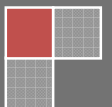


2010

Algoritma ve Programlamaya Giriş - 2

Öğr. Gör. Kürşat Volkan ÖZCAN

Bilgisayar Programcılığı 1. Sınıf öğrencilerinin programlama derslerinde kullanmaları için hazırlanmış ders notlarını içermektedir. Bu çalışma maddi bir amaç taşımamaktadır.



PROGRAMLAMA DİLLERİ

Bilgisayar çok karmaşık bir elektronik cihazdır, şimdilik bizi ilgilendiren kısım onun çalışma prensibi değil, programlama kısmıdır.

Bilgisayar öğrendiğini unutmaz, eğer iyi programlarsanız kusursuz olarak işlemleri yapar, yorulmadan hep aynı işlemi tekrar yapabilir. Programlar bilgisayarın tüm kaynaklarına erişebilir, tabii onu yazan programcı izin vermiş ise. Bilgisayar, bir konuda yorum yapamaz, yeni durumlara uyum sağlamak için çaba sarf etmez.

İnsan unutkanlıdır, hata yapabilir, yorulur ve beyninin tamamını kullanamaz, moral durumu değişebilir ve duygusal olarak etkilenir. İnsanların en büyük avantajı, yeni durumlar karşısında bocalasa bile zamanla uyum sağlayabilmesidir. Başına gelen olaylardan ders çıkartabilir. Tek başına birçok sorunu çözebilirler. Bilgisayar ise her zaman aynı tepkiyi verir, kendini geliştiremez. İnsan tarafından kontrol edilmedikçe etkinliklerini değiştirmezler.

Bilgisayar; ekonomi, bilim, mühendislik, eğitim ve askeri alanlarda yardımcı olması için üretilmiştir. Çok karmaşık formüllerin sonucunu kısa zamanda elde etmek için programlar yazılmıştır. Özellikle İkinci Dünya Savaşı veri şifreleme ve silahların hedefi daha doğru bulması gibi konular sebebi ile bilgisayarın gelişimi hızlanmıştır.

Bir programlama dilini neden öğreniriz? Bu sorunun cevabı, eğlence için, bir ihtiyacı gidermek için, kariyer için veya zekânızı kanıtlamak için olabilir. Para kazanmak her ne kadar birincil hedef gibi görünse de, eğer işinizi severek yapmıyor iseniz, ne kadar kazandığının pek önemi olmaz.

Eğer bilgisayara ne yapması gerektiğini söylemezseniz, hiçbir şey icra etmez. Bilgisayara yaptırmak istediğiniz şeyi iki şekilde gerçekleştirebilirsiniz:

- Adım adım bir program yazarak
- Uygun bir program satın alarak

İyi bir programın temel özellikleri şunlardır:

1. **Doğruluk** : Verilen görevlerin tam olarak yerine getirilmesidir.
2. **Dayanıklılık** : Beklenmedik hatalardan dolayı programın çalışması kesilmemelidir.
3. **Genişletilebilme** : İleri aşamalarda görevlerin değişikliği veya yenilerinin eklenmesi kolay olmalıdır.
4. **Basitlik** : Karmaşık tasarımlardan kaçınmak gerekir.
5. **Modülerlik** : Program kodları başka programlar içinde de kullanılabilir.
6. **Uyumluluk** : Başka bilgisayar ve sistemlerde çalışabilir.
7. **Kontrol edilebilirlik** : Hata olabilecek yerlere açıklayıcı hata mesajları konulmalıdır.
8. **Kolay kullanım** : Kullanıcıya birimi kolay olmalı ve rahat öğrenilebilir.
9. **Parçalanabilirlik** : Problemin küçük parçalara ayrılarak yazılmasıdır.

10. **Anlaşılabilirlik** : Başkasının yazdığı program elden geçirilirken rahatça okunabilmelidir.

11. **Koruma: Modüller** birbirlerine müdahale etmemelidirler.

Bilgisayar Programı İçin Neler Bilmeliyim?

Bir programı kullanmaktan çok, yazma konusunda **istekli** iseniz, zaten program yazmak için gerekli şeyye sahipsiniz demektir.

İstek: Önünüze ne kadar engel çıksa da, isteğiniz varsa öğrenirsiniz. (Kanuni olmayan bir şey ile ilgileniyorsanız, hapisanede geçirilecek zamanınız olabilir!)

Meraklılık: Bu sayede öğreneceğiniz dil size angarya gibi gelmez.

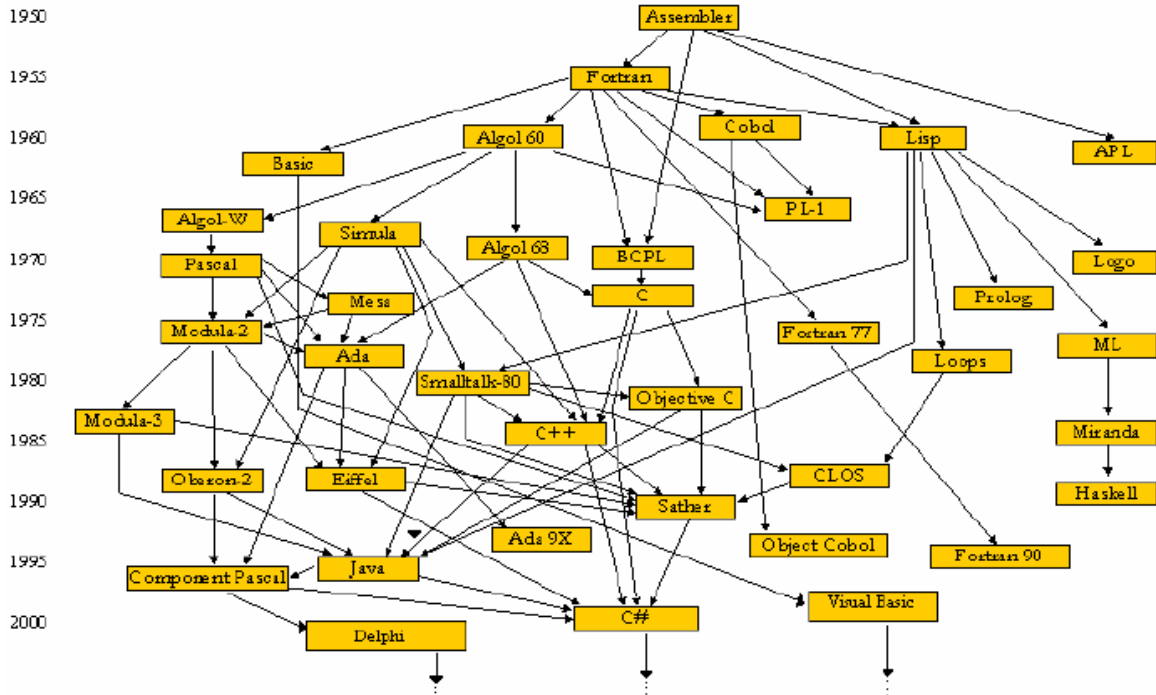
Hayal gücü: Böylece daha ilginç ve faydalı program yapabilirsiniz.

