# TEMEL BASIC

Yazan:
GÜZİNSAĞKAN
TUNÇ GERÇEK

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:** 10
**Bilgisayar Dizisi:** 7

İstanbul – 1991

# TEMEL BASIC

# İÇİNDEKİLER

# ÖNSÖZ

*İnsan zekasının en parlak buluşlarından biri olan bilgisayarlar, kuşkusuz günümüzde ve gelecekte insanlığın vazgeçilmez yardımcıları olma niteliğini taşımaktadır. Çok sayıda uygulama alanı bulunan ve bu uygulama alanları her geçen gün şaşırtıcı boyutlara ulaşan bilgisayarların sahip oldukları özelliklerin en belirgini ise diğer makinelerin aksine bilgisayarlarla anlaşılabilmesi, bir anlamda konuşulabilmesidir. Tıpkı yabancı dil konuşarak yabancı ülke insanlarıyla anlaşılabildiği gibi, bilgisayarlarla da anlaşabilmek için, onların anlayacağı şekilde özel olarak geliştirilmiş dilleri (programlama dilleri) beceriyle kullanabilmek gerekmektedir.*

*Elinizdeki kitap, günümüzde kullanım kolaylığı nedeniyle yaygın bir programlama dili olan BASIC dilinin temel komut ve kullanım kurallarını içermektedir. Kitabın yazımında, bilgisayarla tanışma heyecanını taşıyan geleceğin bilgisayar programcılarına en açık bir dille hitap edebilme amacı ön planda tutulmuştur. Konular teknik ayrıntılardan arındırılmış, temel bilgiler öğrenciye basit ve anlaşılır biçimde bol örnek verilmek suretiyle belirli bir sistem içinde aktarılmıştır.*

*Yararlı olabilmesi dileğiyle...*

*İstanbul, 1991*

# BİLGİSAYAR NEDİR ?

Bilgisayar, bilgileri çok hızlı ve hatasız işleme yeteneği olan elektronik bir makinedir. Bilgisayarın en önemli özellikleri, programlanabilir olması, bilgi saklayabilmesi, aritmetiksel ve mantıksal işlemleri çok hızlı ve doğru olarak yapabilmesi ve elde ettiği sonuçları, istenilen biçimlerde insanlara ulaştırabilmesidir.

Bir bilgisayar, çok küçük boyutta olabileceği gibi, çok büyük makinelerden, hatta uzak yerleşimlerde bulunan birçok bilgisayarın bileşiminden oluşan karmaşık bir bilgisayar sistemi şeklinde de olabilir.

Ondokuzuncu yüzyılda başlayan endüstrileşme hareketinden sonra insanlar bilgi toplamaya çok fazla önem vermişlerdir. Toplanan bilgilerin çok fazla olması da, insanın bu bilgileri işlemedeki yetersizliğini göstermiştir.

İnsanla bilgisayarın karşılaştırması yapılırsa, bilgisayarlar insana göre çok hızlı ve hatasız çalışırlar, fakat bellek kapasitesi yönünden insan beyninin çok üstün olduğu bir gerçektir. Anımsama konusunda insanda kesinlik olasılığı azdır; buna karşın, bilgisayar depoladığı bir bilgiyi kesin olarak anımsar. Sonuç olarak, bilgisayarların, birçok bakımdan insan beyninden çok daha yetersiz olduğu düşünülebilir. Bilgisayar, bütün diğer makineler gibi, insanın yönetim ve denetiminde, kendilerine verilen komutlara göre işlem yapan elektronik makinelerdir. Ancak, olağanüstü hız ve hatasızlık gerektiren işlemlerin milyonlarcası kısa bir sürede bilgisayar tarafından yapılabilmekte ve çeşitli kayıt ortamlarında biriktirilip saklanabilmektedir.

## BİLGİSAYARIN TARİHÇESİ :

Hesaplamanın temel prensipleri, Mısır ve Çinliler tarafından M.Ö. 600 yıllarında tespit edilmiş ve 'ABACUS' isimli bir hesaplama aracı geliştirilmiştir. Bu araç, çeşitli biçimlerde bugün bile kullanılmaktadır. Ondalık sistemin son şekli Araplar tarafından geliştirilmiş ve Avrupa'da 17'nci yüzyılda kullanılmaya başlanmıştır. 1617 yılında John Napier logaritmayı, 1621'de William Oughtred hesap cetvelini bulmuştur. İlk mekanik hesap makinesi, 1642 yılında Pascal tarafından yapılmıştır. Yalnızca toplama ve çıkarma işlemlerini yapabilen bu makine, 1671 yılında Leibniz tarafından çarpma ve bölme işlemlerini de yapabilecek şekilde geliştirilmiştir.

Mekanik hesap makineleri ticari amaçla 1800 yıllarından sonra imal edilerek piyasaya sürülmüştür. 1800 yılında, Jacquard isimli bir saatçi, ilk delikli kart sistemini bulmuştur. 1854 yılında, George Boole'un, 'Düşüncenin Kanunları' adlı eserinin yayınlanması, bilgisayarların gelişmesinde önemli bir aşamadır. 1889 yılında Amerikalı Hollerith ilk delikli kart makinesini imal ederek, 1890 yılında Amerika nüfus sayımında kullanmıştır. Hollerith daha sonra Computing-Recording Company (CTR) firmasını kurmuş, 1924 yılında ise bu firmanın ismi International Business Machine Corporation (IBM) olarak değiştirilmiştir. 1937 yılında, Howard-Aiken matematiksel işlemleri otomatik olarak yapabilen bir makine yapmıştır. 1946 yılında Pennsylvania Üniversitesi'nde Eckert ve Mauchly tarafından mevcut elektronik malzemeler kullanılarak dahilî hafızası olmayan bir hesap makinesi yapılmış ve adına ENIAC (Electronic Numerical Integrator and Calculator) denmiştir. İşlem hızı saniyede 5000 olan bu makinenin ağırlığı 30 ton ve kapladığı alan 130 metrekareydi. 1948 yılında transistörün kullanılması ile bilgisayar teknolojisi ilerlemeye başlamıştır. 1951 yılında UNIVAC-1 (Universal Atomic Computer) ticari amaçla kullanılan ilk elektronik hesaplayıcı olarak bilinir. 1952 yılında ikili sayı sistemini (Binary System) kullanan ve dahilî hafızaya sahip olan EDVAC bilgisayarı yapılmıştır.

İlk defa vakum tüplü ve yalnızca Assembler programlama dili kullanan birinci kuşak bilgisayarlar imal edilmiştir. Daha sonra, 1957 yılında W. Shockley tarafından transistörün bulunmasıyla 1960 yıllarında, lambalı bilgisayarların yerini transistörlü bilgisayarlar almıştır. Bunlar ikinci kuşak olup, ALGOL, COBOL ve FORTRAN gibi yüksek seviyeli programlama dilleri kullanmaktadırlar. Küçük bir alana birçok

transistör ve elektronik devrenin yerleştirilmesiyle gelişen MSI (Medium Scale Integration) entegre devre teknolojisi ile üçüncü kuşak bilgisayarlar imal edilmiştir. Bu bilgisayarlar, gerçek zaman (Real time), zaman paylaşma (Time sharing), uzaktan erişim (Remote access), vb. olanakları yanında küçülme, hızlanma ve ucuzlama gibi üstünlükleriyle daha yaygın bir kullanma alanı bulmuştur.

Dördüncü kuşak bilgisayarlara doğru giderken, mevcut bütünleşik devre teknolojisinin gelişmesi sonucu VLSI (Very Large Scale Integration) bütünleşik devrelerin kullanılması çok daha karmaşık ve yetenekli bilgisayarların ortaya çıkmasını sağlamıştır.

## ÇALIŞMA SİSTEMLERİNE GÖRE BİLGİSAYAR TÜRLERİ :

1 - Sayısal (Digital) bilgisayarlar:

Sayıya dönüştürülebilen bütün harf, sayı ve karakterleri (kesikli bilgiler) okuyarak, bunlarla işlem yapan bilgisayarlardır.

2 - Analog bilgisayarlar:

Basınç, sıcaklık, voltaj gibi devamlı (kesiksiz) olan değişmelerin işlenmesini gerektiren, fiziksel ve elektriksel değerleri ölçen bilgisayarlardır.

3 - Karma bilgisayarlar:

Sayısal ve analog bilgisayarların birleşimiyle oluşan bilgisayarlardır.

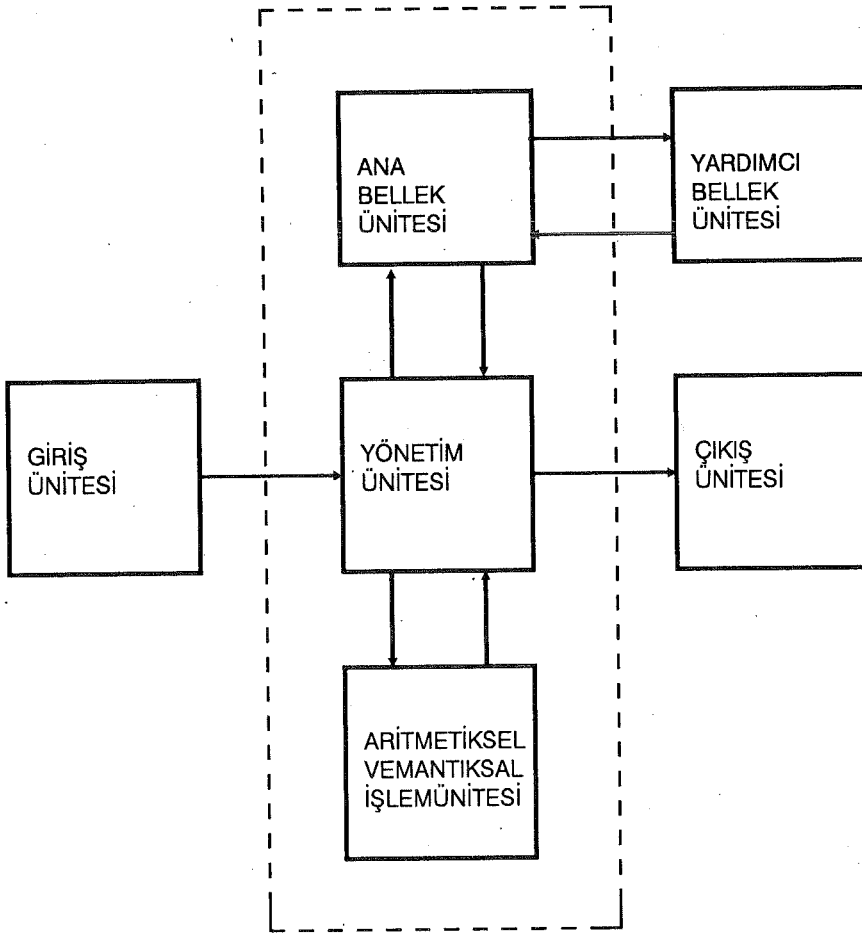
## BİLGİSAYARLARIN KULLANIM ALANLARI :

Çok fazla miktarda bilgi depolanmasını ve çok süratli erişimi gerektiren iş kollarında, bilgisayar kullanımı kaçınılmazdır. Bunlara örnek olarak bankalar, hastaneler, hava limanları, üniversiteler ve vergi daireleri gösterilebilir.

Bilgisayar kullanımının çok gerekli olduğu bir başka alan da, bilimsel araştırmalardır. Gerek çok karmaşık hesapların yapılmasında, gerekse doğal şartlarda gerçekleşmesi olanaksız veya çok uzun sürecek deneylerin bir bilgisayarın belleğinde uygulanmasında (simülasyon) bilgisayarlar laboratuvarların vazgeçilmez yardımcılarıdır.

## BİLGİSAYARIN YAPISI:

### BİLGİSAYARIN BLOK ŞEMASİ



Bilgisayar tarafından herhangi bir problemin program halinde çözümlenerek sonuçların dış ortama verilebilmesi için bilgisayar beş ana bölümden meydana gelmiştir:

**1 - Giriş üniteleri:**

Belirli bir ortamda kodlanmış olan bilgiyi okur ve bu bilgiyi sisteme iletirler. Giriş birimi, bilgisayarda klavye ve monitörden (ekran) meydana gelen bir terminal olabildiği gibi, manyetik teyp ve kasetler, manyetik disk veya disketler ve günümüzde pek fazla kullanılmayan delikli kart ve kağıt şerit okuyucusu olabilir.

**2 - Ana bellek ünitesi:**

Bilgisayara verilen bilgiler belleğe alınmaktadır. Bellek, gerek giriş bilgilerini, gerekse program içerisinde oluşturulan bilgileri bünyesinde bulunan hücrelere yerleştirir. Ana bellek içindeki bilgiler, manyetik disk, disket veya kasetlere yüklenerek saklanabilir.

Bellekteki bütün bilgiler, ikili sayma sistemine (binary) göre kodlanır ve saklanır. Diğer bir deyişle, bellekteki bilgiler 1'ler ve 0'lardan oluşmuştur.

Ana bellek iki bölüme ayrılır:

a)Bilgisayarın kullandığı dilin fonksiyonlarının yazıldığı ve hiçbir zaman silinmeyen ROM bölümü,

b)Programcının kullanımına sunulan, değişik programların depolandığı ve istenildiği zaman silinebilen RAM bölümü.

**3 - Yardımcı bellek ünitesi:**

Yardımcı bellek, bellek kapasitesini arttırma amacıyla kullanılır; giriş ve çıkış üniteleri ve merkezî işlem ünitesiyle doğrudan doğruya bağlı değildir. Yardımcı belleğe bilgi geliş ve gidişi, ana bellek kana-

lıyla yapılır. Günümüzde, bilgisayarların çoğunda yardımcı bellekler, manyetik şerit, disk ya da diskettir. Yardımcı bellek ünitelerinin sakıncası, düşük erişim hızlarıdır. Bunlar büyük bilgi gruplarının saklanması için kullanılır.

**4 - Yönetim ünitesi:**

Bilgisayar içinde işlemlerin yönetilmesini sağlar. Komutlar tarafından belirtilen tüm işlemleri idare ve koordine eder. Bu, giriş-çıkış ünitelerinin kontrolü, bellekten bilginin alınması, ya da belleğe verilmesi, bellekten komut alınması, bunların yorumlanması ve istenildiği şekilde yapılması işlemlerini de kapsar. Bu ünite, bilgisayarların otomatik olarak ve bir bütün halinde çalışmasını sağlar.

**5 - Aritmetik ve mantıksal işlem ünitesi:**

Bilgisayarda yapılan karşılaştırma, toplama, çıkarma, çarpma, bölme gibi işlemler, bu ünitenin elektronik mantık devreleri ile yapılmaktadır.

**6 - Çıkış üniteleri:**

Çıkış üniteleri, bilgiyi sistemden belirli bir ortama çıkarırlar. Bu amaçla, çıkış bilgilerinin kaydedildiği birçok çıkış ünitesi geliştirilmiştir. Bunlar yazıcılar, XY çiziciler (plotter), ekranlar, disk ve disketler, kağıt şerit deliciler ve kart delicilerdir.

# PROGRAMLAMANIN TANIMI VE PROGRAMLAMA DİLLERİ:

Bir problemin çözümüne ulaşabilmek için gerekli adımların herhangi bir bilgisayar dilinde kodlanmasına programlama denir. Bilgisayar programlama dilleri başlıca üç grupta toplanabilir:

1 - Makine dilleri

2 - Birleştirici diller

3 - Üst düzey diller

1 - Bilindiği gibi bilgisayarlar ikili (binary) sistem kullanılmasını gerektiren, elektrik akımının varlığı veya yokluğu temeline bağlı olarak çalışırlar. Akım geçtiğinde 1, geçmediğinde 0 sayısal değeri verilmiş olur. Bu sisteme göre çalışan makine dili programların bütün komutları da 1'ler ve 0'ların kombinasyonlarından oluşmalıdır. Bu dil, bilgisayarın işlemlerine doğrudan hükmeden en alt seviyeli dildir. Makine dilinde yazılan programlar en hızlı çalışan programlardır. Buna karşın, bu programlama dilini öğrenmek ve kullanmak, diğer dillere oranla çok daha zordur.

2 - Makine dilinin kullanımındaki zorluğu ortadan kaldırmak için geliştirilen birleştirici diller (assembler), basit komutların makine dilindeki karşılıklarına dönüştürülmesi amacıyla kullanılır. Birleştirici diller, kullanılan bilgisayara göre farklılıklar gösterebilirler.

3 - Üst düzey diller, programlamanın çok daha kolay bir hale getirilmesi için geliştirilmiştir. Bu dillerde kullanılan program deyimleri, derleyici (compiler) veya yorumlayıcı (interpreter) tarafından makine diline çevrilir. Üst düzey diller genel olarak iki ana bölümde toplanabilir: A) İşleme yönelik diller, B) Özel amaçlı diller.

**A) İşleme yönelik diller:** Bu tür dillerin en büyük özelliği, komutlarının birçoğunun makine dilindeki bir dizi komuta eşdeğer olmasıdır. Dolayısıyla, İngilizce ve matematiksel işlemlere benzerlik gösteren bu terimleri kullanmak programlamayı çok kolaylaştırır. Günümüzde en yaygın olarak kullanılan işleme yönelik diller, FORTRAN, BASIC, PL/1, ALGOL, COBOL ve PASCAL'dır.

**B) Özel amaçlı diller:** Bunlar, işleme yönelik olmayan, özel amaçlara hizmet etmek üzere geliştirilmiş dillerdir. Bu türün en yaygın olarak kullanılan örneği, RPG dilidir.

**Derleyici ve Yorumlayıcı Kavramları:**

Üst düzey dillerde yazılmış programlardaki komutlar, bilgisayar tarafından doğrudan doğruya anlaşılmaz. Bu komutların anlaşılıp uygulanabilmesi için, bir program yardımıyla, makine diline dönüştürülmesi gerekir. Kaynak (source) programı alt seviyedeki makine dilinde yazılmış amaç (object) programa çevirmek için gerekli olan bu yardımcı programlar iki ana bölümde toplanır:

1) Derleyiciler (compiler)
2) Yorumlayıcılar (interpreter).

**1) Derleyiciler:** Kaynak programı tümüyle ana belleğe aktarıp, daha sonra makine dilinde yazılmış amaç programa dönüştüren programlardır. Her dil için, o dile ait özel bir derleyici program kullanılmaktadır. Eğer bir programda derleme hatası bulunursa, o program düzeltilmeden önce çalıştırılamaz.

**2) Yorumlayıcılar:** Kaynak programı bölüm bölüm değerlendirerek amaç programa dönüştüren programlardır. Derleyicilerden farklı olarak, kaynak programın doğruluk kontrolü, çalıştığı süre içinde ya-

pılır. Herhangi bir hata ile karşılaşılıncaya kadar, programın çalışması devam eder.

Yukarıdaki açıklamalardan da anlaşılabileceği gibi, derleyici yoluyla makine diline dönüştürülmüş programlar, yorumlayıcı kullanılarak makine diline çevrilmiş programlardan daha hızlı çalışırlar. Bunun nedeni, derleyici programların hata kontrolünü program çalıştırılmadan önce yapması ve programın çalışma süresinde yorumlayıcı programlar gibi hata aramakla zaman kaybetmemesidir. Buna karşın, yorumlayıcı programların avantajı ise yanlış programların dahi çalıştırılabilmesi özelliği sayesinde programcıya deneme yanılma olanağı sağlamasıdır.

**Programlama Dilleri:**

Dünyada yaygın olarak kullanılan 10-15 kadar dil vardır. Bu dillerden en popüler olanları aşağıda listelenmiştir.

**BASIC** (Beginners' All-Purpose Symbolic Instruction Code) dili, programcılık öğrenimine yeni başlayanlar için geliştirilmiştir. Öğrenim kolaylığının yanısıra, özellikle son yıllarda bu dil üzerinde yapılan çalışmalarla, hemen her konuda kullanılabilir hale gelmiştir.

**COBOL** (Common Business Oriented Language), özellikle ticari uygulamalarda kullanılmak amacıyla geliştirilmiş ilk bilgisayar dilidir. Dosyalama uygulamalarında COBOL dili büyük kolaylıklar getirmektedir.

**FORTRAN** (Formula Translator), bilimsel çalışmalara yönelik bir dildir. 20 yılı aşkın bir süredir kullanılmaktadır. Veri giriş ve çıkışlarındaki hassasiyeti bu dilin özelliklerinden biridir.

**PASCAL,** diğer dillere oranla oldukça yeni sayılabilecek bir dildir. 1970'lerin başında geliştirilmiş olan bu dil, programcılık eğitimine yöneliktir.

**APL** (A Programming Language), bilimsel amaçlı bir dildir. Çok çeşitli karakterlerin kullanımına olanak veren bu dilin, özel klavye ve yazıcı gerektirdiğinden, kurulmuş bir sisteme adapte edilebilmesi oldukça masraflıdır.

**PL/1** (Programming Language - 1), IBM tarafından tasarlanmış, kullanımı oldukça zor olan bir dildir. ALGOL, COBOL ve FORTRAN dillerinin bütün özelliklerini içermekte olan PL/1, diğer dillere göre çok esnek bir yapıya sahiptir.

**ALGOL** (Algorithmic Language), matematiksel problemlerin çözümüne yönelik bir dildir.

**RPG** (Report Program Generator), kullanım amacı açısından COBOL diline benzeyen, veri dosyalarını işleme konusunda çeşitli kolaylıklar getiren bir dildir.

Görüldüğü gibi, hemen hemen bütün dillerin diğerlerine göre çeşitli farkları vardır. Bu farklar kimi zaman avantaj, kimi zaman ise dezavantaj olarak ortaya çıkmaktadır. İyi bir bilgisayar programcısı olabilmek, programlama mantığını tam olarak kavrayabilmenin yanısıra, programın amacına en uygun olan dili seçerek kullanabilmeyi de gerektirmektedir. Bu da, mümkün olduğu kadar çok sayıda programlama dilini öğrenerek bu dillerden etkin bir biçimde yararlanabilme yeteneğiyle gerçekleşebilir.

Programcılık yolunda atılacak ilk adım, çok sayıda dilin hepsini aynı anda öğrenmeye çalışmak yerine, bu dillerden yalnızca biriyle başlamak, bu dilde yeterli seviyede öğrenim gördükten sonra, istenilen başka bir dile geçmek olacaktır.

Daha önce de belirtildiği gibi, programcılık eğitimine başlangıç için en uygun dil BASIC dilidir. Yorumlayıcı kullanımıyla yeni öğrenciye deneme-yanılma olanağı sağlamasından başka, yapısındaki esneklik ve her amaca uygun olma özelliği, bu dilin öğrenimine büyük bir kolaylık ve sürat kazandırmaktadır. BASIC öğrenerek rahatça geliştirilebilecek programlama mantığı, daha sonra diğer dillerin eğitimi ve profesyonel programcılığa geçiş için atılacak en uygun adım olarak görülmektedir.

# İŞLETİM SİSTEMLERİ :

İşletim sistemi, kullanıcıya, bilgisayardan etkin biçimde yararlanma olanağı veren büyük bir program olarak düşünülebilir. Bilgisayarın kaynaklarının kullanımını sağlayan bir program ve prosedürler toplamıdır. Bu tür programlara denetleyici ya da yönetici program denir. Dünyada yaygın olarak kullanılan çeşitli işletim sistemleri vardır. CP/M, MS-DOS, UNIX bunların en tanınmışlarındandır. Bilgisayar üreticisi firmalar tarafından geliştirilen bu programlar, bilgisayarın tüm işlemlerini, işlevlerini ve iş akışını denetlerler.

**Tekli Kullanım ve Çoklu Kullanım:**

Bilgisayarlardan yararlanma, erişim noktaları (klavyeler, ekranlar, yazıcılar, disk veya disket okuyucular, vs.) yoluyla olur. Bazı bilgisayarlarda, aynı anda yalnızca tek bir erişim noktası kullanılabilir. Bu tür çalışmaya, tekli kullanım (single user) denir.

Bazı bilgisayarlarda ise, aynı anda birden çok erişim noktası faaliyet gösterebilir. Bu çeşit kullanıma da çoklu kullanım (multi user) adı verilir.

**Çevrim İçi ve Çevrim Dışı İşlemler:**

Çevrim içi (On-line) işlem, bilgi işlemin, tamamıyla ana işlem biriminin denetimi altındaki donanımla yürütülmesi olarak tanımlanabilir.

Giriş-çıkış üniteleri, terminaller, ana işlem biriminin denetimi altındadır ve işlenecek bilgiler, programlar, bu sistem içinde anında işlenebilirler.

Çevrim dışı (Off-line) işlem ise, ana işlem birimiyle doğrudan doğruya bağıntılı olmayan bilgi işlem donanımı üzerinde, bir dizi bilgi işlem çalışmalarının yürütülmesidir.

## ÇOK KULLANILAN BAZI TUŞLARIN GÖREVLERİ:

### RETURN / ENTER :

Bilgisayara verilmek istenen mesaj veya komutlar yazıldıktan sonra, bunları bilgisayara iletebilmek için kullanılan tuşlardır. RETURN veya ENTER tuşlarının görevleri aynıdır. Bu tuşlardan herhangi birine basıldığında, kursör ekranda bulunduğu yerden bir alt satırın başına geçer. ( Kursör (cursor) , yazılmak istenen karakterlerin (harf, sayı ve diğer özel işaretler) ekranın neresine yazılacağını belirleyen göstericidir. Kursör, kullanılan bilgisayara göre, dikdörtgen veya çizgi şeklinde olabilir; sabit durabilir veya yanıp sönebilir.) Fakat, bu tuşların asıl görevi, yazılacak olan bilginin belleğe aktarılmasıdır. Eğer yazılan bilgi ekrandaki bir satıra sığmazsa, alt satıra geçmek için RETURN tuşuna basmak yanlıştır. Satır dolduğunda, kursör kendiliğinden bir alt satıra geçecektir. Bunun tam tersi olarak, yazılacak bilgi tam bir ekran satırını dolduracak uzunluktaysa, kursör bilgi bitiminde alttaki satıra geçse dahi, RETURN tuşuna basılmalıdır.

**CLR (CLEAR) :**

Kursörün üzerinde bulunduğu karakteri silmek için kullanılan tuştur. Bu tuşa basıldığında, kursörün sağ tarafındaki karakterler bir sola kayarak, silinmek istenen karakterin yerini doldurur.

**DEL (DELETE) :**

En son yazılmış olan (kursörün sol tarafındaki) karakteri silmek için kullanılan tuştur.

**SHIFT :**

Klavyede iki adet SHIFT tuşu bulunur. İkisi de aynı görevleri görürler:

1 - Büyük harf yazımını sağlamak,

2 - Tuşların üst sırasındaki sembolleri kullandırmak.

Tuşlara ikinci bir özellik kazandırmaya yarayan bu tuş kullanılmak istendiğinde basılı tutularak özellik kazandırılacak tuşa basılır.

**CAPS LOCK (Büyük harf kilitleme - CAPITALS LOCK) :**

Sürekli büyük harf yazımını sağlamak için kullanılan tuştur. Bu tuşa bir kez basıldıktan sonra yazılan harfler büyük olarak çıkacaktır. Tuşa tekrar basılması, bilgisayarı eski konumuna getirecektir.

**CONTROL :**

Tuşlara üçüncü bir özellik kazandırılmak için kullanılan bu tuş, SHIFT tuşuyla benzer kullanıma sahiptir. Tuşlara programlanabilen farklı karakter veya karakter gruplarından yararlanabilmek için kullanılır.

