Təhsilə dair axtardığınız bir çox kitabın elektron versiyasını “**Telegram**” kanalımızda tapa bilərsiniz (https://t.me/eSources).

Telegram:
Kanal: @eSources
Reklam, təklif və iradlarınız üçün: @n4hkro

- Kitablar ödənişlidir?

✓ Xeyr, təbii ki.

- Paylaşdığınız kitabları öz kanalımda paylaşa bilərəm?

✓ Bəli. Könül istərdi ki, paylaşarkən mənbə bildirəsiniz, amma təbii ki, heç kim sizi buna məcbur etmir.

- Bədii kitablar da paylaşırsınız?

✓ Xeyr, amma həmin kitab sizə dərs üçün lazımdırsa, istisna edərik.

- Azərbaycan Milli Kitabxanasından kitab yükləyirsiniz?

✓ Bəli. Onlayn şəkildə oxunulması mümkün olan istədiyiniz kitabı yükləyirik.

- Sizə kitab göndərsəm qarşılığında nə alacağam?

✓ Kanaldan maddi qazancımız olmadığı üçün, bizə göndərdiyiniz kitablara görə ən yaxşı halda sizin kanalınızı reklam edə bilərik.

Ə.Ə.ƏLİYEV
A.Y.ƏLİYEV
C.K.KAZIMOV

İNFORMATİKANIN ƏSASLARI

AZƏRBAYCAN RESPUBLİKASI TƏHSİL NAZİRLİYİ
BAKI DÖVLƏT UNİVERSİTETİ

ƏLƏKBƏR ƏLİ AĞA OĞLU ƏLİYEV
AYDIN YUNUS OĞLU ƏLİYEV
CAVANŞİR KAZIM OĞLU KAZIMOV

İNFORMATİKANIN ƏSASLARI

Ali məktəblər üçün dərs vəsaiti

Azərbaycan Respublikası Təhsil Nazirliyi tərəfindən təsdiq edilmişdir (18 iyun 2008-ci il tarixli 771 saylı əmr).

BAKI – 2009

UOT 519.682

Rəyçilər AMEA-nın müxbir üzvü, texnika elmləri doktoru, professor A.Z. Məlikov.
fizika-riyaziyyat elmləri namizədi, dosent R.Ə. Mahmudzadə

Ə.Ə. Əliyev, A.Y. Əliyev, C.K. Kazımov. İnformatikanın əsasları. Ali məktəblər üçün dərs vəsaiti. – Bakı: Mütərcim, 2009. – 298 səh.

Dərs vəsaitində informatika elminin əsasları, kompüterlər, onların inkişaf tarixi, quruluşu, iş prinsipi, şəbəkələri, proqram təminatı (MS DOS, Windows, MS Paint, MS Word, MS Excel) nəzərdən keçirilir. Burada proqramlaşdırmanın əsasları, say sistemləri, alqoritm anlayışı, alqoritmlərin tipləri, ifadə formaları və qurulma qaydaları araşdırılır. TURBO PASCAL proqramlaşdırma dili və DELPHI proqramlaşdırma sisteminə baxılır.

İ $\frac{4306010000}{026}$ 93–09

GİRİŞ

Müasir dövrdə informasiya texnologiyaları və vasitələrinin iqtisadiyyata, təhsilə, idarəetmə məsələlərinə tətbiqi nəticəsində cəmiyyətin həyat və əmək şəraiti kökündən dəyişmiş, insanların informasiyalaşması səviyyəsi yüksəlmişdir. Cəmiyyətin informasiyalaşması informasiyanın rol və qiymətinin getdikcə artması ilə əlaqədardır. İnformasiyalı cəmiyyət informasiyanın toplanması, işlənməsi, saxlanması və ötürülməsini əhatə edən yüksək inkişaf etmiş informasiya sferası ilə xarakterizə olunur.

Cəmiyyətin informasiyalaşması prosesinin elmi əsasını informatika elmi təşkil edir. Hazırda «informatika» anlayışına yanaşmanın bir neçə istiqaməti vardır. Bu istiqamətlərdən biri (V.M. Qluşkov, V.S. Mixoleviç tərəfindən inkişaf etdirilmişdir) informatikaya informasiya texnologiyaları və kompüter sistemləri ilə əlaqəli olan bir elm kimi baxır. «İnformatika» anlayışına yanaşmanın ikinci istiqaməti K. Şennonun informasiya nəzəriyyəsi ilə əlaqədardır. Üçüncü istiqamətdə (E.S. Bernşteyn, Y.A. Şreyder) informasiyanın semantik (məzmunlu) tərəfi əsas götürülür.

Cəmiyyətin informasiyalaşması prosesini sürətləndirmək üçün demək olar ki, respublika ali məktəblərinin hamısında informatika fənni tədris olunur. Azərbaycan dilində bu fənnə aid olan ədəbiyyatların sayı olduqca azdır. Ona görə də Azərbaycan dilində informatika fənninə aid dərsliklərin və dərs vəsaitlərinin yazılmasına böyük ehtiyac vardır.

Bu dərs vəsaitinin əsas məqsədi informatikanın əsasları, sistem və tətbiqi proqram təminatı, alqoritmlər nəzəriyyəsinin elementləri, proqramlaşdırma dilləri (Turbo Pascal və Delphi) haqqında tələbələrə məlumat verməkdir.

Kitab 6 fəsildən ibarətdir. Birinci fəsildə informatika elmi, EHM-lər və onların arxitekturası, say sistemləri haqqında məlumat verilir.

İkinci fəsil kompüter şəbəkələrinə həsr olunub. Burada lokal və qlobal kompüter şəbəkələri, lokal şəbəkələrin qurulma topologiyaları haqqında ətraflı məlumat verilir. Eləcə də, informasiyanın ötürülməsinin fiziki mühiti, standart lokal şəbəkələr və müasir hesablama şəbəkələrinə qoyulan tələblərə baxılır.

Üçüncü fəsildə EHM-lər üçün nəzərdə tutulan proqramların klassifikasiyası verilir. Bu fəsildə həmçinin sistem və tətbiqi proqramlara (MS DOS, Windows əməliyyat sistemlərinə və MS Paint qrafik redaktoruna, MS Word mətn redaktoruna, MS Excel elektron cədvəlinə) və onlardan istifadə edilmə qaydalarına baxılır.

Dördüncü fəsildə alqoritmlər nəzəriyyəsinin elementləri, alqoritmlərin tipləri və hazırlanması qaydaları verilir.

Beşinci fəsildə Turbo Pascal alqoritmik dilinin əsas anlayışları və bu dildə proqram tərtib etmək qaydaları verilir. Kitabda verilən çoxsaylı proqram nümunələri mövzuların asan mənimsənilməsinə xidmət edir. Bu fəsildə Turbo Pascal-ın qrafik imkanları və inteqrallaşmış mühiti haqqında məlumat verilir.

Altıncı fəsildə Delphi proqramlaşdırma sistemi haqqında məlumat verilir. Bu fəsli tələbələr oxumaqla Delphi mühiti, proyektin xarakteristikaları, proqram interfeysinin hazırlanması və yerinə yetirilməsi, inteqrallaşmış mühitin vasitələri, Object Pascalda verilənlərin tipləri, Vizual komponentlərin xassələri və hadisələri, birsətirli və çoxsətirli redaktorlar, sadə və kombinasiyalı sıyahılar, əsas və köməkçi menyunun konsol proqramlarının hazırlanması qaydaları ilə tanış ola bilər və Delphi sistemində proqramlar yaza bilərlər.

Hesab edirik ki, bu kitabdan «informatika», «tətbiqi riyaziyyat», «riyaziyyat», «hesablama maşın və sistemlərinin riyazi və proqram təminatı» və bu istiqamətdə olan digər ixtisaslarda təhsil alan tələbələr və magistrlər istifadə edə bilərlər.

Sonda müəlliflər kitabın hazırlanmasında və onun məzmununa aid olan müzakirələrdə etdiyi məsləhətlərə görə rəyçilər AMEA-nın müxbir üzvü, t.e.d., prof. A.Z. Məlikova, BDU-nun dosenti, f.r.e.n. R.Ə. Mahmudzadəyə, eləcə də, kitabı kompüterdə tərtib etmiş baş laborant R.R. Mirzəyevə öz təşəkkürlərini bildirirlər.

Müəlliflər

I FƏSİL

İNFORMATİKANIN ƏSASLARI, ELEKTRON HESABLAMA MAŞINLARI (EHM)

1.1. İnformatika elmi haqqında

İnformatika – EHM-lər vasitəsilə informasiyanın tədqiqinin (emalının) ümumi qanunlarını, yəni informasiyanın verilməsi, yığılması və tətbiqi proseslərini araşdıran bir elmdir.

İnformatika – «Informatique» termini bizim əsrin 60-70-ci illərində fransızlar tərəfindən təklif olunmuşdur. Lakin onlardan əvvəl amerikalılar, hesablama texnikasının əsasında informasiyanın tədqiqini əhatə edən elmləri işarə etmək üçün «Computer Science» (hesablama elmləri) termini tətbiq etmişlər. Hal-hazırda «Informatique» və «Computer Science» terminlərindən ekvivalent mənada istifadə olunur.

EHM (elektron hesablama maşını) – informasiyanın avtomatik tədqiqini təmin edən elektron qurğudur.

Bizi əhatə edən aləm haqqında bütün elm və məlumatlara – informasiya deyilir. İnsan həmişə informasiyanı tədqiq edir:

1) yeni məlumatlar öyrənir (qəzetlərdən, kitablardan, radio və televiziyadan);

2) məlum informasiyadan istifadə edir (işində, həyatda);

3) yeni informasiyanı yaradır (elmdə, incəsənətdə, ədəbiyyatda və s.).

Müasir aləmdə informasiya çox böyük məna kəsb edir. İnsan fəaliyyətinin bütün sahələrində informasiya tədqiq etmək lazım gəlir. Xüsusi ilə isə elm, idarəetmə və siyasətdə informasiyanın tədqiqi əsas fəaliyyət sahəsidir. Bütün dünya əhalisinin yarısından çoxu informasiya tədqiqi ilə məşğuldur. İnsan təkcə bu qədər nəhəng həcmdə informasiyanı tədqiq etmək iqtidarında deyildir və onun köməyinə EHM-lər gəlir. Buna görə də müasir aləmdə hər bir insan öz işində EHM-dən istifadə etməyi bacarmalıdır.

Qeyd edək ki, öz məsələlərinin həllində hesablama texnikasından istifadə edən şəxslərə kompüter istifadəçiləri deyilir. Onları iki kateqoriyaya bölmək olar:

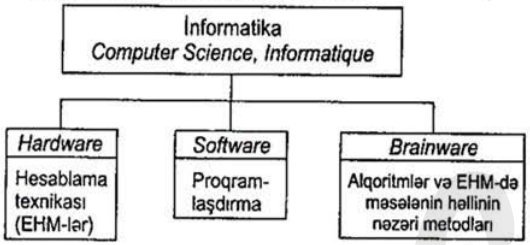
1) proqramçı istifadəçilər

2) proqramçı olmayan istifadəçilər.

Birinci kateqoriyaya proqramçı-analitiklər (məsələlərin qoyuluşu, analizi və həllinin ümumi prinsiplərini araşdıran şəxslər), sistem proqramçıları (sistem və alət proqramlarını hazırlayan şəxslər) və tətbiqi proqramçılar (konkret elm sahəsindəki məsələlərin həlli üçün tətbiqi proqramlar hazırlayan şəxslər) aiddir.

İkinci kateqoriyaya isə kompüter operatorları (kompüterlərin proqram təminatının, birinci növbədə sistem və alət proqramların tətbiqi ilə məşğul olan şəxslər) və sadəcə istifadəçilər (konkret məsələləri tətbiqi proqramlarla həll edən və ya hər hansı bir informasiya alan şəxslər) aiddir.

İnformatikanı üç hissəyə bölmək olar:



1) *Hardware* - hesmablama texnikası, informasiyanın tədqiqi üçün lazım olan qurğular;

2) *Software* - proqramlaşdırma, EHM-də istifadə olunan bütün proqramlar və onların qurulması;

3) *Brainware* - məsələlərin düzgün həlli üçün lazım olan bilik və vərdişlər.

1.2. EHM-lər və onların inkişaf tarixi

EHM-lərin yaranması, birinci növbədə fizika, riyaziyyat və texniki elmlərin tələbatı ilə diktə olunmuşdur. İnkişaf etməkdə olan elm və texnikanın məsələlərinin həlli üçün astronomik həcmdə hesablamalar aparmaq lazım gəlirdi.

EHM-lərin adına baxmaqla, belə fikir yarana bilər ki, bu maşınlar yalnız hesablama aparmağa yarayır. Lakin bu belə deyildir, EHM-lər ixtiyari informasiyanı tədqiq edə bilir. EHM vasitəsilə, misal üçün təyyarələrin uçuş cədvəlini tərtib etmək və

təyyarə biletlərini satmaq, abituriyentlərin imtahan cavablarını yoxlayıb, ali məktəblərə daxil olanların siyahısını tərtib etmək, dünyanın ixtiyari kitabxanası ilə əlaqə saxlayıb, lazım olan kitab və ya məqalənin surətini almaq olar. Xüsusi proqramlarla xarici dilləri öyrənmək, xəstəliyin düzgün diaqnozunu qoymaq, musiqi yazmaq, cizgi filmləri çəkmək, reklam hazırlamaq olar. Eləcədə EHM-lə şahmat və ya digər oyunlar oynamaq olar.

EHM-ləri «computer» - kompüter termini ilə işarə edirlər. Bu söz ingiliscə hesablayıcı, yəni hesablama üçün qurğu deməkdir. Ümumiyyətlə, kompüter - informasiyanı avtomatik tədqiq edə bilən ixtiyari qurğuya deyilir. Hal-hazırda bütün kompüterlər elektron tiplidir. Lakin ən birinci kompüter mexaniki tipli idi. İlk mexaniki kompüterin proyekti 1833-cü ildə İngiltərədə Bebbic tərəfindən təklif olunmuşdu.

İlk elektron tipli kompüter isə 1946-cı ildə ABŞ-da Pensilvan universitetində hazırlanmışdı. Bu EHM ENIAC (Electronic Numerical Integrator And Calculator) adlanırdı. Bu maşın həddindən artıq böyük idi və onu hətta bir yerdən başqa yerə aparmaq belə mümkün deyildi. Onun çəkisi 30 ton idi. Bu maşında 18000 elektron lampadan istifadə olunmuşdu və o, saniyədə 5000 əməliyyat apara bilirdi.

ENIAC və elektron lampalardan istifadə edilən digər EHM-lər, I nəsil EHM-ləri təşkil edir və bu 1940-1955-ci illəri əhatə edir.

1955-ci ildə növbəti II nəsil EHM-lər meydana gəldi. Bu EHM-də elektron lampalar əvəzinə yarımkeçiricilərdən – tranzistorlardan istifadə olunurdu. Bu EHM-lər ölçücə kiçik idi, onların işləməsi üçün daha az elektrik enerjisi tələb olunurdu. Bundan əlavə işləmə sürəti də artmışdı və saniyədə bir neçə on min əməliyyata bərabər idi. Həmin vaxtdan etibarən də proqramlaşdırma dillərindən istifadə edilməyə başlandı.

Bir müddətdən sonra elektron sənayesi inteqral sxemlər hazırlamağa başladı. İnteqral sxemlər – özündə bir neçə yüz və hətta bir neçə min tranzistoru birləşdirən, kiçik yarımkeçirici kristallardır. İnteqral sxemlər əsasında qurulan EHM-lər III nəsil kompüterlərdir. Bu EHM-lər böyük yaddaşa və sürətə - saniyədə bir neçə milyon əməliyyat yerinə yetirmək qabiliyyətinə malikdirlər.

Müasir EHM-lər IV nəsil EHM-ləridir. Onlar 70-ci illərin əvvəllərində meydana gəlmişlər. Bu kompüterlərin əsas elementini mikroprosessor və digər böyük inteqral sxemlər (BİS) təşkil edir. Bu inteqral sxemlər özündə artıq bir neçə yüz min tranzistoru birləşdirir.

Mikroprosessorun icad edilməsi isə informatikada inqilaba səbəb oldu. Nəticədə iş üçün ən əlverişli EHM-lər fərdi kompüterlər (PC – Personal Computer) meydana gəldi. Bu EHM-lər kiçik ölçüyə malikdir, lakin onların böyük yaddaşı var və informasiyanı onlar çox böyük sürətlə tədqiq edirlər. Fərdi kompüterlərin bir-birilə və ya digər böyük EHM-lərlə, məsələn adi telefon xətti ilə birləşdirmək olar. Onda EHM şəbəkəsi əmələ gəlir və digər kompüterlərin də yaddaşında olan çox böyük həcmli informasiyadan istifadə etmək mümkün olur.

1.3. Say sistemləri

EHM - elektron rəqəm qurğusudur. Elektron qurğudur, ona görə ki, burada ixtiyari informasiya elektrik siqnallarının köməyi ilə təsvir olunur. Rəqəm qurğusudur, ona görə ki, EHM-də ixtiyari informasiya rəqəmlərin köməyi ilə ifadə olunur.

Rəqəmləri yazmaq üçün, hər hansı bir say sistemindən istifadə etmək lazımdır. Say sistemləri ədədlərin yazılması və onlar üzərində cəbri əməliyyatların aparılması qaydalarını təyin edir.

Biz əsasən onluq say sistemindən istifadə edirik. Bu say sistemində ixtiyari ədəd on rəqəm, 0, 1, 2, ... , 9 rəqəmlərinin köməyi ilə yazılır.

Ümumiyyətlə isə say sistemləri iki tipə bölünür: mövqeli və mövqesiz say sistemləri. Mövqesiz say sistemlərində ixtiyari rəqəmin qiyməti, onun ədəddə tutduğu mövqedən asılı olmur. Bu say sistemlərinə misal olaraq Roma say sistemini göstərmək olar. Bu sistemdə məsələn, XXX ədədində X rəqəmlərinin durduğu yer onun qiymətinə heç bir tə'sir etmir.

Mövqeli say sistemində isə ixtiyari rəqəmin qiyməti, onun ədəddə tutduğu mövqedən asılıdır. Bizim ən çox istifadə etdiyimiz onluq say sistemi də mövqeli say sistemlərinə aiddir. Məsələn, bu say sistemində götürdüyümüz 1998 ədədində 1 rəqəmi 1×10^3, yəni min, 9 rəqəmi 9×10^2, yəni doqquz yüz, 9 rəqəmi 9×10^1, yəni doxsan və nəhayət 8 rəqəmi 8×10^0, yəni səkkizi göstərir.

Mövqeli say sistemlərində, ədədləri ifadə etmək üçün, bu say sisteminin əsası adlanan müəyyən sayda rəqəmlərdən istifadə edilir. Məsələn, onluq say sistemində bu 0, 1, 2, ... , 9 rəqəmləridir. Məhz buna görə də bu sistem onluq say sistemi adlanır. Mövqeli say sistemlərinə misal olaraq, eləcə də ikilik (0, 1), səkkizlik (0, 1, 2, 3, 4, 5, 6, 7), onaltılıq (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F) və s. göstərmək olar.

Hər bir say sistemində ədədlər rəqəmlər ardıcıllığı kimi yazılır. Rəqəmin ədəddəki yeri onun mərtəbəsini, rəqəmlərin sayı isə ədədin neçə mərtəbələri olduğunu göstərir. Məsələn, onluq say sistemində verilmiş 1998,437 ədədini mərtəbələrə aşağıdakı kimi ayırmaq olar:

$$\overset{3\ 2\ 1\ 0\ -1\ -2\ -3}{1998,437} = 1\times10^3 + 9\times10^2 + 9\times10^1 + 8\times10^0 + 4\times10^{-1} + 3\times10^{-2} + 7\times10^{-3}$$

Bu qaydadan istifadə etməklə, digər say sistemlərində verilmiş ədədləri, dərəcələrinə ayırıb, onların etalon kimi qəbul etdiyimiz onluq say sistemindəki ekvivalent qiymətlərini tapa bilərik. *Məsələn:*

$$\overset{5\ 4\ 3\ 2\ 1\ 0}{110110_{(2)}} = 1\times2^5 + 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0 = 32 + 16 + 4 + 2 = 54_{(10)}$$

$$\overset{2\ 1\ 0}{175_{(8)}} = 1\times8^2 + 7\times8^1 + 5\times8^0 = 64 + 56 + 5 = 125_{(10)}$$

$$\overset{2\ 1\ 0}{A1F_{(16)}} = 10\times16^2 + 1\times16^1 + 15\times16^0 = 2560 + 16 + 15 = 2591_{(10)}$$

Onluq say sistemində verilmiş tam ədədlərin digər say sistemlərindəki ekvivalent ədədlərini tapmaq üçün isə verilmiş ədədi bu say sisteminin əsasına bölərək, qalıq həddlərini yazmaq lazımdır.

Məsələn:

Ədəd	Bölən	Qalıq
54	2	0
27	2	1
13	2	1
6	2	0
3	2	1
1	–	1

$54_{(10)} = 110110_{(2)}$

Ədəd	Bölən	Qalıq
348	8	4
43	8	3
5	–	5

$348_{(10)}=534_{(8)}$

Ədəd	Bölən	Qalıq
875	16	11
54	16	6
3	–	3

$875_{(10)}=36B_{(16)}$

Nəticə alınan qalıq hədlərini əks istiqamətdə, yəni aşağıdan yuxarıya yazdıqda alınır.

Ədədlərin müxtəlif say sistemlərindəki ifadəsini aşağıdakı cədvəllə verək:

Onluq	*İkilik*	*Səkkizlik*	*Onaltılıq*
00	00000	00	00
01	00001	01	01
02	00010	02	02
03	00011	03	03
04	00100	04	04
05	00101	05	05
06	00110	06	06
07	00111	07	07
08	01000	10	08
09	01001	11	09
10	01010	12	A
11	01011	13	B
12	01100	14	C
13	01101	15	D
14	01110	16	E
15	01111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17
24	11000	30	18

Onluq say sistemində verilmiş düzgün kəsrləri digər say sistemlərinə keçirmək üçün isə verilmiş kəsri ardıcıl olaraq keçirilən say sisteminin əsasına vururlar. Vurma nəticəsində alınan ədədlərin tam hissələri həmin say sistemində verilmiş kəsri ifadə edəcəkdir. *Məsələn:*

0,	725×2
1,	450×2
0,	900×2
1,	800
	və s.

0,	873×8
6,	984×8
7,	872×8
6,	976
	və s.

0,	27×16
4,	32×16
5,	12×16
1,	92×16
14,	72

$0{,}725_{(10)}=0{,}101_{(2)}$ $0{,}873_{(10)}=0{,}676_{(8)}$ $0{,}27_{(10)}=0{,}451E_{(16)}$

Nəticəni almaq üçün hər dəfə vurma nəticəsində alınan ədədlərin tam hissələrini yuxarıdan aşağıya doğru yazmaq lazımdır. Qeyd edək ki, hər addımda say sisteminin əsasını, alınan yeni ədədin yalnız kəsr hissəsinə vururuq. Bir də qeyd edək ki, onluq say sistemində verilən düzgün kəsri, digər say sistemlərinə həmişə dəqiq çevirmək olmur və onların təqribi ifadəsindən istifadə olunur.

Səkkizlik say sistemində verilmiş ədədi ikilik say sisteminə keçirmək üçün hər bir səkkizlik rəqəmin yerinə, bu rəqəmə uyğun gələn üç ikilik rəqəmi (triada) qoymaq lazımdır. *Məsələn:*

6	7	5	3	2,	1	0	7
10	111	101	011	010,	001	000	11

yəni $67532{,}107_{(8)}=110111101011010{,}001000111_{(2)}$

Əksinə, ikilik say sistemindən səkkizlik say sisteminə keçdikdə, hər bir ikilik triadaların yerinə onlara uyğun səkkizlik rəqəmlər qoyulur. Bu zaman əgər rəqəmin əvvəli və sonunda tam üçlük (triada) alınmazsa, onlar soldan və sağdan sıfırlarla tamamlanır. *Məsələn:*

010	111	011	101,	110	100
2	7	3	5,	6	4

yəni $10111011101{,}1101_{(2)}=2735{,}64_{(8)}$

Onaltılıq say sistemində verilmiş ədədi ikilik say sisteminə keçirmək üçün hər bir onaltılıq rəqəmin yerinə, bu rəqəmə uyğun gələn dörd ikilik rəqəmi (tetrada) qoymaq lazımdır. *Məsələn:*

3	5	B,	4	5	1
0011	0101	1011,	0100	0101	0001

yəni $35B,451_{(16)}=1101011011,010001010001_{(2)}$

İkilik say sistemində verilmiş ədədi onaltılıq say sisteminə keçirdikdə isə hər bir ikilik tetradaların yerinə onlara uyğun onaltılıq rəqəmlər qoyulur və uclarda tam tetradalar alınmadıqda, bura soldan və sağdan sıfırlar əlavə olunur.

İkilik say sistemində cəbri əməliyyatlar aşağıdakı qaydalar üzrə aparılır:

		Toplama	*Vurma*
0	0	0	0
0	1	1	0
1	0	1	0
1	1	10	1

Çıxma		
0	0	0
1	0	1
1	1	0
10	1	1

Misal:

$$\begin{array}{r} 6 \to 110 \\ + \\ \underline{9 \to 1001} \\ 15 \to 1111 \end{array} \qquad \begin{array}{r} 20 \to 10100 \\ - \\ \underline{15 \to 1111} \\ 5 \to 0101 \end{array} \qquad \begin{array}{r} 6 \to 110 \\ \times \\ \underline{3 \to 011} \\ 18 \to 110 \\ + \\ \underline{110} \\ 10010 \end{array}$$

Bölmə əməli isə, bölənin, bölünəndən ardıcıl olaraq çıxılması ilə əvəz olunur. *Məsələn:*

$$\begin{array}{r|l} 101010 & 101 \\ \underline{101} & 1000,011 \\ 00001000 & \\ \underline{101} & \\ 0110 & \\ \underline{101} & \\ 001 \text{ və s.} & \end{array}$$

Yoxlama:

$1000,011 \times 101 + 0,001 = 101001,111 + 0,001 = 101010$

1.4. EHM-də informasiyanın verilməsi. EHM-in iş prinsipi

İxtiyari informasiyanı kiçik elementar hissələrə bölmək olar. Məsələn, kitabdakı ixtiyari mətn hərf və digər simvollardan ibarətdir. Burada hərf – mətni informasiyanın elementar hissəsidir.

İnformasiyanın ixtiyari elementar hissəsi ədəd şəklində ifadə edilirsə, onda belə informasiyaya kodlaşdırılmış informasiya deyilir. Belə ədədlər isə kodlar adlanır. Əgər mətndəki hər bir hərfi kodlaşdırsaq, məsələn, bu hərfin əlifbadakı sıra nömrəsi ilə, onda bütün mətni kodlaşdırmaq olar.

EHM-lər isə informasiyanı yalnız kodlaşdırılmış şəkildə tədqiq edir. İnformasiyanın verilməsi üçün isə EHM-də yalnız iki simvoldan 0 və 1 rəqəmlərindən istifadə edilir. İnformasiyanın bu cür ifadəsi EHM-lərin texniki xüsusiyyətləri ilə izah olunur. Belə ki, kompüterlərdə ikilik say sistemindən ona görə istifadə olunur ki, bu sistemin rəqəmlərini elektrik siqnalları (cərəyanı) ilə çox asanlıqla ifadə etmək olur: 0 – siqnal yoxdur və 1 – siqnal var.

Minimal informasiya vahidi 1 bit (bit) adlanır. Bir bit informasiya - bu ikilik sistemin rəqəmlərindən biri ya 0, ya da 1 olur. Bu çox kiçik informasiya vahididir, buna görə də kompüterlərdə informasiyanın elementar hissələrinin tədqiqi üçün daha böyük vahiddən - *bayt* (byte) vahidindən istifadə olunur. Bir bayt səkkiz bitə bərabərdir. Bir baytda 256 simvoldan ($256 = 2^8$) birini kodlaşdırmaq olar.

EHM-in yaddaşının tutumu baytlarla ölçülür, lakin çox vaxt digər ölçü vahidlərindən istifadə olunur. Bunlar: kilobayt (1 *Kbayt*=1024 *bayt*); meqabayt (1 *Mbayt*=1024 *Kbayt*), qiqabayt (1 *Qbayt*=1024 *Mbayt*). Əgər bir səhifə mətndə təqribən 2500 işarə varsa, onda 1 *Mbayt* - təqribən 400 səhifə, 1 *Qbayt* isə 400 min səhifə mətn deməkdir.

Bir bayt - informasiya vahidi olmaqla yanaşı, həm də EHM yaddaşının elementar oyuğudur. EHM yaddaşı bu cür oyuqlar ardıcıllığından ibarətdir. Hər bir oyuğun (*bayt*) öz ünvanı - oyuğun nömrəsi var. EHM-in prosessoru informasiyanı tədqiq edərkən, o ünvana görə yaddaşda lazımi oyuğu tapır, oradakı informasiyanı oxuyub, lazımi əməliyyatları aparıb, alınan nəticəni digər bir oyuğa yazır.

Kompüterin işini ümumi şəkildə aşağıdakı kimi təsvir etmək olar. Əvvəlcə hər hansı bir xarici qurğudan istifadə etməklə, məsələn klaviatura vasitəsilə kompüterin yaddaşına proqram daxil edilir. Kompüterin idarəedici qurğusu, proqramın birinci əmri yerləşən yaddaşın oyuğundan bu əmri oxuyub, onun yerinə yetirilməsini təmin edir. Bu əmr isə cəbri və ya məntiqi əməliyyatların yerinə yetirilməsi, bu əməliyyatların yerinə yetirilməsi üçün verilənlərin yaddaşdan oxunması, bu əməliyyatların nəticələrinin yaddaşa yazılması, verilənlərin xarici qurğudan yaddaşa verilməsi və verilənlərin yaddaşdan xarici qurğulara verilməsi kimi işlərin görülməsini təmin edə bilər. Adətən, bir əmr yerinə yetirildikdən sonra idarəedici qurğu yaddaşın bu oyuqdan sonra gələn oyuqdakı əmri yerinə yetirməyə başlayır. Bu qayda yalnız idarəetmənin keçid əmrləri ilə dəyişdirilə bilər. Bu əmrlər idarəedici qurğuya, proqramın yerinə yetirilməsini, yaddaşın hər hansı başqa bir oyuğunda yerləşən əmrdən başlayaraq davam etdirilməsi haqqında göstəriş verə bilərlər.

Beləliklə, idarəedici qurğu proqramın göstərişlərini insanın köməyi olmadan avtomatik olaraq yerinə yetirir. O, kompüterin operativ yaddaşı və xarici qurğuları ilə əlaqə saxlayıb, onlardan informasiya alıb, informasiya verə bilir. Yerinə yetirilmiş proqramın bütün nəticələri idarəedici qurğu tərəfindən xarici qurğulara verilir və bundan sonra kompüter xarici qurğulardan yeni əmrlərin gözləmə rejiminə keçir.

1.5. EHM-in arxitekturası

Konstruksiya etibarilə fərdi EHM-lər aşağıdakı formalarda ola bilər: 1) stolüstü (Desktop); 2) Bloknot (Notebook) 3) Superbloknot (Sub-Notebook); 4) Cibə qoyulan ölçülü (Palmtop); 5) Elektron yazı kitabçası.

Fərdi EHM-lərin aşağıdakı imkanları var:

1) Məhsuldarlığı 1 saniyədə 1 milyondan çox əməliyyat (takt tezliyi 1000 MeqoHers)

2) Operativ yadda saxlama qurğusunun həcmi 128 *Mbayt*

3) Daimi yadda saxlama qurğusunun həcmi 47 *Qbayt*.

Fərdi EHM-lər aşağıdakı əsas (standart) qurğulardan ibarətdir:

1) Sistem bloku

2) Monitor

3) Klaviatura.

Bundan əlavə EHM-ə periferik adlanan aşağıdakı qrup qurğuları qoşmaq olar:

1) daxiletmə qurğuları: skaner, rəqəmsal fotokamera, qrafik planşet

2) xaricetmə qurğuları: printer, plotter

3) xarici yaddasaxlama qurğuları: maqnit və lazer disklərlə iş üçün diskovodlar, strimer

4) idarəetmə qurğuları: siçan, trekbol, djoystik

5) həm daxiletmə, həm də xaricetmə funksiyalarını yerinə yetirən qurğular: modem, şəbəkə platası, səs platası.

Sistem blokunun tərkibinə sistem platası, elastik maqnit disklər üçün diskovodlar, bərk maqnit disk (vinçestr), daxiletmə və xaricetmə üçün giriş-çıxış oyuqları (portlar), elektrik təminatı bloku, səs dinamiki aiddir. Sistem platasında isə mikroprosessor, soprosessor (olmaya da bilər), operativ yaddaş modulları, tez yadda saxlama mikrosxemləri (KEŞ-yaddaş), giriş-çıxış sistemlərinin mikrosxemləri (BİOS), diskovodun, monitorun və digər qurğuların işini idarə edən adapterlər və kontrollerlər, sistem platasının müxtəlif elementlərini əlaqələndirən rabitə kanallarının kompleksi olur.

Mikroprosessor kompüterin əsas elementidir, bütün hesabi və məntiqi əməliyyatları yerinə yetirən, verilmiş proqram üzrə qoyulan məsələnin bütün həll prosesini idarə edən mikrosxemdir. Mikroprosessorun əsas xarakteristikaları: onun emal edə biləcəyi ədədlərdəki rəqəmlərin sayını müəyyən edən qiymət (bu qiymət 8, 16, 32 (köhnə model EHM-lər üçün) və ya 64 ola bilər) və takt tezliyidir – bu prosessorun bir saniyədə yerinə yetirə biləcəyi taktların sayıdır (takt – prosessorun elementar əməliyyatın yerinə yetirilməsinə sərf etdiyi zamandır). Müasir EHM-də bu tezlik 1000 *MHersdir*. Yadda saxlama qurğuları proqramların və verilənlərin qorunub saxlanmasını təmin edir və aşağıdakı tiplərə bölünür: operativ yadda saxlama qurğusu və ya operativ yaddaş, KEŞ-yaddaş, daimi yadda saxlama qurğusu və ya daimi yaddaş, xarici yadda saxlama qurğuları.

Operativ yaddaşda EHM-də aparılan cari əməliyyatlar

yerinə yetirilir. Bu yaddaşdakı informasiya yalnız EHM-in işlədiyi müddət ərzində saxlanılır. Müasir EHM-də operativ yaddaşın həcmi 2 *Qbayta* çata bilər, lakin praktik iş üçün 128 *Mbayt* yaddaş həcmi də kifayətdir.

KEŞ-yaddaş EHM-min yaddaşında aparılan əməliyyatların sürətləndirilməsini təmin edir. KEŞ-yaddaşa prosessorun cari anda üzərində iş apardığı operativ yaddaş hissəsi köçürülür. KEŞ-yaddaş 2 *Mbayta* qədər həcmə malik ola bilər.

Daimi yaddaş bura yazılmış verilənlərin və proqramların oxunması üçün nəzərdə tutulub. IBM tipli kompüterlərdə daimi yadda saxlama qurğusu ayrıca mikrosxem kimi verilir və burada əməliyyat hissəsinin bir bölməsi – daxiletmə və xaricetmənin baza sistemi (BIOS) saxlanılır.

Xarici yadda saxlama qurğusu böyük həcmli informasiyanın əsas yaddaşdan xaricdə uzun müddətə saxlanılmasını təmin edir. Yaddaş «xarici» olsa da onun çox hissəsi sistem blokunun korpusunda yerləşir. Xarici yaddaşa aşağıdakılar aiddir:

1) elastik maqnit disklərdə toplanan informasiya
2) bərk maqnit diskdə (vinçestrdə) toplanan informasiya
3) dəyişdirilə bilən bərk maqnit disklərdə toplanan informasiya
4) lazer kompakt disklərlə iş üçün diskovodlar
5) strimerlər

(Diskovod - elastik, lazer maqnit disklərdəki informasiya ilə işləməyə imkan verən qurğulardır).

Elastik maqnit disklərdə toplanan informasiya daşıyıcıları dəyişdirilə bilən maqnit disklər – disketlərdir. Disketlərdən EHM-lər arasında informasiya mübadiləsi üçün, informasiyanın EHM-dən kənarda saxlamaq üçün istifadə edilir.

Hal-hazırda adi disketlərdən və floppi disketlərdən istifadə olunur. Adi disket 3,5 *dyum* (89 *mm*) ölçüsünə və 1440 *Kbayt* informasiya daşımaq imkanına malikdir, disket bərk plastik korpusdan ibarətdir. Floppi disketlər də adi disketlərə oxşardır, lakin 21 *Mbayta* qədər informasiya daşıya bilir. Son zamanlar HIFT (Sony firması) tipli disket və diskovodlar tətbiq olunur. Bu disketlər də 3,5 *dyum* ölçülüdür, lakin 200 *Mbayt* həcmdə informasiya daşıya bilir. HIFT diskovodu 1,44 *Mbaytlı* disketlər üçün də uyğunlaşdırılıb.

Dəyişdirilməyən bərk maqnit disk (vinçestr) EHM-lə iş zamanı daim istifadə olunan informasiyanın – əməliyyat sisteminin, proqram örtüyü proqramlarının və s. uzun müddətə saxlanmasını təmin edir. Vinçestr sistem blokunun korpusunda yerləşdirilir. Müasir IBM tipli fərdi EHM-də vinçestrin yaddaş həcmi ən azı 20 *Qbayt* ən çoxu 120 *Qbayt* və daha artıq olur. Dəyişdirilə bilən bərk maqnit disklər disketlər kimi informasiyanın saxlanması və daşınmasını təmin edir. Bu cür qurğuların bir neçə tipi mövcuddur, ən çox yayılmış qurğular dəyişdirilə bilən diskinin həcmi uyğun olaraq 200 *Mbayt* və 2 *Qbayt* olan Zip və JAZ adlanan diskovodlardır.

Lazer kompakt disklərlə iş üçün diskovodlar müxtəlif tip kompakt disklərdən informasiyanın oxunmasını təmin edir. Hal-hazırda istifadə olunan disklərin əsas tipləri aşağıdakılardır:

1) CD-ROM (Compact Disk – Read Only Memory) – kompakt disklərin yalnız oxunmasını təmin edir. Kompakt disklərə informasiya istehsalçı tərəfindən yalnız bir dəfə yazıla bilir, disk isə çoxlu sayda oxuna bilər. CD-ROM-un ölçüsü 120 *mm*, yaddaş həcmi ən azı 680 *Mbayt* olur.

2) CD-R disklər. Bu cür disklərə informasiya istifadəçi tərəfindən xüsusi yazı qurğusu (CD-WRITER) ilə bir dəfə yazıla bilər. Diski ixtiyari CD-ROM ilə çoxlu sayda oxumaq olur.

3) CD-RW disklər. Bu cür disklərə informasiya yazan diskovodla istifadəçi tərəfindən çoxlu sayda yazılıb, silinə bilər.

4) DVD disklər. Bu disklərin digər lazer disklərdən fərqi informasiyanın daha sıx yerləşdirilə bilməsindədir. Ən çox istifadə edilən DVD diskin ölçüləri 120 *mm*, informasiya tutumu 4,38 *Qbaytdır*. Bu disklərin tutumu 14 *Qbayta* qədər ola bilər. İki tip disklər mövcuddur: DVD-ROM – yalnız informasiyanı oxumaq üçün, DVD-RAM isə informasiyanı həm oxumaq, həm də yazmaq üçün istifadə olunur. DVD disklərin oxunması üçün xüsusi qurğudan istifadə olunur, bu qurğu adətən CD-ROM-un da oxunmasını təmin edir.

Strimerlər informasiyanın maqnit lentə köçürülməsini təmin edən qurğudur. Bu qurğu informasiyanı qabaqcadan sıxmaqla maqnit lentə köçürür. Bu qurğudan bərk diskdə olan vacib informasiyanın maqnit lentə köçürüb saxlamaq üçün istifadə olunur. Strimerin kasetlərində bir neçə qiqabayt həcmin-

də informasiya saxlamaq olur.

Monitor EHM-ə daxil edilən və ondan alınan həm mətni, həm də qrafik informasiyanın zahiri ifadəsini təmin edən elektron qurğudur. Monitorlar maye kristallar əsasında (LCD displey) və ya elektron-şüa əsaslı olur. Monitor ekrandan və onun idarəetmə sistemindən ibarətdir. Bu sistemin ən vacib hissəsi videoadapter – sistem blokunda videoplatada yerləşdirilir. Bu qurğu monitorun imkanlarını müəyyən edir. Monitorun ekranı piksel adlanan elementar hissələrdən ibarətdir. Monitorun iki iş rejimi var: qrafik iş rejimi və mətn iş rejimi.

Qrafik iş rejimi ekrana qrafiklərin, şəkillərin və s. çıxarılmasını təmin edir. Bu rejimdə ekranın hər bir nöqtəsini idarə etmək, onlara ixtiyari rəng vermək, onlardan müxtəlif təsvirlər qurmaq olur.

Mətn iş rejimi mətnlərin ekrana verilməsi üçün nəzərdə tutulub.

Monitorların əsas xarakteristikaları aşağıdakılardır:

1) Monitorların tipləri: CGA, EGA, VGA, SVGA və s.

2) Videoplataların tipləri: CGA, EGA, VGA, SVGA və s.

3) Rəngin dərinliyi – ekrana eyni zamanda çıxarıla bilən rənglərin sayı ilə müəyyən edilir və videoyaddaşın ölçüsü ilə təyin edilir. SVGA adapteri 128 Mbayt ölçülü videoyaddaşa malikdir ki, bu da High Color iş rejimini (25536 rəng verilə bilir) və True Color iş rejimini (16,8 milyon rəng verilə bilir) təmin edir. Qeyd edək ki, ixtiyari rəngli monitordan monoxrom monitor kimi də istifadə etmək olur.

4) Ekranın ölçüləri – 14, 15, 17, 19, 21, 29 dyum (diaqonal üzrə) ola bilir.

5) Ekrana çıxarılan nöqtələrin sayı məsələn, 640×200 ola bilər. Eyni bir monitorda müxtəlif iş rejimlərində ekrana müxtəlif sayda nöqtələr çıxarıla bilir. *Məsələn*, VGA tipli monitorda ekrana 640×480 və 800×600 sayda nöqtə, SVGA tipli monitorda ekrana 1024×768 və 1280×1024 sayda nöqtə çıxarıla bilir.

6) Nöqtənin (pikselin) ölçüləri. Bu kəmiyyət kiçik olduqca ekranda təsvir bir o qədər keyfiyyətli alınır. Normal ölçü 0,25 *mm*-dir.

7) Kadrların dəyişmə tezliyi. Bu tezlik böyük olduqca istifadəçinin gözləri bir o qədər az yorulur. Normal tezlik 70 *Hersdir*.

Klaviatura informasiyanın EHM-ə daxil edilməsi və onun

işinin idarə edilməsi üçün nəzərdə tutulub. Windows sistemində iş üçün nəzərdə tutulmuş ən geniş yayılmış klaviatura 104 düyməlidir (bura o cümlədən Windows sisteminin pəncərələrinin təsviri olan Win kimi işarə edəcəyimiz 2 düymə də aiddir). Hərf, rəqəm və digər işarələr klaviaturanın əsas hissəsini təşkil edən hərf-rəqəm düymələri ilə daxil edilir. Bundan əlavə klaviaturada aşağıdakı xüsusi düymələr də var:

Caps Lock – bu düymə əlifbanın böyük hərflərinin daxil edilməsi rejimini verir. Bu düymənin təkrar sıxılması ilə böyük hərflərin daxil edillməsi rejimi ləğv olunur.

Enter düyməsi sətrin daxil edilməsinin sona çatdırılması üçün, yaxud hər hansı proqramın bu və ya digər sualına təsdiq edici cavab vermək üçün istifadə olunur.

Back Space – (*Enter* düyməsindən yuxarıda yerləşən və üzərində soldan sağa istiqamətləndirilmiş ox işarəsi olan düymə) kursordan (kursor – ekranda cari anda klaviaturadan daxil ediləcək simvolun yerləşəcəyi yeri bildirən göstəricidir) solda yerləşən simvolu ləğv edir.

Del (Delete) – kursordan sağda duran simvolu ləğv edir.

Ins (Insert) – simvolların daxil edilməsi rejimlərinin dəyişdirilməsi üçün nəzərdə tutulub.

“↑”, “↓”, “→”, “←” düymələri kursoru ekranda oxların göstərdiyi istiqamətlərdə hərəkət etdirmək üçün istifadə olunur.

Home (End) – kursoru mətndəki sətrin əvvəlinə (sonuna) keçirir.

Pg Up (Pg Dn) – kursoru mətn üzrə ekranda yerləşə biləcək sətrlər həcmində yuxarı (aşağı) istiqamətlərdə hərəkət etdirir.

Num Lock düyməsi klaviaturanın sağ kənarında yerləşən düymələr blokundan rəqəmlərin daxil edilməsi üçün istifadə rejimini daxil edib və ləğv etmək üçün nəzərdə tutulub. Bu düymələrdən eləcə də kursorun işini idarə edən yuxarıda qeyd etdiyimiz düymələr kimi də istifadə etmək olar.

Esc (Escape) düyməsindən adətən hər hansı bir əməliyyatı ləğv etmək, cari iş rejimindən, proqramdan çıxmaq üçün istifadə olunur.

F1–F12 – funksional düymələri, cari proqramın təyin etdiyi müxtəlif xüsusi əməliyyatların yerinə yetirilməsi üçün nəzər-

də tutulub.

Ctrl, Alt və Shift düymələri klaviaturanın digər düymələri ilə kombinasiyada işlədilir və bu düymələrin təyinatını dəyişdirmək üçün nəzərdə tutulub.

Space bar (probel) – boş yer daxil etmək üçün istifadə olunan və klaviaturanın aşağı hissəsində yerləşən, üzərində heç bir yazı olmmayan düymədir.

Pause – proqramın yerinə yetirilməsini müvəqqəti dayandıran düymədir.

Bəzi düymə kombinasiyalarını da qeyd edək:

Ctrl+Break kombinasiyası (MS DOS-da) yerinə yetirilən proqram və ya əmrin sona çatdırılmasını təmin edir.

Ctrl+Alt+Del kombinasiyası EHM-in əməliyyat sisteminin yenidən yüklənməsinə gətirir.

Shift+Print Screen ekrandakı informasiyanın çap qurğusu ilə çap olunmasını təmin edən kombinasiyadır.

Ctrl+Num Lock kombinasiyası proqramın yerinə yetirilməsini dayandırır. Proqramın işini davam etdirmək üçün ixtiyari düyməni basmaq kifayət edir.

Klaviatura ilə yanaşı EHM-in işini idarə edən qurğulara siçanı, trekbolu, kontakt panelini, coystiki də aid etmək olar.

Siçan – portativ manipulyator olub, hamar səth üzrə hərəkət etdirildikdə, kursorun da ekranda həmin istiqamətdə hərəkətini təmin edir. Bu qurğu 2 və ya 3 düyməsi olan bir qutu formasında olub, adətən EHM-lə xüsusi kabel ilə əlaqələnir. *Trekbol* – əksinə çevrilmiş siçan qurğusuna bənzəyir. Onu hərəkət etdirmirlər, bunun əvəzinə ondakı kürəciyi hərəkət etdirirlər. Kontakt paneli – EHM-lə kabellə əlaqələndirilmiş, adətən 76×70 *mm* ölçülü qutudur. Qutunun qapağı ekrandan ibarətdir. Kursoru ekran boyunca hərəkət etdirmək üçün istifadəçi barmağını həmin ekran üzrə hərəkət etdirir. Barmağın bu ekrana vurulması, siçanın düyməsinin sıxılması effektini verir. Coystik – oyun proqramlarında istifadə olunur və ekranda kursorun idarə olunması üçün tətbiq olunur.

İndi isə xaricetmə qurğularına baxaq.

Printer – informasiyanın kağız üzərinə köçürülməsini təmin edən qurğudur. Bütün müasir printerlər, kağız üzərinə həm mətn, həm şəkil, həm də qrafiklər çıxara bilir. Hal-hazırda IBM PC tipli

fərdi EHM-lə uyğunlaşdırılmış kompütelərdə istifadə edilə bilən bir neçə min sayda müxtəlif model printer mövcuddur. Bunlar əsasən matris, şırnaq və lazer tipli printerlərdir. Matrisli printerlərdə çap edən başlıqda metal iynələr ardıcıllığı yerləşir, başlıq çap olunan sətr boyunca hərəkət edir və iynələr lazım olan anda rəngləyici lent üzərindən kağıza zərbələr endirir. Bununla da kağız üzərində simvol və təsvirlər formalaşdırılır. Bu tipli printerlərdə çap sürəti böyük olmur (dəqiqədə 0,5 – 2 səhifə), çapın keyfiyyəti də böyük deyil (360 *dpi* (*dpi* – 1 *düym* kağız səthinə düşən nötələrin sayıdır)), lakin bu printerlər ucuz başa gəlir və istifadədə sadədir. Şırnaqlı printerlərdə çap edən başlıqda iynələr mikroskopik borucuqlarla, rəngləyici lent isə rəng konteynerləri ilə əvəz olunub. Rəngli printerlərdə əsas rənglərin sayına uyğun olaraq 4 belə konteyner («kartridj») ola bilər. Buradakı borucuqlardan rəng mikro damlalarla kağız üzərinə püskürdülür və beləliklə simvol və təsvirlər formalaşdırılır. Bu cür printerlərin təsvir keyfiyyəti 600 *dpi* qədər olur. Lazer tipli printerlərdə rəngləyici tozdan (toner) istifadə olunur. Çap edən başlıq rolunu xüsusi lazer sistemi oynayır. Bu sistem təsvirin proyeksiyasını fırlanan işığa həssas baraban üzərinə verir, buradan isə təsvir toner vasitəsilə kağız üzərinə köçürülür. Burada təsvir keyfiyyəti 1200 *dpi* qədər olur.

