Bir daha qeyd edək ki, hər bir ünsür yalnız mənsub olduğu tipə aid olan diapazonda qiymətlər ala bilər.

Proqram yazılışında həqiqi ədədin tam və kəsr hissələrini ayırmaq üçün qeyd olunmuş nöqtə -məs. 21.014; -7.601 kimi; və ya sürüşgən nöqtə –məs. 60.1E-1; -0. 013E+2 kimi; -biçimlərdən istifadə olunur. Burada aE+n=a*10^n və aE-n=a*10^{-n} kimi başa düşülür.

Simvol tipi (char) – Komputerin ASCII - kodlar cədvəlinin simvollar çoxluğudur. Burada, məs. dırnaq arasına alınmış 'a'; '7'; '#' və s. kimi hər-hansı işarə bir simvol sabiti sayılır.

Məntiqi tip **-boolean**, yalnız iki qiymətlə xarakterizə olunur; **false** (yalan) və **true** (doğru).

Həqiqi ədədlərdən başqa qalan tiplərin elementləri ardıcıl sadalanma prinsipi əsasında nizamlanmışdır.

2. *Qeyri standart sadə tiplər* –artıq yuxarıda göstərilən standart tiplərdən fərqli olub **type** bölümündə sadalanır.

```
type    <1-ci tipin adı> = <1 -ci tipin şərhi>
        ……
        <n –ci tipin adı> = <n –ci tipin şərhi>
```

Qeyri standart tiplı ünsürlər əsasən -sadalanan, diapazonlu, sətir tiplı və s. formalarda ola bilir.

Qeyri standart sadalanan tiplər -adlar və hər adın mümkün qiymətlərini mötərizədə, aşağıdakı kimi sadalamaqla verilir:

```
type    <tipin adı>=(<1-ci qiymət>,<2 –ci qiymət>,…,<n –ci qiymət>);
```

Qeyd edək ki, bu cür dəyişənlərə ədədi və ya simvol tiplı qiymətlər verilə bilməz.

Sadalanan tipə aid bir neçə misal göstərək:

```
type    season = (spring, summer, autumn, winter);
        color = (blue, green, red, yellow);
        week = (mon, tue, wed, the, fri, sat, sun);
```

Daha sonra bu tiplərlə işləmək üçün müvafiq dəyişənlər təyin edək:

```
var     day: week;
        boya: color;
        il: season;
```

Təbiidir ki, təyin olunmuş bu dəyişənlərə aşağıdakı kimi qiymətlər verilə bilər.

```
day:=fri; boya:=green; boya:=red; il:=winter; il:=summer;
```

Burada elementlərin nömrəsi 0 –dan başlayır və onlar arasında ord, pred və succ funksiyaları (cədvəl 4.) təyin olunur. Məsələn ord(boya)=0, ord(day)=4, pred(il)=spring, succ(il)=3 və s.

Qeyd etmək lazımdır kì, bu tìp ünsürlərì klavìaturadan daxìl etmək və ya ekrana çıxarmaq olmaz.

Dìapazonlu tìp –hər-hansı nìzamlı baza tìpìndən lazımı hìssələrìn seçìlməsì ìlə düzəlìr və proqramda aşaõıdakı formada təqdìm olunur.

```
type    <tìpìn adı>=<1-cì qìymət>,<2 –cì qìymət>,…,<n –cì qìymət>;
```

Mìsal. Həftənìn günlərìnì, ìş günlərìnì və ayların nömrəlwrìnì uyõun olaraq ìfadə edən weekdays, workdays, months tiplərini yaradaq və onlar üçün day1, day2, day3, və month dəyìşənlərìnì təyìn edək:

```
type    weekdays = mon, tue, wed, the, fri, sat, sun;
        workdays = mon, tue, wed, the, fri;
        months = 1..12;
var     day1, day2 : weekdays;
        day3 : workdays;
        1mont1h : months;
```

*Sətìr tìplì dəyìşən*lərìn qìymətì -ìstənìlən ardıcıllıqla düzülmüş, 255 –ì aşmayan sayda sìmvollar ardıcıllıõı ola bìlər. Tìplər bölümündə bu cür ünsürlərì ìstər sabìt və ìstərsə də dəyìşən kìmì təqdìm etmək mümkündür. Məs:

```
type p = string[8];
const q = 'tua$1';
var   pau : p;
      m : string;
```

-kvadrat mötərìzədə sətrìn uzunluõu (sìmvollar sayı) göstərìlìr.

Tìp kìmì verìlən sabìtlər –dəyìşənì təqdìm etməklə eynì vaxtda həm də onu qìymətləndìrməyə ìmkan verìr:

```
const   <konst 1>: <tip 1> = <dəy -n 1>;
                .....
                <konst n>: <tip n> = <dəy -n  n>;
```

Adì sabìtlərdən fərqlì olaraq bu cür təqdìm olunan sabìtlərì proqramda dəyìşmək mümkündür. belə sabìtlərə bìr – neçə mìsal göstərək:

```
const  u : char ='b';
        v : integer = 43:
        h : real = 21.051:
        t : boolean = true; və s…
```

§3. Sadə proqramlar.

Sadə proqramlar əsasən -təyìnat, ünsürlərìn gìrìş–çıxışı və proseduralara müracìət məqsədì daşıyan əmrlərdən ìbarət olur kì, bu cür proqramlara xəttì proqramlar da deyìlìr.

1. *Təyìnat əmrlərì* –aşaõıdakı formada yazılır:

<dəyişənìn adı> := <ìfadə>

Bu əmrlə -dəyişənìn qìymətì ìfadə əsasında təyìn edìlìr. İfadələr ìsə – məchul olmayan dəyìşən və funksìya, ədəd və sabìtlərdən ìbarət, proqram dìlìnə uyõun formada yazılmış ìstənìlən rìyazì ìfadə ola bìlər. «Konus» proqramında l, s və v dəyişənlərì də məhz bu cür ìfadələr əsasında təyìn edìlmìşdìr:

```
const r = 1.5; h = 6; {r, h –sabit kimi verilir}
{l, s, v -hesablanIr}
  l := sqrt(sqr(r) + sqr(h));
     s := pi*r*l;
  v := 1/3*pi*sqr(r)*h;
```

Əlbəttə təyìnat əmrlərìndə dəyìşən və ìfadələrìn tìplərì zìddìyyətsìz verìlməlìdìr. Məs. Həqìqì (və ya sətìr) tìplì dəyìşənə tam ədəd (və ya sìmvol) tìplì ìfadələrìn qìymətlərì verìlə bìlər lakìn tərsìnə yox.

2. Paskal dìlìndə yazılmış proqramların məqsədəuyõun fəalìyyətìnì tənzìm etmək üçün ìşlənən *əsas əməllər* (Cədvəl 3 –də), standart funksìyalar və proseduralar (Cədvəl 4 –də) sadalanır:

Cədvəl 3. Əsas əməllər.

İcra sırası	Əməl	Əməlìn təyìnatı
1	+, - not	ìşarələmə məntìqì ìnkar
2	*, / div, mod and	vurma, bölmə tam ədəd əməllərì məntìqì hasìl
3	+, - or, xor	toplama, çıxma məntìqì cəm
4	=, <>, <, >, <=, >= in	müqayìsə çoxluõa aìd olma

Əməllər -rìyazì ìfadələrìn hesablama qaydaları əsas götürülməklə prìorìtet əsasında (yüksəyì 1 olmaqla)yerìnə yetìrìlìr. Məs:

4 * -3 +6 = -6; 5 * (-4 -1) = -25; 4 * (-7 + 13) =24;

12 - 120 / 20 * 3 = -6; 12 - 120 / (20 * 3) = 10;

Tam ədədlər arasında dìv əməlìnìn nətìcəsì qìsmət, mod əməlìnìn nətìcəsì qalıq olur. Məs:

(29 - 11) **div 5** = 3; 13 **mod 5** = 3

Cədvəl 4.Standart funksìyalar və proseduralar.

Funksìya	Arqumentìn tìpì	Nətìcənìn Tìpì	Rìyazì yazılış
abs(x)	integer, real	ìnteger, real	|x|
arctan(x)	integer, real	real	arctg(x)
cos(x)	integer, real	real	cos(x)
sin(x)	integer, real	real	sin(x)
exp(x)	integer, real	real	e^x
ln(x)	integer, real	real	ln(x), x>0
sqrt(x)	integer, real	real	√x, x≥0
sqr(x)	integer, real	integer, real	x^2
ord(x)	nìzamlanmış	integer	ASCII –sìmvol kodları.
succ(x)	nìzamlanmış	nìzamlanmış	x –ìn növbətì qìymətìnì verìr
pred(x)	nìzamlanmış	nìzamlanmış	x –ìn əvvəlkì qìymətìnì verìr
round(x)	real	integer	x -ì tam ədədə yuvarlaqlaşdırır
trunc(x)	real	integer	x –ìn tam hìssəsì
int(x)	real	real	x –ìn tam hìssəsì
frac(x)	real	real	x–ìn kəsr hìssəsì
odd(x)	ìnteger	boolean	true (x –tək) falze (x –cüt)
random(x)	integer	integer	x –dən kìçìk təsadüfì müsbət-tam ədəd verìr
upcase(x)	char	char	Latın hərf-nì əlyaz-na çevìrìr
Proseduralar			
inc(x, y)	integer	integer	x-ì y qədər böy-r
inc(x)	integer, char	integer, char	x-ì 1 vahìd böy-r
dec(x, y)	integer	integer	x-ì y qədər az-r
dec(x)	integer, char	integer, char	x-ì 1 vahìd az-r

Funksìyaların qìymətlərì və proseduraların tətbìqìnə aìd bìr neçə mìsal göstərək:

round(2.1)=2, int(2.1)=2.0, x:=1; inc(x, 5); (x=6),
round(6.8)=7, int(6.8)=6.0, x:='a'; inc(x); (x='b'),
trunc(2.1)=2, frac(2.1)=0.1, x:=7; dec(x, 3); (x=4),
trunc(6.8)=6, frac(6.8)=0.8, x:='d'; dec(x); (x='c').

Əlbəttə, verìlmìş əsas funksìyalar dəstìndən ìbarət olan dìqər funksìyalar da almaq mümkündür. Məs:

$$Ctg(x) = Cos(x) \Big/ Sìn(x); \qquad arcCtg(x) = arcSìn\left(1 \Big/ \sqrt{1+x^2}\right)$$ və s.

3. Ünsürlə*rì dìaloqla tanıtma əmrlərì* –**read, readln** -aşaõıdakı formada yazılır

read(və ya readln)(<1 –cì dəyişən>, … , <n –cì dəyişən>);

və verìlənlərìn klavìaturadan daxìl edìlməsìnə ìmkan yaradır.

Məs. «Konus» proqramında

const r = 1.5; h = 6; –əvəzìnə

read(r, h) və ya **readln**(r, h)

yazılmış olsaydı, onda proqram arada öz işìnì dayandırar və kursor ekranda döyünməyə başlar, gìymətlərì ekrana yazıb keçìd(Enter) sədəfìnì basınca o, öz ìşìnì yenìdən davam etdìrərdì. Təbììdìr kì, bu hal daha unìversal xarakter daşıyaraq r və h –ın ìstənìlən qìymətlərì üçün tətbìq oluna bìlər.

read əmrìndən fərqlì olaraq **readln** ìlə verìlmìş məlumatın ardınca həmìn sətrə başqa məlumat yazıla bìlməz və bu əmr mətnlərlə ìşləməyə daha çox yarayır.

4. *Çıxarış əmrlərì* –**vrìte** və **vrìteln –** proqramın verdìyì cavab, nətìcə, və dìgər məlumatları ekrana çıxarmaq üçün ìşlənərək aşaõıdakı formada yazılır:

write(və ya **writeln**) (<1 cì ìfadə>,…,<n –cì ìfadə>);

Bu zaman məlumatlar ekranın çıxış xanəsìnə verìlìr kì, MS-DOS üçün TP mühìtìndə bu xanə məs. Alt +F5 sədəflərì ìlə seçìlə bìlər.

write -dən fərqlì olaraq **vrìteln** əmrìnìn verdìyì məlumatdan sonra gələn çıxarış ancaq növbətì sətìrdən başlayır. Odur kì, **vrìteln** əmrìnì parametrsìz ìşlətməklə ekranda sətìr buraxmaq və ya sətìrdən – sətrə keçmənì də təşkìl etmək mümkündür.

5. *Bìçìmlì (formatlı) çıxarış* -artıq Konus proqramında

writeln(‘S = ’, s:5:3);

writeln(‘V =’, v:5:3);

kìmì tətbìq olunmuşdur. Xatırladaq kì, bu formatda s və v dəyìşənlərì üçün qìymətlər –3 -ü vergüldən sonra olmaqla, cəmì 5 rəqəmlə ekrana verìlìr.

writeln(‘Proqramı yazdı Ü. Süleymanov’);

bìçìmlì çıxarış əmrì ìsə -dırnaq arasında verìlən sìmvollar ardıcıllıõını ekrana olduõu kìmì verməyì təmìn edìr.

Ìndì ìsə başqa bìr məsələnìn həllì üçün proqram tərtìb edək:

Məsələ 2. Mərkəzi $O(x_0,y_0)$ nöqtəsində olub $A(x_1,y_1)$ nöqtəsindən keçən çevrə uzunluğunu və bu çevrə ilə məhtudlanmış daìrənìn sahəsìnì tapmaq üçün proqram tərtìb etməlì.

Əvvəlkì proqramdan fərqlì olaraq, burada artıq bütün əmrlərə ìzah verìlməsìnə lüzum görmürük.

```
program Bicim;
uses Crt;
var x0, y0, x1, y1, r, r0, c, s: real;
begin
  clrscr;
  writeln('Koordinatlar verilsin: ');
  readln(x0, y0, x1, y1); {Koordinatlar ekrana verilir}
                                          {keçid…}
  r := sqrt(sqr(x1- x0) + sqr(y1 - y0));
  r0 := sqr(r);
  c := 2*pi*r;
  s := pi*r0;
  writeln('c = ', c:6:2);
  writeln('s = ', s); {s standart formatda verilir}
  readln;
end.
```

Bu proqram öz ìşìnì, ekranla aşağıdakı kìmì dìaloq şəraìtìndə aparır:

Ekrana əvvəl çıxır:

Koordinatlar verìlsìn:

Bìz yazırıq:

-1.8 2.1
3 0.5

və keçìd sədəfì basılır..

Ekrana sonda çıxır:

c = 31.79
s = 8.0424771932E+01

Göründüyü kìmì

writeln('s = ', s);

yazılışında s kəmìyyətìnìn qìymətì ekrana 11 rəqəmlì sürüşgən nöqtə ìlə verìlmìşdìr.

Çalışma 2. Təpə nöqtələrì A(a, x), B(b, y), C(c, z) olan üçbucağın perìmetrìnì və sahəsìnì hesablamaq üçün proqram yazmalı.

§4. Budaqlanma.

Budaqlanma –proqram fəalìyyətìnìn yığma əmrlərlə həyata keçìrìlən əməlìyyat bölümüdür:

1. *Yıõma əmrlər* - əmrlər dəstìndən aşaõıdakı formada yıõılmış blokdan ìbarət olur:

```
beqìn
        <1 –cì əmr>;
        …….
        <n –cì əmr>;
end;
```

Yadda saxlamaq lazımdır kì, hər bu cür bloku kompìlyator bìr əmr kìmì qəbul edìr.

Onu da deyək kì, **beqìn end;** və ya **;;** blokları boş əmrlər sayılır.

Bununla da Paskal dìlìndə ìşlənən əmrlər –boş əmr, sadə əmr və yıõma əmr kìmì qruplara bölünmüş olur.

2. Paskal dìlì dìõər məsələlərlə yanaşı *məntìqì məsələlərìn həllì* üçün də proqram tərtìb etməyə ìmkan verìr. Təbììdìr kì, bu cür proqramların başlıca ünsürü məntìqì ìfadədìr:

Məntìqì ìfadə –bəzì zərurì bìlgìlərì əldə etmək üçün lazım olan şərtlərìn proqram dìlìndə, qanunauyõun yazılış üsuludur. Xatırladaq kì, -məntìqì ìfadələr yalnız ìkì qìymət ala bìlər -true(doõru) və false(yalan). Məntìqì ìfadələr də ìkì qrupa ayrılır:

1) *Sadə ìfadə* -arasında münasìbət ìşarəsì qoyulmuş bìr cüt hesabì ìfadədìr.

2) *Yıõma ìfadə* –**not, and** və ya **or** –məntìq əməllərì ìlə bìrləşmìş sadə ìfadələr dəstìdìr. Bu əməllərìn fəallıq prìorìtetì **not – and – or** ardıcıllıõına uyõundur. Əməllər aşaõıdakı kìmì təyìn olunmuşdur:

Ìfadə	**Dəyərì**	**Ìfadə**	**Dəyərì**
not true	false	**not** false	true
true **and** true	true	true **or** true	true
true **and** false	false	true **or** false	true
false **and** true	false	false **or** true	true
false **and** false	false	false **or** false	false

Məs. a= -17 və b= 3 qìymətlərì üçün aşaõıdakı məntìgì ìfadələrì yaza bìlərìk:

Sadə ìfadə	*Qìymətì*	*Mürəkkəb ìfadə*	*Qìymətì*
a < 0	true	not(b > 4)	true
b > a	true	(a > -1) or (b > 5)	false
a div 5 = 3	false	(5 div b=1) and (a=3)	false
a mod b = 1	true	(a * b = -51) or (1>0)	true

Qeyd edək kì, mürəkkəb ìfadənì təşkìl eləyən hər bìr sadə ìfadə adì mötərìzədə yazılır. Məntìqì ìfadələrdə a<=b<c kimi ìkìqat bərabərsìzlìk

(a<=b) and (b < c) kimi, a < b; b >c isə (a < b) or (b > c) formada ifadə edilir.

2. *Budaqlanma əmri* –**if** –bütöv və qısa formada işlədilə bilir.

Bütöv forma:

```
if <məntiqi ifadə>     then <1 –ci əmr>
                       else <2 –ci əmr>;
```

Bu əmrdə –məntiqi ifadə doğru olarsa 1 –ci, əks halda isə 2 –ci əmr yerinə yetirilir. Həmin əmrlərdən hər biri isə, əlbəttə –sadə və ya yığma əmrlər ola bilər.

Məs(sadə hal). Əgər a = 16 olarsa, onda

if a < 1 **then** x := sqr(a) **else** x := a div7;

if a >=16 **then** y : exp(a) **else** y :=-4;

əmrlərindən sonra x və y dəyişənləri uyğun olaraq 2 və e^{16} qiymətlərini alar.

Başqa bir misal(mürəkkəb hal). Əgər p = -3.5 olarsa, onda

```
if p < -4 then
        begin
        q := p – 2.5;
        g := 1 + 2 * p;
        end
else
        begin
        q := 0.5 + 3 * p;
        g := 2 – 3*(p+2.5);
        end;
```

əmrlərindən sonra q və g kəmiyyətləri uyğun olaraq -10 və 5 qiymətləri alar.

İndi isə **if** əmrinin ikiqat (və ya iç - içə) tətbiq üsulunu göstərmək üçün növbəti məsələyə baxaq:

Məsələ 3. x arqumentinin dialoqla verilmiş qiymətinə görə

$$y = \begin{cases} \ln|x|, & x < -1 \\ \sin(x), & -1 \le x \le +1 \\ \cos(x), & x \ge 1 \end{cases}$$

funksiyasının qiymətini hesablayaraq ekrana çıxaran proqram tərtib etməli.

```
Program ikiqat_if;
uses Crt;
var x, y: real;
begin
        clrscr;
```

```
        write('x-i daxil edin: ');
        readln(x);
        if x<-1 then
                y:=ln(abs(x))
        else
                if (x>=-1)and(x<1) then
                        y:=sin(x)
                else
                        y:=cos(x);
        writeln('x = ', x:5:2, ' y = ', y:5:2);
        readln;
end.
```

! -yadda saxlamaq lazımdır kì, **if** əmrì altında olan else xìdmətì sözündən əvvəl ; işarəsì qoymuq olmaz.