## BİR PROBLEMİN BİLGİSAYAR YARDIMIYLA ÇÖZÜMÜNÜN ADIM ADIM İFADESİ:

1 - Problemin tanımlanması
2 - Gerekiyorsa akış şemasının çizilmesi
3 - Problemin, kullanılan programlama diline göre kodlanması
4 - Kodlanan programın bilgisayara girişinin yapılması
5 - Programın test verileriyle çalıştırılması
6 - Hata varsa, düzeltilerek 4 ve 5'inci adımların tekrarlanması
7 - Programın gerçek verilerle çalıştırılması
8 - Hata varsa, düzeltilerek 4'üncü adıma geri dönülmesi

## ALGORİTMA:

Algoritma, herhangi bir problemin çözümü için geliştirilmiş olan yöntem veya yöntemler topluluğudur. Algoritma kullanımındaki amaç, çeşitli verilerle istenilen sonuca ulaşmaktır. Program yazımından önce geliştirilen algoritma, ilk olarak test verileriyle denenir; eğer sonuç çıkmaz veya beklenilenden farklı bir sonuç çıkarsa, algoritma adımları yeniden gözden geçirilir. Bu işlem, doğru sonuçlar elde edilinceye kadar tekrarlanır; son olarak gerçek verilerle denendikten sonra algoritma programın temel mantığı olarak kabul edilir ve programın yazımına başlanır.

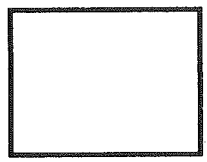
## AKIŞ ŞEMALARI:

Bir problemin çözümü için gereken işlemlerin, kararların ve bunların uygulanacağı sıranın çizimsel gösterimidir. Akış şemaları, bilgisayar tarafından anlaşılması için değil, işlemlerin akışını sistematik biçimde özetlemek için kullanılır. Özellikle basit programlar için akış şeması kullanmak gerekli olmayabilir.

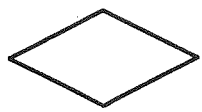
**Akış şemalarında genel olarak kullanılan semboller şunlardır**

Programın başlangıç ve bitişini gösteren semboldür.

Genel giriş ve çıkış işlemleri için kullanılır.

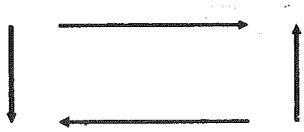


Ana bellekte yapılan aritmetik işlemler ve bilgi atamalar için kullanılır.



Test sembolü olup, karşılaştırma ve karar verme için kullanılır.

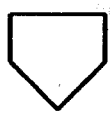
Döngüler için kullanılır.



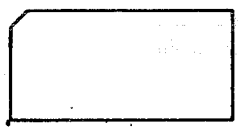
Programın akış yönünü göstermek için kullanılır.

**Akış şemalarında özel amaçla kullanılan bazı semboller şunlardır:**

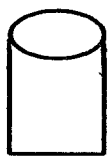
Sayfa içi bağlantı yapmak için kullanılır.



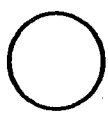
Sayfa dışı bağlantı yapmak için kullanılır.



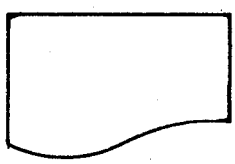
Delikli kart sembolü olup bilgi giriş–çıkışı için kullanılır.



Disk sembolü olup, bilgi giriş–çıkışı için kullanılır.

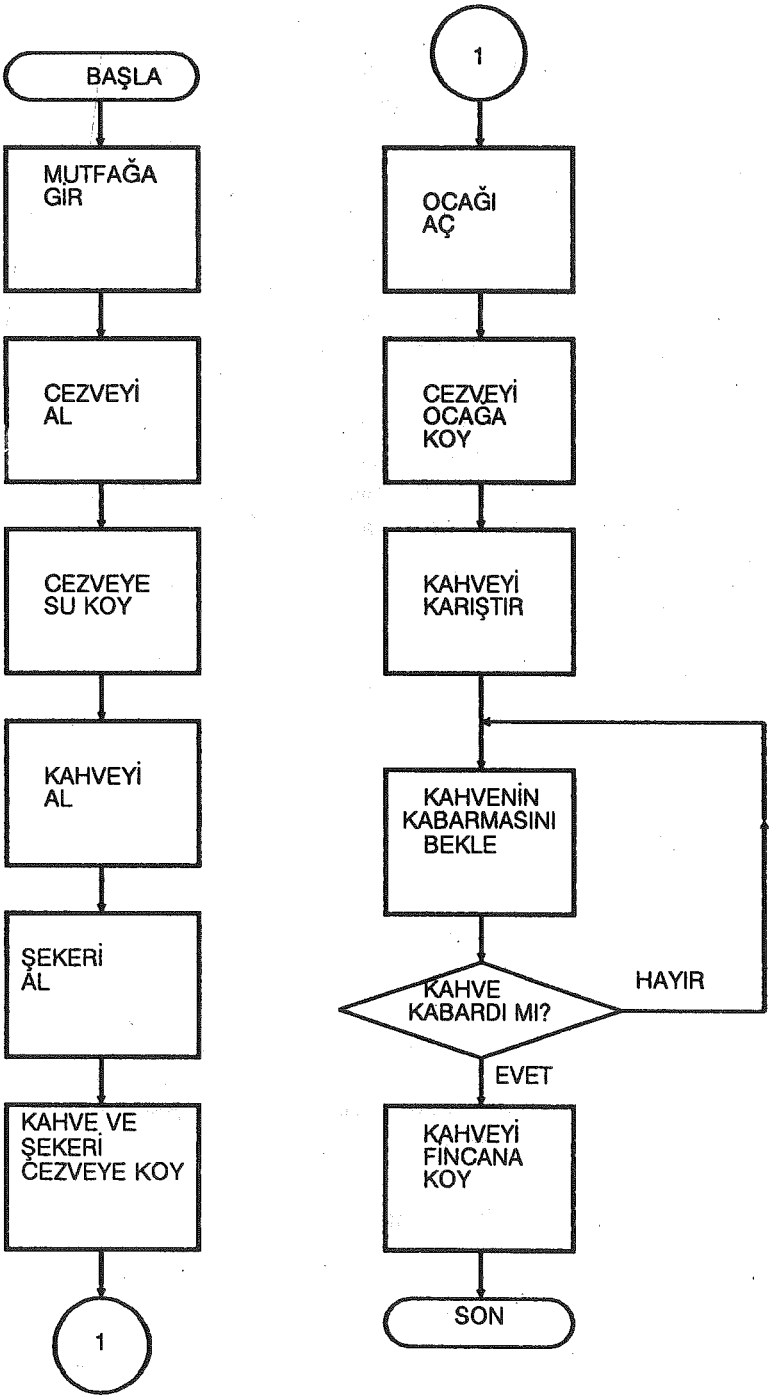


Disket sembolü olup bilgi giriş–çıkışı için kullanılır.



Yazıcı sembolü olup, bilgi çıkışı için kullanılır.

**Aktif şemalarıyla ilgili günlük yaşantıdan bir örnek: Kahve pişirme**



## SAYI SİSTEMLERİ :

Normalde kullandığımız sayı sistemi, 10 tabanlı sayı sistemidir. Bunun nedeni, herhangi bir sayıyı yazarken bir basamakta kullanılabilen en fazla 10 çeşit (0...9) rakam olmasıdır. Buna karşın, bilgisayarlar gibi elektronik hesaplayıcılarda, 2 tabanlı sayı sistemi (Binary) kullanılır. Yani, rakamlar 0 ve 1 olmak üzere yalnızca iki çeşittir.

Elektronik hesaplayıcılar, bütün işlemlerini devrelerindeki elektrik akımının varlığına veya yokluğuna göre yaparlar. Eğer herhangi bir devrede elektrik akımı varsa, bilgisayar için bunun değeri 1'dir; eğer akım yoksa, değeri 0'dır. Bu ikili yapının her bir elemanına bit denir. 8 bitin birleşmesiyle oluşan gruba da byte (bayt) denir. 1024 byte bir kilobyte'a (Kbyte), 1024 kilobyte ise 1 megabyte'a (Mbyte) eşittir.

Kullanıcının onlu sistemde yazmış olduğu komutları bilgisayar ikili sisteme dönüştürerek işler ve sonucu yine onlu sisteme çevirerek kullanıcıya verir. Bu nedenle, bilgisayarı kullanırken, işlemlerin ikili sistemde yapıldığını gözlemek olanaksızdır.

Onlu ve ikili sistemlerde basamakların sağdan sola diziliş şekilleri aşağıdaki gibidir:

| | | | | | |
|---|---|---|---|---|---|
| Onlu sistem : | $10^0$ | $10^1$ | $10^2$ | $10^3$ ... | |
| | 1 | 10 | 100 | 1000 | |
| İkili sistem: | $2^0$ | $2^1$ | $2^2$ | $2^3$ | $2^4$ ... |
| | 1 | 2 | 4 | 8 | 16 |

Onlu sistemdeki 101 sayısı (1 x 1) + (0 x 10) + (1 x 100) şeklinde tanımlanabiliyorsa, ikili sistemdeki 101 sayısı da yalnızca tabanı değiştirerek (1 x 1) + (0 x 2) + (1 x 4) olarak tanımlanabilir. Bu durumda, ikili sistemdeki 101 sayısı, onlu sisteme çevrildiğinde, sonuç olarak 5 (1 + 0 + 4) elde edilir.

1) + (0 x 2) + (1 x 4) olarak tanımlanabilir. Bu durumda, ikili sistemdeki 101 sayısı, onlu sisteme çevrildiğinde, sonuç olarak 5 (1 + 0 + 4) elde edilir.

$$101_{10} = 1 \times 10^0 + 0 \times 10^1 + 1 \times 10^2$$
$$= 1 + 0 + 100$$
$$= 101_{10}$$

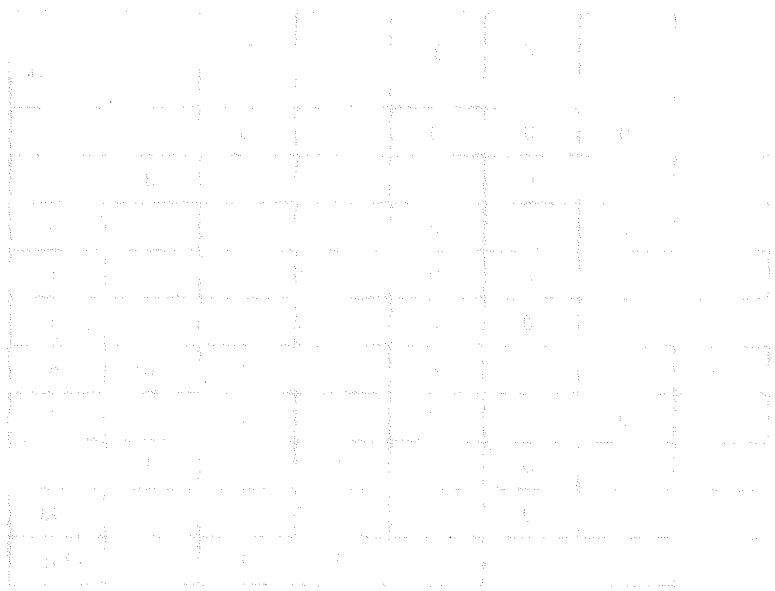
$$101_2 = 1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2$$
$$= 1 + 0 + 4$$
$$= 5_{10}$$

İkili sistemden onlu sisteme çevirme işlemlerini yapmak için en kolay yol, aşağıdaki gibi bir tablo hazırlamaktır.

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | Onlu Sistem |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 3 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 4 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 5 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 6 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 7 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 77 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 203 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 255 |

Örnekte de görüldüğü gibi, ikili sistemde 1 olan basamakların karşılığı olan ikinin kuvvetleri toplanarak onlu sistemde sonuç elde edilir.

NOT: Onlu sistemde olduğu gibi, ikili sistemde de bir sayıyı yazarken sol taraftaki sıfırları yazma zorunluluğu yoktur.



# *PROGRAMLAMAYA GİRİŞ :*

Bilgisayar, hazır duruma getirildiğinde (açıldığında), kullanıcıdan bir komut bekler. Verilen komut eğer bilgisayarın kullandığı programlama diline uygunsa, bilgisayar bu komutu gerektiği biçimde hemen uygular. Eğer komutun başında bir de sayı verilmişse, bilgisayar bu komutu uygulamaz; birçok komuttan oluşabilecek programın bir cümlesi olarak belleğinde saklar.

Program, ard arda dizilmiş bir komutlar zinciridir. Program içindeki bütün komutların belirli bir sırası vardır. Bu sıra, komutların başında verilen sayılara, yani satır numaralarına göre düzenlenir.

Bir satır numarası en küçük 1, en büyük 65535 arasında bir tamsayı olmalıdır. Bu sınırlar değişik bilgisayarlarda farklı olabilir.

**ÖRNEK:**

```
1 KOMUT-1
2 KOMUT-2
3 KOMUT-4
4 KOMUT-5
```

Bu programın satır numaraları, 1, 2, 3, 4 şeklinde sıralanmıştır. Bu şekilde yazılmasında hiçbir sakınca yoktur. Yalnız bu programda 1'inci ve 2'nci komutlardan sonra gelmesi gereken KOMUT-3 unutulmuştur. Satır numaralarının bu şekilde sıralanmış olması nedeniyle KOMUT-3'ün araya girilmesi olanaksızdır, çünkü 2 ve 3 arasında satır numarası olabilecek bir tamsayı yoktur.

**NOT:** Birçok BASIC yorumlayıcıda, satır numaralarını program mantığını bozmayacak şekilde yeniden düzenlemek için kullanılan bir komut vardır. Daha sonra işlenecek olan bu komutun yardımıyla, yukarıdaki gibi arka arkaya satır numaralarının araları açılabilir.

Aynı program yalnızca satır numaraları değiştirilerek aşağıdaki şekilde yazılmış olsaydı, KOMUT-3, uygun bir satır numarasıyla araya girilebilecekti:

```
10 KOMUT-1
20 KOMUT-2
30 KOMUT-4
40 KOMUT-5
```

Görüldüğü gibi, KOMUT-2 ve KOMUT-4 arasında satır numarası olabilecek tamsayılar vardır. KOMUT-3'ün satır numarası olarak 20 ve 30 arasında herhangi bir tamsayı kullanılabilir. Bunun için tüm programı baştan yazmak yerine yalnızca eksik satırı yazmak yeterlidir:

```
25 KOMUT-3
```

Programın son şekli aşağıdaki gibi olacaktır:

```
10 KOMUT-1
20 KOMUT-2
25 KOMUT-3
30 KOMUT-4
40 KOMUT-5
```

## Programlamada ortaya çıkabilecek hata türleri:

**1 - Yazım hataları:** Programlamayla ilgili özel terimlerin yanlış yazılması veya terimler arasında kullanılması gereken boşluk, virgül, nokta, noktalı virgül ve benzeri karakterlerin unutulması veya yanlış kullanılması sonucu meydana gelen hatalar.

**2 - Yapı hataları:** Grup olarak kullanılması gereken terimlerden bazılarının unutulması veya yanlış yerde kullanılmasıyla ortaya çıkan hatalar.

**3 - Mantık hataları:** Hiçbir yazım hatası olmadığı halde, program çalıştırıldıktan ve denendikten sonra ortaya çıkan, programlama mantığının yanlış kullanılmasıyla oluşan hatalar.

# DEĞİŞKENLER:

Bilindiği gibi, bilgisayarın belleği binlerce hücreden oluşmaktadır. Kullanılan bilgisayarın kapasitesine göre bu sayı yüzbinlere, hatta milyonlara varabilir. Değişkenler, bu hücrelerde istenilen bilgileri depolamak amacıyla kullanılır.

Çeşitli bilgileri bilgisayarın belleğinde saklayabilmek için bellekte bilgi depolanacak her birime (değişken) bir isim vermek gerekir.

**Değişkenleri isimlendirirken uyulması gereken kurallar:**

**a)** Değişken adlarında, harf ve sayılardan başka karakter kullanılamaz. Bununla birlikte, birçok bilgisayar değişken adı içinde nokta (.) kullanımını geçerli sayar. Ayrıca kullanılan harflerin İngiliz alfabesinde olması zorunludur.

**b)** Değişken adları mutlaka bir harfle başlamalıdır. İlk karakter olan harften sonra istenirse başka harfler veya sayılar kullanılabilir.

**c)** BASIC diline özgü terimler değişken olarak kullanılamazlar.

**d)** Değişken adında kullanılabilecek karakter sayısı kullanılan bilgisayara göre değişen bir sınır dahilinde olmalıdır. Bu sınır bazı bilgisayarlarda bir veya iki olurken, daha yüksek kapasiteli bilgisayarlarda 40 veya 50'ye kadar çıkabilir.

**NOT:** Bazı mikrobilgisayarlarda değişken seçimi sırasında, BASIC terimleriyle benzer değişkenlerin kullanılmamasına dikkat edilmelidir. Bu bilgisayarlarda, özellikle TOP, ORT gibi değişkenler (TO ve OR terimleriyle karıştırılabilecekleri için) kesinlikle kullanılmamalıdırlar.

**Değişken adlarına örnekler:**

SAYI
NO1
NO2
NO3
A13
BBC
TOPLAM
SAYI.TOPLAMI *(Yalnızca nokta kullanımına izin veren bilgisayarlarda)*
WWXQ *(İngilizce harfler geçerlidir.)*

**Yanlış değişken örnekleri:**

GÜN *(Ü harfi kullanılmıştır; GUN olarak düzeltilebilir.)*
1A *(İlk karakter harf değildir; A1 olarak düzeltilebilir.)*
A*B *(* işareti kullanılmıştır; AXB olarak düzeltilebilir.)*
NOT *(NOT kelimesi bir BASIC terimidir.)*
SAYI TOPLAMI *(Boşluk bırakılmıştır; SAYITOPLAMI olarak düzeltilebilir.)*

**Değişken türleri:**

1) Gerçel sayı değişkenleri

2) Tamsayı değişkenleri

3) Alfabetik değişkenler

**1) Gerçel sayı değişkenleri:**

Bu tür değişkenlerde saklanacak bilgiler sayısal olmalıdır. Bu sayılar tamsayı olabildikleri gibi kesirli sayılar da olabilir.

Örnekler: A, S2, ABC

Gerçel sayı değişkenlerinde şu sayılar saklanabilir: 6, 3.45, -12, vb.

**2) Tamsayı değişkenleri:**

Adından da anlaşılabileceği gibi, tamsayı değişkenlerinde yalnızca tamsayılar saklanabilir. Bir değişkenin tamsayı değişkeni olduğunu belirtmek için değişken adından sonra % işareti kullanılır.

Örnekler: A%, S2%, ABC%

Eğer bir tamsayı değişkenine kesirli bir sayı yerleştirilmek istenirse, bu sayı en yakın tamsayıya yuvarlanarak belleğe kaydedilir.

Tamsayı değişkenleri, bilgisayarın belleğinde gerçel sayı değişkenlerine oranla daha az yer kaplarlar. Eğer bir program içinde kullanılacak sayısal değerler sadece tamsayı değerleri olacaksa, bu değerler için tamsayı değişkenleri kullanmak programın bellekte daha az yer kaplamasını ve daha süratli çalışmasını sağlayacaktır.

Tamsayı değişkenlerinde şu sayılar saklanabilir: 5, 0, -3, -135, vb.

**3) Alfabetik değişkenler:**

Alfabetik (karaktersel) değişkenlerde karakter bilgileri saklanır. Alfasayısal olarak isimlendirilen bu tür bilgiler, harfler, sayılar, özel işaretler ve kullanılan bilgisayara özgü çeşitli şekilleri içermektedir. (Örnek: A, B, 5, 7, *, %, -,...)

Bir harf, kelime veya cümle alfabetik bir değişkende depolana-

rak daha sonra kullanılabilir. Bir değişkenin alfabetik değişken olduğunu belirtmek için değişken adından sonra $ işareti kullanılır.

**Örnekler: A$, S2$, ABC$**

Herhangi bir alfabetik değişkene bilgi yerleştirmek istendiğinde, bu bilgi tırnak işaretleri (" ") arasında yazılmalıdır.

Alfabetik değişkenlerde şu bilgiler saklanabilir: "AHMET", "TENCERE", "45.34-23", vb.

**NOT:** Alfabetik değişkenlerde saklanması istenilen bilgilerin içinde bütün karakterler kullanılabilir. Bu karakterler harf, sayı, aritmetik operatörler veya diğer özel karakterler olabilir.

Değişkenlerin içinde bulunan bilgiler, bilgisayar kapatılmadığı veya herhangi bir komut yardımıyla silinmedikleri sürece hiç değişmeden saklanırlar. Bu bilgiler daha sonra ekrana veya printer'e yazılmak, ya da herhangi bir işlem yapabilmek için kullanılabilir. Sayısal değişkenlerden, istenilen aritmetik işlemlerin yapılabilmesi için yararlanılabilir; alfabetik değişkenler ise yalnızca toplama işlemlerinde yer alabilirler. (Bu toplama sayısal anlamda bir toplama değil, karakter dizilerinin arka arkaya eklenmesidir.)

# KOMUTLAR:

Bilgisayarın yapması istenilen bütün işlemler, bilgisayara komutlar yoluyla ulaştırılır. Örneğin sayıları toplamak, ekrana bir isim yazmak, grafik çizmek, müzik çalmak gibi bütün eylemler ancak gerekli komutların doğru yerlerde ve kurallara uygun olarak kullanılmasıyla yapılabilir. BASIC dilinde en çok kullanılan komutlar şunlardır:

**LET:** Bellekteki belirli bir hücrede istenilen bir bilgiyi depolamak için kullanılır. Bunu yaparken, daha sonra kullanabilmek amacıyla, o hücreye bir de isim verilir.

Genel yazılım şekli:

**LET <değişken> = <Atama yapılacak değer, aritmetik ifade veya karakter dizisi>**

NOT: Konular anlatılırken, yazılması zorunlu olmayan bölümler [], mutlaka yazılması gereken bölümler ise <> işaretleri arasında verilmiştir. Bu işaretler, program uygulamalarında açıklamalarda olduğu gibi kullanılmazlar.

ÖRNEKLER:

1) Gerçel sayı değişkenleri:

```
LET A = 54.12
LET I  = 48
LET G = 34 + 85.2
LET F = H * 37 + 49
```

2) Tamsayı değişkenleri:

```
LET A%  = 54
LET N% = P%
LET P%  = -30
LET UZ%  = 57.28
```
(Bu değer 57 olarak saklanacaktır.)

3) Alfabetik değişkenler:

```
LET A$  = "PROGRAM"
LET T$  = "BASIC"
LET K$ =  R$
LET M$ =  B$ + Z$
```

**NOT:** LET komutunun kullanılması birçok bilgisayarda zorunlu değildir. Bu tür bilgisayarlarda yukarıdaki örnekler LET kullanmadan da yazılabilir.

Örnek:

```
A = 500
F% = 6
S$ = "BASIC"
```

**RUN:**

Satır numaralarıyla birlikte yazılmış olan komutları (programı) çalıştırmak için kullanılır. RUN komutu verildiğinde, satır numarası en küçük olan komut ilk olarak uygulanır; daha sonra, satır numaralarının sırasına göre bütün komutlar uygulanır. İstenirse, RUN komutuyla birlikte bir satır numarası da kullanılabilir. Bu durumda, program verilen satır numarasından başlayarak çalışacaktır.

**END:**

Program içine yazıldığında, programı durdurmak için kullanılan komuttur. Özellikle sonsuz programları belirli durumlarda durdurabilmek için kullanılır.

**PRINT:**

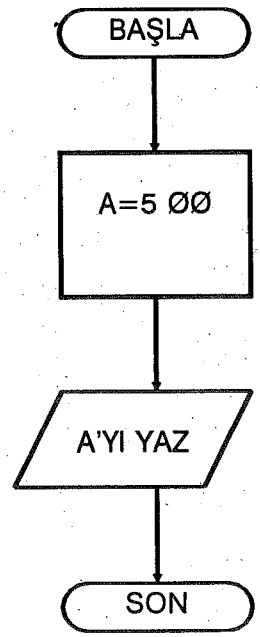
Harfleri, sayıları ve sembolleri herhangi bir çıkış ünitesine aktarmak (yazmak) için kullanılan komuttur.

Genel yazılım şekli:

**PRINT [değişken, sabit, aritmetik ifade veya karakter dizisi.]**

**ÖRNEKLER:**

1 - PRINT komutundan sonra bir değişken yazılması:




```
LET A = 500
PRINT A
```

2 - PRINT komutundan sonra bir sabit yazılması:

```
PRINT 100
```

3 - PRINT komutundan sonra bir aritmetik ifade yazılması:

```
LET A = 100
LET B = 200
PRINT A * B
    veya
PRINT 100 * 200
```

4 - PRINT komutundan sonra bir karakter dizisi yazılması:

```
PRINT "Bilgisayar dünyasına hoş geldiniz."
```

Bu örnekte de görüldüğü gibi, karakter dizisi yazılmak istendiğinde tırnak işaretleri kullanılmalıdır.

PRINT komutundan sonra birden çok ifade kullanılıyorsa, bu ifadelerin her biri arasında bir ayraç kullanmak zorunludur. Bu ayraç amaca göre virgül (,) veya noktalı virgül (;) olabilir. Eğer ifadelerin yan yana yazılması isteniyorsa noktalı virgül, aralarında belirli boşluklar bırakılarak daha önceden saptanmış sütunlara yazılması isteniyorsa virgül kullanılmalıdır.

**ÖRNEKLER:**

```
1 - LET A = 5
    LET B = 3
    PRINT A,B
     5                3

2- PRINT "BASIC";"programlama"
   BASICprogramlama
```

Bilgisayar, bir PRINT cümlesini uyguladıktan sonra normal olarak yazım işlemine bir alttaki satırın başından başlar. Bu durum engellenmek istendiğinde, PRINT cümlesi virgül veya noktalı virgülle bitirilebilir. Sonuç olarak, bilgisayar bir sonraki PRINT komutunun içeriğini, ekran üzerinde en son kaldığı pozisyondan devam ederek yazar.

**ÖRNEKLER:**

```
1 - 10 PRINT "Bilgisayar";
    20 PRINT " eğitimi"

    RUN
    Bilgisayar eğitimi
```

**NOT:** İki kelimenin arasındaki boşluk, ikinci PRINT komutundan sonraki tırnak işaretleri arasındaki boşluk karakteriyle sağlanmıştır. Aksi takdirde, iki kelime bitişik olarak yazılacaktı.

```
2 - 10 PRINT "Bilgisayar",
    20 PRINT "eğitimi"

    RUN
    Bilgisayar  eğitimi

3 - 10 PRINT "Bilgisayar"
    20 PRINT
    30 PRINT "Eğitimi"

    RUN
    Bilgisayar

    Eğitimi
```

**NOT:** Satırlar arasında boşluk bırakabilmek için, PRINT komutu bu örnekte de olduğu gibi, tek başına kullanılabilir.

4 - Daha önce LET komutuyla verilmiş örneklerin PRINT komutuyla birlikte kullanılması:

**A) Gerçel sayı değişkenleri:**

```
10 LET A  = 54.12
20 LET G  = 119.2
30 LET I  = 48
```

```
40 LET F = G * 2 + 49
50 PRINT A,I,G,F

RUN          54.12          48          119.2          287.4
```

B) Tamsayı değişkenleri:

```
10 LET A%  = 54.3
20 LET N%  = A% * 3
30 LET P%  = -30
40 LET UZ% = 57.28
50 PRINT A%,N%,P%,UZ%

RUN          54          162          -30          57
```

C) Alfabetik değişkenler:

```
10 LET A$ = "PROGRAM"
20 LET T$ = "BASIC "
30 LET K$ = "LAMA"
40 LET M$ = T$ + A$ + K$
50 PRINT M$

RUN
BASIC PROGRAMLAMA
```

PRINT komutu, yukarıdaki kullanımların dışında, aşağıdaki fonksiyonlarla da kullanılabilir:

1) PRINT SPC

Genel kullanım:

**PRINT SPC (tamsayı); [yazılacak ifadeler]**

PRINT komutundan sonraki ifadelerin başında, arasında veya sonunda tamsayı ile belirtilen adette boşluk bırakmak için kullanılır.

