

BİLGİSAYAR PROGRAMLAMAYA GİRİŞ

(ALGORİTMALAR, AKIŞ ŞEMALARI ve PASCAL PROGRAMLAMA DİLİ)

DERS NOTLARI

Hazırlayanlar:

Yar. Doç. Dr. Tarık ERFİDAN

Yar. Doç. Dr. Birol ARİFOĞLU

Kocaeli Üniversitesi Elektrik Mühendisliği Bölümü

PROGRAMLAMAYA GİRİŞ

Bilgisayar Programlarının geliştirilmesinde aşağıdaki adımlar gerçekleştirilmelidir:

1. Problemin Tanımlanması : Problemi çözmek için elimizde hangi veriler var ve neler isteniyor,
2. Çözüm methodunun geliştirilmesi (Analiz): Giriş(ler) ve çıkış(lar) belirlenir,
3. Çözüm methodunun adımlandırılması (Tasarım) : Algoritmanın oluşturulması,
 - Ana adımlar belirlenir
 - Ana adımlar belirlendikten sonra her adım detaylandırılır
4. Programın Kodlanması : Algoritması oluşturulan problem, seçilen bir bilgisayar proglama dilinin yapısalılığı ve kuralları ile kodlanır.
5. Programın testi : Yazılan programdaki hatalar giderilir ve farklı girdiler için test edilip, denir.

Algoritma

Algoritma, verilen herhangi bir sorunun çözümüne ulaşmak için uygulanması gerekli adımların hiç bir yoruma yer vermeksizin açık, düzenli ve sıralı bir şekilde söz ve yazı ile ifadesidir. Algoritmayı oluşturan adımlar özellikle basit ve açık olarak sıralandırılmalıdır. Çözüm için kullanılacak bilgilerin nereden alınacağı, nerede saklanacağı ve çözümün program kullanıcılarına nasıl sunulacağı algoritma adımları arasında belirtilmelidir.

Örnek 1: Bir öğrencinin evden çıkıp okula gitme eylemine ait algoritmayı yazınız.

ÖRNEK : Girilecek 2 sayıyı toplayıp, sonucu ekrana yazdıran programı geliştiriniz.

1. Problem tanımlandı,
2. Çözüm methodu : 2 sayı (girişler), toplam sonucu (çıkış),
3. Algoritma :
 1. Başla
 2. Klavyeden girilen 2 sayıyı oku (ekrandan)
 3. Toplamını bul
 4. Toplam sonucu ekrana yaz
 5. Bitir

Sözcüklerin ortaya çıkaracağı yanlış anlaşılmaları ortadan kaldırmak için semboller ve matematiksel kısaltmalar kullanmak uygundur.





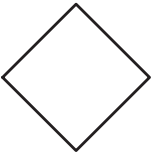

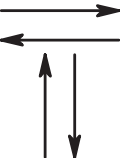
DEĞİŞKEN : Bir problemin çözümünde tanımlanan bir bilgi, farklı adımlarda farklı değerler alıyorsa buna değişken denir.

Birinci sayı : a, ikinci sayı :b, toplam:t olmak üzere, algoritma

1. Başla
2. a,b 'yi oku,
3. $t=a+b$ 'yi hesapla,
4. t yi ekrana yaz,
5. Bitir

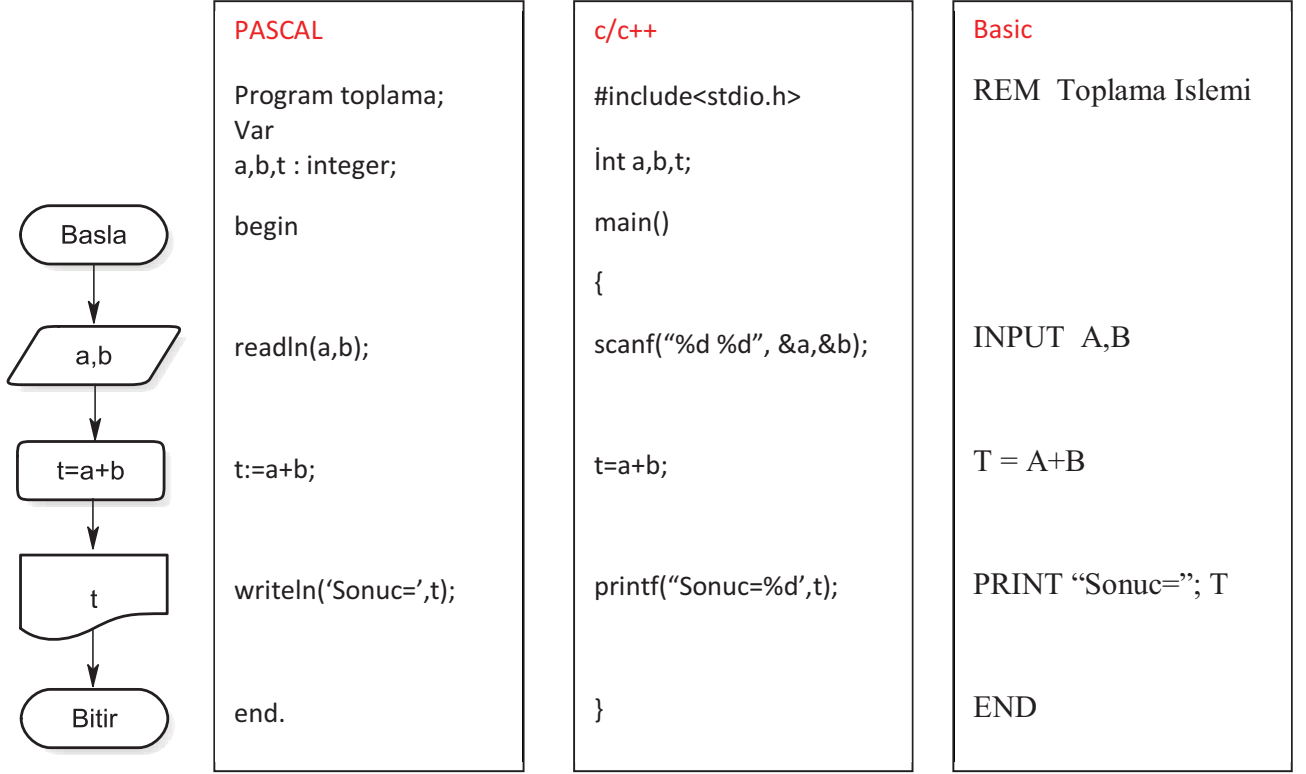
Akış Diyagramları : Programın genel görünümünün, planının, akış yönünün ve problemin çözümündeki adımların şematik gösterimine denir.

Akış diyagramlarında kullanılan semboller ve anlamları;

<u>Simge</u>	<u>Anlami</u>
	Basla, bitir
	Veri girişi
	Aritmetik işlemler ve atama işlemleri
	Çıktı
	Karar verme
	Tekrarlama
	Akış yönleri

Atama : Bir aritmetik ifadenin yada bir deęerin bir deęiřkene tanımlanmasına (yükleneğine) atama denir. Örneęin; toplam=a-5 veya a=2 gibi.

Örneęimize geri dönüp akıř diyagramını çizelim ve 2 farklı programlama dili için kaynak kodlarını yazalım;



Son adım test ve doęrulama.

PASCAL PROGRAMLAMA DİLİ

Bir pascal programı en basit anlamda 3 kısımdan oluşur .

Program Başlığı

Tanımlama Bloğu

BEGIN

.....icra bloğu

END.

Program Başlığı :

Programın yapacağı iş ile ilgili bilgi vermek için kullanılır. Kullanılması mecbur değildir.

Kurallar,

1. Sayı veya özel karakterlerle ile başlayamaz,
2. Birden fazla kelime kullanılacaksa boşluk bırakılamaz, boşluk yerine alt çizgi (_) kullanılabilir,
3. Türkçe karakter kullanılamaz,
4. En fazla 127 karakter kullanılabilir,
5. Satır sonunda noktalı virgül (;) olacak.

Kullanımı  Program program adı;

Ornekler:

Program sayi_toplama;

Program sayi topl; //HATALI

Program sayi2topla;

Program 2sayi_topla; // HATALI

Tanımlama Bloğu :

Tanımlama bloğu aşağıdaki alt blokları içerir.

1. Kullanılacak komutları içeren kütüphaneler (Unit, uses),
2. Etiketler (label),
3. Sabitler (const),
4. Değişkenler (var),
5. Tip Tanımlama (Type).

Sabit Tanımlama Bloğu

Sabit değerlerin tanımlanmasında kullanılır. Sabit tanımlanması yapılırken “ = “ operatörü kullanılır, iki farklı kullanım tipi vardır,

1. Sayısal tiptekiler: doğrudan tanımlanır,
2. Sayısal olmayan (alfanumerik) tiptekiler : Tek tırnak içinde tanımlanır.

Sabit tanımlanması const anahtar kelimesinin altında yapılır.

Kullanımı

Const

Sabit adi = sayısal değer;

Sabit adi = ‘alfanumerik deger’;

Örnek:

Program sabit_kullanimi;

Const

a = 10;

bolum = ‘elektrik’;

Const

GecmeNotu=50;

VizeKatsayisi=0.4;

FinalKatsayisi=0.6;

Const

Pi=3.14159;

HataMesaji=’Yanlis Deger’;

Sayi=sqr(3/2)+Pi;

Islem=sayi-10*5+23;

Değişken Tanımlama Bloğu

Değişkenleri tanımlamak için kullanılır. Matematiksel bir işlemin sonucunda değişkenin tamsayı, real veya alfanumerik tipte olacağı mutlaka belirtilmelidir. Programcılık açısından işlem sonuçlarının tahmin edilmesi önemlidir. Program boyutlarının küçük olması için en az bellek alanı kullanan değişken tipi tanımlanmalıdır.

Pascal Programlama Dilindeki değişken tipleri :

1. Sayısal,
2. Alfanumerik,
3. Boolean.

SAYISAL DEĞİŞKENLER :

A) Tamsayı Tiptekiler

Tip	Hafıza Kullanımı (Byte)	Aralık
SHORTINT	1	-128 to +127
BYTE	1	0 to 255
INTEGER	2	-32768 to +32767
WORD	2	0 to 65535
LONGINT	4	-2,147,483,648 to 2,147,483,647

B) Ondalıklı Tiptekiler

Tip	Hafıza Kullanımı (Byte)	Aralık
REAL	6	2.94E-39 to 1.7E+38
*SINGLE	4	0.71E-45 to 3.4E+38
*DOUBLE	8	4.94E-324 to 1.79E+308
*EXTENDED	10	3.3E-4932 to 1.18E+4932
*COMP	8	±9.2E+18

*Kullanılabilmeleri için programın başına {\$N+} direktifi yazılmalıdır.

Real tipteki veriler nokta içeren ve üstel formda ifade edilen sayılardır. Üst ifadesi olarak E harfi kullanılmaktadır. Örnek : $2.45E + 12 = 2.45 \cdot 10^{12}$

ALFANUMERİK DEĞİŞKENLER :

Sayısal anlamı olmayan değişken veya sabitleri tanımlamak için kullanılır. Sayısal olmayan karakterler 2 şekilde tanımlanır:

1. Bir karakter (char) veya birden çok karakterden (string=karakter dizisi) oluşuyorsa iki apostrof işareti (' ') arasında yazılırlar. Yazılan bilginin uzunluğu en fazla 255 karakter olabilir.
2. Bir karakterlik bilgiler başında diyez (#) işareti ve ASCII kodları ile de tanımlanabilirler. #65='A', #13=enter, .. gibi (Ek A)

Tip	Hafıza Kullanımı (Byte)
Char	1
String	256
String[n]	n+1

String ile en fazla 255 karakterlik bilgi kullanılabilir. String kelimesinden sonra karakter dizisinin uzunluğu köşeli parantez ([]) içinde tanımlanır, tanımlanmassa derleyici bunu 255 kabul eder.

BOOLEAN DEĞİŞKENLER :

Sadece TRUE (doğru) veya FALSE (yanlış) değeri alır.

Tip	Hafıza Kullanımı (Byte)
Boolean	1

Değişkenlerin Tanımlanması :

Değişken tanımlamaları tanımlama bloğunda VAR anahtar kelimesinin altında yapılır.

VAR

değişkenadi1, değişkenadi2, ... : tip1;

değişkenadi3, değişkenadi4,....: tip2;

•
•
•

Örnek:

Program tanımlamalar;

Const

a = 10;

bolum = 'elektrik';

enter = #13;

Var

b, c, d : integer;

ucret : real;

Adres, isim : string[20];

Tus : char;

Devammi : boolean;

i, j, k : integer;

Atama işlemleri :

Pascal programlama dilinde atama işlemleri için := operatörü kullanılır.

Kullanımı

değişkenadi := matematiksel işlem , sayı, char/string;

Sağ taraftaki sayı veya işlem sonucu sol taraftaki değişken adına atanır. Tiplerin birbirine uygun olması gerekir.

GİRİŞ/ÇIKIŞ KOMUTLARI

Klavyeden bilgi girişi yapmak için kullanılan komutlar giriş komutları (Read/ReadLn), ekrana veya dosyaya veri göndermek için kullanılan komutlar çıkış komutları (Write/WriteLn) olarak isimlendirilir.

ÇIKIŞ KOMUTLARI (Write / WriteLn) :

Write komutu kullanıldığında imleç (cursor = kursor) ekrana yazdırılan sayısal veya alfanumerik ifadelerden hemen sonra aynı satırda yer alır, WriteLn (Write Line) komutu kullanıldığında ise imleç ekrana yazdırılan sayısal veya alfanumerik ifadelerden sonraki satıra (bir alt satıra) gider. WriteLn tek başına kullanıldığında imleç bulunduğu noktadan bir alt satır başına geçer.

Alfanumerik İfadeler :

Kullanımı:

Write ('Alfanumerik ifade');

Write ('Alfanumerik ifade1', 'Alfanumerik ifade2', ...);

Write (Alfanumerik değişken1, Alfanumerik değişken2,..);

Formatlı Yazdırma : Alfanumerik ifade veya değişken alan uzunluğunda belirtilen sayı kadar sağa dayalı olarak yazdırılır.

Write (Alfanumerik değişken : Alan uzunluğu);

Write ('Alfanumerik ifade' : Alan uzunluğu);

Örnek :

Var

a,b : string[15];

Begin

a:='Kocaeli';

b:='Universitesi';

Write ('Pascal');

Write (' öğreniyorum');

WriteLn ;

WriteLn ('Pascal');

WriteLn ('öğreniyorum');

WriteLn (a,b);

WriteLn (a, ' ',b);

WriteLn (a:10);

End .

Monitör



Tamsayılar:

Kullanımı:

Write (değişken);

Write (değişken1, değişken2, ..);

Formatlı Yazdırma : Sayı veya değişken alan uzunluğunda belirtilen sayı kadar sağa dayalı olarak yazdırılır.

Write (değişken : Alan uzunluğu);

Değişken yerine direk olarak sayıda kullanılabilir.

Örnek :

Var a,b,c : Integer;

Begin

a := 5 ; b := 7 ; c := - 9 ;

Write (a) ;

Write (5 + 4) ;

Write (3 * b) ;

Write (a * b + c + 4) ;

Writeln ;

Write (a : 5) ;

End .

Monitör



Real Sayılar :

Kullanımı:

Write (değişken);

Write (değişken1, değişken2, ..);

Formatlı Yazdırma : Real sayıların üstsüz formatta yazdırılması için kullanılır.

WriteLn (Değişken : Genişlik : Ondalık);

Değişken ; Ekranaya yazdırılacak real değişkeni veya sayıyı ifade eder.

Genişlik ; Değişkenin sağa dayalı olarak kaç karakterlik alana yazılacağını belirtir.

Ondalık ; Reel değişkenin noktadan sonra kaç basamakla ifade edileceğini belirtir.

a:7:2 için,

real a sayısını için 7 karakterlik yer ayırır. Bu alanın son 2 si ondalıklı kısım için 3. nokta için, geriye kalan kısım (4) ise tam kısım için kullanılır.

Sayının tam kısmı ayrılan alana sığmıyorsa, tam kısım kuralı yazılır, kesirli kısım ayrılan alana sığmıyorsa yuvarlama yapılır.

Örnek :

```
Var d: real;
begin
d := 123.456789 ;
WriteLn('1234567890');
WriteLn('*****');
WriteLn(d:9:4); WriteLn(d:9:2);
WriteLn(d:9:0); WriteLn(d:6:1);
WriteLn(d:5:0);
WriteLn(d:3:5);
WriteLn(d:0:0);
WriteLn(d);
End.
```

Monitör



Karışık Kullanım :

Yukarıda anlatılan formatların karışık kullanım şeklidir.

Örnek :

```
Var a,b,c : Integer;
Begin
  a := 5 ; b := 7 ; c := - 9 ;
  WriteLn ( a , b );
  WriteLn ( a , ' ' , b );
  WriteLn ( ' C'nin degeri = ' , c );
  WriteLn ( ' Toplam = ' , a + b + c , ' dir.' );
End .
```

Monitör



Örnek :

```
Var a,b,c : Integer;
Begin
  a := 5 ; b := 7 ; c := - 9 ;
  WriteLn ( a );
  WriteLn ( 5 + 4 );
  WriteLn ( 3 * b );
  WriteLn ( a * b + c + 4 );
  WriteLn ( ' a ' );
  WriteLn ( ' 5 + 4 ' );
  WriteLn ( ' 3 * b ' );
  WriteLn ( a:7 );
End .
```

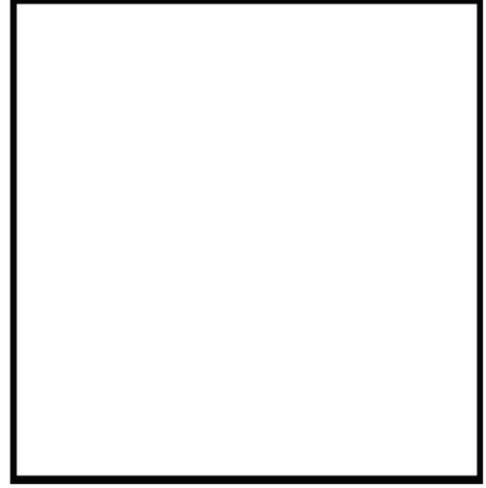
Monitör



Örnek :

```
Var a,b,c : Integer;
Begin
  a := 5 ; b := 7 ; c := - 9 ;
  WriteLn ( a );
  Write ( ' 5 + 5 ' );
  Write ( ' = 5 ' ); WriteLn;
  Write ( ' 3 * b' );
  Write ( ' = ' , a * b + 3 , ' dir. ' );
  WriteLn;
  WriteLn ( 'a = ' , a );
  a := 2 * a ;
  WriteLn ( 'a = ' , a );
  Write ( ' 5 + 4' );
  WriteLn ( 3 * a );
  WriteLn ( 'b = ' , c );
End .
```

Monitör



GİRİŞ KOMUTLARI (Read / ReadLn) :

Read ve ReadLn komutları ekrandan veri girişi için kullanılır. ReadLn komutu kullanıldığı zaman Her satır için değerler okunduktan sonra en az 1 kez enter tusuna basılması gerekir. Read komutunda ise birden fazla read komutu olsa bile her satırdan sonra enter tuşuna basılmasına gerek yoktur ancak ekrandan veri girişinin tamamlandığının bildirilmesi için en az bir kez enter tusuna basılması gerekir.

