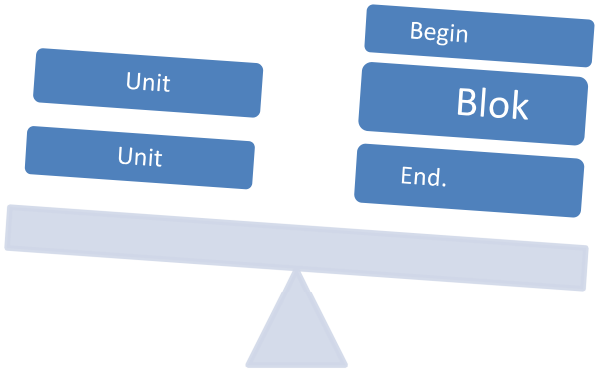
Pascal 7.0

ile Programlama



Prof.Dr. Ali ORAL

Balıkesir Üniversitesi

Makina Mühendisliği Bölümü

Pascal programlama ders notları, Balıkesir Üniversitesinde 2000'li yılların başına kadar Endüstri Mühendisliği Bölümü, Necatibey Eğitim Fakültesi Bilgisayar Öğretmenliği ve Fen Edebiyat Fakültesi Matematik bölümlerinde verdiğim Pascal Programlama dersleri için oluşturulmuştur.

İnternet sayfama da eklediğim ders notları özgürce paylaşılabilir ancak kısmen de olsa **fotokopi ve benzeri yöntemlerle çoğaltılamaz ve asla satılamaz**. Ticari amaç ile kullanıldığının belirlenmesi halinde yasal yollara başvurulacaktır. Bu ders notları, internet sitelerinde ders notlarının bulunduğu internet sayfama köprü verilerek yayınlanabilir.

Okuyuculara yararlı olması dileği ile...

Doç.Dr. Ali ORAL

http://www.alioral.net
aoral@alioral.net

Ocak 2012

Uğruna çaba sarfettiğim herşeyi başaramadım.

Başardıklarım ise uğruna çaba sarfettiklerimdendir.

İÇİNDEKİLER

BÖLÜM 9

BÖLÜM 10

BÖLÜM 11

BÖLÜM 15

BÖLÜM 16

BÖLÜM 1

BILGISAYAR PROGRAMLAMAYA GIRIS

Insanlar her zaman düsünür ve problem çözerler. Bir çok problem, az yada hiç düsünülmeden çözülebilir.

Her gün evden çikarken ne giyilmelidir? Bunun için muhtemelen pencereden disariya bakilir. Hava yagmurlu ise mevsim gereklerine göre giyinmenin yani sira disariya çikarken bir de semsiye alinmasi gerekir. Hava günesli ve sicak ise o taktirde daha ince giyinilerek disariya çikilir. Böylece problemin çözümü kendiliginden olusturulan bir kararla saglanir.

Yukaridaki basit örnekte yapilan is, önce problemin belirlenmesi ve sonra problemin tanimindan yola çikarak çözüm için degisik alternatiflerin degerlendirilmesidir.

Bilgisayar programlamasi sirasinda izlenebilecek bir çok yol ve yöntem vardir. Bilgisayar programcisinin probleme iliskin çözümü ortaya çikarabilmesi için problem çözümü ile ilgili bilgileri bilmesi gerekir. Bilgisayar programlamasinda genel olarak belirli kalip ve kurallara uyulur. Bir bilgisayar yaziliminin olusturulmasi sirasinda asagida siralanan ana adimlara uyulur.

* Problemin tanimi
* Çözüm yönteminin belirlenmesi
* Programin kodlanmasi
* Programin çalisir duruma getirilmesi
* Belgeleme ve güncellestirme

Problemin Tanimi: Problemin normal yazi diliyle tanimlanmasi islemlerini kapsamaktadir. Problem çözümüne iliskin iyi bir program yapabilmek için, problemin iyi bir sekilde tanimlanmasi gerekir.

Çözüm Yönteminin Belirlenmesi: Bu adimda çözümün genel yaklasimi, temel giris/çikislar belirlenir ve problem çözümü adim adim program akis diyagramlariyla gösterilir.

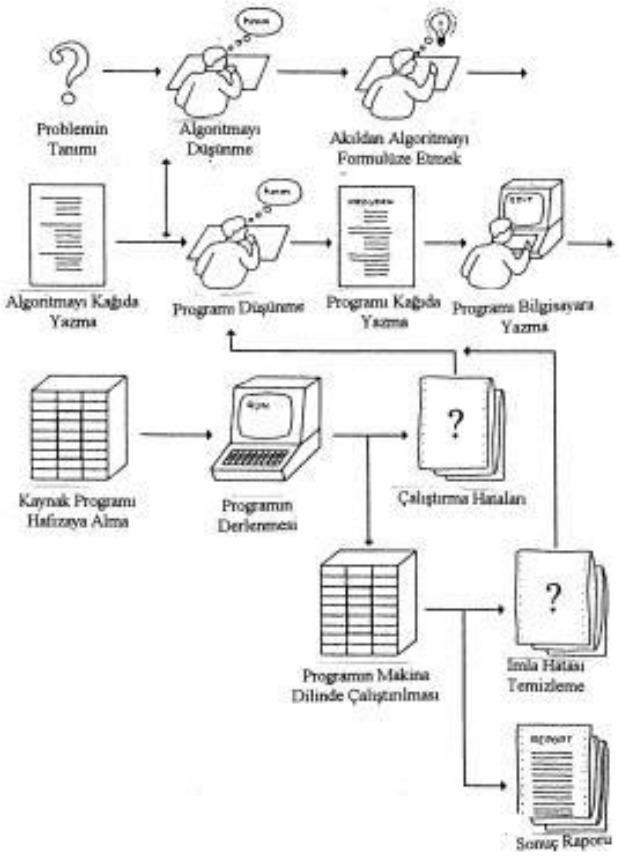
Programin Kodlanmasi: Program ayrintili olarak tanimlanip çözüm yollari açikça belirtildikten sonra program kodlama çalismalarina baslanabilir. Programin bastan sona yapisal bir düzende hazirlanmasi ve uygun bir programlama dili seçilmesi seçim isleminin ilk asamasini olusturur.

Programin Çalisir Hale Getirilmesi: Programin kodlanmasi sirasinda yapilan imla hatalari, kodlama ve mantik hatalarinin giderilmesi islemlerini kapsar. Iyi bir bilgisayar programinin dogrulugundan emin olmak için defalarca test edilmis olmasi gerekmektedir.

Belgeleme ve Güncellestirme: Olusturulan bir yazilimi, sadece o yazilimi gelistiren kisilerin kullanabilmesi gibi bir kisitlamanin önüne geçmek için ayrintili referanslar hazirlanmali ve programla ilgili bilgiler verilmelidir.

Bir yazilimda, o yazilimi kullanan kisi veya kuruluslarin yeni gereksinimleri ve degisen kosullar nedeniyle degisiklikler yapilmasi gerekli olabilir. Bu degisikliklere güncelleme adi verilir. Iyi bir programda bulunmasi gereken özellikler arasinda güncellesebilme ön siralarda yer almaktadir.

Sekil 1.1 Yazilim olusturma evrelerini sematik olarak özetlemektedir.



Sekil 1.1 Yazilim olusturma evreleri

1.1 Algoritma Kurma

Algoritma, verilen herhangi bir sorunun çözümüne ulasmak için uygulanmasi gerekli adimlarin hiç bir yoruma yer vermeksizin açik, düzenli ve sirali bir sekilde söz ve yazi ile ifadesidir. Algoritmayi olusturan adimlar özellikle basit ve açik olarak siralandirilmalidir. Algoritmik çözüm yöntemlerine ilk örnegi günlük yasantimizdan verelim.

Örnek 1: Örnegimiz bir insanin evden çikip ise giderken izleyecegi yolu ve isyerine girisinde ilk yapacaklarini adim adim tanimlamaktadir.

Çözüm 1:

Evden disariya çik
Otobüs duragina yürü
Durakta gidecegin yöndeki otobüsü bekle
Otobüsün geldiginde otobüse bin
Biletini bilet kumbarasina at
Inecegin yere yakinlastiginda arkaya yürü

Inecegini belirten ikaz lambasina bas
Otobüs durunca in
Isyerine dogru yürü
Is yeri giris kapisindan içeriye gir
Mesai arkadaslarinla selamlas
Is giysini giy
Isini yapmaya basla.

Yukaridaki örnekte görüldügü gibi, evden ise gidiste yapilabilecek islemler adim adim sirasiyla, kisa ve açik olarak tanimlanmaya çalisilmistir. Yukaridaki algoritma kisinin otobüsü kaçirma olasiligi düsünülmeden olusturulmustur. Kisi duraga geldiginde binecegi otobüsü kaçirmis ise algoritmamiz asagidaki sekilde degistirilebilir.

Çözüm 2:

Evden disariya çik
Otobüs duragina yürü
Otobüsün saati geçmis ?
Durakta gidecegin yöndeki bir sonraki otobüsü bekle
Bir sonraki otobüs gelene kadar 4. adimi uygula
Otobüsün geldiginde otobüse bin
Biletini bilet kumbarasina at
Inecegin yere yakinlastiginda arkaya yürü
Inecegini belirten ikaz lambasina bas
Otobüs durunca in
Isyerine dogru yürü
Is yeri giris kapisindan içeriye gir
Mesai arkadaslarinla selamlas
Is giysini giy
Isini yapmaya basla.

Her iki örnekte görüldügü gibi sorunu çözüme götürebilmek için gerekli olan adimlar sirali ve açik bir biçimde belirlenmistir. Algoritmanin herhangi bir adimindaki küçük bir yanlislik dogru çözüme ulasmayi engelleyebilir. Bu nedenle algoritma hazirlandiktan sonra dikkatle incelenmeli ve varsa adimlardaki yanlisliklar düzeltilmelidir.

Programlamanin temeli olan algoritma hazirlanmasinda dikkat çekici bir nokta, ayni sorunu çözmek için hazirlanabilecek olasi algoritma sayisinin birden çok olmasidir. Baska deyisle, bir sorunun çözümü için birbirinden farkli birden fazla sayida algoritma hazirlanabilir. Bu da gösteriyor ki herhangi bir problemin çözümü için birbirinden farkli yüzlerce bilgisayar programi yazilabilir.

Bir bilgisayar programi için hazirlanacak olan algoritma da ayni sekilde çözüm yolunu bilmeyen bir kisiye, çözüme ulasmak için neler yapmasi gerektigi anlatiliyormus gibi hazirlanmali ve eksik bir nokta birakmaksizin gerekli tüm adimlari açik ve düzenli olarak içermelidir. Çözüm için kullanilacak bilgilerin nereden alinacagi, nerede saklanacagi ve çözümün program kullanicisina nasil ulastirilacagi algoritma adimlari arasinda belirtilmelidir.

Asagida degisik islemlere iliskin algoritma örnekleri verilmistir.

Örnek 2: Iki sayiyi toplamak için gerekli programa ait algoritmanin olusturulmasi.

Algoritma:

A1 :Birinci sayiyi gir
A2 :Ikinci sayiyi gir
A3 :Iki sayinin toplamini yap
A4 :Toplamin degerini yaz
A5 :Bitir.

Bu tam bir algoritmadir. Sözcüklerin ortaya çikaracagi yanlis anlamalarin ortadan kaldirmak amaciyla semboller ve matematik dilini gerektiren bazi kisaltmalar kullanmak daha uygun olacaktir. Bir algoritma yazilirken su metot izlenmelidir:

Programda kullanilacak elemanlari temsil etmek üzere uygun isimler veya degiskenler seç.
Bazi isimlere baslangiç degeri olarak çözümün gerektirdigi uygun degerler ver.
Gerekirse programa girilecek verileri düzenle.
Cebirsel notasyon ve kararlar kullanarak aritmetik islemleri gerçeklestir.
Çikisi düzenle.
Bitir.

Yukarida iki sayinin toplanmasi için olusturdugumuz algoritmayi bu yeni gereksinimlere uyarak yeniden yazalim.

Toplam adi için **Z**
Birinci Sayi için **X**
Ikinci Sayi için **Y** degerleri kullanilirsa;

Algoritma:

A1 :X degerini gir
A2 :Y degerini gir
A3 :Z ? X+Y
A4 :Z' yi yaz
A5 :Bitir

Görüldügü üzere bu sekilde bir algoritma ile çözüm yolunu izlemek daha kolaydir. Bundan sonra verilen örneklerde bu tip algoritma kullanilacaktir.

Örnek 3: Iki sayinin ortalamasini bulan programa ait algoritmanin olusturulmasi.

Algoritma:

A1 :X degerini gir
A2 :Y degerini gir
A3 :Z ? X+Y
A4 :Ort? Z/2
A5 :Ort degerini yaz
A6 :Bitir

Bu örnekte Ort degeri ile iki sayinin ortalamasi temsil edilmistir.

Örnek 4: Bes sayinin toplamini ve ortalamasini veren programa ait algoritmanin olusturulmasi

Toplam adi için **Top**
Ortalama adi için **Ort**
Girilen sayilar için **X**
Arttirma için **Sayac** kullanilirsa

Algoritma:

A1 :Top ? 0, Sayac ? 0
A2 :X'i gir
A3 :Top? Top+X
A4 : Sayac ? Sayac +1
A5 :Eger Sayac <5 ise A2'ye git
A6 :Ort? Top/5
A7 :Top ve Ort degerlerini yaz
A8 :Bitir

Örnek 5: Kenar uzunluklari verilen dikdörtgenin alan hesabini yapan programa ait algoritmanin hazirlanmasi. Kenar uzunluklari negatif olarak girildigi durumda veri girisi tekrarlanacaktir.

Dikdörtgenin kisa kenari : a
Dikdörtgenin uzun kenari : b
Dikdörtgenin alani : Alan

Algoritma:

A1 :a degerini gir
A2 :a<0 ise 1. adimi tekrarla
A3 :b degerini gir
A4 : b<0 ise 3. adimi tekrarla
A5 :Alan ? a*b
A6 :Alan degerini yaz
A7 :Bitir

Örnek 6: Çapraz döviz kuru hesabi yapan programin algoritmasinin olusturulmasi. Bu algoritmanin olusumunda veriler; 1 Amerikan dolarinin TL karsiligi, hesaplanacak $ miktari, çikis ise verilen $'in TL karsiligi olacaktir.

Dolarin degeri :Doldeg
Girilen Dolar Miktari :Dolar
TL karsiligi :Tlkar

Algoritma:

A1 :Doldeg'i gir
A2 :Doldeg<0 ise 1. adimi tekrarla
A3 :Dolar'i gir
A4 :Dolar<0 ise 3.adimi tekrarla
A5 :Tlkar? Doldeg*Dolar
A6 :Tlkar degerini yaz
A7 :Bitir

Örnek 7: Verilen bir sayinin faktöriyelini hesaplayan programin algoritmasinin olusturulmasi

Sayinin faktöriyeli :Fak
Faktöriyel degiskeni :X
Faktöriyeli hesaplanacak sayi :Y

Algoritma:

A1 :Fak? 1, X? 0
A2 :Y'i gir
A3 :Y<0 ise 2. adimi tekrarla
A4 :X? X+1
A5 :Fak? Fak*X
A6 :X<Y ise 4. adima geri dön
A7 :Fak degerini yaz
A8 :Bitir

Bu algoritmada 1. adimda X 'e 0 ve Fak degiskenine 1 degeri ataniyor. 2. adimda Y degeri giriliyor ve 3. adimda Y degerinin 0 dan küçük bir deger olup olmadigi denetlenerek, sonuca göre gerekli komut veriliyor. 4. adimda X'in degeri 1 arttiriliyor ve 5. adimda X için Fak degeri hesaplaniyor. 6. adimda X in degerinin faktöriyeli hesaplanacak sayidan küçük olmasi durumunda 4. adimdan itibaren islemlerin tekrarlanmasi komutu veriliyor, X' in degerinin Y'ye esit olmasi durumunda islemler tamamlanarak hesaplanan degerin yazdirilmasi isleminden sonra programin çalismasi sona ermektedir.

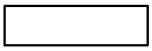
1.2 Akis Diyagramlari

Gelistirilecek olan yazilimin genel yapisinin sematik gösterimine akis diyagrami adi verilir. Akis dyagramlari, yazilimi olusturacak program parçalarini ve bu parçalarin birbirleri ile olan iliskilerini belirler. Bir bilgisayar programinin olusturulmasinda akis diyagramlarinin hazirlanmasi, algoritma olusturma asamasindan sonra gelmektedir. Bilgisayar programinin olusturulmasi sirasinda algoritma asamasi atlanarak, dogrudan akis diyagramlarinin hazirlanmasina baslanabilir. Programlama tekniginde önemli ölçüde yol almis kisiler bu asamayi da atlayarak direkt olarak programin yazimina geçebilirler.

Akis diyagramlarinin algoritmadan farki, adimlarin simgeler seklinde kutular içinde yazilmis olmasi ve adimlar arasindaki iliskilerin (is akisi) oklar ile gösterilmesidir.

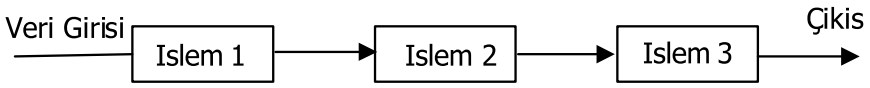
Akis diyagramlarinda kullanilan semboller, anlamlari ve kullanis amaçlari asagidaki tabloda verilmistir.

Tablo 1. Is akis diyagramlarinda kullanilan temel semboller ve anlamlari

Simge	Simgenin Adi	Simgenin Anlami
	Elips	Akis diyagraminin baslangiç ve bitis yerlerini gösterir. Baslangiç simgesinden çikis oku vardir. Bitis simgesinde giris oku vardir.
	**Paralel Kenar**:	Programa veri girisi ve programdan elde edilen sonuçlarin çikis islemlerini gösterir.
	Dikdörtgen	Aritmetik islemler ve degisik atama islemlerinin temsil edilmesi için kullanilir.
	Eskenar Dörtgen	Bir karar verme islemini temsil eder.
	Altigen	Program içinde belirli bloklarin ard arda tekrar edilecegini gösterir.
	**Oklar**	Diyagramin akis yönünü gösterir.
	**Alt Program**	Procedure ve Function Alt programlari gösterir

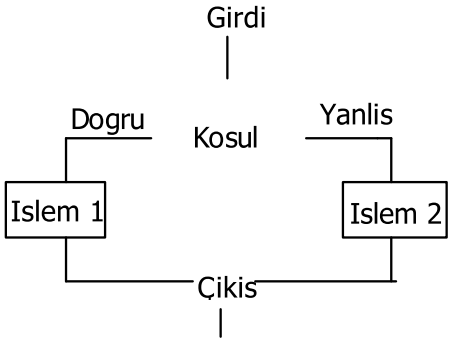
Ayrintili bir akis diyagrami, yazilimi olusturan islemleri ve iliskilerini en küçük detayina kadar belirler.

Bir bilgisayar programinin gelistirilmesinde kullanilan programlama dili ne olursa olsun bu programlarin akis diyagramlarinda genel olarak yalniz üç basit mantiksal yapi kullanilir. Bu mantiksal yapilardan en basiti sirali yapidir. Sirali yapi, hazirlanacak programdaki her islemin mantik sirasina göre nerede yer almasi gerektigini vurgular. Bu yapi sona erinceye kadar ikinci bir islem baslayamaz.



Sekil 1.2 Sirali Yapi

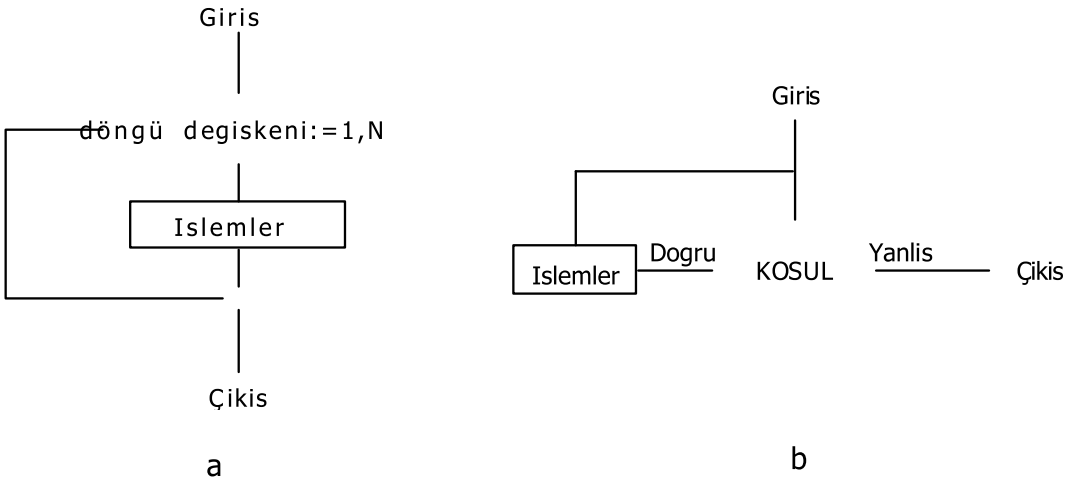
Mantiksal yapilardan ikincisi Karar Verme yapisidir (Sekil 1.3). Programlama sirasinda If...Then... Else yapisi ile taniyacagimiz bu mantiksal yapilar, birden fazla sirali yapi seçenegini kapsayan modüllerde, hangi sartlarda hangi sirali yapinin seçilecegini belirler.



Sekil 1.3 Karar Verme Yapisi

Üçüncü mantiksal yapi çesidini tekrarli yapilar olusturmaktadir. Bu yapilara Pascal programlama dilinde For (Sekil 1.4.a), While ve Repeat..Until (Sekil 1.4.b), yapisi adi da verilir. Sartlara göre degisik islem gruplarinin yapilmasini saglar. Bu yapi yukarida sözü edilen iki yapinin çesitli kombinezonlarin tekrarlanmasindan olusmustur.

Söz konusu üç degisik yapi, degisik kombinezonlarda kullanilarak istenilen islevleri yerine getirecek programlar hazirlanabilir. Programlarin bu üç basit yapi ile sinirlandirilmasi program modüllerinin daha kolay tasarlanmasini saglar.

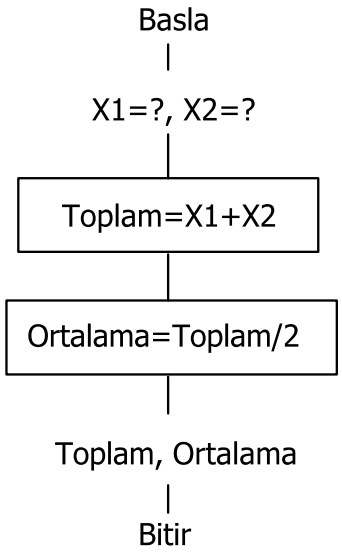


Sekil 1.4. Tekrarli Yapilar

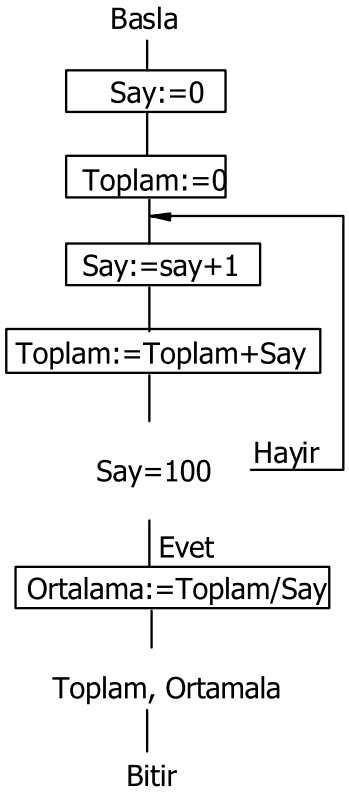
1.3. Akis Diyagrami Örnekleri

Bu bölümde, sözlü veya yazili olarak olusturdugumuz algoritmanin programa dönüstürülmesi sirasinda programin çalisma sirasini da gösteren akis diyagramlariyla ilgili örnekler asagida verilmistir.

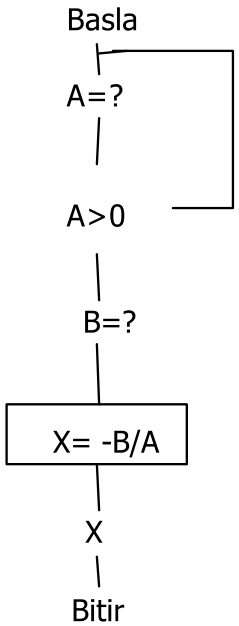
Örnek 1: Iki sayinin toplamini ve ortalamasini yapan bilgisayar programinin akis diyagramini çiziniz.



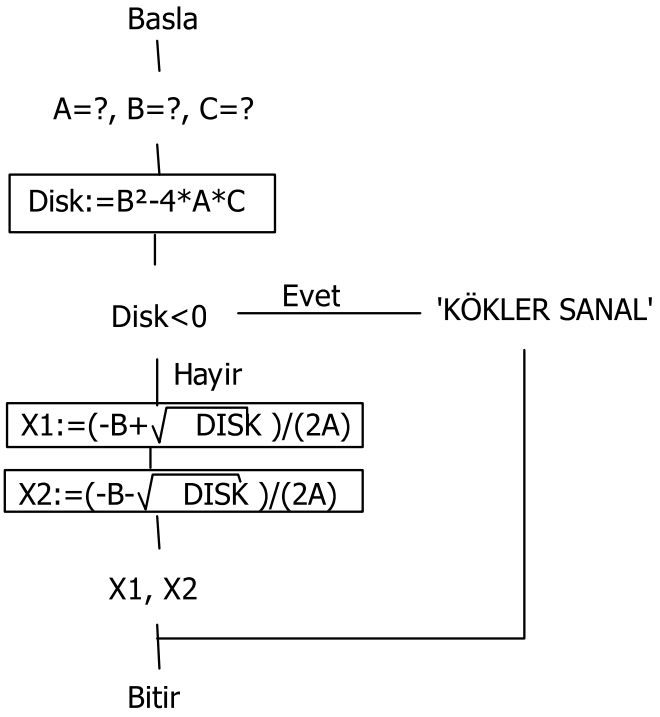
Örnek 2: 1'den 100'e kadar olan sayilarin toplamlarini ve ortalamalarini veren programin akis diyagramini çiziniz.



Örnek 3: Ax+b=0 seklinde verilen 1.derece denklemin çözümünü veren programa ait akis diyagramini çiziniz.

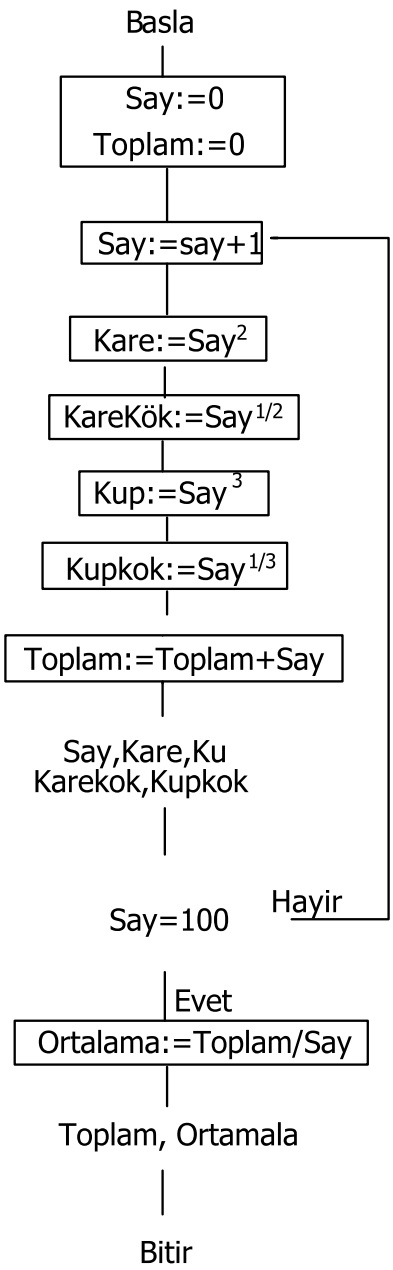


Örnek 4: $Ax^2+Bx+C=0$ seklinde verilen 2. derece denklemin köklerini bulan programin akis diyagramini çiziniz.

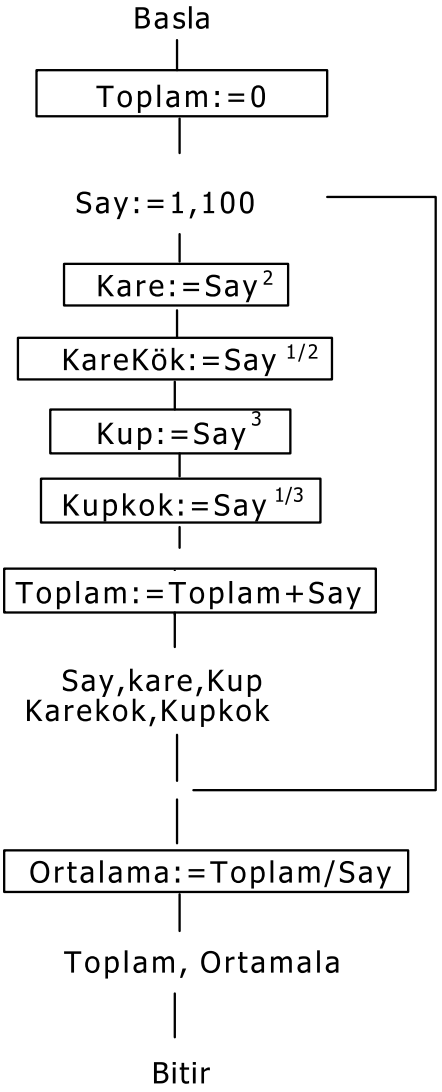


Yukaridaki örnekte A=0 girilmesi durumunda denklem 1.derece olmaktadir. Bu durumu dikkate alarak gerekli çözümü de gösterecek sekilde akis diyagramini degistiriniz.

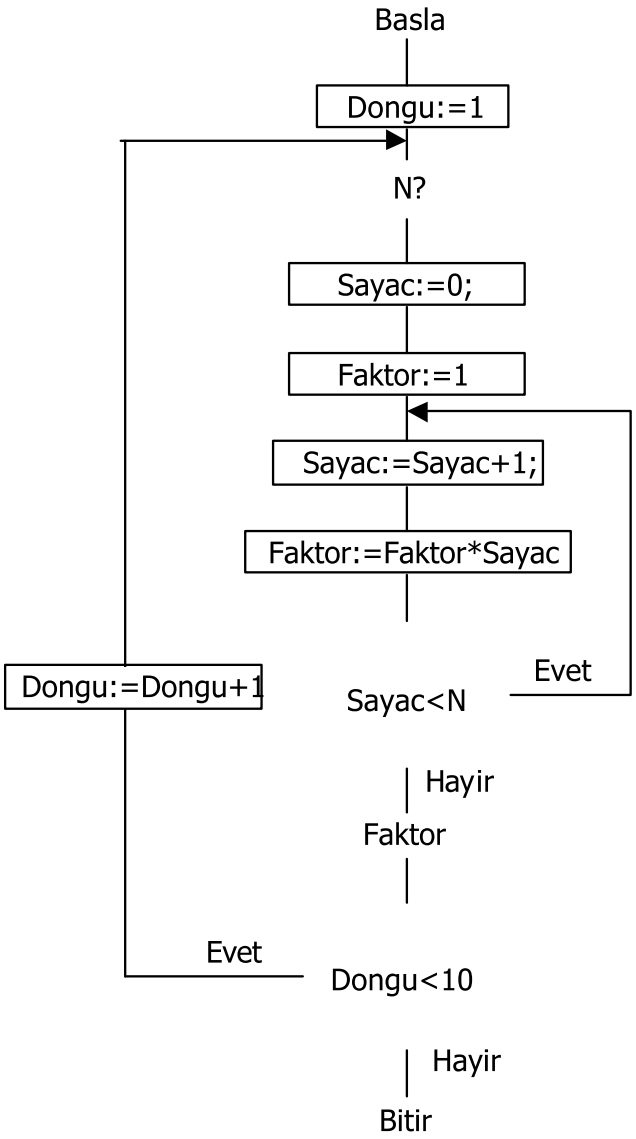
Örnek 5: 1' den 100'e kadar sayilarin karelerini, kareköklerini, küplerini, küpköklerini toplamlarini ve ortalamalarini veren programin akis diyagramini çiziniz.



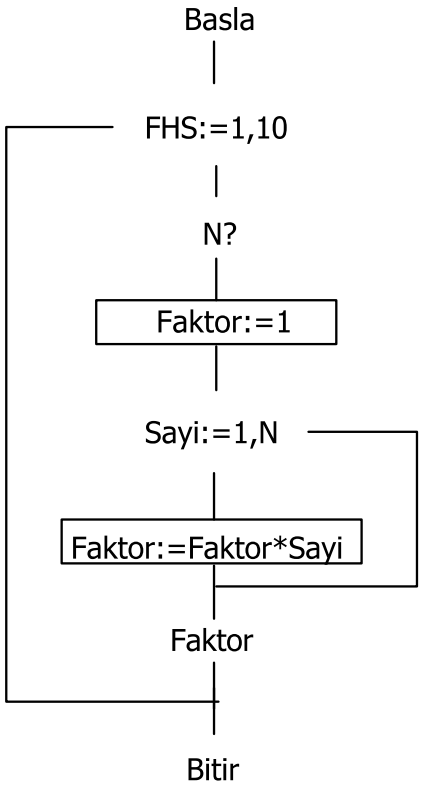
Örnek 6: Yukarida karar mantigi yapisi ile çözümledigimiz problemi döngü yapisi ile çözümleyelim.



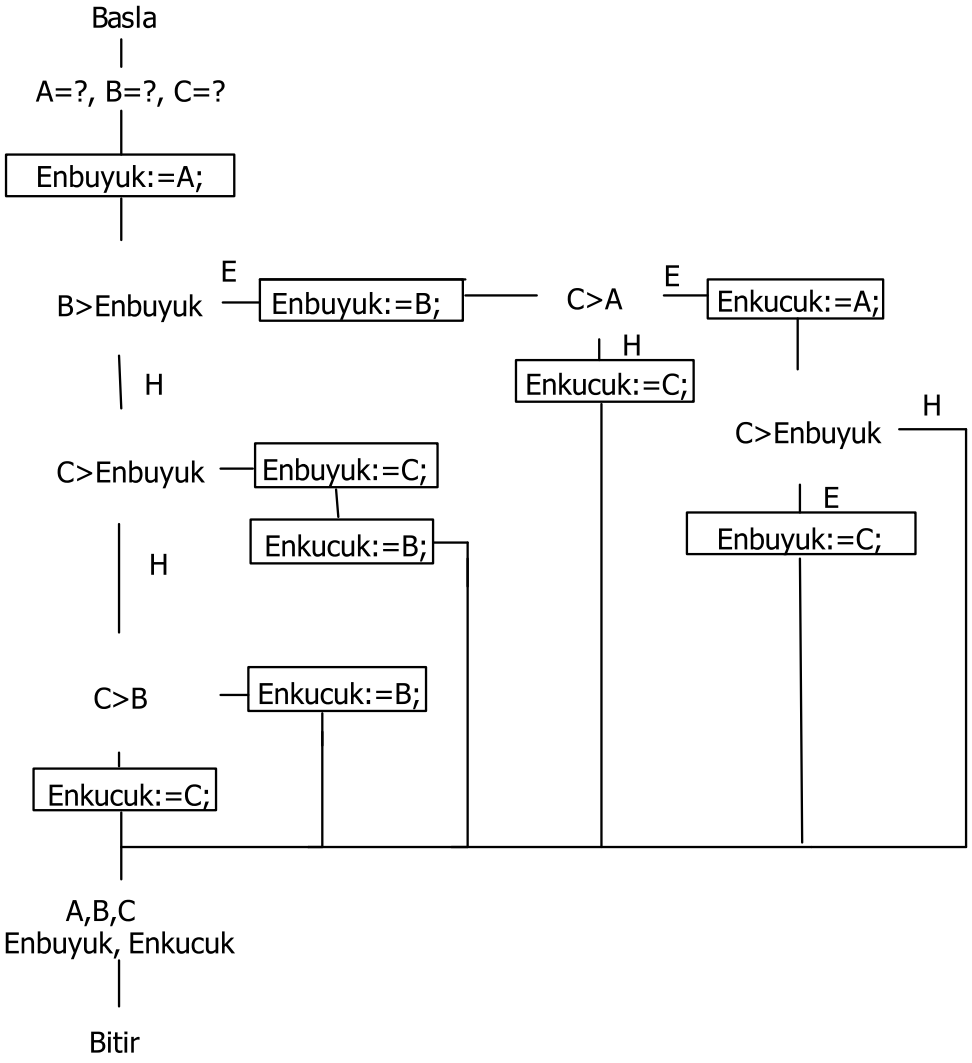
Örnek 7: 10 tane N sayisinin faktöriyelini hesaplayan programin akis diyagramini çiziniz.



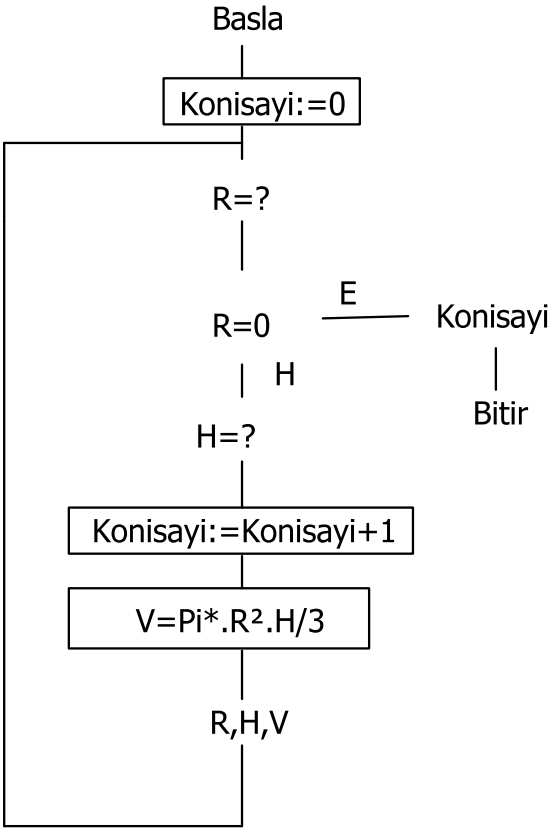
Örnek 8: Yukaridaki örnegi tekrarli yapi olarak tanimladigimiz döngü yapisi ile çözelim.



Örnek 9: Elimizde bulunan A,B, ve C gibi 3 adet sayidan en büyügünü ve en küçügünü bulan programin akis diyagramini çiziniz.



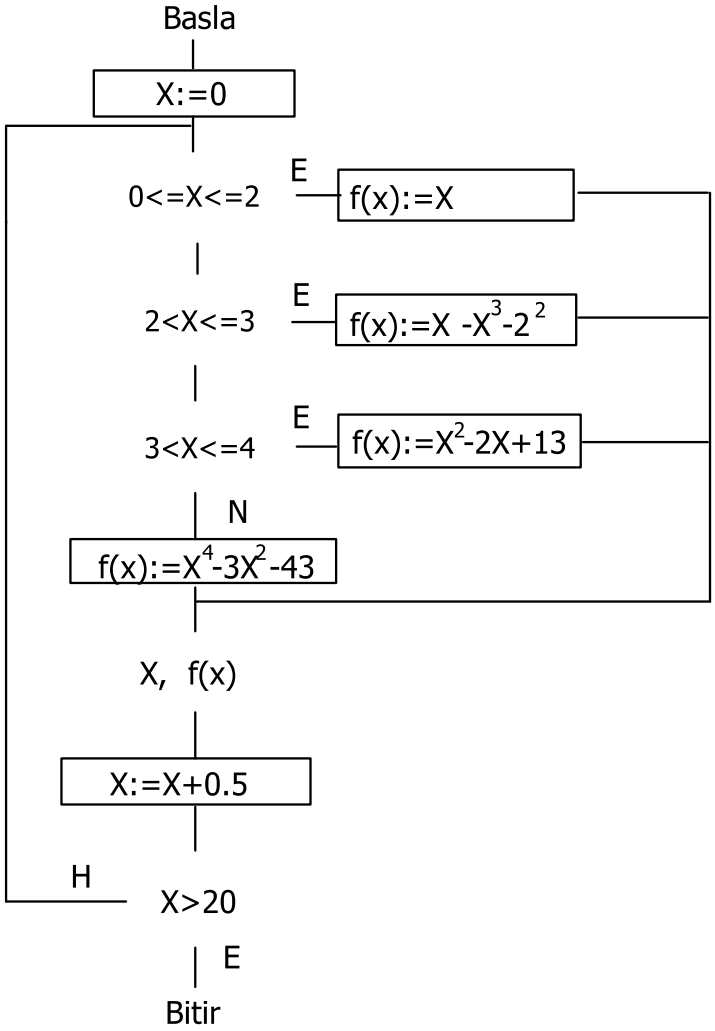
Örnek 10: Elimizde bilinmeyen sayida koni bulunmaktadir ve koniye ait yariçap (R) ve yükseklik (H) degerleri klavyeden girilmek suretiyle V=? *R².H/3 förmülü ile hacim hesabi yapilacaktir. Koniye ait yariçap degeri 0 girildiginde programin çalismasi duracaktir. Programda her girilen veri için hacim degeri rapor edilecek, programin çalismasi bittiginde toplam kaç koni için hacim hesabi yapildigi da belirtilecektir.



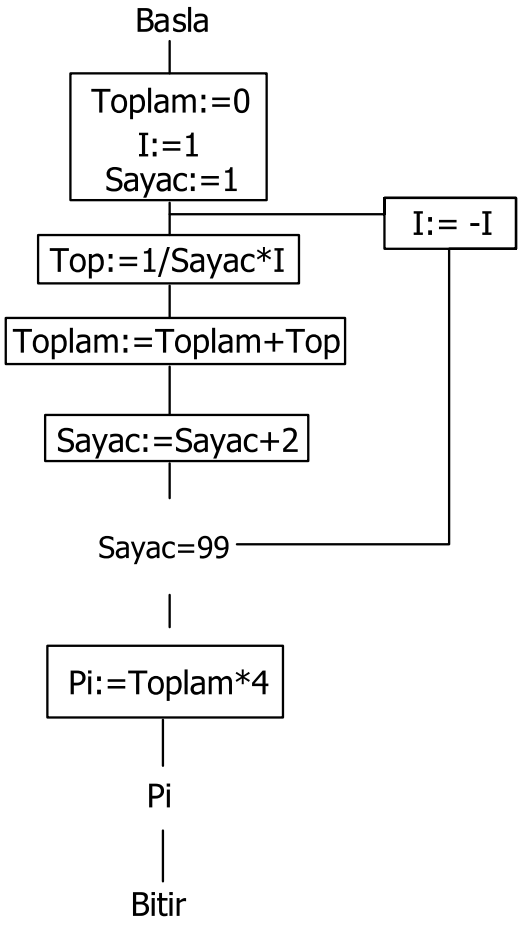
Verilen akis diyagraminda **R** ve **H** degerlerinin negatif girilmesine engel olacak sekilde akis diyagramini yeniden düzenleyiniz.

Örnek 11: F(x) kesikli fonksiyonunun degeri X'in aldigi degerlere göre asagida verilmistir. X'in degeri 0-10 arasinda 0.5 araliklarla arttigina göre her bir X degeri için F(x) fonksiyonunu hesaplayan programin akis diyagramini çiziniz.

0? X ? 2	f(x)= X
2< X ? 3	f(x)=X^3-X^2-2
3< X ?4	f(x)=X^2-2X+13
4< X	f(x)=X^4-3X^2-43



Örnek 12: Pi sayisinin formülü $((1 ? \frac{1}{3} + \frac{1}{5} ? \frac{1}{7} ? \frac{1}{9} ? \ldots).4)$ olduguna göre serinin paydasindaki ifade 99 oluncaya kadar pi sayisini hesaplayan bilgisayar programinin akis diyagramini çiziniz.



BÖLÜM 2

PASCAL EDİTÖRÜ

2.1 Giriş

Pascal programlama dili 1968 yılında Niklaus Wirth tarafından geliştirilmiş üst düzey programlama dilidir. Pascal programlama dilinin günümüzdeki sürümleri Turbo/Borland ve Windows Pascal adları ile bilinmektedir. Turbo Pascal programlama dili mühendislik problemlerinin çözümlerinde, bilimsel projelerde, sağladığı grafik desteği ve program yazmadaki kolaylıklarıyla aranılan bir dil olma özelliğini sürdürmektedir.

Pascal programlama dilinin önemli özelliklerinden biri, program yazmadaki kolaylıkların yanısıra, yazılan bir metinin kolaylıkla değiştirilebilmesi, program içindeki bir metinin istenilen yere taşınabilmesi, kopyalanabilmesi, yazılan programdaki yazım kurallarının kolaylıkla kontrol edilebilmesi, hataların kolaylıkla tespiti ve bu hataların düzeltilmesi için yaptığı öneriler vb. gibi işlemlerin çabuk ve güvenilir bir şekilde yapılabilmesine olanak sağlamasıdır.

Pascal'ın programcıya sunduğu önemli özelliklerden biri de; bazı programlarda ortak olarak kullanılan program parçalarının ayrı bir Pascal dosyası olarak saklanması suretiyle farklı programlarda bu program parçalarının kullanılabilmelerine olanak tanımasıdır.

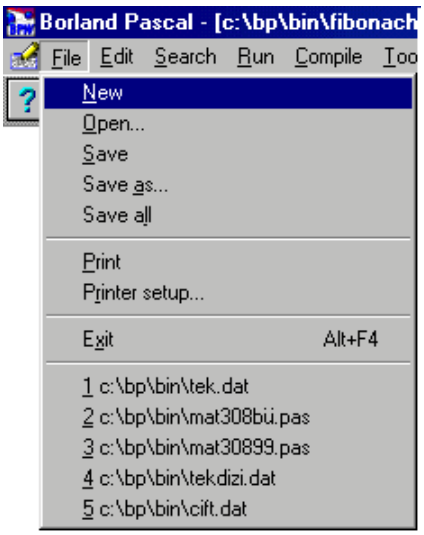
2.2 Pascal Menüleri

Pascal editöründe kullanımı kolaylaştıran bir çok menü vardır. Bu menülere ulaşmak için "Alt " tuşu ile birlikte menü isimlerinin ilk harflerine basmak yeterlidir. Menüye ulaştıktan sonra menü komutlarına erişim için üç yol vardır. 1)Mouse ile, 2)Ok tuşlarıyla 3) menü komutu üzerindeki işaretli harfe basılır.