**ÖRNEK:**

```
PRINT SPC(7);"Bir yılda";SPC(4);"12 ay vardır."
   Bir yılda              12 ay vardır.
```

**2) PRINT TAB**

Genel kullanım:

**PIRNT TAB (tamsayı) ; [yazılacak ifadeler]**

PRINT komutundan sonraki ifadeleri, ekranın sol kenarından itibaren tamsayı ile belirlenen sütundan başlayarak yazmak için kullanılır.

**ÖRNEK:**

```
PRINT TAB(9);"Numara";TAB(20);"İsim-Soyadı"
    Numara        İsim-Soyadı
```

**3) PRINT USING**

Genel kullanım:

**PRINT USING <kalıp>; <yazılacak ifadeler>**

PRINT komutundan sonraki ifadeleri belli bir kalıba uygun olarak yazar.

**ÖRNEKLER:**

```
1)  LET A = 123456.789
    PRINT USING "###,###,###.##";A
      123,456.79

2)  LET B = 8372617.77
    PRINT USING "#,###,###";B
      8,372,618
```

**NOT:** PRINT USING ile ilgili sayısal ve karaktersel birçok kalıp vardır. diğer kalıplar daha sonra, ileri BASIC konularında işlenecektir.

**PRINT komutuyla ilgili örnekler:**

1) Her kelime için bir değişken kullanarak, 'PROGRAM YAZMAK, DÜŞÜNCEYİ GELİŞTİREN BİR UĞRAŞTIR.' cümlesini yazan bir program:

```
10 A$ = "PROGRAM "
20 B$ = "YAZMAK, "
30 C$ = "DÜŞÜNCEYİ "
40 D$ = "GELİŞTİREN "
50 E$ = "BİR "
60 F$ = "UĞRAŞTIR."
70 PRINT A$ ; B$ ; C$ ; D$ ; E$ ; F$

RUN
PROGRAM YAZMAK, DÜŞÜNCEYİ GELİŞTİREN BİR
UĞRAŞTIR.
```

Bu programda, her kelime bir değişkene atanırken, kelimeler arasındaki boşluklar da tırnak işaretleri içine alınmıştır. Aksi takdirde, kelimeler birbirine bitişik olarak yazılacaktır.

2) Cümlenin elemanlarını değişkenlere atamak yoluyla, 'Son 5 yılda şehrimizin nüfusu % 35.8 oranında artmıştır.' yazan bir program:

```
10 A$ = "Son"
20 B% = 5
30 C$ = "yılda şehrimizin nüfusu %"
40 D = 35.8
50 E$ = "oranında artmıştır."
60 PRINT A$ ; B% ; C$ ; D ; E$
```

Bu cümleyi yazabilmek için 3 değişken türü de kullanılmıştır. 20 numaralı satırdaki B% tamsayı değişkeni, B gerçel sayı değişkeni olarak de kullanılabilirdi. Fakat sayı tamsayı olduğu için, bellekte daha az yer kaplayan tamsayı değişkeni tercih edilmiştir. Cümle içinde bulunan '%' işareti, alfabetik bir değişken içinde yer almalıdır. Bu işaretin bu örnekte tamsayı değişkeniyle ilgisi yoktur.

3) Yalnızca sayısal bilgileri değişkenlere atayarak yukarıdaki cümleyi yazan bir program:

```
10 A% = 5
20 B = 35.8
30 PRINT "Son";A%;"yılda şehrimizin nüfusu %";B;
"oranında artmıştır."

RUN
Son 5 yılda şehrimizin nüfusu % 35.8 oranında
artmıştır.
```

Bu programdaki PRINT satırında değişken yerine kelimelerin kendileri kullanılmıştır. Bu durumda bu kelimeler tırnak işaretleri arasında yazılmıştır. Değişkenler yoluyla yazılan bilgilerin tırnak işareti içinde bulunmaması gerektiğine dikkat edilmelidir.

**SORULAR**

1) Aşağıdaki listeyi, değerleri uygun değişkenlere atayarak boş bir ekrana yazdırınız.

X MARKET SATIŞLARI

| MAL ADI | 1987 | 1988 |
|---|---|---|
| YAĞ | 2600 | 2500 |
| BAL | 2600 | 3100 |
| PEYNİR | 3000 | 4000 |

2) Aşağıdaki değerleri uygun değişkenlere atayarak, her birinin arasında 3 boşluk kalacak şekilde ekrana yazdırınız.

(Sayısal değerler de tamsayıya çevrilmelidir.)

2.80, ALİ, 4.30, 5.92

## E NOTASYONU:

Bilgisayar, çok büyük ya da çok küçük sayıları ifade edebilmek için özel bir yöntem kullanır.

**ÖRNEK:** 55000 ve 25000 sayılarının çarpımını hesaplayabilmek için şu komut yazılmalıdır:

**PRINT 55000 * 25000**

Çıkan sonuç şu şekilde olacaktır:

**1.375E+09**

Bu sonuç, $1.375 * 10^9$ sayısını ifade etmektedir.

Bilgisayarın böyle bir yönteme başvurmasının nedeni, bu işlemin sonucunun (1375000000) çok basamaklı olmasıdır.

Aynı şekilde, 5 / 150000 işleminin sonucu da 3.33333E-05 olacaktır.

Sonuç, $3.33333 * 10^{-5}$ ($3.33333 / 10^5$) sayısına eşittir.

Bu sayının E notasyonu kullanılmadan yazılışı 0.000033333 şeklinde olacaktı.

E notasyonu kullanıldığında, E harfinden sonra gelen (+) işareti, noktanın sağ tarafa, (-) işareti ise sol tarafa kaydırılacağını gösterir. Daha sonraki sayı ise noktanın kaç basamak kayacağını belirtir.

# İŞLETİM KOMUTLARI:

En çok kullanılan ve daha önce anlatılmış olan RUN komutunun dışında diğer bazı işletim komutları da şunlardır:

**CLS**
**LIST**
**EDIT**
**NEW**
**RENUM**
**AUTO**
**DELETE**

Bütün komutlarda da olduğu gibi, bu komutları da bilgisayara iletmek için komutu yazdıktan sonra RETURN veya ENTER tuşuna basmak gerekir.

**CLS:**

Ekranı silmek için kullanılan komuttur. Komut uygulandıktan sonra kursör ekranın sol üst köşesine taşınır.

**LIST:**

Bellekte mevcut olan programı listelemek için kullanılan komuttur. Tek başına kullanıldığında, bütün programın listesini verir. Eğer

yalnızca belirli satırlar listelenmek isteniyorsa, komutla birlikte satır numaraları verilir.

Genel kullanım:

**LIST [satır numarası-1] - [satır numarası-2]**

**ÖRNEK:**

```
LIST              (Bütün satırlar listelenir.)
10 CLS
20 LET A = 15
30 LET B = 20
40 LET C = A * B
50 PRINT A,B,C
60 END

LIST 30-  (30 numaralı ve daha sonraki satırlar listelenir.)
30 LET B = 20
40 LET C = A * B
50 PRINT A,B,C
60 END

LIST -40  (40 numaralı ve daha önceki satırlar listelenir.)
10 CLS
20 LET A = 15
30 LET B = 20
40 LET C = A * B

LIST 20-50  (20'den 50'ye kadar bütün satırlar listelenir.)
20 LET A = 15
30 LET B = 20
40 LET C = A * B
50 PRINT A,B,C
```

**EDIT:**

Program satırlarından herhangi birini değiştirmek amacıyla kullanılan komuttur.

Program satırlarından birini değiştirmek için birinci yol satırı yeni şekliyle baştan yazmak, ikinci yol ise EDIT komutunu kullanmaktır. Özellikle uzun satırların düzeltilmesi için birinci yol çok kullanışsızdır.

İstenilen bir satır üzerinde değişiklik yapmak için o satırın numarasını EDIT komutundan sonra yazmak gereklidir. Bundan sonra, ok tuşlarını kullanarak kursör düzeltilecek bölüme taşınır ve DEL ve CLR tuşlarının yardımıyla düzeltme yapılır.

Genel kullanım:

**EDIT <satır numarası>**

**NEW:**

Bellekteki programı tamamen silmek için kullanılan komuttur. Bu komutla birlikte hiçbir parametre kullanılmaz.

**RENUM:**

Programdaki satırların numaralarını yediden düzenlemek için kullanılan komuttur.

Genel kullanım:

**RENUM [yeni satır numarası] , [eski satır numarası] , [artış]**

Eğer parametrelerin hiç birisi kullanılmazsa, satır numaraları 10'dan başlayarak 10'ar 10'ar artacak şekilde dizilir.

**AUTO:**

Program yazarken, satır numaralarının otomatik olarak bilgisayar tarafından yazılması için kullanılan komuttur.

Genel kullanım:

**AUTO [başlangıç satır numarası] , [artış]**

Eğer parametreler kullanılmazsa, satır numaraları 10'dan başlayarak 10'ar 10'ar artar.

**DELETE:**

Programın tümünü veya bir bölümünü silmek için kullanılan komuttur.

Genel kullanım:

**DELETE [satır numarası–1] — [satır numarası–2]**

Bellekteki programın, satır numaralarıyla belirlenen bölümünü siler. Eğer ikinci satır numarası yazılmazsa yalnızca tek satır silinir; iki satır numarası da yazılmazsa, program tamamen silinir.

**ÖRNEK:**

```
LIST
 10 CLS
 20 LET A = 15
 30 LET B = 20
 40 LET C = A * B
 50 PRINT A
```

```
60 PRINT B
70 PRINT C
80 END

DELETE 20

LIST
10 CLS
30 LET B = 20
40 LET C = A * B
50 PRINT A
60 PRINT B
70 PRINT C
80 END

DELETE -30

LIST
40 LET C = A * B
50 PRINT A
60 PRINT B
70 PRINT C
80 END

DELETE 70-

LIST
40 LET C = A * B
50 PRINT A
60 PRINT B

DELETE 50-60

LIST
40 LET C = A * B

DELETE

LIST
```
(Program tamamen silinmiştir.)

# ARİTMETİK İŞLEMLER:

Taşınabilir hesap makineleri gibi, bilgisayar da, istenilen aritmetik işlemi yapma yeteneğine sahiptir. Bunlar dört temel işlem (toplama, çıkarma, çarpma, bölme) olabildiği gibi, kök alma, üs alma veya sinüs, kosinüs, tanjant gibi trigonometrik işlemler de olabilir. Bu bölümde, basit olarak dört temel işlemin BASIC dilindeki yazım kuralları ve kullanımları incelenecektir.

Toplama ve çıkarma işlemleri için genel olarak kullanılan işaretler BASIC'te de aynıdır. Toplama için artı işareti (+), çıkarma için ise eksi (-) işareti kullanılır.

**ÖRNEK:** 2 + 3 veya 5 - 4

Çarpma ve bölme için BASIC dilindeki işaretler normal olarak kullanılanlardan farklıdır. Çarpma işareti olarak (x) yerine (*), bölme işareti olarak (/) kullanmak zorunludur.

ÖRNEK: 8'i 4'e bölmek için **8 / 4**
3 x 5 yerine **3 * 5** kullanılmalıdır.

Bilgisayarda hesap yapmak, normal bir hesap makinesinde hesap yapmaktan farklıdır. Bunun için, bilgisayarın anlayacağı bir komut vermek gereklidir. Örneğin bir işlemin sonucunu ekranda görmek için **PRINT** komutunu kullanmak gerekirken, bu sonucu bellekte saklamak için **LET** kullanılmalıdır.

**ÖRNEKLER:**

1) 4 ve 7 sayılarının toplamını ekranda görmek için yapılması gereken işlem şudur:

```
PRINT 4 + 7
 11
```

Bu örnekteki gibi, iki sayının toplamını ekranda görebilmek için, işlemin başında bir PRINT komutu yazılmalıdır. NOT: PRINT komutu kullanmadan işlem yalnızca '4 + 7' olarak yazılırsa, satır başındaki 4 sayısı, bir satır numarası olarak kabul edilir ve satır, geçersiz bir program satırı olur.

2) Aynı sayıların toplamının ekrana yazılması değil de, bir değişkende saklanarak daha sonra kullanılması isteniyorsa, şu satır yazılabilir:

```
LET A = 4 + 7
```

Bu durumda sonuç (11) ekrana yazılmaz; bellekte A değişkeni olarak saklanır. Daha sonra bu değer kullanılmak istendiğinde, bilgisayara şu komut verilebilir:

```
PRINT A
 11
```

Bu şekil kullanımda, A değişkeninde saklanan değer, istenildiği sürece bilgisayarın belleğinde kalacaktır.

## ARİTMETİK İŞLEMLERİN ÖNCELİK SIRALARI:

Sonucun bulunması için birden fazla işlemin birlikte yapılması gerekiyorsa, bu işlemler yazıldıkları sırada değil, belirli bir öncelik sırasına göre yapılırlar.

**İşlemlerin öncelik sırası şu şekildedir:**

1 - Üs alma ..... ^
2 - Çarpma ve bölme..... * /
3 - Toplama ve çıkarma..... + -

NOT: Üs alma işareti bazı bilgisayarlarda ** olarak kullanılmaktadır.

**ÖRNEKLER:**

**1) PRINT 3 + 8 * 4**
**35** **(8 * 4 = 32 ; 3 + 32 = 35)**

Görüldüğü gibi ilk işlemin toplama olmasına karşın, öncelik sırası çarpma işleminin olduğu için **(8 * 4 = 32)** işlemi daha önce yapılmıştır.

**NOT:** Öncelik sırası aynı olan işlemler soldan sağa doğru yapılır.

**2) PRINT 5 / 2 * 3**
**7.5** **(5 / 2 = 2.5 ; 2.5 * 3 = 7.5)**

NOT: Yapılması istenen işlemde farklı öncelik sıralı işlemler varsa ve önceliği daha sonra gelen bir işleme öncelik kazandırılması isteniyorsa, o işlem parantez içinde belirtilmelidir.

**3) PRINT (3 + 8) * 4**
**44** **(3 + 8 = 11 ; 11 * 4 = 44)**

Parantez içine alınan toplama işlemi, öncelik kazanacak ve çarpma işleminden daha önce yapılacaktır.

4) $2 ( A + B ) + 3 C^2$ işleminin BASIC'te kodlanması:
**2 * ( A + B ) + 3 * C ^ 2**

5) $( ( A^2 - B ) + 3 - ( B - 4 A C ) )^K$ işleminin BASIC'te kodlanması:
**( ( A ^ 2 - B ) + 3 - ( B - 4 * A * C ) ) ^ K**

Görüldüğü gibi, BASIC'te üs alma işlemleri, ^ işaretinin kullanılmasıyla gerçekleştirilmektedir. Ayrıca iç içe parantez açılırken matematikte olduğu gibi köşeli parantezler kullanılamaz; bunun için de örnekteki gibi normal parantez işaretlerinin arka arkaya kullanılmasında sakınca yoktur. (Açılan parantez sayısıyla kapanan parantez sayısının her zaman eşit olması gerektiğine dikkat edilmelidir.)

# KOMUTLAR:

Daha önce işlenilenlerin dışında bazı BASIC komutları da aşağıda verilmiştir:

**REM:**

Program içine açıklayıcı, hatırlatıcı mesajların yazılabilmesi için kullanılır. Bilgisayar, bir program satırında bu komuttan sonraki hiçbir bilgiyi dikkate almaz; işleme koymaz. Dolayısıyla REM komutunun programın çalışması üzerinde hiçbir etkisi olmadığından, bu komutun kullanılması zorunlu değildir.

Genel kullanım:

**REM [açıklama]**

NOT: REM komutu yerine kısaca (') işareti de kullanılabilir.

ÖRNEK:

```
10 REM ** Bu program faiz hesabı yapar. **
20 P = 100000
```

```
30 F = 45
40 Y = 4
50 TF = P * F * Y / 100
60 PRINT "Toplam faiz = ";TF
70 END
```

Bu örnekteki REM satırı, programın sonucunu etkilemez; yalnızca program listesini okuyan kişi için bir yardımcı mesaj görevini görür.

**INPUT:**

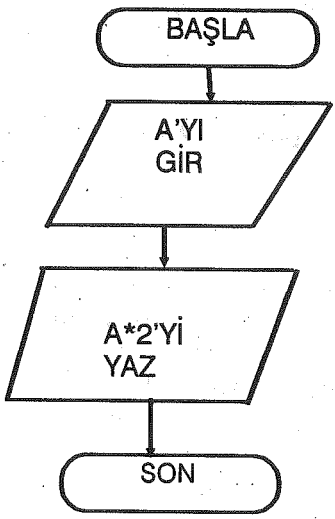
Program için gerekli olan bilgilerin program çalışırken, bilgisayarı kullanan kişi tarafından klavye yoluyla verilmesi için kullanılır. Bir INPUT komutuyla en az bir olmak üzere, istenildiği kadar bilgi bilgisayara iletilebilir.

Genel kullanım:

**INPUT ["açıklama";] <değişken listesi>**

(Buradaki değişken listesi, bir değişkeni veya aralarında virgül kullanılarak yazılmış bir dizi değişkeni ifade etmektedir. Örnek: INPUT A$,B,T$). Program çalıştırıldığında bilgisayar, INPUT satırına geldiği zaman, INPUT komutundan sonra kullanılan değişkenlerin değerleri klavyeden girilene kadar bekler. Programın akışı, ancak bu değişkenlere uygun sırada, sayıda ve nitelikte değerler verilerek sağlanabilir. Aksi takdirde bilgisayar bir hata mesajı (Redo from start) verir ve bilgi girişi başa döner.

**ÖRNEKLER:**




1)

```
10 INPUT A
20 PRINT A * 2
30 END
```

Bu örnekte program çalıştırıldığında bilgisayar ekrana bir soru işareti (?) çıkaracak ve kullanıcı A değişkeni için gerekli bir sayısal değer verene kadar bekleyecektir. Daha sonra klavyeden girilen değer, A değişkenine atanacak ve program akışı 20 numaralı satırdan devam edecektir.

Program çalıştırıldığında:

```
RUN
? Hasan
?Redo from start
? 4
 8
```

Görüldüğü gibi, girilen ilk bilgi sayısal olmadığından bilgisayar 'Redo from start' (baştan yap) mesajını çıkartmış ve doğru bilginin girilmesini beklemiştir. Sayısal bir değer olan 4 girildiğinde, programa uygun olarak iki ile çarpımı bilgisayar tarafından yazılmıştır.

2) **10 INPUT "İki kelime giriniz ";A$,B$**
**20 PRINT A$,B$**
**30 END**

Bu örnekte, INPUT komutuyla kullanılan değişkenlerden önce bir de açıklama yazılmıştır. Bu açıklama, programın çalışmasını ve mantığını etkilemez; bilgi girecek kişiye, gireceği bilginin türünü ve adedini hatırlatmak amacıyla kullanılmıştır. Dikkat edilirse, değişkenlerin sayısı birden fazla olduğu için aralarına virgül koyulmuştur.

**Program çalıştırıldığında:**
**RUN**
**İki kelime giriniz ? HASAN,AYLA**
**HASAN AYLA**

Bu program çalıştırıldığında, bilgisayar kullanıcıdan iki kelime ister. Eğer daha az veya daha çok kelime verilirse, bir önceki örnekte olduğu gibi, bilgi girişi başa döner. Bilgi girişi doğru yapıldığında ise kelimeler aralarında biraz boşluk bırakılarak ekrana yazılır. (PRINT komutuyla virgül kullanılmıştır.)

NOT: Bazı BASIC yorumlayıcılarında, INPUT komutuyla birlikte açıklama verebilme özelliği yoktur. Bu tür yorumlayıcılarla çalışırken, açıklama şu şekilde verilebilir:

**PRINT "İki kelime giriniz ";**
**INPUT A$,B$**

3) Net fiyatı klavyeden girilen bir malın KDV'sini ve toplam fiyatını hesaplayıp ekrana yazan bir program:

```
10 INPUT "MALIN NET FİYATI NEDİR ";NF
20 KDV = NF / 100 * 12
30 TF = NF + KDV
40 PRINT "KATMA DEĞER VERGİSİ = ";KDV
50 PRINT "TOPLAM FİYAT = ";TF
```

4) Klavyeden girilen dolar kuruna göre, yine klavyeden girilen Türk lirası miktarının dolar olarak değerini hesaplayıp ekrana yazan bir program:

```
10 INPUT "DOLAR KURUNU GİRİNİZ ";DK
20 INPUT "TÜRK LİRASI MİKTARINI GİRİNİZ ";T
30 D = T / DK
40 PRINT T;" LİRA ";D;" DOLAR DEĞERİNDEDİR."
```

5) Klavyeden girilen iki kişiye ait isim ve yaşları boş bir ekrana 'İSİM' ve 'YAŞ' başlıkları altında listeleyen bir program:

```
10 INPUT "BİRİNCİ KİŞİNİN ADINI VE YAŞINI VERİNİZ
";A1$,Y1
20 INPUT "İKİNCİ KİŞİNİN ADINI VE YAŞINI VERİNİZ
";A2$,Y2
30 CLS
40 PRINT "İSİM","YAŞ"
50 PRINT "-------","------"
60 PRINT A1$,Y1
70 PRINT A2$,Y2
```

**SORULAR:**

1) KDV'nin % 10 oranında hesaplandığı kabul edilirse, toplam fiyatı klavyeden girilen bir malın KDV'siz fiyatını ve KDV'sini hesaplayarak ekrana yazan bir program yapınız.

2) 750000 liranın %72'den 3 yıllık faizini hesaplayarak ekrana yazan bir program yapınız.

3) Değerlerin ilk ikisini atayarak, son ikisini klavyeden girerek aşağıdaki listeyi boş bir ekrana yazdırınız.

**OKUL İHTİYAÇ LİSTESİ:**

| | | |
|---|---|---|
| KALEM | : | 200 TL |
| SİLGİ | : | 100 TL |
| DEFTER | : | 500 TL |
| KİTAP | : | 1000 TL |

4) Klavyeden girilen üç sayıdan ilk ikisinin toplamının üçüncüsüyle çarpımını yazan bir program yapınız.

5) Klavyeden girilen 5 sayıdan ilk üçünü yan yana, son ikisini alt alta yazan bir program yapınız.

**GOTO komutu:**

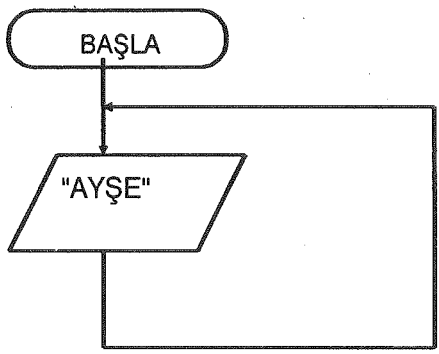
Programı, normal akışının dışında istenilen bir satırdan devam ettirmek için kullanılır.

Genel kullanım:

**GOTO <satır numarası>**

Bu komutla birlikte kullanılan satır numarası, programın gönderilmesi istenilen satırın numarasıdır. Eğer belirtilen numaralı satır programda yoksa, bilgisayar bir hata mesajı (Line does not exist) verir ve program durur.

**ÖRNEK:** 'Ayşe' kelimesini ekrana sonsuz kez yazdıran bir program:




```
10 PRINT "Ayşe"
20 GOTO 10
```

Bu program çalıştırıldığında bilgisayar ilk olarak PRINT komutunu uygulayacak ve 'Ayşe' kelimesini ekrana yazacaktır. Program daha sonra normal akışına göre 20 numaralı satıra gelecek ve GOTO komutunu uygulayacaktır. GOTO komutuyla birlikte kullanılan satır numarası 10 olduğu için program tekrar 10 numaralı satıra geri dönecek ve kelimeyi bir kez daha yazacaktır. Bu şekilde sonsuz bir döngü oluşacak ve program çalışmaya başladıktan sonra hiç dur-

maksızın 'Ayşe' kelimesi ekrana çıkacaktır. Böyle bir programı durdurabilmek için klavyedeki bazı özel tuşlardan yararlanmak gerekecektir.

**ÖRNEK:** Klavyeden girilen kenar uzunluğuna göre bir karenin alanını ve çevresini hesaplayıp ekrana yazan ve hiç durmayan bir program:

```
10 INPUT "Karenin kenar uzunluğunu veriniz ";A
20 ALAN = A * A
30 CEVRE = A * 4
40 PRINT "Karenin alanı =";ALAN
50 PRINT "Karenin çevresi =";CEVRE
60 GOTO 10
```

Bu program, klavyeden girilen kenar uzunluğuna göre karenin alan ve çevresini hesaplayarak ALAN ve CEVRE değişkenlerinde saklayacak ve daha sonra açıklayıcı mesajların da yardımıyla bu değerleri ekrana yazacaktır. Bütün bunları yaptıktan sonra program tekrar INPUT satırına geri dönecek ve işlem sonsuza dek (ya da program özel tuşlar kullanılarak durdurulana kadar) tekrarlanacaktır. Eğer bu programdaki GOTO satırı silinecek olursa, işlem yalnızca bir kez gerçekleşecek ve program duracaktır.

GOTO komutundan sonraki satır numarası 40 olarak değiştirilirse:

(Bunu yapmak için satır baştan yazılabilir.)

```
60 GOTO 40
```

Bu durumda program çalıştırıldığında şu sonuç alınacaktır:

```
RUN
Karenin kenar uzunluğunu veriniz ? 5
Karenin alanı = 25
Karenin çevresi = 20
Karenin alanı = 25
Karenin çevresi = 20
Karenin alanı = 25
```

```
Karenin çevresi = 20
Karenin alanı = 25
.
.
.
```

bu şekilde devam edecek...

Program hiç durmadan aynı sonucu yazacak, bu arada klavyeden bilgi okumayacaktır, çünkü ilk geçişten sonra bilgisayar 10 numaralı satıra hiçbir zaman uğramayacaktır.

GOTO satırı yeniden değiştirilip,

```
(60 GOTO 15)
```

Program tekrar çalıştırılırsa:

```
RUN
Karenin kenar uzunluğunu veriniz ? 5
Karenin alanı = 25
Karenin çevresi = 20
Line does not exist in 60
```

Programda 15 numaralı satır olmadığı için bilgisayar 60 numaralı satıra geldiğinde GOTO 15 komutunu uygulayamayacak ve 'Satır yok' anlamında bir hata mesajı (Line does not exist) çıkaracaktır. Burada dikkat edilmesi gereken bir nokta, programın hatalı olduğu halde çalışmaya başlaması, ancak hatayla karşılaşıldığı zaman durmasıdır.

**ÖRNEK:** Klavyeden girilen bir malın adı, fiyatı ve adedine göre, tutarı hesaplayarak sonucu anlamlı bir cümle içinde yazan bir program:

```
10 INPUT "Malın adı ";A$
20 INPUT "Fiyatı ";F
30 INPUT "Adedi ";A
40 TUTAR = F * A
50 PRINT"Tanesi";F;"liradan";A;"tane";A$;"için";
TUTAR;"lira ödenmelidir."
```

**SORULAR:**

1) Klavyeden girilen kelimeyi boş bir ekrana yazma işlemini sonsuza kadar tekrarlayan bir program yapınız.

2) Klavyeden girilen sayının karesini yazan ve başa dönen sonsuz bir program yapınız.

**READ komutu:**

Bilindiği gibi, INPUT komutu kendisinden sonra gelen değişkenlere uygun bilgileri klavyeden okur. Özellikle sabit bilgiler ile kullanılan programlarda INPUT komutu oldukça kullanışsızdır. LET komutuyla eşitleme yapmak da çok sayıda değer atama için zahmetlidir. Örneğin faiz hesabı yapan bir program her çalıştırıldığında faiz oranını klavyeden girmek, veya 15 - 20 değer atama işlemi için ayrı ayrı komut yazmak zaman kaybına neden olduğu gibi, programlama açısından da karışıklığa yol açabileceğinden sakıncalıdır. Bellekte bilgi saklamak için INPUT ve LET komutlarının dışında bir de READ komutu kullanılabilir.

