1.2. Pascal alqoritmik dilinin əlifbası və identifikatorları

Hər bir alqoritmik dil əlifbadan, sözlərdən, ifadələrdən, operatorlardan ibarətdir. Əlifba xüsusi simvollar yığımıdır. Bu simvollardan müəyyən qaydalarla sözlər tərtib olunur. İfadələr müəyyən məna daşıyan sözlər yığımıdır. Operatorlar isə dilin cümlələridir. Bu operatorlardan proqram formalaşdırılır.

1. Əlifba.

Pascalda əlifba aşağıdakı simvollardan ibarətdir.

- Latm əlifbasının 26 böyük və kiçik hərifləri:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

a b c d e f g h i j k l m n o p q r s t u v w x y z

- _ Sətir altı xətt
- On ərəb rəqəmləri

0 1 2 3 4 5 6 7 8 9

- Xüsusi simvollar

 # $ ' ( ) : = . . + , - * / ' : ; < = > @ [ ] ^ { }

2. Identifikatorlar

Proqramlarda sabitlərin, tiplərin, dəyişənlərin, prosedur və funksiyaların, modulların, proqramların, "yazılarda" sahələrin adları identifikator adlanır.

İdentifikatorlar hərflər, rəqəmlər və "_" (simvol altı xətt) işarələr ardıcıllığı olub, hərflə başlayır. İdentifikatorlara probel

simvolu daxil ola bilməz. İdentifikatorlar istənilən sayda simvoldan ibarət ola bilər. Lakin yalnız ilk 63 simvol identifikatorlar kimi qəbul olunacaq, qalanlar nəzərə alınmayacaqdır.

Pascalda identifikatorlar:

- Standart (qabaqcadan təyin olunmuş)
- İstifadəçi ıılentifikatorlarına

aynhrlar.

Bütün standart prosedur və funksiyalarrı adları (məsələn, read, Write, Sin və s.), tiplərin adları (Integer, Real, və s.)standart identifikatorlardır. Bu adlardan istifadəçi yeni identifikatorlar kimi istifadə edə bilər. Lakin, onlar ilkin mənalarını itirəcəklər. Ona görə də məsləhət görülmür.

Söz ayırıcıları. Adi mətnlərdə xüsusi işarələr vasitəsilə cümlələr sözlərə ayrılır.Bu işarələr vergül, nöqtə, probel (boşluq), mötərizə və s. ola bilər. Pascal proqramlarında da operatorlar belə xüsusi işarələr vasitəsilə sözlərə- leksemlərə ayrılır. Bu işarələr "simvol - ayrıcıları" (символ разделителей) adlanır. Məsələn,

For i:=l to n do writeln(i);

əmri (cümləsi) aşağıdakı sözlərdən ibarətdir.

For , i:= , 1 , to , n , do , writeln , (, i ,) , ;

Burada simvol ayrıcısı kimi probeldən istifadə olunmuşdur. Bəzi yerlərdə isə sözlər arasında probel qoyulmamışdır. Məsələn, **i** ilə := arasında və ya **writeln** ilə (arasında və s. Bu onunla əlaqədardır ki, bəzi sözlər həmdə simvol ayrıcısı funksiyasını yerinə yetirir. Ona görədə belə

sözləri bir-birindən ayırmaq üçün əlavə olaraq xüsusi simvol ayncılarından istifadə olunmamışdır. Lakin istifadə etmək olar. Ona görədə yuxarıdakı əmri, mənasını dəyişmədən

For i: = 1 to n do writeln (i) ; şəklində də yazmaq olar.

Xüsusi simvollar.

Sözlər hesab edilən xüsusi simvollar da mövcuddur:

+ , - * / < = > [ ] ( ) { } , ^ @ #

Aşağıdakı qoşa simvollar da Pascal dilinin sözləridir;

:= •• <= >=(* *) (• •)

Xidməti sözlər.

Alqoritmik dillərdə bəzi sözlər xidməti (açar) sözlər adlanır. Xidməti sözlər Pascalda xüsusi təyinatlı sözlərdir. Onların mənaları dəyişdirilə bilməz. Ona görədə istifadəçi sözləri (identifikatorları) bu sözlərdən fərqli olmalıdır.

Xidməti sözlərə operatorların, tiplərin, proqram bölmələrinin adları, hesab və məntiqi əməllərin adları aiddir.

Pascalda sözlərdəki kiçik və böyük hərflər fərqləndirilmir. Yəni kiçik və böyük hərflər eyni hesab olunur. Məsələn,

PROGRAM, Program, program

sözləri eyni söz kimi qəbul olunur.

1.3. Proqramlarda verilənlər

Verilən kompüterin qəbul edə biləcəyi şəkildə formallaşdırılmış - hazırlanmış informasiyadır (faktdır). Pascalda yazılmış proqram, müxtəlif tipli verilənlər - obyektlər üzərində bir sıra əməliyyatları yerinə yetirir.

1. Tip anlayışı.

Tip dəyişənin ala biləcəyi qiymətlər çoxluğunu və onun üzərində yerinə yetiriləsi mümkün əməliyyatları göstərir. Pascalda müxtəlif tiplər vardır. Lakin onların əsasını- bazasını baza tiplər adlanan

- □ Integer (tam);
- □ Real (həqiqi);
- □ Char (simvol) ;
- □ Boolean (məntiqi)

tipləri təşkil edir. Bü tiplərə sadə tiplər, standart tiplər və s. də deyilir. Bu tiplər əsasında tərtib edilən digər tiplər də vardır.

2. Dəyişənlər və sabitlər

Proqramda verilənin hər bir elementi ya konstant (sabit), ya da dəyişən olur.

Hər bir dəyişən adı və tipi ilə verilir.

Konstantlar (sabitlər) sadə konstantlar və tipləşdirilmiş konstantlar şəklində ifadə oluna bilər. Sadə konstantlardan fərqli olaraq, tipləşdirilmiş konstantları təsvir edərkən həm

sabitin tipi, həm də qiyməti göstərilir. Faktiki olaraq tipləşdirilmiş konstantlar ilkin qiymətlərini almış dəyişənlərdirlər. Proqramın icrası zamanı tipləşdirilmiş konstantların qiymətləri dəyişdirilə bilər, sadə konstantların qiymətləri isə yox.

3. Ədədlər

Pascal proqramlarında tam onluq, tam on altılıq və həqiqi ədədlərdən istifadə olunur.

Tam ədədlər adi qaydada yazılır. Məsələn, -35 .307 və s.

Ədədin on altılıq say sistemində yazıldığını göstərmək məqsədilə ədədin qarşısında $ işarəsi qoyulur. Məsələn, $32, $E2, $A170 və s.

Həqiqi ədədlər iki şəkildə, ya adi onluq kəsr kimi, məsələn, 0.34, 127.098, -56.0 ya da 10 əsaslı üstlü ədədlər kimi yazılır. Belə yazılışlarda 10 əsasının yerinə böyük və ya kiçik "e" hərfi yazılır. Məsələn, 53.6e-4, -16.78E6 və s.

4. İfadələr

İfadə operandlar üzərində yerinə yetirilən əməllər ardıcıllığıdır. İfadələr operandların tiplərindən asılı olaraq riyazi, məntiqi və simvol (mətn) ifadələrinə ayrılırlar.

Riyazi ifadələrin tərtib olunmasında bir sıra standart funksiyalardan da istifadə olunur. Bu funksiyaların riyaziyyatdakı yazılışları ilə proqramlardakı yazılışları, adətən bir-birindən fərqlənir. Məsələn, proqramlarda funksiyaların arqumentləri həmişə mötərizədə yazılır.

Bu yazılışlar aşağıdakı cədvəldə bəzi funksiyalar üçün verilmişdir.

Riyaziyatdakı yazılış	**Proqramdakı yazılış**
I x I	Abs (x)
Sinx	Sin (x)
Cosx	Cos(x)
e^x	Exp(x)
Arctanx	Arctan(x)
arqumentin tam qiyməti	Int(x)
Lnx	Ln (x)
X^2	Sqr(x)
$\sqrt{x}$	Sqrt(x)
x^n	Exp(n*ln (x))

Cədvəldəki x^n=exp(n*ln(x)) yazılışın doğruluğunu aşağıdakı kimi göstərmək olar.

$$z= x^n \rightarrow lnz = n*lnx \rightarrow z = e^{n\cdot lnx} \rightarrow z=exp(n*ln(x))$$

4. Nişanlar (metkalar).

Bəzən proqramın hər hansı əmrinə digər əmr vasitəsilə müraciət etmək üçün bu əmrin yazıldığı sətir xüsusi nişanla qeyd olunur. Bu nişan (metka) həmin sətirdəki operatorlardan iki nöqtə ilə ayrılır. Nişan kimi 0- 9999 diapozonunda dəyişən natural ədəd və ya identifikator istifadə oluna bilər.

1.4. Turbo Pascalda proqramların strukturu

Turbo-pascalda proqramlar aşağıdakı strukturda tərtib alunur.

I Proqramın başlığı

program Proqramın_adı;

II Modullar siyahısı bloku

uses İstifadə olunan modullar siyahısı;

III Verilənlərin təsviri

label Nişanların siyahısı;

const Sabitlərin siyahısı;

type Tiplərin təsviri;

var Dəyişənlərin təsviri;

IV Prosedur və funksiyalar bölməsi

procedure Proqramçı tərəfindən tərtib olunan
function prosedurlar və funksiyalar

V Operatorlar bölməsi

begin

Operatorlar ;

end.

1.Proqramın başlığı

Proqramın başlığını yazmaq məcburi deyildir. Yazıldıqda isə ümumi şəkildə aşağıdakı şəkildə yazıla bilər.

program proqramın_adı (parametrlər);

Məsələn,

```
program Hasanov;
program PR1(Output) ;
program Getput(Input, Output) ;
```

Burada Hasanov, PR1, Getput proqrarnların adları, Input, Output isə standart sistem fayllarıdır. Onlardan əsasən printerdə çap əməliyyatını təşkil etmək üçün istifadə olunur.

2. Modullar siyahısı bloku

Pascalda da, digər alqoritmik dillərdə olduğu kimi, proqramları yazmaq üçün çoxsaylı standart prosedur və funksiyalar nəzərdə tutulmuşdur. Mənaca bir-birinə yaxın olan prosedur və funksiyalar ayrı-ayrı modullarda- kitabxanalarda cəmlənmişlər. Belə modullara misal olaraq

System, Strings, Winapi, Wincrt, Winprn

modullarını göstərə bilərik.

Belə modullardan istifadəçi də tərtib edə bilər.

Tərtib olunan proqramda, System modulundan başqa, bu modulların hansınasa daxil olan prosedur və funksiyalardan istifadə olunursa, həmin modulun adı "Modullar siyahısı bloku"unda elan olunmalıdır.

Məsələn,

uses Wincrt, Strings, Mylip;

Burada Wincrt, Strings standart rnodullar, Mylip isə istifadəçi tərəfindən tərtib olunmuş moduldur.

Əgər proqramda yalnız System moduluna daxil olan prosedur və funksiyalardan istifadə olunursa, belə halda proqrama uses bloku daxil edilməyə bilər.

Qeyd edək ki, System moduluna əksər proqramlar üçün ən zəruri vasitələr cəmlənmişdir. Bu bloka verilənlərin fayllara daxil və xaric etmə prosedurları, tirqonometrik və loqorifmik funksiyalar, kvadrata yüksəltmə, kökalma əməliyyatları, sətirlərin və faylların emalı funksiyaları və s. daxildir.

3. Təsvir etmə bloku

Bu blokda proqramda istifadə olunan dəyişənlərin, sabitlərin siyahıları və tipləri, nişanların siyahıları elan olunur.

Təsviretmə blokunu proqramda buraxmaq olar. Lakin bu zaman olduqca primitiv proqramları yazmaq mümkün olacaqdır.

Nişanların təsviri.

Ümumi yazılışı:

Label <Nişanlar siyahısı> ;

Proqram fraqmenti:

Label 1, Quit ;

```
. . . .
    Goto 1;
    . . . .
1: a:=l;
goto Quit ;
. . . .
Quit: end.
```

Konstantlar sadə konstantlar və tipləşdirilmiş konstantlara ayrılır. Onlar təsvirlərinə görə də bir-birlərindən fərqlənirlər:

Sadə konstantların təsviri:

```
Const
    A = 100;
    Max= 6.8 ;
    Flag= True ; B2= False ;
```

Tipləşdirilmiş konstantların təsviri:

```
Const
    A : integer= 100;
    Max : real = 6.8 ;   Min: real = - 42.0
    Flag:boolean =True ; B2 : boolean = False ;
```

Qeyd edək ki, sadə konstantları elan etdikdən sonra, proqramda dəyişmək olmaz, tipləşdirilmiş konstantları isə dəyişdirmək olar.

Dəyişənlərin təsviri.

Ümumi yazılışı:

```
Var
<Dəyişənlər siyahısı> : tip ;
```

Dəyişənin tipi kimi ya verilənlərin təsvirinin **type** bölməsində təyin olunmuş tiplərindən istifadə etmək olar, ya da tipin özünü. Məsələn,

```
Type Vector= array[1..100] of real ;
Var
   a, b : real ;
   F: Boolean ;
I, j : integer ;
Vectorl, Vector2 : Vector
```

4. Prosedur və funksiyalar bloku

Riyaziyyatda teoremləri isbat edərkən, bəzən teoremin isbatının müəyyən hissəsi ayrıca teorem- lemma şəklində isbat olunur. Sonra teoremin isbatında bu lemmadan bir, hətta bir neçə dəfə istifadə olunur. Bu isə teoremin isbatının gedişini sadələşdirir, həmdə isbatı anlaşıqlı edir.

Buna bənzər ideyadan proqramlaşdırmada da istifadə olunur. Belə ki , proqramın müəyyən hissəsi ayrıca prosedur və funksiya şəklində tərtib olunur. Sonra proqramda bu prosedur və funksiyalardan istifadə olunur. Bu da proqramı həm sadələşdirir, həm oxunaqlı edir, həmdə proqramdakı səhfləri tez aşkarlamağa imkan verir.

Proqramda əgər prosedur və funksiyalar varsa, onlar proqramın *prosedur* və *funksiyalar* blokunda yazılır.

Demək olar ki, prosedur və funksiyaların strukturu proqramların strukturu kimidir. Fərq ondadır ki, prosedur **procedure,** funksiya isə **function** başlıqları ilə başlayır.

5. Operatorlar bölməsi

Operatorlar proqramda nəzərdə tutulmuş nəticəni almaq məqsədilə verilənlərin emalını idarə edən təlimatlar- əmrlərdir. Hər bir proqramda proqramı təşkil edən bölmələrin hamsının olması heç də məcburi deyildir. Operatorlar bölməsi isə proqramın zəruri tərkib hissəsidir.

Əsas operatorlar

Hər bir proqraralaşdırma dilinin əsasını verilənlər və operatorlar təşkil edir. Operatorlar proqramlarda nəzərdə tutulmuş nəticəni almaq məqsədilə verilənləri emal edən təlimatlar- əmrlərdir. Bu operatorlar ardıcıllığı proqram adlanır. Proqramlarda operatorlar bir -birlərindən nöqtəli vergül (;) işarəsi ilə ayrılırlar. Hər bir operator qarşısında nişan (metka) qoyula bilər. Nişan operatordan iki nöqtə (:) işarəsi ilə ayrılır.

Pascalda operatorlar

■ Sadə operatorlar və

■ Mürəkkəb (strakturlu) operatorlar olmaqla iki qrupa ayrılırlar.

Sadə operatorlar elə operatorlara deyilir ki, onlara digər operatorlar daxil deyildir. Belə operatorlara misal olaraq, mənsubetmə operatorunu, şərtsiz keçid operatorunu misal göstərə bilərik.

Mürəkkəb operatorlar isə, bir neçə operator birləşməsidir. Şərti keçid və dövrü operatorlar belə operatorlardandır.

Mənsubetmə operatoru

Mənsubetmə operatoru vasitəsilə hər hansısa ifadənin nəticəsi və ya dəyişənin qiyməti, başqa bir dəyişənə mənimsədilir. Ümumi şəkildə mənsubetmə operatoru aşağıdakı şəkildə yazıla bilər:

İdentifikator := ifadə

Məsələn,

```
y:=x+1;        z:=x ;
z:= sin(x)-sqrt(abs(y-3));
 z:=12.6 ;
```

Bu operatorlar nəticəsində bərabərliyin (:=) sağ tərəfindəki ifadələrin qiymətləri hesablanılır və alınmış nəticələr bərabərliyin sol tərəfindəki dəyişənə mənimsədilir.

Mənsubetmə operatoruna verilən əsas tələb ondan ibarətdir ki, sağ tərəfdəki ifadənin tipi, sol tərəfdəki dəyişənin tipi ilə eyni olsun. Bəzi hallarda bu tələb pozula bilər. Məsələn, sol tərəf həqiqi, sag tərəf isə tam tipli ola bilər. Tərsi isə yol verilməzdir. Yəni tam tipli dəyişənə həqiqi tiplini mənimsətmək olmaz. Belə hallarda bir tipi başqa tipə çevirən operatorlardan istifadə etmək lazımdır.

2.1. Daxil etmə və çap etmə operatorları

1. Daxil etmə operatorları

Verilənləri Read, Readln və mənsub etmə operatoru vasitəsilə daxil etmək olar.

Bu operatorlar vasitəsilə dəyişənlərin qiymətləri klaviaturadan yaddaşa daxil edilir. Klaviaturadan daxil etmə operatorları ümumi şəkildə aşağıdakı kimi yazıla bilər.

Read(Dəyişənlərin siyahısı) ;_

Readln(Dəyişənlərin siyahısı) ;_

Məsələn, Read(a,b,c) ; Readln (a,b, c) ;

Bu operatorlar vasitəsilə klaviaturadan daxil olan qiymətlər-verilənlər a, b, c dəyişənlərinə mənimsədilir.

Read operatoru vasitəsilə verilənlər daxil olunduqdan sonra, kursor yeni sətirə keçmir. Readln operatoru vasitəsilə daxil olunduqdan sonra kursor yeni sətirə keçir.

Verilənləri klaviaturada yığarkən verilənlər arasında probel (boşluq) qoyulmalıdır.

Proqramlar tərtib olunarkən aşağıdakı qaydalar gözlənilməlidir:

Pascalda ancaq tam, həqiqi simvol və sətir tipli verilənləri daxiletmək mümkündür.

Ədədlər ancaq formatsız daxil edilir.

2. Xaric etmə operatorları

Bu operatorlar vasitəsilə deyişənlərin qiymətləri displeyin ekranına yazılır. Xaric etmə operatorlarına çox vaxt çap etmə operatorları da deyilir. Çap etmə operatorları ümumi şəkildə aşağıdakı kimi yazıla bilər.

Write (Dəyişənlərin siyahısı);

Writeln (Dəyişənlərin siyahısı) ;

Write operatoru vasitəsilə verilənlər xaric olunduqdan sonra, kursor yeni sətirə keçmir. **Writeln** operatoru vasitəsilə xaric olunduqdan sonra kursor yeni sətirə keçir.

Yadda saxlamaq lazımdır ki: - *Pascalda ancaq tam, həqiqi simvol, sətir və məntiqi tipli verilənləri çap etmək mümkündür.*

■ *Ədədlər formatlı, formatsız, sabit vergüllü və sürüşkən vergüllü şəkildə çap olunur.*

Verilənlər formatsız çap olunduqda, verilənlərin " çap görünüşünü" sistem, formatlı çapda isə proqramçının özü müəyyənləşdirir. Bu halda verilənlər daha oxunaqlı çap olunur.

Verilənlər formatsız çapa çıxarılanda, verilənlərin adları verilənlər siyahısında bir-birindən vergüllə ayrılmış şəkildə verilir. Formatlı çapda isə siyahıda, hər bir dəyişənin adı ilə yanaşı çap formatı da verilir. Bu zaman dəyişənin adı ilə çap formatı aşağıdakı şəkildə yazılır:

<Dəyişənin adı> : birinci tam ədəd : ikinci tam ədəd ;

Burada

■ Birinci tam ədəd çap üçün sahənin uzunlugunu (simvolların sayını);

- İkinci tam ədəd, yalnız çap olunan ədəd həqiqi ədəd olduqda yazılır və ədədin çapa çıxarılan kəsr hissəsindəki rəqəmlərin sayını göstərir;
- Əgər çap olunan ədəd həqiqi ədəddirsə, və yalnız birinci ədəd göstərilmişdirsə, ədəd sürüşgən vergüllü şəklində (yəni üstlü ədəd kimi) çap olunur. Bu halda birinci tam ədəd sürüşgən vergüllü ədədin mantissasındakı rəqəmlərin sayını;

göstərir.

Sürüşgən vergüllü təsvirdə, mantissa ilə tərtib arasında " E " simvolu çap olunur. Məsələn, -4.5E-05. Bu yazılış — $4.5 \bullet 10^{-5}$ ədədi deməkdir.

Program nümunəsi_:

```
program cap_l;
 Const
       i:Integer=12345;
       r:Real=-123.1234567  ;
       c:Char='$';
       b:Boolean=True;
       s : String='Pascal_Delphi';
begin
    Writeln ('  Formatsiz cap');
    Writeln(i,r,c,b,s);
     Writeln;
     Writeln('  Formatli cap');
       Writeln(i:5,r:10:3,c:5,b:8,s:15);
       Writeln;
      Writeln('  Sabit vergullu formatla cap');
      Writeln(r:3:0);
      Writeln(r:5:3);
      Writeln(r:12:3);
      Writeln;
   Writeln('  Suruskan vergullu formatla cap');
   Writeln(r:3);
    Writeln(r:5);
   Writeln(r:12);
```

```
    readln;
end.
```

Nəticə:

```
Formatsiz cap
12345-
 1.23123456700000E+0002$TRUEPascal_Delphi
Formatli cap
12345   -123.123      $ ,     TRUE   Pascal_Delphi
Sabit vergullu formatla cap
 -123
- 123.123
    -123.123
Sürüşkən vergullu formatla cap
1.2E+0002
 1.2E+0002
 1.231E+0002
```

Proqram nümunəsi 2: Tam, həqiqi, simvol və məntiqi tipli verilənlərin daxil edilməsi və çapı.

```
program cap_2;
const Log=True;
var
    K,L,M:integer; X,Y:real;
SI,S2,S3:Char;
begin
   Writeln('K,L,M - tam adadlarini daxil et');
    Readln(K,L,M);
   Writeln('K=',K:3, ' L=',L:5,' M=',m:2);
   Writeln('X,Y - hagigi adadlarini daxil et');
   Readln(X,Y);
   Writeln('X=',X:8:2,'  Y=',Y:6:3);
   Writeln('SI,S2,S3 - Simvollarini daxil et');
   Readln(SI,S2,S3);
   Writeln('S1=',S1,'  S2=',S2,'  S3=',S3);
{Simvollari asagidaki kimida cap etmak olar}
 {5 adadi simvolun capi ucun ayrilmis sahadir}
   Writeln('SI=' , SI: 5,'  S2=',S2:5,'  S3=',S3:5);
```

```
    readln;
    end.
```

Nəticə:

```
K,L,M - tam adadlarini daxil et
6
98
5
K=    6 L=     98 M= 5
X,Y - hagigi adadlarini daxil et
123.78
98.8
X=   123.78 Y=98.800
S1,S2,S3 - Simvollarini daxil et
ert
Sl=e S2=r S3=t
Sl=       e S2=       r S3=       t
```

Çap qurğusunda çapın təşkili.

