

I Bölmə

TURBO PASCAL 7.0

§1. Əsas anlayışlar.

1. *Dilin quruluşu.* Paskal dili ötən əsrin 60 - 70 –ci illərində İsveçrəli professor N. Virt tərəfindən yaradılaraq dünyada ilk hesablayıcı maşın ixtiraçısı sayılan –fransız riyaziyyatçısı və filosofu Blez Paskalın (1623 – 1662) şərəfinə adlandırılmışdır. Paskal dili quruluşca sadə və tətbiq baxımından münasib olub, daha mütərəqqi imkanlara malik «Delfi» -vizual proqram dilinin də əsasında durur. Müasir kompüter texnologiyasının da elə məhz vizual proqramlama prinsipinə əsaslanması isə artıq bu dilin müstəsna əhəmiyyəti və aktuallığını səciyyəyləndirən başlıca amil sayıla bilər.

Paskal proqramlarını –sintaksis quruluşlu *əmlər* (operator və ya komandalar), onları isə öz növbəsində –söz, ədəd, simvol, əməl və s. bu kimi *leksemlər* (bölünməz elementlər) təşkil edir. Proqramın bitmiş məzmun daşıyan əmləri –*bənd*, **begin** sözü ilə başlayıb **end** sözü ilə bitən hər bir frazası isə *-blok* adlanır. Proqram mətnində işlənən sözlər isə öz təyinatına görə –*xidməti sözlər*, *standart sözlər* və proqramda işlənən ayrı-ayrı obyektləri və ya başqa sözlə desək –proqram üsürlərini özündə əks etdirən *adlar* (identifikatorlar) kimi qruplara bölünürlər:

Dilin əsas *xidməti sözləri*:

and – və	mod – qalıq
array – massiv	nil – sıfır
begin – başlanğıc	not – yox (xeyr)
case – seçim	of – dan (başlayaraq)
const – konstantlar	or – və ya
div – qalıqsız bölmə	packed – qablaşmış
do – icra et	procedure – prosedura
downto – aşağı qədər	program – proqram
else – digər halda	record – yazı
end – son	repeat – təkrar et
file – fayl	set – çoxluq
for – üçün	then – onda
function – funksiya	to – qədər
goto – keç...	type – tip
if – əgər	until – hələlik
in – daxilində	var – dəyişənlər
label – nişan	while – hələlik və s.

! -bu sözləri istər əlyazması və istərsə də basma hərflərlə yazmaq olar.

Standart sözlər – qruplar üzrə:

<p>1) <i>Standart tipli ünsür adları:</i></p> <ul style="list-style-type: none">- boolean – məntiqi- char – simvol- integer – tam- real – həqiqi- text – mətn faylı və s. <p>2) <i>Standart tipli sabitlər:</i></p> <ul style="list-style-type: none">- false – yalan- true – doğru- maxint – maksimal tam- pi – π ədədi və s.	<p>3) <i>Standart funksiyalar:</i></p> <ul style="list-style-type: none">- abs – mütləq qiymət $y= x$- arctan – $y= \arctn(x)$- cos – $y= \text{Cos}(x)$- ln – $y=\ln(x)$- exp – üstlü funksiya və s. <p>4) <i>Standart proseduralar:</i></p> <ul style="list-style-type: none">- read- readln- write- writeln- inc- dec və s.
--	--

! -əgər bu cür standart sözlərlə başqa obyektləri adlandırsaq, onlar öz əsas təyinatını itirmiş olur.

Ad –proqramda əks olunan istənilən ünsürə verilə bilər. Obyektə, proqram və ya içproqrama verilən adlarda yalnız latın hərfləri, rəqəmlər və «_» işarəsi işlənmə bilər, ancaq gərək bu adlar rəqəmlə başlanmasın.

! -bu kitabda:

- Proqram mətnləri və xidməti sözlər xüsusi seçilmiş (Courier New) fontla yazılır.
- Xüsusilə rusdilli ədəbiyyatlardan hərfi tərcüməylə alınmış və məzmunu dərk etməyə ciddi əngəl törədən bəzi terminlər, məs:

Operativ yaddaş –fəal yaddaş;

Komanda –əmr;

Verilən(dannıy) –ünsür;

Altproqram (podproqram) –içproqram;

Pəncərə(okno) –xanə;

Metod(üsul yox, =əməliyyat) –gediş;

Klavışşa –sədəf;

Sədəfi və ya mausun dilini basmaq(najat və ya şelkat) –basmaq və ya tıklamaq

iki dəfə tıklamaq –qoşa tıklama,

sıxıb saxlamaq –tutmaq;

Yanıb-sönmə(miqat) –səyrimə;

Tabulyasiya –darama;

Korrektirovka – düzənləmə;

Kommentari -izah

Format -biçim

Göstərici (ukazatel və ya strelka) –ox;

Sahə(pole, həm də ploşşad) –alan,

redaktə sahəsi –iş alanı;

Probel –ara;

Enter sədəfi –keçid sədəfi(lakin təklikdə elə Enter);

Hər hansı səhifə, xanə və ya alanı təşkil edən hissələr(zakladka) -rəf və s. **kimi işlənmişdir.**

2. *Paskal proqramının strukturası* -aşağıdakı bölmələr əsasında tərtib olunur:

Başlıq	
program	<Proqramın adı>;
Təqdimat bölümü	
uses	– gərəkli kitabxana və proqram modullarının göstərilməsi;
label	– nişanları (metkaları) tanımaq;
const	– sabitləri tanımaq;
type	– tipləri tanımaq;
var	– dəyişənləri tanımaq;
procedure	– proseduraları tanımaq;
function	– funksiyaları tanımaq;
İcra bölümü:	
begin	<əmrilər bölümü>
end.	

Burada digər bölümlər buraxıla bilən, sonuncu isə hər bir proqram üçün zəruridir. Proqram bəndləri (və ya əmr konstruksiyaları) bir - birindən «;» simvolu ilə ayrılmalı və ən axırda mütləq nöqtə qoyulmalıdır.

Proqramda xidməti söz və ya adların böyük, kiçik, əlyazma və ya basma hərflərlə yazılması eyni məna daşıyır.

Məs: «A» və ya «a» («Chinar» və ya «chiNar») adları eyni bir obyektə aid hesab edilir.

Çox vaxt proqram mətnində açıqlama məqsədi daşıyan izahlardan (*kommentarilərdən*) istifadə olunur ki, onlar da –istənilən nahiyədə, füqurlu mötərizə içinə alınmaqla və ya (*izah*) şəklində verilə bilər. Bu yazılar proqramın iş prosesinə heç bir təsir etmir. Yalnız izahların *direktiv* forması - lazımı yaddaş modeli və proqramlama rejimi fəaliyyətini tənzim etmək və s. məqsədlər üçün işləyə bilər. *Direktiv* yazılarkən, açılmış mötərizədən sonra «\$» simvolu qoyulmalıdır.

Təqdimat bölümündə:

lazım olacaq sabit kəmiyyətlər **const** əmrilə, aşağıdakı formada təqdim edilir:

const	<konst. 1> = <qiymət 1>

	<konst. n> = <qiymət n>

Eynilə, əməliyyat prosesində dəyişilə bilən kəmiyyətləri, yəni dəyişən ünsürləri də **var** əmrilə:

var	<dəyişən siyahısı 1> : <tip 1>
-----	--------------------------------

.....
<dəyişən siyahısı n> : <tip n>

kimi tanımaq lazımdır

Bu cür siyahılarda sadalanan elementlər bir - birindən vergüllə ayrılmalıdır.

3. *Proqram nümunəsi*: Proqram –qarşıya qoyulmuş hər-hansı məsələni həll etmək üçün tərtib olunmuş alqoritmi yerinə yetirməli olan əmrlər ardıcılığıdır. Proqram alqoritmi işə: -lazımı ünsürləri tanıtma, məqsədyönlü hesabat və nəticə çıxama –prinsipi əsasında qurulur.

Məsələ 1. Oturacağının radiusu $r = 1.5\text{sm}$ və hündürlüyü $h = 6\text{sm}$ olan düz konusun tam səthinin sahəsini və həcmi hesablamaq üçün proqram yazın.

Məsələni həll etmək üçün:

- Sabitlər: r –oturacağın radiusu, h –konusun hündürlüyü
- Dəyişənlər: d –doğuran, s -tam səth və v –həcm olmaqla işarə etsək, tələb olunan proqramı izahlı şəkildə, aşağıdakı kimi tərtib etmək olar:

```
program Konus; {Adlanma}
uses Crt;
const r = 1.5; h = 6; {r, h –sabit kimi verilir}
var d, s, v: real; {d, s, v- nin tipi verilir}
begin
  clrscr; {Ekran silinir}
  d := sqrt(sqr(r) + sqr(h)); {d –ni tapmaq}
  s := pi*r*d; {s –i tapmaq}
  v := 1/3*pi*sqr(r)*h; {V -ni tapmaq}
  writeln('S = ', s:5:3); {s -ekrana verilir}
  writeln('V = ', v:5:3); {V -ekrana verilir}
  writeln('Proqramı yazdı -Ü. Süleymanov');
  readln;
end.
```

Proqram mətnində işlənən əməl və münasibətlər:

*- vurma

:= - qiymət vermə

sqrt -kvadrat kök alma

sqr -kvadrata yüksəltmə

kimi işarə edilmişdir.

writeln('S = ', s:5:3); yazılışı -tam səth üçün tapılan ədədi qiymətin ekrana – 3 –ü vergüldən sonra olmaqla, cəmi 5 rəqəmlə verilməsini tələb edir.

readln -keçid(Enter) düyməsi basılanadək nəticənin ekranda qalmasını tələb edir.

Beləliklə, proqram öz işini aşağıdakı məlumatı ekrana çıxarmaqla bitirir:

S = 29.145

V = 14.137

Proqramı yazdı -Ü. Süleymanov

Çalışma 1. Katetlərindən biri 3,5sm və hipotenuza endirilmiş hündürlüyü 2,8sm olan düzbucaqlı üçbucağın sahəsini tapmaq üçün proqram tərtib edin

§2. Ünsür tipləri.

Bu kitabda -kəmiyyət və adlarda təzahür edən ədəd, dəyişən, funksiya, xassə, əlamət, rəng, dəlil və s. tərkib elementləri (başqa sözlə, verilənlər)– «ünsür» adlanır. Ünsürlər -*standart və qeyri standart tiplər* olmaqla iki qrupa bölünürlər:

1. *Standart tipli ünsürlər:* Proqramlamada çox vacib anlayışlardan biri sayılan əsas standart tip -*dəyişən kəmiyyətdir*. Burada *dəyişən* –əməliyyat yaddasında müəyyən kəmiyyətin mümkün qiymətlərini saxlaya biləcək ünvanlar məcmusudur –desək, səhv etmərik. Dəyişənə –ad, qiymət, tip, miqdar və s. göstəriciləri əvvəlcədən təqdim edilməlidir.

Ünsürün tipi, onunla aparıla biləcək əməlləri və qiymət diapazonunu da müəyyən edir. Məs. Əvvəlki paraqrafda tərtib olunmuş «Konus» proqramında l, s və v dəyişənləri **real**(həqiqi ədəd) tipi olaraq göstərilirdiyi üçün onlar yalnız $[2.9 \cdot 10^{-39} \text{ — } 1.7 \cdot 10^{38}]$ diapazonunda (cədvəl 2.) qiymətlər ala bilər.

Aşağıda bir sıra çoxişlənən *standart dəyişən tipləri* sadalanır.

Cədvəl 1. *Tam ədəd tipləri.*

<i>Tipin adı</i>	<i>Qiymət diapazonu</i>
byte	0..255
shortint	-128..+127
word	0..65535
integer	-32768..+32767
longint	-2147483648..+2147483647

Cədvəl 2. *Həqiqi ədəd tipləri.*

<i>Tipin adı</i>	<i>Qiymət diapazonu</i>
single	$1.5 \cdot 10^{-45} \text{ .. } 3.4 \cdot 10^{38}$
real	$2.9 \cdot 10^{-39} \text{ .. } 1.7 \cdot 10^{38}$
double	$5.0 \cdot 10^{-324} \text{ .. } 1.7 \cdot 10^{308}$
extended	$3.4 \cdot 10^{-4951} \text{ .. } 1.1 \cdot 10^{4932}$

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

Program 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) – Kompyuterin 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 şərh>
.....
<n -ci tipin adı> = <n -ci tipin şərh>
```

Qeyri standart tipli ünsürlər əsasən -sadalanın, diapazonlu, sətir tipli və s. formalarda ola bilər.

Qeyri standart sadalanın tiplər -adlar və hər adın mümkün qiymətlərini mütərizədə, 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 tipli qiymətlər verilə bilməz.

Sadalanın tiyə 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;
```

Təbii 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 ki, bu tip ünsürləri klaviaturadan daxil etmək və ya ekrana çıxarmaq olmaz.

