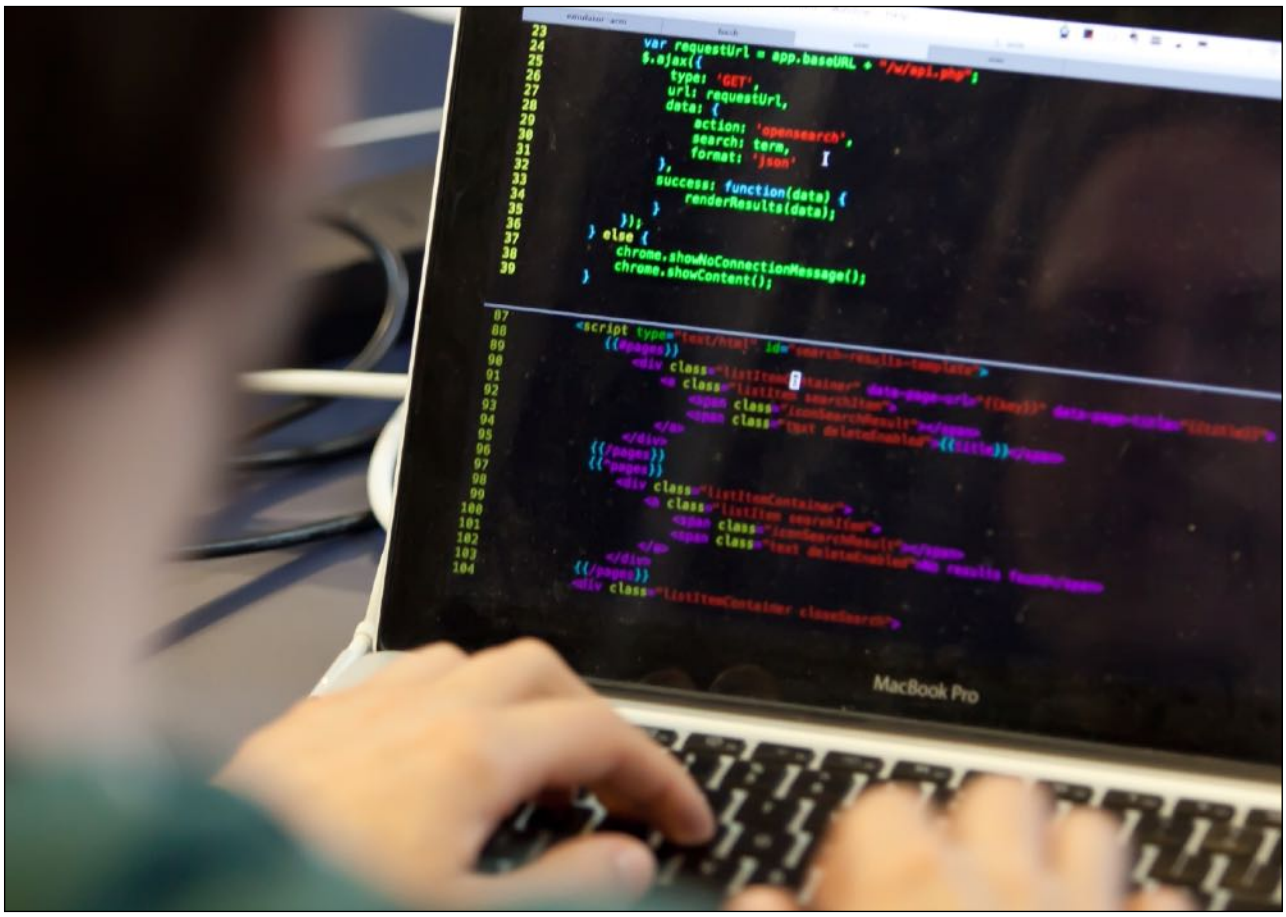
BİLGİSAYAR PROGRAMLAMA

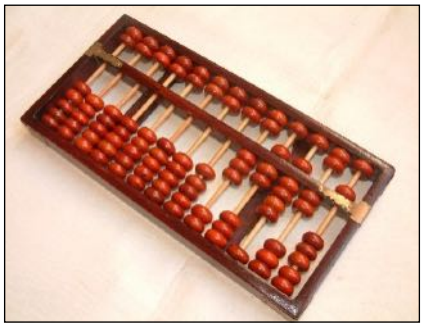
Algoritma ve Programlama Mantığı

ADNAN MENDERES ÜNİVERSİTESİ, ZİRAAT FAKÜLTESİ - 2022

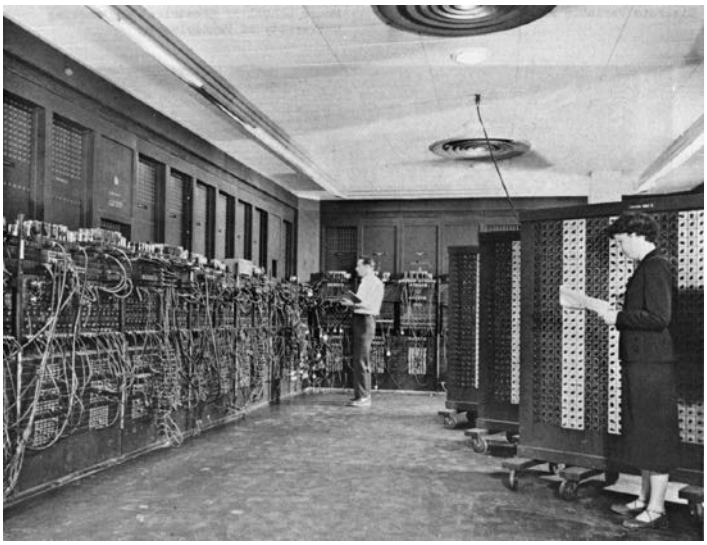


Giriş

Bilgisayarlar bir çok alanda kullanılan, hızlı ve doğru hesaplama yapabilen gelişmiş elektronik aygıtlardır. Günümüz teknolojisindeki gelişme ile birlikte bilgisayarların hızı artarken boyutu küçülmeye başlamıştır. Ayrıca işlem gücünün yanında giriş ve çıkış birimlerindeki bazı yenilikler ile kullanıcılar bilgileri ve istekleri bilgisayara daha rahat iletebilmekte, daha görsel ve farklı duyulara hitap eden çıktılar alabilmektedir. Çoğu uzman tarafından MÖ 2700-2300 yıllarında kullanılan *Abaküs* (solda) ilk bilgisayar olarak işaret edilmektedir. 1900 lü yıllarda Yunanistan açıklarında bulunan ve MÖ 27 de kullanıldığı düşünülen *Antikythera Mekanizması* (ortada) o dönemlerde gezegenlerin pozisyonlarını hesaplamakta kullanıldığı belirlenmiştir. Mekanik parçalar içeren bu tür



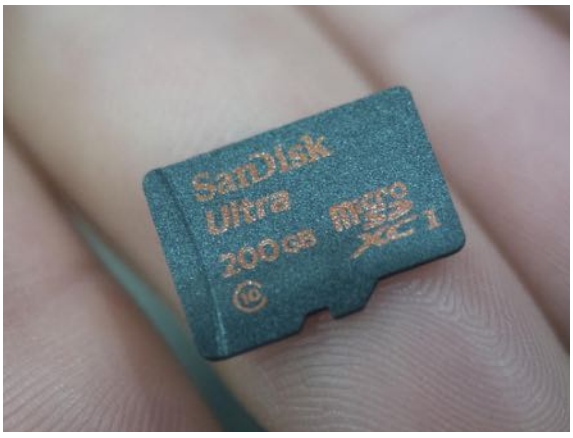
cihazlar belirli bir amaca hizmet etmeleri için tasarlanmışlardır. Daha sonradan kullanıcılar tarafından programlanarak hesaplama yapan *Babbage nin Fark Makinesi* (1833) (sağda) bilgisayarların atası olarak kabul edilmiştir. Mekanik parçalardan oluşan bu makineyi yaklaşık 100 yıl sonra üretilen ve hem mekanik hem de elektronik parçalardan oluşan *Z3* (1938) izlemiştir. 1930 larda ise ilk programlanabilir dijital bilgisayar *Colossus* tanıtılmıştır. Bu bilgisayarlarda günümüz bilgisayarların temelini oluşturan transistörlerin atası *vakum tüpleri* kullanılmıştır. Colossus'u takiben 1946 daha hızlı ve daha esnek olan



