

3. Bölüm

TEMEL GİRİŞ / ÇIKIŞ KOMUTLARI

- Write, Writeln*
 - Read, Readln*
 - Formatlı Çıkış*
 - Veri Kontrolü (Compiler Bildirileri)*
-

WRITELN

Amaç: Karakter dizilerini (string), sayısal sabitleri ya da değişken içerisindeki değerleri ekrana veya yazıcıya göndermek için kullanılır.

Dizilim : **WRITELN** (*çıkış_listesi*) ;

Örnek : **Writeln** (' Merhaba', isim , '.') ;
 Writeln (X , ' TL.');
 Writeln (Lst,'Tuzla Teknik Lisesi');
 Writeln ;

Çıkış_listesi içerisine yazılanlar ekrana aynı satırda yan yana yazılır. *Çıkış_listesi* yazıldıktan sonra imleç bir alt satırın başına döner ve bundan sonra yazılacaklar varsa bu satıra yazılır. *Çıkış_listesi* olmayan bir **Writeln** komutu, imleci bir alt satırın sol başına getirir. **Lst** bildirisi kullanılarak *çıkış_listesi* istenirse yazıcıdan alınabilir.

WRITE

Amaç : Karakter dizilerini (string), sayısal sabitleri ya da değişken içerisindeki değerleri ekrana veya yazıcıya göndermek için kullanılır.


Dizilim : **WRITE** (*çıkış_listesi*) ;

Örnek : **Write** ('İsminizi giriniz >') ;

Çıkış_listesi'nde bulunan değişkenin değeri veya sabit ekrana yazdırılır. Ekrana yazım işi bittikten sonra imleç alt satırın başına dönmez, satırın sonunda kalır. Eğer daha sonra ekrana bir şey yazdırılmak istenirse, aynı satırdan itibaren

devam eder. Çıkış listesi olmayan bir Write komutu ekrana bir şey yazdırmayacağı gibi imleç pozisyonunu da değiştirmeyeceğinden kullanımı etkisiz olur. Bu nedenle kullanılmaz.

NOT: `Ln` kısaltması İngilizce'deki Line (satır) kelimesinin kısaltılmış şeklidir. `Writeln` ile `Write` arasındaki farkı ayırmak için `Ln` harflerinin imleci bir alt satıra taşıdığını hatırlınızda tutunuz.


 Şimdiye kadar öğrendiklerinizi denemeniz için aşağıdaki Pascal programını, gerekli kurallarına göre yazınız.

Örnek 4: `Writeln` komutu ile yapılan string işlemleri. Programı "MERHABA1.PAS" adıyla kaydediniz.


```
program merhaba_1 ;
begin
    writeln('Merhaba Dünya') ;
end.
```


Programı yazıp "Merhaba1" adı ile kaydettikten sonra derleyelim:

 **F10** tuşuna basıp **Compile** menüsünü seçiniz.


 **Compile Alt-F9** komutunu ışıklandırıp **enter** tuşuna basınız.


Yukarda bahsettiğimiz işlemi için daha kısa bir yöntem vardır. Bu da, imleç Edit penceresindeyken **Alt-F9** tuşlarına birlikte basmaktır. Böylece sizin source programınız makine diline çevrilir. Yani, disketinizde **MERHABA1.EXE** adında bir dosya elde edersiniz. Ancak bu dosya disketinize kaydedilmemiş olabilir. Bunu sağlamak için;

 **F10** tuşuna basıp **Compile** menüsünü seçiniz. Daha sonra da **Destination** seçeneğine bakınız. Eğer bu seçeneğin yanında **Disk** kelimesini görüyorsanız, .EXE uzantılı object programınız diskete kaydolmuştur. Şayet aynı yerde **Memory** görüyorsanız, bunu **Disk** olarak değiştirmek için bu seçeneği ışıklandırıp **enter** tuşuna basınız.

 Şimdi, kaynak programı **Alt-F9** tuşlarına basarak yeniden derleyip çalıştırın.

Programlarımızı şimdiye kadar hep Pascal editörü yardımıyla çalıştırdık. Elimizde şu anda ".EXE" uzantılı dosya bulunduğuna göre bunu DOS ortamından da çalıştırabiliriz.

 Geçici olarak DOS ortamına geçmek için **F10**, **File**, **DOS shell** seçeneklerini sırayla seçiniz.

 Şimdi DOS ortamında bulunmaktasınız. **EXE** uzantılı programınızı görmek için **DIR** yazınız.