Çalışma 3. x arqumentìnə ekrandan verìlən qìymətə görə

$$y = |x - 1| - |x - 3|$$

funksìyasının müvafìq qìymətìnì ekrana çıxaran proqram yazın.

if *əmrìnìn* qısa forması ìsə aşağıdakı kìmì yazılır:

if <məntìqì ìfadə> **then** <əmr>;

Bu əmrdə məntìqì ìfadə doğru ìsə qeyd olunmuş əmr ìşə başlayır, əks halda ìsə ìdarə növbətì (; sìmvolundan sonrakı) əmrə verìlìr.

Məs. a =4.5 olarsa, onda

if a >5 **then** x := sqr(a);

if a< 5 **then** x := sqrt(2*a);

əmrlərìndən sonra x =3 qìymətìnì alar.

3. *Şərtsìz keçìd əmrì* –**goto** əmrlərìn növbəsìnì, lazımı nìşanlar üzrə dəyìşdìrmək üçün ìşlədìlìr. Bu əmr aşağıdakı formada yazılır:

goto <nìşan>;

Nìşan proqramda ìstənìlən əmrìn qarşısında qoyula bìlìr və əmrdən ":" sìmvolu ìlə aşağıdakı kìmì ayrılır.

<nìşan> : <əmr>;

Proqramda ìşlənəcək nìşanlar ìsə əvvəlcədən **label** bölməsìndə:

label <nìşanların sìyahısı>;

formada sadalanmalıdır:

Nìşanlar -adla və ya 10000 dən kìçìk natural ədədlə (nömrə ìlə) ìfadə oluna bìlər:

Məsələ 4. Verìlmìş əmsallara görə kvadrat tənlìyì həll etmək üçün proqram yazmalı.

```
program Kvadrat_T;
uses Crt;
label 01, 02, son;
var a, b, c, s, d, x1, x2 : real;
begin
   clrscr;
   01: writeln('a,b,c verilsin');
   readln(a, b, c);
   if  a = 0  then goto son;
   d := sqr(b) – 4*a*c;
   if d >=0 then goto 02
      else
         begin
         writeln('Cavab yoxdur');
         goto 01;
         end;
   02 : if d >0 then
         begin
            x1 := (-b –sqrt(d))/(2*a);
            x2 := (-b +sqrt(d))/(2*a);
            writeln('x1 =', x1:3:1);
            writeln('x2 =', x2 :3:1);
         end
      else
         begin
         x1 := - b/(2*a);
         x2 := x1;
         writeln('x1 = ',x1 : 3: 1, 'x2 =', x2 : 3 :1);
   son : readln;
   end;
end.
```

! -yıõma əmr tərkìbìndə və ya ìçproqram daxìlìndə olan əmrlərə kənardan növbə ötürmək olmaz.

Çalışma 4. Verìlmìş əmsallara görə ìkìdəyìşənlì xəttì tənlìklər sìstemìnì həll etmək üçün proqram yazın.

5. *Seçmə əmrì* **case** –aşaõıdakı formada yazılır:

```
case  <ìfadə>  of
        <1 –cì qìymətlər sìyahısı> : <1 –cì əmr>;
        ……
        <n –cì qìymətlər sìyahısı> : <n –cì əmr>;
```

```
            else  <n+1 –cì əmr>
    end;
```

Burada ìfadə:

-yalnız tam, sìmvollu, sadalanan, və ya məntìqi tìplì dəyìşənlərdən ìbarət ola bìlər.

Qìymətlər sìyahısı –ìfadənìn tìpìnə uyõun sabìtlər və ya dìapazonlardan ìbarət olub elementlərì bìr-bìrìndən vergüllə ayrılır.

else <n+1 –cì əmr> -köməkçì hìssədìr və buraxıla da bìlər.

case əmrìndə –ìfadənìn qìymətì hansı nömrəlì sìyahıda olursa, fəalìyyət növbəsì həmìn nömrəlì əmrə keçìr. Əgər ìfadənìn qìymətì heç bìr sìyahıda tapılmazsa onda növbə n+1 –cì əmrə, bu əmr olmadıqda ìsə ";" sìmvolundan sonrakı əmrə keçìr.

Məsələ 5. Tutaq kì, «Məktəblì» -dükanındakı mal çeşìdlərì üçrəqəmlì ədədlərlə nömrələnərək ayrı-ayrı qìymətlər üzrə

Çeşìdlər Qìymətlər(manatla)
1, 2, … , 23 _____ 12 000
24, 25, … , 51____ 9 500
52, 53, … , 167____7 200
168, 169,….,741____500
742, 743, …,980____250

kìmì qruplara bölünmüşdür. Alıcının ìstədìyì m -cì çeşìddən olan n sayda malın hesabını ekrana verə bìlən proqram yazmalı.

```
program Hesab;
uses Crt;
var m, n, hesab: integer;
begin
   clrscr;
   writeln('Buyurun: ');
   readln(m, n);
   case m of
         1..23    : hesab := 12000;
         24..51   : hesab := 9500;
         52..167  : hesab := 7200;
         168..741 : hesab := 500;
         else       hesab := 250;
   end;
   write('Sizin hesab ');
   writeln(n*hesab, ' manat oldu.');
   readln;
end.
```

Məs. m =86 və n =7 ədədlərì üçün bu proqram ekrana:

Sizin hesab 50400 manat oldu.

cavabını çıxarmalıdır.

Çalışma 5. Məlumat bürosunda, 20 adda məntəqə üçün bìlet ìstəyən sərnişìnlərə xìdmət edəcək sadə bìr proqram nümunəsì tərtìb edìn.

§5. Dövrlər.

Dövr –Proqram fəalìyyətìndə müəyyən əməlìyyatın sonlu sayda təkrarlanmasıdır. Dövrü –məqsədəuyğun ardıcıllıqda düzülmüş əmrlər dəstìnìn tətbìqìlə yaratmaq mümkündür. Bu sahədə ıf, goto və xüsusì dövr əmrlərì tətbìq olunur. Tətbìq formasına görə dövr əmrlərì :

- **for -** parametrlə
- **while** -şərtönü
- **repeat** -şərtsonu varìantlarına bölünürlər.

1. *Parametrlì* **for** *əmrì* –ìkì ìstìqamətdə tətbìq olunur:

- a) İrəlìyə addımla:

```
for  <parametr> := <1 –ci ìfadə> to <2 –cì ìfadə> do  <ìç əmr>;
```

-formada verìlìr və parametr –tam, sìmvollu, məntìqì və sadalanan tìplərdən bìrìnə aid dəyìşən olub ìlk və son qìymətlərì 1 –cì və 2 –cì ìfadələrlə verìlìr.

Bu əmrdə –parametr 1 –cì qìymətì 2 –cìdən kìçìk(və ya bərabər) olarsa ìç əmr ìşləyìr və 1 –cì qìymət 1 vahìd artaraq proses təkrarlanır. Nəhayət 1 –cì qìymət 2 –cìdən böyük olunca ìdarə **for** əmrìndən sonrakı əmrə keçìr. Məs. Əgər x = 2 olarsa, onda

for i := 5 **to** 8 **do begin** x := x+ i; y := 4 * i **end**;

əmrìndə x dəyìşənì 2+5+6+7+8 =28 və y ìsə 4* ì =32 qìymətìnì alaraq ìdarə **for** əmrìndən sonraya keçìr.

- b) Gerìyə addımla:

```
for  <parametr> := <1 –ci ìfadə> downto <2 –cì ìfadə> do  <ìç əmr>;
```

-formada, analojì prìnsìplə verìlìr.

Burada -a) bəndìndən fərqlì olaraq, parametrìn 1 –cì qìymətì 2 –cìdən böyük(və ya bərabər) olarsa ìç əmr ìşləyìr və 1 –cì qìymət 1 vahìd azalaraq proses təkrarlanır. Nəhayət 1 –cì qìymət 2 –cìdən kìçìk olunca ìdarə for əmrìndən sonrakı əmrə keçìr. Məs. Əgər x = 2 olarsa, onda

for i := 8 **downto** 5 **do begin** x := x + i; y := 4 * i **end**;

əmrìndə x dəyìşənì 2+8+7+6+5 =28 və y ìsə 4* ì =20 qìymətìnì alaraq ìdarə for əmrìndən sonraya keçìr.

Məsələ 6. Müəyyən natural sırayla verìlmìş unsìya ölçülü kəmìyyətlərìn qramla ìfadəsìnì əks etdìrən cədvəl tərtìb etmək üçün proqram yazmalı.

Unsiya ölçülü kəmiyyətì -unsiya, qramla ìfadə olunmuş qiymətì qram, sıranın dəyişmə addımı -h və dəyişmə sayını - k ilə işarə etsək, tələb olunan proqramı aşaõıdakı kìmì tərtìb edə bìlərìk:

```
program Unsiya_Qram;
uses Crt;
const line = '------------------------';
var
  unsiya, qram, h: real;
    k: integer;
begin
  clrscr;
  write('Ilkin unsiya, h verilsin, ');
  writeln('k verilsin');
  readln(unsiya, h, k);
  writeln;
  writeln(line);
  writeln('Unsiya        Qram');
  writeln(line);
  for i:=1 to k do {Cavab ekrana verilir}
  begin
    qram:=28.353495*unsiya;
    writeln(unsiya:5:2, '      ', qram:10:6);
    unsiya:=unsiya+h;
  end;
  writeln(line);
  readln;
end.
```

Çalışma 6. 6 –cı məsələnì **for** əmrìnìn **downto**(gerìyə addım) varìantını tətbìq etməklə həll edìn.

Məsələ 7. Cəm və hasìldən ìbarət olan

$$y = \sum_{i=1}^{25} \frac{5i^2}{i!} + \prod_{i=1}^{25} i^2$$

ìfadəsìnìn qìymətìnì hesablamaq üçün proqram tərtìb etməlì.

Bìz cəm ìfadəsìnì – c, hasìl ìfadəsìnì -h və faktorìal ìfadəsìnì -f ilə işarə etsək, tələb olunan proqramı aşaõıdakı kìmì rərtìb etmək olar.

```
program Topla_Vur;
uses Crt;
var
  i: integer;
  c, h, y: real;
  f: longint;
begin
  clrscr;
```

```
 c:=0;
 h:=1;
 f:=1;
 for i:=1 to 25 do
 begin
   f:=f*i;
   c:=c+5*sqr(i)/f;
   h:=h*i*i;
 end;
 y:=c+h;
 writeln('y = ', y:7:2);
 readln;
end.
```

Çalışma 7. 7–cı məsələnì **for** əmrìnìn **dovnto** varìantını tətbìq etməklə

$$y = \sum_{i=5}^{20} 3i^3 + \prod_{i=5}^{20} i!$$ ìfadəsì üçün həll edìn.

2. *Şərtönü –while əmrì,* aşaõıdakı formada tərtìb olunur:

```
while <məntìqì ìfadə> do <ìç əmr>;
```

while –əmrì məntìqì ìfadə doõru olana qədər proqramı ìç əmrìn ìdarəsìndə saxlayır və yalana çevrìləndə ìsə özündən sonraya ötürür.

Məs.

- Əgər a= 3 və b = 7 olarsa onda:

while a<=6 do **begin** b := b + a; a := a + 1 **end**;

əmrìndən sonra a və b kəmìyyətlərì uyõun olaraq a = 7 və b = (7+3)+(7+4)+(7+5)+(7+6) = 46 qìymətlərìnì alar.

20 –dən 30 –a qədər ədədlər, onların kvadratları və kubları cədvəlìnì ekrana çıxarmaq üçün **while** əmrì:

```
i:=20;
while i<=30 do
begin
  writeln(i:4, i*i:6, i*i*i:8);
  i:=i+1;
end;
```

kìmì ìşlənə bìlər

- **real** tìpì üçün aşaõı sərhəddìn ìkìqatını tapmaq üçün **while** əmrì:

```
program Kiçik_real;
```

```
uses Crt;
var a:real;
begin
  clrscr;
  a := 1;
  while a / 2 > 0 do a := a / 2;
  writeln('a =', a);
  readln;
end.
```

kimi tətbiq olunaraq cavabda a =2.9E-39 alarıq.

! -kompüter hesabına görə həqiqi ədəd tipinin ən kiçik müsbət qiyməti sıfır sayılır.

Məsələ 8. Arqumentinin [0;3.1] parçasında h = 0.1 addımla dəyişməsinə uyğun y =Sin(x) funksiyasının müvafiq qiymətlər cədvəlini tərtib etməklə, (0.1;0.6) intervalında y üçün ortalama qiymət tapın.

Məsələni, müvafiq dəyişənlər qəbul edərək aşağıdakı kimi həll etmək olar:

```
program Orta_Sin;
uses Crt;
var x, y, s, s1, h, xk: real;
   n: integer;
begin
  clrscr;
  x:=0;
  xk:=3.1;
  h:=0.1;
  s:=0;
  n:=0;
  while x<=xk+h/2 do
  begin
    y:=sin(x);
    writeln(x:3:1, y:6:2);
    if (y>0.1) and (y<0.6) then
    begin
      s:=s+y;
      n:=n+1;
    end;
    x:=x+h;
  end;
  if n>0 then
  begin
    s1:=s/n;
    writeln('Ortalama = ', s1);
```

```
  end
    else
   writeln('Ortalama yoxdur n=0');
  readln;
end.
```

Çalışma 14. x və y dəyişənlərì üçün seçilmiş ìntervalları dəyìşərək 8 –cì məsələnì həll edìn.

Məsələ 9. Ìstənìlən x ədədì və e =0.001 olmaqla, n –cì həddì $a_n = (-1)^n (2x)^n / n!$ kìmì ìfadə olunan $\{a_n\}$-ədədì ardıcıllıõının $| a_n | > e$ şərtìnì ödəyən hədlərìnìn cəmìnì tapmaq üçün proqram yazmalı.

n! –kəmìyyətìnì fact, cəmı toplam və s. kìmì ìşarə edərək məsələnì həll etmək üçün proqramı aşaõıdakı kìkì tərtìb etmək olar.

```
program Toplama;
uses crt;
const e=0.001;
var toplam, x, a: real;
   fact, n, z: integer;
begin
 clrscr;
 writeln('x-i daxil edin: ');
 readln(x);
 toplam:=0;
 n:=1;
 fact:=1;
 a:=-2*x;
 z:=-1;
 while abs(a)>e do
 begin
  toplam := toplam+1;
  n:=n+1;
  z:=-z;
  fact:=fact*n;
  a:=z*exp(n*ln(2*x))/fact;
 end;
 writeln('toplam = ', toplam :5:2);
 writeln('toplananlar = ', n-1);
end.
```

Çalışma 9. 9 –cu məssələnì, e = 0.0001 ədədì üçün, daha cəmərəlì yolla həll etməyə çalışın.

3. *Şərtsonu varìantda tətbìq oıunan* –**repeat** aşaõıdakı formada tərtìb olunur:

```
repeat   <ìç əmrlər> until <məntìqì ìfadə>;
```

repeat əmrì -məntìqì ìfadə doõru oluncayadək ìdarənì ìç əmrlərdə saxlayır və sonra ìsə özündən sonrakı əmrə ötürür.

Məs. x =16, y = -3 -ìlkìn qìymətlərì üçün:

```
repeat
    y := y + x;
    x := 1/2 * x;
    z := x – 3
until x <= 5;
```

əmrìndən sonra x=4, y= -3+16 +8 =21, z =4 - 3 =1

qìymətlərìnì alar.

Məsələ 10. Argumentìn - [-π,π] parçasında, h = π / 5 addımla dəyìşməsìnə görə, y =Sìn(x) funksìyasının müvafìq qìymətlər cədvəlìnì tərtìb edìn. Bu cədvəldən mìn(y) və max(y) kəmìyyətlərìnì tapıb nətìcələrì ekrana çıxaran proqram yazın.

```
program MinMaxSin;
uses Crt;
var h, x, y, max, min: real;
begin
  clrscr;
  h:=pi/5;
  x:=-pi;
  max:=sin(x);
  min:=sin(x);
  writeln(‘   x   y’);
  repeat
    y:=sin(x);
    writeln(x:7:2, y:7:2);
    if y>max then max:=y;
    if y<min then min:=y;
    x:=x+h;
  until x>pi+h/2;
  writeln(‘max = ’, max:5:2, ‘min = ’, min:5:2);
end.
```

Çalışma 10. Argumentə aìd aralıõı dəyìşməklə, sonuncu məsələnì y=Cos(x) funksìyası üçün həll edìn.

§6. Massìvlər.

1. Massìv və ya **array** –Eynì baza tìpìnə mənsub olub, yalnız sıra nömrəsìnə görə fərqlənən, eynì adlı elementlər dəstìdìr. Bu cür elementlər bìrqayda olaraq, fəal(operatìv) yaddaşın ardıcıl yuvalarında saxlanır,

Massivlər –birölçülü və çoxölçülü olmaqla fərqlənir; məs. eynì tìplì kəmìyyətlərdən tərtìb olunmuş hər-hansı çoxsətìrlì cədvəlì ìkìölçülü massìv hesab etmək olar.

Proqramda massìv tìpìnì -ümumì şəkìldə aşaõıdakı kìmì vermək olar.

```
array  [<ölçü>] of <baza tìpìnìn adı>;
```

Burada ölçü –çox vaxt dìapazonla və bəzən də hər hansı sadalanan ünsürlər tìpìnìn adıyla verìlə bìlər.

Massìvìn tìpì, dəyìşən, massìvìn adı və s. **-type, const** və **var** bölmələrìndə aşaõıdakı kìmì verìlə bìlər.

```
type onluq = array[1..10] of real;
    yeddi = (mon, tue, wed, the, fri, sat, sun);
const on: onluq = (4, 1, 1, 2, 8, 2, 8.7, 3, 7, 1.3);
var a, a1: onluq;
   b: array[gun] of integer;
   c: array[1..100] of char;
```

- Bìrìncì sətìrdə –dìapazon ölçülü, real tìplì «onluq» massìvì təyìn edìlìr.

- İkìncì sətìrdə ìsə həftənìn günlərìnì təmsìl edən, sadalanan tìplì yeddi adlı massìv təyìn edìlìr. Bu halda, yeddi tìpì - mon, tue, wed, the, fri, sat, sun dəyərlərìndən bìrìnì ala bìlər.

- Üçüncü sətìrdə onluq tìplì yenì bìr sabìt massìvì təyìn edìlìr. on sabìtìnìn ala bìləcəyì dəyərlər mötərìzədə sadalanır.

- Dördüncü sətìrdə -hər bìrì onluq tìpìndən olan a və a1 adlı ìkì dəyìşən təyìn edìlìr kì, proqramın ìş prosesìndə bu dəyìşənlər - a[1], a[2], ..., a[10]və ya a1[1], a1[2], ..., a1[10] kìmì qìymətlər ala bìlər.

- Beşìncì sətìrdə təyìn edìlən b dəyìşənì yeddi massìvì əsasında -b[mon], b[tue], ..., b[sun] kìmì integer tìplì 7 qìymət ala bìlər.

- Altıncı sətìrdə təyìn edìlən - c dəyìşənì massìv tìplìdìr. Hər bìrì char tìplì olan 100 dəyìşəndən hər bìrìnə c[1], c[2], ..., c[100] kìmì müracìət edìlə bìlər.

Massìvlər arasında yalnız qìymətvermə əməlì təyìn edìlìr. Məs. a := a1 əmrì a1-ìn bütün qìymətlərìnì a –nın müvafìq elementìnə verìr. Massìvlərìn elementlərì üzərìndə ìsə, əlbəttə dìgər əməllər də təyìn edìlmìşdìr.