Plotter – mürəkkəb sxemlərin, qrafiklərin, keyfiyyətli rəngli təsvirlərin kağız üzərinə çıxarılması üçün nəzərdə tutlub. Ondan adətən proyekt və konstruktor bürolarında, reklamla məşğul olan təşkilatlarda istifadə olunur.

İndi isə daxiletmə qurğularını qeyd edək.

Skaner – kağız üzərindəki ixtiyari informasiyanın oxunub, EHM-ə daxil edilməsini təmin edir. Ağ-qara və rəngli simvollarla iş üçün nəzərdə tutulan skanerlər mövcuddur. Skaner təsviri EHM-ə koordinatları və rəngləri qeyd olunmuş nöqtələr çoxluğu kimi daxil edir, sonra bu verilənlər əsasında ekrana təsvir çıxardılır. Skanerlə mətnləri oxuyub, onlarla mətn kimi işləmək tələb olunduqda, təsviri mətn kimi qəbul etməyə imkan verən xüsusi proqramlardan istifadə olunur. İngilis, rus və digər dillərdə olan mətnlər üçün uyğun proqramlar mövcuddur. Onlar mətnləri, məsələn, MS Word mətn redaktoruna keçirməyə imkan verir. Qeyd edək ki, hətta əlyazma mətnlərini oxumağa qadir olan

proqramlar mövcuddur. Rəqəmsal fotokamera ilə çəkiliş adi qaydalarla aparılır, kamera EHM-ə qoşulduqda çəkilən kadr ekrana çıxarılır və printerlə çap oluna bilər.

Qrafik planşet xüsusi örtüyü olan qurğudur, onun üzərində xüsusi qrafik qələmlə yazmaq olur. Bütün yazılanlar EHM-ə təsvir şəklində daxil edilir.

Həm daxiletmə, həm də xaricetmə funksiyaları olan qurğulara baxaq.

Modem EHM-i qlobal kompüter şəbəkələrinə qoşulmasını təmin edən qurğudur.

Şəbəkə adapteri – EHM-in lokal kompüter şəbəkəsinin tərkibində işləməsini təmin edir.

Səs platası səs informasiyasının rəqəmsal formaya çevirməyə imkan yaradır.

II FƏSİL

KOMPÜTER ŞƏBƏKƏLƏRİ

2.1. Lokal və qlobal kompüter şəbəkələri

Kompüter şəbəkələri sənayesi inkişaf etmiş ölkələrin informasiya infrastrukturunun əsas komponentlərindən biridir. Kompüter şəbəkələri müasir sivilizasiyanın iki əsas elmi-texniki sahəsinin – kompüter və kommunikasiya texnologiyalarının – inkişafının məntiqi nəticəsidir. Bir tərəfdən, kompüter şəbəkələri paylanmış hesablama sistemlərinin xüsusi halıdır. Burada kompüterlər qrupu qarşılıqlı məsələləri yerinə yetirərək verilənlərin mübadiləsini avtomatik rejimdə aparırlar. Digər tərəfdən, kompüter şəbəkələrinə informasiyanın uzaq məsafələrə ötürülməsi vasitəsi kimi də baxmaq olar. Bunun üçün müxtəlif telekommunikasiya sistemlərində geniş yayılmış verilənlərin kodlaşdırılması və multipleksləşdirilməsi üsulları tətbiq olunur.

Kompüter şəbəkələri böyük sürətlə yayılırlar. İyirmi il bundan əvvəl kompüter şəbəkələrinə az sayda insan müraciət edə bilərdi. Hal-hazırda kompüterlərarası verilənlərin mübadiləsi gündəlik həyatımızın ayrılmaz hissəsidir. Şəbəkələr dövlət orqanları və hərbi təşkilatlarda da istifadə olunur. Sənaye və ticarət müəssisələri, inzibati və maliyyə təşkilatları arasında gərgin informasiya mübadiləsi sərfəli bazar iqtisadiyyatı və demokratik sosial mexanizmlərin dəstəklənməsi üçün vacibdir. Qeyd etmək lazımdır ki, terminal və kompüterlərarası informasiyanın ötürülməsi, iki və ya bir neçə kompüterarası əlaqə, kompüterlərin istifadəsi ilə telemetriya, verilənlərin teleemalı texniki olaraq çoxdan həyata keçirilib və praktiki olaraq kompüterlərin yaranması anından istifadə olunur. Bu halda informasiyanın ötürülməsi ayrı-ayrı kompüterlərin birgə işinin təşkili, və məsələnin bir neçə kompüter vasitəsi ilə həllini, resursların birgə istifadəsini və digər problemlərin həllini mümkün edir.

Kompüter şəbəkələrin təsnifatı üçün müxtəlif əlamətlərdən istifadə olunur. Lakin əksər hallarda onları ərazi əlamətinə görə, yəni şəbəkə əhatə etdiyi ərazinin ölçüsünə görə, növlərə ayırırlar. Şəbəkələrin iki əsas kateqoriyası mövcuddur – lokal və qlobal

şəbəkələr. Onlar arasında olan fərq sadədir: lokal şəbəkələr bir-biri ilə yaxın məsafədə yerləşən qurğuların qarşılıqlı əlaqəsinin təşkili üçün istifadə olunurlar; qlobal şəbəkələr isə coğrafi olaraq paylanmış kompüter və ya lokal şəbəkələrin qarşılıqlı əlaqələrinin təşkili üçün istifadə olunurlar. Üçüncü kateqoriyaya regional şəbəkələr daxildir. Bu şəbəkələr lokal şəbəkələrin standartları ilə təyin olunduqlarına baxmayaraq, daha çox qlobal şəbəkələrə oxşayırlar. Regional şəbəkələr nisbətən az tanınır və nadir hallarda istifadə olunurlar.

Lokal şəbəkələrə (*Local Area Networks*, LAN) adətən radiusu 1-2 *km*-dən çox olmayan kiçik məsafədə yerləşən kompüter şəbəkələrini aid edirlər. Lakin, qeyd etmək lazımdır ki, bəzi hallarda lokal şəbəkənin əhatə etdiyi məsafə daha çox, məsələn onlarla kilometr, ola bilər. Ümumi halda lokal şəbəkə bir təşkilata aid olan kommunikasiya sistemidir. Lokal şəbəkələrdə, məsafələr qısa olduğu üçün bahalı və keyfiyyətli rabitə xətlərinin istifadəsi mümkündür. Belə rabitə xətləri informasiyanın ötürülməsinin sadə üsullarını istifadə etməklə verilənlər mübadiləsinin sürətini 100 *Mbit/s*-yə qədər çatdırmağa imkan yaradırlar. Bununla əlaqədar olaraq, lokal şəbəkələrin təqdim etdikləri xidmətlər müxtəlifliyi ilə fərqlənirlər və adətən online rejimində realizasiyanı nəzərdə tuturlar.

Lokal şəbəkə ilə ötürmənin sürəti ən geniş yayılmış kompüterlərin cəldliyi artdıqca mütləq artmalıdır. Bu yaxınlara kimi sürəti 1 – 10 *Mbit/s* ilə verilənlərin mübadiləsi normal hesab olunurdu. Hal-hazırda 100 *Mbit/s* sürəti ilə işləyən şəbəkə orta sürətli hesab olunur və 1000 *Mbit/s* sürətli şəbəkələr üçün vasitələr işlənir.

Lokal şəbəkələrin digər şəbəkələrdən fərqi mübadilənin yüksək sürətli olmasıdır. Lakin bu yeganə fərq deyil, digər daha vacib amillər də mövcuddur. Misal üçün, ötürmə səhvlərinin çox aşağı səviyyəsi prinsipial xarakter daşıyır. Sürətlə ötürülən, lakin səhvlərlə təhrif olunmuş informasiya mənasızdır və onun yenidən ötürülməsi tələb olunur. Ona görə də lokal şəbəkələr xüsusi olaraq çəkilmiş keyfiyyətli rabitə xətlərini tələb edir.

Şəbəkənin həddindən artıq yüklənməsi, yəni mübadilə intensivliyinin böyük olması, xüsusiyyəti də prinsipial məna daşıyır.

Əgər şəbəkədə istifadə olunan mübadilənin idarəolunması mexanizmi səmərəli deyilsə, onda kompüterlər ötürməni yerinə yetirmək üçün növbədə xeyli vaxt gözləməlidirlər.

Mübadilənin idarəolunmasının istənilən mexanizmi şəbəkəyə qoşula biləcək kompüterlərin sayı əvvəlcədən məlum olduqda zəmanətli işləyə bilər. Abunəçilərin sayı əvvəlcədən nəzərə alınmamış çox sayda olduqda, istənilən mexanizm, şəbəkənin yüklənməsi nəticəsində sıradan çıxa bilər.

Beləliklə, lokal şəbəkənin aşağıdakı fərqləndirici əlamətlərini qeyd etmək olar:

- yüksək ötürmə sürəti, yüksək keçirtmə qabiliyyəti;
- ötürmə səhvlərinin səviyyəsinin aşağı olması, yəni yüksək səviyyəli rabitə kanalları;
- səmərəli, tez təsirli mübadilənin idarə olunması mexanizmi;
- şəbəkəyə qoşulan kompüterlərin sayı dəqiq təyin edilmiş və məhdud olmalıdır.

Lokal şəbəkə ilə müxtəlif rəqəmsal informasiya ötürülə bilər: verilənlər, təsvirlər, telefon danışıqları, elektron məktublar və s. Əksər hallarda lokal şəbəkələr disk mühiti, çap qurğuları və qlobal şəbəkəyə çıxış kimi resursların bölünməsi üçün istifadə olunurlar. Bu resurslar müxtəlif növlu kompüterlərarası informasiya mübadiləsini yerinə yetirməyə imkan verirlər. Şəbəkənin abunəçiləri kimi tək kompüterlər deyil, çap qurğusu, skaner, plotter kimi digər qurğular da çıxış edə bilər. Lokal şəbəkələr şəbəkənin bütün kompüterlərində paralel hesablama sistemini təşkil etməyə imkan verir. Bu isə mürəkkəb riyazi məsələlərin həllini sürətləndirməyə imkan yaradır.

Lokal şəbəkələrin bir sıra çatışmamazlıqları da mövcuddur. Avadanlıq və şəbəkə proqram təminatının alınmasına, birləşdirici kabellərin çəkilməsi və işçi heyyətin öyrənilməsinə çəkilən əlavə maddi xərclərdən başqa, şəbəkə administratoru (inzibatçısı) adlanan mütəxəssis də lazımdır. Şəbəkələr kompüterlərin yerlərinin dəyişməsini məhdudlaşdırırlar, belə ki, bu zaman birləşdirici kabellərin yenidən çəkilməsi tələb olunacaq. Bundan başqa, şəbəkələr kompüter viruslarının yayılması üçün "gözəl" bir mühitdir, ona görə də, mühafizə məsələlərinə daha çox diqqət yetirilməlidir.

Qlobal şəbəkələr (*Wide Area Networks*, WAN) müxtəlif şəhər və ölkələrdə yerləşə bilən, məsafəyə görə səpələnmiş kompüterləri birləşdirirlər. Bu şəbəkələr çox sayda son abunəçilərə öz xidmətlərini təqdim etmək üçün istifadə olunurlar. Rabitə kanallarının uzunluğunun böyük olması xeyli məsrəflər tələb edir. Buraya kabellərin alınması və çəkilməsinə, kanalın lazımi keçirtmə qabiliyyətini təmin edən kommunikasiya və aralıq gücləndirici avadanlığa sərf olunan xərclər, şəbəkənin avadanlığının daimi işlək vəziyyətdə saxlanılması üçün istismar xərcləri daxildir. Keyfiyyətli rabitə xətlərinin çəkilməsi baha başa gəldiyi üçün qlobal şəbəkələrdə artıq mövcud olan, əvvəlcədən digər məqsədlər üçün nəzərdə tutulmuş, rabitə xətlərindən istifadə olunur. Misal üçün, qlobal şəbəkələrin əksəriyyəti ümumi təyinatlı telefon və teleqraf kanalları əsasında qurulurlar. Qlobal şəbəkələr onlardan çox əvvəl yaranmış telefon şəbəkələrindən çox şeyi irsən qəbul edirlər. İlk qlobal kompüter şəbəkələrinin yaradılmasının əsas nəticəsi uzun müddət telefon şəbəkələrində uğurla istifadə olunan kanalların kommutasiyası prinsipindən imtina edilməsidir.

Qlobal kompüter şəbəkələrin ənənəvi abunəçiləri müxtəlif şəhər və ölkələrdə yerləşən müəssisələrin lokal şəbəkələridir. Qlobal şəbəkələrin xidmətlərindən ayrı-ayrı kompüterlər də istifadə edilirlər. Adətən qlobal kompüter şəbəkələri iri telekommunikasiya kompaniyaları tərəfindən abunəçilərə pullu xidmətlərin göstərilməsi üçün yaradılır. Belə şəbəkələri ictimayi şəbəkələr adlandırırlar. Şəbəkənin normal işini dəstəkləyən kompaniya şəbəkənin operatoru adlanır. Xidməti təklif edən kompaniyaya adətən *provayder* deyilir.

Hal-hazırda müxtəliflik və xidmətlərin keyfiyyətinə görə qlobal şəbəkələr, uzun müddət bu sahədə liderlik edən lokal şəbəkələrə çatıblar.

Şəhər şəbəkələri (*Metropolitan Area Networks*, MAN) az yayılmış şəbəkələr növüdür. Bu şəbəkələr nisbətən yaxın vaxtlarda yaranıblar. Onlar böyük şəhərin ərazisinə xidmət etmək üçün nəzərdə tutulub. Şəhər şəbəkələri lokal və qlobal şəbəkələr arasında aralıq yer tuturlar. Bu şəbəkələr şəhər miqyasında lokal şəbəkələrin birləşməsi və lokal şəbəkələrin qlobal şəbəkələrə qoşulması üçün nəzərdə tutulublar. Əvvəlcə bu şəbəkələr verilənlərin ötürülməsi üçün nəzərdə tutulmuşdu, hal-hazırda onlar videokonfrans, səs və mətnin inteqral ötürülməsi kimi xidmətləri dəstəkləyir.

Lokal şəbəkələri qlobal şəbəkələrdən ayıran əsas fərqlərə nəzər salaq. 80-ci illərin sonlarında lokal və qlobal şəbəkələr arasında olan fərqlər daha aydın üzə çıxdı.

- **Rabitə xətlərinin uzunluğu və keyfiyyyəti.** Lokal şəbəkələr qlobal şəbəkələrdən qovşaqlararası kiçik məsafə ilə fərqlənir. Bu isə lokal şəbəkələrdə daha keyfiyyətli rabitə xətlərinin istifadəsinə imkan verir.
- **Verilənlərin ötürmə üsullarının mürəkkəbliyi.** Qlobal şəbəkələrdə fiziki kanalların etibarlılığı aşağı olduqda, lokal şəbəkələrdən fərqli, verilənlərin ötürülməsinin daha mürəkkəb üsulları və uyğun avadanlıq tələb olunur.
- **Verilənlərin mübadiləsi sürəti.** Lokal şəbəkələri qlobal şəbəkələrdən fərqləndirən əsas əlamət kompüterlərarası yüksək sürətli verilənlər mübadiləsi kanallarının mövcudluğudur. Onların sürəti (10, 16 və 100 *Mbit/s*) kompüterin qurğularının işləmə sürətləri ilə müqayisə olunur. Qlobal şəbəkələr üçün daha aşağı sürət 2,4 *Kbit/s*-dən 2 *Mbit/s*-yə kimi səciyyəvidir.
- **Xidmətlərin müxtəlifliyi.** Verilənlər mübadiləsinin yüksək sürətli olması lokal şəbəkələrdə geniş çeşidli xidmətlərin yaranmasına gətirdi: fayl xidmətinin müxtəlif xidmətləri, çap xidmətləri, verilənlər bazaları xidmətləri, elektron poçt və s. Qlobal şəbəkələr isə əsasən poçt xidmətləri və bəzən məhdud imkanlı fayl xidmətlərini təqdim edirdilər.
- **Sorğuların yerinə yetirilmə operativliyi.** Paketin lokal şəbəkə ilə keçməsi adətən millisaniyə təşkil edir, paketin qlobal şəbəkə ilə keçməsi isə bir neçə saniyəyə çata bilər. Qlobal şəbəkələrdə verilənlərin ötürülmə sürətinin aşağı olması online rejimi üçün xidmətlərin həyata keçirilməsini çətinləşdirir.
- **Kanalların bölüşdürülməsi.** Lokal şəbəkələrdə rabitə kanalları, bir qayda olaraq, şəbəkənin bir neçə qovşağı ilə eyni zamanda birgə istifadə olunurlar, qlobal şəbəkələrdə isə – fərdi.
- **Miqyaslama.** Lokal şəbəkələr, stansiyanın qoşulması və xətlərin uzunluğunu təyin edən baza texnologiyalarının sərtliyinə görə, pis miqyaslamaya malikdirlər. Belə olduqda qovşaqların

sayı və rabitə xətlərinin uzunluğuna qoyulan məhdudiyyətlər artdıqda şəbəkənin xüsusiyyətləri kəskin pisləşir. Qlobal şəbəkələrə yaxşı miqyaslama məxsusdur, belə ki, onlar əvvəlcədən ixtiyari topologiyalarla işləməyi və abunəçilərin ixtiyari sayda olmasını nəzərə alaraq işlənmişdilər.

Tədricən lokal və qlobal şəbəkələr arasında olan fərqlər silinməyə başladı. Əvvəllər təcrid olunmuş lokal şəbəkələr bir-biri ilə birləşməyə başladılar, bu zaman əlaqələndirici mühit kimi qlobal şəbəkələr istifadə olunurdu. Lokal və qlobal şəbəkələrin sıx inteqrasiyası uyğun texnologiyaların qarşılıqlı birləşməsinə gətirdi. Verilənlərin ötürülməsi üsullarında yaxınlaşma optik-lifli rabitə xətlərinin istifadəsi nəticəsində baş verdi. Verilənlərin ötürülməsinin bu mühitini 100 *metrdən* çox olan məsafədə informasiyanın sürətli mübadiləsi üçün praktiki olaraq lokal şəbəkələrin bütün texnologiyaları istifadə edirlər. Bu mühit əsasında qlobal kompüter şəbəkələrinin avadanlıqları da qurulub.

Lokal və qlobal şəbəkələrin yaxınlaşmasında IP protokolunun rolu vacibdir. Hal-hazırda bu protokol müxtəlif alt şəbəkələrdən vahid tərtibli şəbəkənin qurulması üçün lokal və qlobal şəbəkələrin istənilən texnologiyaları üzərində istifadə olunur. Sürətli rəqəmsal kanallar əsasında işləyən qlobal şəbəkələr təklif etdikləri xidmətlərin sayını artırdılar və bu mənada lokal şəbəkələrə çatdılar. İstifadəçiyə real rejimdə böyük həcmdə informasiyanın (təsvirlərin, videofilmlərin, səsin) çatdırılması ilə bağlı olan xidmətlərin yaradılması mümkün oldu. Buna ən parlaq misal – İnternet-də informasiyanın əsas tədarükçüsü olan World Wide Web hipermətn informasiya xidmətidir. Onun interaktiv imkanları lokal şəbəkələrin analoji imkanlarını üstələyir. Ona görə də lokal şəbəkələrin layihəçiləri bu xidməti qlobal şəbəkələrdən götürüblər. Qlobal İnternet şəbəkəsindən lokal şəbəkəyə xidmətlərin köçürülməsi kütləvi xarakter aldığı üçün İntranet-texnologiya anlayışı da yarandı.

Son zamanlar lokal şəbəkələrdə, qlobal şəbəkələrdə olduğu kimi, icazəsiz müraciətdən informasiyanın qorunmasını təmin edən üsullara böyük diqqət yetirilir. Bu onunla əlaqədardır ki, lokal şəbəkələr daha təcrid olunmuş deyillər və əksər hallarda onların dünyaya çıxışı qlobal şəbəkələr vasitəsi ilə təmin olunub.

Lokal şəbəkələrdə informasiyanın qorunması eyni üsullar əsasında qurulur – verilənlərin şifrlənməsi, istifadəçilərin audentifikasiyası və avtorizasiyası.

Nəhayət, əvvəlcədən hər iki növ şəbəkə üçün nəzərdə tutulmuş yeni texnologiyalar yarandı. Yeni texnologiya nəslinin parlaq nümayəndəsi ATM texnologiyasıdır ki, bu texnologiya vahid nəqliyyat şəbəkəsində mövcud olan trafik növlərini səmərəli birləşdirərək həm qlobal, həm də lokal şəbəkələr üçün əsas kimi götürülə bilər. Digər bir misal – Ethernet texnologiyası ailəsidir. Yeni Ethernet 10G standartı verilənlərin 10 *Hbitls* sürəti ilə ötürməsinə imkan verir və həm qlobal, həm də iri lokal şəbəkələrin magistralları üçün nəzərdə tutulub. Bu yaxınlaşmanın təzahürü böyük şəhərlər miqyasında şəbəkələrin yaranmasıdır. Bu şəbəkələr lokal və qlobal şəbəkələr arasında aralıq yer tutur. Qovşaqlar arasında məsafələr böyük olduqda onlar keyfiyyətli rabitə xətlərinə və yüksək mübadilə sürətlərinə malikdirlər. Bu göstəricilər, klassik lokal şəbəkələrdən fərqli olaraq, yüksəkdir. Lokal şəbəkələrdə olduğu kimi, şəhər şəbəkələrinin qurulması üçün mövcud şəbəkələr istifadə olunmur, yeniləri çəkilməlidir.

2.2. Lokal şəbəkələrin topologiyaları

Lokal şəbəkələrin müxtəlif yaradılma texnologiyaları işləndiyindən onların oxşar və fərqləndirici cəhətlərinin nəzərə alınması vacibdir. Bu texnologiyalar arasında ümumi cəhətləri anlamaq üçün hər bir şəbəkəni, onun topologiya və ya ümumi formasına uyğun olaraq, konkret kateqoriyaya aid edirlər.

Lokal şəbəkələrin topologiyalarını həm fiziki, həm də məntiqi nöqteyi nəzərdən təsvir etmək olar. Fiziki topologiya lokal şəbəkənin komponentlərinin həndəsi yerləşməsini təsvir edir. Topologiya lokal şəbəkənin forma və strukturunu qrafiki əks etdirən nəzəri bir konstruksiyadır. Məntiqi topologiya qarşılıqlı əlaqə yarada bilən şəbəkənin son nöqtələri arasında mümkün olan birləşmələri əks etdirir. Bu informasiya son nöqtələr cütü arasında birbaşa fiziki birləşmələrin mövcudluğunun təyin edilməsi zamanı və bir biri ilə qarşılıqlı əlaqə yarada bilən son nöqtələrin təsviri zamanı faydalı ola bilər. Kompüter şəbəkəsinin topologiyası dedikdə, şəbəkənin kompüterlərinin fiziki yerləşməsi və

onların rabitə xətləri vasitəsi ilə birləşməsi üsulu başa düşülür. Qeyd etmək lazımdır ki, topologiya anlayışı lokal şəbəkələrə aiddir, belə ki, burada əlaqələrin strukturunu asanlıqla izləmək olar. Qlobal şəbəkələrdə əlaqələrin strukturu istifadəçilərdən gizli qalır. Topologiya avadanlığa qoyulan tələbləri, istifadə olunan kabelin növünü, mübadilənin idarəolunmasının daha rahat üsullarını, işin etibarlılığını, şəbəkənin genişlənməsi imkanlarını təyin edir. İstifadəçi tərəfindən şəbəkənin topologiyasının seçilməsi tez-tez baş verməsə belə, o, əsas topologiyaların xüsusiyyətlərini, onların müsbət və mənfi cəhətlərini bilməlidir.

Son zamanlara qədər üç əsas topologiya növü mövcud idi:

- ***Şinvari topologiyası*** – bütün kompüterlər paralel olaraq bir rabitə xəttinə qoşulur və hər bir kompüterdən gələn informasiya eyni zamanda qalan kompüterlərə ötürülür (şəkil 2.1).

- ***Ulduzvari topologiya*** – bir mərkəzi kompüterə qalan periferiya kompüterləri qoşulur, burada hər bir kompüter özünə aid ayrıca rabitə xəttini istifadə edir (şəkil 2.2).

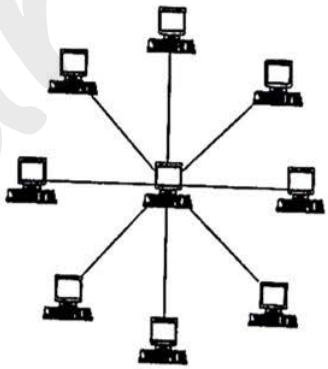
- ***Halqavari topologiya*** – hər bir kompüter halqa üzrə növbəti kompüterə informasiyanı ötürür və yalnız özündən qabaq gələn kompüterdən informasiyanı alır (şəkil 2.3).

Şinvari topologiya. Şin topologiyalı kompüter şəbəkəsində adətən bir kabeldən istifadə olunur və kompüterlər bu kabelə qoşulur. Şinə qoşulmuş hər hansı bir kompüter kabel üzrə siqnal ötürə bilər və bu siqnal bütün kompüterlərlə qəbul olunacaq. Kabelə qoşulan bütün kompüterlər kabel ilə ötürülən elektrik siqnalı qəbul edə biləcəyi üçün, hər bir kompüter digər kompüterə verilənlər ötürə bilər. Aydındır ki, şin topologiyalı şəbəkəyə qoşulmuş kompüterlər öz hərəkətlərini əlaqələndirməlidirlər, belə ki, istənilən zaman anında ancaq bir kompüter siqnal ötürməlidir, əks halda xaos yarana bilər, yəni siqnalların üst-üstə düşməsi nəticəsində informasiya təhrif olunacaq. Beləliklə, şində yarımdupleks (hər iki istiqamətdə, lakin növbə üzrə) mübadilə rejimi həyata keçirilir.

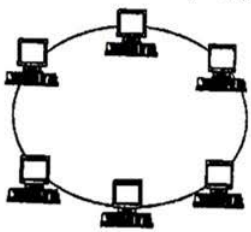
Şin topologiyasında özündən bütün informasiyanı keçirən mərkəzi abunəçi yoxdur, bu da onun etibarlılığını artırır. Şinə yeni abunəçilərin əlavə olunması çox asandır və adətən şəbəkənin işləməsi zamanı mümkündür.

Şində hər hansı bir kompüterin sıradan çıxması, şəbəkənin işinə təsir etmir. Elə bir təsəvvür yarana bilər ki, kabelin qırılması şin üçün təhlükə yaratmır, yəni bu halda iki işlək şin alınır. Lakin uzun rabitə xətləri ilə elekrik siqnalların yayılması xüsusiyyətlərini nəzərə alaraq, şinin uclarında xüsusi qurğular – terminatorlar (şəkil 2.1-də dördbucaqlar şəklində verilib) – yerləşdirilməlidir. Terminatorlar

Şəkil 2.1. Şinvari topologiyası



Şəkil 2.2. Ulduzvari topologiya



Şəkil 2.3. Halqavari topologiya

olmadan siqnal xəttin ucundan əks edilərək elə təhrif olunur ki, şəbəkə ilə əlaqə mümkün olmur. Kabelin qırılması və ya zədələnməsi nəticəsində rabitə xəttinin işi pozulur, və hətta bir-biri ilə əlaqədə olan kompüterlərarası da verilənlər mübadiləsi dayanır. Kabelin hər hansı bir nöqtəsində baş verən qısa qapanma nəticəsində bütün şəbəkə sıradan çıxır.

Şin topologiyalı şəbəkənin uzunluğunu artırmaq məqsədi ilə adətən bir neçə seqmentdən istifadə olunur. Hər bir seqment ayrıca bir şindir. Şəbəkə seqmentləri bir-biri ilə repiterlər və ya təkrarlayıcılar vasitəsi ilə birləşirlər. Şəkil 2.4-də iki şəbəkə seqmentinin birləşməsi göstərilib. Lakin şəbəkənin uzunluğunun sonsuzluğa qədər artırılması mümkün deyil, belə ki, rabitə xətləri ilə siqnalların yayılması sürəti ilə əlaqəli məhdudiyyətlər mövcuddur. Şin topologiyasının kiçik lokal şəbəkələrdə istifadəsi məqsədəuyğundur.

Ulduzvari topologiya. Kompüter şəbəkələrində bütün kompüterlər vahid mərkəzə (şəkil 2.2) qoşulduqda ulduzvari topologiya istifadə olunur. Ulduz formasına malik olan şəbəkədə kompüterlərarası əlaqələr bir yerdə toplandığı üçün, ulduzvari şəbəkənin mərkəzini nüvə və ya konsentrator adlandırırlar. Ənənəvi konsentrator ötürücü kompüterdən verilənləri qəbul edən və onları qəbuledıciyə ötürən elektron qurğudan ibarətdir. Bir qayda olaraq, mərkəzi kompüter ən güclüdür və mübadilənin idarəolunması funksiyaları məhz onun boynuna düşür. Bu topologiyanın şin topologiyasına nisbətən üstünlüyü – etibarlılığın yüksək olmasıdır. Kabeldə olan nasazlıq ancaq bu kabelə qoşulan kompüterə təsir edir və tək konsentratorun nasazlığı şəbəkəni sıradan çıxarır. Təcrübədə ulduzvari şəbəkələr nadir hallarda simmetrik struktura malik olurlar, yəni konsentrator bütün kompüterlərə nisbətən eyni məsafədə yerləşir. Adətən konsentrator ona qoşulmuş kompüterlərdən aralı yerləşir.

Ulduzvari topologiyanın mənfi cəhəti – şəbəkə avadanlığının baha olmasıdır. Ciddi çatışmamazlıq abunəçilərin sayının məhdud olması ilə bağlıdır. Adətən mərkəzi abunəçi 8 – 16 periferiya abunəçilərinə xidmət edir. Bəzən iyerarxik şəkildə bir neçə konsentratorun ulduz tipli əlaqələrlə birləşdirilmiş şəbəkələrinin qurulması məqsədəuyğundur. Hal-hazırda iyerarxik ulduz həm

lokal, həm də qlobal şəbəkələrdə istifadə olunan ən geniş yayılmış topologiya növüdür.

Şəkil 2.2-də göstərilmiş ulduzvari topologiya aktiv və ya əsil ulduz adlanır. Xaricən ulduza oxşayan passiv ulduz adlanan topologiya da mövcuddur (şəkil 2.5). Belə topologiyalı şəbəkənin mərkəzində kompüter deyil, repiterin funksiyalarını yerinə yetirən, konsentrator və ya hab yerləşir. O, gələn siqnalları bərpa edir və onları digər rabitə xətlərinə ötürür.

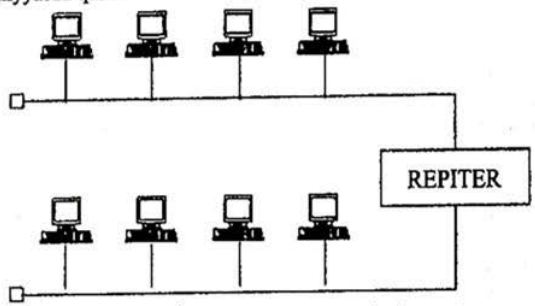
Aktiv və passiv ulduz arasında aralıq topologiya növünü ayırmaq olar. Bu halda konsentrator ona gələn siqnalların retranslyasiyasından başqa, özü mübadilədə iştirak etmədən, mübadilənin idarəolunmasını yerinə yetirir.

Halqavari topologiya. Halqa konfiqurasiyalı şəbəkələrdə (şəkil 2.3) verilənlər, bir qayda olaraq, bir istiqamətdə, dairə üzrə bir kompüterdən digərinə ötürülürlər. Əgər kompüter gələn verilənləri özününkü kimi tanıyırsa, onda onların surətini öz daxili buferinə köçürür. Bu topologiyada bütün kompüterlər qapalı ilməyə birləşiblər: bir kabel birinci kompüteri ikinci kompüter ilə birləşdirir, ikinci kabel ikinci kompüteri üçüncü kompüterlə birləşdirir və bu proses sonuncu kompüterin birinci kompüterlə birləşməsinə qədər davam edir. Qeyd etmək lazımdır ki, ulduzvari topologiyada olduğu kimi, halqavari topologiya kompüterlərin fiziki yerləşməsini deyil (dairəvi şəbəkənin kompüter və birləşmələri dairə şəklində yerləşməyə də bilər), iki kompüter arasında məntiqi birləşməni təsvir edir. Dairəvi şəbəkənin iki kompüteri arasındakı kabel dəhlizdən keçib, şaquli istiqamətdə binanın bir mərtəbəsindən digərinə çəkilə bilər.

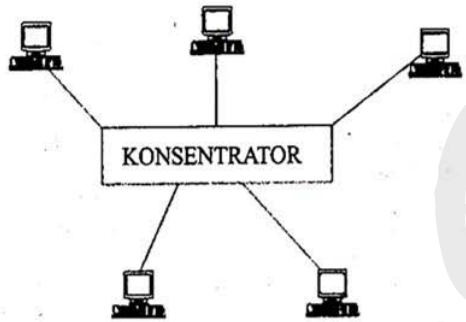
Halqaya yeni abunəçilərin qoşulması zamanı bütün şəbəkənin işinin dayandırılması tələb olunsa da, qoşulma "ağrısız" keçir. Halqavari topologiya artıq yüklənmələrə ən dayanıqlı topologiyadır, belə ki, burada münaqişələr, həmçinin, mərkəzi abunəçi yoxdur.

Siqnal dairənin bütün kompüterlərindən keçdiyi üçün kompüterlərin birinin sıradan çıxması bütün şəbəkəni sıradan çıxarır. Eyni ilə, kabelin qırılması və ya qısa qapanma şəbəkənin işini qeyri-mümkün edir. Dairə kabellərin zədələnməsinə daha çox həssasdır, ona görə də bu topologiyada iki və ya daha çox

paralel rabitə xətlərinin çəkilməsi nəzərdə tutulub, xətlərdən biri ehtiyyatda qalır.



Şəkil 2.4. İki şəbəkə seqmentinin birləşməsi



Şəkil 2.5. Passiv ulduz

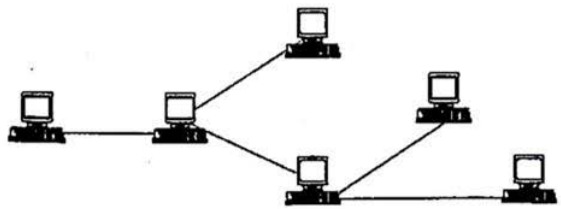
Halqanın mənfi cəhəti ondan ibarətdir ki, hər bir kompüterə iki kabel çəkilməlidir .

Hər bir topologiyanın mənfi və müsbət cəhətləri mövcuddur. Halqavari topologiya kompüterlərin müraciətlərinin əlaqələndirilməsi və şəbəkənin düzgün işləməsinin yoxlanılmasını sadələşdirir. Lakin bir kabelin zədə alması bütün şəbəkəni iflic vəziyyətə gətirir. Ulduzvari topologiya şəbəkəni bir kabelin qırılmasından qoruyur, belə ki, hər bir kabelə yalnız bir kompüter qoşulur. Şin topologiyasında, ulduzvari topologiyaya nisbətən, kabeldən az istifadə olunur, lakin bu toplogiyada, ulduzvari

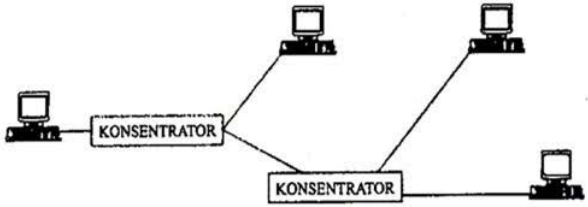
topologiyada olduğu kimi, magistral kabelin zədələnməsi şəbəkənin işini qeyri-mümkun edir.

Baxılan üç əsas baza texnologiyalardan başqa ağac şəbəkə texnologiyasından da istifadə olunur. Bu texnologiyaya bir neçə ulduzun kombinasiyası kimi baxmaq olar. Ulduzvari topologiyada olduğu kimi, ağac da aktiv (şəkil 2.6) və passiv (şəkil 2.7) ola bilər. Aktiv ağac olduqda bir neçə rabitə xətlərinin birləşməsi mərkəzlərində mərkəzi kompüterlər, passiv ağac olduqda isə konsentratorlar yerləşir.

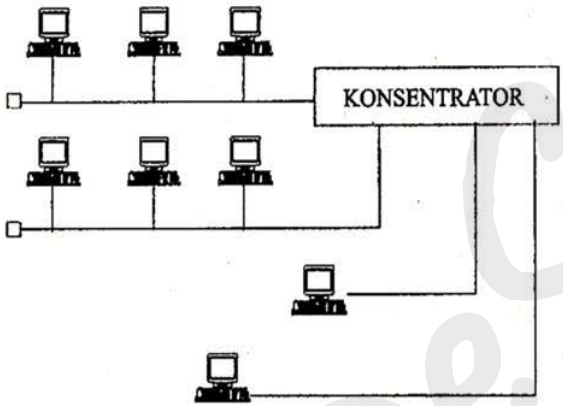
Kombinəedilmiş topologiyalardan da tez-tez istifadə olunur. Onlar arasında ulduz-şin (şəkil 2.8) və ulduz-halqa (şəkil 2.9) topologiyaları ən çox yayılan topologiyalardır. Ulduz-şin topologiyasında şin və passiv ulduzun kombinasiyasından istifadə olunur. Bu halda konsentratora həm ayrı-ayrı kompüterlər, həm bütöv şin seqmentləri qoşulur, yəni əslində şəbəkənin bütün kompüterləri daxil olan fiziki şin topologiyası həyata keçirilir. Bu topologiyada bir-biri ilə birləşmiş və magistral, dayaq şin adlanan şini təşkil edən bir neçə konsentrator da istifadə oluna bilər. Belə olduqda hər bir konsentratora ayrı-ayrı kompüterlər və ya şin seqmentləri qoşulur. Beləliklə, istifadəçi şin və ulduz topologiyalarının üstünlüklərini çevik kombinə etmək, həmçinin şəbəkəyə qoşulan kompüterlərin sayını asanlıqla dəyişdirmək imkanını əldə edir. Ulduz-halqa topologiyası olduqda halqaya kompüterlərin özləri deyil, xüsusi konsentratorlar birləşirlər ki, onlara da, öz növbəsində, ulduzvari qoşa rabitə xətti vasitəsi ilə kompüterlər qoşulur. Bu topologiya ulduz və halqa topologiyalarının üstünlüklərini kombinə etməyə imkan verir.



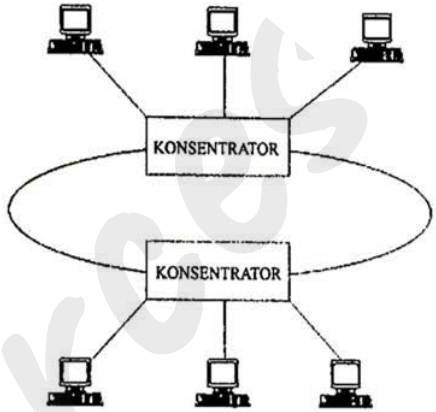
Şəkil 2.6. Aktiv ağac



Şəkil 2.7. Passiv ağac



Şəkil 2.8. Ulduz-şin topologiyası



Şəkil 2.9. Ulduz- halqa topologiyası

2.3. Informasiya ötürülməsinin fiziki mühiti

Şəbəkələrin qurulması zamanı müxtəlif fiziki mühiti (havada asılmış telefon və teleqraf naqilləri, mis koaksial kabellər, mis burulmuş cütlər, lifli-optik kabellər, radiodalğalar) istifadə edən rabitə xətləri istifadə olunurlar. Bu və ya digər növ rabitə xətlərinin seçilməsi zamanı layihəçilər birinci növbədə onların texniki xarakteristikalarını, qiymətini və qurulma sadəliyini nəzərə alırlar.

İnformasiya ötürmə mühiti dedikdə, kompüterlərarası informasiya mübadiləsini aparan rabitə xətləri başa düşülür. İnformasiya ötürülməsinin fiziki mühiti kimi kabellər, yəni naqillər yığımı, izolyasiya və qoruyucu örtüklər, birləşdirici ucluqlar, informasiya siqnallarını yayan yer atmosferi və ya kosmik fəzanı göstərmək olar. Müasir telekommunikasiya sistemlərində informasiya elektrik cərəyanı və ya gərginlik, radosiqnal və ya işıq siqnalları vasitəsi ilə ötürülür.

Verilənlərin ötürülməsi mühitindən asılı olaraq rabitə xətləri aşağıdakı qruplara ayrılırlar:

- simli (hava ilə);

- kabelli (mis və optik-lifli);
- yerüstü və peyk əlaqə radiokanalları.

Simli xətlər havada sallanan və tirlər arasında çəkilmiş hər hansı bir izolyasiya və ya ekranlaşdıran hörgüsü olmayan naqillərdir. Belə xətləri ənənəvi olaraq telefon və ya teleqraf siqnallarının ötürülməsi üçün istifadə edirlər, lakin digər imkanlar olmadıqda onları kompüter verilənlərinin ötürülməsi üçün də istifadə edirlər. Bu xətlərin maniələrə qarşı qorunması və keyfiyyəti çox aşağıdır. Hal-hazırda simli xətlər kabel xətləri ilə əvəz olunurlar.

Kabel xətləri çox mürəkkəb bir konstruksiyaya malikdirlər. Kabel bir neçə izolyasiya qatı (elektrik, elektromaqnit, mexaniki) arasında yerləşən naqillərdən ibarətdir. Bundan başqa kabel ucluqlarla təhciz oluna bilər ki, onların vasitəsi ilə istənilən avadanlığın cəld qoşulması mümkündür.

Çəkilmə və istismar şərtlərindən asılı olaraq kabellər daxili və xarici kabellərə ayrılırlar. Onlar isə öz növbəsində yeraltı, sualtı və hava ilə çəkilən kabellərə ayrılırlar.

Kompüter şəbəkələrində üç növ kabeldən istifadə olunur:

- burulmuş mis naqillər əsasında kabellər;
- mis özəkli koaksial kabellər;
- lifli-optik kabellər.

Hər növ kabel mənfi və müsbət cəhətlərə malikdir. Ona görə də, kabelin seçilməsi zamanı həm həll olunan məsələnin xüsusiyyətləri, həm konkret şəbəkənin xüsusiyyətləri, o cümlədən, istifadə olunan topologiya nəzərə alınmalıdır.

Burulmuş naqillər cütü burulmuş cüt adlanır. Naqillərin burulması kabel ilə ötürülən faydalı siqnallara olan xarici və qarşılıqlı maniələrin təsirini azaldır.

Uzun müddət ərzində səs əlaqəsinin təşkili üçün istifadə olunan burulmuş cüt lokal şəbəkələr üçün kabel texnologiyalarının qeyri-rəsmi standartı oldu. Burulmuş cütlər əsasında kabel vahid dielektrik örtükdə yerləşən bir neçə burulmuş izolyasiya edilmiş mis naqillərdən ibarətdir. Bu kabel elastikdir və çəkilmə üçün rahatdır. Adətən kabelə iki və ya dörd burulmuş cüt daxil olur (şəkil 2.10). Burulmuş cütün iki əsas növü mövcuddur:

- ekranlaşmış;
- ekranlaşmamış.

Ekranlaşmamış burulmuş cütlər xarici elektromaqnit maneələrdən zəif qorunması ilə xarakterizə olunurlar, bundan başqa, misal üçün sənaye casusluğu məqsədi ilə, xəlvəti qulaq asılmadan da zəif qorunurlar. İnformasiyanın ələ keçirilməsi kontaktlı və kontaktsız üsullar vasitəsi ilə əldə edilə bilər. Kontaktsız üsul kabel tərəfindən şüalandırılan elektromaqnit sahələrin radio ilə ələ keçirilməsindən ibarətdir. Bu çatışmamazlıqları aradan qaldırmaq üçün ekranlaşmış kabellərdən istifadə olunur.

Ekranlaşmış burulmuş cütlərdən istifadə etdikdə hər bir burulmuş cüt metal hörgüdə (ekran) yerləşir. Ekran kabelin şualanmasını azaldır, xarici elektromaqnit maniələrdən qoruyur, naqillərin biri-birinə təsirini azaldır. Aydındır ki, ekranlaşmış burulmuş cüt ekranlaşmamışa nisbətən bahadır, bundan başqa onun istifadəsi zamanı xüsusi ekranlaşdırıcı ucluqların tətbiqi tələb olunur. Bu səbəbdən ekranlaşmış burulmuş cüt ekranlaşmamışa nisbətən az istifadə olunur.

Ekranlaşmamış burulmuş cütlərin əsas üstünlükləri kabelin uclarında ucluqların montajının sadəliyi, bundan başqa, digər növ kabellərə nisbətən, istənilən zədələrin təmirinin sadəliyidir. Qalan xarakteristikalar, digər kabellərə nisbətən, pisdirlər.

Standarta uyğun olaraq, ekranlaşmamiş burulmuş cüt əsasında 5 kabel kateqoriyası mövcuddur:

- Birinci kateqoriyaya aid olan kabel səs verilənlərin ötürülməsi üçün istifadə olunur. Bu kabel istənilən növ verilənlərin ötürülməsi üçün nəzərdə tutulmayan adi telefon kabelidir və əksər hallarda rəqəmsal verilənlərin ötürülməsi üçün nəzərdə tutulmur.
- İkinci kateqoriyaya aid olan kabel 1 *MHs*-ə qədər olan tezlik zolağında 4 *Mbit/s*-dən çox olmayan sürətlə verilənlərin ötürülməsi üçün istifadə olunur. Bu kabel növü markerin ötürülməsi protokolunu istifadə edən köhnəlmiş halqa topologiyası üçün səciyyəvidir. Hal-hazırda nadir hallarda istifadə olunur.
- Üçüncü kateqoriyalı kabel 16 *MHs*-ə qədər tezlik zolağında verilənlərin ötürülməsi üçün istifadə olunur, 16 *Mbit/s*-yə qədər sürət ilə verilənlərin ötürülməsi üçün nəzərdə tutulmuşdur. Lokal şəbəkələr üçün standart tərəfindən tövsiyyə olunan ən sadə kabel növüdür.

- Dördüncü kateqoriyalı kabel 20 *MHs*-ə qədər tezlik zolağında 16 *Mbit/s*-yə qədər sürətlə verilənlərin ötürülməsi üçün nəzərdə tutulur. Üçüncü kateqoriyadan az fərqləndiyi üçün nadir hallarda istifadə olunur.

- Beşinci kateqoriyalı kabel ən geniş yayılmış mühitdir və 100 *Mbit/s*-yə qədər sürətlə verilənlərin ötürülməsini dəstəkləyir. 100 *MHs*-ə qədər tezlik zolağında verilənlərin ötürülməsi üçün nəzərdə tutulub. Müasir yüksək sürətli şəbəkələrdə istifadəsi məqsədəuyğundur. Bu kabel təxminən 30–50%-ə qədər digərlərindən bahadır.

Altıncı və yeddinci kateqoriyalı kabellər – uyğun olaraq 200 *MHs* və 600 *MHs* tezlik zolaqlarında verilənlərin ötürülməsi üçün perspektiv kabellərdir.

Burulmuş cütlərin qoşulması üçün RJ-45 ucluqlarından istifadə olunur. Onların 8 kontaktı olur. Ucluqlar kabelə xüsusi sıxlaşdırıcı alətlər vasitəsi ilə birləşdirilirlər. Bu halda ucluğun qızılı iynəcik kontaktları hər bir naqilin izolyasiyasını deşib onun özəkləri arasından keçərək, etibarlı və keyfiyyətli birləşməni təmin edirlər. Adətən burulmuş cütlər verilənlərin bir istiqamətdə ötürülməsi üçün yəni ulduzvari və halqavari topologiyalarında istifadə olunurlar.

Koaksial kabel mərkəzi naqil və metal hörgüdən ibarət, biribirindən dielektrik qatı ilə ayrılmış və ümumi xarici örtükdə yerləşdirilmiş elektrik kabeldir (şəkil 2.11). Son zamanlara qədər koaksial kabel geniş istifadə olunurdu. Bu onun maniələrə qarşı yüksək qorumnası ilə əlaqədar idi. Koaksial kabelin üstünlüyü – təkrarlıyıcılardan istifadə etmədən uzaq məsafədə yüksək keçirtmə qabiliyyətinin dəstəklənməsi imkanıdır. İcazəsiz qoşulma burada çətindir, bu kabel xaricə elektromaqnit şüalanmanı az verir.

Hal-hazırda koaksial kabel burulmuş cütə nisbətən, az istifadə olunur, belə ki, koaksial kabel nisbətən zərif bir konstruksiyadır. O, düyünləri, bükülmələri, hətta statik təzyiqi mümkün etmir. Bu cür təsirlər kabelin strukturunu zədələyə bilər və siqnalların ötürülməsinə əngəl yarada bilər. Bundan başqa koaksial kabelin istifadəsinin mənfi cəhəti onun qiyməti və ölçüsüdür. Daha mürəkkəb quruluşuna görə koaksial kabel, burulmuş cütə nisbətən, bahalıdır.

Koaksial kabellər əsasən şin topologiyalı şəbəkələrdə istifadə olunurlar. Bu halda kabelin uclarında, siqnalın daxili əks olunmasının qarşısını almaq üçün, terminatorlar qurulmalıdır. Qeyd etmək lazımdır ki, terminatorların biri torpaqlanmalıdır. Torpaqlanma olmasa metal hörgü şəbəkəni xarici elektromaqnit maniələrdən qorumur və şəbəkə ilə ötürülən informasiyanın xarici mühitə şualanmasını aşağı salmır.