Read ve readln komutları kullanıldığında:

1. Giriş (ler) için program akışı durur,
2. Giriş (ler) klavye ile yazılır ve ekrandan gözlenir,
3. Giriş (ler) tamamlanınca enter tuşuna basılır,
4. Birden fazla sayısal değer giriliyorsa mutlaka aralarında boşluk bırakılır,
5. Giriş (ler) Alfanumerik ise aralarında boşluk bırakılmaz ve tanımlama bloğunda belirtilen uzunlukta olmalıdır, aşılırsa fazla olan kısım atılır.
6. ReadLn komutu tek başına kullanıldığında, program o satırda enter tuşuna basılıncaya kadar bekler.

Kullanımı:

read (değişken);

read (değişken1, değişken2, ..);

Örnek :

```
Var A,B,C,D,E,F,G : Integer ;
Begin
  Read (A,B,C);
  Read (D,E);
  Read (F,G);
End.
```

```
1 2 3 4 5 6 7 ←
8 9 10 11 12 ←
13 14 15 16 ←
```

A= B= C= D= E= F= G=

Örnek :

```
Var A,B,C,D,E,F,G : Integer ;
Begin
  ReadLn (A,B,C);
  ReadLn (D,E);
  ReadLn (F,G);
End.
```

```
1 2 3 4 5 6 7 ←
8 9 10 11 12 ←
13 14 15 16 ←
```

A= B= C= D= E= F= G=

Örnek :

```
Var A,B,C,D,E,F,G : Integer ;  
Begin  
  Read (A,B,C);  
  Read (D,E);  
  Read (F,G);  
End.
```

```
1 5 ←  
8 15 ←  
17 -8 20 ←  
-4 6 ←
```

A= B= C= D= E= F= G=

Örnek :

```
Var A,B,C,D,E,F,G : Integer ;  
Begin  
  ReadLn (A,B,C);  
  ReadLn (D,E);  
  ReadLn (F,G);  
End.
```

```
1 5 ←  
8 15 ←  
17 19 -8 20 ←  
-4 6 ←
```

A= B= C= D= E= F= G=

Örnek :

```
Var a,b,c : Integer ;  
    c1,c2,c3 : Char;  
Begin  
  WriteLn('Uc harf giriniz =');  
  ReadLn (c1,c2,c3);  
  WriteLn('Uc tam sayi giriniz =');  
  ReadLn (a,b,c);
```

```
Uc harf giriniz =  
dev ←  
Uc tam sayi giriniz =  
456 ←  
4 56 7 ←
```

End.

c1= c2= c3= a= b= c=

Pascal Programlama Dilindeki Matematiksel Operatörler

Matematiksel işlemlerin yapılmasını sağlayan operatörlerdir.

Operatör	+	-	*	/	mod	Div
Anlamı	Toplama	Çıkarma	Çarpma	Bölme	Mod bulma	Tam Bölme

Bu operatörlerin dışında özel operatörler de mevcuttur. Bunlar;

Div (divide) operatörü : Bölme sonucunda oluşan sonucun tam sayı kısmını gösterir ve sonuç integer tipindedir.

Örnek :

```
16 Div 3 = 5
15 Div 3 = 5
17 Div 3 = 5
3 Div 15 = 0
```

```
Var a,b,x : integer;
```

```
Begin
```

```
  a := 9; b := 4;
```

```
  x := a div b ;
```

```
  WriteLn( a, ' sayisinin ', b, ' sayısına tam bolumu = ', x, ' dir.');
```

```
End.
```

Monitör



Mod operatörü : Bölme sonucunda oluşan sonucun kalan kısmını gösterir ve sonuç integer tipindedir.

Örnek :

16 Mod 3 = 1
15 Mod 3 = 0
17 Mod 3 = 2
3 Mod 15 = 3

Var a,b,x,y : integer;

Begin

a := 9; b := 4;

x := a Div b ;

y := a Mod b ;

WriteLn(a, ' sayisinin ', b, ' sayısına tam bolumu = ', x, ' dir. ');

WriteLn(a, ' sayisinin ', b, ' sayısına tam bolumunden kalan = ', y, ' dir. ');

End.

Monitör



Matematiksel Formüller :

Matematiksel formüllerin bilgisayar programında ifade edilmesinde bazı kurallara uyulması gerekmektedir. Bu kurallar çerçevesinde matematiksel operatörlerin öncelik sıraları önem kazanmaktadır. Matematiksel işlemler bu önceliklere göre yapılmaktadır.

Matematiksel operatörlerin öncelik sırası;

1. Parantez işlemleri
2. Üst alma işlemleri
3. Çarpma/Bölme işlemleri
4. Toplama/Çıkarma işlemleri

Örnek : $a = \pi r^2$ formülünün yazılımı ;

$$a := 3.14159 * r * r ;$$

Matematiksel ifade hesaplanırken eşit öncelik sırasına sahip operatörler için işlem önceliği yazım sırasına göre soldan sağa doğru olacak şekilde belirlenir. Buna göre yukarıdaki matematiksel ifade aşağıdaki şekilde işlenir;

- a. 3.14159 ile r çarpılır
- b. Bulunan sonuç ikinci r ile çarpılır
- c. Bulunan sonuç a değişkenine atanır.

Örnek : $v = \frac{P2 - P1}{T2 - T1}$ formülünün yazılımı ;

$$v := (P2 - P1) / (t2 - t1) ;$$

Örnek : $f = a + \frac{b+c}{d + \frac{e}{2}}$ formülünün yazılımı ;

$$f := a + (b + c) / (d + e / 2) ;$$

Matematiksel Komutlar :

1. Sqrt (Square Root) Komutu ; Sayının karekökünü veren komuttur.

Kullanım şekli: $b := \text{Sqrt}(a);$

a : Real veya Integer olabilir

b : Real olmak zorundadır.

$\text{WriteLn}(\text{Sqrt}(25));$ 5. 0000000000E+00

$\text{WriteLn}(\text{Sqrt}(25):4:0);$ _ _ _ 5

2. Sqr (Square) Komutu ; Sayının karesini veren komuttur.

Kullanım şekli: $b := \text{Sqr}(a);$

a : Real veya Integer olabilir

b : Real veya Integer olabilir.

$\text{WriteLn}(\text{Sqr}(5));$ 25

$\text{WriteLn}(\text{Sqr}(-4):4);$ _ _ 16

3. Int Komutu ; Sayının virgülden sonraki kısmını atan ve tam sayı kısmını reel değişkene atayan komuttur.

Kullanım şekli: $b := \text{Int}(a);$

a : Real tanımlanır

b : Real tanımlanır

$\text{WriteLn}(\text{Int}(5.8));$ 5.0000000000E+00

$\text{WriteLn}(\text{Int}(3.2));$ 3.0000000000E+00

4. **Trunc (Truncate) Komutu** ; Sayının virgülden sonraki kısmını atan ve tam sayı kısmını Long Integer değişkene atayan komuttur.

Kullanım şekli: $b := \text{Trunc}(a);$

a : Real tanımlanır

b : Long Integer tanımlanır

$\text{WriteLn}(\text{Trunc}(3.78));$ 3

$\text{WriteLn}(\text{Trunc}(5.3):4);$ _ _ _ 5

5. **Round Komutu** ; Reel sayıyı en yakın tam sayıya yuvarlayan komuttur.

Kullanım şekli: $b := \text{Round}(a);$

a : Real tanımlanır

b : Long Integer tanımlanır

$\text{WriteLn}(\text{Round}(3.78));$ 4

$\text{WriteLn}(\text{Round}(5.3):4);$ _ _ _ 5

$\text{WriteLn}(\text{Round}(-2.67));$ -3

6. **Abs (Absolute) Komutu** ; Sayının mutlak değerini alan komuttur.

Kullanım şekli: $b := \text{Abs}(a);$

a : Real veya Integer olabilir

b : Real veya Integer olabilir.

$\text{WriteLn}(\text{Abs}(-3.8));$ 3.8000000000E+00

7. Exp (Exponent) Komutu ; Sayının e üstünü bulan komuttur.

Kullanım şekli: $b := \text{Exp}(a); = e^a$

a : Real tanımlanır

b : Real tanımlanır

WriteLn(Exp(1)); 2.7182818285E+00

8. Ln (Logarithm) Komutu ; Sayının doğal logaritmasını bulan komuttur.

Kullanım şekli: $b := \text{Ln}(a);$

a : Real veya Integer olabilir

b : Real olmak zorundadır.

WriteLn(Ln(2.7182818285)); 1.0000000000E+00

9. Cos / Sin Komutu ; Radyan olarak belirtilecek sayının Cosünüs / Sinüs 'ünü bulan komuttur.

Kullanım şekli: $b := \text{Cos}(a); b := \text{Sin}(a);$

a : Real veya Integer olabilir

b : Real olmak zorundadır.

WriteLn(Cos(3.1415926536)); -1.0000000000E+00

10. ArcTan Komutu ; Radyan olarak belirtilecek sayının Arctanjant 'ını bulan komuttur.

Kullanım şekli: $b := \text{ArcTan}(a);$

a : Real veya Integer olabilir

b : Real olmak zorundadır.

`WriteLn(ArcTan(3.1415926535 / 2));`

1.0000000000E+00

Örnek :

`WriteLn(Sqr(Round(Abs(-4.7))):5);`

`WriteLn(Sqrt(Trunc(Abs(-25.7))):5);`

Monitör

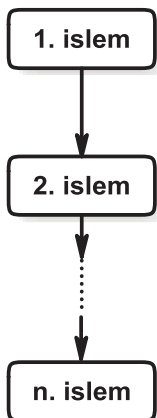


AKIŞ DİYAGRAMLARI (Devam)

Bilgisayar programlarının kodlanmasında kullanılan dil ne olursa olsun, bu programların geliştirilmesinde kullanılan akış diyagramları için 3 yapı kullanılır.

1. Sıralı akış (sequential flow)
2. Şartlı akış (conditional flow)
3. Tekrarlı akış (repetitive flow)

SIRALI AKIŞ :

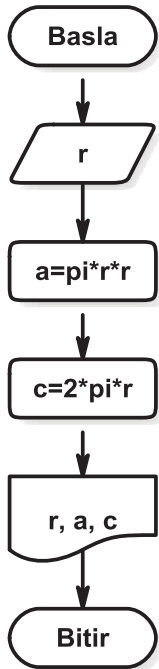


Program içindeki her işlemin/adımın mantık sırasına göre nerede yer alacağını gösterir. Önceki işlem bitmeden diğeri başlıyamaz. Akış yukarıdan aşağıya doğrudur.

Örnek : Yarıçapı girilen birçemberin alanını ve çevresini hesaplayıp bunları ekrana açıklamalı olarak yazdıran programı geliştiriniz.

1. Problem tanımlandı,
2. Analiz (Giriş : Yarıçap (r), Çıkışlar : Çevre (c), alan (a));
3. Tasarım
 1. Başla,
 2. Yarıçapı oku,
 3. Alanı bul,
 4. Çevreyi bul,
 5. Yarıçapı, alanı ve çevreyi ekrana yaz,
 6. Bitir.
4. Program kodu

Akış diyagramı



5. Test ve hata ayıklama

```
Program alan_cevre;

Const
Pi=3.1415;
Var
r, a, c :real;
begin

write('Yaricapi giriniz:'); readln(r);

a:=pi*r*r; {a:=pi*sqr(r); kullanilabilir}

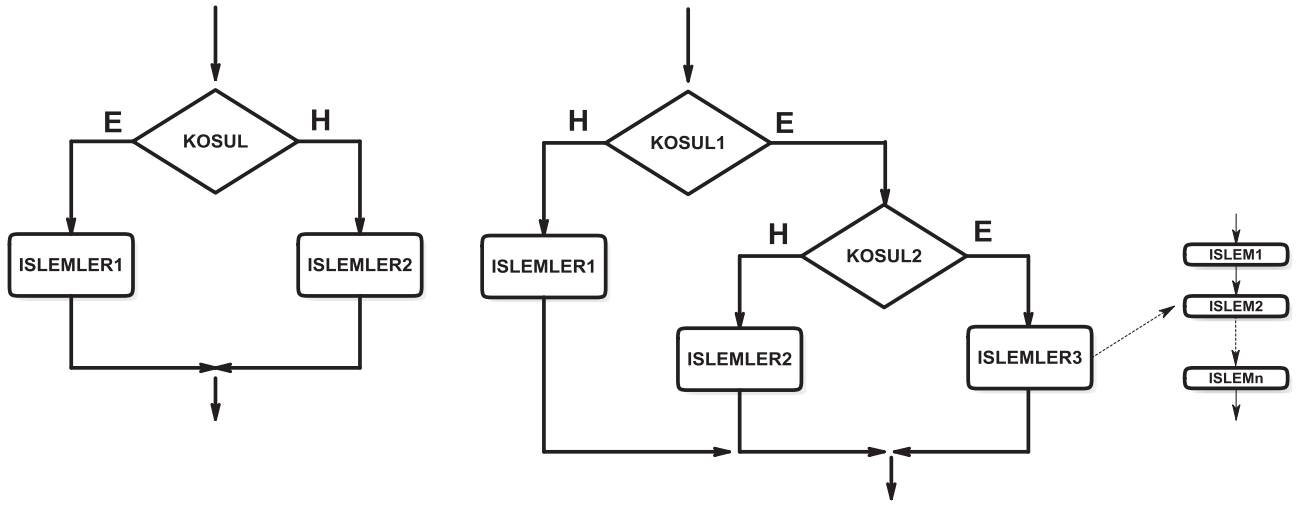
c:=2*pi*r;

Writeln('r=',r,' olan cemberin alanı=',a:6:2,' ve cevresi=', c:6:2);

end.
```

ŞARTLI AKIŞ :

Programın akışını şartlara/koşullara göre değiştirmek için kullanılan komutlardır. Bu yapıda karşılaştırma işlemleri ve birden fazla sıralı yapı kullanılır. Hangi şartlarda hangi sıralı yapının seçileceği karşılaştırma sonucu belirlenir. Pascal programlama dilinde şartlı akış komutları *if....then....else* ve *case....of* 'tur.



Pascal Programlama Dilindeki Karşılaştırma Operatörleri:

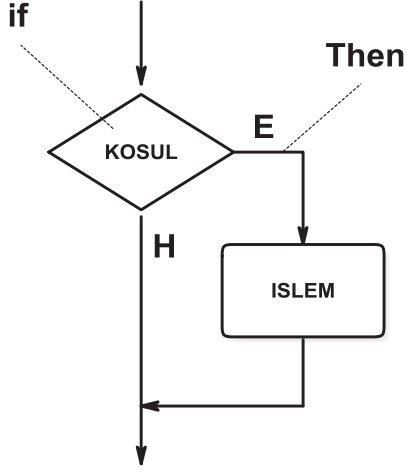
Karşılaştırma işlemlerinin yapılmasını sağlayan operatörlerdir. Bu işlemlerin sonuçları BOOLEAN tiptedir.

Operatör	=	≠	<	>	<=	>=
Anlamı	Eşit	Farklı	Küçük	Büyük	Küçük eşit	Büyük eşit

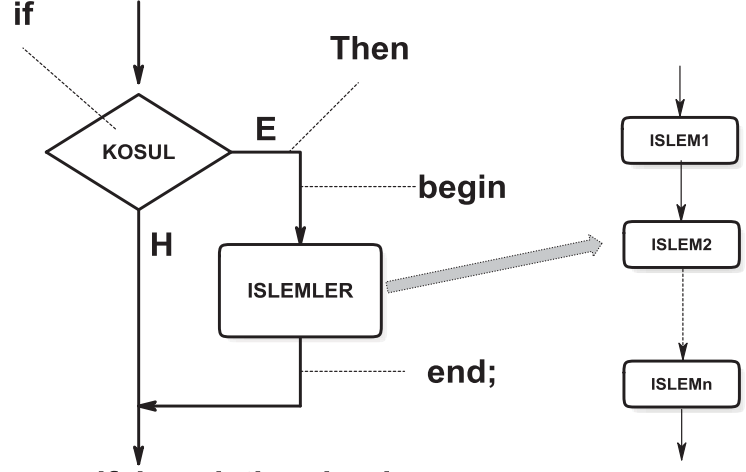
if...then....else yapısı:

Verilen koşula göre programın akışını değiştiren yapıdır. Farklı kullanım şekilleri vardır.

if...then yapısı:



if kosul then islem;



if kosul then begin

islemler;

end;

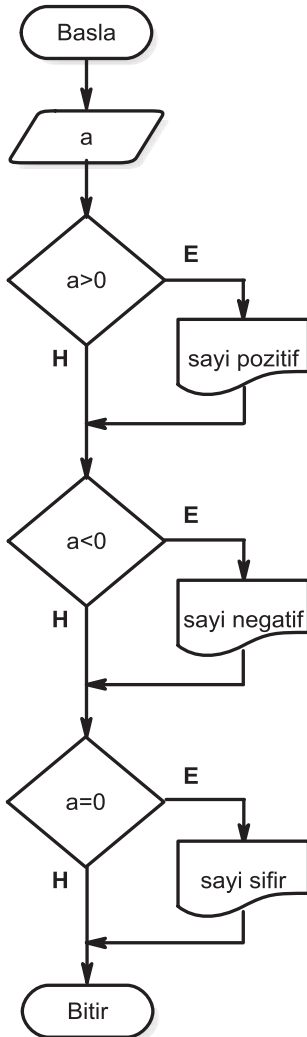
örnek :

```
If a>10 then begin  
x:=x+1; → İşlem  
end;
```

*begin...end; kullanmak mecburi değil

```
If a>10 then  
Begin  
x:=x+1;  
write(x); } İşlemler  
end;
```

Örnek : Klavyeden girilen bir tam sayının pozitif, negatif veya sıfır olduğunu bulan programı yazınız.



Var

a: integer;

begin

Write('Bir tamsayı giriniz:'); Readln(a);

if a>0 then

writeln('Girdiginiz ',a,' sayisi pozitif');

if a<0 then

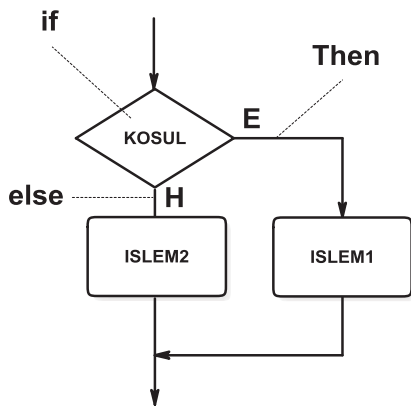
writeln('Girdiginiz ',a,' sayisi negatif');

if a=0 then

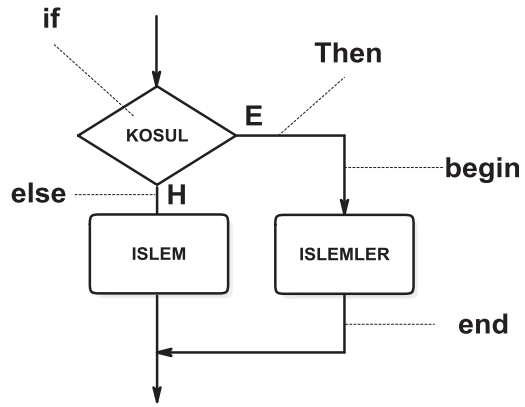
writeln('Girdiginiz ',a,' sayisi sıfır');

end.