ENIAC (yanda) duyurulmuştur. Amerikan ordusu tarafından balistik hesaplamalarda kullanılan bu bilgisayar büyük bir salonu kaplayacak büyüklükteydi. 1980 lerde kişisel bilgisayar (Personel Computer [PC]) kavramı ile bilgisayar ticari kurumlara, okullara ve evlere girmeye başlamış ve bu süreç hızlı bir şekilde devam etmiştir. Bu yıllarda 10 MHz olan bilgisayarların hızları

günümüzde 3000 MHz in üzerine çıkmış ve çok hızlı işlem yapar duruma gelmiştir. Bilgisayarların hızlanması ve gelişmesiyle birlikte bilgilerin kaydedilmesine olanak sağlayan depolama alanlarına ihtiyaç duyulmuş ve bu amaca hizmet eden *Hard Disk Drive* (Hard Disk Sürücüsü) adında depolama ürünleri geliştirilmiştir. Gerek duyulan saklama alanının azlığı ve eldeki mevcut teknoloji nedeniyle ilk *HDD* (Hard Diskler) oldukça büyük ve az kapasiteye sahiplerdi. Her geçen gün daha fazla alana ihtiyaç duyulması nedeniyle hızlı bir şekilde kapasiteleri artmaya başlamıştır. Bunun aksine ise boyutları giderek ufalmıştır. 1950 li yıllarda bir dolap büyüklüğünde olan depolama birimleri (yukarıda) günümüzde bir pulun 4 te 1 büyüklüğüne (en altta) gelmiştir. Boyu ufalırken kapasiteleri ise binlerce kat artmıştır. Günümüzde media adı altında yer alan resim, müzik ve videolar depolama birimlerinde en çok yer kaplayan kaynaklardır.

Kişisel bilgisayar (*PC - Personel Computer*) çeşitli kısımlardan oluşmaktadır. Bu kısımlar hesaplamaların, işlemlerin yapıldığı ana kısım, bilgisayarın ürettiği ya da başkalarının ürettiği bilgilerin saklandığı depolama birimleri (Sabit veya harici diskler), kullanıcıların bilgisayara isteklerini ilettiği veri gerişini yaptıkları giriş birimleri (Klavye ve fare) ve işlenen/bulunan bilgilerin kullanıcaya gösterildiği çıkış birimleri (Ekran, printer, hoparlör vs) kısımlarından oluşmaktadır. Bilgisayar donanım (*Hardware*) ve yazılım (*Software*) kısımlarından oluşmaktadır. Donanım elektronik düzeneklerin herbirine ya da tümüne denilmektedir. Yazılım ise bu donanımın çalışması için programcılar tarafından yazılan ve bilgisayarın belli bir amaca hizmet edecek şekilde çalışmasına yarıyan komutlar dizisidir. Günümüz bilgisayarları ilk



alındıklarında sadece donanım olarak gelmezler. Üzerinde donanımın çalışması, kullanıcıların program yükleyebilmesi için bir yazılımla birlikte gelirler. Bilgisayar açıldığında otomatik olarak yüklenen ve insanların bilgisayarı kullanmasına yarıyan bu yazılımlara işletim sistemi adı verilmektedir. İşletim sistemi olamayan bir bilgisayar hiç bir işe yaramamaktadır. İşletim sistemleri bilgisayarı çalışır vaziyete getirmekle birlikte içlerinde basit işlerin yapılabileceği bazı programları da içerebilmektedir (Hesap makinesi, not defter, takvim vs). Ancak bu programlar çok karmaşık işlemler için tasarlanmadıkları için yetersiz kalırlar. Bu nedenle kullanıcılar bilgisayarlarına hangi amaçla kullanılacaklar ise ona yönelik yazılmış özel programları (Resim işleme, mimari çizim yapma, oyun vb.) bilgisayarlarına kurmak zorundadırlar. Bu özel programlar bilgisayar alındıklarında üzerinde kurulu gelmezler. Bilgisayar için belli bir ücret ödeyen kişi bu tür programalar için de belli bir bedel ödemesi hatta bazı durumlarda işletim sistemi için bile ödeme yapması gerekmektedir. Yazılımların fikir ve sanat eseri çerçevesinde incelendiğinde belli bir emeğin karşılığı olduğu her ne kadar gözle görülür elle tutulur olmasalar da ciddi bir çabanın ürünü olduğu akıldan çıkarılmamalıdır. Bu nedenle bu yazılımların korsan olarak yüklenmesinden ziyade ücretinin ödenerek lisanslı olarak bilgisayara kurulması gerekmektedir. Ancak bazı programlar üreticileri tarafından kullanıcıların para ödemeden kullanabilmesi için yazılmışlardır. Bu tür programlar internetten rahatlıkla bulunabilir ve bilgisayara kurulabilirler. Bu programlara *Freeware* adı verilmektedir. Genellikle belli bir amaca hizmet eden ve para ödenerek bilgisayara kurulabilen programaların *Freeware* alternatifleri vardır. Programlar programcılar tarafından *programla dilleri* kullanılarak oluşturulurlar. Yani kullandığımız programlar da başka bir program tarafından meydana getirilirler. Programcılar kullandıkları programlama diline özgü (ki çok sayıda programlama dili vardı) bir takım komutları alt alta yazarak programı oluştururlar. Bilgisayar programcıların program yazabilmeleri için bu yazdıkları programın hangi sorunu çözeceğini ve neye yarayacağını iyi irdelemesi ve anlaması gerekmektedir. Yani yazılan bilgisayar programının çözeceği sorun tam olarak ortaya konulmalıdır. Sorunu anlayan programcı programı yazmadan önce sorunun nasıl çözüleceğine dair fikir jimnastiği yapar, araştırmalarda bulunur. Sorunun eksiksiz çözülmesinin yanında hızlı bir şekilde çözülmesi de gerekmektedir. Bu konuları göz önünde bulunduran programcı *Algoritma* adı altında neyi, ne zaman ve nasıl yapacağını iyice gözden geçirerek bir *akış şeması* oluşturur. Bu akış şeması üzerinde alıştırmalar

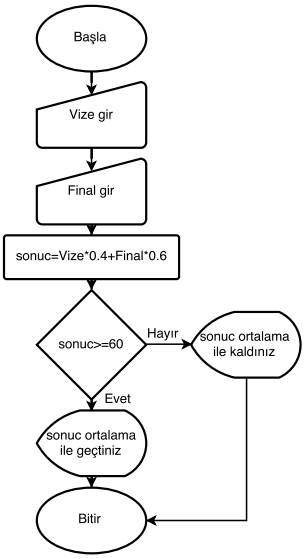
Bilgisayarda bir harfin (karakter) kapladığı yerin büyüklüğüne 1 Byte denir.
1024 Byte = 1 Kilobyte (kb)
1024 kb = 1 Megabyte (Mb)
1024 Mb = 1 Gigabyte (Gb)
1024 Gb = 1 Terabyte (tb)

yaparak sorunun çözülüp çözülmediğini kontrol eder. Daha sonra seçtiği ya da hakim olduğu bir programlama dili ile oluşturduğu bu algoritmayı programlama diline özgü komutları yazarak oluşturur. Çalışıp çalışmadığını farklı koşullarda veriler girer doğru hesaplayıp hesaplamadığını kontrol eder. Buna ek olarak yazdığı programa uygun olmayan veriler girerek bu uygunsuz koşullarda programın nasıl davranacağını hata verip vermeyeceğini kontrol eder ve programı son haline getirir. Bu süreci bir örnekle açıklayalım.

