 60. satırda 1 numara ile açılan sıralı bir kütüğe, bir sayısal değer, yani bir sayı (satır 100) ya da bir karakter zincirinden (satır 110) oluşan ve klavye aracılığıyla verilen verileri yazan bir programın dökümü gösterilmiştir. Sayısal değer olarak 0 verildiğinde, (105. satır) giriş işlemi sona erer ve program diskete yazılanların yeniden okunmasına döner. Kütüğün okunabilmesi için öncelikle kapatılması (''O'' kütüğü olarak tanımlanmasından kaçınmak için) ve bundan sonra ''I'' şartı altında tekrar açılması gerekir. 120. ve 190. satırlardaki PRINT ve INPUT komutlarının, bir kütükle ilgili olduklarından # simgesini içerdiklerine dikkat edilmelidir. Bazı bilgiyazarlarda, bizim örneğimizde olduğu gibi # sembolü £ olarak yazılır. Türkçe karakterleri basabilen bazı bilgiyazarlarda ise bu karakterler yerine Ğ basılır.

Doğrudan erişimli kütüklerde veriler ikili sistemle saklanırlar. Ancak yine de ASCII'de girilmeleri gerekir, çünkü karakter zincirleri için bir yardımcı ara bellek kullanılmaktadır.

Sözü edilen bu durum, bu tip kütüklerin G/Ç işlemlerinde karşılaşılabilecek tek güçlüktür. Yine de BASIC'e özgü fonksiyonların kullanımı ile kolayca üstesinden gelinebilir.

Verilerin dönüşümü için kullanılabilecek en önemli fonksiyonlar şunlardır:

MKI$, MKS$, MKDE$	bilgiyazım sırasında sayısal değerleri karakter zincirlerine dönüştürürler.
CVI, CVS, CVD	okuma sırasında (karakter zincirlerini sayısal değerlere dönüştürürler).

Bu fonksiyonlar, doğrudan doğruya kullanılabilir ve kullanıcıya zorluk çıkarmazlar.

Yan sayfada, üstte, doğrudan erişimli bir kütükteki sayısal verilerin yazılması ve okunması için sözü edilen fonksiyonları kullanan bir programın dökümünü görmekteyiz. Verilerin dönüştürülmesi STR$(R) ve VAL(A$) fonksiyonlarının kullanımı ile de sağlanabilir. Sayfa

Falcon Jet Company'de kullanılmakta olan bir ADM-bilgisayarı



DOĞRUDAN ERİŞİMLİ BİR KÜTÜĞE YAZAN PROGRAM

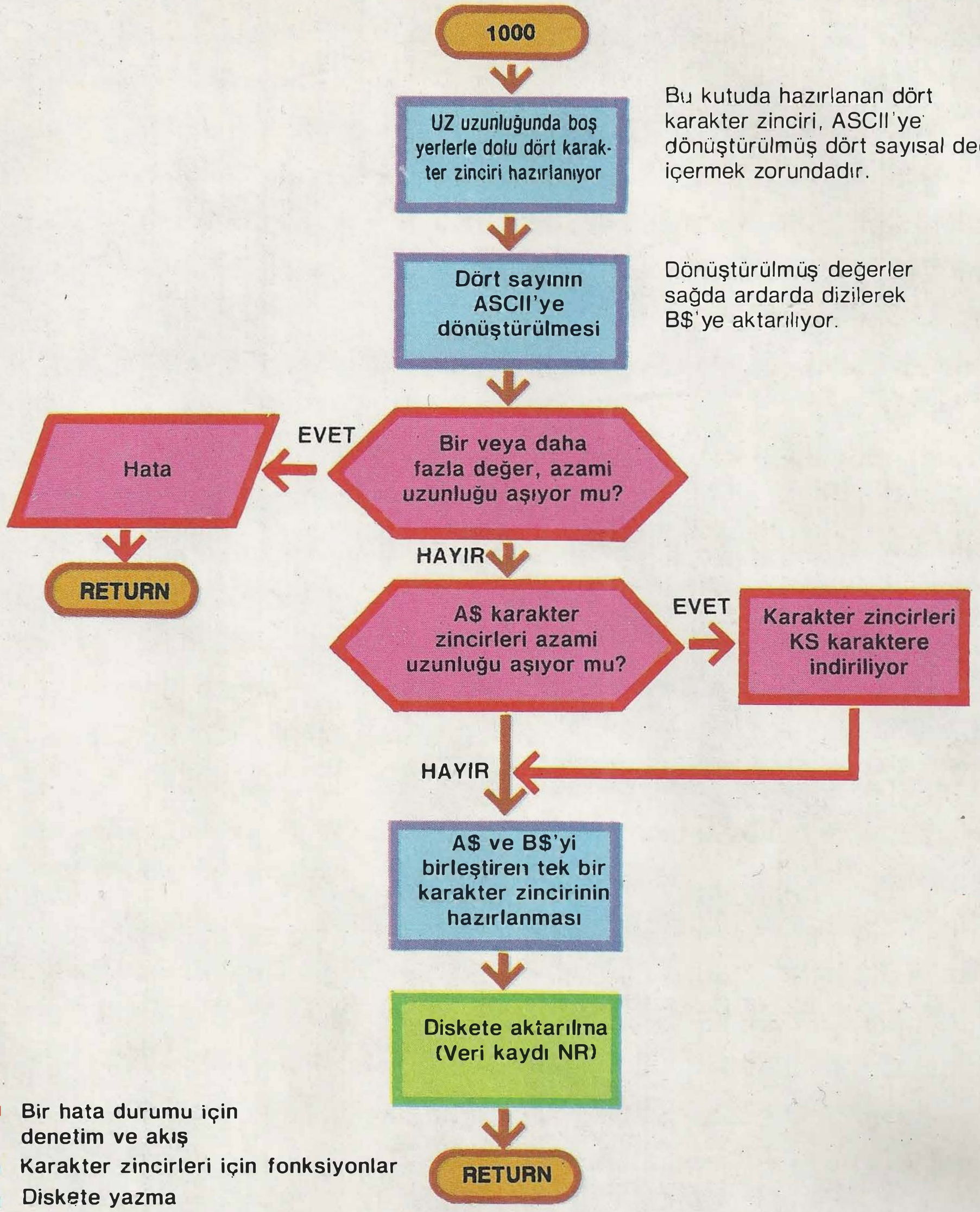
```
5  ' *** DOGRUDAN ERISIMLI BIR KUTUGE YAZAN PROGRAM **
10 ' DISKETTE (G/C) ISLEMLERINDE SAYISAL DEGERLERIN DONUSTURULMESI
20 ' ORNEGI (FILES RANDOM=DOGRUDAN ERISIMLI KUTUKLER
30 ' KUTUK = TRAVAL
40 OPTION BASE 1
50 DEFINT I      ' (I) ILE BASLAYAN ISIMLER TAMSAYILI
60 '  DEGISKENLERLE ILGILIDIR
70 DEFSNG R      ' (R) ILE BASLAYAN ISIMLER TEK HASSASIYETTE
80 ' DEGISKENLERLE ILGILIDIR
90 DEFDBL D      ' (D) ILE BASLAYAN ISIMLER CIFT HASSASIYETTE
100 ' DEDISKENLERLE ILGILIDIR
110 DIM BF$(3)   'G/C ARA BELLEK (HERBIRI BIR DEGISKEN TURU ICIN)
120 OPEN "R",1,"DENEME",30       'VERI KAYDININ UZUNLUGU 30 BAYTTIR
130 FIELD 1,10 AS BF$(1),10 AS BF$(2),10 AS BF$(3)
140 ' ** HER ARA BELLEGIN UZUNLUGU GEREKENDEN FAZLADIR
150 '
160 ' ** DISKETE YAZILACAK SAYISAL DEGERLERIN OKUNMASI
170 '
180 INPUT "TAMSAYILI DEGER";ITAM
190 INPUT "TEK HASSASIYETTE DEGER";RTEK
200 INPUT "CIFT HASSASIYETTE DEGER";DCIF
210 '
220 ' *** ITAM, RTEK,DCIF ISIMLERI ASCII YE
230 ' DONUSTURULMESI GEREKEN GIRIS-DEGISKENLERIDIR
240 '
250 A1$=MKI$(ITAM)       ' (MKI$,MKS$,MKD$) FONKSIYONLARI
260 A2$=MKS$(RTEK)       ' SAYISAL BIR DEGERI ASCII
270 A3$=MKD$(DCIF)       ' KARSILIGINA DONUSTURUR
280 '
290 ' ** ICINE YAZILMAK ISTENEN VERI KAYDININ NUMARASININ GIRILMESI
300 INPUT "HANGI VERI KAYDINA YAZILMASI GEREKIYOR";IR
310 ' ** VERI KAYDININ NUMARASI TAMSAYI OLMALIDIR
320 '
330 ' ** ARA BELLEGE AKTARIM (YARDIMCI BELLEK)
340 '
350 BF$(1)=A1$
360 BF$(2)=A2$
370 BF$(3)=A3$
380 '
390 ' ** DOGRUDAN ERISIMLI BIR KUTUGE YAZMA **
400 ' BIR SONRAKI SATIR 1 KUTUGU VE IR VERI KAYDINDAN (BF$) ARA
405 ' BELLEGININ ICERIGINI DISKETE  AKTARIR
410 PUT #1,IR
420 INPUT "DEVAM (EVET/HAYIR)";CEV$
430 IF CEV$="EVET" GOTO 180      ' YENI YAZIM
440 '
450 ' ** YAZILMIS OLAN VERILERIN YENI OKUNMASI **
460 '
470 INPUT "HANGI VERI KAYDI OKUNMALI";IR
480 GET #1,IR        ' (IR) VERI KAYDINDAN VERILER OKUNUR VE (BF$)
485 '     KUTUGUNE AYRILMIS OLAN ARA BELLEGE AKTARILIR
490 A1$=BF$(1)
500 A2$=BF$(2)
510 A3$=BF$(3)
520 ITAM=CVI(A1$)        '(CVI,CVS,CVD) FONKSIYONLARI
530 RTEK=CVS(A2$)        ' BIR ASCII DEGERINI SAYISAL
540 DCIF=CVD(A3$)        ' KARSILIGINA DONUSTURUR
550 LPRINT "OKUNAN VERILER:";ITAM,RTEK,DCIF
560 INPUT "DEVAM (EVET/HAYIR)";CEV$
570 IF CEV$="EVET" GOTO 470
575 CLOSE #1
580 END
```

656'da bu fonksiyonları kullanarak veri kaydını hazırlayan ve diskete yazan bir alt-programın akış şemasını görmekteyiz.

Sayfa 657'de bunun tersi olan işlevi yerine getiren, yani veri kayıtlarını okuyan bir alt-programın akış şeması verilmiştir. Her iki alt-

BİR VERİ KAYDININ HAZIRLANMASI VE YAZILMASI

Rutine girişler : NR = Yazılan veri kaydının numarası
V(4) = Veri kaydına yazılacak sayısal değerler
UZ = Sayısal değerlerin (rakam sayısı olarak) azami uzunluğu
A$(2) = Sayısal değerlerle birlikte veri kaydına yazılacak karakter zinciri
KS = Her karakter zincirinin karakter sayısı olarak azami uzunluğu

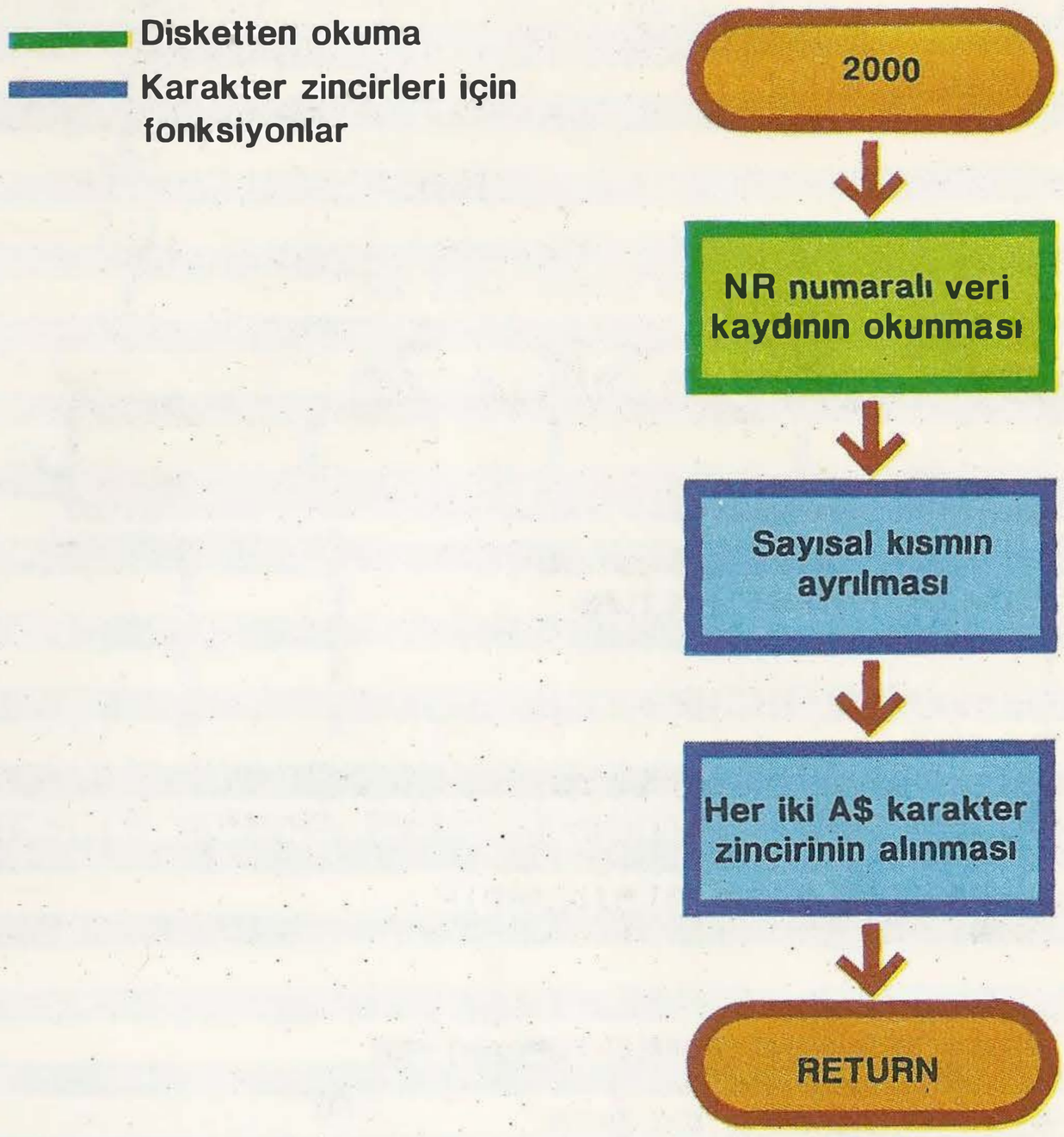


Bu kutuda hazırlanan dört karakter zinciri, ASCII'ye dönüştürülmüş dört sayısal değer içermek zorundadır.

Dönüştürülmüş değerler sağda ardarda dizilerek B$'ye aktarılıyor.

programın dökümleri, bunları kullanan bir deneme ana programı ile birlikte 657. ve 658. sayfalarda bulunmaktadır. 2000 alt-programının anlaşılmasını kolaylaştırma açısından, sayfa 659'da verileri dönüştürmede kullanılan mantığın şeması gösterilmiştir.

BİR VERİ KAYDININ OKUNMASI VE SAYISAL DEĞERLERİN ÇIKIŞI



BİR VERİ KAYDININ HAZIRLANMASI, YAZILMASI VE OKUNMASI

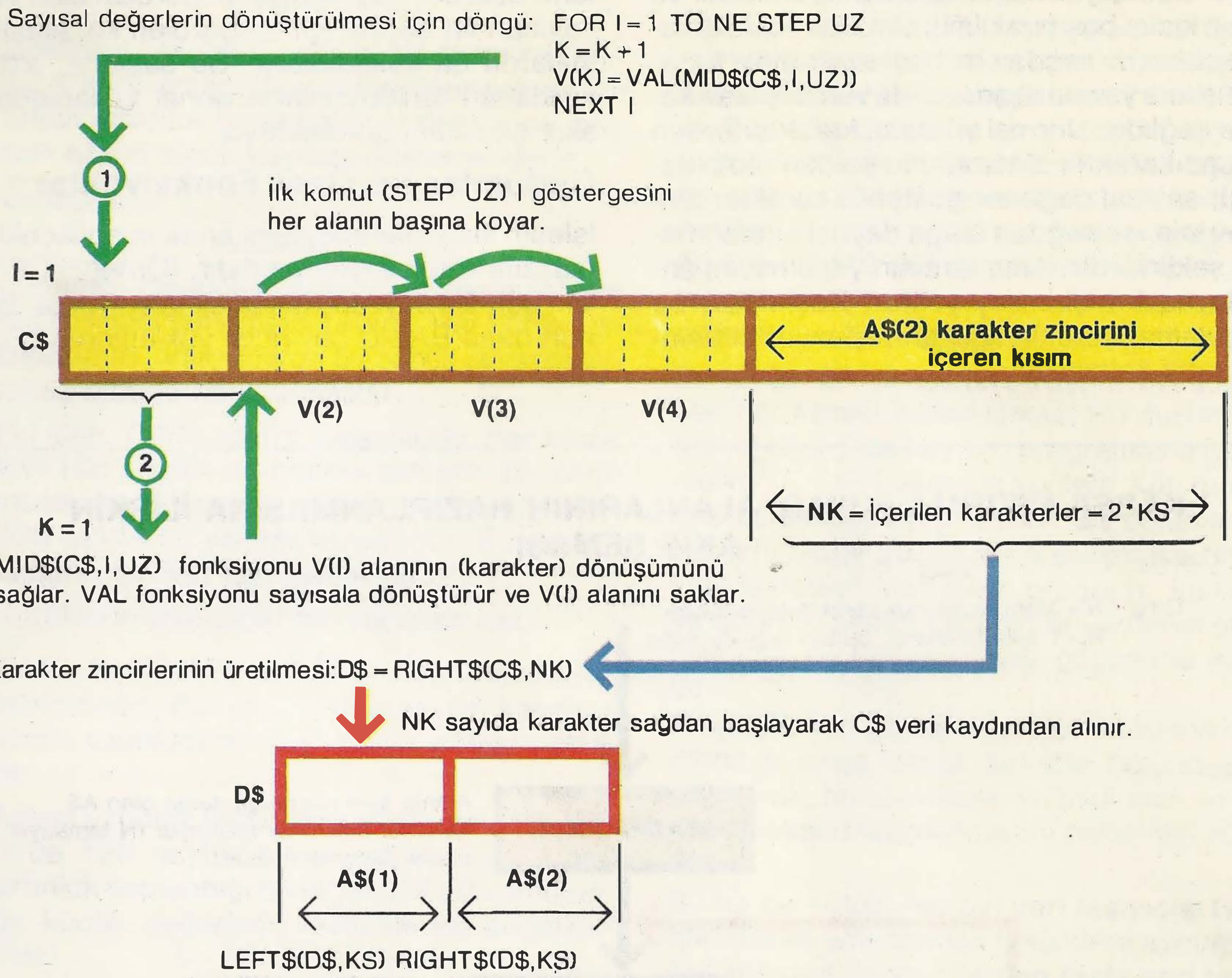
```
10 '
20 '
30 '        KUTUK SCRREC
35 DEFINT I-N
40 ' ** RUTINE GIRISLER **
50 '     NR= YAZILMIS OLAN VERI KAYDININ NUMARASI
60 '     V(4)= VERI KAYDINA YAZILACAK OLAN SAYISAL DEGERLER
70 '     UZ= SAYISAL DEGERLERIN RAKAM SAYISI OLARAK AZAMI UZUNLUGU
80 '     A$(2)=SAYISAL DEGERLERLE BIRLIKTE VERI KAYDINA YAZILACAK OLAN
90 '           KARAKTER ZINCIRLERI
100 '    UK= KARAKTER ZINCIRLERININ KARAKTER OLARAK AZAMI UZUNLUGU
110 DIM B$(4),A$(2)
112 OPEN "R",1,"DENEME",50
114 FIELD #1,50 AS BUF$
120 UK=10
130 UZ=5
140 INPUT "YAZILACAK VERI KAYDININ NUMARAŚI";NR
150 FOR I=1 TO 4
160 PRINT "SAYISAL VERI KAYDI";I
170 INPUT "DEGER (AZAMI 4 RAKAM)";V(I)
180 NEXT I
190 INPUT "BIRINCI KARAKTER ZINCIRI:";D$
200 INPUT "IKINCI KARAKTER ZINCIRI:";E$
202 A$(1)=SPACE$(UK):A$(2)=SPACE$(UK)
204 LSET A$(1)=D$
206 LSET A$(2)=E$
210 '
220 GOSUB 1000
230 '
240 A$(1)="":A$(2)=""
250 FOR I=1 TO 5:V(I)=0:NEXT I
```

```
260 '
270 GOSUB 2000
280 '
290 LPRINT "OKUNAN DEGERLER:"
300 LPRINT "SAYISAL:"
310 FOR I=1 TO 4
320 LPRINT V(I)
330 NEXT I
340 LPRINT
350 LPRINT "KARAKTER ZINCIRLERI:"
360 LPRINT A$(1)
370 LPRINT A$(2)
380 '
390 END
1000 ' ** YAZMA ALT-PROGRAMI **
1010 IF NR=0 THEN PRINT "VERI KAYDININ NUMARASINDA HATA":RETURN
1020 FOR I=1 TO 4
1030 B$(I)=SPACE$(UZ)
1040 NEXT I
1050 FOR I=1 TO 4
1060 C$=STR$(V(I))      ' BIR KARAKTER ZINCIRINE DONUSTURME
1070 IF LEN (C$)>UZ THEN PRINT "UZUNLUK HATASI":RETURN
1080 RSET B$(I)=C$      ' ARDARDA SIRALAMA
1090 NEXT I
1100 C1$=A$(1)          ' BIRINCI KARAKTER ZINCIRININ SAKLANMASI
1110 IF LEN(A$(1))>UK THEN C1$=LEFT (A$(1),UK)
1120 C2$=A$(2)          ' IKINCI KARAKTER ZINCIRININ SAKLANMASI
1130 IF LEN(A$(2))>UK THEN C2$=LEFT(A$(2),UK)
1140 ' ** C1$ VE C2$, A1$ VE A2$ ILE AYNI VERILERI ICERIRLER
1150 ' FAKAT ONGORULEN AZAMI UZUNLUGA GORE KISALTILMISLARDIR
1160 ' **** TOPLAM KARAKTER ZINCIRININ OLUSTURULMASI
1170 C$=""
1180 FOR I=1 TO 4
1190 C$=C$+B$(I)      'C$ DE B$(I) KARAKTER ZINCIRLERI TOPLANIYOR
1200 NEXT I
1210 C$=C$+C1$+C2$    ' SON IKI KARAKTER ZINCIRI EKLENIR
1212 PRINT "VERILER DISKETE SU FORMDA AKTARILIR:"
1214 PRINT C$
1220 BUF$=C$    ' DISKET YONUNDEKI ARA BELLEK BUF$ DIR
1230 PUT #1,NR
1240 RETURN
2000 ' *** VERILERIN OKUNMASI ICIN ALT-PROGRAM
2010 GET #1,NR
2020 C$=BUF$        ' C$ VERILERI OKUNDUGU GIBI
2025 '              ALAN BIR ARA BELLEKTIR
2030 PRINT C$
2040 NE=4*UZ    ' SAYISAL VERILER TARAFINDAN DOLDURULAN KISIM
2050 ' BAYT CINSINDEN UZUNLUGU= UZ UZUNLUGUNDA 4 ALAN
2060 K=0      'V(4) SAYISAL DIZEYLERIN SAYACI
2070 FOR I=1 TO NE STEP UZ      ' V(4) DIZEYI ICIN
2080 '        HER SAYACIN BASLANGICI
2085 K=K+1
2090 V(K)=VAL(MID$(C$,I,UZ))
2100 NEXT I
2110 NK=2*UK        ' NK'=IKI A$(2) KARAKTER ZINCIRININ ORTAKLASA KAPLADIGI YER
2120 D$=RIGHT$(C$,NK)    'D$ VERI KAYDININ SON KISMINI YANI HER IKI
2130 '       KARAKTER ZINCIRINE KARSILIK GELEN KARAKTERLERI ICERIR
2140 A$(1)=LEFT$(D$,UK)
2150 A$(2)=RIGHT$(D$,UK)
2160 RETURN
```

```
OKUNAN DEGERLER:
SAYISAL:
 1111
 2222
 3333
 4444

KARAKTER ZINCIRLERI:
AAAAAAAA
BBBBBBBB
```

VERİ DÖNÜŞTÜRMENİN MANTIK ŞEMALARI



Veri seçimi

Kütükler işletilirken, bir kütükte içerilen bütün veri kayıtlarının işlem görmesi pek seyrek rastlanan bir durumdur. Güncelleştirmeler veya diğer işlemlerin hangi veri kayıtları üzerinde yapılacağı, bu kayıtların yalnızca belli alanlarındaki değerler tarafından belirlenir. Bu alanlar, veri kayıtlarına erişmede anahtar görevi görürler. Bu yüzden eğer sistem elveriyorsa göstergeli bir kütük yapısının kullanılması anlamlı olur. Böylece göstergenin tüm işletimi sistem tarafından gerçekleştirilir ve kullanıcı, veri kaydının seçimi için hangi anahtarın kullanılması gerektiğini kesin olarak saptayan birkaç komutu program üzerinden yürütür.

Ancak kütüklerin anahtar işlevi gören alanlardan yararlanarak işletimini mümkün kılan temel program, her araçta bulunmaz. Bu durumlarda bir kıyaslama alt-programının yazılması gerekmektedir. Böyle bir alt-program özel sorunlar yaratmaz. Yalnızca mantıksal işlemcilerin doğru olarak kullanılması ve tüm alanların aynı şekilde sıralanması gerekir. Genellikle diskete yazılma sırasında, alanlar karakterler tarafından tamamen doldurulmamış durumdadır. Bu nedenle sık sık her alanda birkaç boş yer kalır. Yeniden okuma aşamasında bu boş yerler de belleğe yüklenir. Bu şekilde elde edilen bir alanla, klavye aracılığıyla girilen bir alan kıyaslandığında sıralamada düzensizlikler saptanabilir ve bu durumda, aynı değerleri içermelerine karşın her iki alan farklı gözükebilir.

Örneğin bir şehir isminin girişi için diskette bulunan bir kütükte 10 karakter uzunluğunda bir alan öngörüldüğünü varsayalım. Giriş aşamasında girilen veri kaydı KARS olsun. Bu durumda alan, veri kaydının dört karakterinin yanısıra altı boş yer daha içerir.

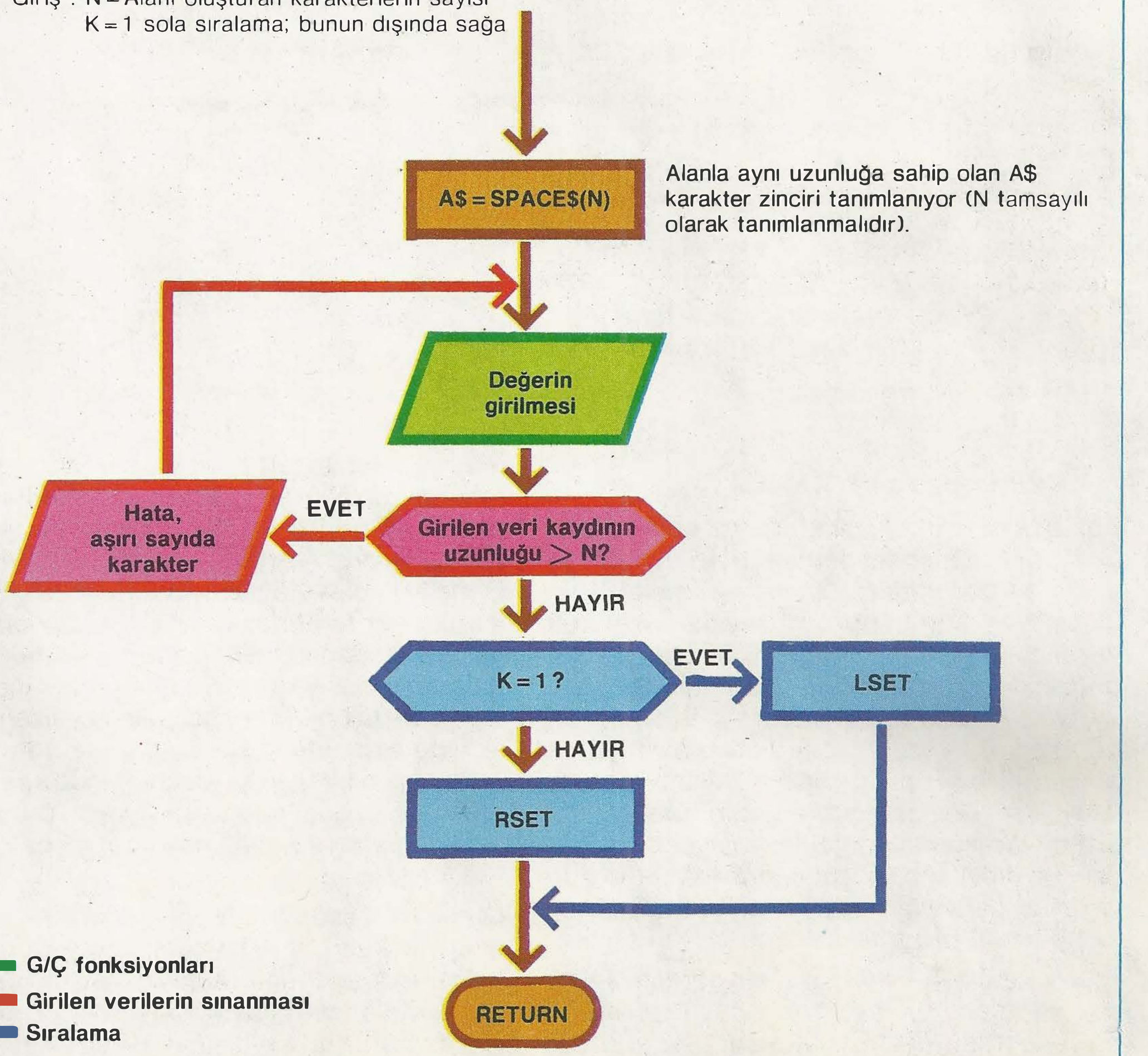
KARS veri kaydını içeren veri kayıtlarının aranabilmesi için, aynı özellikleri gösteren bir kıyaslama karakter zinciri hazırlanmalıdır. Veri kaydı sola dayalı olarak sıralanmış olmalıdır ve artan kısım boş bırakılmış olmalıdır. Sıralamaya soldan mı sağdan mı başlanacağının kararı, diskete yazma aşamasında verilmiş olan karara bağlıdır. Normal yöntem, karakterlerden oluşan karakter zincirlerinin soldan (sola dayalı), sayısal değerler gösteren karakter zincirlerinin ise sağdan (sağa dayalı) sıralanmaları şeklindedir. Ama tersinin yapılmasını engelleyecek hiçbir şey yoktur. Kuşkusuz, bu uygulama, kütükle ilgili tüm işlemler sırasında sabit kalmalıdır.

Tabii deminki örnekte, girilecek karakter zinciri daha baştan öngörülen alanın uzunluğuna eşleştirilerek, her türlü olası belirsizlik bertaraf edilebilir. Bu şekilde doldurulan alan veri kayıtlarının seçimi için öngörülen karşılaştırmalarda da kullanılabilir. Bu sayfada, altta, açıklanan fonksiyonların genel kullanımının akış şemasını görmekteyiz.

Güdümler ve Özel Fonksiyonlar

İşletim sistemlerinin, aynı anda açabilecekleri azami kütük sayısı sınırlıdır. (Örneğin CP/M için üç). Bu sayı değiştirilmek isteniyorsa, bunun daha BASIC çeviricisi yüklenirken bildirilmesi gerekir.

KARŞILAŞTIRMA (KIYAS) ALANLARININ HAZIRLANMASINA İLİŞKİN AKIŞ ŞEMASI



Normal olarak bu durumda, çeviricinin yüklenmesi için gereken komut yazı düzeninin şu sözdizimini alması gerekir.

MBASIC/F:N

/ simgesi, arkadan gelenin bir işlem olduğunu; F harfi, standart sınırlarını aşan sayıda kütüğün açılması gerektiğini; N değeri ise istenen azami kütük sayısını göstermektedir.

Örneğin,

MBASIC/F:5

güdümü, çeviriciye aynı anda beş kütüğün kullanılması gerektiğini bildirir. Her kütük için o kütüğün tarifini içeren bir bellek alanının ayrılmış olduğu hatırlanmalıdır.

Bu alan, CP/M işletim sisteminde, her kütük için 166 baytlık bir hacme sahiptir. Bir önce incelenen durumda, toplam olarak 5*166= 830 baytlık bir yer sözkonusudur. Bu yer kütüklerin tarifleri için ayrılmıştır.(*)

Kütüklerle ilgili diğer bir seçenek ise,

/S:N

şeklindedir. Burada, N değeri veri kaydının azami uzunluğunu bayt olarak göstermektedir.

Azami uzunluk tam olarak gösterilmediği takdirde 128 baytlık bir uzunluk alınır (azami uzunluk saptandığı takdirde, FIELD komutunda küçük değerlerin kullanılması engellenmez).

Bu arada, CP/M işletim sisteminde kullanılabilecek mantıksal veri kayıtlarının azami sayısının 32.767 olduğunu hatırlayalım. Bunun tek istisnası S seçeneğini kullanırken, veri kaydının uzunluğunu 256 bayt olarak tanımlamaktır, bu seçenek kullanıldığında azami adresleme 8 MB olabilir.

Bazı BASIC çeviricilerinde kütük uzunluklarının (kesim olarak) okunması için özel bir fonksiyon öngörülmüştür. BASIC 80'de, bu fonksiyon

LOF(N)

şeklindedir.

Bu fonksiyon, veri kayıtlarının kütüğün son genişletilmesinden sonraki sayısını N ile göstermektedir. Eğer kütük birden fazla genişlemeye sahip değilse, bu sayı kütüğün gerçek uzunluğu ile aynı olmaktadır.

(*) Kütüklerin tarifleri için kullanılan öbeklerin, normal olarak FDB (FILE DESCRIPTION BLOCK) kısaltması ile adlandırıldığını okurlarımıza hatırlatalım.

Bölümlenmiş Kütükler (Multivolume)

Disketlerin (floppy-disks) bellek kapasiteleri ölçüleri ve türlerine göre yaklaşık olarak 160.000 ile 1.000.000 bayt arasındadır. Bazı uygulamalarda, disket tarafından sunulandan daha büyük bir kapasiteye ihtiyaç olabilir. Böyle bir durumda en iyi çözüm, disk gibi uygun araçların kullanılmasıdır. Bazı aygıtlarda, diskler 20 MB'den daha büyük bellek kapasitesi sunabilir. Ama kullanılan aygıtların çoğu 5 ila 10 MB'lik kapasitelere sahiptir. Diskler giriş/çıkış fonksiyonlarının daha hızlı icra edilmelerini de sağlarlar.(**)

Bu çözüm teknik açıdan en uygun olanıdır, ancak her zaman kullanılamaz. Bu durumlarda aracın eksiklikleri uygun programlarla giderilmelidir. En sık kullanılan yöntem, kütüğün birçok bölümlere ayrılmasıdır. Bu bölümlerden her biri, hiçbir güçlükle karşılaşmaksızın diskete alınabilir. Veri işlem programı, kullanıcıya doğru disketin seçiminde yardımcı olabilmek için, bu bölümlenmeyi gözönüne almalıdır.

Program, veri işleme aşamasında istenilen verilerin o anda hangi diskette bulunduğunu saptamak, bunu kullanıcıya bildirmek ve istenilen disketin hazırlanmasını beklemek zorundadır.

Bu tür bir kütük, her biri **veri taşıyıcısı (volume)** olarak adlandırılan bölümlere ayrılabilme özelliğinden dolayı "**birden fazla veri taşıyıcı üzerinde bulunan kütük**" -ya da kısaca "**bölümlenmiş kütük**" adını alır. Kişisel bilgisayarlarda karşılaşılan kısıtlı bellek kapasitelerinin sakıncalarını gidermeye yarayan bu kütük bölümlenmesi, mantıksal bir seçim temeline dayalı olarak da gerçekleştirilebilir.

Zaten verilerin tek bir kütükte saklanması her zaman tavsiye edilemez. Bazen disket tüm kütüğü alabiliyor olsa bile, kütüğün birkaç veri taşıyıcısına bölünmesi avantaj sağlayabilir.

Bu durumlarda, mantıksal bir bölümleme ele alınır: Birçok veri taşıyıcısına bölünme sözkonusu olmakla birlikte fiziksel olarak bütün veri kayıtları aynı diskette bulunurlar. Kütük bu şekilde düzenlendiğinde onu yöneten program, disketlerin düzenlenmesi için gerekecek olan bekleme sürelerini sağlamak zorunluluğundan kurtulur.

(**) Disklerle veri değişimi DBE (doğrudan bellek erişimi) ile yapılır ve bu nedenle de son derece yüksek bir hız ile meydana gelir; disketlerde ise bu teknik normal olarak kullanılamaz.

Bölümlenmiş kütüklerin (birden fazla **mantıksal** veya **fiziksel** veri taşıyıcısıyla) yönetimi, her veri taşıyıcısı için üzerinde hangi verilerin bulunduğunun kaydedilmesi ile gerçekleşir. Böylece veri kaydının değerinden hareketle, veri taşıyıcının numarası (veya ismi) elde edilebilir.

Bu şekilde yapılandırılmış bir işletim programı, dizinli kütüklerle çalışıldığında çok karmaşık hale gelebilir. Buna karşılık doğrudan erişimli kütüklerde durum göreceli olarak daha basittir. Doğrudan erişimli bir kütüğün işletimi için en aracısız yol, veri kaydının numarasının verilere erişmek için anahtar olarak kullanılmasıdır. İlk girilen veri kaydı 1. veri kaydı numarasını ve 1. erişim anahtarını içerir, ikincisi 2'yi alır ve böylece devam eder. Ancak unutulmamalıdır ki, bir veri kaydı numarasından, erişim anahtarı olarak sözetmek doğru bir terminoloji değildir. Veri kayıtlarının anahtarı, normal olarak verileri karmaşık işlemlere tabi tutarak elde edilir. Birden fazla veri taşıyıcısı üzerinde bulunan bir kütüğün yönetimi, içerilen veri kayıtlarının sınır değerlerinin ve veri taşıyıcılarının (yani disketlerin) isimlerinin özel bir kütükte saklanmasından ibarettir. Kütüğün genişleyebileceğini gözönünde bulundurarak belirli bir bellek alanını hazır tutan bu kütük, bir cins kılavuz oluşturur ve normal olarak sürekli güncelleştirilmesi gereken az sayıda veri kaydından (her disket için bir adet) meydana gelir.

Büyük Sistemlerde Kütük Yönetimi

Şimdi küçük bilgisayarları bırakıp büyük sistemleri inceleyelim. Büyük bilgisayarlarda, programcıyı ayrıntılı işlerden büyük ölçüde kurtaran ve veri bankalarının işletimi için genel uygulamaları içeren programlar vardır. Bu programlar, örneğin, verilerin yapısını tanımlayan ve sıralama alt-programlarını içeren gösterge dizilerini otomatik olarak yönetirler.

İşletim programları iki kategoriye ayrılabilir: bağımsız olanlar ve bir programlama diline bağımlı olanlar.

Bağımsız veri işleme programları gerçek bir yüksek düzeyli bir dil oluştururlar. Bunu, kullanılan komutların program dillerindeki tipik olan komutlara benzemesinde de görebiliriz. Ancak diğer dillerdekilere ek olarak kütük yönetimi ile ilgili özel başka komutlar da vardır. Bir programlama diline bağımlı olan programlar, bir öncekinin olanaklarını bütünleyen bir dizi komut içerirler. Büyük sistemlerde en önemli programlama dilleri COBOL ve FORTRAN'dır. Kütük yönetimi için gerekli olan komutlar öncelikle programlama dilinin öngördüğü şekle sokulur ve daha sonra yerine getirilir. Aşağıda bu programlarda öngörülen en önemli fonksiyonlar görülmektedir:

STORE	Bir veri kaydı yazılır ve diğer verilere yönelik tüm göstergeler üretilir.
FIND	Kullanıcı tarafından verilen bir parametreye göre bir veri grubu aranır.
MODIFY	Bir veya daha fazla veri değiştirilir.
ERASE	Bir veri kaydı silinir ve daha sonra ona ilişkin gösterge dizileri değiştirilir.

Her birinin sözdizimi, aygıtın türüne göre değişen bu en önemli komutların dışında, normal olarak BASIC'deki anlamlara sahip olan IF gibi ekler, mantıksal işlemciler (AND, OR vb.) ve aritmetik işlemciler (+, —, /, *) öngörülmüştür. Veri bankalarının işletimine yönelik olarak düzenlenmiş dillerin çoğu, programlamada henüz tecrübesiz olan kişilerin bile veri aramasını mümkün kılacak ''alt sistemler'' içerir. Verilecek komutlar çok basittir ve özel bir anlamı olan ekler kullanırlar. Örneğin, MDQS sistemi (Management Data Query System) kullanıldığında, bir stok mevcudunun sorulması için aşağıdaki komutlar kullanılır.

```
DISPLAY DEPO TÜRÜ MİKTAR
IF KODE = ''A''
LET MİKTAR = GİRDİ—ÇIKTI
END
```

Bu tür kütük yönetim programları büro işlemlerinin otomasyonu için çok önemli olduğundan ilerde ayrıntılı olarak anlatılacaktır.

Stok Hesabı

Stok hesabı bilgisayarlar açısından tipik bir kullanım alanıdır, çünkü yerine getirilmesi gereken fonksiyonlar sürekli olarak tekrarlanır ve büyük miktarda verileri kapsarlar.

Stok hesabı için bir programın oluşturulması, birçok zorluğu beraberinde getirebilir. Bunun sebebi, verilerin farklı yöntemlerle biraraya getirilebilmesi ve ekonomik veya mali türden özel fonksiyonların yerine getirilmesi zorunluluğudur. Belli bir karmaşıklık düzeyindeki uygulamalarda özel bir programın yazılması bir yarar sağlamaz. Bunun yerine, mevcut birçok yöntemden biri kullanılmalı ve bu yöntem gerektiği ölçüde ihtiyaca göre düzenlenmelidir. Bu konunun karmaşıklığı nedeniyle, bundan

sonra stok hesabıyla ilgili olarak ortaya çıkan sorunların yalnızca önemlileri gösterilecektir.

Stok hesabıyla ilgili olarak çözülmesi gereken ana sorunlar aşağıdaki şekilde sıralanabilir:

— (Yeniden değerlendirme) Stokların ve envanterin düzeltilmesi
— Temel alınan mal listesi

TEST 19

1 / Aşağıdaki önermelerden hangileri doğrudur, hangileri yanlıştır?

Sıralı bir kütük:
a) Genişletilemez.
b) Yüksek erişme hızlarına izin verir.
c) Doğrudan erişimli kütüğe göre daha az yer kaplar.
Doğrudan erişimli bir kütük
d) Genişletmelere izin vermez
e) Okuma ve yazma için aynen kullanılabilir
f) EOF (N) fonksiyonunu kullanır

2 / A disket biriminde DENEME kütüğü bulunmaması durumunda aşağıdaki komutlar verildiğinde ne olur?

a) OPEN ''I'',1,''A:DENEME''
b) OPEN ''R'',1,''A:DENEME'',120

3 / PUT ve GET komutunun yerine getirilmesi sırasında son kullanılan veri kaydının numarası hangi komutla saptanabilir?

4 / Aşağıdaki verilerin bir kütüğe girilmesi için bir alt-program yazınız:

C% = 1 ile 99 arasında bir tamsayı (kod)
D$ = Açıklama, 20 karakter
QT% = Miktar, tamsayı
C = Masraf (maliyet)

Kütüğün ismi DENEME'dir ve A birimindeki diskette bulunmaktadır. Alt-program, 1000 satır sayısı ile başlamalıdır.

5 / Yukarıda oluşturulan alt-programları aşağıdaki şartları sağlayacak komutlarla tamamlayınız:

— Önceden girilmiş olan herhangi bir veri kaydının tekrar okunması.
— Önceden varolan QT% miktarına, klavye aracılığıyla girilen yeni bir miktarın eklenmesi.
— Disketteki değişikliklerin (aynı pozisyonlarda) saklanması.

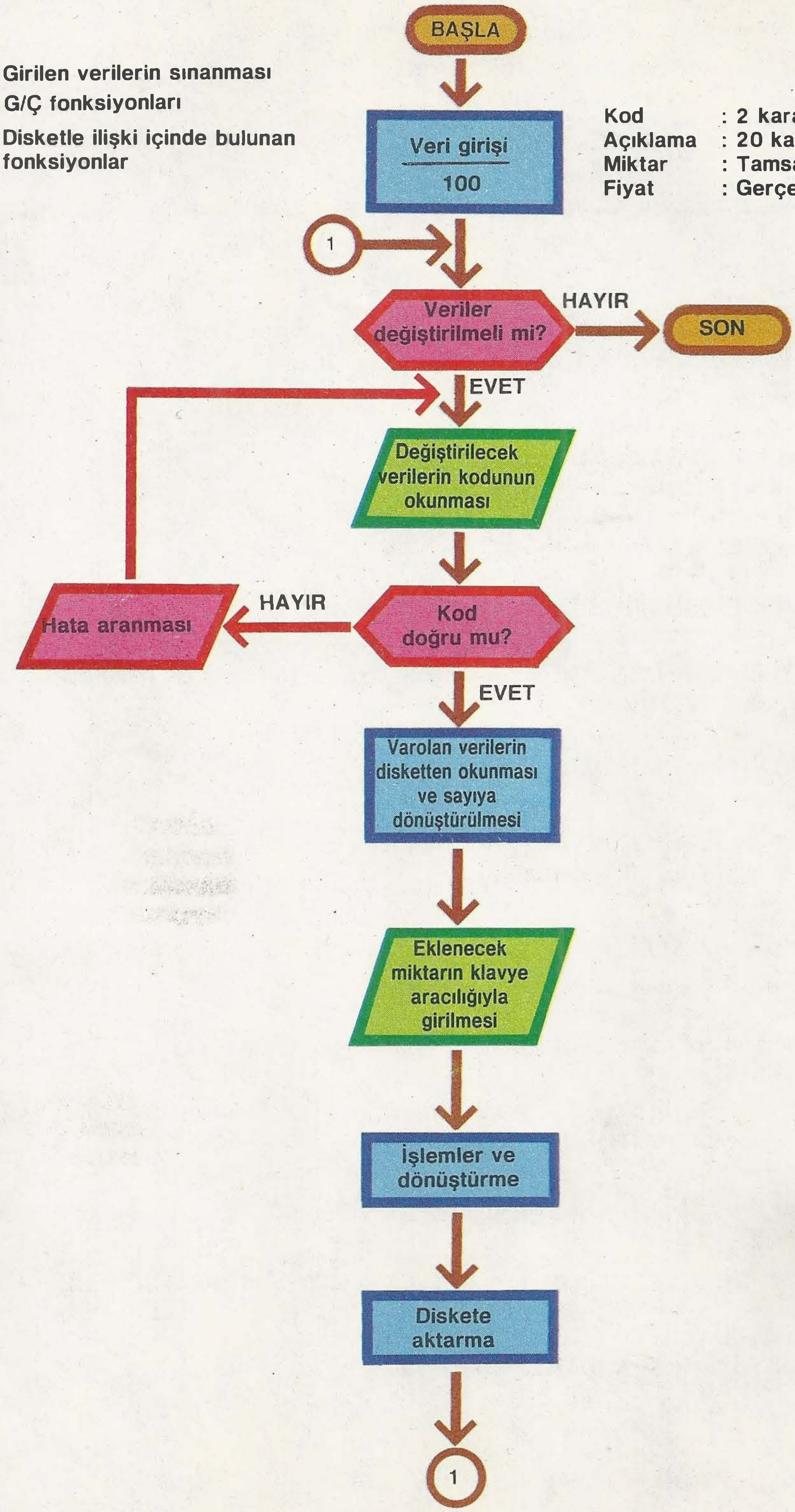
Bu yöntemin akış şeması sayfa 644'te gösterilmiştir.

Testin çözümleri sayfa 671'de bulunmaktadır

Girilen verilerin sınanması

G/Ç fonksiyonları

Disketle ilişki içinde bulunan fonksiyonlar



Kod : 2 karakter (0'dan 99'a)
Açıklama : 20 karakter
Miktar : Tamsayı
Fiyat : Gerçek sayı

— Temsil (Simulate)
— Diğer yöntemlerle ilişkiler

Stokların ve envanterin düzeltilmesi (yeniden değerlendirme). Mal girişi için işlemler, malın giriş tarihine göre birbirini takip eden zamanlarda olabilir. Bu yüzden aynı mal girişi için farklı maliyetler de sözkonusu olabilir. Malın çıkışı anında, giriş tarihinin saklanmış olmasından yararlanıp, giriş tarihindeki maliyetten hareketle fiyatın saptanması için bir yöntem belirlenmiş olmalıdır. Böylece belli bir mal, giriş tarihine göre belli bir maliyet ve dolayısıyla belli bir fiyata sahip olur. Ancak bu yöntem büyük bir bellek kapasitesi gerektirir. Ayrıca deponun belirli vergi normlarına bağlı olduğu bazı durumlarda, bu değerin düzenlenmesindeki mantık keyfi olamayacaktır.

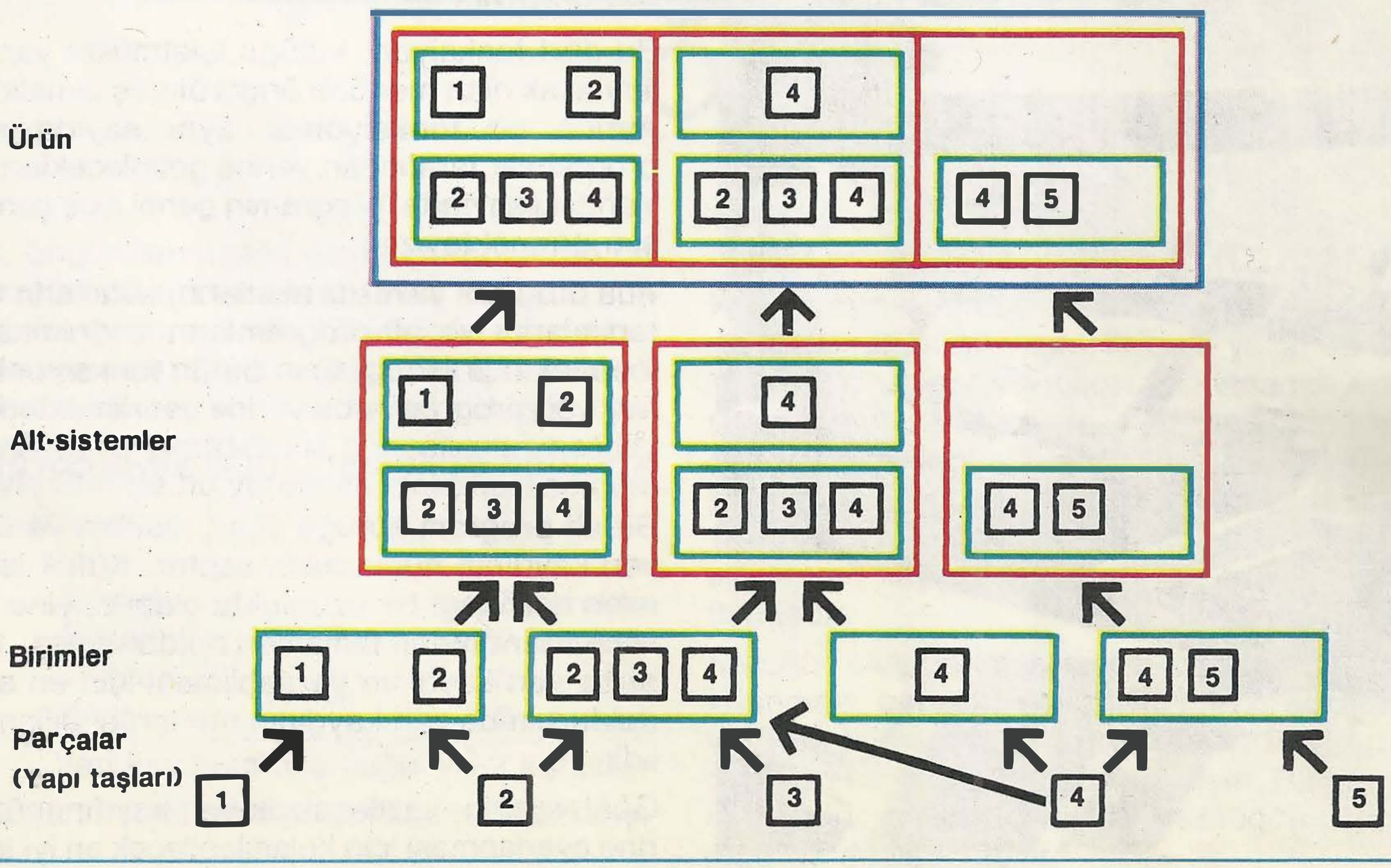
Temel mallar listesi. Endüstride depo durumu, **temel mallar listesi** -yani ürünün imalatında gerekli olan parçaların listesi- ile yakından ilişkilidir. Genelde bir ürün, farklı alt-sistemlerden, bunlar da belli sayıdaki birimlerden, birimler ise belli sayıdaki parçalardan (yapı taşlarından) oluşur. Bu yöntemle, her ürün için alt kademelerde gerekli olan elemanların sayısı ve türünün gösterildiği bir sıralama meydana getirilir. Ters yönde ise, bu sıralama yardımı ile bu parçaların gerekli olduğu üst kademeler ortaya çıkarılabilmelidir. Kolayca tahmin edileceği gibi, bu şekilde üretilen farklı bağımlılık ilişkileri, bu programların yazılmasında büyük zorluklar yaratırlar. Bu sayfada, altta, dört kademeli bir yapı gösterilmiştir. Renkler takip edildiğinde, yapının anlamlı kalması gerekiyorsa, düzenlenmesi gereken göstergenin karmaşıklığı hakkında yaklaşık bir fikir elde edilebilir.

Temsil (Simulation). Temel mallar listesi elde bulunduğu takdirde, fiyat değişikliklerinin veya bir veya daha fazla parça için stok eksikliğinin etkilerini temsil eden özel işlemler yapılabilir. Bilgisayar, bir bağımlılığın mantıksal bağlantılarını takip ederek, farklı sebeplerin hangi ürünler üzerinde ve hangi ölçülerde etkili olduklarını saptayabilir.

Belli olayların temsili ve bir yapı içerisinde gelişimleri sırasında etkilerinin incelenmesi, belli bir işletme politikasının takibi için fevkalade bir araç oluşturmaktadır. Bu yöntemin bu kadar önem kazanması birçok yazılım şirketini bu amaca yönelik, genel kullanıma ve kişisel bilgisayarlara uygun programlar hazırlamaya yöneltmiştir.

Diğer yöntemlerle ilişki. Bir işletmenin başarısı, işletme ne kadar küçük olursa olsun, farklı

TEMEL MALLAR LİSTESİ İÇİN MANTIK ŞEMASI



işletme alanları arasında kapsamlı veri değişimine dayanır. Bu veri değişimi ne kadar kendiliğinden ve hızlı olursa, işletmenin tümünün yürütülmesi o derece kolay olur.

Stok durumunun, fiyat saptanması, teslim süreleri ve bilançolar üzerindeki etkileri hemen saptanabilmelidir. Bu yüzden farklı bürolar aynı verileri kullanabilmelidir. Büyük işletmelerde merkezî kütükleri içeren bir büyük bilgisayarın kullanılmasıyla sorunun bazı yönleri çözülebilir. Fakat bu durumda da sorun tamamen çözülmüş olmaz, çünkü nihai kullanıcı genelleştirilmiş yönteme kolayca erişmek için gerekli özel eğitimden geçmemiş olabilir ve birçok durumda genel programlarda öngörülmeyen ve yalnızca sözkonusu büroya ilişkin özel veri işleme yöntemleri gerekebilir.

Bu zorluklardan ötürü her büroda bilgilerin elle işlenmesi ve veri değişiminin kağıtlar aracılığıyla yapılması zaman kaybına yol açmaktadır. Bu yavaş bir yöntem gerçekten işletmenin randımanlı çalışmasının önünde ciddi bir engel teşkil etmektedir. Kişisel bilgisayarların devreye sokulması bu problemi rahatlıkla çözebilir. Bu arada çözümün pek az ek masrafa yol açtığı ve büro işlerine rahatlıkla uygulanabildiği de unutulmamalıdır.

Bir veri işleme merkezinde manyetik disk birimleri



Nihai kullanıcı olarak bu aygıtların kulalnılması fazla miktarda özel bilgi gerektirmez ve gerekli bilgiler de yalnızca birkaç gün içersinde öğrenilebilir. Ayrıca farklı birimler, uygun bir bağlantı ağıyla kendi aralarında veya büyük bilgisayarla diyalog içine sokulabilir. Böylece bilgiler gerçek zamanda, yani ortaya çıktıkları anda değiş tokuş edilebilirler. Bu araçların programlayıcılarının doğal gelişimi, gelecekte bir sistem uzmanı yaratacaktır. Bu kimseler, aracın yeteneklerini ve farklı kullanım paketlerinin olanaklarını bildiklerinden, kullanıcılara kararlarında ve araçlarının düzenlenmesinde teorik ve pratik düzeylerde destek olacaklardır.

Kullanım Örneği: Bir Müşteri Kartoteksinin Kurulması

Daha önce, kütük yönetiminde yararlanılan komutları bir kez incelemek için, bir müşteri kartoteksi programında kullanılan kütük yönetimini ele alacağız. Bu konu, programın farklı uygulamalarda da kullanılabilmesi için tamamen genel kapsamlı olarak incelenecektir.

Kullanılacak kütük ve öngörülen uygulama şekli ne olursa olsun, aşağıdaki fonksiyonlar yerine getirilmelidir:

— Yeni kütüğün verilmesi
— Daha önce mevcut olan verilerin değiştirilmesi (güncelleştirme)
— Parametrelerle bir veri kaydını arama
— Bilgiyazım

Bu dört fonksiyon, kütüğü işletmekte yararlanılacak olan menüde öngörülmüş olmalıdır. Ayrıca bu fonksiyonlar aynı sayıda alt-programlar tarafından yerine getirileceklerdir. Yandaki sayfada, programın genel akış şemasını görmekteyiz:

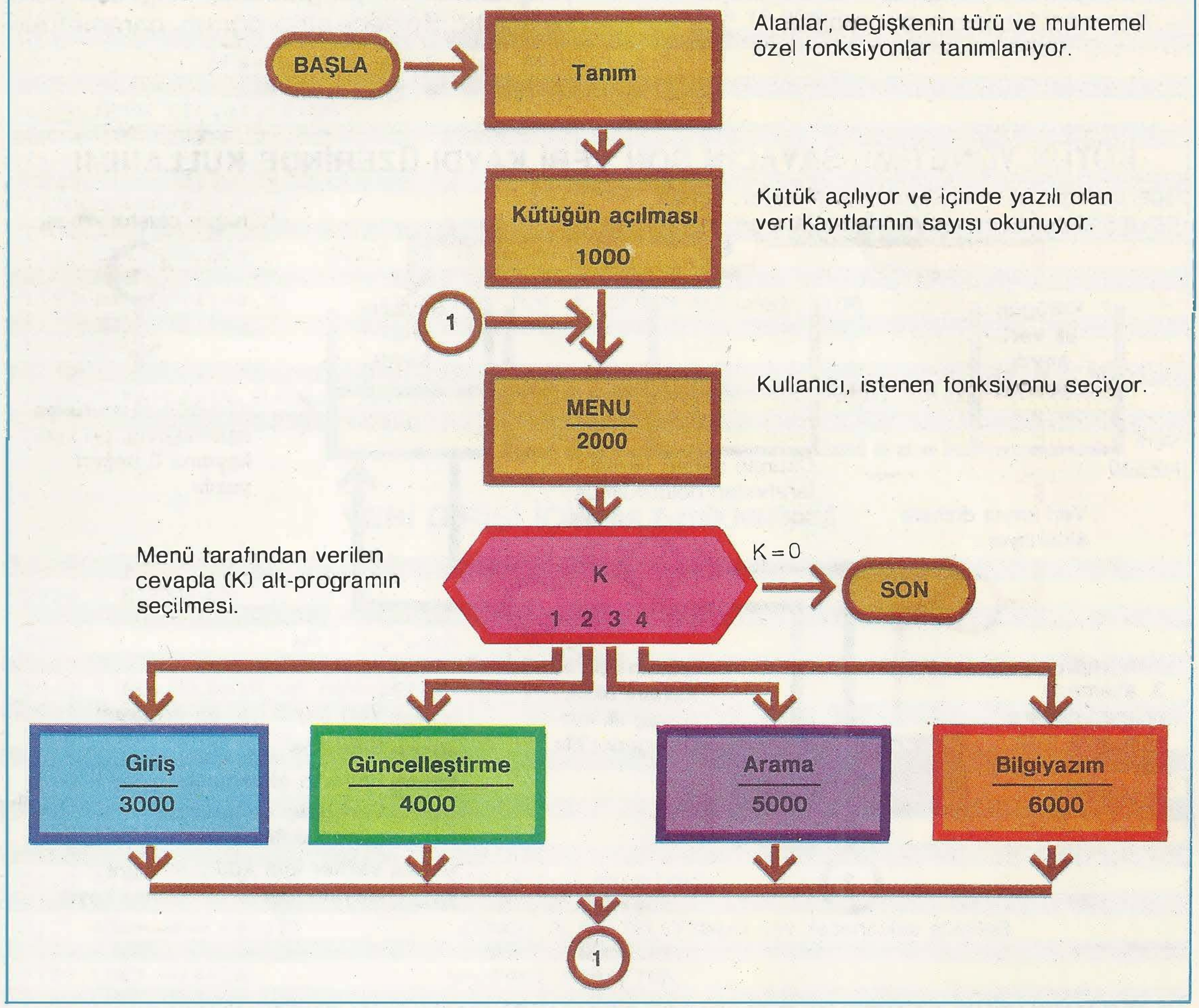
Ana program yalnızca alanların, sabitlerin vb. tanımlarını ve alt-programların çağrılmasını içermektedir. Programın bütün fonksiyonları farklı alt-programlarda yerine getirilmektedir.

Kütüğün Açılması (1000 alt-program)

Bu alt-program kütüğü açar, alanları ve son veri kaydının numarasını saptar. Kütük istenilen herhangi bir uzunlukta olabilir, yine de veriler tarafından tamamen doldurulmaz. Yeni bir veri kaydının yazılabilmesi için en son doldurulmuş veri kaydının numarası bilinmelidir.

Göstergenin, yazılacak ilk veri kaydının üzerine ayarlanması için kulanılabilecek en iyi tek-

DİSKET ÜZERİNDE VERİ İŞLEME, GENEL AKIŞ ŞEMASI



nik, öngörülen azami uzunluğu gösteren bir kütüğün yaratılması ve bu yeni kütüğün son veri kaydının, asıl veri kütüğünün içinde bulunan verilerin sayısını saklamak için kullanılmasıdır. Bu son veri kaydı, veri kütüğüne yeni veri girişi yapıldığında güncelleştirilmelidir. Sayfa 668'de bu yöntemin mantıksal şeması gösterilmiştir.

Soyadı	= 20 karakter
Sokak	= 20 karakter
Telefon	= 20 karakter
Kent	= 20 karakter
Veri kaydının uzunluğu	= 72 karakter

Kütüğün oluşturulması için klavye aracılığıyla,

OPEN ''R'',1,''A:VERILER'',72

komutunun verilmesi yeterlidir. Bunun üzerine sistem, kütüğün ismini dizine yazar ve bir başlangıç uzunluğu ayırır. Saklanacak veri gruplarının azami sayısı 100'e eşit olduğunda, 101. yani sonuncu veri kaydına 0 değeri yazılmalıdır, çünkü başlangıçta kütük henüz veri kaydı içermemektedir ve kullanılmaya hazır ilk veri kaydı 1 numarayı (0 + 1 = 1) taşımaktadır.

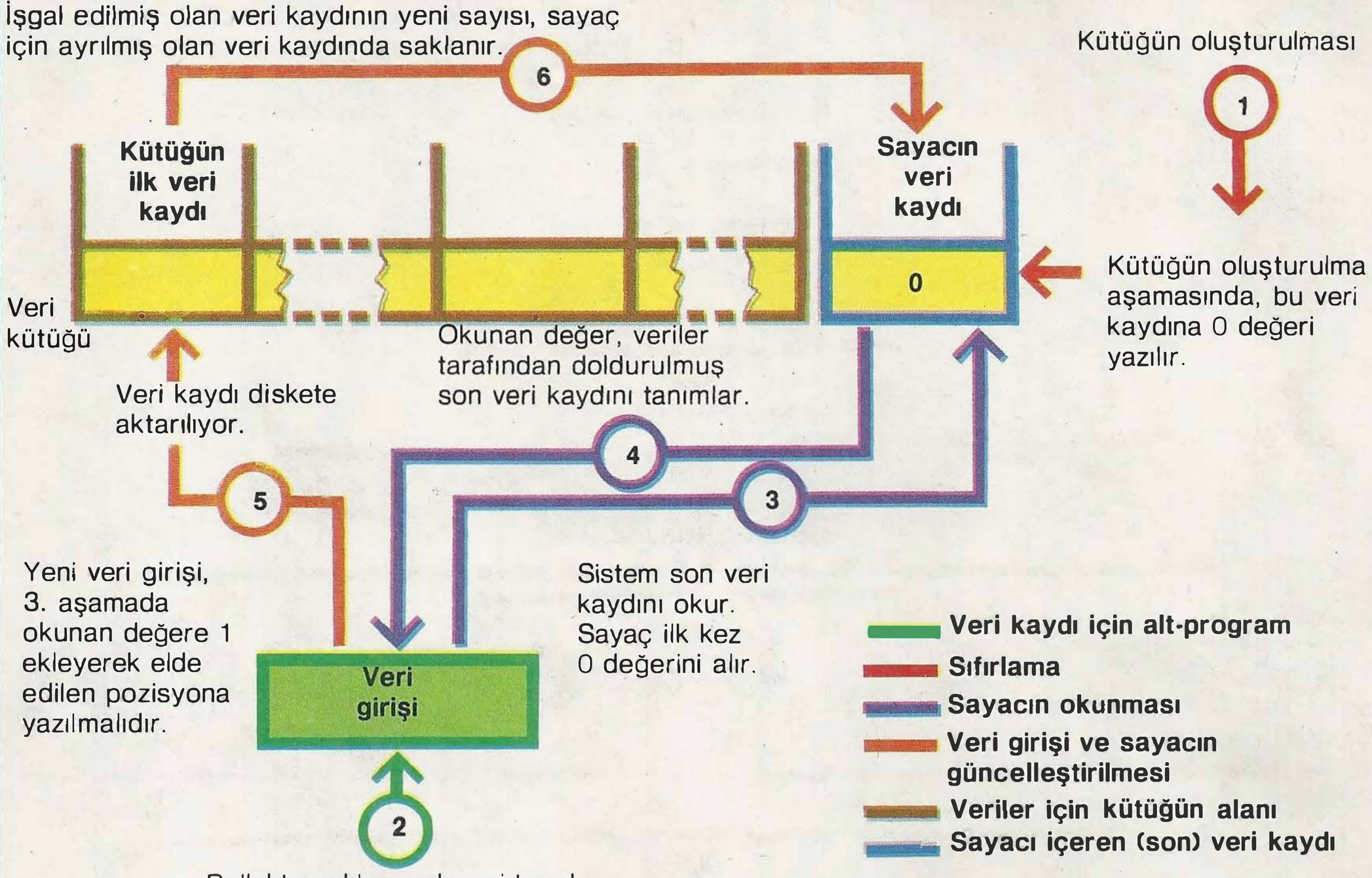
Kütüğün oluşturulması ve son veri kaydının sıfırlanması, genel amaçlı bir **hizmet programı**nın kullanılmasıyla gerçekleştirilir. Böyle bir program sayfa 668'de gösterilmiştir. Bu program birkaç defa işletildiğinde, kütüğün uzunluğu da değiştirilebilir.

Kütüğün oluşturulması ve sıfırlanması süreci tamamlandığında, aşağıdaki fonksiyonları yerine getirmesi gereken 1000 alt-programının yazılmasına geçilebilir:

— Kütüğün açılması
— Son veri kaydının okunması

Bu alt programın yapısı basittir. Buna karşılık gelen dökümü sayfa 669'da görmekteyiz. Alt-programın parametrelenmiş olduğu ve tüm parametrelerin (kütüğün ismi, uzunluğu vb.) ana programdan alt-programa iletildiği son derece açıktır. Böylece alt-program, parametrele-

KÜTÜK YÖNETİMİ. SAYACIN SON VERİ KAYDI ÜZERİNDE KULLANIMI



KÜTÜĞÜN OLUŞTURULMASI VE SON VERİ KAYDININ SIFIRLANMASI

```
10 ' ** KUTUGUN OLUSTURULMASI VE SON VERI KAYDININ SIFIRLANMASI
20 '  ICIN KULLANILABILECEK PROGRAM **
30 '
40 ' KUTUK= CRESCR
50 INPUT "KUTUGUN ISMI";ISIM$
60 INPUT "HANGI DISKET BIRIMINDE (A:/B:)";D$
70 IF D$<>"A:" AND D$<>"B:" GOTO 60
80 IM$=D$+ISIM$
90 INPUT "VERI KAYDININ UZUNLUGU (KARAKTER SAYISI)";U
100 IF U>128 GOTO 90
110 OPEN "R",1,IM$,U
120 FIELD 1,U AS A$
130 INPUT "KAC VERI KAYDI ICERILMELI";NR
140 MX=NR+1
150 B$="0":C$=STRING$(U,B$)     'C$ NIN GOSTERIMI="OOOOO...O"
160 LSET A$=C$   ' TUM O SIMGELERI YAZILIYOR
170 PUT 1,MX
180 PRINT "KUTUK TAMAMLANDI"
190 PRINT "UZUNLUK:";NR
200 PRINT "SAYACIN POZISYONU:";MX
210 END
```

KÜTÜĞÜN AÇILMASI İÇİN ALT-PROGRAM

```
1000 ' ** KUTUGUN ACILMASI ICIN ALT-PROGRAM **
1010 '
1020 ' KUTUK = KUTAC
1030 ' GIRISLER
1040 '         NM$= DISKET BIRMI VE KUTUGUN ADI. ORNEK A:VERILER
1050 '           U= VERI KAYDININ UZUNLUGU (KARAKTER OLARAK)
1060 '          NR= KUTUGUN UZUNLUGU (VERI KAYDI OLARAK)
1070 '         BF$= KUTUGE AYRILMIS  ARA BELLEK
1080 ' ************************************
1090 OPEN "I",ğ1,"B:NM"
1095 CLOSE ğ1
1100 OPEN "R",ğ1,"B:NM"
1110 FIELD ğ1,U AS BF$
1120 MX=NR+1
1130 GET ğ1,MX        ' SAYACI ICEREN VERI KAYDININ OKUNMASI
1140 LSET A$=BF$      'DEGER A$ YE AKTARILIR
1145 PRINT "A$=";A$
1150 B$=LEFT$(A$,3)       'YALNIZCA ILK UC DEGER KULLANILIYOR
1160 MAX=VAL(B$)      'KULLANILABILECEK AZAMI DEGER 999 DUR
1170 '
1180 ' MAX CIKISTA VERI KAYDINDA YAZILMIS OLAN DEGERI YANI SON DOLDURULMUS VERI
1190 ' KAYDININ SUREKLI SAYISINI ICERIR.KULLANILACAK ILK VERI KAYDI
1200 ' BUDEGER +1 DEN OLUSUR
1210 RETURN
```

VERİ GİRİŞİ İÇİN ALT-PROGRAM

```
3000 ' ** GIRIS **
3010 ' GIRISLER
3020 '              MAX = SON DOLDURULMUS  VERI KAYDI
3030 '               NR = KUTUGUN UZUNLUGU
3040 ' VERI GIRISI ICIN ONGORULEN  KISIM ILE ILGILI UYGULAMA
3042 ' BELIRLENIR VE HER DEFASINDA IHTIYACA
3044 ' GORE
3050 ' ** VERILERIN EKRANDAN OKUNMASI **
3060 INPUT "SOYADI";C$
3070 INPUT "SOKAK";V$
3080 INPUT "TELEFON";T$
3090 INPUT "KENT";Y$
3100 '  ** VERIKAYDI DUZENLENIYOR **
3110 A1$= SPACE$(20)          ' BIRINCI ALAN:SOYADI
3120 A2$= SPACE$(20)          ' IKINCI ALAN :SOKAK
3130 A3$= SPACE$(12)          ' UCUNCU ALAN:TELEFON
3140 A4$= SPACE$(20)          ' DORDUNCU ALAN:KENT
3150 LSET A1$=C$              ' SAGDAKI KARAKTER
3160 LSET A2$=V$              ' ZINCIRI SOLDA TOPLANANLARA
3170 LSET A3$=T$              ' AKTARILIYOR
3180 LSET A4$=Y$
3190 ' *** (LSET) KOMUTU SAGDA BULUNAN KARAKTER ZINCIRLERININ
3192 '       FAZLALIK KARAKTERLERINI KISALTIR
3200 A$=A1$+A2$+A3$+A4$
3210 LSET BF$=A$
3220 M=MAX+1
3230 PUT ğ1,M
3240 IF KS=1 THEN RETURN      ' BU KOMUT (4000) ALT-
3250 '                        PROGRAMINDA ACIKLANIYOR
3260 MAX=MAX+1
3270 A$=SPACE$(3)             ' BU SON VERI KAYDINDA ILK UC
3280 '                        GOSTERGEYE AYRILMISTIR
3290 B$=STR$(MAX)
3300 RSET A$=B$
3310 A$=A$+SPACE$(69)         ' VERI KAYDI BOSLUK KARAKTERLERIYLE DOLDURULUYOR
3320 LSET BF$=A$
3330 N=NR+1                   ' N = GOSTERGENIN VERI KAYDI
3340 PUT ğ1, N
3350 RETURN
```

rin değerleri değiştirilerek istenilen her uygulamada kullanılabilir.

1090 komutunun (kütüğün ''I'' şartı altında açılması ve kapatılması) yalnızca kütüğün var olup olmadığının denetlenmesi amacını taşıdığı kesin olarak belirtilmelidir: Kütük ''R'' şartı altında açıldığı takdirde, daha önceden oluşturulmamış ise, komut sıfırlanmamış yeni bir kütük yaratacaktır.

Menü (2000 alt-program)

Bu alt-program için, daha önce genel menülerle ilgili olarak verilmiş bir alt-programa geri dönülebilir. Tek değişiklik gözönüne alınacak metinlerle ilgilidir (1-Giriş, 2-Güncelleştirme vb.)

Giriş (3000 alt-programı)

Burada aşağıdaki fonksiyonlar yerine getirilir:

— Verilerin klavyeden okunması
— Yalnızca ilk boş veri kaydına yazma
— Yeni girilen veri kayıtlarının dikkate alınabilmesi için, son veri kaydının (gösterge) güncelleştirilmesi.

Alt-programın dökümü sayfa 669'da (altta) verilmiştir.

Güncelleştirme (4000 alt-programı)

Aşağıdaki fonksiyonların yerine getirilmesi gerekmektedir:

— Ekrandan, en son haline getirilecek veri kaydının numarasının okunması
— Verilerin disketten okunması
— Mevcut verilerin gösterilmesi
— Yeni değerlerin okunması (şu anda mevcut verilerin değiştirilmesi)
— Değişikliklerin aynı veri kaydında saklanması

Alt-programın dökümünü bu sayfada altta görmekteyiz. Bu alt-programın yazılmasında, giriş alt-programı da kısmen kullanılabilir.

Giriş ve güncelleştirme aşamaları arasındaki fark verilerin yazılmış olduğu pozisyonlara geri dönüştür: Giriş aşamasında her veri girişi yeni bir veri kaydına yerleştirilirken, güncelleştirme aşamasında nihai veri, hangi veri kaydından okunmuşsa yine oraya yazılmaktadır. Verilerin göstergesi yükseltilmemelidir.

Bu özel fonksiyonlar, veri göstergesinin güncelleştirilmemesi gerektiğini 3000 alt-programına gönderen bir çentiğin kullanılmasıyla (dökümde KS) 4000 alt-programından elde edilebilir. Bu özel durumda, 3000 alt-programınca yerine getirilen fonksiyonların değiştirilmesi için bir çentiğin kullanılması özel bir yarar sağlamaz, çünkü gerekli komutlar (3050. satırdan 3230. satıra kadar) doğrudan 4000 alt-programına bir kere daha yazılabilirdi. Bellek alanının büyük ölçüde işgal edilecek olması hiç önemli sayılmamalıdır. Çünkü bunun karşılığında elde edilen şey, son derece açık ve net bir programdır. Çentiğin kullanımı, yalnızca mümkün genel bir yöntemin tasarlanmasında tavsiye edilebilir.

GÜNCELLEŞTİRME ALT-PROGRAMI

```
4000 ' ** GUNCELLESTIRME **
4010 INPUT "GUNCELLESTIRILECEK VERI KAYDININ NUMARASI:";N
4020 IF N>NR GOTO 4010
4030 GET #1,N
4040 LSET A$=BF$
4050 ' ***** ALANLARIN AYRILMASI ****
4060 C$=LEFT$(A$,20)
4070 V$=MID$(A$,21,20)
4080 T$=MID$(A$,40,12)
4090 Y$=MID$(A$,53,20)
4100 PRINT "VERI KAYDINDA ICERILEN VERILER:";N
4110 PRINT C$:PRINT V$:PRINT T$:PRINT Y$
4120 ' *** VERILERDE YAPILMASI ONGURULEBILECEK DEGISIKLIKLERIN GIRISI
4130 ' ICIN (3000) ALT-PROGRAMININ BIR KISMI KS=1 ILE CAGRILARAK
4140 ' KULLANILABILIR. BOYLECE DUZELTMELERIN YENI VERILMIŞ GIBI
4150 ' ALGILANMASI ENGELLENMIS OLUR
4160 MS=MAX          ' (MAX) (MS) DEGISKENINE SAKLANIR
4170 MAX=N-1         ' (3000) ALT-PROGRAMIYLA TEMAS KURABILMEK
4180 '               ' ICIN KULLANILIR
4190 KS=1            ' (3000) ALT-PROGRAMI ICIN CENTIK
4200 GOSUB 3000
4210 MAX=MS          ' (MAX)IN YENI DEGERI YENIDEN GIRILIYOR
4220 RETURN
```

TEST 19'un Çözümleri

1 / Şu kayıtlarla a) doğrudur: Genişletmeler, yalnız yardımcı kütük kullanıldığında mümkündür, doğrudan değil.
b), c), d) yanlış; e) doğru; f) yanlıştır: EOF(N) fonksiyonu sıralı bir kütüğün sonunu göstermek için kullanılır.

2 / a) Bir hata üretir ve programın akışını durdurur.
b) Kütük üretir.

3 / LOC(N) fonksiyonu kullanılmalıdır.

4-5 / 4. ve 5. soruların çözümlerini içeren program dökümü, bu ve bundan sonraki sayfada gösterilmiştir. Programda istenilen fonksiyonları yerine getirebilecek yöntemlerden yalnızca biridir. Bir programın komutlara çevrilme aşaması oldukça özneldir. Programın şekli ve kullanılan komutlar veya fonksiyonlar, bunları yazanın alışkanlıklarına göre çok farklı olabilir.

```
10  ' TEST COZUMU
20  ' KUTUK =TEST 1
100 ' ** ANA PROGRAM **
105 OPTION BASE 1
110 PRINT "ISTENEN FONKSIYONU SECINIZ"
120 PRINT "1=GIRIS 2= OKUMA 3= DEGISIKLIKLER
130 INPUT K
140 IF K<1 OR K>3 THEN GOSUB 2000
150 IF KR<>0 GOTO 110   ' BIR HATANIN ORTAYA CIKMASINDAN
    SONRA YENI SECIM
160 GOSUB 1000
165 MODUS = 0
170 INPUT "DEVAM (EVET/HAYIR)";CEV$
180 IF CEV$="EVET" THEN PRINT CHR$(27)+"+";GOTO 110
190 END
1000 ' ** VERI ISLEME ICIN ALT-PROGRAM ***
1010 ON K GOSUB 1020, 1280, 1440
1020 '
1025 IF MODUS = 1 GOTO 1220
1030 INPUT "KOD:";C%
1040 IF C%<1 OR C%>99 THEN GOSUB 2000
1050 IF KR%<>0 GOTO 1030
1060 '
1070 INPUT "ACIKLAMA";D$
1080 IF LEN(D$)>20 THEN GOSUB 2000    ' UZUNLUK HATASI
1090 IF KR<>0 GOTO 1140
1110 N%=20-LEN(D$)    ' 20 KARAKTERDEN AZ GIRILIRSE, KARAKTER
1120 '          ZINCIRININ TAMAMLANMASI  SAGLANIR
1130 A$=SPACE$(N%)
1140 INPUT "MIKTAR:";QT%
1150 INPUT "MALIYET";C
1160 '
1210 PRINT "VERI GIRISI"
1220 OPEN "R",1,"DENEME",28      ' KUTUGUN ISMI DENEME
1230 FIELD 1,2 AS W1$,2 AS W2$, 4 AS W3$,20 AS W4$
1232 LSET W1$=MKI$(C%)
1234 LSET W2$=MKI$(QT%)
1236 LSET W3$=MKS$(C)
1238 LSET W4$=D$
```

```
1250 PUT 1, C%
1260 CLOSE #1
1270 RETURN
1280 '
1290 PRINT "VERILERIN OKUNMASI"
1300 INPUT "YAZININ KODU";C%
1310 IF C%<1 OR C%>99 THEN GOSUB 2000
1320 IF KR<>0 GOTO 1300
1330 OPEN "R",1, "DENEME",28
1335 FIELD 1,2 AS W1$,2 AS W2$, 4 AS W3$,20 AS W4$
1340 GET 1,C%
1342 C%=CVI(W1$)
1344 OT%=CVI(W2$)
1346 C=CVS(W3$)
1348 D$=W4$
1370 PRINT "** OKUNAN VERILER **"
1380 PRINT "KOD      =";C%
1385 PRINT "ACIKLAMA        =";D$
1390 PRINT "MIKTAR          =";OT%
1400 PRINT "MALIYET         =";C
1410 CLOSE #1
1415 INPUT "VERILERIN OKUNMASINA DEVAM MI";CEV$
1417 IF CEV$="EVET" GOTO 1280
1420 RETURN
1430 '
1440 ' VAROLAN VERILERIN OKUNMASI
1445 GOSUB 1280
1460 '
1470 ' YENI DEGERLERIN OKUNMASI
1495 ' YENI DEGERLERIN YAZILMASI
1500 RETURN
2000 ' ** HATA ALT-PROGRAMI **
2010 '
2020 KR=1      ' HATA CENTIGI
2030 PRINT "**********************"
2040 PRINT "DEVAM EDILECEKSE HERHANGI BIR KARAKTER
     GIRINIZ":S$=INPUT$(1)
2050 PRINT CHR$(27)+"+"
2060 RETURN
```

Arama (5000 alt-programı)

Genelde, arama alt-programlarının yapısı, aranacak verilerle ve bunların miktarları ile yakından ilişkilidir. Kütük büyük ölçülere sahipse, veriler aramada kullanılacak anahtara göre düzenlenmelidir. Daha sonra kodun değiştirilmesiyle durdurulan bir döngünün kullanıldığı bir arama yapılmalıdır (her zaman varolan ön şart: doğrudan verileri gösteren bir gösterge yoktur). Kişisel bilgisayarlardaki uygulamalarda verilerin sayısı hiç önemli değildir. Bu durumlarda bütün kütük, önceden herhangi bir sıralama saptanmaksızın denetlenmeli ve daha sonra, aranılan değerleri oldukları gibi içeren veri kayıtları seçilmelidir. Bu teknikle, aslında gerektiğinden daha fazla zaman harcanır. Çünkü, başka tekniklerle önceden dıştalanabilecek veri kayıtları da okunmaktadır. Fakat zamana ilişkin bu fark, programın yazılması sırasında aşılması gereken zorluklarla telafi edilmektedir.

Bir büyüklük sıralaması elde edilebilmesi için, yüzlerce veri kaydının birkaç dakika içinde okunabileceği varsayılmalıdır, yani gerçekleşebilecek zaman tasarrufu dakikalar ölçüsündedir. Arama alt-programının mantığı ve bununla bağlantılı olarak karmaşıklığı da kullanılmak istenen serbestlik derecesine, yani seçim elemanı olarak kullanılabilecek parametrelerin sayısına bağlıdır. Örneğin, burada sunulan kütükte seçim elemanı olarak yalnızca soyadı kullanılabilir. Bu durumda yalnızca veri kaydının ilk 20 baytı (ilk alan soyadıdır), aranan karakter zinciriyle karşılaştırılmalıdır. Aynı şekilde seçim, kente göre de yapılabilirdi.

Böyle bir durumda son 20 baytın da dikkate alınması gerekir. Böylelikle her bir seçme mantığı için bir alt-program yazılması gerekecektir.

Bu sonuç, arama programının parametrelendirilmesi zorunluğunu açıkça ortaya koyar. İyi bir parametreleme düzeyine ulaşmak için, programın, kullanıcıya veri kaydını oluşturan alanların listesini verecek şekilde yapılandırılması gerekir. Kullanıcı da, aradığı veri kaydını girerek alanlardan birini arama anahtarı olarak seçecektir. 5000 alt-programı bu mantığın uygulanmasına örnektir. Bu durumda (bkz. alttaki akış şeması) alt-program, kullanıcıdan, öngörülen alanlarda hangi değeri aramakta olduğunu sorar. O da, aranan değeri girerek kendisine önerilen belli bir alanı kullanabilir. Veya * simgesini verip, alanın önemli olmadığını, aramanın veri kaydının bütün alanlarında birden yapılmasını istediğini bildilebilir.

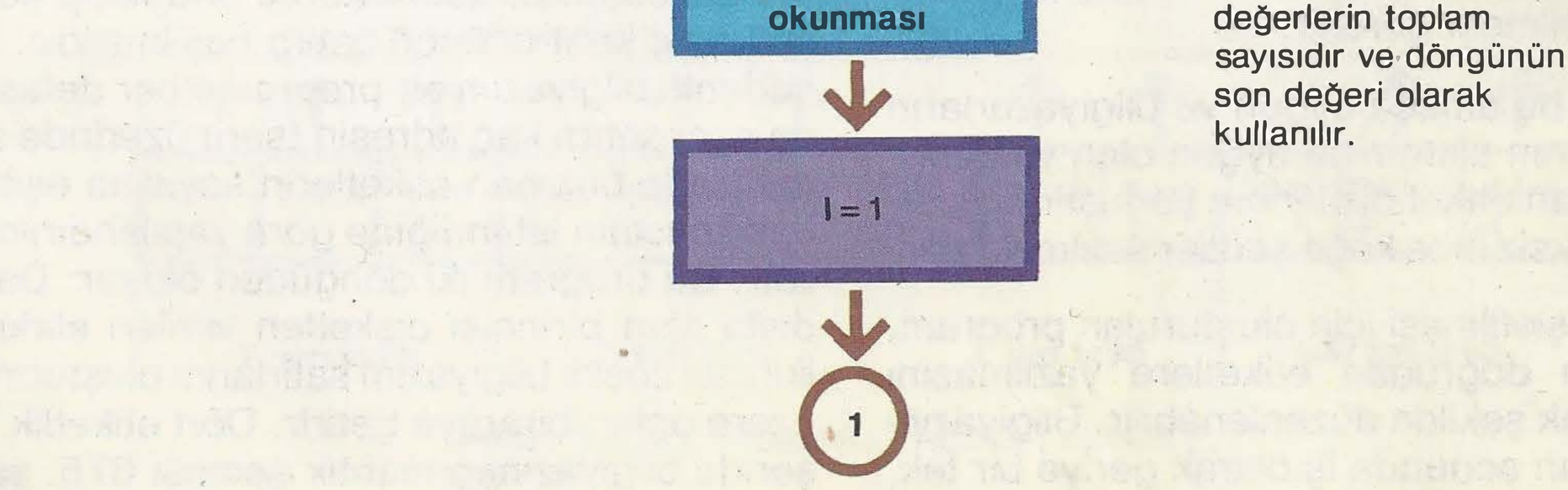
Bilgiyazım (alt-program 6000)

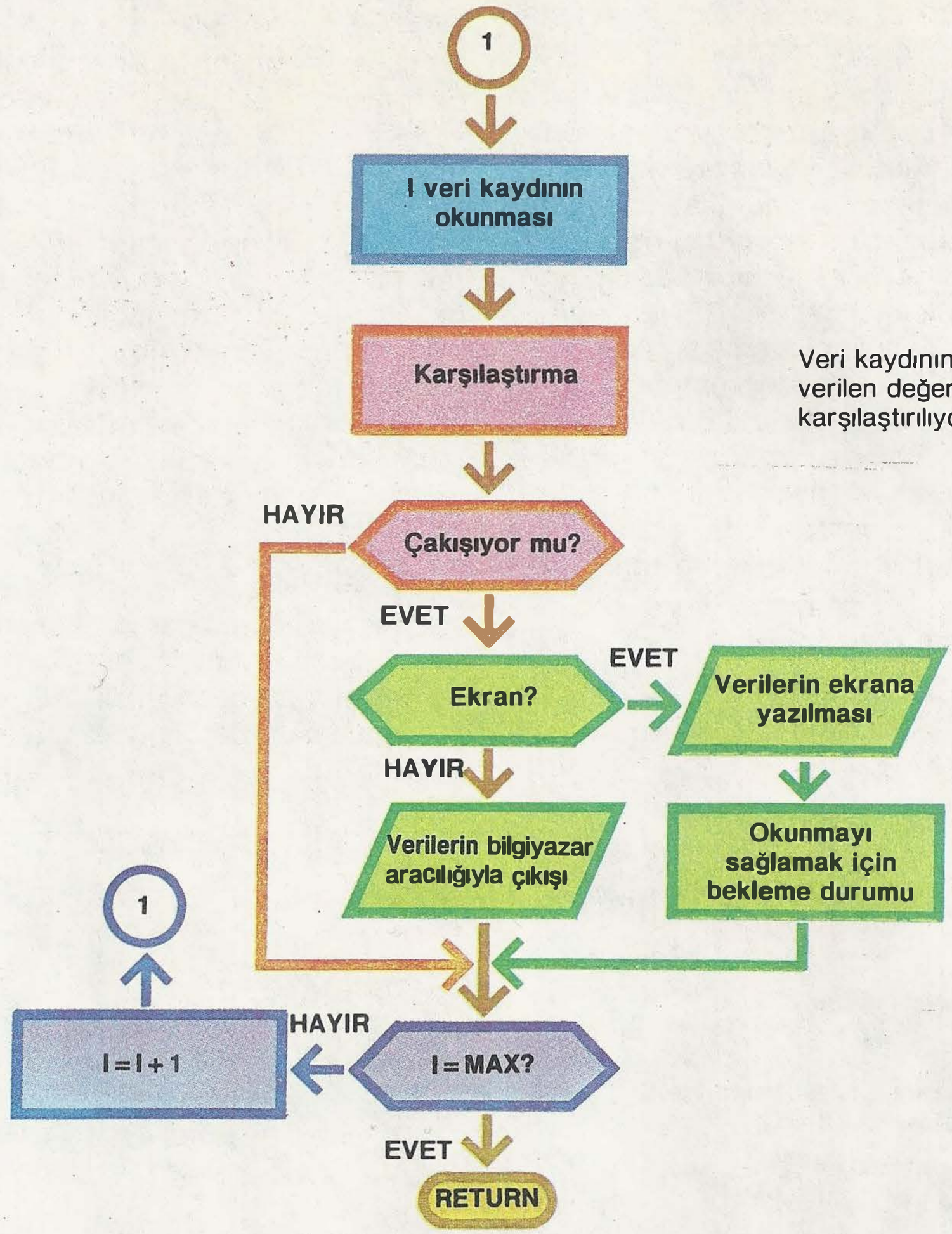
Bilgiyazım alt-programı, kütüğün kağıt üzerine bilgiyazımını eksiksiz yapabilecek durumda olmalıdır. Bu nedenle, bu alt-program, ilk veri kaydının numarasından, daha önce yazmada kullanılan son veri kaydının numarasına kadar tekrarlanan bir döngüden oluşmalıdır. Zaten son yazılmış veri kaydının numarası kütüğün son veri kaydında saklanmaktadır. Prosedürde özel güçlükler yoksa, akış şeması atlanır ve okuyucu, tercih ettiği bilgiyazım düzenine uygun bir program elde edebilir. Bilgiyazım aşamasında daha önce verilen parametreli bilgiyazım alt-programının kullanılıp kullanılmaması gerektiğine karar verilebilir.

Özel Bilgiyazım Fonksiyonları

Yazmış olduğumuz program sadece alanların uzunlukları ve anlamları değiştirilerek genel amaçlı bir hale getirilebilir. Buna rağmen, bazı uygulamalarda bilgiyazım alt-programında

ARAMA ALT-PROGRAMI





Veri kaydının içeriği, verilen değerlerle karşılaştırılıyor.

özel değişikliklerin yapılması gerekebilir. Örneğin, eğer kütük, reklam ya da benzeri malzeme gönderilmesine yönelik bir adres listesi ile ilgili ise, programın, çıktıları (yani basılacak adresleri), kendiliğinden yapışan etiketler gibi, doğrudan kullanılabilir bir malzeme üzerine verebilmesi gerekir.

Piyasada bu amaca uygun ve bilgiyazarların kağıt aktarım sistemine uygun olan ve üzerine yapışkan etiket dizilerinin yerleştirilmiş olduğu kesiksiz ince kağıt şeritler satılmaktadır.

Kütüğün işletilmesi için oluşturulan program, adreslerin doğrudan etiketlere yazılmasını sağlayacak şekilde düzenlenebilir. Bilgiyazım aşamasının sonunda iş olarak geriye bir tek, basılı etiketleri gönderilecek nesnelere yapıştırmak üzere kağıt şeritten sökmek kalacaktır.

Etiketler üzerine bilgiyazım, alışılmış olandan farklıdır, çünkü bilgiyazımın yapılandırılması gerekmektedir. Yani önce isim bir satıra (bütün etiketlerde), sonra adres onu takip eden satıra ve kent üçüncü satıra basılmalıdır. Bu nedenle bilgiyazım alt-programı, her defasında aynı satıra kaç adresin (şerit üzerinde aynı hizada bulunan etiketlerin sayısına eşittir) basılmasının istendiğine göre yapılandırılmalıdır. Bu program iki döngüden oluşur: Daha dışta olan birincisi disketten verileri alırken, ikincisi çeşitli bilgiyazım satırlarını oluşturmak üzere onları biraraya getirir. Dört etiketlik bir şeride bilgiyazımın mantık şeması 675. sayfada verilmiştir. Bu durumda dışta bulunan

döngü 4 adımlık bir uzunluğa sahiptir, içerdeki ise her defasında adı, soyadını, sokağı ve kenti içeren üç satırı oluşturur.

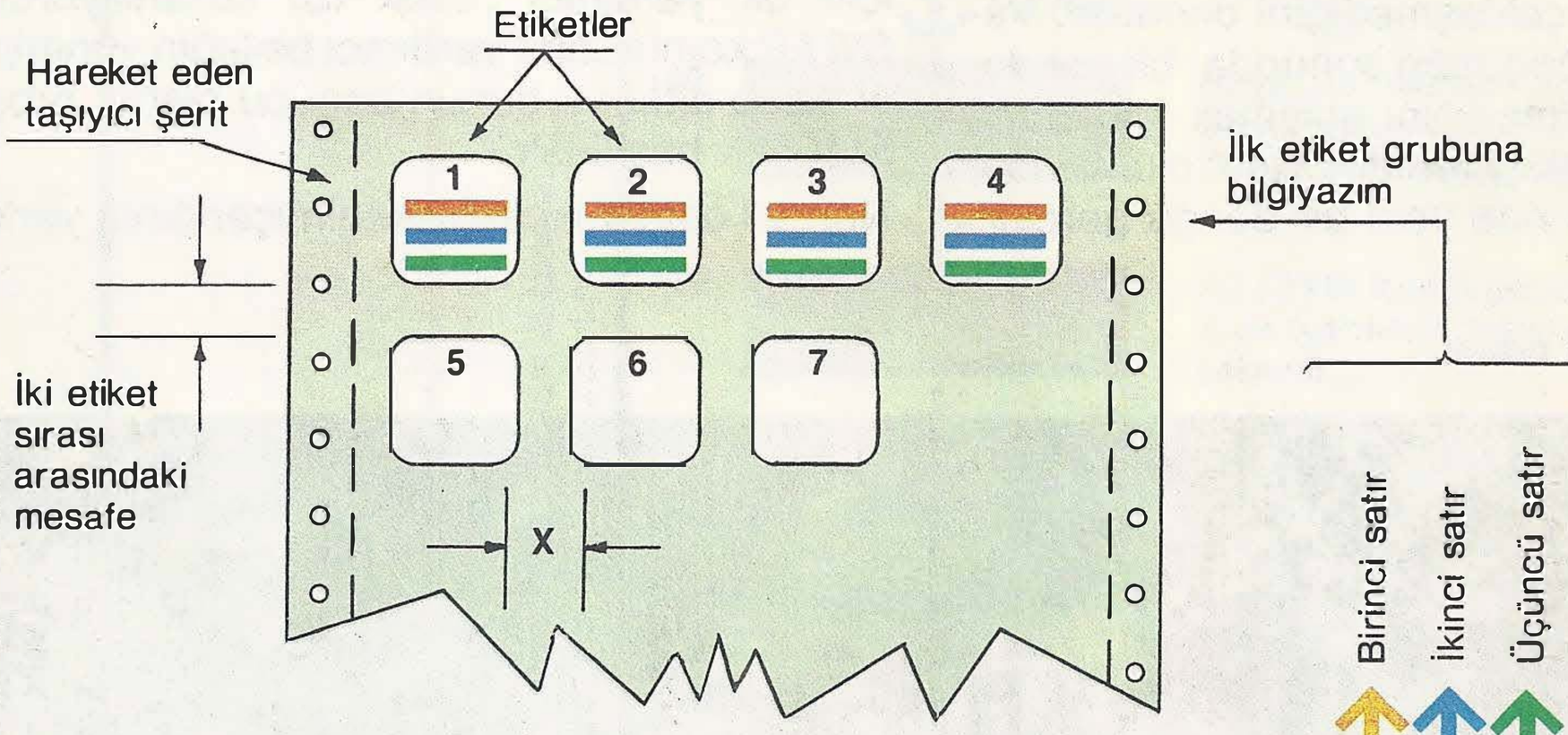
Bilgiyazım komutları tek tek alanlar arasında adreslerin etiketlere doğru yerleştirilebilmesi için yapılması gerekli cetvellemeleri (tabülasyonları) öngörmelidir.

Verilerin Sıralanması

Kütüklerin işletilmesi sırasında, veri kayıtlarını oluşturan alanlardan herhangi birine ilişkin verilerin sıralanması için, alt-programlar bulunması zorunludur.

Sıralama (Sort) olarak adlandırılan genel uygulamaya yönelik bu alt-programlar normalde makinenin yapımcısı tarafından sağlanır veya kolayca elde edilebilir. Bununla birlikte, eğer veri sayısı çok fazla değilse, varolan duruma daha iyi uyabilen, özel alt-programlar kullanmak her zaman için daha yararlıdır. Birçok uygulamada bellekte bulunan tabloları düzenlemek gerekir. Genel uygulamaya yönelik yazılım, kütüklere yani yığın belleklerle (disketler) çalışır ve bu durumlarda kullanılması yarardan çok zarar getirir. Ancak verilerin sıralanması için alt-programlar yazılmasının, ön koşul olarak bir BASIC derleyicisinin bulunmasını gerektirdiği akıldan çıkartılmamalıdır. Çevrilmiş BASIC kullanılması yoluyla yazılım gerçekleştirilebilir, fakat düşük iletişim hızından dolayı çevrilmiş bir programın pratikte kullanımı olanaksızdır. Bu nedenle derlenmesi gerekir. Verilerin sıralanması için Assembler dilinde bir program da yazılabilir. Bu belirli bir açıdan en iyi çözüm yolu olmasına rağmen, diğer yandan ileri boyutlarda tecrübe ve yazma ve deneme aşamaları için çok daha fazla za-

ETİKETLERE BİLGİYAZIM



İsim 1	İsim 2	İsim 3	İsim 4
Sokak 1	Sokak 2	Sokak 3	Sokak 4
Kent 1	Kent 2	Kent 3	Kent 4

Kütük, 4'er veri kaydından oluşan gruplar halinde okunur. İlk etiketin üç satıra bilgiyazımı aynı satıra aynı türde alanların verilmesiyle sağlanır.

İsim 1	Sokak 1	Kent 1	İsim 2	Sokak 2	Kent 2	İsim 3	Sokak 3	Kent 3	İsim 4	Sokak 4	Kent 4
1. veri kaydı			2. veri kaydı			3. veri kaydı			4. veri kaydı		

```
LPRINT TAB(X1) ; ISIM1$ ; TAB(X2) ; ISIM2$ ;
TAB(X3).....
```

man harcamayı gerektirmektedir. İleri sayfalarda verilerin sıralanmasına yönelik iki alt-program sunulacaktır. Bu alt-programlardan biri, bellekte çalışır ve yalnızca incelenen kütüğün tamamının makinenin belleğine sığdırılması koşuluyla disket üzerinde bulunan veriler için kullanılabilir. İkinci alt-program disket üzerinde saklanan kütüklerle çalışır ve iki arama tekniği kullanılır. Bu alt-program büyük verim sağlar ve orta boyutlardaki kütüklerin işletilmesinde kullanılır.

Verilerin Bellekte Sıralanması

677. sayfada A(20) dizeyinin içeriğini artan değerlere göre sıralayan örnek bir program akış şeması verilmektedir.

Sıralama, dizeyde bulunan her bir değerin (büyük göstergelere doğru) hemen yanındaki ile karşılaştırılmasıyla elde edilir; eğer ilk değer daha büyükse ikisi yer değiştirir. Tüm veriler üzerinden yapılan tam genel yoklama artık hiçbir yer değiştirmeyi gerektirmiyorsa, sıralama sona erer; bir yer değiştirme gerektiğinde, 1'e yerleştirilen K çentiği bu koşulun gerçekleşip gerçekleşmediğini denetler. Verilerle ilgili bir döngünün sonunda, bir yer değiştirme olup olmadığını anlamak için K'nin değerine bakmak yeterlidir. Yanıt olumlu ise, tüm veriler üzerinde yeni bir döngü gerekir. Aksi halde, yani çentiğin değeri sıfır ise yer değiştirme olmamıştır. Veriler sıralanmıştır (her veri kaydı bir sonrakinden küçüktür) ve alt-program sona erebilir. Buraya dek betimlenen (ve dökümü s.678-679'da verilen) program yalnızca bir örnektir. Aynı program K çentiğinin işlevinin daha iyi izlenebileceği bir biçimde daha derli toplu yazılabilir.

Bu çentik yalnızca 1 veya 0, yani doğru ve yanlış olmak üzere iki değer alabilen göstergedir. Değerlerden biri belli bir koşulun (verilerin sıralanması) yerine gelmediğini ifade eder; karşıt değer ise aynı koşulun yerine geldiğini gösterir. Veri sıralama döngüsü, koşul yadsınana değin yinelenmelidir. Bu WHILE... WEND komutunun uygulanmasına klasik bir örnektir.

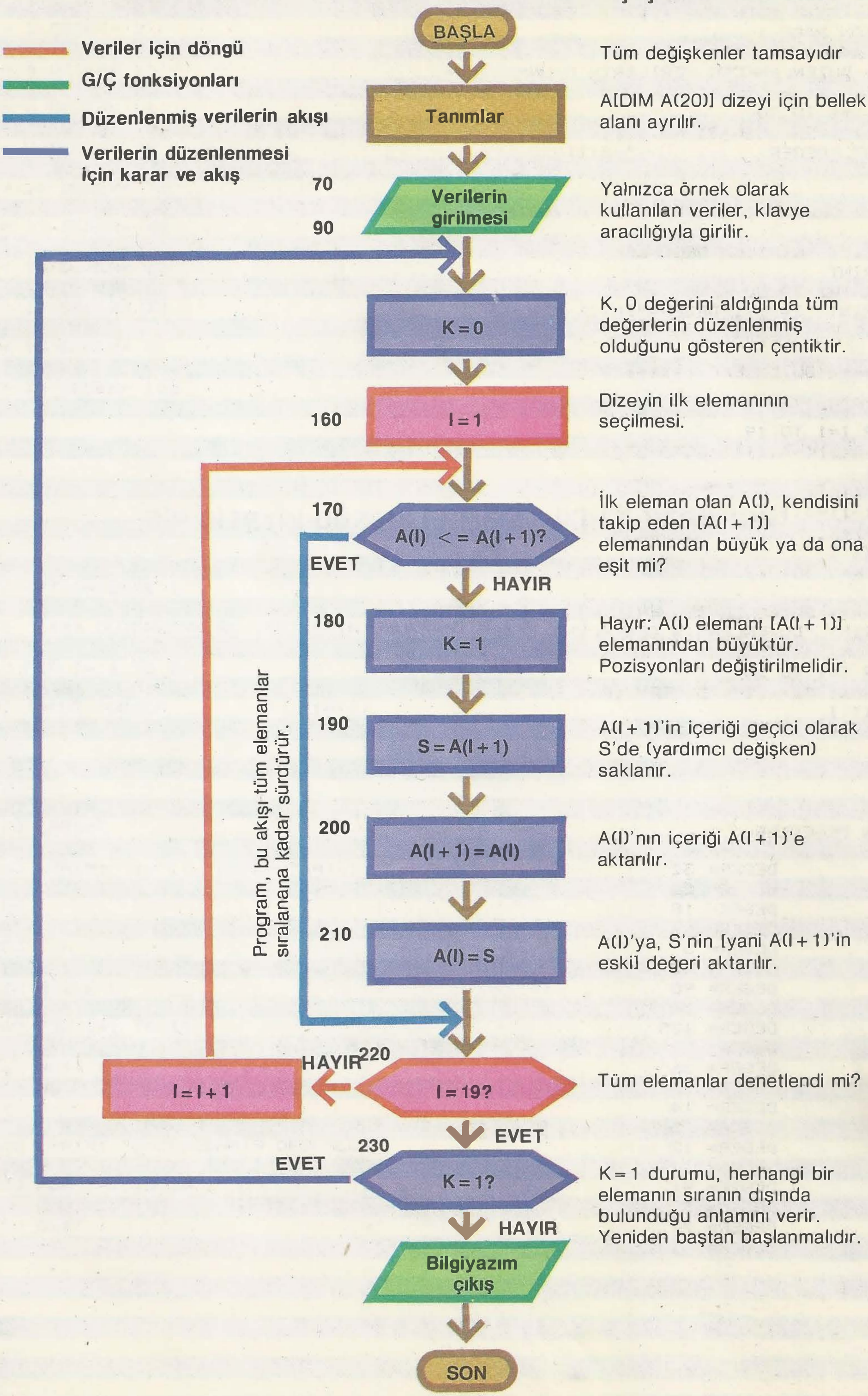
Aynı program sayfa 679'da altta, bu komutun kullanılmasıyla, yeni baştan yazılmış olarak verilmektedir. Ancak bu döküm yeni bir uygulamayla zenginleştirilmiştir: SWAP komutunun kullanımı. Bir önceki programda A(I) ve A(I + 1) belleklerinin içeriklerinin yerlerini değiştirmek için bir yardımcı bellek (S) kullanılıyordu. SWAP komutunda yardımcı belleğin yönetimi sisteme aittir ve bunun sonucu olarak program çok basitleşir.

Normal olarak iki değişkenin içeriğinin yerle-

Bir garın kontrol odası.



BİR DİZEYİN DÜZENLENMESİNİN AKIŞ ŞEMASI



Tüm değişkenler tamsayıdır

A[DIM A(20)] dizeyi için bellek alanı ayrılır.

Yalnızca örnek olarak kullanılan veriler, klavye aracılığıyla girilir.

K, 0 değerini aldığında tüm değerlerin düzenlenmiş olduğunu gösteren çentiktir.

Dizeyin ilk elemanının seçilmesi.

İlk eleman olan A(I), kendisini takip eden [A(I+1)] elemanından büyük ya da ona eşit mi?

Hayır: A(I) elemanı [A(I+1)] elemanından büyüktür. Pozisyonları değiştirilmelidir.

A(I+1)'in içeriği geçici olarak S'de (yardımcı değişken) saklanır.

A(I)'nın içeriği A(I+1)'e aktarılır.

A(I)'ya, S'nin [yani A(I+1)'in eski] değeri aktarılır.

Tüm elemanlar denetlendi mi?

K=1 durumu, herhangi bir elemanın sıranın dışında bulunduğu anlamını verir. Yeniden baştan başlanmalıdır.

BİR DİZEYİN DÜZENLENMESİ (BİRİNCİ VERSİYON)

```
10 ' ** KUTUK = DUZEN
20 OPTION BASE 1
30 DIM A(20)
40 ' ** DUZENLENECEK VERILERIN OKUNMASI **
50 '
60 FOR I=1 TO 20
70 PRINT "VERI GIRISI NO.:";I
80 INPUT "DEGER                ";A(I)
90 NEXT I
100 '
110 ' ** OKUNAN DEGERLERIN BILGIYAZIMI
120 '
130 LPRINT "GIRILEN DEGERLER"
140 LPRINT
150 FOR I=1 TO 20
160 LPRINT "I=";I,"DEGER=";A(I)
170 NEXT I
180 '
190 ' **** DUZENLEME ******
200 '
210 K=0
220 FOR I=1 TO 19
230 IF A(I)<=A(I+1) GOTO 270
240 K=1
250 S=A(I+1)
255 A(I+1)=A(I)
260 A(I)=S
270 NEXT I
280 IF K=1 GOTO 210
290 '
300 ' **** DUZENLENMIS VERILERIN BILGIYAZIMI ***
310 LPRINT "DUZENLENMIS DEGERLER"
320 LPRINT
330 FOR I=1 TO 20
340 LPRINT "I=";I  "DEGER=";A(I)
350 NEXT I
360 END
```

```
GIRILEN DEGERLER

I= 1            DEGER= 32
I= 2            DEGER= 25
I= 3            DEGER= 18
I= 4            DEGER= 0
I= 5            DEGER= 5
I= 6            DEGER= 48
I= 7            DEGER= 98
I= 8            DEGER= 74
I= 9            DEGER= 125
I= 10           DEGER= 2
I= 11           DEGER= 56
I= 12           DEGER= 36
I= 13           DEGER= 14
I= 14           DEGER= 25
I= 15           DEGER= 12
I= 16           DEGER= 10
I= 17           DEGER= 91
I= 18           DEGER= 58
I= 19           DEGER= 77
I= 20           DEGER= 61
```

```
DUZENLENMIS DEGERLER

I= 1 DEGER= 0
I= 2 DEGER= 2
I= 3 DEGER= 5
I= 4 DEGER= 10
I= 5 DEGER= 12
I= 6 DEGER= 14
I= 7 DEGER= 18
I= 8 DEGER= 25
I= 9 DEGER= 25
I= 10 DEGER= 32
I= 11 DEGER= 36
I= 12 DEGER= 48
I= 13 DEGER= 56
I= 14 DEGER= 58
I= 15 DEGER= 61
I= 16 DEGER= 74
I= 17 DEGER= 77
I= 18 DEGER= 91
I= 19 DEGER= 98
I= 20 DEGER= 125
```

BİR DİZEYİN DÜZENLENMESİ (İKİNCİ VERSİYON)

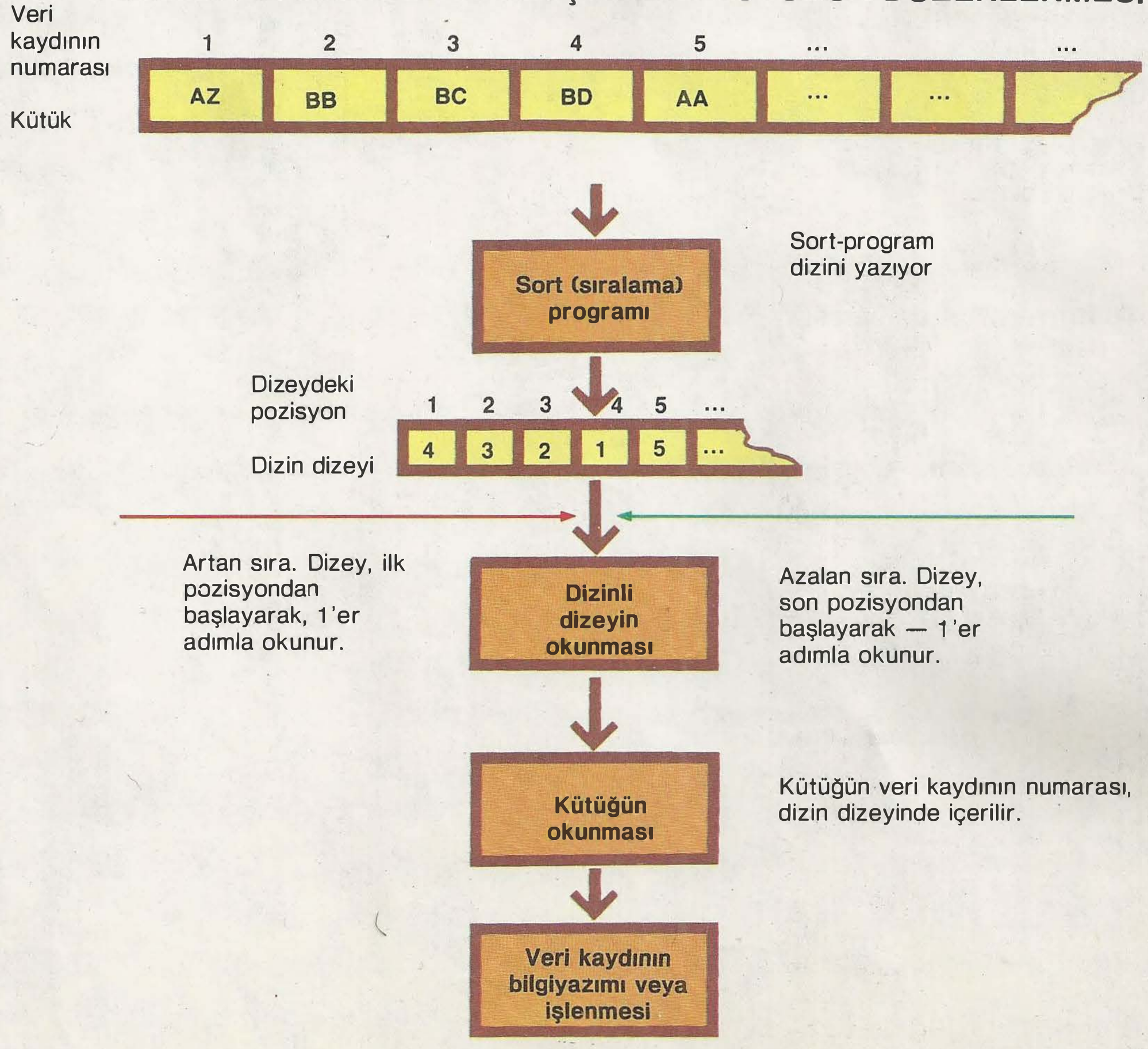
```
10 ' ** KUTUK DUZEN **
20 OPTION BASE 1
30 DIM A(20)
40 ' *** DUZENLENECEK VERILERIN OKUNMASI ****
50 '
60 FOR I= 1 TO 20
70 PRINT "VERI GIRISI NO.:";I
80 INPUT "DEGER              ";A(I)
90 NEXT I
100 '
110 ' OKUNAN DEGERLERIN BILGIYAZIMI ***
120 '
130 LPRINT "GIRILEN DEGERLER"
140 LPRINT
150 FOR I=1 TO 20
160 LPRINT "I=";I,"DEGER=";A(I)
170 NEXT I
180 '
190 ' ** DUZENLEME ***
200 '
210 K=1      ' BIRINCI DONGUYU BASLATMAK ICIN K 1 E ESITLENIR
220 WHILE K
230 K=0
240 FOR I=1 TO 19
250 IF A(I)>A(I+1) THEN SWAP A(I),A(I+1):K=1
260 NEXT I
270 WEND
290 '
300 ' *** DUZENLENMIS VERILERIN BILGIYAZIMI ***
310 LPRINT "DUZENLENMIS DEGERLER"
320 LPRINT
330 FOR I=1 TO 20
340 LPRINT "I=";I "DEGER=";A(I)
350 NEXT I
360 END
```

rini değiştirebilmek için, birinden diğerine geçecek olan değeri kısa bir süre için saklayacak bir yardımcı değişkene gerek vardır. Ama aynı fonksiyon SWAP komutu ile yerine getirilecek olursa, yalnızca değişkenlerin adını değiştirmekten oluşan bir süreç başlatılır. Böylece başlangıç (çıkış) belleği sonuç (varış) belleğinin adını alır veya bunun tersi olur. Bunun programa etkisi verilerin bir bellekten diğerine aktarılmasıyla aynıdır. Yalnızca sistemin adresleri değiştirmesi oldukça yüksek bir iletim hızıyla gerçekleşir. Programı genelleştirmek ve onu böylece herhangi bir büyük N sayısında elemandan oluşan dizeyler için kullanılabilir kılmak için (N sayısı kullanılacak belleğe uygun olmalıdır), 30, 150, 240 ve 330 satırlarında 20 ve 19 değerlerinin yerine yenilerini koymak yeterlidir. Son olarak da, programın, öngörmüş olduğumuz gibi, yalnızca sayısal değerlerle [A(20) dizeyi tamsayılardan oluşur] işlediğine dikkat edilmelidir. Onu karakter zincirleriyle kullanabilmek için, dizeye [A$(N)] ve yardımcı belleğe (S$) karakter zincirleri için gereken simgeyi eklemek gerekir.

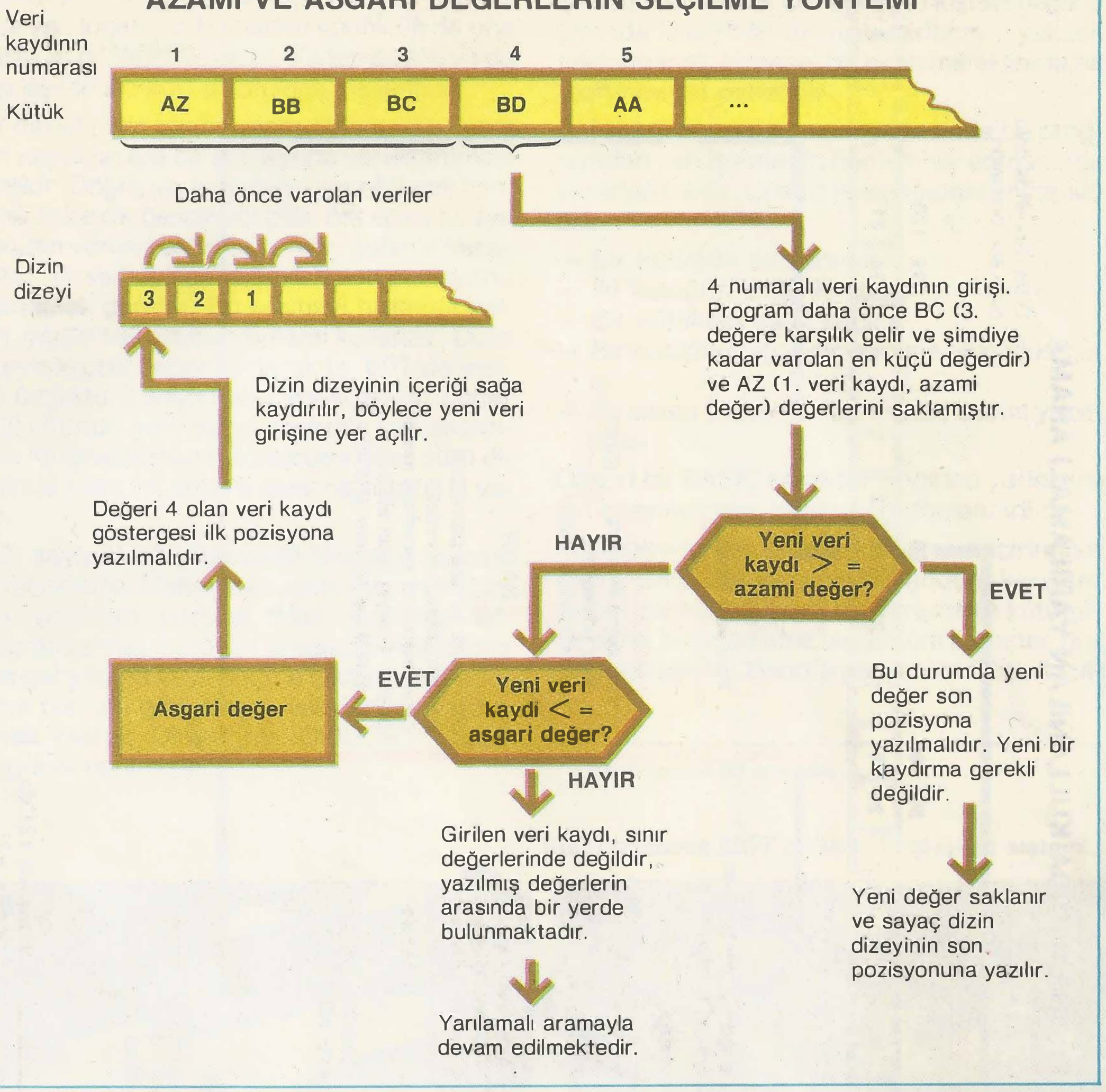
Disket Üzerinde Sıralama

Sunduğumuz örnek, program veri kaydı adresleri için kütükteki veri kayıtlarının sayısına eşit büyüklükte olması gereken bir yardımcı bellek kullanılır. İçeriğin alfabetik sırasına göre (Z en küçük değer, A en büyük değer) seçilen veri kaydı adresleri bu belleğe girilir. Örneğin 15 sayılı veri kaydının verileri 1 ve 2 veri kayıtlarında bulunanlardan daha küçükse, gösterge dizeyine (adreslerin yardımcı belleği) önce 15 değeri veriler için gösterge olarak

DİZİNLİ DİZEYLERDEN OLUŞAN BİR KÜTÜĞÜN DÜZENLENMESİ



AZAMİ VE ASGARİ DEĞERLERİN SEÇİLME YÖNTEMİ



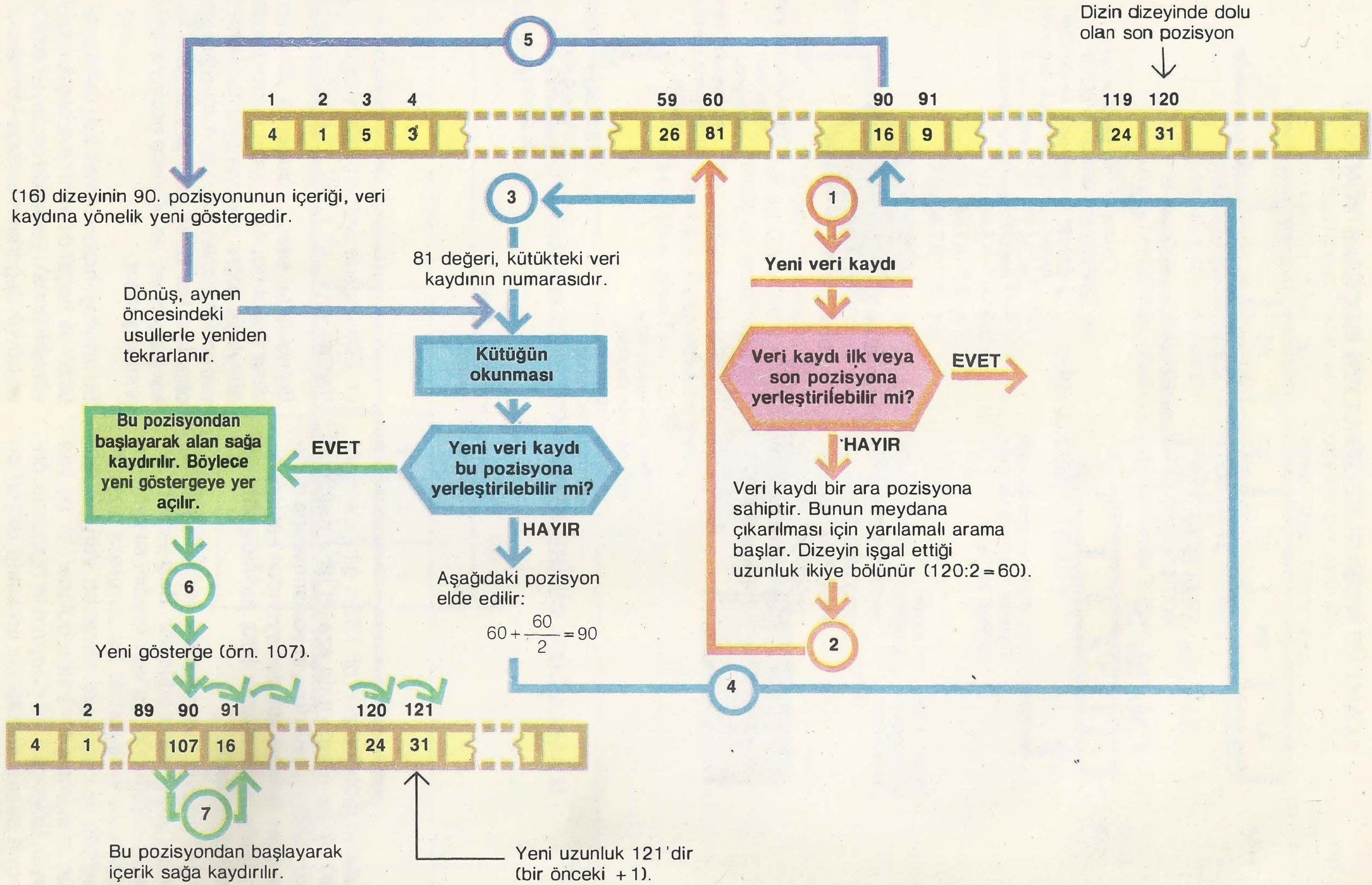
kullanıp (kütük dizinli bir kütük gibi işlem görür) veriler artan ya da azalan bir sıra içinde okunabilir. Dizinli dizeyin yoklanmasına duruma göre ilk veya son pozisyondan başlanabilir. Betimlenen mantık 680. sayfada altta örneklenmektedir.

Dizinli dizeyin hazırlanması yöntemi şematik olarak şöyle tasvir edilebilir. Bazı verileri, örneğin kütüğün ilk üç veri kaydını sıraya soktuğumuzu varsayalım: Bu durumda dizin dizeyinde üç değer yer alacaktır: Veri kütüğünün sıralanmış bir halde bulunan ilk üç veri kaydı. Böylece dizin dizeyinin ilk ögesinde, önceden sıraya sokulanlar arasında değeri en küçük olan veri kaydının numarası bulunacaktır. İkinci ögede ondan büyük veri kaydının numarası vb.

Bu noktada, sıraya sokulacak yeni veri kaydı alınır (kayıt no. 4). Kaydın numarasının dizin dizeyinde doğru yere yerleştirebilmek için, önceden incelenen kayıtların içeriğini yeniden okumak, her birini yeni sayısal veri kaydı ile karşılaştırmak ve böylece pozisyonunu belirlemek gerekir.

Bu aramayı hızlandırmak için (alternatif değil, birbirine bağlı) iki yönteme başvurulabilir. İlk yöntem önceki verilerde raslanan en küçük ve en büyük değerlerin belleğe alınmasına dayanır. Sıraya sokulacak yeni veri kaydının geli-

BİR SORT PROGRAMINDA KULLANILAN YARILAMALI ARAMA



şiyle ilk denetim bu iki değer üzerinde yapılır (bkz. 681. sayfadaki grafik).

Yeni veri kaydı, en küçükten küçük ya da ona eşitse ilk pozisyona, en büyükten büyük ya da ona eşitse en son pozisyona yerleştirilir.

Bu denetimler olumlu sonuçlanmazsa, yeni veri kaydının ara bir pozisyona yerleştirilmesi gerekir. Doğru yerleştirmeyi yapabilmek için, daha önce de belirtildiği gibi, önceden sıraya sokulan verileri okumak, onları gelenle karşılaştırmak ve böylece karşılık gelen pozisyonu belirlemek gerekir. Bu aramayı hızlandırmak için, yarılamalı arama yöntemi kullanılır. Dizin dizeyinin orta değeri alınarak (s. 681'de verilen örnekte 2 sayısı ve karşılık gelen değer (BD) okunur; yeni değer bununla karşılaştırılır ve karşılaştırmanın sonucuna göre dizin dizeyinde sağa mı, sola mı gidileceğine karar verilir.

682. sayfada alt-programın tam akış şeması verilmektedir. Makinenin yapımcısının sağladığı yazılımın parçası olan sıralama alt-programlarının, kütükler arasında eklemleme (merge) yapma veya seçme çıktılar elde edebilme olanağı gibi, başka işlevleri de vardır. Örnek olarak Olivetti'nin OLISORT* hazır programı verilmiştir.

OLISORT Hazır Programı

Olisort özel olarak Olivetti M20 kişisel bilgisayarında işletilmek üzere tasarlanmış yüksek performanslı evrensel bir sıralama/eklemleme (soft/merge) paketidir.

Sağladığı geniş olanaklar sayesinde bu programdan veri işlemenin hemen her safhasında yararlanılabilir. Olisort ile aşağıdakiler yapılabilir.

— Bir kütüğün sıralanması
— İki kütüğün eklemlenmesi
— Bir kütükten kayıt seçimi
— Bir kütüğün diğerinin sonuna eklemlenmesi
— Sıralama yapılırken aynı anda seçim yapılması

Olisort bir BASIC karakter zincirine yerleştirilen parametreler tarafından yönlendirilir.

Parameterlerden oluşan karakter zincirlerinin tümü, BASIC'le yazılmış bir güdüm karakter zinciri yardımıyla veya bir parametre kütüğüne atıfta bulunan kısa bir güdüm karakter zinciri aracılığıyla Olisort'a geçilebilir. Olisort kul-

* Olisort tescilli bir markadır.