Diapazonlu tip –hər-hansı nizamlı baza tipindən lazımı hissələrin seçilməsi ilə düzəlir və proqramda aşağıdakı formada təqdim olunur.

```
type <tipin adı>=<1-ci qiymət>,<2 -ci qiymət>,...,<n -ci qiymət>;
```

Misal. Həftənin günlərini, iş günlərini və ayların nömrələrini uyğun olaraq ifadə edən weekdays, workdays, months tiplərini yaradaq və onlar üçün day1, day2, day3, və month dəyişənlərini təyin 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;  
1month : months;
```

*Sətir tipli dəyişənlər*in qiyməti -istənilən ardıcılıqla düzülmüş, 255 -i aşmayan sayda simvollar ardıcılığı ola bilər. Tiplər bölümündə bu cür ünsürləri istər sabit və istərsə də dəyişən kimi təqdim etmək mümkündür.

Məs:

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

-kvadrat mötərizədə sətirin uzunluğu (simvollar sayı) göstərilir.

Tip kimi verilən sabitlər –dəyişəni təqdim etməklə eyni vaxtda həm də onu qiymətləndirməyə imkan verir:

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

Adi sabitlərdən fərqli olaraq bu cür təqdim olunan sabitləri proqramda dəyişmək mümkündür. belə sabitlərə bir – neçə misal göstərək:

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

§3. Sadə proqramlar.

Sadə proqramlar əsasən təyinat, ünsürlərin giriş–çıxışı və proseduralara müraciət məqsədi daşıyan əmrlərdən ibarət olur ki, bu cür proqramlara xətti proqramlar da deyilir.

1. *Təyinat əmrləri* –aşağıdakı formada yazılır:

$\langle \text{dəyişənin adı} \rangle := \langle \text{ifadə} \rangle$
--

Bu əmrlə -dəyişənin qiyməti ifadə əsasında təyin edilir. İfadələr isə – məchul olmayan dəyişən və funksiya, ədəd və sabitlərdən ibarət, proqram dilinə uyğun formada yazılmış istənilən riyazi ifadə ola bilər. «Konus» proqramında l, s və v dəyişənləri də məhz bu cür ifadələr əsasında təyin edilmişdir:

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

Əlbəttə təyinat əmrlərində dəyişən və ifadələrin tipləri ziddiyyətsiz verilməlidir. Məs. Həqiqi (və ya sətir) tipli dəyişənə tam ədəd (və ya simvol) tipli ifadələrin qiymətləri verilə bilər lakin tərsinə yox.

2. Paskal dilində yazılmış proqramların məqsəduyğun fəaliyyətini tənzim etmək üçün işlənən əsas əməllər (Cədvəl 3 –də), standart funksiyalar və proseduralar (Cədvəl 4 –də) sadalanır:

Cədvəl 3. Əsas əməllər.

İcra sırası	Əməl	Əməlin təyinatı
1	+, - not	işarələmə məntiqi inkar
2	*, / div, mod and	vurma, bölmə tam ədəd əməlləri məntiqi hasil
3	+, - or, xor	toplama, çıxma məntiqi cəm
4	=, <, <=, >, >= in	müqayisə çoxtuğa aid olma

Əməllər -riyazi ifadələrin hesablama qaydaları əsas götürülməklə prioritet əsasında (yüksəyi 1 olmaqla) yerinə yetirilir. 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 div əməlinin nəticəsi qismət, mod əməlinin nəticəsi qalıq olur. Məs:

$(29 - 11) \text{ div } 5 = 3$; $13 \text{ mod } 5 = 3$

Cədvəl 4. Standart funksiyalar və proseduralar.

Funksiya	Arqumentin tipi	Nəticənin Tipi	Riyazi yazılış
abs(x)	integer, real	integer, 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)	nizamlanmış	integer	ASCII –simvol kodları.
succ(x)	nizamlanmış	nizamlanmış	x –in növbəti qiymətini verir
pred(x)	nizamlanmış	nizamlanmış	x –in əvvəlki qiymətini verir
round(x)	real	integer	x –i tam ədədə yuvarlaqlaşdırır
trunc(x)	real	integer	x –in tam hissəsi
int(x)	real	real	x –in tam hissəsi
frac(x)	real	real	x–in kəsr hissəsi
odd(x)	integer	boolean	true (x –tək) falze (x –cüt)
random(x)	integer	integer	x –dən kiçik təsadüfi müsbət-tam ədəd verir
upcase(x)	char	char	Latin hərf-ni əlyaz-na çevirir
Proseduralar			
inc(x, y)	integer	integer	x-i y qədər böy-r
inc(x)	integer, char	integer, char	x-i 1 vahid böy-r
dec(x, y)	integer	integer	x-i y qədər az-r
dec(x)	integer, char	integer, char	x-i 1 vahid az-r

Funksiyaların qiymətləri və proseduraların tətbiqinə aid bir neçə misal 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, verilmiş əsas funksiyalar dəstindən ibarət olan digər funksiyalar da almaq mümkündür. Məs:

$$Ctg(x) = \frac{Cos(x)}{Sin(x)}; \quad arcCtg(x) = arcSin\left(\frac{1}{\sqrt{1+x^2}}\right) \quad \text{və s.}$$

3. Ünsürləri *dialogla tanıtma əmrləri* –**read**, **readln** -aşağıdakı formada yazılır

```
read(və ya readln)(<1 -ci dəyişən>, ... , <n -ci dəyişən>);
```

və verilənlərin klaviaturadan daxil edilməsinə imkan yaradır.

Məs. «Konus» proqramında

const r = 1.5; h = 6; –əvəzinə

read(r, h) və ya **readln**(r, h)

yazılmış olsaydı, onda proqram arada öz işini dayandırar və kursor ekranda döyünməyə başlar, qiymətləri ekrana yazıb keçid(Enter) sədəfini basınca o, öz işini yenidən davam etdirirdi. Təbii ki, bu hal daha universal xarakter daşıyaraq r və h –in istənilən qiymətləri üçün tətbiq oluna bilər.

read əmrindən fərqli olaraq **readln** ilə verilmiş məlumatın ardınca həmin sətərə başqa məlumat yazıla bilməz və bu əmr mətnlərlə işləməyə daha çox yararlıdır.

4. *Çıxarış əmrləri* –**write** və **writeln** – proqramın verdiyi cavab, nəticə, və digər məlumatları ekrana çıxarmaq üçün işlənərək aşağıdakı formada yazılır:

```
write(və ya writeln) (<1 ci ifadə>, ..., <n -ci ifadə>);
```

Bu zaman məlumatlar ekranın çıxış xanəsinə verilir ki, MS-DOS üçün TP mühitində bu xanə məs. Alt +F5 sədəfləri ilə seçilə bilər.

write -dən fərqli olaraq **writeln** əmrinin verdiyi məlumatdan sonra gələn çıxarış ancaq növbəti sətirdən başlayır. Odur ki, **writeln** əmrini parametrsiz işlətməklə ekranda sətir buraxmaq və ya sətirdən – sətərə keçməni də təşkil etmək mümkündür.

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

writeln('S = ', s:5:3);

writeln('V = ', v:5:3);

kimi tətbiq olunmuşdur. Xatırladaq ki, bu formatda s və v dəyişənləri üçün qiymətlər –3 -ü vergüldən sonra olmaqla, cəmi 5 rəqəmlə ekrana verilir.

writeln('Proqramı yazdı Ü. Süleymanov');

bicimli çıxarış əmri isə -dırnaq arasında verilən simvollar ardıcılığını ekrana olduğu kimi verməyi təmin edir.

İndi isə başqa bir məsələnin həlli üçün proqram tərtib 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 dairənin sahəsini tapmaq üçün proqram tərtib etməli.

Əvvəlki proqramdan fərqli olaraq, burada artıq bütün əmrlərə izah verilməsinə 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 işini, ekranla aşağıdakı kimi dialoq şəraitində aparır:

Ekrana əvvəl çıxır:

Koordinatlar verilsin:

Biz yazırıq:

-1.8 2.1

3 0.5

və keçid sədəfi basılır..

Ekrana sonda çıxır:

c = 31.79

s = 8.0424771932E+01

Göründüyü kimi

writeln('s = ', s);

yazılışında s kəmiyyətinin qiyməti ekrana 11 rəqəmli sürüşgən nöqtə ilə verilmişdir.

Çalışma 2. Təpə nöqtələri $A(a, x)$, $B(b, y)$, $C(c, z)$ olan üçbucağın perimetrini və sahəsini hesablamaq üçün proqram yazmalı.

§4. Budaqlanma.

Budaqlanma –proqram fəaliyyətinin yığma əmrlərlə həyata keçirilən əməliyyat bölümüdür:

1. *Yiōma əmrlər* - əmrlər dəstindən aşağıdakı formada yidilmiş bloktan ibarət olur:

```
beqin
    <1 -ci əmr>;
    .....
    <n -ci əmr>;
end;
```

Yadda saxlamaq lazımdır ki, hər bu cür bloku kompilyator bir əmr kimi qəbul edir.

Onu da deyək ki, **beqin end;** və ya **;;** blokları boş əmrlər sayılır.

Bununla da Paskal dilində işlənən əmrlər –boş əmr, sadə əmr və yiōma əmr kimi qruplara bölünmüş olur.

2. Paskal dili diōer məsələlərlə yanaşı *məntiqi məsələlərin həlli* üçün də proqram tərtib etməyə imkan verir. Təbiidir ki, bu cür proqramların başlıca ünsürü məntiqi ifadədir:

Məntiqi ifadə –bəzi zəruri bilgiləri əldə etmək üçün lazım olan şərtlərin proqram dilində, qanunauyōun yazılış üsuludur. Xatırladaq ki, -məntiqi ifadələr yalnız iki qiymət ala bilər -true(doōru) və false(yalan). Məntiqi ifadələr də iki qrupa ayrılır:

- 1) *Sadə ifadə* -arasında münasibət işarəsi qoyulmuş bir cüt hesabı ifadədir.
- 2) *Yiōma ifadə* –**not**, **and** və ya **or** –məntiq əməlləri ilə birləşmiş sadə ifadələr dəstidir. Bu əməllərin fəallıq prioriteti **not** – **and** – **or** ardıcılıōına uyōundur. Əməllər aşağıdakı kimi təyin olunmuşdur:

İfadə	Dəyəri	İfadə	Dəyəri
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 qiymətləri üçün aşağıdakı məntiqi ifadələri yazıb bilərik:

<i>Sadə ifadə</i>	<i>Qiyməti</i>	<i>Mürəkkəb ifadə</i>	<i>Qiyməti</i>
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 ki, mürəkkəb ifadəni təşkil eləyən hər bir sadə ifadə adı mōtərizədə yazılır. Məntiqi ifadələrdə a<=b<c kimi ikiqat bərabərsizlik


```
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 ki, **if** əmri altında olan **else** xidməti sözündən əvvəl ; işarəsi qoymuq olmaz.

Çalışma 3. x arqumentinə ekrandan verilən qiymətə görə
 $y = |x - 1| - |x - 3|$
funksiyasının müvafiq qiymətini ekrana çıxaran proqram yazın.

if əmrinin qısa forması isə aşağıdakı kimi yazılır:

if <məntiqi ifadə> then <əmr>;
--

Bu əmrdə məntiqi ifadə doğru isə qeyd olunmuş əmr işə başlayır, əks halda isə idarə növbəti (; simvolundan sonrakı) əmrə verilir.

Məs. a = 4.5 olarsa, onda

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

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

əmərlərindən sonra x = 3 qiymətini alır.

3. Şərtsiz keçid əmri –**goto** əmərlərin növbəsini, lazımı nişanlar üzrə dəyişdirmək üçün işlədilir. Bu əmr aşağıdakı formada yazılır:

goto <nişan>;

Nişan proqramda istənilən əmrin qarşısında qoyula bilər və əmrdən “:” simvolu ilə aşağıdakı kimi ayrılır.

<nişan> : <əmr>;

Proqramda işlənəcək nişanlar isə əvvəlcədən **label** bölməsində:

label <nişanların siyahısı>;

formada sadalanmalıdır:

Nişanlar -adla və ya 10000 dən kiçik natural ədədlə (nömrə ilə) ifadə oluna bilər:

Məsələ 4. Verilmiş əmsallara görə kvadrat tənliyi 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 -sqr(d))/(2*a);  
      x2 := (-b +sqr(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.
```

! -yığma əmr tərkibində və ya içproqram daxilində olan əmrlərə kənarından növbə ötürmək olmaz.

Çalışma 4. Verilmiş əmsallara görə ikidəyişənli xətti tənliklər sistemini həll etmək üçün proqram yazın.

5. Seçmə əmri **case** –aşağıdakı formada yazılır:

<pre>case <ifadə> of <1 -ci qiymətlər siyahısı> : <1 -ci əmr>; <n -ci qiymətlər siyahısı> : <n -ci əmr>;</pre>
--

```
else <n+1 –ci əmr>  
end;
```

Burada ifadə:

-yalnız tam, simvollu, sadalanan, və ya məntiqi tipli dəyişənlərdən ibarət ola bilər.

Qiymətlər siyahısı –ifadənin tipinə uyğun sabitlər və ya diapazonlardan ibarət olub elementləri bir-birindən vergüllə ayrılır.

else <n+1 –ci əmr> -köməkçi hissədir və buraxıla da bilər.

case əmrində –ifadənin qiyməti hansı nömrəli siyahıda olursa, fəaliyyət növbəsi həmin nömrəli əmrə keçir. Əgər ifadənin qiyməti heç bir siyahıda tapılmazsa onda növbə n+1 –ci əmrə, bu əmr olmadıqda isə “;” simvolundan sonrakı əmrə keçir.

Məsələ 5. Tutaq ki, «Məktəbli» -dükanındakı mal çeşidləri üçrəqəmli ədədlərlə nömrələnərək ayrı-ayrı qiymətlər üzrə

Çeşidlər	Qiymətlər(manatla)
1, 2, ... , 23	12 000
24, 25, ... , 51	9 500
52, 53, ... , 167	7 200
168, 169, ..., 741	500
742, 743, ..., 980	250

kimi qruplara bölünmüşdür. Alıcının istədiyi m -ci çeşiddən olan n sayda malın hesabını ekrana verə bilən program 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.');
```

```
end.
```


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

Sizin hesab 50400 manat oldu.

cavabını çıxarmalıdır.