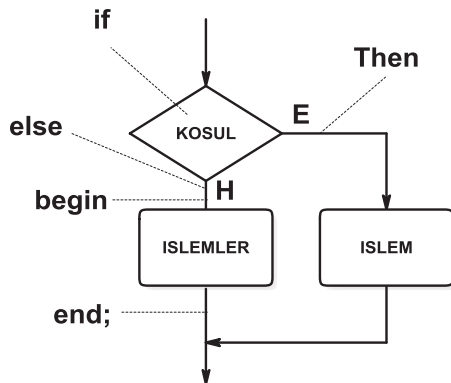
if...then...else yapısı:



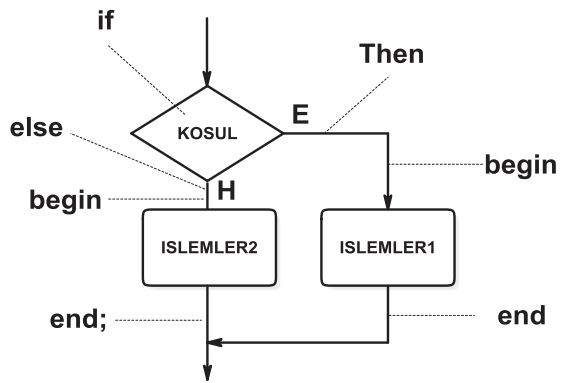
if kosul then islem1
else islem2;



if kosul then begin
islemler;
end
else islem;



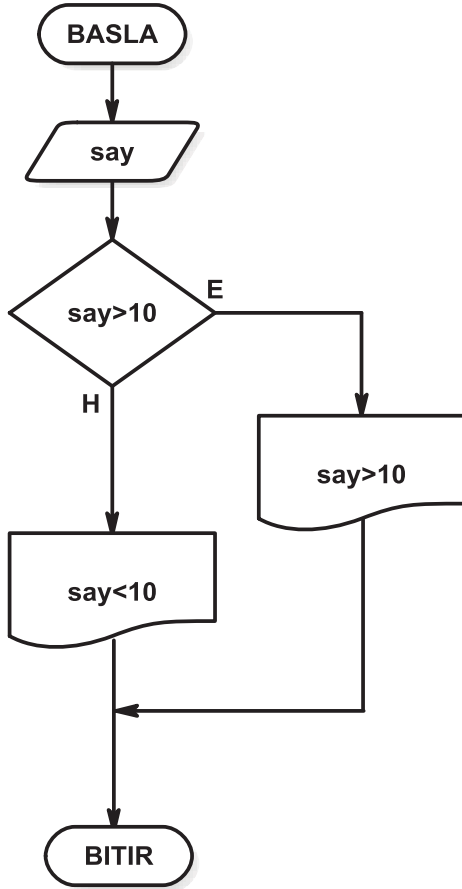
if kosul then islem
else begin
islemler;
end;



if kosul then begin
islemler1;
end
else begin
islemler2;
end;

Örnek : Klavyeden girilecek bir tamsayının 10 sayısından büyük veya küçük olduğunu bulan program.

Analiz : Girişler (sayı:say), Çıkış (10 'dan büyüktür veya değildir)



Var

say: integer;

begin

Write('Bir tamsayi giriniz:'); Readln(say);

if say > 10 then

writeln('Girdiginiz ', say, ' sayisi 10 'dan buyuktur')

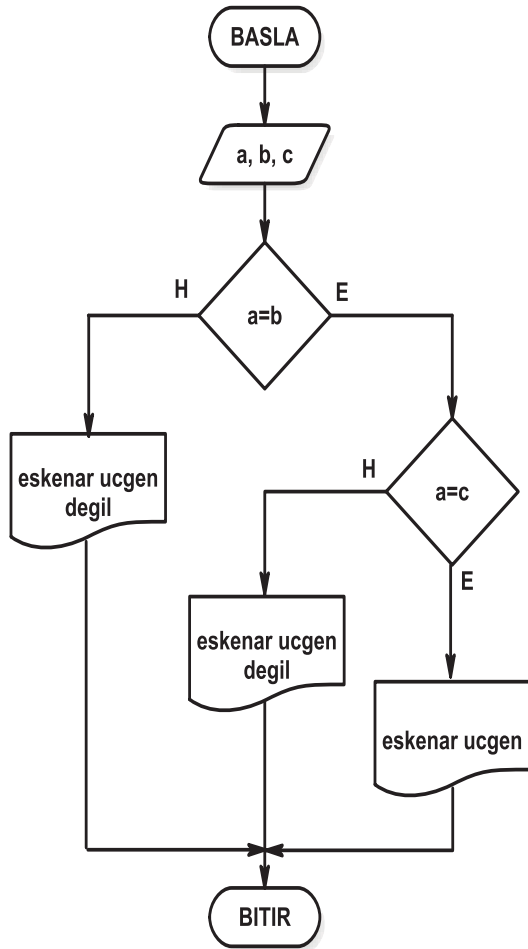
else

writeln('Girdiginiz ', say, ' sayisi 10 'dan kucuktur');

end.

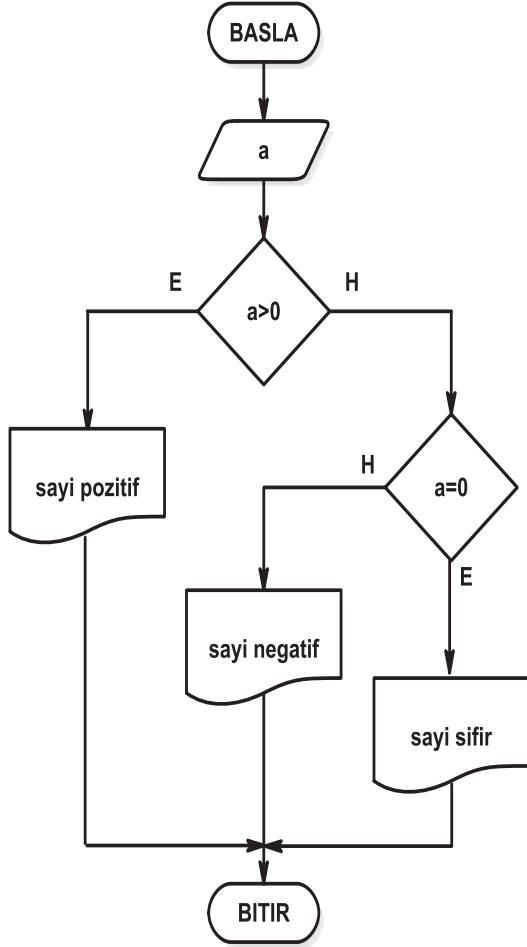
Örnek : 3 kenarının uzunluğu girilecek bir üçgenin eşkenar olup-olmadığını bulacak programı akış diyagramını çizerek yazınız.

Analiz : Girişler (üçgenin kenar uzunlukları,a,b,c), Çıkış (Eşkenar üçgendir veya değildir)



```
Var  
a, b, c: integer;  
  
begin  
Write('Uc tamsayi giriniz:'); Readln(a,b,c);  
  
if a=b then  
    if a=c then writeln ('Eskenar Ucgen')  
    else writeln ('Eskenar Ucgen degil')  
  
else writeln ('Eskenar Ucgen degil');  
  
end.
```

Örnek : Klavyeden girilen bir tam sayının pozitif, negatif veya sıfır olduğunu bulan programı yazınız.



Var

a: integer;

begin

Write('Bir tamsayı giriniz:'); Readln(a);

if a>0 then

writeln('Girdiginiz ',a,' sayisi pozitif')

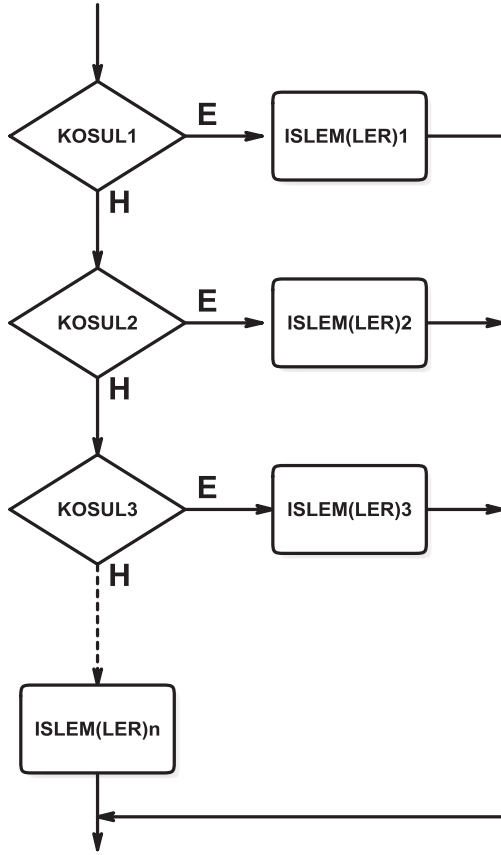
else

if a=0 then writeln('Girdiginiz ',a,' sayisi sıfır')

else writeln('Girdiginiz ',a,' sayisi negatif');

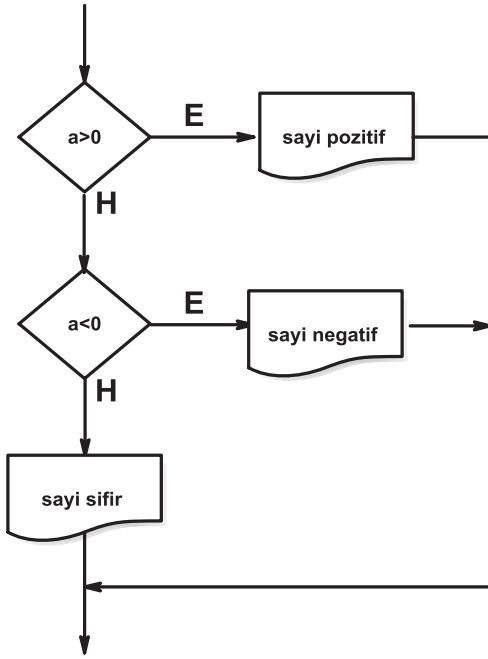
end.

if...then...else if...else yapısı:



```
if kosul1 then islem(ler)1  
else if kosul2 then islem(ler)2  
else if kosul3 then islem(ler)3  
.  
.  
.  
else islem(ler)n;
```

Örnek : Klavyeden girilen bir tam sayının pozitif, negatif veya sıfır olduğunu bulan programı yazınız.



Var

a: integer;

begin

Write('Bir tamsayı giriniz:'); Readln(a);

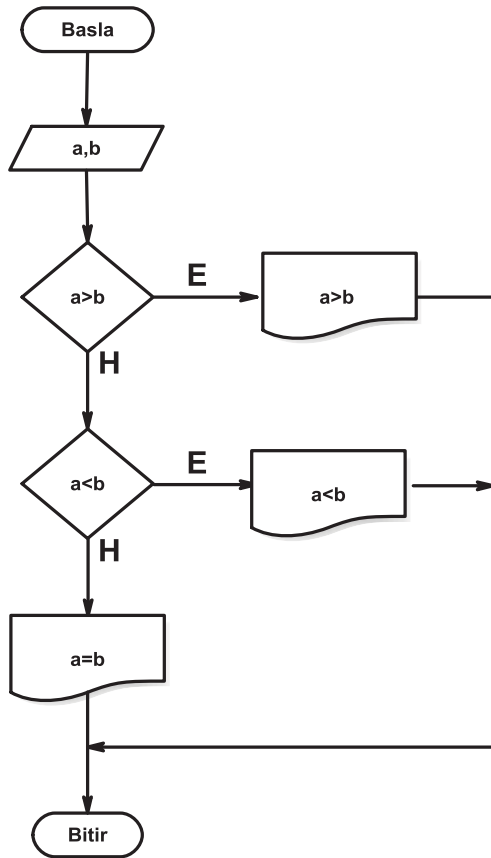
if a>0 then writeln('Girdiginiz ',a,' sayisi pozitif')

else if a<0 then writeln('Girdiginiz ',a,' sayisi negatif')

else writeln('Girdiginiz ',a,' sayisi sıfır');

end.

Örnek : Klavyeden girilen iki tam sayıyı karşılaştıran programı yazınız.



```
Var
```

```
a,b: integer;
```

```
begin
```

```
Write('iki tamsayı giriniz:'); Readln(a,b);
```

```
if a>b then
```

```
writeln('Girdiginiz ',a,' sayisi ',b,' sayisindan buyuktur')
```

```
else if a<b then
```

```
writeln('Girdiginiz ',a,' sayisi ',b,' sayisindan kucuktur')
```

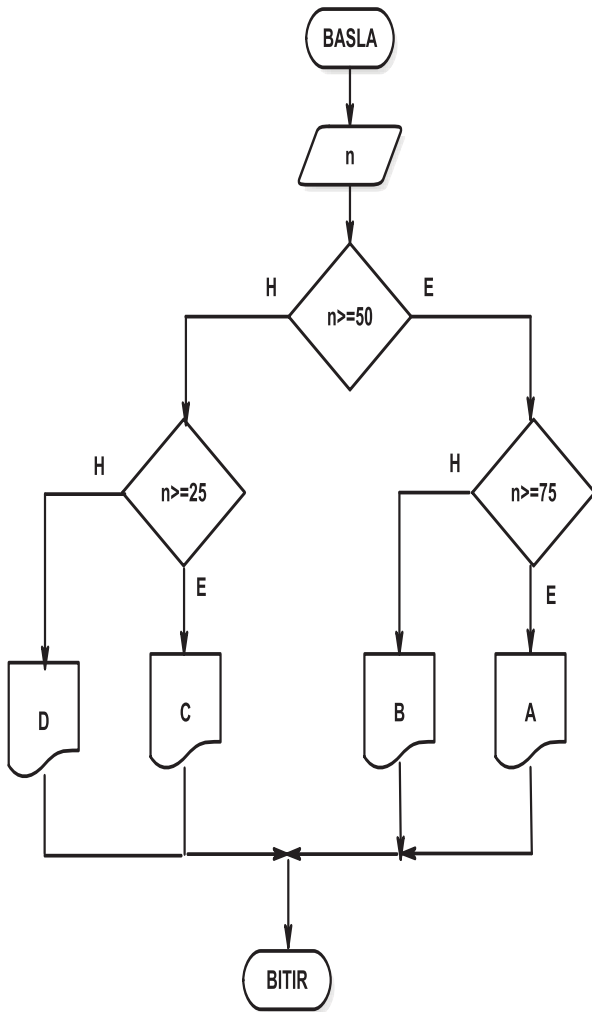
```
else
```

```
writeln('Girdiginiz ',a,' sayisi ',b,' sayisina esittir');
```

```
end.
```

Örnek : Klavyeden girilen notu harf sistemine çeviren programı yazınız.

D:0 .. 24 , C:25..49 , B:50 .. 74 , A:75 .. 100



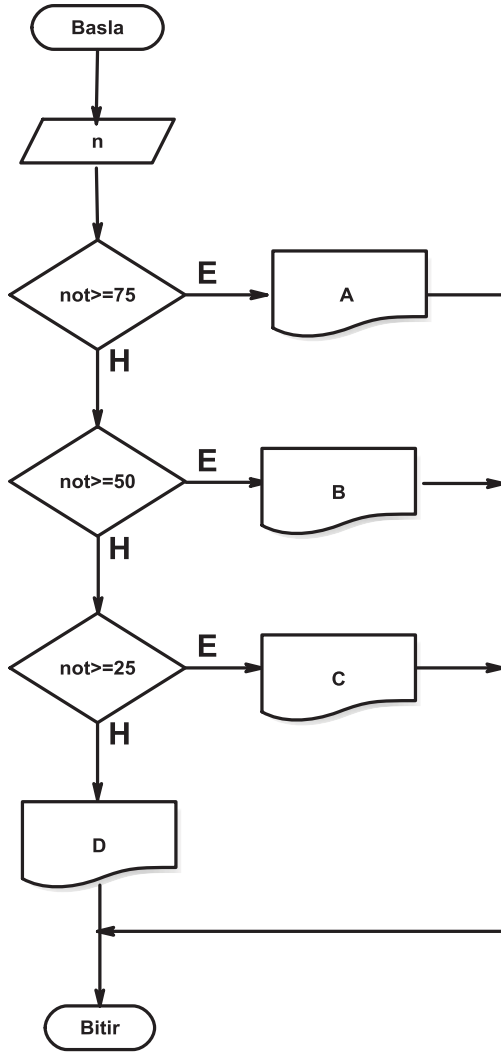
```
Var
n: integer;

begin
Write('Notu giriniz:'); Readln(n);

if n >= 50 then
    if n >= 75 then writeLn('A')
    else writeLn('B')
else
    if n >= 25 then writeLn('C')
    else writeLn('D');

end.
```

2. yol



```
Var  
n: integer;  
  
begin  
Write('Notu giriniz:'); Readln(n);  
  
if n >= 75 then writeln('A')  
else if n >= 50 then writeln('B')  
else if n >= 25 then writeln('C')  
else writeln('D') ;  
  
end.
```

ÖDEV : Klavyeden girilecek 3 tam sayıyı büyükten küçüğe sıralayıp ekrana yazdıran program.

Case .. of Yapısı :

Bir deęişkenin alacağı birden fazla deęer için belirli program parçalarını çalıştıran yapıdır.

Yapısı ;

Secicinin alacağı deęer (tamsayı veya char);

Case secici of

1. Hangi seçenikle uyuyorsa ona ait işlem(ler) çalıştırılır ve blok end; ile biter,

Secenek1 : islem(ler)1;

2. İki seçenikle uyuyorsa ilk seçenek işler,

Secenek2 : islem(ler)2;

3. Hiçbir seçenikle uyumuyorsa else çalışır, else kullanımı zorunlu deęildir. Else kullanıldığı durumda bir üst satırın sonundaki ; işareti silinmez.

Secenekn : islem(ler)n;

Seçenek sabit bir deęer olabileceęi gibi bir aralık veya birden çok deęer olabilir.

Else islem(ler)m;

9..15 : 9,10,11,12,13,14,15 sayılarını gösterir.

end;

1,5,8 : 1,5,8 sayılarını gösterir.

Seçenekten sonra çalıştırılan işlem sayısı birden fazla ise ***begin .. end;*** kullanılır.

Örnek : Klavyeden girilen notu harf sistemine çeviren programı yazınız.

D:0 .. 24 , C:25..49 , B:50 .. 74 , A:75 .. 100

Var

n : integer;

begin

Write('Notu giriniz:'); Readln(n);

Case n of

75..100 : Writeln('A');

50..74 : Writeln('B');

25..49 : Writeln('C');

0..24 : Writeln('D');

Else Writeln('Not 0..100 arasında degil');

End;

End.

TEKRARLI AKIŞ

İşlem yada işlemlerin bir koşula bağlı olarak yada istenilen sayıda tekrarlanmasını sağlayan yapılardır. Programlama dillerinde birden fazla çalışan işlem yada işlemleri sağlayan yapılara DÖNGÜ denir. Pascal Programlama dilinde ,

For..do

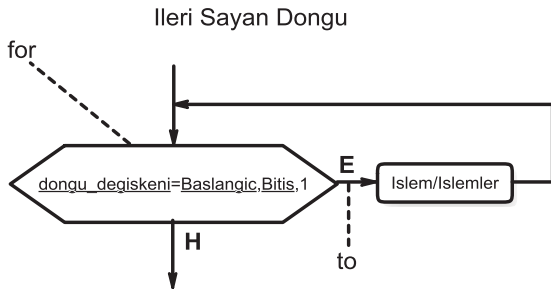
While..do

Repeat..until

Olmak üzere 3 farklı döngü yapısı bulunur.