İki növ koaksial kabel mövcuddur:

- diametri təxminən 0,5 *sm* olan, daha elastik (əyilgən) nazik kabel;

- diametri təxminən 1 sm olan, daha bərk yoğun kabel.

Nazik kabel, yoğun kabelə nisbətən, qısa məsafələrə ötürmə üçün istifadə olunur, belə ki, onda siqnalın sönməsi daha güclü baş verir. Lakin nazik kabellə işləmək daha rahatdır: onun hər bir kompüterə operativ çəkilməsi mümkündür, yoğun kabel isə otağın divarında qeyd olunmasını tələb edir. Yoğun kabel, nazik kabelə nisbətən, təxminən iki dəfə bahadır. Ona görə də nazik kabelə daha tez rast gəlmək olur.

Koaksial kabelin ikiekranlı variantı mövcuddur. Bu kabellər maniələrə qarşı yaxşı qorunur, ona görə də adi koaksial kabellərə nisbətən bahadır.

Hal-hazırda hesab olunur ki, koaksial kabel köhnəlib və əksər hallarda onu burulmuş cüt və ya optik-lifli kabel əvəz edə bilər.

Optik-lifli kabel. Bu kabel prinsipcə elektrik və mis kabeldən fərqlənir. Bu kabeldə informasiya elektrik siqnalla deyil, işıq siqnalı vasitəsi ilə ötürülür. Onun əsas elementi – şəffaf şüşə lifidir ki, onunla keçən işıq böyük məsafələrdən keçərək gücünü az itirir. Optik-lifli kabelin strukturu sadədir və koaksial elektrik kabelin strukturuna oxşayır, lakin mərkəzi mis naqil əvəzinə nazik şüşə lifi, daxili izolyasiya əvəzinə isə şüşə və ya plastik örtük istifadə olunur. Bu örtük işığın optik lifdən kənara çıxmasına imkan vermir (şəkil 2.12). Optik-lifli kabelin bir ucunda yerləşmiş ötürücüdə şüşə lif ilə işıq impulslarının ötürülməsi üçün işıq diodu və ya lazer istifadə olunur. Kabelin digər ucunda yerləşən qəbuledicidə, bu impulsların askar olunması üçün işığa xəssas tranzistor istifadə olunur.

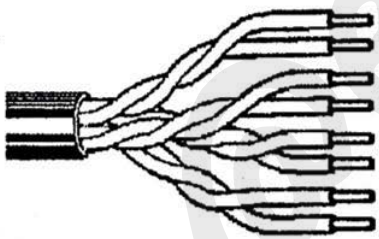
Elektrik kabellərə nisbətən optik-lifli kabellərin dörd əsas üstünlüyünü qeyd etmək olar:

- verilənlərin ötürülməsi üçün işığın istifadə olunması digər kabellərdə elektromaqnit maniələr yaratmır və özləri də elektromaqnit dalğalara həssas deyillər;
- şüşə liflər onlarla keçən işığı yaymadıqlarına görə, optik-lifli kabel, elektrik siqnalı ötürən mis kabelə nisbətən, işıq impulslarını daha böyük məsafəyə ötürməyə qadirdir;
- işıq daha böyük həcmdə informasiyanın kodlaşdırılmasına imkan verdiyi üçün, daha böyük həcmdə informasiyanı ötürə bilər;
- elektrik dövrəsinin yaradılması üçün iki naqil tələb olunur, işıq isə bir kompüterdən digərinə bir optik lif ilə ötürülə bilər.

Sadalanan üstünlüklərə baxmayaraq, optik-lifli kabellərin çatışmayan cəhətləri də mövcuddur. Optik-lifli kabelin çəkilməsi zamanı xüsusi avadanlıq vasitəsi ilə liflərin uclarının hamarlanması tələb olunur ki, onlardan işıq keçə bilsin. Əgər plastmas örtüyün içində şüşə lifi sınıbsa, zədələnmiş yerin tapılması və sınmış optik-lifli kabelin təmiri çox çətindir, belə ki, optik-lifli kabelin hər iki ucunun birləşdirilməsi və işığın birləşmə yerlərindən maniəsiz keçməsinin təmin edilməsi üçün xüsusi avadanlıq tələb olunur.

Optik-lifli kabel ancaq ulduz və halqa topologiyalı şəbəkələrdə istifadə olunur. Bu halda uyğunlaşma və torpaqlanma ilə heç bir problem yaranmır. İki optik-lifli kabel növü mövcuddur:

- ucuz, lakin aşağı keyfiyyətli çoxmodalı kabel;
- bahalı, lakin yaxşı xüsusiyyətlərə malik birmodalı kabel.



Şəkil 2.10. Burulmuş naqillər cütü olan kabel

Şəkil 2.11. Koaksial kabel

Şəkil 2.12. Optik-lifli kabel

Bu tiplər arasında olan əsas fərq kabeldə işıq şualarının keçməsi rejimlərinin müxtəlifliyi ilə bağlıdır. Moda anlayışı kabelin daxili milində işıq şüalarının yayılması rejimini əks etdirir.

Birmodalı kabeldə praktiki olaraq bütün şüalar eyni yol keçirlər, nəticədə onlar eyni anda qəbuledıciyə çatırlar, və siqnalın forması praktiki olaraq təhrif olunmur. Birmodalı kabeldə kiçik diametrli mərkəzi keçirici istifadə olunur. Bu halda xarici ötürücüdən əks olunmayaraq, işıq şüalarının praktiki olaraq hamısı, işıqdaşıyıcının optik oxu boyunca yayılırlar.

Çoxmodalı kabellərdə texnoloji nöqteyi nəzərdən hazırlanması daha asan olan daha enli daxili millərdən istifadə olunur. Ötürmə üçün adi işıqdiodu istifadə olunur, bu da, birmodalı kabelə nisbətən, qiyməti aşağı salır və qəbuledici-ötürücülərin istifadəsi müddətini artırır. Çoxmodalı kabellərin hazırlanması asandır, ona görə də onlar ucuzdurlar, lakin onların xüsusiyyətləri, birmodalı kabellərə nisbətən, pisdir. Nəticədə çoxmodalı kabellər verilənlərin əsasən kiçik məsafələrə (300 ÷ 2000 *m*) sürəti 1 *Hbit/s*-dən çox olmayaraq ötürülməsi üçün istifadə olunurlar, birmodalı kabellər isə çox yüksək sürətlə uzaq məsafələrə verilənlərin ötürülməsi üçün istifadə olunurlar.

Yerüstü və peyk əlaqələrinin radiokanalları. Kabel kanallarından başqa kompüter şəbəkələrində kabelsiz kanallardan da istifadə olunur. Onların əsas üstünlüyü ondadır ki, naqillərin çəkilməsi tələb olunmur. Radiokanallar radiodalğaların ötürücü və qəbuledici vasitəsi ilə təşkil olunurlar. İstifadə olunan tezlik

diapazonları və kanalların uzunluğu ilə fərqlənən müxtəlif tip radiokanallar mövcuddur. Verilənlərin ötürülməsi sürəti yüksək olmadıqda qısa, orta və uzun dalğaların diapazonları uzaq əlaqəni təmin edir.

Lakin lokal şəbəkələrdə bir sıra səbəblərə görə (ötürücü və qəbuledici qurğularının yüksək qiyməti, aşağı maniələrə qarşı qorunması, ötürülən informasiyanın məxfiliyinin qorunmaması və əlaqənin etibarlılığının aşağı olması) radiokanal geniş yayılmamışdır. Qlobal şəbəkələr üçün isə radiokanal adətən yeganə mümkün olan bir çıxış yoludur, belə ki, peyk-retranslyatorlar vasitəsi ilə bütün dünya ilə əlaqə yaratmağa imkan verir. Radiokanal biri-birindən uzaq masafədə yerləşən iki və daha çox lokal şəbəkələrin vahid şəbəkəyə birləşdirilməsi üçün istifadə olunur.

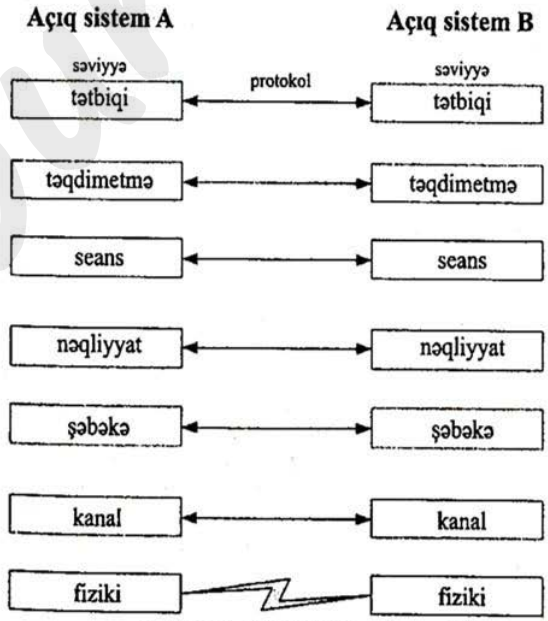
Hal-hazırda kompüter şəbəkələrində yuxarıda təsvir olunan verilənlərin ötürülməsinin fiziki mühitlərinin bütün növləri istifadə olunur, lakin ən perspektivli optik-lifli kabellərdir. Bu gün onlar əsasında iri ərazi və şəhər şəbəkələrinin magistralları və lokal şəbəkələrin yüksəksürətli rabitə xətləri qurulur. Burulmuş cüt də tez-tez istifadə olunan mühitdir, belə ki, bu mühit keyfiyyət və kəmiyyətin uyğunluğu, və qurulmanın sadəliyi ilə xarakterizə olunur. Peyk kanalları və radiokanallar kabel rabitə xətlərinin istifadəsi mümkün olmayan hallarda istifadə olunur.

2.4. Açiq sistemlərin qarşiliqli əlaqələrinin etalon modeli

İnformasiya-hesablama şəbəkələrin, xüsusən kompüter şəbəkələrin, yaradılması və istismarı problemlərinin həlli istifadəçilərin şəbəkələrlə və bir-biri ilə qarşılıqlı əlaqə prinsiplərini təyin edən müəyyən standartlar olmadan mümkün deyil. Açıq sistemlərin qarşılıqlı əlaqələrinin etalon modeli (ASQƏ) Beynəlxalq Standartlar Təşkilatı (BST) ilə verilənlərin ötürülməsi şəbəkələri üçün beynəlxalq standartlar kimi təklif olunaraq, müasir informasiya sistemlərinin əsasıdır. Bu modelin əsas məqsədi müxtəlif açıq sistemlərdə fəaliyyət göstərən uzaqlaşdırılmış obyektlərin (tətbiqi proseslərin) qarşılıqlı əlaqəsidir. ASQƏ-nin fiziki mühiti bütün açıq sistemləri əlaqələndirir. Etalon model kimi ASQƏ-in arxitekturasının yeddi səviyyəli modeli təyin olunub (şəkil 2.13). Hər bir açıq sistemə səviyyələrin məntiqi nizamlanması məcmusu

kimi baxılır. Bir-birindən uzaqlaşdırılmış tətbiqi proseslərin qarşılıqlı əlaqəsi müxtəlif açıq sistemlərin eyni adlı səviyyələrinin qarşılıqlı əlaqəsinə əsaslanır. Bu əlaqə bu səviyyələrin protokollarının mübadiləsi vasitəsi ilə yerinə yetirilir. Protokol dedikdə, eyni səviyyədə informasiya mübadiləsinin idarə olunması qaydası başa düşülür. İdarəetmə iyerarxiyasının hər bir səviyyəsində sistemin fəaliyyəti növbəti aşağı səviyyənin protokolunun prosedurları ilə təyin olunur. Buna görə ən yuxarı (yeddinci), tətbiqi səviyyə tətbiqi proseslərə digər açıq sistemlərdə yerləşən tətbiqi proseslərlə qarşılıqlı əlaqələr yaratmaq üçün xidmətlər təklif edir.

Hər bir səviyyənin protokollarına bir sıra tələblər qoyulur. Bu tələblərin yerinə yetirilməməsi protokolun fəaliyyətinin dayandırılması və bütün sistemin işinin sona çatmasına gətirir.



Şəkil 2.13. ASQƏ-BST etalon modeli

Qoyulan tələblər arasında ən vacibi imtinalara qarşı dayanıqlıq xassəsidir. Bu xassə protokolların ayrı-ayrı aparat və proqram vasitələrinin imkanlarına qarşı yüksək dayanıqlığını təmin edir. Protokolun qoyulan tələblərə uyğun gəlməsini yoxlamaq üçün onu sınaqdan keçirirlər. Bu sınaq ratifikasiya adlanır. Ratifikasiya nəticəsində protokolun qoyulmuş məsələlərin yerinə yetirilməsini təmin edib-etməməsi haqqında məlumat alınır.

ASQƏ-in etalon modelinin yeddi səviyyəsini nəzərdən keçirək. ASQƏ-nin arxitekturasında ən aşağı səviyyə – fiziki səviyyədir. Bu səviyyə fiziki rabitə kanalı ilə birləşdirilmiş istənilən qovşaqlar cütünün qarşılıqlı əlaqəsini təmin edir. Fiziki kanal – ASQƏ mühitində iki fiziki obyekt arasında bitlər ardıcıllığının ötürülməsi üçün bir yoldur. Fiziki səviyyə növbəti qonşu kanal səviyyəsinə xüsusi xidmətləri (məsələn, verilənlərin ötürülməsi üçün kanal səviyyəsinin komponentləri arasında fiziki birləşmənin həyata keçirilməsi) təqdim etməlidir. Hal-hazırda fiziki səviyyəyə aid BST və TTBMK (Telefon və Telefoniya üzrə Beynəlxalq Məsləhət Komitəsi) tərəfindən işlənmiş bir neçə standart mövcuddur. Misal üçün, verilənlərin emalı avadanlığında informasiyanın emalına aid böyük imkanlara malik olan X.21 protokolunun fiziki səviyyəsi. X.21 protokolu tək fiziki səviyyə deyil, həm də kanal və şəbəkə səviyyələrini əhatə edir.

ASQƏ arxitekturasında ikinci səviyyə – kanal səviyyəsidir. Bu səviyyə bəzi verilənlər bloklarının etibarlı ötürülməsini təmin edir. Ötürülmüş verilənlər bloklarında səhvlərin aşkar olunması, bundan başqa, səhvlərin aşkar olunmasından sonra verilənlərin bərpa edilməsi üçün, verilənlər blokunun əvvəli və sonunda müəyyən sayda idarəedici bitlər yerləşir. Bu məsələləri həll edən protokollar çox çətindir, belə ki, idarəedici bitlərin özləri ötürmə səhvlərinə məruz qala bilərlər. Əlavə idarəedici bitlərlə birgə verilənlər bloku (bitlər ardıcıllığı) kadr adlanır. Kadrın böyük ölçüsünü nəzərə alaraq, bəzi hallarda kadrlar hissələrlə ötürülür. Rabitə xəttinin digər başında kanal səviyyəsi bu hissələrdən tam kadr formalaşdırır. Kanal səviyyəsinin protokolları növbəti şəbəkə səviyyəsinə səhvlər haqqında xəbərdarlıq, axının idarə olunması, kanal birləşməsi kimi xidmətləri təklif edirlər, fiziki və şəbəkə səviyyələri arasında qarşılıqlı əlaqəni təmin edirlər.

Üçüncü səviyyə olan şəbəkə səviyyəsi paketlər adlanan ötürülən verilənlər blokları axınının idarə olunmasını təmin edir: paketləri ötürür və qəbul edir, növbəti ötürmə marşrutunu təmin edir. Paketlərin kommutasiyası rejimində ötürmə deytaqram və virtual kanalların köməyi ilə həyata keçirilə bilər. Deytaqramlı şəbəkələrdə hər bir deytaqramın tərkibində ötürücü və qəbuledicinin ünvanları yerləşir və hər bir paket qəbuledicyə müxtəlif yolla çata bilər. Daimi virtual kanal olmadıqda verilənləri ötürməzdən əvvəl ötürücüdən qəbuledicyə qədər virtual kanal təşkil etmək lazımdır. Verilənlərin ötürülməsi başa çatdıqdan sonra yaradılmış virtual kanal ləğv edilməlidir. Şəbəkə səviyyəsi aşağıdakı xidmətləri təqdim edir: şəbəkə birləşmələri, səhvlər haqqında xəbərdarlıq, birləşmənin açılması, axının idarə olunması və s. Paketlərin kommutasiyasına aid ən tanınan standart, TTBMK tərəfindən işlənmiş, X.25 standartıdır. Bu standarta şəbəkə səviyyəsindən başqa, fiziki və kanal səviyyələri də daxildir. X.25 standartı paketlərin kommutasiyası ilə işləyən şəbəkə ilə kompüterləri biri-biri ilə məntiqi birləşdirən virtual kanalların yaradılması, istifadəsi və ləğv edilməsini nəzərdə tutur.

Dördüncü səviyyə – nəqliyyat səviyyəsidir. O, iki qarşılıqlı əlaqədə olan sistemlər (bu sistemlərdə qarşılıqlı əlaqədə olan tətbiqi proseslər yerləşir) arasında informasiyanın etibarlı ötürülməsini təmin edir. Bu səviyyə yuxarı səviyyədən alınan hər hansı bir verilənlər blokunun nəqlini təmin edir. Nəqliyyat səviyyəsi şəbəkənin işləməsi, verilənlər bloklarının ölçüləri haqqında məlumata malikdir. Bu səviyyə aşağıdakı xidmətləri təklif edir: nəqliyyat birləşməsinin təyin edilməsi, verilənlərin ötürülməsi, nəqliyyat birləşməsinin ləğvi (X.224).

ASQƏ arxitekturasında beşinci səviyyə seans səviyyəsidir. Bu səviyyə seans yaradıldıqdan sonra iki tətbiqi proses arasında qarşılıqlı əlaqənin idarə olunmasını təmin edir. Bu səviyyə vasitəsi ilə tətbiqi proses şəbəkə ilə hərəkət edə bilər və digər tətbiqi prosesə ancaq seans birləşməsi vasitəsi ilə müraciət edə bilər. Qeyd etmək lazımdır ki, proses eyni zamanda bir neçə seans birləşməsini dəstəkləyə bilər. Seans səviyyəsi aşağıdakı xidmətləri təklif edir: seans birləşməsinin təyin edilməsi, seans birləşməsinin ləğv edilməsi, adi verilənlərin mübadiləsi, qarşılıqlı əlaqənin idarə olunması (X.225).

Altıncı səviyyə ötürülən informasiyanın sintaksisini təsvir edən və informasiyanı tətbiqi proseslərə verən təqdimedici səviyyədir. Bu səviyyə sintaksisin dəyişməsi, sintaksisin seçilməsi kimi xidmətlər təqdim edir.

Ötürülən informasiyanın semantikasını təyin edən səviyyə yeddinci tətbiqi səviyyədir. Bu səviyyə birbaşa olaraq tətbiqi proseslə bağlıdır və tətbiqi proseslərin ASQƏ mühitinə müraciətini təmin edir. Bu səviyyənin tərkibinə aşağıdakı səviyyələrdə olmayan, lakin açıq sistemlərin qarşılıqlı əlaqəsi üçün lazım olan funksiyalar daxildir. Tətbiqi səviyyə aşağıdakı xidmətləri təqdim edir: əlaqə qurmaq istəyən proseslərin hazır olmalarının təyini, səhvlərin aşkar olunması və verilənlərin tamlığının idarə olunması prosedurlarının cavabdehliyinin uyğunlaşması.

Beləliklə, qeyd olunduğu kimi, üç aşağı səviyyə kommunikasiya alt səbəkəsinə müraciəti təşkil edir və bu alt şəbəkə ilə informasiyanın ötürülməsini, deməli istənilən tətbiqi proseslərin birgə işini təmin edirlər. Üç yüxarı səviyyə tətbiqi proseslərin etibarlı və sərfəli qarşılıqlı əlaqələrini təmin edir. Üç yuxarı və üç aşağı səviyyələr arasında yerləşən orta, nəqliyyat səviyyəsi tətbiqi proseslərin növlərindən və kommunikasiya alt şəbəkəsinin növündən asılı deyil.

2.5. Standart lokal şəbəkələr

İlk lokal şəbəkələrin yaranmasından keçən vaxt ərzində yüzlərlə müxtəlif şəbəkə texnologiyaları işlənmişdir, lakin ancaq bir neçə şəbəkə geniş populyarlıq qazandı. Onların geniş yayılması səbəbi, birinci növbədə, bu şəbəkələrin məhşur firmalar tərəfindən dəstəklənməsi oldu. Standart şəbəkələr heç də həmişə ən yaxşı xüsusiyyətlərə malik deyillər, mübadilənin optimal rejimini təyin etmirlər, lakin onların avadanlığının böyük həcmdə istehsalı, bununla əlaqədar olaraq qiyməti aşağı olması, bu şəbəkələrə böyük üstünlüklər verir.

Bazarda bütün mümkün topologiyalara aid olan standart lokal şəbəkələr mövcuddur və istifadəçi üçün seçim yaranır. Standart şəbəkələr mümkün olan ölçü, şəbəkə abunəçilərin sayı və avadanlığa olan qiymətlər diapazonunu təmin edirlər. Buna baxmayaraq topologiyanın seçilməsi problemi çətin bir problem olaraq qalır. Standartlar ailəsində hər bir lokal şəbəkə özünə məxsus struktura, mühitə müraciət metodologiyası və ötürmənin fiziki mühitinə malikdir. Lokal şəbəkənin düzgün texnologiyasının seçilməsi üçün hər bir strukturun mənfi və müsbət cəhətlərinin öyrənilməsi məqsədəuyğundur.

Ethernet, Fast Ethernet şəbəkələri. Standart şəbəkələr arasında ən geniş yayılmış Ethernet şəbəkəsidir. Ethernet şəbəkəsində şin topologiyası istifadə olunur. Ethernet texnologiyası keçən əsrin 70-ci illərinin əvvəlində Xerox korporasiyasının tədqiqat mərkəzində işlənmişdir. Şəbəkə çox uğurlu alındı, və 1980-ci ildə DEC və Intel kimi məhşur firmalar onu dəstəklədilər. Bu firmalar sənaye standartının işlənməsi üçün öz səylərini birləşdirdilər və bu standart DIX Ethernet adlandırıldı. Bu firmaların səyləri nəticəsində 1985-ci ildə Ethernet standartı beynəlxalq standarta çevrildi və onu standartlara görə beynəlxalq təşkilatlar qəbul etdi: 802 IEEE komitəsi və ECMA. Standart IEEE 802.3 adını aldı. İlkin versiyada lokal Ethernet şəbəkəsi efir adlanan bir koaksial kabeldən ibarət idi və bu kabelə bir neçə kompüter qoşulurdu. Ethernet şəbəkəsinin kəsilməz koaksial kabelinin təyin edilməsi üçün mütəxəssislər seqment anlayışından istifadə edirlər. Ethernet şəbəkəsinin hər bir seqmentinin uzunluğu 500 metrdən çox olmamalıdır, standart hər bir birləşmə cütünün 3 metrdən az olmayan məsafədə yerləşməsini tələb edir. IEEE 802.3 standartının xarakteristikaları aşağıdakılardır:

- topologiya – şin;
- ötürmə mühiti – koaksial kabel;
- ötürmə sürəti – 10 *Mbit/s*;
- şinin maksimal uzunluğu – 5 *km*;
- abunəçilərin maksimal sayı – 1024-ə qədər;
- şəbəkə seqmentinin uzunluğu – 500 *m* qədər;
- bir seqmentdə adunəçilərin sayı – 100-ə qədər.

Klassik Ethernet şəbəkəsində əlli omluk iki növlü (yoğun və nazik) koaksial kabeldən istifadə olunur. Lakin son zamanlar ötürmə mühiti kimi burulmuş cütləri istifadə edən Ethernet versiyası geniş yayılmışdır. Şəbəkədə optik-lifli kabelin istifadəsi üçün də standart təyin olunub. Standarta uyğun əlavələr edilmişdir.

1995-ci ildə 100 *Mbit/s* sürəti ilə işləyən və ötürmə mühiti kimi burulmuş cüt və ya optik-lifli kabel istifadə edən Fast Ethernet standartı qəbul olunmuşdur. 1998-ci ildə 1000 *Mbit/s* sürəti ilə işləyən Gigabit Ethernet standartı qəbul olunmuşdur.

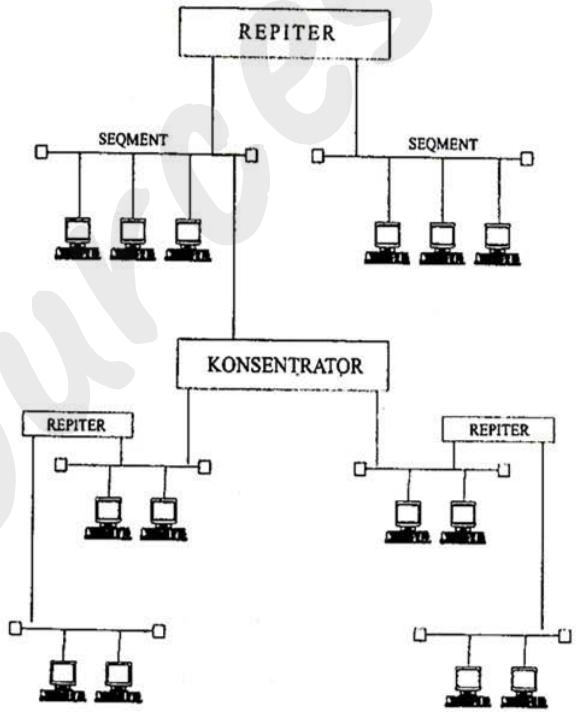
Standart şin topologiyasından başqa passiv ulduz və passiv ağac kimi topologiyalardan da istifadə olunur. Bu halda şəbəkənin müxtəlif hissələrini birləşdirən repiterlər və passiv konsentratorların istifadəsi nəzərdə tutulur (şəkil 2.14). Koaksial kabel şin seqmentləri, burulmuş cüt və optik-lifli kabel isə passiv ulduzda istifdə olunur. Əsas odur ki, alınmış topologiyada qapanmış yollar olmasın. Bütün şəbəkənin kabelinin maksimal uzunluğu 6,5 *km* ola bilər, təcrübədə isə bu göstərici 2,5 *km*-dən çox olmur.

Hal-hazırda 10 *Mbit/s* sürəti ilə işləyən Ethernet texnologiyasının fiziki spesifikasiyaları aşağıdakı verilənlərin ötürmə mühitlərini daxil edirlər:

- 10Base-5 – yoğun koaksial adlanan, diametri 0,5 *düym* olan koaksial kabel. Seqmentin maksimal uzunluğu – 500 *m*-dir.
- 10Base-2 – nazik koaksial adlanan, diametri 0,25 *düym* olan koaksial kabel. Seqmentin maksimal uzunluğu – 200 *m*-dir.
- 10Base-T – ekranlaşmamış burulmuş cüt əsasında kabel. Konsentrator əsasında ulduzvari topologiyanı təşkil edir. Konsentrator və son qovşaq arasında məsafə 100 *m*-dən çox olmamalıdır.
- 10Base-F – optik-lifli kabel. 10Base-T standartının topologiyasına oxşar topologiyadır.

Ötürmə mühitinin işarəsinə üç element daxildir. 10 rəqəmi 10 *Mbit/s* ötürmə sürətini bildirir. Base sözü əsas tezliklər zolağında ötürməni göstərir. Standartın adında olan axırıncı simvol kabelin növünü bildirir: T – burulmuş cüt, F – optik-lifli kabel. Bu standart 100 *Mbit/s* sürəti ilə işləyən Ethernet şəbəkəsi üçün də ötürmə mühitinin növlərini təyin edir.

Token Ring şəbəkəsi. Token Ring – IEEE təşkilatı tərəfindən standartlaşdırılmış, lokal şəbəkələrin digər bir arxitekturasıdır. O, Ethernet və digər şəbəkə texnologiyaları ilə ümumi xassələrə malikdir. Yaranma anından Token Ring texnologiyası bir sıra dəyişikliklərə məruz qaldı. Əvvəlcədən texnologiyada keçirtmə qabiliyyəti 4 *Mbit/s* idi, sonralar 16 *Mbit/s* oldu. Hal-hazırda siqnalların ötürmə sürətlərinin 100, 128 *Mbit/s* qədər, gələcəkdə isə 1 *Hbit/s* qədər artırılması üçün tövsiyələr mövcuddur.



Şəkil 2.14. Ethernet şəbəkəsinin topologiyası

Token Ring texnologiyası 1984-cü ildə IBM kompaniyası tərəfindən işlənmişdir, sonra isə standartın layihəsi kimi IEEE 802 standartına ötürülmüşdür. Onun əsasında 1985-ci ildə 802.5 standartı qəbul olunmuşdur.

Ethernet texnologiyasına nisbətən, Token Ring texnologiyası daha mürəkkəbdir. Bu texnologiya imtinalara qarşı

dayanıqlıq xassəsinə malikdir. Bəzi hallarda şəbəkənin işində aşkar olunan səhvlər avtomatik olaraq ləğv olunur. Başqa hallarda bu səhvlər ancaq qeyd olunur, onların düzəldilməsi isə işçi heyyət tərəfindən əl ilə aparılır.

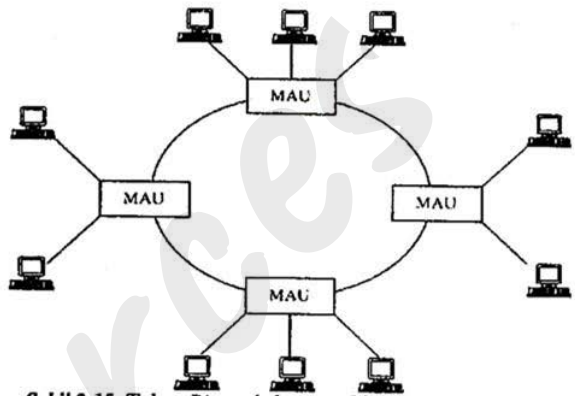
Ethernet avadanlığına nisbətən, Token Ring texnologiyasının avadanlığı hiss olunan dərəcədə bahadır, belə ki, mübadilənin idarə olunmasının daha mürəkkəb üsullarını istifadə edir. Məhz buna görə də Token Ring şəbəkəsi az yayılıb. Lakin mübadilənin yüksək intensivliyi və məhdud müraciət tələb olunduqda onun istifadəsi özünü doğruldur.

Token Ring şəbəkəsinin topologiyası halqadır, lakin xaricən ulduzu xatırladır. Bu onunla bağlıdır ki, ayrı-ayrı abunəçilər şəbəkəyə birbaşa deyil xüsusi konsentratorlar və ya çoxstansiyalı müraciət qurğuları vasitəsi ilə birləşirlər. Ona görə də şəbəkə fiziki olaraq ulduz-halqa topologiyasını təşkil edir (şəkil 2.15). Əslində isə, abunəçilər dairə şəklində birləşirlər, yəni abunəçinin hər biri özündən əvvəl gələn stansiyadan informasiyanı alır, özündən sonrakına isə ötürür.

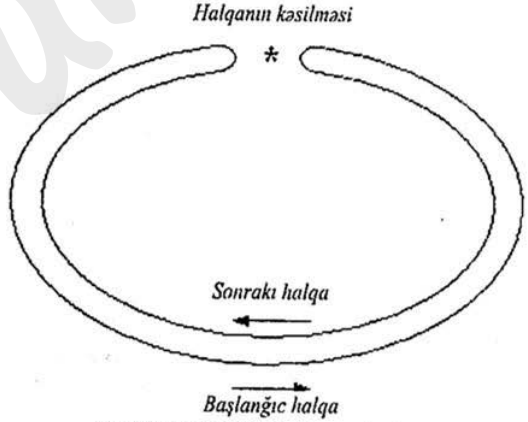
Token Ring şəbəkəsində ötürmə mühiti kimi əvvəllər burulmuş cüt istifadə olunurdu, lakin sonralar koaksial kabel və optiklifli kabel üçün avadanlıqların variantları yaranmışdır. Token Ring şəbəkəsinin əsas texniki xarakteristikaları aşağıdakılardır:

- konsentratorların maksimal sayı – 12;
- şəbəkə abunəçilərin maksimal sayı – 96;
- abunəçi ilə konsentrator arasında kabelin maksimal uzunluğu – 45 *m*;
- konsentratorlararası kabelin maksimal uzunluğu – 45 *m*;
- verilənlərin ötürülməsi sürəti – 4 *Mbit/s* və 16 *Mbit/s*.

Göstərilmiş xüsusiyyətlər ekranlaşmamış burulmuş cütün istifadəsi halına aiddir. Digər ötürmə mühiti istifadə olunduqda şəbəkələrin xarakteristikaları dəyişə bilər.



Şəkil 2.15. *Token-Ring şəbəkəsinin ulduz-halqa topologiyası*



Şəkil 2.16. *FDDI halqalarının imtina zamanı rekonfiqurasiyası*

Şəbəkənin mümkün ölçüsü və abunəçilərin maksimal sayına görə Ethernet şəbəkəsi, Token Ring şəbəkəsinə nisbətən, üstündür. Eyni zamanda Token Ring şəbəkəsi, Ethernet şəbəkəsindən fərqli olaraq, böyük yüklənməyə daha dayanıqlıdır və zəmanətli müraciət üsulunu təmin edir.

FDDI şəbəkəsi. Lokal şəbəkələrin ən köhnə və sərfəli texnologiyası optik-lifli kanalları ilə verilənlərin paylanmış interfeysidir (Fiber Distributed Data Interface, FDDI). Ötürmə mühiti kimi optik-lifin seçilməsi yeni şəbəkənin maniələrə qarşı yüksək dayanıqlıq, ötürülən informasiyanın maksimal məxfiliyi kimi üstünlükləri təyin etdi. FDDI şəbəkəsinin əsas texniki xarakteristikaları aşağıdakılardır:

- şəbəkə abunəçilərinin maksimal sayı – 1000;
- şəbəkənin dairəsinin maksimal uzunluğu – 20 *km*;
- şəbəkənin abunəçiləri arasında maksimal məsafə - 2 *km*;
- ötürmə mühiti – çoxmodalı optik-lifli kabel;
- müraciət üsulu – markerli;
- informasiyanın ötürmə sürəti – 100 *Mbit/s*.

Göründüyü kimi, FDDI texnologiyası əvvəl baxılan texnologiyalara nisbətən böyük üstünlüklərə malikdir. Qeyd etmək lazımdır ki, şəbəkənin ümumi uzunluğuna qoyulan məhdudiyyət kabeldə siqnalın sönməsi ilə deyil, mümkün olan müraciət vaxtının təmin edilməsi üçün siqnalın dairə üzrə tam keçməsi vaxtına qoyulan məhdudiyyətlə əlaqəlidir. Abunəçilər arasındakı məsafə isə məhz kabeldə siqnalın sönməsi ilə bağlıdır.

FDDI texnologiyasında imtinaların qarşısını almaq üçün izafilikdən istifadə olunur. FDDI dairəsi iki tam dairədən ibarətdir: onlardan biri avadanlıq düzgün işlədikdə verilənlərin ötürülməsi üçün istifadə olunur, digəri isə birinci dairə imtina etdikdə işə salınır (şəkil 2.16).

Aşkar üstünlüklərə baxmayaraq bu texnologiya geniş istifadə tapmayıb. Bu əsasən avadanlığın bahalılığı ilə əlaqəlidir. Halhazırda FDDI texnologiyasının istifadə sahəsi – bir neçə şəbəkəni birləşdirən baza şəbəkələridir. FDDI texnologiyası yüksəksürətli mübadilə tələb edən server və ya güclü işçi stansiyaların birləşməsi üçün də istifadə olunurr.

100VG-AnyLAN şəbəkəsi. 100VG-AnyLAN şəbəkəsi – bazarda peyda olan yüksəksürətli lokal şəbəkələrin son işlənmələridir. O, Hewlett-Packard və IBM firmaları tərəfindən işlənib və IEEE 802.1 standartına uyğundur, ona görə də onun standartlaşma səviyyəsi çox yüksəkdir. Bu texnologiyanın əsas üstün cəhətləri aşağıdakılardır: yüksək sürətli mübadilə, avadanlığın nisbətən ucuzluğu, munaqişələr olmadan mərkəzləşdirilmiş mübadilənin idarə olunması üsulu. Şəbəkənin adında 100 rəqəmi 100 *Mbit/s* sürətinə uyğun gəlir, VG hərfləri 3-cü kateqoriyalı ucuz burulmuş cütü bildirir, AnyLAN isə şəbəkənin iki ən yayımış şəbəkələrlə uyğunlaşmasını bildirir. 100VG-AnyLAN şəbəkəsinin əsas xarakteristikaları aşağıdakılardır:

- ötürmə sürəti – 100 *Mbit/s*;
- topologiya – genişlənmə imkan olan ulduz;
- müraciət üsulu – mərkəzləşdirilmiş, münaqişələr olmadan;
- ötürmə mühiti – dördqat ekranlaşmamış burulmuş cüt və optik-lifli kabel;
- konsentratorla abunəçi və konsentratorlararası kabelin maksimal uzunluğu – 100 *m*.

Beləliklə, 100VG-AnyLAN şəbəkəsinin parametrləri Fast Ethernet-in parametrlərinə çox yaxındır. Lakin Fast Ethernet-in əsas üstün cəhəti – ən geniş istifadə olunan Ethernet şəbəkəsi ilə tam uyğunlaşmadır. 100VG-AnyLAN şəbəkəsində iki mübadilə rejimi nəzərdə tutulub: yarımdupleksli və tamdupleksli. Yarımdupleksli mübadilə zamanı dörd burulmuş cütün hamısı eyni zamanda bir istiqamətdə ötürmə üçün istifadə olunurlar. Onlar paketlərin ötürülməsi üçün istifadə olunurlar. Tamdupleksli mübadilə zamanı iki burulmuş cüt bir istiqamətdə ötürməni yerinə yetirir, qalan ikisi isə digər istiqamətdə.

Çoxsaylı müsbət xüsusiyyətlərinə baxmayaraq 100VG-AnyLAN şəbəkəsi geniş istifadəçi kütləsini tapmadı və öz fəaliyyətini dayandırdı. O, standart şəbəkələrin heç biri ilə tam uyğunlaşma xüsusiyyətinə malik deyil. FDDI texnologiyasına nisbətən heç bir yüksək xüsusiyyətlərə malik deyil.

2.6. Müasir hesablama şəbəkələrinə qoyulan tələblər

Şəbəkələrə qoyulan əsas tələb – istifadəçilər üçün şəbəkəyə birləşmiş bütün kompüterlərin bölüşdürülmüş resurslarına müraciətin mümkünlüyünün təmin edilməsidir. Digər tələblər – məhsuldarlıq, etibarlılıq, uyğunluq, idarəolunma, mühafizə, genişlənmə və miqyaslama – əsas tələbin yerinə yetirilməsi keyfiyyəti ilə bağlıdır.

Tələblərin hər birinin vacib olmasına baxmayaraq, adətən kompüter şəbəkəsinin "xidmət keyfiyyəti" (Quality of Service, QoS) anlayışı "dar" mənada izah olunur. Buraya şəbəkənin ən əsas iki xarakteristikası – məhsuldarlıq və etibarlılıq – daxildir.

Şəbəkənin seçilmiş xidmət keyfiyyətinin göstəricisindən asılı olmayaraq onun təmin edilməsi üçün iki yanaşma mövcuddur. Birinci yanaşma, istifadəçi nöqteyi-nəzərindən, daha təbii görünür. Bu yanaşmaya görə şəbəkə tərəfindən (daha doğrusu, ona xidmət edən heyyət tərəfindən) istifadəçiyə təklif olunan xidmətin keyfiyyət göstəricisinin ədədi kəmiyyətinə riayət edilməsi zəmanətini verir.

İkinci yanaşmada şəbəkə istifadəçilərin üstünlüklərinə (prioritetlərinə) görə onlara xidmət edir. Yəni xidmətin keyfiyyəti istifadəçi və ya bir qrup istifadəçinin üstünlüyü dərəcəsindən asılıdır. Bu halda xidmətin keyfiyyətinə zəmanət verilmir, burada istifadəçinin ancaq üstünlük səviyyəsi təmin olunur. Belə xidmət *best effort* (böyük səylə) xidməti adlanır. Şəbəkə imkanı çərçivəsində istifadəçiyə keyfiyyətli xidmət təqdim etməyə çalışır, lakin bu halda heç bir şeyə təminat vermir. Bu prinsip, misal üçün, kadrların üstünlüklərinə görə kommutatorlar əsasında qurulmuş lokal şəbəkələrdə həyata keçirilib.

2.6.1. Məhsuldarlıq. Mümkün olan yüksək məhsuldarlıq – kompüter şəbəkələri aid olan paylanmış sistemlərin əsas xassələrindən biridir. Bu xassə şəbəkənin bir neçə kompüteri arasında işlərin paylanması imkanını təmin edir. Lakin əfsuslar olsun ki, bu imkanı hər dəfə istifadə etmək mümkün olmur.

Şəbəkənin məhsuldarlığının bir neçə əsas xarakteristikası mövcuddur:

- reaksiya vaxtı;
- keçirtmə qabiliyyəti;
- ötürmənin ləngiməsi və ötürmənin ləngiməsinin variasiyası.

Reaksiya vaxtı, istifadəçi nöqteyi-nəzərindən, şəbəkənin məhsuldarlığının inteqral xarakteristikasıdır. "Bu gün şəbəkə ləng işləyir" dedikdə, məhs bu xarakteristikanı nəzərdə tuturlar.

Ümumi halda reaksiya vaxtı istifadəçinin hər hansı bir şəbəkə xidmətinə olan sorğusunun yaranması ilə bu sorğuya cavabın

alınması arasında olan inteqral vaxt kimi təyin olunur.

Aydındır ki, bu göstəricinin qiyməti istifadəçinin müraciət etdiyi xidmətin növündən, hansı istifadəçinin hansı serverə müraciət etdiyindən, şəbəkənin elementlərinin (sorğu keçən seqment, kommutator və marşrutizatorların yüklənməsindən, serverin yüklənməsindən və s.) vəziyyətindən asılıdır.

Ona görə də şəbəkənin reaksiya vaxtının orta ölçülmüş qiymətinin istifadəsi məqsədəuyğundur.

Adətən şəbəkənin reaksiya vaxtı bir neçə tərkib hissədən ibarət olur. Ümumi halda onun tərkibinə aşağıdakılar daxildir: müştəri kompüterində sorğuların hazırlanması vaxtı; şəbəkə seqmenti və aralıq kommunikasiya avadanlıqları ilə müştəri və server arasında sorğuların ötürülməsi vaxtı; serverdə sorğuların emalı vaxtı; serverdən müştəriyə cavabın ötürülməsi vaxtı; serverdən alınan cavabların müştəri kompüterində emalı vaxtı.

Aydındır ki, reaksiya vaxtının tərkib hissələrə bölünməsi istifadəçini maraqlandırmır. Onu maraqlandıran son nəticədir. Lakin şəbəkə mütəxəssisi üçün ümumi reaksiya vaxtından verilənlərin şəbəkə emalı (şəbəkənin seqmentləri və kommunikasiya avadanlığı vasitəsi ilə müştəridən serverə verilənlərin ötürülməsi) mərhələlərinə uyğun gələn tərkib hissələrinin ayrılması vacibdir.

Reaksiya vaxtının tərkib hissələrinin məlum olması aşağıdakılara imkan verir: şəbəkənin ayrı-ayrı elementlərinin məhsuldarlığının qiymətləndirilməsi; "dar" yerlərin təyin edilməsi və şəbəkənin ümumi məhsuldarlığının artırılması üçün şəbəkənin modernizasiyasının yerinə yetirilməsi.

Keçirtmə qabiliyyəti şəbəkə və ya onun hissəsi ilə zaman vahidi ərzində ötürülən verilənlərin həcmini əks etdirir. Keçirtmə qabiliyyəti artıq istifadəçi xarakteristikası deyil, belə ki, o, şəbəkənin daxili əməliyyatlarının (müxtəlif kommunikasiya qurğuları vasitəsi ilə şəbəkənin qovşaqları arasında verilənlər paketlərinin ötürülməsi) yerinə yetirilməsi haqqında məlumat verir. Lakin o, şəbəkənin əsas funksiyasının yerinə yetirilməsi – xəbərlərin nəqliyyatı – keyfiyyətini xarakterizə edir və ona görə də şəbəkənin məhsuldarlığının təhlili zamanı, reaksiya vaxtına nisbətən, ona daha tez müraciət olunur.

Keçirtmə qabiliyyətinin ölçü vahidi ya saniyədə bir bit, ya

da saniyədə bir paketdir. Kecirtmə qabiliyyəti ani, maksimal və orta ola bilər.

Orta keçirtmə qabiliyyəti ötürülmüş verilənlərin ümumi həcmini onların ötürmə vaxtına bölərək təyin olunur. Qeyd etmək lazımdır ki, kifayət qədər böyük vaxt intervalı (saat, gün və ya həftə) seçilir.

Ani keçirtmə qabiliyyəti orta keçirtmə qabiliyyətindən kiçik vaxt intervalının seçilməsi ilə fərqlənir. Bu interval, misal üçün, 10 *ms* və ya 1 *s* ola bilər.

Maksimal keçirtmə qabiliyyəti – nəzarət dövrü ərzində qeyd olunmuş ən böyük ani keçirtmə qabiliyyətidir.

Səbəkənin layihələndirilməsi, sazlanması və optimallaşdırılması zamanı ən çox orta və maksimal keçirtmə qabiliyyətindən istifadə olunur. Şəbəkənin və ya onun ayrıca elementinin orta keçirtmə qabiliyyəti böyük vaxt intervalı ərzində (bu dövr ərzində böyük ədədlər qanununa uyğun olaraq trafikin ən yüksək və ən aşağı intensivliyi biri-birini kompensasiya edir) şəbəkənin işinin qiymətləndirilməsinə imkan verir. Maksimal keçirtmə qabiliyyəti şəbəkənin xüsusi iş dövrləri üçün səciyyəvi olan (məsələn, səhər saatlarında müəssisənin işçiləri eyni vaxtda şəbəkədə qeyd olunur və bölüşdürülmüş fayl və verilənlər bazalarına müraciət edirlər) pik yüklənmələrin öhdəsindən gəlməsi üçün şəbəkənin imkanlarını qiymətləndirməyə imkan verir.

Şəbəkənin istənilən iki qovşaq və ya nöqtəsi arasında (misal üçün, müştəri kompüteri və server arasında, marşrutizatorun giriş və çıxış portları arasında) keçirtmə qabiliyyətini ölçmək olar. Şəbəkənin təhlili və sazlanması üçün şəbəkənin ayrı-ayrı elementlərinin keçirtmə qabiliyyəti haqqında məlumatlı olmaq faydalıdır.

Qeyd etmək lazımdır ki, şəbəkənin müxtəlif elementləri tərəfindən paketlərin ardıcıl ötürülməsi xüsusiyyətinə görə şəbəkənin istənilən tərtibli yolunun ümumi keçirtmə qabiliyyəti marşrutun tərkib elementlərinin keçirtmə qabiliyyətindən ən minimal olana bərabər olacaq. Tərtibli yolun keçirtmə qabiliyyətinin artırılması üçün, birinci növbədə, ən ləng elementlərə fikir vermək lazımdır – bu halda belə element kimi marşrutizator çıxış edəcək. Qeyd etmək lazımdır ki, əgər tərtibli yol ilə ötürülən ən ləng elementin orta keçirtmə qabiliyyətindən üstün olan trafikin intensivliyi orta

olacaqsa, onda bu elementə olan paketlərin növbəsi, nəzəri olaraq sonsuzluğa qədər, praktiki olaraq isə, onun bufer yaddaşı dolana qədər artacaq. Sonra isə paketlər atılacaq və itəcəklər.

Bəzən **şəbəkənin ümumi keçirtmə qabiliyyətini** istifadə etmək məsləhətdir. Şəbəkənin ümumi kiçirtmə qabiliyyəti dedikdə, zaman vahidi ərzində şəbəkənin bütün qovşaqları arasında ötürülən informasiyanın ümumi həcmi başa düşülür. Bu göstərici şəbəkənin keyfiyyətini ayrı-ayrı seqmentlərə və ya qurğulara bölmədən, tam xarakterizə edir.

Adətən seqment və ya qurğunun keçirtmə qabiliyyətini ötürülən verilənlərlə təyin edərək hər hansı bir istifadəçi, əlavə və ya kompüterin paketləri ayrılmır – ötürülən informasiyanın ümumi həcmi hesablanır. Buna baxmayaraq, xidmətin keyfiyyətinin qiymətləndirilməsi üçün belə detallaşdırma məqsədəuyğundur. Son zamanlar şəbəkələrin idarəedilməsi sistemləri detallaşdırmanın yerinə yetirilməsinə icazə verir.

Ötürmənin ləngiməsi dedikdə, hər hansı bir şəbəkə qurğusu və ya şəbəkənin bir hissəsinin girişinə paketin daxil olması anı ilə bu qurğunun çıxışında paketin alınması anı arasında ləngimə başa düşülür. Məhsuldarlığın bu parametri mənaca şəbəkənin reaksiya vaxtına yaxındır, lakin onunla fərqlənir ki, o həmişə şəbəkənin kompüterləri ilə emalın ləngimələri olmadan verilənlərin emalının şəbəkə mərhələlərini xarakterizə edir. Adətən şəbəkənin keyfiyyəti **ötürmənin maksimal ləngiməsi** və **ləngimənin variasiyası** kəmiyyətləri ilə səciyyələnir. Trafikin növlərinin heç də hamısı ötürmənin ləngimələrinə, o cümlədən, kompüter şəbəkələri üçün səciyyəvi olan ləngimələrin kəmiyyətlərinə də həssas deyillər. Fayl xidməti, elektron poçt və ya çap xidməti ilə yaranan paketlərin ləngimələri şəbəkənin istifadəçisi nöqteyi-nəzərindən bu xidmətlərin keyfiyyətinə az təsir edir. Digər tərəfdən, səs verilənlərini və ya videotəsvirləri ötürən paketlərin eyni ləngimələri istifadəçiyə təqdim olunan informasiyanın keyfiyyətinin hiss olunan dərəcədə azalmasına gətirə bilər ("əks səda" effektinin yaranması, bəzi sözlərin araşdırılmasının mümkün olmaması, təsvirin əsməsi və s.).