Printerdə çap etmək üçün Pascalda xüsusi operator nəzərdə tutulmamışdır. Bu əməliyyatları Printer modulundan istifadə etməklə yerinə yetirmək olar. Bu zaman çap etmə prosedurları aşağıdakı şəkildə yazılmalıdırlar:

Write (Lst, Dəyişənlərin siyahısı) ;

Writeln (Lst, Dəyişənlərin siyahısı) ;

Məsələn,

```
Program Prinl ;
   Uses Printer ;
    Begin
Writeln(Lst,  ' Printerda cap')  ;
Writeln(  ' Ekranda cap  ')  ;
End.
```

Bu proqram çap qurğusunda " Printerda cap " , ekranda isə " Ekranda cap " sözlərini çap edəcəkdir. Eyni qayda ilə dəyişənlərin də qiymətlərini çap etmək olar.

Çap qurğusunda çap etmənin digər vasitələri də vardır. Bu vasitələr fayllarla əlaqədar olduğu üçün, bu haqda kitabçanın fayllar bölməsində məlumat veriləcəkdir.

Çap əməliyyatı zamanı ekranın idarə olunması.

Ekranın idarə olunması dedikdə

■ Ekranın təmizlənməsi - silinməsi;

" Kursorun ekranda yerinin dəyişdirilməsi

■ Ekran fonunun və şriftlərin rənglərinin dəyişdirilməsi

■ Və s.

kimi əməliyyatlar başa düşülür.

Bu kimi əməliyyatları yerinə yetirmək üçün Turbo Pascalda crt modulunda bir sıra prosedur və funksiyalar nəzərdə tutulmuşdur.

1. ClrScr - proseduru ekranı təmizləyir və kursoru ekranın sol yuxarı küncünə gətirir.

2. Ekranda normal rejimdə 25 sətir və hər sətirdə 80 simvol çapetmək olar. Ona görədə ekranın sol yuxarı küncünün koordinatları (0,0) saq aşağı küncünün koordinatları isə (24, 79) -dur.

GoTo (x, y) proseduru kursoru (x, y) nöqtəsinə gətirir.

3. Ekranm fonu TextBackGround(rəng) , şriftin rəngi isə TextColor(rəng) proseduru vasitəsilə dəyişdirilə bilər. Rəng parametri kimi rəngin ingilis dilindəki adından- Konstant-dan və ya aşağıdaki cədvəldə verilmiş, onu əvəz edən koddan istifadə etmək olar.

Rəng	Kod	Konstant
Qara	0	Black
Göy	1	Blue
Yaşıl	2	Green
Bürüncrəngi	3	Cyan
Qırmızı	4	Red
Yasəməni	5	Magenta
Şabalıdı	6	Brown
Ağ	7	White
Boz	8	LightGray
Mavi	9	LightBlue
Açıq yaşıl	10	LightGreen
Açıq bürüncvari	11	LightCyan
Açıq qırmızı	12	LightRed
Açıq yasəməni	13	LightMagenta
Sarı	14	Yellow
Parlaq ağ	15	White

Cədvəldəki ilk 8 rəngdən həm fonun, həm də şriftin rəngi kimi istifadə etmək olar. Qalan 7 rəngdən isə yalnız şriftin rəngi kimi istifadə etmək mümkündür.

Program nümunəsi_3:

```
program Ekran;
   uses crt;
    var B:byte;
```

```
begin
  ClrScr;
  Writeln('l-ci sətirdə 1-ci mövqedə cap');
   GotoXY(10,5);
   Writeln(10-ci movqedə 5-ci  sətirdə cap');
  TextColor(Red); TextBackGround(Green);
Writeln('Yasil fonda girmizi rangda cap');
B:=TextAttr;   { Cari rəng və fonu yadda saxlayir}
TextColor(14);  TextBackGround(Brown);
 Writeln('Boz fonda Sari rəngdə cap');
TextAttr:=B;{ Yadda saxlanilmis yasil fon və}
   {  sriftin girmizi rəngini bərpa edir }
 Writeln('Yenidən Yasil fonda girmizi rəngdə cap');
end.
```

2.2. Şərti keçid və Variant operatoru

Şərti keçid və Variant operatorundan budaqlanan strukturlu proqramların yazılışında istifadə olunur. Belə proqramlarda bəzən şərtsiz keçid adlanan operatora da müraciət olunur.

1. Şərtsiz keçid operatoru

Bu operator ümumi şəkildə aşağıdakı kimi yazılır:

goto nişan ;

Məsələn,

```
Label  10
. . . . . .
goto 10 ;
    . . . . . .
10 : s: = a+1 ;
```

Goto operatorundan sonra, qarşısında goto operatorunda göstərilən nişan olan operator və ondan sonrakı operatorlar yerinə yetirilir.

Bu operatora verilən əsas tələb ondan ibarətdir ki, goto operatoru vasitəsilə idarəetmənin ötürüldüyü operator (qarşısında nişan olan operator) eyni modul, prosedur və funksiya daxilində olmalıdır. Bu isə o deməkdir ki, goto operatoru vasitəsilə modula, prosedura və funksiyaya daxil olmaq və kənara çıxmaq olmaz.

2. Şərti keçid operatoru

Şerti keçid operatora iki formada : qısaldılmış və tam formalarda ola bilər.

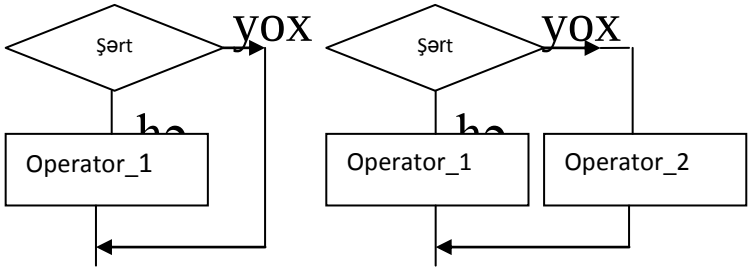
Qısaldılmış forma:

```
If şərt
        then   operator_l;
```

Tam forma:

```
If şərt
    then operator_l
    else  operator_2
```

Şərti keçid operatorlarını aşağıdakı sxemdəki kimi təsvir etmək olar.



a) Qısaldılmış şərti keçid operatoru

b) Tam şərti keçid operatoru

Qısaldılmış formalı şərti keçid operatorunda şərt ödənildikdə hər hənsısa operator yerinə yetirilir, ödənilmədikdə isə yox. Tam formalı şərti keçid operatorunda isə şərt ödənildikdə bir operator, ödənilmədikdə isə başqa operator yerinə yetirilir..

Həm qısaldılmış, həmdə tam formalarda mühüm bir ümumiləşmə ola bilər. Yuxarıdakı yazılış formalarına görə şərt ödənildikdə və ya ödənilmədikdə yalnız bir operator yerinə yetirilirdi. Bəzən tələb olunurki, bir operator deyil, bir qrup operator yerinə yetirilsin. Belə halda həmin operatorlar

qrupunun əvvəlinə begin, sonuna isə end sözlərini əlavə etmək kifayətdir. Belə halda, məsələn, tam şərti keçid operatoru aşağıdakı kimi yazılacaqdır:

```
If şərt begin
                operator_12;
                operator_1n ;
   end
   else begin
                operator_21;
                operator_2m ;
   end ;
```

Diqqət: Bu yazılışdakı birinci end operatorundan sonra " ; " işaresi qoyulmur.

Qeyd edək ki, Pascal proqramlarında begin və end operatorları sanki mötərizə funksiyasını yerinə yetirirlər. Mətinlərdə mötərizəni açandan dərhal sonra, və bağlamamışdan əvvəl vergül qoyulmadığı kimi, begin operatorundan da sonra nöqtəli vergül (;) işarəsi qoyulmur, end-dən əvvəl isə qoyula da bilər, qoyulmayada.

umumi şəkildə aşağıdakı kimi yazılır:

Case <Ifade>

Çalışma 2.1. y funksiyasının qiymətini hesablamalı:

$$y = \begin{cases} a + b, & \text{əgər } a > b \\ a - b, & \text{əgər } a < b \\ \text{hesablama aparmamalı}, & \text{əgər } a = b \end{cases}$$

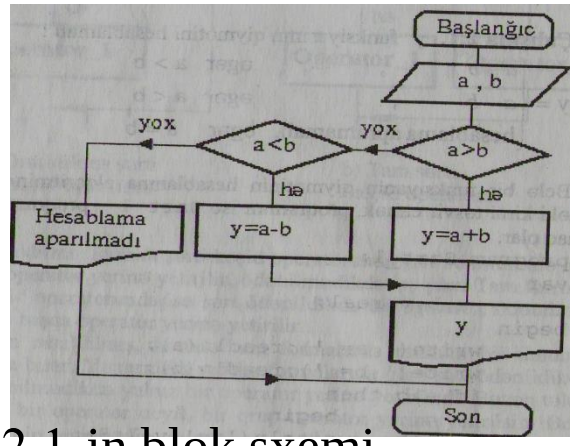
Belə bir funksiyanın qiymətinin hesablanma alqoritmini şekil 2.l-dəki kimi təsvir etmək, proqramını isə Shart_l proqramı kimi yazmaq olar.

```
program SHert_l;
var
    a,b,y:real;
begin
     write (' a= ');readln(a);
     write (' b= ');readln(b);
     if a>b then
                    begin
                    y:=a+b;   writeln('y=',y:7:3);
                    end
  else if a<b then
  begin
  y:= a-b; writeln('y=',y:7:3);
end
else Writeln('hesablama aparilmadi');
 readln;
end.
```

Nəticələr:

Variant_l: a= 12 Variant_2: a= 8
b= 5 b= 12
y= 17 .000. y= -4 . 000
Variant_3: a= 5
b=5

hesablama aparilmadi



Şəkil 2.1.Çalışma 2.1-in blok sxemi

Variant operatoru

IF şərti keçid operatoru iki variantdan birini seçməyə imkan verir. Varantların sayı çox olduqda lazımı varianti seçmək üçün **İF** operatorundan bir neçə dəfə istifadə etmək lazım gəlir . Bu da proqramın murəkkəbləşməsinə səbəb olur.

Belə çox variantlı seçməni yerinə yetirmək üçün Pascalda Case | variant operatoru nəzərdə tutulmuşdu. Bu operatora seçmə operatoru da deyilir.

Case operatoru of

Konstant_1 : operator_1;

Konstant_1 : operator_2;

. . . . . .

Konstant_n : operator_n;

Else

Operator

End ;

Burada ifadənin tipi yalnız sıra tipli ola bilər. İfadənin qiyməti yazılışdakı hər hansısa konstanta bərabər olduqda,

həmin konstantdan (nişandan) sonra yazılmış operator, əks halda else-dən sonra yazılmış operator yerinə yetiriləcəkdir.

Burada bir sıra halların olması mümkündür.

1) İfadənin qiyməti yalnız hər hansı konstantla üst-üstə düşdükdə müəyyən operatoru yerinə yetirmək lazım gəlir.Bu halda ümumi yazılışdakı ”else Operator” əmri buraxılmalıdlr.
2) *Konstantın bir neçə qiymətində eyni operator yerinə yetirilsin*.Bu halda həmin operatorun qarşısında bir konstant əvəzinə,bir birlərindən vergüllə ayrılan həmin konstantların hamsı yazılmallıdır.
3) İfadənin hər hansısa qiymətində bir deyil, bir qrup operator yerinə yetirmək lazım gəlsin. Belə halda, if operatorunda olduğu kimi həmin qrupun əvvəlinə begin sonuna isə end yazılmalıdır.

Nümunə:

```
1. case day of

    1, 2, 3, 4, 5: writeln (  'Dars gunu'  ) ;

                6 : writeln ('  Shanba gunu'  );

                7 : writeln ('Bazar gunu'  );

            End;

2. case day of

                1. .5: writeln ('  Dars gunu '  );

                6 : writeln ('  Sanba gunu'  );
```

7 : writeln (' Bazar gunu');

End;

3. **case** day of

6 : writeln ('Sanba gunu');

7 : writeln (' Bazar gunu');

End;

Bu proqram fraqmentlərinin nəticələri eyni olacaqdır.

Proqram nümunəsi: Həftənin günlərini rəqəmlə daxil edib cavabı sözlərlə çap etməli.

```
program Case_1
var Gun : integer ;
begin
   write (' Haftanin gunu-> ' ) ; readln (Gun) ;
      case Gun of
      1 : writeln ('Bazar ertesi ' ) ;
      2 : writeln ('Carsanba axsami ' ) ;
      3 : writeln ('Carsanba ' ) ;
      4 : writeln ('Cuma axsami ' ) ;
```

```
        5 : writeln ('Cuma gunu ' ) ;
        6 : writeln ('Sanba ' ) ;
        7 : writeln ('Bazar gunu ' ) ;
      end;
       readln ;
end.
```

Dövrü operatorlar

Dövrü proseslərin proqramlarını , if şərti keçid operotorundan istifadə etməklə yazmaq olar. Lakin bu zaman proqram müəyyən qədər mürəkkəbləşir.Ona görədə, Pascalda dövrü alqoritmilərin proqramlarını daha sadə şəkildə yazmaq üçün üç dövrü operator nəzərdə tutulmuşdur:

- Ön şərtli
- Son şərtli
- Parametrli.

Dövrü proseslər dövrlərin sayı qabaqcadan məlum olan və olmayan proseslərə ayrılırlar.

Ön və son şərtli operatorlardan əsasən dövrlərin sayı qabaqcadan məlum olmayan dövrü proseslərin proqramlarının yazılışında, parametrli dövrü operatordan isə dövrlərin sayı məlum olan prosseslərin proqramlarında istifadə olunur.

Ön şərtli dövrü aperotor

Ön şərtli dövrü aperator aşağıdakı variantlara yazıla bilər:

While şərt do və ya while While şərt do

Operator;

begin

Operator_1;

.....

Operator_n;

End;

Şərt ödənilənə qədər (true məntiqi qiymətinə malik olana qədər) birinci yazılışda bir operator, ikinci yazılışda isə, operatorlar qrupu yerinə yetirilir. Ön şərtli dövrü operatoru şəkil 2.2-dəki sxem şəklində təsvir etmək olar.

2.Son şərtli dövrü operator

Bu operator vasitəsilə dövrü proseslər təşkil olunduqda şərt dövrün sonunda yoxlanılır. Dövr şərt ödənilməyənə qədər (false məntiqi qiymətinə malik olduğu qədər- ön şərti dövrü operatorlarda isə tərsinə, şərt ödənilənə qədər) davam edir.

Son şərtli dövrü operator aşağıdakı şəkildə yazılır:

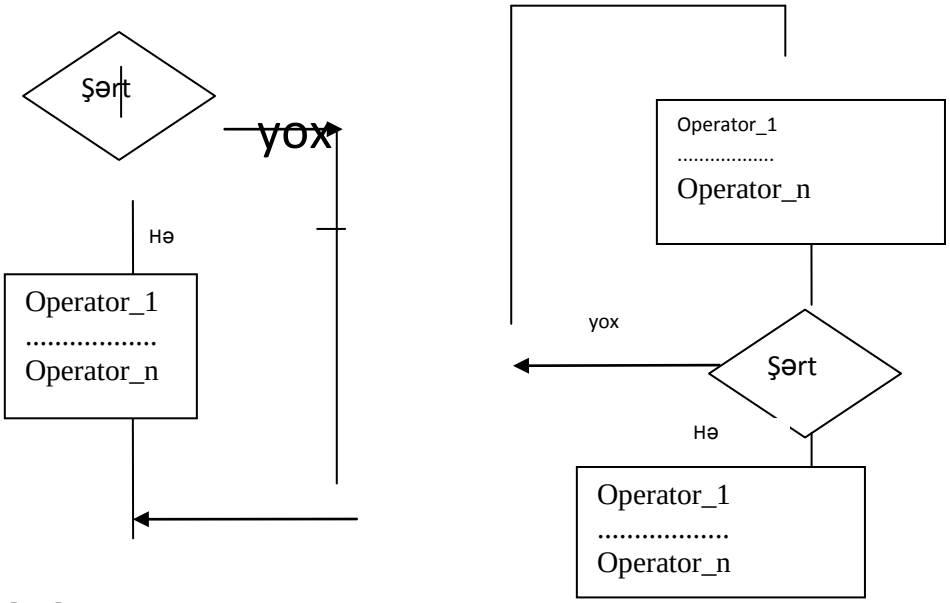
Repeat

Operator_1;

.....

Operator_n;

Until <şərt>;

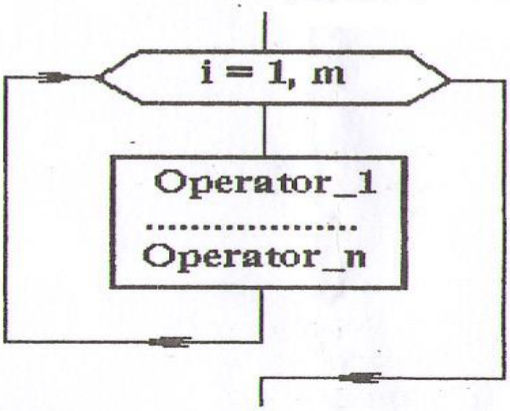


Şəkil 2.2

1. Parametrli dövrü operatorlar

Parametrli dövrü operatorun iki yazılış formiası vardır.

1) `for` parametr `:=` parametrin_ başlanğıc_ qiyməti
`to` parametrin_ son_ qiyməti
`do` operator ;



Parametrli dövrü operatorun blok - sxemdə təsviri

`for` parametr `:=` parametrin_ başlanğıc_ qiyməti
`downto` parametrin_ son_ qiyməti
`do` operator ;

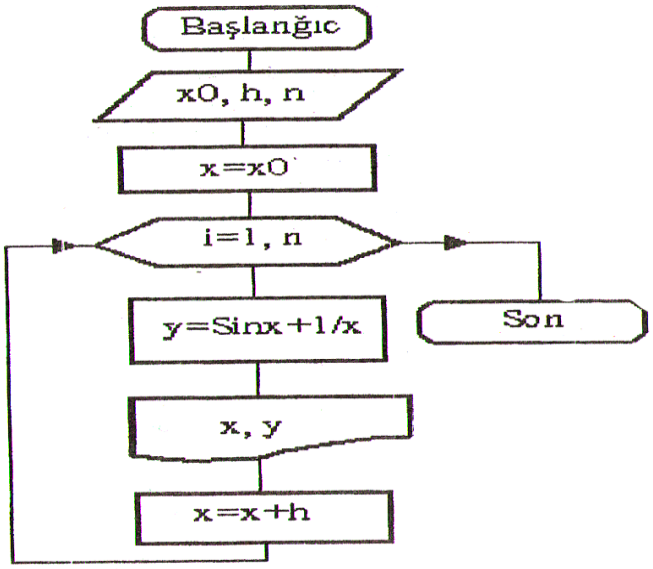
Bu yazılışdakı parametrlər sıra tipli (**порядкоаые**) olmalıdırlar. Dövrü proses parametr özünün son qiymətini alana qədər davam edir.

Birinci yazılışda parametrin qiyməti hər dövrdə bir vahid artır. İkinci dövrdə isə azalır. Ona görədə birinci yazılışda parametrin başlanğıc qiyməti, son qiymətindən kiçik, ikinci yazılışda isə böyük olmalıdır.

Deməli, `for` dövrü operatorunda dövr addımı ya +1, ya da −1 ola bilər.

`For` operatorunun yüxarıdakı yazılışlarında dövrü prosess yalnız bir operatordan ibarətdir. Dövrü prosess operatorlar qrupundan da ibarət ola bilər. Həmişə olduğu kimi bu haldada qrup `begin - end` " proqram mötərizəsinin" içərisində yazılmalıdır.

Çalışma 2.4: `y:=Sinx+1/x` - funksiyasının qiymətlərini `x0` nöqtəsindən başlamaqla `h` addımdan bir, `n` sayda nöqtədə hesablamalı.



Şəkil 2.6. Çalışma 2.4-ün blok- sxemi.

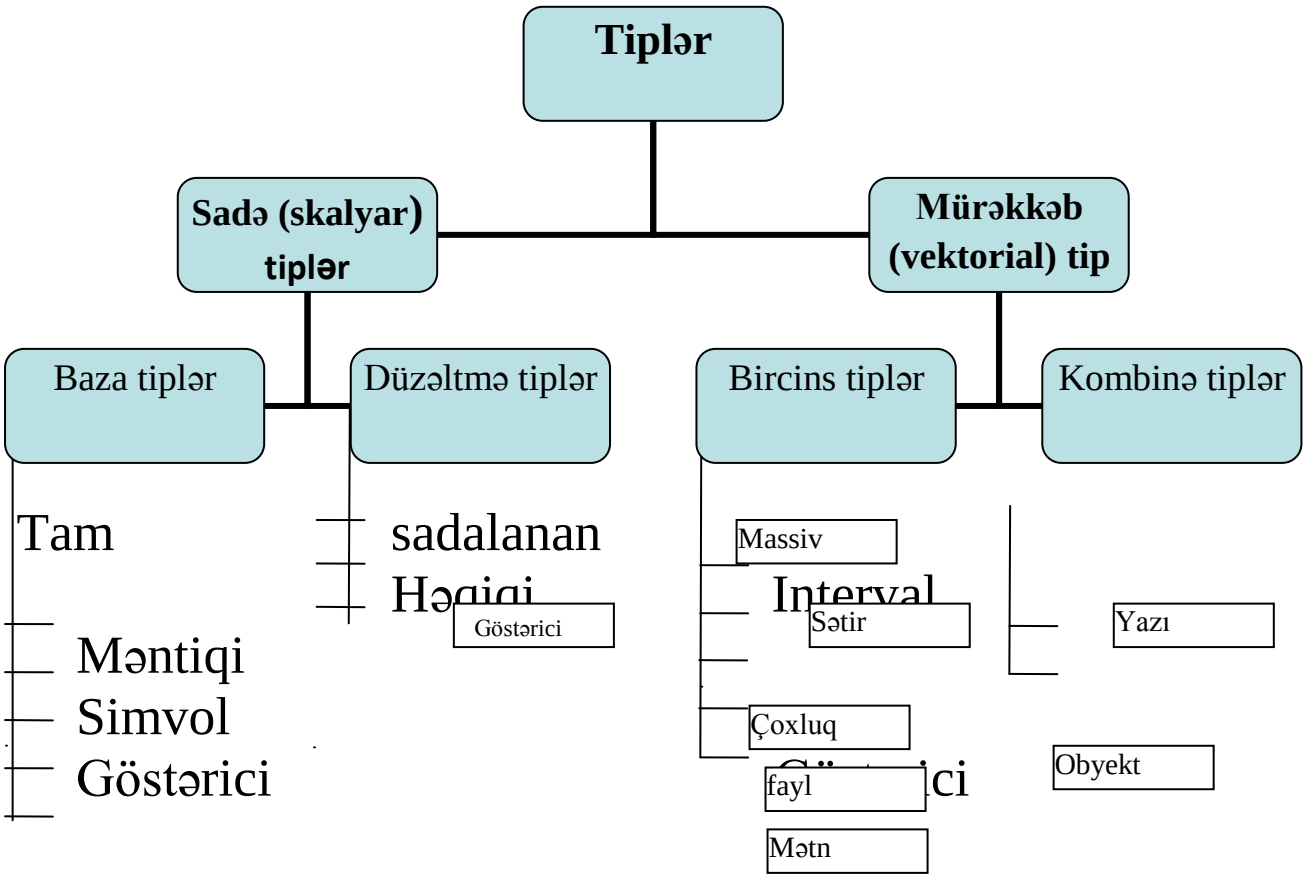
```
program For_1;
    var x0,h,x,y:real;
        n,i:integer;
begin
      write('x0= ');readln(x0);
      write('n= ');readln(n);
      write('h= ');readln(h);
       x:=x0;
       for i:=1 to n do
     begin
       y:=Sin(x)+1/x;
       writeln('x=',x:6:2,'    y=',y:8:3);
         x:=x+h;
     end;
       readln; end.
```

Tiplər

Hər hansı məsələnin həlli üçün proqram tərtib etməmişdən əvvəl proqramda istifadə olunacaq verilənlər və onların strukturu və tipi müəyyənləşdirilməlidir.