Burada Pascal menülerinin tamamını tanıtmayıp sıklıkla kullanılan menü ve menü komutlarının kullanımı anlatılacaktır. Burada tanımı yapılan menüler, Windows Pascal için ifade edilmekle olup, aralarındaki küçük farklarla Turbo ve Borland Pascal 7.0 için de geçerlidir.

2.2.1 File menüsü

Pascal file menüsü üzerinde 10 ayrı menü komutu vardır. Bunların görevleri aşağıda özetlenmiştir.



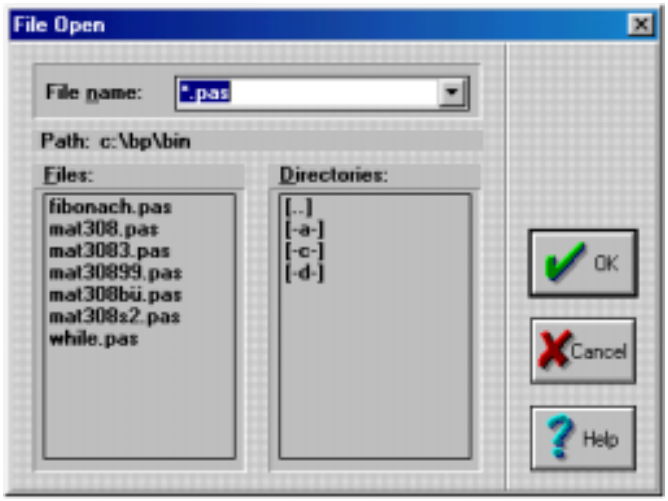
Şekil 2.1. Turbo Pascal File Menüsü

Open: Daha önceden diskte varolan Pascal program dosyasının editöre yüklenmesi için kullanılır. Pascalın DOS sürümü ile çalışırken aynı yöntem geçerli olmakla birlikte, aynı işlemi, F3 tuşuna basarak da yapabiliriz. Bu menü komutu kullanıldığında Şekil 2.2'de verilen diyalog kutusu görülecektir.

Disk üzerinde bulunan bir Pascal dosyasını editöre yüklemek için diyalog kutusunda görülen ***.PAS** yazılı yere programın adı yazılarak "Enter" tuşuna basılır veya önce tab tuşuna bastıktan sonra ok tuşları ile programın adı üzerine gelinerek "Enter" tuşuna basılır. Aynı seçim, mouse ile açılmak istenilen programın üzerine gelip iki kez sol tuşu tıklayarak da gerçekleştirilir.

New: Editörde yeni bir Pascal dosyası yaratmak amacıyla kullanılır.

Save: Yazılan programın hafızaya alınması işlemini gerçekleştirir. Program ilk defa hafızaya alınacak ise Şekil 3'de görülen *Save As Diyalog Kutusu* ekrana gelerek, programa bir isim verilmesini bekleyecektir. Programın hafızaya alınması için program adı yazıldıktan sonra mouse ile "ok" tuşuna veya klavyeden "Enter" tuşuna basılması gerekir. Daha önceden diskte bulunan bir Pascal programı üzerinde yapılan değişiklikler nedeniyle hafızaya alma işlemi için bu komut kullanıldığı zaman, program eski adı ile kaydedileceğinden aynı diyalog kutusu ekranda görülmez.

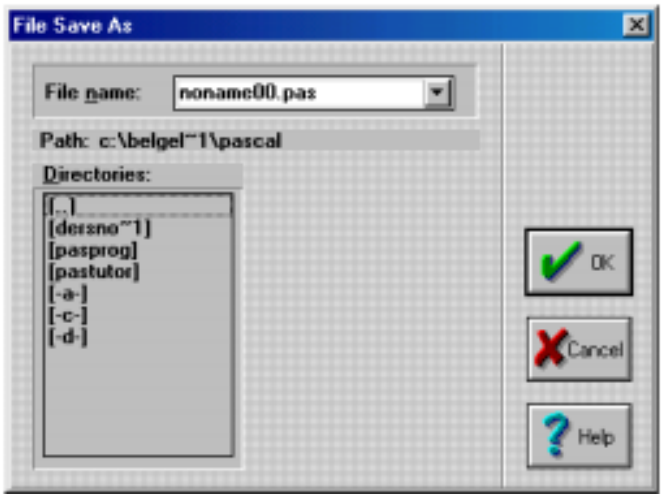


Şekil 2.2. Open Diyalog Kutusu

Pascalın DOS sürümü ile çalışırken yazılan programın hafızaya alınması için bu menü komutu ile aynı işleve sahip olan **F2** tuşuna basılması yeterlidir.

Save As: Editörde aktif penceredeki Pascal programının yeni bir isimle kaydedilmesi amacıyla kullanılır. Komutun kullanılmasıyla ekrana Save As Diyalog Kutusu gelir. Programa verilecek yeni isim yazılarak "Enter" tuşuna basılır.

Save All: Pascal ile program yazarken, birden fazla program penceresi ard arda açılabilir. Komut kullanıldığında açık olan bütün Pascal programları bilinen adları ile



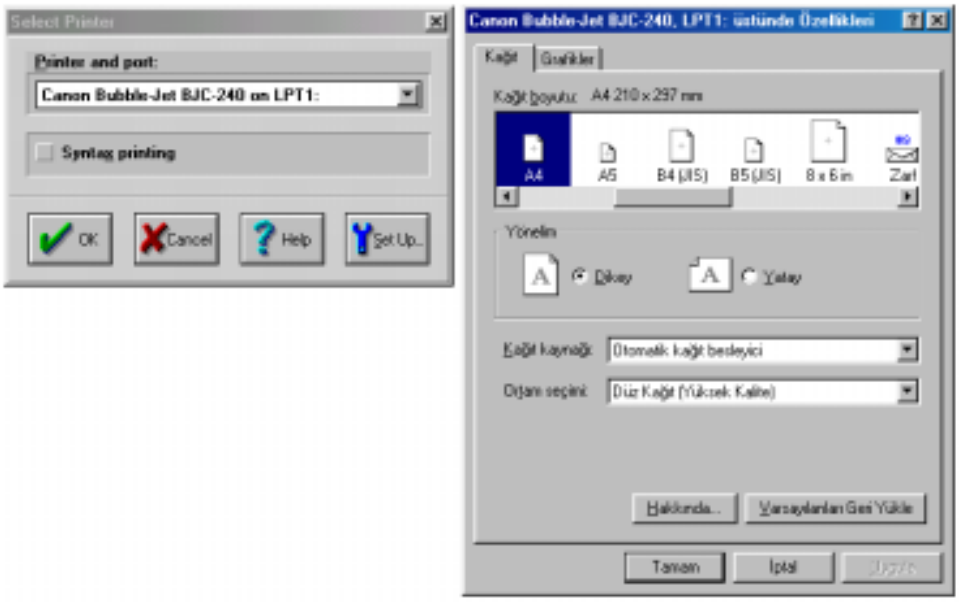
Şekil 2.3. File Save As Diyalog Kutusu

hafızaya alınır. Komutun kullanılması için açık pencerelerdeki programlara daha önceden isim verilmiş olmalıdır.

Change Dir...: Sadece Dos sürümü Turbo/Borland Pascalda bulunan bu komut, aktif çalışma sürücüsü veya dizininin değiştirilmesi amacıyla kullanılır. Komut çalıştırıldığında istenilen dizin veya sürücü seçilerek "Alt+K" tuş kombinasyonuna basılır veya tab tuşu ile "Ok" butonuna gelinerek "Enter" tuşuna basılır.

Print: Aktif Pascal penceresindeki program listesinin yazıcıdan alınmasını sağlar.

Printer Setup : Printer seçimi ve printer ayarlamalarının yapılması için kullanılır. Komut kullanıldığında Şekil 2.4 'te verilen diyalog kutusu ekrana gelir. Printer ve port bölümünden yazıcı seçimi yapılır. Set Up.. tuşuna basıldığında yazıcı ayarlarında istenilen değişikliklerin yapılmasını sağlayan diğer bir diyalog kutusu ekrana gelecektir. Burada kağıt boyutu, kağıdın yazıcıya yerleştirilme şekli, kağıt kaynağı (el ile, üst tepsi, otomatik vb), kullanılacak kağıt kalitesi gibi değişkenlerden istenilenleri seçmemize olanak sağlanmıştır.



Şekil 2.4 Printer Setup Diyalog kutusu

Get Info: Sadece Dos sürümü Turbo/Borland Pascal 'da bulunan bu komut, program ile ilgili bilgileri içeren bir diyalog kutusu ekrana gelir. Amacı programın ile ilgili çeşitli bilgilerin programcıya sunulmasıdır.

Dos Shell: Sadece Dos sürümü Turbo/Borland Pascalda bulunan bu komut, geçici olarak Pascal editörünün terkedilip Dos ortamına dönülmesi amacıyla kullanılır. Dos ortamından tekrar editöre dönülmesi için Dos promptunda EXIT yazılır.

Exit: Pascal 'dan çıkış için kullanılır.

2.2.2 Edit Menüsü

Edit menüsü (Şekil 2.5) yardımıyla program içinde belli blokların silinmesi, başka yere taşınması, kopyalanması gibi işlemleri yapabilmek mümkündür.

Şekil 2.5. Edit Menüsü

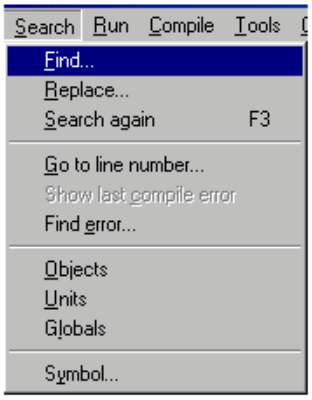
Yukarıda sözü edilen işlemlerin yapılabilmesi için öncelikle üzerinde işlem yapılacak program bloklarının Shift ve Ok tuşlarına aynı anda basılması ile işaretlenmesi gerekmektedir. Metinin işaretlenmesi işleminden sonra edit menüsü üzerinde görülen menü komutları çalıştırılır.

Cut: İşaretli metinin silinmesi amacıyla kullanılır. Kısa yol tuşu (Shift+Del).

Copy: İşaretli metini panoya (clipboard) kopyalar. Bu şekilde panoya kopyalanan metinin istenilen yere kopyalanması işlemi için bu komuttan sonra aşağıda tanımlanan *Paste* komutunun kullanılması gerekmektedir. Kısa yol tuşu (Ctrl+Ins).
Paste: Copy komutu ile panoya kopyalanan metinin editör üzerinde istenilen yere kopyalanabilmesini sağlar. Kısa yol tuşu (Shift+Ins).

2.2.3 Search Menüsü

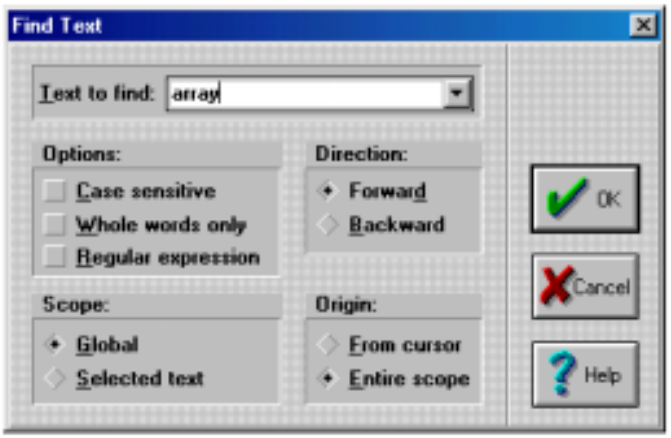
Search Menüsü (Şekil 2.6), program yazımı sırasında belirli sözcüklerin aranması ve değiştirilmesi, istenilen program satırına hızlı bir şekilde ulaşılması vb. gibi işlemlerin kolaylıkla yapılabilmesini sağlar.



Şekil 2.6. Search Menüsü

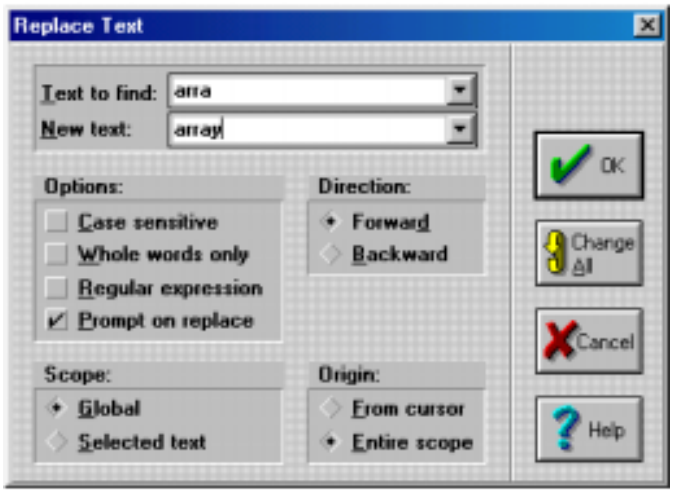
Find: Bu komut program içinde herhangi bir metinin aranması amacıyla kullanılır. Komut aktif hale geldiğinde Şekil 2.7'deki diyalog kutusu ekrana gelir. Aranılacak metin, kendisi için ayrılmış alana yazılarak klavyeden "Enter" tuşuna veya mouse ile Ok butonuna basılır.

Bulunulan noktadan ileriye doğru arama yapılacak ise **Direction** bölümünde *Forward* terimi işaretlenmeli, geriye doğru arama yapılacak ise *Backward* terimi işaretlenmelidir.



Şekil 2.7. Find Diyalog kutusu

Scope bölümünde, sözcüğün tüm dosya içinde aranılması isteniyorsa *global*, sadece seçilen metinde aranılması isteniyorsa *selected text* bölümleri işaretlenmelidir. Origin bölümünde, aramaya başlanacak sözcük, nokta bütün dosya içinde aranılacak ise *entire*



Şekil 2.8. Replace Diyalog kutusu

scope, imleçin bulunduğu noktadan itibaren aranılacak ise *from kursor* bölümleri seçilmelidir.

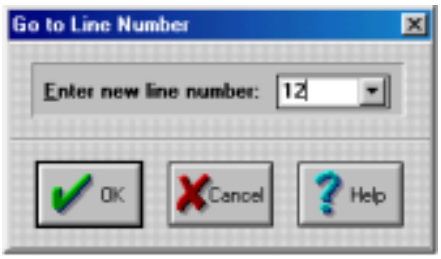
Replace:Yazılan program içinde kullanılan herhangi değişkenlerin veya Pascal sözcüklerinin başka bir değişken veya Pascal sözcüğü ile değiştirilmesi gerekebilir. Bu komut kullanıldığında Şekil 2.8'deki diyalog kutusu ekrana gelir. *Text to find*' 'in karşısında ayrılmış alana değiştirilmesi istenilen sözcük, *New Text*' in karşısındaki ayrılmış alana yeni sözcük yazılır. Değiştirilecek sözcük programın tamamında veya birden çok yerinde değiştirilecek ise **Change all** tuşunun üzerine gelinir ve klavyeden "Enter", mouse ile Ok butonuna basılır. Programda aranılan metin bulunduğunda değişim için kullanıcıdan onay bekleyen bir diyalog kutusu ekrana gelir. Değişime onay için "Y", değişime hayır demek için "N" tuşuna basılır.

Search Again: Find menü komutu ile arama yapıldığında, program içinde aranılan metine ilk karşılaşılan yerde arama kesilir. Aynı metini tekrar aramak için *Search Again* menü komutu kullanılır.

Goto Line Number: Program içinde istenilen program satırına hızlı bir şekilde ulaşılmasını sağlar (Şekil 2.9). Komutun kullanımıyla ekrana gelen diyalog kutusunda ayrılmış alana satır numarası yazılarak "Enter" tuşuna basılır.

2.2.4 Run Menüsü

Run menüsünde yazılan programın çalıştırılması için bazı komutlar bulunmaktadır. Run menüsünde bulunan menü komutları aşağıda özetlenmiştir.



Şekil 2.9. Goto Line Number Diyalog kutusu

Run: Programı derleyerek çalıştırır. Kısa yol tuşu (Ctrl+F9).

Program Reset: Sadece DOS sürümü Turbo/Borland Pascal'de bulunan bu komut ile Step Over veya Trace Into komutlarından biri çalıştırılınca, yapılan kontrolü durdurmak ve hata kontrolünü tekrar programın ilk deyimi üzerine almak için kullanılır. Kısayol tuşu (Ctrl+F2).

Goto Cursor: Sadece DOS sürümü Turbo/Borland Pascal'de bulunan bu komut ile editör ekranındaki program yazılımında kursörün bulunduğu satıra kadar olan program bölümünün derlenerek çalıştırılmasını sağlar. Kısayol tuşu (Ctrl+F4).

Trace Into: Sadece DOS sürümü Turbo/ Borland Pascal'de bulunan bu komut ile editör ekranındaki programın satır satır çalıştırılmasını sağlar. Kısayol tuşu (F7).

Step Over: Sadece DOS sürümü Turbo/ Borland Pascal'de bulunan bu komut ile editör Trace On durumuna geçer. Aktif durumdaki programın çalışması sırasında programın hangi satırları izlediğini ve işlem sırasını izlemek amacıyla kullanılmaktadır. Kısayol tuşu (F8.)

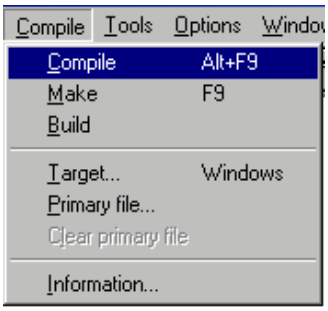
2.2.5 Compile Menüsü

Compile Menüsü (Şekil 2.10) editörde aktif bulunan Pascal programının derlenmesinde, bu programa bağlı unit programların derlenmesinde ve yazılan Pascal programının dos/windows ortamında kendi başına çalışır duruma yani; XXXXXXXX.EXE durumuna getirilmesi için kullanılır. En çok kullanacağımız menü komutları aşağıda özetlenmiştir.

Compile: Editör ekranındaki aktif durumda olan programı derlemek amacıyla kullanılır. Kısa yol tuşu (Alt+F9).

Make: Editör ekranındaki aktif durumda olan programı ve bu programa bağlı unit harici programlarını en son değiştirilmiş halleriyle derleme işlemini yapar. Kısa yol tuşu (F9).

Build: Editör ekranındaki aktif durumda olan programları ve bu programa bağlı unit harici programlarını birlikte derler.



Şekil 2.10. Compile Menüsü

Destination Memory (Disk): Sadece DOS Turbo Pascalda bulunan bu komut ile Programın derlenerek çalıştırılacağı ortamın belirtilmesi için kullanılır. Seçenek üzerine "Enter" tuşuna basılarak Memory yazılı bölüm ***Disk*** veya Disk yazılı ise ***Memory*** durumuna alınır. ***Destination Disk*** durumunda derleme sonucunda program dos ortamında direkt olarak çalıştırılabilir.

Target: Yazılan programın derlenip çalıştırılacağı ortamı belirlemek için kullanılır. Pascal Real mode, Protected mode ve Windows olmak üzere üç ortam sunmaktadır.

Editör ve derleyici ile ilgili detaylar kullanıcı kitaplarından veya Help menüsünden sağlanabilir.

BÖLÜM 3

PASCAL PROGRAMLAMA DİLİ YAPISI

3.1. Giriş

Bir Pascal programı en genel anlamda üç ayrı kısımdan oluşmuştur. Bu kısımlar bulunmaları gereken sıraya göre aşağıda verilmiştir.

```
Program Başlığı;
Tanımlama Bloğu;
BEGIN
    İcra Bloğu;
END.
```

Programların asıl icra bölümü son bölümüdür. Yukarıda icra bloğu olarak gösterilen bu bölüm, Pascal komut cümlelerinden oluşur. ICRA bloğu, "BEGIN" ile başlar "END." ile sona erer. Her program bloğu birden fazla "END" içerebilir. Ancak bu end deyimleri program içinde bulunan değişik blokların sonunu göstermek için kullanılır ve hiç birinin sonunda "." işareti bulunmaz. "." işareti sadece ana programın sonunu göstermek amacıyla kullanılabilir. Ana programın sonu haricindeki diğer "END" deyimlerinin sonunda ";" işareti kullanılır.

Program Başlığı: Bir Pascal programının ilk kısmı, kullanılması programcının seçimine bağlı olan *"program başlığı"* dır. Program başlığı, programa bir isim vermek için kullanılır ve program isimlerinde İngiliz alfabesinde bulunmayan Türkçe karakterler kullanılmamalıdır. Programa uzun isimler verilebilir ancak sözcükler arasında boşluk bulunmamalıdır.

Tanımlama Bloğu: Pascal programının bu bölümünde program icra bloğunda kullanılan sabitler ve değişkenlerin isimleri ve bunların ne tür sabit/değişken olduğu bildirilir. Bu blok Pascal programı içerisinde mutlaka belirtilmelidir. Örnek olarak, aşağıda değişik veri tiplerindeki değişkenlerin bir tanımlama bloğunda nasıl tanımlanabileceği gösterilmiştir.

Tanım	Veri Tipi
ABS	:REAL;
A	:STRING;
B	:INTEGER;
C	:LONGINT;
F	:SHORTINT;
H	:BOOLEAN;
J	:BYTE;
KL	:WORD;

3.2. Özel Semboller ve Pascal Sözcükleri

3.2.1 Özel Semboller

Bir Pascal programında kullanılabilecek özel semboller A'dan Z'ye büyük ve a'dan z'ye küçük İngiliz alfabesi harfleri, 0-9 arası rakamlardır. Pascal programlama dilinde kullanılan diğer bazı semboller ve anlamları aşağıda tanımlanmıştır.

>, <, =	Herhangi değişken ve sabitlerin karşılaştırılmasında kullanılır.
[ ]	Bu işaret içine diziye ait indis numarası yazılır.
( )	Veri okunması, yazılması, yazılacak/okunacak bilgilerin başlangıç ve bitişlerinde kullanılır.
..	İndisli değişken eleman tanım ayırıcı.
;	Pascal komut ve deyimlerini birbirinden ayırmak için kullanılırlar.
' ' (Tek Tırnak)	Alfasayısal bilgilerin tanımlanmasında kullanılır. Bilgilerin başlangıç ve bitişlerinde kullanılmaktadır.
,	Değişken tanımlarında, okunacak yada yazdırılacak bilgiler arasında ayıraç olarak kullanılır.
:	Bilgi karakter yapı tipi tanımlamalarında ve sayısal bilgilerin çıkış formatlarında kullanılır.
.	Bir gruba bağlı bir alt grup veya bir kayıt saha ismi tanımlarında elemanlandırma işareti olarak, sabit bilgilerdeki rakamların ondalık işareti olarak ve Pascal programlarının sonunu belirlemek amacıyla kullanılır.
:=	İşaretin solundaki değere sağındaki bilgiyi atama işlemini yapar.
{ }	Açıklama ifadelerinin başlangıç ve bitimlerinde kullanılır.
(* *)	Açıklama ifadelerinin başlangıç ve bitimlerinde kullanılır.

3.2.2 Aritmetik İşlem İşaret Karakterleri

Program içinde yapılacak aritmetik işlemlerde kullanılır.

+	Toplama işlemi ve karakter katarlarının birleştirilmesinde kullanılır
-	Aritmetik işlemlerin yapılmasında çıkarma işlemi amacıyla kullanılır
*	Aritmetik işlemlerin yapılmasında çarpma işlemi amacıyla kullanılır
/	Aritmetik işlemlerin yapılmasında bölme işlemi amacıyla kullanılır

3.2.2.1 Aritmetik İşlemlerde İşlem Öncelik Sıraları

Öncelik Sırası	Aritmetik İşlem ve bağıntılı tanımlar
1	İşaretli sayısal bilgiler
2	Parantez içindeki işlemler
3	Aritmetik fonksiyonlar
4	Üs alma
5	Çarpma ve Bölme
6	DIV tam sayıya bölme deyimi
7	MOD (Bölme işleminde kalan bulma işlemi)
8	Toplama ve çıkarma
9	Mantıksal ilişki ifadeleri (=, >,<,)
10	Bağlantı ifadeleri (not,and,or,xor)

İşlem Önceliğinde Bazı Kurallar:

- Aynı öncelik sırasına sahip işlemlerde hangisi önce tanımlanmış ise önce onun işlemi gerçekleştirilir.
- İşlem öncelik sırası küçük olan bir aritmetik işlemin önce yapılması isteniyorsa parantez içine alınmalıdır.
- İç içe birden fazla parantez kullanılması durumunda, işlem öncelik sırası daima en içteki paranteze ait işleminindir.

3.2.3 Pascal Sözcükleri

Turbo Pascalda kendi işlevleri dışında kullanılamayacak olan sözcükler vardır. Bu sözcükler ayrılmış sözcükler olarak tanımlanmakta olup programcı tarafından yeniden tanımlanamaz, bir değişken ismi olarak kullanılamaz veya değiştirilemezler. Bu sözcüklerin isimleri aşağıda verilmiştir:

And	Asm	Array	Begin
Case	Const	Constructor	Destructor
Div	Do	Downto	Else
End	File	For	Function
Goto	If	Implementation	In
Inline	Interface	Label	Mod
Nil	Not	Of	Object
Or	Packed	Procedure	Program
Record	Repeat	Set	Shl
Shr	String	Then	To
Type	Unit	Until	Uses
Var	While	With	Xor

3.3. Tanıtıcılar

Pascal programlama dilinde icra bloğu içinde kullanılan değişkenler **VAR** tanım bloğu içinde, sabitler ise **CONST** bloğu içinde tanımlanır. Örneğin *A,B* değerleri değişken ve *C* değeri sabit ise bu değerlerin programın tanım bloklarında ifade edilmesi aşağıdaki şekilde olacaktır.

```
VAR A;B:REAL;
CONST C=12:INTEGER;
```

Burada, VAR'dan sonra gelen REAL ifadesi *A* ve *B* değişkenlerinin gerçel sayı veri tipinde olduğunu belirtirken CONST'dan sonra gelen INTEGER ifadesi C sayısının integer veri tipi olduğunu belirtmektedir.

Pascal Programlama dilinde kullanılan değişik veri tiplerinin tanımları ve geçerli olduğu alanlar aşağıda tanımlanmıştır.

3.3.1 Integer Tipi Veriler

Integer tipindeki veriler tamsayılardan oluşmuşlardır ve kayar nokta içermezler. Turbo Pascal'daki integer tipteki tamsayılar -32768 ile +32767 arasında sınırlandırılmışladır.

3.3.2 Word Tipi Veriler

Word Tipi veriler, 0-65535 arasındaki tamsayılardan oluşmuşlardır.

3.3.3 Shortint Tipi Veriler

-128 ile +127 arasındaki tamsayılardan oluşmuşlardır.

3.3.4 Byte Tipi Veriler

Byte tipi veriler işaret ve desimal nokta içermeyen 0-255 arasındaki değerlerden oluşan tamsayılardır. Programda, 0-255 arasındaki tamsayılar için değişken veya sabitlerin Byte tipi olarak tanımlanması bellekten tasarruf sağlar.

3.3.5 Longint Tipi Veriler

-2147483648 ile +2147483647 arasında görev yaparlar ve tamsayıların menzil olarak en kapsamlısıdır.

3.3.6 Boolean Tipi Veriler

Mantıksal ifadeler olarak da isimlendirilebilen Boolean tipi veriler sadece iki değerden oluşurlar. Bunlar "FALSE" ve "TRUE" dir.

```
Durum:Boolean;
Hata:Boolean;
```

3.3.7 Char Tipi Veriler

Char tipi veriler, ASCII karakter kümesinin 256 adet elemanından oluşurlar. Char tipi veriler bir sabit olarak ele alındığından ' ' içerisinde yazılır.

3.3.8 String Tipi Veriler

Bu tip veriler, karakter kümesi şeklinde olup ' ' içerisinde yazılırlar.

```
A:STRING[5];
B:STRING[16];
```

Burada A isimli karakter kümesi maksimum 5 karakterden, B isimli karakter dizisi ise maksimum 16 karakterden oluşmaktadır.

Bu bilgiler ile ilk Pascal programımızı yazabiliriz.

```
Program Birinci ve ilk prog ram;
Var
Mesaj:string[7];
BEGIN
  Mesaj:='MerhaBA'
   Write('Sayın bilgisayar kullanıcısı ',Mesaj);
END.
```

Programımızın çalıştırılmasıyla ekran görüntüsü aşağıdaki gibi olacaktır.

```
Sayın bilgisayar kullanıcısı MerhaBA
```

3.3.9 Real Tip Veriler

Real tip veriler desimal nokta içeren ve üslü formda ifade edilebilen gerçel sayılardır. Üs ifadesi olarak "E" harfi kullanılmaktadır. Kullanımları ile ilgili örnekler aşağıda verilmiştir. Tanım aralığı;2.9E-39..1.7E38 'dir. Bellekte 6 byte yer kaplar.

-2.3454
123.11223344
-2.45E+12
2.45E+12
1.123123E-12

Olağan koşullarda bir programlama hatası söz konusu değilse veya çok büyük sayılar ile çalışılmıyor ise REAL tip sayılarla çalışmada bir sorun ile karşılaşmayız.

Kişisel bilgisayarların mikro işlemcileri 80x86 olarak isimlendirilir. Bu işlemci ailesinin yükünü hafifletmek için 80x87 olarak bilinen matematik işlemcisi kullanılır. Pascalda bu işlemci ile kullanılmak üzere dört tip tanımlanmıştır. Bunlar;Single, Double, Extended ve Comp 'tur.

3.3.10 Single Tip Veriler

Single tip veriler, 1.5E-45...3.4E38 sayı kümesi aralığında tanımlanabilir. Bellekte 4 byte yer kaplar.

3.3.11 Double Tip Veriler

Double tip veriler, 5.0E-45...3.4E38 sayı kümesi aralığında tanımlanabilir. Bellekte 8 byte yer kaplar.

3.3.12 Extended Tip Veriler

Extended tip veriler, 3.4E-4932...1.1E4932 sayı kümesi aralığında tanımlanabilir. Bellekte 10 byte yer kaplar.

3.3.13 Comp Tip Veriler

Comp tip veriler çok uzun tamsayıların tanımlanması için kullanılır. Comp veri tipi ile 19-20 basamaklı sayılar ifade edilir. Tanım aralığı; $-2^{63}...2^{63}$

80x87 sınıfındaki veri tiplerinin kullanılabilmesi için programın en başına {$N+} derleyici komutu verilmelidir. Bilgisayarda matematik işlemci yok ise {$E+} matematik işlemci emülasyonu da bu satıra eklenir.

Günümüzün Pentium tabanlı bilgisayarlarında matematik işlemci bulunduğundan, matematik işlemcili bilgisayarlarda emülasyon direktifini kullanmaya gerek yoktur.

Örnek:

```
{$N+}{$E+} {Buradaki $E+ ifadesi matematik işlemci emülasyonudur.}
program cok buyuk sayilar;
var
Uses wincrt;{Dos için CRT}
  a:single;      b:double;
  c:extended; d:comp;
begin;
        A:=12E-40; B:=12E+39;D:=1234567890;C:=A+B+D;
Writeln(C);
End.
```

BÖLÜM 4.

OPERATÖRLER

4.1 Giriş

Turbo Pascal programlama dilinde de diğer programlama dillerinde olduğu gibi operatörler, yapılan işlem türüne göre aritmetik, mantıksal ve karşılaştırma operatörleri olmak üzere çeşitlilik gösterirler.

4.2. Aritmetik Operatörler

Aritmetik operatörler dört işlem için kullandığımız +, -, / ve * 'dan oluşmaktadırlar. Sık sık kullandığımız bu operatörlerin yanında iki sayının bölümünden kalanı veren *MOD,* iki tamsayının bölümünün sonucunda tamsayı üreten *DIV* programlarımızda sık kullandığımız aritmetik operatörlerdir. Tam ve gerçel sayıların aritmetik işlemleri için kullanılan operatörlere ilişkin liste aşağıdaki tabloda verilmiştir.

Operatör	İşlem	Operand Tipi	Sonucun Tipi
+	Toplama	real,integer	real,integer
-	Çıkarma	real,integer	real,integer
*	Çarpma	real,integer	real,integer
/	Bölme	Real,integer	Real
Div	Tamsayı Bölme	integer	İnteger
Mod	Kalan Bulma	integer	İnteger

Bu işlemler arasındaki işlem önceliği sırası Bölüm 3'de anlatılmıştı.

4.2.1 Div Operatörü

İki tamsayının bölümünün tamsayı kısmını verir.

123/4=30.75
123 div 4=30

4.2.2 Mod Operatörü

Tamsayı tipindeki operandların bölümünü yapar ve sonucun kalan kısmını bir tamsayı değer olarak üretir.

123/4=30.75
123 mod 4 =3

'/' operatörü operand olarak kullanılan değerlerin tipi yada bölümün sonuç değeri ne olursa olsun real tipte sonuç üretir. Bu nedenle bir bölümün sonucunun bir tamsayı değişkene direkt olarak aktarılması söz konusu değildir.

2/1=1.0000000E+00

Bir bölüm işleminin tamsayı değişkene aktarılabilmesi için real değerleri integere dönüştüren fonksiyonlardan yararlanılabilir.

Trunc(2.123/1)=2

sonucu integer olarak elde edilir.

Aritmetiksel işlemlerde kullanılan operandların biri real diğeri integer ise sonuç real olacaktır.

1+2.0=3.0000000E+00

Pascalda üs almak için bir operatör yoktur. Ancak "Exp" ve "Ln" fonksiyonları yardımıyla üs alma işlemleri yapılabilmektedir.

Örneğin bir a sayısının b. üssünü bulmak için;

Exp(b*Ln(a))

yazılır.

Örnek; Exp(2*ln(2))=4
Exp(3*ln(2))=8

4.3. İlişkisel Operatörler

Diğer programlama dillerinde de olduğu üzere, Pascal programlama dilinde iki veya daha fazla operand arasındaki ilişkilerin araştırılmasını yapan ilişkisel operatörler, toplu olarak aşağıdaki tabloda verilmiştir.

Operatörler	Anlamı
=	Eşit
<>	Eşit Değil
>=	Büyük eşit
<	Küçük
>	Büyük
<=	Küçük eşit

Karşılaştırmalardan elde edilen sonuç daima Boolean tipte olacaktır. Yapılan karşılaştırmanın sonucu doğru ise *true*, yalnış ise *false* değeri elde edilir.

4.4 Mantıksal Operatörler

Aşağıdaki tablolarda görüldüğü gibi 4 adet mantıksal operatör vardır. Bunlar;

AND :İki operand doğru ise sonuç doğru,
OR :İki operanddan en az biri doğru ise sonuç doğru,
XOR :İki operandın sadece birinin doğru olması hali için sonuç doğru,
NOT :Operand yanlış ise sonuç doğru

şeklinde sıralanıp tanımlanabilir. Aşağıdaki tablolarda A ve B nin Doğru (T) ve/veya Yanlış (F) oluşlarına göre işlem sonucu görülmektedir.

A	B	A and B
F	F	F
F	T	F
T	F	F
T	T	T

A	B	A or B
F	F	F
F	T	T
T	F	T
T	T	T

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F

A	NOT A
F	T
T	F

Mantıksal operatörlerin operandları bir ilişkisel operasyon veya sadece bir boolean olabilir. Bir tek koşul cümlesi içinde birden fazla ilişkisel operatör var ise bunlar mantıksal operatörler ile birleştirilmeli ve herbir ilişkisel operasyon ifadesi parantez içinde yazılmalıdır. Aşağıdaki yazım formunu inceleyiniz.

```
if  (a=b) and((c<d) or(c>a)) then...
```

4.5 İşlevsel Operatörler

Programcı tarafından FUNCTION bildiri deyimiyle özel amaçlar için tanımlanan fonksiyonlarla, Turbo Pascalda tanımlı olan standart fonksiyonlar işlevsel fonksiyonlar olarak adlandırılır. Bunlardan sadece bir kaçı aşağıda örnek olarak verilmiştir. İşlevsel operatörler Pascal Arşivi Bölümü altında daha ayrıntılı olarak işlenecektir.

Örnekler:
```
SQRT(4)=2.0000000E+00   (Verilen bir sayının Karekökünün alınması)
SQR(4)=16               (Verilen bir sayının Karesinin alınması)
Sin(60*Pi/180)=8.660E-01  (Bir sayının sinüsünün hesaplanması)
```

4.6 Matematiksel Formüller

Normal olarak elle yazılan formüllerin bir bilgisayar programına yazımı için belirli bazı kurallar vardır. Matematiksel ifadelerin bilgisayarın anlayacağı şekilde yazımı sırasında dikkat edilecek önemli bir nokta, matematik operatörleri arasındaki işlem önceliğine dikkat edilmesidir.

Aşağıda verilen cebirsel ifadelerin Pascal programlama dilinde yazımlarını inceleyiniz.

Örnek 1. *Cebirsel İfade* *Pascal karşılığı*

$Z = 2.\sqrt{Z}\ /(2-x)$ z:=2*sqrt(z)/(2-x)

Yukarıdaki örnekte işlem sırası;
1. Z'nin karekökü alınır
2. 1 nolu işlem 2 sabiti ile çarpılır
3. (2-x) değeri hesaplanır
4. 2 nolu işten elde edilen sonuç 3 nolu işlemin sonucuna bölünür.

Örnek 2. *Cebirsel İfade* *Pascal karşılığı*

$$x = \frac{q}{c-a}$$ x:=q/(c-a)

Örnek 3. *Cebirsel İfade* *Pascal karşılığı*

$\Delta=B^2-4.A.C$ DELTA:=SQR(B)-4*B*C

Aşağıda verilen matematiksel eşitlikleri Pascal programlama dili kurallarına göre yazınız.

$$M = \frac{(C_1 - C_2)}{\sqrt{(C_1 - C_2)^2}}$$

$$\text{Denklem} = \frac{\sqrt{Z1\frac{2}{g}(2x^2 - 2y)^2}}{(ax^2 + bx + c)}$$

$$A = \frac{B1 + 2cd(i)}{\cos x} - 2\sin x + \frac{12cf}{\pi^2}$$

BÖLÜM 5

GİRİŞ/ÇIKIŞ DEYİMLERİ

5.1. Giriş

Giriş/çıkış deyimleri bilgisayarın ana belleği ile çevre aygıtlar arasında bilgi transferi yaparlar. Herhangi bir çevre aygıtından bir verinin belleğe okutulması veya ana bellek üzerindeki bir verinin çevre aygıtlara gönderilmesi giriş/çıkış deyimleri ile gerçekleştirilir. Genellikle giriş çıkış deyimi olarak aşağıda tanımlanan deyimler kullanılmaktadır.

5.2 Read-Readln

Pascal 'da okuma işlemi için kullanılan komutlar *READ* ve *READLN* olup genel kullanım yapısı şu şekildedir;

Read(A1,A2,A3,...,An)

ReadLn(A1,A2,A3,...,An)

Read ile Readln arasındaki fark: Read, okuma işlemi tamamlandıktan sonra aynı satırda kalınmasını, Readln ise yeni bir satıra gelinmesini sağlar. Bir Pascal programı içerisinde bu iki komutun kullanımı aşağıdaki program parçasında verilmiştir.

```
Var
s,a,n:real;
......
begin
     Read(s,a,n)
end.
```

Okunacak değerlerin düzeni;

12.45 43.543 62.12

Aynı değerleri ReadLn ile okutturmak istersek;

```
Var
s,a,n:real;
......
begin
     Readln(s,a,n)
end.
```

Okunacak değerlerin düzeni yukarıdaki okuma düzeni ile aynı olabileceği gibi her değer ayrı bir satırda verilebilir, her hangi satırlarda verilebilir, örnek olarak s, a ve n değerleri aşağıdaki okuma düzeninde okutturulmuştur.

```
12.45      43.543
62.12
```

5.3 Write-Writeln

Pascal programlama dilinde yazma işlemi için *WRITE* ve *WRITELN* komutları kullanılmaktadır. Bu komutların genel kullanım yapısı şu şekildedir;

Write(A1,A2,A3,...,An)

Writeln(A1,A2,A3,...,An)

İki kullanım şekli arasındaki fark, Read ile Readln arasındaki fark gibidir. İfadeler Write ile yazdırıldıktan sonra kursör aynı satırda bekleyecektir, Writeln ile yazdırılması durumunda ise kursör bir sonraki satıra geçecektir.

```
Var
s,a,n:integer;
......
begin
     s:=3;a:=9;n:=5;
     Write(s);
     Write(a);
     Write(n);
end.
```

Programın çalıştırılmasıyla elde edilen sonuç;

```
395
```

olacaktır.

Aynı değişkenleri Writeln ile yazdırdığımızda;

```
Var
s,a,n:integer;
.....
begin
    s:=3;a:=9;n:=5;
    Writeln(s);
    Writeln(a);
    Writeln(n);
End
```

```
3
9
5
```

elde edilecektir.

5.4 Yazım İçin Format Belirleme

Writeln deyimi ile yapılan bilgi çıkışlarını düzenlemek için yazım formatları kullanılır. Yazdırılmak istenilen değerin tipine bağlı olarak iki farklı yazım formatı kullanılır.

1. Yazım Formatı ***(M:n)*** **:**Bu yazım formatına göre yazdırılmak istenilen *M* değeri *n* karakterlik alanda sağa dayalı olarak yazılır. *M* değerini oluşturan dijit veya karakterin uzunluğu *n* sayısından küçük ise, aradaki fark kadar sol tarafta boşluk bırakılır. *M* değerinin uzunluğu *n* sayısına eşit veya büyük ise yazdırma işlemi format belirtilmemiş gibi gerçekleştirilir.

Örnek:

```
Const
i:integer=567;
b:boolean=True;
c:char='b';
h:string[10]='Nasılsınız';
begin
   Writeln('123456789');Writeln;
   Writeln(i:9);
   Writeln(b:9);
   Writeln(c:9);
   Writeln(h:9);
   Writeln(i:2);
   Writeln(h:5);
end.
```

2. Yazım Formatı (M:n:k) **:** Bu yazım formatı real sayılar üzerinde çalışmaktadır. Bu yazım formatıyla reel sayılar üssüz notasyonda yazdırılır. Bunu sağlamak için sadece real sayılar için geçerli olan 2. yazım formatı kullanılır. Yazdırılacak *M* real değeri *n* karakterlik alanda sağa dayalı olarak üssüz yazdırılır, *k* ise *n* karakterin sonundaki kesirli kısmı göstermektedir.

Örneğin; A değeri, *A:7:2* şeklinde yazdırılmak istenirse; program real sayı için 7 karakterlik yer ayırır. Bu alanın son iki hanesi ondalıklı kısım için sondan 3. hanesi ise desimal nokta için kullanılır.

Sayının tamsayı kısmı ayrılan alanın uzunluğundan küçük ise, aradaki fark kadar sol tarafta boşluk bırakılır. Yazdırılacak sayının tam sayı kısmı bu alana sığmıyor ise, tamsayı için ayrılan alanın uzunluğu yazdırılan sayının tamsayı alan uzunluğuna eşit büyüklükte kabul edilir. Kesirli kısmı ayrılan alandan daha küçük ise aradaki fark kadar sağ tarafa *0* dijiti ilave edilir. Kesirli kısım alana sığmıyorsa yuvarlatılarak yazılır.

Yazım sırasında kesirli alanın yazılması istenmiyorsa *k* sayısının "0" yazılması gerekir. *k* sayısının sıfır olması desimal noktanın yok olmasını sağlar.

Örnek:

```
const a:real=123.127927;
begin
  Writeln('123456789');
  Writeln('*********');;
  Writeln(a:9:4);
  Writeln(a:9:3);
  Writeln(a:9:2);
  Writeln(a:9:0);
  Writeln(a:7:2);
  Writeln(a:6:2);
  Writeln(a:5:2);
  Writeln(a:0:0);
end.
```

Programın çalışmasıyla elde edilen sonuç aşağıda verilmiştir. Program sonucuna göre sondan 2. ve 3. satırların aynı olduğuna dikkat ediniz.

```
123456789
*********
 123.1279
  123.128
   123.13
      123
 123.13
123.13
123.13
123
```

BÖLÜM 6

SİSTEM BİRİMİ VE EKRAN KOMUTLARI

Ekran komutları ekrandaki görüntü tasarımı için kullanılan komutlardır. Bu komutların program içinde kullanılabilmesi için, program başlığı satırından sonra USES komutunda CRT (Windows programları için WINCRT) unit isminin yazılması gereklidir. Aksi halde, ekran komutlarıyla ilgili yazılan komutlar Pascal derleyicisi tarafından tanınmayacaktır. En çok kullanılan ekran komutları aşağıda verilmiştir.

6.1 Clrscr

"Clrscr" CRT/WINCRT üniti içinde yer alan bir alt programdır. Ekranda daha önce yazılı olan ifadeleri silerek ekranın temizlenmesi amacıyla kullanılır.

6.2 GotoXY

Kursörü ekranın istenilen sütun ve satırına taşımak için kullanılır. Kullanımı;

```
GotoXy(Sütun,Satır);
```

şeklindedir. Normal bir ekran üzerinde 80 sütun ve 25 satır vardır.

Örnek:

```
Uses  wincrt;
var
j:word;
begin
clrscr;
for j:=1 to 20 do
     begin
          gotoxy(j,j);Write('Balıkesir Üniversitesi');
          gotoxy(50-j,j);Write('Balıkesir Üniversitesi');
     end;
end.
```

6.3. Readkey

Klavyeden basılan karaktesi okumak için kullanılır. Bu fonksiyonun sonucu char tipi bilgidir. Turbo/Borland Pascal 7.0 'da tek başına kullanıldığında program bu komuta rastladığında, herhangi bir tuşa basılıncaya kadar programın çalışması kesilir.

Örnek

```
Uses wincrt;
var
tus:char;
begin   clrscr;
        Write('Balıkesir  ');Readkey;
        Writeln('Üniveristesi');Readkey;
end.
```

Readkey komutunun çok kullanıldığı uygulamalardan biri de basılan veya basılacak karakteri kontrol etmektir.

Aşağıda verilen örnek programda her B tuşuna basıldığında "Balıkesir Üniversitesi" yazmaktadır. Program ESC tuşuna basılarak durdurulmaktadır.

Örnek:

```
Uses  wincrt;
var
tus:char;
j:word;
begin
   repeat
        tus:=readkey;
        if upcase(tus)='B' then
        Write('Balıkesir Üniversitesi');
   until tus=#27;
end.
```

6.4 Keypressed

Klavyeden bir tuşa basılıp basılmadığını kontrol etmek amacıyla kullanılan bir komuttur. Aşağıda verilen örnek programı inceleyiniz.