Çalışma 5. Məlumat bürosunda, 20 adda məntəqə üçün bilet istəyən sərnəşinlərə xidmət edəcək sadə bir proqram nümunəsi tərtib edin.

§5. Dövrələr.

Dövr –Proqram fəaliyyətində müəyyən əməliyyatın sonlu sayda təkrarlanmasıdır. Dövrü –məqsədəuyğun ardıcılıqda düzülmiş əmrlər dəstəsinin tətbiqilə yaratmaq mümkündür. Bu sahədə *if*, *goto* və xüsusi dövr əmrləri tətbiq olunur. Tətbiq formasına görə dövr əmrləri :

- **for** - parametrlə
- **while** -şərtünü
- **repeat** -şərtsonu variantlarına bölünürlər.

1. *Parametrlə for əmr* –iki istiqamətdə tətbiq olunur:

- a) İrəliyə addımla:

```
for <parametr> := <1 –ci ifadə> to <2 –ci ifadə> do <iç əmr>;
```

-formada verilir və parametr –tam, simvollar, məntiqi və sadalanan tiplərdən birinə aid dəyişən olub ilk və son qiymətləri 1 –ci və 2 –ci ifadələrlə verilir.

Bu əmrdə –parametr 1 –ci qiyməti 2 –ciden kiçik(və ya bərabər) olarsa iç əmr işləyir və 1 –ci qiymət 1 vahid artaraq proses təkrarlanır. Nəhayət 1 –ci qiymət 2 –ciden böyük olunca idarə **for** əmrindən sonrakı əmrə keçir. Məs. Əgər $x = 2$ olarsa, onda

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

əmrində x dəyişəni $2+5+6+7+8 = 28$ və y isə $4 * i = 32$ qiymətini alaraq idarə **for** əmrindən sonraya keçir.

- b) Geriyə addımla:

```
for <parametr> := <1 –ci ifadə> downto <2 –ci ifadə> do <iç əmr>;
```

-formada, analoji prinsiplə verilir.

Burada -a) bəndindən fərqli olaraq, parametrin 1 –ci qiyməti 2 –ciden böyük(və ya bərabər) olarsa iç əmr işləyir və 1 –ci qiymət 1 vahid azalaraq proses təkrarlanır. Nəhayət 1 –ci qiymət 2 –ciden kiçik olunca idarə **for** əmrindən sonrakı əmrə keçir. Məs. Əgər $x = 2$ olarsa, onda

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

əmrində x dəyişəni $2+8+7+6+5 = 28$ və y isə $4 * i = 20$ qiymətini alaraq idarə **for** əmrindən sonraya keçir.

Məsələ 6. Müəyyən natural sırayla verilmiş unsiya ölçülü kəmiyyətlərin qramla ifadəsini əks etdirən cədvəl tərtib etmək üçün proqram yazmalı.

Unsiya ölçülü kəmiyyəti -unsiya, qramla ifadə olunmuş qiyməti qram, sıranın dəyişmə addımı -h və dəyişmə sayını - k ilə işarə etsək, tələb olunan proqramı aşağıdakı kimi tərtib edə bilərik:

```
program Unsiya_Qram;  
uses Crt;  
const line = '-----';  
var  
    unsiya, qram, h: real;  
    k: integer;  
begin  
    clrscr;  
    write('İlkin 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 məsələni **for** əmrinin **downto**(geriyə addım) variantını tətbiq etməklə həll edin.

Məsələ 7. Cəm və hasildən ibarət olan

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

ifadəsinin qiymətini hesablamaq üçün proqram tərtib etməli.

Biz cəm ifadəsini – c, hasil ifadəsini -h və faktorial ifadəsini -f ilə işarə etsək, tələb olunan proqramı aşağıdakı kimi tərtib 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 məsələni **for** əmrinin **downto** variantını tətbiq etməklə

$$y = \sum_{i=5}^{20} 3i^3 + \prod_{i=5}^{20} i!$$

ifadəsi üçün həll edin.

2. Şərtöni –*while* əmri, aşağıdakı formada tərtib olunur:

```
while <məntiqi ifadə> do <iç əmr>;
```

while –əmrü məntiqi ifadə doğru olana qədər proqramı iç əmrin idarəsində saxlayır və yalana çevrildə isə özündən sonraya ötürür.

Məs.

- Əgər $a = 3$ və $b = 7$ olarsa onda:

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

əmrindən sonra a və b kəmiyyətləri uyğun olaraq $a = 7$ və

$b = (7+3)+(7+4)+(7+5)+(7+6) = 46$ qiymətlərini alar.

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

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

kimi işləmə bilər

- **real** tipü üçün aşağı sərhəddin ikiye qatını tapmaq üçün **while** əmri:
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.
```

kimisi 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 dəyişənləri üçün seçilmiş intervalları dəyişərək 8 –ci məsələni həll edin.

Məsələ 9. İstənilən x ədədi və $e = 0.001$ olmaqla, n –ci həddi

$a_n = (-1)^n (2x)^n / n!$ kimi ifadə olunan $\{a_n\}$ –ədədi ardıcılığının

$|a_n| > e$ şərtini ödəyən hədlərinin cəmini tapmaq üçün proqram yazmalı.

$n!$ –kəmiyyətini fact, cəmi toplam və s. kimi işarə edərək məsələni həll etmək üçün proqramı aşağıdakı kiki tərtib 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 məsələni, $e = 0.0001$ ədədi üçün, daha cəməməli yolla həll etməyə çalışın.

3. Şərtsonu variantda tətbiq olunan –repeat aşağıdakı formada tərtib olunur:

```
repeat <iç əmrlər> until <məntiqi ifadə>;
```

repeat əmri -məntiqi ifadə dođru oluncayadək idarəni iç əmrlərdə saxlayır və sonra isə özündən sonrakı əmrə ötürür.

Məs. $x = 16$, $y = -3$ -ilkın qiymətləri üçün:

```
repeat  
  y := y + x;  
  x := 1/2 * x;  
  z := x - 3
```

```
until x <= 5;
```

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

qiymətlərini alar.

Məsələ 10. Arqumentin - $[-\pi, \pi]$ parçasında, $h = \pi / 5$ addımla dəyişməsinə görə, $y = \sin(x)$ funksiyasının müvafiq qiymətlər cədvəlini tərtib edin. Bu cədvəldən $\min(y)$ və $\max(y)$ kəmiyyətlərini tapıb nəticələri 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. Arqumentə aiddə aralıdı dəyişməklə, sonuncu məsələni $y = \cos(x)$ funksiyası üçün həll edin.

§6. Massivlər.

1. Massiv və ya **array** –Eyni baza tipinə mənsub olub, yalnız sıra nömrəsinə görə fərqlənən, eyni adlı elementlər dəstidir. Bu cür elementlər birqayda olaraq, fəal(operativ) yaddaşın ardıcıl yuvalarında saxlanır,

Massivlər –biröçlü və çoxöçlü olmaqla fərqlənir; məs. eyni tipli kəmiyyətlərdən tərtib olunmuş hər-hansı çoxsətirli cədvəli ikiöçlü massiv hesab etmək olar.

Proqramda massiv tipini -ümumi şəkildə aşağıdakı kimi vermək olar.

array [<ölçü>] of <baza tipinin adı>;

Burada ölçü –çox vaxt diapazonla və bəzən də hər hansı sadalanan üsürlər tipinin adıyla verilə bilər.

Massivin tipi, dəyişən, massivin adı və s. **-type, const** və **var** bölmələrində aşağıdakı kimi verilə bilər.

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;

- Birinci sətirdə –diapazon ölçülü, real tipli «onluq» massivi təyin edilir.

- İkinci sətirdə isə həftənin günlərini təmsil edən, sadalanan tipli yeddi adlı massiv təyin edilir. Bu halda, yeddi tip - mon, tue, wed, the, fri, sat, sun dəyərlərindən birini ala bilər.

- Üçüncü sətirdə onluq tipli yeni bir sabit massivi təyin edilir. on sabitinin ala biləcəyi dəyərlər mötərizədə sadalanır.

- Dördüncü sətirdə -hər biri onluq tipindən olan a və a1 adlı iki dəyişən təyin edilir ki, proqramın iş prosesində bu dəyişənlər - a[1], a[2], ..., a[10]və ya a1[1], a1[2], ..., a1[10] kimi qiymətlər ala bilər.

- Beşinci sətirdə təyin edilən b dəyişəni yeddi massivi əsasında -b[mon], b[tue], ..., b[sun] kimi integer tipli 7 qiymət ala bilər.

- Altıncı sətirdə təyin edilən - c dəyişəni massiv tiplidir. Hər biri char tipli olan 100 dəyişəndən hər birinə c[1], c[2], ..., c[100] kimi müraciət edilə bilər.

Massivlər arasında yalnız qiymətvermə əməli təyin edilir. Məs. a := a1 əmri a1-in bütün qiymətlərini a –nın müvafiq elementinə verir. Massivlərin elementləri üzərində isə, əlbəttə digər əməllər də təyin edilmişdir.

Massiv elementlərinə müraciət aşağıdakı formada tərtib edilir:

<massivin adı>[<elementin nömrəsi>]

Elementin nömrəsi bəzən indeks kimi də ifadə olunur və kvadrat mötərizə içində yazılır. Məs. a[1] – a massivin 1 –ci elementi və b[tue] –b massivin tue indeksli elementi (massivdə ikinci element) deməkdir.

Məs. İlk 100 müsbət tam ədəddən ibarət massiv quraraq onun elementləri cəmini aşağıdakı kimi 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 massivin müəyyən elementlərini axtarmaq lazım gəlir ki, bu iş -dövr və budaqlanma əməllərinin axtarış şərtləri əsasında həyata keçirilir.

Məsələ 11. ATS –ə k –cı saniyədə daxil olan zənglərin sayını $y(k)$ kimi ifadə edək. Tutaq ki, bu ifadənin qiyməti:

$y(k)=\text{trunc}(\text{abs}(7*\text{Sin}(k)))$ kimi təsadüfi funksiya ilə təyin olunan 0, 1, .. ,6 ədədlərindən biridir. ATS –ə 10 dəqiqə ərzində gələn zənglərin sayını və 1 dəqiqə ərzində daxil olmuş maksimal zəng sayını hesablayaraq nəticələri ekrana çıxara biləcək proqram yazmalı.

Zənglər sayını sadəcə say və s. müvafiq işarələr qəbul etməklə bu proqramı aşağıdakı kimi tərtib 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 məsələni həll edərkən təsadüfi ədəd almaq üçün randomize prosedurasından aşağıdakı kimi istifadə edərək:


```
randomize;  
y[i]:=random(7);  
başqa bir proqram yazın.
```

Məsələ 12. Elementləri $y(k) = \ln(k) - 3$; $k = 1, 2, \dots, 10$ düsturu ilə hesablanan y massivini qurub bu massivdən mənfi elementlərindən ibarət g -iç massivini ayıran və nəticələri ekrana çıxara bilən proqram tərtib etməli.