Tip dəyişənin ala biləcəyi qiymətlər çoxluğunu, həmin dəyişənin üzərində yerinə yetirilməsi mümkün olan əməliyyatları və onlarla işləmək üçün nəzərdə tutulmuş funksiya və prosedurları müəyyən edir.

Pascal digər proqramlaşdırma dillərindən fərqli olaraq, çox saylı tiplər qruplarına malikdir. Bu tipləri aşağıdakı kimi təsnifatlandırmaq olar.



Şəkil 3.3. Tiplərin təsnifatı

38,39,40,41-i yaz

2. Həqiqi tiplər

Həqiqi tiplərə aiddir: S i n g l e, R e a l, D o u b l e, E x t e n d e d və Comp. Bu tiplər əsasən dəyişmə oblastlarına görə bir birlərindən fərqlənirlər.

Həqiqi tiplər

İdentifikatorlar	**Ədədlərin dəyişmə diapozonu**	**Mantissadakı Rəqəmlərin sayı**
S i n g l e	1. 5e – 45.. 3.4 e38	7..8
R e a l	2. 9e – 39.. 1.7e38	11..12
D o u b l e	5e – 324.. 1.7e308	15..16
E x t e n d e d	3.4e – 4932.. 1.1e4932	19..20
C o m p	1.e – 63.. 1e+63	19..20

Həqiqi tipli verilənlər sabit verdüllü (məsələn, 1.42, -0.045, və s.) və sürüşgən vergülü (məsələn, 1.1e7, 2.108e-10, 3.4e-5 və s.) şəklində təsvir oluna bilər.

Tam tipli ədədlər kimi, həqiqi ədədlərə də hesab əməlləri, Müqayisə əməliyyatları tətbiq oluna bilər. Standart riyazi funksiyalarını hesablamaq olar.

Həqiqi ədədlərin tam və kəsr hissələrini ayırıb digər dəyişənlərə mənimsətmək və onu tam tipliyə çevirmək üçün Pascalda funksiyalar nəzərdə tutulmuşdur:

- I n t (x : R e a l) : R e a l – x ədədinin tam hissəsini ayırır.
- F r a c (x : R e a l) : R e a l – x ədədinin kəsir hissəsini ayırır.
- T u r n (x : R e a l) : I i n t e g e r – x ədədinin tam hissəsini tam tipli ədədə çevirir

- R o u n d (x : R e a l) : I integer – x ədədin yuvarlaqlaşdırıb tam hissəsini tam tipli ədədə çevirir

Nümunə.

```
Var x, y, : real;  m: İnteger;
……..
        x:=6.25;     y:= Int(x);          (Nəticə y=6)
        x:=4.73;      y:= Int(x);          (Nəticə y=4)
        x:=-2.43;      y: =Int(x);          (Nəticə y=-2)

        x:=6.25;     y:= Frak(x);          (Nəticə y=0.25)
        x:=4.73;      y:= Frak(x);          (Nəticə y=0.73)
        x:=-2.43;      y: =Frak(x);          (Nəticə y=-0.43)

        x:=6.25;       y:= Trunc(x);          (Nəticə m=6)
        x:=4.73;      y:= Trunc (x);          (Nəticə m=4)
        x:=-2.43;      y: = Trunc (x);          (Nəticə m=-2)

       x:=6.25;     y:= Round(x);          (Nəticə y=6)
        x:=4.73;      y:= Round (x);          (Nəticə m=5)
        x:=-2.43;      y: = Round (x);          (Nəticə m=-2)
```

3.Məntiqi tiplər

Məntiqi (Bul) tiplər qrupuna daxil olan tiplərin xarakteristikaları aşağıdakı cədvəldə verilmişdir.

İdentifikator	**False-a uyğundur**	**True-uə uyğundur**
Boolean	0-ədədi	o-dan fərqli ixtiyari ədəd
Bytebool	0-ədədi	
Wordbool	Hər iki baytda 0	
Longbool	Bütün baytlarda 0	

Məntiqi əməllər

O p e r a n t l a r		**Ə m ə l l ə r**			
A	**B**	**Not A**	**A and B**	**A or B**	**A xor B**
False	False	True	False	False	False

False	True	True	False	True	True
True	False	False	False	True	True
True	True	False	True	True	False

Mərtəbələr üzrə məntiq və sürüşdürmə əməlləri:

Bu əməllər tam ədədlərin ikilik say sistemindəki təsvirləri, yəni “0” və ”1” kodları ilə işləmək üçün nəzərdə tutulmuşdur.

Not (mərtəbələrlə inkar) əməli ikilik say sistemindəki operandı mərtəbələr üzrə inkar edir.Yəni sıfırı (0) birə,biri sıfıra çevirir.

And (mərtəbələrlə məntiqi VƏ) əməli.Məntiqi **”VƏ”** əməlini operandların uyğun mərtəbələrinə ayrı-ayrılıqda tətbiq edir.

Or (mərtəbələrlə məntiqi VƏ YA) əməli.Məntiqi **” VƏ YA”** əməlini operandların uyğun mərtəbələrinə ayrı-ayrılıqda tətbiq edir.

Xor (mərtəbələrlə məntiqi **xor**) əməli.Məntiqi **”xor”** əməlini operandların uyğun mərtəbələrinə ayrı-ayrılıqda tətbiq edir.

Shl, shr əmələri ”0” və ”1” kodları ilə təsvir olunmuş ədədləri mərtəbə-mərtəbə sola və sağa sürüşdürmək üçün nəzərdə tutulmuş əməllərdir.

Bu əməllərin A=11 və b=2 ədədləri üzərində tətbiqləri nəticəsində alınmış nəticələr aşağıdakı cədvəldə verilmişdir.

	Onluq təsvir	**İkilik təsvir**

A ədədi	11	00001011
B ədədi	2	00000010
Not A	244	11110100
A and B	2	00000010
A or B	11	00001011
A xor B	9	00001001
A shl B	44	00101100
A shr B	2	00000010

Bəzi məntiqi funksiyalar:

Succ (x) funksiyası arqumentin (x-in) növbəti, yəni (x+1) qiymətini alır.

Pred (x) funksiyası arqumentin əvvəlki, yəni x-1 qiymətini alır

Ord ( ) funksiyası False məntiqi sabitinə 0 ədədini, True sabitinə 0-dan fərqli ixtiyari ədəd qarşı qoyur.

a

3.3 Mürəkkəb tiplər

Sadə tipli verilənlər ancaq eyni tipli qimətləri alır.Strukturlanan tipli verilənlər vasitəsi isə çox qiymətli verilənlərin yazılmasında istifadə edilir.Bu tipə Mürəkkəb tip, bəzən aqreqativ tip də deyilir.

1.Çoxluq tip

Proqramlaşdırmada çoxluq anlayışı riyaziyyatdakı çoxluq anlayışı ilə eyni mənada işlədilir.Fərqli cəhətləridə vardır. Pascal proqramlarında çoxluqlar ancaq sıra (tam,məntiqi,simvol,sadalanan) tipli elementlərdən tərtib oluna bilər.Çoxluğun elementləri eyni tipli olmalıdır.Bu tipə çoxluğun baza tipi deyilir.Çoxluğun tərtibi [0,255] intervalından kənara çıxmamalıdır.Ona görədə Tam tiplər qrupuna daxil olan Shortint,İnteger,Longint,Word tipli verilənlər də baza tip ola bilməz.

Çoxluq tipin təsviri

Çoxluq tipinin ümumi yazılışı:

```
Type
        <Identifikator> = Set of <Baza tip>
```

Çoxluğa aid proqram fraqmenti.

```
Type
    Charset= set of Char ;
    Tragam=set of 0..9;
Const
   Digits : charset = [`0`..`9`]
   Letters : Tragam =[0,2,4,6,8];
   Radam   :   Tragam=[0,2.4,6,8];
```

Çoxluqlar üzərində əməllər

Proqramlarda da çoxluqlar üzərində hesab əməlləri riyyaziyyatdakı kimi aparılır.

Birləşmə əməli (+).İki A və B çoxluqlarının birləşməsi (A+B) elə C çoxluğuna deyilir ki, bu çoxluğun hər bir elementi ya A, ya da B çoxluğunun elementi olsun.

Çıxma əməli (-). İki A və B çoxluqlarının fərqi (A-B) elə C çoxluğuna deyilir ki, bu çoxluğun hər bir elementi A çoxluğuna daxil olsun, B çoxluğuna isə yox.

Kəsişmə əməli (*). İki A və B çoxluqlarının kəsişməsi (A*B) elə C çoxluğuna deyilir ki, bu çoxluğun hər bir elementi həm A, həm də B çoxluğunun elementi olsun.

Bərabərlik əməli (=). A və B çoxluqları eyni elementlərdən ibarət olduqda **A=B** əməlinin nəticəsi True, əks halda False olur.

Bərabər deyil əməli (<>). A çoxluqu B çoxluqundan heç olmasa bir elementi ilə fərqləndikdə **A<>B** əməlinin nəticəsi True ,əks halda False olur.

Alt çoxluq əməli (<=). A çoxluğunun bütün elementləri **B** çoxluğuna daxil olduqda **A<=B** əməlinin nəticəsi True ,əks halda False olur.

Üst çoxluq əməli (>=). B çoxluğunun bütün elementləri **A** çoxluğuna daxil olduqda **A>=B** əməlinin nəticəsi True , əks halda False olur.

Aid olma əməli (In).Əgər hər hansı sıra tipli **x** elementi, həmin sıra sıra tipli **A** çoxluğunun elementidirsə, onda **x In A** əməlinin nəticəsi True, əks halda False olur.

Bu əməllərin nəticələri aşağidakı cədvəldə verilmişdir:

İfadə	**Nəticə**
{1,2,3,4}+{3,4,5,6}	{1,2,3,4,5,6}
{1,2,3,4}-{3,4,5,6}	{1,2}
{1,2,3,4}*{3,4,5,6}	{3,4}
{1,2,3,4}={3,4,5,6}	False
{1,2,3,4}<>{3,4,5,6}	True
{1,2,3,4}< ={3,4,5,6}	True
{1,2,3,4}>={3,4,5,6}	False
4 **in** {3,4,5,6}	True

2.Massiv tip

Massiv verilənlərin elə strukturudur ki, bu strukturun elementləri bircinsdir, yəni, eyni tipə məxsusdur, sayı və konfiqurasiyası proqramın icrası zamanı dəyişilmir, elementlər

avtomatik surətdə sistem tərifindən nömrələnir və nömrəyə görə nizamlanır.

Massivlər massivin adı, elementin massivdəki yerini göstərmək üçün tələb olunan koor-dinatlaın sayına massivin ölçüsü, koordinatların özünə isə elementlərin indeksi deyilir.

Massiv **array** xidməti sözü vasitəsilə təsvir olunur:

<Tipin_adı>: array [indekslər] of <Tip>

Burada **<İndekslər>** hər hansı sıra (nizamlı) tipə məxsus indekslərin dəyişmə sərhəd-lərini göstərən interval (diapozon), **<Tip>** isə Pascala məxsus hər hansısa tip ola bilər.

Massivin hər hansısa elementinə müraciət etmək istədikdə massivin adı və düzbucaqlı mötərizənin içərisində bir-birindən vergüllə ayrılan indekslər göstərilir. Məsələn,

```
Const     n=100 ;
Type      T_Vectot=array[1..n,6..12] of real ;
Var        A,B : T_Vectot ;
           C: array [1..5] of integer ;
              Y : real ;
Begin
   Y:=a[2,4]+c[3] ;
```

Yığcam (упаковка olunmuş) massiv.

Pascalda adi massivdən başqa,yığcam (sıxılmış) massiv anlayışı da vardır.Bu məqsədlə massivin **array** xidməti sözünün əvvəlinə **packed** sözünü əlavə etmək lazımdır.Belə massivi yadda saxlamaq üçün tələb olunan yaddaş azalır,massivə müraciət vaxtı ilə artır.Yığcam massivin istifadə qaydası adi massivdəki kimidir.Məsələn,

Var AP: packed array [1..3] of boolean ;

ANP: array [1..3] of boolean ;

Təsvir olunmuş hər iki massiv tutumlarına və yaddaşda saxladıqları məlumatların yaddaşdakı təsvirinə görə bir-birindən fərqlənirlər.

Massiv tip tipləşdirilmişdir konstantlar.

Digər tipli konstantlarda olduğu kimi massiv tip konstantlardada eyni yazıda həm massivin tipini göstərmək,həmdə elementlərin qiymətlərini daxil etmək lazım gəlir.bu zaman massiv bir ölçülüdürsə,bütün massiv elementlərini bir mötərizənin içərisində, iki ölçülüdürsə,hər bir sətirin elementləri ayrıca mötərizə içərisində yazılmalıdır.Yüksək ölçülü massivlərində elementləri analogi qaydada yazılır.

Məsələn,

Bir ölçülü ədədi massiv.

```
Const
    Vector : array[1..7] of real =
```

(0.1,3.25,21.32,55,11.98,78.1,4.5) ;

Iki ölçülü ədədi massiv

1 2 3 4

2 3 4 5

3 4 5 6

```
Const
    Matris : array[1..3,1..4] of real =
        ((1,2,3,4),(2,3,4,5),(3,4,5,6)) ;
```

Bir ölçülü simvollar massiv.

```
Const
    Vector : array[1..6] of char=
        (`P`,`A`,`S`,`C`,`A`,`L`) ;
```

Və ya qısa şəkildə :

```
    Vector : array [1..6] of char= `PASCAL`;
```

Massivin elementlərinin daxil və xaric olunması.

Massivin bütün elementlərini bir prosedur ilə daxil və ya xaric etmək mümkün deyildir. Massivdə elementlər bir-bir yaddaşa yazılıb və oxunmalıdır. Ona görədə Massivi daxil (xaric) etmək üçün çox vaxt dövr təşkil olunur. Bunu aşağıdakı proqramın timsalında nümayiş etdirək.

Çalışma:

10 elementdən ibarət olan massivin ən böyük elementini və bu elementin massivdəki yerini tapmalı.

```
Program max;
    Const
        n=10 ;
    Type
        Tvector=array [l .. n]  of  real  ;
    Var
        Vector : Tvector ;
        max :  real ;
        K,   i :  integer   ;
    Beqin
        Writeln (“Massivin elementlərini daxil edin”) ;
        For  i: = l to n  do Readln ( Vector [i ])  ;
            max : = Vector [i ]   ;   k : =l  ;
        For  i: = l to n  do
            If  max < Vector [i ]   then
              Beqin
            max:= Vector [i ]   ;
                k:= i ;
    end;
    Writeln (“max=” ,  max:6:2) ;
    Writeln (k=”, k:2) ;
    End.
```

Massivləri bir-birilə, yalnız element-element , daxil (xaric) etdiyimiz qaydada müqayisə etmək olar. Lakin bir massivi onunla eyni tipli başqa massivə bir prosedur ilə mənimsətmək olar.

Məsələn,

```
B : array [l ..10 ] of real;
. . .
A:=B;
```

proqram fraqmenti düzdür,

```
Var  A, B : array [l ..10 ] of real;
. . .
   If  (A=B)   . . .
```

Fraqmenti isə səhvdir.

Massiv elementlərinin çeşidlənməsi.

Massiv elementlərinin çeşidlənməsi dedikdə massivin elementlərinin artma və ya azalama istiqamətində nizamlanması nəzərdə tutulur. Massiv elementlərini adətən iki üsulla:

- Bir başa seçimlə yerdəyişmə
- Qonşu elementlərinin yerlərinin dəyişdirilməsi adlanan metodlar ilə yerinə yetirirlər.

Artma istiqamətində ***bir başa seşim*** aşağıdakı alqoritmlə yerinə yetirilir.

1. Massivin birinci elementindən başlamaqla minimum elementi

tapılır. Sonra massivdə 1-ci yerdəki elementlə, minimum yerləri dəyişdirilir.

2. Alqoritmin 1-ci bəndi 2-ci elementdən başlamaqla təkrarlanır.
3. Bu qayda sonuncu elementə qədər təkrarlanır.

Proqram nümunəsi.

```
Program Sort__ Massiv__ Birbaşa;
const
            size=5;
     var
       a:   array [l . . size] of integer;
      i,  min, J, buf, k: integer;
begin
      for k:=1 to size do read (a[k ]) ;
     Writeln (‘İlkin massiv’) ;
     For k:=1 to size do Write (a[k],’ ’) ;
    Writeln;
    Writeln (‘Cesidlama’);
    For i:=1 to size do
  begin
        min:=i;
         for J:=i+1 to size do
        if a[j] <a[min] then min:=j;
        buf: =a[i];
        a[i]:=a[min];
         a[min]:=buf ;
```

```
    for k:=1 to size do Write (a [k], ‘ ’) ;
      Writeln;
   end;
   readln (i);
end.
```

Nəticə:

İlkin massiv

12 -3 56 47 10

Cesidlama

-3 12 56 47 10

-3 10 56 47 12

-3 10 12 47 56

-3 10 12 47 56

Artma istiqamətində ceşidləmə qonşu elementlərin yerlərinin dəyişdirilməsi alqoritmi ilə aşağıdakı kimi yerinə yetirilir.

Birinci elementdən başlamaqla qonşu elementlər müqayisə olunur. Birinci qonşu element ikincidən böyükdürsə yerləri dəyişdirilir.

Bu proses massivin elementlərinin sayından bir dəfə az sayda təkrarlanır.

Proqram nümunəsi.

```
Program Sort Massiv Qonsu;
```

```
const
            size=5;
           var
             a: array[1. .size] of integer;
            i,j, buf, k: integer;
 begin
         for  k:=1 to size do read (a[k]) ;
Writeln (` İlkin massiv`) ;
        for  k:=1 to size do Write (a[k ],`  `) ;
       Writeln;
Writeln (` Cesidlama`) ;
for i:=1 to size-1 do begin
  for k:=1 to size-1 do begin
    if a [k] > a[k+1] then begin;
      buf:=a[k];
      a [k]: = a[k+1];
        a [k+1 ]:=buf;
 end;
end;
  for k:=1 to size do Write (a[k], `  `) ;
       Writeln;
     end;
    readln (i) ;
end.
İlkin massiv
3 2 4 5 1
Cesidlama
```

2 3 4 1 5
2 3 1 4 5
2 1 3 4 5
1 2 3 4 5

Çalışma

İki ölçülü massivin daxil olunması, çapı və sütun elementlərinin maxsimumunun tapılmasına aid proqram.

```
program massiv__1 ;
var
    a: array[1..3,1..4] of real;
    max: array [1..4] of  real;
    i , j: integer;
begin
    { Massivin elementlərinin daxil olması}
    for  i:=1 to 3 do
    for  j:=1 to 4 do
    readln (a[i,j]) ;
    { Daxil olunmuş elementlərin iki ölçülü massiv kimi çapı}
    for  i:=1 to 3 do
    begin
    for  j:=1 to 4 do
        Write (a[i,j]:5 : 1, ` `) ;
        Writeln;
    end;
                    {Maksimumun tapılması}
    for j:=1  to 4 do max [j]:=a[i,j];
```

```
for j:=1  to 4 do
for i:=1  to 3 do
if  a[i,j]>max [j] then max[j]:=a[i,j];

                          {Maximumun capi}
for j:=1  to 25 do Write (`-`);
Writeln;
  for j:=1 to 4 do
Write (max[j]:5:1, `  `) ;
Readln;
end.
```

Nəticə:

```
7.6     8.9     4.2     -6.0
8.1     7.4     5.6      9.2
 -5.0   12.6    8.1      5.8
-----------------------------
 8.1   12.6     8.1     9.2
```

Çalışma. [0,100] intervalında dəyişən təsadüfi kəmiyyətlərdən 3 sətirdən 6 sütundan ibarət iki ölçülü massiv tərtib etməli. Bu massivin [25,75] intervalındakı elementlərinin sayını hər bir sətir üçün tapmalı.

```
program massiv__2 ;
var
    a: array [1..3,1..6] of integer;
```

```
    s: array [1..6] of  integer;
    i, j  : integer;
begin
    { Massivin təsadüfi kəmiyyətlərdən tərtibi}
       Randomize;
    for i:=1 to 3 do
    for j:= 1 to 6 do
    a [i,j]:=random (101) ;
    {[25 . . 75 ] interbalına daxil olan sətir elementlərinin
sayının hesablanması }
          for i:=1 to 3 do  s [i]:=0
          for i :=1 to 3 do
          for j :=1 to 6 do
          if (25<=a [i,j]) and (a[i,j]<=75)
                                  then  s[i]:=s[i]=1 ;
    {Nəticənin capı }
Writeln (`                    Massiv
Say `) ;
             for  i:=1 to 3  do
       begin    for  j:=1 to 6  do
                Write (a[i,j] : 3, ` `)  ;
                Writeln (` `    , s [i ] ) ;
       end ;
    readln (i) ;
end.
```

Nəticə

Massiv Say

85	2	87	95	36	91	1
27	12	43	0	69	50	4
48	17	74	98	3	28	3

3. Sətir tip

Pascalda mətn sətirləri ilə işləmək üçün Sətir tip nəzərdə tutulmuşdur. Sətir simvollardan ibarət bir ölçülü xüsusi növlü massivdir.