Programlama Dili

Bilgisayar Türkçe, İngilizce veya başka bir dilden anlamaz. Bilgisayarın fonksiyonel bir beyni olmadığı için insanlar komutlar yazmalıdır. Bu özel dile "**programlama dili**" denir. Komutlar bir araya gelerek "program"ı meydana getirir. Belli bir dil ile yazılmış komutlara "**kaynak kod**" da diyebiliriz.

Neden Birçok Programlama Dili Vardır?


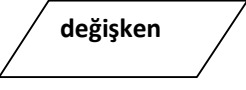

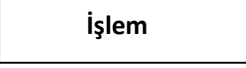
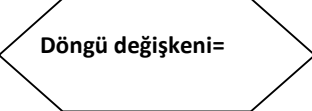
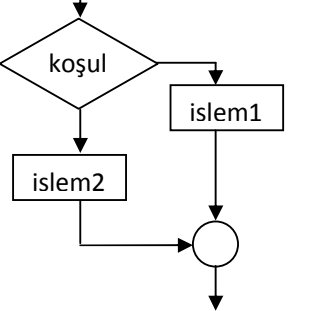
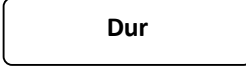
Her programlama dili özel bir amaca hizmet eder. İnsanlar farklı problemleri çözmek için değişik programlama dilleri yazmışlardır.



Bu bölümde fazla ayrıntıya inmeden Basic ve Pascal programlama dillerinin yapıları anlatılacaktır. Daha sonraki bölümlerde algoritma veya akış diyagramının yanında bu iki programlama dillerinde ki yazılımları da verilecektir.

PROGRAMLAMA DİLLERİNİN GENEL YAPILARI

Bilgisayarda program yazmak için ortaya atılan dillerden her biri farklı komut ve yapıya sahip olmasına rağmen, diller arasında büyük benzerlikler mevcuttur. Bu alt bölümde de Basic, Pascal ve C/C++ programlama dillerinin genel yapılarını incelenecektir.

Akış diyagramı şekli	Basic komutu	Pascal komutu
	<ul style="list-style-type: none"> Programla ilgili açıklamalar 	<ul style="list-style-type: none"> Programla ilgili açıklamalar Program ismi verilir Kütüphane bildirilir. Kullanılacak değişkenler ve tipleri belirlenir Bilgi tanıtım bölümü
	INPUT değişken	Readln(değişken)
	PRINT değişken	Writeln(değişken)
	...işlem...	...işlem...
	For döngü deęişkeni =başlangıç ,bitiş , adım	For döngü deęişkeni := başlangıç to bitiş do
	IF koşul THEN işlem1 ELSE işlem2	If (koşul) then işlem1 else işlem2
	END	End.

Akış diyagramı şekli	Bacis	Parcal
	FOR I=1,N,1 işlemler NEXT I	For I:=1 to N do Begin işlemler End;
	WHILE I<=N I=I+1 işlemler WEND	While (I<=N) do Begin I=I+1; işlemler End;
		I:= 0; Repeat I=I+1; işlemler Until I >=N;
	IF komuş THEN işlemler1 ELSE işlemler2	If (koşul) Then Begin işlemler1 End Else Begin işlemler2 End;
	IF koşul THEN işlemler	if (koşul) then Begin işlemler End;

A.Basic Dilinin Yapısı : 'BASIC' kelimesi , "Beginner 's All-Purpose Symbolic Instruction Code" kelimelerinin baş harflerinden türetilmiş olup ,1964 yılında Dartmouth College'de çalışmalar yapan John G.KEMENY ve Thomas E.KURTZ tarafından geliştirilmiş olan en basit programlama dillerindedir.

Basic dilinin temel özellikleri:

- En temel ve en kolay programlama dillerinden biridir.
- Karmaşık bir program yapısına sahip değildir.
- Program yazımı, herhangi bir kalıpla sınırlı değildir.

- d. Komutlar, işlem çözümüne göre sırayla, alt alta satırla yazılırlar. Komutlar, aralarına iki nokta üst üste konularak, aynı satıra da kodlanabilirler. Bir satıra, en fazla 255 karakter yazılabilir.
- e. Komutlar, büyük harflerle yazılırlar. Kullanıcı büyük harflerle komut girse bile; Basic, otomatik olarak komut harflerini büyütür.
- f. Yazılımın başında değişkenleri bildirme zorunluluğu yoktur. Değişken ismi en az 1, en fazla 40 karakter olabilir.
- g. Yazılım başında komutların bulunduğu kütüphaneleri aktifleştirmeye gerek yoktur. Çünkü Basic programı çalıştırıldığında tüm komutlar belleğe alınır.

Pascal Dilinin Yapısı : Fransız matematikçi ve filozof Blaise Pascal'ın adı verilen bu programlama dili, 1971 yılında İsviçre Federal Teknoloji Enstitüsü'nden (ETH-Zürih) Dr. Niklaus WIRTH tarafından geliştirilmiştir.

Pascal dilinin temel özellikleri

- a. Komut yazımında büyük-küçük harf ayrımı yoktur.
- b. Program yazımı ,belirli bloklarda olur.
- c. Bloklar ,”Begin “ ile “end” deyimleri ile oluşturulur.
- d. Komutlar, noktalı virgül (;) ile biter. Yalnız Begin, Do, Label, Type, Const gibi blok başlatan deyimlerden sonra kullanılmaz.
- e. Komut yazımında satır kavramı yoktur. Komut veya ifadeler düz metin şeklinde aynı veya komutları bölmek şartıyla ayrı satırlara yazılabilirler. Bir satıra en fazla 127 karakter yazılabilir.
- f. Programdaki değişkenler mutlaka tanımlanmalı (isim-bilgi tipi) dir. Değişken veya tanımlayıcı isimleri en fazla 63 karakterdir.
- g. Programda kullanılacak olan komutların bulunduğu kütüphaneler aktifleştirilmelidir/çağırılmalıdır.

PROGRAMLAMA DİLLERİNİN KULLANIMLARI**Quick Basic****BASIC Dilinde Kullanılan Değişken Türleri:**

BASIC'te işlenen bilgileri temsil eden değişkenlerin yanına yazılan sembollerle bilgi tipi belirtilir.