Örnek:

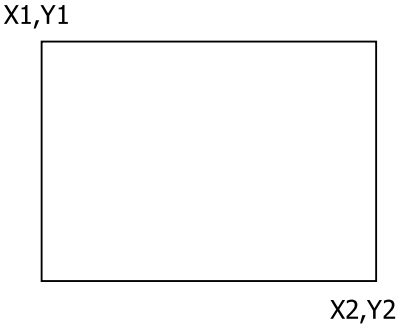
```
Uses  wincrt;
var
tus:char;
j:word;
begin
j:=0;
  repeat
  Write('Balıkesir Üniversitesi');
  j:=j+1;
  if keypressed then tus:=readkey;
  if j=75 then
     begin
          clrscr;
          j:=0;
     end;
  until tus=#27;
end.
```

6.5 Window

Ekranda pencere oluşturmak için kullanılan bir komuttur. Bu komut sadece Turbo/Borland Pascalda kullanılabilir. Kullanımı;

```
Window(X1,Y1, X 2,Y2);
```

X1 :Pencerenin sol sütun numarası (1-80)
Y1 :Pencerenin sol satır numarası (1-25)
X2 :Pencerenin sağ sütun numarası (1-80)
Y2 :Pencerenin sağ satır numarası (1-25)



6.6 Delay

Programın belirtilen süre kadar bekletilmesini sağlar. Delay (1000) komutu, programın 1 sn bekletilmesini sağlamaktadır. Bu komut sadece Turbo/Borland Pascalda kullanılabilir.

Örnek:

```
uses crt;
begin
  window(20,5,60,20);
  REPEAT
    DELAY(5); Write('BAÜ');
  UNTIL keypressed;
end.
```

6.7 ClrEol

Kursörün bulunduğu konumdan satırın sonuna kadar bütün karakterleri silmek için kullanılır. Kursör bulunduğu konumda kalır.

Örnek:

```
uses winCrt;
begin
  ClrScr;
  Writeln('Günaydın Bugün nasılsınız?');
  Writeln('Enter tuşuna Basınız...');
  Readln;
  GotoXY(1,2); {Kursör 2.satır 1.sütuna konumlandırılıyor}
  ClrEol;
  Writeln ('İyi olmanızı duymak güzel.');
end.
```

6.8 Highvideo

Ekrana yazılacak yazının parlak yazılmasını sağlar. Bu komut sadece Turbo/Borland Pascalda kullanılabilir.

6.9 Lowvideo

Ekrana yazılacak yazının mat yazılmasını sağlar. Bu komut sadece Dos için Turbo/Borland Pascalda kullanılabilir.

6.10 Normvideo

Ekrana yazılacak yazının normal parlaklıkta yazılmasını sağlar. Bu komut sadece Turbo/Borland Pascalda kullanılabilir.

Örnek:

```
Uses Crt;
Begin
Clrscr;
        Highvideo;Writeln('Parlak');
        Lowvideo;Writeln('Mat');
        Normvideo;Writeln('Normal');
End;
```

6.11 WhereX

Kursörün üzerinde bulunduğu sütunun numarasını verir.

6.12 WhereX

Kursörün üzerinde bulunduğu satırın numarasını verir.

Örnek:

```
uses winCrt;
begin
  Write('Şu anda kursör pozisyonu :',WhereX,',',WhereY);
end.
```

6.13 TextColor

Ekrana yazdırılacak yazının rengini ayarlamak için kullanılır. Bu komut sadece Turbo/Borland Pascalda kullanılabilir. Kullanımı;

```
Textcolor(Renkkodu/Adı);
```

Renklerin kodları aşağıda verilmiştir.

Renk	Kod	Adı
Siyah	0	Black
Mavi	1	Blue
Yeşil	2	Green
Turquaz	3	Cyan
Kırmızı	4	Red
Pembe	5	Magenta
Kahve	6	Brown
Açık gri	7	Lightgray
Koyu gri	8	Darkgray
Açık mavi	9	Lightblue
Açık yeşil	10	Lightgreen
Açık turquaz	11	Lightcyan
Açık kırmızı	12	Lightred
Açık pembe	13	Lightmagenta
Sarı	14	Yellow
Beyaz	15	White
Yanıp/sönme	128	Blink

Textcolor(0)=Textcolor(black)

6.14 Textbackground

Ekrana yazdırılacak yazının zemin rengini ayarlamak için kullanılır. Bu komut sadece Turbo/Borland Pascalda kullanılabilir. Kullanımı;

Textbackground(renk kodu);

Örnek:

```
uses Crt;
 begin
   { Siyah üzerinde yeşil karakterler}
   TextColor(Green);
   TextBackground(Black);
   WriteLn('Merhaba');
   { Gri üzerinde yanıp sönen kırmızı karakterler}
   TextColor(LightRed+Blink);
   TextBackground(LightGray);
   WriteLn('Günaydın!');
   { Mavi üzerinde sarı karakterler}
   TextColor(14); { Yellow = 14 }
   TextBackground(Blue);
   WriteLn('nasılsın');
   NormVideo; { Orijinal özellik }
End.
```

6.15 Sound/Nosound

Sound, verilen frekansta ses üretmek için kullanılır. Nosound, sound ile üretilen sesi ortadan kaldırmak için kullanılır. Bu komutlar sadece Turbo/Borland Pascalda kullanılabilir.

Örnek:

```
uses Crt;
begin
  Sound(220);
  Delay(200);
  NoSound;
end.
```

Çalışma Sorusu:

Readkey ve/veya keypressed komutlarından yararlanarak bir şifre programı yazınız. Program çalıştırılıp şifrenin girilmesi esnasında her girilen karaktere karşılık ekranda X karakteri görüntülenecektir.

BÖLÜM 7

PASCAL ARŞİVİ

7.1 Giriş

Pascal Arşivi, programcılara sistem, ekran, grafik ve yazıcı birimleri için hazırlanmış standart yardımcı programlardan oluşmuştur. Pascal içinde bulunan arşiv fonksiyonları ana programa otomatik olarak bağlanır.

7.2 Sistem Birimi ve Katarlar

Sistem birimi (system unit) Pascal 'da USES ifadesinde tanımlanması gerekmeyen bir birimdir. Sistem birimi Pascalın temel sabit, değişken ve alt programlarını içermektedir. Bunlar arasında stringler oldukça önemli bir yer tutmaktadır. Turbo Pascal'da String elemanları için kullanabileceğimiz bir çok standart function alt programı bulunmaktadır. Bunların arasında Chr, Concat, Upcase, Copy, Delete, Length, Str, Val, Pos bulunmaktadır.

7.2.1 Chr

Standart CHR fonksiyonu [0..255] arasındaki tam sayıların ASCII karşılığı olan karakteri vermektedir.

CHR fonksiyonuna, doğrudan aktarılan tam sayı sayısal ifadeleri alfabetik karaktere çeviren bu fonksiyonun yaygın kullanımlarından biri de, bilgilerin yazıcıya dökümü sırasında klavye üzerindeki komutların ya da kontrol karakterlerinin üretilmesidir. Aşağıdaki örnek programda bir yazıcıda kağıda üst üste basmak için ENTER tuşuna karşılık gelen ASCII 13 karakterini kullanmak amacıyla CHR fonksiyonundan yararlanılabilir.

Örnek:

```
Program Chr  Ornek;
Uses LST;
begin
        Writeln('Endüstri',Chr(13),'Mühendisliği');
Readln;
end.
```

7.2.2 Concat

Bilgilerin birbirlerine eklenmelerini sağlayan, alfa sayısal fonksiyondur. Ekleme işlemleri CONCAT fonksiyonu ile veya alfa sayısal bilgiler arasına **+** işareti konarak sağlanır. Concat fonksiyonu ile istenilen sayıda katar birbiri ardına eklenebilir. CONCAT fonksiyonunun kullanımı aşağıdaki örnek programda verilmiştir.

Örnek:

```
Program Ornek  Concat;
Uses Crt;{windows için Wincrt}
var
x1,x2,x3,x4:String[20];
x5,x6:=String[50];
begin    clrscr;
         x1:='Balıkesir '; x2:='Mühendisliği';
         x3:='Universitesi ';x4:='Endüstri ';
         x5:=Concat(x1,x3,x4,x2);
Writeln('4 sözcüğün CONCAT ile birleştirilmesi');
Writeln(x5);
x5:=x1+x3+x4+x2;
Writeln('4 sözcüğün    +     ile birleştirilmesi');
Writeln(X5);
While Not keypressed do;
end.
```

7.2.3 Upcase

Pascal 'da okunan veya karşılaştırılması yapılan iki karakter dizisinde yazılan harflerin büyük veya küçük oluşu önemlidir. Upcase fonksiyonu küçük harfleri büyük harfe çevirir. Parametre olarak verilen karakterlerin alfabetik olmaması durumunda herhangi bir işlem yapılmaz.

Örnek :

```
var
  s : string;
  i : Integer;
begin
  Write('Bir Karakter Dizisi Giriniz: ');
  ReadLn(s);
  for i := 1 to Length(s) do
    s[i] := UpCase(s[i]);
  WriteLn('KARAKTER DİZİSİ BUYUK HARFE DÖNÜŞTÜ    ',s);
Readln;
end.
```

7.2.4 Copy

Bir karakter dizisinde belirlenen bir bölgeden istenilen uzunlukta bir parçanın kopyalanması için kullanılan fonksiyondur.

Örnek:

```
uses crt; {windows için Wincrt}
var s: string;
begin
  Clrscr;
  s:= 'Balikesir Üniversitesi'
  WriteLn(Copy(s,11, 4));
  repeat until keypressed;
end.
```

Programın çıktısı; Univ şeklinde olacaktır.

7.2.5 Delete

Delete fonksiyonu, bir karakter dizisi içindeki belirli bir alt karakter dizisinin silinmesi için kullanılan standart fonksiyondur. Copy komutunda olduğu gibi silinecek ilk karakterin karakter dizisinin kaçıncı elemanı olduğu ve kaç eleman silineceği ifade edilmelidir.

Örnek:

```
Var  s: string;
begin
   S:='Endüstri Mühendisliği Bölümü';
   Delete(s,10,21);
   WriteLn(s);
end.
```

Programın Çıktısı: **Endüstri** şeklinde olacaktır.

7.2.6 Length

Length fonksiyonu bir karakter dizisinin uzunluğunu hesaplamak amacıyla kullanılır. Hesaplama sırasında karakter dizisi arasında boşluklar var ise bunları da bir karakter olarak kabul edecektir.

Örnek:

```
Program length  ornek;
uses crt; {windows için Wincrt}
var i:string[39];
begin clrscr;
  i:='Endüstri Mühendisliği Bölümü';
  WriteLn('Uzunluk = ', Length(i));
end.
```

Programın Çıktısı: **Uzunluk = 28** şeklinde olacaktır.

7.2.7 Str ve Val

Sayısal giriş ve çıkış işlemleriyle ilgili olarak Pascal komutları arasında kullanılan Str ve Val deyimleri birbirlerinin tam tersi işlemleri yaparlar. Str, girilen sayısal karakterlerin karakter dizisine çevirirken Val, bunun tersine olarak karakter dizisi olarak girilen rakamsal bilgileri sayısal karakterlere dönüştürür. Grafik ekranda sayısal ifadelerin direkt olarak yazdırılması mümkün olmaması nedeniyle sayısal ifadenin **str** ile string ifadeye dönüştürülerek yazılması gerekir. Yine grafik ekrandan sayısal bilgi girişi yapılamamakta, ancak string olarak bilgi girişi yapılabilmektedir. String olarak girişi yapılan sayısal ifade **val** fonksiyonu ile sayısal ifadeye dönüştürülmesi gerekir.

Aşağıdaki örnekte klavyeden iki veri girilmektedir. İlk Val fonksiyonuyla k1 kontrol değişkeni O iken, S1 değişkenindeki rakam karakterleri sayısal karakterlere çevrilerek R1 ondalıklı sayısal değişkenine verilir. Aynı işlem ikinci Val fonksiyonuyla da yapılır. Elde edilen R1 ve R2 rakamları çarpılarak R3 değişkenine atanır ve uygun yazım formatıyla ekrana yazılmaktadır.

Örnek:

```
var  s1,s2:string;
     k1,k2:integer;
     r1,r2,r3:real;
begin
readln(s1);readln(s2);
Val(s1,r1,k1);  Val(s2,r2,k2);
r3:=r1*r2;
Writeln(r3:10:2);
end.
```

7.2.8. Pos

Pos fonksiyonu bir karakter dizisinin veya karakterin diğer bir karakter dizisi içinde olup olmadığını, varsa kaçıncı karakterden başladığını (Başladığı karakterin ana karakter dizisindeki sıra numarasını) verir. Aşağıdaki 1. örnek programda s isimli karakter dizisinde boşluk olup olmadığı aranmakta ve boşluk var ise bu karakter dizisinin önüne boşluk sayısınca 0 yerleştirilmektedir. 2. örnek programda i ismi verilen karakter dizisi içindeki j karakter dizisinin konumu araştırılmaktadır. 3. Örnekte ise k isimli bir karakter dizi içinde $ işareti olup olmadığı araştırılmaktadır.

Örnek 1:

```
Program Ornek  pos1;
var s: string;
begin
  s := '   123.5';
  { Boşlukları sıfıra çevir }
  while Pos(' ', s) > 0 do
    s[Pos(' ', s)] := '0';
Writeln(S);
Readln;
end.
```

Örnek 2:

```
Program Ornek  Pos2;
var i:string[28];j:String[12];
begin clrscr;
  i:='Balıkesir Üniversitesi';
  j:=' Üniversitesi';
  Writeln(i);Writeln(j);
  Writeln('i içinde J nin başlama konumu =',pos(j,i));
  Writeln;
Readln;
end.
```

Örnek 3:

```
Program Ornek  pos3;
var k:string;
begin
k:='123sadsdg&jdfjdsh$fdsf'
if pos('$',K)<>0 then
   begin
        writeln (' K   isimli karakter dizisi içinde $  işareti bulundu');
   end
else
```

```
            Writeln('Aranılan karakter bulunamadı');
        end;
    end.
```

Bu örnek programda aranılan alt karakter bulunamaz ise bu durumda fonksiyon sıfır değerine sahip olacak ve program else ile başlayan satıra dallanacaktır.

7.3 Sistem Birimi ve Giriş Çıkış Elemanları

Burada kısaca anlatılacak olan giriş/çıkış komutları Turbo Pascal dosyaları anlatılırken detaylı bir şekilde anlatıldığından burada sadece alfabetik sırada isimlerinin verilmesi ile yetinilecektir.

Giriş/çıkış elemanları çok genel olarak, programların çalıştırılmasıyla oluşacak dosyaların manyetik ortama yazdırılmalarıyla ilgili komut ve deyimler olarak tanımlanabilir. Bu deyim ve komutlar alfabetik sıraya göre; Append, Assign, Close, Erase, Read, Readln, Write, Writeln, Reset, Rewrite, Rename, Flush vb. 'dir.

7.4 Sistem Birimi ve Standart Matematik Fonksiyonlar

Turbo Pascal matematik ağırlıklı mühendislik bilimleri içinde son derece uygun bir dildir. Bir çok temel işlevi yerine getiren matematiksel fonksiyonlar Turbo Pascal'ın sistem birimi içerisinde tanımlanmıştır.

Turbo Pascal sistem birimi içinde standart olarak bulunan standart matematiksel fonksiyonlar ve kullanımları ile ilgili örnekler aşağıda alfabetik sırada verilmiştir.

7.4.1 Abs

Herhangi bir sayısal sabit veya değişkenin mutlak (işaretsiz) değerini verir.

Örnek:

```
Program ABS  fonksiyonu;
var
  r: Real;
  i: Integer;
begin
  r := Abs(-4.3);  i := Abs(-157);
Writeln(r:4:1,i:8);
end.
```

7.4.2 ArcTan

Aşağıdaki örnekte R isimli real parametre ile verilen argümanın radyan cinsinden arktanjantını verir. Radyanı dereceye çevirmek için 180/pi ile çarpmak gereklidir.

Örnek:

```
Program ArcTan;
var
 R: Real;
begin
 r:=1;
 R := ArcTan(r)*180/pi;
writeln(r:5:1);
end.
```

7.4.3 Cos

Derece olarak verilen bir açı değerinin radyana çevrilmesiyle trigonometrik kosinüsünü hesaplar. Açı değerini radyana çevirmek için pi/180 ile çarpmak gerekir.

Örnek:

```
Program cosinus  Fonksiyonu;
var
 r,Aci: Real;
begin
 aci:=90;
 R := Cos(aci*pi/180);
 writeln(r:5:1);
end.
```

7.4.4 Exp

Exp fonksiyonu doğal logaritma parametresi olan e'nin üstel değerini hesaplar. Aşağıdaki örnek program parçasında e'nin 1. üstel değeri (e^1) hesaplanılmaktadır.

Örnek:

```
Program Exp  Fonksiyonu;
begin
  Writeln('e = ',Exp(1.0));
end.
```

7.4.5 Frac

Gerçel bir sayının desimal noktadan sonraki kesirli kısmını verir.

Örnek:

```
Program frac  fonksiyonu;
var
 f1,f2: Real;
begin
 f1 := Frac(4233.1987);
 f2 := Frac(-4233.1987);
Writeln(f1:6:4,'  ',f2:6:4);
end.
```

7.4.6 Int

Gerçel sayıların tamsayı bölümünü integer tipinde verir.

Örnek:

```
Program Int  Fonksiyonu;
var f1,f2: Real;
begin
   f1 :=Int(4233.1987);
   f2 :=Int(-4233.1987);
   Writeln(f1:6:0,f2:6:0)
end.
```

7.4.7 Ln

Girilen bir sayının e tabanına göre logaritmasını alır.

Örnek:

```
Program Ln  Fonksiyonu;
var
    e,mn: real;
begin
   mn:=2;
  e := Exp(mn);
  Writeln('ln(e) = ',ln(e));
end.
```

7.4.8 Odd

Tamsayı parametrelerin tek sayı olup olmadığının tesbitinde kullanılır.

Örnek :

```
Program odd  fonksiyonu;
uses crt;
var i:byte;
begin clrscr;
  for i:=1 to 20 do
  begin
   if Odd(i) then
     Writeln(i,' Tek Sayıdır ')
   else
    Writeln(i, ' Çift Sayıdır');
   end;
delay(2000);
end.
```

7.4.9 Ord

Char, byte, integer, boolean vb. şeklinde bildirilmiş parametrelerin tamsayı değerlerini verir. Örneğin Char ile kullanılması durumunda karakterin ASCII kodu elde edilir.

Örnek :

```
Program ord  Fonksiyonu;
var ch:char;
begin
  for ch:='A' to 'Z' do
  Writeln(Ch,' icin ASCII Kod ',Ord(ch));
  Readln;
  for ch:='a' to 'z' do
  Writeln(Ch,' icin ASCII Kod ',Ord(ch));
end.
```

7.4.10 Pi

Pi olarak bilinen matematiksel sabitin değerini verir. Pascal programında diğer programlama dillerinde olduğu gibi pi sayısının ayrıca tanıtılmasına gerek yoktur.

Örnek :

```
Program Pi  fonksiyonu;
var d:real;
begin
Write('Dairenin çapını giriniz :');
  Readln(D);
  Writeln('Dairenin çevresi = ',Pi*D:10:5:);
end.
```

7.4.11 Random

Rastgele bir sayı üretir. Üretilen rastgele sayı random ile belirtilen sayıdan küçük olmaktadır.

Örnek:

```
Program Random  Fonksiyonu;
uses Crt; {Windows için Wincrt}
begin
  Randomize;
  repeat
    Writeln (Random(99));
  until KeyPressed;
end.
```

7.4.12 Round

Gerçel sayıların kurala uygun bir şekilde yuvarlatır ve yuvarlatılmış tamsayı olarak verir. Elde edilen yuvarlatılmış sayı yine real tipte saklanır.

Örnek:

```
Program Round  fonksiyonu;
begin
  Writeln(1.499, '  ',Round(1.499),' a yuvarlatıldı');
  Writeln(1.5, '  ',Round(1.5),' a yuvarlatıldı');
  Writeln(1.499, '  ',Round(-1.499),' a yuvarlatıldı');
  Writeln(-1.5, '  ',Round(-1.5),' a yuvarlatıldı');
end.
```

7.4.13 Sin

Derece olarak verilen bir açı değerinin radyana çevrilmesiyle trigonometrik sinüsünü hesaplar. Açı değerini radyana çevirmek için pi/180 ile çarpmak gerekir.

Örnek :

```
Program sinus  Fonksiyonu;
var
 r,Aci: Real;
begin
 aci:=90;
 R := Sin(aci*pi/180);
 writeln(r:5:1);
end.
```

7.4.14 Sqr

Verilen bir parametrenin karesini verir. Genel kullanım şekli;

```
Kare:=sqr(sayi);
```

7.4.15 Sqrt

Verilen bir parametrenin karekökünü verir. Genel kullanım şekli;

```
Karekok:=sqrt(sayi);
```

Örnek:

```
Program Sqr  ve  sqrt;
var say:real;
begin
  Write('Karesi alınacak sayıyı giriniz  =');
  Readln(say);
  Writeln(say:10:2,' in Karesi   ', Sqr(say):10:2);
  Writeln(Sqr(say):10:2,' nın karekökü',Sqrt(sqr(say)):10:2);
end.
```

7.4.16 Trunc

Verilen bir gerçel sayının tamsayı kısmını verir ve sonucu tamsayı tipinde saklar.

Örnek:

```
Program Trunc  Fonksiyonu;
var f1,f2,f4:real;
    f3:integer;
begin
   f1:=34.56;f2:=11.9;
   f4:=f1/f2;
   f3:=Trunc(f1/f2);
   Writeln(f4:8:2,'  ',f3);
end.
```

Örnek: Aşağıda verilen programda round ve trunc deyimlerinin birlikte kullanımları verilmiştir.

```
Program  round  ve  trunc;
Var
Vize,final,Ort:integer;
Begin
   Readln(vize,final);
   Ort:=Trunc(round(vize*0.4+final*0.6));
   Writeln(Vize:10,Final:10,Ort);
End.
```

BÖLÜM 8

KONTROL DEYİMLERİ

8.1 Giriş

Kontrol komutları olarak; GOTO, IF ve CASE deyimlerini inceleyeceğiz. Bunlardan GOTO deyimi programın akış yönünü değiştirmek için, CASE ve IF deyimleri belirli bir şartın doğru veya yanlış olmasına bağlı olarak programın bir parçasının çalıştırılmasını sağlamak için kullanılır.

8.2 GOTO Deyimi

Programın çalışma yönünün değiştirilmesi için kullanılan GOTO deyiminin genel kullanımı şu şekildedir;

GOTO Etiket;

GOTO deyimi ile kullanılan etiket, çalışma akışının yönlendirileceği noktayı gösterir. Bu etiketin programın tanım bloklarının olduğu yerde LABEL adı verilen bölümde tanımlanması gereklidir.

GOTO deyiminin kullanıldığı program bloğu içinde bulunması zorunludur. Etiket, bir komut cümlesinin yani bir program noktasının etiketlenmesi için yazıldığında kendisinden sonra " : " karakterinin yazılması zorunludur.

GOTO deyimleri ile ana program ve alt programlar arasında veya bir alt program ile başka bir alt program arasında dallanma yapılamaz. Ayrıca, IF, CASE, FOR, REPEAT, ve WHILE deyimlerinin blokları içine dallanılamaz. Ancak bu deyimler ile kullanılan bloklar içinden, dışarıya dallanma yapılabilir.

Turbo PASCAL' daki kontrol ve döngü deyimlerinin etkinliği ve PASCAL programlarının yapısallığı GOTO deyimine ihtiyacı ortadan kaldırır. Basic ve Fortran' da yazılan programlarda bol miktarda görülen GOTO deyimlerine Pascal programlarında pek rastlanmamaktadır.

Örnek:Aşağıdaki program, 1' den n sayısına kadar olan sayıların toplamını hesaplamaktadır. Program içinde üç kez GOTO deyimi kullanılmıştır. GOTO devam şeklindeki komut cümlesi şartsız, diğerleri ise şartlıdır.

```
Program Dallanma;
Uses crt; {Windows için Wincrt}
Var
  I,N : INTEGER;
  Top : REAL;
  Cev : Char;
Label
  Basla, Devam, Son;
Begin
  Başla:
    Clrscr;
    I:=0;
    Top:=0;
    Write('Bir sayı giriniz=');
    Readln (N);
  Devam:
    I:=I+1;
    Top:=Top+1;
    IF I=N then goto son;
    GOTO DEVAM;
  Son:
    Writeln('1 den', N,'e kadar sayıların toplamı=', Top:5);
    Writeln;
    Write ('Devam Edecek misiniz :');
    Readln (Cev);
    IF Cev In ['E' , 'e'] then GOTO Basla;
End.
```

8.3 IF Deyimi

IF deyimi, bir şartın doğru veya yanlış olmasına bağlı olarak programın belirli parçalarının çalışmasını veya çalışmamasını sağlar. IF deyimi,

```
IF şart cümlesi THEN Blok1;
veya
IF şart cümlesi THEN Blok1 ELSE Blok2;
```

şeklinde kullanılabilir.
Buradaki şart cümlesi, birbirlerine mantıksal operatörler ile bağlanmış bir veya birkaç ilişkisel operasyon veya bir tek boolean ifade olabilir. Aşağıdaki örnekleri inceleyelim:

```
IF A=B THEN ...
IF (A=B) AND (A=C) THEN ...
IF (A+1) < (B-1) THEN ...
IF (Cev In ['E' , 'e']) THEN ...
```

Then sözcüğünü takiben, bloklar birden fazla komut cümlesinden oluşurlar ise, bu blokların BEGIN ve END deyimleri içine alınması zorunludur. IF deyiminin ikinci şekli yani ELSE' den sonra gelen komut veya komutlar dizisi geçerlilik kazanmışsa, ELSE' den önce gelen komut cümlesinin yada, End deyiminin sonuna ';' işareti konulmaz. Aşağıdaki örneklerden 1.si hatalı 2.si doğrudur.

```
IF A=B Then Writeln ('Eşit'); Else Writeln ('Farklı');
IF A=B Then Writeln ('Eşit') Else Writeln ('Farklı');
```

IF deyimi içinde kullanılan Blok1 ve Blok2 çeşitli komut cümlelerinden oluşabileceğine göre bu blokların içinde başka IF cümleleri de yer alabilir. IF deyimlerinin bu şekilde kullanılmasına iç içe IF deyimleri adı verilir. IF cümlesinde yer alan her iki bloğun geçerlilik kazanması söz konusu olamaz.

```
IF degisken  1 <kosul> degisken  2   then
  begin
    -----
    islemler
    -----
  end;
```

Bu kullanım şekli IF deyiminin en yalın şekillerinden biridir. Degisken 1 ile Degisken 2 arasındaki koşulun durumuna göre THEN ifadesinden sonra gelen işlemler yapılır.

Örnek: Aşağıda verilen kodlarına göre daire, üçgen ve dikdörtgenin çevresini hesaplayan Pascal programını yazınız**.**

```
Üçgen
Dikdörtgen
Daire

Program Cevre  Hesabi;
 uses crt; {Windows için Wincrt}
 var
 sekil:byte;
 uzun1,uzun2,uzun3:real;
 cevre:real;
 Begin
    Writeln('1. Üçgen');
    Writeln('2. Dikdörtgen');
    Writeln('3. Daire');
    Write('Şeklin Kodunu Giriniz');
    Readln(Sekil);
```

```
If Sekil=1 then
  begin
    Write('Uçgenin 1. Kenar Uzunluğunu giriniz  :');readln(Uzun1);
    Write('Uçgenin 2. Kenar Uzunluğunu giriniz  :');readln(Uzun2);
    Write('Uçgenin 3. Kenar Uzunluğunu giriniz  :');readln(Uzun3);
    Cevre:=uzun1+uzun2+uzun3;
  end;
If Sekil=2 then
  Begin
    Write('Dikdörtgenin Kısa Kenar Uzunluğunu giriniz:') ;
    readln(Uzun1);
    Write('Dikdörtgenin Uzun Kenar Uzunluğunu giriniz  :');
    readln(Uzun2);
    Cevre:=2*(uzun1+uzun2);
  End;
If Sekil=3 then
  Begin
    Write('Dairenin yarıçapını giriniz  :');readln(Uzun1);
    Cevre:=2*pi*Uzun1;
  End;
Writeln('Seçilen Nesnenin Çevre Uzunluğu  =',Cevre:8:2);
End.
```

Örnek : Aşağıda verilen kodlara göre birim dönüşümlerini veren Pascal programı yazınız.

```
Fahrenhayt 'tan (F)  Santigrad 'a (C)        :  C=5/9*(F-32)
İnç 'ten (I) Santimetre 'ye (Cm)             :  Cm=2.54*I
Mil 'den (M) Kilometre 'ye (Km)              :  Km=1.6*M
Pound 'dan (P) Kilogram 'a (Kg)              :  Kg=0.45*P
```

```
Program metrik  cevrimler;
uses crt; {Windows için Wincrt}
var kod:byte;
F,C,I,Cm,M,Km,P,Kg:real;
Begin
Writeln('Çevrim Kodları');
Writeln('1. Fahrenhayt -> Santigrad');
Writeln('2. İnç -> Santimetre ');
```

```
        Writeln('3. Mil ->Kilometre ');
        Writeln('4. Pound -> Kilogram ');
        Write('Çevrim Kodunu Giriniz  ='); Readln(Kod);
        If Kod=1 then  begin
           Write('Fahrenhayt Değerini Giriniz  ='); Readln(F);
           C:=5*(F-32)/9;
           Writeln('Santgrad Değeri  =',C:7:2,' °');
         end;
        If Kod=2 then
         Begin
           Write('İnç Değerini Giriniz  ='); Readln(I);
           Cm:=2.54*I;
           Writeln('Santimetre Değeri  =',Cm:7:2);
         end;
        If Kod=3 then
         Begin
           Write('Mil Değerini Giriniz  ='); Readln(M);
           Km:=1.6*M;
           Writeln('Kilometre Değeri  =',Km:7:2);
         end;
        If Kod=4 then
         Begin
           Write('Pound Değerini Giriniz  ='); Readln(P);
           Kg:=0.45*P;
           Writeln('Kilogram Değeri  =',Kg:7:2);
         End;
      End.
```

8.3.1 If Then Else Yapısı

```
IF degisken 1 <kosul> degisken 2  then
  begin
    -----
    islemler
    -----
  end
 ELSE
  begin
    -----
    islemler
    -----
  end;
```

Bu yapı kullanıldığında, Degisken 1 ile Degisken 2 belirtilen koşulu karşıladıkları zaman THEN ifadesinden sonra gelen işlemler, aksi halde ELSE ifadesinden sonra tanımlanan işlemler yaptırılır.

```
IF degisken  1 <kosul> degisken  2   then
   begin
       islemler1;
   end
   else
if degisken  3 <kosul> degisken4 then
    begin
      islemler2;
    end
   else
 if degisken  5 <kosul> degisken6 then
    begin
   islemler3;
    end;
```

Örnek: Aşağıdaki program klavyeden girilen yılın kaç gün olduğunu vermektedir. Programı inceleyiniz.

```
program deneme;
uses crt; {Windows için Wincrt}
var
yil:integer;
begin
     clrscr;
     Write('Gun Sayisini ogrenmek istediginiz yili giriniz :');
     readln(yil);
     if (yil mod 4 )=0 then
     begin
          Writeln('Yil   366  gun');
         End
         Else
         Begin
           Writeln ('Yil   365  gun');
         end;
     readln;
end.
```

Örnek: Aşağıdaki program klavyeden girilen 3 sayıyı büyükten küçüğe sıralamaktadır.

```
program iciceif;
uses crt; {Windows için Wincrt}
Var   a,b,c:integer;
        min,ort,max:integer;
begin
     Clrscr;
     Write('A Sayisi =');
```

```
            Readln(A);
            Write('B Sayisi =');
            Readln(B);
            Write('C Sayisi =');
            Readln(C);
            if a<=b then
              if a<=c then
                if b<=c then begin
                            min:=A;
                            ort:=B;
                            max:=C;
                           end
                           else begin
                            Min:=A;
                            ort:=c;
                            max:=b;
                      end
                    else begin
                         min:=c;
                         ort:=A;
                         max:=b;
                         end
                    else
                    if b<=C then
                          if a<=C then begin
                                    min:=b;
                                    ort:=A;
                                    max:=c;
                                    end
                               else begin
                                    min:=b;
                                    ort:=c;
                                    max:=a;
                                    end
                           else begin
                                min:=c;
                                ort:=b;
                                max:=a;
                                end;
                    Writeln('Kucuk sayi=',Min);
                    Writeln('Orta Sayi =',Ort);
                    Writeln('Buyuk Sayi=',Max);
        END.
```

8.4 Case ... Of

Bir değerin birden fazla değer ile karşılaştırmasını yapan ve bir eşitliğin bulunması halinde belli program parçalarının çalıştırılmasını sağlayan CASE deyiminin genel formu aşağıda verilmiştir.

```
CASE  Kontrol Değişkeni OF
      Etiket1 : Blok1 ;
      Etiket2 : Blok2 ;
      .
      .
      Etiketn : Blokn ;
     ELSE BLOK;
    END;
```

Kontrol değişkeni real haricindeki standart tiplerden birine sahip olan bir değişkendir. CASE deyimi içindeki etiketler LABEL olarak tanımlananlardan farklıdır. Bunlar sabit değerler olup, kontrol değişkeni ile aynı tipe sahiptir. Etiketi takip eden bloklar ise PASCAL komut cümlelerinden meydana gelir. Komut cümleleri birden fazla ise BEGIN END deyimleri arasına yazılması zorunludur. Etiketlerin dışında yazılan ELSE deyimi ve bunu takip eden blok seçimliktir.

Örnek : Aşağıda kişinin yaşı girildiğinde, yaşına uygun mesajlar veren bir Pascal programı verilmiştir.

```
Program Yasdilim;
Uses wincrt; {Dos için CRT}
Var yas:integer;
Begin
  Clrscr;
    Write('Yaşınızı Giriniz  :');
     ReadLn(YAS);
  CASE yas OF
     0..5:Writeln('Yaşınız 0-5 arasında');
     5..15:Writeln('Yaşınız 5-15 arasında');
     15..35:Writeln('Yaşınız 15-35 arasında');
     35..50:Writeln('Yaşınız 35-50 arasında');
     50..85:Writeln('Yaşınız 50-85 arasında');
  else Writeln('Çok Yaşayın');
  end;
 Readln;
End.
```

Örnek : Bir diyet uzmanı insanlara , geliştirmiş olduğu değişik üç tür diyeti uygulamayı düşünmektedir.

1.tür diyet 60 kilo ve altındaki insanlara uygulanacak kilo aldırıcı program.
2.tür diyet 61 ile 80 kilo arasındaki insanlara uygulanacak kilo aldırmayan program.
3.tür diyet 81 ile 150 kilo arasındaki insanlara uygulanacak kilo zayıflatıcı programdır
Buna göre 10 kişi arasında yapılan bir anketle ,bu üç programa kaçar kişinin düştüğünü ve bu gruplardaki ortalama kiloları bulan bir PASCAL programını yazınız.

```
program rejim  anket;
uses crt; {Windows için Wincrt}
var kilo,grp1,grp2,grp3,i,j,k:integer;
   grp1ort,grp2ort,grp3ort:real;
   grp1top,grp2top,grp3top:integer;
BEGIN
 for i:=1 to 10 do
   begin
     write('Kişinin Kilosunu giriniz :');Readln(Kilo);
     case kilo of
          0..60: begin
                    grp1:=grp1+1;
                    grp1top:=grp1top+kilo;
                 end;
        61..80: begin
                    grp2:=grp2+1;
                    grp2top:=grp2top+kilo;
                 end;
        81..150: begin
                    grp3:=grp3+1;
                    grp3top:=grp3top+kilo;
                 end;
       end;
    end;
    grp1ort:=grp1top/grp1;
    grp2ort:=grp2top/grp2;
    grp3ort:=grp3top/grp3;
Writeln('1. Gruptaki kişi sayısı  =',grp1,'  1. Gruptaki kilo ortalaması  =',grp1ort:8:2);
Writeln('2. Gruptaki kişi sayısı  =',grp2,'  2. Gruptaki kilo ortalaması  =',grp2ort:8:2);
Writeln('3. Gruptaki kişi sayısı  =',grp3,'  3. Gruptaki kilo ortalaması  =',grp3ort:8:2);
repeat until keypressed;
End.
```

Yukarıda verilen programda; *for i:=1 to 10 do* satırı ile yapılan işlemin 10 kez tekrar edileceği belirtilmektedir. (For .. do yapısı için 9. Bölüme bakınız.)

Örnek : Kare, dikdörgen, üçgen ve daire alan hesabı yapan bir Pascal programı hazırlanacaktır. Programda alanı hesaplanacak nesneyi belirttikten sonra ilgili alanın hesaplanabilmesi için CASE...OF yapısı kullanılacaktır.

```
program Sekillerin  alan  hesabi;
uses Crt; {Windows için Wincrt}
var Ch : char;
   uzun,gen,yuk,Taban,ycap: real;
   Alan  Kare,Alan  Dortgen,Alan  Ucgen,Alan  Daire:real;
   i:byte;
begin  (* Ana program *)
     Writeln;
     Writeln('Lütfen Alan Hesabı Yapacağınız Nesneyi seçiniz :');
     Writeln;
     Writeln('[K]are');
     Writeln('[D]ikdörtgen');
     Writeln('[U]çgen');
     Writeln('d[A]ire');
     Writeln('[C]ıkış');
     Write('Alan hesabı yapılacak nesne (K/D/U/A) ?=');
     Readln(Ch);
    case UpCase(Ch) of
     'K' : begin
           Write('Karenin kenar uzunluğu =');
           Readln(uzun);
           Alan  Kare:= uzun*uzun;
           Writeln('Alan ',Alan  kare:12:4);
           end;
     'D' : begin
           Write('Dörtgenin Genişliği  = ');
           Readln(gen);
           Write('Dörtgenin Yüksekliği =');
           Readln(yuk);
           Alan  Dortgen:=gen*yuk;
           Writeln('    Dörtgenin Alanı= ',Alan  Dortgen:12:4);
           end;
     'U' : begin
           Write('Üçgenin Taban Uzunluğu =');
           Readln(Taban);
           Write('Üçgenin Yüksekliği     =');
           Read(yuk);
           Alan  Ucgen:=taban*yuk/2;
           Writeln('Üçgenin Alanı      = ',Alan  Ucgen:12:3);
           end;
     'A' : begin
           Write('Dairenin Çapı  =');
           Readln(Ycap);
```

```
            Alan  Daire:= sqr(ycap)*Pi;
            Writeln('Dairenin Çapı    =',Alan  Daire:12:3);
            end;
       'C' : begin Writeln('Bitti');exit; end;
       else Writeln(' Tanımsız giriş');
   end;
end.  (* Ana program sonu *)
```

BÖLÜM 9

TEKRARLAMA DEYİMLERİ

9.1 Giriş

Bu bölümde program içerisinde belirli blokların herhangi bir şarta bağlı olarak veya şarttan bağımsız bir şekilde ardışık olarak çalıştırılması için kullanılan deyimler üzerinde durulacaktır. Bu tekrarlama deyimleri FOR-DO, REPEAT-UNTIL, WHILE-DO şeklindedir.

9.2 For-Do

For deyimi, bir program parçasının herhangi bir boolean şartına bağlı olmaksızın belirlenen sayıda üstüste çalıştırılması için kullanılır. For deyiminin genel kullanım şekilleri aşağıda verilmiştir.

1. **FOR** *Kontrol Değişkeni* :=*Başlangıç Değeri* **TO** *Son değer* **DO**
   Begin
   İşlemler
   .
   .
   .
   End;

Bu kullanım şeklinde başlangıç değeri bitiş değerinden küçük olmak zorundadır. TO ifadesiyle başlangıçtan bitişe kadar artarak tekrar yapılacağını, DO ifadesi de tanımlanan işlemlerin tekrarlanacağını bildirir.

2. **FOR** *Kontrol Değişkeni* :=*Başlangıç Değeri* **DOWNTO** *Son değer* **DO**
Begin
İşlemler
End;

For deyiminin bu kullanım şeklinde başlangıç değeri bitiş değerinden daima büyük olup DOWNTO ifadesi de döngü değişkeninin tekrarlama işleminde azalacağını belirtir. DO ifadesi ise belirtilen işlemlerin döngü sayısınca tekrarlanacağını bildirir.

Turbo Pascal dilinde diğer programlama dillerinden farklı olarak başlangıç değerinden son değere artışlar/azalmalar birer birer olmaktadır. Aşağıdaki örnek programları inceleyiniz.

Örnek:

```
Program Hesap  plani;
Uses Crt; {Windows için Wincrt}
var       i:byte;
          Hesapkod:String[10];
          Hesapadi:String[20];
begin   clrscr;
          for i:=1 to 5  do
          begin
             Write('Hesap Kodu .....:');Readln(HesapKod);
             Write('Hesap Adı.....:');Readln(HesapAdi);
          end;
Readln;
end.
```

Örnek:Aşağıdaki örnek Program; 1'den 8'e kadar artan ve 9'dan 3'e kadar birer birer azalan içiçe for do döngülerinin kullanımını göstermektedir.

```
Program icice  for  ornek;
Uses crt; {Windows için Wincrt}
Var
i,j:byte;
Begin
          For i:=1 to 8 do
          Begin
                    for j:=9 downto 3 do
                    Begin
                              Write(i*j:6);
                    end;
          Writeln;
```

```
            end;
Readln;
End.
```

Örnek: 'A' dan 'Z' ye kadar büyük harfleri ekrana yazdıran Pascal programı.

```
Program Odev;
var ch:=Char;
begin
Writeln('BÜYÜK HARFLER');
for ch:='A' to 'Z' DO
        Write(Ch,' ');
end.
```

Örnek. Klavyeden girilen bir ifadeyi tersten yazdıran program

```
program terstenyazma;
uses crt; {Windows için Wincrt}
var
mesaj:string;
i,l:byte;
begin
    Write('Bir mesaj yazınız');
    Readln(mesaj);
    l:=length(mesaj);
    Writeln(L);
    for i:=L downto 1 do
        Write(Copy(mesaj,i,1));
end.
```

9.3 Repeat-Until

Bir program bloğunun belli bir şart sağlanıncaya kadar üst üste çalıştırılmasını sağlayan REPEAT deyiminin genel formu aşağıdaki şekildedir.

```
Repeat
.
   Program Satırları
.
Until (Boolean Şartı)
```

Burada repeat, tekrar etme anlamında olup, tekrar etme işi UNTIL deyimindeki boolean ifadesi sağlanıncaya kadar devam eder. Bu deyim program satırları bölümüne herhangi bir şey yazılmaksızın kullanılabilir.

Bu döngünün en büyük avantajı belirli bir sayı ile sınırlandırılmamış olmasıdır. Boolean ifadesindeki şart sağlanıncaya kadar işlemlere devam edilmektedir.
Dikkat edilirse şart cümlesinin aldığı değer ne olursa olsun program bloğu bir kez çalışmaktadır.

Örnek : Aşağıda verilen Pascal programı karton fabrikasındaki kenar kesme ünitesinde, kesme makinasından çıkan kartonların ortalama ağırlıklarını hesaplamaktadır. Karton ağırlığı olarak 0 girildiğinde programın çalışması sona ermektedir.

```
Program ornek  repeat;
uses wincrt;
var
   i,sayi:integer;
   top,ort:real;
begin
     clrscr;
     i:=0;top:=0;ort:=0;
     Repeat
          i:=i+1;
          Write(i,'. Kartonun Ağırlığını Giriniz =');
          Readln(sayi);
          Top:=top+sayi
     Until (Sayi=0);
     ort:=top/(i-1);
     Writeln;Writeln;
     Writeln(i-1,'  Adet Kartona ait Ağırlık Ölçümü Yapılmıştır.');
     Writeln(i-1,' Adet Kartonun Toplam Ağırlığı =',top:6:3);
     Writeln('Girilen sayıların ortalaması    =',ort:6:3);
  repeat until KeyPressed;
end.
```

9.4 While-Do

Bir program bloğunun belli bir şart sağlandığı sürece üst üste icrasını sağlayan WHILE deyiminin genel yazılış şekli aşağıdadır.

While <şart cümlesi> **Do** BLOK

Do kelimesini takibeden blok WHILE deyimi tarafından döngüye sokulacak komut cümlelerini kapsar. Komut cümlesi sayısı birden fazla ise, bu bloğun BEGIN...END deyimleri arasına alınması zorunludur.

WHILE ile REPEAT arasındaki fark; Repeat döngüsü şart cümlesi yanlış olduğu sürece, While döngüsü ise şart cümlesi doğru olduğu sürece devam etmesidir.

Örnek : Repeat -Until ile yapılan örnek programı WHILE-DO ile yapalım.

```
program ornek  While;
uses wincrt;
var
   i,sayi:integer;
   top,ort:real;
begin
     clrscr;
     Write('1. Kartonun Ağırlığını Giriniz =');
     Readln(sayi);
     i:=1;top:=0;ort:=0;
     While Sayi>0 do
     begin
          Top:=top+sayi;
          i:=i+1;
          Write(i,'. Kartonun Ağırlığını Giriniz =');
          Readln(sayi);
     end;
ort:=top/(i-1);
     Writeln;Writeln;
     Writeln((i-1),'  Adet Kartona ait Ağırlık Ölçümü Yapılmıştır.');
     Writeln((i-1),' Adet Kartonun Toplam Ağırlığı =',top:6:3);
     Writeln('Girilen sayıların ortalaması    =',ort:6:3);
repeat until KeyPressed;
end
```

9.5. Blok ve Döngülerin Kırılması

Döngülerin çalışması sırasında belirli koşulların sağlanması durumunda döngünün sona ermesini isteyebiliriz. Bunun için aşağıda örneklerle açıklamaya çalıştığımız BREAK, CONTINUE, EXIT, HALT Pascal deyimlerinden yararlanmaktayız. Bu deyimlerden BREAK ve CONTINUE Pascal 7.0 ile birlikte C programlama dilinden alınmıştır.

9.5.1 Break

Turbo Pascal 7.0 programlama dilinde bir döngüyü kırarak sona erdirmek amacıyla kullanılır. Program içinde BREAK deyimiyle karşılaşıldığında içinde bulunduğu döngüden sonraki program satırının işler hale getirir. BREAK komutu FOR-DO, REPEAT-UNTIL ve WHILE-DO döngülerinin içinde kullanılabilir.