Genel kullanım:

**READ <değişken listesi>**
**DATA <veri listesi>**

**READ komutu, birkaç fark dışında INPUT komutuna çok benzemektedir:**

**1)** INPUT, bilgiyi klavyeden okurken, READ, program içinde bir DATA satırına yazılmış olan bilgileri okur.

**2)** INPUT (gerekli bilgi girilene kadar) programın çalışmasını durdurur; READ ise LET komutu gibi çok kısa bir zamanda uygulanır; programda bir beklemeye neden olmaz.

**3)** INPUT komutundan sonra istenirse, bilginin neyle ilgili olduğunu hatırlatmak için bir açıklama koyulabilir; READ komutunda ise böyle birşey söz konusu olamaz.

**ÖRNEK:** Dört sayının toplamını yazan bir program:
(Üç yolla ayrı ayrı yapılmıştır.)

**I-**
```
10 INPUT A,B,C,D
20 PRINT A + B + C + D
```

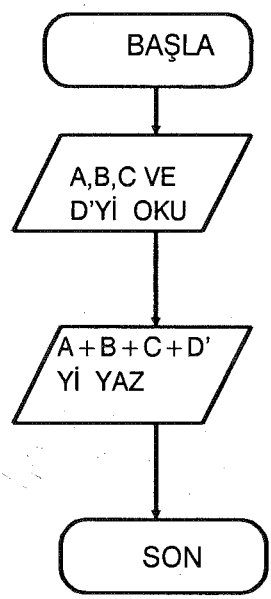
Bu program çalıştırıldığında bilgisayar A, B, C ve D değişkenleri için gerekli sayıları klavyeden okur. Aralarında virgülle birlikte dört sayı girilmediği sürece program 10 numaralı satırda durur.

II-

```
10 LET A = 5
20 LET B = 7
30 LET C = 9
40 LET D = 3
50 PRINT A + B + C + D
```

Sayıların atama yoluyla değişkenlere yerleştirildiği ikinci yolda, bilgisayar değerleri okumak için programı durdurmayacak, program çalıştırıldıktan çok kısa bir süre sonra istenilen sonuç ekrana çıkacaktır.

III–




```
10 READ A,B,C,D
20 PRINT A + B + C + D
30 DATA 5,7,9,3
```

READ kullanılarak yapılan bu program, komutların sayısı ve uygulanışları açısından ilk programa çok benzemekle birlikte, çalıştırıldığında elde edilecek sonuç açısından ikinci programla tamamen aynıdır. Bilgisayar READ A,B,C,D komutunu uygulamak amacıyla program içinde bir DATA satırı arayacak ve bu satırdaki sayısal değerleri sırasıyla A, B, C ve D değişkenlerine atayacaktır.

**NOT:** INPUT komutunda olduğu gibi, READ komutunda da değişkenlere atanacak değerlerin nitelik, sıra ve sayılarının, değişkenlere uygun olması gerekir. Aksi takdirde, bir hata mesajı çıkacak ve program duracaktır.

### DATA satırının yazılması:

DATA satırının en önemli özelliği, program içindeki yerinin hiç önemli olmamasıdır. Önemli olan, daha önce de belirtildiği gibi, bu satırda yazılan bilgilerin sıraları, adetleri ve niteliklerinin READ komutuyla birlikte kullanılan değişkenlere uygunluğudur. Bir DATA satırındaki bilgiler, sıraları bozulmadığı sürece değişik DATA satırlarında da yazılabilir:

```
DATA 5,7,9,3
```

yerine,

```
DATA 5
DATA 7
DATA 9
DATA 3
```

yazılabilir.

Bir DATA satırını birkaç parçaya ayırarak yazmak, genellikle değişik bilgi gruplarını ayırmak için kullanılır. Programın sonucuna bir etkisi olmamakla birlikte, bu tür yazılım, programcının daha sistemli ve güvenilir bir çalışma yapması için yararlıdır.

**ÖRNEK:**

**DATA Ahmet,35,13,Cemal,36,22,Hasan,41,23**

yerine

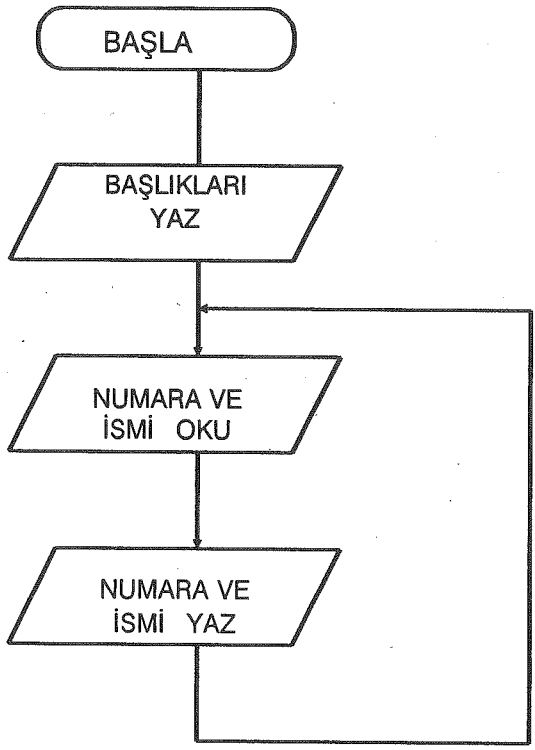
**DATA Ahmet,35,13**
**DATA Cemal,36,22**
**DATA Hasan,41,23**

yazılırsa, programın biraz daha uzamasına rağmen, hem program daha düzenli bir hale getirilmiş olur, hem de meydana gelebilecek bir yanlışlığın bulunması ve düzeltilmesi kolaylaşır. Buna karşın programın çalışması her iki durumda da aynı olacaktır.

DATA satırında yazılan bilgilerin sayısı, READ komutuyla okunması gereken bilgilere eşit olmalıdır. Eğer bu sayı gerekenden daha az olursa, program bir hata mesajı (Out of DATA) vererek duracaktır. DATA satırına gereğinden fazla bilgi yazılması, programda bir hataya yol açmamasına rağmen, özellikle uzun programlarda, daha sonradan ortaya çıkabilecek karışıklıklara neden olabilir.

**ÖRNEKLER:**

1) DATA satırından okunan öğrenci numarası ve isimlerini, boş bir ekran üzerine 'NUMARA' ve 'İSİM' başlıkları altında listeleyen bir program:




```
10 CLS
20 PRINT "NUMARA","İSİM"
30 PRINT "------","-----"
40 READ N,A$
50 PRINT N,A$
60 GOTO 40
70 DATA 464,ALİ,582,HASAN,117,AYŞE,1161,ŞULE,550,
FERHAN
```

Bu program çalıştırıldığında, ekran silindikten sonra şu görüntü elde edilecektir:

```
NUMARA       İSİM
-------      -----
464          ALİ
582          HASAN
117          AYŞE
1161         ŞULE
550          FERHAN
Out of DATA in 60
```

Listenin en altındaki satır, DATA satırının yetersiz olduğunu gösteren hata mesajıdır. Normal olarak, bu istenmeyen bir durumdur. Fakat bu program GOTO komutunun kullanılışıyla sonsuz bir döngüye girecektir ve hata çıkmaması için DATA satırında sonsuz sayıda bilgiye ihtiyaç duyacaktır. Sonsuz sayıda bilgiyi DATA satırına yazmak olanaksız olduğundan, bu yöntem kullanıldığında, bu mesajın alınması kaçınılmazdır.

2) Aynı programın GOTO komutu kullanılmadan yazılması:

```
10 CLS
20 PRINT "NUMARA","İSİM"
30 PRINT "----------","---------"
40 READ N,A$
50 PRINT N,A$
60 READ N,A$
70 PRINT N,A$
80 READ N,A$
90 PRINT N,A$
100 READ N,A$
110 PRINT N,A$
120 READ N,A$
130 PRINT N,A$
140 DATA 464,ALİ,582,HASAN,117,AYŞE,1161,ŞULE,550,
FERHAN
```

Diğer bir yol da şu olabilir:

```
10 CLS
20 PRINT "NUMARA","İSİM"
30 PRINT "----------","-------"
40 READ N1,A1$,N2,A2$,N3,A3$,N4,A4$,N5,A5$
50 PRINT N1,A1$
60 PRINT N2,A2$
70 PRINT N3,A3$
80 PRINT N4,A4$
90 PRINT N5,A5$
100 DATA 464,ALİ,582,HASAN,117,AYŞE,1161,ŞULE,550,
FERHAN
```

Görüldüğü gibi, son iki yol hata mesajı çıkarmamasına rağmen, program yazımı açısından çok daha uzun ve karmaşıktır. Doğru sonuca ulaşabilmek için gerekli bilgiler daha sonraki konular arasında verilecektir.

**RESTORE komutu:**

DATA satırındaki bilgilerin okunma işlemi, her zaman programdaki ilk DATA satırındaki ilk bilgiden başlayarak ve sıralı olarak yapılır. Normalde bir kez okunmuş bir bilginin bir kez daha okunması mümkün değildir. Okunmuş olan bilgilerin baştan başlayarak tekrar okunabilmesi veya okuma işleminin belirli DATA gruplarından başlatılabilmesi için RESTORE komutu kullanılabilir. Diğer bir deyişle, RESTORE komutu, program içindeki DATA satırlarının hiç okunmamış kabul edilmesini sağlar.

**ÖRNEK:** DATA satırından okunan iki sayıyı ekrana yazıp başa dönen sonsuz bir program:

```
10 READ A,B
20 PRINT A,B
30 RESTORE
40 GOTO 10
50 DATA 4,7
```

Bu program RESTORE kullanılmadan yapılırsa, bilgisayar READ komutuyla DATA'daki sayıları okuduktan sonra başa dönüp tekrar bilgi okumak istediğinde DATA satırında başka sayı olmadığı için hata mesajı vererek programı durduracaktır. RESTORE komutu bunu engelleyecek, her READ komutundan sonra veri okunma işlemini başa döndürecektir.

RESTORE komutu, aynı zamanda satır numarası ile de kullanılır. Bu durumda bilgi okuma işlemi, komuttan sonra verilen satır numarasındaki DATA satırından başlayacaktır.

**ÖRNEK:** DATA satırında verilen 'AHMET,23,AYŞE,25' verilerinden yararlanarak AHMET'e ait bilgileri bir kez, AYŞE'ye ait bilgileri iki kez okuyup ekrana yazan bir program:

```
10 READ A$,B
20 PRINT A$,B
30 READ A$,B
40 PRINT A$,B
50 RESTORE 90
60 READ A$,B
70 PRINT A$,B
80 DATA AHMET,23
90 DATA AYŞE,25
```

Bu örnekte, 90 numaralı DATA satırındaki veriler, 30 ve 60 numaralı READ satırlarında iki kez okunacaktır. RESTORE komutundan sonra satır numarası (90) kullanılarak, yalnızca AYŞE'ye ait bilgilerin baştan okunması sağlanmıştır. Eğer bu satır numarası kullanılmamış olsaydı, bilgiler en baştan okunmaya başlayacak, AYŞE yerine AHMET'e ait bilgiler tekrarlanacaktı.

**SORULAR:**

1) Bilgileri DATA satırından okuyarak aşağıdaki listeyi boş bir ekrana yazan bir program yapınız.

| NOTLAR |
|---|
| 9 |
| 8 |
| 5 |
| 4 |
| 8 |
| --- |
| TOPLAM = -- |

2) Klavyeden girilen iki sayının toplamını DATA'dan okunan bir sayıyla çarparak sonucun karesini ekrana yazan bir program yapınız.

# PROGRAM AKIŞINI KOŞULLANDIRMA

## ( IF ... THEN TERİMLERİ) :

Program akışı, belirli durumlarda belirli yönlere gidebilecek şekilde düzenlenebilir. Bu yönlendirme, programın çalışması süresinde gerçekleşecek koşullara göre uygulanır. Bunu yapabilmek için BASIC'te IF ... THEN terimleri kullanılır.

Genel kullanım:

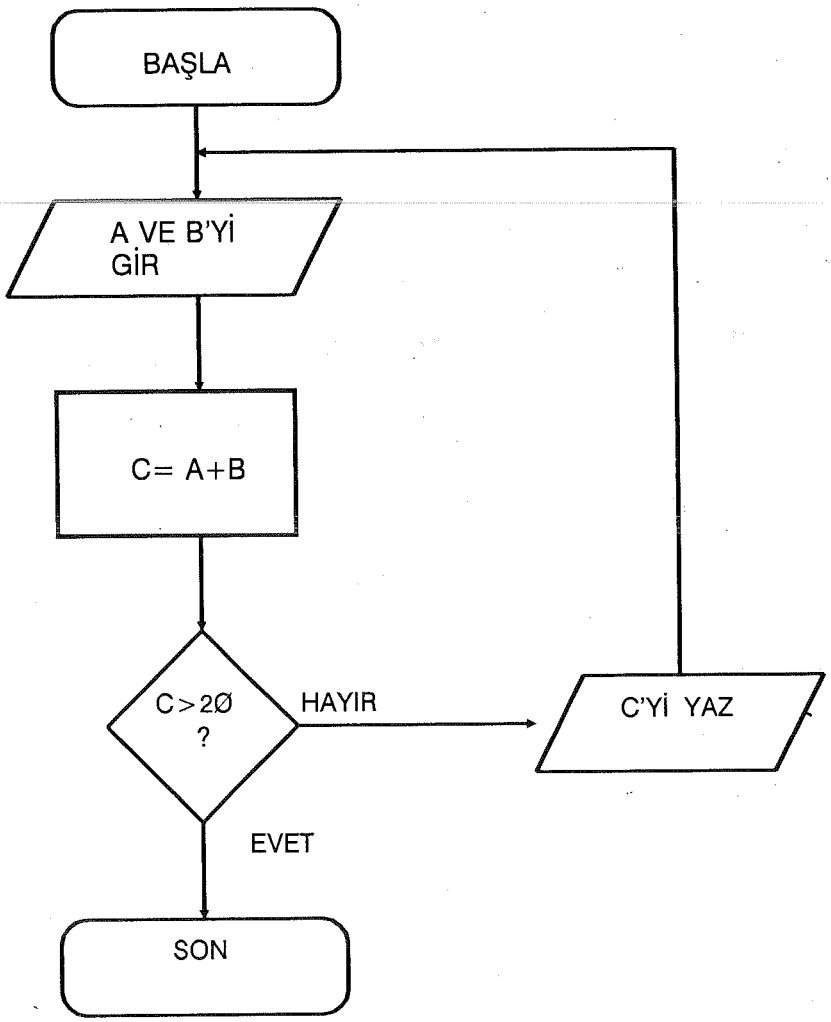
**IF <koşul(lar)> THEN <komut(lar)>**

IF ve THEN arasında yazılacak olan koşul(lar) doğruysa (geçerliyse), THEN teriminden sonraki komut(lar) uygulanır. Aksi takdirde, bilgisayar bu satırı tamamen atlayarak bir sonraki program satırına geçer.

**ÖRNEK:**

```
IF A = 5 THEN PRINT A
```

Bu örneğe göre, A'nın değeri 5 ise ekrana bu değer yazılacak, aksi takdirde programın akışı bir sonraki satırdan devam edecektir.




**ÖRNEK:**

```
10 INPUT A,B
20 C = A + B
30 IF  C>20  THEN  END
40 PRINT C
50 GOTO 10
```

Bu örnekte, klavyeden girilen iki sayının toplamı üçüncü bir değişkende saklanıyor. IF ... THEN satırında bu değişkenin değeri 20 ile karşılaştırılıyor; eğer toplam 20'den büyük olursa program duruyor, değilse bu toplam ekrana yazılarak program tekrar başa dönüyor. Dikkat edilirse, örnek programdaki THEN teriminden sonra END kullanılmıştır. Bunun gibi, BASIC dilindeki bütün komutlar yapılan programın amacına göre THEN teriminden sonra kullanılabilir. Aynı programdaki IF ... THEN satırı şu şekilde yazılabilir:

**30 IF C> 20 THEN GOTO 60**

Bu durumda 60 numaralı satırı eklemek gerekecektir:

**60 END**

BASIC dilinde, karşılaştırma operatörleri olarak aşağıdaki işaretler kullanılmaktadır:

| İşaret | BASIC karşılığı |
|---|---|
| Eşit | = |
| Büyük | > |
| Küçük | < |
| Büyük veya eşit | >= |
| Küçük veya eşit | <= |
| Eşit değil | <> |

Görüldüğü gibi, karşılaştırma operatörleri yalnızca üç karakterin kombinasyonlarıyla oluşmaktadır.

**ELSE terimi:**

Eğer istenirse, bir IF ... THEN cümlesinin sonuna eklenebilen ELSE terimi, koşulun yanlış olduğu durumlarda hangi komutun uygulanacağını belirtmek için kullanılır.

Genel kullanım:

**IF <koşul(lar)> THEN <komut(lar)> ELSE <komut(lar)>**

ELSE teriminin eklenmesiyle, cümlenin anlamına bir yenilik getirilmiştir: Eğer koşul doğruysa THEN teriminden sonra gelen komut, yanlışsa ELSE teriminden sonraki komut uygulanacaktır. Bu durumda koşulun doğru veya yanlış oluşuna göre, bu satırda bulunan komutlardan birisi mutlaka uygulanacaktır.

**ÖRNEK:**

```
IF A = 5 THEN PRINT A ELSE PRINT A * 2
```

Bu satırda, A değişkeninin değeri 5 ise ekrana bu değer yazılacak, 5 dışındaki değerler için A'nın iki katı yazılacaktır. ELSE eklendiğinde, koşulun yanlış olması durumunda program akışı bir sonraki satıra geçmez; ELSE teriminden sonraki komut uygulanır.

**ÖRNEK:**

```
10 INPUT A,B
20 C = A * B
30 IF C>20 THEN PRINT "Sayılar çok büyük" ELSE
PRINT C
40 GOTO 10
```

Bu örnekte, daha önce olduğu gibi, klavyeden girilen A ve B sayılarının toplamı C değişkeninde saklanıyor; eğer C'nin değeri 20'dan büyükse bir uyarı mesajı (Sayılar çok büyük), değilse C'nin değeri, yani sayıların toplamı yazılıyor. Bu işlemlerden herhangi biri gerçekleştikten sonra, program 40 numaralı satırdan devam ediyor.

**AND ve OR terimleri:**

Eğer birden çok koşul bir arada kullanılmak isteniyorsa, bu koşulların arasında AND veya OR kullanmak gerekir.

Sonucun doğru olması için her iki koşulun da doğru olması isteniyorsa, bu koşulların arasında AND (anlamı : ve) kullanılmalıdır:

**IF A = 5 AND B>8 THEN PRINT A * B**

Bu satırdaki PRINT komutunun uygulanabilmesi için her iki koşulun da doğru olması gerekmektedir. Yani hem A'nın 5'e eşit olması, hem de B'nin 8'den büyük olması gereklidir. Koşullardan birinin dahi yanlış olması, sonucun da yanlış olmasına ve PRINT komutunun uygulanmamasına yol açacaktır.

Sonucun doğru olması için iki koşuldan herhangi birinin doğru olması yeterli ise, bu koşulların arasında OR (anlamı : veya) kullanılmalıdır:

**IF C<D OR D = 3 THEN PRINT "Sayılar doğru"**

Bu satırdaki PRINT komutunun uygulanabilmesi için, koşullardan yalnızca birinin doğru olması yeterlidir. Diğer bir deyişle, ya C D'den küçük olmalı, ya da D 3'e eşit olmalıdır. Sonucun yanlış olması için her iki koşulun da yanlış olması gerekmektedir; diğer durumlarda (koşullardan birinin veya ikisinin birden doğru olması halinde) sonuç da doğru olacak ve PRINT komutu uygulanacaktır.

**NOT terimi:**

Türkçe anlamı 'değil' olan bu terim, IF ... THEN cümlelerinde koşulların tersini almak için kullanılır:

**IF NOT A = B THEN PRINT A * B**

Bu satırdaki PRINT komutunun uygulanabilmesi için, 'A = B' koşulunun tersinin doğru olması, yani A'nın B'ye eşit olmaması gerekmektedir. Aynı satır 'IF A <> B THEN PRINT A * B' şeklinde de yazılabilirdi.

Eğer AND veya OR ile birleştirilmiş bir grup koşulun NOT ile birlikte kullanılması isteniyorsa, bu koşul grubu parantez içine alınmalıdır:

**IF NOT ( A>B OR B = C ) THEN GOTO 4Ø**

Bu satırda, 'A>B OR B = C' koşulunun tersinin alınabilmesi için,

bu koşul grubu parantez içine alınmıştır. Eğer parantez kullanılmazsa, NOT terimi, yalnızca 'A>B' koşulu için geçerli olacaktır.

**ÖRNEKLER:**

1) Klavyeden girilen iki sayı birbirine eşitse bu sayıların toplamını, değilse çarpımını yazan bir program:

```
10 INPUT A,B
20 IF A = B THEN PRINT A + B ELSE PRINT A * B
```

Bu program, klavyeden girilen A ve B değişkenlerinin birbirine eşit olduğu durumlarda 'PRINT A + B' komutunu, diğer durumlarda ise 'PRINT A * B' komutunu uygulayacaktır. Böylece, sayılar birbirine eşitse toplam, değilse çarpım yazılacaktır.

2) Klavyeden girilen iki sayı da 8'den büyükse başa dönen, değilse ekranı silip sona eren bir program:

```
10 INPUT A,B
20 IF A>8 AND B>8 THEN GOTO 10 ELSE CLS
```

Bu program, sayıların ikisinin de 8'den büyük olduğu durumlarda başa dönmelidir. Bu yüzden iki koşulun arasında AND terimi kullanılmıştır. dikkat edilecek bir başka nokta da, CLS komutunun uygulanmasından sonra programın durmasıdır. Daha önce de belirtildiği gibi, THEN ve ELSE terimlerinden sonra gelen komutlar uygulandıktan sonra, program akışı bir sonraki satırdan devam eder. Yalnız bu programda THEN'den sonra GOTO komutu kullanılmış olduğundan, program akışı bir sonraki satırdan değil, GOTO'nun belirlediği satırdan devam edecektir.

3) Klavyeden girilen iki sayıdan en az birisi 8'den büyükse başa dönen, aksi takdirde ekranı silip sona eren bir program:

```
10 INPUT A,B
20 IF A>8  OR B>8 THEN GOTO 10 ELSE CLS
```

Bu programın bir öncekinden farkı, GOTO komutunun uygulanması için, sayılardan yalnızca birinin koşula uymasının yeterli olması-

dır. Bu nedenle koşulların arasında OR kullanılmıştır. A>8 veya B>8 koşullarından birinin dahi doğru olması, sonucun doğru olmasını sağlayacaktır. CLS komutunun uygulanması için, iki koşulun da yanlış olması gerekmektedir.

4) Klavyeden girilen iki sayı birbirine eşitse duran, değilse sayıların çarpımını yazıp başa dönen bir program:

```
10 INPUT A,B
20 IF A = B THEN END ELSE PRINT A * B
30 GOTO 10
```

Bu örnekte, sayıların eşit olduğu durumda programın durması isteniyor. Bu nedenle, THEN teriminden sonra END komutu kullanılmalıdır. Sayıların eşit olmadığı durumlarda ise ELSE'ten sonra gelen PRINT komutu uygulanacak ve program bir sonraki satırdan devam edecektir. IF ... THEN satırından sonraki satır GOTO satırı olduğu için, koşula uymayan sayı çiftleri için bilgisayar çarpımlarını yazdıktan sonra 10 numaralı satıra, yani programın başına dönecektir.

5) Klavyeden girilen üç sayıdan ilk ikisi birbirine eşit değilse veya son ikisi birbirine eşitse sayıların çarpımını, aksi takdirde toplamını yazan bir program:

```
10 INPUT A,B,C
20 IF A<>B OR B = C THEN PRINT A * B * C ELSE
PRINT A + B + C
```

Bu programda, ilk iki sayının (A ve B) eşit olmaması koşulu yazılırken, eşit değil anlamında, küçük (<) ve büyük (>) işaretleri birlikte kullanılmıştır. Aynı koşul, 'NOT A = B' şeklinde de yazılabilir.

6) DATA'dan okunan iki kelime birbirine eşitse duran, değilse kelimeleri yan yana yazdıktan sonra duran bir program:

A-
```
10 READ A$,B$
20 IF A$ = B$ THEN GOTO 40
30 PRINT A$,B$
40 END
50 DATA AHMET,MEHMET
```

B-
```
10 READ A$,B$
20 IF A$ = B$ THEN END ELSE PRINT A$,B$
30 DATA AHMET,MEHMET
```

Aynı sonuca ulaşmak için iki ayrı yöntem kullanılmıştır. Daha uzun olan birinci yöntemde ELSE kullanılmamış, GOTO komutuyla program END komutuna gönderilmiştir. İkinci yöntemde ise ELSE kullanılarak program daha kısa yoldan yapılmıştır.

7) Klavyeden girilen iki sayının toplamı 100'den küçükse girilen sayıları, değilse iki sayının toplamını yazan bir program:

```
10 INPUT A,B
20 IF A + B<100 THEN GOTO 50
30 PRINT A + B
40 END
50 PRINT A,B
```

**NOT:** ELSE kullanılarak çok daha kolay bir biçimde yapılabilecek olan bu programda, örnek olarak ELSE kullanılmamıştır.

## SONLANDIRICI KAVRAMI:

Bir program içindeki sonsuz bir döngüden çıkmak için kullanılan yönteme sonlandırıcı kullanımı denir.

## ÖRNEKLER:

1) Klavyeden girilen kişiye ait doğum yılı negatif olana kadar bu kişinin yaşını ekrana yazarak başa dönen, doğum yılı negatif olduğunda duran bir program:

```
10 INPUT "DOĞUM YILI ";DY
20 IF DY<0 THEN END
30 PRINT 1989 - DY
40 GOTO 10
```

Bu programdaki sonlandırıcı, negatif sayı olarak (doğum yılının sıfırdan küçük olması) belirlenmiştir. Girilen yıl negatif bir sayı değilse, yaş ekrana yazılacak ve program başa dönecektir. Negatif yıl girildiğinde ise program duracaktır.

```
RUN
DOĞUM YILI ? 1972
 19
DOĞUM YILI ? 1945
 46
DOĞUM YILI ? 1981
 10
DOĞUM YILI ? -1
 OK
```

Bu tür programlarda sonlandırıcının girilen bilgi için normalde hiçbir zaman gerçekleşmeyecek bir değer olması gerekir. Yıl olarak negatif sayılar kullanılamayacağından, bu programda sonlandırıcı olarak negatif sayılar tercih edilmiştir. Örneğin 1975 gibi bir sayı sonlandırıcı olarak kullanılmış olsaydı, program belki de istenmeyen bir

zamanda (1975 yılında doğmuş birinin yaşını hesaplamak yerine) duracaktı.

2) Klavyeden girilen isim 'XX' olmadığı sürece bu ismi yazıp başa dönen bir program:

```
10 INPUT "İSİM GİRİNİZ ";A$
20 IF A$ = "XX" THEN END
30 PRINT A$
40 GOTO 10

RUN
İSİM GİRİNİZ ? ALİ
ALİ
İSİM GİRİNİZ ? AYŞE
AYŞE
İSİM GİRİNİZ ? AHMET
AHMET
İSİM GİRİNİZ ? XX
OK
```

Bu programda sonlandırıcı olarak 'XX' gibi bir karakter dizisinin seçilmesinin nedeni, hiçbir ismin 'XX' olamayacağıdır.