Değişken Türü	Belirtme Sembolü
Tamsayı	%
Tek duyarlıklı ondalık sayı	!
Çift duyarlıklı ondalık sayı	#
Karakter	\$

BASIC dilinde tamsayılar -32768 ile 32767 arasında bir değer alır. Tek ve çift duyarlıklı ondalık sayılar -1.7E38 ve 1.7E38 aralığında olabilirler. Tek duyarlıklı sayı da anlamlı basamak sayısı 7 veya daha az; çift duyarlıklı sayıda ise anlamlı basamak sayısı 8 ile 18 arasındadır. Aşağıda tek ve çift duyarlıklı sayılara örnekler verilmiştir.

Tek duyarlıklı: 23.29456, -1.234567, -2.34567E-3, 1756.345!

Çift duyarlıklı: 1.2345678901234567, -1.432D-12, 3.141592654#, 56#

Tek duyarlıklı sayılar için (!) belirteci kullanılmayabilir. Programın başında, programda kullanılacak değişkenlerin türü belirtilirse her seferinde tür belirteci kullanılmayabilir.

DEFSTR Karakter tür belirteci

DEFINT Tamsayı tür belirteci

DEFSNG Tek duyarlıklı ondalık sayı tür belirteci

DEFDBL Çift duyarlıklı ondalık sayı tür belirteci

Karakter değişkenler için değişken isminden sonra (\$) belirteci konur ve karakter tırnak (" ") içinde yazılır. Örneğin; isim\$ = "Ali" gibi.

Veri Tipleri Özet Tablosu :

Qbasic'de INTEGER, SINGLE, DOUBLE gibi veri tipleri vardır. Veriler belleğe saklanmadan önce bellekte ne kadar yer kaplayacağını belirtmek için kullanılır. Değişkenin içeriği ancak bu şekilde bellekte düzenli bir şekilde yerleşir.

Örnek

A% = 126

A değişkenin yanındaki % işareti değişkenin bir INTEGER veri tipine sahip olduğunu gösterir. Ona aktarılan bilgi bellekte 2 karakterlik yer kaplar.

Değişkenlerin veri tipini tanımlamak şart değildir. O zaman tanımlanmamış değişkenler Basic tarafından SINGLE olarak kabul edilir.

VERİ TİPİ	TANIMLAMA ŞEKİLLERİ	BELLEKTE KAPLADIĞI ALAN (Byte)	AÇIKLAMA
INTEGER	DIM A AS INTEGER DIM A% DEFINT A	2 (16 bit)	-32768 ile 32767 sayıları arasındaki TAM sayıları tutabilir(2^{15}). Daha büyük sayılar yazmaya çalıştığınızda Owerflow(Taşma) hatası verir. Küsurlu rakamlar vermeye çalıştığınızda yakın olan sayıya tamamlar. A% = 45.12 '--> 45 A% = 51.72 '--> 52
LONG	DIM A AS LONG DIM A& DEFLNG A	4 (32 bit)	-2147483648 ile 2147483647 sayıları arasındaki TAM sayıları tutabilir(2^{31}). Küsurlar INTEGERdeki gibidir.
SINGLE	DIM A AS SINGLE DIM A! DEFSNG A	4 (32 bit)	7 rakamdan oluşan küsurlu veya tam rakamları tutar. A! = 3.141145 Daha fazla rakam girildiğinde bilimsel kullanım şekline dönüştürür. A! = 12345678 ' -> 1.234568E+07 E+07 demek .(nokta) yı 7 rakam sağa kaydırılacak demektir E-07 olsaydı sola kaydırma olacaktı. A! = 50000000000 ' -> 5E+10 A! = 1 / 25000000 ' -> 4E-08
DOUBLE	DIM A AS DOUBLE DIM A# DEFDBL A	8 (64 bit)	15 rakamdan oluşan küsurlu veya tam rakamları tutar. Diğer özellikler SINGLEdaki gibidir.
STRING	DIM A AS STRING DIM A\$ DEFSTR A	Her bir karakter için 1 byte	
Kullanıcı Tanımlı		Tanımlanan genişliğe göre değişir	

Qbasicde Veri Tipleri

Qbasic diğer dillerdeki gibi tüm veri tiplerini kullanmanıza imkan vermez, fakat genellikle diğerlerine ihtiyaç duymayacaksınız. Qbasic ile kullanabileceğiniz veri tipleri INTEGER, LONG, SINGLE, DOUBLE ve STRING dir. Daha detaylı bilgi almak için menüden **HELP/Contents** 'i ve çıkan ekranda **Data Types** i tıklayın.

Veri tipleri nasıl kullanılır?

```
A% = 253
```

```
Y! = 3.141
```

```
Z& = 254144
```

A nın INTEGER olduğunu öğrenmiştik ama Y! ve Z& ! işareti SINGLE verilere sahip değişkenleri tanımlamak için, & işareti LONG verilere sahip değişkenler içindir.

Data tipi kullanımının bir başka yolu

Değişkenlerin data tiplerini tanımlamak için birkaç alternatif yöntem vardır. Bunlar:

```
DIM A AS INTEGER
```

```
DIM B AS SINGLE
```

```
DIM C AS LONG
```

Bu stil en güzel görüneni ve iyi programcıların kullandığı stildir. Tanımları düzgün yaptıktan sonra programın ileriki aşamalarında yalnızca değişken adını kullanmanız yeterlidir.

```
DEFINT A
```

```
DEFSNG B,K,N
```

```
DEFLNG C-D
```

Bu da başka bir stil Değişkenin baş harfine göre tanımlanmamış olan tüm değişkenler DEF in yanındaki değişken türünde olur. İyi bir stil sayılmaz.

```
A% = 253
```

```
Y! = 3.141
```

```
Z& = 254144
```

Bu şekilde tanımlama yaptığınızda aynı değişkeni hep aynı şekilde yazmak zorundasınız. Yani A% nin değerini değiştirmek için A = 100 kullanmak sakıncalıdır.

DİKKAT:

```
A! = 22.125
```

```
A& = 46500
```

```
A% = 255
```

```
PRINT A!, A&, A%
```

```
PRINT A
```

Kullanıcı tanımlı Veri Tipleri

Kendi veri tipinizi belirleyebilirsiniz Bu size bellekte kaplayacak olan verilerinizi kullanmanızı kolaylaştıracaktır. Örneğin kişilerin adres bilgileri üzerinde işlem yapmak istiyorsanız, kişi için kendinizin belirlediği bir veri yapısı oluşturabilirsiniz.

```
TYPE Adrestipi
```

```
Adresi AS STRING * 50
```

```
PostaKodu AS STRING * 7
```

```
Adi AS STRING * 30
```

```
Telefonu AS STRING * 18
```

```
END TYPE
```

Bu açıklamalar sanırım yetersiz olacaktır başlangıç için. Daha detaylı bilgi almak için TYPE yazısı üzerinde iken F1 e basarak yardım alabilirsiniz.