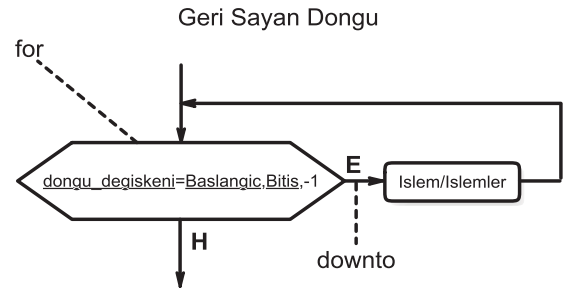
For .. Do Döngüsü:

Belirlenen işlem yada işlemleri istenilen sayıda tekrarlamak için kullanılır. Bu yapının kullanılabilmesi için döngünün kaç kez tekrar edileceği mutlaka bilinmelidir. Döngü içindeki işlem/işlemler, döngü değişkeni (sayaç) ile başlangıç ve bitiş değerleri arasında birer artarak veya azalarak tekrarlanır.



```
for dongu_degiskeni:=Baslangic to Bitis do  
    islem;
```

```
for dongu_degiskeni:=Baslangic to Bitis do  
    begin  
    islemler;  
    end;
```



```
for dongu_degiskeni:=Baslangic downto Bitis do  
    islem;
```

```
for dongu_degiskeni:=Baslangic downto Bitis do  
    begin  
    islemler;  
    end;
```

Döngü değişkeni olarak tamsayı, boolean veya char tipler kullanılabilir.

```
for i:=2 to 6 do  
Writeln(i*i);  
Writeln(i);
```

Monitör



```
For i:=6 downto 4 do  
Begin  
Writeln (i+i);  
Writeln(i);  
End;
```

Monitör



```
For ch:='A' to 'F' do  
Write(ch,'b');
```

Monitör



```
for i:=1 to 10 do ;  
Writeln('Pascal');
```

Monitör



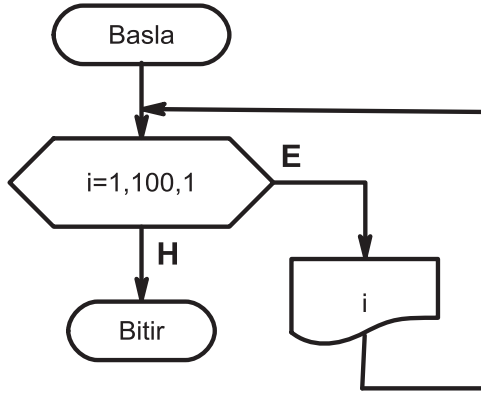
İç-içe Döngüler : Öncelikle içteki döngü sonra dıştaki döngü biter. Dıştaki döngü sayısı kadar içteki döngü çalışır.

```
for i:=1 to 5 do  
begin  
for m:=1 to 5 do  
write('*');  
writeln;  
end;
```

Monitör



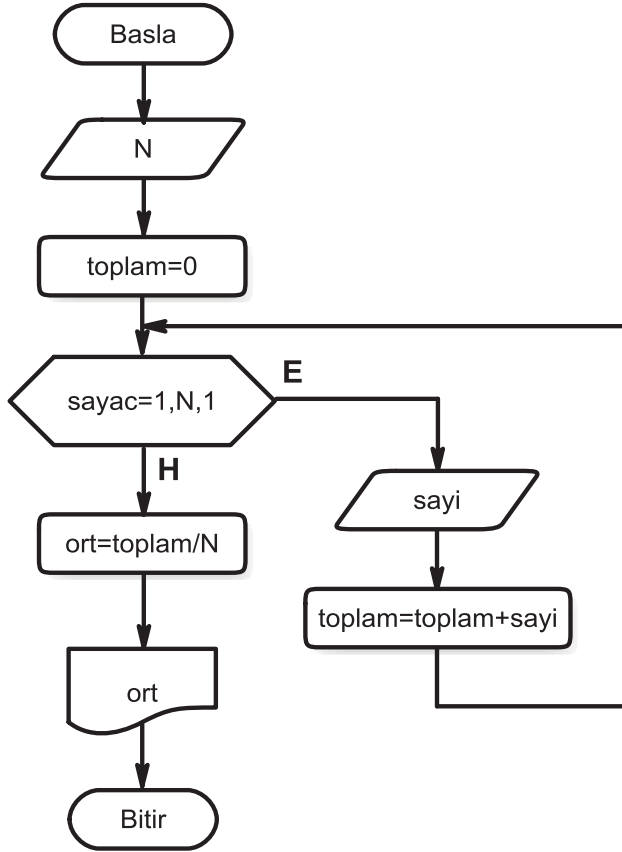
Örnek : 1..100 'e kadar olan sayıları yanyana bir boşlukla ekrana yazdıran program.



```
Var  
i: integer;  
Begin  
for i:=1 to 100 do  
write(i, ' ');  
end.
```

Örnek : Klavyeden girilecek N tane tamsayının ortalamasını bulup ekrana yazan program.

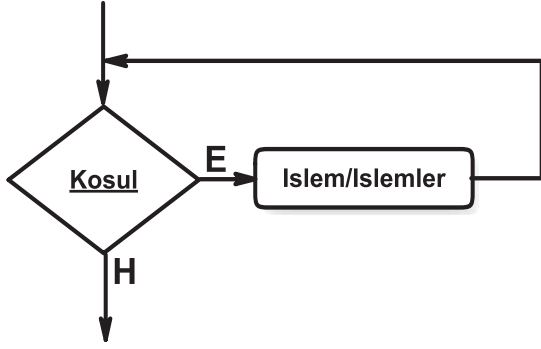
Analiz : Girişler (kaç sayı girilecek:N, sayılar:sayı), Çıkış (ortalama:ort)



```
Var
N,toplam,sayac , sayi: integer;
ort : real;
Begin
Write('Kac sayi girilecek:'); readln(N);
toplam:=0;
for sayac:=1 to N do
begin
write('Sayi=');Readln(sayi);
toplam:=toplam+sayi;
end;
ort:=toplam/N;
writeln('Ortalama=',ort:5:3);
end.
```

While .. Do Döngüsü:

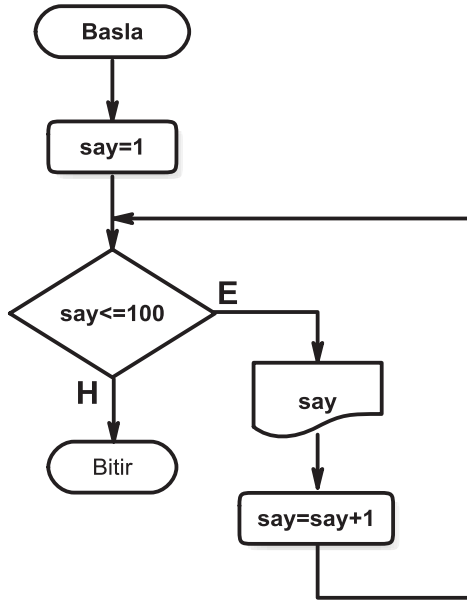
Koşul doğru olduğu sürece istenilen işlem/işlemlerin tekrarlanmasını sağlayan yapıdır.



While koşul **do**
islem;

While koşul **do**
begin
islemler;
end;

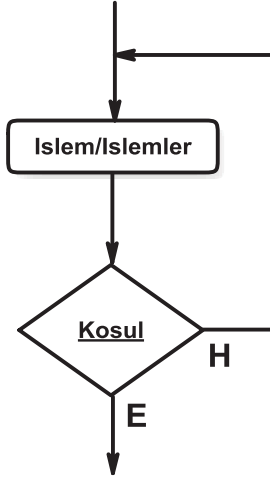
Örnek : 1..100 'e kadar olan sayıları yan yana bir boşlukla ekrana yazdıran program.



```
Var  
say : integer;  
Begin  
say:=1;  
while say<=100 do begin  
    write(say,' ');  
    say:=say+1;  
end;  
end.
```

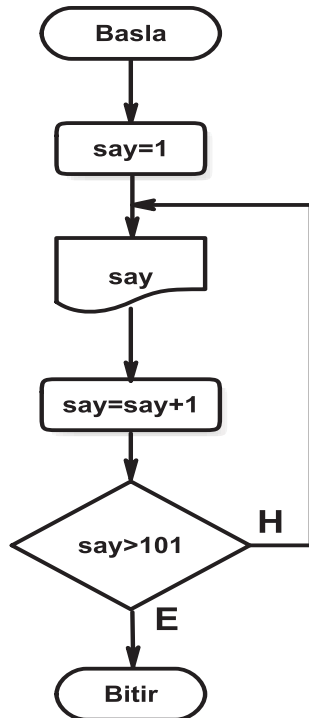
Repeat .. Until Döngüsü:

Koşul sağlanıncaya kadar istenilen işlem/işlemlerin tekrarlanmasını sağlayan yapıdır.



Repeat
islem/islemler;
until kosul;

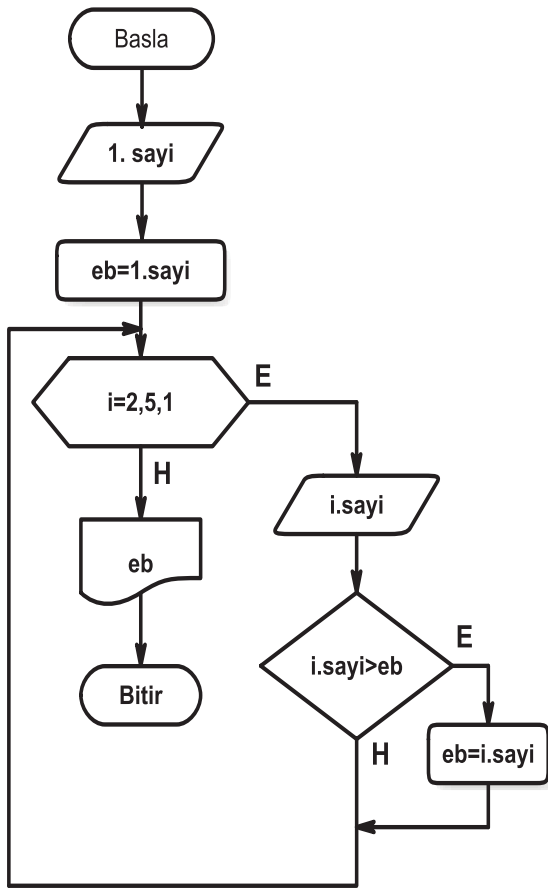
Örnek : 1..100 'e kadar olan sayıları yanyana bir boşlukla ekrana yazdıran program.



```
Var  
say : integer;  
Begin  
say:=1;  
repeat  
    write(say,' ');  
    say:=say+1;  
until say>100;  
end.
```

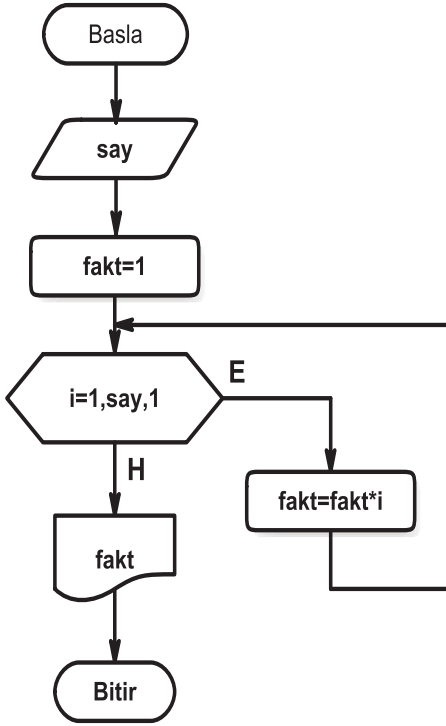
While .. Do yapısında koşul döngüye girmeden, Repeat .. Until yapısında ise döngü sonunda test edilir. Bu yüzden while .. do yapısında döngüye hiç girmeyebilir, repeat .. until döngüsünde ise en az 1 kez girilir.

Örnek : Klavyeden girilecek 5 tam sayının en büyüğünü bulan program.



```
Var
i, say, eb: integer;
ort : real;
Begin
Write('1.sayiyi girin:'); readln(say);
eb:=say;
for i:=2 to 5 do
begin
write(i, '.sayiyi girin:'); readln(say);
if say>eb then eb:=say;
end;
writeln('Girilen sayilarin en buyugu:',eb);
end.
```

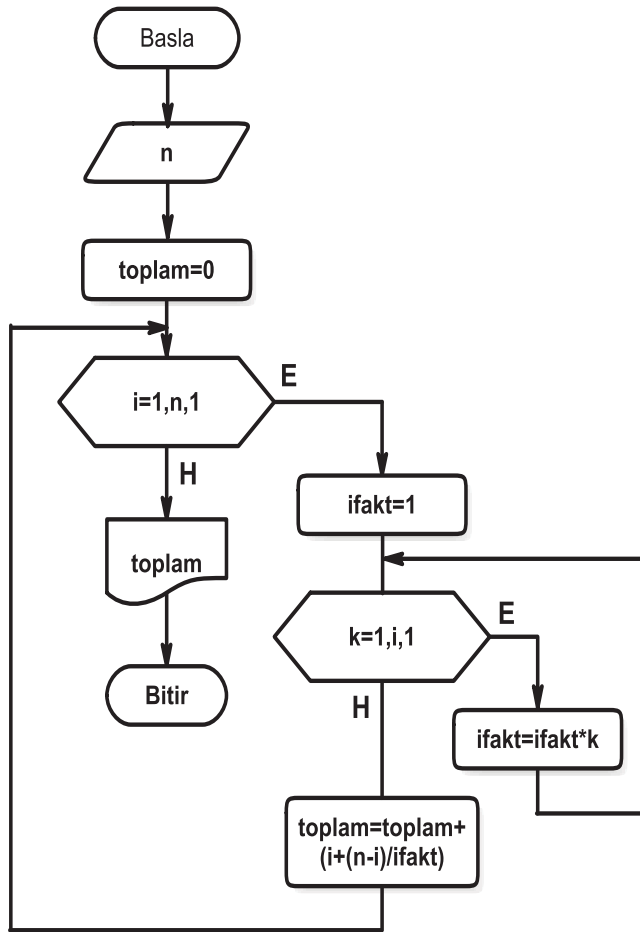
Örnek : Klavyeden girilecek sayının faktoriyelini bulan program.



```
Var  
i, say: integer;  
fakt : longint;  
Begin  
Write('Faktoriyeli hesaplanacak sayi:'); readln(say);  
fakt:=1;  
for i:=1 to say do  
fakt:=fakt*i;  
write(say,'!='); writeln(fakt);  
end.
```

Örnek : Klavyeden girilecek n tam sayısı için aşağıdaki işlemin sonucunu bulan program.

$$\sum_{i=1}^n \left(i + \frac{n-i}{i!} \right) = \left(1 + \frac{n-1}{1!} \right) + \left(2 + \frac{n-2}{2!} \right) + \left(3 + \frac{n-3}{3!} \right) + \dots$$



```

Var
i, n, k : integer;
ifakt : longint;
toplam : real;
Begin
Write('n='); readln(n);
toplam:=0;
for i:=1 to n do
begin
ifakt:=1;
for k:=1 to i do
ifakt:=ifakt*k;
toplam:=toplam+(i+(n-i)/ifakt);
end;
write( 'Sonuc=',toplam:10:2);
end.
  
```

Ödev :

$$\sum_{i=1}^n \left(\frac{1}{i!} + \frac{i}{(n-i)!} \right)$$

Pascal Programlama Dili Genişletilmiş Program Yapısı

Program _____; → Program Başlığı
Uses _____; → Kütüphane (Unit) çağırmak
Type _____; → Özel veri tipi tanımlamak
Const _____; → Sabit tanımlamak
Label _____; → Etiket tanımlamak
Var _____; → Değişken tanımlamak

Procedure/Function Altprogram Adı;
Type _____;
Const _____;
Label _____;
Var _____;
Begin
.
.
End;

Altprogram

Begin
.
.
.
End.

Ana Program

Kütüphane Bloğu Tanımlama (Uses – Unit)

Uses Kütüphane_ismi_1, Kütüphane_ismi_2,....., Kütüphane_ismi_n;

Pascal'ın sistem kütüphanesinde bulunmayan ek prosedür veya fonksiyonların bulunduğu kütüphaneler programın başında Uses ayrılmış kelimesinden sonra tanımlanır. Örneğin ekran modları, ileri seviye klavye işlemleri, renk ve ses ile ilgili prosedür ve fonksiyonlar Crt kütüphanesinde bulunur ve

Uses Crt ; şeklinde ifade edilir.

Yazıcı ile ilgili prosedür ve fonksiyonlar Printer kütüphanesinde, Disk ve işletim sistemi ile ilgili fonksiyonlar DOS kütüphanesinde, grafik ile ilgili fonksiyonlar Graph kütüphanesinde bulunur.

Uses Crt,Dos,Printer,Graph;

Ayrıca programcı, programlarında kullandığı ortak yapı ve alt programları kendi oluşturduğu kütüphanesinde toplayarak Uses ile kullanabilir.

CRT K.

Ekranı Temizleme Komutu (ClrScr – Clear Screen)

ClrScr komutu ekranı temizler ve imleç ekranın sol üst köşesinde konumlanır. Bu komut Crt kütüphanesinin bir komutudur.

ClrScr;

Klavyeden Karakter Okuma Komutu (ReadKey)

Klavyeden girilen komutu ekrana aksettirmeden okuyan komuttur. Bu komut Crt kütüphanesinin bir komutudur.

C1 := ReadKey;

ReadLn ve ReadKey komutu arasındaki farklar;

1. Read(Ln) pekçok tipteki değişkene değer atar, ReadKey komutu ise sadece karakter tipi değişkene değer atar.
2. Read(Ln) komutu ile okunan değerler ekranda gözüktür, ReadKey komutu ile okunan tuş ise ekranda gözükmez.
3. Read(Ln) komutu ile değer okunduğunda en az bir kez Enter tuşuna basılması gerekir, ReadKey komutunda ise Enter tuşuna **basılmasına gerek yoktur.**
4. Readln komutu tek başına kullanıldığında enter, readkey tek başına kullanıldığında herhangi bir tuşa basılıncaya kadar bekler.

Örnek : Readkey komutu ile “enter” tuşunu kontrol etmek

Enter = 13 (ASCII)

Uses crt;

Var

Tus : char;

Begin

Clrscr;

.

.

Repeat

 Write(‘Devam icin enter tuşuna basiniz....’);

 Tus:=readkey;

Until Tus=#13;

.

.

End.

Programın akışı harf tuşlarına göre yönlendirilecekse bunların, büyük veya küçük olması durumları göz önüne alınmalıdır.

MANTIKSAL OPERATÖRLER

Operatör	Anlamı
NOT	Değil
AND	Ve
OR	Veya
XOR	Veya değil

Örnek : Programın akışını “e” tuşuna göre yönlendirmek.

```
Uses crt;
```

```
Var
```

```
Tus : char;
```

```
Begin
```

```
Clrscr;
```

```
.
```

```
.
```

```
Write('Devam için e tuşuna basınız....');
```

```
Repeat
```

```
    Tus:=readkey;
```

```
Until (Tus='e') or (Tus='E');
```

```
.
```

```
.
```

```
End.
```

Büyük Harfe Çevirme Komutu (UpCase)

UpCase komutu, karakter tipteki değişken 'a'.. 'z' arasında ise değişkeni büyük harf eşitine çevirir. Karakter 'a'.. 'z' arasında değil ise dönüştürme işlemi yapmaz.

```
C2 := UpCase(C1);
```

Örnek program parçası;

```
:
```

```
Repeat
```

```
:
```

```
:
```

```
Write('Programdan çıkmak için E harfine basınız :');
```

```
C := UpCase(ReadKey);
```

```
Until C = ' E ' ;
```

```
:
```

In / in Karşılaştırma Operatörü

Belirtilen bir değişkenin veya değerinin belirli bir küme içinde olup olmadığını test etmek için kullanılan komuttur.