Bu proqramda $-g$ massivini elementlərinin sayı n ilə işarə edilir.

```
program İçmassiv;  
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 -ci məsələni $y(k) = \lg(k/100) + 1$; $k = 1, 2, \dots, 120$. ifadəsi üçün həll edin.

2. *İkiölçülü massivlər* – əsasən cədvəllərlə işləmək üçün nəzərdə tutulur və elementləri matris üsuluyla düzülərək qoşa indekslə nömrələnir. Bu indekslər aid olduğu elementin uyğun olaraq, sətir və sütun nömrələrini göstərir. Məs. $A[3,2]$ elementi $-A$ massivinin 3 -cü sətirində 2 -ci elementdir.

Nümunə üçün bəzi massivlərin verilmə üsullarına baxaq:

- Qiymət balı üçün 3 sətir və 4 sütunlu «Bal» adlı bir massiv:

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 tipli «İl» massivini:
var İl: **array**[1..31] **of array**[1..12] **of char**;

- İki rəqəmli ədədlər arasında vurma cədvəlini yazmaq üçün 89 x 89 ölçülü «Vurma» massivini:

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 tanıtıla bilər.

Məsələ 13. İki rəqəmli 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, iti bucağın 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

$a_{ij} = 2|\sin(i)| + j$, $i, j = 1, 2, \dots, 5$ 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 ekrana 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 məsələni 6 növ metaldan 4 adda xəlitə almaq üçün dəyişdirərək həll edin.

Məsələ 15. Elementləri $x_{ij} = i + j^2$, $i, j = 1, 2, \dots, 5$ düsturu ilə hesablanan X massivini və onun 20 –dən böyük elementlərindən yeni bir Y massivini tərtib edərək X, Y massivlərini ekrana çıxara bilən program 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. Elementləri $a_{ij} = (i+1).(j+1)$, $i, j = 1, 2, \dots, 10$ düsturu ilə hesablanan a massivini və onun ikirəqəmli elementlərindən ibarət bir b altmassivini tərtib edərək ekrana verə bilən program yazın.

§7. Sətirlər (string)

Sətir tipli ünsürlər –ixtiyari çeşiddə simvollardan ibarət məhtud sayda (255 –ə qədər) char tipli elementlər dəstidir. Hər belə dəst bir cüt apostrof işarəsi arasında –məs. ‘Bizim diyar – Odlar yurdu!’ və boş sətir isə ‘ ’ (arada yalnız “ara” simvolu olmaqla) kimi verilir.

Sətir tipli dəyişənlər –**var** əmrini bölməsində string sözüylə aşağıdakı formada tanıdılır.

var <dəyişən> : string[n]

Burada n –verilmiş sətir tipli dəyişənin simvollar sayı olub zəruri göstərici sayılmır. Məs:

```
const dil = 'Pascal 7.0 asan dildir';  
var ad : string[10];  
ocean : string;
```

formada təyin olunmuş **dil** sətir sabitinə '**Pascal 7.0 asan dildir**' qiyməti verilir, **ad** dəyişəni ən çoxu 10, lakin **ocean** dəyişəni isə 255 –ə qədər simvoldan ibarət olan qiymətlər ala bilər.

Simvol tipli dəyişənlər arasında: birləşmə (+) və müqayisə (<, <=, >, >=, =, <>) əməlləri təyin olunmuşdur. Müqayisə zamanı sətirlər (eynilə müsbət-tam ədədlərdə olduğu kimi) soldan sağa, ilk fərqli simvoladək yoxlanır və kompüterin kodlar cədvəlində kodu böyük olan simvol böyük sayılır; hərflərin kodu isə bu cədvəldə əlifba sırasına uyğun olaraq artır.

Məs. '**a**' < '**u**', olduğundan '**Şahbaz**' < '**Şahbuz**' və '**s**' < '**t**' olduğu üçün '**Abbat**' > '**Abbas**' və s.

Əgər **ad1** = '**Qara**' və **ad2** = '**Yusif**' olarsa, onda

ad3 := ad1 + ad2 əməlinədən sonra **ad3** = '**Qara Yusif**' qiyməti almış olar ki, burada **ad1** < **ad3** < **ad2** sayılır.

Simvolların kodu **ord** funksiyasıyla - **ord('A')**=65, **ord('B')**=66 kimi, əks əməl isə **chr** funksiyasıyla **chr(67)**='C'-kimi müəyyən edilir. Məs. Latın hərflərinin kodlar siyahısını ekrana çıxarmaq üçün **ord** funksiyasını aşağıdakı formada tətbiq etmək olar:

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

İndi isə sətir tipli ünsürlərlə işləməyə imkan verən -*standart funksiyaları* və dəyişənlə verilən sətirlərlə işləmə imkanı verən -*proseduraları* nəzərdən keçirək.

Standart funksiyalar.

length(<sətir>)	sətrin simvol sayını verir;
copu(r, m, n)	r –ci sətirin m –ci dən başlayaraq n simvolunu verir.
concat(r1,r2,..,rn)	r1, .. , rn nömrəli sətirləri bir sətir halında birləşdirir.
pos(r1, r2)	r1 sətirinin r2 –yə daxil olmağa başladığı simvolun nömrəsini verir.

Məs. **ocean1**='Sakit okeanda adalar boldur', **ad4** ='Atlantik' qiymətli dəyişən sətirlərə bu funksiyaları aşağıdakı formada tətbiq etsək:

<i>Verilən funksiya</i>	<i>Alınan cavab</i>
length(okean1)	27;
copu(okean1, 15,6)	'adalar';
concat(okean1, ',', ad4, 'okeanda az.')	'Sakit okeanda adalar boldur, Atlantk okeanda az.';
pos('bol', okean1)	22;

kimi cavablar alınar.

Proseduralar.

insert(r1, <dəyişən>, n)	dəyişənlə verilmiş sətər, n –ci mövqedən başlayaraq r1 sətəri də daxil edilir;.
delete(<dəyişən>, m, n)	dəyişənlə verilmiş sətirin m –ci mövqedən başlayan n simvolu silinir.
str(<ədəd>, <dəyişən>)	Ədədi sətir tipli ünsürə çevirir.
val(r1, s1, s2)	r1 sətirinin ədədi obrazını s1 ədədi dəyişəni içərisinə göndərir. Bu mümkün olarsa s2 – ədədi dəyişəni, əks halda isə buna əngəl olan ilk simvolun ədədi obrazı 0 –a çevrilir.

dil, okean1 və ad4 dəyişənlərinə bu proseduraları tətbiq etməsək:

<i>Prosedura</i>	<i>Cavab</i>
insert('vulkanik ', 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

kimi əməliyyatlar aparmaq olar.

Beləliklə -string tipi üzərində həm bütöv sətirlər halında və həm də onları simvol massivi kimi qəbul edərək elementlərə ayırmaqla işləmək mümkündür. Bu əməliyyatları isə əsasən, ekranla dialoq rejimində aparmaq münasibdir.

Məs. Sual – cavab qurmaq üçün

```
sual := '-Siz Paskal dilini bilirsinizmi?';
```

```
writeln(sual);
```

```
for i := 1 to 23
```

```
    read(cavab[1]);
```

```
əmrİ əvvəl ekrana
```

```
-Siz Paskal dilini bilirsinizmi?:
```

```
sualını çıxarıb ona 23 simvolla bir cavab yazmağa imkan verərdi.
```

Məsələ 16. (Mətni kodlaşdırma.) Verilmiş a -mətnində bütün aralıqları, nöqtə və vergülləri ləğv edərək qalan simvolları əvəzləyərək nəticəni ekrana verə bilən program 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 ədədi və əlifba sırasındakı hərfləri ardıcıl olaraq nömrələyib, verilmiş mətnin hər bir hərfini eyni nömrəli sadə ədədlə əvəz edən program yazın.

Məsələ 17. Verilmiş «Yazda» adlı mətnə, səhv getmiş «qulaq» sözünü «bulaq» sözü ilə əvəz edən program 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('yazılacaq 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!’ sətirinin simvollar sayını və sətirin 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** prosedurasını 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

praktik proqramların isə nə qədər çoxşaxəli və əndazəsiz olduğunu təsəvvür etmək çətin deyil. Bu cür proqramların mürəkkəb alqoritm və strukturası da öz növbəsində *-struktur proqramı* deyilən bir konsepsiyanın tətbiq olunmasını tələb edir.

Struktur proqramları –özündə əsasən aşağıdakı prinsipləri cəmləşdirir:

- Qarşıya qoyulmuş məsələnin analizi və həll alqoritminin ayrı-ayrı sadə hissələrə bölmək.

- Bütün hissələri detallaşdıraraq müvafiq prioritet əsasında içproqramlar tərtib etmək.

- İçproqramlarda –sadə, budaqlanma, dövr, vaxttənzimləyici və s. kimi baza əmrlərini səmərəli istifadə etmək.

- Proqramı abonentlərə aydın olan bir formada yazmaq.

- Şərtsiz keçid əmrlərindən mümkün qədər az istifadə etmək.

- Proqram fəaliyyətini iş ərəfəsində və iş prosesində məntiqi baxımdan yoxlamaq və s.

Burada detallaşma, əsasən -alqoritm yuxarıdan aşağıya mərhələ – mərhələ şaxələnməsi və addım-addım yoxlanması prinsipi üzrə həyata keçirilir.

Proqramda bu və ya digər bir sahəni əhatə edən içproqramlarda hansı baza əmrlərini işlətməyin səmərəli olacağı da müəyyən edilməlidir. Onu da deyək ki, istənilən dərəcədə mürəkkəb alqoritmlərin –sadə, budaqlanma və dövr bəndləri əsasında qurula bilməsi artıq sübuta yetirilmişdir.

Struktur proqramlarında sistemsiz keçid əmrləri proqramın oxunuşunu çətinləşdirdiyi üçün onların mümkün qədər **if-then-else, case, while** kimi əmrlərlə əvəz olunması daha səmərəli sayılır.

Zəruri hallarda proqramın idarəsini bəzi əmrlərdən almaq üçün **-exit, break, continue** və **halt** kimi çıxış əmrləri tətbiq olunur.

Proqramın işinə nəzarət etmək üçün, əsasən dəyişənlərin qiymətlər cədvəlini tərtib etmək üsulundan geniş istifadə edilir.

Struktur proqramlarını təşkil edən içproqramları istənilən nahiyədən və lazımı dəfə çadırmaq olar. Qeyd edək ki, içproqramlar əsasən iki cür: *funksiya* - *ichproqram* və *prosedura* - *ichproqram* kimi fərqlənir. Proqramda **System, Crt, Dos, Graph** standart modullarının –**procedure** və **function** bölmələrində təqdim olunan standart *abonent içproqramlarından* istifadə etmək isə olduqca sərfəlidir.

2. Procedure –içproqramı:

```
procedure <ad>(<formal parametrlər siyahısı>);  
    <proseduraların təqdimat bölməsi>;  
    beqin  
    <prosedura əmrləri bölməsi>;
```

end;

formada tərtib olunur. Burada:

- formal parametrlər siyahısında dəyişən və onların tipləri sadalanır.
- verilən ünsürlər parametr–arqument, alınan cavablar isə parametr–nəticə adlanır və bu parametrlər dəyişənlər kimi **var** bölməsində sadalanmalıdır.

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

```
procedure Tel_hesab(k:integer; var c:real);
```

```
  begin
```

```
    c := k*500;
```

```
    c := c + 0.2*c;
```

```
  end;
```

kimi bir içproqram yazmaq olar ki, burada: k –danışıq müddətini ifadə edən *arqument* -və ya parametr - arqument, c -isə k müddət üçün yekun danışıq haqqı - və ya parametr - nəticədir.

Proqramın istənilən nahiyəsindən və ya digər bir içproqramdan procedure əmrinə aşağıdakı kimi müraciət etmək olar:

<proseduranın adı> (<faktiki parametrlərin siyahısı>);

Qeyd edək ki, müraciət əmrlərində sadalanan bu cür faktiki para-metrlər də arqument və nəticə kimi fərqlənilir. *Faktiki arqumentlər* –sabit, dəyişən və ya ifadə formasında ola bildiyi halda, *faktiki nəticə* –yalnız dəyişən ola bilər və onların tipi müraciət əmrində verilmir.

Müraciət əmrində -verilən hər faktiki arqumentin qiyməti procedure əmrində sadalanan müvafiq formal arqumentə verilir, sonra isə, hər faktiki nəticə müvafiq formal nəticənin qiymətini mənimsəyir. Əlbəttə bu zaman uyğun faktiki və formal parametrlərin adları eyni olmaya bilsə də tipləri bir-birinə uyğun olmalıdır.

Əsas proqramın **var** əmrilə verilən dəyişənlər *global dəyişənlər* olub istənilən içproqramda belə işləyə bilər, lakin procedure içproqramında verilən dəyişənlər isə *lokal dəyişənlər* sayılır və yalnız öz tərkibində işləyə bilər. Başqa sözlə, procedure əmrində *global dəyişənlər* tanınır və deməli burada formal nəticə parametrlərini sadalamağa ehtiyac qalmır.

Məsələ 19. ATS –ə daxil olan zənglər sayı ilə bağlı verilən 11 –ci məsələni aşağıdakı proseduraların köməyi ilə həll edək:

- 1) Bir_say –hər saniyədə daxil olan zənglərin sayı.
- 2) On_say – ilk 10 saniyədə daxil olan zənglərin sayı.
- 3) Max_say -bir saniyədə daxil olan maksimal zəng sayı.

Tutaq ki, k –cı saniyədə daxil olan zənglərin sayı $y(k)=\text{random}(k)$ təsadüfi funksiyası ilə təyin olunan tam ədəd olsun və

- b.ı.r saniyədə daxil ola biləcək zənglər sayını –birsay
 - bu saylardan ən böyük olanı -maxsay
 - 10 saniyədə daxil ola biləcək cəmi zənglər sayını –onsay
- ilə işarə etməklə bu proqramı aşağıdakı kimi tərtib 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 saniyədə daxil olan zənglər sayını, metro keçidindən hər dəqiqədə keçən sərnişinlər sayı ilə əvəz etməklə dəyişdirərək alınan məsələni həll edin.

3. Function –içproqramı:

```
function <ad>(<formal parametrlwr>;  
                <funksiyanın tipi>;  
            <funksiyanın təqdimat bölümü>;  
begin  
    <ad := ifadə>  
end;
```

formada tərtib olunur və proseduradan fərqli olaraq xitab bölümünə yalnız bir standart tipli cavab qaytarır. function içproqramına aşağıdakı formada xitab edilir

```
<ad> (<faktiki parametrlər>);
```

və cavablar əsas proqrama, standart funksiyalarda olduğu kimi - funksiyanın adıyla qaytarılır.

Məs. $\text{tg}(x)$ –in qiymətini almaq üçün **function** tərtib edərək $\text{tg}(x) + \text{ctg}(x) + \text{tg}^2(x)$ ifadəsini aşağıdakı proqramla hesablaya bilərik.