IBM firmasının MOT 53385 bilgi-işlem sistemi.



Anlayış İçin Bir Kişisel Bilgisayar

Kişisel bilgisayar, haberleşme için bir araç olmaktan öte önemli hareket ve konuşma bozukluklarından muzdarip kişiler için etkili bir rehabilitasyon aracı olabilir. Cenova Laventi'nin XVI no'lu sağlık biriminde (Motor Eğitim Merkezi'nde) sürdürülmekte olan deneyler bu alanda nelerin yapılabileceğini göstermektedir. Bu merkezde özellikle motor (hareki) eğitimi ile ilgilenmekte ve ağırlıklı olarak küçük ya da yetişkin çağdaki sinirsel kaynaklı hareket bozukluklarına yönelinmektedir. Bu merkez, Cenova ve komşu belediyelerdeki 9 yerel sağlık biriminin herbirinde bulunan özerk dispanserlerden oluşmaktadır.

Ayrıca bu merkezde, zihinsel sınırlılıklarından ötürü konuşamayan özürlü bir grup, kişisel bilgisayar kullanımını ekrana mesaj yazabilecek kadar öğrenmiştir. Bu amaçla klavyeyi kullanabilmek için gerekli hareketleri yapamadıklarından büyük bir tuşa hafifçe basılarak harfleri seçip sözcük ve cümleleri oluşturmaya yarayan LOGOS 4 adlı özel bir programdan yararlanmaktadırlar.

Program, IBM kişisel bilgisayar için Cenova'daki A&B yazılım şirketi tarafından Motor Eğitim Merkezi'nin Cenova Üniversitesi Elektronik Enstitüsü ile işbirliği içinde CNR'nin (Ulusal Araştırma Kurumu) Biomedikal Teknolojiler projesi çerçevesinde ortaya koyduğu deneysel prototipler bağlamında geliştirilmiştir. Programı oluşturmak için, İtalyanca hakkındaki istatistiksel verilerden yararlanılmıştır. Bu veriler özellikle ilkokul çocuklarının yazdığı ya da onlar için yazılmış metinlerden elde edilmiştir. Bu verilerden ve önceden seçilmiş karakterlerden yola çıkılarak, bilgisayar birbirini izleyebilen harfleri seçer ve onları azalan olasılıklar sırasına göre ekranda gösterir. Özürlü kişi ya tek bir düğmeye (ki bu durumda harfler ayarlanabilir bir hızla otomatik olarak birbirini izler) basarak ya da biri kabul biri red olmak üzere iki düğmeden birini seçerek, istediği harfleri makineye bildirir.

Bundan sonra LOGOS 4 ekranda yansıtacağı harfe karar verirken İtalyanca istatistiksel verilerden yararlanıyor. Belli bir harf grubundan sonra bütün harflerin kullanılabilme olasılığı eşit değildir. Bunun için kullanıcının girdiği her harf grubundan sonra ekranda farklı bir harf grubu görünür. Örneğin, eğer ''ch'' sessiz harfler grubu seçildi ise ekranda hemen ''e'' ve ''i'' harfleri görülür.

Böylece tüm alfabeyi katetmek gerekmez. Bu ise, araca bir kez alıştıktan sonra oldukça hızlı biçimde iletişim kurmayı mümkün kılar.

Ayrıca makinede karakter dizisini tersine çevirmeye ve otomatik yazıdan elle yazıya geçmeyi mümkün kılan iki tuş daha vardır. Harfler gözün çevre görüşünden yararlanıp seçime zamanında hazırlanabilmesi için özürlü kişiye farklı renklerde ve ''huni şeklinde'' özel başka grafik biçiminde verilir.

Seslendirme etkisini andıran bir sunuş biçimi hareket güçlüğünün aşırı olduğu durumlarda harflerin ekrana yansıma hızını yavaşlatarak seçme işlevinin yerine getirilmesini kolaylaştırır.

Seçilen harflerin renklerinin farklılılaştırılması özellikle görsel sorunları olan kullanıcılar tarafından tanınmalarını kolaylaştırmak için geliştirilmiştir. Harfler önce ekranın solunda gözükür ve hem sözcüklerin kurulmasında, hem de gerekebilecek düzeltmelerde kullanılabilir. Renkli oldukları gibi, ekranın alt kısmında bulunan ve asıl metni oluşturan harflerden daha büyük boyutludurlar. Harflerin istatistiksel bir araştırmanın sonuçlarına göre sunulduğu daha önceden belirtilmişti. İtalyan dilinde özellikle ilkokul metinlerinde bulunan sözcüklerle ilgili veriler belleğe alınmıştır.

İstatistiksel araştırma, yazılan son harf grubundan sonra gelebilecek harflerin olasılıklarının saptanmasına yaramıştır. Alfabenin 26 harfinden başka ara verme, geri dönme, satırbaşı yapma, seçme, hatayı düzeltme, henüz yazılmış ve makine tarafından belleğe alınmamış metnin bilgiyazımını yapma ve programın ilk işlevlerine dönme görevlerinin harekete geçirilmesini sağlayan beş işaret bulunmaktadır.

Daha önce de söylendiği gibi, LOGOS 4 IBM kişisel bilgisayar üzerinde geliştirilmiş bir programdır.

Uygulama için gerekli asgari araçlar şunlardır: Merkezi İşlem Birimi (RAM 128K) + kart(fiş)

— Oyuncak adaptörü
— 1 adet 180K'lık disket okuma birimi
— Klavye
— Sözcükler için bir küçük klavye
Bilgisayar (80 cps)

Çizisel ekran

Programın yapısı LOGOS 4'ün klavyesiz ve bilgiyazarsız kullanılmasını mümkün kılmaktadır. Ancak bu durumda karşılıklı etkileşim ve bilgiyazım işlevlerinin gerçekleştirilemeyeceği açıktır.

Çizisel ekranın da kaldırılarak onun yerine uygun bağlantılarla (modülatör) monitör olarak kullanılacak bir televizyonun geçirilmesi mümkündür. Kullanıcılara bağlı olarak sisteme özel klavyeler (örneğin büyük boyutlu, havayla çalışan vb.) takılması da düşünülebilir. Bu tür klavyelerin farklı gereksinimlere bağlı olarak duruma göre takılacağı dikkate alınmalıdır.

Gelecekteki değişimleri (örneğin istatistiğin sürekli yenilenmesi ya da başka türlü yenilikler) gözönüne alarak en az sayıdaki parçadan oluşan IBM kişisel bilgiyazar, başka bir birim ya da 180K'lık disketler ya da bir önceki birimin yerine 360K'lık bir disket okuma birimi konarak geliştirilebilir.

LOGOS 4 sisteminin kullanımını ''kişiye özgü kılma'' olanakları fazladır. Program her kullanıcıya kendi hareket olanaklarına göre programlanmış bir işletim disketi verebilmektedir. Aslında ekran ve bilgiyazım özellikleri konusunda bazı, tercihler öngören iki ''maske'' bulunmaktadır.

''Maske'', hareket denetim birimleri ve görme yetisi ile ilgili bazı değişkenler sunar.

Harf seçimi iki biçimde yapılabilir: otomatik yazıda harfler sağdan sola yine programlanabilir (bir saniyeden beş saniyeye kadar) bir hızda hareket eder (yer değiştirir); elle (manuel) yönetilen yazıda ise harfler sabittir ve kullanıcı kabul (evet) ya da red (hayır) tuşlarına basarak istediği harfi seçer.

Bu ikinci yazı türünü kullanabilmek için özürlü kişi her iki tuşu kullanabilecek durumda olmalıdır.

Otomatik yazı programı özürlünün harf seçimi için gerekli hareketi en az çabayla yapmasına yarar. (Bazen bir tek hareket bile yeterli olabilir.)

Elle yazı programı ise daha büyük bir hız ve makinenin doğrudan işletilmesini sağlar. Görsel bilgilerin edinilmesini kolaylaştırmak için fonun (16 renk vardır), hunideki harflerin (4 renk), seçilmekte olanın (4 renk) ve seçilip ayrılmış harflerin (4 renk) renklerini değiştirmek mümkündür. Bu değişiklikler renklerin kullanıcının görüş, tercih ve koşullarına uydurulmasına olanak verir.

Renk değişiklikleri kullanarak görsel malzemenin karmaşıklık düzeyi kullanıcının algılama ve kavrama yeteneklerine göre ayarlanabilir. Yine renk farklılaştırmasından yararlanılarak hunideki harflerin ya da metin için seçilip ayrılmış harflerin silinmesi de mümkün olur. Bilgiyazım için satır başına düşen karakter sayısını seçmek ve böylece harflerin büyüklüğünü, bilgiyazımın sıklığını ve boşlukları (satır aralığı) programlamak mümkündür.

Program, kullanıcıya, programı kendi başına işletmesini sağlayacak yedi işlev sunarak başlar.

Özürlü kişi yeni bir metne başlayabilir, isterse belleğe alabilir, aynı metin üzerinde farklı zamanlarda çalışmayı sürdürebilir, disketten belleğe alınmış metni isteyebilir, tüm işlenmiş metnin bilgiyazımını yapabilir.

Bundan başka bir çevre-denetim işlevi olarak bir ''çağrı'' getirilmiştir. Bu işlev seçildiğinde özürlünün muhtemel yardım isteklerini ileten aralıklı bir sesli işaret harekete geçirilir.

Ayrıca, normal klavye ile özürlü kişinin kullandığı arabirimler arasında iletişim kurulması olanağı da vardır.

Bu da işletmenin ekranda, kullanıcı tarafından yanıtlanacak soruları formüle etmesine yarar ki, bu işlev özürlünün programın işleyişini kavramasını kolaylaştırır.

Önemli hareket (motor) güçlükleri olan özürlülerin gereksinimlerini karşılamak üzere planlanan LOGOS 4 yalnızca bir tek tuş ya da algılayıcı (sensor) kullanarak yazmayı da mümkün kılar.

Bu tuş seçilecek harfin kabulüne yarar. Daha önceden ''yitirilmiş'' olan ya da raslantısal olarak şeridin dibinde olan bir harfin bulunmasını sağlamak için harflerin gösterimini tersine çevirebilen ikinci bir tuşu devreye sokarak, işlem hızı özürlünün yeteneklerinin izin verdiği ölçüde artırılabilir. Üçüncü bir tuşa dokunarak otomatik yazıdan elle yazıya geçmek mümkündür. Elle yazı durumunda özürlü, yazı hızını, biri harfi reddetmek diğeri de kabul etmeye yarayan iki tuşla yönlendirilebilir.

Dört tuşun birlikte kullanılması, aracı özürlü-

LOGOS 4 PROGRAMINI KULLANAN DONANIM BİÇİMİ (CONFIGURATION)



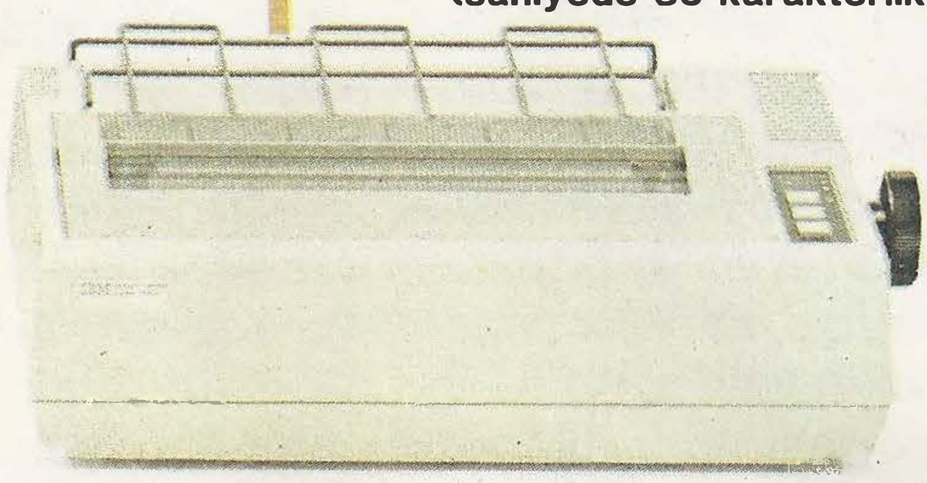
Çizisel ekran

Metin kısmı

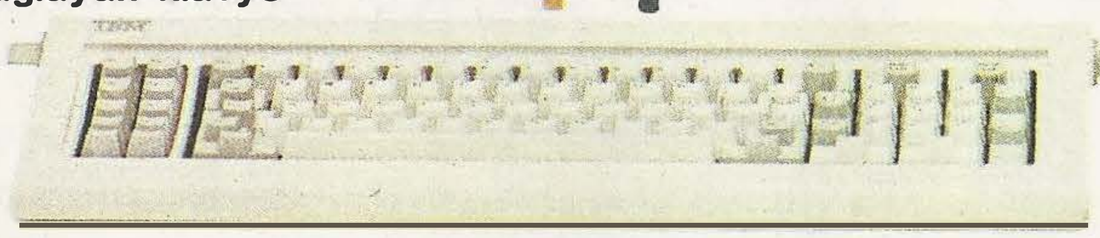
İşlenmiş metin kısmı

180 KB'lik disketler için sürücü birimi

Bilgiyazar
(saniyede 80 karakterlik)



Özürlü kişi ve yardımcısı arasında bilgi alışverişini sağlayan klavye



Klavye/makro klavye ayırımı

İleri/geri

Karakterin kabulü için tuş

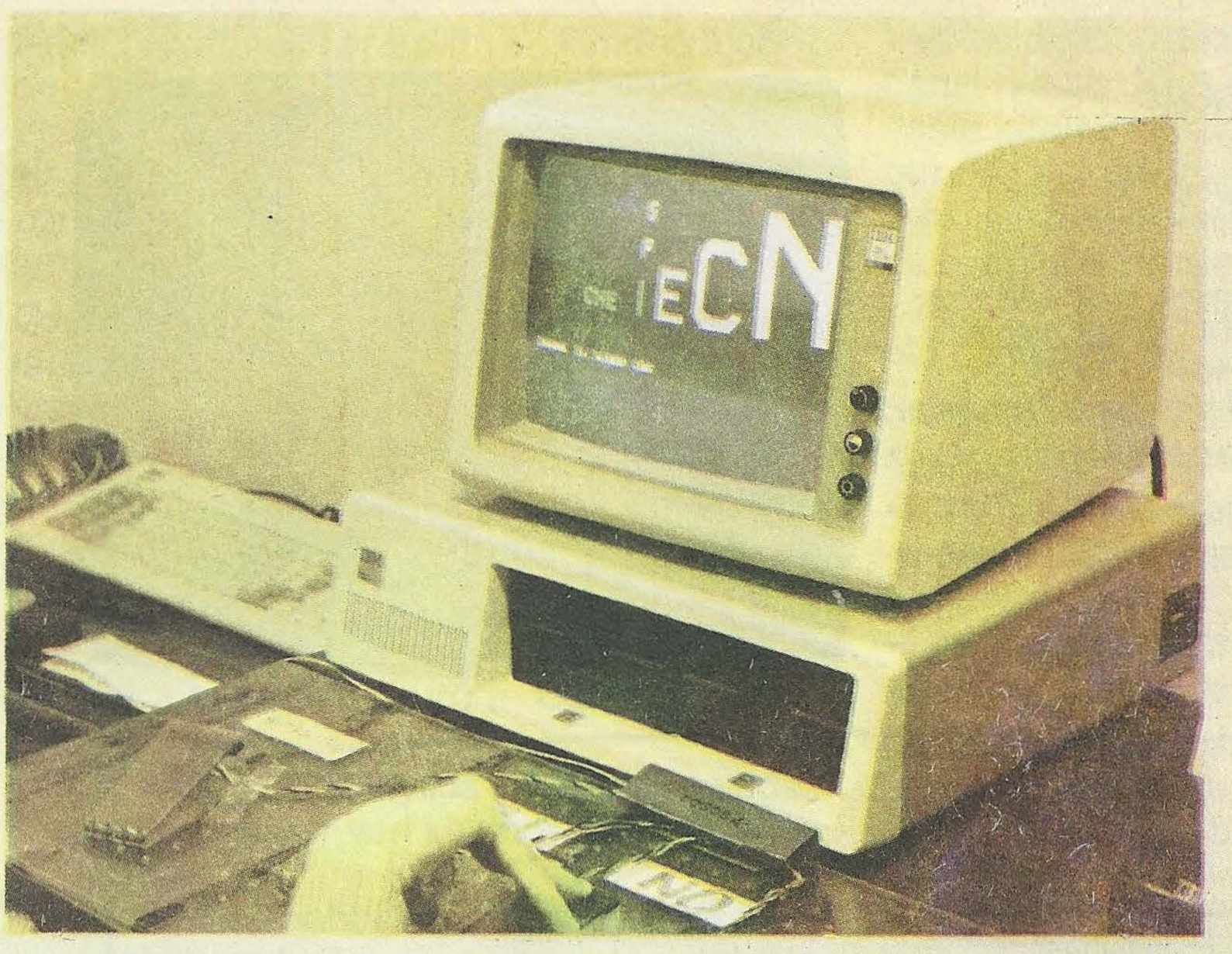
EVET

Karakterin reddi için tuş

HAYIR

Yoklama hızı

Makro klavye ya da özürlü kişinin yeteneklerine uygun başka bir giriş gereci.



İşlemekte olan LOGOS 4 programından bir görünüm. Kişisel bilgisayarların özürlülere yardım amacıyla kullanım olanakları giderek önemli boyutlara ulaşmaktadır.

nün hareket güçlüklerine uydurma isteğinin bir ürünüdür.

Son olarak, daha kolay kullanım biçimlerinden (örneğin bir tek tuş kullanımı) başlayıp, giderek metin yazım süresinin optimizasyonu için, tuşların dördünün de kullanımına yönelerek karmaşıklaşan stratejiler tasarlanmaktadır. Böylece programın sağladığı kolaylığın yanında eğitim için de kullanılabilmesi mümkün olacaktır. Daha kolay stratejilerde -yine ağır bir özrün varlığı halinde- mekanik tuşun yerine daha gelişkin bir algılayıcı geçirilmesi öngörülmektedir.

Havalı tuşlar şimdiden denenmiştir ve ayrıca ultrasonla çalışanlardan, enfrarjlu, radarlı, vb. olanlara kadar her türlü çevresel algılayıcı birarada kullanılabilir. Bugün yalnızca 26 harf çizisel olarak gösterilebilmektedir. Kısa bir süre içinde sayısal ve noktalama ile ilgili simgeler de sisteme katılacaktır.

LOGOS 4'ün evrimiyle ilgili öngörülen hedefler umut vericidir. Şu an sürmekte olan bir projede kullanım alanının çeşitli yaş gruplarına, mesleki eğitimlerine yaygınlaştırılması öngörülmektedir. Bu uygulamalarda kullanılacak istatistikler, programı kullanacak olan özürlü kişinin kendi yazdığı metinlerden elde edilecektir. İkinci bir hedef, kısaltılmış bir sözcüğü ya da büzüşmüş bir tümceyi yazacak sözsel olmayan özel simgesel iletişim sistemlerinin (örneğin BLISS dilini tam olarak okuyacak bir program) geliştirilmesidir.

Geliştirilecek bir kod çözme programı yazı sürelerini büyük ölçüde azaltarak özürlünün yeteneklerini artıracaktır.

Özel olarak, LOGOS 4'te bulunan işlevler aracılığıyla, özürlü, yazılı bir metni belleğe alabilir, bilgiyazımını yapabilir, ekrana aktarabilir, yardım istemek için bir ses işaretine başvurulabilir, aynı amaçla sistemin klavyesinden yararlanan kişilerle iletişim kurabilir.

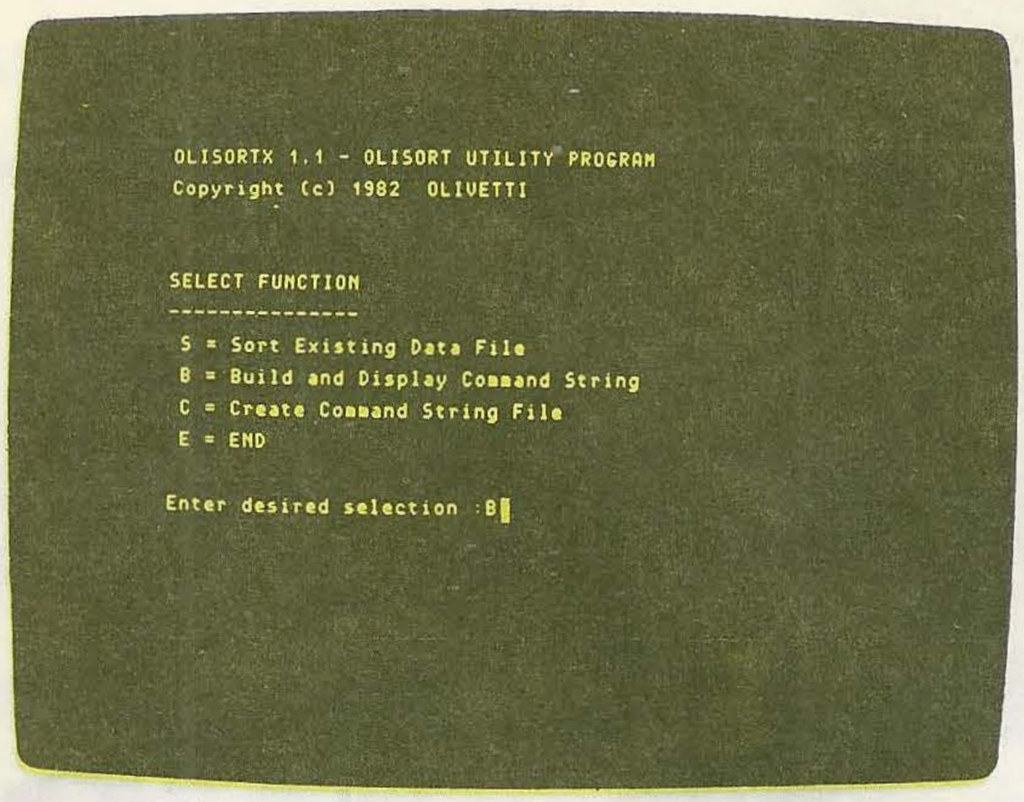
Komutların, renklerin, kullanım biçimlerinin ve hızın değişebilirliği, aracın özürlünün yetilerine uydurulması ve makine ile temas anında onda uyanan genel tedirginliğin azaltılmasını sağlamaktadır.

Ayrıca, yeniden eğitme etkinliği sırasında bazı durumlarda daha kolay kullanım biçimlerinden daha karmaşıklara, giderek daha yaygın kullanımı olan yazı makinesi gibi araçlara geçiş sözkonusu olabilir.

Programın bugünkü biçimi yalnızca okul ve okul sonrası çağdaki özneler üzerinde incelenmektedir, fakat kullanımı ''kişiselleştirilmiş'' yani kullanıcının kendisi tarafından yazılmış istatistiklerin gelişmesiyle, farklı yaş ve hazırlık derecesindeki zihinsel özürlüler tarafından da kullanılabilmesi mümkün olacaktır.

OLISORT PROGRAMI

OLISORT programı, güdümlerden oluşan uygun bir karakter zincirinin herhangi bir BASIC programına katılmasıyla veya OLISORTX paket programının kullanılmasıyla, işletilebilir. Bu sayfada, OLISORTX programı çağrıldığında, ekranda oluşan görüntü sunulmaktadır.




```
OLISORTX 1.1 - OLISORT UTILITY PROGRAM
Copyright (c) 1982  OLIVETTI


SELECT FUNCTION
---------------
 S = Sort Existing Data File
 B = Build and Display Command String
 C = Create Command String File
 E = END

Enter desired selection :B
```

Ekranda beliren yazının ilk iki satırı, kullanıcıya, OLISORT programıyla ilgili denetimin yapıldığını iletmektedir (tam olarak: OLISORT'in 1.1 versiyonu).

Daha önceden varolan bir kütüğün denetlenmesi: S seçeneği.

Bir güdüm karakter zincirinin oluşturulması ve gösterilmesi: B seçeneği.

Güdüm karakter zincirleri içeren bir kütüğün oluşturulması: C seçeneği.

OLISORTX'den çıkış (OLISORT'tan çıkabilmek için): E seçeneği.

Program, kullanıcıdan istenen fonksiyona karşılık gelen kısaltmayı girmesini ister. Örneğimizde B kısaltması seçilmiştir: Fonksiyon, bir güdüm karakter zinciri verilmesini sağlar. Bu karakter zinciri, bir kütük üzerinde SORT programının yürütülmesi için gerekli parametreleri ve değerleri içerir.

lanım programları aracılığıyla, kendi aralarında etkileşebilen bir parametre kütüğü, ya da bir güdüm karakter zinciri yaratılabilir. Bu pozisyon hataları oranını büyük ölçüde azaltarak parametrelerden oluşan uzun karakter zincirleri yaratılabilmesi anlamına gelir.

Olisort, bir BASIC programı tarafından çağrılarak ya da OlisortX kullanıcı programından yararlanılarak işletilebilir. Bu sayede tek bir program olarak işgören bir programlar dizisinde birçok sıralama yapılabilir veya tek bir özel kütük alınır, sıralama programı derhal yürürlüğe konulabilir.

Olisort'un en önemli görevleri şunlardır:

— Veri kaydı başına 10'a kadar sıralama anahtarı ayrılabilir.
— 4'e kadar ''Seç/Ayıkla'' (select/exclude) anahtarı, ile bir veri kaydının seçimi ya da dışlanması
— Seç/ayıkla anahtarı için kıyas limitleri saptanabilir: daha küçük, eşit ya da daha büyük.
— her bir select/exclude anahtarı için * ya da ? kullanılabilir. Birincisi bütün karakterlerin yerine geçebilir, ikincisi yalnız kendinden sonraki bir karakterin. -Seç/ayıkla sort anahtarları ile küçük ya da büyük harflere aynı alfabetik değerin mi verileceği belirlenebilir.
— Olisort'un çalışma kütükleri ve çıktı kütükleri için kullanılan disketler değiştirilebilir; Olisort uygun zamanda bu değişikliği otomatik olarak ister.
— Sıralama ya da eklemlemeye başlamadan bir kütükte baştan 32.767'ye dek başlangıç veri kayıtları atlanabilir.
— Hatalarla ilgilenmek üzere görevler vardır: durum kodu (hata numarası) programa ya da Olisort tarafından çağrılan kullanıcı programına iletilir.
— Olisort'un eldeki BASIC programlarına katılması çok kolaydır ve hiçbir Assembler bilgisi gerektirmez.

Olisort BASIC M20 çeviricide kullanılmak üzere özel olarak planlanmıştır ve sabit uzunlukta alanları olan sabit uzunlukta standart veri kayıtları sağlar.

Yapabildikleri. Birden fazla veri taşıyıcı gerektiren kütükler işlenmediği sürece, Olisort'a yüklenebilecek kütüğün azami hacmi disket üzerinde bulunan alanla sınırlıdır.

Sınırlayıcı etmen giriş kütüğünün hacminden, yalnızca yaklaşık 1.7 kez büyük olabilen sıralama çalışma kütüğünün hacmidir.

Veri kaydının azami uzunluğu 256 bayttır. İstisnai olarak daha uzun olan veri kayıtlarına sahip kütükler sözkonusu olduğunda, bu değer PCOS komutu ile değiştirilebilir. Aşağıda örnekleri basitleştirmek üzere veri kaydının azami uzunluğu olarak 256 baytı alacağız. Tüm kütüklerin veri kayıtlarının sabit uzunluğa sahip olması gerekir.

Sıralama ve seç/ayıkla anahtarları ile aşağıdaki veri türleri işlenebilir:

— harfsayısal karakter zincirleri
— onaltı tabanlı yazı düzenine sahip alanlar
— tamsayılar
— tek hassasiyette sayılar
— çift hassasiyette sayılar

Bir harfsayısal karakter zincirini sıralama programı ya da seç/ayıkla anahtarı ile daha kesin belirleyerek, büyük ve küçük harf karakterlere aynı değerler verilebilir.

Onaltı tabanlı alanlar sıralama ya da seç/ayıkla tuşlarıyla daha kesin olarak belirlendiği takdirde, Olisort'un karakterleri onaltı tabanlı kodlar gibi kullanması istenir. Bunun anlamı, aynı harflerin büyük ve küçük biçimlerinin farklı değere sahip olmasıdır.

On parametre üzerinden tanımlanmış bir sıralama yapıldığında belirleyecek parametreye göre artan ya da azalan bir sıra elde edilir. A (artan) ya da D (azalan) sıralama.

Örneğin, serbest mal kalemlerinin azalan sırasına sahip bir kütüğün sıralanacağını varsayalım. Böylece aşağıda verilene benzeyen bir liste elde edilir:

Serbest Mal Kalemleri

Eyalet	Müşteri	Satış ($)
Nevada	M & M Ltd.	2596.00
	Zilch & Sans	1976.55
	ABC Co.	568.88
New Jersey	NXR Ltd.	8870.77
	Wyatt & Hyatt	8345.98
	Acme Ltd.	5689.00
	Olive Oil Co.	67.43

Sıralanmış bir kütük elde etmek için bir sort programı uygulanabilir veya istenirse sıralı çıkış kütüğü de aynı giriş kütüğünün üstüne yazılabilir.

Bu uygulama özellikle, disket üzerindeki alanın çok küçük olması veya sıralama sırasında disket değiştirmemek istenmesi durumunda, son derece yararlıdır. Fakat sıralama sırasında makinede bir arıza ortaya çıktığı takdirde, giriş kütüğü yitirilebilir veya kullanılmaz hale gelebilir. Bu nedenle olası makine arıza-

larına karşı, bu yöntem kullanılırken sıralama işlemini giriş kütüğünün bir kopyasını içeren ikinci bir disket üzerinde uygulamak gerekir.

Olisort'la, yalnızca giriş kütüğünce seçilen veri kayıtlarını içeren bir çıkış kütüğü elde edilebilir. Sıralama işlemi ile eşzamanlı olarak, bir seçme/ayıklama işlemi de yürütülebilir. Örneğin biraz önce gördüğümüz müşteri kütüğü üzerinde hem sıralama hem seçme/ayıklamayı aynı anda yürütmek ve bu dışlama sırasında mal kalemleri 1000$'ın altında olan müşterileri elemek istiyorsak, sonuç olarak şöyle bir bilgiyazım elde ederiz.

Seçme/Ayıklama ile Satış Listesi

Eyalet	Müşteri	Satış ($)
Nevada	M & M Ltd.	2596.00
	Zilch & Sons	1976.55
New Jersey	NXR Ltd.	8870.77
	Wyatt & Hyatt	8345.98
	Acme Ltd.	5689.00

Dört haneye kadar seçme/ayıklama anahtarı belirleyerek, veri kayıtları seçilebilir ya da dışlanabilir. Bu anahtarlar aynı anda tanımlanabilecek sıralama anahtarlarından tümüyle bağımsızdır. Yani, sıralanmakta olanların dışındaki alanlarda da seçme veya dışlama yapılabilir. Eğer sıralama anahtarları değil de yalnızca seçme/ayıklama anahtarları kesin olarak belirlenirse, giriş kütüğünün sırasını izleyen fakat seçilmiş veri kayıtlarını içeren bir çıkış kütüğü verir. Eğer seçme/ayıklama anahtarı olarak harfsayısal bir karakter zinciri belirlenirse, Olisort, anahtarın içeriği ile veri kaydı içindeki alanın aynı uzunlukta olup olmadıklarını denetler, daha sonra ikisini karşılaştırır. Bu sürecin mantığı şöyledir:

1 / Olisort, seçme/ayıklama anahtarı ile bunun karşılaştırılacağı alanın hangisinin daha uzun olduğuna bakar.
2 / Kıyas yapabilmek üzere, uzun olan kısa olanın uzunluğuna getirilir.
3 / Karakterler tek tek karşılaştırılarak, tüm karakterler eşitse olumlu bir karşılaştırma yapılmış olur.

Örneğin bir veri kaydında aşağıdaki türden bir alan varsa,

Kurşun boru 12 mm X 3 mm

Seçme anahtarı şöyle olabilir:

Kurşun boru

Seçme/ayıklama anahtarına **karşılaştırma karakteri** olarak adlandırılan şu karakterler verilebilir:

anahtardaki karakterin karşılaştırılan alanda
karşılık gelen karakterden küçük olduğunu gösterir.
anahtardaki karakterin karşılaştırılan alanda karşılık gelen karakterden büyük olduğunu gösterir.
= anahtardaki karakterin karşılaştırılan alanda karşılık gelen karaktere eşit olduğunu gösterir.

Karşılaştırma karakterlerinin kullanımı, Olisort tarafından seçme/ayıklama anahtarları için kullanılan **leg** ifadesinin özel tanımına bağlıdır. Karşılık gelen anahtar kendisiyle kıyaslanan alana göre ''küçük'', ''eşit'' veya ''büyük''se, leg aracılığıyla seçme/ayıklamanın işletimi tanımlanabilir. Böylelikle karşılaştırma karakterleri kullanarak kısmi veya genel karşılaştırmalar yapılabilir.

Örneğin aşağıdaki tabloda gösterilen türde kodlar içeren bir imalat parçası kütüğümüz olduğunu varsayalım.

Yalnızca montaj kodu 80'e eşit parçaları ele alarak A sınıfı parçalar arasında kısmi bir değerlendirme yapmak istiyoruz. Bu durumda aşağıdaki gibi bir seçme/ayıklama anahtarı kullanabiliriz.

80= = = = = =A

Parça Kütüğü Özeti

Montaj kodu	Parça kodu	Parça sınıfı
94	543678	B
67	553478	A
80	887690	A
88	113425	D
80	934400	C
80	000789	A
56	722990	B

Verilen parça kodları içinden seçme/ayıklama anahtarımızı kullanarak, aşağıdakiler seçilecektir:

80-887690-A
80-000789-A

Daha fazla seçme/ayıklama anahtarının katılması AND ya da OR gibi bağlantı ögelerinin kullanımını gerektirir. Seçmenin yapılabilmesi için, AND ile bağlanan ike seçme/ayıklama anahtarının her ikisinin birden verilere uyması gerekir. Bu anahtarlar OR ile bağlanmışsa, seçmenin yapılabilmesi için anahtarlardan birinin uyması yeterlidir.

Örneğin, az önceki tabloda gösterilen parçalar arasında seçme yaparken

80= = = = = =A OR 80= = = = = =C

anahtar bileşimini kullanıyorsak aşağıdaki parça kodlarını elde ederiz.

80-887690-A
80-984400-C
80-000789-A

Fakat yine aynı tabloda gösterilen parça kodları üzerinde,

80= = = = = =A AND 56= = = = = =B

anahtar bileşimini kullanırsak, hiçbir parça kodunu elde edemeyiz.

Olisort, sıralanmış iki giriş kütüğünün tek bir çıkış kütüğü halinde eklemlenmesini (merge) veya bir giriş kütüğünün bir diğerinin sonuna katılmasını (append) sağlar. Böylece çıkış olarak, aynı anda hem birinci hem ikinci giriş kütüğünün veri kayıtlarını içeren, tek bir kütük elde edilir.

İki giriş kütüğünün eklemlenmesini yapmak için, bu iki kütüğü sıralamakta, kullanılacak sıralama anahtarını belirlemek şarttır.

Bir giriş kütüğünü diğerinin ardına eklemek için sıralama anahtarları tanımlamak gerekmez. İkinci giriş kütüğü otomatik olarak birincinin sonuna eklenir. Olisort, çalışması sırasında toplam beş kütüğe kadar kullanılabilir, bunlar içinde aşağıdaki üçü şu işleri görür:

— INPUT 1: Paketin kullanımı için mutlaka gereken kütük
— INPUT 2: Eklemleme ve/veya sona ekleme (append) işlevi kullanılmak istendiğinde başvurulur.
-OUTPUT: Sıralama, eklemleme sona ekleme seçme/ayıklama çıkış kütüğüdür. Çıkış girişle aynı adı taşıyorsa, yazılma kendiliğinden giriş kütüğünün üzerine yapılır.

İşletim Türleri: Olisort'u işletmek için kullanılabilecek iki yol vardır:

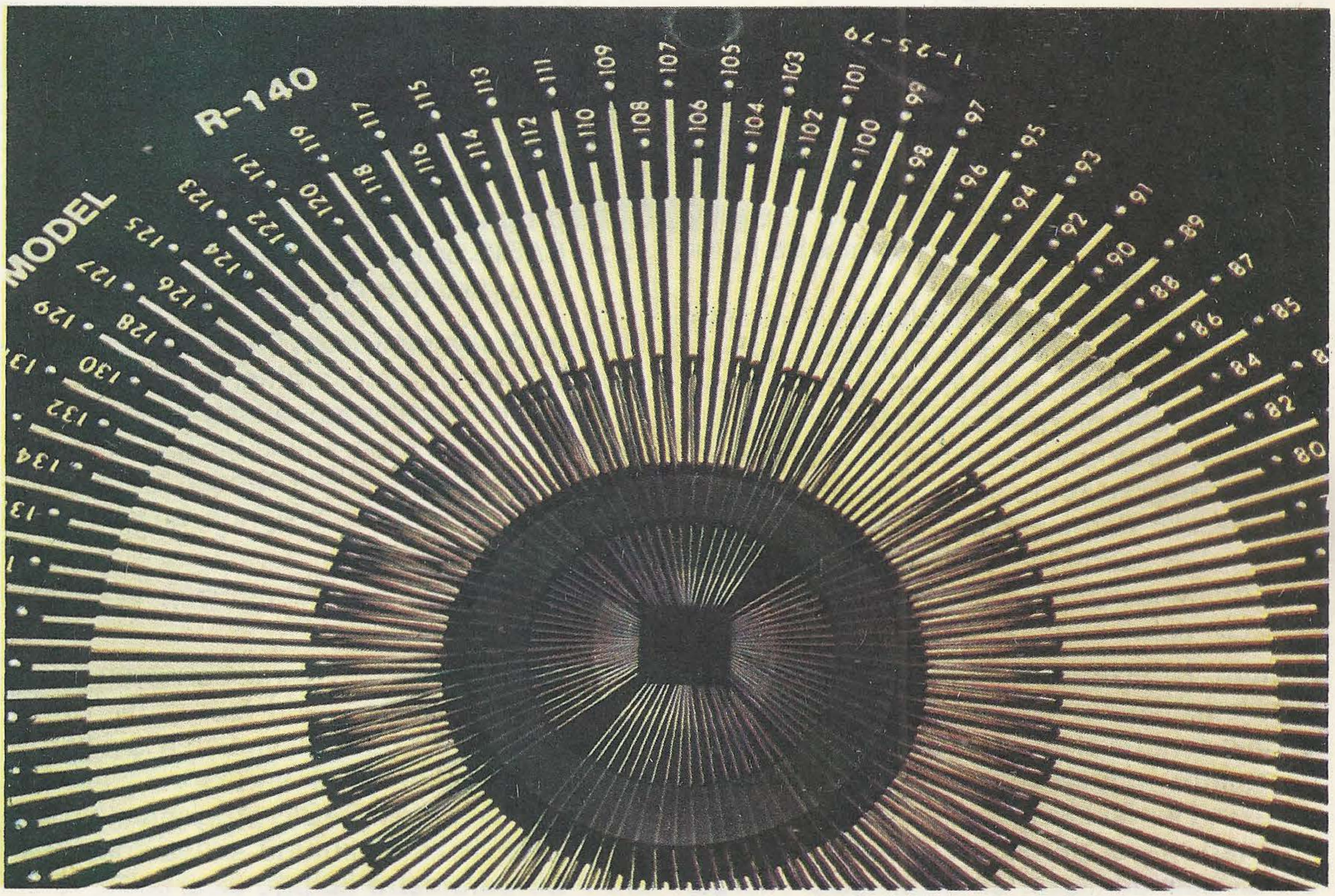
— parametrelerin bir BASIC programının güdüm karakter zincirine entegre edilmesi
— kendi aralarında etkileşen parametreler elde etmek için Olisort kullanıcı programına başvurduktan sonra, Olisort'un çalıştırılması

Her iki durumda da kullanılan parametrelerin yazı düzeni aynıdır.

Parametreler, Olisort'a şu iki biçimde aktarılabilir:

— Disket üzerinde saklanan (aynı sıralama

Bir tümleşik devrenin bağlantıları



programının bir defadan fazla işletilmesi istendiğinde yararlı ve gerekli) bir parametre kütüğünün yardımıyla. Parametre kütüğü, Olisort kullanıcı programı kullanılarak elde edilebilir.

— OLISORT.CMD$ adlı bir BASIC değişkeni olan bir güdüm karakter zinciri yardımıyla. Bu güdüm karakter zinciri iki yolla elde edilebilir: BASIC-çeviricisi veya Olisortx kullanarak.

Sonuncu yöntemle güdüm karakter zinciri disket üzerinde korunabilir.

Bir parametre kütüğünün kullanılıyor olmasının, BASIC programına kısa bir güdüm karakter zincirini dışlamadığına dikkat edilmelidir. Bu güdüm karakter zinciri çok az sayıda parametre içerir ve bunların çoğunu da parametre kütüğünden alır.

Olisort'a iletilen parametrelerden biri istenen işlev tarzını belirler. Dört Olisort konumu vardır.

Konum 0: Bir kütüğün sıralanmasına bir kütüğün veri kıyatlarının seçilmesine ya da ayıklanmasına veya aynı anda hem sıralama hem de seçme/ayıklama işleminin yapılmasına yarar. Sıralama seçme veya sıralama ve seçme/ayıklama için kullanılan parametrelerin, disket üzerindeki bir parametre kütüğünde önceden hazır bulunması gerekir.

Örneğin kodlarına göre sıralanmış stok mallarını içeren bir ana kütüğümüz ve iki ayrı kaynaktan gelen yeni malları içeren iki ek kütüğümüz varsa, yeni kütüklerin sıralamasını şöyle yapabiliriz:

1 / Olisort aracılığıyla, ek kütüklerin sıralamasını yapmak için bir parametre kütüğünün oluşturulması

2 / İki işlem kütüğünün sıralamasını yapmak için Olisort'u iki kez çağıran (her seferinde aynı parametre kütüğünü kullanarak) bir BASIC programının yazılması. Bunun için, bir güdüm karakter zincirinin hazırlanması ve konum 0'ın tanımlanması gereklidir. Parametrelerin büyük bir bölümü disket üzerindeki parametre kütüğünde bulunacağından, bu güdüm karakter zinciri çok kısa olacaktır.

Konum 1: Sıralanmış iki kütüğün bir kütükte eklemlenmesini yapar veya bir kütüğü diğerinin sonuna ekler. Yine disket üzerinde bir parametre kütüğünün oluşturulmuş olması gerekir. Her iki kütük sıralanmış ve sıralama anahtarı bilgisi parametre kütüğünde kesin olarak belirlenmişse, bir eklemleme (merge) yapılacaktır. Parametre kütüğünde sıralama anahtarı kesin olarak belirlenmemişse, iki kütük ardarda eklenecektir.

Örneğin daha önce sözünü ettiğimiz ek kütükleri ve ana kütüğü ele alırsak, ana kütüğün güncelleştirilmesi şu şekilde yapılır:

1 / İki ek kütüğün sıralamasını yapmak üzere oluşturulmuş parametre kütüğü kullanılarak, Konum 1'i kesin olarak belirleyen bir güdüm karakter zinciri içeren bir BASIC programı yazılır.

İki giriş kütüğünün sıralanmış ek kütüklerimiz olduğunu varsayalım.

2 / BASIC programı işletildiği zaman iki ek kütük bir tek kütük halinde eklemlenmiş (merge) olacaktır.

3 / Aynı parametre kütüğünü kullanarak Konum 1'i kesin olarak belirleyen bir güdüm karakter zincirini içeren bir BASIC programı yazalım. Bu kez giriş kütüklerimiz eklemlenmiş (merge) ek işlem kütükleri ve ana kütükten oluşur.

4 / BASIC programı işletilince, güncelleştirilmiş bir ana kütük elde edilecektir.

Konum 2: Konum 0'la aynı işlevleri yerine getirir, fakat tüm parametreler bir BASIC programı üzerine yazılı bir güdüm karakter zinciri aracılığıyla belirlenir.

Örneğin, yukarıdaki ek kütüklerden birini şöyle sıralayabiliriz:

1 / Konum 2'yi kesin olarak belirleyen bir güdüm karakter zinciri içeren bir BASIC programı yazarız.

2 / BASIC programının işletilmesinden sonra sıralanmış bir ek kütük elde edilir.

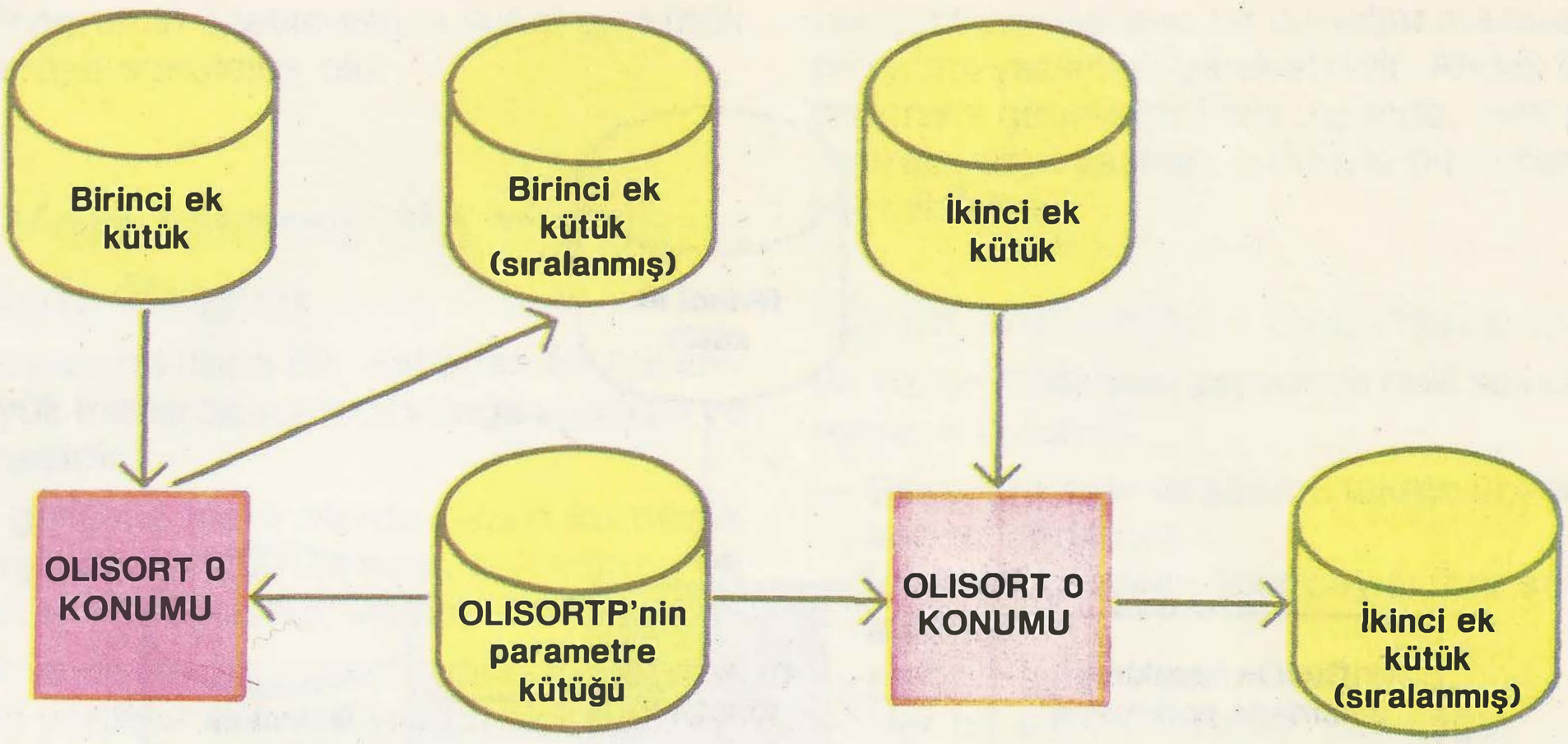
Konum 3: Aynen Konum 2'ye karşılık gelir; fakat diskette güdüm karakter zincirince belirlenen parametrelerden oluşan bir kütük üretilir. Bir BASIC programına bir güdüm karakter zinciri yazıldığında ve bundan bir parametre kütüğü üretmek istendiğinde, bu yol çok yararlıdır.

Örneğin, iki işlem kütüğümüzün sıralaması şöyle gerçekleştirilebilir:

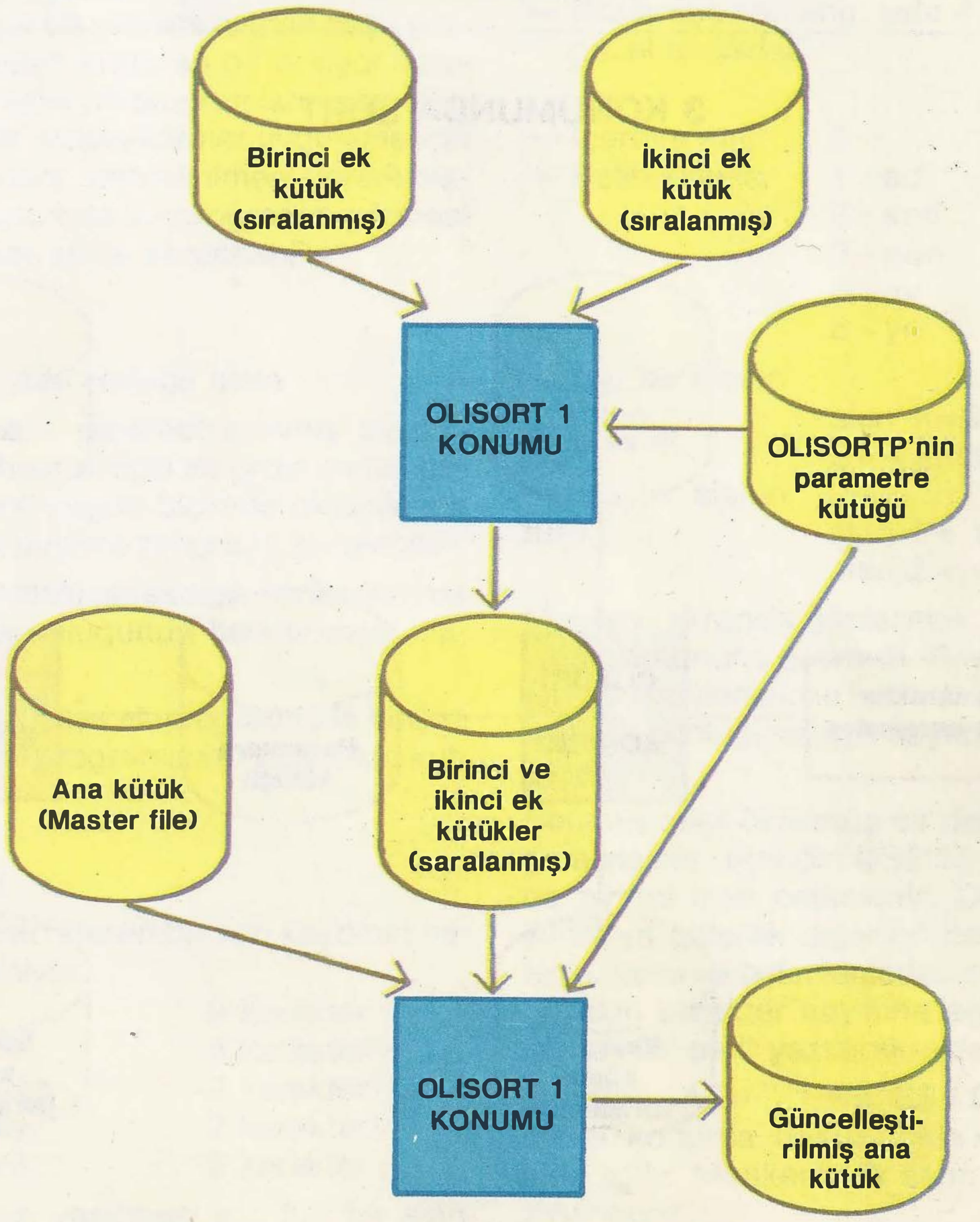
1 / Konum 3'ü kesin olarak belirleyen bir güdüm karakter zinciri içeren bir BASIC programı yazılır. Bu sıralama için gerekli giriş kütüğü ilk ek kütüktür.

2 / 1. maddedeki BASIC programının işletilmesiyle ilk ek kütüğü sıraya sokulur ve bir parametre kütüğü elde edilir.

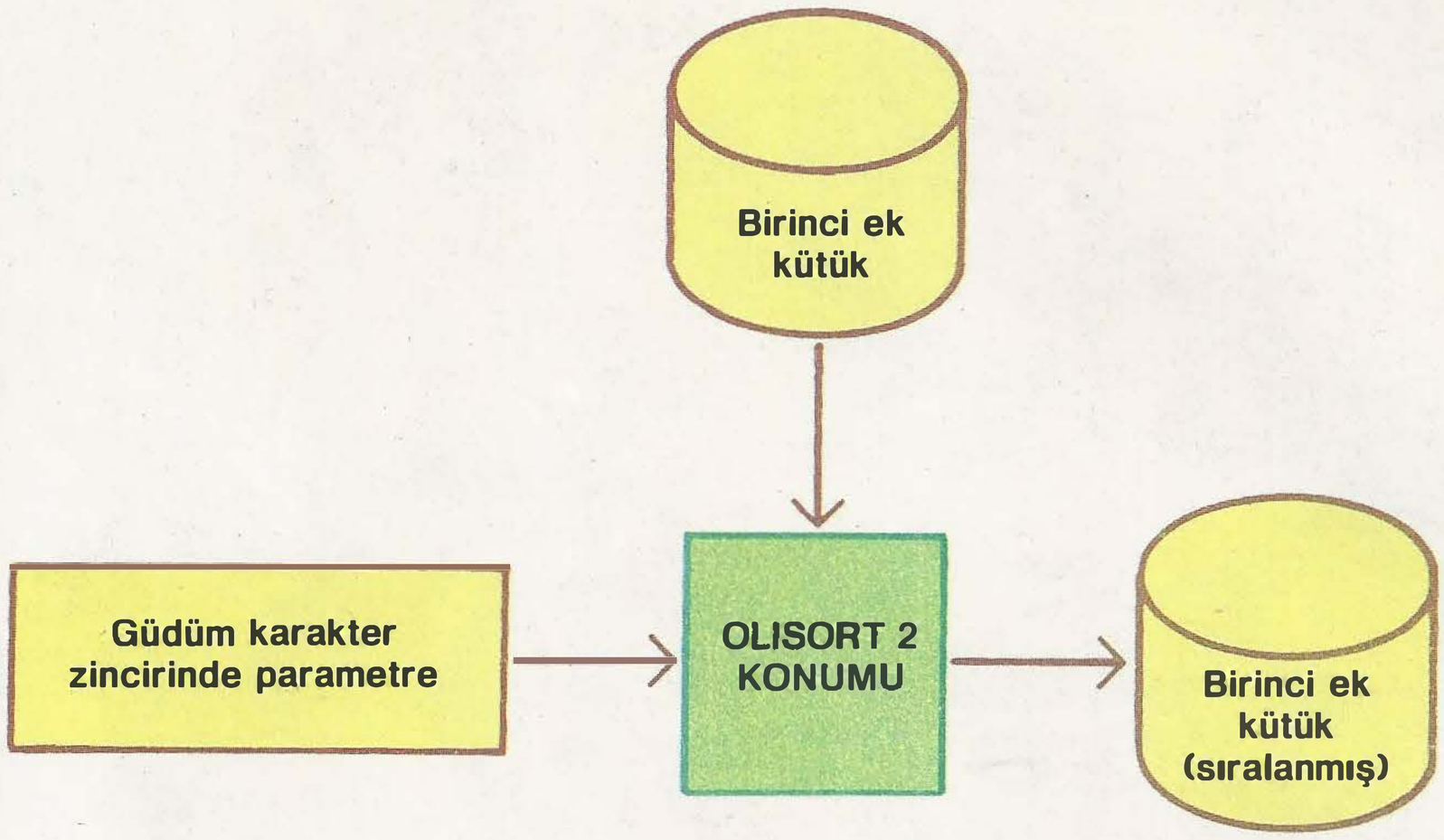
0 KONUMUNDA SORT



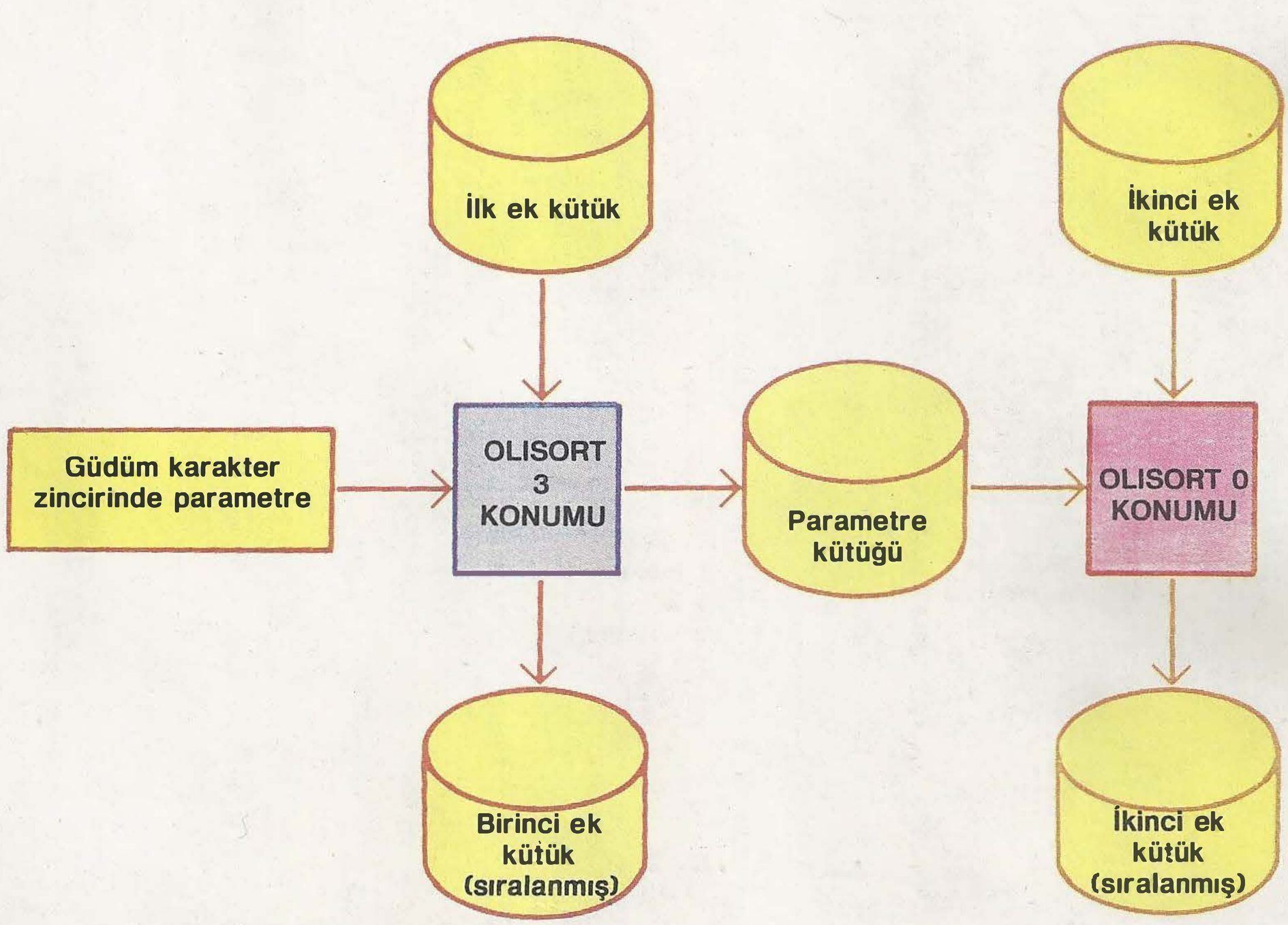
1 KONUMUNDA MERGE



2 KONUMUNDA SORT



3 KONUMUNDA SORT



3 / Bundan sonra Konum 1'i daha net açıklayan bir güdüm karakter zinciri içeren bir başka BASIC programı yazılır. Bu kez giriş kütüğü ikinci ek kütüktür.
4 / Programın işletilmesiyle ikinci ek kütük sıraya sokulmuş olur.

Bir Veri İşleme Yönteminin Geliştirilmesi

Bilgisayarların daha sık rastlanan bir kullanımı büyük miktarda verinin belleğe alınması ve işlenmesidir.

Daha gelişmiş makinelerde bazen karmaşık veri arşivlerinin hazırlanması, saklanması ve işletimi için programlar vardır.

Küçük ve kişisel bilgisayarlarda önceden hazır olan yazılımın daha az yaygın biçimleri vardır ve kullanımı ileri derecede kişiselleştirmeyi gerektirebilir. Bu durumlarda belli bir kullanıma yönelik özel bir program yazmak yerinde olur. Bu son çözüme yönelindiğinde her yeni kullanım için bir yazılımı baştan başa yazma zorunluğundan kurtaran bir ortayol bulunabilir. Çözüm esas alt-programların parametrelendirilmesidir. Böylelikle her uygulama için bazı parameterlerin değiştirilmesi yeterli olacaktır. Çoğu durumda bir programın görmesi gereken işlemler şöyle sıralanabilir:

— Veri girişi
— İşleme
— Disket üzerinde belleğe alma

Her bir işlev için parametrelenmiş bir alt-program hazırlayarak ilgili alt-programları görülecek işlevlere uygun biçimde eklemleyen herhangi bir uygulama programı kurulabilir:

İlk adım, ele alınması gerekecek verilerden her birinin tür ve uzunluğunun belirlenmesi olacaktır.

Bu parametreler, esas alt-programlarla birlikte belleğe alınıp alt-programlarını yazmakta kullanılacaktır.

Giriş Evresi

Aşağıdaki alanları içeren bir veri kaydının hazırlanması isteniyor:

Ad		9 karakter
Kod		4 karakter
Tarih	Gün	2 karakter
	Ay	2 karakter
	Yıl	2 karakter

Verilerin girişinin yapılması için her bir alan için uzunluk ve tür denetimini (örneğin tarihlerin sayısal olması gerekir) öngören bir ekran maskesi kurmak gerekecektir. Programa farklı türden verilerde girilecekse, o zaman her veri türü için **ad hoc** bir denetim maskesi altprogramı yazılması gerekecektir. Ancak ilk altprogramı genelleştirilmiş biçimde, yani parametreleyerek yazmak suretiyle bu durumdan kaçınılabilir.

Ekran Maskesinin Hazırlanması

Bir ekran maskenin yapısında belli sayıda parametre bulunur:

— Başlangıç satır ve sütunu (ekran üzerindeki konumlandırma)
— İçermesi gereken satır sayısı (her satır bir alandır)
— Her bir satırın betimlenmesi
— Her bir alanın karakter uzunluğu
— Alanın türü

Buna göre bir önceki örnek şöyle hazırlanmalıdır:

— Başlangıç konumu: satır 4, sütun 6 (gelişigüzel değerler)

— İçerdiği satır: 5
— Betimlemeler:
1 - ad
2 - kod
3 - gün
4 - ay
5 - yıl
— Her bir alanın uzunluğu:
alan 1 = 9 karakter
alan 2 = 4 karakter
— Her bir alanın türü:
alan 1 = alfabetik
alan 2 = sayısal vb.

Maskeyi ekranda göstermek için imleci başlangıç sütununa getirmek (6 ve 4), betimlemeleri görüntülemek ve herbirinin yanına her bir alanın uzunluğuna eşit sayıda nokta koymak gerekir.

Betimlemeler ölçülmüş bir değişkende belleğe alınabilir, örneğin DS$(5); DS$(1) yazılınca birinci alan betimlenir. DS$(2) yazılınca ikinci ve giderek diğerleri betimlenmiş olur. Noktalama ve betimlemeden sonra alanda öngörülen karakter sayısına eşit sayıda nokta içeren bir şerit yazılarak elde edilebilir.

Örneğin, A$ = '','' ve B$ = STRING$(9,A$) olarak konursa, dokuz nokta içeren B$ şeridi elde edilir. Maskenin ilk satırı şu komutla hazırlanabilir:

```
PRINT DS$(1) + '' '' + B$
```

DS$(1) değişkeni betimlemeyi içerir; B$ ise noktalamayı içerir ve betimleme bölümü ile maskenin diğer bölümleri arasındaki aranın açılmasına yarar.

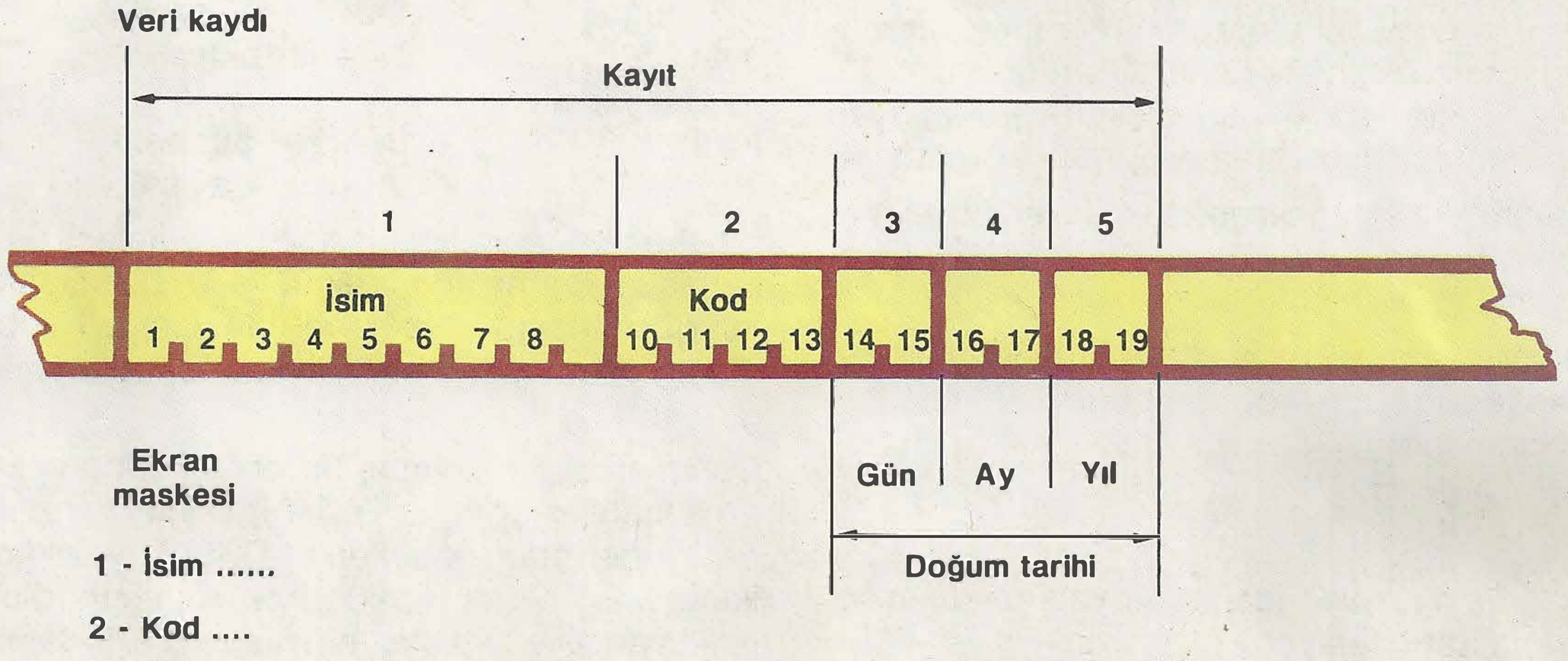
Maske hazırlama alt-programına verileri sunmak için en kolay yol bir dizi DATA ve READ komutunun kullanımıdır. Örneğin daha önce ele aldığımız betimlemeleri DS$ değişkenine (dizey) aktarmak istediğimizde komutlar şunlardır:

```
10 RESTORE 30
20 FOR I=1 TO 5:READ DS$(1):NEXT I
30 DATA ''I- AD''
40 DATA ''2- KOD''
50 DATA ''3- GÜN''
60 DATA ''4- AY''
70 DATA ''5- YIL''
```

Belleğe alma alt-programında herbir alanın uzunluğundan başka konumunu da belirlemek gerekir: Böylece ad alanı (bkz. aşağıdaki grafik) 1'den 9'a kadar baytı kapsar, kod alanı 10'dan 13'e kadar baytı kapsar vb. Bu değerler de DATA'lar ile verilebilir. Aşağıdaki listelerde şu adlar uygun görülmüştür:

```
BB  (I) = I alanının başlangıç baytı
SB  (I) = I alanının bitiş baytı
TUR (I) = alan I'nın türü
```

BİR SİCİL KÜTÜĞÜNÜN VE EKRAN MASKESİNİN SAKLANMASI



Ekran maskesi

1 - İsim .....

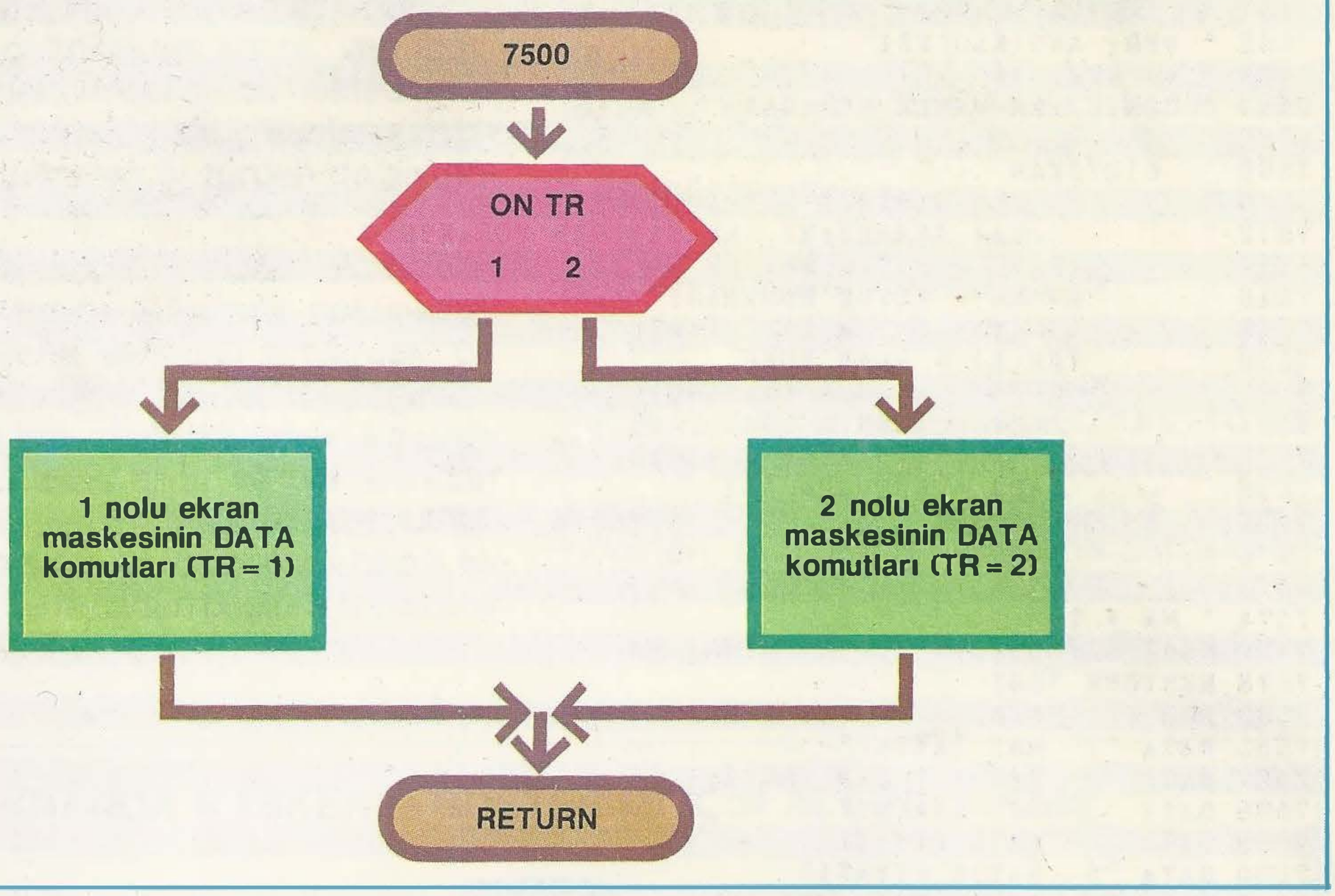
2 - Kod ....

3 - Gün ...

4 - Ay ...

5 - Yıl ...

ALAN UZUNLUKLARINI VE TANIMLARINI BELİRLEME ALT-PROGRAMI



Öbür alanların türleri de benzer biçimde belirlenir. Örneğin, aşağıdaki değerler verilirse:

TUR(I) = 1, sayısal bir değişken için,
TUR(I) = 2, harfsayısal bir değişken için,
TUR(I) = 3, yalnızca görüntü yaratan bir değişken için.

Öyleyse örneğimizdeki birinci alan 2. türden, öbürleri 1. türdendir. 3. tür ise kullanıcı eliyle düzenlenmesi gerekmeyen, yalnız görüntü yaratan alanları tanımlamak için kullanılır. Bazı uygulamalarda (tıpkı birazdan geliştireceğimiz programda olduğu gibi), veri maskelerinin ard arda ekrana getirilmesi gerekebilir. Bu durumda, DATA komutlarından gruplar oluşturulur, sonra, istenen verileri içerenler ayıklanıp kullanılır. Ana program düzeyinde, bir değişken için yaratılan bir maskenin numarası (buna örneğin MN diyelim) yasaklanır ve bu numarayla da alt-program içindeki değer verme komutları seçilir. Alt-program, aynı zamanda verilerin yazılacağı kütüğün adını ve numarasını da belirler (sırasıyla KA$ ve KN)

Sayfanın yukarısında yer alan akış şeması, içinde yalnız iki veri maskesi bulunan (MN yalnız 1 ve 2 değerlerini alabilir) bir alt-programa tekabül etmektedir. Daha genel bir kullanım için, önce DATA gruplarını ve okuma sıralarını belirlemek, sonra da çoklu dallanma komutunu düzenlemek gerekecektir. Bu alt-programın dökümü 698. sayfada verilmiştir. Her yeni DATA grubunun yapısı, 7575-7620 satırları arasında yeralan program parçasında tekrarlanmalıdır.

Birinci komut (satır 7576) kullanılacak maskenin konumunu tanımlamaktadır (SA = 2, SA = 3). Sonra, seçilen maskenin numarasına bağlı olarak program, çalışmasına devam eder (satır 7576).

Her durumda, ilk önce maskenin karakteristik değişkenleri ilk değerlenir (sıfırlanır): Yani alan sayısı (10) ve karmaşık kütüğün mantık munarası (KN). Bu örnekte veri kaydı 100 bayt uzunluktadır (bkz. satır 7612) ve bu uzunluk 7614. satırda (LN = SN(N)), kütüğün adı ile birlikte görülür. Kütüğün adı KA$ karakter zincirinde saklanır. Bu programda, birinci maskeye karşılık olan kütüğün adı MAANA'dır ve B disketinde bulunmaktadır.

Son olarak, 7618. satır ise, dizeyin kullanılmamış elemanlarına yine başlangıç değerleri vermektedir. Böylece 1 maskesi ile, en çok 14 alan kullanılabileceği halde yalnız on alan görüntülenmektedir. Geri kalan değişkenler, türlerine göre, N + 1'den başlayan bir döngü içinde sıfırlanır ya da boşlanır (burada N, gerçek-

TANIMLAR VE ALAN UZUNLUKLARININ YERLEŞTİRİMİ

```
7500 '
7501 '    KUTUK = VERI
7502 ' VER. ANSIKLOPEDI
7503 ' ** VERI **
7504 ' GIRIS :MN=MASKE NUMARASI
7506 '
7508 '  CIKISLAR
7510 '          SU= BIRINCI SUTUNUN POZISYONU
7512 '          SA= EKRANDAKI ILK SATIRIN POZISYONU
7514 '       SS(*) = SATIR SAYISI
7516 '       KN=MN = KUTUK NUMARASI
7518 '       ST$(*)= SATIR TANIMLARI
7520 '       TUR(*) = ALAN TURU
7522 '        BN(*)= ALANIN BASLANGICI
7524 '        SN(*)= ALANIN SONU
7526 '          KS = KARAKTER SAYISI
7528 '         AD$ = KUTUGUN ADI
7534 ON MN GOTO 7576, 7621      ' ISTENEN YAZI DUZENININ SECIMI
7570 '
7572 '
7574 ' MN = 1
7576 SU=2:SA=3:SS(MN)=10:N=SS(MN):KN=1
7578 RESTORE 7582
7580 FOR I=1 TO N:READ ST$(I):NEXT I
7582 DATA "1- MAL TANIMI"
7584 DATA "2- KATMA DEGER VERGISI"
7586 DATA "3-OLCU BIRIMI"
7588 DATA "4- ALIS FIYATI"
7590 DATA "5- SATIS FIYATI"
7592 DATA "6- GIRDI TOPLAMI"
7594 DATA "7- GIRDI TUTARI"
7596 DATA "8- CIKTI TOPLAMI"
7598 DATA "9- CIKTI TUTARI"
7600 DATA "10- ............."
7602 FOR I=1 TO N:READ TUR(I):NEXT I
7604 DATA 2!,1!,2!,2!,2!,3!,3!,3!,3!,3!
7606 FOR I= 1 TO N:READ BN(I):NEXT I
7608 DATA 1!,21!,23!,31!,38!,45!,53!,64!,72!,82!
7610 FOR I=1 TO N:READ SN(I):NEXT I
7612 DATA 20!,22!,30!,37!,44!,52!,62!,71!,81!,100!
7614 LN=SN(N):AD$="B:MAANA"
7616 '
7618 N=N+1:FOR I=N TO 14:ST$(I)="":TUR(I)=0:BN(I)=0:SN(I)=0:NEXT I
7620 RETURN
7621 ' MN=2
7622 SU=2:SA=3:SS(MN)=9:N=SS(MN):KN=2
7624 RESTORE 7628
7626 FOR I=1 TO N:READ ST$(I):NEXT I
7628 DATA "1- MAL KODU"
7630 DATA "2- AKIS TARIHI"
7632 DATA "3- GIRIS MIKTARI"
7634 DATA "4- ALIS FIYATI"
7636 DATA "5- CIKIS MIKTARI"
7638 DATA "6- SATIS FIYATI"
7640 DATA "7- ............."
7642 DATA "8- ............."
7644 DATA "9- AKTARILAN AKIS"
7646 FOR I=1 TO N:READ TUR(I):NEXT I
7648 DATA 1!,2!,2!,2!,2!,2!,2!,2!,3!
7650 FOR I=1 TO N:READ BN(I):NEXT I
7652 DATA 1!,4!,9!,17!,24!,32!,39!,65!,91!
7654 FOR I=1 TO N:READ SN(I):NEXT I
7656 DATA 3!,8!,16!,23!,31!,38!,64!,90!,91!
7658 LN=SN(N):AD$="B:MASET"     ' AD$=2.EKRANDAN VERILERIN SAKLANDIGI KUTUK
7660 '
7662 N=N+1:FOR I=N TO 14:ST$(I)=" ":TUR(I)=0:BN(I)=0:SN(I)=0:NEXT I
7664 RETURN
```

leşen okuma sayısıdır, 1 maskesi için satır 7576).

Veri maskesinin görüntülenmesi: Sözkonusu alt-program çağrıldıktan sonra, veri maskesinin ekrana getirilmesi gerekmektedir. Bu işlem iki zorluk içermektedir: Alanların hizalanması ve muhtemelen 3. türden değişkenlerin varlığı.

Bütün alanların mutlaka aynı uzunlukta olmaları gerekmez. Ama bu durumda, noktalamaları alanların hemen ardından sıralarsak, bir hizada olmayacaklardır. Bu tür bir dağınıklığa yol açmamak için, şöyle davranılır: En uzun alanın uzunluğu esas alınır ve her alandan sonra bu uzunluğa erişinceye kadar boşluk bırakılır. Noktalamaların en uzun alana bitişik olmaması için, bir de fazladan boşluk bırakılması daha uygun olacaktır.

Veri maskelerini ekrana getiren alt-programın akış şeması 699. sayfadan başlayıp, 700 ve 701. sayfalarda devam etmekedir. Bunun karşılığı olan programın dökümü 702. sayfadadır.

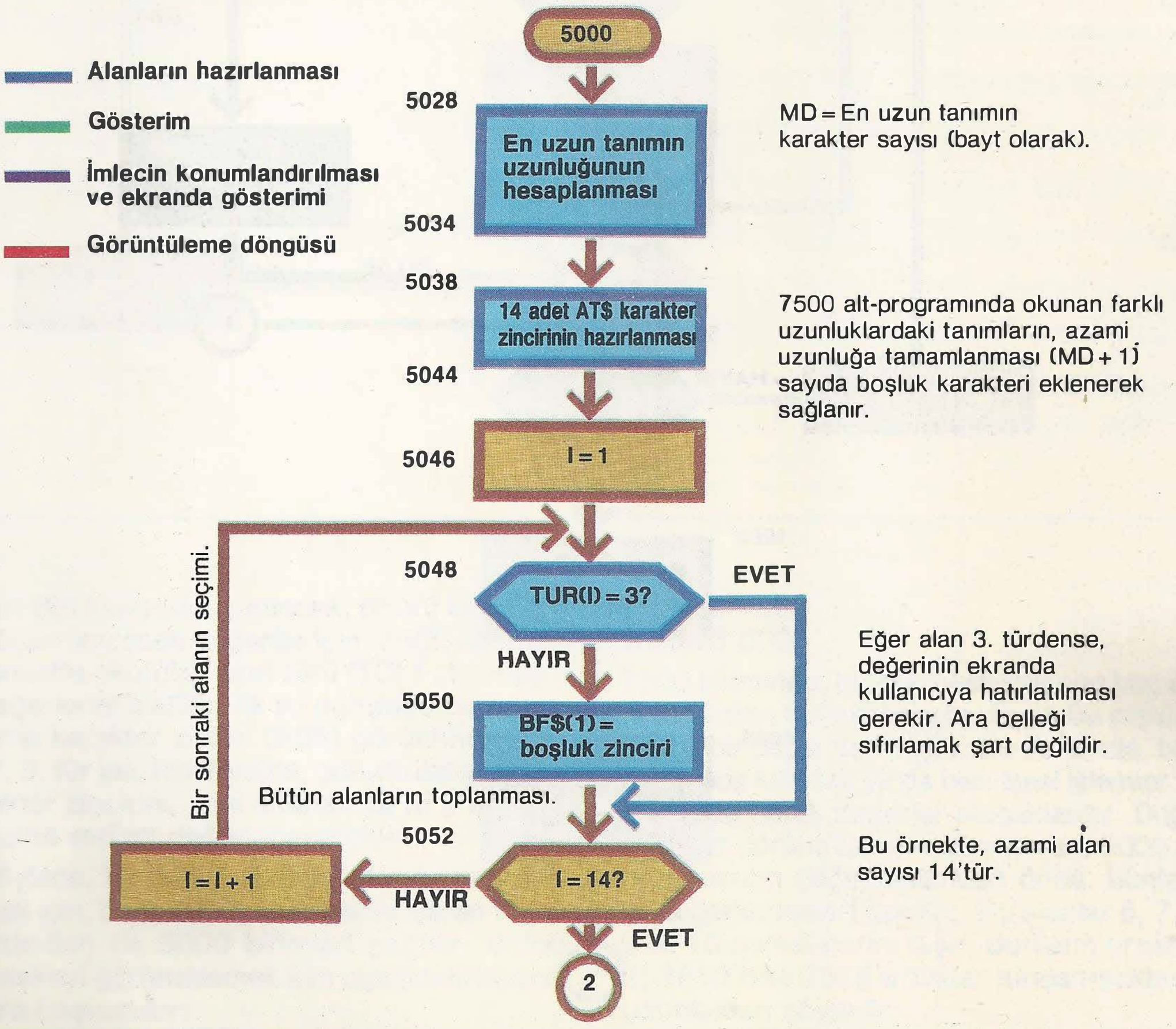
Bu alt-programda adı geçen değişkenler (özellikle de aşağıdakiler) ana programla ilk değerlenir (sıfırlanır):

BL$ = boşluk karakter zinciri
BB$ = noktalama karakter zinciri

En uzun alandaki karakter sayısı 5028'den 5034'e kadar olan komutlarla araştırılmaktadır. Bunu izleyen döngü ise (5070-5110 arası), AT$ kalıplarını tamamlayacak BC$ boşluk karakter zincirlerini oluşturur. BC$ karakter zincirleri, BL$'den N karakterlik parçalar alınarak yaratılır (5042 komutu).

Alt-programda iki sunuş biçimi öngörülmüş-

EKRANDA MASKENİN GÖSTERİMİ İÇİN ALT-PROGRAM

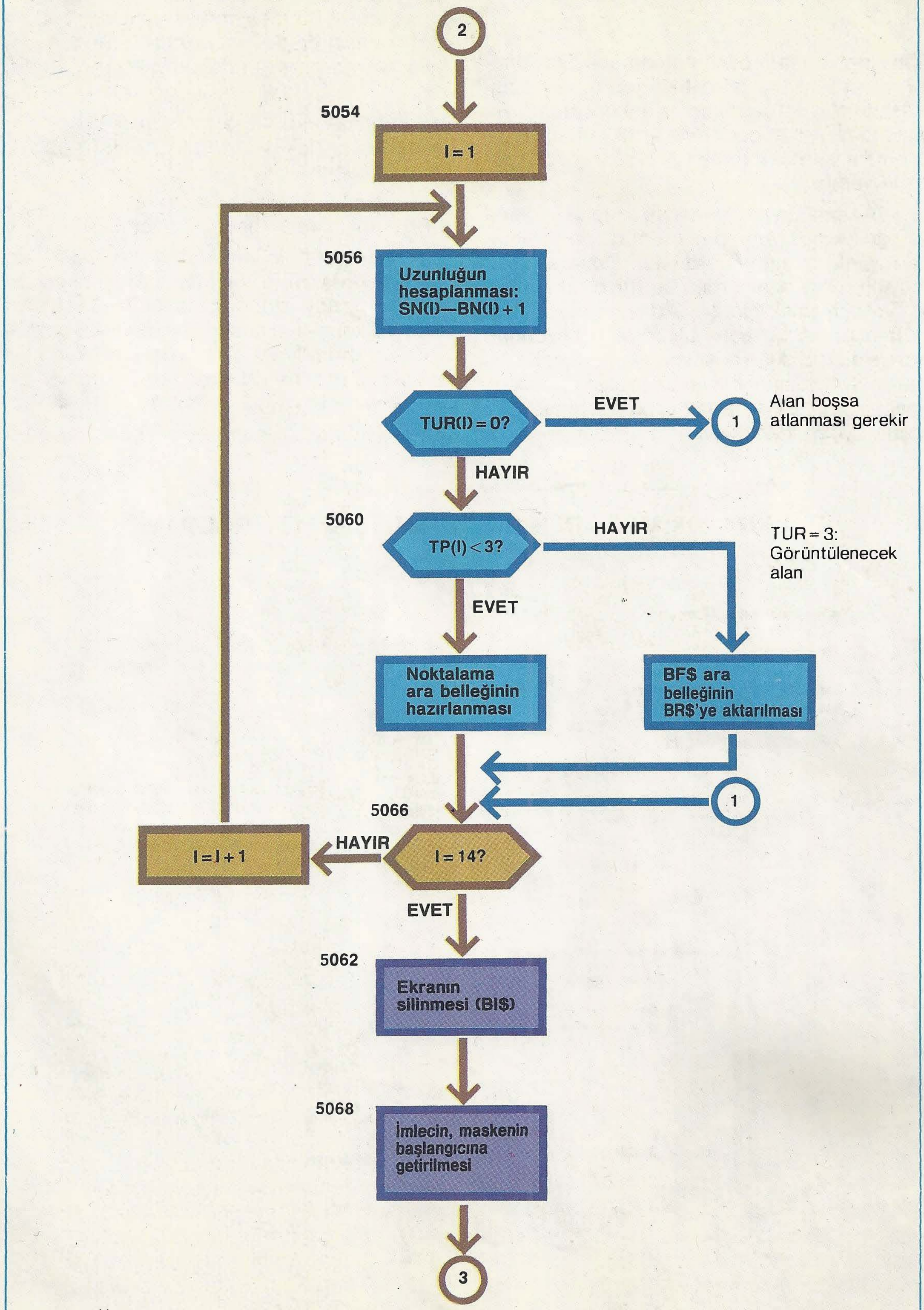


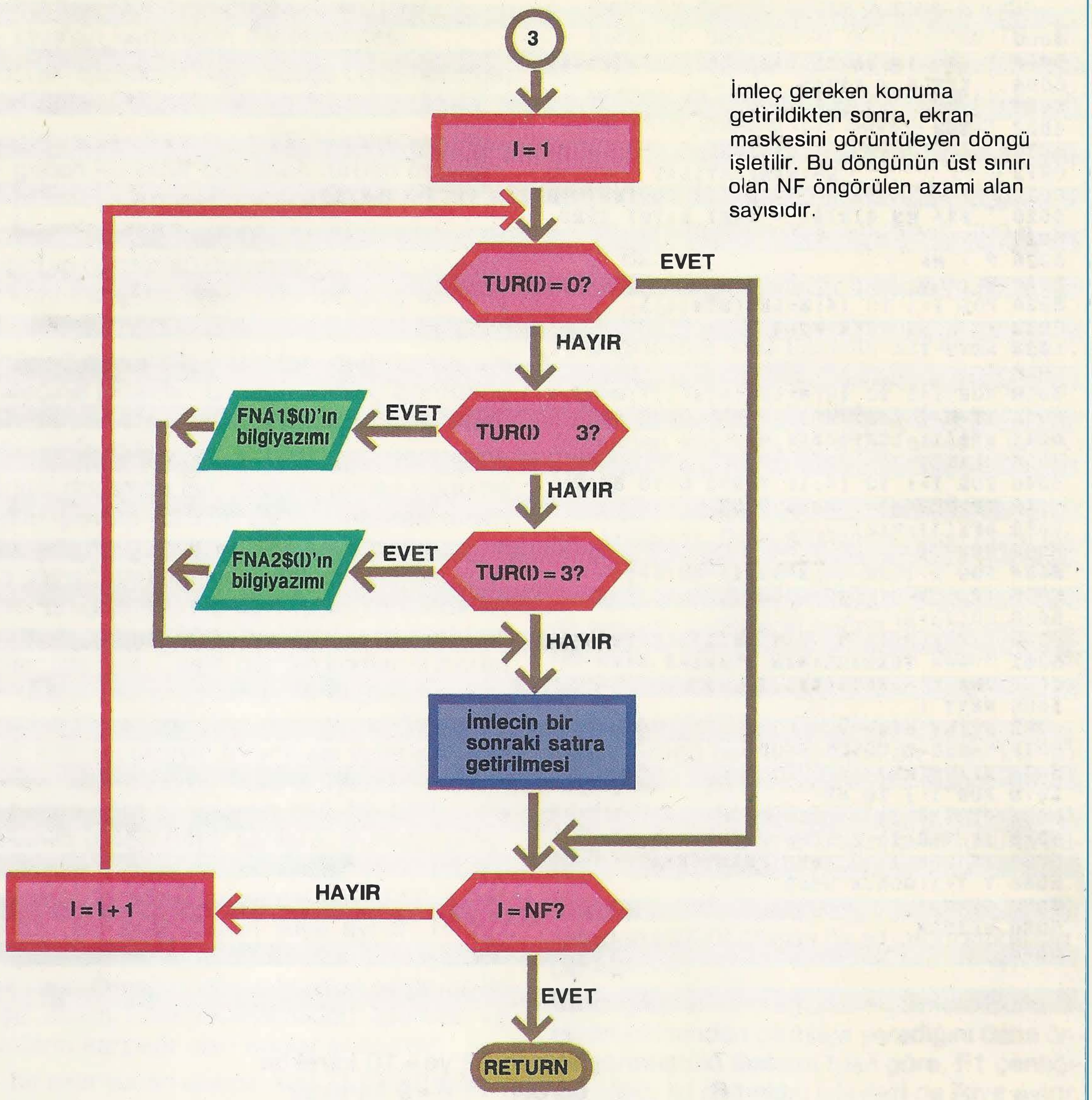
MD = En uzun tanımın karakter sayısı (bayt olarak).

7500 alt-programında okunan farklı uzunluklardaki tanımların, azami uzunluğa tamamlanması (MD + 1) sayıda boşluk karakteri eklenerek sağlanır.

Eğer alan 3. türdense, değerinin ekranda kullanıcıya hatırlatılması gerekir. Ara belleği sıfırlamak şart değildir.

Bu örnekte, azami alan sayısı 14'tür.


2
5054
I = 1
5056
Uzunluğun hesaplanması: SN(I)—BN(I) + 1
TUR(I) = 0?
EVET
1
Alan boşsa atlanması gerekir
HAYIR
5060
TP(I) < 3?
HAYIR
TUR = 3: Görüntülenecek alan
EVET
Noktalama ara belleğinin hazırlanması
BF$ ara belleğinin BR$'ye aktarılması
1
5066
I = I + 1
HAYIR
I = 14?
EVET
5062
Ekranın silinmesi (BI$)
5068
İmlecin, maskenin başlangıcına getirilmesi
3



İmleç gereken konuma getirildikten sonra, ekran maskesini görüntüleyen döngü işletilir. Bu döngünün üst sınırı olan NF öngörülen azami alan sayısıdır.

tür: Biri klavyeden girilecek, öbürü ise yalnız görüntülenecek değerler için. 7500 alt-programında okutulan alan türü (TC) 1, 2, veya 3 değerlerini alabilir. İlk iki durumda bir noktalama karakter zinciri (BB$) görüntülenecektir. 3. tür ise, tam tersine, görüntülenecek karakter zincirini, yani okunan ve BF$ ara bellegine verilen değeri içerecektir.

Böylece, bir veri maskesini ekrana getirebilmek için, önce DATA komutlarını içeren 7500, ardından da 5000 birimleri çağrılır. Birinci maskeyi görüntülemek için aşağıdaki komutlara başvurulur:

```
MN = 1
GOSUB 7500
GOSUB 5000
```

7500 biriminde, birinci maskenin son beş alanı 3. türden tanımlanmıştır. Oysa bu alanlara henüz bir değer verilmiştir. Bu durumda, bunlar ya boş kalacak ya da bazı içsel işlemler sonucunda hatalı değerler alacaklardır. Doğru değerler görüntülenmek isteniyorsa, 5000 alt-programının çağırılmasından önce, bunların ilk değerlenmeleri gerekir. Bu alanlar 6, 7, 8, 9 ve 10 numaralarını taşır. Bunların örnekleri, 7810'dan 7818'e kadar sıralanmaktadır. Uzunlukları şöyledir:

alan 6 45'den 52'ye = 8 karekter

EKRAN YAZI DÜZENLERİ

```
5000 '
5002 ' ****** EKRAN YAZI    ZENLERI*****
5004 ' KUTUK = EKRAN
5006 ' ..............
5008 ' TUR =
5010 '              1- SAYI
5012 '              2- HARF    ISAL
5014 '              3- YALNIZCA GOSTERIM(GIRIS ENGELLENMIS)
5020 ' *** MN GIRISI = VERI KAYDI TURU
5024 '
5026 P = MN
5028 MD = 0
5030 FOR I=1 TO 14:N=LEN(ST$(I))
5032 IF N>MD THEN MD=N
5034 NEXT I
5036 '
5038 FOR I=1 TO 14:N=LEN(ST$(I)):M=MD-N+1
5040 IF M=0 THEN AT$(I)=" ":GOTO 5044
5042 AT$(I)=LEFT$(BL$,M)
5044 NEXT I
5046 FOR I=1 TO 14:IF TUR=0 GOTO 5052
5048 IF TUR(I)=3 GOTO 5052
5050 BF$(I)=BL$
5052 NEXT I
5054 FOR I=1 TO 14:N=SN(I)-BN(I)+1
5056 IF TUR=0 GOTO 5066
5058 LD(I)=N
5060  IF TUR(I)<3 THEN B$(I)=LEFT$(BB$,N):BF$(I)=LEFT$(BL$,N):GOTO 5066
5062 ' *** TEKRARLANAN ALANLAR ****
5064 BR$(I)=LEFT$(BF$(I),N)
5066 NEXT I
5068 PRINT BI$
5072 X=S:Y=Z:GOSUB 6900
5074 NF=SS(MN)
5078 FOR I=1 TO NF
5080 IF TUR(I)=0 GOTO 5088
5082 IF TUR(I)<3 THEN PRINT FNA1$(I)
5084 IF TUR(I)=3 THEN PRINT FNA2$(I)
5086 Y=Y+1:GOSUB 6900
5088 NEXT I
5090 RETURN
5092 '
```

alan 7 54'den 62'ye = 10 karekter
alan 8 63'den 71'e = 9 karekter
alan 9 72'den 81'e = 10 karekter
alan 10 82'den 100'e = 19 karekter

Karakter zincirleri, kendilerine karşılık olan BF$ ara belleklerine aktarılırsa bunlar, veri maskesinin geri kalanıyla aynı zamanda ekrana gelir. Örneğin şöyle:

```
MN = 1
GOSUB 7500
BF$(9) = ''O123456789''
GOSUB 5000
```

Bu komutların yürütülmesi, ''0123456789'' karakter zincirinin, maskenin dokuzuncu satırında yer almasını sağlar.

Verilerin girilmesi. 5000 biriminden sonra, verilerin girilmesi aşamasına geçilir. Yapılacak işlemler şunlardır:

— imlecin konumlandırılması
— bir karakterin okunması (yankısız)
— okunan karakterin denetlenmesi.

Sonraki işlemlerin bağlı olduğu bu denetimde, dört durum olanaklıdır:

1- girilen karakter bir güdüm karakteridir,
2- girilen karakter beklenen türden değildir,
3- beklenen türdendir ve kabul edilir,
4- tanımlanamaz.

İkinci ve dördüncü durumlarda imleç yine aynı konuma getirilerek, yeni bir giriş beklenir (düzeltme).

Üçüncü durumda, karakter, değişkenin ara belleğine aktarılır. Örneğin, imleç dördüncü satırda bulunuyorsa, karakter de 4. ara belleğe yerleştirilir.

Birinci durumda ise, makine girilen güdüme ilişkin işlevleri yürütür. Bir veri maskesi tasarlanırken, her zaman kullanıcıya düzeltme yapma olanağı tanınmalıdır. İmlecin yer değiştirmesi programlanmış dört tuşla yönlendirilir. Bu tuşlara basıldığında, özel bir sayısal kod yollanmış olur. Çeşitli güdüm kodlarını içeren TF dizeyi ana programda ilk değerlenmiştir. Basılan tuşun bir işlevi olup olmadığını anlamak için, bu dizeyin taranması yeterlidir. Yaratılan sayı dizeyde bulunmuyorsa, sayının değerine göre programın belli bir bölümüne dallanılır (6450-6462 satıları). Burada, programlanan tuşlara verilen kodlar 8, 9, 10, 11 (yer değiştirmeler için olmak üzere) 16, 17 ve 13'tür. Bu işlev tuşları ikiye ayrılır: Birinci gruptan olanlar alt-programdan çıkmaya olanak verir. Örneğin, düşey hareket sağlayan iki tuşu alalım. Gerçekleştirdikleri işlemler ve bunların karşılığı olan kodlar şunlardır:

— bir satır yukarı atlama, kod 11 (6240 satırı),
— bir satır aşağı atlama, kod 10 (6210 satırı).

Bu tuşlarla yürütülen işlemler şöylece özetlenebilir:

— satır sayıcının (Z) artırılması ya da (hareketin yönüne göre) azaltılması,
— Y düşey koordinatının artırılması (veya azaltılması),
— karakter sayıcısının (S) sıfırlanması,
— yatay hareketle alan başına gelme (X = XO).

Örneğin, imlecin 3. satırda (Z = 3) bulunduğunu varsayalım. Bu aşamada girilen veriler 3 numaralı ara belleğe (BF$(3)) aktarılmaktadır. 11 kodunun yollanması şunlara yol açar:

— satır sayıcısının eksiltilmesi (Z = 3-1 = 2). Bundan böyle satır sayıcısı 2 numaralı ara belleği gösterecektir.
— düşey koordinatın eksiltilemsi (Y = Y-1), imlecin bir önceki satıra dömesi içindir.
— karakter sayıcısının sıfırlanması (S = 0). Böylece, imleç veriye ayrılan alanın başına yerleşir ve 2 numaralı ara belleğe hiç karakter aktarılmamış gibi devam edilir.

Öbür içsel fonksiyon tuşları da şu işlemleri gerçekleştirir:

— bir konum sağa doğru hareket, kod 9, satır 6232.
— bir konum sola doğru hareket, kod 8, satır 6222,
— doğrulama, kod 13, satır 6274 (RT tuşu).

Yatay hareket işlevinin mekanizması da düşeyinki gibidir. Ancak bu defa, satır sayıcısı değişmeden kalırken, karakter sayıcısı artırılır (ya da azaltılır).

RT tuşuna gelince: Bu tuş, verileri hizalamak için kullanılır. Giriş sırasında imleç önce alanın başına getirilir, sonra da girilen her karakter ile bir konum sağa kayar. Böylece, aslında sayısal değerlerin sağda hizalanması gerekirken, türü ne olursa olsun bütün veriler solda hizalanırlar.

6274'den 6288'e kadar olan komutlar, değişkenin türünü belirleyip TUR'un değerine göre sayısal veya harfsayısal bir alanın soluna sıfırlardan oluşan bir zincir ekleyerek hizalamayı sağlarlar. Bunlar doğrulama tuşuna basıldığı anda harekete geçer.

Nihayet imlecin konumlandırılması da, tek bir satırdan (6930) oluşan bir alt-programda gerçekleştirilir.

İkinci gruptan fonksiyonlara gelince: Bunların ekran biriminden çıkmaya yaradığını daha önce görmüştük. Basılan tuşa göre, F1 çentiğinin aldığı iki değer bu işlevleri de ikiye ayırır. Böylece, ana programda girilen verilerin doğru olduğu (tuş1, F1 = 2) ya da yanlış olup silineceği (tuş2, F1 = 3) anlaşılır. Başka işlevler tanımlamak için diğer tuşlar da programlanmak istenirse, TF dizeyine bunların kodlarını yerleştirip, girişin yapıldığı alt-programa da gereken komutların eklenmesi gerekir.

Disket üzerinde saklama aşaması

F1 = 2 koşulu doğruysa (ki bu durumda veriler doğru demektir), verilerin diskete yazılması gerekir. Bu aşamada da genel kullanıma yönelik bir alt-program tasarlanabilir.

Bir kütük üzerinde yapılabilecek dört işlem vardır:

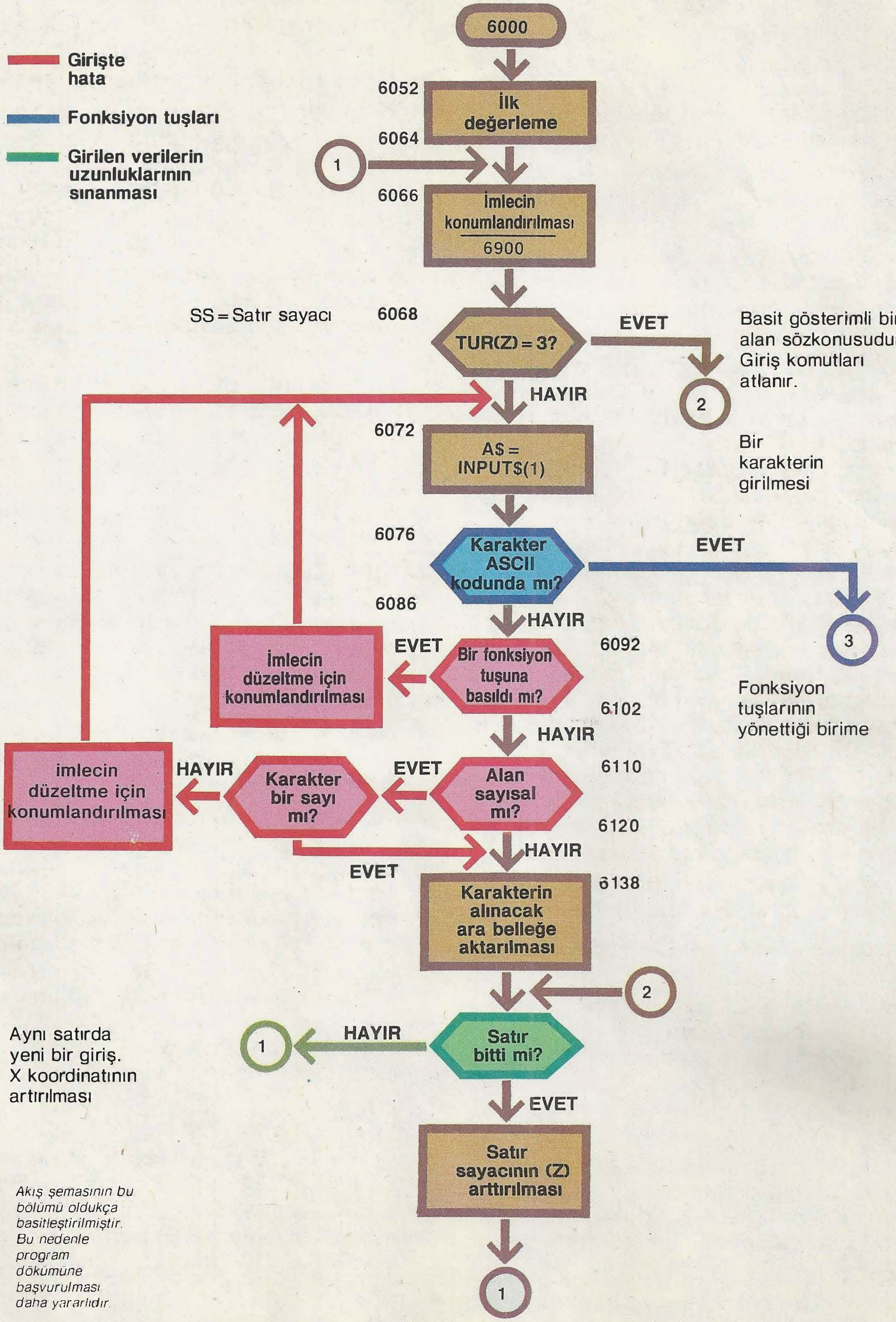
— G/Ç için bir ara belleğin tanımlanması

EKRAN MASKELERİNİ KULLANMA ALT-PROGRAMI

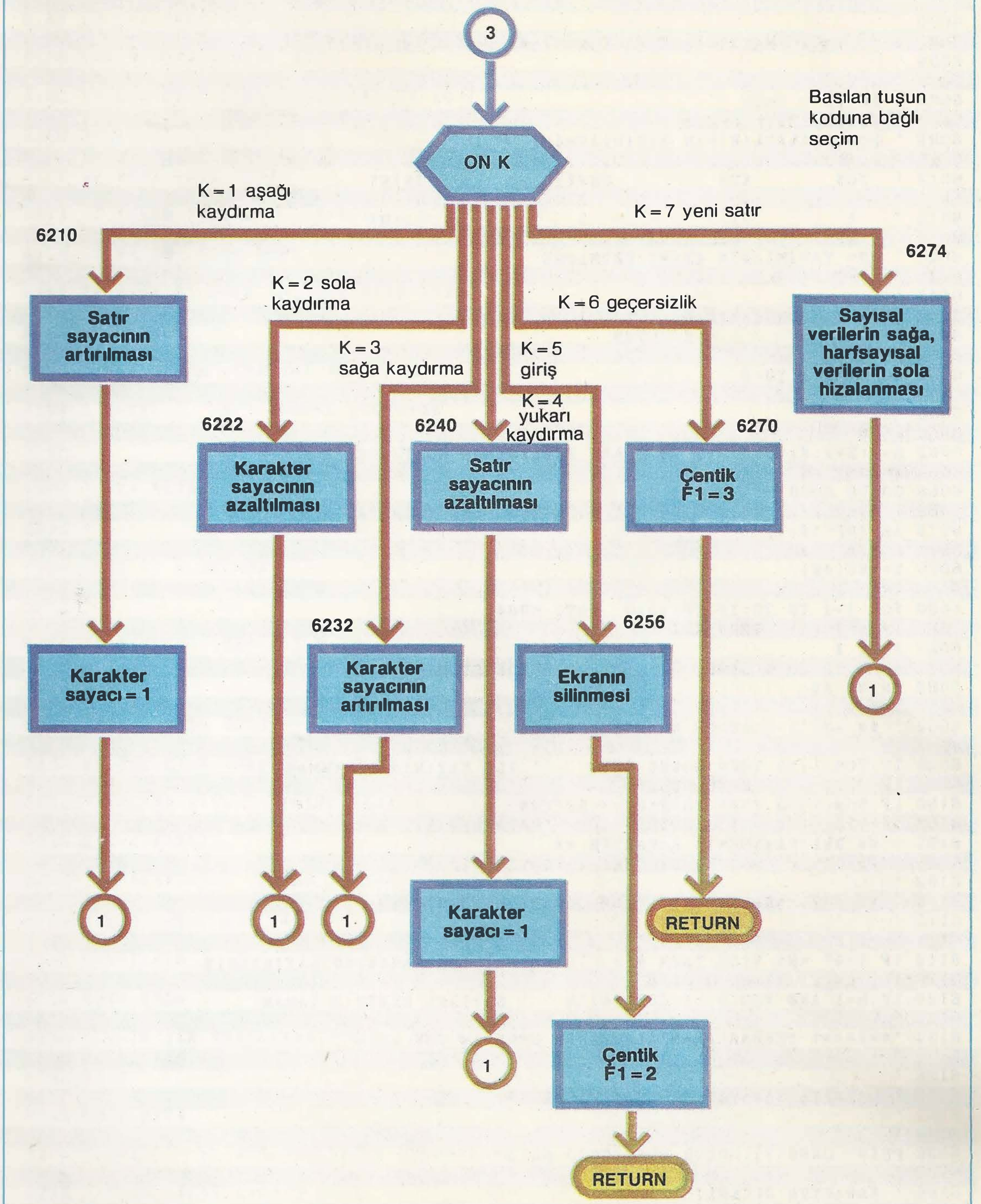
Girişte hata

Fonksiyon tuşları

Girilen verilerin uzunluklarının sınanması



Akış şemasının bu bölümü oldukça basitleştirilmiştir. Bu nedenle program dökümüne başvurulması daha yararlıdır.



Şu belirtilmelidir ki, alt-programdan çıkış, yalnızca giriş veya geçersizlik durumuna ilişkin fonksiyon tuşlarıyla gerçekleştirilebilir. Çentik, duruma göre 2 veya 3 olur. Bu değer daha sonra ana programda sınanarak, ya verinin kabulüne (çentik = 2) ya da geçersiz sayılıp yeni bir veriye geçilmesine (çentik = 3) karar vermek için kullanılır.

EKRAN MASKESİNİN YAPILMASI

```
6000 '
6002 ' *  EKRAN MASKESININ YAPILMASI **
6004 '
6005 '.....................................
6006 ' KUTUK = IMLE
6007 ' VERS. ANSIKLOPEDI
6008 ' SONUC ALANLARININ HAZIRLANMASI
6010 ' .....................................
6012 '  TUS        KOD          CENTIK          ISLEV
6014 '   1          16             2             GIRIS
6016 '   2          17             3             SILME
6038 '    TF(12) DE DEGERLER
6040 ' MD= TANIMLARIN AZAMI UZUNLUGU
6042 ' MN=CARI VERI GIRISI P
6044 '
6046 ' CARI KOORDINATLAR
6048 '                    S=SUTUN
6050 '                    Z=SATIR
6052 X0=S+MD+1:Y0=Z
6054 '
6056 '
6060 P=MN
6062 S=0:Z=1:F1=0     'S VE Z ARA BELLEGINDE GOSTERGELER
6064 X=X0:Y=Y0
6066 GOSUB 6900
6068 IF TUR(Z)=3 GOTO 6174    ' KORUNMUS ALAN ICIN SATIR ATLANIR
6072 A$=INPUT$(1)
6074 GOSUB 6900
6076 V=ASC(A$)
6078 K=0
6080 FOR I=1 TO 20:IF TF(1)=0  GOTO 6084
6082 IF V=TF(I)  THEN K=I
6084 NEXT I
6086 IF K<>0 GOTO 6190     ' *** GUDUM DENETIMI ***
6088 PRINT A$;
6090 '
6092 ' ** ASCII-KODUNUN DENETIMLERI **
6094 '
6096 IF TUR(1)=0 THEN GOSUB 6368     ' ILK SATIRIN SINANMASI
6098 IF TUR(1)=0  GOTO 6066
6100 IF TUR(Z)=0 THEN ER=2:PV=3:RETURN
6102 IF V>31 AND V<123 GOTO 6110   'KARAKTER BIR YONETICI KARAKTER DEGILDIR
6104 ' ** BELIRLENMEMIS KARAKTER **
6106 CK=3:GOSUB 6900:PRINT CHR$(7):GOTO 6072
6108 '
6110 ' ** SAYILARLA ILGILI OLUP OLMADIGININ SINANMASI
6112 '
6114 N=-1
6116 IF V>47 AND V<58 THEN N=1   ' *** N=1 ISE KARAKTER SAYISALDIR
6118 IF TUR(Z)<>1 GOTO 6138
6120 IF N=1 AND TUR(Z)=1 GOTO 6138   ' SAYISAL DENETIM TAMAM
6122 '
6124 '******* TEKRARLANAN ALANLARDA DENETIM YOK ****
6126 '
6128 '
6130 '   **HATA:SAYISAL OLMAYAN KARAKTER
6132 '
6134 '
6136 PRINT CHR$(7):GOSUB 6900:GOTO 6072
6138 '
6140 ' KARAKTER GECERLI
6142 S=S+1
6144 N=LD(Z):N8=LEN(BF$(Z))
6146 IF N8<N THEN N8=N
```

```
6148 M=S-1
6150 BV$=LEFT$(BF$(S),M)+A$
6152 IF S=1 THEN BV$=A$
6154 N2=N-S:IF N2<=0 THEN BR$=" ":GOTO 6158
6156 BR$=RIGHT$(BF$(Z),N2)
6158 BF$(Z)=BV$+BR$
6162 IF S=N THEN Z=Z+1:S=0:Y=Y+1:X=X0-1
6164 IF Z>SS(P) THEN Z=1:Y=Y0
6166 X=X+1
6168 IF TUR(Z)=0 THEN GOSUB 6358
6170 IF RF=1 GOTO 6334
6172 GOTO 6066
6174 IF V=158 THEN Z=Z-1:Y=Y-1
6176 IF Z=0 THEN Z=1:Y=Y0
6178 IF V=158  GOTO 6066
6180 Z=Z+1:Y=Y+1
6182 IF Z>SS(P) THEN Z=1:S=0:X=X0:Y=Y0
6184 IF TUR(Z)=0 THEN GOSUB 6358
6186 GOTO 6066
6188 '
6190 ' *** GUDUM DENETIMI ***
6192 '
6194 ' K=TF(12) DEKI GUDUME GOSTERGE
6195 ' KODLAR:10,8,12,11,16,17,13
6196 '          1     2   3   4   5    6       7
6198 ON K GOTO 6210,6222,6232,6240,6256,6270,6274
6200 ' *** HATA
6202 '
6204 '
6206 '
6208 ER=5:PV=5:RETURN
6210 ' *** V=10 IMLEC ASAGI
6212 '
6214 Z=Z+1:S=0:X=X0:Y=Y+1
6216 IF Z>SS(P) THEN Z=1:Y=Y0
6218 IF TUR(Z)=0 THEN GOSUB 6368
6220 GOTO 6066
6222 ' ****V=8    IMLEC SOLA
6224 IF S=0 GOTO 6072
6226 S=-1:X=X-1
6228 IF TUR(Z)=0 THEN GOSUB 6368
6230 GOTO 6066
6232 ' ****** V=12    IMLEC SAGA
6234 IF S=LD(Z) GOTO 6068
6236 S=S+1:X=X+1
6238 GOTO 6066
6240 ' ******V=11   IMLEC YUKARI
6242 IF Z=1 GOTO 6068
6244 Z=Z-2:Y=Y-1:S=0:X=X0
6246 IF TUR(Z)<>0 GOTO 6066
6248 Z=Z-1:S=0
6250 IF Z=0 THEN Z=1
6252 GOTO 6066
6254 '
6256 ' V=16 ILK FONKSIYON TUSU = GIRIS
6258 PRINT BI$
6262 F1=2
6264 RETURN
6266 GOTO 6274
6268 '
6270 ' V=17    IKINCI FONKSIYON TUSU=SILME
6272 F1=3:RETURN
6274 ' V=13  YENI SATIR
6276 IF TUR(Z)=1 GOTO 6290     ' SAYILAR SAGA ARD ARDA SIRALANIYOR
```

```
6278 AS$=LEFT$(BF$(Z),S):BF$(Z)=" "
6280 N=LD(Z)-S:B1$=LEFT$(BL$,M):PRINT B1$;:BF$(Z)=AS$+B1$
6282 Z=Z+1:S=0:X=X0:Y=Y+1:IF Z>SS(P) THEN Z=1:Y=Y0
6284 IF Z>SS(P) THEN Z=1:Y=Y0
6286 IF TUR(Z)=0 THEN GOSUB 6368
6288 GOTO 6066
6290 ' ** SAYILAR **
6291 'PRINT FNPS$;"6291 S=";S;"Z=";Z;"X=";X;"Y=";Y:S$=INPUT$(1)
6292 A$=""
6294 N=LD(Z)-S:IF N=0 GOTO 6282
6296 FOR I=1 TO N:A$=A$+"0":NEXT I
6298 IF S=0 THEN X=X0:GOSUB 6900:PRINT A$;:BF$(Z)=A$:GOTO 6282
6300 M=S
6302 A$=A$+LEFT$(BF$(Z),M(I)):X=X0:GOSUB 6900:PRINT A$;:BF$(Z)=A$
6304 GOTO 6282
6306 '
6308 '
6364 STOP
6366 ' *******ALAN ATLAMA ****
6368 '
6370 '
6372 Z=Z+1:S=0
6374 IF Z>SS(P) THEN Z=1
6376 RETURN
6900 '  IMLECIN KAYDIRILMASI
6910  ' KUTUK :IMLE-X/Y
6930 PRINT CHR$(27)+CHR$(61)+CHR$(31+Y)+CHR$(31+X);
6940 RETURN
7050 ' ********
7060 ' *********
```

— kayıtların okunması
— kayıtların yazılması
— kütüğün kapanması.

Bir kütüğü kapamak için parametresiz bir komut (CLOSE n) yeterlidir. Öyleyse, bu komutu alt-programa değil de ana programa yerleştirmek daha uygun olacaktır.

Diğer üç işlev ise tersine parametrelidir. öyleyse, 1200 disket yönetimi at-programını çağırmadan önce şu parametreleri tanımlamak gerekmektedir:

MH = istenilen işlemi belirten çentik (açma, okuma, yazma)
KA$ = işlenecek kütüğün adı
KN = bu kütüğün mantık birimi numarası
VKNE = okunacak ya da yazılacak kayıt numarası
LN = bu kaydın uzunluğu

Bunun dışında, giriş/çıkış ara belleği rolü oynayacak ve elemanları aynı numaralı kütükte bulunacak (bkz. satır 1226) harfsayısal bir dizey açmak gerekmektedir (örneğin DX$). Böylece, A disketinde, DENEY adlı, DX$(3) ara belleğine ait ve 100'er karakterlik kayıtlar içeren bir kütük açmak için, parametreler,

NR=3, L+100, KA$="B:DENEME" ve OP=1 şeklinde belirlenmelidir. Okuma ve yazma işlemlerinin bazı karmaşık yanları olduğu için, bunları biraz daha ayrıntılı incelemek istiyoruz.

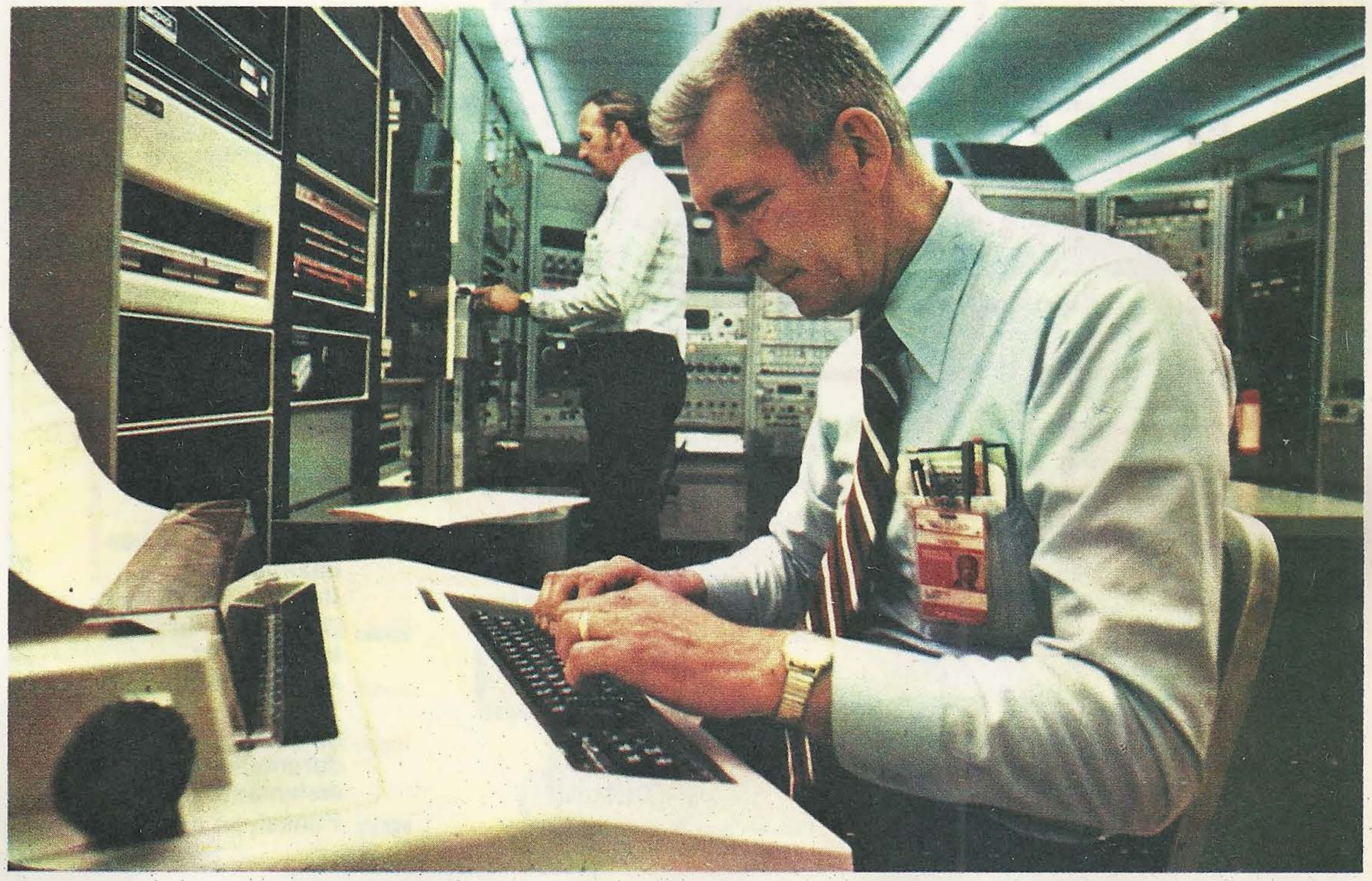
Yazma (1242-1272 arası komutlar). Şimdiye değin hep kullandığımız gibi, yazılan son kaydın numarası, hiçbir veri içermeyen birinci kayıtta saklanır.

Eğer kullanıcı, bu numaraları bizzat denetlemek durumunda olsaydı, bir veriyi okumadan veya yazmadan önce hep bir eklediği için, birinci veride ne yapmak gerektiğini bilemeyecek ve belki de yanlış yapacaktı. Buna engel olmak için bu, sayım alt-program tarafından otomatik olarak yürütülür. Bunun için de birinci kayda 0 numarası verilerek (VK=0) herhangi bir kayda ulaşmak için ardarda 1 eklenir. Tabiî, alt-programdan çıkarken VK'dan da 1 çıkarmak gerekir, çünkü alt-program bir kez daha çağrıldığında, kayıt numarası son kaldığı değerde bulunmalıdır.

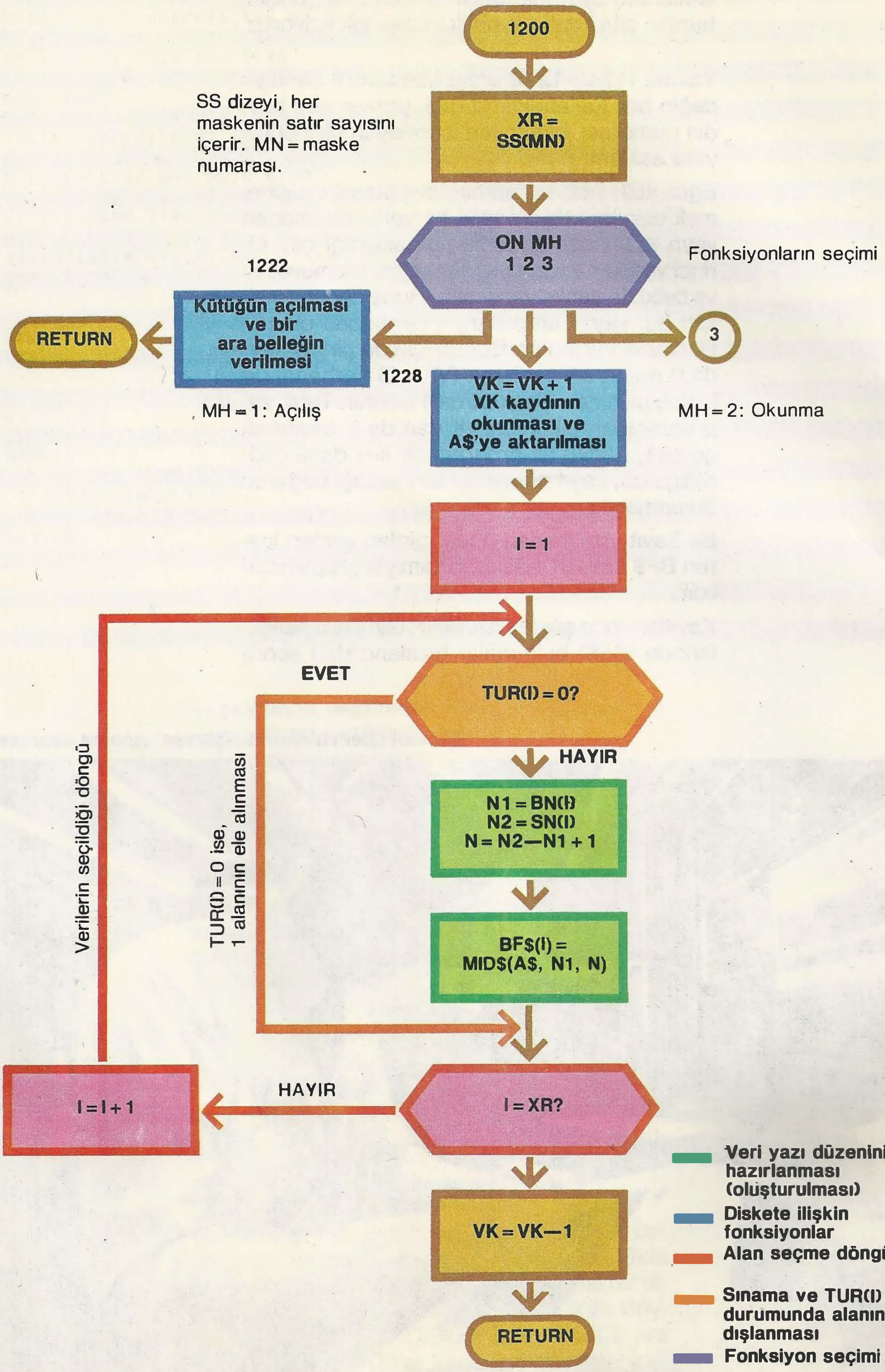
Bir kayıt yazılmadan önce, girilen verileri içeren BF$ alanları maske yardımıyla gruplanmalıdır.

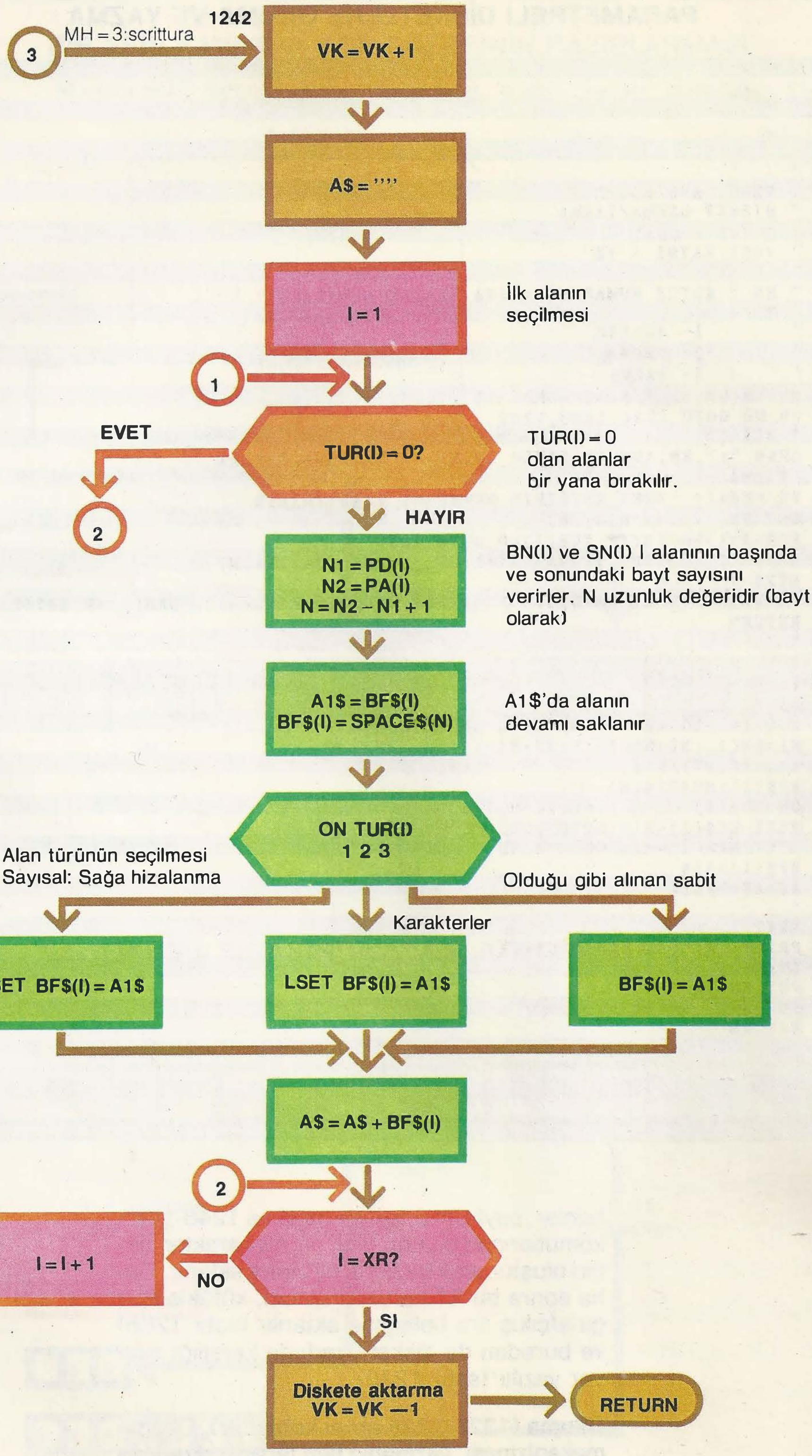
Kayıtlar önce şöyle hazırlanır: Giriş ara bellek-lerinde (BF$) bulunanlar hizalandıktan sonra

Bir veri işlem biriminde işlevsel sınama aşaması.



