AZƏRBAYCAN RESPUBLİKASI KƏND TƏSƏRRÜFATI NAZİRLİYİ
AZƏRBAYCAN RESPUBLİKASI TƏHSİL NAZİRLİYİ
AZƏRBAYCAN RESPUBLİKASI TƏHSİL PROBLƏRI INSTITUTU
AZƏRBAYCAN DÖVLƏT AQRAR UNIVERSITETI

Paskal alqoritmik dili

Azərbaycan Respublikasi Təhsil Nazirliyi Elmi-metodik şurasının "Informatika və kompüter texnologiyası" bölməsinin 08.12.2012-cu il tarixli iclasının qərarı ilə təstiq edilmişdir (protokol № 2025)

G Ə N C Ə – *20*12

Müəlliflər:
Əli Hüseyn oğlu Quliyev
Rəxşəndə Yunis qızı Hüseynova
Ruhiyyə Mustafa qızı Bağıyeva
Yeğanə Əziz qızı Abbasova

Paskal alqoritmik dili. Dərs vəsaiti, səh.102, 2012.

Rəy verənlər:

AzTU-nın İnformatika və telekommunikasiya kafedrasının müdiri, prof. Verdiyev S.Q.
AzTU-nın İnformatika və telekommunikasiya kafedrasının dosenti, t.e.n. Muradov P.C.

ADAU-nun İnformasiya texnologiyaları və sistemləri kafedrasının dosenti, f.r.e.n Şirinov R.H.
ADAU-nun Aqrar fizika və ali riyaziyyat kafedrasının dosenti, f.r.e.n Baratzadə R.Z.

Vəsaitdə alqoritm anlayışı, tipik alqoritmik strukturlar, Paskal dilinin əsas anlayışları, standart funksiyaları, proqramlaşma mühıti, operatorları və müvafiq çalışmaların həlli verililir.
Dərs vəsaitindən ali məktəb və kolleclərin tələbələri, ümumtəhsil məktəblərinin şagirdləri və Paskalda proqramlaşmanı müstəqil öyrənənlər də istifadə edə bilərlər.

Nəşriyyat şöbəsinin baş redaktoru F.Ə.Namazov

GİRİŞ

Yaşadığımız informasiyalı cəmiyyətdə müasir kompüterlərlə təchiz edilmiş təhsil müəssisələrinin əsas vəzifələrindən biri də alqoritmik düşüncə tərzinə malik, informasiya və kommunikasiya texnologiyalarından istifadə etməyi bacaran vətəndaşlar hazırlamaqdan ibarətdir. Bunun üçün təlim prosesində öyrənənlərin qarşısına çıxan çətinlikləri aradan qaldırılmasına xidmət edən dərs vəsaitləri ilə təmin edilməsi zəruridir.

Hesablama maşınlarının ən əsas fərqləndirici xüsusiyyətlərindən biri də onun proqramla idarə olunmasıdır. Yəni istər sadə, istərsə – də mürəkkəb məsələni maşının həll etməsi üçün istifadəçi maşın üçün proqram tərtib etməlidir.

Bu dərs vəsaitində hal-hazırda müasir kompüterdə geniş tətbiq edilən alqoritmlərin ən vacib xassələrini aydın və asan başa düşülən şəkildə əks etdirən Turbo-Pascal dilində proqramlaşdırmanın əsasları şərh edilir. Pascal dili sadəliyinə, məntiqiliyinə və effektlivliyinə görə çox geniş yayılmış dillərdəndir. O geniş, səciyyələndirilmiş imkanlara malikdir.

Kitabda alqoritmlər nəzəriyyəsi**,** Turbo-Pascal dilinin elementləri, dilin əlifbası, işçi sözləri, verilənlərin tipi, proqramın quruluşu, proqramlaşma mühiti və operatorları, böyük həcmli informasiyaların işlənməsi ilə bağlı fayllar və dinamik strukturlu verilənlərlə işləmə, proqram tərtib etməyin əsas qaydaları izah edilir. Bununla yanaşı, dilin müasir versiyaları üçün xarakterik olan mətn və qrafik rejimlərdə ekranı, klaviaturanı, çap və səs qurğularının idarə edilməsi məsələlərinə toxunulur.

Kitabda nəzəri məsələlərin şərhi uyğun çalışmalarla verilir ki, bu da alqoritmik dilin tələbələr tərəfindən mənimsənilməsinə kömək edir. Kitabda Turbo-Pascal 7.0 versiyasının işçi pəncərəsi və onun menyu əmrləri ilə işləmə qaydaları da istifadəçilərə köməklik xarakteri daşıyır.

Kitab haqqında öz təklif və rəylərini bildirən oxuculara əvvəlcədən öz təşəkkürümüzü bildiririk. Müəlliflər

1. ALQORİTMLƏR NƏZƏRİYYƏSİ, ALQORİTM ANLAYIŞI, ONUN XASSƏLƏRİ, TƏSVİR ÜSULLARI

1.1. Məsələnin kompüterdə həllə hazırlığı

Həll yolu (alqoritmi) məlum olan istənilən məsələni kompüterdə həll etmək mümkündür. Xarakterinə görə məsələləri aşağıdakı siniflərə bölmək olar:

- elmi-texniki (və ya riyazi mühəndis);
- iqtisadiyyat –statistika;
- informasiya məntiqi;
- idarəetmə və modelləşdirmə.

Kompüterdə məsələlərin həlli aşağıdakı mərhələlər ardıcıllığı ilə aparılır:

- məsələnin qoyuluşu;
- həll alqoritminin yaradılması;
- verilənlərin strukturlarının təyini;
- proqramlaşdırma dilinin seçilməsi və ilkin proqramın tərtibi;
- proqramın kompüter dilinə çevrilməsi və sazlanması;
- işçi proqramın icrası, nəticələrin alınması və təhlili.

Müəyyən tip məsələlərin həllində bu mərhələlərdən bəziləri tələb olunmaya bilər. Məsələn sistem proqram təminatının yaradılmasında məsələnin riyazi təsviri tələb olunmur. Göstərilən mərhələlər bir-biri ilə əlaqədardır. Məsələn, nəticələrin təhlili proqramda, alqoritmdə və hətta məsələnin qoyuluşunda müəyyən dəyişikliklər etməyə səbəb ola bilər. Bu cür dəyişikliklərin sayını azaltmaq üçün hər mərhələdə sonrakı mərhələlərin tələblərinin mümkün qədər nəzərə alınması lazımdır. Bəzi hallarda müxtəlif mərhələlər arasındakı əlaqələr o qədər sıx olur ki, (məsələn, məsələnin qoyuluşu ilə hesablama üsulunun seçilməsi, alqoritm və proqramın yaradılması mərhələləri), onları bir-birindən ayırmaq çətin olur.

Məsələnin qoyuluşu. Məsələnin müvəffəqiyyətli həlli onun düzgün qoyuluşundan çox asılıdır. Məsələnin qoyuluşunda sadə hallarda aşağıdakılar nəzərdə tutulur: İlkin verilənlərin siyahısı, tipi, dəqiqliyi və ölçüləri; dəyişənlərin dəyişmə hədləri, başlanğıc və sərhəd

şərtləri; nəticələrin siyahısı, tipi, dəqiqliyi, ölçüləri; məsələnin həllini təmin edən hesabat düsturları və tənlikləri.

Bu mərhələdə müəyyən sinif məsələlər üçün onların riyazi formalaşdırılması da aparılır, yəni tədqiq edilən prosesin baxılan halda əlverişli olan formal dildə, formatda riyazi modeli qurulur, bəzi riyazi və mühəndis məsələləri üçün (məsələn, diferensial tənliklərin həlli, müəyyən inteqralların hesablanması və s.) ədədi hesablama üsulu seçilir və ya yaradılır. Burada söhbət tənliklərin, riyazi analiz işarələrinin (inteqrallama, diferensiallama, operator işarələri və s.) hesabi və məntiqi əməllər ardıcıllığına çevrilməsindən gedir.

Həll alqoritminin yaradılması. Bu mərhələdə seçilən həll metoduna uyğun məsələnin həll alqoritmi tərtib olunur. Məsələnin həlli ayrı-ayrı müstəqil bloklara bölünür və həmin blokların yerinə yetirilmə ardıcıllığı təyin edilir. Nəticədə alqoritmin blok sxemi qurulur.

Verilənlərin strukturunun təyini. Bu mərhələdə alqoritmdə iştirak edən verilənlərin tipinə, formasına, mümkün qiymətlərinə və aparılan əməliyyatlara görə onların strukturları seçilir. Yəni verilənlərin tam, həqiqi, simvol və s. tipli olması, massiv, yazı, stek, növbə, siyahı, fayl və s. strukturlarla təşkili müəyyənləşdirilir.

Proqramlaşdırma dilinin seçilməsi və ilkin proqramın tərtibi. Hazırda proqramlaşdırma üçün müxtəlif dillər mövcuddur. Həll olunan məsələnin xarakterinə, tətbiq olunan kompüter üçün mövcud olan translyatorlara, proqramçının hazırlıq səviyyəsinə görə proqramlaşdırma dili seçilir. Sonra isə məsələnin həll alqoritmi əsasında seçilən dildə proqram tərtib edilir. Ona ilkin proqram deyilir.

İlkin proqramın kompüter dilinə çevrilməsi və sazlanması. Bu mərhələdə proqramlaşdırma dilində yazılmış ilkin proqram kompüter dilinə çevrilir. Bu iş translyator adlanan proqram vasitəsilə yerinə yetirilir. Bu zaman ilkin proqramda buraxılmış morfoloji və sintaksis səhvlər aşkar edilib, proqramçıya çatdırılır, səhvlər aradan qaldırıldıqdan sonra tərcümə prosesi davam etdirilir və kompüter dilində proqram alınır. Bu proqrama mütləq və ya işçi proqram deyilir. Bu proqramın düzgün işləməsini yoxlamaq məqsədilə yoxlama misalında o, sınaqdan çıxarılır. Proqramdakı məntiqi səhvlərin aşkarlanıb, aradan qaldırılması prosesinə proqramın sazlanması deyilir.

İşçi proqramın icrası, nəticələrin alınması və təhlili. Proqram sazlandıqdan sonra o, müxtəlif ilkin verilənlər dəsti üçün bir neçə dəfə icra olunur, alınmış nəticə mütəxəssis və məsələni qoyan istifadəçi tərəfindən təhlil olunur. əgər təhlil prosesində nəticələr istifadəçini tan təmin etmirsə, o, yeni tələblər qoya bilər və ya əvvəlki, tələblərdə dəyişiklik edə bilər. Bu halda yeni tələblərin xarakterindən asılı olaraq məsələnin qoyuluşunda, alqoritmdə və ya proqramda müəyyən dəyişiklik edilir. Uzun müddət istifadə olunan proqram kompüterin xarici yaddaşında (diskdə) hazır proqram kimi saxlanılır. Proqrama istifadəçi üçün təlimat da əlavə olunur.

1.2. Alqoritm, onun xassələri

Alqoritm riyaziyyatın mühüm anlayışlarından biri olub, hələ kompüter yaranmamışdan əvvəl mövcud idi. Alqoritm – latınca qayda-qanun deməkdir. Alqoritm sözü IX əsrin məşhur özbək riyaziyyatçısı Məhəmməd İbn Musa əl-Xarəzminin (yəni Xarəzmli Musa oğlu Məhəmməd) adının latın hərflərilə olan "alqoritmi" yazılışıyla bağlıdır. Əl-Xarəzminin yazdığı traktatın XII əsrdə latın dilinə tərcümə olunması sayəsində avropalılar mövqeli say sistemi ilə tanış olmuş, onluq say sistemini və onun hesab qaydalarını alqoritm adlandırmışlar. Çoxrəqəmli onluq ədədlər üzərində hesab əməllərinin aparılması qaydaları (alqoritmləri) ilk dəfə IX əsrdə özbək riyaziyyatçısı Əl-Xarəzmin tərəfindən verilmişdir.

Alqoritm-verilmiş məsələnin həlli üçün lazım olan əməliyyatları müəyyən edən və onların hansı ardıcıllıqla yerinə yetirilməsini göstərən formal yazılışdır.

Ümumi şəkildə desək, alqoritm məsələnin həll yoludur, yəni məsələnin həllini təmin edən formal qaydalar sistemidir. Məsələnin kompüterdə həlli baxımından alqoritm axtarılan cavabların alınması üçün məsələnin verilənləri üzərində icra olunan hesabi və məntiqi əməllər (mərhələlər) ardıcıllığıdır. Bu mərhələlərdə uyğun olaraq hesab və müqayisə əməlləri yerinə yetirilir. Müqayisənin nəticəsindən asılı olaraq bu və ya digər mərhələnin icrasına keçilir.

Alqoritm həll olunan məsələnin xarakteri ilə bağlı olduğu üçün onun yaradılmasında ümumi qaydalar yoxdur. Lakin hər bir alqoritm

tərtib edilərkən onun müəyyən tələblərə cavab verməsi nəzərə alınmalıdır. Bu tələblərə alqoritmin xassələri deyilir. Alqoritmin əsas 4 aşağıdakı xassələri vardır:

1.Müəyyənlik. Alqoritmin tərtibi, məsələnin həllini ardıcıl yerinə yetirilən mərhələlərə bölmək deməkdir. Bu zaman əvvəlki mərhələlərin nəticələri sonrakı mərhələdə istifadə oluna bilər. Əsas tələb ondan ibarətdir ki, hər bir mərhələnin məzmunu və mərhələlərin yerinə yetirilmə ardıcıllığı müəyyən olmalıdır. Bu alqoritmin müəyyənlik xassəsini təşkil edir.

2. **Diskretlik.** Hər bir alqoritm ayrı-ayrılıqda yerinə yetirilə bilən addımlardan ibarət olmalıdır. Alqoritmdə hesablama prosesi əməllər ardıcıllığına bölünməlidir.

3. **Nəticəvilik.** Alqoritmdəki mərhələlərin və onları təşkil edən əməliyyatların sayı sonlu olmalıdır ki, onların yerinə yetirilməsi axtarılan nəticəyə gətirib çıxara bilsin.

4.Kütləvilik. Bu xassədə iki tələb nəzərdə tutulur:

a). müəyyən məsələnin həlli üçün qurulmuş alqoritm həmin tipdən olan bütün məsələlərin həlli üçün yararlı olmalıdır;

b). alqoritm elə təsvir olunmalıdır ki, ondan hamı istifadə edə bilsin.

Alqoritmdəki hesab əməlləri arasındakı məntiqi əlaqələr kompüterin qəbul edə biləcəyi şəkildə verilməlidir. Həmin əlaqələr çox vaxt bu və ya digər hesablama addımlarının seçilməsini təyin edən müəyyən şərtlərin yoxlanması şəklində ifadə olunur. Məntiqi şərtlər içərisində aşağıdakılar xüsusi yer tutur, çünki onların yaranması hesablama prosesinin normal gedişinə imkan vermir:

- hesablamada mütləq qiymətcə kompüterdə təsvir oluna biləcək maksimal ədəddən böyük ədədin alınması;
- sıfırın və ya mənfi ədədin loqarifmalarının hesablanması;
- mənasız hesablamaların aparılmasına cəhd göstərilməsi (məsələn, $|x| \succ 1$ olduqda, arcsin(x) və ya arccos(x)–in hesablanması).

1.3. Alqoritmin təsvir üsulları

Alqoritmin əyani, yığcam və standart vasitələrlə təsviri onun kütləviliyini təmin edən əsas amildir. Alqoritmin təsviri üçün istifadə olunan əsas üsullar aşağıdakılardır:

- sözlə (nəqli) təsvir;
- sxemlə təsvir;
- alqoritmik dillə təsvir.

Sözlə təsvir alqoritmin kütləvilik xassəsini təmin etmədiyindən, o, icrası insan tərəfindən aparılan və nisbətən sadə alqoritmlərin təsvirində istifadə oluna bilər.

Alqoritmin ən yığcam təsvir vasitəsi alqoritmik dildir. Bu üsul alqoritmin icrasının kompüter vasitəsilə yerinə yetirildiyi halda daha əlverişlidir. Çünki alqoritmik dildə təsvir olunan alqoritm həm də məsələnin ilkin proqramıdır. Lakin bu üsul mürəkkəb alqoritmlərin oxunub başa düşülməsini xeyli çətinləşdirir.

Alqoritmin təsvirində ən geniş tətbiq edilən sxem üsuludur. Bu üsulda alqoritm, hər biri müəyyən funksiyanı yerinə yetirən bloklar ardıcıllığı şəklində təsvir olunur. Adətən bir blok alqoritmin bir mərhələsinə uyğun olur. Lakin bir blokda bir neçə eyni tipli mərhələ və ya əksinə, bir mərhələ bir neçə blokda təsvir oluna bilər. Bloklar həndəsi fiqur şəklində ifadə olunur və bir-biri ilə şaquli, yaxud üfüqi xətlərlə birləşdirilir. Əgər xətlərin uclarında istiqaməti göstərən ox işarəsi yoxdursa, onda keçidin şaquli istiqamətdə yuxarıdan aşağıya, üfüqi istiqamətdə isə soldan sağa verildiyi qəbul olunmuşdur. lazım gəldikdə bloklar nömrələnir.

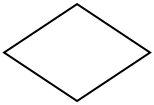
Alqoritmin blok sxemini bütöv şəkildə qurmaq məsləhətdir, lakin lazım gəldikdə, blokları birləşdirən xətləri qırmaq mümkündür.

Alqoritmin blok sxemlə təsvirində hər bir mərhələnin məzmunu, mərhələlərin icra ardıcıllığı, təkrarlanan hissələr (dövrlər) aydın görünür. Mürəkkəb və böyük həcmli məsələlərin həlli zamanı blok sxemin tərtibi çox zəhmət və vaxt tələb edir. Belə hallarda alqoritmin hər bir bloku özündə bir neçə mərhələni əks etdirən ümumiləşdirilmiş blok şəklində təsvir olunur. Alqoritmin sonrakı dəqiqləşdirilməsi isə proqramlaşdırma mərhələsində yerinə yetirilir.

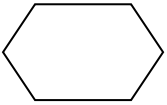
Blokların qrafiki şəkildə ifadə olunması üçün Proqram Sənədlərinin Vahid Sistemi (PSVS) çərçivəsində standart qəbul olunmuşdur və bu aşağıdakı kimidir:

Hesablama bloku

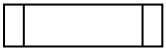
İlkin verilənlərin daxil edilməsi bloku



Şərti (məntiqi blok) bloku

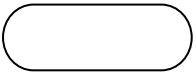


Dövr bloku

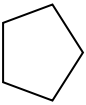


Alt alqoritm və ya alt proqram bloku

Cavabların kağıza çap edilməsi bloku



Başlanğıc və ya sonu göstərən blok



Monitora və ya displeyə çıxış bloku;

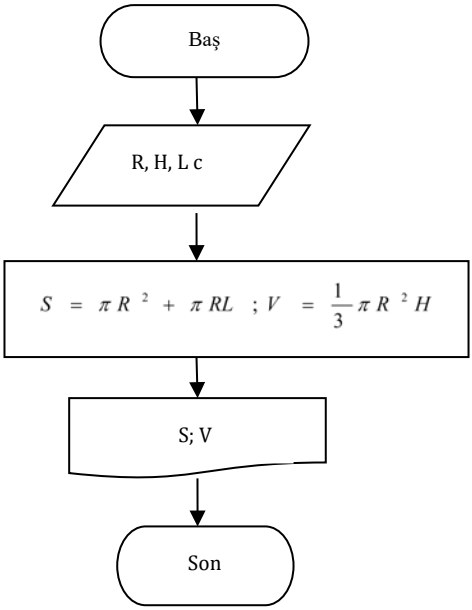
1.4. Tipik alqoritmik strukturlar

İstənilən hesablama prosesi aşağıdakı tipik (elementar) alqoritmik strukturların kombinasiyasından təşkil olunur: xətti, budaqlanan, dövrü (təkrarlanan)

Xətti alqoritmik struktur.

Xətti alqoritmik struktur iki və daha çox prosesin ardıcıllığından ibarətdir. Onun tərkibində seçmə bloku olmur.

Misal: Oturacağının radiusu R, hündürlüyü H, doğuranı L olan konusun tam səthinin sahəsinin və həcminin hesablanması üçün alqoritmin tərtibi.



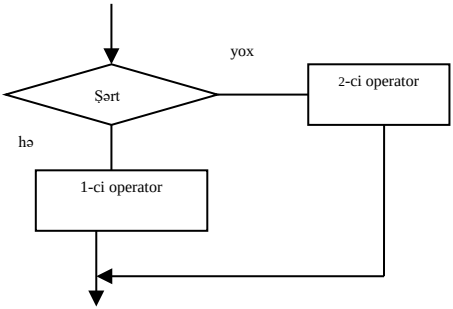
Budaqlanan alqoritmik struktur.

Tərkibində məntiqi blok olan hesablama prosesini təsvir edir. Hər bir budaqlanma nöqtəsi uyğun məntiqi blokla təyin olunur. Bu blokda müəyyən kəmiyyətlərin (ilkin verilənlərin, aralıq nəticələrin və s.) bu və ya digər şərti ödəyib-ödəməməsi yoxlanılır və nəticədən asılı olaraq, bu və ya digər hesablama istiqaməti seçilir. İki blokdan ibarət olan prosesə sadə, ikidən çox budaqdan ibarət olan prosesə isə mürəkkəb budaqlanan struktur deyilir. Mürəkkəb budaqlanan struktur sadə strukturlarla ifadə oluna bilər.

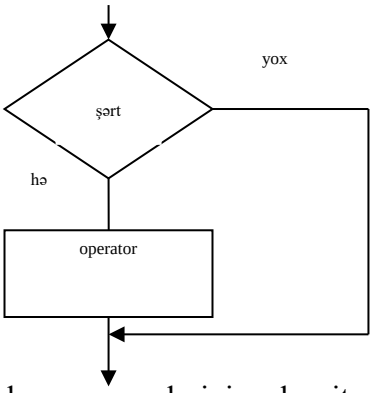
Blok sxemdə hər hansı şərtdən asılı olaraq bütün hesablama istiqamətləri göstərilməlidir. Lakin proqramın icrası zamanı bu istiqamətlərdən yalnız birinə görə hesablama aparılır. Seçilən bir budağa görə hesablama prosesi sonlu nəticəyə gətirilib çıxarılmalıdır.

Alqoritmik dillərdə budaqlanan struktur iki cür olur:

- tam formatlı budaqlanma;



- natamam formatlı budaqlanma



Budaqlanan hesablama proseslərinin alqoritm və proqramlarını tərtib edərkən, əgər ayrı-ayrı budaqlarda təkrarlanan hesablama əməliyyatları varsa, onlar budaqlanmadan əvvəl yerinə yetirilməlidir.

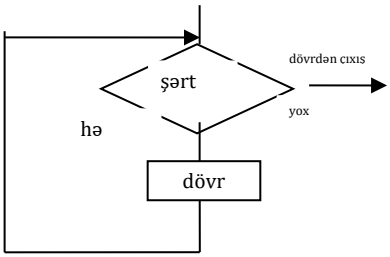
Dövrü alqoritmik strukturlar.

Təcrübədə çox rast gəlinən hesablama proseslərində məsələnin (və ya onun bir hissəsinin) həlli eyni hesablama düsturları ilə dəyişənlərin müxtəlif qiymətləri üçün bir neçə dəfə təkrarən hesablamaların aparılmasını tələb edir. hesablama prosesinin təkrarən yerinə yetirilə hissəsinə dövr deyilir.

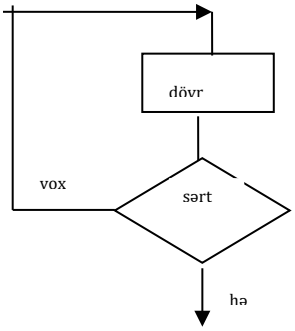
Dövrü strukturlar sadə və mürəkkəb ola bilər. Sadə struktur bir, mürəkkəb struktur isə biri digərinin içərisinə daxil olan iki və daha çox dövrdən ibarət olur.

Dövrü alqoritmik strukturların tərtibi zamanı dövrün girişində hesablama prosesinin tələb etdiyi hazırlıq əməliyyatlarının aparılması nəzərə alınmalıdır (məs., bir neçə ədədin cəminin və ya hasilinin hesablanmasında başlanğıc qiymətlərin mənsub edilməsi).

Mürəkkəb dövrü strukturlarda xarici və daxili dövrlər və onlar arasındakı münasibətlər müəyyənləşdirilməlidir. Proqramlaşdırma dillərində dövrü strukturların reallaşdırılması üçün ön şərtli və son şərtli dövrü operatorlardan istifadə olunur. Aşağıda ön şərtli strukturun sxemi verilmişdir.