☞ Programı çalıştırmak için **MERHABA1** yazıp **enter** tuşuna basınız.

Program çıktısı:

```
Merhaba Dünya
```

☞ Tekrar Turbo Pascal ortamına dönmek için **EXIT** yazıp **enter** tuşuna basınız.

Görüldüğü gibi programlarımızı istersek Pascal menüsünde, istersek DOS ortamından çalıştırabiliyoruz. Programlarımız üzerinde çalışırken sık sık DOS ortamına dönmemiz gerekmez. Ancak programımızı tamamen geliştirip son şekline getirdiyse, DOS ortamına **Alt+X** ile çıkıp orada çalıştırmanız uygundur. Çünkü, Turbo Pascal programı bellekten silinecek ve sizin programınız için daha fazla yer kalacaktır.

Örnek 5 : Writeln komutu ile yapılan stringlerin ekrana yazdırılması işlemleri. Programı "MERHABA2.PAS" adıyla sabit diskinize kaydediniz.

```
program Merhaba_2 ;
begin
    writeln('Merhaba');
    Writeln ;
    writeln(' Dünya');
end.
```

Program çıktısı:

```
Merhaba

Dünya
```

Yukarıdaki örnekte görüldüğü gibi ilk Writeln' le **Merhaba** yazıldıktan sonra imleç bir alt satırın başına döndü ve İkinci Writeln' le imleç bir alt satırın başına daha getirildi. Yani bir satır boş bırakıldı. **Dünya** karakter dizisi (string) ise üçüncü satıra yazıldı. Şimdi write komutunun etkisini görmek için **begin** ile **end** arasındaki satırları biraz değiştirerek yazalım.

```
Write ( 'Merhaba' ) ;
Writeln ( ' Dünya' ) ;
```

Programı çalıştırdığınızda çıktısı aşağıdaki gibi olacaktır. Çünkü, **write** komutu ile **Merhaba** yazıldıktan sonra imleç aynı satırda kalır. **writeln** komutu **Dünya** kelimesini yine aynı satıra önceki karakter dizisinin (string'in) hemen yanına yazdırır.

```
Merhaba Dünya
```

Açıklamaların Yazılması

Amaç: Pascal programları yazılırken üzerinde sonradan yapılacak iyileştirmeler veya eklemeler gerekebilir. Zaman geçtikçe program satırların ne yaptığı genellikle unutulur. Bu nedenle hatırlamayı kolaylaştırmak amacıyla satır aralıklarına açıklamalar yazılması gerekir.

Dizilim: { Açıklama } veya (* Açıklama *)

Örnek : { Bu bir açıklamadır. }
 (* Bu da bir açıklamadır. *)
 { Bir açıklama (* başka bir açıklamanın *) içinde
 yapılmıştır. }

Sağa büyük parantez "{" bir açıklama satırının başlangıcını, sola büyük parantez "}" ise açıklamanın bittiğini belirtir. Aynı işi "(*" ve "*)" sembolleriyle de yapılabilir. Açıklamalar programın anlaşılabilirliğini artıran satırlardır. Pascal tarafından dikkate alınmazlar, yani makine diline çevrilmezler.

NOT: Turbo Pascal'da iç içe açıklamalar da yapılabilir. Ancak ilk açıklama "{" sembolüyle başladıysa içteki açıklama "(*" sembolüyle başlamalıdır.

Writeln Komutu İle Yapılan Diğer İşlemler

Bazı programlama dillerinde olduğu gibi Pascal'da da "+" sembolü karakter dizilerini birleştirmede kullanılır. **writeln** veya **write** komutu içerisinde aritmetik işlemler direkt olarak yapılabilir. Şimdi bunlara ait örnekler yapalım.

Örnek 6 : Karakter dizisi birleştirme ve karakter dizisi atama. Programı "MERHABA3.PAS" adıyla har diskinize kaydediniz.

```

program Merhaba_3 ;
var
Mer_dun : string[20] ; { Mer_dun değişkeni için 20
                        karakterlik yer ayrıldı}
begin
                        (* ana program başlangıcı *)
    Mer_dun := 'Merhaba' + ' Dünya' ;
    Writeln (Mer_dun) ;
    Mer_dun := 'Bu gün Pascal öğrenmeye başladım' ) ;
    Writeln (Mer_dun) ;
    Writeln ('Turbo ' + 'Pascal');
    Writeln ( '1' + '2' + '3' ) ;
    Writeln ( 'A' + 'B' + 'C' ) ;
    Writeln ( 2 * 5 ) ;
    Writeln ( 15 - 3 ) ;
end.                        (* ana program sonu *)