PARAMETRELİ DİSKET YÖNETİMİ ALT-PROGRAMI




1242
MH = 3:scrittura
3
VK = VK + I
A$ = ""
I = 1
İlk alanın seçilmesi
1
EVET
TUR(I) = 0?
TUR(I) = 0 olan alanlar bir yana bırakılır.
2
HAYIR
N1 = PD(I)
N2 = PA(I)
N = N2 – N1 + 1
BN(I) ve SN(I) I alanının başında ve sonundaki bayt sayısını verirler. N uzunluk değeridir (bayt olarak)
A1$ = BF$(I)
BF$(I) = SPACE$(N)
A1$'da alanın devamı saklanır
ON TUR(I)
1 2 3
Alan türünün seçilmesi
Sayısal: Sağa hizalanma
Karakterler
Olduğu gibi alınan sabit
RSET BF$(I) = A1$
LSET BF$(I) = A1$
BF$(I) = A1$
A$ = A$ + BF$(I)
1
2
I = I + 1
NO
I = XR?
SI
Diskete aktarma
VK = VK — 1
RETURN

PARAMETRELİ DİSKETLERE OKUMA VE YAZMA

```
1200 '
1201 ' VERS, ANSIKLOPEDI
1202 ' DISKET OKUMA/YAZMA
1203 ' KUTUK = GDAT
1204 ' VERI KAYDI = VK
1206 '
1208 ' KN = KUTUK NUMARASI = ARA BELLEK NUMARASI
1212 ' MH =
1214 '         1- ACILIS
1216 '         2- OKUMA
1218 '         3- YAZMA
1219 XR=SS(MN)    ' ALAN NUMARASI (7500)
1222 ON MH GOTO 1224,1228,1242
1224 ' ACILIS
1226 OPEN "r",KN,AD$,LN:FIELD KN,LN AS DX$(KN):RETURN
1228 'YAZMA
1229 VK =VK+1   'VERI KAYDININ NUMARASI 1 ARTIRILIR
1230 GET KN, VK:A$=DX$(KN)
1232 FOR I=1 TO XR:IF TUR(I)=0 GOTO 1236
1234 N1=BN(I): N2=SN(I):N=N2-N1+1:BF$(I)=MID$(A$,N1,N)
1236 NEXT I
1237 VK=VK-1   'VERI KAYDININ NUMARASI YENIDEN BASLANGIC DEGERINE GETIRILIR
1238 RETURN
1240 '
1242 ' YAZMA
1243 VK=VK+1
1244 A$=""
1246 FOR I=1 TO XR:IF TUR(I)=0 GOTO 1264
1248 N1=BN(I):N2=SN(I):N=N2-N1+1
1250 A1$=BF$(I):BF$(I)=""
1252 BF$(I)=SPACE$(N)
1254 ON TUR(I) GOTO 1256,1258,1260
1256 RSET BF$(I)=A1$:GOTO 1262
1258 LSET BF$(I)=A1$:GOTO 1262
1260 BF$(I)=A1$
1262 A$=A$+BF$(I)
1263 '
1264 NEXT I
1265 PRINT "A=";A$:S$=INPUT$(1)
1268 LSET DX$(NF)=A$
1270 PUT KN,VK
1271 VK=VK-1
1272 RETURN
```

bunlar, sayfa 713'teki şemada ve 1246-1263 komutlarında olduğu gibi, bir A$ karakter zinciri oluşturacak biçimde birbirine eklenir. Daha sonra bu A$ karakter zinciri, kütükle ilgili giriş/çıkış ara belleğine aktarılır (satır 1268) ve buradan da disket üzerinde karşılığı olan yer yazılır (satır 1270).

Okuma (1228-1238 arası komutlar). Okuma mekanizması, yazmanın tam tersidir. Aranılan kayıt disketten bulunup çıkartıldıktan sonra (1.

BİR VERİ KAYDININ YAZI DÜZENİNİN HAZIRLANMASI

A$ = A$ + BF$(I) komutuyla alanlar birleştirilerek, bir alan ''toplamı'' oluşturulur.

Sayaç(I)	Açıklamalar	Değerler	BN	SN
1	1. satırın açıklanması	A B C	1	5
2	2. satırın açıklanması	3 0	6	8
3	3. satırın açıklanması	X Y Z A B K L	9	15

N1 = BN(I) = 1
N2 = SN(I) = 5
N = N2—N1 + 1 = 5

A$ = BF$(1)

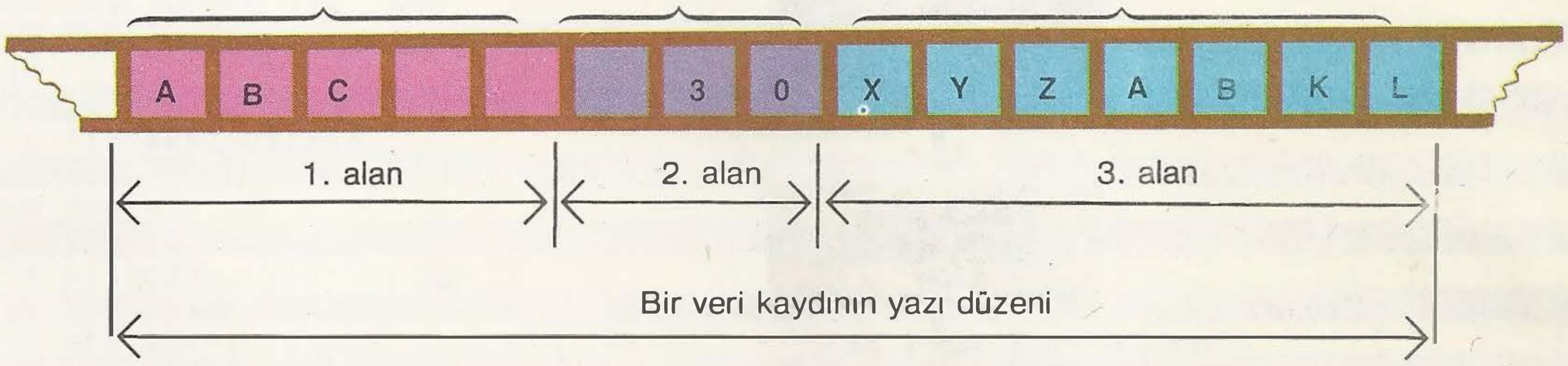
Sola hizalama

A$ = A$ + BF$(2)

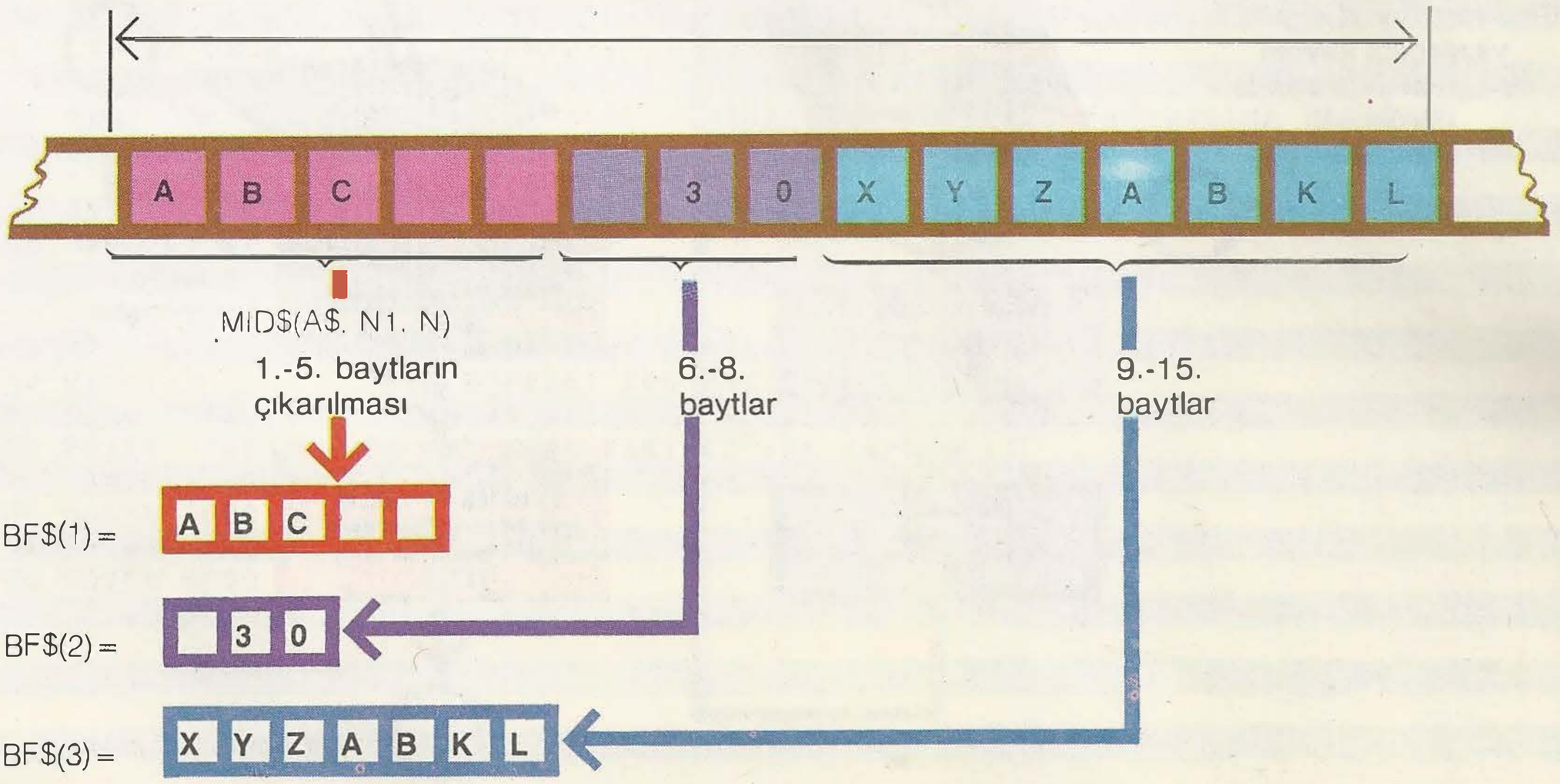
Sağa hizalama

A$ = A$ + BF$(3)

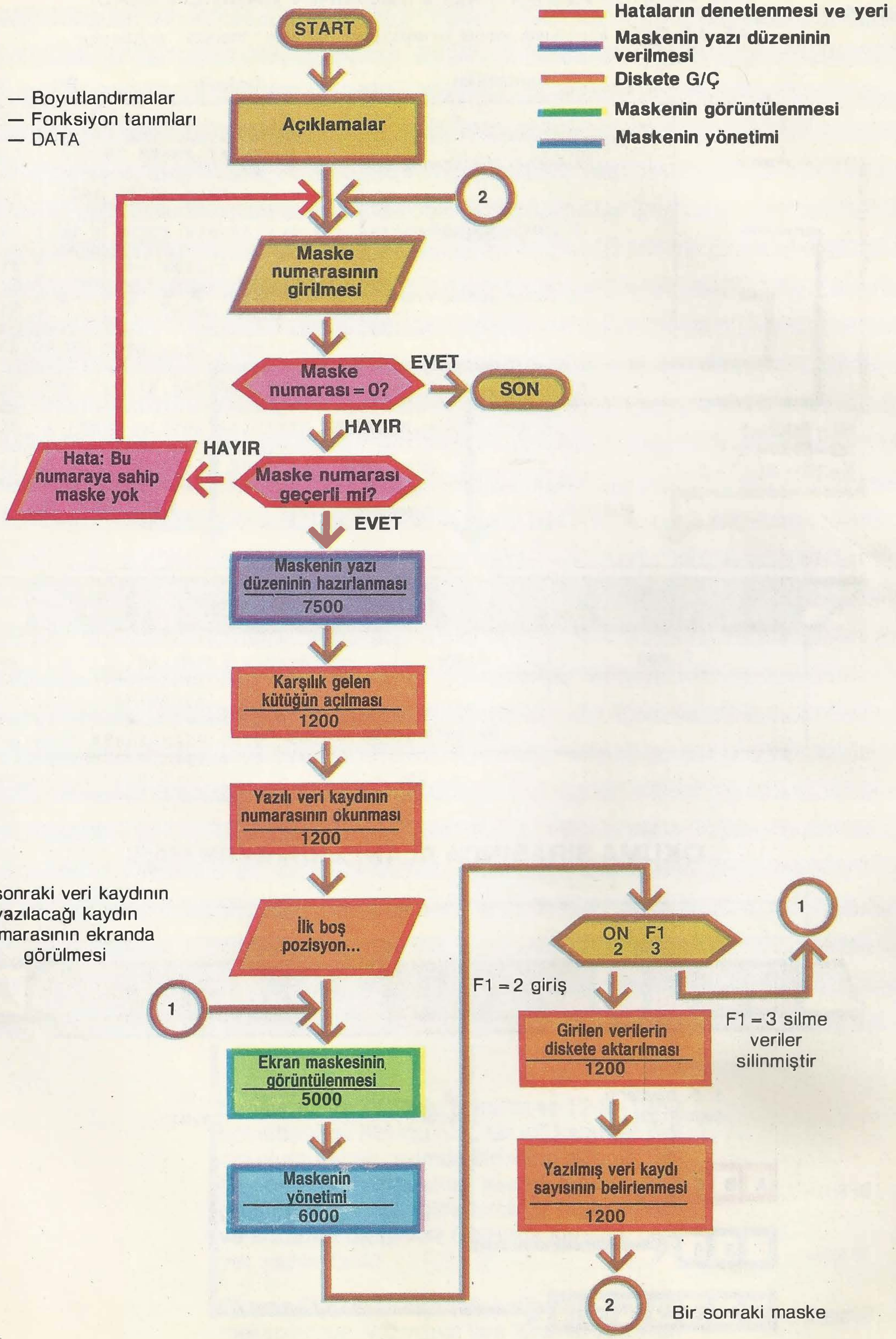
Veriler AS'ye tek tek eklenmekte, sonra diskete aktarılmaktadır.



OKUMA SIRASINDA ALANLARIN AYRILMASI



DENEME ANA PROGRAMI



DENEME ANA PROGRAMI

```
10 OPTION BASE 1
20 '  kUTUK = ANA
30 '
40 ' VERS. ANSIKLOPEDI
50 ' ****** TANIMLAR *******
60 '  TF(*)   = FONKSIYON TUSLARININ KODLARI
70 '  DX$(*) = DISKET G/C  ARA BELLEGI
80 '  TUR(*) = EKRAN MASKELERI TARAFINDAN GOSTERILEN ALANLARIN TURU
90 '  SN(*)   = ALANIN SONU
100 ' BN(*)   = ALANIN BASLANGICI
110 ' ST$(*) = FARKLI ALANLARIN ACIKLAMALARI
120 ' BS$(*) = GIRILEN VERILERIN SIFIRLANMASI ICIN ARA BELLEK
130 ' BF$(*) = VERI GIRISLERI ICIN ARA BELLEK
140 ' AT$(*) = ACIKLAMALAR ILE EKRAN MASKELERINDEKI VERI ALANLARI
150 '            ARASINDAKI AYRIM ICIN ARA BELLEK
160 ' BR$(*) = TEKRARLANAN VERILER ICIN ARA BELLEK
170 '
180 '
190 '
200 '
210 DEFINT A-V
220 DEFDBL Z
230 DIM TF(20)      ' TEK BOYUTLU DIZEY
240 FOR I=1 TO 20:READ TF(I):NEXT I
250 DATA 10!,8!,12!,11!,16!,17!,13!,0!,0!,0!,0!,0!,0!,0!,0!,0!,0!,0!,0!,0!
260 DIM DX$(4)
270 DIM TUR(14),SN(14),BN(14),LD(14),SS(5)
280 DIM ST$(14),BS$(14),AT$(14),BR$(14),CN$(14)
290 C1$="KUTUK VERILERI"
300 '
310 '
320 '
330 '*FONKSIYONLAR*
340 '
350 BL$="                                                  "
360 BB$="....................................."
370 BI$=CHR$(12)+"+"+CHR$(7)
380 DEF  FNP$=CHR$(27)+CHR$(61)+CHR$(31+20)+CHR$(31+2)   'HATA DURUMU
390 DEF FNA1$(I)=ST$(I)+AT$(I)+B$(I)     ' BAK.5000 ALT-PROGRAMI
400 DEF FNA2$(I)=ST$(I)+AT$(I)+BR$(I)    ' BAK.5000 ALT-PROGRAMI
405 GOSUB 9000
410 PRINT BI$           ' EKRANIN SILINMESI VE BIR SINYAL DUYULMASI
500 '  ** MASKENIN SECIMI ***
510 INPUT "ISTENEN MASKENIN NUMARASINI GIRINIZ",N
520 IF N=0 GOTO 900    ' SON
530 IF N<1 OR N>2 THEN PRINT  "HATA:ONGORULMEMIS MASKE"
540 IF N<1 OR N>2 THEN S$=INPUT$(1):GOTO 510
550 ' *** MASKE YAZI DUZENI ***
560 MN=N 'MN DEGISKENINDEKI MASKE NUMARASININ 7500 ALT-PROGRAMINA AKTARIMI
570 GOSUB 7500   'VERILERIN OKUNMASI
580 ' **KUTUGUN ACILMASI**
590 MH=1:GOSUB 1200
600 ' ILK VERI KAYDININ OKUNMASI
610 RE=0:MH=2
620 GOSUB 1200
630 ME=VAL(BF$(1)) 'SON DOLU VERI KAYDI
640 RE=ME+1        'BIR SONRAKI BOS POZISYON
650 PRINT "BIR SONRAKI VERI KAYDI SU POZISYONDADIR:";RE
660 PRINT "DEVAM ICIN BIR TUSA BASINIZ":S$=INPUT$(1)
670 '**EKRAN GOSTERGELERI**
680 GOSUB 5000
690 '  **** KABUL ETME *****
700 GOSUB 6000
```

```
710 '**F1 CENTIGINE GORE SECIM
720 K=F1-1 'F1 2 VEYA 3 DEGERINE SAHIPTIR VE 1 VEYA 2 DEGERINE INDIRILMELIDIR
730 ON K GOTO 740,880
740 ' * F1=2:GIRIS *
750 MH=3          ' MOD=YAZI
760 GOSUB 1200    ' VERILER RE VERI KAYDINA YAZILIYOR
770 ME=RE         ' VERI KAYDININ NUMARASI SAKLANIYOR
780 '** ILK VERI KAYDININ ICERIGININ GUNCELLESTIRILMESI **
790 RE=0  'ILK VERI KAYDI 0 ILE TANIMLANIYOR
800 '** BFğ(1) ALANININ OLUSTURULMASI **
810 N=SN(1)-BN(1)+1   'BFğ(1) DE BULUNAN KARAKTER SAYISI
820 BFğ(1)=SPACEğ(N) 'BFğ(1) ONCE BOSLUK KARAKTERIYLE DOLDURULUYOR
830 Ağ=STRğ(ME)       'VERI KAYDININ NUMARASI ASCII KARAKTERINE DONUSTURULUYOR
840 RSET BFğ(1)=Ağ    'BFğ(1) DE SAGA ARD ARDA SIRALANIYOR
850 MH=3:GOSUB 1200   'YAZI
860 GOTO 640
870 '
880 '
885 PRINT BIğ
890 GOTO 640
900 ' ** SON **
910 PRINT BIğ
920 PRINT " *** RUN END ***"
930 END
```

```
9000 ' ** FONKSIYON TUSLARININ BELIRLENMESI ICIN ALT-PROGRAM **
9010 '
9020 '  ** IMLECIN SEKLI
9040 '
9050 '
9060 '
9070 Ağ=CHRğ(27)+CHRğ(46) 'FONKSIYON TUSLARINI BELIRLEYEN
9080 '                     KOMUTUN ILK KISMI
9090 B=98                 'ILK FONKSIYON TUSUNA KARSILIK GELEN KOD
9100 I1=49                '49 VE 48 ASCII=10 HEX, CUNKU 49=1 VE 48=0
9110 I2=48                '16 TABANLI 10 ON TABANLI 16 YA KARSILIK GELIR
9120 FOR I=1 TO 8         'SEKIZ TUSUN BELIRLENMESI ICIN SAYAC
9122 ' ILK FONKSIYON TUSUNA (NR.98) ON TABANLI 16 DEGERI VERILIYOR
9130 Cğ=CHRğ(I1)+CHRğ(I2) 'I TUSUNA VERILMESI GEREKEN KOD (1...8)
9140 Dğ=CHRğ(B)           'TUSUN SECILMESI
9150 Eğ=Ağ+Dğ+Cğ 'FONKSIYON TUSLARININ BELIRLENMESI ICIN KODLARI ICEREN K.Z.
9160 PRINT Eğ             'FONKSIYON TUSLARININ BELIRLENMESI
9170 B=B+1
9180 I2=I2+1
9190 NEXT I
9200 RETURN
```

veri kaydına eklemeyi unutmadan), kütüğün G/Ç ara belleğinin içeriği A$ karakter zincirine aktarılır. Daha sonra, alanlar ara belleklere (BF$) paylaştırılır. Bu bulup çıkarma alt-programı 713. sayfanın aşağısındaki şemada gösterilmiştir.

Ana program

Çeşitli yöntemleri sınamak amacıyla bir ana program tasarladık. Ana programın akış şeması sayfa 704, dökümü ise, fonksiyon tuşlarını programlama alt-programıyla birlikte, sayfa 715 ve 716'dadır. Bu tuşların karşılığı olan kodlar 240 ve 250 DATA komutlarıyla ilk değerlenir (sıfırlanır). Bundan sonraki gelişmeler de gözönüne alınarak 20 tanesi sıralanmıştır.

Derleme

Programların tasarım ve yazılım aşamalarında çeviricili BASIC'le çalışmak çok daha elverişlidir. Buna karşılık, bir programın çevirici denetiminde çalıştırılması, şu üç nedenle daha uzun sürer:

- ☐ her çağırma komutunda (GOTO veya GOSUB), çevirici, programı, çalışmaya devam etmesi gereken yere kadar tümüyle okur.
- ☐ bir değişkenin adı geçtiğinde, çevirici bunun değerinin bulunduğu bellek alanını, değişkenler tablosundan arar.
- ☐ çevirici her komutu yerine getirmeden önce, onu makine diline çevirir. Özellikle bir döngüde, BASIC komutlarının kodları her dönüşte teker teker çözülecektir. Bir komutun çevirisi kısa sürede yapılabiliyorsa da, uzun programlarda bu durum önemli bir zaman kaybına yol açmaktadır.

Tüm bu elverişsiz yanlar, çeviricinin çalışma biçimi ile ilgilidir. Bunlar, program yazıldıktan sonra bir **derleme** ile ortadan kaldırılabilir. Derleyicinin görevi, komutları 2 tabanına, değişken adlarını da mutlak adreslere çevirmektir. Derlenmiş bir program çok daha hızlı çalışır. (4 ilâ 10 kez daha hızlı). Üstelik, derleyici daha genel bir hata ayıklayıcısı olduğu için, çeviricinin kaçırdığı bazı hataları da yakalayabilir.

Bu durum özellikle de koşullu dallanma komutlarında sözkonusudur. Atlama komutunun yolladığı satır mevcut değilse, çevirici bu durumun, ancak atlama ânında, yani koşul yerine getirildikten sonra, farkına varır (ve haber verir).

Bu durumda, yazılma sırasında,hatanın farkına varılmayabilir. Programın çalışmasından sonra ise, işleme sırasında verilere zarar verme ve hatta onları yok etme tehlikesi vardır, dolayısıyla hata anlaşıldığında iş işten geçmiş olabilir. Oysa derleme, bütün adreslerin sistematik bir biçimde denetlenmesini de içerdiğinden, bu türden hataları düzeltebilir.

718. sayfada bulunan şema bir BASIC programının yazılma, derlenme ve çalışma aşamalarını ardarda göstermektedir. Şimdi bu aşamaları ayrıntılarıyla görelim.

Hazırlık

Derleyici, yalnız ASCII kodu ile saklanmış programlar üzerinde çalışabilir. Öyleyse, her şeyden önce, ham programın bu biçimde saklandığından emin olmalıyız: Program disket üzerinde ''A'' seçeneği ile saklanmış olmalıdır. Böylece, DENEY adındaki bir program için, şöyle davranılır.

```
LOAD ''A:DENEY''
SAVE ''A:DENEY'',A
```

Bundan başka, DEFINT komut veya % işaretiyle (DEFINT I ile I% aynı anlama gelir.) Tüm döngü sayaçları tamsayı olarak tanımlanmış olmalıdır (çalışma süresini kısaltmak için).

Son olarak, derleme ve sonra da ''bağlanma'' (linkage) sırasında, çağırılacak bütün programların (sistemin olsun olmasın) A disketinde yer alması gerekir. Bunun nedeni, derleyicinin A biriminde çalışmasıdır. Bazı işlemler, ilkece öbür birimde yürütülebilirlerse de, tümünün bir birimde yürütülmesi daha uygundur. Her durumda, derlendikten sonra programları aktarmak için hiçbir engel yoktur.

Derleme işlemine ilişkin programlar şunlardır:

BASCOM.COM. Bu, BASİC dilini Assembler'a çeviren bir sistem programıdır. Ham programın derlenmiş bir versiyonunu verir.

BRUN.COM. Program tarafından çağırılan alt-programların birçoğunu içerir. Gereken bağlayıcıların yerli yerine konabilmesi için, program çalışırken hazır tutulmalıdır.

Programlar veya program parçaları arasındaki bağlayıcıların nelerden oluştuğunu biraz ilerde açıklayacağız. Şimdilik, bu terimin, ayrı ayrı derlenmiş birimleri birbirine bağlayan bir dizi işlem simgelediğini bilelim.

---

* Çeşitli programlar ya da program parçaları arasındaki ''bağ'' daha sonra açıklanacaktır. Şimdilik bunu derlenmiş program parçalarını birleştiren bir dizi işlem olarak düşünebiliriz.

DERLEME ÖNCESİ, DERLEME VE SONRASI

Komutlar:

> MBASIC

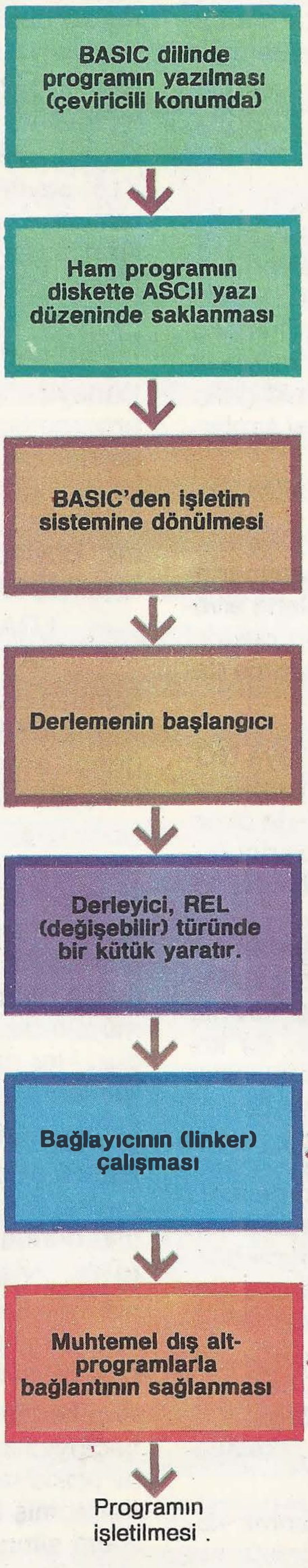
SAVE ''A:DEŅEME'',A

SYSTEM

DERLEYİCİ

BAĞLAYICI

DENEME



Bu noktada diskette DENEME.BAS adlı bir kütük oluşur.

Buradan itibaren diskette bir DENEME.REL ve bir DENEME.BAS programı bulunmaktadır.

Programın son şekli olan DENEME.COM'un diskette saklanması

- Dışardan alt-program eklenmesi
- BASIC dilinde ve ASCII formatında program
- Derleme
- Değişebilir yazı düzeninde program
- Zincirleme yükleyici
- İki tabanlı sistemde program

Diskette programın şu üç versiyonu bulunmaktadır:
DENEME.BAS = BASIC dilindeki ham program
DENEME.REL = Derlenmiş program
DENEME.COM = İki tabanlı, çalışabilir haldeki program

Sammie ve otomobil

Bir otomobilin tasarımı, her zaman dolaysızca birbiriyle bağıntılı hale getirilmeyene birçok ekonomik-teknik özelliğin en dengeli biçimde uzlaştırılmasıyla gerçekleşir. Yüksek performansa sahip bir araç yaratabilmek için, bir yandan sayıları gittikçe artan, bir sürü farklı standarda uyarken, bir yandan da sürekli gelişen bir pazarın isteklerine cevap vermek gerekir. İşte bundan ötürü, dengeyi bulmak çok zordur.

Artık tasarım çalışmasının karmaşıklığı, yalnız teknolojiden değil, tüketiciye kullanışlı bir araç sunabilmek için gerekli koşulların çokluğundan ileri gelmektedir. Bugün bir aile otomobili, basit bir ulaşım aracı olmaktan çok daha fazla şey ifade etmektedir. Bir tek üründe bir yığın öznel isteğe cevap verilmesi gerekmektedir.

İyi bir otomobil, yüksek verime ve uzay aracınınkini andırır bir sürücü mahalline sahiptir. Çekici olabilmek için az benzin yakmalı ve özellikle acemiler için rahat bir kullanım sağlamalıdır. Üretimi, bakımı ve onarımı basit ve ucuz olmalıdır. Ticari açıdan bakıldığında da satışların getirdiği kâr yeni modellerin tasarımını finanse edebilmelidir.

Yeni bir modelin tasarlanmasından, ticari hale gelmesine kadar uzun bir süre geçer. Çoğu kez, mühendisler ve pazarlama elemanları, en az on yıl sonra piyasaya çıkacak bir ürünün teknik özelliklerini öngörmek zorundadırlar. Bu öngörülerden sonra, birbirine bağlı ve oldukça karmaşık bir dizi çalışma yürütülür. Bunlar, maketin, modelin, prototipin ve kalıpların tasarımından üretim araçlarının yapılmasına kadar uzanmaktadır. Seri üretime ise, ancak bu uzun sürecin tamamlanmasından sonra geçilebilmektedir.

***BETA (Bilgisayar eşliğinde tasarım)** sistemleri ise, tasarım sürelerini önemli ölçüde kısaltıp, mukavemet, dinamik ve titreşim çözümleme tekniklerinin kulanılmasını kolaylaştırarak ürünü mükemmelleştirmekte ve dolayısıyla kullanıcının daha çok tatmin edilmesini sağlamaktadır.*

BETA sistemleri özellikle, hatalı tasarımların anında tespiti ve elenmesi konusunda yararlı olmuştur. Eskiden bu hatalar ancak prototipin yapılmasından sonra ortaya çıkmakta ve sonradan beliren sayısız sorun çok fazla para ve vakit kaybına yol açmaktaydı. Bundan başka, tasarımın son aşamaları ve deneme yöntemleri çok büyük yatırımlar gerektiriyordu.

Bugün mevcut birçok yazılım, üç boyutlu sanal maketlerin tasarımını olanaklı kılıyor. Bunlar, önce yüksek düzeyli bir dilin yardımıyla geometrik biçimler tanımlıyor. Daha sonra ise bu basit biçimler, bir bina, bir makine parçası, bir dişli, bir uçak gövdesi, bir otomobil karoserisi gibi çok daha karmaşık bir yapıyı oluşturacak biçimde uzayda biraraya getiriliyor. Bu tip sistemlerden bir kaçı: EUCLİD (Matra-Datavision), CATIA (Dassault-Systems), UNISURF (Renault-Automatismes) ve SISTRID (SNIAS)'dır.

Bir Amerikan BETA yazılımı olan SAMMIE'nin özelliği ise, bir insan oluşturup hareketlerini temsil ederek şu sıralar çok rağbet gören (ergonomi) çalışmalarına kolaylık getirilmesi. Bu olanağın daha iyi anlaşılması için, somut örnek olarak bir otomobilin tasarımını seçtik.

Çizimi yapan kişi, işe, belirleyici temel boyutları (uzunluk, genişlik, yükseklik) ve her düzey için önerilen biçimleri kapsayan bir ön çalışma yaparak başlar. Eldeki birkaç veriden hareketle, üç boyutlu bir başlangıç ön-modeli oluşturur. Bu aşamada, karoserinin eğimli yüzeyleri küçük yassı dörtgenlerin biraraya gelmesinden elde edilmiştir. Görece olarak hızlı yürütülen bu işlem, toplam tasarım süresinin yaklaşık % 30'unu kapsar. Böylece, daha henüz başlangıç verileri kullanıldığı halde, tasarımcı yeni model hakkında oldukça yaklaşık bir fikir edinebilmiştir.

Daha sonra maket, kendi etrafında döndürülerek, tüm açılar için bir perspektif çalışması yapılır. Bu çalışma, geleneksel yöntemlerle yapılacak olsaydı, resim masası üzerinde uzun uzun uğraşılmasını geretirecekti. Aracın karoserisi belirlendikten sonra, sıra sürücü mahallinin tasarımına gelir. Bu aşamada tasarımcı daha önceki modellerde kullanılan tasarım ögelerinden mümkün olduğunca yararlanmaya çalışır. Örneğin, bazı durumlarda, daha önceki bir modelin koltukları alınıp yeni karoserinin boyutlarına uydurulabilir. Böyle bir çalışma için, SAMMIE'nin veri tabanından daha önce yaratılmış olan modeli bulmak yeterli olacaktır. Sonuçta yalnız süre değil, çalışma maliyetleri de azalacaktır. Tabiî ki, bu yöntem sü-

rücü mahallinin tüm ögelerinin tasarlanması için kullanılamaz. Gösterge panosundaki göstergelerin ergonomik ilkelere uygun olarak sürücüyü yormayacak bir biçimde biraraya getirilmesi bazı zorluklar doğurur: sürücü, bir bakışta ve özel bir çaba gerektirmeksizin her an bu göstergeleri denetleyebilmelidir. Öyleyse bunlar sürücüden hem mümkün olduğu kadar uzak, hem de yüksekte yer almalıdır. Bu gereklik, doğal olarak, çevrenin görülebilmesi ve aracın dış boyutlarının getirdiği sınırlamalar ile çatışır.

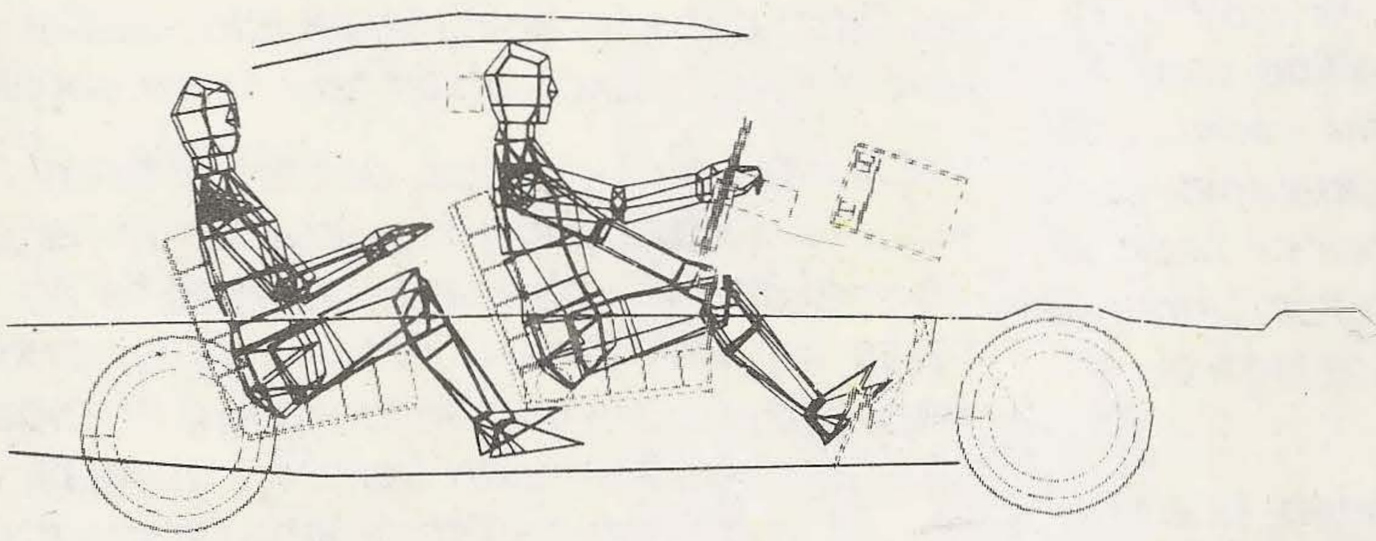
Bu tür sorunlar klasik yöntemlerle de çözülebilir ama, BETA bu iş için biçilmiş kaftandır.

SAMMIE, mekan kullanımı, kesişimler, toleranslar vb. gibi verileri gözönünde tutarak, gerçek anlamda bir ''sınırlamalar altında modelleşme'' çalışması yürütür.

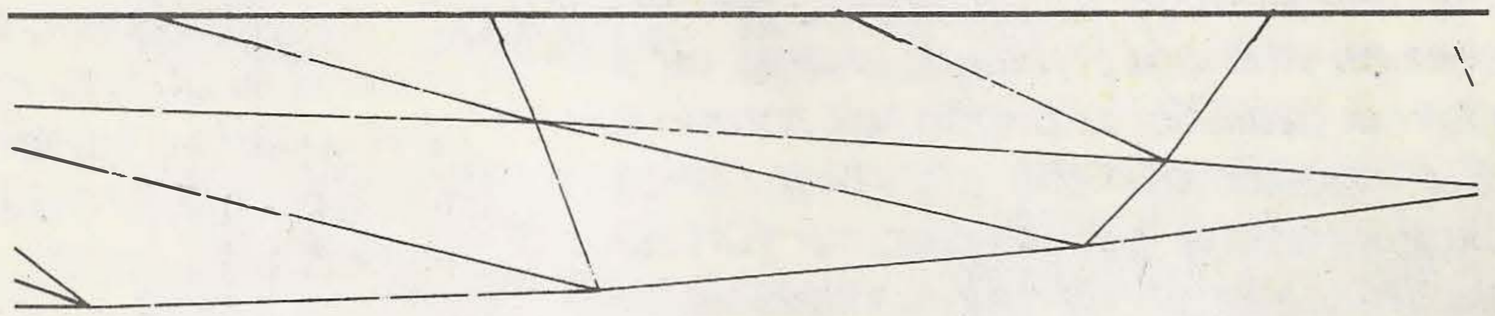
Pratikte tasarımcı, elle yapıldığında haftalar süren yaklaşık bir düzine denemeyi kısa bir sürede yapabilmektedir. Sürücü mahallinin yerleştirilmesi de tasarım süresinin % 25 kadarını almaktadır.

SAMMIE, kullanıcıyı temsil eden üç boyutlu bir insan modeli canlandırıp, onu çalışma ortamının içine yerleştirebilmektedir. Oluşturulan manken, insan vücudunun kinematik özelliklerini taşımaktadır: Organlar ve bunları birbirine bağlayan bir dizi eklem, gerçek birk işinin gövdesini ve hareketini taklit eder.

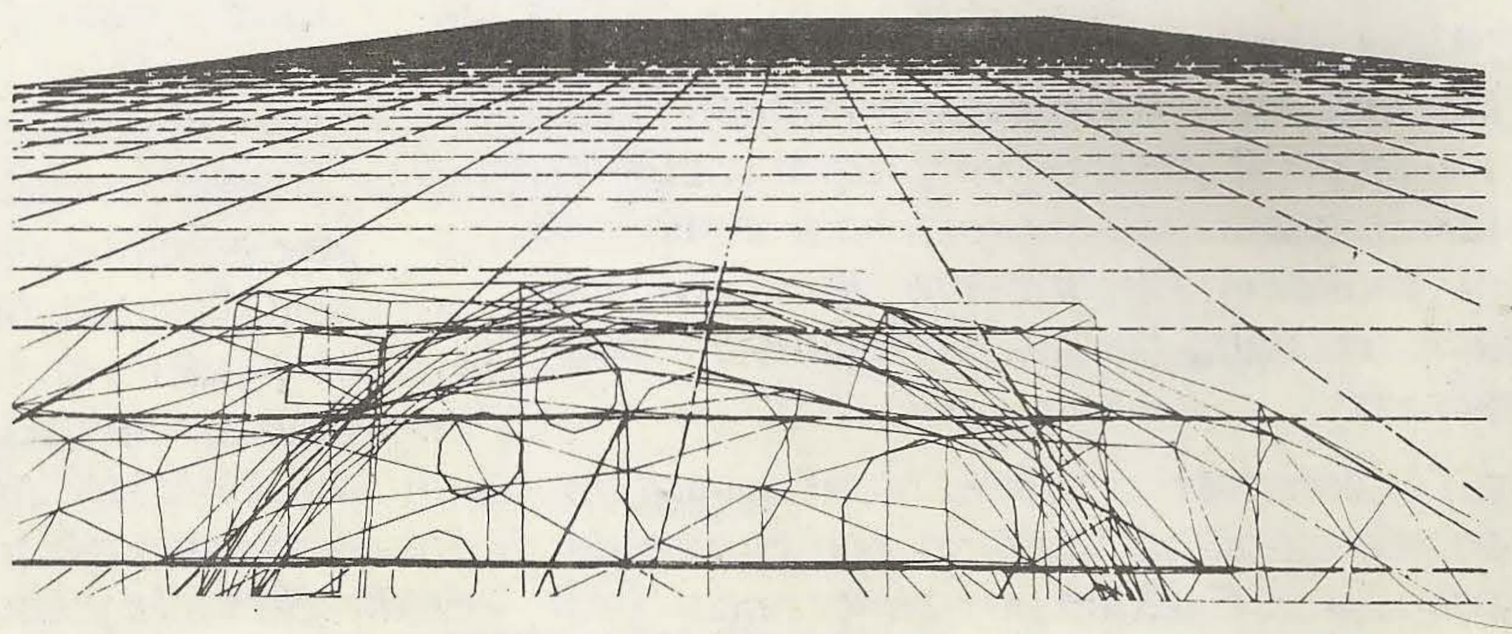
İlk bakışta bu derece katı bir modelleme çalışmasının gereksiz olduğu, aynı sonucun matematiksel denklemler yoluyla da elde edilebileceği düşünülebilir. Ancak bu durumda, görüntü yoluyla kurulan ***anında iletişim*** *ortadan kalkacaktır.*

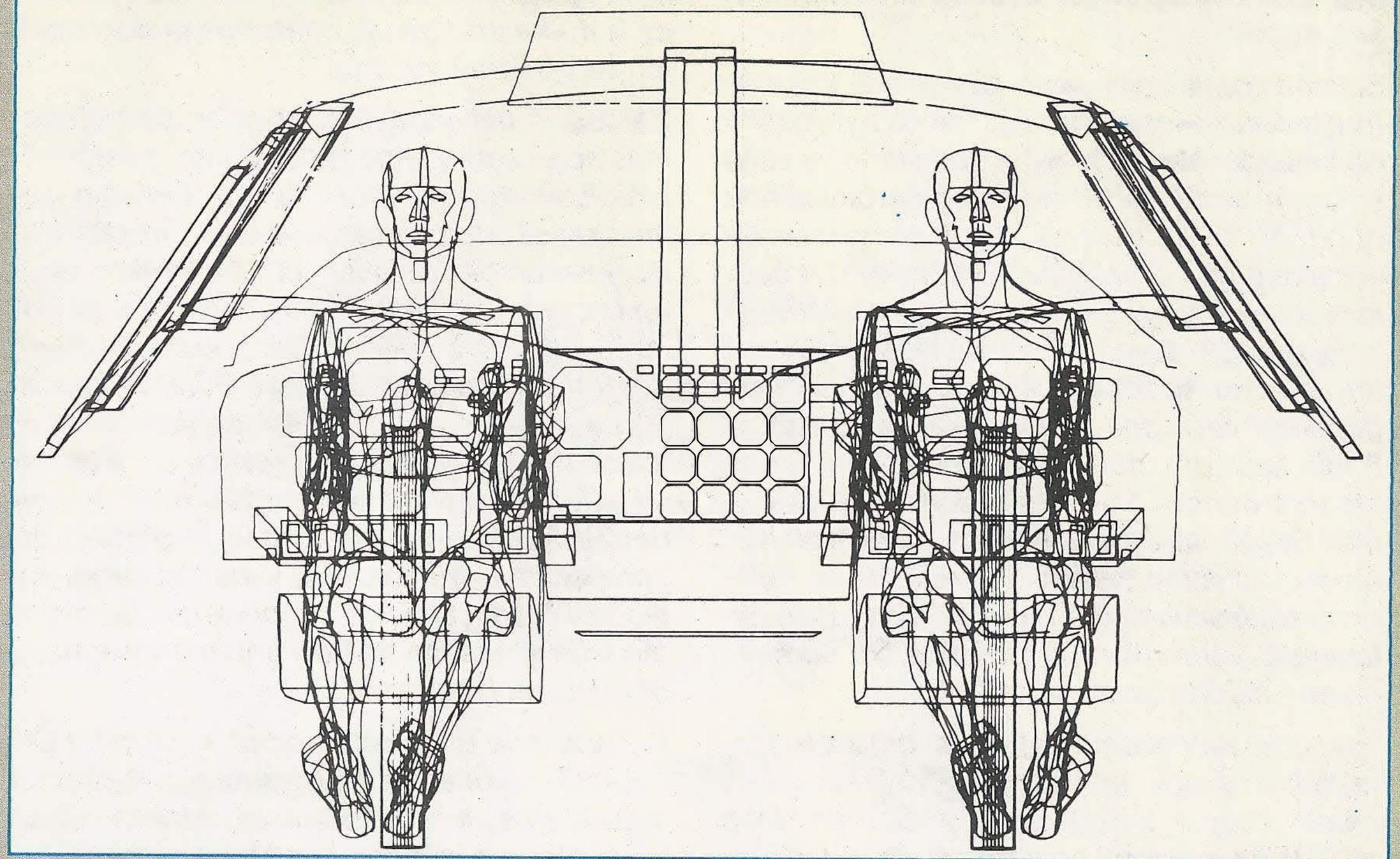


Bilgisayar destekli tasarım bugün bir arabanın çok boyutlu planının çok daha kısa bir süre içinde hazırlanmasını sağlıyor.



Bu resimler bir SAMMIE programının çalışmasından alınmıştır. Yukarda doğru oturma pozisyonları ve gereçlerin yeri. Sağda: Tasarımcı şoförün görüş alanını gözden geçiriyor.





Manken, insan vücudunun aldığı konum ve yaptığı hareket hakkında derhal bir fikir verdiği için, tasarımcı, insan ile çalışma ortamı arasındaki ilişkileri kolaylıkla gözleme şansına sahip olacaktır.

Bundan başka, SAMMIE tarafından kullanılan veri tabanı, bireyler arası farkları gözetmek amacıyla yeniden düzenlenebilir.

Mankenin tüm eklemleri hareket ettirilebilir. Bu hareketler, anormal konumları engelleyen bir denetim tablosuyla denetlenir. Tasarımcı, böylece tablo üzerinde çalışarak, kişinin yapısından ya da ortam tarafından belirlenen bir kısıtlamadan kaynaklanan etkileri görebilir.

*Modelin esnekliği, eklemleri oynatmak için verilen komutlara bağlıdır. Bir komut yollandığında, sistem buna **doğrudan** (komutla bildirilen) ve **dolaylı**, yani ilk hareketler tarafından belirlenmiş ikincil hareketlerle karşılık verir. Hareketin hassaslığı seçilen mankenin türüne bağlıdır.*

Şimdi dış yüzeyler ve iç aksamdan oluşan otomobil maketimize dönelim. Tasarımcı bu aşamada, kendini sürücünün yerine koyabilir. Bunun için, önce ileriye doğru baktığı varsayılan mankenin bakış açısını dakik bir biçimde saptar. Sonra bu açıdan ortaya çıkan görüş alanını ekrana çizdirir.

Böylece gösterge tablosuna bir gözatarak bu iş için gözün ne kadar hareket etmesi gerektiğini saptarken, bu arada yolun ne kadarının görüş alanından kaybolduğunu da belirleyebilir. Bu işlemi daha gerçekçi kılabilmek için, gerçek tablolar ekran üzerinde görüntülenebilir.

Daha sonra SIGHT (görüntü) birimi ve bir direksiyon modelinin yardımıyla gösterge tablolarının arasını açmak, görüş alanını biraz azaltmak gibi çeşitli seçenekler incelenebilir.

Sonra, aynı incelemeler, sürücünün boyu değiştirilerek sürdürülür.

Böylelikle görüş sorunu çözümlendikten sonra sürücünün hareket olanaklarının incelenmesine geçilir. REACH (ulaşma) birimi, tasarımcıya elindeki mankeni etrafındaki noktalara parmaklarıyla, avucuyla ve ayakkabısının tabanı ya da topuğuyla ulaştırma olanağı verir. Birim için gereken konum ya imlecin yardımıyla yaklaşık olarak ya da bir noktanın üç boyutlu uzayda tanımlanmasıyla kesin olarak girilir. Bu birim pratikte sürücünün kumanda

aygıtlarına erişip erişemeyeceğinin anlaşılmasını sağlar.

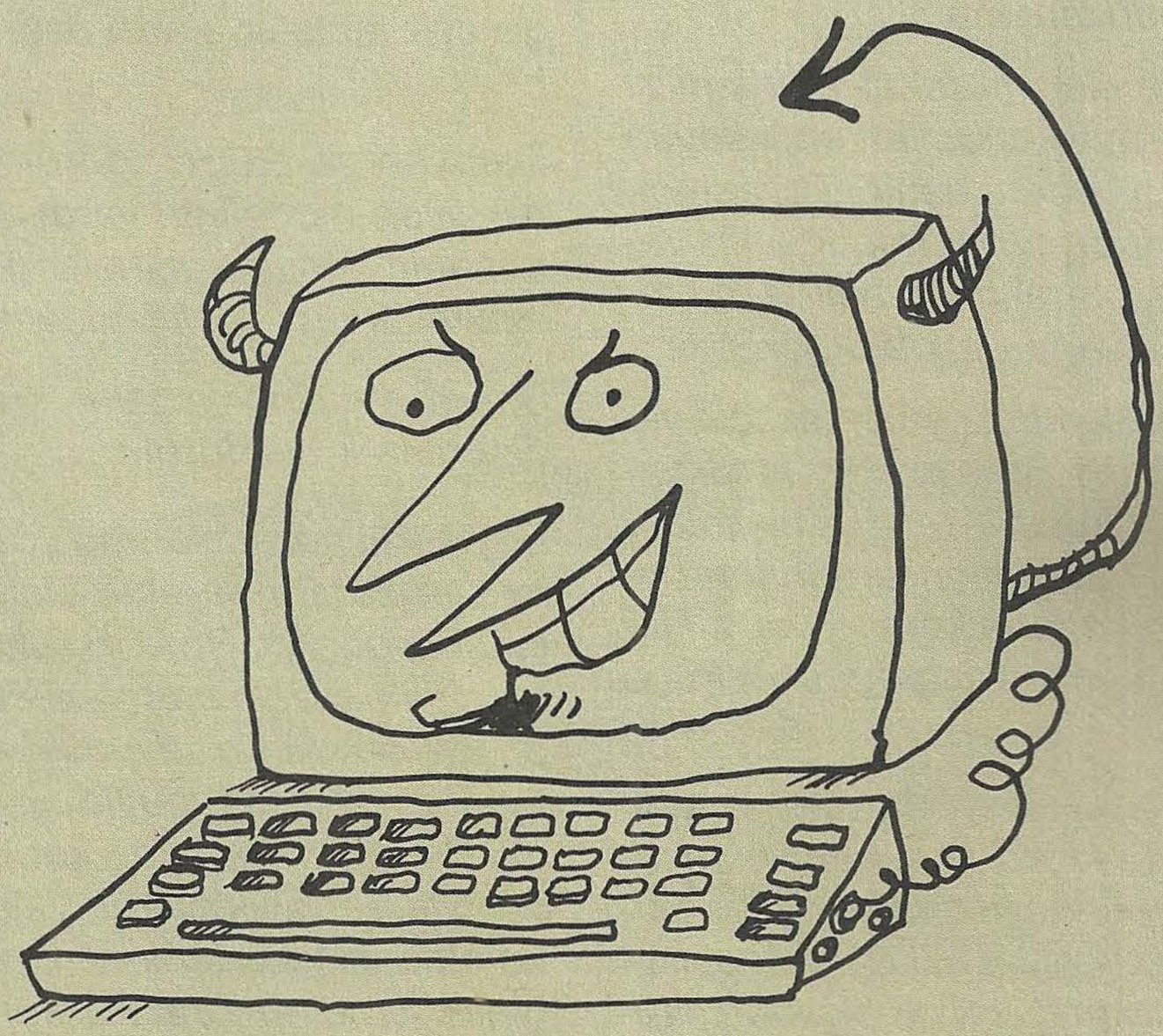
*Sistemin **hiyerarşik** yapısı sayesinde, kinematik işlevler mekanik parçalar için de aynı biçimde hesaplanabilir. Örneğin pedalların ve vites kolunun sabit yerel eksenler çerçevesinde döndüğü kabul edilerek, basit bir yinelemeli yöntemle en uç noktalar ve sürücünün bunlara erişip erişemeyeceği hesaplanabilir. Böylece, örneğin dördüncü vitese takabilmek için sürücülerin % 95'inin kolunu aşırı bükmesi gerektiği veya geri vitese takabilmek için % 5'inin fazla güç harcaması gerektiği bulunup tasarım bunları önleyecek biçimde düzeltilebilir. Çeşitli olasılıkların hesaba katıldığı bu hızlı model düzeltme çalışmaları ve manken tasarımcı etkileşimleri, elle çalışmaya dayanan geleneksel yöntemlere oranla tasarım süresini önemli ölçüde azaltmaktadır.*

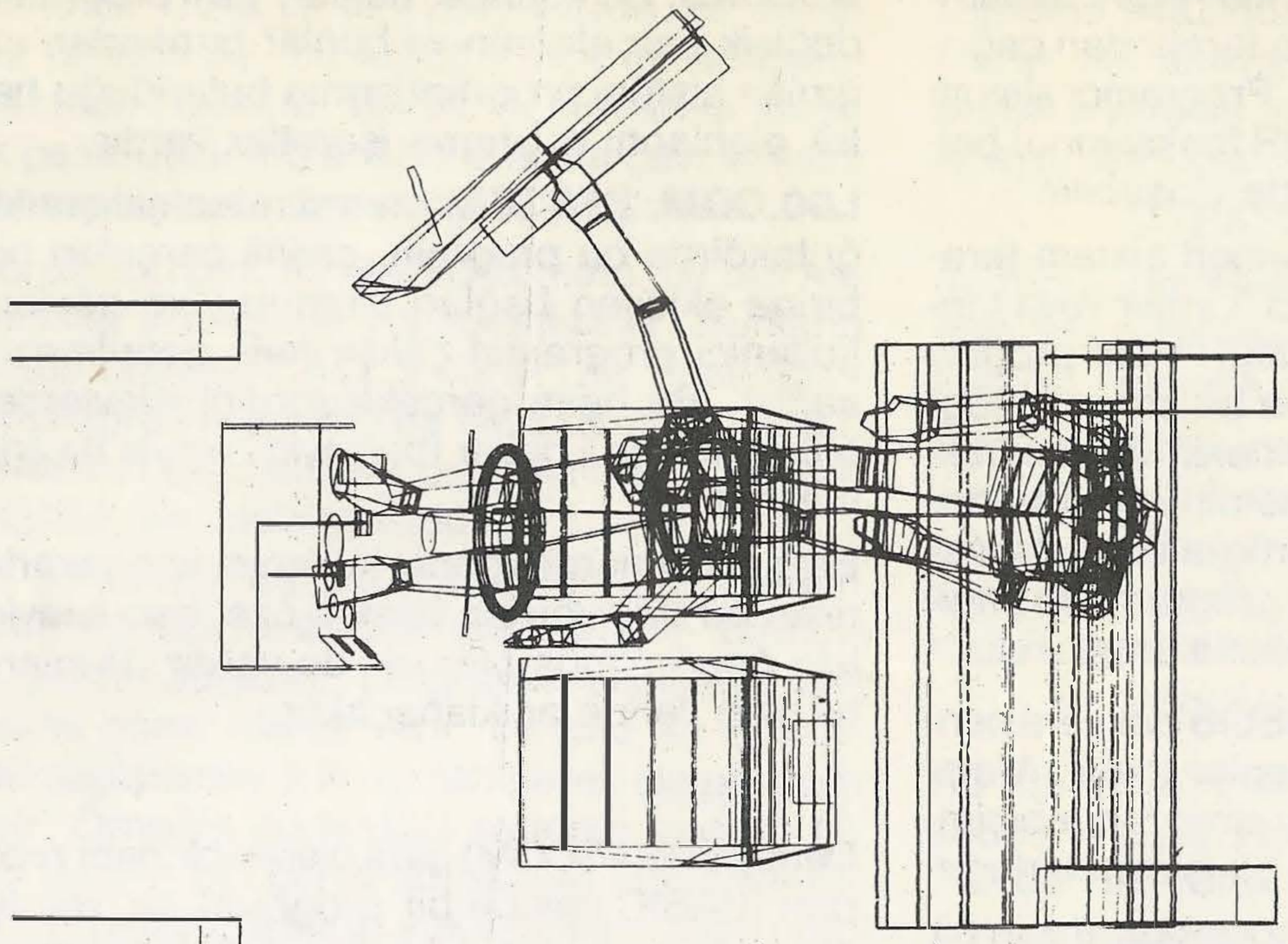
*İç kısmın modelleme çalışması bittikten sonra, tekrar aracın karoserine dönülerek gerek estetik, gerek aerodinamik yönlerden daha dikkatli bir düzeltme çalışmasına geçilir. **Eğimli yüzeyler,** ilk başta küçük üçgenlerin yan yana getirilmesiyle oluşturulmuştu. Artık, çok kesin matematik modellerden hareketle, daha karmaşık yüzeyle de gözönüne alınarak, modelin dış yüzeyine incelmiş olan son biçimi verilir. Bu eğimli yüzeyler, bir kez belirlendikten sonra, maketin yapısına uyarlanır. Böyle kesin bir matematiksel modele sahip olunca, tasarımcı prototipin imal edilmesinin gerektiği klasik çalışmalarda olduğu gibi, model üzerinde bütün aerodinamik, malzeme mukavemeti vs. hesaplarını yürütür.*

*Karoserin dış yüzeyinin böylece **perdah**lanmasından sonra, yine yol ve göstergelerin görülebilirlik çalışmalarına dönülür. Örneğin, zemini temsil eden bir çerçeve içine direkler, trafik işaretleri gibi cisimler yerleştirilerek, ön koltukta oturan bir kişinin, sürücünün görüş alanını ya da dikiz aynasından görülen alanı ne derece etkilediği ölçülebilir. Bundan sonra, görüş alanı tam olarak çizilip, daha prototip ve ahşap maketin yapılmasından çok önce oldukça yararlı bilgiler edinilebilir. SAMMIE'nin, bir de MIRROR (ayna) adı verilen ve görüntüleri perspektife uygun biçimde yansıtılabilecek optik özellikleri içeren bir birimi vardır. Bu birim, modelin iç aksamının belli unsurlarını aynaya dönüştürebilmektedir.*

Kullanılacak bir başka komut ADJUST FOR OBJECT (cisme uyarla) komutudur. Bu komut aynayı arka plandaki bir cismi, iki vektörünün (sürücünün gözünden aynaya ve aynadan bakılan cisme) açı ortayı aynaya dik olacak biçimde ayarlar. Bu işlemin sonucunda, sözkonusu cismin görüntüsü, tüm gölgeleri ile, perspektif içinde aynada belirir.

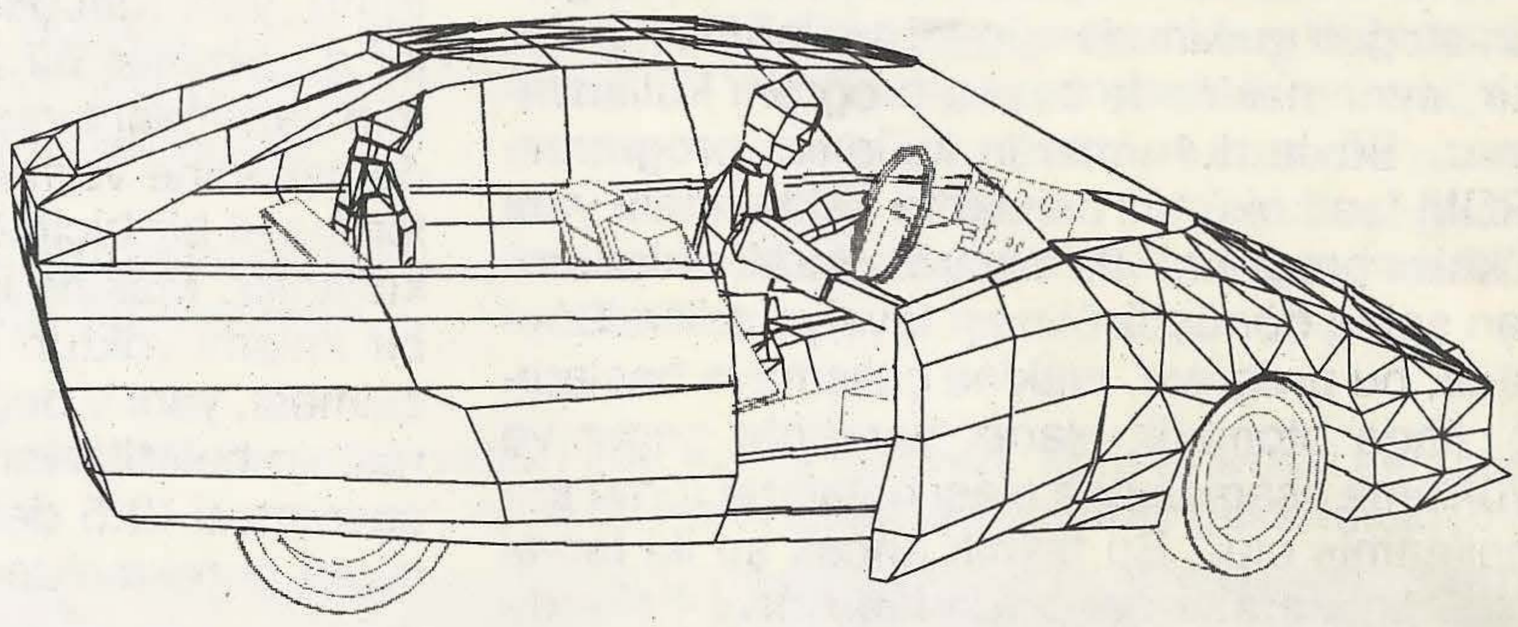
***Kör noktaların elenmesi,** gerektirdiği makine çalışma süresi açısından, BETA'nın yan sistemleri içinde en fazla yer tutanıdır. SAMMIE'nin, kör noktaları elemek için sağladığı*





Çeşitli komutlar kullanıcının hareket ve aletleri kullanım imkânlarının denenmesini mümkün kılar. Bu sayfadaki şekillerde vites kolunun uzaklığının uygun olup olmadığı kontrol edilmektedir.

Bilgisayar destekli tasarımın bir sakıncalı yanı bölümlenmiş zeminin ''okunması''ndaki güçlüktür. Ancak bu bölümde dış yüzeyler tarafından örtülen iç eklemleri belirtir. Bir komut bu iç eklemleri görünür kılabilir.



olanaklar şu işlemlerle sınırlıdır: Yalnız çizgilerden oluşan bir model (kafes yapısı) oluşturup, bu modeli kendi çerçevesinde döndürerek çeşitli perspektifleri beleğe aktarmak.

Böyle bir çalışmanın sonunda, tasarımcı HIDDEN LINES adlı bir işlevin yardımıyla gerçekleştirdiği perspektifleri ''temize çekip'', dokümanların arasına katmaktadır. Sonuçta bütün bu perspektiflerin çakışmasından, hiçbir bakış açısından girilemeyen noktalar, yani kör noktalar tespit edilmiş olur.

Görüldüğü gibi, bilgisayar destekli tasarımlarda da aynen geleneksel yöntemlerle izlenen yol izlenmekte, ama tasarım süresi önemli ölçüde kısalmakta ve sistem, daha ilk denemede başarılı bir maket oluşturabilmek için gerekli tüm verileri sağlamaktadır. Yukarıda sözünü ettiğimiz bütün çalışmalar, kalem, kâğıt, çizim masasında yürütülenlere göre daha hızlı ve kesin olmaktadır. Böylece, tasarımcı, sıkıcı ve tekrara dayanan çalışmalardan kurtularak daha yaratıcı olabilmektedir.

Bugün, BETA sistemleri proje maliyetlerini 5'te birden 100'de bire kadar ve hatta bazen daha büyük oranlarda düşürebilmektedir. Genelde bu maliyetlere gerçek maket çalışmaları eklenmez. Ancak bunlar eklendiğinde bile ortaya çıkan bilanço son derece olumludur.

Günümüzde kullanılan çok yetenekli BETA sistemleri, yalnızca IBM, VAX ... gibi büyük makinelere yüklenebilmekte ve dolayısıyla ancak büyük işletmeler tarafından kullanılabilmektedir. Buna karşın daha basit grafik yazılımlar (bunlar yalnızca iki boyutlu olmaktadır) kişisel bilgisayarlara da yüklenebilmektedir. Ne yazık ki, bugün için bu iki uç arasında başka olanak yoktur. Gelecekte ise, kişisel bilgisayarlarda çalıştırılan BETA sistemlerini kullanan mühendislik, mimarlık ve grafik bürolarının artacağı tahmin edilmektedir.

BASLIB.REL. BRUN.COM kitaplığını tamamlayan ve yürütülen program tarafından çağrılan diğer programları içerir. Programcı sistem programlarına (örneğin, SQR fonksiyonu) bellekteki yerini belirtmeden de ulaşabilir.

Tüm çeviri ve bağlama işlemleri sistem tarafından Derleyici ve Bağlayıcı (Linker veya Linking Loader) yoluyla yürütülür. Ham programın her satırı çözümlenir ve bir sistem programının çağırıldığı yerler belirlenir. Bunun üzerine çağırılan program, sistemin kitaplıklarında aranır ve yürürlükteki programın çalışmasına ara vermeden, sistem programının çağırılmasını sağlayacak adresleme oluşturulur.

OBSLIB.REL. BRUN ve BASLIB'dekilere benzer alt-program ve fonksiyonlar içerir. Aralarındaki fark, kullanıcı programın farklı biçimde kullanılmasından doğan bir işletim farkıdır.

Bazı uygulamalarda, kullanıcı programının bir kez tanımlandıktan sonra bir daha hiç değiştirilmeden makinede sürekli saklanması gerekir, aynı makinede başka program kullanılamaz. Böyle durumlarda kullanıcı programın ROM (salt okunur bellek)'te saklanması yani ''kalıcı program'' (firmware) olarak adlandırılan şekle dönüştürülmesi tavsiye edilir. Böylece, bu program, makine çalışmaya başladığı anda otomatik olarak harekete geçer ve yükleme aşamasında olası hataların tümü engellenmiş olur. Bu teknik ancak şu iki temel ölçütün beraberce bulunduğu özel hallerde kullanılır.

- ☐ makine ve program birlikte bir sistem oluşturmuş ve değişmeyen bir uygulamaya yönelmişlerse (ya da uygulama değiştiğinde bellek de değiştiriliyorsa);
- ☐ bir seri üretim sözkonusu ise; çünkü tek bir uygulama için ROM belleği hazırlamak pek rantabl değildir.

ROM belleğine bir programın yazılması olgusu değerlerin, programda ayrılan bellek alanına artık yazılamaması durumunda, farklı bir yapılanmayı beraberinde getirir. Değişkenler, programı içeren ROM alanından ayrı olan, RAM belleğinde saklanmalıdır. Bu da birtakım karmaşık yöntemlerin kullanılmasını gerektirecektir.

Pratikte, bu yapılanma birbirinden ayrılmış iki bellek alanının kullanılmasını zorunlu kılar; program alanı (ROM belleği) ve veri alanı (RAM belleği). Bu tür olanaklardan, bağlayıcıya özgü mesajlarla birlikte BRUN ve BASLIB kitaplıkları yerine OBSLIB'in yüklenmesiyle yararlanılabilir.

BCLOAD. Bu kütükte, bağlar, yani programın değişik parçalarının ve bunlar tarafından çağırılan sistem programlarının bulunduğu bellek alanlarını gösteren işaretler vardır.

L80.COM. BASCOM'un ardından çalıştırıldığı takdirde bu program, çeşitli parçaları birbirine ekleyen bağları oluşturur ve böylece kullanıcı programın çalışır hale getirilmesini sağlar. Bu birim gerçekleştirdiği işlevlerden ötürü Linking Loader (bağlayıcı) adıyla da anılmaktadır.

Bazı sistemlerde, genel kullanım için yararlanılan bu birimlerden başka, özel bazı işlevler için öngörülmüş birimler de vardır. Bunların bazıları ileride açıklanacaktır.

Derleme

Derleyici (BASCOM) girdi olarak bir ham program (BASIC dilinde bir program) alır ve çıktı olarak bunun yeni adreslenebilir bir versiyonunu, yani tüm göreceli ilişkileriyle Assembler'e çevrilmiş bir program verir. Bir programın yazılması sırasında, 2.5 gibi bir değeri V değişkenine vermek için, yürütülecek işlemleri basit bir biçimde belirten V=2.5 komutu kullanılır. Makine için, böyle bir ifadenin hiçbir anlamı yoktur. Bu komutun kodunun çözülmesi, yani V değişkenine ayrılan bellek adresinin belirtilmesi ve komutun makine diline çevrilmesi (2.5 değerinin V adresine aktarılması) gerekmektedir.

Derleme sırasında, programın tüm komutları makine diline çevrilir ve her değişkene bir bellek alanı ayrılır. Kullanılan bütün adresler, sistem tarafından belirlenen değerler dizisine **göreli**dir: Bir tek değer değiştirilerek, tüm değişkenlerin bellekteki yerleri değiştirilebilir. İşte bu yüzden derlenmiş programa **değişebilir** program da denir.

Göreli değerlerden hareketle, bellekteki mutlak (fiziksel) adresleri elde eden işlem, **bağlayıcı** tarafından yürütülür. Değişebilir ve derlenmiş program. REL türünden bir kütükte saklanabilir. Derleme güdümünün söz dizimi şöyledir:

BASCOM Derlenmiş, döküm = Ham

Komutta kullanılan parametreler şu anlamları taşımaktadır:

Derlenmiş. Programın değişebilir versiyonunun saklanacağı kütüğün (REL türünden) adıdır. Bu tür bir kütüğe ''**object**'' nesne (derlenmiş) adı verilir.

Döküm. Ham programın dökümünü ve çevirisini içeren ''döküm'' kütüğünden söz edil-

mektedir. 726. ve 727. sayfalarda, bir derleme örneğinde, bu kütüğün içeriği sergilenmektedir. Döküm, BASIC komutlarına karşılık gelen Assembler komutlarına çevrilmesini kapsamakta ve derlemeyi izlemektedir.

Görüldüğü gibi, bir BASIC komutu, bir dizi Assembler komutuna yol açmaktadır. Bunların içinde örneğin CALL $ CINA (Bir Assembler alt-programı çağırılırken GOSUB değil, CALL kullanılır) gibi, sistem programlarına yönelik çağrılar yer alabilmektedir.

Bu dökümde, ham kütüğün her komutundan önce 16 tabanlı iki sayı gelmektedir. Birincisi, satırın adresini, programın başlangıç adresine göreli olarak verir. İkincisi ise komuttaki değişkenler için ayrılan bellek alanını gösterir. Örneğin 40 BASIC satırının çevirisi 1F (10 tabanlı 31) göreli adresinde bulunmaktadır. Program belleğe yüklenirken, başlangıç adresi artık sıfır değil, sistem tarafından belirlenen bir başka değerdir. Böylece göreli adresi 1F olan 40 komutu, 1F artı programın başlangıç adresi gibi bir mutlak adres ''taban''ına oturur.

Ham Program. Derlenecek programın bulunduğu ASCII kütüğünden söz edilmektedir.

Derleme sırasında, Derleyici çağırılırken özel olarak adı anılmasa da, derlenmiş bir kütük mutlaka yaratılır. Adı belirtilmezse, ham kütükle aynı adı taşır. Örneğin:

BASCOM = DENE1

güdümü, derlemeyi başlatır ve bağlama aşamasında çağırılması gereken DENE1.REL adlı kütüğü otomatik olarak yaratır ve çağırır.

Ama, dökümü içeren kütük otomatik olarak yaratılmaz. Adıyla istenmek zorundadır. Ayrıca, bu kütük doğrudan ekrana ya da bilgiyazara aktarılabilir. Bu durumda, çağırma işlemi şöyle yapılacaktır:

BASCOM DENE2, LST: = DENE

Bu durumda, makine DENE2.REL adlı değişebilir kütüğü yaratıp, bu derlemenin dökümünden bir kütük oluşturmak yerine bunu bilgiyazara yollayacaktır. LST: kodu, burada bir bilgiyazım isteği gibi yorumlanır (CP/M sisteminde, LST birimi bilgiyazara karşılık gelmektedir). TTY kodu ise ekranda dökümü sağlar. Hareket kaynağı olan (Ham) ve sonuçta yaratılan (Değişebilir döküm) kütükler, çağırma anında belirlenen farklı disket birimlerinde bulunabilirler.

Toronto'daki Hava Kuvvetleri'nde eğitim merkezi. Konsollardaki ekranlar olası stratejik savaş durumlarını göstermektedir.



DERLEME ÖRNEĞİ

```
BASCOM 5.30 - Copyright 1979,80,81 (C) by MICROSOFT - 26322 Bytes Free
 0014 0007          10 ' *** DERLEME TESTI
 0014 0007          20 OPTION BASE 1
 0014 0007          30 DIM A(5)
 0014 0007          40 FOR I=1 TO 5
          ** 0014'I00000: CALL    $530
          ** 0017'L00010: L00020: L00030: L00040:
          ** 0017'        CALL    $LFMA
          ** 001A'        DW      <const>
          ** 001C'        JMP     I00001
          ** 001F'I00002:
 001F 001B          50 B=I*A(I)
          ** 001F'L00050: LXI     H,I!
          ** 0022'        CALL    $CINA
          ** 0025'        DAD     H
          ** 0026'        DAD     H
          ** 0027'        LXI     D,A!+FFFC
          ** 002A'        DAD     D
          ** 002B'        CALL    $LFHA
          ** 002E'        CALL    $FMUC
          ** 0031'        DW      I!
          ** 0033'        CALL    $FASO
          ** 0036'        DW      B!
 0038 0023          60 NEXT I
          ** 0038'L00060: CALL    $FADA
          ** 003B'        DW      I!
          ** 003D'        DW      <const>
          ** 003F'I00001:
          ** 003F'        CALL    $FASO
          ** 0042'        DW      I!
          ** 0044'        CALL    $LEJA
          ** 0047'        DW      I!
          ** 0049'        DW      <const>
          ** 004B'        DW      I00002
 004D 0023          70 FOR I=1 TO 5
          ** 004D'L00070: CALL    $LFMA
          ** 0050'        DW      <const>
          ** 0052'        JMP     I00003
          ** 0055'I00004:
 0055 0023          80 PRINT "I =";I,"A(I) =";A(I)
          ** 0055'L00080: CALL    $PR0A
          ** 0058'        LXI     H,<const>
          ** 005B'        CALL    $PV1D
          ** 005E'        LXI     H,I!
          ** 0061'        CALL    $PV0A
          ** 0064'        LXI     H,<const>
          ** 0067'        CALL    $PV1D
          ** 006A'        LXI     H,I!
          ** 006D'        CALL    $CINA
          ** 0070'        DAD     H
          ** 0071'        DAD     H
          ** 0072'        LXI     D,A!+FFFC
          ** 0075'        DAD     D
          ** 0076'        CALL    $LFHA
          ** 0079'        LXI     H,$AC%
          ** 007C'        CALL    $PV2A
 007F 0023          90 NEXT I
          ** 007F'L00090: CALL    $FADA
          ** 0082'        DW      I!
          ** 0084'        DW      <const>
          ** 0086'I00003:
          ** 0086'        CALL    $FASO
          ** 0089'        DW      I!
          ** 008B'        CALL    $LEJA
```

```
        ** 008E'          DW        I!
        ** 0090'          DW        <const>
        ** 0092'          DW        I00004
 0094 0023        100 END
        ** 0094'L00100:  CALL      $END
 0097 0023
        ** 0097'          CALL      $END
 00B3 002D

00000 Fatal Error(s)
25909 Bytes Free
```

Örneğin,

BASCOM B:DENE2, LST:=DENE

yukarıdaki güdümle aynı işi görür. Bunu yaparken de çıktıyı (DENE2.REL) B disketine yollar.

Derleyici, komutları çevirirken, ham kütüğün içerdiği sözdizimi hatalarını belirler. Bazı durumlarda, derlenmiş kütüğü yaratmadan, yalnızca derleme yapmak yararlı olabilir. Böyle bir durumda kullanılacak güdüm şudur:

BASCOM, TTY=TTY:

güdümü, ham kütük de dahil, hiçbir kütüğü kullanmaz. Komutları teker teker ekrana alır. Her satır, girildiği anda çevrilir ve olası hatalar ekranda da belirir. Bu olanak, genelde, komutların biçimsel (sözdizimsel) doğruluklarını görmek için kullanılır. Özellikle de, hataları her komuttan sonra göstermesi nedeniyle, programlamayı yeni öğrenenler için biçilmiş kaftandır.

Derleyicinin Özel İşlevleri

Derleyiciyi çağıran güdüm, her biri özel işlevleri harekete geçiren bazı anahtarlarla (switches) desteklenebilir. Aynı derleyici, değişik BASIC türlerinde farklı çalışabilir. Bu anahtarlar, BASIC'in türünü ve bazı özel bilgilerin işlenme biçimini belirtir. Anahtarlar üçe ayrılır:

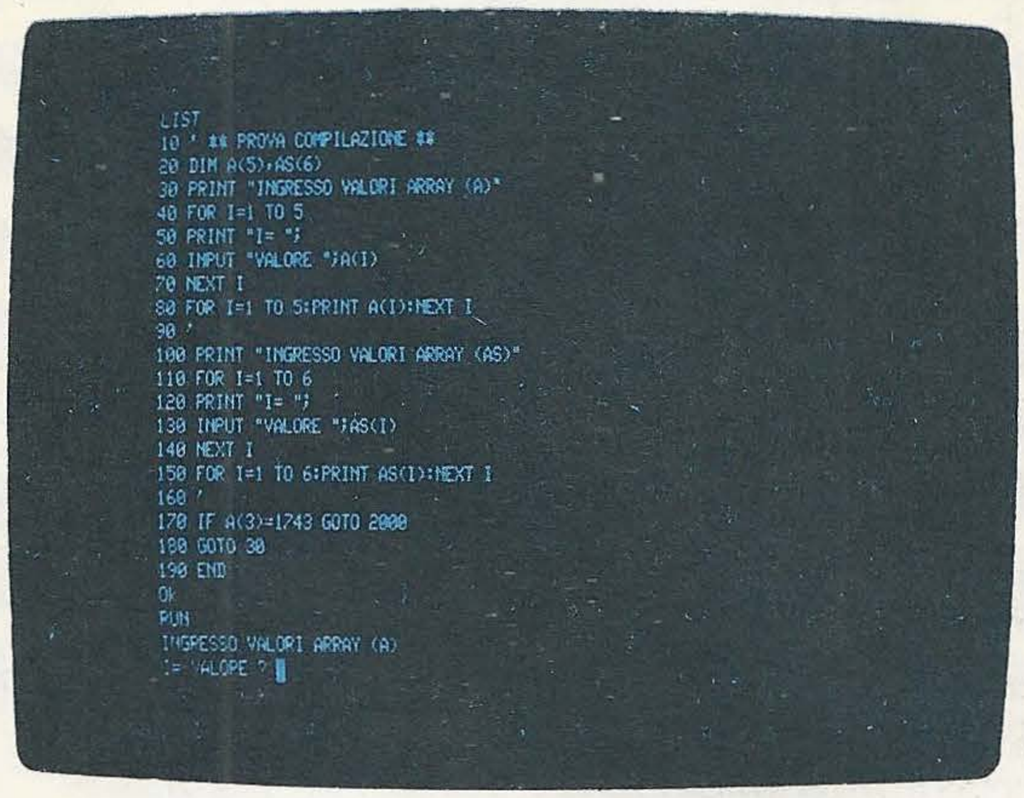
- ☐ uzlaşımların belirlenmesi
- ☐ hataları saptayan fonksiyonlar
- ☐ özel kodlar

Bu anahtarların nitelikleri daha sonra açıklanacaktır. Bunların doğru olarak kullanılması ve karşılık gelen kodlar, eldeki Derleyiciye göre değişebilir.

Uzlaşımların Belirlenmesi. Eldeki BASIC versiyonuna göre, bazı komutların sözdizimi farklı olabilir. Sistem komutları hakkındaki uzlaşım-

DERLEME ÖRNEĞİ (1)

Bu sayfada üç hata içeren basit bir programın dökümü verilmiştir. Programı derlemeden önce, BASIC çeviricisi (* MBASIC) ile çalıştıralım.




```
LIST
10 ' ** PROVA COMPILAZIONE **
20 DIM A(5),AS(6)
30 PRINT "INGRESSO VALORI ARRAY (A)"
40 FOR I=1 TO 5
50 PRINT "I= ";
60 INPUT "VALORE ";A(I)
70 NEXT I
80 FOR I=1 TO 5:PRINT A(I):NEXT I
90 '
100 PRINT "INGRESSO VALORI ARRAY (AS)"
110 FOR I=1 TO 6
120 PRINT "I= ";
130 INPUT "VALORE ";AS(I)
140 NEXT I
150 FOR I=1 TO 6:PRINT AS(I):NEXT I
160 '
170 IF A(3)=1743 GOTO 2000
180 GOTO 30
190 END
Ok
RUN
INGRESSO VALORI ARRAY (A)
I= VALORE ?
```

■ 20, 130 ve 150 satırlarında, AS adlı bir dizeyden bahsedilmektedir. Oysaki bu sözcük sisteme ayrılmış olup, daha önce gördüğümüz gibi bir G/Ç ara belleğinin, bir disket kütüğünün yönetimi için ayrılmasında kullanılıyordu; (örneğin FIELD 1 20, AS BX$...), dolayısıyla bunu bir değişken adı olarak kullanamayız. Bazı çevrilmiş BASIC versiyonlarında, sistem kendisine ait bir sözcüğün hatalı bir biçimde kullanılıp kullanılmadığını belirleyebilmektedir.

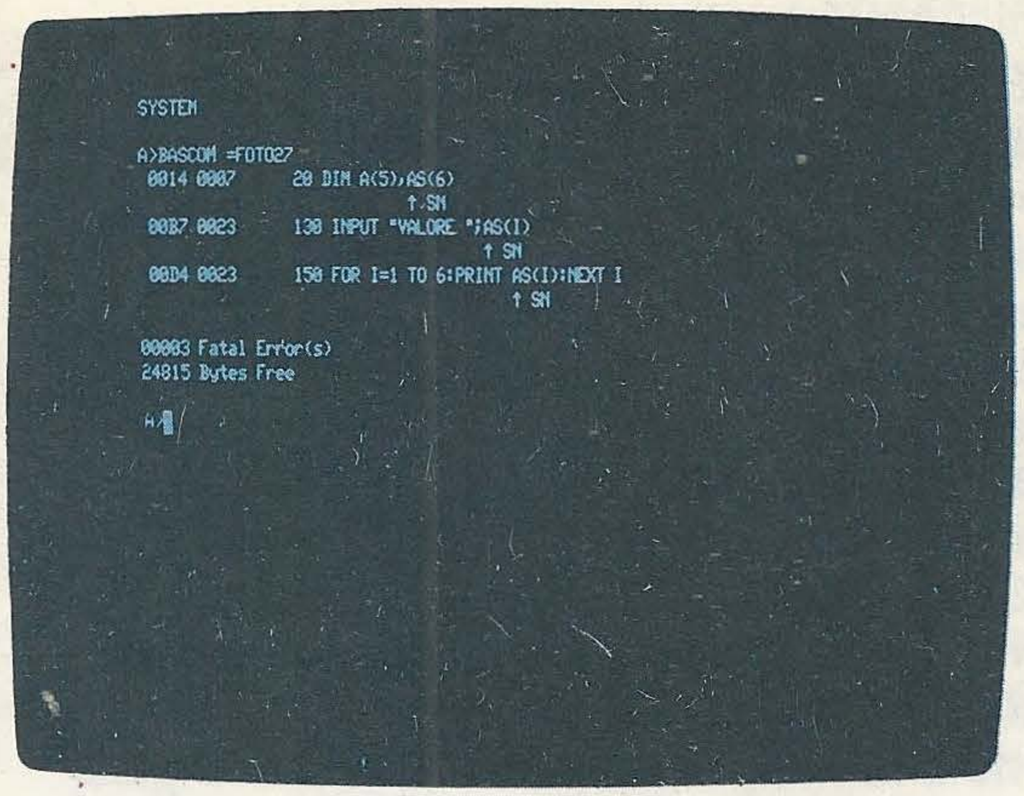
■ İkinci hata, OPTION BASE 1 komutunun yokluğundan kaynaklanmaktadır. Bu durumda A ve AS dizeyinin elemanlarının numaralandırılması sıfırdan başlar. Dizeyler 5 ve 6 olarak boyutlandırıldıklarına göre, A(5) ve AS(6) elemanlarına birer değer atama olanağı kalmamıştır.
BASIC bazı durumlarda, bu türden hataları otomatik olarak düzeltmektedir.

Sonuncu hata da, olmayan bir 2000 satırına gönderme yapan 170 komutunda bulunmaktadır. Çevirici, 170 satırını yürütene kadar, bu hatanın farkına varılmayacaktır. Programın yapısı gereği, kullanıcı A(3) = 1743'ü girmedikçe, hata ortaya çıkmayacaktır.

■ Ekrandaki son üç satırda da görülebileceği gibi, belirtilen hatalara rağmen program, çevirici konumunda o ana kadar normal biçimde çalışmıştır.

DERLEME ÖRNEĞİ (2)

Bu sayfada, bir önceki sayfada yer alan programın derlenme aşamasını görmekteyiz.




```
SYSTEM

A>BASCOM =FOTO27
 0014 0007      20 DIM A(5),AS(6)
                   ↑ SN
 00B7 0023      130 INPUT "VALORE ";AS(I)
                              ↑ SN
 00D4 0023      150 FOR I=1 TO 6:PRINT AS(I):NEXT I
                                    ↑ SN

00003 Fatal Error(s)
24815 Bytes Free

A>
```

Birinci satır, kullanıcının işletim sistemine geçmek istediğini belirten güdümü gösteriyor, ikincisi ise derleyiciyi çağırıyor.

Derleyici, 20. satırda sisteme ait AS teriminin hatalı kullanıldığını saptıyor ve bu satırı ekrana getiriyor.

Bundan sonraki satırda, hatanın belirlendiği yeri gösteren bir okla birlikte, derleyici tarafından, hatanın türünü belirten bir simge yazılıyor: SN, Syntax Error (sözdizimi hatası)'nı simgelemektedir. Aynı türden hatalar 130 ve 150 satırlarında da belirlenmiştir. 170 satırındaki hata ise, öncekiler tarafından maskelendiği için ortaya çıkarılamamıştır. Belirtilen hataların düzeltilmesinden sonra, yapılan ilk derleme işleminde ortaya çıkacaktır.

Derlemenin sonunda, belirlenen hata sayısı da ekrana yansımaktadır.

lara bağlı olarak oldukça önemli farklılıklara rastlanabilir.

BASIC komutları ve fonksiyonlarının adları sisteme ayrılmıştır ve bunlar değişken adı olarak kullanılamazlar. Bazı BASIC türlerinde, bütün ifadelerin boşluklarla ayrılması gerekirken, bazılarında böyle bir zorunluluk yoktur (bu boşlukları Derleyici koyar). Kullanılmakta olan yazı düzeni Derleyiciye bildirilmezse, bazı kaçınılmaz hatalar doğar. Örneğin, FOR I = A TO C komutunun boşlukları çıkarıldığında, FORI = ATOC şeklini alır. Eğer bu durumda derleyici birinci türden ise, komut bir döngü başlangıcı olmaktan çıkıp, FORI değişkenine ATOC'da bulunan değerin verilmesi anlamına gelecektir. Bu tip hatalardan söz ediyorken, konuyla ilgili ilginç bir özellik belirtilmelidir.

Fortran dilinde (BASIC'e çok benzer) döngüler şu tür bir komutla tanımlanır:

```
DO 100 I = 3,10
```

Burada DO kodu FOR'un I = 3, 10 da I = 3 TO 10'un yerine geçer. 100 sayısı ise döngünün tamamlandığı satırı gösterir.

Bu iki döngü biçimini karşılaştırırsak (Fortran ve BASIC döngüleri) şunu elde ederiz:

```
FORTRAN              BASIC
10 DO 100 I = 3,10   10 FOR I = 3 TO 10
— — — —              — — — —
100 CONTINUE         100 NEXT I
```

Fortran derleyicisi, komuttaki tüm boşlukları ayıklar. Böylece 10 satırı DO100I = 3,10 haline gelir ve yorumlamanın doğruluğu sayaçlar arasındaki virgüle bağlanır (3, 10). Uzaya gönderilecek bir uydunun fırlatılmasıyla ilgili bir araştırma programda, ham kütükte yanlışlıkla virgül yerine nokta konmuş ve komut DO-100I = 3.10 şekline dönüşmüştür. Program çalışırken de, bir döngünün oluşması beklenirken, DO100I değişkenine 3.10 değeri atanmıştır. Sonuçta * fırlatma işlemi tam bir felaketle sonuçlanmıştır.

BASIC'te döngünün şekli ve özellikle döngüyü kapayan (NEXT komutunun kullanılması nedeniyle) benzer bir hata olasılığı çok daha azdır. Eğer başlangıçtaki FOR ... hatalı biçimde yorumlanırsa, derleme NEXT satırında, karşılık gelen bir FOR bulamadığı için hata mesajı verecektir. Buna karşın FORTRAN'da, döngüyü kapatan bir komut yoktur (CONTINUE kodu tamamen gereksizdir). Eğer başlangıç satırı doğru biçimde yorumlanmazsa, döngü yok olur ve sonuçta bir hata mesajı ortaya çıkmaz.

BASIC derleyicilerindeki diğer farklılıklar, döngüler, yuvarlamalar ve satır numaralandırmaları ile ilgilidir.

Parametreli bir döngünün çalışması sırasında, iki sınır değerin aynı olması gibi özel bir durumla karşılaşılabilir. Örneğin aşağıdaki döngü, 3 ile başlamakta, yine 3 ile bitmektedir.

```
10 A = 3: B = 3
20 FOR I = A TO B
— — — — —
```

Sırası geldiğinde bu döngünün çalıştırılıp çalıştırılamayacağı konusundaki kararsızlık, döngüyü en az bir kez çalışmaya zorlayan bir seçenek sayesinde ortadan kalkar.

Genelde aynı seçenek çift hassasiyetteki sayıların tamsayıya çevrilmesinde yuvarlama yerine budamayı da sağlar.

Bir başka yararlı seçenek de, satır numaralarının dikkate alınmamasını sağlar. Bu özel yöntem sayesinde satırlar herhangi bir sırayla numaralandırılabilir ya da hiç numaralandırılmayabilir. Ancak tabiî ki, GOTO ve GOSUB ile gönderme yapılan komutlar bunun dışındadır.

Bu biçimde, bir BASIC programı, Fortran'daki dengine oldukça benzer bir hale gelerek, daha derli toplu ve okunabilir bir hal alır.

Hataları saptayan fonksiyonlar. ON ERROR GOTO komutunun kullanılması daha karmaşık bir bağlama sistemini ve dolayısıyla farklı bir derleme biçimini gerektirir.

Hataları işleme seçeneği, özel bazı belirteçlerle desteklenmelidir. Derleyici bu durumda, ham BASIC programının satır numaralarını içeren bir adresler tablosu düzenler. Böylece, eğer programın çalışması sırasında bir hata ortaya çıkarsa, sistem hatalı satırın numarasını bulabilir.

Normal olarak, derlenen programlarda, satır numarası kaybolur ve her komut bellekteki adresiyle tanımlanır. Bir hatanın saptanması sırasında, eğer sistem, belleğinde adresler- satır numaraları değişim tablosunu saklamamışsa, hatanın ortaya çıktığı fiziksel adres, ekranda 16 tabanlı olarak görüntülenir. Bu tür bir adres, hatanın ortaya çıktığı ham satırı bulmamıza yardımcı olmaz, bunu satır numarasına dönüştürmek ise, çok uzun işlemler gerektirir.

Programın hatalardan arındırılması sırasında

Software Reliability (Yazılımın Güvenilebilirliği), Wyler & Sons, New York, 1976, adlı kitaptan.

(debugging), derlenmiş program her ne kadar daha fazla bellek alanı gerektirecekse de, tavsiye edilir. Hatalardan arındırma bitince, hata belirteçleri olmaksızın program bir kez daha derlenebilir.

Her satır numarasına 16 tabanlı bir adres veren tablo, her eleman için bir giriş noktası içerir (**entry point** bir program parçasına veya bir dizeye ulaşmayı sağlayan giriş yeri). Genelde dizeyler yalnızca birinci elemenlarının adreslerine karşılık gelen tek bir giriş noktasına sahiptirler. Diğer elemanları ise, giriş noktasına göreli olarak konumlandırılırlar. Aynı şey, satır numaralarını içeren tablolar için olanaklı değildir. Programın her satırı için bir giriş noktası gerekir. Bellek adreslerinin böylesine genişletilmiş bir yönetimi, programın akışını önemli ölçüde yavaşlaştırmakta ve derlenmiş (2 tabanlı) kütüğün uzunluğunu artırmaktadır. Bu nedenle, yalnızca sinema aşaması için kullanılması yerinde olur.

Özel kodlar. Belirteçlerin ana işlevleri, Assembler türünün seçimi, TRON ve TROFF komutlarının kullanımı, özel derleme uzlaşımlarının ve programın ROM'a aktarılmasını kapsar.

Çeviricili BASIC'de, sorunsuz çalışan TRON ve TROFF komutları, özel olarak belirtilmedikçe, derlenmiş bir programda dikkate alınmazlar.

Özel bir belirteç, dökümün, Assembler'in hangi türünde istendiğini belirler. Bazı ana özellikleri hariç, Assembler'in özelliklerinin çoğu minikişlemciye bağlıdır ve sözdizimi de yine minikişlemci tarafından belirlenir. Bu durumda, aynı komut, farklı minikişlemcilerde farklı biçimde çevrilecektir.

Bağlama

Çalışabilir-derlenmiş programın elde edilmesindeki son aşama, çeşitli parçaların, örneğin derleyiciden çıkan yeni adreslenebilir kütükte sistemin standart kitaplıkları gibi birbiriyle bağlanmasıdır. Bu işlev, Linking Loader adı verilen bir program tarafından gerçekleştirilir. Bu program, girdi olarak, değişebilir kütüğü (derleyicinin çıktısını) alır ve çıktı olarak da çalışan bir program verir.

Linking Loader, bellek yerleşimi hakkında bir dizi bilgiyi ekrana getirir. Bu bilgilerin en önemlileri şunlardır (parantez içindeki numaralar sayfa 732'deki şemayla ilgilidir):

— programın başı (program-start); 16 tabanlı başlangıç adresi **(1)**
— programın sonu (program-end); 16 tabanlı bitiş adresi **(2)**
— Uzunluk: program tarafından kaplanan bayt sayısı **(3)**
— geri kalan yer (kullanılmamış durumdaki bellek alanı)
— 16 tabanlı olarak, çalışmanın başladığı adres, program başı adresiyle aynı olmak zorunda değildir. **(5)**
— sayfa sayısı, program tarafından kullanılan 256 baytlık ''bellek sayfası'' sayısını belirten 16 tabanlı sayı. **(4)**

732. sayfadaki şema, belleğin paylaşımını göstermektedir. Burada dikkati çeken nokta, programın başında COMMON için bir alanın ve DATA komutuyla belirlenen değerlere ayrılan bir alanın yer alması, çalışmanın başladığı adresin ise, yürütülebilecek ilk komuta karşılık gelmesidir. COMMON alanı, değişik programlar tarafından ulaşılabilen ve birinden diğerine verilerin ve parametrelerin aktarılmasına yarayan bir bellek alanıdır.

Değişik programlar, karşılıklı olarak birbirlerini çağırabilirler. Fakat birinin belleğe yüklenmesi, öncekinin bellekte kapladığı alanın üzerini kaplar. Bu sırada bütün değişkenler silinir ve değerlerin aktarılması gerçekleşmez. Bir programdan diğerine verilerin aktarılmasını gerçekleştirmek için, bu değişkenleri ''ortak'' (common) olarak belirtmek gerekir. Böylece bunlar, ayrı bir bellek alanında saklanacaklar ve programların yüklenmesinde silinmeyeceklerdir.

Sisteme ayrılan bölüm, göstergeleri ve işletim sistemiyle etkileşimi sağlayan diğer tüm bilgileri içerir.

''Kitaplık birimleri'' olarak adlandırılan bölüm, kullanıcı program tarafından çağırılan özel BASIC fonksiyonlarını barındırır. Uzun bir araştırma ve bağlama işleminden kurtulmak için, bu bölümde, gerekenden çok fonksiyon saklanır. Her kullanıcı program için yalnızca gerçekten gerekli olan alt-programlar seçilip kullanılması durumunda bile, uzun seçme ve bağlama işlemleri gerekir. Bu işlemler, sistemin bazı alt-programlarının diğerlerini çağırması nedeniyle daha da karmaşıklaşır. Bunu engellemek amacıyla seçmeci bir yükleme yapabilmek için, göstergelerden oluşan karmaşık bir zincir hazırlamak gerekir. Bunun yerine kullanılan yöntem, sistem alt-programlarını ilk kategoriye ayırmaktır: Bir yanda en sık başvurulanlar, diğer yanda ise ender olarak kullanılan fonksiyonlar.

Bağlama aşamasında her zaman kullanılan fonksiyonların saklandığı kitaplık (BRUN. COM) sistematik olarak, diğeri ise (BAS-LIB.REL) yalnızca gerektiğinde yüklenir.

Herhangi bir program, çok küçük de olsa en azından 16 KB'lik bir bellek alanını kaplamak zorundadır. Çünkü bu, sistem alt-programlarını içeren birimin kapladığı alandır.

BASIC 80'i Tamamlayan Diğer Özel Komutlar

Bu bölümde, BASIC dili ve makine yapısı çok iyi bilinmeden anlaşılamayacak bazı özel komut ve güdümlere değineceğiz. Önce genel kullanıma yönelik dört komutu ele alıp, daha sonra da bazı çizim işlevleri ve özel çevresel gereçlerin yönetimiyle ilgili sorunları inceleyeceğiz.

Genel Kullanıma Yönelik Komutlar

BASIC komutlarını sergilerken, bölümleme, programları birbirine ekleme ve alt-programları çağırmaya ilişkin bazı genel noktalara değinmiştik. Şimdi bu konuyu tamamlayan üç komutu da aşağıda anlatacağız:

CALL
COMMON
USR

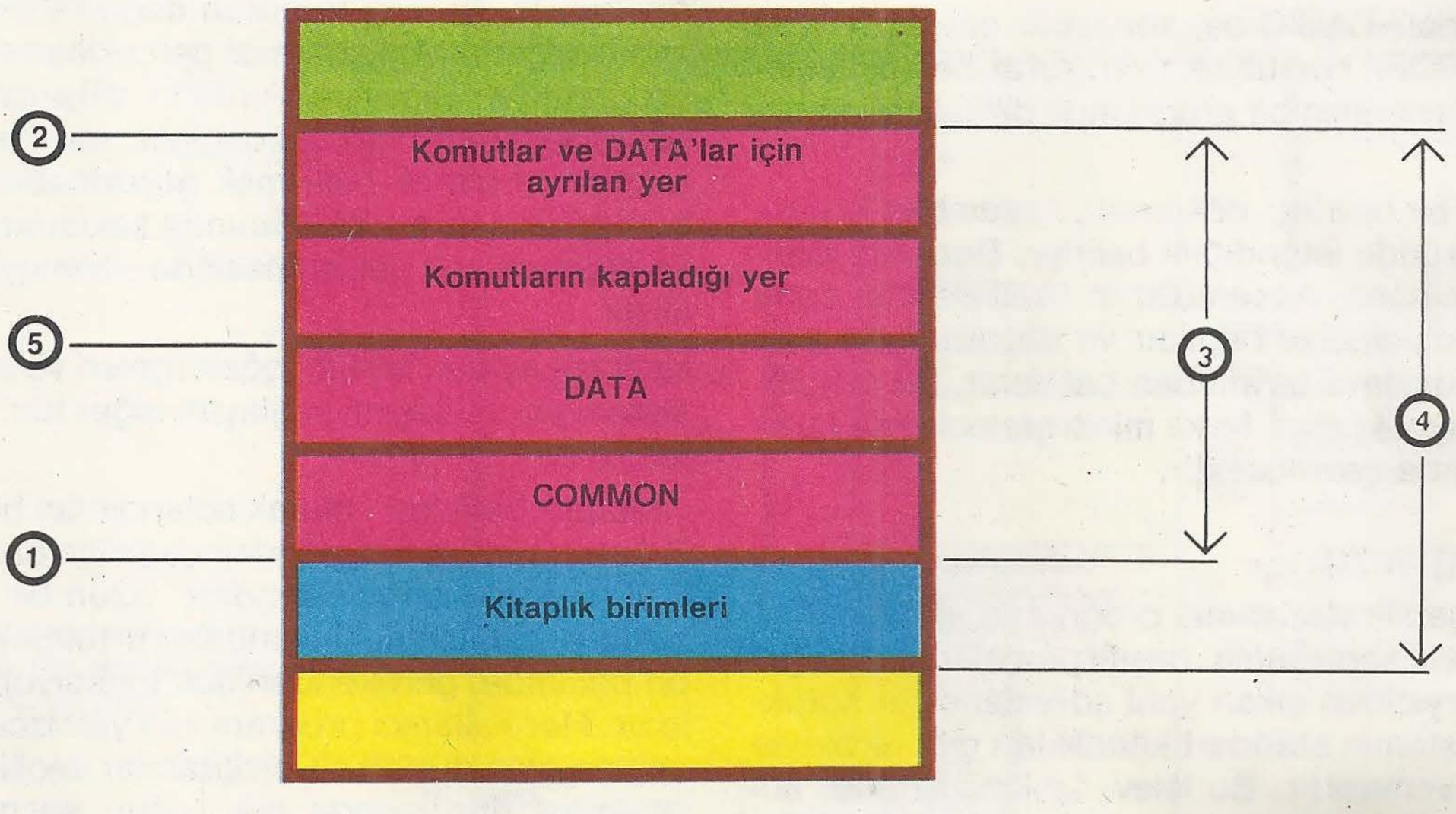
STOP komutu ise daha sonra açıklanacak.

CALL. Assembler dilinde yazılmış bir alt-programı çağırmaya yarar. Komutun sözdizimi şöyledir:

CALL değişken (argümanlar)

Burada değişken çağırılan alt-programın adresini içeren simgesel bir ad, argümanlar ise bu (dış) alt-programa aktarılarak ve böylece

BELLEĞİN PAYLAŞIM HARİTASI



Ham programın komutlarının, DATA ve COMMON'ın kapladığı yer.

Program tarafından çağrılan kitaplık birimlerine (örn. fonksiyonlara) ayrılan yer.

Sisteme ayrılan yer.

Kullanılmayan alan

TEST 20

1 / Disket yönetimi ile ilgili bu alt-program, ana programın birkaç noktasından çağrılmakta ve ona programı bellekte saklayacak veriyi (BF$) ve kayıt numarasını (REC) iletmektedir.

```
110 OPEN ''R'',1, ''A:DENEME'', 128
120 FIELD 1,128 A$ A$
130 LSET A$ = BF$
140 PUT 1, REC
```

İçerdiği hataları bulunuz.

2 / Aşağıdaki önermelerin hangileri doğrudur?

a) Derlenmiş bir program, BASIC dilindeki bir ham programdan daha çok bellek alanını kaplar.
b) Derleyici, çeviricinin atladığı hataları da belirleyebilir.
c) Derlenmiş bir program kopya edilemez.

3 / Ham bir programdan, çalışır bir program elde edebilmek için hangi işlemler yerine getirilmelidir?.

4 / Derleyici bilgisayarın türüne mi bağlıdır, yoksa aynı derleyici her makinede kullanılabilir mi?

Test 20'nin çözümleri s.737'dedir.

onun tarafından kullanılabilecek değişkenlerin bir dökümüdür.

Örneğin şu komutlar:

```
10 SUBA = &HA000
20 CALL SUBA(I,X)
```

Assembler dilinde yazılmış, A000 bellek adresinden başlayan (&H simgesi, onu izleyen sayısal değerin 16 tabanlı olduğunu belirtir) bir alt-programı çağırırken, I ve X değişkenlerindeki değerleri de ona iletmektedir. İletilecek değerler, daha önceden sözkonusu değişkenlere verilmiş olmalıdır. Genelde sabit bir değeri olduğu gibi aktarmak mümkün değildir.

Bu değer önce bir değişkene verilip, sonra değişken iletilir.

Örneğin B010 adresinden başlayan bir alt-programa 157 değerini aktarmayı sağlayan komutlar şunlardır:

```
10 V = 157
20 ADR = &HB010
20 CALL ADR(V)
```

Diğer dillerde, özellikle de FORTRAN'da, CALL kodu BASIC'deki GOSUB'a benzeri ana programla aynı dilden olan bir alt-programı ça-

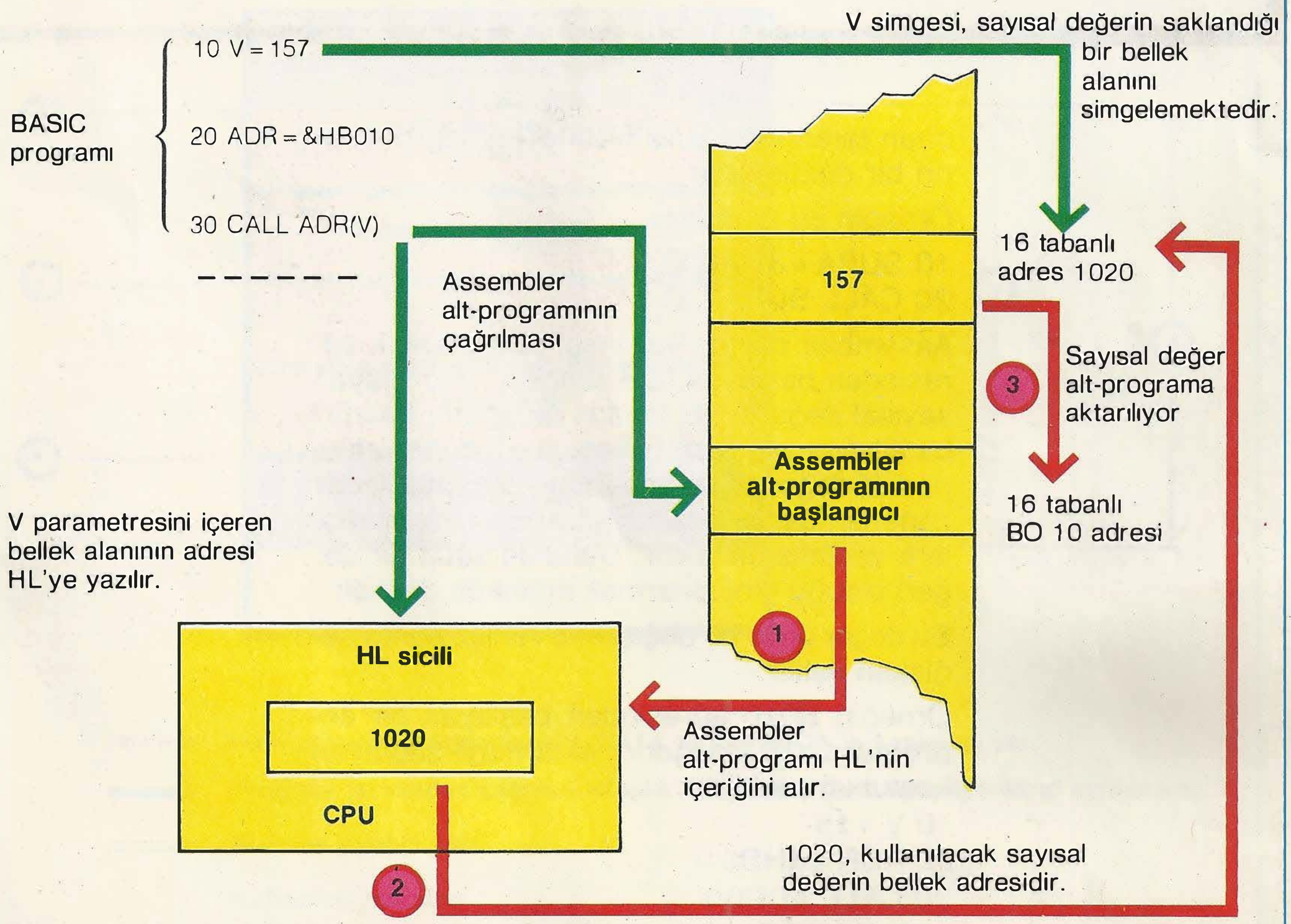
ğırır. CALL ADR(V) komutu, denetimi ADR(V) adlı programa bırakırken V değişkenine verilmiş olan 157 değerini de ona aktarmaktadır.

Derlenmiş BASIC'de, çağırılan alt-programın başlangıç adresinin sayısal olarak tanımlanması şart değildir: Bu adres bir adla simgelenir ve ana program yerini alt-programa bırakırken Bağlayıcı bunu adresler tablosundan arar.

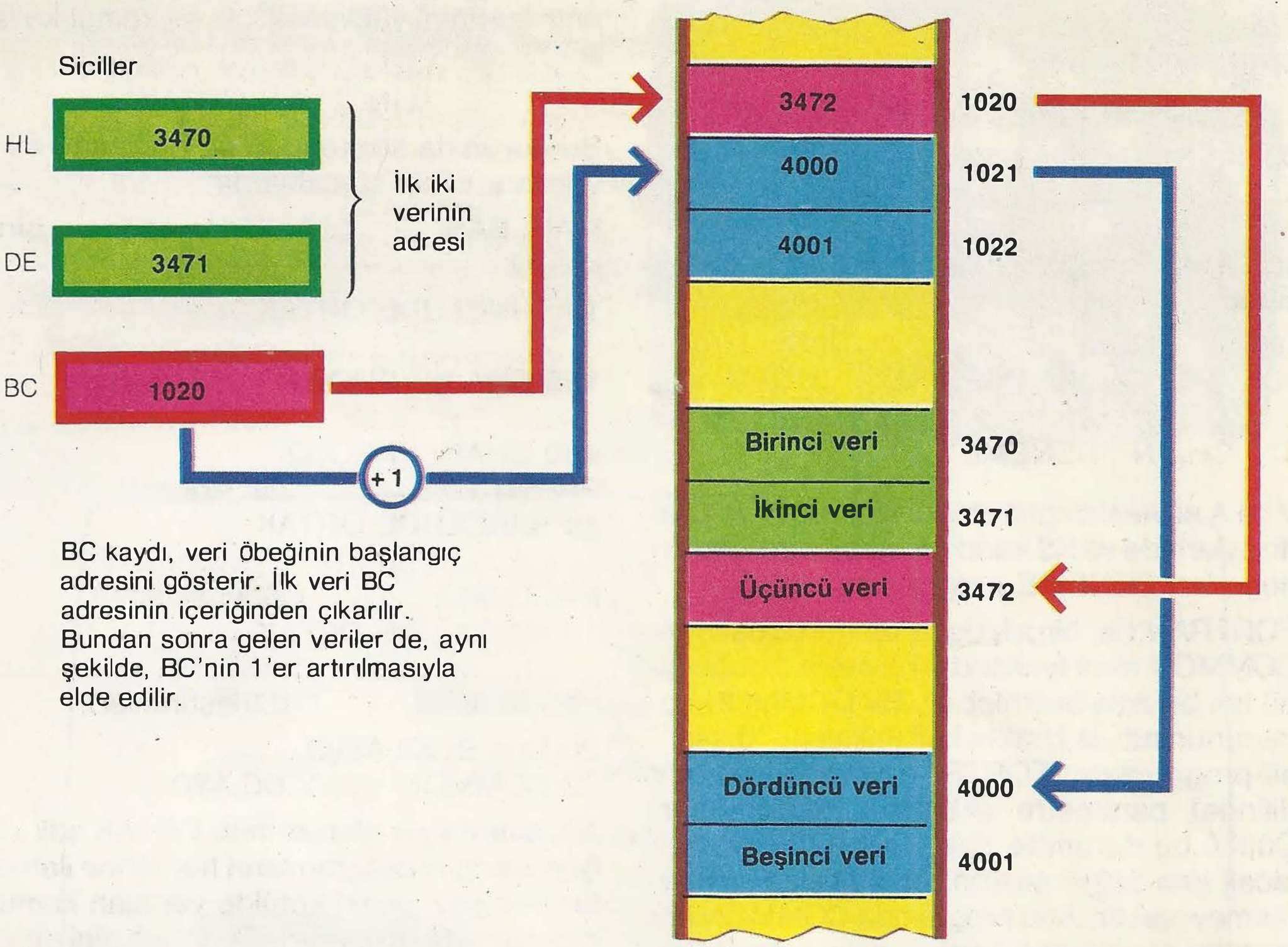
CALL komutunun bir başka özelliği de, BASIC ile ulaşılamayan bellek alanlarına ulaşımı sağlamasıdır: Bazı çevirici ve derleyiciler, çalışma alanlarını belleğin belli bir bölümü ile sınırlarlar. Bu durumda geri kalan alanlara, bir veya birçok Assembler programı yazıp, daha sonra bunları BASIC'de CALL komutu ile çağırmak olanaklıdır.

Bir yüksek düzeyli dilde kolayca formüle edilebilen parametre iletimi, Assembler'deki karmaşık aktarım mekanizmaları yüzünden bazı güçlükler taşımaktadır. Argümanlar, aslında

CALL KOMUTUYLA BİR PARAMETRE AKTARIMI



5 PARAMETRENİN AKTARILMASININ MANTIK ŞEMASI



BC kaydı, veri öbeğinin başlangıç adresini gösterir. İlk veri BC adresinin içeriğinden çıkarılır. Bundan sonra gelen veriler de, aynı şekilde, BC'nin 1'er artırılmasıyla elde edilir.

içlerinde belli değerler bulunan bellek adreslerini gösterirler. Her adres iki bayt olarak kodlanmıştır. Böylece veri türü ne olursa olsun, (tamsayı, yer gerçek sayı, vs.) parametreler hep iki baytlık yer tutar. BASIC'ten Assembler'a parametre iletirken, bu parametrelerin sayısına göre farklı yollar izlenir.

Azami üç parametreye kadar iletim için sicilden* yararlanılır. Aksi halde bitişik bellek alanlarından oluşan bir öbek kullanılır. Örneğin, iki parametrenin aktarımı sırasında, birinci adres HL'ye**, ikincisi ise DE'ye (HL ve DE iki sicil adıdır) yüklenir.

Assembler programı, bu iki sicilin içeriğini alıp, bunları karşılık gelen bellek alanındaki değişkenin değerini bulmak için adres olarak kullanır. Öyleyse CALL ADR(V), HL siciline, V değişkenine ilişkin değeri içeren alanın adresini yüklemektedir. Assembler alt-programı, HL'nin içeriğini bir adres gibi ele alıp, belirtilen bellek alanındaki sayısal değeri bulur. Sayfa 734'teki şema bu süreci göstermektedir.

Üçten fazla parametrenin aktarımı ise daha karmaşıktır ve bellek paylaşımı ile ilgili sistem alt-programlarına başvurmayı zorunlu kılar.

Genelde, ilk iki parametre, deminki gibi iki sicilin yardımı ile taşınır. Diğer veriler ise, bir veri öbeği oluşturan bir bellek bölgesinde saklanır. Değerleri veri öbeğinden alabilmek için, göstergeler için öbekte ayrılmış olan en düşük adres yeni bir sicile aktarılmalıdır (örneğin BC). Değerlerin okunması, (BC) sicilinin içeriğinin alınması ve adres olarak kullanılmasıyla gerçekleşir. Bu şekilde "belirlenmiş" (göstergelenmiş) I bellek alanında ilk değer içerilmektedir, bir sonraki bellekte I + 1 pozisyonundadır vd.

Yukarıdaki şema, aktarma için kullanılan bu ikinci mantığı ayrıntılarıyla göstermektedir.

---

* Hatırlayacağınız gibi, siciller MİB'de yer alan ve belirli adlarla simgelenen özel bellek alanlarıdır.

** HL, bitişik olarak kullanılan bir çift sicildir (H + L).

COMMON. Bu komut, bellekte bulunan bir programın bazı parametrelerini, CHAIN ile çağırılmış olan bir diğerine aktarmak için kullanılır.

Sözdizimi şöyledir:

COMMON Değişkenlerin listesi

Dökümde yer alan değişkenler bir programdan diğerine aktarılırken sıfırlanmazlar. Bir dizeyi aktarmak için, adından sonra () simgelerini de eklemek gerekmektedir. Örneğin şu komutlar:

```
10 DIM B(30),E(9)
20 COMMON V,A,B()E(),K$
.....
276 CHAIN ''DENEME'',10
```

V ve A sayısal değişkenlerinde, B(30) ve E(9) dizeylerinde ve K$ karakter zincirinde içerilen değerleri DENEME programına aktarır.

FORTRAN'da, birçok değişkenden oluşan bir COMMON alanı (yukarıdaki örnekte olduğu gibi) tek bir adla belirtilebilir. Bir COMMON komutunun adıyla birlikte kullanılması, ''dışsal'' alt-programlara (FORTRAN veya Assembler dilinde) parametre aktarımını kolaylaştırır. Çünkü bu durumda, CALL komutuna aktarılacak tüm değişkenlerin listesini eklemek gerekmeyecektir. Ana programda COMMON komutuyla tanımlanan ''ortak'' adın anılması yeterlidir.

Komutun sözdizimi şöyledir:

COMMON/ADI/ VI,B(),VK

Burada yer alan VI ve VK değişkenleri ve B dizeyi ADI terimiyle özetlenmiştir. Burada çağırılan programlardan COMMON komutuna gönderme yapmak için, bu adın belirtilmesi yetecektir. FORTRAN'ın bu özelliği, BASIC'-in bazı geliştirilmiş versiyonlarında da programların zincirlenmesinde kullanılır. Aynı satırların tekrarlanmasına engel olmak için, (eklenen her program için), özel bir BASIC komutu (%INCLUDE) kullanılır. Bu komut çağırılan programa, ayrı bir kütükte bulunan COMMON komutundaki değişkenlerden yararlanma olanağı verir. Böylece değişkenler bir kez yazıldıktan sonra, her gerektiğinde tekrar tekrar yüklenir.

Örneğin, zincirleme eklenecek programların MAIN, PROG1, PROG2 olduğunu, BAS türünden ayrı ayrı kütüklerde saklandıklarını ve ortak olarak V1, V2 B(30) ve A$(6) değişkenlerini kullandıklarını varsayalım.

COMMON komutunun içerdiği birleştirimler ve dizeyler ve dizeylerin boyutlandırmaları ayrı bir kütükte, örneğin ORTAK'ta saklanmaktadır. Bu kütüğün içine eklenen programların her birinin içeriğini yüklemek için şu komut kullanılır:

%INCLUDE ORTAK

Bu durumda sözkonusu üç program, şu tür yapılara sahip olacaklardır.

MAIN.BAS (diğerlerini çağıran birinci kütük)

```
10 %INCLUDE ORTAK.
.....
165 CHAIN ''PROG1''
.....
460 CHAIN ''PROG2''
```

PROG1.BAS (ilk ekleme)

```
10 %INCLUDE ORTAK
.....
```

PROG.BAS (ikinci ekleme)

```
10 %INCLUDE ORTAK
.....
```

ORTAK.BAS (birleştirimler)

```
20 DIM B(30),A$(6)
30 COMMON V1,V2,B(),A$()
```

10. satırda yer alan komut, ORTAK adlı kütüğün içeriğini programların her birine iletecektir. Her şey, sanki kütükte yer alan komutlar diğerlerinde de tekrarlanıyormuş gibi gerçekleşecektir.

Dikkat: %INCLUDE komutu sisteme bağlıdır ve bu nedenle farklı makinelerde farklı şekiller alabilir veya hiç bulunmayabilir.

USR. Assembler dilindeki bir alt-programın başlangıç adresini tanımlar. Sözdizimi şöyledir:

DEF USRN = DEĞER

Burada N (0'dan 9'a) programda tanımlanan alt-programlardan birisini işaret etmektedir (bu biçimde, her Assembler alt-programına bir numara verilir). DEĞER ise başlangıç adresidir.

USRN komutunun amacı, BASIC programının içinde, Assembler alt-programlarını sanki bunlar fonksiyonlarmışcasına tanımlamaktır. DEF USRN, DEF FNX'e benzer: İkincisi BASIC dilinde bir fonksiyonu tanımlar. USRN ise bir BASIC fonksiyonunu bir Assembler alt-programı olarak kullanır.

Örneğin, DEF USR3 = 1740 komutu, 1740 bellek adresinden başlayan Assembler alt-programını tanımlar ve ona 3 numarasını verir. Daha sonra bu alt-program, tipik bir fonk-

TEST 20'nin Çözümleri

1 / İki hata var. Birincisi çok temel bir hata olup programın çalışmasını önler. İkincisi ise ancak belli şartlar altında ortaya çıkar.

Birinci hata: 110 komutu (OPEN) alt-programın her çağırılışında yerine getirilir. İlk defasında, komut doğrudur ve bir kütüğün açılmasını (veya yaratılmasını) sağlar. Ancak ikinci defasında zaten açık olan bir kütüğü bir kez daha açmaya çalışacağı için, bir sistem hatası ortaya çıkar.

İkinci hata: Diskete yazma ya da disketten okuma aşamalarında, veri kaydı numarasının (REC, komut 140), 1 ile kütük için öngörülen en yüksek sayı arasında yer alıp almadığını denetlemek yerinde olacaktır. Alt-programın çağırılması sırasında, REC=0 durumu ortaya çıkabilir. Bu durumda, 0 kaydı bulunmadığı için yürüme duracaktır (bazı makinelerde ise, kütükler 0 numaralı kayıtla başlar). Diğer yandan, REC değişkenine kütükte yer alan azami kayıt sayısından da büyük olan bir değer verilirse, bu defa disket üzerinde kütüğe ait olmayan başka bölümlere taşma olacağından, diğer verilerin silinmesi tehlikesi doğar.

---

2 / a) yanlış, b) doğru, c) önermesi ise şöyle yorumlanmalıdır: Derlenmiş program ham bir program gibi kopya edilemez, ancak başka bir disket üzerine kopya edilebilir. Özel bazı önlemlerle, bir programın kopya edilmesi yasaklanabilir veya sınırlanabilir. Burada yalnızca çok özel programlar için kullanılan çok karmaşık yöntemler sözkonusudur.

---

3 / Yürütülecek işlemler şöyle sıralanabilir:
1- Ham BASIC programının yazılması ve ASCII düzeninde, diskette saklanması
2- Derleyicinin çalıştırılması
3- Bağlayıcının çalıştırılması

Böylece, bir disket üzerinde saklanan ve çalıştırılabilir bir program elde edilir. Çalıştırmak için (işletim sisteminde) adını girmek yeterlidir.

---

4 / Derleyici büyük ölçüde makineye, daha doğrusu kullanılan merkezi işletim birimine bağlıdır.

siyon gibi çağırılıbilir. X=USR3(K) komutu, K parametresinden yararlanarak, 3 Assembler alt-programınca yürütülen işlemin sonucunu X değişkenine verir.

(K) argümanı her türden olabilir (sayısal veya karakter zinciri). Hangi türün sözkonusu olduğu ise, Assembler alt-programına özel bir sicille, A siciliyle haber verilir. Örneğin A'nın alacağı 2 değeri tamsayıyı, 3 değeri karakter zincirini vb. gösterir.

Argümanın (K'nin) değeri ise her iki farklı biçimde aktarılabilir. Sayısal değerler için, bir çift sicil değerin bellek adresini gösterir. Karakter zincirlerinde ise, fazladan iki sicil, zinciri tanımlayan adresi göstermek için kullanılır.

Genelde bu tanımlama öbeği 3 bayttan oluşur: Birincisi zincirin karakter cinsinden uzunluğu (0'dan 255'e) gösterir, diğer ikisi ise değerlerin (zincirdeki karakterlerin) bulunduğu bellek alanının başlangıç adresini taşır.

Sayısal bir veri sözkonusu olduğunda, kayıtlar doğrudan değerin bulunduğu bellek adresini verir ve argümanın değeri bir geçişte elde edilmiş olur. Zincirlerde ise, kayıtlar verinin adresinin bulunduğu tanımlayıcının adresini içerdikleri için, ikinci bir okuma daha gereklidir. Genelde, zincirleri tanımlayan öbek ve zincirlerin kendisi, kullanıcıya ait bellek bölümünde saklıdır. Buna karşılık, sayısal veriler Floating Point Accumulator (FAC: Kayar virgül Depolayıcısı) adı verilen özel bölümlerde saklanırlar. Adres kayıtları da bu depolayıcı üzerindeki özel bir alanı gösterirler.

BASIC'in bazı versiyonlarında, Assembler alt-programları DEF USRN komutuyla tanımlanamazlar. Bu durumda, her bir alt-program çağırılmadan önce, ayrılmış bellek alanlarına başlangıç adresleri yollanır. Bu alanların simgesel adı, kullanılan BASIC versiyonuna göre değişir. Bir alt-programı çağırmak için önce, POKE komutu kullanılarak başlangıç adresi, bu iş için ayrılmış bellek alanına yazılır. Bu tür BASIC versiyonları, alt-programı, numarasını belirterek seçme olanağı vermezler. Alt-program yalnızca POKE komutuyla başlangıç adresi yazılarak çağırılabilir. Bir diğer alt-programı çalıştırmak için ise, önce, başlangıç adresini aktarma işlemini tamamlamış olmak gerekir.

Örnek olarak sayfa 739'da dökümü bulunan program, POKE ve PEEK komutlarını kullanarak, 5 renk simgesi arasından ikisini seçip, yazmakta (satır 90) ve sonra her biri için, seçilen rengi okumaktadır. Program bir COMMODORE VIC-20'de çalıştırılmaktadır. Adı geçen adresler de yalnızca bu makine için geçerlidir.

STOP. Üstlendiği işlev, bir programın çalışmasını durdurmak ve aynı END komutunda olduğu gibi güdüm konumuna dönüşü sağlamaktır. Aradaki fark, sistem tarafından verilen ve STOP komutunun bulunduğu satır numarasını bildiren mesajdır.

Bu komutun en yaygın kullanımı, çok önemli hatalarda programın çalışmasını durdurmaktır.

Normal programın sonu daima END komutuyla tanımlanır. (Bu komut programın içinde yalnız bir tane olmalıdır ve programın olası tüm çıkışiarı bu tek END satırına adreslenmelidir.) Buna karşın hataların ortaya çıkması durumunda STOP komutuyla birden fazla çıkış elde edilebilir.

STOP komutuyla karşılaşıldığında eğer programın akışı askıya alınırsa, o anda kullanılan kütükler kapanmaz (yani, END komutuyla oluşan durumun tersi). Programın çalışması, CONT (devam) güdümü girilerek, bir sonraki satırdan devam ettirilebilir. Programı geçici olarak durdurmak, bize etkileşimli güdümler aracılığıyla olup biteni öğrenme şansı verir. Yapılan düzeltmeden sonra (eğer gerekiyorsa) çalışma devam eder. Örneğin, bir programın önce D1, D2 ve D3 değişkenlerinin değerlerini hesaplayıp, sonra da bunları bölme işlemlerinde payda olarak kullanıldığını varsayalım. Böyle bir durumda, değişkenlerin (paydaların) sıfır olmadığından emin olmamız gerekir. Bu paydaların hesaplanmasındaki bir hata, sonraki komutların yürütülmesinde de kendini gösterecektir. Gerekli denetimler, IF ve STOP komutlarıyla, örneğin aşağıdaki biçimde yapılabilir.

```
IF D1 = 0 THEN STOP
IF D2 = 0 THEN STOP
IF D3 = 0 THEN STOP
```

Bu hataların biriyle karşılaşıldığında, programın yürüyüşü kesilir, ekranda STOP komutunu içeren son komutun numarası belirir. Bu komut aynı zamanda yürütülmüş olan son komuttur. Bu noktada güdüm işletim sistemine geçmiş olduğu için etkileşimli konumda hatalı bölünmeyle ilişkin tüm değişkenler incelenip, daha sonra da, sonuçlar gözlenmek üzere program kaldığı yerden yürütülebilir; ya da hatalı paydaya rasgele bir değer verilerek yürüme devam ettirilebilir. Ama bu, son programda düzeltmeler yapmak mümkün değildir.