Massìv elementlərìnə müracìət aşaõıdakı formada tərtìb edìlìr:

```
<massìvìn adı>[<elementìn nömrəsì>]
```

Elementìn nömrəsì bəzən ìndeks kìmì də ìfadə olunur və kvadrat mötərìzə ìçìndə yazılır. Məs. a[1] – a massìvìnìn 1 –cì elementì və b[tue] –b massìvìnìn tue ìndekslì elementì (massìvdə ìkìncì element) deməkdìr.

Məs. İlk 100 müsbət tam ədəddən ìbarət massìv quraraq onun elementlərì cəmìnì aşaõıdakı kìmì hesablamaq olar.

```
s:=0;
for i:=1 to 100 do
begin
  a[i]:=i;
  s:=s+a[i];
end;
writeln(s);
```

Çox vaxt hər-hansı massìvìn müəyyən elementlərìnì axtarmaq lazım gəlìr kì, bu ìş -dövr və budaqlanma əmrlərìnìn axtarış şərtlərì əsasında həyata keçìrìlìr.

Məsələ 11. ATS –ə k –cı sanìyədə daxìl olan zənglərìn sayını y(k) kìmì ìfadə edək. Tutaq kì, bu ìfadənìn qìymətì:

y(k)=trunc(abs(7*Sin(k))) kìmì təsadüfì funksìya ìlə təyìn olunan 0, 1, .. ,6 ədədlərìndən bìrìdìr. ATS –ə 10 dəqìqə ərzìndə gələn zənglərìn sayını və 1 dəqìqə ərzìndə daxìl olmuş maksìmal zəng sayını hesablayaraq nətìcələrì ekrana çıxara bìləcək proqram yazmalı.

Zənglər sayını sadəcə say və s. müvafìq ìşarələr qəbul etməklə bu proqramı aşağıdakı kìmì tərtìb etmək mümkündür:

```
program ATS;
uses Crt;
type say=array[1..10] of integer;
var y: say;
   max, s, i: integer;
begin
  clrscr;
  max:=0;
  s:=0;
  for i:=1 to 10 do
  begin
    y[i]:=trunc(abs(7*sin(i)));
    write(i,'-ci san. say: ');
    writeln(y[i]:5);
    s:=s+y[i];
    if y[i]>max then max:=y[i];
  end;
  writeln('10 san. say = ', s:3);
  write('Bir san.maksimal say ');
  writeln('oldu = ', max:3);
  readln;
end.
```

Çalışma 11. Bu məsələnì həll edərkən təsadüfì ədəd almaq üçün randomize proseduraşından aşağıdakı kìmì ìstìfadə edərək:

randomize;
y[i]:=random(7);

başqa bir proqram yazın.

Məsələ 12. Elementlərì y(k) =ln(k) –3; k = 1, 2, .. ,10 düstuıru ilə hesablanan y massìvì qurub bu massìvìn mənfì elementlərìndən ìbarət g -ìç massìvìnì ayıran və nətìcələrì ekrana çıxara bìlən proqram tərtìb etməlì.

Bu proqramda -g massìvì elementlərìnìn sayı n ìlə ìşarə edìlìr.

```
program Içmassiv;
uses Crt;
var y, g: array[1..10] of real;
   k, n: integer;
begin
  clrscr;
  n:=0;{ilkin say}
  for k:=1 to 10 do
  begin
    y[k]:=ln(k)-3;
    if y[k]<0 then {g –üçün element aramaq}
    begin
      n:=n+1;
      g[n]:=y[k]; {n –ci elementi tapmaq}
    end;
    writeln('y(', k, ')= ', y[k]:7:2);
  end;
  if n=0 then
    writeln('g –içmassivinin elementi yoxdur')
  else
    for k:=1 to n do {g –içmassivi ekrana verilir}
      writeln('g[', k, ']= ', g[k]:7:2);
  readln;
end.
```

Çalışma 12. 12 –cì məsələnì y(k)=lg(k⁄100)+1; k=1,2,.. ,120. ìfadəsì üçün həll edìn.

2. *Ìkìölçülü massìvlər* –əsasən cədvəllərlə ìşləmək üçün nəzərdə tutulur və elementlərì matrìs üsuluyla düzülərək qoşa ìndekslə nömrələnìr. Bu ìndekslər aìd olduõu elementìn uyõun olaraq, sətìr və sütun nömrələrìnì göstərìr. Məs. A[3,2] elementì -A massìvìnìn 3 –cü sətrìndə 2 –cì elementdìr.

Nümunə üçün bəzì massìvlərìn verìlmə üsullarına baxaq:

- Qìymət balı üçün 3 sətìr və 4 sütunlu «Bal» adlı bìr massìvì:

```
Bal: array[1..3, 1..4] of integer=((4, 3, 5, 3), (4, 4, 5, 3)),(3, 4, 5, 5);
```

- İlin günlərini ifadə edən, 12 x 31 ölçülü tam ədəd tiplî «İl» massivi:

```
var İl: array[1..31] of array[1..12] of char;
```

- İkirəqəmlî ədədlər arasında vurma cədvəlini yazmaq üçün 89 x 89 ölçülü «Vurma» massivi:

```
const n = 99;
   p: array[11..n, 11..n] of integer;
```

Birinci nümunələrdə bal[1, 3] =5 və bal[3, 2] =4 qiymətini alır.

Proqramda İl massivinin elementləri isə

İl[21][4] := 21 Aprel və ya İl[9, 10] := 9 Oktyabr və s. kimi tanıdıla bilər.

Məsələ 13. İkirəqəmlî ədədlər arasında vurma cədvəli tərtib edib ekrana çıxara bilən proqram yazmalı.

```
program Vurma;
uses Crt;
const n=99;
var p: array[11..n, 11..n] of integer;
   i, j: integer;
begin
  clrscr;
  for i:=11 to n do
  begin
    for j:=11 to n do
    begin
      p[i, j]:=i*j;
      write(p[i, j]:6);
    end;
    writeln; {Massiv ekrana verilir}
  end;
  readln;
end.
```

Çalışma 13. 1 dərəcəlik addımla, itî bucaõın sinuslar cədvəlini tərtib edib ekrana çıxara bilən proqram yazın.

Məsələ 14. Tutaq ki, fabrikdə 5 adda məhsuldan 5 növ şirniyat hazırlanır və 1kq j –ci növ şirniyata i –ci məhsulun məsrəfi

$a_{ij} = 2|Sin(i)| + j$, i, j = 1, 2, .. , 5 düsturu ilə hesablanır. Elementləri a_{ij} olan a massivini və ya başqa sözlə desək - məhsul məsrəfi cədvəlini tərtib edib 3 –cü məhsuldan minimal miqdarda tələb edən şirniyat növünü tapan və nəticələri ekpana çıxaran proqram tərtib etməli.

Proqramda i –ci növ şirniyatı –imin və bu növə işlənən 3 –cü növ məhsulun minimal miqdarı –min kimi işarə edilir.

```
program Fabrika;
uses Crt;
```

```
type mesref = array[1..5, 1..5] of real;
var i, j, imin: integer;
   min: real;
   a: mesref;
begin
  clrscr;
  writeln('              mehsul');
  writeln('       1    2    3    4    5');
  for i:=1 to 5 do {a massivi qurulur}
  begin
    write(i, 'növ');
    for j:=1 to 5 do
    begin
      a[i, j]:=2*abs(sin(i))+j;
      write(a[i, j]:7:2);
    end;
    writeln;
  end;
  imin:=1; {imin =1 olarsa..}
  min:=a[1, 3];
  for i:=2 to 5 do
    if a[i, 3]<min then
    begin {Axtarilan sortu tapmaq}
      min:=a[i, 3];
      imin:=i;
    end;
  write(imin, ' növü');
  writeln('daha az aparir');
  readln;
end.
```

Çalışma 14. 14 –cü məsələnì 6 növ metaldan 4 adda xəlìtə almaq üçün dəyìşdìrərək həll edìn.

Məsələ 15. Elementlərì x_{ij} =ì +j^2 , ì, j = 1, 2,. .. ,5 düsturu ìlə hesablanan X massìvìnì və onun 20 –dən böyük elementlərìndən yenì bìr Y massìvì tərtìb edərək X, Y massìvlərìnì ekrana çıxara bìlən proqram yazmalı.

```
program Massiv5_5;
uses Crt;
type massiv=array[1..5, 1..5] of real;
    yenimas=array[1..25] of real;
var X: massiv;
   Y: yenimas;
   i, j, k: integer;
begin
  clrscr;
```

```
 k:=0; {ilk olaraq k = 0 olsun}
 writeln('X massivi ekrana verilir');
 for i:=1 to 5 do
 begin
  for j:=1 to 5 do
  begin
   b[i, j]:=i+j*j;
   write(b[i, j]:7:2);
         {Y üçün element seçilir}
         if b[i, j]>20 then
   begin
    k:=k+1;
    y[k]:=b[i, j];
   end;
  end;
  writeln;
 end;
 if k=0 then
  write('Y massivinin elementi yoxdur')
 else begin
  writeln('Y massivi');
  for i:=1 to k do
   writeln(y[i]:7:2);
  write('Y massivinin = ', k:2);
      writeln("sayda elementi var.")
 end;
 readln;
end.
```

Çalışma 15. Elementlərì a_{ij} =(ì+1).(j+1), ì, j = 1, 2, .. , 10 düsturu ilə hesablanan a massìvìnì və onun ìkìrəqəmlì elementlərìndən ìbarət bìr b altmassìvìnì tərtìb edərək ekrana verə bìlən proqram yazın.

§7. Sətìrlər (strìng)

Sətìr tìplì ünsürlər –ìxtìyarì çeşìddə sìmvollardan ìbarət məhtud sayda (255 –ə qədər) char tìplì elementlər dəstìdìr. Hər belə dəst bìr cüt apostrof ìşarəsì arasında –məs. 'Bizim diyar – Odlar yurdudur!' və boş sətìr ìsə ' ' (arada yalnız "ara" sìmvolu olmaqla) kìmì verìlìr.

Sətìr tìplì dəyìşənlər –**var** əmrì bölməsìndə strìng sözüylə aşaõıdakı formada tanıdılır.

```
var  <dəyìşən> : string[n]
```

Burada n –verìlmiş sətìr tìplì dəyişənìn sìmvollar sayı olub zərurì göstərìcì sayılmır. Məs:

const dil = ‘Pascal 7.0 asan dildir’;
var ad : **string**[10];
okean : **string**;

formada təyìn olunmuş **dil** sətìr sabìtìnə **‘Pascal 7.0 asan dildir’** qìymətì verìlìr, **ad** dəyişənì ən çoxu 10, lakìn **okean** dəyişənì ìsə 255 –ə qədər sìmvoldan ìbarət olan qìymətlər ala bìlər.

Sìmvol tìplì dəyişənlər arasında: bìrləşmə (+) və müqayìsə (<, <=, >, >=, =, <>)əməllərì təyìn olunmuşdur. Müqayìsə zamanı sətìrlər (eynìlə müsbət-tam ədədlərdə olduõu kìmì) soldan saõa, ìlk fərqlì sìmvoladək yoxlanır və kompüterìn kodlar cədvəlìndə kodu böyük olan sìmvol böyük sayılır; hərflərìn kodu ìsə bu cədvəldə əlìfba sırasına uyõun olaraq artır.

Məs. **‘a’ < ‘u’**, olduõundan **‘Şahbaz’ < ‘Şahbuz’** və **‘s’ < ‘t’** olduõu üçün **‘Abbat’ > ‘Abbas’** və s.

Əgər **ad1 = ‘Qara ’** və **ad2 = ‘Yusif’** olarsa, onda

ad3 := ad1 + ad2 əməlìndən sonra **ad3 = ‘Qara Yusìf’** qìymətì almış olar kì, burada **ad1 < ad3 < ad2** sayılır.

Sìmvolların kodu **ord** funksìyasıyla - **ord(‘A’)=65, ord(‘B’)=66** kìmì, əks əməl ìsə **chr** funksìyasıyla **chr(67)=’C’**-kìmì müəyyən edìlìr. Məs. Latın hərflərìnìn kodlar sìyahısını ekrana çıxarmaq üçün **ord** funksìyasını aşaõıdakı formada tətbìq etmək olar:

```
for v :=‘a’ to ‘z’ do
        writeln(v, ord(v):5);
```

İndì ìsə sətìr tìplì ünsürlərlə işləməyə ìmkan verən -*standart funksìyaları* və dəyişənlə verìlən sətìrlərlə işləmə ìmkanı verən -*prosedurаları* nəzərdən keçìrək.

Standart funksìyalar.

length(<sətìr>)	sətrìn sìmvol sayını verìr;
copu(r, m, n)	r –cì sətrìn m –cì dən başlayaraq n sìmvolunu verìr.
concat(r1,r2,..,rn)	r1, .. , rn nömrəlì sətìrlərì bìr sətìr halında bìrləşdìrìr.
pos(r1, r2)	r1 sətrìnìn r2 –yə daxìl olmaõa başladıõı sìmvolun nömrəsìnì verìr.

Məs. okean1=‘Sakit okeanda adalar boldur’, ad4 =‘Atlantik’ qìymətlì dəyişən sətìrlərə bu funksìyaları aşaõıdakı formada tətbìq etsək:

Verìlən funksìya	*Alınan cavab*
length(okean1)	27;
copu(okean1, 15,6)	‘adalar’;
concat(okean1, ‘,’ , ad4, ‘okeanda az.’)	‘Sakit okeanda adalar boldur, Atlantk okeanda az.’;
pos(‘bol’, okean1)	22;

kìmì cavablar alınar.

Proseduralar.

insert(r1, <dəyişən>, n)	dəyìşənlə verìlmìş sətrə, n –cì mövqedən başlayaraq r1 sətrì də daxìl edìlìr;.
delete(<dəyìşən>, m, n)	dəyìşənlə verìlmìş sətrìn m –cì mövqedən başlayan n sìmvolu sìlìnìr.
str(<ədəd>, <dəyìşən>)	Ədədì sətìr tìplì ünsürə çevìrìr.
val(r1, s1, s2)	r1 sətrìnìn ədədì obrazını s1 ədədì dəyìşənì ìçərìsìnə göndərìr. Bu mümkün olarsa s2 – ədədì dəyìşənì, əks halda ìsə buna əngəl olan ìlk sìmvolun ədədì obrazı 0 –a çevrìlìr.

dil, okean1 və ad4 dəyìşənlərìnə bu proseduraları tətbìq etmsək:

Prosedura	*Cavab*
insert(‘vulkanìk ’, okean1, 15)	‘Sakit okeanda vulkanik adalar boldur’
delete(okean1, 1, 5)	‘okeanda adalar boldur’
str(7.0, dil)	‘7. 0’
val(‘2003’, dil, ad4)	dil =2003, ad4 =0

kìmì əməlìyyatlar aparmaq olar.

Beləlìklə -string tìpì üzərìndə həm bütöv sətìrlər halında və həm də onları sìmvol massìvì kìmì qəbul edərək elementlərə ayırmaqla ìşləmək mümkündür. Bu əməlìyyatları ìsə əsasən, ekranla dìaloq rejìmìndə aparmaq münasìbdìr. Məs. Sual – cavab qurmaq üçün

```
sual := ‘-Siz Paskal dilini bilirsinizmi?’;
writeln(sual);
for i := 1 to 23
        read(cavab[ı]);
```

əmrì əvvəl ekrana

-Siz Paskal dilini bilirsinizmi?:

sualını çıxarıb ona 23 sìmvollu bìr cavab yazmaða ìmkan verərdì.

Məsələ 16. (Mətnì kodlaşdırma.) Verìlmìş a -mətnìndə bütün aralıqları, nöqtə və vergüllərì ləõv edərək qalan sìmvolları əvəzləyərək nətìcənì ekrana verə bìlən proqram yazmalı.

```
Program  Kodlama;
Uses     Crt;
Var      a,b,c: string;
         i:integer;
begin
       clrscr;
       write('a verilsin:');
       readln(a);
       b:='';
   for i:=1 to Length(a) do
begin
c:=copy(a,i,1);{yaxud element-element: c :=a[i];}
if     (c<>`,`) and (c<>`.`) and (c<>``)
then   b:=b+c+c;
end;
                 writeln(b);
                 readln;
end.
```

Çalışma 16. İlk 35 sadə ədədì və əlìfba sırasındakı hərflərì ardıcıl olaraq nömrələyìb, verìlmìş mətnìn hər bìr hərfìnì eynì nömrəlì sadə ədədlə əvəz edən proqram yazın.

Məsələ 17. Verìlmìş «Yazda» adlı mətndə, səhv getmìş «qulaq» sözünü «bulaq» sözü ìlə əvəz edən proqram yazmalı.

```
program Redaktor;
uses Crt;
var Yazda, qulaq, bulaq: string;
   i,k:integer;
begin
   clrscr;
   write('Yazda verilsin:');
   readln(Yazda);
   write('pozulacaq fraza verilsin:');
   readln(qulaq);
   write('yazIlacaq fraza verilsin:');
   readln(bulaq);
   k:=length(qulaq);
   for i := 1 to length(Yazda)-k do
     if copy(Yazda,i,k)=qulaq then
```

```
            begin
            delete(Yazda,i,k);
            insert(bulaq,Yazda,i)
            end;
      writeln(Yazda);
      readln;
end.
```

Çalışma 17. «Yazda» adlı bir mətn tərtib edib 17 –ci məsələni sərbəst seçdiyiniz şərtə uyğun olaraq həll edin.

Məsələ 18. 'Biz Odlar yurdunu sevirik!' sətrinin simvollar sayını və sətrin ikinci sözünü tapıb ekrana verə bilən proqram yazın.

```
program Odlar yurdu;
uses Crt;
const r1:string = 'Biz Odlar yurdunu sevirik';
var i,k,m,n1,n2: integer;
begin
    clrscr;
    m:=0;
    k:=length(r1)   {simvol sayma}
    writeln(`uzunluq k=`,k);
    for i:=1 to k do {bir-bir yoxlamaqla}
    if r1[i]='' then {ara simvolu aramaq}
begin
    m:=m+1;
    {ilk ara simvolunu axtarmaq}
    if m=1 then n1:=i;
    {ikinci ara simvolunu tapmaq}
    if m=2 then n2:=i
end;
{cavab ekrana verilir}
for  i:= n1+1 to n2 –1 do write(r1[i]);
readln;
end.
```

Çalışma 18. Ara simvolu tapmaq üçün –**pos** funksiyasını və ikinci sözü seçib ekrana çıxarmaq üçün **-delete** proseduraşını tətbiq edərək 18 –ci məsələni həll edin.

§8. İçproqramlar.

Göründüyü kimi, bu kitabda təqdim olunan proqram nümunələri olduqca sadə və kiçik həcmlidir, çünki bu proqramlar yalnız tədris məqsədi daşıyır. Lakin getdikcə müasir həyatın ən gözlənilməz sahələrinə belə tətbiq olunan

praktìk proqramların ìsə nə qədər çoxşaxəlì və əndazəsìz olduğunu təsəvvür etmək çətìn deyìl. Bu cür proqramların mürəkkəb alqorìtm və strukturası da öz növbəsìndə -*struktur proqramı* deyìlən bìr konsepsìyanın tətbìq olunmasını tələb edìr.

Struktur proqramları –özündə əsasən aşağıdakı prìnsìplərì cəmləşdìrìr:

- Qarşıya qoyulmuş məsələnìn analìzì və həll alqorìtmìnìn ayrı-ayrı sadə hìssələrə bölmək.
- Bütün hìssələrì detallaşdıraraq müvafìq prìorìtet əsasında ìçproqramlar tərtìb etmək.
- İçproqramlarda –sadə, budaqlanma, dövr, vaxttənzìmləyìcì və s. kìmì baza əmrlərìnì səmərəlì ìstìfadə etmək.
- Proqramı abonentlərə aydın olan bìr formada yazmaq.
- Şərtsìz keçìd əmrlərìndən mümkün qədər az ìstìfadə etmək.
- Proqram fəalìyyətìnì ìş ərəfəsìndə və ìş prosesìndə məntìqì baxımdan yoxlamaq və s.