İlk önce bilgisayarla çözülmesi gereken bir sorunun/problemin ortaya konması gerekmektedir. Bizden öğrencilerin aldığı vize ve final notundan ortalamalarının ne olacağı, geçip geçemeyeceklerine dair bir bilgisayar programı yazmamız istensin. Bu soruyu çözebilmemiz için bazı temel bilgilere ihtiyacımız var. Örneğin vizenin % kaçı ve finalın % kaçı ortalamayı etkilemektedir. Ortalama kaç ve üzeri olduğunda öğrenci geçmektedir. Bu bilgiler dikkat edilecek olursa sabittir. Programcı hemen bunu sorgular ve bu sabit değerleri ister. Programcı adım adım neler yapacağını düşünmeye başlar. İlk önce vize ve final notlarının girilmesi istenmelidir. Daha sonra öğrendiğimiz sabit sayılar kullanılarak bu girilen notlar ile ortalama hesaplanmalıdır. En sonunda elde edilen bu ortalama notun öğrencinin geçip geçemeyeceği konusunda karar verilmelidir. Tüm bu işlemler bittikten sonra sonucun ekrana yansıtılması gerekmektedir. Bu aşamaya kadar dikkat edilecek olursa daha program yazmaya başlanmamıştır. Sorunun tanımı ve nasıl yapılacağı konusunda fikirler tartışılmaktadır. Bu aşamaya algoritma denilmektedir. şekiller çizerek sembolize etmeye de akış şeması denilmektedir. Örnek basit gibi görünse de değişik senaryoların da düşünülmesi gerekmektedir. Bu hatalı bir sonucun verilmemesi ve programın hata verip kendini kapatmaması için önemlidir. Örneğin yukarıda verilen örnekte vize notu istendiğinde kullanıcı 120 yazarsa ne olur? Programcı algoritmayı oluştururken bu gibi durumları düşünmek zorundadır. Algoritmaya yani sıra ile yapılması gerekenler şemasına vize ve veya final notu girildiğinde notun 0 ile 100 arasında olup olmadığı kontrol edilmesi varsa bir yanlışlık kullanıcının değeri tekrar girmesi istenmelidir. Bu ve bunun gibi tüm olumsuzluklar düşünülmek zorundadır. Bilgisayar göz açıp kapayıncaya kadar bu bahsedilen problemi tek kişi için çözebilir. Peki bu şekilde notu hesaplanması gereken milyonlarca kişi varsa? Bu noktada ürettiğiniz çözümün kısa ve hızlı olması gerekmektedir. Örneğin karmaşık matematiksel bir işlem yapmanız istendi ve işlem 1 saniyenin yarısı bir sürede hesaplanıp bitti. Gayet kısa bir süre, 0.5 saniyede anca göz açılıp kapanır, sonuç iyi mi peki? Aynı işlemi verilen 1 milyon rakam ile yapmanız istendi, her işlem 0.5 saniye sürüyordu x 1.000.000 = 500.000 saniye bu da yaklaşık 6 gün demektir. Tasarladığınız çözümde yapacağınız bazı değişiklikler ile bu çözümü kısaltmanız gerekmektedir. Tek bir işlemde anlaşılamayan hız sorunu çok

işlemde büyük bekleme sürelerine denk gelebilmektedir. Belki de bu tüm işlemleri 1 dakikanın altında yapmanın başka bir yolu vardır? Belki de! Önünüzde 10 tane farklı sayı olduğunu hayal edin ve bunları sıralamanız gerektiğini düşünün, beyniniz hemen kendine göre bir çözüm bulur ve bunları sıralarsınız. 100 sayı 1.000 sayı ya da 10.000 olduğunu düşünün! Artık beyninizin bunları hemen sıralayamadığını göreceksiniz ve bunları sıralamanın bir yolunu bulmaya çalışacaktır. Ve farklı yöntemlerin olduğunu göreceksiniz, kiminin hızlı, kiminin çok emek istediğini fark edeceksiniz. İşte algoritma bir bilgisayar programının sorunu nasıl çözeceğine dair hangi sıralı adımları işleyeceğini tasarlamaktır. Bu tasarının şekiller ile gösterilmesi ise akış şemasıdır. Programlama bu akış şeması bittikten sonra yazılmaya başlanır.

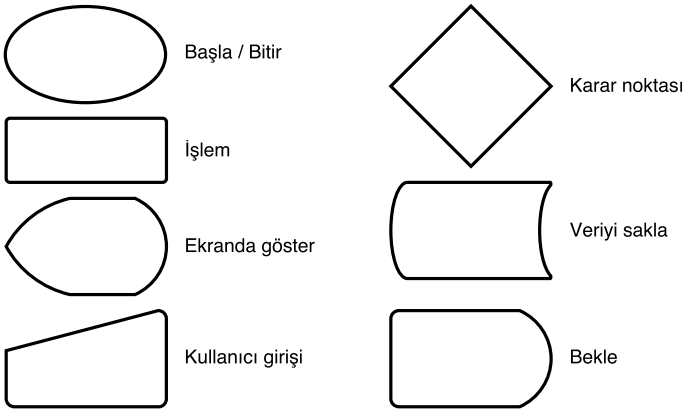
Bu dersin amacı basit problemlerin çözümünde algoritma geliştirmek ve geliştirilen bu algoritmaya göre bir bilgisayar programını yazmayı hedeflemiştir. Bununla birlikte yazılan programın test edilmesi ve varsa hatalarının giderilmesi, muhtemel kullanıcı yanlış davranışlarının önceden sezilerek yazılan programın stabil çalışmasını sağlamayı hedeflemiştir.



Algoritma

Yukarıda da anlatıldığı üzere algoritma bir şeyin nasıl yapılacağı konusunda önceden düşünmek ve bu düşünceyi şematize etmektir. Günlük yaşantımızdaki herşeyin algoritması çıkarılabilir. Örneğin bir çamaşır makinesinin çamaşırı nasıl yıkadığını düşünebilirsiniz. Belli bir sıra içerisinde devam eden olaylar kolaylıkla şematize edilebilir. Örneğin su al, deterjan al, tamburu 50 kez sağa sola çevir, suyu at, tekrar su vb. Ancak daha detaylı düşünüldüğünde bu adımların çok basit kaldığı aslında çok daha detaylı bilgilerin gerektiği ortaya çıkmaktadır. Örneğin kapısı açık bir çamaşır makinesi çalışmaz, demekki algoritmasının içerisinde kapı açık mı > hayır > o zaman çalıma gibi bir adım içermektedir. Bu ve bunun gibi gözden kaçırabileceğimiz çok sayıda adım yer alabilir. Bu adımlar uzun çalışmalar sonucunda mühendislerin yaptığı denemeler ve hesaplamalar ile kesinleştirilmiştir. Mesela tamburu kaç tur attırmak çamaşırı iyi yıkar, ne kadar su çamaşırın iyi yıkanması için yeterli gelmektedir, iyi durulanması için ne kadar suya ihtiyaç vardır? Tüm bunlar araştırılıp, derlenip, planlanıp tek bir detaylı algoritma çıkarılması gerekmektedir. Yapılan algoritmanın iyi bir şekilde algılanabilmesi için şekilsel sembolleştirilmesi çok yararlı olacaktır. Nitekim programcılar belli standart şekilleri algoritmalarında kullanarak şematize ederek akış şemalarını oluştururlar. Böylece yapılan kurgu çok rahat bir şekilde takip edilebilmektedir. Algoritma evrenseldir, programı hangi bilgisayar programında yapılacak olursa olsun tek tiptir. Bu günlük yaşantımızdaki yapacağımız herhangi bir iş planı gibidir. Programlama ise bu algoritmanın verilen akış sırası ile komut yazmasıdır ve her bilgisayar programı için farklılıklar gösterir.