Keçirtmə qabiliyyəti və ötürmənin ləngimələri müstəqil parametrlərdir, belə ki, şəbəkə, misal üçün, yüksək keçirtmə qabiliyyətinə malik ola bilər, lakin hər paketin ötürülməsi zamanı

xeyli ləngimələri daxil edə bilər. Belə effekti geostasionar peyk ilə təşkil olunmuş rabitə kanalı verir. Ötürmənin ləngiməsi 0,24 *s* təşkil edərək, bu kanalın keçirtmə qabiliyyəti çox yüksək ola bilər, misal üçün 2 *Mbit/s*, bu da siqnalın yayılması sürəti (300 000 *km/s*) və kanalın uzunluğu (72 000 *km*) ilə təyin olunur.

2.6.2. Etibarlılıq və təhlükəsizlik. Hesablama səbəkələri də aid olan paylanmış sistemlərin yaradılmasının ilk məqsədlərindən biri, ayrı-ayrı hesablama maşınlara nisbətən, yüksək etibarlılığın əldə edilməsi idi.

Etibarlılığın bir neçə aspektinin ayrılması vacibdir. Texniki qurğular üçün imtinalara gətirən işləmənin orta vaxtı, imtinanın ehtimalı, imtinaların intensivliyi kimi etibarlılıq göstəricilərindən istifadə olunur. Lakin bu göstəricilər iki vəziyyətdə (**işlək** və **qeyri-işlək**) ola bilən sadə element və qurğuların etibarlılığının qiymətləndirilməsi üçün istifadə oluna bilər. Çoxsaylı elementlərdən ibarət olan mürəkkəb sistemlər işlək və qeyri-işlək vəziyyətlərdən başqa, bu xarakteristikaları nəzərə almayan aralıq vəziyyətləri də ala bilərlər. Bununla əlaqədar olaraq mürəkkəb sistemlərin qiymətləndirilməsi üçün digər xarakteristikalardan istifadə olunur.

Hazırlıq və ya **hazırlıq əmsalı** (*availability*) sistemin istifadə oluna biləcəyi vaxtın bir hissəsini göstərir. Sistemin strukturuna izafiliyin daxil edilməsi yolu ilə hazırlıq yaxşılaşdırıla bilər: sistemin əsas elementləri bir neçə nüsxədə mövcud olmalıdır ki, hər hansı birinin imtinası nəticəsində sistemin fəaliyyətini digərləri təmin etsinlər.

Hər hansı bir sistemi yüksək etibarlı sistemlərə aid etmək üçün o, ən azı yüksək hazırlığa malik olmalıdır. Lakin bu kifayət deyil. Verilənlərin saxlanması və təhriflərdən qorunması da təmin olunmalıdır. Bundan başqa verilənlərin uyğunluğu da dəstəklənməlidir, misal üçün, əgər etibarlılığın artırılması üçün bir neçə fayl serverində verilənlərin bir neçə nüsxəsi saxlanılırsa, onda onların eyniliyini daimi təmin etmək lazımdır.

Şəbəkə son qovşaqlar arasında paketlərin ötürülməsi mexanizmi əsasında işlədiyi üçün etibarlılığın səciyyəvi xarakteristikalarından biri təyinat qovşağına paketin təhrif olunmadan **çatdırılması ehtimalıdır**. Bu xarakteristika ilə yanaşı digər göstəricilər də istifadə oluna bilər: paketin itmə ehtimalı (istənilən səbəbə görə: marşrutizatorun buferinin dolması, nəzarət cəminin üst-üstə düşməməsi, təyinat qovşağına işlək yolun olmaması və s.); ötürülən verilənlərin hər hansı bir bitinin təhrif olunması ehtimalı; itmiş paketlərin çatdırılmış paketlərə nisbəti.

Ümumi etibarlılığın digər aspekti **təhlükəsizlikdir** (*security*), yəni sistemin icazəsiz müraciətlərdən qorunması imkanı. Mərkəzləşdirilmiş sistemə nisbətən, paylanmış sistemdə bunu etmək çətindir. Şəbəkələrdə xəbərlər, xətlərə qulaq asma vasitələri asan qurula bilən, ümumitəyinatlı otaqlarda yerləşən rabitə xətləri ilə keçirlər. Digər həssas yer, fərdi kompüterlərin nəzarətsiz qoyulmasıdır. Bundan başqa, əgər şəbəkənin ümumi təyinatlı qlobal şəbəkələrə çıxışı varsa, onda şəbəkənin qorunma sistemi şəbəkədə qeyd olunmamış istifadəçilər tərəfindən pozula bilər.

Etibarlılığın digər xarakteristikası **imtinalara qarşı dayanıqlıqdır** (*fault tolerance*). Şəbəkənin imtinalara qarşı dayanıqlığı dedikdə, sistemin ayrı-ayrı elementlərinin imtinasını istifadəçidən gizlədilməsi imkanının olması başa düşülür. Məsələn, əgər verilənlər bazasının cədvəllərinin nüsxələri bir neçə fayl serverində saxlanılırsa, onda istifadəçilər onlardan birinin imtinasını hiss etməyə də bilər. İmtinalara qarşı dayanıqlı sistemlərdə sistemin hər hansı bir elementinin imtinası sistemin tam dayanmasına deyil, keyfiyyətin aşağı düşməsinə (deqradasiyaya) gətirir.

2.6.3. Genişlənmə və miqyaslama. Bəzən genişlənmə və miqyaslama anlayışlarını sinonim kimi qəbul edirlər, lakin bu düzgün deyil. Bu anlayışların hər biri müstəqil məna daşıyır.

Genişlənmə (*extensibility*) şəbəkənin ayrı-ayrı elementlərinin (istifadəçilərin, kompüterlərin, əlavələrin, xidmətlərin) nisbətən asanlıqla əlavə edilməsi, şəbəkənin seqmentlərinin uzunluqlarının artırılması, mövcud avadanlığın yeni, daha güclüsü ilə əvəz edilməsi deməkdir. Bu zaman qeyd etmək lazımdır ki, sistemin asanlıqla genişlənməsi bəzən müəyyən məhdudiyyətlər çərçivəsində təmin oluna bilər. Məsələn, yoğun koaksial kabelin bir seqmenti əsasında qurulmuş Ethernet lokal şəbəkəsi yaxşı genişlənməyə malikdir, yəni o, stansiyaların asan qoşulmasına imkan verir. Lakin bu şəbəkədə stansiyaların sayına (30 – 40) məhdudiyyət qoyulur. Şəbəkə seqmentinə daha çox stansiyaların (100-ə

qədər) qoşulması fiziki olaraq mümkün olsa da, bu halda şəbəkənin məhsuldarlığı xeyli aşağı düşür. Bu məhdudiyyətin olması pis miqyaslama şəraitində yaxşı genişlənmə əlamətidir.

Miqyaslama (*scalability*) onu bildirir ki, şəbəkə geniş çərçivədə qovşaqların sayı və əlaqələrin uzunluğunu artırmağa imkan verir və bu halda şəbəkənin məhsuldarlığı pisləşmir. Şəbəkənin miqyaslamasını təmin etmək üçün əlavə kommunikasiya avadanlığının istifadəsi və şəbəkənin xüsusi şəkildə strukturlaşması tələb olunur. Məsələn, kommutator və marşrutizatorlar əsasında qurulmuş və iyerarxik struktura malik olan çoxseqmentli şəbəkə yaxşı miqyaslamaya malikdir. Belə şəbəkəyə bir neçə min kompüter qoşula bilər və şəbəkənin hər bir istifadəçisinə lazımi keyfiyyətli xidmət təmin edilir.

2.6.4. Şəffaflıq. Şəbəkə mürəkkəb kabellər sistemi ilə birləşdirilmiş ayrı-ayrı kompüterlər çoxluğu kimi deyil, zamana görə bölünmə sistemi ilə vahid ənənəvi hesablama maşını kimi təqdim olunduqda şəbəkənin **şəffaflığı** (*transparency*) əldə edilir. Sun Microsystems kompaniyasının məhşur "Şəbəkə – bir kompüterdir" şüarı şəbəkənin məhz belə şəffaflığı haqqında bəhs edir.

Şəffaflıq iki müxtəlif səviyyədə əldə edilə bilər – istifadəçi səviyyəsində və proqramçı səviyyəsində. İstifadəçi səviyyəsində şəffaflıq onu göstərir ki, uzaqlaşdırılmış resurslarla işləmək üçün istifadəçi lokal resurslarla işləmək üçün istifadə etdiyi əmr və prosedurları istifadə edir. Proqram səviyyəsində şəffaflıq, əlavənin uzaqlaşdırılmış resurslara müraciət etməsi üçün lokal şəbəkələrə müraciət zamanı istifadə olunan eyni çağırışların istifadə olunması deməkdir. İstifadəçi səviyyəsində səffaflıq daha asan əldə edilir, belə ki, sistemin paylanmış xarakteri ilə bağlı olan prosedurların xüsusiyyətləri, əlavəni yaradan proqramçı tərəfindən maskalanır. Əlavə səviyyəsində şəffaflıq şəbəkə əməliyyat sisteminin vasitələri ilə paylanmış xüsusiyyətin bütün detallarının gizlədilməsini tələb edir.

Əməliyyat sisteminin xüsusiyyətlərini və kompüterlərin növləri arasında olan fərqləri şəbəkə gizlətməlidir. Macintosh kompüterinin istifadəçisi UNIX-sistemi ilə dəstəklənən resurslara müraciət etmə imkanına, UNIX-sistemi isə Windows 95 istifadəçiləri ilə informasiyanın bölüşdürülməsi imkanına malik olmalıdır. IBM 3270 terminalının istifadəçisi, çətin yadda saxlanılan ünvanların sirlərini bilmədən öyrənmədən, fərdi kompüterlər şəbəkəsinin istifadəçiləri ilə məlumatların mübadiləsi imkanlarına malik olmalıdır.

Şəffaflıq konsepsiyası şəbəkənin müxtəlif aspektlərinə tətbiq oluna bilər. Məsələn, yerləşmənin şəffaflığı istifadəçidən prosessorlar, çap qurğuları, fayllar və verilənlər bazaları kimi proqram və aparat resurslarının yerləşməsi haqqında məlumatın bilməsini tələb etmir. Resursun adında onun yerləşməsi haqqında məlumat qeyd olunmamalıdır. Eyniliklə, hərəkət şəffaflığı resursların bir kompüterdən digərinə öz adlarını dəyişmədən azad hərəkət etməsi deməkdir. Şəffaflığın digər mümkün olan aspektlərindən biri də paralelizmin şəffaflığıdır. Hesablamaların paralel yerinə yetirilməsi prosesi proqramçının iştirakı olmadan avtomatik yerinə yetirilir. Bu halda sistem şəbəkənin prosessor və kompüterlərinə görə əlavənin paralel budaqlarını özü bölüşdürür. Hal-hazırda demək olmaz ki, şəffaflıq xassəsi əksər hesablama şəbəkələrinə məxsusdur. Bu bir məqsəddir, və müasir şəbəkələri işləyən ixtisasçılar bu məqsədə nail olmağa çalışırlar.

2.6.5. Müxtəlif növ trafikin dəstəklənməsi. Əvvəlcədən kompüter şəbəkələri istifadəçilərin kompüter resurslarına (fayllar, çap qurğuları və s.) birgə müraciət etməsi üçün nəzərdə tutulmuşdur. Kompüter şəbəkələrinin bu ənənəvi xidmətləri ilə yaranan trafik özünə məxsus xüsusiyyətlərə malikdir, və müəyyən dərəcədə telefon şəbəkələrində və ya, məsələn, kabel televiziyası şəbəkələrində olan, məlumatlar trafikindən fərqlənir. Lakin 90-cı illər kompüter şəbəkələrinə multimedia verilənləri (rəqəmsal şəkildə nitq və videotəsvirlər) trafikinin daxil olması illəri oldu. Kompüter şəbəkələrindən videokonfransların təşkili, videofilmlər əsasında öyrətmə və əyləncə və s. üçün istifadə olunmağa başladılar. Təbiidir ki, multimedia trafikinin dinamik ötürülməsi üçün digər alqoritm və protokollar və, uyğun olaraq, digər avadanlıq tələb olunur. Multimedia trafikinin faizi çox böyük olmasa belə, o həm qlobal, həm də lokal şəbəkələrə daxil olmağa başladı və, aydındır ki, bu proses artan sürətlə davam edəcək.

Səs və təsvirin dinamik ötürülməsi zamanı yaranan trafikin

əsas xüsusiyyəti ötürülən xəbərlərin sinxronluğuna sərt tələblərin qoyulmasıdır. Kəsilməz proses olan səs rəqslərin və ya videotəsvirdə işığın intensivliyinin dəyişməsinin keyfiyyətli yerinə yetirilməsi üçün siqnalların ölçülmüş və kodlaşdırılmış amplitudları alınmalıdır. Siqnalların amplitudları ötürücü də tərəfdə ölçülən tezliyə bərabər olmalıdır. Xəbərlərin ləngiməsi təhriflərə rast gəlinəcək.

Eyni zamanda xəbərlərin sinxron çatdırılmasına qoyulan sərt tələblər olmadıqda kompüter verilənlərinin trafiki şəbəkəyə gələn xəbərlərin qeyri-bərabər intensivliyi ilə xarakterizə olunur. Məsələn, uzaqlaşdırılmış diskdə mətnlə işləyən istifadəçinin müraciəti uzaqlaşdırılmış və lokal kompüterlər arasında təsadüfi xəbərlər axınını yaradır ki, bu axın mətnin redaktə edilməsi ilə bağlı istifadəçinin hərəkətlərindən asılıdır. Qeyd etmək lazımdır ki, müəyyən çərçivədə çatdırılma zamanı ləngimələr şəbəkənin istifadəçisinin xidmətolunması keyfiyyətinə az təsir edirlər. Kompüter əlaqələrinin bütün alqoritmləri, uyğun protokol və kommunikasiya avadanlıqları məhz "nəbz" xarakterli trafik üçün nəzərdə tutulmuşdur. Ona görə də, multimedia trafikinin ötürülməsi tələbi həm protokol, həm də avadanlığın prinsipial dəyişikliklərinin aparılmasını tələb edir. Hal-hazırda, praktiki olaraq bütün yeni protokollar bu və ya digər səviyyədə multimedia trafikinin dəstəklənməsini təqdim edir.

2.6.6. İdarə olunma. Şəbəkənin idarə olunması aşağıdakıları nəzərdə tutur: şəbəkənin əsas elementlərinin vəziyyətlərinə mərkəzləşdirilmiş nəzarət imkanı; şəbəkənin işi zamanı yaranan problemlərin aşkar edilməsi və həlli; məhsuldarlığın təhlilinin yerinə yetirilməsi; şəbəkənin inkişafının planlaşdırılması. İdeal halda şəbəkələrin idarə olunması vasitələri şəbəkənin hər bir elementinin (ən sadə elementdən ən mürəkkəb qurğulara kimi) müşahidəsi, nəzarət və idarə olunması sistemidir. Bu zaman sistem şəbəkəni ayrı-ayrı qurğuların pərakəndə toplusu kimi deyil, vahid qurğu kimi qəbul edir.

Yaxşı idarəetmə sistemi şəbəkəni müşahidə edir və, hər hansı bir problemi aşkar etdikdə, müəyyən əməliyyatı aktivləşdirir, vəziyyəti düzəldir və baş verən hadisə və görülən tədbir haqqında şəbəkənin administratorunu məlumatlandırır. Bununla

yanaşı idarəetmə sistemi müəyyən verilənlər toplamalıdır ki, bu verilənlər əsasında şəbəkənin inkişafının planlaşdırılması mümkün olsun. Nəhayət, idarəetmə sistemi istehsalçıdan asılı olmamalıdır və bir konsoldan bütün əməliyyatları yerinə yetirməyə imkan verən rahat interfeysə malik olmalıdır.

Taktiki məsələləri həll edərkən, şəbəkə administratorları və texniki heyyət hər gün şəbəkənin iş qabiliyyətinin təmin edilməsi problemləri ilə qarşılaşırlar. Bu məsələlər cəld müdaxilə tələb edirlər. Şəbəkəyə xidmət edən heyyət istifadəçilər və ya şəbəkənin idarə olunmasının avtomatik vasitələrindən gələn nasazlıqlar haqqında məlumatlara operativ cavab verməlidir. Tədricən strateji yanaşma (yəni şəbəkənin planlaşdırılması) tələb edən məhsuldarlığın, şəbəkənin konfiqurasiya edilməsinin, imtinaların emalı və verilənlərin təhlükəsizliyi ilə bağlı daha ümumi problemlər qabarır. Bundan başqa, planlaşdırmaya şəbəkəyə qoyulan istifadəçi tələblərinin dəyişməsi proqnozu, yeni əlavələ və yeni şəbəkə texnologiyaların tətbiqi məsələləri və s. daxildir.

İdarəetmə sisteminin faydalılığı ən çox böyük şəbəkələrdə (korporativ və içtimai qlobal şəbəkələrdə) özünü biruzə verir. İdarəetmə sistemi olmadıqda hər şəhərin hər binasında qurulmuş şəbəkə avadanlığının istismarı üzrə yüksək təcrübəli mütəxəssislərin olması vacibdir. Buna görə də çox saylı xidməti heyyət ştatının saxlanılması labüddür.

Hal-hazırda şəbəkənin idarə olunması sahəsində çox sayda həll olunmamış problemlər mövcuddur. Şəbəkənin idarəolunması üçün rahat, kompakt və çoxprotokollu vasitələr kifayət qədər deyil. Mövcud olan vasitələrin əksəriyyəti şəbəkəni idarə etmirlər, onlar ancaq şəbəkənin işinə nəzarət edirlər.

2.6.7. Uyğunlaşma. Uyğunlaşma və ya inteqrallaşdırma onu bildirir ki, şəbəkənin tərkibinə müxtəlif proqram və aparat təminatı daxildir, yəni onun tərkibində müxtəlif kommunikasiya steklərini dəstəkləyən müxtəlif əməliyyat sistemləri mövcud ola bilər, və müxtəlif istehsalçıların aparat vasitələri və əlavələri işləyə bilər. Müxtəlif novlü elementlərdən ibarət olan şəbəkə qeyri-bircins və ya heterogen adlanır. Heterogen şəbəkə problemsiz işləyirsə, o, inteqrallaşdırılmış şəbəkə hesab olunur. İnteqrallaşdırılmış şəbəkələrin qurulmasının əsas yolu – açıq standart və spesifikasiyalara uyğun olaraq yerinə yetirilmiş modulların istifadəsidir.

III FƏSİL

EHM-lərin PROQRAM TƏMİNATI

3.1. EHM-lər üçün proqramlar

EHM-də ixtiyari informasiya proqramlar vasitəsilə tədqiq olunur. Bunun üçün kompüterin qəbul edəcəyi dildə, informasiyanı emal etmək üçün lazım olan dəqiq və ətraflı göstərişlər ardıcıllığını (proqramı) tərtib etmək lazımdır. Öz-özlüyündə heç bir kompüter onun tətbiq edildiyi sahələr üzrə heç bir biliklərə malik deyildir. Bütün bu biliklər kompüterlərdə yerinə yetirilən proqramlarda cəmlənmişdir. Kompüter üçün müxtəlif proqramlar tətbiq etməklə, onu mühəndis və ya mühasibin, aqronom və ya riyaziyyatçının iş yerinə çevirmək olar. Buna görə də kompüterdən effektiv istifadə etmək üçün, onunla iş üçün lazım olan proqramların təyinat və xassələrini bilmək lazımdır.

Hal-hazırda EHM-in hər bir tipi üçün çoxlu sayda proqramlar işlənib hazırlanmışdır. EHM-in işlədilməsi üçün zəruri olan prqramlar külliyatına EHM-in poqram təminatı deyilir. EHM-də istifadə olunan əsas tip proqramlar aşağıdakılardır:

1) *Sistem proqramları.* Bu proqramlar EHM-in işlədilməsi və onlara texniki qulluq, EHM-də ixtiyari konkret məsələnin həlli zamanı hesablama prosesinin təşkili və idarə edilməsi və s. üçün nəzərdə tutulub. Onlara əməliyyat sistemləri, əməliyyat sistemlərini örtükləri, utilit-proqramlar, antivirus proqramları və s. aiddir.

2) *Proqramlaşdırma sistemləri.*

3) *Alət proqramlar.* Bu proqramlardan gündəlik fəaliyyətdə sənədlərin hazırlanması üçün bir alət kimi istifadə olunur. Onlara mətinlərlə iş üçün nəzərdə tutulan proqramlar (mətn redaktorları), qrafik redaktorlar, elektron cədvəllər (cədvəl prosessorları), verilənlər bazasını idarəetmə sistemləri, özündə bir neçə alət proqramı birləşdirən mühitlər və s. aiddir.

4) *Mətnlərin avtomatik tərcümə proqramları*

5) *Tədris proqramları*

6) *Tətbiqi proqramlar*

7) *Oyun proqramları*

8) *Multimedia proqramları.*

Əvvəlcə sistem proqramına baxaq. Əməliyyat sistemləri və əməliyyat sistemlərinin proqram örtükləri sistem proqramlarının əsas növləridir. Bu vacib proqramlarsız müasir EHM-in işi mümkün deyildir. Birinci növbədə bu əməliyyat sistemlərinə aiddir.

Əməliyyat sistemi – məsələlərin həlli prosesində və istifadəçi ilə EHM arasında qarşılıqlı əlaqənin təşkili, EHM-in bütün vasitələrindən effektiv istifadə üçün nəzərdə tutulmuş proqramlar kompleksidir. Fərdi EHM-də əməliyyat sistemi xüsusilə vacib rol oynayır, çünki məhz o, fərdi EHM-lə işi sadə və əlverişli edir. IBM PC tipli EHM-lər üçün də bir çox tip əməliyyat sistemi mövcuddur. Onlardan hər biri öz tətbiq sahəsinə malikdir.

IBM PC tipli kompüterlər üçün yaradılan ilk əməliyyat sistemi Microsoft firmasının MS DOS sistemi idi. Bu sistem ən etibarlı və sadə sistemdir. Sonrakı illərdə firma tərəfindən bu tipli kompüterlər üçün MS DOS-un əsasında Windows 3.1 (3.11) sistemi yaradıldı. Bu əməliyyat sistemi üzərində iş aparılaraq onun daha mükəmməl variantları Windows 95, Windows 98, Windows 2000, Windows NT, Windows XP, Windows Vista meydana gəldi.

Beləliklə, əməliyyat sistemi istifadəçilər ilə EHM arasında əlaqə yaradıb, istifadəçiyə EHM-i idarə etmək imkanını verir. Lakin sistem əmrlərin köməyi ilə EHM-i idarə etmək çətinliklər yaradır, bu çətinlikləri aradan qaldırmaq üçün proqram örtüklərindən istifadə olunur. Praktiki olaraq hər bir əməliyyat sistemi üçün proqram örtükləri mövcuddur. Təkcə MS DOS üçün bu cür proqramlardan bir neçə tip hazırlanmışdır. Onlardan ən geniş yayılanları Norton Commander, Norton Navigator, XTree, Norton Desktop, PC ToolsDesktop, MS DOS Shell və başqalarıdır.

Sistem proqramlarına utilit proqramları adlanan proqramlar da aiddir. Bu proqramlar istifadəçi üçün xeyirli olan müxtəlif əməliyyatlar yerinə yetirir. Bunlar, məsələn disklərdəki verilənləri sıxlaşdıraraq onların tutduğu həcmi kiçildən arxivator proqramlar, EHM-dəki verilənləri qoruyan və bərpa edən proqramlar, disklərdəki informasiyanı optimal yerləşdirən proqramlar və s. Utilitləri adətən müəyyən proqram paketlərində birləşdirirlər. Belə paketlərdən ən çox istifadə olunanı və güclüsü Norton Utilities adlanır. Bu paketdən olan və MS DOS üçün nəzərdə

tutulmuş Ndd (Norton Disk Doctor) proqramı, eləcədə ona uyğun Windows 95 tərkibində olan Scandisk proqramı diskləri (disketləri) onların məntiqi sisteminin düzgünlüyü nöqteyi-nəzərdən testləşdirməyə, disklərin səthindəki zədəli sektorların təyin edilməsinə və bir çox əməliyyatların aparılmasına imkan verirlər. Antivirus proqramlar da utilit proqramlardandır. Virus proqram mətni təhrif edərək, onu tamamilə işə yaramaz hala gətirə bilir. Virus duşmüş proqram, başqa proqramları da bu virusa yoluxdura bilir. Yoluxmaya əsasən *com, exe* ad genişlənmələri olan proqram faylları məruz qalır. Mətni fayllar, proqramlaşdırma dillərində tutulmuş proqram mətnləri, sənəd mətinləri və s. fayllar virusa yoluxmur, onlarda yalnız mətn təhrif oluna bilir. Virusa yoluxmuş EHM-də bir çox lazımlı proqram paketləri sıradan çıxa bilir, lazımı fayllar silinə bilir və viruslarla mübarizə çox zaman tələb edir. Çox saylı viruslarla effektli mübarizə üçün antivirus proqramları yaradılır. Belə proqramların bəzi tiplərini qeyd edək:

1) Həkim – proqramlar, proqramları «müalicə» edir, onları əvvəlki formaya qaytarır və viruslardan təmizləyir;

2) Filtr – proqramlar, əməliyyat sistemlərinə daxil olmağa çalışan virusların qabağını alaraq, onlar haqqında istifadəçiyə məlumat verirlər.

Ən geniş yayılmış antivirus proqramlarına periodik olaraq yeniləşən və tamamlanan AIDSTEST, DRWEB (DOCTOR WEB) proqramlarını göstərmək olar. İnternet şəbəkəsi ilə ötürülən və yeniləşən güclü antivirus proqramlarına misal olaraq AVP (Anti Viral ToolKit Pro), Kasperski və s. göstərmək olar.

Texniki qulluq proqramlarının köməyi ilə kompüter sistemləri testləşmədən keçirilir, tapılmış çatışmamazlıqlar aradan qaldırılır, EHM-in bəzi qurğularının işi optimallaşdırılır.

Konkret tip EHM üçün konkret alqoritmik dildə proqramların yaradılıb, işlədilməsi üçün nəzərdə tutulmuş proqram və digər vasitələr kompleksinə proqramlaşdırma sistemləri deyilir. Proqramlaşdırma sistemi adətən proqramlaşdırma dilinin müəyyən versiyasından, bu dildə verilən proqramların translyatorundan və s. ibarət olur. Hər bir proqramlaşdırma sistemi ilə müəyyən proqramlaşdırma dili bağlı olduğundan, dillər haqqında məlumatlar verək.

Proqramlaşdırma dili – kompüter proqramlarının yaradılması üçün bir alətdir. Çoxlu saylı dillər arasında hal-hazırda geniş yayımış aşağıdakı dilləri qeyd etmək olar:

1) birinci növbədə sistem proqramlarının hazırlanması üçün istifadə olunan C++ dili;

2) tətbiqi proqramların hazırlanmasında geniş istifadə olunan Turbo Pascal dili;

3) təzə başlayan porqramçılar üçün nəzərdə tutulan QBASIC dili.

Hal-hazırda bu dillərin müasir genişləndirilmiş versiyaları olan Visual Basic, Visual C++, Delphi (Visual Pascal) geniş istifadə olunur.

Alət proqramlara birinci növbədə mətn redaktorları, qrafik redaktorlar, cədvəl prosessorları və verilənlər bazasının idarəetmə sistemləri aiddir.

Kompüterlər üçün yaradılan ilk proqramlardan biri, mətnlərin emalı proqramları və ya mətn redaktorları idi. İlk redaktorlar simvolların daxil edilməsi, redaktə edilməsi, alınmış mətnin printerdə çap edilməsini və s. təmin edirdi. Müasir mətn redaktorları isə bununla yanaşı, müxtəlif ölçülü çoxlu sayda şriftlərdən istifadəni, orfoqrafiya və sintaksisin yoxlanılıb düzəldilməsini, təkrarlanan sözlərin sinonimləri ilə əvəz olunmasını, mətnə cədvəl və diaqramların daxil edilməsini və s. təmin edir. Bir çox müasir mətn redaktorları WYSIWYG – What You See Is What You Get («Ekranda nə görürsünüzsə, kağızda da onu alacaqsınız») prinsipini realizə edirlər. Yəni sənəd ekrana real şəkildə çıxarılır, bu isə onun redaktə edilməsini sadələşdirir. Mətn emalının müasir proqramları öz imkanlarına görə bir neçə kateqoriyaya bölünür:

1) Mətnlərin hazırlanması proqramları. Bu kateqoriyaya NC mətn redaktorunu, Bloknot redaktorunu aid etmək olar;

2) Mətn prosessorları. Onlar sənədlərin, məqalələrin, məktubların və s. hazırlanmasını təmin edirlər, onlara misal olaraq MultiEdit, MS Word və s. göstərmək olar;

3) Stolüstü nəşretmə sistemləri; Məsələn, Wentura, Page Maker və s.

Qrafik redaktorlar – şəkil və digər təsvirlər yaradır, onları redaktə etməyə imkan verən proqramlardır. Qrafik redaktorlar iki əsas tipə – nöqtə və vektor tipli redaktorlara bölünür. Nöqtə

redaktorlar, təsviri nöqtələr üzrə qurur, hər bir nöqtə üçün öz rəngi verilir. Vektor redaktorlar, isə bütün xətt üçün verilmiş rənglə xətt, qövs və ya əyri xətti qura bilir. Birincilərə misal olaraq MS Paint, Adobe PhotoShop, ikincilərə misal olaraq isə Adobe Illustrator (7.0 versiyası), Corel Draw göstərmək olar. Hər tip redaktorların öz tətbiq sahələri, öz üstünlükləri və çatışmamazlıqları var. Vektor redaktorları, şəkildə çox mürəkkəb forma çevirmələri aparmağa imkan verir. Buna şəklin sıxılması, dartılması, şəklin ixtiyari elementinin, şəkli təhrif etmədən ixtiyari bucaq qədər döndərilməsi və s. aiddir. Bu redaktorlarda məhz şəkil çəkmək, təsvirləri ixtiyari formada yerləşdirilmiş müxtəlif növ yazılarla uyğunlaşdırmaq əlverişli olur. Onlardan bütün tip reklamların, məhsul nişanlarının və s. hazırlanmasında geniş istifadə olunur. Nöqtə redaktorlardan skanerdən çıxan təsvirlərin – fotoşəkillərin, şəkillərin və s. emalı zaman istifadə olunur. Bu cür redaktorlarla təsvirlərdəki nöqsanlar düzəldilir, rəng və rəng çalarları, kontrastlıq, dəqiqlik, işıqlıq və s. dəyişdirilə bilir. Son zamanlar üçölçülü təsvir redaktorları da meydana gəlib, onlar üçölçülü obyektlərin yaradılmasına imkan verir. Belə redaktorlara misal olaraq 3D Studio-Max, TrueSpace 2 və s. göstərmək olar.

Elektron cədvəllər (cədvəl prosessorları) mühasibat hesabatlarını, elmi təcrübələrin nəticələrini və ümumiyyətlə cədvəl şəklində verilən ixtiyari informasiyanın emalını təmin edən proqramlardır. Müasir cədvəl prosessorları özündə mətn redaktorunun, elektron kalkulyatorun, proqramlaşdırma mühitinin imkanlarını birləşdirən proqramlardır. Onlar hesablama nəticələrini müxtəlif formalı diaqramlarla, şəkillərlə təqdim etməyə, sənədlərdə qrafikadan və digər imkanlardan istifadə etməyə şərait yaradır. Hal-hazırda ən çox istfadə olunan cədvəl prosessorlarına misal olaraq SuperCalc 5.0, MS Excel göstərmək olar.

Hal-hazırda verilənlərin emalı proqramları geniş tətbiq olunur. Xüsusilə də əsasını verilənlər bazası təşkil edən avtomatlaşdırılmış informasiya sistemləri (AİS), informasiya-arayış sistemləri və s. qeyd etmək olar. Burada verilənlər bazasında milyonlarla sətirlərdə yerləşdirilmiş verilənlər çoxluğundan istifadəçini bu anda maraqlandıran verilənləri seçib, redaktə edib, tam bir sənəd şəklində təqdim etmək tələb olunur. Bu mürəkkəb məsələləri həll etmək üçün xüsusi proqramlaşdırma dilləri və hətta xüsusi

proqramlaşdırma sistemləri (verilənlər bazasını idarəetmə sistemi-VBİS) nəzərdə tutulmuşdur. Belə sistemlər adətən proqramlaşdırma dilindən, translyatordan və proqramlaşdırma mühitindən ibarətdir. VBİS bazada olan verilənlər əsasında sənədləri, cədvəlləri tez və dəqiq hazırlamağa imkan verir. Ən çox istifadə edilən sistemlər Foxpro, MS Access, Lotus1-2-3 və s.

Prenzentasiya proqramları geniş kütlə qarşısında çıxışlar üçün nəzərdə tutulmuş slaydların və digər nümayiş materiallarının yaradılmasında istifadə olunur (məsələn, MS PowerPoint). Riyazi proqramlar proqramlaşdırmadan istifadə etmədən riyazi məsələlərin çox geniş sinfini həll etməyə imkan verir (məsələn, Maple, MathCad).

Multimedia proqramlarına səsyazma proqramları, səs və video faylların redaktorları, musiqi sintezatorlarının proqramları və s. aiddir. Nitq proqramları mətnin EHM-ə nitq vasitəsilə daxil edilməsini, sənədlərin məzmununa qulaq asılmasını və s. təmin edir.

Özündə bir neçə alət proqramını birləşdirən mühitə misal olaraq Microsoft Works 3.0 (4.0) mühitini göstərmək olar. Bu mühit mətn prosessorunu, elektron cədvəlini, verilənlər bazasının daxil edilməsi və yaradılması proqramını, qrafik redaktoru özündə birləşdirir. Bir mühitdə müxtəlif növ proqramların birləşdirilməsi mühitdən kənara çıxmadan praktik olaraq ixtiyari sənədlərin hazırlanmasına, ehtiyac olduqda bir proqramdan digərinə keçməyə şərait yaradır. Bundan əlavə verilənləri bir proqramdan digərinə keçirmək olar. Bu cür mühitlərə misal olaraq MS Office, Lotus SmartSuite və s. göstərmək olar.

Mətnlərin avtomatik tərcümə etmə proqramları müəyyən bir dildə verilən mətnlərin başqa bir dilə tərcümə edilməsi üçün nəzərdə tutulmuşdur. Bu proqramlar ayrı-ayrı söz və söz birləşmələrinin tərcüməsini ani olaraq və səhvsiz yerinə yetirir, lakin bütöv bir abzas və bir neçə abzasdan ibarət mətnin düzgün tərcüməsini bu proqramlarla praktik olaraq yerinə yetirmək mümkün olmur. Lakin bununla yanaşı bu proqramlar xarici dili heç bilməyən şəxslər üçün, xarici dildə verilmiş mətn haqqında hər hansı bir anlayış almaq üçün, elektron poçtun qısa məlumatlarının tərcüməsi üçün əlverişli olur. Bu proqramlara misal olaraq Sokrat, Stylus,Promt 98 sistemlərinin göstərmək olar. Sistemlər elmin

müxtəlif sahələri üzrə böyük lüğətlərə, mətnlərin avtomatik tərcümə etmə proqramlarına, Web-səhifələrin sinxron tərcümə proqramlarına və s. malikdir.

Tətbiqi proqramlar elm və texnikanın bu və ya digər sahələrinin tətbiqi məsələlərinin həlli üçün nəzərdə tutulub. Bu proqramlara misal olaraq mühasibat proqramlarını, nəşretmə sistemlərini, maliyyə təhlili proqramlarını, avtomatik proyektləşdirmə sistemlərini, verilənlərin statistik təhlili proqramlarını və s. göstərmək olar.

Tədris proqramları müxtəlif fənlərin öyrədilməsində geniş istifadə olunur. EHM-dəki proqramlarla iş qaydalarını öyrədən tədris proqramları da mövcuddur. Tədris proqramları arasında uşaqların yaradıcılıq qabiliyyətlərini üzə çıxarmağa imkan verən inkişafetdirici proqramları xüsusi qeyd etmək lazımdır.

Son illərdə kompüter oyun proqramları sahəsində çox böyük nəticələr alınmışdır. Bu proqramlarda kompüterin qrafik imkanlarından, üçölçülü təsvirlərdən geniş istifadə olunur.

Multimedia, EHM-dən mətnin, stereosəsin, səs müşayətinin, yüksək keyfiyyətli qrafikanın, videokliplərin, animasiyanın, hətta virtual reallığın tətbiqi ilə istifadə qaydalarına deyilir. Başqa sözlə multimedia EHM-in rəqəmsal və mətni informasiyanın səs siqnalları və video siqnallarla birləşdirilməsi vasitəsidir. Mutimedialı kompüterdə səs stereoplatası, videomaqnitafon, videokamera, rəqəmsal fotokamera, televizorla iş üçün video daxiletmə platası, CD-ROM-la iş üçün diskovod, səs stereokolonkaları, mikrofon və tələb olunan proqram təminatı olmalıdır. Hal-hazırda multimedia üçün zəruri olan bütün proqramları özündə birləşdirən proqram paketləri (məsələn, Multimedia Kit) hazırlanır.

3.2. Fayl və kataloq anlayışları

Maqnit disklərdəki ixtiyari informasiya fayllar şəklində saxlanılır. Hər bir proqram da fayl (File) təşkil edir. Proqramda istifadə olunan ədədi və digər verilənlər, fayllar şəklində saxlanıla bilər.

Faylları çox vaxt iki qrupa – mətni və ikilik fayllarına bölürlər. Mətn faylları insan tərəfindən oxunması üçün nəzərdə

tutulub. Məsələn, bu səhifədə yazılan mətn də fayl şəklində maqnit diskə yazıla bilər və bu fayl mətni fayl olacaqdır. Mətn fayllarında eləcədə proqramların mətnləri və s. yerləşdirilə bilər. Mətn faylı olmayan yerdə qalan bütün mümkün fayllar adətən ikilik faylları adlanır.

Hər bir fayl - faylın adı (file name) ilə işarələnir. Faylın adı bir-birindən nöqtə ilə ayrılan iki hissədən ibarət olur. Birinci hissə - faylın əsas adı, birdən səkkizə qədər simvola malik ola bilər. İkinci hissə isə - faylın adının genişlənməsi (file name extention) üçə qədər simvola malik ola bilər.

Məsələn, DOS sisteminin əsas proqramının faylının adı COMMAND. COM, (COMMAND – əsas ad, COM isə faylın adının genişləndirilməsidir).

Faylın adında latın hərflərindən və kompüterdəki digər simvollardan, rəqəmlərdən istifadə etmək olar. Lakin faylın adını bir-birindən probellə ayrılmış simvollarla vermək olmaz. *Məsələn, paper.doc, autoexec.bat, FILE10.BAS* və s. fayl adları düzgün, *FILE10.TEXT, 2< >1.TXT* və s isə düzgün deyildir.

Fayl adının genişləndirilməsi fayldakı informasiyanın xarakteri haqqında məlumat verir və buna görə də ondan istifadə əlverişli olur. Bundan əlavə, bir çox proqramlar (məsələn, Norton Commander) faylın adının genişləndirilməsinə əsaslanaraq, uyğun proqramı çağırıb, verilən faylı ora yükləməyə imkan verir, bu isə vaxta xeyli qənaət etmək deməkdir. Bəzi fayl adının genişləndirilmələrini qeyd edək:

.com, . exe - yerinə yetirilən fayllar;
.bat - komanda (Batsh) faylları;
.bas - BASIC dilində olan proqramlar;
.pas - PASCAL dilində olan proqramlar;
.cpp – C++ dilində olan proqramlar;
.bak - dəyişdirilməzdən əvvəl faylın yaradılan təkrarı;
.txt - mətn fayllar və s.

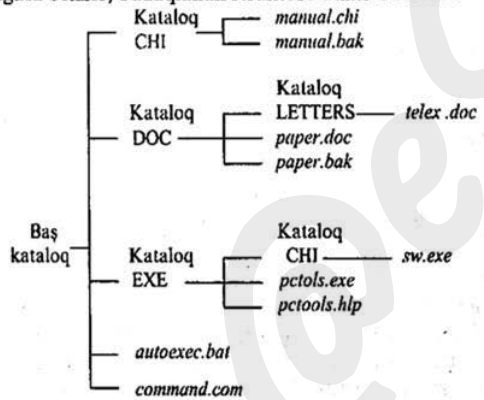
Qeyd edək ki, fayllar üzərində iş zamanı faylın *.bak* genişləndirilməsindən istifadə çox əlverişli olur. Faylın belə bir təkrarının olması, fayl üzərində işləyərkən, əgər müəyyən informasiya səhvən dəyişdirilərsə və ya pozularsa, onu yenidən bərpa etməyə imkan verir. Fayl üzərindəki işi bitirdikdən sonra isə, faylın təkrarını ləğv etmək olar.

Kompüterin işə salınması ilə verilən əməliyyat sistemlərindən başqa, ixtiyari proqram öz tərkibində, bu proqramı işə salan fayla malikdir. Belə fayl yerinə yetirilən fayl adlanır. Başqa sözlə yerinə yetirilən fayl, proqramın baş faylı olub, onun yerinə yetirilməsini təmin edir. Əgər proqram yeganə bir fayldan ibarətdirsə, onda elə bu fayl yerinə yetirilən fayldır. Bu cür fayllar adətən *.com* və *.exe* genişləndirilmələrinə malik olurlar.

Diskdəki bütün faylların siyahısı diskin kataloqu adlanır. Kataloq - diskdə, faylların adları, onların ölçüləri, axırıncı dəfə təzələnməsi vaxtı, xassələri və s. haqqında məlumat olan bir sahədir. Əgər kataloqda hər hansı bir faylın adı varsa, onda deyirlər ki, bu fayl həmin kataloqda yerləşir. Hər bir diskdə bir neçə kataloq ola bilər. Hər bir kataloqda isə çoxlu sayda fayl ola bilər, lakin hər bir fayl yalnız bir kataloqda qeyddən keçir. Hər bir kataloqun öz adı vardır və hər bir kataloq digər bir kataloqun daxilində qeyddən keçə bilər. Əgər X kataloqu Y kataloqunda qeyddən keçirsə, onda X kataloqu Y kataloqunun alt kataloqudur.

Kataloqların adlarına olan tələblər elə fayllarda olduğu kimidir. Lakin kataloqlarda adətən ad genişləndirilməsindən istifadə olunmur.

Hər bir diskdə bir baş kataloq olur. Bu kataloqda fayllar və alt kataloqlar və ya birinci səviyyə kataloqları qeyd olunur. Birinci səviyyə kataloqlarda isə öz növbəsində fayllar və ikinci səviyyə kataloqları qeyd olunur və s. Nəticədə diskdəki kataloqların ağaca bənzər, budaqlanan strukturu alınır. *Məsələn:*



Şəkil 3.1

Kompüterdə işləyən şəxsin hal-hazırki anda işlədiyi kataloqa, cari kataloq deyilir. Məsələn, Windows və ya Norton Commander-də işlərkən, ekranda cari kataloqun tərkibi haqqında məlumat verilir.

DOS sistemində cari kataloqun başlığını vermək üçün Dir komandasını vermək lazımdır, bu kataloqdan digərinə keçmək üçün isə CD komandasından istifadə olunur. Norton Commander, Windows sisteminin fayllar dispetçerində, Windows 95-in bələdçisində və s. isə cari kataloqun dəyişdirilməsi, bir kataloqdan digərinə keçərkən avtomatik yerinə yetirilir.

Ümumiyyətlə, fayllar sistemini kitabxana ilə müqayisə etmək olar:

maqnit diski - kitablar olan rəf;
fayl - kitab;
faylın adı - kitabın adı;
diskin kataloqu - kitabların siyahısı, kataloqu.

Cari kataloqdan olmayan faylla işləyərkən, bu faylın hansı kataloqdan olduğunu göstərmək lazımdır. Bunu isə fayla keçid yolunu göstərməklə etmək olar. Fayla keçid yolu - kataloqların adları və ya « \ » simvolu ilə ayrılan «..» simvolları ardıgıllığından ibarətdir. Bu yol - cari kataloqdan və ya diskin baş kataloqundan, axtarılan fayl yerləşən kataloqa olan istiqaməti göstərir. Əgər yol « \ » simvolundan başlayırsa, istiqamət diskin baş kataloqundan götürülür. Məsələn, tutaq ki, cari kataloq DOC götürülüb (şəkil 2.1.), onda

Baş kataloqdan başlayan yol	Cari kataloqdan başlayan yol	
\CHI	..\CHI	(1)
\DOC\LETTERS	LETTERS	(2)
\EXE\CHI	..\EXE\CHI	(3)

Burada (1) birinci səviyyə SHI kataloquna olan yolu, (2) DOC kataloqunun alt kataloqu olan LETTERS kataloquna olan yolu, və (3) EXE kataloqunun alt kataloqu olan CHI kataloquna olan yolu göstərir.

Kompüterdə adətən bir neçə, diskovod adlanan və bərk disklərdə, disketlərdə, kompakt-disklərdə və s. olan informasi-

yanı yığan və özündə saxlayan disklər olur. Onların hər birində kataloq və fayllar yerləşdirilə bilir. Bizə lazım olan diski çağırmaq üçün, əvvəlcə uyğun diskovoda onun adı ilə müraciət etmək lazımdır. Adətən, diskovodlar A, B, C, D, E və s. kimi adlandırılırlar. Məsələn, kompüterdə iki A və B diskovodları, maqnit elastik disklər, yəni disketlər üçün, bir C diskovodu isə bərk maqnit diski (vinçester) üçün nəzərdə tutula bilər. A və B diskovodları xarici yaddaşda, yəni disketlərdə olan informasiyanı oxumaq üçün, C diskovodu isə adətən əməliyyat sistemlərini kompüterə yükləmək üçün istifadə olunur.

Hal-hazırda işlənən diskovoda - cari diskovod deyilir. Məsələn, Windows və ya Norton Commander sistemində işlərkən ekranda cari diskovoddakı, kataloqun tərkibi verilir.

Beləliklə, bunları nəzərə almaqla, faylın tam adını aşağıdakı şəkildə ifadə etmək olar:

(diskovod:) (Yol \) faylın adı; (harada mötərizə daxilində verilməsi vacib olmayan elementlər göstərilir).

Faylın adı, faylın yerləşdiyi kataloqa olan yol, bundan « \ » simvolu ilə ayrılan faylın adı və bunların hamısının qarşısında diskovodun adından ibarətdir. Əgər diskovodun, adı verilmirsə, onda cari diskovodun, əgər kataloqun yolu göstərilmirsə, onda cari kataloqun olduğu başa düşülür.

Məsələn, tutaq ki, şəkil 3.1-də A diskovodunda olan fayl sistemi təsvir olunub və cari kataloq, burada - A:\ DOC şəklindədir. Onda

a: paper.doc - A diskovodundakı, diskin cari kataloqunun *paper.doc* faylını göstərir, *a:\ paper.doc* isə A diskovodundakı, diskin baş kataloqunun *paper.doc* faylını göstərir və s.

Eyni bir kataloqdan olan fayllar qrupunu göstərmək üçün faylların adlarında «*» və «?» simvolarından istifadə etmək olar. Belə ki, «*» simvolu faylın adında və ya onun adının genişdirilməsində ixtiyari sayda və ixtiyari simvolları göstərir. «?» simvolu isə faylın adı və adının genişləndirməsində ixtiyari bir simvolun olduğunu və ya olmadığını göstərir.

Məsələn: **.bak*-cari kataloqdakı.bak genişdirilməsinə malik,bütün fayları , C*.d* - cari kataloqdakı adı C ilə, adının genişləndirilməsi d ilə başlayan bütün faylları göstərir.

3.3. MS DOS əməliyyat sistemi

MS DOS əməliyyat sistemi 1981-ci ildə Microsoft firması tərəfindən, IBM firmasının sifarişi ilə, o vaxtlar yaradılan IBM PC kompüterləri üçun yaradılmışdır. Hal-hazırda 1994-cü ildə hazırlanmış MS DOS 6.22 versiyasından istifadə olunur. MS DOS əməliyyat sistemi, kompüterə, kompüter elektrik şəbəkisinə qoşulmaqla və ya EHM-nin korpusundakı Reset düyməsini basmaqla avtomatik yüklənir.

DOS sistemi işə hazır olduqda, ekrana məsələn: A> və ya C:\> şəklində dəvət verir. Bu dəvətdə adətən, cari diskovod və cari kataloq haqqında məlumat olur. Məsələn: A:\> yəni A diskovodu və baş kataloq, C:\EXE yəni C diskovodu və \EXE kataloqu. Əmrləri vermək üçün, bu əmri klaviaturadan əmr sətrinə yığıb *Enter* düyməsini basmaq lazımdır.

Mətn faylarının yaradılması. Bunun üçün copy con *faylın adı* – əmrini vermək lazımdır. Bundan sonra faylın sətrlərini daxil etmək lazımdır. Hər bir sətrin sonunda *Enter* düyməsini, mətnin axırıncı sətrindən sonra isə əvvəlcə *F6*, sonra isə *Enter* düyməsini basmaq lazımdır. Nəticədə diskdə göstərdiyimiz adlı fayl yaranacaqdır.

Faylların ləğv edilməsi. Bunun üçün aşağıdakı əmr verilir:

del *faylın adı*

Burada faylın adında * və ? simvollarından da istifadə olunur. *Məsələn:* del **.bak* – cari kataloqdan bak fayl ad genişlənməsinə malik bütün fayllar çıxarılır.