Konunun daha iyi anlaşılabilmesi için aşağıdaki örneği inceleyiniz. Örnek Programda i ve j gibi iki sayının çarpımı yapılmaktadır. i=j olduğu durumda içteki döngü BREAK deyimi ile kırılmakta ve i 'nin değeri bir üst değere arttırılarak programın çalışmasına devam edilmektedir.

Örnek:

```
Program break  kullanimi;
uses crt; {Windows için Wincrt}
var i,j:integer;
  begin
    clrscr;
    for i:=1 to 10 do
      begin
        Writeln(i,'.  değeri ');
          for j:=1 to 10 do
           begin
             if i=j then break; (* i=j olduğu takdirde içteki döngünün çalışması sona  eriyor*)
             Writeln(i:3,' * ',j:3,'= ',i*j:3);
           end;
        Writeln('Devam Etmek için ENTER tuşuna Basınız');Readln;
      end;
  end.
```

9.5.2 Continue

Fortran programlama dilinde olduğu gibi tekrarlama çevrimini yeniden başlatır. Programcı, bilgi girişlerinde kullanıcının sayısal olmayan bir giriş yapacağını varsayarak önlem almak amacıyla tekrarlama işlemlerini yeniden başlatmak için CONTINUE komutu kullanılır. Bu deyim PASCAL 7.0 ' da geçerlidir.

Örnek:

```
Program Continue  Kullanimi;
Uses Crt; {Windows için Wincrt}
Const
  n=5;
var
```

```
  sayac,i:integer;
  rakam :Array [1..n] of Real;
  toplam:Real;
begin
  clrscr;toplam:=0; sayac:=1;
  While sayac < n  do
          begin
              Write(Bir Sayı Giriniz:');
              {$I-}
              Readln(Rakam[sayac]);
              {$I+}
              If   IOResult >0 then
                        begin
                            Writeln('Rakam girmelisiniz');
                            CONTINUE;
                         end;
              sayac:=sayac+1
          end;
  For   I:=1 to sayac do
          toplam:=toplam+rakam[i];
  Writeln('Toplam Sayı        :',Toplam:7:2);
end.
```

Örnek program çalıştırıldığında kullanıcının hatalı bir giriş yapması halinde *Run Time Error* hatası oluşacaktır. Bu hatalı durum IORESULT hata durumu fonksiyonu ile kontrol ettirilerek programın kırılması önlenir ve CONTINUE komutu tekrarlama çevrimini tekrar başlatır. Ancak hata yoklama rutinleri {$I-} ile pasif duruma düşürülmüştür. IORESULT fonksiyon değeri kontrolü yapabilmek için {$I-} derleme komutu verilerek Pascal hata yoklama rutinleri pasif duruma alınmalıdır. Ancak kontrol işlemi bittikten sonra hata yoklama rutinleri {$I+} ile tekrar aktif duruma getirilmelidir.

9.5.3 Exit

Program işlem bloklarında tanımlanan herhangi bir koşulun gerçekleşmesi halinde program bloğunun akışını durdurur.

Örnek:

```
uses Crt; {Windows için Wincrt}
begin
  repeat
    if KeyPressed then Exit;
    Write('Xx');
  until False;
end.
```

9.5.4 Halt

Programın kararlaştırılan bir yerinde durdurulmasını sağlar. ENTER tuşuna basıldıktan sonra varsa HALT komutundan sonra tanımlanan program satırları çalıştırılır.

Örnek :

```
program halt  ornek;
begin
  if 1 = 1 then
    begin
      WritelN('Halt Deneme 1');
      if 2 = 2 then
        begin
          WritelN('Halt Deneme 2');
          if 3 = 3 then
            begin
              WritelN('Halt Deneme 3');
              Halt(1);
            end;
        end;
    end;
  Writeln(' Bu Çalıştırılamayacaktır');
  readln;
end.
```

9.6 ÖRNEK PROGRAMLAR

1. Verilen bir X değerine karşılık $\sum_{a=1}^{N}(3\frac{2a-1}{a}X)$ ifadesini hesaplayacak bir PASCAL programı yazılacaktır. Programda formülasyonun hesaplanması için REPEAT....UNTIL döngüsü kullanılacaktır. X ve N sayısı programcı tarafından girilip, N ve X'in $0 < N < 20$ ve $0 < X \leq 1$ aralıklarında olması gerekmektedir. Hatalı veri girildiği takdirde, veri girişinin yeniden yapılabilmesi için programda gerekli işlemleri de yapılmıştır.

```
program cozum01;
uses crt; {Windows için Wincrt}
var
   a,n:integer;
   x:real;
   topla,islem:real;
begin
islem:=0;
```

```
 CLRSCR;
islem:=0;
Repeat
  Write('X değerini giriniz  =');
  Readln(X);
Until ((x>0) and (x<=1));
Repeat
 Write('N degerini giriniz  =');
 Readln(N);
Until ((N>0) and (N<20));
a:=0;
Repeat
     a:=a+1;
     topla:=3*(2*a-1)*x/a;
     islem:=islem+topla;
until (a=N);
Writeln;Writeln('islemin sonucu=',islem:6:1);
repeat until keypressed;
end.
```

y=f(x) olmak üzere y' nin değeri x' in alacağı değerlere göre değişmektedir.

$0 < x < 1$ $\qquad y = 3x^2 - \sqrt{3x^2 + 0.5x}$

$1 \le x < 2$ $\qquad y = 3x - \sqrt{x}$

$x \ge 2$ $\qquad y = \dfrac{\sqrt{12x^2}}{2\sqrt{2x}}$

Buna göre **x**'in alacağı değerlere göre **y**' nin alacağı değerleri hesaplattırarak, uygun formatta ekrana yazdıran ve sonucu herhangi bir tuşa basıncaya kadar ekranda tutan PASCAL programını yazınız. Programda x=>0 ve y<5 olup olmadığının kontrolünün yapılması gerekmektedir. Hatalı veri girildiği takdirde, veri girişinin yeniden yapılabilmesi için programda gerekli işlemleri de yapınız.

```
Program cozum02;
uses crt; {Windows için Wincrt}
var
x,y:real;
begin
 repeat
    clrscr;
  repeat
    Write('X değerini giriniz =');
```

```
    Readln(X);
  until (x>0);
 if (0<x) and (x<1) then y:=3*x*x-(sqrt(3*sqr(x)+0.5*x));
  if (1<=x) and (x<2) then y:=3*x-2*sqrt(x);
  if (x>=2) then y:=(SQRT(12*sqr(x)))/(2*sqrt(2*x));
  Writeln('y =',y:10:5);
  if y>5 then
  begin
    Writeln('Y değeri istenilen değerin üstünde tekrar deneyiniz...');
    Writeln('Devam Etmek için herhangi bir tuşa basınìz');
    repeat until keypressed;
   end;
 until y<5;
end.
```

3. X tam sayı olmak üzere, *f(x)* kesikli fonksiyonu aşağıdaki aralıklarda tanımlıdır.

$20 < x \leq 30$ $\quad f(x) = x^5 - x^{7/2}$

$30 < x \leq 40$ $\quad f(x) = x^{5/2} - x^2$

$x > 40$ $\quad f(x) = x^{1/2} + x^{3/5} - x^2$

Klavyeden girilen 10 tane *x* değişkeni için *f(x)* kesikli fonksiyonunu hesaplayarak, *x ve f(x)* değerlerini ekrana yazdıran Pascal programını yazınız.
(Not: Program yazımı sırasında f(x) fonksiyonunun x<=20 için tanımsız olduğu dikkate alınarak; x için yirmiden küçük değerler girilmesi yazılacak program tarafından engellenmelidir. Klavyeden girilen x değerine karşılık geçerli olan f(x) 'in bulunması CASE ... OF deyimi ile gerçekleştirilecektir.)

```
program cozum03;
uses crt; {Windows için Wincrt}
Var
fx:Real;
i,x:Integer;

begin
  for i:=1 to 10 do
   begin
     repeat
```

```
          Write(i,'. X degeri = ');
          Readln(X);
        Until (X>20);

        CASE X OF
          20..30:fx:=exp(5*ln(x))-exp((7/2)*ln(x));
          31..40:fx:=exp((5/2)*ln(x))-sqr(x);
         else fx:=sqrt(x)+ exp((3/5)*ln(x))-sqr(x);
        end;
        Writeln(X:5,'   ',fx:5:2);
      End;
    end.
```

4. Bir cebirsel denklemin bilgisayar yardımıyla çözümü istendiğinde, örneğin;

$$x^7 + 5x^3 - 11 = 0$$

denklemi;

$$x = (11 - 5x^3)^{1/7} \qquad (1)$$

formuna getirilir ve bu denklemde eşitliğin sağ tarafındaki X'e bir başlangıç değeri verilir (Örneğin; X=0.1). Bu işlemi takiben X'in değeri bulunur. Bulunan X değeri ile başlangıç değeri olarak verilen X arasındaki fark kontrol edilir. Fark, kabul edilen bir değerden (örneğin 0.001) den büyük ise X' değeri tekrar arttırılarak (örneğin 0.001) işlem tekrar edilir. Hesaplanan X değeri ile eşitliğin sağında yerine konulan X değeri karşılaştırıldığında aralarındaki fark belirli bir değerden küçük ise X'in değeri bulunmuş demektir ve işlemlerin tekrar edilmesine son verilir.

X'e başlangıç değeri olarak 0.1 değerini vererek yeni hesaplanan ve denklemde yerine yazılan X değerleri arasındaki fark 0.001 'den küçük oluncaya kadar X 'in değerini hesaplayıp, denklemin kökünü ve X 'in kaçıncı işlem sonunda bulunduğunu ekrana yazdıran Pascal program aşağıda verilmiştir.

X, her tekrar işleminde 0.001 kadar arttırılacaktır.

```
Program cozum04;
uses crt; {Windows için Wincrt}
var
a,x1,x,fark:real;
m:integer;
begin   clrscr;
x:=0.1;
  repeat
      a:=(11-5*exp(3*ln(X)));
      X1:=exp((1/7)*ln(a));
```

```
            writeln (X1:8:5,'    ',fark:18:12,'    ',X:8:5,'  ',m);
            X:=X+0.0001;
            fark:=abs(X1-X);
            m:=m+1;
        until (fark<=0.001);  writeln;
      writeln ('Denklemin Yaklaşık Kökü :',X1:8:5);
      writeln ('Fark                    :',fark:8:5);
      writeln ('Yapılan iterasyon sayısı:',m);
      REPEAT UNTIL KEYPRESSED;
      end.
```

5. Bir açının sinüs değeri aşağıda verilen bağıntı ile hesaplanabilir;

$$SinX = X - \frac{X^3}{3!} + \frac{X^5}{5!} - \frac{X^7}{7!} + ... \text{ (X radyan cinsinden yazılır)}$$

Verilen ifadenin ilk 10 terimi için Sinüs değerini hesaplayan Pascal programı yazınız.

```
Program Sinus;
Uses crt; {Windows için Wincrt}
Var
X,say,fakt,terim,toplam:real;
İsaret,i,k:integer;
Begin
  Write('Bir X değeri giriniz  =');
  Readln(X);
  Toplam:=X;
  Isaret:=-1;
  Say:=x;
  Fakt:=1;
  K:=3;
  For i:=2 to 10 do
    Begin
      say:=say*sqr(X);
      fakt:=fakt*k*(k-1);
      terim:=isaret*say/fakt;
      toplam:=toplam+terim;
      k:=k+2;
      isaret:=-isaret;
    end;
  Writeln('X     =',X:7:2);
  Writeln('SinX  =',Toplam:7:2);
End.
```

6. İki tamsayının en büyük ortak bölenini bulan Pascal programı yazınız.

```
Program ortak  bolen;
Uses crt; {Windows için Wincrt}
Var
i,j,gecici:integer;
begin
  Write('İki Tam sayı giriniz :');
  Readln(i,j);
  Repeat
    İf i>j then
     Begin
        gecici:=i;
        i:=j;
        j:=gecici;
     end;
     if i>0 then j:=j-i;
  until i<=0;
  Writeln('En Büyük Ortak Bölen =',J);
End.
```

7. İki tam sayının en küçük ortak katların bulan Pascal programı yazınız.

```
Program en  kucuk  ortak  kat;
Uses wincrt; {Windows için Wincrt}
Var
i,j,k,okek,a,b:integer;
dur1,dur2:boolean;
Begin
 Write('İki Tam sayı giriniz   =');
 Readln(i,j);
 Dur1:=true;
 Dur2:=true;
 a:=i;
 b:=j;
 k:=2;
 okek:=1;
 While dur1 or dur2 do
   Begin
     While (a mod k =0) or (b mod k =0) do
        Begin
           Okek:=okek*k;
           If a mod k=0 then a:=a div k;
           if b mod k=0 then b:=b div k;
        End;
        if a=1 then dur1:=false;
```

```
            if b=1 then dur2:=false;
          k:=k+1;
        end;
        Writeln('En Küçük Ortak Kat  =',Okek);
      End.
```

BÖLÜM 10

DİZİLER

10.1 Giriş

Bir dizi, aynı tipteki elemanların yanyana sıralanışı ile elde edilen bir bilgi kümesidir.

Örneğin Matematikte ;

$$X = \begin{bmatrix} X1 \\ X2 \\ . \\ . \\ . \\ Xn \end{bmatrix}$$

şeklinde tanımlanan bir X vektörü tek boyutlu bir dizi,

$$A = \begin{bmatrix} A11 & A12 & . & . & A1N \\ A21 & A22 & . & . & A2N \\ . & . & . & . & . \\ . & . & . & . & . \\ AN1 & AN2 & . & . & ANN \end{bmatrix}$$

şeklinde tanımlanan bir A matrisi iki boyutlu bir dizidir.

Bir çok bilim dalında çeşitli niceliklerin matematiksel gösterimi için yaygın olarak kullanılan vektör ve matrislerle ilgili bir program yapacağımızı düşünelim. Her Xi (I=1,2,...n) ve Aij(i=1,2,..n, j=1,2,..n) elemanı için basit tipte bir değişken kullanacak

olursak karşılaşacağımız güçlüğü görürüz. Örneğin 10 bileşenli bir vektör için 10 basit tip değişken kullanılması gerekirken, 10x10 boyutlarındaki matris için 100 basit tip değişken kullanılması gerekmektedir. Bizim için gerekli olan, aynı tipte elemanlardan oluşan bir veri kümesini tek bir değişken ismi kullanarak , küme içindeki yeri ile erişilmesini sağlayacak veri yapısıdır.

Diziler, kullanım amacına göre tek veya çok boyutlu olabilir. Elemanlandırılmış dizi değişkenlerinin boyut durumu ARRAY ifadesiyle köşeli parantez içinde yapılan tanımlamalarla belirtilir.

Dizi karakter tipi "OF tip tanımı" ifadesi ile mutlaka verilmelidir. PASCAL programlama dilinde diziler, TYPE, VAR veya CONST tanım bloklarından birinde tanıtılır. İşleme alınmaları işlemlerin çalışma durumlarına göre bir döngü ile gerçekleştirilir. Tüm elemanlar ilk elemandan başlayarak işleme alınacaksa FOR döngüsü, herhangi bir şarta bağlı olarak tekrar edilecekse WHILE veya REPEAT döngüleri ile kullanılır.

10.2. Dizilerin Tanıtılması

Bir dizinin tanımı değişik şekillerde yapılabilmektedir. Bunlar aşağıda kısaca özetlenmiştir.

10.2.1. Dizilerin Type Bloğunda Tanıtılması

Dizilerin type bloğunda tanıtılması işleminde örneğin;

Tek boyutlu ve 30 elemanlı Endüstri Mühendisliği 1. Sınıf öğrencilerinin numaralarının programa tanıtımı:

```
TYPE Numara=ARRAY [1..30] of string[10];
VAR     ogr:NUMARA;
```

şeklinde yapılabilir.

Örneğe dikkat edilirse NUMARA adı verilen bir dizi TYPE tanım bloğunda tanıtılmış VAR tanım bloğunda ise bu dizinin OGR adı altında değişken tipte olduğu belirtilmiştir.

Aynı şekilde tek boyutlu ve 30 elemanlı Endüstri Mühendisliği 1. Sınıf öğrencilerinin isimlerin programa tanıtımı ise:

```
TYPE Isimler=ARRAY [1..30] of  String[25];
VAR     OgrIsim:Isimler;
```

şeklindedir.

10.2.2 Dizilerin VAR Bloğunda Tanımlanması

Bir dizi diğer değişkenlerde olduğu gibi Var tanım bloğunda da tanıtılabilir. Bunun için dizinin boyutu belirtildikten sonra diziyi oluşturan elemanların hangi tipte olduğu belirtilmelidir. Aşağıdaki ifadeleri inceleyiniz.

```
VAR
X1:ARRAY [1..10] of Real;
k2:ARRAY [1..35] of İnteger;
```

10.2.3 Dizilerin CONST (Sabit Bilgiler) Bloğunda Tanımlanması

Bir dizi diğer sabitlerde olduğu gibi CONST tanım bloğunda da tanıtılabilir. Aşağıdaki örnek programı inceleyiniz.

Örnek:

```
Program yeni  örnek;
Uses crt; {Windows için Wincrt}
Const
Aylar:Array[1..12] of string[7]=
  ('Ocak','Şubat','Mart','Nisan',
    'Mayıs','Haziran','Temmuz',
    'Ağustos','Eylül','Ekim','Kasım',
    'Aralık');
var
k:integer;
begin
clrscr;
  Write ('kaçıncı ay :');
  Readln(k);
  Writeln('Aranan Ay adı :',Aylar[k]);
end.
```

10.3 Tek Boyutlu Diziler

Tek boyutlu diziler aşağıda görülen genel formda ifade edilebilirler.

Değişken = ARRAY [Başlangıç değeri**..**Bitiş Değeri] OF tip tanımı

Buradaki ARRAY ve OF sözcükleri Pascal'ın anahtar sözcükleridir. Başlangıç değeri ve Bitiş Değeri ise sayılabilir özellikte değerlerdir. Tip tanımı ise hangi tip verileri içerdiğini belirtmektedir. Diziler değişik şekillerde tanımlanabilir. Bunlar Bölüm 10.2'de açıklanmıştır.

Örnek : Aşağıda verilen program tek boyutlu bir dizinin elamanlarını sondan başa doğru tersine çevirmektedir.

```
program ters  cevirme;
uses wincrt;
type
   range=1..10;
   dizi=array[range] of integer;
var
   vektor,tersvektor:dizi;
   i:integer;
begin
  For i:=1 to 10 do
   begin
       write('Vektorun ',i,'. elemanını giriniz   =');
       readln(vektor[i]);
   end;
   for i:=1 to 10 do
     tersvektor[10-i+1]:=vektor[i];
   {veya     for i:=10 downto 1 do
              tersvektor[i]:=vektor[10-i+1];   }
   for i:=1 to 10 do
      writeln(tersvektor[i]);
end.
```

Örnek : Aşağıda verilen örnek programda bir kalite kontrol ünitesinde N adet ölçüm sonunda ürünlerin standart sapmaları aşağıda verilen bağıntı ile hesaplanmak istenilmektedir .

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(X_i - X_m)^2}{n}}$$

Burada;

X_i:i. ölçüm değeri
X_m:Ölçümlerin aritmetik ortalaması
n:Toplam ölçüm sayısı;

```
program standart  sapma;
uses crt; {Windows için Wincrt}
type dizi=ARRAY [1..50] of real;
var
x,top,Sigtop,XM,sigma:real;
Xi:Dizi;
i,n:integer;
Begin
    Clrscr;
    Top:=0;sigma:=0;sigtop:=0;
    Write('Yapılan Ölçüm Sayısını Giriniz = ');
    Readln(n);
    For i:=1 to n do
       Begin
           Write(i,'. Ölçüm Değerini giriniz = ');
           Readln(Xi[i]);
           top:=top+xi[i];
       end;
       Xm:=top/n;
       for i:=1 to n do
        begin
         sigtop:=Sigtop+SQR(Xi[i]-Xm);
         Sigma:=SQRT(Sigtop/n);
        end;
        Writeln('Standart Sapma = ',Sigma:10:4);
Repeat until KeyPressed;
End.
```

10.4 Çok Boyutlu Diziler

İndisli değişkenler iki boyutlu olduğunda boyut durumu, köşeli parantez içinde eleman tanımları arasında virgül verilerek belirtilir.

Örneğin;İki boyutlu ve 32 elemanlı B adı verilen indisli değişkenin programa tanıtımı;

```
Type  DIZI:ARRAY[1..4,1..8] of Real;
Var  B:DIZI;
```

şeklinde yapılabilmektedir.

Aynı şekilde;üç boyutlu ve 64 elemanlı B1 adı verilen indisli değişkenin programa tanıtımı;

```
Type  DIZI1:ARRAY[1..4,1..8,1..2] OF Real;
Var  B1:DIZI1;
```

şeklinde yapılabilmektedir.

B isimli iki boyutlu bu dizinin bellekteki yerleşme durumu aşağıdaki gibi olacaktır.

Satırlar \ Sütunlar	1	2	3	4	5	6	7	8
1	B(1,1)	B(1,2)	B(1,3)	B(1,4)	B(1,5)	B(1,6)	B(1,7)	B(1,8)
2	B(2,1)	B(2,2)	B(2,3)	B(2,4)	B(2,5)	B(2,6)	B(2,7)	B(2,8)
3	B(3,1)	B(3,2)	B(3,3)	B(3,4)	B(3,5)	B(3,6)	B(3,7)	B(3,8)
4	B(4,1)	B(4,2)	B(4,3)	B(4,4)	B(4,5)	B(4,6)	B(4,7)	B(4,8)

Örnek :

```
Uses Crt; {Windows için Wincrt}
Type ikilidizi=array[1..9,1..9] of integer;
Var i,j:integer; dizi:ikilidizi;
Begin clrscr;
  for  I:=1 to 9 do
    begin
       for j:=1 to 9 do
            begin
               dizi[i,j]:=i*j;
               Write(Dizi[i,j]:4);
            End;
            Writeln;
    End;
  end.
```

10.5 Örnek Programlar

1. N elemandan oluşan tek boyutlu bir X dizisinin en büyük elemanı ile en küçük elemanı arasındaki fark ile X dizisinin elemanları çarpılarak yeni bir Y dizisi elde edilmek isteniyor. Yukarıda tanımlanan Y dizisinin elde edilmesi için gerekli olan Pascal programını yazınız.
Not: X dizisinin elemanları klavyeden girilecektir.

```
Program N  elemanli  X  dizisi;
   uses crt; {Windows için Wincrt}
   var
        x,y:array[1..50] of real;
        enkucuk,enbuyuk,fark:real;
        sayi:integer;
        i:byte;
        begin
        clrscr;
        Write('Dizinin Kaç Elemanı Var =');
        Readln(Sayi);
        for i:=1 to sayi do
            begin
              Write(i,'. sayıyı giriniz =');
              Readln(x[i]);
            end;
          enbuyuk:=x[1];enkucuk:=x[1];
        for i:=1 to sayi do
           begin
             if enbuyuk<x[i] then enbuyuk:=x[i];
             if enkucuk>x[i] then enkucuk:=x[i];
           end;
          fark:=enbuyuk-enkucuk;
        Writeln;
        Writeln('En Büyük Sayı=',enbuyuk:6:1,'   ','En Küçük
        Sayı=',enkucuk:6:1,' ','Fark=',Fark:6:1);
        Writeln;
        for i:=1 to sayi do begin
            y[i]:=fark*x[i];
            Writeln('y[',i,'] = ',    y[i]:6:1);
            end;
        repeat until keypressed;
End.
```

2. Aşağıda 3x6 boyutlarındaki A matrisinin transpozesini hesaplayan pascal programı. verilmiştir.

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \end{bmatrix} \qquad A^T = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \\ 4 & 5 & 6 \\ 5 & 6 & 7 \end{bmatrix}$$

```
Program transpoze;
Uses crt; {Windows için Wincrt}
Type
Dizi=array[1..3,1..5] of real;
dizi1=array[1..5,1..3] of real;
var
matA:dizi;
TransA:dizi1;
i,j:byte;
begin
    for i:=1 to 3 do
     begin
       Writeln(i,'. Satir elemanlari');
       for j:=1 to 5 do
           begin
              readln(matA[i,j]);
           end;
           writeln;
     end;
for i:=1 to 5 do
    begin
     for j:=1 to 3 do
       begin
        transA[i,j]:=matA[j,i];
        write(transA[i,j]:6:2);
       end;
     writeln;
    end;
end.
```

3. Aşağıdaki program ile iki vektörün çarpımı yapılmaktadır.

```
program vektor  carpimi;
uses crt; {Windows için Wincrt}
type
    range=1..5;
    dizi=array[range] of integer;
var
    vektor1,vektor2,vektor3:dizi;
    i,j:integer;
begin
  For i:=1 to 5 do
    begin
        write('1. Vektorun ',i,'. elemanını giriniz   =');
        readln(vektor1[i]);
    end;
  For j:=1 to 5 do
```

```
        begin
            write('2. Vektorun ',j,'. elemanını giriniz   =');
            readln(vektor2[j]);
        end;

    for i:=1 to 5 do
      begin
          vektor3[i]:=vektor1[i]*vektor2[i];
          writeln(vektor3[i]);
      end;
    end.
```

4. 100 elemanlı Y(M) ve AG(M) dizileri rastgele sayılardan oluşmuştur. Bu iki dizi okutulduktan sonra.,

$$x = \frac{\sum_{M=1}^{L} Y(M) * AG(M)}{\sum_{M=1}^{L} Y(M)}$$ Ağırlıklı aritmetik ortalama

$$RMS = \sqrt{\frac{\sum_{M=1}^{L} AG(M)^2}{L}}$$ Karesel ortalama

$$HM = \frac{1}{\frac{1}{L}\sum_{M=1}^{L} \frac{1}{AG(M)}}$$ Harmonik ortalama

$$SD = \sqrt{\frac{\sum_{M=1}^{L} AG(M)^2}{L} - \left(\sum_{M=1}^{L} AG(M)\right)}$$

değerlerini bularak ekrana yazdıran bir PASCAL programını yapınız.

```
Program ortalamalar ve standart sapma;
Uses crt; {Windows için Wincrt}
Var
  y,ag:array[1..10] of integer;
  bboluag,x,rms,hm,sd:real;
  bkisim,pay,pay1,payda,m:integer;
const
  L=5;
Begin
```

```
for m:=1 to L do
  begin
    Write('Y[',M,'] degerini giriniz  :');Read(Y[m]);
    Write('AG[',M,'] degerini giriniz :');Readln(AG[m]);
  end;
pay:=0;payda:=0;
 for m:=1 to L do
  begin
    pay:=pay+y[m]*ag[m];
    payda:=payda+y[m];
  end;
x:=pay/payda;
pay1:=0;
 for m:=1 to L do
   pay1:=pay1+sqr(ag[m]);
   rms:=sqrt(pay1/L);
 bboluag:=0;
 for m:=1 to L do
   bboluag:=bboluag+(1/ag[m]);
   hm:=1/((1/L)*bboluag);
 for m:=1 to L do
  bkisim:=bkisim+ag[m];
  sd:=sqrt(pay1/L-bkisim);
Writeln('Agirlikli aritmetik ortalama        =',x:8:2);
Writeln('Karesel  ortalama                   =',rms:8:2);
Writeln('Harmonik ortalama                   =',hm:8:2);
Writeln('Standart Sapma                      =',sd:8:2);
end.
```

5. SON = $\dfrac{(\sum_{i=1}^{n} X^2 - \sum_{i=1}^{n} X) * \sum_{i=1}^{n} \frac{1}{X}}{\sum_{i=1}^{n} X^2}$ ifadesi ile verilen **son** değerini hesaplayarak sonucu yazdıran Pascal programını aşağıda verilmiştir. (N değeri klavyeden girildikten sonra N adet X değeri de klavyeden girilecektir).

```
Program Son  deger;
uses crt; {Windows için Wincrt}
var
x:array [1..100] of real;
n,sayac:shortint;
son,topX,topXkare,top1boluX,TopXkareX:real;
begin
clrscr;
topX:=0;topXkare:=0;top1boluX:=0;TopXkareX:=0;sayac:=0;
Write('N değerini giriniz :');Readln(N);
```

```
For sayac:=1 to n do
 Begin
   Write(sayac,'. X değerini giriniz =');Readln(X[sayac]);
   TopX:=TopX+x[sayac];
   topXkare:=topXkare+sqr(x[sayac]);
   Top1boluX:=Top1boluX+1/x[sayac];
 end;
son :=(TopXkare-TopX)*Top1boluX/TopXkare;
writeln('İşlemin sonucu  =',son:8:2);
repeat until keypressed;
end.
```

6. Endüstri mühendisliği 1.sınıf öğrencilerinin Matematik-2 dersi arasınavı ve final sınavı notları klavyeden girilerek sınav sonuç çizelgesi oluşturulacaktır. Öğrencinin vize ve final notları sırasıyla klavyeden girilecektir. Klavyeden girilen notlar 0-100 aralığında olması sağlanmalıdır, bu koşul sağlanmıyor ise ilgili not klavyeden tekrar girilmelidir. Notların istenilen aralıkta olmasının sağlanması program tarafından otomatik olarak yapılmalıdır. Geçme notu olarak BAÜ'de uygulanan not sistemi kullanılacaktır. Programın çıktısı şu şekilde olacaktır;

Öğr. No	Vize	Final	Ortalama	DURUMU
9420001	87	87	87	BASARILI
9420002	40	57	50	BASARILI

```
program vize  final  ortalamasi;
uses crt; {Windows için Wincrt}
Type
    Notlar=array[1..50,1..50] of byte;
    dersort=array[1..50] of real;
    ders=array[1..50] of integer;
var
osay,dsay,tsay,i,j:byte;
ogrort,ogtop:dersort;
ort:ders;
a:Notlar;
cl:string[5];
kl:string;
begin clrscr;
Write('Dersi Alan Öğrenci sayısını giriniz...=');Readln(osay);
for i:=1 to osay do
 begin
  for j:=1 to 2 do
    begin
      if i<10 then kl:='9420000' else kl:='942000';
```

```
            repeat
              if j=1 then cl:='Vize ' else cl:='Final';
              write(kl,i,' numaralı Öğrencinin ',cl,' Notunu giriniz  = ');
              Readln(a[i,j]);
            until (a[i,j]>0) and (a[i,j]<100)
          end;
       end;
        for i:=1 to osay do
          begin
            ogrort[i]:=round(a[i,1]*0.40+a[i,2]*0.60);
            ort[i]:=trunc(ogrort[i]/1);
          end;
      writeln;
      writeln('Öğr. No    Vize  Final  Ortalama  DURUMU');
      writeln('-----------+-----+------+---------+--------');
       for i:=1 to osay do
         begin
          if i<10 then kl:='9420000' else kl:='942000';
          if (ort[i]>=50) and (a[i,2]>=50.0) then
           begin
            writeln(kl,i,'  ',A[i,1]:3,'   ',A[i,2]:3,'    ',ogrort[i]:6:2,'     BAŞARILI');
           end
           else
           writeln(kl,i,'  ',A[i,1]:3,'   ',A[i,2]:3,'    ',ogrort[i]:6:2,'     BAŞARISIZ');
           end;
      Repeat until keypressed;
      End.
```

7. Aşağıdaki program iki boyutlu iki matrisin toplamlarını hesaplamaktadır.

```
program matris  toplami;
uses crt; {Windows için Wincrt}
type
    boyut=1..20;
    matris=array[boyut,boyut] of integer;
var
   matA,matB,matC:matris;
   i,j,k:integer;
   N:boyut;
Begin
     Write('Matris boyutu (1..20) :');
     Readln(N);clrscr;
     Writeln('1. Matris elemanları');
     for i:=1 to N do
        begin
          for j:=1 to N do
            begin
```

```
                    gotoxy(j*6,i+3);Read(matA[i,j]);
                  end;
                writeln;
              end;
            gotoxy(40,1);Writeln('2. Matris elemanları');
            for i:=1 to N do
              begin
                for j:=1 to N do
                  begin
                    gotoxy(35+j*6,i+3);Read(matB[i,j]);
                  end;
                writeln;
              end;
            {MatA ve matB isimli, matrislerin toplamı}
             Writeln;
             Writeln('Matrislerin Toplamı');
             Writeln;
             for i:=1 to N do
               begin
                 for j:=1 to N do
                   begin
                     matC[i,j]:=matA[i,j]+matB[i,j];
                     Write(matC[i,j]:6,'  ');
                   end;
                 writeln;
               end;
          end.
```

10. Aşağıda iki matrisin çarpımını yapan pascal programı verilmiştir.

```
program matris  carpimi;
uses crt; {Windows için Wincrt}
type
    boyut=1..20;
    matris=array[boyut,boyut] of integer;
var
   matA,matB,matC:matris;
   i,j,k,top:integer;
   N,M,M1,K1:boyut;
Begin
     repeat
       Write('A matrisinin Satır Sayısı          =');Readln(N);
       Write('A matrisinin Sütun Sayısı          =');Readln(M);
       Write('B matrisinin Satır Sayısı          =');Readln(M1);
       Write('B matrisinin Sütun Sayısı          =');Readln(K1);
       if M<>M1 then Writeln ('Hatalı veri, Bu boyutlarda matris çarpılamaz');
```

```
            until M=M1;clrscr;
            Writeln('1. Matris elemanları');
            for i:=1 to N do
              begin
                for j:=1 to M do
                  begin
                    gotoxy(j*6,i+3);Read(matA[i,j]);
                  end;
                writeln;
              end;
            gotoxy(40,1);Writeln('2. Matris elemanları');
            for i:=1 to M do
              begin
                for j:=1 to K1 do
                  begin
                    gotoxy(35+j*6,i+3);Read(matB[i,j]);
                  end;
                writeln;
              end;

            {MatA ve matB isimli, matrislerin çarpımı}
             for i:=1 to N do
              begin
                for j:=1 to K1 do
                  begin
                    for k:=1 to M do
                      begin
                        top:=top+matA[i,k]+matB[k,j];
                        matC[i,j]:=top;
                      end;
                  end;
                writeln;
              end;

             {MatC matrisinin yazdırılması}
             Writeln;
             Writeln('Matrislerin Çarpımı');
             Writeln;
             for i:=1 to N do
              begin
                for j:=1 to K1 do
                  begin
                    write(matC[i,j]:6,'  ');
                  end;
                writeln;
              end;
        end.
```

9. Verilen N adet {x,y} çifti arasında lineer bir ilişki olduğu kabul edilmektedir. Bu sayı .çiftleri için en uygun doğru Y=AX+ B olup burada;

$$a = \frac{1}{d}(n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i)$$

$$b = \frac{1}{d}(\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i y_i)$$

$$d = n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2$$

bağıntıları ile verilmektedir. X ye Y değerlerini ve N sayısını klavyeden girerek **a** ve **b** katsayılarını hesaplayıp ekrana yazdıran pascal programı yazınız.

```
Program katsayi;
uses crt; {Windows için Wincrt}
type dizi=array[1..200] of real;
var n,i:byte;
    x,y:dizi;
    a,b,d,xiyitop,xitop,yitop,xkaretop:real;
 begin
  Write('Nokta sayìsìnì giriniz  :');Readln(N);
   For i:=1 to N do
     Begin
       Write('X[',i,'] =');Readln(X[i]);
       Write('Y[',i,'] =');Readln(Y[i]);
     End;
    for i:=1 to N do
    begin
      xkaretop:=xkaretop + sqr(x[i]);
      xitop := xitop + x[i];
    end;
    d := N * xkaretop - sqr(xitop);   xiyitop:=0;yitop:=0;
  for i:=1 to N do
     begin
       xiyitop:=xiyitop+X[i]*Y[i];
       yitop:=yitop+y[i];
     end;
  a:=(n* xiyitop – Xitop*Yitop)/d;
  b:=(Xkaretop * Yitop - Xitop * Xiyitop)/d;
  Writeln('A  =',a:6:3);   Writeln('B  =',b:6:3);
 End.
```

10. N sayıdaki gözlemle ilgili olan X(i), Y(i) sayı dizileri arasında doğrusal bir ilişki olup olmadığı korelasyon katsayısı hesaplanarak belirlenmektedir.

Korelasyon katsayısı aşağıda verilen ifade ile hesaplanmaktadır. KORKAT =1 ise X(i) ve Y(i) sayı dizileri arasında tam bir doğrusal ilişki vardır. KORKAT≥ 0.75' olması halinde X(i) ve Y(i) sayı dizileri arasında doğrusal ilişki kurulabilir. Aksi halde bu sayı dizileri arasında doğrusal ilişki kurulamaz.

N sayısını ve bu dizileri klavyeden girerek, bu gözlemler arasındaki korelasyon katsayısını hesaplayarak, x(i) ve y(i) sayı dizileri arasında nasıl bir ilişki olabileceğini ve korelasyon katsayısının değerini ekrana yazdıran pascal programını yazınız.

Bu programı, dokuzuncu program ile birleştirerek bir lineer regresyon programı oluşturunuz.

$$KORKAT = \frac{\sum_{i=1}^{N} xy - \frac{\sum_{i=1}^{N} x \sum_{i=1}^{N} y}{N}}{\left\{ \left[\sum_{i=1}^{N} x^2 - \frac{(\sum_{i=1}^{N} x)^2}{N} \right] \left[\sum_{i=1}^{N} y^2 - \frac{(\sum_{i=1}^{N} y)^2}{N} \right] \right\}^{1/2}}$$

BÖLÜM 11

ALT PROGRAMLAR

11.1 Giriş

Yapısal programlamanın en önemli yaklaşımlarından biri, temel uygulamayı bir takım parçalara bölerek gerçekleştirmektir. Pascal'da bu parçalar altyordam (procedure) veya fonksiyon (function) olarak kodlanır. Yapısal programlamanın temel tercih nedeni, program modüllerinin bağımsız olarak geliştirilmeleri ve çalışıp çalışmadığının kolaylıkla anlaşılabilmesidir.

Alt programlar genellikle tekrar edilen işlemleri kolaylıkla yapabilmek için yazılırlar. Tekrar edilen işleme örnek olarak, programımız içinde birden fazla yerde faktöriyel hesabı yapmamız gerekiyor ise bunu bir alt program ile hesaplatmamız halinde aynı işleme sahip program satırlarının tekrarı önlenecek, programın takibi kolaylaşacak ve daha verimli hafıza kullanımı söz konusu olacaktır.

Bilgisayar dillerinin tamamında bulunan alt program yapısı PASCAL Programlama dilinde PROCEDURE ve FUNCTION türü alt programlar olmak üzere iki ayrı türdedir. Bu iki ayrı tür alt programlar birbirlerine benzer özellikler taşımalarına rağmen yapısallık ve icra açısından bazı farklılıklar içerirler.

Alt programlarda yapılacak işlemler, ana program içinde nerede gerekiyor ise o noktada ilgili işlemleri yapan alt programın adı yazılarak devreye sokulur.

11.1.1 Alt Programlar Hakkında Genel Bilgiler

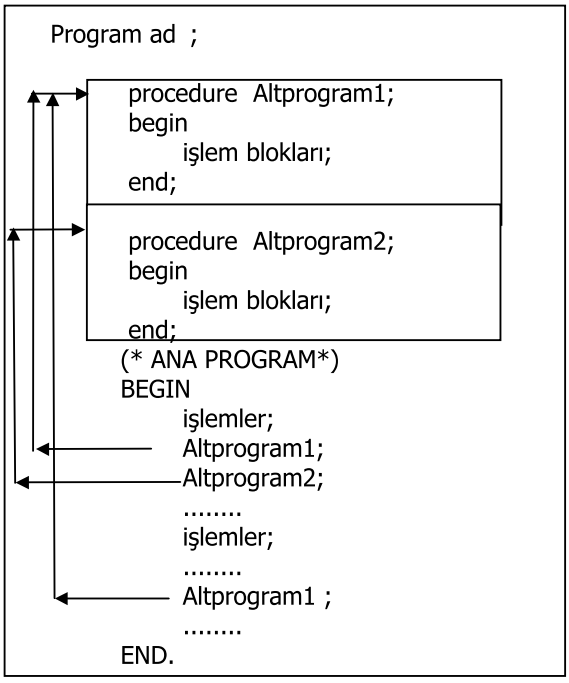
*Her alt programın bir başlığı, bir tanım bloğu, bir de icra bloğu vardır.
*Ana programlar PROGRAM deyimi, prosedür alt programlar PROCEDURE, fonksiyon alt programlar FUNCTION deyimi ile başlar.
*Alt program adının hemen yanında verilen parametreler değer transferi işlemleri için kullanılırlar. Bir alt programa değer transferi yapılarak, her bir çalışmada verilen değişik değerlere bağlı olarak değişik sonuçlar üretmek mümkündür.

*Alt program isminin yanına parametre yazmak seçime bağlıdır. Eğer parametre yazılıyor ise bunlar procedure isminin yanına parantez içinde yazılırlar.

*Pascal programlama dilinde **formal** ve **aktüel** parametreler olmak üzere iki tür parametre tanımlanabilir. **Formal** parametreler, programın tamamına ait olmayıp sadece değer transferi için kullanıldıklarından geçici paramatreler olarak isimlendirilebilirler. Yani bu parametreler ile sadece alt program içinde çalışılabilir. Ana programda, alt programın çalıştırılması için yazılması gereken alt programın yanında yazılan parametreler ise **Aktüel** parametre adını alırlar. Burada aktüel ve formal parametrelerin sıralanış, sayı ve tip bakımından aynı olması gerekir.

11.2 Procedure Alt Programlar

Bir pascal programında Procedure alt programlara erişim ve procedure alt programın program içindeki yeri aşağıdaki genel gösterim şeklinde verilmiştir.



Alt programlarda, ana programlarda olduğu gibi sabit, değişken ve çeşitli tip tanımlamaları yapılabilir. Yapılan bu tanımlamalar, alt program aktif olduğu sürece geçerli olup alt programın çalışması sona erdiğinde alt programda tanımlanan değişken, sabit ve tipler bellekten silinir.

Procedure işlem blokları iki şekilde kullanılır,

1. Parametresiz Prosedürler
2. Parametreli Prosedürler

11.2.1 Parametresiz Prosedürler

Bu tür alt programlarda; alt program içinde kullanılan değişkenler ana program içinde kullanılan değişkenler ile aynıdır. Bu nedenle ana program ile alt program arasında veri transferi yapılmasına gerek yoktur. Aşağıdaki örnek programlar Parametresiz Prosedür kullanımını açıklamaktadır.

Örnek:

```
program faktoriyel  hesabi;
uses crt; {Windows için Wincrt}
var fakt:longint;
    n,i:integer;
procedure islem1;
 begin
    fakt:=1;
    for i:=1 to n do
        begin
         fakt:=fakt*i;
        end;
 end;
begin
   clrscr;
   Write('N sayısını giriniz =');
   Readln(N);
   islem1;
   Writeln('N sayısının Faktöriyeli  =',fakt:10);
   repeat until keypressed;
end.
```

Yukarıda verilen programı çalıştırarak sonucu inceleyiniz.

11.2.1.1 Forward İfadesinin Kullanımı

Program içinde tanımlanan Prosedür blokları içinde bazılarının programın daha sonraki satırlarında tanımlanacağını belirten ifadedir. FORWARD deyimi, sonra tanımlanacak prosedür bloğu devamında kodlanmalıdır.

Aşağıdaki örnek programda ASAMA2 prosedür bloğu devamında görülen FORWARD ifadesi blok bilgilerinin daha sonra tanımlanacağını belirtir. Program I>10 olduğu durumda sona erecektir.

Örnek:

```
program procedure  ornekleri;
uses crt; {Windows için Wincrt}
var i:integer;
Procedure Asama2;forward;
   Procedure Asama1;
      Begin
         i:=i+1;
         Write(I,'.');
         Asama2;
      End;
   Procedure Asama2;
      Begin
    Write(I*i,'.');
         if i>10 then halt;
         readln;
         Asama1;
      End;
begin
  clrscr;
  i:=1;
  asama1;
end.
```

Yukarıda verilen iki örnek programda procedure ismini takiben herhangi bir değişken adı yazılmadığına dikkat ediniz.

11.2.2. Parametreli Prosedürler

Ana program ve alt programlarda tanımlanan değişkenlerin farklı olması durumunda tercih edilirler. Ana programda tanımlanan değişken değerlerinin işleme alınmaları, alt programda tanımlanan değişkenlere aktarılarak yapılır.

Parametreli prosedürlerde de tek yönlü ve çift yönlü olmak üzere iki ayrı tür değer transferi söz konusudur. Eğer procedure ismini takiben parametrelerin tanım cümlesi VAR ile başlıyorsa parametreler arasında çift yönlü değer transferi yapılacak demektir. Yani ana programda kullanılan değişkenler alt programa aktarılacak ve alt programın çalışması sona erdiğinde sonuç değerler ana programa transfer edilir.

Tek yönlü değer transferinde ise parametrelerin tanım cümlesi VAR ile başlamaz, değer transferi ana programdan alt programa olur ve alt programın icrası sonunda ortaya çıkan sonuçlar ana programa transfer edilmez. Her iki tür için yazım örnekleri aşağıda verilmiştir.

Tek yönlü Transfer : Procedure Altyordam(X;Y;Z:integer;KAD:real);
Çift yönlü Transfer : Procedure Altyordam(VAR X;Y;Z:integer;VAR KAD:real);

Veri transferinde String değişken veya bir dizi değişken formal parametre olarak tanımlanamaz. String ve dizi değişkenler formal parametre olarak kullanılacaksa, tip bildirisinin özel bir tip ismi olarak verilmesi gerekir. Aşağıda verilen örnekler sözü edilen kurala göre hatalı yazılmışlardır.

```
Procedure Altyordam(A:Array[1..100] of real);
Procedure Altyordam2(S:String[200]);
```

Uygun tanımlamalar yapmak için Type bölümünden yararlanılmalıdır. Aşağıda verilen örnek tanımlar özel tip isimler kullanıldığı için geçerlidir.

```
Type
        Oz=Array[1..100] of real;
        sozcuk=String[200];

Procedure Altyordam(A:Oz);
Procedure Altyordam2(S:Sozcuk);
```

Ana programdan alt programa transfer sırasında tanım bloklarında kullanılan parametrelerin tipleri aynı olmak zorundadır. Aksi takdirde 'Tip Uyuşmazlığı' hatası meydana gelecektir.

Aşağıdaki örnek programları inceleyiniz.