NOT: Sonlandırıcının seçiminde, girilen bilginin sayısal veya alfabetik olmasına dikkat edilmelidir.

| 3) NUMARA | İSİM |
|---|---|
| 464 | ALİ |
| 582 | HASAN |
| 117 | AYŞE |
| 1161 | ŞULE |
| 550 | FERHAN |

Yukarıdaki listeyi boş bir ekrana çıkaran bir program:

```
10 CLS
20 PRINT "NUMARA","İSİM"
30 PRINT "------","-------"
40 READ N,A$
```

```
50 IF N = -1 THEN END
60 PRINT N,A$
70 GOTO 40
80 DATA 464,ALİ,582,HASAN,117,AYŞE,1161,ŞULE,550,
FERHAN, -1,X
```

Bu program, istenilen sonucu, daha önce belirtilmiş olan hata mesajını (Out Of. DATA) çıkartmaksızın verecektir. Bunu sağlayan, DATA satırındaki sonlandırıcının (-1), 50 numaralı satırdaki IF ... THEN cümlesiyle kontrol edilmesidir. -1 değeri okunduğunda, program sonsuz döngüden çıkacak ve duracaktır.

DATA satırında -1'den sonra yazılmış olan X harfinin hiçbir özelliği yoktur. Veriler numara ve isim olarak sıralı dizildikleri ve çift okundukları için, -1'den sonra herhangi bir veri yazılmaması, 40 numaralı satırdaki READ komutuyla birlikte kullanılan A$ değişkenini karşılıksız bırakacak ve hata mesajının çıkmasına neden olacaktır.

**( : ) işareti :**

BASIC programlamada, eğer komutlar arka arkaya yazılmak istenirse, aralarında ( : ) işareti kullanılmalıdır. Yalnız, tecrübesiz programcıların bu işareti bilinçsizce kullanmaları tavsiye edilmez. İki nokta üst üste işaretinin özellikle kullanılması gereken yer, IF ... THEN cümleleridir. Bir koşula bağlı olarak birden çok komut verilmek isteniyorsa, bu komutlar arasında ( : ) bulunmalıdır.

**ÖRNEKLER:**

1) Klavyeden girilen iki sayı birbirine eşitse ekranı silen ve başa dönen bir program:

```
10 INPUT A,B
20 IF A = B THEN CLS : GOTO 10
```

Bu programda, koşulun doğru olduğu durumda hem ekranın silinmesi, hem de programın başa dönmesi istendiği için, CLS ve GOTO komutları arka arkaya verilmelidir. Bu nedenle bu komutların arasında ( : ) kullanılmıştır.

2) Klavyeden girilen üç sayının toplamı 10'dan büyükse 'BÜYÜK' yazan, değilse 'KÜÇÜK' yazıp başa dönen ve her iki koşulda da sayıları ekrana yazan bir program:

```
10 INPUT A,B,C
20 IF A+B+C>10 THEN PRINT "BÜYÜK":PRINT A,B,C

ELSE PRINT "KÜÇÜK":PRINT A,B,C : GOTO 10
```

Burada, IF ... THEN cümlesindeki ELSE teriminden sonra gerekli olan üç komut, aralarında ( : ) işareti kullanılarak arka arkaya yazılmıştır.

**TRON ve TROFF komutları:**

| | | |
|---|---|---|
| TRACE | : | İz |
| ON | : | Açık |
| OFF | : | Kapalı |

Yukarıdaki üç sözcükten türemiş olan TRON ve TROFF komutları, programın çalışması sırasında satır numaralarının ekranda görüntülenmesiyle ilgilidir.

TRON komutu, satır numaralarının izlenebilmesi için kullanılır. Bu komut verildikten sonra, program çalışırken programın ne zaman hangi satırları uyguladığı sürekli olarak ekranda izlenebilir. Böylece herhangi bir yanlışlığın programın tam olarak hangi satırında bulunduğunu belirlemek mümkündür.

TROFF komutu ise, TRON ile açılmış olan izleme durumunu kapatır. Böylece program tekrar eski çalışma şekline döner.

**SORULAR:**

1) Klavyeden girilen iki sayı birbirine eşitse duran, değilse ikisinin çarpımını yazan ve başa dönen bir program yapınız.

2) Klavyeden girilen iki sayıdan büyük olanın küçük olana bölümünü yazan bir program yapınız.

3) Klavyeden girilen not 5'ten küçükse 'ZAYIF', 5 ve 8 arasındaysa 'İYİ', 8'den büyükse 'PEKİYİ' yazan bir program yapınız.

4) Klavyeden girilen sayı 25'e eşit olana kadar bu sayının iki katını yazan bir program yapınız.

5) Aşağıdaki DATA satırındaki sayıları listeleyen ve toplamlarını hesaplayarak yazan bir program yapınız. (Sonlandırıcı olarak kullanılan negatif sayı yazılmayacak ve toplama dahil edilmeyecektir.)

DATA 23,71,59,43,57,27,33,0,80,54,-1

## SAYAÇ KULLANIMI:

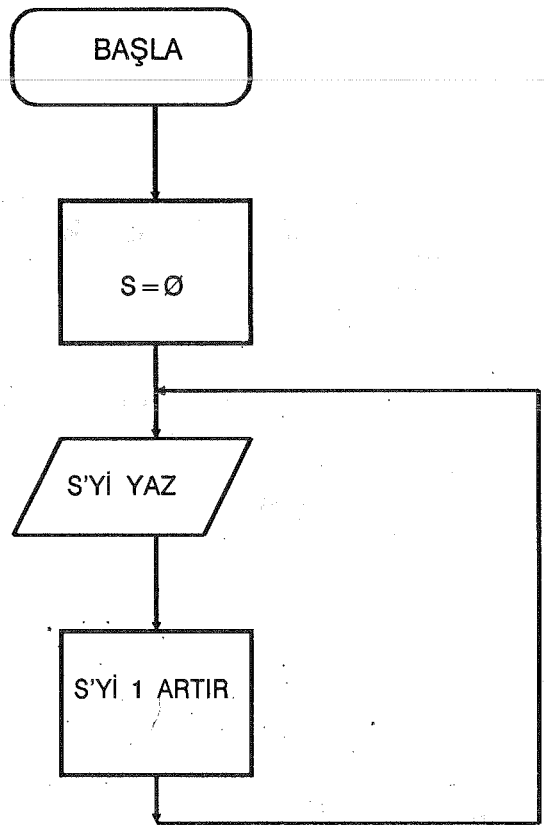
Bir programda, programla ilgili bazı veriler sayılmak istendiğinde, sayaç kullanılır. Sayaç, özel bir amaçla kullanılan herhangi bir sayısal değişkendir. Saymak için gerekli komut her uygulandığında, sayaç olarak kullanılan değişkenin değeri 1 birim artar. Bu komutun özelliği, (=) işaretinin her iki tarafında da sayaç değişkeninin yer almasıdır.

**ÖRNEK:**

```
S = S + 1
```

Bu komutun amacı, sayaç olarak kullanılan S değişkeninin değerini kendisinin 1 fazlasına eşitlemek, yani 1 arttırmaktır. En başta S'nin değerinin 0 olduğu kabul edilirse, bilgisayar bu komutu ilk uyguladığında sayaç değeri 0 + 1'e, yani 1'e eşitlenecektir. Aynı komutun ikinci uygulanışında sayaç 1 + 1, yani 2 olacaktır. Daha sonra 3, 4, 5, 6, 7, 8 değerlerini alacak ve bu sonsuza kadar 1'er artarak devam edecektir.

**S = S + 1** komutunun program içinde uygulanması şu şekilde olabilir:




```
10 S = 0
20 PRINT S
30 S = S + 1
40 GOTO 20
```

Bu programda, 'PRINT S' ve 'S = S + 1' komutları sonsuza dek uygulanacak, her keresinde S'nin değeri 1 artacak ve böylece 0'dan başlayıp birer artarak bütün sayılar ekrana listelenecek, sonuç aşağıdaki gibi olacaktır:

```
RUN
0
```

```
1
2
3
4
5
6
7
8
9
10
11
12
.
.
.
```

Programın ilk komutu olan 'S = 0', birçok bilgisayarda gereksizdir, çünkü programda bir değişkene hiçbir değer verilmemişse bu değişkenin değeri 0 olarak kabul edilecektir. Buna karşın, bazı bilgisayarlar bütün değişkenlerin en başta tanıtılmasını gerektiren BASIC yorumlayıcılarına sahip olduğundan, bu tür bilgisayarlarla çalışırken bu komut gerekli olabilir.

Sayı listesi 0 dışında herhangi bir sayıdan başlatılmak istenirse, başlangıç değerini belirleyen ilk satır, istenildiği gibi değiştirilebilir. Örneğin programın başına yazılacak 'S = 15' komutu, sayıların 15'ten başlayarak sonsuza kadar yazılmasını, 'S = -5' komutu ise listenin -5'ten başlamasını sağlayacaktır.

Bu listede sayıların artış hızı değiştirilmek istendiğinde, 30 numaralı satır yenilenebilir. Örneğin sayıların 1'er 1'er değil de, 2' şer 2'şer artması istenirse, satır şu şekilde yazılabilir:

```
30 S = S + 2
```

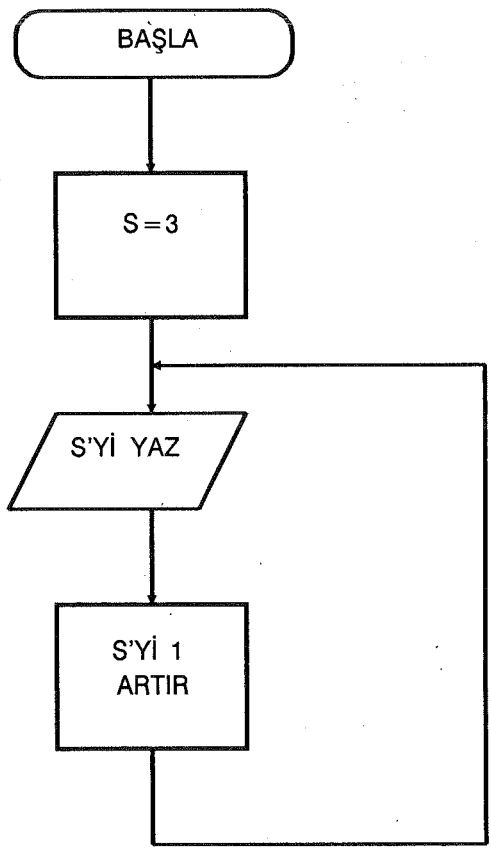
Program çalıştırıldığında elde edilecek sonuç aşağıdaki gibi olacaktır:

```
RUN
0
2
4
6
8
10
12
14
.
.
.
```

Eğer sayıların artması değil de, eksilmesi isteniyorsa, 30 numaralı satır üzerinde yapılacak bir değişiklikle bu da sağlanabilir. Örneğin sayıların 3'er 3'er eksilmesi için 30 numaralı satırdaki komut S = S - 3 olarak yazılabilir.

Şu ana kadar yapılan sayaç kullanımıyla ilgili bütün örnekler, sonsuz programları içermektedir. Yani, program çalışmaya başladıktan sonra kullanıcının müdahalesi olmaksızın hiç durmayacaktır. Bunun nedeni, bu programların sonlarına koyulan ve sona geldiğinde programın tekrar başa dönmesini sağlayan GOTO komutudur. Sayı listesinin belli bir sayıya gelindiğinde durması, basit bir IF ... THEN cümlesi kullanılarak sağlanabilir.

**ÖRNEK**



```
10 S = 3
20 PRINT S
30 S = S + 1
40 GOTO 20
```

Bu program, 3'ten başlayıp 1'er 1'er artmak suretiyle bütün sayıları sonsuza dek yazacaktır. Bu listenin herhangi bir sayıya, örneğin 20'ye geldiğinde durması istenirse, 30 ve 40 numaralı satırlar arasına şu IF ... THEN cümlesi koyulabilir:

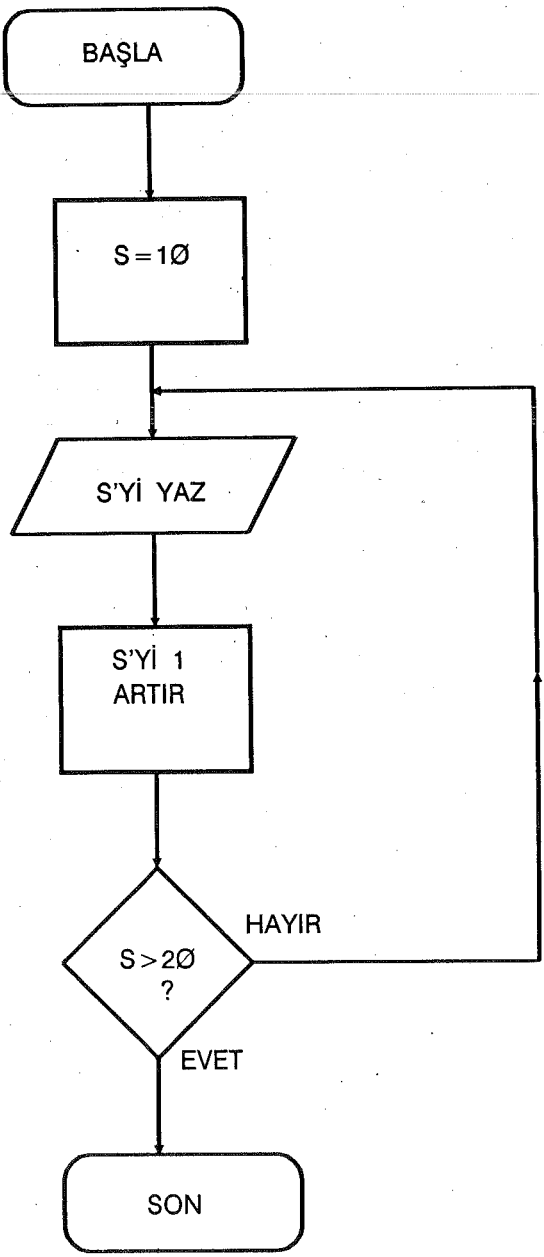
```
35 IF S>20 THEN END
```

Böylece, 20 sayısı ekrana yazıldıktan sonra, S = S + 1 komutu uygulanacak ve sayaç değeri 21 olacaktır. 35 numaralı satırdaki koşula uyum sağlanmış olduğundan, bilgisayar THEN teriminden sonra gelen END komutunu uygulayacak ve program duracaktır. Bu şekilde, yalnızca 3 ve 20 arasındaki sayılar (bu sayılar da dahil olmak üzere) listelenmiş olacaktır.

**NOT:** Bu tür bir program özel tuşlara basmak yoluyla veya bir IF ... THEN satırı yardımıyla durdurulduktan sonra, CONT komutu verilerek kaldığı yerden devam ettirilebilir. Örneğin en son örnekteki program 3'ten 20'ye kadar sayıları yazıp durduktan sonra, kullanıcı tarafından verilecek CONT komutu, programın kaldığı yerden devam etmesini sağlayacaktır. (Ancak bir tek sayı yazıldıktan sonra bu program IF ... THEN satırı nedeniyle tekrar duracaktır.)

**ÖRNEKLER:**

1) 10'dan 20'ye kadar bütün sayıları listeleyen bir program:



```
10 S = 10
20 PRINT S
30 S = S + 1
40 IF S>20 THEN END
50 GOTO 20
```

Bu programda S değişkeni için ilk değer olarak 10 verilmiştir. 30 numaralı satırdaki 'S = S + 1' komutu, sayaç değerini her keresinde 1 arttıracak ve bu değer 20'yi aştığı zaman (ekrana yazılmadan) program duracaktır.

2) 1'den 15'e kadar tek sayıları ekrana listeleyen bir program:

```
10 S = 1
20 PRINT S
30 S = S + 2
40 IF S>15 THEN END
50 GOTO 20
```

Bu kez 30 numaralı satırdaki artış, +2 olarak verilmiştir. Böylece listedeki sayıların 2'şer artması sağlanacaktır.

3) 12'den 3'e kadar sayıları 3'er eksilterek listeleyen program:

```
10 S = 12
20 PRINT S
30 S = S - 3
40 IF S<3 THEN END
50 GOTO 20
```

Bu programda dikkat edilmesi gereken nokta, 40 numaralı satırdaki IF ... THEN cümlesidir. Bu satırda, diğer programların aksine, büyük işareti (>) yerine küçük işareti (<) kullanılmıştır. Bunun nedeni, sayıların büyükten küçüğe doğru listelenmesidir. Bu durumda programı durdurması gereken değer, 3 sayısından büyük değil, küçük olmalıdır. (Bu program için bu değer 0 olacaktır.)

4) 1'den başlayıp 3'er artan ve 25'ten başlayıp 2'şer eksilen sayıları iki ayrı sütun halinde listeleyen bir program:

```
10 S1 = 1
20 S2 = 25
30 PRINT S1,S2
40 S1 = S1 + 3
50 S2 = S2 - 2
60 GOTO 30
```

Bu programda, iki ayrı sayaç değişkeni kullanılmıştır. Başlangıç ve artış değerleri ayrı ayrı verilen bu sayaçlar, tek bir PRINT komutuyla ekrana yazdırılmış, bu sayede iki sütun oluşturulabilmiştir.

5) 4 ve 48 arasındaki tek sayıları ve karelerini iki sütunda listeleyen bir program:

```
10 S = 5
20 PRINT S,S^2
30 S = S + 2
40 IF S>48 THEN END
50 GOTO 20
```

Bu programdaki sayaç için başlangıç değeri olarak 5 ve artış değeri olarak 2 verilmiştir. Bunun nedeni, 4 ve 48 arasındaki tek sayılardan ilkinin 5 ve tek sayılar arasında da 2'şer fark olmasıdır. 20 numaralı satırdaki PRINT komutu, sayının kendisi ve karesi olmak üzere iki ayrı sütun oluşturacak ve sayaç değeri 48'i geçtiğinde (burada 49) program duracaktır.

Sayaç kullanımı, esas olarak program içindeki bir işlemi tekrarlatmak içindir. Örneğin belirli bir sayıda bilginin klavyeden veya DATA satırından okunması veya belli bir ifadenin birkaç kez ekrana yazılması, sayaç kullanımının amaçları arasında sayılabilir.

**ÖRNEKLER:**

1) Klavyeden girilen 5 sayının pi ile çarpımlarını yazan bir program:

```
10 S = 1
20 INPUT A
30 PRINT A * 3.14
40 S = S + 1
50 IF S>5 THEN END
60 GOTO 20
```

Bu programda sayaç değişkenine verilen ilk değer 1'dir. Klavyeden girilen ve pi sayısı ile çarpımı yazılan her sayı için sayacın değeri 1 artacak ve bu döngü sayaç değeri 5'i (amaçlanan tekrar sayısı) aşana kadar sürecektir. Sayacın alacağı 1, 2, 3, 4, 5 değerlerinin her biri için klavyeden girilen bir sayının işleme sokulduğu düşünülürse, işlemin 5 kez tekrarlandığı görülür. Dikkat edilirse, sayacı ekrana yazmak için kullanılan PRINT S komutu bu programda kullanılmamıştır, çünkü bu program için sayaç kullanımının tek amacı klavyeden bilgi girişi ve işlem sonucunu yazma bölümünü tekrarlatmaktır.

2) DATA'dan okunan 7 sayıyı ekrana yazan bir program:

```
10 S = 1
20 READ A
30 PRINT A
40 S = S + 1
50 IF S>7 THEN END
60 GOTO 20
70 DATA 4,7,3,8,9,2,1
```

Programın sonundaki GOTO 20 komutu, 20 ve 60 numaralı satırlar arasında bir döngü oluşturacak, bilgisayar bu döngünün içinden ancak sayaç değeri istenilen sınırı aştığı zaman çıkabilecektir. IF ... THEN satırında kullanılacak sayının değiştirilmesiyle, bu döngü istenildiği kadar tekrarlanabilir. bunu yaparken, DATA satırındaki bilgilerin adedine de dikkat edilmelidir. İstenilenden fazla bilgi olması, programın çalışması açısından bir sorun yaratmaz, yalnızca gereğinden fazla bellek kullandığı için sakıncalıdır. Ancak, DATA satırında sayaç değerinden daha az bilgi varsa, program bütün bu bilgileri okuduktan sonra bir hata mesajı vererek istenmeyen bir biçimde duracaktır.

3) 8 kişiye ait isim ve yaşı DATA satırından okuyarak ekrana listeleyen bir program:

```
10 CLS
20 PRINT "İSİM","YAŞ"
30 PRINT "--------","------"
40 READ A$,Y
```

```
50 PRINT A$,Y
60 S = S + 1
70 IF S>8 THEN END
80 GOTO 40
90 DATA Yeşim, 15, Veli, 32, Hasan, 6, Dilek, 21,
Banu, 19, Ali, 20, Hayri,25,Ayşe,8
```

**SORULAR:**

1) 1 ve 15 arasındaki çift sayıları listeleyen bir program yapınız.

2) 'BİLGİSAYAR' kelimesini alt alta 8 kez yazan bir program yapınız.

3) Klavyeden girilen sayının karesini ekrana yazma işlemini 12 kez tekrarlayan bir program yapınız.

## *FOR ... NEXT Döngüsü:*

Sayaç kullanarak program içindeki bir işlemi tekrarlatmak, programcılıkta çok sık kullanılan bir yöntemdir. Birçok programın temeli, sayaç kullanımıyla elde edilen bir döngü üzerine kurulmuştur. Döngü kullanımının bu denli önemli ve vazgeçilmez olması nedeniyle, programlamayı kolaylaştırmak amacıyla, hemen hemen bütün programlama dillerinde döngü oluşturmak için özel komut grupları geliştirilmiştir. BASIC dilinde bu amaçla kullanılan komut grubuna FOR ... NEXT döngüsü adı verilir.

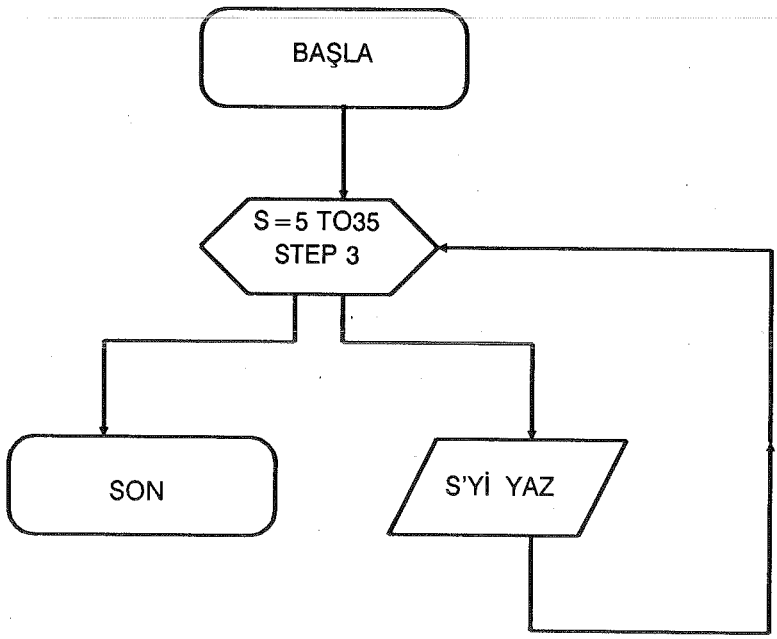
Genel kullanım:

**FOR <döngü değişkeni> = <başl. değeri> TO <bitiş değeri> [STEP<artış değeri>]**
**[komutlar]**
**NEXT [döngü değişkeni]**

**ÖRNEK:**

```
10 S = 5
20 PRINT S
30 S = S + 3
40 IF S>35 THEN END
50 GOTO 20
```

Bu program, 5'ten başlayarak 35'e kadar olan sayıları 3'er atlayarak yazar. Aynı program FOR ... NEXT döngüsü kullanılarak şu şekilde de yazılabilir:




```
10 FOR S = 5 TO 35 STEP 3
20 PRINT S
30 NEXT S
```

FOR ... NEXT kullanılarak yapılan ikinci program, işlev olarak birincinin tamamen aynısıdır. Birinci programda başlangıç değeri 10 numaralı satırda bir değer atama yoluyla belirlenmiştir (S = 5). Son değer 40 numaralı satırdaki IF ... THEN cümlesinde verilmiştir (IF S> 35). Sayıların kaçar kaçar artacağı da 30 numaralı satırdaki S = S + 3 ifadesinde gösterilmiştir. Bütün bu değerler, ikinci programda 10 numaralı satırdaki FOR cümlesinde belirtilmiştir. PRINT komutu iki programda da aynı şekilde kullanılmıştır.

Birinci programda döngünün devam etmesini sağlayan komut, GOTO komutudur. Döngünün ne zaman sona ereceğini belirleyen ise IF ... THEN satırıdır. İkinci programdaki NEXT komutu, her iki

özelliği de içermektedir. Hem döngünün tekrarlanmasını, hem de gerektiği zaman sona ermesini sağlar.

FOR ... NEXT döngüsünün temeli, bir değişkenin aldığı değerler üzerine kurulmuştur. Bu değişkene döngü değişkeni adı verilir. Döngü değişkeni, FOR teriminden sonra mutlaka yazılmalıdır. Bu değişkenin döngü süresince alacağı değerler FOR satırında belirlenir. Başlangıç ve bitiş değerleri TO teriminden önce ve sonra yer almaktadır.

Artış değerini gösteren STEP teriminin kullanılması ise zorunlu değildir. STEP terimi hiç kullanılmadığı zaman, bilgisayar artış değerini +1 olarak kabul eder. Bunun dışındaki değerler için STEP terimini kullanarak artışı belirlemek zorunludur.

Bunların dışında, NEXT komutundan sonraki döngü değişkeninin de kullanılması zorunlu değildir.

**ÖRNEKLER:**

1) 1'den 10'a kadar sayıları ekrana yazan bir program:

```
10 FOR A = 1 TO 10
20 PRINT A
30 NEXT A
```

Görüldüğü gibi, bu örnekteki FOR satırında STEP terimi kullanılmamıştır, çünkü artışın 1 olması istenmektedir. Aynı sonuca ulaşabilmek için 10 numaralı satır 'FOR A = 1 TO 10 STEP 1' şeklinde de yazılabilir. Burada kullanılan STEP 1'in programa hiçbir etkisi olmadığı için, programcılık prensibine göre, hiç yazmamak daha doğrudur.

2) 10'dan 1'e kadar sayıları 1'er eksilterek ekrana yazan bir program:

```
10 FOR B = 10 TO 1 STEP -1
20 PRINT B
30 NEXT
```

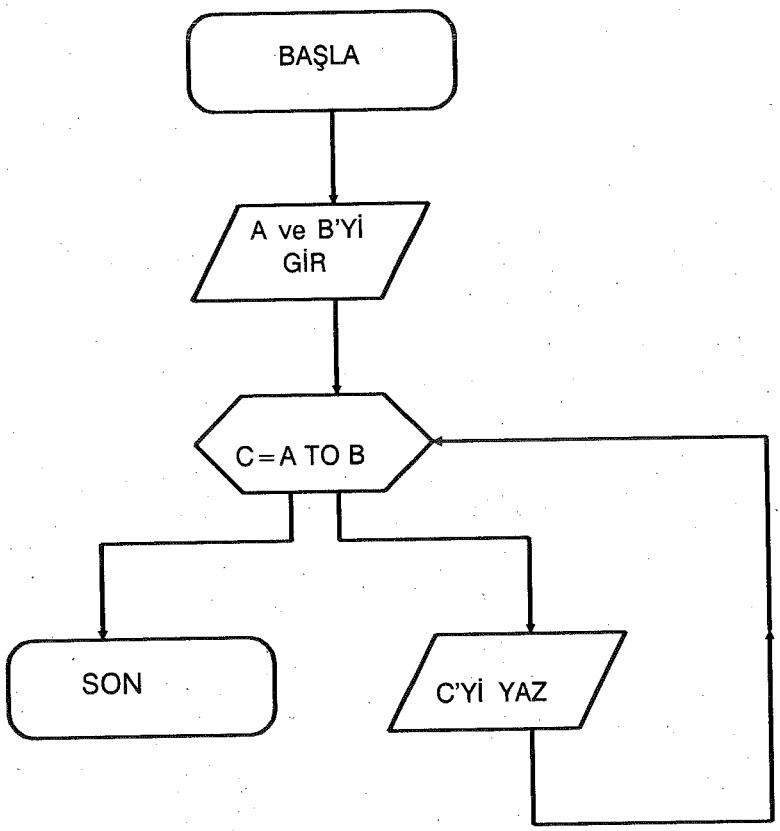
Bir öncekinin tam tersi olan bu örnekte, STEP teriminin kullanılması zorunludur. Bu terim kullanılmadığında artış +1 olarak kabul

edildiğinden, 10'dan 1'e kadar sayılar 1'er artarak yazılamayacağına göre, STEP -1 mutlaka kullanılmalıdır. Dikkat edilirse, bu örnekteki NEXT komutundan sonra döngü değişkeni kullanılmamıştır. Bunun nedeni, daha önce de belirtildiği gibi, kullanılmasına gerek olmamasıdır.