Çizimlerin Komutları

BASIC'in birçok biçimi, çizisel gösterim için özel komutlar içerir. Konu daha sonra derinlemesine incelenecektir. Bu bölümde ise, uygulamalarına fazla ayrıntılı biçimde değinmeden, başlıca çizim komutları ele alınacaktır. Çizim konumunda, ekran parçacıklara (PİXEL-picture elements, görüntü ögeleri) bölünür. 80 karakter sütununun yerine ekranda 512 parçacık, 24 satırın yerine de 256 parçacık vardır. Aşağıdaki komut:

```
HPLOT 10, 5
```

ekran üzerinde (10,5) koordinatlarına sahip parçacığı ışıklandırır. Bir doğru parçasının

POKE ve PEEK KOMUTLARININ KULLANIMINA ÖRNEK

```
0 REM: MAIN (POKE VE PEEK KULLANIMI ICINORNEK
7 PRINT "J": POKE 36879,29
10 DIM VSUT(5), NISUT$(5)
20 FOR I=1 TO 5
30 READ VSUT(I)
40 NEXT I
50 DATA 0,2,5,7,6
60 FOR I=1 TO 5
70 READ NISUT$(I)
80 NEXT I
90 DATA SIYAH,KIRMIZI,YESIL,SARI,MAVI
100 INPUT "BIRINCI RENK"; SUT$
110 K1=0
120 FOR I=1 TO 5
130 IF SUT$= NISUT$ (I) THEN K1=I
140 NEXT I
150 IF K1=0 THEN PRINT "HATA":GOTO 100
160 N1= VSUT(K1)
170 INPUT "IKINCI RENK"; SUT1$
180 K2=0
190 FOR I=1 TO 5
200 IF SUT1$=NISUT$(I) THEN K2=I
210 NEXT I
220 IF K2=0 THEN PRINT "HATA":GOTO 170
225 N2=VSUT(K2)
230 PRINT "RENK KONTROLU"
240 A=PEEK(38760):B=PEEK(38764)
270 PRINT "ZEMIN=";NISUT$(K1)
275 PRINT "EKRAN=";NISUT$(K2)
280 POKE 8040,83
290 POKE 53280,N1
300 POKE 8044,83
310 POKE 53281,N2
320 IF B=N2 THEN PRINT "RENK=";NISUT$(K2):GOTO 360
360 END
```

gösterimi, imleç bu parçanın üzerinde gezdirilerek sağlanır. İki uç noktanın koordinatları, parçacıklar olarak bildirilmelidir. Örneğin şu komutların:

```
LINE (10,5)-(70,100)
HPLOT 10,5 TO 70,100
```

her ikisi de, (10,5) koordinatlarından (70,100)'e uzanan bir doğru çizerler. Bu türden komutlar ardarda sıralanarak, çok değişik şekiller çizdirilebilir. Örneğin bir dörtgen çizmek için, bu komutlar dört kez tekrarlanmalıdır. Bazı sistemlerde, ardarda kullanımlar 2 tabanında bir diskette saklanabilir. Bu durumda tek bir komutla tümü birden belleğe çağırılır. DRAW-''n'' komutu, ekran üzerinde, daha önceden belleğe alınan n numaralı ''çizim kütüğünü'' çizer. Bu kütük, birim çizgilerden hareketle ulaşılan çok karmaşık şekiller içerebilir.

Bu yöntemi kullanan hemen tüm sistemler, ölçek değiştirme, desen kaydırma ya da ekranın bazı bölümlerini renklendirme komutlarına da sahiptirler. Hemen tüm makinelerde

(değişik sözdizimleriyle) varolan bu komutların, kullanımıyla ilgili örnek programlar, bilgisayarla çizimin incelendiği bölümde gösterilecektir.

Özel Komutlar

BASIC dili, son derece basit olması nedeniyle hemen tüm kişisel bilgisayarlarda kullanılır. Böylesine yaygın bir kullanım, üreticileri, çeşitli çevresel gereçlere ulaşabilmenin yollarını aramaya ve çeşitli ara birimleri geliştirmeye zorlamıştır.

Video oyunundan kaynaklanan bu çevresel gereçler daha sonra profesyonel olarak da kullanılmaya başlamış ve kullanıcı ile makine arasında mükemmel bir iletişim kurulması başarılmıştır.

Ses Üretimi. Ses üretimi konusu, basit bir ''bip'' sesinden, insan sesinin bileşimine değin uzanan ve bilgisayara cümleler kurma olanağı veren çok boyutlu ve karmaşık bir konudur. Özel donanımları gerektiren en karmaşık uygulamalar, daha ilerde sergilenecektir. Bu bölümde ise, basit müzik sesleri çıkarmaya yarayan bazı BASIC komutları ele alınacaktır.

Bu komutların sözdizimi, sistemden sisteme değişir. Gene de bunları iki farklı kümeye ayırmak mümkündür: Yüksek düzeyli komutlara bağlı, özel sistem alt-programları içeren makineler ve PEEK komutu veya Assembler dilinin kullanılmasını gerektiren makineler.

Bu tür bir ayırma gidilmesinin nedeni, çevrilmiş BASIC komutlarının işleme hızıdır. Bir ses, oparlöre gerekli uyarı gönderilerek üretilir. Ses inceldikçe, sinyalin frekansı* da yükselir. Belli bir değer aşıldığında ise, yorumlayıcı yeterince hızlı olamadığı için, istenilen frekans yaratılamaz. Bu durumda, Assembler dilinde alt-programlar kullanmak gerekir (Assembler dilinde ses üretimi, bu dile ayrılan bölümde incelenecektir).

Eğer sistem gereken BASIC komutlarını içermiyorsa, PEEK komutu kullanılır. Kişisel bilgisayarların çoğu, ''oparlör anahtarı'' gibi çalışan özel bir bellek bölümüne sahiptir. Bu adrese ulaşıldığında, bir sinyal yayınlanır. Belli incelikteki bir sesi elde edebilmek için, (örneğin DO sesini), oparlöre ayrılan bir bölümü belirli bir süre ardarda uyarmak gerekir. Süreler gerektiği gibi ayarlanırsa, düşük frekanslarda olma koşuluyla, istenilen ses çıkartılabilir.

Daha yüksek frekanslar için ise, aynı yöntem, yalnız bu sefer Assembler dilinde yazılmış bir programla uygulanır.

Bazı sistemlerde, bir BASIC komutu gibi çağırılabilen programlar mevcuttur. Örneğin,

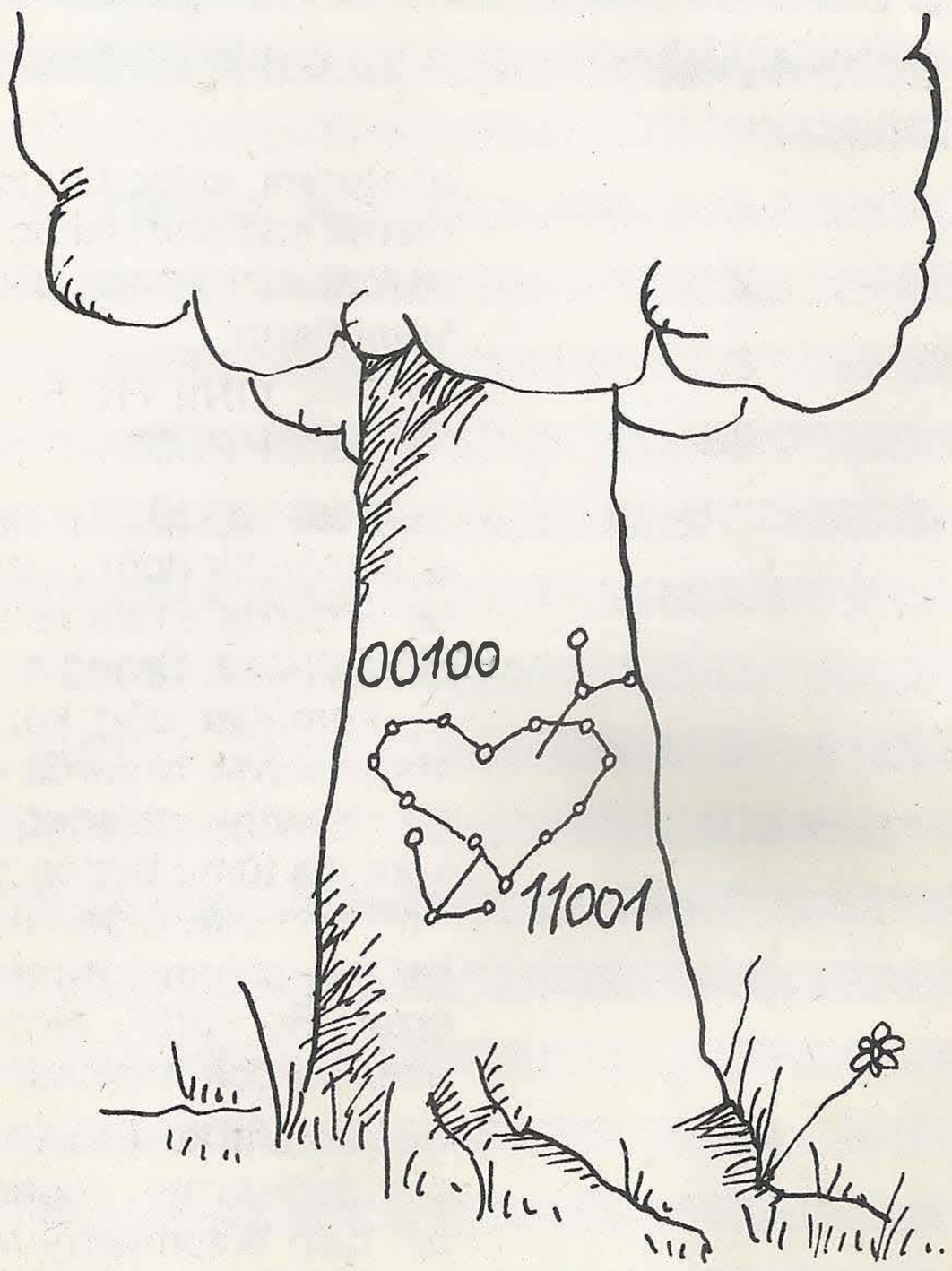
SOUND f,s

komutu, f frekansında (genelde 35 ila 32000 Hz arasında) ve s süreyle (saniye cinsinden) bir sesin üretilmesini sağlar.

PLAY ise, müzik parçaları çalmaya olanak tanıyan, ancak programcının müzik bilgisini gerektiren özel parametrelere dayanan, daha karmaşık bir komuttur.

Joystick (Oyun çubuğu) ve pedal. Tabiî bu sözcükler, ister istemez video oyunlarını çağrıştırıyor. Aslında, kumanda aletlerindeki hareketleri sinyallere dönüştüren ve bazen çok yararlı olabilen özel işlevlerle donatılmış ara birimler sözkonusudur.

Analog/sayısal (A/S) çevirici, bir fiziksel büyüklüğü, bu büyüklükle orantılı bir elektrik sinyaline dönüştürebilen bir aygıttır.



* Kendini tekrarlayan bir sinyalin frekansı, bir sandalyede kaç tekrar olduğunu gösterir. Birimi Hz (Herz)'dir. Örneğin, 35 Hz'de, saniyede 35 tekrar sözkonusudur. Bu tekrarların her biri, saniyenin 35'te biri kadar sürer (bu büyüklüğe de period adı verilmektedir).

Bunun dışında, daha başka çeviriciler de vardır: Örneğin, bir termometre, sıcaklığı uzunluğa çeviren (cıva sütununun boyu) bir çeviricidir. Bilgisayar, analog/sayısal (A/S) çevericileri gerektirir. Makine devreleri tarafından yorumlanabilmek amacıyla, herhangi bir fiziksel büyüklük, (analog) elektrik sinyaline (sayısal) dönüştürülür.

En basit A/S çeviricisi pedal (paddle)'dır. Bu aygıt yer değiştirmeyle orantılı bir sinyal üretir. Buna karşılık gelen BASIC işlevi olan PDL(n), 0 ile 285 aralığı arasında bir tamsayı yollayarak, gerçekleştirilen yer değiştirme ''miktarını'' bildirir. n sayısı, çeviricinin bağlı olduğu giriş/çıkış bağlantı yerini simgeler. Örneğin,

V = PDL(2)

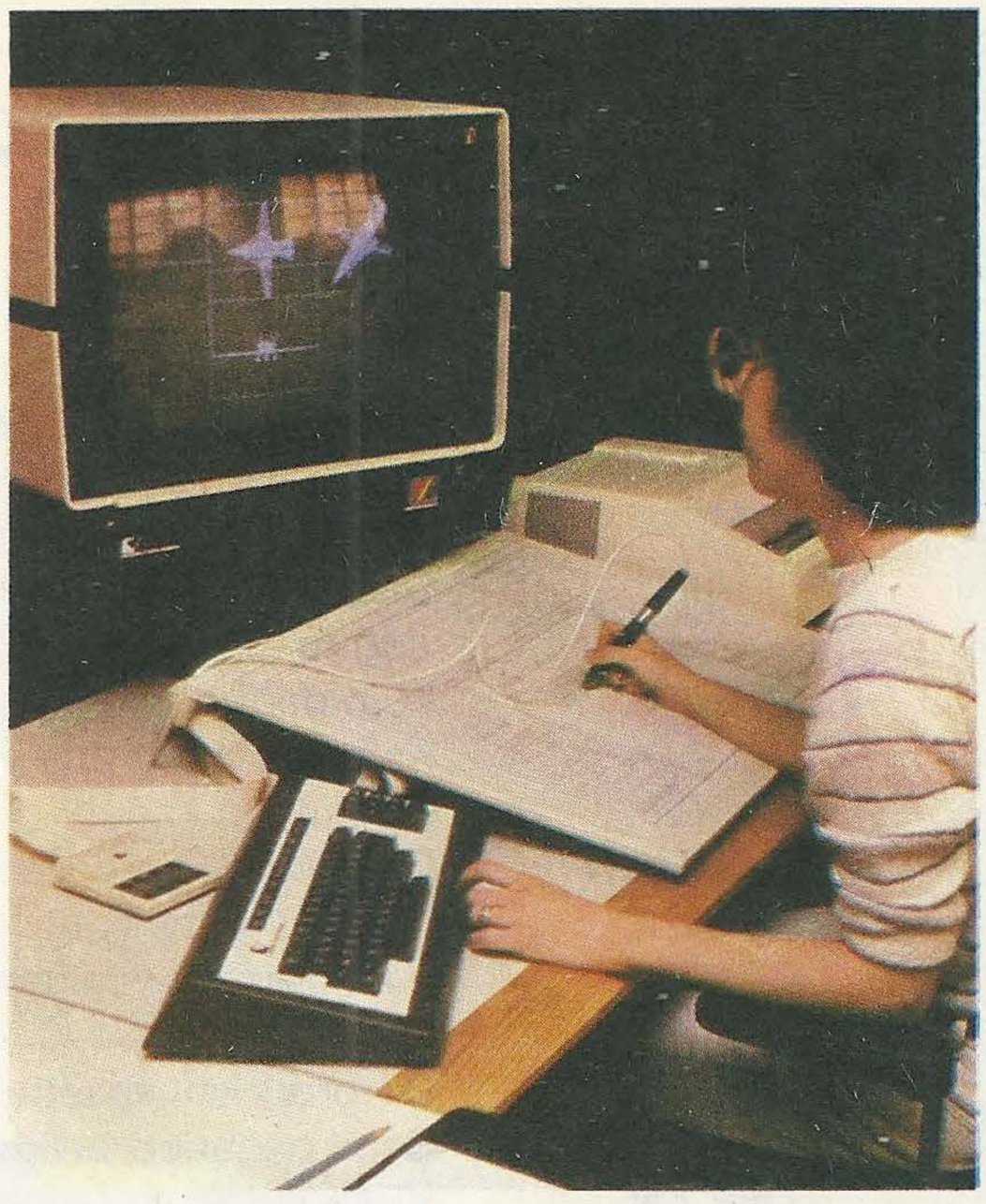
komutu, V'ye 2 bağlantı yerine bağlı pedaldan gelen değeri aktarır. Bu değer bir kez okunduktan sonra, ekrana bir simge yerleştirmekte kullanılır. Bu okuma ve imleci konumlandırma işlemleri yinelendiğinde (bir döngüyle) pedaldaki hareket hemen hemen kesiksiz olarak ekrana yansıtılmış olur. Pedal bir doğru boyunca harekete, yani imleci ekrandaki bir satırın üzerinde dolaştırmaya olanak verir.

Her iki boyutta da (yani tüm ekran üzerinde) yer değiştirme olanağını ise, oyun çubuğu (joystick) verir. Joystick, biri yatay biri dikey olmak üzere iki pedalı olan çift yönlü çubuktur. Bu çubuğun biçimi makineden makineye değişebilir. Başlıca işlevleri şunlardır:

STICK(n). n çizgisi üzerinde, oyun çubuğuna bağlı koordinatları verir. Örneğin, STICK(0), yatay çizgi üzerindeki yer değiştirmeleri. STICK(1) ise düşey çizgilerdeki verir. Bu durumda, ulaşılan noktanın koordinatları şu komutlarla elde edilir:

X = STICK(0)
Y = STICK(1)

STRING. Çubuğun durumunu (video oyunlarında vurma, sıçrama gibi durumları canlandırmak üzere) bildirir. Genelde, -1 çubuğun hareketini, 0 ise aksini gösterir. Bazı makinelerde, birden fazla oyun çubuğu kullanabilmek veya istekler cevaplanana kadar yürüyüşü askıya almak, ya da oyun çubuğu devreye sokup çıkarmak (STRING OFF, STRING ON) amacıyla, bir de parametre öngörülmüştür. Bu durumda komut STRING(n) biçimini alır.



Işıklı kalemle proje çizimi. Kullanıcı, bir F-15 savaş uçağının resmi üzerinde çalışıyor.

ON STRING... yukarıdaki komutun geliştirilmiş biçimi olan bu komut, çubuğun durumuna göre bir alt-program çağırmaya yarar. ON STRING(2), GOSUB 1000 komutu, 2 çubuğunu hareket ettirince 1000 alt-programını çağırır.

Işıklı kalem. Işıklı kalem, bir çizim aygıtından hareketle, grafik çıkış elde etmeye yarar. Bazı BASIC versiyonlarını, bu kalemi devreye sokan ve koordinatlarını bulan komutlar içerir. Başlıcaları şunlardır:

PEN ON Kalemi devreye sokar.

PEN OFF. Kalemi devreden çıkarır.

Z = PEN(N). Koordinatları saptar. Z değişkeninin alacağı değer, N parametresine bağlıdır. N = 1 iken, z, x koordinatını, N = 2 iken de z, y koordinatını verir. N'in alacağı değerler ve bunlara karşılık gelen komutlar büyük ölçüde makineye bağlıdır.

BASIC 80'E ÖZGÜ KOMUT VE GÜDÜMLER

Komut veya Güdüm	Etkisi	Örnek
AUTO	Program satırlarını otomatik olarak numaralandırır	AUTO 10,5
CALL	Assembler dilindeki bir programı çağırır	CALL PROG
CHAIN	Değişkenleri aktararak bir program çağırır	CHAIN ''TEST''
CLEAR	Tüm değişkenleri sıfırlar	CLEAR
CLOAD	Kasetteki programı yükler	CLOAD ''PROG''
CLOSE	Bir kütüğü kapatır	CLOSE 2
COMMON	Değişkenleri programlar arasında taşır (CHAIN ile birlikte)	COMMON A(10)
CONT	Bir programı yeniden işletir (STOP'dan sonra)	CONT
CSAVE	Bir programı kasette saklar	CSAVE ''PROG''
DATA	Sabit değerleri belirler	DATA 3,7,9
DEF FN	Kullanıcı fonksiyonlarını tanımlar	DEF FNX=A+B
DEF INT	Tamsayı değişkenleri tanımlar	DEF INT K
DEF SNG	Tek hassasiyetteki değişkenleri tanımlar	DEF SNG R
DEF DBL	Çifte hassasiyetteki değişkenleri tanımlar	DEF DBL Z
DEF STR	Karakter zincirlerini tanımlar	DEF STR A
DEF USR	Bir Assembler programının başlangıç adresini tanımlar	DEF USR2=1275
DELETE	Program satırlarını siler	DELETE 5—25
DIM	Dizeyleri boyutlandırır	DIM A(3),B$(10)
EDIT	Programı düzeltmek için biçimleme konumuna geçilir	EDIT 20
END	Bir programı bitirir	END
ERASE	Dizeyleri silerek yeniden tanımlanmalarına olanak verir	ERASE A,B$
ERR	Hataya neden olan satırı belirtir	ERROR 3
FIELD	Disketteki bir kütüğe bir veya birçok G/Ç ara belleği verir	FIELD 2,5 ASC$
FOR NEXT	Döngü	FOR I=1 TO 100 NEXT 1
GET	Bir kaydın okunmasını sağlar	GET 1,56
GOSUB	Bir iç alt-programı çağırır	GOSUB 5740
GOTO	Belirlenen satıra atlanır	GOTO 250
IF... THEN	Koşullu komut	IF K=1 THEN...
INPUT	Klavye ile veri girişi	INPUT ''veri'';A,B
INPUT #	Dizisel bir kütükten veri girişi	INPUT # 1,A
KILL	Disketteki bir kütüğü siler	KILL ''veriler''
LET	Bir değişkene değer verir	LET A=3+B
LINE INPUT	Klavyede yazılan tüm bir satırı girer	LINE INPUT A$
LINE INPUT #	Dizisel bir kütükten bir satır okur	LINE INPUT 1,A$

Komut veya Güdüm	Etkisi	Örnek
LIST	Merkez bellekte bulunan bir programın ekranda dökümünü verir	LIST 10—50
LOAD	Disketteki bir programı yükler	LOAD ''A:TEST''
LPRINT	Belirlenen değişkenleri yazar	LPRINT A,B,V
LPRINT USING	Belli bir düzenle yazar	LPRINT USING ''#.#'';A
LSET	Verileri sola yanaştırarak aktarır	LSET A$ = C$
MERGE	Disketteki bir programı bellekteki bir programa ekler	MERGE ''A:PROG''
NAME	Disketteki bir kütüğe yeni bir ad verir	NAME ''DENEME''AS ''YENİ''
NEW	Ana bellekteki programı siler	NE
NULL	Her satırın sonuna eklenecek boşluk sayısını belirler	NULL 5
ON ERROR...	Sistem hatalarını programla yönetme olanağını verir	ON ERROR GOTO 20
ON... GOTO	Bir parametreye bağımlı atlama yaptırır	ON K GOTO 10,20...
ON... GOSUB	Bir parametreye bağımlı olarak bir alt-program çağırır	ON L GOSUB 1000, 2000...
OPEN	Diskette bir kütük açar	OPEN ''I'',2,''KUT1'' OPEN ''R'',1,''KUT2''
OPTION BASE	Dizeyin ilk elemanını tanımlar	OPTION BASE 1
OUT	Çıkış bağlantısına bir bayt aktarır	OUT 2, 170
POKE	Belirli bir bellek bölümüne bir bayt yazar	POKE 2320,170
PRINT	Ekrana yazar	PRINT A,B,C$
PRINT USING	Ekrarıa belli bir düzenle yazar	PRINT USING ''#.#'';A
PRINT #	Dizisel bir kütüğe yazar	PRINT # 3,V,C$
PRINT # USING	Dizisel bir kütüğe bir düzenle yazar	PRINT # 3 USING ''#.#'';A
PUT	Bir kütüğe doğrudan erişimli bir kayıt yazar	PUT 3,26
RANDOMIZE	Raslansal sayı üreticisini çağırır	RANDOMIZE
READ	DATA ile belirlenen değeri alır	READ K,L,N$
REM	Araya açıklama sokmaya yarar. Bazı BASIC versiyonlarında aynı amaçla şu simgeler de kullanılır: ',!,*	10 REM
RENUM	Bellekteki bir programı yeniden numaralar	RENUM 10,20,5
RESTORE	Çentiği bir DATA satırının başına getirir	RESTORE 1475
RETURN	Denetimi çağıran programa bırakır	RETURN
RSET	Verileri sağa yanaştırarak aktarır	RSET A$ = B$
RUN	Programı harekete geçirir	RUN
SAVE	Bir programı diskete saklar	SAVE ''A:TEST''
STOP	Programın çalışmasını keserek güdüm moduna geçmeyi sağlar	STOP
SWAP	İki değişkenin içeriğinin yerlerini değiştirir	SWAP A$,B$

Komut veya Güdüm	Etkisi	Örnek
TRON	Programı izleme konumuna geçirir (yürütülen komutların tek tek gözlenmesi	TRON
TROFF	TRON'u yürürlükten kaldırır	TROFF
WAIT	Belirlenen bağlantıya bir veri gelene kadar bekletir	WAIT 2,2240
WHILE WEND	Bir koşul doğru kaldıkça döngüyü çalıştırır	WHILE K
WIDTH	Bilgiyazara bir satırdaki en büyük karakter sayısını bildirir	WIDTH 60
WRITE	Bir terminale yazar	WRITE A,B
WRITE #	Dizisel bir kütüğe yazar	WRITE #,2,A,B

FONKSİYONLAR

Fonksiyon	Tanımı	Örnek
ABS(V)	Mutlak değerini alır	ABS(—3*5)
ASC(A$)	Zincirin ilk karakterinin ASCII kodunu bulur	ASC(A$)
ATN(A)	A'nın arktanjantını alır	ATN(2)
CDBL(R)	Sayıyı çifte hassasiyete çevirir	CDBL(37.4)
CHR$(N)	10 tabanlı N sayısının ASCII karakterini bulur	CHR$(68)
CINT(K)	K'yı yuvarlayarak tamsayıya çevirir	CINT(4.75)
COS(A)	A'nın kosinüsünü bulur (radyan cins)	COS(2.1)
CSNG(D)	Sayıyı tek hassasiyete çevirir	CSNG(7.569114)
CVI(A$)	Zinciri tamsayıya dönüştürür	CVI(A$)
CVR(A$)	Zinciri gerçek sayıya çevirir	CVR(A$)
CVD(A$)	Zinciri çifte hassasiyette bir sayıya çevirir	CVD(A$)
EOF(N)	Dizisel bir kütüğün sonuna varıldığında —1 yollar	IF EOF(1)...
EXP(N)	e^N'yi hesaplar	EXP(3.5)
FIX (X)	X değerinin belirli kısmını budar	FIX(57.921)
FRE(N)	Ana bellekte mevcut bayt sayısını verir	FRE(0)
HEX$(M)	M'i 16 tabanlı olarak gösteren bir zincir oluşturur	HEX$(75)
INKEY$	Terminalden bir karakter kabul edilir.	A$ = INKEY$
INP(N)	N kapısından (çevresinden) bir karakter kabul edilir	A = INP(4)

Fonksiyon	Tanım	Örnek
INPUT$(N)	Klavyeden yankısız olarak N sayıda karakter kabul edilir	B$ = INPUT$(2)
INSTR(A$,B$)	A$ karakter zincirinin B$ içinde bulunup bulunmadığı sınanır ve varsa konumu bildirilir	K = INSTR(A$,B$)
INT(L)	L'nin tamsayılı karşılığını verir (<!=L)	I = INT(87.3)
LEFT$(A$,N)	Soldan başlayarak A$ karakter zincirinden N kadar karakter çıkarılır	B$ = LEFT$(A$,4)
LEN(A$)	A$ karakter zincirinin uzunluğunu verir	K = LEN(A$)
LOC(N)	Yazmak ya da okumak üzere doğrudan olarak erişilmiş son veri kaydının numarasını verir. Sıralı kütüklerde kesimlerin numarasını verir	N = LOC(1)
LOG(K)	K katsayısının doğal logaritmasını verir	R = LOG(7.21)
LPOS(X)	Bilgiyazar kafasının konumunu verir	N = LPOS(A)
MID$(A$,I,N)	A$ karakter zincirinden I konumundan başlayarak N tane karakter çıkarılır	B$ = MID$(A$,5,3)
MKI$(V)	V'yi 2 baytlık bir karakter zincirine çevirir	A$ = MKI$(2)
MKS$(V)	V'yi 4 baytlık bir karakter zincirine çevirir	A$ = MKS$(71.5)
MKD$(V)	V'yi 8 baytlık bir karakter zincirine çevirir	A$ = MKD$(1521.76)
OCT$(N)	N sayısının 8 tabanlı sayı sisteminde saklanmasını sağlar	A$ = OCT$(12)
PEEK(A)	Bellekteki A hücresinin değerini verir	PEEK(&H4701)
POS(N)	İmlecin ekrandaki konumunu verir	L = POS(1)
RIGHT$(A$,N)	A$ karakter zincirinden sağdan başlayarak N tane karakter çıkarılır	B$ = RIGHT$(A$,7)
RND(K)	0 ile 1 arasında bir rastlantısal sayı verir	R = RND(3)
SGN(A)	A'nın ön işaretine bağımlı olarak —1, 0 veya 1 verir	S = SGN(—3)
SIN(A)	A açısının değerini radyant cinsinden verir	R = SIN(3.14)
SPACE$(N)	N kadar boşluktan oluşan bir zincir üretir	A$ = SPACE$(3)
SPC(I)	Ekrandaki I sayıda boşluğa gönderme yapar	PRINT''A'',SPC(7)
SQR(V)	V'nin karekökünü alır	A = SQR(7)
STR$(N)	N sayısını karakter zincirine çevirir	A$ = STR$(3)
STRING$(N,A$)	A$ zincirinin ilk N karakterinden yeni bir zincir yaratır	B$ = STRING$(5,A$)
TAB(N)	İmleci N sütununa yerleştirir	PRINT ''X''; TAB(6)
TAN(A)	A açısının tanjantını radyant cinsinden verir	Y = TAN(7)
USR(K)	Assembler dilinde yazılmış bir alt-program çağırarak ona K değerini iletir	A = USR1(X)
VAL(A$)	A$ zincirinin sayısal karşılığını verir	N = VAL(A$)
VARPTR(A)	Belirtilen değişkenin adresini verir	VARPTR(X)

Basic'in Lehçeleri

BASIC dilinin kimi birbirine çok benzeyen, kimi de büyük ölçüde farklı, birçok versiyonu vardır.

Genel olarak BASIC 80'deki komut ve fonksiyonlar hemen hemen bütün diğer versiyonlarda bulunmakla birlikte, bazı durumlarda farklı bir sözdizimi ya da sözcük dağarcığı gerektirebilir. En belirgin farklılıkları aşağıdaki işlevlere ayrılan komutlarla ilgilidir:

— Karakter zincirlerinin kullanımı
— Disket üzerindeki kütük ve arşivlerin kullanımı
— G/Ç fonksiyonlarının kullanımı
— Çevresel gereçlerin kullanımı
— Veri kabulü ve iletimi

Karakter zincirlerinin kullanımı

Karakter zincirleri 0 ile her BASIC versiyonunda öngörülen bir üst sınır arasında kalan sayıda karakter içerirler. Bu zincirlere ayrılan bellek alanı, içerilen karakter sayısına bağlı olarak sürekli değişir.

Bir zincirin fiili uzunluğunu bulmak için kullanıcının LEN fonksiyonuna başvurması gerekir. Bazı BASIC biçimlerinde her bir zincire (karakter sayısı cinsinden) azami bir uzunluk vermek mümkündür. Bu iş yapılırken bilinen DIM ya da ALLOCATE komutundan yararlanılabilir. Yalnızca büyük makinelerde bulunan bu sonuncu komut ile bir bellek bölgesi ancak geçici olarak zincirlere tahsis edilebilir.

Örnek olarak, bir alt-programda 1000 adet karakter zincirinin belleğe alınması gerektiğini varsayalım. DIM A$(1000) komutu A$'in 1000 adet karakter zinciri için kalıcı bir bellek alanı ayırır; bu ise, eğer aynı zincirler programın başka bölümlerinde kullanılmıyorsa, önemli bir bellek savurganlığına yol açar.

ALLOCATE A$(1000)

komutu kullanıldığında A$ zincirlerince istenen bellek alanı yalnızca geçici olarak ayrılır. Zincirlerin kullanım evresinin bitiminde, ayrılan bellek alanı serbest bırakılıp başka amaçlar için yeniden kullanılabilir. Daha önce ALLOCATE komutuyla ayrılmış bir bellek bölümünü yeniden serbest bırakan komut ise şudur:

DEALLOCATE A$(1000)

Bellek alanları ayırmak için kullanılan her üç komut da, (DIM, COM, ALLOCATE), köşeli parantez içinde belirtilerek karakter zincirinin azami uzunluğunun belirtilmesine olanak tanır.

Örneğin:

DIM A$[5]	A$ zincirine 5 karakterlik bir uzunluk verir.
COM A$[5]	Bir öncekine benzer, COMMON, yani tüm alt-programlarda ortak bulunan bir bellek alanında işlem yapar (DIM'ın kullanımı ise yalnızca yereldir).
ALLOCATE A$[5]	Öncekilerle aynı etkiyi yapar, fakat bellek alanı ardarda serbest bırakılıp öbür değişkenlere verilebilir.
DIM B$(7) [20]	Her biri yirmi karakterden oluşan 7 zincir için bellek alanı ayırır.

Normalde, bu simgelemeyi kullanan sistemlerin çok önemli iki özelliği vardır:

— Bir zincirin azami uzunluğu 32767 karaktere dek gidebilir.
— Ayrıca alt-zincirler kullanmak da mümkün olabilir.

BASIC 80'de belli bir karakter zincirini bölümlere ayırmada LEFT$, RIGHT$ ve MID$ fonksiyonları kullanılırken, diğer bazı BASIC lehçelerinde aynı işlemi köşeli parantezlerden yararlanarak yapmak mümkündür.

Örneğin A$= "Bu bir denemedir" olsun. "Deneme" sözcüğünü çıkarmak için BASIC 80'de kullanılan komut,

B$ = RIGHT$ (A$,5)

iken diğer bir lehçede şu komut kullanılabilir:

B$ = A$[15]

Komut, 15. sıradakinden itibaren A$ zincirinin tüm karakterlerini çıkarır. A$'den çıkarılan B$ zincirine A$'nin **alt-zinciri** denir. Bu komutun, aktarmanın başlayacağı ve biteceği karakteri seçme olanağı veren bir biçimi de vardır.

Örneğin,

B$ = A$[3,8]

satırı, A$'nin 3. karakterle sekiz sayılı karakter arasında kalan kısmını B$'ye aktarır. Aynı komutun değişik bir biçimi de şudur:

B$ = A$[3;6]

Bu komut ile A$'nin içeriği 3. karakterden başlayarak 6 karakter B$'ye aktarılır. Örnekte kullanılan sayılarla, her iki komutun etkileri öz-

Basic 80	Diğer versiyon	İşlevi
B$ = LEFT$(A$,3)	B$ = A$[1,3]	Soldan başlayarak ilk üç karakter çıkarılır
B$ = RIGHT$(A$,2)	—	Sağdaki son iki karakter çıkarılır
B$ = MID$(A$,4,1)	B$ = A$[4;1]	4. karakter çıkarılır
DIM B$(3)	DIM B$(3)	3 karakter zinciri için bellek alanı ayrılır
Öngörülmemiş	DIM B$(3)[20]	Azami uzunluğu 20 karakteri geçmeyecek 3 zincir için bellek alanı ayrılır
Öngörülmemiş	ALLOCATE B$(3)	Geçici bellek alanı ayrılır
Bu biçiminde öngörülmemiştir; karşılık gelen komut ERASE'dir	DEALLOCATE B$(3)	ALLOCATE ile ayrılmış bellek alanını kapatır

deştir: 3 ile 8. arasında bulunan karakterlerin çıkarılması üçüncü konumda bulunan karakterden itibaren 6 karakterin çıkarılmasıyla aynı anlamı taşır. Yukarıdaki tabloda BASIC 80 ile diğer BASIC lehçeleri arasında zincir kullanımındaki farklılıklar özetlenmektedir.

ALLOCATE ve DEALLOCATE komutlarının genel bir kullanımı vardır ve sayısal değişkenler için de kullanılabilirler. (BASIC 80'de DEALLOCATE'e benzer komut ERASE'dir.) Bu temel farkların ötesinde, bazı biçimsel değişiklikler de sözkonusudur. Örneğin, ASC(A$) fonksiyonu bazı BASIC versiyonlarında VAL(A$), STRING$(n) VAL$(n) biçimini alır. Bununla birlikte yalnızca biçimsel farklılıklar sözkonusudur: Yerine getirilen işlevler değişmez.

Disket Üzerinde Arşivlerin Kullanımı

Çoğunlukla bir makineden diğerine farklılık gösteren giriş ve çıkış komutları, disketlerle ilgili olanlardır.

Değişik makinelerde mantık aşağı yukarı aynı kalmakla birlikte, komutların biçimi önemli ölçüde farklı olabilir.

Örnek olarak, çok yaygın bir işletim sistemi olan DOS'un bir versiyonunda bulunan başlıca disket yönetimi komutlarını aşağıda veriyoruz.

CATALOG	Diskette bulunan kütüklerin listesini ekranda gösterir. CP/M'deki DIR'a benzeyen komuttur.
LOAD, SAVE	İşlevleri CP/M için açıklananların aynısıdır.
INIT	Disketi harekete geçirir. FORMAT'a benzeyen, fakat bazı farklı özellikleri olan bir komuttur. Disket birimi harekete geçer geçmez, her disket takıldığında diskete yüklenmesi gereken bir program kütüğünün disket üzerine aktarılması zorunludur. Programın başlama anında bellekte bulunması gerekir ve bu program otomatik olarak diskete aktarılır. Sözdizimi şöyledir: INIT AD, Sn, Dm burada AD, alt-programın disket üzerinde saklanacağı kütüğün adıdır.

n kullanılmakta olan taşıyıcının (drive) bağlı olduğu yuvanın numarasıdır. m disket biriminin numarasıdır (normal olarak her yuvada ikişer tane bulunur). Ayrıca disket tanımlamak üzere, istendiği takdirde sayısal bir parametre kullanılabilir.
Örneğin,
INIT PROVA, S6, D2, Vp komutu yuva 6(S6)'nın 2. disket biriminde bulunan bir disketi başlatır ve ona p hacim numarasını verir.

DELETE — Bir programı siler; CP/M'deki ERA ile özdeştir.

LOCK — Bir kütüğün silinmesini önlemek suretiyle korunmasına yarar. Kütük UNLOCK komutu ile serbest bırakılabilir.

RENAME — (Tıpkı CP/M'de olduğu gibi) bir kütüğün adını değiştirir.

VERIFY — Bir kütüğün bütünlüğünü denetlemeye yarar. Özel kaza koşullarında bir kütüğün içeriği kısmen bozulabilir; bu komut bütünlüğün sınanmasına yarar. Yöntem, sağlama biti ile aktarmalarda kullanılan yönteme benzer. Her kaydın başlangıcına, kaydın içeriğine bağlı olarak **sınama toplamı** (check-sum) denilen bir değer kaydedilir. Arızi bir nedenle içerikte bir değişme ortaya çıkarsa sınama toplamında buna bağlı olarak bir değişme olmaz ve böylelikle işletim sisteminin açığa çıkaracağı bir aksaklık doğar.

DOS'ta kullanılan kütük tarzları (yani sıralı ve doğrudan erişimli kütükler) CP/M için incelediklerimizle aynıdır. Ancak okuma ve yazma işlevlerinde kullanılan teknik büyük ölçüde farklıdır.

DOS'ta G/Ç işlemleri, READ ve WRITE komutlarıyla yapılır; bu komutlar bir karakter zinciri olarak girilir ve kodları CTRL D tuşunun (yani CTRL ve D tuşlarına aynı anda basılmasının) koduyla [=CHR$(4)] aynıdır.

SIRALI KÜTÜKTE VERİLERİN (DOS) KULLANIMI İÇİN ÖRNEK

```
90 REM:PROG1 PROGRAMI
100 REM:DISKET ICIN GIRIS/CIKIS FONKSIYONLARI
120 REM:GUDUM KARAKTER ZINCIRI
140 A$=CHR$(4)
160 REM:
180 REM:ACILIS
200 PRINT A$;"OPEN DENEME"
220 REM:
260 INPUT "KARAKTER ZINCIRI:";B$
280 REM:YAZMA
300 PRINT A$;"WRITE DENEME"
320 PRINT B$
325 PRINT A$;"CLOSE DENEME"
340 REM:
360 REM:OKUMA
365 PRINT A$;"OPEN DENEME"
380 PRINT A$;"READ DENEME"
400 INPUT D$
420 PRINT A$
440 PRINT "OKUNAN VERILER:"
460 PRINT D$
480 END
```



Petrol kaynaklarının verimliliğinin çizisel gösterimi amacıyla çizicinin kullanılması.

Örneğin, bir okuma komutunun biçimi aşağıdaki gibi olabilir:

```
PRINT CHR$(4); ''READ XYZ''
INPUT V
```

İlk komut, okuma mekanizmasını harekete geçirirken ikincisi bir değer alır ve onu V değişkenine aktarır. Tekrarlayalım, BASIC'in bu türünde de kütükler sıralı ya da doğrudan erişimli olabilir.

Yanda sıralı kütük üzerinde G/Ç tekniğini örnekleyen bir program dökümü verilmektedir. Program, kişisel SIPREL bilgisayarı (model 2040S) üzerinde kurulmuş olmakla birlikte Apple ya da benzer diğer makinelerde de uygulanabilir. Satır 140, A$ değişkenine, diskete yönelik, her komuttan yollanacak olan CHR$(4) (CTRL + D) değerini verir. Kütüğün açılışı (satır 200) A$ karakter zinciri ile OPEN DENEME komutunun (burada DENEME kütüğün adıdır) gönderilmesiyle elde edilir. Verileri yazmak için WRITE komutunun (satır 300) ve ardından aktarılması istenen değişkenlerle birlikte PRINT (satır 320) komutunun gönderilmesi gerekir. Dikkat edilebileceği gibi, disket üzerindeki G/Ç komutlarıyla aynıdır; her iki ekranda da kütüğün belirlenmesi gerekir. 360-460. satırlarda bir okuma örneği verilmektedir. Gönderilecek komutlar şunlardır:

— Kütüğün açılışı (satır 365)
— Okuma komutu (satır 380)
— Disketin açılması (satır 420)
— Veri okuma (satır 400)
— Verilerin ekrana yazılması (440 ve 460 satırları)

Satır 420, A$ = CHR$(4) zincirini yollar ve DOS komutlarını sona erdirerek verilerin sistem birimi (ekran) üzerine bilgiyazımını sağlar. Bu komut, sisteme G/Ç biriminin değiştirilmek istendiğini bildirmek için gereklidir. Örneğin, satır 320 ve 420 karşılaştırılır: Bu iki satır özdeş olmakla birlikte biri diskete aittir (bu, kendinden önceki satır 300'de belirlenmiştir) diğeri ise ekrana aittir.

Sayfa 750'de sıralı bir kütükteki sayısal verilerin yazılması ve sonra okunması ile ilgili bir program dökümü verilmektedir. Yazım komut-

SIRALI KÜTÜKLERDE SAYISAL DEĞERLERLE YAPILAN GİRİŞ/ÇIKIŞ İŞLEMLERİ

```
10 REM:PROG2 PROGRAMI
20 REM:SIRALI KUTUKLER
30 REM:
100 REM:SAYISAL VERILER
110 DIM V(30)
120 INPUT "VERI ADEDI";N
140 PRINT "DEGERLERI GIRINIZ"
160 FOR I=1 TO N
180 PRINT "VERI KAYDI NUMARASI:";I
200 INPUT V(I)
210 NEXT I
220 REM:
240 A$=CHR$(4)
260 PRINT A$;"OPEN DENEME2"
280 PRINT A$;"WRITE DENEME2"
300 FOR I=1 TO N
320 PRINT V(I)
340 NEXT I
360 REM:
380 PRINT A$;"CLOSE DENEME2"
400 PRINT A$;"OPEN DENEME2"
420 PRINT A$;"READ DENEME2"
440 FOR I=1 TO N
460 INPUT V(I)
480 NEXT I
500 PRINT A$;"CLOSE DENEME2"
520 PRINT A$
540 PRINT "**OKUNAN VERILER**"
560 FOR I=1 TO N
580 PRINT "VERI KAYDI NO.:";I,V(I)
600 NEXT I
620 END
```

ları 260, 340 satırlarda bulunur. Kütüğün açılışından (satır 260) ve yazım komutunun (satır 280) gönderilmesinden sonra döngü (300, 340 satırlar), değerleri diskete aktarır. (N klavye aracılığıyla satır 120'de girilmiştir.)

İncelenen makinelerde rastlanan bu işlem biçimi, işletim sisteminin, alanları birbirinden ayırmak için RETURN karakterini [CHR$(13)] kullanılması sayesinde mümkün olmaktadır. Bu yüzden bir sayısal değeri diğerinden ayırmak için iki değerin arasına bir RETURN karakteri sokmak gerekir. Örneğimizde bu iş PRINT komutundan yararlanarak otomatik olarak yapılır. Buna karşılık, CHR$(13) karakteri programın kendisi tarafından yollanabilir. Örneğin, X, Y, Z değerlerini aktarmak için program:

```
A$ = CHR$(4)                          Kontrol kodu
B$ = CHR$(13)                           RETURN
PRINT A$;''OPEN DENEME''
PRINT A$;''WRITE DENEME''
PRINT X; B$; Y; B$; Z
PRINT A$
```

Aşağıdaki program özdeştir:

```
A$ = CHR$(4)
PRINT A$;''OPEN DENEME''
PRINT A$;''WRITE DENEME''
PRINT X
PRINT Y
PRINT Z
PRINT A$
```

Bu durumda sistem, her PRINT komutunun sonuna RETURN kodunu ekler. DOS işletim sisteminde CP/M'de karşılığı olmayan iki güdüm öngörülmüştür.

APPEND
POSITION

APPEND güdümü sıralı bir kütüğe kayıtlar eklenmesine yarar; CP/M sisteminde bir kütü-

SIRALI BİR KÜTÜĞÜN YENİDEN OKUNMASI VE VERİ GİRİŞİ

```
10 REM:PROG4 PROGRAMI
20 REM:SIRALI KUTUKLER
30 REM:APPEND GUDUMU
100 A$=CHR$(4)
105 ON ERR GOTO 1000
120 PRINT A$;"APPEND DENEME"
160 PRINT "VERILERI GIRINIZ"
180 PRINT "BITIRMEK ISTEDIGINIZDE SON GIRINIZ"
190 K=0
200 INPUT B$
220 IF B$="SON" GOTO 320
240 PRINT A$;"WRITE DENEME"
260 PRINT B$
270 PRINT A$
280 PRINT "AKTARILAN VERI KAYDI"
290 K=K+1
300 GOTO 200
320 REM:VERI GIRISININ SONU
340 PRINT A$;"CLOSE DENEME"
360 PRINT A$
365 IF K=0 GOTO 960
380 PRINT "EKLENECEK OLAN:"
400 PRINT K;"VERI KAYDI"
410 PRINT
420 PRINT "VERILERIN YENIDEN OKUNMASI:"
430 PRINT A$;"OPEN DENEME"
680 INPUT "KAC VERI KAYDI OKUNACAK?";NF
700 INPUT "HANGI POZISYONDAN BASLAYARAK?";P%
760 PRINT A$;"OPEN DENEME"
770 PRINT A$;"DENEME POZISYONU,R";P%
780 FOR I=0 TO NF - 1
800 PRINT A$;"READ DENEME"
820 INPUT D$
840 PRINT A$
845 IN%=I + P%
860 PRINT "VERI KAYDI NO.:";IN%
880 PRINT D$
900 NEXT I
920 PRINT A$;"CLOSE DENEME"
940 PRINT A$
960 END
1000 REM: **HATA**
1020 HTAB 10:VTAB 24:TERS GOSTERIM
1030 PRINT "**KUTUGUN SONU**"
1055 NORMAL
1060 PRINT "END"
1080 GOTO 960
```

ğün en sonuna bir kayıt eklemek için önce bütün kütüğün baştan yazılması, sonra da eklenecek kayıtların en sona girilmesi gerekir. POSITION güdümü kütük göstergesinin belli sayıda alan boyunca ileri atılmasını sağlar.

Bu atlamanın yönü yalnızca ileri doğrudur. Gösterge incelenen alandan öncekiler üzerine getirilemez. Her alan, onu izleyen RETURN kodundan tanınır.

Sayfa 751'de sıralı bir kütüğün APPEND güdümü ile genişletilmesi ve tamamen ya da belli bir konumdan itibaren bir bölümünün POSITION güdümü ile yeniden okunması ile ilgili bir örnek program sunulmaktadır.

Uzunluğu önceden bilinmeyen sıralı kütüklerin okunması sırasında, mevcut olmayan bir kaydın okunmaya çalışıldığı durumlar sözko-

ONERR KOMUTUNUN (DOS) KULLANIMI İÇİN ÖRNEK

```
10 REM : PROGRAM PROG5
20 REM :
30 REM : ONERR GOTO KOMUTUNUN KULLANIMI
40 REM :
50 ONERR GOTO 1000
60 REM :
80 A$ = CHR$(4)
100 PRINT A$;"OPEN DENEME"
120 FOR I = 1 TO 100
140 PRINT A$; READ "DENEME"
160 INPUT D$
180 PRINT A$
200 PRINT "VERI KAYDI NO. :";I
220 PRINT D$
240 NEXT I
260 PRINT A$
280 PRINT " ** PROGRAM SONU ** "
300 END
1000 REM : HATALARIN ARANMASI
1020 REM :
1040 REM : ADRES 222
1060 REM :
1080 REM : HATA TANI KODLARI
1100 REM : 5 = VERILERIN ONU
1120 REM : 9 = DISKET DOLU
1140 REM :
1160 T = PEEK (222)
1180 IF T <>5 GOTO 1300
1200 REM : VERILERIN SONU
1220 PRINT A$
1225 HOME
1240 PRINT KUTUGUN SONUNA GELINDIGI SAPTANDI"
1260 K = 1
1280 PRINT A$; "CLOSE DENEME"
1285 INVERSE
1300 PRINT "HATA :";T
1320 PRINT "KURTARILAMAZ"
1325 NORMAL
1340 STOP
```

GET KOMUTUNUN KULLANIMI

```
10 REM :PROGRAM PROG6
20 REM :
30 A$ = CHR$(4)
100 PRINT A$; "OPEN DENEME"
120 PRINT A$; "READ DENEME"
140 PRINT D$
160 PRINT A$
182 PRINT "OKUNAN KARAKTER:";
184 PRINT INVERSE: PRINT " ";D$;" ":PRINT NORMAL
200 PRINT "DEVAM?(E/H)";CEV$
210 PRINT A$
220 PRINT CEV$
225 PRINT A$
240 IF CEV$="E" GOTO 120
260 PRINT " ** PROGRAMIN SONU ** "
280 END
```

nusu olabilir. Bu durumda sistem bir hata mesajı (END OF DATA) iletir ve işlemeyi durdurur. Bir hata halinde denetimi belli bir (nn) satırına geçiren (CP/M'de aynı işlevi gören komuta benzer) ONERR GOTO nn komutu kullanılarak programın tamamen sona ermesi önlenebilir.

Programcı bellekte ortaya çıkan hatanın kodunu gözden geçirerek hatanın nedenini saptayabilir ve gerekli önlemleri alır.

Sayfa 752'de ONERR GOTO... komutunun kullanımıyla ilgili bir program örneği verilmektedir.

Ortaya çıkan hata ile ilgili kod PEEK (222) komutu ile okunur. Bu bellek alanı yalnızca kullanılan makinede (SIPREL 2040S) geçerlidir. Başka makinelerde farklı olabilir. Bu bellek alanında (satır 1160 ve 1180'de) bulunan sayısal değer kontrol edilerek de hata türü bulunabilir.

DOS'un kullanıldığı Apple ve ona uygun sistemlerde tek bir baytı bir defada okumaya yarayan GET komutunu kullanan CP/M'ye göre (bu sistemde tüm kaydı içine alır) önemli bir farklılık vardır.

Bu sayfada bir karakteri okuyan, onu ekranda görüntüleyen ve programın sürmesi için iznin girişini (S karakterini) bekleyen bir program dökümü verilmektedir.

GET komutunun iki olası kullanımı vardır: Satır 140'da disket üzerindeki kütükten bir karakter alınırken, satır 220'deki karakter klavye aracılığıyla okunur.

DOS işletim sisteminde doğrudan erişimli kütükler sıralı kütüklere benzer biçimde işlem görür. Aradaki tek fark, doğrudan erişimli kütüklerde komutla belirtilen belli bir kayıt numarasına doğrudan ulaşılabilmesidir.

Örneğin;

```
A$ = CHR$(4)
PRINT A$;''OPEN AD, L''
.....
PRINT A$;''WRITE AD, R''
.....
PRINT A$;''READ AD R''
```

satırları şu işlevleri görür:

— (bayt cinsinden) L uzunluktaki kayıtları içeren AD kütüğünün açılışı
— AD kütüğüne R kaydının yazımı
— AD kütüğünden R kaydının okunması

Doğrudan erişimli kütüklerin kullanımı ile ilgili bir program dökümü s.754'te verilmektedir.

DOS sisteminin bir başka özelliği bir alan içinde bulunan bir tek baytın adreslenebilmesidir. Bunu yapmak için bayt numarasını READ komutunda vermek yeterlidir.

Örneğin aşağıdaki komut:

PRINT A$;''READ AD,7,2''

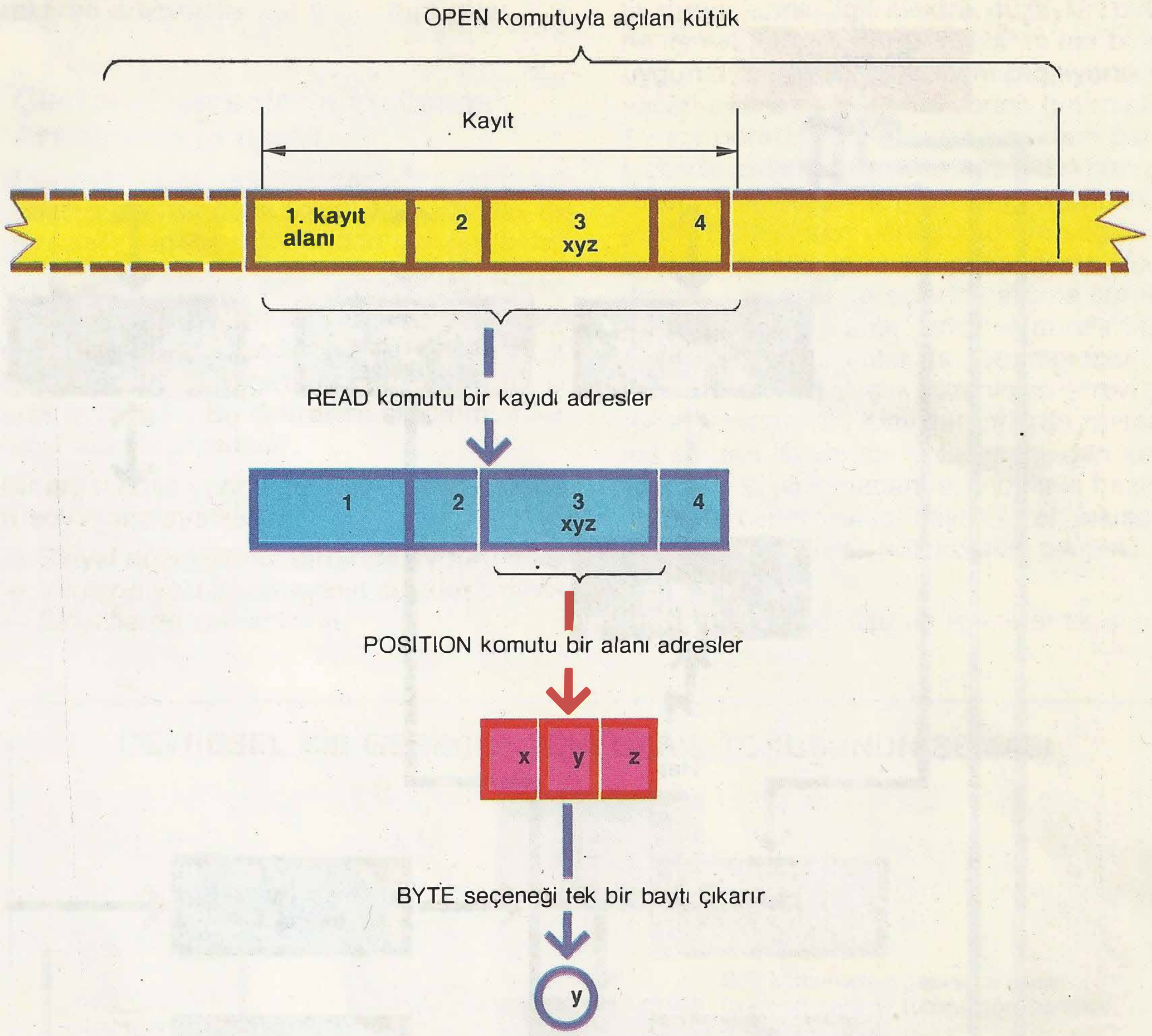
AD kütüğündeki kayıt 7'den, 2 sayılı baytı okur.

755. sayfada kayıtlarla ilgili çeşitli komutların kullanılmasıyla elde edilen mantık şeması ve ayrıntı düzeyleri görülmektedir.

DOĞRUDAN ERİŞİM İÇİN PROGRAM ÖRNEĞİ

```
10 REM : PROGRAM PROG3
20 REM : DOGRUDAN ERISIMLI KUTUK
30 REM :
40 REM : PARAMETRE
50 REM F$=BAYT CINSINDEN UZUNLUK
60 REM : NR=KAYIT NUMARASI
100 A$=CHR$(4)
120 REM :KAYIT UZUNLUGU = 40
125 REM : ARTI 1 KUTUK SONU ISARETI
130 F$="41"
140 PRINT A$;"OPEN DENEME3.""+" L"+F$
160 REM : VERI GIRISI
180 REM :
200 INPUT "VERILERIN SAYISI?";N
210  IF N = 0 GOTO 710
220 FOR I = 1 TO N
225 B$ = "":C$=""
240 PRINT "VERI NUMARASI N. :";I
260 INPUT "DEGER (KARAKTER ZINCIRI)";B$
270 REM : UZUNLUK KONTROLU
280 K = LEN (B$)
290 IF K < 40 GOTO 400
295 IF K = 40 THEN  C$ = B$
300 IF K = 40  GOTO 500
320 REM : UZUNLUGU VERI KAYDININKINDEN
340 REM : FAZLA OLAN KARAKTER ZINCIRI
360 C$ = LEFT$(B$,40)
380 GOTO 500
400 REM : UZUNLUGU VERI KAYDININKINDEN
420 REM   AZ OLAN KARAKTER ZINCIRI
440 M = 40 - K
442 C$ = B$
445 FOR J=1 TO M
460 C$ = C$ + ","
470 NEXT J
500 REM : UZUNLUGU VERI KAYDININKINE
520 REM   ESIT OLAN KARAKTE ZINCIRI
540 REM : DONUSTURME YAPILMIYOR
600 REM.:
620 INPUT "VERI KAYDI NUMARASI?";NP
640 PRINT A$;"WRITE DENEME3.R";NR
660 PRINT C$
685 PRINT A$
700 NEXT I
710 PRINT A$;"CLOSE DENEME3"
740 INPUT "OKUNACAK MI ";CEV$
760 IF CEV$ <>"E" GOTO 880
780 INPUT "HANGI KAYIT";NR
800 PRINT A$;"OPEN DENEME3."+"L"+F$
820 PRINT A$;"READ DENEME3,R";NR
840 INPUT D$
860 PRINT A$;"CLOSE DENEME3"
870 PRINT A$
880 PRINT "OKUNAN VERILER:"
900 PRINT D$
920 INPUT "DEVAM (E/H)";CEV$
940 IF CEV$="E" GOTO 780
960 END
```

BİR KÜTÜĞE ERİŞİM DÜZEYLERİ



G/Ç Fonksiyonlarının Kullanımı

Çeşitli BASIC lehçelerinde G/Ç işlevlerini düzenleyen komutlar konusunda önemli farklılıklar görülür. Bu komutlar, makine ile dış dünya arasındaki iletişimin düzenlenmesi amacıyla kullanılan çevresel gerecin türüne bağlı olarak son derece karmaşık görünümler alabilir. Başka bir deyişle donanım, bu işlevlerin düzenlenmesinde önemli bir rol oynar. Bilgisayarlarla ekran arasındaki iletişim normalde RS 232 türünde bir arabirimle gerçekleşir.

Bunlar farklı makine türlerinde hemen hemen birbirlerinin aynıdır.

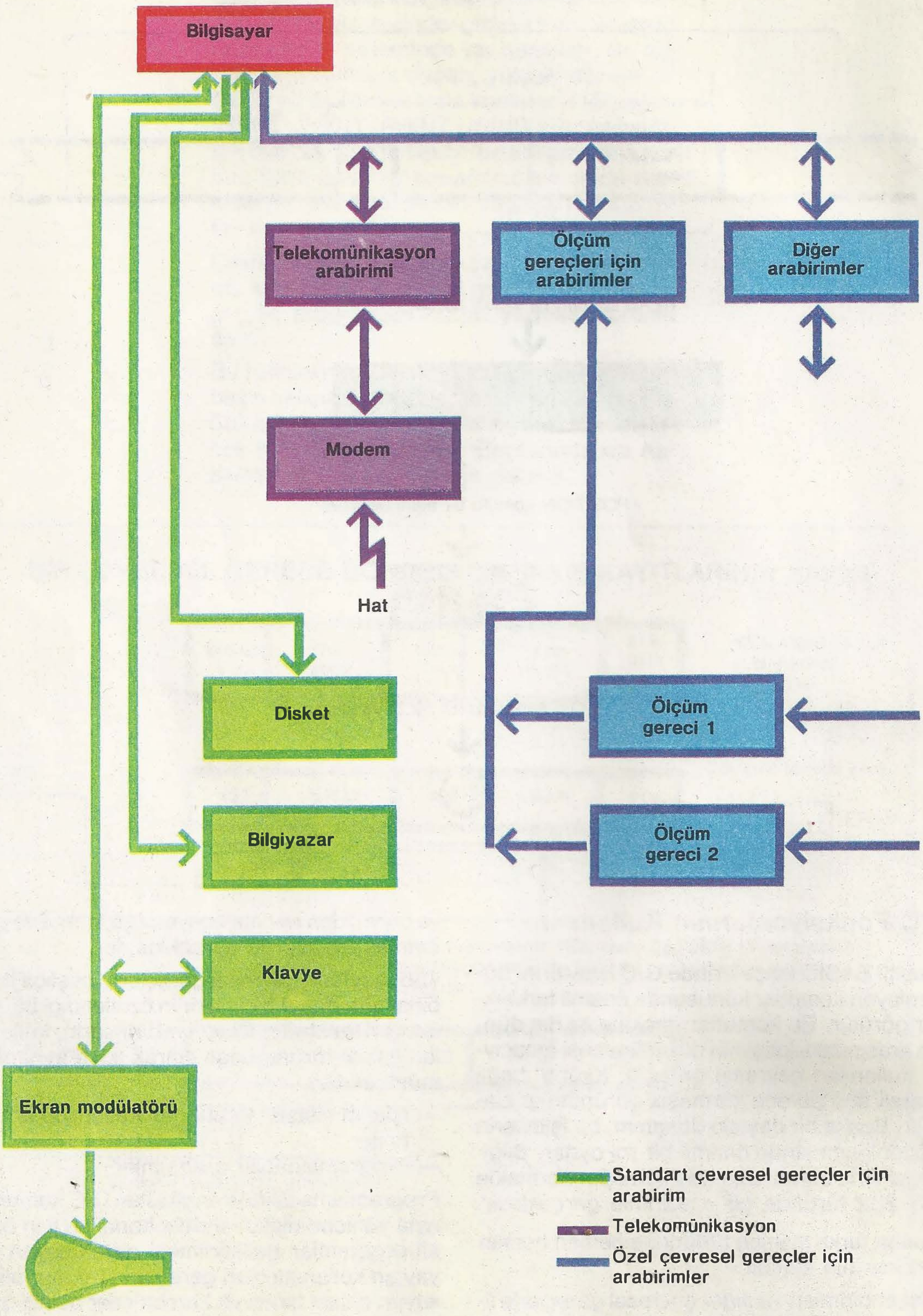
Disket birimleri ve diğer çevresel gereçlerle ilgili düzenlemeler yazılıma sıkı sıkıya bağlıdır ve bu yüzden her makineye özgü komut ve güdümler aracılığıyla gerçekleşirler.

756. sayfada çevresel gereçlerle başlıca arabirim (interface) biçimlerinin özetlendiği bir şema verilmektedir. Olası arabirimlerin, kullanılan yazım türüne bağlı olarak ikiye ayrılması mümkündür:

— İşletim sistemi tarafından kullanılan arabirimler
— Programlanabilir arabirimler

Programlama dilinde öngörülen G/Ç komutlarıyla yönlendirildiklerinden, kendileri için özel alt-programlar geliştirilmesi gerekmeyen ve yaygın kullanımı olan gereçlerle (ekran, bilgisayar, disket birimi vb.) arabirimler birinci gruba girer. Çalıştırılmaları için kullanıcı tarafın-

BİLGİSAYAR ARABİRİMLERİNİN (INTERFACE) ŞEMASI



dan gelişmiş makinelerde simgesel bir dilde (BASIC) diğerlerinde ise Assembler dilinde belli kullanım programları hazırlanmasını gerektiren arabirimler ise ikinci türe girer.

Çevresel gereçlerin kullanımı: Arabirimlerin işlevi

Bilgisayarı oluşturan elektronik devrelerin sinyalleri düşük bir güce sahip olduklarından dış gereçleri yönetme gücünden yoksundurlar.

Bu yüzden, bilgisayarla çevresindeki gereçler arasına, sinyalleri bir yandan bilgisayarın elektroniğine, öte yandan çevresel gerecin yapısına uyumlu hale getiren özel devreler koymak gereklidir. Bu devrelere arabirim (interface) adı verilmektedir.

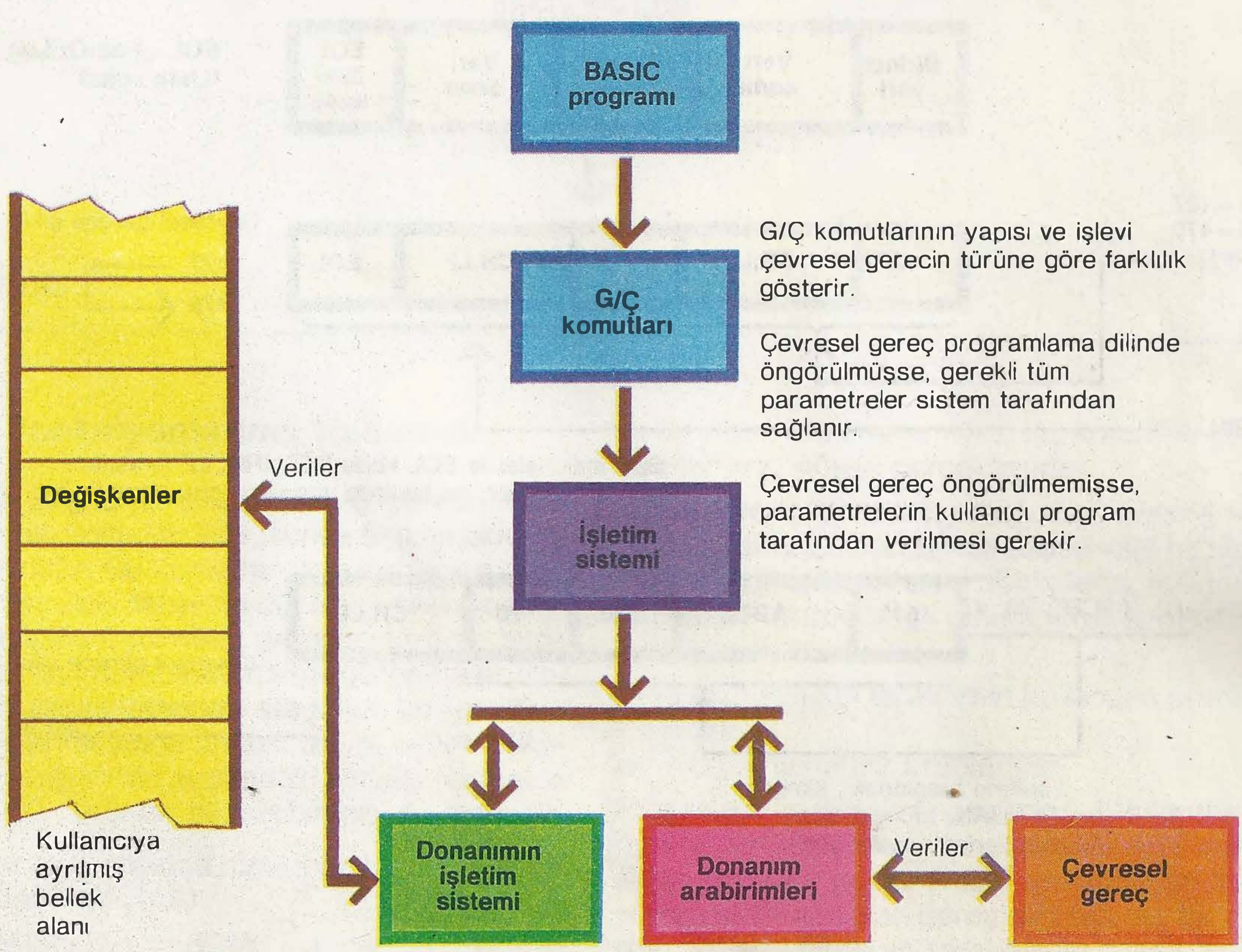
Bir ara birimin yerine getirdiği başlıca işlevleri şöyle sıralayabiliriz:

— Sinyal düzeylerinin birbirine uydurulması
— Verilerin yazı düzenlerinin denkleştirilmesi
— Sinyallerde zamanlama

Birçok uygulamada çevresel gerecin gerektirdiği sinyalin düzeyi ile bilgisayarın verdiğinin aynı düzeyde olmadığı görülür. (1 ve 0 mantık durumlarının ilgili elektrik düzeyleri birbirine uymaz.) Bu durumda arabirim her birimin uygun düzeylerde kullanımını sağlayarak sinyalleri uyarlama görevini yerine getirmelidir. Ayrıca normalde bilgisayar sinyalleri paralel biçimde verilirken (taşıtlar paraleldir) bazı çevresel gereçler için seri girişe gereksinim duyulur. Paralel-seri dönüşümünün sağlanması da ara birimin görevleri arasındadır. Bunun dışında çevresel gereçlerin çalışma sürelerinin bilgisayarın tipik çalışma sürelerinden farklı olduğu durumlar da görülmektedir. Bu durumda da veri akışını düzenleme görevi çevresel gerece aittir. Kimi durumlarda zamanlama sorunu ilk verinin yollanmasından sonra ikinci veriyi yollamadan, alıcı birimin hazır olduğunu bildirmesine dayalı, "**el sıkışma**" (handshake) olarak adlandırılan teknikle aşılabilir.

Bir veri değiş tokuşu ile ilgili mantık şeması

ÇEVRESEL BİR GEREÇLE VERİ DEĞİŞ TOKUŞUNUN ŞEMASI



G/Ç komutlarının yapısı ve işlevi çevresel gerecin türüne göre farklılık gösterir.

Çevresel gereç programlama dilinde öngörülmüşse, gerekli tüm parametreler sistem tarafından sağlanır.

Çevresel gereç öngörülmemişse, parametrelerin kullanıcı program tarafından verilmesi gerekir.

aşağıda gösterilmektedir. Kullanılan çevresel gerecin türüne bağlı olarak uygulama programı iki farklı biçim alabilir.

Çevresel gerecin simgesel dilde öngörülenlerden biri olması halinde, gerekli tüm parametreler işletim sisteminde var demektir. Bu durumda kullanıcıya düşen, yüksek düzeyli dilin bir ya da birden fazla komutuna (örneğin, INPUT, PRINT, PRINT USING vb.) başvurarak ilgili rutini çağırmaktır. Bu komutlardan her biri, kodu çözüldükten sonra, sistemde çevresel gerecinin kullanımını sağlayan bir dizi rutini devreye sokar.

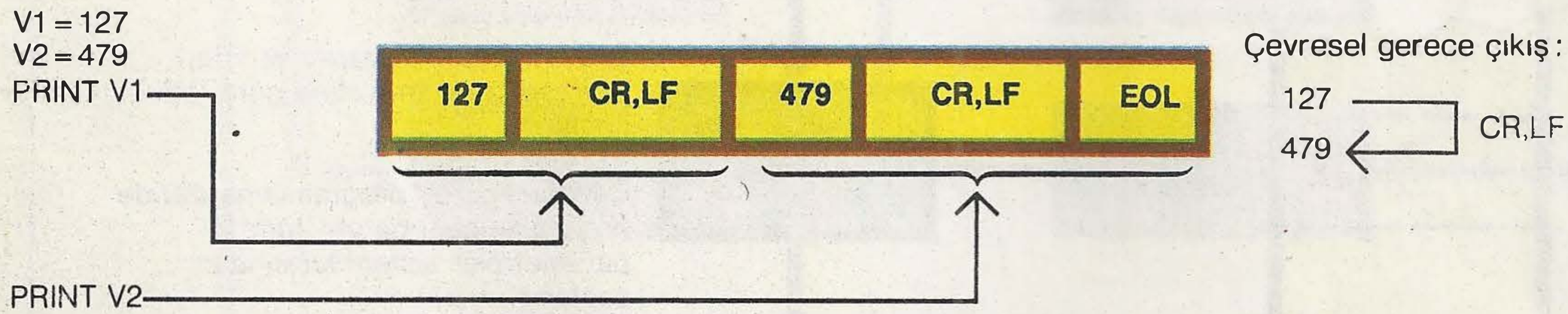
Çevresel gereç simgesel dilde öngörülmemişse, kullanıcının çevresel gereç kullanımı için gerekli programları bizzat yazması zorunlu olur.

Bu rutinlerin yapısı ile kullanılan donanım birbirine yakından bağlıdır. Bazı makinelerde BASIC dilindeki özel komutlarla yönlendirilebilecek arabirimler bulunur. Bazılarında ise Assembler diline başvurmak gerekir.

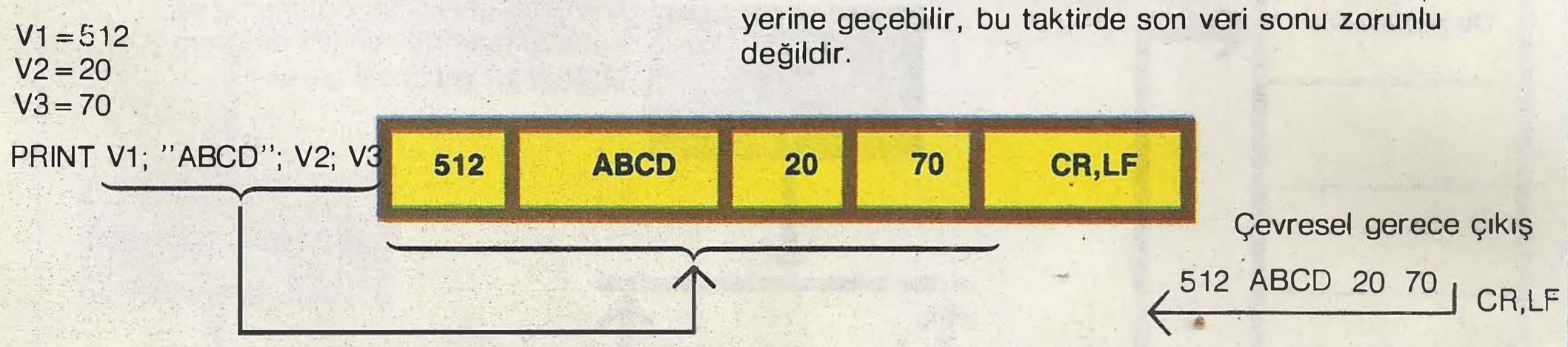
BİR ÇEVRESEL GERECE GÖNDERİLEN ÇIKIŞ KAYITLARININ YAPISI



EOL = End Of List (Liste sonu)

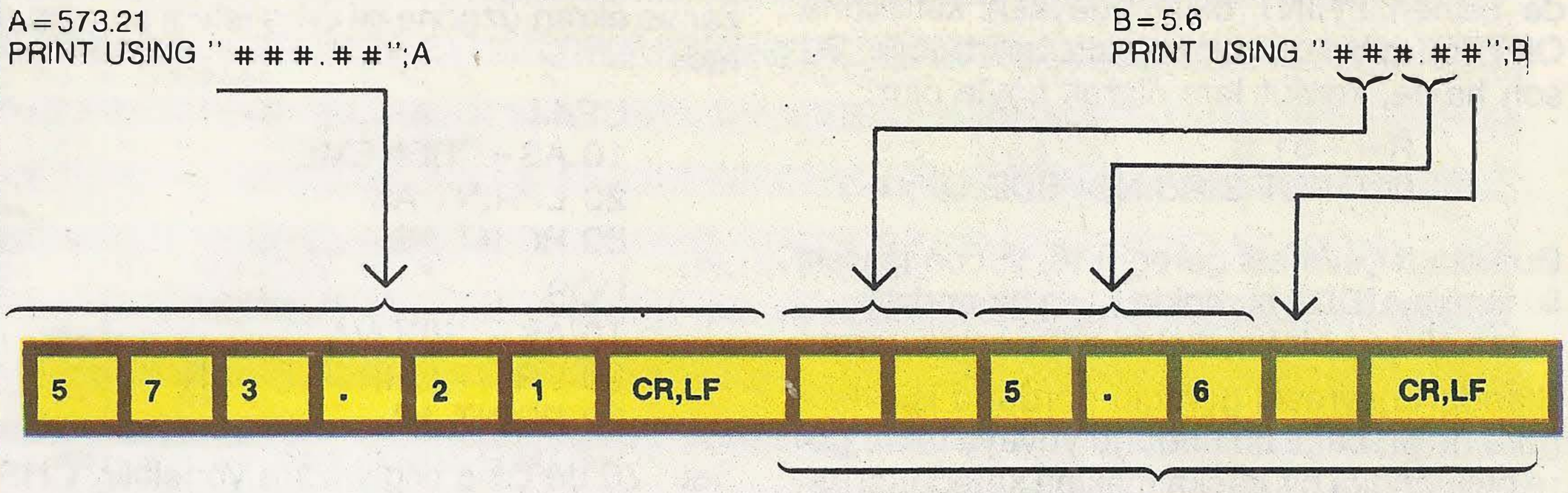


Bazı makinelerde EOL kodu RETURN, LF sırasının yerine geçebilir, bu taktirde son veri sonu zorunlu değildir.



Verilerin arasındaki; simgeleri (RETURN, LF) gibi son işaretlerinin yerine geçer.

ÇEVRESEL GERECE GÖNDERİLEN DÜZENLENMİŞ (FORMATLI) ÇIKIŞ KAYITLARININ YAPISI



Sayfa 758'de çevresel gereçler (örneğin ekran ve bilgiyazar) üzerinde iki çıktı örneği verilmektedir. Süreç bir tek PRINT komutuyla başlatılır, sistemin kendisi de kaydı gerekli kodlarla (örneğin -RETURN ya da LF- satır başı) tamamlar ya da belli bir yazı düzeni sözkonusu ise gerekli yerleri boş bırakır.

Bu sayfada BASIC 80'in simgelerini (#) kullanan iki sistemindeki PRINT USING komutu örneği verilmektedir. Diğer BASIC türlerinde daha karmaşık simgeler sözkonusudur. Başlıca yazı düzeni seçenekleri şunlardır:

A — ASCII formatını belirler: Örneğin USING ''4A'' 4 ASCII karakterinin çıkacağı anlamına gelir.

D — Basamak (= digit); her D harfi bir ondalık sayı (0'dan 9'a) demektir. İşaret sayıların önüne getirilebilir. Örneğin;

USING ''SDD.D''

ifadesi başlarında (S) ön işareti bulunan iki tam ve bir ondalık sayıdan oluşan bir sayısal değerdir.

E — Üs; sayının bilgiyazımı üslü olarak yapılır.

K — Tıkız (compact) bir yazı düzenini gösterir; tüm boş alanlar atlanır.

X — Biraz daha boş alan bırakılması için kullanılır. Örneğin;

USING ''SD.DD,5X,SDD.D''

komutu ilk sayıyı (SD.DD) yazar, beş hane (5X) atlar ve ikinci sayısal değeri (SDD.D) yazar.

Verdiğimiz örneklerde, asıl yapılması gereken işlemi (yani yazımı) ifade eden komutu özellikle atladık. Çünkü bu komut kimi makinelerde bilinen PRINT biçimindeyken kimilerine OUTPUT gibi farklı bir komutla belirtilebilir. Bu son halde, komut tam olarak şöyle olur:

```
R=—51.6
OUTPUT 3 USING''SDD.D'';R
```

Bu komut çevresel gereç 3'te, (S) ön işaretli iki tamsayı (DD), bir nokta (.) ve bir ondalık sayıdan oluşan bir kayıt elde edilmesine yarar.

Standart çevresel gerecin gördüğü işlevlere göre değil, bağlı bulunduğu yuvaya (slot) göre belirlendiği bir başka makine kategorisi daha vardır.

Yuva, bilgisayarı yan gereçlere yani yalnız çevresel gerece değil, aynı zamanda bellek uzantılarına da bağlamaya yarayan bağlantılara denir.

Yuva denildiğinde genelde bilgisayar taşıtına bağlı bir elektronik devre akla gelir. Sistem bu durumlarda G/Ç çevresel gereci olarak ekran ve klavyeyi kullanır. Bir çıkışı, örneğin bilgiyazar üzerine geçirmek için ilgili yuvanın uygun bir komutla devreye sokulması yeterlidir. Çıkış komutu yine PRINT'tir, fakat onun başına bilgiyazarın arabirimini harekete geçiren bir kod getirilmiştir. Kişisel bilgisayarlarda sıkça kullanılan bu çevresel gereç kullanım sisteminin bazı üstünlükleri vardır. Bunlar arasında özellikle kullanıcının sistemi kendi isteklerine uygun olarak biçimlemesi imkânı sayılabilir. Yeni bir çevresel gereç eklemek gerektiğinde ara birim kartını yuvaya sokmak yeterlidir. Daha sonra uygulama programı evresinde, işlevler bu yuvaya doğru yönlendirilecektir.

Bir başka önemli üstünlük, sisteme ikinci bir minik-işlemci katma olanağının ortaya çıkmasıdır.

İşletim sistemi takılan minik-işlemciye sıkı sıkıya bağlıdır. Örneğin çeşitli CP/M türlerinin Z80 minik işlemci (micro processor) ailesiyle (8080, 8086 vb.) birlikte kullanılması mümkündür. DOS gibi diğer bazı sistemlerde ise başka minik işlemciler kullanılır.

Anlattığımız bu yapı kullanıldığında makineye ikinci bir MİB dolayısıyla ilk işletim sistemiyle dönüşümlü olarak işleyecek ikinci bir sistemin eklenmesi mümkün olur.

Bu durumlarda ilk MİB yalnızca ikincil işlevlere ayrılırken bilgisayarın kendisi farklı bir makine ailesine aitmiş gibi kullanılır. Bu ikilik her problem için o probleme en uygun işletim sisteminin seçilmesini mümkün kılar; aynı şey kullanılacak yazılım cinsi için de geçerlidir.

Aşağıda CP/M ve DOS sistemlerinde bilgiyazar ve ekran üzerine iki çıkış isteği sunulmaktadır:

```
CP/M
10 A$=''DENEME''
20 LPRINT A$
30 PRINT A$

DOS
10 A$=''PROVA''
20 PRINT CHR$(4); ''PR # 1''
30 PRINT A$
```

Satır 20'de çıkış bilgiyazara yöneltilir. CHR$ (4) kodu komutu devreye sokar, PR # 1 karakterleri ise komutun kendisini oluşturur.

Verilen kodlar 6502 minik işlemcisini kullanan makinelere (Apple, SIPREL) uygun olanlardır. Bu makineler ailesi CP/M'i kullanan bilgisayarların en geniş alternatifidir.

Ekran kullanımında da bazı ayırdedici özellikler sözkonusudur. CP/M sisteminde imleci BASIC diliyle yönlendirmeye yarayan özel komutlar yoktur. İmleci ekranın herhangi bir bölgesine yerleştirebilmek için sıra ve değer yönünden kullanılan ekran türüne bağlı özel bazı kodlar yollamak gerekir.

Oysa 6502 minik işlemci kullanan diğer bilgisayar ailesinde BASIC dilinde iki imleç yönlendirme komutu vardır:

```
Yatay yerleştirme için HTAB n
Dikey yerleştirme için VTAB m
```

Örneğin ''yerleştirme denemesi'' yazısını yollamaya yarayan bir programın biçimi şöyle olabilir:

```
10 INPUT ''Sütun'';C
20 INPUT ''Satır'';R
30 A$=''Yerleştirme denemesi''
100 HTAB C:VTAB R
110 PRINT A$
120 END
```

Satır 100'ün dışında kalan komutlar BASIC 80'de kullanılanların aynıdır.

Veri Kabulü ve İletimi

Haberlerin anında alınması ile ilgili gitgide artan gereksinim, yaygın işletim sistemlerinin, yani kendi aralarında ya da bir büyük bilgisayar aracılığıyla birbirine bağlı ve birden fazla küçük bilgisayardan oluşan şebekelerin (ağların) önemini artırmıştır. Normal bir terminal yerine bir küçük minik-işlemcinin kullanılma-



Bilgisayar kullanılan bir borsa komisyoncusu bürosu.

sının başlıca nedeni, ikincisinin hem terminal hem de bağımsız bir veri işlemcinin işlevlerini aynı anda ve birlikte yerine getirmesidir.

En basit veri aktarma protokolü EIA RS-232 —C ara birimiyle sağlanır. Bu nedenledir ki, pek çok bilgisayar yapımcısı makinelerinde yüksek düzeyli yani BASIC dilinde komutlar aracılığıyla bu protokolü yönetebilecek ara birimler öngörmüşlerdir.

Ana komut protokolün parametrelerini ortaya koyar. Bu parametreler şunlardır:

— **Aktarma hızı:** Her saniyede iletilen bit sayısını ifade eder (bps = saniye başına düşen bit sayısı). Kullanılan değerler 75, 110, 150, 300, 600, 1200, 1800, 2400, 4800, 9600, 19200 bps'dir.

— **Tamlama:** Tamlamanın nasıl denetlendiğini, yani tamlama bitinin ele alınma biçimini gösterir. Beş olasılık vardır:

— İletilmedi
— Daima 0
— Daima 1
— Teke tamlama olarak iletildi
— Çifte tamlama olarak iletildi

— **Veri bitleri:** Veri bitleri 4 ya da 8 adet olabilir. Bu yüzden ara birim gerekli (format) düzene göre bölümlenmiş olmalıdır.

— **Durdurma bitleri:** Her verinin sonuna durdurma bitleri konulabilir ya da konmayabilir. Üç olasılık sözkonusudur:

— Durdurma biti
— 1 durdurma biti
— 2 durdurma biti

— **Kütüğün sayısı:** Bir iletişim kanalından yararlanabilmek için, GET ve PUT komutlarının uygulanacağı kütüğe bir sayı vermek gerekir.

Örneğin şu komutu ele alalım:

```
OPEN ''COM 1:2400,S,8,1'' AS # 3
```

Bu komut, ara birim 1 (COM 1) için, aktarma hızı 2400 bps, tamlama biti olmayan (S işareti), 8 bit artı bir durdurma bitlik düzende kayıtlar içeren 3 numaralı kütüğün açılmasını sağlar.

Komutun genelleştirilmiş biçimi ile satır denetim kodları da belirtilebilir. Kullanımı kolay olan RS-232 protokolü iki tek birim arasında bağlantı kurmak gerektiğinde eşsiz bir üstünlüğe sahiptir. Aynı bilgisayardan değişik çevresel gereçlerle iletişim kurabilmek için daha fazla giriş kapısına ya da dışsal bir seçim sistemine gereksinim duyulur. Ancak böyle bir seçim işlevi görmek için eklenecek yeni bir donanım durumu fazlasıyla karmaşıklaştıracaktır. Bu ise RS-232 protokolünün yaygınlaşması önünde ciddi bir engel teşkil eder.

Normalde, bir bilgisayarla çevresel gereçler arasındaki etkileşim ölçü araçlarının otomatik denetimi ile gerçekleşir. Bu amaçla özel bazı protokoller geliştirilmiştir. Bunlar arasında en bilinen IEC (International Electrotechnical Commission - Uluslararası Elektroteknik Komisyonu)'de standart olarak önerilen HP-IB (Hewlett Packard Interface Bus - Hewlett Packard Ara Birim Taşıtı)'dir.

HP-IB bağlantısı (ya da taşıtı) 8'i veri aktarılmasına, diğer 8'i de güdüm işaretlerine ayrılmış 16 kablodan oluşur. Veriler 7 bitlik (sekizincisi tamlama bitidir) ASCII koduna aktarılır. Gereçler ya da genel bir ifadeyle bağlanabilecek birimler üç bölümde toplanabilir:

ALICI	Başka birimlerden veri alabilen, örnek olarak bilgiyazar ya da ekran vb. birimler.
VERİCİ	Bir fizik büyüklüğün ölçümünü yapıp elektrik (sayısal) işarete dönüştürerek onu taşıt üzerine yollayan bir araç ya da teyp benzeri veriler iletebilen birimler.
DENETLEYİCİ	Diğer birimler arasındaki iletişimden sorumlu denetim birimi. Bilgisayar genellikle bu işlevi görür.

Birimlerden herhangi biri zaman zaman bu üç rolden birini oynayabilir: Örneğin bir ölçü aracı bilgisayardan komut aldığında ALICI durumundayken ölçü araçlarının sonuçlarını aktardığında VERİCİ işlevi görür. Bununla birlikte, HP-IB sisteminde aynı anda bir tek VERİCİ ile bir tek DENETLEYİCİ'nin çalışır durumda olması öngörülmüştür. Bu durumda devrede bulunan diğer gereçler ALICI rolü oynar. Kimi BASIC lehçelerinde ise uzaktan iletilen veriler için **kapan** (trap) komutları öngörülmüştür. Bu komutlar programın işleyişini durdurarak denetimi bizzat veri girişini yapan bir hizmet alt-programına geçirirler.

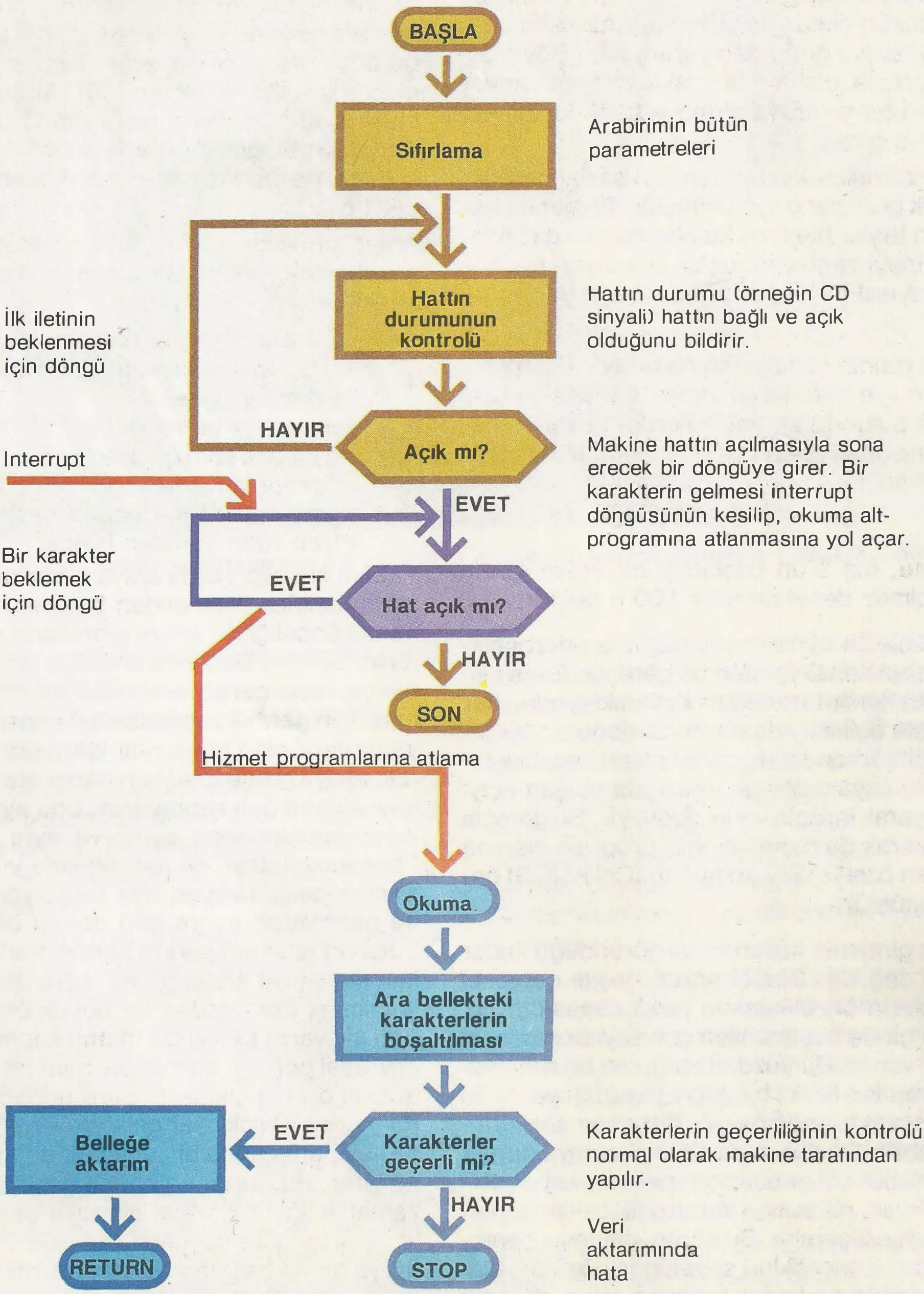
Bu komutlar aracılığıyla makinenin, bir karakterle bir sonraki karakterin gelişi arasında geçen süreyi beklerken bir başka iş yapması mümkün olur.

Karşı sayfada uzaktan veri iletilirken başvurulan mantığın akış şeması görülmektedir.

Bir hesap merkezinde bilgiyazım odası. Kısaltmalar aracılığıyla çıktı sahibinin kim olduğu belirlenir.



TELEKOMÜNİKASYON PROGRAMI ÖRNEĞİ



Bir verinin alınması ile ilgili ana fonksiyona **araya girme** (interrupt) adı verilmektedir. Araya girmede amaç bir dış müdahale durumunda özel alt-programları devreye sokmaktır. Hemen akla geliveren bir örnek herhangi bir programın işleyişini durduran BREAK komutunun (CP/M'de CTRL*C kodları ile elde edilir) yerine getirdiği işlevdir. Uygulama programının işleyişi sürerken klavyeye başvurulmadığından, makinenin yeni komutları alabilmesi mümkün olmaz. İşleyişin durdurulabilmesi için bir araya girmeden yararlanılır. Böyle bir olay ortaya çıktığında makinenin çalışması durdurulur ve araya girmeye bağlı fonksiyon devreye girer.

Kimi makinelerde araya girmeyi başlatmak için değişik gereçler öngörülmüştür. Bunların başlıcaları tuşlar (key) ve kulplar (knob)'dır. Araya girmeyi sağlayan tuşlar fonksiyon tuşlarıdır. Sayısal bir değer göndermeye yaramazlar.

Hangi rutinin çağırılacağına ortaya çıkan araya girmeye göre karar verilir. Fonksiyon tuşlarının bulunduğu makinelerde onların kullanımına yönelik bazı BASIC komutları öngörülür. Örneğin;

ON KEY 3 GOTO 100

komutu, tuş 3'ün başlattığı bir araya girme olur olmaz denetimi satır 100'e geçirir.

Kulp imlecin oynamasını sağlayan darbecikler **(impulsions)** yaratan bir gereçtir. Esas olarak, kullanıcı tarafından kullanıldığında, her dönüşte belli sayıda darbecik doğuran bir jeneratör sözkonusudur. Darbelerin hesabını tutan bilgisayar, dönme sırasında oluşan açıyı saptayarak imlecin yerini değiştirir. Bu gereçle ilgili olarak da denetimi ilgili program alanına aktaran özel BASIC komutları (ON KNOB) öngörülmüştür.

Araya girmenin kullanılması göründüğü kadar kolay değildir. Birinci sorun çeşitli çevresel gereçlerin önceliklerinin farklı olmasıdır. Bir araya girme başlatabilen çok sayıda çevresel gereç vardır. Bu yüzden, çağrılan bir rutin sona ermeden ikinci bir çağrı yapıldığı ya da iki çağrının aynı anda geldiği durumlar sözkonusu olabilir. Bu durumda sistemin araya girmelerin kabul edilip edilmeyeceğine, kabul edileceklerse, ne zaman kabul edileceklerine karar vermesi gerekir. Bu sorun, çevresel gereçlerin farklı öncelikleri sayesinde çözülür. Öncelik değeri ne kadar fazlaysa araya girme o kadar erken devreye sokulur, öncelik sırasının sonlarında yer alan bazı istekler ise ihmal edilebilir.

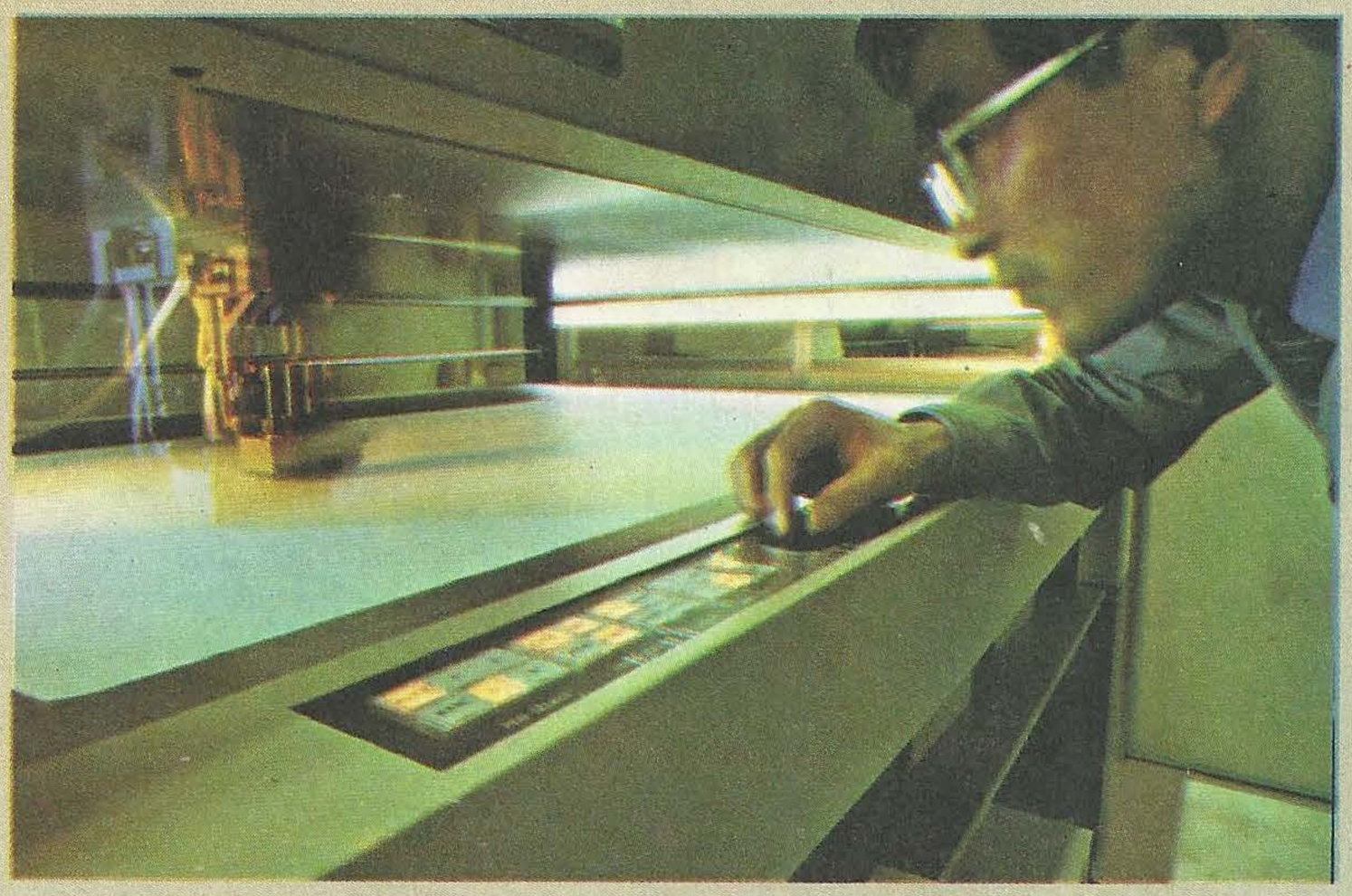
Bir olayın öncelik durumu yazılım ya da donanım tarafından belirlenebileceği gibi, bir hata ya da benzeri olağan dışı durumların ortaya çıkmasına bağlı yazılım öncelik düzeyleri özel komutlarla da belirlenebilir. Burada şu mantık geçerlidir:
Araya girme yoksa, yani programın işleyişi normal olarak sürüyorsa, öncelik düzeyi 0'dır (sistem tarafından otomatik olarak belirlenen bu düzeye sistem önceliği ''system priority'' adı verilir). Bu biçimde düşük düzeyde de olsa (0'ın altına inilemeyeceğinden) herhangi bir araya girme, ânında (daha doğru deyişle yürütülmekte olan komutun bitiminden itibaren) etkili olur.

Araya girme sırasında işletilen rutinin çalışması sırasında yeni bir araya girme ortaya çıkarsa iki şık sözkonusu olur:

1 / Yeni araya girme daha az önceliklidir. Beklemeye alınır ve bir öncekinin bitiminde devreye girer.
2 / Yeni araya girme geçerli olandan fazla (ya da ona eşit) öncelik taşır. Önceki araya girme bekletilir ve yürütme yeni fonksiyona devredilir. Bunun sonucunda bekletilen rutin yeniden işletilir.

Yazılım önceliği çeşitli araya girmelerin etki sırasını belirler. Yazılımdan bağımsız olan donanım önceliği ise araya girmelerin alınma sırasını belirler. Donanım önceliği, araya girmenin çevresel gereç tarafından devreye sokulması için gerekli zamanlamayı yapar. Bir araya girmeyi alan bilgisayar işleyişini durdurarak, araya girme isteğinin hangi ara birimden kaynaklandığını saptayarak, onu aynı birimin diğer isteklerinden ayırır ve aynı anda da ''bekleme listesi''ne alır. Makine yüksek yazılım önceliği taşıyan tüm fonksiyonları yerine getirdikten sonra geri dönüp beklemede bulunan istekleri yerine getirir. Veri aktarılırken kullanılan araya girme, süre denetiminin yapılması bakımından da büyük önem taşır. Veri alışverişi sırasında (hattın kopması ya da çevresel gerecin bozulması gibi) bir donanım arızası ortaya çıkabilir. Süre denetimi öngörülmemişse bilgisayar gelecek veriyi beklemeye koyularak sonu olmayan bir döngünün içine girer. Bu durumun önüne geçebilmek için zamanla ilgili bir araya girmeye başvurulabilir: Belli bir süre boyunca veri aktarılmazsa bir araya girme başlatılır ve bir hata mesajı üretilerek program durdurulur ya da başka bir işe başlanır.

Silisyumdan Bilgisayara

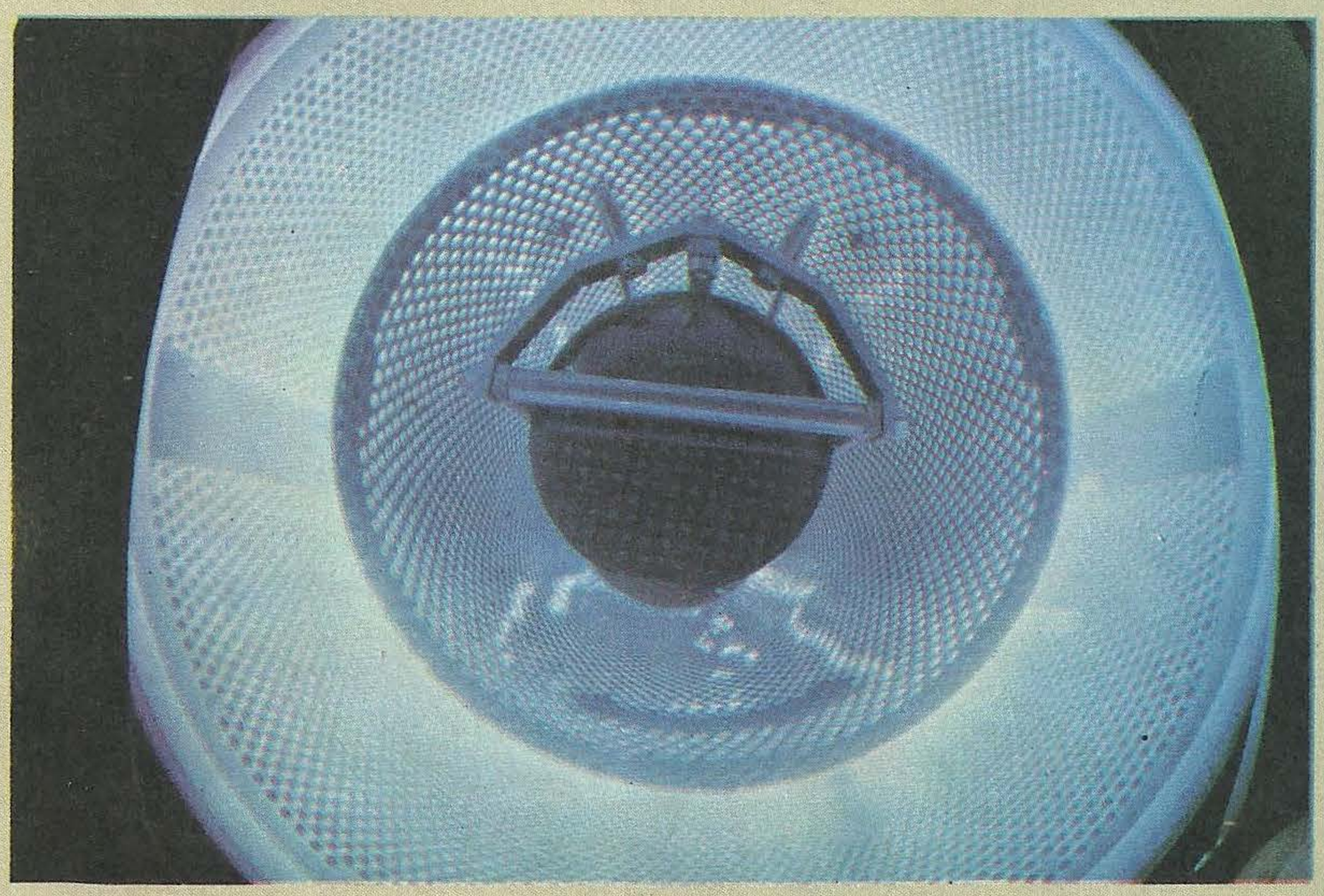


Mikro devrelerin tasarımı ve çiziminde CAD (Computer Assisted Design- bilgisayar destekli tasarım)'ın sunduğu tüm olanaklardan yararlanılmaktadır. Yandaki fotoğrafta yüksek ayrımlılığa (resolution) sahip çizici (plotter) ile bir devre tasarımı görülmektedir. Bilgisayar, yapıyı belleğine almakta ve istendiği anda bir terminalin ekranında görüntülemektedir.

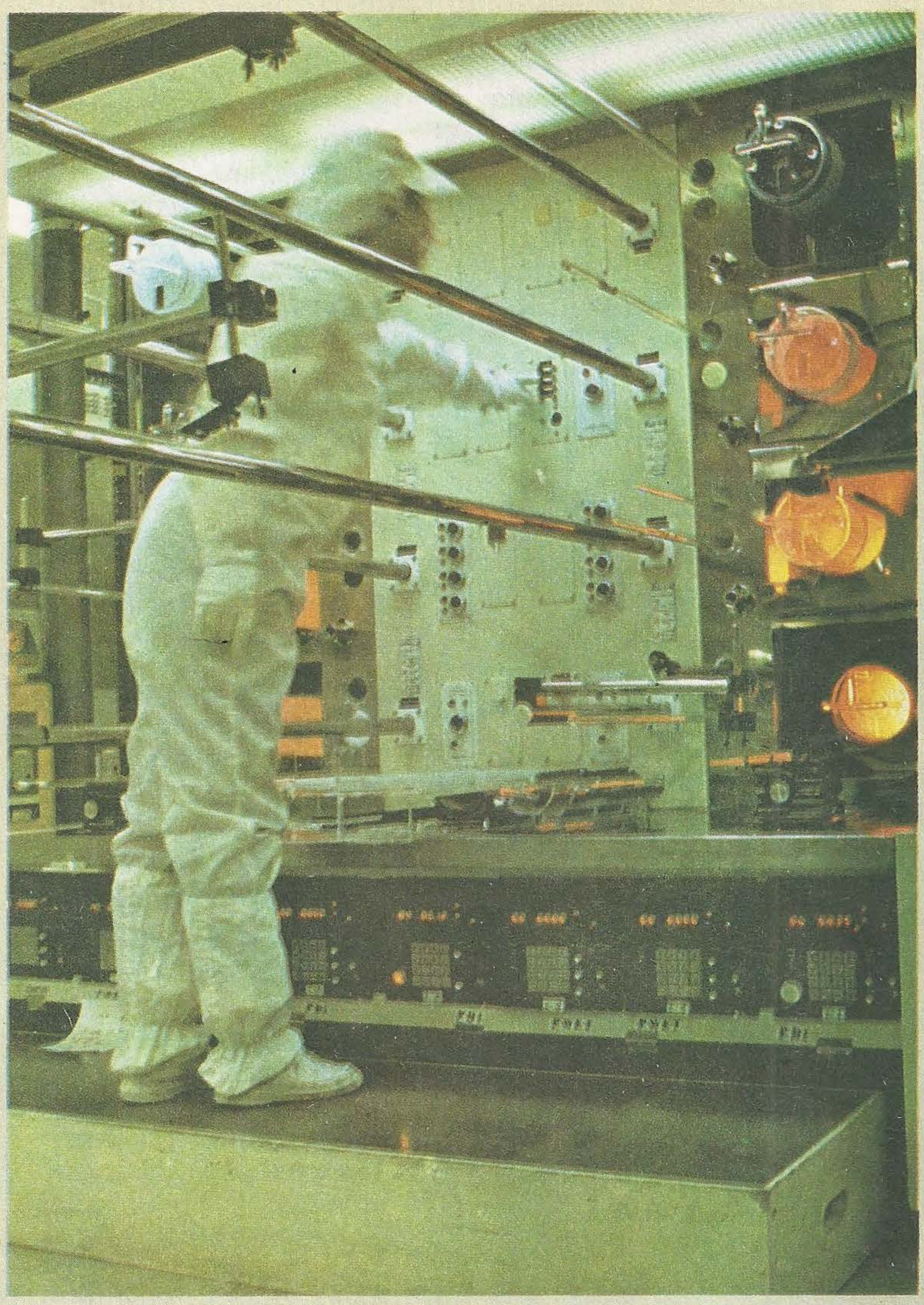
Böylece tasarımcılar yapı kesinleşinceye dek onun üzerinde değişiklik yapabilir ve fotolitografi yöntemi için gerekli kalıpları çıkarmak üzere fotoğrafını çekebilirler.

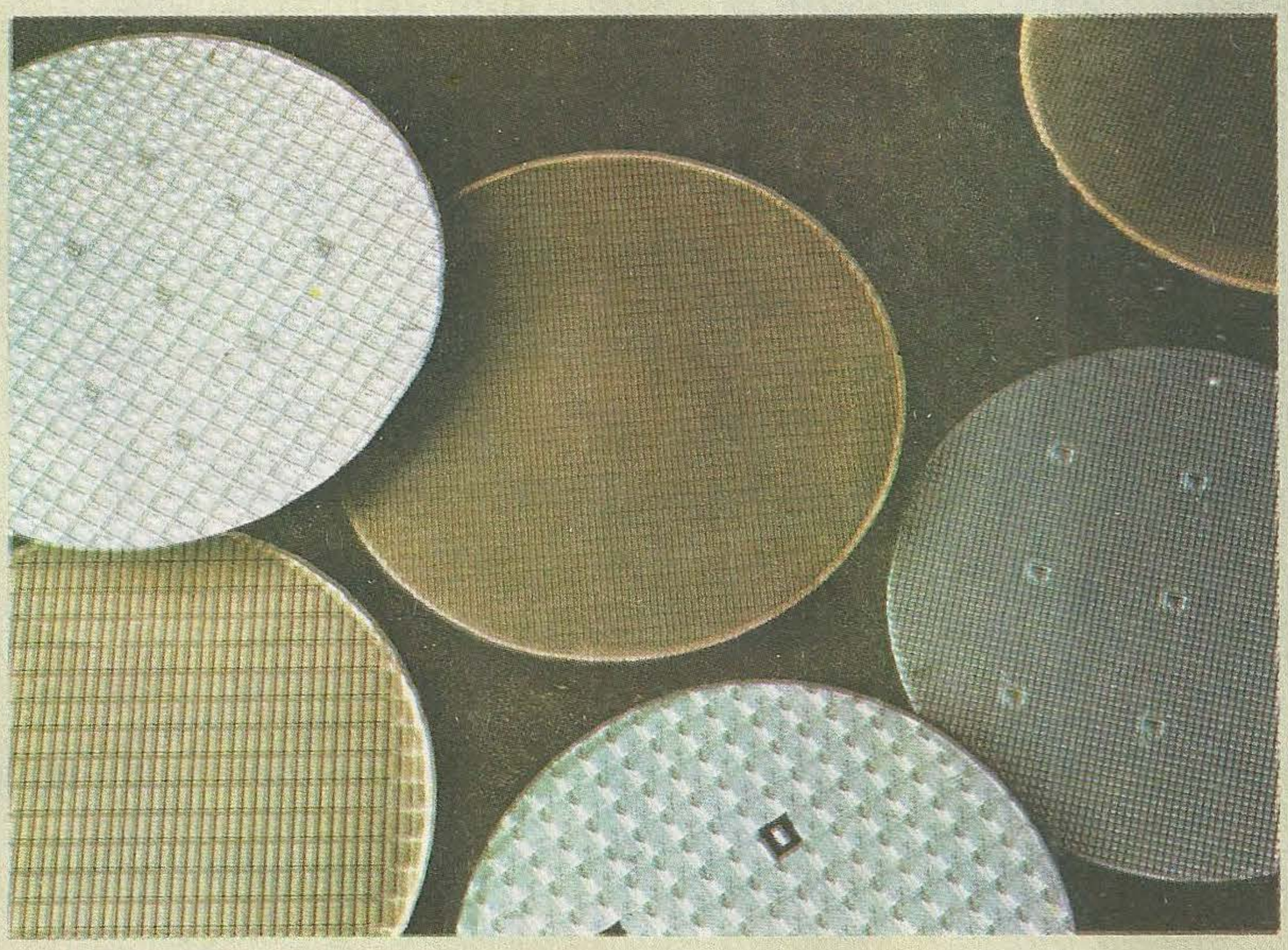


Yuvarlak plakalar, istenen elektriksel özellikleri elde etmek üzere, üzerine bazı katkı maddeleri eklenmiş saf bir silisyum eriyiğinden oluşan yarı-iletken bir çubuktan (mono kristal) kesilerek alınır. Silisyum atomları bir ''çekirdeğin'' çevresinde toplaşarak mono kristali büyütür ve hemen hemen kusursuz bir kristal yapı meydana getirirler. Ortadaki fotoğrafta bir silisyum çubuğun elde edilmesi görülmektedir.



Daha sonra mono kristal işlenerek çapı eşitlenir. Bu çubuk enine ince dilimler halinde kesilerek pullar (wafers) elde edilir. Bu dilimler özenle düzeltilerek parlatılır. Mikro devrenin çeşitli tabakaları fotolitografi (fotografik baskı) yöntemiyle (bkz. s.360) pulun üzerine basılır. Yandaki fotoğrafta iletken alanların oksijen plazmasıyla bombardıman edilmesi görülüyor.





Özel katkı maddeleriyle birlikte quartz tüplere yerleştirilen plakalar (yandaki fotoğraf) yüksek ateşte bir fırına yerleştirilir. Buharlaşma sonucu katkı maddeleri gaz haline geçince atomları dağılır ve böylece fotolitografi yöntemiyle oluşturulan ''pencereler''den silisyumun içine nüfuz ederler. Böylelikle mikro devrenin transistörleri, rezistansları ve diyotları oluşur. Baskı sona erdiğinde yarı-iletken plakalar alttaki fotoğraftaki görünümü alır.

Plakaların yüzeyi tamamıyla kaplanıncaya dek, mikro devre kalıbı baskı yöntemiyle üzerlerine uygulanır.

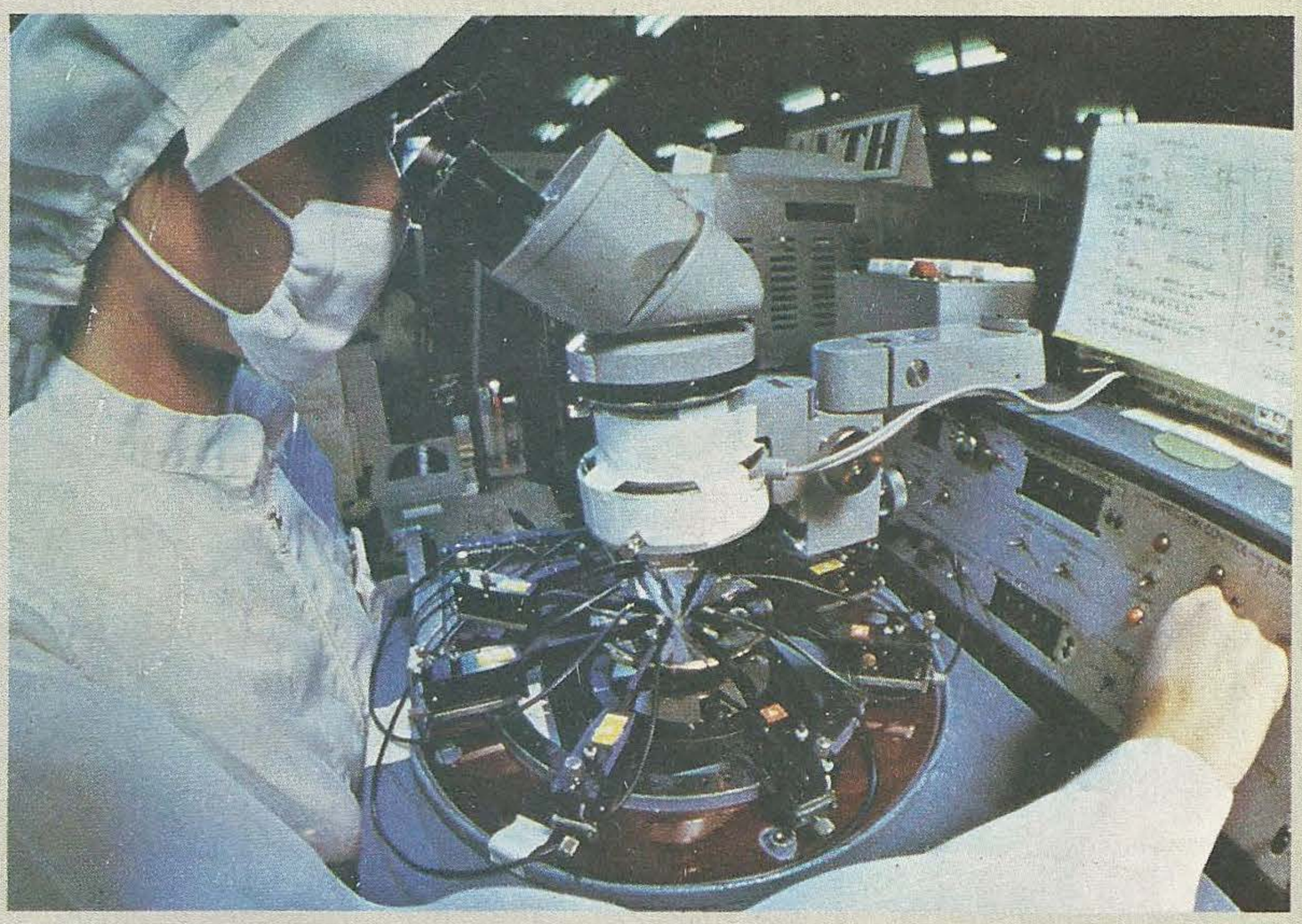
Mikro devrenin boyutları bir yuvarlak plakadan elde edilebilecekk opya sayısını ve buna bağlı olarak her bir yongarın maliyetini belirler. Fotoğrafta yuvarlak plakanın sınırında bulunan mikro devrelerin, dilimlerin yuvarlak olmasından ötürü, eksik olduğu açıkça görülmektedir. Bu devreler daha sonra dışarıda bırakılacaktır.

Kesme işleminden önce her yuvarlak plaka mikroskop altında özenle incelenerek bozuk devreler saptanır. İleride görülebilmelerini sağlamak üzere bunlar, bir cila ile işaretlenir.

Bozukluklar çok değişik nedenlerden kaynaklanabilir. Fotolitografi yönteminin etki yaptığı alanların çok küçük olmasından ötürü, çıplak gözle görülemeyecek denli küçük toz zerrecikleri bazı bağlantıların oluşumunu engelleyebilir.

Bu nedenle, tüm çalışma

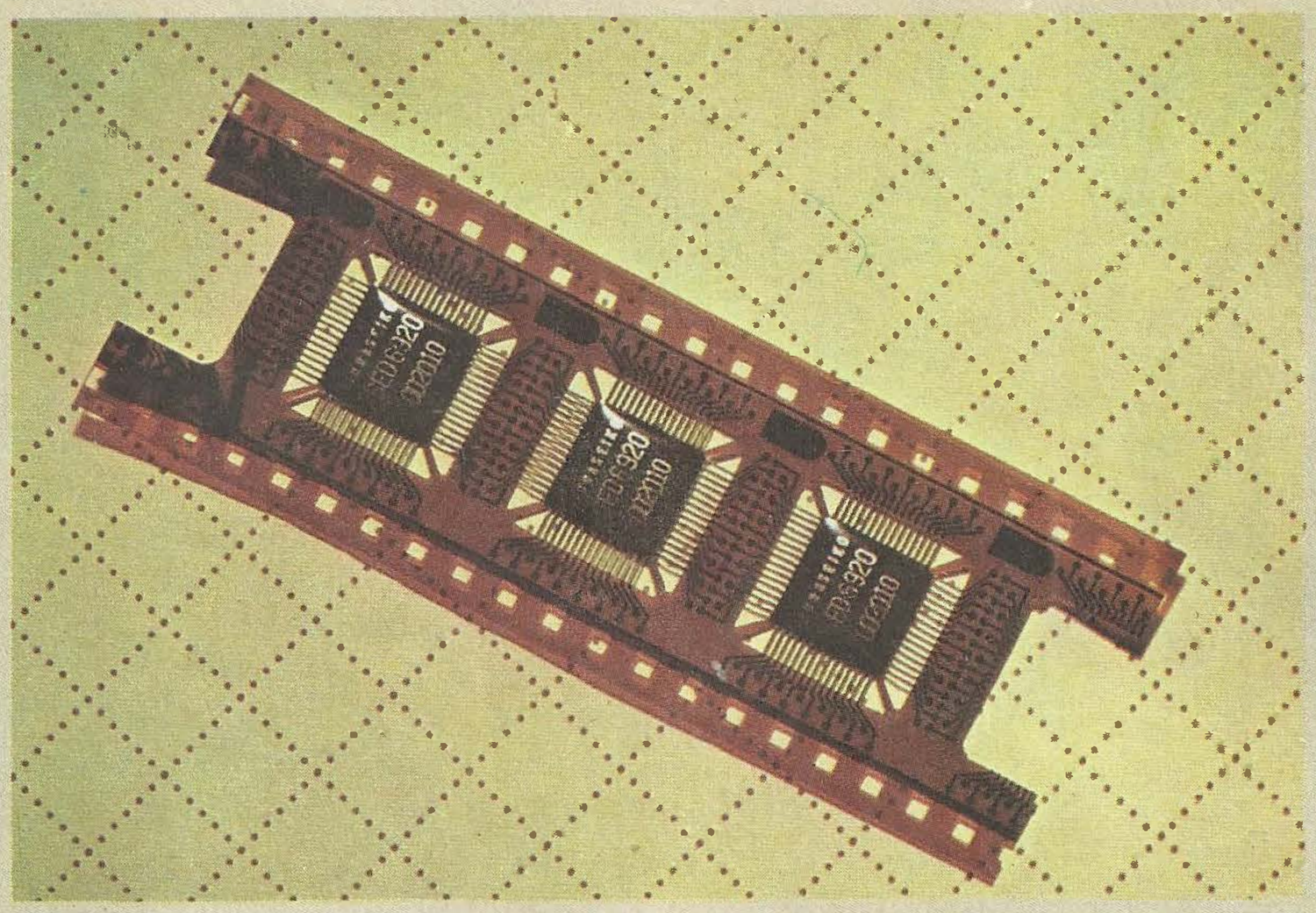


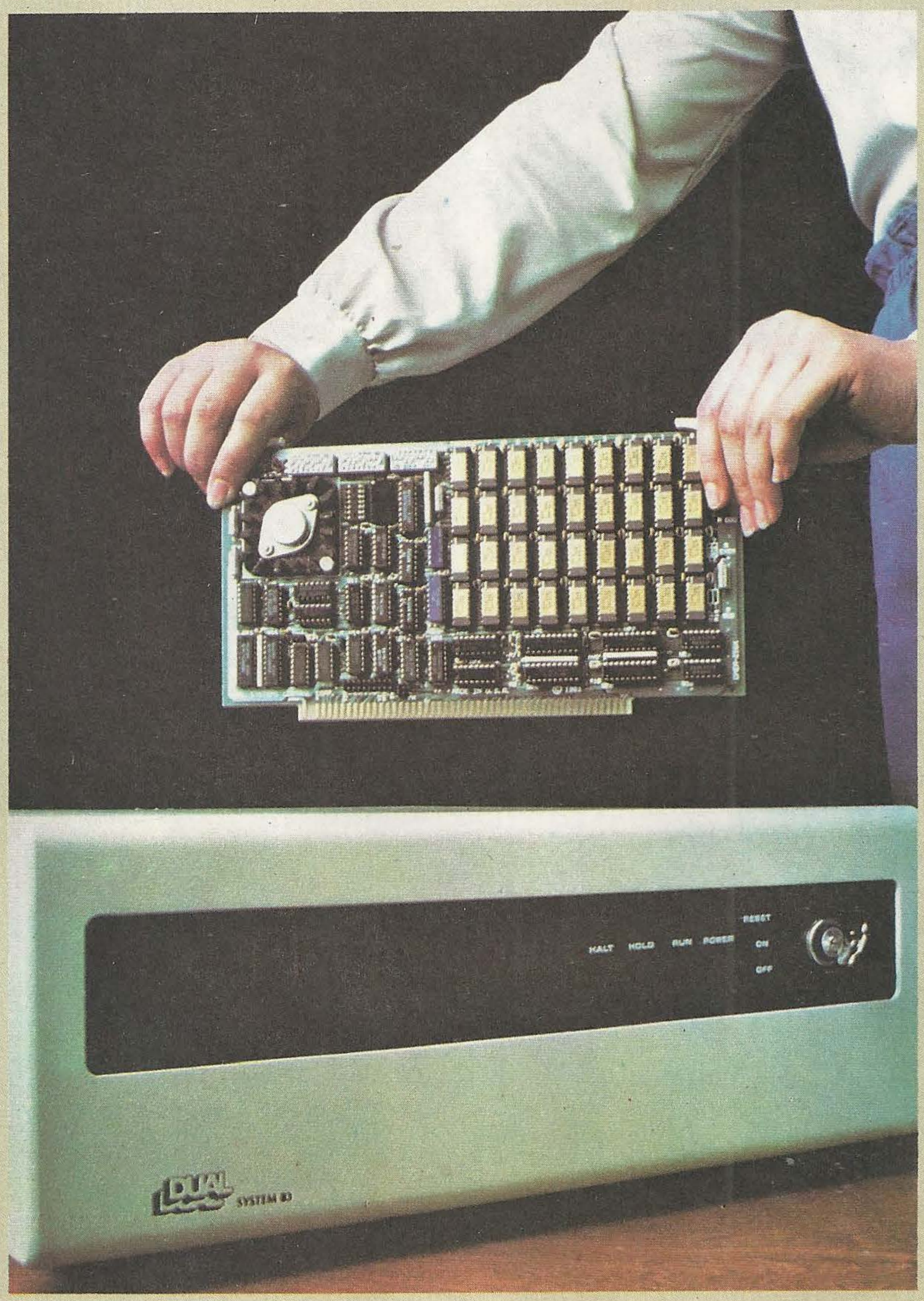




sıkı sıkıya denetlenen bir ortamda ve hava arıtılarak yürütülür. İşlemci bayanın giysisi (yanda ortadaki fotoğraf) denetimin sıkılığı konusunda bir fikir vermektedir. IBM şirketinin East Fishkill'deki (New York) laboratuvarında, bu yakınlarda, yonga üretim randımanını büyük ölçüde artırabilecek otuza yakın aşamadan oluşan bir gözetim tekniği geliştirilmiştir. Bir transistör üretmek için, bir silisyum ''demiri'' üzerine doping adı verilen katkı maddeleri konulmaktadır. İşte bu dopingler transistörlerin karakteristik ögeleri ya da bileşkeleridir. Doping atomları önceden saptanan yerlerine büyük bir hassaslıkla yerleştirilmelidir. Bu hassaslık gösterilmezse ürün bozuk çıkar. Doping maddelerini önceden saptanan noktalara yerleştirebilmek için son derece özenli boyut denetimleri yapılması gerekir. Yeni geliştirilen bir deneysel yöntemle kalıplar, devrelerin minik bir yonga üzerine doğru olarak yerleştirilmesini sağlayacak biçimde sıralanmaktadır. Kalıpları yerleştirirken büyük dikkat göstermek gerekir. Bir insan saçından 2500 kez daha ince, yani bir mikron boyutundaki yüzbinlerce bileşkenin yalnızca 40 mm² bir alan kaplayan yongalar üzerine büyük bir titizlik içinde yerleştirilmesi gerekir.