```
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 ki, boyaqçı emalatxanasında 6 növ rəngdən istifadə edərək 10 çeşiddə boya hazırlanır və j –ci çeşiddə vurulan i –ci növ rəngin mayası (məsrəfi)

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

Elementləri m_{ij} olan m massivini tərtib edərək 3 –cü rəngdən minimal miqdarda tələb edən çeşidi tapıb nəticələri ekrana çıxaran struktur proqramı tərtib etməli.

Proqramda i –ci çeşidi –imin və ona işlənən 3 –cü növ rəngin minimal miqdarı –min kimi işarə edilir.

```
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, 'növu');
writeln('daha az maya aparır');
readln;
end;
begin
  clrscr;
  Mass(a);
  Minimal(m, imin);
  readln;
end.
```

Çalışma 20. 14 –cü çalışmanı struktur proqramı tərtib etməklə həll edin.

Əlbəttə bu cür - tədris xarakterli, sadə proqramlardan fərqli olaraq, əməli məqsəd daşıyan -çox böyük və mürəkkəb proqramlar adətən kollektiv işin məhsulu olub çoxsaylı praktik yoxlamalardan keçirilir. Təsadüfi deyildir ki, Microsoft Corporation, IBM Corporation və s. bu kimi nəhəng firmalarda, ayrı-ayrı sahələr üzrə yüzlər və bəzən də minlərlə mütəxəssis çalışır. İstehsalatın, nəqliyyatın bu və ya digər sahələrinə xidmət edən bir çox proqramlar, bəzən hətta ümumdövlət səviyyəli bir məsələ kimi qarşıya qoyularaq bir çox struktur proqramlarından ibarət sistem kimi işlənilib hazırlanır.

4. *Rekursiv funksiyalar* –əsasən functionların öz-özünə xitab etməsilə təşkil olunan dövrü proses kimi təzahür edir. Başqa sözlə desək, *rekursiya* – funksiya qiymətlərini özünə argument kimi təqdim etməklə əldə edilir. Rekursiyanın təşkilində *axın* anlayışı da geniş tətbiq olunur. *Axın* –fəal yaddaşda, qəbul etmə ardıcılığına əks növbəylə xaric etmə prinsipi üzrə saxlanan elementlər toplusudur.

Məs. m –dən n –ə qədər tam ədədlərin cəmini hesablamaq üçün aşağıdakı kimi rekursiv funksiya qurmaq olar.

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

Bu içproqram, məs. Toplama(3,5)ifadəsini aşağıdakı ardıcılıqla hesablayır:

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

Yəni, blok toplanmanı iki etapda yerinə yetirir: -əvvəlcə 5, 4, 3 ədədlərindən axın təşkil edir, -sonra isə onları tərs sırayla toplayır.

Məsələ 21. İlk n natural ədədin hasilini $n! = n(n-1)! -$ düsturunu ($0!=1$, $1!=1$) ardıcıl olaraq tətbiq etməklə hesablamaq üçün rekursiv funksiya qurmalı.

```
function Vurma(n:integer):integer;
```

```
begin
```

```
    if n =0 then Vurma:=1
```

```
    else Vurma := n * Vurma(n-1)
```

```
end;
```

Bu functionla $4!$ -in qiyməti aşağıdakı 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 daxil olub hasil $1.1.2.3.4= =24$ kimi ardıcılıqla yerinə yetirilir.

Çalışma 21. Birinci həddi 30, 8 -ci həddi 46 olan ədədi silsilənin ilk n həddinin cəmini tapmaq üçün rekursiv funksiya daxil olan proqram yazın.

5. *Açıq massivlər* -içproqramların formal parametrləri siyahısında verilən, ölçüsü əvvəlcədən bəlli olmayan massivlər olub aşağıdakı formada təqdim olunur:

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

Bu cür formal massivlərin elementləri 0 -dan başlayaraq nömrələnir və sonuncu elementinin nömrəsi standart **high** funksiyasıyla:

```
high(<massivin adı>).
```

formada öyrənilə bilər.

Açıq massivlərdən, prosedurada dəyişən ölçülü massivlərlə işləmək üçün istifadə olunur.

Məsələ 22. İçproqramların köməyilə, elementləri uyğun olaraq

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

$n = 1, 2, \dots, 9$ düsturlarıyla hesablanan y və g massivləri qurun. Hər massivdə 4 -dən böyük elementlərin sayını tapıb nəticələri 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(' Axtarılan element sayı y - ');
    writeln(' k=' ,top(y):3);
    write(' Axtarılan element sayı g - ');
    writeln(' k=' ,top(g):3);
    readln
end.
```

Diqqət edin ki, bu proqramda ilk dəfə olaraq. type fu =function(n:integer):real; bölümündə function tipi verilərək fy(x) və fg(x) funksiyaları da bu tipə aid edildi, beləliklə də bir prosedurayla müxtəlif massivlər qurmaq mümkün oldu.

Həmçinin bu səbəbdən proqrama zəruri içproqram modelini(far-model) tənzim edən {\$F+} direktivi daxil edildi.

Çalışma 22. 22 –ci məsələni $y_m = fy(m) = \text{random}(m)$, $m = 1, 2, \dots, 9$ və $g_n = fg(n) = (n + 1)^2 / 2$; $n = 1, 2, \dots, 14$ düsturləriylə hesablanan y və g massivləri üçün dəyişdirməklə həll edin.

6. *Standart modullar* –geniş istifadə olunan universal təyinatlı içproqramlar olub bir «kitabxana» adı altında cəmləşmiş olurlar.

Modul –özündə sabit, dəyişən ünsür tipləri və içproqramlar birləşdirən proqram vahidi olub standart və abonent modulları kimi fərqlənirlər.

Standart modullar.

System	Çoxişlənən prosedura və funksiyalar sist.
String	Sətir dəyişənləri ilə işləyən funksiyalar.
Printer	Printerlə işləmək üçün proqram modulu.
Graph	Qrafikqurma prosedura və funksiyaları.
Overlay	Böyük proqramlarla işləmək üçün modul.
Dos, Windows	Paskal proqramı gedişində Əməliyyat sistemi əməllərinin işləməsinə imkan verir.
Graph3, Turbo3	TP versiyaları arasında uyarlıq saxlayır.

Modulları **uses** əmrilə hər hansı bir proqramın işinə

```
uses <modulların adlar siyahısı>;
```

formada cəlb etmək olur.

Sustem modulundan isə, proqramlar elə-belə də (yəni, ona müraciət etmədən belə -avtomatik olaraq) istifadə edə bilirlər. **read**, **readln**, **write**, **writeln** proseduralarını və **sin**, **cos** və **s**. standart funksiyaları da proqramlar elə məhz bu moduldan alırlar.

Sustem və Crt modullarına aid olan əsas proseduralar.

exit	Əsas proqramın işini saxlamaq və içproqramdan çıxmaq üçün işlənir.
halt	Proqramı saxlayıb idarəni Əməliyyat sisteminə verir.
break	for, while, və repeat dövrlərindən məcburi çıxma.
continue	for, while, və repeat dövrlərində növbəti iterasiyanı başlamaq üçün işlənir.
delay(n)	Cari əmrin işini abonentin verdiyi vaxt (n san) ərzində (mikro san. dəqiqliklə) saxlayır.
clrscr	ekranı təmizləyir.
textcolor	Mətnə 0, 1, .., 15 ədədləri ilə verilən rəng.
textbackground	ekran fonuna rəng verir.

Habelə:

- **Crt** modulunda, -proqramı gözləmə rejiminə keçirməklə abonentin qırpdıdı sədəfin qiymətini alan **readkey** və istənilən sədəfi qırpanda **true** məntiqi dəyərini alan **keypressed** funksiyaları və s. yerləşir.
- Paskal proqramı rejimində fayllar sistemi ilə işləmək üçün nəzərdə tutulmuş prosedura və funksiyalar isə **Dos** modulunda yerləşir.

Onlara, cari proqram daxilindən:

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

kimi müraciət etmək olar.

Lakin **exec** prosedurası işlənən proqrama **M** direktivini də:

–məs. {\$M \$2000,0,1000} parametrlə- qoşmaq lazımdır.

- Proqram və ya onun fraqmentlərinin iş vaxtının xronometriyası, və tarix göstəricilərilə işləmək üçün aşağıdakı proseduralar tətbiq edilir:

```
GetTime(Hour, minute, Second, SotiSec)
```

-{cari vaxta -saat, dəqiqə, saniyə və millisaniyə ölçülərinə müvafiq qiymət verilir}.

```
GetDate(Year, Month, Day, Number)
```

-{ cari tarixə -**Vord** tipi siyahısından ay, gün, həftənin günü göstəricilərinə uyğun qiymətlər verir }

Tapşırıq. Standart modullar və yuxarıda şərh olunan proseduralardan istifadə etməklə bir proqram nümunəsi hazırlayın.

7. *Abonent modulları* –**unit**, müəyyən qaydalara uyğun olaraq aşağıdakı strukturda tərtib olunur:

```
unit <ad>;  
interface {İnterfeys bəndi}  
    <şərh bölməsi>  
implementation {işçi blok}  
    <abonent içproqramları mətni>  
begin  
    <hazırlıq bloku>  
end.
```

Abonent öz moduluna münasib ad seçir. Şərh bölməsində gərəkli modul və içproqramlar göstərilir, sabit, dəyişən və s. ünsür tipləri tanıtılır.

İşçi blokda içproqram mətnləri, şərh bölümündə adlarının sadalanma sırasına müvafiq ardıcılıqla yazılır. Parametrlər artıq başlıqların şərhində verildiyi üçün burada buraxıla bilər. Bir çox proqramlar üçün ümumi olan

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

Hazırlıq bölümündə ilkin verilənləri tanımaq, fayl açmaq və s. mümkündür. Odur ki, bu blok hələ əsas proqramın əmrlərindən əvvəl fəaliyyətə başlayır. Bu blok lazım olmadıdı hallarda işə **begin** əmrini də yazmaıda ehtiyac qalmır.

Məs. $\text{tg}(x)$ 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** dəyəri verilərək translyasiya (Ctrl + F9) edilir. Nəhayət diskdə yalnız **.tpu** genişləməsilə eyni adda fayl alınır.

Məs. **BizimModul** –dan istifadə edərək $\text{tg}(x)$ 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 lazım 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. *Yazı* –**record**: müxtəlif tiplərdən ibarət qarışıq yazıları fəal yaddaşa 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* köməyiylə 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 verilərək:

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

kimi qiymətləndirilir.

Yazının hər hansı alanı ilə təmas -qondarma adın köməyi ilə:

```
<yazının adı>.<sahənin adı>
```

formada qurulur.

Lakin qondarma adların tətbiqi, bir çox hallarda böyük ifadələrin meydana çıxması ilə müşayət olunur. Bu halda birləşdirici **with** əmrindən istifadə etmək daha sərfəlidir.

2. *Birləşdirici with əmri* –proqramda yazı tipinin dəyişəni:

```
with < yazı dəyişəni tipinin adı > do <əmr>;
```

Verilmiş əmrdə -adı çəkilən dəyişən tipinə müvafiq alanların adları verilə bilər. Məs. ötən mäsaldan bir dəyişəni aşağıdakı kimi tərtib 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 massivini tipindən istifadə etməklə, satışda olan avtomobillər haqda məlumat almaq üçün proqram tərtib etməli. Qiyməti 3000\$ -dan az olan modellər və onların buraxılış ili barədə ekrana ayrıca arayış çıxarmalı.

Yazı alanları: model -marka, buraxılış ili -il, və qiyməti -haqq kimi adlandırsaq aşağıdakı proqramı yazıb baxalım.