Sonşərtli dövrü alqoritmik struktur isə aşağıdakı kimidir:



2. PROQRAMLAŞMA DİLLƏRİ

Məsələ həll edərkən əvvəlcə yerinə yetiriləcək əməliyyatların alqoritmi tərtib edilir, daha sonra bu əməliyyatlar hər-hansı alqoritm (proqramlaşdırma) dilində əmrlər şəklində yazılır.

Hesablama maşınlarının əsas fərqləndirici xüsusiyyətlərindən biri də onun proqramla idarə olunmasıdır. Yəni, istər sadə, istərsə də mürəkkəb məsələni maşının həll etməsi üçün proqram tərtib edilməlidir.

Proqram - maşının addım-addım yerinə yetirəcəyi təlimatlar və yaxud əmrlər toplusudur. Hər bir proqram tərtib edilərkən müəyyən bir alqoritmdən istifadə edilir. Yəni, proqram hər bir alqoritmi maşının başa düşəcəyi formada ifadə edir. Başqa sözlə proqram – maşının girişinə verilən informasiyaları çıxış informasiyalarına çevirən, xüsusi şəkildə tərtib olunmuş sonlu sayda ardıcıl əmrlərdən ibarət alqoritmdir.

EHM-də proqram yazmaq üçün istifadə olunan formallaşmış dillərə proqramlaşdırma dilləri deyilir. Proqramlaşdırma dilləri-proqram modullarından ibarət olub, məsələnin həll mərhələsinə hazırlığını təmin edir. Proqramlaşdırma dili süni dil olub, təbii dillərdən məhdud sayda sözlərin olması ilə fərqlənirlər. Bu dillərlə hazır proqramlar deyil, yalnız proqramın mətni yaradılır. Proqramlaşdırma dili proqramın tərkibində istifadə olunan qayda və əmrləri əhatə edir. Proqramlaşdırma dili-kompüterin alqoritimi başa düşməsi üçün istifadə edilir.

Proqram dilini kompüterin başa düşdüyü maşın dilinə çevirmək üçün translyatorlardan (translator – tərcüməçi) və kompilyatorlardan (compiler – tərtibatçı) istifadə edilir. Hər bir proqramlaşdırma dilinin öz translyatoru (kompilyatoru) olur.

Proqramlaşdırma dilləri çoxdur. Onlar iki hissəyə bölünür:

Aşağı səviyyəli dillər (Assembler, Avtokod və s.),

Yüksək səviyyəli dillər (Fortran, Alqol, Kobol, Basic, Pascal, Ci və s.).

Assembler – bu proqram vasitəsilə effektiv və kompakt proqramlar yaradılır. Assemblerlərdən sistem proqramlarının, drayverlərin, kompüterin aparat resurslarına müraciət üçün və s. proqramarın yaradılmasında istifadə edilir.

Bu dillər istifadə edildiyi maşının tipindən asılı olduğundan maşın yönümlü dillər adlanır. Maşın dillərindən istifadə hazırda geniş

yayılmamışdır. Belə ki, bu cür dillərdən istifadə edilməsi xüsusi hazırlıq və bacarıq tələb edir. Bu proqramlarla adətən peşəkar proqramçılar məşğul olurlar. Maşın dilində proqramın tərtibi olduqca çox zəhmət tələb edir, sonradan oxunması çətin olur. Başqa çatışmamazlığı ondan ibarətdir ki, eyni bir alqoritmin müxtəlif kompüterlərdə yerinə yetirilməsi üçün müxtəlif proqramlar yaratmaq lazım gəlir.

Hal-hazırda maşın kodundan istifadə olunmur və kompüter üçün proqram hər hansı proqramlaşdırma dillərinin birində tərtib olunur. Proqramlaşdırma dilləri bir qayda olaraq, verilənlərin təsviri, hesabi operatorlar, dövrün təşkili və idarəedici vasitələr, informasiyanın daxil və xaric olunma vasitələri ilə təmin olunub. Dillərin çoxunun proqramın tərtibində oxşar prinsipdən istifadə etməsinə baxmayaraq, onların sintaksisi müxtəlifdir.

Yüksək səviyyəli proqramlaşdırma dilləri (alqoritmik dillər) isə müxtəlif tipli maşınlarda istifadə edilə bilər. Alqoritmik dillər maşın dillərinə nisbətən daha sadə olub, geniş istifadəçi kütləsini əhatə edir. Proqramlaşdırma dillərində əmrlər operatorlar və ya təlimat adlanır. Hər hansı alqoritm üçün tərtib olunan bu cür əmrlər ardıcıllığı ilkin proqram və ya sadəcə ilkin mətn adlanır. Ilkin mətn çevirici proqram (translyator) vasitəsilə çevriləndən sonra icra olunur. Proqramlaşdırma dilləri də öz növbəsində interpretator və kompilyatorla işləyən dillərə bölünürlər. İnterpretatorla işləyən dillərdə proqram tərtib edildikdə hər yeni proqram sətrinin daxil edilməsi zamanı həmin sətirdə səhvin olub-olmadığı yoxlanılır və əgər səhv olarsa, yerinə yetirmə avtomatik olaraq dayandırılır. Kompilyatorla işləyən dillərdə isə proqram mətni tam daxil edildikdən sonra mövcud səhvlər haqqında məlumat verilir. Səhvlərin göstərilən nömrə və izahatlarına əsasən onlar uyğun şəkildə aradan qaldırılır.

İstənilən proqramlaşdırma dilinin əsas elementləri bunlardır: dilin əlifbası, sintaksisi və semantikası.

Dilin əlifbası dedikdə, həmin dildə işlənən bütün simvollar nəzərdə tutulur.

Sintaksis-əlifbada olan simvollardan dilin ayrı-ayrı konstruksiyalarının (komandaların, operatorların) düzəldilməsinin formal

qaydalarıdır.Bu qaydalar müxtəlif həll alqoritmlərini proq-ramlaşdırmağa imkan verir.

Semantika-dilin bu və ya digər sintaksis konstruksiyalarının təsviridir. Məsələn, əgər proqramın bu yerində y=a*(b+c) ifadəsinin hesablanması yazılıbsa, onda semantika qaydaları maşına "göstərir" ki, əvvəlcə (b+c) cəmini tapsın, sonra həmin cəmi a-ya vursun.

Qeyd etmək lazımdır ki, proqramlaşdırma dillərinin yaranma tarixi və inkişafı EHM-larının yaranma tarixi ilə sıx əlaqədar olmuşdur. Hal-hazırda bir çox proqramlaşdırma dilləri mövcuddur.

Alqoritmik dillərə FORTRAN, PL, Ada, C, Modula-2, COBOL, BASİC, Pascal və s. dillərini aid etmək olar. Maşın dillərinə isə ASSEMBLER tipli dilləri aid etmək mümkündür. Biz alqoritmik dillərdən biri olan Pascal dilinin Turbo Pascal 7.0 versiyası ilə tanış olacağıq. Pascal dili də yüksək səviyyəli proqramlaşdirma dilidir.

1971-ci ildə İsveçrənin Şürix şəhər Stenford Texniki Universiteti, informatika kafedrasının professoru Niklaus Virt yeni proqramlaşdırma dilini işləyib hazırladı və onu dahi fransız filosofu, riyaziyyatçısı və fiziki, hesabi əməliyyatları mexaniki yerinə yetirmək üçün ilk mexaniki hesablama maşını ixtira etmiş Blez Pascalın (1623-1662) şərəfinə Pascal adlandırdı.

ABŞ-ın Borland İnternasional, İnc firması fərdi kompüterlər üçün Turbo Pascal proqramlaşma sistemini işləyib hazırladı.

Turbo- Pascal dili 1992-ci ildə yaradilmisdir. Pascal dilində tərtib olunmus istənilən proqramı üç elementar strukturun kompozisiya-sı şəklində göstərmək mumkundur.

Bu elementar strukturlar aşağıdakılardır:

1. Xətti struktur
2. Seçmə və şərti struktur
3. Dövrü struktur

Xətti struktura malik olan proqramları təşkil edən əmrlər bir-birinin ardınca icra olunurlar.

Seçmə və şərti struktura malik olan proqramları təşkil edən əmrlərin ardıcıl yerinə yetirilməsi verilmiş şərtdən asılı olaraq pozulur və hesablama prosesi əvvəlcədən məlum olan istiqamətlərdən biri üzrə ğedir.

Bir və ya bir neçə əmrlər qrupunun yerinə yetirilməsi sonlu sayda təkrar olunarsa, onda dövrü strukturadan istifadə edilir.

2.1. Pascal dilinin əlifbasi

Turbo Pascal dilinin əlifbasına hərflər, onluq say sisteminin rəqəmləri, onaltılıq say sisteminin rəqəmləri, xüsusi simvollar, açar sözlər və s. daxildir.

Əlifbanın tərkib hissəsi aşağıdakı kimidir:

1. Dilin əlifbası–latın əlifbasının böyük və kiçik hərfləri (A ÷ Z);
2. 0-dan 9-a qədər ərəb rəqəmləri;
3. 0...9, A, B, C, D, E, F –onaltılıq say sisteminin rəqəmləri və onlardan düzəldilmiş ədədlər;
4. Xüsusi işarələr: +, -, *, /, =, ., ,, ;, :, <, >, [ ], ( ), { }, ^, @, $, #, &, (* *), (. .).

5. Münasibət işarələri:

< - kiçikdir; > - böyükdür; >= - böyükdür bərabərdir; <=- kiçikdir bərabərdir; = - bərabərdir; := - mənimsətmə operatoru.

6. Hesabi əməllər:

+ - toplama; - - çıxma; *- vurma; / - bölmə; div – tam bölmə; mod- bölmə əməlindən alınan qalıq

7. Məntiqi əməllər:

and- iki tam ədədin bitləri üzrə məntiqi **və** əməliyyatı;

or – iki tam ədədin bitləri üzrə məntiqi **və ya** əməliyyatı;

not – tam ədəin bütün bitləri üzrə **unar inversiya** əməliyyatı;

xor –iki tam ədədin bitləri üzrə **istisnalı məntiqi və ya** əməliyyatı.

Turbo Pascal dilində bir sıra açar sözlərdən istifadə edilir. Turbo Pascal dilində xidməti sözlər öz vəzifəsinə görə dəqiq təyin edilərək, dəyişdirilə bilməz. Buna görə də identifikatorların yazılışında açar sözlərdən istifadə edilə bilməz. Yəni, bu sözlərdən dəyişən və ya sabitlərin adlandırılmasında istifadə edilməməlidir. Bu sözlərdən bəzilərini qeyd edək: case, begin, const, div, do, downto, else, end, real, read, file, for, array, function, goto, if, in, label, mod, not, of, packed, proce-

dure, set, concat, succ, length, insert, val, ord, inc, pred, chr, string, program, record, repeat, then, to, type, until, uses, var, while, with, və s.

Turbo Pascal dilində sabit, dəyişən, nişan, tip, prosedur, funksiya, modul, proqram və yazı sahələrinin adlandırılması məqsədilə identfikatorlardan istifadə edilir. İdentifikator ixtiyari uzunluğa malik, ilk simvolu hərf olan simvollar ardıcıllığıdır. Proqram daxilində identifikatorun yalnız ilk 63 simvolu nəzərə alınır.

3. PASCAL DİLİNİN ELEMENTLƏRİ

3.1. Verilənlərin növləri

Pascal dilində verilənlər ədədi, məntiqi, simvol, sətir, göstərici, sadalanan və məhdud tipli olur.

Pascal dilində verilənlər sadə, struktur, göstərici, sətir və prosedura tipli olur. Sadə tiplərə nizami və həqiqi tiplər aiddir. Nizamlanan tiplərə tam, məntiqi, simvol, sadalanan və diapozon tipləri aiddir. Struktura tiplərinə massivlər, yazılar, çoxluqlar və fayllar aiddir.

Turbo-Pascal dilində verilənlərin aşağıdakı növləri vardır:

1. Tam tiplərə aşağıdakılar aiddir:

-İşarəsiz qısa tam (byte) (1b, $0 \div 255$)
- işarəli qısa tam (shortint) (1b, $128 \div 127$)
- işarəsiz tam (word) (2b, $0 \div 65535$)
- işarəli tam (integer) (2b, $32768 \div 32767$)
-işarəli uzun tam (longin) (4b, $2147483648 \div 2147483647$)

2. Həqiqi tiplərə aşağıdakılar aiddir:
- siqle (4 bayt) –(7 – 8 rəqəm)
- real (6 bayt) – (11-12 rəqəm)
- double (8 bayt) – (15-16 rəqəm)
- extended (10 bayt) – (19 - 20 rəqəm)
- comp (8 bayt) – (19 - 20 rəqəm)

3. Məntiqi və ya bul növü (boolean).
4. Simvol(char)
5. Skalyar və məhdud növlər
6. Sətir(string)
7. Ünvan-göstərici (pointer)
8. Mürəkkəb növlər:
- düzüm (array…of)
- yazı (record)
- çoxluq (set of…)
- obyekt (object)
- fayl (text, file, file of…)
- müraciət (baza tipi)

Pascal dilində proqram tərtib edərkən identifikatorlardan sabitlərə, dəyişənlərə, prosedura və funksiyalarə ad vermək üçün istifadə edilir. İdentifikator hərf və rəqəmlərdən ibarətdir və onun birinci simvolu hərf olmalıdır. Standart adlar və işçi sözlər identifikator olaraq işlənə bilməz. Standart adlar standart funksiyaların, proseduraların, standart faylların və sabitlərin növlərinin adları ola bilər.

Proqramda verilənlərin təsviri nişanların və sabitlərin təyinindən sonra gəlir. Əgər proqramda nişan və sabitlər yoxdursa, onda verilənlərin təsviri proqramda birinci yazılır. Proqramda verilənlərin təsviri olan hissədə dəyişənlərin hansı növə aid olduğu göstərilir.

Verilənlərin təsviri (VARIABLES) VAR işçi sözü ilə başlayır.

Formatı belədir:

VAR <dəyişənlərin siyahısı>:<tipi>;

3.1.1. Tam dəyişənlər (integer və byte)

Tam dəyişənlər -32768 ÷+32767 qiymətlər alır, yaddaşda 2 bayt yer tutur. Tam dəyişənlər üzərində aşağıdakı əməliyyatları aparmaq olar.

:=(mənimsətmək), +,-,*, div (tam bölmə), mod(bölmə nəticəsində alınan qalığın hesablanması).

Misal:

Tutaq ki, A, B –tam dəyişən, A=17, B=3 olarsa:

```
VAR  A, B: INTEGER;
A DIV B=5
A MOD B=2
```

olar.

Tam dəyişənin qiymətini həqiqi dəyişənə mənimsətmək olar, əksinə olmaz.

Byte tipli dəyişənlər də tam tipli dəyişənlərin bir növüdür. Bu tipli dəyişənlər 0÷255 qiymətlər ala bilər. Yaddaşda 1 bayt yer tutur.

3.1.2. Həqiqi dəyişənlər (real)

Həqiqi ədədlər mantissası 11 ikilik mərtəbə olan həqiqi dəyişənlər 2,9E-39÷1,7E+38 qiymətlər alır və bu tipli dəyişənlər yaddaşda 6 bayt yer tuturlar.

Həqiqi dəyişənlər üzərində aşağıdakı əməliyyatlar aparmaq olar: :=; +; -; *; /

Əgər ifadədə həm həqiqi, həm də tam dəyişən iştirak edirsə, onda tam dəyişən həqiqi dəyişənə çevrilir:

VAR A,B,Y:REAL; I:INTEGER;

olarsa, onda Y=A+B*I ifadəsinin qiyməti həqiqi ədəd olar.

Aşağıdakı funksiyaların arqumentləri həqiqi ədədlər olduqda, qiymətləri tam ədədlər olur.

trunc(x) funksiyası x-in tam hissəsini hesablayır.

trunc (3,15)=3, trunc (-4,87)=-4

round(x)- bu funksiyanın qiyməti x-ə yaxın olan tam ədəddir:

round(3,15)=3, round(3,97)=4

3.1.3. Məntiqi və ya bul dəyişənlər (boolean)

Bu tipə malik dəyişənlər iki qiymət ala bilər: true (doğru) və false(yalan). Yaddaşda bir bayt yer tutur.

Bu qiymətlər üçün aşağıdakı münasibətlər doğrudur:

true >false

Bul dəyişənlər üzərində aşağıdakı əməliyyatları aparmaq olar:

Or-məntiqi toplama əməli; And-məntiqi vurma əməli; Not-inkar

Bul tipli dəyişənlər üzərində müqayisə əməliyyatları aparmaq olar: >, <, =, <=, >=, <>

VAR A,B,C,D : BOOLEAN;

D:=(AB) AND (C=A) və ya

C:=NOT(AB) OR (B)

Bul tipli dəyişənlər üzərində standart funksiyalarla işləmək olar.

Odd(x)- x arqumenti tam tipə malik dəyişən olmalıdır.

Əgər x cüt olarsa, onda funksiyanın qiyməti true (doğru), əks halda false(yalan) olur:

ODD(2) TRUE; ODD(19) FALSE

EOLN- ilkin verilənləri daxil edən zaman sətrin sonu çatdıqda funksiya doğru qiymət, əks halda yalan qiymət alır.

EOF- giriş verilənlərinin oxunması sona çatdıqda bu funksiyanın qiyməti doğru, əks halda yalan olur.

3.1.4. Simvol tipli dəyişənlər (char)

Bu tipə malik dəyişənlərin qiymətləri dırnaq arasında yazılır. Simvol tipli verilənlər üzərində müqayisə və mənimsətmə əməliyyatları aparmaq olar.

Misal:

```
VAR  A, B, R : CHAR;
C:='ADAU';
R:= 'Gəncə';A:= '+-*/';
```

3.1.5.Sətir tipli dəyişənlər (string)

Sətir tipli dəyişənlər simvol tipli dəyişənlərin bir növüdür. Sətir tipli dəyişənləri təyin edən zaman sətrin maksimal uzunluğunu (simvolun sayını) göstərmək lazımdır. Bu tipə malik olan dəyişən STRING sözünün köməyi ilə təyin olunur.

String tipli dəyişən aşağıdakı üsullarla təsvir edilə bilər:

VAR <sətir_dəyişəninin_adı>:STRING; və ya

VAR <sətir_dəyişəninin_adı>:STRING[N];

Məsələn,

VAR S1:STRING[10]; S2:STRING[128]; S2:STRING;

-**Göstərici tipi.** Standart göstərici tipinin identifikatoru Pointer-dir. Pointer tipinin elementləri, istifadəçi tərəfindən təyin edilən göstərici tiplərindən fərqli olaraq, ixtiyari tipli dəyişənin ünvanını özündə saxlayır.

-**Mətn tipi.** Mətn fayllarını təsvir etmək üçün Text standart mətn tipindən istifadə edilir.

3.2. PASCAL DİLİNDƏ İSTİFADƏ OLUNAN DİGƏR TİPLƏR

Sadalanan tip – verilənlərin nizamlanmış çoxluğundan ibarətdir. Sadalanan tiplərin yazılışı bölməsində göstərilir və TYPE açar sözü ilə bağlayır.

Məsələn:

```
TYPE
DIRECNION = (NORTH, SOUTH, EAST,WEST);
GEOFIG = (UCBUJ, KWADR, DUZBUJ, QURE, CHOXB);
UZUN = (MMETR, CMETR, DMETR, METR, KMETR);
```

Yeni tip müəyyən edildikdən sonra həmin tip üzrə dəyişənləri elan etmək mümkündür.

```
VAR  ROUT, ALTERNATE: DIRECNION;
   FIGUR: GEOFIG;    MESAFE: UZUN;
```

Hər iki yazılışı bir bölmədə birləşdirmək olar:

```
VAR  OUT, ALTERNATE: (NORTH, SOUTH, EAST, WEST );
FIGUR: (UCBUJ, KWADR, DUZBUJ, QURE, CHOXB);
MESAFE: (MMETR, SMETR, DMETR, METR, KMETR);
```

Sadalanan tipə daxil olan sabitlər o-dan başlayaraq ardıcıl tam ədədlər uyğun gəlir. Standart funksiyanın qiyməti isə sadalanan tipli dəyişənin, yaxud sabitin sıra nömrəsinə bərabərdir.

Məsələn, baxdığımız nümunələr üçün aşağıdakılar daxildir:

```
ORD (EAST)=2  PRED (WEST)=EAST
ORD (UCBUJ)=0    Ssucc (METR) = KMETR
```

Sadalan tipdər üzərində müqaisə əməliyyatları aparmaq olar. Məs.,

```
   EAST > NORTH = 3  > 0 NƏTICƏ TRUE
   UCBUJ  > QURE = 0 > 3 NƏTICƏ FALSE
```

Məhdud tip - hər hansı baza tipli ardıcıl kəmiyyətlər çoxluğudur. Məhdud tip qiymətlərin dəyişmə diapozonunu verən iki sabitlə ifadə olunur.

```
TYPE    temperatur = 20..40;
il = 2000 .. 2004;
```

Məhdud tipin aşağı sərhədi yuxarı sərhəddən böyük olmalıdır. Məhdud tip dəyişənlər bölməsində yazılır.

VAR size : temperatur;

year : il; xd, yp, zp: k;

Məhdud tipli dəyişəni yazmaq üçün istifadə olunan iki bölməni VAR bölməsində birləşdirmək olar:

VAR size : 20..40;

YEAR: 2000..2004;

Məhdud tipli dəyişənlər mənimsətmə operatorunun hər iki tərəfində işlədilə bilər. Sadalanan tiplər üçün müəyyən olunmuş funksiyaların məhdud tiplər üçün də tətbiqi mümkündür.

Yazılar

Yazı məntiqi baxımdan bir-biri ilə bağlı olan elementlərin birləşməsidir. Yazının elementinə sahə deyilir.

Yazının ümumi forması aşağıdakı kimidir:

P=RECORD u_1, t_1; u_2, t_2; u_3, t_3; …u_n:t_n END;

P – yazının a, u_i – sahənin adı, t_i – REAL, INTEGER, CHAR və ya əvvəlcədən müəyyən edilmiş, tipli sahənin tipidir. t – tipi D-yə nəzərən baza tipidir.

TYRE DUZBU = RECORD HPR, WPR:REAL END;

TYPE UCHBU= RECORD SID1, SID2, ALFA:REAL END;

Burada, duzbu düzbucaqlasi eni (hpr), uzunluğu (wpr), uchbu üçbucaği (sid1, sid2) tərəfi və onlar arasinda qalan alfa bucaği ilə ifadə olunub.

Birləşdirmə operatoru WITH

Eyni bir yazının sahələrinə bir neçə dəfə müraciət etmək lazım gəlirsə, birləşmə operatorundan istifadə edilir. Birləşdirmə operatorunun ümumi yazılışı

WITH U DOS;

kimidir. U- yazının adı, S – operatordur.

Çoxluqlar

Çoxluq – eyni tipli elementlərin yığımıdır. Çoxluğun elementləri şkalyar tipli (REAL – dan başqa) olur. Çoxluğun elementlərinin tipi baza hesab edilir.

Çoxluq ümumi şəkildə:

TYPE <növün adı>= SET OF <1-ci...n-ci element>

VAR <identifikator>: SET OF <1-ci...n-ci element>

kimi yazılır.

4. PASCAL DİLİNDƏ İFADƏLƏR

İfadə-operandlardan, dairəvi mötərizələrdən və əməl işarələrdən ibarət olub, verilənlərin elementləri üzərində əməllərin yerinə yetirilməsi qaydasını müəyyən edir. İfadələrin aşağıdakı növləri var:

-hesabi ifadələr; -müqayisə əməliyyatları ilə olan ifadələr;- məntiqi ifadələr; -simvol və sətir ifadələr.

Bəzi triqonometrik funksiyaları Pascal dilinin standart funksiyaları ilə ifadə etmək üçün aşağıdakı düsturlardan istifadə edilir:

$$\mathrm{tgx} = \frac{\mathrm{sinx}}{\mathrm{cosx}}$$

$$ctgx = \frac{\cos x}{\sin x}$$

$$\arcsin x = arctg\,\frac{x}{\sqrt{1-x^2}}$$

$$\arccos x = arctg\,\frac{\sqrt{1-x^2}}{x}$$

$$\sec x = \frac{1}{\cos x}$$

$$\cos ecx = \frac{1}{\sin x}$$

Pascal dilində standart funksiyalar

Funksiyanın adı	Yerinə yetirdiyi funksiya	Arqumentin tipi	Funksiyanın tipi
abs(x) sqr(x)	$\|x\|$ x^2	real, integer	real, integer
sin(x) cos(x) exp(x) ln(x) sqrt(x) arctan(x) exp(a*ln(x))	sinx cosx exp x ln x $\sqrt{x}$ arctg(x) x^a	real, integer	real
trunc(x) round(x)	Ədədin tam hissəsi Ədədin yuvarlaq qiyməti	real	integer
pred(x) succ(x)	Əvvəlki element Sonrakı element	Integer real, boolean	integer real,boolean
ord(x)	Simvolun sıra nömrəsi	Char,boolean	integer
chr(x)	sıra nömrəsinə görə simvolun işarəsi	integer	Char
odd(x)	Ədədin cüt olub-olmamasını müəyyən edir	integer	boolean

Mötərizələrin və əməliyyat işarələrinin köməyi ilə əlaqələndirilmiş sabitlər, dəyişənlər və standart funksiyalardan ibarət ifadələrin yazılışına dair nümunələr göstərək.