Faylların adlarının dəyişdirilməsi. Bunun üçün aşağıdakı əmr verilir:

ren *faylın adı1 faylın adı2*

Burada *faylın adı1*, adı dəyişdiriləcək faylı, *faylın adı2* isə bu fayla veriləcək yeni adı göstərir. Məsələn: ren *fox.doc fox.txt* – cari kataloqdakı *fox.doc* faylının adı *fox.txt* adı ilə dəyişdirilir.

Faylların təkrarının alınması. Bunun üçün aşağıdakı əmr verilir:

copy *faylın adı1 faylın adı2*

və ya

copy *faylın adı1 (kataloqun adı2)*

Burada faylın adı 1 parametri təkrarı alınacaq faylı, faylın adı 2 isə bu təkrarı alınan faylın yeni adını göstərir. Faylın

göndərildiyi kataloqu, kataloqun adı 2 parametri ilə, ya da faylın adı 2 parametrində kataloqun da adını verməklə təyin etmək olar. Əgər faylın təkrarının göndərildiyi kataloq göstərilmirsə, onda həmin fayl təkrarı cari kataloqa göndərilir və nəhayət əgər faylın adı 1 parametrində kataloqun adı göstərilirsə, onda fayl bu kataloqdan götürülərək təkrarı alınır, əks halda bu fayl cari kataloqdan götürülür. Burada faylın adında * və ? simvollarından istifadə etmək olar.

Məsələn: copy *c1.doc c1.txt* -cari kataloqdakı *c1.doc* faylının təkrarı alınır və oradakı *c1.txt* adlı faylında yerləşdirilir; copy A: \ *.* - A diskinin baş kataloqundakı bütün faylların təkrarı alınaraq cari kataloqa köçürülür və s.

Faylın digər kataloqa göndərilməsi. Bunun üçün aşağıdakı əmrdən istifadə olunur:

move *faylın adı kataloqun adı*

Burada faylın adında * və ? simvollarından istifadə edilə bilər. Bu əmr yerinə yetirilərkən adı göstərilən fayl, adı göstərilən kataloqa göndərilir. Məsələn: move *.doc d: -cari kataloqdakı .doc ad genişlənməsinə malik fayllar, d diskindəki cari kataloqa göndərilir.

DOS-da kataloqlarla iş. Cari diskovodu dəyişdirmək üçün, cari olacaq diskovodun adını yığıb, iki nöqtə simvolunu qoymaq lazımdır. *Məsələn:* A: , B: və ya C: və s. Bu komandalar verildikdən sonra *Enter* düyməsi basılmalıdır.

Cari kataloqun adına dəyişdirmək üçün aşağıdakı komandadan istifadə etmək olar:

move *kataloqun adı kataloqun yeni adı*

Məsələn: move *vin vin1* - cari kataloqun *vin* adı *vin1*-ə dəyişdirilir.

Yeni kataloq yaratmaq üçün aşağıdakı əmrdən istifadə olunur;

md *(diskovod): yol*

Məsələn: md S1 - cari kataloqda S1 adlı alt kataloq yaradılır; md a: \ S2 - A diskovodunun baş kataloqunda S2 adlı alt kataloq yaradılır və s.

Boş kataloqu ləğv etmək üçün aşağıdakı əmrdən istifadə olunur;

rd *(diskovod): yol*

Məsələn: rd *r1* - cari kataloqdakı *r1* alt kataloqu ləğv edilir; rd *a:* \ *r2* – A diskovodundakı baş kataloqda olan *r2* alt kataloqu ləğv edilir.

Cari kataloqu dəyişdirmək üçün aşağıdakı əmrdən istifadə olunur:

cd *(diskovodun adı): yol*

Əgər burada diskovodun adı verilirsə, onda cari kataloq bu diskovodda, əks halda isə cari diskovodda dəyişdirilir. *Məsələn:* cd \ – cari diskin baş kataloquna keçidi, cd\exe\doc - bu isə /exe/doc kataloquna keçidi göstərir və s.

Kataloqun tərkibini nəzərdən keçirmək üçün isə aşağıdakı əmrdən istifadə olunur: dir *diskovod: yol/faylın adı* parametrlər.

Burada faylın adında «*» və «?» simvollarından istifadə etmək olar. Əgər faylın adı verilmirsə, kataloqda olan bütün fayllar haqqında, əks halda isə adı göstərilmiş fayl və ya fayllar haqqında məlumat verilir. Əgər burada diskovod və yol göstərilmirsə, onda cari diskovoddakı, cari kataloq haqqında məlumat verilir.

dir əmri parametrlərsiz verildikdə, bu əmr hər bir fayl üçün onun adını, ad genişləndirilməsini, faylın baytlarla ölçüsünü, bu faylın yaradıldığı və ya axırıncı dəfə təzələndiyi tarixi göstərir. *dir* əmri eləcədə parametrlərlə də verilə bilər. Belə parametrlər çoxdur, bunlardan bəzilərini qeyd edək:

/p - bu parametr verildikdə, kataloqun tərkibi ekrana səhifə-səhifə verilir, yəni bu parametr verildikdə, DOS sistemi, ekran kataloqdakı məlumatla dolduqdan sonra gözləmə rejiminə keçir və klaviaturadakı ixtiyari bir düymə basıldıqda növbəti səhifəyə keçir.

/w - bu parametr verildikdə isə, yalnız faylların və alt kataloqların adları verilir (alt kataloqların adı kvadrat mötərizə daxilində verilir), buna verilənlərin geniş formatda verilməsi deyilir.

/ o - bu parametr verildikdə isə, əvvəlcə alt kataloqların adları əlifba sırası ilə, sonra isə faylların adları əlifba sırası ilə verilir.

Misallar: *dir* - cari kataloqun tərkibi verilir; *dir*.exe* - cari kataloqdakı *.exe* ad genişlənməsi olan fayllar haqqında məlumat verilir; dir a:\ - A diskovodundakı baş kataloqun tərkibi verilir;

dir /p - cari kataloqun tərkibi ekrana səhifə-səhifə verilir; *dir/w* - cari kataloqun tərkibi geniş formatda verilir və s.

Kompüterdə tarix və zaman haqqında informasiyanın verilməsi və dəyişdirilməsi. Tarix haqqında informasiyanın verilməsi və onun dəyişdirilməsi üçün aşağıdakı əmrdən istifadə olunur:

date

Bu əmr ilə ekrana il, gün və ay haqqında məlumat verilir. Əgər tarix dəyişdirilməyəcəksə, *Enter* düyməsini basmaq lazımdır. Yeni tarix vermək üçün ayın gününü (1 - 31), ilin ayını (1-12) və ili və ya ilin axırıncı iki rəqəmini vermək lazımdır. Burada MS DOS-da onların hansı ardıcılıqla veriləcəyi göstərilir (DD - gün, MM - ay, YY - il), ədədlər arasında isə «-» işarə qoyulur. *Məsələn:*

enter new date (dd-mm-yy): 21-04-1998

Zaman haqqında məlumat almaq və onu dəyişdirmək üçün aşağıdakı əmr var:

time (saat, dəqiqə),

harada saat 0-dan 24-ə qədər ədədlər, dəqiqə isə 0-dan 59-a qədər ədədləri göstərir. Əgər əmr parametrsiz, yəni time kimi verilirsə, onda DOS cari zamanı verir və yeni zamanı qoymağı tələb edir. Əgər zamanı dəyişdirməyə ehtiyac yoxdursa, onda *Enter* düyməsini basmaq lazımdır.

Məsələn: time - cari zaman verilir, time 11:45 - 11:45 zamanı qoyulur.

DOS-da ekran və printerlə iş qaydaları. Mətn faylını ekrana çıxarmaq üçün

type *faylın adı*

əmrindən istifadə olunur. Məsələn: type *k1.doc* – cari kataloqdakı *k1.doc* faylı ekrana verilir.

Faylın ekrana çıxarılmasını dayandırmaq üçün *ctrl* düyməsini basıb saxlamaqla *C* düyməsini basmaq lazımdır. Bu düymələri təkrarən basmaqla, faylın ekrana çıxarılmasını davam etdirmək olar. Faylın ekrana çıxarılmasını dayandırmaq üçün isə *Ctrl* və *C* düymələrini və ya *Ctrl* və *Break* düymələrini basmaq kifayətdir.

Monitorun ekranını informasiyadan təmizləmək üçün aşağıdakı əmrdən istifadə olunur:

cls

Bu əmr yerinə yetirilərkən monitorun ekranı təmizlənir və ekranın birinci sətrində DOS sisteminə dəvət haqqında məlumat verilir.

Mətn faylını çap etmək üçün aşağıdakı əmrdən istifadə olunur:

copy / b *faylın adı* prn

Bu əmr verilməmişdən qabaq, çap qurğusu işə hazır vəziyyətdə olmalıdır.

3.4. Windows əməliyyat sistemi

Windows əməliyyat sistemi – müxtəlif təyinatlı proqramların mürəkkəb kompleksidir. Bura Windows sisteminin iş rejimlərinin və onların müxtəlif parametrlərinin quraşdırılması proqramları, fayllarla iş proqramları, alət prqoramları və s. daxildir. Windows sisteminə daxil olan bəzi proqramları qeyd edək:

1) *Kalkulyator* (Calc.exe). Bu proqram onluq, ikilik, onaltılıq ədədlərlə işləməyə və onları birini digərinə keçirməyə imkan verir.

2) *Bloknot* (NotePad.exe). Kiçik mətni fayllarla iş üçün nəzərdə tutulmuş mətn redaktorudur.

3) *Lazer səsləndiricisi* (CdPlayer.exe). Kompakt disklərin səs fayllarını səsləndirməyə imkan verir.

4) *Fonoqraf* (Sndrec32.exe). Mikrofon vasitəsilə daxil edilən səs fayllarını yazmağa, səsləndirməyə və redaktə etməyə imkan verir.

5) *Bələdçi* (Explorer.exe). Fayllarla bütün mümkün əməliyyatları aparmağa, ixtiyari proqramları işə salmağa və s. imkan verir.

6) *Öyrədici proqram* (WinTutorial.exe). Windows-da işi öyrədən proqram.

Bunlardan əlavə ScanDisc proqramı bərk diskdəki ola biləcək səhvləri yoxlayır və aradan qaldırır, DriveSpace proqramı diskdəki verilənləri sıxlaşdırır və əlavə boş yerlər ayırır.

Windows-un tərkibinə WordPad mətn redaktoru, Paint qrafik redaktoru, elektron poçt, faksimil məlumatlar qəbul edib, göndərmək qabiliyyəti olan (Exchange.exe) proqram, İnternet şəbəkəsində işi təmin edən MS Explorer proqramı da daxildir.

Windows sistemində işlə bağlı bəzi əsas anlayışları verək:

1) **Sənəd anlayışı.** Yadda saxlanılması tələb olunan ixtiyari informasiya verilən adlanır. Bütün verilənlər isə proqramlara və sənədlərə bölünür. Sənədlər – proqramlarla yaradılan və emal edilən verilənlərdir. Hər iki tip verilənlər fayllarda saxlanılır. Sadəlik üçün proqram saxlanılan fayl «proqram», sənəd saxlanılan fayl isə «sənəd» adlandırılır.

2) **Qovluq anlayışı.** Qovluq daxinə digər qovluqlar, fayllar və digər obyektlər yerləşdirilə bilən konteyner rolunu oynayır.

3) **Pəncərə anlayışı.** Pəncərə ekranın düzbucaqlı çərçivə ilə məhdudlaşdırılmış və onu ayrıca ekran kimi işlətməyə imkan verən hissəsidir. Ekranda eyni zamanda bir neçə pəncərə yerləşdirilə bilər, lakin onlardan yalnız biri – cari (aktiv) adlanan pəncərə ilə işləmək olur. Hər bir açılan proqram, qovluq və sənəd öz pəncərəsində yerləşdirilir. Ekranda pəncərələrin ölçülərini, vəziyyətini və bir-birinə nəzərən qarşılıqlı vəziyyətini dəyişdirmək olur.

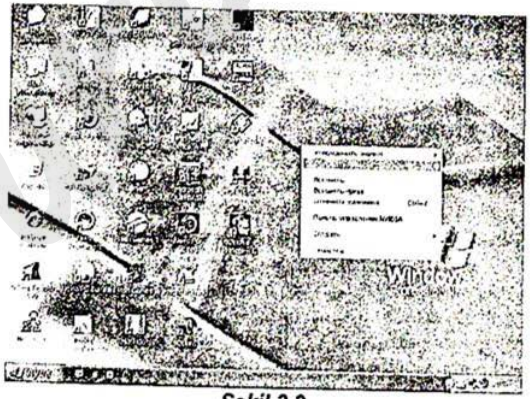
Windows sisteminin əsas elementləri aşağıdakılardır: fayl sistemi, istifadəçinin əməliyyat sistemi ilə ünsiyyətini təmin edən qrafik örtük, periferik qurğuların qoşulma və tənzimləmə sistemləri, əməliyyat sisteminin nizamlama proqramları, arayışlar sistemi, Windows sisteminə daxil olan tətbiqi və xidməti proqramlar külliyatı.

Fayl sistemi. Windows-da fayl sistemi MS DOS sistemində olan təyinata malikdir və onun genişlənməsidir. Fayl sisteminin əsas anlayışları fayl və qovluqdur. Fayl – artıq qeyd etliyimiz kimi ad ilə işarələnən və xarici yaddaş daşıyıcılarında (bərk və ya elastik disk, CD-ROM) saxlanılan verilənlər külliyatıdır. Windows-da fayl MS DOS-da verdiyimiz xassələri özündə saxlayır, bundan əlavə ümumiləşmə xatirinə buradakı qurğular da (printer, Lpt1, Lpt2, Com1 və s. qoşquları) fayl kimi qəbul olunur. Fayl MS DOS-da olduğu kimi əsas ad və ad genişlənməsi ilə ifadə olunur və xüsusi nişanla işarə edilir. Nişan – kiçik simvoldan, şəkildən ibarətdir. Windows-da hər bir obyektin (fayl, qovluq) adı öz nişanı ilə təmin edilir, bu isə obyekti daha tez «tapmağa» imkan verir. MS DOS-dan fərqli olaraq Windows-da fayllara uzun adlar vermək olur. Lakin fayl adında MS DOS-da olduğu kimi ən çoxu 8, ad genişlənməsində isə ən çoxu 3 simvol vermək daha əlverişli olur. Ad genişlənmələri kimi MS DOS-da olduğu standart işarələmələrdən (*exe, com, bat, bmp, doc* və s.)

istifadə olunur.

Qovluq bildiyimiz kataloq, alt kataloq anlayışlarının analoqu olub faylların, başqa qovluqların yerləşdirilməsi üçün nəzərdə tutulub. Hər bir qovluğa ad verilir və hər bir qovluq xüsusi nişanla işarələnir. Burada da faylın tam adı, ünvanı, fayla gedən yol anlayışları saxlanılır.

Sistemin qrafik örtüyü – istifadəçiyə sistemlə sadə, əlverişli ünsiyyət və idarəetmə imkanları verir. Bu örtük isttifadəçi interfeysi adlanır. İstifadəçi interrfeysinin imkanlarına baxaq. Windows sistemi EHM-ə yüklәndikdə ekrana sistemin iş stolu adlanan pəncərə (şək. 3.2) verilir.



Şəkil 3.2

Bu pəncərə sisteminin əsas pəncərəsidir, burada sistemin əsas idarəetmə elementləri yerləşir. Onlara məsələlər lövhəsi, *«Пуск»* (baş menyu) düyməsi, kontekst menyusu, sistem nişanları külliyatı aiddir. Bundan əlavə istifadəçi iş stoluna qovluqlar, proqramlar, sənədlər və müxtəlif obyektlərin nişanları (yarlıklarını) yerləşdirə bilər. Məsələlər lövhəsi adətən, pəncərənin aşağı hissəsində yerləşdirilir. Bu lövhənin sol kənarında *«Пуск»* düyməsi və bir-birinin ardınca bir neçə düymə yerləşdirilə bilər. Sağ kənarında isə cari zaman və klaviaturadakı şrift seçimi (rus, ingilis, azərbaycan) indikatorları yerləşir. Qeyd edək ki, tələb olunduqda burada digər indikatorlar da yerləşdirilə bilər. *«Пуск»* düyməsi ekrana baş menyunu çağırmaqla bütün fayllara, sistemin

imkanlarına, onun tənzimlənməsi vasitələrinə, EHM-in işinin sona çatdırılmasına keçidi təmin edir. Aktiv sənəd və proqramların düymələri *«Пуск»* düyməsindən sağda yerləşir və həmin obyektlərə keçidi (bu düymələri sıxmaqla) təmin edir.

Sistem nişanlarına aşağıdakılar aiddir:

1) **Mənim kompüterim (Мой компьютер).** Bu nişan EHM-in ixtiyari fayl və qovluqlarına (bərk və elastik disklər, CD-ROM, printer və s.), sistemi idarəetmə və tənzimləmə vasitələrinə və s. keçidi təmin edən qovluq açır.

2) **Zibil qutusu (Корзина).** Bura ləğv olunan fayl və qovluqlar köçürülür. Səhvən ləğv olunan obyektləri bərpa edərək bu qutudan çıxarmaq olur.

3) **Şəbəkə göstəricisi (Сетевое окружение).** Bu nişan kompüter şəbəkəsinə qoşulduğu halda EHM-in bu şəbəkədəki yerini, şəbəkə serverini, qoşulma vəziyyətini və s. göstərir.

4) **Mənim sənədlərim (Мои документы).** Bura istifadəçinin hazırladığı sənədlər, ən çox istifadə etdiyi digər sənəd və proqramlar da yerləşdirilə bilir.

5) **İnternet Explorer.** Bu nişan İnternet qlobal kompüter şəbəkəsinə keçməyə imkan verən tətbiqi proqrama girişi təmin edir.

Bundan əlavə istifadəçinin istəyi ilə iş stoluna onu maraqlandıran obyektlərin – qovluq, sənəd, proqram və s. nişanları da çıxarıla bilər. Lakin sistemin baş qovluğunu, onun işi üçün əhəmiyyətli olmayan obyektlərlə doldurmaq rasional addım deyil. Bunun əvəzinə iş stolunda bu obyektlərin yarlıklarından isifadə etmək daha əlverişlidir. Yarlık – hər hansı bir obyekti (fayl, qovluq, proqram və s.) ifadə edən nişandır, lakin hər bir obyekt üçün istənilən sayda yarlık yaratmaq mümkündür. Yarlık obyektin təkrarını almağa, onu bir yerdən başqa yerə köçürməyə imkan vermir, ondan yalnız proqramı işə salmaq, sənəd və ya qovluğu açmaq üçün istifadə etmək olur. Ümumiyyətlə, işarə etdiyi obyektin yarlıkı *.lnk* tipli kiçik (374 *bayt*) bir fayldır və burada həmin obyektin parametrləri və yerləşdiyi yer haqqında məlumatlar saxlanılır. Yarlıka müraciət etdikdə sistem bu yarlıkda saxlanılan məlumatlardan istifadə edərək obyekti tapıb açır və ya iş vəziyyətinə gətirir. Nişan, yarlık və s. üçün kontekst menyusunu çağırmaq olur. Bunun üçün siçan göstəricisini obyekt üzərinə yerləşdirib, siçanın sağ düyməsini sıxmaq, menyu bölmələrini obyekt üçün seçmək üçün isə sol düyməsini sıxmaq lazımdır. Bu menyuda *açmaq (открыть), göndərmək (отправить), kəsmək (вырезать), təkrarını almaq (копировать), yarlık yaratmaq (создать ярлык), ləğv etmək (удалить), adını dəyişdirmək (переименовать)* və *xassələr (свойства)* bölmələri var. Burada *açmaq*la obyekti açmaq, *göndərmək*lə onu diskə, yaddaşın digər hissəsinə və s. göndərmək olur, *kəsmək* obyekti iş stolundan kəsib götürərək, mübadilə buferində saxlayır, *təkrarını almaq* ilə obyektin təkrarını alıb mübadilə buferində saxlamaq olur, *yarlık yaratmaq* ilə obyekt üçün daha bir yarlık yaradılır və iş stolunda yerləşdirilir, *ləğv etmək* isə obyekti iş stolundan zibil qutusuna göndərir, *adını dəyişdirmək*lə obyektə başqa ad vermək olur, nəhayət, *xassələr* ilə obyekt haqqında tam informasiya – onun tipini, tutduğu yaddaş həcmini, diskdəki vəziyyətini öyrənmək olur.

Baş menyu *Mənim kompüterim* nişanı kimi Windows sisteminin bütün proqramlarına və tənzimləmə vasitələrinə keçməyə imkan verir. Baş menyu iş stoluna *«Пуск»* düyməsi ilə çağırılır. Əvvəlcə ekrana menyunun birinci (ali) səviyyəsi çağırılır.

Baş menyunun əsas bölmələrinin təyinatına baxaq:

1) ***Proqramlar (Программы).*** Bu bölmə kompüterdə işə salınması mümkün olan proqram və ya proqramlar qovluğunun siyahısını verir. Bu siyahıya daxil olan bəzi proqramları qeyd edək: Standart proqramlar qovluğu WordPad, Paint, Bloknot, kalkulyator kimi alət proqramları birləşdirir. MS DOS seansı MS DOS-un işini təmin edən proqram işə salır. Bələdçi fayl sistemi ilə işi təmin edən və onun bütün obyektlərinə keçidi təmin edən proqramdır. Buraya eləcə də MS Word, MS Excel, MS Power Point, MS Access və s. proqramlar da daxil edilir.

2) ***Sənədlər (Документы).*** Bu bölmə sistemdə işlənmiş axırıncı 15 sənəd fayllarının siyahısını ekrana çıxarır. Bu siyahıdakı ixtiyari sənədi təkrarən açmaq üçün onun adını siçanın sol düyməsi ilə sıxmaq kifayətdir.

3) ***Tənzimləmə (Настройка).*** Bu bölmə ayrı-ayrı qurğularda təmzimləmə işlərini aparmağa imkan verən sistem elementlərinin siyahısını çıxarır.

4) ***Axtarış (Поиск).*** Bu bölmə fayl və qovluqların axtarışını

təmin edir.

5) ***Arayış (Справка).*** Bu bölmə Windows-un arayışlar sistemi ilə işləməyə imkan yaradır.

6) ***Yerinə yetirmək (Выполнить).*** Bu bölmə ilə adına və ya ona gedən yola əsasən proqramı iş vəziyyətinə gətirmək və ya qovluq açmaq olur.

7) ***İşi sona çatdırmaq (Завершение работы).*** Bu bölmə ilə kompüterdə işi sona çatdırmaq və ya sistemi yenidən yükləmək olur.

Qeyd etdiyimiz kimi Windows-un tərkibinə çoxlu sayda müxtəlif təyinatlı proqramlar daxildir. Hər bir belə proqramın işə salınması zamanı o, öz pəncərəsində ekrana gətirilir. Bu proqramlar da öz növbəsində sənəd fayllarının açılması üçün pəncərələr yaradır. Ümumiyyətlə, sistemdə proqram, sənəd və qovluq, elcəcə də dialoq və arayış pəncərələri mövcuddur. Qovluq pəncərəsinə bu qovluğun tərkibi çıxarılır. Əvvəlcə proqram və ona yaxın olan qovluq pəncərəsinə baxaq. Hər bir belə pəncərənin, buradakı vəziyəti idarə edən əmr düymələri, alətlər lövhəsi, menyu və s. kimi elementləri var. Qeyd edək ki, Windows sisteminin hər bir pəncərəsində bəzi istisnalar olmaqla, demək olar ki, eyni elementlər ardıcıllığı olur.

Pəncərələrdə yerləşən əsas idarəetmə elementlərini qeyd edək:

1) *Əmr düyməsi.* Düymə düzbucaqlı formada olub, düymənin təyinatını göstərən simvol (məsələn, «×») və ya yazı (məsələn, *OK*, *İmtina* və s.) ilə müşayət olunur. Bunlardan birincilər pəncərə və dialoq lövhələrinin vəziyyətinin idarə edilməsi üçün, ikincilər isə dialoq lövhələrindəki dialoqun təşkil edilməsi üçün istifadə olunur. Düyməyə uyğun əməliyyatı yerinə yetirmək üçün onu siçanın sol düyməsi ilə sıxmaq lazımdır. Burada *OK* düyməsi əməliyyat və ya əmrin yerinə yetirilməsinin zəruri olduğunu təsdiq edir. *İmtina (Отмена)* düyməsi isə əvvəlki əməli ləğv edir.

2) *Pəncərə menyusu* – proqram və qovluq pəncərələrində bir sətirdə üfüqi yerləşmiş bölmələr ardıcıllığından ibarətdir. Menyu bölmələrinin sayı Windows-un müxtəlif pəncərələrində dəyişir, belə ki, qovluq pəncərəsində 4, MS Office proqramlarında 9 bölüm var. Bu bölmələrin ad və təyinatı da müxtəlif olur, lakin ***Fayl, Düzəliş, Görünüş*** və ***?*** bölmələri pəncərələrin çoxunda mövcuddur.

Menyu bölmələrini açmaq üçün onları siçanın sol düyməsi ilə sıxmaq lazımdır. Nəticədə ekrana onun alt bölmələrinin ardıcıllığı gətirilir. Analoji qaydada bu ardıcıllıqdan lazımi alt bölmə seçilir, bu isə hər hansı bir əmr və ya əməliyyatın yerinə yetirilməsinə səbəb olur.

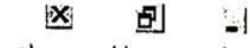
3) *Alətlər lövhəsi.* Bu lövhə düymələr ardıcıllığından ibarətdir. Hər bir düymə pəncərəyə yerləşdirilmiş sənədlə müəyyən əməliyyatın yerinə yetirilməsini təmin edir. Əməliyyatın növü düymədə təsvir olunmuş nişanla müəyən edilr. Bu düymənin təyinatı haqqında qısa arayış almaq üçün siçan göstəricisini düymə üzərinə yerləşdirmək kifayətdir.

4) *Dialoq lövhəsi.* Bu lövhə istifadəçi ilə dialoq qurmaq üçün nəzərdə tutulub, burada sual verilir, istifadəçi isə həmin suala verilə biləcək cavabı seçir. Bundan əlavə o, xəbərdarlıq edə bilər, müəyyən informasiya çatdıra bilər, köməkçi məsləhət verə bilər və s.

İndii isə pəncərə elementlərinin yerləşdirilməsinə baxaq:

1) *Sistem menyusunun düyməsi.* Bu düymə pəncərəyə çıxarılmış obyektin nişanı ilə ifadə olunur və pəncərənin birinci sətrinin sol hissəsində yerləşir. O, pəncərənin işini idarə edən sistem menyusunu çağırır.

2) *Pəncərənin başlığı.* Başlıq pəncərənin birinci sətrinin mərkəzində yerləşir və pəncərənin adından ibarətdir. Ad kimi adətən, bu pəncərədə gətirilən faylın (qovluğun) adından istifadə olunur. Başlığın fonunun rəngi açıq və ya tünd ola bilər. Tünd rəngli fon pəncərənin aktiv, yəni cari anda işə hazır olduğunu bildirir. Bu cür pəncərəyə cari pəncərə deyilir.



a) b) c)

Şəkil 3.3

3) *a) – c) formalı düymələr* (şək. 3.3) pəncərənin birinci sətrinin sağ tərəfində yerləşir. Burada *a)* düyməsi pəncərəni ekrandan çıxarır, bu zaman sistem pəncərənin obyekti ilə işi davam etdirir və pəncərənin düyməsi iş stolunun məsələlər lövhəsində saxlanılır. *b)* düyməsi uyğun olaraq pəncərəni bütün ekran boyunca açır və əvvəlki ölçülərinə qaytarır. *c)* düyməsi pəncərəni ekrandan çıxarır və pəncərənin obyekti ilə işi sona çatdırır.

4) *Pəncərənin menyusu* qeyd etdiyimiz kimi pəncərənin ikin-

ci sətrində yerləşir, bölmələrinin sayı və adları isə müxtəlif pəncərələrdə müxtəlif olur.

5) *Alətlər lövhəsi* pəncərə menyusundan aşağıda yerləşir. Bəzi pəncərələrdə bu cür bir neçə lövhə ola bilir. Həmin lövhədə uyğun pəncərədə iş üçün tələb olunan alətlər ardıcıllığı verilir.

6) *Cari vəziyyət sətiri* pəncərənin axırıncı sətirini tutur və alətlər lövhəsinin düymələrinin, menyu bölmələrinin və s. təyinatı haqqında məlumat verilməsi üçün istifadə olunur.

7) *İşçi sahə* – pəncərənin yerdə qalan hissəsini tutur və bura istifadəçini maraqlandıran obyekt çıxarılır.

8) *Pəncərənin çərçivəsi* – pəncərənin ölçülərinin və onun ekrandakı vəziyyətinin idarə edilməsində əhəmiyyətli elementdir.

9) *Çevirmə zolaqları* ölçüləri işçi sahənin ölçülərini aşan sənədləri nəzərdən keçirməyə imkan verir. Bu zolaqlar pəncərənin işçi sahəsinin sağ və aşağı tərəflərində yerləşir.

Pəncərələrlə aparıla bilən əməliyyatları qeyd edək:

1) Pəncərəni aktivləşdirmək, yəni cari anda onu iş vəziyyətinə gətirmək üçün pəncərənin ixtiyari yerini siçanın sol düyməsi ilə sıxmaq lazımdır, nəticədə başlığın fonunun rəngi tündləşir, pəncərə isə tam formada əks olunur. Bu əməliyyatdan ekranda bir neçə pəncərə olduqda və bir pəncərədə işi qurtarıb onu ekrandan çıxarmamaq şərtilə digər pəncərədə işə keçmək tələb olunduqda istifadə edilir.

2) Pəncərənin ölçülərini maksimum artırmaq (əvvəlki ölçülərini qaytarmaq) üçün *b)* düyməsini sıxmaq (*b)* düyməsini təkrar sıxmaq) (şək.3.3) lazımdır.

3) Pəncərənin ölçülərini dəyişdirmək üçün siçan göstəricisini pəncərənin çərçivəsi üzərinə yerləşdirib siçanın sol düyməsini sıxıb saxlamaq şərtilə siçanı pəncərə lazımi ölçüləri alana qədər hərəkət etdirmək lazımdır.

4) Pəncərənin ekrandakı vəziyyətini dəyişdirmək üçün siçan göstəricisini pəncərə başlığı üzərinə yerləşdirib siçanın sol düyməsini sıxıb saxlayaraq pəncərəni ekranın tələb olunan hissəsinə qədər çəkib aparmaq lazımdır.

5) Pəncərəni oradakı obyektlə işi sona çatdırmadan ekrandan kənarlaşdırmaq üçün siçanla *a)* düyməsini (şək. 3.3) sıxmaq lazımdır.

6) Pəncərəni oradakı obyektlə işi sona çatdıraraq ekrandan çıxarmaq üçün siçanla *c)* düyməsini (şək. 3.3) sıxmaq (və ya

Alt+F4 kombinasiyasından istifadə etmək) tələb olunur.

İndi isə dialoq pəncərələrinə baxaq. Bu pəncərələrdən istifadəçiyə hər hansı bir informasiyanı çatdırmaq, hər hansı bir sorğuya cavab almaq, hər hansı bir obyekti, məsələn, faylı seçmək üçün istifadə olunur. Birinci halda istifadəçi ona çatdırılan informasiyanı nəzərə alaraq *OK* düyməsini sıxmalı, ikinci halda sorğunu cavablandırmaq üçün uyğun düyməni sıxmalıdır. Üçüncü halda istifadəçi dialoq pəncərəsindəki daxil etmə sahəsində klaviaturadan obyektin adını (məsələn, fayl adı), hər hansı parametrin qiymətini və s. yığmalıdır. Qeyd edək ki, pəncərədə bu cür bir neçə daxil etmə sahələri ola bilər. Bu zaman bir sahədən digərinə keçmək üçün *Tab* düyməsindən istifadə etmək olar. Burada lazım olan əməliyyatların yerinə yetirilməsinin təsdiqi üçün *OK* və ya *Enter* düymələrini, imtina üçün isə *İmtina* və ya *Esc* düymələrini sıxmaq lazımdır.

Nəhayət, arayış pəncərələrinə baxaq. Əməliyyat sistemində bütün arayışlar müxtəlif ölçülü səhifələrə bölünüb. Hər bir səhifədə onun başlığında göstərilmiş bir mövzu tam şəkildə çatdırılır. Səhifələr öz növbəsində Windows sisteminin standart pəncərələrinə (menyunun olmamasını nəzərə almasaq) çıxarılır. Bu pəncərənin də ölçülərini dəyişdirmək, ekran boyunca sürüşdürmək olur. Pəncərədə *«Mündəricat» (Содержание), «Geriyə» (Назад)* və *«Parametrlər» (Параметры)* düymələri olur.

«Parametrlər» düyməsi ilə ekrana səhifədəki mətnlə müxtəlif əməliyyatlar aparmağa imkan verən menyu çıxarılır. Bəzi səhifələrdə adətən, hər hansı bir abzasın əvvəli və ya sonunda, ya da səhifənin sonunda düymələr olur. Bu düymələri siçanın sol düyməsi ilə sıxmaqla bu abzasların izahını verən və ya səhifəyə əlavə olan yeni səhifəyə keçmək olur. *«Geriyə»* düyməsi ilə isə əvvəlki səhifəyə qayıtmaq olur. *«Mündəricat»* düyməsi isə sistemləşdirilmiş şəkildə bütün arayışları çıxarılır. Çıxan siyahıdan arayış alınacaq bölməni axtarıb taparaq, açıb bu arayışa yiyələnmək olur.

Windows sistemində bəzi iş qaydalarına baxaq. Əvvəlcə iş stolunda qovluq və fayllarla iş qaydalarına baxaq:

1) Qovluğun (faylın) nişan və ya yarlıkını qeyd etmək üçün onu siçanın sol düyməsi ilə sıxmaq lazımdır.

2) Qovluğu (faylı) açmaq üçün onun nişanını siçanın sol düyməsi ilə ikiqat sıxmaq lazımdır. Nəticədə qovluğun (faylın)

pəncərəsi ekrana çıxarılır.

3) Hər hansı X qovluğunun (faylının) nişanını çıxarmaq üçün *Mənim kompüterim* qovluğunu açıb, burada X qovluğuna (faylına) rast gəlinənədək onun daxil olduğu diskin qovluğunu və digər qovluqları açmaq lazımdır.

4) X qovluğunu bağlamaq üçün *Backspase* düyməsini sıxmaq və ya pəncərənin alətlər lövhəsindəki başlığı əks çevrilmiş ox işarəsini sıxmaqla X qovluğunun yerləşdiyi daha yüksək səviyyəli qovluğa keçmək lazımdır.

5) X qovluğunun elementlərinin təsvir görünüşünü dəyişmək üçün X qovluğunu açıb, buradakı *Görünüş* menyu bölməsinə daxil olaraq, onun *Siyahı, Cədvəl* və s. alt bölmələrindən hansısa birini seçirik.

6) X qovluğundakı proqramı işə salmaq və ya sənəd faylını açmaq üçün X qovluğunu açdıqdan sonra obyektin nişanı və ya yarlıkını siçanın sol düyməsi ilə ikiqat sıxmaq lazımdır.

7) X qovluğunun (faylının) təkrarını digər Y qovluğuna yerləşdirmək üçün X obyektinin nişanını (yarlıkını deyil!) qeyd edib *Düzəlişlər* menyu bölməsinin *Təkrarını almaq* alt bölməsini seçmək, sonra isə Y qovluğunu açaraq *Düzəlişlər* menyu bölməsinin *Daxil etmək* alt bölməsini seçmək (yəni siçanın sol düyməsi ilə sıxmaq) lazımdır.

8) X qovluğunu (faylını, yarlıkını) Y qovluğuna göndərmək üçün 7-ci bənddə verdiyimiz əməliyyatlarda *Düzəlişlər* menyu bölməsində *Təkrarını almaq* əvəzinə *Kəsmək* alt bölməsini seçərək, onları təkrar etmək kifayətdir.

9) X qovluğunu (faylını, yarlıkını) ləğv etmək üçün X obyektinin nişanını qeyd edib, *Del* düyməsini və ya alətlər lövhəsindəki ləğv etmək düyməsini sıxmaq lazımdır.

10) X diskinin Y qovluğunda qovluq yaratmaq üçün *Mənim kompüterim* qovluğunda X diskini və Y qovluğunu açıb, *Fayl* menyu bölməsinin *Yaratmaq* alt bölməsində *Qovluq* sətrinə keçmək lazımdır. Sonra isə pəncərənin axırıncı sətrində yaranan daxiletmə sahəsində qovluğun adını daxil edib, *Enter* düyməsini sıxmaq lazımdır. Fayl yaratmaq üçün isə *Fayl* menyu bölməsinin *Yaratmaq* alt bölmməsində faylın yaradılacağı uyğun proqram adını seçib sıxmaq lazımdır. Nəticədə pəncərənin son sətrindəki sahədə faylın şərti adı və ad genişlənməsi verilir. Burada şərti adı dəyişdirib, ad genişlənməsini isə saxlayıb *Enter* düyməsini sıxmaq

lazımdır.

11) X qovluğunun (faylının) yarlıkını yaratmaq üçün əvvəlcə X qovluğunun (faylının) nişanını qeyd edib, sonra pəncərənin *Fayl* bölməsinin *Yarlık yaratmaq* alt bölməsinə keçmək lazımdır.

İndi isə *Bələdçi* proqramı ilə işə baxaq. Bu proqramda fayllarla ixtiyari əməliyyatlar yerinə yetirmək olar. Proqramı sistemin Baş menyusunun *Proqramlar* bölməsindən tapıb açmaq olur. Nəticədə ekrana Windows sisteminin həmin adlı standart pəncərəsi çıxarılır. Onun işçi sahəsində fayl sisteminin strukturunu görmək olar. Pəncərənin işçi sahəsi şaquli istiqamətdə iki hissəyə bölünüb. Sol tərəfdə qrafik şəkildə EHM-dəki qovluqlar ağacı təsvir olunub ki, bura *İş stolu, Mənim kompüterim, Disk qovluqları və s.* daxildir. Hər bir qovluğun öz nişanı var. Daxilində digər qovluqlar olan qovluqların nişanlarının sol tərəfində «+» nişanı qoyulur. Əgər bu işarəni siçanın sol düyməsi ilə sıxsaq, onun daxilindəki qovluqların siyahısı çıxarılır və «+» işarəsi «-» ilə əvəz olunur. Əgər indi «-» işarəsini sıxsaq, siyahı ləğv olunacaq və əvvəlki vəziyyət bərpa olunacaq. Qovluğun nişanını siçanın sol düyməsi ilə sıxdıqda pəncərənin sağ tərəfində qovluğun tərkibi göstəriləcəkdir. Bələdçi proqramının axırıncı sətrində cari vəziyyət sətri yerləşir. Burada cari qovluğun tutduğu yaddaş həcmi və cari diskdə qalan azad yaddaş həcmi haqqında məlumat verilir. Bələdçi proqramında aparıla bilən əsas əməliyyatları qeyd edək:

1) Qovluq və ya faylı cari etmək üçün onun nişanını siçanın sol düyməsi ilə sıxmaq lazımdır.

2) Qovluqlar (fayllar) qrupunu cari etmək üçün (məsələn, k-cıdan n-yə qədər) k-cı obyektin nişanını siçanın sol düyməsi ilə sıxıb, *Shift* düyməsini sıxıb saxlayaraq k-cı obyektin nişanını siçanın sol düyməsi ilə sıxmaq lazımdır.

3) Qovluğu açmaq üçün qovluq pəncərənin sol tərəfində olduqda onun nişanını siçanın sol düyməsi ilə bir dəfə, sağ tərəfində olduqda isə ikiqat sıxmaq lazımdır.

4) Qovluğu bağlamaq üçün yəni, onu yerləşdiyi daha ali səviyyəli qovluğa qaytarmaq üçün pəncərənin alətlər lövhəsindəki başlığı əks tərəfə çevrilmiş ox işarəsini siçanın sol düyməsi ilə ikiqat sıxmaq lazımdır. Qeyd edək ki, burada qovluq (fayl, yarlık) yaratmaq, təkrarını almaq, yerini dəyişdirmək və ləğv etmək yu-

xarıda təsvir etdiyimiz qaydalarla yerinə yetirilir.

5) Proqramı (exe və ya com tipli faylı) işə salmaq və ya sənəd faylını açmaq üçün obyekt nişanını qeyd edib, onu siçanın sol düyməsi ilə ikiqat sıxmaq lazımdır.

6) Fayl sistemində qovluğu (faylı) axtarmaq üçün pəncərənin *Servis* menyu bölməsinin *Tapmaq alt* bölməsində *Qovluq* və fayl sətrini seçib, ekrana verilən dialoq pəncərəsində «Fayl» sahəsinə faylın adını, «*Qovluq*» sahəsinə diskin adını (C:,D: və s.) yazıb, bu pəncərədəki «*Tapmaq*» düyməsini sıxmaq lazımdır.

3.5. MS Paint qrafik redaktoru

Windows əməliyyat sisteminin tərkibinə daxil olan MS Paint qrafik redaktoru çox da mürəkkəb olmayan təsvirlərin qurulması üçün nəzərdə tutulub. Proqram sistemin Baş menyusunun *Proqramlar* bölməsinin *Standart* alt bölməsinə keçməklə işə salınır. Bu zaman ekrana proqramın iş stolu adlanan pəncərə verilir (şək. 3.4).

Şəkil 3.4

Redaktorun iş stolunun birinci iki sətri başlıqdan, idarəetmə düymələrindən və menyudan ibarətdir. Pəncərənin yerdə qalan hissəsi aşağıdakı beş oblasta bölünüb:

1) Alətlər lövhəsi (adətən pəncərənin sol tərəfində yerləşir) hər birində bir cüt düymə olan 8 sırada yerləşdirilmiş 16 düymədən ibarətdir. Sadəlik üçün alətləri, məsələn 1.1 aləti, yəni birinci sətirdəki birinci alət kimi işarə edək. Burada 1.1 və 1.2 düymələri uyğun olaraq rəsmin ixtiyari sahəsini və düzbucaqlı sahəsini qeyd edir. 2.1 düyməsi ilə rəsmin seçdiyimiz hissəsini silə bilirik. 2.2 düyməsi rəsmin seçdiyimiz qapalı bir hissəsini verdiyimiz rənglə rəngləyə bilir. 3.1 düyməsi rəng seçimini təmin edir. Bu aləti seçdikdən sonra kursoru təsvirdəki ixtiyari rəng üzərinə qoyub siçanın sol düyməsini sıxmaqla həmin rəngi seçmiş oluruq və ondan rəsmin ixtiyari hissəsində istifadə edə bilərik. 3.2 düyməsi ilə təsvirin görünüş miqyasını dəyişdirmək olur. 4.1 aləti xətlər çəkilməsi üçün nəzərdə tutulub. Bunun üçün siçanın sol düyməsini sıxıb saxlayaraq kursoru lazım olan istiqamətdə siçan vasitəsilə hərəkət etdirmək lazımdır. 4.2 aləti isə 4.1 alətinə nisbətən daha qalın xətlərin çəkilməsini təmin edir. 5.1 düyməsi ilə kursorun cari anda durduğu nöqtəyə rəng çiləmək olur. 5.2 düyməsi ilə təsvirə mətn əlavə etmək olur. 6.1 düyməsi ilə rəsmə düz xətt parçası, 6.2 düyməsi ilə isə əyri xətt daxil etmək olur. 7.1 düyməsi düzbucaqlı (kvadrat), 7.2 düyməsi – çoxbucaqlı fiqur, 8.1 düyməsi ellips (çevrə), 8.2 düyməsi isə oval künclü düzbucaqlı çəkməyə imkan verir.

2) Əlavə xarkteristikalar sahəsi. Bu sahə alətlər lövhəsindən aşağıda verilir və məsələn alətlə təsvir olunan xəttin qalınlığını, obyektin tipini (çevrə, rəngləmiş çevrə, konturlu və ya konturusuz çevrə və s.) təyin etməyə imkan verir. Bunun üçün kursoru tələb olunan formalı obyektin təsviri üzərinə yerləşdirib siçanın sol düyməsini sıxmaq lazımdır.

3) Palitra pəncərənin aşağı hissəsində yerləşir və müxtəlif rəngli 28 düymədən ibarətdir. Palitra pəncərənin fonunun rəngini, təsvir olunan elementlərin rəngini təyin etməyə imkan verir.

4) Rəng pəncərəsi palitranın sol tərəfində yerləşir və iki düzbucaqlıdan ibarətdir. Aşağıdakı düzbucaqlının rəngi fonun rənginə, yuxarıdakının rəngi isə təsvir olunacaq elementlərin (xətt, nöqtə və s.) rənginə uyğun gəlir.

5) Cari vəziyyət sətri – pəncərənin son sətridir. Burada alətlər lövhəsinin düymələrinin təyinatı və siçan kursorunun koordinatları haqqında məlumatlar verilir.

6) Redaktorun iş stolunun yerdə qalan hissəsi isə bilavasitə təsvirin qurulması üçün nəzərdə tutulmuş işçi sahəsidir. Bu sahənin aşağı və sağ tərəflərində nəzərdən keçirmə zolaqları yerləşir. Təsvirin ölçüləri işçi sahənin ölçülərini üstələdikdə bu zolaqların köməyi ilə təsviri üfüqi və şaquli istiqamətlərdə hərəkət etdirərkən onu tam şəkildə nəzərdən keçirmək olur.

Redaktorun menyu sətri 6 bölmədən: *Fayl (Файл), Düzəlişlər (Правка), Görünüş (Вид), Şəkil (Рисунок), Palitra (Палитра)* və *Arayış (Справка)* bölmələrindən ibarətdir. Bu bölmələrə baxaq:

1) ***Fayl*** bölməsinin aşağıdakı alt bölmələri var:

a) *Yaratmaq (Создать)* əmri ilə yeni təsvir yaratmaq üçün rəsm sahəsi açmaq olur.

b) *Açmaq (Открыть)* əmri ilə bərk və ya elastik diskdə olan mövcud rəsm fayllarını proqram pəncərəsinə gətirmək olur. Bu zaman açılan dialoq pəncərəsində lazım olan faylı tapıb, qeyd edərək bu pəncərədəki *Açmaq* düyməsini sıxmaq lazımdır.

c) *Saxlamaq (Сохранить)* əmri yaradılmış təsviri və ya onun üzərində edilmiş dəyişiklikləri diskdə yadda saxlayır. Bu zaman açılan dialoq pəncərəsindəki *Faylın adı* yazı sahəsində təsvir faylının adını yığıb, buradakı *Saxlamaq* düyməsini sıxmaq lazımdır.

ç) *Necə saxlamaq (Сохранить как)* əmri ilə cari sənədi başqa qovluqda, diskdə başqa adla saxlamaq olur.

d) *İlkin baxış (Предварительный просмотр)* əmri cari səhifənin çapa göndərilməzdən əvvəl, onu tam şəkildə nəzərdən keçirməyə imkan verir.

e) *Səhifə maketi (Макет страницы)* əmri ilə açılan pəncərədə cari sənəddə səhifənin ölçülərini, formatını təyin etmək olur.

ə) *Çap (Печать)* əmri səhifəni çapa göndərir. Bu zaman açılan pəncərədə çap olunacaq səhifələrin nömrələrini, sayını və s. təyin etmək olur.

f) *Göndərmək (Отправить)* əmri ilə cari rəsmi elektron poçtla digər istifadəçiyə göndərmək olur.

g) *İş stolunu örtmək (Заполнить рабочий стол)* əmri ilə cari sənəddəki təsviri iş stolunun mərkəzində yerləşdirmək olar.

h) *Çıxış (Выход)* əmri cari proqram pəncərəsini bağlayır. Bu zaman əgər cari sənəd yadda saxlanılmayıbsa, onda onu ekrana verilən dialoq pəncərəsindəki *Hə* düyməsini sıxmaqla yadda saxlamaq olar.

2) ***Düzəlişlər*** menyu bölməsinin aşağıdakı alt bölmələri var:

a) *Ləğv etmək (Отмена)* əmri axırıncı yerinə yetirilmiş əməliyyatın nəticəsini ləğv edir.

b) *Təkrar (Повтор)* əmri ilə axırıncı yerinə yetirilmiş əməliyyatı təkrar yerinə yetirmək olur.

c) *Kəsmək (Вырезать)* əmri ilə cari təsvirdə qeyd olunmuş hissəni, mübadilə buferində saxlamaq olar və yeni kəsmə əməliyyatı aparılana qədər bu hissə cari təsvirin ixtiyari yerində bərpa oluna bilər və ya Windows-un ixtiyari proqramında açılan sənədə daxil edilə bilər.

ç) *Təkrarını almaq (Копировать)* əmri təsvirin qeyd olunmuş hissəsinin təkrarını alıb, mübadilə buferində saxlayır.

d) *Daxil etmək (Вставить)* əmri buferdə saxlanılmış obyekti cari təsvirdə ixtiyari yerə daxil edə bilir.

e) *Hər şeyi qeyd etmək (Выделить все)* əmri təsviri tamamilə qeyd edir.

ə) *Qeyd edilənin silinməsi (Очистить выделения)* əmri qeyd olunmuş hissəni təmizləyir.

f) *Təkrarını fayla köçürmək (Копировать в файл)* əmri ilə cari təsviri tam və ya onun bir hissəsini qeyd edərək başqa fayla köçürmək olur. Bunun üçün uyğun dialoq pəncərəsində köçürülmə yerinə yetiriləcək faylın adını verib *OK* düyməsini sıxmaq lazımdır.

g) *Fayldan daxil etmək (Вставить из файла)* əmri digər fayldakı təsviri cari sənəddəki rəsmə əlavə edir.