➤ ***If A In [0..9] Then***

➤ ***If A In [16, 80, 0..9, 18, 22, 85, 99] Then***

➤ ***If C In ['ş', 'ğ', 'ü', 'Ş', 'Ğ', 'Ü'] Then***

➤ ***If A In ['a'..'z', 'A'..'Z', 'ç', 'ı', 'ğ', 'ö', 'ş', 'ü', 'Ç', 'İ', 'Ğ', 'Ö', 'Ş', 'Ü'] Then***

Repeat

```
Tus:=readkey;
```

```
Until Tus In ['E', 'e'];
```

Etiket Tanımlama Bloğu (Label)

Programın akışını farklı satırlara yönlendirmek amacıyla, bu konumlara işaret eden etiketlerin tanımlandığı bloktur.

Label Etiket_ismi_1, Etiket_ismi_2,.....;

Label Baslangic, Son;

Label ile tanımlanan etiket ismi, programın ancak bir satırında kullanılabilir, iki veya daha fazla satırda kullanılırsa derleyici 'Label already defined' hatası verir.

Programın Akışını İstenen Satıra Yönlendirme (GoTo)

Bu komut ile program akışı etiket isminin bulunduğu satıra yönlendirilir. GoTo komutu ile bir prosedür veya fonksiyonun dışına çıkılamaz.

Örnek program parçası;

```
Label bir,iki ;
Var A,B : Integer;
Begin
  Write ('İki tam sayı giriniz :');
  ReadLn (A,B);
  If A > = B
    Then GoTo bir
    Else GoTo iki;
  bir : WriteLn('A>=B''dir. ');
  iki : WriteLn('B>A''dir. ');
End.
```

```
Label bir,iki,son;
Var A,B : Integer;
Begin
  Write ('İki tam sayı giriniz :');
  ReadLn (A,B);
  If A > = B
    Then GoTo bir
    Else GoTo iki;
  bir : WriteLn('A>=B''dir. ');
  GoTo son;
  iki : WriteLn('B>A''dir. ');
  son:
End.
```

Örnek : Ekrandan girilen pozitif tam sayılardan en büyüğünün hangisi olduğunu bulan ve ekrana yazdıran programı geliştiriniz. Ekrandan sayı okuma işlemi negatif bir sayı girilene kadar devam edilecektir, eğer ekrandan hiç pozitif sayı girilmeden negatif bir sayı girilirse ekrana 'Hiç Pozitif Sayı Girilmedi' yazdırılacaktır.

```
Label Oku;
Var EnBuyukSayi,Sayi : Integer;
Begin
  EnBuyukSayi := 0 ;
  Oku: Write ('Bir tam sayı giriniz :');
  ReadLn (Sayi);
  If Sayi >= 0 Then
    Begin
      If Sayi > EnBuyukSayi
        Then EnBuyukSayi := Sayi;
        GoTo Oku;
    End
  Else
    If EnBuyukSayi > 0
      Then WriteLn('En Büyük Sayı=', EnBuyukSayi)
      Else WriteLn('Hiç Pozitif Sayı Girilmedi. ');
End.
```

Örnek : Hem komut hemde iterasyonla karakök bulan programı yazınız. (Tolerans : 2 iterasyon sonucu arasındaki farktır, iterasyon: istenilen sonuca ulaşmak için tekrar edilen her adım)

S pozitif bir sayı olmak üzere karakök bulan iterasyon formülü;

$$y_{i+1} = \frac{y_i + \frac{S}{y_i}}{2}$$

S=16 olsun, ($y_0=16$)

$$i = 0 \Rightarrow y_1 = \frac{y_0 + \frac{S}{y_0}}{2} = \frac{16 + \frac{16}{16}}{2} = 8.5, \text{Tolerans} = 16 - 8.5$$

$$i = 1 \Rightarrow y_2 = \frac{y_1 + \frac{S}{y_1}}{2} = \frac{8.5 + \frac{16}{8.5}}{2} = 5.1911, \text{Tolerans} = 8.5 - 5.1911$$

:

$$i = 4 \Rightarrow y_5 = 4.0000060$$

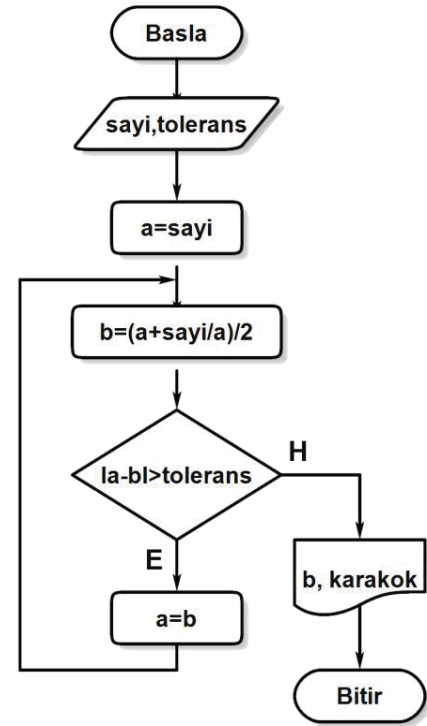
$y_i = a, y_{i+1} = b$

```
Uses crt ;  
Label tekrar;  
Var  
Sayi, tolerans,a,b : real;  
iterasyon_sayisi : integer;
```

```
begin  
clrscr;  
write ('Karakoku bulunacak sayi:'); Readln(sayi);  
write ('Tolerans:'); Readln(tolerans);
```

```
iterasyon_sayisi:=0;  
a:=sayi;  
tekrar:  
    b:=(a+sayi/a)/2;  
    iterasyon_sayisi:= iterasyon_sayisi+1;  
    if ABS (b-a)>tolerans  
    then begin  
        a:=b;  
        goto tekrar;  
    end;
```

```
Writeln(iterasyon_sayisi,' iterasyon sonunda yaklasik karakok=',b:8:6);  
Writeln('Komutla bulunan karakok=',sqrt(sayi):6:3);  
Readln;  
End.
```



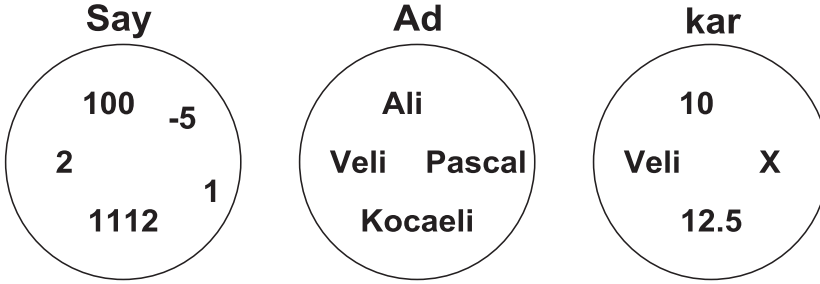
DİZİLER

Programlama dillerinde diziler, aynı tipteki birden fazla veriyi bir isim altında tanımlamak ve kullanmak için kullanılır.

Diziler farklı boyutlarda kullanılabilir.

1. Tek Boyutlu Diziler

Tek boyutlu diziler, aynı tipteki verilerden oluşmuş küme tanımlar.



Dizi içindeki her elemana indis numarası ile erişilir.

Kullanımı;

Var

Dizinin adi : *Array* [Alt sınır .. Üst sınır] of *tip* ;

Örnekler

Var

isim : Array [1..50] of string;

notlar : Array [1..10] of byte;

Z : Array [-10..10] of integer;

Harf_dizisi : Array ['A'..'F'] of real;

Kurallar;

1. Alt sınır ve üst sınır, pozitif veya negatif tamsayı olabilir fakat üst sınır alt sınırdan büyük olmak zorundadır. Bu sınırlar ve arasında kalan değerler dizi elemanlarının indis numarasıdır.
2. Dizi elemanları, indis numaraları ile normal değişkenler gibi kullanılır (dizi ismi ve köşeli parantez içinde indis numaraları ile).

isim[2]:= 'Kocaeli'; => isim dizisinin 2. Elemanına 'Kocaeli' bilgisi yüklendi.

notlar[5]:=15 ; => notlar dizisinin 5. Elemanına 15 bilgisi yüklendi.

Z[-3]:= notlar[5]*2 ; => Z dizisinin -3. Elemanına 15*2 bilgisi yüklendi.

3. İndis numaraları üzerinde matematiksel işlemler yapılabilir.

```
i:=2;  
Z[i+3]:=100; =>Z dizisinin 5. Elemanına 100 bilgisi yüklendi.
```

4. Dizi elemanlarının birden fazlasına veya hepsine atama yapmak için döngü kullanılır.

```
for i:=1 to 10 do  
notlar[i]:=i*10;
```

```
for i:=20 to 40 do  
isim[i]:='AB';
```

5. Dizinin sınırları sabit bloğunda tanımlanabilir.

```
Const  
max =100;  
var  
or : Array [10 .. max] of real;
```

6. Dizi elemanları sabit olarak tanımlanabilir.

```
Const  
S: Array [1 .. 5] of integer = (1, 100, 200, 3, 10);  
isim : Array [1 .. 3] of string[10]=('Ali', 'Veli', 'Ayse');
```

Örnek : Klavyeden girilecek 10 tane tam sayının ortalamasını bulan program

```
Uses crt;  
var  
sayilar : Array [1 .. 10] of integer;  
i: byte;  
ort: real;  
topla : integer;  
Begin  
for i:=1 to 10 do begin  
write(i, ' .sayiyi giriniz:');readln(sayilar[i]); end;  
  
topla:=0;  
for i:=1 to 10 do  
topla:=topla+sayilar[i];  
  
ort:=topla/10; (ort:=topla/i);  
  
write('Sayilarin ortalamasi:',ort:6:2);  
end.
```

Örnek : En fazla 100 öğrencinin vize final notunu okuyup bunların ortalaması ile birlikte listeleyen programı yazınız.

```
Uses crt;
Const
SINIR=100;
vizekat=0.4; finalkat=0.6;
Var
vize, final, basari: Array [1..100] of byte;
i, ogrencisay : byte;
begin
repeat
    clrscr;
    write('Ogrenci sayisi :'); Readln(ogrencisay);
    until (ogrencisay<=100);

for i:=1 to ogrencisay do begin
    write(i, '.ogrencinin vize notu=');readln(vize[i]);
    write(i, '.ogrencinin final notu=');readln(final[i]);
    basari[i]:=vize[i]*vizekat+ final[i]*finalkat;
end;

writeln('Vize notu      ','Final notu      ','Basari notu');
for i:=1 to ogrencisay do
    writeln(Vize[i],',',final[i],',',basari[i]);
readln;
end.
```

Gotoxy komutu : İmleci ekranın isenilen satır ve sütununa taşımak için kullanılır. Gotoxy komutu *crt* kütüphanesinde bulunur.

Kullanımı => ***gotoxy(sütun no, satır no);***
Sütun no= 1..80
Satır no=1..25

Örnek:
Uses crt;
Var
s:integer;
Begin
Clrscr;
For s:=1 to 10 do begin
Gotoxy(s,s); Write('Pascal');
End;
Readln; End.

Arttırma-Azaltma Komutları :

Inc(x); =>x=x+1

Dec(x); =>x=x-1

Inc(x,5); =>x=x+5

Dec(x,5); =>x=x-5

KARAKTER DİZİLERİ

Her string değişken veya sabit, elemanları karakter (char) olan bir boyutlu dizidir. Bir boyutlu dizilerdeki kurallar geçerlidir. Karakter dizilerinin alt indisi "1" dır.

Örnek:

```
Var  
kelime : string[50];  
Begin  
Readln(kelime);  
Readln(kelime[3]);  
Writeln(kelime);  
Writeln(kelime[4]);  
End.
```

```
Hasan←  
  
v←  
  
Havan  
  
a
```

```
kelime='Hasan'  
kelime[1]='H'  
kelime[2]='a'  
kelime[3]='s'  
kelime[4]='a'  
kelime[5]='n'
```

```
kelime[3]='v' → kelime='Havan'
```

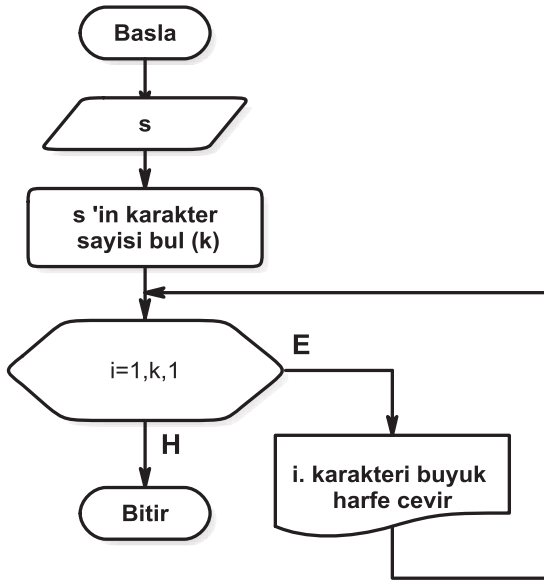
Length Komutu : String ifadedeki karakter sayısını verir. Sonuc byte tipinde bir veridir.

Kullanımı => **length(string);**

```
Const  
a='Elektrik';  
var  
b:byte;  
begin  
b:=length(a);  
writeln(a, ' ', b, ' karakterlidir');  
readln;  
end.
```

```
Elektrik 8 karakterlidir  
-
```

Örnek : Klavyeden girilen küçük harfli string bir ifadeyi büyük harfe çeviren program.



Uses crt;

Var

i:integer; s:string; k:byte;

begin

write('String ifadeyi giriniz:'); readln(s);

k:=length(s);

for i:=1 to k do

write(Upcase(s[i]));

end.

```
String ifadeyi giriniz: Pascal←
```

```
PASCAL_
```

Ord (Ordinal-Order) Komutu : Verilen karakterin ASCII kodunu verir.

```
Write(Ord('A')); => 65
```

String ifadenin kaç karakterden oluştuğu, dizinin 0. İndisinde saklıdır. 0. İndisin ASCII karşılığı stringin karakter sayısını verir.

```
Length(kelime) = Ord(kelime[0])
```

Chr (Character) Komutu : ASCII kodun (0..255) karakter karşılığını verir.

```
Write(chr(65)); => A  
Kelime[0] := Chr(5);
```

```
Uses crt;  
Var  
kelime: string[10];  
Begin  
kelime:='Ahmet';  
writeln(kelime);  
kelime[0]:=Chr(3);  
writeln(kelime);  
End.
```

Stringlerde Toplama Operatörü (+):

```
kelime1:='ALI';  
kelime2:='VELI';  
kelime:=kelime1+kelime2; =>kelime='ALIVELI'
```

Concat (Concatenate) Komutu : Birden fazla stringi birleştirmek için kullanılır (Yukarıdaki toplama işlemi gibi).

Kullanımı => **concat(st1, st2, ...,stn);**

```
Var  
a, b, c, d: string[10];  
Begin  
a:='ALI';  
b:='VELI';  
c:=' ';  
Writeln(concat(a,b));  
Writeln(concat(a,c,b));  
End.
```

ALIVELI
ALI VELI_

Copy Komutu : String ifadenin belirli bir kısmının alınması için kullanılır. İşlem sonucunda string ifade de bir değişiklik olmaz.

Kullanımı => **copy(string, Başlangıç sırası, Karakter sayısı);**

Write((copy(kelime, 7, 6)); =>kelime değişkeninin 7.karakterinden itibaren 6 karakteri ekrana yaz.

Örnek:

```
kelime:='turbo pascal';
kelime2:=copy(kelime,2,7);
Writeln((copy(kelime, 13, 5)));
Writeln((copy(kelime, 10, 5)));
Write((copy(kelime2, 1, 3)));
```

Monitör

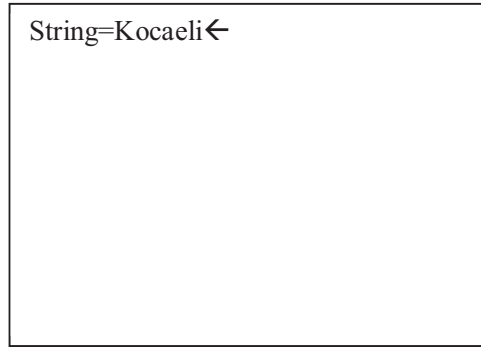


Örnek:

```
Uses crt;
Var
mesaj : string;
i : integer;
begin
clrscr;
write('String=');readln(mesaj);
for i:=1 to length(mesaj) do
writeln(copy(mesaj,1,i));
readln;
end.
```

Monitör

String=Kocaeli←



Delete Komutu : String ifadenin belirli bir kısmının silinmesi için kullanılır. İşlem sonucunda string ifade değişir.

Kullanımı => **delete(string, Başlangıç sırası, Karakter sayısı);**

```
kelime:='turbo pascal';
delete(kelime,6,1);
delete(kelime,13,5);
delete(kelime,10,5);
write(kelime);
```

Monitör



Insert Komutu : String ifadeye başka bir string eklemek için kullanılır.

Kullanımı => **insert(string1(sabit veya değişken), string2(değişken), sıra);**

insert('pascal', kelime, 7); => pascal stringini kelime değişkenine 7.karakterden itibaren ekler.

Örnek:

```
kelime:='aet';
insert('hm', kelime, 2);
writeln(kelime);
ek:='caliskan';
insert(ek, kelime, 1);
writeln(kelime);
```

Monitör



Pos (Position) Komutu : Bir String içinde başka bir string veya karakterin bulunup bulunmadığını, bulunuyorsa kaçınca karakterden itibaren yer aldığı bilgisini verir.

Kullanımı => **pos(string1 veya char, string2);**

```
Writeln(pos('s','pascal')); =>3
Writeln(pos('a','pascal')); =>2
Writeln(pos('say','bilgisayar')); =>6
Writeln(pos('z','pascal')); =>0
```

Str (String) Komutu : Sayısal bir ifadeyi stringe çevirir.

Kullanımı => **str(sayısal sabit veya değişken, string değişken);**

Örnek:

```
Var
s: string[20];
begin
str(15,s); =>s='15'
str(123.45:7:2,s); =>s=' 123.45'
```

Val (Value) Komutu : Stringi sayısal ifadeye çevirir. Çevirme işlemi sırasında, string içinde sayısal olmayan bir karaktere rastlanırsa hata kodu üretilir ve işlem sona erer.

Kullanımı => **val(string,sayısal değişken, kod);**

kod : Dönüştürme işlemi hakkında bilgi veren tamsayı tipinde bir değişkendir. Bu değer “0” olması işlemin başarılı, “0” dan farklı olması dönüştürme işlemi sırasında sayısal olmayan bir karaktere rastlandığını gösterir. Üretilen “0” dan farklı kod, sayısal olmayan karakterin indis numarasıdır.

Örnek:

Val('2345', v, kod); => v=2345, kod=0

a:='abc5';

val(a[4],b,c); => b=5, c=0

val(a[3],b,c); => b=0, c=3

Örnek : Ekrandan girilen 1 ile 99 arasındaki tam sayının Romen rakamı karşılığını bulup ekrana yazdıran program yazınız.