Misal 1

z=$\sin^2 x$ funksiyasının Pascal alqoritmik dilində yazışı aşağıdakı kimidir:

z:=sqr(sin(x))

Misal 2

z=$\sin x^2$ funksiyasının Pascal alqoritmik dilində yazışı aşağıdakı kimidir:

z:= sin(sqr(x))

Misal 3

$$z = \sqrt{\ln\left|(a+5)-4\right|} \cdot \sqrt{\left|a+y\right|} + 8{,}294e^{\frac{x-4}{2x+3}}$$

funksiyasının Pascal alqoritmik dilində yazışı aşağıdakı kimidir:

z:=sqrt(ln(abs((a+5)-4))*sqrt(abs(a+y))+8.294*exp((x-)/(2*x+3))

Misal4

$$p = \frac{a+b}{a-b}\sqrt{\left|b^6 - 3b\right|} + 2{,}51\cdot 10^3 tg\left(\frac{b-3}{a+2}\right) - \frac{\sin^2(x-3)}{\cos(x+2)^2}$$

funksiyasının Pascal alqoritmik dilində yazışı aşağıdakı kimidir:

p:=(a+b)*sqrt(abs(exp(6*ln(b))))/(a-b)+2.51e+3*sin((b-3)

/(a+2))/cos((b-3)/(a+2))-exp(2*ln(sin(x-3)))/cos((x+2)*(x+2))

Misal 5

$$a = \frac{\left|x+1\right| - \sqrt[3]{\left|y\right|}}{1 + x^2/2 + y^2/4}$$ funksiyasının Pascal alqoritmik dilində yazışı aşağıdakı kimidir:

a:=(sqrt(abs(x+1))-exp(1/3*ln(abs(y))))(1+sqr(x)/2+ sqr(y)/4

5. PASCAL DİLİNDƏ PROQRAMIN QURULUŞU

Proqramdan məsələnin həll alqoritminin icrası zamanı istifadə olunur. Pascal dilində proqram, maksimal uzunluğu 127 simvoldan az olan sətirlərdən ibarətdir. Sətirlər istənilən sütundan başlaya bilər. Pascal dilində proqramın strukturu proqram başlığı və blokdan ibarətdir.

Proqram başlığı birinci sətirdə yazılır və PROGRAM işçi sözündən, proqramın adından və dairəvi mötərizə içərisində yerləşən parametrlərdən ibarətdir:

PROGRAM <proqramın adı><(parametr)>;

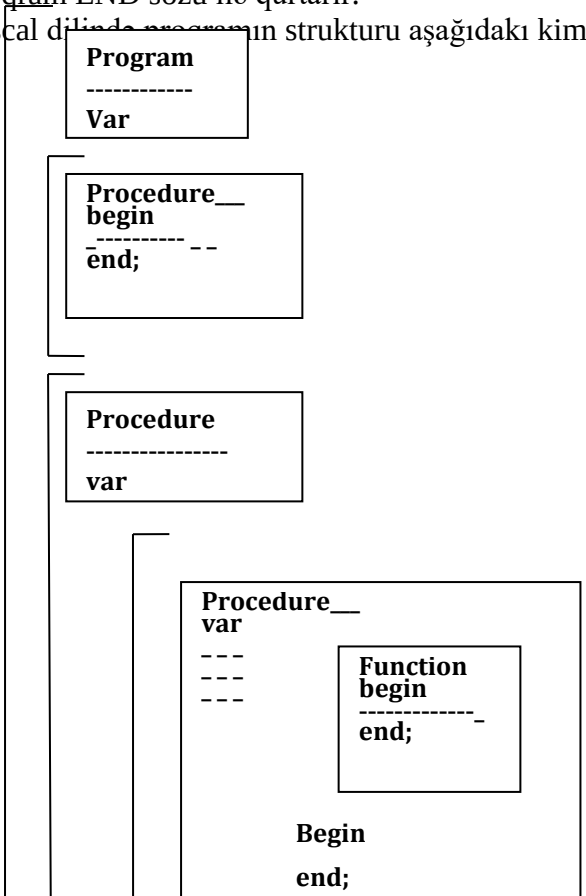
PROGRAM ADAU (INPUT,OUTPUT);

Burada INPUT və OUTPUT standart giriş və çıxış fayllarıdır.

Proqram 6 bölmədən ibarətdir: sabitlər, nişanlar, tiplər, dəyişənlər, funksiya və proseduralar, operatorlar. Bölmələr arasında nöqtəli vergül qoyulur.

Proqram END sözü ilə qurtarır.

Pascal dilində proqramın strukturu aşağıdakı kimidir:



5.1. Nişanlarin təsviri

Pascal dilində hər bir operatordan əvvəl nişan qoymaq olar. Proqramın istənilən yerində həmin nişandan istifadə etməklə onun aid olduğu operatora müraciət etmək olar. Nişan identifikatordan və ondan sonra dələn iki nöqtədən ibarətdir. Nişan ən çoxu dörd simvoldan ibarətdir. Nişanlar istifadə olunmamışdan əvvəl LABEL işçi sözü ilə təsvir olunmalıdır:

```
LABEL <nişanın adı>;
Misal:
LABEL M1;
LABEL M2;M3;
……
BEGIN
…….
M1: <operator>;
…….
M2: <operator>;
…….
M3:<operator>;
END;
LABEL 50;
.....
BEGIN
…….
50: <operator>;
.....
END.
```

5.2. Sabitlərin təsviri

Sabitlərin qiymətləri proqram yerinə yetirilməmişdən əvvəl məlum olur və həmin proqram yerinə yetirilib qurtarana kimi dəyişməz qalır.

Sabitlərin qiymətlərini proqramda vermək üçün CONST işçi sözündən istifadə olunur. İstifadə olunan sabitlərin qiymətləri proqramın əvvəlində, yəni verilənlərin təsvirindən qabaq verilir. Sabitlər tam, həqiqi, simvol olur.

Formatı aşağıdakı kimidir.

CONST <identifikator>=<qiymət>;

Misal:

CONST M=4;M7=8; KOD:='İNFORMATİKA';

Sabit müəyyən olunduqdan sonra ona başqa qiymət mənimsətmək olmaz.

5.3. Dəyişənlərin təsviri

Dəyişənlər VAR açar sözü ilə başlayır və ondan sonra tipləri göstərməklə dəyişənlər yazılır.

```
VAR
A,B,C:INTEGER;          [ tam tiplidir]
R,Z:REAL;                  [həqiqi tipli]
T:BOOLEAN                [məntiqi tipli]
K:CHAR                      [simvol]
G:R; D:ARRAY[1..60] OF INTEGER;[tam tipli massiv]
```

5.4. Prosedura və funksiyalar bölməsi

Pascal alqoritmik dilində proqramın tərtibini asanlaşdırmaq məqsədilə çoxsaylı prosedurlar nəzərdə tutulmuşdur.

Bu bölmədə alt proqramlar yerləşir. Alt proqram ada malik proqram vahididir. Ona proqramın digər yerlərindən müraciət etmək olar.

Pascal dilində alt proqram rolunu prosedura və funksiyalar oynayır. Alt proqramı təsvir etmək üçün PROCEDURE və FUNCTION işçi sözlərindən istifadə olunur. Onlar alt proqramın əvvəlində yazılırlar.

Funksiya proseduraya oxşardır. Onların iki fərqi vardır:

- müraciət nöqtəsinə funksiya skalyar qiymət ötürür;
- funksiyanın adı ifadədə iştirak edə bilir.

Pascal dilində bütün prosedurlar və funksiyalar iki qrupa ayrılır:

- proqramlaşdırıcı tərəfindən təyin olunan prosedura və funksiyalar;
- standart prosedura və funksiyalar.

Mənaca bir-birinə yaxın olan proseduralar qrup halında birləşdirilir ki, bu da modul adlanır. Əsas modullar aşağıdakılardır:

Sistem- standart funksiya, giriş-çıxış operatorlar;
Strings- sətirlərlə işləmək üçün;
Winapi- dinamik dəyişənlərlə işləmək üçün;
Windos- fayl və kataloqlarla işləmək üçün;
Winert- Ekran və pəncərələrlə işləmək üçün;
Winprn- müxtəlif çap məhsulları almaq üçün nəzərdə tutulur.

Əgər bu modulların hər hansı birindən istifadə olunarsa, onda o, hökmən aşağıdakı operatorların vasitəsilə qeyd olunmalıdır:

Uses- <modulların siyahısı>

Qeyd edək ki, bu modullardan yalnız sistem modulu yazılmaya bilər.Bəzi hallarda istifadəçilər tərəfindən modullar yaradılır və belə modullar qeyri-standart prosedura adlanır və Uses operatorunun köməyilə qeyd olunur.

5.5. Operatorlar bölməsi

Bu bölmə proqramın əsas bölməsidir və bütün əvvəlki bölmələr burada döstərilən operatorların yerinə yetirilməsinə xidmət edir. Operatorlar bölməsi BEGIN işçi sözü ilə başlayır, sonra isə bir-birindən nöqtə vergüllə (;) ayrılan dilin operatorları gəltr. Bölmənin sonu END işçi sözü ilə qurtarır. END-dən sonra nöqtə qoyulur.

```
BEGIN
<operator>
.....
<operator>
END.
```

6. ŞƏRHLƏRİN VERİLMƏSİ

Şərhlər proqramın istənilən yerində yazıla bilən izahedici mətnlərdən təşkil olunurlar.

Şərhin mətni { } və ya (* *) simvolları arasında yazılır. Onlar proqramın yerinə yetirilməsinə təsir göstərmirlər.

```
PROGRAM SS;
(* SAHƏNİN HESABLANMASI*)
```

7. GİRİŞ OREPATORLARI

Verilənlərin girişi dedikdə, informasiyanın xarici daşıyıcıdan əsas yaddaşa ötürülməsi prosesi başa düşülür.

Giriş operatorları ədədi verilənləri əsas yaddaşa daxil edilməsini təmin edir. Giriş əməliyyatlarını yerinə yetirmək üçün operatorları READ və READLN operatorlarından istifadə edilir. Operatorun formatı belədir:

```
READ(X1, X2,…XN);
```

və ya

```
READ(FV, X1, X2,…XN);
READLN(X1, X2,…XN);
```

Burada X1, X2,…XN-mümkün növə aid olan dəyişənlər, FV isə fayla əlaqədar olan dəyişəndir. Klaviatura üçün təyin olunmuş standart giriş faylının adı INPUT-dur. (LN-LINE sözüdür)

X1, X2,…XN-dəyişənlərinin qiymətləri arasında Ən azı bir probel olmaqla klaviaturadan yığılır və ekranda göstərilir.

Misal:

```
VAR  I:INTEGER;  J:REAL;
K: CHAR;
BEGIN
READ(I,J,K);
```

Həllə göndərən zaman

15.28 75 'G'

yazıb ENTER düyməsi sıxılır.

Bir proqramda bir neçə READ operatorundan istifadə etmək olar.

READLN operatoru ilə READ operatorunun fərqlənən cəhəti ondan ibarətdir ki, birinci READLN operatorundakı axırıncı dəyişənin qiyməti daxil edildikdən sonra növbəti READLN operatorundakı dəyişənlərin qiymətləri yeni sətirdən oxunur.

8. ÇIXIŞ OPERATORLARI

Çıxış və ya yazma operatorlarından (WRITE, WRITELN) ədədi verilənləri, simvolları, sətir və bul qiymətlərini çıxış qurğularına(ekrana

və ya çap qurğusuna) vermək üçün istifadə olunur. Operatorun formatı belədir:

```
WRITE (Y1, Y2,…YN)
```

və ya

```
WRITE (FV, Y1, Y2,…YN)
WRITELN (Y1, Y2,…YN)
```

Burada, Y1, Y2,…YN INTEGER, REAL, CHAR və sair növə malik ifadələrdir. FV isə çıxış faylının adıdır. Standart çıxış faylının adı OUTPUT-dur.

Tam ifadələri vermək üçün :

```
WRITE(tam ifadə:sahənin uzunluğu);
WRITE(N:6)
```

Əgər N=250 olarsa, onda nəticə

```
  250
```

olar.

Həqiqi ifadələri:

```
WRITE(həqiqi ifadə:sahənin uzunluğu:dəqiqlik);
```

Dəqiqlik onluq nöqtədən sonra gələn rəqəmlərin sayını göstərir. Bu halda standart Pascal dilində tam ədədlər üçün 10 mövqe, həqiqi ədədlər üçün isə 20 mövqe götürmək olar.

Misal:

```
WRITE(X) ;
WRITE('Y=', Y:5:2);
WRITE('MASSİV');
WRITE(2.25+7.1*2.49);
WRITE(' Z=', Z, ' A=', A);
```

WRITELN operatoru WRITE operatoruna oxşardır. Fərqi odur ki, birinci WRITELN operatorunda olan axırıncı dəyişənin qiyməti çıxışa verildikdən sonra növbəti sətrə keçid təmin olunur. Növbəti çıxış operatorundakı dəyişənlərin qiymətləri təzə sətirdən çap olunurlar.

9.MƏNİMSƏTMƏ OPERATORU

Mənimsətmə operatoru yerinə yetirilən zaman mənimsətmə işarəsindən (:=) sağ tərəfdəki ifadənin qiyməti hesablanır və bu qiymət sol tərəfdə duran dəyişənə mənimsədilir. Dəyişən və ifadə eyni növə

malik olmalıdır. Xüsusi halda, dəyişən həqiqi növə, ifadə isə tam növə aid ola bilər. Fayl növündən başqa növlər üçün mənimsətmə operatorundan istifadə etmək olar. Ümumi formatı belədir:

<identifikator>:=<ifadə>;

Misal:

V:=30;S1:=-12.5;N:='informatika';Z:=z+1;

10.ŞƏRTİ KEÇİD OPERATORLARI

Pascal dilində alqoritmləri proqramlaşdırılması üçün şərtsiz keçid və şərti keçid operatorlarından istifadə edilir.

Şərtsiz keçid operatoru.

Proqramlaşdırmada bəzi hallarda operatorun yerinə yetirilməsi ardıcıllığının dəyişdirilməsi lazım gəlir. Bunun üçün GOTO şərtsiz keçid operatoru mövcuddur ki, onun yazılışı aşağıdakı kimidir:

GOTO nişan;

Bu operatorun köməyilə idarəni prosedur və funksiyanın daxilinə və ya xaricinə ötürmək olmaz. Turbo Pascalda istifadə olunan nişanın 0÷9999 intervalında tam ədəd və adi identifikator tipi var. Bütün nişanlar Label xidməti sözü ilə başlayan nişanın təsvir bölməsində göstərilməlidir. Məs., Label 5, 6, a;

Qeyd etmək lazımdır ki, goto operatoru struktur proqramlaşdırmanın əksinədir və ondan proqramlaşdırmada istifadə edilməsi məsləhət görülmür. Bunu nəzərə alaraq, goto operatorunun tez-tez istifadə edildiyi hallar üçün Turbo Pascala **Break** və **continue** prosedurları daxil edilib.

Boş operator. Bu operator heç bir əməliyyatı yerinə yetirmir və şərtsiz keçid operatorundan müraciət lazım olduqda istifadə olunur.

Şərti keçid operatorları verilmiş şərtdən asılı olaraq hər hansı operatorun və ya operatorlar qrupunun yerinə yetirilməsini və ya yerinə yetirilməməsini təmin edir. İki şərti keçid operatoru vardır : **İF və Case**

I. İF operatoru aşağıdakı iki haldan biri kimi istifadə oluna bilər:

1. IF <şərt> THEN <operatorlar1>ELSE<operatorlar2>;
2. IF <şərt> THEN <operatorlar1>;

Burada, <şərt>- məntiqi ifadədir. Birinci halda, məntiqi ifadənin qiyməti doğru olarsa, onda <operatorlar1>, əks halda <operatorlar2>

yerinə yetirilir. Burada <operatorlar1>, <operatorlar2> tək bir operatordan və ya BEGİN və END arasında yerləşən operatorlar qrupundan ibarətdir. İkinci halda məntiqi ifadənin qiyməti doğru olarsa, onda <operatorlar1>, əks halda proqramdakı növbəti operator yerinə yetirilir.

Bir IF operatoru digər IF operatorunun tərkibinə daxil ola bilər:

IF <şərt> THEN IF <şərt> THEN <operatorlar1> ELSE <operatorlar 2>;

Bu halda hər bir ELSE özündən əvvəl gələn THEN- ə aid olur.

Misal1:

```
PROGRAM MAX;
VAR A,B,M: REAL;
BEGIN
READ (A,B);
IF A>B THEN M=A ELSE M=B;
WRITE (M) ;END.
```

Misal 2:

```
IF (A=C) AND C=D THEN F:=2*X ELSE
BEGIN
F:=5*X;
E:=2*F;
END.
```

II. CASE () of ()

CASE seçmə və ya variant operatoru İF operatorunun ümumiləşdirməsidir. Bu operator yerinə yetirildikdə mümkün variantlardan birini seçir. Bu operator selektor (seçmə şərti-seçici) adlanan ifadədən və seçmə üçün istifadə olunan sabitlərin siyahısından ibarətdir. İF operatorunda olduğu kimi burada da ELSE sözü işlənə bilər.

Ümumi formatı aşağıdakı kimidir:

```
CASE <selektor ifadə> OF
   <Sabit 1>:<operatorlar 1>;
 <Sabit 2>:<operatorlar 2>;
. . .        . . .
<Sabit n>:<operatorlar n>
   ELSE <operatorlar>
END;
```

Burada, <Sabit 1>,<Sabit 2>,…<Sabit n>-ə seçmənin qiymətləri (nişanlar) deyilir.

CASE operatoru aşağıdakı kimi işləyir:

Əvvəlcə selektor – ifadənin qiyməti hesablanır, sonra isə selektor – ifadənin qiymətinə olan sabit müəyyənləşdirilir və ona uyğun olan operator yerinə yetirilir. Selektorun qiymətinə bərabər olmadıqda ELSE operator yerinə yetirilir. Əgər ELSE sözü yoxdursa, onda END- dən sonra gələn operator yerinə yetirilir. Həqiqi və sətir növünə aid olan dəyişənləri selektor olaraq istifadə etmək olmaz.

Misal: Aşağıdakı funksiyanın qiymətinin hesablanmasını CASE operatorundan istifadə etməklə yazaq.

$$Z = \begin{cases} i + 10, & \text{i} = 1 \\ i + 20, & \text{i} = 2 \\ i + 30, & \text{i} = 3 \end{cases}$$

```
CASE İ OF
        1: Z:=İ+10
        2: Z:= İ+20
        3: Z:=İ+30
      END;
```

Burada selektorun qiymətləri 1, 2, 3 tam ədədlərdir.

Misal:

```
PROQRAM RR;
VAR A: INTEGER;Y,X: REAL;
BEGIN
READ(A);
CASE A*A OF
16: X:=4;
9: X:=3;
END; X:=5;WRITELN (‘X=’, X:3:1)
END.
```

CASE-nin qarşında A ifadəsi hesablanıb, alınan cavaba uyğun nömrəli sətrə (16 və ya 9-a) müraciət edir, əks halda END-dən sonrakı operatora (X:=5;) kecir.

11. DÖVR OPERATORLARI

Dövr operatorları dövrü alqoritmlərin proqramlarını tərtib etmək üçün istifadə edilir. Üç növ təkrarlanma operatoru vardır:

1. **FOR** (parametrli dövr operatoru)
2. **REPEAT** (son şərtli dövr operatoru)
3. **WHILE** (ön şərtli dövr operatoru)

Dövrlərin sayı məlum olduqda **FOR** operatorlarından, əks halda **REPEAT və WHILE** operatorlarından istifadə edilir.

I. Parametrli dövr operatoru aşağıdakı kimi iki formada verilə bilər:

1. FOR <dövrün parametri>:=<s1> TO <s2> DO <operator>;
2. FOR <dövrün parametri>:=<s1> DOWNTO <s2> DO <operator>.

Burada, s1 və s2 dövrün parametrinin uyğun olaraq başlanğıc və son qiymətlərini müəyyən edən ifadələrdir. Dövrün parametrinin, s1 və s2 –nin qiymətlərinin növü eyri olmalıdır. Bunlar həqiqi növə aid ola bilməzlər.

<operator>- dövrün gövdəsi adlanır. <operator>- sadə (tək bir operatordan) və mürəkkəb operatordan (BEGIN və END arasında yerləşən operatorlar qrupundan) ibarət ola bilir.

Əgər dövrün başlığı FOR … TO isə, onda dövrün parametrinin qiyməti hər dəfə vahid qədər artırılır, yox əgər FOR … DOWNTO isə, onda dövrün parametrinin qiyməti hər dəfə vahid qədər azalır. Bu proses dövrün parametri son qiymət alana kimi davam edir.

Misal:

FOR I:=1 TO 50 DO K:=I;

Burada - K=1,2,…50 qiymətlərini alır.

FOR I:=50 TO 1 DOWNTO K:=I;

Burada, K=50, 49, … , 1 qiymətlərini alır.

FOR C:='A' TO 'E' DO

Burada, C ardıcıl olaraq A, B, C, D, E qiymətlərin alır.

Göründüyü kimi, bu operatorda dövrün parametri ancaq vahid qədər artıb-azala bilər. Bu isə FOR operatorunun çatışmayan cəhətidir. Bu çatışmayan cəhəti REPEAT və WHILE operatorlarının köməyilə aradan qaldırmaq mümkündür.

Qeyd etmək lazımdır ki, dövrün gövdəsində dövrün parametrinin qiymətini dəyişmək olmaz. Dövr qurtardıqdan sonra dövrün parametrinin qiyməti son qiymətə bərabər olur. Dövr sona çatmamış dövrün daxilindən idarəni kənara (GOTO operatoru vasitəsilə) vermək olar.

Misal:

$B = \sum_{i=1}^{n} (i^2 - a) + \prod_{i=1}^{n} (i^2 + a)$ hesablayın.

n=10; a=203.

```
PROGRAM DOVR1;
CONST N=7;A=203;
VAR B,S,P:REAL;
I:INTEGER;
BEGIN
P:=1;S:=0;
WRITELN('N=',N:3,'  A=',A:6);
FOR I:=1 TO N DO BEGIN
P:=P*(SQR(I)+A);
S:=S+SQR(I)-A;
WRITELN('I=',I:3) END;
B:=S+P;
WRITELN('  S=',S:5:2,'  P=',P:5:2,'   B=',B:5:2);
END.
```

II. Son şərtli dövr operatoru başlıqdan **(REPEAT)**, dövrün gövdəsindən və dövrün qurtarmasını müəyyən edən şərtldən (**UNTİL**) ibarətdir.

Operatorun formatı belədir:

REPEAT operatoru təkrarlanmaların sayı məlum olmadan dövrləri qurmaq üçündür.

REPEAT

<operator>;

<operator>;

UNTİL <şərt>;

Burada, <şərt> məntiqi ifadədir. Əvvəlcə REPEAT və UNTİL arasında olan operatorlar yerinə yetirilir, sonra isə dövrün qurtarması şərti yoxlanılır. Əgər məntiqi ifadənin qiyməti false (yalan) olarsa, onda

dövrün gövdəsini təşkil edən operatorlar yenidən yerinə yetirilir. Əgər məntiqi ifadənin qiyməti true (doğru) olarsa, onda dövrdən çıxma baş verir. Dövrün gövdəsini təşkil edən operatorlardan biri elə olmalıdır ki, o dövrün qurtarması şərtinə təsir edə bilsin. Əks halda dövr etmə sonsuz olaraq davam edə bilər.

Misal: Son şərtli dövr operatorundan istifadə edərək h_x addımı ilə dəyişdikdə y-i hesablayan proqram tərtib edək.

$$y = \frac{\sin\frac{3}{\pi} + x}{x + \sqrt{\left|\frac{3\pi}{2} + x\right|}}; x = [x_0; x_n]$$

```
PROGRAM DOVR2;
CONST PI=3/14;
VAR X0,HX,XN,X,Y:REAL;
BEGIN
WRITELN('X0,HX,XN-DAXIL ET:');
READ(X0,HX,XN);
X:=X0;
REPEAT
Y:=SIN(3*PI/2+X)/(X+SQRT(ABS(3*PI/2+X)));
WRITELN('  X=',X:3:2,'   Y=',Y:5:3);
X:=X+HX
UNTIL X>XN;
END.
```

III. Ön şərtli dövr operatoru WHILE – operatoru da REPEAT operatoruna oxşardır. Fərqi odur ki, dövrün qurtarmasını müəyyən edən şərt dövrün gövdəsindən əvvəl gəlir. Operatorun ümumi formatı belədir:

WHILE <şərt> DO <dövrün gövdəsi>;

Burada, <şərt> məntiqi ifadə, <dövrün gövdəsi> isə sadə və mürəkkəb operatorlardır.