Sətirlər təsvir və uzunluqlarının göstərilmə üsullarına görə iki tip sətirə

- *String* - sətirinə və
- *Turbo* - sətirə

ayrılırlar.

Turbo-sətirə sıfır sonluqlu sətir, bəzən simvollar massivi də deyilir.

String tipli dəyişən aşağıdakı üsullarla təsvir edilə bilər:

Var <Sətir _ dəyişənin_ adı > : *String ;*

və ya

Var <Sətir _ dəyişənin_ adı > : *String [n] ;*

Məsələn,

```
Var
        S1: String [10 ] ;
        S2: String [128 ] ;
        S3: String ;
```

Əgər sətir elan olunarkən sətirdəki simvolların sayı göstərilməyibsə, bu sətirdəki simvolların maksimal sayı sistem tərəfindən 225 qəbul olunur.

Turbo – sətir tipli dəyişən

Var

<Sətir _ dəyişənin_ adı > : array *[0 . . n] of Char*

;

şəklində təsvir olunur.

Sətir tipli verilənlər iki apostrov (′′) işarəsi altında simvollar ardıcıllığı şəklində göstərilir.

Məsələn, *S2 : =′ Turbo Pascal ′* .

Bildiyimiz kimi, massivlər element-element yaddaşa daxil (xaric) edilir. Sətirlərdə isə bu ′əməliyyatları bir prosedur ilə də yerinə yetirmək mümkündür.

Sətir tipli verilənləri emal etmək üçün Pascalda bir sıra funksiya və prosedurlar nəzərdə tutulmuşdur.

■ *Copy (st : String, **İndex, Count** : integer) : string ;*

- *st* sətirindən *İndex* mövqeyindən başlamaqla *Count* sayda simvolu köçürdür.

Nümunə:

Var *st : String ; Mt1 , Mt2 : string [10] ;*

. . .

st := `Azərbaycan Dövlət Neft Akademiyası `

mt1 : = ***Copy*** *(st, 12, 6) ;* {Nəticə `Dövlət`}

mt2 : = ***Copy*** *(st, 24,9) ;* {Nəticə `Akademiya`}

■ ***Pos*** *(fraqment, st : String) : byte ;* - *st* – sətirinə fraqmentin daxil olub- olmamasını müəyyənləşdirir. Fraqment daxildirsə funksiyanın qiyməti fraqmentin *st* – sətirindəki başlanğıc mövqeyinə bərabər olacaqdır. Fraqment st- sətirində

yoxdursa funksiya sıfır qiymətini alacaqdır. Fraqment bir neçə dəfə sətirə daxildirsə, Funksiya ilk fraqmentin başlanğıc mövqeyini göstərəcəkdir.

Nümunə:

Var *st : String ; Mt1 , MT2 : string [10] ;*
k1, k2 : byte ;
. . .
st : = ` Azərbaycan Dövlət Neft Akademiyası `
Mt1 : = `bay` ; t1 : = pos (Mt1, st) ; {Nəticə 5}
Mt1 : = `Bay` ; t1 : =pos (Mt1, st) ; {Nəticə 0}
t1 : =pos (`gul`, st) ; {Nəticə 0}

■ ***Lenght*** *(s : String) : byte ;* Sətirin cari uzunluğunu təyin edir.

Məsələn,
Lenght (`Məktəb`) ; {Nəticə 6}
Lenght (`İsayeva G.A . `) ; {Nəticə 13}

■ **Concat** *(s_1, s_2, ...s_n : String) : String ;* Bir neçə sətiri birləşdirir. Məsələn,

Var *st : Strinq ; Mt1, Mt2 : string [10] ;*
. . .
Mt1 : = ` Neft ` *;*
Mt2: = ` Akademiyası` *;*
St := Concat (Mt1, Mt2) ; {Nəticə *st : =* ` Neft Akademiyası`}

■ **UpCase** *(s: String) : string ;* Sətirdəki kiçik hərifləri bir – bur
böyük hərflərə çevirir. *Delphidə* isə bir əmrlə *AnsiUpperCase (s : String) : string ;* proseduru vasitəsilə ***s*** sətrini böyük hərflər sətirinə çevirmək mümkündür.

Məsələn,

```
program St_1;
{ŞAPPTYPE CONSOLE}
uses
   SysUtils ;
var
  s , st : string ;
      i : integer ;
begin
    st: = “ İsayeva Gülarə “ ;
    s : = “ “ ;
   for  i : = 1  to Length ( st )  do
   s : = s + UpCase ( st [i ])  ;

      Writeln ( s ) ;
      Writeln ( s ) ;
      s : = AnsiUpperCase ( st) ;
      Writeln ( s ) ;
      readln ;
      { TODO – oUser  - cConsole Main   :  İnsert code  here}
```

end.

Bu proqram *“İsayeva Gülarə”* sətrini *“ İSAYEVA GÜLARƏ “* Sətrinə çevirəcəkdir. (Proqram Delphinin Konsul rejimində yazılıb).

■ **Delete** *(st : String , İndex , Count : integer) ; st* sətirdən, *İndex*

mövqeyindən başlamaqla *Count* sayda simvol silir.

■ **İnsert** *(fraqment, st: String, İndex : integer) ; st* sətirinə *İndex*

mövqeydən başlamaqla fraqment əlavə edir.

proqram st 1 ;

Nümunə: S sətrindəki *“ Neft”* sözünü *“ Musiqi “* sözü ilə əvəz etməli.

```
var.
        s, s1 : string ;
          k: integer ;
begin
        s : = `Azərbaycan Dövlət Neft Akademiyası `;
        Writeln ( s ) ;
        k : = pos ( ` Neft ` , s );
        delete ( s, k, 5) ;
        insert (`  Musiqi ` ,  s,  k )  ;
        Writeln ( s ) ;
        readln ;
end.
```

■ ***StrToFloat*** *(st : String) : Real ;* proseduru ***st –*** simvol sətrini
həqiqi tipli ədədə çevirir.

■ **FloatToStr** *(st : Real) : Strinq ;* proseduru ***st-*** həqiqi tipli ədədi
simvol sətrinə şevirir.

■ **StrTolnt** (*st : String): İnteger ;* proseduru ***st –*** simvol sətrini
tam tipli ədədə çevirir.

■ **İntToStr** *(st: İnteger) : String ;* proseduru ***st-*** tam tipli ədədi
simvol sətrinə çevirir.

Çalışma_1. Verilmiş tam ədədin

- rəqəmləri cəmini,
- tək mərtəbədəki rəqəmləri cəmini,
- cüt mərtəbədəki rəqəmləri cəmini

tapmalı.

```
program srr_val;
var  s : string  [10 ]  ;
     x, kod, n, i, Sum,sl, s2: integer;
begin
          Write(` x=`) ;  readln (x) ;
          Str (x, s) ;
          Writeln ( `s=` , s) ;
          n: = length (s) ;
          Sum:= 0 ;  s1 :=0 ; s2:= 1 ;
          For  i:= 1  to  n   do
```

```
    Begin
            Val (s [i ], x,kod)  ;
    Sum:= Sum +x ;
    If ( i mod 2) = 0  then  s2:=s2*x
                        else s1:=S1+x ;
  end;
    Writeln (`sum=` ,Sum) ;
    Writeln ( `taklarin cəmi =` , s1) ;
    WriteLn ( `cutlarin hasili=` , s2) ;
   readln ;
end.
```

Çalışma_2. 10-luq say sistemində verilmiş ədədi 2-lik, 8-lik və 16-lıq say sistemlərindən birinə çevirməli.

```
Program Say_Sistem;
var
        s1:string [1 ]  ;
        s, s2: string [10 ]  ;
        ss, p, q, n, k, i: integer ;
begin
        Write(`Adad:`)  ; readln (k) ;
        Write(`Say_ sistem :`) ; readln (ss) ;
            i:=0 ;  s:= ``  ;
        While  k>0   do
        begin
        p: =k  div  ss ;
        q:=k   mod ss ;
                  case q of
                                          10:   s1 := "A"  ;
                                          11:   s1 := "B" ;
                                          12:   s1 := "C" ;
```

```
                                    13:   s1 := “D”;
                                    14:   s1 := “E”;
                                    15:   s1 := “F”;
                        else str (q, s1) ;
                        end ;
                    s:= s+s1 ;
                k:= p ;
        end;
            n := length (s) ;
           Writeln ( `n=` , n) ;
                s2 := `   ` ;
           for  i:=1  to   n  do
           s2:=s2+s [n+1-i ]  ;
            Writeln ( `s_s_` , ss,  `=`, s2) ;
                readln ;
end.
```

Çalışma_3. 2-lik, 8-lik və ya 16-lıq say sistemlərinin birində verilmiş ədədi 10-luq say sistemində təsvir etməli.

```
program say_s_10
var
      s: strinq [10 ]  ;
     m , n , ss , i , kod : integer ;
     a : real ;
begin
        Write (` Say_ sistemi = `) ; readln (ss) ;
         Write (`Adad =: `) ; readln (s) ;
                n:=  Length (s) ;
            a:=0
```

```
                for  i:=1  to  n  do
        begin
            case  s [i]   of
                    “A” :    m: = 10 ;
                    “B”:     m: = 11;
                    “C” :    m:= 12 ;
                    “D”:     m: = 13;
                    “E” :    m: = 14 ;
                    “F”      m: = 15;
                 else      val (s  [i ], m, kod)  ;
            end;
                    a:=a+m*Exp ((n-i) * 1n (ss) ) ;
        end.
        Writeln ( `10_luq_ədəd=` ,  a:8:0) ;
           readln;
    end.
```

Çalışma_4. Verilmiş cümlədə, klaviaturadan daxil edilmiş simvolun sayını və cümlənin uzunluğuna nəzərən faizini tapmalı.

```
program Str1;
var   s,t : strinq ;
        t1: string [1]  ;
        i , j , l , m, n, k: integer;
       fz :real ;
begin
     s:= `xalqımızın ana kitabı “dede qorqud”dastanıdır.` ;
```

```
        t:=` ` ; m:= length (s);
        Writeln ( `m =` , m) ;
        Write ( `Simvol: `);  readln (tl) ;
          n:= length (t) ;   1 :=0;
      k:=0;
  for   j:=1   to   m  do
      if   copy (s, j, 1) =t1    then   k: =k+1;
  fz :=k*100/m;
       Write (`   `, t1, `_`, k , `_`, fz : 4 : 1 , "%" ) ;
       read (1) ;
      end.
```

Çalışma_5. Verilmiş cümlədə hər bir hərfin sayını və ümumi hərflərə nəzərən faizini tapmalı.

```
program Str;
const  d="   ,  . !"? : ; "   ;
   var   s, t : string  ;
         t1, t2 : string [1]  ;
        i, j, l, m, n, k : integer  ;
        fz : real  ;
   begin
   s:= `xalqımızın ana kitabı "dede qorqud"dastanıdır.` ;
          t:= ``  ;     m:= lehgth (s)   ;
         for  i:=1   to m  do  begin
        t1:= copy (s, i, 1) ;    {t1 := s[i] ;}
       if (pos (t1,  t)=0)   and (pos (t1, d )=0)
       then   t:= concat (t, t1)     {t := t+t1 ;}
```

```
end ;
 Writeln ( `s_sətri :  ` )  ;
 Writeln ( `   ` , s) ;
 Write  ( `s_sətrindəki_ hərifl ər :   ) ;
 Writeln ( “     ” , t ) ;
 n:= lenght (t)   ;     1:=0   ;
Writeln ( “s_sətrinin uzunluğu = ” ,m ) ;
Writeln ( “müxtəlif hərifl ərin sayı = ” ,n ) ;
Writeln  ;
Writeln ( “hər bir hərfin s_ sətrindəki sayı və %-i ” ) ;
for  i:=1   to  n  do  begin     k:=0  ;
for  j:=1   to m  do
   if {copy (s , j , 1) = copy (t , i , 1 ) }
         s [j] =  t [i]          then  k:= k+1;
fz := k*100/ m ;
Write  ( “   “ ,  copy (t , i , 1) , “__”,  k , “_” , fz  : 4 : 1 , “ %
“) ;
 1:=1+1  ;
    If   ( 1 mod   4 ) = 0  then Writeln ;
   end ;
```

Nəticə :

s_sətri :

xalqımızın ana kitabı “dede qorqud”dastanıdır.

s_sətrindəki hərifl ər : xalqımznktbdeorus

s_sətrinin uzunluğu = 47

müxtəlif hərifl ərin sayı = 17

hər bir hərfin s- sətrindəki sayı və %-i

x- 1 - 2.1% a - 6- 12.8% 1 -1- 2.1% q- 3- 6.4 %

i- 7 -14.9% m- 1- 2.1% z -1- 2.1% n-3- 6.4%

k-1- 2.1% t- 2- 4.3% b-1- 2.1% d-5- 10.6%

e-2- 4.3% 0-1- 2.1% r-2- 4.3% u-1- 2.1%

s-1- 2.1%

4. Yazı tip

Massivlərdən, çoxluqlardan, fayllardan fərqli olaraq yazılar müxtəlif tipli verilənlərin məcmusudur. Bu tiplər siyahısını "yazı" tipi şəklində tərtib etmək istədikdə siyahının əvvəlinə *"record"* sonuna isə *"end"* sözləri əlavə olunur. "Yazı" ümumi şəkildə aşağıdakı kimi təsvir oluna bilər:

<Yazının__ adı > = **record**

<Sahələrinin__ adları və tipi> ;

end;

Pascalda "yazı"lar adi (fiksasiyalı) və variantlı yazılara ayrılır.

- **Adi (fiksasiyalı) yazılar.**

Adi yazılar bir və ya bir neçə dəyişəndən (sahədən) ibarət olur. Bu dəyişənlər yazının təşkiledičiləri kimi elan olunduqda onların adı (identifikatoru) və tipi göstərilir. Nümunə kimi

tələbənin şəxsi kartoçkasını əks etdirən “yazı” aşağıda verilmişdir.

```
Type
    String7= String [7]
    String20= String [20]
    TstudentCard = record
        SurName     :  String20   ;  {Familiya}
        Name        :  String10   ;  {Ad}
        FatherName  :  String10   ;  {Atasının adı}
        Year        :  İnteger    ;  {Anadan olduğu tarix}
        HomeAddress :  String20   ;  {Ev ünvanı}
        GrupCode    :  String7    ;  {Qrup nömrəsi}
        MathAnal    :  Byte       ;  {Riyazi analiz}
        LinAl       :  Byte       ;  {Xətti cəbr}
        Informatika :  Byte       ;  { İnformatika }
        Phys        :  Byte       ;  { Fizika }
    End;
```

Göründüyü kimi bu “yazıda” eyni məna daşıyan sahələr {MathAnal , LinaL, İnformatika, Phys } vardır. Proqramı daha oxunaqlı etmək, əyyaniliyi artırmaq və qrup şəklində əməliyyatların yerinə yetirilməsini sadələşdirmək məqsədilə bu sahələri ayrıca “yazı” strukturlu verilən kimi tərtib etmək olar. Belə halda yuxarıdakı proqram fraqmenti aşağıdakı şəkildə yazıla bilər:

```
Type
```

```
String7 =  String [7] ;
String20 =  String [20] ;
     Tmarks    :  record
          MathAnal      :  Byte      ;
          LinAl         :  Byte      ;
          Informatika   :  Byte      ;
          Phys          :  Byte      ;
     End;
TstudentCard = record
      SurName      :  String20     ;
     Name          :  String20     ;
     FatherName    :  String20     ;
     Year          :  İnteger      ;
     HomeAddress   :  String       ;
     GrupCode      :  String7      ;
     Marks         :  Tmarks       ;
 End;
```

■ **Variantlı yazılar**

Bəzən proqramlarda yalnız bəzi sahələrinə görə bir-birindən fərqlənən bir neçə "yazı" dan istifadə etmək lazım gəlir. Belə hallarda proqramı təşgil edən operatorların sayını azaltmaq, yaddaşa qənaət etmək, proqramın tərtıb olunmasını sadələşdirmək, nəticədə proqramı daha oxunaqlı etmək məqsədilə bir neçə adi (fiksatsiyalı) " yazı" əvəzinə bir "variantlı" yazıdan istifadə etmək olar.

Variantlı "yazı" iki hissədən ibarət olur.

1) Birinci hissə adi fiksatsiya olunmuş “yazı” dan;
2) İkinci hissə seçmə əlamətindən asılı olaraq variant operatoru (*Case)* vasitəsilə seçilən variantlar siyahıısından;

Yuxarıda baxılan proqram fraqmentləri bir smestrdəki imtahan qiymətləri əsasında tələbələrin şəxsi kartoçkasını tərtib etmək üçün yazılmışdı.

Tutaq ki, ikinci smestr üçün də belə bir kartoçka tərtib etmək lazımdır.

Bu iki kartoçka aydındır ki, bir-birindən imtahanların adlarına və imtahan qiymətlərinə görə fərqlənəcəklər. Tələbələrin ad-familiyaları, ev ünvanları, qrup nömrələri isə dəyişməz qalacaqdır.

Hər iki kartoçkadakı məlumatları Variantlı “yazıdan” istifadə etməklə bir “yazı” şəklində tərtib etmək olar.

```
Type
    String7 =  String [7] ;
    String20 =  String [20] ;
      TmarksSem1      :   record
          MathAnal      :  Byte       ;
          LinAl         :  Byte       ;
          Informatika   :  Byte       ;
          Phys          :  Byte       ;
       End;

     TmarksSem2       :   record
```

```
    MathAnal 12      :  Byte       ;
    Elektron         :  Byte       ;
      Informatika2   :   Byte      ;
      DioqAutom      :   Byte      ;
End;
TstudentCard  = record
        SurName     :  String20    ;
       Name         :  String20    ;
       FatherName   :  String20    ;
      Year          :  İnteger     ;
      HomeAddress   :  String      ;
      GrupCode      :   String7    ;
Case  Semestr     :  Byte  of
1: (MarksSem1 :    TmarksSem1)  ; {1-ci Sm.qiym.}
2: ( MarksSem2 :    TmarksSem2)  ; {2-ci Sm.qiym.}
end;
end;
```

Qeyd edək ki, variantlı yazını təşkil edən fiksatsiyalı (yəni dəyişməz) hissə əvvəl, variantlar siyahısı isə sonra yazılmalıdır. Başqa sözlə, Variantlı “yazı” ya daxil olan *Case* operatorundan sonra heç bir “yazı” sahəsi yazmaq olmaz.

■ **“Yazı” sahələrinə müraciət qaydaları.**

Massivlərdən fərqli olaraq, yazıların komponentlərinə (sahələrinə) müraciət, indekslərə görə deyil, adlarına-identifikatorlara görə edilir. Kvalifikasiya idetifikatorları adlanan bu identifikatorlar, “yazı” tipli dəyişəndən başlamaqla

tələb olunan sahənin identifikatoruna qədər bütün marşurutu özündə əks etdirməlidir.

Nümunə üçün tutaq ki, aşağıdakı tip və dəyişənlər verilmişdir:

```
Type
    Tgroup= array [1 . . 20]   of TstudentCard  ;
Var
    Group_KB51,    Group_KB52  :    Tgroup  ;
```

Bu halda aşağıdakı proqram fraqmentində verilmiş operatorlar və "yazı" nın sahələrinə müraciətlər korrekt olacaqdır.

```
    Group_KB51  [1] . Name   : =  "Gasanov"   ;
    Group_KB51  [1] . Year    : =  1965 ;
    Group_KB51  [1] .  Marks . İnformatika :5  ;
    Group_KB51  [1] .  Marks=   ; Group_KB52  [1] .
Marks;
    Group_KB51  [5] := Group_KB52  [5]   ;
    Group_KB51  [1] := Group_KB52 ;
```

■ **With birləşdirici operatoru vasitəsilə müraciət.**

"Yazı"lar ilə işi sadələşdirmək və proqrama əyyanilik vermək məqsədilə Pascalda xüsusi *With* birləşdirici operatoru mövcuddur. Qoşma operator da adlanan bu operatorun ümumi yazılışı aşağıdakı kimidir:

With < yazı_ Dəyişənin_ adı > do < Operatorla r>

Bu operatordan istifadə etməklə yuxarıdakı proqram fraqmentində verilmiş operatorlar aşağıdakı şəkildə yazıla bilər.

```
        With Group_KB51  [1]  do
Begin
    Name:= " Gasanov"   ;
    Year := 1965   ;
    Marks. İnformatika : =5 ;
    Marks:= Group KB52[1]. Marks ;
End;
    Group_KB51  [5] := Group_KB52 [5]  ;
    Group_KB51  := Group_KB52  ;
```

"Yazı" tipli dəyişənin qiymətlərini daxil (xaric) etdikdə element-element daxil etmə konsepsiyasından istifadə etmək lazımdır. Buna görədə "Yazı"nı təşkil edən hər bir sahənin qiyməti ayrılıqda, hətta bəzi hallarda massivlərdə olduğu kimi dövr təşgil etməklə daxil (xaric) olunmalıdır.