Tanımlanmış olduğumuz ADRESTIPI veri tipini kullanmak için bellekte yer açmalıyız. Bunun için

```
DIM Personel AS Adrestipi
```

komutunu kullanırız. Şimdi bu değişkenin(PERSONEL) elemanlarına değer aktaralım

```
Personel.Adresi = "Ankara Caddesi"
```

```
Personel.PostaKodu = "33522"
```

```
Personel.Adi = "Murat Velioğlu"
```

```
Personel.Telefonu = "0866-945 44 21"
```

Bu tip değişken kullanımı QBasic de yeni iseniz ve ya diğer BASIC dillere alışkanlığınız varsa garip gelecektir. Ama bu tip değişkenler diğer dillerde de kullanılmaktadır. Şu an için telaşlanmanıza gerek yoktur. Zamanla bu yapıya alışıp gerekli yerlerde rahatlıkla kullanacaksınız.

Qbasicde satır numaraları

Satır numaraları gereksizdir. Ama kullanabilirsiniz de. Kullanırsanız; numaraların birbirini takip etmesi şart değildir. QBasicde satır numaraları yerine okunurluğu kolaylaştırması için ETİKET kullanılır. Etiket de sadece GOTO veya GOSUB ile gidilecek satırlara koymak yeterlidir.

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

Yukarıdaki program kodları yerine; aynı işi yapan, satır numarası vermeden, etiket kullanarak yazılmış hali aşağıda.

```
başla:
A = A + 1
IF A > 20 THEN GOTO bitis
PRINT A
GOTO başla
bitis:
END
```

Bir satırda birden fazla komut da kullanılabilir. İki komutu ayırmak için : (2 nokta üst üste) kullanılır

```
CLS
PRINT "QBASIC"
PRINT "MERHABA DÜNYA !"
```

Üstteki ile alttaki kodlar aynı işi yapar. Farkı yoktur.

```
CLS : PRINT "QBASIC" : PRINT "MERHABA DÜNYA !"
```

BASIC'teki Komutlar

1. INPUT Komutu

Programın çalışması sırasında klavyeden programa veri girilmesini sağlar.

```
INPUT değişken1, değişken2, .....
```

```
INPUT "mesaj", değişken1, ....
```

Değişkenlere A = 45 gibi bir satır yazarak bir değer aktarabiliyoruz. Bazen program çalışırken değişkenlere programı kullanan kişinin veri aktarması istenebilir. INPUT komutu ile istediğimiz değişkenlere program çalışırken değer aktarılabilir.

INPUT kullanılırken; INPUT un ardından verilecek mesaj tırnaklar arasında yazılır sonra (,) veya (;) konulur ve klavyeden yazdıklarımızı aktaracağımız değişken ismi yazılır.

```
CLS
INPUT "ADINIZ "; AD$
INPUT "YAŞINIZ ", YAS%
PRINT "Sayın "; AD$ ; YAS% ; " yaşındasınız."
```

Yaşımızı sorduğunda rakam dışında bir şey yazarsak, bir uyarı ile aynı soruyu tekrar sorulur

Örnek:

```
.....
INPUT a
INPUT "b ve c sayılarını giriniz", b, c
d=a+b+c
.....
```

INPUT komutundan sonra girilen değişken değeri ile istenen değişken türü aynı olmalıdır.

Örnek:

```
.....
INPUT "bir sayı giriniz", a
PRINT "girilen sayının karesi="; a^2
.....
```

Bu örnekte eğer a için bir karakter girilirse hata mesajı alınır.

Örnek:

```
.....
INPUT "iki sayı giriniz", a$,b$
PRINT "girilen sayıların toplamı=";a$+b$
.....
```

Burada 5 ve 6 değerleri a\$ ve b\$ için girilirse program a ve b birer karakter değişken olduğu için onları sayı olarak değil sadece karakter olarak algılar ve 11 yerine 56 sonucunu verir. Buradaki + operatörü toplama işlemi yerine karakterleri birleştirme işlemini yapar.

2. PRINT Komutu

Ekrana bir mesaj veya bir değişkenin değerini yada bir fonksiyon yada işlemin sonucunu yazdırmak için kullanılır. PRINT yazmak yerine yalnızca ? yazın.

```
CLS
PRINT 3 * 8 + 12 ' Sonuç 36 olarak ekranda görünecek
YASI = 32
PRINT "Yaşı = " ; YASI ; " dir"
```

```
A% = 15
B! = 3.14
ADI$ = "Kürşat"
PRINT A% , B! , ADI$
```

```
SA$ = "Özcan"
PRINT ADI$ ; SA$
```

PRINT komutunda bir işlemin sonucunu da yazdıracağımızı söylemişim.

```
PRINT 12+33 ' Ekrana toplamı(45) verir
PRINT 120+38*10 ' 500 yazar. Öncelik sırası: Parantez içi, * / + - dir
PRINT 12-(80/4-23)+54/9 ' sonuç: 21 ??
```

```
PRINT 1 + 1 ' Toplam olan 2 çıkar
PRINT "1" + "1"
```

' 11 çıkar. Çift tırnak içindeki sayılar sayı olarak görülmez.

' Burada sayı olmadığı için toplama işlemi değil

```

      ' tırnaklar içindekileri birleştirme işlemi uygulanır
PRINT "QUICK" + "basic"   ' QUICKbasic
PRINT 8 + "elma"         ' !!! Hata !!!
PRINT "9" + "Ders"      ' 8Ders
PRINT 9 ; "Ders"        ' 9 Ders
PRINT 8/4 ; "Altan Kalan Ders"   ' 2 Altan Kalan Ders

```

Uygulama: Klavyeden girilen sayının karesini ve küpünü hesaplayan programı yazınız.

```

CLS
PRINT "Çıkmak için 0 yaz"
başla:
INPUT "Bir sayı yaz "; SAYI
IF SAYI = 0 THEN END
CLS
PRINT "Verdiğiniz sayı "; SAYI
PRINT SAYI; " 'nın karesi "; SAYI ^ 2; " dır."
PRINT SAYI; " 'nın küpü "; SAYI ^ 3; " dır."
GOTO başla

```

PROBLEM:

Dairenin alanını hesaplayıp ekrana yazan bir programın algoritmasını yazıp, akış çizelgesini çiziniz ve akış çizelgesine uygun Qbasic dili ile programını yazınız.