```
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.
end.
```

Çalışma 23. Ötən proqramda alan tiplərinə *rəng alanı* də əlavə edərək ekrana son yeddi ilin buraxılışı olan qırmızı rəngli avtomobillər barədə məlumat verə bilən proqram yazın.

§10. Fayllar

1. *Tipləri tanımaqla faylların təşkili.* Çox zaman kənar yaddaşda (diskdə) saxlanan iri həcmli məlumatlarla işləmək zərurəti meydana çıxır ki, bu da *fayllar* vasitəsilə həyata keçirilir.

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 siyahısı> = 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 işə bağlamaq lazımdır. Faylın hər-hansı elementi, başdan başlayaraq ardıcıl arama yoluyla tapılır və faylın sonu işə

eof (<faylın adı>);

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

System modulunun aşağıdakı proseduralarıyla fayllarla müvafiq əməliyyatlar aparmaq mümkündür:

<code>assign(<faylın adı>, <faylın kənar adı>)</code>	Faylın kənar adı və fayl dəyişəni arasında əlaqə yaratmaq üçün.
<code>reset(<fayl adı>)</code>	Fayldan lazımı hissəni oxumaq üçün.
<code>read(<faylın adı>, <dəyişənin adı>)</code>	Fayldan lazımı hissəni fəal yaddaşa verir
<code>close(<fayl adı>)</code>	faylı bağlayır
<code>rewrite(<fayl adı>)</code>	fayla yazı daxil etmək üçün onu açır
<code>write(<fayl adı>, <dəyişənin adı>)</code>	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), vinqester 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 fayll 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 əməliyyat yaddaşına hər-hansı fayldan **while**, **read**, **eof** əmrlərilə daxil etməklə 24 –cü məsələni həll edin.

Məsələ 25. Ötən proqramda tərtib olunmuş fayldan, əməliyyat tezliyi 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 fl:myfile;  
    i,n:integer;  
    c1:comp;  
begin  
    clrscr;  
    assign(fl,'d:\computer');  
    {Oxuma faylı ayarlamaq}  
    reset(fl);  
    while not eof(fl)do {fayl sona kimi yoxlanır}  
    begin {fayldan oxuma}  
    read(fl,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 proseduralarından istifadə etməklə sərbəst seçdiyiniz bir faylla əməliyyat aparan sadə proqram tərtib edin.

Ardıcıl arama rejimində başqa, fayllarla birbaşa seçmə yoluyla da işləmək mümkündür. Bu cür faylların birbaşa k –cı elementinə read və write əmrlərini tətbiq etməzdən əvvəl seek əmrilə:

```
seek(<faylın adı>, k);
```

kimi hazırlıq görmək lazımdır.

3. *Mətn faylları*. Yuxarıda tanış olduqumuz -tip formasında müəyyən edilən fayl elementləri kompüter tərəfindən kodlaşdırılır və həmin fayllarda düzəlişlər aparmaq və ya onlara adi mətn redaktorlarını tətbiq etmək mümkün olur. Odur ki, bir çox hallarda bu cəhətdən daha münasib imkanlara malik *mətn fayllarından* istifadə olunur.

Mətn faylının elementləri –hərflər, rəqəmlər, işarələr və ara(probel) kimi simvollar ardıcılığından ibarət olan sətirlərdir. Sətirdə ünsürlər ara ilə ayrılır, hər sətir sonunda keçid sədəfi qırpılır. Belə faylları mətn redaktoru ilə tənzimləmək mümkün olur. Sətirdə simvolun varlığı **coln** əfunksiyası ilə

```
coln(<faylın adı>);
```

formada müəyyən edilir. Sətrin sonunda bu funksiya **true** qiymətini alır. Mətn faylı **var** bölməsində

```
var <dəyişənlərin siyahısı>: text;
```

kimi verilir.

Mətn faylı elementlərini **read** və ya **readln** əmrlərlə

```
read(və ya readln)(<faylın adı>,<parametrlərin siyahısı>);
```

kimi oxumaq mümkündür. readdan fərqli olaraq readln əmri sətrin parametrlər siyahısında verilməyən hissəsini oxumur.

Mətn faylına yazı write və ya writeln əmrləri ilə:

```
write(və ya writeln)(<faylın adı>,<ifadələrin siyahısı>);
```

kimi verilir.

Mətn faylına əlavələr isə **append** əmri ilə

```
append(<faylın adı>);
```

formada daxil edilə bilər.

Məsələ 26. Hər-hansı mətn redaktoru ilə tərtib 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 etməli.

```
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 alanII 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 çayın adı-',adlmax);  
readln
```

end.

Çalışma 26. Azərbaycan çayları üçün BizimÇaylar adlı fayl tərtib etməklə ötən məsələni həmin fayl üzündə həll edin.

4. *Ünsürlərin çeşidlənməsi*. Massiv və fayl tərkibində çeşidləmə alqoritmlərindən *-yerdəyişmə ilə artım* üsulunu nəzərdən keçirək. n element üçün bu üsul:

1- ci mərhələdə

- birincidən başlayaraq ilk element cütü müqayisə olunur və birinci böyük olan halda yerdəyişmə aparılır.
- ikincidən başlayaraq proses təkrarlanır və cəmi n-1 addımdan sonra ən böyük element sona qoyulmuş olur

2 – ci mərhələdə

- 1 –ci mərhələni, n-2 –addımda təkrarlamaqla ikinci böyük ədəd n-1 – ci mövqeyə qoyulmuş olur.

.....

n-1 –ci mərhələdə, nəhayət sonuncu bir cüt element də müqayisə olunur və lazım gələrsə yerdəyişmə aparmaqla çeşidləmə sona yetir.

Məsələ 27. 26 –cı məsələdə verilmiş faylı ekrana çıxararaq onu massiv strukturlu ünsür kimi fəal yaddaşa ötürməli. Daha sonra, fayldakı çay adlarını əlifba sırasına görə düzməklə yenidən ekrana verməli.

```
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 massivə strukturlu ünsür kimi fəal yaddaşa ötürməklə 27 –ci məsələni həll edin.

§11. Siyahılar

Məlum olduğu kimi, proqramda işlənən hər bir dəyişən üçün fəal yaddaşa 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 bəlli olmur. Təbii 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. *Dinamik yaddaş*, *dinamik dəyişən* və *ox*. Yaddaşdan dinamik istifadə – dəyişənlərə yalnız zəruri anlarda yer ayıraraq lazım olmadıqda isə onu boşaltma prinsipi ilə xarakterizə olunur. Bu şəraitə uyğun seçilmiş dəyişənlər *dinamik dəyişənlər* adlanır. Dinamik dəyişənlərlə işləmək üçün *ox* dediyimiz xüsusi ünsür tipi müəyyən edilmişdir. *Ox* –təhkim olunduğu dəyişənin yaddaşda yerini deyil yalnız onun tipini müəyyən edir və yalnız zəruri anlarda ona yaddaşdan yer ayrılmasını təmin edir. *Ox -type* bölümündə ^ simvoluyla

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

kimi şərh olunur.

Bəlli 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 oxların özü isə cüzi yer tutar. Dəyişənlərə yalnız onların gərəkli olduğu anlarda, oxların bildirişi ilə

```
new(<dəyişənin oxu>;
```

prosedurası əsasında yer ayrılacaq bilər və yalnız bundan sonra

```
<dəyişənin oxu>^
```

adlı dinamik dəyişən tanıtıla bilər.

Oxlar üzərində

```
<1 -ci ox> := <2 -ci ox>;
```

```
<ox> := nil;
```

kimi ünvan dəyişmə 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

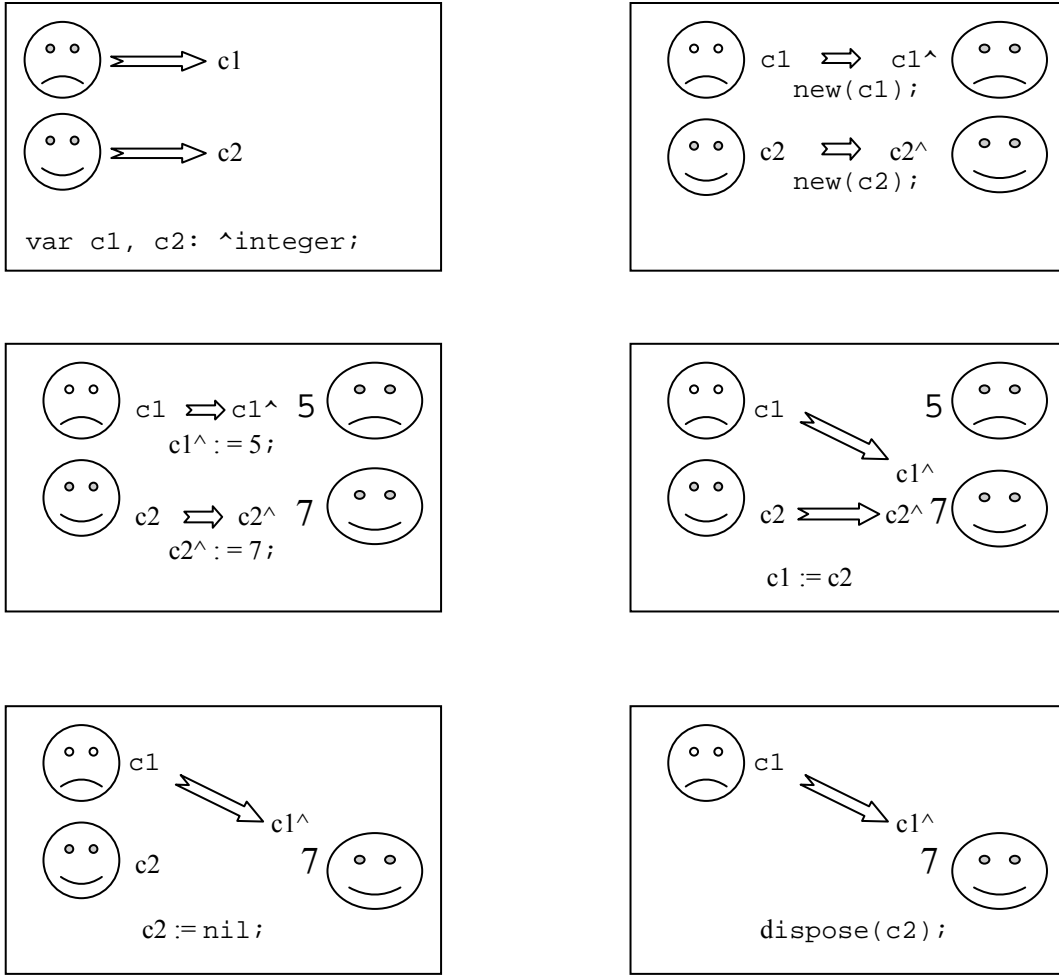
<code>dispose(<dəyişənin oxu>).</code>
--

kimi tərtib olunmuş dispose prosedurasıyla 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 yaddaşdan 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 yeni 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 tiptən 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 <siyahı elementinin adı> = ^<yazı>;
    <yazı> = record
    <l -ci ünsür alanı> : <l -ci ünsür tipi>;
    ...
    <n -ci ünsür alanı> : <n -ci ünsür tipi>;
    <ox alanı> : <siyahı elementinin adı>
end;
```

Məs. 26 -cı məsələdə verilmiş Çaylar siyahısını -çaylar, elsiya(siyahı elementi)yazı tipini çay adlandırsa, onun üçün tip və dəyişənləri aşağıdakı kimi təyin etmək olar:

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

Burada -elox, elbir, elqabaq, elsonra dəyişənləri -elsiya yazı tipindən olmaqla:

- elox -siyahıda növbəti elementi göstərən ox
- elox^ -yazı tipinin dinamik dəyişəni olarsa onda: müvafiq ifadələr
- elox^.uz -tam ədəd tipli dinamik dəyişən (çayın uzunluğu)
- elox^.iz -siyahıda növbəti elementin oxu
- elox^.iz^.uz -növbəti çayın uzunluğu və
- elox^.iz^iz isə -daha sonrakı çayın göstəricisi və s. kimi xarakteri-zə edilir.

Məsələ 28. Tutaq ki, diskdə yerləşən hər hansı fayla yeni bir ünsür əlavə etmək lazımdır. Mövcud faylın bazası əsasında, daxil edilməli ünsürlə başlayan yeni fayl tərtib etməli.

Məsələni həll etmək üçün: əvvəlcə ünsürləri fayldan fəal yaddaşa gətirmək və lazımı düzənləmə işlərindən sonra kənar yaddaşda yeni fayl tərtib etmək lazımdır. Bu zaman faylı massiv tipi kimi deyil, (mümkün olarsa) siyahı kimi götürmək bir çox cəhətdən daha sərfəli olardı. Məsələni həll etmək üçün aşağıda tərtib edilmiş Çaylar_**Siyahı** proqramını, eyni

zamanda siyahılarla işləməyin əsas üsulları da praktik olaraq nümayiş etdirir.

```
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 – artıq element silinir}  
    elqabaq^.iz:=nil  
  end;  
procedure EkранаYazma;  
begin  
  writeln(`Fayl ayarlandı`);  
  writeln;  
  element:=elbir;  
  while element< > nil do  
    begin  
      with element^ do  
        writeln(ad:11,uz:8,al:12);  
        element:=element^.iz  
    end
```

```
        end;
end;
procedure YeniElement;
begin
    new(yeni);
    writeln;
    writeln(`Yeni elementi yazmaq:`);
    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);
    EkранаYazma;
    YeniElement;
    {Yeni element yazma}
    element:=elbir;
```

```
yeni^.iz:=element;  
elbir:=yeni;  
EkранаYazma;  
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 ünvanlaşdırma ilə silmək olar. Bu da əlbəttə 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 qırılır),
```

```
2201(keçid sədəfi qırılır ),
```

```
504000 (keçid sədəfi qırılır).
```

```
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 köçürməsinə 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. *Axin və növbə.* *Axin* -elementləri qatarın vaqonları kimi düzülmüş nizamlı çoxluq kimi təsəvvür edilə bilər. Qatar daxil olduğu stansiyadan geriyyə necə çıxırsa, həmin çoxluq da qəbul olunduğu yaddaşdan o cür ardıcılıqla istifadə olunur. *Növbə* – elementləri qatarın vaqonları kimi

düzülmüş nizamlı çoxluq kimi təsəvvür edilə bilər. Qatar daxil olduđu stansiyadan qabađa necə qədirsə, həmin çoxluq da qəbul olunduđu yaddaşdan o cür ardıcılıqla 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.

Çoxluqlar 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 iki 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 ardıcılığı 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 uyğun 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:

<pre>if <baza tipli element> in <çoxluq> then <1 -ci əmr> else <2 -ci əmr>;</pre>

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 tərtib 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!

