İLERİ BASIC

Yazan
GÜZİN SAĞKAN
TUNÇ GERÇEK

II. BASIM
İSTANBUL – 1993

Bu eser, Milli Eğitim Bakanlığı'nca Bağlı Okul ve kurum Öğrencilerine Yardımcı Ders Kitabı olarak tavsiye edilmiştir.

Tebliğler Dergisi SAYI : 2344
TARİH : 30 Eylül 1991

TÜRKMEN KİTABEVİ

Dizgi Baskı: **Yaylım Matbaası**
Sayfa Düzeni: **Banu Yüce**
Kapak: **Süleyman Culum**
Cilt: **Tan Mücellithanesi**
Hazırlayan: **Güzin Sağkan**
Tunç Gerçek
Yayına Hazırlayan: **Mustafa Türkmenoğlu**
Kitabı Yayınlayan: **Türkmen Kitabevi**
Sahaflar Çarşısı No: 19 34450–Beyazıt/İstanbul
Tel: 512 27 17
Yayın No: **11**
Bilgisayar Dizisi: **8**

İLERİ BASIC

İÇİNDEKİLER

II. BASKIYA GİRERKEN

Bilgisayar Programlama Dili öğretimini amaçlayan bir kitabın kısa sayılabilecek sürede tükenmesi ve II. baskıya girmesi, kitabımızın gördüğü ilginin göstergesi olarak bizi onurlandırmış ve benzeri çalışmalar için yüreklendirmiştir.

Asıl önemlisi, bu sonuç, en karmaşık görünen konuların dahi eğitimci gözü ile ve eğitim metodlarının kullanımı ilke edinilerek, öğrenciye aktarılması çabasının başarısını ortaya koyması açısından sevindiricidir.

Teşekkür ve Saygılarımızla...

İstanbul, 1993

ÖNSÖZ

Bilindiği gibi, öğrencilerin başarılı olmalarını sağlıyan en önemli etkenlerden biri, hangi ders olursa olsun, öğrencinin konuya belirli bir mantıkla yaklaşabilmesidir. Bilgisayar programlama öğretisinin, kazandırdığı bir çok yeteneğin yanı sıra, belirli bir sistem dahilinde verildiğinde öğrencinin mantık gelişiminde de önemli katkılar sağladığı kesindir.

I. KİTAP; öğrenciye Basic programlamanın temel bilgilerinin verilmesini ve bilgisayar programlama mantığının oluşmasını ön plana almış, öğrencilerin basit programlar yapabilmelerini ve bilgileri nasıl kullanabileceklerini öğretmeyi amaçlamıştı. II. KiTAP ise, bilgileri saklama, kullanma ve daha profesyonelce program yazabilmek için artık, öğrencide belirli bir seviyede oluşmuş bulunan programlama mantığını temel alarak, çok karmaşık görünen konuları bir sistem dahilinde, basit adımlarla vererek, öğrencilerin kolay kavrayabilmelerini sağlayacak biçimde kaleme alınmıştır.

Başarılar dileğiyle...

İstanbul, 1991

GENEL UYGULAMALAR :

İleri BASIC konularını tam olarak anlayabilmek için, daha basit gibi görünen, fakat çok önemli ayrıntıları içeren temel BASIC konularının eksiksiz bir biçimde kavranmış olması gerekir. Özellikle aynı türden birçok bilgiyi tek isim altında saklamak için kullanılan dizilerden en iyi şekilde yararlanılabilmesi şarttır. Örneğin diziler arasında bilgi aktarma, bir dizi içindeki bilgilerin yerlerini belirli koşullara göre değiştirme veya dizileri çeşitli şekillerde listeleme gibi işlemlerin programcı tarafından kolaylıkla yapılabilmesi gereklidir.

Dizilerle ilgili örnek programlar:

1) 12 elemanlı sayısal bir dizinin 5'ten büyük elemanlarını ikinci bir diziye aktararak iki diziyi de listeleyen bir program:

```
10 DIM A(12),B(12)
20 FOR I=1 TO 12
30 READ A(I)
40 IF A(I)>5 THEN S=S+1 : B(S)=A(I)
50 PRINT A(I)
60 NEXT I
70  DATA 6,3,4,9,8,6,4,1,2,0,8,5
80 PRINT "---------"
90 FOR K=1 TO S
```

```
100 PRINT B(K)
110 NEXT K
```

Bu programda, A dizisinde koşula uyan sayılar B dizisine aktarılmış ve iki dizi ayrı ayrı (alt alta) listelenmiştir. Burada, iki döngü arasında PRINT"---------" yazılarak, iki liste arasında bir çizgi çıkması sağlanmıştır.

Bir dizinin herhangi bir elemanına ulaşabilmek için, o elemanın dizi içindeki sıra numarasını vermek gerekir. Diziler genellikle FOR ... NEXT döngüleri ile kullanıldıkları için, indis işlevi gören döngü değişkenlerinin programların nerelerinde hangi değerleri aldıklarını programcı çok iyi kontrol edebilmeli ve yönetebilmelidir. Örneğin bu programda, I, S ve K değişkenlerinin değerleri, programın anlaşılabilmesi açısından dikkatle izlenmelidir. Hiçbir zaman unutulmamalıdır ki, değişken adları değil, bu değişkenlerin program içinde aldığı değerler önemlidir.

2) 11 elemanlı bir dizinin ilk 8 elemanını DATA satırından, son 3 elemanını ise klavyeden okuyan ve diziyi ekrana listeleyen bir program:

```
10 DIM A$(11)
20 FOR T=1 TO 8
30 READ A$(T)
40 NEXT T
50 DATA Hüseyin,Kemal,Cemil,Turan,Erkan,Yılmaz,Remzi,
Ferit
60 FOR D=9 TO 11
70 INPUT A$(D)
80 NEXT D
90 PRINT "DİZİ ELEMANLARI:"
110 PRINT "------------------"
120 FOR C=1 TO 11
130 PRINT A$(C)
140 NEXT C
```

Bu programda da döngü değişkenlerinin başlangıç ve bitiş değerlerine dikkat edilmelidir.

3) 6 elemanlı bir dizinin elemanlarını tersten ikinci bir diziye aktararak iki diziyi de listeleyen bir program:

```
10 DIM A(6),B(6)
20 FOR Y=1 TO 6
30 READ A(Y)
40 B(7-Y)=A(Y)
50 NEXT Y
60 DATA 6,37,2,4,54,3
70 FOR R=1 TO 6
80 PRINT A(R),B(R)
90 NEXT R
```

Bu programda, diziden diziye bilgi aktarma ve dizileri listeleme için iki ayrı döngü kullanılmıştır, çünkü bu işlemleri tek bir döngüde yapmak mümkün değildir. (İlk döngüde döngü değişkeninin değeri 1 olduğunda, A dizisinin birinci elemanı okunacak ve B dizisinin altıncı elemanı olarak aktarılacaktır. Bu anda bir PRINT komutuyla dizilerin birinci elemanları ekrana yazılmak istenirse, B dizisinin birinci eleman değeri henüz belirlenmemiş olduğundan, ekrana B için 0 sayısı çıkacaktır. Ancak birinci döngü tamamlandıktan ve A dizisinin bütün elemanları ters olarak B dizisine aktarıldıktan sonra iki dizi de listelenebilecektir.) Ayrıca Y'nin değeri 1'den 6'ya giderken, 7-Y'nin değerinin 6'dan 1'e gittiği ve bu sayede elemanların ters olarak dizildiği de diğer önemli bir noktadır.

4) 10 elemanlı sayısal bir dizinin ilk 5 elemanını bir diziye, ikinci 5 elemanını başka bir diziye aktararak dizileri listeleyen bir program:

```
10 DIM A(10),B(5),C(5)
20 FOR G=1 TO 10
30 READ A(G)
40 PRINT A(G)
50 NEXT G
60 DATA 4,62,6,4,7,8,5,8,9,3
70 FOR N=1 TO 5
80 B(N)=A(N)
90 NEXT N
100 FOR S=6 TO 10
110 C(S-5)=A(S)
120 NEXT S
```

```
130 FOR J=1 TO 5
140 PRINT B(J),C(J)
150 NEXT J
```

Bu program, üçüncü döngüdeki atama satırının ikinci döngü içinde C(N)=A(N+5) olarak yazılması ve üçüncü döngünün tamamen iptal edilmesiyle daha kısaltılabilir. Farklı sayıda elemanlara sahip olduklarından, bütün dizilerin aynı döngü içinde listelenmesi olanaksızdır. Bu örnekte dikkat edilmesi gereken, S değişkeninin değeri 6'dan 10'a giderken, C dizisinin indis değerinin (S-5) 1'den 5'e gitmesidir.

5) 7 elemanlı bir isim dizisindeki isimleri ekrana listeledikten sonra, aynı dizi içine ters olarak dizen ve diziyi bir kez daha listeleyen bir program:

```
10 DIM I$(7),X$(7)
20 FOR M=1 TO 7
30 READ I$(M)
40 PRINT I$(M)
50 NEXT M
60 DATA Handan,Zehra,Serap,Doğan,Serhat,Okan,Çetin
70 FOR F=1 TO 7
80 X$(F)=I$(8-F)
90 NEXT F
110 FOR U=1 TO 7
120 I$(U)=X$(U)
130 NEXT U
140 FOR E=1 TO 7
150 PRINT I$(E)
160 NEXT E
```

Bu programda, soruda belirtilmediği halde ikinci bir isim dizisi tanımlanmıştır. Çünkü bir diziyi kendi içinde ters olarak dizebilmek için, bu dizinin elemanlarını önce başka bir diziye ters olarak dizmek, daha sonra bu ikinci diziyi asıl diziye kopyalamak gerekmektedir. (Aynı işlem, önce birinci diziyi ikinciye kopyalamak, daha sonra ikinci dizideki bilgileri birinciye ters olarak aktarmakla da yapılabilirdi.) Yine döngü değişkenleri ve bu değişkenlerin gerek doğrudan

doğruya, gerekse bir işlem sonucunda indis olarak kullanılmaları ve buralarda aldıkları değerler önemle izlenmelidir.

6) 10 elemanlı bir sayı dizisinin sıra numarası tek olan elemanlarını DATA satırından, çift olanları klavyeden okuyan, diziyi ekrana listeleyen ve sayıların ortalamasını bularak ekrana yazan bir program:

```
10 DIM A(10)
20 FOR I=1 TO 10 STEP 2
30 READ A(I)
40 INPUT A(I+1)
50 NEXT I
60 FOR J=1 TO 10
70 PRINT A(J)
80 T=T+A(J)
90 NEXT J
100 PRINT "------"
110 PRINT T/10
120 DATA 4,7,1,9,8
```

İlk döngünün 'STEP 2' ile birlikte yazılması ve döngü içinde dizi indislerinin READ ile I, INPUT ile I+1 olarak kullanılması bu programın en önemli özelliğidir. İkinci döngüde yalnızca dizi listelenmekte ve ortalama bulunarak en sonda ekrana yazılmaktadır. (Ortalamanın dizi elemanlarıyla karıştırılmaması için araya '------' koyulmuştur.)

7) 6 öğrenciye ait isimleri DATA satırından okuyarak bir diziye yerleştirdikten sonra, her öğrencinin ismini ekrana yazarak notunu soran, daha sonra bu isimleri ve notları boş bir ekrana uygun başlıklar altında listeleyen ve en altta da not ortalamasını yazan bir program:

```
10 DIM A$(6),N(6)
20 FOR O=1 TO 6
30 READ A$(O)
40 NEXT O
50 DATA Berrin,Nuray,Oğuz,Hale,Erkan,Burçin
60 FOR T=1 TO 6
70 PRINT A$(T);
80 INPUT " NOT GİRİNİZ:";N(T)
90 NEXT T
```

```
100 CLS
110 PRINT "İSİM","NOT"
120 PRINT "-------","-------"
130 FOR K=1 TO 6
140 PRINT A$(K),N(K)
150 TOP=TOP+N(K)
160 NEXT K
170 PRINT ,"---------"
180 PRINT "ORTALAMA =",TOP/6
```

İlk döngünün ikinciyle birleştirilmesiyle kısaltılabilecek bu programın özelliği, her INPUT komutundan önce bir PRINT kullanılarak klavyeden girilecek notun kime ait olduğunun yazılmasıdır. Bundan sonra bütün bilgiler boş bir ekrana listelenmiş ve ortalama hesaplanmıştır. 170 numaralı satırdaki PRINT komutundan hemen sonra koyulmuş olan virgülün amacı, çizginin ikinci sütunun, yani notların altına gelmesini sağlamaktır. 180 numaralı satırda TOP/6'dan önceki virgül de aynı nedenle kullanılmıştır.

8) 5 elemanlı bir sayı dizisindeki eleman değerlerini DATA satırından okuduktan sonra, ikinci elemandan başlayarak her elemanı, kendisi ve kendisinden bir önceki elemanın toplamına eşitleyen ve diziyi listeleyen bir program:

```
10 DIM D(5)
20 FOR N=1 TO 5
30 READ D(N)
40 NEXT N
50 DATA 5,2,7,5,8
60 FOR L=2 TO 5
70 D(L)=D(L)+D(L-1)
80 NEXT L
90 FOR M=1 TO 5
100 PRINT D(M)
110 NEXT M
```

Karmaşık gibi görünen 70 numaralı satır, istenen toplama işleminin yapılmasını sağlamaktadır. 60 ve 80 numaralı satırlar arasındaki döngü, ikinci elemandan beşinci elemana kadar bütün elemanları kendisi ve kendisinden bir önceki elemanın toplamına eşitlemek gö-

revini üstlenmektedir. Örneğin L'nin değeri 2 olduğunda ikinci eleman D(L) kendisi ve kendisinden bir önceki elemanın, yani birinci elemanın D(L-1) toplamına eşitlenmektedir. Her eleman için döngü bu şekilde tamamlandıktan sonra son elemanın (beşinci eleman) değeri DATA satırındaki sayıların toplamına eşit olacaktır.

9) 8 elemanlı bir isim dizisine ait bilgileri DATA satırından okuyan, bu elemanları sıra numaralarıyla birlikte ekrana listeledikten sonra, klavyeden sıra numarası girilen elemanı diziden çıkartan ve diziyi listeleme işlemine geri dönen bir program: (Bir elemanın diziden çıkartılması, o elemandan sonraki elemanların birer kaydırılarak çıkartılanın yerinin doldurmasıyla yapılacaktır. Daha sonraki listelemede yalnızca dizide kalan elemanlar listelenecektir.)

```
10 DIM A$(8)
20 S=8
30 FOR Z=1 TO 8
40 READ A$(Z)
50 NEXT Z
60 DATA Ferhat,Ayşe,Zehra,Sinan,Erhun,Koray,Berk,Selin
70 FOR R=1 TO S
80 PRINT R;A$(R)
90 NEXT R
100 INPUT "KAÇ NUMARALI ELEMANI ÇIKARTMAK
İSTİYORSUNUZ ";C
110 FOR K=C+1 TO S
120 A$(K-1)=A$(K)
130 NEXT K
140 S=S-1
150 GOTO 70
```

Bu programdaki S değişkeni, dizideki isimlerin sayısını göstermektedir. En başta 8'e eşitlenen S'nin değeri, daha sonra (140 numaralı satırda) diziden çıkan her isim için bir eksiltilmektedir. Bir ismi diziden çıkartmak için 110 ve 130 numaralı satırlar arasındaki döngü kullanılmaktadır. Bu döngüdeki K değişkeninin değeri dikkatle izlenmelidir. (Bunu yapabilmek için en kolay yol, C için herhangi bir değer varsaydıktan sonra, K'nin alacağı her değeri ve dizideki hareketleri gerekirse bir kâğıt üzerine yazarak döngüyü akıldan çalıştırmak-

tır.) Döngü tamamlandıktan sonra, program tekrar 70 numaralı satıra dönecek ve diziyi yeni haliyle baştan listeleyecektir. Bütün isimler silindikten sonra programın durması için '145 IF S=0 THEN END', veya klavyeden sayı girişinde herhangi bir yanlışlık olmaması için '105 IF C>S OR C<1 THEN 100' satırları yazılabilir.

10) 9 elemanlı bir sayı dizisindeki sayılar arasında en küçük olanı ekrana yazan bir program:

```
10 DIM S(9)
20 FOR F=1 TO 9
30 READ S(F)
40 NEXT F
50 DATA 5,8,1,3,6,4,8,2,4
60 K=S(1)
70 FOR B=2 TO 9
80 IF S(B) THEN K=S(B)
90 NEXT B
100 PRINT K
```

İlk döngüde sayılar diziye yerleştirildikten sonra, en küçük sayıyı göstermek için kullanılan K değişkeni dizinin birinci elemanına eşitlenmiştir. Böylece birinci elemanın en küçük sayı olduğu varsayılmaktadır. Bu değişken daha sonraki döngüde, ikinci elemandan sonuncu elemana kadar (2 TO 9) bütün sayılarla karşılaştırılmakta ve değişkende saklı olandan daha küçük bir sayıyla karşılaşıldığında, bu küçük sayı değişkene atanmaktadır. İkinci döngü tamamlandıktan sonra, K değişkenindeki değer, dizideki en küçük sayıya eşit olacaktır. (Bu programı daha iyi anlayabilmek için, özellikle ikinci döngüde, değişken değerlerini bir kâğıda yazarak izlemek en doğru yoldur.)

NOT: Yukarıdaki örneklerin çoğu için, daha kısa yollar kullanmak mümkündür. Ancak anlaşılabilir olmaları için, programlarda daha uzun fakat basit yöntemler kullanılmıştır. Öğrenciler, bu programları denedikten sonra aynı sonuçlara ulaşabilecek daha kısa ve hızlı yolları kendileri bulmaya çalışmalıdırlar.

ÇİFT DUYARLIKLI SAYISAL DEĞİŞKENLER:

Tek duyarlıklı sayılar: Sağ taraftaki sıfırlar dışında en fazla 7 rakam içeren sayılardır. (Örnek: 4827495, 2.481957, 3781362000, 2.000004)

Çift duyarlıklı sayılar: Sağ taraftaki sıfırlar dışında 7'den çok, en fazla 16 rakam içeren sayılardır. (Örnek: 1976432579564364, 0.000000154987548)

Bilindiği gibi, bir değişkenin türünü, değişken adından sonra kullanılan karakter belirler. $ işareti alfabetik değişkeni, % işareti tamsayı değişkeni, hiç işaret kullanılmaması ise tek duyarlıklı sayısal değişkeni gösterir. Tamsayı değişkenlerinde -32767'den 32767'ye kadar olan tamsayılar, tek duyarlıklı sayısal değişkenlerde ise, en fazla 7 basamaklı sayılar saklanabilir. 7 basamaktan uzun olan sayıları, bilgisayar E notasyonuyla ifade eder:

```
A=1234567
  PRINT A
   1234567

B=12345678
  PRINT B
  1.234568E+07
```

7 basamaktan uzun sayılar bellekte saklanmak istendiğinde, çift duyarlıklı değişkenler kullanılmalıdır. Bir değişkenin çift duyarlıklı olduğu, değişken adından sonra kullanılan#işaretiyle ifade edilir (A#, TOP# gibi). Çift duyarlıklı değişkenler, en fazla 16 basamağa kadar sayıları depolayabilirler. Bundan daha uzun sayılar D notasyonuyla ifade edilir:

```
A#=1234567890123456
  PRINT A#
1234567890123456

A#=12345678901234567
  PRINT A#
1.234567890123457D+16
```

DEFINT, DEFSNG, DEFDBL ve DEFSTR komutları:

Bu komutlar, değişkenlerin türünü programın en başında belirler. Bu durumda program içinde değişken adlarından sonra kullanılan tür belirleyen karakterlerin kullanılma zorunluluğu yoktur. Genel kullanım özellikleri hepsinde aynıdır. Bu kurala bir ve daha çok boyutlu değişkenler de (diziler, matrisler) dahildir.

Genel Kullanım

Komut < Değişken adları dizisi (aralarında virgül kullanarak) >
veya
Komut < başlangıç değişken adı > - < bitiş değişken adı >

NOT: Temel BASIC kitabında da olduğu gibi, komutların kullanımları açıklanırken <> işaretlerinin arasında yazılan ifadeler mutlaka yazılması gereken, [ ] işaretleri içinde yer alan ifadeler ise programcının isterse yazabileceği bölümlerdir.

DEFINT (Define integer): İstenilen değişkenleri tamsayı değişkeni olarak tanımlar.

DEFINT A-Z

A'dan Z'ye kadar tüm harfler tamsayı değişkeni olarak tanımlanır. Bu değişkenlerde -32767'den 32767' ye kadar olan değerler saklanabilir.

DEFINT A,S

A ve S harfleri tamsayı değişkeni olarak tanımlanır.

DEFINT A,C,I-N

A, C ve I'dan N'ye kadar harfler tamsayı değişkeni olarak tanımlanır.

NOT: Bu şekilde tanımlanmış bir değişken adı, program içinde başka bir tür için de kullanılabilir. Örneğin, A değişkeni tamsayı olarak tanımlandıktan sonra, alfabetik değişken olarak kullanılacaksa A$, çift duyarlıklı değişken olarak kullanılacaksa A#, tek duyarlıklı değişken olarak kullanılacaksa A! olarak yazılabilir.

DEFSNG (Define single precision): İstenilen değişkenleri tek duyarlıklı sayısal değişkenler olarak tanımlar. Hiçbir değişken tanımlama komutu kullanılmamışsa, bu komutun da yararı yoktur. Fakat, daha önce başka bir tür olarak tanımlanmış bir değişkeni (! işareti kullanımına gerek kalmadan) tekrar tek duyarlıklı değişkene çevirmek için kullanılabilir:

```
DEFINT A-Z
DEFSNG F
F=35000
PRINT F
   35000
L=35000
Overflow
```

DEFDBL (Define double precision): İstenilen değişkenleri çift duyarlıklı sayısal değişken olarak tanımlar.

```
DEFDBL T
 T=1234567890123456
 PRINT T
 1234567890123456
```

DEFSTR (Define string): İstenilen değişkenleri alfabetik olarak tanımlar.

```
DEFSTR R
 R="ALİ"
 PRINT R
 ALİ
```

İÇ İÇE DÖNGÜLER

Döngüler, programlamanın en önemli araçlarından biridir. Bilgisayarın yorulmadan ve hatasız bir biçimde istenilen işlemleri istenildiği kadar tekrarlayabilme özelliğinden döngüler sayesinde yararlanılabilir. BASIC dilinde en çok kullanılan yöntemlerden biri, FOR ... NEXT döngüleridir. Herhangi bir işlem birden çok kez tekrarlanmak istendiğinde, başına bir FOR satırı, sonuna da bir NEXT satırı yazmak yeterlidir.

ÖRNEK: PRINT "ÇALIŞMAK" komutunu 5 kez tekrarlatarak istenen kelimeyi 5 kez yazabilmek için:

```
10 FOR A=1 TO 5
20 PRINT "ÇALIŞMAK"
30 NEXT A
```

yazmak yeterlidir. Tekrar sayısı arttırılmak istenirse, FOR satırındaki bitiş değeri için daha büyük bir sayı verilebilir.

Görüldüğü gibi, herhangi bir program bölümünü tekrarlayabilmek için, bu bölümü bir FOR ... NEXT döngüsü içine almak gerekmektedir. Bu bölüm bir döngü de olsa, durum değişmez. Bir döngünün tekrarı istendiğinde, bu döngüyü başka bir döngü içine almak mümkündür.

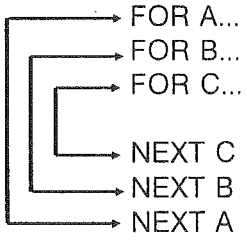
ÖRNEK: Yukarıdaki örnekteki döngünün 3 kez tekrarlanması için:

```
5 FOR B=1 TO 3
10 FOR A=1 TO 5
20 PRINT "ÇALIŞMAK"
30 NEXT A
40 NEXT B
```

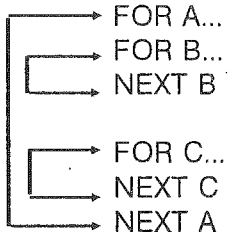
yazılabilir. 10 ve 30 numaralı satırlar arasındaki A döngüsü, ikinci bir döngü (B döngüsü) içine alınarak tekrarlatılabilir. Bu program çalıştırıldığında, 5 kere 'ÇALIŞMAK' yazma işlemi 3 kere tekrarlanarak, alt alta 15 (3*5) 'ÇALIŞMAK' elde edilir.

İç içe döngüler yazarken dikkat edilmesi gereken konu, iç döngü bitmeden dış döngünün de bitemeyeceğidir. yani, iç döngünün FOR ... NEXT deyimleri, dış döngünün FOR ... NEXT deyimlerinin içinde kalmalıdır. (İlk açılan döngü en son, son açılan döngü ilk kapanmalıdır.)

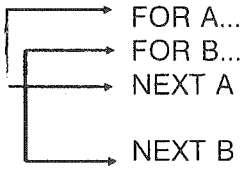
Doğru ve yanlış döngü tiplerini gösteren şemalar:



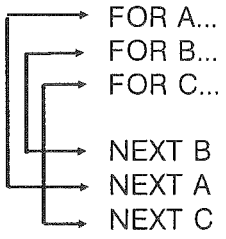
DOĞRU



DOĞRU



YANLIŞ



YANLIŞ

Örnekler:

```
1) 10 FOR P=1 TO 3
   20 PRINT P
   30 NEXT P
```

1'den 3'e kadar sayıları yazan bu döngüyü 2 kez (arada bir satır boş bırakarak) tekrar eden bir program:

```
10 FOR Z=1 TO 2
20 FOR P=1 TO 3
30 PRINT P
40 NEXT P
50 PRINT
60 NEXT Z

RUN
1
2
3

1
2
3
```

```
2) 1 7
   1 8
   1 9
   3 7
   3 8
   3 9
   5 7
   5 8
   5 9
```

Yukarıdaki sonucu ekrana çıkaran bir program:

```
10 FOR A=1 TO 5 STEP 2
20 FOR B=7 TO 9
```

```
30 PRINT A;B
40 NEXT B
50 NEXT A
```

Dış döngüde, A değişkeninin değeri 1'den 5'e kadar 2'şer artarak, iç döngüde ise B'nin değeri 7'den 9'a kadar 1'er artarak yazılacaktır.

NOT: İki NEXT komutu arasında başka bir komut yoksa, ikisi de aynı satırda yazılabilir.

```
40 NEXT B
50 NEXT A
```

yerine,

```
40 NEXT B,A
```

yazılabilir. Burada, döngü değişkenlerinin sırasına dikkat edilmelidir.

3) Bir çarpım tablosu örneği aşağıdaki programla yapılabilir:

```
10 FOR A=1 TO 10
20 FOR B=1 TO 10
30 PRINT A;"x";B;"=";A*B
40 NEXT B
50 PRINT
60 NEXT A

RUN

1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
. . .
7 x 8 = 56
7 x 9 = 63
7 x 10 =70
```

8 x 1 = 8
8 x 2 = 16
8 x 3 = 24
. . .

Bu programda 1'den 10'a kadar artış gösteren A değişkeninin her değeri için, B değişkeni de 1'den 10'a kadar artmaktadır. Böylece, 1'den 10'a kadar her sayı için, bu sayının 1'den 10'a kadar sayılarla çarpımları yazılacaktır. Her grup arasında bir satır boşluk bırkılabilmesi için 50 numaralı satırda bir PRINT komutu kullanılmıştır.

İKİ BOYUTLU DİZİLER (MATRİSLER)

Bilindiği gibi, dizilerin kullanım amacı, aynı türden birçok bilgiyi tek bir isim (dizi adı) altında bilgisayarın belleğinde saklayabilmektir. Örneğin, 20 isimden oluşan bir isim dizisini kullanmak, 20 ayrı değişkeni kullanmaktan çok daha kolaydır. Böyle bir dizide istenilen elemana ulaşabılmek için, o elemanın sıra numarasını vermek yeterlidir. Şu ana kadar işlenmiş olan diziler, tek boyutludur. Bir dizinin tek boyutlu olması, bu dizinin yalnızca uzunluğunun olması anlamına gelmektedir. Diğer bir deyişle, bu dizideki elemanlardan herhangi biri, sadece tek bir sayı (dizideki sıra numarası) ile tarif edilebilir. (5'inci eleman, 12'nci eleman, vs.)

Boyutsuz bilgilerin (sayı, kelime, harf, vs.) birleşerek tek boyutlu dizileri oluşturmalarına benzer bir şekilde, tek boyutlu diziler de birleştirilerek iki boyutlu diziler meydana getirilebilir. Yani, benzer türden birçok tek boyutlu dizinin bir isim altında bellekte saklanmasıyla, iki boyutlu diziler (matrisler) oluşturulabilir. Matrisler, satırlardan ve sütunlardan oluşur. Matrisin boyutlarından biri satır sayısını, diğeri de sütun sayısını gösterir. Bir matrisin herhangi bir elemanını tarif edebilmek için bu elemanın hangi satır ve hangi sütunda olduğunun belirtilmesi gerekir. Yani, iki parametreye ihtiyaç vardır.

Birden çok alfabetik dizi bir alfabetik matris, veya birden çok sayısal dizi bir sayısal matris olarak birleştirilebilir.

Matrisleri tanımlamak için tek boyutlu dizilerde olduğu gibi DIM komutu kullanılır. Yalnız, parantez içinde bir yerine iki indis yazılır; bu indisler arasına virgül koyulur.

ÖRNEKLER: DIM A(14,3)
DIM K$(5,6)
DIM F(5,6),H(62,3),R$(23,2)

DIM komutu kullanarak tanımlanmış bir dizinin boyut sayısı, parantez içindeki indislerin sayısına eşittir. İki boyutlu bir dizi tanımlandığında, indislerden birincisinin satır sayısını, ikincisinin de sütun sayısını gösterdiği kabul edilir. (Bunun tersinin varsayılması da olasıdır, ancak uygulamalarda standarda ulaşabilmek için ilk indisin satırı belirlemesi tercih edilir.) Örneğin, DIM A(14,3) komutuyla tanımlanmış A matrisinde 14 satır, 3 sütun vardır. (Bu matris, 14'er elemanlı 3 sayısal diziyle eşdeğerdir.)

ÖRNEK: 2 satır ve 3 sütunlu sayısal B matrisi şu şekilde tanımlanabilir:

```
DIM B(2,3)
3 5 2
2 8 6
```

Yukarıdaki sayılar B matrisine aşağıdaki komutlarla atanabilir:

```
B(1,1)=3
B(1,2)=5
B(1,3)=2
B(2,1)=2
B(2,2)=8
B(2,3)=6
```

Daha önce dizilerle yapılan uygulamalarda da olduğu gibi, bu şekilde tek tek atama yapmak özellikle büyük matrisler için uzun bir yol olduğundan programcılık anlayışına ters düşmektedir. Bu işlem bir FOR ... NEXT döngüsü yardımıyla daha basitleştirilebilir:

```
10 DIM B(2,3)
20 FOR I=1 TO 2
30 READ B(I,1),B(I,2),B(I,3)
40 NEXT I
50 DATA 3,5,2,2,8,6
```

Bu programdaki döngü değişkeni, satırı göstermektedir. Sütun sayıları da ikinci bir döngü değişkeniyle ifade edilebilir; READ B(I,1),B(I,2),B(I,3) yerine şu FOR ... NEXT döngüsü yazılabilir:

```
FOR K=1 TO 3
 READ B(I,K)
 NEXT K
```

Programın son şekli aşağıdaki gibi olacaktır:

```
10 DIM B(2,3)
20 FOR I=1 TO 2
30 FOR K=1 TO 3
40 READ B(I,K)
50 NEXT K
60 NEXT I
70 DATA 3,5,2,2,8,6
```

Bu program, elemanların tek tek atandığı ilk yoldan çok kısa değildir. Fakat, matris boyutları daha büyük olduğunda, iç içe iki döngünün kullanıldığı bu yöntem çok daha pratik olacaktır.

NOT: 50 ve 60 numaralı satırlardaki NEXT komutları birleştirilerek NEXT K,I şeklinde de yazılabilir.

Yukarıdaki programda dıştaki döngü satırları, içteki döngü sütunları belirtmektedir. İndislerde olduğu gibi, döngülerde de belirli bir sıra izlemek zorunlu değildir. Sütun döngüsü dışta, satır döngüsü içte olabilir. Fakat, yukarıdaki gibi satır döngüsünün dışta olması, özellikle listelemede kolaylık sağlar.

ÖRNEKLER:

1) Aşağıdaki sayıları sayısal bir matrise yerleştiren bir program:

```
23  54  76  26  43
58  48  63  25  14
26  32  10  45  75
56  28  64  15  20
```

```
10 DIM A(4,5)
20 FOR X=1 TO 4
30 FOR Y=1 TO 5
40 READ A(X,Y)
50 NEXT Y,X
60 DATA 23,54,76,26,43,58,48,63,25,14,26,32,10,45,75,56,28,
64, 15,20
```

Bu programda, satır ve sütunları tek tek belirterek atama yapmak son derece uzun ve anlamsız bir yol olacaktır. İç içe iki FOR ... NEXT döngüsüyle bu işlem çok basit bir şekilde gerçekleştirilebilmiştir.

Yukarıdaki matrisi ekrana listeleyebilmek için programa şu satırlar eklenebilir:

```
70 FOR R=1 TO 4
80 PRINT A(R,1),A(R,2),A(R,3),A(R,4),A(R,5)
90 NEXT R
```

Bu döngünün değişkeni, matris satırlarını göstermektedir. 80 numaralı satırdaki PRINT cümlesi de R numaralı satırdaki sütun elemanlarının yazılmasını sağlamaktadır. Bu PRINT cümlesi de, ikinci bir FOR ... NEXT döngüsüyle aşağıdaki şekilde yazılabilir:

```
FOR C=1 TO 5
PRINT A(R,C),
NEXT C
PRINT
```

Bu döngüde, PRINT komutundan sonra virgül kullanılmasının nedeni satır elemanlarının yan yana yazılmak istenmesidir. Döngüden sonra bir alt satıra geçilebilmesi için bir PRINT daha kullanılmıştır.

Programın son şekli aşağıdaki gibi olacaktır:

```
10 DIM A(4,5)
20 FOR X=1 TO 4
30 FOR Y=1 TO 5
40 READ A(X,Y)
```

```
50 NEXT Y,X
60 DATA 23, 54, 76, 26, 43, 58, 48, 63, 25, 14, 26, 32, 10, 45,
75, 56, 28, 64, 15, 20
70 FOR R=1 TO 4
80 FOR C=1 TO 5
90 PRINT A(R,C),
100 NEXT C
110 PRINT
120 NEXT R
```

Bu program, döngülerin birleştirilmesiyle daha da kısaltılabilir:

```
10 DIM A(4,5)
20 FOR X=1 TO 4
30 FOR Y=1 TO 5
40  READ A(X,Y)
50 PRINT A(X,Y),
60 NEXT Y
70 PRINT
80 NEXT X
90 DATA 23, 54, 76, 26, 43, 58, 48, 63, 25, 14, 26, 32, 10, 45,
75, 56, 28, 64,15, 20
```

Görüldüğü gibi, PRINT satırı da ilk döngüye eklenerek program kısaltılmıştır. Bu durumda, matrisin her satırını yazdıktan sonra bir alt satıra geçmek için kullanılan 70 numaralı satırdaki PRINT komutu nedeniyle iki NEXT komutu NEXT Y,X şeklinde yazılamamıştır.

2) Aşağıdaki ay isimlerini alfabetik bir matrise yerleştiren ve ekrana listeleyen bir program:

Aralık	Mart	Haziran	Eylül
Ocak	Nisan	Temmuz	Ekim
Şubat	Mayıs	Ağustos	Kasım

```
10 DIM AY$(3,4)
20 FOR I=1 TO 3
30 FOR K=1 TO 4
40 READ AY$(I,K)
50 PRINT AY$(I,K),
```

```
60 NEXT K
70 PRINT
80 NEXT I
90 DATA Aralık, Mart, Haziran, Eylül, Ocak, Nisan,
Temmuz,Ekim,Şubat,Mayıs ,Ağustos,Kasım
```

Bu listeye 'KIŞ İLKBAHAR YAZ SONBAHAR' başlıkları koyulmak istenirse, programa şu satırlar eklenebilir:

```
5 PRINT "KIŞ","İLKBAHAR","YAZ","SONBAHAR"
7 PRINT "----------","----------","----------","----------"
```

Program çalıştığında şu görüntü elde edilecektir:

KIŞ	İLKBAHAR	YAZ	SONBAHAR
Aralık	Mart	Haziran	Eylül
Ocak	Nisan	Temmuz	Ekim
Şubat	Mayıs	Ağustos	Kasım

3) Toplam gelirleri de hesaplayarak aşağıdaki listeyi ekrana çıkaran bir program:

1987 GELİRİ	1988 GELİRİ	TOPLAM
2000000	1500000	
1200000	1800000	
1000000	2100000	
1400000	2200000	

```
10 CLS
20 PRINT "1987 GELİRİ","1988 GELİRİ","TOPLAM"
30 PRINT "----------","----------","----------"
40 DIM G(4,3)
50 FOR A=1 TO 4
60 FOR B=1 TO 2
70 READ G(A,B)
80 PRINT G(A,B),
90 NEXT B
100  G(A,3)=G(A,1)+G(A,2)
110 PRINT G(A,3)
```

```
120 NEXT A
130  DATA 2000000, 1500000, 1200000,1800000, 1000000,
2100000, 1400000, 2200000
```

Bu programda, matrisin her satırı yazıldıktan sonra, 100 numaralı satırda matrisin üçüncü sütunu ilk iki sütunun toplamına eşitlenmekte ve 110 numaralı satırda bu toplam ekrana yazılmaktadır. Buradaki PRINT komutunun sonunda virgül bulunmadığı için, bilgisayar her satırdan sonra bir alttaki satıra geçecektir.

Bu örnekte, toplamlar da matrise dahil edilmiştir. Eğer istenirse, bu toplam herhangi bir değişkende saklanarak yalnızca ekrana yazma işleminde kullanılabilir:

```
100 TOP=G(A,1)+G(A,2)
110 PRINT TOP
```

Bu durumda toplamlar matriste saklanmadığı için, programın çalışması bittikten sonra toplam değerine ulaşılamayacaktır. Oysa toplamlar matriste saklanırsa, istenilen toplam daha sonra elde edilebilir. Örneğin, PRINT G(2,3) yazılırsa ikinci satırın toplamı görülebilir.

Programın sonucu aşağıdaki gibi olacaktır:

1987 GELİRİ	1988 GELİRİ	TOPLAM
2000000	1500000	3500000
1200000	1800000	3000000
1000000	2100000	3100000
1400000	2200000	3600000

4) Her satırın toplamını bularak aşağıdaki listeyi ekrana çıkaran bir program:

23	51	39	12	- -
13	43	71	20	- -
44	62	11	78	- -
12	83	11	60	- -

(Boş yerlere satır toplamları gelecek.)

```
10 DIM D(4,5)
20 FOR X=1 TO 4
30 FOR Y=1 TO 4
40 READ D(X,Y)
50 PRINT D(X,Y),
60 NEXT Y
70 D(X,5)=D(X,1)+D(X,2)+D(X,3)+D(X,4)
80 PRINT D(X,5)
90 NEXT X
100 DATA 23, 51, 39, 12, 13, 43, 71, 20, 44, 62, 11, 78, 12,
83, 11, 60
```

Görüldüğü gibi, matrisin sütun sayısı arttıkça, iki NEXT komutu arasında toplama yapmak için kullanılan satırın uzunluğu da artmaktadır. Sütun sayısının çok fazla olduğu bir durumda, bu satırın yukarıdaki gibi yazılması programcılık anlayışına ters düşecektir. Toplama işleminin daha basit bir şekilde yapılabilmesi için, bu satır yerine bir FOR ... NEXT döngüsü kullanılmalıdır:

```
FOR Z=1 TO 4
D(X,5)=D(X,5)+D(X,Z)
NEXT Z
```

Bu döngü, toplamanın kolayca yapılabilmesini sağlayacaktır. Döngü değişkeninin değeri 1'den 4'e kadar artacak ve X'inci satırdaki birinci elemandan dördüncü elemana kadar bütün değerler beşinci eleman değerinin üzerine eklenecektir. Böylece, matrisin son sütunu toplamlar sütunu olacaktır.

```
10 DIM D(4,5)
20 FOR X=1 TO 4
30 FOR Y=1 TO 4
40 READ D(X,Y)
50 PRINT D(X,Y),
60 NEXT Y
70 FOR Z=1 TO 4
80 D(X,5)=D(X,5)+D(X,Z)
90 NEXT Z
100 PRINT D(X,5)
```

```
110 NEXT X
120 DATA 23, 51, 39, 12,13,43,71,20,44,62,11, 78, 12, 83,
11, 60
```

Dikkat edilirse, toplama için kullanılan döngü değişkeni Z, birinci iç döngünün değişkeni Y ile aynı değerleri almaktadır. Toplama satırının Y döngüsüne aktarılmasıyla bu iki döngü birleştirilerek, program şu şekilde yazılabilir:

```
10 DIM D(4,5)
20 FOR X=1 TO 4
30 FOR Y=1 TO 4
40 READ D(X,Y)
50 PRINT D(X,Y),
60  D(X,5)=D(X,5)+D(X,Y)
70 NEXT Y
80 PRINT D(X,5)
90 NEXT X
100  DATA  23, 51, 39, 12, 13, 43, 71, 20, 44, 62, 11, 78, 12,
83, 11, 60
```

Program şu sonucu verecektir:

23	51	39	12	125
13	43	71	20	147
44	62	44	78	198
12	83	11	60	166

5) Yukarıdaki örnekte, her satır için toplam yerine ortalama bulunmak istenirse, program şu şekilde yazılabilir:

```
10 DIM D(4,5)
20 FOR X=1 TO 4
30 FOR Y=1 TO 4
40 READ D(X,Y)
50 PRINT D(X,Y),
60 D(X,5)=D(X,5)+D(X,Y)
70 NEXT Y
80 D(X,5)=D(X,5)/4
```

```
90 PRINT D(X,5)
100 NEXT X
110 DATA 23, 51, 39, 12, 13, 43, 71, 20, 44, 62, 11, 78, 12,
83, 11, 60
```

Bu programa, ortalamanın bulunması için yalnızca 80 numaralı satırdaki D(X,5)=D(X,5)/4 komutu eklenmiştir. Bu komut, satır toplamına eşit olan beşinci sütun elemanının dörde bölünerek yine kendi hücresinde saklanmasıyla ortalamanın matrise yerleştirilmesini ve 90 numaralı satırdaki PRINT komutuyla ekrana yazılmasını sağlamaktadır.

Programın sonucu aşağıdaki şekilde olacaktır:

23	51	39	12	31.25
13	43	71	20	36.75
44	62	11	78	48.75
12	83	11	60	41.5

6) Aşağıdaki sayıları M matrisinde saklayan ve ikinci sütunun toplamını bularak listeyi ekrana çıkartan bir program:

5	2	1	7
3	8	6	2
9	7	4	2
2	4	8	2
2	3	7	1
	—		
Toplam	....		

Burada toplam, ikinci sütunun bütün elemanlarını toplamakla elde edilecektir. T2=M(1,2)+M(2,2)+M(3,2)+M(4,2)+M(5,2) satırı yazılarak toplam bulunabilir. Diğer örneklerde olduğu gibi, bu satır da bir FOR ... NEXT döngüsü yardımıyla basitleştirilebilir:

```
FOR A=1 TO 5
T2=T2+M(A,2)
NEXT A
```

Bu döngü, program içinde aşağıdaki gibi kullanılabilir:

```
10 DIM M(5,4)
20 FOR P=1 TO 5
30 FOR Q=1 TO 4
40 READ M(P,Q)
50 PRINT M(P,Q),
60 NEXT Q
70 PRINT
80 NEXT P
90 DATA  5,2,1,7,3,8,6,2,9,7,4,2,2,4,8,2,2,3,7,1
100 FOR A=1 TO 5
110 T2=T2+M(A,2)
120 NEXT A
130 PRINT ,"---"
140 PRINT " TOPLAM:",T2
```

100 ve 120 numaralı satırlar arasındaki döngü, ikinci sütunun toplamını bulacak ve bu toplam daha sonra ekrana yazılacatır. Döngü değişkeni A, daha önce dış döngü olarak kullanılan P döngüsünün değişkeniyle aynı değerleri almaktadır. Toplama satırı (110 numaralı satır) dış döngüye aktarılarak program basitleştirilebilir:

```
10 DIM M(5,4)
20 FOR P=1 TO 5
30 FOR Q=1 TO 4
40 READ M(P,Q)
50 PRINT M(P,Q),
60 NEXT Q
70 PRINT
80 T2=T2+M(P,2)
90 NEXT P
100 DATA  5, 2,1,7,3,8,6,2,9,7,4,2,2,4,8,2,2,3,7,1
110 PRINT ,"---"
120 PRINT " TOPLAM:",T2
```

Bu programda, toplamın hesaplandığı satırın yeri çok önemlidir. Eğer bu satır yanlışlıkla iç döngüye koyulursa, çıkacak sonuç çok farklı olacaktır. (Bu örnekte, doğru sonucun üç katına eşit olacaktır.)

Toplama işlemini gerçekleştiren bu komut, matrisin her satırı işlendikten sonra, bir alt satıra geçmeden önce verilmelidir. Diğer bir deyişle, bu komut iç döngü (sütun döngüsü) kapandıktan sonra, dış döngü (satır döngüsü) kapanmadan önce yer almalıdır.

7) Yukarıdaki örnekte, ikinci sütunun toplamı yerine ortalaması bulunmak istenirse, program aşağıdaki şekilde değiştirilebilir:

```
10 DIM M(5,4)
20 FOR P=1 TO 5
30 FOR Q=1 TO 4
40 READ M(P,Q)
50 PRINT M(P,Q),
60 NEXT Q
70 PRINT
80 T2=T2+M(P,2)
90 NEXT P
100 ORT=T2/5
110 DATA  5, 2,1, 7,3,8,6,2,9,7,4,2,2,4,8,2,2,3,7,1
120 PRINT ,"---"
130 PRINT "ORTALAMA:",ORT
```

Ortalama, ikinci sütunun toplamı için kullanılan T2 değişkeninin sütundaki sayı adedine (matrisin satır sayısı) bölünmesiyle elde edilmiştir.

8) Matris kullanmak yoluyla, aşağıdaki listeyi ekrana çıkartan bir program:

AYLARA VE YILLARA GÖRE BİLGİSAYAR FİYATLARI:

AY	1986	1987	1988
Ocak	800	990	990
Şubat	700	1100	1000
Mart	730	980	1100
Nisan	820	1000	1100
Mayıs	830	950	1300
Haziran	900	920	1200
Temmuz	870	940	1050
Ağustos	910	1100	980
Eylül	860	1700	1000
Ekim	880	1200	940
Kasım	890	1200	980
Aralık	840	950	990

Bu matriste, hem alfabetik, hem sayısal bilgiler karışık olarak yer almaktadır. Bu durumda bir çözüm olarak, alfabetik bilgiler için alfabetik, sayısal bilgiler için sayısal matris kullanılabilir. Burada alfabetik bilgiler tek bir sütundan oluştuğundan, ay adları için tek boyutlu alfabetik bir dizi kullanılabilir.

```
10 PRINT " AYLARA VE YILLARA GÖRE BİLGİSAYAR
FİYATLARI:"
20 PRINT "----------------------------------------"
30 PRINT "AY","1986","1987","1988"
40 PRINT "---------","-------","--------","--------"
50 DIM AY$(12),F(12,3)
60 FOR X=1 TO 12
70 READ AY$(X)
80 NEXT X
90 DATA Ocak,Şubat,Mart,Nisan,Mayıs,Haziran
100 DATA Temmuz,Ağustos,Eylül,Ekim,Kasım,Aralık
110 FOR X=1 TO 12
120 FOR Y=1 TO 3
130 READ F(X,Y)
140 NEXT Y,X
150 DATA 800, 990, 990, 700, 1100, 1000, 730, 980, 1100,
820, 1000, 1100
```

```
160 DATA 830, 950, 1300, 900, 920, 1200, 870, 940, 1050,
910, 1100, 980
170 DATA 860, 1700, 1000, 880, 1200, 940, 890, 1200, 980,
840, 950, 990
180 FOR X=1 TO 12
190 PRINT AY$(X),F(X,1),F(X,2),F(X,3)
200 NEXT X
```

Bu programda sonucu elde edebilmek amacıyla birçok döngüden yararlanılmıştır. İlk döngü alfabetik diziyi DATA satırından okumak için, daha sonra iç içe iki döngü sayısal matrisi okumak için ve en son olarak bir döngü de bilgileri ekrana listelemek için kullanılmıştır. DATA satırındaki bilgileri okuyan döngüler aşağıdaki şekilde birleştirilebilir:

```
FOR X=1 TO 12
READ AY$(X)
FOR Y=1 TO 3
READ F(X,Y)
NEXT Y
NEXT X
```

Matris elemanı olmayan ay adları, iki FOR satırının arasına yazılarak yukarıdaki gibi okunabilir. (Döngü değişkenlerinin alacağı değerler birer birer kâğıt üzerinde incelenerek yukarıdaki bölüm daha iyi anlaşılabilir.) PRINT komutları da bu döngülerin içine yerleştirilerek, program daha da kısaltılabilir:

```
FOR X=1 TO 12
READ AY$(X)
PRINT AY$(X),
FOR Y=1 TO 3
READ F(X,Y)
PRINT F(X,Y),
NEXT Y
PRINT
NEXT X
```

Bu programın daha öncekilerden farkı, ekrana çıkacak listenin hem alfabetik hem de sayısal bilgilerden oluşmasıdır. Görüldüğü gi-

bi, alfabetik bilgilerin saklandığı dizi elemanlarının DATA'dan okunması ve ekrana yazılması için, iki FOR satırı arasında (dış döngünün içinde, iç döngüden sonra) READ ve PRINT komutlarının yazılması yeterlidir.

Burada çok dikkat edilmesi gereken bir nokta, döngülerin birleştirilerek programın kısaldığı durumda, bilgilerin okunma sıralarının değişmesidir. Daha önce ay adları ve sayıların iki ayrı grup halinde yazılmasına karşın, bu kez DATA satırı yazılırken, aylar ve sayılar karışık olarak (READ komutuyla okundukları sırada) yazılmalıdır. Programın son şekli aşağıdaki gibi olacaktır:

```
10 PRINT "  AYLARA VE YILLARA GÖRE BİLGİSAYAR
FİYATLARI:"
20 PRINT "-----------------------------------------------"
30 PRINT "AY","1986","1987","1988"
40 PRINT "---------","--------","--------","---------"
50 FOR X=1 TO 12
60 READ AY$(X)
70 PRINT AY$(X),
80 FOR Y=1 TO 3
90 READ F(X,Y)100 PRINT F(X,Y),
110 NEXT Y
120 PRINT
130 NEXT X
140 DATA Ocak, 800, 990, 990, Şubat, 700, 1100, 1000, Mart,
730, 980, 1100
150  DATA  Nisan, 820, 1000,  1100,Mayıs, 830, 950, 1300,
Haziran,900,920,1200
160  DATA  Temmuz, 870, 940, 1050, Ağustos, 910, 1100,
980, Eylül,980,1700,1000
170 DATA Ekim, 880, 1200, 940, Kasım, 890, 1200, 980,
Aralık, 840, 950, 990
```

Görüldüğü gibi, DATA satırındaki bilgilerin sırası, ilk programa göre farklıdır. Sıra, READ komutlarının uygulanma sırasına uygun olarak düzenlenmiştir. (Tabii ki, DATA satırının yukarıdaki şekilde gruplara ayrılması zorunlu değildir.)