```

Program çıktısı:

```

Merhaba Dünya
Bu gün Pascal öğrenm
Turbo Pascal
123
ABC
10
12

```

Bu programda **Mer_dun** değişkeni için bellekte 20 karakterlik yer ayrılması gerektiğini **string[20]** tanımlayıcısı ile belirtildi. İlk atama işleminde toplam karakter uzunluğu 20'yi aşmadığından tamamı ekrana yazdırılabildi. İkinci atamada ise karakter uzunluğu 20'yi aştığından ilk 20 karakter **Mer_dun** değişkenine yerleştirilebildi, bu nedenle cümle ekranda eksik olarak görüldü. **Writeln** komutu içerisinde karakter dizisi (string) birleştirmesi yapılabildiği gibi aritmetik hesaplamalar da yapılabilir.

READLN

Amaç : Çevre aygıtlardan (klavye , disk ...) değişkenler içerisine veri girmek için kullanılır.

Dizilim : **READLN** (*Değişken_listesi*) ;

Örnek : **Readln** (*isim*) ;
Readln (*soyad* , *yas*) ;
Readln ;

Program `readln` satırına gelince durur ve dışarıdan veri girilmesini bekler. *Değişken_listesi* içerisinde bulunan değişkenler `char`, `string` veya `sayısal tipler` olması gerekir. Diğer standart tip değişkenler ve özel veri tipi değişkenleri bu listede yer alamazlar. Programı kullanan kişi listedeki her bir değişken için bir veri girmelidir. Değişken isimleri arasında mutlaka virgül ',' bulunmalıdır.

Klavyeden girilen verilerin sırası değişken listesindeki sıraya uygun olmalıdır. Aksi halde veriler yanlış değişken içerisine girilmiş olur. `Readln` komutu ile değişken içerisine girilecek karakter sayısı aksi belirtilmedikçe 127 karakter uzunluğundadır ve klavyeden girilen her karakter, girildiği anda ekranda görülür. Veriyi değişken içerisine girmek için muhakkak enter tuşuna basmak gerekir. Enter tuşuna basıldığında ise imleç bir alt satırın başına döner. `Readln` procedure içerisine yazılan birden fazla string değişkeni içerisine veri girmek mümkün değildir. İlk veri girildikten sonra enter tuşuna basılacağından `Readln` komutunun değer okuma işlemi sona erdirilmiş olur.

`Readln` komutu tek başına kullanıldığında, program durur, ancak değişken olmadığından değişkene veri aktarılması söz konusu değildir. Bu kullanım şekli genellikle, kullanıcının enter tuşuna basmasını beklemek için tercih edilir.

Örnek 7: `Readln` komutu ile yapılan işlemler. Programı sabit diskinize "READLN.PAS" adıyla kaydediniz.

```
Program readln_komutu;
var
  st1 , st2 : String;  ch1 , ch2 : char ;
begin
  Write('Bir satırlık bir cümle giriniz: ');
  Readln(st1);
  Writeln('Sizin yazdığınız: ',st1);
  Writeln('Devam için <Enter> tuşuna basınız...');
  Readln;
  Write('Yanyana iki karakter yazıp<Enter>tuşuna basınız: ');
  Readln(ch1,ch2);
  Writeln('Önce ',ch1,' sonra da ',ch2,' yazdınız. ');
  Writeln('Devam için <Enter> tuşuna basınız...');
  Readln;
  Write('Adınızı ve soyadınızı yazıp Entere basınız: ');
  Readln(st1,st2);
  Writeln('Adınız: ',st1:20,'. ');
  Writeln('Soyadınız: ',st2:20,'. ');
  Writeln('Programdan çıkış için <Enter> tuşuna basınız.. ');
  Readln;
end.
```

Program çıktısı:

```
Bir satırlık bir cümle giriniz: Pascal öğrenmek çok kolay
Sizin yazdığınız: Pascal öğrenmek çok kolay
Devam için <Enter> tuşuna basınız...
Yanyana iki karakter yazıp <Enter> tuşuna basınız: tp
Önce t sonra da p yazdınız.
Devam için <Enter> tuşuna basınız...
Adınızı ve soyadınızı yazıp Entere basınız: Orhan Altınbaşak
Adınız:      Orhan Altınbaşak.
Soyadınız:
Programdan çıkış için <Enter> tuşuna basınız..
```