BASIC programlamada her zaman olduğu gibi, bir FOR cümlesindeki sabit sayılar yerine de aritmetiksel ifadeler veya değişkenler kullanılabilir.

**ÖRNEKLER:**

1) Klavyeden girilen iki sayı arasındaki sayıları listeleyen bir program:



```
10 INPUT A,B
20 FOR C = A TO B
30 PRINT C
40 NEXT C
```

INPUT komutundan sonra kullanılan değişkenler, FOR cümlesinde, döngünün başlangıç ve bitişini belirleyen değerler olarak kullanılmıştır.

2) Belirli değerleri ifade etmek için basit değişkenler yerine daha karmaşık aritmetik ifadeler de kullanılabilir:

```
10 READ A,B,C,D,E
20 FOR P = A * B TO C * E STEP D ^ 2
30 PRINT P * 3
40 NEXT P
50 DATA 1,4,15,2,3
```

Bu programdaki döngünün başlangıç, bitiş ve artış değerleri, aritmetiksel ifadelerin aldıkları değerlere göre belirlenir.

3) 1.5'ten 4.5'e kadar yarımşar artan sayıları listeleyen bir program:

```
10 FOR T = 1.5 TO 4.5 STEP 0.5
20 PRINT T
30 NEXT T
```

Görüldüğü gibi, döngü değişkeninin alacağı değerleri belirleyen sayılar tamsayı olmak zorunda değildir.

4) Klavyeden girilen 10 sayının karesini yazan program:

```
10 FOR K = 1 TO 10
20 INPUT A
30 PRINT A ^ 2
40 NEXT K
```

Bu örnekte, döngü değişkeni döngü içinde (FOR ve NEXT satırları arasında) hiç kullanılmamıştır. Bu gibi durumlarda FOR ... NEXT

döngüsünün tek amacı döngü içindeki komutların belirli sayıda tekrarını sağlamaktır. Eğer amaç yalnızca tekrar yapabilmek ise, döngünün başlangıç değeri 1, bitiş değeri de istenilen tekrar adedi olarak kullanılır. (Bu örnekte döngü 10 kez tekrarlanmaktadır.) Döngünün 10 kez tekrarlanabilmesi için başka olasılıklar da vardır. Örneğin '1 TO 10' yerine '2 TO 11' veya '25 TO 34' gibi sayılar da yazılabilir. Fakat, bu tür sayıları kullanmak çeşitli karışıklıklara yol açabileceğinden, başlangıç olarak kullanmak her zaman kolaylık sağlar.

5) 4 kişiye ait isim ve yaşları DATA satırından okuyarak ekrana listeleyen bir program:

```
10 FOR E = 1 TO 4
20 READ A$,Y
30 PRINT A$,Y
40 NEXT E
50 DATA Hasan,32,Necla,23,Can,41,Burak,12
```

Bu program çalıştırıldığında, aşağıdaki sonuç elde edilecektir:

```
RUN
Hasan        13
Necla        23
Can          41
Burak        12
```

6) Aynı isim listesinden yalnızca 35 yaşının altındaki kişileri ekrana listeleyen bir program:

```
10 FOR E = 1 TO 4
20 READ A$,Y
30 IF Y<35 THEN PRINT A$,Y
40 NEXT E
50 DATA Hasan,32,Necla,23,Can,41,Burak,12
```

Yalnızca PRINT satırının başına bir IF ... THEN ifadesi koyularak koşula uyan bilgilerin ekrana listelenmesi sağlanabilir. Diğer veriler DATA satırından okunmalarına karşın, koşula uymadıkları için listeye çıkmayacaklardır.

Bu program çalıştırıldığında, aşağıdaki sonuç elde edilecektir:

```
RUN
Hasan      32
Necla      23
Burak      12
```

7) DATA satırından okunan 5 sayının toplamını hesaplayarak ekrana yazan bir program:

```
10 FOR R = 1 TO 5
20 READ A
30 T = T + A
40 NEXT R
50 PRINT "TOPLAM =";T
60 DATA 6,8,3,5,2
```

Bu programda toplama işleminin yapılmasını sağlayan komut, 30 numaralı satırdaki 'T = T + A' komutudur. Bu komutun özelliği, daha önce sayaç kullanımında da görüldüğü gibi, aynı değişkenin eşit işaretinin iki tarafında da yer almasıdır. 'S = S + 1' komutunun S değişkeninin değerini 1 arttırması gibi, 'T = T + A' komutu da T değişkeninin değerini A sayısı kadar arttıracak, yani A'yı T'nin üzerine ekleyecektir. Bu komut FOR ... NEXT döngüsü içinde tekrarlandığında, T değişkeninin sonuç olarak vereceği değer bütün A'ların, yani DATA satırındaki bütün sayıların toplamı olacaktır.

Programın sonucu aşağıdaki gibi olacaktır:

```
RUN
TOPLAM = 24
```

8) Bir önceki örnekteki sayıların ortalamasını bulan bir program:

```
10 FOR R=1 TO 5
20 READ A
30 T = T + A
40 NEXT R
50 ORT = T / 5
60 PRINT "ORTALAMA =";ORT
70 DATA 6,8,3,5,2
```

Bu programda, bir öncekinden farklı olarak yalnızca ortalama eklenmiştir. 50 numaralı satırdaki 'ORT = T / 5' komutu, DATA'da 5 sayı olduğu için toplamın 5'e bölümüne eşit olan ortalamayı ORT değişkeninde saklamaktadır.

**NOT:** Bu komutun döngü dışında yazılmasına dikkat edilmelidir. Döngü içinde yazıldığında da aynı sonucu vermesine rağmen, bu durumda hem programı yavaşlatacak, hem de programlama mantığına uymayacaktır.

Programın sonucu şu şekilde olacaktır:

```
RUN
ORTALAMA = 4.8
```

9) DATA'dan okunan 8 sayıdan 15'ten büyük olanları listeleyen ve yalnızca bu sayıların toplamını hesaplayarak ekrana yazan bir program:

```
10 FOR K=1 TO 8
20 READ A
30 IF A>15 THEN PRINT A : T = T + A
40 NEXT K
50 PRINT "TOPLAM =";T
60 DATA 35,12,-3,0,67,19,8,16
```

Bu programda, IF ... THEN satırındaki PRINT ve toplama komutlarının arasında ':' işareti kullanılmıştır. Bunun nedeni, koşula uyan sayıların hem ekrana yazılması, hem de toplamlarının bulunmasının istenmesidir. Döngü içinde elde edilen bu toplam, döngü dışında ekrana yazdırılmıştır. Program çalıştırıldığında aşağıdaki sonuç çıkacaktır:

```
RUN
 35
 67
 19
 16
TOPLAM = 137
```

10) Klavyeden girilen 6 kesirli sayıyı tamsayıya yuvarlayarak yazan bir program:

```
10 FOR F=1 TO 6
20 INPUT A%
30 PRINT A%
40 NEXT F
```

Sayıları tamsayı olarak saklayabilmek için bu programda tamsayı değişkeni kullanılmıştır. Girilen sayılar kesirli dahi olsa, A% değişkeninin kullanılması sayesinde bu sayılar kendilerine en yakın tamsayıya yuvarlanarak ekrana yazılacaktır.

11) Olimpiyat Oyunları 4 yılda bir kez düzenlendiğine göre, 1988 ve 2020 arasında hangi yıllarda ve toplam kaç kez Olimpiyat yapılacağını listeleyen bir program:

```
10 FOR O=1988 TO 2020 STEP 4
20 PRINT O
30 S = S + 1
40 NEXT O
50 PRINT S;"KEZ OLİMPİYAT YAPILACAKTIR."

RUN
1988
1992
1996
2000
2004
2008
2012
2016
2020
9 KEZ OLİMPİYAT YAPILACAKTIR.
```

Burada başlangıç sayısı olarak 1988, bitiş sayısı olarak 2020 ve artış değeri olarak 4 verilmiştir. Bir sayaç kullanarak, ekrana yazılan yıllar da sayılmış ve program sonunda anlamlı bir cümle ile bu sayacın değeri ekrana yazılmıştır.

12) DATA satırından okunan 5 kişiye ait isim, yaş ve meslekleri uygun başlıklar altında ekrana listeleyen bir program:

```
10 PRINT "İSİM","YAŞ","MESLEK"
20 PRINT "-------","-----","---------"
30 FOR I=1 TO 5
40 READ A$,Y,M$
50 PRINT A$,Y,M$
60 NEXT I
70 DATA HASAN,34,MİMAR
80 DATA LEYLA ,42, AVUKAT
90 DATA HÜSNÜ,19,MARANGOZ
100 DATA ÇAĞATAY,27,MÜHENDİS
110 DATA FUNDA,29,KUAFÖR
```

```
RUN
İSİM        YAŞ        MESLEK
---------   ------     ----------
HASAN       34         MİMAR
LEYLA       42         AVUKAT
HÜSNÜ       19         MARANGOZ
ÇAĞATAY     27         MÜHENDİS
FUNDA       29         KUAFÖR
```

Başlıkların yazılması döngüden önce olmalıdır. Aksi takdirde, ekrana yazılan her satır için bir de başlık çıkacaktır. Bu programdaki DATA satırının bu şekilde gruplar halinde yazılması zorunlu değildir. Bu yolun, ortaya çıkabilecek yanlışlıkların düzeltilmesi açısından daha güvenli olmasına karşın, bütün bilgilerin tek bir DATA satırında yazılması daha kısa bir yoldur.

13) Bir şirkette çalışan kişi sayısı klavyeden girildikten sonra, her kişi için yine klavyeden girilecek maaş 75000 liradan az ise % 42, 75000 - 150000 lira arasında ise %40, 150000 liradan fazla ise % 38 oranında zam miktarını hesaplayarak bulunacak yeni maaşları ekrana yazan bir program:

```
10 INPUT "KAÇ KİŞİ ÇALIŞIYOR ";K
20 FOR G = 1 TO K
```

```
30 INPUT "MAAŞ ";M
40 IF M<75000 THEN Z = M * 0.42
50 IF M>= 75000 AND M<=150000 THEN Z = M * 0.40
60 IF M>150000 THEN Z = M * 0.38
70 PRINT "ZAMLI MAAŞ =";M + Z
80 NEXT G
```

Bu programdaki döngü, en başta girilen işçi sayısına göre açılmıştır (1 TO K).

**SORULAR:**

1) 8 ve 24 arasındaki tek sayıları listeleyen ve toplamlarını yazan bir program yapınız.

2) Aşağıdaki bilgilere göre, 100000 ve 200000 TL arasında maaş alanları listeleyen bir program yapınız.

| İSİM | MAAŞ |
|---|---|
| Hüseyin | 172000 |
| Bülent | 56000 |
| Enver | 210000 |
| Selami | 110000 |
| Kaya | 100000 |

3) 5'ten 25'e kadar sayıların toplamını bulup ekrana yazan bir program yapınız.

4) Klavyeden girilen kelimeyi, arada birer satır boş bırakarak 7 kez alt alta ekrana yazan bir program yapınız.

**LOCATE komutu:**

Kursörün (gösterici) ekranın istenilen herhangi bir yerine taşınarak bir bilginin ekrana yazılması veya klavyeden okunması için kullanılan komuttur.

Genel kullanım:

**LOCATE <satır>, <sütun>**

Ekranın sol üst köşesi başlangıç olarak kabul edilerek, LOCATE komutundan sonra kullanılan sayılar, kursörün kaç karakter aşağı (kaçıncı satıra), kaç karakter sağa (kaçıncı sütuna) taşınacağını belirler. LOCATE komutundan sonra genellikle ekranla ilgili komutlardan biri (PRINT, INPUT) kullanılır.

NOT: Dünyada çok yaygın olarak kullanılan IBM PC ve IBM PC uyumlu bilgisayarların dışında bazı BASIC yorumlayıcılarında, LOCATE komutunun yazılımı farklıdır. LOCATE teriminden sonraki ilk sayı sütunu, ikinci sayı ise satırı ifade eder.

**ÖRNEKLER:**

1) 'BELLEK' kelimesini ekranın 7'nci sütunu ve 13'üncü satırından başlayarak yazan bir program:

```
10 LOCATE 13,7
20 PRINT "BELLEK"
```

2) Bilgi girişini ekranın 2'nci sütunu ve 5'inci satırından başlayarak yapan ve klavyeden okunan iki sayının çarpımını boş bir ekrana yazan bir program:

```
10 LOCATE 5,2
20 INPUT A,B
30 CLS
40 PRINT A * B
```

Bu programda da görüldüğü gibi, LOCATE komutu INPUT komutuyla girilecek bilgilerin ekranın neresinden girileceğini belirlemek için de kullanılabilir.

# KARAKTER FONKSİYONLARI:

**LEN fonksiyonu:**

Herhangi bir alfabetik bilginin karakter sayısını bulmak için kullanılır. Karakterler arasındaki boşluklar da sayıma girer.

Genel kullanım:

**LEN <(alfabetik bilgi)>**

**NOT:** Buradaki alfabetik bilgi, tırnak içinde yazılmış bir karakter dizisi veya bir alfabetik değişken olabilir.

**ÖRNEKLER:**

1) 'PENCERE' kelimesinin uzunluğunu (karakter sayısını) yazan bir program:

```
A - 10 A$="PENCERE"
    20 PRINT LEN(A$)

B - 10 PRINT LEN("PENCERE")
```

Görüldüğü gibi, aynı sonuca ulaşmak için iki yol kullanılabilir. Birinci yolda 'PENCERE' kelimesi bir değişkende saklanıp, LEN fonksiyonuyla birlikte bu değişken kullanılabilir. Daha kısa olan ikinci yolda ise kelime doğrudan doğruya fonksiyonla birlikte kullanılabilir. Bunu yaparken tırnak işaretlerini unutmamaya dikkat edilmelidir.

2) 'Bilgisayarlar, çok hızlı çalışırlar.' cümlesinin uzunluğunu yazan bir program:

```
10 PRINT LEN("Bilgisayarlar, çok hızlı çalışırlar.")

RUN
 36
```

Dikkat edilirse, program çalıştırıldığında elde edilen sonuca, cümle içindeki boşluklar ve noktalama işaretleri de dahildir.

3) DATA satırından okunan 6 isim ve bu isimlerin uzunluklarını uygun başlıklar altında listeleyen bir program:

```
10 PRINT "İSİM","HARF SAYISI"
20 PRINT "------","----------"
30 FOR H = 1 TO 6
40 READ A$
50 PRINT A$,LEN(A$)
60 NEXT H
70 DATA BELMA, GÜLAY, OKAN, CEM, MUSTAFA,
HÜSAMETTİN

RUN
İSİM          HARF SAYISI
---------     -------------
BELMA              5
GÜLAY              5
OKAN               4
CEM                3
MUSTAFA            7
HÜSAMETTİN        10
```

**LEFT$, RIGHT$ ve MID$ Fonksiyonları:**

Alfabetik bilgilerin program içerisinde bütün olarak değil de, belirli parçalar halinde işlenmesi istendiğinde, bu fonksiyonlar kullanılır.

**LEFT$:**

Genel kullanım:

**LEFT$ <(alfabetik bilgi,karakter sayısı)>**

LEFT$ fonksiyonu, bir alfabetik bilginin sol tarafından (başından) başlayarak istenilen karakter sayısı kadar bir bölümünün ayrılması için kullanılır.

**ÖRNEKLER:**

1) 'Bilgisayar' kelimesinin baştan 5 harfi şu şekilde yazdırılabilir:

```
PRINT LEFT$("Bilgisayar",5)
Bilgi
```

2) 'Türkiye' kelimesi bir değişkene atanarak, 'Türk' bölümü ekrana şu şekilde yazdırılabilir:

```
D$="Türkiye"
PRINT LEFT$(D$,4)
Türk
```

**NOT:** Hiçbir zaman unutulmamalıdır ki, yukarıdaki örneklerde de görüldüğü gibi, BASIC programlamada herhangi bir karakter dizisinin (tırnak işaretleri arasına alınmış bilgi) yerini bir alfabetik değişken alabilir. Klavyeden bilgi girişi, READ komutu ve DATA satırına yazılan kelimeler bu kurala dahil değildir: INPUT "AHMET" , DATA A$,B$ gibi satırlar veya INPUT komutu karşılığı olarak klavyeden bilgi girişi sırasında A$, A gibi değişkenler kullanılamaz.

**RIGHT$**

Genel kullanım:

**RIGHT$ <(alfabetik bilgi,karakter sayısı)>**

RIGHT$ fonksiyonu, bir alfabetik bilginin sağ tarafından (sonundan) başlayarak istenilen karakter sayısı kadar bir bölümünün ayrılması için kullanılır.

**ÖRNEKLER:**

1) 'Bilgisayar' kelimesinin sondan 4 harfi şu şekilde yazdırılabilir:

```
PRINT RIGHT$("Bilgisayar",4)
ayar
```

2) 'Türkiye' kelimesi bir değişkene atanarak, 'iye' bölümü ekrana şu şekilde yazdırılabilir:

```
D$="Türkiye"
PRINT RIGHT$(D$,3)
iye
```

**MID$:**

Genel kullanım:

**MID$ (alfabetik bilgi,başlangıç pozisyonu [,karakter sayısı] )**

MID$ fonksiyonu, bir alfabetik bilginin herhangi bir karakterinden başlayarak, istenilen sayıda karakterinin veya kelimenin geri kalan bölümünün ayrılması için kullanılır.

**ÖRNEKLER:**

1) 'Bilgisayar' kelimesinin 6'ncı harfinden başlayarak 3 harflik bölümü ekrana şu şekilde yazdırılabilir:

```
PRINT MID$("Bilgisayar",6,3)
say
```

2) 'Türkiye' kelimesi bir değişkene atanarak, 'ürk' bölümü ekrana şu şekilde yazılabilir:

```
D$="Türkiye"
PRINT MID$(D$,2,3)
ürk
```

3) 'Kahramanmaraş' kelimesinin 9'uncu harfinden başlayarak geri kalan bölümü şu şekilde yazdırılabilir:

```
PRINT MID$("Kahramanmaraş",9)
maraş
```

**Karakter Fonksiyonlarının Kullanımlarıyla İlgili Örnekler:**

1) 'AFYONKARAHİSAR' kelimesinden yararlanarak 'AFYON', 'KARA' ve 'HİSAR' kelimelerini alt alta yazan bir program:

```
10 A$="AFYONKARAHİSAR"
20 PRINT LEFT$(A$,5)
30 PRINT MID$(A$,6,4)
40 PRINT RIGHT$(A$,5)

RUN
AFYON
KARA
HİSAR
```

2)

```
D
DI
DIS
DISK
DISKE
DISKET
DISKETT
DISKETTE
```

Yukarıdaki görüntüyü ekrana çıkaran bir program:

```
10 A$="DISKETTE"
20 PRINT LEFT$(A$,1)
30 PRINT LEFT$(A$,2)
40 PRINT LEFT$(A$,3)
50 PRINT LEFT$(A$,4)
60 PRINT LEFT$(A$,5)
70 PRINT LEFT$(A$,6)
80 PRINT LEFT$(A$,7)
90 PRINT LEFT$(A$,8)
```

Dikkat edilirse, 20 numaralı satırdan 90 numaralı satıra kadar bütün komutlar, karakter sayıları dışında tamamen aynıdır. Bu sayıların

yerine bir döngü değişkeni kullanarak, bu satırların yerine tek bir satır yazmak mümkündür. Bu döngü değişkeni, 1'den başlayarak, kelimenin karakter uzunluğuna kadar 1'er 1'er artmalıdır. Bu yolla, program büyük ölçüde kısaltılmış ve basitleştirilmiş olacaktır:

```
10 A$="DISKETTE"
20 L = LEN(A$)
30 FOR M = 1 TO L
40 PRINT LEFT$(A$,M)
50 NEXT M
```

3) Klavyeden girilen bir kelimenin harflerini alt alta yazan bir program:

```
10 INPUT A$
20 UZ = LEN(A$)
30 FOR X = 1 TO UZ
40 PRINT MID$(A$,X,1)
50 NEXT X

RUN
? BİLGİSAYAR
B
İ
L
G
İ
S
A
Y
A
R
```

Görüldüğü gibi, FOR ... NEXT döngüsü kullanılarak bu program kolayca yapılabilir. 40 numaralı satırda MID$ fonksiyonuyla birlikte kullanılan X (döngü değişkeni), kaçıncı harften başlanacağını, daha sonraki 1 sayısı ise her satırda kaç harf yazılacağını gösterir. Buna göre, her keresinde X'inci harf yazılacak ve kelimenin bütün harfleri alt alta listelenecektir.

4) Klavyeden girilen kelimenin harflerini yan yana, her harften sonra bir süre bekleyerek ekrana yazan bir program:

```
10 INPUT A$
20 FOR T = 1 TO LEN(A$)
30 PRINT MID$(A$,T,1);
40 FOR K = 1 TO 1000 : NEXT K
50 NEXT T

RUN
?PROGRAMLAMA
PROGRAMLAMA
```

(Her harf yazıldıktan sonra bir süre beklenecektir.)

Bu programda, girilen kelimenin harfleri, 30 numaralı satırın sonundaki ( ; ) işareti sayesinde yan yana yazılacaktır. 40 numaralı satırdaki içinde hiçbir komut bulunmadan açılıp kapanan döngü, her harf yazıldıktan sonra bilgisayarın bir süre beklemesini (1'den 1000'e kadar saymasını) sağlayacaktır. Ayrıca, T döngüsünün bitiş değeri olarak, girilen kelimenin uzunluğu yerine geçecek LEN fonksiyonu kullanılmış, bu değer daha önceden bir değişkende saklanmadan FOR satırına yazılmıştır.

5) DATA satırından okunan 4 kelimeden, 3'üncü harfi ve son harfi eşit olanları listeleyen bir program:

```
10 FOR X = 1 TO 4
20 READ A$
30 IF MID$(A$,3,1) = RIGHT$(A$,1) THEN PRINT A$
40 NEXT X
50 DATA BENAN,OKTAY,SERDAR,HARUN

RUN
BENAN
SERDAR
```

6) DATA satırındaki 8 kelimeden 'A' ile başlayanları listeleyen bir program:

```
10 FOR C = 1 TO 8
20 READ A$
```

```
30 IF LEFT$(A$,1) = "A" THEN PRINT A$
40 NEXT C
50 DATA SİNAN, ADNAN, ALEV, GÜLTEN, AYLA,
SUNA,ZEYNEP,HURİ

RUN
ADNAN
ALEV
AYLA
```

Bu programda, 30 numaralı satırdaki 'A' harfinin tırnak işaretleri arasında yazılmasına dikkat edilmelidir. Eğer tırnak işaretleri unutulursa, bilgisayar bu A'yı sayısal bir değişken olarak değerlendirecek, alfabetik bir ifadeyle (LEFT$) bir sayı karşılaştırılamayacağından, bir hata mesajı ile program duracaktır.

7) Harf sayısı 5'ten fazla olan 6 şehir adından 'R' ile bitenleri listeleyen bir program:

```
10 FOR A = 1 TO 6
20 READ S$
30 IF LEN(S$)>5 AND RIGHT$(S$,1)="R" THEN
PRINT A$
40 NEXT A
50 DATA İZMİR, İSTANBUL, BURDUR, BOLU,
ESKİŞEHİR,NİĞDE

RUN
BURDUR
ESKİŞEHİR
```

8) Klavyeden girilen 5 isimden kaç tanesinin 4'üncü harfinin 'E' olduğunu bulan bir program:

```
10 FOR B = 1 TO 5
20 INPUT P$
30 IF MID$(P$,4,1)="E" THEN S = S + 1
40 NEXT B
50 PRINT "BU İSİMLERDEN";S;"TANESİNİN 4'ÜNCÜ
HARFİ E'DİR."
```

```
RUN
? KEMAL
? SANEM
? LALE
? KAYA
? SİBEL
BU İSİMLERDEN 3 TANESİNİN 4'ÜNCÜ HARFİ E'DİR.
```

9) 'SONRA' ve 'İLKBAHAR' kelimelerinden 'SONBAHAR' kelimesini türeten ve ekrana yazan bir program:

```
10 A$="SONRA"
20 B$="ILKBAHAR"
30 C$=LEFT$(A$,3) + RIGHT$(B$,5)
40 PRINT C$

RUN
SONBAHAR
```

**SORULAR:**

1) Klavyeden girilen iki kelimeden ikisinin de uzunlukları 4 harften fazlaysa sona eren, değilse başa dönen bir program yapınız.

2) Klavyeden girilen iki kelimeden birincisinin son harfi ikincisinin ilk harfine eşitse ekranı silip başa dönen, aksi durumda 'STOP' yazıp duran bir program yapınız.

3) Klavyeden girilen kelimenin ikinci harfi 'T' ise veya ilk harfi son harfine eşitse ekranı silen, değilse kelimenin uzunluğunu yazan bir program yapınız.

4) Klavyeden girilen kelimeyi tersten yazan bir program yapınız.

# DİZİLER:

Bilindiği gibi, bir değişken, herhangi bir zamanda yalnızca bir tek değer içerebilir. Diğer bir deyişle, bir değişkende aynı anda birden çok bilgi saklamak olanaksızdır. Diğer birçok programlama dilinde de olduğu gibi, BASIC'te de bir tür değişken vardır ki, bu tür değişkenin içinde aynı anda benzer cinsten birçok bilgi saklanabilir. Bellekte birden fazla hücreyi kaplayabilen bu tür değişkenler, indisli değişkenler ( diziler ) olarak tanımlanmaktadır.

**DIM komutu:**

Genel kullanım:

**DIM <değişken-1(indis-1,indis-2,...)>[,<değişken-2(indis-1, indis-2,...)>,...]**

Programlamada indisli değişkenler kullanılırken, bu değişkenlerin maksimum eleman sayısı kadar bellekte yer ayrılmalıdır. İngilizce'deki dimension (boyut) kelimesinden türemiş olan DIM komutu, dizilerin kaç boyutlu olduklarını ve kaçar elemandan oluştuklarını tanımlamak amacıyla kullanılır. Diziler 1 boyutlu olabildikleri gibi, kullanılan bilgisayarın kapasitesine göre, daha fazla (2, 3, 4,...) boyutlu olabilirler. Bu bölümde, bir boyutlu diziler (vektörler) incelenecektir.

**İndisli Değişkenlere Ait Kurallar:**

1) İndisler en az bir tane olur; maksimum indis sayısı bilgisayarın kapasitesine göre değişir.
2) İndisler tamsayı sabit, tamsayı değişken veya aritmetik ifade olabilirler ve birbirlerinden virgülle ayrılırlar.
3) İndisler gerçel büyüklükler olarak verilirlerse sonuçta tamsayıya dönüştürülürler.
4) İndisler negatif olamazlar.
5) İndisler bir başka indisli değişken olabilirler.

NOT: Birçok BASIC yorumlayıcı, en fazla 10 elemana kadar dizilerin tanımlanması için DIM komutunun kullanılmasını gerektirmez.