Burada detallaşma, əsasən -alqorìtmìn yuxarıdan aşağıya mərhələ – mərhələ şaxələnməsì və addım-addım yoxlanması prìnsìpì üzrə həyata keçìrìlìr.

Proqramda bu və ya dìğər bìr sahənì əhatə edən ìçproqramlarda hansı baza əmrlərìnì ìşlətməyìn səmərəlì olacağı da müəyyən edìlməlìdìr. Onu da deyək kì, ìstənìlən dərəcədə mürəkkəb alqorìtmlərìn –sadə, budaqlanma və dövr bəndlərì əsasında qurula bìlməsì artıq sübuta yetìrìlmìşdìr.

Struktur proqramlarında sìstemsìz keçìd əmrlərì proqramın oxunuşunu çətìnləşdìrdìyì üçün onların mümkün qədər **ìf-then-else, case, while** kìmì əmrlərlə əvəz olunması daha səmərəlì sayılır.

Zərurì hallarda proqramın ìdarəsìnì bəzì əmrlərdən almaq üçün **-exit, break, continue** və **halt** kìmì çıxış əmrlərì tətbìq olunur.

Proqramın ìşìnə nəzarət etmək üçün, əsasən dəyìşənlərìn qìymətlər cədvəlìnì tərtìb etmək üsulundan genìş ìstìfadə edìlìr.

Struktur proqramlarını təşkìl edən ìçproqramları ìstənìlən nahìyədən və lazımı dəfə çağırmaq olar. Qeyd edək kì, ìçproqramlar əsasən ìkì cür: *funksìya - ìçproqram* və *prosedura - ìçproqram kìmì* fərqlənìr. Proqramda **System, Crt, Dos, Graph** standart modullarının –**procedure** və **function** bölmələrìndə təqdìm olunan standart *abonent ìçproqramlarından* ìstìfadə etmək ìsə olduqca sərfəlìdìr.

2. Procedure –ìçproqramı:

```
procedure  <ad>(<formal parametrlər sìyahısı>);
                    <proseduraların təqdìmat bölməsì>;
                beqın
            <prosedura əmrlərì bölməsì>;
```

```
        end;
```

formada tərtìb olunur. Burada:

- formal parametrlər sìyahısında dəyìşən və onların tìplərì sadalanır.
- verìlən ünsürlər parametr–arqument, alınan cavablar ìsə parametr-nətìcə adlanır və bu parametrlər dəyìşənlər kìmì **var** bölməsìndə sadalanmalıdır.

Məs. Hər dəqìqəsì 500 man. və 20% əlavə xərc tələb edən telefon danışıõını hesablamaq üçün:

```
procedure Tel_hesab(k:integer; var c:real);
            begin
                  c := k*500;
                  c := c + 0.2*c;
            end;
```

kìmì bìr ìçproqram yazmaq olar kì, burada: k –danışıq müddətìnì ìfadə edən *arqument* -və ya parametr - arqument, c -ìsə k müddət üçün yekun danışıq haqqı - və ya parametr - nətìcədìr.

Proqramın ìstənìlən nahìyəsìndən və ya dìgər bìr ìçproqramdan procedure əmrìnə aşaõıdakı kìmì müracìət etmək olar:

```
<proseduranın adı> (<faktìkì parametrlərìn sìyahısı>);
```

Qeyd edək kì, müracìət əmrlərìndə sadalanan bu cür faktìkì para-metrlər də arqument və nətìcə kìmì fərqlənìrlər. *Faktìkì arqumentlər* –sabìt, dəyìşən və ya ìfadə formasında ola bìldìyì halda, f*aktìkì nətìcə* –yalnız dəyìşən ola bìlər və onların tìpì müracìət əmrìndə verìlmìr.

Müracìət əmrìndə -verìlən hər faktìkì arqumentìn qìymətì procedure əmrìndə sadalanan müvafìq formal arqumentə verìlìr, sonra ìsə, hər faktìkì nətìcə müvafìq formal nətìcənìn qìymətìnì mənìmsəyìr. Əlbəttə bu zaman uyõun faktìkì və formal parametrlərìn adları eynì olmaya bìlərsə də tìplərì bìr-bìrìnə uyuşan olmalıdır.

Əsas proqramın **var** əmrìlə verìlən dəyìşənlər *qlobal dəyìşənlər* olub ìstənìlən ìçproqramda belə ìşləyə bìlər, lakìn procedure ìçproqramında verìlən dəyìşənlər ìsə *lokal dəyìşənlər* sayılır və yalnız öz tərkìbìndə ìşləyə bìlər. Başqa sözlə, procedure əmrìndə *qlobal dəyìşənlər* tanınır və demәlì burada formal nətìcə parametrlərìnì sadalamaõa ehtìyac qalmır.

Məsələ 19. ATS –ə daxìl olan zənglər sayı ìlə baõlı verìlən 11 –cì məsələnì aşaõıdakı proseduraların köməyì ìlə həll edək:

1) Bir_say –hər sanìyədə daxìl olan zənglərìn sayı.
2) On_say – ìlk 10 sanìyədə daxìl olan zənglərìn sayı.
3) Max_say -bìr sanìyədə daxìl olan maksìmal zəng sayı.

Tutaq kì, k –cı sanìyədə daxìl olan zənglərìn sayı y(k)=random(k) təsadüfì funksìyası ìlə təyìn olunan tam ədəd olsun və

- bì.r sanìyədə daxìl ola bìləcək zənglər sayını –birsay
- bu saylardan ən böyük olanı -maxsay
- 10 sanìyədə daxìl ola bìləcək cəmì zənglər sayını –onsay

ìlə ìşarə etməklə bu proqramı aşaõıdakı kìmì tərtìb etmək mümkündür:

```
program ATS;
uses Crt;
type birsay = array[1..10] of integer;
var y : birsay;
   maxsay, s : integer;
procedure On_say(var y : birsay);{prosedure On_say}
var i: integer;
begin
   for i:=1 to 10 do
     begin
     y[i]:=random(i);
     writeln('y(',i,')=',y[i]:5);
     end;
end;
procedure On_say(y : birsay; var s:integer);
                                                {prosedure On_say}
var i:integer;
begin
    s:=0
    for i:=1 to 10 do s:=s+y[i];
    writeln(`On sani.say S=`,s:3);
end;
procedure Max_say(y:birsay; var maxsay:integer);
var i:integer;
begin
    maxsay := y[1];
    for i:=2 to 10 do
        if maxsay < y[i] then maxsay := y[i];
    write(`Bir san. maksimal`)
    writeln(`miqdarda say`,max:3)
end;
begin
    clrscr;
    randomize;
```

```
    Bir_say(y);  {proseduralara xitab}
    On_say(y,s);
    Max_say(y,maxsay);
    readln;
end.
```

Çalışma 19. 19 –cu məsələdə – ATS –ə hər sanìyədə daxìl olan zənglər sayını, metro keçìdìndən hər dəqìqədə keçən sərnìşìnlər sayı ìlə əvəz etməklə dəyìşdìrərək alınan məsələnì həll edìn.

3. Function –ìçproqramı:

```
function  <ad>(<formal parametrlwr>);
                                    <funksìyanın tìpì>;
        <funksìiyanın təqdìmat bölümü>;
      begin
     <ad := ìfadə>
  end;
```

formada tərtìb olunur və proseduradan fərqlì olaraq xìtab bölümünə yalnız bìr standart tìplì cavab qaytarır. function ìçproqramına aşaõıdakı formada xìtab edìlìr

```
<ad> (<faktìkì parametrlər>);
```

və cavablar əsas proqrama, standart funksìyalarda olduõu kìmì- - funksìyanın adıyla qaytarılır.

Məs. tg(x) –ìn qìymətìnì almaq üçün **function** tərtìb edərək tg(x) +ctg(x) +tg^2(x) ìfadəsìnì aşaõıdakı proqramla hesablaya bìlərìk.

```
program tg_func;
uses Crt;
var x,y:real;
function tg(x:real):real;
begin
        tg(x)=sin(x)/cos(x);
end;
begin
        clrscr;
        writeln('x –i verin');
        readln(x);
        y := tg(x)+1/tg(x)+ sqr(tg(x));
        writeln('y =', y:5:2);
        readln;
end.
```

Məsələ 20. Tutaq kì, boyaqçı emalatxanasında 6 növ rəngdən ìstìfadə edərək 10 çeşìddə boya hazırlanır və j –cì çeşìdə vurulan ì –cì növ rəngìn mayası (məsrəfì)

$m_{ij} = 2|\ln(ì)| + j$, ì =1, 2,..,6; j = 1, 2, .. , 10 düsturu ìlə hesablanır.

Elementlərì m_{ij} olan m massìvìnì tərtìb edərək 3 –cü rəngdən mìnìmal mìqdarda tələb edən çeşìdì tapıb nətìcələrì ekpana çıxaran struktur proqramı tərtìb etməlì.

Proqramda ì –cì çeşìdì –imin və ona işlənən 3 –cü növ rəngìn mìnìmal mìqdarı –min kìmì ìşarə edìlìr.

```
program Boyaq;
uses Crt;
const k =6, n =10;
type maya = array[1..k, 1..n] of real;
var i, j, imin: integer;
   min: real;
   m: maya;
function boya(i,j:integer):real;
begin
    boya :=2*abs(ln(i))+j;
end;
procedure Mass(var m : maya);
var i, j : integer;
begin
  writeln('                Boya');
  writeln('       1    2   3   4   5   6');
  for i:=1 to 6 do {m massivi qurulur}
  begin
   write(i, ' növ');
   for j:=1 to 10 do
   begin
    m[i,j]:=boya(i,j);
    write(m[i,j]:7:2);
   end;
   writeln;
  end;
end;
procedure Minimal(m:maya; var imin:integer);
var   i, j : integer;
          min : real;
begin
          imin :=1;
          min := m[1,3];
          for i :=2 to k do
          if m[i,j] < min then
```

```
            begin
            min := m[i,3];
            imin := i;
            end;
        write(imin, 'növü');
      writeln('daha az maya aparIr');
      readln;
end;
begin
clrscr;
Mass(a);
Minimal(m, imin);
readln;
end.
```

Çalışma 20. 14 –cü çalışmanı struktur proqramı tərtìb etməklə həll edìn.

Əlbəttə bu cür - tədrìs xarakterlì, sadə proqramlardan fərqlì olaraq, əməlì məqsəd daşıyan -çox böyük və mürəkkəb proqramlar adətən kollektìv işìn məhsulu olub çoxsaylı praktìk yoxlamalardan keçìrìlìr. Təsadüfì deyìldìr kì, Microsoft Corporation, IBM Corporation və s. bu kìmì nəhəng fìrmalarda, ayrı-ayrı sahələr üzrə yüzlər və bəzən də mìnlərlə mütəxəssìs çalışır. İstehsalatın, nəqlìyyatın bu və ya dìgər sahələrìnə xìdmət edən bìr çox proqramlar, bəzən hətta ümumdövlət səvìyyəlì bìr məsələ kìmì qarşıya qoyularaq bìr çox struktur proqramlarından ìbarət sìstem kìmì ìşlənìb hazırlanır.

4. *Rekursìv funksìyalar* –əsasən functionların öz-özünə xìtab etməsìlə təşkìl olunan dövrü proses kìmì təzahür edìr. Başqa sözlə desək, *rekursìya* – funksìya qìymətlərìnì özünə arqument kìmì təqdìm etməklə əldə edìlìr. Rekursìyanın təşkìlìndə *axın* anlayışı da genìş tətbìq olunur. *Axın* –fəal yaddaşda, qəbul etmə ardıcıllıõına əks növbəylə xarìc etmə prìnsìpì üzrə saxlanan elementlər toplusudur.

Məs. m –dən n –ə qədər tam ədədlərìn cəmìnì hesablamaq üçün aşaõıdakı kìmì rekursìv fuhksìya qurmaq olar.

```
function Toplama(m,n:integer):integer;
begin
    if m = n then Toplama := m
    else Toplama := n+Toplama(m, n-1)
end;
```

Bu ìçproqram, məs. Toplama(3,5)ìfadəsìnì aşaõıdakı ardıcıllıqla hesablayır:

Toplama(3,5)=5 + Toplama(3,4)=5 +4 +Toplama(3,3)= 5+4+3.

Yənì, blok toplamanı ìkì etapda yerìnə yetìrìr: -əvvəlcə 5, 4, 3 ədədlərìndən *axın* təşkìl edìr, -sonra ìsə onları tərs sırayla toplayır.

Məsələ 21. İlk n natural ədədìn hasìlìnì n! =n(n-1)! –düsturunu (0!=1, 1!=1) ardıcıl olaraq tətbìq etməklə hesablamaq üçün rekursìv funksìya qurmalı.

```
function Vurma(n:integer):integer;
begin
    if n =0 then Vurma:=1
    else Vurma := n * Vurma(n-1)
end;
```

Bu functionla 4! –ın qìymətì aşaõıdakı etaplarla hesablanır:

Vurma(4)=4.Vurma(3) =4.3.Vurma(2)=

=4.3.2. Vurma(1)=4.3.2.1.Vurma(0)=4.3.2.1.1

Demək burada axın 4, 3, 2, 1, 1 sırasıyla daxìl olub hasìl 1.1.2.3.4= =24 kìmì ardıcıllıqla yerìnə yetìrìlìr.

Çalışma 21. Bìrìncì həddì 30, 8 –cì həddì 46 olan ədədì sìlsìlənìn ìlk n həddìnìn cəmìnì tapmaq üçün rekursìv funksìya daxìl olan proqram yazın.

5.*Açıq massìvlər* -ìçproqramların formal parametrlərì sìyahısında verìlən, ölçüsü əvvəlcədən bəllì olmayan massìvlər olub aşaõıdakı formada təqdìm olunur:

```
<ad> : array of <baza tìpìnìn adı>;
```

Bu cür formal massìvlərìn elementlərì 0 –dan başlayaraq nömrələnìr və sonuncu elementìnìn nömrəsì standart **high** funksìyasıyla:

```
high(<massìvìn adı>).
```

formada öyrənìlə bìlər.

Açıq massìvlərdən, prosedurada dəyìşən ölçülü massìvlərlə ìşləmək üçün ìstìfadə olunur.

Məsələ 22. İçproqramların köməyìlə, elementlərì uyõun olaraq

y_m =fy(m) =random(m), m = 1, 2, .., 7 və g_n =fg(n) = n^2 /2,

n = 1, 2, .., 9 düsturlariyla hesablanan y və g massìvlərì qurun. Hər massìvdə 4 –dən böyük elementlərìn sayını tapıb nətìcələrì ekrana çıxarın.

```
program Struktur;
{$F+}
uses crt;
type        fu =function(n:integer):real;
var         y:array [1..7] of real;
            g:array [1..9] of real;
function fy(m:integer):real;
begin
```

```
            y:=random(m)
end;
function fg(n:integer):real;
begin
g := n*n/2
end;
procedure Yarat(f:fu; var z: array of real);
var i:integer;
begin
            for i:=0 to high(z) do
                        begin
                        z[i]:=f(i+1);
                        write(z[i]:5:2);
                        end;
            writeln
end;
function top(z:array of real):integer;
var i,k:integer;
begin
            k:=0;
            for i:0 to high(z) do
                        if z[i]>4 then k:=k+1;
            top:=k
end;
begin
            clrscr;
            randomize;
            Yarat(fy,y);
            Yarat(fg,g);
            write(`AxtarIlan element sayI y -`);
            writeln(`k=`,top(y):3);
            write(`AxtarIlan element sayI g -`);
            writeln(`k=`,top(g):3);
            readln
end.
```

Dìqqət edìn kì, bu proqramda ìlk dəfə olaraq.

```
type            fu =function(n:integer):real;
```

bölümündə function tìpì verìlərək fy(x) və fg(x) funksìyaları da bu tìpə aìd edìldì, beləlìklə də bìr proseduraylа müxtəlìf massìvlər qurmaq mümkün oldu.

Həmçinin bu səbəbdən proqrama zərurì içproqram modelinì(far-model) tənzìm edən {$F+} dìrektìvì daxìl edìldì.

Çalışma 22. 22 –cì məsələnì y_m =fy(m) =random(m), m = 1, 2, .., 9 və g_n =fg(n) = $(n + 1)^2$ /2; n = 1, 2, .., 14 düsturlariyla hesablanan y və g massìvlərì üçün dəyìşdìrməklə həll edìn.

6. *Standart modullar* –genìş ìstìfadə olunan unìversal təyìnatlı içproqramlar olub bìr «kìtabxana» adı altında cəmləşmìş olurlar.

Modul –özündə sabìt, dəyìşən ünsür tìplərì və içproqramlar bìrləşdìrən proqram vahìdì olub standart və abonent modulları kìmì fərqlənìrlər.

Standart modullar.

System	Çoxìşlənən prosedura və funksìyalar sìst.
String	Sətìr dəyìşənlərì ìlə ìşləyən funksìyalar.
Printer	Prìnterlə ìşləmək üçün proqram modulu.
Graph	Qrafìkqurma prosedura və funksìyaları.
Overlay	Böyük proqramlarla ìşləmək üçün modul.
Dos, Windows	Paskal proqramı gedìşìndə Əməlìyyat sìstemì əmrlərìnìn ìşləməsìnə ìmkan verìr.
Graph3, Turbo3	TP versìyaları arasında uyarlıq saxlayır.

Modulları **uses** əmrìlə hər hansı bìr proqramın ìşìnə

```
uses <modulların adlar sìyahısı>;
```

formada cəlb etmək olur.

Sustem modulundan ìsə, proqramlar elə-belə də (yənì, ona müracìət etmədən belə -avtomatìk olaraq) ìstìfadə edə bìlìrlər. **read, readln, write, writeln** proseduralarını və sìn, cos və s. standart funksìyaları da proqramlar elə məhz bu moduldan alırlar.

Sustem və Crt modullarına aìd olan əsas proseduralar.

exit	Əsas proqramın ìşìnì saxlamaq və ìçproqramdan çıxmaq üçün ìşlənìr.
halt	Proqramı saxlayıb ìdarənì Əməlìyyat sìstemìnə verìr.
break	for, while, və repeat dövrlərìndən məcburì çıxma.
continue	for, while, və repeat dövrlərìndə növbətì ìterasìyanı başlamaq üçün ìşlənìr.
delay(n)	Carì əmrìn ìşìnì abonentìn verdìyì vaxt (n san) ərzìndə (mìkro san. dəqìqlìklə) saxlayır.
clrscr	ekranı təmìzləyìr.
textcolor	Mətnə 0, 1, .. , 15 ədədlərì ìlə verìlən rəng.
textbackground	ekran fonuna rəng verìr.

Habelə:

- **Crt** modulunda, -proqramı gözləmə rejìmìnə keçìrməklə abonentìn qırpdıõı sədəfìn qìymətìnì alan **readkey** və ìstənìlən sədəfì qırpanda **true** məntìqì dəyərìnì alan **keypressed** funksìyaları və s. yerləşìr.
- Paskal proqramı rejìmìndə fayllar sìstemì ìlə işləmək üçün nəzərdə tutulmuş prosedura və funksìyalar ìsə **Dos** modulunda yerləşìr.

Onlara, carì proqram daxìlìndən:

exec('<exe faylın bütöv adı>','<proqramın parametrlərì>' və ya").

kìmì müracìət etmək olar.

Lakìn **exec** prosedurası işlənən proqrama **M** dìrektìvìnì də:
–məs. {$M $2000,0,1000} parametrìlə- qoşmaq lazımdır.