Bu programda ilk `readln` deyimi ile `st1` değişkeni içerisine bir cümle girilmiştir. Kelimeler arasına bırakılan boşluklar da karakter sayılarak cümlenin tamamı yazdırılmıştır. 5, 10 ve 16. satırlardaki `Readln` deyimleri, sadece enter tuşuna basılmasını beklemek için kullanılmıştır. 3. `Readln` deyimi, yan yana yazılan ilk karakter olan `t`' yi `ch1` değişkenine, `p`' yi ise `ch2` değişkenine atamıştır. Şayet, karakterler girilirken aralarına boşluk bırakılsaydı ("`t p`" gibi), ilk karakter yine `t`, ikinci karakter ise boşluk karakteri (`` ``) olacaktı.

Beşinci `Readln` satırında ise iki string değişkeni içerisine değer girmeye çalışılmış, ancak başarılı olunamamıştır. Çünkü aralarına boşluk bırakılarak yazılan isim ve soyad "`st1`" değişkeni içerisine enter tuşuyla girildikten sonra `Readln` deyiminin değer okuma işlemi sona erdirilmiştir. Eğer, isim ve soyad verileri `st1` ve `st2` gibi iki ayrı değişkene girilmek isteniyorsa, `Readln(st1);` ve `readln(st2);` deyimlerini ayrı ayrı yazmak gerekirdi.

NOT: Burada şu soru akla gelebilir. `Readln` deyimi ile sadece bir değişkene mi değer girilebilir ? Cevap: Hayır, daha fazla değişkene de değer girilebilir. Ancak değişken tiplerinin birbirinden farklı olması ya da iki string değişkenin aynı `Readln` deyimi içerisinde olmaması gerekir.

READ

Amaç : Çevre aygıtlardan (klavye , disk ...) değişkenler içerisine veri girmek için kullanılır.

Dizilim : `READ (değişkenler_listesi) ;`

Örnek : `Read (isim) ;`
`Read (yas,isim) ;`
`Read ;`

Read ve **Readln** komutları arasında imleç pozisyonunun "ln" ekiyle alt satırın başına getirilmesi dışında hiçbir farklılık yoktur. **Read** komutunun işletilmesinden sonra, imleç son girilen karakterin hemen yanında kalır. **Readln** komutu için geçerli olan kurallar **Read** komutu için de geçerlidir. **Read** komutu parametresiz olarak kullanıldığında ise **Readln** komutu gibi enter tuşuna basılması için programı durdurur ve değer atama söz konusu değildir.

Örnek 8 : **Read** komutu ile yapılan işlemler. Programı sabit diskinize "READ.PAS" adıyla kaydediniz.

```
Program read_procedure;
var
    ch1,ch2,ch3 : char ;
begin
    Writeln ('Ad ve soyadınızın baş harflerini ');
    Write('aralarına nokta koyarak yazınız: ');
    Read(ch1,ch2,ch3);
    Writeln('İsminizin kısaltılmış şekli: ',ch1,ch2,ch3);
    Readln;
end.
```

Program çıktısı:

```
Ad ve soyadınızın baş harflerini
aralarına nokta koyarak yazınız: O.A
İsminizin kısaltılmış şekli: O.A
```

Bu programda **Read** deyimi ile **ch1** değişkenine oharfi, **ch2** değişkenine "." karakteri ve **ch3** değişkenine ise **A** harfi girildi. En sona yazılan **Readln** satırıyla programın durdurulması ve enter tuşuna basılmasının beklenilmesi amaçlanmıştı. Ancak, bu işlemin atlandığı yani program **writeln** deyimi ile **O.A** yazdıktan sonra Pascal menüsüne kendiliğinden döndüğü gözlemlendi. Bunun sebebi ise, üç değişkenin içerisine üç adet karakter **O.A** okunduktan sonra imleç aynı satırın sonunda kaldı ve **Readln** deyimi ise satır sonunda basılan enter tuşunu okumak zorunda bırakıldı.

FORMATLI ÇIKIŞ

Program içerisindeki değişkenlerin içerisindeki sayısal veya string değerler ekran veya kağıda yazarken özel formatta yazılabilir. Bu işlem verilerin daha kolay okunmasını ve tertipli görülmesini sağlar. Her verinin kendine has formatlanma biçimi vardır. Şimdi bunları sırasıyla görelim.