**ÖRNEKLER:**

1) 12 elemanlı ve tek boyutlu bir A dizisini tanımlamak için şu komut yazılmalıdır:

```
DIM A(12)
```

2) 15 elemanlı M ve 18 elemanlı N dizilerini tanımlamak için şu komut yazılmalıdır:

```
DIM M(15),N(18)
```

veya

```
DIM M(15)
DIM N(18)
```

şeklinde de yazılabilir.

3) 10 elemanlı bir T$ dizisini tanımlamak için şu komut yazılmalıdır:

**DIM T$(10)** (Eleman sayısı 10'u geçmediği için, birçok bilgisayarda bu satıra gerek yoktur.)

4) 5 elemanlı F dizisinin 3'üncü elemanını 7'ye eşitlemek için şu komut yazılabilir:

**LET F(3) = 7** (LET kullanılmadanda aynı satır yazılabilir.)

**Dizilerin Kullanım Amaçları:**

Daha önce de belirtildiği gibi, diziler, aynı türden birçok bilgiyi aynı anda tek bir isim altında bellekte saklayabilmek için kullanılırlar. Bu bilgilerden herhangi birine ulaşabilmek için, o bilginin indisini (dizinin kaçıncı elemanı olduğunu) belirtmek yeterlidir. Örneğin, bir grup insanın ismi, bir ülkedeki şehir adları, telefon numaraları veya bir deney sonucunda elde edilen bilgiler dizilere yerleştirilerek, bu bilgilerden herhangi birine istenildiği anda ulaşılabilir.

**ÖRNEKLER:**

1) DATA satırındaki 6 sayıyı bir diziye yerleştiren ve dizi elemanlarını boş bir ekrana listeleyen bir program:

```
10 DIM A(6)
20 READ A(1)
30 READ A(2)
40 READ A(3)
50 READ A(4)
60 READ A(5)
70 READ A(6)
80 DATA 5,2,8,6,4,5
90 CLS
100 PRINT A(1)
110 PRINT A(2)
```

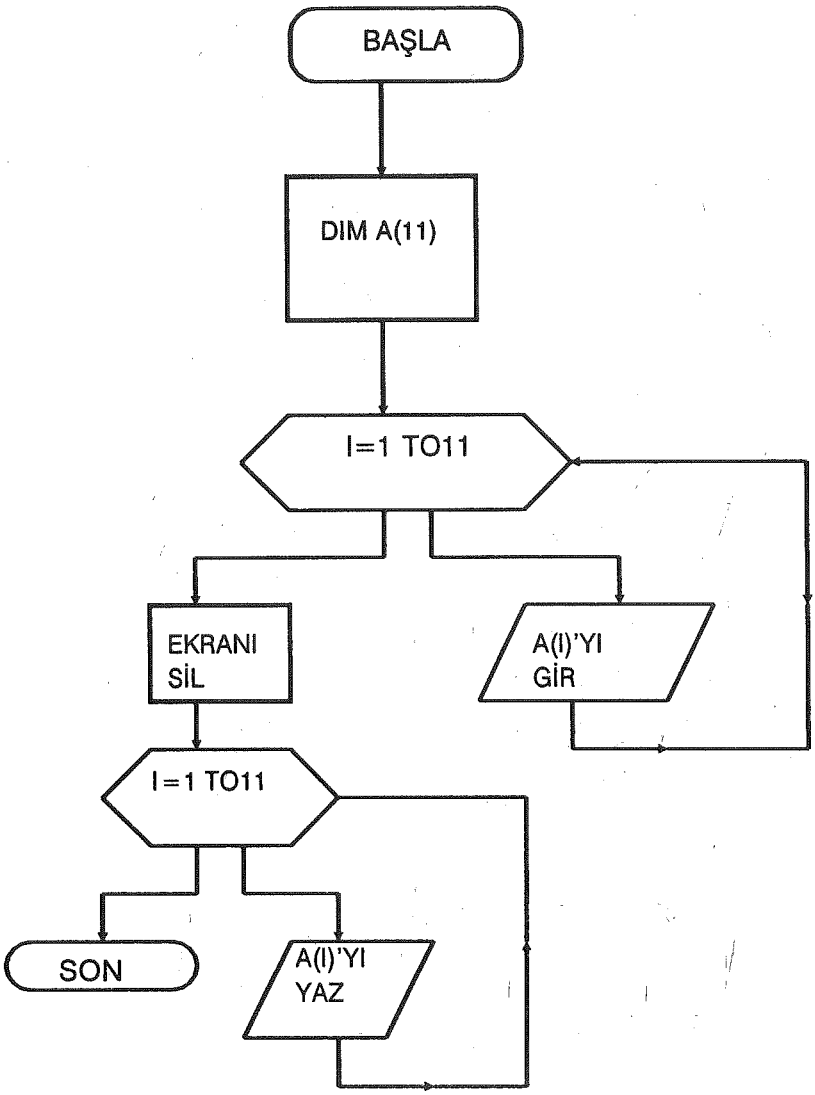
```
120 PRINT A(3)
130 PRINT A(4)
140 PRINT A(5)
150 PRINT A(6)
```

Dizinin bütün elemanlarını birer birer DATA satırından okuyup, daha sonra hepsini ekrana listeleyen bu programda, 1'den 6'ya kadar 1'er artan dizi indisi yerine bir döngü değişkeni kullanılabilir. Bu şekilde program büyük ölçüde kısalacak ve basitleşecektir:

```
10 DIM A(6)
20 FOR I = 1 TO 6
30 READ A(I)
40 NEXT I
50 DATA 5,2,8,6,4,5
60 CLS
70 FOR K = 1 TO 6
80 PRINT A(K)
90 NEXT K
```

Bilgileri DATA satırından okuyan bu programın, klavyeden girilecek bilgilerle çalışması için READ komutunun yerine INPUT kullanılabilir.

2) Klavyeden girilen 11 sayıyı boş bir ekrana listeleyen bir program:




```
10 DIM A(11)
20 FOR I = 1 TO 11
30 INPUT A(I)
40 NEXT I
50 CLS
60 FOR I = 1 TO 11
70 PRINT A(I)
80 NEXT I
```

Dizi kullanılmadığında, klavyeden girilen bilgilerin hepsinin aynı anda bellekte tutularak daha sonra ekrana yazılması, her eleman için ayrı bir değişken kullanılmasıyla mümkün olacaktır. Özellikle eleman sayısı çok olduğunda, bu yol çok zahmetli olacak ve programcılık anlayışına ters düşecektir. Bu gibi durumlarda, dizi kullanılması kaçınılmazdır.

Yukarıdaki programda, A(I), döngü değişkeni olan I'nın aldığı değerlere göre (1 TO 11), 11 elemanlı A dizisinin eleman değerlerini ifade etmektedir. Başta A(1), daha sonra A(2), A(3), A(4) ... A(10), A(11) anlamına gelecek bu ifade, önce klavyeden girilen sayıların dizide saklanması, daha sonra bu dizi elemanlarının ekrana listelenmesi için kullanılmıştır. Sayıların boş bir ekrana listelenmesi istendiğinden, iki döngü arasında CLS komutu vardır.

3) 12 kişinin ismini DATA satırından okuyarak bir diziye yerleştiren ve ekrana listeleyen bir program:

```
10 DIM A$(12)
20 FOR W = 1 TO 12
30 READ A$(W)
40 NEXT W
50 DATA CAN, CEM, AYŞE, GÜL,MİNE, YASEMİN,
SEMA, NİHAL, ÇİĞDEM, GİZEM,ANI,NURFER
60 FOR Q = 1 TO 12
70 PRINT A$(Q)
80 NEXT Q
```

4) 5 öğrenciye ait DATA satırından okunan numara, isim ve adresleri dizilere yerleştirdikten sonra numarası 100'den büyük olan öğrencilere ait bilgileri boş bir ekrana uygun başlıklar altında listeleyen bir program:

```
10 DIM NO(5),I$(5),ADR$(5)
20 FOR R = 1 TO 5
30 READ NO(R),I$(R),ADR$(R)
40 NEXT R
50 DATA 122,YEŞİM, FENERBAHÇE, 84, MERT,
BOSTANCI, 92, GÜL, AKSARAY, 196, BORA,
ERENKÖY, 101, ŞEBNEM, ETİLER
60 CLS
```

```
70 PRINT "NUMARA","İSİM","ADRES"
80 PRINT "------","---------","----------"
90 FOR N = 1 TO 5
100 IF NO(N)>100 THEN PRINT NO(N),I$(N),
ADR$(N)
110 NEXT N
```

Aynı program, dizi kullanmadan aşağıdaki şekilde yapılabilirdi:

```
10 CLS
20 PRINT "NUMARA","İSİM","ADRES"
30 PRINT "------","---------","----------"
40 FOR N = 1 TO 5
50 READ NO,I$,ADR$
60 IF NO>100 THEN PRINT NO, I$, ADR$
70 NEXT N
80 DATA 122, YEŞİM, FENERBAHÇE, 84, MERT,
BOSTANCI, 92, GÜL, AKSARAY, 196, BORA,
ERENKÖY, 101, ŞEBNEM, ETİLER
```

Ekrandaki görüntü açısından, iki program da tamamen aynıdır. İkisi de çalıştırıldıklarında aynı listeyi çıkaracaktır. Dizi kullanılan birinci programın ikinciden farkı, DATA satırından okunan bilgilerin hepsinin ayrı ayrı bilgisayarın belleğinde saklanmasıdır. Dolayısıyla, birinci program çalıştırıldıktan sonra, istenilen öğrenciye ait bilgiler verilecek uygun bir komutla ekranda görülebilecektir:

```
PRINT NO(3),I$(3),ADR$(3)
```

komutuyla, dizideki üçüncü öğrenciyle ilgili bilgiler öğrenilebilecektir.

Dizi kullanılmadan yapılmış olan ikinci program çalıştırıldıktan sonra, PRINT NO,I$,ADR$ komutu, sonuç olarak DATA satırından okunmuş olan en son bilgiyi verecek, daha önceki bilgiler ekranda görünmekte olmalarına karşın bellekten tamamen silinmiş olacaktır.

**NOT:** Dizi kullanımıyla ilgili olan yukarıdaki programlar, yalnızca tek bir döngü ile yapılabilirler. Adımların daha açık bir şekilde görülebilmesi için bu programlarda DATA satırından okuma için bir, ekrana yazma için bir olmak üzere ikişer döngü kullanılmıştır.

5) 11 öğrencinin isim ve notlarını birer diziye yerleştirerek, not ortalamasını hesapladıktan sonra ortalamanın üzerinde not alan öğrencilerin isimlerini listeleyen bir program:

```
10 DIM S$(11),N(11)
20 FOR T = 1 TO 11
30 READ S$(T),N(T)
40 TOP = TOP + N(T)
50 NEXT T
60 DATA CAN, 8, CEM, 6, AYŞE, 4, GÜL, 7, MİNE, 9,
YASEMİN, 3,
SEMA, 5, NİHAL, 6, ÇİĞDEM, 5, GİZEM, 8, ANI, 9
70 ORT = TOP / 11
80 FOR G = 1 TO 11
90 IF N(G)>ORT THEN PRINT S$(G)
100 NEXT G
```

Bu programda, birinci döngüde DATA satırındaki bilgiler okunmuş ve notların toplamı bulunmuştur. Ancak bundan sonra ortalama hesaplanarak, ikinci döngüde ortalamanın üzerinde not alanlar listelenebilmiştir. Aynı işlemin tek döngüde yapılabilmesine olanak yoktur.

Aynı program, dizi kullanılmadan aşağıdaki şekilde yapılabilir:

```
10 FOR T = 1 TO 11
20 READ S$,N
30 TOP = TOP + N
40 NEXT T
50 DATA CAN, 8, CEM, 6, AYŞE, 4, GÜL, 7, MİNE, 9,
YASEMİN, 3, SEMA, 5, NİHAL, 6, ÇİĞDEM, 5,
GİZEM, 8, ANI, 9
```

```
60 ORT = TOP / 11
70 RESTORE
80 FOR G = 1 TO 11
90 READ S$,N
100 IF N>ORT THEN PRINT S$
110 NEXT G
```

Bu ikinci yolda, RESTORE komutu kullanılarak, aynı DATA satırındaki bilgiler ikinci bir FOR ... NEXT döngüsü içinde okunmuş ve dizi kullanımına ihtiyaç kalmamıştır. Ancak, bu programın çalışma hızı birinciden daha düşüktür.

6) 6'şar elemanlı A ve B dizilerinin karşılıklı elemanlarının çarpımlarını C dizisine atayarak, üç diziyi de ekrana listeleyen bir program:

```
10 DIM A(6),B(6),C(6)
20 FOR F = 1 TO 6
30 READ A(F),B(F)
40 C(F)=A(F) * B(F)
50 PRINT A(F),B(F),C(F)
60 NEXT F
70 DATA 2,4,3,5,7,5,9,7,8,6,4,2
```

Bu programdaki 30, 40 ve 50 numaralı satırların her biri, farklı birer döngü içinde kullanılabilir. Programı daha kısa bir yoldan yapabilmek için, bu örnekte hepsi tek bir döngü içine yazılmıştır.

7) 11 öğrencinin notlarını ve isimlerini iki ayrı diziye yerleştirerek, kaç kişinin 6 ve daha yukarı not aldığını yazan bir program:

```
10 DIM S$(11),N(11)
20 FOR T = 1 TO 11
30 READ S$(T),N(T)
40 IF N(T)>= 6 THEN S = S + 1
50 NEXT T
60 DATA CAN, 8, CEM, 6, AYŞE,4, GÜL, 7, MİNE, 9,
YASEMİN, 3,SEMA, 5, NİHAL, 6, ÇİĞDEM, 5,
GİZEM, 8, ANI, 9
70 PRINT S;"kişi 6 ve daha yukarı not almıştır."
```

Bu programda, 'KAÇ' sorusuna cevap verebilmek için sayaç kullanılmalıdır. 40 numaralı satırdaki 'S = S + 1' komutu, 6 ve daha yukarı not alan kişiler için uygulanacaktır. Daha önce değer verilmemiş olduğu için başta sıfıra eşit olan sayacın değeri, döngü dışına çıkıldığında, koşula uyan bilgilerin sayısını verecektir. Dikkat edilirse, bu program çalıştırıldığında ekrana hiçbir liste çıkmayacak, yalnızca 70 numaralı satırdaki PRINT komutunun sonucu olan sayaç değeri ekrana yazılacaktır. Ayrıca, DATA satırındaki isimler bir dizide saklanmalarına rağmen kullanılmamıştır.

8) Bir önceki örnekteki sonucu veren, aynı zamanda koşula uyan kişilerin isimlerini listeleyen bir program:

```
10 DIM S$(11),N(11)
20 FOR T = 1 TO 11
30 READ S$(T),N(T)
40 IF N(T)>= 6 THEN S = S + 1 : PRINT S$(T)
50 NEXT T
60 DATA CAN, 8, CEM, 6, AYŞE, 4, GÜL, 7, MİNE, 9,
YASEMİN, 3, SEMA, 5, NİHAL, 6, ÇİĞDEM, 5,
GİZEM, 8, ANI, 9
70 PRINT S;"kişi 6 ve daha yukarı not almıştır."
```

Bu kez koşula uyan bilgilerin listelenmesi de istendiğinden, IF ... THEN cümlesinde 'S = S + 1' komutunun dışında bir de PRINT komutu kullanılmıştır. İki komut arka arkaya verilmiş olduğundan, aralarında ( : ) bulunmalıdır.

9) Bir önceki örneği kullanarak, bu kez koşula uyan kişilerin isimlerini önce başka bir diziye aktaran, daha sonra bu diziyi ekrana listeleyen bir program:

```
10 DIM S$(11),F$(11),N(11)
20 FOR T = 1 TO 11
30 READ S$(T),N(T)
40 IF N(T)>= 6 THEN F$(T) = S$(T)
50 NEXT T
60 DATA CAN, 8, CEM, 6, AYŞE, 4, GÜL, 7, MİNE, 9,
YASEMİN, 3, SEMA, 5, NİHAL, 6, ÇİĞDEM, 5,
GİZEM, 8, ANI, 9
```

```
70 FOR Y = 1 TO 11
80 PRINT F$(Y)
90 NEXT Y
```

| S$ dizisi | SAYAÇ | F$ dizisi |
|---|---|---|
| CAN | 1 | CAN |
| CEM | 2 | CEM |
| AYŞE | 3 | |
| GÜL | 4 | GÜL |
| MİNE | 5 | MİNE |
| YASEMİN | 6 | |
| SEMA | 7 | |
| NİHAL | 8 | NİHAL |
| ÇİĞDEM | 9 | |
| GİZEM | 10 | GİZEM |
| ANI | 11 | ANI |

Görüldüğü gibi elemanların ekrana çıkmasına karşın, koşula uymayan her eleman için birer satır boşluk oluşmuştur. Bunun nedeni, dizi oluşturulurken, koşula uymayan elemanların aktarılmaması, onların yerinde daha önceden varolan boşlukların ekrana çıkmasıdır. Burada dikkat edilmesi gereken bir başka nokta da, yeni oluşturulacak dizinin, dizilerin tanımlandığı DIM satırında yer alması gerektiğidir.

10) Bir önceki örneği kullanarak, bu kez koşula uyan kişilerin isimlerini ikinci diziye başından başlayıp sıralı olarak (arada boşluk bırakmadan) aktaran ve daha sonra bu diziyi ekrana listeleyen bir program

```
10 DIM S$(11),F$(11),N(11)
20 FOR T = 1 TO 11
30 READ S$(T),N(T)
40 IF N(T)>= 6 THEN  S = S + 1 : F$(S) = S$(T)
50 NEXT T
60 DATA CAN,8,CEM,6,AYŞE,4,GÜL,7,MİNE,9,
YASEMİN,3,SEMA,5,NİHAL,6,ÇİĞDEM,5,GİZEM,8,ANI,9
70 FOR Y = 1 TO S
80 PRINT F$(Y)
90 NEXT Y
```

| S$ dizisi | SAYAÇ | F$ dizisi |
|---|---|---|
| CAN | 1 | CAN |
| CEM | 2 | CEM |
| AYŞE | 3 | GÜL |
| GÜL | 4 | MİNE |
| MİNE | 5 | NİHAL |
| YASEMİN | 6 | GİZEM |
| SEMA | 7 | ANI |
| NİHAL | 8 | |
| ÇİĞDEM | 9 | |
| GİZEM | 10 | |
| ANI | 11 | |

Bu kez dizi sıralı olarak oluşturulduğu için, boşluklar kelimelerin aralarında değil, dizinin sonunda kalmıştır. Diziler arasında bu türlü bilgi aktarma yöntemine sayaçlı (sıralı) atama denir. 40 numaralı satırda görüldüğü gibi, sayacın değeri koşula uyan elemanlar için arttırıldığından, yeni dizi sıralı olarak meydana getirilir. Atama işlemini gerçekleştiren 'F$(S) = S$(T)' komutunda, yeni oluşturulan dizinin indisi sayaç, diğerininki ise döngü değişkenidir.

Yeni oluşturulan dizideki isimleri listelemek için, 70 ve 90 numaralı satırlar arasındaki döngü kullanılmıştır. Bir önceki örnekte '1 TO 11' şeklinde açılmış olan bu döngü, bu kez '1 TO S' olarak açılmıştır. Bunun nedeni, sayaçlı atama kullanıldığında, yeni diziye atama yoluyla aktarılmış bilgi adedi, sayaç değişkeninde (S) saklanmaktadır.

11) 15 elemanlı A dizisindeki sayılardan negatif olanları N, pozitif olanları P dizisine aktarıp, daha sonra bu iki diziyi listeleyen program:

```
10 DIM A(15),P(15),N(15)
20 FOR X = 1 TO 15
30 READ A(X)
40 IF A(X)>0 THEN S1 = S1 + 1 : P(S1) = A(X)
50 IF A(X< 0 THEN S2 = S2 + 1 : N(S2) = A(X)
60 NEXT X
70 DATA 23,4,-5,6,12,-45,3,-78,3,98,-123,42,-45,4,6
```

```
 80 PRINT "Pozitif sayılar:"
 90 FOR X = 1 TO S1
100 PRINT P(X)
110 NEXT X
120 PRINT "Negatif sayılar:"
130 FOR X = 1 TO S2
140 PRINT N(X)
150 NEXT X
```

Bu programda yeni oluşturulan iki ayrı dizi olduğundan, bu dizilerin her biri için ayrı birer sayaç kullanılmalıdır. En sondaki listelemelerde ise döngüler her dizinin kendi sayacına göre açılmalıdır. Bu programda ayrıca her dizi için, o dizinin özelliğini gösteren bir başlık yazılmıştır.

12) DATA satırındaki 11 sayıyı sondan başa doğru listeleyen bir program:

```
10 DIM A(11)
20 FOR I = 1 TO 11
30 READ A(I)
40 NEXT I
50 DATA 2,5,8,13,24,29,34,35,46,61,69
60 FOR J = 11 TO 1 STEP -1
70 PRINT A(J)
80 NEXT J
```

Bilindiği gibi, DATA satırındaki bilgileri yazıldıkları sıranın dışında bir sırada okumak olanaksızdır. Bu sayıları tersten listelemek için tek yol, hepsini bir diziye yerleştirdikten sonra bu diziyi tersten listelemektir. 60 numaralı satırda da görüldüğü gibi, indis olarak kullanılan döngü değişkeninin değeri, 11'den başlayıp istendiği gibi geriye doğru 1'er eksilerek gitmektedir. Böylece 11'inci elemandan başlayarak 1'inci elemana kadar dizi tersten listelenmiş olacaktır.

13) 5 kişiye ait isim ve telefon numaralarının kayıtlı olduğu iki diziden yararlanarak, klavyeden girilen kişiye ait telefon numarasını ekrana yazan bir program:

```
10 DIM A$(5),T$(5)
20 FOR K = 1 TO 5
30 READ A$(K),T$(K)
40 NEXT K
50 DATA MELEK, 192 42 67, MURAT, 234 52 88,
HASAN, 525 88 99, AYŞE, 912 34 17, MEHMET,
624 81 02
60 INPUT B$
70 FOR I = 1 TO 5
80 IF A$(I) = B$ THEN PRINT T$(I)
90 NEXT I
```

Programdaki ilk döngüde DATA satırındaki bilgiler ait oldukları dizilere yerleştirilecektir. daha sonra klavyeden girilen isim ikinci bir döngüyle isim dizisinde aranacak, eğer bulunursa bu ismin karşılığı olan telefon numarası ekrana yazılacaktır. Eğer isim bulunamazsa, ekrana hiçbir şey yazılmayacaktır. Bu programda telefon numaraları için sayısal değişken değil, alfabetik değişken kullanılmıştır, çünkü telefon numaralarıyla ilgili aritmetiksel işlemler (toplama, çıkarma, vb.) yapılması söz konusu değildir. Bu tür değişken kullanmanın yararı, telefon numaralarının aralarında boşluklarla veya açıklamalı olarak ( Örnek: 572 18 46 / 4 hat ) yazılabilme olanağının sağlanmasıdır.

**NOT:** Yukarıdaki program, örnek olarak 5 kişi için hazırlanmıştır. İstenirse gerekli sayılar ve DATA satırındaki bilgiler arttırılarak gerçek anlamda bir telefon rehberi olarak kullanılabilir. Ayrıca programın en sonuna eklenebilecek bir 'GOTO 60' komutuyla, isim arama sürekli olarak devam ettirilebilir.

14) 8 elemanlı bir dizideki şehir isimlerinden 1'inci ve 4'üncü harfleri aynı olanları başka bir diziye aktarıp bu yeni diziyi listeleyen bir program:

```
10 DIM A$(8),B$(8)
20 FOR R = 1 TO 5
30 READ A$(R)
40 IF LEFT$(A$(R),1) = MID$(A$(R),4,1)
THEN S = S + 1 : B$(S) =A$(R)
```

```
50 NEXT R
60 DATA ADANA, ANTALYA, BOLU, KARS, SAMSUN,
MUŞ, İZMİR,MERSİN
70 FOR D = 1 TO S
80 PRINT B$(D)
90 NEXT D
```

Bu program çalıştırıldığında, aşağıdaki görüntü ekrana çıkacaktır:

```
ANTALYA
SAMSUN
İZMİR
```

1'inci ve 4'üncü harflerin karşılaştırması 40 numaralı IF ... THEN satırında yapılmıştır. Bu satırdaki parantezlerin kullanımına özellikle dikkat edilmelidir. Buradaki karakter fonksiyonlarının gerektirdiği parantezlerin içinde, dizi elemanları kullanılmıştır. Bu durumda dizi indislerinin içine yazıldığı parantezlerin de kullanımına özen gösterilmelidir.

15) DATA satırındaki 12 ismi bir diziye yerleştirdikten sonra, klavyeden girilen sıra numarasına göre diziden istenen bir ismi ekrana yazan ve bu işlemi sürekli olarak tekrarlayan bir program:

```
10 DIM A$(12)
20 FOR H = 1 TO 12
30 READ A$(H)
40 NEXT H
50 DATA EBRU, HALE, CAN, GÜLAY, SERPİL,
SELMA, CELAL, DURSUN, FERDA, EKREM, AYHAN,
SADRİ
60 INPUT "KAÇINCI İSMİ GÖRMEK İSTİYORSUNUZ
";S
70 PRINT A$(S)
80 GOTO 60
```

```
RUN
KAÇINCI İSMİ GÖRMEK İSTİYORSUNUZ ? 4
GÜLAY
KAÇINCI İSMİ GÖRMEK İSTİYORSUNUZ ? 9
FERDA
KAÇINCI İSMİ GÖRMEK İSTİYORSUNUZ ? 2
HALE
KAÇINCI İSMİ GÖRMEK İSTİYORSUNUZ ? 12
SADRİ
KAÇINCI İSMİ GÖRMEK İSTİYORSUNUZ ? 15
Subscript out of range in 70
```

Bu programın bir hata mesajı vererek durmasının nedeni, indis değeri olarak 12'den büyük bir sayı kullanılarak dizide bulunmayan bir elemana ulaşılmak istenmesidir. Bu hata mesajı, dizi sınırları dışına çıkıldığını belirtir. Bu tür bir yanlışlıkla bu programın durmasını engellemek için, INPUT satırından sonra şu kontrol satırı eklenebilir:

```
65 IF S>12 OR S<1 THEN ?"DİZİNİN YALNIZCA 12 ELEMANI
VAR.":GOTO 60
```

16) 1987 ve 1988 yılları için klavyeden girilen aylık bilgisayar fiyatlarını iki ayrı diziye yerleştiren ve bu bilgileri uygun başlıklar altında, DATA satırından okunarak başka bir diziye yerleştirilmiş olan ay adlarıyla birlikte listeleyen bir program:

```
10 DIM F87(12),F88(12),AY$(12)
20 FOR M = 1 TO 12
30 READ AY$(M)
40 INPUT "1987 ";F87(M)
50 INPUT "1988 ";F88(M)
60 NEXT M
70 DATA OCAK, ŞUBAT, MART, NİSAN, MAYIS,
HAZİRAN,TEMMUZ, AĞUSTOS, EYLÜL, EKİM,
KASIM, ARALIK
80 CLS
90 PRINT "    İKİ YILLIK FİYAT LİSTESİ:"
```

```
100 PRINT "-----------------------------------------"
110 PRINT "AY","1987","1988"
120 PRINT "-----","-----","------"
130 FOR N = 1 TO 12
140 PRINT AY$(N),F87(N),F88(N)
150 NEXT N
```

Bu programda dizi adı olarak, kullanım amacını hatırlatması amacıyla F87 ve F88 kullanılmıştır. Buradaki 87 ve 88 sayılarının program için özel bir anlamı yoktur.

Sonucu almak için ikinci bir döngü açılmasının nedeni, bilgilerin bir bölümünün klavyeden girilmesidir. Bilgi girişiyle listeleme bir arada yapılamayacağı için, listeleme için ikinci döngü kullanılmıştır.