Dövrün ifadəsi yerinə yetirilməmişdən əvvəl məntiqi ifadənin qiyməti hesablanır. Əgər nəticə doğru (true) olarsa, onda dövrün gövdəsini təşkil edən operatorlar yerinə yetirilir və yenidən məntiqi ifadənin qiyməti hesablanır. Əks halda dövrdən çıxma baş verir və proqramda WHILE –dən sonra gələn operator yerinə yetirilir. Göründüyü kimi, <şərt >-in qiyməti yalan (false) olarsa, onda dövrün gövdəsini təşkil edən operatorlar bir dəfə də olsun yerinə yetirilmirlər.

REPEAT operatorunda olduğu kimi, WHILE operatorunda dövrün qurtarması şərtini proqramlaşdırıcı müəyyən etməlidir.

Misal: Məsələnin həllini ön şərtli dövr operatorundan istifadə edərək yerinə yetirək.

```
PROGRAM DOVR3;
CONST PI=3/14;
VAR X0,HX,XN,X,Y:REAL;
BEGIN
READ(X0,HX,XN);
X:=X0;
WHILE X<=XN DO
BEGIN
Y:=SIN(3*PI/2+X)/(X+SQRT(ABS(3*PI/2+X)));
WRITELN('  X=',X:3:2,'   Y=',Y:5:3);
X:=X+HX
END
END.
```

Repeat, While, For dövrlərində iki standart Break və Continue prosedurlarından istifadə etmək olar. Break proseduru çıxış şərtini yerinə yetirilməsini gözləmədən dövrdən çıxmağa imkan verir. Continue proseduru isə dövrün əvvəlki iterasiyası sona çatmadan yeni iterasiyanın başlanmasına imkan verir. Prosedurların icrasını aşağıdakı misalda əyani öyrənə bilərik.

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

```
PROGRAM MƏNFI;
USES CRT;  CONST N=4;
A:ARRAY[1..N] OF REAL=(12,-3,6,-7);
 VAR I,K: INTEGER;
BEGIN    CLRSCR;
       FOR I:=1 TO N DO
WRITE(A[I]:7:2,' ':2); WRITELN;
         K:=0;
       FOR I:=1 TO N DO
BEGIN
   IF A[I]>0  THEN CONTINUE;
BEGIN
K:=K+1;
    WRITELN(K:1,'  MENFI ELEMENTI',A[I]:4:2);
     BREAK; END;END;END.
```

Və ya aşağıdakı proqramdan istifadə etmək olar:

```
PROGRAM MASSIV;
USES CRT;
CONST N=4;
A:ARRAY[1..N] OF REAL=(2,3.4,-5,-6);
VAR I:INTEGER;
BEGIN
CLRSCR;
FOR I:=1 TO N DO
IF A[I]>0 THEN CONTINUE ELSE BEGIN
WRITELN('A[',I,']=',A[I]:5:3);
BREAK END;END.
```

12. MASSİVLƏR

Massivlər- bircins, sabit ölçülü, nömrələrinə görə nizamlanmış elementlərdən təşkil olunmuş verilənlərin strukturudur. Massiv adı və ölçüsü ilə təyin olunur. Massivin ayrı-ayrı elementlərinə müraciət massivin ölçüsündən asılı olaraq bir və ya bir neçə indeksin köməyilə mümkündür. İndeks kimi sabit və dəyişən sıra tipindən istifadə etmək olar.

İndekslər – həqiqi növdən başqa istənilən skalyar növə malik ifadələrdir. İndeksin növü onun dəyişmə sərhəddini müəyyən edir. Massivin elementləri isə həmin sadə dəyişənin ixtiyari tipi, həm də dəyişənin mürəkkəb tipi (massiv, sətir, yazı və s.) ola bilər.

Məsələlərin həllində adətən birölçülü, ikiölçülü, üçölçülü massivlərdən istifadə olunur. Praktikada nadir hallarda daha böyük ölçülü massivlərə rast gəlinir.

Massiv ARRAY xidməti sözü vasitəsilə təsvir olunur. Massiv tipini vermək üçün aşağıdakı strikturdan istifadə olunur:

Type

<Tipin adı>:ARRAY[< indekslər>] of <elementlərin tipi>;

Burada < indekslər> hər hansı sıra (nizamlı) tipə məxsus sabit və ya dəyişən, <elementlərin tipi> isə Pascala məxsus hər hansısa tip ola bilər.

Ikiölçülü massivin (düzümün) ümumi yazılış aşağıdakı kimidir:

```
P: ARRAY [ K..N, L..M ] OF T;
```

Burada, P düzümün adı, K, N massivin sətir, L, M isə sütun indeksinin uyğun olaraq aşağı və yuxarı sərhəddini göstərir. T isə massivin elementlərinin tipidir.

İki ölçülü massivin hər hansı bir elementinə müraciət etdikdə massivin adı və kvadrat mötərizə içəricində massivin indeksləri göstərilir. Məs., A[3, 2] yazdıqda massivin 3-çü sətir, 2 sütununda yerləşən elementinə müraciət olunur.

Məsələn, {3 sətir və 2 sütundan təşkil olunmuş tam ədədlərdən ibarət ikiölçülü massiv }

type B=array[1..3, 1..2] of integer;

vasitəsilə təsvir olunur. Bu misalda kvadrat mötərizə əvəzinə nöqtəli dairəvi mötərizədən istifadə etmək olar:

Type B= array(.1..3,1..2.) of integer;

Massiv tipli sabitlərdən istifadə etdikdə massivin elemenləri dairəvi mötərizədə bir-birindən vergüllə ayrılmaqla verilir. Çoxölçülü massivlərdə xarici dairəvi mötərizə sol indeksə, daxili dairəvi mötərizə isə növbəti indeksə və s. aid olur. Belə ki, əgər massiv ikiölçülü isə xarici dairəvi mötərizə sətir indeksinə, daxili dairəvi mötərizə isə sütun indeksinə aid olur.

Məsələn, Birölçülü ədədi massiv

```
TYPE K=ARRAY[1 . . 200] OF CHAR;
VAR  C: K;D:ARRAY[1 . . 60] OF INTEGER;
```

Burada, K, C, D massivlərdir. K və C-nin elementlərinin sayı 200-ə bərabərdir. D massivinin elementlərinin sayı 60-a, Hər bir massivin elementlərinin növü OF işçi sözündən sonra gələn sözlərlə müəyyən olunur.

İkiölçülü ədədi massiv

```
Const
A:array[1..3,1..2] of real=((2.5,-4.5),(78.5,1.7),(6.1,-7.2));
```

Üçölçülü ədədi massiv

```
Const
D:array[1..4,1..3,1..2]=of
Byte=(((1,2),(2,3),(3,4))((1,2),(2,3),(3,4)), (( 1,2),(2,3),(3,4)),
((1,2),(2,3),(3,4)));
```

Pascalda adi massivdən başqa, yığcam (sıxılmış) massiv anlayışı da vardır. Bu məqsədlə massivin array xidməri sözünün əvvəlinə

packed sözünü əlavə etmək lazımdır. Belə massivi yadda saxlamaq üçün tələb olunan yaddaş azalır, massivə müraciət vaxtı isə artır.Yığcam massivin istifadə qaydası adi massivdəki kimidir. Məsələn,

Var D: packed array[1..2,1..3] of real;

Massivin elementlərinin qiymətlərinin əsas yaddaşa daxil olması və çıxışa verilməsi prosesi bir-bir yerinə yetirilir.

Məsələn,

```
FOR İ:=1 TO 4 DO
READLN (A[İ]);          və ya
FOR İ:=1 TO 10 DO
FOR J:=1 TO 10 DO
READLN (D[İ,J]);
```

Massivin elementlərinin çıxışa verilməsi WRITE və WRITELN operatorlarının köməyilə yerinə yetirilir.

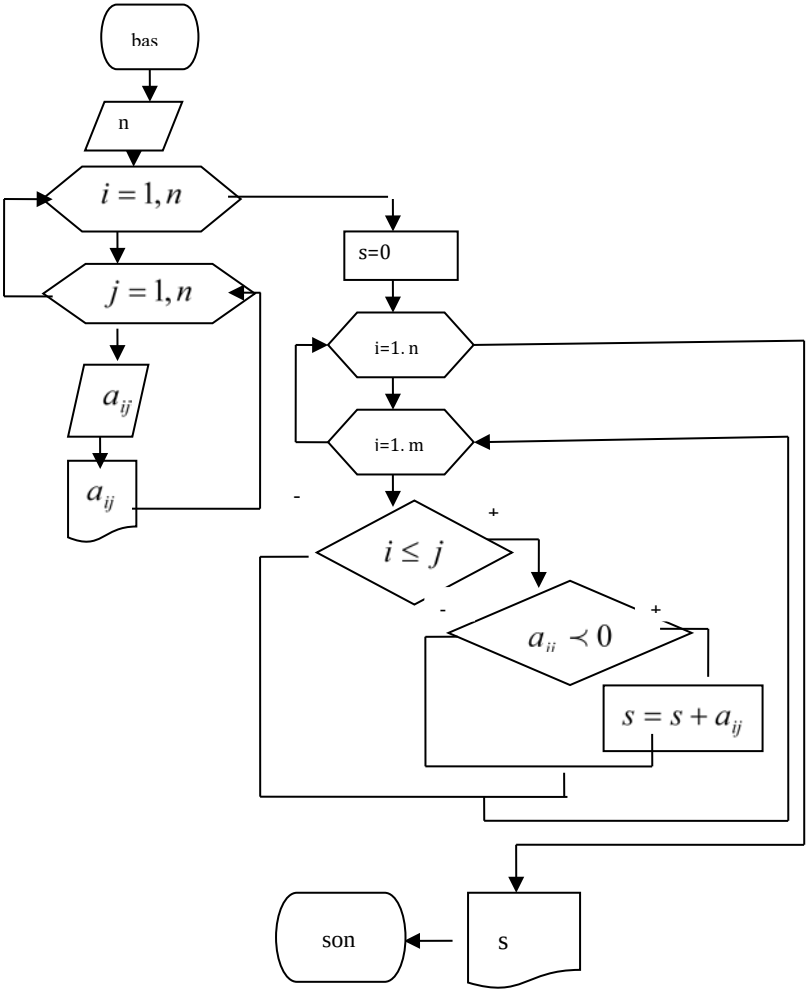
```
FOR İ:=1 TO 4 DO
WRITELN (A[İ]);          və ya
FOR İ:=1 TO 10 DO
FOR J:=1 TO 10 DO
WRITELN (D[İ,J]);
```

A massivində sıfır qiymət alan elementlərin sayını tapmaq üçün

```
K:=0; FOR İ:=1 TO 4 DO
İF A[İ]=0 THEN K:=K+1;
```

əmrlərindən istifadə etmək olar.

Misal: Matrisin baş diaqonalından yuxarıda yerləşən mənfi elementlərin cəmini hesablayan proqram tərtib edin.Blok sxem tərtib edək:



Blok sxemə əsasən proqram tərtib edək:

```
PROGRAM DIOQONAL;
USES CRT;  CONST N=4;
A:ARRAY[1..N,1..N] OF REAL=((-3,5,7,5.7),
(9,-9,-7,4.9),(-2,3,-5,9.7),(3,-4.6,7.-5,3));
 VAR J,I:INTEGER;S:REAL;
    BEGIN    CLRSCR;
WRITELN('ilkin informasiya A matrisi');
FOR I:=1 TO N DO BEGIN
  FOR J:=1 TO N DO
 WRITE(A[I,J]:5:2,' ':2); WRITELN; END;
 WRITELN;   S:=0;
 FOR I:=1 TO N DO
FOR J:=1 TO N DO
IF I<=J THEN BEGIN IF A[I,J]<0 THEN S:=S+A[I,J];
END;WRITELN('S=',S:5:2);
END.
```

13. PROSEDURALAR

Hər hansı bir məqsədə çatmağı reallaşdıran operatorlar qrupunun proqramın bir neçə yerində dəyişməz olaraq təkrarlanması məsələsi praktikada tez-tez qarşıya çıxır. Proqramlaşdırma dillərində alt proqramlardan istifadə etməklə bu məsələni səmərəli şəkildə həll edirlər.

Alt proqramlar –ada malik olan, məntiqi olaraq qurtaran dilin operatorlar qrupundan ibarətdir, onun adına proqramın müxtəlif yerlərində dəfələrlə müraciət etmək olar. Pascal dilində alt proqram təşkil etmək üçün prosedura və funksiyalardan istifadə olunur.

Prosedura – proqramın asılı olmayan və ada malik olan bir hissəsi olub, müəyyən əməliyyatların yerinə yetirilməsinə xidmət edir. Proseduraların adından istifadə etməklə, ona proqramın müxtəlif yerlərindən müraciət etmək olar. Proseduraların adı ifadədə identifikator kimi istifadə edilə bilməz.

Proqramlaşdırıcı tərəfindən təyin olunmuş prosedura və funksiyalarda prosedurlar başlıqlardan və proseduranın gövdəsindən ibarətdir. Proseduranın başlığı **PROCEDURE** sözündən, proseduranın adından və dairəvi mötərizənin içərisində yazılan formal parametrlərdən ibarətdir. Hər bir formal parametrin növü göstərilməlidir. Ümumi formatı aşağıdakı kimidir:

PROCEDURE <adı> {(formal)};

Misal:

PROCEDURE SUM (A:İNTEGER; B: REAL);

PROCEDURE RUX;

Proseduraya müraciət olunan zaman faktiki və formal parametrlər arasında qarşılıqlı birqiymətli əlaqə yaranır.

Faktiki parametrlər vasitəsilə lazım olan dəyişənlərin qiymətləri proseduraya ötürülür və ya prosedurada alınan nəticələr əsas proqrama qaytarılır. Faktiki və formal parametrlərin sayı, növü və yazılma ardıcıllıqları bir-birinə uyğun olmalıdır.

Misal:

Verilmiş C və B matrislərinin maksimum elementlərinin fərqini tapmaq üçün proqram tərtib edin. Matrisin maksimum elementini təyin etmək üçün prosedurdan istifadə edin.

```
PROGRAM MAXMATRIS;
USES CRT;
CONST N=3; L=3;
TYPE TA=ARRAY[1..N,1..L] OF REAL;
VAR A,C,B:TA;
I,J:INTEGER; M,W,V,F:REAL;
PROCEDURE MAX(A:TA;K,KK:INTEGER;VAR M:REAL);
BEGIN
M:=A[1,1];
FOR I:=1 TO K DO
FOR J:=1 TO KK DO
IF M<A[I,J] THEN M:=A[I,J];
END;
BEGIN
CLRSCR;
WRITELN('C MATRISINI DAXIL EDIN');
FOR I:=1 TO N DO
```

```
        FOR J:=1 TO L DO READ(C[I,J]);
WRITELN('B MATRISINI DAXIL EDIN');
        FOR I:=1 TO N DO BEGIN
        FOR J:=1 TO L DO READ(B[I,J]);WRITELN;END;
WRITELN('    C MATRISI');
        FOR I:=1 TO N DO BEGIN
        FOR J:=1 TO L DO WRITE( C[I,J]:5:2,' ':2);WRITELN;END;
        WRITELN; MAX(C,N,L,M);W:=M; WRITELN('MAX_C=',M:6:1);
WRITELN('    B MATRISI');
        FOR I:=1 TO N DO BEGIN
        FOR J:=1 TO L DO WRITE(B[I,J]:5:2,' ':2); WRITELN;END;
        MAX(B,N,L,M); V:=M;
        WRITELN('MAX_B=',M:6:1);F:=W-V; WRITELN('F=',F:5:2);
        READLN; END.
```

Eyni bir proqramda bir neçə prosedura ola bilər. Bu halda onlar bir-birinin daxilində yerləşirlər.

14. FUNKSİYALAR

Proqramlaşdırıcı tərəfindən təyin olunan funksiya başlıqdan və funksiyanın gövdəsindən ibarətdir. Başlıq **Function** sözündən, funksiyanın adından, dairəvi mötərizə içərisində formal parametrlərin siyahısından və funksiyanın qiymətinin növündən ibarət ola bilər. Proseduralarda olduğu kimi funksiyalarda da formal parametrlər zəruri olduğu halda yazılır:

Ümumi formatı aşağıdakı kimidir:

Function <adı>{(formal parametrlər)}:<nətisənin növü>;

Nəticənin növü istənilən skalyar növ və string növü ola bilər.

Funksiyanın gövdəsinin quruluşu aşağıdakı kimidir:

Function<adı>{(formal parametrlər)}: <nətisənin növü>; <dəyişənlərin təsviri forması>

BEGIN

<operator bölməsi>
END;

Funksiyaya müraciət etmə qaydası aşağıdakı kimidir:

<funksiyanın adı>{(faktiki parametrlər)};

Proseduralarda olduğu kimi funksiyalarda da faktiki və formal parametrlərin sayı, növü və yerləşmə ardıcıllığı bir-birinə uyğun olmalıdır. Funksiyanı proseduradan fərqləndirən cəhətlər aşağıdakılardır:

-funksiyanın başlığında onun növü göstərilir. Prosedurlarda bir neçə dəyişənin qiyməti müraciət edən proqrama qaytarıldığı halda funksiyada ancaq bir qiymət – funksiyanın qiyməti müraciət edən proqrama qaytarılır.

Funksiyanın başlığında göstərilən növ funksiyanı hesablamadan alınan qiymətlərinin növünə uyğun olmalıdır.

-funksiyanın gövdəsində ən azı bir mənimsətmə operatoru olmalıdır ki, bu operatorun sol tərəfində funksiyanın adı göstərilsin.

Qeyd etmək lazımdır ki, Turbo Pascal dilində funksiya və ya prosedura özü-özünə də müraciət edə bilər. Bu rekursiya adlanır.

Bir çox hallarda Turbo Pascal dilində proseduranın özü verilməmişdən də ona müraciət etmək mümkündür. Bu halda sadəcə olaraq proseduranın adı verilir və onun sonuna FORWARD sözü də əlavə olunur.

Alt proqramın adından dərhal sonra aşağıdakı standart direktivlərin birinin adını vermək olar: ASSEMBLER, INLINE, FAR, EXTERNAL, INTERRUPT, FORWARD, NEAR.

Bu direktivlər proqramın kompilyasiyasına təsir edir.

Direktivlərin təyinatları aşağıdakı kimidir:

ASSEMBLER–standart maşın instruksiyalarını ləğv edir və alt proqramin realizəsi daxili assemblerlə həyata keçirilir;

EXTERNAL–alt proqram xarici alt proqram kimi élan olunur;

FAR–alt proqram uzaq çağırış üçün kodlaşdırılır (yəni bu alt proqrama əsas proqramın ixtiyari yerindən müraciət etmək olar);

NEAR- alt proqram yaxın çağırış üçün kodlaşdırılır.

(yəni alt proqrama yalnız 64 Kbayt hüdudunda müraciət etmək olar);

FORWARD–kompilyatora məlumat verir ki. alt proqramin elanı sonra veriləcək;

İNLİNE–alt proqramın daxili maşın instruksiyaları ilə realizə (yerinə yetirmə) olunmasını göstərir;

İNTERRUPT–icra zamanı əmələ gələn kəsilmələri emal edir.

15. SƏTİR TİPLİ VERİLƏNLƏR

Pascalda mətn sətirləri ilə işləmək üçün sətir tip nəzərdə tutulmuşdur. Turbo Pascalda sətir tipli verilənlər klaviatuta ilə daxil edilməsi mümkün olan xüsusi simvollar ardıcıllığından əmələ gəlir. Bu simvolların hər biri yaddaşda ASCII kodu ilə 1bayt yer tutur. Simvol və sətir tipli verilənlər konstant və ya dəyişən şəklində verilməklə ‘ (apostrof) işarələri arasında yazılır

‘n’,’2000’, ‘Pascal programlaşma dili’

Sətir tipli dəyişənlər simvol tipli dəyişənlərin bir növüdür. Sətir tipli dəyişənləri təyin edən zaman sətrin maksimal uzunluğunu (simvolun sayını) göstərmək lazımdır. Bu tipə malik olan dəyişən STRING sözünün köməyi ilə təyin olunur.

Tiplərin təsviri bölməsində

Type Tipin adı=String[smu];

Var Siyahı: Tipin adı;

Dəyişənlərin təsviri bölməsində isə

Var Siyahı: String[smu];

şəklində təsvir edilir.

Burada, string –sətir mənasına uygun gəlir;

Tipin adı – sətir tipli sabit dəyişən və ya ifadə;

Siyahı – bir-birindən vergüllə ayrılmış sətir tipli dəyişənlər;

Smu (sətrin maksimal uzunluğu və ya indeksi) isə sətri sabitin simvollarının sayını (uzunluğunu) bildirən və kvadrat mötərizələri daxilində yazılan və 0,1,...,255 qiymətləri alan tam tipli sabit, dəyişən və ya ifadə ola bilər. Əgər sətir elan olunarkən sətirdəki simvolların sayı göstərilməyibsə, bu sətirdəki simvolların maksimal sayı sistem tərəfindən 256 qəbul olunur.

Məsələn , 4 simvoldan ibarət N dəyişənini:

1. Tiplərin təsviri

Type S=String[4];
Var N:S;
2. Dəyişənlərin təsviri
var siyahi: string[14];b: string[155];
şəklində təsvir etmək olar.

Sətir tipli dəyişənlərin yaddaşda tutduqları yer onların uzunluğundan 1 bayt artıqdır. Sətrə aid olan simvollar 1-dən başlayaraq sətrin uzunluğunu göstərən ədədə qədər indeksləşirlər. Sətrin hər bir simvoluna müraciət etmək üçün bu indekslərdən istifadə olunur. Sətir və simvol tipli dəyişənlər eyni bir ifadədə iştirak edə bilərlər.

Sətir simvollardan ibarət bir ölçülü xüsusi növlü massivdir. Sətirlər təsvir və uzunluqlarının göstərilmə üsullarına görə iki tip sətirə String –sətrinə və Turbo –sətrinə ayrılırlırlar.

Turbo sətrinə sıfır sonlu sətir, bəzən simvollar massivi də deyilir.

Turbo – sətir tipli dəyişən

var <sətir_dəyişəninin_adı>: array [0..n] of char;

şəklində təsvir olunur.

Sətir tipli verilənlər iki apastrof işarəsi arasında simvollar ardıcıllığı şəklində gösrətilir.

Məsələn,

S2:=’Turbo Pascal’.

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

15.1. Concat funksiyası

Pascalda bir neçə sətir tipli veriləni ardıcıl birləşdirmək üçün

CONCAT (S1, S2,... ,SN: STRING): STRING;

standart sətri funksiyasından istifadə edilir.

Burada, concat (CONCATenation)-birləşdirmək, bağlamaq və ya əlaqələndirmək mənasına uyğun gəlir.

s1, s2,... ,sn sətir tipli verilənlər;

s1, s2,... ,sn simvolların birgə sayı ən çoxu 256 ola bilər.

Misal: Azərbaycan Dövlət, Aqrar Universiteti ifadələrinin birləşdirilməsi üçün proqram tərtib edin.

PROGRAM BIRLƏŞMƏ;

```
VAR MT1, MT2,ST: STRING;
BEGIN
 MT1:= ’AZƏRBAYCAN DÖVLƏT ’;
MT2:= ’AQRAR UNIVERSITETI’;
ST:=CONCAT(MT1,MT2); WRITELN(‘ST=’;ST) ;END
```

Nəticə: Azərbaycan Dövlət Aqrar Universiteti olacaqdır.

15.2. Ord və Chr funksiyaları

Pascalda ASCII kodu adlanan bir ədəd qarşı qoyulur. Məsələn, latın əlifbasının B hərfinə 66, b hərfinə isə 98 kodu uyğun gəlir. Pascalın əlifbasının müxtəlif simvollarına müxtəlif kodların uyğun olması simvollar üzərində müəyyən əməliyyatları yerinə yetirməyə imkan verir.

-Ord funksiyası

Pascalda simvola uyğun kodu təyin etmək üçün

Ord(x:char):tip;

standart funksiyasından istifadə edilir.

Burada, ord (ORDinal)-nizamlı sıra sayı mənasına uyğun gəlir. X-dəyişəni ASCII cədvəlində verilə ixtiyari simvolu, tip isə həqiqi tipdən başqa, istənilən nizamlı tipi bildirir.