Yuvarlak plakalar üzerinde henüz toplanmış olan mikro devreleri optik incelemeden sonra (ortadaki fotoğraf) işlevsel denetimden de geçirmek gerekir. Her devreye bilgisayarın gözetiminde bulunan bir denetleme programı uygulanır.





Makine iletken uçları mikro devrenin ilgili yerlerine otomatik olarak getirerek zayıf güçte elektrik sinyalleri yollar. Devreden gelen yanıt tasarımda yer alan standart yanıtla karşılaştırılır. Eğer farklılıklar ortaya çıkmışsa daha sonra kolayca saptanması için işaretlenmiş olan bileşke elenir.

Denetimler bitince mikrometrik bir makinenin ucunda bulunan elmaslı bir levha yardımıyla mikro devreler birbirinden tek tek ayrılır. Yongalar kablo makinesine sokularak ilgili noktalarına devrenin dış dünya ile bağlantısını kuran teller konur (önceki sayfa en alttaki fotoğraf).

Bağlantı tellerinin konulmasından sonra yonga seramik bir kaba yerleştirilir. Yongayı tamamen içine alan bu kap telleri dışarda bırakır. Tek tek birimler daha sonra, fotoğraf filmine çok benzeyen ve yukarıdaki fotoğrafta görülen seramik kapların dışına taşan telleri koruma işlevine sahip kesintisiz bir şeride monte edilir. Kenarlarındaki delikler bileşkeler monte edilirken ve şeritlere tutturulurken birimlerin zarar görmemesi içindir. Yandaki fotoğrafta bir makine üzerinde kendisine ayrılan yere girmeye hazır bir plaka bileşkesi görülüyor. Bu yerleştirme işlemi, bağlantıların yapılması için çoğu kez yeterli olmakla birlikte bazen kaynak daha güvenli sonuçlar verir.

Yazılım

Belli bir sorunun çözümüne yönelik olarak geliştirilen yazılım, ancak şu iki temel koşulu yerine getiriyorsa, işe yarayabilir.

- İstenen işlemleri yapması ve arzulanan çıkışları vermesi
- Yüksek bir güvenilirlik düzeyine sahip olması

İlk koşulun tartışılacak yanı yoktur. Program arzulanan sonucun elde edilmesini sağlamıyorsa iki seçenek sözkonusudur. Ya bu programda öngörülen talepler ya da programın kendisi değiştirilmelidir. Talepler ile elde edilecek sonuçlar arasındaki uyuşmanın sınanması doğrudan gerçekleşir; öngörülen girdiler verilince onların karşılığı olan tüm çıktıların elde edilmesi gerekir.

İkinci koşulun sınanması daha güçtür ve sonuç aynı ölçüde açık olmayabilir. Son derece gelişmiş olan sınama tekniklerinden yararlanılsa da, hata bulunmaması pek büyük bir anlam ifade etmez. Çünkü, ancak özel durumlarda farkına varılabilecek bir hata daima sözkonusu olabilir.

Yazılımda saklı olabilecek hataların boyutunu kavrayabilmek için Apollo Uzay Projesi çerçevesinde programların oluşturulması amacıyla 600 milyon dolar harcanmış olmasına karşın, tüm projede karşılaşılan aksaklıkların büyük kısmının yazılım hatalarından kaynaklandığı akıldan çıkarılmamalıdır. Yine bu tür bir hata, Venüs'ü araştırmaya yönelik bir projeyi başarısızlığa uğratmıştır.

Yine de bu yüzden tüm bu saydığımız örnekler bilgisayarların yetenekleri konusunda genel bir güvensizlik yaratmamalıdır. Olağan program uygulamaları çok karmaşık olmadığından, hata olasılığı yüksek değildir. Ayrıca ortaya çıkan bir hatayı gidermenin de oldukça kolay olduğu söylenebilir.

Hatasız bir program elde etmeye yönelik yöntem, birbirini izleyen ve bütünleyen iki evreden oluşur: **Önleme** ve **arayıp bulma.**

Önleme evresi projelendirme (tasarım) evresinden başlayarak program yazılımının tüm gelişim evrelerinde belirleyici bir rol oynamak zorundadır. Hataların aranıp bulunması ise sınama işlemi sırasında son anda devreye girer. Ne var ki, o da başlangıçta benimsenen yaklaşımdan etkilenir.

Yazılım projesi, hata içerme olasılığı yüksek

İtalyan iletişim şebekesi kontrol odasındaki cihazların etkileyici bir görüntüsü.



alanları ortaya çıkaracak biçimde yapılırsa, hataların önüne geçilmesi mümkün olabilir.

Bu konuda son derece karmaşık ve ayrıntılı yöntemler vardır. Aşağıda, burada anlattıklarımız da gözönünde tutulan makinelerde kullanılan belli başlı yöntemleri aktarıyoruz. İşlevsel yönden eksiksiz bir uygulama programı elde edilmek isteniyorsa aşağıdaki noktalara dikkat etmek gerekir:

— Hedeflerin tanımlanması
— Sistemin, özellikle dış dünya ile bağlantıyı sağlayan ara birimlerin yapısı
— Yazılımın yapısı
— Programlama yöntemleri ve dillerinin seçimi

Ulaşılması gereken ilk sonuç varılacak hedeflerin çok iyi bilinmesiyle sağlanır.

İki tür hedef vardır: Kullanıcı tarafından belirlenen işlevsel hedefler ve bir programda bulunması zorunlu genel hedefler.

Her gelişmiş programda öngörülen hedefler şunlardır:

Genelleştirme. Bir programın geliştirilebilmesi için gerekli yatırım daima büyük paralar harcanmasını gerektirdiğinden, çok özel bir amaca hizmet eden (başka deyimle az sayıda kullanıcıya yönelik olan) bir proje ekonomik açıdan tutarlı olmayabilir. Buna karşılık aşırı bir genelleştirme, programın gereğinden fazla yaygınlaşmasına, dolayısıyla hata olasılığının artmasına yol açar.

Başarılı bir proje mühendisi, her durumda belli bir genelleştirmenin son kullanıcıya ne kadar yarar sağlayacağını belirleyebilmelidir. Çoğu kez, bir dizi komut ve tercihin eklenmesiyle gerçekleşen aşırı bir genelleştirme akıl karıştırır ve programı yararsız kılar.

Uyarlanabilirlik. Genellikten yoksun bir programın eksikliği, uygulanabilirlik, yani yeni fonksiyonların eklenebilmesi ya da çıkarılabilmesi özelliği ile giderilebilir. Böylece, bir program, fazlaca karmaşık bir yapı yaratılmadan değişik kullanımlara hizmet etmek üzere genelleştirilebilir ya da uyarlanabilir.

Bir yazılım paketinin uyarlanabilirliğini sağlamak için, farklı işlevler için çeşitli alt-programlar ya da daha karmaşık durumlarda, çeşitli programlar yazmak ve gereksinimlere göre birimler eklemek ya da çıkarmak gerekir.

Bakım. Bir yazılım üzerindeki düzeltme ya da yeni gereklere uydurma işlemlerinin büyük güçlükler doğmadan yapılabilmesi önem taşır. Bu özellik, güvenilirlikle doğru orantılıdır: Bir program ne kadar kolay düzeltilebiliyorsa, o ölçüde güvenilirlik özelliğine sahip demektir.

Program Mimarisi

Belli bir sorunun çözümlenmesini amaçlayan bir uygulama programı, her biri bir program ya da birkaç alt-program tarafından yerine getirilen farklı fakat birbirine bağlı bir işlevler bütünüdür. Bu işlevler arasındaki bağlantılar sistemin **yazılım mimarisinin** konusudur. Bu mimari aslında, fiziksel bileşkenlerin, aralarındaki bağlantılar ve ara birimlerin ekleme koşullarını belirleyen donanım mimarisi tarafından belirlenir.

Yaklaşımın temelinde, genel sorunun her biri farklı bir soyutlama düzeyinde bulunan ikincil sorunlara ayrılması yatmaktadır.

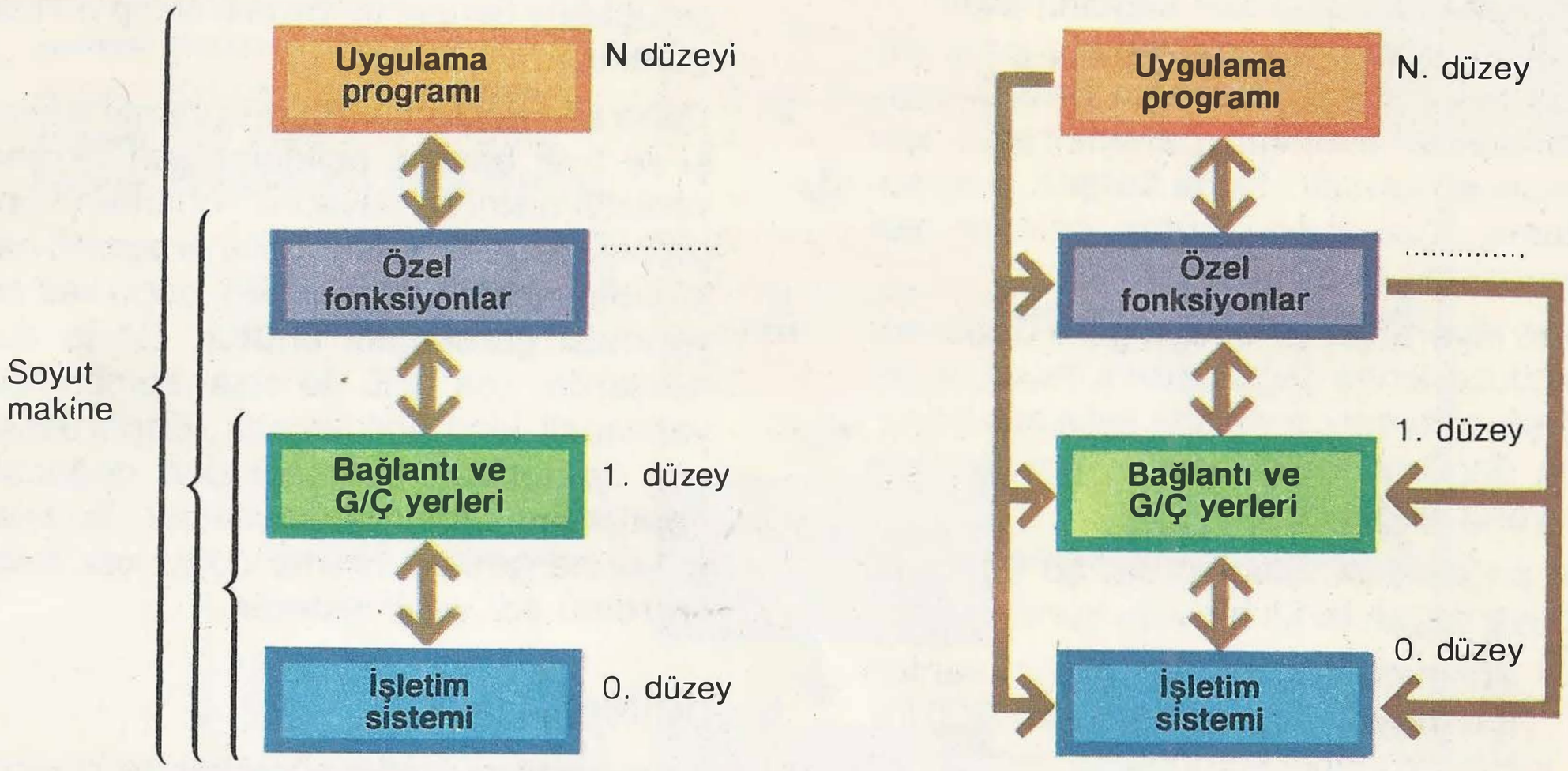
Bu çeşitli düzeylerindeki modeller iki farklı biçimde tasarlanabilir.

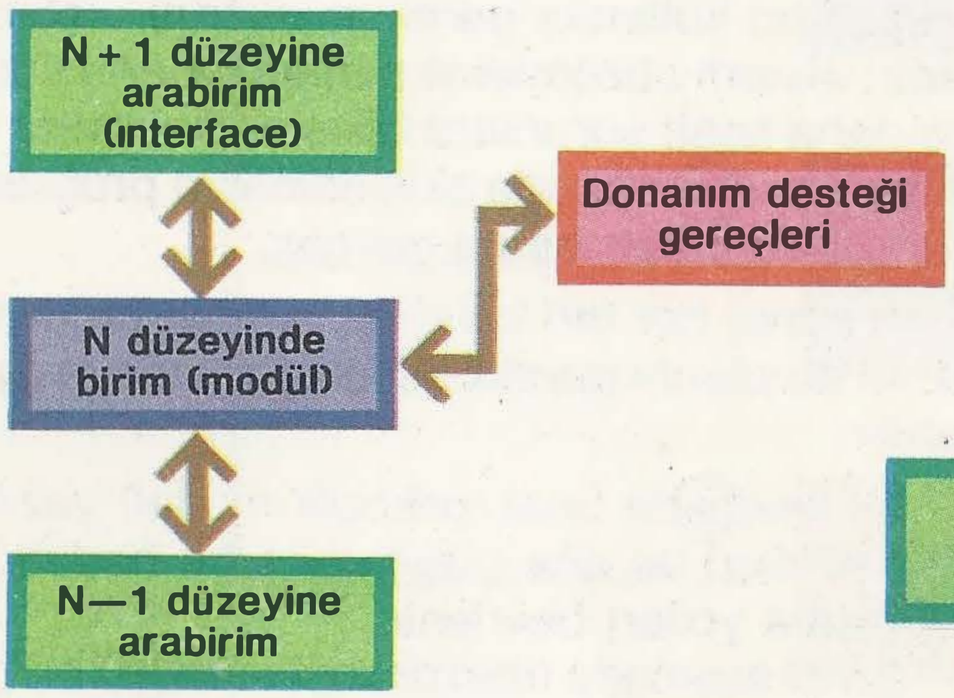
Seri olarak adlandırılan ilk yöntemde her düzey kendinden bir önceki ve bir sonraki ile bağlantılıdır. İkinci yöntemde ise bitişik düzeyler arasındaki bağlantıdan başka, belli bir düzey ile altındaki tüm düzeylerin ilişki içinde olmaları öngörülür. Karşı sayfada bu iki farklı yapıyla birlikte her düzeyin olası içeriği şematik biçimde verilmektedir.

Bu denli genel bir yaklaşıma dayalı bir çözümleme, özellikle büyük sistemler kurmaya yönelik karmaşık yazılım projeleri için yarar sağlamaktadır. Bununla birlikte, bu çözümlemenin genel mantığının ve genel kavramlarının küçük makinelerde uygulanması da mümkündür.

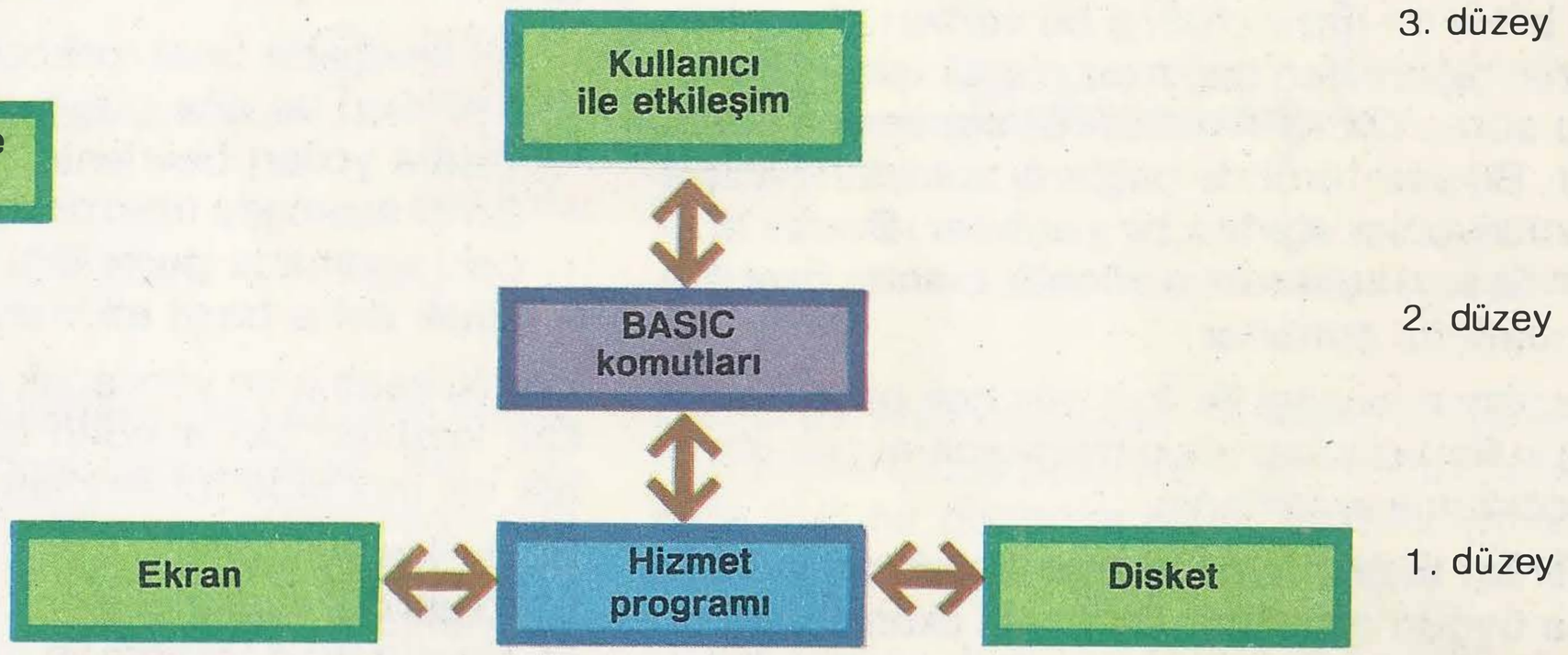
Her bir soyutlama düzeyi, yazılım-donanım bütünlüğünün, yetenekleri soyutlama düzeyine bağlı **soyut bir makine** olarak kavranmasına yarar: Soyutlama düzeyi 0 olan makinede sadece işletim sisteminden oluşur; 1 düzeyinde dış dünyayla makine arasında bağlantı noktaları ve sistemin alt-programları yer alır ve düzeyler böylece sürüp gider. Düzey yükseldikçe soyutlama derecesi de artarak sonunda en yüksek soyutlama derecesine, yani son kullanıcı ile diyaloğa ulaşılır. Örneğin, belli bir arşivi okumaya yönelik bir programın en üst soyutlama düzeyinde kullanıcıya hangi kaydı okumak istediğini soran alt-program yer alır. Onun bir altındaki düzeyde GET (kaydı okuma) komutu vardır. Bu komut bir alt düzey ile (Assembler diline çevirme) ve onun aracılığıyla 0 düzeyi ile (diskete erişim hizmet programları) bağlantı halindedir. Verilen örnekten bu tür yazılımın bazı özelliklerini çıkarmak müm-

YAZILIMIN YAPISININ SOYUTLAMA DÜZEYLERİ





Bir düzeyin tipik yapısı



Üç soyutlama düzeyinde bir BASIC programının şeması

kündür. Bunların başlıcaları şöyle sıralanabilir:

— Hiçbir düzey kendinden bir öncekinin özelliklerini bilmez.
—Farklı düzeyler arasındaki bağlantı önceden saptanmış bağlantı noktalarıyla sağlanır.

— Her düzey kendi yardımcı elemanlarına sahiptir ve bunların işlevsel yararlığından sorumludur. Bu nedenle bu elemanların diğer düzeylerinkilerle bağlantı noktaları aracılığıyla temas kurabilmesi sağlanmalıdır.
— Farklı düzeyler arasında veri alışverişi yalnızca belli değerler (örneğin BASIC fonksiyonlarında aktarılan parametreler) için sözkonusu olabilir. Buna karşılık aynı soyutlama düzeyindeki bütün birimler veri alışverişinde bulunabilir.

İşlevlerin hiyerarşik bir sıraya göre (farklı soyutlama düzeylerine dağıtılmasına dayalı bir yapı, değişik sistemler arasında iletişimin kurulmasının öngörülmediği yazılım projelerinde büyük yarar sağlayabilir.

İletişim sağlanmak isteniyorsa, şu üç temel kategoriye dayalı farklı bir yapı kurulmalıdır:

- Yönetim komutlarını içerip verileri içermeyen birimler (örneğin GOTO)
- Hem yönetim komutlarını hem de verileri içeren birimler (alt-programlar)
- Yalnızca verileri aktaran birimler (giriş kapıları arasında program bağlantıları)

Her biri birden çok giriş kapısına sahip iki birim arasında veri aktarımı aynı anda gerçekleşmeyen (eşzamanlı olmayan) bir süreç olarak görülür: İlk birim aktarılması gereken verileri ikinci verinin giriş noktasına yollar. İkinci birim de hazır olunca bu verileri okur. Her birim diğerinden bağımsız olarak işlediğinden, bu süreç (SEND/RECEIVE) eşzamanlı değildir. Bu yazı türünde bağlantı noktalarıyla ilgili fonksiyonlar ağırlıklı bir yer tutar. Bunlar arasında son kullanıcıya yönelik olanlar özellikle önemli rol oynarlar.

Yazılımın işleyişi ile ilgili pek çok hata sistem ile kullanıcı arasındaki birimlerin esnek olmayışından kaynaklanır.

Her bir bağlantı noktasının yapısı her kullanıma uygun olmalıdır. Bu genel ilkenin yanı sıra bazı temel kurallara da dikkat edilmelidir:

1 / Mesajlar, kullanımlarına bağlı olarak, net ve eksiksiz olmalıdır. Tek tük bir kullanım için kısaltılmış mesajlar yeterli sayılsa bile, sık sık yinelenen mesajların, kullanıcının özel bir işlem yapmasına gerek kalmaksızın, anında anlayabileceği kadar açık olması gerekir.

2 / Veri, parametre ve komut girişlerinde kullanıcının, hata payı en aza indirgenmelidir. Öyle ki, girişte istenen bir parametre mutlak gerekli değilse kullanıcı onu dikkate almadan gerekli işlemleri sürdürebilsin.

3 / Tüm giriş ve çıkış mesajlarının kullanıcı ile diyalogu sürekli kılmak üzere mümkün olduğunca benzer bir biçime sahip olması gerekir.

4 / Daha ileri yöntemlerde programın işleyişi ile ilgili başlıca açıklamaların ekrana yansıtılmasını sağlayacak komutların öngörülmesi gerekir. Kullanıcı, programı nasıl kullanacağını öğrenirken, çoğu kez ne yapması gerektiğini unutur. Böyle durumlarda, çok özlü de olsa, zamanında yapılacak kimi açıklamalar, öngörülmeyen komutların girilmesinden doğacak hatalardan kaçınılmasını sağlar. Bu işlevi yerine getiren birime çoğu kez **help** (yardım) adı verilmektedir.

Modelleme

Uygulama yazılımı üretim sürecinin bir dizi çeviriden ibaret oluştuğunu söylemek yanlış olmaz. Son kullanıcı gereksinimlerini sisteme iletir; sistem çözümleme evresinde ana sorunu daha basit sorunlara (akış şemalarına) indirger ve en sonunda akış şemaları programlama dillerinden birine çevrilir.

Tüm süreç her biri kendi açısından farklı aktarım evrelerini gerekli kılan iki aşamaya ayrılabilir:

— İlk aşamada (makroskopik model) yazılım hedefleri ve ana çizgileriyle bu hedeflere varma yolları belirlenir.
— İkinci aşamada (mikroskopik model) bir önceki aşamada gerekliliği saptanan her etkinlik daha basit etkinliklere indirgenir.

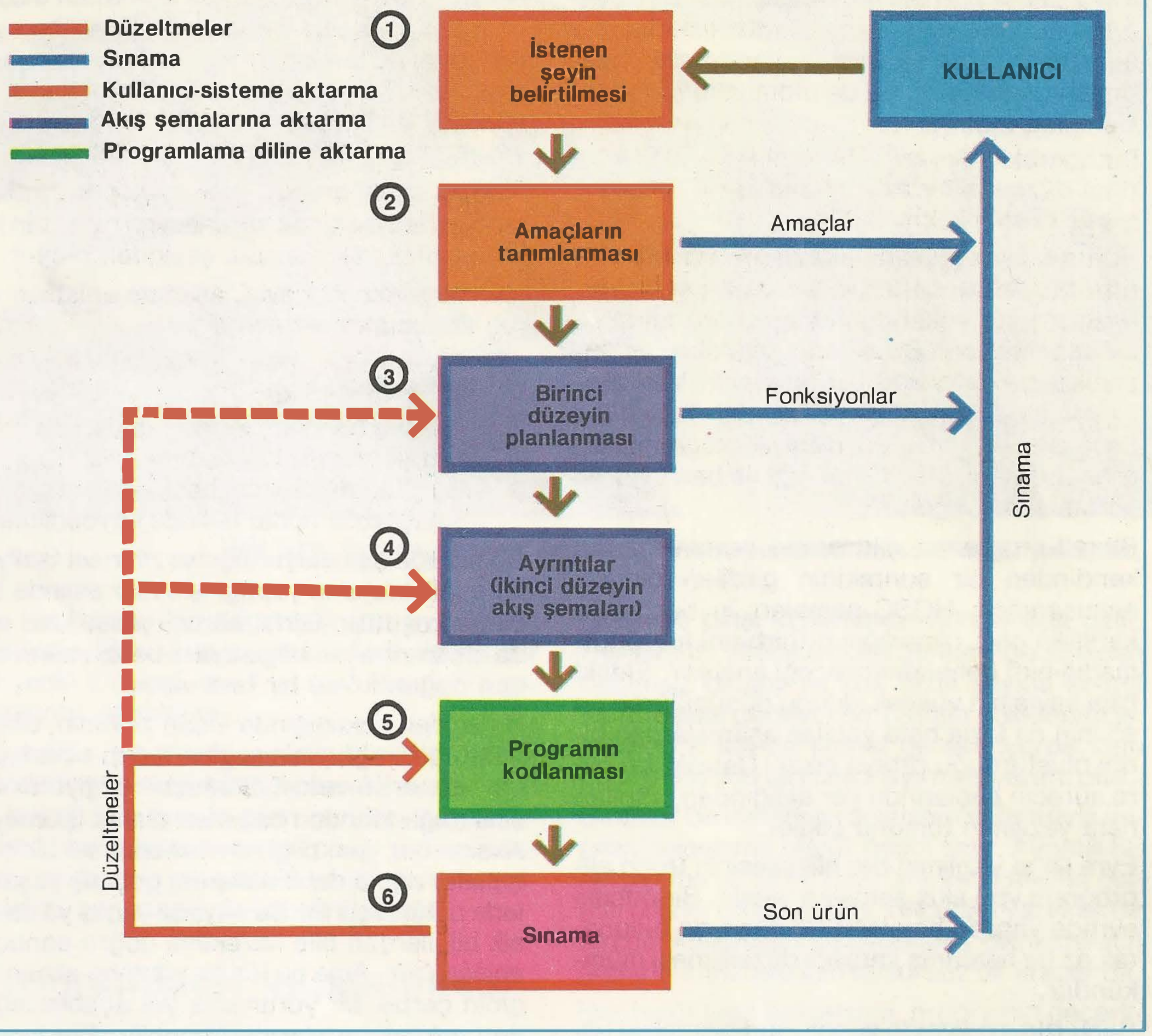
Her iki aşamanın yapılacak işlemler ve uyulacak koşullar bakımından bütünlük gösteren tek bir modelde toplanması mümkündür.

Böylelikle makroskopik ve mikroskopik olmak üzere iki düzeyden oluşan bir model elde edilir. Bu düzeyler tasarım ve özellikle hatalardan arındırma sırasında yol gösterici rol oynarlar. Hemen tüm durumlarda, hatalar, modellerdeki çeviricilerin birinden (ya da birkaçından) kaynaklanır. Model baştan incelendiğinde hatanın nedeni (kaynağı) ortaya çıkarılabilir.

Makroskopik model. Yukarıdaki şemada bir yazılım üretim sürecinde kullanılan makroskopik modelin basit bir biçimi görülmektedir.

Kullanıcı isteklerinin dile getirilmesiyle süreç başlar (1). Bu evrede yapılan bir hatanın geç farkına varılması pahalıya malolabilir. Çünkü,

YAZILIM ÜRETİMİNİN MAKROSKOPİK MODELİ



bu hatanın giderilebilmesi için yazılımın büyük ölçüde değiştirilmesi gerekecektir. Bu yüzdendir ki, son kullanıcının gereksinimlerini tam olarak ifade etmesinin ne kadar güç olduğunu unutmaksızın, daha ilk andan itibaren gerekli tüm önlemler alınmalıdır. Dolayısıyla çözümleyici (ya da programcı) makine-kullanıcı diyaloğunun tasarımını, kullanıcının tüm olası gereksinimlerini büyük bir titizlikle gözönünde bulundurarak yapmalıdır.

İkinci adımda ulaşılacak hedefler tanımlanır (2). Bu adım, bir önceki düzeyin makineyle daha yakından ilgili bir başka düzeye çevrilmesi olarak düşünülebilir. Bir başka söyleşiyle bu düzeyde, hedeflerden her birinin programlama zamanı ve donanım tasarımı cinsinden maliyeti saptanarak, maliyet/sonuç dengelerinin kurulması sözkonusudur. Kullanıcı ile sürekli bilgi alışverişinde bulunmayı gerektiren bu evrede belli bir donanım biçimi ve belli yazılım sınırları içinde gerçekleşecek olan ilk gerekler saptanır.

Kullanıcı otomasyon konusuna ne kadar yabancıysa, hedeflerin saptandığı evre o kadar önem taşır. Çoğu kez kullanıcılar yazılımla ilgili sorunları azımsayarak makineye gerçekte sahip olmadığı bazı üstünlükler atfederler. Bu yüzden kullanıcı, donanımın maliyetinin büyük ölçüde yazılımın maliyetine bağlı olduğunun ve bir dizi gereksinimin yanına pek de önemli olmayan bir diğerini eklemenin orantısız bir maliyet artışına yol açtığının farkına varmaksızın, makineden kendine gerekli olandan daha çok fazlasını bekleme eğilimindedir. Öte

yandan programcı da, kullanıcıyı, yalnızca en önemli gereksinimlere yanıt veren bir yazılıma yöneltmekten sakınmalıdır. Çoğu kez yan işlevler olarak görülen şeyler aslında zorunlu veri işlemleri durumunda olabilir. Başlangıçta bu işlemler dikkate alınmazsa, ileride elle yapılmaları gerekebilir. Bu da otomasyonun yararını sıfıra indirir.

Bir sonraki adım (3), tüm yazılım yapısının birinci düzeyde bir mantık şemasına (akış şeması) çevrilmesidir.

Bütün bu süreç uygulama programının tümü gözönünde bulundurularak izlenmelidir. En kestirme yol yukarıdan aşağı giden tekniğin uygulanmasıdır. Bu evrede bilgi akışı ve yapılması gereken işlemler belirlenir. Veri alışverişinin çeşitli noktaları arasındaki tutarlılığın denetlenmesi amacıyla, olası alt-programların, örneğin HGSÇ (HIPO) tekniği ile belirlenerek yorumlanması gerekir.

Bir alt-programın çıktılarının normal olarak kendinden bir sonrakinin girdileri olduğu anımsanınca, HGSÇ şemaları ile her şeyin karşılıklı olup olmadığının (birbirini tutup tutmadığının) denetlenebileceği anlaşılır. Eldeki bilgi sayısının yüksek olduğu düşünülürse bu adımın en fazla hata yapılan aşamalardan birini oluşturduğu ortaya çıkar. Üstelik, bu evre sürecin başlarında yer aldığından, yapılan hata yazılımın tümünü etkiler.

Evre (4)'te yazılımın her bileşkesinin (rutin alt-program vb.) akış şemaları yapılır. Bir önceki evrede yapılan hataların bu düzeyde bulunarak az bir harcama karşılığı düzeltilmesi mümkündür.

Bunu izleyen evrede (5) sorunun mantıksal çözümü bir programlama diline çevrilir (genel olarak son ''çeviri'' bu değildir, bir de Derleyicinin yaptığı son çeviri vardır).

Çoğunluğu bu evrede ortaya çıkan hatalar kolayca bulunup düzeltilebilir.

Sürecin bu noktada sona ermesi gerektiği düşünülebilir. Ne var ki, programlamanın bir uzantısı halindeki deneme ve sınamalar (6) (en azından başlangıçta) onun ayrılmaz bir parçası durumundadır. Sınama hiç olmazsa ilk ağızda, denemenin bir uzantısı olarak ele alınabilir. Çünkü her iki eylem içiçe geçmektedir. Programın belli bir bölümü hazırlanınca sınanır ve bu sınamanın sonuçlarına göre gerekli değişiklikler yapılır.

Sınama, en azından başlangıç aşamalarında, kavramlaştırmanın genişletilmesi olarak anlaşılabilir, çünkü her iki iş akışı birbiriyle içiçe geçmektedir: Programın belirli bir kısmı hazır duruma geldiyse, sınanır. Ama bu aşamadaki sonuçlar da potansiyel değişikliklere neden olabilir. Dolayısıyla, burada açıklanan model, hem program planlamasında temel oluşturabilir, hem de tamamlanmış ya da yazılma aşamasındaki bir program hakkında bir karar verebilmek için, hata aramada bir araç ve bir karşılaştırma noktası olarak kullanılabilir.

Mikroskopik model. Makroskopik modelin formüle edilmesinde yararlanılan evrelerin her biri sayısız mikroskopik çeviriden oluşur.

Her mikroskopik çeviri, aşağıda anlatılan süreç sonucunda elde edilir:

1 / Bilginin, örneğin nihai kullanıcının ilettiği bir talebin kabulü
2 / İstenen sonucun elde edilmesi için gereken yöntemlerin düşünülmesi
3 / Bu düşünülenlerin başkaları tarafından anlaşılabilir notlar halinde kaydedilmesi.

Bu şekilde şemalaştırdığımız zihinsel faaliyetlerle, bilgisayarın yaptığı işlemler aslında birbirine koşuttur: Giriş, süreç, çıkış. Ama arada insan zihninin bilgisayara benzememesinden doğan köklü bir fark vardır.

Bu açıdan bakıldığında insan zihninin, olağanüstü bir çağrışımlar ağına sahip olmak gibi bir ''kusur''u vardır. Biz hiçbir bilgiyi tek başına, diğerlerinden bağımsız olarak işlemeyiz. Aksine her yeni bilgi, derhal zihinsel bir çağrışımlar ağına dahil edilerek, geçmiş yaşantılarla bütünleştirilir. Bu sayede yanlış ya da eksik bilgilerden bile hareketle doğru sonuçlar anlaşılabilir. Ama bu tür bir yöntem, alınan bilginin çarpık bir yorumuna yol açabileceğinden, sık sık hatalar da üretebilir. Son sonuç hatalı olmasa bile, bu sonuca çağrışımsal bir süreç aracılığıyla varılmış olabilir.

Bu sürecin yol açabileceği ikinci bir hata türü ise insan belleğinin sınırlılıklarından kaynaklanmaktadır. İnsan sık sık önemli bir noktayı unutabilir; bu takdirde süreç eksik bir halde yürütülmeye çalışılacaktır.

Bu tür hatalar, işin incelikleri bilindiği, örneğin belgeleme çok ayrıntılı bir biçimde yapıldığı, çözümleme yöntemleri çok açıkça tanımlandığı, probleme en uygun olan programla dili seçildiği takdirde asgariye indirilebilir. Ancak bu takdirde bile, aynı program üzerinde birden fazla kişinin çalışıyor olması özel bir durum yaratır. Böyle durumlarda, üzerinde farklı kişilerin çalışmış olduğu program parçalarında çıkan hataların, son aşamada herkes birlikte çalışırken düzeltilmesi gerekir.



IBM 370. Ortada görülen veri aktarım sisteminin ekranının kontrol panelidir.

Bu tür hatalardan kaçınmak için uygulanan yönteme ''bir fazla-bir eksik'' adı verilir: Bir projenin n evresi, n—1 ve n + 1 evrelerini yapanlarca denetlenir.

Böylelikle, bir önceki evrenin (n—1) projecisi kendi biriminin çıkışlarıyla, incelenmekte olan (n birimi) için gerekli olan girişleri karşılaştırırken, bir sonraki evrenin (n + 1) projecisi incelenen birimin çıkışları ile kendisininkinin girişlerini uygunluk yönünden denetler.

Programların yapılanması

Genel proje evresi (yazılım mimarisi) sona erdikten sonraki adım tüm yazılımda öngörülen birimlerin belirlenmesidir.

Her birim alt-programlara ayrılabilir ya da kendi başına bir program oluşturulabilir. Her iki halde de üç temel özelliği vardır:

1 / Gördüğü iş
2 / İşleyiş mantığı
3 / İşlediği bağlam

Görülen iş bir önceki evrede belirlenmiştir. Bu betimleme birim çağrıldığında yaptığı işlemlerin betimlenmesinden oluşur.

Bu işlerin ''nasıl'' yerine getirildiği sorusu (örneğin algoritmalar) işleyiş mantığını verir ve kullanılan dil ile birlikte programcının tercihleriyle yakından bağlantılıdır.

Yazılım ve donanımda birimin işlediği çerçeve (bağlam) denince, içine yerleştirilebileceği çevre ve olası kullanım özellikleri akla gelir. Her birimin, mümkün olduğunca diğerlerinden bağımsız olması ve çok fazla sayıda komut içermemesi gerekir. Bir birimin yüklendiği işlevler çok sayıda komut gerektiriyorsa, onu küçük boyutlu birimlere ayırmak yararlı olur.

Birimler belirlenirken açıklama satırlarıyla yapılan belgeleme büyük önem taşır. Ayrıntılı bir belgeleme bir birim hazırlanırken çalışma planını ortaya çıkardığı gibi, önceden yazılmış bir birimin kalitesinin ve uygulama olanaklarının değerlendirilmesi için kusursuz bir araçtır da.

Her halükârda belgeleme, programın geçerliliğinin sınanması için vazgeçilmez bir araçtır. Belgeleme en azından aşağıdaki bilgileri içermelidir:

- **Birimin adı:** Bazı BASIC biçimlerinde (ve örneğin FORTRAN gibi diğer dillerde) alt-programlar simgesel (alfabetik) bir adla belirtilebilir. Örneğin standart BASIC'de satır 1000'de başlayan disketten veri okuma alt-programı GOSUB 1000 komut ile çağırılabilir. Bu alt-programın giriş noktası (entry point) satır 1000'dir. Başka BASIC türlerinde (ya da başka dillerde) alt-programa DISKET gibi bir ad vermek mümkündür. Bu durumda

giriş noktası DISKET'tir ve çağrı komutu da GOSUB DİSKET olur.
Her iki durumda da dökümantasyonda birimin adının (BASIC'te 1000, diğer durumda DİSKET) bulunması gereklidir. Genelde, bir birim, her biri ayrı ayrı belirtilen birden fazla giriş noktasına sahip olabilir.

- **Yapılan işler** Birim tarafından işlerin açıklanması görülen işler
- **Parametreler** Kullanılan değişken ve sabitler listesi
- **Girişler** Girişteki verilerin betimlenmesi
- **Çıkışlar** Elde edilen verilerin tanımlanması
- **Hatalar** Olası hata koşullarının, onlarla ilgili çentik değerlerinin ve olası düzeltme eylemlerinin betimlenmesi

Yukarıdan-Aşağı Geliştirme Tekniği

Yukarıdan aşağı yöntemi, özet olarak, akış şemaları ve programların oluşturulmasına ilişkin bölümlerde sergilenen kavramların sentez halinde uygulama düzeyine geçirilmesidir. BASIC bilgisi bu konunun anlaşılmasını kolay ve somut kıldığından, mantıksal sırayı bozmak pahasına da olsa, bu tekniği burada anlatıyoruz.

Yazılım projesi yapılırken ana sorun kolayca yorumlanabilecek (çevrilebilecek), düzeltilebilecek ya da yeni fonksiyonlarla zenginleştirilebilecek bir yapı elde etmektir. Belli zorluk derecelerindeki bilinen pek çok teknikten biriyle bu hedeflere varılabilir.

Yukarıdan aşağı tekniği fazlaca karmaşık olmadığı gibi tümüyle geçerli bir yöntemdir.

Yöntemin temelinde, sorunun ana çizgilerinin saptanmasından sonra belli sayıda basit soruna ayrıştırılması yatar. İkinci düzeydeki bu sorunlar ya doğrudan çözümlenir ya da daha basit başka işlevlere ayrılır. Sonuçta ortaya çıkan yapı bir ağaç ve dallarına benzer. Ana sorundan yola çıkılarak, yerine getirilecek işlevleri genelden özele giden biçimde yansıtır. Her bir işlevi ya da işlevler grubunu yerine getirme görevi farklı bir alt-programa verilmiştir. Her alt-programın çıkışı bir sonrakinin girişi durumundadır. Ancak kullanılan değişken ad ve türlerinin birbirini tutup tutmadığına dikkat edilmelidir.

HGSÇ tekniği ile yapılan belgeleme burada büyük yarar sağlayabilir.

Yukarıdan aşağı yöntemi ile basamaklardan oluşan bir program yaratılabilir. Önce en önemli birimler hazırlanarak diğerleri yalnızca tasarlanır. Daha sonra bu sonuncuların yerine gerçek birimler konur. Her birim devreye sokuldukça gerekli denemeler yapılır.

Ortaya çıkan son derece esnek, kolayca değiştirilebilen (herhangi bir işlevi değiştirmek için onu yerine getiren birim ya da alt-programın değiştirilmesi yeterlidir) ve tüm yazılımın kolayca ve güvenli biçimde denetlenmesini sağlayan bir yapı çıkar. Bu tekniğin uygulanmasına şöyle bir örnek verilebilir:

Bir şirketin çeşitli satış elemanları vardır; her birinin sattığı mal (kod, miktar ve satış tutarı) ve müşterilerin belleğe alınması istenmektedir.

Bu satış verileri ile, eldeki stok miktarı ve her müşterinin alımlarının toplamı da düzenlenmek istenmektedir.

Çıkışta istenen bilgiler şunlardır:

— Her satıcının gerçekleştirdiği satış toplamı
— Stok durumu
— Müşteri durumu

Burada satışlar giriş durumundadır. Yapılması gereken işlemler ise stok, müşteri ve satıcı arşivlerinin tutulmasıdır.

Böylece dört kütük gereklidir:

— Satış muameleleri: Satışlarla ilgili veriler
— Stoklar: Her malın stok miktarı
— Müşteriler: Her müşterinin alım toplamı
— Satıcılar: Her birinin satış toplamı

Girilen veriler ''satış muameleleri'' kütüğünde toplanacak ve sonuçta (örneğin her akşam) bilgiyazım istenecek ve hatalar düzeltildikten sonra muamelelerle ilgili veriler arşivlenecektir. Aşağıdaki şemada kütüklerin yapısı ve veri kayıtlarının yazı düzeni verilmektedir.

Sayfa 783'te programın birinci düzey atış şeması görülüyor. Başlangıçta tüm alt-programlar taslak biçimindedir ve yalnızca ekran üzerinde bir mesajın görünmesine yarar. Böylelikle mantığın bir bütün olarak doğru olup olmadığına bakılır. Daha sonra her defasında bir alt-program sisteme eklenir ve sınanır.

Ana programda açıklamalara (boyutlamalar, değişkenlerin türü ile ilgili açıklamalar, fonk-

Hangi Yazılım?

B.W.Boehm'ün öngördüğü, ''yazılım bunalımı'', 50'lerin ortasından beri bilgi-işlem dünyasında fırtınalar koparıyor. İlk kez 1973'te ***Datamation*** *dergisinde yayınlanan bir şema (bkz. 778. sayfada sol alttaki şema) bu şiddetli alt üst oluşu ortaya koyuyordu. Gerçekte ise, yazılım maliyetlerinin donanım maliyetlerini çok gerilerde bıraktığının anlaşılabilmesi için daha birkaç yılın daha geçmesi gerekecekti. Ancak bundan sonradır ki, kalite denetiminin rolünün de ne kadar önemli olduğu anlaşılabildi.*

Yazılım bunalımı, içinde bir ''yazılım kalitesi bunalımını'' da barındırmaktadır. 778. sayfada bulunan şekildeki eksenler değiştirilerek, bu bunalım bir öncekine benzer bir eğriyle gösterilebilir (bkz. sağ alttaki şekil).

Bu bakımdan, yapılandırılmış programlamanın başlamasıyla yazılımda çok önemli bir ileri adım atılmıştır. Okura bu yöntemsel ve kavramsal yeniliğin önemini anlatabilmek için, başlıca evrelerini kısaca özetleyeceğiz.

FORTRAN, BASIC, ASSEMBLER vb. gibi geleneksel dillerle yazılan programların karmaşıklığının artmasıyla, programlama yöntemlerini gözden geçirme gereği kendini iyice duyurmaya başlamıştır. Bu yönde ilk adımı atan Prof. Edsger M.Dijkstra, ta 1965'te yazdığı makalede şunları söylüyordu: ''... Yazılımın ya da daha doğru deyişle bir yazılım programının kalitesi, içindeki GOTO tümcelerinin yoğunluğu ile ters orantılıdır.''

İkinci adım 1966 Mayıs ayında yapı teoremini ortaya atan C.Boehm ve Jacopini tarafından ortaya atıldı. Bu teoreme göre, ''... Herhangi bir yapılandırılmamış programın mantıksal şemasında yapılacak ufak-tefek birkaç değişiklikle, yapılandırılmış bir program elde edilebilir.''

1971'de Niklaus Wirth, yapılandırılmış programlamanın temel kurallarını formüle etti. Yine aynı yıl içinde, kısa bir süre sonra Dr. Halan Mills ve F. Terry Baker yeni yöntemin endüstriye uygulamasından doğacak yararları sergilediler.

M.Parnas tarafından geliştirilen ***birim ayrıştırma*** *yapılandırılmış programlama tarihinde bir ileri evreyi temsil eder. Bu tekniğin sonucu, programın son biçiminin belli bir ''soyutlama düzeyindeki'' birimlere ayrılmasıdır.*

Yine 1972'de Weinberg, Dahl, Dijkstra ve Moore konuyla ilgili ilk üniversite ders kitaplarını yayınladılar. Aralık 1973'te Datamation dergisinde yer alan bir dizi makale, yapılandırılmış programlamanın ''bir programlama devrimi'' olduğunu ilan ediyordu. O zamandan bu yana çok sayıda bu tekniği hayranlıkla benimseyerek hiç tartışmasız, adeta bir din gibi uygulamak için ellerinden geleni yaptılar.

Yapılandırılmış programlamada anahtar kavram, belli bir sorunun yalnızca esaslı noktalarını öne çıkarıp, buna bağlı olarak esaslı olmayan ayrıntıları ikinci planda bırakan ***soyutlama****dır. Örneğin, normalde programlama dillerini kullananlar, bellek alanlarının yerleşimleri gibi esaslı olmayan bazı ayrıntıların dışlandığı ''soyut makineler'' kullanırlar. Ayrıntıların bir yana bırakılmasına yarayan soyutlama, görevin görünürdeki karmaşıklığının azaltılmasına yarar. Başka deyişle soyutlama, belli bir sistemin kimi özelliklerini öne çıkarıp diğerlerini görmezden gelerek yapılan basitleştirilmiş bir sistem betimlemesidir. İyi bir soyutlama, önemli bilgiyi vurgulayıp önemsiz ya da tutarsız ayrıntıları, en azından geçici olarak, gözardı eder.*

Programlama sistemlerindeki soyutlama başka birçok alanda ''analitik model'' olarak adlandırılan modele çok yakındır. Gerçekten her iki süreçte de çözüm bekleyen ortak sorunlar vardır. Bunlar arasında, sistemin hangi özellikleri daha önemlidir; parametrelerin değişebilirlik sınırları nedir; hangi betimleme biçimleri kullanılmalı; geçerlilik denetiminde hangi yollar izlenmeli vb. sorular yer alır.

Başka alanlarda olduğu gibi, burada da, çoğu kez, model hiyerarşilerine başvurulur. Alt düzeylerdeki modellerde, daha üst düzeydeki modellerde görülen olguların ayrıntıları açıklığa kavuşturulur. Ancak kuramsal modeller gerçek sistemlerden o kadar farklıdır ki, bunları daima açık bir geçerlilik denetimi evresinden geçirmek gerekir. Bir modelin soyut anlatımı onun temel niteliklerini (specification) saptar. Bir alt düzeydeki model ise onun ''uygulamaya konulmasını'' ifade eder. Yazılımda kullanılan soyutlamalarla işlevsel özellikler ve elde edilecek sonuçlar öne çıkarılıp, tüm bunların ***nasıl*** *yapıldığına ilişkin ayrıntılar bir yana bırakılır.*

Prog ram ve dillerin organizasyonu ile ilgili tekniklerin birçoğu soyutlama ilkesine dayanır. Soyutlama tekniklerinin gelişmesi programlama öğrenimine değil, aynı zamanda sistemlerin **biçimsel niteliklerini** algılayabilme yeteneğimizin gelişmesine de bağlıdır.

Örneğin 1960'ta programlama yöntemleri ve dilleri konusundaki en önemli gelişme, programın bir bölümünü bir ad ve bir parametre listesiyle özetleme olanağı veren fonksiyon ve rutinleri mümkün kılıyordu. O zamanlar yalnızca sözdizimi denetimler yapılabiliyor ve belli bir soyut niteleme ile genellikle birden fazla teknik çağrılamıyordu. 1970'te gelişmeler, veri yapısı projeleri ve çeşitli teknikleri çağırmakta kullanılacak soyut nitelemeler üzerinde yoğunlaştı. Soyut nitelemelerle çağrılan tekniklerle programın gereklerinin gerçekten birbirlerine uyup uymadığı, matematiksel mantık ve dilin anlambilimine dayalı yöntemlerle sınanıyordu.

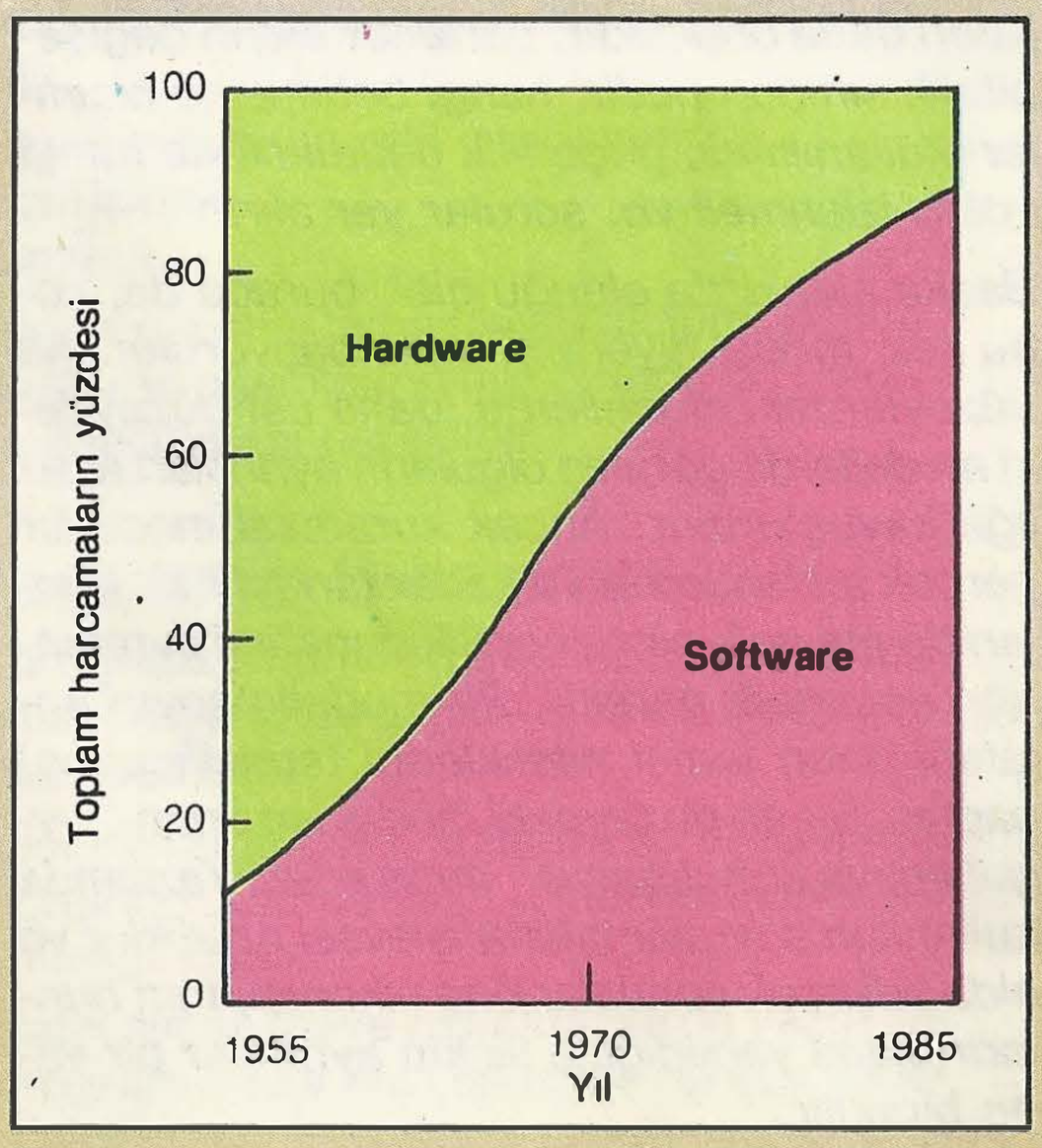
Yeni metedoloji ve programlama dilleri, eski diller (BASIC, FORTRAN, ALGOL vb.) ve programlama sistemlerinin karmaşık sorunlarının doğurduğu gereksinimlere yanıt olmak üzere ortaya çıktı ve gelişti. Bu yeni düşüncelerin ürünü olan dil evrim geçirdikçe, sorunların ortaya çıkarılarak, deneyler ışığında çözümler önerilmesi kolaylaştı.

Böylece **üst-dil** kavramına ulaşıldı. Bu kavram, başka diller "konuşan" biçimsel bir dili ifade eder.

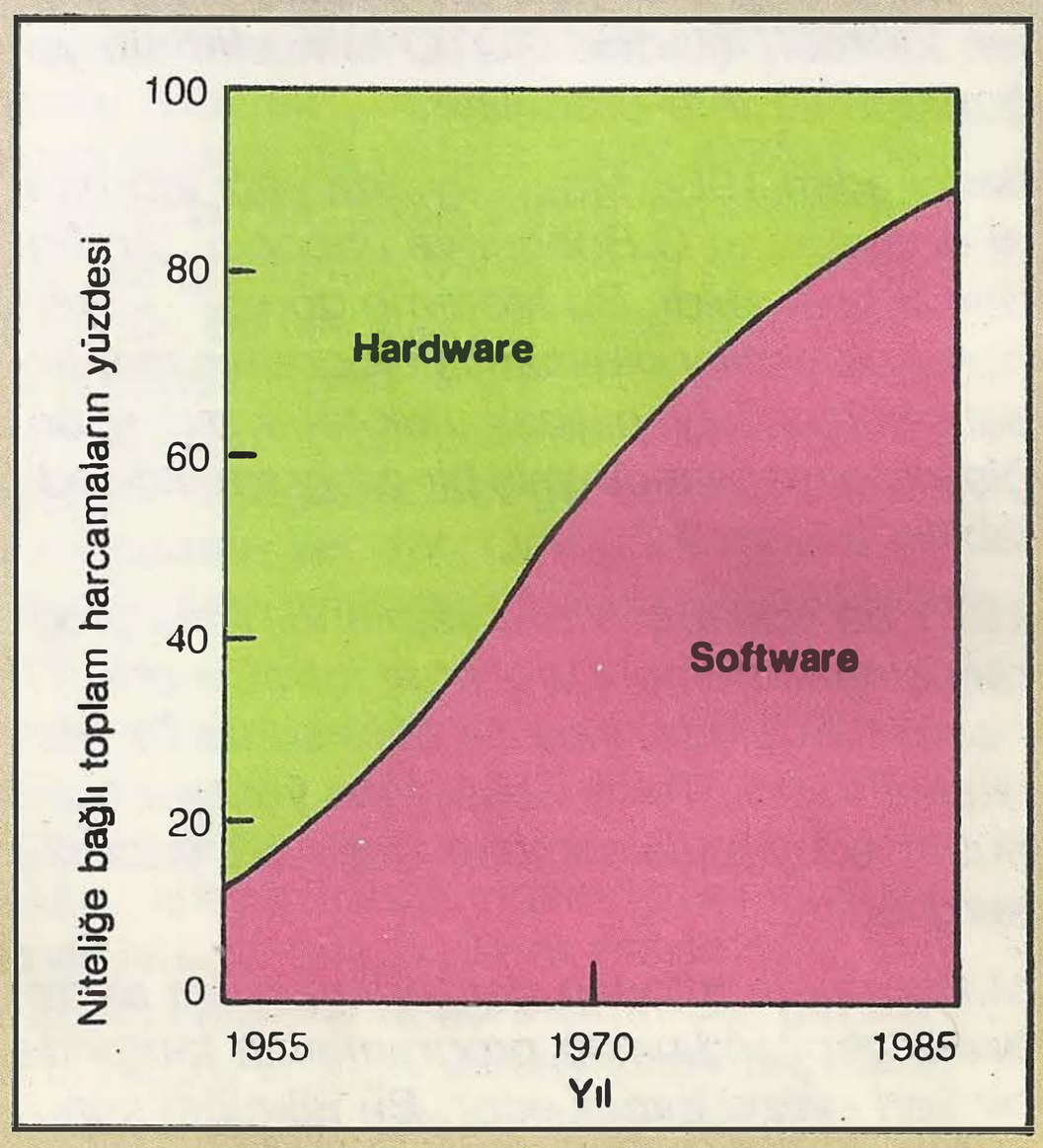
Şu anda, yeni diller ve özellikle cebirsel ya da biçimsel diller gibi üst düzeydeki dillerin bir s isteme oturtulması üzerindeki araştırmalar yoğunlaşmış durumdadır. Dünyaca ünlü matematikçi ve mantıkçılar vargüçleriyle evrensel geçerliliği olacak güçlü bir biçimsel dil bulabilmek için uğraşmaktadır. Birkaç örnek vermek gerekirse, tek geçerli programlamanın en ileri ve kesin matematik dalı olan soyut ileri cebire (VSM dili) dayandırılması gerektiğini savunan Viyana okulu, ya da "fonksiyonellere" (FP türü) dayalı bir programlama üslubuna sahip olan John Backus ve Naur gibi programcıların başını çektiği IBM okulu ya dayine soyut ileri cebir ve soyut diller kuramına bağlı SRI Uluslararası Okulu sayılabilir.

Bu arada, Amerika Birleşik Devletleri Savunma Bakanlığında modern dil algoritmalarını alışılmış denetim yapılarıyla birlikte kullanarak alt-prog ramları ve türlerini belirleyen ve sorunları paralel programlama yoluyla ve istisnaları kullanarak anında çözen ADA dili geliştirildi. Böylece gelecek hakkında kesin bir şey söylemek mümkün olmamakla birlikte, aşağıdaki üç gerekçeden ötürü, ADA türü dillerin gelişeceği rahatlıkla söylenebilir:

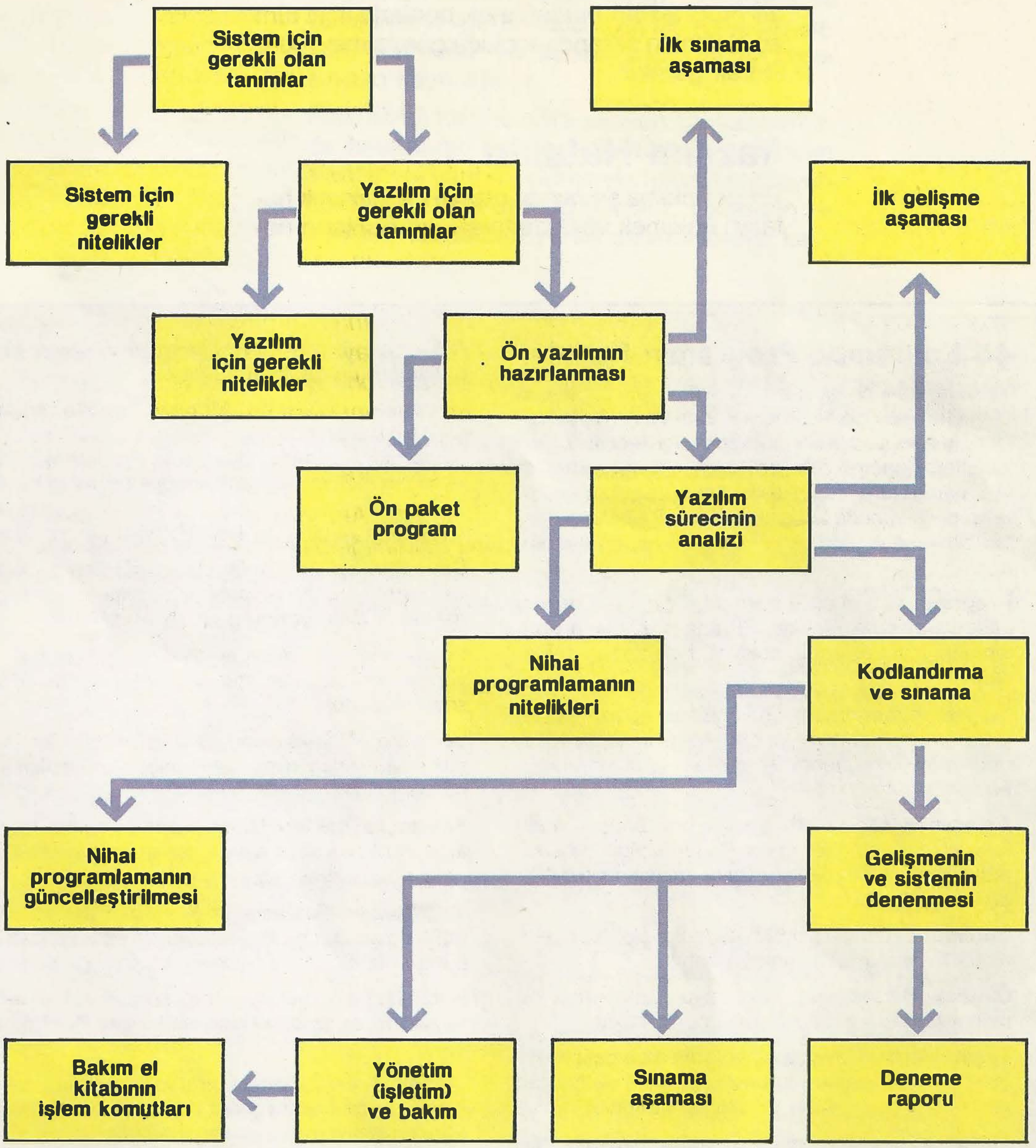
Bir veri işleme sisteminin maliyet bileşimi yüzdeleri



Bir veri işleme sisteminin kalite maliyetinin bileşim yüzdeleri



YAZILIMIN GELİŞME AŞAMALARI



1) Kalite özelliklerine ve bu arada güvenilirlik ve bakıma verilen önemin genel kabul görmesi.
2) Programlamanın insanî bir etkinlik (human engineering-warmware) olarak görülmesi
3) Gerek tek tek gerek toplu bir verimliliğe ulaşılması

(Giuseppe Quartieri'nin "La qualita del software" adlı ***DATA MANAGER,*** *no: 27 Ekim 1983'teki makalesinden)*

siyonların tanımlanması vb.) önemli bir yer ayrılır. Çeşitli alt-programlar hazırlandıkça bu açıklamaları azar azar yazmak ve sonuçta tüm alt-programları birleştirerek bunlarla ilgili tüm açıklamaları ekranda topluca görebilmeyi sağlamak gerekir.

Yazılım Hataları

Programlama sırasında ortaya çıkabilecek hataları önlemek ve düzeltmek için bunların ne-

45 Kavramda Programın Kalitesi

Yazılım. Belli bir işi yerine getirmek için bilgisayara yüklenmesi gereken tüm komut ve verilerin toplamı. İşletim sistemleri, güdümleri (gözeticiler), derleyiciler, deneme rutinleri, uygulama programları ve "paperware", başka deyimle programı tanımlayıp betimlemede kullanılan planlar, sınıflandırmalar, deneme verileri ve kullanım kurallarından oluşur.

Program. α gibi bir program bir dizi (s, t) durum çiftleri arasındaki ilişkidir. Burada s, α 'nın hesaplanması için başlangıç noktası, t ise sonuç noktasıdır.

Yazılım mühendisliği. Güvenilir ve doğru işleyen programlar yapmaya yönelik olarak kullanılan, verimli ve etkili mühendislik yöntem ve ilkeleri bütünüdür.

Anlaşılırlık. (Açıklık) Bir yazılım ürününün sonucu (son amacı) denetleyici tarafından açıklıkla kavranabiliyorsa, sözkonusu yazılım anlaşılırlık özelliğine sahiptir.

Bütünlük. Tüm parçaları hazırlanıp işler hale getirilmiş bir yazılım, bütünlüğü sahiptir.

Özlülük. Bir yazılımda bilgi savurganlığından kaçınılmışsa, onun özlü olduğu söylenebilir.

Taşınabilirlik. Kolayca aktarılabilen ve geliştirildiği sistemden başka sistemler üzerinde de çalışan yazılımlar taşınabilirlik özelliğine sahiptir.

Tutarlılık. İşaretler, simgeler ve terminolojide birlik içinde olan bir program tutarlıdır. Ayrıca dış tutarlılık için de içerikle projenin ilk özelliklerinin (primitive) türdeş olması gerekir.

Bakıma uygunluk. Bir yazılım ürününün bakıma uygunluğu, ürünün yeni taleplere kolayca adapte olabilmesine bağlıdır.

Kanıtlanabilirlik. Teknik özellikleri için kabul ve geçerlilik ölçütleri belirlenebilen her yazılım kolayca kanıtlanabilir. Kanıtlanabilirliğin iki yönü vardır:

— Kanıt ölçütlerinin tanımlanması
— Belirlenen ölçütlerin geçerliliğinin saptanması

Kullanışlılık. Bir yazılım ürünü, yöneldiği kullanım için uygun ve pratik ise kullanışlılık özelliğini taşır. Kullanışlılığın da iki yönü vardır:

— Başka projelerde tümüyle ya da kısmen kullanılabilme olanağı (taşınabilirlikle karıştırmamak gerekir)
— İnsan-makine bağlantısının ve çıktıların kalitesi ile kütüklerin ve giriş verilerinin doğru tanımlanması

Güvenilirlik. Geniş anlamda, öngörülen zamanlar ve çevre koşulları altında bir yazılımın kendinden beklenen işleri yerine getirme olasılığıdır.

Yapılanma. Bölümleri arasındaki ilişkiler belli bir yapıya göre düzenlenmiş yazılımlar için bu özellik sözkonusudur.

Verimlilik. Proje sırasında öngörülen hedeflere kaynak savurganlığı olmaksızın ulaşan bir yazılım ürünü verimlidir.

Yazılım kaynakları. Geniş anlamda, bellek büyüklüğü, toplam komut sayısı, açılan arşivler, G/Ç kanalının kapasitesi vb.

Bağımsızlık. Başlangıçta planlanan donanımdan farklı donanım gereçleriyle birlikte ve onlardan bağımsız olarak işletilen yazılım bu özelliğe sahiptir.

Kesinlik. Bir yazılımın verdiği sonuçlar, belirlenen uygulama açısından kesin ise bu özellik sözkonusudur.

Ulaşılırlık. Bir yazılımda giriş verileri açıkça tanımlanmış ve çıkışlar da gerek biçim gerekse içerik yönünden kolayca anlaşılır ve yararlıysa, yazılımın ulaşılırlık özelliği vardır.

Okunabilirlik. İyi işleyen ve komutların okunmasıyla mantığı kavranabilen bir yazılım ürünü okunabilirdir.

Artabilirlik. Verileri ya da denetim işlevleri kolayca artırılabilen bir yazılım için sözkonusudur.

Ergonomiklik. Kullanıcıya zaman ve enerji yitirtmeden, kullanıcının moralini olumsuz yönde etkilemeden işlevini yerine getiren yazılım, ergonomiktir.

Değişebilirlik. Yeni bir varyantı kolaycaeldе edilebilen her yazılım değişebilirlik özelliğine sahiptir.

Öz-betimleme. Bir okuyucu, hedefleri, ilişkileri, gi-

denlerini bilmek gerekir. Sistem tarafından otomatik olarak tanısı yapılan ve bu yüzden önemli bir sorun yaratmayan, sözdizimi ve sözcüklerle ilgili dil hatalarını bir yana bırakırsak; geri kalan yanlışların yarısı yazım hatasından kaynaklanır.

Fonksiyon türüne göre yapılan sınıflandırmada hatalarının yaklaşık % 15'inin ara birim alanlarıyla yani dış dünyayla teması sağlayan bölümlerle ilgili olduğu görülür. Hataların % 25'i ise komutların yanlış dizilmesinden kay-

riş/çıkışları çeşitli programların bileşke ve işlevlerini kolayca çıkarabiliyorsa yazılım bu özelliğe sahip demektir.

İletilirlik. Girişlerin tamamlanmasını kolaylaştıran ve biçim ve içerik yönünden kolayca anlaşılan ve yararlı sonuçlar üreten bir yazılım iletilirlik özelliğini taşır.

İşaretlilik. İşleme evresinde istenen sıranın izlenmesi için bir programa çentikler, etiketler (gerekirse açıklama tümceleriyle tamamlanabilecek başka işaretler) konulmasıdır.

Programın sınanması. Bir programın başlangıçta konan gerekliliklere yanıt verip vermediğinin belirlenmesine yarar.

Programın geçerliliği. Topyekün bir sistem çerçevesinde programın doyurucu olup olmadığının saptanmasıdır. Bunun için programdan beklenen niteliklerin değerlendirilmesi gerekir.

Programın onaylanması. Programın kalitesinin (güvenilirlik vb.) onanmasıdır. Yalnızca yetkili bir kuruluş tarafından yapılabilir.

Hatalardan arınma (debugging). Hataların belirlenmesi, yerlerinin saptanması ve giderilmesi süreci.

Programın denenmesi. Bir programın, elde edilecek sonuçların doğruluğunu saptamaya yönelik olarak işletilmesidir.

Yazılımın denetimi. Son denetim. Program, kullanıcının yararlanabileceği şekilde düzeltilir.

Özelliklerin (görevlerin) denenmesi. Bir yazılım ürününün harcanan makine zamanı ya da kullanılan bellek boyutu gibi işleyişle ilgili gereksinimlere yanıt olup olmadığının ortaya konması süreci.

Kabul. Bir yazılımın kullanıcı tarafından kabul edilir olup olmadığının ortaya konmasına yönelik süreçler toplamı.

Yazılımın yaşam döngüsü. Gereksinim ve görevlerin tanımlanmasıyla başlayıp, çözümleme, programlama, gerçekleşme, işletme ve bakımla süren döngü.

Yazılım modeli. Bir süreç ya da ürünü belirleyen bir büyüklüktür.

Algoritma. Genel olarak belli sayıda işletme bölümleri ya da sıralı süreçler toplamını tanımlayan bir metin (daha doğru bir deyişle, bir simgeler bütünü).

Güvenilirlik aralığı. Belli bir t zamanında sistemin çalışır durumda olması ve x süresince işlerliğini sürdürme olasılığıdır.

Hazır olma. İkiye ayrılır:

— Anlık hazır olma: Belli bir t zamanında yazılımın öngörülen teknik ve çevre sınırları içinde çalışması olasılığıdır.
— Hazır olma aralığı: Yazılımın, öngörülen sınırlar içinde olasılık hesaplarına uygun olarak işleyebilme süresidir.

Hata (Raslantısal). Yazılımda kusurlara yol açabilecek projeyle ilgili sözdizimsel ya da (alt düzeyde) işlemsel bir aksaklıktır.

Arıza (Sistematik). Yazılımın kendinden beklenen görevleri yerine getirmesini engelleyecek biçimde bozulmasıdır. Bir hata değişik arızalar doğurabilir.

Yanlış. Bir giriş verisi istenen işlevin doğru biçimde gerçekleşmesini önleyecek bir program hatası doğuruyorsa sözkonusudur. Bir yanlış,

— fiilen görülüyorsa açıktır
— aksi halde üstü kapalıdır (örtüktür)
— yazılımın işleyişini durduruyorsa kritik bir hal alır.
— tüm sistemi kilitliyorsa fiziksel, fiziksel olmadan kritikse mantıksaldır.

Güvenli yazılım. Yazılım hatalarından hiçbiri kritik bir arıza doğurmuyorsa güvenlidir.

Yapılandırılmış kod. Program çözümünden doğrudan mantıksal yapıların vurgulanması.

Arıza ağacı yöntemi. İlk ve raslantısal hataların tüm programa yayılmasını ve daha üst düzeydeki arızaların ortaya çıkışını inceleme yöntemidir.

*[Giuseppe Quartireri'nin **Data Manager**, No: 27, Ekim 1983 sayısında yayınlanan "La Qualita del Software" (yazılımın kalitesi] adlı makalesinden).*

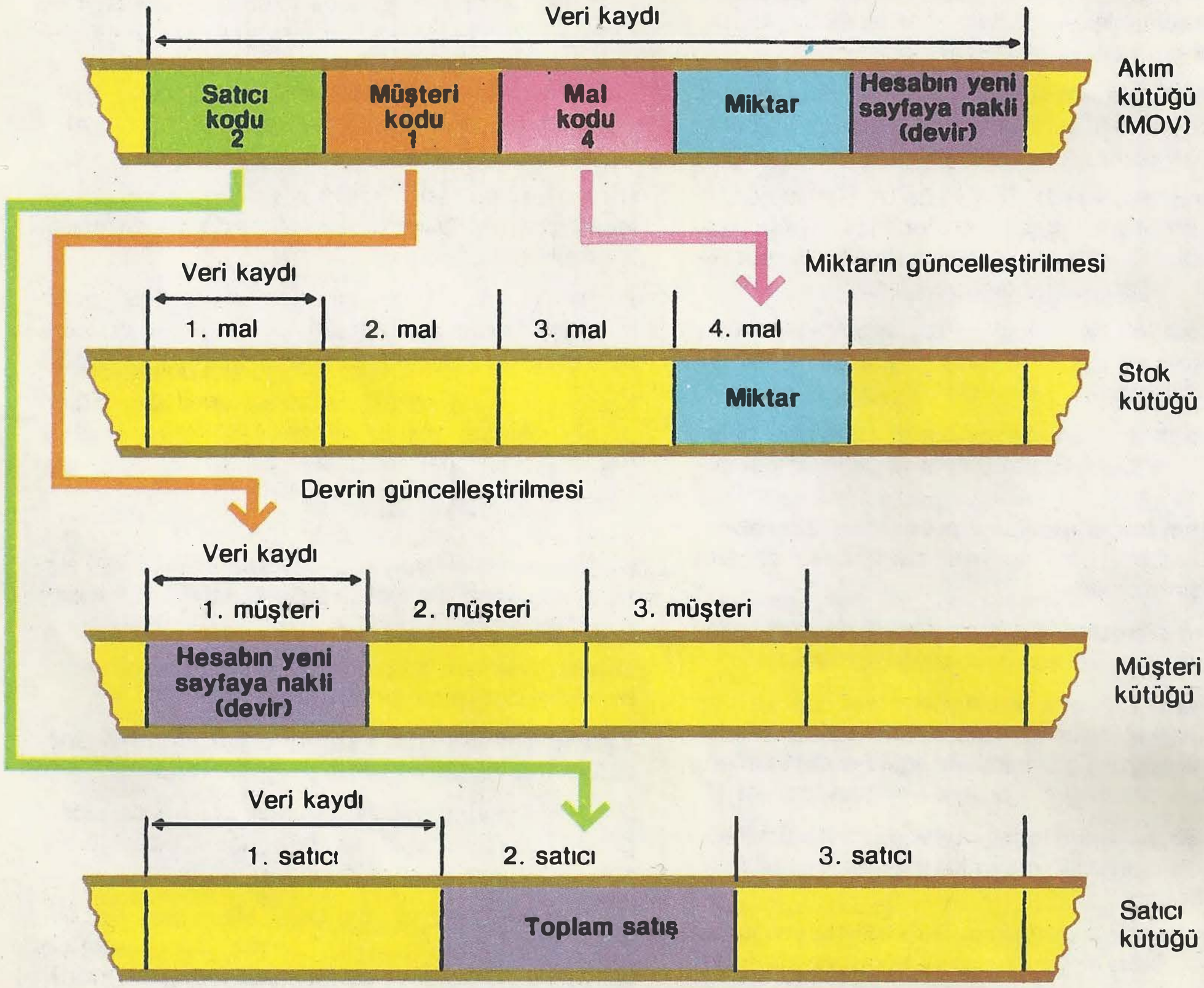
naklanır. Komut türlerine göre yapılan sınıflandırmada hatalarının yaklaşık % 30'unun DATA'lardan ya da başka veri yerleştirmelerden, % 20'sinin de koşullu komutlardan geldiği görülür.

Hataların araştırılması yapılırken anılan istatistik veriler aşağıdaki öncelik sırasına sokulabilir:

- Yerleştirme komutları (% 30)
- Komutların dizilişi (% 25)
- Koşullu komutlar (% 20)
- Ara birimler (% 15)
- Diğer (% 10)

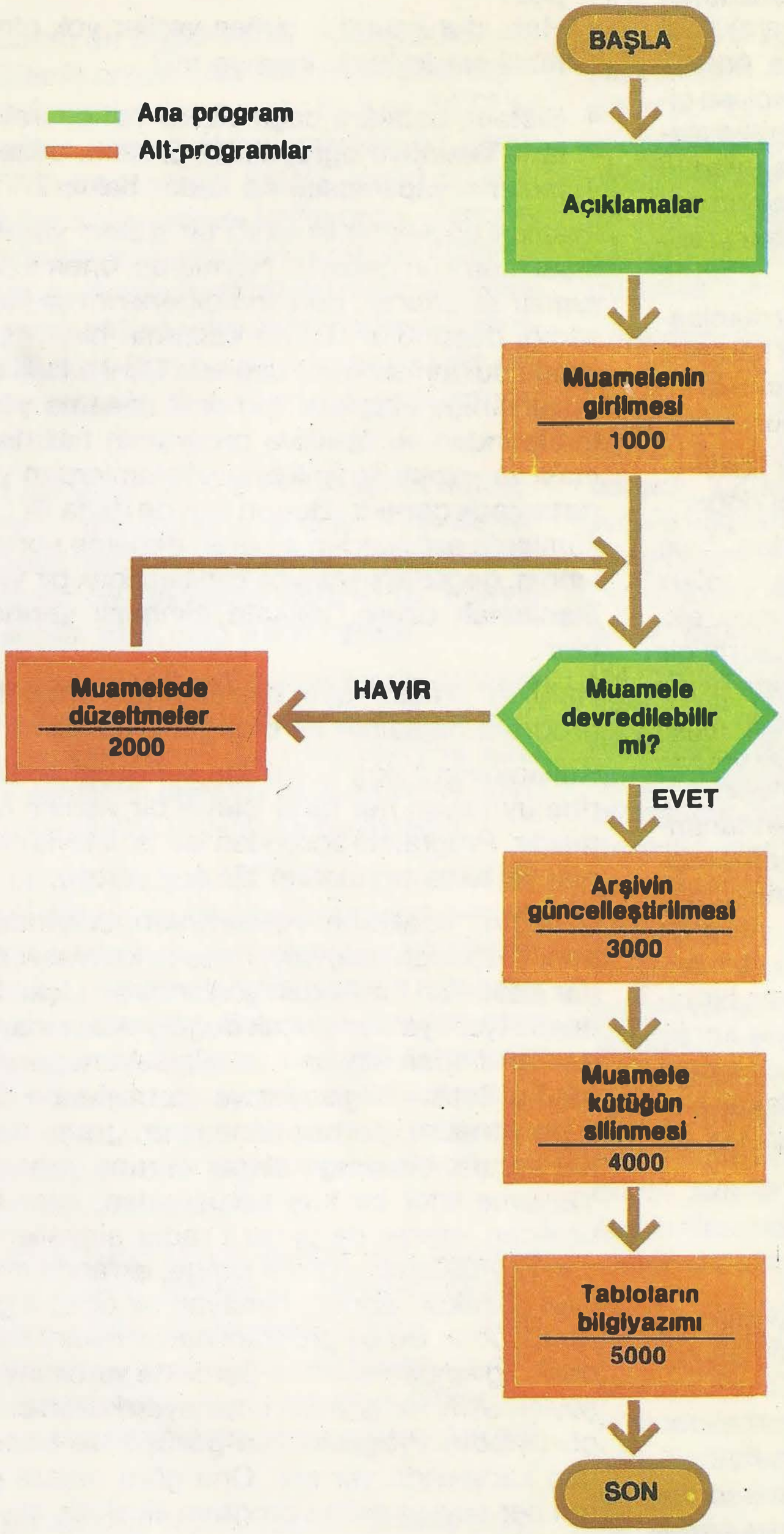
Yani hataların nedenleri programın yapısında aranmalıdır. Bunların yaklaşık yarısı da kolayca bulunabilecek hatalardır. Sorunların diğer yarısının ise nedenleri açık değildir ve yorum-

MAĞAZA YÖNETİMİ PROGRAMININ KÜTÜKLERİNİN YAPISI



MOV adlı kütükte bulunan satıcılara ait satış. Toplam satış kayıtları güncelleştirilir.

MAĞAZA YÖNETİMİ PROGRAMININ AKIŞ ŞEMASI



lanmaları bazen çok güçtür. Bu hatalar bilgi aktarımındaki muğlaklıklara indirgenebilecek olanlardır.

Bir programın üretilmesi, belli bir sonucun elde edilmesine yönelik eylemlerin makine tarafından anlaşılır bir dilde betimlenmesini ge-

rektirir. Çözümleme evresinde, elde edilecek sonuç ve bu sonuç için gerekli algoritmalar belirlenir. Bu evrede son kullanıcı, çözümleyiciye (ya da doğrudan programcıya) özel olarak sorunun içeriğini ilgilendiren önemli miktarda bilgi aktarmalıdır. Çözümleyici ise elle çalışacak şekilde betimlenen yöntemleri makinenin gereksinim ve kısıtlamalarına uyumlu kılmayı ve kullanıcıyı otomatik yöntemlerin getirdiği alternatif olanaklara kanalize etmeyi bilmelidir.

Burada, her ikisi de belli alanlarda uzmanlaşmış iki kişinin nesnel gereksinimlerini karşılıklı olarak birbirlerine aktarmaya çabalamalarından oluşan bir diyalog sözkonusudur.

Dolayısıyla, kısmen de olsa, klasik bir iletişimsizlik olayının ortaya çıkması olasılığı doğar.

Taraflardan her biri, farkına varmadan, bazı noktaların atanmasını kabullenerek, onları ayrıntılarıyla ortaya koymayabilir. Sonuç, eksik bir çözümleme ve ister istemez kusurlu bir program olacaktır. Bu takdirde deneme evresinde aksaklıklar tümüyle ya da kısmen ortaya çıkacak ve bu da önceden yeterince değerlendirilmemiş yeni çözümler geliştirilmesini zorunlu kılacaktır. Düzeltmeler, önceden tanımlanan ve bu işlevler için hazır olmayan programa yeni parçalar eklenmesiyle sonuçlanır. Böylece program özgün mimarisini yitirerek birbirine az ya da çok bağlı ve uyumlu bir bütünlüğü olmayan işlevleri yerine getiren bir birimler toplamına dönüşme tehlikesiyle karşılaşır. Bu kötü sonuçtan kurtulmanın tek çaresi ilk evreden itibaren şu üç aşamaya dayalı sistematik bir mantığa uygun olarak hareket etmektir:

- Gereksinimlerin dile getirilmesi
- Veri işleme
- Sınama

Program paketini tamamlayacak tüm işlemleri oluşturan aşamaların her birinde bu mantığın uygulanması gereklidir.

Programın Sınanması

Bir programı sınamak demek, onun olası tüm işlemsel koşullarda geçerli olup olmadığını sınamak demektir.

Sınamanın sonuçları, kullanıcı ile programcının önünde bulunan bazı temel gereklere yanıt niteliğindedir:

- Satın almak üzere olduğum ya da yazdığım program işliyor mu? İşliyorsa nereye kadar işliyor?
- Gelecekte üzerinde bir değişiklik yapılması mümkün olur mu? Bunun maliyeti ne olur?
- Hata durumunda, girilen veriler yok olma tehlikesiyle karşı karşıya mı?
- Sistem, hatalara bağlı olarak hangi aralıklarla kesintiye uğrayabilir? Sistem, düzeltmelerin yapılmasını ne kadar bekler?

Pratikte böylesine eksiksiz bir sistem yaratılması mümkün değildir. Normalde, önemli durumlar sınanarak, bunların diğerlerini de kapsadığı düşünülür. Böyle kestirme bir varsayımda bulunmak kolay değildir. Çünkü belli bir güvenilirliğe ulaşmak için özel deneme yöntemlerinden ve özellikle programın hazırlanması ve yazımı ile ilgili özel yöntemlerden yararlanmak gerekir. Bunun için de daha ilk çözümleme evresinden itibaren deneme sorunlarının, geçerliliği kolayca kanıtlanacak bir yapı planlamak üzere, dikkate alınması yerinde olur.

Hataları araştırmaya başlamadan önce bir program hatasının ne olduğuna bakalım.

Kullanıcının bakış açısından kendi beklentilerine uymayan her türlü işleyiş bir yazılım hatasıdır. Programcı açısından ise bu işleyiş normal ve hatta arzulanan bir şey olabilir.

Örneğin, uçakların havaalanları üzerindeki transit uçuşları, bilgisayara bağlı karmaşık radar sistemleri tarafından yönlendirilir. Uçan bir nesne (yani yalnızca uçak değil) yaklaşınca radar tarafından algılanır ve bilgisayara gerekli bilgiler iletilir. Bilgisayar ya da başka bir deyişle yönetim için hazırlanmış program, nesneyi grafik bir simge olarak ekrana yansıtır. Yansıma ister bir kuş sürüsünden, ister bir uçaktan isterse de parazit radar sinyallerinden kaynaklansın, grafik simge, ekranda mutlaka gözükür. Sonuç, olmayan bir uçağın görüntüsüdür. Bu bir program hatası mıdır? Kendisini ilgilendirmeyen ve gerçekte varolmayan şeyleri ekranda görmek istemeyen kullanıcı bu görüştedir. Programcının görüşü ise bunun tam karşısında yer alır. Ona göre radara giren her şeyi yansıtan program eksiksiz işliyor demektir.

En basit çözüm, programın gerekliliklere (gereklilik deyimiyle programın yerine getirmesi gereken işlevler ve bu işlevler için işlenen veri miktarı, işleyiş hızı vb. cinsinden sınırları anlatılmak istenir) uygun olarak çalışmadığı tüm durumları hata olarak görmektir. Bu, gerekliliklerin doğru olduğu varsayılması anlamına gelir. Oysa ki pek çok durumda hatalar, ge-

rekliliklerin yanlış ve eksik betimlenmesinden kaynaklanır.

Böylece, program hatalarının, ortadan kolayca kaldırılabilecek olanları bir yana bırakıldığında, programlamaya ait (içsel) olmayıp, çeşitli nedenlerin bir araya gelmesinden kaynaklandığı sonucuna ulaşılabilir. Bu nedenlerin başlıcaları gerekliliklerin hazırlanma biçiminde aranmalıdır. Bu yüzdendir ki, ''profesyonel'' bir ürün elde etmek için olanaklar ve sınırların kesin olarak bilinmesi gerekir.

Gerekliliklerin doğru ve titiz bir şekilde hazırlanmış olması da programı tümüyle hatalardan arındırmaz. Çok önemli ve dengelenmesi neredeyse olanaksız bir hata faktörü daha vardır ki, bu da insanın sınırlılığıdır. Kullanıcının, bilgisayarların işleyişi hakkında bilgisi olmayabilir. Bu durumda onu son derece betimsel ve kullanıcıya dönük programlarla yönlendirmek gerekir. Bu ise programın gelişkinliğini (karmaşıklığını) ve dolayısıyla hata olasılığını artırır. Bir kez daha, doğru bir orta yol arayışı gündeme gelir.

Her halükârda kullanıcı tarafından beklenmeyen bir olay bir hataya yol açabilir. Oysa, tüm olası hatalı eylemleri öngörerek her biri için bir denetim mekanizması getirmek olanaksızdır. Program, bu olaylar içinde ancak en tehlikeli olanları öngörebilir.

Programın tam olarak sınanması için gözönünde tutulması gereken ikinci parametre, güvenilirliktir. Bu deyimle programın belli bir zaman süresince hataya düşmeden çalışması anlatılır.

Yazılımın güvenilirliği zamanın bir fonksiyonudur. Başlangıçta düşüktür (sistem harekete geçtiği anda bir hata yoğunlaşması olur) ve zamanla düzeltmeler yapıldıkça yükselir. Donanımın güvenilirliği ise farklı bir süreç izler. Çok kısa bir süre için hata sözkonusu değildir. Uzun bir dönem boyunca arıza sayısı sınırlı kalır. Bu dönem geçince bileşkelerin eskimesinden dolayı güvenilirlik azalır.

786. sayfada her iki olgunun gidişini gösteren bir grafik yer almaktadır. Sistemin güvenilirliği her iki güvenliğe bağlıdır ve bir donanım arızası ya da bir program hatasından kaynaklanan ve birbirini izleyen iki durma arasında geçen zaman aralığı ile ölçülür. Örnekteki şemalarda program üzerinde yapılan düzeltmelerin başka hatalara yol açmadığı varsayılmaktadır. Gerçekte ise, programın iyi yapılandırılması durumu dışında, durum pek böyle olmaz. Normalde, yapılan düzeltmeler bir hata türünü ortadan kaldırırken bir başkasını doğururlar. Bu sorunla başedilmesi ancak programların çok çizgisel (düz) bir şekilde yapılması ve işleyişleri adım adım açıklayan çok sayıda açıklamayla donatılması ile mümkündür. İleride ayrıntılarıyla anlatılacak olan bu yapılanma biçimi iyi bir program bakımının temel önkoşuludur.

Bir programın durağan bir ürün olması son derece güçtür. Hemen her zaman değişiklik ve düzeltmelere uğrar. Ekonomik ve idari nitelikteki programlarda, örneğin yeni kurallara ilişkin değişiklik yapılması gerekecek, bilimsel olanlarda ise hesap logaritmaları ya da kullanılan parametreler değişecektir. Ekonomik bakımdan bu düzenlemelerin gerçekleştirilmesi, ancak program basit bir anlaşılırlık düzeyinde ise sözkonusu olabilir. Sayfa 787'deki grafikte 100 olarak kabul edilen programın toplam maliyeti üzerinde çeşitli işlemlerin etkisi görülmektedir: Bakım maliyeti toplamın % 50'si kadardır (büyük sistemlerde bu oran % 75'e varabilir) ve dolayısıyla sistemin ekonomikliğinde en önemli rolü oynar. Programın yanlış yapılanması, varolanın üzerinde değişiklikler yapıp durmaktansa, baştan başa yeniden yazılmasını pek çok durumda daha anlamlı kılabilir.

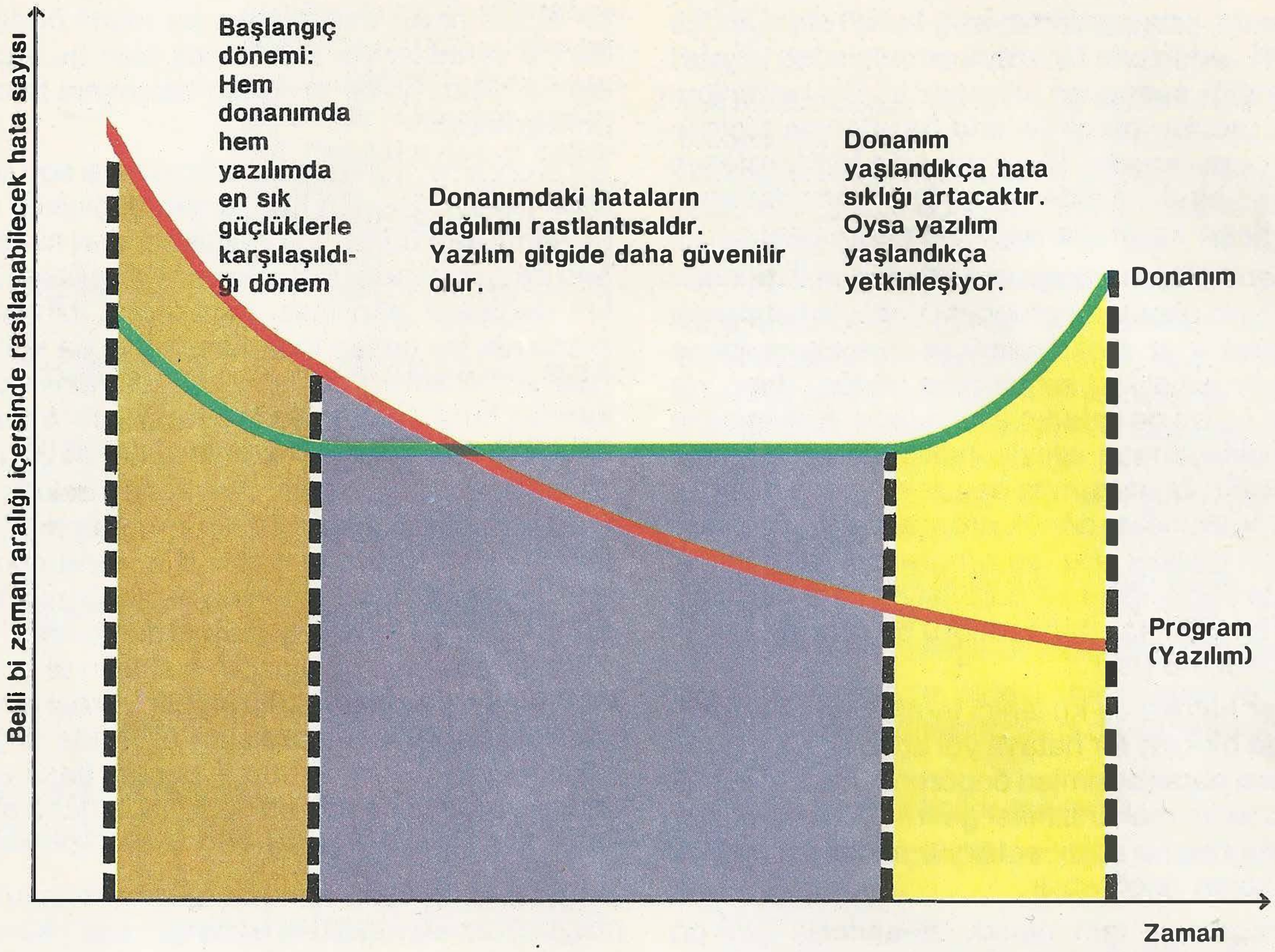
Sınama Yöntemleri

Tüm program üretim eylemleri içinde en hassas ve karmaşık olanı, sınama evresidir. Bunun başlıca iki nedeni vardır:

Öncelikle bu iş iyi bilinmemekte ve doğru ele alınmamaktadır. İkinci olarak bir sınamanın amacının çoğunlukla hata olmadığının kanıtlanması olduğu sanılmaktadır. Gerçekte bunun tam tersi olması gerekir. Bir sınamanın amacı, program hatalarını harekete geçirerek onları ortaya çıkarmak olmalıdır. Bu iki bakış açısı birbirinin aynı değildir ve hatta karşıt sonuçlara götürür. İlk yöntemde yanlışa yol açabilecek her türlü eylemden sakınılır. İkinci durumda ise program, özellikle yanlış kullanılarak, kullanıcının yapabileceği hataların bertaraf edilebilme derecesi ölçülür. Programın sınanması bir programı yazmakta kullanılan yöntemlere benzer biçimde planlanarak, adım adım yürütülebilir.

Bir başka deyimle, yapılacak işleri bir akış şeması halinde sıraya sokmak, her biri için kullanılacak verileri, ulaşılmak istenen sonuçları ve varılmak istenen ayrıntı düzeyini belirlemek doğru olur. Mütevazi boyutlarla da olsa, bir

BİR KİŞİSEL BİLGİSAYAR SİSTEMİNİN GÜVENİLİRLİĞİ



programın daima çok sayıda uzantısı vardır ve izlenen yol onun nasıl kullanıldığına bağlıdır.

Olası tüm yolların bir kez denenmesi genel olarak olanaksızdır. Örneğin TİTAN füzelerini yöneten programın tüm mantıksal seçeneklerini denetleyebilmek için hızlı (işleyen) bir bilgisayarın 60.000 saat (yani yaklaşık yedi makine yılı) boyunca çalışması gerekeceği tahmin edilmektedir. Hata, sayısal değişkenlerin aldığı değerlere bağlı olabilir. Bu durumda da tüm ikili olasılıkları denemek için yıllarla ölçülebilecek bir sürenin geçmesi gerekir. Bu güçlükler ancak son derece ileri yöntemlerin kullanılmasıyla aşılabilir. Kişisel ya da küçük bilgisayarlardaki deneme daha basit olarak belli başlı durumların, öncelikle kullanıcıyla sistem arasına konulan birimlerin sınanmasıyla gerçekleştirilir. Başlıca sınama teknikleri şöyle adlandırılır:

- Aşağıdan-yukarı yöntem
- Yukarıdan aşağı (alçalan) yöntem

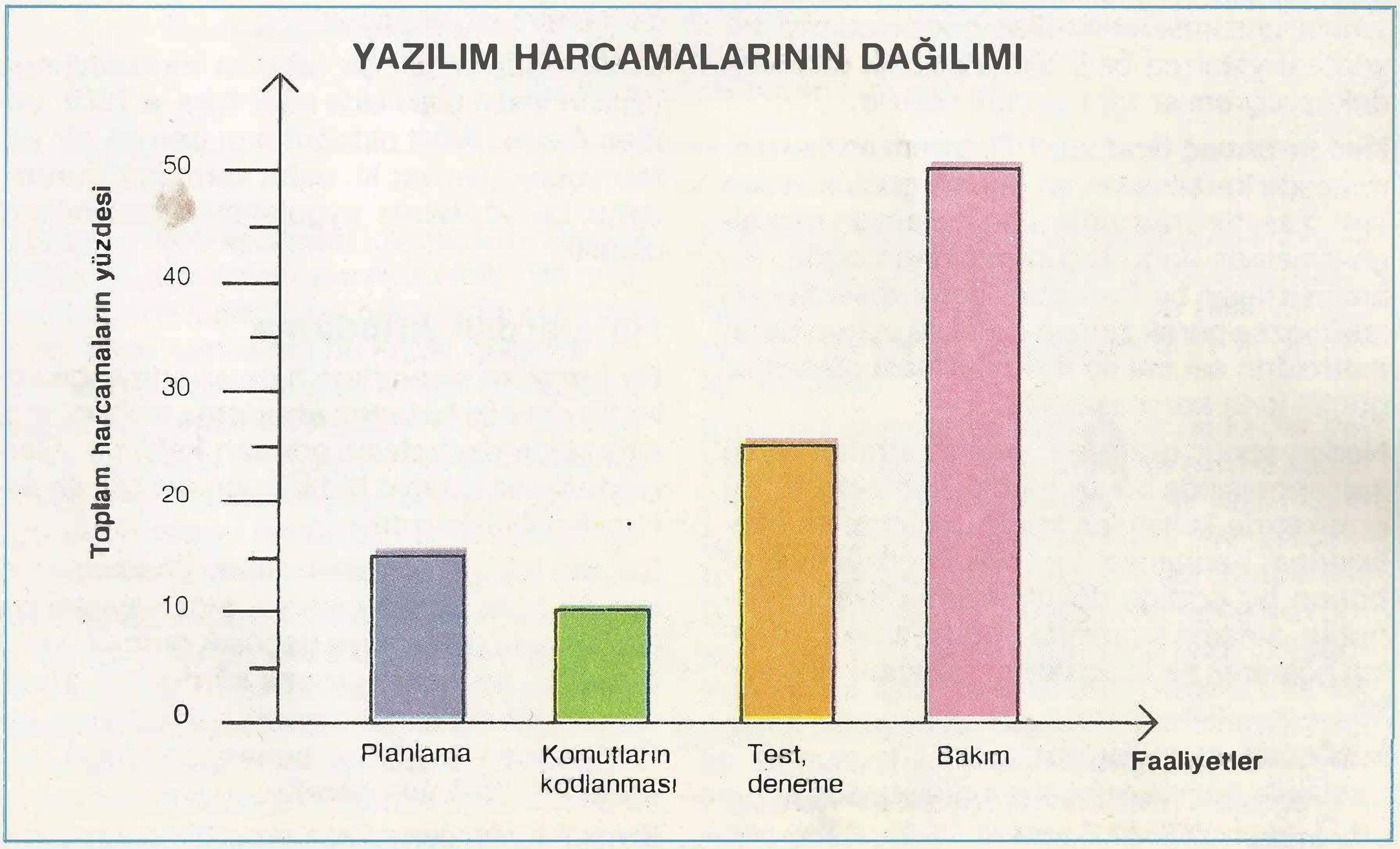
- Big-Bang yöntemi
- Sandviç yöntemi
- Neden-Sonuç grafikleri

Aşağıdan-Yukarı Yöntem (bottom-up). Normalde bir program, bir ana birimin (main) yaptığı çağrılarla (GOSUB) işlemeye başlayan bir birimler bütünüdür. Her durumda bir bitiş birimi vardır. Yöntem önce bu sonuç biriminin ayrı olarak sınanmasından sonra bir üst düzeyle bütünleştirilmesi ve giderek birinci düzeye erişilmesine dayalıdır. En üst noktaya varıldığında tüm program birleştirilmiş (bütünlenmiş) ve sınanmış olur.

Aşağıdan yukarı yöntem, incelenmekte olan program bölümlerini yöneten ve yalnızca denemede kullanılan hizmet birimlerini gerektirir. Örneğin, sınanan birim bir bilgiyazım altprogramı ise, sınanması için değişkenleri hazırlayacak bir birim gerekir. İkinci adımda doğru alt-programı birleştirirken onun için gerekli yeni verileri hazırlayan bir hizmet birimi kullanılacak ve bu böylece sürüp gidecektir.

''Pilotaj'' işlevini yerine getiren bu hizmet birimlerine **sürücü** (drivers) adı verilir.

Yukarıdan Aşağı Yöntem. Bir öncekinin karşıtıdır. İlk denetimi yapılan birim başlangıçta verilen ana birimdir (main). Onun ardından programın akış yönü izlenerek diğer birimler eklemlenir ve sınanır. Bu durumda da henüz



eklemlenmemiş birimlerin işlevlerini üstlenecek hizmet birimlerine gerek duyulur. ''Stub'' (kırıntı) olarak adlandırılan bu ikame birimlerinin gelişkinlik düzeyi, yerini aldıkları birimlere bağlıdır. Çoğu durumda kırıntılar yalnızca RETURN komutunu içerir ve birinin doğru biçimde çağrılmasına yarar. Başka durumlarda ise veri işleminin belirli bir kısmını kapsamalıdırlar. Yukarıdan aşağı yöntemin iki sakıncası vardır. İlk sakınca, kırıntıların eksik parçaları denetleyebilmek için çok gelişkin bir hal alma tehlikesi ile karşı karşıya olmalarıdır. İkinci sakınca ise en son tanımlanacak bölüm olan program ilk bölümünün, diğer birimlerde gerçekleştirilen değişikliklerden etkilenmesidir.

Oysa, bu teknik uygulandığında başlangıç bölümü denemeyi başarıyla atlatması gereken ilk bölümdür. Bu engel, çeşitli birimlerin birleştirilmeden önce ayrı ayrı sınanmasıyla aşılabilir. Ancak bu da gerekli sürücü ve kırıntıların yazılmasını gerektirir. Gerçekten de ayrılan birimin, girişleri hazırlayacak sürücüye ve çıkışları denetleyecek kırıntıya gereksinimi vardır.

Big-Bang Yöntemi. Çeşitli birimlerin ayrı ayrı sınandıktan sonra, birleştirilmesi ve ardından genel bir deneme yapılmasıdır. Bu yöntem yalnızca küçük boyutlu programlar üzerinde uygulanabilir. Diğer durumlarda ise aşılması olanaksız güçlükler yaratır.

Sandviç Yöntemi. Yukarıdan aşağı ve aşağıdan yukarı yöntemlerin bir arada kullanılmasıdır. Böylelikle bazı sürücü ve kırıntıların yazılması gerekmeyebilir. Bazı güçlükleri olan bu yöntem yalnızca belli bir gelişkinlik düzeyindeki programlar için geçerli olabilir.

Neden-Sonuç Grafikleri. Programların sınanmasında karşılaşılan en önemli güçlüklerden biri, olası tüm durumlar için kullanılan tüm algoritmaların doğruluğunun sınanmasıdır. Bu konuya kesin bir metedoloji çerçevesinde yaklaşılmazsa gerek zaman gerekse uygun parametrelerin seçimi ile ilgili, aşılması olanaksız güçlüklerle karşılaşılabilir.

Neden-sonuç grafikleri, sınama stratejilerinin saptanmasında büyük yardım sağlayabilir. Bu grafiklerde kullanılan teknik, programın niteliklerinin ''anlamsal'' içeriğinin çözülmesi ve bunun bir grafiğe dökülmesine dayanır. Sonuçta, sınama sırasında araştırılacak durumları gösteren bir tablo **(karar tablosu)** elde edilir.

İzlenecek aşamalar şunlardır:

— Programın niteliklerinin tek tek işlevlere ayrılması
— Her olası neden (giriş) ve her olası sonucun (çıkış) saptanması için işlevlerden her birinin çözümlenmesi
— Grafiğin çizilmesi
— Grafiğin bir karar tablosuna dönüştürülmesi

Neden-sonuç ilişkileri grafiği, olası tüm basit ilişkileri betimleyen belirli bir simgeler bütünü yardımıyla çizilir. Karşı sayfada üstte, belli başlı simgeler görülüyor.

Yöntemi örneklemek için bir karakter zincirinin ilk karakterini büyük harfe çevirecek bir program oluşturmak istediğimizi varsayalım.

Böyle bir durumda olası nedenler şunlardır:

a- ilk karakter bir sayıdır
b- ilk karakter bir büyük harftir
c- ilk karakter bir küçük harftir
d- ilk karakter ne bir sayı ne bir harftir

Olası sonuçlar ise şunlardır:

x karakter zinciri aynı kalır (neden a)
x karakter zinciri aynı kalır (neden b)
y ilk karakter büyük harfe çevrilir (neden c)
z bir hata mesajı verilir (neden d)

Ayrıca,

1/ bir hata varsa programın işleyişi durur
2/ bir hata yoksa program sürer

Karşı sayfada alttaki diyagram eldeki simgelerle sol tarafa neden, sağ tarafa sonuç sıralanarak oluşturulur. Karar tablosu ise her sonucun gözönüne alınıp olası tüm nedenlerin işaretlenmesiyle yapılır.

Grafik ifadeler tek bir tabloda toplandığında yapılan testin planı elde edilir (bkz. s.790). Verilen örnek, basit olduğundan gerçek bir yararı yoktur. Ne var ki, daha karmaşık durumlarda bu yöntemin uygulanması kaçınılmaz olabilir.

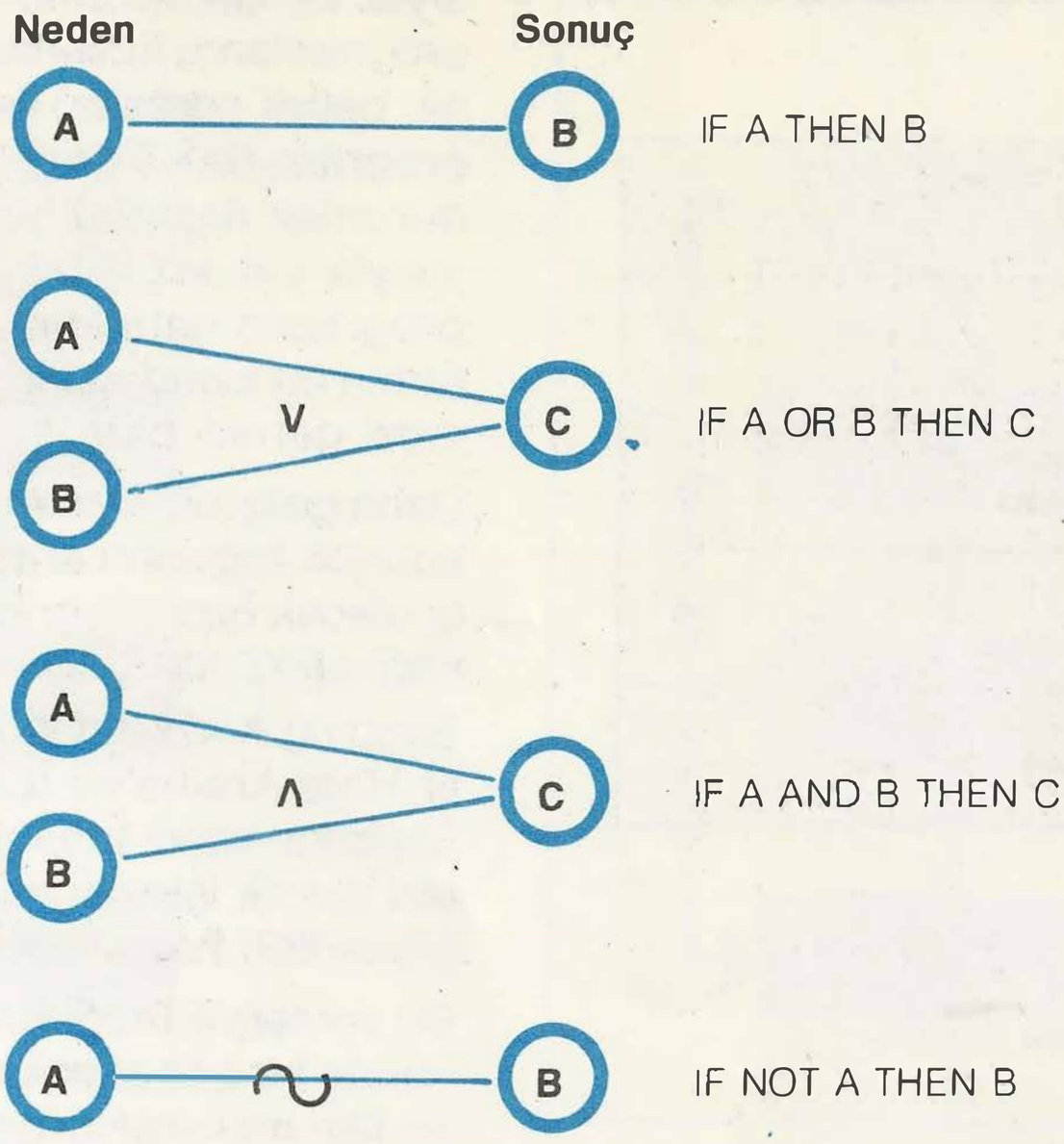
Hatalardan Arındırma

Bir program sınanırken hatalar ortaya çıkarılır; bir sonraki **hatadan arındırma** (debugging) evresinde ise hatalar ortadan kaldırılır. Gerçekte bu iki işlemin birbirinden net olarak ayrılması olanaksızdır.

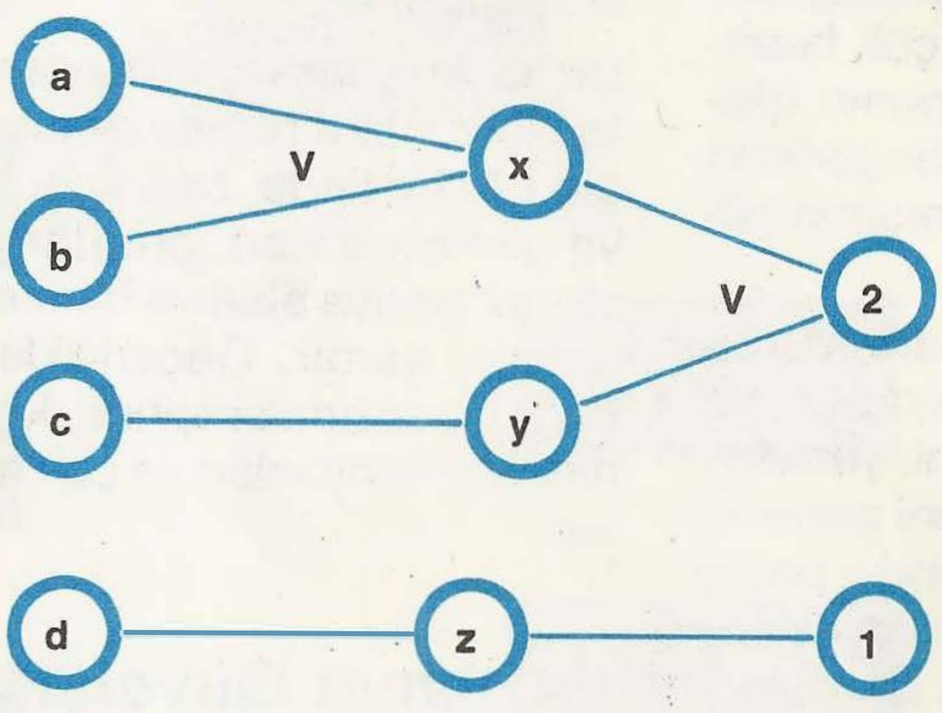
Seçilen teknik ne olursa olsun, sınama sürerken bir hata ortaya çıkarsa, sınamayı askıya alarak hata gidermeye geçmek gerekir. Hata düzeltildikten sonra sınama sürdürülür. Hatadan arındırma işinin de belli bir yönteme göre, önceden belirlenen bir mantığa bağlı kalınarak yürütülmesi gerekir.

En çok kullanılan çözümleme türlerinden biri

YAZILIM HARCAMALARININ GÜNCELLEŞTİRİLMESİ



TOPLAM HARCAMALARIN YÜZDESİ



şu üç ana parametreden oluşan bir tablonun çizimine dayalı ''yöntem'' (''the method'')'dir:

— Ne?: Hatanın temelinde yatan aksaklığın tanımlanması
— Nerede?: Hatanın bulunduğu program alanının betimlenmesi
— Ne zaman?: Hatanın gerçekleştiği koşulların betimlenmesi

Yöntemde yalnızca ileri sistemlerde ender olarak kullanılan dördüncü bir parametre (hatanın arazı) de öngörülmüştür. Hata nedenlerinin araştırılmasında en büyük yol gösterici, bu soruların yanıtlarının verilmesidir.

NEDEN-SONUÇ TABLOSU

Neden	Sonuç				
	x	y	z	1	2
a	Sİ	—	—	—	Sİ
b	Sİ	—	—	—	Sİ
c	—	Sİ	—	—	Sİ
d	—	—	Sİ	Sİ	—

Sİ = Test başarılı
— = Gözönüne alınmama durumu

BASIC'de, bu dili anında kullanma olanağı bulunduğundan hata giderme işlemi çok basittir. Bir hata ortaya çıktığında programın işleyişi durdurulabilir ve olası normal değerlerin bulunabilmesi için tüm ilgili değişkenlerin bilgiyazımı istenebilir.

Aynı işlem yalnızca derleyicisi olan dillerde biçimlenmekte olan programın (bilgiyazım komutları eklemek üzere) düzeltilmesini, yeniden derlenmesini ve yeniden işletilmesini gerektirir. İşte ancak bu ikinci işleyiş sırasında, programın hata içeren bölümünde kullanılan değişkenlerin sayısal değerleri elde edilir. Daha karmaşık sorunlarda hatayı ortaya çıkarabilmek için bu işlemlerin birkaç kez yinelenmesi gerekli olur. Bununla birlikte, (çeviricili) BASIC'de çok fazla zaman yitirilmediği halde, diğer dillerde bu iş çok pahalıya malolabilir. Bu nedenle hataların araştırılmasında yardımcı olarak kullanılan programlar yapılmıştır.

Bu program türü **(hata arama araçları)**, incelenen programda kullanılan belleklerin içeriğinin araştırılmasını sağlar, yani değişkenlerin aldığı değerler bilgiyazım komutlarını yazmaya gerek kalmadan ekrana yansıtılır. Bu programın daha ileri biçimleriyle, belleklerin içeriğinin de değiştirilmesi mümkün olur. Bu yöntemde, hatanın nedenini bulup çıkarana dek girişimler sürdürülür. Tekniğin aksayan yönü Assembler dilini bilme zorunluluğudur.

Derlenen programlar iki tabanlı şekilde belleğe yüklenir. Her komut, koduyla ifade edilir. Öyle ki, değişkenler, ortaya çıkmakta olan programlarda kullanılan simgesel adlarıyla değil, bellek adresleri ile tanımlanır. Hata arama programı (büyük makinelerde uygulanan özel durumlar dışında) hep aynı biçimdedir. Dolayısıyla yüksek düzeyde bir dille yazılmış bir programın hatalarını arayabilmek için, programın en azından kısmen Assembler'e çevrilmesi gerekli olur.

Daha gelişmiş sistemlerde kullanıcıyla diyalog kurarak hataların araştırılmasında ona yardımcı olacak özel programlar öngörülmüştür. Örneğin EXDAMS sisteminde (Extandable Debugging And Monitoring System-Genişletilebilir Hata Arama ve İzleme Sistemi) programın üzerinde kaydı tutulur. Kullanıcı, bilgileri banttan alarak işleyişi baştan inceleyebilir ve değişkenleri her an görüntüleyebilir.

Bu yöntemle programa, akış yönünün tersi yönünde bir göz atmak da mümkündür (program bir film makarasını geri çevirir gibi geriye döndürülebilir). Fakat inceleme, programın bitişinden sonra yapıldığından herhangi bir değişiklik yapılamaz.

Bir başka yöntem, kullanılan dilde varolan bazı fonksiyonlara başvurulmasına dayanır. Örneğin PL/1 dilinde, belli koşullar altında bir araya girme yaratan, programın işleyişini geçici olarak askıya alan ve bir mesaj veren CHECK komutu vardır. Geçerlilikleri kullanılan dilde bağlı bu çözümler içinde, kişisel bilgisayar alanında kullanılanları ya çok azdır ya da hiç yoktur.

Yazılımın Güvenliği

Veri sayısı arttıkça verilerin korunmasını sağlama sorunu önem kazanır. Büyük sistemlerde çok gelişmiş ve özel donanım koşullarına bağlı koruma mekanizmaları mevcuttur. Kişisel bilgisayarlarda ise çözümün uygulama programında öngörülmüş olması gereklidir.

Verilerin yitirilmesinin başlıca nedenleri, donanım bozuklukları ile işlemcinin yaptığı hatalarda aranmalıdır.

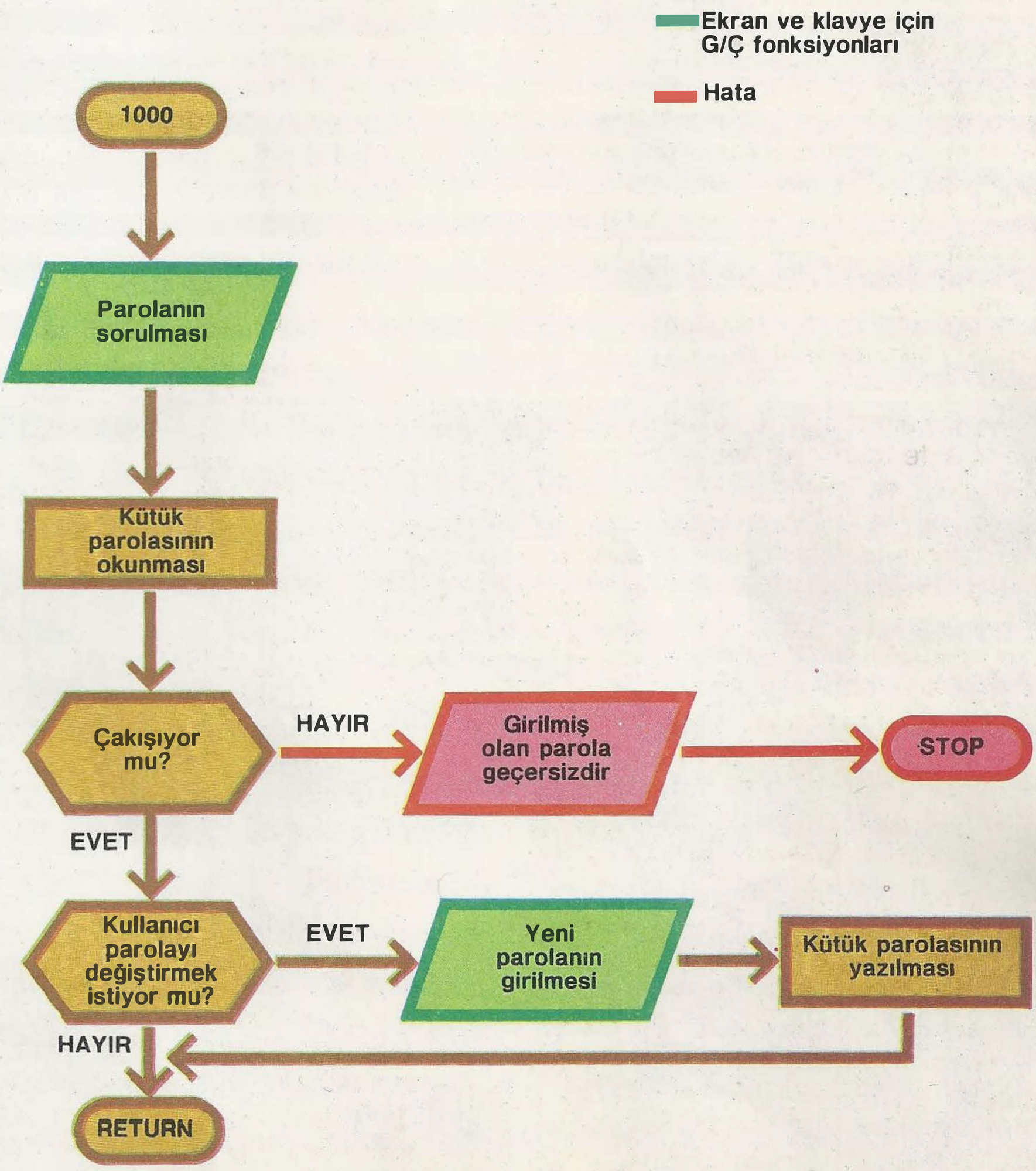
Donanım bozukluklarından kaçınmanın yolu veri içeren disketlerden sürekli birer kopya çıkarmaktan geçer. Uygulama programında, son çıkarılan kopyanın tarihi çok eskidiğinden, işlemciyi yeni ve güncel bir kopya çıkarması konusunda uyaran bir denetim öngörülebilir.

İşlemcinin hata yapmasını önlemeye yönelik

sistemler daha karmaşıktır ve her zaman etkili olmazlar. Programın belli noktalarının ancak parolayı bilen yetkili kişilerce kullanılması sağlanarak birinci güvence düzeyine ulaşılır. Bundan sonraki düzeylerde ise her silme ya da değişiklik istendiğinde önce onay isteminde bulunulması öngörülür. Onay istemiyle kullanıcının dikkati çekilir; o da hata durumunda işlemeyi durdurabilir.

791. sayfada parolayı denetleyen bir alt-programın basit akış şeması verilmektedir. Aşağıda ise onun kullanıldığı bir program dökümü verilmektedir. Satır 200'ün ekrana yansıma-

PAROLANIN KONTROLÜ VE DEĞİŞTİRİLMESİ İÇİN AKIŞ ŞEMASI



yan INPUT$ fonksiyonuyla girildiğine dikkat ediniz. Böylelikle olası bir gözlemcinin girişi yapılan parolayı okuması mümkün olamaz. Alt-program derlendiğinde tam bir geçerliliğe sahiptir. Aksi halde, program dökümünden parolayı içeren kütüğün adını okumak yeterlidir; böylece kütüğün içeriğini okumak ve değiştirmek ve bu arada denetimlerden kaçınmak mümkündür. Gerekli değişiklikler yapılarak uzman kişilerin dahi bunlara ulaşması önlenebilir. Böylelikle, kullanılan maskenin değerini bilenler dışında okuyanlar herhangi bir şey anlamazlar.. Derleyicide bulunan bu değerin okunması olanaksızdır.



BİR PAROLANIN SINANMASI İÇİN PROGRAM

```
10 '****DENEME ANA PROGRAMI****
12 INPUT "KUTUK SIFIRLANMALI MI";CEV$
14 IF CEV$<>"EVET" GOTO 95
20 OPEN "R",1,"A:WW",5
30 FIELD 1,5 AS W$
40 A$=SPACE$(5)
50 LSET W$=A$
55 ' ***ILK DOKUZ VERI KAYDI SIFIRLAMA ICIN SILINIYOR
60 FOR R%=1 TO 9
70 PUT 1,R%
80 NEXT R%
90 CLOSE 1
95 GOSUB 1000
96 END
1000 '**PAROLANIN SINANMASI ICIN ALT-PROGRAM**
1010 '     KUTUK=KELIME
1020 K=0
1030 PRINT "KARSILIK GELEN KODU GIRINIZ"
1040 ' HER KULLANICI, KENDI PAROLASININ BULUNDUGU
1050 ' VERI KAYDINI GOSTEREN BIR
1060 ' NUMARAYA SAHIPTIR
1070 A$=INPUT$(2)
1080 IF VAL(A$)=0 OR VAL(A$)>10 GOTO 1290
1090 'SIFIRLANMIS KODLAR 01 DEN 10 KADARDIR
1100 OPEN "R",1,"A:WW",5      'WW, HERBIRI BES KARAKTER UZUNLUGUNDA
1110 '                         PAROLALARDAN OLUSAN PAROLA KUTUGUDUR
1120 FIELD 1,5 AS W$
1130 R%=VAL(A$)   'KULLANICININ KULLANACAGI VERI KAYDININ NUMARASI
1140 B$=SPACE$(5)
1150 GET 1,R%     'DISKETTE BULUNAN PAROLANIN OKUNMASI
1160 LSET B$=W$
1170 IF B$=SPACE$(5) GOTO 1240  BASLANGICTA KUTUK BOSTUR VE
1180 '                          PAROLANIN YAZILMASI GEREKIR
1190 PRINT "PAROLAYI GIRINIZ"
1200 A$=INPUT$(5)   'YANKISIZ GIRIS
1210 IF A$<>B$ GOTO 1290
1220 INPUT "PAROLA DEGISTIRILECEK MI":CEV$
1230 IF CEV$<>"EVET" GOTO 1280
1240 PRINT "YENI PAROLAYI GIRINIZ"
1250 A$=INPUT$(5)
1260 LSET W$=A$
1270 PUT 1,R%       'DISKETE YAZMA
1280 CLOSE 1 : RETURN
1290 PRINT "**KABUL EDILMEYEN PAROLA**"
1300 IF K=1 THEN STOP
1310 PRINT "YENI BIR DENEMEDEN SONRA PROGRAM DURUR"
1320 K=1:GOTO 1150
1330 '********************
```

Genel Uygulama Programları

Bilgisayarların iş dünyasındaki kullanımı sanıldığı kadar yaygın değildir. Bilgisayardan en çok yararlanılan görevlerin dikkatli bir analizi sonucunda, bunlar arasında ikisinin öne çıktığı görülür:

— Arşiv tutulması (yönetimi)
— Veri işleme

Bu görevler farklı uygulamalarda değişik biçimlerde, sık sık karşımıza çıkar. Ancak, veriler ve hesap işlemleri ne kadar değişirse değişsin, programın mantığı ve dolayısıyla temel yapısı hep aynı kalır.

Örneğin telefon fihristi tutmaya yönelik bir programda görülmesi gereken başlıca işler, verileri girme, düzeltme, disket üzerinde belleğe alma ve (gereğinde) arayıp bulmadır. Bu işler daha önceden anlatılan birimlerin (veri girişi, disket kullanımı vb.) doğru bir şekilde kullanılmasıyla gerçekleştirilebilir. Aynı programın bir kütüphane yönetimine uyarlanabilmesi için yalnızca verilerin yapısının değiştirilmesi yeterlidir. Bunun dışında yapılması istenen bütün işler aynıdır ve dolayısıyla aynı birimlerin kullanılması mümkündür.

Bu düşünceler, programlama konusunda çok az bilgisi olan kişiler tarafından da kullanılabilecek ve veri işleme ile arşivleme gibi iki ana işlevi yerine getiren bazı hazır programların geliştirilmesine yol açmıştır. Bu tür yazılımların gelişimi, tüm bu genel işleri yapabildiği gibi, grafiklerle ilgili özel fonksiyonlar da içeren eksiksiz programların yapılmasıyla sonuçlanmıştır.

Yerleşim Haritası Tekniği

Esas olarak veri işleme işlevi gören programların (örneğin Visicalc, Multiplan, Lotus) yapısı hep aynıdır. Makine belleğinin, satır ve sütunlara ayrılmış bir büyük kara tahta olarak kullanılmasına olanak veren bu yapıya **yerleşim haritası** adı verilir.