3) ***Görünüş*** menyu bölməsinin aşağıdakı alt bölmələri var:

a) *Alətlər lövhəsi (Набор инструментов)* əmri pəncərədəki alətlər lövhəsini ekrandan çıxarır və ya yerinə qaytarır.

b) *Palitra (Палитра)* əmri pəncərədəki palitranı ekrandan çıxarır və ya yerinə qaytarır.

c) *Cari vəziyyət sətri (Строка состояния)* əmri pəncərədəki cari vəziyyət sətrini ekrandan çıxarır və ya yerinə qaytarır.

ç) *Miqyas (Масштаб)* əmri ilə rəsmin görünüş miqyasını dəyişdirmək olur.

d) *Rəsmə baxış (Просмотреть рисунок)* əmri ilə pəncərənin bütün elementlərini ekrandan çıxararaq rəsmə tam baxmaq olur.

e) *Mətn atributları lövhəsi (Панель атрибутов текста)* əmri təsvirə mətn daxil edərkən şriftin, onun ölçüsünün, stilinin və s. seçilməsi üçün istifadə olunan lövhəni ekrana çıxarır və ya oradan ləğv edir.

4) ***Şəkil*** bölməsində aşağıdakı alt bölmələr var:

a) *Çevirmək və döndərmək (Отразить)* əmri ilə təsviri şaquli, üfüqi istiqamətlərdə çevirmək və 90, 180, 270 dərəcəli bucaqlar altında döndərmək olur.

b) *Uzatmaq və əymək (Растянуть и наклонить)* əmri təsviri üfüqi və şaquli istiqamətlərdə uzatmağa və əyməyə imkan verir.

c) *Rəngləri çevirmək (Обратить цвета)* əmri təsvirdə qeyd olunmuş hissənin rəngini ona əks olan rəngə (məsələn, ağ rəngi qara rəngə) dəyişdirir.

ç) *Atributlar (Атрибуты)* əmri təsvirin verildiyi iş sahəsinin ölçülərini, ölçü vahidlərini və s. parametrləri seçməyə imkan verir.

d) *Təsvirin silinməsi (Очистить)* əmri iş sahəsindəki təsviri tam şəkildə silir.

5) ***Palitra*** menyu bölməsinin aşağıdakı alt bölməsi var:

Palitranı dəyişdirmək (Изменить палитру) əmri proqram pəncərəsindəki palitrada verilmiş 28 rəngi proqramda olan daha 48 rəng çalarlarından hər hansı biri ilə dəyişdirməyə imkan verir.

6) ***Arayış*** menyu bölməsi ilə isə proqramda iş qaydaları haqqında müxtəlif arayışlar almaq olur.

İndi isə qrafik redaktorda bəzi iş qaydalarına baxaq:

1) İşə başlamazdan əvvəl ekranın təmizlənməsini *Şəkil* menyu bölməsinin *Təsvirin silinməsi* əmri ilə yerinə yetirmək olar.

2) Rənglərin təyin olunması. Bunun üçün kursoru palitranın lazım olan rənginin üzərinə qoyub, fonun rəngini vermək üçün sağ düyməsini, xətlərin, nöqtələrin rəngini vermək üçün isə sol düyməsini sıxmaq lazımdır.

3) İş sahəsini bütün ekran boyunca açmaq üçün *Görünüş* bölməsindəki *Rəsmə baxış* əmrini vermək, əvvəlki vəziyyətə qayıtmaq üçün isə pəncərənin ixtiyari nöqtəsini siçanın sol düyməsi ilə sıxmaq lazımdır.

4) Pəncərədəki təsvirin ölçülərini kiçiltmək üçün *Fayl* bölməsindəki *İlkin baxış* əmrindən, böyütmək üçün isə *Görünüş* bölməsinin *Miqyas* alt bölməsindəki *İri ölçü* və ya *Seçmək (ölçünü)*

əmirlərindən istifadə edilir.

5) Düz xətt parçasını qurmaq üçün əvvəlcə 6.1 düyməsini sıxıb, əlavə xarakteristikalar pəncərəsində xəttin qalınlığını seçməli, sonra kursoru parçanın başlanğıc nöqtəsinə qoyub, siçanın sol düyməsini sıxıb saxlayaraq kursoru parçanın son nöqtəsinə doğru sürüşdürmək lazımdır. Qeyd edək ki, ciddi üfüqi və ya şaquli xətt çəkərkən *Shift* düyməsini sıxıb saxlamaq lazımdır.

6) Təsviri silmək üçün 2.2 düyməsini sıxıb, əlavə xarakteristikalar pəncərəsində pozanın qalınlığını seçib siçanın sol düyməsini sıxıb saxlayaraq silinən zolağın əvvəlindən sonunadək kursoru sürüşdürmək lazımdır. Qeyd edək ki, pozan təsviri fonun rəngi ilə silir (rəngləyir).

7) Düzbugaqlını (oval künclü düzbucaqlını) qurmaq üçün 7.1 (8.2) düyməsini sıxıb, əlavə xarakteristikalar lövhəsində düzbucaqlının tipini (çərçivəli, çərçivəsiz, rənglənmiş) seçirik, sonra kursoru çəkəcəyimiz düzbucaqlının yuxarı sol küncünə yerləşdirib, siçanın sol düyməsini sıxıb saxlayaraq, kursoru düzbucaqlının diaqonalı boyunca yuxarıdan aşağıya doğru sürüşdürmək lazımdır.

8) Çoxbucaqlını qurmaq üçün 7.2 düyməsini sıxıb, əlavə xarakteristikalar pəncərəsindən çoxbucaqlının tipini (çərçivəli, çərçivəsiz, rənglənmiş) seçirik, çoxbucaqlının rənglənəcəyi rəngi seçirik, sonra isə yuxarıda düz xətt parçasını qurmaq üçün verdiyimiz qaydadan istifadə edərək, çoxbucaqlının tərəflərini çəkirik. Bu zaman növbəti düz xəttin başlanğıc nöqtəsi əvvəlkinin son nöqtəsi olacaqdır.

9) Çevrə və ya ellipsin qurulması üçün 8.1 düyməsini sıxıb, əlavə xarakteristikalar pəncərəsində fiqurun tipi (rənglənmiş, konturlu və ya kontursuz fiqur) seçilir, fiquru rəngləyəcəyimiz rəngi seçirik, sonra isə siçanın sol düyməsini sıxıb saxlayaraq, kursoru sürüşdürməklə ellipsi çəkirik. Çevrə qurarkən isə *Shift* düyməsini sıxıb saxlamaq lazımdır.

10) Qapalı konturla məhdudlaşdırılmış oblastın rənglənməsi üçün 2.2 düyməsini sıxmaq, rəngi seçmək və kursoru kontur daxilindəki ixtiyari nöqtəyə yerləşdirib, siçanın sol düyməsini sıxmaq lazımdır.

11) Əyri xəttin qurulması üçün 6.2 düyməsini sıxmaq və düz xətt parçasının qurulması üçün yuxarıda verdiyimiz əməliyyatlar

ardıcıllığını yerinə yetirmək lazımdır. Bundan sonra siçanın sol düyməsini sıxıb saxlayaraq, siçanı sürüşdürməklə, düz xətt parçasını da istənilən istiqamətlərdə əymək olur.

12) Karandaşın (fırçanın) köməyi ilə müxtəlif fiqurların qurulması üçün 4.1 (4.2) düyməsini sıxıb, əlavə xarakteristikalar pəncərəsində karandaş (fırça) üçün xəttin qalınlığını seçmək, sonra siçanın sol düyməsini sıxıb saxlayaraq, siçanı hərəkət etdirmək lazımdır.

13) 5.1 düyməsi ilə müxtəlif fiqurların qurulması üçün bu düyməni sıxıb, sonra 12-ci bənddəki əməliyyatları təkrar etmək lazımdır.

14) Ekrana mətn hissəsi daxil etmək üçün 5.2 düyməsini sıxıb, sonra kursoru mətnin veriləcəyi yerin əvvəlinə yerləşdirib, siçanın sol düyməsini sıxmaq lazımdır. Nəticədə ekrana mətnin çıxarılacağı oblastı işarə edən çərçivə verilir, bura klaviaturadan mətni yığmaq lazımdır.

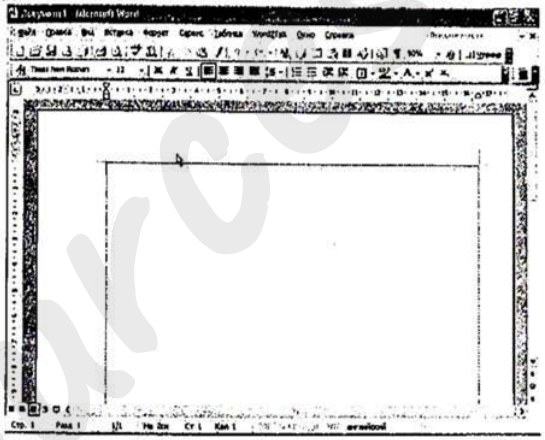
15) Lupa rejimində işləmək üçün 3.2 düyməsini sıxmaq lazımdır. Bu zaman ekrana düzbucaqlı çərçivə verilir, siçanın sol düyməsini sıxıb saxlamaqla çərçivəni böyüdülməsi tələb olunan təsvir hissəsinə sürüşdürmək lazımdır. Sonra isə siçanın sol düyməsini sıxmaq lazımdır. Nəticədə həmin hissədəki təsvir böyüdülmüş olur.

16) Təsvirin diskə, fayla yazılması və təsvirin fayldan ekrana çıxarılması əməliyyatları Windows sisteminin proqramları üçün yuxarıda verdiyimiz standart prosedurlara uyğundur.

3.6. MS Word mətn redaktoru

Microsoft Word 2000 mətn redaktoru mətnlərin, sənədlərin hazırlanması və emalı proqramıdır. MS Word yükləndikdə ekrana proqramın iş stolu adlanan pəncərə çıxarılır (şək. 3.5).

Burada birinci sətir başlıq – Windows sisteminin standart pəncərə başlıqları ilə eynidir. İkinci sətirdə proqramın menyu sətiri yerləşir. Menyu sətrindən aşağıda piktoqramlı düymələr ardıcıllığından ibarət alətlər lövhəsi yerləşir. Hər bir düyməyə hər hansı bir əmr uyğun gəlir, düymə üzərindəki piktoqram isə əmrin mahiyyətini ifadə edir. Düymələrin çoxu menyuda olan ən çox istifadə olunan əmrləri təkrarlayırlar. Adətən, menyu sətri altında iki növ alətlər lövhəsi – *Standart* və *Formatlaşdırma* lövhələri yerləşir. Bu lövhələri, ixtiyari düyməni siçanın sağ düyməsini sıxmaqla ekrana gələn kontekst menyusu ilə idarə etmək əlverişli olur.



Şəkil 3.5

Proqramın iş stolunun yuxarı və sol tərəfində yerləşən üfüqi və şaquli koordinat xətkeşlərinin köməyi ilə səhifənin kənarlarından buraxılan sahələri, abzas ölçülərini təyin etmək, sütunların enini dəyişdirmək və s. yerinə yetirmək olur. Proqram pəncərəsinin son sətri cari vəziyyət sətri müxtəlif məlumat və arayışların gətirilməsi üçün nəzərdə tutulub.

MS Word redaktoru sənədi müxtəlif rejimlərdə nəzərdən keçirməyə imkan verir:

1) *Adi*-əməliyyatların bir çoxunun yerinə yetirilməsi üçün ən əlverişli rejimdir.

2) *Web-səhifə* – sənədi Web-səhifə formasında ifadə edir;

3) *Səhifə ayırıcları* – sənədi çapa çıxarılacaq formada ifadə edir;

4) *Struktur* – sənədin strukturu ilə işi təmin edir, mətn və başlıqları göstərməyə və ya gizlətməyə imkan verir, alt sənədlərin yaradılmasını və onlarla işi təmin edir.

Rejimlər arasında keçid *Görünüş* menyu bölməsinin uyğun

alt bölmələrinin və ya üfüqi çevirmə zolağından solda yerləşən düymələrin köməyi ilə yerinə yetirilir. Pəncərələrin sağ və aşağı hissələrində yerləşən şaquli və üfüqi çevirmə zolaqları mətnin redaktorun pəncərəsində şaquli və üfüqi istiqamətlərdə hərəkət etdirilməsini təmin edirlər.

Redaktorun menyu bölmələrinə baxaq. Menyu sətrində 9 bölmə var: *Fayl (Файл), Düzəlişlər (Правка), Görünüş (Вид), Daxil etmək (Вставка), Format (Формат), Servis (Сервис), Cədvəl (Таблица), Pəncərə (Окно)* və *Arayış (Справка).*

1) ***Fayl*** menyu bölməsinin aşağıdakı alt bölmələri var:

a) *Yaratmaq (Создать)* əmri *Ümumi, Məktublar* və *Fakslar, Qeydlər, Hesabatlar, Nəşrlər, Digər sənədlər* və *Web-səhifələr* şablonları əsasında yeni sənəd yaradır.

b) *Açmaq (Открыть)* əmri mövcud sənədi yaddaşdan proqram pəncərəsinə çağırır.

c) *Bağlamaq (Закрыть)* əmri cari sənəd olan proqram pəncərəsini bağlayır. Bu zaman sənəddə dəyişikliklər olmuşdursa, onda bu dəyişikliklərin yadda saxlanılmasını təklif edən dialoq pəncərəsi açılır.

ç) *Saxlamaq (Сохранить)* əmri cari sənədi yaddaşda saxlayır. Bu zaman açılan dialoq pəncərəsində faylın adını verib *Saxlamaq* düyməsini sıxmaq lazımdır.

d) *Necə saxlamaq (Сохранить как)* əmri mövcud sənədi başqa adla, başqa yerdə saxlanılmasını təmin edir.

e) *Web-səhifə kimi saxlamaq (Сохранить как Web-страницу)* əmri ilə cari sənəd Web-səhifə kimi HTML formatında yaddaşda saxlanılır.

ə) *Səhifə parametri (Параметры страницы)* əmri ilə açılan eyni adlı dialoq pəncərəsində sənədin səhifələrinin lazım olan parametrlərini vermək olur.

f) *İlkin baxış (Предварительный просмотр)* əmri cari sənədin çap edilməzdən əvvəl səhifələrinə baxışı təmin edir.

g) *Web-səhifə kimi ilkin baxış (Предварительный просмотр Web-страницы)* əmri ilə cari sənədə Web-səhifə kimi ilkin baxışı təmin etmək olur.

ğ) *Çap (Печать)* əmri ilə cari sənədi çapa göndərmək olur. Bu zaman ekrana çıxarılan eyni adlı dialoq pəncərəsində çapın parametrlərini müəyyən etmək olur.

h) *Göndərmək (Отправить)* əmri cari sənədi elektron məktub və ya faks məlumat kimi göndərməyə imkan verir.

x) *Xassələr (Свойства)* əmri cari sənəd haqqında əlavə məlumatlar almağa imkan verir.

2) ***Düzəlişlər*** bölməsinin aşağıdakı alt bölmələri var:

a) *İmtina (Отменить)* əmri yerinə yetirilmiş axırıncı əməliyyatı ləğv edir.

b) *Təkrar etmək (Повторить)* əmri axırıncı yerinə yetirilmiş əməliyyatı təkrar edir.

c) *Kəsmək (Вырезать)* əmri cari sənəddə qeyd olunmuş mətn hissəsini kəsib mübadilə buferində saxlayır.

ç) *Təkrarını almaq (Копировать)* əmri cari sənəddə qeyd olunmuş mətn hissəsinin təkrarını alıb mübadilə buferində saxlayır.

d) *Daxil etmək (Вставить)* əmri bufer yaddaşdakı mətn hissəsini cari sənəddə ixtiyari yerə daxil edir.

e) *Xüsusi daxiletmə (Специальная вставка)* əmri bufer yaddaşda olan və Windows-un digər proqramlarında yaradılmış obyektləri cari sənədə xüsusi format əsasında daxil etməyə imkan verir. Bu zaman açılan dialoq pəncərəsindən həmin format seçilə bilər.

ə) *Hiperistinad kimi daxil etmək (Вставить как гиперссылку)* əmri digər sənəddən mübadilə buferinə köçürülmüş mətn hissəsini hiperistinad şəklində cari sənədə daxil etməyə imkan verir.

f) *Silmək (Очистить)* əmri cari sənəddə qeyd olunmuş hissəni və ya obyekti ləğv etməyə imkan verir.

g) *Hamısını qeyd etmək (Выделить все)* əmri cari sənəddəki mətni tamamilə qeyd edir.

ğ) *Tapmaq (Найти)* əmri ilə açılan dialoq pəncərəsində *Tapmaq* yazı sahəsində söz və ya söz birləşmələrinin nümunəsini verməklə onların cari sənəddəki mətndən tapılması təmin edilir.

h) *Əvəz etmək (Заменить)* əmri açılan dialoq pəncərəsində *Tapmaq* yazı sahəsində verilən söz və ya söz birləşmələrini cari sənəddəki mətndə axtarıb taparaq dialoq pəncərəsinin *Əvəz etmək* sahəsində verdiyimiz söz və ya söz birləşmələrinin nümunələri ilə əvəz edilməsini təmin edir.

x) *Keçmək (Перейти)* əmri ilə cari sənədin konkret səhi-

fəsinə, sətrinə, obyektinə və s. keçmək olur. Bunun üçün açılan dialoq pəncərəsində keçiləcək obyektin koordinatlarını vermək lazımdır.

ı) *Əlaqələr (Связи)* əmri seçilmiş obyekt və onun yaradıldığı proqram arasındakı əlaqəni göstərir. Əmrin yerinə yetirilməsi nəticəsində açılan pəncərədə bu əlaqəni qurmaq, yeniləşdirmək, obyekti başqası ilə əvəz etmək olur.

i) *Obyekt (Объект)* əmri cari sənədə daxil edilmiş və digər proqramlarda hazırlanmış hər hansı bir obyekti redaktə etmək və ya yenisi ilə əvəz etməyə imkan verir.

3) ***Görünüş*** menyu bölməsinin aşağıdakı alt bölmələri var:

a) *Adi (Обычный)* əmri cari sənədin adi formada görünüşünü təmin edir.

b) *Web-sənəd (Web-документ)* əmri cari sənədin Web-sənəd görünüşünü təmin edir.

c) *Səhifə ayırıcıları (Разметка страницы)* əmri cari sənədə çapa çıxarılacaq formaya uyğun görünüş verir.

ç) *Alətlər lövhəsi (Панели инструментов)* əmri ilə proqram pəncərəsinə lazım olan alətlərin verilməsini və lazım olmayanların çıxarılmasını yerinə yetirmək olar.

d) *Struktur (Структура)* əmri ilə struktur iş rejiminə keçmək olar.

e) *Xətkeş (Линейка)* əmri ilə ekrana şaquli və üfüqi xətkeşləri vermək və ya çıxarmaq olur.

ə) *Sənəd sxemi (Схема документа)* əmri ilə sənəd pəncərəsi üfüqi istiqamətdə iki hissəyə bölünür. Sol tərəfdə sənədin strukturu, sağda isə sənədin özü verilir.

f) *Kolontitullar (Колонтитулы)* əmri səhifədə yuxarı və aşağı kolontitulların yaradılması və redaktə edilməsini təmin edir.

g) *İstinadlar (Сноски)* əmri ilə cari sənəddəki nömrəli istinadlar redaktə edilə bilir.

ğ) *Qeydlər (Примечания)* əmri ilə cari sənəddə olan qeydlər redaktə edilir.

h) *Bütün ekran boyu (Во весь экран)* əmri cari sənədin bütün ekran boyu əks olunmasını təmin edir.

x) *Miqyas (Масштаб)* əmri cari sənədin görünüş miqyaslarını dəyişdirməyə imkan verir.

4) ***Daxil etmək*** menyusuna aşağıdakı alt bölmələr aiddir:

a) *Bölünmə (Разрыв)* əmri ilə səhifədəki mətn bölünür və kursorun durduğu mövqedən aşağıdakı mətn hissəsi yeni səhifəyə keçirilir.

b) *Səhifələrin nömrələnməsi (Номера страниц)* əmri cari sənədin səhifələrinin, bu zaman açılan dialoq pəncərəsində verəcəyimiz parametrlərə uyğun nömrələnməsini təmin edir.

c) *Tarix və zaman (Дата и время)* əmri cari sənədə, bu zaman açılan dialoq pəncərəsində seçdiyimiz formalı cari tarix və zaman daxil edir.

ç) *Avtomətn (Автотекст)* əmri cari sənədə tarix, zaman, ad, soyad, müxtəlif termin və mətn hissələri avtomatik daxil etməyə imkan verir.

d) *Sahə (Поле)* əmri cari sənədə düstur, simvol, mündəricat və paraqraf nömrələrini daxil edir və s. kimi əməliyyatlar yerinə yetirir.

e) *Simvol (Символ)* əmri cari sənədə klaviaturada nəzərdə tutulmamış simvolların daxil edilməsini təmin edir.

ə) *Qeyd (Примечания)* əmri ilə cari sənəddə mətnə kursorun durduğu mövqedən qeydlərin daxil edilməsi təmin olunur.

f) *İstinad (Сноска)* əmri ilə səhifənin, sənədin sonuna cari sənəddəki ixtiyari söz və ya söz birləşməsi üçün əlavə nömrələnmiş informasiyanın daxil edilməsi təmin edilir.

g) *Ad (Имя)* əmri mətndə rəsmlərlə, cədvəllərlə, düsturlarla və s. avtomatik nömrələnmiş adların verilməsini təmin edir.

ğ) *Kəsişən istinad (Перекрестная ссылка)* əmri mətndə müxtəlif paraqrafların başlığına, cədvəllərə və s. istinad edilməsini təmin edir.

h) *Mündəricat və göstəricilər (Оглавление и указатели)* əmri cari sənəddə mündəricatın, şəkillərin siyahısının və s. müəyyən formatla yaradılmasını təmin edir.

x) *Rəsm (Рисунок)* əmri ilə cari sənədə şəkillər, avtofiqurlar və s. daxil etmək olur. Bu əmrin menyusuna *Şəkillər*, *Avtofiqurlar*, *WordArt obyekti* və *Diaqram* bölmələri aiddir. Birinci iki bölmə proqramda olan hazır şəkilləri və digər fayllarda olan şəkilləri sənədə daxil etməyə imkan verir. Avtofiqurlar bölməsi ilə proqramda olan hazır fiqurları, WordArt ilə proqramdakı hazır yazı stillərinə gətirməklə ixtiyari yazını və *Diaqram* bölməsi ilə müxtəlif tipli diaqramları cari sənədə gətirmək olur.

ı) *Yazı* (*Надпись*) əmri mətn, şəkil, qrafik və s. üzərində digər mətn fraqmentinin, şəklin, cədvəlin və s. yerləşdirilməsi əməliyyatını həyata keçirir.

i) *Fayl* (*Файл*) əmri ilə cari sənədə kursorun durduğu mövqeyə digər fayldan sənəd daxil etmək olur. Həmin faylın adı və ya ona gedən yol bu zaman açılan dialoq pəncərəsində göstərilməlidir.

j) *Obyekt* (*Объект*) əmri ilə cari sənədə kursorun durduğu mövqedən Windows-un digər proqramlarında yaradılan obyektləri daxil etmək olur.

k) *İşlik* (*Закладка*) əmri cari sənəddə avtomatik olaraq müəyyən seçilmiş sözü, obyekti nişanlamağa imkan verir.

q) *Hiperistinad* (*Гиперссылка*) əmri cari sənəddən müxtəlif fayllara istinadı təmin edir.

5) ***Format*** menyu bölməsinə aşağıdakı bölmələr aiddir:

a) *Şrift* (*Шрифт*) əmri cari sənəddə istifadə ediləcək şrifti, onun ölçülərini, rəngini, şriftlərarası intervalı və s. parametrləri seçməyə imkan verir.

b) *Abzas* (*Абзац*) əmri ilə cari sənəddə abzasların verilmə qaydaları müəyyən olunur. Bu zaman açılan pəncərədə abzas üçün uyğun parametrləri seçmək olur.

c) *Siyahı* (*Список*) əmri cari sənəddə sətirlər qarşısında müxtəlif nişanlar, rəqəmlər yerləşdirilməsini təmin edir.

ç) *Sərhəd və rəngləmə* (*Границы и заливка*) əmri ilə cari sənəddə qeyd olunmuş mətn hissəsini və ya bütün səhifəni çərçivə daxilinə almaq olur. Burada çərçivə üçün rəng və naxışlar da daxil etmək olur.

d) *Sütunlar* (*Колонки*) əmri cari sənəddəki mətnin bir neçə sütunlara bölünməsini təmin edir. Bu zaman açılan pəncərədə sütunların tipini, sayını, onlar arasındakı məsafəni vermək olur.

e) *Tabulyasiya* (*Табуляция*) əmri cari sənəddə aparılan tabulyasiyanın mövqeyini, mövqelərarası intervalları təyin edir.

ə) *Bukvisa* (*Буквица*) əmri abzasın birinci simvolunun, bu zaman açılan pəncərədə parametrləri verilən xüsusi formada çıxarılmasını təmin edir.

f) *Mətnin istiqaməti* (*Направление текста*) əmri ilə cari sənəddə cədvəldəki, mətndəki sözlərin şaquli və ya üfüqi istiqamətlərdə verilməsini təmin etmək olur.

g) *Registr* (*Регистр*) əmri ilə cari sənəddə qeyd olunmuş mətn hissəsində cümlənin ilk hərfinin böyük olmasını, bütün hərflərin kiçik olmasını, böyük hərflərin kiçik hərflərlə və kiçik hərflərlərin böyuk hərflərlə əvəz olunmasını təmin etmək olur.

ğ) *Fon* (*Фон*) əmri ilə cari sənəddə mətnin verildiyi fonun rəngini seçib təyin etmək olur.

h) *Mövzu* (*Тема*) əmri ilə cari sənəd üçün hazır yazı mövzularını seçib tətbiq etmək olur.

x) *Avtoformat* (*Автоформат*) əmri cari sənədi seçilmiş şablona uyğun avtomatik formatlaşdırır.

ı) *Stil* (*Стиль*) əmri ilə cari sənəddə qeyd olunmuş mətn hissəsi üçün yazı stili seçib tətbiq etmək olur.

i) *Obyekt* (*Объект*) əmri ilə cari sənəddə daxil edilmiş obyektin formatını seçmək olur.

6) ***Servis*** menyu bölməsində aşağıdakı alt bölmələr var:

a) *Düzgün yazılış* (*Правописание*) əmri cari sənəddəki mətndə orfoqrafik və qrammatik səhvlərin təyin edilməsi və aradan qaldırılmasını təmin edir.

b) *Dil* (*Язык*) əmri cari sənəddəki qeyd olunmuş müxtəlif dillərdə verilən mətn hissələrinin səhvlərinin düzəldilməsi üçün dillərin seçimini, eləcə də bu cür mətn hissələrində naməlum sözlərin sinonim və ya mənaca yaxın sözlərlə əvəz edilməsini və sətrin sonunda sözü hecaya bölməklə yeni sətrə keçilməsini təmin edir.

c) *Statistika* (*Статистика*) əmri cari sənəddə istifadə olunmuş sözlərin, durğu işarələrinin, simvolların, eləcə də səhifələrin, abzasların, sətirlərin sayını müəyyən edir.

ç) *Xülasə* (*Автореферат*) əmri cari sənəddəki mətnin xülasəsini hazırlayır.

d) *Avtoəvəz* (*Автозамена*) əmri cari sənəddə mətnin daxil edilməsi zamanı buraxıla biləcək səhvləri qabaqcadan avtomatik olaraq düzəltməyə və mətnin daxil edilməsi zamanı bir sıra simvolların başqaları ilə avtomatik əvəz olunmasına imkan verir.

e) *Düzəlişlər* (*Исправления*) əmri cari sənəddə edilmiş düzəlişləri mətndə əks etdirir, onları qəbul etmək və ya ondan imtina etməyə imkan verir.

ə) *Birləşmə* (*Слияние*) əmri yaradılmış məktub mətninə müxtəlif ünvanları və ünvan sahibinin informasiyasını birləşdirməklə çox saylı məktubların avtomatik yaradılmasını təmin edir.

f) *Müdafiə qurmaq* (*Установить защиту*) əmri ilə cari sənədi, ona bir çox düzəlişlər edilməsindən müdafiə etmək olur.

g) *Müdafiədən imtina* (*Снять защиту*) əmri tətbiq olunmuş müdafiəni ləğv edir.

ğ) *Konvertlər və poçt nişanları* (*Конверты и наклейки*) əmri poçt konvertləri və nişanları hazırlayıb çap edilməsini təmin edir.

h) *Məktub ustası* (*Мастер писем*) əmri ilə avtomatik olaraq müxtəlif məktub mətnlərinin hazırlanması təmin edilir.

x) *Tənzimləmə* (*Настройка*) əmri ilə ekranda alətlər lövhəsinin verilməsi, bu lövhəyə yeni düymələrin əlavə edilməsi və ya düymələrin çıxarılması və s. əməliyyatları yerinə yetirmək olur.

ı) *Parametrlər* (*Параметры*) əmri ilə proqramda çap parametrlərinin dəyişdirilməsi, səhifənin müxtəlif ölçü vahidləri ilə ölçülməsi, avtomatik orfoqrafik yoxlama aparmaq və s. parametrlər təyin etmək olur.

7) ***Cədvəl*** bölməsinə aşağıdakı alt bölmələr aiddir:

a) *Cədvəlin çəkilməsi* (*Нарисовать таблицу*) əmri ilə ekrana verilən pəncərədəki alətlərdən istifadə etməklə cədvəlin damalarını, sətir və sütunlarını çəkmək olur.

b) *Əlavə etmək* (*Добавить*) əmri aşağıdakı bölmələri olan kontekst menyu açır:

1) *Cədvəl* əmri. Bu zaman açılan dialoq pəncərəsində göstərdiyimiz sayda sətir və sütunları olan cədvəl yaradılır.

2) *Soldan sütunlar* əmri ilə cari sütunun sol hissəsindən cədvələ sütun əlavə olunur.

3) *Sağdan sütunlar* əmri isə cari sütunun sağ tərəfindən cədvələ sütun əlavə edir.

4) *Yuxarıdan sətir* əmri cari sətirdən yuxarıda cədvələ yeni sətir əlavə edir.

5) *Aşağıdan sətir* əmri cari sətirdən aşağıda cədvələ yeni sətir əlavə edir.

6) *Damalar əmri* cari damaları sola və yuxarı sürüşdürməklə cədvələ yeni dama, sətir və sütun əlavə edir.

c) *Ləğv etmək* (*Удалить*) əmrinin kontekst menyusundan *Cədvəl*, *Sütunlar və Sətirlər* əmrləri cari və ya qeyd olunmuş cədvəlləri və cədvəl sətir və sütunlarını ləğv edir. *Damalar* əmri isə cari damaları sola və yuxarı sürüşdürməklə cari və ya

qeyd olunmuş dama, sətir və sütunları ləğv edir.

ç) *Qeyd etmək* (*Выделить*) əmri cari cədvəlin, sütunun, sətrin və damanın qeyd edilməsini təmin edir.

d) *Damaların* birləşdirilməsi (*Объеденить ячейки*) əmri cədvəldə qeyd edilmiş damaları birləşdirir.

e) *Damaların* bölünməsi (*Разбить ячейки*) əmri sədvəldə cari damanın bir neçə sətir və sütuna bölünməsini təmin edir.

ə) *Cədvəli bölmək* (*Разбить таблицу*) əmri cədvəli kursorun durduğu yerdən iki hissəyə bölür.

f) *Avtoformat* (*Автоформат*) əmri cədvələ proqramda olan hazır formatların tətbiq olunmasını təmin edir. Bu zaman açılan dialoq pəncərəsindən format nümunəsi seçilir.

g) *Avtoseçim* (*Автоподбор*) əmri ilə cədvəldəki informasiyanın eninə görə cari sütunun eni, pəncərənin eninə görə cədvəlin eni, sütunların hündürlüyü və s. nizamlanır.

ğ) *Başlıqlar* (*Заголовки*) əmri çox səhifəli cədvəllərdə birinci sətri başlıq kimi seçir və başlıq avtomatik olaraq cədvəlin bütün səhifələrində əks olunur.

h) *Mətni cədvələ keçirmək* (*Преобразовать в таблицу*) sənəddə qeyd olunmuş mətn hissəsini cədvəl formasına keçirir. Bu zaman açılan pəncərədə sətir və sütunların sayı göstərilməlidir.

x) *Cədvəli mətnə keçirmək* (*Преобразовать в текст*) əmri cari sənəddə qeyd olunmuş cədvəli mətn formasına keçirir.

ı) *Çeşidləmə* (*Сортировка*) əmri cədvəldəki informasiyanı sütun boyu əlifba sırasına görə, artma və azalmaya görə çeşidləməyə imkan verir. Bu zaman açılan dialoq pəncərəsində sütunları və çeşidləmə qaydasını müəyyən etmək tələb olunur.

i) *Düstur* (*Формула*) əmri ilə açılan dialoq pəncərəsində müxtəlif funksiya və düsturları daxil edərək onlar üzrə hesablamalar aparmaq olur.

j) *Cədvəl torunu əks etdirmək və ya gizlətmək* (*Отобразить или скрыть сетку*) əmri cədvəlin torunu gizlətmək və ya əks etdirməyə imkan verir.

k) *Cədvəlin xassələri* (*Свойства таблицы*) əmri cari sənəddə cədvəli sol, sağ və ya mərkəz istiqamətində nizamlayır, cədvəlin sətir və sütunlarının ölçülərini tənzimləyir və s.

8) ***Pəncərə*** menyu bölməsinin aşağıdakı alt bölmələri var:

a) *Yeni* (*Новое*) əmri cari proqram pəncərəsinin təkrarını

alıb, onu yeni pəncərədə əks etdirir.

b) *Hamısını nizamlamaq* (*Упорядочить все*) əmri vasitəsilə açılmış pəncərələri onların açılma ardıcıllığına görə düzmək olur.

c) *Bölmək* (*Разделить*) əmri ilə cari pəncərəni kursoru tələb olunan yerə qoyub, siçanın sol düyməsini ikiqat sıxmaqla iki yerə bölmək olur. Buradakı *Bölünmənin ləğv edilməsi* əmri ilə pəncərənin bölünməsi əməliyyatını ləğv etmək olur. Qeyd edək ki, *Pəncərə* menyu bölməsinin alt bölmələrinin sonunda açılmış pəncərələrin adlarının siyahısı gətirilir. Həmin siyahıdan lazım olan pəncərəni seçib, açmaq olur.

9) Arayış menyu bölməsi proqramda iş qaydaları haqqında arayışlar əldə etmək üçün nəzərdə tutulmuşdur.

İndi isə MS Word 2000 mətn redaktorundakı bəzi vacib iş qaydalarını qeyd edək. Yeni sənəd yaratmaq üçün *Fayl* menyu bölməsinin *Yaratmaq* əmrini seçmək lazımdır. Açılan dialoq pəncərəsində lazım olan şablonu seçib, *OK* düyməsini sıxmaq tələb olunur. MS Word-ün şablonları *dot* ad genişlənməsinə malikdir. Adi sənədlər isə *Yeni sənəd* şablonu əsasında qurulur və bunun üçün düyməsindən istifadə olunur. Mövcud sənədin açılması üçün *Fayl* bölməsinin *Açmaq* əmrini seçmək və ya düyməsini sıxmaq lazımdır. Nəticədə açılan dialoq pəncərəsinin *Qovluq* yazı sahəsində diski seçib, ondan aşağıdakı sahədə qovluğu siçanın sol düyməsinin ikiqat sıxılması ilə açaraq, lazım olan faylı seçmək lazımdır. MS Word sənədi *doc* ad genişlənməsinə malikdir. Dialoq pəncərəsinin yuxarı sətrində açılmış qovluğun tərkibini aşağıdakı 4 formada göstərməyə imkan verən düymələr yerləşir:

1) Fayl və qovluqların siyahısı kimi;

2) Fayl və qovluqlar haqqında informasiya verilən cədvəl kimi;

3) Sol sahədə seçilmiş faylın xassələri göstərilən forma;

4) Sağ sahədə seçilmiş faylın fraqmenti göstərilən forma.

Qeyd edək ki, siyahıda yalnız MS Word sənədləri olan fayllar veriləcəkdir. Digər tip və ya bütün tip faylları siyahıya çıxartmaq üçün pəncərədəki *Fayl tipləri* yazı sahəsindən istifadə edilir.

Sənədi yadda saxlamaq üçün *Fayl* bölməsinin *Saxlamaq* əmrinə keçmək və ya düyməsini sıxmaq lazımdır. Sənəd yaddaşda birinci dəfə saxlanıldıqda ekrana verilən dialoq pəncərəsinin *Qovluq* sahəsində sənədin saxlanılacağı diski, ondan aşağıdakı sahədə isə qovluğu seçmək lazımdır. Faylın tipi sahəsində faylın formatını seçib, *Faylın adı* sahəsində sənəd faylına ad verib, buradakı *Saxlamaq* düyməsini sıxmaqla sənəd faylını yaddaşda saxlamaq olur. Qeyd edək ki, faylı təkrar yadda saxlayarkən, sənəd avtomatik olaraq elə həmin adla saxlanılır. Sənədi başqa adla və ya başqa qovluqda saxlamaq tələb olunduqda *Fayl* menyu bölməsinin *Necə saxlamaq* əmrin seçib, yuxarıdakı əməliyyatları yerinə yetirmək lazımdır. Sənədin bağlanması üçün *Fayl* bölməsinin *Bağlamaq* əmrini seçmək və ya sənəd pəncərəsindəki düyməsini sıxmaq lazımdır.

İndi isə mətnlə iş qaydalarına baxaq. Sənəd pəncərəsində kursor mətnin daxil ediləcəyi mövqeyi bildirir. Səhifənin kənarına çatdıqda kursor avtomatik olaraq növbəti sətrin əvvəlinə keçir. Növbəti abzasın əvvəlinə keçmək üçün *Enter* düyməsini sıxmaq lazımdır. Sənəddə mətnin daxil edilməsinin iki rejimi var: *daxil etmək* və *əvəz etmək*.

Daxil etmək rejimində yeni simvolları daxil edərkən, buradakı mövcud mətn sağa doğru yerdəyişmə edir. *Əvəz etmək* rejimində isə köhnə mətn yenisi ilə əvəz olunur. Rejimlər arası keçid cari vəziyyət sətrindəki **ЗАМ** indikatorunun siçanın sol düyməsi ilə ikiqat sıxmaqla yerinə yetirilir. Mətnin müyyən bir hissəsi ilə iş aparmaqdan əvvəl onu qeyd etmək tələb olunur. Bunun üçün istifadə olunan qaydalara baxaq: kursoru ayrılacaq mətn hissəsinin əvvəlinə yerləşdirib, siçanın sol düyməsini sıxıb saxlayaraq kursoru fraqmentin sonuna qədər sürüşdürmək, bir sözü ayırmaq üçün siçanın sol düyməsi ilə onu ikiqat sıxmaq, bir cümləni ayırmaq üçün *Ctrl* düyməsini sıxıb saxlayaraq cümləni siçanın sol düyməsi ilə sıxmaq və s. Ayrılışı siçanın sol düyməsini sənəddə ixtiyari yerdə sıxmaqla ləğv etmək olar. Yeni ayrılış apardıqda əvvəlki ləğv olunur. Kursordan sağdakı simvolu *Delete*, soldakını isə *Backspace* düyməsi ilə silmək olur. Mətn hissəsini silmək üçün isə, əvvəlcə onu qeyd etməli, sonra *Delete* düyməsini sıxmaq lazımdır. Əgər mətn hissəsi ayırıb, sonra klaviaturadan yeni mətn yığsaq, yeni mətn qeyd etdiyimiz hissəsini əvəz edəcək. Abzası iki abzasa bölmək üçün kursoru birinci abzasın sonuna yerləşdirib, *Enter* düyməsini sıxmaq lazımdır. İki abzasdan birini qurmaq

üçün kursoru birinci abzasın sonuna yerləşdirib, *Delete* düyməsini sıxmaq və ya kursoru ikinci abzasın əvvəlinə qoyub *Backspace* düyməsini sıxmaq lazımdır. Axırıncı yerinə yetirilmiş redaktə etmə əmliyyatını ləğv etmək üçün *Düzəlişlər* bölməsinin *Ləğv etmək* əmrini və ya düyməsini sıxmaq, bu əməliyyatı bərpa etmək üçün isə həmin bölmənin *Təkrar etmək* əmrini seçmək və ya düyməsini sıxmaq lazımdır. Mətn hissəsinin təkrarını almaq üçün əvvəlcə bu hissəni ayırmaq, sonra *Düzəlişlər* bölməsinin *Təkrarını almaq* əmrini seçmək və ya düyməsini sıxmaq lazımdır. Bundan sonra kursoru mətndə, təkrarın veriləcəyi yerə yerləşdirib *Düzəlişlər* bölməsinin *Daxil etmək* əmrini və ya düyməsini sıxmaq lazımdır. Nəticədə qeyd olunmuş mətn hissəsinin təkrarı *Clipboard* adlanan mübadilə buferində yerləşdirilir və sonra sənəddə kursorun durduğu mövqedən daxil edilir. Buferdəki fraqmentdən ixtiyari sayda istifadə etmək olar, lakin buferdə yeni mətn hissəsi verildikdə əvvəlki hissə ləğv edilir. Mətn hissəsini bir yerdən başqa yerə köçürmək üçün isə bu hissəini qeyd edərək, *Düzəlişlər* bölməsinin *Kəsmək* əmrini seçmək və ya düyməsini sıxmaq, sonra kursoru mətn hissəsinin köçürüləcəyi yerə yerləşdirib, *Düzəlişlər* bölməsinin *Daxil etmək* əmrini seçmək və ya düyməsini sıxmaq lazımdır. Mətn hissəsini qeyd etdikdən sonra siçanın sol düyməsini sıxıb saxlayaraq, bu hissəsini mətnin lazım olan hissəsinə sürüşdürmək olar, bu zaman *Ctrl* düyməsini də sıxıb saxlasaq, həmin mətn hissəsinin təkrarı alınacaq.

MS Word 2000 redaktorunda 12 oyuqdan ibarət mübadilə buferi mövcuddur. Burada təkcə MS Word deyil, həm də digər proqramlardan məsələn, MS Excel-dən də fraqmentlərin təkrarını alıb, yerləşdirmək olur. *Mübadilə buferinin lövhəsini* ekrana çıxarmaq üçün *Görünüş* bölməsinin *Alətlər lövhəsi* alt bölməsinin *Mübadilə buferi* əmrinə keçmək lazımdır. Fraqmenti buferə yerləşdirmək üçün onu qeyd edib düyməsini sıxmaq, buferdən sənədə gətirmək üçün isə onun buferdəki nişanını sıxmaq lazımdır.

Klaviaturada olmayan simvolun mətnə daxil edilməsi üçün kursoru simvolun daxil ediləcəyi yerə qoyub, *Daxil etmək* bölməsində *Simvol* əmrini seçmək lazımdır. Bu zaman açılan dialoq pəncərəsində *Simvollar* bölməsini seçib, *Şrift* sahəsində şrift tipini təyin edib, cədvəldə lazımi simvolu seçdikdən sonra buradakı

Daxil etmək düyməsini sıxmaq lazımdır.

Mətn hissəsini sənəddə axtarıb tapmaq üçün *Düzəlişlər* bölməsinin *Tapmaq* əmrindən istifadə edilir. Açılan dialoq pəncərəsində *Tapmaq* yazı sahəsində axtarılan mətn hissəsini yığıb buradakı *Tapmaq* düyməsini sıxmaq lazımdır. Burada pəncərədə axtarışla əlaqədar əlavə məlumatlar verməklə axtarış yönümünü təyin etmək olar. Müəyyən mətn hissəsini digəri ilə əvəz etmək üçün isə *Düzəlişlər* bölməsinin *Əvəz etmək* əmrini seçmək lazımdır. Açılan pəncərədə *Tapmaq* sahəsində əvəz ediləcək mətn fraqmentini, *Əvəz etmək* sahəsində isə onun yerinə veriləcək fraqmenti yığmaq lazımdır. Bundan sonra əvvəlcə *Tapmaq*, sonra isə *Əvəz etmək* düymələrini sıxmaq lazımdır.

Mətni formatlaşdırmaq üçün onu qeyd etmək lazımdır, əks halda seçilən format yalnız yeni yığılacaq mətn hissəsinə şamil ediləcək. Mətndəki simvolların parametrlərinin dəyişdirilməsi üçün Format bölməsinin Şrift əmrindən istifadə edilir. Bu zaman ekrana verilən eyni adlı dialoq pəncərəsinin Şrift yazı sahəsində şriftin tipini, *Yazı forması* sahəsində isə şrift üçün *Adi, Kursiv, Qalın, Qalın kursiv* formalarından hər hansı birini seçmək olur. *Ölçü* sahəsi ilə şriftin punktlarla (*1 punkt*=0,375 *mm*) ölçüsünü, *Altından xətt çəkmək* sahəsi ilə simvolların altından çəkilən xəttin tipini, *Rəng sahəsi* ilə simvolların rəngini seçmək olur. *Effektlər* çərçivəsindəki üstündən xətt çəkmək və üstündən ikiqat xətt çəkmək əmrləri ilə mətndə simvollar üzərindən uyğun olaraq bir və ikiqat xətlər vermək olur. Burada *Yuxarı indeks* və *Aşağı indeks* əmrləri ilə simvolların ölçüləri kiçildilir və uyğun olaraq mətn yuxarı və aşağı indekslərdən yerləşdirilir. Kölgə əmri ilə simvolların yanında kölgə yaranır, *Kontur* əmri simvolların yalnız konturunu təsvir edir, *Qaldırılmış* əmri simvolları vərəq səthindən qaldırılmış formada, *Həkk edilmiş* əmri isə vərəq səthinə həkk olunmuş formada təsvir edirlər. *Kiçildilmiş böyük hərflər* əmri kiçik hərfləri kiçildilmiş böyük hərflərə, *Hamısı böyük hərflərlə* əmri isə kiçik hərfləri böyük hərflərə çevirir. Nəhayət, *Gizlədilmiş* əmri qeyd olunmuş simvolları çap zamanı kağız üzərinə çıxarmır. Simvolların interval və vəziyyətini dəyişdirmək üçün *Şrift* dialoq pəncərəsinin İnterval bölməsindən istifadə edilir.

Yığılmış mətndə simvolların registrini dəyişdirmək üçün *Format* bölməsinin *Registr* əmrini seçmək lazımdır. Açılan dialoq pəncərəsindəki Cümlələrdə olduğu kimi, *Hamısı kiçik hərflə,*

Hamısı böyük hərflə, Böyük hərflə başlamaq və Registri dəyişdirmək bölmələrindən birini seçib *OK* düyməsini sıxmaq lazımdır. Bu bölmələr qeyd olunmuş hissədə uyğun olaraq cümlədəki birinci sözü böyük hərflə başlayır, bütün hərfləri kiçik edir, bütün hərfləri böyuk edir, bütün sözləri böyük hərflə başlayır və böyük hərfləri kiçik, kiçik hərfləri isə böyük hərflərlə əvəz edir.

Abzasların parametrlərinin müəyyən edilməsi üçün *Format* bölməsinin *Abzas* əmrindən istifadə edilir. Bu zaman açılan dialoq pəncərəsində abzaslar üçün lazım olan parametrləri seçib, *OK* düyməsini sıxmaq lazımdır.

Tabulyasiyadan mətn və ədəd sütunlarının dəqiq düzümünü təmin etmək üçün istifadə edilir. Tabulyasiya mövqelərini təyin etmək üçün *Format* bölməminin *Tabulyasiya* əmrindən istifadə edilir. Bu zaman açılan dialoq pəncərəsində *Sol kənarına* görə, *Mərkəzə görə, Sağ kənarına* görə bölmələri mətni tabulyasiya mövqelərinə nəzərən uyğun olaraq sol kənarına, mərkəzinə və sağ kənarına görə düzümünü tənzimləyir. Buradakı *Bölgüyə görə* bölməsi ədədləri onluq vergülə görə, mətni isə sağ kənarına görə tənzimləyir.

MS Word redaktorunda qeydli, nömrələnmiş və çoxsəviyyəli nömrələnmiş siyahılar qurmaq olur. Siyahı elementi mətnin abzasları qəbul edilir. Siyahını yaratmaq üçün siyahı elementləri abzasları qeyd edib, *Format* bölməsinin *Siyahı* əmrini seçmək lazımdır. Bu zaman açılan dialoq pəncərəsində siyahı parametrlərini vermək olur.

Cari sənəddə səhifələrin parametrlərini təyin etmək üçün *Fayl* bölməsinin *Səhifənin parametrləri* əmri seçilir. Bu zaman açılan dialoq pəncərəsində *Sahələr bölməsi* ilə səhifənin kənarlarından buraxılacaq məsafələr santimetr ölçü vahidi ilə təyin edilir. Cilidləmə sahəsi ilə soldan və ya yuxarıdan cilidləmə üçün sahənin eni müəyyən olunur. Eləcə də burada *Kağızın ölçüləri* bölməsi ilə səhifənin ölçüləri, onun *kitab* və ya *albom* formalı olması təyin edilir.

Mətn avtomatik olaraq səhifələrə bölmək üçün kursoru bölgü aparılacaq yerə qoyub, *Daxil etmək* bölməsinin *Bölmək* əmrinə keçmək lazımdır. Açılan dialoq pəncərəsindəki *Yeni səhifə* sahəsinə keçib, *OK* düyməsini sıxmaq lazımdır. Əgər sənəddə səhifələr müxtəlif parametrli olmalıdırsa, onda onu bir neçə bölməyə ayırmaq lazımdır. Sənədə yeni bölmə daxil etmək üçün pən-

cərədəki *Növbəti səhifədən, Cari səhifədən, Cüt səhifədən* və ya *Tək səhifədən* sahələrinin birindən istifadə edilir. Sənəddə səhifələri nömrələmək üçün *Daxil etmək* bölməsinin *Səhifə nömrələri* əmrini seçmək lazımdır. Açılan pəncərədə *Vəziyyət* sahəsində nömrənin səhifədəki mümkün qoyulma vəziyyətlərindən (yuxarı və ya aşağı) birini seçmək tələb olunur. Burada eləcə də nömrəni səhifənin solunda, mərkəzində, sağında və s. qaydada yerləşdirmək olur. Pəncərədəki Birinci səhifədəki nömrə bölməsi ilə birinci səhifədəki nömrəni qoymamaq da olar.