Misal1. ‘c’ simvolunun kodunu tapmaq üçün proqram tərtib etməli.

```
(*Simvolun kodunun tapılması*)
VAR X:CHAR;
BEGIN
X:=’C’;
WRITELN(ORD(X));
END.
```

Proqram icra edildikdə c simvolunun kodu (99 ədədi) çap edilir.

-chr funksiyası

Kodu məlum olan simvolu yazmaq üçün chr funksiyasından istifadə edilir. Bu funksiya ord funksiyasının əksinə olan əməliyyatı yerinə yetirir. Yəni bu funksiyanın arqumenti tam ədəd, qiyməti isə sıra nömrəsi bu tam ədədə uyğun olan simvol olur. Bu məqsədlə

```
CHR (X:BYTE):CHAR;
```

standart sətri funksiyasından istifadə olunur.

Burada, Chr(ChrRacter)-simvol mənasını verir.

Misal. Kodu 86 olan simvolu tapmaq üçün proqram tərtib etməli.

```
(*Simvolun tapılması*)
 X=86;Writeln(chr(x));
```

Proqram icra edildikdə ekranda V hərfi (simvolu) görünür.

Misal. Kodu məlum olan bütün simvolları çap etmək üçün proqram tərtib etməli.

```
PROGRAM KODA_UYĞUN_SIMVOLUN_TAPILMASI;
VAR N:BYTE;
BEGIN
FOR N:=0 TO 255 DO;
WRITELN('CHR(',N,')=',CHR(N));END.
```

Proqram icra edildikdə, ekranda kodlara uyğun simvollar çap olunur.

15.3. Pred və Succ funksiyaları

Pascalda ASCII cədvəlinin ixtiyari simvolundan əvvəl və ya sonra gələn simvolunu müəyyən etmək üçün aşağıdakı standart funksiyalardan istifadə edilir.

```
PRED (X:CHAR):tip;
 SUCC(X:CHAR):tip;
```

Burada, pred-özündən əvvəl gələn ;

Succ--özündən sonra gələn mənasını;

x-simvol tipli dəyişən, tip isə həqiqi tipdən başqa, istənilən nizamlı tipi bildirir.

Misal. İxtiyari simvoldan əvvəl və sonra gələn simvolu çap edin.

```
PROGRAM SONRA_ GƏLƏN_ SIMVOLUN_TƏYINI;
VAR X:CHAR;
BEGIN
READLN(X);
WRITELN(SUCC(X));
END.
```

Proqramda Succ funksiyasını Pred funksiyası ilə əvəz etdikdə simvoldan əvvəlki simvol təyin edilir.

15.4. Length funksiyası

Pascalda sətri sabiti əmələ gətirən görünən (hərf rəqəm və s.) və görünməyən (probel) simvolların birgə sayını tapmaq üçün

```
LENGTH(VAR S:STRING): INTEGER;
```

və ya

```
N: LENGTH(VAR S:STRING): INTEGER;
```

standart sətri funksiyasından istifadə edilir.

Burada, length-uzunluq mənasını; s sətir tipli ifadəni; n dəyişəni s sətir tipli ifadəsində görünən və görünməyən simvolların birgə sayını bildirir və 0;1;... ;256 qiymətlərini ala bilər.

Misal.:'İnformatika'sətri sabitinin uzunluğunu tapan proqram yazın.

```
PROGRAM SƏTIR;
VAR X: STRING;N:BYTE;
BEGIN
X:= 'İNFORMATIKA' ;
N: = STRING(X);
WRITELN('N=';N) ;
END.
```

Nəticə:n=11 olacaqdır.

15.5. Pos funksiyası

Pascalda s1 sətri sabitinin s2 sətri sabitində ilk dəfə rast gəldiyi pozisiyasını müəyyən etmək üçün

```
POS (S1, S2: STRING):BYTE;
```

Standart sətri funksiyasından istifadə edilir.

Burada pos (POSition)-pozisiya mövqe mənasını;

s1və s2 sətr tipli sabit, dəyişən və ya ifadəni;

Byte-nəticənin byte tipli olmasını bildirir.

Misal: 'Azərbaycan Dövlət Aqrar Universiteti' sətri sabitində ilk dəfə rast gələn 'r' simvolunun yerini təyin edən proqram tərtib edin.

```
PROGRAM Simvolun_pozisiyasının_təyini;
VAR S1,S2: STRING;
S3: BYTE;
BEGIN
S1:='R';
S2:= 'AZƏRBAYCAN DÖVLƏT AQRAR UNIVERSITETI';
S3:=POS(S1,S2) ;
```

```
WRITELN('S3=';S3) ;END.
```

Bu proqram s1 sətri sabitini s2–nin əvvəlindən başlayaraq axtarır və onun ilk dəfə rast gəlindiyi pozisiyanın nömrəsini tapır:s3=4. Proqramda r simvolu əvəzinə ət sətri sabiti daxil edildikdə s3=16 olduğu alınar.

15.6. Copy funksiyası

Sətri verilənin hər hansı simvolundan başlayaraq ixtiyari sayda simvolunun surətini almaq üçün aşağidakı şəkildə yazılan **Copy** standart funksiyasından istifadə olunur:

```
COPY (ST: STRING;ISM, SAY: INTEGER): STRING;
```

Burada, COPY-surətini almaq mənasını ;

St- sətri sabit, dəyişən və ya ifadəni;

Ism-surəti alınacaq sətri sabitin ilk simvolunun mövqeyini;

Say- surəti alınan sətri sabitin ilk simvollarının sayını bildirən tam tipli sabit, dəyişən və ya ifadəni bildirir.

Misal: 'Kompüter texnologiyaları və proqramlaşdırma fənni' sətri sabitində proqram sözünü ayırıb çap edən proqram tərtib edin. Proqram sətri sabitinin ilk simvolunun mövqeyinin 29 və simvollarının sayının 7 olduğunu bilərək proqramı belə yazmaq olar:

```
(* Copy-nin tətbiqinə aid*)
PROGRAM köçürülmə;
VAR ST, AYS: STRING; ISM, SAY: INTEGER;
BEGIN
ST:='KOMPÜTER TEXNOLOGIYALARI VƏ PROQRAMLAŞDIRMA
FƏNNI ';
ISM:=29;SAY:=7;
AYS:=COPY(ST, ISM, SAY) ;
WRITELN(AYS) ;END.
```

15.7. UpCase funksiyası

Pascalda sözlərin yazılışında istifadə olunan ilk kiçik hərfi uyğun böyük hərflə əvəz etmək üçün aşağıdakı şəkildə yazılan UpCase standart funksiyasından istifadə olunur:

```
UPCASE(CH:CHAR) :CHAR
```

Burada , upCase – yuxarı variant(registr) mənasına , ch isə kiçik hərfi bildirən simvol tipli dəyişənə uyğun gəlir.

Misal: Aşağıda verilən proqram istənilən kiçik şrifti böyük şriftlə əvəz edir:

```
(*Kiçik şrifti böyük şriftlə əvəz etmə*)
PROGRAM ƏVƏZ_ ETMƏ;
VAR CH: CHAR;
BEGIN
READLN(CH);WRITELN(CH, UPCASE(CH)) ;END.
```

15.8. Str və Val prosedurları

Str funksiyası. Pascalda ədədi sabitləri uyğun sətri sabitlərlə əvəz etmək üçün

```
STR(ƏS:ƏSSS:ƏSOIS:INTEGER; SS: STRING) ;  VƏ YA
STR(ƏS:ƏSSS:ƏSOIS, SS) ;
```

şəklində yazılan standart funksiyadan istifadə olunur.

Burada, str(STRing)-sətir mənasını verir;

Əs-tam və ya həqiqi tipli ədədi sabit, dəyişən və ya ifadəni;

Əsss- tam tipli ifadə olub ədədi sabitin simvollarının sayını;

Əsois- tam tipli ifadə olub ədədi sabitin onluq işarələrinin sayını;

Ss isə alınan sətri sabit, dəyişən və ya ifadəni bildirir.

Misal: 17 ədədi sabitini sətri sabitə çevirən proqram tərtib edin. Tam tipli konstantları sətir tipli sabitlərə çevirmək üçün funksiyanın str(Əs,Ss) yazılışından istifadə etməklə proqramı tərtib edək:

```
(* Ədədi sabitin sətri sabitə çevrilməsi*)
PROGRAM ÇEVIRMƏ1;
VAR ƏS: INTEGER;SS: STRING;
BEGIN
ƏS:=17;STR(ƏS,SS);
WRITELN('SS=', SS) ;END.
```

Proqram yerinə yetirildikdə Ss=‘17’ olması alınır.

Val funksiyası. Pascalda ədədlərdən simvol və ya sətri sabit kimi istifadə edilir. Bu zaman ədəd apostrof işarələri arasında yazılır. Məsələn, M:= ‘2011’ olduqda bu say bildirən ədəd kimi yox, 4 simvoldan ibarət sətri sabit (mətn) kimi başa düşülür.

Rəqəmlər ardıcıllığından ibarət sətir tipli sabiti ədədi sabitlərlə əvəz etmək üçün Val standart funksiyadan istifadə olunur:

VAL(SS: STRING; ƏS:INTEGER[REAL];SP:INTEGER); VƏ YA ƏS:=VAL(,ƏS,SP) ;

Burada, val(Value)-qiymət mənasını verir;

Ss-rəqəmlərdən ibarət sətir tipli sabit;

Əs-nəticədə alınan tam və ya həqiqi tipli ədədi sabit;

Sp isə Ss sətri sabitində ilk dəfə rast gəlinən rəqəm olmayan simvolun pozisiyasını bildirən tam ədəddir.

Misal: Verilmiş '2011' sətri sabitini ədədi sabitə çevirən proqram tərtib edin.

```
(*Sətri sabitin ədədi sabitə çevrilməsi*)
PROGRAM ÇEVIRMƏ2;
VAR SS: STRING; ƏS,SP INTEGER;
BEGIN
SS:= '2011';
VAL(SS, ƏS,SP);
WRITELN('ƏS=', ƏS, 'SP=', SP) ;END.
```

Proqram yerinə yetirildikdə Əs=2011 və Sp=0 olması alınır.

15.9. Delete və İnsert prosedurları

Sətir tipli sabitin müəyyən hissəsini silmək və ya ona müəyyən əlavələr etmək üşün Delete və İnsert prosedurlarından istifadə olunur.

Delete proseduru.

Pascalda verilmiş sətrin (mətnin) hər hansı pozisiyasından başlayaraq müəyyən uzunluğa malik hissəsini silmək üçün

DELETE (ST:STRING, İNDEX, COUNT: INTEGER);

standart prosedurundan istifadə edilir.

Burada, Delete –silmək mənasını;

St-sətri sabit, dəyişən və ya ifadəni;

İndex-silinəcək sətri sabitin ilk simvolunun nömrəsini;

Count-silinəcək sətri sabitin simvollarının sayını bildirir.

Misal: 'İnformatika fənni'sətri sabitinin ilk 11 simvolunu silmək üçün proqram yazın.

```
(*Mətnin nüəyyən hissəsinin silinməsi*)
PROGRAM SILMƏK;
VAR ST: STRING; İNDEX, COUNT: INTEGER;
BEGIN
ST:= 'İNFORMATIKA FƏNNI' ;
```

İNDEX: = 1; COUNT: = 7;
DELETE (ST,İNDEX,COUNT);
WRITELN('N=';N) ;END.

Proqram icra edildikdə, onun ilk 11 simvolundan ibarət İnformatika sözü silinir və ekranda fənni sətri sabiti görünür.

İnsert proseduru

Pascalda s1 sətri sabitini s2 sətri sabitinə onun (s2-nin) s3 pozisiyasından başlayaraq əlavə etmək üçün açağıdakı şəkildə yazılan İnsert standart prosedurundan istifadə edilir.

İNSERT (S1, S2: STRING; S3:BYTE);

Burada, İnsert –yerləşdirmək, əlavə etmək mənasını;

S1 və s2 -sətir tipli sabit, dəyişən və ya ifadələri;

S3- byte tipli sabit olub s1-in s2-də yerləşdiriləcəyi ilk pozisiyanın nömrəsini təyin edən natural ədədi bildirir;

Misal: s1 sətri sabitini s2 sətri sabitinin ixtiyari pozisiyasından başlayaraq əlavə etmək üçün proqram yazın.

```
(*Mətnə müəyyən hissənin əlavə edilməsi*)
PROGRAM ƏLAVƏ_ETMƏK;
VAR S1, S2: STRING; S3:BYTE;
BEGIN
READLN(S1, S2);READLN(S3);
İNSERT (S1, S2, S3);
WRITELN('S2=';S2) ;END.
```

16. Çoxluqlar

Çoxluq verilənlərin strukturlaşmış növü olub, müəyyən əlamətə görə qarşılıqlı əlaqəli olan obyektlər yığımıdır. Hər bir obyekt çoxluğun elementi adlanır. Çoxlugun hər bir elementi həqiqi növdən başqa eyni bir skalyar növə malik olmalıdır. Bu skalyar növ çoxluğun baza növü adlanır. Çoxluq növləri dəyişənin qiymətlər oblastı baza növlü elementlərdən təşkil olunmuş bütün mümkün alt çoxluqlar yığımıdır. Əgər baza növlü n qiymət alarsa, onda çoxluq növlü dəyişən 2^n sayda müxtəlif qiymətlər alar. İfadədə çoxluğun elementləri kvadrat və ya dairəvi mötərizə ilə göstərilir:

$[1, 2, 3, 4]$, $['a', 'ə', 'c']$, $['a',...'z']$

Boş çoxluq [ ] işarə olunur. Çoxluq növlü dəyişənləri təsvir

etmək üçün **SET**(çoxluq) **OF** (-dan, dən) işçi sözlərindən istifadə olunur. Çoxluq tipli dəyişən:

1. tiplər bölməsində:

TYPE <tipin adı>=SET OF <baza tipi>;

VAR <çoxluq tipli dəyişənin adı>: <tipin adı >;

2. dəyişənlər (var) bölməsində:

VAR<çoxluq tipli dəyişənin adı>: Set Of <baza tipi>;

Şəklində təsvir edilir.

Burada, <tipin adı> -çoxluq tipinin adını;

Set-çoxluq mənasını;

Of-aidolmanı;

<baza tipi>-çoxluğun elementlərinin tipini;

<çoxluq tipli dəyişənin adı>-çoxluğun təsvir olunduğu adı(identifikatoru) bildirir;

Misal:

```
VAR y1,y2:set of  char;
Begin
Y1:=[‘a’,’d’]; y2:=[‘c’,’d’];
```

Misal:

```
Type
P= set of (3,5,7,9,11,13); Nomer= set of 1..31;
Var   Pr:p; N:Nomer; B:set of (‘a’,’b’,’c’,’d’)
```

Bu misalda Pr dəyişəni 3,5,7,9,11,13 qiymətlərini, N dəyişəni 1-dən 31-ə qədər olan tam ədədləri, B isə ‘a’,’b’,’c’,’d’ qiymətlərini ala bilər. Bu dəyişənlərə göstərilən qiymətlərdən əlavə qiymətlərin mənimsədilməsi proqramın kəsilmə halına səbəb olur. Çoxluğun elementlərinin sayı 256-dan çox olmamalıdır.

Çoxluğun növünə aid olan dəyişənlər üzərində aşağıdakı əməliyyatları yerinə yetirmək mümkündür.

-müqayisə əməlləri:=,<>,>=,<=

-birləşdirmə; kəsişmə, çoxluqların fərqi, in əməli.

Əgər A və B çoxluqları eyni elementlərdən ibarət olarsa, onda bu çoxluqlar bərabərdir(A=B). Əks halda bu çoxluqlar bir-birinə bərabər deyildir(A<>B).

A>=B əməlinin nəticəsi o vaxt true olur ki, B-nin bütün

elementləri A çoxluğuna daxil olsun. Əks halda A>=B-nin nəticəsi false olur. A<=B əməlinin nəticəsi o vaxt true qiymətini alır ki, A çoxluğunun bütün elementləri B-yə daxil olsun. Əks halda A<=B əməlinin nəticəsi false olur.

[1,3] = [3,1] TRUE [1,2] < = [1,2,3] TRUE

[2,3] = [1] FALSE

A və B çoxluqlarının birləşməsi (A+B) bu çoxluqlarının elementlərindən ibarət olan 3-cü bir çoxluqdur: [1,2]+ [1,3] =[1,2,3]

A və B çoxluqlarının kəsişməsi (A* B) eyni zamanda hər iki çoxluğa daxil olan elementlərdən ibarət olan 3-cü bir çoxluqdur:

[1,2] * [1,3] = [1]

A və B çoxluqlarının fərqi (A-B) birinci çoxluğun (A) ikinci çoxluğa (B-yə) daxil olmayan elementlərindən ibarət 3-cü bir çoxluqdur: [1,2,3] - [1,2] = [3]

In (include)-daxilolma əməlindən hər hansı elementin verilən çoxluğa daxil olmasını yoxlamaq üçün istifadə olunur. Adətən, şərti keçid operatorlarından istifadə olunur.

Proqramda çoxluqlardan istifadə olunması nəticəsində if mürəkkəb operatoru və məsələnin həlli alqoritmi sadələşir, müəyyən hallarda yaddaşa qənaət edilir, komplasiya və yerinə yetirilmə vaxtı azalır.

Çoxluq növlü dəyişənlərdən istifadə olunmasının çatışmayan cəhəti odur ki, onlara giriş və çıxış operatorlarını tətbiq etmək olmaz.

in əməli və if operatoru vasitəsilə olan mürəkkəb yoxlamaların yazılışını sadələşdirir.

Məsələn, if(a=1) or (a=2) or (a=3) or (a=4) əvəzinə if a in [1..5] yazmaq olar.

Çoxluqların bir növü də sabit çoxluqlardır. Sabit çoxluqlar bir-birindən vergüllə ayrılan və kvadrat mötərizənin arasında yerləşən elementlərdən ibarətdir.

Misal1:

A:=[‘f’,’ə’,’r’,’i’,’d’] çoxluğunun F:=[‘f’,’ə’,’x’,’r’,’d’ ,’d’,’i’,’n’]-in altçoxluğu olub-olmadığını yoxlayan proqram tərtib edin. A və F çoxluqlarını simvol tipli təyin edib, proqramı belə yazaq:

```
PROGRAM ALTÇOXLUĞUN_TƏYIN_EDILMƏSI;
VAR A, F :SET OF CHAR;
```

```
BEGIN
A:=['F','Ə','R','I','D'] ;
F:=[ 'F','Ə','X','R','D' ,'D','I','N'];
İF A<=F THEN WRITELN('A çoxluğu f-in alt çoxluğudur.') ELSE ('A çoxluğu F-in alt çoxluğu deyil.')
 END.
```

Proqram kompilyasiya edilib icra olunan kimi nəticə görinür:

A çoxluğu F-in alt çoxluğudur.

Misal2:

'e' elementinin S:=['f','e','v','i','n','c'] çoxluğuna daxil olub-olmadığını təyin etmək üçün proqramı aşağıdakı kimi yaza bilərik:

```
PROGRAM Elementin_çoxluğa_daxil_olması;
VAR S:SET OF CHAR;SIM:CHAR;
BEGIN
S:=['F','E','V','I','N','C'];
SIM:='E';
İF SIM IN S THEN WRITELN('E SIMVOLU S-Ə DAXILDIR.') ELSE WRITELN('E SIMVOLU S-Ə DAXIL DEYIL.') ;
 END.
```

Proqramda Sim:='e' mənsubetmə operatorunu readln(Sim) giriş operatoru ilə əvəz etdikdə programın ümumi şəkli alınır.

17.YAZILAR

Yazılar -verilənlərin stukturlaşmış növü olub, qeyd olunmuş sayda müxtəlif növlü elementlərdən ibarətdir. Massivlərdən fərqli olaraq yazılarda eyni bir obyektə aid olan müxtəlif növlü verilənlər qruplaşır.

Yazıların təyin olunması **RECORD (yazı)** sözü ilə başlayıb **END (son)** sözü ilə qurtarır. Bunların arasında yazının elementlərinin siyahısı gəlir. Elementlərin siyahısına yazının sahələri də deyilir. Hər bir sahə identifikatorlardan və onun növündən ibarətdir. Yazıları, dəyişənlərin elan edildiyi bölmədə və ya **TYPE** tipli bölmədən istifadə edərək elan etmək olar. Bu zaman yazının tipi aşağdakı kimi elan olunur:

```
   TYPE
<növün adı>=RECORD
<sahənin identifikatoru,…>:<elementin növü>;
……
```

<sahənin identifikatoru,…>:<elementin növü>;

END;

VAR <identifikator,…>:<növün adı>;

Burada < növün adı> –düzgün identifikatordur. <sahələrin identifikatoru >-bir-birindən nöqtəli vergüllə ayrılmış yazılar bölmələri ardıcıllığıdır .

Dəyişənlər bölməsində yazıların tipi aşağıdakı kimi elan olunur:

VAR <yazının adı>: RECORD

1 Elementin adı: tipi;

2 Elementin adı: tipi;

3 ----------

N elementin adı: tipi;

END;

Məsələn,

```
VAR B: RECORD
N: INTEGER;
FAM: PACKED ARRAY[1..20] OF CHAR;
BAL: ARRAY[1..3] OF INTEGER
END;
```

Hər bir yazılar bölməsi bir-birindən vergüllə ayrılmış identifikatorlardan ibarət olur . İdentifikatorun hər birindən sonra ":"qoyularaq onun tipi göstərilir .

Məsələn,

```
TYPE
TALABA=RECORD
AD, SOYAD : STRING;
TAVALLÜD : WORD;
END;
VAR  A,B,C: TALABA;
```

Bu nümünədə TALABA yazısının üç sahəsi var: ad, soyad, tavallüd . A, B, C dəyişənləri TALABA tipli yazılardır. Massivlərdə olduğu kimi yazılarda da eyni tipli yazıları bir-birinə mənimsətmək mümkündür. Məsələn,

A:=C;

B:=C;

Yazının hər bir komponentinə ayrılıqdada müracət etmək olar . Bu halda əvvəlcə RECORD tipli dəyişənin adını göstərir, nöqtə qoyulur, sonra sahənin adı verilir .

Məsələn ,

```
B.TAVALLÜD:=1985;
A.AD:=’ARİF’;
C.SOYAD :=’ADİLOV’;
```

17.1. Birləşdirmə operatoru WITH.

Yazının sahələri ilə işləməni sadələşdirmək, eyni bir yazının sahələrinə bir neçə dəfə müraciət etmək məqsədilə birləşdirmə operatorundan istifadə olunur. Birləşdirmə operatorunun ümumi yazılışı aşağıdakı kimidir :

```
WITH  U DO S;
```

Burada, U- yazının adı, S – operatordur.

WITH operatorunda yazı növlü dəyişənin adını bir dəfə göstərməklə onun sahələrinin adları ilə adi dəyişən kimi işlətmək olar. Bu zaman sahənin adından əvvəl ondan nöqtə ilə ayrılan dəyişənin adını göstərmək olar.

Misal :

```
WITH  M DO
BEGIN
N:=2545;MM:=’ADAU’;NAME:=’RZAYEV S.E’;
ADRES:=’CAVAD XAN ‘;
END;
```

MISAL :

```
WİTH B DO
BEGİN
N:=2;
S:=QIYMƏT[1]+ QIYMƏT [2]+ QIYMƏT [3];
READ (N)
END;
```

Turbo-Pascalda ciddi qeyd olunmuş verilənlərdən ibarət sahələrə malik yazılarla bərabər variantlı sahələrə malik yazılardan da istifadə etmək mümkündür. Variantlar CASE..OF sözlərinin köməyi ilə verilir. CASE sözlərdən sonra seçmə sabiti göstərilir .OF sözündən sonra isə

variantlar və onlara uyğun sahələr verilir (“ : ”-dən sonra). Sahələr adi mötərzələr daxilində göstərilir. Məsələn,

```
TYPE
MALUMAT=RECORD
CASE BOOLEAN OF
TRUE:(KUCA :STRINC[30];EVN:INTECER);
FALSE:(BOLGA:STRINC  [35];
RAYON:STRINC [13];
KAND : STRINC [20];END;
```

Burada istifadə olunan CASE..OF açar sözləri ilə seçmə operatorlardakı sözlərin fərqi ondadır ki, CASE..OF –a uyğun olan END verilmir. Nümunədəki END RECORD-a uyğun olan END-dir (Yazıda variant hissəsi həmişə sonuncu olur, buna görə də END opiratorunun verilməsinə ehtiyac olumur).