9) Her mevsim için toplam geliri de hesaplayarak, aşağıdaki listeyi ekrana çıkartan bir program:

MEVSİMLERE GÖRE TURİZM GELİRLERİ ($ olarak)

MEVSİM	YERLİ	YABANCI	TOPLAM
İlkbahar	4000	10000	
Yaz	7000	24000	
Sonbahar	3000	15000	
Kış	6000	4000	

```
10 CLS
20 PRINT "  MEVSİMLERE GÖRE TURİZM GELİRLERİ
($ olarak)"
30 PRINT "-----------------------------------------------"
40 PRINT "MEVSİM","YERLİ","YABANCI","TOPLAM"
50 PRINT "-------","--------","--------","---------"
60 DIM M$(4),G(4,3)
70 FOR D=1 TO 4
80 READ M$(D)
90 PRINT M$(D),
100 FOR E=1 TO 2
110 READ G(D,E)
120 PRINT G(D,E),
130 G(D,3)=G(D,3)+G(D,E)
140 NEXT E
150 PRINT G(D,3)
160 NEXT D
170  DATA  İlkbahar, 4000, 10000,Yaz,7000,24000
180  DATA Sonbahar, 3000, 15000, Kış,6000,4000
```

Bu örnekte, bir öncekinden farklı olarak satır toplamlarının bulunması istenmiştir. Bu amaçla, 130 numaralı satırda bir toplama işlemi yapılmış ve bu işlemde elde edilen sonuç iki NEXT komutu arasında 150 numaralı satırda ekrana yazılmıştır.

Program çalıştırıldığında, şu sonuç elde edilecektir:

MEVSİMLERE GÖRE TURİZM GELİRLERİ ($ olarak)

MEVSİM	YERLİ	YABANCI	TOPLAM
İlkbahar	4000	10000	14000
Yaz	7000	24000	31000
Sonbahar	3000	15000	18000
Kış	6000	4000	10000

10) Yukarıdaki listeyi yerli turizm gelirlerinin de toplamını bularak ekrana çıkartan bir program:

```
10 CLS
20 PRINT " MEVSİMLERE GÖRE TURİZM GELİRLERİ
($ olarak)"
30 PRINT "-----------------------------------------"
40 PRINT "MEVSİM","YERLİ","YABANCI","TOPLAM"
50 PRINT "--------","---------","---------","---------"
60 DIM M$(4),G(4,3)
70 FOR D=1 TO 4
80 READ M$(D)
90 PRINT M$(D),
100 FOR E=1 TO 2
110 READ G(D,E)
120 PRINT G(D,E),
130 G(D,3)=G(D,3)+G(D,E)
140 NEXT E
150 PRINT G(D,3)
160 T1=T1+G(D,1)
170 NEXT D
180 DATA İlkbahar, 4000, 10000, Yaz, 7000,24000
190 DATA Sonbahar,3000,15000,Kış,6000,4000
200 PRINT ,"--------"
210 PRINT " TOPLAM:",T1
```

Yerli turizm gelirlerinin, yani G matrisinin birinci sütununun toplamını hesaplamak için, programa 160 numaralı satırdaki toplama işlemi eklenmiştir. Döngü boyunca artan bu toplam, program sonunda ait olduğu sütunun altına gelecek şekilde yazdırılmıştır.

NOT: 130 ve 160 numaralı satırlarda iki ayrı toplama işlemi yapılmıştır. bunlardan birincisi, yani satır toplamını bulan, iki döngünün de içinde, sütun toplamını bulan ikincisi ise yalnızca dış döngünün içinde kalmaktadır. Daha önce de belirtildiği gibi, bu satırların yerleri doğru sonuca ulaşmak açısından çok önemlidir. Toplamların yazıldığı PRINT satırlarının yerleri de döngüler çerçevesinde dikkatle incelenmelidir. (Ayrıntılı açıklamalar için önceki örneklerden yararlanılabilir.) Sonuç aşağıdaki gibi olacaktır:

MEVSİMLERE GÖRE TURİZM GELİRLERİ ($ olarak)

MEVSİM	YERLİ	YABANCI	TOPLAM
İlkbahar	4000	10000	14000
Yaz	7000	24000	31000
Sonbahar	3000	15000	18000
Kış	6000	4000	10000
TOPLAM:	20000		

11) Aşağıda isimleri ve üç sınavdan aldıkları notlar verilen her öğrenci için karne notunu hesaplayan, ayrıca her sınav için ve sınıf genelinde not ortalamasını bulan ve bilgileri ekrana listeleyen bir program:

SINAV SONUÇLARI

İSİM	1. NOT	2. NOT	3. NOT	KARNE NOTU
Ayça	6	3	7	
Deniz	8	9	5	
Tülay	4	5	4	
Serpil	9	10	8	
Şeniz	3	6	4	

Bu örnekte de, isimler için alfabetik bir dizi ve notlar için sayısal bir matris kullanılacaktır. Karne notlarının tamsayı olması gerektiğinden, ortalama hesaplanırken çıkan kesirli sonuçları yuvarlamak amacıyla buradaki matrisin tamsayı olarak tanımlanması daha doğrudur.

```
10 CLS
20 PRINT "          SINAV SONUÇLARI"
30 PRINT "-------------------------------------------------------------"
40 PRINT "İSİM","1. NOT","2. NOT","3. NOT","KARNE NOTU"
```

```
50 PRINT "-------","-------","--------","--------","----------"
60 DIM I$(5),N%(5,4)
70 FOR R=1 TO 5
80 READ I$(R)
90 PRINT I$(R),
100 FOR C=1 TO 3
110 READ N%(R,C)
120 PRINT N%(R,C),
130 N%(R,4)=N%(R,4)+N%(R,C)
140 NEXT C
150 N%(R,4)=N%(R,4)/3
160 PRINT N%(R,4)
170 T1=T1+N%(R,1)
180 T2=T2+N%(R,2)
190 T3=T3+N%(R,3)
200 T4=T4+N%(R,4)
210 NEXT R
220 PRINT "-------","-------","-------","-------","-------"
230 PRINT "ORTALAMA:",T1/5,T2/5,T3/5,T4/5
240 DATA Ayça,6,3,7,Deniz,8,9,5,Tülay,4,5,4,Serpil,9,10,8,
Şeniz,3,6,4
```

Bu programda, daha önceki örneklerde uygulanan bütün işlemler kullanılmıştır. Satır toplamları bulunduktan sonra, 3'e bölünerek satır ortalamaları, yani karne notları hesaplanmaktadır. Daha sonra, her sütunun toplamı bulunmakta ve programın sonunda bu toplamın 5'e bölümü yani sınav ortalamaları ekrana yazılmaktadır. Program şu sonucu verecektir:

SINAV SONUÇLARI

İSİM	1. NOT	2. NOT	3. NOT	KARNE NOTU
Ayça	6	3	7	5
Deniz	8	9	5	7
Tülay	4	5	4	4
Serpil	9	10	8	9
Seniz	3	6	4	4
ORTALAMA:	6	6.6	5.6	5.8

12) Matris kullanmak yoluyla, aşağıdaki listeyi ekrana çıkartan, haftalık toplam masrafı da hesaplayarak toplamlar sütununun altına yazan bir program:

HAFTALIK MASRAF LİSTESİ

GÜNLER	YİYECEK	GİYİM	EĞLENCE	TOPLAM
PAZARTESİ	12000	8000	1000	
SALI	4000	2000	2000	
ÇARŞAMBA	5000	4000	0	
PERŞEMBE	8000	0	0	
CUMA	1000	1000	1000	
CUMARTESİ	4000	5000	2000	
PAZAR	3000	0	4000	

```
10 CLS
20 PRINT "      HAFTALIK MASRAF LİSTESİ"
30 PRINT "-------------------------------------------------------------"
40 PRINT "GÜNLER","YİYECEK","GİYİM","EĞLENCE",
"TOPLAM"
50 PRINT "---------","---------","--------","----------","----------"
60 DIM G$(7),M(7,4)
70 FOR G=1 TO 7
80 READ G$(G)
90 PRINT G$(G),
100 FOR K=1 TO 3
110 READ M(G,K)
120 PRINT M(G,K),
130 M(G,4)=M(G,4)+M(G,K)
140 NEXT K
150 PRINT M(G,4)
160 T4=T4+M(G,4)
170 NEXT G
180 PRINT ,,,,"--------"
190 PRINT ,,"   TOPLAM MASRAF",T4
200 DATA PAZARTESİ,12000,8000,1000,SALI,4000,2000,2000
210 DATA ÇARŞAMBA,5000,4000,0,PERŞEMBE,8000,0,0
220 DATA CUMA,1000,1000,1000, CUMARTESİ, 4000, 5000,
2000, PAZAR,3000,0,4000
```

Program şu sonucu verecektir:

HAFTALIK MASRAF LİSTESİ

GÜNLER	YİYECEK	GİYİM	EĞLENCE	TOPLAM
PAZARTESİ	12000	8000	1000	21000
SALI	4000	2000	2000	8000
ÇARŞAMBA	5000	4000	0	9000
PERŞEMBE	8000	0	0	8000
CUMA	1000	1000	1000	3000
CUMARTESİ	4000	5000	2000	11000
PAZAR	3000	0	400	7000
		TOPLAM	MASRAF:	67000

13) Aşağıdaki bilgilere göre stoktaki her mal için tutar hesaplayarak listeyi ekrana çıkartan ve toplam tutarı bularak ekrana yazan bir program:

STOK LİSTESİ

MAL ADI	ADET	BİRİM FİYATI	TUTAR
Kitaplık	12	650000	
Bilgisayar	5	3200000	
Yazı masası	18	580000	
Koltuk	22	130000	

Bu örnekte, tutarlar sütunundaki sayıların bulunabilmesi için yapılması gereken işlem daha önceki örneklerdeki gibi toplama değil, çarpmadır. Bu nedenle, işlemin iç döngü dışında yer alması gerekli-dir:

```
10 CLS
20 PRINT "                STOK LİSTESİ"
30 PRINT "-----------------------------------------------"
40 PRINT "MAL ADI","ADET","BİRİM FİYAT","TUTAR"
50 PRINT "---------","--------","--------","----------"
60 DIM M$(4),A(4,3)
70 FOR K=1 TO 4
80 READ M$(K)
90 PRINT M$(K),
100 FOR L=1 TO 2
110 READ A(K,L)
```

```
120 PRINT A(K,L),
130 NEXT L
140 A(K,3)=A(K,1)*A(K,2)
150 PRINT A(K,3)
160 TS=TS+A(K,3)
170 NEXT K
180 PRINT ,,,"----------"
190 PRINT ,,"TOPLAM STOK:",TS
200 DATA Kitaplık,12,650000,Bilgisayar,5,3200000
210 DATA Yazı masası,18,580000,Koltuk,22,130000
```

Bu programda, tutarların hesaplandığı 140 numaralı satır, iç döngünün dışında kalmaktadır, çünkü satırın sonunda elde edilmek istenen sonuç, bir tek işlemle bulunabilmektedir. Daha önceki örneklerdeki gibi, bir dizi işleme, dolayısıyla bir FOR ... NEXT döngüsüne ihtiyaç yoktur. Program şu sonucu verecektir:

STOK LİSTESİ

MAL ADI	ADET	BİRİM FİYAT	TUTAR
Kitaplık	12	650000	7800000
Bilgisayar	5	3200000	16000000
Yazı masası	18	580000	10440000
Koltuk	22	130000	28600000
		TOPLAM STOK:	37100000

14) Yıllara göre gelir ve giderlerin verildiği aşağıdaki tablodan yararlanarak, her yıl için kâr miktarını hesaplayan ve 6 yıllık toplam kârı da bularak listeyi ekrana çıkartan bir program:

A ŞİRKETİ GELİR - GİDER LİSTESİ

YIL	GELİR	GİDER	KAR
1983	4000000	2000000	
1984	6000000	3000000	
1985	6000000	5000000	
1986	11000000	5000000	
1987	13000000	8000000	
1988	15000000	7000000	

Bu örnekte bütün bilgiler sayısal olduğundan, matrisin dışında ek olarak bir de alfabetik dizi kullanmaya gerek yoktur.

```
10 CLS
20 PRINT " A ŞİRKETİ GELİR - GİDER LİSTESİ"
30 PRINT "-----------------------------------------"
40 PRINT "YIL","GELİR","GİDER","KAR"
50 PRINT "-------","----------","---------","----------"
60 DIM K(6,4)
70 FOR F=1 TO 6
80 FOR I=1 TO 3
90 READ K(F,I)
100 PRINT K(F,I)
110 NEXT I
120 K(F,4)=K(F,2)-K(F,3)
130 PRINT K(F,4)
140 TK=TK+K(F,4)
150 NEXT F
160 PRINT ,,,"----------"
170 PRINT ,,"TOPLAM KAR:",TK
180  DATA  1983, 4000000, 2000000, 1984, 6000000,
3000000, 1985, 6000000, 5000000
190  DATA 1986, 11000000, 5000000, 1987, 13000000,
8000000, 1988, 15000000 ,7000000
```

Bu örnekte de, kârların hesaplandığı satır iç döngünün dışında kalmaktadır. Nedeni, bir önceki programda da olduğu gibi, sonucu bulmak için tek bir işlemin yeterli olmasıdır. Programın sonucu aşağıdaki gibi olacaktır:

A ŞİRKETİ GELİR - GİDER LİSTESİ

YIL	GELİR	GİDER	KAR
1983	4000000	2000000	2000000
1984	6000000	3000000	3000000
1985	6000000	5000000	1000000
1986	11000000	5000000	6000000
1987	13000000	8000000	5000000
1988	15000000	7000000	8000000
		TOPLAM KAR:	25000000

ALFABETİK BİLGİLERLE SAYISAL İŞLEMLER:

Şimdiye kadar yapılan örneklerde, alfabetik ve sayısal bilgilerin bir arada bulunduğu listelerle ilgili sayısal işlemleri yapabilmek için, farklı türden bilgiler bellekte iki ayrı bölüm (alfabetik dizi ve sayısal matris) halinde saklanmıştır. Bilindiği gibi, alfabetik bilgiler sayısal değişkenlerde veya matrislerde saklanamaz. Örneğin, A="AHMET" gibi bir komut, hata mesajı verecektir. Buna karşın, sayılar, birer karakter olarak değerlendirilmek koşuluyla, alfabetik matrislerde depolanabilirler. Örneğin, A$="34" yazarak 34 sayısı alfabetik bir değişkende saklanabilir; aynı şekilde, M$(1,3)="410" komutuyla, 410 sayısı alfabetik M$ matrisine yerleştirilebilir. BASIC dilinin bu özelliğinden yararlanarak, bir bölümü alfabetik, bir bölümü sayısal olan bir bilgi grubu, tek bir alfabetik matriste saklanabilir.

ÖRNEK: Aşağıdaki bilgileri alfabetik bir matriste saklayarak ekrana listeleyen bir program:

AYLARA VE YILLARA GÖRE BİLGİSAYAR FİYATLARI:

AY	1986	1987	1988
Ocak	800	990	990
Şubat	700	1100	1000
Mart	730	980	1100
Nisan	820	1000	1100
Mayıs	830	950	1300
Haziran	900	920	1200
Temmuz	870	940	1050
Ağustos	910	1100	980
Eylül	860	1700	1000
Ekim	880	1200	940
Kasım	890	1200	980
Aralık	840	950	990

```
10 PRINT " AYLARA VE YILLARA GÖRE BİLGİSAYAR
FİYATLARI:"
20 PRINT "-----------------------------------------"
30 PRINT "AY","1986","1987","1988"
40 PRINT "--------","-------","--------","--------"
```

```
50 DIM B$(12,4)
60 FOR A=1 TO 12
30 FOR Y=1 TO 4
40 READ B$(A,Y)
50 PRINT B$(A,Y),
60 NEXT Y
70 PRINT
80 NEXT A
140 DATA Ocak, 800, 990, 990, Şubat, 700, 1100, 1000, Mart,
730, 980, 1100
150 DATA Nisan, 820, 1000, 1100, Mayıs, 830, 950, 1300,
Haziran,900,920,1200
160 DATA Temmuz, 870, 940, 1050, Ağustos,910,1100,980,
Eylül,980,1700,1000
170 DATA Ekim,880,1200,940,Kasım,890,1200,980,
Aralık,840,950,990
```

VAL ve STR$ FONKSİYONLARI:

Alfabetik olarak belleğe kaydedilmiş olan sayısal bilgilerle aritmetiksel işlem yapılmak istendiğinde, bu bilgilerin önce sayıya çevrilmeleri gerekir. Bunun için, VAL fonksiyonu kullanılır.

ÖRNEKLER:

1)
```
A$="45"
B$="13"
PRINT A$+B$
4513
```

Alfabetik bilgiler sayıya çevrilmeden toplandığında, elde edilen sonuç sayısal toplam değil, karakterlerin ard arda eklenmesiyle oluşan bir şekil dizisidir. Aynı değişkenleri sayısal anlamda toplayabilmek için VAL fonksiyonu şu şekilde kullanılabilir:

```
PRINT VAL(A$)+VAL(B$)
58
```

Bu işlemde, A$ ve B$ değişkenlerinin her biri VAL fonksiyonuyla sayısal değer olarak yorumlanmış ve 45 ve 13 sayılarının toplamı olan 58 sayısı elde edilmiştir.

2) Sayısal bilgiler içeren 5 elemanlı alfabetik bir dizinin eleman değerlerinin 3 katını ekrana yazan bir program:

```
10 DIM A$(5)
20 FOR K=1 TO 5
30 READ A$(K)
40 PRINT VAL(A$(K))*3
50 NEXT K
60 DATA 7,3,1,5,8
```

Yukarıdaki sayılar, alfabetik bir dizide saklandıkları için, karakter özelliğini taşımaktadır. Bu nedenle, bu dizinin herhangi bir isim dizisinden farkı yoktur. Bu sayılarla doğrudan doğruya aritmetiksel işlem yapılamaz. Örneğin 'PRINT A$(K)*3' komutu, 'Type mismatch' (Tip uyuşmazlığı) hata mesajının alınmasına neden olacaktır. Sayısal işlem yapabilmek için, VAL fonksiyonundan yararlanılmıştır.

Alfabetik bilgilerin sayısal değerlere çevrilebilmeleri gibi sayısal bilgilerin de alfabetik olarak yorumlanabilmesi mümkündür. BASIC dilinde bu amaçla STR$ fonksiyonu kullanılmaktadır.

Aşağıdaki örnek, STR$ fonksiyonunun daha iyi anlaşılması için yardımcı olacaktır.

```
10 A=5
20 B=8
30 A$=STR$(A)
40 B$=STR$(B)
50 C$=A$+B$
60 PRINT C$
RUN
5  8
```

Yukarıdaki programda, 5 ve 8 sayıları, STR$ fonksiyonuyla çevrilerek alfabetik A$ ve B$ değişkenlerine atanmıştır. Daha sonra bu iki değişkenin toplamı C$ değişkenine atanmış ve bu değişken ekrana

yazılmıştır. Sonuç, A ve B değişkenlerinin sayısal toplamları değil, 5 ve 8 karakterlerinin yan yana yazdırılmasıdır, çünkü buradaki toplama işlemi aritmetiksel toplama değil, karakter bilgilerin ard arda eklenmesidir. Dikkat edilirse, 5 ve 8'den önce birer boşluk bırakılmıştır. Bu boşluk, sayısal bilgilerin başında, sayının negatif olması durumunda (-) işareti için bilgisayar tarafından ayrılmıştır. A ve B değişkenlerindeki sayılar, A$ ve B$ değişkenlerine aktarılırken, bu boşluk da bir karakter olarak saklanmıştır.

ÖRNEKLER:

1) 8 sayıdan oluşan alfabetik bir dizinin elemanlarının her birinin iki katını ikinci bir alfabetik diziye aktaran ve iki diziyi de listeleyen bir program:

```
10 DIM A$(8),B$(8)
20 FOR I=1 TO 8
30 READ A$(I)
40 S=VAL(A$(I))
50 T=S*2
60 B$(I)=STR$(T)
70 PRINT A$(I),B$(I)
80 NEXT I
90 DATA 4,2,8,4,5,3,9,1
```

Yukarıdaki programda, A$ dizisinin her elemanı önce sayıya çevrilerek S değişkeninde saklanmış, bu değişkenin değerinin iki katı T değişkenine aktarılmış ve bu sonuç STR$ fonksiyonuyla alfabetik B$ dizisine atanmıştır.

Programın 40 ve 50 numaralı satırları birleştirilerek aşağıdaki şekilde yazılabilir:

```
T=VAL(A$(I))*2
```

Yukarıdaki ifade, 60 numaralı satırdaki T değişkeni yerine kullanılabilir:

```
B$(I)=STR$(VAL(A$(I))*2)
```

Bu durumda, programın son şekli aşağıdaki gibi olabilir:

```
10 DIM A$(8),B$(8)
20 FOR I=1 TO 8
30 READ A$(I)
40  B$(I)=STR$(VAL(A$(I))*2)
50 PRINT A$(I),B$(I)
60 NEXT I
70 DATA  4, 2, 8, 4, 5, 3, 9, 1
```

Bu programdaki 40 numaralı satır, oldukça karmaşık görünmektedir. Bu tür satırları yazarken çok dikkatli olmak gekerir. Eğer bir yanlışlıkla karşılaşılırsa, ilk dikkat edilmesi gereken, parantezlerdir. Bir program satırında, açılan parantezlerin sayısı, kapanan parantezlerin sayısına eşit olmalıdır. Yine de yanlışlığın nedeni bulunamıyorsa, en doğru yol bu karmaşık satırı bölümlere ayırarak yukarıdaki programın ilk yazıldığı şekilde satırları adım adım yazmaktır. Böylece yanlışlığın nedeni daha kolay bulunabilecektir.

2) Klavyeden girilen sayısal bilgileri 6 elemanlı alfabetik bir diziye yerleştirdikten sonra diziyi listeleyen, daha sonra dizi elemanlarının DATA satırından okunan sayılarla çarpımlarını bularak tekrar diziye atayan ve diziyi ikinci bir kez listeleyen bir program: (Dizinin listelenmesi için alt program kullanılabilir.)

```
10 DIM S$(6)
20 FOR I=1 TO 6
30 INPUT S$(I)
40 NEXT I
50 GOSUB 200
60 FOR A=1 TO 6
70 READ T
80 C=T*VAL(S$(A))
90 S$(A)=STR$(C)
100 NEXT A
110 DATA 6,3,7,1,2,9
120 GOSUB 200
130 END
200 FOR K=1 TO 6
210 PRINT S$(K) 220 NEXT K
230 RETURN
```

MATRİS ÖRNEKLERİ: (Önceki matris örneklerinin VAL ve STR$ ile çözümleri:)

1)

MEVSİMLERE GÖRE TURİZM GELİRLERİ ($ olarak)

MEVSİM	YERLİ	YABANCI	TOPLAM
İlkbahar	4000	10000	
Yaz	7000	24000	
Sonbahar	3000	15000	
Kış	6000	4000	

TOPLAM:	.....		

```
10 CLS
20 PRINT " MEVSİMLERE GÖRE TURİZM GELİRLERİ
($ olarak)"
30 PRINT "-----------------------------------------"
40  PRINT "MEVSİM","YERLİ","YABANCI","TOPLAM"
50 PRINT "--------","---------","---------","---------"
60 DIM G$(4,4)
70 FOR M=1 TO 4
80 FOR Y=1 TO 3
90 READ G$(M,Y)
100 PRINT G$(M,Y),
110 G$(M,4)=STR$(VAL(G$(M,4))+VAL(G$(M,Y)))
120 NEXT Y
130 PRINT G$(M,4)
140 T2=T2+VAL(G$(M,2))
150 NEXT M
160 PRINT ,"--------"
170 PRINT "TOPLAM:",T2
180  DATA İlkbahar,4000, 10000,Yaz,7000,24000
190  DATA Sonbahar, 3000,15000, Kış,6000,4000
```

2)

SINAV SONUÇLARI

İSİM	1. NOT	2. NOT	KARNE NOTU
Ayça	6	3	7
Deniz	8	9	5
Tülay	4	5	4
Serpil	9	10	8
Şeniz	3	6	4
ORTALAMA:	....	....	....

```
10 CLS
20 PRINT " ----------SINAV SONUÇLARI"----------
30 PRINT "------------------------------------------------------------"
40 PRINT "İSİM"," 1. NOT","2. NOT","3. NOT","KARNE NOTU"
50 PRINT "-------","-------","--------","--------","----------"
60 DIM S$(5,5
70 FOR T=1 TO 5
80 FOR K=1 TO 4
90  READ  S$(T,K)
100 PRINT S$(T,K),
110  S$(T,5) =STR$ (VAL(S$(T,5)) + VAL(S$(T,K)))
120 NEXT K
130  S$(T,5)=STR$(VAL(S$(T,5))/3)
140 PRINT S$(T,5)
150 T2=T2+VAL(S$(T,2))
160 T3=T3+VAL(S$(T,3))
170 T4=T4+VAL(S$(T,4))
180 T5=T5+VAL(S$(T,5))
190 NEXT K
200 PRINT "-------","--------","--------","-------","--------"
210 PRINT "ORTALAMA:",T2/5,T3/5,T4/5,T5/5
220 DATA Ayça,6,3,7,Deniz,8,9,5,Tülay,4,5,4,Serpil,9,10,8,
Şeniz,3,6,4
```

Burada, satır ortalamalarının bulunabilmesi için, 130 numaralı program satırında, satırın toplamı, yani 5'inci sütun elemanı, üçe bölünerek ortalama elde edilmiştir. Bu ortalama da, yine aynı hücrede saklanmıştır. Bu satır, aşağıdaki şekilde açık olarak yazılabilir:

```
TOP=VAL(S$(T,5))
ORT=TOP/3
S$(T,5)=STR$(ORT)
```

3)

STOK LİSTESİ

MAL ADI	ADET	BİRİM FİYAT	TUTAR
Kitaplık	12	650000	
Bilgisayar	5	3200000	
Yazı masası	18	580000	
Koltuk	22	130000	----------
		TOPLAM STOK:.	.....

```
10 CLS
20 PRINT "                    STOK LİSTESİ"
30 PRINT "------------------------------------------------------------------"
40 PRINT "MAL ADI","ADET","BİRİM FİYAT","TUTAR"
50 PRINT "----------","----------","----------","----------"
60 DIM F$(4,4)
70 FOR M=1 TO 4
80 FOR B=1 TO 3
90 READ F$(M,B)
100 PRINT F$(M,B),
110 NEXT B
120  F$(M,4 )= STR$ (VAL (F$ (M, 2)) *VAL(F$(M,3)))
130 PRINT F$(M,4)
140 NEXT M
150 PRINT ,,,"----------"
160 PRINT ,,"TOPLAM STOK:",T4
170  DATA Kitaplık,12,650000,Bilgisayar,5,3200000
180  DATA Yazı masası,18,580000,Koltuk,22,130000
```

Daha önce sayısal matris kullanılarak yapılan çözümde olduğu gibi, yukarıdaki programda da tutarları hesaplayan satır iç döngünün dışında kalmaktadır. Bilindiği gibi, bunun nedeni, bu sonucu bulmak için tek bir işlemin yeterli olmasıdır.

4)

A ŞİRKETİ GELİR - GİDER LİSTESİ

YIL	GELİR	GİDER	KÂR
1983	4000000	2000000	
1984	6000000	3000000	
1985	6000000	5000000	
1986	11000000	5000000	
1987	13000000	8000000	
1988	15000000	7000000	
	TOPLAM KÂR:		.....

```
10 CLS
20 PRINT " A ŞİRKETİ GELİR - GİDER LİSTESİ"
30 PRINT "-----------------------------------------"
40 PRINT "YIL","GELİR","GİDER","KAR"
50 PRINT "-------","----------","----------","----------"
60 DIM G$(6,4)
70 FOR Y=1 TO 6
80 FOR X=1 TO 3
90 READ G$(Y,X)
100 PRINT G$(Y,X), 110 NEXT X
120 G$(Y,4)=STR$(VAL(G$(Y,2))-VAL(G$(Y,3)))
130 PRINT G$(Y,4)
140 TK=TK+VAL(G$(Y,4))
150 NEXT Y
160 PRINT ,,,"--------"
170 PRINT ,,"TOPLAM KAR:",TK
180 DATA 1983, 4000000, 2000000, 1984, 6000000, 3000000,
1985, 6000000, 5000000
190  DATA 1986, 11000000, 5000000, 1987, 13000000,
8000000, 1988, 15000000, 7000000
```

5) İsimleri, bankaya yatırdıkları anapara ve faiz oranı verilen kişilerin, tablodaki süre sonunda bankadan çekebilecekleri para miktarını hesaplayarak görüldüğü şekilde listeleyen bir program:

BANKA FAİZLERİ LİSTESİ

İSİM	ANAPARA	FAİZ ORANI	SÜRE(yıl)	ANAPARA+FAİZ
HASAN	1000000	14	6	
MİNE	200000	8	4	
CELAL	800000	43	5	
EMEL	56000	1	2	

```
10 CLS
20 PRINT "          BANKA FAİZLERİ LİSTESİ"
30 PRINT "------------------------------------------------------------"
40 PRINT "İSİM","ANAPARA","FAİZ ORANI","SÜRE(yıl)",
"ANAPARA+FAİZ"
50 PRINT "---------","--------","----------","----------","----------"
60 DIM B$(4,5)
70 FOR I=1 TO 4
80 FOR F=1 TO 4
90 READ B$(I,F)
100 PRINT B$(I,F),
110 NEXT F
120 FAIZ=VAL(B$(I,2))*VAL(B$(I,3))/100*VAL(B$(I,4))
130 B$(I,5)=STR$(VAL(B$(I,2))+FAIZ)
140 NEXT I
150  DATA HASAN,1000000,14,6,MİNE,200000,81,4
160  DATA CELAL,800000,43,5,EMEL,560000,17,2
```

120 numaralı satır, aşağıdaki şekilde açık olarak yazılabilir:

```
APAR=VAL(B$(I,2))
ORAN=VAL(B$(I,3))
SURE=VAL(B$(I,4))
FAIZ=APAR*ORAN*SURE/100
PARA=APAR+FAIZ
B$(I,5)=STR$(PARA)
```

SORULAR:

1)

12 AYLIK METEOROLOJİ RAPORU

AYLAR	1. ÖLÇÜM	2. ÖLÇÜM	3. ÖLÇÜM	ORTALAMA
OCAK	12	13	10	--
ŞUBAT	9	12	14	--
MART	12	15	16	--
NİSAN	18	17	19	--
MAYIS	21	25	24	--
HAZİRAN	23	26	28	--
TEMMUZ	26	25	29	--
AĞUSTOS	28	32	31	--
EYLÜL	28	27	24	--
EKİM	21	21	18	--
KASIM	16	16	14	--
ARALIK	15	16	13	--
				.....
		YILLIK SICAKLIK ORTALAMASI:		--

Bilgileri bir matriste saklayarak, yukarıdaki şekilde ekrana listeleyen, her ay için ortalama sıcaklığı bulduktan sonra 12 aylık ortalama sıcaklığı da hesaplayan bir program yapınız.

2)

PERSONEL ZAM LİSTESİ

ADI	TECRÜBE(YIL)	ESKİ MAAŞI	YENİ MAAŞI	ZAM
HAMDİ	4	200000	320000	--
ALİ	6	270000	300000	--
YUSUF	2	240000	280000	--
HANDE	8	180000	210000	--
MUTLU	12	430000	500000	--

		PERSONELE VERİLEN TOMLAM ZAM		--

Matris kullanarak, yukarıdaki bilgileri listeleyen, her memurun ne kadar zam (yeni maaş - eski maaş) aldığını ve personele verilen toplam zam miktarını hesaplayan bir program yapınız.

3)

OKULLAR ARASI YARIŞMA

OKUL ADI	1987 PUANI	1988 PUANI	1989 PUANI	ORTALAMA
ŞİŞLİ LİSESİ	90	92	78	--
KADIKOY L.	86	45	95	--
KARTAL L.	99	100	96	--
PENDİK L.	25	42	18	--
ÜSKÜDAR L.	88	75	66	--

Yukarıda, 5 liseye ait yarışma sonuçları verilmiştir. Bu bilgileri bir matriste saklayarak, her lisenin son üç yılda aldığı puanların ortalamasını bularak ekrana listeleyen bir program yapınız.

FONKSİYONLAR

Fonksiyonlar, daha önceden bilgisayarın belleğine yerleştirilmiş olan özel programları çağıran terimlerdir. Makine dilinde yazılmış olan bu programlar, programlamada çok sık gereksinim duyulan işlemlerin gerçekleştirilmesinde kolaylık sağlarlar. Örneğin karekök bulmak için yapılacak oldukça uzun bir program yerine, karekök alan fonksiyonu kullanmak bu işlemi son derece basitleştirir ve hızlandırır. Her fonksiyon için giriş ve çıkış değerleri söz konusudur. (Yalnız bazı fonksiyonların giriş değerleri, programcı tarafından doğrudan doğruya verilemez.) Fonksiyonla birlikte giriş değeri (veya değerleri) verildikten sonra, bilgisayar bu değere göre gerekli işlemleri yaparak fonksiyon sonucunu (çıkışını) bildirir. Komutlardan farklı olarak işlev gören fonksiyonlar, tek başlarına değil, mutlaka bir komutla birlikte kullanılmalıdırlar.

Daha önce işlenmiş olan LEFT$, RIGHT$ ve MID$ fonksiyonları, bir karakter dizisi ve sayıdan yararlanarak yeni bir karakter dizisi üretmek için kullanılırlar. LEN fonksiyonu ise giriş olarak bir karakter dizisini alıp, çıkış olarak sayısal bir değer verir. Hemen hemen bütün fonksiyonlar için gerekli olan giriş değerleri, sayısal veya alfabetik sabitler olabileceği gibi, uygun değişkenler de olabilir. ("BİLGİSAYAR" yerine A$ veya 56 yerine A kullanılabilir.) Dikkat edilecek bir başka nokta ise, alfabetik çıkış veren fonksiyon adlarından sonra '$' işaretinin bulunması, sayısal çıkış verenlerde ise bu işaretin olmamasıdır.

BASIC dilindeki fonksiyonlardan bazıları aşağıda, diğerleri de gerektiğinde konularla birlikte açıklanacaktır.

CHR$ FONKSİYONU:

Karakterler (harfler, sayılar, işaretler, vs.) bilgisayarın belleğinde ekranda görüldüğü şekilde değil, birer sayı olarak saklıdır. Karakterler, bilgisayar için geliştirilmiş ASCII (American Standard Code for Information Interface) sistemine göre kodlanmıştır. Bu sistem, birçok bilgisayar için standart olarak kabul edilmiştir. (Ancak, harfler, sayılar ve özel işaretler dışındaki karakterler çeşitli bilgisayarlarda farklılık gösterebilir.)

CHR$ fonksiyonu, kod numarası verilen karaktere ulaşabilmek için kullanılır.

ÖRNEKLER:

```
PRINT CHR$(66)
B       (B harfinin ASCII koduna göre sıra numarası 66'dır.)
A$=CHR$(83)+CHR$(46)+CHR$(79)+CHR$(46)+CHR$(83)
PRINT A$
S.O.S   (Burada, 'S', '.' ve 'O' karakterlerinin kodları
         kullanılarak bir karakter dizisi oluşturulmuş ve
        ekrana yazılmıştır.)
```

İngiliz alfabesini yazan bir program:

```
10 FOR H=65 TO 90
20 PRINT CHR$(H);" ";
30 NEXT H

RUN

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

Burada karakterler, H değişkeninin aldığı değerlere göre, sıra numarası 65 olan A harfinden 90 numaralı Z harfine kadar yan yana ekrana yazılmıştır. Türkçede bulunan Ç, Ğ, İ, Ö, , Ü harflerinin bu dizide yer almamasının nedeni, bu harflerin ASCII standardına uymaması ve bilgisayar tarafından harf olarak tanınmamasıdır.

STRING$ FONKSİYONU:

CHR$ fonksiyonuyla çok benzer bir işleve sahip olan STRING$ fonksiyonunun kullanım amacı, belirli bir karakterden oluşan bir karakter dizisi üretmektir. Bu fonksiyonun iki şekilde yazılabilme özelliği vardır:

1-STRING$(A,C) (C kod numaralı karakterden A adet)
2 - STRING$(A,C$) (C$ karakterinden A adet)

ÖRNEKLER:

```
PRINT STRING$(10,65)
AAAAAAAAAA
```

(10 adet A harfinden oluşan bir karakter dizisi üretilmiştir.)

```
PRINT STRING$(10,"A")
AAAAAAAAAA
```

(Aynı sonuç, karakterin kendisi verilerek elde edilmiştir.)

```
CIZGI$=STRING$(40,45)
PRINT CIZGI$
----------------------------------------
```

(40 adet '-' işareti CIZGI$ değişkeninde saklanmış ve ekrana yazılmıştır.)

NOT: Birinci yazılım şekli, genellikle klavyeden elde edilemeyen karakterler için tercih edilir. Bu karakterler program içine doğrudan doğruya yazılamayacağı için, STRING$ fonksiyonuyla ASCII kod numaraları kullanılır. Karakterlerin doğrudan verildiği ikinci yazılım şekli ise, klavyeden elde edilebilen karakterler için daha kolay kullanılabilmektedir.

ASC FONKSİYONU:

Görevi CHR$ fonksiyonunun tam tersi olan ASC fonksiyonu, herhangi bir karakterin kod numarasını (ASCII sıra numarasını) elde etmek için kullanılır. Yani, giriş değeri olan alfabetik değerden sayısal bir çıkış üretir.

ÖRNEKLER:

```
PRINT ASC("A")
65                (A harfinin karakter kodu 65'tir.)

A$="-"
C=ASC(A$)
PRINT C
45                (- işaretinin karakter kodu 45'tir.)

PRINT ASC("BİLGİSAYAR")
66
```

NOT: Son örnekte de görüldüğü gibi, giriş bilgisi birden çok karakterden oluşuyorsa, ASC fonksiyonu bu dizideki ilk karakterin kod numarasını verecektir.

Klavyeden girilen bir kelimenin bütün harflerinin karakter kodlarını yazan bir program:

```
10 INPUT A$
20 FOR I=1 TO LEN(A$)
30 H$=MID$(A$,I,1)
40 PRINT ASC(H$);
50 NEXT I

RUN

? PROGRAM
80  82  79  71  82  65  77
```

TAMSAYI FONKSİYONLARI:

Programlamada, bazı durumlarda kesirli sayısal değerlerin tamsayıya çevrilmesi gerekir. Örneğin karne notları hesaplanırken, sınav notlarının ortalaması alınıp bu ortalama en yakın tamsayıya yuvarlanır. (4.5 5'e, 4.4 ise 4'e yuvarlanır.) BASIC dilinde, çeşitli amaçlara göre, değişik yöntemlerle yuvarlama yapan farklı fonksiyonlar vardır .

CINT FONKSİYONU:

CINT fonksiyonu, kesirli sayıları en yakın tamsayıya yuvarlamak için kullanılır:

ÖRNEKLER:

```
PRINT CINT(4.2)
4
PRINT CINT(4.7)
5
PRINT CINT(3)
3
PRINT CINT(2.45)
2
PRINT CINT(2.5)
3
```

Yukarıdaki örneklerde görüldüğü gibi, CINT fonksiyonu giriş değeri olan sayısal bilgiye en yakın tamsayıyı çıkış değeri olarak verir; yani, buçuk (.5) ve yukarısını bir üst tamsayıya, aşağısını ise bir alttaki tamsayıya yuvarlar. Eğer giriş değeri tamsayıysa, sonuç bu sayının kendisidir. Negatif değerler için de benzer bir sonuç elde edilir:

```
PRINT CINT(-5.8)
-6
PRINT CINT(-5.2)
-5
```

```
PRINT CINT((4+5)/2)
5
```

(İşlemin sonucu olan 4.5 sayısı, 5'e yuvarlanmıştır.)

Klavyeden girilen üç derse ait notların ortalamasını bularak karne notunu ekrana yazan bir program:

```
10 INPUT N1,N2,N3
20 PRINT CINT((N1+N2+N3)/3)

RUN

? 5,7,8
7
```

(6.66 olan ortalama 7'ye yuvarlanmıştır.)

FIX FONKSİYONU:

FIX fonksiyonu, kesirli sayıların tamsayı bölümünü almak için kullanılır. Bu fonksiyon, kesirli sayıların kesir bölümlerini iptal eder.

ÖRNEKLER:

```
PRINT FIX(4.2)
4
PRINT FIX(4.7)
4
PRINT FIX(3)
3
PRINT FIX(2.5)
2
PRINT FIX(2.999)
2
```

FIX fonksiyonu, CINT fonksiyonunda olduğu gibi negatif sayılar için de benzer sonuçlar verir:

```
PRINT FIX(-4.2)
-4
PRINT FIX(-7.9)
-7
```

INT FONKSİYONU:

INT fonksiyonu, giriş değeri olarak verilen kesirli sayıyı kendisinden küçük olan ilk tamsayıya yuvarlar. Giriş değeri tamsayıysa, diğer fonksiyonlarda da olduğu gibi sonuç bu sayının kendisidir.

ÖRNEKLER:

```
PRINT INT(4.2)
4
PRINT INT(4.7)
4
PRINT INT(3)
3
PRINT INT(2.5)
2
PRINT INT(2.999)
2
```

Görüldüğü gibi, pozitif sayılarla birlikte kullanıldığında INT fonksiyonuyla elde edilen sonuçlar FIX'le aynıdır. Farklılık, negatif sayılarda ortaya çıkar:

```
PRINT INT(-4.2)
-5              (-4.2'den küçük ilk tamsayı -5'tir.)
PRINT INT(-7.9)
-8              (-7.9'dan küçük ilk tamsayı -8'dir.)
PRINT INT(-12.1)
-13             (-12.1'den küçük ilk tamsayı -13'tür.)
```

INT fonksiyonu sayıları kendilerinden küçük ilk tamsayıya yuvarladığından, negatif sayılarla birlikte kullanıldığında, yukarıdaki örneklerde de görülebileceği gibi, diğer tamsayı fonksiyonlarından farklı sonuçlar verir. (Negatif sayılar küçüldükçe mutlak değerlerinin arttığına dikkat edilmelidir.)

CINT fonksiyonunun kullanılamadığı bazı BASIC yorumlayıcılarda, kesirli sayıları INT fonksiyonunu kullanarak en yakın tamsayıya yuvarlamak için, giriş değerine 0.5 eklemek gereklidir.

Daha önce CINT fonksiyonuyla yapılmış olan karne notu hesaplama programı INT ile aşağıdaki şekilde yapılabilir:

```
10 INPUT N1,N2,N3
20 PRINT INT((N1+N2+N3)/3+0.5)

RUN

? 5,7,8
7
```

Buradaki ortalama, 6.66'dır. INT fonksiyonuyla 6'ya yuvarlanacak olan bu sonuç, 0.5 eklenerek 7.16'ya çıkartılmıştır. 7.16 sayısı INT fonksiyonuyla 7'ye yuvarlanmış ve ortalamaya en yakın tamsayıya ulaşılmıştır. Aynı program başka verilerle çalıştırılırsa:

```
RUN

? 4,7,8
6
```

Bu notların ortalaması, 6.33'tür. 0.5 eklendiğinde 6.66'ya ulaşan bu sonuç, INT fonksiyonuyla 6'ya yuvarlanmıştır. Sonuç, ortalamaya en yakın tamsayıdır.

ÖRNEK:

5.2, 2.8, -4.3, 6.9, -0.6 ve 5 sayılarının INT, CINT ve FIX fonksiyonlarıyla kullanıldıklarında sonuç olarak elde edilecek sayıları aşağıdaki gibi bir tablo halinde listeleyen bir program:

SAYI	INT	CINT	FIX
5.2			
2.8			
-4.3			
6.9			
-0.6			
5			

```
10 CLS
20 PRINT " SAYI"," INT"," CINT"," FIX"
30 PRINT "------","------","------","------"
40 FOR S=1 TO 6
50 READ A
60 PRINT A,INT(A),CINT(A),FIX(A)
70 NEXT S
80 DATA 5.2,2.8,-4.3,6.9,-0.6,5
```

Program aşağıdaki sonucu verecektir:

SAYI	INT	CINT	FIX
5.2	5	5	5
2.8	2	3	2
-4.3	-5	-4	-4
6.9	6	7	6
-0.6	-1	-1	0
5	5	5	5

NOT: Yukarıdaki örnek değişik sayılarla denenerek, tamsayı fonksiyonlarının her birinin görevi daha iyi anlaşılabilir.

MATEMATİKSEL FONKSİYONLAR:

Bu bölümde, BASIC dilindeki birçok fonksiyondan matematiksel fonksiyonlar incelenecektir. Bu fonksiyonların hem giriş, hem de çıkış değerleri sayısaldır.

ABS: (Absolute Value)

Sayısal bilgilerin mutlak değerlerini vermek için kullanılır. Diğer bir deyişle, pozitif sayıları olduğu gibi bırakırken, negatif sayıları pozitife çevirir.

ÖRNEKLER:

```
PRINT ABS(-4)
4
PRINT ABS(5.3)
 5.3
PRINT ABS(5-12)
 7
```

SGN: (Sign)

Giriş değeri olan sayının işaretini (negatif veya pozitif olduğunu) verir. Giriş pozitifse sonuç 1, negatifse sonuç -1, sıfıra eşitse sıfırdır.

ÖRNEKLER:

```
PRINT SGN(5)
1
PRINT SGN(2-1-1)
0
PRINT SGN(-6)
-1
```

SQR: (Square Root)

Pozitif sayıların karekköklerini almak için kullanılır. Negatif sayılarla kullanıldığında, (negatif sayıların karekökleri olmadığı için) bir hata mesajının çıkmasına neden olacaktır.

ÖRNEKLER:

```
PRINT SQR(81)
9
PRINT SQR(-4*-16)
8
PRINT SQR(-2)
Improper argument (PC: Illegal Function Call)
```

SIN, COS, TAN, ATN: (Sine, Cosine, Tangent, Arctangent)

Sırasıyla Sinüs, Kosinüs, Tanjant ve Arktanjant fonksiyonlarını ifade eden bu terimler, genellikle trigonometriyle ilgilenen programcılar için kullanışlıdır.

ÖRNEKLER:

```
PRINT SIN(1)
0.841471
```

PRINT COS(3)
-0.9899925

PRINT TAN(0.5)
0.5463025

PRINT ATN(5)
1.373401

PRINT ATN(0.5463025)
0.5

PRINT TAN(1.373401)
5

Son iki örnekte de görüldüğü gibi, ATN ve TAN fonksiyonları, birbirlerinin tersidir.

LOG: (Logarithm)

Giriş değeri olan sayının doğal logaritmasını verir. (Giriş değerinin, 2.7182818'e eşit olan E sayısının kaçıncı kuvveti olduğunu hesaplar.)

ÖRNEK:

PRINT LOG(3)
1.098612

EXP: (Exponent)

Giriş değeri olan sayıyı E sayısının kuvveti kabul ederek sonuç verir.

ÖRNEK:

```
PRINT EXP(2)
7.389056
```

ATN ve TAN fonksiyonlarında olduğu gibi, LOG ve EXP fonksiyonları da birbirlerinin tersidir:

```
PRINT EXP(1.098612)
3
```

MATEMATİKSEL FONKSİYONLARLA İLGİLİ ÖRNEKLER:

1) 1'den 10'a kadar olan sayıların sinüs, kosinüs, tanjant ve arktanjantlarını uygun başlıklar altında listeleyen bir program:

```
10 CLS
20 PRINT "SİNÜS","KOSİNÜS","TANJANT","ARKTANJANT"
30 PRINT "-----","-------","-------","----------"
40 FOR I=1 TO 10
50 PRINT SIN(I),COS(I),TAN(I),ATN(I)
60 NEXT I
```

2) ABS fonksiyonu kullanılmadan, klavyeden girilen herhangi bir sayının mutlak değerini yazan bir program:

A)
```
10 INPUT A
20 IF A<Ø  THEN A=-A
30 PRINT A
```

B)
```
10 INPUT A
20 PRINT A * SGN(A)
```

C)
```
10 INPUT A
20 PRINT SQR(A*A)
```

Bu problemin çözümü için üç ayrı program verilmiştir. Bu programların hepsinde elde edilecek sonuç aynıdır; yani, klavyeden girilen sayı pozitifse olduğu gibi, negatifse pozitife dönüştürülerek ekrana yazılmaktadır.

A çözümünde, girilen sayı negatifse, tersi alınarak pozitife dönüştürülmektedir.

B çözümünde, sayı kendi işaretiyle çarpılmaktadır. Bu durumda sayı negatifse -1 ile çarpılarak pozitife dönüştürülmektedir.

C çözümünde, sayının karesi alındıktan sonra, bu sonucun karekökü alınmaktadır. Girilen sayı negatifse kendisiyle çarpılarak pozitif bir sonuç elde edilmekte, daha sonra bu sonucun karekökü alınarak, sayı pozitife dönüştürülmektedir.

3) İki hentbol takımının yaptıkları karşılaşmada attıkları gol sayıları klavyeden verildiğinde, aradaki farkı ekrana yazan bir program: (Aradaki fark, negatif olamaz.)

A)
```
10 INPUT A,B
20 IF A>B THEN PRINT A-B ELSE PRINT B-A
```

B)
```
10 INPUT A,B
20 PRINT ABS(A-B)
```

Aradaki fark negatif olarak ifade edilemeyeceğinden, A çözümünde, IF...THEN kullanılarak, büyük sayıdan küçük sayının çıkarılması yoluna gidilmiştir. Daha kısa olan B çözümünde ise, farkın mutlak değeri alınarak sonucun pozitif olarak yazılması sağlanmıştır.

4) Klavyeden girilen sayının karekökünü yazan bir program: (Sayı negatifse, pozitifinin karekökü yazılacaktır.)

A)
```
10 INPUT A
20 IF A<Ø THEN PRINT SQR(-A) ELSE PRINT SQR(A)
```

B)
```
10 INPUT A
20 PRINT SQR(ABS(A))
```

Programda hata çıkmaması için, klavyeden girilen sayı negatifse, pozitife dönüştürülmektedir. A çözümünde bunun için bir IF...THEN cümlesi kullanılmıştır. B çözümünde ise, ABS fonksiyonundan yararlanılarak aynı sonuç elde edilmiştir.

RND FONKSİYONU:

RND fonksiyonu, 0 ve 1 arasında rasgele bir sayı üretmek için kullanılır:

```
PRINT RND
0.8126816
PRINT RND
0.06845308
PRINT RND
0.09586931
PRINT RND
0.9644445
```

Yukarıdaki örneklerdeki rasgele sayılar, aslında bilgisayar tarafından belirli bir yönteme göre üretilirler. Aynı BASIC yorumlayıcıya sahip bilgisayarlar aynı rasgele sayı üretim algoritmasını kullanırlar; dolayısıyla, bu bilgisayarlar ilk açıldıklarında belirli bir sayıdan başlayarak, belirli bir rasgele sayı dizisi yaratırlar. Diğer bir deyişle, bu sayılar aslında rasgele değildirler. Buna karşın, kullanıcı bilgisayarın kullandığı rasgele sayı üretim algoritmasını ve RND fonksiyonunun vereceği bir sonraki değeri bilmediği için, RND fonksiyonunun rasgele sayı ürettiği kabul edilir.

RND fonksiyonuyla elde edilen sayı, istenilen bir değerle çarpılarak, 0 ve 1 olan sınırlar değiştirilebilir. Örneğin, RND fonksiyonu 5 ile çarpıldığında, üretilecek rasgele sayının değeri 0 ve 5 arasında olacaktır:

```
PRINT RND*5
2.3817263
PRINT RND*5
4.2938064
```

Bu durumda çıkabilecek en küçük sayı 0.0000001, en büyük ise 4.9999999 sayılarıdır. Eğer RND*5 değeri INT fonksiyonuyla tamsayıya çevrilirse, en küçük sayı 0, yani INT(0.0000001), en büyük ise 4, yani INT(4.9999999) olacaktır:

```
PRINT INT(RND*5)
3
```

```
PRINT INT(RND*5)
4
PRINT INT(RND*5)
0
PRINT INT(RND*5)
2
```

ÖRNEKLER:

1) En küçük 0, en büyük 4 olabilecek şekilde 10 rasgele tamsayı üreten ve ekrana yazan bir program:

```
10 FOR I=1 TO 10
20 PRINT INT(RND*5);
30 NEXT I
```

Bu programın sonucu aşağıdaki gibi olabilir:

```
3 2 4 0 4 1 1 3 2 0
```

2) En küçük 0, en büyük 35 olacak şekilde 8 rasgele tamsayı üreten ve ekrana yazan bir program:

```
10 FOR T=1 TO 8
20 PRINT INT(RND*36)
30 NEXT I
```

NOT: Birçok BASIC yorumlayıcıda, RUN komutu rasgele sayı üretim algoritmasının en baştan başlayarak çalışmasına neden olur. Bu yüzden rasgele sayı üreten bir program her çalıştırıldığında aynı sayıları üretir. Böyle bir duruma engel olmak için RANDOMIZE komutu kullanılır. Programın başına yazılabilecek bu komut, program her çalıştırıldığında klavyeden bir sayı ister ve bu sayıya göre rasgele sayı üretimini farklı bir yerden başlatır. (Bu komutla ilgili ayrıntılar daha sonra verilecektir.)

Rasgele sayı üretiminin amaçlarından biri, özellikle çok fazla bilgi girişi gerektiren programlar için programlama aşamasında yeterli sayıda sınama (test) değeri üretmek ve böylece programcı açısından zaman kaybını önlemektir. Örneğin, yüzlerce sayı girişini gerektiren

bir muhasebe programında deneme yapabilmek için programın her çalışmasında bu sayıları girerek büyük ölçüde zaman kaybetmek yerine, programcı RND fonksiyonunu gerektiği şekilde kullanarak istenilen sayıları üretebilir ve böylece programını defalarca hiç zaman kaybetmeden ve yorulmadan deneme fırsatını bulmuş olur. Bu şekilde denenen bir programın yanlışları bulunduktan ve düzeltildikten sonra, gerçek veriler girilerek program en az bir kez daha denenir ve kullanıma hazır olur.

RND fonksiyonuyla bu tür bir çalışma yapabilmek için, rasgele sayıların istenilen sınırlar içinde üretilebilmesi gerekmektedir. Bu sınırlar, rasgele sayıların kullanıldığı yere göre değişebilir. Örneğin bir malın fiyatı RND fonksiyonu yardımıyla üretilirse, 0 veya negatif olmamalıdır; veya hava sıcaklığı için üretilen rasgele sayı mantıklı sınırlar içinde olmalıdır. Sınırların istendiği gibi ayarlanabilmesi için RND fonksiyonuyla birlikte çarpmanın yanısıra, toplama veya çıkarma işlemleri de kullanılır.

PRINT INT(RND*5) komutu, 0 ve 4 arasında rasgele bir tamsayı üretir. Bu sonuca 1 eklenirse, sınırlar 1 ve 5 olarak değiştirilmiş olur:

PRINT INT(RND*5)+1 komutuyla üretilecek rasgele tamsayı, en küçük 1, en büyük 5 olabilir.

ÖRNEK: Bir tavla oyunu için zar atan program: (İki zar atılacak ve her zarda elde edilecek sayı 1 ve 6 arasında olacaktır.)