Kolontitul – sənədin hər bir səhifəsində yuxarı və ya aşağı hissələrdə çap olunan mətn və ya şəkildir. Kolontitul yaratmaq üçün Görünüş menyu bölməsinin *Kolontitul* əmrini seçmək lazımdır. Bu zaman ekran səhifələrinin nişanlanması iş rejiminə keçir və ekrana Kolontitulların alətlər lövhəsi çıxarılır. Lövhədən istifadə edərkən, kolontitulların parametrlərini təyin etmək olur.

Sənədi çapa göndərmək üçün *Fayl* bölməsinin *Çap* əmrindən istsfadə edilir. Bu zaman açılan dialoq pəncərəsində Ad yazı sahəsində printerin tipini seçmək, Səhifələr çərçivəsində isə çapa verilən səhifələrin diapazonunu təyin etmək olur. Bu çərçivədə *Hamısı, Cari, Qeyd olunmuş fraqment* və *Nömrələr* bölmələrindən birini seçmək olar. *Nüsxələr* yazı sahəsində çapa verilən səhifələrdən neçə nüsxə çap olunacağı müəyyən olunur. Çapa çıxarmaq siyahısında *Diapazonun bütün səhifələrini, Tək səhifələrini* və *Cüt səhifələrini* qiymətlərindən hər hansı birini seçmək olur.

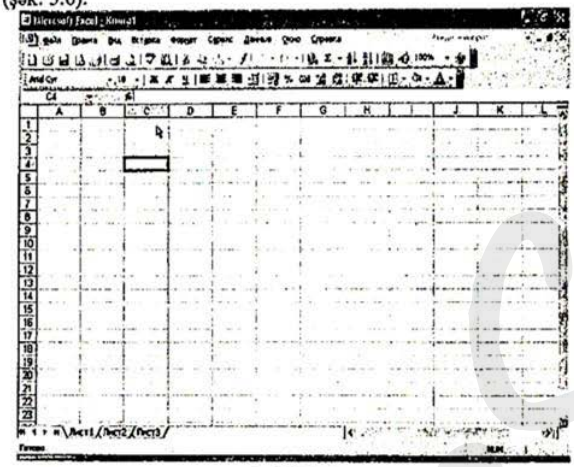
MS Word digər proqramlarda yaradılmış, eləcə də proqramın öz şəkil çəkmək bölməsinin köməyi ilə yaradılan qrafik obyektləri sənədə daxil etməyə imkan verir. *Şəkil çəkmək* lövhəsini *Daxil etmək* bölməsinin *Alətlər lövhəsi* alt bölməsindən çağırmaq olur. Bu lövhənin köməyi ilə xətt, ox, ellips, düzbucaqlı, çevrə və s. qurmaq olur. Daxil edilmiş qrafik obyekti rəngləmək, formasını, rəngini dəyişmək, ona digər obyektləri əlavə etmək və s. mümkündür. Digər proqramlarda yaradılan qrafik obyekti sənədə daxil etmək üçün *Daxil etmək* bölməsinin rəsm alt bölməsindəki *Fayldan* əmrini seçmək lazımdır. Açılan dialoq pəncərəsində *Qovluq* yazı sahəsində diski, bu sahədən aşağıdakı sahədə isə rəsm olan faylın yerləşdiyi qovluğu seçmək lazımdır. Proqramda olan hazır şəkilləri sənədə daxil etmək üçün isə *Daxil etmək* bölməsinin *Rəsm* alt bölməsindəki *Şəkillər* əmrini seçmək lazımdır.

Sənədə cədvəl daxil etmək üçün *Cədvəl* menyu bölməsinin

Əlavə etmək alt bölməsinin *Cədvəl* əmrini seçmək lazımdır. Açılan dialoq pəncərəsində cədvəlin sətri və sütunlarının sayını verib, *OK* düyməsini sıxmaq lazımdır. Cədvəllə iş üçün nəzərdə tutulmuş bütün əmrlər menyusunun *Cədvəl* bölməsində yerləşir.

3.7 MS Excel cədvəl prosessoru

Microsoft Excel 2000 – elektron cədvəllərinin yaradılması və emalı üçün nəzərdə tutulmuş proqramdır. Proqram EHM-ə yükləndikdə ekrana proqramın iş stolu adlanan pəncərə verilir (şək. 3.6).



Şəkil 3.6

Pəncərənin birinci sətri başlıqdan, ikinci sətri isə menyu sətrindən ibarətdir. Menyu sətrindən aşağıda piktoqram formasında verilmiş düymələr ardıcıllığından ibarət alətlər lövhəsi yerləşir. Bu düymələr proqramın menyusunda olan və ən çox istifadə edilən əmrləri təkrarlayır və onlardan istifadəni sadələşdirir. Alətlər menyu sətrində aşağıdakı iki növ alətlər lövhəsi – *Standart* və *Formatlaşdırma* lövhələri yerləşir. Bu lövhələrdən aşağıda adətən düsturlar sətri, pəncərənin aşağı hissəsində isə *Cari vəziyyət sətri* yerləşir. Proqram cədvəllərlə aşağıdakı iki rejimdə: *Adi* (əməliyyatların çoxunun yerinə yetirilməsi üçün əlverişli olan rejim) və *Səhifələrin nişanlanması* (çapdan əvvəl cədvəlin son formatlaşdırılması üçün əlverişli rejim) rejimlərində işi təmin edir.

Əvvəlcə proqramın menyu bölmələrinə baxaq. Menyu sətrində 9 bölmə var: *Fayl (Файл), Düzəlişlər (Правка), Görünüş (Вид), Daxil etmək (Вставка), Format (Формат), Servis (Сервис), Verilənlər (Данные), Pəncərə (Окно)* və *Arayış (Справка).*

1) *Fayl* menyu bölməsinə aşağıdakı alt bölmələr aiddir:

a) *Yaratmaq* (*Создать*) əmri *Ümumi* və *Qərar* bölmələri olan dialoq pəncərəsi açır. *Ümumi* bölmədə kitab və istifadəçinin yaratdığı şablonlar, *Qərar* bölməsində isə *Sifarişlər*, *Avans hesabatları* və s. şablonlar yerləşir. Bu şablonlardan ixtiyari birini seçməklə yeni kitabı (sənədi) onun əsasında yaratmaq olur.

b) *Açmaq* (*Открыть*) əmri mövcud sənədi proqram pəncərəsinə çağırır.

c) *Bağlamaq* (*Закрыть*) əmri cari sənəd pəncərəsini bağlayır.

ç) *Saxlamaq* (*Сохранить*) əmri sənədi yaddaşda saxlayır.

d) *Necə saxlamaq* (*Сохранить как*) əmri cari sənədi başqa adla, başqa yerdə, digər tip sənəd kimi yaddaşda saxlamağa imkan verir.

e) Web-*səhifə* kimi saxlamaq (*Сохранить как Web-страницу*) əmri sənədi yaddaşda Web-səhifə kimi saxlamağa imkan verir.

ə) *İşçi oblastı yadda saxlamaq* (*Сохранить рабочую область*) əmri işçi oblastı yadda saxlayır. Bu zaman açılan dialoq pəncərəsində əvvəlcə açılmış kitab səhifələrini yadda saxlayıb, sonra işçi oblastı yadda saxlamaq olur.

f) Web-*səhifəyə ilkin baxış* (*Предварительный просмотр Web-страницы*) əmri Web-səhifə kimi yadda saxlanılmış sənədin ilkin baxışını təmin edir.

g) *Səhifələrin parametrləri* (*Параметры страницы*) əmri səhifənin ölçülərini və s. parametrləri müyyən etməyə imkan verir.

ğ) *Çap sahəsi* (*Область печати*) əmri Müəyyən etmək və Ləğv etmək bölmələri olan menyu açır. Onlardan birincisi cədvəl-

də qeyd olunmuş damaları çap olunacaq fraqment kimi təyin edir, ikincisi isə ayrılmış fraqmentin çap sahəsi kimi təyin olunmasını ləğv edir.

h) *Çap* (*Печать*) əmri cari sənədi çapa göndərir. Bu zaman açılan dialoq pəncərəsində çap parametrləri verilir.

x) *Xassə* (*Свойства*) əmri cari sənəd haqqında məlumatlar almağa və əlavə məlumatlar daxil etməyə imkan verir.

ı) *Göndərmək* (*Отправить*) əmri cari sənədi elektron poçt və ya faksla başqa istifadəçiyə göndərə bilir.

i) *Çıxış* (*Выход*) əmri cari proqram pəncərsini bağlayıb, proqramdan çıxışı təmin edir.

2) *Düzəlişlər* menyu bölməsində aşağıdakı alt bölmələr var:

a) *Ləğv etmək* (*Отменить*) əmri axırıncı yerinə yetirilmiş əməliyyatı ləğv edir.

b) *Təkrar etmək* (*Повторить*) əmri axırıncı yerinə yetirilmiş əməliyyatı təkrar edir.

c) *Kəsmək* (*Вырезать*) əmri qeyd olunmuş fraqmenti kəsib, mübadilə buferinə yerləşdirir.

ç) *Təkrarını almaq* (*Копировать*) əmri qeyd olunmuş fraqmentin təkrarını alıb, mübadilə buferində saxlayır.

d) *Daxil etmək* (*Вставка*) əmri mübadilə buferindəki fraqmenti cari damaya kursorun durduğu mövqedən daxil edir.

e) *Xüsusi daxil etmək* (*Специальная вставка*) əmri mübadilə buferindəki informasiyanı tamamilə və ya digər formalarda cari damaya daxil etməyə imkan verir.

ə) *Hiperistinad kimi daxil etmək* (*Вставить как гиперссылку*) əmri digər proqramlarda yaradılmış və mübadilə buferində saxlanmış informasiyanı hiperistinad şəklində cari damaya daxil etməyə imkan verir.

f) *Tamamlama* (*Заполнить*) əmri qeyd olunmuş damalara informasiyanın daxil edilməsini avtomatlaşdırır, təkrarlanan və müəyyən addımla artan ədədlər daxil edir.

g) *Silmək* (*Очистить*) əmri ilə cari damadakı informasiyanı tam şəkildə, yalnız formatı, yalnız informasiyanı və ya yalnız qeydi silmək olur.

ğ) *Ləğv etmək* (*Удалить*) cari sənəddə qeyd olunmuş damaları və ya cari sətri, sütunu, damanı ləğv edir.

h) *Vərəqi ləğv etmək* (*Удалить лист*) əmri cari vərəqi

ləğv edir.

x) *Vərəqin təkrarının alınması və ya yerinin dəyişdirilməsi* (*Переместить/копировать лист*) əmri ilə cari vərəqin təkrarını almaq və ya kitabda vərəqlərin yerləşmə ardıcıllığını dəyişdirmək olur.

ı) *Tapmaq* (*Найти*) əmri cari sənəddə verilmiş simvol, söz və ya söz birləşmələrini nümunə əsasında axtarıb tapmağa imkan verir.

i) *Əvəz etmək* (*Заменить*) əmri cari sənəddə simvol, söz və ya söz birləşmələrini nümunə əsasında taparaq, verilmiş simvol, söz və ya söz birləşmələri ilə əvəz etməyə imkan verir.

j) *Keçid* (*Перейти*) əmri ünvanı göstərilən damaya avtomatik keçidi təmin edir.

k) *Əlaqələr* (*Связи*) əmri cari sənəddə olan Windows-un digər proqramlarında yaradılmış obyektlərin mənbələri ilə əlaqəsini göstərir.

q) *Obyekt* (*Объект*) əmri cari sənəddə olan digər proqramlarda yaradılmış digər obyektlər üzərində iş aparmağa imkan verir.

3) *Görünüş* menyu bölməsinə aşağıdakı alt bölmələr daxildir:

a) *Adi* (*Обычный*) əmri cari sənəddə adi iş rejimini təyin edir.

b) *Səhifənin nişanlanması* (*Разметка страницы*) əmri isə eyni adlı iş rejimini təyin edir.

c) *Alətlər lövhəsi* (*Панел инструментов*) əmri ilə iş stoluna alətlərin daxil edilməsini və ya çıxarılmasını təmin etmək olar.

ç) *Cari vəziyyət sətri* və *Düstur sətri* (*Строка состояния, Строка формул*) əmrləri ilə proqram pəncərəsinə uyğun olaraq cari vəziyyət və düstur sətrlərini daxil etmək və ya oradan çıxarmaq mümkündür.

d) *Kolontitullar* (*Колонтитулы*) əmri cari vərəqə kolontitulların daxil edilməsini və onların redaktə edilməsini təmin edir.

e) *Qeydlər* (*Примечания*) əmri cari vərəqədəki bütün qeydləri ekrana çıxarır.

ə) *Təqdim olunma* (*Представления*) əmri cari vərəqin bir

neçə ifadə formasını, çap parametrlərini yadda saxlayıb, ehtiyac olduqda istifadə etməyə imkan verir.

f) *Bütün ekran boyu* (*Во весь экран*) əmri ekranda yalnız menyu sətrini və sənəd pəncərəsindəki informasiyanı əks etdirməyə imkan verir.

g) *Miqyas* (*Масштаб*) əmri cari sənəddə pəncərənin görünüşünü müxtəlif miqyaslarda ifadə edə bilir.

4) *Daxil etmək* menyu bölməsinə aşağıdakı alt bölmələr aiddir:

a) *Dama* (*Ячейка*) əmri ilə açılan dialoq pəncərəsindəki *Sağa sürüşdürməklə* damalar, *Aşağı sürüşdürməklə* damalar, *Sətir*, *Sütun* bölmələri vasitəsilə uyğun olaraq cari damadan sağda, aşağıda yeni dama, sətir və sütünlar əlavə etmək olur.

b) *Sətirlər* (*Строки*) əmri cari damadan əvvəl yeni sətir əlavə edir.

c) *Sütun* (*Столбец*) əmri cari damadan əvvəl yeni sütun əlavə edir.

ç) *Vərəq* (*Лист*) əmri cari vərəqdən əvvəl yeni vərəq əlavə edir.

d) *Diaqram* (*Диаграмма*) əmri verilən qiymətlər əsasında diaqram, qrafik, histoqram qurulmasını təmin edir.

e) *Səhifənin bölünməsi* (*Разрыв страницы*) əmri cari səhifəni bölərək yeni səhifəyə keçidi təmin edir.

ə) *Funksiya* (*Функция*) əmri ilə açılan pəncərədə lazım olan funksiyanı seçib, funksiya arqumentini verib, *OK* düyməsini sıxmaqla funksiyanı hesablayıb, nəticəni cari damada yerləşdirmək olur.

f) *Ad* (*Имя*) əmri cari damaya və ya qeyd olunmuş damalara müəyyən adlar daxil etməyə imkan verir.

g) *Qeyd* (*Примечание*) əmri verildikdə açılan yazı sahələrinə damalardakı verilənlər haqqında izahedici qeydlər daxil etmək olur.

ğ) *Rəsm* (*Рисунок*) əmri cari sənəddə proqramda və ya fayllardan şəkillər, avtofiqurlar, WordArt mətnləri və s. daxil etməyə imkan verir.

h) *Xəritə* (*Карта*) əmri cari sənədə proqramda olan xəritələri daxil etməyə imkan verir.

x) *Obyekt* (*Объект*) əmri Windows-un digər proqramlarında hazırlanmış obyektləri cari sənədə daxil etməyə imkan verir.

i) *Hiperistinad* (*Гиперссылка*) əmri ilə açılan dialoq pəncərəsində istinad olunacaq məlumatın ünvanını və ya fayla gedən yolu göstərib *OK* düyməsini sıxmaqla həmin istinadı cari damaya yerləşdirmək olur.

5) *Format* menyu bölməsinə aşağıdakı alt bölmələr aiddir:

a) *Dama* (*Ячейка*) əmri ilə açılan dialoq pəncərəsindəki *Ədəd*, *Nizamlama*, *Şrift*, *Sərhəd*, *Görünüş* və *Müdafiə* bölmələri ilə uyğun olaraq cari damada və ya qeyd olunmuş damalarda ədədlərin verilmə formasını (ədəd, pul vahidi, tarix və zaman, kəsr və s.) təyin etməklə informasiyanın yazılış istiqamətini müəyyən etmək, qeyd olunmuş damaları birləşdirmək və s., şrift seçmək, cari damanı seçilmiş formalı çərçivəyə almaq, cari dama və ya qeyd olunmuş damaları seçdiyimiz rənglə rəngləmək və nəhayət, cari damadakı informasiyanı düzəlişlərdən müdafiə etmək olur.

b) *Sətir* (*Строка*) əmri ilə cari sətrin hündürlüyünü təyin etmək, bu sətrin hündürlüyünü mətnin şriftinə uyğunlaşdırmaq və cari sətri gizlətmək və ya ekrana çıxarmaq olur.

c) *Sütun* (*Столбец*) əmri ilə cari sütunun enini təyin etmək, onu gizlətmək və ya əks etdirmək olur.

ç) *Vərəq* (*Лист*) əmri cari vərəqin adını, fonunun rəngini dəyişdirməyə, onu gizlətməyə və ya əks etdirməyə imkan verir.

d) *Avtoformat* (*Автоформат*) əmri ilə cari sənəddə qeyd olunmuş damaları, bu əmrlə açılan dialoq pəncərəsindəki formatlardan hər hansı birinə uyğunlaşdırmaq olur.

e) *Şərti formatlaşdırma* (*Условное форматирование*) əmrinin açdığı dialoq pəncərəsində tələb olunan şərtləri daxil etməklə yeni format təyin etmək olur.

ə) *Stil* (*Стиль*) əmri ilə cari sənəddəki yazı stilini dəyişdirmək olur.

6) *Servis* menyu bölməsinə aşağıdakı alt bölmələr aiddir:

a) *Orfoqrafiya* (*Орфография*) əmri cari sənəddə olan mətnin orfoqrafik və qrammatik yazılışını yoxlayır.

b) *Avtoəvəz* (*Автозамена*) əmri cari sənəddə mətnin daxil edilməsi zamanı mətndə avtomatik düzəlişlərin aparılmasını və qeyd olunan simvolların başqaları ilə avtomatik əvəz olunmasını təmin edir.

c) *Düzəlişlər* (Исправления) əmri cari sənəddə aparılmış düzəlişlərə nəzarət etməyə, onların qəbul olunması və ya onlardan imtina edilməsinə imkan yaradır.

d) *Müdafiə* (*Защита*) əmri cari vərəqi, kitabı edilə biləcək düzəlişlərdən müdafiə edir.

e) *Parametrin seçilməsi* (*Подбор параметра*) əmri ilə açılan pəncərədə nəticənin veriləcəyi damanın ünvanını, onun qiymətini və dəyişiləcək damanın ünvanını göstərib *OK* düyməsini sıxmaqla düsturda iştirak edən damalardakı ədədi qiyməti tələb olunan nəticəyə uyğun nizamlamaq olar.

ə) *Asılılıqlar* (Зависимости) əmri cari sənəddə düsturda baş vermiş səhvlərin mənbəyini, asılı və təsvir edən damaları göstərir.

f) *Tənzimləmə* və *Parametrlər* (Настройка, Параметры) əmrləri ilə proqram pəncərəsini tələb olunan formada tənzimləmək olur.

7) *Verilənlər* menyu bölməsində aşağıdakı alt bölmələr var:

a) *Çeşidləmə* (Сортировка) əmri cari sənəddəki verilənləri əlifba sırası ilə, eləcə də, artma və azalma üzrə çeşidləməyə imkan verir.

b) *Filtr* (*Фильтр*) əmri ilə cari sənəddə yalnız müəyyən şərtləri ödəyən verilənlərin əks olunmasını təmin etmək olur.

c) *Forma* (*Форма*) əmri ilə açılan pəncərədəki *Əlavə etmək*, *Ləğv etmək*, *Geriyə*, *Davamı* və *Kriteriya* bölmələrinin köməyilə cari vərəqə verilənlər daxil etmək, onları ləğv etmək, əvvəlki, sonrakı verilənlərə baxmaq və müəyyən şərtləri ödəyən verilənləri axtarıb tapmaq olur.

ç) *Yekun* (*Итоги*) əmri aralıq, yekun və ümumi yekun nəticələri tapmağa imkan verir.

d) *Yoxlama* (*Проверка*) əmri ilə açılan pəncərədə daxil ediləcək verilənlərin tipini, alacaq qiymətlər oblastını verməklə, verilənlərin daxil edilməsi zamanı səhvlərin qarşısını almaq olur.

e) *Sütünlar üzrə mətn* (*Текст по столбцам*) əmri ilə cari damadakı mətn bir neçə sütun şəklində ifadə olunur.

ə) *Konsolidasiya* (*Консолидация*) əmri ayrı-ayrı diapazonlarda olan ədədi verilənlər üzərində əməliyyatlar aparmağa imkan yaradır.

f) *Qruplar və strukturlar* (*Группы и структуры*) əmri

cari sənəddə qeyd olunmuş sətir və ya sütunları qruplaşdırır, eləcədə, düstur və istinadlar əsasında qeyd olunmuş səhifənin strukturunu yaradır.

g) *Yekun cədvəllər* (*Сводные таблицы*) əmri cari verilənlər üçün yekun cədvəl hazırlayır.

ğ) *Xarici verilənlər* (*Внешние данные*) əmri ilə verilənlər bazasına göndərilən sorğu əsasında alınan verilənləri cari sənədə daxil etmək olur.

h) *Verilənlərin yeniləşdirilməsi* (*Обновление данных*) əmri sorğu ilə cari sənədə daxil edilmiş verilənləri yeniləşdirə bilir.

8) *Pəncərə* menyu bölməsinin aşağıdakı alt bölmələri var:

a) *Yeni* (*Новое*) əmri cari pəncərənin tərkibi ilə eyni olan yeni proqram pəncərəsi açır.

b) Yerləşdirmək (*Расположить*) əmri ilə ekrana verilən dialoq pəncərəsində Yanaşı, Aşağıdan yuxarı, Soldan sağa, Pillələrlə formalarından hər hansı birini seçərək açılmış bütün pəncərələri ekrana həmin formada gətirmək olur.

c) *Gizlətmək* (*Скрыть*) əmri ilə cari pəncərəni gizlətmək olur.

ç) *Əks etdirmək* (*Отобразить*) əmri ilə cari pəncərəni ekranda əks etdirmək olur.

d) *Bölmək* (*Разделить*) əmri ilə cari pəncərəni seçdiyimiz hissədən iki yerə bölmək olar.

e) *Sahəni qeyd etmək* (*Закрепить области*) əmri cari pəncərədə seçilmiş sahəni qeyd etməyə imkan verir.

9) *Arayış* menyu bölməsi proqramda iş qaydaları haqqında müxtəlif arayışlar almağa imkan verir.

İndi isə MS Excel 2000 proqramında iş qaydalarına baxaq. MS Excel faylı işçi kitab adlanır. İşçi kitab isə adları (Vərəq 1, Vərəq 2,...) işçi kitabın pəncərəsinin aşağı hissəsində sadalanan işçi vərəqlərdən ibarətdir. Bu vərəq adlarını siçanın sol düyməsi ilə sıxmaqla kitab daxilində bir vərəqdən digərinə keçmək olur. İşçi vərəq 256 sütundan və 65536 sətirdən ibarət cədvəldir. Sütunlar latın əlifbasının hərfləri ilə, sətirlər isə rəqəmlərlə adlandırılır. Cədvəlin hər bir daması sətrin və sütunun adından ibarət ünvana malikdir. Məsələn, *F7* ünvanı, damanın *F* sütunu ilə 7 sətrinin kəsişdiyi yerdə durduğunu bildirir. Cədvəlin damalarından biri həmişə aktiv olur. Aktiv dama çərçivəyə alınır. Damanı aktiv etmək

üçün onu siçanın sol düyməsi ilə sıxmaq kifayətdir. Bir-birinin yanında yerləşən bir neçə damanı qeyd etmək üçün kursoru damalardan birinə yerləşdirib siçanın sol düyməsini sıxıb saxlamaqla siçanı hərəkət etdirmək lazımdır. Bir-birinə yaxın olmayan dama qrupları ayırarkən, əvvəlcə bir qrup dama qeyd edib, sonra *Ctrl* düyməsini sıxıb saxlayaraq digər dama qruplarını da qeyd etmək lazımdır. Cədvəldə bütöv sütun və ya sətr qeyd etmək üçün, onun adını siçanın sol düyməsi ilə sıxmaq lazımdır. Bir neçə sütun və ya sətir qeyd etmək tələb olunduqda isə, birinci sütun və ya sətrin adını siçanın sol düyməsi ilə sıxıb saxlayaraq, qeyd etməni bütün oblasta şamil etmək olar. Bir neçə vərəqi qeyd etmək üçün *Ctrl* düyməsini sıxıb saxlayaraq vərəq adlarını siçanın sol düyməsi ilə sıxmaq lazımdır.

Damaya verilənləri daxil etmək üçün onu aktiv edib, verilənləri klaviaturadan daxil etmək lazımdır. Bu zaman verilənlər dama ilə yanaşı redaktə etmə sətrinə də çıxarılır. Daxil etməni sona çatdırmaq üçün *Enter* düyməsini sıxmaq lazımdır. Bu zaman verilənlərin daxil edilməsi prosesi sona çatır və növbəti dama aktiv olur. Damadakı verilənləri redaktə etmək üçün damanı aktiv edib, *F2* düyməsini sıxmaq və ya damanı siçanın sol düyməsi ilə ikiqat sıxmaq lazımdır. Bu zaman kursor damaya gətirilir, redaktə işinin sonunda *Enter* düyməsini sıxmaqla redaktə etmək rejimindən çıxmaq olur. Yeni işçi kitab yaratmaq üçün *Fayl* bölməsindəki *Yaratmaq* əmrini seçib, açılan dialoq pəncərəsində işçi kitabın yaradılacağı şablonu qeyd edib, *OK* düyməsini sıxmaq lazımdır.

Adi işçi kitablar kitab şablonu əsasında yaradılır. Bu cür işçi kitabı  düyməsini sıxmaqla da yaratmaq olar. Mövcud işçi kitabın açılması üçün Fayl bölməsinin *Açmaq* əmrini seçmək və ya  düyməsini sıxmaq lazımdır. Nəticədə açılan pəncərənin *Qovluq* sahəsində kitabın yerləşdiyi diski, ondan aşağıdakı sahədə isə qovluğu və kitabın özünü seçmək lazımdır. İşçi kitabı yaddaşda saxlamaq üçün *Fayl* bölməsinin *Saxlamaq* əmrini seçmək və ya  düyməsini sıxmaq lazımdır. Sənəd yaddaşda ilk dəfə saxlanılarkən, ekrana verilən pəncərədə *Qovluq* sahəsində kitabın saxlanılacağı diski, ondan aşağıdakı sahədə qovluğu, *Fayl tipi* sahəsində formatı, *Fayl adı* sahəsində isə kitabın adını vermək və buradakı *Saxlamaq* düyməsini sıxmaq lazımdır. Kitab

yaddaşda təkrarən saxlandıqda, o avtomatik olaraq həmin faylda saxlanılacaqdır. Kitabı başqa adla və ya başqa qovluqda yadda saxlamaq tələb olunduqda, Fayl bölməsinin Necə saxlamaq əmrini seçmək lazımdır. İşçi kitabı bağlamaq üçün *Fayl* bölməsinin *Bağlamaq* əmrini seçmək və ya  düyməsini sıxmaq lazımdır.

Cədvəldəki hesablamalar düsturlarla yerinə yetirilir. Düstur riyazi operatorlardan, qiymətlərdən, funksiya və dama adlarına istinadlardan ibarət ola bilər. Düsturun yerinə yetirilmə nəticəsi düsturun yerləşdiyi damadakı qiymət olur. Düstur «=» işarəsi ilə başlayır və düsturda +, -, *, / hesab əməl operatorlarından istifadə edilə bilər. Düsturdakı hesablama qaydaları adi riyazi qanunlarla tənzimlənir. Düstura misal olaraq *=(A5+B9)*C7; =D7*T13+K17* və s. göstərmək olar. Sabitlər – damaya daxil edilən və hesablamalar zamanı dəyişdirilə bilməyən mətni və ədədi qiymətlərdir. Damaya müraciət onun koordinatlarının daxil edilməsi ilə yerinə yetirilir. Boş damanın qiyməti sıfır qəbul edilir. Damaya müraciət iki tip ola bilir:

1) nisbi müraciət – bu zaman dama, düstur verilmiş damaya nisbətən sürüşdürülməklə işarə olunur (məsələn, *F7*).

2) mütləq müraciət – bu zaman dama $ işarəli dama koordinatları ilə işarə olunur (məsələn, *F7*). Eləcə də bu tiplərin kombinasiyalarından da istifadə olunur (məsələn, *F$7*).

Damalar qrupuna müraciət üçün xüsusi işarələrdən istifadə olunur:

1) : (iki nöqtə) işarəsi damalar blokuna müraciəti formalaşdırır. Bu işarə ilə blokun yuxarı sol və aşağı sağ damaları işarələnir. Məsələn, *C4:D6* ifadəsi *C4, C5, C6, D4, D5, D6* damalarına müraciət deməkdir.

2) ; (nöqtə vergül) işarəsi damaların birləşdirilməsi deməkdir. Məsələn, *D2:D4; D6:D8* ifadəsi *D2, D3, D4, D6, D7, D8* damalarına müraciət deməkdir. Düsturu damaya daxil etmək üçün «=» işarəsini və hesablama düsturunun ifadəsini vermək lazımdır. *Enter* düyməsini sıxmaqla damada hesablamanın nəticəsini almaq olar.

MS Excel proqramında funksiya müəyyən məsələlərin həlli üçün bir neçə hesablama əməliyyatını birləşdirə bilir. Burada funksiyalar bir və ya bir neçə arqumenti olan düsturlardır. Arqu-

ment kimi ədədi qiymətlər və ya dama ünvanları göstərilir. Məsələn, *=СУММ(A5:A9) A5, A6, A7, A8, A9* damalarının cəmini, *=СРЗНАЧ(B4:B6)* isə *B4, B5, B6* damalarındakı qiymətlərin ədədi ortasını tapır. Funksiyalar biri digərinin tərkibində də ola bilir. *Məsələn,*

=СУММ(C1:C20)+ОКРУГЛ(СРЗНАЧ(D4:D9);2); və s.

Funksiyanı damaya daxil etmək üçün damanı düstur üçün qeyd edib sonra *Daxil etmək* bölməsinin *Funksiya* əmrini seçməklə və ya düyməsini sıxmaqla açılan dialoq pəncərəsində *Kateqoriya* sahəsində funksiyanın tipini, *Funksiya* sahəsində isə funksiyanı seçib *OK* düyməsini sıxmaq lazımdır. Burada pəncərənin Число1, Число2 və s. sahələrində funksiyanın arqumentlərini (ədədi qiymət və ya damalara müraciət) daxil etmək tələb olunur. Qeyd edək ki, damaya СУММ cəmləmə funksiyasını düyməsini sıxmaqla da daxil etmək olar.

Cədvəl şəklində daxil edilmiş verilənlərin emalı üçün və eyni tipli düsturların daxil edilməsi üçün düsturlar massivindən istifadə etmək əlverişli olur. Məsələn, *B1, C1, D1, E1* damalarındakı ədədlərin mütləq qiymətlərini hesablamaq üçün hər bir damaya ayrı-ayrılıqda düstur tətbiq etmək əvəzinə, bütün damalar üçün massiv daxil etməklə bir düsturla kifayətlənmək olar. Proqramda massivi qeyd etmək üçün düsturlar massivi ətrafına { } fiqurlu mötərizə əlavə edilir. Məsələn, *{=ABS(B1:E1)}*. Düsturlar massivini yaratmaq üçün aşağıdakı əməliyyatlar ardıcıllığını yerinə yetirmək lazımdır: düsturlar massivinin yerləşdiyi damaları qeyd etmək, düsturu adi qayda ilə daxil edib, arqumentlər kimi arqument-damalar qrupunu daxil edib, sonda *OK* düyməsi əvəzinə *Ctrl+Shift+Enter* düymələri kombinasiyasından istifadə etmək lazımdır.

Damaların daxil edilməsi üçün onların əvəz edəcəkləri damaları cədvəldə qeyd edib, *Daxil etmək* bölməsində *Damalar* alt bölməsini seçmək lazımdır. Nəticədə açılan pəncərədə daxil edilən elementin tipini (sağa sürüşdürməklə dama, aşağı sürüşdürməklə dama, sətir, sütun) seçib, *OK* düyməsini sıxmaq lazımdır. Sətir və ya sütunların daxil edilməsi üçün onların əvəz edəcəkləri sətir və ya sütunları qeyd edib, *Daxil etmək* bölməsinin *Sətirlər* və ya *Sütunlar* alt bölməsini seçmək lazımdır. Cədvəlin elementlərini ləğv etmək üçün onları qeyd edib, *Düzəlişlər* menyu

bölməsinin *Ləğv etmək* əmrini seçmək lazımdır. Cədvəldəki damaların daxilindəkiləri onların özlərini ləğv etmədən silmək üçün bu damaları qeyd edib, *Düzəlişlər* menyu bölməsinin *Təmizləmək* əmrini seçmək və ya *Delete* düyməsini sıxmaq lazımdır. Damaların tərkibinin təkrarını almaq və yerini dəyişdirmək üçün mübadilə buferindən (Clipboard) istifadə edilir. Damanın tərkibinin təkrarını almaq üçün əvvəlcə həmin damaları qeyd etmək, sonra isə *Düzəlişlər* bölməsinin *Təkrarını almaq* əmrini seçmək və ya düyməsini sıxmaq lazımdır. Buferdə saxlanılan verilənləri damaya daxil etmək üçün həmin damaları qeyd edib *Düzəlişlər* bölməsinin *Daxil etmək* əmrini seçmək və ya düyməsini sıxmaq lazımdır. Damaların tərkibini başqa damalara köçürmək üçün tərkibi köçürüləcək damaları qeyd edib, *Düzəlişlər* menyu bölməsinin *Kəsmək* əmrini seçmək və ya düyməsini sıxmaq lazımdır. Sonra bu fraqmentin köçürüləcəyi oblastın yuxarı sol damasını qeyd edib *Düzəlişlər* bölməsinin *Daxil etmək* əmrini seçmək və ya düyməsini sıxmaq lazımdır. Cədvəlin ixtiyari obyekti üzərində siçanın sağ düyməsini sıxmaqla bu obyektin emalı üçün tələb olunan əmrlər ardıcıllığı verilən kontekst menyusu açılır.

MS Excel 2000 proqramında 12 yaddaş oyuğu olan mübadilə buferi var. Onun köməyi ilə cədvəl fraqmentlərinin təkcə Excel daxilində deyil, həm də Windows-un digər proqramları daxilində təkrarını alıb köçürmək olur. Buferi ekrana çıxarmaq üçün *Görünüş* menyu bölməsinin *Alətlər lövhəsi* alt bölməsinin *Mübadilə buferi* əmrini seçmək lazımdır. Fraqmentin təkrarını buferə salmaq üçün onu qeyd edib, düyməsini sıxmaq lazımdır. Buferdəki fraqmenti sənədə yerləşdirmək üçün onun buradakı nişanını siçanın sol düyməsi ilə sıxmaq lazımdır.

Kitabdakı vərəqlərin adını dəyişdirmək üçün onun yarlıkını siçanın sol düyməsi ilə ikiqat sıxıb yeni adı daxil edirlər. İşçi kitabın vərəqlərinin təkrarını almaq və yerlərini dəyişdirmək üçün vərəqlərinin təkrarı alınacaq kitabı və vərəqlər köçürüləcək kitabı açmaq, vərəqi qeyd edib, *Düzəlişlər* menyu bölməsinin *Yerini dəyişmək/Təkrarını almaq* əmrini seçmək lazımdır. Bu zaman açılan pəncərənin *Kitaba* yazı sahəsində vərəqləri qəbul edəcəyi kitabın adını seçmək, *Vərəqin qarşısına* yazı sahəsində isə qarşısında köçürülən vərəq yerləşdiriləcək vərəqi seçmək lazımdır. Bu zaman

vərəqin təkrarını almaq tələb olunarsa, onda pəncərədəki uyğun bölməni qeyd etmək lazımdır. Vərəqin ləğv edilməsi üçün onu aktiv edib, *Düzəlişlər* bölməsinin *Vərəqi ləğv etmək* əmrini seçmək lazımdır. Vərəq daxil etmək üçün əvəzlənəcək vərəqi qeyd edib *Daxil etmək* menyu bölməsinin *Vərəq* əmrini seçmək lazımdır.

Cədvəldə hər bir ədədi müxtəlif formatlarda vermək olar. Formatı dəyişdirmək üçün damanı qeyd edib, *Format* bölməisnin *Damalar* əmrini seçmək lazımdır. Nəticədə açılan pəncərədə *Ədəd* bölməsini seçib, *Ədəd formatları* siyahısında formatın tipini, ondan sağdakı sahədə isə formatın parametrlərini seçmək lazımdır.

Cədvəldə susmaqla damalar standart en və hündürlüyə malikdir. Sətrin hündürlüyü şriftin ölçüsü ilə təyin edilir. Sətrin hündürlüyünü və sütunun enini onların başlıqlarının sərhədlərini lazım olan qiymətə qədər sürüşdürməklə dəyişmək olar. Əgər sütunların başlıqları sərhəddində siçanın sol düyməsini ikiqat sıxsaq, sütunun eni ən uzun tərkibli damanın eni qədər olur. Cədvəldə sütunların (sətirlərin) enini (hündürlüyünü) dəqiq təyin etmək üçün sütunları (sətirləri) qeyd edib, *Format* menyu bölməsinin *Sütun (Sətir)* alt bölməsində *En (Hündürlük)* əmrini seçmək lazımdır. Nəticədə açılan pəncərədə *Sütunun eni (Sətrin hündürlüyü)* yazı sahəsində qiymətlər daxil edib, *OK* düyməsini sıxmaq lazımdır.

Sətir və ya sütunları cədvəldə gizlətmək üçün həmin sətir və ya sütunları qeyd edib, *Format* bölməsinin *Sətirlər* və ya *Sütunlar* alt bölmələrini seçib, buradakı *Gizlətmək* əmrini vermək lazımdır. Sətir və ya sütunları cədvəldə əks etdirmək üçün isə gizlədilmiş sətir və ya sütunun hər iki tərəfindən sətirləri və ya sütunları qeyd edib, *Format* bölməsində *Sətirlər* və ya *Sütunlar* alt bölmələrini seçib *Əks etdirmək* əmrini vermək lazımdır.

MS Excel-in köməyilə verilənlər bazası yaradıb onu emal etmək olur. Burada verilənlər bazası – eyni tipli yazılışlardan (sətirlərdən) ibarət cədvəldir. Cədvəlin sütunları verilənlər bazasında yazı sahələridir. Yazı sahələrinin adları üçün verilənlər bazasının birinci sətri ayrılır. Verilənlər bazası ilə işləmək üçün əvvəlcə uyğun cədvəli yaratmaq lazımdır. Əgər cədvəldə dama qeyd edib *Verilənlər* bölməsinin verilənlər bazasının emal etmə əmrlərindən ixtiyari birini seçsək, proqram avtomatik olaraq bütün cədvəli təyin edir və emal edir. Verilənlərin cədvəldə çeşidlənməsi üçün bir damanı qeyd edib *Verilənlər* bölməsinin *Çeşidləmə* əmrini seçmək lazımdır. Bu zaman ədədlər artım və azalma sırası ilə, mətn əlifba sırası ilə, məntiqi ifadələr isə əvvəlcə yalan, sonra doğru qiyməti verilməklə çeşidlənirlər. Verilənlər bazası ilə iş üçün xarakterik olan əməliyyatlar (axtarış, çeşidləmə, yekunların vurulması) zamanı, proqram cədvəli avtomatik olaraq verilənlər bazası kimi qəbul edir.

Diaqramma – cədvəllərdəki verilənlərin müqayisəsi və analizi üçün onların qrafik şəkildə ifadəsidir. Diaqrammada damalardakı ədədi verilənlər nöqtə, xətt, zolaq, sütün, sektor və s. şəklində təsvir olunur. Diaqrammanı qurmaq üçün iş vərəqində onun uyğun verilənlərini qeyd edib, *Daxil etmək* bölməsinin *Diaqramma* əmrini seçmək lazımdır. Açılan dialoq pəncərəsində diaqrammanın tipini, formatını və digər parametrlərini təyin etmək olar. Diaqrammanın tip və qurulma parametrlərini dəyişdirmək üçün siçanın sağ düyməsini onun üzərində sıxmaqla açılan kontekst menyusunda uyğun əmri seçmək lazımdır. Diaqrammanı ləğv etmək üçün onu qeyd edib *Delete* düyməsini sıxmaq lazımdır.

Cədvəlləri çapa göndərməzdən əvvəl *Fayl* bölməsinin *Səhifənin parametrləri* əmri ilə səhifələrin parametrlərini təyin etmək olur. Açılan pəncərədə *Səhifə* bölməsində vərəqin ölçüləri və oriyentasiyası, təsvirin miqyası və çapın keyfiyyəti təyin edilir. Cədvəli çapa vermək üçün *Fayl* menyu bölməsinin *Çap* əmrini seçmək lazımdır. Açılan pəncərədə *Ad* sahəsində printeri, *Çap* bölməsində çapa veriləcək səhifələri təyin etmək olur. Burada *Çapa çıxarmaq* bölməsi ilə qeyd olunmuş diapazon vərəqlərini, qeyd olunmuş vərəqləri və ya bütün kitabı çapa vermək olar. *Surətlərin sayı* bölməsi çapa çıxarılan nüsxələrin sayını təyin edir.

IV FƏSİL

PROQRAMLAŞDIRMAYA GİRİŞ

4.1. Alqoritm anlayışı

İnformatika elminin əsaslarından biri olan alqoritm anlayışı, EHM-dən əvvəl yaranmış və riyaziyyatın əsas anlayışlarından biri olmuşdur. «Alqoritm» sözü özbək riyaziyatçısı Əl-Xorezminin (IX əsr) adından götürülüb və riyaziyatda dörd cəbri əməliyyatın (toplama, çıxma, vurma və bölmə) yerinə yetirilmə qaydalarını işarə etmək üçün istifadə olunub. Hal-hazırda alqoritm anlayışı təkcə riyaziyyatda istifadə olunmur. Bu anlayışdan insan fəaliyyətinin bir çox sahələrindən istifadə olunur. Məsələn, istehsal prosesinin idarə edilməsi alqoritmindən, şahmat oyununun alqoritmindən, labirintdə yolun tapılması alqoritmindən, kosmik raketin uçuşunun idarə edilməsi alqoritmindən və s. danışmaq olar.

Alqoritm – verilmiş məsələnin həllinin tapılması üçün zəruri olan əməliyyatlar ardıcıllığının dəqiq və sadə təsviridir. Alqoritm anlayışının izahı üçün «alqoritmin icraçısı» anlayışının böyük əhəmiyyəti vardır. Belə ki, alqoritm hər hansı konkret bir icraçı üçün, məsələn, insan üçün, xüsusi maşın – EHM üçün və s. qurulur. Beləliklə, demək olar ki, alqoritm – qoyulmuş məsələni həll etmək və ya müəyyən məqsədə çatmaq üçün müəyyən əməllər ardıcıllığının yerinə yetirilməsi haqqında icraçıya verilən dəqiq və aydın göstərişlərdir.

Alqoritm, onun tərifini açıqlayan aşağıdakı əsas xassələrə malikdir:

1. ***Diskretlik xassəsi.*** Alqoritm diskret (latınca- fasiləli) olmalıdır. Diskret dedikdə, fasilələk kəsilməzliyə qarşı qoyulur. Məsələn, hər hansı kəmiyyətin zamana görə diskret dəyişməsi - müəyyən fasilələrlə (sıçrayışlarla) baş verən dəyişmədir və ya tam ədədlər çoxluğu həqiqi ədədlər çoxluğuna qarşı diskretdir.

Bu xassə ondan ibarətdir ki, alqoritm məsələnin həlli prosesini, sadə addımların yerinə yetirilməsi ardıcıllığı şəklində ifadə etməlidir və hər bir addımın yerinə yetirilməsi üçün sonlu zaman fasiləsi tələb olunur, yəni başlanğıc verilənlərin araşdırılması və nəticənin alınması zamana görə diskret yerinə yetirilir.

2. ***Müəyyənlik xassəsi.*** Bu xassə ondan ibarətdir ki, alqoritmin hər bir addımı dəqiq, birqiymətli və aydın olmalıdır. Bu xassəyə əsasən alqoritmin yerinə yetirilməsi mexaniki xarakter daşımalı və həll olunan məsələ haqqında əlavə məlumat və ya göstəriş tələb etməməlidir.

3. ***Kütləvilik xassəsi.*** Bu xassəyə əsasən müəyyən məsələnin həlli üçün qurulan alqoritm ümumi şəkildə qurulur, yəni alqoritm yalnız başlanğıc qiymətləri ilə fərqlənən müəyyən sinif məsələlərin həlli üçün tətbiq edilə bilsin. Burada başlanğıc qiymətlər də, alqoritmin tətbiq oblastı adlanan hər hansı eyni bir oblastdan seçilə bilər. (Bəzi hallarda başlanğıc qiymətlər heç verilməyə də bilər).

4. ***Nəticəlilik və ya sonluluq xassəsi.*** Bu xassə onu bildirir ki, alqoritm, qoyulmuş məsələnin həllinə sonlu sayda addımların yerinə yetirilməsi ilə gətirməlidir, yəni alqoritm sonlu sayda addımdan sonra başa çatmalı və verilmiş məsələnin həlli tapılmalıdır.

4.2 Alqoritmlərin tipləri və ifadə formaları

Qurulmuş alqoritmi bir neçə üsulla ifadə etmək olar:

1. *Sözlərlə*
2. *Xüsusi sxemlər – blok-sxemlərlə*
3. *Alqoritmlərin yazılışı üçün xüsusi dildə, alqoritmik dildə.*

Alqoritmin sözlərlə ifadəsindən, alqoritm icraçısı insan olduqda istifadə olunur. Bu zaman alqoritmin addımları nömrələnir ki, onlara müraciət etmək mümkün olsun. Misal üçün Evklid alqoritmini nəzərdən keçirək. Bu alqoritm verilmiş iki natural ədəd üçün ən böyük ortaq böləni (ƏBOB) təyin edir. Verilmiş ədədləri M və N ilə işarə edək, onda

1) Əgər $M<>N$ olarsa, 2 bəndinə, əks halda 5 bəndinə keç,
2) Əgər $M>N$ olarsa, 3 bəndinə, əks halda 4 bəndinə keç,
3) M-dən N-i çıxmalı və bu fərqin nəticəsini M-ə mənimsətməli. 1 bəndinə keçməli.
4) N-dən M-i çıxmalı və bu fərqin nəticəsini N-ə mənimsətməli. 1 bəndinə keçməli.
5) ƏBOB-un M-ə bərabər olduğunu qəbul etmək.
6) Son.

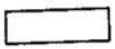
Digər bir misal, verilmiş kvadrat tənliyin həlli üçün alqoritm:

1) Tənliyin a, b, c əmsallarını daxil etmək.
2) $D = b^2 - 4ac$ ifadəsini hesablamaq.
3) Əgər $D < 0$ olarsa, 5 bəndinə, əks halda 4 bəndinə keç.
4) $x_1 = \frac{-b + \sqrt{D}}{2a}$, $x_2 = \frac{-b - \sqrt{D}}{2a}$ hesablamalı
5) Hesablamaları qurtarmalı. Son.

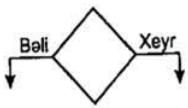
Alqoritmlərin blok-sxemlər vasitəsilə ifadəsi zamanı onlar əyani qrafik formada ifadə olunurlar. Alqoritmin əmrləri onların yerinə yetirilmə ardıcıllığını göstərən oxlarla birləşdirilən bloklar içərisinə yerləşdirilir.Bu blokların qrafik təsviri üçün aşağıdakı standartlar qəbul edilmişdir:

Başlanğıc və sonu göstərmək üçün.

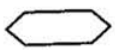
Paraleloqram. Daxil etmə və çıxışı göstərmək üçün.



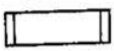
Düzbucaqlı. Hesablamaları göstərmək üçün.



Romb. Şərtin yoxlanılması üçün.

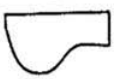


Dövrün başlanğıcını göstərmək üçün.



Alt proqramların göstərilməsi üçün.

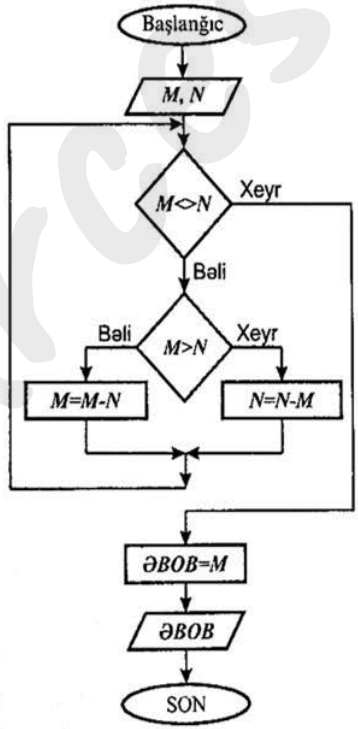
Eyni bir səhifədə yerləşən blokları birləşdirmək üçün.