Tamsayıların (Integer) Formatlanması

Dizilim : *Intr_değ_ismi : alan*

Örnek : **x:4 sayi:3**

Integer sayıların ekran, printer ya da diske yazılışları düzenlenirken değişken veya sabitten sonra " : " yazılır, daha sonra da sayı için ayrılacak karakter genişliği yazılır (Örneğin **x:6** , **yas:4** , **dog_yili:7** gibi..) Eğer, alan genişliği değişken içerisindeki sayının rakam adedinden daha küçükse, Pascal yeterli alanı otomatik olarak açar.

Örneğin, **x:=456** ise, **writeln(x:8);** deyimini ile sayının ekrana yazılışı aşağıdaki gibi olur.

						4	5	6
--	--	--	--	--	--	---	---	---

Örnek 9: Integer sayıların formatlanması ile ilgili örnekler. Programı sabit diskinize "INTFORMA.PAS" adıyla kaydediniz.

```
program int_formatlama;
const
  X = 234 ;
  Y = -234 ;
  Alan = 6 ;
begin
  Writeln(X:4);
  Writeln(X:5);
  Writeln(X:6);
  Writeln(Y:4);
  Writeln(Y:5);
  Writeln(Y:6);
  Writeln(X:Alan);
  Writeln(X:1);
end.
```

Program çıktısı:

```

234
 234
 234
-234
-234
-234
 234
234

```

Ondalıklı (Real) Sayıların Formatlanması

Dizilim : *Değişken veya sabit : tamsayı alanı : ondalık alanı*

Örnek : **X:4:2 , Toplam:9:3**

Ondalıklı sayıları formatlı olarak yazmak için önce tam sayı alanı ve sonra da ondalık kısım için alan ayırmak gerekir. Ondalık hane alanı sıfır olan ondalıklı sayıların ondalık kısmı yazılmaz.

Örneğin **x:=34.56** ise, **writeln(x:8:3)** deyimi ile sayının ekrana yazılışı aşağıdaki gibi olur.

			3	4	.	5	6	0
--	--	--	---	---	---	---	---	---

Örneğin **x:=23.456** ise, **writeln(x:10:2)** deyimi ile sayının ekrana yazılışı aşağıdaki gibi olur.

						2	3	.	4	6
--	--	--	--	--	--	---	---	---	---	---

Görüldüğü gibi, ondalık hane için ayrılan alan az olduğunda, sayı yuvarlatıldıktan sonra yazılır.

Program İçerisinde Metin Kopyalama İşlemi

10. örneği yazmak için, **writeln** komutunu program içerisinde 12 defa yazmak zorunda kalacaksınız. Turbo Pascal editöründe bir bölgeyi alıp başka bir yere kopyalama imkanınız vardır. Böylece program yazım hızını artırıp sıkıcı tekrarlamalardan kurtulabilirsiniz. Şimdi metin kopyalama uygulamasını 10. Örneği yazarken yapalım. **writeln(MyPi)** ; deyimini yazdıktan sonra aşağıdaki işlemleri yapınız:

- ☞ İmleci `Writeln` deyimindeki `w` harfinin üzerine getiriniz.
- ☞ Shift tuşunu basılı tutup **sağ yön tuşunu** kullanarak bu satırdaki en son harfe kadar olan bölgeyi ışıklandırınız.
- ☞ **Ctrl+Ins** tuşlarına basarak ışıklı bölgeyi belleğe alınız.
- ☞ Belleğe aldığınız ışıklı bölgeyi kopyalamak istediğiniz yer olan bir alt satıra kopyalamak için imleci alt satırın başına getirip, **Shift+Ins** tuşlarına basınız.
- ☞ Alttaki diğer satırlara da kopyalama için aynı işlemi tekrar ediniz.
- ☞ Kopyalama işlemleri bittikten sonra ışıklı bölgeyi gizlemek için **Ctrl** tuşu basılı olarak tutulurken önce **K**, daha sonra da **H** tuşlarına basınız. (Ctrl-K H tuş kombinasyonu).