$$\text{Dairenin Alanı} = \text{Daire çapı} * \text{Daire çapı} * \text{Pi sayısı}$$

Basic'te işlenen bilgileri temsil eden değişkenleri ve tiplerini özel olarak belirtmeye gerek yoktur. Değişkenin temsil ettiği bilginin tipi, değişkenin hemen yanına yazılan özel sembollerle belirlenir.

```

10 Rem *** Sayısal Bilgi Tipleri***
20 PRINT "22/7 işleminin sonucu:"
30 A%=22/7 : PRINT "Tamsayı Bilgi Tipinde ",A%
40 A=22/7 : PRINT "Tek Duyarlıkl Bilgi Tipinde ",A
50 A#=22/7 : PRINT "Çift Duyarlıkl Bilgi Tipinde ",A#
60 END

```

Programın sonucu

```

22/7 işleminin sonucu
Tam sayı Bilgi Tipinde 3
Tek Duyarlıkl Bilgi Tipinde 3,142857
Çift Duyarlıkl Bilgi Tipinde 3,142857074737549

```

```

10 Rem *** Alfasayısal Bilgi Kullanımı ***
20 A$= "Ögr. Gör. Kürsat Volkan ÖZCAN"
30 PRINT A$
40 END

```

Programın sonucu

Ögr. Gör. Kürsat Volkan

3. IF ... THEN ... ELSE Komutu

Programlama dillerinde döngüler ve akış kontrol komutları çok sık kullanılır. Programları program yapan esas kısımlar bu komutlarla sağlanır. Qbasicde diğer programlama dillerindeki benzer yapıda döngüler ve mantıksal karşılaştırmalar yapılabilir.

Mantıksal karşılaştırma için kullanılır. Karşılaştırma işleminin sonucunda bir değer döner bu değer ya mantıksal DOĞRU dur ya da mantıksal YANLIŞ. Lise 1 deyken matematik dersinde 1 ve 0 lar ile, doğru ve yanlışlar ile işlemler yapardık. Birçok kişide ne işe yarıyor bunlar diye söylenip dururlardı. Demek ki bir gerekliliği varmış. İşte onlar burada gerekecek, isterseniz MANTIK ile ilgili kısımları bir daha gözden geçirin. :)

Mantıksal karşılaştırma için basit bir örnek:

```
IF A = 20 THEN B = 20
```

Burada A değişkeninin değeri 20 ise B nin değeri de 20 olacaktır. Eğer A nın değeri 20 dan farklı ise bu satırın hiçbir etkisi olmayacaktır.

Bir başka kullanımı:

```
A = 25
IF A > 30 THEN
    M$ = "Sayı 30 dan büyük"
ELSE
    M$ = "Sayı 30 dan küçük"
PRINT M$
```

Üstte A değişkeninin değerinin 30 dan büyük olup olmadığı kontrol ediliyor. Mantıksal karşılaştırmanın sonucunda ancak iki değer dönebilir. DOĞRU veya YANLIŞ. Doğru olması durumunda THEN den sonraki işlem yapılır, YANLIŞ olması durumunda ise ELSE den sonraki işlem. A ya 25 aktardığımız için A>30 mantıksal karşılaştırmanın sonucu YANLIŞ olacaktır. Çünkü 25, 30dan büyük değil. Bu durumda M\$ a "Sayı 30 dan küçük" değeri aktarılır.

Soru: Üstteki programı denedikten sonra A = 30 olsaydı sonuç ne olurdu? diye düşünüp cevabı bulmaya çalışın. Sonra Qbasic'de deneyerek düşündüğünüzü kontrol edin.

IF ile karşılaştırma yaptığımızda dönen değerlere göre çok sayıda komut yürüteceksek aşağıdaki yapıyı kullanırız. Bu şekilde kullanımda karşılaştırma bloğunu bitiren END IF kullanmak zorundayız.

```
IF A > 40 THEN
    'doğruysa yapılacaklar
    ....
ELSE
    'Yanlışsa yapılacaklar
    ....
END IF
```

Örnek:

```
INPUT "iki sayı giriniz", a, b
  IF a > b THEN
    PRINT "a > b"
  ELSE
    IF a = b THEN
      PRINT "a = b"
    ELSE
      PRINT "a < b"
    END IF
  END IF
END
```

Örnek :

```
INPUT "ADINIZ "; AD$
  IF AD$ = "Kursat" THEN
    PRINT "SİZİN ADINIZ Kursat"
  ELSE
    PRINT "SİZİN ADINIZ Kursat DEĞİL"
  END IF
END IF
```

İç içe IF kullanımı:

```
INPUT A
  IF A > 40 THEN
    IF A < 60 THEN
      PRINT "SAYI 40 ile 60 arasında"
    ELSE
      PRINT "SAYI 60 ya da 60dan büyük"
    END IF
  ELSE
    IF A = 40 THEN
      PRINT "SAYI 40a eşit"
    ELSE
      PRINT "SAYI 40dan küçük"
    END IF
  END IF
END IF
```

4. CLS komutu

Bu komut çalıştırıldığında daha önceden ekrana yazılmış olan yazılar silinir. Genelde programlar, ilk olarak ekranı temizleyerek, sonraki yazılacaklara temiz bir ekran hazırlar. Burada dikkat edilecek şey CLS den bir önceki zemin rengi ne ise ekran o renk ile doldurulur. CLS den önce bir renk ayarı yapılmamışsa ekran siyah renk ile doldurularak temizlenir.

```
COLOR , 4
CLS
PRINT "Merhaba"
```

5. SELECT CASE Komutu

Bir değişkenin aldığı birçok değere göre ayrı komutların çalıştırılması gereken durumlarda if yapısı yerine Select Case yapısı kullanmak daha avantajlıdır. Kullanımları ;

Select Case Değişken

```
Case durum1 :Komutlar
Case durum2, durum3: Komutlar
Case durum4 to durum7:Komutlar
Case a < durum8:Komutlar
Case b < durum9:Komutlar
```

Case Else: Komutlar

IF THEN ELSE yapısı ile bir program yazalım bunu daha sonra SELECT CASE yapısına çevirelim;

```
INPUT "1 ile 3 arasında sayı girin " ; A
IF A = 1 THEN
    PRINT "SAYI = 1"
ELSE
IF A = 2 THEN
    PRINT "SAYI = 2"
ELSE
IF A = 3 THEN
    PRINT "SAYI = 3"
ELSE
    PRINT "HATALI SAYI"
END IF
```