Turbo-Pascalda seçmə açarı sahəsində sadalanan tipli dəyişəni elan etmək də olar və hətta ona qiymət də mənimsətməyə icazə veriir. Lakin bütün bunlar sahənin seçilməsinə təsir göstərmir. Seçmə sabitinin qiyməti Turbo-Pascalda ixtiyari və hətta təkrarlanan da ola bilər .

Qeyd edək ki, yazılarda istifadə olunan sahələrin adları təkrarlanmamalıdır. Əgər yazı bir-birinə daxil olan yazı sahələrinə malikdirsə, onda müxtəlif səvyyələrdə olan sahələrin adları eyni bilər. Belə halda proqramın tərtibi və istifadəsində daha diqqətli olmaq tələb olunur.

Misal: tələbələrin siyahısını daxil və xaric etmək üçün proqram tərtib edək:

```
    PROGRAM TƏLƏBƏ;
TYPE MAS = ARRAY [ 1..25] of CHAR;
VAR SIYAHI: ARRAY [ 1...25] of MAS;     I,J : INTEGER;
BEGIN
FOR I: = 1 TO 25 DO
BEGIN
FOR J: = 1 TD 25 DO
READ (SIYAHI [I, J] );READLN; END;
FOR I: = 1 TO 25 DO
WRITELN (SIYAHI [I] );END.
```

Misal: Tutaq ki, avtomobilin sahibi haqqında olan verilənləri (avtomobilin nömrəsi, növü, sahibinin adı, və familyası, ünvanı) bir yazı kimi təyin etmək tələb olunur.

```
TYPE
MASH=RECORD
NOMER:INTEGER (*NÖMRƏSI*)
MARKA:STRING[20]  (*AVTOMOBILIN NÖVÜ*)
NAME: STRING[40]    (*SAHIBININ ADI VƏ FAMILYASI*)
ADRES: STRING[60]   (*SAHIBIN ÜNVANI*)
END;
VAR     M, V: MASH;
```

Bu misalda Mash yazılışının dörd elementi vardır: Nomer, Marka, Name, Adres. M, V-də Mash növlü yazılardır. Onların da dörd elementi vardır.

Yazılar üçün tələb olunan yaddaşın həcmini sahələrin uzunluqlarını toplamaqla müəyyən etmək olar. Yazının sahələrinin qiyməti ifadələrdə istifadə etmək üçün yazının adı və sahənin identifikatoru bir-birindən nöqtə ilə ayrılmış şəkildə yazılmalıdır. Məsələn, Mash yazısının sahələrinə aşağıdakı kimi mürasiət yazmaq olar:

M.Nomer, M.Marka, M .Name, M.Adres və ya

V.Nomer, V.Marka, V.Name, V.Adres

Belə adlara mürəkkəb adlar deyilir. Yazının sahələrinə qiymətlər vermək üçün mənimsətmə operatorundan istifadə etmək lazımdır.

Misal:

```
M.Nomer:=4880;
M.Marka:=’QAZ-24’;
M.Name:=’Məmmədov A.Q.’;
M.Adres:=’Nizami küç. ’;
```

Mürəkkəb adları giriş və çıxış operatorlarından da istifadə etmək olar:

```
Read(M.Nomer, M.Marka, M.Name, M.Adres);
Write (M.Nomer, M.Marka, M.Name, M.Adres);
```

Əgər iki yazı eyni növə malik olarsa, onda onların sahələrinin qiymətlərini bir-birinə mənimsətmək olar. Məsələn,V:=M; Əğər tək bir sahənin məzmununu dəyişmək tələb olunarsa, onda

```
V.Adres =M.Adres
```

yazmaq kifayətdir. Bir sıra hallarda elementləri yazılar olan massivlərdən istifadə etmək əlverişli olur. Onları aşağıdakı kimi təsvir edirlər.

```
TYPE    PRIM:=RECORD;
F10:STRING[20]; A:0..90;
PROF: STRING[30]; END;
VAR LIST ARRAY[1..50] OF PRIM;
```

Burada, List massivinin hər bir elementi Prim növünə malik yazıdır.

18.FAYLLAR

Optimal planlaşdırma, hesabat-uçot xarakterli və s. məsələlərin böyük həcmə malik olan informasiyaların klaviaturadan daxil edilməsi çox əziyyətli olur və xeyli vaxt itkisinə səbəb olur. Belə məsələlər üçün proqramlaşdırma dillərində fayllardan istifadə olunur. Fayların köməyilə klaviaturadan daxil edilən informasiya xarici yaddaşda saxlanılır, proqramm vasitəsilə həmin informasiyadan istifadə etmək olar.

Pascalda ən geniş yayılan fayllar standart fayylar və proqramlaşdırıcı yaratdığı fayllardır. Standart faylları sistemi hazırlayanlar müəyyən edir və ondan proqramlaşdırıcı ancaq istifadə edə bilər.

Proqramlaşdırıcının faylları çevik və ya vinçestr disklərdə yerləşir. Bu fayllarda müxtəlif proqramlar, verilənlər,sənədlərin mətni, qaydalar saxlanıla bilər. Hər bir fayl giriş və ya çıxış faylı ola bilər. Proqram giriş faylından verilənləri alır, çıxış faylına isə verir. Hər bir fayl ada malik olur. Faylın adından istifadə etməklə onu dəyişdirmək, genişləndirmək, ekrana və ya çapa vermək və digər mümkün əməliyyatları yerinə yetirmək olar. Pascalda fayl-eyni növə və uzunluğa malik olan elementlər ardıcıllığıdır. Bu zaman faylın elementlər ardıcıllığıdır. Bu zaman faylın elementləri yazı adlanır.

Pascalda fayllarla işləməyi effektiv və əlverişli təşkil etmək üçün bir sıra prosedura və funksiyalardan istifadə edilir. Aşağıdakı işarələmələri qəbul edək:

-fayl tipli dəyişəni FV ilə; sətir simvolları STR ilə; elementlərinin növü FV dəyişəninin elementlərinin növü ilə eyni olan dəyişənləri P ilə; tam ifadəni isə n ilə işarə edək.

Fayllar xarici yaddaşda adlandırılmış hər hansı bir sahədir. Bu faylın fiziki mövcudluğudur ki, bu da fiziki fayl adlanır.

Digər tərəfdən fayl proqramlaşdırmada istifadə olunan çoxlu sayda verilənlərin strukturundan biridir. Bu halda məntiqi fayl terminindən istifadə olunur. Yəni proqram tərtibində faylın mövcudluğu haqqında bizdə yalnız məntiqi təsəvvür var. Proqramlarda məntiqi fayllar fayl dəyişəni ilə təsvir olunur. Fiziki faylın strukturu informasiya daşıyıcılarında ardıcıl baytlardan ibarətdir.

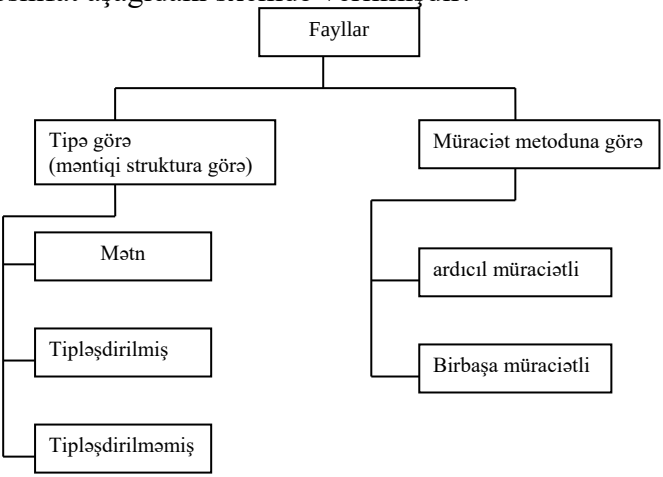
Faylın məntiqi strukturu prinsipcə massivə oxşayır. Amma fayl və massivin bəzi fərqləri mövcuddur. Massiv əməli yaddaşda təşkil olunur və onun elementlərinin sayı məlumdur. Faylda isə proqramın işi prosesində elementlərinin sayı dəyişir və o, xarici yaddaşda yerləşir. Faylın sonuna ASCII kodu 26(CTRL+Z) olan Eof simvolu qoyulur.

Faylların təsnifatı

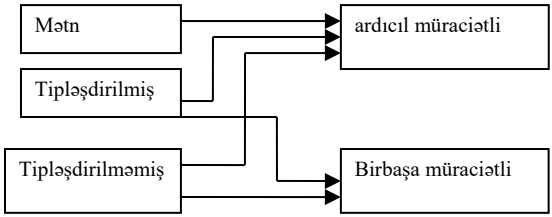
Fayllar iki əlamətə görə təsnifləşdirilir:

- Tipi (məntiqi srtuktura görə) görə,
- faylın elementlərinə müraciətə görə.

Bu təsnifat aşağıdakı sxemdə verilmişdir:



Faylların mümkün müraciət üsulları aşağıdakı sxemdə verilmişdir:



fayllarla işləmək, yəni xarici yaddaş daşıyıcılarına (disketlərə) məlumatın yazılıb oxunma texnologiyasını aşağıdakı qaydada yerinə yetirmək olar:

1. Faylın strukturu-tipi elan olunmalıdır. Bu proqramın verilənləri təsvir etmə bölməsində fayl dəyişəni vasitəsilə edilir.

2. Tərtib olunacaq (və ya əvvəllər tərtib olunmuş) faylın adı və yaddaşdakı yeri – faylın yolu göstərilməlidir. Bu əməliyyata fayl dəyişəni ilə fiziki faylın əlaqələndirilməsi deyilir və Assign proseduru vasitəsilə yerinə yetirilir.

3. Pascal proqramlaşdırma sistemi tərəfindən tərtib olunmuş fayldan istifadəyə icazə verilməlidir. Bu icazəyə faylın açılması deyilir.

4. İcazə alarkən istifadənin məqsədi göstərilməlidir. Bu məqsəd

- fayla yazmaq,
- əlavə etmək,
- oxumaq ola bilər.

Sonra bu əməliyyatlar yerinə yetirilməlidir. Qeyd edək ki, fayl açılarkən bu əməliyyatlardan yalnız biri yerinə yetirilə bilər.

5. Sonra başqa əməliyyat yerinə yetirmək üçün açılmış fayl bağlanılmalıdır.

6. Bu əməliyyatları yerinə yetirmək üçün istifadə olunan prosedur və funksiyalar faylın strukturundan – tipindən asılı olaraq müəyyən qədər bir-birindən fərqlənə bilər. Buna görə də faylın tipinə müvafiq bu əməliyyatlara ayrı-ayrılıqda baxaq.

18.1.Mətn faylları

1. **fayl dəyişəni.** Mətn faylında f fayl dəyişəni

f: text

əmri ilə müəyyənləşdirilir.

Mətn faylları da **fayl of char** tipli fayllar kimi simvol tipli verilənlərdən ibarət olsa da, onların bir sıra fərqli cəhətləri də vardır. Bildiyimiz kimi fayllarda Eof aldı simvol faylın sonunu göstərir. Mətn fayllarında, digər fayllardan fərqli olaraq sətrin sonu da göstərilir. Bu simvolun kodu “vozvrat karetki” və “perevod stroki” adlı simvolların kodlarından (13 və 10) ibarətdir.

2. **Fayl dəyişəni ilə fiziki faylın əlaqələndirilməsi.** Bu əməliyyat, məsələn,

Assign (f,’mf.dat’)

əmri ilə yerinə yetirilə bilər. Burada mf.dat adı f məntiqi faylın cari diskdə, cari kataloqda olan faylın adıdır. Fayl cari diskdə, cari kataloqda deyilsə, faylın yolu göstərilməlidir. Bu faylın yolunu başqa bir dəyişənlə də əvəz edib, Assign əmrində bu addan istifadə etmək olar. Məsələn, Name:='a:/mf/mf.dat';

```
Assign (f, name);
```

3. **Faylların açılması və bağlanması**

Mətn fayllarının açılması Reset (f), rewrite (f) və ya Append (f) əmrləri ilə, bağlanması isə close(f) əmri ilə yerinə yetirilir.

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

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

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

Əgər Append (f), Reset (f) əmrində göstərilən f məntiqi faylına müvafiq adda fiziki fayl yoxdursa, proqram bu haqda məlumat verəcək və proqramın icrası dayandırılacaqdır. Bu dayanmanı {Si-} translyator direktivi ilə ləğv etmək olar.

Bu translyator direktivinin nəticəsini İOResult funksiyası vasitəsilə təhlil etmək olar. Bu funksiya mövcud faylı açmaq istədikdə heç bir səhv yaranmasa "0", əks halda "1" qiymətini alacaqdır. Məsələn,

```
Assign (f, file_name);
{Si-}
Append (f);
if İOResult<>0 {əgər göstərilən adda fayl yoxdursa }
then rewrite (f) ; {yenisini yaratmalı }
```

4. **Fayllara yazıb oxuma.** Mətn faylında verilənlər read, readln, write, writeln əmrləri vasitəsilə daxil edilir. İndiyə kimi istifadə etdiyimiz bu klaviaturadan oxuma, monitora və ya printerə yazma (çap etmə) əmrlərindən fərqli olaraq, bu zaman bu əmrlərin ilkin parametrləri kimi fayl dəyişəni göstərilməlidir. məsələn,

read (f, a, b);

Burada f fayl dəyişəni a, b isə dəyişənlərdir.

Eof(f) funksiyası faylın sonuncu elementi oxunduqda True, əks halda false məntiqi qiymətini alır.

Misal. Prd.txt adlı mətn faylı yaradın və "Pascal proqramlaşma dili" sətri sabitlərini həmin fayla yazın Program Fayl_yaratma_ ;

```
var fa,x1,x2,x3:string;
fd:Text;
Begin
Readln(fa);
Assign(fd,fa)
Rewrite(fd);
X1:='Pascal';
X2:='proqramla.d.rma';
X3:='dili"
Writeln(fd,x1,x2,x3);
Close(fd) ; end.
```

Proqram icra olunduqda, daxil edilən adla (Prd.txt) xarici yaddaşda fayl yaradılır və x1, x2, x3 dəyişənlərinin uyqun qiymətləri həmin fayla yazılır.

18.2. Tipləşdirilmiş fayllar

1. **Fayl dəyişəni.** Tipləşdirilmiş faylların f fayl dəyişəni

f: file of <tip>

əmri ilə müəyyənləşdirilir.

Bu fayllarda elementlərin tipi fayldan başqa istənilən tipdə ola bilərlər. Tipləşdirilmiş faylların elementləri sıfırdan başlamaqla ardıcıl olaraq nömrələnir. Bu da belə fayllara həm ardıcıl, həm də birbaşa qayda ilə müraciət etməyə imkan verir.

2. Fayl dəyişəni ilə fiziki faylın əlaqələndirilməsi

Bu əməliyyat, məsələn, Assign (f, 'mf.dat');

və ya Name:='a:/mf/mf.dat';

Assign (f, Name); əmrləri ilə yerinə yetirilə bilər.

3. Faylların açılması və bağlanması

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

Reset (f) əmri ilə f məntiqi faylına müvafiq fiziki fayl açılır. Əgər bu fayl tipləşdirlmiş faylıdırsa, onu həm oxumaq, həm də yazmaq olar.

Close (f) əmri ilə fayllar bağlanır.

4. Fayllara yazıb oxuma Tipləşdirilmiş fayllardan oxumaq read, fayla yazmaq isə write əmrləri ilə yerinə yetirilir. Bu zaman yazma oxuma vahidini faylın tipi ilə eyni tipdə olan dəyişən müəyyənləşdirir.

Read və write operatorları aşağıdakı şəkildə yazılır.

read(fayl dəyişəni, dəyişənlərin siyahısı);

write (fayl dəyişəni, dəyişənlərin siyahısı);

Misal: C diskində 20 həqiqi ədəddən ibarət fayl yaratmalı. Sonra bu faylın elementlərini 5 sütunlu, 4 sətirli massivə mənimsətməli, massivin minimum elementini tapmalı.

```
PROGRAM fayl;USES CRT;
VAR F:FILE OF REAL;
M:ARRAY [1..4,1..5] OF REAL;
MS,max:REAL;
  I,J:INTEGER;
BEGIN
clrscr;
ASSIGN(F,'C:/BP/BIN\verilen.DAT');
REWRITE(F);
FOR I:=1 TO 20 DO
BEGIN
READ(MS);  WRITE(F,MS);
END; CLOSE(F);
ASSIGN(F,'C:/BP/Gun\DAN.DAT');
RESET(F);
 FOR I:=1 TO 4 DO
BEGIN
FOR J:=1 TO 5 DO
BEGIN
READ(F,M[I,J]);  WRITE(M[I,J]:6:1);
END; WRITELN; END;
min:=m[1,1];
FOR I:=1 TO 4 DO
BEGIN
```

```
FOR J:=1 TO 5 DO
if min>=m[i,j] then max:=m[i,j];
END; writeln('min=',min:6:1);
CLOSE(F); END.
```

18.3. Tipləşdirilməmiş fayllar

Belə faylların fayl dəyişəni

f: file;

əmri ilə elan edilir.

Bu fayl dəyişəni vasitəsilə ixtiyari məntiqi struktura malik fayla, simvol fayllarına müraciət etmək olar. Fərq ondadır ki, belə fayllarda oxuma-yazma zamanı tipləşdirilmiş fayllardan fərqli olaraq, məlumatlar qabaqcadan faylın tipi ilə müəyyənləşdirilən vahidlərlə yazılıb oxunmur. yazılıb-oxuma vahidləri proqramçı tərəfindən müəyyənləşdirilə bilər. Bu da belə fayllara yazıb-oxumanı sürətləndirir.

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

Tipləşdirilməmiş fayllarda məlumatlar BlockRead, Block- Write operatorları ilə yazılıb-oxunur. Faylları açan (fayllarla işləməyə icazə verən) Reset və ReWrite prosedurlarından fayl dəyişənindən başqa , ikinci dəyişəndən də istifadə etmək olar. Bu parametr məlumatları yazarkən və oxuyarkən yazının ölçüsünü göstərir. Əgər bu parametr buraxılmışdırsa, onda parametr sistem tərəfindən 128 bayt qəbul olunur.

18.4. Fayllara müraciət

1. Fayllara ardıcıl müraciət

Ardıcıl müraciət qaydası fayllarla adi işləmək qaydasıdır. Fayllara ardıcıl müraciət, yəni fayllara yazıb / oxuma aşağıdakı əməliyyatlar vasitəsilə yerinə yetirilir. Faylın dəyişənini təyin etməklə faylın tipi təyin olunur.

2. Fayllara bir başa müraciət

Bunun üçün aşağıdakı funksiya və prosedurlar nəzərdə tutulmuşdur.

Fayllarla işləmək üçün prosedurlar

Prosedurun adı	Yerinə yetirdiyi vəzifə
Assign (FV,Str)	FV fayl tipli dəyişəninə ad verilir. Bu ad Str sətir növlü dəyişənin qiyməti olur.
Rewrite (FV)	Diskdə yeni fayl yaratmaq üçün istifadə olunur.
Reset (FV)	Diskdə olan faylı oxumaq üçün istifadə olunur.
Read (FV, P)	Fayl oxumaq üçün istifadə olunur. Prosedura yerinə yetirildikdə FV fayl tipli dəyişəni ilə müəyyən olunan fayldan P dəyişəninin qiyməti oxunur.
Write (FV, P)	Proseduradan fayla yazmaq üçün istifadə edilir. Bu zaman P-nin qiyməti FV ilə müəyyən olunan fayla yazılır.
Seek (FV, n)	Faylın n-ci elementinin qeyd olunmasını təmin edir (elementlər sıfırdan nömrələnir).
Close (FV)	Faylı bağlayır.

Fayıllarla işləmək üçün funksiyalar

Funksiyanın adı	Vəzifəsi
Eof (FV)	Faylın sonunu müəyyən edir.
File Pos (FV)	Funksiyanın qiyməti tam ədəddir. Həmin ədəd FV faylın göstəricisinin yerləşdiyi elementinin nömrəsidir.
File Size (FV)	Funksiyanın qiyməti FV dəyişəninə uyğun olan faylın elementlərinin sayını göstərir.

19.MODULLAR

Pascal dilinin imkanlarını genişləndirmək məqsədilə bir sıra standart kitabxanalardan-modullardan istifadə edilir.

Turbo Pascal dilinin standart modulları aşağıdakılardır:

SYSTEM- Turbo Pascalın bütün standart funksiya və proseduralarını özündə saxlayır. Bu modulu göstərmək lazım deyil, çünki o avtomatik yüklənir;

CRT- Bu modul mətn rejiminin idarə olunmasını, ekranda pəncərələrin təşkili, simvolların rəngləndirilməsi, kursorun idarəolunması, klaviaturadan daxil olan sorğuların emalı kimi funksiyaları yerinə yetirir. Bum odul Turbo Tpl adlı sistem kitabxanasında saxlanılır. Modulu proqramda istifadə etmək üçün proqramın başlığında **USES CRT** direktivi yazılmalıdır. Modulda bir sıra prosedur və funksiyalardan istifadə olunur. Məsələn,

ClrScr-cari pəncərəni bağlayır;

INSLine-cari sətrin yerində boş sətir yaradır;

NoSound –səslənməsini dayandırır;

Delay (ms:Word)-ms millisaniyə müddətində proqramın işini dayandırır və s.

PRİNTER- printerə çıxışı təmin edir. Printerə çıxış üçün modul göstərildikdən sonra çıxış proseduru olan *WRITE* və ya *WRİTELN* daxilində LST fayl dəyişəni göstərilməlidir. Məsələn x dəyişəninin qiymətini çap etmək üçün

WRİTELN (LST, X:8:2);

yazılır. Burada x- həqiqi tiplidir, onun qiymətindəki ümumi rəqəmlərin sayı 8, vergüldən sonrakı rəqəmlərin sayı isə 2-dir.

GRAPH- Qrafika ilə işləmək üçün istifadə olunur. Bu modulda bütün prosedur və funksiyaları aşağıdakı qruplara bölmək olar:

1. Qrafik rejimlərin idarə olunmasının təhlili;
2. Qrafik funksiyaların çəkilişi;
3. Müxtəlif şablonların idarə olunması;
4. Bit əməliyyatları;
5. Səhifələrin idarə olunması;
6. Qrafik nəticələr;
7. Mətnlərin idarəedilməsi;

8. Qrafik rejimlərin idarə olunması.

TURBO3 və GRAPH3- Turbo Pascalın əvvəlki versiyaları ilə uyğunlaşmanı təmin edir (uyğun olaraq adi proqram və qrafik rejimlərdə):

DOS- PS DOS (MS DOS) sisteminin proqramlarına çıxışı təmin edir;

OVERLAY- Bu modul proqramlaşdırmada overleylərdən istifadəyə imkan verir. Adətən overleylər iri həcmli proqramlarla işləyərkən lazım olur. Proqram həcmcə iri olduqda yaddaş çatışmamazlığı problemi ilə qarşılaşmaq olar. Əgər OVERLAY modulunun imkanlarından istifadə olunsa proqram yerinə yetirilərkən yalnız əməli yaddaşa proqram daxilində olan və cari vaxtda istifadəsi tələb olunan proqram cağrılır. Ona müraciət qurtardıqda avtomatik olaraq geriyə qaytarılaraq yeni cağrilan funksiya və ya prosedura əməli yaddaşa yüklənir. Beləliklə yaddaş çatışmamazlığı aradan qaldırılır.

20. DINAMIK STRUKTURLU VERILƏNLƏR

İndiyə kimi baxdığımız proqram parametrləri üçün müəyyən olunmuş yaddaş ayrılır və bundan başqa proqramın ayrı-ayrı elementləri arasında əlaqə hələ kompilyasiya mərhələsində yaranır. Proqramın iş prosesi vaxtı ayrılmış yaddaş ölçüsünü və ya təyin olunmuş hər hansı əlaqəni dəyişmək mümkün deyil.

Pascal dilində proqramın yerinə yetirilməsi anında müxtəlif verilənləri yerləşdirmək üçün lazım olan yaddaş ayırmaq və boşaltmaq imkanı mövcuddur. Beləliklə verilənləri dinamik təşkil etmək mümkündür. Bu imkan verilənlərin xüsusi tipi olan göstərici ripi ilə əlaqədardır.

Turbo pascal-da göstərici parametrin bilavasitə ünvanla işləməsinə imkan verir. Göstərici tipin təyinində qarşısında ^ - göstərici işarəsi qoyulmuş baza tipindən istifadə olunur. Məs.,

```
type
Mas=array[1..10] of real;          {baza tipi}
Pmas=^Mas;          {10 həqiqi ədəddən ibarət massivin göstərici tipi}
```

Göstərici tiplərlə işləyərkən, heç nəyi göstərməyən **nil** standart tipindən istifadə olunur. Dinamik strukturlu verilənlərin təsvirində hər hansı dəyiçənin özü yox, onun göstəricisi göstərilir. Nəticədə göstərici adi dəyişən olur, dəyişəni göstərən isə dinamik olur. Məs.,

```
var
P:Char;
Begin
P^='*';
...
end.
```

P dəyişəni aşağıdakı üç vəziyyətdə ola bilər.