Örnek 10: Ondalıklı (real) sayıların formatlanmasına örnek program. Programı "REALFORM.PAS" adıyla kaydediniz.

```
program ondalikli_sayilar;  
uses crt ;  
const  
    Mypi = 3.14159 ;  
var  
    Vergi , Toplam : real ;  
begin  
    clrscr ;  
    Vergi := -0.006;  
    Toplam := 1.23456789;  
    Writeln(Mypi) ;  
    Writeln(Mypi:5:2) ;  
    Writeln(Mypi:3:2) ;  
    Writeln(Mypi:8:5) ;  
    Writeln(Mypi:9) ;  
    Writeln(Mypi:4:0) ;  
    Writeln(Mypi:3:2) ;  
    Writeln(Vergi:4:2) ;  
    Writeln(Vergi:9) ;  
    Writeln(Vergi:8:5) ;  
    Writeln(Vergi:8:0) ;  
    Writeln(Toplam) ;  
end.
```

Program çıktısı:

```
3.1415900000E+00
3.14
3.14
3.14159
3.14E+00
3
3.14
-0.01
-6.00E+03
-0.00600
-0
1.2345678900E+00
```

Ondalıklı sayıların formatlı çıkışlarını inceleyerek yukarıda verilen farklı örneklerle karşılaştırınız.

Karakterlerin (Char) Formatlanması

Dizilim: *Karakter_değişken_ismi : alan;*

Örnek: `Writeln(c:6);` `Writeln('A':4);`

Karakter kendisine ayrılan alan içerisinde sağa yanaşık olarak yazılır. Ayrılan alan 1'den az olsa bile alan otomatik olarak en küçük alan olarak 1 alınır.

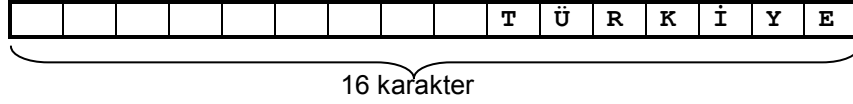
Örneğin, `a:='K'` ise, `Writeln(a:7)` deyimini ile karakterin ekranda görünümü aşağıdaki gibi olur.

						K
--	--	--	--	--	--	---

Örneğin `Writeln('B':4)` deyiminin ekrandaki çıktısı aşağıdaki gibi olur.

			B
--	--	--	---

Verilen bu örneğin ekrandaki görüntüsünü şematik olarak gösterelim:



Örnek 12 : Karakter dizisi (string) formatlamayla ilgili örnekler. Programı "STRFORMA.PAS" adıyla sabit diskinize kaydediniz.

```

program string_formatlama ;
const
    st = 'Atatürk' ;      { stringin uzunluğu 7 karakterdir. }
begin
    Writeln (st:12);      { solda 5 boşluk }
    Writeln (st:9);       { solda 2 boşluk }
    Writeln (st:7);       { boşluksuz }
    Writeln (st:2);       { boşluksuz }
    Writeln (st:0);       { boşluksuz }
end.

```

Program çıktısı:

```

    Atatürk
  Atatürk
Atatürk
Atatürk
Atatürk

```

Mantıksal (Boolean) Verilerin Formatlanması

Dizilim: `Boolean_değişken_ismi:alan;`

Örnek: `Writeln(Durum:8);`

Yukarıdaki örnekte `Durum:=FALSE` ise deyimmin ekrandaki çıktısı aşağıdaki gibidir.

			F	A	L	S	E
--	--	--	---	---	---	---	---

`Writeln('A' < 'Z':10)` deyiminin çıktısı aşağıdaki gibi olur.

						T	R	U	E
--	--	--	--	--	--	---	---	---	---

Örnek 13: Mantıksal (boolean) tipi değişkenlerle yapılan işlemler. Programı "BOOLEAN.PAS" adıyla kaydediniz.

```

Program boolean_deg;
uses crt;
const
    bo1 = False ;
    bo2 = True ;
var
    a, b, c, durum : boolean ;
    d , e : integer;
begin
    clrscr;
    writeln(True) ;
    writeln(False:10) ;
    Writeln(bo1>bo2:10) ;
    Writeln(bo2>bo1) ;
    a:=True;
    b:=False;
    c:=a>b;
    writeln(c) ;
    d:=10;    e:=15;
    writeln(d>e:10) ;
    Writeln(True>False) ;
    writeln(d=10) ;
    durum := (d>11) or (e<20) ;
    writeln(durum) ;
    durum := ('A'>'a') and ('M'<'P') ;
    writeln(durum:10) ;
    writeln('ANKARA'>'ANKA') ;
    writeln(maxint>32768:10) ;
end.

```

Program çıktısı:

```

TRUE
    FALSE
    FALSE
TRUE
TRUE
    FALSE
TRUE
TRUE
TRUE
    FALSE
TRUE
    FALSE

```

Yukarıdaki programa dikkat edecek olursanız, **true**, **false** ve **maxint** sabitleri **Const** bloğunda tanımlanmadığı halde, program içerisinde kullanıldığı zaman hata mesajı verilmedi. Çünkü bu sabitler Pascal derleyicisi tarafından tanımlanmıştır. **Maxint** sabiti derleyici tarafından önceden tanımlanmıştır ve içerisinde 32767 sayısı vardır.