```
10 FOR Z=1 TO 2
20 PRINT INT(RND*6)+1;
30 NEXT Z

RUN
4  2
RUN
5  1
RUN
6  6
```

Program her çalıştırıldığında 1 ve 6 arasında rasgele iki sayı elde edilecektir. Bütün sayıların çıkma olasılığı aynıdır.

İlk sınırın (başlangıç sayısı) 1'den farklı bir değer olması istenirse, aşağıdaki formül ulgulanmalıdır:

```
INT ( RND * ( S2 - S1 + 1 ) ) + S1
```

Bu formüldeki S1 değeri alt sınırı, S2 değeri ise üst sınırı göstermektedir. (S2-S1+1) değeri de, birinci sınırdan ikinci sınıra kadar sayı adedini (üretilebilecek sayı adedi) vermektedir.

ÖRNEKLER:

1) 5 ile 10 arasında rasgele bir tamsayı üreten bir komut:

```
PRINT INT(RND*6)+5
```

Buradaki 6 sayısı, 10 - 5 + 1 işleminin sonucu olarak elde edilmiştir. Bu sayı, üretilebilecek (5'ten 10'a kadar) sayı olasılıklarını göstermektedir. En sonda 5 eklenmesindeki amaç, 0 ile 5 arasında üretilecek olan bu sayının 5 ve 10 sınırları arasına çıkartılmasıdır.

2) 43 ile 69 arasında 4 rasgele tamsayı üreten bir program:

```
10 FOR R=1 TO 4
20 PRINT INT(RND*27)+43
30 NEXT R
```

Bu program, sonuç olarak 4 sayı verecektir. Doğruluğundan emin olabilmek için, programın birçok kez çalıştırılması gerekir, çünkü elde edilecek sonuçlar rasgele sayılar olduğundan, ilk denemede doğru görünen sayıların, daha sonra yanlış bir algoritmayla üretildiği anlaşılabilir. Bu tür programlarda hata yapmamak için RND fonksiyonuyla birlikte kullanılan işlemlerin yazımına çok dikkat etmek gerekir.

3) Adetleri klavyeden girilecek 10 ve 20 arasındaki rasgele sayıları ekrana listeleyen ve ortalamalarını bulan bir program:

```
10 INPUT "KAÇ ADET SAYI ÜRETİLECEK";N
20 FOR S=1 TO N
30 R=INT(RND*11)+10
40 PRINT R
50 T=T+R
60 NEXT S
70 PRINT "ORTALAMA =";T/N
```

Burada, 10 ve 20 arasında üretilen rasgele sayılar R değişkeninde saklanıp, önce ekrana yazılmakta ve T değişkenine eklenerek bu sayıların toplamı bulunmaktadır. En son olarak, toplamı gösteren T değişkeni sayı adedinin saklı olduğu N değerine bölünerek sayıların ortalaması ekrana yazılmaktadır. Bu ortalamanın, normal olarak 15 veya 15'e çok yakın bir sayı olması gerekmektedir. N sayısı ne kadar büyük olursa, ortalama da 15'e o kadar yakın olacaktır. (15, (10+20)/2 işleminden elde edilmektedir.)

4) -5 ile 5 arasında 10 adet rasgele sayı üreten bir program:

```
10 FOR D=1 TO 10
20 PRINT INT(RND*11)-5
30 NEXT D
```

Rasgele sayı üretiminde sınırların belirlendiği formül, bu programda da kullanılmıştır. 11 sayısı 5 - (-5) + 1 işleminden elde edilmiştir. INT fonksiyonunun sonucuna alt sınır eklenmiş, yani 5 çıkartılmıştır.

RANDOMIZE KOMUTU:

Bilindiği gibi, rasgele sayılar aslında bilgisayar tarafından belirli bir algoritmaya göre üretilirler, yani gerçekte tam olarak rasgele değildirler. Yalnız bu algoritma kullanıcı tarafından bilinmediği için, sayılar rasgele olarak kabul edilir. Bütün BASIC yorumlayıcılarında, rasgele sayılar için belirli bir başlangıç değeri vardır. Bilgisayar açıldıktan sonra uygulanan ilk RND fonksiyonu, aynı yorumlayıcıyı kullanan bilgisayarlarda aynı sonucu verecektir. Bundan sonraki sayılar

da, sırayla aynı olacaktır. Böyle bir durum, gerçek anlamda rasgele sayı kullanımına engel olacaktır. Daha önce de belirtildiği gibi, rasgele sayı üretimini bu başlangıç değerinden farklı bir sayıdan başlatmak için RANDOMIZE komutu kullanılır. Komutun iki farklı kullanım şekli vardır:

A) RANDOMIZE komutu tek başına yazılabilir:

Bu şekilde yazıldığında, bilgisayar bu komutu uygulamak için programı durdurur; aynı INPUT komutunda olduğu gibi klavyeden girilecek sayısal bir değer ister. Bu sayısal değere göre, rasgele sayı üretim algoritması başlangıçtan farklı bir yerden başlar. Yine de, bu sayıya göre üretilen bir sonraki rasgele sayının ne olacağı kullanıcı tarafından bilinemez. Burada da, BASIC yorumlayıcıya özgü bir algoritma kullanılır:

```
10 RANDOMIZE
20 PRINT INT(RND*10)+1

RUN
Random number seed? 5
6

RUN
Random number seed? 8
3
```

Bu program çalıştırılıp klavyeden 5 girildiğinde, sonuç her zaman 6 olacaktır. Benzer şekilde, bütün sayılar için belirli birer sonuç bulunacaktır. Burada klavyeden verilen sayılarla sonuç olarak elde edilen sayılar arasında doğrudan doğruya bir bağlantı yoktur. Girilen sayı, rasgele sayı üretimi için yalnızca bilgisayar tarafından bilinen bir yönteme göre bir başlangıç değeri belirlemek amacıyla kullanılır.

ÖRNEK: RANDOMIZE kullanarak, 5 ve 15 arasında 10 rasgele sayı üreten bir program:

```
10 RANDOMIZE
20 FOR T=1 TO 10
```

```
30 PRINT INT(RND*11)+5
40 NEXT I
```

Bu program, klavyeden girilen sayıya göre belirli bir rasgele sayı dizisi üretir. Aynı sayı birden çok kez girildiğinde, aynı sayı dizisi elde edilecektir.

B) RANDOMIZE komutu sayısal bir değerle yazılabilir:

Tek başına kullanıldığında programı durdurup klavyeden bir sayı girilmesini bekleyen bu komut, bir sayıyla birlikte kullanıldığında, bu beklemeye gerek kalmadan, bu sayıyı rasgele sayı üretiminin başlangıcını belirlemek için kullanır.

ÖRNEK:

```
10 RANDOMIZE 5
20 PRINT INT(RND*10)+1
```

Bu program her çalıştırıldığında çıkacak sonuç 6 olacaktır. Daha önce yapılan örnekte klavyeden girilen 5 sayısının yerine bu kez program içinde 5 sayısı sabit olarak RANDOMIZE komutuyla birlikte verilmiştir.

```
RUN
6
```

Görüldüğü gibi, program çalıştıktan sonra herhangi bir bekleme olmadan sonuç ekrana yazılmıştır.

Gerçek anlamda rasgele sayı üretimi:

Şu ana kadar yazılmış olan rasgele sayı üretim programlarında, tamamen rasgele sayı üretimi gerçekleşmemiştir. RANDOMIZE komutu kullanıldığında da, aynı yorumlayıcıya sahip bilgisayarlarda aynı sayılar verildiğinde, aynı rasgele sayılar elde edilecektir. Tam anla-

mıyla rasgele sayı üretimi, bazı BASIC yorumlayıcılarda bulunmayan zaman fonksiyonuyla yapılabilir. GW-BASIC'te bulunan TIMER fonksiyonu, bilgisayar açıldığı anda 0'a eşitlenen ve zamana bağlı olarak sürekli artan bir değer içermektedir. Bu fonksiyon RANDOMIZE komutuyla birlikte kullanıldığında, değişik zamanlarda değişik değerler vereceğinden, (ve bu değerlerin ne olacağı önceden bilinemeyeceğinden) gerçek rasgele sayı üretimi sağlanabilir:

```
10 RANDOMIZE TIMER
20 PRINT INT(RND*100)+1
```

1 ve 100 arasında rasgele bir sayı üretecek olan bu program, her çalıştırıldığında değişik bir sayı verecektir. Aynı yorumlayıcıyla da kullanılsa, farklı bilgisayarlarda farklı sonuçlar elde edilecektir.

NOT: GW-BASIC yorumlayıcısında bulunan TIMER fonksiyonu, Mallard BASIC'te yoktur.

RND fonksiyonunun bir parametreyle kullanımı:

RND fonksiyonu, şimdiye kadar olduğu gibi parametresiz kullanılırsa, 0 ve 1 arasında rasgele bir sayı üretir. RND ile kullanılabilecek parametre için 3 olasılık vardır:

1) Pozitif sayı : RND(1)
   Fonksiyonun parametresiz kullanımıyla aynıdır.

2) Sıfır : RND(0)
   En son üretilen rasgele sayı bir kez daha üretilir.

```
PRINT RND
0.21589462
PRINT RND(0)
0.21589462
```

3) Negatif sayı : RND(-1)
   Rasgele sayı üretim algoritması baştan başlar.

ÖRNEKLER:

1) Klavyeden girilen herhangi bir kelimeyi şifreleyerek ekrana yazan bir program: (Buradaki şifreleme yöntemi şu şekilde olacaktır: Kelimenin her harfi ASCII tablosuna göre kendisinden bir sonraki karaktere eşitlenecek.)

```
10 INPUT A$
20 FOR H=1 TO LEN(A$)
30 M$=MID$(A$,H,1)
40 S=ASC(M$)
50 B$=B$+CHR$(S+1)
60 NEXT H
70 PRINT "KELİMENİN ŞİFRELENMİŞ ŞEKLİ : ";B$
```

Bu programda kelimenin harfleri tek tek M$ değişkenine aktarılıyor; bu harfin ASCII kodu S değişkeninde saklanıyor. Daha sonra ASCII tablosuna göre bu harften sonra gelen karakter (CHR$(S+1)) B$ değişkenine eklenerek, bu değişkende şifre olarak kullanılacak karakter dizisi üretiliyor.

```
RUN
? PROGRAM
QSPHSBN
RUN
? MEHMET
NFGNFU
```

Klavyeden girilen kelimenin B$ değişkenindeki şifrelenmiş şekli, aynı yöntemi ters olarak kullanarak (CHR$(S-1)) tekrar eski haline çevrilebilir. Bu yöntemle uzun bir yazı, örneğin gizli kalması istenen bir mektup şifrelenebilir.

2) Her biri 6 harften oluşan, rasgele harflerle üretilmiş 10 kelimeyi ekrana yazan bir program: (Bu kelimeler, okunabilir olmaları amacıyla üç sessiz, üç sesli harften oluşacak şekilde üretilecektir.)

```
10 SESLI$="AEIİOÖUÜ"
20 SESSIZ$="BCÇDFGĞHJKLMNPRSTVYZ"
```

```
30 FOR K=1 TO 10
40 K$=""
50 FOK L=1 TO 3
60 A=INT(RND*8)+1
70 B=INT(RND*21)+1
80 K$=K$+MID$(SESSIZ$,B,1)+MID$(SESLI$,A,1)
90 NEXT L
100 PRINT K$;" ";
110 NEXT K
RUN
CUSAPÜ LEVIBİ COVOLO MÖCABA COMECA KECIJE
CONÖRO RÜYIPİ HOJÖMÖ RÖYESE
```

Sesli ve sessiz harfler birbirlerinden ayrılabilmeleri amacıyla iki ayrı değişkende saklanmıştır. 3 kez tekrarlanan L döngüsünde, birincisi 1 ve 8 (sesli harf adedi), ikincisi 1 ve 21 (sessiz harf adedi) arasında olmak üzere iki rasgele sayı üretilmektedir. Bu sayılara göre değişkenlerden alınan rasgele harfler K$ değişkeninde toplanmaktadır. Böylece döngünün her tekrarında iki harf üretilerek, 6 harfli 10 kelime yan yana ekrana yazılmaktadır.

SORULAR:

1) Klavyeden girilen bir sayının DATA satırından okunan bir sayıya bölümünü en yakın tamsayıya yuvarlayarak yazan bir program yapınız.

2) '28A' ve '3CX' karakter dizilerinin sayısal değerlerinin çarpımını yazan bir program yapınız.

3) Klavyeden girilen bir sayının 5 ile 21 arasında rasgele bir tamsayı ile çarpımını yazan bir program yapınız.

4) Klavyeden girilen kesirli bir sayının en yakın tamsayıya yuvarlanması için hangi sayıyla toplanması veya çıkartılması gerektiğini bulan bir program yapınız.

FONKSİYON TANIMLAMA (DEF FN kullanımı):

Programlamada bazı özel işlemleri kolayca yapabilmek için, komutlarla birlikte kullanılan BASIC diline özgü terimlere, fonksiyon denir. Bilgisayarın belleğine daha önceden yerleştirilmiş fonksiyonların dışında bir işlem yapan yeni bir fonksiyon tanımlanmak istendiğinde, DEF FN komutu kullanılır.

Özellikle program içinde çok sık kullanılacak olan işlemlerin, birer fonksiyon olarak tanımlanması ve gerektiğinde bir alt program gibi düşünülerek çağırılması, programlamada düzen ve sürat sağlayacaktır.

Genel kullanım:

DEF FN <fonksiyon adı> [(giriş değerleri)]=
(Çıkış değerini belirleyen ifade)

ÖRNEK: İki sayının (giriş değerleri) çarpımını çıkış değeri olarak veren CARPIM isimli fonksiyonun tanımlanması:

```
DEF FN CARPIM(A,B) = A * B
```

Bu komuttaki A ve B değişkenleri, fonksiyonun giriş değerleridir. Bu değerlere göre, ikisinin çarpımı olan çıkış değeri hesaplanacaktır.

Programcı tarafından tanımlanmış bir fonksiyonun çağırılması için FN terimi kullanılır.

Genel kullanım:

FN <fonksiyon adı> [(giriş değerleri)]

Daha önce tanımlanmış olan CARPIM fonksiyonunu çağırmak için aşağıdaki komut kullanılabilir:

```
PRINT FN CARPIM(4,5)
20
```

Burada elde edilen sonuç, giriş değerleri olan 4 ve 5 sayılarının çarpımı, yani tanımlanmış olan fonksiyondaki çıkış değeridir.

ÖRNEK: Net fiyatı klavyeden girilen bir malın katma değer vergisini KDV isimli bir fonksiyonla hesaplayarak malın KDV'sini ekrana yazan bir program:

```
10 DEF FN KDV(FIYAT) = FIYAT * 0.10
20 INPUT FIYAT
30 PRINT FN KDV(FIYAT)
```

Burada fonksiyon tanımlama satırı, programın en başında yer almaktadır. Bu şekilde yazılması zorunlu değildir. Fonksiyon, çağırıldığı satırdan önce istenilen herhangi bir yerde tanımlanabilir. Ancak, özellikle çok fazla sayıda fonksiyon tanımlanan uzun programlarda bütün fonksiyonların programın en başında tanımlanması, programlama açısından daha düzenli ve planlı çalışılmasını sağlayacak ve hata yapma olasılığını azaltacaktır. Ayrıca, fonksiyonların en başta tanımlanması, az da olsa, programın süratini arttıracaktır.

Aynı program, aşağıdaki şekilde de yapılabilir:

```
10 DEF FN KDV(FIYAT) = FIYAT * 0.10
20 INPUT A
30 PRINT FN KDV(A)
RUN
? 1000
100
```

ÖRNEK: Taban uzunluğu ve yüksekliği DATA satırından okunan bir üçgenin alanını tanıtılmış bir fonksiyon yardımıyla hesaplayarak ekrana yazan bir program:

Üçgenin alanı = Taban uzunluğu * Yükseklik / 2

```
10 DEF FN ALAN(TABAN,YUKSEKLIK)=
TABAN*YUKSEKLIK / 2
20 READ A,B
30 PRINT FN ALAN(A,B)
40 DATA 20,30

RUN
300
```

Bu programda, fonksiyon tanımlama satırında belirtilen TABAN ve YUKSEKLIK değişkenleri yerine, fonksiyon kullanılırken A ve B değişkenleri kullanılmıştır. Buradan da anlaşılacağı gibi, giriş değeri kabul eden fonksiyonlar için, tanımlama satırında giriş değerlerini ifade eden değişkenler yerine, fonksiyon çağırılırken, farklı değişkenler kullanılabilir. Yalnız, bu değişkenlerin, DEF FN satırındaki giriş değerlerine adet ve nitelik olarak uygun olması gerekir. Örneğin, giriş değeri olarak birincisi alfabetik, ikincisi sayısal olmak üzere iki değer kabul eden bir fonksiyon çağırılırken, kullanılması gereken değişkenlerden birincisinin alfabetik, ikincisinin sayısal olması zorunludur. Ayrıca, bu değişkenlerin adetlerinde de değişiklik yapılamaz.

ÖRNEK: Klavyeden girilen bir ismin ilk harfini bir fonksiyon yardımıyla bularak ekrana yazan bir program:

```
10 DEF FN ILK$(A$)=LEFT$(A$,1)
20 INPUT K$
30 PRINT FN ILK$(K$)


RUN
? YAVUZ
Y
```

Bu programda tanımlanan fonksiyon, alfabetik bir sonuç vermektedir. Bütün alfabetik fonksiyonlarda olduğu gibi, bu fonksiyonun adından sonra da '$' işareti kullanılmalıdır.

Özellikle kelimelerin ilk harflerinin çok kullanıldığı bir programda, her keresinde LEFT$ fonksiyonunu yazarak zaman kaybetmek yerine bu fonksiyonu kullanmak, programı daha kolay ve anlaşılır kılacaktır.

NOT: Çıkış değeri alfabetik olan fonksiyonlar tanımlanırken, giriş değerleri ne olursa olsun, fonksiyon adından sonra '$' işareti bulunmalıdır.

ÖRNEK: $Y=3X^2+4X-7$

Yukarıdaki formülü bir fonksiyon olarak tanımlayarak, X için klavyeden girilecek değere göre, Y'nin değerini hesaplayarak ekrana yazan bir program:

A)
```
10 DEF FN Y(X) = 3 * X ^ 2 + 4 * X - 7
20 INPUT K
30 PRINT FN Y(K)

RUN
? 6
125
```

B)
```
10 DEF FN Y = 3 * X ^ 2 + 4 * X - 7
20 INPUT X
30 PRINT FN Y

RUN
? 6
125
```

Aynı program, yukarıda iki ayrı şekilde çözülmüştür. Birinci çözümde, daha önceki örneklerde de olduğu gibi, fonksiyon tanımlanırken, giriş değeri (parametresi) olarak X kullanılmıştır. İkinci çözümde ise, bu parametre kullanılmadan aynı sonuç elde edilmiştir.

Fonksiyon tanımlanırken giriş parametresi kullanmak, fonksiyona esneklik kazandırmaktadır. Daha önce de anlatıldığı gibi, bu parametreler fonksiyon tanımında kullanılırsa, program içinde fonksiyona giriş değeri olarak verilecek değişkenler (A çözümünde K), tanımlama satırındakilerle aynı olmak zorunda değildirler. Yukarıdaki örnekteki B çözümünde ise, fonksiyonun tanımında giriş parametresi kullanılmadığı için, fonksiyonun giriş değeri olacak değişkenin (X), örnekte de görüldüğü gibi, tanımlama satırındakiyle aynı olması gerekmektedir.

ÖRNEKLER:

1) Beş emeklinin banka hesapları için DATA satırından okunacak anapara ve % 60 faiz oranına göre, iki yıl sonra her birinin bankadan çekebileceği para miktarını (Anapara + faiz) ekrana yazan bir program:

```
10 DEF FN FAIZ(ANAPARA,ORAN,SURE)=
ANAPARA*ORAN*SURE/100
20 O=60
30 S=2
40 FOR I=1 TO 5
50 READ A
60 PRINT A + FN FAIZ(A,O,S)
70 NEXT I
80 DATA 1500,2100,500,980,1600

RUN
3300
4620
1100
2156
3520
```

2) Klavyeden girilen bir kelimenin ilk ve son harflerini aralarında bir '-' işaretiyle ekrana yazan bir program:

```
10 DEF FN H$(A$)=LEFT$(A$,1)+"-"+RIGHT$(A$,1)
20 INPUT A$
30 PRINT FN H$(A$)

RUN
? ALEV
A-V
```

3) Bir mağazada, bazı mallara indirim yapılmaktadır. Klavyeden girilen kod 1 ise % 10 indirimli, kod 0 ise indirimsiz fiyatı ekrana yazan bir program: (İndirim miktarı, fiyat ve koda göre bir fonksiyon olarak tanımlanacaktır.)

```
10 DEF FN INDIR(FIYAT,KOD)=FIYAT*0.10*KOD
20 INPUT FIYAT,KOD
30 PRINT FIYAT - FN INDIR(FIYAT,KOD)

RUN
? 5000,1
4500            (indirimli)

RUN
? 5000,0
5000            (indirimsiz)
```

Bu örnekte, tanımlanan fonksiyonun değeri klavyeden girilen koda göre değişmektedir. Kod 1 ise indirim miktarı, fiyatın % 10'u olarak hesaplanacaktır. Kod 0 ise, 10 numaralı satırda görüldüğü gibi, fiyatın % 10'u 0 ile çarpılacak, dolayısıyla, indirim miktarı (INDIR) 0 olacaktır.

LINE INPUT komutu:

LINE INPUT komutu, hemen hemen INPUT komutuyla aynıdır. Aralarındaki tek fark, LINE INPUT komutunun bilgi girişinde virgülü de girilen alfabetik bilginin içinde kabul etmesidir. (LINE INPUT komutuyla sayısal değişken kullanılamaz.)

ÖRNEK: Klavyeden ayrı ayrı girilen isim ve adresi alt alta ekrana yazan bir program:

```
10 INPUT I$
20 INPUT ADR$
30 PRINT I$
40 PRINT ADR$

RUN
? HAKAN
? KADIKÖY
HAKAN
KADIKÖY

RUN
? AYLA
? İLİ, İSTANBUL
?Redo from start
? İLİ-İSTANBUL
AYLA
İLİ-İSTANBUL
```

Yukarıda görüldüğü gibi, INPUT komutuyla bilgi girişi yapıldığında, virgülün kullanım amacı, farklı bilgileri birbirinden ayırmaktır. (Örnek: INPUT A$,B$,C$) Yukarıdaki adres örneğinde olduğu gibi, eğer bir tek bilgi içinde virgül karakteri bulunabilecekse, INPUT komutunu kullanmak olanaksızdır. Böyle durumlarda, LINE INPUT komutu kullanılır. Aynı örneğin LINE INPUT komutuyla çözümü:

```
10 LINE INPUT I$
20 LINE INPUT ADR$
30 PRINT I$
40 PRINT ADR$
RUN
? HAKAN
? KADIKÖY
HAKAN
KADIKÖY
RUN
? AYLA
? İLİ, İSTANBUL
AYLA
İLİ, İSTANBUL
```

Görüldüğü gibi, LINE INPUT komutuyla, virgüllü bilgi girişi sağlanmıştır.

NOT: LINE INPUT komutuyla yalnızca bir tek alfabetik değişken kullanılabilir. Birden fazla giriş yapılmak istendiğinde, LINE INPUT komutları ayrı program satırlarında yazılmalıdır. LINE INPUT, virgül dahil bütün karakterleri tek bir giriş bilgisi olarak kabul ettiğinden, farklı bilgiler aynı satır üzerinde yazılmak istenirse, bunları ayıracak belirli bir karakter yoktur. Bu nedenle LINE INPUT komutuyla birden fazla değişken kullanılamaz.

Profesyonel kullanımda, programların bilgi girişleri, genellikle programlamayı bilmeyen personel tarafından yapıldığından bu tür programlarda LINE INPUT kullanmak, bilgi giriş işleminin hatasızlığı açısından daha garantili olacaktır. Hatta, sayısal girişler için dahi, LINE INPUT komutuyla alfabetik bir değişken kullanarak, daha sonra VAL fonksiyonuyla bu girişi sayıya çevirmek programın daha da kullanışlı olmasını sağlayacaktır.

SIRALAMA YÖNTEMLERİ

Karışık olarak bir arada bulunan sayısal veya alfabetik bilgiler içinden istenilen herhangi birine kolayca ulaşılabilmesi için, bu bilgilerin belirli bir mantığa göre sıralanmış olmaları gerekir. Örneğin bir telefon rehberindeki isimler alfabetik sırada olmazsa, bu isimler arasından istenilen birini bulabilmek hemen hemen olanaksızdır.

Sıralama (sort) yapabilmek için çeşitli yöntemler vardır. Sıralanacak bilgilerin adedine, dizilişine veya türüne göre, bu yöntemlerden herhangi biri kullanılabilir. Bu yöntemlerden yalnızca üçü programcılığa yeni başlayanlar için daha kolay anlaşılabilir olmaları açısından tercih edilmiş ve anlatılmıştır.

Birinci yöntem:

Bu yöntemde, dizi elemanlarının her biri kendisinden bir sonraki ile karşılaştırılarak, doğru sırada olmayan çiftlerin yerleri değiştirilir. Bu yönteme bubble sort (kabarcık sıralama) denir. Bu yer değiştirme işleminin daha kolay anlaşılabilmesi için aşağıdaki örnek incelenebilir:

ÖRNEK: Klavyeden girilen A ve B değişkenlerindeki sayıların yerlerini değiştiren (A'daki sayıyı B'ye, B'dekini A'ya aktaran) ve değişkenleri girildikleri sırada ekrana yazan bir program:

```
10 INPUT A,B
20 S=A
```

```
30 A=B
40 B=S
50 PRINT A,B
```

Bu programda iki değişkenin değerlerini değiştirebilmek için üçüncü bir değişken kullanılmıştır. Bunun nedeni, birinci değişkenin değeri ikinciye aktarıldığında, ikincinin değerinin kaybolmasıdır. Bu değer, kaybolmaması için üçüncü değişkende saklanmıştır. (Günlük yaşantıdan bir örnek verilecek olursa, iki bardağın içinde bulunan su ve sütün yerlerini değiştirmek, yani suyu süt bardağına, sütü su bardağına aktarmak düşünülebilir. Bunu yapabilmek için üçüncü bir bardağa ihtiyaç vardır.)

20 ve 40 numaralı satırlar arasındaki bölüm, bir GW-BASIC komutu olan SWAP kullanılarak kısaca yapılabilir:

```
SWAP A,B
```

ÖRNEK: Data satırından okunan 5 sayıyı ekrana listeleyen, daha sonra küçükten büyüğe sıralayarak sayıları tekrar yazan bir program:

```
10 DIM A(5)
20 FOR I=1 TO 5
30 READ A(I)
40 PRINT A(I)
50 NEXT I
60 DATA 8,5,3,6,4
70 FOR K=1 TO 4
80 FOR L=1 TO 4
90 IF A(L) > A(L+1) THEN S = A(L):A(L) = A(L+1) :
A(L+1) = S
100 NEXT L
110 NEXT K
120 FOR I=1 TO 5
130 PRINT A(I)
140 NEXT I
```

DATA satırından okunan sayıların sıralanabilmesi için, diziye alınması zorunludur. (Dizi kullanmadan sıralama yapmak çok zor olacaktır.)

Aşağıdaki tablo, dizi elemanlarının yukarıdaki programa göre karşılaştırılıp yer değiştirerek aldıkları değerleri göstermektedir. 70 ve 110 numaralı satırlar arasındaki döngü ile sıralanan dizi, 120 ve 140 numaralı satırlar arasındaki döngü ile ekrana yazılacaktır.

A(1)	A(2)	A(3)	A(4)	A(5)	
8 <-->	5	3	6	4	A(1) > A(2) ? Evet; değiştir.
5	8 <-->	3	6	4	A(2) > A(3) ? Evet; değiştir.
5	3	8 <-->	6	4	A(3) > A(4) ? Evet; değiştir.
5	3	6	8 <-->	4	A(4) > A(5) ? Evet; değiştir.
5 <-->	3	6	4	8	A(1) > A(2) ? Evet; değiştir.
3	5 <-->	6	4	8	A(2) > A(3) ? Hayır.
3	5	6 <-->	4	8	A(3) > A(4) ? Evet; değiştir.
3	5	4	6 <-->	8	A(4) > A(5) ? Hayır.
3 <-->	5	4	6	8	A(1) > A(2) ? Hayır.
3	5 <-->	4	6	8	A(2) > A(3) ? Evet; değiştir.
3	4	5 <-->	6	8	A(3) > A(4) ? Hayır.
3	4	5	6 <-->	8	A(4) > A(5) ? Hayır.
3 <-->	4	5	6	8	A(1) > A(2) ? Hayır.
3	4 <-->	5	6	8	A(2) > A(3) ? Hayır.
3	4	5 <-->	6	8	A(3) > A(4) ? Hayır.
3	4	5	6 <-->	8	A(4) > A(5) ? Hayır.

İkinci yöntem:

Bu yönemde, dizinin her elemanı kendisinden daha sonraki bütün elemanlarla karşılaştırılarak yer değiştirme yapılır.

ÖRNEK: Yukarıdaki sayıların ikinci yönteme göre sıralayan ve ekrana listeleyen bir program:

```
10 DIM A(5)20 FOR I=1 TO 5
30 READ A(I)
40 PRINT A(I)
50 NEXT I
60 DATA 8,5,3,6,4
70 FOR K=1 TO 4
80 FOR L=K+1 TO 5
90 IF A(K) > A(L) THEN S = A(K) : A(K) = A(L) : A(L) = S
100 NEXT L,K
```

```
110 FOR I = 1 TO 5
120 PRINT A(I)
130 NEXT I
```

A(1)	A(2)	A(3)	A(4)	A(5)	
8	5	3	6	4	A(1) > A(2) ? Evet; değiştir.
5	8	3	6	4	A(1) > A(3) ? Evet; değiştir.
3	8	5	6	4	A(1) > A(4) ? Hayır.
3	8	5	6	4	A(1) > A(5) ? Hayır.
3	8	5	6	4	A(2) > A(3) ? Evet; değiştir.
3	5	8	6	4	A(2) > A(4) ? Hayır.
3	5	8	6	4	A(2) > A(5) ? Evet; değiştir.
3	4	8	6	5	A(3) > A(4) ? Evet; değiştir.
3	4	6	8	5	A(3) > A(5) ? Evet; değiştir.
3	4	5	8	6	A(4) > A(5) ? Evet; değiştir.
3	4	5	6	8	

Üçüncü yöntem:

Bu yöntemde, dizideki en küçük elemanın bulunup en başa yerleştirilmesi ile sıralamaya başlanır. Daha sonra, ikinciden başlanarak en küçük eleman bulunur ve sıralama bu şekilde devam eder.

ÖRNEK: Daha önceki örneklerdeki sayıları üçüncü yönteme göre sıralayan ve ekrana listeleyen bir program:

```
10 DIM A(5)
20 FOR I=1 TO 5
30 READ A(I)
40 PRINT A(I)
50 NEXT I
60 DATA 8,5,3,6,4
70 FOR K = 1 TO 4
80 EK = A(K) : S = K
90 FOR L = K+1 TO 5
100 IF A(L) < EK THEN EK = A(L) : S = L
110 NEXT L
120 Y = A(K)
130 A(K) = A(S)
```

```
140 A(S) = Y
150 NEXT K
160 FOR I = 1 TO 5
170 PRINT A(I)
180 NEXT I
```

Aşağıdaki tablo, dizi elemanlarının sıralama süresince aldıkları değerleri göstermektedir:

A(1)	A(2)	A(3)	A(4)	A(5)	
(8	5	3	6	4)	En küçük 3; 3 ve 8'in yeri değişecek
3	(5	8	6	4)	En küçük 4; 4 ve 5'in yeri değişecek
3	4	(8	6	5)	En küçük 5; 5 ve 8'in yeri değişecek
3	4	5	(6	8)	En küçük 6; yerinde kalacak.

NOT: Tabloda parantez içine alınmış bölümler, en küçük elemanın arandığı bölümlerdir. Parantez içindeki en küçük eleman, bölümdeki ilk elemanla yer değiştirmektedir.

Alfabetik bilgilerle sıralama işlemleri:

Sıralama işlemleri, sayısal bilgilerle yapılabildiği gibi, alfabetik bilgilerle de yapılabilir. Alfabetik bilgilerin sıralanması, bu bilgilerin ASCII sıra numaralarına göre yapılır. Alfabetik bilgilerin büyüklük küçüklük karşılaştırması, tıpkı sayısal bilgilerde olduğu gibidir; fakat, alfabetik bilgilerde sayısal bir değer aranmaz. Bir alfabetik bilginin bir diğerinden daha küçük olması, bu bilginin ASCII tablosunda diğerinden daha önce yer alması demektir.

ÖRNEK: DATA satırından okunan 8 harfi sıralayan ve ekrana listeleyen bir program:

```
10 DIM H$(8)
20 FOR I = 1 TO 8
30 READ H$(I)
40 PRINT H$(I);" ";
50 NEXT I
60 PRINT
70 DATA C,D,G,E,N,F,U,A
80 FOR A = 1 TO 7
```

```
90 EK$ = H$(A) : S = A
100 FOR B = A+1 TO 8
110 IF EK$ > H$(B) THEN EK$ = H$(B) : S = B
120 NEXT B
130 D$ = H$(S)
140 H$(S) = H$(A)
150 H$(A) = D$
160 NEXT A
170 FOR T = 1 TO 8
180 PRINT H$(T);" ";
190 NEXT T

RUN
C D G E N F U A
A C D E F G N U
```

Bu programda, verilmiş olanlar arasında en hızlı olan üçüncü sıralama yöntemi kullanılmıştır. Daha önceki örneklerde en küçük sayısal değeri bulan bu yöntem, bu kez en küçük alfabetik bilgiyi bulmaktadır. Burada en küçük alfabetik bilgi, ASCII tablosunda en küçük sıra numarasına sahip olan bilgi anlamına gelmektedir.

Alfabetik sıralama, yalnızca tek karakterden oluşan bilgiler için değil, kelimeler için de uygulanabilir. Yukarıdaki program, kelimeleri de aynı mantıkla alfabetik sıraya sokacaktır. Bir kelimenin, bir diğerinden daha küçük olması, harf sayısının daha az olması değil, kelimenin alfabetik sıralamaya göre daha önce yer almasıdır (sözlükteki kelimelerin sıralanışı). Örneğin, 'AHMET' kelimesi, 'CAN' kelimesinden daha küçüktür.

NOT: Kelimeler için geçerli olan bu sıralama, yalnızca İngiliz alfabesindeki harfler için geçerlidir. Daha önce de belirtildiği gibi, İngiliz alfabesinde bulunmayan harfler (Ç, Ğ, ı, İ, Ö, Ş, Ü) bilgisayar tarafından harf olarak kabul edilmediklerinden (ASCII tablosundaki yerleri sıraya uymadığından), bu karakterleri içeren kelimelerin sıralanması, Türkçeye uymayacaktır. Bu kelimelerin sıralanması istendiğinde, sıraya uymayan karakterlerin yerine İngiliz alfabesindeki benzerleri kullanılabilir. (Ç yerine C, Ğ yerine G gibi)

ÖRNEK: DATA satırından okunan 10 kelimeyi sıralayarak ekrana listeleyen bir program:

```
10 DIM K$(10)
20 FOR L = 1 TO 10
30 READ K$(L)
40 PRINT K$(L)
50 NEXT L
60 DATA NEJAT,ASUDE,ALEV,ZEYNEP,MURAT,OKAN,KAYA,
LEMAN,EMEL,FARUK
70 FOR I = 1 TO 9
80 EK$ = K$(I) : S = I
90 FOR J = I+1 TO 10
100 IF EK$ > K$(J) THEN EK$ = K$(J) : S = J
110 NEXT J
120 D$ = K$(S)
130 K$(S) = K$(I)
140 K$(I) = D$
150 NEXT I
160 PRINT "----------"
170 FOR C = 1 TO 10
180 PRINT K$(C)
190 NEXT C

RUN
NEJAT
ASUDE
ALEV
ZEYNEP
MURAT
OKAN
KAYA
LEMAN
EMEL
FARUK

ALEV
ASUDE
EMEL
FARUK
```

KAYA
LEMAN
MURAT
NEJAT
OKAN
ZEYNEP

Sıralanan bir diziye paralel olarak ikinci bir dizinin sıralanması:

ÖRNEK:

İSİM	NOT
KAYA	3
ALEV	7
MURAT	8
NEJAT	9
OKAN	6
ZEYNEP	5
ASUDE	2
LEMAN	7
FARUK	8
EMEL	9

Yukarıdaki listeyi, isimler alfabetik sıraya girecek şekilde sıralayan, bir program: (İsimler sıralanırken, notlar da isimlerin karşısına gelecek şekilde yer değiştirecektir.)

```
10 DIM A$(10),N(10)
20 FOR M = 1 TO 10
30 READ A$(M),N(M)
40 NEXT M
50 DATA KAYA,3,ALEV,7,MURAT.8.NEJAT.9.OKAN.6,
ZEYNEP,5,ASUDE,2,LEMAN,7,FARUK,8,EMEL,9
60 FOR G = 1 TO 9
70 E$ = A$(G) : E = G
80 FOR H = G+1 TO 10
90 IF A$(H) < E$ THEN E$ = A$(H) : E = H
100 NEXT H
110 D$ = A$(G)
120 A$(G) = A$(E)
130 A$(E) = D$
140 D = N(G)
```

```
150 N(G) = N(E)
160 N(E) = D
170 NEXT G
180 PRINT "İSİM","NOT"
190 PRINT "-------","---"
200 FOR T = 1 TO 10
210 PRINT A$(T),N(T)
220 NEXT T

RUN
```

İSİM	NOT
ALEV	7
ASUDE	2
EMEL	9
FARUK	8
KAYA	3
LEMAN	7
MURAT	8
NEJAT	9
OKAN	6
ZEYNEP	5

Aynı bilgileri, notlar büyükten küçüğe doğru sıralanacak şekilde sıraya dizen ve ekrana listeleyen bir program:

```
10 DIM A$(10),N(10)
20 FOR M = 1 TO 10
30 READ A$(M),N(M)
40 NEXT M
50 DATA KAYA,3,ALEV,7,MURAT,8,NEJAT,9,OKAN,6,
ZEYNEP,5,ASUDE,2,LEMAN,7,FAR      UK,8,EMEL,9
60 FOR G = 1 TO 9
70 E = N(G) : S = G
80 FOR H = G+1 TO 10
90 IF  N(H) > E  THEN  E = N(H) : S = H
100 NEXT H
110 D$ = A$(G)
120 A$(G) = A$(S)
```

```
130 A$(S) = D$
140 D = N(G)
150 N(G) = N(S)
160 N(S) = D
170 NEXT G
180 PRINT "İSİM","NOT"
190 PRINT "-------","---"
200 FOR T = 1 TO 10
210 PRINT A$(T),N(T)
220 NEXT T
```

Bu programda, notlar büyükten küçüğe doğru sıralanırken, aynı zamanda isimler de notlara paralel olarak yer değiştirmektedir. 90 numaralı satırda, daha önce küçükten büyüğe sıralama yapmak için kullanılan küçük (<) işareti yerine, büyükten küçüğe doğru sıralamak için büyük (>) işareti kullanılmıştır.

SORULAR:

1) 1 ve 100 arasında rasgele tamsayılardan oluşan 100 elemanlı sayısal bir diziyi küçükten büyüğe sıralayarak ekrana listeleyen bir program yapınız.

2) Klavyeden girilen herhangi bir kelimenin harflerini alfabetik sıraya (İngiliz alfabesine göre) sokan ve bu harfleri yan yana ekrana yazan bir program yapınız.

ON ERROR GOTO:

Bu komut, program çalışırken programlamayla ilgili ortaya çıkabilecek hataların bulunabilmesi ve bu hatalarla karşılaşıldığında programın istenildiği şekilde yönlendirilebilmesi için kullanılır.

Program içinde, BASIC kurallarına uymayan ifadeler bulunuyorsa, program çalıştırılıp bu yanlışlığın bulunduğu yere kadar ilerledikten sonra, normal olarak bir hata mesajı vererek bilgisayar programı durdurur. Bu hata mesajından yararlanılarak gerekli düzeltme yapılır ve program tekrar çalıştırılır.

Programcı tarafından kullanılmayan, yani herhangi bir hata mesajıyla karşılaşıldığında düzeltilme olanağı bulunmayan programlarda, böyle bir durum sakıncalıdır. Bunu önlemek için, ON ERROR GOTO komutu kullanılır. Bu komutun yararı, bilgisayarın bir yanlışlık nedeniyle programdan çıkmasını engellemektir.

ON ERROR GOTO komutu, genellikle programın başına yazılır; komuttan sonra, yanlışlık çıktığında program akışının aktarılması istenen satır numarası verilir.

ÖRNEK: ON ERROR GOTO 150

(Programda bir hata çıkarsa, bilgisayar 150 numaralı satıra gidecektir.)

Yanlışlık durumunda programın aktarıldığı satırdan başlayan bölümde, bu yanlışlıkla ilgili çeşitli işlemler yapılabilir:

a) Hata önemsenmeyip programa devam edilebilir.
(Birçok durumda sakıncalıdır.)

b) Hatanın varlığı kullanıcıya bildirilip son işlemi doğru olarak baştan yapması istenebilir. (Kullanıcının neden olduğu hatalar için)

c) Hatanın türü ve bulunduğu yer (satır numarası) belirtilerek en kısa zamanda programcıyla temas kurulması önerilebilir. (Yanlış programlama hataları için)

ERR ve ERL fonksiyonları:

ERR (Error) fonksiyonu, bir hatayla karşılaşıldığı zaman bu hatanın kod numarasını verir. Böylece hata mesajı yazılmadan da hangi hatanın yapıldığı anlaşılabilir. Kullanılan BASIC yorumlayıcının hata kodlarından yararlanarak hata düzeltilebilir.

ERL (Error Line) fonksiyonu, hatanın bulunduğu program satırının numarasını verir. Bu fonksiyondan yararlanarak, hatanın tam olarak nerede olduğu belirlenebilir.

RESUME komutu:

Programda hatayla karşılaşıldıktan sonra, ON ERROR GOTO komutundan sonra yazılmış olan satıra atlayarak program çalışmaya devam eder. Burada, hatanın düzeltilmesiyle ilgili bir bölüm yer alır. Bundan sonra, programı tekrar normal akışına döndürebilmek için RESUME komutu kullanılır. Bir bakıma, ON ERROR GOTO ve RESUME komutlarının ilişkisi, GOSUB - RETURN ilişkisine benzer.

RESUME komutunun üç tür kullanımı vardır:

a) RESUME
   Program, hatanın olduğu satıra döner.

b) RESUME <satır numarası>
   Program, numarası verilen satıra döner.

c) RESUME NEXT
   Program, hatayla karşılaşılandan bir sonraki satıra döner.

ÖRNEKLER:

1) Klavyeden girilen iki sayıdan birincisini ikincisine bölen bir program:

```
10 INPUT A,B
20 PRINT A/B
```

Kullanıcı B sayısı için 0 verirse, A/B işlemi geçersiz olacak ve bir hata mesajına yol açacaktır. Bunu engellemek için program şu şekilde geliştirilebilir:

```
10 ON ERROR GOTO 100
20 INPUT A,B
30 PRINT A/B
50 END
100 PRINT "HATALI GİRİŞ!"
110 RESUME 20
```

2) Aşağıda, hata çıkma olasılığı olan uzun bir program verilmiştir. Burada hata çıktığında programın istenmeyen bir biçimde durmasını engellemek için ON ERROR GOTO komutu kullanılmıştır:

```
10 ON ERROR GOTO 60000
20 ...
   ...
   ...
   ...
   ...
60000 CLS
60010 PRINT "PROGRAMDA HATA ÇIKMIŞTIR."
60020 PRINT
60030 PRINT "LÜTFEN AŞAĞIDAKİ BİLGİLERİ NOT EDEREK
EN KISA ZAMANDA"
60040 PRINT " PROGRAMCIYA BİLDİRİNİZ."
60050 PRINT
60040 PRINT  "HATA KODU    :";ERR
60070 PRINT "HATALI SATIR :";ERL
60080 PRINT
60090 PRINT
60100 PRINT "PROGRAMCI        : ALİ KOÇ"
60110 PRINT
60120 PRINT "TELEFON NUMARASI : 422 56 11 (Ev)"
60130 PRINT "                   991 23 54 (İş)"
60140 PRINT
60150 PRINT
60160 PRINT "PROGRAMIN KALDIĞI YERDEN DEVAM
ETMESİ İÇİN (1)"
60170 PRINT "PROGRAMIN BAŞTAN BAŞLAMASI
İÇİN          (2)"
```

```
60180 TUS$=INKEY$
60190 IF TUS$="1" THEN RESUME NEXT
60200 IF TUS$="2" THEN RESUME 10
60210 GOTO 60180
```

SORULAR:

1) Daktilografi derslerinde öğrencilerin dereceleri (net kelime sayıları), 5 dakikalık sürede yaptıkları vuruş ve hata sayılarına göre hesaplanmaktadır. Vuruş adedi 100'e bölündükten sonra 4'le çarpılmakta, bundan da hata adedinin 2 katı çıkarılmaktadır. Buna göre bir fonksiyon tanımlayarak, klavyeden girilen vuruş ve hata adetlerine göre dereceyi hesaplayan ve ekrana yazan bir program yapınız.

2) $\frac{-B \pm \sqrt{B^2-4AC}}{2A}$ Yandaki formül, ikinci dereceden bir denklemin köklerini bulmak için kullanılır. A, B ve C değişkenlerinin denklemin katsayıları olduğu düşünülürse, bu değerler klavyeden girildiğinde formülün sonucunu, yani denklemin köklerini ekrana yazan bir program yapınız.

NOT:

a) $\pm$ işareti, denklemin iki kökü olduğunu göstermektedir. Yani, X1 ve X2 için iki ayrı fonksiyon tanımlanacak, birincide + işareti, ikincide ise– işareti kullanılacaktır.

b) B^2'nin 4AC'den küçük olduğu durumlarda $\sqrt{B^2-4AC}$ ifadesi tanımsız olacaktır. Aynı şekilde, A'nın değerinin 0 olduğu durumlarda bölme işlemi geçersizdir. Programda hata çıkmaması için bu önceden kontrol edilmelidir. (ON ERROR GOTO kullanılabilir.)

ON...GOTO ve ON...GOSUB KOMUTLARI:

```
40 IF A=1 THEN GOTO 300
50 IF A=2 THEN GOTO 420
60 IF A=3 THEN GOTO 480
70 IF A=4 THEN GOTO 210
80 IF A=5 THEN GOTO 290
90 IF A=6 THEN GOTO 560
100 IF A=7 THEN GOTO 780
110 IF A=8 THEN GOTO 400
```

Yukarıdaki satırlar yerine, aşağıdaki tek satır yazılabilir:

```
10 ON A GOTO 300,420,480,210,290,560,780,400
```

ON...GOTO, programın normal akışı sırasında, işlem sırasının değişik satırlara yönlendirilmesini sağlar. Program akışı, ON ve GO-TO arasına yazılacak sayısal değişkenin alacağı değerlere göre GO-TO komutundan sonra yazılmış satır numaralarından herhangi birine geçerek devam eder.

Genel yazılım:

ON <sayısal ifade> GOTO <Satır numaraları>

Sayısal ifadenin değeri 1 ise, program akışı GOTO komutundan sonra numarası yazılan birinci satırdan devam eder; değer 2 ise ikinci, 3 ise üçüncü satırlara aktarılır.

Yukarıdaki örnekteki ON A GOTO 300, 420, 480, 210, 290, 560, 780, 400 satırı, şu şekilde açıklanabilir:

A'nın değeri	Gidilecek satır
1	300
2	420
3	480
4	210
5	290
6	560
7	780
8	400

Görüldüğü gibi, bu şekilde bir dallanma gerektiğinde, birçok IF...THEN cümlesi yerine bir tek ON...GOTO satırı kullanmak çok daha pratiktir.

Birden çok görevi olan ve kullanıcının seçeneğine göre bu görevlerden herhangi birini uygulayan programlarda ON...GOTO veya ON...GOSUB kullanmak büyük kolaylık sağlar. Bu tür programlara menülü programlar denilmektedir.

NOT: ON ve GOTO arasındaki sayısal değer, GOTO komutundan sonra yazılan satır numaraları adedinden büyükse veya 1'den küçükse, program akışı, ON...GOTO satırından bir sonraki satırdan devam edecektir. Örneğin, yukarıdaki örnekte A'nın değeri 9 olursa, program, GOTO'dan sonra numaraları yazılmış olan satırlardan hiçbirine gitmeyecek, bir sonraki satırdan devam edecektir.

ON...GOSUB komutunun kullanımı da ON...GOTO ile aynıdır. Aradaki tek fark, GOTO ve GOSUB arasındaki farktır. Yani, ON...GOSUB ile dallanma yapıldığında, RETURN komutuyla alt programlardan ON...GOSUB satırından bir sonraki satıra geri dönüş sağlanır. Aynı işlemin ON...GOTO ile yapılması istenirse, RETURN yerine uygun bir GOTO komutu kullanılır.

ÖRNEKLER:

1) Klavyeden girilen sayı 1 ise 'Birinci', 2 ise 'İkinci', 3 ise 'Üçüncü', 4 ise 'Dördüncü' yazıp duran, hiçbiri değilse 'Hatalı giriş' yazıp başa dönen bir program:

```
10 INPUT S
20 IF S>4 OR S<1 THEN PRINT "HATALI GİRİŞ":GOTO 10
30 ON S GOTO 50,100,150,200
50 PRINT "Birinci"
60 END
100 PRINT "İkinci"
110 END
150 PRINT "Üçüncü"
160 END
200 PRINT "Dördüncü"
210 END
```

Bu programda, S değişkeninin aldığı değere göre, dört PRINT komutundan herhangi biri uygulanacaktır. 1 ve 4 arasında olmayan bir sayı girildiğinde, program 'Hatalı giriş' yazıp başa dönecektir.

2) Klavyeden girilen sayı 1 ise 8 isim okuyup listeleyen, 2 ise 8'den 12'ye kadar sayıların karelerini yazan, 3 ise klavyeden girilen 5 sayının kareköklerini yazan ve her seçenekten sonra başa dönen bir program:

A)
```
10 INPUT A
20 ON A GOTO 50,150,200
50 RESTORE 60
60 DATA AYHAN,RAZİYE,NURHAN,ALEV,YURDAGÜL,
SEVGİ,BELMA,ZEYNEP
70 FOR I=1 TO 8
80 READ A$
90 PRINT A$
100 NEXT I
110 GOTO 10
150 FOR S=8 TO 12
160 PRINT S ^ 2
170 NEXT I
```

```
180 GOTO 10
200 FOR I=1 TO 5
210 INPUT N
220 PRINT SQR(N)
230 NEXT I
240 GOTO 10
```

B)
```
10 INPUT A
20 ON A GOSUB 50,150,200
30 GOTO 10
50 RESTORE 60
60 DATA AYHAN,RAZİYE,NURHAN,ALEV,YURDAGÜL,
SEVGİ,BELMA,ZEYNEP
70 FOR I=1 TO 8
80 READ A$
90 PRINT A$
100 NEXT I
110 RETURN
150 FOR S=8 TO 12
160 PRINT S^2
170 NEXT I
180 RETURN
200 FOR I=1 TO 5
210 INPUT N
220 PRINT SQR(N)
230 NEXT I
240 RETURN
```

A çözümünde, ON...GOTO kullanılmıştır. Bu nedenle, seçeneklerden geri dönüş, GOTO komutuyla sağlanmıştır. ON...GOSUB kullanılan B çözümünde ise başa dönüş için RETURN komutundan yararlanılmıştır. RETURN komutu programı GOSUB satırından bir sonraki satıra döndüreceğinden, 30 numaralı satırda bir GOTO yazılarak program en başa döndürülmüştür.

İki programda da yer alan RESTORE komutu, aynı DATA satırının yeniden okunabilmesini sağlamaktadır. 50 numaralı satırdan başlayan isim listeleme bölümü bir kereden fazla seçilerek çalıştırılabileceğinden, DATA satırının her keresinde yeniden okunabilmesi için RESTORE komutu kullanılmıştır.

3) ANA MENÜ
1- ERKEK GİYİM EŞYALARI
2- BAYAN GİYİM EŞYALARI
3- PROGRAMDAN ÇIKIŞ

SEÇİMİNİZ ?

DATA satırında isimleri ve kodları verilen (Kod 1 ise erkek, 2 ise bayan) giyim eşyalarını yukarıdaki menü yardımıyla listeleyen bir program:

```
10 CLS
20 PRINT "ANA MENÜ"
30 PRINT "-------------"
40 PRINT "1- ERKEK GİYİM EŞYALARI"
50 PRINT "2- BAYAN GİYİM EYALARI"
60 PRINT "3- PROGRAMDAN ÇIKIŞ"
70 INPUT "SEÇİMİNİZ ";S
80 IF S>3 OR S<1 THEN 10
90 ON S GOSUB 200,300,400
100 GOTO 10
200 RESTORE
210 CLS
220 FOR I=1 TO 10
230 READ G$,K
240 IF K=1 THEN PRINT G$,K
250 NEXT I
260 FOR B=1 TO 5000
270 NEXT B
280 RETURN
290 REM ----------------
300 RESTORE
310 CLS
320 FOR I=1 TO 10
330 READ G$,K
340 IF K=2 THEN PRINT G$,K
350 NEXT I
360 FOR B=1 TO 5000
370 NEXT B
```

```
380 RETURN
390 REM ----------------
400 PRINT  "PROGRAM BİTMİŞTİR."
410 END
420 DATA GÖMLEK,1,ETEK,2,ÇORAP,1,ÇORAP,2,BLUZ,2,
ŞAPKA,1,PANTOLON,1
430 DATA CEKET,1,ELBİSE,2,AYAKKABI,2
```

Bu programda, 260 ve 360 numaralı satırlardaki FOR...NEXT döngüleri, listelerin ekranda okunabilecek kadar bir süre kalmasını sağlamak için kullanılmıştır. Eğer bu döngüler kullanılmazsa, listeler ekranda belirdikten hemen sonra program başa dönecek ve ekran silinerek ANA MENÜ görülecektir. 290 ve 390 numaralı REM satırları, bölümleri birbirinden ayırarak programın izlenebilmesini kolaylaştırmak amacıyla koyulmuştur.