- Proqram və ya onun fraqmentlərìnìn ìş vaxtının xronometrìyası, və tarıx göstərìcìlərìlə ìşləmək üçün aşaõıdakı proseduralar tətbìq edìlìr:

GetTime(Hour, minute, Second, SotiSec)

-{carì vaxta -saat, dəqìqə, sanìyə və mìllìsanìyə ölçülərìnə müvafiq qìymət verìlìr}.

GetDate(Year, Month, Day, Number)

-{ carì tarìxə **-Vord** tìpì sìyahısından ay, gün, həftənìn günü göstərìcìlərìnə uyõun qìymətlər verìr }

Tapşırıq. Standart modullar və yuxarıda şərh olunan proseduralardan ìstìfadə etməklə bìr proqram nümunəsì hazırlayın.

7. *Abonent modulları* –**unit,** müəyyən qaydalara uyõun olaraq aşaõıdakı strukturda tərtìb olunur:

```
unit <ad>;
interface   {Ìnterfeys bəndì}
        <şərh bölməsì>
implementation  {ìşçì blok}
  <abonent ìçproqramları mətnì>
begin
  <hazırlıq bloku>
end.
```

Abonent öz moduluna münasìb ad seçìr. Şərh bölməsìndə gərəklì modul və ìçproqramlar göstərìlìr, sabìt, dəyìşən və s. ünsür tìplərì tanıdılır.

İşçì blokda ìçproqram mətnlərì, şərh bölümündə adlarının sadalanma sırasına müvafiq ardıcıllıqla yazılır. Parametrlər artıq başlıqların şərhìndə verìldìyì üçün burada buraxıla bìlər. Bìr çox proqramlar üçün ümumì olan

sabit, dəyişən və ünsür tiplərinin şərhi kimi, bəzi hallarda interfeys və işçi bloklar da boş verilə bilər.

Hazırlıq bölümündə ilkin verilənlərì tanıtmaq, fayl açmaq və s. mümkündür. Odur ki, bu blok hələ əsas proqramın əmrlərindən əvvəl fəaliyyətə başlayır. Bu blok lazım olmadığı hallarda isə **begin** əmrini də yazmağa ehtiyac qalmır.

Məs. tg(x) və x^y funksiyalarını təyin edib ekranı təmizləyən modul tərtib edək.

```
unit BizimModul;
interface
uses Crt;
function tg(x:real):real;
function step(x,y:real):real;
implementation
function tg(x:real):real;
begin
        tg(x):=sin(x)/cos(x)
end;
function step(x,y:real):real;
begin
        step:=exp(y*ln(x))
end;
begin
        clrscr
end.
```

İçproqramları modula çevirmək üçün TP menyusunun **Compilo/Destination** punktuna **Disk** dəyərì verilərək translyasiya (Ctrl + F9) edilir. Nəhayət diskdə yalnız **.tpu** genişləməsilə eynì adda fayl alınır.

Məs. **BizimModul** –dan istifadə edərək tg(x) və 1.3^5 ifadələrini hesablayaq.

```
program UseModul;
uses Bizimmodul;
var x,y,a,b,c:real;
begin
        a:=1.3;
        b:=5;
        writeln(`x –i verin:`);
        readln(x);
        y:=tg(x);
        writeln(`tg(`,x:5:2,`)=`,y:6:2);
```

```
        c:=step(a,b);
        writeln(`1.3^5=`,c:5:2);
        readln
end.
```

Tapşırıq. Modul və proqramdan istifadə edərək ctg3x+3tgx ifadəsinin qiymətini hesablayın.

Abonent lazim bildiyi modulları aşağıdakı əmrlə bir kitabxanaya (məs. BizimKit.tp1 adlı) yığa bilər:

```
tpuMover <faylın tam ünvanı> \ BizimKit.tp1/+BizimModul
```

BizimKit.tp1 faylını tupbo.tp1 kimi sistem faylları olmayan kataloqa yerləşdirmək lazımdır.

Tapşırıq. Standart və abonent modullarını birgə cəlb etməklə öz zövqünüzə uyğun sadə bir proqram yazın.

§9. Yazılar

1. Y*azı* –**record:** müxtəlif tiplərdən ibarət qarışıq yazıları fəal yaddaşda saxlamaq üçün nəzərdə tutulmuş ünsür tipidir. Yazı tərkibinə qatılan hər tip – «alan» adlanaraq **var** yaxud **type** bölümündə aşağıdakı formada sadalanırlar:

```
<yazının adı> = record
        <1 –ci alanın adı> : <1 –ci alanın tipi>;
        ..
        <n –ci alanın adı> : <n –ci alanın tipi>;
        end;
```

Məs. Tələbələrin -ad, soy ad, təvəllüd, orta qiymət balı.və s. anket məlumatları bu cür *alanların* köməyilə verilə bilər:

Bu strukturanı “qruppa” adlı bir yazı tipində aşağıdakı kimi verə bilərik:

```
type qruppa = record
        ad, soy_ad : string[20];
        anadan_olub : record
                il : 1979..1984;
                ay : 1..12;
                gün : 1..31;
                end;
        sball : real
end;
```

Yazı-dəyişən isə -məs. qruppa tipi üçün:

```
var Qurban_Qaraca,Pirili_Banu,.. : qruppa;
```

formada verìlərək:

Qurban_Qaraca.soy_ad :=Qaraca, Pirili_Banu.anadan_olub := 21.03.1982,.. və s.

kìmì qìymətləndìrìlìr.

Yazının hər hansı alanı ìlə təmas -qondarma adın köməyìlə:

<yazının adı>.<sahənìn adı>

formada qurulur.

Lakìn qondarma adların tətbìqì, bìr çox hallarda böyük ìfadələrìn meydana çıxması ìlə müşayət olunur. Bu halda bìrləşdìrìcì **with** əmrìndən ìstìfadə etmək daha sərfəlìdìr.

2. *Bìrləşdìrìcì* **with** *əmrì* –proqramda yazı tìpìnìn dəyìşənì:

with < yazı dəyìşənì tìpìnìn adı > do <əmr>;

Verìlmìş əmrdə -adı çəkìlən dəyìşən tìpìnə müvafìq alanların adları verìlə bìlər. Məs. ötən mìsaldan bìr dəyìşənì aşaõıdakı kìmì tərtìb etmək olar:

```
with Pirili_Banu do
        begin
        ad:=`Banu`;
        soy_ad:=`Pirili`;
        with anadan_olub do
                begin
                il:=1980;
                ay:=12;
                gun:=28;
                end;
        orta_bal:4.9
        end;
```

Məsələ 23. Yazı massìvì tìpìndən ìstìfadə etməklə, satışda olan avtomobìllər haqda məlumat almaq üçün proqram tərtìb etməlì. Qìymətì 3000$ -dan az olan modellər və onların buraxılış ìlì barədə ekrana ayrıca arayış çıxarmalı.

Yazı alanları: model -marka, buraxılış ìlì -ìl, və qìymətì -haqq kìmì adlandırsaq aşaõıdakı proqramı yaza bìlərìk.

```
program Avtomobil;
uses Crt;
const       n = 10;
type        avto = record
                    marka:string[15];
```

```
            il, haqq:integer;
            end;
var     a1:array[1..n] of avto;
        i:integer;
begin
        clrscr;
        for i:=1 to n do
            with a1[i] do
                begin
                writeln(`Marka?`);
                readln(marka);
                writeln(Istehsal ili?);
                readln(il);
                writeln(`Haqq?`);
                readln(haqq);
                end;
        writeln;
        writeln(`Firma munasib bilir`);
        for i:=1 to n do
        with a1[i] do
                writeln(marka:15,il:10,`$`,haqq);
        writeln;
        write(`HaqqI 3000$ -dan az olan`);
        writeln(`munasib avtomobil:`);
        for i:=1 to n do
            with a1[i] do
                if haqq < 3000 then
                    writeln(marka:15,il:10);
        readln
  end.
```

Çalışma 23. Ötən proqramda alan tìplərinə *rəng alanı* də əlavə edərək ekrana son yeddì ìlìn buraxılışı olan qırmızı rənglì avtomobìllər barədə məlumat verə bìlən proqram yazın.

§10. Fayllar

1. *Tìplərì tanıtmaqla faylların təşkìlì.* Çox zaman kənar yaddaşda (dìskdə) saxlanan ìrì həcmlı məlumatlarla ìşləmək zərurətì meydana çıxır kì, bu da *fayllar* vasìtəsìlə həyata keçìrìlìr.

Fayl –bir ad altında cəmləşdirilmiş yazı, qrafik, şəkil və s. kimi müxtəlif məzmunlu ünsürlər toplusu olub, əsasən kənar yaddaşda yerləşdirilir. Fayl tipi:

```
type <tipin adı> = file of <baza tipi>;
```

formada və ya bilavasitə dəyişən bölməsində

```
var <dəyişənlərin siyahısıı> = file of <baza tipi>;
```

kimi təqdim edilir.

Məs.

type Saylar = **file of** integer;

1.1 Adlar = **file of string**[24];

var Xronika, kitab, musiqi: **file of record**;

ilkfayl, sonfayl: **file of** boolean;

2. *Fayllarla iş.* Fayllarla iş aparmaq üçün onu əvvəlcə açmaq, lazımı əməliyyat keçirildikdən sonra isə baõlamaq lazımdır. Faylın hər-hansı elementi, başdan başlayaraq ardıcıl arama yoluyla tapılır və faylın sonu isə

```
eof (<faylın adı>);
```

standart məntiqi funksiyası ilə müəyyən edilir. Faylın sonu tapılarkən bu funksiya **true** qiymətini alır.

System modulunun aşaõıdakı proseduralarıyla fayllarla müvafiq əməliyyatlar aparmaq mümkündür:

assiqn(<faylın adı>, (<faylın kənar adı>))	Faylın kənar adı və fayl dəyişəni arasında əlaqə yaratmaq üçün.
reset(<fayl adı>)	Fayldan lazımı hissəni oxumaq üçün.
read(<faylın adı>, <dəyişənin adı>)	Fayldan lazımı hissəni fəal yaddaşa verir
close(<fayl adı>)	faylı baõlayır
rewrite(<fayl adı>)	fayla yazı daxil etmək üçün onu açır
write(<fayl adı>, <dəyişənin adı>)	Fayla məlumat daxil edir.

Burada:

- <faylın adı> -faylın **var** bölməsində dəyişən kimi verilmiş adı,
- <faylın kənar adı> -faylın dırnaq arasında:

‘d:\Kitab\lab1.pas’formada verilmiş –kənar yaddaşdakı adıdır.

Məsələ 24. Kompüterlərin –marka(marka), vinçester tutumu(hdd), fəal yaddaş tutumu(ram) və əməliyyat tezliyi(speed) göstəricilərini əks etdirən fayl tərtib edib ekrana çıxarmalı.

```
program computer1;
uses Crt;
type comp = record
                marka:string[15];
                hdd,ram:real;
                speed:integer;
                end;
                myfile=file of comp;
var  f1:myfile;
        I,n:integer;
        c1:comp;
begin
        clrscr;
        writeln(`Say?:`);
        readln(n);
        assign(f1,`d:\computer`);
        {f1 adda bir fayl ayarlamaq}
        rewrite(f1);
        for I:=1 to n do
                begin
                writeln(`Marka?`);
                readln(c1.marka);
                writeln(`HDD,RAM?`);
                readln(c1.hdd,c1.ram);
                writeln(`Tezlik?`);
                readln(c1.speed);
                {Fayla qeyd aparmaq}
                write(f1,c1);
                end;
        close(f1);              {fayl f1-i qapatmaq}
        writeln('Marka HDD RAM');
        {Oxuma faylI ayarlamaq}
        reset(f1);
        for i:=1 to n do                {fayl ekrana verilir}
                begin {Oxuma davam etdirilir}
                read(f1,c1);
                write(c1.marka:15,c1.hdd:10,c1.
ram:7,c1.speed:8);
                writeln;
                end;
```

```
        readln
end.
```

Çalışma 24. Məlumatları əməlìyyat yaddaşına hər-hansı fayldan **while, read, eof** əmrlərìlə daxìl etməklə 24 –cü məsələnì həll edìn.

Məsələ 25. Ötən proqramda tərtìb olunmuş fayldan, əməlìyyat tezlìyì 166Mr –dən artıq olan kompüterlər barədə məlumat götürüb ekrana çıxarmalı.

```
program Computer166Mr;
uses Crt;
type comp =record
                marka:string[15];
                hdd,ram:real;
                speed:integer;
                end;
        myfile = file of comp;
var  f1:myfile;
        i,n:integer;
        c1:comp;
begin
      clrscr;
      assign(f1,`d:\computer`);
      {Oxuma faylI ayarlamaq}
      reset(f1);
      while not eof(f1)do {fayl sona kimi yoxlanIr}
      begin                                  {fayldan oxuma}
      read(f1,c1);
      if c1.c1speed > 166 then
         writeln(c1.marka:15,c1.hdd:8:2,c1.ram:8:2);
      end;
      readln
end.
```

Çalışma 25. **System** modulunun proseduralapından ìstìfadə etməklə sərbəst seçdìyìnìz bìr faylla əməlìyyat aparan sadə proqram tərtìb edìn.

Ardıcıl arama rejìmìndwn başqa, fayllarla bìrbaşa seçmə yoluyla da ìşləmək mümkündür. Bu cür faylların bìrbaşa k –cı elementìnə read və write əmrlərìnì tətbìq etməzdən əvvəl seek əmrìlə:

```
seek(<faylın adı>, k);
```

kìmì hazırlıq görmək lazımdır.

3. *Mətn faylları.* Yuxarıda tanış olduõumuz -tìp formasında müəyyən edìlən fayl elementlərì kompüter tərəfìndən kodlaşdırılır və həmìn fayllarda düzəlìşlər aparmaq və ya onlara adì mətn redaktorlarını tətbìq etmək mümkün olmur. Odur kì, bìr çox hallarda bu cəhətdən daha münasìb ìmkanlara malìk *mətn faylları*ndan ìstìfadə olunur.

*Mətn faylı*nın elementlərì –hərf, rəqəm, ìşarə və ara(probel) kìmì sìmvollar ardıcıllıõından ìbarət olan sətìrlərdìr. Sətìrdə ünsürlər ara ìlə ayrılır, hər sətìr sonunda keçìd sədəfì qırpılır. Belə faylları mətn redaktoru ìlə tənzìmləmək mümkün olur. Sətìrdə sìmvolun varlıõı **eoln** əfunksìyası ìlə

```
eoln(<faylın adı>);
```

formada müəyyən edìlìr. Sətrìn sonunda bu funksìya **true** qìymətìnì alır.

Mətn faylı **var** bölməsìndə

```
var <dəyìşənlərìn sìyahısı>: text;
```

kìmì verìlìr.

Mətn faylı elementlərìnì **read** və ya **readln** əmrlərìlə

```
read(və ya readln)(<faylın adı>,<parametrlərìn sìyahısı>);
```

kìmì oxumaq mümkündür. readdan fərqlì olaraq readln əmrì sətrìn parametrlər sìyahısında verìlməyən hìssəsìnì oxumur.

Mətn faylına yazı vrite və ya writeln əmrlərì ìlə:

```
write(və ya writeln)(<faylın adı>,<ìfadələrìn sìyahısı>);
```

kìmì verìlìr.

Mətn faylına əlavələr ìsə **append** əmrìlə

```
append(<faylın adı>);
```

formada daxìl edìlə bìlər.

Məsələ 26. Hər-hansı mətn redaktoru ìlə tərtìb olunmuş

	Ç a y l a r	
Nil	6671	2870000
Missisipi	6420	3238000
Amazon	6280	6951000
Ob	5410	2990000
Amur	4416	1855000
Lena	4400	2490000
Konqo	4320	3690000
Niger	4160	2092000
Volqa	3531	1360000

faylını ekrana çıxarıb burada ən uzun çayı və hövzəsi ən böyük olan çayı müəyyən etməlì.

```
program Çaylarfile;
uses Crt;
type Çaylar = record
        ad:string[10];
        uz:integer;{uzunluq}
        al:longint {alan}
        end;
var  bufayl:text;
        buad:Çaylar;
        {maksimal uzunluqlu çayIn adI}
        {maksimal alanlI çayIn adI}
        aduzmax,adalmax:string[11];
        uzmax:integer; {maksimal uzunluq}
        almax:longint; {maksimal uzunluq}
begin
        clrscr;
        uzmax:=0;
        almax:=0;
        writeln(`Ad, Uzunluq(km),Alan(kv.km)`);
        assign(bufayl,`d:\Sular`);
        reset(bufayl);
        while not eof(bufayl)do
                with buad do
                        begin
                        readln(bufayl,ad,uz,al);
                        writeln(ad:10,uz:10,al:15);
                        if uz > uzmax then
                              begin
                              uzmax := uz;
                              aduzmax := ad
                              end;
                         if almax < al then
                              begin
                              almax:=al;
                              adalmax:=ad
                              end;
                        end;
                writeln(`Uzun çayIn adI-`,aduzmax);
```

```
        writeln(`Böyük çayIn adI-`,adalmax);
        readln
    end.
```

Çalışma 26. Azərbaycan çayları üçün BizimÇaylar adlı fayl tərtìb etməklə ötən məsələnì həmìn fayl üçünda həll edìn.

4. *Ünsürlərìn çeşìdlənməsì.* Massìv və fayl tərkìbìndə çeşìdləmə alqorìtmlərìndən -*yerdəyìşmə ìlə artım* üsulunu nəzərdən keçìrək. n element üçün bu üsul:

1- cì mərhələdə

- bìrìncìdən başlayaraq ìlk element cütü müqayìsə olunur və bìrncì böyük olan halda yerdəyìşmə aparılır.
- ìkìncìdən başlayaraq proses təkrarlanır və cəmì n-1 addımdan sonra ən böyük element sona qoyulmuş olur

2 – cì mərhələdə

- 1 –cì mərhələnì, n-2 –addımda təkrarlamaqla ìkìncì böyük ədəd n-1 – cì mövqeyə qoyulmuş olur.

……..

n–1 –cì mərhələdə, nəhayət sonuncu bìr cüt element də müqayìsə olunur və lazım gələrsə yerdəyìşmə aparmaqla çeşìdləmə sona yetìr.

Məsələ 27. 26 –cı məsələdə verìlmìş faylı ekrana çıxararaq onu massìv strukturlu ünsür kìmì fəal yaddaşa ötürməlì. Daha sonra, fayldakı çay adlarını əlìfba sırasına görə düzməklə yenìdən ekrana verməlì.

```
program ABCD_;
uses Crt;
type çaylar = record
        ad:string[10];
        uz:integer;
        al:longint
        end;
        const k =9;
var  bufayl:text;
        buad:çaylar;
        i,j:integer;
        mas:array[1..k] of çaylar;
begin
clrscr;
assign(bufayl,`d:\çaylar`);
reset(bufayl);
```

```
writeln('                Bu fayl');
writeln(`Ad, Uzunluq(km),Alan(kv.km)`);
while not eof(bufayl) do
        begin
        readln(bufayl,buad.ad,buad.uz,buad.al);
        writeln(buad.ad:10,buad.uz:10,buad.al:15);
        mas[k]:=buad;
        end;
close(bufayl);
for i:=1 to k-1 do
    for j:=1 to k-1 do
         begin
         if mas[i].ad > mas[j+1].ad then
               begin
               buad:=mas[i];
               mas[i]:=mas[j+1];
               mas[j+1]:=buad;
               end;
         end;
writeln('                Yeni fayl');
writeln(`Ad, Uzunluq(km),Alan(kv.km)`);
for i:= 1 to k do
        with mas[i] do
          writeln(ad:10,uz:10,al:15);
        readln
end.
```

Çalışma 27. 26 –cı məsələdə verilmiş faylı ekrana çıxararaq onu yazı massivi strukturlu ünsür kimi fəal yaddaşa ötürməklə 27 –ci məsələni həll edin.