VERİ KONTROLÜ (DERLEYİCİ BİLDİRİLERİ)

Derleyici (Compiler) Bildirisi Tanımlamak

Amaç: Turbo Pascal derleyicisinin bazı özelliklerini, derleyici (compiler) bildirileri ile kontrol altında tutmak için kullanılır.

Dizilim : `{ $ Bildiri_harfi +/- }`

Örnek :
`{ $B- }`
`{ $R+ }`
`{ $ INCLUDE .PAS }`
`{ $B- , $R+ , $I }`

Derleyici bildirileri, program içerisinde aktif ya da pasif duruma geçirilmek istenilen herhangi bir yerde kullanılabilir.

derleyici bildirilerinin yazılışı "{ " sembolü ile başlar. Daha sonra "\$ " sembolünü takiben bildiri harfi (R, I, B gibi..) ve "+" ya da "-" sembolü yazılır. En sonunda ise "}" sembolü yazılarak bildirim sona erdirilir. Yazılan bu semboller arasında boşluk bırakılmamalıdır. Bildiri harfini takiben yazılan "+" sembolü sağlanan özelliğin aktif duruma, "-" sembolü ise pasif duruma geçmesi için kullanılır. Yukarıdaki örneklerde görüldüğü gibi "{ " ve "}" sembolleri içerisinde birden fazla bildiri yazmak mümkündür.

Her derleyici bildirisinin bir kabul edilen (default) değeri vardır. Yani, aksi belirtilmedikçe bazı derleyici özellikleri aktif, bazıları ise pasif durumdadır.

NOT : Derleyici bildirileri Turbo Pascal'ın versiyonlarına göre değişiklik gösterirler. Örneğin `{ $T+ }` bildirisi 4. versiyonda kullanıldığı halde, 5. ve 6. versiyonlarda kullanılmaz. `{ $B+ }` bildirisinin 4. versiyonda derleyiciye verdiği özellik 5. ve 6. versiyonlarda verdiği özellik arasında çok büyük fark vardır. Bu nedenle biz burada sadece 5. ve 6. versiyonlarda aynı işlevi gören bildirileri vereceğiz. Diğer bildiriler için Turbo Pascal başvuru kitaplarına bakınız. Sadece forward bildirisi yukarıda açıklanan diziliminden farklı olarak yazılır.

FORWARD bildirisi

Programcı tarafından tanımlanan alt programlar birbirini çağırarak kullandığında bazı problemlerle karşılaşılabilir. Eğer önceden tanımlanmış bir A programı daha sonra tanımlanan bir B programı içinden çağırılırsa, bu doğru bir yöntemdir. Ancak bunun tersi yapılırsa problemle karşılaşılır. Çünkü A programı derlenirken B alt programı derleyici tarafından henüz bilinmiyor. Bu nedenle B alt programının ismine, A alt programında rastlandığında hata mesajı ile karşılaşılır.

Forward bildirisi, tanımlanmadan önce kullanılmak istenen bir alt programın daha sonra tanımlanacağını, fakat gerçek tanım yapılmaya kadar olan program kesiminde, bu alt programın isminin kullanılabilceğini belirtir.

Aşağıdaki örnekte **B** fonksiyonu kendisinden daha önce yazılan **A** prosedürü içerisinde kullanılabilmesi için **B** fonksiyonu parametreleri ile birlikte önceden yazılır ve sonuna **Forward** yazılır. Fonksiyon ikinci defa tanımlanırken parametrelerinin tekrar yazılmasına gerek yoktur.

```
Function Bfonk (x,y:integer):integer; forward;
Procedure Aproc (var m,n :real);
begin
    c:=Bfonk (c,d);
    .
end;
function Bfonk;
begin
    .
    .
end;
```

R bildirisi

Amaç : İndis değerleri üzerinde veya bayt ve dizisel tipteki değerler üzerinde sınır kontrolü yapmak için kullanılır.

Dizilim : { $\$R+$ }

Kabul edilen değeri (default) **R-** dir. Yani, değişkenler içerisine atanan verilerin sınır kontrolü yapılmaz. Bu da programın hatalı sonuç üretmesine yol açar. Bu bildirinin etkisini gösteren program daha ilerdeki bölümlerde ele alınacaktır.