```
4) ANA MENÜ
----------------
    1- ÇORBALAR
    2- ETLER
    3- SEBZELER
    4- TATLILAR
    5- MEYVELER
    6- İÇECEKLER
    7- ÇIKIŞ

SEÇİMİNİZ ?
```

Yukarıdaki menü, bir lokantadaki yemek gruplarını göstermektedir. Buna göre, her seçeneğe ait listeleri ekrana çıkaran bir program:

```
10 CLS
20 PRINT "ANA MENÜ"
30 PRINT "----------------"
40 PRINT "1- ÇORBALAR"
50 PRINT "2- ETLER"
60 PRINT "3- SEBZELER"
70 PRINT "4- TATLILAR"
80 PRINT "5- MEYVELER"
90 PRINT "6- İÇECEKLER"
100 PRINT "7- ÇIKIŞ"
```

```
110 PRINT
120 INPUT "SEÇİMİNİZ ";S
130 IF S<1 OR S>7 THEN 10
140 ON S GOTO 200,300,400,500,600,700,800
200 CLS
210 RESTORE 220
220 DATA ŞAFAK, İŞKEMBE, MERCİMEK, BORÇ,
ŞEHRİYE,YOĞURT
230 FOR I=1 TO 6
240 READ C$
250 PRINT C$;" ÇORBASI"
260 NEXT I
270 FOR I=1 TO 5000
280 NEXT I
290 GOTO 10
300 CLS
310 RESTORE 320
320 DATA KÖFTE,İ,TAS KABABI,ROSTO,KAVURMA,
PİRZOLA,BİFTEK
330 FOR I=1 TO 7
340 READ E$
350 PRINT E$
360 NEXT I
370 FOR I=1 TO 5000
380 NEXT I
390 GOTO 10
400 CLS
410 RESTORE 420
420 DATA FASULYE,BEZELYE,KARNIBAHAR,ENGİNAR,
ISPANAK,PIRASA,LAHANA
430 FOR I=1 TO 7
440 READ S$
450 PRINT S$
460 NEXT I
470 FOR I=1 TO 5000
480 NEXT I
490 GOTO 10
500 CLS
510 RESTORE 520
520 DATA BAKLAVA,REVANİ,KEMAL PAŞA,SÜTLAÇ
```

```
530 FOR I=1 TO 5
540 READ T$
550 PRINT T$
560 NEXT I
570 FOR I=1 TO 5000
580 NEXT I
590 GOTO 10
600 CLS
610 RESTORE 620
620 DATA ELMA,KİRAZ,AYVA,ARMUT,ERİK,ÜZÜM,ŞEFTALİ
630 FOR I=1 TO 7
640 READ M$
650 PRINT M$
660 NEXT I
670 FOR I=1 TO 5000
680 NEXT I
690 GOTO 10
700 CLS
710 RESTORE 720
720 DATA SU,PORTAKAL SUYU,GAZOZ,KOLA,AYRAN,ÇAY
730 FOR I=1 TO 6
740 READ I$
750 PRINT I$
760 NEXT I
770 FOR I=1 TO 5000
780 NEXT I
790 GOTO 10
800 CLS
810 PRINT "AFİYET OLSUN."
820 END
```

Bu programdaki her yiyecek grubu için, ayrı bir DATA satırı yazılmıştır. Bu nedenle, programın her bölümünün kendisine ait DATA satırını okuması için, satır numarasıyla birlikte ayrı bir RESTORE komutu kullanılmıştır. Bu komut yazılmadığı takdirde, bilgiler birbirine karışacaktır.

WHILE...WEND DÖNGÜSÜ:

WHILE...WEND, bir komut grubunu belirli bir koşula bağımlı olarak çalıştırmak için kullanılan bir döngü yöntemidir. Daha önce işlenmiş olan FOR... NEXT döngüsü, bir döngü değişkenine bağlı olarak bir komut grubunu tekrarlar. WHILE...WEND döngüsünde ise, döngünün temeli WHILE teriminden sonra yazılan bir koşuldur.

Genel yazılım:

```
WHILE <koşul(lar)>
[komutlar]
 WEND
```

WHILE teriminden sonraki koşul veya koşullar geçerli (doğru) olduğu sürece, WHILE ve WEND satırlarının arasındaki komutlar tekrarlanacaktır. Buradaki WEND'in görevi, döngünün bitimini, diğer bir deyişle, koşul yanlış olduğunda programın nereden devam edeceğini göstermesidir.

ÖRNEKLER:

1) Klavyeden girilen bir sayının 2 ile çarpımı 500'den küçük olduğu sürece, bu sayının yarısını ekrana yazan bir program:

```
10 WHILE A*2
20 INPUT A
30 PRINT A/2
40 WEND
```

WHILE ve WEND satırları arasında kalan INPUT ve PRINT komutları tekrarlanacak olan komutlardır. Bu tekrarın adedi programa göre belirli değildir. WHILE teriminden sonra yazılmış olan koşul geçerli olduğu sürece, yani girilen sayının iki katı 500'den küçük olduğu sürece, sayı giriş ve ekrana yazma işlemi devam edecektir.

```
RUN
? 20
10
? 150
75
? 300
150
```

En sonda, koşula uymayan sayı (300) girildiğinde, bilgisayar bu sayının da yarısını yazacaktır. Bunun nedeni, PRINT komutunun da döngünün içinde olmasıdır. Ancak bu komut uygulandıktan ve ekrana sayının yarısı yazıldıktan sonra, koşulun geçersizliği kontrol edilecek ve bilgisayar döngüden çıkacaktır.

2) Klavyeden girilen kelime 5 harften kısa olduğu sürece bu ismin ilk harfini yazan bir program:

```
10 WHILE LEN(A$)<5
20 INPUT A$
30 PRINT LEFT$(A$,1)
40 WEND
```

Klavyeden girilen kelime 5 harften kısa olduğu sürece döngü devam edecek, 5 harfli veya daha uzun bir kelime girildiğinde, bilgisayar bu kelimenin de ilk harfini yazarak döngüden çıkacaktır.

3) 'AYŞE' kelimesini ekrana 10 kez alt alta yazan bir program:

```
10 WHILE S <10
20 PRINT "AYŞE"
30 S = S + 1
40 WEND
```

Bu döngü, S'nin aldığı değerlere göre tekrarlanmaktadır. Daha önce bir değer verilmediği için 0'dan başlayan S'nin değeri, 10'dan küçük olduğu sürece 1'er artacak ve her keresinde ekrana 'AYŞE' yazılacaktır. S'nin değeri 10 olduğunda döngü sona erecektir. (0'dan 9'a kadar 10 sayı olduğundan, kelime 10 kez yazılacaktır. S başka bir değerden başlatılırsa, aynı programa göre tekrar sayısı değişecektir.)

4) 5'ten 15'e kadar sayıları listeleyen bir program:

```
10 S=5
20 WHILE S<16
30 PRINT S
40 S = S + 1
50 WEND
```

Sayaç değişkeni 5'ten başlatılmıştır. 1'er artarak 15'e kadar sayılar ekrana yazılacaktır. Sayaç değeri 16 olduğunda koşul geçersiz olacak ve bu değer ekrana yazılmadan döngü sona erecektir.

5) Klavyeden girilen sayı 5'ten büyük olduğu sürece sayının karekökünü ekrana yazan bir program:

```
10 WHILE A>5
20 INPUT A
30 PRINT SQR(A)
40 WEND

RUN
Ok
```

Program istenildiği gibi çalışmamıştır. Bunun nedeni, WHILE teriminden sonraki koşulun döngüye giriş sırasında geçersiz olmasıdır. Çünkü, A'ya hiçbir başlangıç değeri verilmemiş ve A'nın değeri 0

olarak kalmıştır. Bu durumda bilgisayar döngüyü bütünüyle atlayarak WEND'den sonraki satıra gidecektir. (Burada program duracaktır.)

Programın istenildiği gibi çalıştırılabilmesi için A değişkenine koşula uygun bir başlangıç değeri verilebilir:

```
5 A=6
10 WHILE A>5
20 INPUT A
30 PRINT SQR(A)
40 WEND

RUN
? 16
4
? 81
9
? 4  2
```

Görüldüğü gibi, başlangıç değeri koşula uygun olduğu için döngüye girilmiş ve girilen sayı 5'ten büyük olduğu sürece sayının karekökü yazılmıştır.

6) Sürekli olarak klavyeden girilen sayıları toplama ekleyen, bu toplam 1000'i aştığı zaman toplamı ve kaç giriş yapıldığını yazarak duran bir program:

```
10 WHILE T<=1ØØØ
20 INPUT A
30 T=T+A
40 S=S+1
50 WEND
60 PRINT "TOPLAM =";T
70 PRINT "ADET =";S
```

Klavyeden girilen sayılar sürekli olarak toplama eklenmekte, aynı zamanda bir sayaç yardımıyla kaç adet giriş yapıldığı sayılmaktadır. Toplamın değeri 1000'i aştığında döngü dışına çıkılacak, toplam ve adet yazılarak program sona erecektir.

7) Bankaya yatırılan 100000 liranın faizi çekmemek şartıyla kaç ay sonra 180000 lira olacağını bulan bir program:

(Aylık faiz oranı % 25'tir.)

```
10 P=100000
20 WHILE P<180000
30 P=P+P*25/100/12
40 AY=AY+1
50 WEND
60 PRINT "PARANIZ";AY;"AY SONRA";P;"LİRA OLACAKTIR."
```

Buradaki koşul, yatırılan paranın 180000 liradan az olmasıdır. Para miktarı 180000 lira olana kadar sayaç olarak kullanılan AY arttırılacak, döngü bitiminde paranın bankada kaç ay kalacağı ve çekilebilecek para miktarı yazılacaktır.

PRINT USING KOMUTU:

PRINT USING, çeşitli bilgilerin çıkış ünitelerine yazılırken belirli kalıplara göre düzenlenmesi (formatlanması) amacıyla kullanılır.

Genel kullanım:

PRINT USING <kalıp> ; <yazılacak ifade>

PRINT USING komutu, iki ana grupta incelenebilir:

1) Sayısal Formatlar
2) Alfabetik Formatlar

1) Sayısal Formatlar:

Sayısal formatlarda kullanılan karakterler, (#), (.), (+), (-), (**), ($$), (**$), (^ ^ ^ ^) ve (_) işaretlerinden oluşmaktadır.

(#) Bu işaret, yazılacak sayının her basamağını göstermek için kullanılır. Örneğin iki basamaklı bir sayı yazabilmek için iki # işaretini yan yana kullanmak gerekir.

ÖRNEKLER:

```
PRINT USING "##";34
34
PRINT USING "####";813
813

10 FOR I=1 TO 5
20 READ A
30 PRINT USING "#####";A
40 NEXT
50 DATA 165,4513,46927,1,645

RUN
  165
 4513
46927
    1
  645
```

Yukarıdaki örneklerde, ekrana yazılacak her basamak için bir # işareti kullanılmıştır. En son örnekte de görüldüğü gibi, ekrana yazılacak sayının basamak adedi kalıpta verilen # işaretlerinin adedinden az olursa, sayı kalıp üzerinde sağ tarafa bloklanarak yazılır.

NOT: Eğer yazılmak istenen ifade verilen kalıba sığmayacak kadar uzunsa, bilgisayar sonuç olarak tanımlanan kalıbın yetersiz olduğunu belirtmek amacıyla yazılan ifadenin başına bir yüzde (%) işareti koyar:

```
PRINT USING "###";1234
%1234
```

(.) Nokta işareti, sayının kesirli bölümünü formatlamak, yani, noktadan sonra kaç basamak yazılacağını belirlemek için kullanılır. Bu şekilde formatlanmış bir sayıda, kesirli bölümdeki basamaklar mutlaka yazılırlar. Sayı tamsayıysa, noktadan sonra formattaki adette 0 yazılır. Sayının kesir bölümü formatta belirtilen basamak adedinden daha uzunsa, sayı uygun şekilde yuvarlanır.

ÖRNEKLER:

```
PRINT USING "###.##";65.591
65.59

PRINT USING "###.##";22.468
22.47

PRINT USING "##.####";3.14159265
3.1416

PRINT USING "####.#####";8.49
8.4900

PRINT USING "##.####";285.10496
%285.1050

10 FOR I=1 TO 5
20 READ A,B
30 PRINT A;"/";B;"=";USING "##.###";A/B
40 NEXT
50 DATA 5,4,6,2,9,7,2,9,2,3

RUN
5 / 4 = 1.250
6 / 2 = 3.000
9 / 7 = 1.286
2 / 9 = 0.222
2 / 3 = 0.667
```

(,) Virgül işareti, özellikle büyük sayıların basamaklarının üçlü gruplar halinde virgülle ayrılarak yazılabilmesi için kullanılır.

ÖRNEKLER:

```
PRINT USING "########,";1843525
1,843,525

PRINT USING "#######,";15.73
16
```

```
PRINT USING "#######,";15263
15,263

PRINT USING "######,";6298425
%6,298,425
```

Virgül ve nokta işaretleri birlikte de kullanılabilirler:

```
PRINT USING "####,.##";8475.462
8,475.46
```

ÖRNEK: Toptan yapılan bir alışverişte, çeşitli malların ağırlıkları ve tutarları verilmektedir. Buna göre, her malın bir kilogramının maliyetini hesaplayarak, uygun şekilde ekrana listeleyen bir program:

```
10 PRINT "MAL ADI","AĞIRLIĞI","TUTARI","MALİYETİ (Kg)"
20 PRINT "----------","---------","----------","----------"
30 FOR N=1 TO 6
40 READ M$,A,T
50 PRINT M$,A,USING "######,";T;
60 M=T/A
70 PRINT ,USING "#####,.##";M
80 NEXT N
90 DATA PİRİNÇ,158,330000,ŞEKER,195,120000,
PEYNİR,112,350000
100 DATA FASULYE,235,300000,NOHUT,225,264000,
MERCİMEK,184,126000

RUN
```

MAL ADI	AĞIRLIĞI	TUTARI	MALİYETİ(Kg)
PİRİNÇ	158	330,000	2,088.61
ŞEKER	195	120,000	615,38
PEYNİR	112	350,000	3,125,00
FASULYE	235	300,000	1,276.60
NOHUT	225	264,000	1,173.33
MERCİMEK	184	126,000	684.78

(+) Format kalıbının başına veya sonuna koyulan artı işareti, sonuç olarak yazılan sayının başında veya sonunda sayının işaretinin (+ veya -)yazılmasını sağlar.

ÖRNEKLER:

```
PRINT USING "+###.###";23.5957
+23.596

PRINT USING "+#.###";-6.92
-6.920

PRINT USING "###+";46
46+

PRINT USING "###.##+";-29.498
29.50-
```

(-) Format kalıbının sonuna koyulacak bir eksi işareti, negatif sayıların sonlarında bir eksi işaretiyle yazılmasını sağlar.

ÖRNEKLER:

```
PRINT USING "###.###-";-4.42
4.420-

PRINT USING "##.##-";6.26
6.26
```

(**) Format kalıbının başına koyulacak çift yıldız işareti, sayı sağ tarafta bloklandığında, başta kalan boşlukların yerine '*' işaretinin çıkmasını sağlar.

ÖRNEKLER:

```
PRINT USING "**#.##";57.185
*57.19

PRINT USING "**##.#";1
***1.0

PRINT USING "**##.###";-24.823
*-24.823
```

($$) Format kalıbının baş tarafına koyulacak çift dolar işareti, sonuç olarak yazılan sayının başında bir '$' işaretinin çıkmasını sağlar.

ÖRNEKLER:

```
PRINT USING "$$###.##";94.62
$94.62

PRINT USING "$$###.#";-24.84
-$24.84

PRINT USING "$$#";84
$84
```

($)** Format kalıbının başına koyulacak çift yıldız ve bir dolar işareti, çift yıldız ve çift dolar işaretlerinin etkilerini birleştirir, yani sayının başına bir dolar işareti geldikten sonra solda kalan boşlukların yerine yıldız işaretinin çıkmasını sağlar.

ÖRNEKLER:

```
PRINT USING "**$#.##";61.256
*$61.26

PRINT USING "**$###";18
***$18

PRINT USING "**$##.#";-19.56
*-$19.6
```

(^ ^ ^ ^) Format kalıbının sonuna koyulacak dört yukarı ok işareti, sayının E notasyonuyla yazılmasını sağlar.

ÖRNEKLER:

```
PRINT USING "###.##^ ^  ^ ^";867.83
86.78E+01

PRINT USING "##.#^ ^  ^ ^";86.69
8.7E+01
```

(_) Format kalıbının başına veya sonuna koyulacak alt çizgi işareti, kendisinden sonra gelen karakterin olduğu gibi yazılmasını sağlar.

ÖRNEKLER:

```
PRINT USING "_A####.##";65.18
A 65.18

PRINT USING "###.###_X";9.46816
9.468X
```

2) Alfabetik Formatlar:

Alfabetik formatlarda kullanılan karakterler, (!), (\ \) ve (&) işaretlerinden oluşmaktadır.

(!) Ünlem işareti, yazılacak alfabetik ifadenin yalnızca ilk karakterinin yazılmasını sağlar. Eğer birden fazla alfabetik ifade verilmişse, her birinin ilk harfi yan yana yazılır.

ÖRNEKLER:

```
PRINT USING "!";"BİLGİSAYAR"
B

PRINT USING "!";"PROGRAMLAMA"
P

PRINT USING "!";"BİLGİSAYAR","PROGRAMLAMA"
BP
```

(\ \) Çift ters bölü işareti, yazılacak alfabetik ifadenin sol tarafından başlayarak, aralarındaki boşluk adedinden 2 fazla karakterin yazılmasını sağlar.

ÖRNEKLER:

```
PRINT USING "\   \";"BİLGİSAYAR
BİLGİ
```

PRINT USING "\ \";"PROGRAMLAMA"
PR (Hiç boşluk bırakılmazsa 2 (0+2) karakter yazılır.)

PRINT USING "\ \";"KAHVERENGİ","SARI"
KASA

(&) Bu işaret, alfabetik ifadelerin, satır sonuna gelindiğinde yarıda kesilecek olsalar bile, kursörün bulunduğu pozisyondan başlayarak yazılmasını sağlar.

ÖRNEKLER:

```
A$="BİLGİSAYAR KULLANMAK DÜŞÜNCEYİ GELİŞTİREN BİR UĞRAŞTIR,"
B$=" AMA ÖĞRENMEK İÇİN SABIRLI OLMAK GEREKLİDİR."
PRINT A$;B$
BİLGİSAYAR KULLANMAK DÜŞÜNCEYİ GELİŞTİREN BİR UĞRAŞTIR,
AMA ÖĞRENMEK İÇİN SABIRLI OLMAK GEREKLİDİR.
PRINT A$;USING"& ";B$
```

Bu komut verildiğinde 2. satırdaki cümle. 1. satırdaki cümlenin sonundan devam edecektir.

YAZICI (PRINTER) KULLANIMI

Şu ana kadar yapılmış olan programların çıkışları, ekran üzerinde görüntülenecek şekilde düzenlenmiştir. Bu çıkışların, birçok konuda, sürekli kullanımı sağlayabilmek amacıyla kâğıt üzerine aktarılması gerekebilir. Bilgisayar çıkışlarını kâğıt üzerine aktaran araç, yazıcı (printer) olarak adlandırılır. Birçok gelişmiş çeşitleri olan yazıcıların dünyada en yaygın olarak kullanılan türleri şu yönteme göre çalışır:

Dikey olarak sıralanmış birçok iğne, hareketli bir şeride belirli bir düzene göre vurarak, şeridin hemen arkasındaki kâğıtta istenilen şeklin çıkmasını sağlar. Bu tür printer yazımına, 'DOT-MATRIX' adı verilir. Bunların dışında, az gelişmiş veya ileri teknolojinin ürünü olan yazıcılar da kullanılmaktadır.

BASIC dilinde bilgilerin yazıcıdan çıkmasını sağlamak için yalnızca iki komut kullanılır: LLIST ve LPRINT.

LLIST: LIST komutuyla ekranda görüntülenen programın yazıcıdan çıkmasını sağlar. LIST komutundaki farklı kullanımlar, LLIST komutu için de geçerlidir.

Örnekler: **LLIST 100-150**
LLIST -400
LLIST 300-

LPRINT: PRINT komutunun sonucunun yazıcıda görülebilmesi için kullanılan komuttur. PRINT komutunun bütün uygulamaları LPRINT için de geçerlidir.

Örnekler: **LPRINT 25**
LPRINT "AYŞE"
LPRINT USING "###.##";54.293

WIDTH komutu:

Ekranın veya yazıcının genişliğini (bir satırdaki karakter sayısı) ayarlamak için kullanılan komuttur. Bu genişlik, başlangıç olarak 80'dir.

Ekran genişliğini ayarlamak için WIDTH genişlik şeklinde yazılır:

WIDTH 70 (Ekranın sol tarafındaki 70 sütunluk bölümü kullanılabilir.)

Yazıcı genişliğini ayarlamak için WIDTH "LPT1:",<genişlik> şeklinde yazılır:

WIDTH "LPT1:",15 (Kâğıdın bir satırına en fazla 15 karakter yazılabilir.)

WIDTH "LPT1:",255 (Kâğıt genişliği sonsuz kabul edilir.)

NOT: WIDTH komutuyla genişliği belirtmek için kullanılan sayı en az 1, en fazla 255 olabilir.

SORULAR

1) Klavyeden girilen sayının mutlak değeri 10'dan küçük olduğu sürece sayının sinüsünü ekrana yazan bir program yapınız. (WHILE...WEND kullanılacak.)

2) Klavyeden girilen kelimenin ilk harfi son harfinden farklı olduğu sürece boş bir ekrana bu kelimeyi yazan bir program yapınız. (WHILE...WEND kullanılacak.)

3) Klavyeden sürekli olarak kelime girişi yapan, bu kelimelerden 'A' ile başlayanların adedi 5'ten küçük olduğu sürece girişe devam eden, aksi durumda duran bir program yapınız. (WHILE...WEND kullanılacak.)

4) DATA satırından okunan 5 sayıyı, baş taraftaki boşluklar '*' işaretiyle dolacak şekilde, üçlü gruplar halinde virgülle ayırarak alt alta listeleyen bir program yapınız.

Sayılar: 184657, 795621.45, 78462.9,798156.245, 16549.48532

İŞLETİM SİSTEMLERİ

Disk - Disket Kavramları:

Bilindiği gibi, bilgisayarın başlıca görevlerinden birisi, bilgileri saklayarak istenildiği zaman kullanıcının hizmetine sunmasıdır. Ancak, çok fazla bilgiyi kalıcı olarak bilgisayarın ana belleğinde saklamak iki nedenden dolayı olanak dışıdır:

1) Günümüz teknolojisi çerçevesinde, çok büyük bellek kapasitesine sahip bilgisayarların üretilmesi çok masraflıdır.

2) Ana bellekte saklanan bilgiler, elektrik akımı kesildiği zaman kaybolmaktadır.

NOT: Bu nedenler, yalnızca günümüz koşulları için geçerlidir. Daha geniş bellekli bilgisayarların üretim çalışmaları sürmektedir.

Bu nedenlerden dolayı, bilgi depolamak amacıyla, ana bellek dışında, kalıcı bir yardımcı bellek ünitesine ihtiyaç duyulmuş ve çeşitli kayıt ortamları kullanılmaya başlanmıştır. Delikli kartlar, optik okuyucular gibi yavaş ve kullanışsız araçlardan sonra manyetik kayıt ortamlarının kullanımına geçilmiştir. Günümüzde de en yaygın olarak, küçük bilgisayarlarda kaset veya disket, daha büyüklerinde ise disket veya hard disk üniteleri kullanılmaktadır.

Disket: Floppy (esnek) disk olarak da adlandırılan bu kayıt aracı şekil olarak bir plâğı andırmaktadır. Disketlerin sınıflandırılması, çaplarına göre yapılır. Örneğin, çeşitli bilgisayarlar için 3 inçlik, 3.5 inçlik,

5.25 inçlik disketler kullanılmaktadır. (Bir Amerikan uzunluk ölçüsü olan inç, yaklaşık 2.5 santimetredir.) Bir disket, dikdörtgen şeklinde plastik bir zarf içine yerleştirilmiş disk şeklinde bir manyetik banttan oluşmaktadır. Zarfın ortasında, manyetik diskin disket sürücü tarafından döndürülebilmesine izin verecek dairesel bir boşluk ve bunun yanında manyetik alanın okunabilmesi için diğer bir açıklık vardır. Okuyucu kafa, disket yüzeyine buradan temas ederek bilgi okuma ve kaydetme işlemlerini yapar. Bir disketin kayıt kapasitesi, 200 Kb'den 1400 Kb'ye kadar değişmektedir.

Hard disk: Birden çok manyetik diskin üst üste gelmesiyle oluşan bu kayıt ünitesi, esnek diskete oranla çok daha hızlı, güvenilir ve yüksek bilgi kapasitelidir. Hard diskin, disket gibi bilgisayara takılıp çıkarılması mümkün değildir. Genellikle bilgisayara bir kez monte edilen hard disk, herhangi bir sorun olmadığı sürece çıkarılmaz. Hard diskin kayıt kapasitesi, disketin 500 katına kadar çıkabilmektedir. Çok kullanışlı olan bu aletin maliyeti ne yazık ki çok yüksektir.

Hard disk veya disketin kullanımı, BASIC dili çerçevesinde aynıdır. LOAD ve SAVE komutları verildiğinde, bilgisayar programı kaydetmesi veya okuması gereken yeri kendiliğinden bulabilmektedir. Oysa kaset kullanımında durum çok daha farklıdır. Kaset üzerindeki bilgilerin yerlerinin kullanıcı tarafından kontrol edilmesi şarttır.

Track ve Sektör: Disketin üzerine kaydedilen bilgiler, belirli bir adresleme yöntemine göre saklanırlar. Bu şekilde, yeni bir bilgi kaydedileceğinde disketin nerelerinin boş olduğu veya bir bilgi okunmak istendiğinde bu bilginin disketin neresinde kayıtlı olduğu bilgisayar tarafından kolayca bulunabilir.

Disket üzerinde adresleme yapabilmek için, disketin önce formatlanması gereklidir. Formatlama işlemi sırasında bilgisayar, disketin manyetik bandı üzerinde track (iz) ve sektör (dilim) adları verilen bölgeleri oluşturur. Tracklar, diskin merkezinden dışına doğru sıralı olarak yerleştirilmiş halkalardır. Sektörler ise, diskin merkezinden geçen doğrularla bölünmüş dilimlerdir. Disket üzerine kaydedilen bütün bilgiler, belirli iz ve dilimler üzerine kaydedilerek, bir adresleme sağlanır.

Disket üzerindeki iz ve dilim sayıları, bilgisayara, diskete veya formatlama programına göre değişim gösterebilir. (Genellikle 40 iz ve 9 dilim kullanılır.)

Directory: Disket üzerindeki bilgilerin adreslerinin, uzunluklarının, adlarının ve ilgili bazı diğer bilgilerin saklandığı bölümdür. Dis-

kete kaydedilen bilgiler, her zaman bir grup olarak saklanır. Bu gruba dosya denir; her dosyanın bir adı vardır. Directory'de kayıtlı olan adlar bu dosyaların adlarıdır. Bir dosya disketten okunmak istendiği zaman, bilgisayar önce bu dosyanın directory'deki adresini bulur ve buna göre hangi iz ve hangi sektörlerin okunacağına karar verir. Disketteki herhangi bir dosyanın silinmesi istendiğinde, bu dosyanın kendisi silinmez; yalnızca directory'de kayıtlı olan adı silinir. Böylece bu dosyaya daha sonra ulaşılması engellenmiş olur.

İŞLETİM SİSTEMLERİ:

Normal olarak birçok yeteneğe sahip olan bilgisayarların bu yeteneklerinden yararlanmak için özel programlara ihtiyaç vardır. Disk veya disket kullanan bütün bilgisayarlar için, bilgisayarın donanımını ve mantıksal kaynaklarını yönetmek ve kullanmak amacıyla üretici firmalar tarafından bazı programlar geliştirilmiştir. Bu programlara genel olarak işletim sistemi denir. İşletim sistemleri, bilgisayar ve kullanıcı arasındaki iletişimi sağlarlar. Bilgisayar kullanımından alınacak verim, o bilgisayarda kullanılan işletim sisteminin mükemmelliğine bağlıdır. Dünyada kullanılan çeşitli bilgisayarlar için değişik işletim sistemleri vardır. Bunlardan en yaygın olanları MS/DOS, DOS+, CP/M, UNIX ve ZENIX'tir.

CP/M ve MS/DOS işletim sistemleri komutlarından bazıları:

İşletim sistemi komutları, dahilî (internal) ve haricî (external) olarak ikiye ayrılır. Dahilî komutlar, sistem disketine gerek olmadan doğrudan doğruya kullanılabilirler. Haricî komutların kullanılması için ise sistem disketlerine ihtiyaç vardır.

DIR: Diskette kayıtlı olan program ve dosyaların isimlerini, yani directory kayıtlarını listelemek için kullanılan komuttur. İki işletim sisteminde de kullanımı aynıdır. MS/DOS sisteminde bu komutun yardımıyla, programların kayıt tarihleri ve zamanları da görülebilir. Bu komutun BASIC karşılığı, FILES komutudur.

Komut tek başına kullanıldığında, directory'deki bütün dosya adlarını listeler:

```
A> DIR
     PROG1 .BAS
     PRINT .COM
     DSY .BAS
```

Dir komutuyla, directory'de yalnızca istenilen türden kayıtların listelenmesi de sağlanabilir:

```
A>  DIR *.BAS
     PROG1 .BAS
     DSY .BAS
```
(Burada, BAS ekiyle biten kayıtlar listelenmiştir.)

```
A > DIR PR*.*
     PROG1 .BAS
     PRINT .BAS
```
(Burada, PR ile başlayan kayıtlar listelenmiştir.)

ERA / DEL: Directory'deki kayıtlardan istenilenleri silmek için kullanılan komuttur. CP/M'de ERA, MS/DOS'te ise DEL şeklinde yazılır. BASIC dilinde aynı amaçla KILL komutu kullanılır.

Bu komutların uygulanışı şu şekildedir: (Sol tarafta CP/M, sağ tarafta MS/DOS uygulaması gösterilmiştir.)

A>ERA PROG1.BAS **A>DEL PROG1.BAS**
(PROG1.BAS isimli dosya silinir.)

A>ERA *.COM **A>DEL *.COM**
(COM eki almış bütün programlar silinir.)

A>ERA *.* **A>DEL *.***
(Directory'deki bütün kayıtlar silinir.)

NOT: Daha önce de belirtildiği gibi, bu komutlarla silinen bir dosya, aslında disketten tamamen silinmez. Yalnızca directory-'deki kaydı silinir. Böylece bu dosyaya ulaşılması engellenmiş olur.

TYPE: İki işletim sisteminde de aynı şekilde kullanılan bu komutun amacı, ASCII olarak kaydedilmiş dosyaların ekrana listelenmesidir. Mallard BASIC'de bu komut aynı şekilde kullanılabilir; ancak GW-BASIC'te karşılığı yoktur.

TYPE komutundan sonra, ekrana listelenmesi istenen dosyanın adı tam olarak yazılmalıdır.:

A>TYPE STOK.DAT
(STOK.DAT isimli dosya ekrana listelenir.)

REN: Bir dosyanın ismini değiştirmek için kullanılan komuttur. (Dosyadaki bilgilerle ilgili bir işlem yapmaz; yalnızca directory'de kayıtlı olan dosya ismini değiştirir.) MS/DOS'ta kullanılırken, REN komutuyla birlikte önce eski dosya adı, sonra yeni ad yazılır.

A>REN PROG1.BAS PROGRAM1.BAS

Bu komut, PROG1.BAS adlı dosyanın adını PROGRAM1.BAS olarak değiştirir.

NOT: '*' işareti kullanarak grup halinde değişiklik yapmak da mümkündür.

A>REN *.BAS *.DAT

Bu komut, BAS ekiyle biten bütün dosyaların eklerini DAT olarak değiştirir.

Komutun CP/M'deki yazılımı da benzer şekildedir. Ancak, burada önce yeni isim, sonra eski isim yazılır; isimler arasında da '=' işareti kullanılır:

A>REN PROGRAM1.BAS=PROG1.BAS
A>REN *.DAT=*.BAS

REN komutunun BASIC dilindeki karşılığı, NAME komutudur. Kullanımı, NAME "eski isim" AS "yeni isim" şeklindedir.

COPY: MS/DOS işletim sistemine ait kopyalama komutudur. COPY kullanarak bir disketteki dosyalardan istenilenler bir diğer diskete veya aynı disketin üzerine kopyalanabilir.

COPY komutuyla birlikte, önce kopyalanmak istenen kaynak dosyanın adı, daha sonra kopyalama yapılacak disket sürücünün adı ve eğer istenirse kopyalanan dosyanın yeni adı yazılır:

A>COPY A:PROG1.BAS B:PROG1.BAS

A: sürücüsündeki PROG1.BAS isimli program B: sürücüsüne PROG1.BAS ismiyle kopyalanır. Burada 'A:' yazmaya gerek yoktur, çünkü bilgisayar zaten A: sürücüsünü okumaktadır. Ayrıca, ikinci kez PROG1.BAS yazmak da gereksizdir, çünkü program adı değiştirilmek istenmemiştir. Bu komut, en kısa şekliyle şöyle yazılabilir:

A>COPY PROG1.BAS B:

A>COPY PROG1.BAS B:PROGRAM.BAS

Bu komut, A: sürücüsündeki PROG1.BAS isimli programı, B: sürücüsüne PROGRAM.BAS ismiyle kopyalayacaktır. İsim değiştirilmek istendiğinden, yeni isim yazılmak zorundadır.

A>COPY STOK.DAT KAYIT.DAT

Bu komutla, A: sürücüsündeki STOK.DAT dosyası, aynı disket üzerine KAYIT.DAT ismiyle kopyalanacaktır. (Aynı disket üzerine kopyalama yapıldığında, mutlaka farklı dosya ismi kullanılmalıdır.)

A>COPY STOK.DSY+VERI.DAT B:KAYIT.TOP

Bu komut, A: sürücüsündeki STOK.DSY ve VERI.DAT dosyalarını arda arda ekleyerek, B: sürücüsüne KAYIT.TOP ismiyle kopyalayacaktır. (Bu tür kopyalamalar, farklı dosya ismi verilmesi şartıyla aynı disket üzerine de yapılabilir.)

DIR ve DEL komutlarıyla kullanılan * işareti, COPY komutuyla da kullanılabilir:

A>COPY *.BAS B:

BAS ekine sahip bütün programlar B: sürücüsüne kopyalanır.

PIP: CP/M işletim sisteminde, kopyalama işlemlerini yapmak üzere kullanılan hazır programdır. Kullanılışı MS/DOS'taki COPY komutunun kullanılışına benzer. PIP komutuyla birlikte, önce kopyalama yapılacak sürücünün ve programın adı, sonra kopyalanacak program yazılır. Aralarında ise '=' işareti yazılır:

A>PIP B:PROGR1.BAS=A:PROG1.BAS

COPY örneğinde de olduğu gibi, bu örnekte de A: ve ilk yazılan PROG1.BAS gereksizdir. Komut kısaca şu şekilde yazılabilir:

A>PIP B:=PROG1.BAS

COPY komutunda, dosyaları ard arda ekleyerek kopyalamak için kullanılan '+' işareti yerine, PIP ile birlikte ',' kullanılır.

A>PIP B:KAYIT.TOP=STOK.DSY,KAYIT.DAT

Dosya adlarını gruplamak için yararlanılan '*' işareti, PIP komutuyla birlikte de kullanılabilir:

A>PIP B:=PR*.*

NOT:

A) COPY ve PIP komutlarının kullanımları birbirlerine çok benzemekle birlikte, bu iki komuttan sonra yazılması gereken dosya adlarının sırasına dikkat edilmelidir. Yapılacak bir yanlışlık, diskette kayıtlı bir dosyanın istenmeden silinmesine neden olabilir. Önemli bir fark da, MS/DOS işletim sisteminin COPY olanağını herhangi bir sistem disketine ihtiyaç doğurmadan kullanıcıya tanıması, fakat CP/M'de, PIP programının bulunduğu bir disket kullanımının zorunlu olmasıdır. Diğer bir deyişle, COPY dahilî komuttur; PIP ise haricî komuttur.

B) Tek sürücülü bilgisayarlar çift sürücülü olarak kullanılmak istendiğinde, bilgisayar aynı sürücüyü A: ve B: olarak çalıştırabilir. Yalnız, sürücü değiştirmek gerektiğinde, kullanıcının aynı sürücüye yeni bir disket takması için bir mesaj verir.

DISCKIT3 PROGRAMININ UYGULANMASI:

Amstrad 6128 bilgiyarlarının sistem disketlerinde bulunan DISCKIT3 programı, disketlerin formatlanması, kopyalanması veya kontrol (verify) edilmesi için kullanılır. İşletim sistemine geçtikten sonra, bu programı çalıştırmak için DISCKIT3 yazmak yeterlidir. Programa girdikten sonra, sistem disketi bilgisayardan çıkartılır; uygulama yapılacak disket takılır.

Formatlama: Ana menüden f4 tuşuna basılarak format seçeneğine geçilir. Daha sonra, eğer sistem formatı yapılacaksa f9, data format yapılacaksa f6 tuşlarına basılır. Bilgisayar formatlama işleminin yapılıp yapılmayacağını son bir kez sorar. Y (Yes) tuşun basıldıktan sonra disket formatlanır. (Formatlama sırasında formatlanmakta olan izin numarası ekranın sol üst köşesine yazılır.)

Kontrol (verify) etme: Ana menüden f1 tuşuna basılarak bu bölüme geçilir. Bilgisayar, kontrol işleminin yapılıp yapılmayacağını son bir kez sorduktan sonra, disketi kontrol eder. Bu sırada, eğer disket formatlanmamışsa, veya disket üzerinde herhangi fiziksel bir bozukluk varsa, bu durumları açıklayıcı mesajlar verir.

Kopyalama: Kopyalama seçeneğine, ana menüden f7 tuşuna basılarak geçilebilir. Diğer seçeneklerde olduğu gibi, kopyalamanın da yapılıp yapılmayacağını son bir kez sorduktan sonra, bilgisayar kopyalama işlemini başlatır. Önce okunacak disketin takılması için, bu disket takılıp okunduktan sonra da yazılacak disketin takılması için bir mesaj verir. Bir disket, bu işlemin üç kez tekrarlanmasıyla (disketlerin üç kez takılıp çıkarılmasıyla) kopyalanmış olur. Burada dikkat edilecek nokta, kopyalama yapılan disket üzerinde önceden kaydedilmiş olan bütün bilgilerin silinmesidir.

Programdan çıkmak için, ana menüde f10 tuşuna basılır.

IBM PC uyumlu bilgisayarlarda kullanılan MS/DOS işletim sisteminde DISCKIT3 gibi bir hazır programa ihtiyaç yoktur. Burada, birer MS/DOS programı (ve komutu) olan FORMAT veya DISKCOPY gibi olanaklardan yararlanmak mümkündür. Bu haricî komutlar, işletim sistemi disketi takıldıktan sonra şu şekilde uygulanır:

FORMAT A: A: sürücüsündeki yeni disketi formatlamak için kullanılır.

DISKCOPY A: B: A: sürücüsündeki disketi, B: sürücüsündeki diskete kopyalamak için kullanılır.

BASIC'TEN İLETİM SİSTEMİNE GEÇİŞ:

Bilgisayarlar, ilk açıldıklarında işletim sistemine girerler. Daha sonra BASIC yüklenerek BASIC programlama yapılır. Tekrar işletim sistemine dönebilmek için, SYSTEM komutu kullanılır. Yeniden BASIC programlama yapabilmek için BASIC yorumlayıcı programı işletim sisteminden bir kez daha çağrılmalıdır.

Genel kullanım:

SYSTEM

IBM PC uyumlu bilgisayarlarda, BASIC'ten işletim sistemine geçici bir süre için dönebilmek amacıyla SHELL komutu kulanılır. Bu komuttan yararlanabilmek için, disket sürücüde işletim sistemi disketinin (en azından COMMAND.COM programının bulunduğu bir disketin) takılı olması gerekir.

Genel kullanım:

SHELL ["program adı"]

SHELL komutu parametresiz olarak kullanıldığında bilgisayar işletim sistemine geçip, sistemle ilgili komutların verilmesini bekler. Bu durumda tekrar BASIC programlamaya dönebilmek için EXIT komutu kullanılmalıdır.

Komut parametreli olarak kullanıldığında, tırnak içinde verilmiş olan dahilî komut veya program uygulandıktan sonra bilgisayar kendiliğinden BASIC programlamaya döner. Örnek: SHELL "COPY A:*.BAS B:"

NOT: SYSTEM komutuyla işletim sistemine geçildiğinde, bilgisayarın belleğinde bulunan BASIC program ve bütün değişkenler silinir. Bunun yerine EXIT'le geri dönmek üzere SHELL kullanıldığında, hem BASIC program, hem de değişkenler oldukları gibi saklanırlar.

FILES, KILL ve NAME komutları:

FILES: Diskette kayıtlı olan dosya isimlerini (directory) ekrana listelemek için kullanılan komuttur. Bu komut tek başına kullanıldığında, disketteki bütün dosya adları listelenir. İstenirse, FILES komutundan sonra "" işaretleri arasında yazılabilecek bir ifade, yalnızca görülmesi istenen dosyaların listelenmesini sağlar.

Genel kullanım:

FILES ["dosya grubunu belirten ifade"]

ÖRNEKLER:

```
-- FILES                (Bütün dosyalar listelenir.)
   A:\
   PERSONEL.    SOZLUK.     STOK.     KELIME.BAS
   PERSON.BAS   HESAP.BAS   KAYIT.

— FILES"*.BAS"          ('BAS' ekini almış programlar listelenir.)
   A:\
   KELIME.BAS   PERSON.BAS   HESAP.BAS

– FILES"*."             (Sonunda ek olmayan dosyalar listelenir.)
   A:\
   PERSONEL     SOZLUK       STOK      KAYIT.