Örnek 1:

```
program paramli1;
uses crt; {Windows için Wincrt}
var
  a,b,t:integer;
procedure toplam(x,y:integer);
  begin
    t:=x+y;
  end;
begin
  clrscr;
  readln(a);
  readln(b);
  Toplam (a,b);
  Writeln(t);
end.
```

Örnek 2:

```
program paramli2;
uses crt; {Windows için Wincrt}
var
  a,b,t:integer;
```

```
procedure toplam(x,y:integer;var z:integer);
  begin
    z:=x+y;
  end;
begin  clrscr;
  readln(a);
  readln(b);
  Toplam (a,b,t);
  Writeln(t);
end.
```

Yukarıdaki programda X ve Y değerlerinin toplamları *Writeln(t);* program satırında ekrana yazdırılmaktadır. Aynı programda toplamı yazdırmak istenilen parametre değiştirildiğinde program hata verecektir. Aşağıdaki program derlendiği zaman çalışmayacak ve 3 nolu hata meydana gelecektedir. Programı yazarak hatanın nedenini anlamaya çalışınız.

Örnek 3:

```
program hatali  paramli;
uses crt; {Windows için Wincrt}
var
  a,b,t:integer;
procedure toplam(x,y:integer;var z:integer);
  begin
    z:=x+y;
  end;
begin  clrscr;
  readln(a);
  readln(b);
  Toplam (a,b,t);
  Writeln(z);
end.
```

Örnek 4: Aşağıdaki örnek programda dizgi isimli tek boyutlu bir dizi oluşturulmuş ve bu dizinin elemanları sıra ile 1-100 arasındadır. Bu dizi değer aldıkça altprograma değer transfer etmekte ve alt programda bu dizinin değerleri *random komutu* ile rastgele sayılardan oluşan yeni bir diziye aktarılmakta ve ana programa geri dönmektedir.

```
Program degerparametresi;
uses crt; {Windows için Wincrt}
type DiziTipi=Array[1..100] of integer;

var
Dizgi:DiziTipi;
Yenidizi1,Yenidizi:DiziTipi;
K:word;
```

```
PROCEDURE Ornekleme(Pardizi:DiziTipi;var YeniDizi:DiziTipi);
  begin
    Yenidizi[k]:=Random(Pardizi[k])
  end;
begin
    for k:=1 to 100 do
    begin
      Dizgi[k]:=K;
      Ornekleme(Dizgi,YeniDizi1);
    Writeln(Dizgi[k],'    ',YeniDizi1[k]);
    end;While not keypressed do
end.
```

11.3 Kullanıcı Tanımlı Function Alt Programları

Pascal'da altprogramlar başlığı altındaki yapısal modüllerden bir diğeri de fonksiyonlardır (function). Function ve Procedure arasındaki farkı anlayabilmek için procedure ve function olarak ayrı ayrı yazılmış bir uygulama üzerinde duralım.

```
PROCEDURE Toplam(X,Y:integer; var Z:integer);
  begin
     Z:=X+Y;
  end;


FUNCTION Toplam (X,Y:integer):Integer;
  begin
     TOPLAM:=X+Y;
  end;
```

Örnekte toplam isimli alt yordam (Procedure) X ve Y tamsayılarının toplamını referans parametresi türündeki Z değişkeni aracılığı ile getirir. Procedure durumunda geri döndürülecek değer, VAR ile tanımlanmış bir değişkene atanmaktadır. Fonksiyon olarak tanımlanan TOPLAM modülü ise toplamı kendisi getirmektedir. Fonksiyon sona erdirilmeden önce geri döndürülecek, fonksiyonun ismine atanmaktadır. Toplam isimli fonksiyon, integer tiptedir.

Procedure olarak yazılan TOPLAM alt programı ;

Toplam (A;B;T); (Procedure)

şeklinde çağırılırken function şeklinde geliştirilmiş Toplam alt programı ;

T:=Toplam(A;B); (Function)

komutuyla çağırılır.

11.4 Örnek Programlar

Örnek 1. $Y=x^2+5x*Sin(x)$ fonksiyonunun, X'in 0<= X<17 aralığındaki değerlerini veren bir Pascal programını oluşturunuz.

```
program fonksiyonlar;
uses crt; {Windows için Wincrt}
var x,y:real;
Function Fonk(x:Real):real;
begin
    Fonk:=sqr(x)+5*x*Sin(x*Pi/180);
end;
Begin
    Writeln('Y =x^2+5*x*Sin(x)   Denkleminin aldığı değerler');
    x:=0;
      Repeat
        y:=fonk(x);
        Writeln(x:2:1,'  Değeri İçin  =',y:5:2);
        x:=x+0.5
      until x>=17
end.
```

Yukarıda verilen örnek program için y fonksiyonunun değerini f(i) şeklinde bir diziye aktararak dizinin en küçük ve en büyük elemanları arasındaki fark ile bu diziyi çarptırıp g(i) şeklinde yeni bir dizi elde ediniz. f(i) dizisinin en büyük ve en küçük elemanları program tarafından bulunacaktır.

Örnek2. Aşağıda verilen örnek program faktöriyel hesabının hem procedure hem de function türü altprogramlar ile ayrı ayrı hesaplanmasını göstermektedir. Programı inceleyerek procedure alt programlar ile function alt programlar arasındaki farkı inceleyiniz.

```
program faktoriyel1;
uses crt; {Windows için Wincrt}
var x:integer;
   sonuc,sonuc1,fakt:longint;
   n,i:integer;
Procedure islem1;
begin
   fakt:=1;
   for i:=1 to x do
      begin
       fakt:=fakt*i;
      end;
```

```
end;
Function faktoriyel(xx:integer):longint;
begin
    faktoriyel:=1;
    if xx= 0 then exit;
    Faktoriyel:=xx*faktoriyel(xx-1);
end;
begin
    clrscr;
    Write('X sayısını giriniz =');
    readln(X);
    islem1;
    sonuc:=Faktoriyel(x)-fakt;
    Writeln('X  sayısının Faktöriyeli  =',Faktoriyel(x):10,'    ',sonuc);
while not keypressed do
end.
```

Örnek 3. Aşağıdaki örnek program klavyeden girilen N adet sayıyı büyükten küçüğe sıralamaktadır. Programı çalıştırınız ve çalışma prensbini anlamaya çalışınız.

```
Program Buyukten  Kucuge  SiralaMa;
uses crt; {Windows için Wincrt}
type
Sira=array[1..1000] of real;
var
x:sira;
n,i:integer;

Procedure Degistir(var V1,V2:real);
 var
  vd:Real;
   begin
    Vd:=V1;
    V1:=V2;
    V2:=Vd;
   end;

Procedure Sirala(Var xx:Sira);
 Var
  i,j:Integer;
 begin
  for I:=1 to n do
   for J:=1 to n do
    if xx[J] < xx[I] then Degistir(xx[i],xx[j]);
end;

begin
```

```
 clrscr;
Write(‘Kaç Sayı Girişi Yapacaksınız  =‘);
ReadLn(N);
for i:= 1to N do
  begin
 Write(i,’. Değeri giriniz  =‘);
   ReadLn(X[i]);
   end;
   Sirala(X);
   WriteLn;Writeln(‘Büyükten küçüğe sıralanmış sayılar’)
   for i:=1 to n do
   WriteLn(X[i]:4:1);
end.
```

Verilen örnek programı, girilen sayıları küçükten büyüğe sıralama yapacak şekilde değiştiriniz.

Örnek 4. Büyük/Küçük harfle yazılmış olan dosyayı küçük/büyük harfe çevirerek başka bir dosyaya yazmaktadır.

```
uses crt; {Windows için Wincrt}
var Dosya:TEXT;
   Dosya1:TEXT;
   X:char;Y:byte;
   Giris:String[64];
   Cikis:String[64];

Procedure KucukHarfleYaz;
Begin Write('Küçük harfe cevrilecek : ');readln(giris);
     Write('        Hedef Program : ');readln(cikis);
     Assign(dosya,giris);assign(dosya1,cikis);Reset(dosya);ReWrite(Dosya1);
     while not eof(dosya) do
          begin while not eoln(dosya) do
               begin read(dosya,x);y:=ord(x);
                    if (y>=65)and(y<=90) then y:=y+32;
                    x:=chr(y);write(dosya1,x);
               END;
          readln(dosya);writeln(dosya1);
          END;
          close(dosya);close(dosya1);
end;
Procedure BuyukHarfleYaz;
Begin Write('Büyük harfe cevrilecek : ');readln(giris);
     Write('        Hedef Program : ');readln(cikis);
     Assign(dosya,giris);assign(dosya1,cikis);Reset(dosya);ReWrite(Dosya1);
     while not eof(dosya) do
          begin while not eoln(dosya) do
               begin read(dosya,x);y:=ord(x);
```

```
                    if (y>=97)and(y<=122) then y:=y-32;
                    x:=chr(y);write(dosya1,x);
                END;
            readln(dosya);writeln(dosya1);
            END;
            close(dosya);close(dosya1);
    end;

    Begin
    KucukHarfleYaz;
    BuyukHarfleYaz;
    end.
```

Program çalıştığında büyük harfi küçük harfe dönüştüren alt program çalışır. Bu alt program çalıştığında aşağıdaki bilgiler programcıdan istenir.

Küçük harfe cevrilecek : {dosya adı girilir}
Hedef Program :{küçük harfe dönüşüm sonucunun yazılacağı dosya adı}

Dosya aşağıdaki bilgilerden oluşmuştur.

BALIKESİR ÜNİVERSİTESİ
MÜHENDİSLİK MİMARLIK FAKÜLTESİ

Programın çalışması sonrasında hedef dosya aşağıdaki forma dönüşmüştür.

balikesİr Ünİversİtesİ
mÜhendİslİk mİmarlik fakÜltesİ

Küçük harfle yazılan bir dosyayı büyük harfe çeviren alt program icra edilmeye başladığında ;

Büyük harfe cevrilecek : {dosya adı girilir}
Hedef Program :{büyük harfe dönüşüm sonunda bilgilerin yazılacağı dosya adı}

Dosya aşağıdaki bilgilerden oluşmuştur.

balikesİr Ünİversİtesİ
mÜhendİslİk mİmarlik fakÜltesİ

Programın çalışması sonrasında hedef dosya aşağıdaki forma dönüşmüştür.

BALIKESİR ÜNİVERSİTESİ
MÜHENDİSLİK MİMARLIK FAKÜLTESİ

Örnek 5. Yukarıdaki örnekte Türkçe karakterlerin büyük küçük harfe veya büyük dönüştürülmesi yapılamıyor. Klavyeden girilen Türkçe karakterleri küçükten büyüğe çeviren Pascal programı aşağıda verilmiştir.

```
program kucuk harfleri buyuk yapma;
uses wincrt;
var
s:string;
i:byte;
function buyukharf(S:string):string;
begin
    for i:=1 to length(s) do
     case s[i] of
        'a'..'h':S[i]:=upcase(S[i]);
        'j'..'z':S[i]:=upcase(S[i]);
        'ç':S[i]:='Ç';
        'ğ':S[i]:='Ğ';
        'ı':S[i]:='I';
        'i':S[i]:='İ';
        'ö':S[i]:='Ö';
        'ş':S[i]:='Ş';
        'ü':S[i]:='Ü';
     end;
     buyukharf:=s;
end;
begin
    WRiTE('küçük harfle sözcük yazınız    =');
    readln(s);
    Writeln(Buyukharf (S));
    readln;
end.
```

BÖLÜM 12

DOSYALAR

12.1 Giriş

Bir program yaparken, bilgilerin her zaman ekrana yazdırılması sorunumuzu çözemeyebilir. Örneğin; bir işyerinde stok kontrol bilgilerinin bilgisayar ile denetimi söz konusu ise ilgili işyerindeki stok kayıtlarının manyetik veya optik ortamda yüklü olması gerekmektedir. Belli bir konuya ait bilgiler üzerinde değişiklikler yapmak, raporlar almak ve çeşitli kontroller yapabilmek için manyetik veya optik ortamda saklanırlar. Bilgilerin manyetik veya optik ortama aktarılması değişik yöntemlerle yapılır. Pascal programlama dilinde Text ve binary olmak üzere iki çeşit dosyalama yöntemi kullanılır.

a-Text Türü Dosyalar

Bilgilerin manyetik veya optik ortama yazılması ve okutulması sıralı bir metin halindedir. Bilgiler manyetik veya optik ortama girildikleri gibi yazdırılırlar. Bu dosyalar herhangi bir metin editörü ile okunabilirler.

b-Binary Türü Dosyalar

Kayıt tipi belirli ve kayıt tipi belirsiz olarak iki gruba ayrılırlar. RANDOM dosyalar ve anahtar terimlere göre erişim yapılan (INDEXED) dosyalamalar binary türündendir.

12.2 Text Dosyalar

Bu dosyalar genellikle Turbo Pascal editörü ile yazılarak saklama ortamına aktarılan veya Turbo Pascal programının çalışması ile saklanan ortamda verilen bir isim ile ulaşılan ASCII formda dosyalardır. Turbo Pascal derleyicisi Text dosyalar için ana bellekte 128 byte'lik bir tampon bellek açar. Bilgiler önce tampon belleğe ve dış belleğe aktarılır. Manyetik ortamdan bilgi okunması sırasında önce bilgiler tampon belleğe aktarılır sonra okunur. Tampon belleğin boyutu büyütülebilir.

Turbo pascalda bir dosyanın TEXT türü olduğu, VAR tanım bloğu içinde TEXT ifadesi verilerek bildirilir.

```
VAR   DosyaTanimi:TEXT
```

12.3 Text Dosyalarında Kullanılan Komutlar

12.3.1 Append

Daha önce oluşturulmuş olan bir dosyanın sonundan başlayarak, dosyaya ilave girişler yapılmasını sağlar.Genel kullanımı;

```
APPEND(Dosya Tanımı);
```

Parantez içinde verilen dosya tanımı dosyanın program içinde kullanılacak adıdır.

12.3.2 Assign

Text dosyanın manyetik ortamda hangi isimle bulunacağı belirtilir. Komut dosyayı açar ve işleme hazır hale getirir. Text dosya daha önceden yaratılmamış ve ilk bilgi girişleri yapılmamışsa tanımlanan dosyanın REWRITE komutu ile manyetik ortamda oluşması sağlanır.

```
var
dosya:TEXT
...
Assign (Dosya,'C:\deneme.dat'
```

Yukarıdaki tanımda Dosya isimli bir dosya tanıtıcısı oluşturulmuş ve bu dosya tanıtıcısının ana dizindeki deneme.dat isimli dosya ile ilişkilendirilmesi yapılmıştır.

12.3.3. Close

Bellekteki tüm bilgileri dosyaya yazdırarak dosyayı kapatır. Örnek program için program deneme text isimli programı inceleyiniz.

Örnek :

```
Program deneme  text;
var A: Text;
begin
  Assign(A, 'deneme.son');
  Rewrite(A);   { Yeni bir dosya oluşturuyor. }
  Writeln(A, 'Deneme amaçlı dosyadaki bir sözdizimi ');
```

```
  Close(A);   { Dosyayı kapatır ve değişiklikleri hafızaya alır }
  Append(A);   { Dosya sonuna eklentiyapmak için dosyayı açar }
  Writeln(A, 'Dosyanın sonuna bu satırı ilave ettim');
  Close(A);   { Dosyayı kapatır ve değişiklikleri hafızaya alır }
end.
```

12.3.4 Erase

Manyetik ortamda adı verilen programın veya dosyanın silinmesi için kullanılır. Silinecek dosya veya programın ilk karakteri FAT ve directory sektörlerinden silinerek manyetik ortamdaki alan korumaları ortadan kaldırılır.

Örnek :

```
var
 F:TEXT;
 Ch: Char;
begin
Assign(F, 'c:\deneme.son');
 {$I-}
 Reset(F);
 {$I+}
 if IOResult <> 0 then
   Writeln('Bulunamayan Dosya ', 'c:\deneme.son')
 else
 begin
   Close(F);
   Write('Erase ', , 'c:\deneme.son', '? ');
   Readln(Ch);
   if UpCase(CH) = 'E' then
     Erase(F);
 end;
end.
```

12.3.5 Read

Text dosyalarındaki kayıdın satır sonu kontrol karakterleri de dahil bilginin tamamını tanımlanan değişkenlere okur. Kullanımını daha önceden bildiğimiz READ deyiminin iki ayrı kullanımı vardır;

- Klavyeden veri girişi yapılması amacıyla kullanılabilir.
- Doğrudan erişim yapılan dosyalarda, sıralı ve rastgele kayıt okutulmasında kullanılır.

Genel Kullanımı aşağıda görüldüğü gibidir.

Read(Dosya Tanımı, Değişken1,değişken2,...,değişkenN);

Örnek:

```
Program text  dosyalarda  komutlar;
var
  F: Text;
  Ch: Char;
begin
Assign(F, 'c:\deneme.son');
  Reset(F);
  while not Eof(F) do
  begin
    Read(F, Ch);
    Write(Ch);
 end;
end.
```

12.3.6 Readln

Text dosyalarındaki kayıdın satır sonu kontrol karakterleri hariç bilginin tamamını tanımlanan değişkenlere okur.ReadLn komutunun da iki ayrı kullanım şekli vardır;

Klavyeden veri girişi için,
Sıralı erişim yapılan dosyalarda bilgi okutulmasında kullanılır. Her okunan bilgiden sonra okuyucu kafasını ve kayıt göstergesini bir sonraki bilgi üzerine kaydırır.

Genel Kullanımı aşağıda görüldüğü gibidir.

Readln (Dosya Tanımı, Değişken1,değişken2,...,değişkenN);

Örnek:

```
var
  s : String;
begin
  Write('klavyeden bir sözdizimi yazınız ');
  Readln(s);
  Writeln('Siz : ',s,'  Yazdınız');
  Writeln('Çıkış için  <Enter>');
  Readln;
end.
```

12.3.7 Rename

Manyetik ortamdaki bir program veya dosya adını değiştirmek amacıyla kullanılır. Aşağıdaki örnek programı inceleyiniz.

Örnek:

```
var
  f : file;
begin
  Assign(f,'c:\deneme.son');
  WriteLn('c:\Deneme.son',' isimli dosyanin adi ','c:\Deneme.dat',' olarak ',
                'değiştiriliyor');
  Rename(f,'c:\Deneme.dat');
end.
```

12.3.8 Reset

Daha önce oluşturulmuş olan bir Text dosyasından kayıt okunmasını sağlar. Dosya ASSIGN komutu ile açıldıktan sonra RESET dosyanın ne amaçla açıldığını kullanıcıya bildirir. Oluşturulmamış bir dosya okunmak için açılmak istenirse hata meydana gelecektir.

Örnek *:*

```
var
  F: Text;
  Ch: Char;
begin
Assign(F, 'c:\deneme.dat');
  Reset(F);
  while not Eof(F) do
  begin
    Read(F, Ch);
    Write(Ch);
  end;
end.
```

12.3.9 Rewrite

Tanımı verilen dosyanın manyetik ortamda yaratılmasını ve ilk kez bilgi girişi yapılmasını sağlar. Var olan ve daha önce bilgi girişi yapılmış olan bir text dosyası için kullanılırsa bilgilerin tamamı silinir.

Örnek:

```
var F: Text;
begin
 Assign(F, 'YENI.DAT');
 Rewrite(F);
 Writeln(F, '  YENI.DAT ISIMLI BIR TEXT DOSYA OLUŞTURULDU ');
 Close(F);
end.
```

12.3.10 Write

Kullanımını daha önce bildiğimiz WRITE deyiminin üç ayrı fonksiyonu vardır.

- Ekranda kendisinden sonra tanımlanan bilgilerin görüntülenmesini ve kursörün bu bilgilerin sonunda kalmasını sağlar.
- Girilen kayıt bilgilerinin dosyaya yazdırılmasında kullanılır.
- Dosya kayıtlarının LST parametresi ile yazıcıya gönderilmesini sağlar.

Örnek :

```
var
  F: Text;
  Ch: Char;
begin
  Assign(F, c:\deneme.dat);
  Reset(F);
  while not Eof(F) do
  begin
    Read(F, Ch);
    Write(LST,Ch);
  end;
end.
```

12.3.11 Writeln

Writeln komutunun da Write gibi üç ayrı fonksiyonu vardır.

- Ekrana, beraberinde tanımlanan bilgilerin görüntülenmesini ve kursörün bir alt satıra geçmesini sağlar. Kendisinden sonra herhangi bir parametre yazılmamış ise bir boş satır atlatır.
- Girilen kayıt bilgilerinin dosyaya yazdırılmasında kullanılır.
- Dosya bilgilerinin LST parametresi ile yazıcıya gönderilmesini sağlar.

12.4 Text Dosyalarında Kullanılan Fonksiyonlar

Text dosyalarında kullanılan fonksiyonlar, alfabetik sıraya göre aşağıda tanımlanmış ve iyi anlaşılabilmesi amacıyla kullanımlarına ilişkin birer örnek verilmiştir.

12.4.1 Blockread

Tipi belirli olmayan bir dosyadaki kayıt adedini okuyarak tanımlanan başka bir değişkene aktarır ve okunan toplam kayıt sayısını gösterir.

12.4.2 BlockWrite

Tipi belirli olmayan bir dosyadan (girdi) okunarak başka bir değişkene aktarılan kayıtların, tipi belirli olmayan başka bir dosyaya (çıktı) yazılmasını sağlar ve yazılan gerçek kayıt sayısını belirtir.

Örnek:

```
var
   girdi,cikti:file
  i:byte;
  NumRead, NumWritten: Word;
  Buf: array[1..2048] of Char;
begin
  Assign(girdi 'c:\deneme.son'); { Girdi dosyasını açıyor}
  Reset(girdi, 1);  { Kayıt boyutu = 1 }
  Assign(cikti, 'c:\deneme1.son'); { Çıktı dosyası açılıyor }
  Rewrite(cikti, 1);  { Kayıt boyutu = 1 }
  Writeln('Copying ', FileSize(girdi), ' byte...');
  for i:=1 to 5 do begin
    BlockRead(girdi, Buf, SizeOf(Buf), NumRead);
    BlockWrite(cıktı, Buf, NumRead, NumWritten);
    Writeln(NumRead, '  ',NumWritten);
  end;
  Close(girdi);
  Close(cikti);
ReadLn;
end.
```

12.4.3 Chdir

Manyetik ortamda çalışılan dizinin değiştirilmesini sağlar.

Örnek :

```
uses crt; {Windows için Wincrt}
var dizin:string;
begin clrscr;
  Write('Dosyanın yazdırılacağı çalışma dizini  =');
  Readln(dizin);chdir(dizin);
  if ioresult =0 then writeln ('çalışma dizini değiştirildi');
  Writeln('Çıkış için ENTER');
  Readln;
end.
```

12.4.4 Eof

(End Of File) Dosya sonu göstergesidir. Program içinde okunmakta olan dosya sonuna gelinip gelinmediğini kontrol etmek amacıyla kullanılır. Son kayıttan sonra <EOF> işareti görüldüğünde kontrol göstergesi TRUE durumuna geçerek dosya sonuna ulaşıldığını belirtir.

Örnek:

```
var
  F: Text;
  Ch: Char;
begin
Assign(F, 'deneme.son');
  Reset(F);
  while not Eof(F) do
  begin
    Read(F, Ch);
    Write(Ch);
  end;
end.
```

12.4.5 Eoln

(End Of Line) satır sonu göstergesidir. Text dosyalarında okunan satır sonuna gelinip gelinmediğini kontrol etmede kullanılır. Satırın sonunda <Cr> veya <LF> işareti görüldüğünde kontrol göstergesi TRUE durumuna geçerek satır sonu olduğunu belirtir.

Örnek:

```
uses crt;{Windows için Wincrt}
var x1:text
```

```
satır :string[50];
begin       clrscr;
assign(x1,'c:\deneme.son');
rewrite(x1);
for i:=1 to 4 do
  begin
    write(i'. Satır');
    Readln(satir);
  end;
  close(x1);
assign(x1,'c:\deneme.son');
reset(x1);
if eoln(x1) then halt;
  readln(x1,satir);
  Writeln(Satir);
Close(x1);
end.
```

12.4.6 Filepos

Bir program çalıştırıldığında erişilen bilgi dosyasında okuma / yazma kafasının konumlandırıldığı kayıt numarasını verir.

12.4.7 Filesize

Pascal programının çalıştırılmasıyla aktif duruma alınan dosyaların kayıt uzunluğunu verir.

Örnek:

```
var
  f: file of byte;
  size : Longint;
begin
  Assign(f, '.\deneme.son');
  Reset(f);
  size := FileSize(f);
  Writeln('dosyanın boyutu (byte)',size);
  Seek(f,size div 2);
  Writeln('Dosyanın Şu anki konumu ',FilePos(f));
  Close(f);
end.
```

12.4.8 Flush

Sıralı erişim dosyalarında (TEXT) girilen bilgilerin biriktirildikleri geçici bellek dolmadan, bilgileri manyetik ortama yazdırır.

Örnek:

```
uses crt;{windows için Wincrt}
var          con:text;
  satir:array[1..4] of string;
  i:byte;
begin       clrscr;
  Assign(con,'a:\deneme.son');
  Rewrite(con);
  for I:=1 to 10 do
    begin
      Write(i,'. satır');Readln(satir[i]);
      Writeln(con,satir[i]);
    end;
  flush(con);close(con);
  assign(con,'a:\deneme.son');
  reset(con);clrscr;i:=0;
  While not eof(con) do
     begin
       i:=i+1;
       Readln(con, satir[i]);
       Writeln(i, '. satir  =',satir[i]);
     end;
  Readln;Close(con);
  End.
```

12.4.9 Getdir

Aktif durumda olan çalışma ortamı (sürücü) içinde aktif çalışma dizinini verir.

Sürücü Tanımında	0	Aktif durumdaki sürücü
	1	A flopy disk sürücüsü
	2	B flopy disk sürücüsü
	3	C sabit disk sürücüsü

Örnek:

```
var
  s : String;
begin
  GetDir(0,s); { 0 = geçerli sürücü }
```

```
  Writeln(' Gecerli surucu ve dizin: ',s);
end.
```

12.4.10 Ioresult

Bir program çalıştırıldığında giriş/çıkış işlemleri sırasında oluşan hatanın kodunu belirtir. Dönen değer sıfırdan farklı ise bir giriş/çıkış hatasının oluştuğu anlaşılır.

Örnek:

```
var F:TEXT;
begin
 Assign(F, 'c:\deneme.son');
 {$I-}
 Reset(F);
 {$I+}
 if IOResult = 0 then
   Writeln('Dosya boyutu    (byte) ', FileSize(F))
 else
   Writeln('Dosya Bulunamadı ');
End.
```

12.4.11 Mkdir

Programın çalıştırılması sırasında gerekli olduğu hallerde dış ortamda yeni bir çalışma dizini açılmasını sağlar.

Örnek:

```
VAR I:STRING;
begin
 {$I-}
 I:='COM';
 MkDir(I);
 if IOResult <> 0 then
   Writeln('Dizin yaratılamıyor')
 else
   Writeln(i, ' isimli yeni dizin yaratıldı');
Readln;end.
```

12.4.12 Rmdir

Manyetik ortamda bulunan bir çalışma dizininin silinmesini sağlar.

Örnek:

```
VAR I:STRING;
begin
  I:='COM';
  {$I-}
  RmDir(I);
  if IOResult <> 0 then
    Writeln(i,' İsimli Dizin Silinemiyor')
  else
    Writeln(i,' İsimli  Dizin Silindi');
end.
```

12.4.13 Seek

Manyetik ortamda, adı tanımlanan bir dosya göstergesini bir değişken ile belirtilen kayıt üzerine yönlendirir. Böylece manyetik ortamda sürücü okuma/yazma kafası doğrudan erişilmek istenilen kayıt üzerine kaydırılır.

Örnek:

```
var
  f: file of Byte;
  size : Longint;
begin
Assign(f, 'c:\deneme.dat');
  Reset(f);
  size := FileSize(f);
  Writeln('Dosya Boyutu: ',size);
  Seek(f,size div 2);
  Writeln('Dosyanın Pozisyonu ',FilePos(f));
  Close(f);
end.
```

12.4.14 Seekeof

EOF kontrolü gibi çalışır. Ancak bu fonksiyon dosya sonu işaretinden sonra olan boşlukları veya satır sonu işaretlerini dikkate almaz.

12.4.15 Seekeoln

EOLN kontrolü gibi çalışır. Ancak bu fonksiyon satır sonu CR ve LF işaretinden sonraki boşlukları dikkate almaz.

Herbiri en fazla 70 karakter uzunluğundaki satırların, manyetik ortamda 'ENDUST.SON' isimli bir dosyaya yazdırılması ve Seekeoln ile satır sonuna kadar, Seekeof ile dosya sonuna kadar kontrol edilerek okunması ile ilgili örnek program aşağıda verilmiştir.

Örnek:

```
Uses crt; {windows için Wincrt}
var Dosya:file of string;
     Line:String[70];
begin
clrscr;
  Assign(Dosya,'\END.SON');
  ReWrite(Dosya);
  Writeln(Dosya,'Endüstri mühendisliği son yıllarda ');
  Writeln(Dosya,'çağımızın kilit mesleği haline gelmiştir.');
  Writeln(Dosya,'Bir  mühendis, düşünce geliştirebilen kişidir');
  Writeln(Dosya,'Mühendislik, düşünce geliştirildiği sürece anlam kazanır');
  Reset Dosya);
  While not SeekEof (Dosya) do
            begin
              if SeekEoln(Dosya) then Readln;
              (* Satır sonu ise okuma kafası bir sonraki satırın başına atlatılır.*)
              Read(dosya,Line);
              Writeln(line);
            end;
  Close(Dosya);
  Writeln('Çıkış için herhangi bir tuşa basınız');
  repeat until keypressed;
  end.
```

12.4. 16 Settextbuf

Pascal, text bilgileri için bellekte standart 128 byte' lik yer ayırır. Settextbuf fonksiyonu ile bu alan büyüklüğü değiştirilebilir. Aşağıda verilen örnekte bu alan büyüklüğü 10 Kbyte'e yükseltilmiştir. Böylece verilerin daha hızlı okunması da sağlanmış olmaktadır.

Örnek:

```
uses crt; {windows için Wincrt}
var
  k  : text;
  ch : Char;
  buf: array[1..10240] of Char; { 10K buffer}
begin
  clrscr;
  Assign(k, 'c:\deneme.dat');
  SetTextBuf(f, buf);
```

```
  Reset(k);
  while not Eof(k) do
  begin
    Read(k, ch);
    Write(ch);
  end;
end.
```

12.4.17 Truncate

Bir dosyadan herhangi bir kayıdın silinmesi işleminin yapılmasından sonra devamındaki kayıtların kaydırılarak yazılması sonucu dosya sonunda bulunan <EOF> göstergesinin kaydırılmasını sağlar.

Örnek:

```
var
  f: file of Integer;
  i,j: Integer;
begin
  Assign(f,'c:\deneme.dat');
  Rewrite(f);
  for i := 1 to 6 do
    Write(f,j);
  Writeln('Bilgilerin kesilmesinden önce:');
  Reset(f);
  while not Eof(f) do
  begin
    Read(f,j);
    Writeln(j);
  end;
  Reset(f);
  for i := 1 to 3 do
  Read(f,j);
  Truncate(f); { Bilgiler kesiliyor }
  Writeln;
  Writeln('Bilgilerin kesilmesinden sonra :');
  Reset(f);
  while not Eof(f) do
  begin
    Read(f,j);
    Writeln(j);
  end;
  Close(f);
  Erase(f);
end.
```

12.5 Tipleri Belirlenmiş Dosyalar

Text dosyalarında depolanan bilgilerin ASCII formatında saklandığını ve bunların çok değişik veriler içerebildiğini biliyoruz. Tiplendirilmiş veya tipleri belirlenmiş dosyalar ise belirli türden verileri veya veri gruplarını içerirler.

Tipleri belirlenmiş dosyalar, PASCAL'ın yapısal özelliğini yansıtırlar. Her kaydın sabit bir tipi ve büyüklüğü olması nedeniyle text dosyalara oranla daha yapısal bir görünümleri vardır. Manyetik ortama yazdırılacak bilgiler üç ayrı yapıda FILE OF ifadesi ile verilir.

12.5.1 Integer Tipi

Dosyanın, sadece tam sayılardan oluşan sayısal bilgilerden oluştuğunu ifade etmek için kullanılır.
Dosyadaki sayısal bilgiler -32768 ile 32767 arasındaki tamsayılardan oluşabilir. Tam sayılardan oluşan ve DOSYA isimli dosya tanıtıcısı ile temsil edilen bir dosya Var bloğunda aşağıdaki şekilde tanıtılır.

```
VAR  DOSYA:FILE OF integer;
```

12.5.2 Real Tipi

Dosyanın, sadece reel sayılardan oluşan sayısal bilgilerden oluştuğunu ifade etmek için kullanılır. Dosyadaki sayısal bilgiler, 2.9*19E-39 ile 1.7*10E+38 arasındaki üssel formatta sadece ondalıklı sayılardan oluşabilir. Reel sayılardan oluşan ve DOSYA isimli dosya tanıtıcısı ile temsil edilen bir dosya Var bloğunda aşağıdaki şekilde tanıtılır.

```
VAR DOSYA:FILE OF Real;
```

Örnek:

```
var
  DOSYASON: file of REAL;
  size : Longint;
begin
  Assign(DOSYASON, 'c:\aodrcl\deneme.txt');
  Reset(DOSYASON);
  size := FileSize(DOSYASON);
  Writeln('DOSYANIN BOYUTU  ',size,' BYTE');
  Close(f);
end.
```

12.5.3. Record

Kayıt yapısında birbirinden farklı karakter tipleri olan satır bilgileri topluluğudur. Bilgilerin, kayıt yapısında tanımlanacağı TYPE bloğunda RECORD ifadesiyle belirtilir. Aşağıda verilen örneği inceleyiniz.

```
TYPE    StokKontrol=RECORD
          cinsi:string[20];
          miktar:integer;
          brmfiat,tplmtut:real;
end;
VAR     STOK:StokKontrol;
          StkDosya:FILE of STOK;
```

Type bloğunda kayıt yapısı içinde tanımlı olan alanların bilgi tipleri STRING, REAL veya INTEGER olabilir. VAR bloğunda program içerisinde kullanılacak kayıt adı ve dosya adı tanımlanır.

Örnek

```
program record  ornek;
uses crt; {windows için Wincrt}
type
   DT  Tipi=RECORD
      gun,ay,yil:word;
          end;
   KimlikTipi=RECORD
         adi:string[10];
       soyadi:string[20];
  DogumTarihi:DT  Tipi;
    DogumYeri:String[15];
       geliri: Real;
       Evlimi: Char;
       end;
var personel :KimlikTipi;
   personelkutugu:file of KimlikTipi;
begin
    assign(personelkutugu,'.\kutuk');
    Rewrite(personelkutugu);
    clrscr;
    WITH personel do
     begin
        Write('Personelin Adı          :');Readln(Adi);
        Write('Personelin Soyadı       :');Readln(SoyAdi);
        WITH Dogumtarihi do
         begin
           Write('Personelin Doğduğu Gün  :');Readln(gun);
           Write('Personelin Doğduğu Ay   :');Readln(Ay);
           Write('Personelin Doğduğu Yıl  :');Readln(Yil);
```

```
            end;
            Write('Personelin Doğum Yeri   :');Readln(Dogumyeri);
            Write('Personelin Geliri       :');Readln(Geliri);
            Write('Personel Evli mi?  [E/H]:');Readln(Evlimi);
         end;
        Write(personelkutugu,personel);
        end.
```

12.6 Tipleri Belirli Olmayan Dosyalar

Bu tür dosyalar, daha önceden oluşturulmuş, fakat kayıt yapıları ve bilgi tipleri belirli olmayan dosyalarla yapılacak işlemlerde kullanılmalarında verilen bir genel tanımlamadır. Bilgi tipleri bilinmeyen dosyalar, manyetik ortamda kapladıkları alan ölçüsüne göre blok blok okutulurlar ve yine gerekiyor ise blok blok başka bir dosyaya yazdırılırlar. VAR bloğunda okunacak dosyanın bilgilerinin aktarılacağı tampon bellek için BYTE olarak, dosyanın alanına eşit veya daha büyük dizi yaratılır. Şayet tampon bellek için alan az ise okuma bu oranda yapılır. Tipi belirsiz bir dosya açıldıktan sonra RESET komutuyla bilgiler okutulmalıdır.

```
RESET(DOSYAADI,BYTE  miktarı);
```

Komut parametresinde byte miktarı olarak 1 verilirse, bir defada 256 karakter okutulacağı belirtilir. Ancak, yazdırma işleminde de 1 byte verilirse hedef dosyaya bir defada 256 karakter yazdırılması sağlanır.

```
RESET(Dosya,1);

REWRITE(Dosya,1)
```

Örnek:

```
Uses crt; {windows için Wincrt}
var
Dosya1,Dosya2:file;
  alani:integer;
  Buf:array[1..1024] OF BYTE;
  DosyaA1,DosyaB2:string[20];
begin clrscr;
  Write('Blok Blok okuma yapılacak dosya adı :');
  ReadLn(DosyaA1);
  Write('Blok Blok yazdırma yapılacak dosya adı :');
  ReadLn(DosyaB2);
  Assign(Dosya1,DosyaA1);
  Reset(Dosya1,1);
  BlockRead(Dosya1,Buf,SizeOf(Buf),Alani);
  Assign(Dosya2,DosyaB2);
  Rewrite(Dosya2);
```

```
    BlockWrite(Dosya2,Buf,Alani);
    Close(Dosya1);Close(Dosya2);
    writeln('işlem tamamlandı');ReadLn;
end.
```

12.7 Doğrudan Erişimli Dosyalar

Aynı konuyla ilgili bilgilerin, belirtilen isim altında manyetik ortama yazdırılma ve girilecek bilgilerin neler olduğu, hangi karakter tipinde oldukları uzunlukları ile verilerek kayıt düzeni tanımlanır. Bir kayıt düzeni içinde girilecek bilgiler farklı karakter ve uzunluklarda olabilir. Ancak bu bilgilerin toplamı, dosyanın standart kayıt uzunluğu ve dosya kapasitesi ise girilen kayıt adedi toplamı olacaktır. Her kayıt içinde satır satır girilecek farklı bilgiler tanımlanmadan önce herbirine girilecek en uzun karakter miktarı tasarlanmalıdır. Bu satırlara saha denilmektedir. Örneğin, isim girişleri yapılacak saha uzunluğu olabilecek en uzun isme göre verilir. Bir kayıt için her sahaya ayrı ayrı isimler verilerek, girilen bilgilere erişimde bu isimler kullanılır. Örneğin bir işletmedeki personele ilişkin bilgiler aşağıdaki şekilde bulunabilir.

```
PersonelAdı      :Sabahattin Kavaklıgillerden
Görevi           :Kalite Kontrol  Mühendisi
Çalıştığı Birim  :ARGE
Yaşı             :28
```

Görüldüğü gibi dosyaya yazdırılan kayıtlar ilgili kayıt alanlarına göre belirli uzunlukta olurlar ve kayıt düzeni içinde sahaların bilgi tipleri kesinlikle verilmelidir.

Değişik kayıtların pozisyonlanması SEEK komutu ile gerçekleştirilir. SEEK komutunun genel kullanımı aşağıdaki şekildedir.

```
SEEK(Var Dosya;AnahtarNo:longint)
```

Bu yazım şeklinde DOSYA parametresi tiplendirilmiş veya tiplendirilmemiş bir kütük tanıtıcısını gösterir. AnahtarNo ise olunacak kayıt numarasıdır. İlk kayıt numarası 0 ile başlar. Kayıt numarası longint ile verildiğinden doğrudan erişimli dosyalarda bulunabilecek kayıt sayısı iki milyarı geçer.

Giriş, arama ve düzeltme, kayıt iptal, bilgilerin ekranda listelenmesi ve bilgilerin yazıcıda yazdırılması işlemlerini kapsayan doğrudan erişimli bir programın yapısı ve algoritması en basit şekli ile aşağıdaki örnekte olduğu gibi verilmelidir.

12.8 Örnek Programlar

Örnek: Aşağıdaki örnek program ile text dosya kullanarak bir rehber oluşturulmuştur.

```
uses crt; {windows için Wincrt}
var f,t:text;
  YAN,ch:char;
  s,z:string;
  i:byte;
Procedure buyuk;
begin
     for i := 1 to Length(s) do
      s[i] := UpCase(s[i]);
end;
procedure ekle;
      begin
       clrscr;
       append(f);
       write('Ad soyad.(max.20 harf): ');readln(s);
       buyuk;
       s:=s+'                    ';          s:=copy(s,1,20);write(f,s);
       write('Telefon..(max.20 harf): ');readln(s);
       s:=s+'                    ';          s:=copy(s,1,20);write(f,s);
       write('Adres....(max.30 harf): ');readln(s);
       buyuk;
       s:=s+'                              ';s:=copy(s,1,30);writeln(f,s);
       close(f);
      end;
procedure liste;
       begin
       clrscr;
       writeln('AD SOYAD            TELEFON             ADRES');
       writeln('-------------------- -------------------- ------------------------------');
       reset(f);
       repeat
        readln(f,s);
        writeln(s);
       until eof(f);
       ch:=readkey;if ch=#0then ch:=readkey;
      end;
procedure ara;
  begin
       clrscr;
       write('Ad soyad.(max.20 harf): ');readln(z);
       for i := 1 to Length(z) do
          z[i] := UpCase(z[i]);
       z:=z+'                    ';z:=copy(z,1,20);
```

```
            writeln('Aranıyor...');
            reset(f);
            repeat
             readln(f,s);
             if copy(s,1,20)=z then
              begin
               writeln('AD SOYAD            TELEFON            ADRES');
               writeln('-------------------- -------------------- ------------------------------');
               writeln(s);
               s:='';
               write('Aramaya devam?(E/H): ');
               repeat ch:=upcase(readkey);until ch in['E','H'];
               if ch='E'then begin gotoxy(1,wherey-3);
        end else break;
              end;
            until eof(f);
            write('Dosya sonu!'#7);
            ch:=readkey;if ch=#0then ch:=readkey;
           end;
        procedure silme;
          begin
            clrscr;
            write('Ad soyad.(max.20 harf): ');readln(z);
            for i := 1 to Length(z) do
                z[i] := UpCase(z[i]);
            z:=z+'                    ';z:=copy(z,1,20);
            writeln('Aranıyor...');
            assign(t,'rehber.tmp'); rewrite(t);
            reset(f);
            repeat
             readln(f,s);
             if copy(s,1,20)<>z then writeln(t,s);
            until eof(f);
            close(f);
            close(t);
            erase(f);
            rename(t,'rehber.tel');
           end;
        procedure degistir;
          begin
            clrscr;
            write('Ad soyad.(max.20 harf): ');readln(z);
            for i := 1 to Length(z) do
              z[i] := UpCase(z[i]);
            z:=z+'                    ';z:=copy(z,1,20);
            writeln('Aranıyor...');
            assign(t,'rehber.tmp'); rewrite(t);
            reset(f);
            repeat
```

```
                readln(f,s);
                if copy(s,1,20)<>z then writeln(t,s);
              until eof(f);
              close(f);
              close(t);
              erase(f);
              rename(t,'rehber.tel');
              clrscr;
              append(f);
              write('Yeni Ad soyad.(max.20 harf): ');readln(s);
              buyuk;
              s:=s+'                    ';          s:=copy(s,1,20);write(f,s);
              write('Yeni Telefon..(max.20 harf): ');readln(s);
              s:=s+'                    ';          s:=copy(s,1,20);write(f,s);
              write('Yeni Adres....(max.30 harf): ');readln(s);
              buyuk;
              s:=s+'                              ';s:=copy(s,1,30);writeln(f,s);
              close(f);
             end;
          procedure tumunusil;
            begin
             write('BÜTÜN KAYITLARI SİLMEK İSTEDİĞİNİZE EMİN MİSİNİZ ? (E/H):');
             repeat
               READLN(YAN);
                    Yan:=UPCASE(yan);
             Until yan in ('E','H');
             IF YAN='E' THEN
               begin
                 rewrite(f);
                 close(f);
               end;
          end;
          procedure cikis;
          begin
          clrscr;
          writeln('REHBER 1.0, BALIKESİR ÜNİVERSİTESİ');
          halt;
          end;
          begin
          assign(f,'rehber.tel');
          {$i-}reset(f);{$i+}
          if ioresult<>0then rewrite(f);
          repeat
           repeat
            clrscr;
            writeln('1) Kayıt ekle');
            writeln('2) Kayıt listele');
            writeln('3) Kayıt arama');
            writeln('4) Kayıt silme');
```

```
  writeln('5) Kayıt değiştirme');
  writeln('6) Kayıtları sıfırla!');
  writeln('7) DOS''a çıkış!');
  ch:=readkey;
 until ch<>#7;
  case ch of
    '1':Ekle;
    '2':Liste;
    '3':Ara;
    '4':Silme;
    '5':Degistir;
    '6':Tumunusil;
    '7':cikis;
  end;
until ch=#7;
end.
```

Örnek: Aşağıdaki örnek program random dosya kullanarak bir işyerindeki personel ile ilgili bilgileri işlemektedir.

```
uses printer,crt {windows için Wincrt};
label basla,gir,giris,arama,iptal,liste1,liste2,son;
type PerKayit=Record
     isim:string[20];
     sicno:integer;
     bolum:string[10];
     sskno:longint;
     tahsil:string[15];
     girtar:string[9];
     maas:real;
end;
var
   Kayit:PerKayit;
   Dosya:File Of Perkayit;
   KayitNo:Integer;
   Sec,i:byte;
   sor:char;
procedure menu;
begin
        gotoxy(28,08);Write('Program Yönetim Menüsü');
        gotoxy(26,10);Write('1. Bilgi Girişi');
        gotoxy(26,11);Write('2. Arama ve Düzeltme');
        gotoxy(26,12);Write('3. Kayıt Silme');
        gotoxy(26,13);Write('4. Ekrana Listeleme');
        gotoxy(26,14);Write('5. Yazıcıya Listeleme');
        gotoxy(26,15);Write('6. Çıkış');
        gotoxy(30,18);Write('Seçeneğiniz :');
```

```
            Readln(sec);
        end;

        Procedure BilgiSatir;
        begin
            with kayit do
              begin
                gotoxy(17,10);Write('Persolelin Adı       :');
                gotoxy(17,11);Write('Kurum Sicil No     :');
                gotoxy(17,12);Write('Çalıştığı Birim       :');
                gotoxy(17,13);Write('SSK Sicil No         :');
                gotoxy(17,14);Write('Öğrenim Durumu  :');
                gotoxy(17,15);Write('İşe Başlama Tarihi :');
                gotoxy(17,16);Write('Aylık ücreti       :');
              end;
        end;

        Procedure KayitGiris;
        begin
            with kayit do
              begin
                gotoxy(38,10);Readln(isim);
                gotoxy(38,11);Readln(sicno);
                gotoxy(38,12);Readln(bolum);
                gotoxy(38,13);Readln(sskno);
                gotoxy(38,14);Readln(tahsil);
                gotoxy(38,15);Readln(girtar);
                gotoxy(38,16);Readln(Maas);
              end;
            Write(Dosya,Kayit);
        end;
        Procedure DoluKayit;
        begin
            with kayit do
              begin
                gotoxy(17,10);Write('Persolelin Adı     :',isim);
                gotoxy(17,11);Write('Kurum Sicil No     :',sicno);
                gotoxy(17,12);Write('Çalıştığı Birim    :',bolum);
                gotoxy(17,13);Write('SSK Sicil No       :',sskno);
                gotoxy(17,14);Write('Öğrenim Durumu     :',tahsil);
                gotoxy(17,15);Write('İşe Başlama Tarihi :',girtar);
                gotoxy(17,16);Write('Aylık ücreti       :',maas);
              end;
        end;

        Begin
        Basla:clrscr;Menu;
              Case sec of
                1:goto giris;
```

```
            2:goto arama;
            3:goto iptal;
            4:goto liste1;
            5:goto liste2;
            6:goto son;
          else goto basla
          end;
Giris:
      Assign(dosya,'a:\per.dat');
      Rewrite(Dosya);
Gir:clrscr;
      gotoxy(17,08);Write('Bilgi Girişleri');
      BilgiSatir;KayitGiris;
      gotoxy(17,18);Write('Başka Giriş  Var mı [E/H]:');ReadLn(Sor);
      if sor in['E','e'] then goto gir else close(Dosya); Goto Basla;

Arama:Assign(dosya,'a:\per.dat');
      Reset(Dosya);Clrscr;
      Gotoxy(17,08);Write('Aranan Kayıt Numarası :');
      Readln(Kayitno);Seek(Dosya,KayitNo-1);
      Read(Dosya,Kayit);DoluKayit;
      Gotoxy(17,18);Write('Başka Arama Yapılacak mı [E/H]:');Readln(Sor);
      if sor in['E','e'] then goto arama else close(Dosya); Goto Basla;

Iptal:clrscr;
      Assign(dosya,'a:\per.dat');
      Reset(Dosya);Clrscr;
      Gotoxy(17,08);Write('İptal Edilecek Kayıt Numarası :');
      ReadLn(KayitNo);DoluKayit;
      for i:=kayitno+1 to Filesize(Dosya)-1 do
         begin
           seek(dosya,i); Read(Dosya,Kayit);
           seek(dosya,i-1); Write(Dosya,Kayit);
         end;
      Truncate(Dosya);
      Gotoxy(17,19);Write('Başka İptal  [E/H]');
      ReadLn(Sor);
      if sor in['E','e'] then goto Iptal else close(Dosya); Goto Basla;
Liste1:
      Assign(dosya,'a:\per.dat');
      Reset(Dosya);Clrscr;
      While not eof(Dosya) do
           begin read(Dosya,kayit);
             with kayit do
             begin
               Writeln(isim,Chr(32):20-length(isim),sicno:5,' ',
                       bolum,chr(32):10-length(bolum),sskno:8,' ',
                       tahsil,chr(32):15-length(tahsil),girtar,' ',
                       maas:8:0);
```

```
                end;
            end;Writeln;
            repeat
              Write('Menüye Dönüş İçin Herhangi bir Tuşa basınız');
            Until Keypressed;
            Close(dosya);Goto Basla;
Liste2:Assign(dosya,'a:\per.dat');
      Reset(Dosya);Clrscr;
      While not eof(Dosya) do
            begin read(Dosya,kayit);
              with kayit do
              begin
                Writeln(LSt,isim,Chr(32):20-length(isim),sicno:5,' ',
                        bolum,chr(32):10-length(bolum),sskno:8,' ',
                        tahsil,chr(32):15-length(tahsil),girtar,' ',
                        maas:8:0);
              end;
            end;Close(dosya);Goto Basla;
son:
end.
```

Programın inceliklerinden biri, listelemede bilgilere verdirilen boşluklardır.

```
Writeln(isim,chr(32):20-length(isim),.....
```

satırında isim sahasındaki bilginin uzunluğu ne olursa aolsun bu alan 20 karakterden oluşmuşcasına boşluklar verilmiştir. Şayet listeleme komutu *Writeln(isim:20...)* şeklinde ise saha bilgisi aşağıda görüldüğü gibi sağa dayalı olarak görülecekti.

```
----------ATIL
```

Diğer bir incelik, kayıt iptal bölümünde, iptal için girilen kayıt nosu devamından başlayarak son kayıda kadar olan tüm kayıtlar bir döngü ile birer kayıt geri kaydırılırlar. Silinmek için girilen kayıttan bir sonraki kayıda erişilerek içeriği okutulur.

```
SEEK(DOSYA,i);  Read(Dosya,Kayit);
```

Bilgiler okunduktan sonra silinmesi istenilenin üzerine okuma/yazma kafası kaydırılarak yazdırılır.

```
SEEK(DOSYA,i-1);          Write(Dosya,Kayit);
```

BÖLÜM 13

UNIT PROGRAMLAR

Pascalın en büyük özelliklerinden biri yazılan program parçalarının diğer Pascal programları tarafından paylaşılıp, o program içindeki alt programların diğer programlar tarafından kullanılabilmesidir. Pascalda, diğer programların kullanabileceği pascal programlarına unit programlar denilmektedir.