Algoritma ve akış şemasında kullanılan başlıca şekiller ve anlamları aşağıda verilmiştir. Buna göre yukarıdaki problemin akış şeması çıkarılacak olursa resimdeki (önceki sayfa) gibi bir akış çıkmaktadır. Bu örnek kullanıcı girişi, işlem, karar verme ve ekrana gösterme adımları içeren basit bir örnektir. Dikkat edilecek olursa vize çarpanı 0.4, final çarpanı 0.6 ve geçme sınırı 60 sabit olarak verilmiştir. Bu tür değeri değişmeyen ve program içinde sıklıkla kullanılan değerlere *sabitler* adı verilmektedir.



Programlama

Neyin nasıl yapılacağı akış şeması ve algoritma oluşturulduktan sonra sıra bunun bilgisayar programının yazılmasına gelmektedir. Bilgisayar programları programlama dilleri kullanılarak yazılan komut dizilerinden ibarettir. Yazılan program programlama dili içerisinde çalıştırılarak kullanılabileceği gibi, kendi başına çalışır ve başka bilgisayarlara gönderilerek başka kullanıcıların da çalıştırmasına olanak sağlayan kendi çalışır bir forma (derleme) da sokulabilmektedir. Yazdığınız program çeşitli komutlardan oluşmaktadır. Program yazma işi bittiğinde bu komutlar derlenir ve *EXE* uzantılı bir dosya oluşturulur. Bu dosya istenilen bir bilgisayara kopyalanarak çalıştırılabilir yapıdadır ve sizin komutlarını yazdığınız programı çalıştırır. Yazdığınız kodların yalın halde başka bir bilgisayara kopyalanıp çalıştırılması mümkün değildir, muhakkak derlenip EXE uzantılı dosyanın oluşturulması gerekmektedir. Bu çalışan EXE uzantılı dosya sizin istediklerinizi tek başına yapmasına karşın bu dosyanızın içine girilip yadığınız komutlar görülememektedir. Bu dosyaların içinin okunup yazdığınız kodların görülmesi imkansızdır. Programda yaptığınız bir hatayı farkederseniz yazdığınız kodlar üzerinde gerekli değişiklikleri yapıp tekrar derleyerek EXE dosyanın oluşturulması gerekmektedir. EXE dosyası üzerinde değişiklik yapılması imkansızdır. EXE dosyası üzerinde yapılacak herhangi bir değişiklik bilgisayarın kilitlenmesine dahi neden olabilecek sorunlar çıkarabilir.

Peki bilgisayar programı nasıl yapılmaktadır? Bilgisayar programı kullanılan dile özgü belli komutlar içermektedir. Klavyeden giriş yapmanın komutu farklıdır, ekrana birşey yazdırmanın farklıdır, karar vermenin farklıdır. Bu komutlara ileride bahsedilecektir. Program aynı bir mektup gibi yukarıdan aşağıya doğru yazılır ve bilgisayar bu programı aynı sizin yazdığınız satır sırasına göre bunu çalıştırır. Yani yapmak istediklerinizi mantık ya da oluşturduğunuz algoritma sırasına göre komut satırlarını yazmanız gerekmektedir. Yukarıda bahsedilen örnekte ekrandan ilk önce vize notu istenmeli, daha sonra final notu istenmeli, işlem yapılmalı, karar verilmeli ve sonuç ekrana yazılmalıdır. Bu sıranın değişmesi kurgu hatasıdır ve istenmeyen bir durumdur, programın amacına ulaşması imkansıdır. Program yukarıdan aşağıya doğru komutları çalıştıra çalıştıra inmektedir. Peki en başa ya da istediğimiz bir noktaya geri dönmek mümkün olur mu? Evet bazı durumlarda bu akışın içinde bilgisayarı istediğimiz noktaya yönlendirmek mümkündür, bilgisayar bu noktaya gider ve buradan komut akışını aşağıya doğru işletmeye devam eder. Artık tüm satırlar biter çalıştırılacak komut kalmadığında ise program sonlanır ve otomatikman programdan çıkılır.

Programlama mantığının anlatıldığı bu notta BASIC programlama dili esas alınmıştır. Bu programlama dili eski olmasına karşın, anlaşılması basit ve günümüz

modern programlama dillerinin sahip olduğu hazır fonksiyonlar içermemektedir. Programlama mantığı tüm programalama dilleri için yaklaşık olarak aynıdır. Bir programalama dilinin öğrenilmesi diğer bir programlama dilinin öğrenilmesini ciddi ölçüde kolaylaştırmaktadır.

Tek komutluk ilk programımız şöyle olsun;

```
PRINT "Merhaba Dünya"
```

```
Merhaba Dünya
```

Yukarıdaki örnekte komut `PRINT "Merhaba Dünya"` dır sonucu ise kutu içine alınmış olan ve bilgisayarın ekranına yansıyacak olan **Merhaba Dünya** dır. Verilen örnekler bu şekilde aktarılacaktır. BASIC programlama dilinde ekrana herhangi birşey yazdırılmak istendiğinde PRINT komutu kullanılmaktadır. Her PRINT komutu ile yapılan yazdırılma işleminden sonra imleç aşağıya düşer. Eğer yeni bir satırdan ziyade yan yana yazılması isteniyorsa ilk yazdırılma işleminden sonra ; konulmalıdır.

```
PRINT "Merhaba"
PRINT "Dünya"
PRINT "Algoritmalar ve Bilgisayar ";
PRINT "Programlama"
PRINT "Ders Notları"
```

```
Merhaba
Dünya
Algoritma ve Bilgisayar Programlama
Ders Notları
```

PRINT <değer> ekrana istenilen bir bilgiyi yazdırmak için kullanılır

Nümerik, Alfanümerik Değişkenler ve Sabitler

Programlamada işlem ve hesap yapmak için sayılar ve harfler kullanılır. Sayılara *nümerik* değişkenler denilirken isim, adres gibi matematiksel işlem yapılamayacak değişkenlere ise *alfanümerik* değerler adı verilmektedir. Alfanümerik değerler sayı içeriyor olsalar bile bunlar ile matematiksel işlemler yapılamamaktadır. Alfanümerik değerler "" içinde bilgisayara verilirler. Bu tırnak işareti içindeki bilginin harflerden sayılardan oluşan ve matematiksel işlem yapılamayacak bir değer olduğunu göstermektedir. *Değişken* ise nümerik ya da alfanümerik değerlerin atandığı kelimelerdir ve her programlama dilinde kullanılmaktadır. Örneğin vize=67 demek vize değişkenine 67 nümerik değerini atamakta ve vize değişkeninin değeri bundan sonra değiştirilinceye kadar 67 olmaktadır. Başka bir örnek verecek olursak soyadi$="yılmaz", bu örnekte soyadi değişkeninin değeri yılmaz olarak atanmıştır, bilgisayara ne zaman soyadi$ nin değeri sorulsa size yılmaz değerini verecektir. Dikkat edecek olursanız alfanümerik değişkenler sonuna $ işareti almaktadır ve "" içinde ifade edilirler. sonuc$="37" örneğinde sizce 37 nin atandığı sonuc$ değişkeni nümerik midir, alfanümerik midir. Cevap alfanümeriktir, her nekadar sayı gibi görülse de $ ve "" den dolayı bilgisayar bunu sayı olarak değil bir isim gibi algılar ve matematiksel işlem yapamaz. Yukarıdaki örneklerde vize, soyadi ve sonuc değişkenlerinde dikkat edildiği üzere Türkçe karakterler kullanılmamıştır. Değişken isimleri özgürce seçilebilir. Ancak değişken isimleri verilirken bazı kurallar vardır. Değişken isimleri içinde kesinlikle Türkçe karakter ve noktalama işareti kullanılmamalıdır. Boşluk karakteri içermemelidir. "-" yerine "_" kullanılmalıdır. Sayı ile başlamamalıdır (örnek `2nciadi` kullanılamaz) ancak içinde sayı kullanılabilir (örnek `adi2nci` kullanılabilir). Değişkenlerin ne olduğu konusunda açıklık kazandırmak açısından bazı örnekler aşağıdadır. Her örneği anlamaya çalışınız.

```
PRINT "Deneme"
```

```
Deneme
```

```
PRINT 67
```

```
67
```

```
PRINT 2 + 4
```

```
6
```

```
vize = 40
PRINT vize
```

```
40
```

```
vize = 34
PRINT vize + 2
```

```
36
```

```
sayi1 = 3
sayi2 = 5
sayi3 = 7
PRINT sayi1 + sayi2 + sayi3
```

```
15
```

```
sayi1$ = "24"
sayi2$ = "85"
PRINT sayi1$ + sayi2$
```

```
2485
```

sonuca dikkat ediniz, değişkenler “” dolayı alfanümerik. Matematiksel bir toplama işleminden ziyade iki değişken birbirine uç uca eklendi.

```
vize = 50
final = 65
PRINT vize * 0.4 + final * 0.6
```

```
59
```