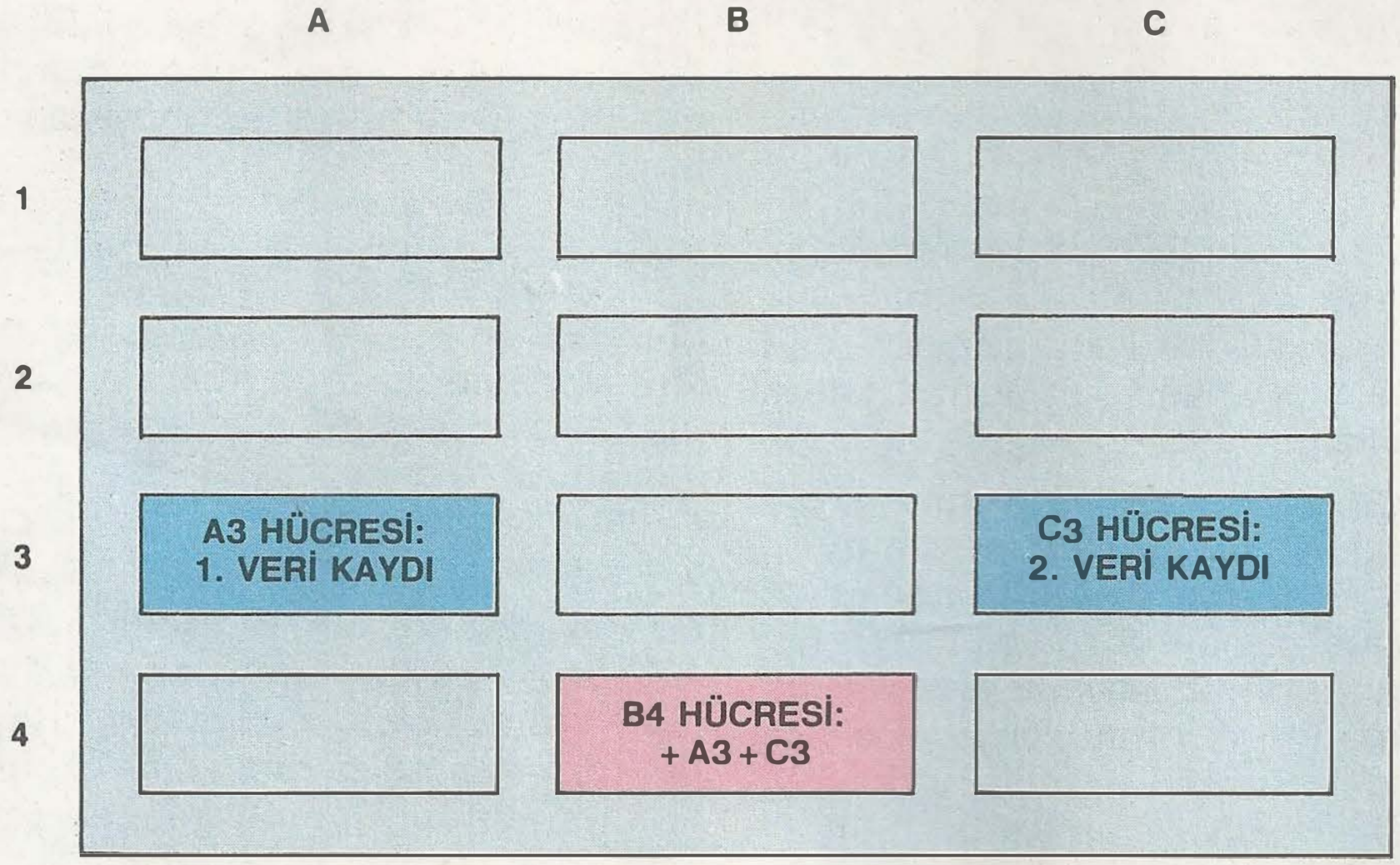
Bir satırla bir sütunun keşismesiyle içine (sayısal ya da karakter zinciri) bir veri ya da bir formül yerleştirilebilen bir **hücre** ortaya çıkar. Hücrelerin içeriği her an değiştirilebilir. Hücrenin içinde bir formül varsa, sonuç anında hesaplanır.

Bu fonksiyondan ve aşağıda ayrıntılarıyla an-

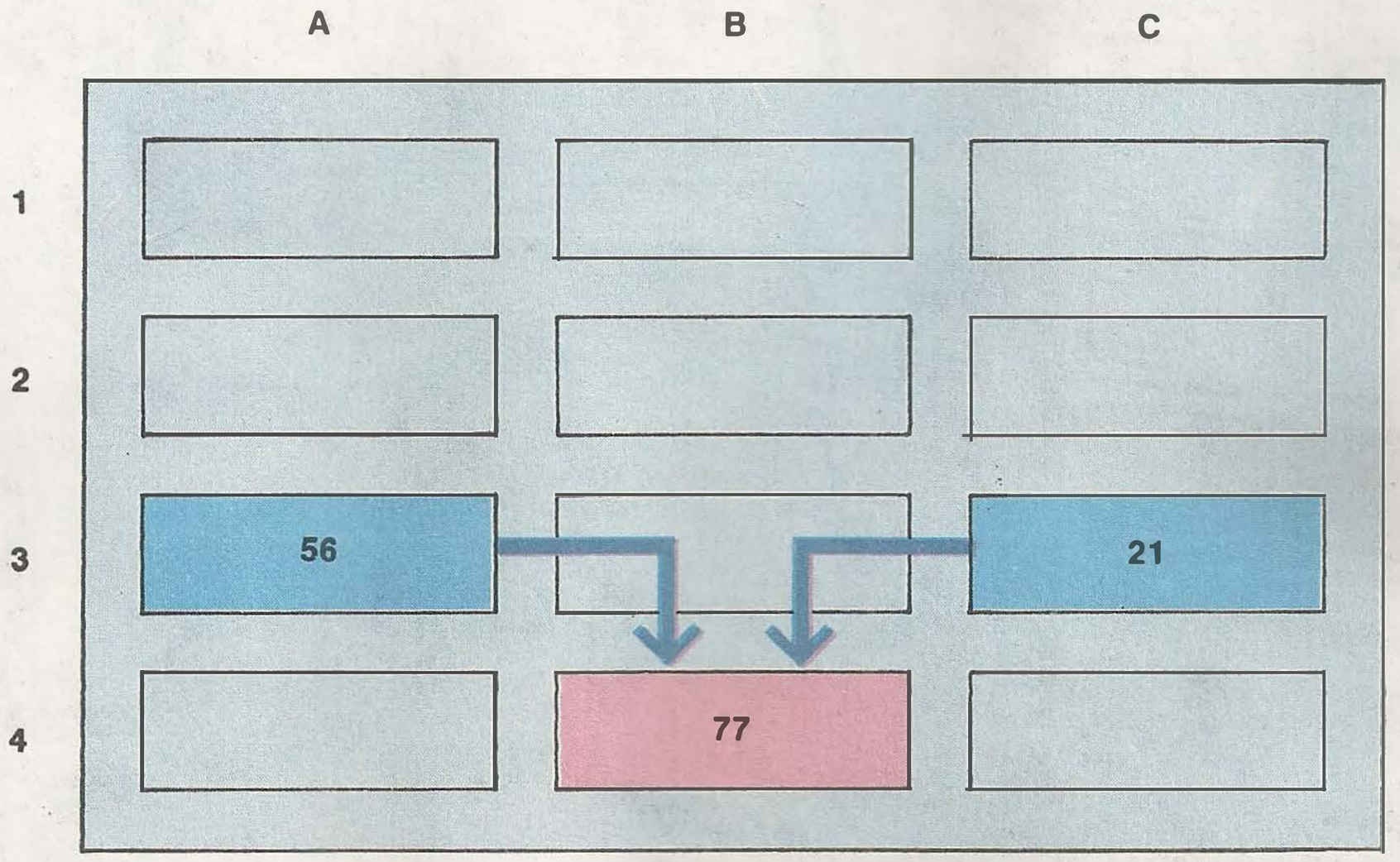
1980 Moskova Olimpiyatları'nda bilgisayar kullanımı



YERLEŞİM HARİTASINDA BİR TOPLAMANIN YAPILIŞI



B4 hücresi, yürütülecek hesaplama işlemi ile ilgili kaydı içerir:
+A3 +C3'ün anlamı: ''A3 ve C3 hücrelerinde verilerin toplamını göster.'' A3 ve C3 hücrelerine iki sayı verildiği takdirde, istenen toplamın hesaplanması ve sunulması B4 hücresinde otomatik olarak gerçekleşir.



latılacak olan diğerlerinden yararlanarak bir hesap yapmak ve bu hesabı, yeni veriler ekleyerek tekrarlamak mümkündür. Bunun için kullanıcının bir ön bilgiye sahip olması gerekli değildir. Ona düşen yalnızca, çeşitli işlemleri, tıpkı elle yapar gibi yerlerine koymaktır. Program bu işlemlerin gereği olan hesaplamaları yapacaktır. Karşı sayfada, her sütunun bir harfle ve her satırın da bir sayıyla gösterildiği bir yerleşim haritası şeması görülüyor. Buna göre B4, B sütunu ile 4. satırın kesişme yerindeki hücre olmaktadır. Bir hücrenin kapasitesi normalde 8-9 karakter olmakla birlikte, bu sayının, uygun güdümler yardımıyla değiştirilmesi mümkündür.

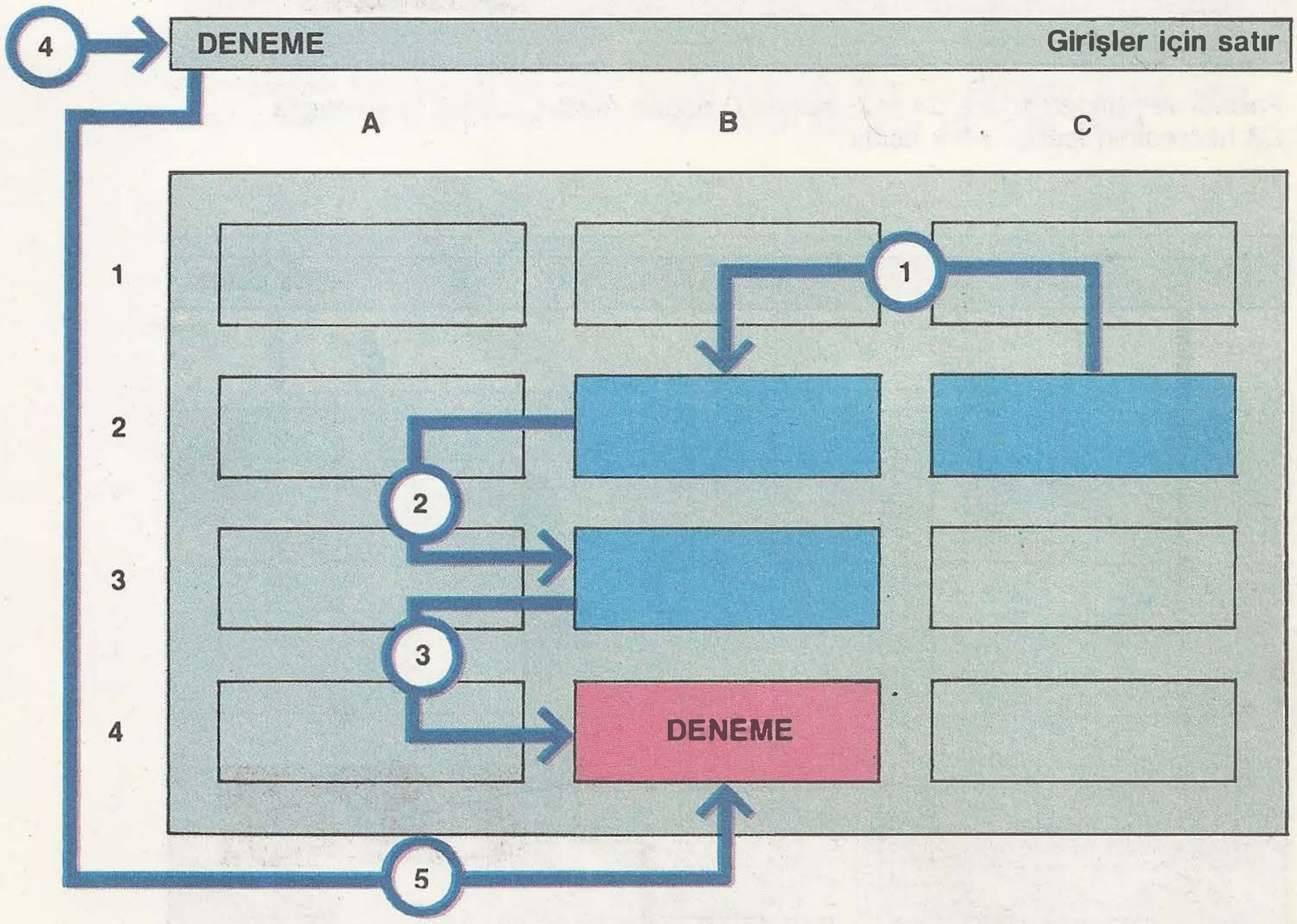
Sözü edilen şemada bir hücrenin başlıca iki kullanımı gösterilmektedir: Verilerin ve formüllerin belleğe alınması. İşlenecek verileri içermesi gereken A3 ve C3 hücreleri başlangıçta boştur. B4 hücresine ise işlemin formülü (örneğin +A3+C3 yani A3 ve C3'ün içeriklerinin toplamı) yazılır. Program, B4'te bulunan formülün farkına varınca hücrenin içeriğini (+A3+C3 biçiminde) ekrana hiç yansıtmadan, verilere uygulanan formülün sonucunu görüntüler. Başlangıçta B4 hücresi 0 değerini içermekteydi, çünkü henüz daha bir değer verilmemişti. Ancak işlem yapılır yapılmaz, B4'ün değeri A3 ve C3'ün değerlerine göre kendiliğinden uyarlanır.

Satır ve sütunların sayısı her hücrenin ve her ekranın büyüklüğüne göre değişir. Normal olarak ekranda 80 sütun ve 24 satır bulunur.

Bazı modellerde ise yerleşim haritasının payı bunların yarısıdır ve 40 sütun bulunur. Örneğin, ekran üzerinde görünen kısmın sağında kalan hücreleri görüntüleyebilmek için, imleci ekranın sağ yanından dışarı çıkarmak istercesine sağa kaydırmak yeterlidir. Böylece sol baştaki sütunlar (A, B, vb.) kaybolacak ve sağda yeni sütunlar ortaya çıkacaktır. Aynı yolla tüm yerleşim haritasını dikey yönde hareket ettirmek de mümkündür. Yerleşim haritasının bu intibak özelliğine **scroll** adı verilmektedir.

Yerleşim haritasının kullanıldığı programlarda

BİR VERİ KAYDININ BİR HÜCREYE GİRİŞİ



1 İmlecin başlangıç pozisyonu (C2)
İlk kaydırma: yatay yerleştirme (← tuşu)
2-3 Yatay yerleştirmeden sonra imleç, ardarda iki dikey kaydırma (Boşluk + ← , Boşluk + ←) ile giriş yerine (B4) götürülür.
4 Veri kaydı
5 İmleç doğru yere getirildikten sonra, her veri kaydı otomatik olarak gösterilen hücreye aktarılır.

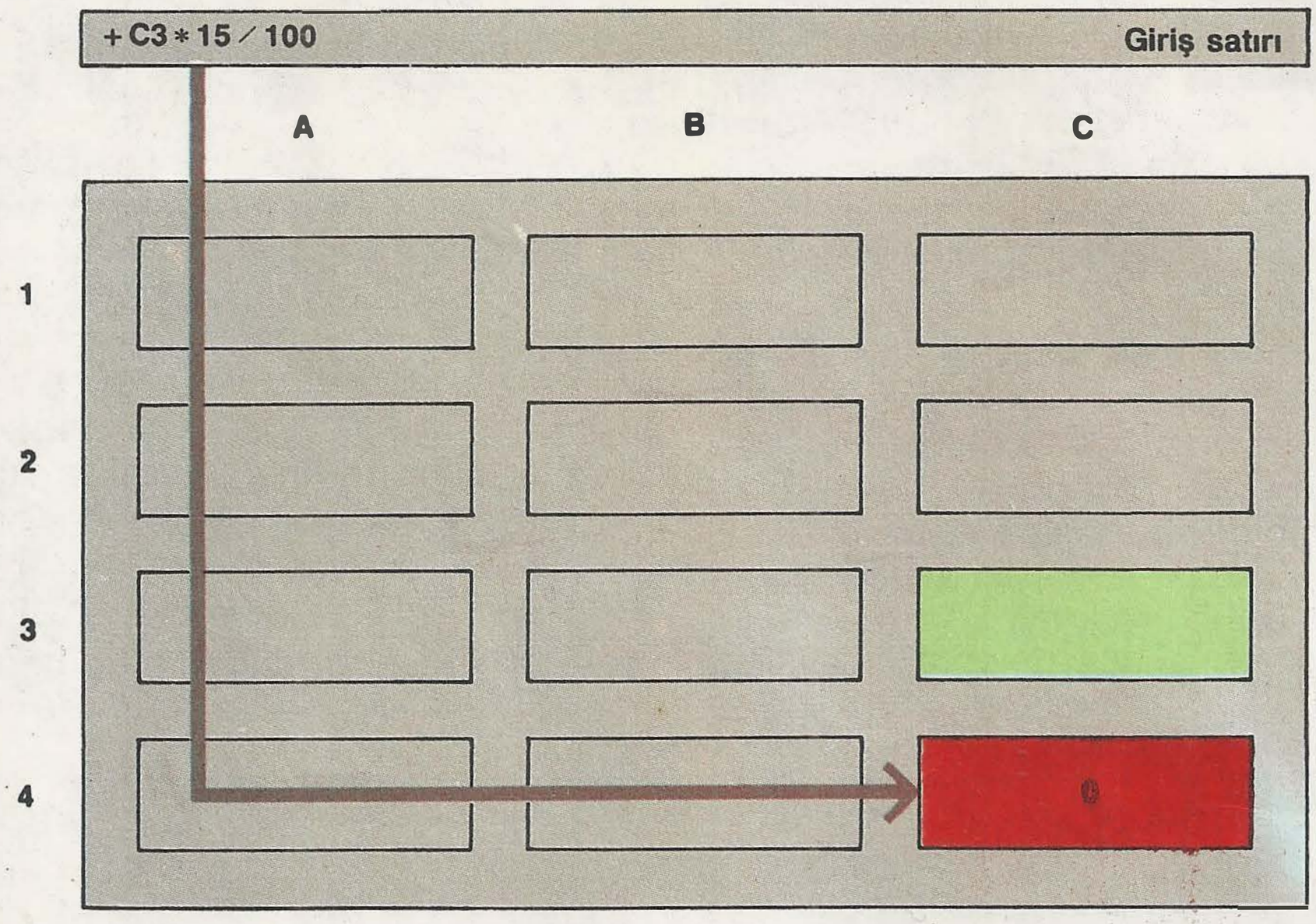
BİR FORMÜLÜN GİRİLMESİ

Veri kaydının hücresi

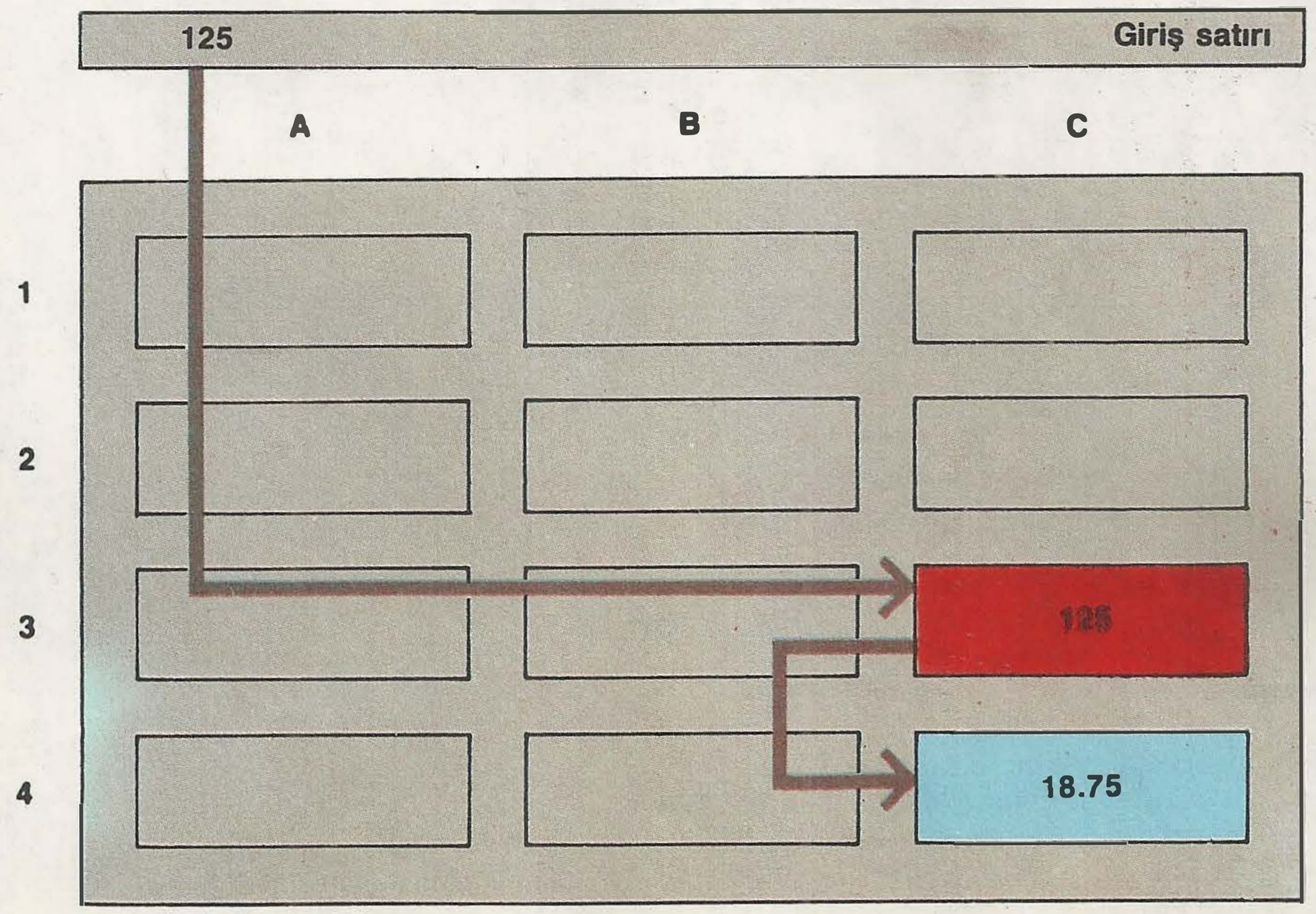
İmlecin bulunduğu hücre

Formülün hücresi

DOĞRU



Formül verildikten sonra C4 hücresinde 0 değeri oluşur, çünkü başlangıçta C3 hücresinin içeriği sıfıra eşittir.



C3 hücresine bir değer verildiği takdirde, C4 hücresi sonucu görünür kılacaktır. Bu, kendiliğinden gerçekleşir.

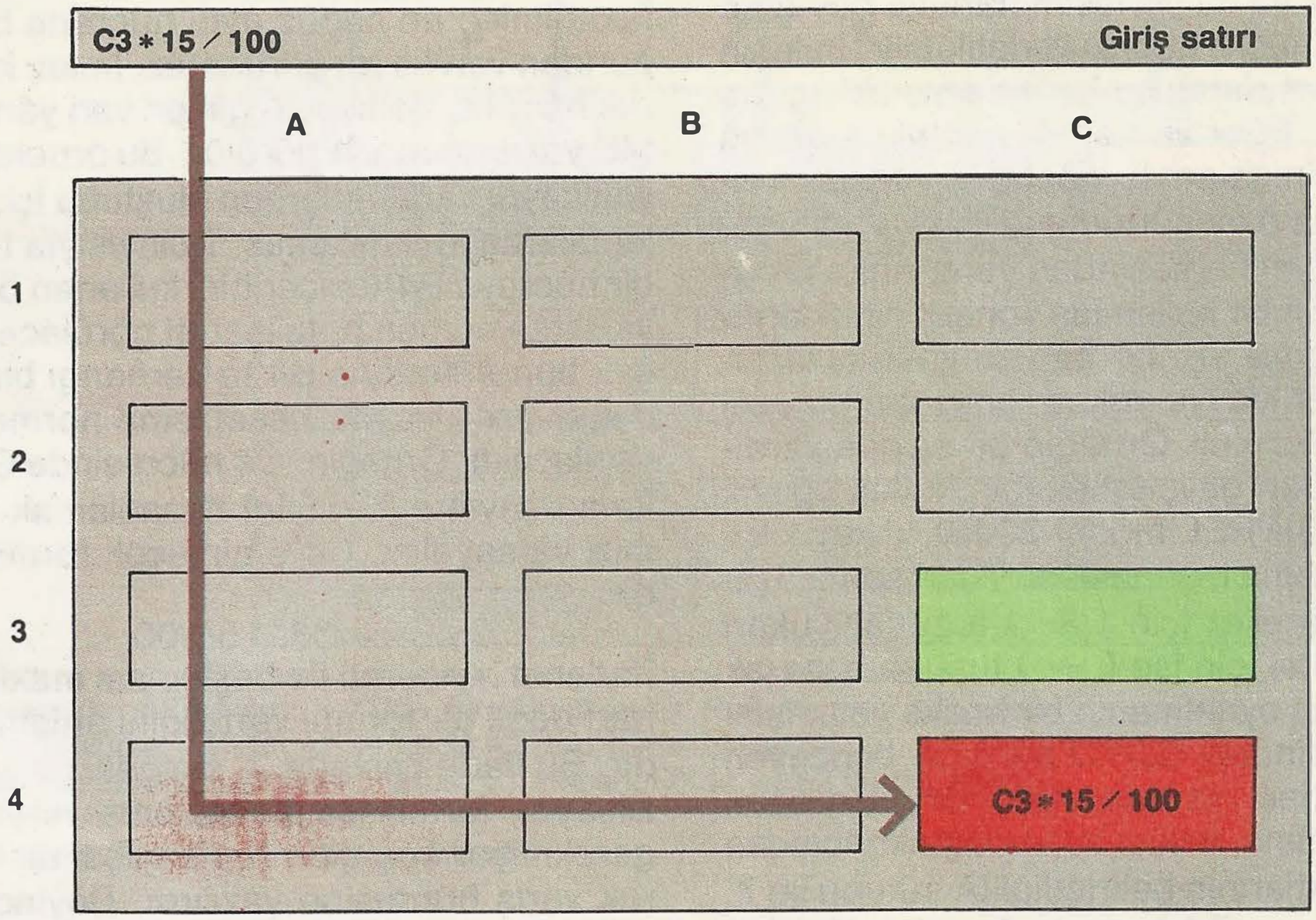
BİR FORMÜLÜN GİRİLMESİ

Veri kaydının hücresi

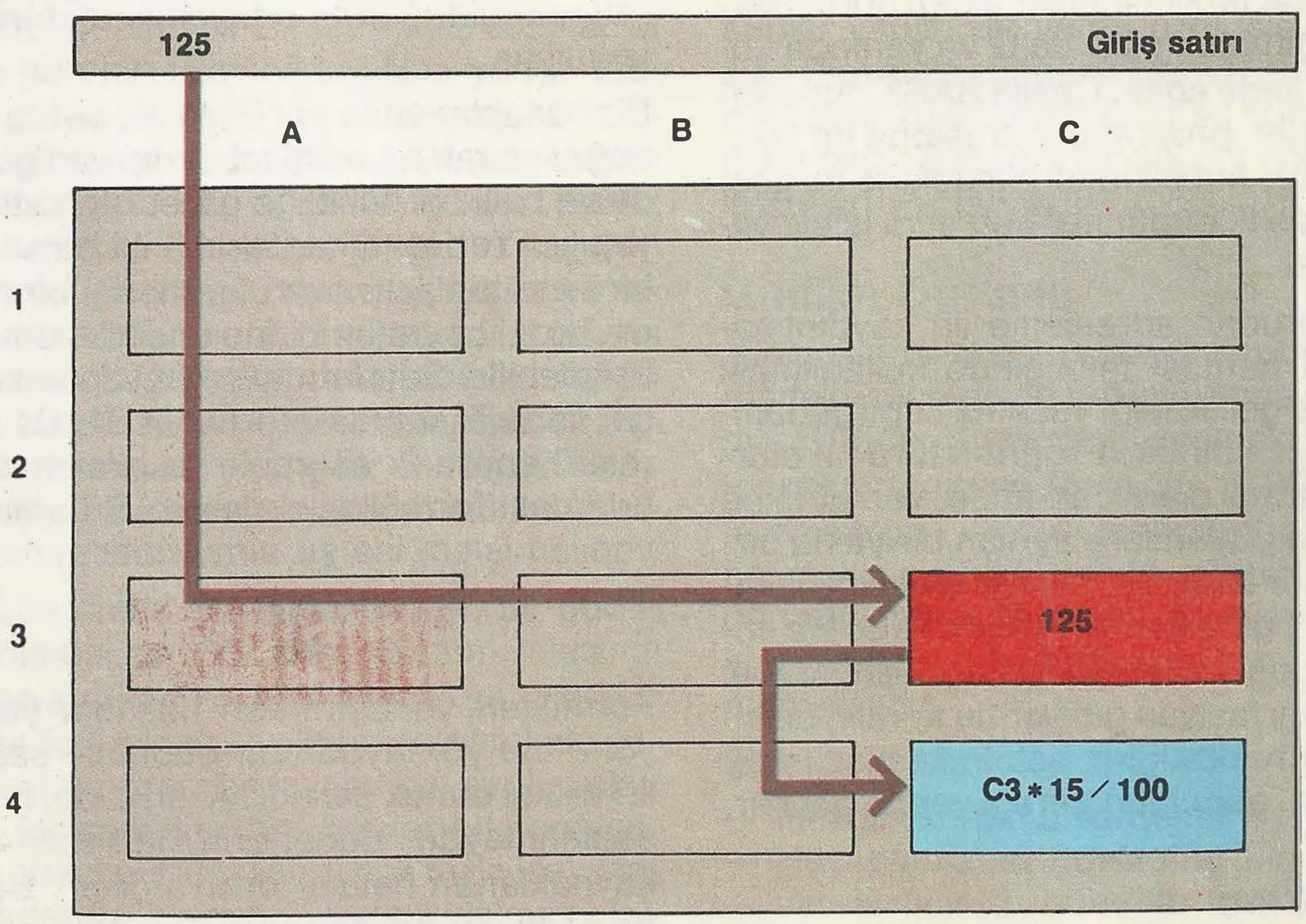
İmlecin bulunduğu hücre

Formülün hücresi

YANLIŞ



+ simgesinin konmasıyla bir çeviri hatası üretilmiş olur. Girilen satır, başlık satırı olarak anlaşılır ve dolayısıyla imlecin bulunduğu satıra bir kez daha yazılır.



C3 hücresine ne giriş yapılırsa yapılsın, C4 hücresinin içeriğinde hiçbir değişiklik yaratmaz, çünkü C4 hücresinde bulunan, bir formül değildir.

imleç alışılmış biçiminde (çizgisel ya da dikdörtgen) olmayıp, normal olarak hücrenin boyutlarına eşit bir dikdörtgen biçimindedir, fakat ters görüntülenir. İmlecin yeri bu dikdörtgenin istenen hücreye yerleştirilmesiyle değiştirilir. İmleç iki biçimde oynatılabilir: Belli bir güdüm aracılığıyla varış hücresine doğrudan yöneltilebilir ya da yerleşim haritası boyunca adım adım (hücre hücre) ilerletilebilir. İmlecin hareketi, özel olarak bu işe ayrılmış birkaç tuş ile sağlanır. Eğer varsa, klavyedeki alışılmış kaydırma simgeleriyle gösterilen tuşlar kullanılır; yoksa özel güdümlere başvurmak gerekir. Yerleşim haritasından yararlanan programlar yaygın bir kullanıma yönelik olduklarından, donanımla sıkı bir bağları yoktur. Makinede tam bir klavye yoksa, program bu özel duruma uydurulur. Örneğin bu sayıda verdiğimiz örnekleri geliştirmek için kullandığımız makinede (SIPREL model 2040), dikey kaydırmalar, atlama tuşuna basıldıktan sonra aşağı yönde hareket için (←) tuşuna, yukarı doğru hareket için ise (→) tuşuna basarak yapılır. İmleci oynatmanın bir başka yolu, işlevi BASIC dilindeki GOTO'nunkine benzeyen bir güdümden yararlanmaktadır. Örneğin, > A7 güdümü, imleci bulunduğu konumdan A7 koordinatlarının belirlediği (A sütunu ile 7. satırın kesişmesi) hücreye yöneltir. > simgesi GOTO komutuna karşılık gelir ve A7 koordinatları da GOTO'yu izleyen satır sayısıyla aynı anlamı taşır. Bu güdümün başka program türlerinde farklı biçimler aldığı görülmektedir. Örneğin Multiplan'da G ve yanında koordinatlarla ifade edilir. Çeşitli BASIC türlerinde olduğu gibi, programlar arasında da farklılıklar görülür. Ancak farklı simgelerle de gösterilseler, çeşitli güdümler hep aynı işlevi yerine getirirler.

İmlecin doğrudan adresleme ile kaydırılması, yerleşim haritası tekniğinde kullanılması öngörülen ve gelişkinliği yazılıma bağlı güdümlerden yalnızca biridir. Program türü ne olursa olsun, normal olarak ekranda, veri girişine ayrılan bir ve güdümlere ayrılan bir ya da birden fazla satır bulunur. Klavye aracılığıyla girilen her veri, giriş satırına yazılır ve sonra imlecin bulunduğu hücreye yerleştirilir. Ancak güdüm anlamı taşıyan girişler bu kuralın dışındadır. Bu durumda giriş satırında hiçbir şey görünmeden, güdümün gereği yerine getirilir.

Giriş Durumu

795. sayfada, bir veriyi B4 hücresine yerleştirmek için, imleç başlangıçta C2 pozisyonunda iken yapılması gereken işlemlerin bir şeması verilmektedir.

İlk aşamada imleç B4 hücresine yerleştirilir. Bundan sonra, makinede yazılırken giriş satırında görünen veri kaydının (bizim örneğimizde DENEME sözcüğü) girişi yapılabilir. Giriş biter bitmez program varış hücresini doldurur, fakat imleç de henüz aynı hücrede bulunduğundan veri kaydı görünmez. İmleç kaydırılarak hücre boşaltılınca, girilen veri yani DENEME yazısı ekranda görülür. Bu örnekte girilen veri kaydı karakterlerden oluştuğu için hesaplamalarda kullanılamaz. Dolayısıyla herhangi bir hücreye B4'ün içeriğini kullanan bir formül yazılırsa hemen hata işareti görülecektir. Ancak bunun tersine B4'te herhangi bir sayısal değer yer alsaydı, hesaplama normal olarak yapılacaktı. Örneğin, C4 hücresinde C3'te bulunan sayının % 15'ini hesaplamak istediğimizi varsayalım. C4'e girilecek formül şöyledir:

+C3*15/100

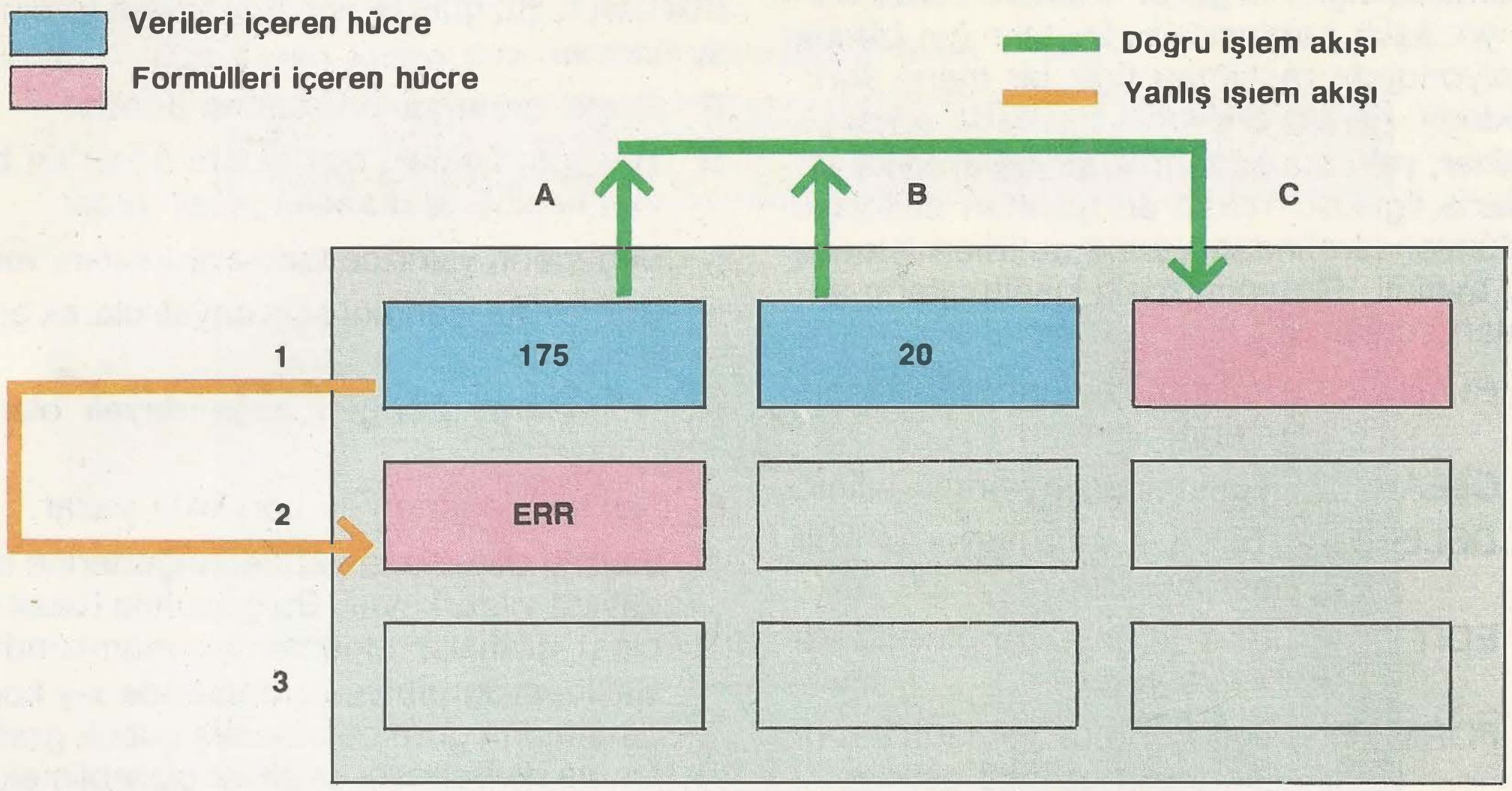
İfadenin + işareti ile başlaması makineye bir harf değil bir formül verildiğini anlatmak içindir. Bu ifade başında + simgesi olmadan yazılsaydı, formül bir hesaplama ve yapılması gerekmiyormuş gibi, harfsayısal bir veri olarak varış hücresine yazılırdı. Böylece ifade, yalnızca bir karakter zinciri olarak yerleşim haritasına yazılmakla kalır ve hiçbir sonuç vermezdi. 796-797. sayfalarda formülü girmek için yapılacak işlemler gösterilmiş ve + işareti unutulduğunda ortaya çıkan durum açıklanmıştır.

Bir hesaplamanın yürütülüşü, bütün adımlar doğru olarak izlendiği takdirde, veri girişi anında ve belli bir önceliğe göre, otomatik olarak yapılır. Yerleşim haritasının iki boyutu (satırlar ve sütunlar) vardır. Herhangi bir hesaplama bu iki boyuttan birine öncelik verilerek geliştirilebilir. Satırlara göre hesaplama yapılırsa, önce ilk satırdaki (örneğin B1, D1 vb. hücreleri) sonra 2. satırdaki ve sırayla diğer satırlardaki formüller ele alınır. Sütunlara göre yapılan işlem ise şu sırayı izler:

A1, A2, A3, vb.
B1, B2, B3, vb.

Formülleri yerleştirirken hesabın yapılacağı yöne (bu yön uygun bir güdümle saptanabilir) dikkat etmek gerekir. Aksi halde, henüz hesaplanmayan değerlerin kullanılmasından kaynaklanan hatalar ortaya çıkar. Bu sayfada üstte, bir ara sonuç kullanılan bir hesaplama örneği verilmektedir. A2 hücresinde, kendisi de bir önceki hesabın bir sonucu olan

YANLIŞ BİR İŞLEM YÖNÜ SEÇİLMESİNDEN KAYNAKLANAN BİR HATA ÖRNEĞİ



C1 hücresi +A1 —B1 formülünü içerir, bunun sonucu A2 hücresinde kullanılacaktır. A2 hücresi +A1/C1 formülünü içerir. Hesaplamalar sütunlara göre yürütülürse, A1'den A2'ye geçiş A2'deki işlemi başlatır.
C1'deki hesaplama yapılmadan önce, C1'in içeriği 0'dır ve A2'de 0'a bölme nedeniyle bir hata ortaya çıkar.
Doğru işlem akışı satırlara göre yapılandır.

C1'in içeriği kullanılır. Hesaplama yönü A2 hücresine, geçerli bir C1 içeriğiyle ulaşılacak biçimde satırları izlemelidir. Aksi yönde (sütunlara göre) gidilerek yapılan hata, A2 hücresinde 0'a eşit bir bölen bulunmasından (A2'nin içeriği C1'den önce hesaplanır, henüz hesaplanmayan C1'de ise 0 vardır) kaynaklanmaktadır. Sistem bu anormal durumu (hiçbir sayı sıfıra bölünemez) farkeder ve bir hata mesajı verir. A2 hücresindeki ifade bir toplam olsaydı, sistem hatanın farkına varmaz ve hiçbir uyarı olmaksızın yanlış bir sonuç elde edilirdi. Hatayı hâlâ tespit edebilme imkânları oldukça sınırlı olurdu.

Hesap yöntemini değiştirmek için, bir güdümün girilmesi gerekir. Her güdüm, ekranın kendisine ayrılmış bir bölümünde yer alan bir menüden ilgili şıkkın seçilmesiyle harekete geçirilir. Bazı programlarda menüye ayrılan alan ekranın ikinci satırıyken, bazılarında son satır ya da şayet güdüm sayısı çok fazla ise, sondaki satırlardır. Her halükârda iki işleyiş biçimi vardır: **Yürütme durumu** ve **güdüm durumu.**

Yürütme durumunda veriler girilebilir. Özel hesaplar yapılabilir ya da özel işlevler (bilgiyazım, disket üzerinde saklama, grafiklerin çizimi) yerine getirilebilir. Güdüm durumunda, yeni bir değişikliğe kadar geçerli olacak çalışma koşulları saptanabilir.

Güdüm Durumu

İşin başında imleç, yerleşim haritasının sol üstündeki hücresiride (A1) yürütme durumundadır. Güdüm durumuna geçmek için özel bir fonksiyon tuşuna [örneğin, SIPREL makinelerde simge (÷)'dır] basmak gerekir. Bu tuşa basmakla elde edilen kod, güdüm menüsünün görüntülenmesini sağlar. Güdüm durumundaki bir yerleşim haritası örneği bu sayfada sunulmaktadır.

Eldeki parametreler şunlardır:

— İmlecin konumu (örnekte B1)
— Hesaplamanın yürütülüş biçimi (örneğin satırlar için Z, sütunlar için S)
— Güdüm menüsü

İlk iki parametrenin işlevi bilgilendirmekken,

(bu arada, hesap biçiminin öneminden dolayı, Z ya da S'nin hep görüntülendiğine dikkat edelim) menü, tercihi yönlendirmeye yarayan bir önbellek işlevini görür. Yukarıda sözü edilen yerleşim haritasında, bu tekniğin birçok versiyonunda rastlanan tipik bir menü verilmektedir. Farklı versiyonlarda rastlanan değişiklikler, yalnızca güdümleri simgeleyen kısaltmalarla ilgilidir. Yoksa en azından bellibaşlı güdümler tarafından yerine getirilen işlevler hep aynıdır. Örneğimizdeki kısaltmaların anlamları şunlardır:

B	BLANK	İmlecin bulunduğu alanın içeriği silinir.
C	CLEAR	Tüm yerleşim haritası silinir.
D	DELETE	Tüm bir satırı veya tüm bir sütunu siler.
E	EDIT	Bir hücrenin içeriğini düzeltmeye yarar.
F	FORMAT	İlgili hücrede verilerin sunuluş yazı düzenini belirler.
G	GLOBAL	Tüm yerleşim haritasındaki verilerin sunuluş yazı düzenini değiştirir.
I	INSERT	Yerleşim haritasına bir satır veya sütun ekler.
M	MOVE	Bir satır veya sütunu haritanın bir yerinden başka bir yerine kaydırır.
P	PRINT	Bilgiyazım için parametreleri saptar.
R	REPLICATE	Haritanın bir bölümünü (source range-kaynak) bir başka yerine (target range-erek) kopya eder.
S	STORAGE	Haritanın (bulunan verilerle birlikte) disket üzerinde bir kütükte saklanmasını veya daha önceki bir haritanın geri getirilmesini sağlar.
T	TITLE	Başlıkları, yani harita (scroll) boyunca sabit kalacak satır veya sütunların girişini sağlar.
W	WINDOW	Ekranda bir pencere açar.
—	REPEAT	Bir simgeyi tüm bir alan boyunca yineler.

Yukarıdaki güdümlerden bazılarının işlevlerinin betimlenmesi, kullanımlarına yeterince ışık tutmaktadır. Daha karmaşık güdümler daha ayrıntılı bazı seçenekler de sunar.

Aşağıda bu karmaşık güdümlerin bazılarını, değişik versiyonlarda farklı kısaltmalar olabileceği uyarısıyla birlikte sunuyoruz:

FORMAT. Bu güdüm için öngörülen tercihler şunlardır.

D Başlangıçta yazı düzenine dönüş

G Hesapları eldeki olanaklara göre, en büyük kesinlikle (hassasiyetle) yapar.

I Değerlerin yalnızca tamsayılı kısmını verir.

L Hücrelerin içeriğini sola dayalı olarak gösterir.

R Hücrelerin içeriğini sağa dayalı olarak gösterir.

$ Sayısal değerleri iki ondalıkla yazar.

***** Sayısal değerlerin yerine, değerlerine eşit sayıda yıldız koyar. Bu güdümle (basit de olsa) grafikler elde etmek mümkündür. Başka program versiyonlarında x-y koordinatlarına göre belirlenmiş çubuk grafikler, ya da dairesel grafikler çizilebilmektedir.

802-805. sayfalarda, FORMAT güdümünün başlıca seçeneklerinin kullanımına ilişkin bazı örnekler verilmektedir.

GLOBAL. Bu güdüm için geçerli başlıca seçenekler şunlardır:

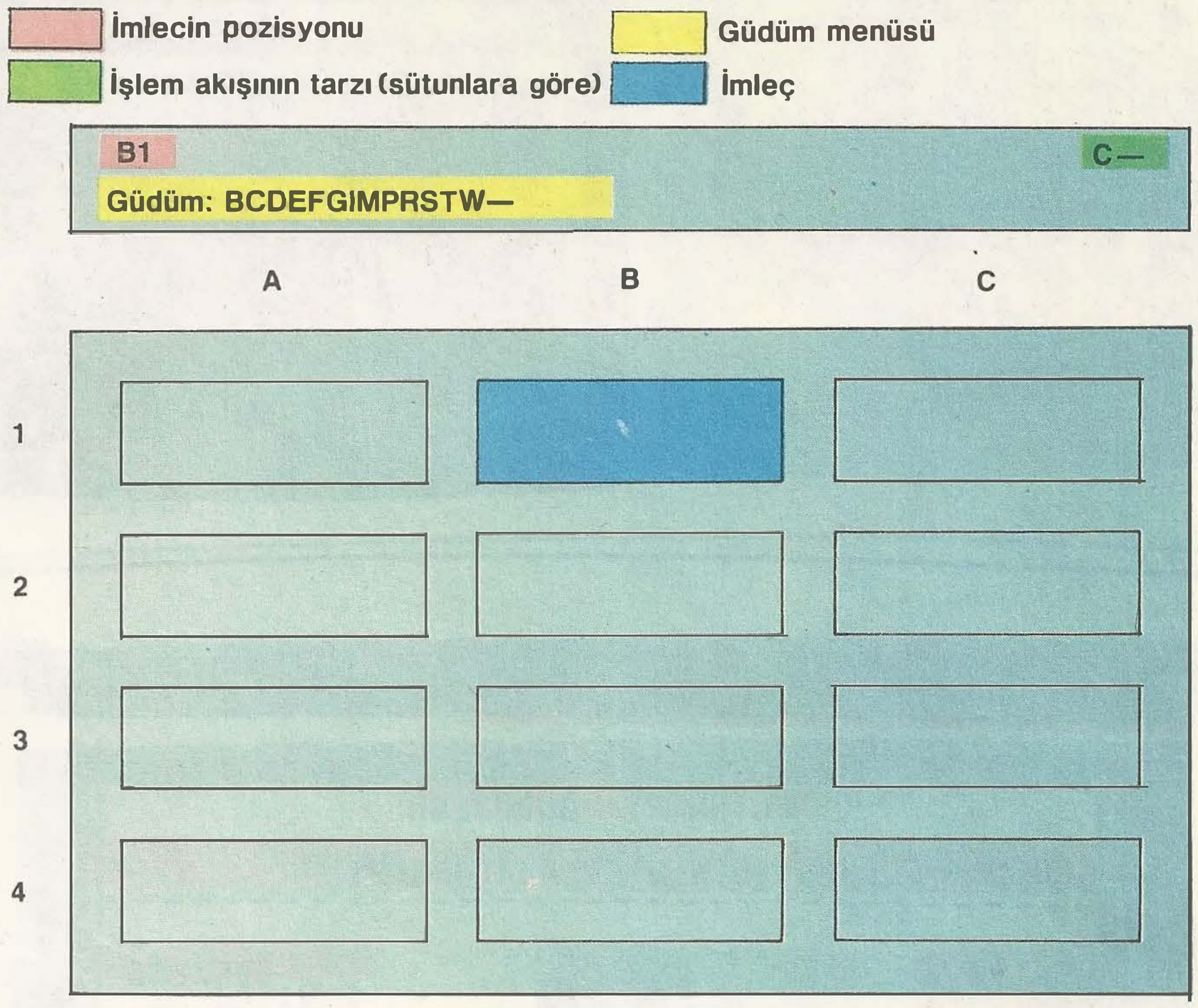
C Sütun genişliğini değiştirir. Yerleşim haritasının kapasitesi, her hücrenin karakter sayısının geniş sınırlar içinde artırılıp azaltılmasıyla, uygulamanın özel gereklerine uydurulabilir.

R Bu seçenekle hesap yöntemi seçilir. Ya her veri girişinde tüm harita otomatik olarak baştan hesaplanır ya da sistem elle girilmesi gereken bir onay [örneğin (!) simgesi] aldıktan sonra hesaplamaya girişir. Bu tercih çok sayıda veri girildiğinde yararlı olur. Otomatik yöntemde, girilen her yeni veri için baştan hesaplama yapılır, bu da büyük zaman kaybına yol açar. Elle hesaplama yönteminde, veriler hiç beklemeden birbiri peşi sıra girilebilir. Sonuçta uygun bir güdümle tüm hesaplama yapılır.

O Hesaplama sırasının tercihi (satır, sütun), bu tercihle hesabın izleyeceği sıra saptanır.

TITLES. Yerleşim haritasının kaydırılması (scroll) sırasında sabit kalacak bazı satır ve sütunların başlık olarak belirlenmesine yarar. Bu komutun kullanılmasıyla, haritanın uzak bölümlerinde çalışırken bile (zorunlu bir kaydır-

BİR GÜDÜM MENÜSÜNÜN OLUŞTURULMA ÖRNEĞİ



ma, belli bir bölümün ekrana görünmesi ve diğerlerinin kaybolması anlamına gelir) her satırın (ya da sütunun) açıklamasının her an görüntülenebilmesi mümkündür. Bu seçenekle, başlık olarak kullanılan bölümler sürekli görüntülenir. Ne var ki, bu durum çalışma alanının daralması anlamına gelir. 802-805. sayfalarda TITLE güdümünün kullanımına bir örnek verilmektedir.

Yerleşim Haritası Tekniğinin Fonksiyonları

Yerleşim haritası tekniğini kullanan programlarda hesapların ya da mantık işlemlerinin yapılabilmesini mümkün kılacak çok sayıda fonksiyon vardır. Sayısal fonksiyonlar, fazla karmaşık olmayan haritaların hazırlanmasında çok yararlı bir araç durumundayken, mantıksal fonksiyonlar bazı kısıtlamalarla da olsa, giderek gerçek programlara dönüşen karmaşık uygulamaların geliştirilmesine olanak verirler. Bazı yerleşim haritası tekniklerinde (örneğin Lotus'da kullanılanlarda) kullanıcının tanımlayıp yazdığı ve belli bir tuşa basılarak devreye sokulan fonksiyonlar olan **makrogüdümler** (ya da makrolar) bulunmaktadır. Bu güdümlerin yapısı ve kullanımı BASIC dilinin alt-programlarındakine çok benzer. Temel bilgilere sahip olan bir kişi, makroları kullanarak sistemi tümüyle programlayabilir.

Kullanılacak fonksiyonun biçimi ne olursa olsun, onu bir hücreye yerleştirmek mümkündür. İşlem evresinde, fonksiyonun yerine getirilmesiyle ortaya çıkan sayısal değer bu hücrede yer alır. Genelde, bir fonksiyonun argümanları diğer hücrelerin sayıları ya da adreslerinden oluşabilir. Birinci durumda, fonksiyonun yürütülmesi, argümanların değeri üzerinden anında yapılırken, ikinci de sayısal değer belirlenen adresten (satır, sütun) alınır. Bir fonksiyonu girmek için kullanılan genel sözdizimi şöyledir:

@ xyz(n)

Burada

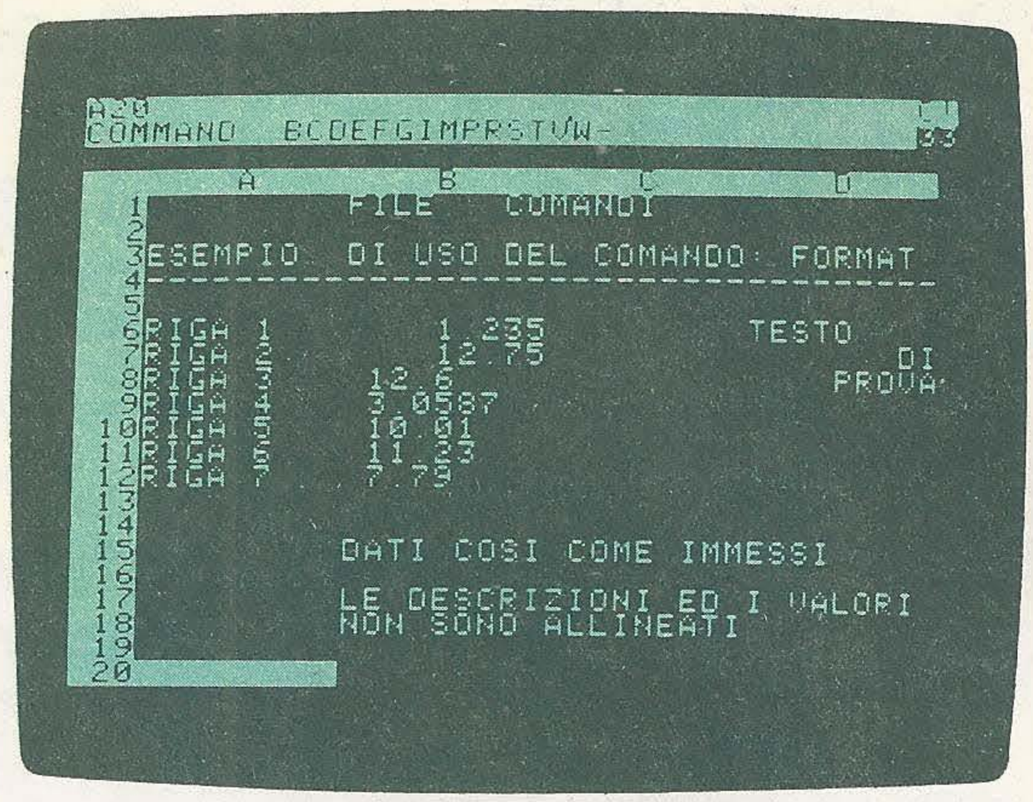
@ fonksiyonu tanıtan simgedir (program türüne göre değişir).

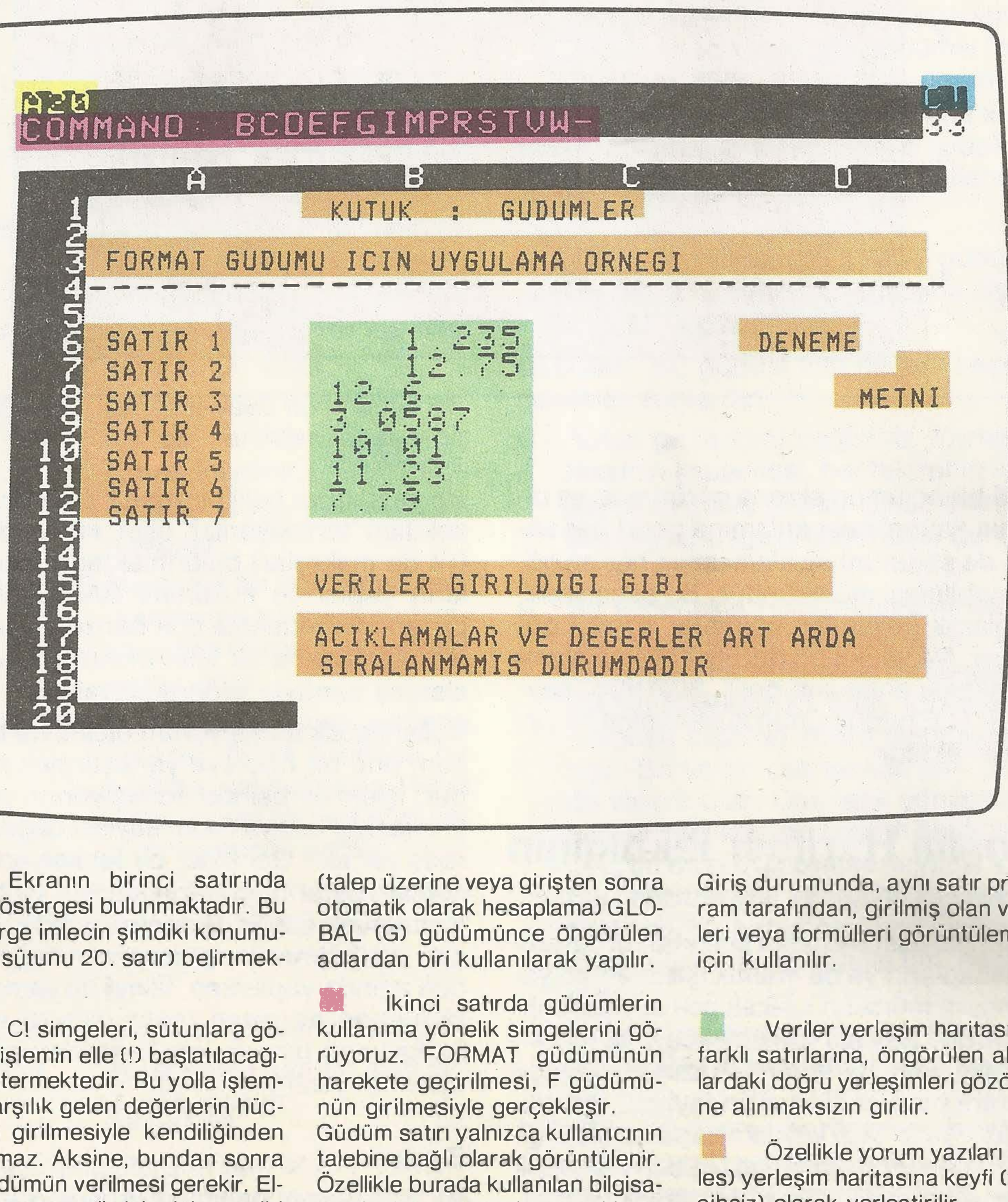
xyz fonksiyonun adıdır

FORMAT GÜDÜMÜNE İLİŞKİN UYGULAMA ÖRNEĞİ (1)

Bu ve bunu takip eden üç sayfada dört ekran çıkışı görüyoruz. Bunlarda yerleşim haritasında FORMAT güdümünün kullanımı açıklanmaktadır.

Örneklerde, SIPREL 2030 S kişisel bilgisayarda işletilen bir versiyondan yararlanılmıştır. Farklı versiyonlarda aynı işlevleri yerine getiriyor olsalar da, farklı güdüm kodları kullanılabilir.





Ekranın birinci satırında A20 göstergesi bulunmaktadır. Bu gösterge imlecin şimdiki konumunu (A sütunu 20. satır) belirtmektedir.

C! simgeleri, sütunlara göre (c) işlemin elle (!) başlatılacağını göstermektedir. Bu yolla işlemler, karşılık gelen değerlerin hücrelere girilmesiyle kendiliğinden başlamaz. Aksine, bundan sonra bir güdümün verilmesi gerekir. Elle veya otomatik işleyişin tercihi (talep üzerine veya girişten sonra otomatik olarak hesaplama) GLOBAL (G) güdümünce öngörülen adlardan biri kullanılarak yapılır.

İkinci satırda güdümlerin kullanıma yönelik simgelerini görüyoruz. FORMAT güdümünün harekete geçirilmesi, F güdümünün girilmesiyle gerçekleşir. Güdüm satırı yalnızca kullanıcının talebine bağlı olarak görüntülenir. Özellikle burada kullanılan bilgisayarda, bu ÷ tuşuyla gerçekleştirilir.

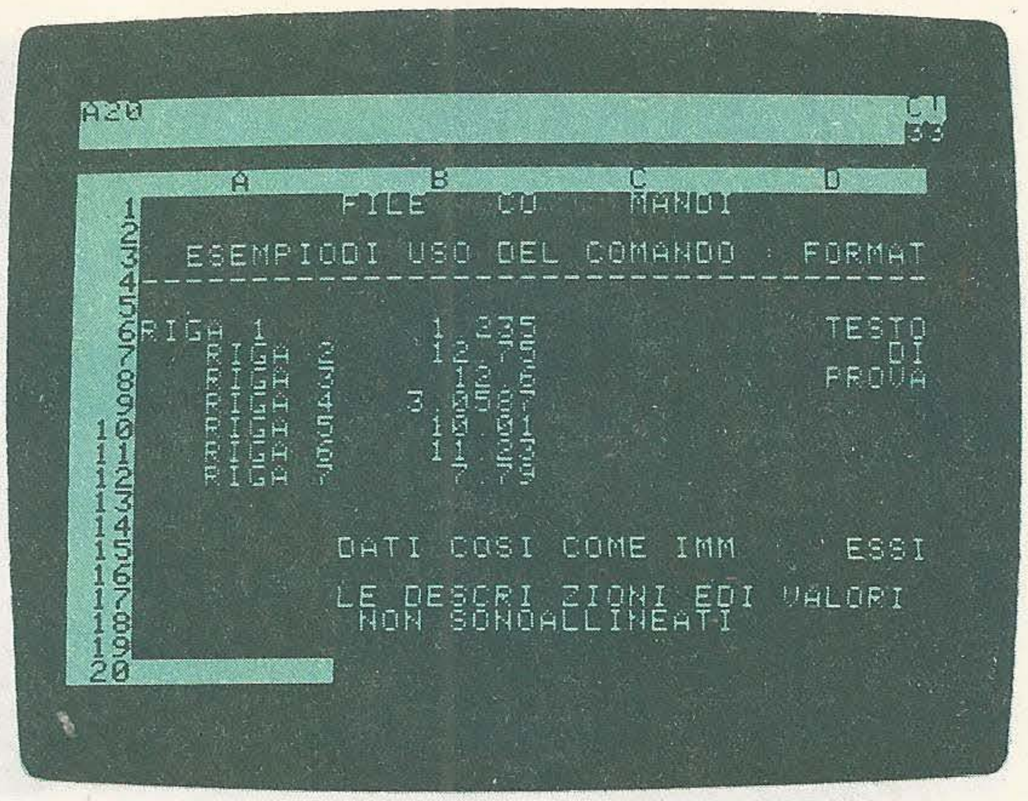
Giriş durumunda, aynı satır program tarafından, girilmiş olan verileri veya formülleri görüntülemek için kullanılır.

Veriler yerleşim haritasının farklı satırlarına, öngörülen alanlardaki doğru yerleşimleri gözönüne alınmaksızın girilir.

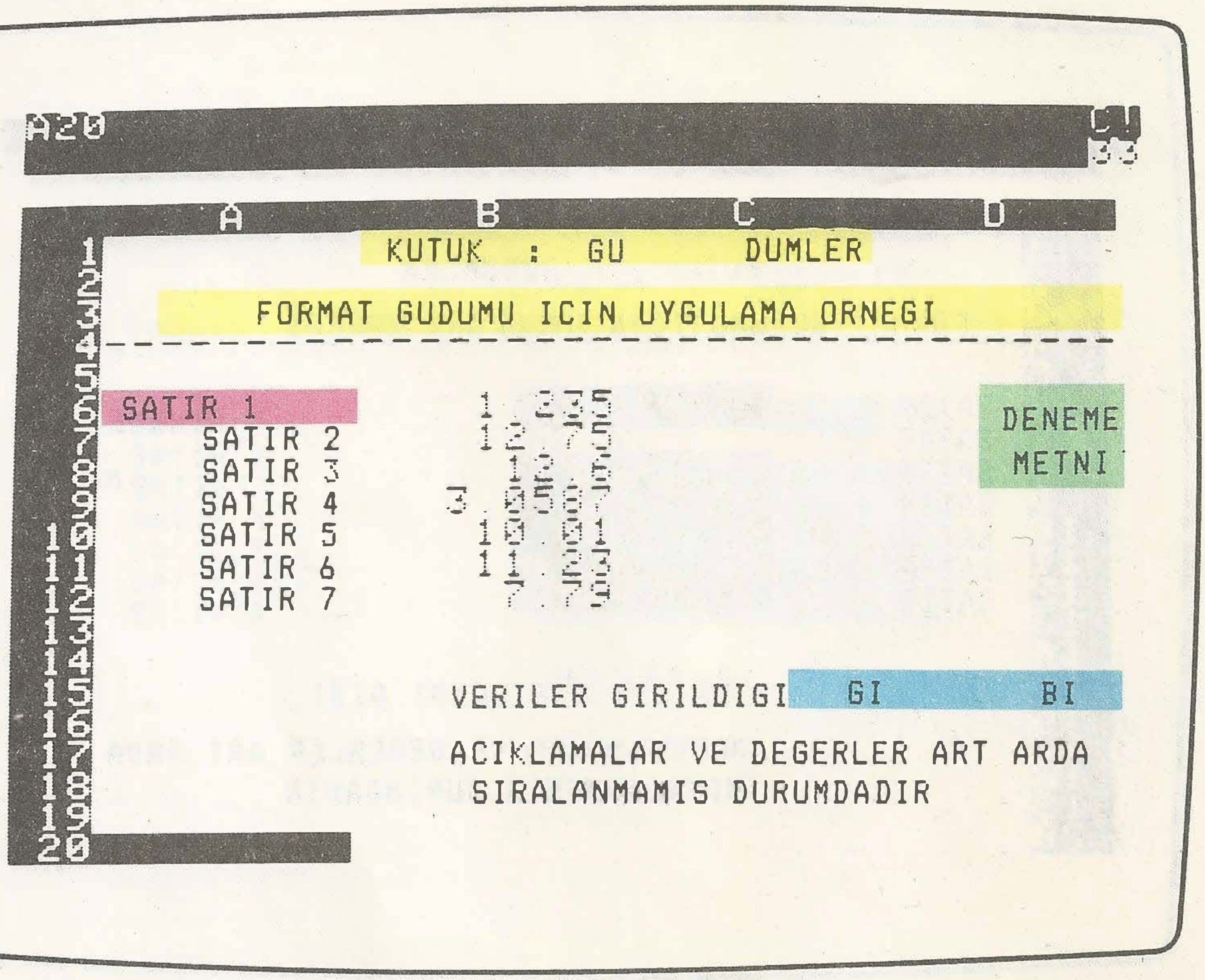
Özellikle yorum yazıları (titles) yerleşim haritasına keyfi (tercihsiz) olarak yerleştirilir.

FORMAT GÜDÜMÜNÜN KULLANIMINA İLİŞKİN UYGULAMA ÖRNEĞİ (2)

Bu ekranda, bir önceki ekranda girilen veriler, bu kez G(GLOBAL) güdümü, F(FORMAT) seçeneği ve R(RIGHT) şıkkının yardımıyla (ardarda bir sıralamadan sonra) sağa dayalı olarak yazılmıştır.



Daha önce sola dayalı olarak yazılmış olması nedeniyle tek istisnayı oluşturan A6 hücresi dışında, tüm hücreler sağa dayalı yazılır. A6 hücresinde, bir tek hücreyi ardarda sıralamayı sağlayan komut kullanılmıştır.



Genel olarak yerleşim haritalarında bir yazı düzeni oluşturmak için yararlanılabilecek iki yol vardır. F güdümüyle harekete geçirilen ilk yöntem, güdümün girildiği anda imlecin bulunduğu hücre ile ilgilidir. İkinci yöntem (GLOBAL) ise, daha önce F güdümünün yöneltildiği hücre dışındaki tüm hücrelerle ilgilidir. Yeni bir ardarda sıralanma nedeniyle metinler parçalara ayrılır.

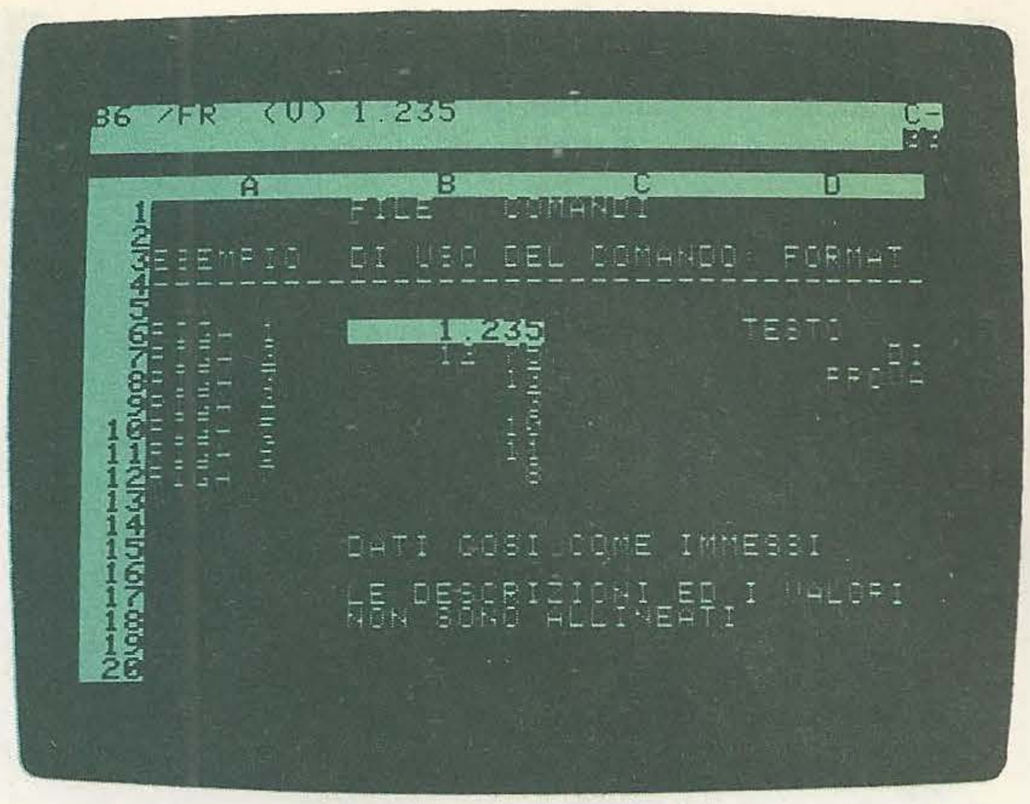
Örneğin, bitişik iki hücreyi (C15 ve D15) işgal eden GIBI kelimesi ikiye bölünür.

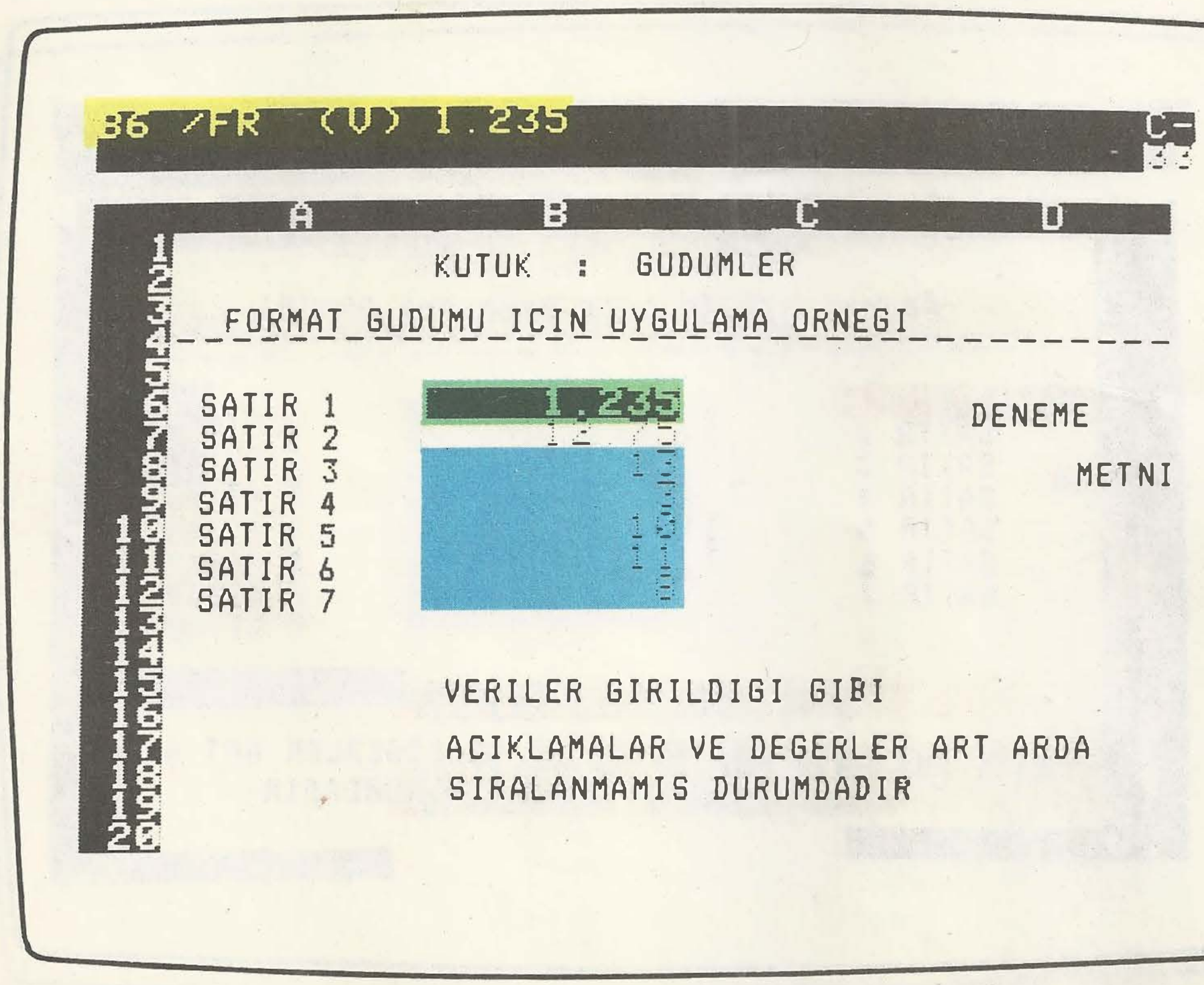
Aynı sorun, KUTUK: GUDUMLER ve UYGULAMA ORNEGI metinleri için de geçerlidir.

Aynı sütunda bulunan farklı üç hücrenin içeriklerinden oluşan DENEME METNI yazısı ise, burada herhangi bir sorun yaratmaz.
Yerleşim haritasına bazı metinler girildiğinde, doğal olarak yanyana sıralanmış birden fazla hücrenin kullanılması gerekir. Bu çözüm ancak GLOBAL-FORMAT güdümünü kullanmamaya dikkat edildiği takdirde, kabul edilebilir. Aksi takdirde, hücrelerin kapasitesi, en uzun metinlerin girilebileceği kadar artırılabilir, fakat bu durumda aynı anda ekranda görülebilecek hücre sayısı azalır. Çünkü, genişleme için kullanılacak tek bir GLOBAL güdümü, otomatik olarak tüm hücreleri etkileyecektir.

FORMAT GÜDÜMÜNE İLİŞKİN UYGULAMA ÖRNEĞİ (3)

Bu sayfadaki ekranda, belli bir hücreye verilen içerik ve yazı düzeninin özelliklerini göstermeye yarayan seçeneğin işleyişi yer almaktadır.





Belli bir hücrenin içerik ve yazı düzenine ilişkin bilgiler, imlecin incelenecek hücreye getirilmesiyle elde edilir.

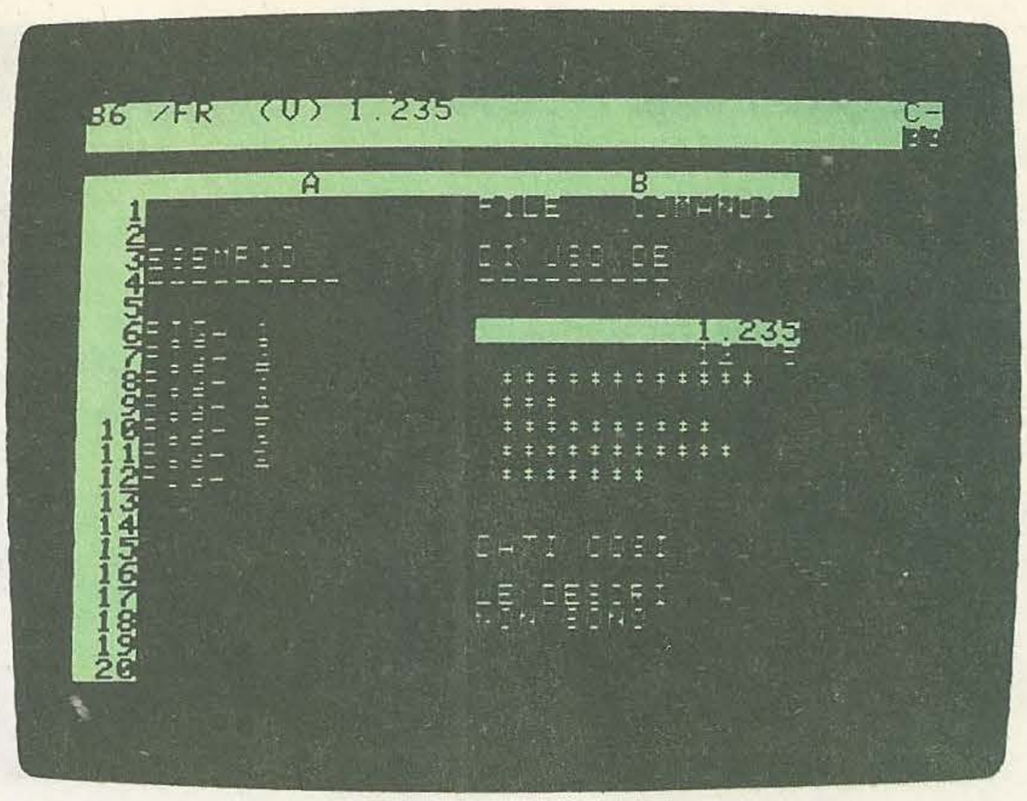
İlk ekran satırında, seçilen hücrenin adresi (B6), içeriği (1.235) ve verilen özellikleri [/FR ve (V)] görülür. Bu durumda, B6 hücresinde sağa ardarda sıralama yapan bir yazı düzeni (/ güdüm, F yazı düzeni, R right = sağ anlamındadır) vardır ve 1.235'e eşit bir veri kaydı [(V) = value = değer] içerir. Aynı veri kaydı bir karakter zinciri olarak da verilebilir; bu durumda metin (V) bulunmayacak ve 1.235 veri kaydı işlem yapmakta kullanılamayacaktı.

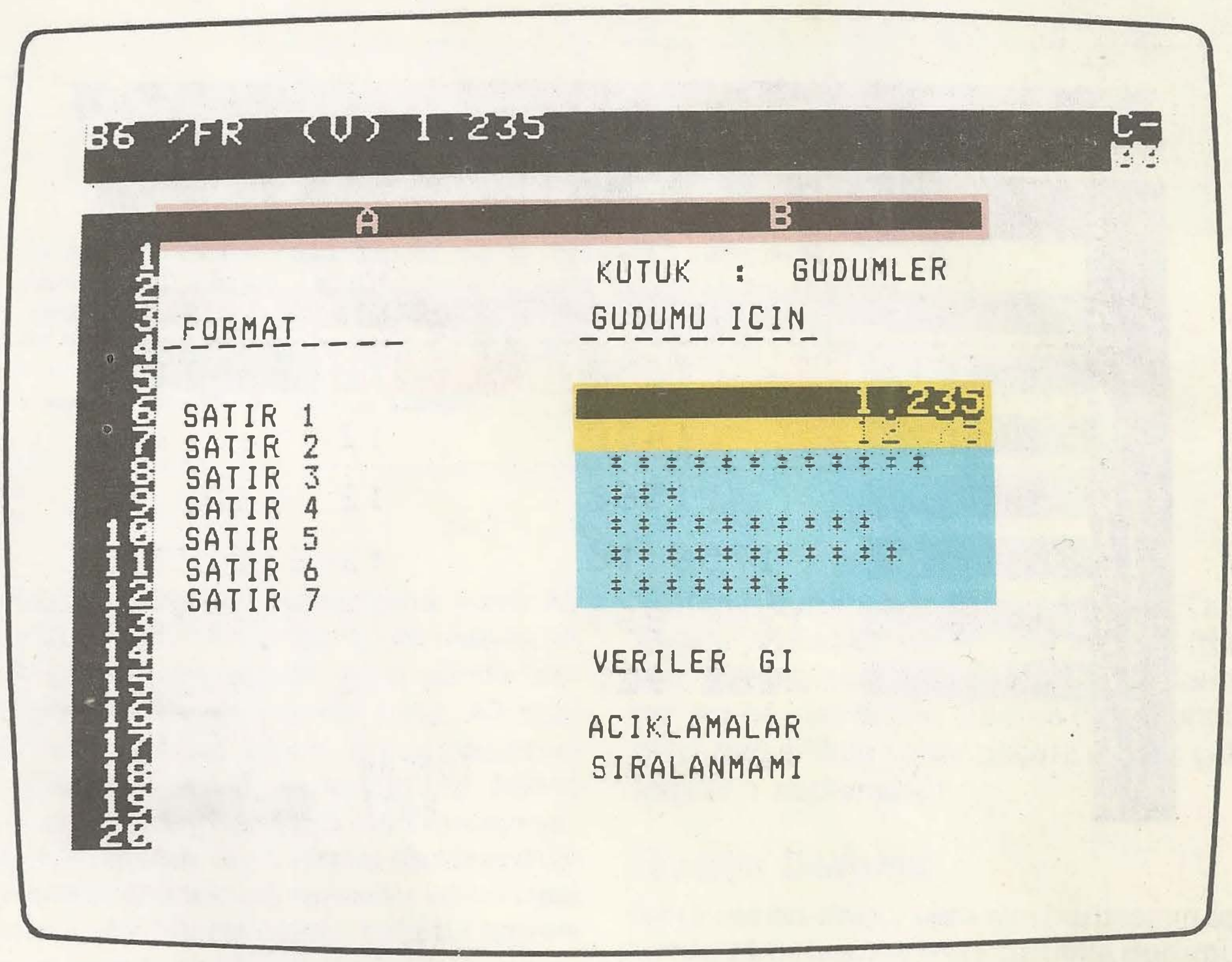
Veriler, giriş evresindekilerin aynı olmakla birlikte (bir önceki fotoğrafa bakınız), bu fotoğraf, tüm hücrelere tamsayı düzenini veren G/I (GLOBAL/INTEGER) güdümünün girilmesinden sonra çekilmiştir.

Değerler otomatik olarak yuvarlanmış (örneğin 7.79, 8 olmuştur) ve ondalık kısımları olmaksızın gösterilmişlerdir. Daha önce "lokal" yerel/FR düzeni verilen B6 ve B7 hücrelerindeki veriler, bir istisna oluşturmaktadır.
Yuvarlamanın yalnızca görüntüde gerçekleştiğini belirtmek gerekir; her hücreye verilmiş olan değerler aslında aynı kalmaktadır.

FORMAT GÜDÜMÜNE İLİŞKİN UYGULAMA ÖRNEĞİ (4)

Bu sayfadaki fotoğraf, önceki sayfalarda gösterilen yerleşim haritasının aynısını göstermektedir. Fakat bu kez, (talebin girilmesinden sonra) hücreler genişletilmiştir.





Hücre başına karakter sayısı iki kat artırılmış ve bunun sonucunda aynı anda ekranda bulunabilen sütun sayısı yarıya inmiştir.
G,* seçeneğinin girilmesi, girilen verilerin basitleştirilmiş bir çizisel gösterimini mümkün kılmaktadır.

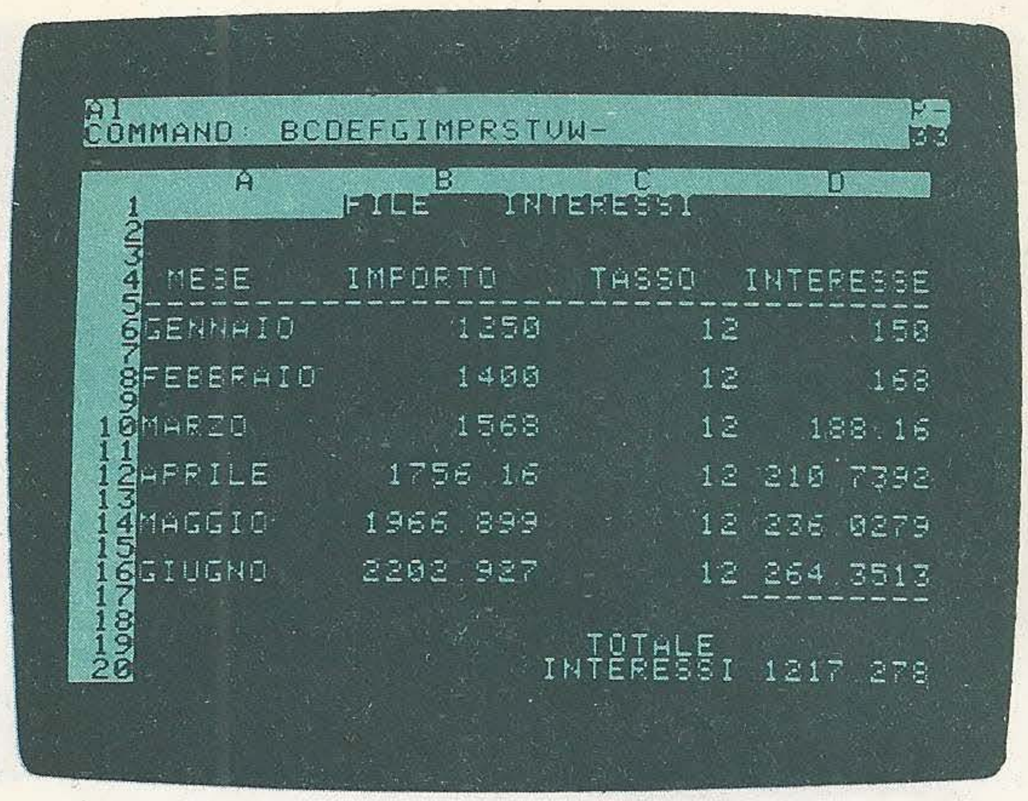
Her bir hücrede, sayıları alınan sayısal değere orantılı olarak bir dizi yıldız görülmektedir.

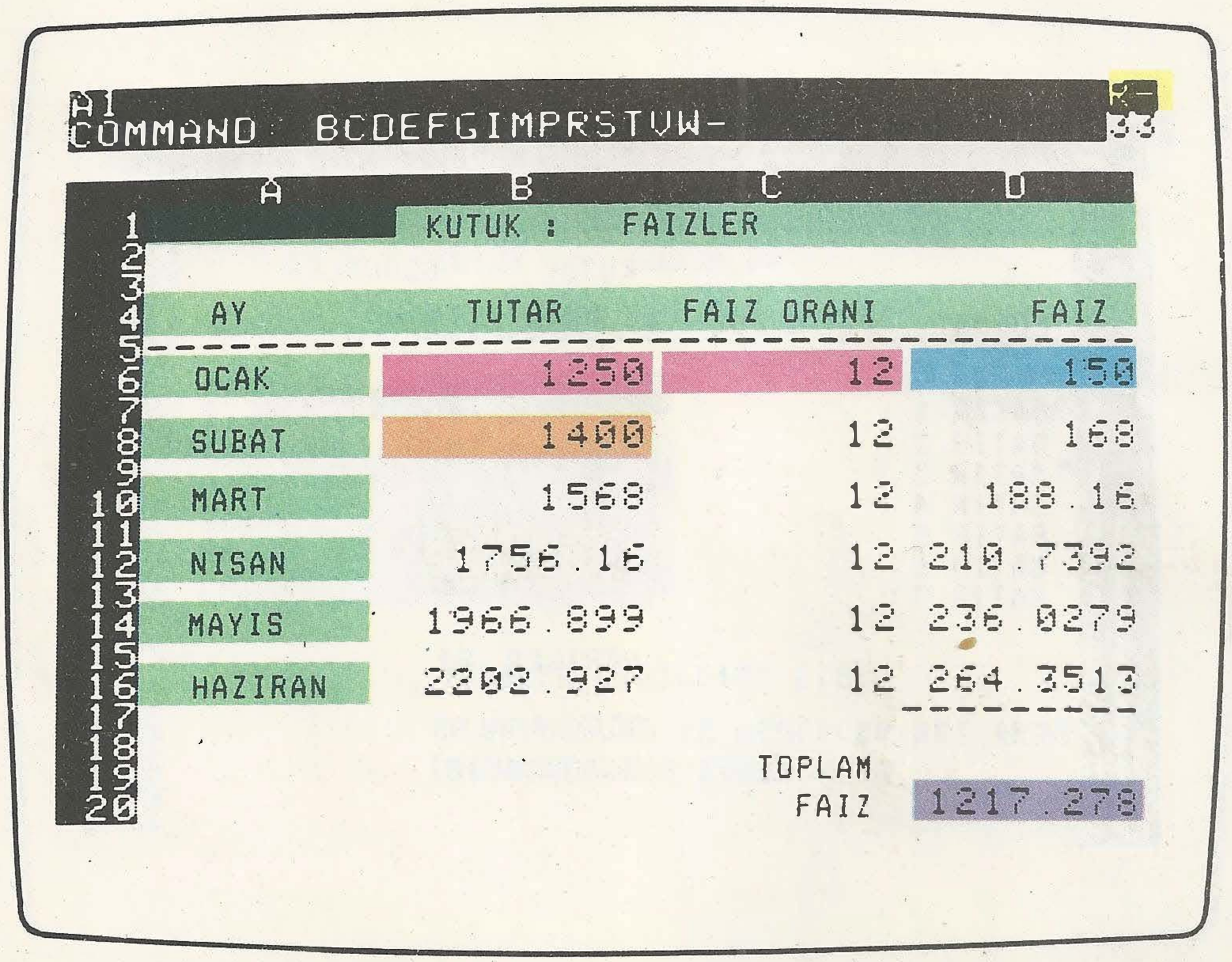
Daha önce de olduğu gibi, B6 ve B7 hücreleri verilmiş olan ''lokal'' yazı düzeni özellikleri nedeniyle bunun dışında kalmıştır.

Bu gösterim şekli, girilen sayısal verilerin akışına genel bir bakış sağlamak istendiğinde, çok yararlı olabilir. Daha özel yerleşim haritalarında, doğru çizisel gösterimleri gerçekleştirebilen seçenekler vardır. Bu durumlarda, ek metinlerin tümü ekrandan yok olur ve yalnızca istenen çizisel gösterim görülür.

BİLEŞİK FAİZ HESABI

Fotoğrafta, belli bir başlangıç sermayesi için bileşik faiz ödeme planını oluşturmak için hazırlanmış bir yerleşim haritası görülmektedir.
Her ay için esas alınacak sermaye, bir önceki ayın sermayesi artı faizlerden oluşur.





Birinci satır, satırlara göre hesaplama biçiminin seçildiğini göstermektedir.

1., 4. satırların ve A sütununun hücreler, açıklamalara ayrılmıştır.

Başlangıç sermayesi B6'ya girilmiştir, faiz oranı (% 12) ise C6'da bulunmaktadır.

D6 hücresine, B6 ve C6'ya girilen verileri kullanarak faizi hesaplayan, + B6* C6/100 formülü girilmiştir. Satırlara göre hesaplama yöntemi seçildiğinden, 1250 (B6) ve 12 (C6) verileri girilir girilmez, 150 (D6) değeri otomatik olarak belirir.

İkinci ayın faizini hesaplamak için, birinci ayın sermayesi artı ona düşen faizin gözönüne alınması gerekir. Bu nedenle B8 hücresine, otomatik olarak sermayenin değerini hesaplayan + B6 + D6 formülü girilir.
Bu yöntem tüm hücreler için geçerlidir. B ve D sütunlarındaki hücrelerin içeriklerinin, B6'daki başlangıç sermayesinin girilmesinin ardından aynı anda hesaplandığına dikkat edilmelidir.

Faiz toplamı, SUM (D6.... D16) fonksiyonu kullanılarak, D20'de hesaplanır.

Bürolarda Otomasyon

''Office automation'' ve ''office augmentation'' yani büro işinin otomatikleştirilmesi ve rasyonelleştirilmesi, sanayi toplumlarının ufkunda belirmekte olan bir gerçekliğin iki ayrı ifade biçimidir.

Görünen, ekonomik verimlilikte eşi görülmemiş bir sıçramanın gerçekleşmekte olduğudur. Bu sıçrama, iyi kullanılmazsa, teknolojik savurganlık yaratıp, iş dünyasındaki gerginlikleri artırabilir.

Olgunun toplumsal, yöntemsel ve örgütsel bileşkeleri öylesine belirleyici rol oynamaktadır ki, bu olguyla sadece teknolojik araçlar ve uzmanlıkların yardımıyla başedilebileceğini düşünmek safdillik olacaktır. Büro gerçekliği, tanımı gereği, (çok sayıda uzmanlık dalından kaynaklanan) çok disiplinli bir yaklaşımla bile üstesinden gelinemeyecek ölçüde karmaşık ve resmi ya da gayrı-resmi ilişkiler yönünden de oldukça yoğundur.

Bu yüzden, herhangi bir otomasyon süreci, yönetici kadrolar tarafından yönlendirilmesi ve denetlenmesi gereken karmaşık bir yönetsel sınama (revizyon) ve yeniden tasarlama etkinliğinin ilk değil, son aşamasını oluşturur. Özellikle süreç büro yaşamını kapsıyorsa, ilgilendirdiği kişi sayısı, çalışma biçimi üzerindeki etkisi ve uygulamaya değin mesleki ve yönetsel rollerin yapıları dolayısıyla, bu tespit daha doğru ve geçerli olur.

Böylece büro otomasyonu işletme yönetiminde hem sıkı sıkıya bağlı olduğu, insanî ve malî kaynaklardan, hem de planlama ve güncelleştirmenin nasıl oluşturulduğundan etkilenmektedir. Herhangi bir otomasyon stratejisi rekabet yeteneğinin bu stratejiden ne şekilde etkilendiğine bağlı olarak değişebilmek zorundadır.

Biz de ***büroda otomasyon*** *deyimini kullanmaktayız. Fakat* ***büro yaşamında enformatikten*** *söz ederken, konunun doğru anlaşılabilmesi için, bize göre belirleyici bir önem taşıyan iki noktanın altını çizmek istiyoruz.*

Her ne kadar enformatik, büro işinde (ya da daha doğru deyişle büro personelinin çalışmasında) yirmi yıldır uygulanmakta ise de teknolojik açıdan bakıldığında bugün geleneksel etki alanlarının yaygınlaştırılması zamanı gelmiştir. İkinci olarak, ''büroda otomasyon'' deyimi, tek başına, ele aldığımız sorunun karmaşıklığını yansıtmada yetersiz kalmaktadır. Örneğin, fabrika işindeki işbölümü modelleri, iş akışının hedefleri ve yapısı açısından tümüyle farklı olan büro çevresinde uygulanmak istendiğinde somut anlamlarını yitirmektedirler. Bu, Batı toplumlarında birkaç onyıldır yürürlükte bulunan iş yaşamının enformatize edilmesi sürecinin bugünkü gelişim aşamasını betimlemek için artık evrensel olarak kabul gören bu deyimin kullanılamayacağı anlamına gelmez.

Bilgilerin işlenmesinde otomatikleşme talebinin katlanarak artmasının altında yatan dürtülerin tümü, kavramsal olarak enformasyonun (bilginin) ve onun kullanımının ileri sanayi toplumlarında tuttuğu yerin büyüklüğüne bağlanabilir.

Gerçekte Batı (ve aynı zamanda Japonya) ekonomilerinde uzun süreden beri işgücünün giderek artan biçimde mal üretiminden (fabrika işi veya Amerikalıların deyimiyle mavi yakalıların işi) herhangi bir türde bilgi işlemeye (beyaz yakalılar ya da büro işçilerinin çalışması) doğru kaydığına tanık olunmaktadır.

809. sayfadaki grafik bu olgunun Ekonomik İşbirliği Kalkınma Örgütü'ne (OECD) üye ülkeler (bu arada İtalya) ekonomilerinde ne gibi görünümler aldığını ortaya koymakta ve bu ekonomilerde bilgi-işlem alanında istihdamın % 50'i oranını çoktan aşmış olduğunu göstermektedir.

Daha az dikkat çekmekle birlikte daha ilginç olan bir olgu ise, bu değişikliğin yaklaşık yirmi yıldır gitgide artan bir hızla yayılmakta olmasıdır. Özellikle Avrupa açısından neredeyse sosyo-ekonomik bir örnek oluşturma durumundaki İsveç'te bu olgu sözkonusudur. Bu satırların yazarının bildiği kadarıyla ülkemizle (İtalya ç.n) ilgili benzer istatistikler olmamasına rağmen, İtalya'da da tıpatıp benzer gelişmeler vardır: Üçüncü sektörün (hizmet sektörü) etkinliklerinde görülen patlama ve yönetim güçlerinin işlerine gösterilen büyük ilgi bunun kanıtlarıdır.

Bazı iktisatçılara göre, değişim malları enformasyon, uzmanlık ve bilgi olan ve ''ileri üçüncü sektör'' adını taşıyan başlıbaşına yeni bir sektörün ortaya çıktığı gözlenmektedir. Sanayileşmiş ulusların bu hedefe yöneldikleri düşünüldüğünde, bilgi-işlem etkinliklerinin maliyet bileşkelerinin, toplam maliyet içinde-

ki göreli payları ve önce eğitim, sonra işletme düzeyinde zaman içindeki evrimlerinin ne olduğunu sorgulamak yerinde olur.

Yine OECD ülkeleriyle ilgili tabloda bugünkü tahminlerin, personel giderlerinde net (yani enflasyonun etkisinden arındırılmış) bir artış ile arşivleme, işlem ve iletişim alanlarında, bilgi-işlemle ilgili insanların kullandığı gereçlerin yıllık gerçek fiyatlarında (ücretlerin eşit kaldığını varsaydığımızı unutmadan) ise net bir azalmaya işaret ettiği gösterilmektedir:

Personel	*+ % 8*	
İletişim	*— % 10*	
Hesap	*— % 25*	*önümüzdeki 10 yıl için yıllık olarak*
Bellek	*— % 35*	

Bu ikinci olgu (yani azalma) şaşırtmamalıdır. On yılı aşkın süredir gelişmekte olan minyatürleşme süreci (yani mantıksal işlevler aynı kalmak koşuluyla boyutların küçülmesi) ve orantısal olarak maliyetlerin düşmesi ile belirlenen elektronik teknoloji içinde, her üç malzeme türünün erimekte olduğu düşünülmelidir.

Yine de, teknolojik maliyetlerdeki bu önemli düşüşün nasıl olup da tabloda görüldüğü gibi (OECD ülkelerinde ortalama olarak geçerli olan) iletişim araçlarının fiyatlarındaki bu kadar küçük bir azalmaya tekabül ettiği ya da hatta İtalya'da görüldüğü gibi, artışlara bile yol açtığı sorusu sorulmalıdır.

Görünürdeki bu sapmanın pek çok nedeni bulunmakla birlikte, bunların hemen hepsi, iletişim araçları üzerinde devlet tekelinin geçerli olduğu Avrupa ülkelerinde (ve bu arada İtalya'da) fiyatların yalnızca maliyete göre değil, siyasal değerlendirmeler ve toplumsal gereksinimlere göre de belirlenmesine bağlanabilir.

Ayrıca, yine iletişim araçlarıyla ilgili olarak, elektronik teknolojisinin elektromekanik teknolojiyi henüz pazardan tümüyle uzaklaştıramadığı ve bunun oldukça yüksek işletme giderleri gerektirdiği eklenmelidir. Özellikle İtalya'da iletişim şebeke merkezlerinin % 5'ten azının elektronik türde olduğu tahmin edilmektedir.

Bilgi-işlem için gerekli bileşkelerin maliyetlerindeki değişmelerle ilgili bu basit gözlemlerden çıkarılacak önemli bir sonuç vardır. Daha önce söylediklerimizi hesaba katarak herhangi bir otomasyon (yani elektronik arşivleme, hesap ve iletişim teknolojilerinin kabul görmesi) olmaması halinde, yaklaşık sekiz yıllık bir süre içinde bilgi sisteminin maliyeti, işlenen bilgi hacminin aynı kaldığı varsayılarak, 2,14 kat artacaktır. Başka bir deyişle, sekiz yıl sonra aynı işler (en azından hacim bakımından) iki katından fazla bir maliyetle yapılıyor olacaktır.

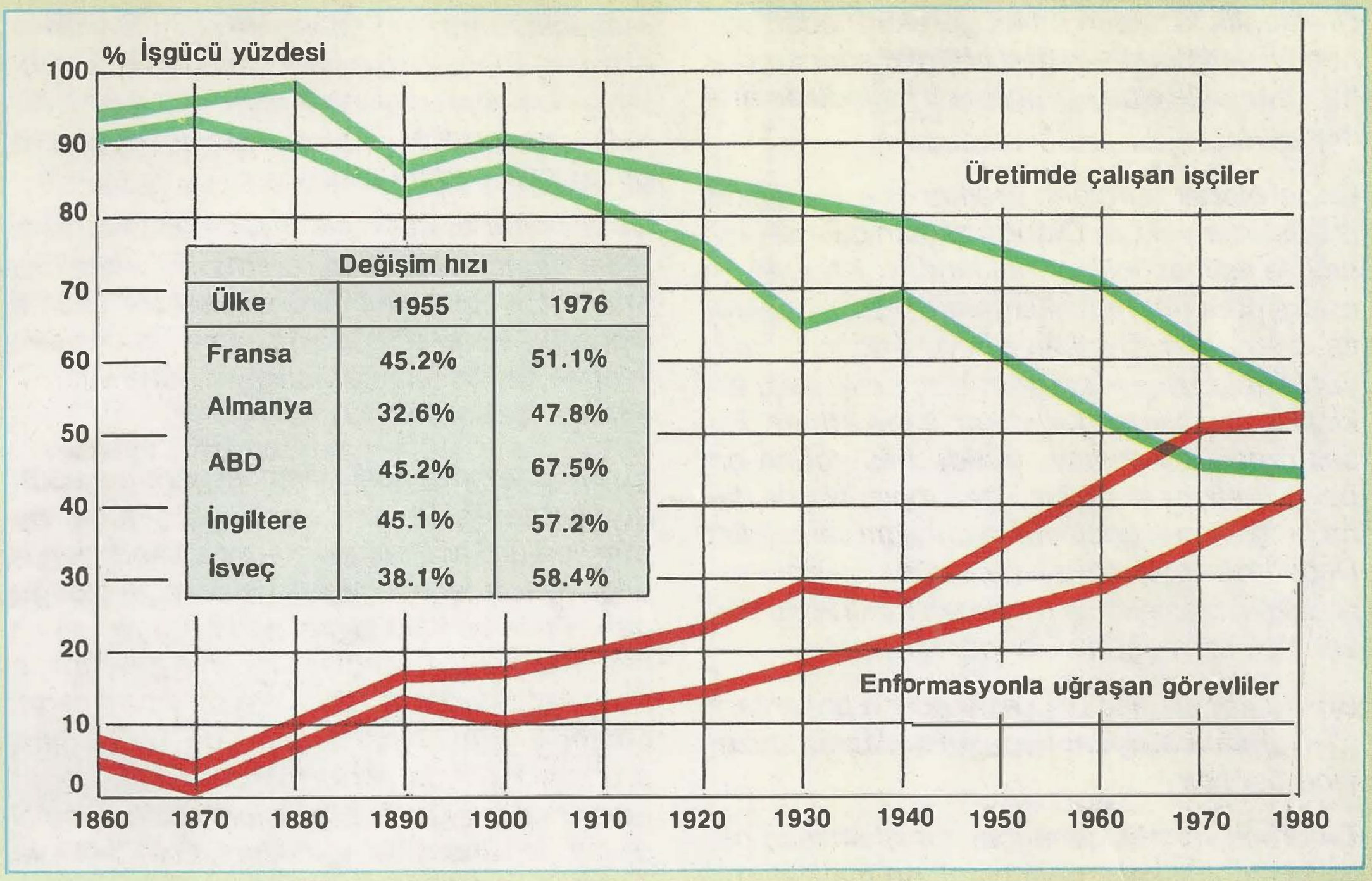
Bunun aksine, kullanılan teknoloji ile enformasyon sisteminin toplam maliyeti arasındaki ilişki olarak tanımlanan bir otomasyon endeksi alınıp, bu endeksin değeri 0.5 olduğunda maliyetlerdeki artış sınırlanarak sıfıra indirgenebilecek ve bu düşme, zaman içinde sabit tutulabilecektir.

% 50'yi aşan otomasyon oranları yardımıyla enformasyon sisteminin maliyeti düşürülerek % 100 otomasyon sınırına varıldığında başlangıçtakinin 00.16'sına eşit bir asgari düzeye ulaşılır.

Şurası açıktır ki, % 100'lük bir otomasyona asla erişilemeyecektir. Bununla birlikte, bazı sanayi kesimlerinde (sürekli üretim yapan işletmelerde) ve bazı işletme alanlarında (anlık üretim yapan çok sayıda işletmenin sipariş-ürün-üretim-satıcı devreleri ya da birçok kredi kuruluşundaki gişe işlemlerinin yönetimi) % 80-90'lık oranlara ulaşılması olanaksız değildir.

İleri sanayileşmiş dünyada enformasyonun oynadığı ve giderek önem kazanan rolü daha önce vurgulamıştık. Bu olgu, işletmelerin yönetimindeki enformasyon gereksinimlerinde, nicel açıdan olduğu kadar, nitel açıdan da katlanan bir artışın görüldüğü son on-onbeş yılda iyice ağırlık kazanmıştır. Önümüzdeki yıllarda daha da hızlanması gereken bu gidişin birçok nedeni vardır. Bunlar, pazarlık uluslararası bir nitelik kazanmasına, yaygın ekonomik ve siyasal istikrarsızlığa, döviz kurlarındaki değişmelere ve geleneksel enerji kaynaklarındaki bunalıma bağlanabilir. Tüm bu nedenlerin, dikkatli biçimde incelendiğinde, şu kavramsal çerçeveye oturtulması mümkündür: ***Belirsizlik durumlarında enformasyon, ekonomik ve/veya toplumsal etkinlik kurmayı amaçlayan herhangi bir örgütü, yönetmeye yarayan en uygun araçtır.***

Enformasyonun alternatifi, beklenmeyen (ve



Değişim hızı		
Ülke	1955	1976
Fransa	45.2%	51.1%
Almanya	32.6%	47.8%
ABD	45.2%	67.5%
İngiltere	45.1%	57.2%
İsveç	38.1%	58.4%

Üretim ve enformasyon sektörlerinde yönelim gelişimi. Her iki eğri, az ya da çok, ''bilgi işlem alanında çalışan görevliler'' kavramının sınırlayıcı tanımlarına karşılık gelmektedir.

genel belirsizlik çerçevesi gözönüne alındığında kestirilemeyen) olayın ortaya çıkmasıyla, ona karşı önlem almak ve onu aşmak için seferber edilebilecek kaynakların (araç-gereç, personel, sermaye) birikimidir. Genellikle bu birikim, bugünkü maliyetler gözönüne alındığında, işletmenin bulunduğu ortama devingen olarak uyum sağlamasını ve belirsizlik koşullarında ondan azami yararı elde etmesini sağlayan yeterince modern ve esnek bir enformasyon sistemini uygulamaya başlamasının gerektireceği harcama ile kıyaslandığında daima ciddi kayıplara yol açabilecek niteliktedir.

Bu düşünce ışığında önceki gözlemler farklı bir değer kazanır. Artık sözkonusu olan yalnızca bilgi-işlem maliyetlerini en aza indirmek değil, daha çok karar verme sürecini yetkinleştirerek, işletmenin rekabet düzeyini koruyup yükseltecek (esas dayanak olarak yalnızca otomasyona yaslanmayacak) bir enformasyon kapasitesinin yaratılmasıdır.

Genel kabul gören tanıma göre, üretkenlik ister nicel, ister nitel (bu ikinci durum için tanımı ayrıca açıklığa kavuşturmak gerekir) belli bir sonuç elde etmek için gerekli emek miktarına eşit bir verimlilik ölçüsüdür.

Üretkenlik fazladan emek gerektirmeden sonuç iyileştiğinde arttığı gibi, aynı sonucu daha az emekle elde edilebilmesi halinde de artmış olur.

Genel olarak büroların üretkenliklerine ilişkin istatistikler yoktur. Olduğu zaman da mal üretimiyle ilgili etkinlikler hakkındaki istatistik ilkelere göre hazırlandıklarından fazla bir anlam ifade etmezler. Üretkenliğin yukarıdaki tanımı, yürütme düzeyindeki yönetsel ve mesleki etkinlikler açısından bir anlam ifade etmez. Bu sektördeki bir memur günde beş yerine on belge üretiyor ya da bir diğeri toplantılarda daha fazla zaman geçiriyorsa daha mı verimlidir? Doğallıkla hayır. Dolayısıyla büro işlerinin verimliliğini ölçmek için tamamen farklı istatistiksel ölçüler geliştirilmesi gereği açıktır.

Bu ölçüler de olsa olsa elde edilen başarılara (performansa), yani hedeflere ulaşma düzeyine bağlıdır.

Bir örnek vermek gerekirse, bir işletmenin pazarlama politikası, hem planların maliyeti, niteliği ve uygunluğuna (yapılabilirliğine), hem de ticari sorumlularla işbirliği içinde gerçekleştirilebilmesine bağlıdır. Gerçekten, yürütme görevlerine ilişkin olarak, işletmedeki bir kişi, bir grup ya da bir işletme kolunun verimliliğinin en mükemmel ölçüsü başarıdır.

Başarı (performans), bir bakıma, üretkenlikten daha geniş bir kavramdır. Performansın artması, yalnızca üretkenliğin artmasıyla sınırlı olmayan, saptanmış işletme hedeflerine uygun son derece etkin sonuçlar alınması anlamına gelir.
İdareci ik işinde sorunu nicel üretkenlikle sınırlayıp basitleştirmek mümkün olmakla birlikte, bu durumda bile üretkenlik kendi başına bir değer olmayıp işletmenin hedeflerine uydurulması gereken bir değerdir. Bununla birlikte, orta-üst mesleki düzeylerin başarılarıyla otomasyon araçlarının kullanımı arasındaki bağlantıyı saptamaya yarayacak az sayıda veriye sahibiz. Bunun nedeni, klasik enformatiğin bugüne kadar yürütme rollerinde çoğu kez doyurucu sonuçlar vermiş olmasıdır.

Bu bağlantı, sürekli yinelenen bir ekonomik kuralı gün ışığına çıkarmaktadır: Bir sektör, kendinden önce gelen sektörün daha üretken kılınması amacıyla otomasyonu için yapılan yatırımların yardımıyla oluşmaktadır. Bu, (kaynak olarak tarım, madencilik, imalat veya hizmet sektörlerinden gelen) değişim mallarının artı değerleriyle gerçekleşmektedir. Bunun yaygınlaşması ve giderek daha çok kaynak tüketmesi, kendi açısından sistemin düşük verimliliğinin nedeni olacaktır.

Bu biçimde buhar makinesi ile önce tarımda (yani birinci gıda sektörünü daha verimli kılmak için) kullanıldı; üçüncü sektör (hizmet sektörü), ikinci sektör olan imalat sektörünün işletme donanımları aracılığıyla daha etkin kılınması için ortaya çıktı.

Şu anda daha verimli kılınması gereken üçüncü sektördür. Bu sektördeki gelişim de, bugün ileri üçüncü sektör olarak adlandırılan ve bilgisayarın tanımı gereği bir parçası olduğu, daha önce belirtildiği gibi enformasyon ve uzmanlıktan oluşan değişim malının geliştirilmesiyle sağlanacaktır. Otomasyon yatırımlarına değindik. Fakat daha önce de belirttiğimiz üretkenlik (ya da performans) ile ilgili değerlendirmeler ışığında bilgi-işlem sözkonusu olduğunda, öncelikle ağırlık verilecek noktalar saptanmalıdır.

Karşı sayfadaki grafik ileri ikinci (bu durumda işçi bileşkesi dışarda tutularak) ya da üçüncü sektörde büro işinin maliyet dağılımını göstermektedir.

Görüleceği gibi, maliyetin üçte birinden fazlası idari görevler üzerinde yoğunlaşırken, yaklaşık üçte ikisi uzmanlık ve yönetim görevlerinde toplanmaktadır.

Bu tespit, derhal daha ileri düşünceleri zorunlu kılmaktadır. İlk olarak, emek maliyetinin payının oldukça düşük olması, yirmi yıllık enformatik yatırımına bağlanmalıdır. Gerçekten de, tarihsel olarak enformatik uygulamaların en çok yoğunlaştığı alanın, sürekli yineleme niteliği taşıyan görevler olduğu bilinmektedir.

Bunun sonucu olarak, emeğin bu tür işlerde toplam maliyet içindeki payı azalmıştır.

İkinci sonuç, idari görevler alanında metin işlemeye yönelik işler, özellikle de sekreterlik işlerinin toplam maliyet içindeki payının % 1'den biraz fazla olmasıdır. Bunun anlamı, eğer büro otomasyonunun amacı üretkenliği yükseltmekse, daktilo yazımıyla ilgili işlere daha uygun çalışmalar yöneltmemek gerektiğidir.

Bunun yerine, enformatiğin, ufak istisnalar dışında, olsa olsa çok sınırlı etki yapabildiği ve

toplam maliyetin yaklaşık üçte ikisini oluşturan uzmanlık ve yönetim etkinlikleri üzerinde durmak gereklidir.

Öte yandan, metin işlemenin, enformasyonun büroda otomasyon sistemine katılmasının bir aracı olarak tartışılmaz bir önemi vardır. Geleneksel veri işlemede veri girişi neyse, daktilo yazımı da büro otomasyonu için odur. Fakat bu, yine belirtilen nedenlerden ötürü, büro otomasyonu için yapılan yatırımlar, daktilo yazımı sistemlerine yapılan yatırımların birbirine karıştırılmasını haklı göstermez. Çünkü daktilo yazımı ile ilgili yatırımlar ya belli bir ortamla sınırlıdır ya da aşılması olanaksız uyuşmazlıklara ve işin, üretkenliği ile verimliliğini mutlak olarak olumsuz etkileyecek şekilde parçalanmasına yol açar.

Büroda otomasyonun öncelikle üzerine eğilmesi gereken işlevler belirlenince, yukarıda anlatıldığı gibi performansın nasıl artırılabileceği sorunu gündeme gelir.

Bu amaçla, bilgi-işlemin özellikleri ile otomasyon koşullarının ayrıntılarına inmek gerekir.

Büro işi şu şekilde bölümlenebilir:

— İşlevler (müşteri muhasebesi, siparişlerin alınması, üretimin denetlenmesi vb.)
— Etkinlikler (bilgi-işlem ve bilgi değiş tokuşu)

İşlevler işletmenin yapısıyla ilgilidir.

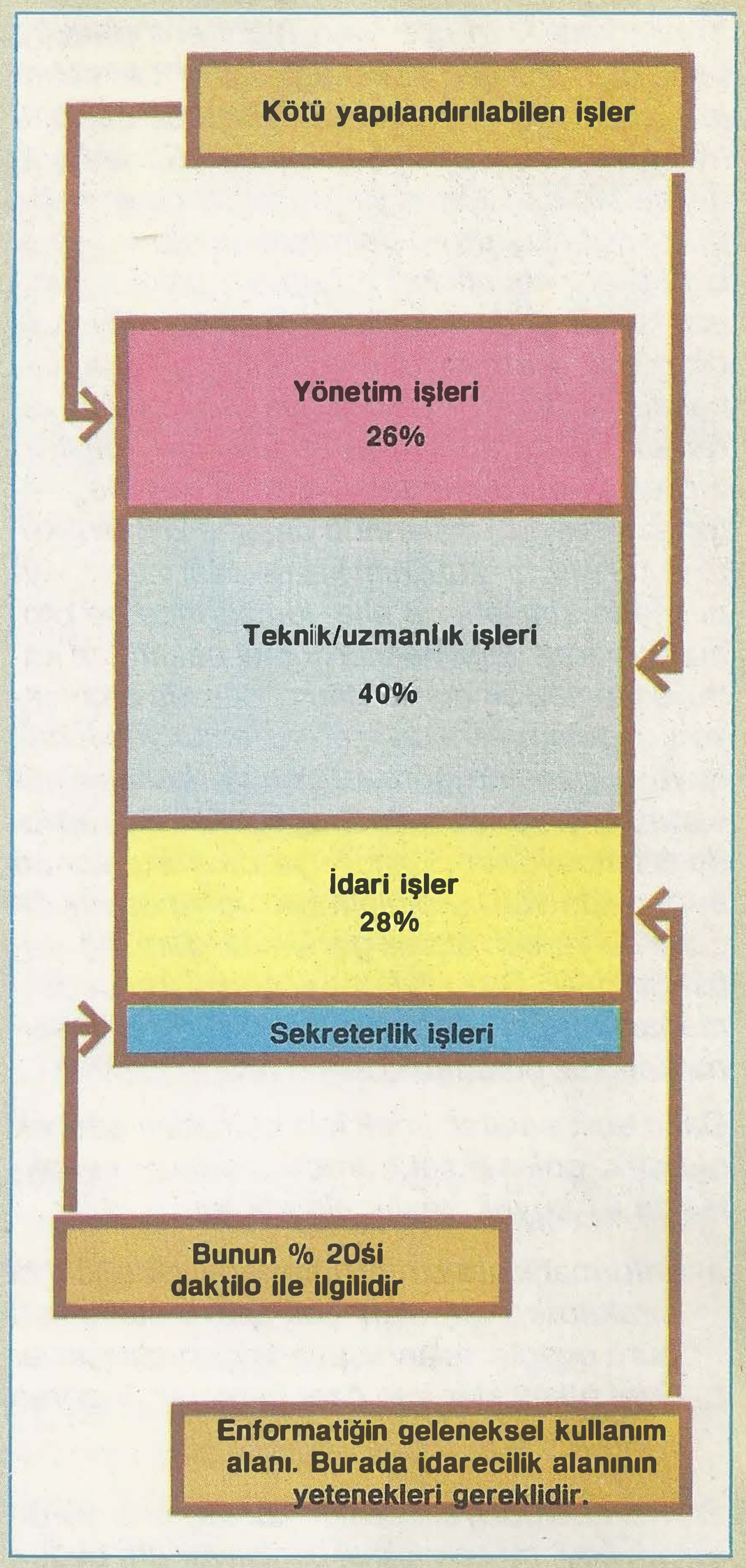
Etkinlikler birden çok işlevle ilişkilidir ve tekil etkinliklerdir.

Office Autcmation ''Büroda Otomasyon'' (OA) etkinliklerde, Data Processing ''Veri İşleme'' (DP) ise işlevlerde otomasyonu öngörür.

Geleneksel enformatik uygulamalarının gelişimi, bu konuya yatkın işletme işlevlerinin (''yapılanabilir'' olarak adlandırılan işlevlerin) ''prosedüre bağlanmasına'' yaradı. Bu şlevler, idare etkinlikleriyle sıkı bir ilişki içinde olan işlevlerdir. Örnek olarak sanayi alanındaki çeşitli muhasebe türleri, siparişlerin yönetimi, üretim, alımlar vb. yönetimi, bankacılık sektöründeki çeşitli hizmetler, küçük cari hesaplar, tahviller, döviz işlemleri sayılabilir.

Enformatik bunları yaparken temel arşivleme, kayıt ve bilgi değişimi etkinliklerini bu işlevlerinb ütünleyici birer parçası olarak görüp, onlarla ilgili prosedürlere ilgili bilgisayar programlarında önemli bir yer verdi. Gerçekte bu etkinlikler, tanımları gereği herhangi bir enformasyon işlevine destek özelliği taşıdıklarından, uygulamadan büyük ölçüde bağımsızdırlar. Bundan dolayı onları prosedürlerden çıkarıp standartlaştırarak yalnızca yürütme uygulamaları için değil, aynı zamanda klasik enformatiğe pek uymayan, yapılanamaz işlerde, her gerektiğinde kullanılır hale getirmek mümkündür. Bu tür etkinliklerde otomasyon, büroda otomasyonun öncelikli hedeflerinden ve onu klasik enformatikten ayıran ögelerden biridir.

Bir başka nokta, otomatikleşen hizmetin yöneldiği kişilerle ilgilidir.



Büro işinin maliyeti

Yapılanmış idari işlevlerin (yani enformatiğe konu olan işlevler) yöneldiği kişiler (hedef kitle), işletmenin örgütsel yapısı içinde çalışma hedefleri ve koşulları bakımından türdeş (yani belli bir idari işlevin yerine getirilmesinde bir prosedür oluşturan) dairelere dahildir. Temel bilgi-işlem etkinlikleri açısından bakıldığında ise, kullanıcı, destek niteliklerinden dolayı belli bir görev grubuna dahil olmakla birlikte, tek başına bir birey de olabilir. Otomasyondan yararlanıp yararlanmamak kendisi için mesleki bir tercih durumundadır. Başka deyişle, enformatik uygulamalarında başarısızlık tehlikesine karşılık bir kullanım kısıtlaması vardır: Aynı işlerin kimilerince elle, kimilerince de otomatik olarak yapılması örgütsel bakımdan kabul edilemez bir durumdadır. Büroda otomasyon uygulamalarında ise, kullanıcı yapılacak iş ve becerisine göre bilgisayar kullanımına karar verip vermemekte serbesttir. Bu, büroda otomasyonun, uzman ya da sorumlunun bugün için yapılandırılamayan işine büyük ölçüde bir yardım olarak görülmesi gerektiği anlamına gelir. Bu haliyle de yöneticilerin performansının nitel yönden geniş ölçüde artmasına katkıda bulunabilir.

Daha açık anlatabilmek için şematize edecek olursak, geleneksel büroda otomasyon yaklaşımı iki büyük başlık altında toplanabilir.

a- Enformatik ara birimi (interface) olduğu gibi bırakarak, var olan çok-işlevli sistemleri büro uygulamalarına uyarlayan yaklaşımlar
b- Özel etkinlikler için özel gereçler öngören yaklaşımlar

Birinci kategoriye girenler küçük, orta ya da büyük boyutlu çok-işlevli bilgisayarlara uygun programların geliştirilmesi suretiyle gerçekleştirilen çözümlerdir.

Bu programlar bir terminal aracılığıyla kullanılabilir, genel metin, enformasyon bürolarında saklanabilir ve gerektiğinde yeniden kullanılabilir. Bu, bir terminale ve bu büroya başvurarak bilgileri alma ve gerektiğinde onları değiştirme yetkisine sahip olan ilgililer için ''elektronik posta'' (electronic mail) aracılığı ile mümkündür.

İşletmenin bilgi prosedürlerinin otomasyonunun da aynı bilgisayar üzerinde gerçekleştirilmesi mümkün olduğundan, bu çözüm türünün üstünlüğü, geleneksel veri işleme ile büro bilgilerinin işlenmesinin birliğini mantıksal olmasa bile fiziksel olarak sağlamasındadır. Yani çözüm, bizim görüşümüze göre (yalnızca bizim görüşümüze göre de değil) idari bilgi sistemini geliştirmeye yönelik her strateji için vazgeçilmez olan iki kullanım alanı arasındaki bütünleyiciliği gerçekleştirmektedir.

Bu eğilimin sınırları, veri işleme sistemleri için karakteristik olan kuralları kabullenmeye yanaşmayan kullanıcılara yaygınlaştırmanın zorluğu tarafından çizilmektedir. Bu kuralların basitleştirilmesi, sistemin genel niteliklerini (gördüğü işleri) sınırlarken, ek kullanım yükü getirilmesi anlamına gelir.

Büroda otomasyonun temel işlevlerinden birinin, bilgisayar alanında uzmanlığı olmayan personele seslenmek olması gerektiği düşünülürse, bu çözümün şimdiye kadar çok dar bir alanda uygulandığı anlaşılır.

İkinci kategoride ise bazı özgül veri işleme etkinliklerini daha etkili kılmayı amaçlayan gereçler topluluğu yer alır. Örneğin elektronik yazı makinesi metin işleme ile ilgili sekreterlik maliyetini düşürür, programlanabilen fotokopi makinesi çoğaltım sürecini rasyonelleştirir, bellekli ve tuşlu telefon numaraların çevrilmesi ve en sık kullanılanların yeniden aranmasını hızlandırır vb. Kendi başına bir anlam taşıyıp genel bir amaca hizmet etmeyen bu uygulama hakkında iki değerlendirme yapılabilir.

Her şeyden önce, bu gereçlerin etkinliğe katkısı, daha öncede vurgulandığı gibi, esas olarak, büro işindeki maliyetler içinde ancak sınırlı bir payı olan sekreterlik hizmetleri (bkz. s.811'deki grafik) alanındadır. Belki bundan da önemli olan nokta, bu katkının, bugün büro işinde verimsizliğin (ve sonuçsuzluğun) başlıca nedenini oluşturan, işlerin parçalara ayrılması sürecini hızlandırarak artırılabilmesidir. Bunun sonucu olarak tek tek etkinliklerin veriminin artmasını sağlayan bu çözüm, görev düzeyindeki işlerin yeniden tanımlanıp bütünleştirilmesi hedefiyle çelişir niteliktedir. Dolayısıyla bu çözüm temelde sağlıklı değildir ve katkıları ancak sınırlıdır.

[G. Occhini'nin **Qaderni d'Informatica,** anno X,n. 1.1983, Honeywell Information Systems Italia'da yayınlanan ''L'informatica nel lavoro d'ufficio'' (''Büro işinde Enformatik'') adlı makalesinden.]

n argümandır (ya da bir argüman dizisidir).

Örneğin, bir sayının mutlak değeri, hemen tüm yerleşim haritası versiyonlarında ABS fonksiyonuyla hesaplanır. Bu durumda sözdizimi şu biçimi alır:

ABS(—7 + 10) doğrudan sayı belirtilerek,
ABS(EM + D3) dolaylı olarak D3 hücresindeki değer belirtile-,-rek,

Fonksiyonların çeşitliliği, kullanılan yerleşim haritası tekniğine bağlı olmakla birlikte, tüm harita versiyonlarında ortak olarak görülen bazı fonksiyon kategorileri vardır.

Matematik Fonksiyonlar

Matematik fonksiyonlar BASIC'de kullanılan fonksiyonlara çok benzer. Tek farklılık, fonksiyonun adından önceki tanıtma simgesidir. Bird eğer ya da bir adresi n ile gösterecek olursak, başlıca matematik fonksiyonlar şunlardır:

ABS(n)	Mutlak değerini alır
EXP(n)	Üs fonksiyonu
INT(n)	Tamsayılı kısım
LN(n)	Doğal logaritma (e tabanında)
LOG(n)	Ondalık logaritma
SQRT(n)	Karekök
COS(n)	Kosinüs
ACOS(n)	Kosinüs yayı (kosinüsün karşıt fonksiyonu)
SIN(n)	Sinüs
ASIN(n)	Sinüs yayı (sinüsün karşıt fonksiyonu)
TAN(n)	Tanjant
ATAN(n)	Tanjant yayı (tanjantın karşıt fonksiyonu)

Kullanılan program türüne göre farklılık gösterdiğini belirttiğimiz tanıtma simgesine bu listede yer vermedik.

Bu fonksiyonların (ki hepsi de tıpkı BASIC diliyle çalışıyormuşcasına kullanılabilir) kullanımı yerleşim haritası tekniği hakkında özel bilgi sahibi olmayı gerektirmez. Örneğin, iki kenar uzunluğu bilinen bir dik üçgenin hipotenüsünü Pisagor teoremine göre hesaplamak için bir program yazmaya çalışalım. Yapılacak işler şunlardır:

— İki kenarın girilmesi
— Hipotenüsün hesaplanması
— Elde edilen değerin bilgiyazımı

BASIC'te aynı program şöyle yapılanabilir:

```
10 INPUT ''A KENARI''; A
20 INPUT ''B KENARI''; B
30 V = SQR (A ∧ 2 + B ∧ 2)
40 PRINT ''HİPOTENÜS-''; N
50 END
```

Aynı program, BASIC güdümlerini bilmeden, yerleşim haritası tekniğiyle şöyle yazılabilir:

A1, A2, A3 hücrelerine 10, 20, 40 güdümlerinde verilen bilgiler girilir.

B1 ve B2 hücrelerinde (A ve B değişkenleriyle ilgili) veriler ve B3 hücresinde hesaplama formülü yer alır.

Son hücrede aynı zamanda hesabın sonucu da görülecektir. Dolayısıyla bu tek hücre, BASIC programındaki 30 ve 40. satırların yerini almış olur.

Özel Sayısal Fonksiyonlar

Yerleşim haritası tekniği esas olarak, gider tahminleri, faizler, yatırım planları, model kurulması gibi ekonomik veya mali hesap alanlarında uygulanmaktadır. Bu özel sorunları çözmek için BASIC'te bulunmayan (en azından fonksiyon adı altında var olmayan) bir dizi fonksiyon geliştirilmiştir.

Bu fonksiyonların başlıcaları şöyle sıralanabilir:

AVERAGE (n1, n2,...) Parametre olarak belirlenen hücrelerin içeriklerinin ortalama değerini hesaplar.

Örneğin;

AVERAGE (+B1, +B2, +C6)

B1, B3 ve C6'nın (ki tümü de 3'e bölünür) içeriklerinin ortalamasını verir.

Birçok hücrenin ortalama değerini hesaplamak gerektiğinde, tümünün birden belirtilmesi güçleşir.

Bunun için bir dizi değeri **(kaplam-range)** aynı anda ifade etmeye yaranan bir simgeleme vardır. Böyle bir kaplamın kullanılabilmesi verilerin uygun biçimde yerleştirilmesine bağlıdır.

Önceki örneğimizdeki (B1,B3,C6) dağınık yerleştirme (ya sütun ya satır atlanmaktadır), kaplam kullanımını önler.