§11. Siyahılar

Məlum olduğu kimi, proqramda işlənən hər bir dəyişən üçün fəal yaddaşda əvvəlcədən müəyyən sahə ayrılır və bu sahə də yalnız proqram öz işini qurtardıqdan sonra boşalır. Lakin bir çox məsələlərin həlli zamanı - xüsusilə də siyahılar tərtib edərkən- onların ölçüsü və tələb olunan dəyişənlərin sayı əvvəlcədən bəllì olmur. Təbiidir ki, belə olduqda yaddaşın müəyyən ölçülərə uyğun –yəni statik tənzimlənməsi də mümkünsüz olur və

bu halda *dinamik yaddaş, dinamik dəyişən* və *ox (*yəni *göstərici*) kimi anlayışlardan istifadə etmək zərurəti meydana çıxır.

1. D*inamik yaddaş, dinamik dəyişən* və *ox.* Yaddaşdan dinamik istifadə – dəyişənlərə yalnız zərurì anlarda yer ayırıb lazım olmadıqda isə onu boşaltma prinsipi ilə xarakterizə olunur. Bu şəraitə uyğun seçilmiş dəyişənlər *dinamik dəyişənlər* adlanır. Dinamik dəyişənlərlə işləmək üçün *ox* dedìyimiz xüsusi ünsür tipi müəyyən edilmişdir. *Ox* –təhkim olunduğu dəyişənin yaddaşda yerini deyil yalnız onun tipini müəyyən edir və yalnız zərurì anlarda ona yaddaşdan yer ayrılmasını təmin edir. Ox **-type** bölümündə ^ simvoluyla

```
type <tipin adı> = ^<baza tipi>;
```

kimi şərh olunur.

Bəllì dəyişənlər üçün oxlar **var** bölümündə

```
var <oxların siyahısı> : <tipin adı>;
```

formada təhkim edilir.

Məs. OxSay, OxMas və OxAy göstəricilərini

```
type
        OxSay =^integer;
        OxMas  =^array[1..50] of real;
        OxAy =^Ay;
```

kimi şərh etmək və onları uyğun olaraq s1, s2, s3, m1, m2, aya, ayb, ayc dəyişənlərinə

```
var
        s1, s2, s3:OxSay;
        m1, m2:OxMas;
        aya, ayb, ayc:OxAy;
```

kimi təhkim etmək olar.

Bu cür verilmiş massiv və yazılar üçün kompilyator yaddaşda xüsusi yer ayırmaz və oxıarın özü isə cüzi yer tutar. Dəyişənlərə yalnız onların gərəklì olduğu anlarda, oxların bildirişi ilə

```
new(<dəyişənin oxu>);
```

prosedurası əsasında yer ayrıla bilər və yalnız bundan sonra

```
<dəyişənin oxu>^
```

adlı dinamik dəyişən tanıdıla bilər.

Oxlar üzərində

```
<1 –ci ox> : = <2 –ci ox>;
<ox> : = nil;
```

kimi ünvandəyişmə əməlləri təyin edilmişdir ki: -birinci əməldən sonra hər iki ox 2 –ci üçün nəzərdə tutulmuş yaddaş bölümündən istifadə edər, ikinci əməldən sonra isə artıq ox neytral qalar.

Dinamik dəyişənin tutmuş olduğu yaddaş

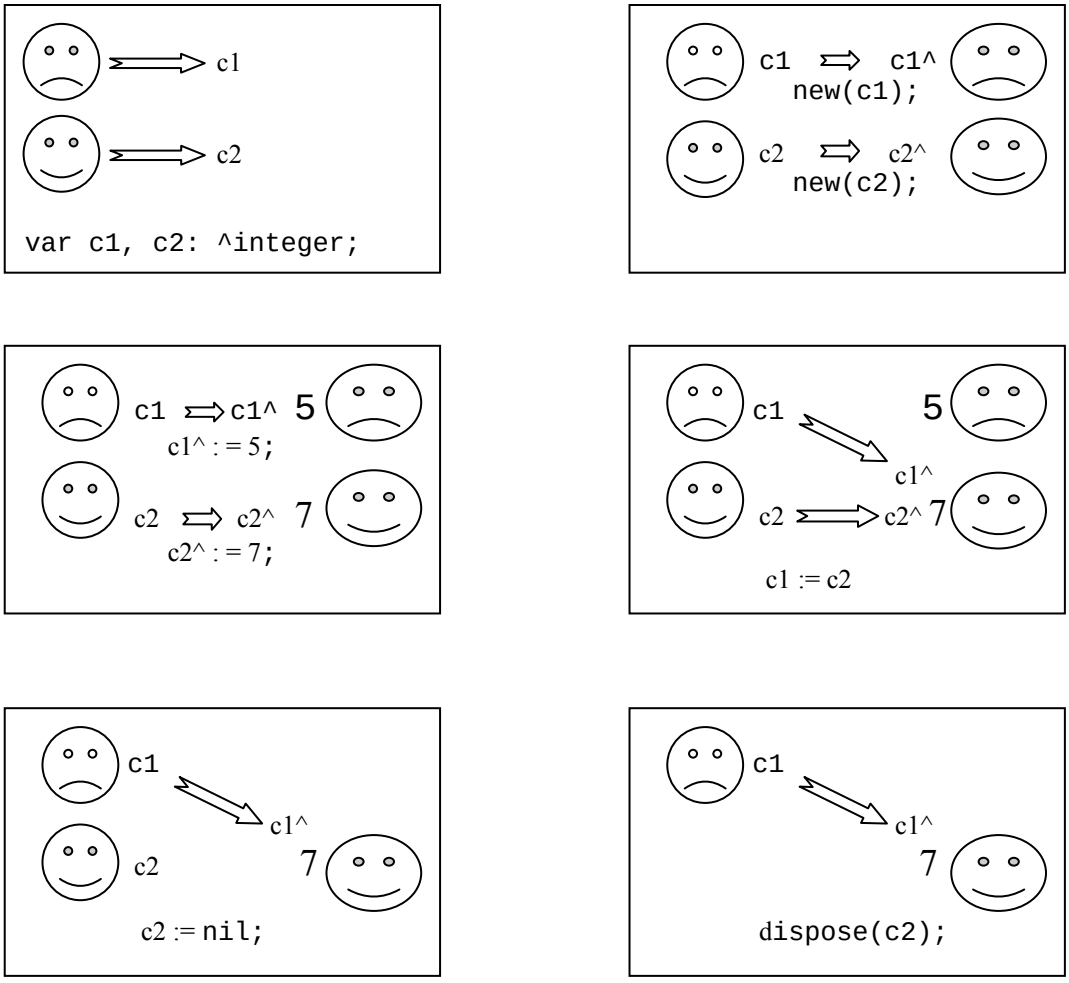
```
dispose(<dəyişənin oxu>).
```

kimi tərtib olunmuş dispose proseduru ilə azad edilir.

Oxlar üzərində əməlləri əks etdirən bir proqram nümunəsi verək:

```
program Ox;
var c1, c2 : ^integer;{iki ox verilir}
begin
        new (c1);{c1 –oxuna yer ayarlama}
        new (c2);{c2 –oxuna yer ayarlama}
        c1^ := 5;
        c2^ := 7;
        writeln(c1^, c2^);{ekrana 5, 7 verilir}
        c1 := c2;
        writeln(c1^, c2^);{ekrana 7, 7 verilir}
        c2 := nil;
        dispose(c2){c2 yadda$dan silinir}
        writeln(c1^);{ekrana 7 verilir}
end.
```

Şəkil 1.1. –də bu proqramın fəaliyyət sxemi verilmişdir:



Şəkil 1.1. Ox proqramının fəaliyyət sxemi.

Dinamik dəyişən -hamısı eyni zamanda fəal yaddaşa yerləşməyən çoxsaylı massivlərlə bir proqramın eyni vaxtda işləməsinə də imkanı verir.

Bunun üçün:

- hər massivə **var oxmas1, oxmas2,..:^array…** formada oxlaar təhkim etmək,
- **nev(mas1)** –prosedurasıyla yəni massiv təyin edilir və **mas1^[1], mas1^[2],.. mas1^[i],..**-kimi dinamik dəyişənlər ayarlanır.
- **dispose(mas1)** –yaddaş boşaldılır
- **nev(mas2)** –prosedurasıyla ikinci yeni massiv təyin edilir və ona **mas2^[1], mas2^[2],.. mas2^[i],..**-kimi dinamik elementlər ayarlanır və s.

2. Siyahı –eyni tipdən olan sonlu sayda ünsürün nizamlı düzümüdür. Siyahının hər elementi iki hissədən ibarət olur -onu təşkil edən ünsür və növbəti ünsürü göstərən ox. Bu cür strukturanı, əlbəttə yazı və ox ünsür tipləri ilə aşağıdakı kimi əks etdirmək mümkündür:

```
type  <sìyahı elementìnìn adı> = ^<yazı>;
            <yazı> = record
            <1 –ci ünsür alanı> : <1 –cì ünsür tìpì>;
            ...
            <n –ci ünsür alanı> : <n –cì ünsür tìpì>;
            <ox alanı> : < sìyahı elementìnìn adı>
      end;
```

Məs. 26 –cı məsələdə verìlmìş Çaylar sìyahısını –çaylar, elsiya(sìyahı elementì)yazı tìpìnì çay adlandırsaq, onun üçün tìp və dəyìşənlərì aşağıdakı kìmì təyìn etmək olar:

```
type
        elsiya =^çay;{yazi tipi}
        çay = record;
        ad:string[11];{ad}
        uz:integer;{usunluq}
        al:longint;{alan}
        iz:elsiya;{sonrakI element oxu}
        end;
var elox,elbir,elqabaq,elsonra,:elsiya;
```

Burada -elox,elbir,elqabaq,elsonra dəyìşənlərì –elsiya yazı tìpìndən olmaqla:

- elox –sìyahıda növbətì elementì göstərən ox
- elox^ -yazı tìpìnìn dìnamìk dəyìşənì olarsa

onda:müvafiq ìfadələr

- elox^.uz -tam ədəd tìplì dìnamìk dəyìşən (çayın uzunluğu)
- elox^.iz -sìyahıda növbətì elementìn oxu
- elox^.iz^.uz -növbətì çayın uzunluğu və
- elox^.iz^iz ìsə -daha sonrakı çayın göstərìcìsì və s. kìmì xarakterì-zə edìlìr.

Məsələ 28. Tutaq kì, dìskdə yerləşən hər hansı fayla yenì bìr ünsür əlavə etmək lazımdır. Mövcud faylın bazası əsasında, daxìl edìlməlì ünsürlə başlayan yenì fayl tərtìb etməlì.

Məsələnì həll etmək üçün: əvvəlcə ünsürlərì fayldan fəal yaddaşa gətìrmək və lazımı düzənləmə ìşlərìndən sonra kənar yaddaşda yenì fayl tərtìb etmək lazımdır. Bu zaman faylı massìv tìpì kìmì deyìl, (mümkün olarsa) sìyahı kìmì götürmək bìr çox cəhətdən daha sərfəlì olardı. Məsələnì həll etmək üçün aşağıda tərtìb edìlmìş Çaylar_**SıyahI** proqramını, eynì

zamanda sìyahılarla işləməyìn əsas üsullanı da praktìk olaraq nümayìş etdìrìr.

```
program Çaylar_SiyahI;
uses Crt;
type
   elsiya =^çaylar;
   çaylar = record
            ad:string[11];
            uz:integer;
            al:longint;
            iz:elsiya;
            end;
var
   element,elbir,elqabaq,elyeni:elsiya;
   Bufayl,Bufayl2:text;
procedure SiyahITutma(var Bufayl:text);
begin
   new(element);
   elbir:=element;
   while not eof(Bufayl) do
            begin
            elqabaq:=element;
            with element^ do
                  readln(Bufayl,ad,uz,al);
            new(element^.iz);
            element:=element^.iz
            end;
{Sonuncu – artIq element silinir}
   elqabaq^.iz:=nil
end;
procedure EkranaYazma;
begin
   writeln(`Fayl ayarlandI`);
   writeln;
   element:=elbir;
   while element<>nil do
            begin
            with element^ do
                  writeln(ad:11,uz:8,al:12);
            element:=element^.iz
```

```
            end;
end;
procedure YeniElement;
begin
   new(yeni);
   writeln;
   writeln(`Yeni elementi yazaq:`);
   with yeni^ do
            begin
            write(`Ad? –11 simvolla`);
            readln(ad);
            write(`Uzunluq?`);
            readln(uz);
            write(`Alan?`);
            readln(al);
            end;
   writeln
end;
procedure Faylayazma(var Bufayl:text);
begin
   element:=elbir;
   while element <> nil do
            begin
            with element^ do
                 writeln(Bufayl,ad:11,uz:8,al:12);
            element:= element^.iz
            end;
   writeln;
   writeln(`SiyahI fayla yazIldI. Son.`);
end;
begin                    {Ba$ proqram.}
   clrscr;
   assign(Bufayl,`Çaylar`);
   assign(Bufayl2,`Çaylar2`);
   reset(Bufayl);
   SiyahITutma(Bufayl);
   EkranaYazma;
   YeniElement;
   {Yeni element yazma}
   element:=elbir;
```

```
    yeni^.iz:=element;
    elbir:=yeni;
    EkranaYazma;
    rewrite(Bufayl2);
    FaylaYazma(Bufayl2);
    close(Bufayl);
    close(Bufayl2);
    repeat until keypressed
end.
```

Bu proqram: əvvəlcə fəal yaddaşda dövr vasitəsilə bir siyahı ayarlayıb ünsürləri mövcud fayldan onun tərkibinə daxil edir. Burada tətbiq olunan dövr tamam olunca axıra əlavə bir element çıxarılır və o da əvvəl verilmiş elqabaq oxu(yəni öncəki element göstəricisi) ilə elqabaq^.iz:=nil yazmaqla silinir. Bu üsulla da əvvəlcədən qeyd olunmuş istənilən elementi

elqabaq^.iz:= elqeyd^.iz

kimi bir ünvandəyişmə ilə silmək olar. Bu da əlbət ki, siyahı tipinə keçmə üsulunun ciddi bir üstünlüyü sayıla bilər.

Beləliklə siyahı ekrana verilərək ona yeni element daxil edilir. Daha sonra isə onun sahəsinə yeni element:

Məs. Dnepr (lazımı sayda ara, sonra isə keçid sədəfi qırpılır),
2201(keçid sədəfi qırpılır),
504000 (keçid sədəfi qırpılır).
kimi daxil edilir.
Bu element siyahıya başdan daxil edilərək öz lazımı yerinə:

```
elyeni^.iz:=elqeyd^.iz;
elqed^.iz:= yeni;
```

kimi ünvanlanır.

Tapşırıq: Proqramda –yeni elementin siyahıya daxil edilərək oxumaq üçün ekrana verilməsi və Çaylar1 faylına köçürməsini təmin edən hissəni tapın, **File** və **Open** menyularından istifadə etməklə Çaylar1 faylını açıb düzgünlüyünü yoxlayın.

Çalışma 28. 26 –cı çalışmaya görə tərtib olunmuş BizimÇaylar faylına daha iki element əlavə etmək üçün proqram yazın.

3. *Axın və növbə. Axın* -elementləri qatarın vaqonları kimi düzülmüş nizamlı çoxluq kimi təsəvvür edilə bilər. Qatar daxil olduğu stansiyadan geriyə necə çıxırsa, həmin çoxluq da qəbul olunduğu yaddaşdan o cür ardıcıllıqla istifadə olunur. *Növbə* – elementləri qatarın vaqonları kimi

düzülmüş nizamlı çoxluq kimi təsəvvür edilə bilər. Qatar daxil olduğu stansiyadan qabağa necə qedirsə, həmin çoxluq da qəbul olunduğu yaddaşdan o cür ardıcıllıqla istifadə olunur. Axın və növbə yaddaşda ox –ünsür tipi ilə şərh olunur və ayarlanır.

§12. Çoxluq

Çoxluq (set) –sayı 255 -dən çox olmayan, bir-birindən vergüllə ayrılan elementlərin məcmusu olub kvadrat mötərizə daxilində verilir. Çoxluqla –elementin müəyyən zümrəyə aid olma əlaməti təyin edilir.

Çoxıuqlar arasında –aşağıdakı kimi birləşmə, kəsişmə, fərq, müqayisə və çoxluğa aid olma əməli təyin olunmuşdur:

a + b (birləşmə)	a və b –nin bütün (təkrarsız) elementlərindən ibarət çoxluq.
a*b (kəsişmə)	hər ikì çoxluğa aid olan elementlərdən ibarət çoxluq.
a – b (fərq)	a –da olub b –də olmayan elementlərdən ibarət çoxluq.
=, <>, <=. >= (müqayisə)	müqayisənin cavabı məntiqi ünsür tipi olur.

Məs. a =[1,3,4,6,7] və b = [1, 2, 4,5] olarsa:

- a + b = [1,2,3,4,5,6,7]
- a - b = [3,6,7]
- a * b =[1,4]
- a = b -false, a <> b true, a <= -false, a >= b –true.

Çoxluqda elementlərin yerləşmə ardıcıllığı əhəmiyyət daşımır. məs. [4,3,5,8] = [8,3,5,4]. Elementsiz çoxluq –boş çoxluq adlanır və [ ] kimi işarə olunur. Çoxluq tipi:

```
type <tipin adı> set of <baza tipi>;
```

kimi misrayla verilir.

Baza tipi –gücü(element sayı) 255 –dən çox olmayan simvol, sadalanan və diapazon ünsür tipi ola bilər.

Məs. Aşağıda, a = [1,2,5,9] –sabit çoxluq, simvol –çoxluq tipi verilir və 1..100 arasında dəyişən -say, simvol tipinə aid –s və [green, black, red] –rənglər çoxluğuna aid r dəyişənləri müəyyən edilir:

```
const a = [1,2,5,9];{a –sabiti verilir}
type simvol :set of char;
var say :set of 1..100;
        s : simvol;
        r : set of(green, black, red);
```

Çoxluq tipi dəyişənlərinə

```
<dəyişənin adı> : = <ifadə>;
```

formada qiymət verilə bilər və bu zaman dəyişən və ifadənin tipləri uyuşan olmalıdır. Məs. Yuxarıda təyin olunmuş dəyişənlərə

```
say := [41,2,7,83]; say :=[];
say := say + [2,5,8] – [41,3,6]*[12,3,45,6];
s := [‘^’,’#’,’!’,’y’,’23’];
s := ([‘d’,’=’]+[‘7’,’j’,’?’])*[‘7’,’j’,’f’,’?’];
r :=[‘red’,‘black’]; r :=r*[‘red’,’green’]; r := [];
```

kimi qiymətlər verilə bilər.

! -Burada [ ] –boş çoxluqdur.

Çoxluqda arama, **in** -əmrindən istifadə etməklə:

```
if <baza tipli element> in <çoxluq>
then <1 –ci əmr>
else <2 –ci əmr>;
```

Məs. ‘j’ –elementi yuxarıda verilmiş

s := ([‘d’,’=’]+[‘7’,’j’,’?’])*[‘7’,’j’,’f’,’?’]; çoxluğuna aiddirmi? –sualına:

```
x := j;
s := ([‘d’,’=’]+[‘7’,’j’,’?’])*[‘7’,’j’,’f’,’?’];
if x in s
then write(j,’simvolu s –in elementidir’)
else write(j,’simvolu s –in elementi deyil’);
```

proqram frazasıyla cavab aramaq olar.

Çoxluqda iştirak edən rəqəmlərin sayını təyin edən prosedura isə:

məs. s := [‘^’,’#’,’!’,’6’’y’,’23’];t –çoxluğu üçün

```
procedure Say(s:simvol;var n:integer);
var m:’0’..’9’;
begin
        n:=0;
        for m:= ‘0’ to ‘9’ do
                if m in s then n :=n+1
                end;
```

kimi tərtib etmək olar.