1. Yaddaşda yeri ayrılan hər hansı dəyişənin ünvanını saxlayır.
2. xüsusi nil boş ünvanı saxlayır.

Naməlum vəziyyətdə yerləşir.

Naməlum vəziyyətin nil - dən fərqi ondadır ki, bu iki göstərici dəyişəni bərabər deyil.

Dinamik dəyişənlər üçün yaddaşın ayrılması və boşaldılmasında New, Dispose, GetMem, Freemem, Mark və Release standart prosedurlarından istifadə olunur.

New (A) proseduru A göctəricisi üçün təsvir olunan tipə uyğun yaddaş sahəsini ayırır və ayrılan yaddaşın ünvanını göstəriciyə yazır.

Dispose (A) göctəricisini göstərən yaddaş sahəsini boşaldır. Bundan sonra bu sahə digər dinamik dəyişənlər üçün ayrıla bilər.

Tipləşdirilməmiş göstəricilər üçün bu əməliyyatlar GetMem(P, Size), Freemem(P, Size) prosedurları ilə yerinə yetirilir.

Burada P – göstərici, Size – baytlarla ayrılan yerin ölçüsüdür.

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

Dinamik verilənlər əlaqəsiz əlaqəli və əlaqəli əlaqəsiz bilər.

Əlaqəsiz dinamik verilənlər statik verilənlər kimi təsnif olunur. Bu verilənlərdən istifadənin aşağıdakı iki fərqli cəhəti var:

- var bölməsində tələb olunan tipin dəyişəni yox, bu tipin göstəricisi təsvir olunur; Məs., P: ^Char;

- istifadə olunmazdan əvvəl New proseduru, istifadədən sonra isə Dispose proseduru çağırılır.

Əlaqəli dinamik strukturlu verilənlərin strukturu aşağıdakı şəkildə verilmişdir

Xətti siyahılar - xətti əlaqəli bircins elementlərdən təşkil olunmuş dinamik strukturlu verilənlərdir. Burada iki element arasına element əlavə etməyə və silməyə icazə verilmir.

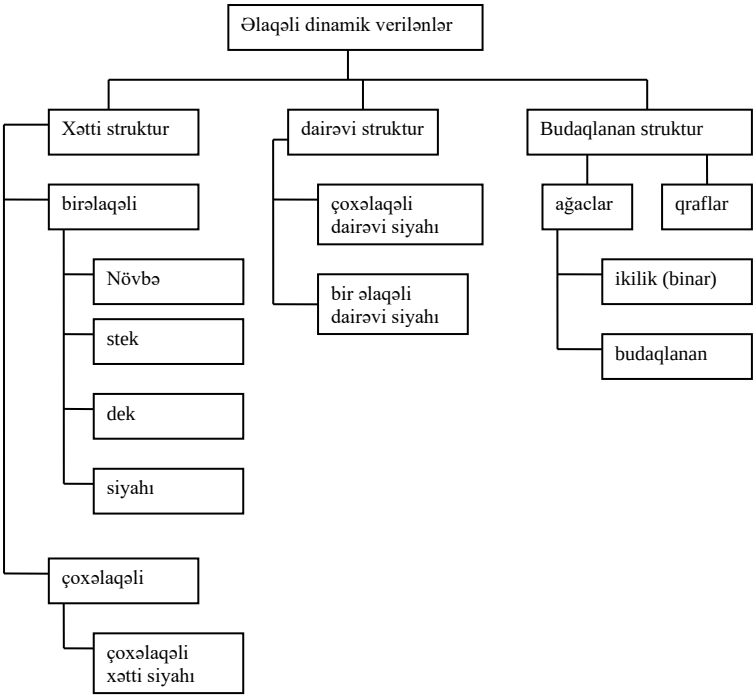
Dairəvi siyahılar - bu xətti siyahılardakı kimidir. Amma burada siyahının sonuncu və birinci elementləri əlaqəlidir.

Növbə - xətti bir əlaqəli siyahının xüsusi halıdır. Amma burada yalnız iki əməliyyata icazə verilir. Növbənin sonuna elementin əlavə edilməsi və növbənin başlanğıcından elementin silinməsi.

Stek - xətti bir əlaqəli siyahının xüsusi halıdır ki, elementin əlavə olunmasına və silinməsinə siyahının bir tərəfindən icazə verilir. Bu stekin təpəsi adlanır.

Ağaclar – sərbəst konfiqurasiyalı iyerarxik strukturlu dinamik verilənlərdir. ağacın elementləri təpə adlanır.

Piramida (nizamlanmış ağac) – bir səviyyədən növbəti səviyyəyə keçdikdə təpənin qiymətləri artan və ya azalan ağacdır.



Misal1:Verilmis tam ədədlərin cəmlənməsi, hasili, fərqi və quvvətə yüksəldilməsi üçün proqram tərtib edin.

```
PROGRAM DIN_DƏYISƏN;
  VAR P1,P2,P3,P4,P5,P6,P7:^INTEGER;
  BEGIN
  {P1,P2,P3,P4,P5 GÖSTƏRICILƏRI MÖVCUDDUR VƏ ONLARIN ILKIN
QIYMƏTLƏRI-NIL-SABITDIR}
  NEW(P1);  (*TAM TIPLI P1^,P2^,P3^,P4^,P5^ DƏYISƏNLƏRI
YARADIRIQ*)
  NEW(P2); NEW(P3); NEW(P4); NEW(P5);NEW(P6);NEW(P7);
  READLN(P1^,P2^,P3^);
  P4^:=P1^+P2^;P5^:=P1^*P2^;
  P6^:=P5^*P5^;
  P7^:=P6^-P5^;
  WRITELN('P4=',P4^,'P5=',P5^,'P6=',P6^,'P7=',P7^);
  READLN;
  END.
```

Misal2: Aşağıdakı ifadələri hesablamaq üçün proqram tərtib edin. (proqramda iştirak edən dəyişənlər dinamik dtrikturlu dəyişənlərdir)

p6= $\sqrt[P5]{p1}/7$, P4:=P1+P2; P5:=P1*P4; P7:=sin(P6/P5);

P1 dəyişəni [3;5] parçasında h=0,5 addımı ilə dəyişir.

```
PROGRAM DIN_DƏYISƏN;    USES CRT;
  VAR P2,P3:^INTEGER;P1,P4,P5,P6,P7:^REAL; X:REAL;
 BEGIN    CLRSCR;
 {P1,P2,P3,P4,P5 GÖSTƏRICILƏRI MÖVCUDDUR VƏ ONLARIN ILKIN
QIYMƏTLƏRI-NIL-SABITDIR}
 NEW(P1); (*TAM TIPLI P1^,P2^,P3^,P4^,P5^ DƏYISƏNLƏRI
YARADIRIQ*)
 NEW(P2);  NEW(P3);  NEW(P4);  NEW(P5);NEW(P6);NEW(P7);
 READLN(P2^,P3^);  WRITELN('P2=',P2^:3,'  P3=',P3^:3);
 P1^:=3;    REPEAT
 P4^:=P1^+P2^;
 P5^:=P1^*P4^;
 P6^:=EXP(1/P5^*LN(P1^))/7;
 P7^:=SIN(P6^/P5^);
 WRITELN('P1=',P1^:5:2,'  P4=',P4^:5:2,'  P5=',P5^:5:2,'   P6=',P6^:6:4,'
P7=',P7^:6:4);
 P1^:=P1^+0.5;
 UNTIL P1^>=5;
 READLN;   END.
```

21. QRAFİK REJİMƏ KEÇİD VƏ MƏTN REJİMİNƏ QAYITMA QAYDALARI

Müasir dövrdə mövcud olan kompüterlər işə salınan andan və Turbo Pascal mühiti yükləndikdə ekran mətn rejimində işləməyə başlayır.Bu səbəbdən kompüterin qrafik imkanı və vasitələrdən istifadə olunan istənilən proqramın icra olunması üçün müəyyən olunmuş qaydada displey adapterinin qrafik iş rejiminə keçməsi tələb olunur. Bu məqsədlə qrafik drayverin işə qoşulmasını təmin etmək lazımdır. Drayver-kompüterin bu və ya digər texniki qurğusunun idarə olunmasını həyata keçirən xüsusi proqramdır. Qrafik drayver, kompüterin displey adapterini qrafik rejimdə idarə edir. Adətən, bu drayverlər diskdə ayrıca kataloqdur (BGI). BGI genişlənməsinə malik

olan fayllar şəklində saxlanılır (BGI-Borland Graphics İnterface- Borland şirkətinin qrafik interfeysi). EGA və VGA tipli adapterlər üçün drayver- EGAVGA. BGI, CGA adapteri üçün- CGA.BGI adlı fayllar şəklində verilir.

Hər bir konkret adapterin qrafik imkanları ekranın əks etdirmə qabiliyyətindən, ekrandakı piksellərin (işıqlanan nöqtələrin), rənglərin sayından asılıdır. Adapterlərin bir çoxu eyni zamanda bir neçə qrafik səhifə ilə işləyə bilir. Qrafik səhifə əməli yaddaşın hər bir rikselin işıqlanması haqqında informasiyanı özündə saxlayan ekran "xəritəsini" özündə saxlayan hissəsidir. Aşağıda bəzi adapterlərin qısa olaraq qrafik rejimdə işləmə xarakteristikasını verək:

CGA (Color Graphics Adapter – rəngli qrafik adapter): 5 qrafik rejimə malikdir. Bunlardan dörd rejim aşağı əksetdirmə imkanına malik olan ekranlar üçündür. Bu ekranların ölçüsü 320x200-dür. Yəni üfiqi istiqamətdə 320 piksel, şaquli istiqamətdə 200 pikselə malikdir. Bu ekranların 4 rəng politrası mövcuddur: 0-cı palitra (açıq yaşıl, sarı, qızılgül rəngi), 1-ci palitra (açıq-firuzəvi, moruq rəngi,ad), 2 –ci palitra (yaşıl, qırmızı, qəhvəyi), 3-cü palitra (firuzə rəngi, bənövşəyi, açıq boz), 4-cü palitra (qara-işıqlanmayan piksellər).

Adapterin 5-ci rejimi yüksək əksetdirmə qabiliyyətinə malik olan ekranlı olub 640x200 ölçülüdür.

CGA adapteri qrafik rejimdə yalnız bir qrafik səhifədən istifadə edir.

EGA (Enhanced Graphies Adapter- gücləndirilmiş qrafik adapter)

21.1. Qrafik rejimə keçid və mətn rejiminqə qayıtmaq məqsədilə istifadə olunan prosedura və funksiyalar

1. İnitGraph prosedurası

Adapteri qrafik iş rejiminə keçirir. Ümumi formatı aşağıdakı kimidir:

Procedure Graph (var Driver, Mode: Integer:Path:String);

Burada Driver- Integer tipli dəyişən olub, qrafik drayver tipini təyin edir;

Mode-Integer tipli dəyişən olub, qrafik adapterin iş rejimini verir;

Path-String tipli ifadə olub, drayver faylının adını və onun tapılması marşurunu göstərir.

Bu proseduranın köməyilə drayver əməli yaddaşa yüklənir və adapteri qrafik rejimə keçirir. Drayverin tipii qrafik adapterin tipinə uyğun olmalıdır. Drayverlərin tipləri aşağıdakı sabitlərlə verilmişdir:

Detect =0;-tipin avtomatik müəyyən edilməsi rejimi.

CGA=1; MCGA=2; EGA=3; EGA64=4; EGAMONO=5;

IBM 8514=6; HEREMONO=7; ATT400=8; VGA=9; PC 3270;

Adapterlərin əksəriyyəti müxtəlif rejimlərdə işləyə bilirlər. Adapterə tələb olunan rejimi göstərmək məqsədilə Mode dəyişənindən istifadə olunur.

2. GraphResult funksiyası

Qrafik proseduraya sonuncu müraciət nəticəsi kimi Integer tipli qiymət qaytarılır. Əgər mürasiət zamanı heç bir səhvə yol verilməyibsə, funksiyanın qiyməti sıfırdır, əks halda mənfi qiymət olur.

GraphResult funksiyasına müraciət olunduqdan sonra səhvlərin əlamətləri ləğv edilir, məhz buna görə də ona təkrarən müraciət "0" qaytarır.

3. GraphErrorMsg funksiyası

Göstərilən səhv koduna uyğun String tipli nəticəni, mətn məlumatını verir. Yazılışı aşağıdakı kimidir:

Funcsion GraphErrorMsg (Code:Integer): String;

Burada, Code- GraphResult funksiyası ilə verilən səhvin kodudur.

4. CloseGraph prosedurası

Adapterin qrafik rejimdə işini başa çatdırır və ekranın işinin mətn rejimini bərpa edir. Ümumi yazılışı belədir:

Procedure CloseGraph;

5. Restore CRTMode prosedurası

Qısa müddətə müfəqqəti olaraq mətn rejimini qaytarır. CloseGraph proseduraşından fərqli olaraq qrafik rejimin müəyyən olunmuş parametrləri ləğv edilmir və qrafik drayverin yerləşdiyi yaddaş hissəsi silinmir. Ümumi yazılışı belədir:

Procedure Restore CRTMode;

6. Get GraphMode funksiyası

Integer tipli qiyməti verir. Bu qiymət qrafik adapterin iş rejiminin kodudur. Funksiya aşağıdakı kimi yazılır:

Funcsion GetGraphMode: Integer;

7. Set GraphMode prosedurası

Adapterin yeni qrafik iş rejimini təyin edir. Ümumi yazılışı belədir:

Procedure Set GraphMode (Mode: Integer);

Burada, Mode –təyin olunan rejimin kodudur.

Göstərilən funksiya və proseduralardan istifadə edərək aşağıdakı proqramı nəzərdən keçirək:

```
Program grafikrejim;
Uses Grt, Graph;
Var
Driver, Mode, Sahv: Integer;
Begin
Driver:=Deteet;          [drayverin tipii]
İnitGraph (Driver, Mode, 'C:\TP\BGI'); [qrafik rejimə keçid]
Sahv:= GraphR                [sahvin nəticəsi]
IF Sahv <> grok then         [sahvi yoxlayırıq]
Writeln(GraphErrorMsg (Sahv))    [sahv var]
    Else
Begin  [sahv yoxdur] Writeln ('Biz qrafik rejimdəyik');
Writeln ('Enter düyməsini basın');
Readln;
Restore CRTMode; [qısa müddətə mətn rejiminə keçid] Writeln
('İndi mətn rejimindəyik');
Readln;
Set GraphMode (Get GraphMode);  [yeni qrafik rejim verilir]
Writeln ('Qrafik rejimə qayıdırıq');
Readln;
CloseGraph;
End;End.
```

8. DeteetGraph prosedurası

Drayverin tipini və onun iş rejimini verir. Ümumi yazılışı belədir:

Procedure DeteetGraph (var Driver, Mode:Integer);

Burada, Driver- drayverin tipli, Mode- iş rejimidir. Bu prosedura Get GraphMode funksiyasından fərqli olaraq Mode dəyişəni üçün qrafik rejimin Ən böyük nömrəsini qaytarır.

9. Get DriverName funksiyası

Yüklənmiş qrafik drayverin adını verir. Bu ad String tipli qiymətdir. Funksiyanın başlığı belədir:

Funcsion Get DriverName:String;

10. Get ModeName funksiyası.

String tipli nəticə verir. Bu nəticə nömrəyə uyğun adapterin iş rejiminin adıdır. Funksiyanın başlığı belədir:

Funcsion Get ModeName(Mod Number:Integer):String;

Burada, Mod Number-rejimin nömrəsidir.

11. Get MaxMode funksiyası.

Integer tipli nəticə verir. Nəticə adapterin mümkün iş rejimini sayını göstərir. Funksiyanın başlığı belədir:

Funcsion Get MaxMode (Mod Number:Integer):String;

12. Range prosedurası.

İstifadə olunan qrafik adapterin mümkün iş rejimləri diapozonunu verir. Proseduranın başlığı belədir:

Procedure Get Mode Range (Drv:Integer;var Min, Max :Integer);

Burada Drv – adapterin tipii, Min və Max – Integer tipli dəyişən olub, uyğun olaraq rejimin nömrələrinin mümkün ən kiçik və ən böyük qiymətləridir.

Əgər Drv parametrinin qiyməti düzgün verilməyibsə, onda həm Min, həm də Max dəyişənləri -1 qiymətini alacaq.

21.2. Koordinatlar, rəncərələr və səhifələrlə işləmə qaydaları

Kompüterin ekranda qrafik rejimdə koordinatlar ekranın yuxarı sol küncünə nəzərən müəyyən edilir. Ekranan sol küncünün koordinatları (0,0) qəbul edilir. Buna görə də, üfüqi koordinatlar soldan sağa, şaquli koordinatlar isə yuxarıdan aşağı artır.

İndi isə qrafik ekranda koordinatlar, müxtəlif rəncərələr və qrafik səhifələrlə işləmək üçün lazım olan funksiya və proseduralarla tanış olaq.

1. Get MaxX və GetMaxY funksiyaları

Word tipli qiymətlə qaytarır. Bu funksiyaları qrafik rejimdə kursorun mövqeyini üfqi və şaquli koordinatlarını göstərir. Koordinatlar cari pəncərənin və ya ekranın yuxarı sol küncünə nəzərən müəyyən olunur.

2. GetX və GetY funksiyaları

Integer tipli qiymətlə qaytarır. Bu funksiyaları ekrana cari qrafik rejimin maksimal koordinatlarını (uyğun olaraq üfqi və şaquli) göstərir.

3. Set View Port prosedurası

Qrafik rejimdə düzbucaqlı rəncərə müəyyən edir. Proseduranın başlığı aşağıdakı kimidir:

Procedure SetViewPort (X1,Y1,X2,Y2:Integer;Clipon:Boolean);

Burada -X1,Y1 – pəncərənin yuxarı sol küncünün koordinatları;

X2,Y2 – pəncərənin aşağı sol küncünün koordinatları;

Clipon- Boolean tipli ifadə olub, ekranın rəncərə daxilində yerləşməyən elementlərinin kəsilərək atılmasını (əks etdirilib-etdirilməməsini) müəyyən edir. Belə ki, Clipon parametrinin qiyməti True (doğru) olarsa, onda ekranın təyin olunmuş pəncərədən kənardakı elementləri silinir və görünmür, əks halda pəncərənin sərhədləri nəzərə alınmır və bütün ekran elementləri olduğu kimi qalılır. Qeyd edək ki, Clipon əvəzinə Glipoff parametri verilərsə, yuxarıdakı ikinci hal baş

verir, yəni pəncərə sərhədləri nəzərə alınmır və bütün ekran elementləri əks olunur.

4. Get View Settings prosedurası

Cari qrafik pəncərənin koordinatlarını və ' kəsib atma ' əlamətini verir.

Proseduranın başlığı aşağıdakı kimidir:

Procedure SetViewSettings (var View Info: View Port Type) ;

Burada -View Info – View Port Type tipli dəyişəndir. Bu tip Graph modulunda aşağıdakı şəkildə müəyyən olunub:

```
type
View Port Type=record
   X1,Y1,X2,Y2: Integer;Clip:Boolean;End;
```

5. Move To prosedurası.

Ekran göstəricisinin yeni mövqeyini müəyyən edir. Proseduranın başlığı aşağıdakı kimidir:

Procedure Move To (X,Y: Integer) ;

Burada, X,Y- göstəricinin yeni koordinatlarıdır. Bu koordinatlar təyin olunmuş pəncərənin yuxarı sol küncünə nəzərən müəyyən olunur, ədər rəncərə yoxdursa, onda ekranın yuxarı sol küncünə görə koordinatlar müəyyən edilir.

6. Move Rel prosedurası

Ekran göstəricisinin nisbi koordinatlarla yeni mövqeyini göstərir. Proseduranın başlığı aşağıdakı kimidir:

Procedure Move Rel (DX, DY: Integer) ;

Burada, DX, DY- göstərici koordinatlarının üfüqi və şaquli istiqamətdə uyğun olaraq artımıdır. Artım proseduraya müraciət anındakı göstəricinin mövqeyinə görə müəyyən edilir.

7. Clear Device prosedurası

Bu prosedura qrafik ekranı təmizləyir. Bu proseduraraya mürasiət etdikdən sonra qrafik funksiya və proseduraların

parametrlərinin əvvəlki qiymətləri bərpa olunur. Məsələn, göstərici yuxarı sol küncə qoyulur, pəncərələr ləğv olunur və s. Proseduranın başlığı aşağıdakı kimidir:

Procedure Clear Device;

8. ClearViewPort prosedurası

Bu prosedura qrafik pəncərəni təmizləyir. Bu zaman pəncərə 0 nömrəli rənglə və cari palitra ilə rənglənir. Proseduranın başlığı aşağıdakı kimidir:

Procedure ClearViewPort;

21.3. Nöqtələr və düz xəttlərin çəkilməsi

1. PutPixel prosedurası

Göstərilən kordinatlarla verilmiş rənglə nöqtə çıxarır(çəkir). Proseduranın başlığı aşağıdakı kimidir:

Procedure PutPixel (X,Y: Integer;Color:Word);

Burada, X,Y-nöqtənin koordinatları, Color-nöqtənin rəngidir.

Koordinatlar yuxarı sol küncə nisbətən verilir. Aşağıdakı proqramm vasitəsilə ekranın müxtəlif hissələrində, müxtəlif rəngli nöqtələrin çəkilməsi və ya verilməsini müşahidə edək:

```
    PROGRAM NOQTA;
    USES CRT, GRAPH;
    VAR
D, M, S:INTEGER;
R,X,Y,X1,Y1,X2,Y2: INTEGER;
    BEGIN
D=:DETECT;
INITGRAPH (D, M,'C:\TP\BGI');
S:= GRAPHRESULT;
IF S<>GROK THEN WRITELN(GRAPH ERROR MSG(S))
ELSE
BEGIN                    [EKRANDA PƏNCƏRƏ YARADIRIQ]
    X1:=GETMAX X DIV 4;
    Y1:= GETMAX Y DIV 4;
    X2:=3*X1;
    Y2:=3*Y1;
RECTANGLE (X1,Y1,X2,Y2);
SETVIEWPORT (X1+1,Y1+1,X2-1,Y2-1,CLIPON);
```

```
FOR R:=1 TO 15 DO
BEGIN                              [NÖQTƏLƏRI ÇƏKIRIK]
X:=RANDOM(X2-X1);
Y:= RANDOM(Y2-Y1);
    PUTPIXEL (X,Y,R);
END; READLN;
CLOSEGRAPH;
 END; END.
```

2. GetPixel funksiyası

Word tipli qiymət olub, göstərilən koordinatlardakı pikselin rəngini verir. Funksiyanın başlığı belədir:

Funcsion Get Pixel(X,Y;INTEGER): Word;

Burada, X,Y- pikselin koordinatlarıdır.

3. Line prosedurası

Başlanğıc və son koordinatları verilmiş xətti çəkir. Proseduranın ümumi şəkildə yazılışı aşağıdakı kimidir:

Procedure Line (X1,Y1,X2,Y2: Integer);

Burada, X1,Y1 –xəttin başlanğıc, X2,Y2-son nöqtəsinin koordinatlarıdır. Xəttlər cari üslubda və cari rənclə çəkilir.

17. Line To prosedurası

Göstəricinin cari mövqeyi ilə koordinatları verilmiş nöqtəni birləşdirən parça çəkir. Ümumi yazılışı aşağıdakı kimidir:

Procedure Line To (X,Y: Integer);

Burada, X,Y –xəttin sonunun koordinatlarıdır.

5. Line Rel prosedurası

Göstəricinin cari mövqeyi ilə onun koordinatlarına verilmiş artımı verməklə alınan nöqtəni birləşdirən parça çəkir. Ümumi yazılışı aşağıdakı kimidir:

Procedure Line Rel (DX, DY: Integer);

Burada, DX, DY –yeni koordinatların alınması üçün verilən uyğun addımlardır. Son iki proseduraların hər ikisində xəttlər cari təyin olunmuş üslub və rənglə çəkilir.

6. SetLine Style prosedurası

Çəkiləcək xətlərin yeni üslubunu müəyyən edir. Proseduranın ümumi şəkildə yazılışı aşağıdakı kimidir:

Procedure SetLine Style (Type, Pattern, Thick: Integer);

Burada, Type–xəttin tipi, Pattern- xəttin nümunəsi və Thick xəttin qalınlığıdır.