Bunun için tümü aynı satırda ya da aynı sütunda yer alan verileri kullanmak gerekir. Ancak bazı durumlarda bu verilerin birbirini izleyen birden fazla satır ya da sütuna yayılması mümkündür. Örneğimize dönersek, C6'nın içeriği B2'ye kaydırılınca veriler B1, B2, B3 konumuna gelir, yani aynı sütun (B) üzerinde

sıralanır. Bu durumda, B1'den B3'e dek adresleri kapsayan bir kaplam oluşturmak mümkündür. Bu kaplamı tanımlamak için en çok kullanılan sözdizimi şöyledir:

B1.....B3

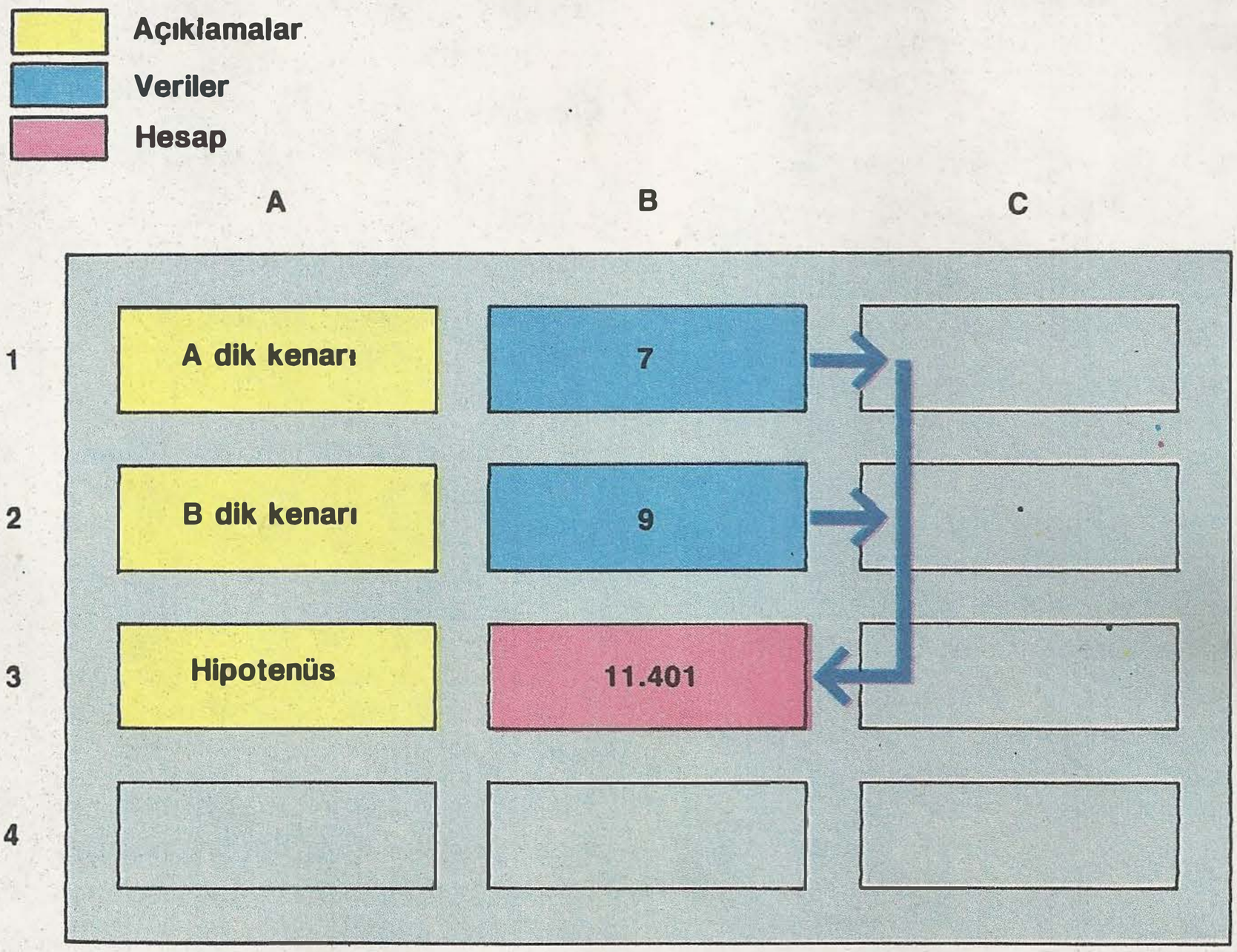
Burada ilk değer kaplamın başlangıcı,.. simgeleri ''ya dek'' son değer de kaplamın sonu anlamına gelir.

Bu simgelemede ortalama değer hesabının fonksiyonu şöyle olur:

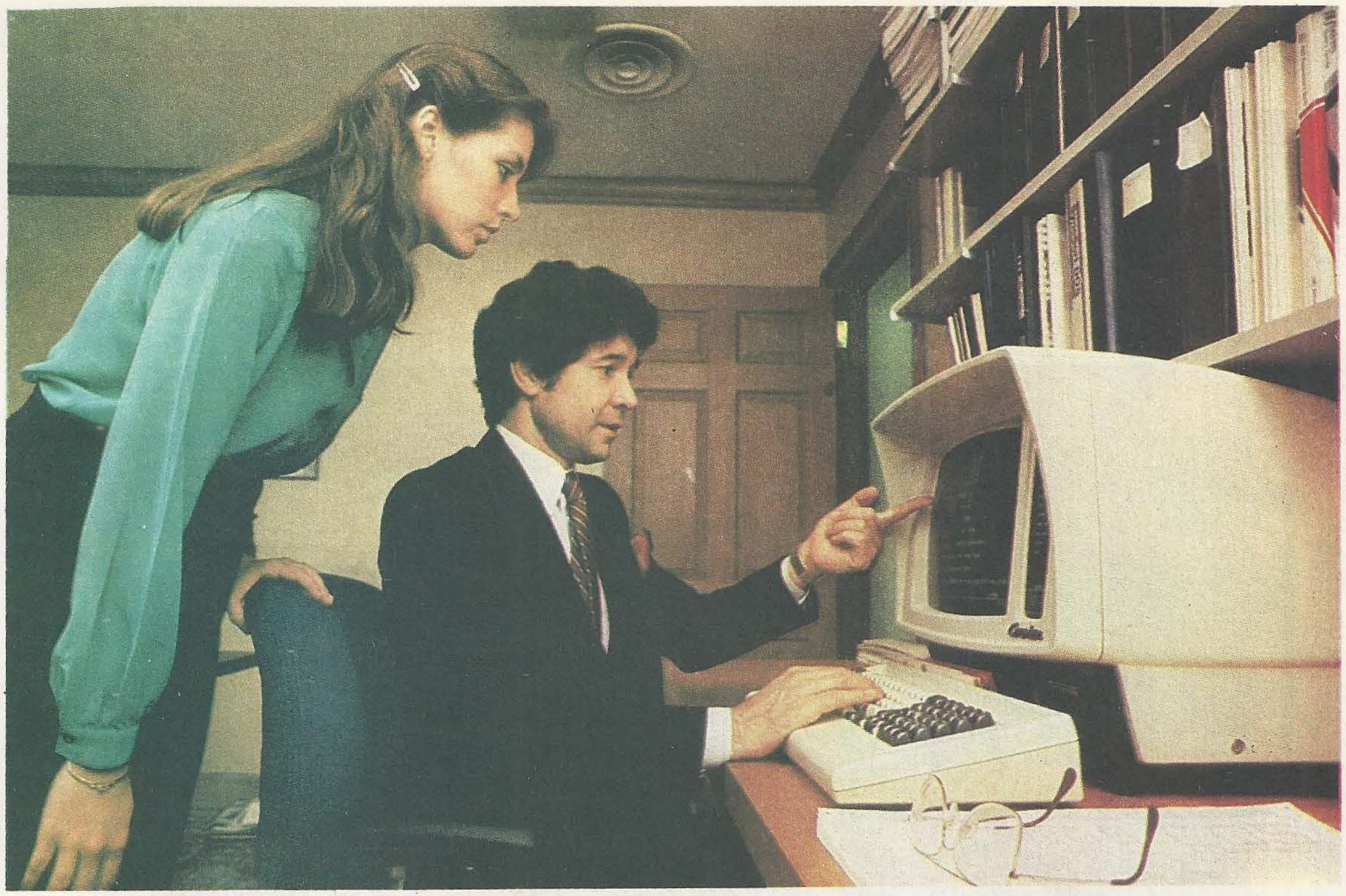
AVERAGE (B1....B3)

Kaplam kullanımı bir başka önemli yarar daha sağlar. Bir fonksiyon tanımlarken (çıkış ve varış) hücrelerinin adreslerini doğrudan yazmak gerekmez: İmlecin ilgili hücrenin üzerine getirilmesi yeterlidir. Bu durumda sistem doğru adresi yazar. Adresleri kullanırken bir önlem almak gerekir: Sistemin mutlak mı, yoksa göreli biçimde mi çalıştığına dikkat edilmelidir. Herhangi bir hücreyi, örneğin B8'i, ele almak gerektiğinde, normal olarak yerleşim haritasında B sütunu ile 8. satırın keşismesinde yer alan hücre anlatılmak istenir. Bu doğrudan (ya da mutlak) yöntemdir. Bunun dışın-

YERLEŞİM HARİTASI TEKNİĞİNİN UYGULAMA ÖRNEĞİ



B1 ve B2 hücreleri, verileri almak için öngörülmüştür (Dikkenarların değerleri).
B3 hücresine hipotenüsün hesaplanması için gerekli formül girilir:
@ SQRT (+B 1 ∧ 2 + B2 ∧ 2)



Veri işleme sistemlerinin kullanılması büro işinin rasyonelleşmesine büyük katkılar sağlamaktadır.

da göreli olarak adlandırılan, ikinci bir adresleme yöntemi daha vardır: İmlecin yerleşim haritası üzerinde B9'da bulunduğunu varsayalım. Göreli adreslemeyle B8'e yerleşimini işaret etmek demek, aynı sütunda (B), bir üst satırda (satır sayısı 9'dan 8'e) bulunan hücreye gönderme yapmak demektir.

Yine buna benzer biçimde, imleç B9'da iken, C10'u işaret etmek demek bir sağdaki sütunda (B'den C'ye) ve bir alt satırdaki (9'dan 10'a) hücreye gönderme yapmak demektir.

B9 hücresinde B8 ve C8'in toplamlarının (+B8+C8) hesaplanması için bir formül bulunduğunu varsayalım. Mutlak adreslemede, B9 hücresinin içeriğinin başka bir yere (örneğin E3'e) kaydırılması halinde hesap değişmez. Sistem, verileri belirtilen yerlerden almaktadır. Bunun aksine, göreli adresleme durumunda yanlış sonuçlar elde edilir. Sistem önceki yerlere değil (yani B8 ve C8), bunların B9'a göre durumu neyse E3'e göre aynı konumda bulunan yeni hücrelere göndermede bulunduğundan, formül değişikliğe uğrar. İki yöntem arasındaki bu farklılık sayfa 816'daki grafikte gösterilmektedir.

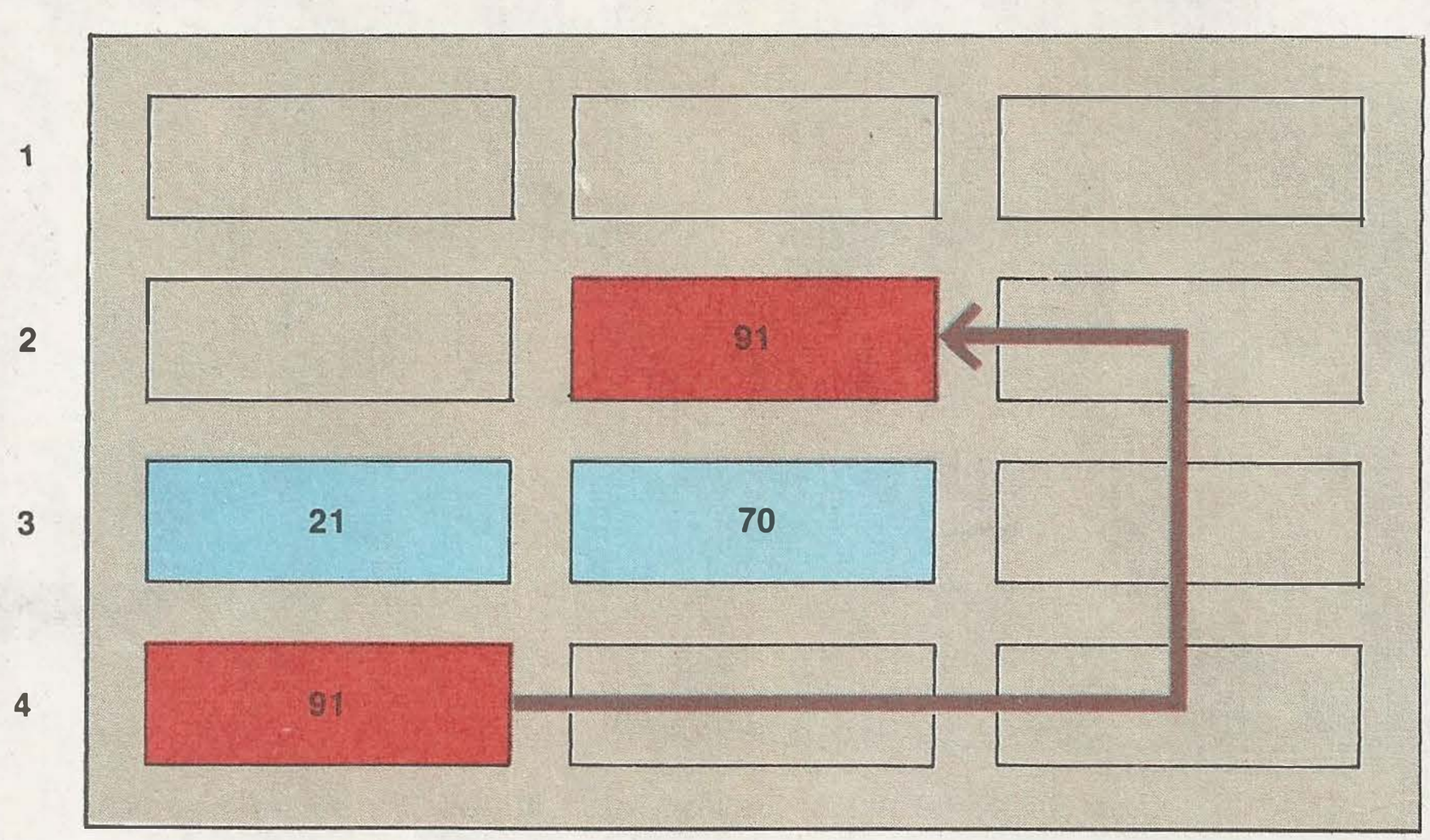
MUTLAK VE GÖRECELİ ADRESLEME

Formülü içeren hücre

Geçersiz verileri içeren hücre

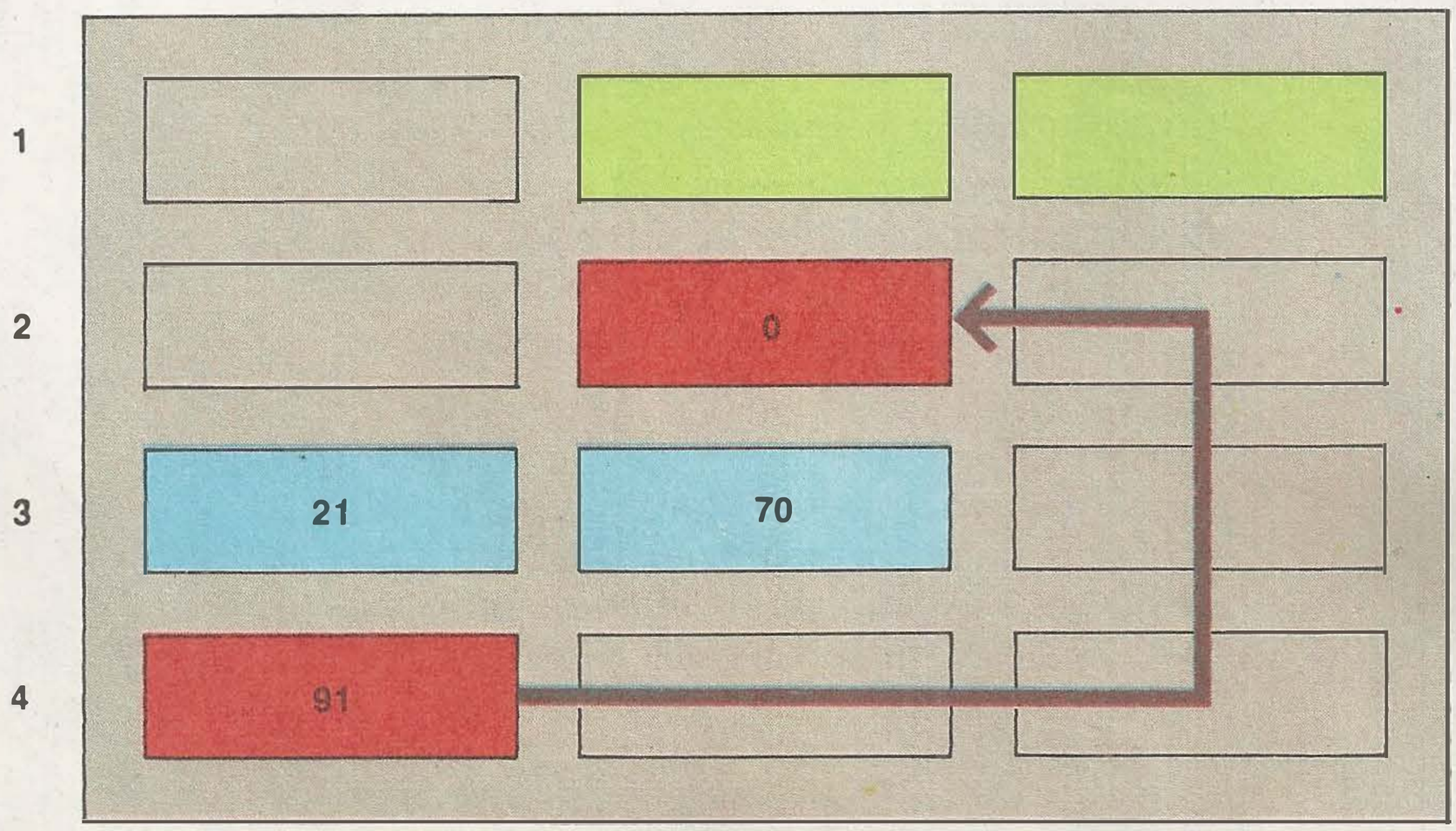
Geçerli verileri içeren

Mutlak adresleme



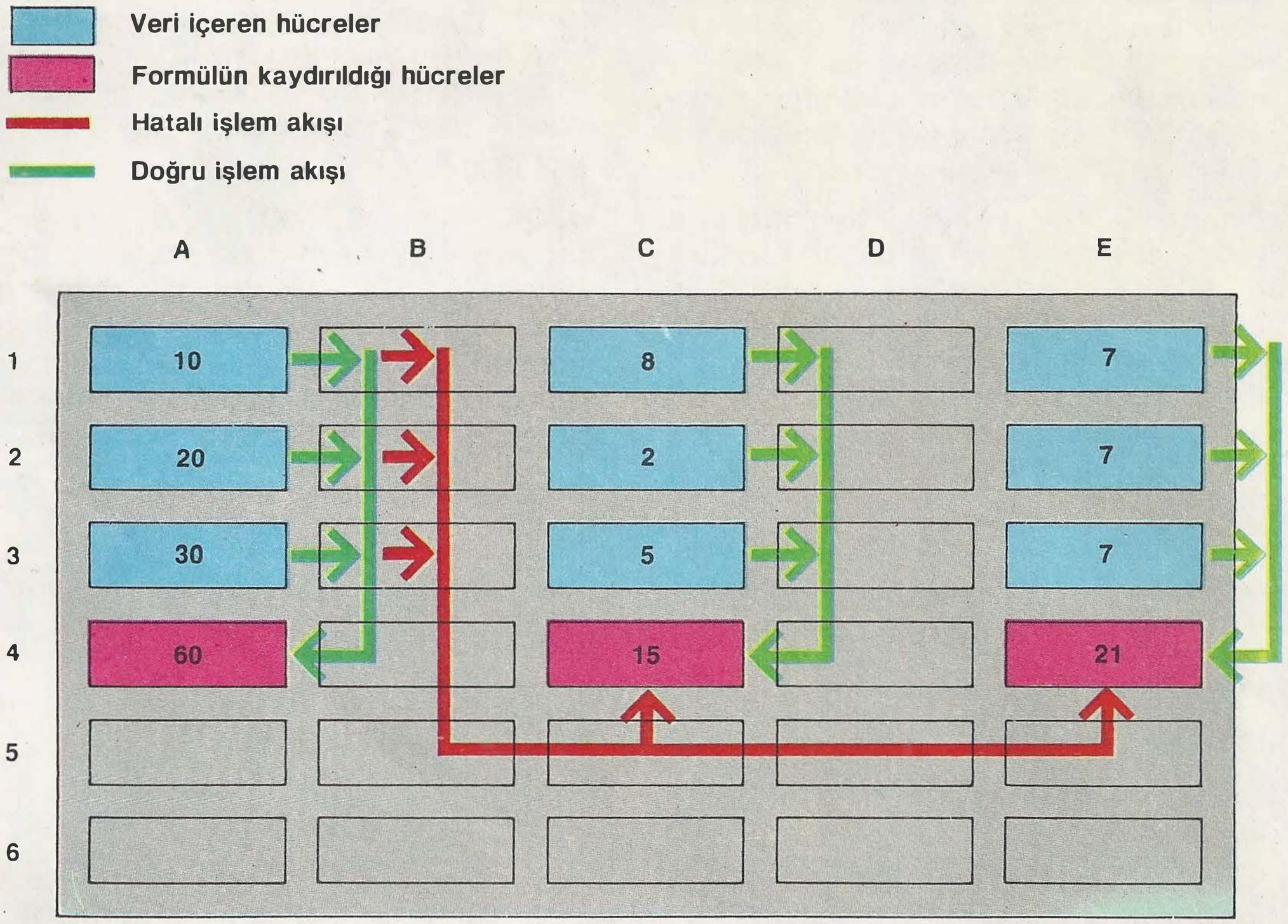
A4 hücresi +A3+B3 formülünü içerir ve A3 ve B3 hücrelerinin içeriklerinin toplamını verir. Formül bir başka hücreye yerleştirilirse, hesaplamaya ilişkin hücrelerin koordinatlarının aynı kalması nedeniyle, sonuç aynı kalır.

Göreli adresleme



Sistem göreli adreslemeyi kullanıyorsa, hesaplamanın B2 hücresine kaydırılması yanlış bir sonuca neden olur. Sistem, kaydırmadan sonra formülü +B1+C1'e dönüştürülmüştür, çünkü B1 ve C1 hücreleri B2'ye göre A3 ve B3 hücrelerinin A4'e göre bulunduğu konumla aynı göreli konumdadırlar.

GÖRELİ ADRESLEMENİN SÜTUNLARA GÖRE TOPLAMAYA UYGULANMASI



A4 hücresi, A sütunundaki verilerin toplamını görüntülemeye yarayan +A1+A2+A3 formülünü içerir. Sistem, A4'ün içeriğini önce C4'e, ardından E4'e kaydırarak göreli adreslemeyi kullanırsa, formül otomatik olarak kendini bu duruma uyarlar ve sonuç yine C ve E sütunlarındaki verilerin toplamına eşit olur.

Her şeye rağmen göreli adreslemenin yadsınarnaz bir yararı vardır.

Örneğin, bu sayfada üstte, sütunlara göre bazı toplamalar yaparken kullanılacak yerleşim haritasını ele alalım. Göreli adresleme kullanılarak, A4 hücresi C4 ve E4'e kaydırılırsa içerik (formül) yeni duruma uyum sağlar ve hesap bu nedenle geçerliliğini korur. Mutlak adresleme ile C4 ve E4 hücrelerinde, A4 hücresindeki formülün aynısı bulunacak, fakat sonuçlar yanlış çıkacaktı.

COUNT (NS..NE). NS ve NE adreslerinin belirlediği alanda (range) bulunan dolu hücrelerin sayısını verir. Sayfa 818'de stok mallarının sayımının yapılmasında COUNT fonksiyonunun kullanımıyla ilgili bir örnek verilmektedir. Yeni bir malın kaydı için ayrılan 4. satır bu malın açıklaması girilmedikçe hesaba katılmaz.

SUM (NS..NE). Arka sayfadaki örnekte çeşitli satırlarda görülen değerlerin toplamını almak için SUM fonksiyonu kullanılmıştır. Bu fonksiyonun kullanımında, sonuç derhal elde edilir: SUM (C2..C5), +C2+C3+C4+C5 anlamına gelir. COUNT(...) ve SUM(...) fonksiyonlarının birlikte kullanılmasıyla, AVERAGE(...) fonksiyonunun ortaya çıktığına dikkat ediniz:

$$\text{AVERAGE (A1..A9)} = \frac{\text{SUM(A1..A9)}}{\text{COUNT(A1..A9)}}$$

MAX (NS..NE), MIN (NS..NE). Belirlenen

alanda sırayla azami (en büyük) ve asgari (en küçük) değerleri bulur.

NPV (S, NS .. NE). Net Mevcut Değer (Net Present Value) anlamına gelen bu fonksiyon, belli bir faiz oranı ile sonuçta istenen miktarı elde etmek için gerekli başlangıç sermayesini hesap eder. S parametresi faiz oranını gösterir. NS..NE ise, dönem sonunda elde edilmek istenen değerin basamaklarıdır.

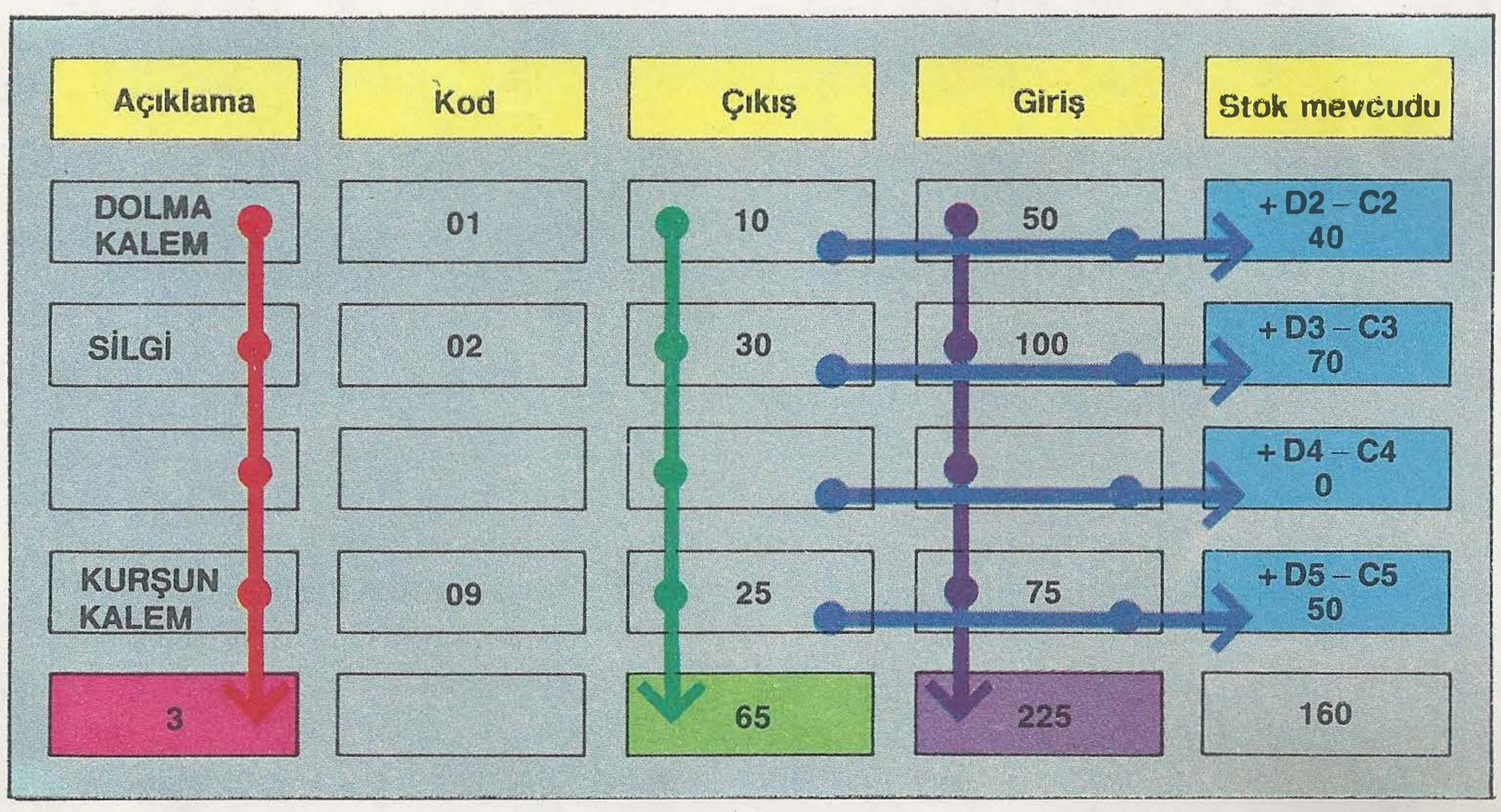
Örneğin % 20'lik (S = 0.2) bir faiz oranı ile bir tek dönemde 1500'e eşit bir para miktarı (başlangıçtaki miktar artı faiz) elde edilmek isteniyorsa, 1250 tutarında bir sermaye yatırmak gerekir. Gerçekten bu miktar % 20'lik faiz oranı ile 250 getirir, bunu da başlangıç sermayesiyle toplayınca 1500 elde edilir.

Arama Fonksiyonları

Veriler arasında seçme yapma olanağı veren tüm fonksiyonlar, bu gruba girer. Bu fonksiyonların çeşitliliği, kullanılan yerleşim haritasının türüyle yakından ilişkilidir. Bu ara başlık altında, yalnızca tüm programlarda aynı olan en önemli fonksiyonu, LOOKUP'ı açıklamak istiyoruz.

LOOKUP (M, NS.. NE). NS..NE aralığında M değerini arar. Bulunan parametre, kullanılan yerleşim haritasının türüne göre değişir. Bazı durumlarda (örneğin Multiplan'da) M'nin bulunduğu sütun ya da satırın son hücresindeki değeri gösterir. Diğerlerinde, örneğin (Visicalc'de) M'nin bulunduğu ögenin hemen so-

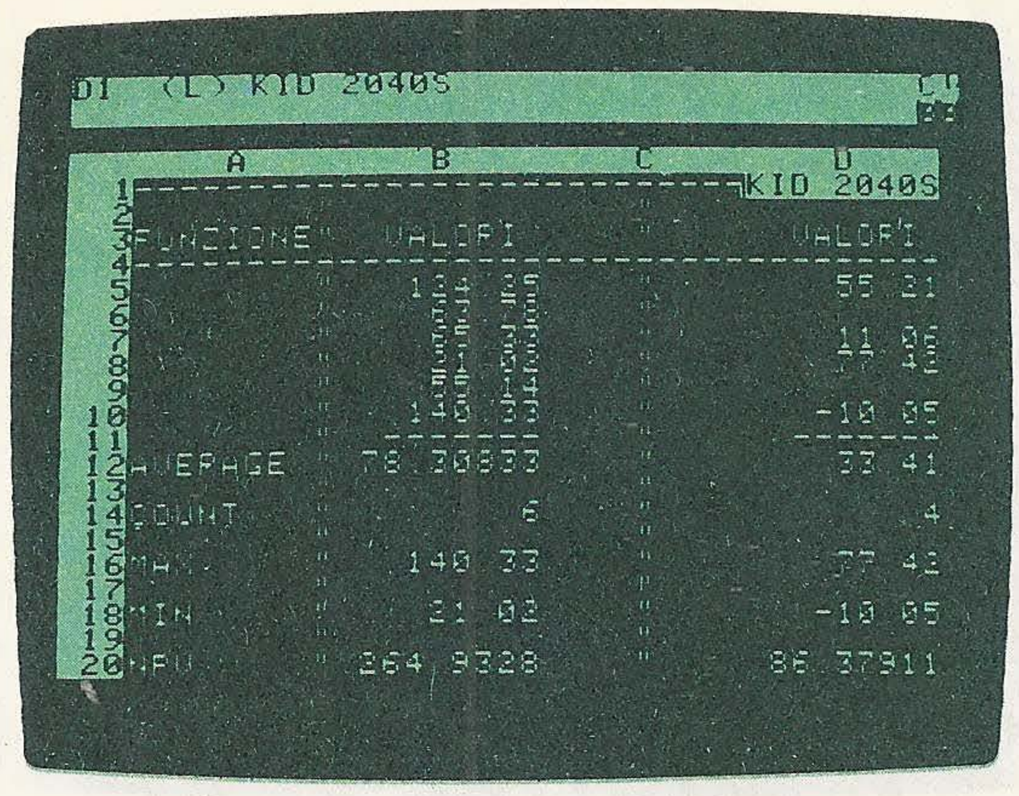
STOK HESABINDA COUNT VE SUM FONKSİYONLARININ KULLANIMLARI

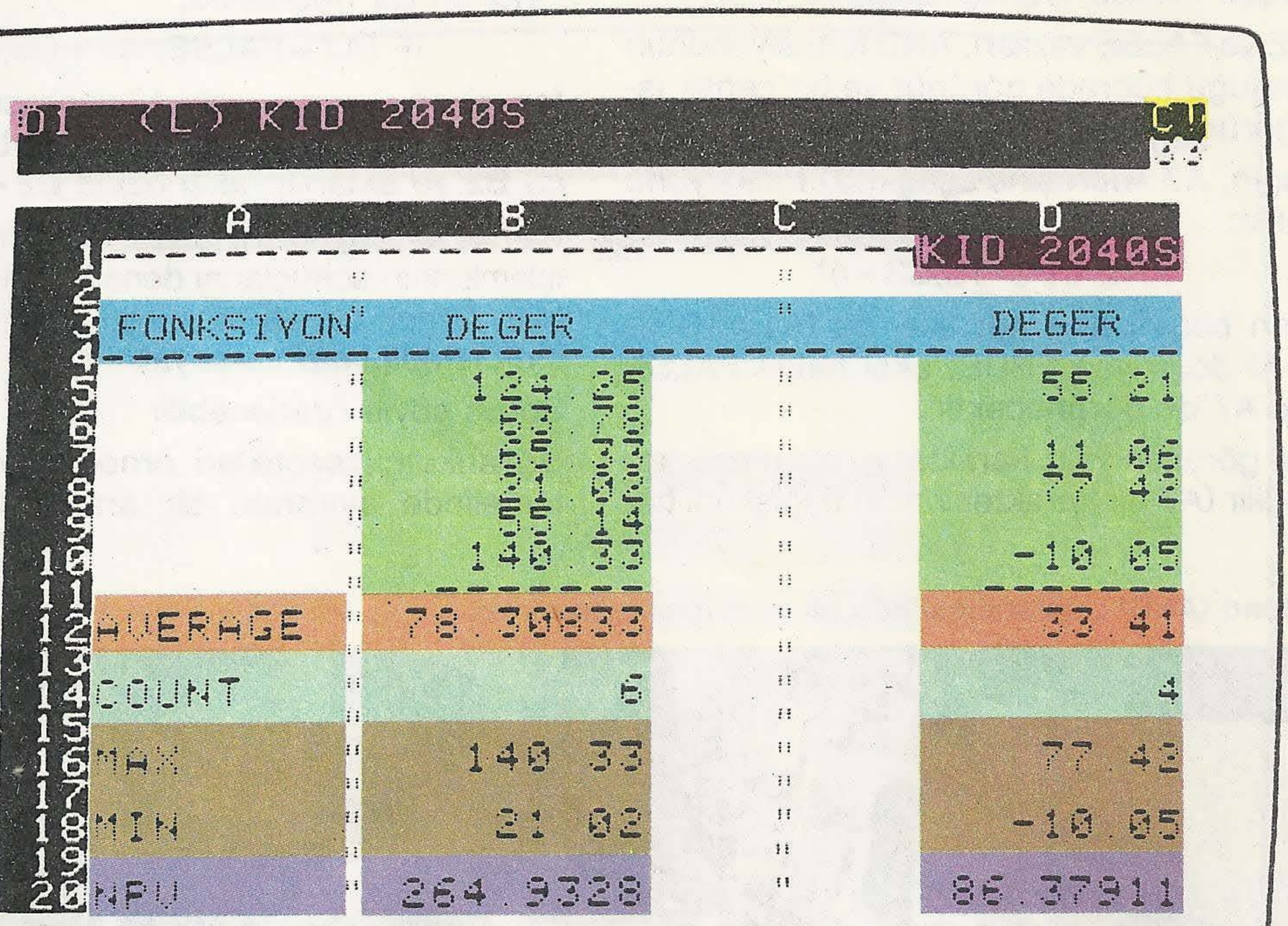


Girilen hücre	Fonksiyon	Görünen
A6	@COUNT (A2..A5)	Çeşitli depolarda bulunan malların sayısı
C6	@SUM (C2..C5)	Toplam çıkış miktarı
D6	@SUM (D2..D5)	Toplam giriş miktarı
E6	@SUM (E2..E5)	Toplam depo mevcutu

YERLEŞİM HARİTASININ FONKSİYONLARI

Ekran, tüm yerleşim haritalarında aynı olan bazı fonksiyonların hesaplanmasını göstermektedir.





Sütunlara göre hesaplama istenmektedir.

D1 hücresi, kullanılan gerecin (SIPREL KID 2040S) adını içermektedir. Burada belirtilen fonksiyonlar, başka sistemlerde aynı işlevi görseler bile farklı adlara sahip olabilirler.

3. satır, sütunların açıklamalarını içermektedir.

B ve D sütunlarının hücreleri, fonksiyonların hesaplanması için kullanılan verileri içermektedir.

AVERAGE fonksiyonu, uygulandığı değerlerini ortalamasını verir. Böylece, B sütunundaki değerlerin ortalaması B12'de, D sütunundaki değerlerin ortalaması da D12'de görüntülenir.

COUNT fonksiyonu, sayısal değerlerin her iki sütunda kapladığı hücrelerini sayısını verir. Boş hücrelerin gözönüne alınmaması nedeniyle, D sütununda değerin 4 olduğuna dikkat edilmelidir.

MAX ve MIN fonksiyonları, iki sütunun her birinde bulunan azami ve asgari değerleri hesaplarlar.

NPV fonksiyonu, belirli bir faiz oranı üzerinden belirli bir nihai sermayeyi elde etmek için gereken başlangıç sermayesini hesaplar.

lunda ya da üzerindeki sütun ya da satırın değeridir.

Mantık Fonksiyonları

Yerleşim haritalarında seçmeler yapmak ya da yapılan bazı işlemlerin yürütülmesini koşula bağlamak için kullanılabilecek mantık fonksiyonları öngörülmüştür. Bu fonksiyonların kullanımı, karşılık gelen BASIC komutlarınınkine benzer. Mantık fonksiyonlarının çeşitliliği, kullanılan yerleşim haritasına bağlıdır. Aşağıda, en çok kullanılan fonksiyonlar açıklanmaktadır.

AND (ifade listesi). Listede sayılan koşulların tümü yerine gelmişse TRUE (doğru), gelmemişse FALSE (yanlış) değerini verir.

TRUE ve FALSE yazıları, AND fonksiyonunun bulunduğu hücrede görünür ve bir çentik işlevi görürler.

Örneğin, A7 hücresine aşağıdaki fonksiyonu koyalım:

AND (A1>10,C3=5)

Verilen bağıntıların her ikisi de (A1>10 ve C3=5) doğruysa TRUE, aksi halde FALSE yazısı A7'de görünecektir.

A7'de görünen yazı, karakter zinciri olarak kabul edilir (A7 bir karakter zinciri değişkeni hafini alır) ve başka fonksiyonlarda kullanılabilir. Örneğin A11 hücresine,

AND (A7=TRUE, A9=FALSE)

fonksiyonunu yazarsak, sözkonusu koşulları elde etmiş oluruz. Bu koşulların kendisi de başka koşullara, A7 ve A9 hücrelerinde içerilen değerlere bağlı kılınabilirler.

821. sayfada AND fonksiyonunun uygulanmasına ilişkin bir örnek bulunmaktadır.

IF (koşul, doğru, yanlış). BASIC'teki IF'in anlamını taşır. ''Koşul'' denetlendiğinde, gerçekleşmiş olup (''doğru'') olmadığına (''yanlış'') göre, ''doğru'' ya da ''yanlış'' alanlarına girilmiş olanlar yerine getirilir.

Örneğin B2 hücresine,

IF (A7=FALSE, A1+C3,0)

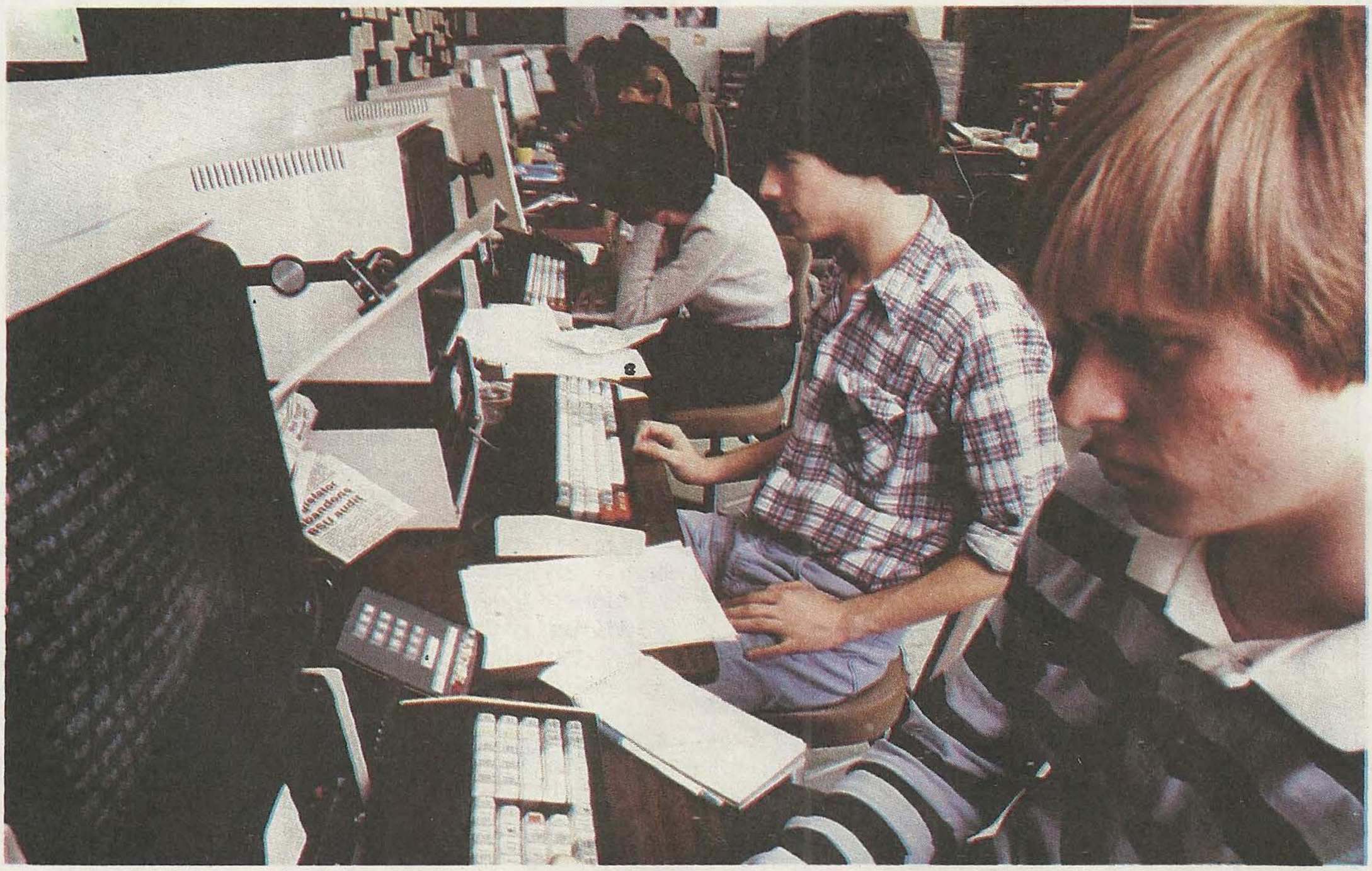
fonksiyonunu yazarsak, A7 yanlışsa (yani, koşul doğrulanıyorsa) A1+C3 hesabının sonucu B2'ye aktarılır, aksi halde B2=0 olur.

Mantık fonksiyonları, girilen verilerin ya da veri işlemlerinin sonuçlarını denetlemek için yararlıdır. Örnek olarak bazı istatistik verileri işlemek istediğimizi varsayalım.

Sorun şöyle özetlenebilir:

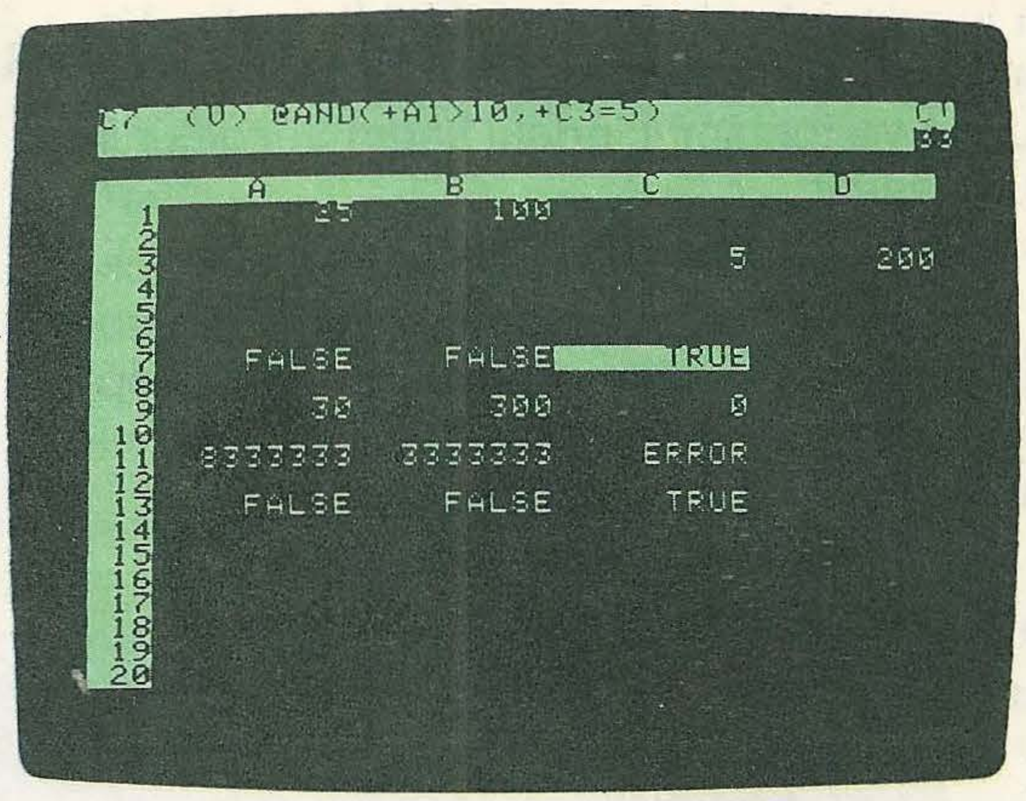
Üç anketçi, seçtikleri örneklemeyi cinsiyet temelinde ayırarak, bir araştırma yapmış-

Maryland (ABD) Üniversitesi'nde bir veri işleme merkezi.



AND FONKSİYONUNUN KULLANIMI

Ekran, AND fonksiyonunun kullanan birkaç mantıksal ifade girilmiş olan bir yerleşim haritasını göstermektedir.



C7 hücresine @AND (+A1>10, +C3=5) formülü girilmiştir.

A1=25 ve C3=5 olması nedeniyle, cevap TRUE (doğru) olur. TRUE, C7 hücresinin başlangıçtaki içeriğinin (mantıksal fonksiyon) yerini almaz. Yalnızca görüntü halinde olan ve verilerdeki değişmelere göre ilgili hücrelerin içeriklerini güncelleştirmeye yarayan bir çentik sözkonusudur.

C11 hücresinde ERROR çentiğinin kullanımı gösterilmektedir. C11'e, C9'un içeriğini bölen olarak kullanan bir formül girilmiştir. Sistem, bir 0'a bölme olduğunu farkeder ve bir hata iletisi verir.

C9'a 0'dan farklı bir değer girildiğinde, hata koşulu otomatik olarak ortadan kalkar ve sistem, hesabın sonucunu görüntüler.

tır. Her biri anket yapılan kişi sayısını ve bu sayı içinde erkeklerin payını bildirmiştir. Yapılmak istenen, incelenen her üç insan topluluğunda ve toplam içindeki kadın yüzdesinin bulunmasıdır.

Haritanın hazırlanmasında hiçbir güçlük yoktur. Hesap komutları arasına bir yönetim komutu sokmak yeterlidir. Kadınların sayısı, toplamdan erkeklerin sayısının çıkarılmasıyla elde edilir:

Kadınlar = Toplam—Erkekler

Kadınların toplam içindeki yüzdesi şöyle bulunur:

$$P = \frac{\text{Kadınlar}}{\text{Toplam}} * 100$$

$$= \frac{\text{Toplam—Erkekler}}{\text{Toplam}} * 100$$

Bir örneklemede yalnızca erkeklerle karşılaşılmışsa, bu durumda kadınların sayısı ve yüzdesi 0'dır. Böylece IF fonksiyonu kullanılarak hesap yapmaya gerek kalmayabilir (gerçekte bu yolun izlenmesi için bir neden yoktur; tablo yalnızca örnek olarak düzenlenmiştir).

Hesapları yapabilmek için yerleşim haritasına verilecek biçim karşı sayfada gösterilmektedir. Veriler B2, B4, B6 (toplamlar) ile C2, C4, C6 (erkek sayısı) hücrelerine girilir; D sütununun karşılık gelen hücrelerinde kadınların yüzdesi, satır 8'de ise genel toplamlar hesaplanır.

İstenen verilerin elde edilmesi, kadın yüzdelerin hesaplanması için ayrılan hücrelerin (D sütunu), ilgili giriş hücrelerinde (B ve C sütunları) bulunan değerlere bağımlı kılınmasıyla sağlanır. Örneğimizde IF fonksiyonu D8 hücresinin içeriğini hesaplamak üzere kullanılmaktadır. Diğer hücrelerde formül biçimsel yönden aynıdır, yalnızca adresler değişir. Örneğin D4 hücresi (2. anketçi) için formül şöyledir:

IF (C4 <> B4,100* ((B4—C4)/B4),0)

NOT (koşul). Koşul yerine geliyorsa FALSE'ı, aksi halde TRUE'yu verir. Örneğin,

NOT (A3 = C6)

fonksiyonunun bulunduğu hücrede A3 = C6 ise FALSE'ı A3 = C6 ise TRUE'yu verecektir. Bu durumda da hücrenin içeriği (FALSE, TRUE) başka hesapları koşullamak için çentik olarak kullanılabilir.

ISNA (hücre). Belirlenen hücrede NA (Not Available-namevcut) fonksiyonunun varlığı ya da yokluğuna göre TRUE ya da FALSE değerini alır.

NA fonksiyonu (önünde hiçbir argüman olmaksızın) henüz kullanılmayan ve dolayısıyla anlamı olmayan hücrelere gönderme yapan hesapları bloke etmede kullanılabilecek bir çentiktir.

Örneğin, A1 bir veri girişi hücresi olsun. Bir hesabı A1'de herhangi bir değerin varlığı ya da yokluğu ile koşullamak için, A1@NA yazılarak, NA fonksiyonu hücreye yerleştirilir (@ simgesinin yalnızca kullanılan sistemde bir anlam ifade ettiği, diğer sistemlerde farklı olabileceği unutulmamalıdır). Diğer bir hücreye (örneğin B1'e) ise ISNA(A1) fonksiyonu yazılır. A1'de verilerin varlığı ile koşullanması gereken tüm hesaplar B1'deki TRUE/FALSE koşulu denetlenerek yazılmalıdır. Örneğin aşağıdaki biçimde:

IF (B1 = FALSE,.....)

A1 hücresinde NA göründüğü sürece, B1'in içeriği TRUE olur. A1'e herhangi bir veri girildiğinde B1 hücresine FALSE aktarılır ve bu çentiğin kullanıldığı tüm hücrelerde hesaplamalar yerine getirilir.

ISERROR (hücre). Belirlenen bir hücrede bir hatanın ortaya çıkıp çıkmadığını denetler. Örneğin;

ISERROR(B6)

B6 hücresinde bir hata görülmüşse TRUE, aksi halde FALSE verir.

Hatanın ortaya çıktığı hücrede ERROR yazısı göründüğünden, çalışan kişi normal olmayan bir durumdan haberdar olur. Bu fonksiyon çok sayıda hücrenin etkin olarak kullanıldığı uygulamalarda kullanılır. Bu durumlarda, çalışılırken haritanın ancak bir bölümü görülebildiğinden, görüntüde olmayan hücrelerden birindeki olası bir hata görülemez. ISERROR (.....) fonksiyonunun kullanılması, haritanın görüntüde olmayan bölgelerinin de denetim altına alınmasını sağlar.

OR (koşullar). BASIC'teki OR ile aynı anlamı taşır. Örneğin, A3 hücresine,

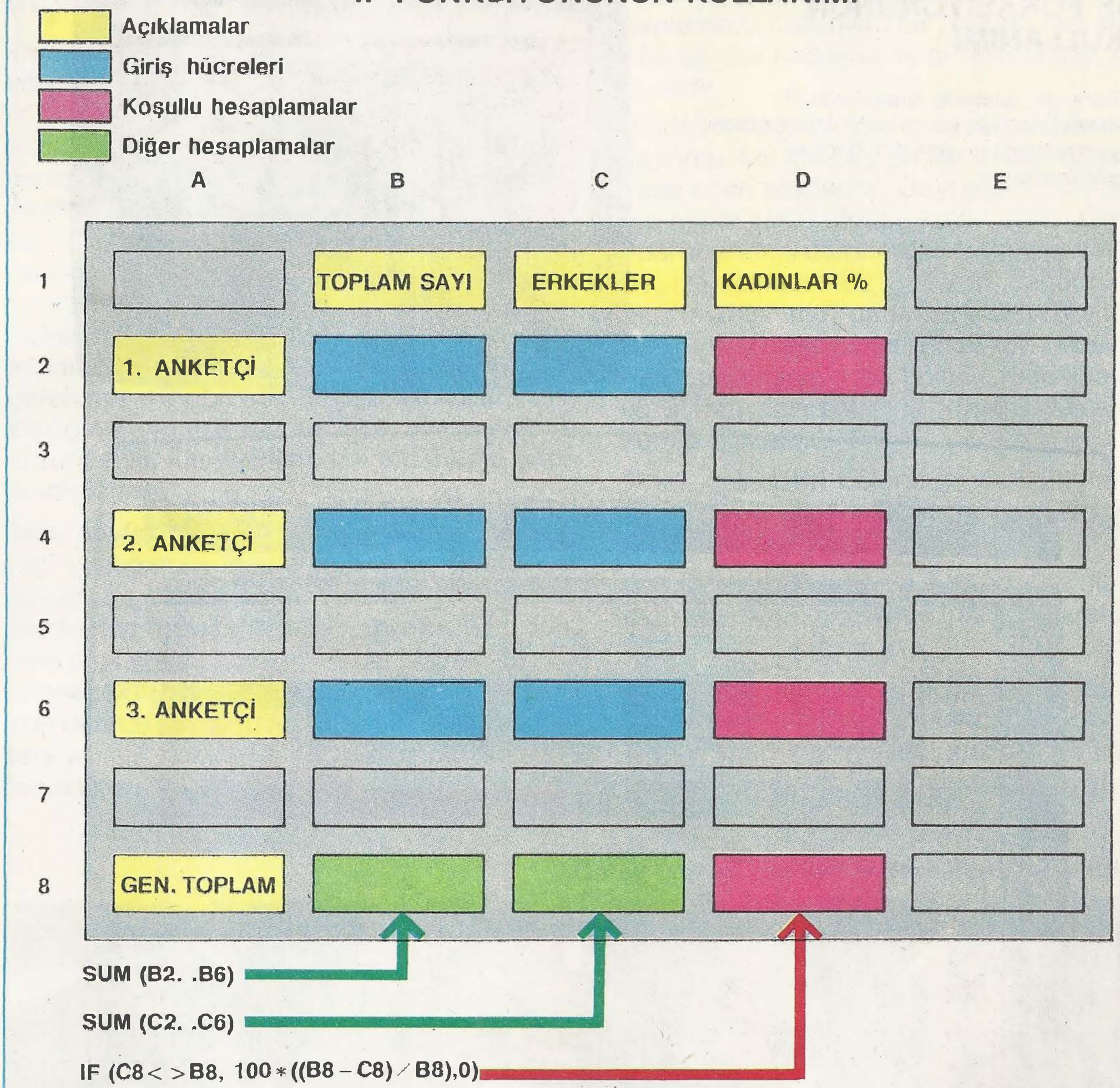
OR (A1 = 15, C3>D2, F4 <> 0)

ifadesini yazarsak sayılan koşullardan herhangi birinin gerçekleşmesi halinde A3 = TRUE olur.

Argümansız Fonksiyonlar

Yerleşim haritalarında kullanılan fonksiyonların bazıları sayısal ya da mantıksal sabit de-

IF FONKSİYONUNUN KULLANIMI



ğerler verir. Dolayısıyla bir argümanları yoktur. Bunların başlıcaları şöyle sayılabilir:

PI. 3,1415926536 değerini (pi'yi) verir.

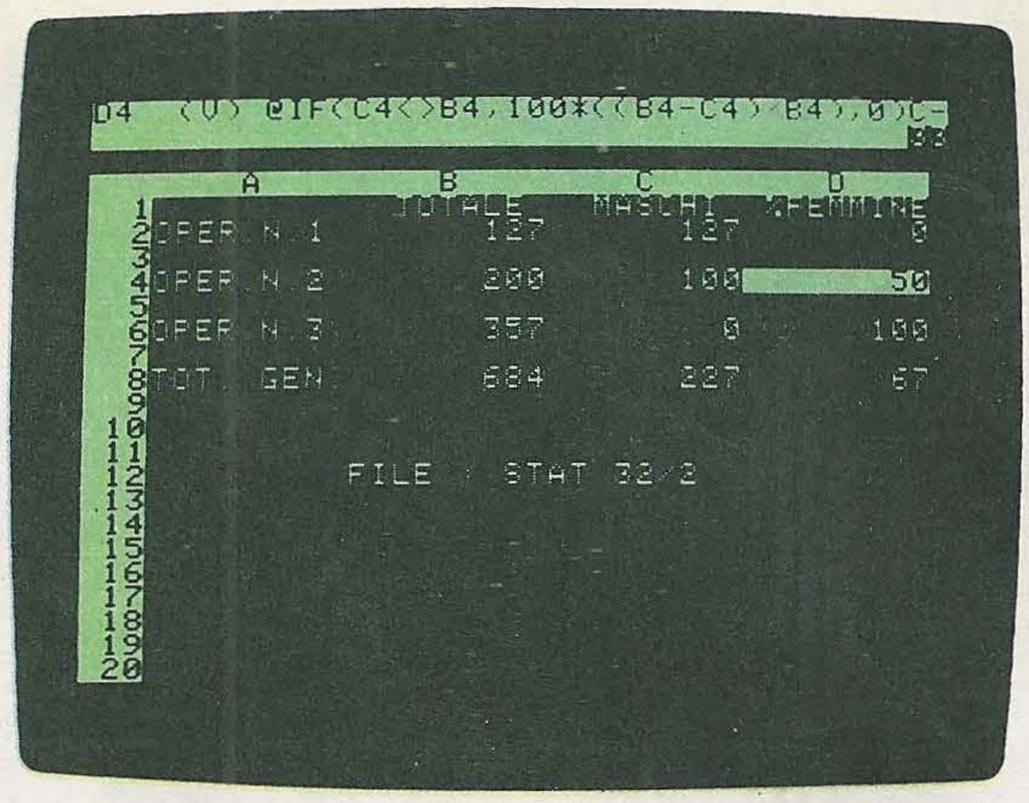
ERROR. Bu fonksiyona gönderme yapan tüm hücrelere, içlerinde yazı biçiminde görünen ERROR çentiğini iletir.

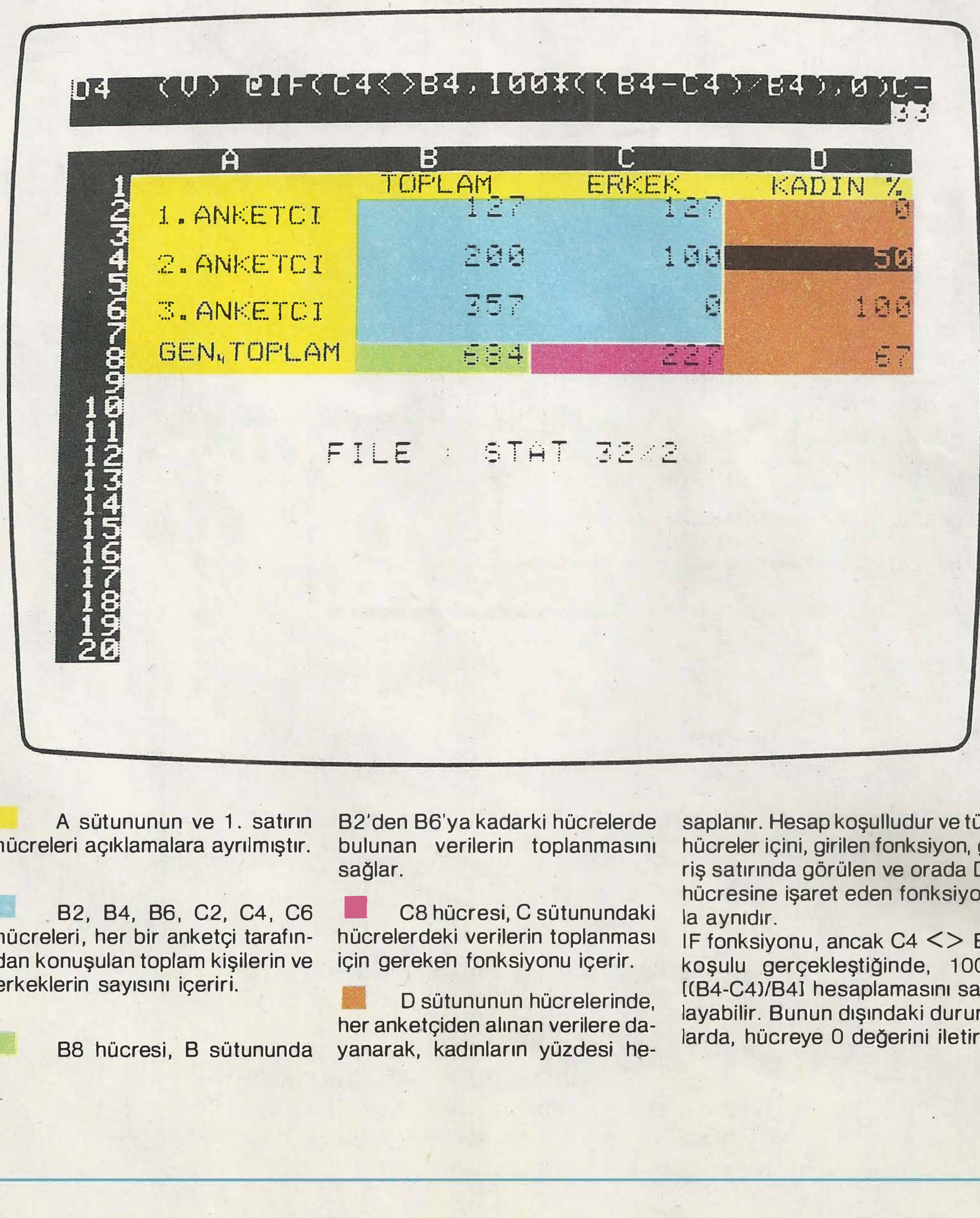
FALSE/TRUE. Karşılık gelen çentikleri üretirler.

NA. NA fonksiyonunu içeren hücreye gönderme yapan tüm hücrelerdeki hesapları bloke eder (durdurur). Başlangıçta NA'nın bulunduğu hücreye yeni bir veri girilene dek, bu hücrelerde de NA yazısı görülür.

IF FONKSİYONUNUN KULLANIMI

Ekranda, istatistik hesaplarda IF fonksiyonunun kullanımına örnek olarak yararlandığımız yerleşim haritası görülmektedir.





A sütununun ve 1. satırın hücreleri açıklamalara ayrılmıştır.

B2, B4, B6, C2, C4, C6 hücreleri, her bir anketçi tarafından konuşulan toplam kişilerin ve erkeklerin sayısını içeriri.

B8 hücresi, B sütununda B2'den B6'ya kadarki hücrelerde bulunan verilerin toplanmasını sağlar.

C8 hücresi, C sütunundaki hücrelerdeki verilerin toplanması için gereken fonksiyonu içerir.

D sütununun hücrelerinde, her anketçiden alınan verilere dayanarak, kadınların yüzdesi hesaplanır. Hesap koşulludur ve tüm hücreler içini, girilen fonksiyon, giriş satırında görülen ve orada D4 hücresine işaret eden fonksiyonla aynıdır.
IF fonksiyonu, ancak C4 <> B4 koşulu gerçekleştiğinde, 100*[(B4-C4)/B4] hesaplamasını sağlayabilir. Bunun dışındaki durumlarda, hücreye 0 değerini iletir.

Yerleşim Haritası Tekniğine İlişkin Uygulama Örnekleri

Yerleşim haritası tekniği, muhasebe ile ilgili sorunların çözümlerini en kestirme ve anlaşılır biçimde verdiğinden, tüm işletme uygulaması paketlerinde (Visicalc, Multiplan, Lotus vb.) geniş ölçüde kullanılmaktadır. Haritanın yapısı sayesinde kullanılan hesap yönteminin genel bir çerçevesi her an kullanıcının gözü önünde bulunmakta ve birçok bakımdan geleneksel muhasebe defterlerini çağrıştırmaktadır. Kullanıcının önceden belirlenen birim ve şemalara uyum sağlamak için çaba harcaması gerekmez. Her prosedür doğal ve kullanıcının kişiliğine uygun biçimde, tipik programlama sorunlarıyla karşılaşılmadan tasarlanıp yapılandırılabilir.

İlerki sayfalarda üç tipik yerleşim haritası tekniği uygulaması incelenmektedir. Birinci uygulamada bir apartmanın ortak giderlerinin bindeli (üç haneli) dağılımı yapılmaktadır; ikincide stok yönetiminde haritanın yapılanışı sergilenecektir. Son uygulamada ticari pratikte son derece yaygın olan WHAT IF adlı, tahminlere yönelik çözümleme yönteminden söz edilecektir.

Apartman Giderlerinin Bölünmesi

Ortak bir gayrımenkulün işletme giderlerini hesaplamada kullanılan bir yöntem, her bir mülkün bindeli katsayısına orantılı olarak bölünmesidir.

Bindeli katsayılar, her bireysel mülkün toplam gayrımenkul içindeki payını (binde cinsinden) ifade eden sayılardır. Gayrımenkulün değerinin binde 1000 olduğu kabul edilir. Örneğin, apartmandaki dairelerden birinin payının binde 100 olması, bu dairenin değerinin tüm apartmanınkinin onda birine (100/1000 = 1/10 = %10) eşit olması demektir: Dairenin sahibi, apartmanın ortak bölümlerinin tüm bakım ve onarım giderlerinin % 10'unu ödemekle yükümlü olacaktır.

Ortak giderlerin bölünmesiyle ilgili haritanın yapısı 826. sayfadaki şematik cetvelde görülmektedir.

A sütununda (haritanın A sütununu oluşturacak) apartmanın daire numaraları sıralanmıştır.

B,C,D sütunlarında farklı giderlerin farklı bindeli bölünme katsayıları olacağı öngörülerek her bir dairenin üç farklı harcama türü için, bindeli katsayıları yer almaktadır. Bu yerleşim

Gereç üretimi için iletken levhaların birleştirilmesi.



BİNDELİ TABLOLAR ARACILIĞIYLA GİDER DAĞILIMI

Formüller

Geri dönmeyen girişler

Fonksiyonlar

Geri dönen girişler

	A	B	C	D	E	F	G	H
1		BİNDELİ DAĞILIMLAR			TUTARLAR			TOPLAM
2								
3	Ev numarası	TAB.1	TAB.2	TAB. 3	TAB. 1	TAB. 2	TAB. 3	
4	1	250	65	120	+B15*B4/1000	+B16*C4/1000	+B17*D4/1000	@SUM(E4..G4)
5	2	100	35	80	+B15*B5/1000	+B16*C5/1000	+B17*D5/1000	@SUM(E5..G5)
6	3	50	290	250	+B15*B6/1000	+B16*C6/1000	+B17*D6/1000	@SUM(E6..G6)
7	4	300	410	150	+B15*B7/1000	+B16*C7/1000	+B17*D7/1000	@SUM(E7..G7)
8	6	100	100	300	+B15*B8/1000	+B16*C8/1000	+B17*D8/1000	@SUM(E8..G8)
9	10	200	100	100	+B15*B9/1000	+B16*C9/1000	+B17*D9/1000	@SUM(E9..G9)
10								
11								
12								
13								
14								
15	1. Çıkış	500						
16	2. Çıkış	360						
17	3. Çıkış	2500						

haritasında üç farklı bölünme öngörülmekle birlikte, bu sayı istenildiği kadar artırılabilir. Haritanın ikinci bölümünde (E,F,G sütunları) her bir dairenin belli bir gidere katılma payının hesabı görülmektedir. Sons ütunda ise, üç ayrı kalemin toplamından oluşan, her dairenin ödeyeceği toplam miktar yer almaktadır.

Her giderin girişi 15, 16 ve 17. satırlarda yapılmaktadır. İlk gider tablo 1'e (B sütunu), ikinci tablo 2'ye (C sütunu) vb. gönderme yapar.

Her daireye düşen gider payı, toplam rakamla bu giderle ilgili tablo bindeleri çarpımının 1000'e bölünmesiyle elde edilir.

Büroda Otomasyon (Bürokratik) (2)

Buraya kadar geliştirilen çözümlemenin ışığında, büroda otomasyon kavramından aşağıdakilerin anlaşılabileceği açığa çıkmıştır:

- Müdürler, yöneticiler ve idareden sorumlu görevlilere doğrudan destek, bilgi ve hizmet sağlamak üzere, otomatik veri işleme ve iletme araçlarının koordine kullanımına dayalı teknik-örgütsel model.
- Bu model, şunları içerir:
  Metin işleme
  Elektronik posta
  Arşivlerden (kütüklerden) yararlanma
  Hesap (muhasebe)
  Karar verme sürecine destek
  Kişisel işlerin görülmesine destek (ajanda, borç-alacak defteri vb.)
  Olası diğer özel uygulamalar
- Hizmetlerin, entegre bir arabirimi (interface) tarafından, başka ara gereçlere gerek kalmadan iletilebilir olması gerekir.

Bu tanım, daha net bazı ifadelere gerek göstermektedir. Birkaç yıl öncesinin aksine günümüz teknolojisi, bu modelin ekonomik bakımdan elverişli bir biçimde uygulanmasını mümkün kılmaktadır. Burada sözkonusu olan, büro işini ve çalışanların iş alanını ilgilendirmesi nedeniyle, öncelikle örgütsel bir modeldir. Dolayısıyla modele ilişkin uygulamalarda, kaynaştırma ve uyumun öncelikli bir anlam taşıması zorunludur.

Çeşitli otomasyon hizmetlerinin amacı, daha önce anlatılmıştı. Amaç tarafından belirlenen işlevlerin önceliği, bunların büro işinin toplam maliyeti üzerindeki etkilerine bağlıdır.

Bu işlevler, yalnızca temel bilgi işlem etkinliklerini değil, daha gelişkin bazı etkinlikleri de kapsar. Bunların otomatikleştirilmeleri etkisini nihai olarak bireysel alanda da gösterir, çünkü bunlar eninde sonunda kullanıcının iş alanını ve kişisel tarzını dönüştürecektir.

Ayrıca kütüklerde bulunan verilerin çözümlenme ve hesaplanmaları için basit yöntemler de sözkonusudur. Bu yöntemler, karar alma sürecine destek olma, işin planlanması ve kişisel işlerin yerine getirilmesi için planlanır.

Yukarıdaki değerlendirmeler ışığında, çeşitli hizmetlerin yerine getirilmesi için, veri işleme alanı ile büro alanı arasında anlamsız bir engel oluşturmayan, standartlaşmış bir arabirimin (interface) zorunluluğu açıkça ortaya çıkmaktadır.

Ayrıca kişilerin bu hizmetlerden hiçbir aracıya gerek kalmadan doğrudan yararlanabilmesi önemlidir. Bu da, büro enformatiğini klasik enformatikten farklılaştıran bir başka ögedir. Yapılanamayan bir etkinliğe destek olacak bilgi araştırmasının, tanımı gereği devredilemeyeceği (zira bilgilerin seçimi, işin görülmesiyle ilgili bireysel koşullarla yakından ilişkilidir) olgusu üzerinde biraz düşününce, bu zorunluluk kolayca anlaşılır. Başka biçimde söylersek, eldeki bilgiler arasında yararlı olacakları seçecek ve olası bilgi eksikliklerini telafi edebilecek olan kişi, aynı zamanda hangi bilgilere (ve niçin) gereksinim duyulduğunu bilen kişidir.

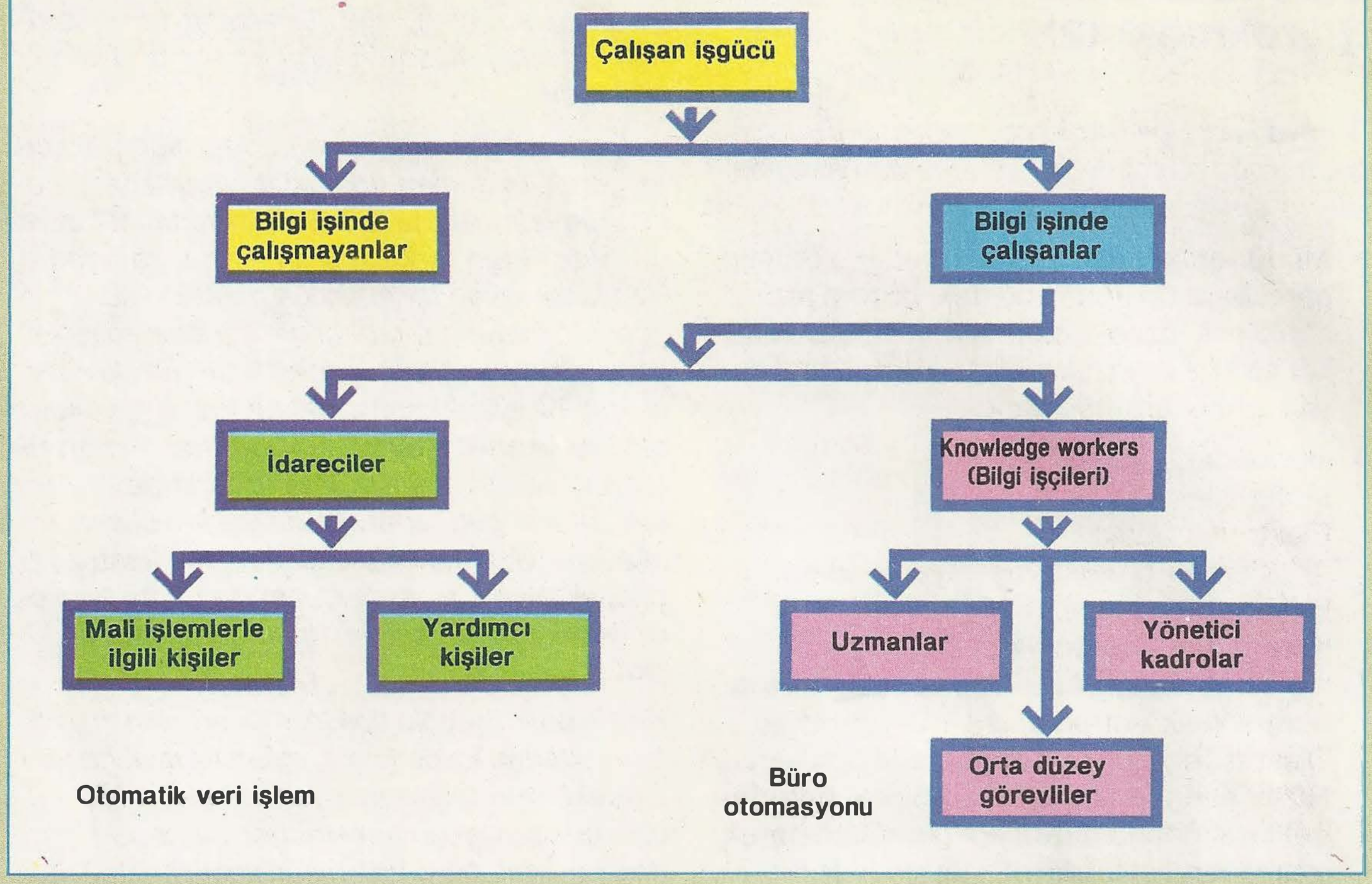
Klasik enformatiğin tipik özellikleri olan ara rollerin ortadan kaldırılması, insan ile makine arasındaki ilişki tarzlarının radikal biçimde yeniden gözden geçirilmesini gerektirir. Bu gözden geçirme olayı halen sürmekle birlikte, kuramsal araştırma açısından bile katedilmesi gereken uzun bir yol vardır. Özellikle dilbilimden başka, yapay zeka olarak bilinen enformatik koluyla bilişsel psikoloji bu konuda ideal bir uygulama alanı bulacaktır.

Büroda otomasyon ve rasyonelleştirme sürecinin, işletmenin rekabet şansını koruyabilmek için neden vazgeçilmez bir tercih olmaya başladığını ortaya koyduktan ve klasik enformatikle benzerlik ve farklılıklarını inceledikten sonra, şimdi de büroda otomasyonun planlanmasıyla ilgili bazı ögelerin üzerinde durmak istiyoruz.

Daha önce de belirttiğimiz gibi, uygulamalarla ilgili özelliklerin yanı sıra, yönetici kadroların üzerinde titizlikle duracakları inancıyla, ekonomik ve mali özellikleri de hesaba katacağız. Yönetici kadroların duyarlılığının başlıca nedeni, orta ve uzun dönemde yapılması gereken yatırımın büyüklüğüdür. Kısa dönemde dar kapsamlı bir denemeyle yetinilirse, maliyet çok düşük olabilir.

828. sayfadaki şemada daha önce yapılan değerlendirmeler yinelenerek, geniş anlamda büro otomasyonunun uygulanması gereken işlevler gösterilmektedir.

Bilgilerin işlenmesiyle ilgili işlerde çalışan kişiler, idareciler ve işletmelerde gerek uzman-



EDV ve OA (Office Automation = Büro Otomasyonu) arasındaki ilişkiler ve bölümlenme.

lık, gerek yöneticilik açısından yüksek bir mesleki gelişkinlik düzeyi gerektiren i leri yapan bilgi işçileri (knowledge workers) olmak üzere iki ana gruba ayrılır.

İdare alanında da mali işlemlerle ilgilenen, yani daha önce anlatılan belirli yönetim görevlerini yürüten kişilerle, bilgi işçilerinin (knowledge workers) işini daha verimli kılmakla yükümlü sekreterler, daktilolar ve arşivciler gibi yardımcı personel birbirinden ayrılabilir.

Daha önce birçok kez belirtildiği gibi, mali konularla ilgilenen kişilerin işleri, klasik enformatiğin uygulamaları yardımıyla zaten yoğun bi-

çimde otomatikleştirilmiştir. Dolayısıyla, büroda otomasyonun ise öncelikle diğer gruba girenlerin işlevleri üzerinde durması gereklidir.

Bu sınıflandırmaya dayanarak yanda aşağıdaki tabloda uzmanlarla orta ve üst düzey yönetim kadrolarının günlük iş zamanı (ve dolayısıyla maliyet) dağılımları verilmektedir.

Genelde dağılımı verilen beş etkinlik, üç meslek kategorisinin yaptığı işlerin tümünü mantıksal olarak kapsar niteliktedir.

İletişim etkinliklerine (olası gidiş-geliş süreleriyle birlikte) toplantılar ve telefon görüşmeleri girer.

Belge (ya da mesaj) üretimi etkinliği, incelenen belgelere (ya da mesajlara) not düşülmesi ve bunların havale edilmesinden oluşur.

V eri ya da durumların çözümlenmesi karar almaya yöneliktir. Okuma ise mesleki bilgilerin artırılması amacını taşır.

Doğrudan üretken olmayan etkinlikler arasında, bekleme süreleri, kendisinin ya da iş arkadaşlarının çalışmasının organize edilmesi, işlerin anında koordinasyonu, bilgi ya da kişi-

Etkinlik	Uzmanlar %	Orta düzey görevlileri %	İdare görevlileri %
İletişim	38	54	64
Belge üretme	15	11	8
Çözümlemeler	9	7	4
Okuma	4	8	6
Diğer, doğrudan üretici olmayan etkinlikler	34	20	18

lerin (doğrudan ya da telefonla) aranması, arşivleme vb. sayılabilir.

Bu üç mesleki kategoride çalışanların ortalaması dikkate alınırsa, aşağıdaki tabloda da görüldüğü gibi, bu beş etkinliğin zamana göre dağılımının nesnel değerlendirmesinin, bu kişilerce yapılan öznel tahminlerden farklı olduğu anlaşılır.

Etkinlik	Toplam sayının yüzdesi		Sapma yüzdesi
	Tahmin	Geçerli ortalama	
İletişim	29.1	45.6	—36
Belge üretimi	12.7	12.9	— 2
Çözümlemeler	14.6	8.2	+78
Okuma	9.1	7.9	+20
Diğer, doğrudan üretici olmayan etkinlikler	34.5	25.4	+34

Özellikle iletişim etkinlikleri, bunlarla ilgili kişiler tarafından büyük ölçüde azımsanırken, çözümleme iled oğrudan üretken olmayan etkinliklerin önemi abartılmaktadır. İletişime ayrılan zamanın düşüklüğü bilgi alışverişi sürecinin gerçekte olduğundan daha etkin sanılmasından kaynaklanmaktadır.

Yapılani şe büyük değer kattığı varsayılan çözümleme etkinliğine fazla ağırlık verilmektedir.

Üretken olmayan görevlerin fazla abartılması ise, onların hakkında, bilinçsiz de olsa olumsuz bir yargının varlığından kaynaklanmaktadır. Oysa bu kanıya pek uymayan bir işlemsel etkinlikleri vardır

Sonuçta, fiili dağılıma dönecek olursak, incelediğimiz mesleki fonksiyonlar açısından, çalışma zamanının % 70'i iletişim ve doğrudan üretken olmayan etkinliklerle geçmektedir.

Bu değerlendirme, en azından hedeflerinin açıkça belirlenmiş olması bakımından, diğerlerinden üstün olan basit bir büroda otomasyon projesi stratejisinin belirlenebilmesi için yeterlidir.

Özellikle doğrudan üretken olmayan etkinliklerden yola çıkarak bunların yapıları incelendiğinde, yardımcı personelin sayısını artırarak ya da daha doğrusu verimliliklerini artırabilecek araçlarla donatarak bu etkinliklerin onlara rahatça devredilebileceği görülür.

Şayet, doğru biçimde (yani aşama aşama ve otomasyonun yanı sıra formasyona "eğitime" de yatırım yapılarak) yapılacak olursa, bu ilk müdahale iş örgütlenmesine iki önemli üstünlük kazandırabilecek niteliktedir: Birinci üstünlük, yardımcı personelin performansının nitel açıdan yükseltilmesi, ikincisi de uzmanların ve yöneticilerin görev yükünün hafifletilmesidir.

Bu üstünlükler, iş yükünün eşit olduğu hallerde, kurumsal görevlere daha fazla zaman ayrılması anlamına gelir. Güdülenme açısından bakıldığında, her iki durumda da, profesyonelliğin artışı yönünde bir hareket söz konusudur.

İletişim etkinlikleri ile ilgili olarak ise aşağıda telefon görüşmeleri ile elektronik posta iletişimi karşılaştırılmaktadır:

TELEFON MESAJI:

1. İlk aramadan yalnızca % 26 oranında bir sonuç alınır.
2. Geri kalan % 74 ise,
   % 38 görüşülecek kişi rahatsız edilemez
   % 38 karşı numara yarıt vermez
   % 14 meşgul çalar
   % 10 diğer nedenler
3. Mesajın anımsanması (bir kez daha telefonla iletilmesi) olanaksızdır (ancak dikte etmek-yazdırmak-dosyalamak-aramak koşuluyla bu mümkün olur).
4. Aynı mesajın birden fazla kişiye ulaştırılması zordur.
5. Bu yol ekonomik değildir.
6. Nezaket tümcelerinden kaçınmak mümkün değildir.

ELEKTRONİK MESAJ:

1. Mutlaka bir sonuç alınır: Görüşülecek kişinin yerinde bulunması gerekmez.
2. Görüşülecek kişinin işine ara vermesi gerekmez.
3. Mesajın alınıp alınmadığı denetlenebilir.
4. Basit işlemlerle yeniden aranabilir.
5. Kolayca sirküler mesaj yollanabilir.
6. Mesajın ileti süresini kısaltır.

Telefon mesajının ortalama süresi 4.8 dakika
Elektronik mesajın ortalama süresi 1.3 dakika
Mesaj başına kazanılan süre 3.5 dakika

Elektronik posta, ısaca bir mesajın (ya da belgenin), görüşülecek kişinin aynı anda terminalinin başında (ya da hattın öbür ucunda) olmasına gerek kalmadan, bir yada birçok ki-

şiye iletilmesi olarak tanımlanabilir. Elektronik posta sistemi, bilgileri belleğine alıp istendiğinde ilgili kişilere iletebilmektedir.

Görülebileceği gibi elektronik posta, (işlemsel ve teknik nitelikte) bir dizi büro iletişimi için telefona tercih edilmesine yol açan (ve başlıcası arşivleme yeteneğine bağlı olarak, bilgi değişimini eşzamanlılık zorunluluğundan kurtarmak olan) üstünlüklere sahiptir.

Stanford Araştırma Enstitüsü'nde yapılan bir inceleme, elektronik postanın sistematik olarak kullanılmasının yalnızca yönetsel görevlerde, % 20 civarında bir zaman tasarrufu sağlayabileceğini göstermiştir.

Her halükarda elektronik postanın yardımıyla iletişimlerin içeriğine bağlı olarak üç farklı aktarım kanalına sahip olan iletişim akışı rasyonel bir yapıya kavuşur.

''Siyasal'' nitelik taşıyan argümanlar, gelecekte de kişisel birlikteliklerde konuşulacaktır. Acele kararları gerektiren konular telefonla çözülecektir. Tüm diğer konularda ise, işin planlaması açısından bu yeni kanaldan rahatlıkla yararlanılacaktır. Planlama alanında geleneksel iletişim yolları kullanıldığında ara vermeler ve zaman kayıplarının sekiz çalışma saatinin 1.5'unu oluşturduğunu unutmayalım.

Aşağıdaki tabloda, bu stratejinin uzmanlar ve yöneticiler tarafından benimsenmesinin üretkenlik artışı yönünde sağlayabileceği etki irdelenmektedir.

Araç	Potansiyel zaman tasarrufu (%)	Toplam zaman tasarrufu yüzdesi
Telekonferans	0.5	2.7
Bilgi aktarımı	4.4	29.9
Bilgilerin saklanması ve yeniden bulunması	4.0	27.2
Kişisel işlemler	4.0	27.2
İş yönetimi	1.7	11.6
	14.8	100

Ayrıca, eldeki en gelişkin teknolojiye bağlı olarak, bir bölümü daha önceki bölümlerde açıklanan ve bu stratejiyi destekleyen, uygulama araçları da bu tabloda sergilenmektedir.

Bilgi aktarımı ile ilgili olarak elektronik posta kullanılabilir; **bilgilerin bellekte saklanması ve yeniden bulunması** ise bilgilerin yetkili kişilerce depolanıp gerektiğinde hizmet sunulabilmesidir.

İş yönetimi, elektronik ajanda ve alacak-verecek defteri vb. olarak adlandırılan hizmetleri içermekte ve eskiden doğrudan üretken olmayan etkinlikler alanını kapsamaktadır.

Kişisel işlem ile, kullanıcının (uzman, ara ya da üst düzey yöneticisi) gerekli gördüğü takdirde, kendisini özel olarak ilgilendiren bir dizi veriden yararlanarak basit işlemler yapabilmesi imkanının tanınması kastedilir.

Telekonferans ise, orta ve uzun dönemde, insanlar arasında telefon ve elektronik posta yardımıyla başarı ile kullanılabilecek bir iletişim aracıdır.

Bu araçla, işitsel ya da hatta görsel (bu durumda telefon frekanslarından iki kat yüksek frekans kullanıldığından, bu yatırım bugün için pahalıdır) iletişim kanalları kullanılarak iki ya da daha fazla insan topluluğunun toplantı yapabilme olanağı sağlamaktadır. Yıllardan beri NASA'da (Amerikan Uzay Kuruluşu) başarı ile kullanılan bu olanak, henüz Avrupa'daki işletmelerde yaygın değildir. Ayrıca maliyetinin ötesinde çok uzun bir adaptasyon süresi de gerektirmektedir (olası zaman tasarrufuna çok az bir katkı olabileceği yönünde bir anlayış buradan kaynaklanmaktadır).

İş yönetimi ile ilgili olarak son bir belirtme yapmak gerekir. Bu öge, istihdam yönetimi, vade takvimi (alacak-verecek defteri), ortak araçların kullanımının planlanması vb. için otomatik bir yardımdan oluşur. Bu ögenin uzmanların ve yöneticilerin işlerinin düzeltilmesi için yapabileceği katkı, göreli olarak sınırlıdır. Çünkü değerlendirilmesinde, doğrudan üretken olmayan etkinliklerin, kendi açısından üretkenlik artıracak her türlü araca sahip ve bu nedenle bunları, doğrudan üretken olmayan etkinliklere uygulayabilecek durumda olan yardımcı personel tarafından önerilen stratejiye göre dağıtıldığı varsayımı yatmaktadır.

Büro otomasyonu ile ilgili ya da daha doğru bir deyişle büroda yürütülen faaliyetlerin rasyonelleştirilmesi ve otomasyonu ile ilgili bu gelecek manzaraların analizinin sonunda, bugüne kadar alınan yolu bir kez daha düşünmek ve yeni birtakım değerlendirmeler yapmak gerekiyor.

İleri sanayi toplumları için enformatiğin vazgeçilmez bir tercih olduğu kanıtlandı. Bu genel çerçeve içinde, büroda otomasyon ile klasik veri işleme uygulamaları arasındaki farklılıklar ve farklı bakış açıları tartışıldı. Ayrıca, olumlu sonuçlara götürülmesi gereken otomasyon stratejileri önerildi.

Büro etkinliklerinin değişmesi konusunu makalemize bilerek almadık. Bunun birinci nedeni bu kuşbakışı yazının bir giriş niteliğinde olması, ikinci nedeni ise konu üzerine bütünleşmiş bir görüşün egemenlik kuramamış olması nedeniyle, sorunu bu bakış açısından irdelemek için henüz erken olmasıdır. Önerilen strateji, bu bakış açısını bir yana bırakmış ve basit bir çıkış noktası olarak otomasyonun gelecekteki işlevlerinin çözümlenmesini koymuştur.

Bu temel üzerinde, büroda otomasyonla ilgili bir projenin planlama süreci şu aşamalara ayrılabilir:

Başlangıç modeli, büroda otomasyon sisteminin olası kullanıcılarının toplamından oluşur. Bu kullanıcıların yaptıkları işlerin profili anket formları aracılığıyla belirlenir.

Daha sonra bu işin verimliliğinin, daha önce birçok kez gündeme getirdiğimiz ölçüde geliştirilmesine yarayabilecek araçlar saptanır.

En önemli nokta bu araçların uygun bir sistematik çerçeveye oturtulmuş olmasıdır. Bu sistematik çerçevenin enformatik bir çerçeveyle uyumlu olması zorunludur.

Bu proje, önemi ve iş organizasyonu üzerindeki etkisinden ötürü, işletmenin hedef ve stratejileri ile bir bütün oluşturmalıdır. Gelişimi, yöneticiler tarafından, özellikle klasik enformatikte olduğundan daha yoğun şekilde, yakından izlenmelidir. Bu, işletmenin orta ve uzun dönemli planları (3-5 yıl) ile kaynaklarının uyumlu hale getirilmesi anlamını taşır.

Bu noktada ürünlerin mali plan ve kazanç planı, işletmede geliştirilen stratejik otomasyon planının bir uzantısıdır.

Bu planı uygulamaya koymak sorumluluğu kime verilecektir? Onbeş günde bir yayınlanan ''Computer World'' (Bilgisayar Dünyası) adlı dergi tarafından, 1981 sonunda abone 400 işletmeye uygulanan bir soruşturmada bu soruya bir yanıt alınmıştır.

İşletmelerin yarısı, büroda otomasyon için özel uzmanlığın gerekli olduğuna inanarak, sorumluluğu kendi bünyelerindeki bilgisayar servisine vermekten yanadır. Büro dünyası ile bilgisayar dünyasını kaynaştırma zorunluluğu bu tezi desteklemektedir.

İşletmelerin % 20'si, işletmenin projeye bir bütün olarak katılımını sağlamak amacıyla, bilgisayar servisinin yanı sıra plan ile ilgili tüm diğer servislerin temsilcilerinden oluşan özel bir komite kurulmasını savunur.

Bu çerçevede, soruşturmada öngörülmemiş olmakla birlikte, hedefe yönelik daha özgün görevlerin enformatik alanında dikkate alınacağı açıktır.

İşletmelerin % 30'u ise büroda otomasyonu bir maliyet sorunu, dolayısıyla idari bir sorun olarak görülmektedir. Ancak ABD'de bile bilgisayar servislerinin idari müdürlükler içinde yer aldığını unutmayalım.

Bu soruşturmaya verilen yanıtlar incelenirken işletmelerin enformatiğe giderek artan ölçüde önem verdiği anlaşılmaktadır.

Sonuçlara bakılırsa, 1981 sonuna dek ancak küçük bir azınlık (% 3) otomasyon sorununu henüz gözönüne almamış durumdadır; işletmelerin neredeyse tamamı bu projeleri ya uygulamaya ya koymuş, ya koymakta, ya da ciddi biçimde düşünmektedir.

Burada, ABD'de bilgisayar kültürünün çok yaygın olduğunu bir kez daha yinelemek istiyoruz. Bu yazının başında açıklanan nedenlerden dolayı, yakın gelecekte ekonomik sistemin rekabet şansı, bilgi işlem alanındaki başarılarıyla ölçülecektir. Bu nedenle, geri kalmak istemeyen her ülkenin bu alanda ABD (ve unutmayalım Japonya) ile bu alanda yarışması gerekecektir.

Dolayısıyla bu gecikmeyi gidermek için mümkün olan en yüksek hızla karar alınmalı ve gerçekleştirilen deneylerden azami yararı sağlamak için harekete geçilmelidir.

G.Ochhini'nin **Quaderni di Informatica**'da, (anno X, n.1,1983, Honeywell Information Systems Italia) yayınlanan ''L'informatico nel lavoro d'ufficio'' adlı yazısından.

Her daireye düşen gider payı, toplam rakamla bu giderle ilgili tablo bindeleri çarpımının 1000'e bölünmesiyle elde edilir.

Tablo iki aşamada hazırlanabilir:

— Bindeli bölünmelerin (dağılımların) ve formüllerin diskete girilmesi ve disket üzerinde saklanması. Bu andan itibaren veriler arşivin bir parçasıdır ve güncelleştirme, değiştirme ya da bilgiyazım için yeniden çağırılabilirler.

— Giderlerin girilmesi. Önceden hazırlanan harita bellekten çağırılır ve her yeni harcama, karşılık gelen alana girilir.

Program, otomatik olarak hesapların yapılıp sonucun verilmesini sağlar. Bilgiyazım istenirse her dairenin durumu ve toplam elde edilir.

Haritanın bu yapılanış biçiminde giderlerin toplamı için bellek ayrılmaz. Örneğin, 15. satıra bir gider girilince, yalnızca bu sayı için her dairenin ödeyeceği miktar hesaplanır. Daha sonra yine 1. dağılım listesine dahil yeni bir harcama yapılırsa, harita, toplamı değil bu giderlerle ilgili borç miktarını (yani bu giderle ilgili borç artı bir önceki borç) verecektir. Bu nedenle, çeşitli girişlerin toplanacağı ve her daire için genel toplamın gösterileceği yeni bir sütunun oluşturulması gereklidir. Toplamlar sütununun ayrıca saklanması ve sonra, bunları önceki sonuçlarla toplayıp veren yeni bir harita için yeniden çağrılması bir çözüm olabilir.

Veriler, kütüklere, BASIC'te kullanılanlara benzer güdümlerle (SAVE, LOAD, vb.) aktarılır. Buna ek olarak kütüğün özelliğini gösteren (bir harita olmayıp bir veri kütüğü olduğundan formül içermez) uygun karakterler eklenir.

Örneğin bazı yerleşim haritalarında, # simgesi, verilerin saklanmasının istendiğini gösterir. Verilecek parametreler, veri kütüğünün adı ve aktarılacak satır ya da sütundur. Birbirini izleyen haritalar böylece, bellekte saklanan değerleri hep aynı seçenekle (örneğin L= LOAD güdümü ardından # simgesi) belleğe yükleyerek kullanabilirler.

Ancak aktarılan değerler konusunda dikkatli olmak gerekir. Örneğimizdeki H sütunu, bir dizi sayıdan (kısmi tutarların toplamı) oluşmakla birlikte, gerçekte, sayısal değerleri (yani verileri) değil, bunların hesaplanması için gerekli formülleri içerir.

Bu sütun, verilerden oluşuyormuş gibi belleğe alınırsa, hataya yol açacaktır.

Ortaya başlıca iki durum çıkabilir: ya program formülleri farkettiği için hiçbir şey aktarmaz, ya da formülleri aktarır; bu sütun yeni bir haritaya taşınırken formüller denetimsiz biçimde uygulandığından hatalı değerler görüntülenir.

Bu engeli aşmak için bir öncekini içeren yeni bir sütun açmak yeterlidir. Aktarma, sayısal düzeyde gerçekleşir ve formülleri değil, hesapların sonucunu içeren bir sütun elde edilmiş olur.

Örneğin H1'de @ SUM(E4..G4) hesabı, yani E4 ve G4 arasında kalan değerlerin toplamı bulunur.

Sayısal bir değer olan sonuç, bu haliyle H1 hücresinde sunulmakla birlikte, gerçekte hücrenin içeriği bir formülden oluşmaktadır. Bu hücre yeni bir yerleşim haritasına aktarılırsa (burada programın formül aktarımını öngördüğünü varsayıyoruz), sonuç istenildiği gibi olmayacaktır. Hesabın yapıldığı H1 hücresinin sayısal içeriğini kopya etmek için, önce sonuçta ortaya çıkan değeri aynı haritanın başka bir hücresine aktarmak ve bu yeni hücreyi veri düzeninde belleğe almak gerekir. Böylelikle bu hücrede bulunan sayısal değer başka haritalarda kullanılabilecek hale gelir. Bir ve aynı yerleşim haritası içinde sayının aktarımı kopya güdümü ile gerçekleştirilemez. Çünkü kopya güdümü, sonucu değil hücrenin içeriğini, yani formülü aktaracaktır.

Sonucu aktarmak için, giriş hücresine çıkış hücresinin adresini (koordinatlarını) yazmak yeterlidir. Örneğin L1'e +H1 yazarak, H1'de bulunan değeri L1 hücresine aktarabiliriz.

+ simgesi programı bir karakter zinciri değil, bir adres aktarıldığı konusunda uyarmaya yarar; bu simge unutulursa H1 yazısı hiçbir güdüm anlamı taşımadan L1 hücresine aktarılmış olur.

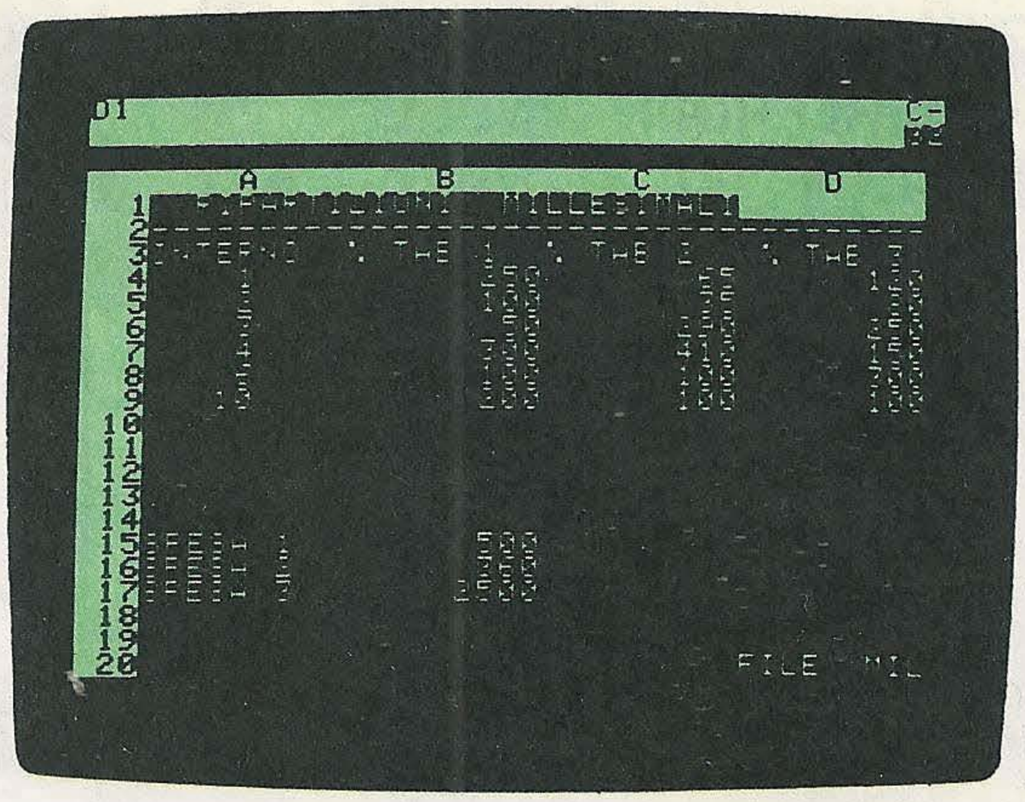
Stok yönetimi

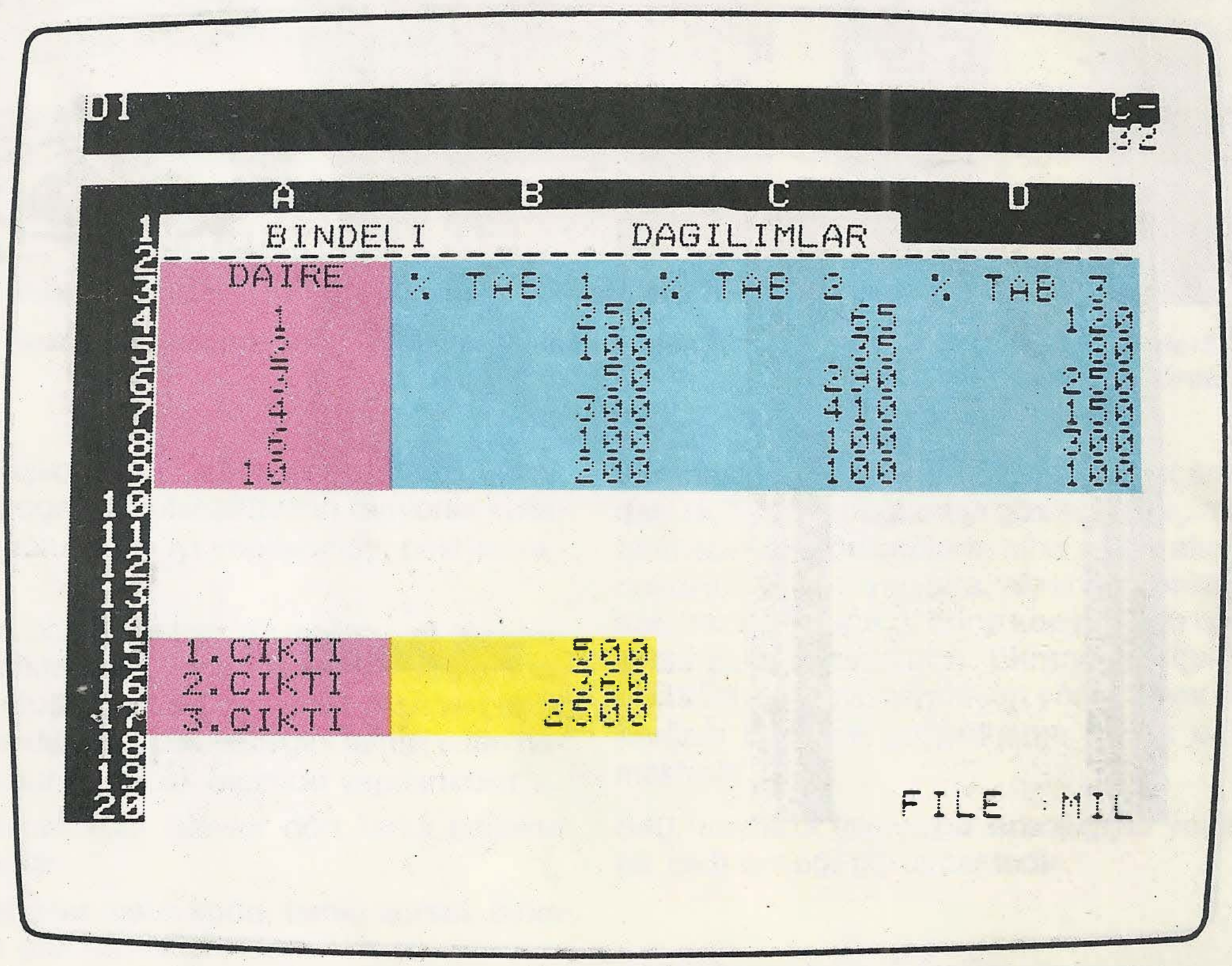
Modern kişisel bilgisayarların büyük bellek kapasitesinden yararlanılarak, maliyeti çok sınırlı olan bu makinelerde de karmaşık uygulamalar geliştirilebilir.

Yerleşim haritaları tekniğinin kullanılmasıyla, program geliştirme için gerekli büyük yatırımdan kaçınılabilir. Ayrıca uygulama, ekonomik açıdan çok rantabl olabilir. Bununla birlikte, yerleşim haritalarını kullanabilmek için de ortada duran sorunla ilgili doğru bir yapıya erişilmesini sağlayacak bir çözümleme evresinden geçilmesi zorunludur. Çözümleme evre-

GAYRIMENKUL (APARTMAN) GİDERLERİNİN BİNDELİ DAĞITIMI (1)

Bu ve bundan sonraki sayfalarda, bindeli tablolara dayanarak apartman giderlerinin yerleşim haritasının kullanımı gösterilmektedir.





A sütununda, bir dizi değiştirilmeyecek açıklama (her apartman dairesinin numarası ve giderler) yer alır. Bu nedenle, A sütununun dikey başlık (TITLE) olarak tanımlanması yerinde olur.
A sütununun içeriğinde değişiklikler yapılması istendiğinde, TITLE seçeneği kaldırılarak, bir ya da birden fazla hücrenin içeriğini değiştirmek ve tercihi yeniden oluşturmak gerekir.

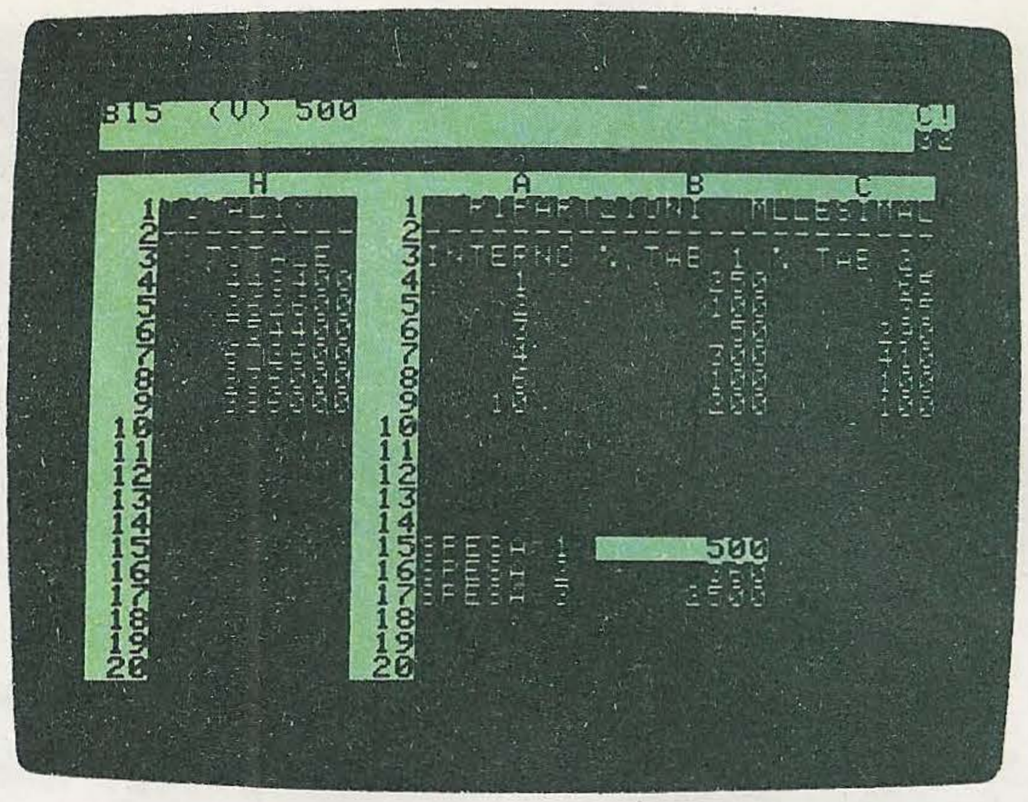
B, C ve D sütunlarında bin-de olarak ifade edilen bindeli katsayılar yer alır. Açıklamada % simgesi kullanılır, çünkü doğru simge olan (‰ 10), standart USA-ASCII'-de öngörülmemiştir.
Üç sütunun her birinde, özel bir gider endeksine bağlı olarak, farklı bir dağılım bulunur. Bindeli tablolar değişemez ve bir kez girildikten sonra tüm işlemlerde kullanılır.

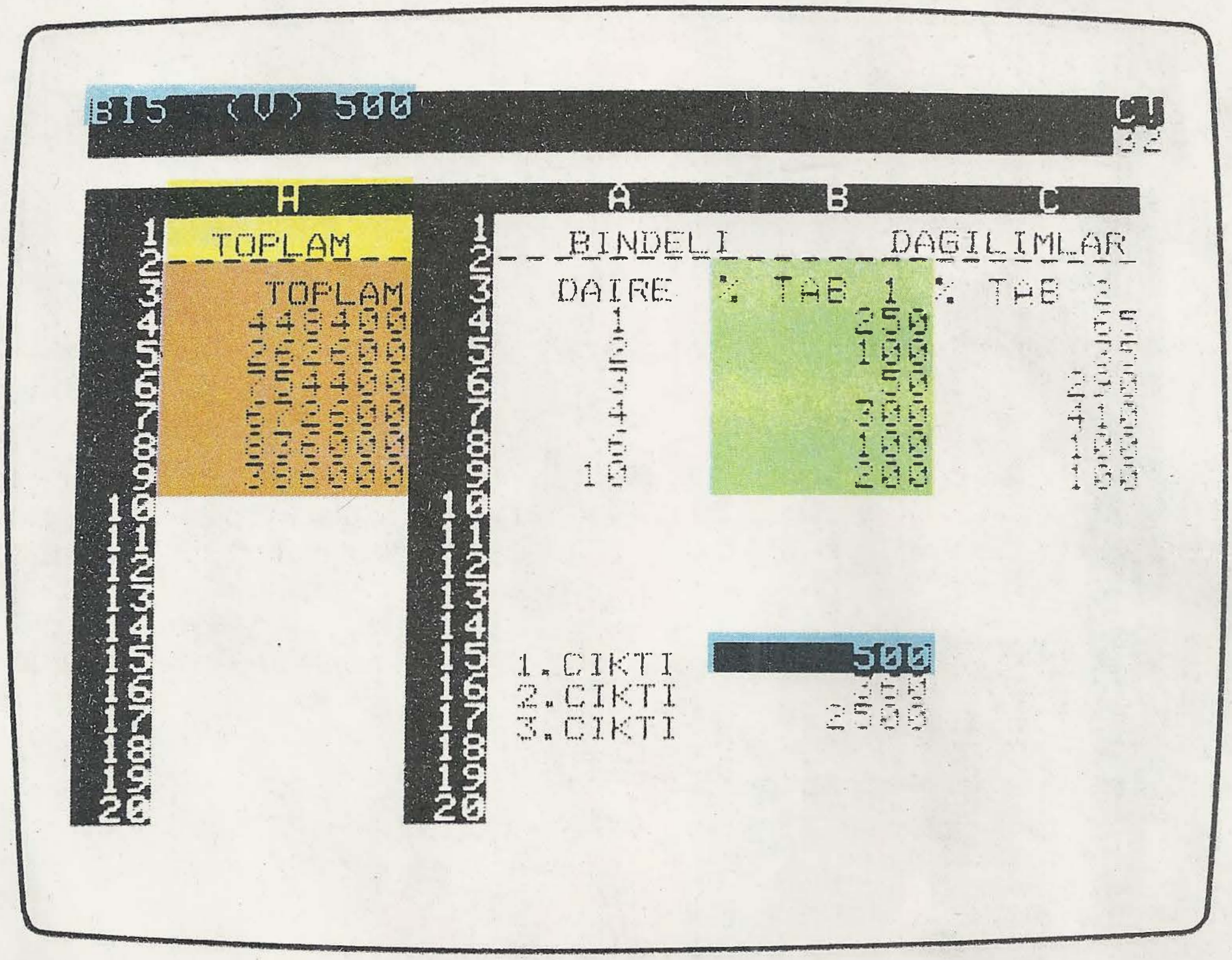
B15, B16 ve B17 hücreleri, paylaştırılacak toplamların girişine ayrılmıştır. Bu hücrelerden birine bir değerin girilmesiyle, karşılık gelen bindeli tablo temelinde dağılım hesaplanması başlar.

Hesaplanan tutarların gösterimi için "Scroll" (ekran içeriğinin kaydırılması) gereklidir. Yatay yönde yapılacak bu kaydırmayı sağlamak için, imlecin, ekranın görünen bölümünün dışına doğru sağa kaydırılması gerekir.

GAYRIMENKUL (APARTMAN) GİDERLERİNİN BİNDELİ DAĞITIMI (2)

Ekranda, yerleşim haritasında bir gösterim penceresi oluşturmakta yararlanılan özel bir seçeneğin kullanımını görüyoruz.





Soldaki ilk sütun olarak görünen H sütunu, aslında yerleşim haritasının en sağındaki sütundur. W(WINDOW = pencere) güdümü kullanılarak, yerleşim haritasında fiziksel olarak birbirinden çok uzakta olan alanlar elde edilebilir ve karşılaştırma için yanyana getirilebilir.

B15'te bulunan imleç, 1. ÇIKIŞ'a ait olan tutarın girişini sağlar. Tutar, girişlere ayrılan satırda sayı (V) olarak görüntülenir.

Her dairenin gider payı, 1. tablo (B sütunu) temelinde, otomatik olarak hesaplanır.

Hesaplanan paylar, her daire için oluşturulan toplam giderlerde toplanır.



Bir hesap merkezinde manyetik bant takmakta olan bir sekreter. Solda bir dizi katmerli disk (''disk-packs'') görülüyor.

sinde başlıca hedef, akış şemaları hazırlamak değil, programın olanaklarının rasyonel kullanımına yönelik en iyi yapılanmayı belirlemektir.

Örneğin, stok yönetimi, ya malların ve elde bulunan miktarların (stok mevcudu) sıralanmasından oluşur ya da ihtiyat miktarları ve çeşitli durumlarda neler olabileceğini temsil eden hesaplamalar yapacak biçimde yapılanabilir...

Yürütülecek tüm işlevler dört farklı başlıkta toplanabilir:

— **Kayıt.** Her malın kodu, temin süresi, birim fiyatı gibi özellikleri içerir.
— **İşlemler.** Stoğun muhasebe durumunu verir. İşlenen veriler, mal giriş ve çıkışlarında değişmeler ve stokların durumudur.
— **Değerleme.** Kayıt bölümündeki birim fiyatlara göre değerlendirilmiş veriler, bu bölümde yer alır.
— **Üretim maliyetleri.** Bu stokta bulunan malzemeler nihai malların üretiminde kullanılıyorsa, bu mallardan her biri için gerekli olan malzeme miktarı belleğe alınarak, tek tek malzemelerin toplam maliyetleri ve her nihai malın maliyeti otomatik olarak hesaplanır. 836-840. sayfalarda uygulama örneği verilmektedir.

Ekrandaki görüntü ile verdiğimiz örnek arasındaki doğrudan bağlantıyı göstermek için, birbirini izleyen görüntülerin birer fotoğrafını sunuyoruz. Bunun yanısıra, ekranda görüntülenen her şeyi kağıt üzerine kopya etme olanağı her zaman mevcuttur. Ekranın içeriğini bilgiyazara aktarmaya yarayan yöntem, yerleşim haritası kullanım programının içinde bulunmaktadır.

840. sayfada bilgiyazar aracılığıyla yapılmış bir çıktı örneği görülmektedir.

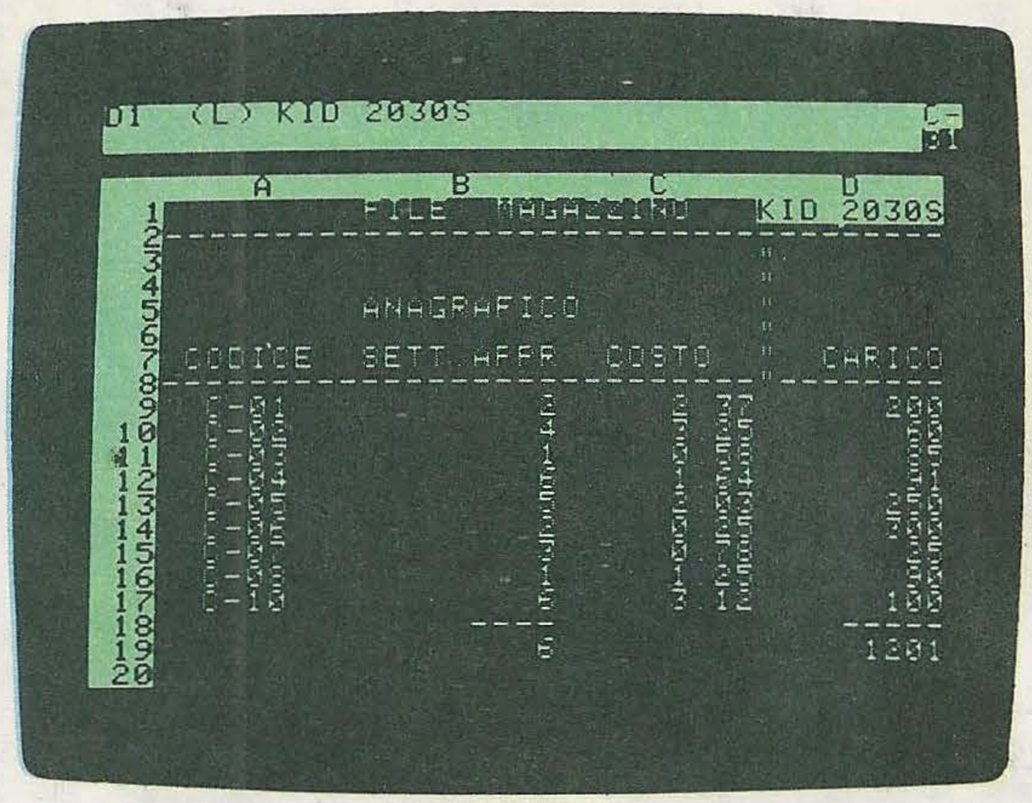
Değişmelerin Çözümlenmesi (WHAT IF)

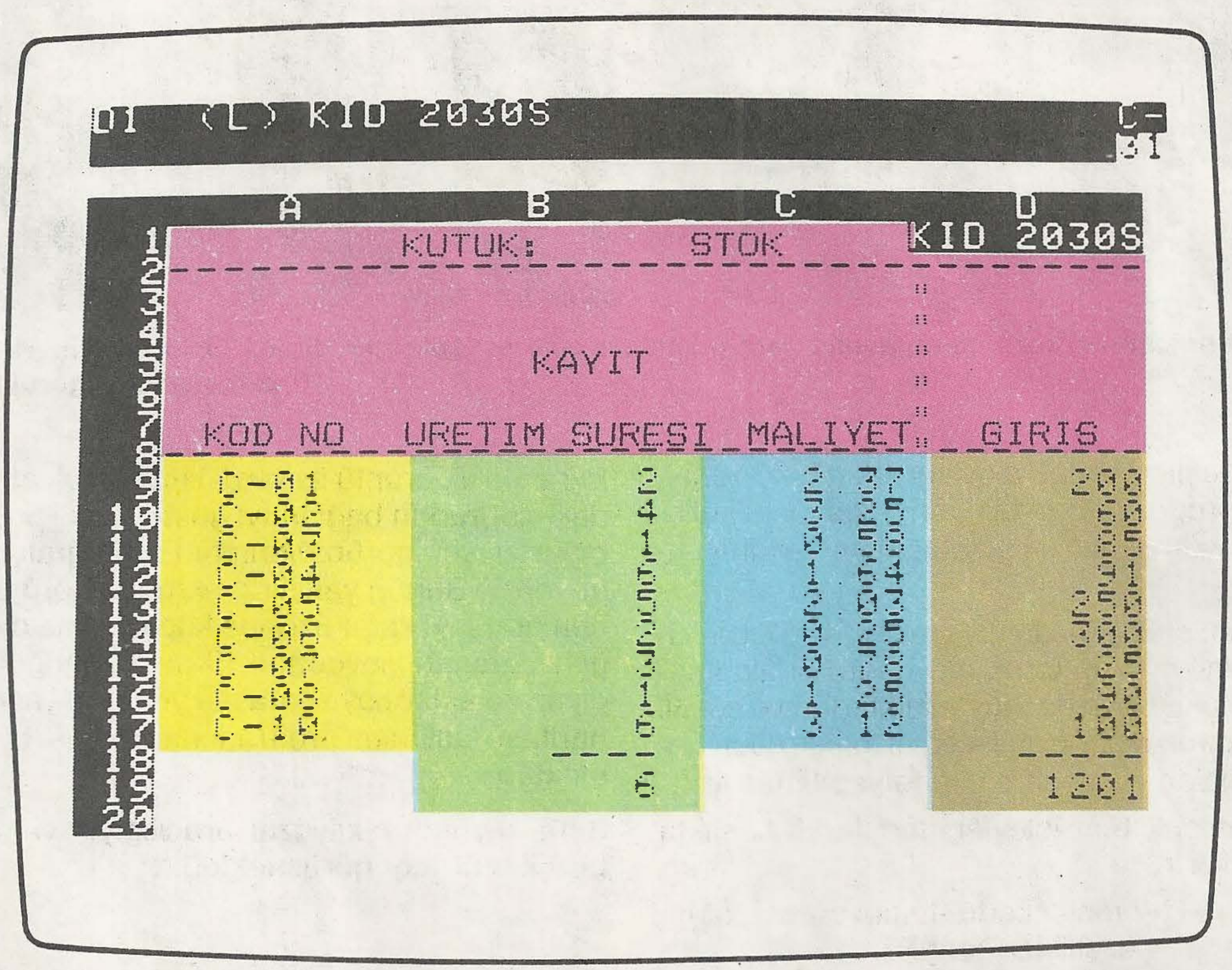
Bu deyimle, parametrelerden birinin ya da birkaçının değişmesiyle genel durumun nasıl değiştiğini belirlemek amacıyla, ekonomik ya da ticari nitelik taşıyan olgulara uygulanan bir çözümleme türü anlatılmaktadır.

WHAT IF (''..... sa ne olur'') tekniğinde, durumu belirleyen verilerde değişiklik yapılarak, bunun sonuç üzerindeki etkileri gözlenir. Karmaşık durumlarda son derece gelişmiş algoritmalar kullanılarak olgunun matematik bir modeli yapılabilir. Daha sık rastlanan durum-

STOK YÖNETİMİ: KAYIT KISMI

Ekranda, stok yönetimi için bir yerleşim haritasının kayıt kısmı görülmektedir.





1.-8. satırlar başlıkları (TITLE) içermektedir.

Her mal, bir kod numarası temelinde belirlenmiştir. Kod numaraları, A sütunundaki hücrelere kaydedilir.

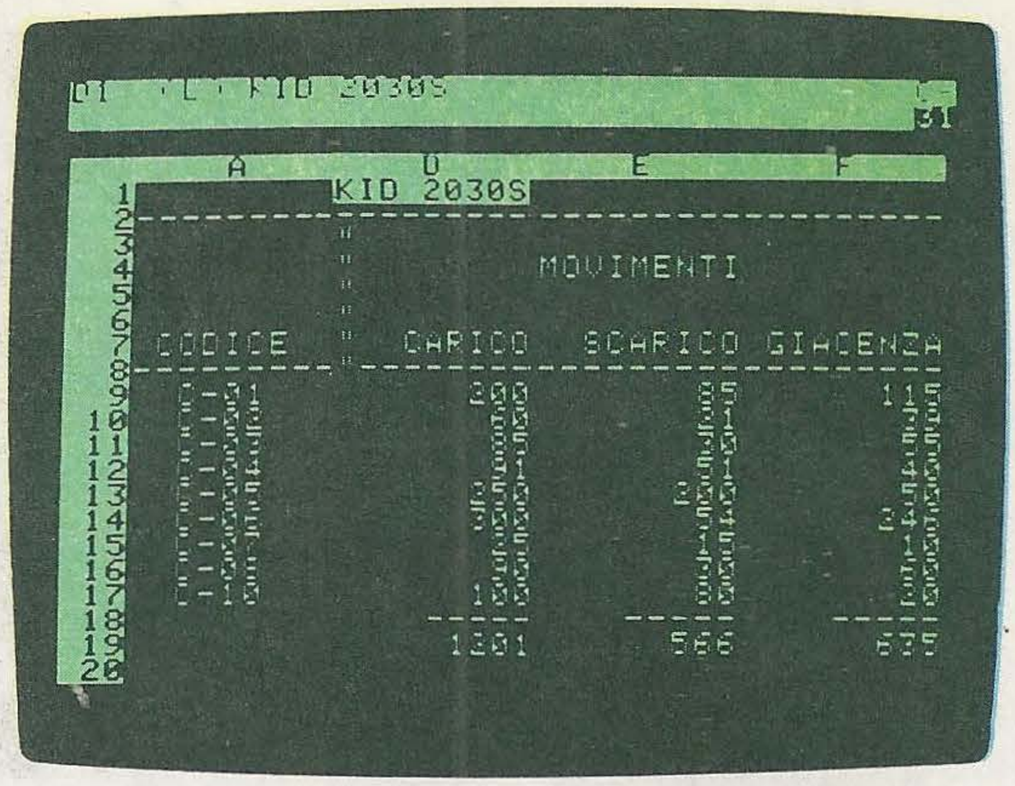
B sütununda, her mal için temin süreleri (hafta cinsinden) kaydedilmiştir. B19 hücresinde, B sütununun hücrelerinde bulunan azami değeri hesaplayan, @ MAX (B9,...,B17) fonksiyonu bulunmaktadır. 6 haftalık temin süresi, C-04 ve C-10 mallarının her ikisi için geçerlidir.

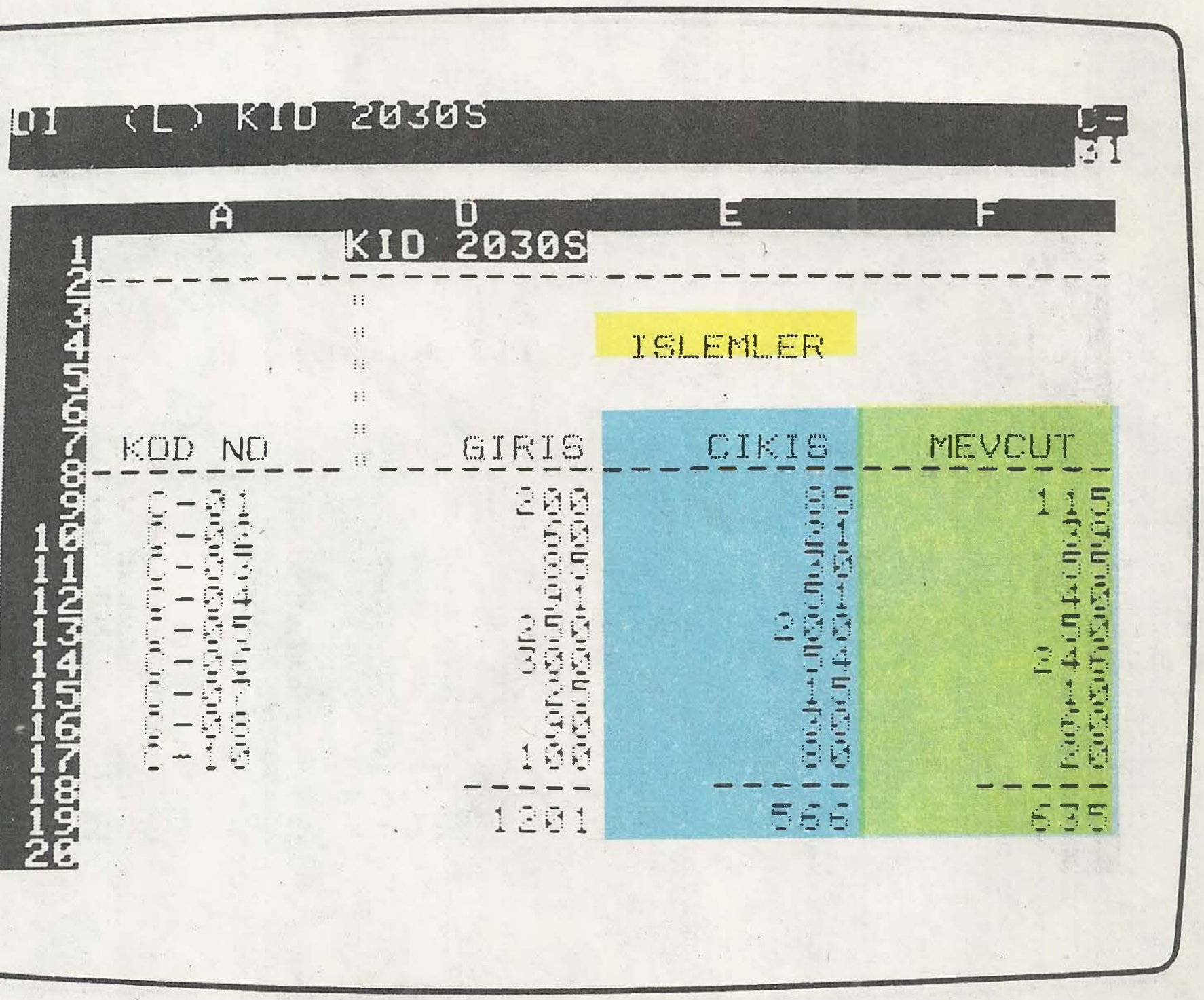
C sütununda farklı malların maliyetleri bulunmaktadır.