Kullanıcı Girişi

Yukarıdaki örnekler incelendiğinde tüm kullanılan değerlerin her bir örnek için sabit olduğu görülmektedir. Program içinde kullanıcının giriş yapılması istenmemekte her seferinde program içine sabit yazılmış olan sayılar için işlem yapılmaktadır. Bu örnekler derlenip çalışır program haline getirildiklerinde kullanıcıdan hiç bir şey istenmeden hepsi aynı sonuçları verecektir. Kullanıcıların program içinde sayıları ve değerleri değiştirmesi gibi birşey söz konusu değildir. Yukarıda yazılan örnekler program satırlarıdır, kullanıcı girişi yapılması istenmemiştir.

Program çalışırken kullanıcının giriş yapabilmesi için ne yapmak gerekmektedir?

INPUT "ne istendiğine dair açıklayıcı yazı"; <değişken> programın bu noktasında bilgisayar durur, ekrana komutun yanında "" içinde verilen yazıyı yazar ve kullanıcıdan bir değer girmesini ister. Giriş işlemi bittiğinde kullanıcı ENTER tuşuna basar ve girilen değer <değişken> ne atanır.

```
INPUT "vize notunuzu giriniz: ";vize
PRINT vize
```

```
vize notunuzu giriniz: 65 <65 kullanıcı tarafından girilmiştir
65
```

Dikkat edilecek olursa bir değişken sadece tek bir değeri tutabilmektedir. Değişkene yeni atanan değer eskisinin yerine geçmekte ve hafızadan atılmaktadır. Peki bir değişkene birden fazla değer atamanın yolu var mıdır. Evet. Aşağıdaki örnekleri inceleyiniz.

```
DIM boy(10)
boy(0) = 155
boy(1) = 175
boy(2) = 190
boy(3) = 165
hangisi = 2
PRINT boy(0)
PRINT boy(1)
PRINT boy(2)
PRINT boy(3)
PRINT boy(hangisi)
```

```
155
175
190
165
190
```

Örnekte boy adı altında bir değişken kullanılmıştır. Dikkat edilecek olursa programın ilk satırında **DIM** boy(10) diye bir komut bulunmaktadır. Bu komut boy değişkeninin 11 taneye kadar (0 dahil 0..10) değer alabileceğini bilgisayara bildirip bellekte rezervasyon yapmak için kullanılmaktadır. Bu tür çok sayıda değer alabilecek değişkenlerde DIM komutu ile bellekte rezervasyon yapılması zorunludur. Program tasarlanırken değişkenin kaç değer alacağı tahmini olarak bilinmeli ona göre rezervasyon yapılmalıdır. Program çalışırken değişkene ait az sayıda alan kullanılabilir, hepsi kullanılmak zorunda değildir, ancak rezervasyon sayısından fazla değişken atanamaz. Programın en son satırındaki **PRINT** boy(hangisi) komutu çok ilginç gözükmektedir. hangisi değişkeninin değeri programda daha önceden 2 olarak atanmıştır. Bu son satırdaki komut boy değişkenlerinden 2nci sıradaki gösterir.

Matematiksel işlemlerde kullanılan karakterler ve fonksiyonlar;
3*2 (çarpma), 3/2 (bölme), 3+2 (toplama), 3-2 (çıkarma), 3^2 (üslü işem) ve 3 mod 2 (bölmede kalan)

Matematik hesaplamalardaki işlem önceliği aynı şekilde programlama için de geçerlidir.
Formül içinde ilk önce çarpma ve bölme işlemleri yapılır daha sonra ise toplama ve çıkartma işlemleri. Çarpma ve bölme aynı önceliğe sahiptir, aynı şekilde toplama ve çıkartma da aynı önceliğe sahiptir. Eğer formül içinde önceliğe sahip olmadığı halde belli bir işleme öncelik tanınmak isteniyorsa ( ) içine alınmalıdır. ( ) içindeki işlemlerin önceliği vardır. Ondalıklı sayılarda ayraç olarak “.” kullanılır.

```
INPUT "vize notunuzu giriniz: ";vize
INPUT "final notunuzu giriniz: ";final
PRINT "notunuz"
PRINT vize * 0.4 + vize * 0.6
```

```
vize notunuzu giriniz: 65 <65 kullanıcı tarafından girilmiştir
final notunuzu giriniz: 55 <55 kullanıcı tarafından girilmiştir
notunuz
59
```

```
INPUT "Adınız nedir: ";ad$
PRINT "Merhaba “;
PRINT ad$;
PRINT " nasılsın"
```

```
Adınızı nedir: mehmet <mehmet kullanıcı tarafından girilmiştir
Merhaba mehmet nasılsın
```

Bazı durumlarda yazılan kodların arasına açıklayıcı bir satır yazılması gerekebilir. Bu yazılanlar kod değildir, sadece kodu yazana veya okuyana açıklayıcı bilgiler vermesi, neler yapması gerektiğini hatırlatması için yazılır. Bu açıklayıcı satırlar ***REM*** ile başlar ve bilgisayar tarafından program çalıştırılırken bu satırlar dikkate alınmaz. Sadece açıklama amacı ile kodların arasına eklenir.

REM <açıklayıcı bilgi> kod içine açıklayıcı bilgi yazmak için kullanılır

Tüm satırlar aşağı doğru işlenirken yazdırılan her şey yavaş yavaş ekranı doldurmaya başlayacaktır. Eğer çıkış ekranının temizlenmesi isteniyorsa ***CLS*** komutu kullanılır. Bu komut çıkış alınan ekrandaki tüm karakterleri temizler ve imleci en başa taşır. Bundan sonra yazılacak olanlar bu temiz sayfadan yazılmaya başlar.

CLS Çıkış ekranındaki tüm karakterleri temizler. Ekranı siler.

PRINT komutu her kulanıldığında imleci aşağı indirmekte (eğer komuttan sonra ; kullanıldıysa yanında beklemekte) ve çıktı ekranının düzgün bir şekilde kullanılmasına bazen mani olmaktadır. Ekranın herhangi bir noktasına bir şeyler yazmak istediğimizde imleci oraya taşımanın bir yolu vardır. ***LOCATE*** komutu. Bu komut imleç yanında verilen sütün ve satıra gider ve bekler, ekrana yapılacak olan herhangi bir yazdırılma işlemi bu noktadan itibaren başlar.

LOCATE X,Y İmleci ekranın X sütununun Y satırına götürür ve bundan sonra ekrana yapılacak yazdırılma işlemlerinin bu noktadan itibaren yapılmasını sağlar. Text ekranların ekran genişliği 80, satır sayısı da 25 karakterdir. Günümüz bilgisayarları artık bu text ekranlar yerine grafik ekranları kullanmaktadır.