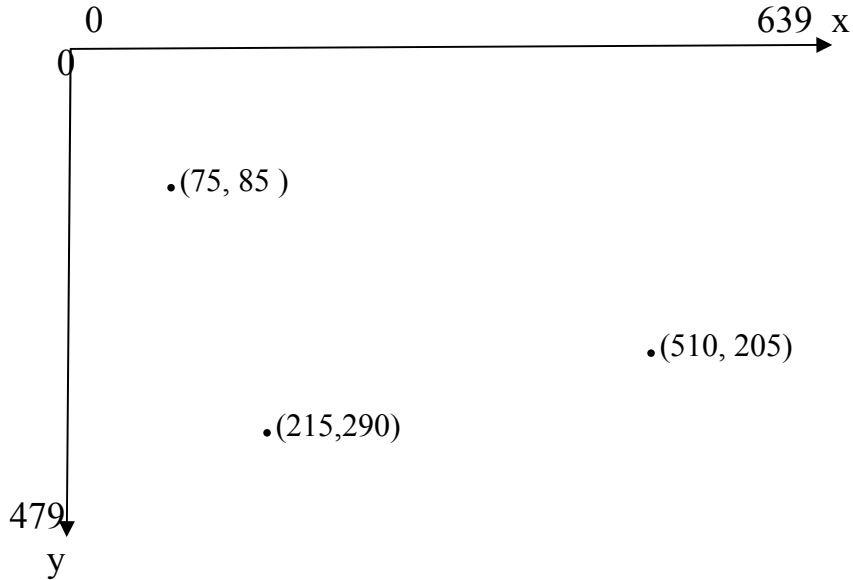
yazısı çıxmaldır.

Çalışma 29. 29 –cu məsələni, bir ambarda olmayıb digərlərində olan mallar üçün dəyişdirərək həll edin. Ambarlardakı malların çeşidini 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 xətti proqramlar tətbiq olunub **Graph.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

<bəlli proqramın təyinat bölümü>;

{Displenin xarakteristik 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 baş tutmadı');  
halt  
end;  
<proqramın davamı>  
end.
```

2. *Grafika prosedur və funksiyaları.* Əvvəlcə **Graph** modulunun prosedura və funksiyalarının təsnifatını verək.

Graph modulunun proseduraları

-initgraph (driver,mode <'drayverin ünvanı'>)-Qrafik rejim təsbit edir. Əgər drayver fayl kataloqda olmazsa onda onun ünvanı (apastroflar arasında) verilməlidir.

-detectgraph (<drayver>,<rejim>) –Displenin xarakteristik qiymətlərini verir.

-setcolor (<rəng>) –fona rəng verir.

-putpixel (x,y, <rəng>) –(x,y) nöqtəsinə rəng verir.

-line (x1,y1,x2,y2) –ucları (x1,y1) və (x2,y2) nöqtələri olan parça çəkir.

-lineto (x,y) –carri nöqtəni (x,y) –lə birləşdirir.

-linerel (dx,dy) –carri nöqtədən verilmiş artımlarla xətt çəkir.

-rectangle (x1,y1,x2,y2) –sol-yuxarı və sağ-aşağı qarşı küncləri (x1,y1) və (x2,y2) olan düzbucaqlı qurur.

-setviewport (x1,y1,x2,y2, true) –yeni qrafik xanə koordinatları verir. true –qrafikin xanədən kənar hissəsini silmə rejimi təyin edir.

-bar (x1,y1,x2,y2) –rəngli düzbucaqlı çəkir.

-bar3d (x1,y1,x2,y2, <həcm dərinliyi>,true) –parallelepiped çəkir.

-circle (x,y,R) –mərkəzi (x,y) və radiusu R olan çevrə çəkir.

-arc (x,y, <ilkın bucaq>, <son bucaq>, <radius>) qövs çəkir.

-pieslice (x,y, <ilkın bucaq>, <son bucaq>, <radius>) -rəngli sektor çəkir.

-ellipse (x,y, <ilkın bucaq>, <son bucaq>, <üfüqi ox>, <şaquli ox>) – ellips və ya ellips qövsü çəkir.

-setfillstyle (<rəngləmə rejimi>,<rəng>) - rəngləmə rejimi: 0 –fon rəngdə, 1 –bütöv, 2 –qalın üfüqi xətlərlə, 3 –maili xətlərlə, ..., 10 –nöqtələrlə, 11 –sıx nöqtələrlə.

-floodfill (x,y,<sərhəd rəngləri>) –(x,y) nöqtəsinə əhatə edən qapalı sahəni rəngləyir.

-closegraph –qrafik rejimi bağlayır.

-outtext (<mətn>) –verilmiş mətni cari mövqedən başlayır.

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

-settextstyle (<şrift>,<istişamət>,<ölçü>) –şrift: simvollara forma təyin edir, istişamət: 0 –üfiqi, 1 –şaquli, ölçü: 1,2,3.

Graph –modulunun funksiyaları.

-graphresult –qrafik rejim təsbit edildikdə ekrana 0, əks halda isə səhvin kodunu verir.

-getmaxx –ekranın üfiqi ölçüsünü göstərir.

-getmaxy –ekranın şaquli ölçüsünü göstərir.

-getcolor –cari rəngi göstərir.

-getcolor(x,y)-(x,y) nöqtəsinin rəngini verir.

-getx,gety –cari pikselin rəngini verir.

3. *Rənglər* –aşağıda sadalanan müsbət tam ədəd və ya rəngin ingiliscə adıyla verilə bilər:

Rəngin ingiliscə adı və ədədlə ifadəsi	Rəngin azərbaycan dilində adı
black=0	qara
blue =1	göy
green=2	yaşıl
cyan =3	mavi
red =4	qırmızı
magenta=5	bənövşəyi
brown =6	Şabalıdı(qəhvəyi)
lightgray =7	açıq-boz
darkgray =8	tünd-boz
lightblue =9	parlaq –göy
lightgreen =10	parlaq –yaşıl
lightcyan =11	parlaq –mavi
lightred =12	parlaq –qırmızı
lightmagenta =13	parlaq –bənövşəyi
yellow =14	sarı
white =15	ağ

Məsələ 30. Bir kvadrat daxilində, mərkəzi (320,240) nöqtəsi olan 10 konsentrik çevrə çəkməli.

```
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. İhtiyari seçdiyiniz üçbucađı, tərəflərinin orta nöqtələrini birləşdirməklə dörd yerə bölün və alınmış üçbucaqların daxilini müxtəlif rənglərlə boyayın.

Məsələ 31. Qrafik ekranın yuxarı sol küncünə qara fonda mavi kvadrat, onun daxilinə isə qırmızı çevrəli yaşıl dairə çəkərək ortadan, üfiqi 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 istifadə etməklə sevdiyiniz məktəbə emblem çəkin.

Məsələ 32. $h=0,1$ artımla, $[0,2\pi]$ parçasında $y = 2\cos(3x)+1$ funksiyasının qrafikini qurmalı. Nəticəni yoxlamaqla, amplitudanı $k*y$ ifadəsinin köməyi ilə tənzimləyərək daha münasib k əmsalı seçməli.

```
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 oxları}
  x:=a
  x1:=x0+trunc(k*x);
  {x,y –riyazi koordinatlardır}
  y:=f(x);
  y1:=y0-trunc(k*y);
  {x1,y1-qrafik koordinatlardır}
  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 –ci məsələni sinusoida qurmaq üçün dəyişməklə proqramı yenidən tərtib edin.

4. *Animasiya anlayışı* –piksəllərin rəng və yanib-sönməsinə tənzimləmə əsasında, təsvirlərin ekranda hərəkət görüntüsünü (imitasiyasını) almağa xidmət edir. Animasiya zamanı –hər işıqlanmadan qabaq obektin yerini seçilmiş səmtə dəyişdirmək lazımdır. Bir sözlə: *animasiya* –təsvirin ekranda hərəkət etməsidir.

Məs. Animasiya aşağıdakı kimi alqoritmlə verilə bilər:

1. Obektin təsvirini lazımı (x,y) koordinatlı nöqtədən çəkərək fasilə vermək
2. Fon rəngi verməklə obektə görünməz etmək
3. (x,y) –koordinatlarını dəyişdirmək
4. 1 –ci punkta qayıtmaq.

Məsələ 33. Mavi səmada günəşin üfqi istiqamətdə hərəkətini əks etdirməli.

```
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 seçdiyiniz qövs üzrə qaranlıq səmada ayın hərəkətini təsvir edən animasiya qurun.

§14. Obektlər

Turbo Paskal dilinin sonuncu versiyası obekt ünsür tipi də daxil edilməklə genişlənməmişdir.

Obekt tipi –özündə bir çox ünsürlər və onlarla işləməyə imkan verən alqoritmləri birləşdirir. Eyni obektdə birləşmiş hər ünsürə –alan, alqoritmə – gediş və birləşdirmənin özünə isə *-inkapsulyasiya* deyilir. Obektin icra mexanizmini –demək olar ki, yalnız gediş-alqoritmlər təşkil edir. Onları həm iç(obyektə aid olan) və həm də çöl(obyektə aid olmayan) ünsürlərə tətbiq etmək mümkündür. Bu zaman eyni adlı gediş müxtəlif obyektlərdə müxtəlif cür məzmun daşıya bilər və bu xüsusiyyət gedişlərin polimorfizmi adlanır. Obektlər də pöhrə-budaq prinsipi üzrə bənzərlik əlamətinə malik olub: obyekt tərkibində hər alanın bir çox pöhrəsi olsa belə o, yalnız bir budağa bağlı ola bilər.

1. *İnkapsulyasiya* –yəni obyektin təşkili:

```
type <obyektin adı> = object  
  <alanlar> : <alanların tipləri>;  
  <gedişlər>;  
end;
```

kimi tərtib olunur.

Məs. Alanları –pikselin x, y koordinatları və gedişləri –parlama, sönmə və yerdəyişmədən ibarət olan Piksel adlı obekt tipini aşağıdakı kimi tərtib etmək olar:

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

Burada:

- **Create** gedişi obyektı yaradıb alanlara bəlli ölçülər təyin edir və bu cür prosedura da **constructor** adlanır.
- **Draw** gedişində işlənən virtual kəlməsi aşağıda açıqlanacaqdır.
- **private** direktivi –alan və gedişlərlə programın digər proseduraları arasında əlaqə yaradır. Belə hallarda səhv etməmək üçün alanlara aid olan qiymətləri gedişlərin köməyiylə dəyişmək daha sərfəli olar. Məs. Pikselin yerini **Move** gedişilə dəyişmək olar.
- **public** direktivi isə digər proqramlarla əlaqə qurmaq üçün nəzərdə tutulmuşdur. Lakin **protected** ilə verilən komponentdən isə yalnız obyektin öz alanları(pöhrələri) istifadə edə bilər.

Gedişin icra bölümündə başlıdı:

```
procedure <obyektin adı>,<gedişin adı>(<formal parametrlər>);
```

kimi yazıla bilər.

Yuxarıda verilmiş obyektin gedişləri isə aşağıdakı kimi 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 tipinin dəyişəni –nüsxə adlanır və

var bölümündə:

<nüsxənin adı>.<gedişin adı>(<faktik parametrlərin siyahısı>);
--

formada verilir.

Məs. Piksel -obekt tipinin Pxel –adlı nüsxəsini:

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

..kimi və ya dinamik dəyişənlə isə:

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

formada ifadə etmək olur.

Burada:

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

-qoşa əmrini:

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

ilə əvəz etmək olar.

Yazı tipində olduğu kimi obekt tipinə də **With**(birləşdirmə) əmrini aşağıdakı qaydada tətbiq etmək mümkündür.

With Point do

begin

Create(100,100);

On;

Move(50,-10)

end;

2. *Əxzətmə* –obekt tipinin pöhrə-budaq prinsipinə məxsus bənzərlik əlaməti olub

type <pöhrənin adı> = **object**(<budaq tipinin adı>)

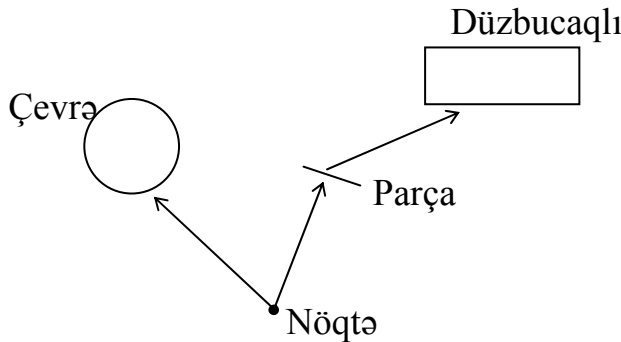
<əlavə olunmuş alanlar> : <alanların tipləri>;

<əlavə olunmuş və yenidən təyin olunmuş gedişlərin deklarasiyası>;

end;

formada tərtib olunur.

Əxzətmə prinsipini aşağıdakı sxem üzrə(Şəkil 1.3.) daha aydın şərh etmək mümkündür.



Şəkil 1.3.