Bu programlar derlendikten sonra Turbo Pascal tarafından .TPU (Windows 'ta .TPW, korumalı modda .TPP) uzantısını alırlar. Unitler birer bağımsız programdırlar, ancak kendi başlarına çalıştırılamazlar. Kendilerini çalıştıran bir ana program vasıtasıyla çalışabilirler.

Unitler, bir program hacminin 64 KB sınırını aşması veya birden fazla program tarafından kullanılabilecek alt programları tekrar yazmamak için oluşturulurlar. Ana programın derlenmesi sırasında çalıştırılmak istenilen unitler de derlenirler.

Bir unit programın genel yapısı şu şekildedir.

```
Unit  Unitadi;
interface
procedure  altprogadi1(...);
procedure  altprogadi2(...);
function function1(....):...;
function function2(....):...;
var
....
type
....
const

implementation
```

```
procedure  altprogadi1(...);
begin
        {işlemler}
end;

procedure  altprogadi2(...);
begin
        {işlemler}
end;

function function1(....):...;
begin
        {işlemler}
end;

function function2(....):...;
begin
        {işlemler}
end;
```

end.

Ana program ile unitler arasındaki önemli farklılıklardan biri de unitlerde, mutlaka UNIT komutu ile başlayan ve unitin adını belirten bir unit başlığı kullanılması zorunludur. Unit adı sekiz karakteri geçemez ve burada verilecek unit adı, programın diske kaydedilirken kullanılan ad ile aynı olmalıdır.

Bir unit programın ana program tarafından kullanılabilmesi için ana programda USES komutuyla birlikte unit adının yazılması gereklidir. Örneğin; Sekiller isimli unitin ana program tarafından kullanılabilmesi için ana programda aşağıdaki satırlar yazılmalıdır.

```
Program sekillerin  alan  hesabi;
USES sekiller;
.....
```

Bir unit, interface ve implementation olmak üzere iki bölümden oluşur.

Interface bölümünde, bu uniti kullanan program tarafından kullanılacak olan değişken, sabit, procedure ve function alt programlar yazılır. Ayrıca bu bölümde, bu unit programın kullanacağı hazır unitler ve kullanıcı unitleri de USES komutu ile tanıtılır. Bu bölümde yapılan tüm tanımlamalar ana program tarafından tanınmaktadır.

Implementation bölümünde ise geçerli unit içinde bulunan ancak ana program tarafından kullanılmayan alt programlar yazılır. Burada değişken, sabit ve tip tanımlamaları da yapılabilir. Burada yapılan tanımlamalar sadece unit içerisinde geçerlidir.

Unite ait en son "End." deyiminden önce yazılan komutlar varsa, ana program çalışmaya başladığında ilk olarak bu komutlar çalıştırılır. Bu komutların çalışması için end 'den

önce begin deyimi kullanılmalıdır. İki unit program birbiri içindeki alt programları da kullanabilirler.

```
unit sekiller;
interface
var
uzunluk,alan:integer;
procedure alan  hesabi;
implementation

procedure alan  hesabi;
begin
    write('Karenin kenar uzunluğunu giriniz  =');
    Readln(uzunluk);
    alan:=sqr(uzunluk);
    Writeln('Karenin Alanı =',Alan);
end;
begin
     writeln('KARE ALANININ HESAPLANMASI');
     writeln;
end.
```

```
program kare  alan  hesabi;
uses sekiller;
begin
    alan  hesabi;
    writeln('Alan hesabı işlemi tamamlandı');
end.
```

Program çalıştırıldığında once unit programdaki;

```
writeln('KARE ALANININ HESAPLANMASI');
writeln;
```

satırları işleme alınır, sonra alan hesabi isimli procedure alt programı çalışır. Alt programdaki işlemler yapıldığında ise ana programdaki;

```
writeln('Karenin Kenar uzunluğu =',uzunluk);
writeln('Alan hesabı işlemi tamamlandı');
```

satırı çalışarak programın çalışması sona ermektedir. Ana programda uzunluk isimli bir değişken tanımlı olmamasına rağmen, uzunluk değişkeni unit programda interface altında tanımlandığından ana program tarafından tanınmaktadır.

Aşağıda verilen örnek programda iki unit programın birbiri içindeki alt programları kullanmasına aittir. Burada birici unitteki alt program kullanılacak ise interface 'den sonraki komut satırında USES ile birlikte kullanılacak unitin adı yazılır. Birinci unit programda ise ikinci unit programın kullanılacağı implementation deyiminden sonraki USES komutunda verilecektir.

Aşağıda verilen örnekte bir karenin alanı ve çevresi hesaplanmaktadır. USES komutlarının nasıl kullanıldığına ve deger gir isimli alt programın parametreli olduğuna dikkat ediniz. Deger gir isimli alt programda parametre transferi yapılmaz ise diğer unit bu değeri sıfır olarak algılamaktadır.

```
unit sekiller;
interface
uses wincrt;
var
uzun,uzunluk,alan:integer;
procedure alan  hesabi;
procedure deger  gir(VAR UZUNLUK:INTEGER);
implementation
USES CEVRE;
procedure deger  gir(VAR UZUNLUK:INTEGER);
begin
  write('Karenin kenar uzunluğunu giriniz  =');
  Readln(uzun);
  uzunluk:=uzun;
end;
procedure alan  hesabi;
begin
    cevre  hesapla;
    alan:=sqr(uzun);
    Writeln('Karenin Alanı          =',Alan,' birim²');
end;
begin
writeln('KARENİN ÇEVRE ve ALANININ HESAPLANMASI');
writeln;writeln;
end.

unit cevre;
interface
uses SEKILLER,wincrt;
var
uzunluk:integer;
procedure cevre  hesapla;
implementation
procedure cevre  hesapla;
begin
   deger  gir (UZUNLUK);
```

```
    Writeln('Karenin Çevresi          =', 4*uzunluk;,' birim');
end;
end.

Program kare  ile  ilgili;
uses sekiller,WINCRT;
begin
     alan  hesabi;
     writeln('Karenin Kenar uzunluğu  =',uzun,' birim');
     writeln('Alan hesabı işlemi tamamlandı');
end.
```

Örnek: Aşağıda verilen örnekte daire, kare,dörtgen ve üçgen gibi geometrik şekillerin alan hesapları yapılmaktadır.

```
unit Alanlar;
interface
   function Alan  daire(Ycap:real):real;
   function Alan  Kare(uzun:real):real;
   function Alan  Dortgen(uzun,Yuk:real):real;
   function Alan  Ucgen(Taban,Yuk:real):real;

implementation

var My  Pi : real;

procedure Mult  Two  Numbers(Number1, Number2 : real;
                        var Result : real);
begin
   Result := Number1 * Number2;
end;
function Alan  daire(Ycap:real):real;
begin
   Alan  Daire:= sqr(ycap)*My  Pi;
end;

function Alan  Kare(Uzun:real):real;
begin
   Alan  Kare:= uzun*uzun;
end;

function Alan  Dortgen(uzun,Yuk:real):real;
begin
   Alan  Dortgen:= uzun*yuk;
end;

function Alan  Ucgen(Taban,Yuk:real):real;
begin
```

```
  Alan  Ucgen := Taban*Yuk/ 2.0;
end;

begin
  My  Pi := 3.14159267;
end.
```

```
program Sekillerin  alan  hesabi;

uses Alanlar,Crt;

var Ch : char;
    uzun,gen,yuk,Taban,ycap: real;

begin  (* Ana program *)
  repeat
   Writeln;
   Writeln('Lütfen Alan Hesabı Yapacağınız Nesneyi seçiniz :');
     Writeln;
     Writeln('[K]are');
     Writeln('[D]ikdörtgen');
     Writeln('[U]çgen');
     Writeln('d[A]ire');
     Writeln('[C]ılkış');
     Write('Alan hesabı yapılacak nesne (K/D/U/A)?=');
      repeat
       Readln(Ch);
       ch:=upcase(ch);
      until Ch in ['K','D','U','A','C'];
       case Ch of
      'K' : begin
        Write('Karenin kenar uzunluğu =');
        Readln(uzun);
        Writeln('Alan                    =',Alan  kare(uzun):12:4);
        end;

      'D' : begin
        Write('Dörtgenin Genişliği   = ');
        Readln(gen);
        Write('Dörtgenin Yüksekliği            =');
        Readln(yuk);
        Writeln('DörtgeninAlan=,Alan  Dortgen(gen,yuk):12:4);
           end;

  'U' : begin
         Write('Üçgenin Taban Uzunluğu =');
         Readln(Taban);
         Write('Üçgenin Yüksekliği            =');
```

```
                Read(yuk);

          Writeln('Üçgenin Alanı          = ',Alan  Ucgen (Taban,Yuk):12:3);
                end;

            'A' : begin
                Write('Dairenin Çapı              =');
                Readln(Ycap);
                Writeln('Dairenin Çapı    =',Alan  Daire(Ycap):12:3);
                end;

            'C' : begin Writeln('Bitti');exit; end;
            else Writeln(' Tanımsız giriş');
           end;
       until (Ch = 'c') or (Ch = 'C');
    end.  (* Ana program sonu *)
```

Örnek: Aşağıda verilen örnek program 10 tane X değeri için f(x) fonksiyonunu değerlerini hesaplamaktadır. Unit program içinde us alma işlemi gerçekleştirilmektedir. Bu şekilde karmaşık üs alma işlemleri kısaltılmıştır. Kesikli fonksiyonun değeri;

$x < 1 \le 2 \quad f(x) = x^5 - x^{1/3}$

$x < 2 \le 3 \quad f(x) = x^{2/5} - x^{4/3}$

$x > 3 \quad f(x) = x^2 - x^{3/5} - x$

dir.

```
unit usalma;
interface
function us(ust,asil:real):real;
Implementation
function us(ust,asil:real):real;
begin
  us:=exp(ust*ln(asil));
end;
end.
```

```
Program Ana  program;
uses wincrt,usalma;
Var
x:real;
x2,fx:Array [1..10] of Real;
i,x1:Integer;
```

BÖLÜM 14

KAYITLAR

14.1 Giriş

Bir kayıt, bir nesneyle ilgili verilerin bir araya getirilmesidir. Öğrenci kayıtları, taşıt kayıtları, stok kayıtları günlük yaşantımızda karşılaştığımız yaygın kayıtlardır.

Kayıtlar da diziler gibi ögelerden oluşmuşlardır. Ancak aralarında temel bir farklılık vardır. Bir dizi, tümü aynı tipte olan ögelerden oluşmuştur. Bir kaydın ögeleri ise değişik tiplerde olabilir. Örneğin bir öğrenciye ait bilgiler tipik olarak aşağıdaki gibi olabilir.

```
Ad  Soyad      :30 elemanlı karakter dizisi
Cinsiyet       :K/E tek elemanlı karakter
Sınıf          :1-4 arasında tamsayı
Not Ort.       :0-100 arasında gerçel sayı
```

Bu bilgi, ögeleri birbirinden farklı olduğu için dört ögeli tek bir dizide saklanamaz ve üzerinde işlem yapılamaz. Bu tür bilgilerin işlenmesinde ve saklanmasında "RECORD" sözcüğü ile ifade edilen kayıtlar kullanılır. Alan adı verilen ögelerden her biri ayrı bir isme sahiptir.

Kayıt tanımı işlemi aşağıda verildiği şekilde yapılır.

```
Type    kayit  tipi=record
        {Alan listesi}
        end;
```

Alan listesi, ögeler ve ögelerin veri tiplerinden oluşmaktadır. Yukarıda verilen öğrenci bilgileri için aşağıdaki şekilde bir kayıt tipi tanımlanabilir.

```
Type    ogrencikaydi=Record
                Adsoyad:string[30];
                Cinsiyet:char;
                Sinif:byte;
                Not ort:real;
        end;
```

Öğrenci kaydı tipte bir kayıt değişken;

```
Var
OGRENCI: ogrencikaydi;
```

şeklinde tanımlanabilir.

Örnek

```
Type    tarih=record
                gun:1..31;
                ay:(ocak,subat,mart,nisan,mayis,haziran,temmuz,
                    agustos,eylul,ekim,kasim.aralik=;
                yil:integer;
        end;
Var
Dun,bugun,yarin:tarih;
```

Örnek

```
Type    evkaydi=record
                adres:string[40];
                alan:real;
                oda sayisi:integer;
                fiat:real;
        end;
Var
Kiralik ev,satilik ev:evkaydi;
```

Bu kayıtların alanlarından birine erişmek için bir alan belirticisi kullanılır. Alan belirticileri bir kayıt değişkeninin sahip olduğu alanları gösterir. Alan belirticisini tanımlamak için kayıt değişkeninden sonra "." işareti ve bunu takiben alan belirleyicisi yazılır.

```
Alanbelirticisi=Kayıt degisken "."alan belirleyicisi
```

Öğrenci değişkeni için;

```
Ogrenci.Adsoyad
Ogrenci.Cinsiyet
Ogrenci.Sinif
Ogrenci.Notortalamasi
```

alan belirticileri sırasıyla; adsoyad, cinsiyet, sinif ve not ortalaması alanlarını belirtmektedir.

14.2 Kayıt işleme

Pascalda herhangi bir alan belirticisi, aynı tipte bir değişkenin kullanabileceği herhangi bir yerde kullanılabilir. Örneğin; Ogrenci isimli değişkenin üç alanına aşağıdaki program parçası ile değerler atanabilir.

```
for i:=1 to 20 do
        Readln(Ogrenci.adsoyad, Ogrenci.cinsiyet, Ogrenci.notortalamasi);
```

Alan kullanımlarına ilişkin örnekler aşağıda verilmiştir.

Örnekler:

```
if Ogrenci. notortalamasi=50 then

Writeln(hasta.isim);

Writeln(Kiralikev.adres, Kiralikev.fiat);

Case isci.deneyim of
                Usta:isci.saatucreti    :=...
                Kalfa:isci.saatucreti   :=...
                Cirak:isci.saatucreti   :=...
        End;
```

Programda bir kayıt değeri, tüm olarak özdeş tipte başka bir kayda atanabilir. Tarih tipte bildirilen *dun* ve *bugun* isimli değişkenleri ele alırsak;

```
Dun:=bugun;
```

ile bildirilen atama deyimi;

```
Dun.gun:=bugun.gun;
Dun.ay:=bugun.ay;
Dun.yil:=bugun.yil;
```

atamalarına özdeştir. Böyle bir atama daha kısa, açık ve bir çok uygulamalarda etkin olduğu için tercih edilir.

Pascalda bir kayıt, alt programlara parametre olarak transfer edilebilir. Kayıtların hem değer hem de değişken parametre olarak kullanılması mümkündür. Parametre olan kayıtlar, birkaç parametre yerine yalnızca bir parametre transferi yapılacağından parametre listesini kısaltmak açısından yararlıdır. Öğrenci kayıt değişkeninin alanlarına birer değer atamak üzere bir alt program yazacak olursak alt program bildiriminde bir değişken kayıt parametreye ihtiyaç duyulur.

```
Procedure Ogrencikaydioku(var ogrenci:ogrenci  kaydi);
Var
i:integer;
begin
        readln(ogrenci.adsoyad);
     readln(ogrenci.cinsiyet, ogrenci.sinif, ogrenci.notortalamasi);
end;
```

Örnek: Aşağıdaki örnek program, 10 öğrenciden not ortalaması 85 ve üstünde olanöğrencileri onur listesine kaydetmektedir

```
Program Onur  Listesi;
uses wincrt;
type ogrencikaydi=record
       adsoyad:string[30];
       cinsiyet:char;
       sinif:1..4;
       notortalamasi:0..100;
     end;
var
k:byte;
ogrenci:ogrencikaydi;
procedure kayitoku(var ogrenci:ogrencikaydi);
begin
   Gotoxy(15,7);Write('Adı Soyadı          :');
   Gotoxy(15,8);Write('Cinsiyeti           :');
   Gotoxy(15,9);Write('Sinifi              :');
   Gotoxy(15,10);Write('Notortalamasi      :');
   Gotoxy(38,7);readln(ogrenci.adsoyad);
   Gotoxy(38,8);readln(ogrenci.cinsiyet);
   Gotoxy(38,9);readln(ogrenci.sinif);
   Gotoxy(38,10);readln(ogrenci.notortalamasi);
end;
procedure onurlistesi(ogrenci:ogrencikaydi);
begin
      if (ogrenci.notortalamasi >85) then
       begin
            gotoxy(10,15+k); Writeln(Ogrenci.adsoyad:20,'   ',Ogrenci.notortalamasi);
       end;
end;
Begin
    for k:=1 to 10 do
     begin
       kayitoku(ogrenci);
       onurlistesi(ogrenci);
     end;
End.
```

14.3 With Deyimi

Çoğu kez bir program içinde bir kayıt değişkeninin alanlarına erişmek için değişken ismini birkaç kez kullanmak gerekebilir. Onur Listesi programında öğrenci kaydının okunduğu alt programda "ögrenci" isimli değişken birkaç kez kullanılmıştır. Değişken isimlerinin alanlarını nitelemek için değişken isimlerini bir çok kez kullanmak sıkıcı olduğu gibi editör belleğini (64 Kbyte) gereksiz yere işgal edilir. Pascalda alanlar ile ilgili işlemlerde kayıt değişken isminin yinelenmemesi için **With** deyimi kullanılmaktadır. Genel kullanım şekli;

With "Kayıt değişkeni" **do**

şeklindedir.

With deyimiyle öğrenci kaydının okunduğu alt program şu şekilde yazılabilir.

```
procedure kayitoku(var ogrenci:ogrencikaydi);
begin
With ogrenci do
        Begin
                Readln(adsoyad);
                Readln(cinsiyet);
                Readln(sinif);
                Readln(notortalamasi);
        end;
```

With deyimiyle programın kısaltılmasıyla birlikte okunabilirlik de artmaktadır. With deyimi gerektiğinde iç içe de kullanılabilir. Genel olarak;

```
With  Kd1 do                                {(Kd:kayıt değişkeni)}
        With  Kd2 do
                With  Kd3 do
                        .....
                        With  KdN do
```

Şeklinde yazılabilir ve bu yazım şekli;

With Kd1,Kd2, Kd2, Kd3,..., KdN do

İle özdeştir. Burada erişim önceliği en son yazılandan ilk yazılana doğrudur.

14. 4 Hiyerarşik Kayıtlar

Bir kaydın alanı herhangi bir tipte olabilir. Buraya kadar verilen kayıt örneklerinde basit veya dizi tipteki alanlar verilmiştir. Bir kaydın ögesi başka bir kayıt olabilir. Ögelerinin kendiside kayıt olan bir kayda hiyerarşik kayıt adı verilir. Örneğin aşağıdaki yapıya sahip olan bir kaydı, hiyerarşik bir kayıt olarak ifade edebiliriz.

MÜŞTERİ										
Ad Soyad		Adres				Kişisel Bilgiler				
						Doğum Tarihi				
İlk	son	Sokak	Numara	Şehir	Posta kodu	Gün	Ay	Yıl	Cinsiyet	Medeni Durum

```
Adsoyadkaydi=Record
                İlk:string[10];
                Son:string[15];
             End;

Adreskaydi=Record
                Sokak:string[30];
                Numara:integer;
                Sehir: string[20];
                Postakodu:strin[5];
              End.

TarihKaydi=Record
                Gun:1..31;
                Ay:1..12;
                Yil:string[4];
             End;

Kisiselbilgikaydi=Record
                Dogumtarihi:tarihkaydi;
                Cinsiyet:char;
                Medenidurum:char;
             End;

Musterikaydi=Record
                Adsoyad:adsoyadkaydi;
                Adres:Adreskaydi;
                KisiselBilgiler:kisiselbilgikaydi;
             End;
```

Bir hiyerarşik kayıtta bir alana erişmek için, en üst düzeyden alanın bulunduğu düzeye doğru bir yol izlenir. Musterikaydi tipte bir müşteri kayıt değişkeninin her düzeydeki tüm kayıtları aşağıda verilmiştir.

```
Musteri.adsoyad;
Musteri.adsoyad.ilk;
```

```
Musteri.adsoyad.son;
Musteri.adres;
Musteri.adres.sokak;
Musteri.adres.numara;
Musteri.adres.sehir;
Musteri.adres.postakodu;
Musteri.kisiselbilgiler;
Musteri.kisiselbilgiler.dogumtarihi;
Musteri.kisiselbilgiler.dogumtarihi.gun;
Musteri.kisiselbilgiler.dogumtarihi.ay;
Musteri.kisiselbilgiler.dogumtarihi.yil;
Musteri.kisiselbilgiler.cinsiyet;
Musteri.kisiselbilgiler.medenidurum;
```

Görüldüğü gibi hiyerarşik yapılarda alan belirtimi çok uzun olabilmektedir. Müşterinin doğum tarihi ile ilgili bilgiler girilecek ise;

```
Readln(Musteri.kisiselbilgiler.dogumtarihi.gun,
       Musteri.kisiselbilgiler.dogumtarihi.ay,
       Musteri.kisiselbilgiler.dogumtarihi.yil);
```

deyimi yerine with kullanılarak;

```
With musteri do
    With kisiselbilgiler do
        With dogumtarihi do
            Readln(Gun,ay,yil);
```

yazılabilir.

Hiyerarşik kayıtlar kullanılırken alanların açıkça belirtilmesi gerekir.

```
"Musteri.sehir"
```

geçersiz bir alan belirticisidir. Bu alana erişilmek istenildiğinde;

```
"Musteri.adres.sehir"
```

yazılmalıdır.

14.5 Örnek Programlar

Örnek: Bir sınıfta bulunan 20 öğrenci 5 dersten sinava girmişlerdir. Her öğrencinin sinav notlarının ortalamasını bulan Pascal programını yazalım.

```
program ogrenciler;
uses wincrt;
type
   range=1..5;
   range1=1..20;
   ogrencikaydi=Record
                 Numara:array[range1] of string[10];
                 sinav:array[range] of integer;
               end;
var
   ogrenci:ogrencikaydi;
   ortalama:array[range] of real;
   ort  ogrenci:array[range1] of real;
   i,j:integer;
begin
   For  i:=1 to 5 do
        ortalama[i]:=0;
   for i:=1 to 20 do
       begin
           clrscr; ort  ogrenci[i]:=0;
           Write('Ogrenci Numarası    :'); readln(Ogrenci.numara[i]);
           Writeln(Ogrenci.numara[i],' no `lu Ogrencinin sinav notlarını girin');
           for j:=1 to 5 do
             begin
               Write(j,'. sinav notu  =');
               Readln(ogrenci.sinav[j]);
               ort  ogrenci[i]:=ort  ogrenci[i]+ogrenci.sinav[j]
             end;
       end; writeln;writeln;
         for i:=1 to 20 do
           begin
               ort  ogrenci[i]:=ort  ogrenci[i]/5;
               Writeln(Ogrenci.numara[i],' no `lu Ogrencinin ortalaması =',
                              ort  ogrenci[i]:5:1);
           end;
end.
```

Örnek: Bir fabrikada çalışan işçilerin maaşları hesaplanmak istenmektedir. Fabrika işçilerine ödediği saat ücretini, işçinin firmada çalışma süresine göre şu şekilde belirlemiştir.

1..5 yıl arası için 43000 TL/Saat
6..10 yıl arası için 44000 TL/Saat
11..15 yıl arası için 45000 TL/Saat
15..20 yıl arası için 46000 TL/Saat
20..40 yıl arası için 47000 TL/Saat

işçinin brüt maaşı, çalışma saati ile saat ücretinin çarpımıyla elde edilmektedir. Brüt maaştan, %20 vergi, %15 SSK kesintisi, %2 tasarruf kesintisi çıkartılarak net maaş hesaplanmaktadır. İşçinin Brüt maaşı, kesintilerini ve net maaşını hesaplayan program aşağıda verilmiştir.

```
program isci ucretleri;
uses wincrt;
type
   range=1..50;
   iscikaydi=Record
            Adi:string[10];
            Soyadi:string[15];
            sicilno:string[7];
            calisma yili:byte;
            calisma saati:integer;
            saat ucreti:real;
            brut aylik:real;
            net aylik:real;
            vergi:real;
            tasarruf:real;
            ssk kesinti:real;
          end;
var
isci:array [range] of iscikaydi;
i:byte;
saatucreti:real;

function maas(saat ucreti:real):real;
begin
   isci[i].brut aylik:=saat ucreti*10000*isci[i].calisma saati;
   isci[i].vergi:=isci[i].brut aylik*0.20;
   isci[i].ssk kesinti:=isci[i].brut aylik*0.15;
   isci[i].tasarruf:=isci[i].brut aylik*0.02;
   with isci[i] do
     net aylik:=brut aylik-(vergi+ssk kesinti+tasarruf);
end;
```

```
begin
  for i:=1 to 10 do
    begin clrscr;
        Write('İscinin Adı                    =');
        Readln(isci[i].adi);
        Write('İscinin Soyadı                 =');
        Readln(isci[i].soyadi);
        Write('İscinin Sicil Numarası         =');
        Readln(isci[i].sicilno);
        Write('İscinin calisma yılı           =');
        Readln(isci[i].calisma  yili);
        Write('Aylık çalışma süresi           =');
        Readln(isci[i].calisma  saati);

        case isci[i].calisma  yili of
                1..5  :maas(4.3);
               6..10 :maas(4.4);
              11..15:maas(4.5);
              15..20:maas(4.6);
              20..40:maas(4.7);
            end;
    end;
     CLRSCR;
     for i:=1 to 10 do
       begin
        with isci[i] do
         begin
           writeln('ADI SOYADI                  :',adi,' ',Soyadi);
           Writeln('Sicil NUMARASI              :',SicilNo:11);
           Writeln('BRÜT MAAŞ                   :',Brut  Aylik:11:0);
           Writeln('SSK KESİNTİSİ               :',Ssk  kesinti:11:0);
           Writeln('VERGİ                       :',Vergi:11:0);
           Writeln('TASARRUF KESİNTİSİ  :',Tasarruf:11:0);
           Writeln('KESİNTİ TOPLAMI       :',
                    (Ssk  kesinti+VERGI+TASARRUF):11:0);
           Writeln('ELE GEÇEN                :',nET  Aylik:11:0);
           WRITELN;
           WRITELN('Sonraki sonucu gormek için herhangi bir tuşa basınız');
           REPEAT UNTIL KEYPRESSED;
         end;
       end;
end.
```

Örnek: Bir sınıftaki öğrencilerin aldığı dört dersin ağırlıklı ortalamaları hesaplanarak puanları aşağıdaki gibi verilecektir.

```
91..100 :'AA';
81..90  :'BB';
71..80  :'CC';
61..70  :'DD';
60>     :'FF';
```

Dört dersin ağırlıkları sırasıyla; %15, %25, %30 ve %30 'dur. İlgili program aşağıda verilmiştir.

```
Program ogrenci  kayitlari;
uses wincrt;
type
   isim=record
      ad:string[10];
      soyad:string[15];
    end;
    sayi=1..4;
    ogrencikaydi=record
      adi:isim;
      numarasi:string[10];
      sinavi:array[sayi] of byte;
      notortalamasi:real;
      yazi:char;
     end;
var
  ogrenci:array [1..40] of ogrencikaydi;
  N,k,i:byte;
  top:integer;
  ort:integer;
  ad:string[25];
procedure veriler;
    begin
    Write('Öğrenci sayısı  =');
    Readln(N);clrscr;
    for i:=1 to N do
     begin
       with ogrenci[i] do
         begin
          Write('Adı            :');
          readln(adi.ad);
          Write('Soyadı         :');
          readln(adi.soyad);
          Write('Numarası       :');
          readln(numarasi);
```

```
            for k:=1 to 4 do
            with ogrenci[i] do
              begin
                write(adi.ad+' '+adi.soyad,' isimli öğrencinin ',k,'. notu =');
                Readln(sinavi[k]);
                top:=top+sinavi[k];
              end;
        end;
       end;
end;

procedure hesaplar;
begin
    for i:=1 to N do
       begin
         with ogrenci[i]do
           begin
      notortalamasi:=0.15*sinavi[1]+0.25*sinavi[2]+0.3*sinavi[3]+0.3*sinavi[4];
                ort:=trunc(round(ogrenci[i].notortalamasi));
                case ort of
                     91..100:yazi:='AA';
                      81..90:yazi:='BB';
                      71..80:yazi:='CC';
                      61..70:yazi:='DD';
                     else      yazi:='FF';
                  end;
           end;
        end;
end;

procedure sonuclar;
begin
     Writeln('Adı Soyadı                  Ortalama   Derece');
     for i:=1 to N do
          with ogrenci[i] do
            begin
              ad:=adi.ad+' '+adi.soyad+'                          ';
              ad:=copy(ad,1,25);
              Writeln(ad,'  ',notortalamasi:6:1,'      ',yazi);
            end;
end;

BEGIN
     veriler;
     hesaplar;
     sonuclar;
END.
```

BÖLÜM 15

TURBO PASCALDA BELLEK YÖNETİMİ

15.1 Giriş

Bir bilgisayar programcısı için en önemli bellek bileşeni RAM (Random Access Memory) dir. Bir program çalıştırıldığı zaman program komutlarını depolamak için bir veya bir kaç bölüt (segment), program bölütü olarak ayrılır (Code segment). Yine program için bir veri bölütü (data segment) ve geçici bellek olarak kullanılmak üzere yığıt bölütü (stack segment) ayrılır. Bölütlenmiş adres, bölüt ve uzanım (offset) olmak üzere iki parçadan oluşur. 80*86 ailesi, 8086 gerçek konumu ve DOS işletim sistemi koşullarında 640 Kb RAM bellek ile sınırlanmıştır. 80286 ve sonrası bilgisayarlarda korumalı konuma geçilmesi durumunda (protected mode) bellek kısıtlaması ortadan kalkar ve varolan RAM belleğin tamamı adreslenebilir.

Pascal PC belleğini 4 ayrı bölüt olarak değerlendirir.

1. Program bölütü (Code segment)
2. Veri bölütü (Data segment)
3. Yığıt bölütü(Stack segment)
4. Yığın (Heap)

Pascal programlama dilinde ana program ve her unit için bir program bölütü ayrılır. Çok büyük programlar veya unitler ile çalışırken "program bölütü çok büyük" (Code segment too large" hatasının nedeni budur. Bu hata ile karşılaştığımızda ana programı veya unitleri daha küçük unitlere bölerek sorun çözülür.

Diğer taraftan programın global değişkenleri ve tiplendirilmiş sabitleri veri bölütü olarak adlandırılan bellek parçasında depolanırlar ve bu bölüntünün de maksimum büyüklüğü 64 Kb tır. (65520 byte) 'tir. Veri bölütünün yetersiz kalması durumunda "veri bölütü çok büyük" (Data segement too large) hatasını verecektir. Bu hatanın aşılmasını yolu, global değişkenlerin bir bölümünü dinamik değişkenler olarak düzenlemektir.

Verilen depolanabileceği tek yer veri bölütü değildir. Pascal bellek kullanım düzeninde yığıt (stack) ve yığın (heap) önemli yer tutarlar. Yığıt denetimi, Pascal tarafından gerçekleştirilir ve programcının konuyla ilgilenmesine gerek yoktur. Diğer yandan

dinamik bellek kullanımının ham maddesi olan yığın hemen hemen tümüyle programcıya bırakılmıştır.

Procedure ve function alt programlarında tanımlanan yerel değişkenler yığıt bölütü adı verilen dinamik bellekte depolanırlar. Alt program çalıştığında, yerel değişkenler için yığıtta yer ayrılır. Alt programın çalışması sona erince yığıttaki değişkenler yok edilir.

15.2 Segment, Offset ve Heap

Bilgisayar belleğinin her noktası bir segment ve offset adresi ile ifade edilir. Segmentlerin başlangıç adresleri 16 ile bölünebilen tam sayıdır.

SEG ve OFS deyimleri Pascalın adreslerle işlem yapmak için kullanılan iki standart komutudur.

15.2.1 SEG (D)

D değişkeninin içinde bulunduğu bölütün adresini tam sayı olarak verir. Üretilen değer, D değişkenini bulunduran gerçek adres değeri değildir. Gerçek adres değerini bulmak için bu sayının 16 ile çarpılması gerekir.

15.2.2 OFS (D)

D değişkeninin offset adresini tam sayı olarak verir. Üretilen değer değerin gerçek adresi değildir. Gerçek adresin bulunması için bu değerin D değişkenini bulunduran segment adresine ilave edilmesi gerekir.

Örnek:

```
uses wincrt;
var
ad:string;
s,o:integer;
R:real;
begin
r:=0;
S:=Seg(ad);
o:=OFS(ad);
R:=16.0*s+o;
Writeln ('Ad değişkeninin Segment adresi  =',S);
Writeln ('Ad değişkeninin Offset adresi   =',o);
Writeln ('Ad değişkeninin gerçek adresi   =',r:15:1);
end.
```

15.2.3 HEAP (Yığın)

Pascal, programı çalıştırmak üzere belleğe yüklemeden önce belleği kontrol eder. Bellekte bulduğu ilk boş alandan itibaren Kod, veri ve yığıt bölütleri için yer ayırır.

Heap, programdaki kod, veri ve yığıt tararından kullanılamayan, işletim sisteminde kullanılabilir RAM 'in tamamıdır. {$M+} derleme direktifini kullanarak kullanılabilir heap boşluğu miktarını kontrol edebiliriz.

15.3 Statik Değişkenler

Programcının tanımladığı bütün basit değişkenlerin statik değişken olduğu söylenebilir. Bu tür değişkenlerin, bellekte kaplayacakları alan uzunlukları ve bellekteki yeri program boyunca sabittir. Programın herhangi bir noktasında yeri değiştirilemez.

15.4 Dinamik Değişkenler

Çalışma sırasında formu ve uzunluğu değiştirilebilen değişkenlere ihtiyaç duyulur. Bu tür değişkenlere dinamik değişken adı verilir. Dinamik değişkenlerin ayrı bir özelliği dinamik değişken için ayrılan alan gerektiğinde yok edilebilir. Dinamik değişkenler için ayrılan alanlar heap üzerindedir.

Dinamik değişkenler birer değişken ismiyle direkt olarak temsil edilmezler. Bunun için dinamik değişkenin bellekteki adresini içeren özel bir değişken kullanılır. Bu özel değişken pointer değişkendir.

15.5 Pointer Değişkenler

Dinamik değişkenlerin bellekteki adresini gösteren değişkenlere pointer değişken denir. Bir pointer, programımızdaki bir veriye veya koda referanstır ve verinin bellekte yerleşeceği adresi belirtir.

Pointer kullanımıyla daha büyük ve daha esnek yazılması mümkün olmaktadır. Program içinde pointer kullanımının nedenlerini aşağıdaki gibi özetleyebiliriz.

Program 64 KByte 'tan fazla veri içeriyorsa
Derleme sırasında veri boyutu bilinmiyorsa
Programımız çoklu veri tipleri içeriyorsa
Program record ve object bağlantı listesi kullanıyorsa pointer kullanırız.

Bu nedenleri kısaca inceleyecek olursak;

Programlar, Borland Pascal 'ın ayırmış olduğu 64 KB veri sınırını aşarsa verilerin bir kısmı kaybolacak ve yapılmak istenilen işlem başarısızlıkla sonuçlanacaktır. Global

değişkenler tanımlanırken derleyici, değişkenler için kod bölütü (data segment) olarak tanımlanan yer ayırır. Kod bölütünün maksimum boyutu 64 KB 'tır. Bunun taşıdığı anlam, program için kullanılabilen global değişkenlerin tümünün toplamı sadece 64 KB olabilir. Bir çok program için bu limit sorun oluşturmayabilir, ancak bazen daha büyük veri kullanmak gerekebilir.

Örneğin Programınızda;

```
A:Array[1..400] of string [100];
```

şeklinde tanımlanması gereken bir dizi olsun. Bu dizinin programda işgal edeceği alan kabaca 40 KB 'tır. Bu değer 64 KB 'tan daha küçüktür. Diğer bütün değişkenlerin toplamı 24 KB 'tan küçük ise böyle bir dizi tanımı halinde program için bir sorun oluşmaz.

Fakat, bazen A dizisi gibi iki dizi kullanılması gerekebilir. Bu durumda gerekli data 80 KB olacaktır ki Pascal 64 KB sınırının aşılmasına izin vermez ve "too many variables (çok fazla değişken var)" şeklinde bir hata mesajı ile programın çalışmasını keser.

Daha büyük veri ile çalışmak için heap kullanılması gerekir. Heap üzerinde, program 80 KB ile çalışmaya izin verir. Pointer, veri bölütünde 4 byte 'lik bir yer kaplar ve bunu 2 byte 'i segment diğer 2 byte 'i ise offset adresini oluşturur.

Bilinmeyen Boyutta Veri ile Çalışma durumunda ise; Pascalda dizi ve string elemanlarla çalışırken elemanın alacağı maksimum değer bilinmiyorsa genelde ihtiyaçtan daha fazla miktarda boyut açarak sorun giderilmeye çalışılır. Örneğin;

```
A:array[1..5000] of byte;
```

şeklinde bir dizi tanımlamış olalım. Program içinde A dizisi 100 eleman kullanıyorsa geriye kalan 4900 dizi elemanı bellekte "0" olarak tanımlanacak ve gereksiz yere veri bölütünü işgal edecektir. Bu nedenle böyle uygulamalarda pointer kullanımı gereklidir.

15.5.1 Pointer Değişkenlerin Tanımlanması (Göstergeler)

Pointer değişkenlerin tanım cümlesinde, tip bildirisi olarak yer alacak isimler TYPE bölümünde önceden oluşturulmalıdır. TYPE bölümünde kullanılan tanım cümlesi, oluşturulmak istenilen pointer tipini gösteren bir isim ve bu tiple gösterilecek dinamik değişkenleri içeren bir tip bildirisinden oluşur. Tip bildirisinin önüne "**^**" karakteri ön ek olarak yazılır. Tanımlamada kullanılan genel form;

```
Type
        Tipismi=^Tipbildirisi
```

şeklindedir.

Pointer değişkenlerin tanımlanması sırasında, VAR bölümünde kullanılacak tip bildirisi TYPE bölümünde pointer olarak oluşturulmuş bir tip ismi veya gelişigüzel tip bildirisi olarak seçilebilir. VAR bölümünde kullanılan tip bildirisi pointer tipine sahip değilse ilk

karakter olarak **^** ön eki kullanılmalıdır. Aşağıda verilen birinci örnekte satir isimli pointer değişken, bir pointer ismiyle, indis isimli değişken ise standart tip bildirisi ile tanımlanmıştır.

Örnek:

```
Type
Goster=^Kayit;
        Kayit=record;
                Ad,soy:string[10];
             End;
VAR
        SATIR:GOSTER;
        BILGI:STRING[50];
        INDIS:^INTEGER;
```

Örnek:

```
Type
        Kitapgos=^Kitaptipi;
        Kitaptipi=Record;
                Adi:string[40];
                Yayinevi:string[20];
                Basimyili:word;
                Sonraki:kitapgos;
            End;
Var
        Kitap:Kitapgos;
```

Bu örnekte, kitapgos isimli pointer kitaptipi tipinde bir kayda işaret etmektedir. Burada Pascalın kuralcılığını bozduğu nadir örneklerden biri görülmektedir. Kitapgos tipinin tanımlanması sırasında Kitaptipi tipinden yararlanılmıştır. Bu durum Pascal için olağandışı durumlardan biridir.

15.5.2 Pointer Değişkenlerin Kullanılması

Pointer tipi verileri bir basit değer ancak kullanılmalarına izin verildiği formlar altında bir function veya procedure alt programların parametresi olarak yer alabilirler. Pascalda 0 pointer değerine karşılık gelen özel bir değer vardır. Bu değer **NIL** olarak bilinir. Pointer değişkenlere sabit değerler atamak için **PTR** fonksiyonundan yararlanılır.

a-NIL

Bir pointer değişkenin kapsadığı 4 Byte 'lik bir alan sadece sıfır değerinde oluşuyorsa bu pointer değişkenin Nil değerine sahip olduğu söylenir. Özel bir pointer değerine sahip olan Nil;

```
Pointer:=Nil;
IF  pointer<>nil then
```

şeklinde pointer değişkenler için atama ve karşılaştırma cümlelerinde kullanılabilir.

Nil değerine sahip olan bir pointer değişken, o anda temsil edilen bir dinamik değişken olmadığını, diğer bir deyişle bu değişken tarafından temsil edilen bir adresin bulunmadığını gösterir.

b-PTR(X,X)

Bu fonksiyon, iki tamsayı değeri parametre olarak kullanıp bir pointer değer üretir. Parametrelerden ilki segment, ikincisi offset adresidir.

Pointer değişkenlerin temsil ettiği değerler iki farklı şekilde incelenir. Bu değerlerden birincisi, değişken için ayrılan alanda bulunan segment ve offset, ikincisi ise bu adres ile gösterilen alandaki değerdir. İlk değeri ifade etmek için pointer değişken kendi başına kullanılır. İkinci değeri ifade etmek için sağ tarafına **^** son ekinin getirilmesi gerekir.

Bir pointer değişkenin içeriği sayısal bir değer olarak incelenecek ise Seg ve Ofs fonksiyonları kullanılmalıdır. Değişkenin bulunduğu adres ile gösterdiği adres birbirine karıştırılmamalıdır.

15.5.3 Heap Üzerinde Alan Kullanımı

Tanım bloklarının icrasından sonra sadece statik değişkenler için yer ayrılır. Bu değişkenlere pointer tipi değişkenler dahildir. Tanım bloğu icra edilirken tanımlarda yer alan dinamik değişkenler için bellekte yer ayrılmaz. Dinamik değişkenlere heap üzerinde ayrılacak yerler icra bölümünde tahsis edilir. Bu amaçla New procedür alt programı kullanılır.