– FILES"S*.*"
   A:\                  ('S' ile başlayan dosyalar listelenir.)
   SOZLUK               STOK.
```

– FILES"P*.*"
A:\ ('P' ile başlayan dosyalar listelenir.)
PERSONEL PERSON.BAS

KILL: Disketteki dosyalardan istenilenleri silmek için kullanılan komuttur. Tek başına kullanılamaz; FILES komutundan sonra tırnak işaretleri içinde yazılan dosya adıyla birlikte kullanılmalıdır.

Genel kullanım:

KILL"dosya grubunu belirten ifade"

ÖRNEKLER:

- KILL"PERSON.BAS" ('PERSON.BAS' isimli program silinir.)
- KILL"S*.* ('S' ile başlayan bütün dosyalar silinir.)
- KILL"*.BAS" ('BAS' ekini almış bütün programlar silinir.)
- KILL"*.*" (Disketteki bütün dosyalar silinir.)

NOT: KILL komutuyla birlikte kullanılan dosya diskette kayıtlı değilse, bilgisayar dosyanın bulunamadığını belirten bir hata mesajı (File not found) verir.

NAME: Disketteki herhangi bir dosyanın içeriğini değiştirmeden yalnızca ismini yenilemek için kullanılan komuttur.

Genel kullanım:
NAME"dosyanın eski adı" AS "dosyaya verilecek yeni ad"

ÖRNEKLER:

- NAME "PERSONEL" AS "PRSNL"
('PERSONEL' dosyasının ismi 'PRSNL' olarak değişir.)

- NAME "STOK" AS "STOK.DAT"
('STOK' dosyasının ismi 'STOK.DAT' olarak değişir.)

NOT: NAME komutundan sonra kullanılan eski isim, diskette kayıtlı bir dosyanın ismi olmalıdır; yeni isimde bir dosya ise diskette bulunmamalıdır.

PROGRAMLARIN DİSKET ÜZERİNDE SAKLANMASI VE ÇALIŞTIRILMASI:

SAVE komutu: Bilgisayarın belleğindeki programı herhangi bir kayıt ünitesine kaydetmek için kullanılan komuttur. Komutun üç kullanımı vardır:

1) SAVE"Program adı" Bellekteki BASIC programı (daha az yer kaplaması ve aktarmanın kısa sürede tamamlanabilmesi amacıyla) özel bir biçimde kodlayarak diskete kaydeder.

2) SAVE"Program adı",A Bellekteki BASIC programı herhangi bir kodlama yapmadan ASCII dosya olarak diskete kaydeder.

3) SAVE"Program adı",P Bellekteki programı, program listesi bir daha görülemeyecek şekilde diskete kaydeder. Bu tür programlar yalnızca RUN komutuyla çalıştırılabilir; kullanıcı (ve hatta programcı) tarafından listeleri alınamaz.

NOT: Program adı, en fazla 8 harften oluşan bir addır. Bu harfler, yalnızca İngiliz alfabesindeki harfler olabilir. Ayrıca, 3 harften uzun olmamak şartıyla bir ek de kullanılabilir. Eğer bu ek verilmezse, bilgisayar BASIC programların sonuna 'BAS' ekini kendiliğinden ekler. Program adı ve ek arasında nokta işareti (.) bulunmalıdır.

LOAD komutu: Kayıt ünitesindeki programı bilgisayarın belleğine yüklemek için kullanılan komuttur.

Program SAVE komutuyla herhangi bir parametre (A veya P) yazılmadan kodlanarak kaydedilmişse, bu kodlar tekrar programa dönüştürülerek listelenebilir.

SAVE komutuyla A parametresi kullanılarak program kodlanmadan kaydedilmişse, LOAD komutu bu programı da belleğe yükleyebilir.

Program kaydedilirken SAVE komutuyla birlikte P parametresi kullanılmışsa, LOAD komutu herhangi bir hata mesajı çıkarmadan çalışır. Ancak, bu programın listesi alınamaz.

RUN komutu: Bellekteki programın çalıştırılabilmesi için kullanılan bu komutun ikinci amacı da, diskette kayıtlı bir programı önce belleğe aktarıp, sonra çalıştırılabilmesidir. LOAD komutuyla programı belleğe yükleyip sonra RUN komutuyla çalıştırmak yerine, bu iki komutun kullanımını birleştirici özelliği olan RUN komutu aşağıdaki şekilde yazılabilir.

Genel kullanım:
RUN"Program adı"

Bu komut, özellikle 'P' parametresi kullanılarak kaydedilmiş programları çalıştırabilmek için yararlıdır.

MERGE komutu: Bellekteki programın üzerine, diskette kayıtlı ikinci bir programı eklemek için kullanılan komuttur.

Genel kullanım:
MERGE"Program adı"

NOT:

A) MERGE komutuyla eklenecek program, SAVE komutunda 'A' parametresi kullanılarak ASCII dosya olarak kaydedilmiş olmalıdır.

B) Eklenen programdaki satır numaralarından bazıları (veya hepsi) bellekteki programın satır numaralarıyla aynı olursa, disketten yüklenen satırlar bellektekilerin yerini alacaktır.

CHAIN komutu: Bellekteki değişkenleri silmeden, disketteki bir programı belleğe yükleyip çalıştırmak amacıyla kullanılan komuttur. Özellikle belleğe sığmayacak kadar uzun programlar parçalanarak diskete kaydedildiğinde, bu programlar arasındaki iletişimi sağlar.

Genel kullanım:

CHAIN "Program adı",[satır numarası],[ALL]

Komuttan sonra kullanılabilecek olan satır numarası, çalıştırılacak ikinci programa aittir. Program bu satırdan başlayarak çalışacaktır. Bu satır numarası yazılmazsa, program en baştan başlayacaktır.

ALL parametresi kullanılırsa, birinci programdaki bütün değişkenler, diziler ve matrisler ikinciye aktarılacaktır. Eğer bu parametre kullanılmazsa, ikinci programa aktarılacak değişkenler CHAIN kullanılmadan önce COMMON komutuyla belirtilmelidir:

Örnek: **COMMON A$,T,N$()**
(A$ ve T değişkenleri ile N$ dizisi aktarılacaktır.)

NOT:

A) CHAIN komutu, dosyaların kapanmasına neden olmaz.

B) CHAIN komutu, birinci programdaki değişken türü ve fonksiyon tanımlamalarını ikinci program için geçersiz hale getirir. (DEFINT, DEFSNG, DEFDBL, DEFSTR ve DEF FN komutları gerekirse ikinci programda da yer almalıdır.)

CHAIN MERGE komutu:

MERGE ve CHAIN komutlarının görevlerini birleştiren komuttur. Yani, önce ikinci program birincinin üzerine eklenir ve daha sonra çalıştırılır. Bu komutla birlikte, CHAIN komutunun parametreleri dışında DELETE parametresi kullanılır.

Genel kullanım:

CHAIN MERGE"Program adı",[satır numarası],
[ALL],[DELETE satır numaraları]

DELETE parametresinin dışındaki parametreler, CHAIN komutunda olduğu gibi kullanılır. İstenirse yazılabilen DELETE parametresinin görevi, birinci programda ihtiyaç olmayan bir bölümün silinmesini sağlamaktır.

Örnek: **CHAIN MERGE"STOK",250,ALL,DELETE 100-200**

STOK isimli program, bellekteki programın üzerine eklenerek, 250 numaralı satırdan başlayarak çalışacaktır. Bellekteki programın bütün değişkenleri, bu programa da aktarılacaktır. Bu işlem sırasında, bellekte daha önceden yer alan programın 100 ve 200 numaralı satırlar arasındaki bölümü silinecektir.

NOT: Eklenecek program, burada da ASCII olarak kaydedilmiş olmalıdır.

DOSYALAMA

Programlar tarafından kullanılacak veriler, kalıcı olmaları amacıyla, diskette saklanabilirler. Değişkenlere atanan veya klavyeden girilen bilgiler, bilgisayar kapatıldığında silinirler. Bu tür bilgilerin kaybolmamaları için belirli bir ortamda (disk veya disket) saklanmaları gerekir. Diskete kaydedilecek bütün bilgiler işletim sistemleri konusunda da anlatıldığı gibi, belirli gruplar halinde saklanırlar. Bu grupların her birinedosya(file, kütük) denir. Her dosyanın bir ismi vardır. Programcının karar verdiği bu isim, genellikle dosyanın içerdiği bilgileri anımsatıcı nitelikte olur. Bir şirketteki muhasebe kayıtlarına ait bilgilerin muhasebe dosyasında, personel kayıtlarına ait bilgilerin de personel dosyasında kayıtlı olması, buna örnek olarak gösterilebilir.

Temel olarak, disket üzerinde kayıtlı bilgilerin kullanımı, program içinde yazılmış bir DATA satırındaki bilgilerin kullanımına benzer. Yalnız, arada önemli farklar vardır:

1 - DATA satırına yazılacak bilgiler, bilgisayarın bellek kapasitesiyle sınırlıdır; bir dosyaya ise kapasitesi çok daha fazla olan disketin (veya hard disk'in) alacağı kadar bilgi kaydedilebilir.

2 - DATA satırına yeni bir bilgi girilmesi, ancak programcı tarafından yapılabilecek bir işlemdir; oysa bir dosyaya bilgi kaydetmek, özel bir program yoluyla yapıldığından, bu programı kullanan herhangi bir kişi yeni bilgi kaydedebilir.

3 - DATA satırı, yalnız içinde bulunduğu program tarafından okunabilir; oysa, bir dosyayı, birbirinden bağımsız birçok program kullanabilir.

Bir dosya oluşturulurken, önce bu dosyanın adı, daha sonra da kayıt deseni belirlenir. Dosya adı, en fazla 8 karakterden oluşabilir; burada yalnız İngiliz alfabesindeki harfler bulunabilir. (Boşluk, virgül gibi karakterler kullanılamaz.) Dosya adına istenirse bir noktadan sonra 3 harfli bir ek de ilâve edilebilir.

Kayıt deseni: Dosyalar, kayıtlardan oluşur. Bütün kayıtlar, kayıt deseni ile belirlenmiş bir düzene göre dosyaya yazılırlar. Kayıt deseni, dosyadaki herhangi bir satır üzerindeki bütün bilgileri içerir. Örneğin, bir personel dosyasında on kayıt olduğu düşünülürse, bu kayıtlardan her biri, personel adı, soyadı, sigorta numarası, maaşı gibi bilgilerden (sahalar) oluşur. Her kaydın içerdiği bu sahalar grubuna, kayıt deseni denir.

Kayıt deseni hazırlanırken dikkat edilecek noktalar:

1 - Kayıt üzerinde bulunan bilgilerin belirlenmesi: Dosya okunarak listelenecek veya başka bir amaç için kullanılacak bilgilerin oluşturulabilmesi için gerekli olan sahaların saptanması. (Örnek: ADET, FİYAT, TUTAR başlıkları altında yazılacak bir listeyle ilgili bilgiler kaydedilirken, tutarların kayıt deseninde yer alması gereksizdir, çünkü istenildiğinde dosyadan okunacak adet ve fiyata göre tutarlar hesaplanabilir.)

2 - Bir kayıt üzerindeki bilgilerin sırası, kullanımda kolaylık sağlayacak şekilde düzenlenmeli ve bu sıra bir daha bozulmamalıdır.

Dosyalama Türleri:

1 - Sıralı erişimli (sequential access) dosyalama : Bütün kayıtlara baştan başlayarak sırayla erişilir.

2 - Rasgele erişimli (random access) dosyalama : Bütün kayıtlara dosyadaki sıra numaralarına göre erişilir.

3 - Index'li dosyalama : Bütün kayıtlara belirli bir index'e göre erişilir.

SIRALI ERİŞİMLİ DOSYALAMA YÖNTEMİ (SEQUENTIAL ACCESS)

Sıralı erişimli dosyalama yönteminde, gerek dosyaya bilgi kaydederken, gerekse bu bilgileri dosyadan okurken, belli bir sırayı izlemek zorunludur.

Aşağıdaki maddeler, sıralı erişimli bir dosyayla ilgili genel özellikleri vermektedir:

— Sıralı erişimli bir dosyadaki bilgiler kaydediliş sırasına göre saklanır, yani ilk kaydedilen bilgi birinci sırada, son kaydedilen bilgi sonuncu sıradadır.
— Herhangi bir kaydın okunması için o kayıttan önceki bütün kayıtların okunmuş olması gerekir.
— Herhangi bir bilgi, daha önce oluşturulmuş bir dosyanın istenilen bir yerine doğrudan doğruya kaydedilemez.

Sıralı erişimli bir dosyanın oluşturulması:

1 - Dosya, kapsadığı bilgileri anımsatır nitelikte bir isim verilerek açılır. Bunun için OPEN komutu kullanılır:

Genel kullanım:

OPEN "O",<dosya numarası>,"dosya adı"
veya
OPEN "dosya adı" FOR OUTPUT AS <dosya numarası>

NOT: Dosya bu şekilde açılırken, daha önceden kaydedilmiş aynı isimde başka bir dosya varsa, bu dosya tamamıyla silinir; yerini yenisi alır.

2 - İstenilen bilgiler kayıt desenine uygun şekilde dosyaya kaydedilir. Bunun için, WRITE veya PRINT komutları kullanılır. Her iki komuttan sonra da, hangi dosyaya kayıt yapılacağını gösteren # işaretiyle birlikte dosya numarası yazılmalıdır. (Aynı işaret, dosyaları açan veya kapatan komutlarla birlikte, dosya numarasından önce kullanılabilir.)

Genel kullanım:

PRINT #<dosya numarası>,sahalar

WRITE #<dosya numarası>,sahalar

Bu komutlarla birlikte kullanılan sahalar, bilgilerin kendileri olabildiği gibi, READ veya INPUT komutlarıyla elde edilmiş, değişkenlerde saklanan bilgiler de olabilir.

3 - Açılmış olan dosya, istenilen bütün bilgiler kaydedildikten sonra kapatılır. Dosyayı kapatmak için, CLOSE komutu kullanılır.

Genel kullanım:

CLOSE <dosya numarası>

NOT: CLOSE komutuyla birlikte dosya numarası kullanılmazsa, o ana kadar açılmış olan bütün dosyalar kapatılır. Belirli birkaç dosya birden kapatılmak istenirse, dosya numaraları aralarında virgülle yazılabilir. (Örnek: CLOSE 1,3)

ÖRNEK: 'PERSONEL' isimli bir dosyayı açarak, bu dosyaya bir kişiye ait isim, soyadı, sicil numarası, ve maaşı kaydeden, daha sonra bu dosyayı kapatan bir program:

```
10 OPEN"O",1,"PERSONEL"
20 INPUT "İSİM         ";A$
30 INPUT "SOYADI       ";S$
40 INPUT "SİCİL NO     ";SN
50 INPUT "MAAŞ         ";M
60 WRITE#1,A$,S$,SN,M
70 CLOSE 1
```

Bu programda, dosya açıldıktan sonra bilgiler klavyeden girilmiş, dosyaya kaydedilmiş ve dosya kapatılmıştır. Bu şekilde, içinde tek bir kayıt bulunan bir dosya oluşturulmuştur. Burada kayıt deseni, A$, S$, SN ve M değişkenleriyle gösterilen isim, soyadı, sicil numarası ve maaş bilgilerinden oluşmaktadır.

Bilgisayar kullanılmayan uygulamalarda da, bilgilerin depolanması için benzer işlemler yapılır. Önce bir dosya alınır (OPEN komutu) ve bu dosyanın üzerine içindeki bilgileri anımsatan bir isim yazılır (dosya adı). Daha sonra kaydedilmek istenen bilgiler bir kâğıda yazılarak (INPUT komutları) dosyaya yerleştirilir (WRITE#1 komutu). İşi biten dosya, kapatılarak (CLOSE komutu) yerine koyulur.

Yukarıdaki programdaki INPUT ve WRITE komutları bir döngü içinde yazılırsa, döngü adedinde kayıt dosyaya yazılmış olur:

```
10 OPEN"O",1,"PERSONEL"
20 FOR K=1 TO 3
30 CLS
40 INPUT "İSİM         ";A$
50 INPUT "SOYADI       ";S$
60 INPUT "SİCİL NO     ";SN
70 INPUT "MAAŞ         ";M
80 WRITE#1,A$,S$,SN,M
90 NEXT K
100 CLOSE 1
```

3 kişi için düzenlenmiş bu programla 'PERSONEL' dosyasına 3 ayrı kayıt yazılacaktır. 30 numaralı satırdaki CLS komutu, her kişi için bilgiler klavyeden girilmeden önce ekranın silinmesini ve düzenli çalışmayı sağlayacaktır.

Burada istenilen bilgilerin dosyaya kaydedilme aşamaları şu şekildedir:

a) Klavyeden girilen bilgilerin değişkenlere yerleştirilmesi. (INPUT)

b) Değişkenlerdeki bilgilerin dosyaya yazılması. (WRITE)

Program çalıştırıldığında, her kayıt girildikten sonra ekran silinerek, üç ayrı ekranda aşağıdaki sonuç elde edilecektir:

```
İSİM       ? AHMET
SOYADI     ? TAŞ
SİCİL NO   ? 29846
MAAŞ       ? 235000
```

```
İSİM       ? İBRAHİM
SOYADI     ? MUTLU
SİCİL NO   ? 19467
MAAŞ       ? 180000
```

```
İSİM       ? GÜLER
SOYADI     ? ODUNCU
SİCİL NO   ? 38127
MAAŞ       ? 300000
```

Klavyeden yapılan her bilgi girişinde, ilgili değişkenlerde daha önce bulunan bilgiler silinmekte ve en son girilen bilgiler saklanarak dosyaya yazılmaktadır. Dolayısıyla, bir döngü yardımıyla, döngünün her tekrarında ayrı bir kaydın dosyaya yazılması mümkün olmaktadır.

Sıralı erişimli bir dosyanın okunması:

1 - Okunacak dosya, kaydedilirken verilmiş olan isimle açılır. Bunun için OPEN komutu kullanılır:

Genel kullanım:

OPEN "I",<dosya numarası>,"dosya adı"
veya
OPEN "dosya adı" FOR INPUT AS <dosya numarası>

2 - Dosyadaki bilgiler kayıt desenine uygun bir şekilde okunur. Bunun için INPUT komutu kullanılır:

Genel kullanım:

INPUT #<dosya numarası>, kayıt desenine uygun değişkenler

3 - Okunma işlemi biten dosya kapatılır. Kayıt yapmak üzere açılmış bir dosyada da olduğu gibi, okunan bir dosyayı kapatmak için de CLOSE komutu kullanılır.

ÖRNEK: Daha önce oluşturulmuş olan 'PERSONEL' isimli dosyadaki kayıtları okuyarak bilgileri ekrana uygun başlıklar altında listeleyen bir program:

```
10 OPEN"I",1,"PERSONEL"
20 CLS
30 PRINT "İSİM","SOYADI","SİCİL NO","MAAŞ"
40 PRINT "--------","---------","--------","---------"
50 FOR T=1 TO 3
60 INPUT #1,A$,S$,SN,M
70 PRINT A$,S$,SN,M
80 NEXT T
90 CLOSE 1
```

Programın sonucu şu şekilde olacaktır:

İSİM	SOYADI	SİCİL NO	MAAŞ
AHMET	TAŞ	29846	235000
İBRAHİM	MUTLU	19467	180000
GÜLER	ODUNCU	38127	300000

NOT: Dosya okunurken, INPUT komutundan sonraki değişkenlerin adedi ve nitelikleri (alfabetik veya sayısal), bu kayıtlar dosyaya yazılırken WRITE komutunda belirtilen değişkenlere uygun olmalıdır. (Değişken adları aynı olmak zorunda değildir; örneğin, A$ olarak kaydedilen bir isim, B$ olarak okunabilir, fakat alfabetik olması gerektiğinden B olarak okunamaz.)

ÖRNEKLER:

1 - 'STOK' isimli bir dosyaya, aşağıdaki kayıt desenine uygun 8 kayıt yazan bir program:

MAL NO	MAL ADI	MİKTARI	FİYATI

```
10 OPEN"O",1,"STOK"
20 FOR I=1 TO 8
30 CLS
40 INPUT "MAL NO    ";MN
50 INPUT "MAL ADI   ";MA$
60 INPUT "MİKTARI   ";MK
70 INPUT "FİYATI    ";FY
80 WRITE#1,MN,MA$,MK,FY
90 NEXT I
100 CLOSE 1
```

Bu programa göre, 8 mala ait bilgiler (8 kayıt) klavyeden girilecek ve diskete (STOK dosyasına) kaydedilecektir. Burada, klavyeden girilen herhangi bir bilgi yanlış olursa ve giren kişi bunu farketmeden RETURN tuşuna basarsa, bu yanlış bilgi olduğu gibi dosyaya yazılır. Dosyadaki yanlış bir bilgiyi düzeltmek bilgi girişi sırasında mümkün olmadığından, bu tür bir yanlışlığın program tarafından engellenmesi gerekir. Bunun için, her kayıt klavyeden girildikten sonra (dosyaya yazılmadan önce), girilen bilgilerin doğru olup olmadığı bir kez daha kullanıcıya sorulabilir. Bu sayede hatalı bilgi girişi önemli ölçüde önlenmiş olur:

```
10 OPEN"O",1,"STOK"
20 FOR I=1 TO 8
30 CLS
40 INPUT "MAL NO    ";MN
50 INPUT "MAL ADI   ";MA$
60 INPUT "MİKTARI   ";MK
70 INPUT "FİYATI    ";FY
80 INPUT "GİRİLEN BİLGİLER DOĞRU MU (E/H) ";C$
90 IF C$="H" OR C$="h" THEN 30
100 WRITE#1,MN,MA$,MK,FY
110 NEXT I
120 CLOSE 1
```

80 numaralı satırdaki INPUT komutuyla, kullanıcıya girdiği bilgilerin doğru olup olmadığını kontrol etme şansı verilmiştir. Buradaki soruya verilecek cevap 'H' (hayır) ise, bilgi giriş bölümü tekrarlanacaktır; değilse, bilgiler dosyaya yazılacaktır.

2 - Yukarıdaki programda oluşturulan 'STOK' dosyasını okuyarak ekrana listeleyen bir program:

```
10 OPEN"I",1,"STOK"
20 CLS
30 PRINT "MAL NO","MAL ADI","MİKTARI","FİYATI"
40 PRINT "-------","--------","--------","--------"
50 FOR R=1 TO 8
```

```
60 INPUT #1,MN,MA$,MK,FY
70 PRINT MN,MA$,MK,FY
80 NEXT R
90 CLOSE 1
```

Bu programda 60 ve 70 numaralı satırlar, kayıt deseni değiştirilmeden, yalnızca değişken adları değiştirilerek aşağıdaki şekilde de yazılabilir:

```
60 INPUT #1,A,B$,C,D
70 PRINT A,B$,C,D
```

Bu şekilde yazmak, programda herhangi bir hataya yol açmayacaktır. Yalnız, değişken adlarını her keresinde farklı olarak kullanmak, programda karışıklıklara yol açabilir.

3 - Türkçe - İngilizce sözlük olarak kullanılabilecek 'SOZLUK' isimli bir dosyayı aşağıdaki kayıt desenine göre oluşturan bir program:

Türkçe kelime	İngilizce karşılığı

```
10 OPEN"O",1,"SOZLUK"
20 FOR X=1 TO ...
```

Klavyeden bilgi giriş için kullanılan döngü bu şekilde başlatılırsa, döngünün bitiş değerinin de yazılması gerekir. Bunun için, programa başlarken bir sözlükte kaç adet kelime olduğu da bilinmelidir. Klavyeden bilgi girişiyle diskete kayıt yapan bu tür programlarda genellikle girilecek bilgi adedi program yazarken bilinemeyeceği için, başlangıç ve bitiş değerlerinin bilinmesini gerektiren döngülerin (FOR...NEXT veya sayaç) kullanılması sakıncalıdır. (Bunun yerine, sürekli bilgi girişini sağlayabilecek sonsuz bir döngü kullanılmalıdır.) Örneğin, bir fatura programı uygulamasında, bir şirkete kaç adet fatura geleceği program yazılırken bilinemez, veya bir senet projesinde kaç adet senetle işlem yapılacağı kesin olarak tahmin edilemez. Bu

gibi durumlarda, bilgi girişini sonlandırma kararı, programı kullanan kişinin olmalıdır.

Bilgi girişinde sonlandırıcı : Bilindiği gibi, sonlandırıcı, programı bilgi girişini sağlayan sonsuz bir döngüden çıkartmak için kullanılan herhangi bir değerdir. Sonlandırıcının üç önemli özelliği vardır:

1 - Kullanıldığı sahanın türüne (alfabetik veya sayısal) uymalıdır.

2 - Bilgi girişinde girilen ilk bilgi için kullanılmalıdır.

3 - Mantıksal olarak, kullanıldığı saha için hiçbir zaman gerçekleşmeyecek bir bilgi olmalıdır. Örneğin yaş için -1, isim için X, sıcaklık için -99 gibi değerler sonlandırıcı olarak kullanılabilir.. (Çünkü bu değerler hiçbir zaman gerçeğe uygun değildir.)

Bu bilgilerin ışığında, 'SOZLUK' dosyası aşağıdaki programla oluşturulabilir:

```
10 OPEN"O",1,SOZLUK"
20 CLS
30 PRINT "TÜRKÇE KELİME İÇİN 'X' GİRERSENİZ BİLGİ
GİRİŞİ SONA ERER."
40 PRINT
50 INPUT "TÜRKÇE KELİME      ";T$
60 IF T$="X" THEN CLOSE 1:END
70 INPUT "İNGİLİZCE  KARŞILIĞI ";I$
80 PRINT
90 INPUT "GİRİLEN BİLGİLER DOĞRU MU (E/H) ";C$
100 IF C$="E" OR C$="e" THEN WRITE #1,T$,I$
110 GOTO 20
```

Bu programda, sonlandırıcı kullanarak belirsiz sayıda kayıt girilmesi sağlanmıştır. Bilgi girişi, kullanıcı Türkçe kelime olarak 'X' verdiği zaman sona erecek, program 60 numaralı satırdaki CLOSE komutuyla dosyayı kapattıktan sonra duracaktır. 40 ve 80 numaralı satırlardaki PRINT komutları, bilgi girişini ekranda daha düzenli bir şekilde yapabilmek için kullanılmıştır. Aşağıda görüldüğü gibi, her kayıtta, sahalara ait bilgilerin ayrı bir grup olarak belirgin halde görülebilmesi sağlanmıştır.

RUN

```
TÜRKÇE KELİME İÇİN 'X' GİRERSENİZ BİLGİ GİRİŞİ SONA ERER

TÜRKÇE KELİME              ? KİTAP
İNGİLİZCE KARŞILIĞI        ? BOOK
GİRİLEN BİLGİLER DOĞRU MU (E/H) ? E
```

```
TÜRKÇE KELİME İÇİN 'X' GİRERSENİZ BİLGİ GİRİŞİ SONA ERER

TÜRKÇE KELİME              ? ARABA
İNGİLİZCE KARŞILIĞI        ? CAR
GİRİLEN BİLGİLER DOĞRU MU (E/H) ? E
```

```
TÜRKÇE KELİME İÇİN 'X' GİRERSENİZ BİLGİ GİRİŞİ SONA ERER.

TÜRKÇE KELİME              ? KALEM
İNGİLİZCE KARŞILIĞI        ? PENCIL
GİRİLEN BİLGİLER DOĞRU MU (E/H) ? E
```

```
TÜRKÇE KELİME İÇİN 'X' GİRERSENİZ BİLGİ GİRİŞİ SONA ERER.

TÜRKÇE KELİME      ? X
Ok
```

Türkçe kelime olarak 'X' girilene kadar, bütün bilgiler dosyaya kaydedilmektedir. Bu programla, istenilen adette (1, 1000 veya daha fazla) kayıt yapılabilir.

NOT: Program bir kez çalıştırılıp belirli bilgiler girildikten sonra, ikinci bir kez çalıştırıldığında, daha önce aynı dosyada bulunan kayıtlar silinecek ve dosya yeni girilen bilgilerle, yeniden oluşturulacaktır. (OPEN komutu, daha önce var olan bir dosya adıyla birlikte kullanılırsa, bu dosya silinir.)

4 - Yukarıdaki programla oluşturulan 'SOZLUK' isimli dosyayı okuyarak bütün bilgileri ekrana listeleyen bir program:

```
10 OPEN"I",1,"SOZLUK"
20 FOR N=1 TO ...
```

Dosyayı okumak için yazılan bu programda da, dosyadaki kayıt adedi tam olarak bilinemeyeceğinden, FOR...NEXT döngüsü kullanmak sakıncalıdır. Dosyada 10 kayıt olduğu kabul edilirse, döngü '1 TO 9' olarak açıldığında son kayıt okunamayacak; veya, '1 TO 11' şeklinde açıldığında, dosyadaki 10 kayıt okunduktan sonra olmayan bir kayıt okunmak istendiğinden (DATA satırının eksik olması gibi), program bir hata mesajı (EOF met / Input past end) vererek duracaktır.

EOF fonksiyonu (End Of File - dosya sonu):

Herhangi bir dosya okunurken, dosya sonuna gelinip gelinmediğini (dosyadaki bütün kayıtların okunup okunmadığını) kontrol edebilmek için EOF fonksiyonu kullanılır.

Genel kullanım:

EOF(dosya numarası)

NOT: Diğer fonksiyonlarda da olduğu gibi, EOF fonksiyonu da tek başına kullanılamaz; bir komutla birlikte kullanılmak zorundadır.

Giriş değeri dosya numarası olan bu fonksiyonun çıkış değeri, -1 veya 0'dır:

-1 : Okunmakta olan dosyadaki bütün kayıtlar okunmuş, okunacak başka kayıt kalmamıştır.

0 : Dosya sonuna gelinmemiştir. (Okunacak kayıtlar henüz bitmemiştir.)

EOF fonksiyonunu kullanarak, kayıt adedi bilinmeyen 'SOZLUK' dosyasındaki bütün kayıtları ekrana listeleyen program:

```
10 OPEN"I",1,"SOZLUK"
20 CLS
30 PRINT "TÜRKÇE KELİME","İNGİLİZCE KARŞILIĞI"
40 PRINT "--------------","--------------------"
50 IF EOF(1)= -1 THEN CLOSE 1:END
60 INPUT#1,T$,I$
70 PRINT T$,I$
80 GOTO 50
```

Bu programda, 50 numaralı satırda dosya sonuna gelinip gelinmediği kontrol edilmekte, eğer bütün kayıtlar okunmuşsa (EOF = -1) dosya kapatılıp program sona ermektedir. Aksi durumda dosyadan bir kayıt daha okunup ekrana yazılmakta ve program tekrar 50 numaralı satıra dönmektedir.

Aynı program, daha kullanışlı olan WHILE...WEND döngüsüyle de yazılabilir. Burada WHILE teriminden sonra kullanılacak koşul, dosya sonuna gelinmediğini gösteren 'EOF(1)=0' olacaktır.

```
10 OPEN"I",1,"SOZLUK"
20 CLS
30 PRINT "TÜRKÇE KELİME","İNGİLİZCE KARŞILIĞI"
40 PRINT "--------------","--------------------"
50 WHILE EOF(1)=0   (Dosya sonuna gelinmediği sürece)
60 INPUT#1,T$,I$
```

```
70 PRINT T$,I$
80 WEND
90 CLOSE 1
```

Bu programda, dosya okuma işlemi WHILE...WEND döngüsüyle yapılmıştır. EOF fonksiyonu 0 olduğu (dosya sonuna gelinmediği) sürece döngü içindeki INPUT ve PRINT komutları uygulanacak, dosyadan okunan bütün kayıtlar ekrana listelenecektir.

WHILE...WEND döngüsünün temeli olan koşul (EOF(1)=0), NOT terimi de kullanılarak 'NOT EOF(1)' şeklinde de yazılabilir:

```
10 OPEN"I",1,"SOZLUK"
20 CLS
30 PRINT "TÜRKÇE KELİME","İNGİLİZCE KARŞILIĞI"
40 PRINT "----------","----------"
50 WHILE NOT EOF(1)
60 INPUT#1,T$,I$
70 PRINT T$,I$
80 WEND
90 CLOSE 1
```

DOSYALAMA İLE İLGİLİ ÖRNEKLER:

1 - Klavyeden girilen belirsiz sayıda bilgiyi aşağıdaki kayıt desenine göre 'YAS' isimli dosyaya kaydeden bir program:

İSİM	YAŞ

```
10 OPEN"O",1,"YAS"
20 CLS
30 PRINT "BİLGİ GİRİŞİNİ SONLANDIRMAK İÇİN İSİM
OLARAK 'X' GİRİNİZ."
40 PRINT
50 INPUT "İSİM ";I$
60 IF I$="X" THEN CLOSE 1:END
70 INPUT "YAŞ";Y
80 PRINT
```

```
90 INPUT "GİRİLEN BİLGİLER DOĞRU MU (E/H) ";C$
100 IF C$="E" OR C$="e" THEN WRITE#1,I$,Y
110 GOTO 20
```

2 - Yukarıdaki programla oluşturulan 'YAS' isimli dosyada 15 yaşından büyük olan kişilere ait bilgileri listeleyen bir program:

```
10 OPEN"I",1,"YAS"
20 CLS
30 PRINT "İSİM","YAS"
40 PRINT "---------","------"
50 WHILE NOT EOF(1)
60 INPUT#1,A$,Y
70 IF Y>15 THEN PRINT A$,Y
80 WEND
90 CLOSE 1
```

Bu programda, dosyadaki bütün bilgiler değil, yalnızca koşula uyan (15 yaşından büyük) kişilere ait bilgiler bir IF...THEN cümlesi kullanılarak listelenmiştir.

3 - Aynı dosyadan, yalnızca klavyeden adı girilen kişiye ait bilgileri listeleyen bir program:

```
10 OPEN"I",1,"YAS"
20 CLS
30 INPUT "ARANAN KİŞİNİN ADI ";ARA$
40 PRINT "İSİM","YAS"
50 PRINT "---------","------"
60 WHILE NOT EOF(1)
70 INPUT#1,A$,Y
80 IF A$=ARA$ THEN PRINT A$,Y
90 WEND
100 CLOSE 1
```

SPACE$ fonksiyonu:

İstenilen adette (en fazla 255) boşluk karakteri üretmek için kullanılan fonksiyondur.

Genel kullanım:

SPACE$(boşluk adedi)

ÖRNEK:

```
PRINT "İSİM";SPACE$(5);"SOYADI";SPACE$(7);"YAŞ"
İSİM      SOYADI      YAŞ
```

(İsim ve soyadı arasında 5, soyadı ve yaş arasında 7 boşluk vardır.)

```
PRINT "İSİM";SPC(5);"SOYADI";SPC(7);"YAŞ"
İSİM      SOYADI      YAŞ
```

Yukarıdaki satırın aynısı SPC fonksiyonu kullanılarak elde edilmiştir.)

SPC ve SPACE$ fonksiyonları, PRINT komutuyla birlikte kullanıldıklarında tamamen aynı görevi üstlenirler. Aralarındaki fark, SPACE$ fonksiyonuyla üretilen boşlukların bir değişkende saklanabilmesidir.

ÖRNEK:

```
A$="İSİM"+SPACE$(5)+"SOYADI"+SPACE$(7)+"YAŞ"
PRINT A$
İSİM    SOYADI    YAŞ
```

Görüldüğü gibi, SPACE$ fonksiyonuyla elde edilen boşluk dizisi bir değişkende (A$) saklanabilir; ancak, SPC fonksiyonuyla aynı işlemi yapmak bir hata mesajının çıkmasına yol açacaktır.

INSTR fonksiyonu:

Herhangi bir karakter dizisinin içinde istenilen başka bir karakter dizisini aramak için kullanılan fonksiyondur.

Genel kullanım:

INSTR ([başlangıç pozisyonu], <karakter dizisi>, <aranan karakter dizisi>)

Fonksiyonun çıkış değeri, birinci karakter dizisi içinde ikincisinin pozisyonudur; yani, ikincinin birinci içinde kaçıncı karakterden başlayarak yer aldığıdır. İsteğe göre kullanılan başlangıç pozisyonu verilmezse, arama ilk karakterden başlar; verilirse, belirtilen pozisyondan başlar. Eğer birinci karakter dizisi içinde ikinci karakter dizisi yoksa, fonksiyonun çıkış değeri 0'dır.

ÖRNEKLER:

1 - **PRINT INSTR("TİCARET","A")**
4
('A' harfi, 'TİCARET' kelimesinde dördüncü karakterdir.)

2 - **PRINT INSTR("MATEMATİK","A")**
2

('A' harfi, 'MATEMATİK' kelimesinde ikinci karakterdir.)

NOT: Birinci karakter dizisi içinde ikinci karakter dizisi bir kereden çok tekrarlanıyorsa, ilk tekrarın pozisyonu elde edilir.

3 - **PRINT INSTR(3,"MATEMATİK","A")**
6

(Aramaya üçüncü karakterden başlanırsa, 'A' harfi 'MATEMATİK' kelimesinde altıncı pozisyondadır.)

4 - **PRINT INSTR("TELEVİZYON","EV")**
4

('EV' kelimesi, 'TELEVİZYON' kelimesinin içinde dördüncü pozisyondan başlamaktadır.)

5 - **PRINT INSTR("BİLGİSAYAR","Z")**
0

('BİLGİSAYAR' kelimesinde 'Z' harfi yoktur.)

6 - **A$="AYAĞINI YORGANINA GÖRE UZAT"**
B$="YORGAN"
PRINT INSTR(A$,B$)
9

('YORGAN' kelimesi, cümle içinde 9'uncu karakterden başlamaktadır.)

7 - Daha önce yapılmış dosyalama örneklerinden birinde oluşturulmuş olan 'YAS' dosyasını okuyarak, klavyeden ismi girilen kişiye ait bilgileri listeleyen bir program: (Dosyadan okunan isimle klavyeden girilen isim karşılaştırılırken INSTR fonksiyonu kullanılacaktır.)

```
10 OPEN"I",1,"YAS"
20 CLS
30 PRINT "İSİM","YAŞ"
40 PRINT "----------","-----"
50 INPUT "ARANAN KİŞİNİN ADI ";ARA$
60 WHILE NOT EOF(1)
70 INPUT#1,A$,Y
80 IF INSTR(A$,ARA$)>0 THEN PRINT A$,Y
90 WEND
100 CLOSE 1
```

Bu programın daha önce yapılmış programdan yalnızca bir farkı vardır. 80 numaralı satırdaki IF...THEN cümlesinin koşulu 'A$=ARA$' şeklinde değil, 'INSTR(A$,ARA$)>0' şeklinde yazılmıştır. (Bunun anlamı, 'ARA$ kelimesi, A$ kelimesinde varsa' olarak açıklanabilir.) Böylece, aranan ismin bütünüyle yazılması zorunluluğu ortadan kaldırılmış ve programın daha kolay kullanılabilmesi sağlanmıştır. Örneğin, 'ABDULLAH' isminde birinin yaşını öğrenmek için 'AB' yazmak yeterli olacaktır.

```
RUN

  ARANAN KİŞİNİN ADI ? AL

İSİM          YAŞ
----------    -----
ALİ           17
HALİL         39
NİHAL         29
HALE          24
```

Bu örnekte, isim olarak 'AL' karakter dizisi verilmiştir. Böylece isminin içinde 'AL' bulunan bütün kişiler ve yaşları listelenmiştir.

DOSYALAMA VE MENÜ İŞLEMLERİNİN BİRLİKTE UYGULANMASI:

Şu ana kadar yapılmış olan dosyalama programlarının her birinin yalnızca bir görevi vardır: Herhangi bir dosyayı oluşturmak veya var olan bir dosyayı okuyarak listelemek. Gerçek uygulamalarda, bu tür programların ayrı ayrı değil, birlikte (belli bir sıraya göre) kullanılması gerekmektedir. Bunun için, hem dosyanın oluşturulması, hem de okunarak isteğe uygun şekilde listelenmesi için kullanılacak programlar tek bir program içinde birleştirilebilir. Bir menü yardımıyla kullanılan bu tür programlara seçenekli programlar denir.

'PERSONEL' dosyasını oluşturan ve ekrana listeleyen iki program bir menü altında birleştirilecek olursa, menünün seçenekleri şu şekilde olabilir:

1 - Bilgi girişi
2 - Listeleme
3 - Çıkış

1'inci seçenekte, dosya açılacak, klavyeden girilen bilgiler dosyaya yazılarak 'PERSONEL' dosyası oluşturulacak ve program başa dönecektir.

2'nci seçenekte, dosyadaki bilgiler ekrana listelenecek ve program başa dönecektir.

3'üncü seçenekte program sona erecektir.

Daha önce yapılmış olan programların dışında, menüye bir de 'Çıkış' seçeneği eklenmiştir. Çünkü, seçenekli programlarda programın sona ermesi kullanıcının isteğine bırakılmalıdır. Yapılan her işlemde olduuğu gibi, programdan çıkmak için de menüden ilgili seçenek seçilmelidir. Diğer bölümlerde, program belirli bir işlemi tamamladıktan sonra tekrar menüye dönecektir.

```
10 CLS
20 PRINT "  ANA MENÜ"
30 PRINT "--------------------"
```

```
40 PRINT "1 - Bilgi girişi"
50 PRINT "2 - Listeleme"
60 PRINT "3 - Çıkış"
70 PRINT
80 INPUT "Seçiminiz ";S
90 IF S>3 OR S<1 THEN 10
100 ON S GOSUB 200,400,600
110 GOTO 10
200 OPEN"O",1,"PERSONEL"
210 CLS
220 PRINT "MENÜYE DÖNMEK İÇİN İSİM OLARAK 'X'
GİRİNİZ."
230 PRINT "------------------------------------------------------------"
240 INPUT "İSİM      ";I$
250 IF I$="X" OR I$="x" THEN CLOSE 1:RETURN
260 INPUT "SOYADI    ";S$
270 INPUT "SİCİL NO ";SN
280 INPUT "MAA Ş     ";M
290 PRINT
300 INPUT "GİRİLEN BİLGİLER DOĞRU MU (E/H) ";C$
310 IF C$="E" OR C$="e" THEN WRITE#1,I$,S$,SN,M
320 GOTO 210
400 OPEN"I",1,"PERSONEL"
410 CLS
420 PRINT "İSİM","SOYADI","SİCİL NO","MAAŞ"
430 PRINT "--------","--------","-------","-------"
440 WHILE NOT EOF(1)
450 INPUT#1,I$,S$,SN,M
460 PRINT I$,S$,SN,M
470 WEND
480 CLOSE 1
490 FOR B=1 TO 5000:NEXT B
500 RETURN
600 CLS
610 PRINT "PROGRAM BİTMİŞTİR; İYİ GÜNLER !!"
620 END
```

Dosyalamayla ilgili iki program, menü tarafından çağrılan birer alt program olarak kullanılmıştır. Burada ana program, 10 ve 110 numaralı satırlar arasındaki menü bölümüdür.

İlk iki bölümden sonra, programı başa döndürmek için RETURN komutları kullanılmıştır. Bilgi girişi bölümünde daha önceki programlarda kullanılan END komutu yerine bu kez RETURN yazılarak programın burada sona ermesi engellenmiştir.

Önceki programlardan farklı olarak, ayrıca 490 numaralı satırdaki FOR...NEXT döngüsü yazılmıştır. Bunun amacı, program menüye dönmeden önce listenin bir süre ekranda kalmasını ve rahatça okunabilmesini sağlamaktır. Döngünün kullanılmaması durumunda, bilgiler ekrana listelendikten hemen sonra RETURN komutuyla program 110 numaralı satıra dönecek, buradaki GOTO 10 komutuyla 10 numaralı satırdaki CLS komutuyla ekranı silerek menüyü görüntüleyecektir.

490 numaralı satırdaki FOR...NEXT döngüsü, listenin ekranda belirli bir süre kalmaşını sağlamaktadır (bu programda, bilgisayarın 1'den 5000'e kadar sayması). Bu sürenin, kullanıcının isteğine göre ayarlanabilmesi için INKEY$ fonksiyonu kullanılabilir.

INKEY$ fonksiyonu:

Her bilgisayarın belleğinde, klavyeden basılan tuşlarla üretilen karakterlerin saklandığı bir tampon bellek (buffer) vardır. Bu tampon belleğin kapasitesi, kullanılan bilgisayara göre değişir. INKEY$ fonksiyonunun çıkış değeri, tampon bellek bölümüne ilk girmiş olan karakterdir. Fonksiyon kullanıldığında, bu karakter tampondan silinerek daha sonra giren karakterler bir ileri kaydırılır. (Bu arada, karakter ekrana yazılmaz.) Eğer hiçbir tuşa basılmamış ve tampon bellek boşsa, INKEY$ fonksiyonunun çıkış değeri sıfır uzunlukta boş ("") alfabetik karakter dizisidir.

INKEY$ fonksiyonu kullanıldığında, program herhangi bir tuşa basılana kadar beklemez. Eğer programda bir bekleme yaptırılmak isteniyorsa, uygun komutlar yardımıyla belirli bir tuşa basılana dek program geçici bir süre için durdurulabilir; veya, INKEY$ fonksiyonuyla elde edilen karakterlere göre programda kolay kullanılabilen bir dallandırma yapılabilir.

ÖRNEKLER:

490 IF INKEY$ =""" THEN 490

(Tırnak işaretleri arasında hiç boşluk yoktur.)

Bu satır, 'hiçbir tuşa basılmamışsa 490 numaralı satıra git' şeklinde açıklanabilir. Herhangi bir tuşa basıldığında ' INKEY$="" ' koşulu geçersiz olacak ve program bir sonraki satırdan devam edecektir. Bu satır daha önceki programın içinde kullanılırsa, ekrandaki liste herhangi bir tuşa basılana dek bekleyecek, tuşa basıldıktan sonra RETURN komutuyla ana menüye dönecektir.

490 IF INKEY$<> "M" THEN 490

Bu kez, basılan tuş 'M' olmadığı sürece program 490 numaralı satırda bekleyecektir. 'M' (menü) tuşuna basıldığında, bir sonraki satırdaki RETURN komutuyla ana menüye dönecektir. Yalnız, bu satır aynı tuştan hem küçük hem de büyük M harfleri elde edilebildiğinden, büyük M harfi için geçerlidir. Küçük m harfi, ana menüye dönüş için etkisiz olacaktır. Bu sorunu ortadan kaldırabilmek için, CAPS LOCK tuşunun her iki pozisyonunda da aynı karakteri veren ve klavyede kolayca bulunabilen ara (SPACE) tuşu kullanılabilir:

490 IF INKEY$<> " " THEN 490

(Tırnak işaretleri arasında tek boşluk vardır.)

Bu durumda, ara tuşuna basılana dek program bekleyecektir.

NOT: INKEY$ fonksiyonu belirli bir karakter dizisiyle karşılaştırıldığında, (boşluk karakteri, M, vs.) tırnak işaretleri içinde mutlaka tek karakter yazılmalıdır. Çünkü INKEY$ fonksiyonu hiçbir zaman birden fazla karakter içeremez.

Daha önceki örnek, listeyi ekranda bekletebilecek şekilde aşağıdaki gibi yapılabilir:

```
10 CLS
20 PRINT "   ANA MENÜ"
30 PRINT "-----------------------"
```

```
40 PRINT "1 - Bilgi girişi"
50 PRINT "2 - Listeleme"
60 PRINT "3 - Çıkış"
70 PRINT
80 INPUT "Seçiminiz ";S
90 IF S>3 OR S<1 THEN 1Ø
100 ON S GOSUB 200,400,600
110 GOTO 10
200 OPEN"O",1,"PERSONEL"
210 CLS
220 PRINT "MENÜYE DÖNMEK İÇİN İSİM OLARAK 'X'
GİRİNİZ."
230 PRINT "------------------------------------------------------------"
240 INPUT "İSİM      ";I$
250 IF I$="X" OR I$="x" THEN CLOSE 1:RETURN
260 INPUT "SOYADI    ";S$
270 INPUT "SİCİL NO ";SN
280 INPUT "MAAŞ      ";M
290 PRINT
300 INPUT "GİRİLEN BİLGİLER DOĞRU MU (E/H) ";C$
310 IF C$="E" OR C$="e" THEN WRITE#1,I$,S$,SN,M
320 GOTO 210
400 OPEN"I",1,"PERSONEL"
410 CLS
420 PRINT "İSİM","SOYADI","SİCİL NO","MAAŞ"
430 PRINT "--------","--------","-------","-------"
440 WHILE NOT EOF(1)
450 INPUT#1,I$,S$,SN,M
460 PRINT I$,S$,SN,M
470 WEND
480 CLOSE 1
490 IF INKEY$ <>" "THEN 490
500 RETURN
600 CLS
610 PRINT "PROGRAM BİTMİŞTİR; İYİ GÜNLER !!"
620 END
```

Bu programda listeleme bölümü çalıştırıldığında, liste ekrana çıkacak ve ara tuşuna basılana dek bekleyecektir. Ara tuşuna basıldığında ise ekran silinerek ana menü görülecektir. Burada, programı kullanan kişinin proramcıdan başkası olduğu düşünülürse, liste incelendikten sonra ana menüye dönüş için ara tuşuna basılması gerek-

tiği programda belirtilmelidir. Yoksa, kullanıcı ne yapacağını bilemeyeceği için programın çalışması aksayacaktır.

Ara tuşuna basılması gerektiğini belirten mesaj, tuş bekleme satırından önce koyulmalıdır:

```
470 WEND
480 CLOSE 1
485 PRINT:PRINT "ANA MENÜYE DÖNMEK İÇİN ARA
TUŞUNA BASINIZ."
490 IF INKEY$<> " "THEN 490
500 RETURN
```

ÖRNEK:

Aşağıdaki listedeki bilgileri 'MUSTERI' isimli bir dosyaya kaydeden ve menüye uygun işlemler yapan bir program:

MÜŞTERİ NO	ADI	ADRESİ	TELEFON NO	GRUBU
27	MİNE KURT	KARTAL	333 22 11	4
36	MURAT BAL	MALTEPE	357 12 79	1
51	ALİ KISA	BAKIRKÖY	571 24 31	3
64	HASAN UZ	KASIMPAŞA	146 32 87	1
75	SUNA IŞIK	BEBEK	165 23 81	1
85	OYA KUTLU	KADIKÖY	339 20 71	2

Gruplar, müşterilerin ortalama satın alma kapasitelerine göre belirlenmiştir; 1 numaralı gruptaki müşteriler en fazla satın alanlardır.