D sütunu, yerleşim haritasının bir sonraki (işlemler) bölümüdür.

STOK YÖNETİMİ: İŞLEMLER

Yerleşim haritasının yana kaydırılması (scroll) yapılarak, işlemler bölümünün görüntülenmesi sağlanır.





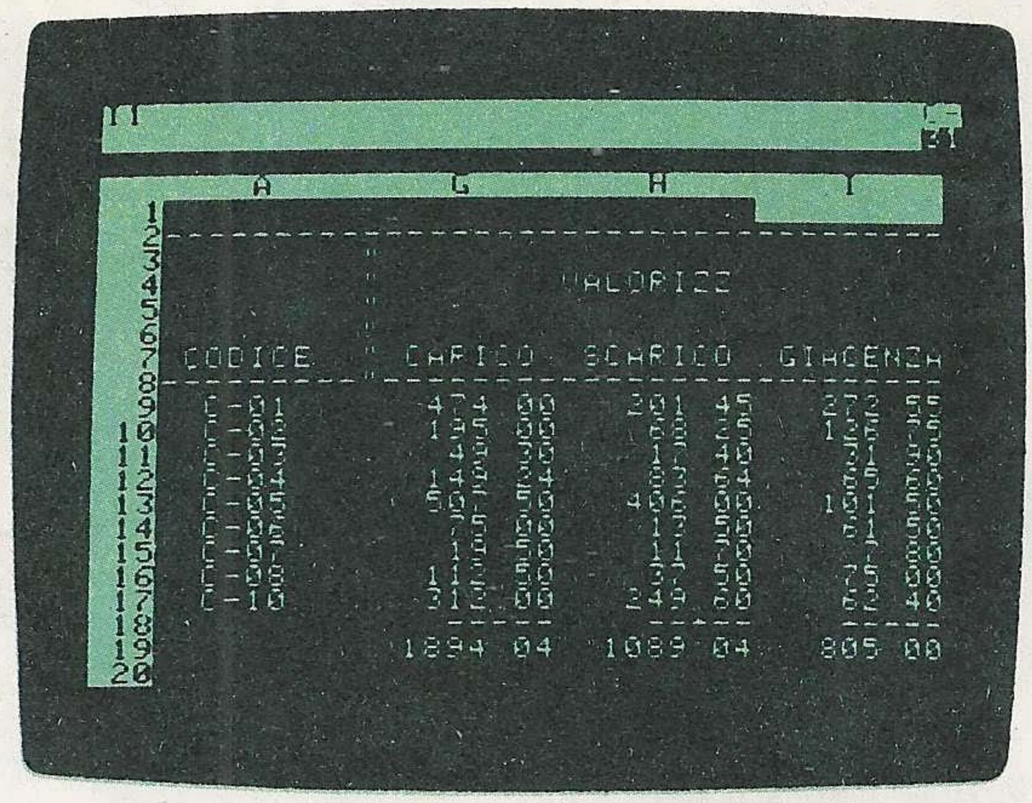
İşlemler bölümü, yerleşim haritasında, mal kodlarının bulunduğu A sütunundan oldukça uzak bir bölümde bulunur. Kodların ve işlemlerin aynı anda görüntülenmesi isteniyorsa, A sütununun dikey başlık (TITLE) olarak tanımlanması gerekir.

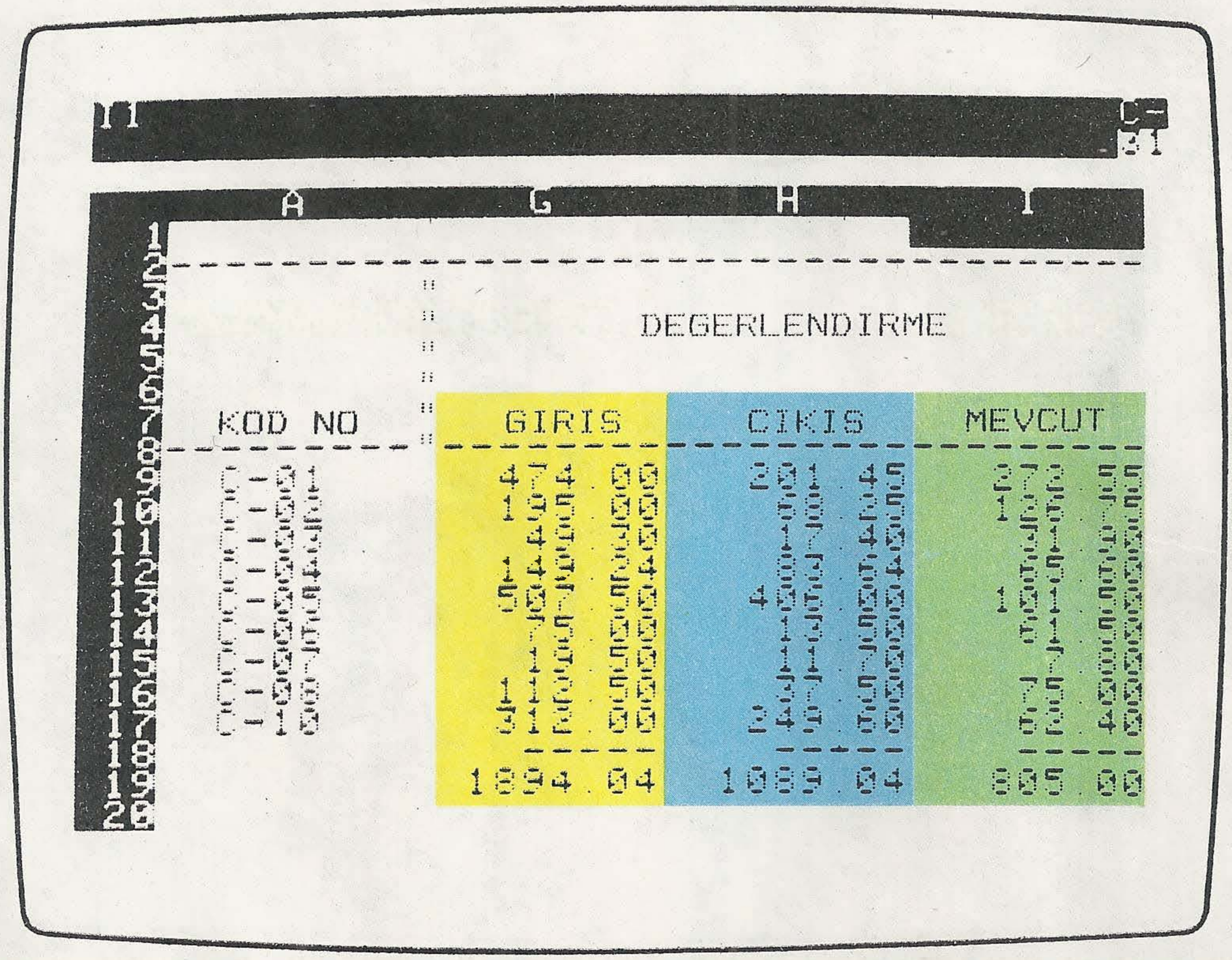
Mal çıkışına ait bir veri kaydı girilmesi durumunda (E sütunu), E19 hücresinde mal çıkışının yeni toplam tutarı elde edilir (E19 hücresi SUM(E9...E17) fonksiyonunu içerir). Hesaplamanın sütunlara göre yapıldığı hatırlanmalıdır.

Hemen ardından, girilen mal çıkışı işlemine konu olan malın stok mevcudu hesaplanır (F sütunu). Örneğin, C-01 malından belirli bir miktar çıkış yapılmışsa, F9 hücresi, D9'daki mal girişi ile E9'daki mal çıkışı arasında bir çıkarma işlemi ile yeniden hesaplanır.

STOK YÖNETİMİ: DEĞERLEMELER

Değerlendirmeler bölümünde, her mal için giriş, çıkış ve stok mevcudu ile ilgili sürekli değerler görüntülenir.
19. satırda ise toplam değerler yer alır.





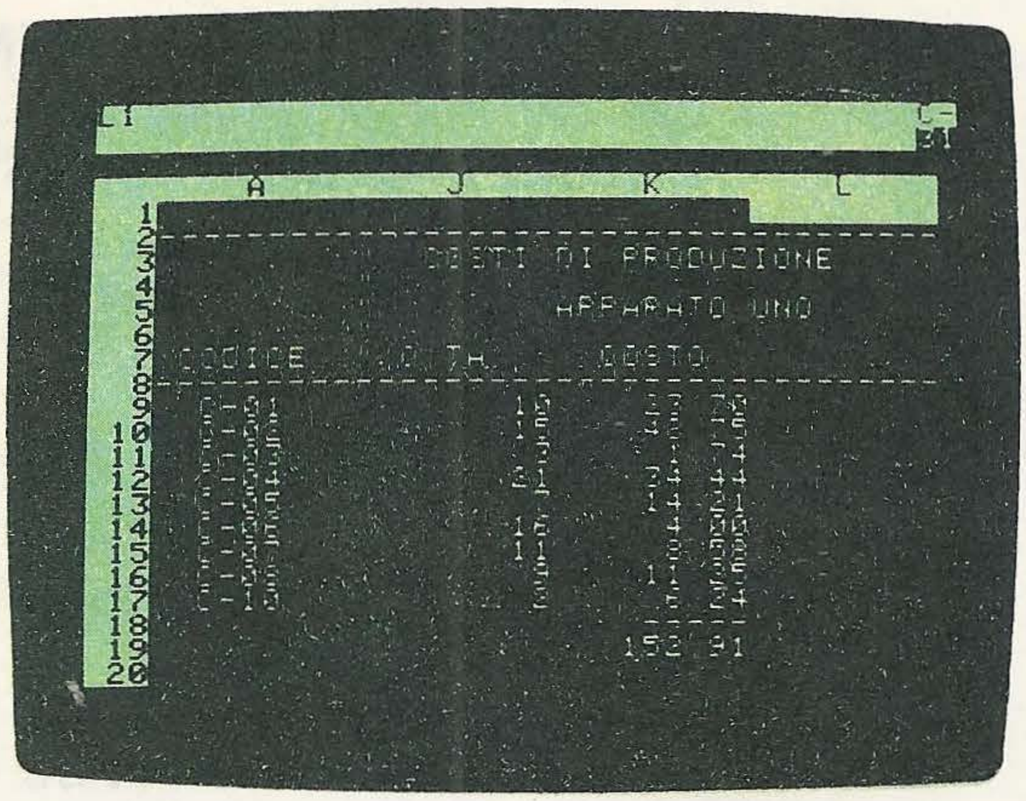
Mal girişi ile ilgili değerlendirmeler (G sütunu), belirli bir malın girişi (işlemler bölümü, D sütunu) ile birim başına fiyatın (kayıt bölümü, C sütunu) çarpımı sonucunda elde edilir. Örneğin, +C9*D9 hesaplaması ile, G9 hücresinde 474.000. veri kaydı elde edilir.

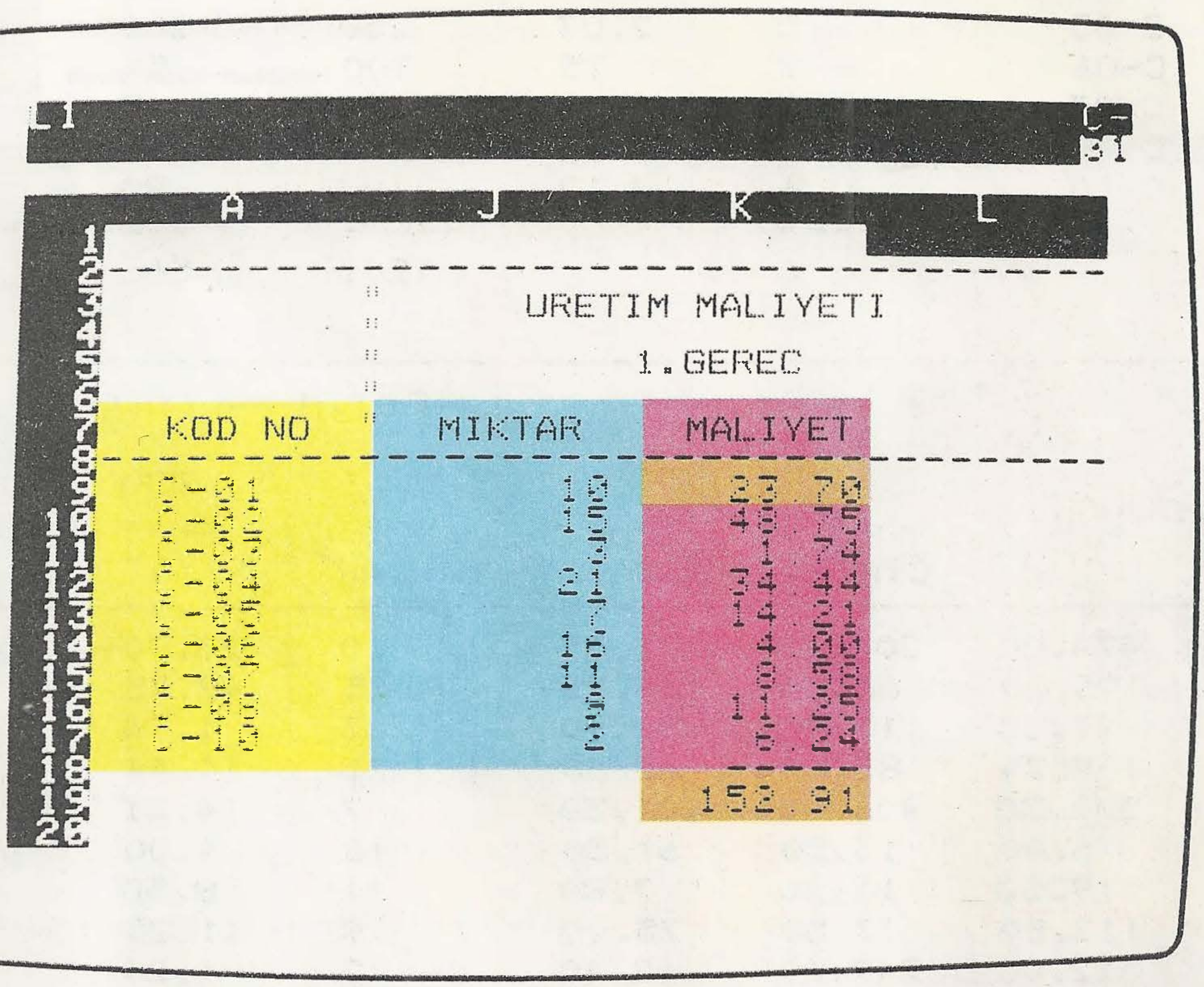
Mal çıkışı ile ilgili sürekli değerler (H sütunu), belirli bir malın çıkış miktarı (işlemler, E sütunu) ile birim başına fiyatın (kayıt bölümü, D sütunu) çarpımı sonucunda elde edilir. H9'daki 201.45 veri kaydı, +C9*E9 formülünden kaynaklanmaktadır.

Stok mevcudu değerlendirmesi, karşılık gelen şekilde, +C9*F9 formülü ile sağlanır.
İşlemlere ayrılmış bölümdeki miktarlardan biri değiştirildiğinde, tüm değerlendirme durumu otomatik olarak yeniden hesaplanmış şekilde elde edilir.

STOK YÖNETİMİ: ÜRETİM MALİYETLERİ

Bu ekranda, malzeme olarak kayıt bölümünde belirtilen malları içeren gereçlerin üretim maliyetlerinin bulunduğu bölüm gösterilmektedir.





A sütunu, eldeki malların kodlarını içermektedir ve yalnızca bellek olarak kullanılmaktadır.

J sütunu, nihai ürünü oluşturacak tek tek malzemelerin miktarını yerleştirmede kullanılır. Örneğimizde, BIR gerecinin (başlıkta adlandırılmıştır) üretiminde C-01'den 10 birim, C-02'den 15 birim vb. kullanılacaktır.

K sütununda, emek hariç, üretim maliyetinin hesaplanmasını sağlayan formüller bulunmaktadır. Yerleşim haritasının biraz daha genişletilmesiyle, bunlar birim başına fiyat olarak da bölünebilir ve unsurların birleştirilmesi için kullanılacak zaman üzerine kayıtlar yapılabilir.

K9'da görülen miktar, +C9*J9 formülüyle elde edilir. Yerleşim haritasının yapısı, maliyet analizi yapılmasını da sağlayabilmektedir. Örneğin, malzemelerin farklı bir satıcıdan alınması sonucunda bazı mallarda maliyet düşüşü ve diğer bazı mallarda maliyet artışları olduğunda, program, K19'da hemen gerecin yeni fiyatını saptayabilir. Buna dayanarak, ilgili malzemelerin yeni satıcıdan alınıp alınmayacağına karar verilebilir.

YERLEŞİM HARİTASININ BİLGİYAZIM ÖRNEĞİ

KUTUK : STOK KID 20305

KOD	KAYIT BOLUMU URETIM SURESI	FIYAT	ISLEMLER MAL GIRISI	MAL CIKISI	STOK MEVCUDU
C-01	2	2.37	200	85	115
C-02	4	3.25	60	21	39
C-03	1	0.58	85	30	55
C-04	6	1.64	91	51	40
C-05	5	2.03	250	200	50
C-06	2	0.25	300	54	246
C-07	3	0.78	35	15	10
C-08	1	1.25	90	30	60
C-10	6	3.12	100	80	20
	6		1201	566	635

DEGERLEME MAL GIRISI	MAL CIKISI	STOK DURUMU	URETIM MALIYETI 1. GEREC MIKTAR	FIYAT
474.00	201.45	272.55	10	23.70
195.00	68.25	126.75	15	48.75
49.30	17.40	31.90	3	1.74
149.24	83.64	65.60	21	34.44
507.50	406.00	101.50	7	14.21
75.00	13.50	61.50	16	4.00
19.50	11.70	7.80	11	8.58
112.50	37.50	75.00	9	11.25
312.00	249.60	62.40	2	6.24
1894.04	1089.04	805.00		152.91

larda (satış tahminleri, yatırım kârları vb.), yerleşim haritasının olanaklarından yararlanılır.

En basit yöntem ise, çözümlenecek akışın evrimini adım adım betimleyen bir yerleşim haritası yapmak ve çeşitli giriş verilerini girmektir. Bunların her biri, toplam yapının yeniden hesaplanması ve nihai sonucun görüntülenmesi sonucunu doğurur.

Bu sayfada altta, imal edilen parça sayısına bağlı olarak kârı hesaplayan yerleşim haritasının şeması görülmektedir. Örnekte, bilançonun yalnızca bir bölümünü oluşturan bazı kayıtlar gözönüne alınmıştır. Diğer faktörleri de gözönüne alan yeni satırların katılmasıyla, yapı kolayca genişletilebilir. Yeni satırların katılması, dört aydan daha uzun bir sürenin analiz edilmesi istendiğinde başvurulacak bir yoldur.

Genel olarak bilanço toplam gelir ile toplam maliyet arasındaki farktan oluşur.

KÂRIN HESAPLANMASI İÇİN YERLEŞİM HARİTASI ŞEMASI

Veri girişi

Maliyetlere ilişkin hesaplar

Gelirlere ilişkin hesaplar

Kâra ilişkin hesaplar

	A	B	C	D	E	F	G
1							
2							
3							
4			OCAK	ŞUBAT	MART	NİSAN	TOPLAM
5							
6		ÜRETİM MİKTARI					@SUM (C6. .F6)
7							
8		MALİYET					
9							
10		ÜRETİLEN	+C6*0.2+50				
11		SATIŞ	+C6*0.1				
12		PASİF FAİZLER	+0.18*(C10+C11)				
13							
14		TOPLAM MALİYET	@SUM (C10...C12)				
15							
16							
17		GELİRLER	+C6*1.02				
18							
19		KÂR	+C17−C14				
20		KÂR %	+100*(C19/C14)				

Örneğimizde üç maliyet türü öngörülmüştür:

— **Üretim maliyeti** (ÜRETİLEN), sabit bir oran (50) artı üretilen parça sayısına göre değişen orantılı bir oran (50 + 0.2 * parça sayısı).
— **Sürüm maliyeti** (SÜRÜM), üretim miktarıyla orantılıdır (0.1 * üretim miktarı).
— **Pasif faizler** (ACİYOLAR). Bu bölüm maddelerin ve diğer girdilerin alımı için gerçekleştirilmiş borçlardan gelen aciyoları içerir. Dolayısıyla önceki tutarlarla orantılıdır [0.18 * (ÜRETİLEN + SÜRÜM)].

Örnek olarak verilen bu formüllerin, incelenmekte olan her bir duruma göre uyarlanmaları gerekmektedir.

Toplam maliyet, bu üç verinin toplanmasıyla elde edilmektedir. Her ay için (6. satır) öngörülen üretim miktarı girilecek olursa, program bütün formüller için gerekli hesapları yapmakta ve sonuçları bildirmektedir. Bu rakamların anlamı kullanıcı tarafından benimsenen ölçü birimine göre değişir. Örneğin üretim miktarı, parça sayısı, ağırlık ya da hacim olarak ifade edilebilir, aynı durum para miktarlarını ifade eden rakamlar için de geçerlidir.

Altta, yerleşim haritasının kağıt üzerinde dökümünü gösteren tabloda, yalnızca tamsayılar görüntülenmektedir. Bu nedenle ondalık sayılar görünmemekte, fakat hesaplarda gözönüne alınmaktadır. Bu şekilde oluşan yapı saklanabilir ve birçok defa kullanılabilir. (Örneğimizde PROJEKSİYONLAR adlı bir kütük kullanılmıştır.) Bir ya da birçok giriş verisini (burada üretim miktarı) değiştirerek ve bunun toplam üzerindeki etkilerini çözümleyerek WHAT IF (...'sa ne olur) yöntemi uygulanır. Örneğin, eğer üretim Mart ayında tesadüfi olaylar sonucunda düşmüş ve üretim miktarı 75 parça olarak belirlenmişse, üç aylık toplam kâr (TOP sütunu) % 12'ye eşit olsa bile, o ayda % 11'lik bir zarar olduğu söylenecektir. Diyelim ki, çeyrek yılda varılması gereken ekonomik hedef % 20'lik bir kâr olsun; Mart ayındaki bu düşüşü telafi etmek için Nisan ayının üretimi ne olmalıdır? Cevabın elde edilebilmesi için en basit yöntem Nisan'da artan üretim miktarlarını girerek sonuçtaki değişimleri G20 hücresinde gözlemlenmesidir (toplam kâr yüzdesi). Bu süreç arzu edilene yakın bir son değer elde edilene kadar yinelenir. Son verilen tahmin, aranan veridir. Yapılması gereken hesapta, kendini yineleyen bir gidiş sözkonusudur; tahmini bir değer verilerek ilk sonuç hesaplanır ve buradan hareketle yeni bir tahmin için yeni bir hesaplamanın gerekip gerekmediğine karar verilir. Yerleşim haritalarında, yinelemeler, BASIC (veya başka bir dil) prog-

ÜRETİM ARTIŞI İLE BAĞLANTILI OLARAK KÂR ARTIŞI

```
2/94                KUTUK:   KAR

                    "
                    "  OCAK SUBAT MART NISAN        TOPLAM

URET.MIK.           "   104   96   75   150          423

                    "
MALIYET             "

                    "

URETILEN            "    71   69   65    80          135
SATIS               "    10   10    8    15           43
PASIF FAIZ          "    15   14   13    17           32
                    "   ----- ----- ----- -----      -----
TOPLAM MALIY.       "    96   93   86   112          386

                    "

BELIRLEN            "   106   98   77   153          434
                    "   ----- ----- ----- -----      -----
KAR                 "    10    5   -9    41           47
KAR %               "    11    5   11    36           12
```

ramlarında olduğundan çok daha kolaylıkla gerçekleştirilebilmektedir.

Gerçekten de, yerleşim haritasının temel özellikleri yapının tümünü sunması ve bunu yaparken de verilerin herhangi bir noktada girilmesine izin vermesidir. Örneğin Ocak ayındaki olası üretim değişikliklerinin etkilerini irdelemek istersek, yeni ve tam bir hesap elde etmek için C6 hücresinin içeriğini değiştirmek yeterli olacaktır. Veri girişinin yönelik olduğu ayı belirlemek ve bu aydan itibaren tüm hesapların yapılmasını bir BASIC programıyla gerçekleştirmek için çok sayıda komut gerekecektir.

Yerleşim haritasının sağladığı bir başka avantaj, işlem formüllerinin değiştirilebilmesi imkanıdır. Sürüm maliyetlerinde ortaya çıkan bir değişikliğin, kâra etkisini incelemek istersek, yalnızca C11 hücresinin (ve diğer aylarda da karşılık gelenlerin) içeriğini değiştirmekle yetinebiliriz.

BASIC'te ise programın tümünün değiştirilmesi gerekir, çünkü yerleşim haritasına benzer bir yapı şekli elde etmek için program tümüyle parametrelendirilmiş bir şekilde yazılmıştır. Çeviricisiz diğer dillerde, sonuçlar daha da ağırdır. Ancak, bundan yerleşim haritası tekniğinin, programlama dillerinin yerini alabileceği anlaşılmamalıdır. Daha karmaşık sorunlarda ya da farklı mantıksal işlemler gerektiren uygulamalarda yerleşim haritası tekniği uygun ve kullanışlı değildir.

Yukarıda verilen örnek kullanıcı tarafından yönetilen bir karşılıklı ilişkiler durumunu ele almaktadır. Hesabın yinelenmesi ve sonucun geçerliliğinin kontrolü yalnızca kullanıcıya bağlıdır. Daha karmaşık uygulamalarda iki güçlük ortaya çıkmaktadır: 1. Her iki yineleme arasında değişkenler uygun olmayan şekilde büyümektedir. 2. Tatmin edici bir sonuç elde etmek için gereken yineleme sayısı, çok yüksek olmaktadır. Bu sorunları aşmak için, bazı yerleşim haritası türlerine yinelemelerin yürütülmesini kolaylaştıran özel fonksiyonlar katılmıştır.

Yinelenen İşlemler İçin Fonksiyonlar

Yerleşim haritasının basit biçimlerinde işlem, her halûkarda, kullanıcı tarafından üretilen güdümler aracılığıyla gerçekleştirilir. Daha gelişmiş biçimlerinde ise yinelenen işlemler için daha özgül fonksiyonlar bulunmaktadır. Bunların en önemlileri şunlardır:

ITERCNT ( ). Bu fonksiyon yinelenmekte olan değeri, önce 1 ile başlayarak değiştirir. Burada herhangi bir parametre yoktur; çağırmada kullanılan ( ) simgesi bir parametrenin yarı anlamını taşır. En önemli kullanımı, yinelenen programın bitiminde sınama hizmeti görmesidir. Örneğin, ITERCNT( )=10 koşulu, onuncu yinelemede işlemin durdurulmasını sağlar.



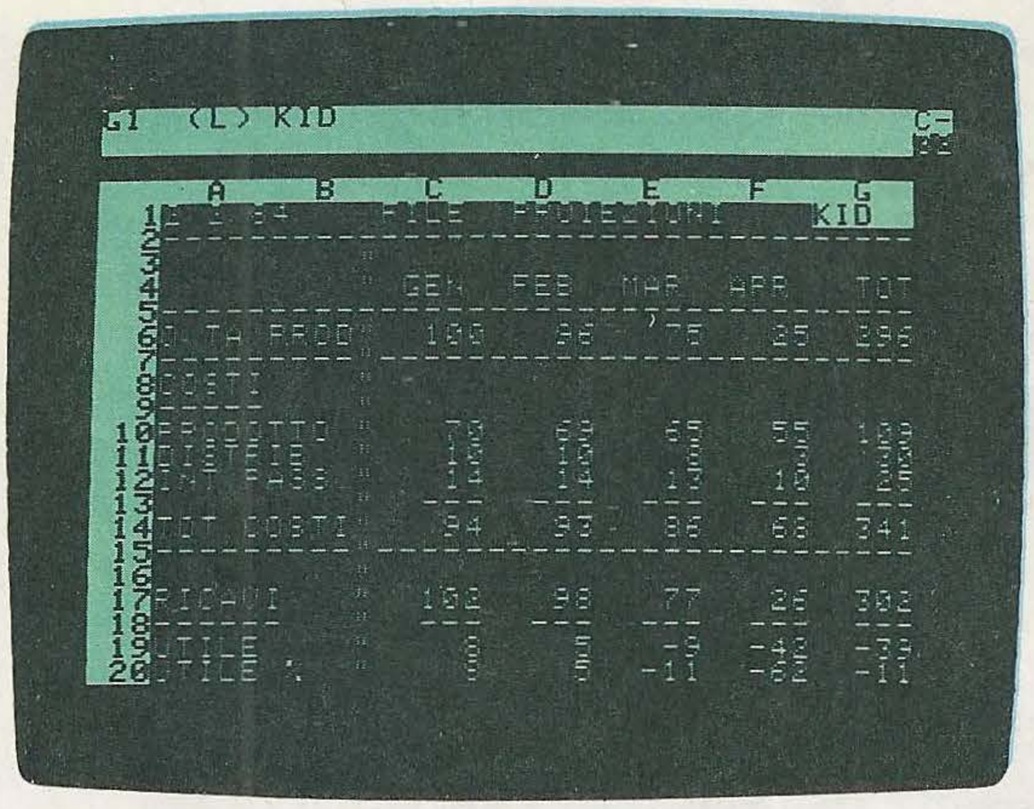
Elektronik gereçlerin son montajı.

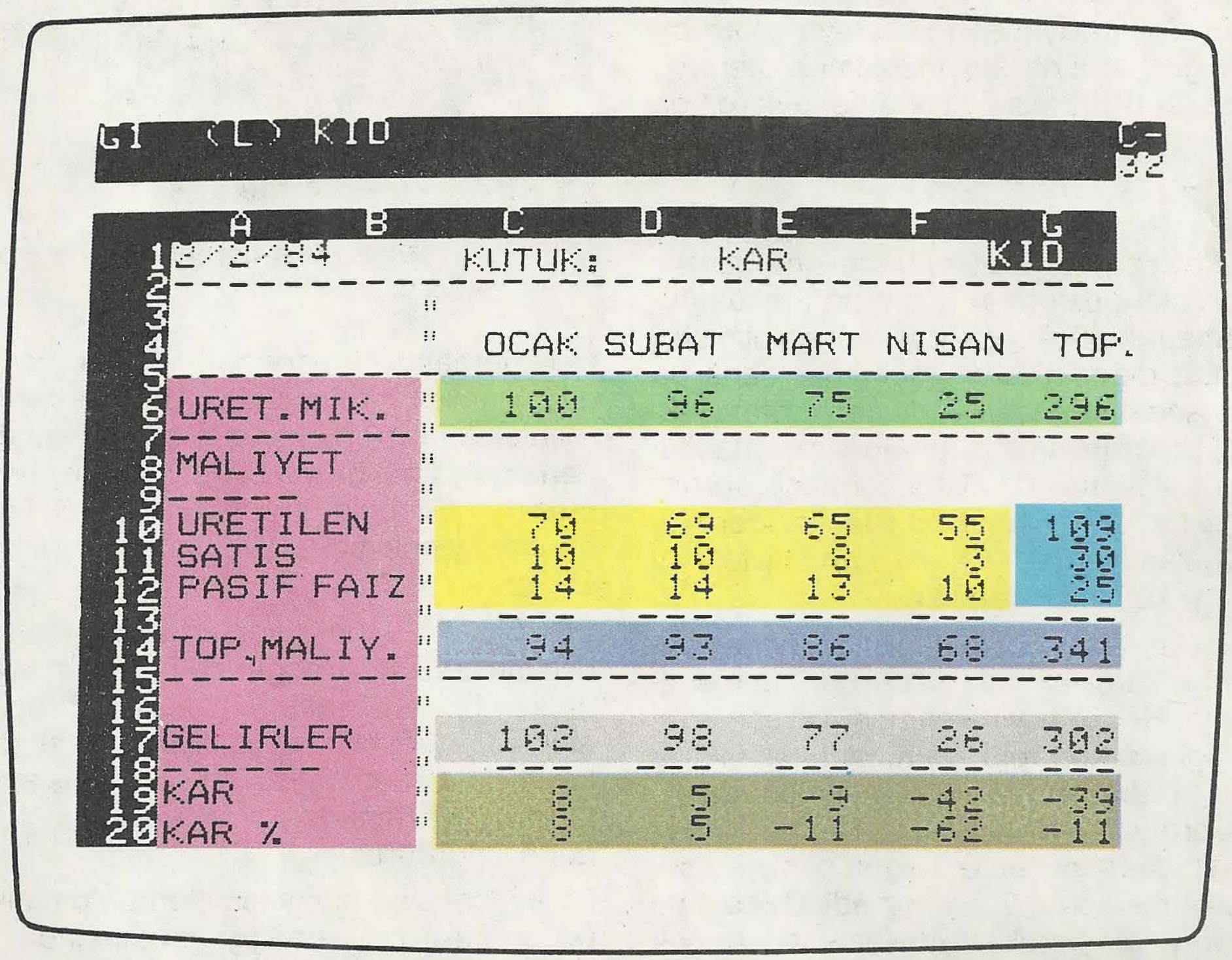
DELTA ( ). Bu fonksiyon art arda gelen iki yineleme arasındaki değerlerin değişimi sırasında kaydedilen azami mutlak değeri değiştirmektedir. Bir yakınlaşma ölçütü olarak da kullanılabilir. Örneğin DELTA ( ) < 0.1 işlemi, değişimlerin azami değeri 0.1'den küçük olduğunda durdurur.

Bu iki fonksiyon, yinelemelerin otomatik olarak gerçekleştiği yerleşim haritalarında işlemleri durdurmak için mutlaka gereklidirler. Bu durumlarda yinelenen işlem sürecini otomatik olarak harekete geçiren (ve genellikle ITERATION olarak adlandırılan) bir güdüm mevcuttur. Yerleşim haritası tekniğinin diğer türlerinde iki yöntem kullanılmaktadır. Ya, ITERATION güdümüne benzetmek için özel bir takım aldatmacalara başvurulacak, ya da bir yinelemeden öbürüne, (bağımsız) değişkenin yeni değerini giren ve işlemi yeniden harekete geçiren, kullanıcının kendisi olacaktır. Bu durumlarda, denemenin sonu için bir otomatikleştirme gerekmez: kullanıcı, işlemin ne zaman kesileceğini kendisi belirleyebilir.

DEĞİŞİKLİKLERİN ANALİZİ: WHAT IF METODU

Bu sayfada, yerleşim haritasında değişikliklerin analizi için WHAT IF yönteminin uygulamasını görüyoruz. Ekran yine, yerleşim haritasının daha önce verilmiş olan yapısını göstermektedir. Kullanılan tüm sütunları birarada görebilmek için (hesaplamalar için değil, yalnızca gösterim amacıyla), tamsayılı yazı düzeni seçilmiştir.





A sütununda bulunan açıklamalar, hücrelerin genişletilmesinin izin verdiğinden daha fazla yer işgal etmektedirler. Bu yüzden, A ve B sütunlarının komşu hücrelerinin kullanılmasıyla yazılmışlardır.
Bunun sonucu olarak, hücrelerin içeriklerinin art arda sıralanması imkânsızlaşmıştır. Çünkü böyle bir durumda, açıklamaların parçalanması tehlikesi doğmaktadır.

6. satırın hücreleri giriş için kullanılmaktadır.

C 10 ve F 12 köşe noktalarının sınırladığı alandaki hücreler, maliyete ilişkin hesapları içeren hücrelerdir.

G sütununun 10., 11. ve 12. hücreleri, SUM fonksiyonunu kullanan ÜRETİLEN, SATIŞ ve PASİF FAİZ satırlarının toplam değerlerinin hesaplanmasında kullanılmaktadır.

14. satırın hücreleri, bunun üstündeki üç satırın toplam değerlerinin hesaplanmasını içermektedir.

17. satır, gelirlerin hesaplarını içermektedir.

Son iki satırın hücreleri, toplam kârların ve yüzdelerin hesaplanması için gereken formülleri içermektedir.
WHAT IF yöntemi, yeni bir veri kaydının 6. satırın herhangi bir hücresine girilmesiyle kolayca kullanılabilir. Anında yerleşim haritasının hesaplanması gerçekleşir.

Yerleşim Haritasının Gelişimi

İncelediğimiz güdüm ve fonksiyonlar yerleşim haritası tekniğinin basit şekillerine ilişkin olanlardır. BASIC'te olduğu gibi, bu programlar da kısa zaman içinde daha mükemmel ve yeni güdüm ve fonksiyonlarla zenginleşmiş yeni versiyonlara kavuşmuşlardır.

Örnek olarak aşağıdaki tabloda Visicalc'te öngörülen fonksiyonları ve Multiplan'dakilerle karşılaştırdık. Versiyonlar, Olivetti M20 için geçerlidir.

Temel fonksiyonların, bazen farklı bir simgeleştirme biçimi kullansalar da, değişmeden kaldıkları görülmektedir. Ama diğer yandan yeni fonksiyonlar da ortaya çıkmaktadır. (Sayfa 846'da soldaki tablo.) Sağdaki tabloda ise Multiplan ve Visicalc güdümlerinin bir karşılaştırması verilmektedir. Burada da temel güdümler her ikisi için de aynı kalmakla birlikte, Multiplan için yeni versiyonlara raslanılmaktadır.

MULTİPLAN FONKSİYONLARI VE KARŞILIK GELEN VISICALC FONKSİYONLARI

Multiplan	Visicalc
ABS(N)	@ABS( )
PR( )/2-ATAN(N/SQRT(1-N*N))	@ACOS(N)
AND(Liste)	@AND(Liste)
ATAN(N/SQRT(1-N*N))	@ASIN(N)
ATAN(N)	@ATAN(N)
AVERAGE(Liste)	@AVERAGE(Liste)
INDEX(Alan, Göstergeler)	@CHOOSE
COS(N)	@COS(N)
COUNT(Liste)	@COUNT(Liste)
	@ERROR
EXP(N)	@EXP(N)
FALSE()	@FLASE
IF(1,v1,v2)	@IF(1,v1,v2)
INT(N)	@INT(N)
ISERROR(N)	@ISERROR(N)
ISNA(N)	@ISNA(N)
LN(N)	@LN(N)
LOG10(N)	@LOG10(N)
LOOKUP(N, Alan)	@LOOKUP(N),range)
MAX(Liste)	@MAX(Liste)
MIN(Liste)	@MIN(Liste)
NA( )	@NA
NOT(1)	@NOT(1)
NPV(dr, Liste)	@NPV(dr, range)
OR(Liste)	@OR(Liste)
PI()	@PI
SIN(N)	@SIN(N)
SQRT(N)	@SXRT(N)
SUM(Liste)	@SUM(Liste)
TAN(N)	@TAN(N)
TRUE( )	@TRUE

ÖZEL MULTIPLAN FONKSİYONLARI

Fonksiyon	Açıklama
COLUMN( )	Kullanılan sütunun numarası.
DOLLAR(N)	N'nin içeriği, dolar cinsinden ifade edilen tutar olarak görüntülenir. N negatif ise, rakam paranteze alınır.
FIXED(N,d)	N'nin d ondalık haneyle formatlanmış çıkış düzeni.
LENT(T)	T metninin karakter olarak uzunluğu.
NID(T,s,c,)	T metninin s pozisyonundan başlayan ve c kadar ilerlenen karakter zincirini üretir.
MOD(N1,N2)	N1/N2'den artan.
REPT(T,N)	T metninin n sayıda tekrarından oluşan metin.
ROUND(N,d)	N'nin d sayıda ondalık haneye yuvarlanmış değeri.
ROW( )	Kullanılan satırın numarası.
SIGN(N)	N'nin negatif, sıfır veya pozitif olmasına göre —1,0 veya +1,0.
STDEV(Liste)	Listedeki değerlerin standart sapması.
VALUE(T)	T mletninin sayısal değeri; T, bir dolar tutarı da dahil olmak üzere, herhangi bir yazı düzeniyle kullanılabilir.

MULTIPLAN GÜDÜMLERİ VE KARŞILIK GELEN VISICALC GÜDÜMLERİ

Multiplan	Visicalc
Blank	/B
Transfer Clear	/C
Delete Column, Delete Rows	/D
Edit, Alpha	/E
Format Cells	/F
Format Width	/GC
Format Default	/GF
kullanılmaz; metne bakın	/GO
Option	/GR
Insert Columns, Insert Rows	/I
Move Columns, Move Rows	/M
Print	/P
Copy	/R
Transfer Load	/SL
Window Split Titles	/T
Option	/V
Window Open, Window Split, vb.	/W
Window Link	/WS,/WU
Goto Row-Col	
NEXT WINDOW key	;
RECALC key	!
use references	
REPT	/
Format Options	
Help	
Lock	
Name	
Sort	
Window	
External	

Multiplan güdümlerinin seçimi için güdümün adının yalnızca ilk harfi kullanılır.

Bugnların arasından en önemlilerini şöyle sıralayabiliriz:

Format Options: Hücre içeriklerinin görüntülenmesi için yazı düzenlerinin birkaç seçeneğini seçer.

Help: Ekranda, programın kullanımıyla ilgili bir dizi açıklama sunar.

Window: Ekrandaki pencerelerin yönetimini sağlar. Bu güdüm sayesinde ekran ''pencerelere'' bölünür, bunların her biri bağımsız bir yerleşim haritası olarak kullanılabilir.

External: İşyerinin dışında bir yerlerde saklanmış verilerin kullanılmasını sağlar. Örneğin bu, daha önce tanımlanmış bir pencere olabilir.

Lock: Bir (ya da birden çok) hücrenin değiş-

tirilmesini engeller: Yerleşim haritasında araya sokulmuş veri girişi hücreleri ya da işlem hücreleri ortaya çıktığında çok yararlıdır. Verileri girmek için imlecin yerini değiştirdiğimizde pozisyonları rahatlıkla birbirine karıştırabilir ve bir değeri, daha önceden bir formülün saklanmakta olduğu yere yazabiliriz; böyle bir durum formülün yitirilmesine yol açar. Oysa, hücreyi koruma altına alacak olursak, bir veri kaydının girişi engellenmiş, dolayısıyla olabilecek hatalardan kaçınılmış olacaktır.

Name: Birçok hücreyi aynı simgesel ad aracılığıyla tanımlamayı sağlar. Bu güdümün harekete geçirdiği mantık, BASIC'te DIM fonksiyonu ile yapılanın aynısıdır. Bazı yerleşim haritası tekniklerinde aynı simgesel adla tanımlanmış hücreler topluluğu ranj (''range'') olarak adlandırılır. Bu hücrelere yönelik tüm fonksiyonlar, parametre olarak adres yerine ranjın (range) simgesel adını belirterek yerine getirebilirler.

Sort: Bir sütunun içeriğinin düzenlenmesini sağlar. Bu düzenleme sayıları veya metinleri kapsayabilir; yukarıdan aşağıya ve aşağıdan yukarıya doğru gerçekleştirilebilir.

Bazı yeni yerleşim haritası versiyonları (örneğin Lotus 1-2-3), grafik çizme ve bir kütüğü hücrelerin içerikleriyle yönetme olanaklarını da sağlarlar.

Yerleşim haritasını gerçekleştiren program çeşitleri hakkında yaklaşık bir fikir vermek amacıyla 848. sayfada bir Multiplan uygulaması örneği (Olivetti M20 versiyonu) betimlenmiştir.

Kütüklerin Yönetimi ile İlgili Programlar

Genel uygulama programların ikinci türü kütüklerin yönetimiyle ilgilidir. En basit programlar genellikle yalnızca tek bir kütüğün ve kısıtlı sayıda fonksiyonların kullanımını öngörmektedir. Daha karmaşıkları ise, BASIC diline çok yakın programlama olanakları sunarak, gerçek bir veri bankasının idaresine olanak sağlamaktadır.

Bu programlar tarafından yerine getirilen en önemli işlevler şunlardır:

— Kütüklerin oluşturulması ve alanların tanımlanması
— Verilerin girilmesi ve güncelleştirilmesi
— Arama
— Ayıklama
— Verilerin işlenmesi
— Tabloların bilgi yazımı

Kütüklerin Oluşturulması ve Alanların Tanımlanması

Bir veri bankası oluşturmak için ilk olarak devreye girmesi gereken fonksiyondur. Kullanılacak parametreler şunlardır:

— Kütüğün (ya da) kütüklerin adı
— Disket birimi (işletim sistemine göre A/B veya 0/1)
— Değişik alanların ad ve nitelikleri

Güdüm sözdizimi genellikle şu şekildedir:

CREATE X : AD

Burada X seçilmiş disket birimi, (bazen bulunmayabilir, bu durumda sistem kendiliğinden varmış gibi kabul eder). AD da oluşturulacak kütüğün adıdır. Program, disket üzerinde kullanılabilecek alan olup olmadığına bakıldıktan sonra, adı bir dizinde saklı tutar ve veri kaydını oluşturacak olan alanların niteliklerini sorma işlemine geçer. Bu programlarda bir ve aynı kütüğe ait veri kayıtları aynı uzunluktadır.

Bu uzunluk bayt cinsinden ifade edilmektedir ve açıklama girildikçe, veri kaydını oluşturan alanların uzunluklarının toplamından oluşur. Genellikle bir kez veriler girilmeye başlandıktan sonra, alanların uzunluklarını veya tiplerini, daha önceden girilmiş verileri yitirmeksizin, değiştirmeye imkan kalmamaktadır. Değişik alanları tanımlayan parametreler şunlardır:

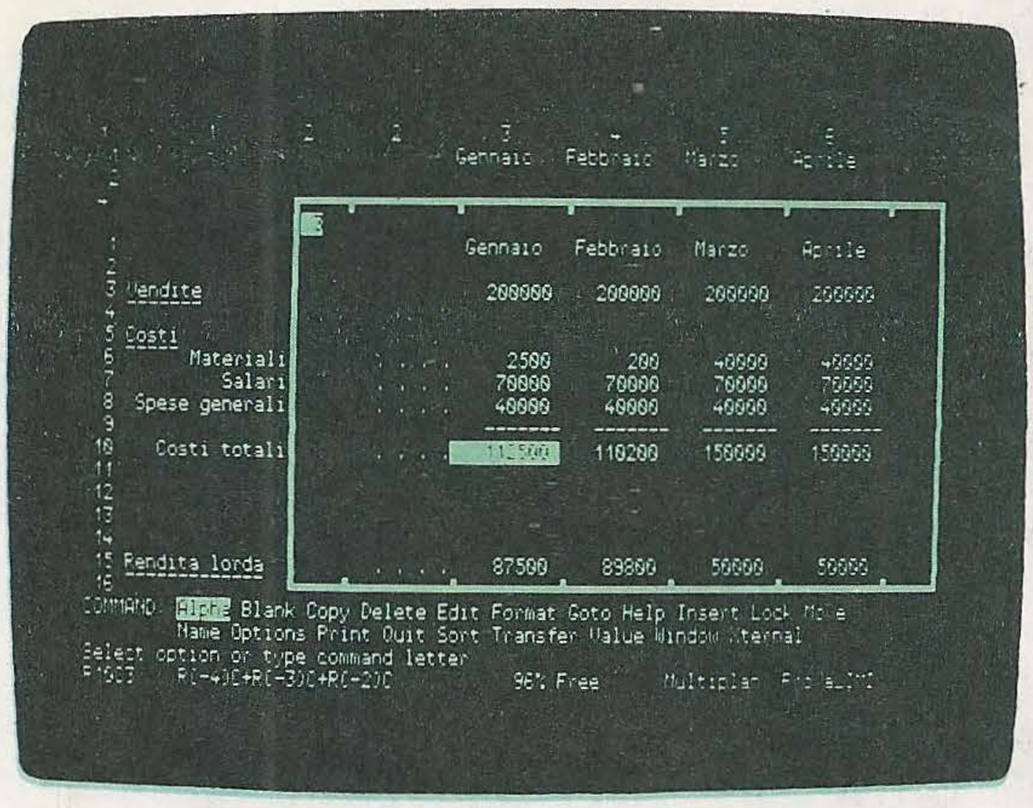
— Ad: Kullanıcının ve programın, her bir alanın içeriğini belirlemek için kullandıkları simgesel ad.
— Tip: Alanın tipini tanımlar (harfsayısal ya da sayısal).
— Uzunluk: Alanda öngörülen azami karakter sayısı.

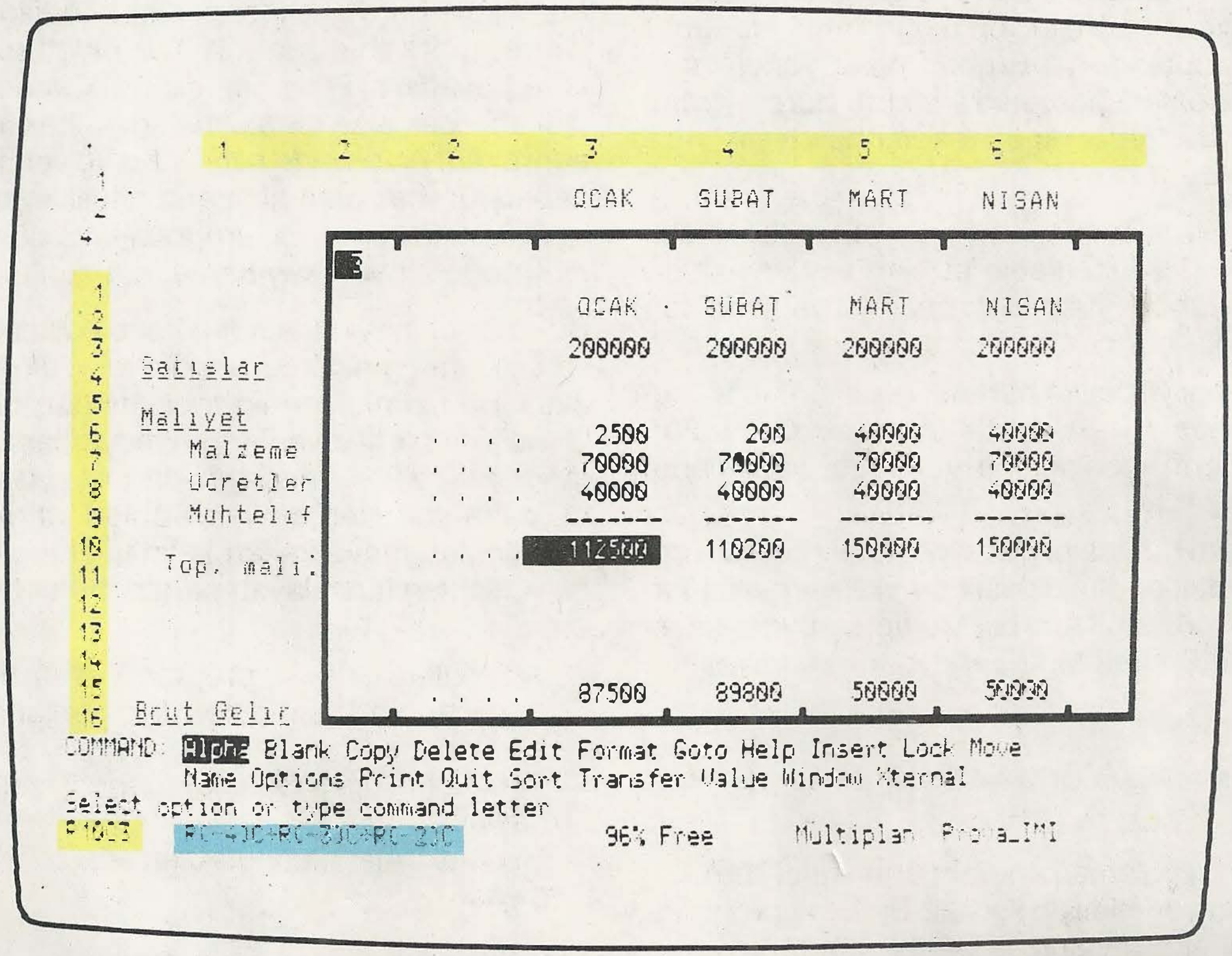
Sayısal alanlar için ayrıca ondalıklı sayıların miktarı da gereklidir. Toplam uzunluğun, ondalık sayılara ayrılmış alanı da hesaba kattığını unutmamak gerekir. Örneğin üç tamsayı ve iki ondalık sayı içeren bir sayısal alan altı karakter işgal edecektir (XXX.XX). Bir alan sayısal olarak saptandığında, giriş aşamasında otomatik olarak bir veri denetimi oluşmaktadır (sayıların sınanması); bu durumda yalnızca sayısal değerler kabul edilecektir.

Bir veri kaydında tanımlanabilen alan sayısı, program türüne göre değişiklik gösterir. Genellikle bu sayı 30'un üstündedir ve böylece çoğu talepleri karşılamaya yetmektedir.

MULTIPLAN'IN YERLEŞİM HARİTASI

Multiplan'ın yerleşim haritası, bazı açılardan, şimdiye kadar öğrenmiş olduklarımızdan farklıdır. Örneğin, güdümler ve imleç tarafından işaret edilen satırın içeriği altta görüntülenmektedir.





En belirleyici farklılık, hücrelerin adreslenmesine ilişkindir. Adresler, önlerinde Z ve S harfleri bulunan iki sayıdan oluşur. Z10S3, 10. satırla 3. sütunun kesiştiği noktada bulunan hücrenin adresidir. Fotoğrafta, imleç bu noktada bulunmaktadır.

Formüllerde kullanılan göreli adreslemeler, formülü içeren hücrenin bulunduğu pozisyondan kaydırma ile ifade edilir. İmleç, Z10S3'te bulunuyorsa, Z6S3 hücresine gönderme yapan simge düzeni, Z(—4)S'tir. Böylece, hücrenin aynı sütunda (S), fakat 4 satır üstte bulunduğu ifade edilir.

Yukarıya veya sola kaydırmalar negatif sayılarla, sağa veya aşağıya kaydırmalar ise pozitif sayılarla ifade edilir. Z10S3 hücresine yerleştirilen formül, Z6S3, Z7S3 ve Z8S3 hücrelerinin içeriklerinin toplamını hesaplar.

Verilerin Girilmesi ve Güncelleştirilmesi

Kütüğün yapısı tanımlandıktan sonra, verilerin girilmesi işlemine geçilmektedir. Normal olarak program, alan adlarını ve her birinde ayrılan kullanılabilir alanları görüntüleyen bir ekran maskesi sunmaktadır.

Daha gelişmiş programlarda, kullanıcı, maskenin görüntülemesi için bazı unsurları özellikle belirtebilir: Örneğin, her bir alan için pozisyonu veya gösterim şekli (tersine çevrilmiş, altı çizili vb.).

Programlar tarafından benimsenen gösterim türleri arasındaki temel fark, ekran maskesi tarafından yönetilen farklı azami sütun sayılarından kaynaklanmaktadır. Bazı programlar tüm alanları tanımlanma sıralarına göre tek bir sütunda sunarlar (her bir satır için tek bir alan).

Diğerlerinde ise farklı alanlar, herhangi bir gösterim sıralaması seçilerek, tek ve aynı satırda (bu durumda çok sütunlu bir maske elde etmiş oluruz) sunulabilir.

Veri girişi aşaması, zaten mevcut olan bir kütüğün veya yeni oluşturulmakta olan bir kütüğün üzerinde çalışılıyor olunmasına göre farklı iki görünüm arzeder. Birinci durumda yeni veriler ekler olarak ele alınmakta ve aktarmayı devreye sokan güdüm normal olarak APPEND olmaktadır. İkinci durumda ise, kütük oluşturma aşamasının bitiminde giriş aşamasına geri dönen, programın kendisidir. Biçimlerinin farklılığının dışında, iki yöntem de aynı işlevi yerine getirmektedir: Verileri, her yeni veri kaydına artan bir kayıt numarası vererek kütüğün düzenlenmesi sırasında tanımlanan yazı düzenine sokmak.

Bir üçüncü veri girme yolu araya sokmadır. Normal olarak bu fonksiyonu devreye sokan anahtar sözcük INSERT'tür; bu güdümle veriler istenilen pozisyonda araya sokulabilir. Giriş verilerinin yazılması aşamasında, APPEND'deki gibi oluşturma durumunda, veriler giriş sıralarına göre birbiri ardınca saklanmaktadırlar; buna karşılık bazı uygulamalar -örneğin, arama aşamasını hızlandırmak için yapılanda olduğu gibi- araya sokmaları herhangi bir ara pozisyonda kabul edebilmektedirler (adreslendirilmiş araya sokma). Adreslendirilmiş araya sokma için kullanılan güdümün sözdizimi kullanılan programa bağlıdır.

Programların büyük bir bölümü iki yöntem kullanmaktadır: Gösterge kullanma ve doğrudan adresleme.

Birinci durumda, girilen her bir veri kaydı kendi yerleşim adresini de içermektedir (kayıt numarası). Birçok programda kütük içinde belirli bir kayıt üzerine yerleşmek mümkündür. Daha sonra verilecek bir araya sokma güdümü, yeni verinin böylece tanımlanmış olan yere yazılmasını sağlayacaktır. Doğal olarak, takip eden pozisyonlarda bulunan tüm veriler kayacak ve yeni verinin araya sokulmasından önce bulunulan pozisyon bir sonrakine aktarılacaktır. Birçok veri bankası türünde belirli bir veri kaydına ulaşma GOTO n şeklinde bir güdümle gerçekleştirilmektedir. Buradaki n, veri kaydının numarasıdır. Örneğin,

GOTO 3
INSERT BEFORE

güdüm dizisi, göstergeyi 3 numaralı veri kaydına yerleştirmekte ve yeni veriyi (BEFORE)'dan önce araya sokmaktadır. Böylece gösterge, 2 ve 3 numaralı eski veriler arasına yerleşecek ve 3 numaralı veri kaydını işgal edecektir. Daha önce 3 numaralı veri kaydı işgal eden veri ise 4 numaraya kayacak, bu kayma kütüğün bitimine kadar sürecektir. Mevcut veriler üzerinde, bu gösterge sistemi yardımıyla düzeltmeler yapmak da mümkün olmaktadır. Bu programlarda çok geçerli olan bir güdümde EDIT n'dir. Bu güdüm, belirlenmiş olan n kaydının içeriğinde gerekli düzeltmeleri yapmaktadır.

Bu verilerin, veri kaydı numaraları temelinde göstergelenmesi yöntemi, yalnızca bazı özel durumlarda gerekli olmaktadır. Normal olarak bu yöntemin kullanılması zorluk yaratmaktadır. Çünkü, her veri ifadesi için veri kaydının göreli numarasını içeren bir ilişki tablosu kullanmayı zorunlu kılmaktadır. Ayrıca bir araya sokma veya ekleme işleminden önce geçerliliği olan bir ilişki sistemi kullanılacak olursa, hata yapma olasılığı artmaktadır. Bu programların sunduğu, arama olanaklarını kullanmak daha yararlı yoldur. Örneğin belirli bir alanın içeriğini arama işlemi, FIND (bulmak) olarak adlandırılmaktadır; bu terim genellikle güdümü kısaltması olarak kullanılır.

Arama (FIND)

Arama fonksiyonuyla veriler, bir ya da birden çok alanda içerilen değer temelinde seçilmektedir. Kütük yönetimiyle ilgili temel fonksiyonlardan biridir. Veri saklama, özellikle belirli bir veriyi elde edebilmek veya olası veri eklemelerini sağlayabilmek için kullanılacak olan arama işlemlerini mümkün kılması açısından son derece yararlıdır. Bu kategoriye giren prog-

ramların tümü büro işlerinin otomatikleşmesine katkıda bulunmaktadır. Bir büronun temel işi diğer bürolardan ve çevre birimlerden gelen bilgilerin toplanması ve bu bilgilerden firma içi birtakım karar ve stratejilerin planlanmasına temel oluşturacak ilişkiler üretmektir.

Bu süreç birbirini takip eden ve tek tek detaylardan kapsamlı bir durum oluşturmayı sağlayan veri eklemeleri aracılığıyla gerçekleşir.

Seçim mantığı, yararlanılan uygulamaya bağlıdır. Hatta karmaşık bir dizi seçim, birkaç alanın değerlerini kullanan tek bir seçime indirgenebilir. Diğer bir deyişle, böyle bir kütüğün temel işlevi aramadır. FİND fonksiyonu, veriler için geçerli olan düzenlenme biçimine göre üç yoldan yürütülebilir:

- Eğer kütükte ulaşım anahtari yoksa ve daha önce bir düzenlemeden geçmediyse, arama, bütün veri kayıtları okunarak ve istenilen değerin (ya da sınırın) bulunduğu seçilmiş bölümün içeriğinin bulunduğu bu kayıtlarla karşılaştırılarak yürütülür.
- Eğer kütük bir düzenlemeden geçtiyse kodla durdurma tekniği benimsenebilir. Bu durumda arama önceki gibi yürütülür, fakat istenen alanın bulunmasıyla sona erer.
- Eğer kütük bir ya da birden fazla göstergeye sahipse, daha hızlı bir yöntem kullanmak mümkündür. Kütüğün oluşturulması aşamasında, belirli bir alanın anahtar alan seçilmesini sağlayan bir gösterge oluşturulabilir. Bu durumda kayıtlara ulaşmayı sağlayan anahtarların sayısı aramanın sınırını belirleyecektir.

Bazı durumlarda, birden çok anahtar kullanılabilir ve dolayısıyla çok sayıda ulaşım olanağı elde edilir; ancak bunun bazı sakıncalarını da gözönünde bulundurmak gerekir. Bu yöntem ancak, kullanılan anahtara eşit sayıda göstergelendirilmiş kütük üretildiği takdirde kullanılır. Bu da bellek ve çalışma zamanı açısından israf anlamına gelir, çünkü girilen her yeni veri tüm göstergelerin yeniden güncelleştirilmesini gerektirmektedir.

Büyük sistemlerde bellek kapasitesi fazla bir sorun olmaz, göstergelerin güncelleştirilmesi için gerekli zaman ise pek önemli değildir. Ancak kişisel bilgisayarlarda daha büyük bir ekonomiklik düzeyine ihtiyaç vardır.

Diğer bir sakınca ise, bu tür bir yapının fazla esnek olmamasıdır. Önceden bir dizi gösterge oluşturulduğunda, en önemli durumlar kapsanmış olur, ancak olası durumların tümünü öngörmeye olanak yoktur. Uygulamalar geliştikçe, yeni bir ihtiyaç ortaya çıkar ve öngörülmemiş, göstergesi üretilmemiş bir sıralamanın kullanımını gerektirebilir.

Bu düşünceler, uzmanları göstergelerin kullanımında yeni bir anlayışa yöneltmiştir. Başlangıçta hiçbir gösterge oluşturulmaz ve veriler, hiçbir anahtar yokmuş gibi ele alınırlar. Kullanılacak alan ya da alanlar için bir gösterge üretilmesi ancak birleşme anında veya herhangi bir seçme işlemi sırasında yapılır. Bu yöntem hemen hemen kütük yönetimine ilişkin tüm programlarda kullanılabilir.

Yerine getirilecek işlevlerin akışı şu şekilde tanımlanabilir:

— Kütüğün oluşturulması
— Verilerin girilmesi
— Seçme anahtarlarının (anahtar alan) tanımlanması
— Anahtar alanın içeriğine dayanarak göstergenin oluşturulması.

Bu işlemlerin bitiminde, verileri her bir alana dayalı olarak göstergeleyen bir kütük elde edilmiş olur: Göstergenin kullanılmasıyla, tüm arama ve seçme işlemleri yürütülebilir. Kütüğün oluşturulması için kullanılan sözdizim şu şekildedir:

INDEX ON ''ALAN'' TO ''KUTUK''

Burada ALAN, göstergelenecek alanın simgesel adıdır ve veri kütüğünün oluşturulması aşamasında belirlenen alan adlarından biriyle uyuşması gerekir. KUTUK ise göstergelenmiş kütüğün adıdır. Bir seçim veya arama yapmak için, sisteme kullanılacak göstergenin adını vermek gerekir (bunlardan bir çoğu aynı anda varolabilir).

Bu durumda sözdizim,

USE ''AD'' INDEX ''ALAN''

olmaktadır.

Anlamı: ALAN'ın içeriği üzerine göstergelenmiş AD kütüğünü (INDEX) kullanın (USE). Bu güdümle sisteme, AD göstergelenmiş kütüğünü çağırma ve ALAN simgesel adını taşıyan alan üzerinde (bir sonraki güdüm tarafından tanımlanmış olan) arama işlemlerini yürütme buyruğu verilmektedir.

Sonraki komut, karşılaştırma yapılacak değeri sağlayacaktır.

Sözdizimi şöyledir:

FIND XXXX'dır.

Burada XXXX, aranması gereken veridir (anahtar).

Örneğin,

1. INDEKS ON AD TO ANAGR
2. USE ANAGR INDEKS AD
3. FIND Ahmet Caner

güdüm zinciri şu işlevleri yürütür:

1. Kütüğün oluşturulması aşamasında tanımlanmış olan AD alanının içeriğine ilişkin ANAGR adlı göstergelendirilmiş yeni bir kütük oluşturulur.
2. ANAGR kütüğünü AD alanının göstergesi (INDEX) olarak kullan (USE)!
3. AD alanındaki Ahmet Caner verisini içeren tüm kayıtları ara!

Güdümler, belirli bir program tarafından kullanılan biçimde betimlenmektedir.

Diğer durumlarda aynı güdümlerin sözdizimi farklı olabilir, ancak yerine getirilen fonksiyonlar genellikle aynıdır.

Sıralama (SORT)

Kütük yönetimine ilişkin tüm programlarda SORT (sıralama) fonksiyonu bulunmaktadır. En çok kullanılan yöntem, veri ve veya adreslerin istenilen sıralama içinde yeniden yazıldıkları bir tamamlayıcı kütük oluşturmaktır.

Kullanıcı, bu kütüğün gerçek içeriğini bilmek zorunda değildir. Eğer program SORT fonksiyonunu yerine getiriyorsa, girilen veriyi kütükten kayıtların adresi temelinde bulabilecektir.

Kullanıcının tek yapacağı, sıralanması gereken alanı ve sıralanmış verileri (veya adreslerini) içermesi gereken kütüğün adını belirleyerek fonksiyonu harekete geçirmektir. Sıralama programının harekete geçirilmesi için kullanılan tipik şekil şöyledir:

SORT ON nnn TO mm

Fonksiyon, SORT güdümüyle harekete geçirilir; ON nnn ifadesi sıralama yapmayı arzuladığımız alanı (nnn kullanılan veri kaydında tanımlanmış alanlardan birinin simgesel adı olmalıdır), TO mm ifadesi ise çıkış kütüğünün adını belirtir. Örneğin,

SORT ON Adres TO Kutuk A

tümcesi Adres alanında yapılacak bir sıralama işlemini harekete geçirir ve çıkışları, bir A kütüğü içinde saklar. Bu kütüğü bir bilgiyazım için kullanmak istersek, verileri Adres alanının içeriğindeki sıralamayla elde ederiz. Veri kütüğü değiştirilmez ve giriş sırasını izler. Eğer veri kütüğünün baştaki sıralanmasını korumak istersek, sıralama programının çıkışı için başka bir kütüğün kullanılması gerekir. Böyle olmazsa, son işletilmiş olan sıralama programına bağlı olarak önemli bir hata riski doğar, çünkü veriler şimdi farklı pozisyonlardadır.

Verilerin İşlenmesi

Verilerin saklanması, hiçbir zaman başlı başına bir amaç oluşturmaz. Farklı alanların içeriklerinin herhangi bir şekilde işlenmesi gerekir. Birçok yönetim programında bu gereksinimi karşılamak için bağlantı hesaplama fonksiyonları öngörülmüştür. Normal olarak bu fonksiyonlar, karşılık gelen fonksiyonlardaki BASIC sözdizimlerini ve kuralları takip ederler. Ancak kullanılan programa özgü bazı güdümler tarafından yürürlüğe sokulurlar. En çok kullanılan güdümlerden biri REPLACE'dir. Herhangi bir alanın içeriğinin yerine, bir diğerini ya da bir işlemin sonucunu koymayı sağlar.

Örneğin,

REPLACE ALL Toplam WITH Miktar * Fiyat

tümcesi, bütün kütük içinde (REPLACE ALL) Toplam alanının içeriğinin yerine (WITH) iki diğer alanın çarpımının (Miktar * FIYAT) sonucunu koyar. Miktar ve FIYAT aynı veri kaydının alanlarına ilişkin olabilirler veya herhangi bir dış değişkeni (veri tabanına göre dış) belirtebilirler.

Bazı veri tabanı türlerinde REPLACE güdümü (ve diğerleri) koşula bağlı olarak kullanılabilir. Bu durumda bir önceki işlem şu şekilde yazılacaktır:

REPLACE ALL Toplam WITH Miktar *
Fiyat FOR Toplam = 0

FOR TOPLAM = 0 bölümü, Toplam alanının içeriğinin yerine bir başkasının konulmasını koşula bağlar. Eğer bu alan henüz hesaplanmadıysa (içeriği 0'a eşittir) işlem yerine getirilir; bunun tersi durumda veri kaydı işlenmez ve program bir sonraki veri kaydına geçer.

Daha karmaşık programlar için BASIC'inkine çok benzeyen komutlar öngörülmüştür. Bunlarla veri kayıtlarının içerikleri için bağlantı-ve işleme alt-programları yazmak mümkündür. Sori örnekte verilen komut, basit olmakla birlikte, yine de bir program biçimidir. BASIC'te şu şekilde olurdu:

IF Toplam = 0 THEN Toplam = Miktar * Fiyat

Kuşkusuz sözkonusu olan yine aynı komuttur, sadece biçimi değişmiştir. Aslında önemli bir

Büroların Otomasyonu (3)

Bir büroda etkinliklerin otomasyonu amacıyla bir sistem tanımlamak, geleneksel bir ''işyeri'', örneğin yazı masası ve bununla ilgili olası diğer düzenlemeleri tasarlayabilmek için, kullanılacak alışılmış gereçler, ilgili işlevler, farklı etkinlikler arasındaki karşılıklı ilişkiler ve otomasyon ihtiyacı hakkında derinlemesine bir incelemeden yola çıkmak gerekir.

Otomasyonu gerçekleştirmiş bir işyeri, daha önceleri elle, sözel olarak ya da elektronik bir belleği olmayan makineler tarafından gerçekleştirilen işlemlerin aynılarının ve eşdeğerlerinin (sözlü veya yazılı bilgilerin oluşturulması ve/veya güncelleştirilmesi ve/veya değiştirilmesi, iletilmesi, arşivlenmesi, bulunması, kopya edilmesi) elektronik olarak (yani büro personeli müdahalesi ve kağıt üzerinde yapılan işlemler asgariye indirilerek) gerçekleştirmesine imkân vermelidir.

Bazı etkinlikler işyerinin içersinde yürütülür (veriler ve bunların işlendiği gereçler işyeri dahilinde kullanılır), diğerleri ise firmanın diğer sektörleri ile ya da dışarıyla bağlantılar gerektirir. Bu çok yönlü durumun şematik olarak görüntülenmesine imkan veren bir yöntemi, karşı sayfada görmekteyiz. Şemada, günümüzde büro işlemlerini içerebilecek alt sistemleri görüyoruz. Şemada ''büro alt sistemleri'' olarak adlandırılan bölümün bileşkeleri olarak aşağıdakileri öngörebiliriz:

— *İç haberleşmeler:*
  İç yazışma ve haberle melerin yönetimi; gayrıresmi haberleşmeler;
— *Büro arşivleri:*
  Birçok kişinin kullanabileceği büro belgeleriyle ilgili arşivler: Kataloglar, adres defterleri, telefon fihristleri, personel sicilleri;
— *Büro prosedürleri:*
  Değişik etkinlikleri ve bilgi akışlarını düzenleyen (resmî veya yarı resmî) kurallar bütünü.

Resimde bağımsız bir alt-birim olan elektronik veri birimleri görülmektedir. Bu terimle sistemde elektronik olarak yerine getirilen bütün bilgi işlem faaliyetleri ve diğer faaliyetler kastedilmektedir. Bütün bu faaliyetlerin belli bir büro için geliştirilmiş bir modelin ölçütlerine uyarlanması gerekir.

Dış çevre kavramıyla anlatılmak istenen, büronun da dahil olduğu global sistemdir: Büronun, bir parçası olduğu firma ve bu firma düzeyindeki diğer örgütler, yasal normlar, dış kuruluşlarla yazışma, kamuya ait bilgi arşivleri vb.

Buraya kadar anlatılan fikirlerden yola çıkılacak ve herhangi bir şekilde birbiriyle bağıntılı olan işyerlerinden oluşan bir kompleksin yönetiminde merkezî bir bilgisayarın (küçük ya da orta boy) kullanımı üzerine düşünülecek olursa s.855'te verilmiş olan şemadaki genel mimarî yapı öngörülebilir. Bir kere daha belirtmek gerekir ki, genel olarak elektronik veri işlem süreçleri için kullanılmakta olan bir bilgisayardan söz edilmektedir.

Tüm önemli alt-sistemlerin (verilerin işlenmesi, büro, iş merkezleri, iletişim ve merkezi bilgiyazım servisleri) kendi aralarında bağlantı içinde olduklarını varsayabiliriz. Terminal aracılığıyla kullanıcı ile kurulan bağlantı noktası, işletme sırasında, aşağıda belirtilen basitlik ve esneklik taleplerini karşılamak zorundadır:

— *Yönlendirilmiş diyalog (listeler, kendini açıklayan menü ve talep simgeleri, işletimsel ilişkilerin sürekli görüntülenmesi, etkili hata göstergeleri, yardımcı fonksiyonlar);*
— *Diyalogda ve bilgiye ulaşmada büyük esneklik (kullanıcı işletimsel ortamı her an değiştirilebilme durumunda olmalıdır);*
— *Güdüm dili en aza indirgenmiştir (fonksiyon tuşlarının yaygın kullanımı seçme yapmak için imlecin kullanımı...);*
— *Sistemden bir bütün olarak veya sadece bazı temel fonksiyonlarından yararlanılabilmeli (yani sistem ''otonom'' fonksiyon gruplarına (''subsetting'') bölünebilmeli; bu bölümlerden biri yalnızca bürodaki asgari teme l ihtiyaçları karşılayacak işlemleri içermeli (''training base'');*
— *Kullanıcıyla bağlantı noktasını kişiselleştirme olanağı;*
— *Bağlantı noktasının ''sağlamlığı''.*

Bir büroda kullanılan nesnelerin tümü, şu veya bu biçimde yazılı bilgilerden oluşur. Onları ayıran, yapıları, ür tilme ve kullanılma biçimleridir.

Buna karşın büronun tüm nesnelerinin or k noktası metindir. Bu ise bütün işlerin üretilmesi için türdeş gereçlerin ve sözkonusu metinlerin saklanması ve bulunması için benzer enformatik ölçütlerin kullanılmasını mümkün kılar. Belirtilen alanların yapıları, katı oluşum kuralları olmama ı nedeniyle başka biçimler de

alabilir: Üzerinde notlar bulunan bir kağıt mektup haline gelebilir; birden çok kağıt birleştirilerek bir belge oluşturulabilir; raporlar ve metinler bir belgenin bir bölümü olabilir vb. Büro alanlarının yapı ve kullanım tarzlarının analizi, beş farklı metin türü saptanmasını sağlamıştır.

- ***Belgeler***

Bunlar, yapılandırılmamış olarak kabul edilen ve belirli miktarda ''karakteristik özellik''le donatılmış metinlerdir. Bu karakteristik özellikler ise aynılaştırma sınıflama/arşivleme ve içerdikleri bilgilere ulaşmayı kolaylaştırma gibi işlevlere yöneliktir. Her belgenin içermesi gereken en önemli ''karakteristik özellikler'': Belgenin adı, sınıflandırması, gizlilik derecesi, versiyonu, yazarı, imzası, başlığı, ek yorumlar/notlar, anahtar kelimeler, dizin, bilgiyazım için gerekli ölçütler ve sayfa düzeni.

Bu özelliklerin tümünün bulunması zorunlu değildir, bazıları da örneğin tanımlamalar (tanımlama, sistemdeki her olguyu tek anlamlı şekilde adlandırır), sistem tarafından otomatik olarak yaratılabilir.

- ***Mektuplar***

Bunlar kendine özgü özelliklere sahip metinlerdir: Gönderen (belgedeki yazarın yerine), konu (başlığın yerine), alıcı listesi (bir dağıtım listesi adı ve ilerek de elde edilebilir); bazıları ''bilgi için'' öngörülmüş veya ''bizim/sizin yazınıza dayanarak'' şeklinde bir formül içerebilir, belli bir gizlilik derecesine sahip olabilir veya ekte olabilecek dokümanların sayısı, gönderme tarihi vb. belirtilebilir. Kullanıcı, mutlaka ''gönderen'' ve ''alıcı'' alanlarını girmelidir. Sistem, tarihi ve gönderme zamanını otomatik olarak belirtir. Bu mektupların geleneksel bir büronun formel yazışmaları içinde yer alan standartlaştırılmış nesneler olduğunu belirtelim. ''Mesajlar''dan ayrıldıkları yön budur.

- ***Mesajlar***

Bunlar genellikle kısa ve formel olmayan haberleşmelere tekabül eden ''kullanım'' nesneleridir. Normal olarak, alıcının onları okumasından sonra sistem tarafından ortadan kal-



Her büroda karşılaşılabilecek karmaşık işlerin şeması. Üç alt-sistem birbirinden ayırdedilebilir. Büro alt-sistemi, elektronik veri işlem ve dış bağlantılar.

dırılırlar. Eğer alıcı isterse, onları bellekte saklayabilir ve yeniden kullanabilir.

Nitelikleri en aza indirgenmiştir: .....den ..... deye..... gibi.

- **Listeler**

Burada benzer yapıya sahip veri kaydı sıralarından oluşmuş nesneler sözkonusudur. Bunlar, dağıtım listelerinin oluşturulması, mektupların ya da değişken alanlar içeren belgelerin (form belgeleri) kişiselleştirilmesi gibi amaçlarla ve veri işleme arşiviyle veri alışverişi sırasında kullanılabilirler.

- **Formülerler**

Listelere oranla daha karmaşık olan yapılandırılmış nesnelerdir. Geleneksel büro alışkanlıklarına karşılık gelirler ve örneğin resmi büro işlemlerinde kullanılırlar. İsimlerle donatılmış alanlardan oluşurlar ve bu şekilde standartlaştırılmışlardır (belirlenmiş bir alan, verilerin yalnızca belirli bir türünü içerebilir). Biçimlerinin çeşitliliğine rağmen donatıldıkları sabit özellikler en aza indirgenmiştir. Genel olarak bir nesne oluşturulur, arşivlenir, arada sırada güncelleştirilir, diğer nesnelerle birleştirilir, gönderilir, kopya edilir, notlar eklenir vb.

Üzerinde bu tür işlemler yapabilmek için, bir nesnenin diğer nesnelerle birlikte veya tek başına ''el altında'' olabilmesi gerekir. Burada farklı tabiattaki nesnelerin birleştirilmesinin veya onları (niteliklerini değiştirmek yoluyla) dönüştürmenin zorunluluğunun altınıç izelim.

Tüm nesneler arşivlerde saklanır. Bu arşivler kişisel veya kollektif, yerel veya merkezileşmiş olabilir. Ayrıca bu arşivler, değişik tür nesneler de içerebilirler.

Her kullanıcıya iki ''mektup kutusu'' verilmiştir (biri girişler, diğeri çıkışlar için). Bunlar, gelenve giden ''posta''yı muhafaza ettiğimiz arşivlerdir. ''Posta'' terimi ile, değişik kullanıcılar arasında belirli kurallara göre gönderilmiş tüm nesneleri kastediyoruz. Kullanıcı için olası azami örgütsel esnekliği sağlayabilmek için, her arşive (ya da posta kutusuna-**mailbox**), arşive erişme yetkisine hangi kullanıcının sahip olduğunu gösteren bir ''erişim listesi'' verilmiştir.

Farklı erişim düzeyleri bulunması mümkündür: Yalnızca okuma, okuma ve yazma, okuma ve aslını değiştirmeden suret çıkarabilme olanağı. Mantıksal açıdan, sistem, ''düğüm''lerden oluşan bir ağ gibi tasavvur edilebilir.

Her düğüme bir çift posta kutusu ve erişim listeleri aracılığıyla birkaç dal verilmiştir. Sistemin kullanıcılara sunduğu fonksiyonlar, sınıflara ayrılmıştır.

1. NESNELERİN OLUŞUMU VE GÖZDEN GEÇİRİLMESİ

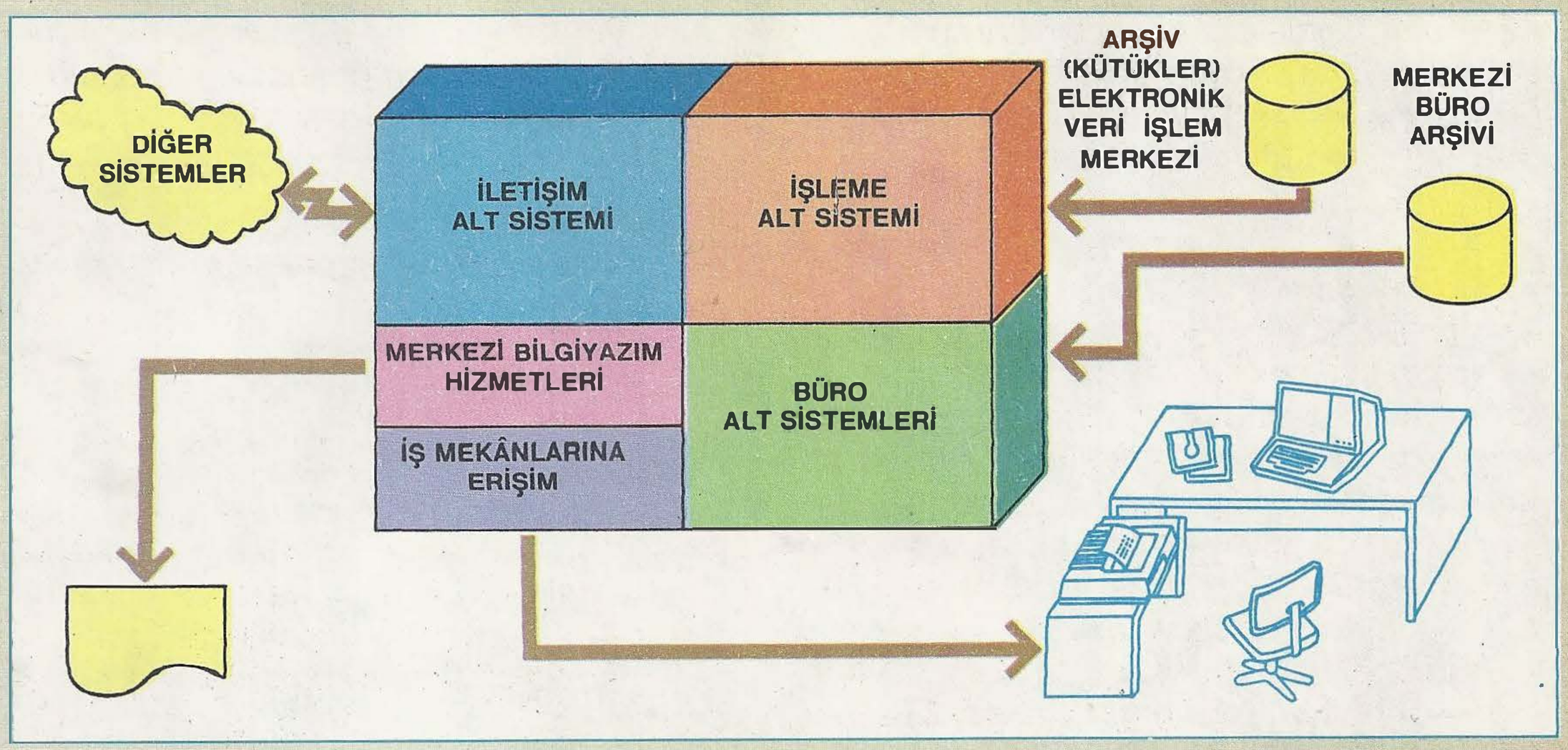
Bu fonksiyonlar, arşivin dışında bulunan bir ''yinelenen nesne'' ile çalışırlar. Bu nedenle, bir nesnenin oluşturulması ve gözden geçirilmesi arşivin durumunu bozmaz. Sistem, kullanıcı için, kendiliğinden ve görüntülenmeyen bir iş sureti oluşturur. Gözden geçirme ve düzeltme sürecinin sonunda kullanıcı, suretin arşivdeki versiyonun yerini almasını talep edebilir.

Verimli bir metin işlem sistemi (**word processing**) tarafından sağlanan tüm oluşturma/gözden geçirme fonksiyon arı mevcuttur; bunlar:

— Kes biç (''cut and paste'') fonksiyonları: Nesnelerin, diğer nesnelerden hareketle birleştirilmesi, daha sonra yeniden kullanılmak üzere ''kesitlerin'' belleğe alınması, ''form belgeleri''nin belleğe alınması.
— Metin dışı notların yerleştirilmesi;
— Belgelerin ''paragraflarına'' doğrudan erişmeyi sağlamak için kullanılacak göstergelerin otomatik olarak oluşturulması gerekir.

Sayfa düzeni verilmiş nesnelerde (listeler ve formülerler) ise:

— Sayfa düzeninin özel fonksiyonlar aracılığıyla tanımlanması (kullanıcıya formü erin ya da listenin görüntüsünü tanımlama olanağı veren yönlendirilmiş diyalog yardımıyla);
— Sayfa düzeni tarafından yönlendirilen girdilerle birlikte derleme fonksiyonları (dikey/yatay tablolar, veri türlerinin değerlendirilmesi, hesaplanmış alanların otomatik olarak uyarlanması, örneğin yüzdeler, genel toplamlar); - listeler üzerinde seçme (örneğin, başka bir listede belirli ortak özelliklere sahip değerlerden hareketle, bir alt listesinin oluşturulması);
— Birleştirme fonksiyonları (**merge**) (başka bir listedeki verilerin yerleştirilmesiyle mektup/belge oluşturulması);
— Arşivlere giren ya da arşivlerden çıkan verilerin kabulü/aktarılması gerekir.



Büro etkinliklerinin yürütülüş şeması.

2. BİLGİYAZIM

***Merge printing** olarak adlandırılan bilgiyazım, öncelikle yerel veya uzaktaki bir bilgisayar aracılığıyla bilgiyazım için formatlamanın (yazı düzeni vermenin) normal fonksiyonlarını sağlar, ayrıca birleştirmenin sonuçlarının bir kopyası arşivde korunmaksızın, değişken bölümler ve listeler içeren belgelerin bilgiyazım aşamasında doğrudan birleştirilmelerini de sağlar.*

3. ARŞİVLERDE (KÜTÜKLERDE) ARAMA

Bu fonksiyon grubu, nesnelerin bir ya da birçok arşivde aranmalarına imkan verir.

Kullanıcı nesne gruplarını, üç ayıklama düzeyine göre seçer:

1- Kesin belirlenmiş bir arşiv çerçevesinde ya da kullanıcının görebileceği tüm arşivlerde seçme,
2- Nesne türlerine göre seçme: Mektuplar, belgeler vb.
3- — Nesnenin giriş aşamasında belirtilen anahtar sözcüklerle,
— Tespit edilmiş belli değerlere (başlık, yazar vb.) dayanan seçme.

Seçmenin bu üç düzeyi aynı anda kullanılabilir.

Seçme fonksiyonu tüm seçilmiş nesnelerin listesini görüntüler. Bu liste aracılığıyla kullanıcı, ya tek bir nesneyi seçip inceleyebilir, ya da değişik nesneleri aşağıdan yukarıya ya da yukarıdan aşağıya doğru sırayla inceler (burada herhangi bir ögeden başlayabilir). Seçilen nesne tüm karakteristik özellikleriyle birlikte, alışılmış belgelerin veya mektupların sayfa düzenine benzer bir biçimde görüntülenir.

*Ayrıca bu sistemlerde önceden varolan metnih içbir şekilde değiştirme ecek bazı işlemler de mümkündür: Yatay ve dikey kaydırmalar (**scroll**), metin bölümlerinin ya da belirli bir paragrafın aranması, daha sonra kullanmak üzere metinden bazı bölümlerinin çıkartılması vb.*

Diğer taraftan, bir arşivin incelenmesi aşamasında nesneleri değiştirmek mümkün olmasa bile (bunun için iş çerçevesi içinde bir kopya yapmak gerekir), orjinal metne dahil edilmeyecek açıklamalar eklemek her zaman mümkündür. Listelenmiş nesneler ortak olarak işlenebilirler: Silme/başka bir arşive aktarma ya da kopya etme, listelenmiş tüm unsurların bilgiyazımı, listenin bilgiyazımı.

4. GİDEN POSTANIN YÖNETİMİ

*Bir mektubun "elektronik posta" aracılığıyla gönderilebilmesi için, giden mektup kutusuna (**output mailbox**) bırakılmış olması gerekir. Genellikle arşivlerde gerçekleştirilen işlemlerin tümü buradada yapılabilir; ayrıca şunlar da gerçekleştirilebilir:*

İmza — *Kutunun sahibi (amay alnızca o; erişim listesinde başka hiç kimse bu işi yapamaz), o sırada görüntülenmiş olan mektuba, bir güdüm aracılığıyla kendi imzasını atar. Bu*

imza, mektubun birlikte sunulacak bir karakteristik özelliği olur.

Gönderme Mektup bitince, basit bir güdüm aracılığıyla gönderilir. Mektup dışında, diğer nesneler de mektuba ekli belgeler olarak gönderilebilirler.

5. GELEN POSTANIN YÖNETİMİ

Gelen mektuplar, sistem tarafından gelen mektup kutusuna (**input mailbox**) bırakılırlar. Kullanıcı talep ettiğinde, bu kutunun içeriğinin d ökümünü görebilir: Mektuplar ve muhtemel ekler. Bir kutunun yalnızca onun sahibi tarafından değil, erişim listesinde bulunan her kişi tarafından incelenebileceğini kaydedelim. Ancak gönderen tarafından gizli kaydıyla belirtilmiş mektuplar, sadece yetkili alıcı tarafındang örülebilir.

Mektup kutusu ile gerçekleştirilebilecek işlemler, genellikle arşivler için de uygulanabilen işlemlerdir.

Alınan mektuplar değiştirilemezler. Kullanıcı mektubu okuduktan sonra, istediği arşivi belirterek saklayabilir. Aksi takdirde mektup arşive konmaz ve kutunun bir sonraki açılmasında yeniden görüntülenir.

6. KİŞİSEL AJANDA

Bu fonksiyon grubu, kişisel notlar için bir not defterinin (ajandanın) yönetilmesini sağlar. Kullanıcı, günü (bugün, yarın, kesin bir tarih) belirleyerek veya belirli bir haftayı seçerek kolayca danışabileceği bir ajandaya sahiptir. Sistemi, ajan danın içindebu lunan mesajları gerekli tarihlerde kullanıcıya hatırlatmak üzere programlamak mümkündür.

7. SÜREÇLERİN TANIMLANMASI

Buraya kadar anlattığımız sistem, olabilecek en büyük işletimsel esneklik ölçütüne göre oluşturulmuştur: Temel olarak kullanıcı tarafından tanımlanması gereken hiçbir işe ihtiyaç göstermez. Sistem, gerçekten çok çeşitli örgütlenme durumlarına, bunları aksatmaksızın, olabilecek en yüksek esneklik düzeyiyle uyum sağlayabilmelidir. Kullanıcı için sistemin her tekil örgütlenme boyutu, gene' olarak, belirli miktarda az ya da çok formelleştirilmiş işletim normlarına ve süreçlerine sahip olmalıdır.

Örneğin büro etkinliklerinin otomatizasyonuna yönelik gelişkin bir sistem, bu normlara ve süreçlere uymak zorundadır. Ayrıca, her kullanıcı, yürürlükte olan değişik prosedürlere göre sistemi kişiselleştirme olanağına sahip olmalıdır. Örneğin, önceden saptanmış bir örgütlenme bağlamında, sisteme aşağıdaki unsurları iletmek mümkün olabilmelidir:

— ''Bir mektup ancak gönderenin imzasını taşırsa gönderilebilir.''
— ''Belirli özelliklerle donatılmış olarak gelen bir mektup, bu işe ayrılmış bir arşivde saklanmalı ve özel bir açıklamayla birlikte önceden belirlenmiş bir alıcıya yollanmalıdır.''
— ''Belirli arşivlere ait erişim listesinde herhangi bir değişiklik, tüm ilgili kullanıcılara bir mektup aracılığıyla bildirilmelidir'' vb.

Bu ancak, kullanıcının makro güdümler (örn: GÖNDER/SIRALA...) tanımlama olanağı varsa, mümkün olur:

— Temel güdümler (sistem tarafından işlenen ve ilerki sayfalarda özetlenen tüm güdümler),
— Bu güdümlerin uygulanmasını olanaklı kılan koşullar (örneğin; ''imza var'', ''özellik x = yyyy'').

8. KÜTÜKLERİN (ARŞİVLERİN) TANIMLANMASI

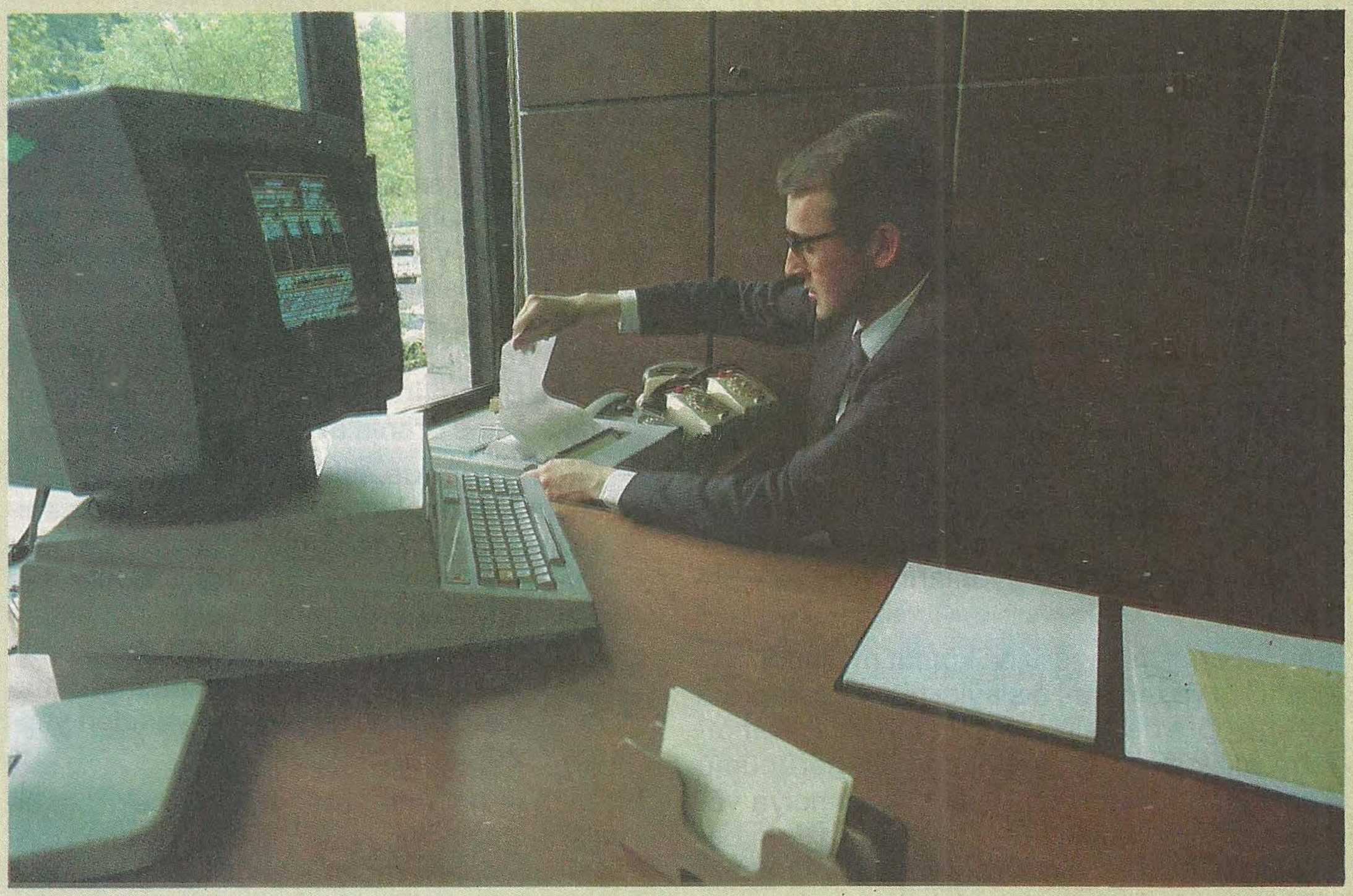
Bu fonksiyon grubu, normal kütük yönetimini (oluşturma-silme-çoğaltma) ve bunlara karşılık gelen erişim listelerinin yönetimini sağlar.

9. SİSTEMİN YÖNETİLMESİ

Fonksiyonların bu ana bölümü yalnızca, sistemi yönetmekle yükümlü olan belirli bir kullanıcıya ilişkindir:

— Tüm kullanıcıların ve karakteristiklerinin belirlenmesi ve yönetimi,
— Farklı kullanıcılara, sistem tarafından sunulan farklı fonksiyon sınıflarının öğretilmesi,
— Koruma arşivlerinin (''back-ups'') ve merkez arşivlerinin oluşturulması ve yönetimi,
— Prosedürlerin global düzeyde tanımlanması.

Her fonksiyon sınıfına, büro etkinliklerine ilişkin alt sistemin ana menüsünden hareketle seçilebilecek ve her kullanıcının ihtiyaçlarına göre kişiselleştirilebileceği bir temel fonksiyonlar alt menüsü ayrılmıştır. Sistemin bütün fonksiyonlarının kullanıcı tarafından standart bir bağlantı noktası üzerinden harekete geçirilmesi mümkün ise de kullanıcı sistemi kullanmayı; örneğin, şu alıştırma/kullanım sırasıyla adım adım öğrenebilir:



Olivetti M20 bilgisayarında yerleşim haritası tekniğinin kullanılması.

— *Temel metin işlem: Oluşturma/gözden geçirme/birleştirme, bilgiyazım, arşivlerde kütük arama (tüm fonksiyon biçimleri değil). Uygun şekilde üretilmiş ve önceden tanımlanmış kütükler üzerinde çalışır. Bu, arşivlerin oluşturma ve yeniden yapılandırma fonksiyonlarını içermediği gibi, diğer işyerleriyle olan haberleşme fonksiyonlarını da içermez.*

— *Haberleşme çekirdeği: Gelen ve giden postaların yönetimi. Bu fonksiyonlar tümüyle, bir önceki noktada kullanıldıkları gibi (giriş ve çıkış mektup kutuları kütüklerdir), kütük fikrine (ve kütüklerle ilgili temel işlemlere) dayalıdır. Ayrıca bunlara basit, özel işlemler de eklenir.*

— *Gelişmiş metin işlem: Kütüklerin tanımlanması, kütüklerde arama (tüm fonksiyon usülleri). Her kullanıcının girebileceği kaynaklar üzerinde global nitelikteki işlemlere olanak verir; kütüklerin yeniden yapılanmaları ya da tanımlanmaları; bütün kütüklerde bilgi arama vb.*

— *Kişisel fonksiyonların esası: Kişisel ajanda.*

— *Yerel prosedürler çekirdeği: Prosedürlerin tanımlanması, sık kullanılan işlemlerin ve prosedürlerin tümünü niteleyen makro-güdümlerin tanımlanması.*

— *Sistemin yönetiminin esası: Sistem yönetimi. Yeni kullanıcıların katılımını ve farklı yerel kullanıcıların uymak zorunda oldukları işletim normlarının tanımlanmasını sağlar. Bunun için kapsamlı sistem bilgisi gereklidir.*

Büro işinin otomasyonu alanındaki gezimiz böylece noktalanıyor. Sistemin işlevsel talepleri anlatıldı, geleneksel işlem tarzları (örneğin: metin işlem) ve daha inceltilmiş işlem tarzları (örneğin: elektronik posta, veri işlem aracılığıyla entegrasyon, büro nesnelerinin arşivlenmesi ve bulunması). Bütün bunlar, sistemin belirli bir büro için karakteristik olan normlara ve süreçlere uyarlanmasını sağlayan işlem tarzları da dahil olmak, birbiriyle birleştirildi.

("Un modello di automazione dell'ufficio" adlı makaleden, A.Cicu, M.Gualzetti, R.Polillo, QADERNI DI INFORMATICA, 9. Yıl, No: 2, 1982, Honeywell Information Systems Italien.)

fark vardır: Komuta tekabül eden arama tekniği değişiktir. BASIC'te bir kütüğün herhangi bir veri kaydının içeriğinin yerine başka birini koyabilen bir program, kütüğün ilk kaydından son kaydına kadar, bir okuma ve yazma döngüsünün varlığını gerektirir.

Yapılması gereken işlemler şunlardır:

— Kütüğün uzunluğunun okunması (örneğin 1 numaralı veri kaydında saklanmış)
— İlk ve son veri arasında döngüyle birlikte bir kayıtın okunması-işlem-yazmadan oluşan bir döngü.

Programın döngü, okuma ve yazma komutları REPLACE ALL güdümünde birleştirilişlerdir. Sistem kütüğün uzunluğunu dizinden okur ve döngü ile G/Ç fonksiyonlarını harekete geçirir.

Veri tabanında öngörülen komut mevcudunun çeşitliliği ve birbirini dışlayıcı niteliği, program türüne göre değişir. Gelişkin türlerde, komutların miktarı bir programlama dili görünümü kazanır ve kullanıcıdan çok programcıya yönelik bir araç haline gelir. İşte bu yüzden, bazı programlarda, yürütülecek olan komutların disket üzerinde ve komut biçiminde saklanmaları öngörülmüştür. Bu takdirde komutlar bağımsız bir kütük içersinde saklanır. Bu kütüğün adı bir güdüm işlevi görebilir. Ve böylece tek bir güdümle yürürlüğe sokulmaları sağlanmıştır. Programcı böylelikle komut ve güdümleri BASIC'in özel bir versiyonundaymışçasına kullanabilir; kullanıcıya ise daha önceden tanımlanmış olan prosedürleri yürütmekten başka bir iş kalmaz.

Tabloların Bilgiyazımı

Tabloların hazırlanışı ve çıkışı veri tabanının yönetimine ilişkin tüm programlarda öngörülmüş son fonksiyonu oluşturur.

Kullanıcı bilgiyazımını yapmak istediği alanları ve bunların hangi pozisyonda basılacağını belirleyerek bilgiyazımı yapılandırabilir. Daha gelişkin prosedürlerde genel toplam alınması ve bunun tablonun sonunda gösterilmesi gibi basit hesaplar da yapmak mümkündür. Bazı durumlarda, bir alanın içeriğini durdurma kodu olarak kullanarak (bilgiyazım ele alınan bölümün içeriği değiştiği an harekete geçirilir) ara sonuçların da bilgiyazımı mümkün olmaktadır.

Bu işlemlerin koşullandırma biçimi bazı veri tabanlarında mevcut olmakla birlikte, ilişki oluşturmaya yönelik RPG (Report Program Generator) diline özgüdür.

Metinlerin İşlenmesi

Büro işinin otomatizasyonu metin, mektup, belge vb. metinlerin bilgisayarla yönetimini kapsamaktadır. Bu amaçla bir metnin, herhangi bir veri bütünüymüş gibi ele alınarak hazırlanmasını sağlayan değişik programlar (metin işlemciler) yazılmıştır. Böylece metinde ileride karşılaşılabilecek düzeltiler çok kolay yapılabilir. Ayrıca bir metnin istendiği kadar bilgiyazım kopyasının elde edilebilmesi de mümkündür. Bir metin işlemci programının bir yazı makinesine oranla avantajları çok fazladır:

— Metnin bellekte saklanması; her şeyi yeni baştan yazmaksızın düzeltme ya da değişiklik yapma olanağını sağlar.
— Diskette saklama, kağıt üzerinde olandan çok daha yoğundur.
— Birçok baskı elde etme olanağı, fotokopiyi gereksiz kılar ve orjinalin mükemmel olmaması durumunda karşılaşılan yeniden yapma sorunlarını bertaraf eder.

Bu alanda da yine yararlanılabilecek fonksiyonların çeşitliliği, kullanılan programa göre değişir. Modern metin işlemcilerde çok özel ve yararlı fonksiyonlar öngörülmüştür.

— Kullanıcı tarafından tanımlanan bir sözlüğe göre istenen dilde gerçekleştirilen yazım hataları araştırması.
— Önceden tanımlanmış olan bir metnin arasına, bir ya da birkaç sözcük ekleme ve bundan sonra kendiliğinden gerçekleşen yeni bir sayfa düzenlemesi olanağı.
— Satır sonlarında hece bölünmelerinin sınanması.
— Tabloların oluşturulması.
— Belgelerin içeriğine yönelik basit arama fonksiyonları.

Büyük sistemlerde belgelerin doğrudan girilmesini de mümkün kılan özel gereçler bulunmaktadır. Karmaşık gereç sistemleri, sayfada yazılı metni okuyarak, verileri tek tek değişkenlere aktarırlar. Bunlar bilgisayar tarafından işlenir ve saklanır. Yine aynı yolla elle yazılmış veriler, resimler veya şemalar da işlenebilirler veya değiştirilebilirler.

SÖZLÜK

MULTIPLAN SÖZLÜĞÜ (OLivetti M20 versiyonu)

Ad (bir veya bir grup hücrenin). Bir veya birden çok hücreye NAME güdümüyle yöneltilen belleksel bağıntı. Ad, formüllerde hücreye veya hücre grubuna bağıntı olarak kullanılır.

Aktif. Derhal kullanılabilecek ve ulaşılabilecek olanı ifade eder. Örneğin aktif pencere, aktif hücre, bir güdümün aktif alanı.

Aktif hücre. Hücre göstergesi tarafından işaretlenen hücre. Aktif hücrenin içeriği durum satırında görülebilir ve EDIT güdümüyle değiştirilebilir.

Aktif pencere. Aktif hücreyi içeren pencere. Ekranda, yüksek parlaklıkta bir pencere numarasıyla belirtilir.

Aktif tablo. O sırada işlenmekte olan değerleri içeren hücrelerin tümü. Azami boyutları 63 (satır) çarpı 255 (sütun) hücreden oluşan bir matrise karşılık gelir. Tablonun her hücresi, sayısal değerler, metinler veya formüller içerebilir.

Alan. Bir güdümün, yürütülmesi için usüllere ilişkin esasların Multiplan'a verilebilmesi için, içine bir cevap girilen kısmı.
Multiplan bir alanı görünür kıldığında, otomatik olarak bir cevap sunar. Sunulan cevap, kabul edilebilir veya bu alan için tanımlanmak istenen geçerli bir cevapla değiştirilir.

Bağımlı tablo. Bir diğer tablodan kaynaklanan değerleri kullanan tablo. Bağımsız tabloya, bir diğer saklanmış tabloyla hesaplanmış olan ve EXTERNAL COPY güdümüyle bu tabloyla bağlanmış olan değerler tabi kılınmıştır. Bakınız **Korelasyon.**

Bilgi (Information). Multiplan tarafından, bir işletim veya kullanım hatasının nedenlerini açıklamak veya sistemin beklediği girişin türünü göstermek için, bilgilere ayrılmış satıra verilen mesaj.

Bilgi satırı. Görüntülenen tablonun en son satırının hemen üstündeki satır. Multiplan bilgilerinin görüntülendiği ekran alanı.

Bir sonraki korunmamış hücre tuşu. Hücre göstergesini bir sonraki boş veya bloke edilmiş olmayan hücreye kaydırır. Değişikliklerin tablodaki etiketlerini sınamak amacıyla değiştirilebilecek sayılar (formül veya metin değil) içeren hücreler oluşturmak için kullanılır (örn. WHAT IF yöntemi).

Bloke etme. Formül veya metin içeren hücrelerin yanlışlıkla yapılacak değişikliklerden korunması.

Durum satırı. Ekranın en alttaki satırı. Multiplan bu satırda, sistemin durumu üzerine bilgileri verir. Örneğin, aktif hücrenin pozisyonu ve içeriği.
Ekran. Sistemin, sayıların ve/veya metinlerin görüntülenmesini sağlayan elemanı. Ekran yatay olarak 25 hücreye bölünmüştür. Multiplan, son 4 satırı bilgiler ve yönetim komutları için kullanır. En alttaki satır durum satırıdır, üstündeki bilgi satırıdır, bundan önceki iki satır ise güdüm satırlarıdır. Tüm diğer satırlar ise tabloya ayrılmıştır ve tablonun bir bölümünü bir veya birden çok pencere aracılığıyla görüntülemek için kullanılır.

Fihrist (directory). Diskette saklanan kütüklerin ad listesi, yani repertuar, pratikte kütüklerin listesine karşılık gelir.

Fonksiyon. Multiplan'ın bir veya birden çok değerde yerine getirebileceği, önceden belirlenmiş matematik veya statik hesap; örneğin AVERAGE, SUM.

Fonksiyon tuşları. Multiplan tarafından bir faaliyeti derhal yerine getiren tuşlar. Fonksiyon tuşları, CANCEL, NEXT WINDOW ve CR (RETURN)'yi kapsar.

Formül. Bir değerin hesaplanma şeklinin betimlenmesi. Bir hücrenin değerinin her değişmesinde Multiplan, tablonun tüm formüllerinin sonuçlarını yeniden hesaplar (çünkü, otomatik yeniden hesaplama engellenmiştir).

Gönderme. Tablonun içinde bir veya birden çok hücrenin adlandırılması. En basit bağıntı tek bir satır üzerindeki Z9S2'dir.

Bir gönderme,
— Göndermeyi içeren hücrenin pozisyonuyla ilişkili olabilir, Z (—1) S gibi.
— Önceki maddede belirtildiği gibi tek bir hücrede veya bir hücre grubunda gerçekleşebilir: Z6, 6. satırda bulunan tüm hücrelere ilişkindir ve göndermelerin kesişmelerinden, range'lerden veya göndermelerin bağlantılarından oluşabilir.
— Bir veya birden çok hücreyi adlandıran, önceden tanımlanmış bir ad olabilir.

Ranj ve **Ad**'a bakınız.

Göreli Gönderme. Bir hücreye kendi pozisyonu ile ilişkili olarak gönderme, örneğin Z (—1) S, "aynı sütunda bir sonraki satırda bulunan hücre" anlamına gelir. Mutlak göndermenin tersi, örn. Z17S12.

Güdüm. Multiplan'a bir komut verir. Bir güdüm, kendisinin yerine getirilişine ilişkin usülleri içeren bir veya birden çok alan öngörür.

Güdüm satırı. Tablonun görünür kılınan kısmının hemen altındaki ekran satırları. Başında COMMAND kelimesi bulunur, ardından güdümlerin ana menüsü gelir. Bu, güdümlerin girildiği yerdir.

Hücre. İçine sayısal değerler, formüller veya metinler yerleştirilebilen tablo ögeleri. Bir hücre, koordinatlarıyla tanımlanır ve bir adla bağıntı kurulabilir. Bir hücrenin içeriği değerini belirler, hücrenin yazı düzeni, değerin ne şekilde görüntüleneceğini belirler.

Hücre göstergesi. Ekran üzerinde aktif hücreyi işaret eden yüksek parlaklık alanı. Hücre göstergesi, yönetici tuşları ile veya GOTO güdümüyle bir hücreden diğerine kaydırılır.

Hücre Grubu. Tablonun, bir ad alabilen bir veya birden çok hücresi. Örneğin, ''satış''.

İçerik (bir hücrenin). Bir hücreye girilen. Hiçbir şey girilmezse hücre boştur ve boşluk yerleri içerir. Aksi takdirde hücre, metinler, sayısal değerler veya bir formül içerir. Eğer bir hücre bir formül içeriyorsa, formülün sonucu olan, hücredeki değer görüntülenir.

İmleç. Bakınız **yazımlama imleci.**

İmleç yönlendirme tuşları. Hücre imlecini hareket ettiren tuşlar. , ve tuşları hücre göstergesini bir hane kaydırır. HOME tuşu, göstergeyi aktif pencerenin sol üst köşesindeki hücreye kaydırır.

Karakter. Ekranda görüntülenen simge. Karakterler, harf, rakam, noktalama işareti veya $, + , % gibi özel işaretler olabilir.

Kaydırma (scroll). Ekran içeriğinin satırlar veya sütunlar bağlamında kaydırılması. Bu işlem imleç kaydırma tuşlarıyla yapılır. Örneğin, tuşuna, hücre göstergesi ekranın sağ kenarına gelene kadar ve uzun süre basıldığında, Multiplan görüntüyü sütun sütun sola kaydırır. Kaydırmanın bu türü (scroll) ''yatay kaydırma'' olarak adlandırılır. Ekran içeriği satır satır kaydırılırsa, buna ''dikey kaydırma'' adı verilir.

Korelasyon. Multiplan'da korelasyon kavramı, aktif olmayan bir tablonun, aktif bir tablonun işlemlerinde kullanılması ilişkisini belirtir. Aktif olmayan tablo, yardımcı tablo olarak adlandırılır. Kopya edilecek verilere, NAME güdümüyle bir ad verilmeli veya bu veriler bir mutlak gönderme ile belirlenmiş olmalıdır. Daha sonra, yardımcı tablonun verileri aktif tablonun formüllerinde kullanılabilir. Korelasyon ayrıca, iki pencereyi aynı anda kaydırarak (scroll) birleştirmeye yarar.

Kütük. Diskette saklanmakta olan verilerin homojen akışı. Bir tablo tamamlandığında, bir kütüğe yazılır. Tüm tablolar, oluşturulmuş tablolar değildir, fakat böyleleri belleğe yüklenebilir ve başka tablolarla bağlanabilir.

Kütüğün adı. Bir tablonun saklanması, yüklenmesi veya bir diğeriyle bağlanmış olması durumunda, onunla bağıntılı olarak kullanılan ad.

Menü. Seçeneklerin listesi. Bir menünün seçimi şu şekilde olur: Liste üzerinde SPACE ve S2 ile hareket edilir (yüksek parlaklık menüde o andaki tercihi göstermektedir) ve istenen öge CR tuşuyla seçilir veya istenen ögenin baş harfi girilir.

Mutlak gönderme. Açık satır ve sütun numaralarının kullanılmasıyla bir hücreye yönelik bağıntı; örneğin Z17S12. Göreli göndermenin tersi.

Önerilmiş cevap. Multiplan tarafından otomatik olarak sunulan cevap. Genellikle, kullanıcı tarafından önceden verilmiş cevaplara veya Multiplan'ın haldeki durumuna dayalıdır.

Pencere. Multiplan tarafından aktif tablonun bir bölümünün görüntülendiği ekran bölümü. Aynı anda 8 pencere açılabilir: Bunların açılması ve kapanması WINDOW güdümüyle gerçekleştirilir. Her pencerenin numarası (1'den 8'e kadar) sol üst köşesinde bulunur. Aktif pencerenin numarası daha parlaktır; hücre göstergesi ve aktif hücre bu pencerede bulunmaktadır.

Ranj. İki gönderme arasındaki en küçük hücre dikdörtgeni. İki nokta (:) simgesi ile tanımlanır. 23:28 bölgesi 3 ila 8 arasındaki tüm hücreleri tanımlar (3, 4, 5, 6, 7, 8). Bakınız **Gönderme.**

Saklama. Bir kütükteki aktif tablonun diskette korunmasını sağlayan işlem.

Satır. Tabloda yatay hücre sırası. 1'den 255'e kadar numaralanmış 255 satır olabilir.

Sıralama. Bir hücrenin içeriklerinin hücre içindeki pozisyonları. Değerler sağa dayalı, sola dayalı veya ortada sıralanmış olabilir.

Silme tuşu. Yerine getirilmekte olan güdümü iptal eden ve güdüm ana menüsünün görüntülenmesini sağlayan tuş.

Sütun. Tablo üzerinde dikey hücre sırası. 1'den 63'e kadar numaralanmış 63 satır vardır.

Yardımcı tablo. Diğer bir tablodan (bağımlı tablo) kullanılan değerleri içeren tablo. Bağımlı tablonun kopyasıyla ilgili değerler, EXTERNAL NAME güdümüyle belirlenen bir adla adlandırılmalıdır. **Korelasyon**'a bakınız.

Yazı düzeni (format). Bir hücrenin içeriğinin görüntülenme şekli. Yazı düzeni, sayıdaki virgülün yerini ve görüntülenecek değerin düzenlenişini belirler. Bir yazı düzeni, FORMAT CELL güdümüyle bir ya da birkaç hücre için belirlenebilir; özel bir yazı düzeni bulunmayan hücreler, FORMAT DEFAULT güdümüyle, önceden belirlenmiş default (aksine bir durum yoksa) yazı düzeniyle görüntülenir.

Yazımlama (Edit). Bir güdüm alanında bir cevabın değiştirilmesi. Yazımlama tuşları, imleci cevabın üzerine kaydırmaya yarar, karakter tuşları ise karakterleri değiştirmek veya karakter eklemek için kullanılır.

Yazımlama imleci. Bir güdümün, güdüm satırında yüksek parlaklıktaki kısmı. Boyutları bir karakter ile tüm bir alan arasında olabilir. Yazımlama imleci, yazımlama tuşları ile kaydırılır. Güdümün değiştirilebileceği yerleri işaretler.

Yazımlama tuşları. Yazımlama imlecini güdüm satırında kaydırabilen tuşlar. Örneğin: WORD RIGHT, WORD LEFT, CHARACTER RIGHT, CHARACTER LEFT.

Yineleme. Bir işlemin, istenilen sonuca yakın bir sonuç elde edilene kadar yinelenmesi. Yinelemede, önceki işlemlerin sonuçları, bilinmeyen bir büyüklüğün yerine kullanılır.

Yükleme. Bununla, saklanmakta olan bir tabloyu yeniden harekete geçirmek mümkündür. Yüklenecek tablo, TRANSFER SAVE güdümüyle güvence altına alınmış olmalıdır. TRANSFER LOAD güdümü, tablonun disketten sistemin belleğine yüklenmesini sağlar.

Yüksek parlaklık. Ekranın, özellikle güçlü bir ışıkla öne çıkarılmış bölümü. Yüksek parlaklık, çıkış-(yazımlama) imleci, aktif hücre, aktif pencerenin numarası ve güdüm ana menüsünde seçilmiş olan güdümde kullanılır.

MULTIPLAN GÜDÜMLERİNİN YAPISI

Güdümler ana menüsünde belirtilen güdümler, baş harfleri (örneğin ALPHA için A) girilerek veya imleci S1 ve S2 tuşlarıyla istenen güdümün üzerine kaydırıp ve aynı anda güdümün uygulanması için RETURN tuşuna basarak seçilir.
Sayılar aktif hücreye, VALUE güdümünün seçimini gerektirmeksizin girilebilir. Güdümlerin büyük çoğunluğu alt güdümlere sahiptir. Bunların menüsü ana güdümün seçilmesinden sonra görüntülenir.

CTRL + C — Bir güdümü iptal eder ve ana menüye geri döner.

S1 — İmleci, menünün bir sonraki güdümüne/alt güdümüne doğru iter. İmleci, alt güdümün bir sonraki alanına kaydırır.

S2 — İmleci, menünün bir önceki güdümüne/alt güdümüne kaydırır.

SHIFT + ↑ ↓ ← → — İmleci okların yönünde hareket ettirir ve girişi, ALPHA/VALUE durumunu koruyarak, durdurur.

CR — Güdümü yürürlüğe sokar, metni ya da değeri kabul eder.

@ — Hücrelere yönelik göndermeleri göreliden mutlağa dönüştürür.

A ALPHA — Kullanılmayan bir metnin ya da değerin aktif hücreye girilmesine izin verir.

CR o SHIFT + ↑ ↓ ← → — İmlecin hareketini sağlayan tuşların kullanılması girişi gerçekleştirir ve göstergeyi verilen yönde hareket ettirir.

B BLANK — Bir ya da birden çok hücreye yönelik göndermeleri belirler.

CR — Belirlenmiş hücrelerin içeriği silinir.

C COPY

R Right — Aktif hücrenin içeriğinin, kopya edilmek istendiği hücrenin sağında yer alan hücrelerin sayısının tanımlanmasını sağlar.

CR

D Down — Aktif hücrenin içeriğinin, kopya edilmek istendiği hücrenin altında yer alan hücrelerin sayısının tanımlanmasını sağlar.

CR

F From — Alıcı hücreler olarak adlandırılan bir ya da birden çok hücre ile kopya edilecek hücrelerin arasındaki ilginin tanımlanmasını sağlar.

CR — COPY FROM güdümü COPY RIGHT'ı olduğu kadar COPY DOWN'ı da içerir. Ayrıca bitişik olmayan hücrelerin kopya edilmesine de olanak tanır.

D DELETE

R Row — Bir ya da birkaç satırda yer alan hücrelerin bütününün veya bir bölümünün göndermelerinin tanımlanmasını sağlar.

CR — Belirlenmiş bölümler tablodan çıkarılır. Güdümün yürütülmesi tablonun yukarıya doğru sıkıştırılmasına yol açar.

C Column — Bir ya da daha fazla sütunda yer alan hücrelerin bütününe veya bir bölümüne yapılacak göndermelerin tanımlanmasını sağlar.

CR — Belirlenmiş bölüm tablodan çıkartılır. Güdümün yürütülmesi tablonun sola doğru sıkışmasına yol açar.

E EDIT — Metinlerin ve formüllerin EDIT tuşları aracılığıyla değiştirilmesini sağlar.

CR o SHIFT + ↑ ↓ ← →

F FORMAT

C Cells — Görüntülenmeyen yazı düzenini değiştirmek istediğimiz bir veya birçok hücrenin belirlenmesini sağlar.

CR

D Default

C Cells — Hücrelerin görüntülenme yazı düzeninin ve sıralamada normal durumda (default) kullanılan ölçütlerin tanımlanmasını sağlar.

CR

W Width — Tablodaki tüm sütunların normal durumlarda (default) için geçerli olacak genişliğini, karakter sayısı cinsinden tanımlar.

CR

	O Options	Hücrelere özgü yazı düzeni seçeneklerini tanımlayıp doğruluklarını denetler.
	CR	Güdüm alanlarının bir parçası olarak verilen cevaplar yürürlükteki seçeneklere tekabül ederler.
	W Width	Belirlenmiş sütunlar için normal olanlardan (= Default) farklı bir genişlik (Bkz. Format Default Width güdümleri bakınız) tanımlar.
	CR	
G GOTO	**N Name**	Hücre göstergesini belirlenmiş ada karşılık gelen hücreye yerleştirir.
	CR	Adın bir hücre bölgesini adlandırdığı durumda, hücre göstergesi belirlenmiş bölgenin ilk hücresinin üzerine (yukarıda soldaki hücre) yerleşir.
	R Row-Col	Adresi (satır/sütun) verir.
	CR	Hücre göstergesini, gönderme değeri kesin olarak belirlenmiş olan hücrenin üzerine yerleştirir.
	W Window	Hücre göstergesini, verilen pencere içinde gönderme değeri önceden kesin olarak saptanmış olan hücrenin üzerine yerleştirir.
	CR	Güdüm, belirlenmiş hücrenin pencerenin üstünde solda ilk pozisyonuna erişene dek, pencerenin kaydırılmasını sağlar.
H HELP	**R Resume**	Aktif tabloyu HELP güdümünün çağırıldığı noktadan itibaren yeniden harekete geçirir.
	S Start	HELP kütüğünün ilk sayfasını gösterir.
	N Next	HELP kütüğünün bir sonraki sayfasını gösterir.
	P Previous	HELP kütüğünün bir önceki sayfasını gösterir.
	A Applications	Multiplan kullanımı sırasında sık sık ortaya çıkabilecek sorunların ve bunları çözümlemeye yönelik güdümlerin bir listesini sunar.
	C Commands	Birincisinden (ALPHA'dan) başlayarak, Multiplan'ın her bir güdümünün açıklamasını sunar.
	E Editing	Multiplan biçimleme (Edit) fonksiyonunun açıklamasını sunar.
	F Formulas	İzlenmesi gereken kurallarla ilgili bilgileri ve formüllerin tanımlanmasında kullanılan fonksiyonların bir listesini verir.
	K Keyboard	Multiplan'da kullanılabilecek tuşların ve anlamlarının listesini verir.
I INSERT	**R Row**	İçeriksiz bir ya da birkaç satırın bütününü ya da bir bölümünü, aktif satırdan önceki pozisyona yerleştirir.
	CR	
	C Column	İçeriksiz bir ya da birçok sütunun bütününü ya da bir bölümünü, belirlenmiş sütundan önceki pozisyona yerleştirir.
	CR	
L LOCK	**C Cells**	Bir ya da birçok hücreyi, değerlerini olası bir değişimine karşı korumak amacıyla bloke eder/açar.
	CR	
	F Formulas	Metin ya da formül içeren hücrelerin değerlerinin istenmeyen bir şekilde değiştiğini haber vermek amacıyla, bu hücreleri bloke eder/açar.
	Y	Talebi onaylar.
M MOVE	**R Row**	Tablo içinde yer değiştirecek satırların gönderme değerlerini ve aynı zamanda üzerinde yer değiştirebilecekleri pozisyonların gönderme değerlerini tanımlar.
	CR	
	C Column	Tablo içinde yer değiştirecek sütunların gönderme değerlerini ve aynı zamanda yer değiştirmeleri gereken pozisyonların gönderme değerlerini tanımlar.
	CR	

N NAME

Güdüm alanlarında satırında belirlenmiş olan bir ya da birçok hücreye bir ad verir.

CR
SHIFT + ↑ ↓ ← →

O OPTIONS

Tabloların hesaplanması için gereken seçenekleri devreye sokar. Örneğin hesaplamalarda kullanılacak komşu hücrelerin ya da açıklayıcı metin içeren hücrelerin tanımlanması gibi.

CR

P PRINT

P Printer

Aktif tablonun Print Margins ve Print Options güdümleri tarafından tanımlanan yazı düzenine göre bilgiyazımını yapar.

CR

F File

Aktif tabloyu disket üzerinde, bilgiyazımı yapılabilecek bir yazı düzeniyle saklar.

CR

M Margins

Bilgiyazım sayfalarının sayfa düzenini sol marj, satır uzunluğu ve sayfa başına düşen satır sayısı olarak tanımlar.

CR

O Options

Bilgiyazım seçeneklerini tanımlar:
1 / Tablonun bir bölümünün bilgiyazımı
2 / Değerler yerine formüllerin bilgiyazımı
3 / Satır veya sütun numaralarının bilgiyazım sırasında çıkartılıp atılması
4 / Bazı gereç parametrelerinin (bilgiyazar) tanımı

CR

Q QUIT

İncelenmekte olan Multiplan programından çıkılmasını sağlar. Güdümün yerine getirilebilmesi için onay istenir. Aktif tablo diskette saklanmaz. Güdümün yürütülmesine yarar. Disket üzerinde sayfa saklı tutulmamıştır.

Y

S SORT

Değerlerinin sıralanması gereken sütunun bütününü veya bir bölümünü tanımlar. Sıralama, sayısal değerler için olduğu gibi metinler veya mantıksal değerler için de giderek artan ya da azalan bir düzen içinde yapılır.

CR

T TRANSFER

L Load

İşlenmek için belleğe alınacak olan bir kütüğün adını tanımlar.

SHIFT + ↑ ↓ ← →

Yön tuşlarını kullanarak ve ad belirtmeksizin (boş alan), disket üzerindeki kütükler ekrana çağrılır. Bu şekilde belleğe yüklenecek olan kütük, listede işaretlenerek seçilmiş olur.

CR

S Save

Aktif tabloyu disket üzerinde saklı tutar ve ona belirlenmiş adı verir.

CR

Daha önceden, disket üzerinde aynı adı taşıyan bir tablo varsa, güdümün yürütülmesi için onay istenir.

Y

C Clear

Aktif tabloyu bellekten siler. Güdümün yürütülmesi için onay istenir.

Y

D **Delete**
SHIFT + ↑ ↓ ← →
CR

Belirlenen tabloyu disketten siler. Disket üzerindeki tüm kütüklerin listesini görüntülemek için yön tuşları kullanılabilir.

O **Options**
CR

Default-disket birimini ve sayfaların belleğe alınma/ve bellekten çağrılma yazı düzenini tanımlar.

R **Rename**
CR

Aktif tabloyu, değişik bir ad vererek disket üzerinde saklar. Güdüm olası dış bağıntıları yeniden tanımlar.

V **VALUE**
CR

Bir formülü girer.
Formül kabul edilir.

W **WINDOW**

S **Split**

H **Horizontal**
CR

Tablonun alanı içinde bir pencereyi tanımlar. Aktif tablo yatay olarak bölünmüştür.

V **Vertical**
CR

Tablonun alanı içinde bir pencere tanımlar: Aktif tablo dikey olarak bölünmüştür.

T **Titles**
CR

Görüntülenmiş satır ve sütunların başlıklarını içeren bir pencere tanımlar.
Aktif pencere, yalnızca satırların başlıklarına, yalnızca sütunların başlıklarına ya da her ikisinin başlıklarına birden ait olmasına göre 2 veya 4 pencereye ayrılmıştır.

B **Border**
CR

Belirlenmiş pencerenin kadrajını yeniden düzenler veya yok eder.

C **Close**
CR

Belirlenmiş pencereyi ekrandan siler.

L **Link**
CR

Belirlenmiş pencereler arasında bağlantıyı tanımlar/ortadan kaldırır.
Gönderme, birbirine bağlanmış pencerelerin aynı anda kaydırılmaları sonucunu doğurur.

X **EXTERNAL**

C **Copy**
CR

Tablonun numarasını, alıcı hücrelerin adresini (sıra-sütun) ve biçimini verir.
Değerler dış tablodan kopya edilir.

L **List**

Aktif tabloya dayanarak oluşturulan tablonun adı yeniden oluşturulur.

U **Use**
CR

Kütüğün adını verir.
Bir adı daha önce kullanılan yerine koyar.

Hones