17) 11'er elemanlı üç ayrı dizinin karşılıklı elemanlarının toplamı 20 ile 30 arasında ise bu toplamı 4'üncü bir diziye aktarıp, bu diziyi boş bir ekranda 15'inci sütun üzerine listeleyen bir program:

```
10 DIM A(11),B(11),C(11),D(11)
20 FOR Y = 1 TO 11
30 READ A(Y),B(Y),C(Y)
40 TOP = A(Y)+B(Y)+C(Y)
50 IF TOP>20 AND TOP<30 THEN S = S + 1 :
D(S) = TOP
60 NEXT Y
70 CLS
80 PRINT TAB(15);"D DİZİSİ"
90 PRINT TAB(15);"--------"
100 FOR U = 1 TO S
110 PRINT TAB(15);D(U)
120 NEXT U
```

18) 13 elemanlı bir dizinin 1000'den büyük elemanlarının % 20'sini ikinci bir diziye atayarak bu yeni diziyi ekrana yazan bir program:

```
10 DIM A(13),B(13)
20 FOR D = 1 TO 13
30 READ A(D)
40 IF A(D)>1000 THEN S = S + 1 : B(S) =
A(D) * 0.20
50 NEXT D
60 PRINT "B DİZİSİ :"
70 FOR G = 1 TO S
80 PRINT B(G)
90 NEXT G
```

19) 11 elemanlı bir dizide, değeri 5'e eşit olan elemanların kaçıncı sırada olduklarını yazan bir program:

```
10 DIM A(11)
20 FOR Z = 1 TO 11
30 READ A(Z)
40 IF A(Z) = 5 THEN PRINT Z
50 NEXT Z
60 DATA 4,3,5,8,9,5,2,3,8,5,3
```

20) Bir fabrikada çalışan 8 işçinin saat ücretleri ve bir haftada kaç saat çalıştıkları klavyeden girildiğinde, bütün işçilerin haftalık maaşlarını listeleyen ve işçilere ödenecek toplam para miktarını yazan bir program:

```
10 DIM MAAŞ(8)
20 FOR I = 1 TO 8
30 INPUT "SAAT ÜCRETİ ";SU
40 INPUT "ÇALIŞMA SAATİ ";CS
50 MAAS(I) = SU * CS
60 TOP = TOP + MAAS(I)
70 NEXT I
80 PRINT "İŞÇİLERİN TOPLAM MAAŞLARI:"
90 PRINT "--------------------------"
100 FOR C = 1 TO 8
110 PRINT MAAS(C)
120 NEXT C
130 PRINT "---------"
140 PRINT "TOPLAM =";TOP
```

21) 13 elemanlı sayısal bir dizide, değeri 12 ile 20 arasında olan elemanların ortalamasını bularak, bu ortalamadan büyük olan elemanları listeleyen bir program:

```
10 DIM A(13)
20 FOR J = 1 TO 13
30 READ A(J)
40 IF A(J)>12 AND A(J)<2Ø THEN S = S + 1 :
TOP = TOP + A(J)
50 NEXT J
60 DATA 11,42,16,20,13,7,68,19,10,14,11,39,17
70 ORT = TOP / S
80 FOR L = 1 TO 13
90 IF A(L)>ORT THEN PRINT A(L)
100 NEXT L
```

Ortalamanın hesaplanabilmesi için, ortalaması bulunacak sayıların toplamının ve adetlerinin bilinmesi gerekmektedir. Burada koşula uyan sayıların ortalamasının bulunması istendiği için, IF ... THEN satırının sonunda hem sayma, hem de toplama işlemleri yapılmıştır. Daha sonra bu toplam, adedi gösteren sayaç değişkenine bölünerek yalnızca koşula uyan sayıların ortalaması hesaplanmıştır.

22) 6 mal için klavyeden girilen miktar ve birim fiyatına göre, boş bir ekrana 'A ŞİRKETİ SATIŞ LİSTESİ' başlığı altında 'MİKTAR', 'FİYAT' ve 'TUTAR' sütunları halinde bir liste çıkartan ve en altta da toplam satış tutarını yazan bir program:

```
10 DIM M(6),F(6),T(6)
20 FOR B = 1 TO 6
30 INPUT "MİKTAR VE FİYATI GİRİNİZ ";M(B),F(B)
40 T(B) = M(B) * F(B)
50 TS = TS + T(B)
60 NEXT B
70 CLS
80 PRINT " A ŞİRKETİ SATIŞ LİSTESİ:"
90 PRINT "-----------------------------------"
100 PRINT "MİKTAR","FİYAT","TUTAR"
110 PRINT "------","------","-------"
```

```
120 FOR Q = 1 TO 6
130 PRINT M(Q),F(Q),T(Q)
140 NEXT Q
150 PRINT ,,"---------"
160 PRINT ,"TOPLAM SATIŞ =";TS
```

Bu programda 150 ve 160 numaralı satırlarda PRINT komutundan hemen sonra kullanılan virgüller, çizgiyi ve toplam satış değerini üçüncü sütunun tam altına getirebilmek içindir.

23) İki futbol takımı için DATA satırında verilen 6 haftalık gol sayılarına göre her takım için ortalama gol sayısını bulan ve kaçıncı haftalarda ortalamanın altında gol atıldığını listeleyen bir program:

```
 10 DIM A(6),B(6)
 20 FOR W = 1 TO 6
 30 READ A(W),B(W)
 40 TA = TA + A(W)
 50 TB = TB + B(W)
 60 NEXT W
 70 DATA 3,4,1,0,2,4,2,6,4,2,1,1
 80 ORTA = TA / 6
 90 ORTB = TB / 6
100 CLS
110 PRINT "ORTALAMANIN ALTINDA GOL ATILAN
HAFTALAR:"
120 PRINT "---------------------------------------"
130 PRINT "A TAKIMI"
140 PRINT "--------"
150 FOR H = 1 TO 6
160 IF A(H)<ORTA THEN PRINT H
170 NEXT H
180 PRINT
190 PRINT "B TAKIMI"
200 PRINT "--------"
210 FOR H = 1 TO 6
220 IF B(H) <ORTB THEN PRINT H
230 NEXT H
```

24) Eleman sayısı klavyeden girilen bir diziye, yine klavyeden girilerek yerleştirilen isimleri boş bir ekran üzerine sondan başa doğru listeleyen bir program:

```
10 INPUT "ELEMAN SAYISI ";E
20 DIM A$(E)
30 FOR X = 1 TO E
40 INPUT A$(X)
50 NEXT
60 CLS
70 FOR Y = E TO 1 STEP -1
80 PRINT A$(Y)
90 NEXT
```

Bu programda dizinin eleman sayısı klavyeden girilmiş ve DIM komutuyla birlikte dizinin eleman sayısını belirleyen değer olarak E değişkeni kullanılmıştır.

**SORULAR:**

1) Aşağıdaki isim ve yaşları iki diziye yerleştiren ve yaşı 18'in üzerinde olanları listeleyen bir program yapınız.

| İSİM | YAŞ |
|---|---|
| MEHMET | 16 |
| NİL | 21 |
| ANI | 11 |
| SERPİL | 35 |
| SAMİ | 19 |

2) 5'er elemanlı A ve B dizilerinin karşılıklı elemanlarından çarpımları 200'den büyük olanların toplamlarını sıralı olarak C dizisine atayan ve en son olarak C dizisini listeleyen bir program yapınız.

| A | B |
|---|---|
| 15 | 12 |
| 19 | 21 |
| 11 | 10 |
| 13 | 12 |
| 18 | 15 |

3) Klavyeden girilen 8 ismin ilk harflerini boş bir ekrana listeleyen bir program yapınız.

4) A ve B dizilerindeki karşılıklı elemanlardan ikisi de 3'ten büyükse bu sayıların çarpımlarını, değilse toplamlarını C dizisine atayan ve üç diziyi de listeleyen bir program yapınız.

| A | B |
|---|---|
| 4 | 2 |
| 3 | 5 |
| 4 | 7 |
| 1 | 3 |
| 8 | 9 |

# DİSKET VE KASET KULLANIMI:

Bilgisayarın belleğindeki program, bilgisayar kapatıldığında veya elektrik akımının herhangi bir nedenden dolayı kesilmesi durumunda silinir. Uzun bir süre emek verilerek hazırlanmış bir program, daha sonra kullanılmak istendiğinde, kalıcı bir kayıt ortamında (disket, kaset, disk, vb.) saklanmalıdır. Örneğin, her gün kullanılacak bir programı, sabah yazmak, kullandıktan sonra silmek ve ertesi gün tekrar yazmak, özellikle uzun programlar için son derece zahmetli ve anlamsız bir işlemdir.

Programları daha sonra kullanmak amacıyla bir kayıt ortamı üzerine kaydetmek için BASIC'te **SAVE** komutu kullanılır.

Genel kullanım:

**SAVE <"program adı"> [ , P ]**
veya
**SAVE <"program adı"> [ , A ]**

SAVE komutundan sonra tırnak işaretleri arasında kullanılan program ismi en fazla 8 karakterden oluşabilir ve bu karakterler içinde harfler ve sayılardan başka karakter bulunamaz.

Program adı yazılıp tırnak kapatıldıktan sonra eğer ',P' ifadesi de komuta eklenirse, program korumalı (Protection) olarak kaydedilir;

daha sonra, programın çağırılıp listesinin görülebilmesi engellenmiş olur. Bu şekilde kaydedilmiş olan programlar yalnızca çalıştırılabilir.

P harfi yerine A harfi kullanılırsa, program ASCII (makina dilinde) kodunda saklanır.

**ÖRNEKLER:**

**1) SAVE "STOK"**

Bu örnekteki STOK ismi, bellekteki programa verilmiş olan isimdir. Program isimleri, programın amacıyla uyumlu olursa, daha sonra kullanılacağı zaman hangi program isminin hangi programa ait olduğu kolayca hatırlanabilir.

Bir program diskete kaydedildiğinde, program isminin yanına, bu programın BASIC dilinde yazılmış olduğunu gösteren '.BAS' eki bilgisayar tarafından otomatik olarak yerleştirilir. Diğer programlama dillerinde de bilgisayar değişik ekler vermektedir. (Bu ekler, istenirse programcı tarafından da belirlenebilir. Örnek: SAVE "STOK.PRG" )

**2) SAVE "B: STOK.BAS"**

Bu örnekte 'STOK' isimli program B sürücüsünde bulunan diskete kaydedilir.

**3) SAVE "TRIGONOMETRI"**

Bu örnekteki programın adı 8 karakterden uzun olduğu için, bilgisayar bu komut verildiğinde baştan 8 karakteri alarak kayıt yapacaktır. Trigonometri ile ilgili bir programa isim verilmek istendiğinde bu isim kısaca TRGNMTR olabilir.

NOT: Bir program kaydedilmek istendiğinde, aynı isim kullanılarak daha önceden kaydedilmiş başka bir program varsa, en son kaydedilen program saklanır, daha önceki silinir. (Bazı BASIC yorumlayıcılarında eski program da .BAK ekiyle saklanır.)

Disket üzerinde kayıtlı olan programların listelenmesi için, **CAT** komutu kullanılır. (Bu komut, PC türü bilgisayarlarda kullanılan BASIC yorumlayıcılarında **FILES** , işletim sisteminde ise **DIR** şeklindedir.)

Kayıt ünitesinde saklanmış bir programı belleğe geri almak amacıyla BASIC dilinde **LOAD** komutu kullanılır.

Genel kullanım:

**LOAD "program adı"**

LOAD komutuyla bir program adı kullanılırken, SAVE komutuyla geçerli olan kurallara uymak gerekir.

NOT: Disket kullanılıyorsa, programların disket üzerindeki yerini bilgisayar kendiliğinden belirler. Böylece birden çok programın yanlışlıkla aynı bölgeye kaydedilmesinin engellendiği gibi, bir programa ulaşmak istendiğinde, o programın disket üzerinde aranması, bilgisayar tarafından gerçekleştirilir. Kaset kullanılıyorsa, programların kasedin neresinde kayıtlı olduğunu bilmek, programcının görevidir. Kaset kullanırken programların üst üste kaydedilmesi gibi yanlışlıklardan kaçınmaya çok dikkat edilmelidir. Bunun için en iyi yardımcı, hemen hemen bütün kaset ünitelerinin üzerinde bulunan sayaçtır (numeratör). Bazı gelişmiş bilgisayarlarda, kaset üzerindeki programların aranma işlemi, bilgisayar tarafından otomatik olarak yapılabilmektedir.

**ÖRNEK:**

**LOAD "STOK"**

Bu komut, STOK adıyla daha önceden kaydedilmiş olan programı belleğe geri alır. Bu arada, eğer bellekte başka bir program varsa, STOK programı onun yerini alır ve önceki program bellekten silinir. STOK isimli program daha önceden kaydedilmemişse (veya yanlış diskette aranıyorsa), bilgisayar bir hata mesajı (File not found - Dosya bulunamadı) vererek işlemi gerçekleştirmez.

# ALT PROGRAMLAR:

Bir program içerisinde sık sık tekrarlanacak bir grup deyim veya fonksiyonu programın çeşitli yerlerinde bir kereden çok kullanmak yerine bu bölüm yalnızca bir kez yazılıp, program içinde gerekli olduğu yerlerde özel bir komut yardımıyla çağırılarak kullanılabilir. Bu tür kullanım, programın hazırlanmasını, kontrolünü ve hataların bulunarak düzeltilmesini büyük ölçüde kolaylaştırır.

Asıl programla birlikte, fakat ondan bağımsız olarak yazılan programın bu bölümüne alt program (subroutine) adı verilir. Tüm programın, alt program dışındaki bölümüne ise ana program (veya çağıran program) denir. Bir programda, istenilen sayıda alt program olabilir. Bir alt program, genellikle ana program tarafından çağırıldığı gibi, başka bir alt program tarafından da çağırılabilir.

Bu işlemleri gerçekleştirmek için BASIC'te **GOSUB** ve **RETURN** komutları kullanılır.

**GOSUB komutu:**

GOSUB komutu, bir program içinde belirli bir alt programa giderek bu alt programın uygulanmasını sağlamak için kullanılır. GOTO ve SUBROUTINE kelimelerinden türemiştir.

Genel kullanım:

**GOSUB <satır numarası>**

Buradaki satır numarası, gidilecek alt programın ilk satırının numarasıdır. (BASIC dışındaki birçok dilde ve yeni çıkan bazı BASIC derleyicilerde satır numarası yerine, alt programın tümüne verilmiş olan bir isim kullanılabilir.)

GOSUB ve GOTO komutları, kullanım ve işlev olarak birbirlerinin hemen hemen aynıdır. Program akışı, iki komutta da numarası verilmiş olan satırdan devam eder. Tek ve çok önemli olan fark, GOSUB komutunun, program akışının aktarıldığı alt programdan ana programa dönüş yapılacağını belirtmesidir.

**RETURN komutu:**

Alt programdan ana programa geri dönüşü sağlayan komut, RETURN komutudur. Bu komut, bir alt programın 'geri dön' anlamına gelen en son komutudur.

Genel kullanım:

**RETURN**

RETURN komutu program akışını, ait olduğu alt programa geçişi sağlayan GOSUB komutundan bir sonraki komuta gönderir.

GOSUB ve RETURN komutlarının program içinde kullanımlarıyla ilgili bir örnek:

```
10 Komut-1
20 GOSUB 200
30 Komut-4
40 Komut-5
50 END
200 Komut-2
210 Komut-3
220 RETURN
```

Bu programda 10 numaralı satırdaki Komut-1 uygulandıktan sonra program akışı GOSUB 200 komutuyla 200 numaralı satıra geçer. 200 ve 210 numaralı satırlardaki komutlar da uygulandıktan sonra, 220 numaralı satırdaki RETURN komutu programı 30 numaralı satıra geri gönderir ve program 30, 40 ve 50 numaralı satırlardaki komutların uygulanmasıyla sona erer. Bu programda 10 ve 50 numaralı satırlar arasındaki bölüm ana program, 200 ve 220 numaralı satırlar arasındaki bölüm ise alt programdır.

Ana programdan alt programa geçiş yalnızca GOSUB komutuyla olmalıdır. Bunun dışında, program satır numaralarının sırasına göre normal akışıyla alt programa geçmemelidir, çünkü bu tür bir geçiş sonucunda alt program bir kez daha uygulanacak ve bilgisayar RETURN komutuyla karşılaştığında, GOSUB kullanılmadan RETURN kullanılmış olduğu için bir hata mesajı (RETURN without GOSUB) vererek programı durduracaktır. Bu tür hatalı geçişleri engellemek için, ana program ve alt program arasında STOP, END veya GOTO komutlarından birisi kullanılmalıdır.

**ÖRNEKLER:**

1) Klavyeden girilen bir sayının iki katını bir alt programda hesaplayıp ana programda sonucu yazan bir program:

```
10 INPUT A
20 GOSUB 100
30 PRINT B
40 END
100 B = A * 2
110 RETURN
```

Bu programda, A sayısı klavyeden girildikten sonra, bilgisayar GOSUB 100 komutuyla 100 numaralı satıra gidecek, burada B'nin değerini hesaplayacak, RETURN komutuyla GOSUB satırından bir sonraki satır olan 30 numaralı satıra dönecek ve B'nin değerini yazarak sona erecektir.

2) 12 elemanlı bir isim dizisinin klavyeden girilen bilgilerini bir alt programda listeleyen ve sona eren bir program:

```
10 DIM A$(12)
20 FOR C = 1 TO 12
30 INPUT A$(C)
40 NEXT C
50 GOSUB 200
60 END
200 FOR D = 1 TO 12
210 PRINT A$(D)
220 NEXT D
230 RETURN
```

Klavyeden girilen bir dizi isim, 200 numaralı satırdan başlayan alt programda listelenecek ve RETURN komutuyla doğrudan doğruya END'e gelerek duracaktır.

3) 8 elemanlı bir dizinin eleman değerlerini DATA satırından okuyarak listeleyen, daha sonra dizinin 3'üncü, 4'üncü ve 5'inci elemanlarının değerlerini klavyeden girilecek sayılara eşitleyerek dizi elemanlarının hepsini tekrar ekrana listeleyen bir program:

```
10 DIM A(8)
20 FOR R = 1 TO 8
30 READ A(R)
40 NEXT R
50 DATA 4,6,3,7,9,8,1,3
60 GOSUB 500
70 FOR T = 3 TO 5
80 INPUT A(T)
90 NEXT T
100 GOSUB 500
110 END
500 FOR I = 1 TO 8
510 PRINT A(I)
520 NEXT I
530 RETURN
```

Bu programda, dizi elemanlarının hepsinin yazdırılması işlemi iki kez tekrarlanmaktadır. Program içinde iki ayrı FOR ... NEXT döngüsü kullanarak listelemek yerine bu döngü yalnızca bir kez, fakat alt program olarak kullanıldığında, istenildiğinde çağırılarak (GOSUB

500) listeleme yapılabilir. 60 numaralı satırdaki GOSUB komutu, alt programın bir kez uygulanmasını sağlar. Bundan sonra program 70 numaralı satıra geri döner. 100 numaralı satırdaki GOSUB komutundan sonra ise bilgisayar alt programı ikinci kez uygulayarak 110 numaralı satıra döner ve sona erer. Diğer bir deyişle, RETURN komutu en son kullanılan GOSUB komutundan bir sonraki komuta dönüşü sağlar.

4) 8 çeşit mala ait mal adı ve fiyat listesini klavyeden girilen bilgilere göre boş bir ekrana uygun başlıklar altında çıkartan, bu listeyi ekranda bir süre beklettikten sonra aynı işlemi 12 çeşit mal için tekrarlayan bir program:

```
 10 DIM M$(12),F(12)
 20 FOR X = 1 TO 8
 30 GOSUB 200
 40 NEXT X
 50 GOSUB 300
 60 FOR T = 1 TO 5000
 70 NEXT T
 80 CLS
 90 FOR X = 1 TO 12
100 GOSUB 200
110 NEXT X
120 GOSUB 300
130 END
200 INPUT M$(X),F(X)
210 RETURN
300 CLS
310 PRINT "MAL ADI","FIYATI"
320 PRINT "-------","------"
330 FOR X = 1 TO 12
340 PRINT M$(X),F(X)
350 NEXT X
360 RETURN
```

**SORULAR:**

1) DATA satırından isimleri ve doğum yılları okunan beş kişinin yaş ortalamasını bir alt programda hesaplayıp yazan ve ana programda yaşları ortalamadan küçük olanların isimlerini listeleyen bir program yapınız.

DATA Ahmet,1971,Mehmet,1959,Ayla,1954,Hakan,1973,Turan,1968

2) Bir dikdörtgenin iki kenarının uzunlukları klavyeden girildiğinde, bir alt programda dikdörtgenin alanını hesaplayıp, ana programda sonucu yazan, eğer alan 50'nin altındaysa başa dönen bir program yapınız.

# BİLGİSAYARIN GRAFİK, RENK VE SES ÖZELLİKLERİ:

Bilgisayar ekranındaki görüntü, satırlar ve sütunlar halindeki noktalardan oluşmaktadır. Piyasada en çok kullanılan bilgisayarlarda ekran soldan sağa 640, aşağıdan yukarı 200, yani toplam 128000 (640 x 200) noktadan oluşmaktadır. Normalde ekran üzerine yazılan karakterler de belirli bölgelerde belirli noktaların yan yana ve alt alta gelmesiyle oluşmaktadır.

Ekran üzerinde, standart karakterler dışında şekiller oluşturulmak istenirse, hemen hemen her bilgisayarda farklı olarak kullanılan birkaç komuttan yararlanmak gereklidir.

Ekranın grafik yapısı, koordinat sistemine benzer olarak düzenlenmiştir. Sol üst köşe merkez (0,0) noktası olarak kabul edilirse, sağ üst köşe (639,0), sağ alt köşe (639,199) ve sol alt köşe (0,399) olacaktır. (639 ve 199 sayıları, 0'ıncı nokta satırı ve sütunu da ekran üzerinde olduğu için kullanılmaktadır. 0'dan 639'a kadar 640 nokta vardır.)

**ÖRNEKLER :**

1) Ekranın tam ortasına bir nokta koyan bir komut:

**PSET (320,100)**

PSET komutu, kendisinden sonra verilmiş olan sayılara göre ekran üzerinde istenilen bir yere nokta koymak amacıyla kullanılır. Komuttan sonra verilen ilk sayı merkezden başlayarak kaç nokta sağa gidileceğini (x koordinatı), ikinci sayı ise bu noktadan itibaren kaç nokta aşağı gidileceğini (y koordinatı) belirler. Ekranın ortasına ulaşmak için gerekli koordinatlar 320 (640/2) ve 100 (200/2) olarak hesaplanabilir.

2) Ekranın sol üst köşesinden sağ alt köşesine bir doğru çizen bir program:

**LINE (0,0)-(639,199)**

LINE komutu, kendisinden sonra verilmiş olan koordinatlara göre ekran üzerinde istenilen bir yere çizgi çizmek amacıyla kullanılır. Birinci koordinat çizginin başlangıç noktasını, ikincisi ise bitiş noktasını belirler.

LINE komutunun ikinci kullanılış şekli ise aşağıdaki gibidir:

**LINE -(50,100)**

Bu durumda bilgisayar, grafik kursörünün en son kaldığı noktadan (50,100) noktasına bir çizgi çizecektir.

**Renk ve ses komutları:**

Ekran üzerinde elde edilen şekiller ve yazıların rengi kullanılan bilgisayarın kapasitesine göre istenildiği gibi düzenlenebilir. Eğer renkli bir monitör kullanılıyorsa, bilgisayarın özelliklerine göre çeşitli renkler elde edilebilir. Monitör tek renkli (siyah-beyaz veya yeşil) ise, yalnızca bu renklerin tonları (renkli yayınların siyah-beyaz televizyonda izlenmesi gibi) görülebilir.

Renk seçimi için COLOR komutu kullanılır. Genel kullanım, COLOR A,B,C şeklindedir. Burada A ekrana yazılan şekiller, B zemin, C ise ekran çerçevesinin rengini gösterir. Bu komuttan sonra ekrana yazdırılan yazı veya şekiller, belirlenen renklere uygun olacaktır.

Grafik konusunda olduğu gibi, ses konusunda da özel komutlar kullanılarak bilgisayarın istenilen sesi çıkarması sağlanabilir.

Bilgisayarın herhangi bir sesi çıkarması için bu sesin periodu ve süresi verilmelidir.

**ÖRNEK:**

**SOUND A,B**

SOUND komutundan sonra verilen 2 parametre (A ve B) mutlaka kullanılmalıdır. A sayısı sesin periodunu (incelik-kalınlık ölçüsü) ve B sayısı süresini belirler. A sayısı küçüldükçe ses kalınlaşır; büyüdükçe incelir. B sayısı küçüldükçe sesin süresi kısalır; büyüdükçe uzar.

# HATA MESAJLARI

**1) NEXT without FOR**

NEXT komutu FOR...NEXT döngüsü dışında kullanılmış veya döngü değişkeni yanlış yazılmıştır.

**2) Syntax error**

Yazım hatası yapılmıştır. BASIC dilinin tanıdığı komutların dışında bir kelime yazılmış veya komutlardan biri yanlış olarak kullanılmıştır.

**3) RETURN without GOSUB**

RETURN komutu alt program dışında kullanılmıştır. (Daha önce GOSUB kullanılmamıştır.) Genellikle ana programın durdurulmaması ve program akışının istenmeyen bir biçimde alt programa geçmesi sonucunda karşılaşılır.

**4) Out of DATA**

READ komutuyla kullanılan DATA satırındaki veriler eksik yazılmış veya DATA satırı hiç yazılmamıştır.

**5) Illegal function call**

Bir komut veya fonksiyonla kullanılması gereken parametreye geçersiz bir değer verilmiştir.

**6) Overflow**

Bir aritmetik işlemle elde edilen sonucun mutlak değeri, bilgisayarın kapasitesini aşacak derecede büyüktür.

**7) Out of memory**

Bellek kapasitesi dolmuş veya GOSUB komutu çok fazla iç içe kullanılmıştır.

**8) Undefined line number**

Programda var olmayan bir satıra ulaşmak veya bu satırı silmek istenmiştir.

**9) Subscript out of range**

Bir matris veya dizinin indislerinden herhangi biri çok büyüktür veya boyut sayısı yanlış olarak kullanılmıştır. (Dizide var olmayan bir elemana ulaşılmak istenmiştir.)

**10) Duplicate definition**

Bir dizi veya matris ikinci bir kez tanımlanmak istenmiştir.

**11) Division by zero**

Sıfıra bölme yapmak istenmiştir.

**12) Illegal direct**

Direkt modda (program dışında) geçersiz olan bir komut verilmiştir.

**13) Type mismatch**

Sayısal bir bilgiyle alfabetik bir bilgi aynı işlemde kullanılmış veya karşılaştırılmıştır.

**14) Out of string space**

Programda belleğe sığmayacak kadar çok karakter üretilmiş ve bellekte saklanmak istenmiştir.

**15) String too long**

Çok uzun bir karakter dizisi (255 karakterden fazla) oluşturulmak istenmiştir.

**16) String formula too complex**

Karakter dizisi çok karmaşık ifadelerden oluşturulmak istenmiştir. (Çok fazla karakter fonksiyonu iç içe kullanılmıştır.)

**17) Can't continue**

Hatalı olduğu için devam etmesi olanaksız bir program durdurulduktan sonra devam ettirilmek istenmiştir.

**18) Missing operand**

Bir komutun veya işlemin gerektirdiği ifadeler, sabitler veya değişkenlerden biri veya birkaçı eksik bırakılmıştır.

**19) Line buffer overflow**

Bir program satırına sığmayacak kadar çok karakter satıra yazılmaya çalışılmıştır.

**20) FOR without NEXT**

Bir FOR...NEXT döngüsünde NEXT kullanılmamıştır.

**NOT: Diğer hata mesajları İleri Basic kitabının sonunda verilmiştir.**