Komutlar yukarıdan aşağıya doğru ilerler ve normalde her satırda bir komut yer almaktadır. Bir satırda birden fazla komut verilmek istenirse komutların arasına iki nokta üst üste ":" işareti konulmalıdır. Bu kullanım programın işleyişini kesinlikle bozmamaktadır ve bazı durumlarda programcıya yer sağladığı için tercih edilebilirler. Aşağıda buna bir örnek verilmiştir.

```
vize = 50 : final = 90
PRINT vize * 0.4 + final * 0.6
```

hatta şu şekilde de yazılabilir

```
vize = 50 : final = 90 : PRINT vize * 0.4 + final * 0.6
```

Koşullu işlemler

Programın belli bir noktada karar vermesi ve buna göre işlem yapması isteniyorsa kullanılan bir fonksiyondur. Normal koşullar altında programın yukarıdan aşağıya doğru komutları çalıştırdığını öğrenmiştik. Ancak belli bir koşul sağlandığı zaman programın bu yönü değiştirilebilir. Bir örnekle açıklamak gerekirse; öğrenci ortalama notunun 60 ve yukarısı olması durumunda öğrenci dersten geçmekte, aksi halde kalmaktadır. Dikkat edilecek olursa burada bir koşul vardır. Öğrenciye geçtiği mi yoksa kaldığı mı bildirilmesi gerekmektedir? Koşul şudur; notu 60'a eşit veya büyük mü? Bunun tek bir cevabı vardır. Evet veya hayır. Evet ise öğrenciye geçtiği, hayır ise kaldığı bildirilecektir. Komut şu şekildedir.

```
IF ortalamanot >= 60 THEN PRINT "Geçti" ELSE PRINT "Kaldı"
```

Görüldüğü üzere komut 3 kısımdan oluşmaktadır. İlk kısım ***IF*** den sonra yazılan koşul kısmıdır. Bu kısımda belirtilen koşulun cevabı evet ise ***THEN*** kısmı, hayır ise ***ELSE*** kısmı yapılmaktadır. Örnekte ortalamanot değişkeninin değerinin 72 olduğunu düşünürsek bu koşulun cevabı evet olacaktır ve ***THEN*** kısmı gerçekleştirilerek ekrana "Geçti" ibaresi yazılacaktır. Bu durumda ***ELSE*** kısmı atlanarak işlem görmez. ortalamanot değerinin 40 olduğunu düşündüğümüzde koşul hayır cevabını verecektir ve hayır cevabı verildiğinde işlem yapılan ***ELSE*** kısmı devreye girecektir, ekrana "Kaldı" ibaresi yazılacaktır. Bu durumda da ***THEN*** kısmı işlem görmeden atlanacaktır.

IF <koşul> **THEN** komut **ELSE** komut

koşul gerçekleşirse THEN den sonraki kısımda yer alan, koşul gerçekleşmezse ELSE den sonraki kısımda yer alan komut işlem görmektedir. Aşağıda koşullar verilmiştir;

`x>y`	x, y den büyükse
`x<y`	x, y den küçükse
`x=y`	x, y ye eşitse
`x>=`	x, y ye eşit veya büyükse
`x<=`	x, y ye eşit veya küçükse
`x<>y`	x, y den farklıysa

Bu tek satırlık bir komuttur, eğer ***THEN*** veya ***ELSE*** kısmında yazılacak komut sayısı fazla ilse aralarına ":" işareti konularak sorun çözülebilir. Ancak yine de bu alanlar çok sayıda komut yazılması için yeterli gelmiyorsa ***IF*** komutunun şu şekilde kullanımı da vardır.

```
IF <koşul> THEN
komut
komut
komut
komut
```

```
ELSE
komut
komut
komut
END IF
```

Görüldüğü üzere ***THEN*** veya ***ELSE*** kısmına çok sayıda komut bu şekilde eklenebilmektedir. Ancak ***END IF*** ile komutun sonlandığını bildirmek gerekmektedir. Bu durumda END IF konulması unutulmamalıdır. Tek satırlık ***IF*** satırında ***END IF*** kullanımı yoktur.

ELSE kısmının kullanımı opsiyoneldir. Yani kullanılması zorunlu değildir. Bu durumda eğer koşul sağlanıyorsa ***THEN*** kısmı yapılır, koşul sağlanmıyorsa hiç bir şey yapılmaz.

```
IF <koşul> THEN komut
```

Bu komutun kullanılması ile örnekler verelim

```
vize = 50 : final = 70 : sonuc = vize * 0.4 + final * 0.6
IF sonuc >= 60 THEN PRINT "Geçtiniz" ELSE PRINT "Kaldınız"
```

Yukarıdaki örnek ekrana “Geçtiniz” yazacaktır. Örneği biraz daha karıştıralım.

```
INPUT "Vize notunuzu giriniz(girmediyseniz boşgeçiniz)";vize
INPUT "Final notunuzu giriniz(girmediyseniz boşgeçiniz)";final
IF vize = 0 THEN PRINT "Vizeye girilmemiş"
IF final = 0 THEN PRINT "Finale girilmemiş"
sonuc = vize * 0.4 + final * 0.6
IF sonuc >= 60 THEN
          PRINT "Geçtiniz"
                ELSE
          PRINT “Kaldınız..!”
          gerekennot = (60 - (vize * 0.4)) / 0.6
          PRINT "Finalden "; gerekennot;" alsaydın geçerdin.”
          PRINT "Hoca sana "; gerekennot - final;
          PRINT "puan daha verseydi geçecektin.”
          PRINT "Bütünlemeden"; gerekennot;" ve üzeri al.”
                  END IF
```

Yukarıdaki örnek zor bir örnektir. Bilgisayara komutları yazarak denemeniz anlamanıza yardımcı olacaktır.

Normal koşullar altında bilgisayar programı çalışırken satırlar aşağı doğru akmaktadır. İstediğimiz bir noktadan program içinde istediğimiz başka bir noktaya yönlendirilmek istendiğinde ***GOTO*** komutu kullanılmaktadır. Bu komutun yanına gidilmesi gereken yer tarif edilmelidir. Bu da satır numaraları ile olmaktadır. Gidilmesi istenen yere bir satır numarası verilerek gidilecek yer işaretlenebilir.

```
PRINT "Merhaba"
GOTO 1
PRINT "Nasılsınız"
1 PRINT "İyi misiniz”
PRINT "ben iyiyim"
```

```
Merhaba
İyi misiniz
ben iyiyim
```

Örnekte görüldüğü üzere ekrana Nasılsınız yazmamaktadır çünkü Merhaba yazdıktan sonra bilgisayar 1 nolu satıra ***GOTO*** ile yönlendirilmekte ve program bu noktadan itibaren akmaya devam etmektedir.

GOTO X Program bu komutu gördüğü zaman normal akışından çıkar ve X ile belirtilen satır numarasındaki komuta gider ve buradan işlemeye devam eder.

Döngüsel işlemler

Döngüsel işlemler programcılıkta çok kullanılan fonksiyonlardandır. Aynı işlemin birden fazla kez yapılması için kullanılır. Örneğin adınızı aşağıya doğru 10 kere yazmak isterseniz normalde yazmanız gereken aşağıdaki gibi bir komut dizisidir.

```
PRINT "Hüseyin"
PRINT "Hüseyin"
PRINT "Hüseyin"
PRINT "Hüseyin"
PRINT "Hüseyin"
PRINT "Hüseyin"
PRINT "Hüseyin"
PRINT "Hüseyin"
PRINT "Hüseyin"
PRINT "Hüseyin"
```

Ancak bunun yarine PRINT "Hüseyin" komutunun 10 kere tekrarlanması bu garip durumu ortadan kaldırcaktır. Bu amaç için kullanılan komut ***FOR..NEXT*** döngüsüdür.

FOR sayaç= başlangıçdeğeri **TO** bitişdeğeri **STEP** adımdeğeri **.. NEXT**

FOR ve NEXT satırları arası sayaç değeri bitişdeğerine ulaşıncaya kadar tekrar edilir. Bu tekrar sırasında sayaç değeri adımdeğeri kadar arttırılır.