I bildirisi

Amaç : Giriş/çıkış işlemleri sırasında yapılacak hataların kontrol edilmesi için kullanılır.

Dizilim : `{ $I + }`

Bu bildirinin amacı Pascal'da hata yakalama rutinlerini aktif duruma geçirmektir. Normal olarak kabul edilen durumu (default) `I+` dir. Pascal'da bir giriş/çıkış hatası yapılırsa, örneğin tamsayı tipi bir değişken içerisine karakter dizisi girilirse, program akışı hemen o anda kesilir ve DOS ortamına dönülür. Bu durum profesyonelce hazırlanmış bir program için çok büyük bir hatadır. Eğer `I` bildirisi aktif `{ $I+ }` yapılırsa, böyle bir hata sonucu programın akışı kesilmez. Program içerisinde kullanılan `IOresult` fonksiyonuna yapılan hatanın kod numarası (Run-time error) aktarılır. Programcı `IOresult` fonksiyonuna aktarılan bu kod numarasını değerlendirerek, programı kullanan kişiye gerekli uyarıyı vererek hatanın düzeltilmesini sağlar. Bu bildiri ile ilgili örnekler "`IOresult`" fonksiyonu açıklanırken verilmiştir inceleyiniz.

I (INCLUDE DOSYA) Bildirisi

Amaç : Bir program içerisinde, derlenmemiş başka Pascal programlarını kullanabilmeyi sağlar.

Dizilim : `{ $I program_ismi }`

Örnek : `{ $I Pencere.pas }`
`{ $I Menu }`

Turbo Pascal editöründe yazılabilecek programların maksimum uzunluğu 64 Kb. dir. Geniş kapsamlı programlar için bu kapasite yetersiz kaldığında, dahil edilen (include) dosyalar kullanılırlar. Dahil edilen (include) dosyalarının kütüphane (unit) dosyalarından farkı, derlenmemiş olmalarıdır.

Include dosya bildirisi, editör içerisinde yazılmayan ve çevre bellekte bulunan bir programı derleme esnasında editördeki program içerisine dahil etmek için kullanılır. Include bildirisi içerisinde program isminin uzantısı yazılmadığında uzantı `.PAS` olarak kabul edilir. Bir include dosya içerisinde, başka bir include dosyası kullanılamaz.

Include dosyaların kullanılmasının en büyük faydası, disket veya sabit diske kaydedilmiş olan `.PAS` uzantılı diğer kaynak programların birer kütüphane programları gibi istenildiğinde kullanılabilmesidir.

NOT : Bu derleyici bildirisi yukarıda açıklanan giriş/çıkış işlemlerini kontrol eden `{ $I+ }` bildirisi ile karıştırılmamalıdır.

Örnek 14: Dahil edilen (include) dosya olarak kullanılan bir program. Bu program içerisine ana programda aktarılan iki sayının toplamı ve farkı bulunur ve sonuç tekrar ana programa transfer edilir. Bu programı "INC_DOSY.PAS" adıyla sabit diskinize kaydediniz.

```
Procedure top_cik(a,b:integer;var c,d:integer);
begin
    c:=a+b;
    d:=a-b;
end;
```

Örnek 15: Yukarıda yazılan programı, şimdi yazacağınız programa dahil (include) eden ve klavyeden girilen iki sayının toplamını ve farkını bulan program. Programı yazdıktan sonra "INCLUDE.PAS" adıyla sabit diskinize kaydediniz.

```
Program ana_prog;
var
    a,b,c,d:integer;
    {$I inc_dosy}
begin
    write('1.Sayıyı giriniz >');readln(a);
    write('2.Sayıyı giriniz >');readln(b);
    top_cik(a,b,c,d);
    writeln('Sayıların toplamı=',c);
    writeln('Sayıların farkı=',d);
end.
```

Program çıktısı:

```
1. Sayıyı giriniz >5
2. Sayıyı giriniz >3
Sayıların toplamı=8
Sayıların farkı=2
```

ÖNEMLİ NOT: Sabit diskinize program olarak kaydettiğiniz ".PAS" uzantılı kaynak programlarınızı prosedür haline getirip, istediğiniz bir program içerisine dahil (include) edebilirsiniz. Böylece, önceden yaptığınız program parçalarını, yeni yaptığınız programlar içerisine yazmak zorunda kalmazsınız. Bu sayede, çok uzun olan programlar bilgisayar belleğinde daha az yer kaplarlar.