1. Yol : mod ve div operatörleri ile;

2. Yol : Karakter dizileri ile;

Bu yontem okunan sayının kaç basamaklı olduđu belirlenmelidir. Eger okunan sayı tek basamaklı ise str komutu ile stringe cevrıldığında bu dizinin 1 elemanı olur, yani birler basamağı vardır ,onlar basamağı ise sıfırdır,

Eger okunan sayı çift basamaklı ise str komutu ile stringe cevrıldığında bu dizinin 1. Elemanı onlar basamağı, 2. Elemanı ise birler basamağıdır

İki Boyutlu Diziler

Bilgisayarda iki boyutlu olarak değişken tanımlamak için de tek boyutlu dizilerde olduğu gibi 'Array' kullanılır. Ancak iki boyutlu dizilerde tek boyutlu dizilerden farklı olarak birinci indis (sıra sayısı) aralığının yanında ikinci indis (sütun sayısı) aralığı da tanımlanır. Böylece bilgisayarda matris yapısı gibi iki boyutlu değişken kümesi tanımlanmış olur.

Kullanımı;

Var

Dizinin adı : *Array* [Alt sınır1 .. Üst sınır1, Alt sınır2 .. Üst sınır2] of tip ;

Satır sayısı

Sütun sayısı

Örnek:

Var Not_Tablosu: Array[1..5 , 1..3] Of Integer;

Var Tablo: Array['A'..'G',1..10] Of real;

	vize	final	ortalama
1. ogrenci			
2. ogrenci			
3. ogrenci			
4. ogrenci			
5. ogrenci			

Begin

ReadLn(Not_Tablosu[1,1]);

ReadLn(Not_Tablosu[1,2]);

70 ←
60 ←

Not_Tablosu[1,3]:= round(Not_Tablosu[1,1]*0.4+ Not_Tablosu[1,2]*0.6);

İki boyutlu dizilerde dizi elemanlarının tamamını ekrandan okumak, ekrana yazmak veya tablo elemanlarının üzerinde işlem yapmak için içiçe For döngüleri kullanılmaktadır.

```
For k:=1 To 5 Do  
  For p:=1 To 3 Do  
    Read (Not_Tablosu[k,p]);
```

Örnek : Beş öğrenciden oluşan bir sınıftaki öğrencilerin birinci ve ikinci vize notları sırayla bilgisayar girilecektir. Öğrencilerin vize not ortalaması hesaplanıp bilgisayara kaydedilecek ve ekranda gösterilecektir.

```
Program iki_boyutlu_dizi_ornegi;  
Var k,p : Integer;  
  Toplam : Real ;  
  NotTablosu : Array[1..5 , 1..3] of Integer;  
Begin  
  Toplam:=0;  
  WriteLn('Öğrencilerin Notlarını Giriniz :');  
  For k:=1 To 5 Do  
    Begin  
      For p:=1 To 2 Do  
        Begin  
          Write( k , ' öğrencinin ' , p , ' . notunu giriniz : ');  
          ReadLn ( NotTablosu[k,p] );  
          Toplam := Toplam + NotTablosu[k,p] ;  
        End;  
      NotTablosu[k,3]:= (Toplam/2) ;  
      Write( k , ' öğrencinin not ortalaması = ' , NotTablosu[k,3] );  
      Toplam:=0;  
    End;  
  End.
```

İki Boyutlu Diziler İle Matris İşlemleri

Matris değeri okumak ve ekrana yazmak için de içiçe For döngüsü kullanılır.

```
Uses crt;
Var A : Array[1..3,1..2] Of Integer;
    i,j : Integer;
Begin
WriteLn('3X2 boyutlarındaki matris elemanlarını giriniz :');
For i:=1 To 3 Do
    For j:=1 To 2 Do
        Begin
            Write( 'A[ ', i, ', ', j, ' ]=? ');
            ReadLn ( A[i,j] );
        End;
    End;

For i:=1 To 3 Do
    Begin
        For j:=1 To 2 Do
            Write ( A[i,j]:5);
            WriteLn;
        End;
    End.
End.
```

Örnek : Matrislerin boyutlarının ve elemanlarının ekrandan kullanıcı tarafından belirlendiği, A ve B matrisinin toplamını bulan programı yapınız. (Maksimum matris boyutu 10X10 olacaktır).

$$A_{m,n} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdot & \cdot & a_{1,10} \\ a_{2,1} & a_{2,2} & \cdot & \cdot & a_{2,10} \\ \cdot & & & & \\ \cdot & & & & \\ a_{10,1} & a_{10,2} & \cdot & \cdot & a_{10,10} \end{bmatrix}, B_{m,n} = \begin{bmatrix} b_{1,1} & b_{1,2} & \cdot & \cdot & b_{1,10} \\ b_{2,1} & b_{2,2} & \cdot & \cdot & b_{2,10} \\ \cdot & & & & \\ \cdot & & & & \\ b_{10,1} & b_{10,2} & \cdot & \cdot & b_{10,10} \end{bmatrix}$$

Bu iki matrisin toplanabilmesi için $A_m = B_m$ ve $A_n = B_n$ olmalıdır.

$$C_{m,n} = \begin{bmatrix} a_{1,1} + b_{1,1} & a_{1,2} + b_{1,2} & \cdot & \cdot & a_{1,10} + b_{1,10} \\ a_{2,1} + b_{2,1} & a_{2,2} + b_{2,2} & \cdot & \cdot & a_{2,10} + b_{2,10} \\ \cdot & & & & \\ \cdot & & & & \\ a_{10,1} + b_{10,1} & a_{10,2} + b_{10,2} & \cdot & \cdot & a_{10,10} + b_{10,10} \end{bmatrix} = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdot & \cdot & c_{1,10} \\ c_{2,1} & c_{2,2} & \cdot & \cdot & c_{2,10} \\ \cdot & & & & \\ \cdot & & & & \\ c_{10,1} & c_{10,2} & \cdot & \cdot & c_{10,10} \end{bmatrix}$$

```

Var A,B,C : Array[1..10,1..10] Of LongInt;
    i,j,Am,An,Bm,Bn,sonuc : Integer;
Begin
Repeat
    WriteLn('Toplamı yapılacak matrisin boyutlarını giriniz : ');
    Write('A matrisinin satır sayısı = ? '); ReadLn(Am);
    Write('A matrisinin sütun sayısı = ? '); ReadLn(An);
    Write('B matrisinin satır sayısı = ? '); ReadLn(Bm);
    Write('B matrisinin sütun sayısı = ? '); ReadLn(Bn);
    If (Am=Bm) AND (An=Bn) Then sonuc:=1
        Else WriteLn('Bu iki matris toplanamaz.Boyutlari uyusmuyor! ');
Until sonuc = 1 ;

For i:=1 To Am Do
    For j:=1 To An Do
        Begin Write( 'A[ ' , i , ', ' , j , ' ]= ? ' ); ReadLn ( A[i,j] ); End;
For i:=1 To Bm Do
    For j:=1 To Bn Do
        Begin Write( 'B[ ' , i , ', ' , j , ' ]= ? ' ); ReadLn ( B[i,j] ); End;

For i:=1 To Am Do
    For j:=1 To An Do
        Begin
            C[i,j]:=A[i,j] + B[i,j] ;
            WriteLn( 'C[ ' , i , ', ' , j , ' ]= ' , C[i,j] );
        End;
End.

```

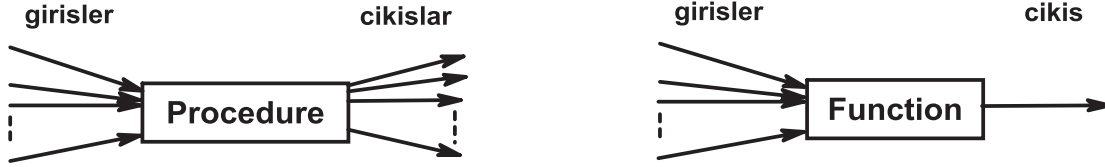
ALT PROGRAMLAR

Programlama dillerinde alt programların kullanılma amaçları;

1. Programlar kısılır: Tekrarlanan program parçaları alt program ile tanımlanarak sadece bir kez yazılır.
2. Programlar takip etmek kolaylaşır: Benzer işi yapan komutlar bir alt program içinde tanımlandığında programı takip etmek kolaylaşır.
3. Yazılan programlarda hata yapma olasılığı azalır.

Herhangi bir alt program çağrıldığında (ana programdan veya başka bir alt programdan) o alt programdaki bütün komutlar çalışır. Daha sonra işlem kendisini çağırın satıra veya bir sonraki satıra geçer.

Pascal programlama dilinde *Procedure* ve *function* olmak üzere iki farklı altprogram yapısı bulunur.



Bu iki altprogram arasındaki temel fark; procedure kendini çağırın satıra 0,1,2,...,n , function ise sadece 1 değer gönderebilir.

Procedure Alt Programları

Parametresiz Procedure: Ana programdaki değişkenler alt programdaki değişkenler olarak kullanılır. Ana program alt programa parametre(değişken) aktarımı yapamaz. Altprogramın çağrılması için sadece adı yazılır.

Procedure

Kullanımı →

Procedure Altprogram adı;

Tanımlamalar: → **Yerel tanımlama, sadece tanımlandığı alt programda geçerli**

Begin

·
·
·

End;

Örnek:

```
Program Prosedur_Ornegi;  
  Procedure IlkProsedur;  
  Begin  
    WriteLn('İlk proseduru isletiyorum !... ');  
  End;  
Begin {Ana Program}  
  IlkProsedur;  
  WriteLn('Su anda ana programdayım !... ');  
  IlkProsedur;  
End.
```

Örnek: Klavyeden girilen bir sayının küpünü parametresiz procedure yapısı ile bulan program;

```
Uses crt;  
Var x, kup : integer;  
  procedure kupbul;  
  begin  
    kup:=x*x*x;  
    writeln(x, ' sayisinin küpü=',kup);  
  end;  
  
begin  
  clrscr;  
  write('Ucuncu kuvveti bulunacak sayı=');readln(x);  
  kupbul;  
end.
```

Örnek : Ekrandan girilen yarıçap ve isteğe bağlı olarak dairenin alanını veya çevresini hesaplayan prosedürlü programı yazınız.

```
Program ProsedurDenemesi;
Var  r,Alan,Cevre : Real;
secim : Integer;
  Procedure AlanHesabi;
  Begin
    Alan := 3.14 * r * r ;
    WriteLn (' Alan = ', Alan :6:2 );
  End;

  Procedure CevreHesabi;
  Begin
    Cevre := 2 * 3.14 * r ;
    WriteLn (' Çevre = ', Cevre :6:2 );
  End;

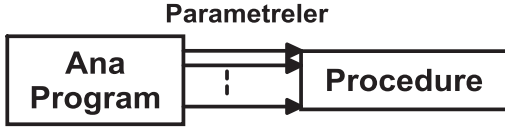
Begin {Ana Program}
  Write ('Yarıçapı giriniz = '); ReadLn( r );
  WriteLn ('Alan Hesabı İçin 1 ');
  WriteLn ('Çevre Hesabı İçin 2 ');
  Write (' Seciminiz = ');  ReadLn( secim );

  Case secim Of
    1 : AlanHesabi;
    2 : CevreHesabi;
  Else WriteLn (' Yanlış Seçim !... ');
  End;
End.
```

Parametrel Procedure

Ana programla altprogram arasında parameter (değişken) aktarımı vardır. Bu tip altprogramlar tanımlanırken ve çağrılırken ismi yanında parametreleri verilmelidir.

Tek taraflı çalışan Parametrel Procedure: Ana programdaki değişkenler alt programdaki değişkenlere aktarılır. Transfer edilen değişkenleri tipleri mutlaka aynı olmalıdır.



Kullanımı →

Procedure *Altprogram adı* (*parametreler:tip*);

Tanımlamalar; → **Yerel tanımlama, sadece tanımlandığı alt programda geçerli**

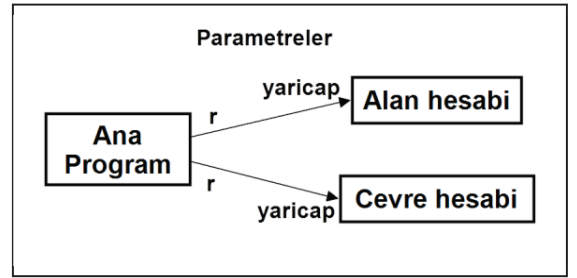
Begin

-
-
-

End;

Bir önceki örneğin Prosedüre Parametre gönderilerek çözümü...

```
Program ParametreGonderiliProsedur;  
Var r : Real;      { Alan ve Çevre değişkenleri silindi }  
    secim : Integer;  
  
    Procedure AlanHesabi( yarıcap : Real ) ;  
    Var Alan : Real ;  
    Begin  
        Alan := 3.14 * yarıcap * yarıcap ;  
        WriteLn ( ' Alan = ', Alan :6:2 );  
    End;  
  
    Procedure ÇevreHesabi( yarıcap : Real );  
    Var Çevre : Real ;  
    Begin  
        Çevre := 2 * 3.14 * yarıcap ;  
        WriteLn ( ' Çevre = ', Çevre :6:2 );  
    End;  
  
Begin  
    Write ('Yarıçapı giriniz = '); ReadLn( r );  
    WriteLn ('Alan Hesabı İçin 1 ');  
    WriteLn ('Çevre Hesabı İçin 2 ');  
    Write ('Seciminiz = ');    ReadLn( secim);  
  
    Case secim Of  
        1 : AlanHesabi ( r );  
        2 : ÇevreHesabi ( r );  
        Else WriteLn ( ' Yanlış Seçim !... ');  
    End;  
End.
```



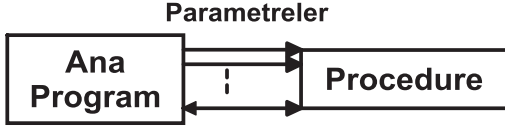
Örnek :

```
Program prosedur;  
Var x,y : Integer;  
  
  Procedure Deneme ( x,y : Integer);  
  Var z:Integer;  
  Begin  
    z:=y;  
    y:=x;  
    x:=z;  
    WriteLn (x,y);  
  End;  
  
Begin  
  x:=5;  
  y:=2;  
  WriteLn (x,y);  
  Deneme (x,y);  
  WriteLn (x,y);  
End.
```

Monitör



Çift taraflı çalışan Parametrelili Procedure: Ana programdaki parametreler alt programdaki parametrelere (tek taraflı) aktarılır. Bu parametrelere göre alt programdaki bulunan sonuçlar çift taraflı parametreler ile ana programda kendini çağıran satıra geri gönderilir. Procedure tanımlamasında kullanılan *var* kelimesi o parametrelerin çift taraflı olduğunu gösterir.



Kullanımı→

Procedure Altprogram adı (değişkenler1 : tip , *var* değişkenler2 : tip);

Tanımlamalar; →Yerel tanımlama, sadece tanımlandığı alt programda geçerli

Begin

.

.

.

End;

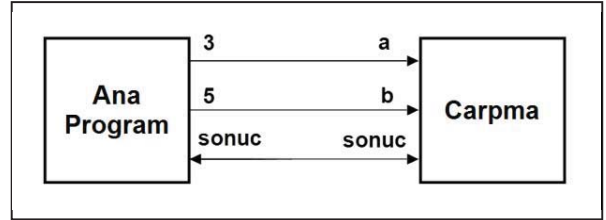
Örnek :

```
Program ProsedurCarpma;  
Var x,y,sonuc : Integer;
```

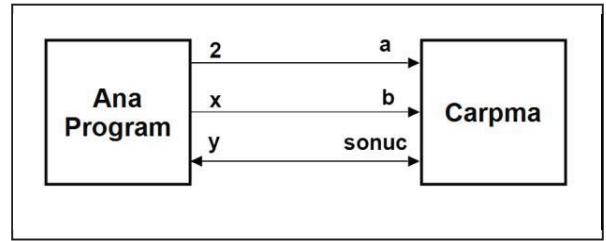
```
Procedure Carpma ( a,b : Integer; Var sonuc :Integer);
```

```
Begin  
sonuc := a * b ;  
End;
```

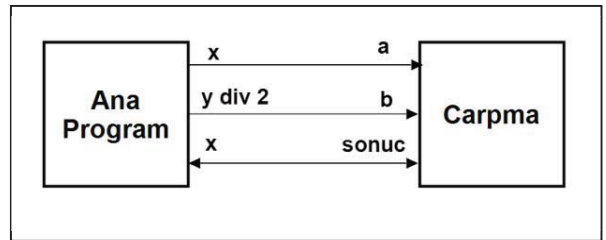
```
Begin  
Carpma(3 , 5 , sonuc);  
WriteLn (sonuc);
```



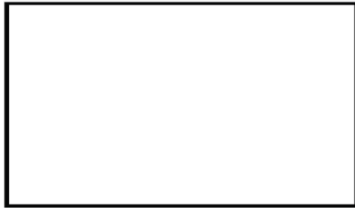
```
x:=3;  
y:=8;  
Carpma(2 , x , y);  
WriteLn (y);
```



```
Carpma(x, y Div 2 , x);  
WriteLn (x);  
End.
```



Monitör



FONKSİYON ALT PROGRAMI (FUNCTION)

Matematik fonksiyonlara benzer (sqrt, abs, round, ..) komutlar hazırlamak için kullanılır. Fonksiyonun prosedürden en önemli farkı, fonksiyon daima kendini çağıran satıra bir sonuç bildirir. Gönderilecek değer fonksiyon adına atanır.

Kullanımı→

Function Altprogram adı (değişkenler : tip) : tip;

Tanımlamalar; →Yerel tanımlama, sadece tanımlandığı alt programda geçerli

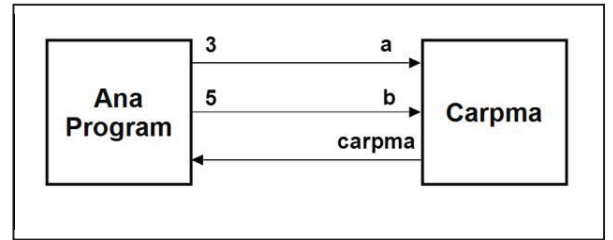
Begin

•
•
•

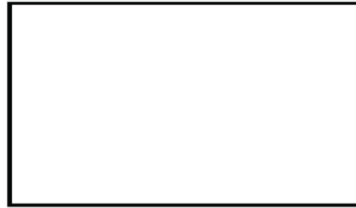
End;

Örnek:

```
Program CarpmaFonksiyonu;  
Var x,y : Integer;  
Function Carpma ( a,b : Integer ) : Integer;  
Begin  
    Carpma := a * b ;  
End;  
  
Begin  
    WriteLn (Carpma( 3 , 5 ));  
    x:=3;  
    y:=8;  
    y := Carpma( 2 , x );  
    WriteLn (y);  
    WriteLn (Carpma ( x, y Div 2 ));  
End.
```



Monitör



Ne Zaman Prosedür ?... Ne Zaman Fonksiyon ?...

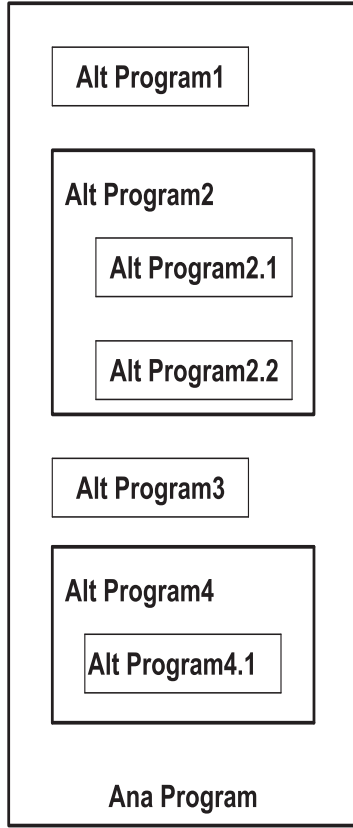
--Ana programa sadece tek deęer dönecek (gönderilecek) ise Fonksiyon kullanılır.