■ **"Yazı" tip tipləşdirilmiş konstantlar.** "Yazı" tip konstantlar nümunə kimi aşağıdakı proqram fraqmentində verilmişdir.

```
type
    Rec= record
            R: real  ;
            B:boolean;
```

```
            C:char;
    End;
     ArrayOfRec = array[1 . . 3  ] of  Rec;
    RecOfRec = record
                    I:  integer;
                    S: string ;
                    Z:Rec;
        End;
Const
    RecElem:Rec= (R:3. 1415, B:True;C : “*”) ;
ArrayRec: ArrayOfRec=
                        ((R:3.1415,B:True; C:”*”) ,
                        (R:0.0,B:False; C:”Ş”),
                        (R:6.2832,B:True; C: “&”)) ;
    RecRec:  RecOfRec =
                        (I: 32767; S: “Pascal” ;
                        Z:(R:3. 1415; B:True; C:” *”)) ;
```

Proqram nümunəsi:

3 sayda yazı tipli elementdən ibarət bir ölçülü Massiv verilmişdir. Hər bir yazı aşağıdakı sahələrdən təşkil olunub:

- Tələbənin familiyası
- Tələbənin adı
- Riyazi analizdən imtahan qiyməti
- Cəbrdən imtahan qiyməti
- Fizikadan imtahan qiyməti
- İnformatikadan imtahan qiyməti

1. Tələbələrin siyahısını (Ad – familiyalarını) tiplp sabit kimi daxil etməli.
2. İmtahanların qiymətlərini klaviaturadan daxil etməli və siyahını çap etməli.
3. Əlaçı tələbələrin siyahısını çap etməli.
4. İnformatika fənnindən 4,5 qiymət almış tələbələrin siyahısını
   çap etməli

```
program Yazı_1 ;
const m=3 ;
 type
    Ttalaba=record
        Fam  :  string [12] ;
        Ad    :   string [12] ;
       Riy_Analiz   : Byte ;
       Cəbr          : Byte ;
      Fizika        : Byte ;
      İnformatika : Byte ;
  end;
var
  Tal_M: array [1.   . m]  of Ttalaba=
     ((Fam: “Gasanov” ; Ad: “Gulxan”) ,
     (Fam: “Nuriyeva” ; Ad : “Elza”) ,
       (Fam:”Tağıyev” ; Ad : “Firuz” )) ;
       I: integer ;
       B: Boolean;
Begin
       For i:=1 to m do
```

```
Begin
        With  Tal_M [i] do begin
Writeln (i, “.  “ , Fam , “  “ , Ad ) ;
Write (“ Riy_Analiz- “) ; readln (Riy_Analiz) ;
Write (“ Cəbr- “) ; readln (Cəbr) ;
Write (“Fizika- “) ; readln (Fizika) ;
Write (“ İnformatika- “) ; readln (İnformatika) ;
Writeln;
      end;
end;
          Writeln (“T’ələbələrin siyahısı : “) ;
            For i:=1  to  m  do
With Tal_M [i] do begin
Writeln (Fam, “  “, Ad , “   “  , Riy Analiz ,
    “   “, Cəbr, “  “ , Fizika, “  “ , İnformatika ) ;
      End ;
       Writeln ;
      Writeln ( “ Əlaçı tələbələr: );
        For i:=1  to  m  do
With Tal_M [i] do begin
B:=(Riy_Analiz=5 ) And(Cəbr=5) And (Fizika=5)
          And (İnformatika=5);
          If  B=True  then Writeln ( Fam, “   “ , Ad) ;
      end;
       Writeln;
Writeln( “ İnformatikadan 4,5- qiymətlərlə oxuyan
tələbələr : “) ;
```

```
            For  i:=1  to  m do
              With Tal_M [i] do begin
if İNformatika>=4 then  Writeln (Fam, “  “ , Ad ) ;
            end ; readln ;
end.
```

4. Alt proqramlar və modullar

4.1. Prosedurlar və funksiyalar

1. Prosedurlar və funksiyaların strukturu

Proqramların tərtib olunmasını asanlaşdırmaq məqsədilə bütün alqoritmik dillərdə olduğu kimi, Pascalda da bir sıra prosedur və funksiyalar nəzərdə tutulmuşdur. Belə prosedur və funksiyalar standart prosedur və funksiyalar adlanır.

Prosedur və funksiyaları proqramçıların özləri də yaza bilər. Proqramçıların özlərinin yazdıqları prosedur və funksiyalara istifadəçi prosedur və funksiyaları, bəzən isə alt proqramlar da deyilir.

Prosedur və funksiyalann strukturları aşağıdakı şəkildə olub, demək olarkı, proqramların strukturlarından yalnız başlıqlarına görə fərqlənirlər.

Prosedurun strukturu:

Prosedure **Pros_adı** (formal parametrlər);
Label
Const { lokal verilənlərin təsviri }
Type
Var
Prosedure {daxili prosedurlar}
Function {daxili funksiyalar}
begin
Operatorlar

end.

Funksiyanın strukturu:

Function **func_adı** (formal parametrlər): **nəticənin tipi;**
Label
Cons t {lokal verilənlərin təsviri }
Type
Var
Procedure { daxili prosedurlar}
Function {daxili funksiyalar}
begin
Operatorlar;
func_ adi: = Nəticə ;
end.

Göründüyü kimi funksiya ilə prosedur bir-birindən iki cəhətə görə fərqlənir.

1)Adlarına görə.
2)Operatorlar siyahısına görə.

Prosedurun adından fərqli olaraq funksiyanın adı proqramlarda həmdə dəyişən rolunu oynayır və bu dəyişənin adı başlıqda göstərilir. Funksiya vasitəsilə alınmış nəticə də operatorlar siyahısında müvafiq mənimsəmə operatoru vasitəsilə bu dəyişənə - funksiyanı adına mənimsədilir. Ona görə funksiyanın adı riyazi funksiyalarda olduğu kimi, funksiya vasitəsilə hesablanılan nəticəni də göstərir.

1. Prosedur və funksiyalar ilə proqramlar arasında verilənlər mübadiləsi

Program ilə prosedur və funksiyalar arasında verilənlər mübadiləsi iki üsulla reallaşdırılır.

1) Qlobal identifikatorlar vasitəsilə
2) Prosedur (funksiyanın) başlıqlarındakı parametrlər vasitəsilə.

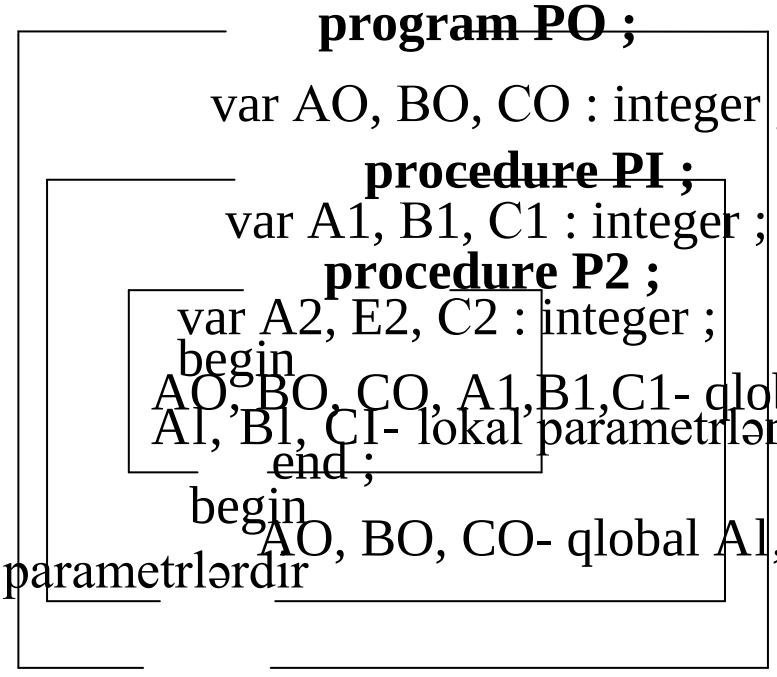
1) Qlobal və lokal identifikatorlar

Prosedur və funksiyalarda istifadə olunan identifikatorlar təsir sferasına görə lokal və qlobal adlı iki qrupa ayrılırlar.

Lokal identifikatorlar elə identifikatorlara deyilir ki, onların təsir dairəsi daxil olduqları prosedur/ funksiyadan kənara çıxmır. Qlobal identifikatorlar isə elə identifikatorlara deyilir ki, onların məzmunu həm prosedur/ funksiya daxilində həmdə kənarda saxlanılır.

Qlobal və lokal identifikatorların yaradılma mexanizmi, həmdə təsir dairəsi aşağıdakı sxemdə nümayiş etdirilmişdir.

Sxemdən göründüyü kimi, sxemdəki struktura malik programda A0, B0, C0 identifikatorları proqrama daxil olan bütün prosedurlarda qlobal, A1,B1,C1 identifikatorları P1-ə daxil olan prosedurda (verilmiş sxemdə P2 proseduru) qlobal, A2, B2, C2 isə yalnız elan olunduqları P2 prosedurunda lokal identifikatorlar olacaqlar.



```
        end ;
    Ancaq AO, BO, CO -dan istifadə etmək olar
        end ;
```

Qeyd edək ki, identifikatorların "lokal" və ya "qlobal " identifikator olması da nisbi anlayışdır. Belə ki, bir prosedur daxilində qlobal olan identifikator, başqa bir prosedurda lokal ola bilər.

Lokal parametr ilə qlobal identifikator üst-üstə düşə bilər. Belə halda ancaq lokal identifikatorların təsiri olacaqdır. Bu qayda aşağıdakı proqram fraqmentində şərh (komentariya) şəklində verilmişdir.

```
Program RQ ;
Var A, B, C : integer;
    Procedure P1 ;
Var A, B, C : char;
begin
    {Bu bölmədə A, B, C -char tipli olacaq}
    end ;

Begin
    {Bu bölmədə A, B, C-integer tipli olacaq}
end.
```

3. Parametrlərin mübadiləsi üsullarının təsnifatı

Proqramlar ilə prosedur və funksiyalar arasında verilənlər mübadiləsi parametrlər vasitəsilə yerinə yetirilir. Proqramın verilənlər bölməsində prosedur/ funksiyanın adını parametrlər ilə birlikdə yazmaqla proqram ilə prosedur/ funksiya arasında verilənlər mübadiləsi yaradılır. Yəni parametrin təsnifatından asılı olaraq parametr vasitəsilə məlumat prosedur/ funksiyanın daxilinə və ya tərsinə prosedur/ funksiyadan proqrama ötürülür.

Prosedur və funksiyalar yazılarkən onların adlarında göstərilən parametrlər formal, müraciət vaxtı göstərilən parametrlər isə faktiki parametrlər adlanır. Faktiki parametrlədə tiplər göstərilir, faktiki parametrlərdə göstərilmir.

Formal və faktiki parametrlər adlarına görə fərqlənə bilərlər. Lakin məzmunları eyni olacaqdır.

Melumatlar mübadiləsinə görə Pascalda parametrlər aşağıdakı qruplara ayrılırlar.

1)Parametr -məzmun;
2)Parametr- dəyişən;
3)Parametr- konstant;

1) Parametr -məzmun

Bu parametrlər vasitəsilə məlumatlar proqramdaın prosedur/ funksiyaya ötürülür. Geri proqrama qaytarılmır. Ona görədə bu parametrlər prosedur/ funksiya daxilində dəyişilsədə, kənarda əvvəlki qiymətlərində qalırlar.

Belə parametrlərə malik prosedur başlıqları

```
Procedure My_Prpg(Par1, Par2 : Type1; Par3: Type2);
                                  ;
```

şəklində yazılır.

Bu başlıqdakı Par1, Par2; Par3 parametrləri "parametr-məzmun" qrupuna daxil olan parametrlərdir. Göründüyü kimi, belə para-metrin qarşısında heç bir "açar" sözü əlavə olunmamışdır.

Parametr- məzmun parametrləri **file** tipdən başqa ixtiyari tipli dəyişən və konstant ola bilərlər.

2)Parametr -dəyişən

Bu parametrlər vasitəsilə məlumatlar prosedur/ funksiyadan proqrama ötürülür. Ona görədə adətən, alınmış nəticələr bu parametrlərə mənimsənilir.

Prosedur/ funksiya başlıqlarında formal parametrlərin "parametr dəyişən" olduğunu göstərmək üçün parametrin qarşısında **var** sözü əlavə olunur. Məsələn,

```
Procedure My_Prpg(var Par1, Par2 : Typel; var
                  Par3: Type2) ;
```

Parametr- dəyişən kimi, **file** tipidə daxil olmaqla ixtiyari tipli dəyişəndən istifadə etmək olar. Lakin Parametr- dəyişən kimi konstantdan istifadə etmək olmaz.

3) Parametr -konstant

Bu parametrlər vasitəsilə konstantlar prosedur/ funksiyaya göndərilir.

Parametr -konstanta malik prosedur başlığı

```
Procedure My_Prpg(const Par1, Par2: Typel; const
Par3:Type2) ;
```

şəklində yazılır.'Göründüyü kimi parametr- konstantı göstərən parametrlərin qarşısında **const** sözü.əlavə olunur.

Parametr- konstant kimi **file** tipdən başqa müxtəlif tipli konstantlardan və dəyişənlərdən istifadə etmək olar.

Bütün konstantlar kimi, parametr - konstanıtda dəyişmək, yəni ona nəsə mənimsətmək olmaz. Həmçinin parametr - konstant digər prosedur/ funksiyalara faktiki parametr kimi ötürülə bilməz. Bu parametr vasitəsilə mürəkkəb strukturlara malik verilənlərin strukturunu mübadilə etmək məqsədə uygundur.

Calışma 1. Verilmiş al və a2 tam ədədlərinin ən böyük ortaq bölənin (ƏBOB) tapılması. (Evklid alqoritmi).

```
program ABOB_1;
 var
nd,al,a2:integer;
function ABOB(a,b:integer):integer;
var
   r:integer;
   begin
        while  (a mod b)<>0 do
        begin
          r:=a mod b ;
           a: =b ;
                      b:=r;
       end;
        ABOB:=b;
       end;
{ Əsas proqram }
Begin
 readln(al,a2);
nd:=ABOB(al,a2);
```

```
  writeln (al ,' va  ',a2 ,' adadlarinin ABOB-u=',nd);
    readln;   end.
```

Çalışma 2: $s = \sum_{i=1}^{m} a_i^k$. prosedurundan istifadə etməklə

$$sum = \frac{\sum_{i=1}^{10} a_i^2}{\left|\sum_{i=1}^{7} a_i\right|} + \sqrt{\sum_{i=1}^{6} b_i}$$ cəmini hesablamalı.

```
program pros_l;
type Tc = array[1..10]  of real;
var
     a,b,c:Tc;
     sl,s2,s3,sum :real;
      i:byte;
 procedure p(m,k:byte; C:Tc; var s:real);
 begin
        s :=0;
        for i:=l to m do
        s:=s+exp(k*ln(c [i])) ;
end;
                  { Əsas proqram }
begin
   for i:=1 to 10 do readln(a[i]);
   for i:=l to 6 do readln(b[i]);
    p(10,2,a,sl);     writeln('sl=',si:6:2);
    p(7,l,a,s2);      writeln ('s2=',s2:6:2);
   p(6,l,b,s3);      writeln ('s3=',s3:6:2)
■ mi: =sl /abs (s2 ) +sqrt (s3) ;
      writeln('sum=',sum:6:2);
      readln(i); end.
```

Çalışma 3: **m** = $\max_{i \in [1,k]} \{a_i\}$ prosedurundan istifadə etməklə

$MaxA = \max_{i \in [1,10]} \{a_i\}$; $MaxB = \max_{i \in [l,12]} \{b_i\}$

maxsimumlarini hesablamalı.

```
Program r4;
VI>e TA= array [1.. 12] of real;
var
a,b:TA;
  i:integer;
M:real;
```

```
procedure Max(a:Ta;κ:integer;  var M:real);
 begin
     M:=a[1] ;
     for i:=1 to κ do
     if M<a[i]  then     M:=a[i];
end;
begin
    for i:=1 to 10 do readln(a[i]);
    Max(a,10,M);
    writeln('MaxA=',M:6:1);
    for i:=l to 12 do readln (b [i]);
     Max(b,12,M) ;
writeln( 'Maxb=',M:6:1);
 readln; end.
```

5. Rekursiv prosedur və funksiyalar Hər hansı məsələnin həll proqramını yazmamışdan əvvəl onun ilk alqoritmi tərtib olunur. Alqoritmləri yığcam şəkildə tərtib etmək üçün müxtəlif üsul və vasitələrdən istifadə olunur. Belə vasitələrldən biri də rekursiv obyektlərdən istifadədir. Rekursiv obyekt elə obyektlərə deyilir ki, onlar qismən özü vasitəsilə təyin olunur. Rekursiv obyektə misal olaraq,

$$y_{i+1} = f(y_i, x)$$

Funksiyasını misal göstərə bilərik. Göründüyü kimi, funksiyanm i + 1- addımındakı qiymətini hesablamaq üçün onun-i-ci addımdakı qiyməti hesablanılmalıdır. Bu isə öz-özü ilə təyin olunma deməkdir.

Başqa bir misal. Faktorialın

$$n! = n*(n-1)!$$

düsturu ilə hesablanmasını misal göstərə bilərik.

Faktorialın yuxarıdakı düsturla hesablanmasını proqramlaşdırma dilinə çevirsək alarıq:

```
program Factorial;
var
n,i:integer;
  function Fact(n:integer):  Longint ;
    begin
       for i:=1 to n do
         if i=l then Fact:=l
                else Fact:=i*Fact(i-1);
    end;
begin
   Write('n adadini daxil edin  '  );
   Readln(n);
   Writeln ('Factorial n! = ',Fact(n));
    readln(n)
end.
```

Çalışma 4 (Fibonaççi ədədləri). $F_1 = 1, F_2 = 1, F_3 = F_1 + F_2, \ldots, F_n = F_{n-2} + F_{n-1}, \ldots$ qaydası ilə düzəldilmiş ədədi ardıcıllığa daxil olan ədədlər Fibonaççi ədədləri adlanır. Qiymətləri 3000 natural ədədindən kiçik olan Fibonaççi ədədlərini tapmalı.

```
program Fibonacci;
var
  n,Fl,F2,F3:integer;
   function Fib(F1,F2:integer):integer;
          begin
            Fib:=Fl+F2;  end;
  begin
   write('n=');readln(n);
   Fl:=l;  F2:=l;  F3:=Fib(Fl,F2);
   Write( F l ,  ' ,  ' , F2);
   repeat
        Write( ' ,  ' , F3);
     Fl:=F2;F2:=F3;F3:=Fib(Fl,F2);
     until F3>n ;
      readln;       end.
```

Cavab:

n=3000

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584

Prosedurları da rekursiv prosedur şəklində ifadə etmək olar. Rekursiv prosedura misal olaraq aşağıdakı formanı misal göstərə bilərik:

```
Program Rec;
 Begin
                S1;
        If      Şərt then Rec;
```

S2;

End.

Proqram nümunəsi.

Simvollar sətirinin simvollarını əks istiqamətdə çapa çıxarmalı. Bu məsələni həll etmək üçün aşağıda verilmiş Reverse_String proqramında Reverse rekursiv funksiyasından istifadə olunmuşdur.

```
program Reverse_String;
Procedure Reverse ;
var Ch :  Char ;
begin
      if not eoln then
       begin
            read(Ch);              Reverse;
            Write(Ch);
       end;
  end ;  { Procedure-  son }
   begin
      Reverse      ;
   end.
```

Proqramın işləmə mexanizmini başa düşmək üçün xatırladaq ki, bu proqramda istifadə olunan Eoln funksiyası sətir sona çatdıqda false, əgər hələ sətir sona çatmamışdırsa True qiymətini alır.

4.2. Modullar

Modul ayrılıqda icra olunan proqram deyildir. O proqramlar kitabxanasıdır. Bu kitabxanadakı Moduldakı prosedurlardan funksiyalardan və digər resurslardan (tiplərdən, verilənlərdən) baş proqram adlanan proqramlar götürüb istifadə edə bilərlər. Bu da daha mürəkkəb proqramların tərtibını sadələşdirir, onların etibarlı işləməsini təmin edir, yaddaşa qənaət edir və s. Bu məqsədlə Pascal proqramlaşdırma sistemlərində bir sıra standart modullar adlanan modulla daxil edilmişdir. Belə modulları proqramçılarda tərtib edə bilərlər.

Pascal-a

System, Dos, Ctr, Graph, Strings, Overlay, Winapi, Windos,Wincrt,Winprn modulları daxildir.

Tərtib olunan proqramda hər hansısa modulun vasitələrindən istifadə olunduqda həmin modulun adı proqramın verilənlərin təsvirı bölməsində:

Uses <Modullar sıyahısı>

yazılmalıdır. Yalnız **System** modulu istisnalıq təşkil edir. Bu modulu təsvir etmədən də onun vasitələrindən, o cümlədən prosedur və funksiyalarından hər bir proqramda istifadə etmək olar.

Dediyimiz kimi Pascalda proqramçıların özləridə tərib etdikləri proqram və prosedurları "bir yerə cəmləyərək" modul yarada bilərlər. Belə modullara adətən istifadəçi modulları deyirlər. İstifadəçi Modulunun strukturu aşağıdakı kimidir:

Unit Modulun__adı;

Interface

{İnterfeys bölməsi}

implementation

{Realizasiya bölməsi}

begin

{İnsializatsiya bölməsi}

End.

İnterfeys **(Interface)** bölməsində digər modulların və baş proqramın istifadəsinə icazə verilən verilənlərin təsviri, prosedur və funksiyalanrın adlarının siyahısı verilir. Prosedur və funksiyaların formal parametrləri və onların tipləri də göstərilməlidir.

Realizasiya bölməsində**(implementation)** "interfeys" bölməsində adları çəkilən prosedur və funksiyaların proqramları yazılır. Belə prosedur və funksiyaların adları parametrsiz yazıla bilər. Əks halda parametrlərin tipləri göstərilməklə yazılmalıdır.

İnsializasiya bölməsində **(begin... end.)** modulun konekt işləməsi üçün zəruri olan operatorlar-direktivlər yazılır. Əksər halda bu bölmə buraxılır.

Proqram nümunəsi_l.

```
unit Modul_Sum;     (Modul)
interface
 var x,y: integer;
  function Sum(x,y:integer):integer;
  implementation
     function Sum(x,y:integer):integer;
      begin
          sum:=x+y;
     end;
end.
program Main_Sum;     {Baş proqram}
uses
       Modul_Sum;
begin
write('x='); readln(x);
 write('y='); readln(y);
 writeln('sum=',sum(x,y):3);
 readln;
end.
```

Burada göründüyü kimi, Main_Sum - baş proqram, Modul_Sum-isə moduldur. Moduldakı Sum(x,y) funksiyası iki ədədi cəmləyir.

Diqqət. *Modulların adları ilə onların yaddaşdakı fayllarının adları eyni olmalıdır. Məsələn, yuxarıdakı nümunədə* Modul_Sum *modulu yaddaşa da* Modu l_Sum. pas *adı ilə yazdmalıdır.*

Proqram icraya çağırılarkən adi qaydada baş proqram, yuxarıdakı nümunədə, Main_Sum proqramı icraya çalışılmalıdır. Özü avtomatik olaraq modula müraciət edəcəkdir.

Proqram nümunəsi 2.