15.5.4 New-Dispose

Tanımlanan pointerler için New deyimi ile heap' te yer ayrılır. Dispose deyimi ile pointer değişkene ayrılan yerler silinerek iptal edilir. Program içinde kullanılan her new deyimi için bir dispose deyimi kullanılmalıdır.

Örnek:

```
uses wincrt;
type
   str18=string[18];
var
   p:^str18;
begin
  new(p);
  P^:='İyi çalışmalar....';
  Writeln(P^);
  dispose(P);
end.
```

15.5.5 Mark ve Release

New ve dispose deyimleri yerine dinamik belleklerde kullanılır. Bu deyimlerden mark ile hepa segmentin en üst bölümünde aktif pointer değişkeni için yer açar ve heap 'in devamını boş bırakarak diğer pointerlerce kullanılmasını sağlar.

15.5.6 Getmem-FreeMem

Dinamik bellek kullanımının üçüncü yoludur. Getmem ve Freemem de New ve Dispose gibi programın çalışması süresince Heap bellekte yer açar ve açılan yeri boşaltırlar. Getmem deyimi ile değişken tipine bağlı olarak gerekli bellek ayrılmasını sağlar. Örneğin; değişken için 100 Byte 'lik bir yer açılması gerekiyorsa;

```
Getmem(degisken,100);
```

olarak belirtilir. Freemem ile ayrılan bu alan boşaltılır.

```
Freemem(degisken,100);
```

Örnek:

```
type
   kitapgos=^Kitaptipi;
   Kitaptipi=record
            adi:string[30];
            yazari:string[30];
            yayinevi:string[20];
            basimyili:word;
            sonraki:kitapgos;
          end;
var
  kitap:kitapgos;
```

Yığında kitap tipinde bir alan yaratılması için;

```
New(kitap);
```

veya

```
Getmem(kitap,sizeof(kitaptipi));
```

komutlarından birinin verilmesi gerekir.

Belleği iyi kullanmak için bağlı listeler kullanılması gerekir. Bağlı liste, bir boncuk dizisi gibi birbiriyle ilişkili elemanlar topluluğudur. Topluluğu oluşturan elemanlar , birbirini işaret etmek üzere bir düzen oluştururlar. Tek bağlı listelerde, listedeki her eleman kendisinden sonra gelen elemana işaret eder. Yukarıda verilen tip bildiriminde **sonraki** isimli alan bu amaçla geliştirilmiştir.

Tek bağlı listenin geliştirilmesi üç tane göstergeyi gerektirir. İlk gösterge, önceki kayda işaret etmek üzere kullanılır. İkinci gösterge o an işlenmekte olan kaydı gösterir. Diğer bir gösterge ise liste başını işaret eder. Yukarıdaki örnek üzerinde çalışmaya devam ettiğimizde var bloğu aşağıdaki şekilde yazılabilir;

```
Var
        listebasi,oncekikitap,islenenkitap:kitapgos;
```

Eğer listeyi baştan sona kadar işlemek istersek, listedeki ilk kaydı bilmemiz gerekir. Islenenkitap 'a göre önceki kayda işaret eden göstergenin ismi oncekikitap 'tir.

Listenin üretilmesi işlemine ilk kayıt göstergesinin (listebasi) sıfırlanmasıyla başlanır.

```
Listebasi:=nil;
```

Listenin devamının üretilmesi için önce liste başının nil olup olmadığı kontrol edilir. Daha sonraki işlemler, aşağıdaki örnekte verilmiştir.

```
if  listebasi=nil then
        begin
                New(islenenkitap);
                Listebasi:=islenenkitap;
                islenenkitap^.sonraki:=nil;
        end
        else
        begin
                oncekikitap:=islenenkitap;
                New(islenenkitap);
                Oncekikitap^.sonraki:=islenenkitap;
                islenenkitap^.sonraki:=nil;
        end;
```

Örnekte önce liste başının NIL 'e eşit olup olmadığı kontrol edilmektedir. Eğer listebasi nil 'e eşit ise yeni bir kayıt oluşturulup adresi, Islenenkitap isimli göstergeye konmakta ve Listebasi bu göstergeye eşitlenmektedir. Bu yeni kayıttan başka kayıt bulunmadığı için kaydın sonraki isimli alanına nil değeri atanmaktadır.

İlk kaydın listeye eklenmesiyle artık Listebasi, nil 'e eşit olmadığından IF-THEN-ELSE grubunun ELSE bölümüyle devam etmek gereklidir. ELSE bölümünde yeni bir kayıt oluşturulmadan önce varolan kayıt (islenenkitap) önceki kayıt haline getirilmektedir. Daha sonra yeni bir kayıt oluşturulup önceki kaydın bu kaydı işaret etmesi sağlanmalıdır. Yeni kayıttan sonra başka kayıt olmadığından yeni kaydın sonraki isimli alanına nil değeri atanmaktadır.

Örnek:

```
Program Kitap  liste;

        {$M 16384,32768}
        program kitap  listesi;
        uses wincrt;
        type
            Kitapgos=^Kitaptipi;
            KitapTipi=Record
                        Adi:string[20];
                        Yazar:string[25];
                        Yayinevi:string[15];
                        basimyili:word;
                        sonraki:kitapgos;
                      end;
        var
        listebasi,oncekikitap,islenenkitap:kitapgos;
        c:char;
        Procedure kitapgir;
        begin
          with islenenkitap^ do
            begin
              write('Kitap Adı  :');
              readln(adi);
              write('Yazarı     :');
              readln(yazar);
              write('Yayınevi   :');
              readln(yayinevi);
              write('Basım Yılı :');
              readln(Basimyili);
            end;
          end;
        Procedure Kayitekle;
        begin
```

```
        clrscr;
        if listebasi=nil then
          begin
            New(islenenkitap);
            Kitapgir;
            Listebasi:=islenenkitap;
            islenenkitap^.sonraki:=nil;
          end
          else
          begin
            Oncekikitap:=islenenkitap;
            New(islenenkitap);
            Kitapgir;
            Oncekikitap^.Sonraki:=islenenkitap;
            islenenkitap^.sonraki:=nil;
          end;
        end;
      Procedure Kayitlistele;
      var
      kitap:kitapgos;
      begin
        kitap:=listebasi;
        while kitap<>nil do
          begin
            with kitap^ do
              begin
                Adi:=adi+'                                    ';
                Adi:=copy(adi,1,20);
                Yazar:=yazar+'                                    ';
                Yazar:=copy(yazar,1,25);
                Yayinevi:=yayinevi+'                                        ';
                Yayinevi:=copy(Yayinevi,1,15);
                Writeln(Adi,'  ',Yazar,'  ',Yayinevi,'  ',basimyili);
              end;
             Kitap:=Kitap^.sonraki;
          end;
          Writeln('Devan için bir tuşa basınız');
          Repeat Until keypressed;
      end;
      begin
       listebasi:=nil;
       repeat
         clrscr;
        REPEAT
         Writeln('"E"kle');
         Writeln('"L"istele');
         Writeln('"S"on');
         Writeln;Write('Yapmak istediğiniz işlemi seçiniz  ..');
         Readln(C);c:=upcase(C);
```

```
    UNTIL C in['E','L','S'];
      case C of
        'E':kayitekle;
        'L':KayitListele;
        ELSE;
      end;
    until c='S';
end.
```

Örnek:

Pointer tipi değişkenler kullanarak bir sınıftaki öğrencilere ait aşağıda verilen bilgi tipleri girilecektir.

Ad Soyad,
Vize Notu
Final notu

Bu bilgilerden yaralanarak öğrencinin ortalaması;

ortalama:=0.4*vize+0.6*final;

şeklinde hesaplatılacaktır. Hesaplama sonuçlarına göre aşağıda verildiği şekilde çıktı oluşturulacaktır.

Ad Soyad	Vize	Final	Ortalama	Durum
**********	****	****	********	******

Notun 49.5 ile 50 arasında olması halinde ortalama not 50 'e yuvarlatılacaktır. Geçme şartının sağlayan öğrencilerin durumu "geçti", şartı sağlayamayan öğrencinin durumu "kaldı" şeklinde bildirilecektir.

```
program sinav  sonuclari;
type
   goster=^ogr;
   ogr=record
   adsoy:string[25];
   vize:byte;
   final:byte;
   ort:integer;
   durum:string[5];
   sonadr:goster;
  end;
var
```

```
    yenikayit,eskikayit,ilkkayit:goster;
    yanit:char;
    tepe:^integer;
    i,n:byte;
begin
i:=(0);
mark(tepe);
new(eskikayit);
ilkkayit:=eskikayit;
Write('Öğrenci Sayısını Giriniz   =');
Readln(n);
     Repeat
       i:=i+1;
       New(yenikayit);eskikayit^.sonadr:=yenikayit;
       Write('Adı Soyadı            =');
       readln(Yenikayit^.adsoy);
       Write('Vize Notu             =');
       readln(Yenikayit^.vize);
       Write('Final Notu            =');
       readln(Yenikayit^.final);
       yenikayit^.ort:=trunc(round(0.4*yenikayit^.vize+0.6*yenikayit^.final));
           if (yenikayit^.ort>=50) and (yenikayit^.final>=50) then
                                       yenikayit^.durum:='GEÇTİ'
                                       else
                                       yenikayit^.durum:='KALDI';
           eskikayit:=yenikayit;
           writeln;
     Until i=n;
     eskikayit^.sonadr:=nil;

while ilkkayit^.sonadr<>nil do
          begin
            ilkkayit:=ilkkayit^.sonadr;
           ilkkayit^.adsoy:=ilkkayit^.adsoy+'                    ';
           ilkkayit^.adsoy:=copy(ilkkayit^.adsoy,1,25);
             writeln(ilkkayit^.adsoy,'   ',ilkkayit^.vize,'   ',
             ilkkayit^.final,'   ',ilkkayit^.durum);
       end;
     release(tepe);
end.
```

BÖLÜM 16

PASCAL ile GRAFİK

Günümüzde grafiklere olan ihtiyaç, başta mühendislik, fen bilimleri ve ticari çalışmalar olmak üzere bir çok alanda artmaktadır. Günümüzün gelişmiş Pentium tabanlı PC 'leri bu ihtiyaca oldukça hızlı bir şekilde cevap verebilmektedir. Kişi, grafik çalışmalarını genel olarak paket programlar aracılığı ile gidermektedir. Ancak bazı özel durumlarda paket programlarla bu ihtiyacın giderilmesi mümkün olmayabilir. Bu durumda yapılması gereken işlem, kişinin özel amacına yönelik grafik programı yazılmasıdır.

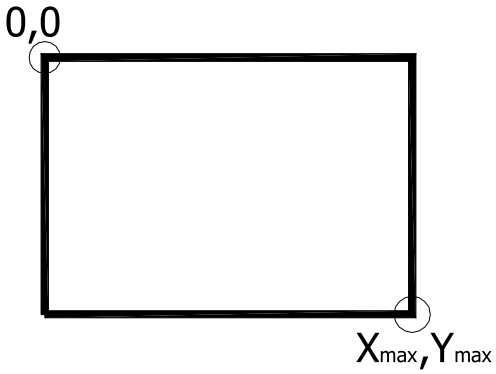
Borland firması 1985 'te Turbo Pascal 3.0 sürümü için hazırlamış olduğu Turbo Graphix Toolbox grafik paketini programcılara sunmuştur. Pakette bulunan grafik komutlarının zenginliği Turbo Pascal 'ın grafik çalışmalarında üst düzeye çıkmasını sağlamıştır.

Günümüzde Borland Pascal 7.0, Borland Delphi, Visual Basic, Visual C++ gibi program geliştirme setlerinin oldukça güçlü grafik paketleri vardır.

16.1 Grafik Ekranı

Grafik ekranı, ekranın yatayda ve düşeyde eşit aralıklar bölünmesinden elde edilen matris elemanlarından oluşur. Bu matris elemanları, grafiği oluşturan en küçük parçacıklardır. Bilgisayarda bu parçaların her birine piksel (pixel) denir. Ekrandaki her pikselin yatay ve düşey olmak üzere bir koordinat numarası vardır. Ekrandaki grafik çizimleri bu koordinat numarasından yararlanılarak gerçekleştirilir.

Borland grafik sisteminde ekranın sol üst köşesi, piksel matris alanının başlangıç noktasıdır. X koordinatı soldan sağa doğru, Y koordinatı yukarıdan aşağıya doğru artar. Bu ifadeye göre bir grafik ekranın koordinat sistemi aşağıda verildiği gibidir.



Burada X_{max} ve Y_{max} grafik ekranın aldığı maksimum X ve Y koordinatlarıdır.

Grafik sistemi başlatıldığında fiziksel ekranın sol üst köşesi orijin (0,0) olmak üzere x ekseni sağa doğru, y ekseni de aşağıya doğrudur. Fiziksel ekran koordinat sisteminde Y ekseninin aşağıya doğru oluşu alışılagelmiş olduğumuz kartezyen koordinat sistemine göre zorluklar yaratır. Grafik uygulamalarında X sağa doğru Y yukarıya doğru olduğunda bu zorluklar aşılabilir. Bunun için yapılması gereken işlem, orijin noktasının sol alt köşeye alınmasıyla birlikte çizimde kullanılan y koordinatlarını -1 ile çarparak negatiflerinin kullanılmasıdır. Orijin Setviewport komutunun kullanımıyla ekran üzerinde istenilen konuma alınabilir.

16.2 DOS için Pascal Grafiği

Borland grafik paketi, Borland firmasının Turbo C, Turbo Pascal ve Turbo Prolog programlama dilleri için hazırladığı hızlı, büyük, gelişmiş ve grafik adaptöründen bağımsız çalışabilen komutları içeren bir yazılım olup iki kısımdan meydan gelmiştir.

* Grafik komut kütüphanesi
* Grafik arabirimi

Grafik komut kütüphanesi, grafikle ilgili tüm komutları içerir. Bu kütüphanedeki komutlar grafik adaptörü tipinden bağımsız olarak ayarlanmıştır. Grafik arabirimi ise, grafik programının bilgisayarda bulunan grafik adaptörüyle iletişimi sağlar. Grafik kütüphanesindeki tüm komutların uygun şekilde çalışması bu arabirimin kullanılmasıyla gerçekleşir. Grafik arabirimi, çizime başlamadan önce grafik kütüphansesindeki komutlar yardımıyla belleğe yüklenir.

Grafik arabirimi, her adaptörü için ayrı ayrı oluşturulmuş ve manyetik diske "BGI' uzantılı olarak yüklenmiştir. Bunlara kısaca, Borland Grafik Arabirimi (Borland Graphics Interface-BGI) dosyalar denir.

Bir grafik programının derlenip çalıştırılabilmesi için Pascal programlama dilinde aşağıda belirtilen dosyalara ihtiyaç duyulur. Bu dosyalar;

1. Grafik Komut Kütüphanesi

   * GRAPH.TPU komut kütüphanesi (Gerçek mod - Real mode)
   * GRAPH.TPP komut kütüphanesi (Korumalı mod - Protected mode)

2. Bilgisayarda bulunan grafik adaptörüne ait bilgilerin bulunduğu aşağıda ilgili BGI dosyalarından biri,

   ATT.BGI
   CGA.BGI
   EGAVGA.BGI
   HERC.BGI
   IBM8514.BGI
   PC3270.BGI
   VESA16.BGI

3. Eğer grafik ekranda bir metin yazımında çeşitli fontlar kullanılacaksa, kullanılacak font tiplerine ait karakter font dosyaları. Bu dosyalar CHR uzantılıdır.

   TRIP.CHR
   LITT.CHR
   SANS.CHR
   SIMP.CHR
   TSCR.CHR
   LCOM.CHR
   EURO.CHR
   BOLD.CHR

Derleyiciye program içinde grafik komutlarının bulunduğu USES komut satırında aşağıdaki komut ile bildirilir.

```
USES GRAPH;
```

16.2.1 DetectGraph

Bu alt program, bilgisayarda kullanılan grafik uyarlayıcıya ait grafik sürücünün ve grafik konumunun otomatik olarak belirlenmesini sağlar. Bu alt programın çalıştırılması için;

```
DetectGraph (Grafiksurucu,Grafikkonumu);
```

Satırının yazılması gerekir.

Eğer grafik uyarlayıcı hakkındaki bilginin yazılım tarafından belirlenmesi istenirse DetectGraph komutu kullanılarak bu bilgiler elde edilir. Komutun kullanımı;

```
DetectGraph (Grafiksurucu,Grafikkonumu);
```

şeklindedir. Bu komut ile grafik uyarlayıcı hakkında bilgi elde edildikten sonra Initgraph komutu uygulanır.

Grafiksurucu ve grafikkonumu programın VAR bölümünde tamsayı (örneğin:integer) tipte tanıtılır. Komutun bu şekilde kullanılmasında tanımlanan bu değişkenlere grafik sürücüyle ilgili bilgiler atanabilir. Örneğin; VGA uyumlu bir grafik uyarlayıcıda grafik ortamının başlatılması için gerekli komutlar aşağıda verilmiştir.

```
Grafiksurucu:=VGA;
Grafikkonumu:=VGAHi;
```

Grafik ortamının başlatılması için ise ***Initgraph*** komutu kullanılır.

16.2.2 Initgraph

Grafik sürücü ve grafik konumu ile tanımlanan grafik ortamını başlatır. Grafik ortamın başlatılması için detectgraph alt programının çalıştırılmasından komutundan sonra;

```
Initgraph (Grafiksurucu,Grafikkonumu,'sürücü yolu');
```

Grafik sürücü yerine sıfır verilirse alt program otomatik olarak grafik uyarlayıcıyı tanır ve grafik konumunu en yüksek ayrıntıya ayarlar. Burada kullanılan sürücü yolu; bilgisayarda BGI dizininin bulunabileceği yoldur.

Initgraph ile kullanılan parametreler grafik sürücünün ve grafik uyarlayıcı belirlenmesi için kullanılmaktadır. **"C:\bp\bgi"** ise, grafik sürücülerinin bulunduğu klasörü belirtmektedir.

Daha önce de belirtildiği gibi grafik ekranın kooordinatları 0,0 ile başlar. Hercules standardındaki ekranda sol alt köşenin koordinatı (0,347), sağ üst köşenin koordinatı (719,0), sağ alt köşenin koordinatı ise (719, 347) dir. 640*480 piksel matris alanına sahip VGA (VGAHi) konumunda, koordinatlar yukarıda belirtildiği sırayla (0,0), (0,479), (639,0) ve 639,479) dur. Kullandığınız ekranın köşe ve merkez koordinatlarını sergileyen örnek program aşağıda verilmiştir. Bu programda kullanılan komutlar daha sonra tanımlanacaktır.

Örnek:

```
uses graph;
var
x,y,renk,grafiksurucu,grafikkonumu, hatakodu:integer;
Hata:Boolean;
Function Kataryap(x,y:integer):string;
var
```

```
sx,sy:string[5];
begin
  str(x,sx);
  str(y,sy);
  Kataryap:='('+ Sx+','+Sy+')';
end;
begin
detectgraph(grafiksurucu,grafikkonumu);
initgraph(grafiksurucu,grafikkonumu,'c:\bp\bgi');
hatakodu:=Graphresult;
hata:=hatakodu<>grOK;
if hata then
  begin
     Writeln('Grafik konumu hatası  :',Grapherrormsg(hatakodu));
  end;
  moveto(0,0);
  outtext(kataryap(0,0));
  moveto(0,getmaxY-10);
  outtext(kataryap(0,getmaxY));
  moveto(getmaxX-80,0);
  outtext(kataryap(getmaxX,0));
  moveto(getmaxX-80,getmaxY-10);
  outtext(kataryap(getmaxX,GetmaxY));
  moveto(Trunc(getmaxX/2),Trunc(GetmaxY/2));
  outtext(kataryap(Trunc(getmaxX/2),Trunc(GetmaxY/2)));
  Renk:=Blue;
  X:=0;
  Y:=X;
  repeat
     putpixel(x,y,renk);
   inc(x);inc(y);
  until (X>GetmaxX) or (Y>GetmaxY);
readln;
closegraph;
end.
```

Yukarıda verilen örnekte ekranın maksimum ve minimum noktaları GetmaxX ve GetmaxY komutları ile belirlenmektedir.

16.2.3 ClearDevice

Grafik ekranı temizlemek amacıyla kullanılır. Komutun aktif hale getirilmesi için Cleardevice yazmak yeterlidir. 80*25 yazı ekranındaki CLRSCR komutuna benzer. Kullanımı;

```
Cleardevice;
```

16.2.4 Closegraph

Grafik ortamı kapatarak ekranı grafik ortama geçmeden önceki konumuna getirir ve grafik sistemi tarafından kullanılan belleği serbest duruma getirir. Grafik ortamdan çıkışta kesinlikle kullanılmalıdır.

16.2.5 GetmaxX

Kullanılan grafik konumu için yatay koordinatın maksimum değerini getirir.

16.2.6 GetmaxY

Kullanılan grafik konumu için düşey koordinatın maksimum değerini getirir.

16.2.7 GraphErrorMsg

Grafik ortamında oluşan hataya ilişkin bir rapor verir. Kullanım şekli;

```
GraphErrorMsg(GrafikHata);
```

Burada kullanılan Grafikhata isimli değişken integer tipte bir değişken olup GraphErrorMsg altprogramı çalıştığında hata söz konusu olduğunda bir değer alır. Eğer bir grafik hatası söz konusu değilse Grafikhata isimli değişkenin aldığı değer sıfır olur.

16.2.8 Graphresult

Grafik işleminde oluşan hatanın kodunu getirir. Hata kodu sıfırdan farklı ise, grafik işeminde hata olduğu görülür. Örnek programı inceleyiniz.

Örnek

```
uses Graph;
var
  Hatakodu: Integer;
  Grafiksürücü, Grafikkonumu: Integer;
begin
  GrDriver := Detect;
  InitGraph(Grafiksürücü, Grafikkonumu, 'c:\bp\bgi');
  Hatakodu:= GraphResult;
  if Hatakodu <> 0 then
  begin
    Writeln('Grafik hatası:');
```

```
    Writeln(GraphErrorMsg(Hatkodu));
    Writeln('Program Kesildi');
    Halt(1);
  end;
  ClearDevice;
  Rectangle(0, 0, GetMaxX, GetMaxY);
  Readln;
  CloseGraph;
end.
```

16.2.9 SetViewport

Bu komut grafik ekranda bir grafik penceresinin tanımlanması amacıyla kullanılır. Grafik ekrana geçildiğinde başlangıç olarak getmaxX+1 genişliğinde ve getmaxY+1 yüksekliğinde bir grafik pencere oluşur. Programcı gerektiğinde bu pencerenin boyutlarını değiştirerek daha küçük boyutlara sahip bir alanda çizim yapabilir. Aşağıda verilen Setviewport komutunun kullanımında X1,Y1, X2 ve Y2 pencerenin tanımlanmasında, Clipoff/Clipon ise çizimin pencere dışına taşan kısmının kesilip kesilmeyeceğine karar vermek için kullanılır. Kesme yapılmaması için Clipoff, kesme yapılması için Clipon kullanılır.

Setviewport iki amaç için kullanılır. Bunlar;

Grafik ekranın (0,0) koordinatının ekranda başka bir yere taşınması
Grafik ekran içinde bir pencere tanımlanmasıdır.

Orijinin Taşınması: Grafik sistemi başlatıldığında (0,0) noktası, ekranın sol üst köşesidir. X sağa doğru (+), Y aşağıya doğru (+) değerlerine sahiptir. Setviewport komutuyla eksenlerin yönünü değiştirmeden (0,0) koordinatını başka bir yere taşımak için komutla birlikte kullanılması gereken parametreler şu şekildedir;

X1,Y1 :Yeni orijinin grafik ekrandaki yeni koordinatı
X2,Y2 :X1<X2 ve Y1<Y2 olmak üzere herhangi bir koordinat
Kesme :Clipoff;

Orijini ekranın ortasına almak için aşağıda verilen komut kullanılabilir.

```
Setviewport(Trunc(getmaxX/2), Trunc(getmaxY/2), getmaxX, getmaxY,Clipoff);
```

Yeni Pencere Oluşturulması: tüm grafik ekranı içinde daha küçük bir grafik ekranın tanımlanması işlemidir. Tanımlanan grafik penceresi boyutları hariç, tüm grafik ekranın taşıdığı özelliklere sahiptir.

Setviewport ile bir grafik pencerenin tanımlanması için verilen parametrelerin alması gereken değerler şu şekilde verilebilir.

X1,Y1:Yeni grafik pencerenin sol üst köşe koordinatı
X2,Y2:X1<X2 ve Y1<Y2 olmak üzere yeni grafik pencerenin sağ alt köşe koordinatı
Kesme:Taşan kısımın kesilmesi için Clipon, kesilmemesi için Clipoff

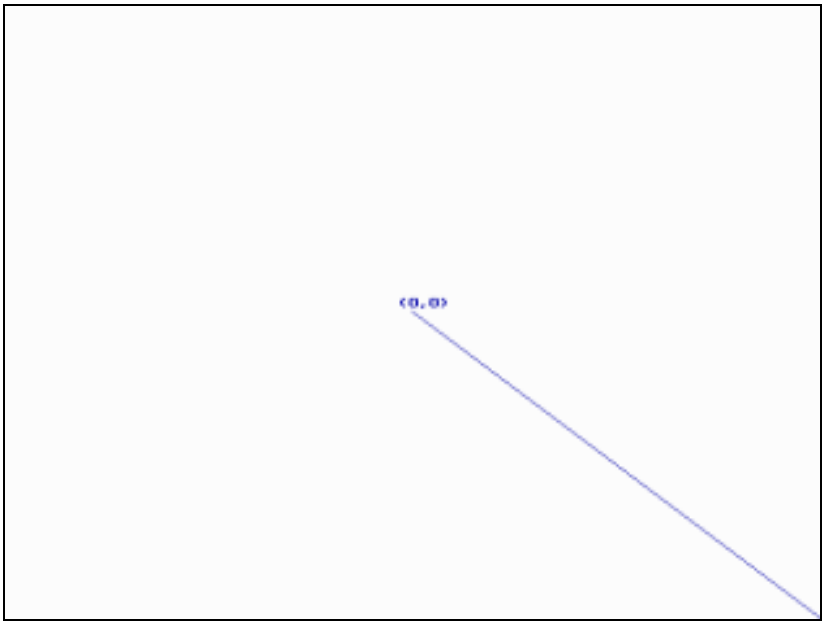
Setviewport(X1,Y1,X2,Y2;Clipon); şeklinde kullanılır.

Aşağıda verilen örnek programda orijin ekranın merkezine taşınmış ve orijinden ekranın sağ alt köşesine bir çizgi çizilmiştir. Programın çalıştırılması sonucu oluşan ekran görüntüsü programın sonunda verilmiştir.

Örnek:

```
uses graph;
var
x,y,grafiksurucu,grafikkonumu:integer;
Function Kataryap(x,y:integer):string;
var
sx,sy:string[5];
begin
  str(x,sx);  str(y,sy);
  Kataryap:='('+ Sx+','+Sy+')';
end;
begin
  detectgraph(grafiksurucu,grafikkonumu);
  initgraph(grafiksurucu,grafikkonumu,'c:\bp\bgi');
  setbkcolor(white);
  setcolor(1);
  setviewport(trunc(getmaxX/2),trunc(getmaxy/2),getmaxX,getmaxy,clipoff);
  line(0,0,getmaxX,getmaxy);
  moveto(-10,-10);
  outtext(kataryap(0,0));
readln;
closegraph;
end.
```

Orijin noktası değiştirildiğinde orijinin üstünde kalan bölgeler kullanılacak ise kesme yapılmayacağını ifade etmek için setviewport komutuna *Clipoff* parametresi yazılır. Sadece yeni pencere kullanılacak ise *Clipon* yazılır. Örnek programda kesme kullanılmamış olup, moveto (-10,-10) komutu ile tanımlanan pencere dışına taşılmıştır. Moveto komutu grafik imleci, belirtilen koordinata konumlandırmak amacıyla kullanılır.



Programın çalışması sonucu elde edilen grafik ekran

Cleardevice komutunun setviewport komutundan sonra kullanılması durumunda sadece tanımlanan yeni pencere içindeki grafik ekran temizlenir.

16.3 Çizim Komutları

En çok kullanılan çizim komutları aşağıda özetlenmiştir.

16.3.1 PutPixel

PutPixel komutu yardımıyla koordinatı verilen pixel, belirtilen renge boyanır. Kullanım şekli;

```
PutPixel(x,y,renk);

X:pikselin kaçıncı sütuna karşılık geldiği;
Y:Pixelin kaçıncı satıra karşılık geldiği;
Renk:pixelin boyanacağı renk;

Putpixel(100,20,blue);
```

Şeklinde yazılan bir komut ile, 100. sütun ve 20. satıra karşılık gelen pixel mavi renk ile boyanır.

Bir doğrunun denklemi analitik olarak **Y=ax+b** şeklinde ifade edilir. Verilen X1,Y1 ve X2,Y2 noktalarından geçen doğrunun a ve b parametrelerini ;

Y1=ax1+b
Y2=ax2+b

Denklem takımını çözerek bulabiliriz.

a=(x1.y2-x2.y1)/(x1-x2)
b=(y2-y2)/(x2-x1)

a ve b değerleri elde edildikten sonra x1>x2 olduğu kabul edilerek, x ekseninde X1 'den X2 'ye ilerlerken y eksenindeki değerleri de;

y=ax+b

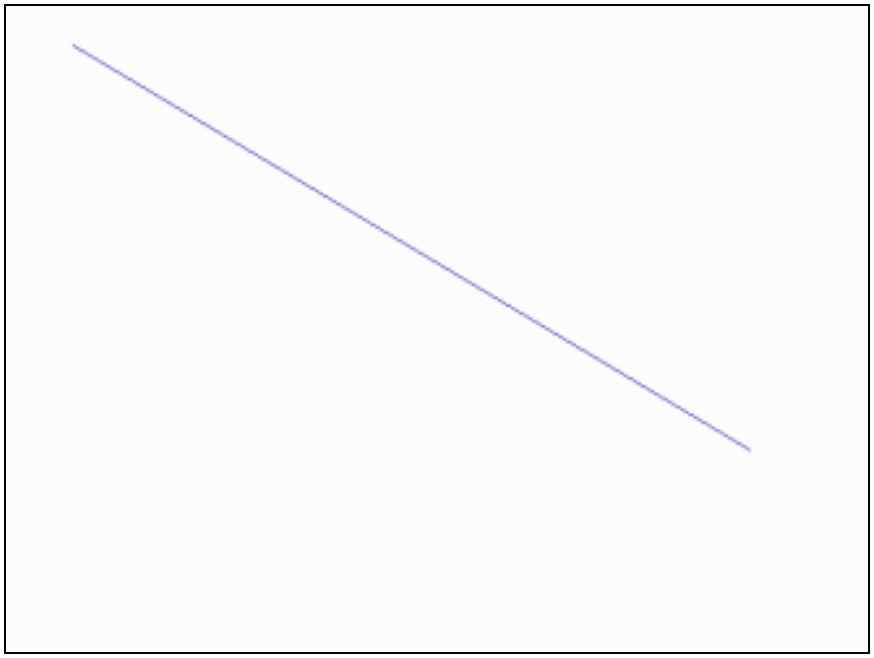
eşitliğinden elde edebiliriz. Pixel komutu kullanarak y=ax+b şeklindeki doğrunun çizimi aşağıdaki programda verilmiştir.

Örnek:

```
uses graph;
var
x1,y1,x2,y2,grafiksurucu,grafikkonumu,renk:integer;
dosya :text;
Procedure degistir(var d1,d2:integer);
var z:integer;
begin
   z:=D1;
   d1:=d2;
   d2:=z;
end;
procedure cizgi(x1,y1,x2,y2,renk:integer);
var
a,b:real;
xy,x,y:integer;
begin
if (x1=x2) and (y1=y2) then
  begin
   putpixel(x1,y1,renk);
   exit;
  end;
if x1=x2 then
   begin
    if y1>y2 then degistir (y1,y2);
    for y:=y1 to y2 do
      putpixel(x1,y1,renk);
      exit;
```

```
        end;
      if y1=y2 then
        begin
         if x1>x2 then degistir (x1,x2);
           for X:=x1 to x2 do
        putpixel(x1,y1,renk);
        exit;
        end;
       {denklem parametrelerinin hesaplanması}
      if x2<>x1 then
        begin
          a:=(y2-y1)/(x2-x1);
          b:=(x1*x2-x2*y1)/(x1-x2);
        end;
      if X1>X2 then degistir(x1,x2);
        for X:=x1 to x2 do
          begin
            Y:=trunc(a*x+b);
            putpixel(x,y,renk);
          end;
      end;
      begin
       detectgraph(grafiksurucu,grafikkonumu);
       initgraph(grafiksurucu,grafikkonumu,'c:\bp\bgi');
       x1:=50;
       y1:=50;
       X2:=550;
       Y2:=350;
       RENK:=white;
       cizgi(x1,y1,x2,y2,renk);
      readln;
      closegraph;
      end.
```

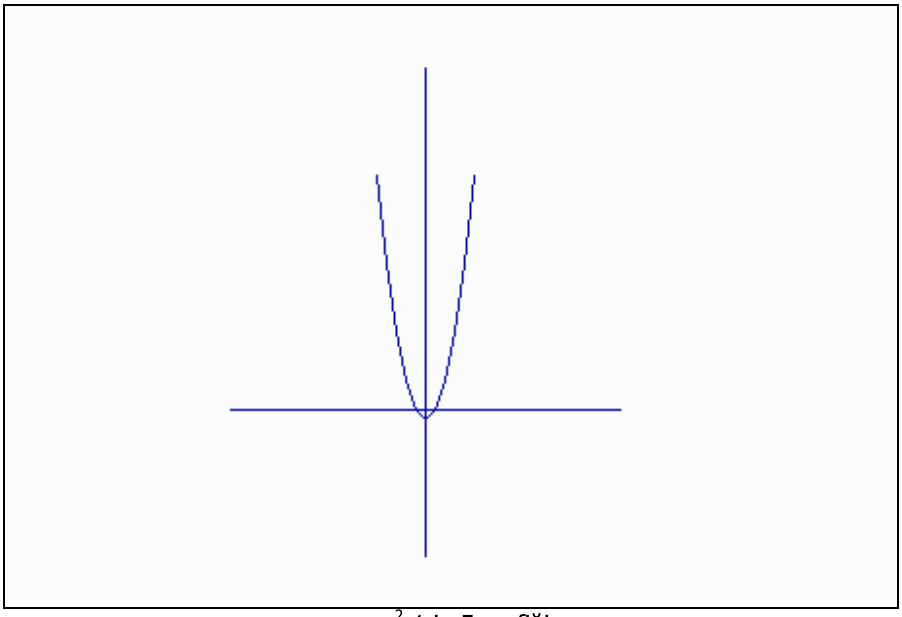
Programın çalışması sonucunda elde edilen doğru aşağıda verilen şekilde görülmektedir.



Örnek:Aşağıdaki program ikinci derece denklemin grafiğini çizmektedir.

```
uses graph;
var
i,grafiksurucu,grafikkonumu:integer;
x1,y1:array[1..50] of integer;
begin
  detectgraph(grafiksurucu,grafikkonumu);
  initgraph(grafiksurucu,grafikkonumu,'c:\bp\bgi');
  setbkcolor(white);
  setcolor(1);
  setviewport(trunc(getmaxX/2),trunc(getmaxy/2),getmaxX,getmaxy,clipoff);
  line(0,0,100,0);  {eksen takımının çizilmesi}
  line(0,0,-100,0);
  line(0,0,0,75);
  line(0,0,0,-175);
  for i:=0 to 5 do
     begin
        Y1[i]:=sqr(i)-1;
        putpixel(i*5,-y1[i]*5,3);      {Grafiğe ait noktalar beş kat abartılı çiziliyor}
        putpixel(-i*5,-y1[i]*5,3);
     end;
  for i:=1 to 5 do
```

```
        begin
          line((i-1)*5,(-y1[i-1])*5,i*5,(-y1[i]*5)); {Putpixel ile konan noktalar line ile
                                                      birleştiriliyor}
          line(-(i-1)*5,(-y1[i-1])*5,-i*5,(-y1[i]*5));
        end;
      readln;
      closegraph;
    end.
```



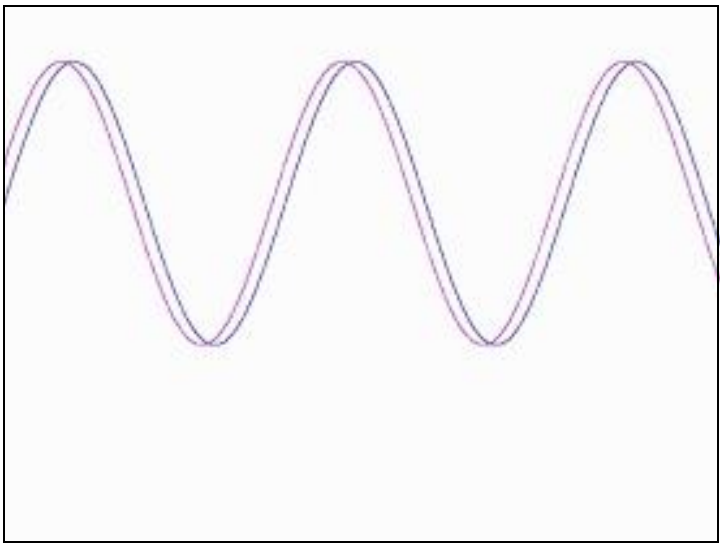
$y=x^2-1$ in 5 grafiği

Programda kullanılan **Setbkcolor**, zemin rengini, **Setcolor** ise çizgi rengini düzenlemek için kullanılır. Renklerin isimleri ve sayısal değerleri aşağıdaki tabloda gösterilmiştir.

Renk	Renk kodu	Anlamı	Renk	Renk kodu	Anlamı
Black	0	Siyah	DarkGray	8	Gri
Blue	1	Mavi	LightBlue	9	Açık Mavi
Green	2	Yeşil	Lightgreen	10	Açık Yeşil
Cyan	3	Turkuaz	Lightcyan	11	Açık Turkuaz
Red	4	Kırmızı	Lightred	12	Açık Kırmızı
Magenta	5	Eflatun	Lightmagenta	13	Açık Eflatun
Brown	6	Kahverengi	Yellow	14	Sarı
Lightgray	7	Açık gri	White	15	Beyaz

Örnek: Verilen program ile sinüs-cosinüs grafiği çizilmektedir.

```
uses graph;
var  gd,gm,y2:integer;
     i,s,c:real;
Begin
  Gd:=Detect;
  initgraph(Gd,Gm,'c:\bp\bgi');
  i:=0; setbkcolor(white);
  y2:=GetmaxX div 5;
    Repeat
      s:=1-Sin(i); C:=1-Cos(i);
      Putpixel(Trunc(40*i),Trunc(y2*s+50),1);
      Putpixel(Trunc(40*(i-5)),Trunc(y2*c+50),5);
      i:=i+0.001;
    until i>=10 *pi;
    readln;
end.
```



Programın çalıştırılmasıyla elde edilen grafik

16.3.2 Çizgi Tipleri

Pascal grafik ortamı başlatıldığında sürekli ince çizgi tipi aktif haldedir. Çizgi tipini ve kalınlığını değiştirmek için Setlinestyle komutu kullanılır. Komutun genel kullanımı;

```
SetLineStyle(çizimtipi,Pattern,Kalınlık);
```

Çizimtipi için kullanılabilecek sabit isimler ve bu isimlerin içerdikleri değerler aşağıdaki tabloda verilmiştir.

Sabit isim	Sayısal değer	Çizim Tipi
SolidLn	0	
DottedLn	1	————————
CenterLn	2	.....
DashedLn	3	—·—·—·—·—·—·—
UsetBitLn	4	?----------

Bu tabloda görülen UserbitLn, ancak kullanıcı tarafından tanımlandığında geçerli olabilir.

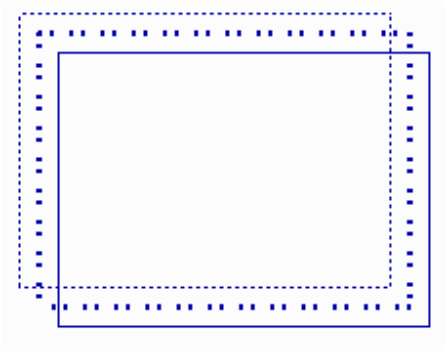
Pattern parametresi, grafik sisteminde tanımlı dört tane doğru çizim tipi dışında kullanıcının kendi çizim tipini tanımlamakta kullandığı değeri içeren parametredir.

Kalınlık parametresi; çizilen çizginin kalınlığının tanımlandığı bir tamsayı değerdir. Bu parametre için sadece 2 değer tanımlanmıştır. Bunlar aşağıda verilmiştir.

NormWidth (1) 1 pixel kalınlığında,
ThickWidth (3) 3 pixel kalınlığında,

Örnek:

```
uses Graph;
var
X1, Y1, X2, Y2,Gd, Gm: Integer;
 begin
 Gd := Detect;
 InitGraph(Gd, Gm, 'c:\bp\bgi');
 setbkcolor(white);setcolor(blue);
 X1 := 10; Y1 := 10; X2 := 200; Y2 := 150;
 SetLineStyle(DottedLn, 0, NormWidth);
 Rectangle(X1, Y1, X2, Y2);
 SetLineStyle(UserBitLn, $C3, ThickWidth);
 Rectangle((X1+10), (Y1+10), (X2+10), (Y2+10));
 SetLineStyle(SolidLn, 0, NormWidth);
 Rectangle(X1+20, Y1+20, X2+20, Y2+20);
 Readln;CloseGraph;
end.
```



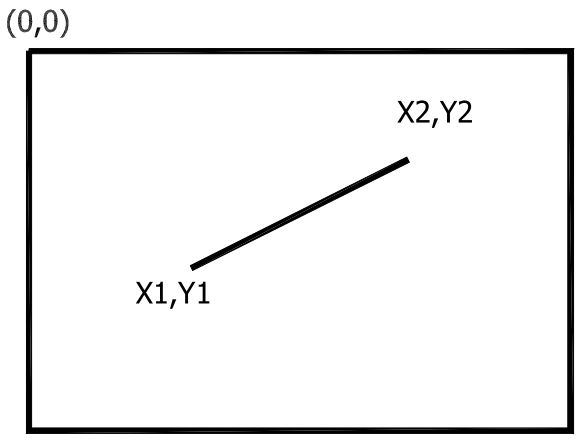
Programın çalıştırılması sonucu elde edilen grafik görüntüsü

16.3.3 Line Komutu

Bir doğru parçası çizimini gerçekleştirilmesi için Line komutu kullanılır. Line komutu ile bir doğru parçası çizimi için, Borland Grafik Koordinat sistemine göre belirlenmiş olan bir başlangıç ve bir de bitiş koordinatı kullanılır. Komutun kullanımı;

```
Line(X1,Y1,X2,Y2);
```

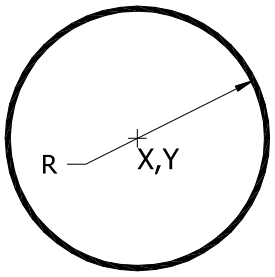
şeklindedir. Koordinat tanımlamalarında kullanılan değişkenlerin tamsayı olması gerekir.



Line Komutunda kullanılan parametreler

16.3.4 Circle Komutu

Circle Komutu, daire çizimi için kullanılır. Çizimin gerçekleştirilmesi için; X,Y şeklinde dairenin merkez koordinatı ve dairenin yarı çapı değerleri verilir.



Circle komutunda kullanılan parametreler

Komutun kullanılması;

```
Circle(X,Y,R);
```

şeklindedir. Reel sayılarla çizim yapmak mümkün olmadığından Burada kullanılan bütün değerlerin tam sayıya yuvarlatılması gerekir.

16.3.5 Arc Komutu

Bir daire parçası çizimi için kullanılır. Komutun kullanımı;

```
Arc(X,Y,Açı1,Açı2,R);
```

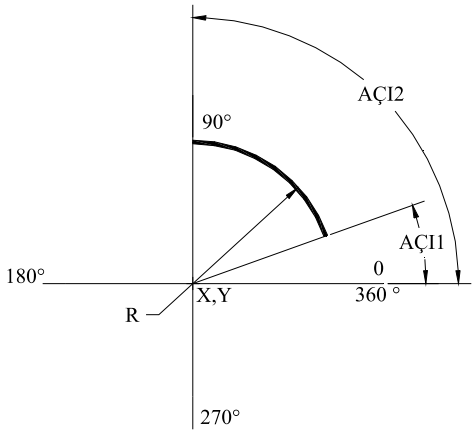
şeklindedir. Burada;

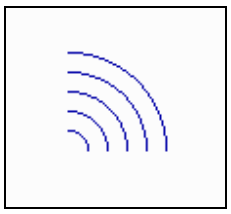
X :yayın X koordinatı,
Y :yayın Y koordinatı
Açı1 :yayın başlangıç açısı,
Açı2 :yayın bitiş açısı,
R :yayın yarı çapıdır.

Örnek:

```
uses Graph;
var
  Gd, Gm: Integer;
  Radius: Integer;
begin
  Gd := Detect;
```

```
        InitGraph(Gd, Gm,'c:\bp\bgi ');
        setbkcolor(15);
        setcolor(1);
          for Radius := 1 to 5 do
          Arc(100, 100, 0, 90, Radius * 10);
        Readln;
        CloseGraph;
      end.
```





Programla elde edilen grafik görüntüsü

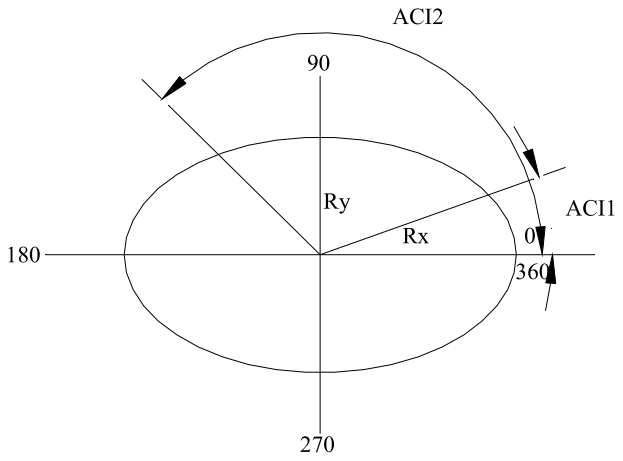
16.3.6 Ellipse Komutu

Bir elips veya eliptik yay çizimi için kullanılan komut "ellipse" ' dir. Komutun kullanımı;

```
Ellipse(X,Y,Açı1,Açı2,Rx,Ry);
```

şeklindedir. Komut ile birlikte kullanılan paramatreler;

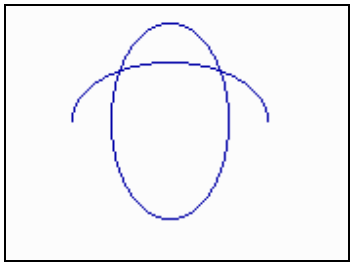
X :Elips merkezinin X koordinatı
Y :Elips merkezinin Y koordinatı
Açı1,Açı2:Elips veya eliptik yayın başlangıç ve bitiş açısı
Rx :Elipsin X yönündeki yarıçapı,
Ry :Elipsin Y yönündeki yarıçapıdır.