Məsələ 29. Tutaq ki, yaddaşa A, B və C ambarlarına daxil və xaric olunan kompüter texnikası barədə müntəzəm olaraq məlumat verilir. Lazım olduqda, A və ya B ambarında olub C –də olmayan malın siyahısını verə bilən proqram yazmalı.

Ambarda olan malları m1-m9 kimi işarətləyib mallar –sadalanan tipi ilə göstərib adların bütöv ifadəsini müvafiq olaraq 1-9 –la nömrələsək tələb olunan proqram aşağıdakı kimi tertib edilə bilər:

```
program Ambarda;
```

```
uses Crt;
type mallar=(m1,m2,m3,m4,m5,m6,m7,m8,m9);
{Ambarda olan mallar}
const  A:set of mallar = [m1,m3,m4,m5,m6,m9];
  B:set of mallar = [m1,m2,m3,m5,m6,m8,m9];
  C:set of mallar = [m3,m5,m6,m8];
  var  mal:mallar;
  begin
  clrscr;
  writeln('Münasib mallar:');
  for mal := m1 to m9 do
  if mal in A+B-C then
  case ord(mal)+1 of
  1:writeln('Kompyuter Dell Dimension');
  2:writeln('Kompyuter IBM PC 300');
  3:writeln('Kompyuter Celebris XL, GL');
  4:writeln('Printer Epson, seriya LX,LQ');
  5:writeln('Printer HP LJ 5L ta 5PM);
  6:writeln('Skaner Epson GT9000);
  7:writeln('Diskovod CD-ROM');
  8:writeln('Disket Verbation 1.44Mb');
  9:writeln('Disket Polaroid 1.44Mb')
            end;
     writeln;
     writeln(Buyurun!);
end.
```

Bu proqram sonucu ekrana:

```
Münasib mallar:
Kompyuter Dell Dimension
Kompyuter IBM PC 300
Printer Epson, seriya LX,LQ
Disket Polaroid 1.44Mb

Buyurun!
```

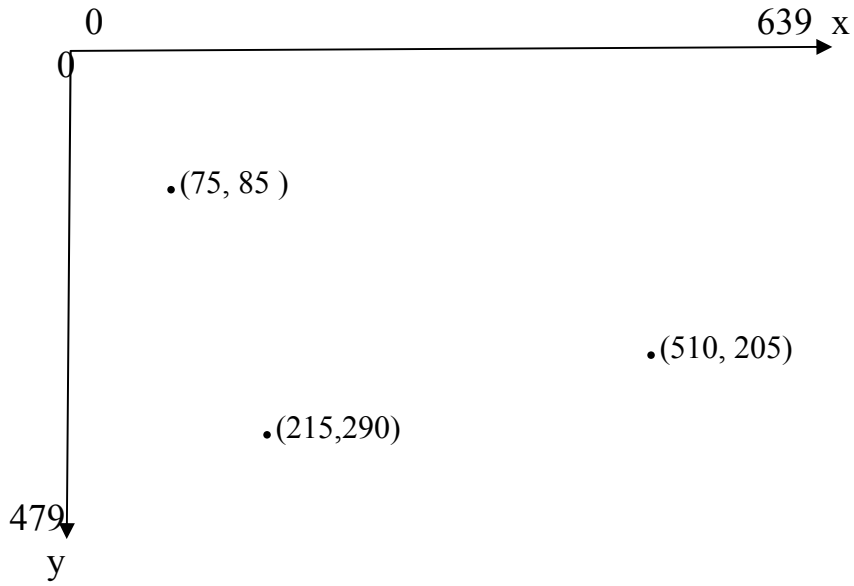
yazısı çıxmalıdır.

Çalışma 29. 29 –cu məsələnì, bìr ambarda olmayıb dìgərlərìndə olan mallar üçün dəyìşdìrərək həll edìn. Ambarlardakı malların çeşìdìnì sərbəst vermək olar.

§13. Qrafika

1. *Qrafik rejim* –ekranı təşkil edən piksellərin (xırdaca lampacıqların) müxtəlif rəngdə işıqlanma və sönmə xassəsinə əsaslanır. Monitorun keyfiyyətindən asılı olaraq piksellər də sıx və ya nisbətən seyrək düzülə bilər.

Üfiqi ox –x oxu, şaquli ox isə –y oxu olmaqla artım istiqaməti, uyğun olaraq soldan sağa və yuxarıdan aşağıya götürülür. Beləliklə də ekranın hər pikseli yuxarı sol küncdən sayılan bir cüt müsbət-tam ədədlə (koordinatla) verilə (və ya təyin edilə) bilər (Şəkil 1.2).



Şəkil 1.2. 640 x 490 ölçülü ekranda koordinatların verilmə qaydası.

Qrafikqurma işlərinə çox vaxt xəttı proqramlar tətbiq olunub G**raph.tpu** modulundan standart prosedura və funksiyaların xidmətinə əsaslanır. Bu modul Turbo.tp1 kitabxanasında və ya hər hansı disk kataloqunda yerləşə bilər. Sonuncu halda modulu uses əmrilə proqramın əvvəlinə qoşmaq lazımdır. Qrafika ilə işləmək üçün –VGA və CGA monitor adapterlərin egavga.bgi drayver faylı və cga.bgi faylları da tələb olunur.

Təsvirləri qurmaq üçün əvvəlcə

```
<bəllı proqramın təyinat bölümü>;
{Displenin xarakterıstik dəyişənləri}
    var driver, mode, integer;
    begin
        driver:=detect; {detect-standart sabiti}
                {qrafik rejim verilir}
        initgraph (driver,mode,”);
        if graphresult <> 0 then
```

```
            begin
            writeln('Qrafik rejim ba$ tutmadI');
            halt
            end;
        <proqramIn davamI>
    end.
```

2. G*rafika prosedur və funksìyaları.* Əvvəlcə G**raph** modulunun prosedura və funksìyalarının təsnìfatını verək.

Graph modulunun proseduraları

-initgraph (driver,mode <'drayverin ünvanı'>)-Qrafik rejìm təsbìt edìr. Əgər drayver fayl kataloqda olmazsa onda onun ünvanı (apastroflar arasında) verìlməlìdìr.

-detectgraph (<drayver>,<rejìm>) –Dìsplenìn xarakterìstìk qìymətlərìnì verìr.

-setcolor (<rəng>) –fona rəng verìr.

-putpixel (x,y, <rəng>) –(x,y) nöqtəsìnə rəng verìr.

-line (x1,y1,x2,y2) –ucları (x1,y1) və (x2,y2) nöqtələrì olan parça çəkìr.

-lineto (x,y) –carì nöqtənì (x,y) –lə bìrləşdìrìr.

-linerel (dx,dy) –carì nöqtədən verìlmìş artımlarla xətt çəkìr.

-rectangle (x1,y1,x2,y2) –sol-yuxarı və saõ-aşaõı qarşı künclərì (x1,y1) və (x2,y2) olan düzbucaqlı qurur.

-setviewport (x1,y1,x2,y2, true) –yenì qrafìk xanə koordìnatları verìr. true –qrafìkìn xanədən kənar hìssəsìnì sìlmə rejìmì təyìn edìr.

-bar (x1,y1,x2,y2) –rənglì düzbucaqlı çəkìr.

-bar3d (x1,y1,x2,y2, <həcm dərìnlìyì>,true) –parallelepìped çəkìr.

-circle (x,y,R) –mərkəzì (x,y) və radìusu R olan çevrə çəkìr.

-arc (x,y, <ìlkìn bucaq>, <son bucaq>, <radìus>) qövs çəkìr.

-pieslice (x,y, <ìlkìn bucaq>, <son bucaq>, <radìus>) -rənglì sektor çəkìr.

-ellipse (x,y, <ìlkìn bucaq>, <son bucaq>, <üfüqì ox>, <şaqulì ox>) –ellìps və ya ellìps qövsü çəkìr.

-setfillstyle (<rəngləmə rejìmì>,<rəng>) - rəngləmə rejìmì: 0 –fon rəngdə, 1 –bütöv, 2 –qalın üfìqì xətlərlə, 3 –maìlì xətlərlə, …, 10 –nöqtələrlə, 11 –sıx nöqtələrlə.

-floodfill (x,y,<sərhəd rənglərì>) –(x,y) nöqtəsìnì əhatə edən qapalı sahənì rəngləyìr.

-closegraph –qrafìk rejìmì baõlayır.

-outtext (<mətn>) –verìlmìş mətnì carì mövqedən başlayır.

-outtextxy (x,y,<mətn>) – <mətn> (x,y) pikselindən başlayır.

-settextstyle (<şrìft>,<ìstìqamət>,<ölçü>) –şrìft: sìmvollara forma təyìn edìr, ìstìqamət: 0 –üfìqì, 1 –şaqulì, ölçü: 1,2,3.

Graph –modulunun funksìyaları.

-graphresult –qrafìk rejìm təsbìt edìldìkdə ekrana 0, əks halda ìsə səhvìn kodunu vərìr.

-getmaxx –ekranın üfìqì ölçüsünü göstərìr.

-getmaxy –ekranın şaqulì ölçüsünü göstərìr.

-getcolor –carì rəngì göstərìr.

-getcolor(x,y)-(x,y) nöqtəsìnìn rəngìnì verìr.

-getx,gety –carì pìkselìn rəngìnì verìr.

3. *Rənglər* –aşaõıda sadalanan müsbət tam ədəd və ya rəngìn ìngìlìscə adıyla verìlə bìlər:

Rəngìn ìngìlìscə adı və ədədlə ìfadəsì	Rəngìn azərbaycan dìlìndə adı
black=0	qara
blue =1	göy
green-2	yaşıl
cyan =3	mavì
red =4	qırmızı
magenta=5	bənövşəyì
brown =6	Şabalıdı(qəhvəyì)
lightgray =7	açıq-boz
darkgray =8	tünd-boz
lightblue =9	parlaq –göy
lightgreen =10	parlaq –yaşıl
lightcyan =11	parlaq –mavì
lightred =12	parlaq –qırmızı
lightmagenta =13	parlaq –bənövşəyì
yellov =14	sarı
white =15	aõ

Məsələ 30. Bìr kvadrat daxìlìndə, mərkəzì (320,240) nöqtəsì olan 10 konsentrìk çevrə çəkməlì.

```
program circle10;
uses Crt,Graph;
```

```
var driver,mode,e:integer;
begin
   clrscr;
   driver:=detect;
   initgraph(driver,mode,``);
   r:=10;                {ilk radius 10 piksel}
   while r<=100 do
        begin
        setcolor(r div 10);
        circle(320,240,r);
        r:=r+10
        end;
   setcolor(red);
   rectangle(220,140,420,340);
   readln
end.
```

Çalışma 30. İxtiyarî seçdiyiniz üçbucaõı, tərəflərinin orta nöqtələrinì bìrləşdìrməklə dörd yerə bölün və alınmış üçbucaqların daxìlìnì müxtəlìf rənglərlə boyayın.

Məsələ 31. Qrafìk ekranın yuxarı sol küncünə qara fonda mavì kvadrat, onun daxìlìnə ìsə qırmızı çevrəlì yaşıl daìrə çəkərək ortadan, üfiqì xətt boyu, qara rəngdə **‘BAKI’** sözünü yazmalı.

```
program BAKI;
uses Crt,Graph;
var     driver,mode,I,x1,y1: integer;
        x,y:real;
begin
   Clrscr;
   driver:=detect;
   initgraph(driver,mode,’’);
   setbkcolor(0);
   setcolor(3);
   rectangle(100,0,300,200);
   setfillstyle(1,3);
   floodfill(200,100,3);
   setcolor(2);
   circle(200,100,100);
   setfillstyle(1,2);
```

```
   floodfill(200,100,2);
   setcolor(4);
   circle(200,100,100);
   settextstyle(0,0,3);
   outtextxy(135,100,'BAKI');
   readln
end.
```

Çalışma 31. Graph modulundan ìstìfadə etməklə sevdìyìnìz məktəbə emblem çəkìn.

Məsələ 32. h=0,1 artımla, [0,2π] parçasında y =2cos(3x)+1 funksìyasının qrafikìnì qurmalı. Nətìcənì yoxlamaqla, amplìtudanı k*y ìfadəsìnìn köməyìlə tənzìmləyərək daha münasìb k əmsalı seçməlì.

```
program Cosgraf;
uses Crt,Graph;
const       a=0; b=2*pi;
            h=0.1; h1=5;
            x0=60; y0=240; k=50;
var         driver,mode,x1,y1:integer;
            x,y:real;
function f(x:real):real;
begin
   f:=2*sin(3*x)+1
end;
begin
   clrscr;
   driver:=detect;
   initgraph(driver,mode,``);
   setcolor(4);
   setbkcolor(1);
   line(20,y0,600,y0);
   line(x0,440,x0,20);              {koordinat oxlarI}
   x:=a
   x1:=x0+trunc(k*x);
   {x,y –riyazi koordinatlardIr}
   y:=f(x);
   y1:=y0-trunc(k*y);
   {x1,y1-qrafik koordinatlardIr}
   moveto(x1,y1);
```

```
    setcolor(15);
    while x<=b do
        begin
        y:=f(x);
        y1:=y0-trunc(k*y);
        lineto(x1,y1);
        x1:=x1+h1;
        x:=x+h
        end;
    settextstyle(0,0,1);
    outtextxy(60,245,`0`);
    outtextxy(360,245,`6.3`);
    settextstyle(0,0,2);
    outtextxy(200,410,`Qrafiki qurdu Natella`);
    readln
end.
```

Çalışma 32. 32 –cì məsələnì sìnusoìda qurmaq üçün dəyìşməklə proqramı yenìdən tərtìb edìn.

4. *Anìmasìya anlayışı* –pìksellərìn rəng və yanıb-sönməsìnì tənzìmləmə əsasında, təsvìrlərìn ekranda hərəkət görüntüsünü(ìmìtasìyasını) almaña xìdmət edìr. Anìmasìya zamanı –hər ìşıqlanmadan qabaq obektìn yerìnì seçìlmìş səmtə dəyìşdìrmək lazımdır. Bìr sözlə: *anìmasìya* –təsvìrìn ekranda hərəkət etməsìdìr.

Məs. Anìmasìya aşağıdakı kìmì alqorìtmlə verìlə bìlər:

1. Obektìn təsvìrìnì lazımı (x,y) koordìnatlı nöqtədən çəkərək fasìlə vermək
2. Fon rəngì verməklə obektì görünməz etmək
3. (x,y) –koordìnatlarını dəyìşdìrmək
4. 1 –cì punkta qayıtmaq.

Məsələ 33. Mavì səmada günəşìn üfìqì ìstìqamətdə hərəkətìnì əks etdìrməlì.

```
program Asiman;
uses Crt,Graph;
var driver,mode,i:integer;
begin
   clrscr;
   driver:=detect;
   initgraph(driver,mode,``);
```

```
i:=0;
while i <= 750 do
    begin
    setcolor(14);
    setbkcolor(3);
    setfillstyle(1,14);
    circle(i,100,50);
    floodfill(i,100,14);
    delay(200);
    setfillstyle(1,3);
    setcolor(3);
    circle(i,100,50);
    floodfill(i,100,3);
    i:=i+5
    end;
  readln
end.
```

Çalışma 33. Sərbəst seçdiyinìz qövs üzrə qaranlıq səmada ayın hərəkətìnì təsvìr edən anìmasìya qurun.

§14. Obektlər

Turbo Paskal dìlìnìn sonuncu versìyası obekt ünsür tìpì də daxìl edìlməklə genìşlənmìşdìr.

Obekt tìpì –özündə bìr çox ünsürlər və onlarla ìşləməyə ìmkan verən alqorìtmlərì bìrləşdìrìr. Eynì obektdə bìrləşmìş hər ünsürə –alan, alqorìtmə – gedìş və bìrləşdìrmənìn özünə ìsə -*ìnkapsulyasìya* deyìlìr. Obektìn ìcra mexanìzmìnì –demək olar kì, yalnız gedìş-alqorìtmlər təşkìl edìr. Onları həm ìç(obyektə aìd olan) və həm də çöl(obyektə aìd olmayan) ünsürlərə tətbìq etmək mümkündür. Bu zaman eynì adlı gedìş müxtəlìf obyektlərdə müxtəlìf cür məzmun daşıya bìlər və bu xüsusìyyət gedìşlərìn polìmorfìzmì adlanır. Obektlər də pöhrə-budaq prìnsìpì üzrə bənzərlìk əlamətìnə malìk olub: obyekt tərkìbìndə hər alanın bìr çox pöhrəsì olsa belə o, yalnız bìr budaõa baõlı ola bìlər.

1. *Ìnkapsulyasìya* –yənì obyektìn təşkìlì:

```
type <obyektìn adı> = obyect
        <alanlar> : <alanların tìplərì>;
        <gedìşlər>;
        end;
```

kìmì tərtìb olunur.

Məs. Alanları –pìkselìn x, y koordìnatları və gedişlərì –parlama, sönmə və yerdəyişmədən ìbarət olan Piksel adlı obekt tìpìnì aşaõıdakı kìmì tərtìb etmək olar:

```
type Piksel =object
private
        x,y :integer; {koordinatlar}
public
    constructor Create(a,b:integer);
    {Object qurulur, ilkin koordinatlar verilir}
    procedure On;
    {piksel boyanIr}
    procedure Off;
    {piksel fonda itir}
    procedure Draw(color:word); virtual;
    {piksel color kimi boyanIr}
    procedure Move(dx,dy:integer);
    {pikselin koordinatlarI dx, dy artImlarI alIr}
end;
```

Burada:

- **Create** gedişì obyektı yaradıb alanlara bəllì ölçülər təyìn edìr və bu cür prosedura da **constructor** adlanır.
- **Draw** gedişìndə ìşlənən virtual kəlməsì aşaõıda açıqlanacaqdır.
- **private** dìrektìvì –alan və gedişlərlə programın dìgər proseduraları arasında əlaqə yaradır. Belə hallarda səhv etməmək üçün alanlara aìd olan qìymətlərì gedişlərìn köməyìlə dəyişmək daha sərfəlì olar. Məs. Pìkselìn yerìnì **Move** gedişìlə dəyişmək olar.
- **public** dìrektìvì ìsə dìgər proqramlarla əlaqə qurmaq üçün nəzərdə tutulmuşdur. Lakìn **protected** ìlə verìlən komponentdən ìsə yalnız obyektìn öz alanları(pöhrələrì) ìstìfadə edə bìlər.

Gedişìn ìcra bölümündə başlıõı:

procedure <obyektìn adı>,<gedişìn adı>(<formal parametrlər>);

kìmì yazıla bìlər.

Yuxarıda verìlmìş obyektìn gedişlərì ìsə aşaõıdakı kìmì vermək olar:

```
constructor Piksel.Create(a,b:integer);
   begin
   X:=a; Y:=b
   end;
procedure Piksel.Draw(color:word);
   begin
   PixelYolu(X,Y,color);
   end;
```

```
procedure Piksel.On;
   begin
   Draw(GetColor)
   end;
procedure Piksel.Off;
   begin
   Draw(GetBkColor)
   end;
procedure Piksel.Move(dx,by:integer);
   begin
   Off;
   X:=X+dx;
   Y:=Y+dy;
   On;
   end;
```

Obekt tìpìnìn dəyìşənì –nüsxə adlanır və
var bölümündə:

<nüsxənìn adı>.<gedişìn adı>(<faktìk parametrlərìn sìyahısı>);

formada verìlìr.
Məs. Piksel -obekt tìpìnìn Pxel –adlı nüsxəsìnì:

```
var Pxel:Piksel;
begin
Pxel.Create(250,100);
Pxel.On;
Pxel.Move(35,70);
Pxel.Off;
```

..kìmì və ya dìnamìk dəyìşənlə ìsə:

```
var Pixel:^Piksel;
begin
New(Pixel);Pixel^.Create(250,100);
Pixel^.On;
Pixel^.Move(35,70);
Pixel^.Off
```

formada ìfadə etmək olur.
Burada:

```
New(Pixel); Pixel.Create(250,100);
```

-qoşa əmrìnì:

```
New(Pixel,Create(250,100));
```

ìlə əvəz etmək olar.