Kompleks ədədlər üzərində hesab əməllərini yerinə yetirmək üçün proqram. Bu proqramda Modul_HCmplx modulundan istifadə olunmuşdur.

```
program Main_HCmplx;
 uses
        Modul_HCmplx;
var
 x, y,z:  Complex;
  s:char;
  begin
     write('x.re=');readln(x.re);
      write('x.im=');readln(x.im);
     write('y.re=');readln(y.re);
     write('y.im=');readln(y.im);
      write('s=');readln(s);
      case 5 of
      '+':  z:=Add.C(x,y) ;
       '-':'z:=SubC(x,y);
        '*':  z:=MulC(x,y);
     '/':  z:=DivC(x,y);
    end;
       write ( ' (',x.re:4:1,',';x.im:4:1, ') ')';
       write (S);
       write ( ' (',y.re:4:1,`,`,y.im:4:1, ') ' );
       write('=');
     writeln('(',z.re:4:1, ', ',z.im:4:1,')');
      Readln;
    end.
unit Modul_HCmplx;
interface
type
        Complex=record
                  re,im:  real;
          end;
        function AddC(x,y:Complex):Complex;
         function SubC(x,y:Complex):Complex;
         function MulC(x,y:Complex):Complex;
         function DivC(x,y:Complex):Complex;
var
     Result: Complex;
      implementation
        function AddC (x, y: Complex):Complex;
        begin
          Result.re:=x.re+y.re;
        Result.im:=x.im+y.im;
        end;  //Addc
        function SubC(x,y:Complex):Complex;
         begin
          SubC.re:=x.re-y.re;
          SubC.im:=x.im-y.im;
          end;  //SubC
        function MulC(x,y:Complex):Complex;
        begin
        MulC.re:=x.re*y.re-x.im*y.im;
        MulC.im:=x.re*y.im+x.im*y.re;;
```

```
end; //MulC
function DivC(x,y:Complex):Complex;
var
    z:real;
begin
    z:=sqr(y.re)+ sqr(y.im);
    DivC.re:=(x.re*y.re+x.im*y.im)/ z
    DivC.im:=(x.re*y.im-x.im*y.re)/z
    end; //DivC
    end.
```

4.3. DLL proqramlar kitabxanası

1. Ümumi məlumat

Proqramlarda hər hansısa modula daxil olan prosedur və ya funksiyadan istifadə etdikdə həmin prosedur/ funksiyalar proqraınm kompilyasiyası zamanı əsas proqrama əlavə olunaraq, vahid proqram əmələ gətirir. Ona görədə əsas proqram icra olunduqda onunla birlikdə istifadə olunan prosedur / funksiyalar da operativ yaddaşa yüklənir və proqraınm icrası sona çatana qədər yaddaşda qalır. Bu da proqramların yüklənməsi zamanı müəyyən çətinliklər törədir.

Prosedur / funksiyaları əsas proqramla kompilyasiya zamanı statik şəkildə deyil, proqramın icrası zamanı dinamik şəkildə rabitələndirmək olar. Bu məqsəd üçün proqramlarda modullar əvəzinə, ona oxşar olan dinamik rabitəli proqramlar kitabxanasından-**Dynamically Linced Libraries** (DLL) istifadə olunur. DLL-ə daxil olan prosedur / funksiyalar

dinamik dəyişənlər kimi, lazım olduqda operativ yaddaşa yüklənir, lazım olmadıqda isə onların yaddaşdakı yeri boşaldılır.

Dinamik rabitəli proqramlar kitabxanasi " modul" anlayışının davamı və inkişafı olsada, ona nəzərən müəyyən məhdudiyyətlərə də malikdir. Məhdudiyyətlər ondan ibarətdir ki, modullara modul vahidi kimi prosedur və funksiyalardan başqa, tiplər, dəyişənlər, konstantlar və s. də daxil ola bildiyi halda, DLL -lər yalnız prosedur və funksiyalardan təşgil oluna bilər.

Dinamik rabitəli proqramlar kitabxanasının strukturu praktiki olaraq adi proqramlarin strukturu kimidir. Fərq ondadır ki, proqram başlığındakı Program xidməti sözü, DLL-də Library sözü ilə əvəz olunur.

2. Dinamik rabitəli proqramlar kitabxanasının (DLL) strukturu

Library Kitabxananin_adı;

Eksport olunacaq prosedur və funksiyalar yazılır. Prosedur və funksiyaların adları parametrlər və onların tipləri göstərilməklə tam yazılır və adlarının sonuna **export** sözü əlavə olunur.

exports

Eksport olunacaq prosedur və funksiyaların adları parametrlər göstərilməməklə yazılır. Adlarının sonuna **index** sözü və hər hansısa tam ədəd əlavə olunur. Bu ədəd müəyyən mənada prosedur və funksiyanın vaciblik dərəcəsini göstərir. Prosedur və funksiyalar başqa adlada eksporta göndərilə bilər.Belə halda ilkin addan sonra, **name** sözü və yeni ad yazılmalıdır.

Məsələn,

```
Vex_Interse index 3 name Interse;
Begin
        İnsializasiya bölməsi. Çox vaxt buraxılır.
End.
```

3. DLL-dəki prosedur və funksiyalara müraciət qaydaları.

DLL-dəki prosedur və funksiyalara üç üsulla əsas proqramdan və ya hər hansısa moduldan müraciət oluna bilər. Adı ilə, Məsələn,

```
Procedure MyProc; external; 'MyLib' ;
```

Yeni ad ilə, Məsələn,

```
Procedure MyProc; external;
        ' MyLib'   ;name  'NewName'
```

Tərtib nömrəsi ilə, Məsələn,

```
Procedure MyProc;  external;  ' MyLib'   ;index 7;
```

DLL prosedur və funksiyalarını həm proqramlar, həmdə modullar özlərinə import edə bilərlər. Yəni onlara müraciət edə bilərlər. Bu müraciət əmrləri proqramlarda operatorlar bölməsində, modullarda isə, realizatsiya (implementation) bölməsində yazılmalıdır.

Proqram nümunəsi. Yuxarıda modul şəklində tərtib etdiyimiz proqramları DLL şəklində tərtib edək.

İki ədədin cəmlənməsi

```
Library Sum_L;
    function Sum(x,y:integer):integer;  export;
     begin
       sum:=x+y;
       end;
        Exports    Sum;
        Begin    end.

program Main_LSum;          {Baş proqram}
  function .Sum(x,y:integer) :integer; External
  'Sum_L';
  var  x,y,s:integer;
  begin
  write('x='); readln(x);
   write('y='); readln(y);
   s:= Sum(x,y)  ;
  writeln('Sum=',Sum(x,y));
  readln;
  end.
```

Qeyd: Library Sum_L proqramı kompilyasiya olunmalıdır.

2. Kompleks odədlərin cəmlənməsi. Bu proqram Delphinin Konsul rejimində yazılmışdır.

■ **DLL proqramı**

```
library Complex    ;
 uses SysUtils;
 type
TComplex= record
 Re,  Im:  Real;
end;
{$R *.res}
 function CmplxAdd(x,y:TComplex):TComplex;
  Export;
        begin
Result.re:=x.re+y.re;
Result.im:=x.im+y.im;
end;  //Addc
 function CmplxSub(x,y:TComplex):TComplex;
Export ;
        begin
      Result.re:=x.re-y.re;
      Result.im:=x.im-y.im;
       end;  //SubC
     function CmplxMul(x,y:TComplex):TComplex;
     Export;
        begin
      Result.re:=x.re*y.re-x.im*y.im;
      Result.im:=x.re*y.im+x.im*y.re;;
       end;  //MulC
       function CmplxDiv(x,y:TComplex):TComplex;
Export;
        var
                z:real;
        begin
          z:=sqr(y.re)+ sqr(y.im);
           Result.re:=(x.re*y.re+x.im*y.im)/z;
          Result.im:=(x.re*y.im-x.im*y.re)/z;
          end;  //DivC
          exports
      CmplxAdd index 1 name  ' ADDC'  resident,
       CmplxSub index 2,
      CmplxMul index 3,
      CmplxDiv index 4;
      begin
      end.
```

DelPhinin Консул rejimində tərtib olunmuş library Complex DLL proqramı Проект | Компилировать əmrı ilə kompilirovat olunmalıdır.

■ **Əsas (Baş) proqram.**

```
program DLL;
```

```
{$APPTYPE CONSOLE}
 uses
      SysUtils;
  type
        TComplex=record
         Re,Im:  real;
         end;
   function ADDC(x,y:TComplex):TComplex;External
    'Complex';

   function SubC(x,y:TComplex):TComplex;External

  'Complex'  index 3;

   function MulC(x,y:TComplex):TComplex;External

  'Complex`  index 4;

 function DivC(x,y:TComplex):TComplex;External

'Complex' name  'CmplxDiv';

 var

 x,y,z: TComplex;

 s:char;

 begin

     write ( ' x. re=' ) ; readln (x . re) ;
      write ( ' x . im==' ) ; readln (x . im) ;
     write('y. re=');readln(y.re);
      write ( ' y . im=' ) ; readln (y. im) ;
      write ( ' s=' ) ; readln (s) ;
      case s of
     '+':   z:=AddC(x,y) ;
     '-':   z:=SubC(x,y) ;
     '*':   z:=MulC(x,y);
     '/':   z:=DivC(x,y);
      end;
       write( ' ( ' , x . re : 4 :1, ' , ' , x . im: 4 : 1, ' ) ' ) ;
       write(S);
       write ('(',y.re:4:1,',',y.im:4:1,')');
        write(' = ');
writeln(' (' ,z.re:4:1, ', ',z.im:4:1, ') ');
```

```
{ TODO -oUser -cConsole Main :  Insert code here }
  ReadIn;
  end.
```

4.4. Turbo Pascal-ın qrafik imkanlari

Qrafik rejimin insializasiyasi.

Turbo Pascal proqramdarı icra olunarkən Monitor

- Mətn və ya
- Qrafik rejimdə ola bilər.

Mətn rejimindən adətən rəqəm-hərf şəklində olan məlumatları əks etdirmək üçün istifadə olunur. Adi halda, yəni Turbo Pascal yüklənərkən monitor Mətn rejimində olur. Monitoru mətn rejimindən qrafik rejimə gətirilməsi əməliyyatına monitorun qrafik rejimdə insializasiyasi deyilir və bu əməliyyat

```
InitGraph (grDriver,grMode,grPath);
```

proseduru vasitəsilə edilir.

Burada

- GrDriver- parametri integer tipli konstanta (məsələn VGA) olub, monitorun işini idarə edən adapteri göstərir.

- GrMode- parametridə integer tipli konstanta olub, VidioSistemin rejimini göstərir. Bu rejim təsvirlərin kefiyyətini müəyyən edir.
- GrPath- parametridə string tipli olub, drayverin diskdəki yerini- yolunu göstərir. Turbo Pascalda bu drayverlər BGI qovluğunda saxlanılır.

İnsializasiyanın normal yerinə yetirilib, yetirilməməsini GraphResult funksiyası vasitəsilə yoxlamaq olar.

İnsializasiyadan sonra bu funksiyanın qiyməti grOk sabitinə bərabərdirsə, insializasiya normal getmişdir, əks halda yox.

Qrafik rejimdən mətn rejiminə CloseGraph əmri ilə qayıdılır.

Sonda qeyd edək ki, Mətn rejimində monitoru idarə edən prosedur və funksiyalar Turbo Pascalın Crt modulunda, Qrafik rejimdə idarə edən prosedur və funksiyalar isə Graph modulundadır.

Qrafik rejimin insializasiyasına aid proqram nümunəsi.

```
program Graf_Ins;
      uses Graph;
       var
         grDriver,grMode,ErrCode:integer;
```

```
        grPath:string;
  begin
        grDriver:=VGÄ;
         grMode:=VGAHi;
        grPath:='C:\BP\BGI';{drayver
        Egavga.bgi bu katalogdadir}
      InitGraph(grDriver,grMode,grPath);
      ErrCode:=GraphResult;
      If  ErrCode<>grOk then    begin
      Writeln ('Insializasiyasida sahv var');
       readln;
      Halt (1);
       end;
Writeln(‘ Grafik rejimin
 insializasiyasi yrina yetirildi');
                  readln;
               CloseGraph;
end.
```

Turbo Pascalda Qrafik rejimdə işləmək üçün bir sıra prosedur və funksiyalar nəzərdə tutulmuşdur:

1. GetMaxX , GetMaxY funksiyaları ekranın işçi oblastının sol aşağı küncünün kordinatlarını göstərir. VGA rejimində bu koordinatlar (639, 479) -dur. Sol yuxarı küncünün koordinatları isə (0,0) -dır.
2. GetX , GetY funksiyaları kursorun cari koordinatlarını göstərir.
3. MoveTo (x: integer, y: integer) - kursoru (x, y) nöqtəsinə gətirir.
4. SetColor(rəngin_adı) ekrana çıxarılacaq elementin (məsələn, nöqtə, xətt, və s.) rəngini müəyyən edir. *Rənglər siyahısı 3.1 paraqravfında verilmişdir.*
5. PutPixel (x, y,rəng) - (x,y) nöqtəsini çəkir.
6. Line (xl,yl,x2,y2) - (xl,yl) nöqtəsi ilə (x2,y2) nöqtəsini birləşdirən düz xətt çəkir.
7 . LineTo (x, y) - kursorun cari koordinatı ilə (x, y) nöqtəsini birləşdirən düz xətt.
8. Circle(x,y,r) - mərkəzi (x, y) - nöqtəsində olan r radiuslu çevrə.
9 . Ellipse(x, y,başlanğıc_bucaq, son_bucaq, RadusX, RadiusY)-ellips sektoru. Burada(x,y) - ellipsin mərkəzi, RadusX, RadiusY-üfıqi (harizontal) və şaquli(vertikal) oxlar.

10. Xüsusi halda başlanğıc_bucaq=0, son_bucaq=360 olsa sektor əvəzinə tam ellips çəkiləcəkdir.
11. RadusX= RadiusY olsa ellips sektoru əvəzinə çevrə sektoru çəkiləcək.
12. Rectangle (xl,yl,x2,y2) - Burada (xl, yl) ,(x2,y2) nöqtələri düzbucaqlının baş dioqonalının başlanğıc və son nöqtələridir. Məsələn Rectangle (0, 0, GetMaxX , GetMaxY)-düzbucaqlısı ekranın işçi oblastını çərçivəyə alacaqdır.

Multiplikasiya. Multiplikasiya sözünün altında təsvirlərin ekranda hərəkət etdirilməsi başa düşülür. Bunun sadə alqoritmi mövcuddur:

1. Təsvir ekrana çıxardılır;
2. Qısa fasilədən sonra o silinir
3. Təsvir yəni yerdə əvvəlki təsvirə nəzərən müəyyən qədər sürüşdürülmüş yerdə ekrana yenidən çıxardılır.

Aşağıdakı proqram dairənin ekranın solundan sağına hərəkətini nümayiş etdirir.

```
program move;
      uses Graph,Crt;
```

```
var
  x,y,r:integer;
  dx,dt:integer;
   grDriver,grMode,ErrCode:integer;
   grPath:string;
begin
   grDriver:=VGA;
    grMode: ==VGAHi ;
   grPath:= 'C:\BP\BGI';{drayver
   Egavga.bgi bu katalogdadir}     \
InitGraph(grDriver,grMode,grPath);
 ErrCode:=GraphResult;
if ErrCode<>grOk then   begin
Writeln('Insializasiyasida sahv var');
readln;
     Halt(1);          end;
       x:=0;    y:=0;   r:=25;   dx:=2;
                                dt:=200;
                          while   x<639 do
                                    begin
      SetColor(Yellow);
```

```
        Circle(x,y, r) ;
        Delay(dt);
        {atmiktosaniya
        saxlayır}
      SetColor(0) ;
      Circle(x,y,r);
      x:=x+dx;
       end; readln;
      CloseGraph;
end.
```

Qrafik rejimdə mətinlərin çapı.

1. `Write` (Sətir) , `Writeln` (Sətir) - prosedurları qrafik rejimdədə, mətn rejimindəki kimi çap edir. Xüsusi əmrlərlə çap zamanı şriftləri, ölçülərini, çapın istiqamətini dəyişmək mümkün deyil.
2. `OutText` (Sətir)- Kursorun cari mövqeyindən başlamaqla bir sətir çap edir.
3. `OutTextXY (x, y,` Sətir)- Bir sətir çap edir. Burada `(x,y)` –nöqtəsi çap oblastının sol yuxarı küncünü göstərir.

4. SetTextStyle(Şrift, Çap_istiqaməti, Ölçü)- proseduru

OutText və OutTextXY- vasitəsilə çap olunan mətnin

xüsusiyyətlərini müəyyən edir.

- Şrift parametri - 0,1,2,3,4- qiymətlərini ala bilər.
- Çap_istiqaməti= 0, olduqda OutText və OutTextXY prosedurları üfüqi istiqamətdə, =1 olduqda isə şaquli istiqamətdə çap edəcəklər.
- Ölçü - şriftin ölcüsünü göstərir.

Məsələn,

```
SetTextStyle (0, 1, 2);
OutTextXY (100,200,' Vertikal istigamatda cap') ;
SetTextStyle (0, 0, 2);
OutTextXY (100,200, Horizontal istigamatda cap');
```

5. Fayllar

I. Məntiqi və fiziki fayl anlayışı

Kompüterin yaddaşı daxili yaddaş və xarici yaddaş adlandırılmaqla iki yerə ayrılır. Daxili yaddaşa əməli yaddaş, xarici yaddaşa isə disklər (və ya disketlər) də deyilir. Əməli yaddaşa yazılan məlumat, proqramdan çıxdıqda- proqram bağlandıqda və ya kompüter cərəyan şəbəkəsindən ayrıldıqda itir - pozulur. Belə məlumata yenidən ınüraciət etmək olmur. Bu deyilənlər xarici yaddaşa -disklərə aid deyildir.

İndiyə kimi tərtib etdiyimiz məlumatlar daxili yaddaşa- əməli yaddaşa yazılıb- oxunurdu. İndi isə xarici yaddaşla işləməyi, yəni məlumatların yaddaş disklərinə yazılıb oxunmasını öyrənək. Bu isə fayllarla işləməyi öyrənmək deməkdir.

Məlumatlar yazılmış və ya yazılacaq müəyyən adla adlandırılmış xarici yaddaş daşıyıcısının hər hansısa oblastı fayl adlanır. Belə fayla fiziki fayl, deyilir. Fayla başqa cür də tərif verirlər. Fayl proqramda istifadə olunan müəyyən strukturlu verilənlərdir. Belə fayla məntiqi fayl deyilir. Proqramda məntiqi fayl, fayl dəyişəni adlanan müəyyən tipli dəyişənlə ifadə olunur. Fiziki fayl sərt və ya elastiki disklərdə bir-birinin ardınca yazılmış yaddaş baytları ardıcılığıdır.

Məntiqi fayl isə faylın proqramda və fiziki faylda ifadə olunma
strukturudur. Obrazlı desek, məntiqi fayl fiziki fayla baxmaq üçün
"şablondur" (pəncərədir).

Faylların məntiqi strukturu massivlərin məntiqi strukturu kimidir. Fərq ondadır ki,

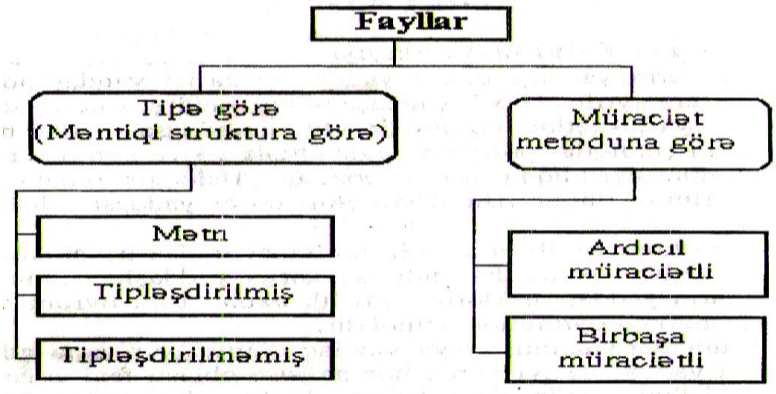
- Massivlər operativ yaddaşda saxlanılır, fayllar xarici yaddaşda.
- Massivlər üçün yaddaşda yer qabaqcadan ayrılır. Onun ölçülərini dəyişmək olmaz. Fayldakı elementlərin sayı qabaqcadan göstərilmir. Proqram yerinə yetirilərkən bu elementlərin sayı - faylın ölçüsü dəyişir. Fayldakı elementlərin sayını müəyyənləşdirmək üçün faylın sonunda yerləşdirilmiş Eof adlı simvoldan istifadə olunur.

2. **Faylların təsnifatı**

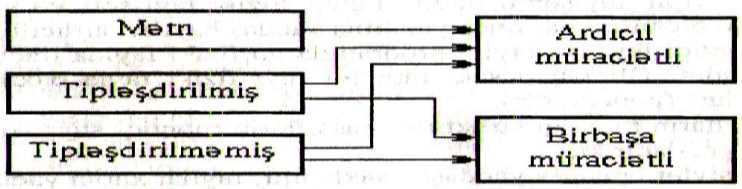
Fayllar iki əlamətə görə təsnifatlandırılır:

- Tipə (məntiqi struktura) görə ;
- Faylın elementlərinə müraciətə görə.

Bu təsnifat aşağıdakı sxemdə verilmişdir.



Şəkil 5.1 Faylların təsnifatı



Şəkil 5.2. Faylların mümkün müraciət üsulları

Fayllarla işləmək, yəni xarici yaddaş daşıyıcılarına (disklərə) məlumatın yazılıb oxunma texnalogiyasını aşağıdakı konseptual qaydada yerinə yetirmək olar:

1. Faylın strukturu- tipi elan olunmalıdır. Bu proqramın verilənləri təsvir etmə bölməsində fayl dəyişəni vasitəsilə edilir.
2. Tərtib olunacaq (və ya əvvəllər tərtib olunmuş) faylın adı və yaddaşdakı yeri- faylın yolu göstərilməlidir. Bu əməliyyata fayl dəyişəni ilə fiziki faylın əlaqələndirilməsi deyilir və Assign proseduru vasitəsilə yerinə yetirilir.

Pascal proqramlaşdırma sistemi tərəfindən tərtib olunmuş fayldan istifadəyə icaza verilməlidir. Bu icazəyə faylın açılması deyilir. İcazə alınarkən istifadənin məqsədi göstərilməlidir. Bu məqsəd

- Fayla yazmaq,
- Əlavə etmək və
- Oxumaq

ola bilər. Sonra bu əməliyyatlar yerinə yetirilməlidir. Qeyd edək ki Гну 1 açılarkən bu əməliyyatlardan yalnız biri yerinə yetirilə bilər. Və Sonra başqa əməliyyat yerinə yetirmək üçün açılmış fayl bağlanılmalıdır.

Bu əməliyyaları yerinə yetirmək üçün isifadə olunan prosedur və funksiyalar faylın strukturundan-tipindən asılı olaraq müəyyən qədər bir-birindən fərqlənə bilər. Buna görədə faylın tipinə müvafiq bu əməliyyatlara ayrı-ayrılıqda baxaq.

5.1. **Mətn faylları**

1. **Fayl dəyişəni**. Mətn fayllarında `f` fayl dəyişəni

```
f: text
```

əmri ilə müəyyənləşdirilir.