Bunun yerine buna benzer yapıya sahip anlaşılabilirliği ve kodlaması kolay olan SELECT CASE yapısı kullanılır.

```
INPUT "1 ile 3 arasında sayı girin " ; A
SELECT CASE A
```

```
CASE 1
    PRINT "SAYI = 1"
CASE 2
    PRINT "SAYI = 2"
CASE 3
    PRINT "SAYI = 3"
CASE ELSE
    PRINT "HATALI SAYI"
END SELECT
```

Dört işlemi yapmamızı sağlayacak kullanıcının sayı girmesini isteyip bu sayılar üzerinde istenilen 4 işlemden birini uygulayacak örnek program;

```
CLS
BaşDon:
INPUT "1. Sayıyı Giriniz";Sayı1
INPUT "2. Sayıyı Giriniz";Sayı2
    PRINT "Toplama için 1"
    PRINT "Çıkartma için 2"
    PRINT "Çarpma için 3"
    PRINT "Bölme için 4"
INPUT "işlem giriniz";islem
SELECT CASE islem
    CASE 1
        sonuc = sayı1 + sayı2
        PRINT "Toplama işleminin sonucu"; sonuc
    CASE 2
        sonuc = sayı1 - sayı2
        PRINT "Çıkartma işleminin sonucu"; sonuc
    CASE 3
        sonuc = sayı1 * sayı2
        PRINT "Çarpma işleminin sonucu"; sonuc
    CASE 4
        sonuc = sayı1 / sayı2
        PRINT "Bölme işleminin sonucu"; sonuc
    CASE ELSE
        PRINT "Yanlış Seçenek "
    END SELECT
INPUT "Tekrarlamak için (T)"; tekrar$
    IF tekrar$ = "T" THEN GOTO BaşDon
END
```

Aynı işlemi yapan programın farklı bir uygulaması konunun pekiştirilmesi maksadı ile verilmiştir.

```

CLS
BaşDon:
INPUT "1.Sayıyı Giriniz";sayi1
INPUT "2.Sayıyı Giriniz";sayi2
    PRINT "Toplama için +"
    PRINT "Çıkartma için -"
    PRINT "Çarpma için x"
    PRINT "Bolme için /"
INPUT "İşlem giriniz";islem
SELECT CASE islem
CASE 1
    Sonuc = sayi1 + sayi2
    PRINT "Toplama işlemi Sonucu = "; Sonuc
CASE2
    Sonuc = sayi1 - sayi2
    PRINT "Çıkartma işlemi Sonucu = "; Sonuc
CASE3
    Sonuc = sayi1 x sayi2
    PRINT "Çarpma işlemi Sonucu = "; Sonuc
CASE4
    Sonuc = sayi1 / sayi2
    PRINT "Bolme işlemi Sonucu = "; Sonuc
CASE ELSE
    PRINT " Yanlış Secenek Sectin Dostum"
END SELECT

```

6. GOTO Komutu

En basit döngü GOTO ile yapılan döngüdür.

```

giris:
    x = x + 2
    PRINT A
GOTO giris

```

Yukarıdaki program bir sonsuz döngü oluşturur. Durdurmak için CTRL-PAUSE tuşlarına basınız. Aşağıdaki şekilde değişiklik yaparsak döngüyü kontrol altına almış oluruz.

```

giris:
    x = x + 1
    IF x>10 THEN END

```

```
PRINT x
GOTO giriş
```

Örnek: 1 den 10 a kadar olan sayıları ekrana yazan programı oluşturalım.

```
10 CLS
20 C=1
30 PRINT C
40 C=C+1
50 IF C<=10 THEN GOTO 30 ELSE GOTO 60
60 END
```

7. FOR NEXT Komutu

Belirli sayılarda işlemlerin tekrar etmesi için kullanılır.

```
FOR N = 1 TO 25
PRINT N
NEXT N
```

1 den 25 e kadar olan sayıları yazacaktır. Her döngüde N değişkeninin değeri 1 artacaktır. Eğer ilk satırı

```
FOR N = 1 TO 25 STEP 4
```

yapacak olursak. N nin ilk değeri 1 olacak sonra her seferinde

üzerine 4 eklenerek devam edecektir.

```
FOR N = 25 TO 1 STEP -1
```

yazılacak olursa 25 den 1 e doğru N nin değeri her seferinde 1

azaltılır. NEXT in arkasına değişkeni yazmak şart değildir ama okunurluğu kolaylaştırmak için yazmakta fayda vardır. FOR ları içi içe koyarak da kullanılabilir. İçerdeki FOR un NEXT i dışarıdakinin NEXT'inden sonra gelmemesine dikkat etmelisiniz.

```
FOR N = 1 TO 10
FOR M = 1 TO 4
PRINT N * M ,
NEXT M
PRINT
NEXT N
```

FOR döngüsünden çıkma gereği olursa EXIT FOR ile çıkılabilir.

Örnek: 5 defa "İYİ İNSAN" yazan programı yazalım

```
CLS
FOR A=1 to 5
PRINT "İYİ İNSAN"
NEXT A
```

Örnek: 0 dan 10000 e kadar olan çift sayıları yazan programı yazalım.

```
CLS
FOR T=0 TO 10000 STEP 2
PRINT T
```

```
    NEXT T
END
```

Örnek : Farklı bir FOR NEXT uygulaması.

```
CLS
FOR X = 1 TO 20 STEP 2
    COLOR X, 7
    PRINT X
NEXT X
END
```

8. DO LOOP Komutu

```
DO
    PRINT A
    A = A + 1
LOOP
```

Bu da sonsuz döngü oluşturacaktır. Döngüyü kırmak, kontrol altına almak için çeşitli yollar var

EXIT DO ile döngüden çıkmak:

```
DO
    IF A > 40 THEN EXIT DO
    PRINT A
    A = A + 1
LOOP
```

WHILE kullanarak döngüyü kontrol altına almak:

```
'1 . program
CLS
DO WHILE A < 40 ' A, 40 dan küçük iken döngüye devam
    PRINT A
    A = A + 1
LOOP
```

```
'2 . program
CLS
DO
    PRINT A
    A = A + 1
```

LOOP WHILE A < 40 ' A, 40 dan küçük iken döngüye devam

Yukarıdaki iki programı denediğinizde hiçbir fark göremeyeceksiniz. Şimdi ilk satırlarına A = 45 komutunu ekleyip deneyin ve farkı anlamaya çalışın. Eğer WHILE i LOOP un yanına koyarsak döngüde şart aranmaksızın en az 1 kere döner. DO nun yanına konulursa döngü başlamadan şart kontrol edilir, şart uygun değilse döngü gerçekleşmez.