--Alt programın belirli bir hizmeti gerçekleştirip sonuç bildirmesi gerekmiyorsa veya birden çok sonucu bildirmesi gerekiyorsa Prosedür kullanılmalıdır.

İç İçe Birbirini Çağırın Alt Programlar

Alt Programların Birbirini Çağırma Şartları

1. Ana programın alt programları ana program tarafından çağrılabilir.
2. Kodlanış sırasına göre aşağıdaki alt programlar, kendilerinin üzerinde kodlanmış alt programları çağırabilirler. Ancak üstteki alt programlar aşağıdaki alt programları çağıramazlar. (Forward komutu kullanılmadığı müddetçe.)
3. İç içe alt program tanımlandığında içteki alt programlar ancak o alt program içindeki diğer alt programlardan (Kural 2) ve ana alt programdan çağrılabilir. Ana programın diğer alt programları ve ana program tarafından çağrılmaz.
4. Bir alt program kendini tekrar çağırabilir. Buna özyineleme (recursion) adı verilir.



	AltPrg 1	AltPrg 2	AltPrg 3	AltPrg 4	AltPrg2.1	AltPrg2.2	AltPrg4.1
Ana Program	√	√	√	√	×	×	×
AltProg 1	√	×	×	×	×	×	×
AltProg 2	√	√	×	×	√	√	×
AltProg 3	√	√	√	×	×	×	×
AltProg 4	√	√	√	√	×	×	√
AltProg 2.1	√	√	×	×	√	×	×
AltProg 2.2	√	√	×	×	√	√	×
AltProg 4.1	√	√	√	√	×	×	√

Örnek :

```
Program ic_ice_altprogramlar;
Var b : Real;

[ Procedure Toplama ( a : Integer; b : Real ; Var toplam : Real );
  Begin
    toplam := a + b ;
  End;

[ Procedure Carpma ;
  Var a : Real;

  [ Function TamCarpma(a,b:Integer):Integer;
    Begin
      TamCarpma := a * b ;
    End;

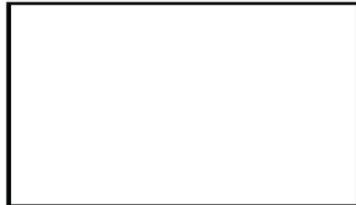
  [ Function ReelCarpma(a,b:Real):Real;
    Begin
      Toplama(2 , b , b);
      ReelCarpma := a * b ;
    End;

  Begin { Prosedür Çarpma }
    WriteLn ( TamCarpma(5,7) );
    Toplama ( 3, 6.2 , a );
    WriteLn ( a : 6 : 2 );
    WriteLn ( ReelCarpma( 2.1 , 4.2 ) : 6 :2 );

  End;

Begin { Ana Program }
  Carpma;
  Toplama ( 2, 3 , b );
  WriteLn ( b : 6 : 2 );
End.
```

Monitör



Rastgele Sayı Üretimi (Random)

$I := \text{Random}(\text{SonDeger});$ $0 \leq X < \text{SonDeger}$ aralığında rastgele bir tam sayı üretir ve I'ya atar.

$N := \text{Random};$ $0 \leq X < 1$ aralığında rastgele bir reel sayı üretir ve N reel değişkenine atar.

$B := \text{Random}(20);$ $0 \leq X < 20$ aralığında rastgele bir tam sayı üretir ve B'ye atar.

Sayı Üreticisinin Başlatılması (Randomize)

Randomize ;

Bilgisayarın sayı üreticisini sistem saatine göre rastgele bir sayı ile başlatır. Bu komut kullanılmasa da Random fonksiyonu ile rastgele sayı üretilir. Ancak programın her çalıştırılmasında ilk olarak hep aynı rastgele sayı ile başlanır.

Örnek : Sayı tahmin oyunu.

```
Program Oyun;
Uses Crt;
Var Uretilen,Sayi : Integer;

Function Karar (Uretilen, Sayi: Integer): Boolean;

    Procedure Mesaj (Uretilen : Integer);
    Begin
        If Sayi > Uretilen
            Then WriteLn('Buyuk Sayi Girdiniz...')

                Else WriteLn('Kucuk Sayi Girdiniz...');

        Karar := False;
    End;
    Begin
        If Sayi = Uretilen
            Then Karar := True
            Else Mesaj (Uretilen);
        End;

Begin
    ClrScr;
    Randomize;
    Uretilen := Random(101);
    WriteLn('Bu bir sayi tahmin oyunudur...');

    WriteLn('Tahmin edeceginiz sayi 0-100 arasinda olmalidir...');

    Write('Tahmin Ettiginiz Sayiyi Giriniz = ? '); ReadLn(Sayi);

    While NOT(Karar(Uretilen,Sayi)) Do

        Begin
            Write('Tahmin Ettiginiz Sayiyi Giriniz = ? ');

                ReadLn(Sayi);
            End;
        WriteLn(' Tebrikler Basardiniz...');
        ReadLn;
    End.
```

DOSYALAMALAR

Programlama bilgilerin her zaman ekrana yazdırılması veya deęişkenlerde tutulması yeterli olmayabilir. Programın alıřması sonucu girilen yada hesaplanan her bilgi manyetik ortama programda tanımlanan dosya adı altında saklanır. Bilgiler, daha sonra yeni bilgilerin eklenmesi, kaydedilen bilgiler üzerinde deęişiklik yapmak, silmek yada görüntülemek için manyetik ortama yazılırlar. Bilgilerin manyetik ortama yazılmaları farklı yollarla yapılır, bu yollar bilgilere erişim şekliyle adlandırılır. Pascal programlama dili içinde text ve binary tipinde dosyalama yöntemi vardır.

Text Tipte Dosyalama

Bilgilerin manyetik ortama sıralı olarak kaydedilmesine ve okutulmasına dayanan dosyalama şeklidir. İlk yapılan kayıt dosyanın başında son kayıt ise dosyanın sonunda yer alır. Bir bilgiye erişmek için, o kayda kadar bütün kayıtların okutulması gerekir. Çok kayıtlı bilgilerden oluşacak bir dosyalamada text tipi dosyalama tercih edilmez çünkü erişim yavaş olur. Derleyici text dosyaları için girilen bilgilere ana bellekte 128Byte 'lık tampon bellek açar. Bilgiler önce tampon belleğe daha sonra manyetik ortama aktarılır. Aynı şekilde manyetik ortamdan okunacak bilgiler tampon belleğe alınıp okunabilir.

Text Tipteki Dosyaların Tanımlanması

Pascalda oluşturulacak dosyalar deęişkenler gibi programın başında tanımlanır.

Var

Dosya_deęişkeni : text; //tanımlaması, programın içinde text dosyanın kullanılacağını ve işlem yapılacak dosyanın isminin aktarıacağı deęişkeni tanımlar.

Örnek ;

Var

Dosya : text ;

Assign Komutu

Text dosyanın manyetik ortamda hangi isimde bulucađını ve bu dosyanın dosya_deđiřkenine aktarılmasını sađlar (assign=devretmek). Her seferinde dosyanın yolu, ismi ve uzantısı kullanmak yerine dosya_deđiřkenini kullanabilmemizi sađlar. Ayrıca bu komut dosyayı hazır hale getirir (Dosyayı oluřturmaz)

Assign(dosya_deđiřkeni, 'Manyetik ortamdaki ismi ve uzantısı');

İsim en fazla 8, uzantı ise en fazla 3 karakter uzunluđunda olabilir.

Assign(dosya, 'kayit.txt'); //yolu verilmediđi için alıřılan dizinde dosya oluřturulaçaktır.
(c:\dev-pas)

Assign(dosya, 'c:\giris.dat');

Assign(dosya, 'd:\belgelerim\rehber.txt');

Rewrite Komutu

Text dosyayı ilk-kayıt amacıyla açmak için kullanılır. Eđer dosya daha önceden varsa içindeki bütün kayıtlar silinir.

rewrite(dosya_deđiřkeni);

Append Komutu

Daha önce oluřturulmuř Text dosyayı ek-kayıt amacıyla açmak için kullanılır. Eđer dosya daha önceden yoksa hata oluřur ve alıřmakta olan program o satırda sona erer.

append(dosya_deđiřkeni);

Reset Komutu

Daha önce oluřturulmuř Text dosyadan bilgi okumak amacıyla açmak için kullanılır. Eđer dosya daha önceden yoksa hata oluřur .

reset(dosya_deđiřkeni);

Close Komutu

Herhangibir sebeple açılan bir Text dosyayı kapatmak için kullanılır. Açılan her dosya işlem bitince mutlaka kapatılmalıdır. Mesela dosya rewrite veya append durumunda açılırsa close komutu kullanılmassa, dosyaya yazdırılması gereken veriler yazılamaz, close ile birlikte yazılır. (close ile Tampon bellek->manyetik ortam)

close(dosya_değişkeni);

dosyaya bilgi yazmak için -> writeln(dosya_değişkeni,string veya değişken(ler));//Bu komutun çalışabilmesi için dosyanın rewrite veya append ile açılmış olması gerekir.

dosyadan bilgi okumak için-> readln(dosya_değişkeni, değişken(ler));// Bu komutun çalışabilmesi için dosyanın reset ile açılmış olması ve eof sorgusunun true olması gerekir.

Read(dosya_değişkeni,char değişken); //dosyadaki karakterleri tek tek okur,

Readln(dosya_değişkeni,string değişken); //dosyadaki tüm satırı okur.

Ornek : Klavyeden girilen string ile dosya ismi atama;

var

dosya :text;

filename:string;

hata : byte;

begin

write('Dosya adini giriniz:');

readln(filename); {dosyanın ismi ve tam yoluda girilebilir.}

assign(dosya,filename+'.txt'); {bu sadece dosya degiskenine isim atama, bu komutla hdd de dosya olusmaz bunun icin rewrite mutlaka kullanılmalıdır}

rewrite(dosya);

close(dosya); {hangi sebeple olursa olsun acilan dosya mutlaka kapatılmalıdır.}

readln;

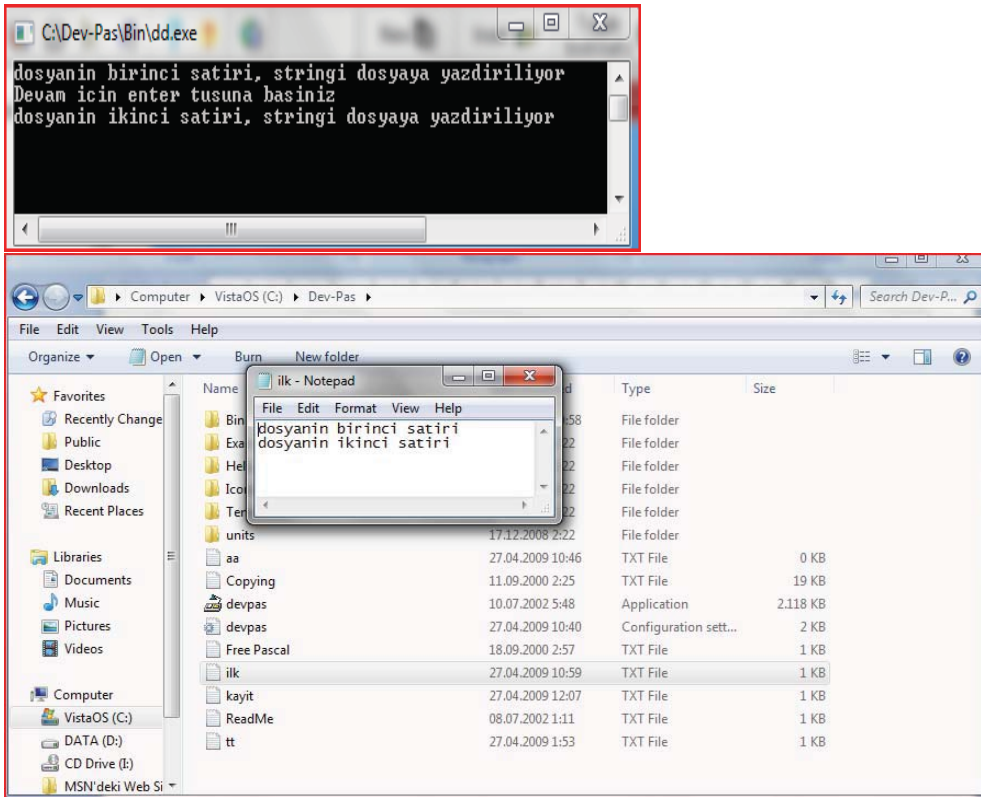
End.

Örnek : Bir text dosyası açılması ve içine ilk kayıt ve ek kayıtların yapılması;

```
uses crt;
var
dosya :text;
begin
clrscr;
assign(dosya,'ilk.txt');
  rewrite(dosya); //dosya ilk kayıt için açıldı
  writeln('dosyanın birinci satırı, stringi dosyaya yazdırılıyor');
  writeln(dosya,'dosyanın birinci satiri');
  close(dosya); // değişiklikler dosyaya kaydedildi ve kapatıldı, ilk kaydın sonu
  write('Devam için enter tusuna basınız');readln;

  append(dosya); //ek kayıt için dosya açılıyor
  writeln('dosyanın ikinci satırı, stringi dosyaya yazdırılıyor');
  writeln(dosya,'dosyanın ikinci satiri');
  close(dosya); //değişiklikler dosyaya kaydedildi ve kapatıldı, ek kaydın sonu
  readln;
End.
```

Ekran ve ilk.txt görüntüleri



eof Komutu

(eof=end of file) dosya sonu göstergesidir. Okuma amacıyla açılan dosyanın sonuna gelinip gelinmediğini anlamak amacıyla kullanılır. Son kayıttan sonra <eof> işareti vardır(Ekranla ilgili yapılan işlemlerin sonunda imleç () konumuyla çok benzerdir). <eof> işareti sorgulanırsa son kayıttan sonra true, diğer kayıtlarda false cevabı alınır.

eof(dosya_değişkeni)=True yada false

Ornek : Dosyadaki tüm kayıtları okuma

```
Var
F: text;
Ch : char;
Begin
Assign(f,'deneme.son');
Reset(f);
While not eof(f) do
Begin
Read(f,ch);
Write(ch);close(f);
End;
End.
```

erase Komutu

Text dosyayı manyetik ortamdan siler.

erase(dosya_değişkeni)

rename Komutu

Manyetik ortamdaki Text dosyanın adını değiştirmek için kullanılır.

rename(dosya_değişkeni, 'yeni isim ve uzantı');

```
assign(file, 'sinav.dat');
```

```
.
```

```
.
```

```
.
```

```
Rename(file, 'sonuc.dat');
```

Reset ve append durumunda oluşabilecek hatalar : Bu iki komutun kullanımı dosyanın olmaması durumunda programın *“File not found”* hatası vermesine neden olur. Bunu engellemek için;

{I-} direktifi :dosya işlemleri sırasına bir hata olursa bunu dikkate alma direktifidir. Mesela olmayan programı açmak istediğimizde, program o satırda kesilip “file not found” hatası vermesini engeller.

{I+} direktifi: giriş-çıkış hata kontrolünü tekrar aktif et.

Halt Komutu : Çalışmakta olan programı durdurup, işletim sistemine geçilir. Yani programı o satırda sonlandırır.

IOresult : Giriş-çıkış hatası oluşursa, hatanın kodunu verir. IOresult=2 ise ilgili dosya bulunamadı anlamındadır.

Örnek: Hata Kontrollu dosyadan veri okumak

```
Uses crt;
Var
T : text;
S : string;
Begin
Assign (t, 'd:\abc.def');
  {I-}
Reset(t);
  {I+}
If (IOresult<>0) then begin
Writeln('Acılmak istenen dosya yok');
Readln; halt;
End
Else
Begin
Readln(t,s);
Writeln('dosyanın ilk satırı:',s); close(t);
End; End.
```

Ornek : Bir firmanın bolum takibi yapilmak isteniyor. Bu amaçla Firmada çalışanların;

1. Unvani,

2. Adi soyadi,

3. Çalıştıkları bölümleri kaydeden, listeleyen, istenilen kayıtları silen, ek-kayıt yapan programı ana menu altında yazınız.

```
procedure kayit;
begin
clrscr;
assign(dosya,'d:\ilk.txt');
rewrite(dosya);
c:='E';
while c<>'H' do begin
clrscr;
write('Unvani.....');readln(un);
write('Adi Soyadi.....');readln(ad);
write('Bolumu.....');readln(bo);
writeln(dosya,un);writeln(dosya,ad);writeln(dosya,
bo);
repeat
write ('Devammi(E/H):');
readln(c);
c:=upcase(c);
until c in ['E','H'];
end;
close(dosya);
end;
```

```
procedure ekkayit;
begin
clrscr;
assign(dosya,'d:\ilk.txt');
{$i-}
append(dosya);I:=Ioresult;
{$i+}
if I<>0 then begin writeln('Dosya
yok');readln;halt;end;
c:='E';
while c<>'H' do begin
clrscr;
write('Unvani.....');readln(un);
write('Adi Soyadi.....');readln(ad);
write('Bolumu.....');readln(bo);

writeln(dosya,un);writeln(dosya,ad);writeln(dosya,
bo);
repeat
write ('Devammi(E/H):');
readln(c);
c:=upcase(c);
until c in ['E','H'];
end;
close(dosya);
end;
```

```

procedure listeleme;
var
s:integer;
begin

assign(dosya,'d:\ilk.txt');
{$i-}
reset(dosya);I:=Ioresult;
{$i+}
if I<>0 then begin writeln('Dosya
yok');readln;halt;end;
s:=1;
clrscr;
gotoxy(1,s);writeln('Unvani');
gotoxy(10,s);writeln('Adi Soyadi');
gotoxy(30,s);writeln('Bolumu');
s:=2;
while (NOT eof(dosya)) do begin
readln(dosya,un);readln(dosya,ad);readln(dosya,bo
);
s:=s+1;
gotoxy(1,s);writeln(un);
gotoxy(10,s);writeln(ad);
gotoxy(30,s);writeln(bo);

end;
writeln;writeln('devam icin enter tusuna
basiniz');readln;
close(dosya);

end;

```

```

procedure kayitsilme;

var
ara:string[20];
begin
assign(dosya,'d:\ilk.txt');reset(dosya);
assign(dosya1,'d:\gecici.txt');rewrite(dosya1);
//amac silinecek kayıt haric diger kayitlari
//yeni dosyaya yazmak, yeni dosyayi tekrar eski
dosyaya
//kopyalamak ve silmek
clrscr;
write('Silinecek Kisinin Adini ve soyadini
giriniz:');readln(ara);
while (NOT eof(dosya)) do begin
readln(dosya,un);readln(dosya,ad);readln(dosya,bo
);
if ara<>ad then begin
writeln(dosya1,un);writeln(dosya1,ad);
writeln(dosya1,bo);
end;
end;
close(dosya);close(dosya1);
//I.yol->erase(dosya);
rename(dosya1,'ilk.txt');end;
//II.yol->
reset(dosya1);rewrite(dosya);
while (NOT eof(dosya1)) do begin
readln(dosya1,un);readln(dosya1,ad);
readln(dosya1,bo);
writeln(dosya,un);writeln(dosya,ad);writeln(dosya,
bo);
end;
close(dosya);close(dosya1);erase(dosya1);
end;