Mətn faylları da **f ayl of char** tipli fayllar kimi simvol tipli verilənlərdən ibarət olsada, onların bir sıra fərqli cəhətləri də vardır. Bildiyimiz kimi fayllarda **Eof** adlı simvol faylın sonunu göstərir. Mətn fayllarında, digər fayllardan fərqli olaraq sətirin sonu da göstərilir. Bu məqsədlə **Eoln** adlı simvoldan istifadə olunur. Bu simvolun kodu **"возврат каретки"** və **"перевод строки"** adlı simvolların kodlarından (13 və 10) ibarətdir.

2. **Fayl dəyişəni ilə fiziki faylın əlaqələndirilməsi.** Bu əməliy-

yat, məsələn,

```
Assign(f, 'mf.dat')
```

əmri ilə yerinə yetirilə bilər. Burada, f- məntiqi faylın, mf. dat-cari diskdə, cari kataloqda olan faylın adıdır. Fayl cari diskdə, cari kataloqda deyilsə faylın yolu göstərilməlidir. Bu faylın yolunu başqa bir dəyişənlədə əvəz edib, Assign əmrində bu addan istifadə etmək olar. Məsələn,

```
Name:='a:/mf/mf.dat' ;
Assign(f, Name) ;
```

3- Faylların açılması və bağlanması

Mətn faylının açılması Reset (f), Rewrite (f) və ya Append (f) əmrlərinin biri ilə, bağlanması isə Close (f) əmri ilə yerinə yetirilir.

Rewrite (f) proseduru ilə yazmaq üçün f məntiqi faylına müvafiq yeni fiziki fayl yaradılır və açılır. Əgər belə fayl mövcuddursa, bu faylın elementləri silinir, boş fayla çevrilir.

Append (f) əmri ilə yeni fayl yaradılmır. Mövcud faylın sonuna yeni elementlər əlavə etmək üçün açılır.

Reset (f) əmri ilə f məntiqi faylına müvafiq fiziki fayl açılır. Əgər bu fayl mətn faylıdırsa, onu yalnız oxumaq mümkündür. Yox əgər tipləşdirilmiş fayldırsa onu həm oxumaq, həmdə ora yazmaq olar.

Əgər Append (f) , Reset (f) əmrlərində göstərilən f məntiqi faylına müvafıq adda fiziki fayl yoxdursa, proqram bu haqda məlumat verəcək və proqramın icrası dayanacaqdır. Bu dayanmanı {S i -} transilyator direktivi ilə ləğv etmək olar.

Bu transllyator direktivinin nəticəsini IOResult funksiyası vasitəsi ilə təhlil etmək olar. Bu funksiya mövcud faylı açmaq istədikdə heç bir səhv yaranmasa " 0 " , əks halda " 1" qiymətini alacaqdır. Məsələn,

```
Assign(f, File_Name) ;
{Si-}
Append(f) ;
If IOResult<>0 {əgər göstərilən adda fayl yoxdursa }
Then rewrite (f) ; {yenisini yaratmalı}
```

4. Fayllara yazıb- oxuma. Mətn fayllarında verilənlər **Read, Readln, Write** və **Writeln** əmrləri vasitəsilə daxil və xaric edilir. İndiyə kimi istifadə etdiyimiz bu klaviaturadan oxuma, monitorda və ya printerdə çap etmə əmrlərindən fərqli olaraq, bu zaman bu əmrlərin ilkin parametrləri kimi fayl dəyişəni göstərilməlidir. Məsələn,

```
Read(f,a,b) ;
```

Burada, **f** fayl dəyişəni, **a,b** isə adi dəyişənlərdir.

Eof (f) - ***funksiyası haqqında.*** Bu funksiya faylın sonuncu elementi oxunduqda True əks halda False məntiqi qiymətini alır.

Nümunə **1.** Bu proqram

satirl, satir2, satir3, satir4, satir5

sözlərini A:\test.txt faylına yazır.

```
program Fayll;
var
  f:text;
 i:integer;
 begin
assign(f,'A:\test.txt');
rewrite(f);
for i:=l to 5 do
 writeln(f,'satir' ,i) ;
close(f);
readln;
end.
```

Nümunə 2. Bu proqram A:\test.txt faylını oxuyur.

```
program Fayl4;
 var
   f:text;
   i:integer;
 s:string;
 begin
```

```
    assign(f, 'A:\test.txt') ;
    reset(f);
  while not eof(f)  do
   begin
   readln(f,s);
  writeln(s);
  end;
   close (f);
    readln;
   end.
```

Nümunə 3. Bu proqram A: \test. txt faylını silir və ora satir6, satir7 sözlərini yazır.

```
program Fayl2;

var
f: text;
i:integer;
begin
 assign(f, 'A:\test.txt') ;
```

```
 rewrite(f) ; for i:=6 to 7 do
  writeln(f,   'satir',i);
  close(f);
  readln;
  end.
```

Nümunə 4. Bu proqram satir6, satir7 sözlərindən ibarət olan A:\test.txt faylına satir6, satir7 sözləri əlavə edir.

```
program Fayl3;
var
   f:text;
  i:integer;
   begin
    assign(f, 'A:\test.txt');
     append (f);
     for i:=8 to 9 do
    writeln(f,'ctroka' , i) ;
    close(f) ;
    readln;
```

end.

Standart mətn faylları.

Klaviatura, printer və ekranla işi sadələşdirmək məqsədilə Pascal Sisteminə Input вя Output adlı iki mətn tipli fayl dəyişəni daxil edilmişdir.

Pascal proqramı icra olunarkən, Input klaviatura "fiziki faylı" ilə, Output isə ekran "fiziki faylı" ilə əlaqələnir və bu fayllar avtomatik olaraq açılır.

Ona görədə mətn fayllarının oxunma və yazma qaydalarına müvafiq klaviaturadan verilənlər

Readln(Input, <dəyişənlər siyahısı>);

Read(Input, <dəyişənlər siyahısı>);

prosedurları ilə oxunur,

Writeln(Output, <dəyişənlər siyahısı>);

Write(Output, <dəyişənlər siyahısı>);

prosedurları ilə isə ekrana yazılır.

Xüsusi hal kimi, bu prosedurlarda Input və Output fayl dəyişənləri buraxıla bilər.

Pascalda qəbul edilmiş mexanizmə uyğun olaraq, Input və Output fayl dəyişənləri digər fiziki fayllarla da əlaqələndirilə bilər.

Məsələn, tutaq ki, A: diskində yerləşən "Name . dat" faylına X, Y, Z dəyişənlərinin qiymətlərini yazmaq lazımdır. Adi qaydada bu əməliyyatı aşağıdakı proqramla yerinə yetirmək mümkündür.

```
program Prl;
  var f: text;
      x, y,z :real;
begin
      Read (x, y, z) ;

      Assign( f,'A:\name.dat') ;
       Rewrite( f ) ;
      writeln(f,x,y,z);
       close( f ) ;
end.
```

Bu proqramın gördüyü işi aşağıdakı proqramla da yerinə yetirmək olar.

```
program Pr2 ;
    x,y,z  :real;
begin
    Read(x,y,z);
    Assign(output,'A:\name.dat')  ;
    Rewrite(output);
      writeln(x,y,z);
      close(output);
end.
```

Printerdə çap.

Pascal proqramlaşdırma dilində klaviaturaya Input, ekrana isə Output fayl dəyişənləri, yəni məntiqi faylları əlaqələndirildiyi kimi, ekrana da "Con" adlı fiziki fayl kimi, printerə isə "Prn" və yal "Ltpl" adlı fiziki fayl kimi baxılır. Digər tərəfdən bildiyimiz kimi, bir məntiqi fayl ardıcıl surətdə bir neçə fiziki faylla əlaqələndirilə bilər, yəni bir neçə fiziki fayla "xidmət" göstərə bilər. Bu isə Output məntiqi faylını "Con" -ekran və "Prn" -printer fiziki faylları ilə əlaqələndirib, verilənləri ekranda və printerdə çap etməyə imkan verir.

Proqram nümunəsi:

```
program Output;
 uses
  WinCrt;
  Begin
       ClrScr;
         {Ekranda çap}
          Writeln('N.  Virt');
           {Çap qurğusunja çap}
           Assign(Output,'Prn');
            Rewrite(Output)  ;
            Writeln('Pascal-Delphi');
            Close(Output)  ;
            {Ekranda cap}
            Assign(Output,'Con');
             Rewrite(Output)  ;
          Writeln('Pascal-Delphi');
          Close(Output)  ;
   end.
```

Bu proqram nəticəsində əvvəlcə ekranda " N. Virt'" sözləri, sonra çap qurğusunda "Pascal-Delphi" və yenidən ekranda "Pascal-Delphi" çap olunacaqdır.

Bu proqram nümunəsində göstərilmiş qaydada Delphidə printerdə çap etmək mümkün deyildir. Delphidə çap etmə əməliyyatını aşağıdakı proqram nümunəsində göstərilmiş qaydada yerinə yetirmək etmək olar.

```
program Printer;
uses
      Printers;
       var
       MyFile:TextFile;
         к:integer;
begin
       k:=12346;
       AssignPrn(MyFile);
       Rewrite(MyFile);
       Writeln (MyFile, ' Принтер  ' ,'k)
       System.CloseFile(MyFile);
end.
```

Mətn fayllarının **file of char** fayllarından fərqli cəhətləri.

1.Ədədləri Mətn fayllarına yazarkən onlar avtomatik olaraq simvollar ardıcıllığına və tərsinə, oxuyarkən ədədlərə çevrilirlər.

2.Mətn fayllarına birbaşa müraciət qeyri-mümkündür.

3. Mətn fayllarına standart fayllar kimi, ancaq tam, həqiqi, simvol və sətir tipli dəyişənləri oxumaq və yazmaq olar.

4. Digər fayllardan fərqli olaraq bu fayllarardan oxumaq və fayllara yazmaq **Readln** və **Writeln** əmrləri ilə də mümkündür.

5.Bu fayllarda sətirin sonunu göstərən simvol da yadda saxlanılır.

6. Mətn faylları **Append (f)** əmri ilə açıla bilər. Bu zaman faylın sonuna yeni elementlər əlavə etmək mümkün olacaqdır.

5.2. Tipləşdirilmiş fayllar

1. Fayl dəyişəni. Tipləşdirilmiş faylların f fayl dəyişəni

f: file of <tip>

əmri ilə müəyyənləşdirilir.

Bu fayllarda elementlərin tipi fayldan başqa istənilən tipdə ola bilərlər.

Tipləşdirilmiş faylların elementləri sıfırdan başlamaqla ardıcıl olaraq sistem tərəfindən avtomatik olaraq nömrələnir. Bu da belə fayllara həm ardıcıl, həmdə birbaşa qayda ilə müraciət etməyə imkan verir.

2. Fayl dəyişəni ilə fiziki faylın əlaqələndirilməsi.

Bu əməliyyat, məsələn,

```
Assign(f, 'mf.dat');
```

və ya

```
Name:='a:/mf/mf.dat' ;
Assign(f, Name) ;
```

əmrləri ilə yerinə yetirilə bilər.

3. **Faylların açılması** və **bağlanması**

Rewrite (f) - Yazmaq üçün f məntiqi faylına müvafiq yeni fiziki fayl yaradılır və açılır. Əgər belə fayl mövcuddursa, bu faylın elementləri silinir, boş fayla çevrilir.

Reset (f) əmri ilə f məntiqi faylına müvafiq fiziki fayl açılır. Tipləşdirilmiş fayldırsa onu həm oxumaq, həmdə ora yazmaq olar.

Close (f) -əmri ilə fayllar bağlanılır.

4. Fayllara yazıb- oxuma.

Tipləşdirilmiş fayllardan oxumaq **Read,** fayla yazmaq isə **Write** əmri ilə yerinə yetirilir. Bu zaman yazma- oxuma vahidini (porsiyasını), faylın tipi ilə eyni tipdə olan dəyişən müəyyənləşdirir.

Read və Write operatorları aşağıdakı şəkildə yazılır.

Read(fayl dəyişəni, dəyişənlər siyahısı)

Write(fayl dəyişəni, dəyişənlər siyahısı)

Proqram nümunəsi:

```
program Fayle1;
```

```
type
      TFC= file of Char;
 var
      FC:  TFC;
      FB:  file of byte;
       Ch:  Char;
      B:  Byte;
 begin
    Assign(FC,'Test.dat' ) ;
Rewrite(FC);
   For ch:='0'  to  '9'  do write(FC,Ch);
   For ch:='A'  to 'J' do write(FC,Ch);
   Close(FC);
     Assign(FB,'Test.dat');
      Reset(FB);
     While not Eof(FB)  do
      begin
        Read(FB,B);
        Write(B:8);
   end;
      Close(Fb);
```

```
    Read(b);
end.
```

Nəticə :

48 49 50 51 52 53 54 55 56 57

65 66 67 68 69 70 71 72 73 74

5. 3. Tipləşdirilməmiş fayllar

Belə faylların fayl dəyişəni

```
f: file ;
```

əmri ilə elan olunur.

Bu fayl dəyişəni vasitəsilə ixtiyari məntiqi struktura malik fayla, simvol fayllarına müraciət kimi müraciət etmək olar. Fərq ondadır ki, belə fayllarda oxuma- yazma zamanı tipləşdirilmiş fayllardan fərqli olaraq, məlumatlar qabaqcadan faylın tipi ilə müəyyənləşdirilən porsiyalarla (vahidlərlə) yazılıb oxunmur. Yazdıb- oxuma porsiyası proqramçı tərəfindən müəyyənləşdirilə bilər. Bu da belə fayllarla işləməyi sürətləndirir.

Fayl dəyişəni ilə fiziki faylm əlaqələndirilməsi və faylların yazmaq və oxumaq üçün **açılması və bağlanması** əməliyyatları tipləşdirilmiş fayllarda olduğu kimidir.

Tipləşdirilməmiş fayllarda məlumatlar BlockRead, BlockWrite operatorları vasitəsilə yazılıb- oxunur.

Faylları açan (faylarla işləməyə icazə verən) Reset və Rewrite prosedurlarında fayl dəyişənindən başqa, ikinci dəyişəndəndə istifadə etmək olar. Bu parametr məlumatları yazarkən və oxuyarkən yazının ölçüsünü (porsiyasını) göstərir. Əgər bu parametr buraxılmışdırsa, onda bu parametr sistem tərəfindən 128 bayt qəbul olunur.

Proqram nümunəsi.

```
program TZF;
var
  F:file;
Buff:array[1..10]  of real;
i,j:integer;
begin
Assign (f, 'DD.dat');
Rewrite(f);
```

```
for i:=l to 3 do
 begin
for j:=l to 5 do Read(buff[j]);
 BlockWrite(f,Buff,5);
 end;
Close (f);
                {*--- *----*--- *   *---*}
      Assign(f,'DD.dat');
       Reset(f); for i:=l to 3 do
      begin
    BlockRead(f,buff, 5) ;
for j:=.l to 5 do Write(buff[j] : 8 : 2);
writeln;
 end;
Close(f);
readln(j);
end.
```

1. Fayllara ardıcıl müraciət

Ardıcıl müraciət qaydası fayllarla adi işləmək qaydasıdır.

Fayllara ardıcıl müraciət, yəni fayllara yazıb\ oxuma aşağıdakı əməliyyatlar vasitəsilə yerinə yetirilir.

1. Faylın dəyişənini təyin etməklə faylın tipi təyin olunur. Məsələn,

```
Type
    FTl=file of real ;              {Tipləşdirilmiş fayl}
    Ftext= text ;   {Mətnfaylı}
    FTm=file;     {Tipləşdirilməmiş fayl}
 Var
    F:   FT1;
    Ft:Ftext;
    Fm:FTm;
```

2. Fayl dəyişəni ilə fiziki fayl əlaqələndirilir. Məsələn,

```
Assign(f,'SD.dat');
 Assign (f, 'C:/Data/Sd.dat');
Name='A:/mf/mf.dat'   ;
Assign (f, Name) ;
```

3. Yazmaq üçün fayl açılır. Məsələn,

```
Rewrite(f);
 Rewrite(fm);
```

Rewrite (fm, 10) ; {10 baytdan ibarət blok}

4 . Fayla yazmaq

Write (f, X, Y) ; { Tipləşdirilmiş fayllara yazmaq }

BlockWrite (fm, A) ; { Tipləşdirilməmiş fayllara

BlockWrite (fm, A, 4) ; yazmaq}

5. Oxumaq üçün faylı açmaq

Reset (f);

Reset(fm,4);

6. Fayldan oxumaq

Read(f,X,Y);

BlockRead(fm,A);

BlockRead(fm,A,4);

7. Faylların bağlanması

Close(f);

Proqram nümunəsi.

A: diskində 15 həqiqi ədəddən ibarət fayl yaratmalı. Sonra bu faylın elementlərini 5 sütunlu, 3 sətirli massivə mənimsətməli və massivi 8:1 formatı ilə çap etməli.

```
program Massiv;
```

```
var
      F:  file of real;
       Mas:  array[1..3,1..5]  of real;
      MS:  real;
      i,j:  integer;
begin
  Assign(f,'A:\Dan.dat');
  Rewrite(f);
   For i:=l to 15 do
   begin
   Read(MS);
   Write (f,MS);
   end;
   Close(f);
  Assign(f, 'A:\Dan.dat') ;
   Reset(f);
   For i:=l to 3 do
  begin
          for j:=1 to 5 do
           begin
```

```
            Read(f,Mas[i,j]);
             Write(Mas[i,j] :8 : 1);
        end;
        Writeln;
        end;
        Close (f) ;
end.
```

Nəticə:

```
    82.5   32.0   43.9   4.3   85.2
    56.0   98.7   671.6   9.0   30.9
       11.9  312.0  913.6  81.3   98.1
```

8. Fayllara birbaşa müraciət

Fayllara birbaşa müraciət etmək üçün aşağıdakı funksiya və prosedurlar nəzərdə tutulmuşdur. \

- **FilePos-** göstəricinin fayldakı cari mövqeyini müəyyən edir *(mövqeylər sıfırdan başlamaqla nömrələnir).*
- **FileSize** - faylın cari ölçüsünü müəyyən edir (faylın elementlərini sıfırdan saymaqla).
- **Seek** göstəricinin cari mövqeyini göstərilən mövqeydə yerləşdirir.

■ **Truncate-** Faylın ölçüsünü göstəricinin cari mövqeyinə qədər azaldır. Cari mövqeydən sonrakı fayl elementləri silinir və göstəricinin cari mövqeyi son mövqey ilə eyniləşir (Eof(f)True).

Proqram nümunəsi.

Yuxarıdakı Massiv proqramı ilə A: diskində yaradılmış 15 ele-mentdən ibarət Dan. dat faylının 4-cü, 8-ci və sonuncu elementlərini birbaşa müraciətlə oxumalı və çap etməli. 8-ci elementi oxumamışdan əvvəl və sonra FilePos(f) funksiyası ilə kursorun fayldakı mövqeyini müəyyənləşdirib, çap etməli. **program** MasRead;

```
    Type
     TF= file of real;
 var
       f: TF;
        MS: real;
        Fp,Sz:  integer;
begin
   Assign(f, 'A:\Dan.dat');  Reset(f) ;
Seek(f,3);  Read(f,MS); Writeln(MS:8:1)  ;
         Seek (f, 7) ;
```

```
Fp:=FilePos (f) ; Write ( ' Fp_n= ' , Fp: 2) ;
Read(f,MS); Writeln(MS:8:1) ;
Fp:=FilePos(f);  Write('Fp_k=',Fp:2);
Read(f,MS);  Writeln(MS:8:1) ;
Sz:=FileSize(f);  Seek(f,Sz-l) ;
Read(f,MS);  Writeln(MS:8:1);
Close (f) ;
end.
```

Nəticə:

```
4.3
Fp_n=7      671.2
Fp_k=8       9.0
98.1
```

6. Dinamik verilənlər

6.1. Göstərici tip və dinamik dəyişən

Fayl tip fayllarla- xarici yaddaşla (disklərlə) işləmək üçün nəzərdə tutulmuş tipdirsə, göstərici tip daxili yaddaş (əməli yaddaş) ilə işləmək üçün nəzərdə tutulmuşdur. Ümumiyyətlə hər bir dəyişənin ünvanı və məzmunu (qiyməti) olur. İndiyə kimi istifadə etdiyimiz dəyişənlərin tipləri yalnız dəyişənlərin məzmunları ilə işləməyə imkan verirdi. Belə dəyişənlərin ünvanları proqramın kompilyasiyası zamanı sistem tərəfindən avtomatik olaraq müəyyənləşirdi. Proqramçı bu ünvanlara müdaxilə edib dəyişdirə bilmirdi. Ona görədə belə dəyişənlərə statik dəyişənlər də deyilirdi.

Dəyişənlərin ünvanları ilə işləyə bilmək üçün göstərici tip adlanan xüsusi tip və bir sıra prosedurlar mövcuddur. Bu tip proqramçıya imkan verir ki, proqramın icrası zamanı dəyişənin ünvanına müdaxilə olunsun, yəni dəyişən üçün operativ yaddaşda yer ayrılsın, bu yer lazım olmadıqda ləğv olunsun, həmin ünvana başqa dəyişən göndərilsin, dəyişənlərin strukturu

dəyişdirilsin.və s. Belə dəyişənlər üçün yaddaşda yer sonradan, proqramın icrası zamanı ayrıldığı üçün, onlara dinamik dəyişənlər deyilir. Dinamik dəyişənlər operativ yaddaşdan daha səmərəli istifadə etməyə imkan verir.

Ümumiyyətlə dinamik dəyişənlərə- dinamik yaddaşa müraciət aşağıdakı hallarda zəruridir:

- Eyni vaxtda emal olunmayan böyük massivlərdən istifadə etdikdə.
- Massiv və yazılar üçün 64Kbayt-dan çox yaddaş tələb olunduqda;
- Verilənlər üçün tələb olunan yaddaşı proqramın icrası zamanı ayırdıqda;
- Dinamik strukturlu verilənlərdən istifadə etdikdə.

1. Göstərici tip.

Göstərici tiplər tipləşdirilmiş və tipləşdirilməmiş tiplər olmaqla iki qrupa bölünür. Tipləşdirilmiş göstəriciləri elan edərkən tipin qarşısına "^" işarəsi əlavə olunur :

Göstərici_Tipin_adı= ^ Tipın_adı;

Tipləşdirilməmiş göstərici "Pointer" identifikatoru vasitəsilə yaradılır . Məsələn,

```
type
   P1=^ integer ;
   P2= array[1..10] of^ rea1;
   A1=array[1..5] of char;
    St= record X, Y: real  end;
    Pa1=^ A1 ;
   Ps1=^ St ;
    Pp1=^ P1 ;
   Tp= pointer;
```

Strukturu və tipi proqramın gedişi zamanı dəyişən verilənlərlə işləyərkən əsasən tipləşdirilməmiş göstəricidən istifadə olunur.