Xəttin tipini müəyyənləndirmək məqsədilə aşağıdakı sabitlərdən istifadə olunur:

CONT

SolidLn=0; (bütöv xətlər)

DottedLn=1; (nöqtəvi xətlər)

CenterLn=2; (ştrix-punktir xətlər)

DashedLn=3; (punktir xətlər)

UserBitLn=4; (istifadəçinin müəyyən etdiyi növ)

Pattern parametri yalnız xəttin çəkilməsi naxışını istifadəçi təyin etdikdə istifadə olunur (başqa sözlə Type=UsesBitln olduqda). Bu halda Pattern parametrinin iki baytı xəttin nümunəsini müəyyən edir. Bu baytların 1-olan biti xəttin işıqlanan pikselinə, 0 isə işıqlanmayan pikselinə uyğun olur.

Thick parametri aşağıdakı iki qiymətdən birini alır:

Norm Width=1; (xəttin qalınlığı 1 pikselə bərabərdir)

Thieh Width=3; (xəttin qalınlığı 3 pikselə bərabərdir)

21.4. Çoxbucaqlıların qurulması

1. Rectangle prosedurası

Künclərin koordinatları verilmiş düzbucaqlı çəkir. Proseduranın başlığı belədir:

Procedure Rectangle (X1,Y1,X2,Y2: Integer);

Burada, (X1,Y1) –düzbucaqlının yuxarı sol küncünün, (X2,Y2)- aşağı sağ küncünün koordinatlarıdır. Düzbucaqlı cari rəng və cari xətt üslubu ilə çəkilir.

2. Draw Poly prosedurası

Sınma nöqtələrinin verilmiş koordinatlarına uyğun olaraq istənilən sınıq xətti çəkir. Proseduranın başlığı belədir:

Procedure Draw Poly (N:Word;var Points);

Burada, N –sınma nöqtələrinin sayıdır, sınıq xəttin hər iki uc nöqtələri də bura daxildir;

Points-Point tipli dəyişən olub, sınma nöqtələrinin koordinatlarını verir. Sınma nöqtələrinin koordinatları cütlər şəklində Word tipli qiymətlərdir. Sınıq xətt sari rəng və cari xətt üslubunda çəkilir. Bu cütdəki birinci rəqəm üfiqi, ikinci isə şaquli koordinatı müəyyən edir. Tip modulda aşağıdakı kimi verilir:

```
Type
Point Type=record
x, y: Word;
end;
```

Misal: Yuxarı sol küncünün koordinatı (x1;y1), aşağı sağ küncünün koordinatı (x2;y2) olan düzbucaqlı çəkən proqramın tərtibi.

```
PROGRAM DUZBUCAQLI;
USES GRAPH,CRT;
VAR D,E,M,X1,Y1,X2,Y2:INTEGER;
BEGIN
D:=DETECT;
INITGRAPH(D,M,'C:\BP\BGI ');
E:=GRAPHRESULT;
IF E<>GROK THEN WRITELN (GRAPHERRORMSG(E))
ELSE BEGIN
READLN(X1,Y1,X2,Y2);
SETCOLOR(13);
RECTANGLE(X1,Y1,X2,Y2);
IF READKEY=#0 THEN D:=ORD(READKEY);
CLOSEGRAPH; END; END.
```

Misal: Dioqanalının uç nöqtələrinin koordinatları (x_1,y_1) və (x_2,y_2) olan düzbucaqlı çəkmək üçün proqram tərtib edin.

```
PROGRAM DDD;
USES GRAPH,CRT;
VAR D,R,E,X1,X2,Y1,Y2:INTEGER;
BEGIN
D:=DETECT;
INITGRAPH(D,R,'C:\BP\BGI');
E:=GRAPHRESULT;
IF E<>GROK THEN WRITELN (GRAPHERRORMSG(E))
ELSE
BEGIN
X1:=GETMAXX DIV 4;
```

```
  Y1:=GETMAXY DIV 4;
  X2:=3*X1;
  Y2:=3*Y1;
RECTANGLE(X1,Y1,X2,Y2);
READLN;
CLOSEGRAPH;
END; END.
```

21.5. Qövslərin, çevrələrin və ellipslərin çəkilməsi qaydaları

1. Circle prosedurası

Çevrə çəkir. Proseduranın ümumi forması aşağıdakı kimidir:

Procedure Circle (X,Y:Word;var Points);

Burada, X, Y çevrənin mərkəzinin koordinatları, R-çevrənin piksel verilmiş radiusudur. Xəttin növü həmişə Solidln (bütöv) olub, qalınlığı çəkilən andakı cari üslubla müəyyən olunur. Düzgün çevrənin çəkilməsi məqsədlə qrafik ekranın tərəflərinin ölçüləri nisbətini nəzərə almaq və GETAspect Ratio əmsalından istifadə etmək lazımdır.

Misal: Mərkəzinin koordinatları (120,160), radiusu r=100 olan cevrə çəkmək üçün proqram tərtibi.

```
PROGRAM CEVRƏ; USES GRAPH,CRT;
CONST X=120;Y=160; VAR D,R,E:INTEGER;
      BEGIN
      D:=DETECT;INITGRAPH(D,R,'C:\BP\BGI ');
      E:=GRAPHRESULT;
      IF E<>GROK THEN WRITELN (GRAPHERRORMSG(E))
      ELSE BEGIN
      CIRCLE(X,Y,100); READLN;
                        CLOSEGRAPH; END; END.
```

2. Arc prosedurası

Çevrə qövsünü çəkir. Proseduranın ümumi forması aşağıdakı kimidir:

Procedure Arc (X,Y:INTEGER;BB,SB,R:Word);

Burada, X, Y mərkəzinin koordinatları, BB,SB-qövsün başlanğıc və son bucaqları, R-çevrənin radiusudur.

Bucaqlar saat əqrəbinin əksi istiqamətdə hesablanır ıə dərəcə ilə göstərilir. '0’ dərəcəli bucaq soldan sağa üfüqi istiqamətdə çəkilən vektordur.

Misal: Mərkəzinin koordinatları (120,160), başlanğıc bucağı bb=60, son bucağı sb=120 və radiusu r=100 olan qövs çəkmək üçün proqram tərtib edin.

```
        PROQRAM DDD;
 (*BB-QOVSUN BASLANGICI, SB-QOVSUN SONU,R-RADIUSU*)
USES GRAPH,CRT;CONST X=120;Y=160;BB=60;SB=120;
VAR D,R,E:INTEGER;
BEGIN
D:=DETECT;
INITGRAPH(D,R,'C:\BP\BGI ');
E:=GRAPHRESULT;
IF E<>GROK THEN WRITELN (GRAPHERRORMSG(E))
        ELSE BEGIN
        AR C(X,Y,BB,SB,100);
        READLN;
        CLOSEGRAPH; END;  END.
```

3. Ellipse prosedurası

Ellips qövsünü çəkir. Proseduranın ümumi forması aşağıdakı kimidir:

Procedure Ellipse (X,Y:INTEGER:BA,SA,RX,RY:Word);

Burada, X, Y-mərkəzin koordinatları, BA,SA-qövsün başlanğıc və son bucaqları, RX,RY- ellipsin pikselə verilmiş üfüqi və şaquli radiuslarıdır.

21.6 Ekranla işləmək üçün olan proseduralar

1. CrtExit prosedurası

sintaksisi: CrtExit

Əməliyyatlar sisteminin yüklənməsi zamanı olan rejimi bərpa edir.

2. Crinnit prosudarısı

sintaksisi: Crinnit

Sistemin qurulması zamanı müəyyən olunan terminalın inisiallaşmış sətirlərini ekrana çıxarır.

3. ClrEol prosedurası

sintaksisi: ClrEol

kursorun duruduğu yerdən sətrin sonuna qədər olan bütün simvolları silir.

4. Clrscr prosedurası

sintaksisi: Clrscr

ekranı tamamilə təmizləyir və kursoru ekranın yuxarı sol küncündə yerləşdirir. Ekranın mətn rejimində yerinə yetirili.

5. Delline prosedurası

sintaksisi: Delline kursorun durduğu sətri tamamilə silir və ondan aşağıda duran sətirləri bir sətir yuxarı qaldırır.

6. Gotoxy prosudurası

sintaksisi: Gotoxy (x,y:integer);

Kursoru x və y koordinatları ilə müəyyən olunan mövqeyə gətirir.

7. İnsline prosedurası

sintaksisi: İnsline

kursorun durduğu sətri boş sətir edir və ondan aşağıda duran bütün sətirlərin aşağıya doğru bir sətir yerini dəyişdirir.

Digər prosedura və funksiyalar

8. Delay prosedurası

sintaksisi: Delay (ms : integer);

Proqramın yerinə yetirilməsini ms millisaniyə qədər gecikdirir. Məs; Delay (2500) proqramın yerinə yetirilməsini 2,5 saniyə gecikdirir.

9. Halt prosedurası

sintaksisi :Halt

proqramın yerinə yetirilməsini dayandırır.

Move, Random, Randomize, Addr, Hi, KeyPressed, Lo, Sizcof, Swap, Upcase kimi digər prosedura və funksiyalar da vardır.

22. PASCAL ALQORİTMİK DİLİNİN MENYULAR SİSTEMİ

I File menyusu

Bu menyuya daxil olan rejim və əmrlər ilə tərtib olunmuş proqramın mətni ilə redaktorda əsas işləmə imkanları həyata keçirilir. Bu rejimlər aşağıdakılardır:

- **NEW** əmri vasitəsilə redaktorun yaddaşı təmizlənir və onun yeni fayla yazılması hazırlanır;
- **OPEN** Mövcud olan Turbo Pascal faylı açılaraq redaktora verilir;
- **SAVE** Bu əmrlə fayl redaktordan diskə yazılır (öz adı ilə);
- **SAVE AS** faylı redaktordan diskə yazır Save –dən fərqli olaraq yazılacaq faylın atributları istifadəçi tərəfindən təyin edilir;
- **SAVE ALL** açıq elan edilmiş bütün faylları diskə yazır (onların bütün parametrləri və müxtəlif variantları ilə);
- **CHANGE DIR** istifadə olunan diskin qurğu və qurğudakı kataloqun dəyişdirilməsini təmin edir;
- **PRİNT** faylı çapa göndərir;
- **PRİNT SETUP** istifadə olunacaq çap qurğusunun parametrlərinin və xüsusiyyətlərinin verilməsini təmin edir;
- **Dos SHELL** müvəqqəti olaraq Turbo Pascalı operativ yaddaşdan silmədən əməliyyat sisteminə çıxışı təmin edir. Bu halda əməliyyat sisteminin müxtəlif əmrlərindən istifadə etmək, nizamlama işlərini yerinə yetirmək mümkündür. Yenidən Turbo Pascala qayıtmaq üçün EXİT əmri verilir;
- **EXİT** Turbo Pascaldan çıxmaq onu operativ yaddaşdan silmək üçün istifadə olunur.

II Edit menyusu

Menyu redaktorda olan faylın redaktə olunmasını, düzəlişini və əlavələrini yerinə yetirməyi təmin edən əmrləri özündə saxlayır. Əmrlər aşağıdakılardır:

- **UNDO** son dəfə yerinə yetirilən əməliyyatı ləğv edir;
- **Redo** son ləğv edilmiş əməliyyatı bərpa edir;
- **Cut** bloka ayrılmış proqram mətnini ekrandan silir və buferə göndərir;
- **Copy** bloka ayrılmış proqram mətninin surətini buferə göndərir;
- **Paste** buferin məzmununu kursor duran yerə əlavə edir;

- **Clear** bloka ayrılmış proqram mətnini ekrandan silir;
- **Show clipboard (mübadilə buferini göstərmək)** Müvəqqəti mübadilə buferinin məzmununu göstərir.

III Search

Menyu proqram və onun müxtəlif elementlərinin axtarılıb tapılmasını təmin edir. Aşağıdakı alt menyuları var:

- **Find** verilən elementlərin axtarılıb tapılmasını təmin edir;
- **Replace** hər hansı elementin digəri ilə əvəz olunmasını təmin edir;
- **Search again** elementin yenidən axtarılıb tapılmasını təmin edir:
- **Goto Line number** nömrəsi göstərilən proqram sətrinə keçidi təmin edir;
- **Show last compiter error s**onuncu kompilyasiyada sintaksis səhv tapılmış proqram sətrini göstərir;
- **Find error** proqram mətnində səhvi axtarıb tapır;
- **find procedure** nizamlama rejimində proqram mətnində lazımi proseduru axtarıb tapır;
- **Provions browser** ümumi browzerə baxışı təmin edir;
- **Object**s tərtib olunmuş proqramın OBJ modulunu tapır;
- **Onits** proqramın tam mətnini axtarıb tapır;
- **Global**s ümumi qurğuda axtarışı təmin edir;
- **Simbol** proqram mətnindən verilən simvolu axtarıb tapır.

IV Run

Menyu proqram və onun elementlərinin yerinə yetirdiyi icranı təmin edir. Alt menyular aşağıdakılardır:

- **Run** redaktorda olan proqramı kompilyasiyasını və icrasını təmin edir;
- **Step** over proqramın yerinə yetirilməsini addım rejimində həyata keçirir**;**
- **Trace into** əmri əvvəlki əmr kimi işləyir, ondan fərqli olaraq prosedura və ya funksiyaya rast gəldikdə idarəetmə onların daxilinə verilir;
- **Goto** cursor kompilyasiya və komponovkadan sonra icra adi qaydada yerinə yetirilir, lakin yerinə yetirilmə kursor duran sətirdən həyata keçirilir və həmin sətirin rəngi digərlərindən fərqlənir. adətən qırmışı rəngdə olur**;**

- **Proqram** reset proqramın icrasını dayandırır, yaddaşdan yerinə yetirilən proqram silinir və açıq olan bütün fayllar bağlanır;

- **Parametrs** proqramın yerinə yetirilməsilə əlaqədar olan parametrlərin müəyyən olunması və yenidən təyininə imkan verir.

V Compile

Menyunun əmrləri redaktorda olan proqramın kompilyasiyasını təmin edir. Aşağıdakı alt menyuları var:

- **Compile** cari vaxtda redaktorda olan proqramı kompilyasiya edir. Əgər proqramda qeyri-standart modullara müraciət edilmişdirsə, bu modullar əvvəlcədən kompiiyasiya edilməli və diskdə TPU faylı kimi saxlanmalıdır.

- **Make** kompilyasiya olunmuş fayl yaradır. əvvəlcə ilkin faylı kompilyasiya edir. Müraciət olunan modul kompilyasiya olunmayıbsa, kompilyasiya edir və yenidən ilkin proqrama qayıdılır. əgər müraciət olunan modulun yerləşdiyi proqram tapılmazsa, onun oxşarı və ya köhnə variantından istifadə olunur;

- **Build** əvvəlki əmr kimi işləyir, fərqi ondan ibarətdir ki, müraciət olunan modullara uyğun Pas variantı tapılır, onun kompilyasiya olumasından asılı olmayaraq yenidən kompilyasiya edir;

- **Primary file** ilkin faylın adını göstərir. Bu proqramın hansı hissəsinin yerinə yetirilməsindən asılı olmayaraq, kompilyasiyada adı göstərilən fayldan başlayır. Əgər ilkin fayl göstərilməzsə, kompilyasiya yalnız əsas proqramda yerinə yetirilir;

- **Clear Primary file** ilkin fayl kimi göstərilən faylı silir. adətən bu əmr prosesin yenidən təyini və ya digər tələbata uyğun olaraq verilir;

- **Information** kompilyasiya olunmuş fayl haqda aşağıdakı məlumatları ekrana verir: son qurğu, kataloq və faylın ölçüsü, əlavə yaddaşın ölçüsü, kompilyasiya olunmuş sətirlərin sayı, proqram seqmentinin ölçüsü, verilənlər seqmentinin ölçüsü və s.

VI Debug

Menyu əmrlərinin köməyilə nizamlanan proqramın ixtiyari dəyişənini yoxlamaq və ya dəyişmək, proqram stekinin məzmununa baxmaq olur. lazımi prosedur və ya funksiyanı tapmaq mümkündür.

Aşağıdakı alt menyuları var:

-Break points proqramın mətnində nəzarət nöqtəsinin müəyyən olunması, növbəti nəzarət nöqtəsinə keçimi təyin edir;

-Callstack proqram stekini və onun haqda məlumatı ekrana çıxarır;

-Registr kompüterə proqram mətni ilə işləyən registr haqqında məlumat verir;

-Watch əmri nizamlama pəncərəsini aktivləşdirir;

-Output yerinə yetirilən proqramın nəticəsinin pəncərəyə çıxışını təmin edir;

-Uses Screen proqram pəncərəsini aktiv edir, proqramın nəticələri olan pəncərəni bütün ekran böyu açır;

-Evaluate/modify... Bu rejim nizamlama prosesində istənilən dəyişənin məzmununa və ya idafənin qiymətinə baxmağa imkan verir. Lazım olduqda onun köməyilə istənilən dəyişənə yeni qiymət vermək olar. bu rejimə müraciətdən sonra ekranda yeni dialoq pəncərəsi açılır. Pəncərədə üç parametr (sahə) verilir:

-EXPRESSİON (ifadə) sahədə istənilən dəyişənin və ya ifadənin adı göstərilir;

-RESULT (nəticə) sahəsində uyğun qiymət əks olunur. Əgər belə dəyişən və ya ifadə proqramda yoxdursa, onda **unknown identifier(müəyyən olunmuş identifikator)** məlumatı verilir;

-NEW valie (yeni qiymət) sahəsində dəyişən və ya ifadəyə qiymət verilir

Bu dialoq pəncərəsindən çıxmaq üçün ESC düyməsindən istifadə olunur;

- **Add Watch** proqramı nizamlayan zaman istifadəçiyə işləmək istədiyi dəyişən və ya ifadəni pəncərəyə gətirir;

- **Add break point** proqram sətrinə nəzarət nöqtəsinin əlavə olunmasını təmin edir.

VII Tools menyusu

Menyu əmrləri əlavə servis xidmətləri yerinə yetirməni həyata keçirir. Bunlar aşağıdakılardır:

-Messages müxtəlif məlumar alınması;

-Goto next növbəti addıma keçməni;

-Goto provions əvvəlki addıma keçidi;

-GREP grep utilitlərinin işləməsini təmin edir.Açılan dialoq pəncərəsindəki enter proqram arquments sətrində GREP-in çağırılması arqumentlərini göstərmək lazımdır. Bu arqumentlər mətn faylında axtarılacaq prosedura, funksiya və dəyişənlərin adı və faylın adıdır. Susmaya görə GREP üçün *.pas qəbul olunur ki, bu da öz növbəsində bütün cari kataloqun pas fayllarında axtarışın aparılmasına səbəb olur. Əgər çağırış zamanı kursor hər hansı prosedura, funksiya və ya dəyişənin qarşısındadırsa, onda onun adı arqument kimi qəbul olunur. Axtarışın nəticəsi kimi Message pəncərəsində tapılan ada uyğun faylın adı, sətir nömrəsi və proqram mətnini özündə əks etdirən GREP məlumatı görünür.

-Turbo Assembler assembler dili ilə əlaqədar işin təşkilini

-Turbo Debuqqer proqram nizamlayıcısı ilə əlaqədar xidmətlərdən istifadə imkanlarını yerinə yetirir.

VIII Options

Bu menyu Turbo Pascal mühitinin işini proqram köməyilə və onun komponovkası rejimlərini idarə edən əmrlərdən ibarətdir. Bu əmrlər aşağıdakılardır:

- **Compile** əmri proqramın maşın kodlarına generasiyasını (çevrilməsini) idarə etməyən imkan verən bir neçə parametr verir. Rejim yerinə yetirilərkən bir neçə dialoq pəncərəsi əmrləri əks olunur.
- **Memore size** stekə və ya dinamik yaddaşa müraciət etdikdə proqram üçün mümkün olan yaddaşın yaddaşın ölçülərini müəyyən edilir;
- **Linker** əmri ilə Turbo Pascal –ın komponovkasının iş rejimləri tənzimlənir;
- **Debuqqer** Bu rejim istifadə olunan nizamlayıcını və nizamlama vaxtı displey ekranının yeniləşvəsirejimini müəyyən edir. Onun bir neçə variantları mövcuddur;
- **Direktories** əmr verildikdə açılan menyu əmrləri ilə diskin və ya qurğuların kataloqları ilə işləməni təmin etmək olar;
- **Browser** əmr ümumi browserdən istifadəni təmin edir

- **Tools** servislə əlaqədar olan parametrlərin və atributların təyin olunmasını təmin edir;

- **Enviroment** Turbo Pascal mühitinin idarəolunması, bununla əlaqədar əlavə menyularla işləməni təmin edir;

- **Open** yaddaşa çağırılacaq və redaktora yüklənəcək fayllara uyğun parametrləri təyin müəyyən edir;

- **Save** proqramla əlaqədar rejimlərdə edilən dəyişikliyin saxlanılmasını təmin edir;

- **Save as** əvvəlki rejimlə eynidir. Fərq ondadır ki, yaddaşda saxlanılan informasiyanı istifadəçinin istəyi ilə həyata keçirir.

IX Window menyusu

Menyunun əmrləri proqram və onun yerinə yetirdiyi əmrləri ilə əlaqədar pəncərələrlə işləməni təmin edir. Bunlar aşağıdakılardır:

-**Tile** başlıqlar pəncərəsinin verilməsi

-**Cascade** açılan pəncərələrin kaskad şəklində verilməsi;

-**Close all** bütün açıq pəncərələrin bağlanması;

-**Refrest display** ekranın ilkin məzmununu bərpa edir;

-**Size/Move** pəncərənin ölçüsünü və yerini təmin edir;

-**Zoom** pəncərənin görünmə miqyasını təmin edir;

-**Next** növbəti pəncərəyə keçid;

-**Provions** əvvəlki pəncərəyə qayıdış;

-**Close** aktiv pəncərəni bağlayır;

-**List** pəncərədə proqram mətnini verir.

X Help menyusu

Menyu müxtəlif əmrlərə aid sorğulara uyğun məlumatların alınmasını təmin edir. Bu menyunun alt menyuları aşağıdakılardır:

-**Contents** verilənlərə uyğun sorğulara cavab verir;

-**İndex** indekslər haqqında məlumat verir;

-**Topic Searsh** ilkin axtarışın təşkili qaydalarını təmin edir;

-**Provions topic** əvvəlki axtarışlara uyğun məlumatları axtarır;

-**Using help** istifadəçi köməyinin alınmasını təmin edir;

-**Files** Bu rejimin köməyilə istifadəçi sorğu xidmətinin lazımi fayllarına sazlaya bilər;

-**Compiler direktories** kömpilyasiya kataloqlarına uyğun köməyin alınmasını təmin edir;

-**Prosedures and Funktions** prosedura və funksiyaya uyğun köməkliklərin alınmasını təmin edir;

-**Reserved Words** dilin ümumi təyinatlı sözləri haqqında məlumat verir;

-**Standart inites** standart funksiyanın proqramları haqda məlumat verir**;**

-**Borland Pascal language** pascal dili haqda məlumat verir;

-**Error messages** səhvlər haqda məlumat verir;

-**Aboud...** menyu və onun elementləri haqda ümumi məlumat verir.

ƏDƏBİYYAT SIYAHISI

1. Грогоно П. Программирование на языке Паскаль/ пер. с англ.- М. Мир, 1982.

2. Сергеев Н. П., Доминин Л. Н. Алгоритмизация и программирование – М. Радио и связь, 1982.

3. Тихонов А.Н., Костомаров Д. П. Вводные лекции по прикладной математике – М. Наука, 1984.

4. Перминов О.Н. Программирование на языке Паскаль – М. Радио и связь, 1988.

5. Форсайт Р. Паскаль для всех/ Пер. с англ. – М. Машиностроение, 1986.

6. Абрамов С.А., Зима Е. В., Начала программирования на языке Паскаль – М. Наука, 1987.

7. Прайс Д. Программирование на языке Паскаль. Практическое руководство/ пер. с англ.- М. Мир, 1982.

8. Семашко Г. Л., Салтыков Л. М., Голубев – Новожилов Ю.С. Программирование на языке Паскаль – М. Наука, 1988.

9. Шаньгин В. Ф., Поддубная Л. М., Голубев – Новожилов Ю.С. Программирование на языке Паскаль – М. Высшая школа, 1988.

10. П. Грогоно. Прогромирование на языке ПАСКАЛЬ – М. Мир, 1982.

11. Н. Вирт. Алгоритмы+структуры данных=Программы – М. Мир, 1985.

12. К.Боон. Паскаль для всех. Пер. с голланд. М. Энергоатомизадат, 1988.

13. В.С.Новичков, Н.И. Парафилова, А.Н. Пылькин. ПАСКАЛЬ. М. Высшая школа – 1990.

14. С. В. Глушаков, В.Н. Зорянский, С.Н. Хоменко. Для высших и средних учебных заведений. Turbo Pascal 7.0. Харков – Фолио, 2002.

MÜNDƏRİCAT