```

```
uses crt;
var
dosya,dosya1 :text;
I : byte;
c:char;
un,ad,bo:string[20];
sec:char;

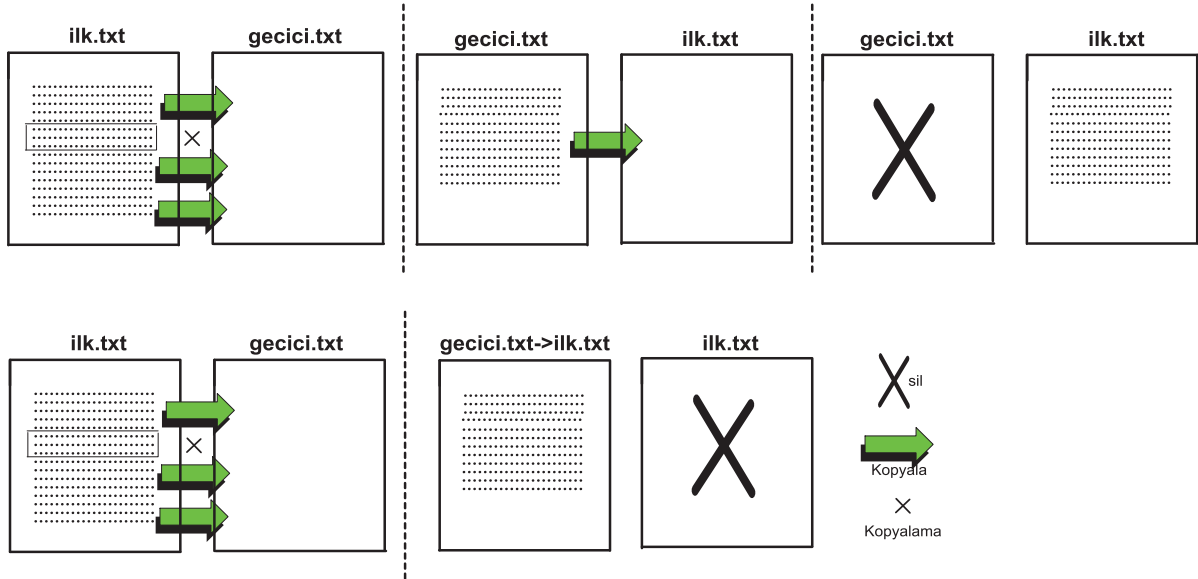
begin
repeat
clrscr;
//menu
    gotoxy(30,10);write('1-Kayit');
    gotoxy(30,12);write('2-Listeleme');
    gotoxy(30,14);write('3-Ek Kayit');
    gotoxy(30,16);write('4-Kayit silmek');
    gotoxy(30,21);write('seciminiz(cikis icin
esc):');
    repeat
    sec:=readkey;
    case sec of
    '1':kayit;
    '2':listeleme;
    '3':ekkeyit;
    '4':kayitsilme;
    end;
    Until sec In ['1'..'4',#27];

    until sec=#27;

End.
```

Kayıt Silme Yapıları

II.yol



I.yol

Örnek : Aşağıdaki program kodunun ekran çıktısı ve dosya içeriği nedir? (İlgili dosya daha önce oluşturulmuş ve içindeki veriler yazdırılmıştır).

```
uses crt;
var
t : text;
k,toplam: integer;
a: char;
kelime : string[15];
hata,i : integer;
.
begin
assign (t,'d:\tt.txt');
reset (t);
k:=0; toplam:=0;
while NOT (eof (t)) do
begin
read(t,a);
val(a,k,hata);
if (k=1) then
k:=5;
toplam:=toplam+k;
end;
close (t);
write(toplam);
append(t);
writeln(t,"");
str(toplam, kelime);
for i:=length(kelime) downto 1 do
write(t, kelime[i]);
close(t);
readln;
End.
```

Monitör



d:\tt.txt

121714

EK A : ASCII TABLOSU

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	\$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(72	48	H	104	68	h
9	09	Horizontal tab	41	29)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	
29	1D	Group separator	61	3D	=	93	5D]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	Ł	224	E0	α
129	81	ù	161	A1	í	193	C1	ł	225	E1	β
130	82	é	162	A2	ó	194	C2	Ṭ	226	E2	Γ
131	83	â	163	A3	ú	195	C3	ł	227	E3	π
132	84	ä	164	A4	ñ	196	C4	—	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	†	229	E5	σ
134	86	ã	166	A6	²	198	C6	‡	230	E6	μ
135	87	ç	167	A7	°	199	C7	‡	231	E7	τ
136	88	ê	168	A8	¿	200	C8	Ł	232	E8	φ
137	89	ë	169	A9	ƒ	201	C9	ƒ	233	E9	θ
138	8A	è	170	AA	¬	202	CA	Ł	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	ƒ	235	EB	δ
140	8C	î	172	AC	¼	204	CC	‡	236	EC	∞
141	8D	ì	173	AD	;	205	CD	=	237	ED	⊗
142	8E	Ë	174	AE	«	206	CE	‡	238	EE	ε
143	8F	Ā	175	AF	»	207	CF	Ł	239	EF	∩
144	90	É	176	B0	⋯	208	DO	Ł	240	FO	≡
145	91	æ	177	B1	⋮	209	D1	ƒ	241	F1	±
146	92	Æ	178	B2	⋭	210	D2	π	242	F2	≥
147	93	ó	179	B3		211	D3	Ł	243	F3	≤
148	94	ö	180	B4	†	212	D4	Ł	244	F4	[
149	95	ò	181	B5	‡	213	D5	ƒ	245	F5]
150	96	û	182	B6	‡	214	D6	π	246	F6	÷
151	97	ù	183	B7	π	215	D7	‡	247	F7	≈
152	98	ÿ	184	B8	ƒ	216	D8	‡	248	F8	°
153	99	Ö	185	B9	‡	217	D9	∟	249	F9	•
154	9A	Ü	186	BA		218	DA	ƒ	250	FA	·
155	9B	ø	187	BB	ƒ	219	DB	■	251	FB	√
156	9C	£	188	BC	Ł	220	DC	■	252	FC	²
157	9D	¥	189	BD	Ł	221	DD	■	253	FD	ˆ
158	9E	€	190	BE	ƒ	222	DE	■	254	FE	■
159	9F	f	191	BF	ƒ	223	DF	■	255	FF	□

EK B : ÖRNEK ÇALIŞMA SORULARI

- 1) Klavyeden eleman sayısı girilen bir kümenin belirtilen kombinasyonlarının sayısını hesaplayan programı yazınız.
(n elemanlı bir kümenin r’li kombinasyonlarının sayısı aşağıdaki formülden bulunmaktadır.)

$$C(n, r) = \binom{n}{r} = \frac{n!}{r!(n-r)!}$$

- 2) Klavyeden girilen bir A tamsayısının tam bölenlerini veren programı yazınız.
- 3) Klavyeden girilen bir sayının, belirtilen pozitif tam sayı üssünü alan programı yazınız.
- 4) Bir ve kendisinden başka tamsayıya tam bölünemeyen tamsayılara “asal sayı” denir. Buna göre klavyeden girilen üst sınıra kadar olan asal sayıları ekrana yazdıran programı yazınız.
- 5) Girilen bir pozitif tamsayının, iki sayının kareleri toplamı şeklinde yazılabilenleri hesaplayıp ekrana yazdıran programı yapınız.
- 6) Klavyeden girilen b tamsayısına göre $a^3 - a^2 = b$ şartını sağlayan $0 < a < 100$ arasındaki tam sayı veya sayıları bulup ekrana yazdıran programı yazınız.
- 7) Basamaklarının toplamı basamak sayısına eşit olan 100-999 arasındaki tamsayıları bulan programı yapınız. (Örnek : $102=1+0+2=3$, $111=1+1+1=3$ gibi)
- 8) Kendisi hariç bütün pozitif çarpanları (tam bölenleri) toplamı yine kendisine eşit olan sayılara “mükemmel sayı” denir. (Örnek : $6=1+2+3$, $28=1+2+4+7+14$ gibi) Buna göre,
- a) Klavyeden girilen bir tamsayının “mükemmel sayı” olup olmadığını kontrol eden programı,
b) 0 ile 999 arasındaki mükemmel sayıları bulup ekrana yazdıran programı yazınız.
- 9) Bir tam sayının en büyük bölenini bulup ekrana yazdıran programı yazınız.

10) Ekrandan girilen iki tam sayının Ortak Bölenlerinin En Büyüğünü (OBEB) bulan ve ekrana yazdıran programı yazınız.

11) Aşağıda verilen işlemin sonucunu, ekrandan girilen herhangi bir d tam sayısı için bulup, sonucunu ekrana yazan programı yazınız.

$$\sum_{k=1}^d \left(\frac{(d+k)!}{k!} + \frac{k!}{k} \right)$$

12) Aşağıda verilen işlemin sonucunu, ekrandan girilen herhangi bir d tam sayısı için bulup, sonucunu ekrana yazan programı yazınız.

$$\prod_{k=1}^d \left(\frac{(d-k)!}{k!} + \frac{k!}{k} \right)$$

13) Matrislerin boyutlarının ve elemanlarının ekrandan kullanıcı tarafından belirlendiği A ve B matrisin toplamını bulan Pascal programı yapınız. (Maksimum matris boyutu 10X10 olacaktır.)

14) Ekranın EN ÜST SATIRINA yazdıracağınız isim ve soyadınızın, ekranda 500ms.'lik aralıklar ile aşağı kayarak hareket edip, alt kenara geldikten sonra kaybolup tekrar en üst satırdan aşağı doğru kaydığı ve herhangi bir tuşa basılana kadar devam eden programı yapınız.

15) Klavyeden girilen; sıfırdan farklı, üç haneli bir tamsayıyı yazı ile ekrana yazdıran program yazınız.

- 16) Klavyeden girilen bir noktanın $ax+by+c=0$ doğrusu üzerinde olup olmadığını, eğer değil ise noktanın doğruya olan uzaklığını veren programı yazınız.
Bilgi Notu: $P(x_1, y_1)$ noktası doğru üzerinde ise $ax_1+by_1+c=0$ doğru denklemini sağlar. Eğer değilse de noktanın doğruya olan uzaklığı $u_z = |ax_1+by_1+c| / (a^2+b^2)^{1/2}$ formülü ile hesaplanır.
- 17) Her bir hanenin basamak sayısı kadar üstlerinin toplamı yine aynı sayıya eşit olan tam sayılara “Amstrong Sayıları” denir. Örneğin $153=1.1.1+5.5.5+3.3.3$ gibi. Buna göre 100-999 arasındaki Amstrong sayılarını bulan programı yazınız.
- 18) Kendi hanelerine tam bölünen 11 ile 99 arasındaki tam sayıları veren programı yazınız.
Örneğin : 24 sayısı hem 2'ye hem de 4'e tam bölünmektedir (15, 36 vb).
- 19) Basamaklarının toplamı basamak sayısına eşit olan 100 ile 999 arasındaki sayıları bulan programı yazınız. Örnek : 102 sayısı için $1+0+2=3$ dir.
- 20) Klavyeden girilen N elemanlı A dizisinin en büyük ve en küçük elemanlarını ve bu elemanların dizinin kaçınca elemanı olduğunu bulan programı yazınız.

EK C : ÖRNEK SINAV SORULARI

A'd Soyadı: No:

1) Aşağıdaki akış diyagramının çıktısı nedir? (15 Puan)



Monitör

Monitör

4) Aşağıdaki programda gerekli yerlere () koyunuz ve Program kodunun çıktısını monitörde belirtiniz? (10 Puan + 10 Puan)

```

Var A,B,C:Integer
Begin
  A:=18; B:=6; C:=9;
  If A>B Then
  Begin
    A:=A+B
    B:=B+C
    C:=-C+A
    WriteLn(A,'+',B,'+',C)
  End
Else
  Begin
    If B>C
    Then B:=B+C
    Else C:=C+A
    WriteLn(A,'+',B,'+',C)
  End
End
  
```

5) Aşağıdaki program kodunun çıktısı nedir? (20 Puan) (Boşlukları b harfi ile gösteriniz)

Monitör

```

Var k,m,n,p:Integer;
Begin
  m:=10; p:=2;
  Repeat
  While m>2 Do
  Begin
    WriteLn('B');
    m:=m-2;
  End;
  WriteLn('C');
  m:=10; p:=p+1; m:=m-p;
  Until p=5;
End.
  
```

2) Aşağıdaki programı kodunun çıktısı nedir? (20 Puan) (Boşlukları b harfi ile gösteriniz)

Monitör

```

Var d,g:Integer;
Begin
  For d:=1 To 10 Do
  Begin
    d:=d+2;
    For g:=3 DownTo 1 Do
    WriteLn('d:',g,'g:','g+6,' ');
    WriteLn('t');
  End;
End.
  
```

3) Aşağıdaki programa göre Degisken1 ve Degisken2 için sırasıyla 11 ve 8.4 girilirse programın monitör görüntüsü ne olur? (20 Puan) (Boşlukları b harfi ile gösteriniz)

Monitör

```

Var Degisken1:Integer;
    Degisken2:Real;
Begin
  WriteLn('iki sayı giriniz=');
  ReadLn(Degisken1,Degisken2);
  WriteLn(Degisken1 mod 2, 3*Degisken2/7-2);
  Degisken2:=Degisken1 - Degisken2/3;
  WriteLn(Degisken2/8:3);
End.
  
```

Ad Soyadı:

No:

B GRUBU

BİLGİSAYAR PROGRAMLAMA I VEZİ 2 SINAV SORULARI (İ.Öğretim)

1) Aşağıdaki program kodunun çıktısı nedir? (20 Puan) (Boşlukları b harfi ile gösteriniz)

Var E,sayı :integer;

Begin

E:=17; sayı:=23;

Case E Of

2,4,7..10: Case sayı Of

1..10: WriteLn(sayı-E);

11..20: WriteLn(E+sayı);

End;

11..20: Begin

If E In [15,18]

Then WriteLn('Sayı 15"dir.');

WriteLn(E*3, ' oldu');

End;

23,28,40,45..90: Write('sayı 23"e eşittir.');

End;

End.

İşinize değerli sorular.

Monitör

Monitör

Var A,B,C:integer

Begin

A:=18; B:=6; C:=9;

If A>B Then

Begin

A:=A+B

B:=B+C

C:=C+A

WriteLn(A, ' B, ', C)

End

Else

Begin

If B>C

Then B:=B+C

Else C:=C+A

WriteLn(A, ' B, ', C)

B:=B+C

End

WriteLn(A, ' B, ', C)

End.

2) Aşağıdaki program kodunun çıktısı nedir? (20 Puan) (Boşlukları b harfi ile gösteriniz)

Var d,g:integer;

Begin

For d:=1 To 10 Do

Begin

d:=d+2;

For g:=3 DownTo 1 Do

Write('G(', d, ', ', g, ') = ', g+6, '');

WriteLn('');

End;

End.

Monitör

3) Aşağıdaki programa göre Degisken1 ve Degisken2 için sırasıyla 11 ve 8,4 girilirse programın monitör görüntüsü ne olur? (20 Puan) (Boşlukları b harfi ile gösteriniz)

Var Degisken1:integer;

Degisken2:Real;

Begin

WriteLn('iki sayı giriniz');

ReadLn(Degisken1,Degisken2);

WriteLn(Degisken1 mod 2, 3*Degisken2:7:2);

Degisken2:=Degisken1 - Degisken2/3;

WriteLn(Degisken2:8:3);

End.

Monitör

5) Aşağıdaki program kodunun çıktısı nedir? (20 Puan) (Boşlukları b harfi ile gösteriniz)

Var k,m,n,p:integer;

Begin

m:=10; p:=2;

Repeat

While m>2 Do

Begin

Write('B');

m:=m-2;

End;

WriteLn('C');

m:=10; p:=p+1; m:=m-p;

Until p=5;

End.

Monitör

10) Diferensiyel verilen sorular

idi Soyadı :

no, Şube:
Aşağıdaki program kodunun çıktısı nedir? (15 Puan) (Boşluktan b harfi ile gösteriniz)

Monitör

```
Var a: Integer;  
Function Top (a:n,Integer):Longint;  
Var k:Integer;  
Begin  
a:=1;  
For k=1 To n Do  
s:=a*k;  
Top:=s;  
End;  
Function Ana(c:Integer):Integer;  
Var t:Integer;  
Begin  
t:=0;  
While s<=c Do  
Begin  
t:=t+s Mod 10;  
s:=s Div 10;  
End;  
Ana:=t;  
End;  
egin  
s:=30;  
WriteLn(Top(2,6));  
WriteLn(Ana(6));  
nd.
```

A SINAV SORULARI

Aşağıdaki program kodunun çıktısı nedir? (15 Puan) (Boşluktan b harfi ile gösteriniz)

Monitör

```
Var a,b,c:Integer;  
Function Dene(a,c:Integer):Integer;  
Var k:Integer;  
s:=Longint;  
Begin  
If a<c Then  
a:=a+c;  
Else a:=a-b;  
a:=1;  
For k=1 To c Do s:=a*c;  
WriteLn(s);  
WriteLn(a);  
Dene:=s+2;  
End;  
Begin  
a:=3; b:=4; c:=5;  
a:=Dene(b,a);  
WriteLn(a);  
End.
```

4) Dizi [15 9 7 6 16 4 1 7 3]

Yükarıdaki diziye göre aşağıdaki program kodunun çıktısı nedir? (15 Puan) (Boşluktan b harfi ile gösteriniz)

Monitör

```
Var Dizi : Array[1..8] Of Integer;  
k,p : Integer;  
Begin  
k:=3;  
p:=2;  
While (k<=8) Do  
Begin  
WriteLn(Dizi[k]);  
k:=k+1;  
p:=p+2;  
End;  
WriteLn(p);  
End.
```

Aşağıdaki program kodunun çıktısı nedir? (15 Puan) (Boşluktan b harfi ile gösteriniz)

Monitör

```
Var sayi,deger:Integer;  
eger:=10; sayi:=20;  
ileri:=deger:=deger-2; sayi:=sayi-2;  
Write(deger,' ');  
Else GoTo ilet;  
ileri:=deger-4; Then Go To son;  
son:=WriteLn(sayi,' ');  
sayi:=sayi Div 2;  
eger:=8;  
If sayi>2 Then GoTo ilet;  
End.
```

5) Aşağıdaki program kodunun çıktısı nedir? (15 Puan) (Boşluktan b harfi ile gösteriniz)

Monitör

```
Var k:Integer;  
Kalime:String[15];  
Begin  
Kalime:="ARABA";  
For k=1 To 6 Do  
If Kalime[k]#='A'  
Then  
Begin  
Insert(' ',Kalime,k+1); k:=k+1;  
End  
Else Begin  
Delete(Kalime,k,1);  
k:=k-1;  
End;  
WriteLn(Kalime);  
End.
```

Aşağıdaki program kodunun çıktısı nedir? (15 Puan) (Boşluktan b harfi ile gösteriniz)

Monitör

```
Var Dizi : Array[1..5] Of Integer;  
k,p,t: Integer;  
Begin  
Dizi[1]:=10; Dizi[2]:=6; Dizi[3]:=27;  
Dizi[4]:=21; Dizi[5]:=1;  
For k=1 To 4 Do  
For p=k+1 To 5 Do  
If Dizi[k]<Dizi[p] Then  
Begin  
t:=Dizi[k]; Dizi[k]:=Dizi[p];  
Dizi[p]:=t;  
End;  
For k=1 To 5 Do Write(Dizi[k],4);  
End.
```

Aşağıdaki program kodunun çıktısı nedir? (10 Puan) (Boşluktan b harfi ile gösteriniz)

Monitör

```
Type SayiKume Type= Set of Byte;  
Var A,B,C,D: SayiKume Tip;  
k:Integer;  
Begin  
A:=[]; B:=[]; C:=[]; D:=[];  
For k=3 To 12 Do  
Begin  
If k In C Then WriteLn(C:=k);  
If k In D Then WriteLn(D:=k);  
End;
```