UNTIL kullanarak döngüyü kontrol altına almak:

```
CLS
```

```
DO UNTIL A > 40 ' şart DOGRU olan A KADAR dön. A , 40 dan büyük olana kadar devam
```

```
PRINT A
```

```
A = A +1
```

```
LOOP
```

WHILE için verdiğim açıklamalar bunda da geçerli UNTIL i DO'nun yanına ya da LOOP un yanına koyabiliriz. Kısaca WHILE, şartın DOĞRU olmasında; UNTIL, şartın YANLIŞ olmasında döngüye devam eder.

9.

Pascal'da Kullanılan Bilgi Tipleri: Pascal programlarındaki değişkenleri ve bilgi tiplerini, kullanmadan önce tanımlamak (bildirmek) gerekir.

1.Sayısal Bilgi Tipleri: Basic'te olduğu gibi Pascal'daki sayısal bilgi tiplerini de iki grup altında inceleyeceğiz: Tamsayı bilgi tipleri ve ondalıklı bilgi tipleri.

a.Tamsayı Bilgi Tipleri: Sadece tam kısmı bulunan sayısal bilgiler için kullanılan bu bilgi tipleri bilginin büyüklüğüne göre farklı şekillerde isimlendirilirler.

Pascal'daki tamsayı bilgi tipleri.

Bilgi Tipi	Aktarılabilecek En küçük değer	Aktarılabilecek En büyük değer	İşaret	Bellekte kapladığı Alan(byte)
Byte	0	255	Yok	1
Shortint	-128	127	Var	1
Integer	-32 768	32 767	Var	2
Word	0	65 535	Yok	2
Longint	-2 147 483 648	2 147 483 647	Var	4

b.Ondalık Sayı Bilgi Tipleri: Hem tam hem de ondalıklı kısmı bulunan sayısal bilgiler için kullanılan bu bilgi tipleri,yine sayısal bilginin büyüklüğüne,duyarlılığına göre farklı şekillerde tanımlanırlar.

Pascal'daki ondalıklı sayı bilgi tipleri.

Bilgi tipi	Alt Sınır	Üst sınır	Bellekte kapladığı Alan (byte)	Duyarlılık (hane)
Real			6	11-12
Single			4	7-8
Double			8	15-16
Extended			10	19-20
Comp			8	19-20

Sayı duyarlılıklarını daha açık bir şekilde görmek için Basic'te olduğu gibi “ π ”sayısı için örnek programı yazıp çalıştıralım.

```
{$E+, N+}
```

```
Program duyarlilik;
```

```
Uses Crt;
```

```
a:Real;  
b:ingle;  
c:Double;  
d:Extended;  
e:Comp
```

```
Begin
```

```

Clrscr; Writeln ('öndalıklı sayılarda duyarlılık : ');
Writeln ('=====');
Writeln;writeln('Pi sayının ==>');writeln
a:=Pi ;b: =Pi; c :=Pi;d:=Pi ;e:=Pi;

Writeln ('Reel tipinde duyarlılığı => ', a:1:20);
Writeln ('Single tipinde duyarlılığı => ',b:1:20);
Writeln('Double tipinde duyarlılığı => ',c:1:20);
Writeln('Extended tipinde duyarlılığı =>',d:1:20);
Writeln('Comp tipinde duyarlılığı =>',e:1:20);

Readln;

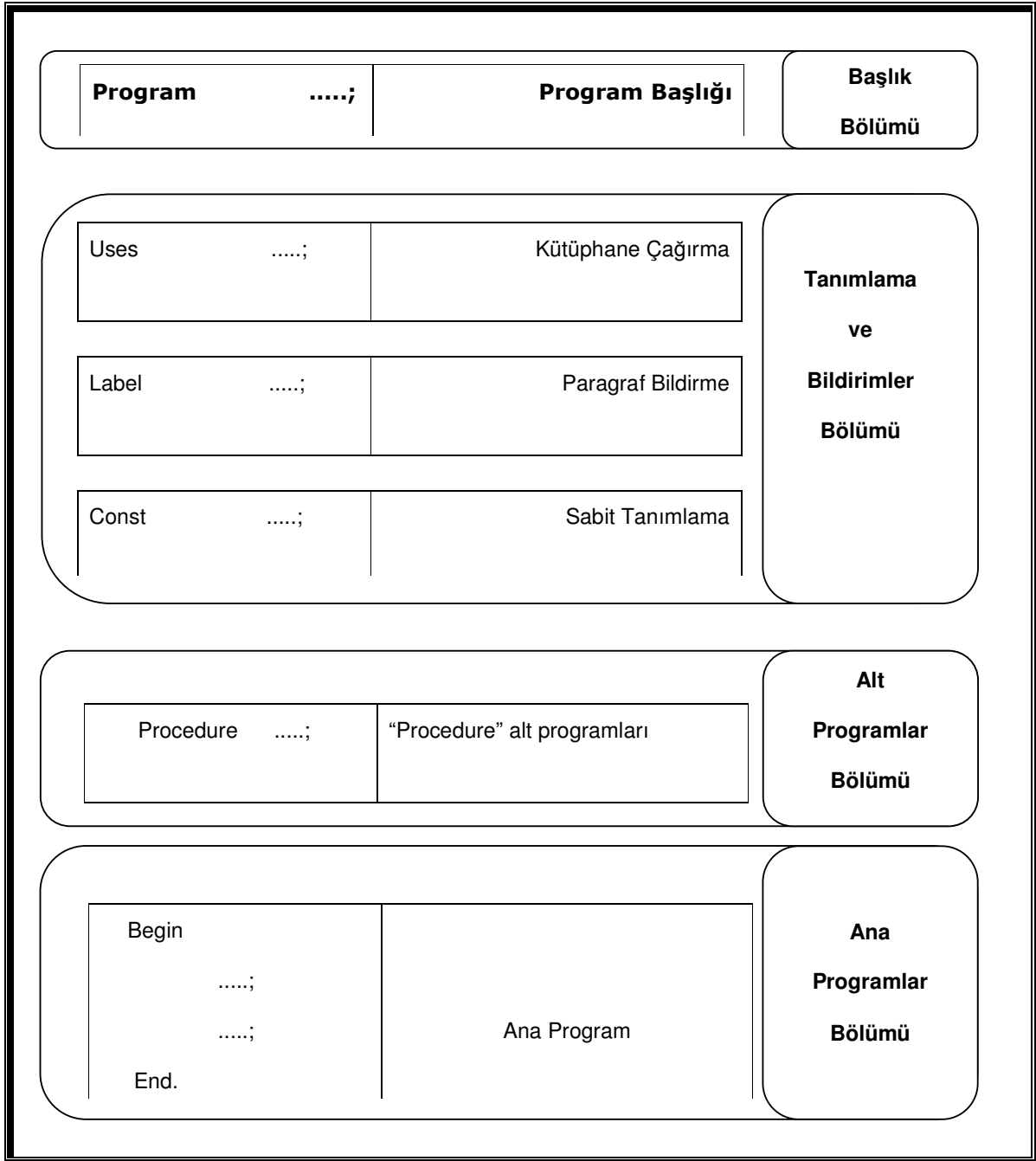
End.