Açı1=0° ve Açı2=360° olduğunda tam bir elips elde edilir.

Örnek:

```
uses Graph;
var Gd, Gm: Integer;
begin
  Gd := Detect;
  InitGraph(Gd, Gm, 'c:\bp\bgi ');
  setbkcolor(15);
  setcolor(1);
  Ellipse(100, 100, 0, 360, 30, 50);
  Ellipse(100, 100, 0, 180, 50, 30);
  Readln;
  CloseGraph;
end.
```



Program ile elde edilen grafik

16.3.7 Grafik İmlecin Konumunun Değiştirilmesi

Grafik imlecin bulunduğu konumun değiştirilmesi amacıyla Moveto ve Moverel komutları kullanılır.

Moveto ile grafik imlecin yerinin değiştirilmesi işleminde yeni konumun koordinatları tanımlanır. Moverel komutunda ise, grafik imlecin bulunduğu konumdan itibaren X ve Y yönündeki yerdeğişim miktarları tanımlanır.

```
Moveto(X,Y);

Moverel(dx,dy);
```

Çizimlerde grafik imlecin kullanılması için lineto ve linerel komutları kullanılır. Kullanımları moveto ve moverel gibidir. Grafik imlecin bulunduğu noktadan belirilen koordinata kadar doğru çizimi için Lineto, bulunduğu noktadan dx ve dy miktarlarında yer değiştirerek doğru çizimi için Linerel komutları kullanılır. Kullanımları;

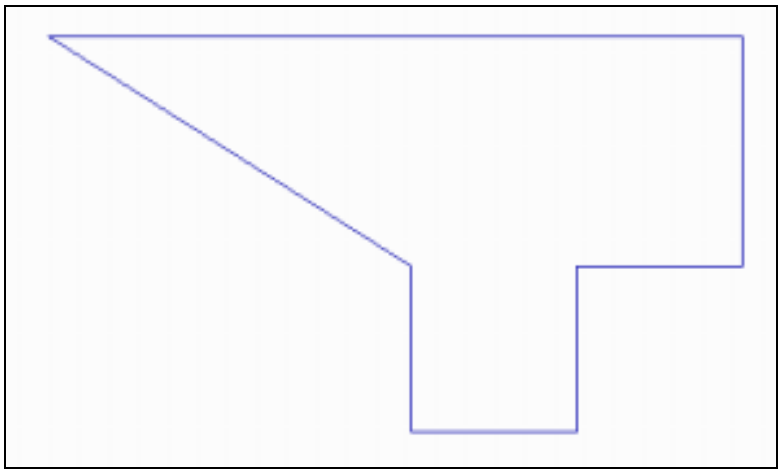
```
Lineto(X,Y);
Linerel(dx,dy);
```

şeklindedir.

Örnek:

```
uses Graph;
var Gd, Gm: Integer;
begin
  Gd := Detect;
  InitGraph(Gd, Gm, 'c:\bp\bgi ');
  setbkcolor(15);setcolor(1);
  MoveTo(100,100); { Upper left corner of viewport }
  LineTo(trunc(GetMaxX/2), trunc(GetMaxY/2));
  Linerel(0,100);
  Linerel(100,0);
  Linerel(0,-100);
  Linerel(100,0);
 Linerel(0,-139);
```

```
        lineto(100,100);
        Readln;
        CloseGraph;
      end.
```



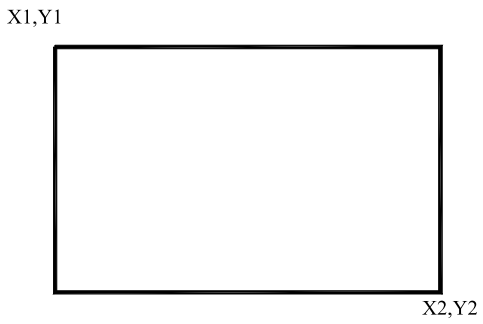
Programla elde edilen çizim

16.3.8 Rectangle Komutu

Rectangle komutu, dörtgen çizimi için kullanılır. Komutun kullanımı;

```
Rectangle(X1,Y1,X2,Y2);
```

şeklindedir. X1,Y1; dörtgenin sol üst köşe koordinatı, X2,Y2; dörtgenin sağ alt köşesinin koordinatlarıdır.



Örnek:

```
uses crt,Graph;
var
  GraphDriver, GraphMode: Integer;
  X1, Y1, X2, Y2: Integer;
begin
  GraphDriver := Detect;
  InitGraph(GraphDriver, GraphMode, 'c:\bp\bgi');
  Randomize;
  repeat
    X1 := Random(GetMaxX);
    Y1 := Random(GetMaxY);
    X2 := Random(GetMaxX - X1) + X1;
    Y2 := Random(GetMaxY - Y1) + Y1;
    Rectangle(X1, Y1, X2, Y2);
    Delay(100);
 until KeyPressed;
  CloseGraph;
end.
```

Yukarıda verilen örnek program herhangi bir tuşa basılıncaya kadar rastgele büyüklükte ve koordinatta dörtgen çizmektedir.

16.3.9 DrawPoly Komutu

Drawpoly komutu ihtiyaç duyulan üçgen, yamuk, değişik sayıda kenarlara sahip çokgenlerin çizilebilmesi için kullanılır. Bu komut genelde poligonların çizimi için hazırlanmıştır. Kullanım şekli;

```
DrawPoly (noktaadedi, noktakoordinatlari)
```

şeklindedir. Burada nokta adedi, poligonu oluşturan köşe sayısı, nokta koordinatlari ise bu noktaların koordinatlarıdır. Komutun kullanılması için poligonun köşe noktalarını içeren bir bir dizi tanımlanır. Örneğin 4 köşe noktası olan bir poligon tanımlanmak istenirse; her bir köşenin bir elemanı X ve bir elemanı Y için olmak üzere 8 elemanlı bir dizi tanımlamak gerekir.

Tanımlama bloğunda;

```
Var
    Poly:array[0..7] of integer;
```

Şeklinde bir dizi tanımladıktan sonra bu elamanlara aşağıdaki şekilde uygun koordinat değerleri atanır.

```
Poly[0]:=100;          Poly[1]:=150;
```

```
Poly[2]:=50;        Poly[3]:=75;
Poly[4]:=75;        Poly[5]:=50;
Poly[6]:=200;       Poly[7]:=75;
```

Bu dizide çift indis numaralı elemanlar poligonun X koordinatını, tek indis numaralı elemanlar Y koordinatını tanımlamaktadır.

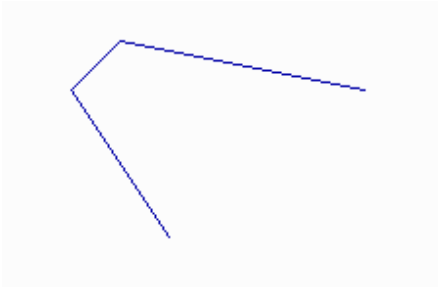
Koordinatlara değer ataması yapıldıktan sonra;

```
Drawpoly(4,poly);
```

Komutu ile poligonun çizimi gerçekleştirilir. Yukarıda tanımlanan koordinatlardan yararlanarak poligon elde edilmesi için yazılan Pascal programı aşağıda verilmiştir.

Örnek:

```
uses graph;
Var
    Poly:array[0..7] of integer;
    gd,gm:integer;
begin
    Poly[0]:=100;       Poly[1]:=150;
    Poly[2]:=50;        Poly[3]:=75;
    Poly[4]:=75;        Poly[5]:=50;
    Poly[6]:=200;       Poly[7]:=75;
    detectgraph(gd,gm);
    initgraph(gd,gm,'c:\bp\bgi');
    setbkcolor(white);setcolor(1);
    drawpoly(4,poly);
    readln;
    closegraph;
end.
```



Program ile elde edilen çizim

16.3.10 FillPoly

Fillpoly, içi dolu çokgen oluşturmak için kullanılır. Kullanım şekli drawpoly ile aynıdır.

16.3.11 Setfillstyle

Setfillstyle, çizilen alan içini doldurmak için kullanılır. Setfillstyle komutunun kullanımı;

Setfillstyle(tarama deseni,Renk);

şeklindedir. Tarama desenleri aşağıdaki tabloda verilmiştir.

Tarama Deseni	Sabit	Anlamı
Emptyfill	0	Tarama deseni zemin rengiyle aynı tarama stili
Solidfill	1	Belirtilen renk kodu ile taranır.
Linefill	2	Yatay çizgiler
Ltslashfill	3	Sağa doğru 45° eğimli ince tarama çizgisi
Slashfill	4	Sağa doğru 45° eğimli kalın tarama çizgisi
Bkslashfill	5	Sola doğru 45° eğimli kalın tarama çizgisi
LtBkSlashfill	6	Sola doğru eğimli çift kalınlıklı tarama çizgisi
Hatchfill	7	Düşey ve yatay kombine tarama çizgilileri
Xhatchfill	8	Çapraz çizgiler
Interleavefill	9	Kesikli çizgiler
Widedotfill	10	Aralıklı noktalar
Closedotfill	11	Sık noktalar
Userfill	12	Kullanıcının tanımlayacağı tarama stili

Pascal ile yazılan programlarla simulasyon programları da yazmak mümkündür. Aşağıda örnek olarak verilen programda eşkenar dörtgen herhangi bir tuşa basılıncaya kadar belirlenen yörüngede hareket etmektedir.

```
UNIT HAREKET; {unit program}
INTERFACE
USES graph;
VAR A,C:INTEGER;
PROCEDURE TAKIM (x,y:integer);
PROCEDURE BASLA;
IMPLEMENTATION
PROCEDURE TAKIM (x,y:integer);
  var poly: array [0..9] of integer;
  begin
   setfillstyle(1,red);
   poly[0]:=x;poly[1]:=y;poly[2]:=x+15;poly[3]:=y-5;poly[4]:=x+17;
```

```
    poly[5]:=y-18;poly[6]:=x+4;poly[7]:=y-15;
    poly[8]:=poly[0];poly[9]:=poly[1];
    setcolor(blue);
    fillpoly(5,poly);
    setcolor(white);
    line(x,y,x+16,y-4);
    line(x+16,y-4,x+18,y-19);
    line(x+18,y-19,x+3,y-16);
    line(x+3,y-16,x,y);
  end;
PROCEDURE BASLA;
BEGIN
SETCOLOR(BLUE);
  REPEAT
    A:=200; C:=400;
    REPEAT
      C:=C-1;
      TAKIM(C,A);DELAY(25);
    UNTIL  (C=200) OR KEYPRESSED;
    REPEAT
      C:=C+1;
      TAKIM(A,C);DELAY(25);
    UNTIL  (C=400) OR KEYPRESSED;
     REPEAT
      C:=C-1;
      A:=A+1;
      TAKIM(A,C);DELAY(25);
    UNTIL  (C=200) OR KEYPRESSED;
  UNTIL KEYPRESSED;
END;
BEGIN
END.
```

```
uses crt,graph,hareket;  {Ana program}
var
   gd,gm:integer;
begin
  gd:=detect;
  initgraph(gd,gm,'c:\bp\bgi');
  setbkcolor(white);
  basla;
  closegraph;
end.
```

Yukarıda verilen programda delay değerini arttırmak/azaltmak ile cismin hızı azalır veya artar.

16.3.12 Bar

Bar komutu ile içi doldurulmuş dikdörtgenin çizimi gerçekleştirilir. Dikdörtgenin içi **setfillstyle** veya **setfillpattern** komutu ile belirlenmiş olan tarama deseniyle doldurulur. Bar komutunda verilen parametreler rectangle komutunda verilenler ile aynıdır.

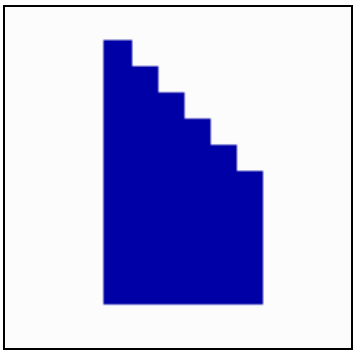
```
Bar(X1,Y1,X2,Y2);
```

Burada;

X1,Y1:Dörtgenin sol üst köşesi
X2,Y2: Dörtgenin sağ alt köşesi

Örnek:

```
uses Graph;
var
  Gd, Gm, I, genislik: Integer;
begin
  Gd := Detect; InitGraph(Gd, Gm,'c:\bp\bgi');
  genislik := 10;
  setbkcolor(white);
  setfillstyle(solidFill,1);
  for I := 10 to 15 do
    Bar(I*genislik, I*10, (I-1)*genislik, 200);
  ReadLn;
  CloseGraph;
end.
```



Örnek programın çalışması sonucunda elde edilen grafik

16.3.13 Bar3D

Bu komut,3 boyutlu görünüşe sahip bir dörtgenin çizimini gerçekleştirir. Kullanım şekli;

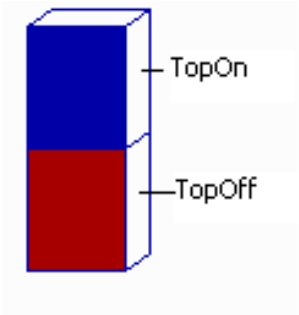
```
Bar3d (X1,Y1,X2,Y2,D,UST);
```

X1,Y1:Dörtgenin sol üst köşesi
X2,Y2:Dörtgenin sağ alt köşesi
D:3 boyutlu dörtgenin pixel olarak derinliği
UST:3 boyutlu dörtgenin üstünün çizilip çizilmeyeceğini belirtir.

Ust ifadesi için kullanılacak parametreler; TopOn (True) ve Topoff (False) 'dir. Bu parametre TopOn olarak kullanılırsa 3 boyutlu dörtgenin üst kısmı çizilir, diğer halde çizilmez. TopOff parametresinin kullanım amacı, farklı 3 boyutlu dörtgenlerin üst üste çizilmesini sağlamaktır.

Örnek:

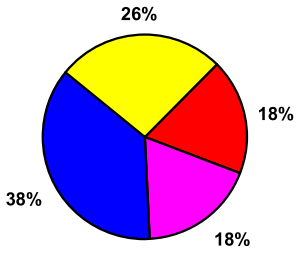
```
uses Graph;
var
  Gd, Gm: Integer;
  Y0, Y1, Y2, X1, X2: Integer;
begin
  Gd := Detect;
  InitGraph(Gd, Gm,'c:\bp\bgi');
  Y0 := 10;
  Y1 := 60;
  Y2 := 110;
  X1 := 10;
  X2 := 50;
  setbkcolor(white);
  setcolor(blue);
  setfillstyle(solidFill,blue);
  Bar3D(X1, Y0, X2, Y1, 10, TopOn);
  setfillstyle(solidFill,red);
  Bar3D(X1, Y1, X2, Y2, 10, TopOff);
  Readln;
  CloseGraph;
end.
```



Bar ve Bar3d komutları kullanılarak amaçlarımız doğrultusunda histogram grafikler çizilebiliriz.

16.3.14 Pieslice

Histogram grafiklerinden başka çok kullanılan diğer bir grafik tipi de pie (pay-dilim) grafikleridir. Pie grafikler bir daire %100 'ü göstermek üzere; olaylara ait yüzde orana göre oluşturulmuş grafiklerdir.



Bir Pie Grafiği

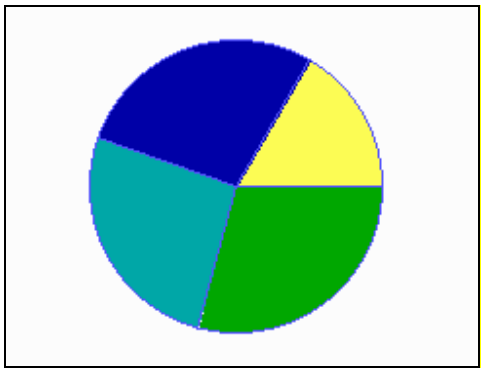
Komutun kullanımı;

```
Pieslice (X,Y,Aci1,Aci2,R);
```

Görüldüğü gibi komutun kullanımı Arc komutuna benzerdir. Pieslice, X,Y merkez koordinatlı, Açı1 'den başlayıp Açı2 'de biten R yarıçaplı daire dilimi çizer.

Örnek:

```
uses Graph;
const Radius = 75;
var Gd, Gm: Integer;
begin
  Gd := Detect;
 InitGraph(Gd, Gm, 'c:\bp\bgi');
  setbkcolor(white);
  setcolor(lightblue);
  SetFillStyle(solidFill,yellow);
  PieSlice(200, 200, 0, 60, Radius);
  SetFillStyle(solidFill,blue);
  PieSlice(200, 200, 60, 160, Radius);
  SetFillStyle(solidFill,cyan);
  PieSlice(200, 200, 160, 255, Radius);
  SetFillStyle(solidFill,green);
  PieSlice(200, 200, 255, 360, Radius);
  ReadIn;
  CloseGraph;
end.
```



Örnek programın çalıştırılması sonucu elde edilen Pie grafiği

16.4 Grafik Ortamda Yazı

Grafik ekranda yazı yazmak, DOS ekranında yazı yazmadan oldukça önemli farklılıklar içermektedir. DOS ekranı üzerinde yazı yazmak için WRITE, WRITELN komutlarının kullanıldığını biliyoruz. Bu komutlar ile aynı büyüklükte yazı yazabilmekteyiz. Grafik ekranda bu komutlar kullanılamaz. Grafik ekranda yazı yazabilmek için BGI dizini içinde .CHR uzantılı dosyaların bulunması gerekir. Grafik ekranda yazı yazarken farklı karakterlerde ve büyüklüklerde yazı yazabilmek mümkündür.
Yazım işlemine başlamadan önce yazım ile ilgili bilgilerin grafik sistemine verilmesi

gerekir. Yazım ile ilgili bilgiler 3 tanedir; Bunlar; yazının font tipi,yazım yönü ve büyüklüğüdür. Bu özelliklerin bildirilmesi için **Settextstyle** komutu kullanılır. Komutun kullanılması;

```
SetTextStyle(Font,Yon,Boyut);
```

şeklindedir. Bu komutu kullanarak, grafik ekrana yazdırılacak yazının hangi font tipi kullanılarak, hangi yönde ve hangi büyüklükte olacağı belirtilir.

16.4.1 Font

Yazının ekrana yazdırılmasında kullanılacak olan font karakter dosyasını belirten tamsayı değerdir. Bu değerler ve bu değerleri temsil eden sabit isimler aşağıdaki tabloda verilmiştir.

Sabit isimler	Sabit Değerler
DefaultFont	0
TriplexFont	1
SmallFont	2
SansSerifFont	3
GothicFont	4

Programda yazı fontu belirtilmediği takdirde font, Pascal 7.0 tarafından DefaultFont olarak kabul edilecektir.

16.4.2 Yazı Yönü

Bu parametre, grafik ekrana yazdırılacak olan yazının hangi yönde yazılacağını belirten tamsayı değerdir. Yazı sağdan sola yazılacak ise yon değerine; "0", aşağıdan yukarı yazılacaksa "1" değerleri atanır.

16.4.3 Yazı Büyüklüğü

Yazı büyüklüğü, boyut parametresi ile belirtilir. Bu değer büyütme faktörü olarak adlandırılabilir. 1-10 arasında tamsayı değer almaktadır. Boyut değerine atanan tamsayı "1" ise yazının boyutu font dosyasında tanımlandığı şekilde olacaktır. 1 'den büyük değerler verildiğinde, yazı boyutu verilen değerin katı olarak büyüyecektir.

16.4.4 Metnin Ekrana Yazdırılması

Ekrana bastırılacak yazı özelliği SettextStyle ile verildikten sonraki aşama yazının ekranın hangi koordinatına bastırılacağının belirtilmesi gerekir. Yazının yazdırılması için iki komut vardır. Bunlar; Outtext ve OuttextXY 'dir. Her iki komutun kullanımı birbirine benzerdir.

Outtext komutuyla yazının yazılacağı yerin belirlenmesi işleminde **Moveto** komutu kullanılır. Moveto komutu kullanılmadan Outtext komutu kullanılırsa yazı, imlecin bulunduğu noktadan itibaren yazdırılır.

```
Moveto(100,100);
Outtext('Yazilacak Yazi');
```

OuttextXY komutunda ise moveto komutu kullanmaya gerek olmadan yazının yazılacağı koordinatı belirtmek mümkündür. Komutun kullanımı;

```
OuttextXY(X,Y,'yazilacak Yazi');
```

Bu komutların kullanılmasında dikkat edilecek en önemli özelliklerden biri, program içinde hesaplanan değişkenler direk olarak yazdırılamaz. Değişkenlerin aldığı değerler öce string ifadeye dönüştürülüp sonra bu komutlar yardımıyla yazdırılır. (Bu bölümde verilen ilk örnekteki Kataryap isimli fonksiyonu inceleyiniz).

Aşağıda verilen örnek programda Pascal 'da kullanılan fontlar ve büyüklüklere örnekler verilmiştir.

```
uses Graph;
var i, Gd, Gm: Integer;
begin
Gd := Detect;
  InitGraph(Gd, Gm, 'c:\bp\bgi');
  setbkcolor(yellow);
  setcolor(blue);
  outtextxy(30,10,'FONT ORNEKLERi');
  outtextxy(30,20,'***************');
  outtextxy(30,30,'BALIKESiR UNiVERSiTESi,  {Default font}');
    settextstyle(1,0,1);
    Moveto(30,50);
    Outtext('BALIKESiR UNiVERSiTESi');
    settextstyle(1,0,2);
    Moveto(30,80);
    Outtext('BALIKESiR UNiVERSiTESi');
    settextstyle(1,0,3);
```

```
          Moveto(30,110);
          Outtext('BALIKESiR UNiVERSiTESi');
        for i:=1 to 3 do
        begin
          settextstyle(2,0,i+3);
          outtextxy(30,(i*20)+130,'BALIKESiR UNiVERSiTESi');
        end;
        for i:=1 to 3 do
        begin
          settextstyle(3,0,i);
          outtextxy(30,(i*20)+190,'BALIKESiR UNiVERSiTESi');
        end;
        for i:=1 to 3 do
        begin
          settextstyle(4,0,i);
          outtextxy(30,(i*20)+260,'BALIKESiR UNiVERSiTESi');
        end;
        for i:=1 to 3 do
        begin
          settextstyle(5,0,i);
          outtextxy(30,(i*20)+330,'BALIKESiR UNiVERSiTESi');
        end;
        Settextstyle(0,1,0);
        outtextxy(10,10,'Balikesir Universitesi Muhendislik-Mimarlik Fakultesi');
        outtextxy(380,30,'Makine Muhendisligi Bolumu, Konst. ve imalat ABD.');
        Readln;
        CloseGraph;
      end.
```

Örnek programın çalışması sonucu elde edilen yazı örnekleri

EK A.

HATA MESAJLARI

Pascal ile yazilan programlarin derlenmesinde derleyicinin saptadigi hatalar ve programin çalistirilmasi sirasinda ortaya çikan hatalar olmak üzere iki çesit hata durumu vardir.

1. Compiler Errors Messagee
2. Run Time Errors Messages

Compiler Errors (Derleme Hatalari)

Yazim kurallarina ters düsen, hatali kodlamalar yapilan programin derlenmesi sirasinda ortaya çikan hatalardir. Saptanan hata, hataya iliskin kod ile birlikte açiklanir.

1) Out of memory
Hafizanin yetersiz oldugunu belirtir. Bu problemi çözmek için asagidaki yollar denenmelidir.
✍✍ Compile menüsündeki "Destination memory" seçenegi "Disk" olarak degistirilir.
✍✍ "Options" menüsündeki "Linkler" seçenegide "Linker buffer" alani "Disk" yapilir.
Bu islemin DOS komutu satirindaki derleme seçenegi /L 'dir. (>TPC/L...)
✍✍ Hafizada çalisan TSR programlarinin hafizadan silinmesini saglar.
Bu islemlere ragmen hata çözülmezse unit programlarin hacimleri küçültülmelidir.

2) Identifier expected
Turbo Pascalin bir kelime (reserved word) kelime, tanitici ismi olarak kullaniyor.

3) Unknown identifier
Tanimlanmis bir degisken kullaniyor.

4) Duplicate identifier
Tanitici isim mevcut program blogu içerisinde daha önce tanimlanmis.

5) Syntax error
Kaynak text içerisinde geçersiz karakter bulundu.

6) Error in real constant
Sabit real taniminda hata var.

7) Error in integer constant
Sabit integer taniminda hata var.

8) String constant exceeds line
String sabitlerde kullanilan tirnak eksik veya fazla.

9) To many nested files
Içiçe çok fazla dosya baglantisi var

10) Unexpected end of file
Program sonundaki END yazilmamis veya dosya sonunu belirten (.) unutulmus.

11) Line too long
Bir satir üzerinde 120 karakterden fazla bilgi bulunamaz.

12) Type identifier expected
Type tanimlayici gerekli

13) Too many open files
Ayni anda açik olan dosya sayisi çok fazla. Config.Sys dosyasindaki Files=xx seçeneginde xx arttirilmalidir. xx ayni anda açilabilecek dosya sayisini gösterir.

14) Invalid file name
Geçersiz dosya adi veya bulunmayan directory.

15) File not found
Belirtilen dosya aktif directory içerisinde bulunmuyor.

16) Disk full
Disk dolu. Diskten silinebilecek dosyalari silin.

17) Invalid compiler direvtive
Geçersiz compiler bildirisi.

18) Too many files
Programin tamaminda cok fazla dosya veya unit var.

19) Undefined type in pointer definition
Pointer taniminda bilinmeyen tip.

20) Variable identifier expected
Degisken tanimlayicisi bekleniyor.

21) Error in type
Tip hatasi. Sembol ile tip tanimina baslanamaz.

22) Structure too large
Yapi çok büyük. Yapisal tip taniminda yapi olusturan alanlarin toplam uzunlugu 65520 byte olmalidir. Bunun için alan tanimlamalarinda, alan uzunluklari kisaltilmalidir.

23) Set base type out of range
Küme çok uzun. Küme tipi taniminda kümeyi olusturan toplam eleman sayisi 256k yi geçemez. Küme tanimlamalarinda eleman sayisi azaltilmalidir.

24) File compenents may not be files
Bir dosyaya ait tip olarak baska bir dosya tipi kullanilmaz

25) Invalid string length
String'in uzunlugu geçersiz. Bir stringin maksimum uzunlugu 255 karakter olabilir. Bu uzunluk kisaltilmali veya parçalanmalidir.

26) Type mismatch
Tip uyusmazligi. Asagidaki durumlarda meydana gelir:
- Atama veya hesaplama islemlerinde sol taraftaki degisken ile sag taraftaki ifadelerin tipleri ayni olmayabilir.
- Procedure veya function'larda karsilikli gelen parametrelerin tipleri farkli.
- Islemlerde tip uyusmazligi var.

27) Invalid subrange base type
Sayilabilir tipteki elemanlara geçerli tanimlama yapilmalidir.

28) Lover bound greater than upper bound
Alt sinir üst sinirdan büyük.

29) Ordinal type expected
Sayilabilir tip gerekir. Real, String, yapisal tip veya Pointer tipleri burada kullanilmaz.

30) Integer constant expected
Tamsayi sabiti gerekiyor. Tamsayi tipinde tanimlanmamis bir alana tam sayi konulmaktadir. Saha tanimi veya degeri düzeltmek gerekir.

31) Constant expected
Sabit gerekiyor. 'const' yazilmis fakat, sabit tanimi olmadigi zaman bu hata mesaji gerekir.

32) Integer or real constant expected
Tam sayi veya gerçek sayi sabiti gerekiyor. Sayisal tanimlanmis bir alana, sayisal bilgi disinda bir bilgi yerlestiriliyor.

33) Type indentifier expected
Tip tanimi gerekiyor. Tipin belirlenmesi gerekli olan bir degisken tipi belirtilmemis.

34) Invalid function resul type
Fonksiyon sonucunda tipi hatali. Bir fonksiyonun neticesi yapisal tip olamaz.

35) Label isentifier expacred
Etiket tanimi gerekiyor. LABEL kelimesinden sonra bir etiket tanimlamak gerekir. Ya 'LABEL' kelimesi kaldirilmali yada etiket tanimlanmalidir.

36) Begin expected
BEGIN gerekiyor. Ana program veya procedure/function' larda islemler BEGIN komutundan sonra yazilmalidir.

37) End expected
END bekleniyor.

38) Integer expression expected
Tam sayi bir ifade gerekiyor. Tam sayi kullanilmasi gerekli olan bir islemde baska tip kullanilmistir.

39) Ordinal expression expected
Sayilabilir tipte bir ifade gerekiyor. Sayilabilir tip yerine, islemde baska tip kullanilmistir.

40) Boolean expression expected
Mantiksal tipte bir ifade gerekiyor.

41) Operand types do not match operator
Islem ile isareti arasinda uyusmazlik var.

42) Error in expression
Ifadede hata var. Aritmetik bir islemde bir taraf eksik olabilir veya sabit taniminda sag taraf yazilmamistir.

43) Illegal assignment
Hatali atama yapilmis Bir atama isleminde Fonksiyonlar sag tarafta bulunamazlarr.

44) Field identifer expected
Alan tanimi gerekir. RECORD taniminda alan ismi belirtilmemistir.

47) Object file too large
Object dosya hacmi 64 K' dan daha büyük.

48) Code segment too large
Program çok büyük. Bir program veya ünite maksimum 65520 byte olabilir.Programda unitler kullanilmalidir.Unit çok büyük ise, Unit içerisinde baska unit' ler kullanilarak program kisaltilmalidir.

49) Data segment too large

Tanimlanan sabit ve degiskenlerin hacmi çok fazla. Bir programda tanimlanan sahalarin toplami maksimum 65520byte olabilir. Buna Unit' lerde kullanilan alanlar da dahildir. Hatayi gidermek için alan sayisi veya uzunluklar azaltilmalidir.

50) Do expected

DO gereklidir. WHILE, FOR komutlarinda DO gerekir.

54) Of expected

OF gereklidir. CASE komutunda, ARRAY taniminda vs.. gerekli olan OF kelimesi kullanilmistir.

55) Interface expected

INTERFACE gereklidir. UNIT içerisinde "INTERFACE" kelimesi yazilmamis.

57) Then expected

THEN gereklidir. IF komutunda THEN kelimesi yazilmamis.

58) To or downto expected

TO veya DOWNTO gereklidir. FOR komutunda TO/DOWNTO kelimelerinden biri kullanilmamis.

59) Undefined forward

FORWARD tanimlanmamis.

- Unit' in INTEFACE bölümünde tanimlanan PROCEDURE veya FUNCTION IMPLEMENTATION bölümünde tanimlanmamistir.
- PROCEDURE/FUNCTION' da FORWARD kullanildigi halde PROCEDURE/FUNCTION hiç tanimlanmamistir.

61) Invalid typecast

Geçersiz çevrim yapilmis. Bazi tipler, tanimlandigi gibi degil de uyumlu uyumlu bir tipte de kullanilabilir.

62) Division by zero

0' a bölme. Bilindigi gibi bir sayinin sifira bölünmesi tanimsizdir.

63) Invalid file type

Geçersiz dosya tipi. Örnegin, tipli dosyada Readln veya text dosyada Seek komutu kullanilmaz.

64) Connot read or write variables of this type

Bu tip için okuma veya yazma yapilamaz. Yapisal tipler ekranda dogrudan görüntülenemezler ve ekrandan alinamazlar.

66) String variable expected

String degisken gerekiyor. String tipte olmasi gereken bir degisken yerine baska tipte bir degisken kullanilmistir.

67) String expression expected
String ifade gerekli.

68) Circular Unit
Unitlerin birbirini kullanimi hatali tanimlanmis

69) Unit name mismatch
Unit ismi uyusmazligi var. Unit programinin disk üzerindeki ismi (dosya adi) ile Unit komutundan sonra yazilan isim ayni olmamalidir.

70) Unit version mismatch
Unit versiyonlari uyusmuyor. Derlenen Unit versiyonu ile, su anda kullanilan Turbo Pascal versiyonu uyusmuyor. Unit yeniden derlenmelidir. Bunun için;
✍✍ 'Compile' seçeneginden 'Make' veya 'Build' ile yeniden derleme yapilmalidir.
✍✍ TPC ile derleme yapiliyorsa, /M veya /B parametreleri kullanilmalidir.

71) Internal stack overflow
Stack segment 'te tasma var. Uzun Procedure alt programi bölün

72) Unit file format error
Unit program kullanilmasi halinde, Dos için .TPU, .TPP, Windows için .TPW uzantili dosya yok. .PAS dosyaya uygun .TPU, .TPP veya .TPW dosya olusturulmali.

73) Impelementation expected.
Unit tanimi içerisinde zorunlu olarak bulunmasi gereken IMPELEMENTATION kullanilmamis.

74) Constant and case types do not match
Sabit ve CASE' de kullanilan tipler uyusmuyor. CASE komutunda kullanilan alan tipi, test edildigi sabitlerle ayni tipte olmalidir.

75) Record variable expected
Kayit degiskeni gerekiyor. Su anda kullanilan degisken tipi RECORD tipinden olmali.

76) Constant out of range
Sabit, olmasi gereken sinirlari tasiyor.

77) File variable expected
Dosya degiskeni gerekli.

79) Integer or real expression expected
Tam sayi veya gerçek sayi ifade gerekli.

80) Label not within current block
Etiket mevcut blok disinda. Goto komutu ile program akisinin gönderilecegi etiket yanlis yerde kullanilmis.

81) Label already defined
Etiket zaten tanimlanmis. Ayni etiket ismi ikinci defa tanimlaniyor.

82) Undefined label in preceding statment
Tanimlanmamis etiket kullaniliyor. Etiket tanimi yapilmalidir.

84) UNIT expected
UNIT kelimesinin yazilmasi gerekir.

85) ";" expected
Noktali-virgül isaretinin yazilmasi gerekir.

86) ":"expected
':' isaretinin yazilmasi gerekir.

87) "," expected
Virgül isaretinin yazilmasi gerekir.

88) "(" expected
'(' Isaretinin yazilmasi gerekir.

89) ")" expected
')' isaretinin yazilmasi gerekir.

90) "=" expected
'=' isaretinin yazilmasi gerekir.

91) ":=" expected
':=' isaretinin yazilmasi gerekir.

92) "[" or "(." expected
'[' veya '(.' isaretinin yazilmasi gerekir.

93) "]" or ".)" expected
']' veya '.)' isaretinin yazilmasi gerekir.

94) "." expected
'.' isaretinin yazilmasi gerekir.

95) ".." expected
'..' isaretinin yazilmasi gerekir.

96) Too many variables
Degiskenlerin toplam uzunlugu 64 K sinirini asiyor.

97) Invalid FOR control variable
FOR komutu ile kullanilan degiskenin tipi yanlis. Örnegin; For komutunda Real degisken kullanilamaz.

98) Integer variable expected
Integer degisken kullanilmalidir.

99) Files are not allower here
Dosyalar burada kullanilamaz.

100) String length mismatch
Alfasayisal degisken için tanim blogunda verilen uzunlukla islem tanimindaki uzunluk ölçüleri uygun degil.

101) Invalid ordering of fields
Alanlarin siralamalari geçersiz.

102) String constant expected
Alfasayisal degismez bilgi tanimi yapilmamis.

103) Integer or real variable expected
Tam sayi veya ondalikli sayi degisken tanimi yapilmamis.

104) Ordinal variable expected
Sira belirleme degisken tanimi yapilmamis.

105) Inline error
Turbo Pascal Assembler kodlarinin kullanildigini belirten INLINE komut tanimi hatali.

106) Character expression expected
Karakter ifadesi yapilmamis.

107) Too many relocation items
Çok fazla yeniden konumlandirma basliklari kullanilmis.

108) Overflow in aritmetic operation
Aritmetik islemlerde tasma hatasi.

109) No enclosing For, While,0r Repeiat statement
Islem ifadeleri; FOR, WHILE, veya REPEAT deyimleriyle kapatilmaz.

112) Case constanl out of range
CASE deyimindeki degismez bilgi tanimlanan sinirlan asiyor.

113) Error in statement
Deyim kullaniminda hata var.

114) Cannot call an interrupt procedure
Program akisini bir durduran (interrupt) prosedürü çagrilamaz.

116) Must be in 8087 morle to compile this
Programda tanimlanan genisletilmis sayisal bilgi tiplerinin derlenebilmeleri için 8087 matematik islemcinin olmasi gereklidir.

117) Target address not found
Bilgilerin aktarilacagi hedef adresi bulunamadi.

118) Include files are not allowed here
INCLUDE program yazilanlar burada kullanilamaz.
119) No inherited methods are accessible here
INHERITED metod yazilimlar verildikleri satirlarda erisilemez.

120) Nil expected
NIL fonksiyonu verilmemis.

121) Invalid qualifier
Geçersiz tanimlama.

122) Invalid variable reference
Geçersiz degisken referans tanimi.

123) Too many symbols
Asiri sembol tanimlamasi.

124) Statement part too largee
Program yazilimdaki deyimler bölümü asiri uzun.

126) Files must be Var parameters
Dosyalarin tanimlarinda VAR parametreleri verilmeli.

127) Too many conditional symbols
Çok sayida kosul sembolleri kullanilmis.

128) Misplacced conditional directive
Kosul direktifi hatali yerlestirilmis.

129) ENDIF directive missing
ENDIF direktifi verilmemis.

130) Error in initial conditional defines
Baslangiç kosul tanimlarinda hatali durum.

131) Header doees not match previous defination
Baslik bir önceki tanima uymuyor.

132) Critical disk error
Disk ortaminda çok önemli hata var.

133) Cannot evaluate this expression
Tanimlanan ifade sonuçlandirilamaz.

134) Expression incorrectly terminated
hatali ifade nedeniyle islem kesildi.

135) Invalid format specifier
Geçersiz format tanimi.

136) Invalid indirect reference
Geçersiz dolayli referans.

137) Structureed variables are not allowed here
Yapisal degiskenler bu satirlarda kullanilmaz.

138) Cannot evaluate without System unit
SYSTEM unit olmadan islem yapilamaz.

139) Cannot access this symbol
Bu yazilim sekline (sembole) erisilemez.

140) Invalid, floating-point operation
Geçersiz ondalikli sayisal islemi.

141) Cannot compile overlays to memory
Bellekte OVERLAY yazilimlari derlenemez.

142) Pointer or procedural variable expected
Pointer veya prosedür degiskenleri tanimlanmamis.

143) Invalid procedure or function reference
Geçersiz prosedür veya fonksiyon referansi kullanilmis.

144) Cannot overlay this unit
Overlay olarak tanimlanan unit, overlay unit olamaz.

146) File access denided
Program veya dosya erisimi yapilamiyor.

147) Object type expeected
OBJECT bilgi tipi yapilmamis.

148) Local object types are not allowed
Çagiran programdaki lokal nesne bilgi tiplerihatali

149) Virtual expected
VIRTUAL tanimi yapilmamis.

I50) Method identifier expected
METHOD yazilim tanimi yapilmamis.

151) Virtual constructor are not allowed
VIRTUAL CONSTRUCTOR tip belirleyicisi hatali taniinlanmis.

152) Constiuctor identifier expected
CONSTRUCTOR bilgi kesim tanimi yapilmamis.

153) Destructor identifier expected
DESTRUCTOR tammi yapilmamis.

154) Fail only allowed within constructors
Sadece CONSTRUCTOR yapisal tammlannda hata olabilir. 1 SS Invalid combination of opcorle a
operandsGeçersiz TURBO ASSEMBLER yazilim opcode kombinasyonu veya islemi.

156) Memory reference expected
Bellek referansi yapilmamis.

157) Cannot add orsubtruct relocatable symbol.
Yeniden yerlestirilebilir yazilimlardaki sayisal degerlerde toplama veya çikarma yapilamaz.

158) Invalid register combination .
Geçersiz yazmaç kombinasyonu.

159) 286/287 instructions are not enabled
286/287 yapisal bilgileri etkin duruma alinamadi.

160) Invalid symbol reference
Geçersiz yazilim kod referansi.

161) Code generation error
Kod yaratma hatasi.

162) Asm expected
TURBO ASSEMBLER yazilimlarinda ASM ifadesi yapilmamis.

163) Duplicate dynamic method index
Ikinci kez tammlanan dinamik (Heap) bellek method indis elemani

164) Duplicate resource identifier
Ikinci kez tanimlanan RESOURCE WORKSHOP bilgi tanimi.

165) Duplicate or invalid export index
Ikinci kez tanimlanan veya geçersiz export (baska bir harici programlara veya dosyalara aktarilr
üzere hazirlanan bilgi yapilan) indisi.

166) Procedurie or function identifier expected
Prosedür veya fonksiyon kesim adi tanimi yapilmamis.

167) Cannot export this symbol
Tanimlanan yazilim sekli (sembol) harici bir programa veya dosyaya aktarilamaz.

168) Duplicate export name
Ikinci kez tanimlanan export bilgi tanim adi.

169) Executable file header too large
Çalistirilacak program basligi çok uzun.

Run-Time Errors Messages

Turbo/Borlan Pascal DOS modunda çalisirken asagidaki hatalarla karsilasilabilir.

1) Invalid function number
Geçersiz fonksiyon numarasi. Mevcut olamayan DOS fonksiyonu çagriliyor.

2) File not found
Reset, Append, Rename veya Erase komutlariyla kullanilan dosya degiskenine bilinmeyen dosya ismi atanmis. Dosya bulunamiyor.

3) Path not found
Reset, Rewrite, Append, Erase, ChDir, MkDir veya RmDir komutlari ile belirtilen ortam/yol (path) bulunamiyor.

4) Too many open file
Ayni anda açilan dosya sayisi fazla. Bunu engellemek için CONFIG.SYS dosyasi içerisinde tanimlanan Files=xx tanimindaki dosya sayisi arttirilmalidir.

5) File access error
- Reset veya Append komutlarinda belirtilen dosya degiskenine read-only özelligi olan directory/dosya adi atanmis.
- ReWrite komutu ile olusturulmaya çalisilan dosyanin bulundugu directory dolu veya ayni isimli read-only özelligi olan bir dosya mevcut.
- Rename komutu ile dosyaya verilecek yeni isim altinda baska bir dosyanin mevcut olmasi
- Erase komutu ile silinecek olan dosya read-only özelligine sahip.
- MkDir komutu ile var olan bir directory olusturulmaya çalisiliyor.
- RmDir komutu ile silinmeye çalisilan directory'nin içi bos degil.
- Açik olmayan bir dosyadan Read veya BlockRead komutlariyla bilgi okunmaya çalisiliyor.
- Açik olmayan bir dosyaya Write veya BlockWrite komutlariyla bilgi okunmaya çalisiliyor.

6) Invalid file handle
Bu hata geçersiz bir dosya istegi DOS sistemine iletilirse ortaya çikar.

12) Invalid file access code
Geçersiz dosya erisim kodu. Reset veya Append komutlarinda geçersiz dosya modu degeri kullanilmis.

15) Invalid drive number
GetDir veya Chdir komutlari ile geçersiz bir sürücü numarasi kullanilmis.

16) Cannot remove current directory
RmDir komutu ile aktif olan directory silinmeye çalisiliyor.

17) Cannot rename across drives
Rename komutunda belirtilen her iki dosya ayni sürücüde degildir.

I/O Errors

Giris/Çikis islemleriyle ilgili programlarda asagidaki hatalarla karsilasilabiliriz.

100) Disk read error
Read komutu ile dosya sonu isaretinden sonra okuma islemi yapiliyor.

101) Disk write error
Close, Write, Writeln veya Flush komutlari ile dolu bir disk'e islem yapilmaya çalisiliyor.

102) File not assigned
Reset, ReWrite, Append, Rename veya Erase komutlarinda kullanilan 'dosya degiskenine' dosya ismi atanmamis.

103) File not open
Üzerinde islem yapilmaya çalisilan dosya açik degil.

104) File not open for input
Text dosyasi giris için açik degil. Bu durumdaki dosyaya Read, Readln, Eof Eoln, SeekEof veya SeekEoln komutlarinin kullanilmasi sirasinda olusur.

105) File not open for output
Text dosyasi çikis için açik degil. Bu durumdaki dosyaya Write veya Writeln komutlarinin kullanimi sirasinda olusur.

106) Invalid numeric format
Bir text dosyadan Read veya readln komutu ile sayisal bilgi okunmasi sirasinda olusur

150) Disk is write protected
Disk yazmaya karsi korumali.

151) Unknown unit
Tanimsiz unit.

152) Drive not ready
Sürücü hazir degil.

154) CRC error in data
Datada hata var

156) Disk seek error
Bilgi okuma sirasinda hata olustu. Diskte belirtilen kayita konumlanilamiyor.

157) Unknown media type
DOS tarafindan taninmayan dosya. DOS uyumlu olmayan bir format sekli kullanilmis.

158) Sector not found
Disk'teki bozukluktan dolayi bilgiye erisilemiyor.

159) Printer out of paper
Printer'e kagit takili degil.

160) Device write fault
Donanim hatasindan dolayi disk'e kayit yapilamiyor.

161) Device read fault
Donanim hatasi nedeniyle bilgi okunamiyor.

162) Hardware failure
Donanim hatasi

200) Division by zero
/, div veya mod operatörleri ile yapilan bölme isleminde sayi sifira bölünüyor. Bölme isleminin paydasi sifir. Programda bölme hatasi olmasa da Win98 isletim sisteminde CRT uniti kullanilmasi halinde bu hata mesaji ile karsilasilmaktadir. Win 98 altinda Turbo Pascal ve Bp çalistirildiginda CRT uniti kullanilmamalidir.

201) Range check error
{$R+} compiler bildirisi kullanildiginda, degiskene sinirlari disinda bilgi giriliyor veya dizinli degiskenin boyutlari sinir disinda.

205) Floting point overflow
Turbo Pascalda tanimlanan degisken alanlarinin disina tasiliyor.