Yukarıdaki örneği bu komut ile yazacak olursak

```
FOR a = 1 TO 10
PRINT "Hüseyin"
NEXT
```

Bu komut dizilimi bir yukarıdaki komut dizilimi ile aynı çıktıyı verecektir. Çalışma sistemi ise şu şekildedir. Bilgisayar ***FOR..NEXT*** arasında döngüye girer ve ***FOR..NEXT*** arasındaki komut ya da komutları uygular. örnekte ***FOR***'un yanındaki "a" değişkendir ve değeri örnekte 1 dir, bu "a" değişkeninin başlangıç değerini vermektedir, ***TO*** dan sonradaki sayıya kadar (örnekte 10) "a" değeri 1 arttırılır, bu değere ulaşıldığında artık ***FOR..NEXT*** arası yapılmaz ve program ***NEXT*** ten sonraki satıra gider ve normal akışına devam eder. Başka bir örnek vererek durumu netleştirelim .

```
PRINT "Sayım başlıyor"
FOR sayac = 3 TO 7
```

```
PRINT sayac
NEXT
PRINT "Sayım bitti"
```

```
Sayım başlıyor
3
4
5
6
7
Sayım bitti
```

Örnek incelendiğinde ***FOR..NEXT*** döngüsünde sayaç değişkeninin değeri 3 ten 7 ye 1'ere basamak adımlarla arttırılmış ve döngü içinde sayaç değişkeninin değeri yazdırılmıştır. Normalde artış değeri 1 olarak bilgisayar tarafından sabit olarak alınır. Eğer artış değeri 1 den farklı ise bu ***FOR*** satırının sonuna ***STEP*** yazılarak verilmelidir. Aşağıdaki örneği inceleyiniz.

4 ten başlayarak 2şer 2şer 140'a kadar saymak istiyoruz. Yazacağımız döngü şu şekilde olacaktır.

```
FOR say = 2 TO 140 STEP 2
PRINT say
NEXT
```

Geriye doğru bir sayım söz konusu ise ***STEP*** kısmına - değer verilir. Normalde artış değeri 1 ise ***STEP*** kısmının yaılmasına gerek yoktur, ancak azalma olarak verilecek ise ve azalma değeri 1 olsa bile ***STEP*** kısmına -1 yazılmalıdır. Azalma değeri 3 olmasını istiyorsak ***STEP*** kısmına -3 yazılmalıdır. Aşağıdaki örnekte 10 dan 0 a 2şer 2şer düşülecektir.

```
FOR say = 10 TO 0 STEP -2
PRINT say
NEXT
```

Daha karmaşık bir işlemle ***FOR..NEXT*** döngüsüne örnek verelim. Örneğin sınıf mevcudu belli olmayan bir sınıfta boy ortalamasını almak isteyelim. Fakat kaç kişinin

boyunun girilmesi gerektiği sabit değildir. Yani program bize kaç kişinin boyunun hesaplanması gerektiğini sorması gerekmektedir. İlk önce kodu yazalım, daha sonra açıklayalım

```
10 INPUT "Sınıf mevcudu nedir";mevcut
20 toplamboy = 0
30 FOR say = 1 TO mevcut
40 PRINT say;
50 INPUT ". kişinin boyunu giriniz ";boy
60 toplamboy = toplamboy + boy
70 NEXT
80 PRINT "Sınıfın boy ortalaması ";
90 PRINT toplamboy / mevcut
```

Çok karışık bir örnek gibi gözükse de satır satır incelemeye başlayalım. İlk önce satırların başında 10nar 10nar artan satır numaralarından bahsedelim. Bu daha önce görmediğimiz birşeydi. Bunlara satır numaraları denilmektedir ve kullanılması zorunlu değildir. Komutların yerlerini belirlemek için kullanılmaktadır. Genelde 10nar 10nar arttırılır. Bu eğer araya bir satır eklenecek ise kolaylık sağlaması açısındandır. 10 ile 20 satır arasında isterseniz 15. satırı ekleyebilirsiniz ya da 17. satırı ekleyebilirsiniz. Satır numaralarını isterseniz 1'er 1'er artacak şekilde isterseniz 1000ner 1000ner artacak şekilde verebilirsiniz. Bu tamamiyle programcının tercihine bırakılmıştır. İsterseniz satır numarası yazmak zorunda da değilsiniz. Ancak satır numaralarının olması ileride de anlatılacağı üzere size bazı kolaylıklar sağlayacaktır. Yukarıdaki örnek programın açıklanmasında satır numaları baz alınacaktır. Örneğimize geri dönecek olursak. 10 nolu satırda sınıfın mevcudu istenmekte ve girilen değer mevcut değişkenine atanmaktadır. Dikkat ederseniz ***FOR..NEXT*** mevcut kadar dönecek (30. satır) ve bizden mevcut kadar sayısı girişi isteyecektir. Bu kod dizisi içinde alışık olmadığımız bir yapı daha göze çarpmaktadır. O da 60. satırdadır. Dikkat edecek olursanız toplamboy değeri 20. satırda 0 değerini alıyor. Ancak ***FOR..NEXT*** döngüsü içinde bu değerin üzerine toplana toplana geliyor. 60. satırın anlamı şudur toplamboy değişkeninin yeni değeri şu andaki değerine boy değişkenin eklenmiş halidir. Döngü her döndüğünde toplamboy değerine boy değeri eklenecek ve en sonda elde edilen toplamboy değeri girilen tüm boyların toplamı olacaktır. Bilindiği üzere ortalama boytoplamının mevcuda bölünmesi ile bulunmaktadır. 90. satırda toplamboy / mevcut ortalamayı hesaplamakta ve yazmaktadır. Yukarıdaki örnekle ister 10 kişinin ister 10000 kişinin boy ortalaması alınabilmektedir. Görüldüğü üzere döngünün ne kadar döneceği mevcut ile bizden girilmesi istenmektedir.

Dikkat edilecek olursa program içerisindeki bu döngü başta belirlenmiş bir koşul kadar tekrar edilmektedir. Döngü içine girildiğinde ne kadar döneceği bellidir. Fakat bazı döngüler vardır ki belirli koşullar sağlandığında döngüden çıkılmakta ve program satırlarında istenilen bir noktaya gidilebilmektedir. Bu şu an için anlaşılması zor bir konudur. "Koşullu döngüler ve yönlendirmeler" başlığı altında anlatılmıştır.

Dikkat edilirse ***FOR-NEXT*** döngüsünün kaç kere döneceği programcı tarafından belirlenen bir durumdur. Buna benzer bir döngü daha vardır ***DO WHILE-LOOP*** döngüsü. Bu döngüde ise belirlenen bir koşul gerçekleştiği sürece döngü sonsuza kadar dönmektedir. Ne zaman koşul gerçekleşmezse döngüden çıkılır ve programın akışına devam eder. Bir örnekle açıklarsak

```
a = 0
DO WHILE a < 5
   a = a + 1
   PRINT a
LOOP
```

```
1
2
3
4
5
```

Örnekte ***DO WHILE-LOOP*** arası **a** değişkeni 5 ten küçük olduğu sürece sonsuza kadar dönecektir. Ancak dikkat edilecek olursa **a** değişkeninin değeri döngü içinde 1 arttırılmaktadır. Bu nedenle değişkenin değeri 5 ten büyük hale geldiğinde artık koşul sağlanamamakta ve döngüden çıkılmaktadır. Bu komut dikkatli kulanılmalıdır, çünkü eğer döngünün gerçekleşmesini sağlayan koşul (örnekte **a < 5**) döngü içinde değiştirilmediği sürece kısır döngü gerçekleşecek ve bilgisayaranız kilitlecektir. Dikkat edilecek olursa bu döngünün kaç kere döneceğini koşulun gerçeklemesi etkilemektedir.

Matematiksel fonksiyonlar

BASIC programlama dilinde bir çok matematiksel fonksiyon otomatik olarak tanımlıdır. Bu fonksiyonların kullanımı ise oldukça kolaydır. Basit bir örnek verecek olursak;

```
a = INT(3.99)
b = SQR(36)
PRINT a
PRINT b
```

```
3
6
```

Diğer fonksiyonların kullanımı da örnektekine benzemektedir. Aşağıda sıklıkla kullanılan matematiksel fonksiyonlar listelenmiştir.