```

2.Alfasayısal Bilgi Tipleri: Sayısal bilgilerde olduğu gibi, alfasayısal bilgiler de farklı bilgi tiplerinde tanımlanabilirler (Tablo-4.7) ve değişkenlere aktarılırken tek tırnak (')içinde verilirler.

Pascal'daki alfasayısal bilgi tipleri.

Bilgi tipi	Aktarılabilen en az Karakter sayısı	Aktarılabilen en fazla karakter sayısı	Bellekte kapladığı Alan (Byte)
Char	1	1	1
String	1	255 ≈ String	n+1



1.Program Başlığı

İsmlendirme kurallarına uygun olarak yazılan ve kullanımı zorunlu olmayan bu ifade, genellikle program hakkında açıklayıcı bilgi içerir.

Kullanım şekli:

Program program adı;

2.Tanımlama ve Bildirimler Bölümü

a.Uses : Yapısal dillerde(Pascal,C,C++ vb.); komutlar,belirli kütüphanelerde bulunmaktadır.Yazılım editörü çalıştırıldığında;tüm komutlar,belleğe yüklenmez.Yalnızca ,yazılımın başında çağırılan\belirtilen\aktifleştirilen kütüphanelerdeki komutlar yüklenir.Dolayısıyla yazılımda hangi komutlar kullanılacaksa,bunların bulunduğu kütüphaneler aktifleştirilmelidir.Aksi durumda; derleyici,programı çalıştırdığında yazım hataları verecektir.

Program başlığından sonraki satırda yer alan "Uses" ifadesi ,kullanılacak kütüphaneleri bildirilir.

Kullanım şekli:

Uses kütüphane-1 , kütüphane-2 , kütüphane-3;

Pascal'daki bazı kütüphaneler

Kütüphane	İçeriği
Crt	Temel ve ekran komutları
Graph	Grafik komutları
Dos	DOS işletim sistemiyle ilgili komutlar
Printer	Yazıcı ile ilgili komutlar

b.Label : Eğer programdaki işlem akışı normal seyrini izlemeyip belirli paragraflara (bloklara) dallanacaksa, gidilecek yerleri tanımlayan isimler gerekir. Programlarda ; işlem akışının geçeceği (atlayacağı/dallanacağı) bu paragraflar (bloklar) "Label" ile bildirilir ve dallanma ' Goto ' komutuyla yapılır.Basic programlama dilinde ; satır numaraları kullanıldığından paragraf tanımlamaya gerek kalmamaktadır.

Kullanım şekli

Label paragraf ismi 1 , paragraf ismi 2 ;

Paragraf isimlerinin program içinde kullanımı;

Paragraf ismi ;

.....;

}

.....;

Paragraf

şeklindedir.

Program Paragraf_tanımlama;

Uses Crt ;

Label geri;

Var

a: integer ;

Begin

ClrScr ;

geri : Write (' iki haneli bir tamsayı giriniz : ') ;

 Readln (a) ; a := abs (a) ;

 if ((a<10) or (a>99)) then goto geri ;

 Writeln;

 Write (' Girilen sayının onlar basamağı : ' ,

 A div 10 , ' ve birler basamağı : ' , a mod 10);

 Readln ;

End .

İki haneli bir tamsayı giriniz : 5

İki haneli bir tamsayı giriniz : 125

İki haneli bir tamsayı giriniz : 67

Girilen sayının onlar basamağı : 6 ve birler basamağı : 7

c. Const : Birçok programda , sabit değerler kullanılır . Pascal programları için kullanılacak olan sayısal veya alfasayısal sabitler , " Const " bloğunda tanımlanırlar. Sabit tanımlama iki şekilde yapılabilir.

Tip belirtmeden : Bilgi tipi verilmeden yapılan bu sabit atamada ; atanan sabit bilgi , program boyunca **değiştirilemez.**

Kullanım şekilleri

Const sabit adı 1 = sayısal değer ;

Sabit adı 1 = alfasayısal ifade ' ;

Tip belirtilerek : bilgi tipi belirtilerek yapılan sabit tanımlamada ; aktarılan sabit bilgi ; "**başlangıç değeri**" görevinde olup programın herhangi bir yerinde değiştirilebilir.

Kullanım şekilleri

Const sabit adı 1 : bilgi tipi = sayısal değer ;

```

Sabit adı 2 : bilgi tipi = alfasayısal değer ' ' ;

Program sabit_tanımlama ;

Uses Crt ;

Const pi_sayisi=3.14 ;

      r : integer=2 ;

Var

      Alan : Real ;

Begin

      ClrScr ; Alan :=pi_sayisi *r*r ;

      Writeln ( ' varsayılan alan : ', alan :0:3 ) ;

      Writeln ; write ( yeni_yarıçap : ) ; Readln (r) ;

      Alan :=pi_sayisi *r*r ;

      write ( 'hesaplanan alan : ', alan :0:3 ) ; Readln ;

End

```

Varsayılan alan : 12.560

Yeni yarıçap : 3

Hesaplanan alan : 28.260

d. Type : Pascal standart bilgi tipleri dışında , programcılar kendi tiplerini veya veri alanlarını tanımladıkları yerdire.

Kullanım şekli :

Type özel bilgi tipi adı=standart tipler cinsinden tanıımı ;

Örneğin ; Type

Kelime 15 =string [15] ;

Dizi 20 =Array [1..20] of Integer ;

Veri /kayıt alanı tanımlama : birçok bilgiyi içeren veri /kayıt alanları da "type " bloğunda tanımlanırlar.

Kullanım şekli :

Örneğin ;

Type öğrenci =Record

Adsoyad:String [25] ;

Numara:Longint ;

Vize , Final , Ort : Integer ;

End

Yukarıdaki örnekte 'ogrenci ' adlı veri alanı tanımlanmıştır. Bu veri alanı içerisinde " String " bilgi tipinde ' Adsoyad ' , " Longint " tipinde ' numara ' ve " Integer " tipinde ' Vize ' ; ' Final ' ve ' Ort ' değişkenleri vardır. Bu alt alanlara değer aktarmak için ; bu veri tipinde tanımlanan değişken ile alt alan adı arasına nokta konulur.

Veri / kayıt alanı

Ad Soyad	Numara	Vize	Final	Ort
-----------------	---------------	-------------	--------------	------------