Verilmiş sxemdə nöqtə –budaq, çevrə və parça isə onun pöhələri. Odur ki, əgər nöqtə: parlama, sönmə və hərəkət gedişlərinə malikdirsə, onda həmin gedişlər çevrə və parçaya da aid olub üstəlik də çevrə -radius (r), parça isə – məs. ikinci ucunun yerdəyişməsi (ShiftX,Shifty kimi) əlavə gedişlə zənginləşə bilər. Lakin göründüyü kimi, düzbucaqlı, -parçanın nöqtədən aldığı gedişləri ilə yanaşı (o diaqonalın uc nöqtələri əsasında qurulur) həm də ikinci ucun yerdəyişmə gedişinə də malik olmalıdır.

Çevrəni -Circle və nöqtəni –Point adlandırmaqla Circle tipi **draw** gedişi və **constructor** ilə:

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** –budağın gedişlərini pöhrəyə ötürən Create konstruktunu çađırmaq üçündür. Hərçəndi bu çađırışı Point.Create(a.b) kimi də vermək olardı.

Parçanı Line adlandırmaqla işə müvafiq obekt tipini:

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

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

Tapşırıq. Düzbucaqlını Rect adlandırmaqla müvafiq obyekt tipini təyin edin.

3. *Polimorfizm* –eyni adların müxtəlif obyekt gedişlərində işləmə bilməsidir. Məs. bütün qurmaları həyata keçirən Create və draw gedişləri eyni 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 yanır}  
  end;
```

prosedurası əsasında aramsız olaraq çağıracaqdır.

Bu vəziyyəti aradan qaldırmaq üçün **virtual** xidməti sözündən

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

formada istifadə olunur. İndi artıq lazımı Draw gedişinin ünvanı yalnız iş ərəfəsində müəyyən ediləcəkdir. Məs. Circle tipinin obyekt 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, burada virtual kimi verilmiş gediş bütün pöhrələrdə də məhz bu cür göstərilməlidir. Lakin obyekt tipində bir dənə də olsa virtual gediş 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 biri çevrə, parça və düzbucaqlı fiqurları üçün yazılmaqla 9 dəfə təkrarlanardı. Bu cür izafi işin böyük həcmli proqramlarda hansı həddə varsa 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 Turbo Pascal 7.0 proqram mətnləri hazırlama və işə buraxmada geniş tətbiq

olunur. Bu mühit üçün (Windowsda qrafika işindən başqa) bütün növ proqramlama prinsipləri eynidir.

Turbo Pascal 7.0 proqram mühitinin əsas faylları.

turbo.exe	əsas işlək fayl, həcmi –400Kbayt
turbo.tpl	kıtabxana –48Kbayt (konfiqurasiyadan da asılıdır)
turbo.tph	köməkçi –730Kbayt
graph.tpu	qrafikayla iş üçün modul –33Kbayt

Mühitə daxil olmaq üçün turbo.exe əmrini yerinə yetirilir və ekranın yuxarı hissəsində baş menyü, aşağıda isə bəzi funksional düymələrin şərhini çıxır. Bu zaman baş menyuya girmək üçün F10 sədəfini qırmaqla aşağıdakı bəndlər abonentin sərəncamına verilir:

File	faylla iş üçün
Edit	faylda düzəlişlər üçün
Search	verilmiş fraqment mətnində arama və ya dəyişmə üçün
Run	proqramın icrası üçün
Compile	proqramın kompilyasiyası və exe-fayl yaratmaq üçün
Debug	proqramı kökləmək (nizamlamaq) üçün
Options	mühitin konfiqurasiyası üçün
Window	pəncərələrlə iş və onların konfiqurasiyası
Help	köməkçini çağırmaq üçün

Menyudan lazımı bəndi, əlbəttə kursorla seçib keçid sədəfini qırmaq gərəkdir. Hərçəndi bu işi Alt və baş hərf sədəflərini eyni vaxtda qırmaqla da etmək mümkündür.

Bir iş seansının əsas mərhələlərinə nəzər salaq:

- 1) Baş menyü açılır, buradan **File** bəndi seçilir. Bu zaman əlavə bir menyü də alınır ki, buradan da New əmrini seçməklə proqram mətnini yazmaq üçün NONAME00.PAS adlı fayl açılmış olur. Beləliklə də F10⇒File⇒New addımlarını atmaqla fayl açaraq proqram mətni yazılır.
- 2) Mətnə düzəlişlər –bəziləri aşağıda şərh olunmuş- ənənəvi düzənləmə üsullarıyla aparıla bilər:

Shift+oxlar	Əməliyyat aparılacaq fraqmenti (qara rənglə) seçmə
Ctrl+Insert	Bloku buferə də köçürmək
Shift+del	Bloku buferə gətirmək
Shift+Insert	Fraqmenti kursurun olduğu yerə köçürmək
Ctrl+Del	Seçilmiş bloku silmək
Ctrl+Y	Kursor olan sətiri silmək
Ctrl+Q,Y	Sətrdə kursordan sonrakı hissəni silmək
Ctrl+n	Sətir yerləşdirmək

Ctrl+PgUp	Mətnin başlanğıcına keçmək
Ctrl+PgDn	Mətnin sonuna keçmək

- 3) Əgər aşkar səhvlər yoxdursa, proqramı kompilyasiya edib F10⇒Run⇒Run əməlləri və ya Ctrl+F9 sətəflərilə işə buraxmaq olar.
- 4) Əgər sintaksis səhvlər varsa, onda cursor səhvin olduğu sətirdə və ya düz həmin mövqedə duracaq, yuxarı sətirdə isə səhvin məzmunu barədə məlumat veriləcəkdir. Bu zaman mətn artıq düzənləmə rejimində gözləyir və çox vaxt itirmədən səhvi düzəltmək mümkün olur. Bu minvalla hər dəfə səhvlər düzəldilərək 3) –cü bənd təkrarlanır və s.
- 5) Əgər sintaksis səhvlər yoxdursa onda proqram işləməli və cavablar isə nəticə xanəsinə verilməlidir ki, bunun üçün də Alt+F5 sətəfləri qırılmalı və ya Debug bəndi gedişlərindən istifadə edilməlidir. Nəticələri gözdən keçirərək istənilən sətəfi qırmaqla yenidən düzəliş rejiminə qayıtmaq olar.
- 6) Proqramın exe-faylını F10⇒compile⇒Destination⇒disk ardıcıl əməliyyatlarından sonra almaq olar. Daha sonra Alt+F9 sətəflərini qırılmaqla diskın cari kataloqunda, heç bir proqram mühiti olmadan da işləyə biləcək exe-fayl yazılmış olur.
- 7) Proqram mətnini də genişlənməmiş .pas faylında saxlamaq: a) adını dəyişməklə -F10⇒file⇒Sava As, əks halda isə F10⇒file⇒Sava (və ya sadəcə F2 sətəfini qırmaq).
- 8) Seansı bitirib mühitdən çıxmaq üçün F10⇒file⇒Exit və ya sadəcə Alt+x sətəfləri qırılmalıdır.
- 9) Diskdə yerləşən proqramla işləmək üçün F10⇒file⇒Open və ya F3 sətəfi qırılır. Tab sətəfini qırmaqla, aşağı xanəyə keçib fayl menyusundan lazımı faylı tapmaq və keçid sətəfini qırmaqla onu düzənləmə rejiminə keçirmək mümkündür.
- 10) Açıq xanəli proqramlar çox olduqda, birindən digərinə keçmək üçün F6, genişlətmə və ya daraltmanı isə F5 –in köməyiylə icra etmək olar. xanəni ekranda münasib formada yerləşdirmək üçün Windowdan istifadə edilir. Fəal xanəni bağlamaq üçün isə ya Alt+F3 sətəflərini qırmaq ya da ki, mausun dilini yuxarı sol küncdəki nişan üzərinə qırmaq gərəkdir.
- 11) Turbo Paskal barədə məlumatlar almaq üçün isə F1 sətəflərini qıraraq ekrana verilən məlumatları oxumaq olar.

Borland Pascal for Windows mühitinin də yuxarıda sadalanan bəndlərdən elə bir ciddi fərqi yoxdur. Bu halda sadəcə proqramın əvvəlində **uses Crt** əvəzinə **uses WinCrt** yazmaq və bir də **Sava As, Open** kimi fayl sistemi əməlləri ilə işləmə vərdisi tələb olunur.

2. *Tipik səhv kodları* –Tupbo Pascal 7.0 mühitində səhvlər kompilyasiya səhvləri və icra səhvləri kimi iki qrupa bölünür.

Bəzi xarakterik kompilyasiya səhvlərinin şərhı.

Kod	Məlumat	Məzmun
3	Unknown identifier	şərh olunmamış identifikator
4	Duplicate identifier	təkrar identifikator
5	Syntax error	sintaksis səhv
14	Invalid file name	Faylın adı və ya ünvanı düz deyil
26	Type mismatch	Tiplərin uyşmazlıdı
42	Error in expression	İfadədə səhv
62	Division by zero	sıfıra bölmə
64	Cannot Read or Write variables of this type	bu tipin dəyişənlərini yazmaq və hesablamaq mümkün deyil
85	<<» expected	<<» simvolu gözlənilir(və ya çatmır)
91	<<:=> expected	<<:=> simvolu gözlənilir
94	<<.> expected	<<.> simvolu gözlənilir
95	<<..> expected	<<..> simvolu gözlənilir

İcra səhvləri:

- DOS səviyyəli səhvlər –kod 1..99
- Giriş-çixış səhvləri –kod 100..149
- Kritik səhvlər –kod150..199
- Fatal səhvlər –kod 200..255

Bəzi xarakterik icra səhvlərinin şərhı

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

3. *Kompilyatorun direktivləri* –proqramlara qeyri standart kompilyasiya rejimləri vermək üçün nəzərdə tutulmuşdur ki, onları da: –rejimdəyişdirici direktivlər və parametrlı direktivlər kimi iki qrupa ayırmaq olar. Birincilər {<simvol> <işarə>} kimi ümumi formaya malikolub Option ⇒ Compiler ⇒ <kompilyatorun əmri> ifadəsilə kompilyasiya rejimlərini və ya «+», «-» işarələrini verə ya da lədv edə bilər. Onu da deyək ki, normal vəziyyətdə kompilyator əsas direktivləri onsuz da cəlb edir və xüsusi zərurət olmadıqda rejimdəyişməyə ümumiyyətlə çox ehtiyac qalmır.

Rejimdəyişdirici direktivlərin təsnifatı:

{A+} –ünsürlərin baytın deyil sözün sərhəddində bərabərləşdirir. Çox yaddaş alırsa da proqramın sürətlə işləyir. Elə-belə də işləyir. Kompilyatorun əmri –Word AllignData.

{B-} –məntiqi ifadə hesablamasının müfəssəl sxemi.Nəticə aydın olan kimi hesablama kəsilir. Elə-belədə {B+} –Complete Boolean Evaluation direktivi işləyir.

{D+} –Debug Internation –proqram mətnində səhv verən əmrlər barədə məlumat verir. Elə-belə də işləyir.

{E+} –Emulation – proqram yoluyla hwmprocessoru emulyasiya edir. Elə-belə də işləyir.

{F+} –Force Far Calls – Prosedura və funksiyaların “uzaqlaşdırılmış” çadırı tipidir. Elə-belədə işləmir.

{G+} –286 Instructions –yalnız 80286 tip processorla işləyən maşınlar üçündür. Elə-belədə işləmir.

{I+} –I/O Checking –Giriş-çıxış səhvləri olan kimi proqram dayanır. Elə-belə də işləyir.

{L+} –Local Sumbols –lokal dəyişənlər barədə məlumat verir. {D+} işləyən zaman o da elə-belə işləyir.

{N+} –8087/80287 sürüşgən vergüllü hesablamada həmprocessor kimi işlədilir. Həmprocessoru olmayan maşınlarda o, {N-} ilə ləöv olunmalıdır.

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

{P+} –Open Parameters açıq tip massivləri parametr kimi işlətməyə imkan verir. Elə-belədə işləmir.

{Q+} –Overflow Checking – Riyazi proseslərdə tıxanma hallarını bildirir.Elə-belədə işləmir.

{R+} –Range-Checking –Ünsürlərin verilmiş diapazondan kənara çıxma hallarını bildirir. Elə-belədə işləmir.

{S+} –Stack Checking –axının tıxanmasını yoxlayır

{V+} –Strict Var String –formal və faktiki parametrlərin eyni uzunluqda olmasını yoxlayır.

{X+} –Exstended Syntax –proseduranı funksiya kimi təqdim etməyə imkan yaradır. Elə-belədə işləmir.

Parametrlı tiplərdən:

{I <faylın adı>} –Include Directories –kompilyasiya zamanı proqram mətninə əlavə mətn qoşmağa imkan verir. Məs. {I *.dfm} –dfm əlavə olunmuş bütün faylları (Delphidə işlənir) birləşdirir.

{L <obyekt faylının adı>} –Object Directories - kompilyasiya zamanı proqrama bəzi obekt kodları daxil etməyə imkan verir.

{\$M <1024-65520 aralığında axın ölçüsü>, <0-655360 aralığında dinamik sahə ölçüsü>} –Memore Sizes –Yadda;I proqrama münasib şəkildə bölür.

Burada A, D, E, L, N, O, P, Q, X və M -qlobal direktivlər olub tələb olunduqda məhz proqramın adından sonraya yerləşməli, digər direktivlər isə lokal səciyyəli olub ara-bərayə də salına bilər.