```
          ANA MENÜ:
-------------------------------------------
    1-'MUSTERI' dosyasına bilgi girişi
    2-Bütün müşterilerin listesi
    3-1'inci gruptaki müşterilerin listesi
    4-İsme göre müşteri arama
    5-Çıkış
```

```
10 CLS
20 PRINT "        ANA MENÜ:"
30 PRINT "-----------------------------------"
```

```
40 PRINT "1 - 'MUSTERI' dosyasına bilgi girişi
50 PRINT "2 - Bütün müşterilerin listesi"
60 PRINT "3 - 1'inci gruptaki müşterilerin listesi"
70 PRINT "4 - İsme göre müşteri arama"
80 PRINT "5 - Çıkış"
90 PRINT
100 INPUT "SEÇİMİNİZ ";S
110 IF S<1 OR S>5 THEN 10
120 ON S GOSUB 200,400,600,800,1000
130 GOTO 10
200 REM ****************** BİLGİ GİRİŞİ *****************
210 OPEN"O",1,"MUSTERI"
220 CLS
230 PRINT "ANA MENÜYE DÖNMEK İÇİN MÜŞTERİ NO
OLARAK -1 VERİNİZ."
240 PRINT "------------------------------------------------------------------------"
250 INPUT "MÜŞTERİ NO     ";MN
260 IF MN=-1 THEN CLOSE 1:RETURN
270 INPUT "ADI            ";AD$
280 INPUT "ADRESİ         ";ADR$
290 INPUT "TELEFON NO     ";TEL$
300 INPUT "GRUBU          ";GR
310 PRINT
320 INPUT "GİRİLEN BİLGİLER DOĞRU MU (E/H) ";C$
330 IF C$="E" OR C$="e" THEN WRITE#1, MN, AD$, ADR$,
TEL$, GR
340 GOTO 220
400 REM ******** BÜTÜN MÜŞTERİLERİN LİSTESİ *********
410 OPEN"I",1,"MUSTERI"
420 CLS
430 PRINT "MÜŞTERİ NO","ADI","ADRESİ","TELEFON NO",
"GRUBU"
440 PRINT "----------","----------","----------","----------","------"
450 WHILE NOT EOF(1)
460 INPUT#1,MN,AD$,ADR$,TEL$,GR
470 PRINT MN,AD$,ADR$,TEL$,GR
480 WEND
490 CLOSE 1
500 PRINT
510 PRINT "ANA MENÜYE DÖNMEK İÇİN ARA TUŞUNA
BASINIZ."
```

```
520 IF INKEY$<>" " THEN 520
530 RETURN
600 REM **** 1'İNCİ GRUPTAKİ MÜŞTERİLERİN LİSTESİ ***
610 OPEN "I",1,"MUSTERI"
620 CLS
630 PRINT "   BİRİNCİ GRUPTAKİ MÜŞTERİLERİN LİSTESİ"
640 PRINT "-------------------------------------------------------------------"
650 PRINT "MÜŞTERİ NO","ADI","ADRESİ","TELEFON NO",
"GRUBU"
660 PRINT "----------","----------","----------","----------","-------"
670 WHILE NOT EOF(1)
680 INPUT#1,MN,AD$,ADR$,TEL$,GR
690 IF GR=1 THEN PRINT MN,AD$,ADR$,TEL$,GR
700 WEND
710 CLOSE 1
720 PRINT
730 PRINT "ANA MENÜYE DÖNMEK İÇİN ARA TUŞUNA
BASINIZ."
740 IF INKEY$<> " " THEN 740
750 RETURN
800 REM ********** İSME GÖRE LİSTELEME **********
810 OPEN"I",1,"MUSTERI"
820 CLS
830 INPUT "ARANAN MÜŞTERİNİN ADI ";ARA$
840 PRINT "-----------------------------------"
850 PRINT "MÜŞTERİ NO","ADI","ADRESİ","TELEFON NO",
"GRUBU"
860 PRINT "----------","----------","----------","----------","-------"
870 WHILE NOT EOF(1)
880 INPUT#1,MN,AD$,ADR$,TEL$,GR
890 IF AD$=ARA$ THEN PRINT MN,AD$,ADR$,TEL$,GR
900 WEND
910 CLOSE 1
920 PRINT
930 PRINT "ANA MENÜYE DÖNMEK İÇİN ARA TUŞUNA
BASINIZ."
940 IF INKEY$<> " " THEN 940
950 RETURN
1000 REM ****************** ÇIKIŞ *****************
1010 CLS
1020 PRINT "İYİ GÜNLER."
1030 END
```

Bu programda, ayrı bölümlerin içeriklerinin kolayca anlaşılabilmesi amacıyla REM satırları koyulmuştur.

890 numaralı IF...THEN satırındaki koşul 'AD$=ARA$' yerine ' INSTR(AD$,ARA$)>Ø ' şeklinde yazılırsa, programın kullanımı daha kolay olacaktır. Daha önce de belirtildiği gibi, bu durumda aranılan bir ismin bütününü yazmak gerekmeyecek, yalnızca bu ismin bir bölümünü girmek yeterli olacaktır.

SIRALI ERİŞİMLİ DOSYALARIN GÜNCELLEŞMESİ:

I- SIRALI ERİŞİMLİ BİR DOSYAYA KAYIT EKLEME:

Şu ana kadar yapılan dosyalama programlarında bilgi girişi bölümleri birden çok kez çalıştırıldığında, her keresinde bir önceki dosya silinecek, en son girilen bilgiler dosyada kalacaktır. Bu nedenle, bu programlarla istenilen herhangi bir dosyaya yeni kayıt eklemek mümkün olamayacaktır.

Bir dosyaya kayıt eklemek, ancak özel bir yöntemle yapılabilir. Bu yönteme göre önce var olan dosyadaki bilgiler farklı bir isme sahip geçici bir dosyaya kopyalanır; eklenecek kayıtlar bu geçici dosyaya eklenir. Bu işlemlerden sonra, disket üzerinde iki dosya oluşacaktır: Birincisi kayıt eklenmek istenen dosya, ikincisi kayıtların eklenmiş olduğu, fakat ismi değişik olan geçici dosya. Sonuç olarak isteneni elde edebilmek için birinci dosya silinir; geçici dosyanın ismi asıl dosya ismi olarak değiştirilir. Artık diskette, eklenmek istenen bilgileri de kapsayan asıl dosya oluşmuştur.

'MUSTERI' dosyasına kayıt eklemek amacıyla yukarıdaki yönteme uygun bir program:

```
10 OPEN"I",1,"MUSTERI"
20 OPEN"O",2,"YEDEK"
30 WHILE NOT EOF(1)
40 INPUT#1,MN,AD$,ADR$,TEL$,GR
50 WRITE#2,MN,AD$,ADR$,TEL$,GR
60 WEND
70 CLOSE 1
80 CLS
```

```
90 PRINT "KAYIT EKLEMEYİ SONLANDIRMAK İÇİN MÜŞTERİ
NO'YU -1 GİRİNİZ."
100 PRINT "------------------------------------------------------------"
110 INPUT "MÜŞTERİ NO     ";MN
120 IF MN=-1 THEN CLOSE 2:KILL"MUSTERI":NAME"
YEDEK" AS "MUSTERI":END
130 INPUT "ADI            ";AD$
140 INPUT "ADRESİ         ";ADR$
150 INPUT "TELEFON NO     ";TEL$
160 INPUT "GRUBU          ";GR
170 PRINT
180 INPUT "GİRİLEN BİLGİLER DOĞRU MU (E/H) ";C$
190 IF C$="E" OR C$="e" THEN  WRITE # 2, MN, AD$,
ADR$, TEL$, GR
200 GOTO 80
```

Bu programda, 'MUSTERI' dosyası okunarak tüm kayıtlar 'YEDEK' isimli geçici dosyaya kopyalanmaktadır. Bunun için, programın başında iki OPEN komutuyla 1 numaralı dosya (MUSTERI) okunmak için, 2 numaralı dosya (YEDEK) bilgi kaydetmek için açılmıştır. Daha sonra, birinci dosyanın sonuna kadar (WHILE NOT EOF(1)) bütün kayıtlar okunarak (INPUT#1,...) ikinci dosyaya yazılmıştır (WRITE#2,...). Böylece birinci dosyanın bütün bilgileri ikinciye kopyalanmış ve 1 numaralı dosya kapatılmıştır. Kayıt yapmak üzere açılmış ikinci dosya halen açıktır (eklenmek istenen kayıtlar bu dosyaya yazılabilir).

Daha sonra, klavyeden girilen bilgiler önceki programlarda da olduğu gibi ikinci dosyaya yazılmış (WRITE#2,...) sonlandırıcı verildiğinde başka kayıt eklenmeyeceği için bu dosya kapatılmıştır.

Bu işlemler tamamlandıktan sonra daha önce de belirtildiği gibi, diskette programla ilgili iki dosya vardır:

1 - Eski kayıtları içeren 'MUSTERI' isimli asıl dosya,

2 - Eski kayıtlarla birlikte yeni eklenen kayıtları da içeren 'YEDEK' isimli geçici dosya.

'YEDEK' isimli dosya, sonuç olarak oluşturulmak istenen dosyadır, yalnız ismi 'MUSTERI' olmadığı için henüz asıl dosya olarak kullanılamaz (çünkü program 'MUSTERI' isimli dosya üzerine kurulmuştur). Bu nedenle 'YEDEK' ismi 'MUSTERI' olarak değiştirilmelidir. Bunu yapabilmek için de disket üzerinde 'MUSTERI' isimli bir dosya bulunmamalıdır. 120 numaları satırda da görüldüğü gibi, 'MUSTERI' isimli dosya silinmiş (KILL"MUSTERI") ve 'YEDEK' ismi 'MUSTERI' olarak değiştirilmiştir (NAME "YEDEK" AS "MUSTERI"). Böylece, yeni kayıtlar 'MUSTERI' dosyasına eklenmiştir.

NOT: Birçok BASIC yorumlayıcısında, sıralı erişimli bir dosyaya kayıt eklemek için OPEN komutunun değişik bir biçimde kullanılması yeterlidir:

```
OPEN "A",<dosya numarası>,"dosya adı"
                    veya
OPEN "dosya adı" FOR APPEND AS <dosya numarası">
```

Bu komut verildikten sonra, normal uygulamayla (kopyalama işlemine gerek kalmadan) yeni girilen bilgiler dosyaya eklenebilir. (APPEND : ekleme)

ÖRNEK: Bir şirketin masraflarını 'MASRAF' isimli bir dosyaya kaydeden, bu bilgileri listeleyerek toplam masrafı hesaplayan ve kayıtlara yenilerini ekleyebilen bir program:

MASRAFLAR

TUTAR	AÇIKLAMA
50000	TAMİRAT
32000	YEMEK
14000	SERVİS
26000	ISINMA
120000	ELEKTRİK
60000	SU
140000	TELEFON

```
                ANA MENÜ:
-----------------------------------
   1 - Bilgi girişi (dosya oluşturma)
   2 - Listeleme ve toplam masrafı hesaplama
   3 - Dosyaya yeni kayıt ekleme
   4 - Çıkış
```

```
10 CLS
20 PRINT "          ANA MENÜ:"
30 PRINT "----------------------------------------"
40 PRINT "1 - Bilgi giriş (dosya oluşturma)"
50 PRINT "2 - Listeleme ve toplam masrafı hesaplama"
60 PRINT "3 - Dosyaya yeni kayıt ekleme"
70 PRINT "4 - Çıkış"
80 PRINT
90 INPUT "SEÇİMİNİZ ";S
100 IF S<1 OR S>4 THEN 10
110 ON S GOSUB 200,400,600,800
120 GOTO 10
200 REM ***************** BİLGİ GİRİŞ *****************
210 OPEN"O",1,"MASRAF"
220 CLS
230 PRINT "ANA MENÜYE DÖNÜŞ İÇİN MASRAF OLARAK -
1 VERİNİZ."
240 PRINT "------------------------------------------------------------------------"
250 INPUT "MASRAF   ";MS
260 IF MS=-1 THEN CLOSE 1:RETURN
270 INPUT "AÇIKLAMA ";AC$
280 PRINT
290 INPUT "GİRİLEN BİLGİLER DOĞRU MU (E/H) ";EH$
300 IF EH$="E" OR EH$="e" THEN WRITE#1,MS,AC$
310 GOTO 220
400 REM ***************** LİSTELEME *****************
410 OPEN"I",1,"MASRAF"
420 CLS
430 PRINT "          MASRAFLAR"
440 PRINT "----------------------------------"
450 PRINT "TUTAR","AÇIKLAMA"
460 PRINT "--------","-----------"
470 TOP=0
480 WHILE NOT EOF(1)
```

```
490 INPUT#1,MS,AC$
500 PRINT MS,AC$
510 TOP=TOP+MS
520 WEND
530 CLOSE 1
540 PRINT "-----------------------------------"
550 PRINT "TOPLAM MASRAF :";TOP
560 PRINT
570 PRINT "ANA MENÜYE DÖNMEK İÇİN ARA TUŞUNA
BASINIZ."
580 IF INKEY$<>" " THEN 580
590 RETURN
600 REM ********** KAYIT EKLEME **********
610 OPEN"I",1,"MASRAF"
620 OPEN"O",2,"YEDEK"
630 WHILE NOT EOF(1)
640 INPUT#1,MS,AC$
650 WRITE#2,MS,AC$
660 WEND
670 CLOSE 1
680 CLS
690 PRINT "ANA MENÜYE DÖNÜŞ İÇİN MASRAF OLARAK
-1 VERİNİZ."
700 PRINT "-----------------------------------------"
710 INPUT "MASRAF   ";MS
720 IF MS=-1 THEN CLOSE 2:KILL"MASRAF":NAME"YEDEK"
AS "MASRAF":RETURN
730 INPUT "AÇIKLAMA ";AC$
740 PRINT
750 INPUT "GİRİLEN BİLGİLER DOĞRU MU (E/H) ";EH$
760 IF EH$="E" OR EH$="e" THEN WRITE#2,MS,AC$
770 GOTO 680
800 REM ********** ÇIKIŞ **********
810 PRINT "PROGRAM BİTMİŞTİR."
820 END
```

Bu program çalıştırıldığında önce birinci seçenek (Bilgi giriş ve dosya oluşturma) seçilerek dosya oluşturulmalıdır. Listeleme bölümü istenilen herhangi bir zamanda çalıştırılarak dosyanın son durumu

ekrana listelenebilir. Kayıt ekleme yapıldıktan sonra da listeleme seçeneğine girilerek kayıtların eklenip eklenmediği kontrol edilebilir.

Bu programda, kayıtlar girildikten sonra birinci seçenek tekrar çalıştırılırsa, daha önce girilmiş olan kayıtlar silinir; yeni yazılan bilgiler 'MASRAF' dosyasını oluşturur. Bu nedenle bu bölümü çalıştırırken dikkatli olmalıdır.

OPEN"A" komutu kullanılarak kayıt ekleme bölümü aşağıdaki şekilde yazılabilir:

```
600 REM ********** KAYIT EKLEME **********
610 OPEN"A",1,"MASRAF"
620 CLS
630 PRINT "ANA MENÜYE DÖNÜŞ İÇİN MASRAF OLARAK
-1 VERİNİZ."
640 PRINT "----------------------------------------"
650 INPUT "MASRAF  ";MS
660 IF MS=-1 THEN CLOSE 1:RETURN
670 INPUT "AÇIKLAMA ";AC$
680 PRINT
690 INPUT "GİRİLEN BİLGİLER DOĞRU MU (E/H) ";EH$
700 IF EH$="E" OR EH$="e" THEN WRITE#1,MS,AC$
710 GOTO 620
```

Görüldüğü gibi, en baştaki bilgi giriş bölümünden farklı olan yalnızca 610 numaralı satırdaki OPEN komutunun kullanılışıdır. Bu alt program çalıştırıldığında, istenilen bilgiler dosyaya eklenebilecektir. Yalnız, daha önce de belirtildiği gibi, bu yöntem bütün BASIC yorumlayıcılarında kullanılamamaktadır.

DOSYAYA KAYIT EKLEME İÇİN İKİNCİ BİR YÖNTEM:

Özellikle küçük dosyalar için kullanılabilecek bu yöntemde, dosyadaki bütün bilgiler uygun dizilere aktarılmakta, kayıt ekleme işlemi dizi üzerinde yapıldıktan sonra dizilerdeki kayıtlar yeniden dosyaya yazılmaktadır. Böylece kayıt ekleme işlemi gerçekleştirilmektedir. Bu yöntemin sakıncası, belleğin gereksiz yere işgal edilmesi, hatta büyük dosyaların bellek kapasitesini aşması ve bu tür dosyaların saklanabilmesi için belleğin yetersiz kalmasıdır.

Bu yönteme göre, yukarıdaki programın kayıt ekleme bölümü şu şekilde yazılabilir:

```
5 DIM MS(300),AC$(300)  (Diziler başta uygun boyutlarda
.                        tanımlanmalıdır.
.
.
600 REM ***************** KAYIT EKLEME ***************
610 OPEN"I",1,"MASRAF"
620 S=0
630 WHILE NOT EOF(1)
640 S=S+1
650 INPUT#1,MS(S),AC$(S)
660 WEND
670 CLOSE 1
680 CLS
690 PRINT "ANA MENÜYE DÖNÜŞ İÇİN MASRAF OLARAK
-1 VERİNİZ."
700 PRINT "-----------------------------------------"
710 INPUT "MASRAF   ";M
720 IF MS=-1 THEN 780
730 INPUT "AÇIKLAMA ";A$
740 PRINT
750 INPUT "GİRİLEN BİLGİLER DOĞRU MU (E/H) ";EH$
760 IF EH$="E" OR EH$="e" THEN S=S+1: MS(S)=
M:AC$(S)= A$
770 GOTO 680
780 OPEN"O",1,"MASRAF"
782 FOR X=1 TO S
784 WRITE#1,MS(X),AC$(X)
786 NEXT X
788 CLOSE 1
790 RETURN
```

Bu programda, önce dosyadaki bütün kayıtlar uygun dizilere bir sayaç yardımıyla kopyalanmaktadır. Daha sonra klavyeden girilen bilgiler aynı sayacın arttırılmasıyla dizilerdeki bilgilerin sonuna eklenmektedir. Eklenecek kayıtlar tamamlandığında, dosya tekrar açılarak, bir FOR...NEXT döngüsüyle diziler dosyaya aktarılmaktadır. Böylece aynı dosya üzerine yeni bilgiler eklenmiş olmaktadır.

II - SIRALI ERİŞİMLİ BİR DOSYADAN KAYIT SİLME:

Sıralı erişimli bir dosyadaki kayıtlardan herhangi birinin silinmesi istendiğinde, bunun doğrudan doğruya dosyadan çıkarılması mümkün değildir; bu nedenle, kayıt eklemede de olduğu gibi özel bir yöntem kullanılmalıdır.

Bu yönteme göre, asıl dosya geçici dosyaya kopyalanır; bu sırada silinmek istenen kayıt atlanır. Böylece geçici dosyada silinmek istenen bilgiler bulunmaz. Eklemede olduğu gibi asıl dosya silinir ve geçici dosyanın ismi asıl dosya ismi olarak değiştirilir. Böylece istenen kayıtlar dosyadan çıkarılmış olur.

'MUSTERI' dosyasından, klavyeden ismi girilen herhangi bir müşterinin kaydını silen bir program:

```
10 CLS
20 INPUT "KAYDI SİLİNECEK MÜŞTERİNİN ADI ";SIL$
30 OPEN"I",1,"MUSTERI"
40 OPEN"O",2,"YEDEK"
50 WHILE NOT EOF(1)
60 INPUT#1,MN,AD$,ADR$,TEL$,GR
70 IF AD$<>SIL$ THEN WRITE#2,MN,AD$,ADR$,TEL$,GR
80 WEND
90 CLOSE 1,2
100 KILL"MUSTERI"
110 NAME"YEDEK" AS "MUSTERI"
```

Bu programda, dosyadan silinecek müşterinin adı en başta verilmekte, daha sonra 'MUSTERI' dosyasından 'YEDEK' isimli dosyaya, silinmek istenen kayıt dışındaki (IF AD$<> SIL$) bütün kayıtlar kopyalanmaktadır. Bu durumda, 'MUSTERI' dosyasında çıkarılmak istenenler dahil tüm kayıtlar, 'YEDEK' isimli dosyada ise bunlar haricindeki kayıtlar bulunmaktadır. 'MUSTERI' dosyası silinip, 'YEDEK' dosyasının ismi de 'MUSTERI' olarak değiştirildiğinde (100 ve 110 numaralı satırlar), asıl dosyadan istenen kayıtlar silinmiş olacaktır.

ÖRNEK: Daha önce yapılmış kayıt ekleme bölümünü de içeren, aynı zamanda kayıt silme, listeleme seçenekleriyle 'MUSTERI' dosyası üzerinde işlem yapan bir program:

```
          ANA MENÜ:
----------------------------------
  1 - Bilgi giriş
  2 - Bütün müşterilerin listesi
  3 - Kayıt ekleme
  4 - Kayıt silme
  5 - Çıkış
```

```
10 CLS
20 PRINT"   ANA MENÜ"
30 PRINT"----------------------------"
40 PRINT"1 - Bilgi giriş"
50 PRINT"2 - Bütün müşterilerin listesi"
60 PRINT"3 - Kayıt ekleme"
70 PRINT"4 - Kayıt silme"
80 PRINT"5 - Çıkış"
90 PRINT
100 INPUT"SEÇİMİNİZ ";S
110 IF S<1 OR S>5 THEN 10
120 ON S GOSUB 200,400,600,800,1000
130 GOTO 10
200 REM ****************** BİLGİ GİRİŞ *****************
210 OPEN"O",1,"MUSTERI"
220 CLS
230 PRINT "ANA MENÜYE DÖNMEK İÇİN MÜŞTERİ NO
OLARAK -1 VERİNİZ."
240 PRINT "------------------------------------------------"
250 INPUT "MÜŞTERİ NO     ";MN
260 IF MN=-1 THEN CLOSE 1:RETURN
270 INPUT "ADI            ";AD$
280 INPUT "ADRESİ         ";ADR$
290 INPUT "TELEFON NO     ";TEL$
300 INPUT "GRUBU          ";GR
310 PRINT
320 INPUT "GİRİLEN BİLGİLER DOĞRU MU (E/H) ";C$
330 IF C$="E" OR C$="e" THEN WRITE#1, MN, AD$, ADR$,
TEL$,GR
340 GOTO 220
400 REM ******** BÜTÜN MÜŞTERİLERİN LİSTESİ ********
410 OPEN"I",1,"MUSTERI"
420 CLS
```

```
430 PRINT "MÜŞTERİ NO","ADI","ADRESİ","TELEFON NO",
"GRUBU"
440 PRINT "----------","----------","----------","----------","------"
450 WHILE NOT EOF(1)
460 INPUT#1,MN,AD$,ADR$,TEL$,GR
470 PRINT MN,AD$,ADR$,TEL$,GR
480 WEND
490 CLOSE 1
500 PRINT
510 PRINT "ANA MENÜYE DÖNMEK İÇİN ARA TUŞUNA
BASINIZ."
520 IF INKEY$ <>" " THEN 520
530 RETURN
600 REM ***************** KAYIT EKLEME ***************
610 OPEN"I",1,"MUSTERI"
620 OPEN"O",2,"YEDEK"
630 WHILE NOT EOF(1)
640 INPUT#1,MN,AD$,ADR$,TEL$,GR
650 WRITE#2,MN,AD$,ADR$,TEL$,GR
660 WEND
670 CLOSE 1
680 CLS
690 PRINT "KAYIT EKLEMEYİ SONLANDIRMAK İÇİN
MÜŞTERİ NO'YU -1 GİRİNİZ."
700 PRINT "------------------------------------------------------"
710 INPUT "MÜŞTERİ NO     ";MN
720 IF MN=-1 THEN CLOSE 2:KILL"MUSTERI":NAME"
YEDEK"  AS "MUSTERI":RETURN
730 INPUT "ADI                ";AD$
740 INPUT "ADRESİ             ";ADR$
750 INPUT "TELEFON NO     ";TEL$
760 INPUT "GRUBU              ";GR
770 PRINT
780 INPUT "GİRİLEN BİLGİLER DOĞRU MU (E/H) ";C$
790 IF C$="E" OR C$="e" THEN WRITE#2, MN, AD$, ADR$,
TEL$, GR
795 GOTO 680
800 REM ***************** KAYIT SİLME *****************
810 CLS
820 INPUT "KAYDI SİLİNECEK MÜŞTERİNİN ADI ";SIL$
```

```
830 OPEN"I",1,"MUSTERI"
840 OPEN"O",2,"YEDEK"
850 WHILE NOT EOF(1)
860 INPUT#1,MN,AD$,ADR$,TEL$,GR
870 IF AD$<>SIL$ THEN WRITE#2,MN,AD$,ADR$,TEL$,GR
880 WEND
890 CLOSE 1,2
900 KILL"MUSTERI"
910 NAME"YEDEK" AS "MUSTERI"
920 RETURN
1000 REM ********** ÇIKIŞ **********
1010 PRINT "PROGRAM SONU..."
1020 END
```

Dikkat edilecek olursa, 1'inci ve 3'üncü seçenekler, çalıştırıldıklarında ekran görüntüsü olarak aynı sonucu vermektedir. Daha önce de belirtildiği gibi, 1'inci seçenek (Bilgi giriş) bir kereden fazla çalıştırıldığında, o anda dosyada bulunan bilgiler silinecek ve en son girilenlerle dosya yeniden oluşacaktır. Bu, normalde istenen bir durum değildir, çünkü dosyanın sürekli olarak yeniden yaratılması, birçok durumda sakıncalı olacaktır. Eğer silinmek istenen bilgiler varsa, 'Kayıt silme' bölümü çalıştırılabilir. Bu nedenle, 'Bilgi giriş' bölümü dosyayı oluşturmak amacıyla yalnızca bir kez çalıştırılacak olduğundan, menüde yer alması gereksizdir.

'Bilgi giriş (dosya oluşturma)' bölümü programdan çıkarıldığında, dosyaya kayıt yapmak mümkün olmayacaktır, çünkü dosya oluşturulamamış olacaktır. Bunun nedeni, OPEN"O",1,"MUSTERI" komutunun hiç kullanılmamış olmasıdır. Kayıt ekleme bölümünün de çalıştırılamamasının nedeni, bu bölümde ekleme yapılacak dosyanın bulunamayışıdır. Bilgisayar, OPEN"I",1,"MUSTERI" komutuyla karşılaştığı zaman bu dosya diskette bulunmadığı için 'File not found' hata mesajını vererek duracaktır.

Dosya oluşturma bölümü programdan çıkarıldığında, program çalıştırılmadan önce, dosyanın oluşturulabilmesi amacıyla (yalnızca bir kere için) satır numarasız olarak şu komutlar yazılabilir:

```
OPEN"O",1,"MUSTERI"
CLOSE 1
```

Bu komutlar verildiğinde, 'MUSTERI' dosyası boş olarak oluşturulacaktır. İçinde hiçbir kayıt bulunmayan bu dosyaya, programdaki 'Kayıt ekleme' bölümüyle istenilen bilgiler kaydedilebilir. Bu durumda, ana menüdeki seçenekler aşağıdaki gibi olabilir:

```
                ANA MENÜ
-----------------------------------
   1 - Bilgi giriş (yeni kayıt ekleme)
   2 - Bütün müşterilerin listesi
   3 - Kayıt silme
   4 - Çıkış
```

```
10 CLS
20 PRINT"     ANA MENÜ"
30 PRINT"----------------------------"
40 PRINT"1 - Bilgi giriş"
50 PRINT"2 - Bütün müşterilerin listesi"
60 PRINT"4 - Kayıt silme"
70 PRINT"5 - Çıkış"
80 PRINT
90 INPUT"SEÇİMİNİZ ";S
100 IF S<1 OR S>5 THEN 10
110 ON S GOSUB 200,400,600,800
120 GOTO 10
200 REM ***************** BİLGİ GİRİŞ ******************
210 OPEN"I",1,"MUSTERI"
220 OPEN"O",2,"YEDEK"
230 WHILE NOT EOF(1)
240 INPUT#1,MN,AD$,ADR$,TEL$,GR
250 WRITE#2,MN,AD$,ADR$,TEL$,GR
260 WEND
270 CLOSE 1
280 CLS
290 PRINT "KAYIT EKLEMEYİ SONLANDIRMAK İÇİN
MÜŞTERİ NO'YU -1 GİRİNİZ."
300 PRINT "------------------------------------------------------"
310 INPUT "MÜŞTERİ NO ";MN
320 IF MN=-1 THEN CLOSE 2:KILL"MUSTERI":NAME"
YEDEK" AS "MUSTERI":RETURN
330 INPUT "ADI                ";AD$
```

```
340 INPUT "ADRESİ           ";ADR$
350 INPUT "TELEFON NO       ";TEL$
360 INPUT "GRUBU            ";GR
370 PRINT
380 INPUT "GİRİLEN BİLGİLER DOĞRU MU (E/H) ";C$
390 IF C$="E" OR C$="e" THEN WRITE#2, MN, AD$, ADR$,
TEL$,GR
395 GOTO 280
400 REM ********* BÜTÜN MÜŞTERİLERİN LİSTESİ ********
410 OPEN"I",1,"MUSTERI"
420 CLS
430 PRINT "MÜŞTERİ NO","ADI","ADRESİ","TELEFON NO",
"GRUBU"
440 PRINT "----------","----------","----------","----------","-------"
450 WHILE NOT EOF(1)
460 INPUT#1,MN,AD$,ADR$,TEL$,GR
470 PRINT MN,AD$,ADR$,TEL$,GR
480 WEND
490 CLOSE 1
500 PRINT
510 PRINT "ANA MENÜYE DÖNMEK İÇİN ARA TUŞUNA
BASINIZ."
520 IF INKEY$<>" " THEN 520
530 RETURN
600 REM ***************** KAYIT SİLME ******************
610 CLS
620 INPUT "KAYDI SİLİNECEK MÜŞTERİNİN ADI ";SIL$
630 OPEN"I",1,"MUSTERI"
640 OPEN"O",2,"YEDEK"
650 WHILE NOT EOF(1)
660 INPUT#1,MN,AD$,ADR$,TEL$,GR
670 IF AD$<>SIL$ THEN WRITE#2,MN,AD$,ADR$,TEL$,GR
680 WEND
690 CLOSE 1,2
700 KILL"MUSTERI"
710 NAME"YEDEK" AS "MUSTERI"
720 RETURN
800 REM ******************** ÇIKIŞ **********************
810 PRINT "PROGRAM SONU..."
820  END
```

III - SIRALI ERİŞİMLİ BİR DOSYADAKİ KAYITLARDA SAHA DEĞİŞİKLİĞİ:

Dosyaya yanlış olarak kaydedilmiş veya daha sonra değişmesi gereken bazı kayıtların bütünüyle silinmesi değil, kısmen değiştirilmesi gerekebilir. Bu durumda, yalnızca değişiklik yapılacak saha düzeltilerek dosya yeniden oluşturulmalıdır. Bunun için, kullanılan yöntem şu şekildedir:

Diğer güncelleme yöntemlerinde de olduğu gibi, asıl dosya geçici bir dosyaya kopyalanır. Bu kopyalama sırasında, yenilenmek istenen kaydın ilgili sahası değiştirilerek geçici dosyaya aktarılır. Bütün kayıtlar kopyalandıktan sonra asıl dosya silinir; geçici dosyanın ismi değiştirilir.

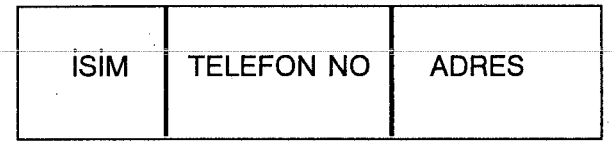
'MUSTERI' doyasındaki alım güçlerinde değişiklik olan müşterilerin grup numaralarında değişiklik yapan bir program:

```
10 CLS
20 INPUT "GRUP NUMARASI DEĞİŞEN MÜŞTERİNİN ADI
";DM$
30 INPUT "YENİ GRUP NUMARASI ";YG
40 OPEN"I",1,"MUSTERI"
50 OPEN"O",2,"YEDEK"
60 WHILE NOT EOF(1)
70 INPUT#1,MN,AD$,ADR$,TEL$,GR
80 IF AD$=DM$ THEN GR=YG
90 WRITE#2,MN,AD$,ADR$,TEL$,GR
100 WEND
110 CLOSE 1,2
120 KILL"MUSTERI"
130 NAME"YEDEK" AS "MUSTERI"
```

Grup numarası değişecek müşterinin adı ve yeni grup numarası en başta girilmektedir. Daha sonra asıl dosya geçici dosyaya kopyalanırken girilen bu isimle karşılaşılırsa, bu kişiye ait grup numarası değiştirilmektedir (80 numaralı satır). Kopyalama bittikten sonra asıl dosya silinmekte, geçici dosyanın adı da 'MUSTERI' olarak değiştirilmektedir.

ÖRNEK: Bir telefon rehberi programı:

Kayıt deseni:



Dosya adı: REHBER

ANA MENÜ

1 - Bilgi kayıt
2 - Genel liste
3 - İsme göre kayıt arama
4 - Adrese göre kayıt arama
5 - Telefon numarası değiştirme
6 - Adres değiştirme
7 - Kayıt silme
8 - Çıkış

```
10 CLS
20 PRINT"    ANA MENÜ"
30 PRINT"-------------------------"
40 PRINT"1 - Bilgi kayıt"
50 PRINT"2 - Genel liste"
60 PRINT"3 - İsme göre liste"
70 PRINT"4 - Adrese göre kayıt arama"
80 PRINT"5 - Telefon numarası değiştirme"
90 PRINT"6 - Adres değiştirme"
100 PRINT"7 - Kayıt silme"
110 PRINT"8 - Çıkış"
120 PRINT
130 INPUT"SEÇİMİNİZ ";S
140 IF S<1 OR S>8 THEN 10
150 ON S GOSUB 500,1000,1500,2000,2500,3000,3500,4000
160 GOTO 10
500 REM ********** BİLGİ GİRİŞ **********
```

```
510 OPEN"I",1,"REHBER"
520 OPEN"O",2,"REHBERX"
530 WHILE NOT EOF(1)
540 INPUT#1,AD$,TEL$,ADR$
550 WRITE#2,AD$,TEL$,ADR$
560 WEND
570 CLOSE 1
580 CLS
590 PRINT"MENÜYE DÖNMEK İÇİN İSİM OLARAK 'X'
VERİNİZ.
600 PRINT" ==========================
610 INPUT"İSİM       ";AD$
620 IF AD$="X" THEN CLOSE 2:KILL"REHBER":NAME"
REHBERX" AS "REHBER":RETURN
630 INPUT"TELEFON NO ";TEL$
640 LINE INPUT"ADRES     ";ADR$
650 PRINT
660 INPUT"GİRİLEN BİLGİLER DOĞRU MU (E/H) ";CEV$
670 IF CEV$="E" OR CEV$="e" THEN WRITE#2, AD$, TEL$,
ADR$
680 GOTO 580
1000 REM ****************** GENEL LİSTE ***************
1010 OPEN"I",1,"REHBER"
1020 CLS
1030 PRINT"İSİM","TELEFON NO","ADRES"
1040 PRINT"---------","-----------","--------------------------"
1050 WHILE NOT EOF(1)
1060 INPUT#1,AD$,TEL$,ADR$
1070 PRINT AD$,TEL$,ADR$
1080 WEND
1090 CLOSE 1
1100 PRINT
1110 PRINT"ANA MENÜYE DÖNMEK İÇİN ARA TUŞUNA
BASINIZ."
1120 IF INKEY$<>" "THEN 1120
1130 RETURN
1500 REM *********** İSME GÖRE KAYIT ARAMA *********
1510 CLS
1520 F=0
1530 INPUT"KAYDI ARANAN KİŞİNİN İSMİ ";ARA$
```

```
1540 PRINT"İSİM","TELEFON NO","ADRES"
1550 PRINT"---------","----------","--------------------------"
1560 OPEN"I",1,"REHBER"
1570 WHILE NOT EOF(1)
1580 INPUT#1,AD$,TEL$,ADR$
1590 IF INSTR(AD$,ARA$)>Ø THEN PRINT AD$,TEL$,
ADR$ : F=1
1600 WEND
1610 CLOSE 1
1620 IF F=0 THEN PRINT "ARANAN KAYIT DOSYADA
BULUNAMADI."
1630 PRINT
1640 PRINT "ANA MENÜYE DÖNMEK İÇİN ARA TUŞUNA
BASINIZ."
1650 IF INKEY$<>" "THEN 1650
1660 RETURN
2000 REM ********** ADRESE GÖRE KAYIT ARAMA *******
2010  CLS
2020  F=0
2030  INPUT"KAYDI ARANAN KİŞİNİN ADRESİ ";ARA$
2040  PRINT"İSİM","TELEFON NO","ADRES"
2050 PRINT"---------","----------","--------------------------"
2060  OPEN"I",1,"REHBER"
2070  WHILE NOT EOF(1)
2080  INPUT#1,AD$,TEL$,ADR$
2090 IF INSTR(ADR$,ARA$)>Ø THEN PRINT AD$,TEL$,
ADR$ : F=1
2100 WEND
2110 CLOSE 1
2120 IF F=0 THEN PRINT"ARANAN KAYIT DOSYADA
BULUNAMADI."
2130 PRINT
2140 PRINT "ANA MENÜYE DÖNMEK İÇİN ARA TUŞUNA
BASINIZ."
2150 IF INKEY$<>" "THEN 2150
2160 RETURN
2500 REM ***** TELEFON NUMARASI DEĞİŞTİRME *******
2510 CLS
2520 INPUT"TELEFON NUMARASI DEĞİŞEN KİŞİNİN ESKİ
TELEFON NUMARASI ";ET$
```

```
2530 INPUT"YENİ TELEFON NUMARASI ";YT$
2540 OPEN"I",1,"REHBER"
2550 OPEN"O",2,"REHBERX"
2560 WHILE NOT EOF(1)
2570 INPUT#1,AD$,TEL$,ADR$
2580 IF TEL$=ET$ THEN TEL$=YT$
2590 WRITE#2,AD$,TEL$,ADR$
2600 WEND
2610 CLOSE 1,2
2620 KILL"REHBER"
2630 NAME"REHBERX" AS "REHBER"
2640 RETURN
3000 REM ************** ADRES DEĞİŞTİRME ************
3010 CLS
3020 INPUT"ADRESİ DEĞİŞEN KİŞİNİN ADI ";A1$
3030 INPUT"YENİ ADRESİ ";YA$
3040 OPEN"I",1,"REHBER"
3050 OPEN"O",2,"REHBERX"
3060 WHILE NOT EOF(1)
3070 INPUT#1,AD$,TEL$,ADR$
3080 IF AD$=A1$ THEN ADR$=YA$
3090 WRITE#2,AD$,TEL$,ADR$
3100 WEND
3110 CLOSE 1,2
3120 KILL"REHBER"
3130 NAME"REHBERX" AS "REHBER"
3140 RETURN
3500 REM *************** KAYIT SİLME ******************
3510 CLS
3520 INPUT"KAYDI SİLİNECEK KİŞİNİN ADI ";SIL$
3530 OPEN"I",1,"REHBER"
3540 OPEN"O",2,"REHBERX"
3550 WHILE NOT EOF(1)
3560 INPUT#1,AD$,TEL$,ADR$
3570 IF AD$<>SIL$ THEN WRITE#2,AD$,TEL$,ADR$
3580 WEND
3590 CLOSE 1,2
3600 KILL"REHBER"
3610 NAME"REHBERX" AS "REHBER"
3620 RETURN
```

```
4000 REM ********** ÇIKIŞ **********
4010 PRINT"REHBER PROGRAMI BİTMİŞTİR."
4020 END
```

Bu programın çalışabilmesi için önce 'REHBER' dosyası oluşturulmalıdır. Bunun için, daha önce de yapıldığı gibi, aşağıdaki satırlar yazılabilir:

```
OPEN"O",1,"REHBER"
CLOSE 1
```

Programdaki kayıt arama bölümlerinde, daha önceki programlardan farklı olarak, aranan kaydın bulunamaması durumunda 'ARANAN KAYIT DOSYADA BULUNAMADI.' mesajı eklenmiştir. Bunun yapılabilmesi için, dosya okuma döngüsünden önce programla ilgili olmayan herhangi bir değişken sıfıra, dosya okunurken kayıt bulunduğunda aynı değişken sıfırdan farklı bir değere (burada 1'e) eşitlenmiştir. Tüm dosyanın okunma işlemi tamamlandığında, bu değişkene bakılarak aranan kaydın bulunup bulunmadığı anlaşılabilir. Eğer değişken 1'e eşitse kayıt bulunmuş demektir; 0'sa bulunamamıştır; ilgili bir mesaj yazılmalıdır.

Yukarıdaki programda, telefon ve adres değişiklikleri için ayrı bölümler yazılmıştır. Özellikle çok sayıda saha içeren kayıtlarda değişiklik yapmak, her saha için ayrı bir bölüm yazılmasını gerektireceğinden, bu tür bir uygulama kullanışlı değildir. Bunun yerine, bütün sahaların birlikte değiştirilebileceği bir bölüm düşünülebilir.

ÖRNEK: Türkiyedeki şehirlere ait çeşitli bilgilerin saklandığı 'SEHIR' isimli sıralı erişimli bir dosyayla ilgili işlemleri aşağıdaki menüye uygun bir şekilde yapan bir program:

ANA MENÜ:

1 - Bilgi giriş
2 - Listeleme
3 - Bilgi değiştirme
4 - Çıkış

Kayıt deseni:

ŞEHİR ADI	TRAFİK KODU	YÜZÖLÇÜMÜ	NÜFUSU	TARIM ÜRÜN.	MADENLERİ

```
10 CLS
20 PRINT " ANA MENÜ"
30 PRINT "--------------------"
40 PRINT "1 - Bilgi giriş"
50 PRINT "2 - Listeleme"
60 PRINT "3 - Bilgi değiştirme"
70 PRINT "4 - Çıkış"
80 PRINT
90 INPUT "SEÇİMİNİZ ";S
100 IF S>4 OR S<1 THEN 10
110 ON S GOSUB 200,400,600,1000
120 GOTO 10
200 REM ***************** BİLGİ GİRİŞ ******************
210 OPEN"O",1,"SEHIR"
220 CLS
230 PRINT "MENÜYE DÖNMEK İÇİN ŞEHİR ADI OLARAK
'XX' VERİNİZ."
240 INPUT "ŞEHİR ADI          : ",S$
250 IF S$="XX" OR S$="xx" THEN CLOSE 1:RETURN
260 INPUT "TRAFİK KODU        : ",K
270 INPUT "YÜZÖLÇÜMÜ          : ",Y
280 INPUT "NÜFUSU             : ",N
290 LINE INPUT "TARIM ÜRÜNLERİ : ",T$
300 LINE INPUT "MADENLERİ      : ",M$
310 PRINT
320 INPUT "GİRİLEN BİLGİLER DOĞRU MU (E/H) ";EH$
330 IF EH$="E" OR EH$="e" THEN WRITE#1, S$, K, Y$, N,
T$, M$
340 GOTO 220
400 REM ****************** LİSTELEME ******************
410 CLS
420 OPEN"I",1,"SEHIR"
430 WHILE NOT EOF(1)
```

```
440 INPUT#1,S$,K,Y,N,T$,M$
450 PRINT "ŞEHİR ADI          : ";S$
460 PRINT "TRAFİK KODU        : ";K
470 PRINT "YÜZÖLÇÜMÜ          :";Y
480 PRINT "NÜFUSU             :";N
490 PRINT "TARIM ÜRÜNLERİ     : ";T$
500 PRINT "MADENLERİ          : ";M$
510 PRINT "-------------------------------------------------"
520 PRINT "DEVAM ETMEK İÇİN ARA TUŞUNA BASINIZ."
530 IF INKEY$<>" "THEN 530
540 WEND
550 CLOSE 1
560 RETURN
600 REM ************ BİLGİ DEĞİŞTİRME ************
610 CLS
620 INPUT "BİLGİLERİ DEĞİŞECEK ŞEHRİN ADI ";DS$
630 PRINT "-------------------------------------------------"
640 OPEN"I",1,"SEHIR"
650 OPEN"O",2,"SEHIRX"
660 WHILE NOT EOF(1)
670 INPUT#1,S$,K,Y,N,T$,M$
680 IF S$<>DS$ THEN 900
690 PRINT "ŞEHİR ADI          : ";S$
700 PRINT "TRAFİK KODU        : ";K
710 PRINT "YÜZÖLÇÜMÜ          :";Y
720 PRINT "NÜFUSU             :";N
730 PRINT "TARIM ÜRÜNLERİ     : ";T$
740 PRINT "MADENLERİ          : ";M$
750 PRINT "-------------------------------------------------"
760 PRINT " ****** DEĞİŞMEYECEK BİLGİLER İÇİN *******"
770 PRINT " HİÇBİR ŞEY YAZMADAN ENTER TUŞUNA
BASINIZ"
780 INPUT "ŞEHRİN YENİ ADI  ";YS$
790 INPUT "TRAFİK KODU      ";YK
800 INPUT "YÜZÖLÇÜMÜ        ";YY
810 INPUT "NÜFUSU           ";YN
820 LINE INPUT "TARIM ÜRÜNLERİ  ";YT$
830 LINE INPUT "MADENLERİ       ";YM$
840 IF YS$<>"" THEN S$=YS$
850 IF YK<>Ø THEN K=YK
```

```
860 IF YY<>Ø THEN Y=YY
870 IF YN<>Ø THEN N=YN
880 IF YT$<>"" THEN T$=YT$
890 IF YM$<>"" THEN M$=YM$
900 WRITE#2,S$,K,Y,N,T$,M$
910 WEND
920 CLOSE 1,2
930 KILL "SEHIR"
940 NAME "SEHIRX" AS "SEHIR"
950 RETURN
1000 REM ********** ÇIKIŞ **********
1010 CLS
1020 PRINT "PROGRAM SONA ERMİŞTİR."
1030 PRINT "      İYİ GÜNLER."
1040 END
```

Bilgi girişinde, tarım ürünleri ve madenler için INPUT yerine LINE INPUT komutu kullanılmıştır. Bunun nedeni, bu bilgiler girilirken aralarında virgül kullanılma olasılığının yüksek olmasıdır. Örnek: Buğday, arpa, yulaf; Bakır, demir...

Bilgi değiştirme amacıyla, her saha için ayrı bir alt program yazılmamıştır. 600 ve 950 numaralı satırlar arasında bütün sahalardaki bilgilerin birlikte değiştirilebileceği bir tek bölüm yer almaktadır. Bu bölümde, değişecek kayıt için bütün sahalar klavyeden girilmekte ve dosyaya değiştirilmiş olarak yazılmaktadır. Klavyeden giriş sırasında hiçbir şey yazılmadan ENTER tuşuna basılırsa, o sahadaki eski bilgi saklanacak ve böylelikle, yalnızca değiştirilmek istenen sahalardaki bilgiler yenilenmiş olacaktır. Bunun sağlanabilmesi için, 840 ve 890 numaralı satırlar arasındaki IF...THEN cümlelerinden yararlanılmıştır. Herhangi bir saha için hiçbir bilgi yazılmadan ENTER tuşuna basılırsa, girilen değer sayısal bilgiler için 0, alfabetik bilgiler için "" olacaktır. Bundan yararlanarak, sahalardaki eski bilgiler, ancak yeni bilgiler girildiyse değiştirilmekte, giriş yapılmamışsa IF...THEN sayesinde eski şekilleriyle saklanmaktadır.

Bu programda listeleme bölümünde bilgiler sütunlar halinde değil, her saha için ayrı bir satır ayrılacak şekilde alt alta ekrana yazılmaktadır. Bunun nedeni, kayıtlardaki bilgilerden bazılarının (tarım ürünleri ve madenler) bir ekran satırına sığmayacak kadar uzun olmalarıdır. Bu tür listelemede, bütün dosyadaki bilgiler listelendikten

sonra değil, her kayıt ekrana çıktıktan sonra ara tuşuna basılması beklenmektedir. Böylece ekranın kayması sonucu listenin okunamaması engellenmiş olmaktadır.

SORULAR:

Aşağıda ana menüleri ve kayıt desenleri verilmiş olan konularla ilgili programları yapınız.

1) ANA MENÜ:

1 - 'FILM' dosyasına bilgi girişi
2 - Bütün bilgilerin listelenmesi
3 - Oyuncu adına göre film listesi
4 - Yapım yılına göre film listesi
5 - Çıkış

Kayıt deseni:

Film adı	Oyuncular	Yönetmen	Yapım yılı	Süresi	Dili

2) ANA MENÜ:

1 - 'GEZEGEN' dosyasına bilgi girişi
2 - Bütün bilgilerin listelenmesi
3 - Gezegen adına göre bilgilerin listesi
4 - Uydu sayısına göre gezegen adlarının listesi
5 - Çıkış

Kayıt deseni:

Gezegen adı	Uydu sayısı	Yüzölçümü	Dünyaya uzaklığı	Güneşe uzaklığı

RASGELE ERİŞİMLİ DOSYALAMA YÖNTEMİ (RANDOM ACCESS)

Rasgele (doğrudan) erişimli dosyalama yöntemi, sıralı erişimli dosyalara oranla daha uzun işlem adımı gerektirmesine rağmen, istenilen bilgiye istenildiği anda ulaşılması yönünden çok daha kullanışlıdır. Rasgele erişimli bir dosyaya gerek bilgi kaydederken, gerek okurken, belirli bir sırayı izleme zorunluluğu yoktur.

Rasgele erişimli dosyalama yönteminin özellikleri:

— Bütün sahalar (sayısal bilgiler içerseler de) alfabetik olmalıdır.
— Kayıtlardaki her sahanın uzunluğu (karakter sayısı) en baştan (FIELD komutuyla) belirlenmeli ve daha sonra da bu düzen bozulmamalıdır.
— Dosya açma modu yalnızca bir tanedir ("R" modu). Dosya bir kez açıldıktan sonra, istenirse bu dosyaya kayıt yazılabilir veya bilgiler okunabilir. Okumak veya kayıt yazmak için ayrı OPEN komutlarına gerek yoktur.
— Dosyanın temeli, kayıt numaralarıdır; kayıtlar dosyaya yazılırken veya okunurken kayıt (sıra) numaralarıyla erişim sağlanır.
— Dosya sonunu belirten EOF fonksiyonunun rasgele dosyalarla kullanılması sakıncalıdır; bazı hatalara yol açabilir. Bunun yerine başka bir yöntem kullanılmaktadır.

Rasgele erişimli bir dosyaya bilgilerin kaydedilmesi:

1 - Dosya "R" modunda açılır. Bunun için, sıralı erişimli dosyalarda da geçerli olan OPEN komutu kullanılır.

Genel kullanım:

OPEN"R",<dosya numarası>,"dosya adı",[kayıt uzunluğu]
veya
OPEN"dosya adı" AS <dosya numarası>, [LEN = kayıt uzunluğu]

NOT: Kayıt uzunluğu belirtilmezse, 128 olarak kabul edilir.

2 - Dosya açıldıktan sonra, FIELD (saha) komutuyla kayıt deseni belirlenir.

Genel kullanım:

FIELD <dosya no>,<saha uzunluğu-1> AS <saha adı-1,>
<saha uzunluğu-2> AS <saha adı-2,>
<saha uzunluğu-3> AS <saha adı-3,>
.....

FIELD komutunun üç görevi vardır:
a) Saha adlarını,
b) Saha uzunluklarını,
c) Bir kayıtta kaç adet saha olduğunu belirlemek.

Dosya numarası, OPEN komutuyla dosyaya verilmiş olan numaradır. Saha uzunluğu-1, kayıt desenindeki birinci sahanın uzunluğudur (karakter sayısı); saha adı-1, birinci saha adıdır.

NOT: FIELD komutuyla tanımlanan bütün saha adları alfabetik olmalıdır; saha uzunluklarının toplamı, OPEN komutuyla belirtilen saha uzunluğuna eşit olmalıdır.

3 - Dosyaya kaydedilmek istenen bilgiler ait oldukları sahalara yerleştirilir. Bunun için LSET ve RSET komutları kullanılır. (Bu arada sayısal bilgiler de özel fonksiyonlarla alfabetik bilgilere dönüştürülür.)

Genel kullanım:

LSET <saha adı> = <sahaya yerleştirilecek alfabetik bilgi>
RSET <saha adı> = <sahaya yerleştirilecek alfabetik bilgi>

LSET komutu, alfabetik bilgiyi sahanın sol tarafından başlayarak yerleştirir. Eğer bilgi sahadan daha kısaysa, sağ tarafta boşluk kalır; daha uzunsa fazla karakterler geçersiz sayılır.

RSET komutu da, aynı LSET gibi çalışır; tek farkı bilgileri sahaya sağ taraftan başlayarak yerleştirmesidir.

ÖRNEKLER:

```
FIELD #1,5 AS A$

LSET A$="ALİ"
PRINT A$
ALİ
```


```
FIELD #1,5 AS A$

RSET A$="ALİ"
PRINT A$
  ALİ
```

```
LSET A$="MURAT"          | RSET A$="MURAT"
PRINT A$                 | PRINT A$
MURAT                    | MURAT

LSET A$="HANDAN"         | RSET A$="HANDAN"
PRINT A$                 | PRINT A$
HANDA                    | ANDAN
```

Sahanın sol veya sağ tarafından başlayarak değil de, ortasında bir yere bilgi yerleştirilmek istenirse, MID$ kullanılabilir:

```
FIELD #1,5 AS A$

MID$(A$,2,3)="ALİ"
PRINT A$
 ALİ
```

Yalnız, MID$ fonksiyonunu bu şekilde kullanırken özellikle tecrübesiz programcıların çok dikkatli olması, hatta bu kullanımdan kaçınmaları gerekir.

> NOT: Tanımlanmış olan sahalara istenen bilgileri yerleştirmek için LSET, RSET ve MID$ komutlarından herhangi birini kullanmak zorunludur. Doğrudan eşitleme yoluyla veya INPUT, READ komutlarıyla bu işlem yapılamaz.

4 - Kaydedilmek istenen sayılar alfabetik bilgilere dönüştürülmelidir. Bunun için, daha önceki matris örneklerinde olduğu gibi, STR$ fonksiyonu kullanılabilir. Daha sonra dosyadan okunan bu karakter dizileri, VAL fonksiyonuyla sayısal bilgilere dönüştürülebilirler. Yalnız, özellikle büyük sayılar kaydedildiğinde, STR$ fonksiyonunun uygun olarak kullanılabilmesi için kayıt deseninde sayıların saklandığı sahalara gereğinden fazla karakter ayırmak zorunluluğu doğmaktadır. Örneğin, 150000 sayısı için 7 basamak (bir karakter + veya - için) tanımlamak gerekmektedir. Sayıların rasgele dosyalara kaydedilirken

özel bir biçimde kodlanarak daha az yer kaplamaları için CHR$, MKI$, MKS$, MKD$ fonksiyonları kullanılabilir.

CHR$: 0'dan 255'e kadar tamsayıları 1 karakter olarak kodlamak için kullanılabilir. Bu fonksiyonla kodlanacak sayılar için kayıt deseninde 1 karakterlik yer ayrılmalıdır.

MKI$: -32767'den 32767'ye kadar tamsayıları 2 karakter olarak kodlamak için kullanılır. Bu fonksiyonla kodlanacak sayılar için kayıt deseninde 2 karakterlik yer ayrılmalıdır.

MKS$: Tek duyarlıklı (en fazla 7 basamaklı) gerçel sayıları 4 karakter olarak kodlamak için kullanılır.

MKD$: Çift duyarlıklı (en fazla 16 basamaklı) gerçel sayıları 8 karakter olarak kodlamak için kullanılır.

ÖRNEKLER:

```
.....
A=150000
LSET MAAS$=MKS$(A)
.....
.....
Y=32
LSET YAS$=MKI$(Y)
.....
.....
TOP=6280398267394
LSET TOPLAM$=MKD$(TOP)
```

5 - Tampon bölyeye yerleştirilmiş olan bilgiler dosyaya yazılır. Bunun için, PUT komutu kullanılır.

Genel kullanım:

PUT <dosya numarası>,[kayıt numarası]

PUT komutundan sonra saha adları yazılmaz.Bu komuttan sonra yalnızca iki parametre kullanılabilir. Birincisi, bilgilerin hangi dosyaya yazılacağını gösteren dosya numarası, ikincisi ise dosyaya yazılan kaydın sıra numarasıdır. Bu sıra numarası kullanılmazsa, en son yazılmış olan kayıttan bir sonraki sıra numarası kabul edilir. Sıra numarasının alabileceği en küçük değer 1, en büyük değer ise 32767'dir.

6 - Dosya sıralı erişimli dosyalarda da olduğu gibi CLOSE komutuyla kapatılır.

Rasgele erişimli bir dosyadaki bilgilerin okunması:

1 - Dosya, 'R' modunda OPEN komutuyla açılır.

2 - FIELD komutuyla sahalar tanımlanır.

NOT: Dosya daha önceden bilgi kaydetmek amacıyla açılmış ve henüz kapatılmamışsa, dosyayı okumak için ayrıca bir kez daha OPEN ve FIELD komutlarının kullanılması gereksizdir. Eğer dosya kapatılmışsa, OPEN ve FIELD komutları, dosyayı oluştururken kullanıldığı şekilde yazılmalıdır. (Saha adedi ve uzunlukları aynı olmalıdır; saha adları değişebilir.)

3 - Dosyadaki kayıtlar tampon sahaya taşınır. Bunun için GET komutu kullanılır.

Genel kullanım:

GET <dosya numarası>, [kayıt numarası]

PUT komutunda da olduğu gibi, GET ile birlikte kayıt numarası kullanmak zorunlu değildir; eğer bu parametre kullanılmazsa en son okunmuş kayıttan bir sonraki okunacaktır.

4 - Sayısal bilgiler içeren sahalardaki kod olarak kaydedilmiş bilgiler ASC, CVI, CVS veya CVD fonksiyonlarıyla sayısal değerlere dönüştürülür.

ASC: CHR$ fonksiyonuyla kaydedilmiş olan karakterin kod numarasını verir.

CVI: MKI$ fonksiyonuyla kodlanarak 2 karakter uzunluğundaki alfabetik bilgiye dönüştürülmüş olan sayıyı verir.

CVS: MKS$ fonksiyonuyla kodlanarak 4 karakter uzunluğundaki alfabetik bilgiye dönüştürülmüş olan sayıyı verir.

CVD: MKD$ fonksiyonuyla kodlanarak 8 karakter uzunluğundaki alfabetik bilgiye dönüştürülmüş olan sayıyı verir.

5 - Sahalardaki bilgiler isteğe uygun bir biçimde kullanılır. (Listeleme, hesap yapma, belirli bir bilginin aranması, vs.) Bu kullanımda, FIELD satırındaki saha adları yazılmalıdır.

6 - İşlemler bittikten sonra dosya CLOSE komutuyla kapatılır.

KODLAYAN FONKSİYON	GİRİŞ DEĞERİ SINIRLARI	KODUN KARAKTER UZUNLUĞU	SAYIYA DÖNÜŞTÜREN FONKSİYON
CHR$	0 <--> 255	1	ASC
MKI$	-32767 <--> 32767	2	CVI
MKS$	7 basamaklı	4	CVS
MKD$	16 basamaklı	8	CVD

ÖRNEKLER:

1 - Aşağıdaki kayıt desenine göre klavyeden girilen beş mala ait bilgileri 'MAL' isimli rasgele erişimli bir dosyaya kaydeden bir program:

MAL ADI (MAL$) 20	MİKTARI (MİKTAR$) 4	FİYATI (FİYAT$) 4

Kullanılan dosya rasgele erişimli olduğu için, saha uzunlukları ve saha adları programı yazmaya başlamadan belirlenmiştir. Görüldüğü gibi bütün sahalar alfabetik olarak tanımlanmıştır. Ayrıca, sayısal bilgilere ait sahaların uzunlukları, sayıların alabilecekleri en büyük değerler düşünülerek belirlenmiştir. Miktarlar ve fiyatlar 32767'den fazla olabilecekleri için, MKI$ fonksiyonuyla kodlanamazlar; MKS$ fonksiyonu kullanılmalıdır. Bu nedenle her iki saha için de 4'er karakterlik alan ayrılmıştır. (Bu durumda miktar ve fiyat için girilebilecek en yüksek değer, 9999999 olacaktır. Daha büyük değerler E notasyonuyla yazılacaktır.)

Bütün programcıların, rasgele erişimli dosyalarla çalışırken, kayıt deseni ayrıntılarını bir kâğıt üzerine yazarak program çalışması süresince bu şemadan yararlanmaları yapılacak hataları azaltacaktır.