INT(*n*)	Sayının tam sayı kısmını verir, INT(3.89) = 3
SQR(*n*)	Sayının karekökünü verir, SQR(36) = 6
ABS(*n*)	Sayının mutlak değerini verir, ABS(-5) = 5
ATN(*n*)	Sayının arctanjantını verir, ATN(5) = 1.37340077
TAN(*n*)	Sayının tanjantını verir, TAN(5) = -3.38051501
COS(*n*)	Sayının kosinüsünü verir, COS(45) = 0.52532199
SIN(*n*)	Sayının sinüsünü verir, SIN(45) = 0.85090352
LOG(*n*)	Sayının logaritmasını verir, LOG(120) = 4.78749174

Yazıya ve arka alana renk verme

`Color` komutu bu işe yaramaktadır. Kullanımı `Color x,y` şeklindedir. `x` yazının rengini `y` ise arka alanın rengini belirlemek için kullanılır. Bu komut bir kere uygulandığı anda tekrar değiştirilinceye kadar aynı özellikte yazılar yazmanıza yardımcı olur. `x` ve `y` değişkenleri 0 ile 15 arasında değer alabilir ve farklı renkleri göstermektedir. Eğer yazının aynı zamanda yanıp sönmesi isteniyorsa `x` değerine 16 sayısı eklenmelidir (Arka alanın [`y`] yanıp sönme özelliği yoktur).

```
Color 15,3
PRINT "Deneme"
```

yukarıdaki örnek yazının rengini 15 arka alanın rengini 3 yapar ve Deneme yazar. Bundan sonra yazılacaklar da aynı özellikte olacaktır. Hangi sayıların hangi renklere denk geldiğini deneyerek bulabilirsiniz.

Alfanümerik fonksiyonlar

Matematiksel fonksiyonlara benzer olarak alfanümerik değişkenler için de bazı fonksiyonlar bulunmaktadır. Bu fonksiyonlar sayesinde alfanümerik değişkenler içinden bazı karakterleri alıp çıkarmak oldukça kolaydır. Örneğin

```
ad$ = "Mehmet"
a$ = LEFT$(ad$,2)
b$ = RIGHT$(ad$,3)
c$ = MID$ (ad$,2,3)
d = LEN(ad$)
PRINT ad$
PRINT a$
PRINT b$
PRINT c$
PRINT d
```

```
Mehmet
Me
met
ehm
6
```

Aşağıda alfanümerik fonksiyonların açıklaması verilmiştir.

LEN(*n$*) Alfasayısal dizinin uzunluğunu verir, LEN(“Mehmet”)=6

LEFT$(*n$,x*) Alfasayısal dizinin solundan *x* kadarını verir, LEFT$(“ali”,2)=“al”

RIGHT$(*n$,x*) Alfasayısal dizinin sağından *x* kadarını verir, RIGHT$(“ali”,2)=“li”

MID$(*n$,x,y*) Alfasayısal dizinin *x* inci sırasından sonra *y* kadar karakteri verir, MID$(“merhaba”,3,4)=“rhab”

Çeviri fonksiyonları

BASIC programlama dilinde bazı çevi fonksiyonları da mevcuttur. Nümerik bir sayının alfanümeriğe, alfanümerik bir değerin nümeriğe çevirilmesi oldukça kolaydır. Aşağıdaki örneği inceleyiniz. Ancak incelerken değişkenin nümerik mi alfanümerik mi olduğuna dikkat ediniz

```
no$ = "35"              <35 sayı gibi gözükse de aslında alfanümeriktir
yenino = VAL(no$)       <değişkenin nümerik olduğuna dikkat ediniz
PRINT no$
PRINT yenino
PRINT no$ + no$
PRINT yenino + yenino
```

```
35      <sayı gibi gözükse de sayı değildir
35      <sayı olarak 35
3535    <iki 35 yan yana eklendiği için böyle gözükmektedir
70      <sayısal olarak iki 35 toplanmıştır
```

Yukarıdaki örnekte alfanümerik bir değişken nümerik bir değere ***VAL*** fonksiyonu ile çevirilmiştir. Bu fonksiyonun tam tersi olarak nümerik bir değer alfanümeriğe de çevirilebilmektedir. Kullanımı yukarıdakine benzemektedir ancak komut ***STR$*** dır.

STR$(*n*) Sayıyı alfasayısala çevirir, STR$(100)=“100”
VAL(*n$*) Alfasayısalı sayıya çevirir, VAL(“100”)=100

Bilgisayarda kullanılan her bir karakterin aslında bir sayısal değeri vardır.Küçük büyük harfler, sayılar, noktalama işaretleri, boşluk karakteri, diğer ülkelere ait karakter ve diğer karakterler toplam 256 adettir. 0-31 arası kontrol karakterleridir ve sadece bilgisayarın kullanımına açıktır. Bu karakterlerin kullanımı programda bazı sıkıntılara neden olabilmektedir. 32-127 arası sıklıklar kullandığımız harfler, sayılar, noktalama işaretlerini içermektedir. 128-255 arası ise bazı uluslararası karakterler ve şekiller içermektedir (Bkz ek ASCII tablosu). İşte bu karakterler ve ASCII tablosundaki sayısal değerleri birbirine çevirilebilmektedir.

```
FOR say = 65 TO 90
PRINT CHR$(say);
NEXT
```

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```
<ASCII tablosunu inceleyiniz

Bu işlemin tersi yani karakterden sayısal değerini de elde etmek mümkündür. Bunun için kullanılan komut ise ***ASC*** dir. Örneğin ASC("a") nın değeri sayısal olan 97 dir.

CHR$(*n*) Sayının ASCII tablosundaki karakter karşılığını verir, CHR$(97)="a"

ASC(*n$*) Bir karakterin ASCII tablosundaki yerini verir, ASC("a")=97

ASCII Tablosu

0		32		64	@	96	`	128	Ç	160	á	192	└	224	Ó
1	☺	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
2	☻	34	"	66	B	98	b	130	é	162	ó	194	┬	226	Ô
3	♥	35	#	67	C	99	c	131	â	163	ú	195	├	227	Ò
4	♦	36	$	68	D	100	d	132	ä	164	ñ	196	─	228	õ
5	♣	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	Õ
6	♠	38	&	70	F	102	f	134	å	166	ª	198	ã	230	µ
7	•	39	'	71	G	103	g	135	ç	167	º	199	Ã	231	Þ
8	◘	40	(	72	H	104	h	136	ê	168	¿	200	╚	232	þ
9	○	41	)	73	I	105	i	137	ë	169	®	201	╔	233	Ú
10	◙	42	*	74	J	106	j	138	è	170	¬	202	╩	234	Û
11	♂	43	+	75	K	107	k	139	ï	171	½	203	╦	235	Ù
12	♀	44	,	76	L	108	l	140	î	172	¼	204	╠	236	ý
13	♪	45	_	77	M	109	m	141	ì	173	¡	205	═	237	Ý
14	♫	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	¯
15	☼	47	/	79	O	111	o	143	Å	175	»	207	¤	239	´
16	►	48	0	80	P	112	p	144	É	176	░	208	δ	240	-
17	◄	49	1	81	Q	113	q	145	æ	177	▒	209	Ð	241	±
18	↕	50	2	82	R	114	r	146	Æ	178	▓	210	Ê	242	=
19	‼	51	3	83	S	115	s	147	ô	179	│	211	Ë	243	¾
20	¶	52	4	84	T	116	t	148	ö	180	┤	212	È	244	¶
21	§	53	5	85	U	117	u	149	ò	181	Á	213	ı	245	§
22	▬	54	6	86	V	118	v	150	û	182	Â	214	Í	246	÷
23	↨	55	7	87	W	119	w	151	ù	183	À	215	Î	247	¸
24	↑	56	8	88	X	120	x	152	ÿ	184	©	216	Ï	248	°
25	↓	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	¨
26	→	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
27	←	59	;	91	[	123	{	155	ø	187	╗	219	█	251	¹
28	∟	60	<	92	\	124	¦	156	£	188	╝	220	▄	252	³
29	↔	61	=	93	]	125	}	157	Ø	189	¢	221	¦	253	²
30	▲	62	>	94	^	126	~	158	×	190	¥	222	Ì	254	■
31	▼	63	?	95	_	127	⌂	159	ƒ	191	┐	223	▀	255	