2. Göstərici tip dəyişən.

Göstərici tip dəyişənə sadəcə olaraq Göstərici də deyilir. (göstərici tip dəyişən bütün dəyişənlər kimi, iki üsulla əvvəlcədən təyin olunmuş göstərici tiplər vasitəilə, ya da bir başa tipin qarşısına " ^ " işarəsini əlavə etməklə təsvir olunur. Məsələn,

```
Var
    Vp1: ^ Integer;
    Vp2: ^ Real;
    PP: Pointer;
   Va1: ^ A1 ;
   P: Tp;
    Va2: Pa1;
```

Aşağıdakı yazılışlar səhvdir.

```
Type
    PV^string [5] ;
     P8=^ array[1..10] of real ;
     P9=^ Record X, Y : real ; end;
```

```
Var
    Vpl0: ^string[5];
    Vpl: ^array[1..10] of real;
```

MS-DOS əməliyyat sistemində Turbo-Pascalda (Delphi-də yox) operativ yaddaş hər birinin tutumu 64Kbayt olan seqmentlərə bölünür. Ona görədə Turbo Pascalda hər bir obyektin operativ yaddaşdakı ünvanı iki parametrlə müəyyən olunur:

- Daxil olduğu seqmentin nömrəsi
- Seqmentin başlangıcından meyl etmə-sürüşmə.

Bu parametrlər-ünvan təsgiledıcıları Seg(X):Word və Ofs(X):Word funksiyaları vasitəsil müəyyənləşdirilir. Məsələn,

```
Writeln('SegX=', Seg(X));
Writeln('OfsX=', Ofs(X));
```

operatorları X obyektinin seqmentini və sürüşməsini çap edəcəkdir.

```
Ptr(Seg, Ofs:word):Pointer
```

funksiyası isə Seg-seqmentini və Ofs-sürüşməsini ünvana göstəriciyə çevirir.

Obyekt ünvanının Seqmentini və Sürüşməsini təyin etmədən birbaşa ünvanı -göstəricini tapmaq olar. Bunun üçün "@" əməlindən , və ya Addr prosedurundan istifadə olunur. Müəyyənləşdirilmiş ünvan göstərici tipli dəyişənə mənimsənilə bilər. Məsələn,

```
Vpl:=@(P)
```

Burada Vpl- integer tipli göstərici olduğu üçün P operantı da həmin tipli olmalıdır.

Bu əməliyyat

```
Vpl:=Addr(P)
```

operatoruna ekvivalentdir.

3. Dinamik dəyişən.

Yuxarıda deyildiyi kimi, dinamik dəyişənlər elə dəyişənlərə deyilir ki, onlar üçün Operativ yaddaşda yer proqramın icrası zamanı ayrılır. Bu əməliyyat tipləşdirilmiş p göstərici üçün

```
new(p);
```

proseduru ilə yerinə yetirilir. Məsələn tutaq ki, p göstəricisi real tipli göstəricidir, yəni:

```
var
  p:^real;
```

Onda, **new**(p) proseduru nəticəsində real tipli ^p dəyişəni yaranacaqdır və bu dəyişənlə real tipli adi dəyişənlər kimi işləmək olar.

Operativ yaddaşda ayrılmış yer

```
dispose(Göstərici)
```

proseduru ilə ləğv olunur. Lakin p dəyişəni (ünvanları) yerlənlər seqmentində yerləşdiyi üçün öz qiymətini saxlayır. Tipləsdirilməmiş göstəricilər üçün bu əməliyyatlar **GetMem**(P, Size) ; **FreMem**(P, Size) ; Prosedurları ilə yerinə yetirilir.

Burada,P-göstərici,Size-baytlarla ayrılan yerin ölçüsü.

Dinamik dəyişənin öz adı olmur. Ona da göstərici vasitəsilə müraciət olunur. Bu məqsədlə Göstəricinin adının sağına "^" işarəsi əlavə olunmalıdır. Yəni dinamik dəyişənin adı "Göstərici^" şəklində yazılmalıdır. Proqramlarda

dinamik və adi dəyişənlərdən istifadə müqayisəli şəkildə aşağıdakı cədvəldə verilmişdir.

Verilənlər	Adi dəyişənlər	Dinamik dəyişənlər
1.Sadə dəyişənlər	Type Tine=1..100; Var X: char ; Y: Tine; Begin X:=' * ' ; Y:= 3; . . . end.	Type Tine=1..100; Var PX: ^char ; PY: ^Tine; Begin New(PX);New(PY); PX^:=' * ' ; PY^:= 3; . . . Dispose(PX); Dispose(PY); End.

2. Massiv	Type
	Vector=
	rray[1..100]
	Of byte;
	Var X: Vector ;
	i: Byte;
	Begin
	For i:= to 3 do
	Read(X[i]);
	. . .
	end.
		Type
		Vector=array[1..100]
		of byte;
		Var PX: ^Vector ;
		I: Byte;
		Begin New(PX);
		For i:= to 3 do
		Read(PX^[i]);
		. . .
		Dispose(PX);
		end.

Yazı	Type Rec= record A: char; B: Byte; End; Var X: Rec;	Type Rec= record A: char; B: Byte; End; Var PX: ^Rec;

115

	Begin X. A:= ' * ' ; X.B: = 3; . . . . end	Begin New (PX) ; PX^.A:= ' * ' ; PX^.B: = 3; . . . Dispose (PX) ; end

Burada,
^char ; ^Vector- göstərici tip
PX, PY - göstərici (göstərici tip dəyişən)
PX^, PY^ - dinamik dəyişəndir.

Proqram nümunəsi: İki tam ədədin cəmlənməsi.

```
program Din_dayisan;
var p1,p2,p3:^integer;
begin
```

```
{ pl,p2,p3 göstariciləri mövcuddur
və onların ilkin giymətləri- nil -sabitidir}
new (p1) ;      //integer tipli p1^
new (p2) ;      //p2^ ,p3^ dəyisanlarini
new(p3);        //   yaradirig
readln (pl^,p2^);
p3^:=pl^+p2^;
Writeln('cam= ',p3^)  ;
readln;
end.
```

6.2. Rabitəli strukturlu dinamik verilənlər

İndiyə kimi baxdığımız dinamik verilənlər bir-biri ilə əlaqəsi olmayan ayrı-ayrı verilənlər idi. Onlara ***rabitəsiz strukturlu*** dinamik verilənlər deyilir. Bir-birilə rabitələnmiş daha mürəkkəb strukturlu verilənlər də tərtib etmək olar. ***Rabitəli strukturlu*** adlanan belə verilənlərdə hər bir element iki hissədən ibarət "yazı" kimi tərtib olunur. Birinci hissə - elementin məlumat hissəsidir, digər hissə isə növbəti və ya əvvəlki elementlə rabitə yaradan ünvan- göstərici olur. Məsələn,

```
Type
   P_talaba= ^Talaba ;
   Talaba= record
      Name: string[20];
      Next: P_talaba ;
      End;
```

Rabitəli strukturlu dinamik verilənlərdə asanlıqla iki verilən arasında üçüncü verilənı əlavə etmək, verilənı silmək, çeşidləmək kimi əməliyyatları yerinə yetirmək olur.

Proqram nümunəsi.

Rabitəli dinamik verilənlərdən istifadə olunmuş aşağıdakı proqrama baxaq. Bu proqram tələbələrin siyahısını tərtib edir. Sonra çapa çıxardır. Həmçinin çeşidləyir. Lazım olmayan elementi ləğv edir- silir. Bu proqramdakı bir sıra prosedurlardan istifadə olunmuşdur:

- Procedure Add; - Siyahının əvvəlinə yeni element əlavə edir.
- procedure Cap_S;- Tərtib olunmuş siyahını çap edir- ekrana çıxardır.
- procedure S_Eim_Sil;-Siyahıdan elementi-klaviaturadan daxil edilmiş familyanı silir.
- procedure Sort;- Siyahını çeşidləyir.

```
Program dinlist;
type
  p_talaba=^talaba;
  talaba=record
            ad:string;
            next:p_talaba;
            end;
   var
    head:p_talaba;   {siyahinin baslangici}
```

```
        p,pl,node,curr:p_talaba;{cari element}
         buf,bufl:  string[20];   {klaviaturadan daxil etmak ucun
bufer}
              Procedure Add;
          begin
          repeat
          write('Familya->');  readln(buf);
          if length(buf)<>0 then
          begin
               new(curr);
                curr^.ad:=buf;
                curr^.next:=head;
               head:=curr;
         end;
 until length(buf)=0;        end;
procedure Cap_S;
            begin
               curr:=head;
              while curr<>nil do begin
              write(curr^.ad,'  ');
               curr:=curr^.next;
              end;
               writeln;
              end;
       procedure S_Eim_Sil;
       {Siyahidan elementin xaric olunmasi}
       function DelNode:boolean;
    var
       p:p_talaba;
      done:boolean;
      begin
      p:=nil;
      curr:=head;
       done:=False;
     while  (done=False)  and  (curr<>nil)  do begin
      if curr^.ad=buf
     then begin
     if p=nil
       then head:=curr^.next
       else p^.next:=curr^.next;
       done: ==True;
       end
       else begin
       p:=curr;
        curr:=curr^.next;
         end;
        end;
           DelNode:=done;
                       end;
                       begin
            repeat
              write('Silinacak familya->');
              readln(Buf);
```

```
            if length(buf)<>0 then
            if DelNode
            then
writeln ('"', buf, '" ' ,' siyahidan silindi ' )
    else writeln('"',buf, '"', 'siyahida yoxdur') ;
    until length(buf)=0;
    end;
procedure Sort_S;
                {_ Cesidlama    }
                var   k,i,m:  integer;
                procedure Sort;
                begin
              k:=0;
          curr:=head;
          while  (curronil)  and  (curr^ . next<>nil)  do
           begin
                p:=curr;  k:=k+l;
                curr:=curr^.next;
                if   p^.ad>curr^.ad then
            begin
      buf:=p^.ad;  p^.ad:=curr^.ad;  curr^.ad:=buf;
      end;
          end;
            end;
    begin Sort;m:=k+l;
    writeln('siyahidaki elementlarin sayi=',m);
     for i:=l to m do sort;
    end;
        begin {Asas program }
        {**    Siyahinin yaradilmasi **}
        Add;
      writeln('** Siyahi **');
        Cap_S;
     writeln('** Cesidlamadan sonraki Siyahi'**');
        Sort_s;   Cap_S;
        {Siyahidan elementin silinmasi}
          S_Elm_Sil;
          writeln('Elementlar silindikdan sonraki siyahi');
          Cap_S;   readln;   end.
```

6.6 Verilənlərin strukturları

Proqramlaşdırmada istifadə olunan verilənlər iki böyük qrupa bölünür:

- Statik strukturlu verilənlər;
- Dinamik strukturlu verilənlər.

Elementlərinin qarşılıqlı yerləşməsi və qarşılıqlı əlaqəsi sabit qalan verilənlər **statik strukturlu verilənlər** adlanır.

Elementlərin sayı, onların qarşılıqlı yerləşməsi və qarşılıqlı əlaqəsi proqramın yerinə yetirilməsində müəyyən qayda üzrə dinamik dəyişən verilənlər **dinamik strukturlu verilənlər** adlanır.

Statik strukturlu verilənlər. Statik strukturlu verilənlər hər hansı qayda üzrə sadə strukturlardan təşkil olunmaqla **sadə (skalyar)** və mürəkkəb **(aqreqativ)** ola bilər.

Proqramlaşdırma dillərində **sadə** verilənlərə verilənlərin standart (əvvəldən təyin olunmuş) tipləri uyğundur. Bura hesabi (natural, tam, həqiqi, kompleks), simvol, məntiqi və göstərici tipləri aiddir. Turbo Pascal-a Byte, Word natural tiplər, İnteger, Shortint, Longint tam tiplər, Real, Single, Double, Extended, Comp həqiqi tiplər, Boolean , Byte Bool, Word Bool, Long Bool məntiqi tiplər, Char simvol tipi və Pointer göstərici tipi daxil edilib. Həqiqi ədədlər sabit və sürüşən nöqtəli təsvir oluna bilər. Kompleks ədədlərin təsviri üçün isə Turbo Pascalda standart tiplər yoxdur.

Bunlardan başqa, bəzi dillərdə olduğu kimi, Turbo Pascal-da da istifadəçi tərəfindən təyin olunan sadalanan və interval tiplərindən də istifadə etmək mümkündür.

Mürəkkəb strukturlu verilənlərə bircins, yəni bütün elementləri eyni tipdən olanlar, və **qeyri-bircins** (kombinə edilmiş), yəni müxtəlif tip elementlərin birləşməsindən təşkil olunmuş verilənlər aiddir. Bircins strukturlu verilənlərə isə sadə yazılar, variantlı yazılar, birləşmələr və obyektrlər aiddir.

Dinamik strukturlu verilənlər. Dinamik strukturlu verilənlərə fayllar, əlaqəsiz və əlaqəli dinamik verilənlər aiddir.

Fayylar mətn, tipləşdirilmiş olur.

Əlaqəsiz dinamik verilənlər statik stukturlu verilənlərə analoji olaraq təsnif olunur. Əlaqəli dinamik verilənlər isə xətti, dairəvi və budaqlanan struktura malikdir.

Dinamik strukturlu verilənlər haqqında geniş məlumat dərsliyin 6.7.6-cı bölməsində veriləcək.

6.7 . Statik strukturlu verilənlərlə iş

6.7.1. Sadə və sətir tipli dəyişənlərin qiymətlərinin daxiledilməsi və xaricedilməsi

Turbo Pascal dilində standart daxiletmə Read. Readln və standart xaricetmə Write, Writeln prosedurları vasitəsi ilə həyata keçirilir.

Bu prosedurların yazılış forması aşağıdakı kimidir:

Read (fayl dəyişəninin adı, dəyişənlərin siyahısı);

Readln (fayl dəyişəninin adı, dəyişənlərin siyahısı);

Write (fayl dəyişənin adı, xaricedilən elementlərin siyahısı);

Writeln (fayl dəyişənin adı, xaricedilən elementlərin siyahısı);

Standart daxiletmə əvvəldən təyin olunmuş, klaviatura ilə əlaqəli İnput adlı mətn faylından yerinə yetrilir. Standart xaricetmə isə əvvəlcədən təyin olunmuş, displey ilə əlaqəli Output adlı mətn faylında yerinə yetirilir. Susmaya görə daxiletmə üçün İnput, xaricetmə üçün isə Output götürülür. Bunları nəzərə alsaq aşağıdakı proqram fraqmenrləri ekvivalentdir.

```
Readln (İnput, A,B);
Writeln (Output, 'A=', A, 'B'=', B;
```

və

```
Readln (A,B);
Writeln ('A=', 'B=', B);
```

Standart daxiletmə və xaricetmə prosedularından istifadə edərkən aşağıdakılar nəzərə alınmalıdır:

- Read və Readln prosedurları ilə yalnız tam, həqiqi, simvol və sətir tipli verilənlər oxunur;
- Write və Writeln prosedurları ilə yalnız tam, həqiqi, simvol , sətir və tipli verilənlər yazılır.

Daxiletmə. Daxiletmədə Read prosedurundan fərqli olaraq Readln proseduru verilənlərin növbəti sətrin başlanğıcından oxunmasını həyata keçirir. Parametrsiz Readln-dən istifadə etdikdə, verilənlərin oxunması növbəti sətrin başlanğıcından başlayır.

Daxiletmə prosedurları fayldan yalnız bir simvol oxuyaraq dəyişənə mənsub edirlər. Ədədlərin oxunması isə birinci boşluğa, tabulyasiya simvoluna, sətrin sonu işarəsinə və yataq faylın sonu işarəsinə kimi həyata keçirilir.

Read proseduru sətir dəyişəni neçə simvol elan olunubsa, o qeder simvol oxuyur. Bu zaman boşluq ayırıcı rolunu oynamır. Bu prosedur yeni sətrə keçidi yerinə yetirmir, bu əməliyyatı Readln proseduru yerinə yetirir.

Misal:

```
var A: array [0..5] of Char;
    S1, S2,S3: string [10];
begin
```

```
Read (A);
Read (Sl);
Read (S2);
Readln;
    Read (S3);
end.
```

Xaricetmə. Writeln prosedurunun hər bir elementi aşağıdakı kimidir:

Expr [:M [:D]]

burada:

Expr-tam, həqiqi, simvol, sətir və Bul tipindən olan xaricediləsi ifadə:

M- xaricedilmə sahəsinin uzunluğunu göstərən sıfırdan böyük tam tipli ifadə;

D-xaricedilənin onluq işarədəki rəqəmlərinin sayını göstərən tam tipli ifadədir.

D-yalnız Expr həqiqi tipdə və M parametri verildikdə göstərilir. Əgər D göstərilibsə, ədəd sabit nöqtəli formatda, göstərilməyibsə sürüşən nöqtəli formatda xaric olunur.

Sabit nöqtəli xaricedilmənin formatı:

[<boşluq> [[-] <rəqəm> [.kəsr hissənin rəqəmləri]

Sürüşən nöqtəli xaricedilmənin formatı:

[-] <rəqəm> [.kəsr hissənin rəqəmləri] E [+/-< tərtibin dərəcəsi>]

Writeln proseduru mətn faylları üçün Write prosedurunun genişlənmiş variantıdır. Parametrsiz Writeln proseduru faylın sonuna yalnız sətrin sonu işarəsini yazır.

Misal:

```
proqram misal_8;
uses Crt;
const
    i: İnteger =12345:
    r: Real= -123.1234567;
    c: Char='$';
     b: Boolean=True;
      s: string= 'KT və proqramlaşdırma';
   begin
        ClrScr;
          Writeln ('Formatsız çap');
          Writeln (i,r,c,b,s);
          Writeln;
          Writeln ('Formatlı çap);
          Writeln (i:10,r:10:3, c:10,b:10,s:20):
           Writeln;
           Writeln ('Həqiqi ədədlərin  sabit formatda çapı');
           Writeln (r:3:0);
```

```
        Writeln (r:6:3);
        Writeln (r:13:7);
        Writeln (r:25:9);
        Writeln;
        Writeln ('Həqiqi ədədlərin sürüşən formatda çapı');
        Writeln (r:3);
        Writeln (r:6);
        Writeln (r:13);
        Writeln (r:15);
    end.
```

6.7.2. İstifadəçinin sadə tipləri

Sadə istifadəçi tiplərə sadalanan və interval tipləri aiddir.

Sadalanan tip. Sadalama proqramçıya verilənlərin yeni tiplərini təsvir etməyə imkan verir. Sadalanan tiplərin təsviri dairəvi mö'tərizə daxilində elementlərin siyahısından təşkil olunur:

```
type
    Fasil= (Yaz, Yay, Payız, Qısh);
```

Sadalanan tipin bütün elementlərinin identifikatorları sabit kimi interpretasiya olunur. Bu identifikatorlar sətir sabitləri olmadığından dırnağa alınmır. Qeyd etmək lazımdır ki, eyni identifikatorun müxtəlif tiplərdə təsviri səhvdir. Məsələn:

```
    proqram  Duplicate;
```

```
Hafta_1= (Mon, Tue, Wed, Thu, Fri, Sat, Sun);
Hafta_2= ((Mon, Tue, Wed, Thu, Fri);
begin
. . .
end.
```

Bu proqramın translyasiyasında “Error: Duplicate identifier (Mon). “ məlumatı alınır.

Proqramda sadalanan tip dəyişəndən istifadə etdikdə aşağıdakıları nəzərə almaq lazımdır:

- sadalanan tiplər sıra tipinə aiddir;
- Read, Readln, Write və Writeln prosedurlarında arqument kimi sadalanan tipin qiymətlərindən istifadə etmək olmaz;
- sadalanan tiplərdə yalnız nisbət əməliyyatlarından istifadə etmək olar:
- sadalanan tip dəyişənlərdən mənsubetmə operatorlarında, massivin indekslərində və for operatorunun sərhədlərində istifadə etməyə icazə verilir.

İnterval tipi. İnterval tipi hər hansı sıranın qiymətlər diapazonundan (interval) təşkil olunur. İnterval tipinin təsvirində , bir-birindən iki nöqtə (..) ilə ayrılan qiymətlər diapazonun ən kiçik və ən böyük qiymətləri göstərilir.

Məsələn:

```
0..600
-1..1
-128..127
'A' .. 'Z'
   YAZ . . QISH
```

```
tupe t_rec=record
      A:record
         B:record
           X:char;
           Y:byte;
            end;
         C:real;
         end;
       D:string;
      end;
   var rec:t_rec;
```

Burada Y sahəsinə 0, C sahəsinə isə 3.14159 qiymətlərinin mənsubedilməsi tələb olunursa, bu aşağıdakı kimi yazılmalıdır:

```
rec.A.B.Y:=0;
rec.A.C:=3.14159;
```

Sadə halda with birləşdirmə operatoru yazı sahələrinin adını ağaıdakı kimi ixtisar etməyə imkan verir:

```
with rec do
begin
```

```
        A.B.Y:=0;
        A.C:=3.14159;
    end;
```

Əgər bir yox, iki with operatorundan istifadə olunarsa, onda o aşağıdakı kimi yazılır:

```
    with rec do
      with A do
      begin
        B:Y:=0;
        C:=3.14159;
    end;
```

Burada rec və A –nın adını bir siyahıda göstərməklə with operatorunun köməyi ilə yuxarıdakı proqram fraqmentini daha yığcam yazmaq olar.

```
    with rec A do
      begin
        B:Y:=0;
        C:=3.14159;
    end;
```

Yalnız X və Y sahələrinə müraciət tələb olunursa , onda bu ağaşıdakı kimi yazmaq olar.

```
with rec, A,B do
  begin
    X:='*'
          Y:=0;
```

və continue prosedurlarından istifadə etmək olar.break proseduru çıxış şərtinin yerinə yetirilməsini gözləmədən dövrdən çıxmağa imkan verir.

continue proseduru isə dövrün əvvəlki iterasiyası sona çatmadan yeni iterasiyanın başlanmasına imkan verir. Prosedurları aşağıdakı misalda göstərək.

Misal: Tam ədədlərdən təşkil olunmuş massivin birinci mənfi elementinin axtarışı proqramı.

```
proqram misal_7;
const n = 15;
yes:boolean =false;
var
   mas:array [1..n]  of integer:
   i:byte;
begin
    writeln ('Massivin elementlərini daxil edin’);
     for i:=1 to n do
        begin
       write (‘mas[‘.i,’]=’);
       readln (mas [i’]);
          end;
      for i:=l to n do
```

```
            begin
              if mas [i]>=0 then continue:
              writeln (‘Birinci mənfi element =’, mas [i],
‘Nömrə=’, i);
            yes:=true;
            break;
              end;
        if not yes then writeln (‘Mənfi element yoxdur’);
        readl;
      end.
```