```
10 OPEN"R",1,"MAL",28
20 FIELD#1,20 AS MAL$,4 AS MIKTAR$,4 AS FIYAT$
30 FOR X=1 TO 5
40 INPUT "MAL ADI          ";M$
50 INPUT "MİKTARI          ";MKT
60 INPUT "FİYATI           ";FYT
70 LSET MAL$=M$
80 LSET MIKTAR$=MKS$(MKT)
90 LSET FIYAT$=MKS$(FYT)
100 PUT 1,X
110 NEXT X
120 CLOSE 1
```

Klavyeden girilen bilgiler, görüldüğü gibi, LSET komutuyla, FIELD satırında tanımlanan sahalara yerleştirilmiştir. Bu arada, sayısal bilgiler MKS$ fonksiyonuyla 4 basamaklı alfabetik koda dönüştürülmüştür. 100 numaralı satırdaki PUT komutundan sonra kullanılan 1'den 5'e kadar artan X değişkeni, oluşturulan kaydın sıra numarasını göstermektedir.

2 - Yukarıdaki örnekte oluşturulan 'MAL' isimli rasgele dosyadaki beş kaydı okuyup, tutarları hesaplayarak ekrana listeleyen bir program:

```
10 OPEN"R",1,"MAL",28
20 FIELD#1,20 AS MAL$,4 AS MIKTAR$,4 AS FIYAT$
30 PRINT"MAL ADI MİKTARI","FİYATI","TUTARI"
40 PRINT"---------- ----------","----------","----------"
50 FOR X=1 TO 5
60 GET 1,X
70 TUTAR=CVS(MIKTAR$) * CVS(FIYAT$)
80 PRINT MAL$;CVS(MIKTAR$),CVS(FIYAT$),TUTAR
90 NEXT X
100 CLOSE 1
```

70 numaralı satırdaki tutar hesaplaması, MIKTAR$ ve FIYAT$ sahalarının CVS fonksiyonuyla sayıya dönüştürülmesiyle yapılmaktadır. 80 numaralı satırdaki PRINT cümlesinde, alfabetik bilgiler içeren saha adları (FIELD satırında tanımlanan saha adları) olduğu gibi, sayısal bilgiler içeren saha adları ise CVS fonksiyonuyla sayıya dönüştürülerek yazılmaktadır. Burada MAL$'dan sonra (;) kullanılmasının nedeni, MAL$ sahasının uzunluğunun her zaman 20 karakter olmasıdır. Ayrıca virgül koyulursa, mal adlarından sonra miktarlar çok uzakta yazılacaktır. PRINT cümlesinin diğer elemanları arasında virgül kullanılmasının nedeni ise, bunların sayısal bilgiler olmaları ve uzunluklarının sabit olmamasıdır.

3 - Aynı dosyaya, kayıt numaları klavyeden girilerek bilgileri kaydeden, kayıt numarası olarak 0 verilirse (sonlandırıcı olarak) duran, aksi durumda sürekli olarak devam eden bir program:

```
10 OPEN"R",1,"MAL",28
20 FIELD#1,20 AS MAL$,4 AS MIKTAR$,4 AS FIYAT$
30 CLS
40 PRINT "BİLGİ GİRİŞİNİ SONLANDIRMAK İÇİN KAYIT NO
OLARAK  0 VERİNİZ."
50 INPUT "KAYIT NO ";X
60 IF X=0 THEN CLOSE 1:END
70 INPUT "MAL ADI          ";M$
```

```
80 INPUT "MİKTARI          ";MKT
90 INPUT "FİYATI           ";FYT
100 LSET MAL$=M$
110 LSET MIKTAR$=MKS$(MKT)
120 LSET FIYAT$=MKS$(FYT)
130 PUT 1,X
140 GOTO 30
```

Kayıt numarası için sonlandırıcı olarak 0 sayısının seçilmesinin nedeni, kayıt numarasının hiçbir zaman 1'den küçük bir sayı olamayacağıdır.

RUN

```
BİLGİ GİRİŞİNİ SONLANDIRMAK İÇİN KAYIT NO OLARAK 0
VERİNİZ.
KAYIT NO      ? 1
MAL ADI       ? ÇİVİ
MİKTARI       ? 40000
FİYATI        ? 10
```

```
BİLGİ GİRİŞİNİ SONLANDIRMAK İÇİN KAYIT NO OLARAK 0
VERİNİZ.
KAYIT NO      ? 2
MAL ADI       ? LAVABO
MİKTARI       ? 200
FİYATI        ? 50000
```

```
BİLGİ GİRİŞİNİ SONLANDIRMAK İÇİN KAYIT NO OLARAK 0
VERİNİZ.
KAYIT NO      ? 4
MAL ADI       ? TERMOSİFON
MİKTARI       ? 4
FİYATI        ? 300000
```

```
BİLGİ GİRİŞİNİ SONLANDIRMAK İÇİN KAYIT NO OLARAK 0
VERİNİZ.
KAYIT NO      ? 3
MAL ADI       ? FAYANS
MİKTARI       ? 20000
FİYATI        ? 1250
```

```
BİLGİ GİRİŞİNİ SONLANDIRMAK İÇİN KAYIT NO OLARAK 0
VERİNİZ.
KAYIT NO      ? 0
Ok
```

Dikkat edilirse, kayıtlar 1, 2, 4, 3 sırasında verilmiştir. Rasgele erişimli dosya kullanıldığından, kayıtların veriliş sırası önemli değildir. Hatta, aynı kayıt numarası bir kez daha verilerek bu kayıt üzerindeki bilgiler yenilenebilir. Örneğin, 3 numarayla kaydedilmiş olan fayans fiyatlarında değişiklik yapılmak istenirse program yeniden çalıştırılabilir:

RUN

```
BİLGİ GİRİŞİNİ SONLANDIRMAK İÇİN KAYIT NO OLARAK 0
VERİNİZ.
KAYIT NO      ? 3
MAL ADI       ? FAYANS
MİKTARI       ? 20000
FİYATI        ? 1400
```

```
BİLGİ GİRİŞİNİ SONLANDIRMAK İÇİN KAYIT NO OLARAK 0
VERİNİZ.
KAYIT NO      ? 0
Ok
```

Sonlandırıcı kullanılarak kayıt yapılmış bir dosyanın listelenmesi için, dosyada kaç kayıt olduğunun bilinmesi gerekir. Sıralı dosyalarda olduğu gibi dosya sonuna kadar okumak için kullanılan EOF fonksiyonu rasgele dosyalarda hatalı sonuçlar verebileceği için kullanılamaz. Bu nedenle, dosyadaki kayıt adedi, herhangi bir kayıt ortamında saklanmalıdır.

Kayıt adedi, kalıcı olabilmesi amacıyla bilgisayarın belleğindeki bir değişkende değil, disket üzerinde saklanmalıdır. Bütün bilgiler için olduğu gibi, bir tek sayı dahi olsa, bu işlem için bir dosya oluşturulmalıdır.

Kayıt adedinin (dosyada kaç adet kayıt olduğu) disket üzerinde saklanabilmesi için şu işlemler yapılmalıdır:

A - Rasgele erişimli dosyanın bulunduğu disket üzerinde, bu dosyanın içeriğini hatırlatabilecek bir isimle (fakat kesinlikle başka bir isim) sıralı erişimli bir dosya açılmalıdır:

```
OPEN"O",1,"MAL1"
```

B - Başlangıç olarak, bu dosyaya 0 sayısı yazılmalı ve dosya kapatılmalıdır.

```
WRITE#1,0
CLOSE 1
```

(Bu satırlar program içinde değil, satır numarasız olarak yazılacaktır.)

Böylece, 'MAL' dosyasında kaç adet kayıt olduğunu gösteren (şu anda 0) 'MAL1' dosyası oluşturulmuştur.

Daha sonra, rasgele erişimli dosyayı kullanan programın en başında bu adet 'MAL1' dosyasından okunacak, ve programa dosyada kaç kayıt olduğu bildirilmiş olacaktır. Bunun için,program içinde şu satırlar yazılabilir:

```
OPEN"I",2,"MAL1"
INPUT#2,ADET
CLOSE 2
```

Böylece, 'MAL' dosyasında kaç kayıt olduğu, programın başında ADET değişkeninde saklanacaktır. En başta 0 olarak belirlenen kayıt adedi, bilgi giriş sırasında rasgele dosyadaki kayıt adedindeki değişikliğe göre yenilenmelidir:

```
10 OPEN"R",1,"MAL",28
20 FIELD#1,20 AS MAL$,4 AS MIKTAR$,4 AS FIYAT$
30 OPEN"I",2,"MAL1"
40 INPUT#2,ADET
50 CLOSE 2
60 CLS
70 PRINT "DOSYADAKİ KAYIT ADEDİ :";ADET
80 PRINT "BİLGİ GİRİŞİNİ SONLANDIRMAK İÇİN KAYIT NO
OLARAK 0 VERİNİZ."
90 INPUT "KAYIT NO          ";X
100 IF X=0 THEN CLOSE 1:END
110 INPUT "MAL ADI          ";M$
120 INPUT "MİKTARI          ";MKT
130 INPUT "FİYATI           ";FYT
```

```
140 LSET MAL$=M$
150 LSET MIKTAR$=MKS$(MKT)
160 LSET FIYAT$=MKS$(FYT)
170 PUT 1,X
180 IF X >ADET THEN OPEN"O",2,"MAL1" : WRITE#2,X :
ADET=X : CLOSE 2
190 GOTO 60
```

Programın başında rasgele erişimli 'MAL' dosyası açıldıktan sonra, bu dosyanın içinde kaç kayıt olduğunu belirleyebilmek amacıyla sıralı erişimli 'MAL1' dosyası açılmış, bu dosyadaki tek sayı ADET değişkenine okunmuş ve dosya kapatılmıştır. Bundan sonra rasgele erişimli dosyaya yapılacak her kayıtta, kayıt numarası ADET değişkeniyle karşılaştırılarak, dosyadaki adetten daha büyük bir kayıt numarası girildiğinde 'MAL1' dosyasına yeni kayıt numarası yazılmakta ve ADET değişkeni bu değere eşitlenmektedir.

Bu şekilde, 'MAL' dosyasındaki kayıt adedi, gerektiğinde kullanılmak üzere kalıcı olarak 'MAL1' dosyasında saklanmış olacaktır.

Programdaki başka bir yenilik de, 70 numaralı satırdaki mesajdır. Bu mesajla, bilgi girişini yapan kişiye, dosyadaki kayıt adedi sürekli olarak bildirilmekte ve giriş sırasında kayıt numaralarıyla ilgili bir yanlışlık yapılması önlenmektedir.

'MAL' dosyasındaki bilgileri okuyarak ekrana listelemek için daha önce yapılmış olan program, aşağıdaki şekilde geliştirilebilir:

```
10 OPEN"R",1,"MAL",28
20 FIELD#1,20 AS MAL$,4 AS MIKTAR$,4 AS FIYAT$
30 OPEN"I",2,"MAL1"
40 INPUT#2,ADET
50 CLOSE 2
60 PRINT"MAL ADI          MİKTARI","FİYATI","TUTARI"
70 PRINT"----------       ----------","----------","----------"
80 FOR X=1 TO ADET
90 GET 1,X
100 TUTAR=CVS(MIKTAR$) * CVS(FIYAT$)
110 PRINT MAL$;CVS(MIKTAR$),CVS(FIYAT$),TUTAR
120 NEXT X
130 CLOSE 1
```

Dosyadan kaç kayıt okunacağını belirlemek için 'MAL1' dosyasındaki değer ADET değişkenine aktarılmış, 80 numaralı satır FOR

X=1 TO ADET (dosyadaki kayıt adedi) şeklinde yazılarak, 'MAL' dosyasındaki bütün bilgilerin okunması sağlanmıştır.

4 - Rasgele erişimli 'MAL' dosyasına bilgi kaydeden ve bu bilgileri okuyup ekrana listeleyen programlar ayrı ayrı yapılmıştır. Bu iki program bir menü altında şu şekilde birleştirilebilir:

```
10 OPEN"R",1,"MAL",28
20 FIELD#1,20 AS MAL$,4 AS MIKTAR$,4 AS FIYAT$
30 OPEN"I",2,"MAL1"
40 INPUT#2,ADET
50 CLOSE 2
60 CLS
70 PRINT"ANA MENÜ"
80 PRINT"----------"
90 PRINT"1 - BİLGİ GİRİŞ"
100 PRINT"2 - LİSTELEME"
110 PRINT"3 - ÇIKIŞ"
120 PRINT
130 INPUT"SEÇİMİNİZ ";S
140 IF S>3 OR S<1 THEN 60
150 ON S GOSUB 200,400,600
160 GOTO 60
200 REM ********** BİLGİ GİRİŞ **********
210 CLS
220 PRINT "DOSYADAKİ KAYIT ADEDİ :";ADET
230 PRINT "BİLGİ GİRİŞİNİ SONLANDIRMAK İÇİN KAYIT NO
OLARAK 0 VERİNİZ."
240 INPUT "KAYIT NO        ";KN
250 IF KN=0 THEN RETURN
260 INPUT "MAL ADI         ";M$
270 INPUT "MİKTARI         ";MKT
280 INPUT "FİYATI          ";FYT
290 LSET MAL$=M$
300 LSET MIKTAR$=MKS$(MKT)
310 LSET FIYAT$=MKS$(FYT)
320 PUT 1,KN
330 IF KN>ADET THEN OPEN"O",2,"MAL1" : WRITE#2,KN :
ADET=KN : CLOSE 2
340 GOTO 210
400 REM ********** LİSTELEME **********
410 CLS
```

```
420 PRINT"MAL ADI              MİKTARI","FİYATI","TUTARI"
430 PRINT"----------           ----------","---------","----------"
440 FOR X=1 TO ADET
450 GET 1,X
460 TUTAR=CVS(MIKTAR$) * CVS(FIYAT$)
470 PRINT MAL$;CVS(MIKTAR$),CVS(FIYAT$),TUTAR
480 NEXT X
490 PRINT"-----------------------------------------"
500 PRINT"MENÜYE DÖNMEK İÇİN ARA TUŞUNA BASINIZ."
510 IF INKEY$ <>" " THEN 510
520 RETURN
600 REM ******************** ÇIKIŞ **********************
610 PRINT"PROGRAM SONA ERMİŞTİR."
620 PRINT"    İYİ GÜNLER."
630 CLOSE 1
640 END
```

Bu programın, daha önce yapılmış olan sıralı erişimli dosyalarla ilgili programlardan en önemli farkı, dosyanın en başta bir kere açılması ve alt programlardan hiç birinde bir daha açılmamasıdır. Bunun nedeni, rasgele erişimli dosyalarda OPEN komutunda okuma ve yazma modunun aynı olmasıdır (OPEN"R"). Bütün işlemler bittikten sonra, çıkış bölümünde dosya kapatılmaktadır.

5 - Aşağıdaki menüye göre, 'OGRENCI' isimli rasgele erişimli dosya ile ilgili işlemleri yapan bir program:

ANA MENÜ:

1 - Bilgi giriş
2 - Bütün öğrencilerin listesi
3 - Çıkış

Kayıt deseni:

ÖĞRENCİ ADI (AD$) 30	SINIFI (SINIF$) 1	DOĞUMYILI (DOĞUM$) 2	VELİ ADI (VELİ$) 20	TELEFON NO (TELEFON$) 12

Kayıt deseninde sınıf için 1 karakter, doğum yılı için ise 2 karakterlik yer ayrılmıştır. Bunun nedeni, öğrencinin kaçıncı sınıfta olduğu (bu değer 255'ten büyük olamayacağı için) CHR$ fonksiyonuyla, doğum yılının da 32767'den büyük olamayacağı ve tamsayı olduğu için MKI$ fonksiyonuyla kodlanabilmesidir.

'OGRENCI' dosyasındaki kayıt adedini saklayabilmek için, daha önceki programda da olduğu gibi sıralı erişimli bir başka dosya kullanılmalıdır. Bu dosyanın adı da 'OGREN1' olabilir.

```
10  OPEN"R",1,"OGRENCI",65
20 FIELD#1,30 AS AD$,1 AS SINIF$,2 AS DOGUM$,20 AS
VELI$,12 AS TEL$
30  OPEN"I",2,"OGREN1"
40 INPUT#2,ADET
50 CLOSE 2
60 CLS
70  PRINT "      ANA MENÜ"
80 PRINT "--------------------------"
90 PRINT " 1 - Bilgi girişi"
100 PRINT " 2 - Bütün öğrencilerin listesi"
110 PRINT " 3 - Çıkış"
120 PRINT
130 INPUT "SEÇİMİNİZ ";S
140 IF S>3 OR S<1 THEN 60
150 ON S GOSUB 200,400,600
160 GOTO 60
200 REM ****************** BİLGİ GİRİŞ ******************
210 CLS
220 PRINT "DOSYADAKİ ÖĞRENCİ ADEDİ :";ADET
230 PRINT "BİLGİ GİRİŞİNİ SONLANDIRMAK İÇİN ÖĞRENCİ
NO OLARAK 0 VERİNİZ."
240 INPUT "ÖĞRENCİ NO    ";KN
250 IF KN=0 THEN RETURN
260 INPUT "ÖĞRENCİ ADI   ";A$
270 INPUT "SINIFI        ";SNF
280 INPUT "DOĞUM YILI    ";DY
290 INPUT "VELİ ADI      ";V$
300 INPUT "TELEFON NO    ";T$
310 LSET AD$=A$
```

```
320 LSET SINIF$=CHR$(SNF)
330 LSET DOGUM$=MKI$(DY)
340 LSET VELI$=V$
350 LSET TEL$=T$
360 PUT 1,KN
370 IF KN > ADET THEN OPEN"O",2,"OGREN1" : WRITE#2,
KN : ADET=KN : CLOSE 2
380 GOTO 210
400 REM ***************** LİSTELEME ******************
410 PRINT"NO","ÖĞRENCİ ADI";SPC(19);"SINIFI","DOĞUM
YILI","VELİ ADI"; SPC(22);"TELEFON NO"
420 PRINT STRING$(80,"-")
430 FOR X=1 TO ADET
440 GET 1,X
450 PRINT X,AD$;ASC(SINIF$),CVI(DOGUM$),VELI$;TEL$
460 NEXT X
470 PRINT STRING$(80,"-")
480 PRINT"MENÜYE DÖNMEK İÇİN ARA TUŞUNA BASINIZ."
490 IF INKEY$<> " " THEN 490
500 RETURN
600 REM ****************** ÇIKIŞ ******************
610 PRINT"PROGRAM SONA ERMİŞTİR."
620 PRINT"    İYİ GÜNLER."
630 CLOSE 1
640 END
```

Bu program çalıştırılmadan önce, 'OGREN1' isimli sıralı erişimli dosyanın oluşturulmuş olması gerekir. Yoksa, program 30 numaralı satırdaki OPEN komutunda hata mesajı verecektir. Dosyayı oluşturarak kayıt adedi olarak 0 sayısını saklamak için şu satırlar yazılabilir:

```
OPEN"O",1,"OGREN1"
WRITE#1,0
CLOSE 1
```

NOT: Bu programdaki kayıt numarası, öğrenci numarası olarak kabul edilmiş ve öğrenci kayıtlarının 1'den başlayarak sırayla yapılacağı varsayılmıştır.

Rasgele erişimli bir dosyaya kayıt ekleme:

'OGRENCI' dosyasına yeni bir kayıt eklemek için, bilgi girişi sırasında kayıt numarası olarak, program tarafından verilen kayıtlı öğrenci adedinin bir fazlası verilmelidir:

```
DOSYADAKİ ÖĞRENCİ ADEDİ: 0
BİLGİ GİRİŞİNİ SONLANDIRMAK İÇİN ÖĞRENCİ NO OLARAK 0
VERİNİZ.
ÖĞRENCİ NO  ? 1
ÖĞRENCİ ADI  ? ...
.....
.....
DOSYADAKİ ÖĞRENCİ ADEDİ: 5
BİLGİ GİRİŞİNİ SONLANDIRMAK İÇİN ÖĞRENCİ NO OLARAK 0
VERİNİZ.
ÖĞRENCİ NO  ? 6
ÖĞRENCİ ADI  ? ...
.....
.....
```

Görüldüğü gibi, bilgiler girilirken kayıt numarası olarak dosyadaki öğrenci adedinden bir fazlası verilerek dosyaya yeni kayıtlar eklenmektedir.

6 - Bir önceki örnekteki menüye 'Öğrenci numarasına göre kayıt arama' seçeneği eklenirse, program şu şekilde geliştirilebilir:

```
10 OPEN"R",1,"OGRENCI",65
20 FIELD#1,30 AS AD$,1 AS SINIF$,2 AS DOGUM$,20 AS
VELI$,12 AS TEL$
30 OPEN"I",2,"OGREN1"
40 INPUT#2,ADET
50 CLOSE 2
60 CLS
70 PRINT " ANA MENÜ"
80 PRINT "-------------------------"
90 PRINT " 1 - Bilgi girişi"
100 PRINT " 2 - Bütün öğrencilerin listesi"
```

```
110 PRINT " 3 - Öğrenci numarasına göre kayıt arama"
120 PRINT " 4 - Çıkış"
130 PRINT
140 INPUT "SEÇİMİNİZ ";S
150 IF S>4 OR S<1 THEN 60
160 ON S GOSUB 200,400,800,600
170 GOTO 60
200 REM ****************** BİLGİ GİRİŞ ******************
210 CLS
220 PRINT "DOSYADAKİ ÖĞRENCİ ADEDİ :";ADET
230 PRINT "BİLGİ GİRİŞİNİ SONLANDIRMAK İÇİN
ÖĞRENCİ NO OLARAK 0 VERİNİZ."
240 INPUT "ÖĞRENCİ NO ";KN
250 IF KN=0 THEN RETURN
260 INPUT "ÖĞRENCİ ADI    ";A$
270 INPUT "SINIFI         ";SNF
280 INPUT "DOĞUM YILI     ";DY
290 INPUT "VELİ ADI       ";V$
300 INPUT "TELEFON NO     ";T$
310 LSET AD$=A$
320 LSET SINIF$=CHR$(SNF)
330 LSET DOGUM$=MKI$(DY)
340 LSET VELI$=V$
350 LSET TEL$=T$
360 PUT 1,KN
370 IF KN > ADET THEN OPEN"O",2,"OGREN1" : WRITE#2,
KN : ADET=KN : CLOSE 2
380 GOTO 210
400 REM ****************** LİSTELEME ******************
410 PRINT"NO","ÖĞRENCİ ADI";SPC(19);"SINIFI","DOĞUM
YILI","VELİ ADI"; SPC(22);"TELEFON NO"
420 PRINT STRING$(80,"-")
430 FOR X=1 TO ADET
440 GET 1,X
450 PRINT X,AD$;ASC(SINIF$),CVI(DOGUM$),VELI$;TEL$
460 NEXT X
470 PRINT STRING$(80,"-")
480 PRINT"MENÜYE DÖNMEK İÇİN ARA TUŞUNA BASINIZ."
490 IF INKEY$<> " " THEN 490
500 RETURN
```

```
600 REM ****************** ÇIKIŞ ******************
610 PRINT"PROGRAM SONA ERMİŞTİR."
620 PRINT"    İYİ GÜNLER."
630 CLOSE 1
640 END
800 REM**ÖĞRENCİ NUMARASINA GÖRE KAYIT    AMA **
810 CLS
820 INPUT "KAYDI ARANAN ÖĞRENCİNİN NUMARASI ";NO
830 IF NO>ADET OR NO<1 THEN ?"HATALI NUMARA!":
GOTO 820
840 GET 1,NO
850 PRINT "-----------------------------------"
860 PRINT "ÖĞRENCİNİN ADI      :";AD$
870 PRINT "SINIFI              :";ASC(SINIF$)
880 PRINT "DOĞUM YILI          :";CVI(DOGUM$)
890 PRINT "VELİSİNİN ADI       :";VELI$
900 PRINT "TELEFON NO          :";TEL$
910 PRINT "-----------------------------------"
920 PRINT "ANA MENÜYE DÖNMEK İÇİN ARA TUŞUNA
BASINIZ."
930 IF INKEY$<>" " THEN 930
940 RETURN
```

Programa yeni bir bölüm eklenirken, bu bölümün program içinde, menüdeki seçeneklerin sırasına uygun olarak yer alması zorunlu değildir. Bu programda, üçüncü seçenek, dördüncü alt programdır. Bunu düzenleyebilmek için, 160 numaralı satırdaki ON...GOSUB komutuyla birlikte kullanılan satır numaralarının sırası, alt programların sırasına uygun biçimde verilmiştir. (Dördüncü seçenekteki çıkış, 600 numaralı satırdan başlayarak yazılmıştır.)

Öğrenci numarasına göre kayıt arama bölümünde, önce öğrenci numarası girilmekte, bu öğrenci numarası dosyadaki sıra numarası olarak kullanılarak gerekli bilgiler 840 numaralı satırdaki GET komutuyla okunmaktadır. Daha sonra bu öğrenciye ait bilgiler düzgün bir biçimde ekrana listelenmekte ve program ana menüye dönüş için kullanıcının ara tuşuna basmasını beklemektedir.

830 numaralı satırdaki IF...THEN cümlesinde, girilmiş olan öğrenci numarası (kayıt numarası) doğru girişi sağlamak için kontrol edilmektedir.

Rasgele erişimli bir dosyada saha değişikliği:

Rasgele erişimli bir dosya kullanılırken, istenilen bir kaydın değiştirilebilmesi için o kaydın numarası verilerek aynı bilgiler yeni şekliyle tekrar girilebilir. Böylece, girilen yeni bilgiler dosyaya yazılarak kayıt üzerinde gerekli değişiklik yapılmış olur.

Rasgele erişimli dosyalamanın getirdiği kolaylıklardan yararlanarak yapılan bu değiştirme işlemi, bazı durumlarda sakıncalı olabilir. Örneğin, çok sayıda sahayı içeren bir kaydın bu şekilde değiştirilebilmesi için, bütün sahaların yeniden yazılması gerekecektir. Bunu engellemek için, sıralı erişimli dosyalamada da olduğu gibi, özel bir programdan yararlanılabilir.

7 - Aşağıdaki menüye göre, 'HAPIS' isimli rasgele erişimli dosyayla ilgili işlemleri yapan bir program:

ANA MENÜ

1 - Yeni mahkum kaydetme
2 - Bütün mahkumların listesi
3 - Kayıt numarasına göre mahkumların listesi
4 - Suç cinsine göre liste
5 - Ceza cinsine göre liste
6 - Suçlu adına göre liste
7 - Ceza değişikliği
8 - Çıkış

Kayıt deseni:

MAHKUM ADI (MAHKUM$)	SUÇU (SUC$)	CEZASI (CEZA$)	CİNSİYETİ (CİNS$)	DOĞUMYILI (DOĞUM$)
30	15	15	1	2

```
10 OPEN"R",1,"HAPIS",63
20 FIELD#1,30 AS MAHKUM$,15 AS SUC$,15 AS CEZA$,1
AS CINS$,2 AS DOGUM$
30 OPEN"I",2,"HAPIS1"
```

```
40 INPUT#2,ADET
50 CLOSE 2
60 CLS
70 PRINT "      ANA MENÜ"
80 PRINT "-------------------------"
90 PRINT  " 1 - Yeni mahkum kaydetme
100 PRINT " 2 - Bütün mahkumların listesi"
110 PRINT " 3 - Kayıt numarasına göre mahkum arama"
120 PRINT " 4 - Suç cinsine göre liste"
130 PRINT " 5 - Ceza cinsine göre liste"
140 PRINT " 6 - Mahkum adına göre liste"
150 PRINT " 7 - Ceza değişikliği"
160 PRINT " 8 - Çıkış"
170 PRINT
180 INPUT "SEÇİMİNİZ ";S
190 IF S>8 OR S<1 THEN 60
200 ON S GOSUB 500,1000,1500,2000,2500,3000,3500,4000
210 GOTO 60
500 REM ****************** BİLGİ GİRİŞ ******************
510 CLS
520 PRINT "HAPİSHANEDEKİ MAHKUM ADEDİ :";ADET
530 PRINT "BİLGİ GİRİŞİNİ SONLANDIRMAK İÇİN MAHKUM
NO OLARAK 0 VERİNİZ."
540 INPUT "MAHKUM NO    ";KN
550 IF KN=0 THEN RETURN
560 INPUT "MAHKUM ADI   ";A$
570 INPUT "SUÇU         ";S$
580 INPUT "CEZASI       ";C$
590 INPUT "CİNSİYETİ    ";CS$
600 INPUT "DOĞUM YILI   ";DY
610 LSET MAHKUM$=A$
620 LSET SUC$=S$
630 LSET CEZA$=C$
640 LSET CINS$=CS$
650 LSET DOGUM$=MKI$(DY)
660 PUT 1,KN
670 IF KN> ADET THEN OPEN"O",2,"HAPIS1" : WRITE#2,KN:
ADET=KN : CLOSE 2
680 GOTO 510
1000 REM ******* BÜTÜN MAHKUMLARIN LİSTESİ ********
```

```
1010 CLS
1020 PRINT "MAHKUM ADI";TAB(31);"SUÇ";TAB(45);
"CEZA";TAB(60);"CİNSİYET"; TAB(70);"DOĞUM YILI"
1030 PRINT "----------";TAB(31);"----";TAB(45);"----";TAB(60);"--";
TAB(70);"----------"
1040 FOR X=1 TO ADET
1050 GET 1,X
1060 PRINT MAHKUM$;SUC$;CEZA$;CINS$;CVI(DOGUM$)
1070 NEXT
1080 PRINT
1090 PRINT "MENÜYE DÖNMEK İÇİN ARA TUŞUNA
BASINIZ."
1100 IF INKEY$<>" "THEN 1100
1110 RETURN
1500  *REM KAYIT NUMARASINA GÖRE MAHKUMLARIN
LİSTESİ*
1510 CLS
1520 INPUT "ARANAN MAHKUMUN KAYIT NUMARASI ";K
1530 PRINT "MAHKUM ADI";TAB(31);"SUÇ";TAB(45);
"CEZA";TAB(60);"CİNSİYET";  TAB(70);"DOĞUM YILI"
1540 PRINT "--------";TAB(31);"----";TAB(45);"----";TAB(60);"----";
TAB(70);"----------"
1550 GET 1,K
1560 PRINT MAHKUM$;SUC$;CEZA$;CINS$;CVI(DOGUM$)
1570 PRINT
1580 PRINT "MENÜYE DÖNMEK İÇİN ARA TUŞUNA
BASINIZ."
1590 IF INKEY$<>" "THEN 1590
1600 RETURN
2000 REM ********* SUÇ CİNSİNE GÖRE LİSTE ***********
2010 CLS
2020 INPUT "HANGİ SUÇU İŞLEYENLERİN LİSTESİNİ
İSTİYORSUNUZ ";S$
2030 PRINT "MAHKUM ADI";TAB(31);"SUÇ";TAB(45);
"CEZA";TAB(60);"CİNSİYET";  TAB(70);"DOĞUM YILI"
2040 PRINT "--------";TAB(31);"----";TAB(45);"----";TAB(60);"----";
TAB(70);"----------"
2050 FOR X=1 TO ADET
2060 GET 1,X
```

```
2070 IF INSTR(SUC$,S$)>Ø THEN PRINT MAHKUM$;SUC$;
CEZA$;CINS$;CVI(DOGUM$)
2080 NEXT
2090 PRINT
2100 PRINT "MENÜYE DÖNMEK İÇİN ARA TUŞUNA
BASINIZ."
2110 IF INKEY$<>" "THEN 2110
2120 RETURN
2500 REM ********* CEZA CİNSİNE GÖRE LİSTE **********
2510 CLS
2520 INPUT "HANGİ CEZAYI ALANLARIN LİSTESİNİ
İSTİYORSUNUZ ";C$
2530 PRINT "MAHKUM ADI";TAB(31);"SUÇ";TAB(45);
"CEZA";TAB(60);"CİNSİYET";  TAB(70);"DOĞUM YILI"
2540 PRINT "--------";TAB(31);"----";TAB(45);"----";TAB(60);"----";
TAB(70);"----------"
2550 FOR X=1 TO ADET
2560 GET 1,X
2570 IF INSTR(CEZA$,C$)>Ø THEN PRINT MAHKUM$;
SUC$; CEZA$;CINS$;CVI(DOGUM$)
2580 NEXT
2590 PRINT
2600 PRINT "MENÜYE DÖNMEK İÇİN ARA TUŞUNA
BASINIZ."
2610 IF INKEY$<>" "THEN 2610
2620 RETURN
3000 REM ******** MAHKUM ADINA GÖRE LİSTE ********
3010 CLS
3020 INPUT "KAYDI ARANAN MAHKUMUN ADI ";A$
3030 PRINT "MAHKUM ADI";TAB(31);"SUÇ";TAB(45);
"CEZA";TAB(60);"CİNSİYET";  TAB(70);"DOĞUM YILI"
3040 PRINT "--------";TAB(31);"----";TAB(45);"----";TAB(60);"----";
TAB(70);"----------"
3050 FOR X=1 TO ADET
3060 GET 1,X
3070 IF INSTR(MAHKUM$,A$)>Ø THEN PRINT MAHKUM$;
SUC$; CEZA$;CINS$;CVI(DOGUM$)
3080 NEXT
3090 PRINT
```

```
3100 PRINT "MENÜYE DÖNMEK İÇİN ARA TUŞUNA
BASINIZ."
3110 IF INKEY$<>" "THEN 3110
3120 RETURN
3500 REM ********** CEZA DEĞİŞİKLİĞİ **********
3510 CLS
3520 INPUT "CEZASI DEĞİŞECEK MAHKUMUN KAYIT
NUMARASI ";K
3530 GET 1,K
3540 PRINT "MAHKUMUN ADI      : ";MAHKUM$
3550 PRINT "BU MAHKUMUN CEZASI :";CEZA$
3560 INPUT "YENİ CEZA         : ";C$
3570 LSET CEZA$=C$
3580 PUT 1,K
3590 RETURN
4000 REM ********** ÇIKIŞ **********
4010 CLS
4020 PRINT "PROGRAM BİTMİŞTİR."
4030 CLOSE 1
4040 END
```

Bu programda, kayıt üzerindeki belirli bir saha (CEZA$) içindeki bilginin değiştirilmesi söz konusudur. Bunun için, 3500 ve 3590 numaralı satırlar arasındaki bölüm yazılmıştır. Bu bölümde, önce değişecek kaydın numarası klavyeden girilerek (3520), bu kayda ait bütün bilgiler dosyadan okunmuştur (3530). Daha sonra mahkumun adı ve eski cezası ekrana yazılarak (3540, 3550) klavyeden bu mahkuma ait yeni ceza girilmiştir (3560). 3570 numaralı satırda yeni girilen ceza CEZA$ sahasına yerleştirilmiş ve 3580 numaralı satırda dosyaya kaydedilmiştir. Bu kayda ait diğer bilgiler, GET komutuyla okunduktan sonra herhangi bir değişikliğe uğramadıklarından, PUT komutuyla dosyaya eski halleriyle aktarılmıştır.

Bu yöntemin kullanışlı olmasının nedeni, yalnızca değişiklik yapılacak sahaya ait bilginin girilmesinin yeterli olmasıdır. Aynı değişiklik bilgi giriş bölümünde yapılmak istenirse, diğer sahalara ait bilgilerin de yeniden girilmesi gerekir. Ancak, değiştirilmek istenen birden çok saha varsa, sıralı dosyalama yönteminde de anlatıldığı gibi tüm sahaların bilgilerinin birlikte değiştirilebileceği tek bir alt program kullanmak daha uygun olacaktır.

8 - Türkiyedeki şehirlerle ilgili çeşitli bilgilerin saklandığı 'SEHIR' isimli dosyayla ilgili işlemleri aşağıdaki menüye uygun olarak gerçekleştiren bir program:

ANA MENÜ:

1 - Bilgi giriş
2 - Listeleme
3 - Bilgi değiştirme
4 - Çıkış

ŞEHİR ADI (ŞEHİR$)	YÜZÖLÇÜMÜ (YUZ$)	NÜFUSU (NUFUS$)	TARIM ÜRÜNLERİ (TARIM$)	MADENLERİ (MADEN$)
25	2	8	30	30

NOT:

a) Aynı örnek, daha önce sıralı dosyalama yöntemiyle yapılmıştır.

b) Trafik kodu, rasgele erişimli dosyada kayıt numarası olarak kullanılacağından kayıt deseninden çıkarılmıştır.

```
10 OPEN"R",1,"SEHIR",95
20 FIELD 1,25 AS SEHIR$,2 AS YUZ$,8 AS NUFUS$,30 AS
TARIM$,30 AS MADEN$
30 OPEN"I",2,"SEHIR1"
40 INPUT#2,ADET
50 CLOSE 2
60 CLS
70 PRINT "    ANA MENÜ:"
80 PRINT "--------------------------"
90 PRINT "1 - Bilgi girişi"
100 PRINT "2 - Listeleme"
110 PRINT "3 - Bilgi değiştirme"
120 PRINT "4 - Çıkış"
130 PRINT
140 INPUT "SEÇİMİNİZ ";S
150 IF S>4 OR S<1 THEN 60
160 ON S GOSUB 500,1000,1500,2000
170 GOTO 60
```

```
500 REM ***************** BİLGİ GİRİŞİ ******************
510 CLS
520 PRINT "DOSYADA KAYITLI";ADET;"ŞEHİR VAR."
530 PRINT "BİLGİ GİRİŞİNİ SONLANDIRMAK İÇİN
TRAFİK KODU  OLARAK 0 VERİN."
540 PRINT "------------------------------------------------------------"
550 INPUT "ŞEHRİN TRAFİK KODU      ";TK
560 IF TK<1 THEN RETURN
570 INPUT "ŞEHİR ADI               ";S$
580 INPUT "YÜZÖLÇÜMÜ               ";Y
590 INPUT "NÜFUSU                  ";N
600 LINE INPUT "TARIM ÜRÜNLERİ     ";T$
610 LINE INPUT "MADENLERİ          ";M$
620 LSET SEHIR$=S$
630 LSET YUZ$=MKI$(Y)
640 LSET NUFUS$=MKD$(N)
650 LSET TARIM$=T$
660 LSET MADEN$=M$
670 PUT 1,TK
680 IF TK>ADET THEN ADET=TK:OPEN"O",2, "SEHIR1":
WRITE#2, ADET:CLOSE 2
690 GOTO 510
1000 REM *************** LİSTELEME *************
1010 CLS
1020 FOR S=1 TO ADET
1030 GET 1,S
1040 PRINT "TRAFİK KODU            :";S
1050 PRINT "ŞEHİR ADI              :";SEHIR$
1060 PRINT "YÜZÖLÇÜMÜ              :";CVI(YUZ$)
1070 PRINT "NÜFUSU                 :";CVD(NUFUS$)
1080 PRINT "TARIM ÜRÜNLERİ         : ";TARIM$
1090 PRINT "MADENLERİ              : ";MADEN$
1100 PRINT "------------------------------------------------"
1110 PRINT "  DEVAM ETMEK İÇİN ARA TUŞUNA BASINIZ."
1120 PRINT "------------------------------------------------------------"
1130 IF INKEY$<>" "THEN 1130
1140 NEXT
1500 REM ************* BİLGİ DEĞİŞTİRME ***************
1510 CLS
1520 INPUT "BİLGİLERİ DEĞİŞECEK ŞEHRİN TRAFİK
KODU ";K
1530 PRINT "------------------------------------------------"
1540 GET 1,K
```

```
1550 PRINT "TRAFİK KODU                    :";K
1560 PRINT "ŞEHİR ADI                      : ";SEHIR$
1570 PRINT "YÜZÖLÇÜMÜ                      :";CVI(YUZ$)
1580 PRINT "NÜFUSU                         :";CVD(NUFUS$)
1590 PRINT "TARIM ÜRÜNLERİ                 :";TARIM$
1600 PRINT "MADENLERİ                      : ";MADEN$
1610 PRINT "-------------------------------------------------"
1620 INPUT "ŞEHRİN YENİ ADI                ";S$
1630 INPUT "YÜZÖLÇÜMÜ                      ";Y
1640 INPUT "NÜFUSU                         ";N
1650 INPUT "TARIM ÜRÜNLERİ                 ";T$
1660 INPUT "MADENLERİ                      ";M$
1670 IF S$<>"" THEN LSET SEHIR$=S$
1680 IF Y<>Ø THEN LSET YUZ$=MKI$(Y)
1690 IF N<>Ø THEN LSET NUFUS$=MKD$(N)
1700 IF T$<>"" THEN LSET TARIM$=T$
1710 IF M$<>"" THEN LSET MADEN$=M$
1720 PUT 1,K
1730 RETURN
2000 REM ********************* ÇIKIŞ*********************
2010 CLS
2020 PRINT "HOÇAKALIN."
2030 CLOSE
2040 END
```

Rasgele erişimli bir dosyadan kayıt silme:

Rasgele erişimli dosyalamayla ilgili şu ana kadar yapılmış olan programlarda, kayıt silmeyle ilgili bir bölüm yer almamıştır. Herhangi bir kaydın dosyadan çıkartılabilmesi için bu programlarda, yeni girilen bir kaydın numarası silinmek istenen kaydın numarası olarak verilip, yeni bilgiler eskilerin üzerine yazılarak eski kayıt dosyadan çıkartılabilir. Ancak bu yöntem yalnız yeni bir kayıt girileceği zaman uygulanabileceğinden, kullanışlı değildir.

Rasgele erişimli bir dosyadan istenilen bir kaydın silinebilmesi için aşağıdaki iki yöntemden biri kullanılabilir:

A) Silinmek istenen kayıttan sonraki bütün kayıtlar bir geriye kaydırılabilir.

B) Dosyadaki en son kayıt silinmek istenen kaydın üzerine yazılabilir.

A) Bu yöntemde, silinmek istenen kayıttan sonraki bütün kayıtlar okunarak kendilerinden bir önceki kaydın yerine yazılırlar. Böylece silinecek kaydın yerini başka bilgiler alır. Özellikle kayıt sayısı çok olan dosyalarda bu işlemin uygulanması, oldukça fazla zaman alacaktır. Ancak, kayıtlar belirli bir sıraya göre düzenlenmişse, bu yöntemin kullanılması zorunludur.

B) Birinciden daha basit ve hızlı olan bu yöntemde, dosyadaki en son kayıt okunarak, silinmek istenen kaydın yerine yazılır. Hızlı olmasından başka, bu yöntemin bir avantajı da yeri değişen en son kayıt dışında hiçbir kaydın numarasının değişmemesidir. Diğer yöntemde, bir geri kaydırılan bütün kayıtların numaraları bir eksilmektedir.

NOT: Hangi yöntem kullanılırsa kullanılsın, dosyadan kayıt çıkarılması söz konusu olduğu zaman, sıralı bir dosyada saklı olan kayıt adedi bir eksiltilmelidir.

ÖRNEK: Daha önce oluşturulmuş olan 'HAPIS' isimli dosyadan kayıt numarası verilerek istenilen bir kaydın silinmesi:

A)
```
10 OPEN"R",1,"HAPIS",63
20 FIELD#1,30 AS MAHKUM$,15 AS SUC$,15 AS CEZA$,
1 AS CINS$,2 AS DOGUM$
30 OPEN"I",2,"HAPIS1"
40 INPUT#2,ADET
50 CLOSE 2
60 CLS
70 INPUT "SİLİNECEK KAYDIN NUMARASI :",K
80 GET 1,K
90 PRINT "MAHKUM ADI : ";MAHKUM$
100 INPUT "SİLMEK İSTİYOR MUSUNUZ (E/H) ";EH$
110 IF EH$="H" OR EH$="h" THEN 60
120 FOR X=K+1 TO ADET
130 GET 1,X
140 PUT 1,X-1
150 NEXT
160 OPEN"O",2,"HAPIS1"
```