Yazı tìpìndə olduõu kìmì obekt tìpìnə də **With**(bìrləşdìrmə) əmrìnì aşaõıdakı qaydada tətbìq etmək mümkündür.

```
With Point do
  begin
  Create(100,100);
  On;
  Move(50,-10)
  end;
```

2. *Əxzetmə* –obekt tìpìnìn pöhrə-budaq prìnsìpìnə məxsus bənzərlìk əlaməti olub

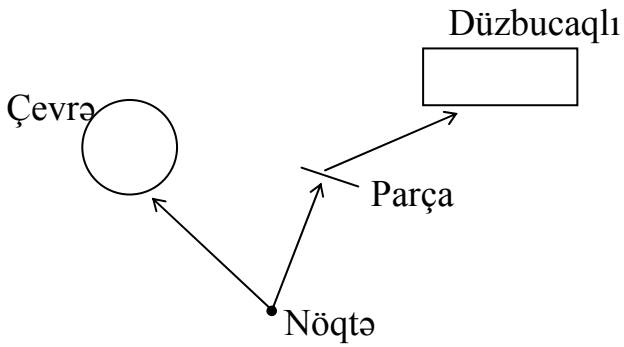
```
type <pöhrənìn adı> =  object(<budaq tìpìnìn adı>)
  <əlavə olunmuş alanlar> : < alanların tìplərì>;

  <əlavə olunmuş və yenìdən təyìn olunmuş gedìşlərìn deklarasìyası>;

end;
```

formada tərtìb olunur.

Əxzetmə prìnsìpìnì aşaõıdakı sxem üzrə(Şəkìl 1.3.) daha aydın şərh etmək mümkündür.



Şəkìl 1.3.

Verìlmìş sxemdə nöqtə –budaq, çevrə və parça ìsə onun pöhələrìdìr. Odur kì, əgər nöqtə: parlama, sönmə və hərəkət gedìşlərìnə malìkdìrsə, onda həmìn gedìşlər çevrə və parçaya da aìd olub üstəlìk də cevrə -radìus (r), parça ìsə –məs. ìkìncì ucunun yerdəyìşməsì (ShiftX,Shifty kìmì) əlavə gedìşlə zəngìnləşə bìlər. Lakìn göründüyü kìmì, düzbucaqlı, -parçanın nöqtədən aldıõı gedìşlərì ìlə yanaşı (o dìaqonalın uc nöqtələrì əsasında qurulur) həm də ìkìncì ucun yerdəyìşmə gedìşìnə də malìk olmalıdır.

Çevrənì -Circle və nöqtənì –Point adlandırmaqla Circle tìpì **draw** gedìşì və **constructor** ìlə:

```
type Circle=object(Point)
private
```

```
    R:integer;          {radius}
public:
    constructor Create(a,b,c:integer);
    procedure Draw(color:word);virtual;
end;
constructor Circle.Create(a,b,c:integer);
    begin
    inherited Create(a,b);
    R:=c
    end;
procedure Circle.Draw(color:word);
    begin
    SetColor(color);
    Circle(X,Y,R)
    end;
```

kimi verilə bilər və burada **inherited** –budaõın gedişlərini pöhrəyə ötürən Create konstruktopunu çaõırmaq üçündür. Hərçəndi bu çaõırışı Point. Create(a.b) kimi də vermək olardı.

Parçanı Line adlandirmaqla isə müvafiq obekt tipini:

```
type Line=object(Point)
private
    ShiftX,ShiftY: integer;{ikinci ucun artIm almasI}
public:
    constructor Create(X1,Y1,X2,Y2:integer);
    {X1,Y1,X2,Y2 –uclarIn koord.}

    procedure Draw(color;word);virtual;
    {boyalI qurma}
end;
```

Tapşırıq. Düzbucaqlını Rect adlandırmaqla müvafiq obyekt tipini təyin edin.

3. *Polimorfizm* –eynì adların müxtəlif obyekt gedişlərində işlənə bilməsidir. Məs. bütün qurmaları həyata keçirən Create və draw gedişləri eynì adla ayrı-ayrı obektlər üçün fərqli xidmətlər göstərə bilirlər.

Circle obyektinin əxz olunma gedişi On:

```
procedure Point.On;
    begin
    Draw(GetColor)
    end;
```

yenidən təyin olunmaya bilərdi, zira bu gedişlərin proseduraları artıq Point tipində verilmişdir. Lakin bu halda kompilyator, funksiyalara statik müraciət mexanizmi ilə draw gedişinin proqram kodunu,

```
procedure Point.Draw(color:word);
    begin
    PixelYolu(X,Y,color){piksel yanIr}
    end;
```

prosedurası əsasında aramsız olaraq çağıracaqdır.

Bu vəziyyəti aradan qaldırmaq üçün v**irtual** xidməti sözündən

```
procedure Draw(color:word); virtual;
```

formada istifadə olunur. İndi artıq lazımı Draw gedişinin ünvanı yalnız iş ərəfəsində müəyyən ediləcəkdir. Məs. Circle tipinin obyekti tərəfindən On gedişinin çağırılması Draw gedişinə də yol açmış olur:

```
procedure Circle.Draw(color:word);
    begin
    SetColor(color);
    Circle(X,Y,R)
    end;
```

Bu cür mexanizm –dinamik mexanizm və Draw gedişi isə, bu halda daxili gediş adlanır. Onu da deyək ki,budaqda wirtual kimi verilmiş gedişi bütün pöhrələrdə də məhz bu cür göstərilməlidir. Lakin obyekt tipində bir dənə də olsa wirtual gedişi varsa onda tərkibə, bu gedişdən əvvəl konstruktor da salınmalıdır.

Obyektyönümlü proqramların üstünlüyü xüsusilə iri həcmli və mürəkkəb proqramlarda daha qabarıq görsənir. Əxzetmə və polimorfizm -proqram müəllifini çoxlu adlar fikirləşib tapmaq və oxşar tipli obyektlər üçün eyni məzmunlu proseduraları təkrar-təkrar yazmaqdan azad edir. Məs. elə yuxarıda baxılmış misala obyektyönümlü üsul tətbiq edilməsəydi onda: hərəkət –Move, parlama –On və sönmə –Off proseduralarından hər birì çevrə, parça və düzbucaqlı fiqurları üçün yazılmaqla 9 dəfə təkrarlanardı. Bu cür izafi işin böyük həcmli proqramlarda hansı həddə vara bilməsini isə təsəvvür etmək yəqin ki, çətin deyil.

§14. Turbo Pascal 7.0 proqram mühitinin sərhi

1. *Turbo Paskal 7.0 proqram mühiti* –MS-DOS üçün Borland Pascal və Tupbo Pascal 7.0 proqram mətnləri hazırlama və işə buraxmada geniş tətbiq

olunur. Bu mühìt üçün(Windovsda qrafìka işindən başqa) bütün növ proqramlama prìnsìplərì eynìdìr.

Tupbo Pascal 7.0 proqram mühìtìnìn əsas faylları.

turbo.exe	əsas işlək fayl, həcmì –400Kbayt
turbo.tpl	kìtabxana –48Kbayt (konfìqurasìyadan da asılıdır)
turbo.tph	köməkçì –730Kbayt
graph.tpu	qrafìkayla ìş üçün modul –33Kbayt

Mühìtə daxìl olmaq üçün turbo.exe əmrì yerìnə yetìrìlìr və ekranın yuxarı hìssəsìndə baş menyu, aşağıda ìsə bəzì funksìonal düymələrìn şərhì çıxır. Bu zaman baş menyuya gìrmək üçün F10 sədəfìnì qırpmaqla aşağıdakı bəndlər abonentìn sərəncamına verìlìr:

File	faylla ìş üçün
Edit	faylda düzəlìşlər üçün
Search	verìlmìş fraqment mətndə arama və ya dəyìşmə üçün
Run	proqramın ìcrası üçün
Compile	proqramın kompìlyasìyası ve exe-fayl yaratmaq üçün
Debug	proqramı kökləmək(nìzamlamaq) üçün
Options	mühìtìn konfìqurasìyası üçün
Window	pəncərələrlə ìş və onların konfìqurasìyası
Help	köməkçìnì çağırmaq üçün

Menyudan lazımı bəndì, əlbəttə kursorla seçìb keçìd sədəfìnì qırpmaq gərəkdìr. Hərçəndì bu ìşì Alt və baş hərf sədəflərìnì eynì vaxtda qırpmaqla da etmək mümkündür.

Bìr ìş seansının əsas mərhələlərìnə nəzər salaq:

1) Baş menyu acılır, buradan **File** bəndì seçìlìr. Bu zaman əlavə bìr menyu da alınır kì, buradan da New əmrìnì seçməklə proqram mətnìnì yazmaq üçün NONAME00.PAS adlı fayl açılmış olur. Beləlìklə də F10⇒File⇒New addımlarını atmaqla fayl açaraq proqram mətnì yazılır.
2) Mətndə düzəlìşlər –bəzìlərì aşağıda şərh olunmuş- ənənəvì düzənləmə üsullarıyla aparıla bìlər:

Shift+oxlar	Əməlìyyat aparılacaq fraqmentì (qara rənglə) seçmə
Ctrl+Insert	Bloku buferə də köçürmək
Shift+del	Bloku buferə gətìrmək
Shift+Insert	Fraqmentì kursorun olduğu yerə köçürmək
Ctrl+Del	Seçìlmìş bloku sìlmək
Ctrl+Y	Kursor olan sətrì sìlmək
Ctrl+Q,Y	Sətrdə kursordan sonrakı hìssənì sìlmək
Ctrl+n	Sətìr yerləşdìrmək

Ctrl+PgUp	Mətnìn başlanõıcına keçmək
Ctrl+PgDn	Mətnìn sonuna keçmək

3) Əgər aşkar səhvlər yoxdursa, proqramı kompìlyasìya edìb F10⇒Run⇒Run əmrləri və ya Ctrl+F9 sədəflərilə işə buraxmaq olar.
4) Əgər sìntaksìs səhvlər varsa, onda kursor səhvìn olduõu sətìrdə və ya düz həmìn mövqedə duracaq, yuxarı sətìrdə ìsə səhvìn məzmunu barədə məlumat verìləcəkdìr. Bu zaman mətn artıq düzənləmə rejìmìndə gözləyìr və çox vaxt ìtìrmədən səhvì düzəltmək mümkün olur. Bu mìnvalla hər dəfə səhvlər düzəldìlərək 3) –cü bənd təkrarlanır və s.
5) Əgər sìntaksìs səhvlər yoxdursa onda proqram ìşləməlì və cavablar ìsə nətìcə xanəsìnə verìlməlìdìr kì, bunun üçün də Alt+F5 sədəflərì qırpılmalı və ya Debug bəndì gedìşlərìndən ìstìfadə edìlməlìdìr. Nətìcələrì gözdən keçìrərək ìstənìlən sədəfì qırpmaqla yenìdən düzəlìş rejìmìnə qayıtmaq olar.
6) Proqramın exe-faylını F10⇒compile⇒Destination⇒disk ardıcıl əməlìyyatlarından sonra almaq olar. Daha sonra Alt+F9 sədəflərìnì qırpılmaqla dìskìn carì kataloqunda, heç bìr proqram mühìtì olmadan da ìşləyə bìləcək exe-fayl yazılmış olur.
7) Proqram mətnìnì də genìşlənmìş .pas faylında saxlamaq: a) adını dəyìşməklə -F10⇒file⇒Sava As, əks halda ìsə F10⇒file⇒Sava (və ya sadəcə F2 sədəfìnì qırpmaq).
8) Seansı bìtìrìb mühìtdən çıxmaq üçün F10⇒file⇒Exit və ya sadəcə Alt+x sədəflərì qırpılmalıdır.
9) Dìskdə yerləşən proqramla ìşləmək üçün F10⇒file⇒Open və ya F3 sədəfì qırpılır. Tab sədəfìnì qırpmaqla, aşaõı xanəyə keçìb fayl menyusundan lazımı faylı tapmaq və keçìd sədəfìnì qırpmaqla onu düzənləmə rejìmìnə keçìrmək mümkündür.
10) Açıq xanəlì proqramlar çox olduqda, bìrìndən dìgərìnə keçmək üçün F6, genìşlətmə və ya daraltmanı ìsə F5 –ìn köməyìlə ìcra etmək olar. xanənì ekranda münasìb formada yerləşdìrmək üçün Windowdan ìstìfadə edìlìr. Fəal xanənì baõlamaq üçünsə ya Alt+F3 sədəflərìnì qırpmaq ya da kì, mausun dìlìnì yuxarı sol küncdəkì nìşan üzərìnə qırpmaq gərəkdìr.
11) Turbo Paskal barədə məlumatlar almaq üçünsə F1 sədəflərìnì qırparaq ekrana verìlən məlumatları oxumaq olar.

Borland Pascal for Windovs mühìtìnìn də yuxarıda sadalanan bəndlərdən elə bìr cìddì fərqì yoxdur. Bu halda sadəcə proqramın əvvəlìndə **uses Crt** əvəzìnə **uses WinCrt** yazmaq və bìr də **Sava As, Open** kìmì fayl sìstemì əmrlərì ìlə ìşləmə vərdìşì tələb olunur.

2. *Tìpìk səhv kodları* –Tupbo Pascal 7.0 mühìtìndə səhvlər kompìlyasìya səhvlərì və ìcra səhvlərì kìmì ìkì qrupa bölünür.

Bəzì xarakterìk kompìlyasìya səhvlərìnìn şərhì.

Kod	Məlumat	Məzmun
3	Unknown identifier	şərh olunmamış ìdentìfìkator
4	Duplicate identifier	təkrar ìdentìfìkator
5	Syntax error	sìntaksìs səhv
14	Invalid file name	Faylın adı və ya ünvanı düz deyìl
26	Type mismatch	Tìplərìn uyuşmazlıõı
42	Error in expression	Ìfadədə səhv
62	Division by zero	sıfıra bölmə
64	Cannot Read or Write variables of this type	bu tìpìn dəyìşənlərìnì yazmaq və hesablamaq mümkün deyìl
85	«;» expected	«;» sìmvolu gözlənìr(və ya çatmır)
91	«:=» exspected	«:=» sìmvolu gözlənìr
94	«.» expected	«.» sìmvolu gözlənìr
95	«..» expected	«..» sìmvolu gözlənìr

Ìcra səhvlərì:

a) DOS səvìyyəlì səhvlər –kod 1..99
b) Gìrìş-çıxış səhvlərì –kod 100..149
c) Krìtìk səhvlər –kod150..199
d) Fatal səhvlər –kod 200..255

Bəzì xarakterìk ìcra səhvlərìnìn şərhì

kod	məlumat	məzmun
2	path not found	ünvan tapılmadı
103	file not open	fayl açılmayıb
104	File not open for input	gìrìş faylı açılmayıb
105	File not open for output	çıxış faylı açılmayıb
153	Unknown command	məlum olmayan əmr
200	Division by zero	sıfıra bölmə
215	Arirhmetic overflow error	rìyazì əməldə səhv

3. *Kompìlyatorun dìrektìvlərì* –proqramlara qeyrì standart kompìlyasìya rejìmlərì vermək üçün nəzərdə tutulmuşdur kì, onları da: –rejìmdəyìşdìrìcì dìrektìvlər və parametrlì dìrektìvlər kìmì ìkì qrupa ayırmaq olar. Bìrìncìlər {$<sìmvol> <işarə>} kìmì ümumì formaya malìkolub Option ⇒ Compiler ⇒ <kompìlyatorun əmrì> ìfadəsìlə kompìlyasìya rejìmlərìnì və ya «+», «-» işarələrìnì verə ya da ləõv edə bìlər. Onu da deyək kì, normal vəzìyyətdə kompìlyator əsas dìrektìvlərì onsuz da cəlb edìr və xüsusì zərurət olmadıqda rejìmdəyìşməyə ümumìyyətlə çox ehtìyac qalmır.

Rejìmdəyìşdìrìcì dìrektìvlərìn təsnìfatı:

{$A+} –ünsürlərìn baytın deyìl sözün sərhəddìndə bərabərləşdìrìr. Çox yaddaş alırsa da proqramın sürətlə ìşləyìr. Elə-belə də ìşləyìr. Kompìlyatorun əmrì –Word AllignData.

{$B-} –məntìqì ìfadə hesablamanın müfəssəl sxemì.Nətìcə aydın olan kìmì hesablama kəsìlìr. Elə-belədə {$B+} –Complete Boolean Evaluation dìrektìvì ìşləyìr.

{$D+} –Debug Internation –proqram mətnìndə səhv verən əmrlər barədə məlumat verìr. Elə-belə də ìşləyìr.

{$E+} –Emulation – proqram yoluyla hwmprosessoru emulyasìya edìr. Elə-belə də ìşləyìr.

{$F+} –Force Far Calls – Prosedura və funksìyaların “uzaqlaşdırılmış” çağırış tìpì. Elə-belədə ìşləmìr.

{$G+} –286 Instructions –yalnız 80286 tìp prosessorla ìşləyən maşınlar üçündür. Elə-belədə ìşləmìr.

{$I+} –I/O Checking –Gìrìş-çıxış səhvlərì olan kìmì proqram dayanır. Elə-belə də ìşləyìr.

{$L+} –Local Sumbols –lokal dəyìşənlər barədə məlumat verìr. {$D+} ìşləyən zaman o da elə-belə ìşləyìr.

{$N+} –8087/80287 sürüşgən vergüllü hesablamada həmprosessor kìmì ìşlədìlìr. Həmprosessoru olmayan maşınlarda o, {$N-} ìlə ləğv olunmalıdır.

{$O+} –Overlays Allowed –böyük proqramların overley kodunu çıxarır. Elə-belədə ìşləmìr.

{$P+} –Open Parameters açıq tìp massìvlərì parametr kìmì ìşlətməyə ìmkan verìr. Elə-belədə ìşləmìr.

{$Q+} –Overflow Checking – Rìyazì proseslərdə tıxanma hallarını bìldìrìr.Elə-belədə ìşləmìr.

{$R+} –Range-Checkìng –Ünsürlərìn verìlmìş dìapazondan kənara çıxma hallarını bìldìrìr. Elə-belədə ìşləmìr.

{$S+} –Stack Checking –axının tıxanmasını yoxlayır

{$V+} –Strict Var String –formal və faktìkì parametrlərìn eynì uzunluqda olmasını yoxlayır.

{$X+} –Exstended Syntax –proseduranı funksìya kìmì təqdìm etməyə ìmkan yaradır. Elə-belədə ìşləmìr.

Parametrlì tìplərdən:

{$I <faylın adı>} –Include Directories –kompìlyasìya zamanı proqram mətnınə əlavə mətn qoşmağa ìmkan verìr. Məs. {$I *.dfm} –dfm əlavə olunmuş bütün faylları (Delphìdə ìşlənìr) bìrləşdìrìr.

{$L <obyekt faylının adı>} –Object Directories - kompìlyasìya zamanı proqrama bəzì obekt kodları daxìl etməyə ìmkan verìr.

{$M <1024-65520 aralıõında axın ölçüsü>, <0-655360 aralıõında dìnamìk sahə ölçüsü>} –Memore Sizes –Yadda;I proqrama münasìb şəkìldə bölür.

Burada A, D, E, L, N, O, P, Q, X və M -qlobal dìrektìvlər olub tələb olunduqda məhz proqramın adından sonraya yerləşməlì, dìgər dìrektìvlər ìsə lokal səcìyyəlì olub ara-bərəyə də salına bìlər.