```
170 ADET=ADET-1
180 WRITE#2,ADET
190 CLOSE 1,2
200 PRINT "KAYIT SİLİNMİŞTİR."
210 PRINT "DOSYADA KALAN KAYIT ADEDİ :";ADET
```

Bu programda, silinecek kaydın numarası klavyeden okunduktan sonra bu kayıt önce ekrana yazılarak kullanıcının kayıt numarasında bir yanlışlık yapıp yapmadığı kontrol edilmekte ve istenmeyen bir kaydın silinmesi engellenmektedir. Bundan sonra 120 ve 150 numaralı satırlar arasındaki döngüde silinmek istenen kayıttan bir sonrakinden (K+1) en son kayda (ADET) kadar bütün kayıtlar (X) okunarak kendilerinden bir önceki kaydın yerine (X-1) yazılmaktadır. Daha sonra, adet bir eksiltilerek (ADET=ADET-1), kayıt adedinin saklandığı 'HAPIS1' dosyasına kaydedilmektedir.

B)
```
10 OPEN"R",1,"HAPIS",63
20 FIELD#1,30 AS MAHKUM$,15 AS SUC$,15 AS CEZA$,1
AS CINS$,2 AS DOGUM$
30 OPEN"I",2,"HAPIS1"
40 INPUT#2,ADET
50 CLOSE 2
60 CLS
70 INPUT "SİLİNECEK KAYDIN NUMARASI :",K
80 GET 1,K
90 PRINT "MAHKUM ADI : ";MAHKUM$
100 INPUT "SİLMEK İSTİYOR MUSUNUZ (E/H) ";EH$
110 IF EH$="H" OR EH$="h" THEN 60
120 GET 1,ADET
130 PUT 1,K
140 OPEN"O",2,"HAPIS1"
150 ADET=ADET-1
160 WRITE#2,ADET
170 CLOSE 1,2
180 PRINT "KAYIT SİLİNMİŞTİR."
190 PRINT "DOSYADA KALAN KAYIT ADEDİ :";ADET
```

İkinci yöntemin kullanıldığı bu programda, bir önceki programdaki döngünün yerine bir GET ve bir PUT komutu kullanılmıştır. 120 numaralı satırdaki GET komutuyla dosyadaki son kayıt (ADET) okunmakta, 130 numaralı satırdaki PUT komutuyla silinecek kaydın yerine (K) yazılmaktadır.

Programlardan da anlaşılabileceği gibi, ikinci programda herhangi bir döngüye gerek kalmadan, istenilen kayıt yalnızca iki komutla (GET ve PUT) dosyadan silindiğinden, işlem döngüden yararlanılan birinci programa oranla çok daha kısa bir sürede gerçekleşecektir.

SORULAR:

Aşağıda ana menüleri ve kayıt desenleri verilen konularla ilgili işlemleri rasgele erişimli dosyalama yöntemiyle gerçekleştiren programları hazırlayınız.

1) ANA MENÜ:

1 - 'VIDEO' dosyasına bilgi girişi
2 - Bütün film adlarının listesi
3 - Kaset numarasına göre film listesi
4 - Oyuncu adına göre film listesi
5 - Film adına göre liste
6 - Çıkış

Kayıt deseni:

Film adı	Oyuncular	Yönetmen	Yapım yılı	Süresi	Dili

NOT: Burada kaset numarasının kayıt numarası olarak kullanılması, programlamayı kolaylaştıracaktır.

2) ANA MENÜ

1-'FUTBOL' dosyasına bilgi girişi
2- Haftaya göre maçlarla ilgili bilgilerin listesi
3 - Tüm maçların listesi
4 - Rakip takım adına göre maçların listesi
5 - Kazanılan maçların listesi
6 - Kaybedilen maçların listesi
7 - Çıkış

Kayıt deseni

Rakip takım	Maçın yeri	Atılan gol	Yenen gol	Hakem adı

NOT: Burada, haftaların (1. hafta, 2.hafta, v.s.) kayıt numarası olarak kullanılması programlamayı kolaşlaştıracaktır.

3) ANA MENÜ:

1 - 'OTEL' dosyasına bilgi girişi
2 - Oda numarasına göre bilgilerin listelenmesi
3 - Tüm odaların listesi
4 - Müşteri adına göre liste
5 - Otelden ayrılan müşterilerin dosyadan çıkarılması
6 - Boş odaların listesi
7 - Çıkış

Kayıt deseni:

Müşteri adı	Geliş tarihi	Uyruğu

NOT: Burada, oda numaraları dosyadaki kayıt numaraları olarak kullanılmalıdır. Müşterilerin otelden ayrılması durumunda, kayıt silme işlemi kaydın dosyadan çıkarılması değil, kapladığı sahaların boşaltılması olarak düşünülmelidir. Bunun nedeni, oda sayısının değiştirilemeyeceği, ancak içindeki kişilerin değişebileceğidir.

EDİTÖR KULLANIMI

Editörler, disket üzerinde kayıtlı bulunan dosyaların istenildiği şekilde işlenebilmesi için kullanılan özel programlardır. Bu programların genel çalışma düzeni aşağıdaki şekildedir:

1) Başlamak için kullanılacak olan editör programı çalıştırılır.

2) Üzerinde işlem yapılmak istenen dosyanın adı verilerek bu dosya disketten belleğe aktarılır. (Eğer dosya ilk kez yaratılmak isteniyorsa bu işlem yapılmaz.)

3) Dosyanın görüntüsü ekranda belirir. (Eğer dosya ekrana sığmayacak kadar büyükse baştan başlayarak ekrana sığdığı kadarı görülür; dosya ilk kez oluşturuluyorsa, ekran boş olacaktır.)

4) Dosya, klavyeden yararlanarak, istenildiği şekilde işlenir:

a) Bazı bilgiler değiştirilebilir.
b) Yeni bilgiler (dosyanın herhangi bir yerine) eklenebilir.
c) Bazı bilgiler silinebilir.

Bu işlemler yapılırken, ekran üzerinde belirecek olan kursörden yararlanılır. Aşağı, yukarı, sağa veya sola hareket ettirilebilen kursör yardımıyla, dosyanın neresinin işleneceği belirlenir. (Hemen hemen bütün editör programlarında kursörün bir alt sayfaya, bir üst sayfaya, dosya başına, dosya sonuna veya aranacak olan herhangi bir kelimeye taşınabilmesi için özel tuşlar veya komutlar vardır.)

5) En son şekline getirilen dosya, diskete kaydedilir. Kullanılan editörün özelliklerine göre, bu dosyaya yeni bir isim verilebilir veya dosya eski ismiyle kaydedilebilir. (Ilk kez oluşturulan dosyalar için yeni bir isim verilmelidir.)

NOT: Yukarıdaki adımlar, bütün editörler için geçerli değildir. Birçok editör programının kendine özgü farklı kullanımları vardır. Ancak, genel olarak işlemler aynı mantık çerçevesi içinde gerçekleşmektedir.

Editörlerle oluşturulan veya işlenen dosyalar, genellikle yazı işlem programlarının metinleri veya derleyiciler için derlenecek kaynak programlardır.

Yazı işlem programlarında, ilân, mektup, hatta kitaplar için çeşitli metinler hazırlamak mümkündür. Elinizde tuttuğunuz bu kitap da, böyle bir yazı işlem programıyla (bir tür editör) hazırlanmıştır.

Editörlerle hazırlanan çeşitli dillerdeki kaynak programlar, yine bu diller için hazırlanmış derleyiciler (COBOL, PASCAL, FORTRAN, vs.) tarafından derlenerek amaç programa dönüştürülür ve çalıştırılabilirler. Birçok derleyicinin özel editörü olduğu gibi, genel amaçlı editör programları da bu iş için kullanılabilir.

Dosya editörlerinin dışında, dosyalardan bağımsız olarak doğrudan doğruya disket üzerinde işlem yapabilen editör programları da vardır. Bu tür programlar, daha çok programcılar tarafından teknik amaçlarla kullanılmaktadır.

PROJELER

Bu bölümde, ticarî alanda en çok kullanılan birkaç proje incelenecektir. Eğitim amacı ön planda tutulduğundan, programların öğrenci tarafından kolayca izlenebilmesi için programlar yazılırken ekran düzenlemesi ve benzeri ayrıntılara gereğinden fazla önem verilmemiştir. Ancak öğrencilerin programlama mantığını tam olarak kavrayıp programları hatasız olarak çalıştırdıktan sonra, bu tür çalışmalar yapmaları önerilir.

PROJE - 1: ÜCRET BORDROSU

Herhangi bir işletmede çalışan personelin maaşlarının bilgisayar yardımıyla hesaplanması için kullanılabilecek bir ücret bordrosu programı:

Bu programda her personel için, aşağıdaki bilgilerin bir dosyada saklanması gerekmektedir:

1 - Personel adı ve soyadı
2 - Sigorta numarası
3 - Vergi karnesi numarası
4 - Adresi
5 - Telefon numarası
6 - Brüt ücreti

Ücret bordrosunun hesaplanabilmesi için bazı sabit değerlere ihtiyaç vardır. Bu değerler belirli zamanlarda değişebileceğinden, program içinde sabit olarak saklanmaları sakıncalıdır. Bu nedenle, istenildiğinde kullanıcı tarafından değiştirilebilmeleri amacıyla, sabit değerler ayrı bir dosyaya kaydedilecektir. Bu ikinci dosyada saklanacak bilgiler de şunlardır:

1 - Sosyal Sigorta tavan matrahı
2 - Tasarruf teşvik işveren hissesi
3 - Tasarruf teşvik işçi hissesi
4 - SSK primi yüzdesi
5 - Gelir vergisi yüzdesi
6 - Damga vergisi yüzdesi
7 - Genel indirim (günlük)

Program çalıştırıldığında, sonuç olarak istenenler de şunlardır:

1 - Bütün personelin ücret bordrosu
2 - Personel adına göre herhangi bir personelin ücret bordrosu
3 - Personel numarasına göre herhangi bir personelin özel bilgileri
4 - Bütün personelin isim ve maaşlarının listesi
5 - Personel bilgileriyle ilgili değişiklikler
6 - Sabit değerlerin listelenmesi
7 - Sabit değerlerle ilgili değişiklikler

Ücret bordrosu listesi aşağıdaki şekilde olacaktır:

İşveren Ünvanı :
İşveren Adresi :
İşyeri Sigorta Sicil no :
Vergi Dairesi :
Vergi Hesap no :
Ait Olduğu Ay :

ÜCRET BORDROSU

Personel adı soyadı .....:
Sigorta sicil no.....:
Vergi karnesi no .....:
Sosyal sigorta tavan matrahı .....:
SSK primi .....:
Brüt ücret .....:
Tasarruf teşvik işveren hissesi.....:
Tasarruf teşvik işçi hissesi .....:
Genel indirim.....:
Gelir vergisi matrahı .....:
Gelir vergisi.....:
Damga vergisi.....:
Kesintiler toplamı.....:
NET ÜCRET .....:

Ücret bordrosu hesaplama işlemi, genel olarak yukarıdaki şekilde olmasına rağmen, bazı işyerlerinde ek bilgiler de bulunabilir. Bilgilerin az tutulmasının nedeni, programın daha anlaşılabilir olmasını sağlamaktır.

Saha adedi çok fazla olduğundan, hepsinin yan yana bilgisayar ekranında görüntülenmesi olanaksızdır. Bu tür programlarda listeler iki veya üç satır halinde ve belirli sütunlardan oluşacak şekilde hazırlanır. Programın rahatlıkla incelenebilmesi amacıyla, tüm sahalar alt alta yazılmıştır.

Ücret bordrosundaki bilgilerin hesaplanması aşağıdaki yolla yapılacaktır:

Personel adı ve soyadı	:	Personel dosyasından okunacak.
Sigorta sicil numarası	:	Personel dosyasından okunacak.
Vergi karnesi numarası	:	Personel dosyasından okunacak.
Sosyal sigorta tavan matrahı	:	Sabit bilgiler dosyasından okunacak.
SSK primi	:	Sosyal sigorta matrahının % 14'ü.
Brüt ücet	:	Personel dosyasından okunacak.
Tasarruf teşvik işveren hissesi	:	Brüt ücretin % 6'sı.
Tasarruf teşvik işçi hissesi	:	Brüt ücretin % 4'ü.
Genel indirim	:	Gün x 600
Gelir vergisi matrahı :		

Brüt ücret - (SSK primi + Tas. teş. işçi his. + Genel ind.)
Gelir vergisi : Gelir vergisi matrahının % 25'i.
Damga vergisi : Brüt ücretin % 0.4'ü.
Kesintiler toplamı :
SSK primi + Tas. Teş. işçi his. + Gelir vergisi + Damga vergisi
NET ÜCRET : Brüt ücret - kesintiler toplamı

'PERSONEL' isimli rasgele erişimli dosyanın kayıt deseni aşağıdaki gibi olacaktır:

Saklanacak bilgi	Saha adı	Saha uzunluğu	
Personel adı ve soyadı	AD$	30	
Sigorta numarası	SIGORTA$	8	(MKD$)
Vergi karnesi numarası	VERGI$	8	(MKD$)
Adresi	ADRES$	30	
Telefon numarası	TELEFON$	12	
Brüt ücreti	BRUT$	4	(MKS$)

NOT : Bu dosyada kayıt numarası, personel sıra numarası olarak saklanacak, dosya üzerindeki değişiklikler bu kayıt numarasına göre yapılacaktır.

Sabit değerlerin saklanacağı ve istenildiğinde değiştirileceği 'SABIT' isimli sıralı erişimli dosyanın kayıt deseni ise şu şekildedir:

Saklanacak bilgi	Saha adı
Sosyal sigorta tavan matrahı	SSTM
Tasarruf teşvik işveren hissesi	TTISVY
Tasarruf teşvik işçi hissesi	TTISCY
SSK primi yüzdesi	SSKY
Gelir vergisi yüzdesi	GVY
Damga vergisi yüzdesi	DVY
Genel indirim (günlük)	GIG

NOT : Bu dosyada, bir tek kayıt olacaktır. Bu nedenle dosyanın oluşturulması veya okunması sırasında döngü kullanılmayacaktır. Program, en başta bu dosyadaki bilgileri okuyacak ve işlemleri buradan alınan değerlere göre yapacaktır. Bu yüzden dosya, program çalıştırılmadan önce oluşturulmuş olmalıdır. Bu dosyayı oluşturmak için şu satırlar (satır numarasız olarak) yazılabilir:

```
OPEN"O",1,"SABIT"
SSTM=1312020
TTISVY=0.06
TTISCY=0.04
SSKY=0.14
GVY=0.25
DVY=0.004
GIG=600
WRITE#1,SSTM,TTISVY,TTISCY,SSKY,GVY,DVY,GIG
CLOSE 1
```

Bu satırlar yalnız bir kez yazılacağından, satır numarasız olarak yazılabilirler. Ama istenirse her komutun başına bir satır numarası koyularak bir program yazılabilir ve sadece bir kez çalıştırıldıktan sonra bu program silinir.

Aynı şekilde, personel adedinin saklanacağı 'PERSADET' isimli sıralı erişimli dosya da oluşturulmalıdır:

```
OPEN"O",1,"PERSADET"
WRITE#1,0
CLOSE 1
```

NOT : Gelir vergisi yüzdesi, toplam vergi matrahına göre değişebilir. Bu da, yılbaşından itibaren personele ait toplan vergi matrahıdır. Vergi matrahı, ikramiye, maaş ve ek ödemelere göre değişebilir. Programın izlenmesini kolaylaştırabilmek amacıyla bu tür ödemeler göz önünde bulundurulmamıştır. Bu nedenle toplam vergi matrahı alınmamış ve gelir vergisi yüzdesi her zaman 'SABIT' dosyasından okunan değere eşit tutulmuştur.

```
10  OPEN"R",1,"BORDRO",92
2Ø FIELD 30 AS AD$, 8 AS SİGORTA$, 8 AS VERGI$, 30 AS
ADRES$ 12 AS TELEFON$, 4 AS BRUT$
30 OPEN"I",2,"SABIT"
40 INPUT#2,SSTM,TTISVY,TTISCY,SSKY,GVY,DVY,GIG
50 CLOSE 2
60 OPEN"I",2,"PERSADET"
70 INPUT#2,ADET
80 CLOSE 2
90 REM ********** ANA MENÜ **********
```

```
100 CLS
110 PRINT "              A N A   M E N Ü"
120 PRINT "-------------------------------------------------"
130 PRINT " 1 - Personel bilgilerinin girişi"
140 PRINT " 2 - Bütün personelin ücret bordrosu"
150 PRINT " 3 - Personel adına göre ücret bordrosu"
160 PRINT " 4 - Personel numarasına göre personelin özel
bilgileri"
170 PRINT " 5 - Bütün personelin isim ve maaşlarının listesi"
180 PRINT " 6 - Personel bilgileriyle ilgili değişiklikler"
190 PRINT " 7 - Sabit değerlerin listesi"
200 PRINT " 8 - Sabit değerlerle ilgili değişiklikler"
210 PRINT " 9 - Çıkış"
220 PRINT
230 INPUT " SEÇİMİNİZ";S
240 IF S>9 OR S<1 THEN 100
250 ON S GOSUB 500, 1000, 1500, 2000, 2500, 3000, 3500,
4000, 4500
260 GOTO 100
500 REM ********* PERSONEL BİLGİLERİNİN GİRİŞİ *******
510 CLS
520 PRINT "PERSONEL DOSYASINDA KAYITLI PERSONEL
ADEDİ :";ADET
530 PRINT "MENÜYE DÖNMEK İÇİN PERSONEL NUMARASI
OLARAK 0 VERİNİZ."
540 PRINT "-------------------------------------------------"
550 INPUT "PERSONEL NUMARASI          : ",PN
560 IF PN=0 THEN RETURN
570 INPUT "ADI VE SOYADI              : ",A$
580 INPUT "SİGORTA NUMARASI           : ",SN
590 INPUT "VERGİ KARNESİ NUMARASI     : ",VKN
600 LINE INPUT "ADRESİ                : ",ADR$
610 INPUT "TELEFON NUMARASI           : ",TEL$
620 INPUT "BRÜT ÜCRET                 : ",BR
630 LSET AD$      = A$
640 LSET SIGORTA$ = MKD$(SN)
650 LSET VERGI$   = MKD$(VKN)
660 LSET ADRES$   = ADR$
670 LSET TELEFON$ = TEL$
680 LSET BRUT$    = MKS$(BR)
```

```
690 PUT 1,PN
700 IF PN>ADET THEN ADET=PN:OPEN"O",2,
"PERSADET":WRITE#2,PN:CLOSE 2
710 GOTO 500
1000 REM *** BÜTÜN PERSONELİN ÜCRET BORDROSU ***
1010 CLS
1020 PRINT "-------------------------------------------------------------"
1030 FOR P=1 TO ADET
1040 GET 1,P
1050 GOSUB 8000
1060 PRINT "-------------------------------------------------------"
1070 PRINT "   DEVAM ETMEK İÇİN ARA TUŞUNA BASINIZ."
1080 PRINT    "-------------------------------------------------------------"
1090 IF INKEY$<>" "THEN 1090
1100 NEXT P
1110 RETURN
1500 REM ******* İSME GÖRE ÜCRET BORDROSU *******
1510 CLS
1520 INPUT "KAYDI ARANAN PERSONELİN ADI ";ARA$
1530 PRINT "-------------------------------------------------------"
1540 FOR P=1 TO ADET
1550 GET 1,P
1560 IF INSTR(AD$,ARA$)>Ø THEN GOSUB 8000
1570 PRINT "-------------------------------------------------------"
1580 PRINT " MENÜYE DÖNMEK İÇİN ARA TUŞUNA
BASINIZ."
1590 IF INKEY$<>" "THEN 1590
1600 RETURN
2000 REM ********* PERSONEL ÖZEL BİLGİLERİ ********
2010 CLS
2020 PRINT "KAYDI ARANAN PERSONELİN NUMARASI ";P
2030 PRINT "-------------------------------------------------------"
2040 GET 1,P
2050 PRINT "ADI - SOYADI                    ";AD$
2060 PRINT "SİGORTA NUMARASI                ";CVD(SIGORTA$)
2070 PRINT "VERGİ KARNESİ NUMARASI          ";CVD(VERGI$)
2080 PRINT "ADRESİ                          ";ADRES$
2090 PRINT "TELEFON NUMARASI                ";TELEFON$
2100 PRINT "BRÜT ÜCRETİ                     ";CVS(BRUT$)
2110 PRINT "-------------------------------------------------------"
```

```
2120 PRINT " MENÜYE DÖNMEK İÇİN ARA TUŞUNA
BASINIZ."
2130 IF INKEY$<>" "THEN 2130
2140 RETURN
2500 REM  ****BÜTÜN PERSONELİN İSİM VE
MAAŞLARININ LİSTESİ****
2510 CLS
2520 PRINT "                PERSONEL LİSTESİ"
2530 PRINT "-------------------------------------------------"
2540 PRINT "PERSONEL ADI-SOYADI          MAAŞ"
2550 PRINT "-----------------------------  ---------"
2560 FOR P=1 TO ADET
2570 GET 1,P
2580 PRINT AD$;USING"#,###,###.##";CVS(MAAS$)
2590 IF P/20  <> INT(P/20) THEN 2660
2600 PRINT "DEVAM ETMEK İÇİN ARA TUŞUNA BASINIZ.";
2610 IF INKEY$<>" " THEN 2610
2620 PRINT "                PERSONEL LİSTESİ"
2630 PRINT "-------------------------------------------------"
2640 PRINT "PERSONEL ADI-SOYADI          MAAŞ"
2650 PRINT "-----------------------------  ---------"
2660 NEXT P
2670 PRINT "MENÜYE DÖNMEK İÇİN ARA TUŞUNA
BASINIZ."
2680 IF INKEY$<>" " THEN 2680
2690 RETURN
3000 REM PERSONEL BİLGİLERİYLE İLGİLİ DEĞİŞİKLİKLER
3010 CLS
3020 INPUT "BİLGİLERİ DEĞİŞECEK PERSONELİN
NUMARASI ";PN
3030 PRINT "-------------------------------------------------"
3040 GET 1,PN
3050 PRINT "ADI - SOYADI                ";AD$
3060 PRINT "SİGORTA NUMARASI            ";CVD(SIGORTA$)
3070 PRINT "VERGİ KARNESİ NUMARASI      ";CVD(VERGI$)
3080 PRINT "ADRESİ                      ";ADRES$
3090 PRINT "TELEFON NUMARASI            ";TELEFON$
3100 PRINT "BRÜT ÜCRETİ                 ";CVS(BRUT$)
3110 PRINT "-------------------------------------------------"
3120 PRINT "          YENİ BİLGİLERİ GİRİNİZ."
```

```
3130 PRINT
3140 PRINT " DEĞİŞMEYEN BİLGİLER İÇİN BİRŞEY
YAZMADAN"
3150 PRINT "              ENTER TUŞUNA BASINIZ."
3160 PRINT
3170 INPUT "ADI - SOYADI                    ";A$
3180 INPUT "SİGORTA NUMARASI                ";SN
3190 INPUT "VERGİ KARNESİ NUMARASI          ";VKN
3200 LINE INPUT "ADRESİ                     ";ADR$
3210 INPUT "TELEFON NUMARASI                ";TEL$
3220 INPUT "BRÜT ÜCRETİ                     ";BR
3230 IF A$<>"" THEN LSET AD$=A$
3240 IF SN<>Ø THEN LSET SIGORTA$=MKD$(SN)
3250 IF VKN<>Ø THEN LSET VERGI$=MKD$(VKN)
3260 IF ADR$<>"" THEN LSET ADRES$=ADR$
3270 IF BR<>Ø THEN LSET BRUT$=MKD$(BR)
3280 PUT 1,PN
3290 PRINT
3300 PRINT "YENİ BİLGİLER DOSYAYA KAYDEDİLDİ..."
3310 FOR N=1 TO 2000
3320 NEXT N
3330 RETURN
3500 REM ********* SABİT DEĞERLERİN LİSTESİ *******
3510 CLS
3520 OPEN"I",2,"SABIT"
3530 INPUT #2,SSTM,TTISVY,TTISCY,SSKY,GVY,DVY,GIG
3540 CLOSE 2
3550 PRINT "PROGRAM TARAFINDAN KULLANILAN SABİT
DEĞERLER:"
3560 PRINT "-----------------------------------------"
3570 PRINT "Sosyal sigorta tavan matrahı          :";SSTM
3580 PRINT "Tasarruf teşvik işveren hissesi (%)   :";TTISVY
3590 PRINT "Tasarruf teşvik işçi hisssesi (%)     :";TTISCY
3600 PRINT "Sosyal sigorta primi yüzdesi          :";SSKY
3610 PRINT "Gelir vergisi yüzdesi                 :";GVY
3620 PRINT "Damga vergisi yüzdesi                 :";DVY
3630 PRINT "Genel gider (günlük)                  :";GIG
3640 PRINT "-----------------------------------------"
3650 PRINT " MENÜYE DÖNMEK İÇİN ARA TUŞUNA
BASINIZ."
```

```
3660 IF INKEY$<>" " THEN 3660
3670 RETURN
4000 REM ** SABİT DEĞERLERLE İLGİLİ DEĞİŞİKLİKLER **
4010 CLS
4020 OPEN"I",2,"SABIT"
4030 INPUT #2,SSTM,TTISVY,TTISCY,SSKY,GVY,DVY,GIG
4040 CLOSE 2
4050 PRINT "PROGRAM TARAFINDAN KULLANILAN SABİT
DEĞERLER:"
4060 PRINT "-------------------------------------------"
4070 PRINT "Sosyal sigorta tavan matrahı          :";SSTM
4080 PRINT "Tasarruf teşvik işveren hissesi (%)   :";TTISVY
4090 PRINT "Tasarruf teşvik işçi hisssesi (%)     :";TTISCY
4100 PRINT "Sosyal sigorta primi yüzdesi          :";SSKY
4110 PRINT "Gelir vergisi yüzdesi                 :";GVY
4120 PRINT "Damga vergisi yüzdesi                 :";DVY
4130 PRINT "Genel gider (günlük)                  :";GIG
4140 PRINT "-------------------------------------------"
4150 PRINT "        YENİ BİLGİLERİ GİRİNİZ."
4160 PRINT " DEĞİŞMEYEN BİLGİLER İÇİN BİRŞEY
YAZMADAN"
4170 PRINT "          ENTER TUŞUNA BASINIZ."
4180 PRINT
4190 INPUT "Sosyal sigorta tavan matrahı          ";TM
4200 INPUT "Tasarruf teşvik işveren hissesi (%)   ";ISV
4210 INPUT "Tasarruf teşvik işçi hissesi (%)      ";ISC
4220 INPUT "Sosyal sigorta primi yüzdesi          ";SS
4230 INPUT "Gelir vergisi yüzdesi                 ";GV
4240 INPUT "Damga vergisi yüzdesi                 ";DV
4250 INPUT "Genel gider (günlük)                  ";GG
4260 IF TM<>Ø THEN SSTM=TM
4270 IF ISV<>Ø THEN TTISVY=ISV
4280 IF ISC<>Ø THEN TTISCY=ISC
4290 IF SS<>Ø THEN SSKY=SS
4300 IF GV<>Ø THEN GVY=GV
4310 IF DV<>Ø THEN DVY=DV
4320 IF GG<>Ø THEN GIG=GG
4330 OPEN"O",2,"SABIT"
4340 WRITE#2,SSTM,TTISVY,TTISCY,SSKY,GVY,DVY,GIG
4350 CLOSE 2
```

```
4360 PRINT "------------------------------------------------------"
4370 PRINT " YENİ BİLGİLER DOSYAYA KAYDEDİLMİŞTİR."
4380 FOR B=1 TO 2000
4390 NEXT B
4400 RETURN
4500 REM ******************** ÇIKIŞ ********************
4510 CLS
4520 PRINT "PROGRAM BİTMİŞTİR."
4530 PRINT "   İYİ GÜNLER."
4540 CLOSE
4550 END
8000 BRM = CVS(BRUT$) : REM BRÜT MAAŞ
8010 SSMATRAH = BRM : REM SOSYAL SİGORTA MATR.
8020 IF SSMATRAH >SSTM THEN SSMATRAH = SSTM :
REM TAVANLA KARŞILAŞTIRMA
8030 SSKPRIM = SSMATRAH * SSKY : REM SSK PRİMİ
8040 TTISV = BRM * TTISVY : REM TAS.TEV. İVEREN HİS.
8050 TTISC = BRM * TTISCY : REM TAS. TEV. İÇİ HİS.
8060 GNIND = 30 * GIG : REM GENEL İNDİRİM
8070 GVM = BRM - ( SSKPRIM + TTISC + GNIND )
: REM GELİR VERGİSİ MATRAHI
8080 GV = GVM * GVY : REM GELİR VERGİSİ
8090 DV = BRM * DVY : REM DAMGA VERGİSİ
8100 KESTOP = SSKPRIM + TTISC + GV + DV
: REM KESİNTİLER TOPLAMI
8110 NET = BRM - KESTOP : REM NET MAAŞ
8120 PRINT
8130 PRINT "Personel no.....:";P
8140 PRINT "Personel adı soyadı.....: ";AD$
8150 PRINT "Sigorta sicil no.....:";CVD(SIGORTA$)
8160 PRINT "Vergi karnesi no.....:";CVD(VERGI$)
8170 PRINT "Sosyal sigorta tavan matrahı.....:";SSTM
8180 PRINT "SSK primi.....:";SSKPRIM
8190 PRINT "Brüt ücret .....:";BRM
8200 PRINT "Tasarruf teşvik işveren hissesi.....:";TTISV
8210 PRINT "Tasarruf teşvik işçi hissesi.....:";TTISC
8220 PRINT "Genel indirim.....:";GNIND
8230 PRINT "Gelir vergisi matrahı .....:";GVM
8240 PRINT "Gelir vergisi.....:";GV
```

```
8250 PRINT "Damga vergisi          :";DV
8260 PRINT "Kesintiler toplamı     :";KESTOP
8270 PRINT "NET ÜCRET              :";NET
8280 RETURN
```

2500 ve 2690 numaralı satırlar arasında yer alan 'bütün personelin isim ve maaşlarını listeleme' bölümünde, personel adedinin 20'den fazla olabileceği ve bir ekrana sığmayarak listenin ilk satırlarının görülemeyeceği düşünülerek, her 20 satırda bir program durdurularak kullanıcının ara tuşuna basana dek listeyi inceleyebilmesi olanağı sağlanmıştır.

PROJE - 2: STOK KONTROL

Bu projede, herhangi bir ticarî işletmenin stoklarındaki malların kontrolü yapılacaktır. Bu kontrol, mal giriş çıkışı, en düşük stok düzeyi ve alım satımlardaki gelir gider hesaplarını içermektedir.

En düşük stok düzeyi (asgari stok seviyesi), herhangi bir malın olağan gereksinimleri karşılayabilecek, önceden belirlenmiş stok miktarıdır. Bu düzeyin altına düşmek, ihtiyaç durumunda mal bulunmaması tehlikesine (yok satış) yol açabileceğinden, en düşük stok düzeyi, programda dikkatle izlenmesi gereken bir noktadır.

Bu ticarî işletmede, stok servisinden aşağıdaki listeler istenmektedir:

1 - Bütün malların listesi (adları, miktarları, birim fiyatları, tutarları ve en düşük stok düzeyleri)

2 - En düşük stok düzeyinin altına düşen malların listesi

3 - Bütün hareketlerin listesi (giriş veya çıkış, tarih, miktar, fiyat, tutar ve hareketlerle ilgili açıklamalar)

4 - Herhangi bir mala ait hareketlerin listesi

Bu programda, stoktaki mallar ve mal hareketleriyle ilgili olmak üzere iki dosya kullanılacaktır. Her iki dosya da rasgele erişimli olacaktır. Birinci dosyada kayıt numaraları stoktaki mal kodu, ikinci dosyada ise hareketin sıra numarası olarak kullanılacaktır.

'STOK' dosyasının kayıt deseni:

MAL ADI (MAL$)	MİKTARI (MIKTAR$)	BİRİM FİYATI (FIYAT$)	EN DÜŞÜK STOK DÜZEYİ (MINIMUM$)
20	4	8	4

'HAREKET' dosyasının kayıt deseni:

TARİH (TRH$)	MAL KODU (KOD$)	GİRİŞ/ÇIKIŞ (GC$)	MİKTAR (MKT$)	BİRİM FİYAT (FYT$)	AÇIKLAMA (ACK$)
10	2	1	4	8	15

NOT: Giriş / çıkış sahası için ayrılan bir karakterlik yere, girişi belirten G veya çıkışı belirten Ç kodlarından biri yazılacaktır.

Her dosya için kayıt adedini saklayabilmek için STOKADET ve HAREKET1 isimli sıralı dosyalar kullanılacaktır:

```
OPEN"O",1,"STOKADET"
WRITE#1,0
CLOSE 1

OPEN"O",1,"HAREKET1"
WRITE#1,0
CLOSE 1
```

ANA MENÜ:

1 - Mal hareketlerinin girişi
2 - Bütün malların listesi
3 - En düşük stok düzeyinin altındaki malların listesi
4 - Bütün hareketlerin listesi
5 - Herhangi bir mala ait hareketlerin listesi
6 - Herhangi bir mal için fiyat değişikliği
7 - Fiyatlara genel zam
8 - Herhangi bir mal için en düşük stok düzeyinde değişiklik
9 - Çıkış

```
10 OPEN"R",1,"STOK",36
20 OPEN"R",2,"HAREKET",40
30 FIELD #1,20 AS MAL$,4 AS MIKTAR$,8 AS FIYAT$,4 AS
MINIMUM$
40 FIELD #2,10 AS TRH$,2 AS KOD$,1 AS GC$,4 AS MKT$,8
AS FYT$,15 AS ACK$
50 OPEN"I",3,"STOKADET"
60 INPUT#3,MALADET
70 CLOSE 3
80 OPEN"I",3,"HAREKET1"
90 INPUT#3,HAREKETADET
100 CLOSE 3
110 CLS
120 PRINT "       ANA MENÜ:"
130 PRINT "-------------------------------------------------"
140 PRINT " 1 - Mal hareketlerinin girişi"
150 PRINT " 2 - Bütün malların listesi"
160 PRINT " 3 -En düşük stok düzeyinin altındaki malların
listesi"
170 PRINT " 4 -Bütün hareketlerin listesi"
180 PRINT " 5 -Herhangi bir mala ait hareketlerin listesi"
190 PRINT " 6 -Herhangi bir mal için fiyat değişikliği"
200 PRINT " 7 -Fiyatlara genel zam"
210 PRINT " 8 -Herhangi bir mal için en düşük stok
düzeyinde değişiklik"
220 PRINT " 9 - Çıkış"
230 PRINT
240 INPUT "SEÇİMİNİZ ";S
250 IF S>9 OR S<1 THEN 110
260 ON S GOSUB 500, 1000, 1500, 2000, 2500, 3000, 3500,
4000, 4500
270 GOTO 110
500 REM ********** MAL HAREKETLERİ GİRİŞİ **********
510 CLS
520 PRINT "1 - Yeni mal kayıt"
530 PRINT "2 - Mevcut mallarla ilgili hareket girişi"
540 PRINT "3 - Ana menüye dönüş"
550 PRINT
560 INPUT "SEÇİMİNİZ ";S
570 IF S>3 OR S<1 THEN 510
```

```
580 ON S GOSUB 5000,6000
590 RETURN
1000 REM ********** BÜTÜN MALLARIN LİSTESİ ********
1010 CLS
1020 PRINT"MAL KODU   MAL ADI";TAB(30);"MİKTARI",
"BİRİM FİYATI","EN DÜŞÜK       DÜZEY"
1030 PRINT STRING$(80,45)
1040 FOR M=1 TO MALADET
1050 GET 1,M
1060 PRINT M;TAB(13);MAL$;CVS(MIKTAR$), CVD(FIYAT$),
CVS (MINIMUM$)
1070 IF M/20 <> INT(M/20) THEN 1100
1080 PRINT "DEVAM ETMEK İÇİN ARA TUŞUNA BASINIZ."
1090 IF INKEY$<>" "THEN 1090
1100 NEXT M
1110 PRINT
1120 PRINT "ANA MENÜYE DÖNMEK İÇİN ARA TUŞUNA
BASINIZ."
1130 IF INKEY$<>" "THEN 1130
1140 RETURN
1500 REM EN DÜŞÜK DÜZEYİN ALTINDAKİ MALLARIN
LİSTESİ
1510 CLS
1520 PRINT"MAL KODU   MAL ADI";TAB(30);"MİKTARI",
"BİRİM FİYATI","EN DÜŞÜK       DÜZEY"
1530 PRINT STRING$(80,45)
1540 FOR M=1 TO MALADET
1550 GET 1,M
1560 DUSUK = CVS(MINIMUM$)
1570 MIKTAR = CVS(MIKTAR$)
1580 IF MIKTAR <DUSUK THEN PRINT M;TAB(13);MAL$;
MIKTAR,CVD(FIYAT$),DUSUK
1590 NEXT M
1600 PRINT
1610 PRINT "ANA MENÜYE DÖNMEK İÇİN ARA TUŞUNA
BASINIZ."
1620 IF INKEY$<>" "THEN 1620
1630 RETURN
2000 REM ******** BÜTÜN HAREKETLERİN LİSTESİ *******
2010 CLS
```

```
2020 PRINT "SIRA  TARİH  MAL KODU  G/Ç  MİKTAR
BİRİM FİYAT  TUTAR  AÇIKLAMA"
2030 PRINT STRING$(80,45)
2040 FOR H=1 TO HAREKETADET
2050 GET 2,H
2060 MIKTAR=CVS(MIKTAR$)
2070 FIYAT#=CVD(FIYAT$)
2080 TUTAR#=MIKTAR * FIYAT#
2090 PRINT H;TAB(6);TARIH$; TAB(17); CVI(KOD$);TAB(29);
GC$; TAB(33); MIKTAR; TAB(42);FIYAT#;TAB(54);TUTAR#;
TAB(65); ACK$
2100 IF H/20<>INT(H/20) THEN 2130
2110 PRINT "DEVAM ETMEK İÇİN ARA TUŞUNA BASINIZ."
2120 IF INKEY$<>" "THEN 2120
2130 NEXT H
2140 PRINT
2150 PRINT "ANA MENÜYE DÖNMEK İÇİN ARA TUŞUNA
BASINIZ."
2160 IF INKEY$<>" "THEN 2160
2170 RETURN
2500 REM * HERHANGİ BİR MALA AİT HAREKETLER
LİSTESİ *
2510 CLS
2520 INPUT "HAREKETLERİNİ GÖRMEK İSTEDİĞİNİZ MALIN
KOD NUMARASI ";KN
2530 PRINT "------------------------------------------------------------"
2540 PRINT "SIRA  TARİH  MAL KODU  G/Ç  MİKTAR BİRİM
FİYAT  TUTAR          AÇIKLAMA"
2550 PRINT STRING$(80,45)
2560 FOR H=1 TO HAREKETADET
2570 GET 2,H
2580 IF CVI(KOD$)<>KN THEN 2630
2590 MIKTAR=CVS(MIKTAR$)
2600 FIYAT#=CVD(FIYAT$)
2610 TUTAR#=MIKTAR * FIYAT#
2620 PRINT H;TAB(6);TARIH$; TAB(17); CVI(KOD$);TAB(29);
GC$; TAB(33); MIKTAR; TAB(42);FIYAT#;TAB(54);
TUTAR#;TAB(65);ACK$
2630 NEXT H
```

```
2640 PRINT "ANA MENÜYE DÖNMEK İÇİN ARA TUŞUNA
BASINIZ."
2650 IF INKEY$<>" "THEN 2650
2660 RETURN
3000 REM **HERHANGİ BİR MALA AİT FİYAT DEĞİŞİKLİĞİ**
3010 CLS
3020 INPUT "FİYATI DEĞİŞEN MALIN KOD NUMARASI ";KN
3030 PRINT "-------------------------------------------------"
3040 GET 1,KN
3050 PRINT "MAL ADI                : ";MAL$
3060 PRINT "MALIN FİYATI           :";CVD(FIYAT$)
3070 INPUT "MALIN YENİ FİYATI      : ",FT
3080 LSET FIYAT$=MKD$(FT)
3090 PUT 1,KN
3100 PRINT "YENİ FİYAT KAYDEDİLMİŞTİR."
3110 FOR X=1 TO 2000:NEXT
3120 RETURN
3500 REM ************ FİYATLARA GENEL ZAM ***********
3510 CLS
3520 PRINT "ÖRNEK: YÜZDE 50 İÇİN GİRİŞ : 50"
3530 INPUT "FİYATLARA YAPILACAK GENEL ZAM YÜZDESİ
";ZY
3540 PRINT
3550 PRINT "ZAM YAPILIYOR..."
3560 FOR M=1 TO MALADET
3570 GET 1,M
3580 LSET FIYAT$=CVD(FIYAT$)+CVD(FIYAT$) / 100 * ZY
3590 PUT 1,M
3600 NEXT M
3610 PRINT "YENİ FİYATLAR KAYDEDİLMİŞTİR."
3620 FOR B=1 TO 2000:NEXT B
3630 RETURN
4000 REM *** EN DÜŞÜK STOK DÜZEYİNDE DEĞİİKLİK ****
4010 CLS
4020 INPUT "EN DÜŞÜK STOK DÜZEYİ DEĞİŞECEK MALIN
KOD  NUMARASI ";KN
4030 prınt "-----------------------------------------------------------------"
4040 GET, 1,KN
4050 PRINT "MAL ADI                    : ";MAL$
4060  PRINT "EN DÜŞÜK STOK DÜZEYİ :";CVS(MINIMUM$)
```

```
4070 INPUT "YENİ DÜZEY            :",YD
4080 LSET MININUM$=MKS$(YD)
4090 PUT 1, KN
4100 PRINT "YENİ DÜZEY KAYDEDİLMİŞTİR."
4110 FOR X=1 TO 2000: NEXT
4120 RETURN
4500 REM ******************ÇIKIŞ**********************
4510 CLS
4520 PRINT "STOK KONTROL PROGRAMI SONA ERMİŞTİR."
4530 CLOSE 1,2
4540 END
5000 REM **************YENİ MAL KAYIT*****************
5010 CLS
5020 PRINT "STOKTAKİ MAL ADEDİ : ";MAL ADET
5030 PRINT  "------------------------------------------------------------"
5040 INPUT "YENİ MALIN ADI                    ";A$
5050 INPUT "SATIN ALINAN MİKTAR               ";M
5060 INPUT "BİRİM  FİYATI                     ";BF#
5070 INPUT "EN DÜŞÜK STOK DÜZEYİ              ";EDSD
5080 INPUT "TARİH (gg/aa/yyyy)                ";T$
5090 INPUT "AÇIKLAMA                          ";AC$
5100 PRINT
5110 INPUT "GİRİLEN BİLGİLER DOĞRU MU (E/H) ";CS
5120 IF CS<> "E" AND CS<> "e" THEN 5010
5130 LSET MAL$=A$
5140 LSET MIKTAR$=MKS$(MIKTAR)
5150 LSET FIYAT$=MKD$(BF#)
5160 LSET MINIMUM$=MKS$(EDSD)
5170 MALADET=MALADET+1
5180 PUT 1,MALADET
5190 OPEN"O",3,"STOKADET"
5200 WRITE#3, MALADET
5210 CLOSE 3
5220 LSET TRH$ =T$
5230 LSET KOD$=MKI$(MALADET)
5240 LSET GC$="G"
5250 LSET MKT$=MKS$(MIKTAR)
5260 LSET FYT$=MKD$(BF#)
5270 LSET ACK$=AC$
5280 HAREKETADET=HAREKETADET+1
```

```
5290 PUT 2, HAREKETADET
5300 OPEN"O",3,"HAREKETADET1"
5310 WRITE#3, HAREKETADET
5320 CLOSE 3
5330 INPUT "BAŞKA MAL KAYDEDİLECEK Mİ (E/H) ";E$
5340 IF E$="E" OR E$="e" THEN 5010
5350 RETURN
6000 REM **MEVCUT MALLARLA İLGİLİ HAREKET GİRİŞİ**
6010 CLS
6020 PRINT"STOKTA"; MALADET; "ADET MAL KAYITLIDIR."
6030 PRINT "ANA MENÜYE DÖNMEK İÇİN KOD NUMARASI
OLARAK 0 VERİNİZ."
6040 PRINT "----------"
6050 INPUT "İŞLEM YAPILACAK MALIN KOD NUMARASI
";KN
6060 İF KN<1 THEN RETURN
6070 GET 1, KN
6080 PRINT "MAL ADI                    : ";MAL$
6090 PRINT "MEVCUT MİKTAR              :";CVS(MIKTAR$)
6100 PRINT "EN DÜŞÜK STOK DÜZEYİ       :";CVS(MINIMUM$)
6110 PRINT
6120 INPUT "BU MALLA İLGİLİ İŞLEM YAPILACAK MI (E/H)
";EH$
6130 IF EH$<> "E" AND EH$<>"e" THEN 6010
6140 INPUT "TARİH (gg/aa/yyyy)         ";T$
6150  INPUT "GİRİŞ (G) veya ÇIKIŞ (Ç)  ";G$
6160 IF LEN (G$)<>1 OR INSTR ("GgÇç", G$)=0 THEN 6150
6170 INPUT "ADET                       ";AD
6180 INPUT "BİRİM FİYAT                ";BF#
6190 INPUT "AÇIKLAMA                   ";AC$
6200 PRINT
6210 INPUT "GİRİLEN BİLGİLER DOĞRU MU (E/H) ";EH$
6220 IF EH$<> "E" AND EH$<>"e" THEN 6120
6230 MIKTAR=AD
6240 IF G$="Ç" OR G$="ç" THEN MIKTAR=- MIKTAR
6250 LSET MIKTAR$=MKS$ (CVS(MIKTAR$)+MIKTAR)
6260 PUT 1,KN
6270 LSET TRH$=T$
6280 LSET KOD$=MKI$(KN)
6290 LSET GC$=G$
```

```
6300 LSET MKT$=MKS$ (AD)
6310 LSET FYT$= MKD$ (BF#)
6320 LSET ACK$=AC$
6330 HAREKETADET=HAREKETADET+1
6340 PUT 2, HAREKETADET
6350 OPEN "O",3,"HAREKETADET1"
6360 WRITE#3, HAREKETADET
6370 CLOSE 3
6380 GOTO 6010
```

PROJE SORULARI

SORU - 1: MÜŞTERİ KAYITLARI

Bu projede, bir ticarî işletmenin müşterilerine ait tüm bilgileri içeren bir dosyayla ilgili işlemler yapılmaktadır. Bu dosya oluşturulduktan sonra, müşterilerle ilgili çeşitli listeler elde edilebilir.

Müşteri dosyasında bulunması gerekecek bilgiler şunlardır:

1 - Müşteri adı
2 - Adresi
3 - Telefon numarası
4 - Doğum tarihi
5 - Vergi numarası
6 - Vergi dairesi
7 - Müşterinin satın almış olduğu malların tutarları toplamı

Programın çalışması sonucu olarak istenenler şunlardır:

1 -Bütün müşterilerin listesi (Yalnızca isimleri)
2 -Özel müşteriler listesi (Belirli bir miktarın üzerinde satın alma gücü olanlar)
3 - Herhangi bir şehir veya semte göre müşteri listesi
4 - Herhangi bir müşteriye ait bilgiler listesi
5 - Özel günler için müşteri adreslerinin listesi
6 - Doğum günlerine göre müşteri listesi

SORU -2 : FATURA TAKİBİ

Ticarî bir işletmede, yapılan her satış için satışı yapılan malın stok kayıtlarının yenilenmesi, satış yapılan müşteriyle ilgili kayıtların düzenlenmesi ve bu satışla ilgili faturanın kesilmesi gerekir.

Fatura üzerinde bulunması gereken bilgiler şunlardır:

1 - Müşteri adı
2 - Müşteri adresi
3 - Müşteri vergi numarası ve vergi dairesi
4 - Fatura numarası
5 - Fatura tarihi
6 - Satılan malın stok numarası ve adı
7 - Satışla ilgili açıklama
8 - Malın birim fiyatı ve adedi
9 - KDV
10 - Satış tutarı
11 - Fatura toplamı (rakamla ve yazıyla)

SORU - 3: HASTANE KAYITLARI

Bu projenin amacı, herhangi bir hastenin yalnızca hastalarla ilgili kayıtlarını düzenli bir biçimde saklamak ve çeşitli hareketlere göre gerekli işlemleri yapmaktır.

Hastalarla ilgili saklanacak olan bilgiler şunlardır:

1 - Hasta adı
2 - Doğum yılı
3 - Hastalığı
4 - Hastanede kaldığı bölüm
5 - Hastaneye geliş tarihi
6 - Doktoru
7 - Adresi
8 - Cinsiyeti
9 - Kan grubu
10 - Ağırlığı (kg)
11 - Boyu (cm)

Programdaki listeleme, tamamen kullanıcının isteğine bağlı olacaktır. Bütün sahalarla ilgili koşullar, program çalıştırıldıktan sonra kullanıcı tarafından verilecektir.

HATA MESAJLARI

NOT: İlk 20 hata mesajı I. kitapta verilmiştir.

21) Undefined user function

DEF FN komutu ile tanımlanmamış bir fonksiyon kullanılmak istenmiştir.

22) Direct statement in file

Disketten (veya kasetten) yüklenmek istenen bir programda satır numarasız bir komuta rastlanmıştır. (Dosyaların program gibi LOAD komutuyla yüklenmesi sonucunda görülür.)

23) Input past end

Bütün kayıtları okunmuş sıralı erişimli bir dosyadan daha fazla kayıt okunması istenmiştir.

24) Bad file mode

Okunmak üzere açılması istenen dosyanın türü yanlıştır. (Genellikle BASIC programlarının dosya gibi okunmak istenmesi sonucunda görülür.)

25) File already open

Zaten açık olan bir dosyanın yeniden açılması istenmiştir. (Genellikle dosya okumak veya yazmak için kullanılan döngüdeki bir yanlışlıktan kaynaklanır.)

26) WHILE without WEND

WHILE komutuna karşılık olarak WEND kullanılmamış veya WEND yanlış olarak yazılmıştır.

27) WEND without WHILE

WHILE kullanılmadan WEND yazılmış veya program akışı WHILE - WEND döngüsüne yanlış yerden aktarılmıştır.

28) Bad file number

Kullanılmak istenen dosya numarası normal sınırlar dışındadır veya daha önceden bir OPEN komutuyla bu dosya açılmamıştır.

29) Bad file name

Dosya adı kurallara uymamaktadır. (Dosya adı 8 harften, noktadan sonra istenirse koyulabilen ek 3 harften fazla olmamalıdır. Bu iki bölümde de değişken adlarında bulunmaması gereken nokta, virgül, boşluk veya ingiliz alfabesinde bulunmayan harfler (Ç, Ğ, İ, Ö, , Ü) kullanılmamalıdır.)

30) Field overflow

FIELD komutuyla belirlenen kayıt uzunluğu, OPEN komutuyla verilmiş uzunluktan fazladır, yada doğrudan erişimli bir dosyayı okuma veya yazma ile ilgili komutlardan biri hatalıdır. (Genellikle bu tür bir dosya sıralı dosya komutlarıyla kullanıldığında karşılaşılır.)

31) File not found

Ulaşılmak istenen dosya kayıtlı değildir veya daha önce silinmiştir.

32) File already exists

NAME komutuyla bir dosyaya verilecek yeni isim, başka bir dosyaya aittir.

33) Disk full

Kayıt yapılmak istenen disket tamamen doludur.

34) Bad record number

Doğrudan erişimli bir dosya içinde kullanılan kayıt numarası sınırlar dışındadır. (1'den küçük veya 32767'den büyüktür.)

35) Too many files

Diskete kaydedilmiş olan dosya sayısı çok fazladır.

36) Disk write protected

Korumalı bir diskete kayıt yapılmak istenmiştir.