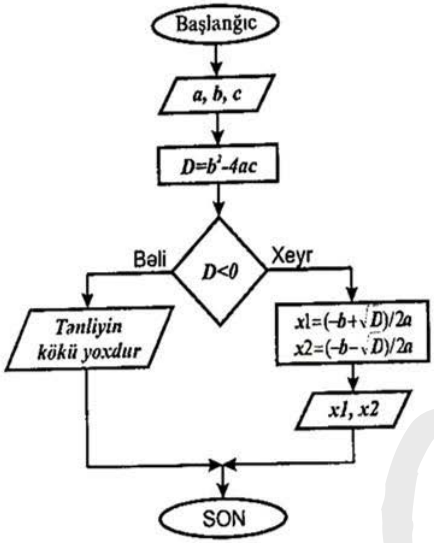
Cavabın çap qurğusuna verildiyini göstərmək üçün.

Müxtəlif səhifələrdə olan blokları birləşdirmək üçün.

Blok-sxemlərdə alqoritmin bloklar daxilində verilən əmrləri daha qısa şəkildə ifadə olunur, belə ki, burada həndəsi fiqurlar sözləri əvəz edir. Məsələn, Evklid alqoritmi üçün uyğun blok-sxem aşağıdakı şəkildə olar.

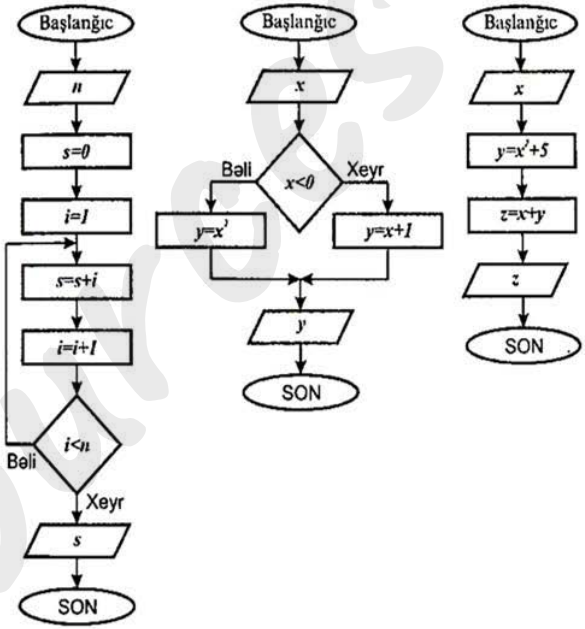


Kvadrat tənliyin həlli üçün alqoritmin blok-sxemi isə aşağıdakı kimi olar:



Alqoritmlər strukturuna görə üç əsas tipə bölünürlər: xətti strukturlu, budaqlanan strukturlu və dövr strukturlu alqoritmlər. Əgər alqoritmin təşkil etdiyi N sayda addımlar, bir-birinin ardınca başlanğıcdan sona qədər ardıcıl yerinə yetirilirsə, belə alqoritmə xətti strukturlu alqoritm deyilir. Əgər alqoritmin addımlarının yerinə yetirilmə ardıcılığı, müəyyən şərtlərin ödənilməsindən asılı olaraq dəyişirsə, belə alqoritm budaqlanan strukturlu alqoritm adlanır. Şərt isə məntiqi ifadə olub, yalnız iki qiymət ala bilər: «hə» - əgər şərt doğrudursa və «yox» - əgər şərt yalandırsa. Əgər alqoritmin müəyyən addımlar, ardıcılığı, müəyyən şərtin ödənməsindən asılı olaraq bir neçə dəfə təkrarlanırsa, belə alqoritm dövr strukturlu alqoritm adlanır.

Məsələn, bu alqoritmlər üçün aşağıdakı misallara baxaq:



Bu məsələlərin şərtləri uyğun olaraq aşağıdakı kimidir: 1) İlk n natural ədədin cəmini tapmalı; 2) Əgər $x<0$ olarsa, y-ə x^2, əks halda y-ə $x+1$ mənimsətməli ; 3) x ədədi verilib, $y=x^2+5$, $z=x+y$ ifadələrini hesablayıb, z-i çapa verməli.

4.3 Alqoritmlərin qurulma qaydaları

Alqoritmlərin qurulma qaydalarını, aşağıdakı misallar üzərində tədqiq edək:

1) İki natural N və M ədədlərinin K nisbətini və L qalıq həddinin tapılması, harada $L<M$ və K-tam ədəddir (sxem1).

2) $S=\begin{cases}-1, & x<0\\ 0, & x=0\\ 1, & x>0\end{cases}$ funksiyasının hesablanması (sxem2).

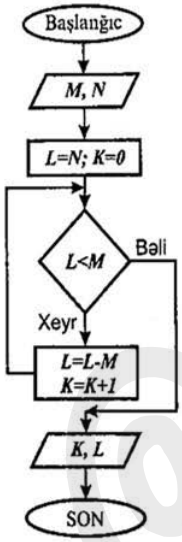
3) x, y, z ədədləri içində maksimalının tapılması (sxem3).

4) $y=f(x)$ funksiyasının x_0-dan x_k-ya qədər h sabit addımı ilə dəyişən x arqumenti üçün qiymətlərinin hesablanması (sxem4).

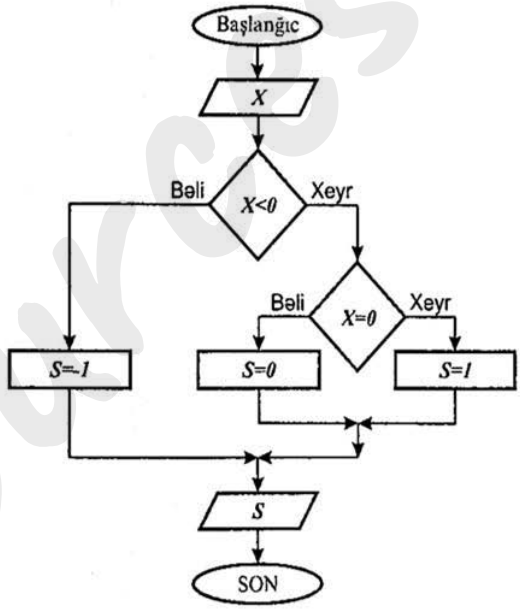
5) $S=\sin x+\sin(3x)/3+\ldots=\sum_{i=1}^{\infty}\frac{\sin(2i-1)x}{2i-1}$ sırasının cəmini verilmiş x üçün ε dəqiqliyi ilə hesablamalı (sxem5).

6) Verilmiş sonlu ədədlər ardıcıllığında maksimal və minimal elementlərin təyini (sxem6).

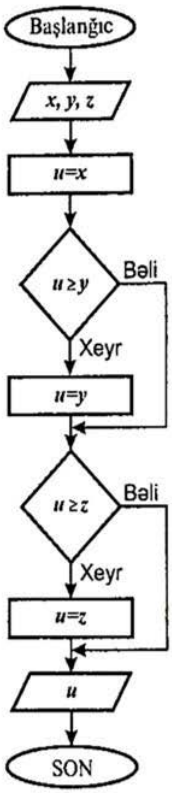
7) Verilmiş müəyyən inteqralı trapeslər düsturu ilə ε dəqiqliyi ilə hesablamalı (sxem7).



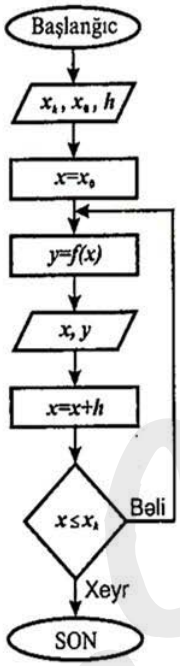
Sxem 1.



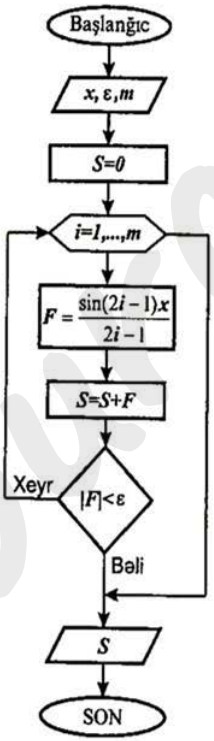
Sxem 2



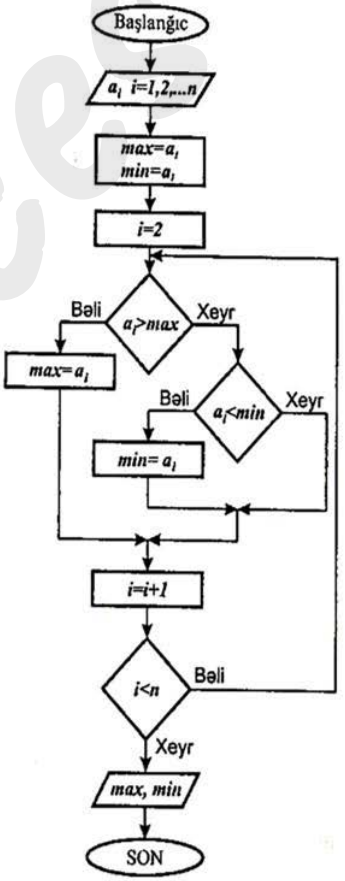
Sxem 3.



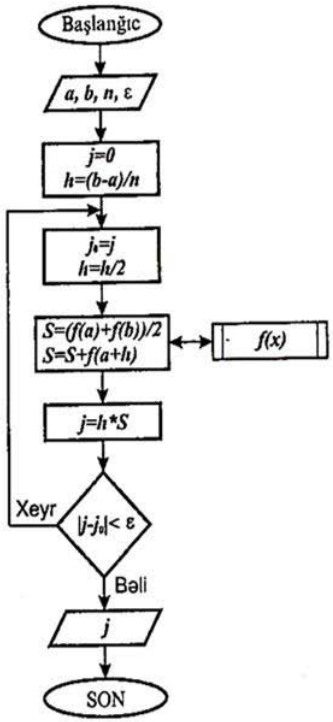
Sxem 4.



Sxem 5.



Sxem 6.



Sxem 7.

4.4. Alqoritmik dillər

Hesablama texnikasının inkişafı proqramlaşdırma dillərinin yaranmasını təmin etdi. Proqramlaşdırma dili - məsələnin EHM-də həlli üçün verilən alqoritminin yazıldığı dildir.

Birinci nəsl EHM-də proqramlaşdırma, yalnız maşın dilində aparılırdı. Maşın dili müəyyən əməliyyatların, əsasən cəbri əməliyyatların ədədi şəkildə kodlaşdırılma qaydalarının ardıcıllığından ibarət olur. EHM-də yerinə yetirilməli olan hər bir əməliyyat, maşın dilində əmrlər şəklində ifadə olunur. Hər bir əmr isə, maşın əməliyyatı adlanan və informasiya emalı prosesinin hər hansı elementar hissəsi olan bir əməliyyatı ifadə edir.

Əmrdə ümumi şəkildə, aparılacaq əməliyyatın məzmunu haqqında məlumat, üzərində maşın əməliyyatı aparılacaq başlanğıc verilənlərin yerləşdiyi yer - ünvan, nəticənin ünvanı və bu əmrdən sonra yerinə yetiriləcək əmr haqqında məlumat verilməlidir.

İkinci nəsl EHM-lərin yaranması, konkret maşının yox, qoyulmuş məsələnin xüsusiyyətlərindən asılı olan dillərə ehtiyac yaratdı. Bu dillərə formal alqoritmik və ya sadəcə alqoritmik dillər deyilir. Bu dillər üzərinə bir çox şərtlər qoyulur. Belə ki, birincisi bu dillər əyani olmalı, bunun üçün isə burada məlum riyazi simvolika və digər asan başa düşüləcək təsviri vasitələrdən istifadə olunmalıdır, ikincisi bu dillərdə ixtiyari alqoritm asanlıqla ifadə edilə bilməlidir, üçüncüsü, bu dillərdə qurulan alqoritm birqiymətli qəbul edilməli və digər məna daşımamalıdır, dördüncüsü, qurulan mürəkkəb alqoritm daha sadə alqoritmlərin vəhdəti şəklində ifadə edilə bilməlidir. Bunlardan əlavə bu dillər insanla maşın arasında ünsiyyət yaratmalıdır, belə ki, alqoritmik dildən maşın dilinə tərcümə EHM tərəfindən xüsusi proqram- translyator vasitəsilə yerinə yetirilir. Proqram dedikdə isə biz, EHM-ə verilən göstərişləri ifadə edən operatorlar yığını başa düşürük. Birinci və kifayət qədər yaxşı alınmış dil 1954-cü ildə IBM firmasının yaratdığı FORTRAN dili idi. Bu dilin adı FORTRAN-FORmulae TRANslation - formulaların tərcüməsi sözündən götürülmüşdür. Bu dil çox sadə struktura malik olduğundan ondan hal-hazırki vaxta qədər istifadə olunur. Fortranda proqram operatorlar ardıcıllığı şəklində yazılır. Bu dildə yazılan proqram bir və ya bir neçə seqmentlərdən (alt proqramlar) ibarət olur. Bütün proqramın işini idarə edən seqment əsas proqram adlanır.

Fortran dili elmi və mühəndis - texniki hesablama sahələrində istifadə edilmək üçün nəzərdə tutulmuşdu. Lakin bu dildə budaqlanan strukturlu məsələlər (istehsal prosesinin modelləşdirilməsi və s.), bəzi iqtisadi məsələlər və redaktə etmə məsələləri (cədvəl, arayış və s. qurulması) üçün proqramlar da qurula bilər. Sonrakı illərdə bu dilin müxtəlif modifikasiyaları yaradılmışdı.

Fortran dilinin əsasında 1966-cı ildə Dartmut kollecinin hesablama mərkəzində Beyzik dili (BASIC- Beginner's All-purpose Symbolic Instruction Code - yeni başlayanlar üçün çox-məqsədli simvolik komandalar dili) və 1975-ci ildə Digital Equipment Corporation firması tərəfindən Beyzik- plyus (BASIC - PLUS) (Beyzik dilinin genişlənməsi) dili yaradıldı. Hal-hazırda da bu dillər proqramlaşdırma praktikasında alqoritmik dillərdən istifadə etmək vərdişlərinin alınması üçün ən yaxşı dillərdən biridir.

1960-cı ildə Alqol-60 (ALGOritmic Language - alqoritmik dil) dili yaradılmışdı. Fortran, Alqol-60 dilləri əsasən elmi texniki məsələlərin həlli üçün nəzərdə tutulmuşdu və bu dillər heç də bütün mümkün məsələlər üçün yaramırdı. Buna görə də intensiv inkişaf edən elm və texnikanın yeni sahələrinin tələbatını ödəmək üçün yeni proqramlaşdırma dilləri yaradılır. Məsələn, iqtisadi məsələlərin həlli üçün 1959-cu ildə IBM firması tərəfindən Cobol (Common Business Oriented Language) dili yaradılır. Simvol informasiyanın emalını təmin edən dillərdən biri, Massaçuset texnoloji institutunda 1960-cı ildə yaradılmış Lisp dilidir. Digər dil Snobol - isə təbii dildə yazılmış mətnlərin maşın analizi üçün tətbiq olunur.

Üçüncü nəsl EHM-in yaranması, universal alqoritmik dillərin yaradılması məsələsini qarşıya qoydu. Bu cür dillərin yaradılması üçün edilən cəhdlərdən biri nəticəsində IBM firması tərəfindən PL/1 (Programming Language/1-proqramlaşdırma dili-1) dili yaradılır. Bu dil Fortran, Alqol və Cobol dillərinin əsasında yaradılmış və bu dillərin üstünlüklərini özündə birləşdirmişdi.

1971-ci ildə isə dünyada ilk dəfə olaraq ədədləri cəmləməyə imkan verən avtomatik qurğu yaratmış XVII əsr böyük fransız alimi Paskalın şərəfinə adlandırılmış Paskal dili yaradılır. Bu dil Alqol və PL/1 dillərinin varisidir. Paskal dili də Beysik dili kimi çox sadədir, lakin Paskal dili müasir proqramlaşdırma texnologiyasının geniş tətbiqini təmin edir. Bu dil struktur proqramlaşdırma ideyasının, yəni proqramın kiçik, dəqiq təyin edilmiş prosedurlardan tədricən qurulması ideyasının həyata keçirilməsinə imkan yaradır. 70-ci illərin sonunda Paskal dilinin əsasında Ada dili yaradılır. Bu dil çox böyük tətbiq sahələrinə malikdir. Dil birinci proqramçı qadın Ada Lavlaysın şərəfinə adlandırılmışdır. 80-cı illərin əvvəllərində C/C++ proqramlaşdırma dili (dilin adı bir C latın hərfindən ibarətdir) yaradılmışdır. C/C++ dili - universal dil olub, geniş yayılmış UNIX əməliyyat sistemi ilə sıx bağlıdır, belə ki, UNIX sistemi və onun proqram təminatı C/C++ dilində yazılıb. C/C++ dili, müasir kompüterlərin imkanlarından tam istifadə etməyə imkan verən dildir. Qeyd etdiyimiz dillərdən başqa digər dillər də mövcuddur və dillərin yaradılması prosesi davam etdirilir.

V FƏSİL

TURBO PASCAL ALQORİTMİK DİLİ.

5.1. Dilin əlifbası. Verilənlər. Proqramın strukturu

Dilin əlifbası hərf, rəqəm və xüsusi simvollardan ibarətdir. Hərflər – latın əlifbasının böyük (A-Z) və kiçik (a-z) hərfləri; rəqəmlər – on ərəb (0-9) rəqəmləri və 0,1,...,9,A,B,C,D,E,F onaltılıq say sisteminin rəqəmləri; xüsusi simvollar **+ - * / = > < . , ' : ; [ ] ( ) { } ^ @ $ #**.

Xüsusi simvollara eləcə də aşağıdakı simvol cütləri də (onları probellə ayırmaq olmaz) aiddir:

:= (mənimsətmə işarəsi), **>=** (böyük bərabər), **<>** (fərqli), **<=** (kiçik bərabər), **(* *)** **({}** işarəsi ilə eynigüclü olan qeyd məhdudlaşdırıcısı) **(. .)** **([ ]** işarəsinin ekvivalenti). Burada probel, boş yer işarəsi də xüsusi yer tutur. Bu simvol identifikator, sabit ədəd, işçi sözlər üçün məhdudlaşdırıcı kimi nəzərdən keçirilir. Bir-birinin ardınca verilən bir neçə probel işarəsi bir işarə kimi qəbul edilir (sətir sabitləri istisna olmaqla).

Turbo Pascal dilində aşağıdakı işçi sözlərdən istifadə olunur: **and, asm, array, begin, case, const, constructor, destructor, div, do, downto, else, end, file, for, function, goto, if, implementation, in, inline, interface, label, mod, nil, not, object, of, or, packed, procedure, program, record, repeat, set, shl, shr, string, then, to, type, unit, until, uses, var, while, with, xor.**

İşçi sözlərdən başqa məqsədlər üçün istifadə edilə bilməz. Dil nöqteyi-nəzərdən onlar vahid simvol hesab edilirlər. Dildə standart elanlar kimi aşağıdakı işçi sözlərdən də istifadə edilir: **absolute, assembler, external, far, forward, interrupt, near, private, virtual.**

Turbo Pascal dilində identifikator – sabit, dəyişən, nişan, tip, obyekt, prosedur, funksiya, modul, proqram və yazılışlardakı sahə adlarıdır. İdentifikator – hərflə başlayan ixtiyari hərflər və rəqəmlər ardıcıllığıdır. Turbo Pascal-da altdan xətt çəkmə («_»)

işarəsi də hərflərə aiddir. İdentifikator ixtiyari uzunluqlu ola bilər, lakin burada yalnız birinci 63 simvol nəzərə alınacaqdır. İdentifikatorda probeldən və dilin xüsusi simvollarından istifadə edilə bilməz. *Məsələn*: **x, y, ALPHA, _beta, _1, z12, max, MIN** və s.

Sabitlər kimi Turbo Pascal-da tam, həqiqi, onaltılıq ədədlər, məntiqi sabitlər, simvollar, sətirlər, çoxluq konstruktorları və qeyri-müəyyən göstərici əlaməti – NIL istifadə edilə bilər. Tam ədədlər adi qayda ilə işarə və ya işarəsiz yazılır və -2147483648-dən +2147483647-dək qiymətlər ala bilər. Həqiqi ədədlər işarə və ya işarəsiz onluq nöqtə ilə və ya eksponensial hissə ilə yazılırlar. Eksponensial hissə **e** (**E**) simvolu ilə başlayır, ondan sonra «+» və ya «-» işarəsi gələ bilər və onluq tərtib verilir. Burada **e** (**E**) simvolu onluq tərtib deməkdir. Məsələn, **3.5E5** (yəni $3{,}5*10^5$), **-17e-3** (yəni $-17*10^{-3}$) və s. Onaltılıq ədədlər, qarşısında $ işarəsi olan onaltılıq say sisteminin ədədləridir. Onların dəyişmə diapazonu $00000000-dan $FFFFFFFF-ə qədərdir. Məntiqi sabit – **FALSE** (yalan) və ya **TRUE** (doğru) sözlərindən biridir. Simvol sabitlər klaviaturanın apastrof işarələri arasına alınmış ixtiyari simvollarıdır. Məsələn, 'z', 'A', '9', və s. Sətir sabiti apastrof işarələri arasına alınan ixtiyari simvollar ardıcıllığıdır. Məsələn, 'Turbo Pascal alqoritmik dili'. Sətirdə heç bir simvol verilməzsə, belə sətir boş sətir adlanır. Çoxluq konstruktoru – kvadrat mötərizə daxilində verilən çoxluq elementlərinin siyahısıdır. Məsələn, **[1,2,3..8,12], [true], [ ], [blue,red]** və s. Qeyd edək ki, standart Pascal dilindən fərqli olaraq Turbo Pascal dilində sabitlər kimi elementləri əvvəlcədən elan edilmiş sabitlər, tip adları, obyektlər və funksiyalar olan ixtiyari ifadələrdən də istifadə edilə bilər.

Proqram vahidi başlıqdan, təsvirlər bölməsindən, operatorlar bölməsindən və proqramın sonunu bildirən nöqtədən ibarətdir:

program <proqramın adı> – başlıq
uses – modullar bölməsi
label – nişanlar bölməsi
const – sabitlər bölməsi
type – tiplər bölməsi
var – dəyişənlər bölməsi

procedure (function) – alt proqramlar bölməsi
begin
<operatorlar bölməsi>
end.

Başlıqda proqram vahidinə verilən ad yerləşdirilir, ad üzərinə identifikatorlar üçün təyin edilmiş şərtlər qoyulur. Turbo Pascal dilində başlıq verilməyə də bilər. Ümumiyyətlə, proqramın bölmələrində heç bir təsvir və yerinə yetirilən operatorlar verilməyə də bilər. Standart Pascal dilindən fərqli olaraq Turbo Pascal dilində **label**, **const**, **type**, **var** bölmələri bir-birinin ardınca ixtiyari qaydada verilə bilər və təsvirlər bölməsində ixtiyari sayda ola bilər. Təsvirlər bölməsində operatorlar bölməsində istifadə edilən bütün identifikatorlar təsvir edilməlidir. Burada modulların interfeys hissələrində təyin edilən və posedurların (funksiyaların) qlobal identifikatorları istisna təşkil edir. Əgər proqram vahidində hər hansı bir modulun identifikatorundan istifadə edilirsə, onda **uses** bölməsində bu modulun adı elan edilməlidir. Burada da **system** modulu istisna təşkil edir. Çünki bu modul əvvəlcədən elan edilmiş hesab edilir. Təsvirlər bölməsində tiplərin, obyektlərin, sabitlərin, dəyişənlərin identifikatorları, həmçinin nişanlar, prosedur və funksiyalar elan edilir. Tip və obyektlərin təsviri – **type** bölməsində, sabitlərin təsviri – **const**, dəyişənlərin təsviri – **var**, nişanların təsviri isə **label** bölməsində yerinə yetirilir. *Məsələn*,

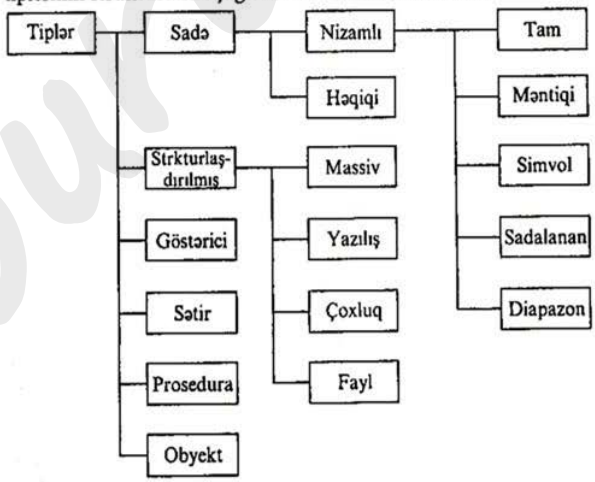
```
type b=array[1..10] of real;
     a=set of '0'..'9';
     c=string[100];
const n=100; eps=1e-3;
var x,y:real; s1:c; k:b;
label 1,2,lb1,lb2;
```

Turbo Pascal dilində proqram vahidində gedən prosesləri izah etmək üçün şərhlərdən geniş istifadə edilir. Burada şərh figurlu mötərizələr daxilində gətirilən ixtiyari simvolların istənilən ardıcıllığıdır. Şərhlər proqramın ixtiyari yerində verilə bilər və proqramın yerinə yetirilməsinə heç bir təsir göstərmir. Şərh məhdudlaşdırıcısı kimi figurlu mötərizələrlə yanaşı (* şərh *) işarəsindən də istifadə etmək olar. Məsələn, {Daxil etmək}, (* Cavab*)

və s. Qeyd edək ki, proqram mətnində hər sətrin sonunda qoyulan nöqtə vergül işarəsi sətrin sonunu bildirir.

5.2. Verilənlərin tipləri. Tiplərin uyğunluğu və çevrilməsi

İxtiyari verilənlər, yəni sabitlər, dəyişənlər, funksiya qiymətləri və ya ifadələr, Turbo Pascal-da öz tipləri ilə xarakterizə olunurlar. Tip bu və ya digər obyektin ala biləcəyi qiymətlər çoxluğunu və bu obyektlərə tətbiq oluna biləcək əməliyyatlar çoxluğunu təyin edir. Bundan əlavə tip, verilənlərin EHM yaddaşında daxili ifadə formatını müəyyən edir. Turbo Pascal-da verilənlərin tiplərinin strukturunu aşağıdakı kimi ifadə etmək olar:



Turbo Pascal-da verilənlərin yeni tiplərinin yaradılması mexanizmi nəzərdə tutulduğundan, proqramda istifadə edilən tiplərin sayı ixtiyari qədər böyük ola bilər.

Sadə tiplərə nizamlı və həqiqi tiplər aiddir. Nizamlı tiplərin hər biri sonlu sayda mümkün qiymətə malikdir. Bu qiymətləri müəyyən qaydada nizamlamaq olur (tipin adı da buradan irəli gəlir) və deməli, onların hər birinə qarşı hər hansı bir tam ədəd-qiymətin sıra nömrəsini qoymaq olur. Həqiqi tiplər də həqiqi ədə-

din daxili ifadə formatı ilə müəyyən edilən sonlu sayda qiymətə malikdir. Lakin həqiqi tiplərin ala biləcəyi qiymətlərin sayı o qədər böyükdür ki, onların hər birinə qarşı tam ədəd (onun nömrəsini) qoymaq mümkün deyil.

Nizam tipinə tam, məntiqi, simvol, sadalanan və diapazon tipləri aiddir. Bütün bu tiplərə **ord(x)** funksiyası tətbiq edilə bilər, həmin funksiya **x** ifadəsinin qiymətinin sıra nömrəsini təyin edir. Tam tiplər üçün bu funksiya **x**-ın aldığı qiyməti verir, yəni **ord(x)=x**, burada **x**-ixtiyari tam tipə aiddir. Bu funksiya məntiqi, simvol və sadalanan tiplərə tətbiq olunduqda, məntiqi tip üçün 0-1, simvol tip üçün 0-255, sadalanan tip üçün 0-65535 diapazonunda müsbət tam ədəd alınır. Nizamlı tiplərinə həmçinin **pred(x)** – nizam tipinin əvvəlki qiymətini (**ord(x)-1** sıra nömrəsinə uyğun) təyin edən, yəni **ord(pred(x))=ord(x)-1** funksiyasını və succ(x) – nizam tipinin sonrakı qiymətini (**ord(x)+1** sıra nömrəsinə uyğun) təyin edən, yəni **ord(succ(x))=ord(x)+1** funksiyasını tətbiq etmək olar.

Tam tiplərə aşağıdakılar aiddir:

Adı	*Baytlarla uzunluğu*	*Qiymətlər diapazonu*
Byte	1	0...255
ShortInt	1	-128 ... +127
Word	2	0 ... 65535
Integer	2	-32768 ... +32767
LongInt	4	-2147483648 ... +2147483647

Tam tiplərə tətbiq olunan prosedur və funksiyalar aşağıdakılardır:

Funksiya	*Təyinatı*	*Arqumentin tipi*	*Nəticənin tipi*
abs(x)	**x**-in mütləq qiymətinin təyini	İxtiyari tam tip	Arqumentin tipi
chr(x)	Simvolu onun **x** kodu üzrə təyin edir	**Byte**	**Char**
dec(x[,i])	**x**-in qiymətini **i** qədər, **i** verilmədikdə bir vahid azaldır	İxtiyari tam tip	Arqumentin tipi

inc(x[,i])	**x**-in qiymətini **i** qədər, **i** verilmədikdə bir vahid artırır	İxtiyari tam tip	Arqumentin tipi
hi(x)	**x**-in yüksək baytını təyin edir	**Integer Word**	**Byte**
lo(x)	**x**-in aşağı baytını təyin edir	**Integer Word**	**Byte**
odd(x)	**x**-in tək ədəd olduqda **True** qiymətini alır	**LongInt**	**Boolean**
random(x)	**0 ... x-1** diapazonunda bərabər paylanmış təsadüfi ədədi təyin edir	**Word**	**Word**
sqr(x)	**x**-in kvadratını təyin edir	İxtiyari tam tip	Arqumentin tipi
swap(x)	Sözdə yüksək və aşağı baytların yerini dəyişdirir	**Integer Word**	**Integer Word**
randomize	Təsadüfi ədədlər generatorunun aktivləşdirilməsi	–	–

Qeyd edək ki, tam tipli parametrləri olan prosedur və funksiyalardan istifadə edərkən nəzərə almaq lazımdır ki, əgər **Word** tipindən istifadə edilirsə, burada **Byte** tipindən də istifadə oluna bilər (əksinə yox), eləcə də **Integer** tipi **LongInt**-ə daxildir, **ShortInt** tipi isə **Integer**-ə daxildir. Tam ədədlərlə aparılan əməliyyatların nəticəsi əməliyyatda iştirak edən verilənlərin tipi ilə eyni olur. Onların tipi müxtəlif olduqda isə nəticənin tipi buradakı ən yüksək qiymətlər diapazonu olan tiplə üst-üstə düşür. Turbo Pascal-da əvvəlcədən elan edilmiş **Integer** tipli sabit **MaxInt** 32767 qiymətini alır. Məntiqi tipin qiyməti əvvəlcədən elan edilmiş **False** (yalan) və ya **True** (doğru) sabitlərindən hər hansı biri ola bilər. Onlar üçün aşağıdakı qaydalar doğrudur:

```
ord(False)=0;
ord(True)=1;
False<True;
succ(False)=True;
pred(True)=False.
```

Simvol tipin qiymətləri EHM-dəki bütün simvollar çoxluğudur. Hər bir simvola 0 ... 255 diapazonunda bir tam ədəd uyğun gəlir. Bu ədəd simvolun daxili ifadə kodudur və **ord** funksiyası ilə təyin edilə bilir. Kodlaşdırma üçün ASCII (American Standard Code for Information Interchange – informasiya mübadiləsi üçün Amerika standart kodu) kodundan istifadə edilir. **Char** tipinə münasibət əməliyyatları və aşağıdakı funksiyalar tətbiq edilə bilir:

chr(x) – **char** tipli funksiya olaraq byte tipli **x** ifadəsini simvola çevirir.

upcase(x)-də **char** tipli funksiya olub, char tipli **x** arqumentini kiçik latın hərfi olduqda onu uyğun böyük latın hərfinə çevirir, əks halda isə **x** simvolunun özünü qaytarır.

Sadalanan tip, onun ala biləcəyi qiymətlərin sadalanması ilə verilir. Hər bir qiymət müəyyən identifikatorla adlandırılıb, yumru mötərizələrlə məhdudlaşdırılan siyahıda verilir. *Məsələn*,

```
type c=(qirmizi,sari,qara,boz);
var ay:(yan,fev,mart,apr,may,iyun,iyul,
        avq,sen,okt,noy,dek);
```

Sadalanan tipdə siyahıdakı birinci elementin sıra nömrəsi sıfır, ikinci elementin nömrəsi bir və s. olur. Sadalanan tipin maksimal gücü 65536 qiymətdir. *Məsələn*,

```
type gaz=(c,o,n,f);
     metal=(fe,co,na,cu);
var g1,g2,g3:gaz; m1,m2,m3:metal;
```

Diapazon tipi özünün baza tipinin alt çoxluğudur. Baza tipi diapazon tipdən başqa ixtiyari nizam tipi ola bilər. Diapazon tipi baza tipi daxilində öz qiymətlərinin sərhədləri ilə verilir: <min. qiymət> .. <maks. qiymət>. Burada <min. qiymət> - diapazon tipinin minimal qiyməti, <maks. qiymət> isə maksimal qiymətidir. *Məsələn*,

```
type k1='0'..'9';k2=1966..2007;
```


və ya

```
var date:1..31; month:1..12;
    k3:'A'..'Z';
```

və s.

Burada «..» simvolu bir simvol kimi qəbul edildiyindən nöqtələr arasında probel qoyula bilməz və diapazonun sol sərhəddi sağ sərhəddini aşa bilməz. Diapazon tiplərlə iş üçün aşağıdakı funksiyalar nəzərdə tutulub:

high(x) – funksiyası **x**-in aid olduğu diapazon tipinin maksimal qiymətini verir,

low(x) - funksiyası **x**-in aid olduğu diapazon tipinin minimal qiymətini verir. *Məsələn*,

```
var k:integer;
begin
     writeln(low(k),'...',high(k))
end.
```

Nəticədə alarıq: -32768 ... 32767.

Həqiqi tiplərə aşağıdakılar aiddir:

Adı	*Baytlarla uzunluğu*	*Əhəmiyyətli rəqəmlərin sayı*	*Qiymətlər diapazonu*
Real	6	11 – 12	$2.9\cdot10^{-39}$ ... $1.7\cdot10^{38}$
Single	4	7 – 8	$1.5\cdot10^{-45}$ ... $3.4\cdot10^{38}$
Double	8	15 – 16	$5\cdot10^{-324}$ ... $1.7\cdot10^{308}$
Extended	10	19 – 20	$3.4\cdot10^{-4932}$ ... $1.1\cdot10^{4932}$
Comp	8	19 – 20	$-2\cdot10^{63}+1$ ... $+2\cdot10^{63}-1$

EHM-də mütləq dəqiqliklə ifadə olunan nizam tiplərindən fərqli olaraq həqiqi tiplərin qiyməti ixtiyari ədədi yalnız müəyyən sonlu dəqiqliklə təyin edir. Burada hər şey həqiqi ədədin daxili ifadə formatından asılı olur. Yuxarıda verdiyimiz birinci beş həqiqi tip bir-birindən qiymətlər diapazonu və dəqiqliyinə görə fərqlənir. Kəsr və eksponensial hissəsi olmayan **Comp** tipi faktiki olaraq 19-20 əhəmiyyətli onluq rəqəm saxlayan, işarəli böyük tam ədəddir. Lakin eyni zamanda ifadələrdə **Comp** tipi digər ixtiyari

həqiqi tiplərlə uyğunlaşır və onun üzərində həqiqi tiplərə uyğun bütün əməliyyatlar aparıla bilər. **Comp** tipinin ən əlverişli tətbiq sahəsi mühasibat işləridir, belə ki, burada pul kütləsi qəpik və ya sentlərlə ifadə olunur və onlar üzərindəki əməliyyatlar kifayət qədər uzun tam ədədlərə gətirib çıxarır.

Həqiqi tiplərə tətbiq olunan funksiyalar aşağıdakılardır:

Funksiya	*Təyinatı*	*Arqumentin tipi*	*Nəticənin tipi*
abs(x)	**x**-in modulunu təyin edir	İxtiyari həqiqi tip	Arqumentin tipi
arctan(x)	**x**-in arktangensini (radianlarla) təyin edir	İxtiyari həqiqi tip	Arqumentin tipi
cos(x)	**x**-in kosinusunu (radianlarla) təyin edir	İxtiyari həqiqi tip	Arqumentin tipi
exp(x)	**x**-in eksponentasını təyin edir	İxtiyari həqiqi tip	Arqumentin tipi
frac(x)	**x**-in kəsr hissəsini təyin edir	İxtiyari həqiqi tip	Arqumentin tipi
int(x)	**x**-in tam hissəsini təyin edir	İxtiyari həqiqi tip	Arqumentin tipi
ln(x)	**x**-in natural loqarifmini təyin edir	İxtiyari həqiqi tip	Arqumentin tipi
pi	$\pi = 3.141592\ ...$ ədədini verir	–	İxtiyari həqiqi tip
random	[0,1) intervalında təsadüfi ədəd təyin edir	–	İxtiyari həqiqi tip
sin(x)	**x**-in sinusunu (radianlarla) təyin edir	İxtiyari həqiqi tip	Arqumentin tipi
sqr(x)	**x**-in kvadratını təyin edir	İxtiyari həqiqi tip	Arqumentin tipi
sqrt(x)	**x**-dən kvadrat kök alır	İxtiyari həqiqi tip	Arqumentin tipi

trunc(x)	**x**-i modulca aşmayan ən yaxın tam ədədi verir	İxtiyari həqiqi tip	**Integer**
round(x)	**x**-i ən yaxın tam ədədə qədər yuvarlaqlaşdırır.	İxtiyari həqiqi tip	**Integer**

İki tip öz arasında aşağıdakı şərtlər daxilində uyğun hesab edilir:

1) hər ikisi eyni tipə aiddirsə,
2) hər ikisi həqiqi tiplidirsə,
3) hər ikisi tam tiplidirsə,
4) bir tip ikinci tipin diapazon tipidirsə,
5) hər ikisi eyni bir baza tipinin diapazon tipidirsə,
6) hər ikisi eyni bir baza tipinin elementlərindən qurulmuş çoxluqlardırsa,
7) hər ikisi eyni maksimal uzunluqlu (**packed** sözü ilə təyin edilmiş) sətirlərdirsə,
8) tipin biri sətir tip, digəri isə sətir tip, sətir və ya simvoldursa,
9) bir tip ixtiyari göstərici, digəri isə qeyri-tip göstəricidirsə,
10) bir tip obyektə göstərci, digəri isə həmin obyektə qohum olan obyektə göstəricidirsə,
11) hər ikisi eyni tip nəticə alan prosedur tiplərdirsə.

Mənimsətmə operatorunda **t1:=t2;** (**t1** – dəyişənin tipi, **t2** – ifadənin tipi) mənimsətmə aşağıdakı hallarda mümkündür:

1) **t1** və **t2** eyni tiplidirsə və bu tip fayl və fayllar massivinə aid deyilsə və ya fayl-sahələri olan yazılışlara və ya bu yazılışların massivlərinə aid deyilsə,
2) **t1** və **t2** uyğunlaşmış nizam tiplərinə aiddirsə və **t2**-nin qiyməti, **t1**-in mümkün qiymətləri diapazonuna daxildirsə,
3) **t1** və **t2** həqiqi tiplidirsə və **t2**-nin qiyməti **t1**-in mümkün qiymətləri diapazonuna daxildirsə,
4) **t1** – həqiqi tip, **t2** isə tam tiplidirsə,
5) **t1** – sətir, **t2** isə simvoldursa,
6) **t1**-sətir, **t2** isə sıxılmış sətirdirsə,
7) **t1** və **t2** uyğunlaşdırılmış sıxılmış sətirlərdirsə,
8) **t1** və **t2** uyğunlaşdırılmış çoxluqlardırsa və **t2**-nin

bütün hədləri **t1**-in mümkün qiymətləri çoxluğuna daxildirsə,

9) **t1** və **t2** uyğunlaşdırılmış göstəricidirlərsə,

10) **t1** və **t2** uyğunlaşdırılmış prosedur tipləridirsə,

11) **t1** – obyekt, **t2** isə onun nəslindəndirsə.

Proqramda bir tip verilənlər digər tip verilənlərə çevrilə bilər. Bu cür çevirmələr aşkar və ya qeyri-aşkar ola bilər. Tiplərin aşkar çevirmələri zamanı arqumentləri bir tipə, qiymətləri isə başqa tipə aid olan xüsusi funksiyalardan (**ord, trunc, round, chr, ptr**) istifadə olunur. Turbo Pascal-da tiplərin çevrilməsinin daha ümumi mexanizmindən də istifadə edilə bilər. Burada çevirmə standart tip identifikatorunun və ya istifadəçi tərəfindən çevrilən tip ifadəyə çevirmə funksiyasının identifikatoru kimi təyin edilən identifikatorun tətbiqi ilə həyata keçirilir. Turbo Pascal-da tiplərin çevrilməsi üçün daha bir aşkar üsul təyin olunub: müəyyən tip dəyişənin yaddaşda tutduğu sahəyə, onun yaddaşda daxili ifadə uzunluğuna bərabər uzunluqlu digər tip dəyişəni yerləşdirmək olar.

Tiplərin qeyri-aşkar çevirməsi yalnız aşağıdakı iki halda mümkündür:

1) həqiqi və tam tipli dəyişənlərdən təşkil olunmuş ifadələrdə tam tipli dəyişənlər avtomatik olaraq həqiqi tipə çevrilir və ifadə həqiqi qiymət alır.

2) yaddaşın eyni bir sahəsində növbə ilə gah bir, gah da digər tip verilənlərin yerləşdiyi elan edilir.

5.3. Əməllər. İfadələr

Turbo Pascal dilində aşağıdakı əməliyyatlar təyin edilib:

unar əməliyyatlar: **not, @**;

multiplikativ əməliyyatlar: ***, /, div, mod, and, shl, shr**;

additiv əməliyyatlar: **+, –, or, xor**;

münasibət əməliyyatları: **=, <>, <, >, <=, >=, in**.

Bu əməliyyatların yerinə yetirilmə üstünlüyü verdiyimiz ardıcıllığa uyğundur, yəni ifadələrdə birinci növbədə unar, axırıncı növbədə isə münasibət əməliyyatları yerinə yetiriləcəkdir. Əməliyyat iştirakçılarını operandlar adlandıracağıq. Müxtəlif tip operandlarla əməliyyat qaydaları aşağıdakı cədvəldə verilir:

İşarə	*Əməliyyat*	*İfadə*	*Operandların tipləri*	*Nəticənin tipi*
not	İnkar	**not A**	Məntiqi	Məntiqi
not	İnkar	**not A**	İxtiyari tam	Operandın tipi
@	Ünvan	**–**	İxtiyari tam	Göstərici
*****	Vurma	**A*B**	İxtiyari tam	Ən kiçik tam tip
*****	Vurma	**A*B**	İxtiyari tam	**Extended**
*****	Çoxluqların kəsişməsi	**A*B**	Çoxluq	Çoxluq
/	Bölmə	**A/B**	İxtiyari həqiqi	**Extended**
div	Tam bölmə	**A div B**	İxtiyari tam	Ən kiçik tam tip
mod	Bölmə qalığı	**A mod B**	İxtiyari tam	Ən kiçik tam tip
and	Məntiqi vurma	**A and B**	Məntiqi	Məntiqi
and	Məntiqi vurma	**A and B**	İxtiyari tam	Ən kiçik tam tip
shl	Sola sürüşmə	**A shl B**	İxtiyari tam	Ən kiçik tam tip
shr	Sağa sürüşmə	**A shr B**	İxtiyari tam	Ən kiçik tam tip
+	Toplama	**A+B**	İxtiyari tam	Ən kiçik tam tip
+	Toplama	**A+B**	İxtiyari həqiqi	**Extended**
+	Çoxluqların birləşdirilməsi	**A+B**	Çoxluq	Çoxluq
+	Sətirlərin birləşdirilməsi	**A+B**	Sətir	Sətir
–	Çıxma	**A–B**	İxtiyari tam	Ən kiçik tam tip
–	Çıxma	**A–B**	İxtiyari həqiqi	**Extended**

or	Məntiqi toplama	A or B	Məntiqi	Məntiqi
or	Məntiqi toplama	A or B	İxtiyari tam	Ən kiçik tam tip
=	Bərabər	A=B	İxtiyari sadə və ya sətir	Məntiqi
<>	Fərqli	A<>B	İxtiyari sadə və ya sətir	Məntiqi
<	Kiçik	A<B	Məntiqi	Məntiqi
<=	Kiçik və ya bərabər	A<=B	Məntiqi	Məntiqi
>	Böyük	A<B	Məntiqi	Məntiqi
>=	Böyük və ya bərabər	A>=B	Məntiqi	Məntiqi

Unar @ əməliyyatı ixtiyari tip operanda tətbiq edilə bilir və operandın ünvanı olan **pointer** tipli nəticə verir. Əgər bu əməliyyat prosedur, funksiya və ya obyektə tətbiq olunursa, onun nəticəsi bu proseduraya (funksiyaya, obyektə) giriş nöqtəsinin ünvanı olacaqdır.

Turbo Pascal dilində aşağıdakı məntiqi əməliyyatlar təyin edilib:

not – məntiqi inkar;
and – məntiqi vurma (konyunksiya);
or – məntiqi toplama (dizyunksiya);
xor – məntiqi toplamanın inkarı.

Məntiqi əməliyyatlar tam və məntiqi tipli operandlara tətbiq edilə bilir. Əgər operandlar tam tiplidirsə, məntiqi əməliyyatın nəticəsi də tam ədəd olacaqdır. Məntiqi tipli verilənlər üzərindəki məntiqi əməliyyatın nəticəsi məntiqi tip olacaqdır. **Integer** tipli verilənlər üzərindəki məntiqi əməliyyatların nəticələri aşağıdakı cədvəllə verilir:

Operand1	Operand2	not	and	or	xor
1	–	0	–	–	–
0	–	1	–	–	–
0	0	–	0	0	0
0	1	–	0	1	1

1	0	–	0	1	1
1	1	–	1	1	0

Boolean tipli verilənlər üzərindəki məntiqi əməliyyatların nəticələri aşağıdakı cədvəllə verilir:

Operand1	Operand2	not	and	or	xor
True	–	False	–	–	–
False	–	True	–	–	–
False	False	–	False	False	False
False	True	–	False	True	True
True	False	–	False	True	True
True	True	–	True	True	False

Turbo Pascal dilində məntiqi tipə, tam ədədlər üzərində aparılan aşağıdakı iki sürüşdürmə əməliyyatı da aid edilir: **i shl j** – əməliyyatı nəticəsində **i** – nin tərkibi j sayda mərtəbə sola sürüşdürülür, bu zaman boşalan kiçik mərtəbələr sıfırlarla doldurulur; **i shr j** – əməliyyatı nəticəsində isə **i**-nin tərkibi j sayda mərtəbə sağa sürüşdürülür, bu zaman boşalan yüksək mərtəbələr sıfırlarla doldurulur. Bu əməliyyatlarda **i** və **j** ixtiyari tam tipli ifadələrdir.

In münasibət əməliyyatı iki operanda tətbiq edilir. Sol operand ixtiyari nizam tipli, ikinci operand isə həmin tip elementlərdən ibarət çoxluq və ya çoxluq tipli identifikator olmalıdır.

Proqramın yerinə yetirilən hissəsinin qurulduğu əsas elementlər, sabitlər, dəyişənlər və funksiyalara münasibətlərdir. Bu elementlərin hər biri öz qiyməti ilə xarakterizə edilir və verilənlərin hansısa bir tipinə aid olur. Əməliyyat işarələri və mötərizələrin köməyi ilə onlardan ifadələr təşkil etmək mümkün olur. İfadələr isə faktiki olaraq, yeni qiymətlərin alınması qaydalarıdır. İfadənin xüsusi halı sadəcə bir element, yəni sabit, dəyişən və ya funksiyaya müraciət ola bilər. Bu cür ifadənin qiyməti, aşkardır ki, elementlə eyni bir tipə aid olacaqdır. Daha ümumi halda ifadə bir neçə elementdən (operanddan) və əməliyyat işarələrindən ibarətdir. İfadənin qiymətinin tipi, operandların tipi və onlara tətbiq olunmuş əməliyyatların tipi ilə müəyyən olunur. Hesabi ifadələr ədədi kəmiyyətlər üzərindəki əməllərin yerinə yetirilmə ardıcıllığını müəyyən edir. Bu ifadələr hesabi əməliyyatlardan,

funksiyalara müraciətlərdən, operandlardan (sabitlər, dəyişənlər) və yumru mötərizələrdən ibarət olur. *Məsələn,*

$(2*a+sqrt(2*\sin(x+y)))/(0.2*c-\ln(x-y))$.

Bu ifadələrdə, yerinə yetirilmə üstünlüyu daha yüksək olan əməliyyatlar əvvəl yerinə yetirilir. Burada yerinə yetirilmə üstünlüyü azalma sırası ilə aşağıdakı kimidir:

1) Funksiyaların hesablanması;
2) (-) işarənin dəyişdirilməsinin unar əməliyyatı;
3) *****, **/**, **div**, **mod**;
4) **+**, **-**.

Bir-birinin ardınca gələn eyni hüquqlu əməliyyatlar, ifadədə soldan sağa doğru yerinə yetirilir. İfadədə mötərizə daxilindəki əməliyyatlar, yernə yetirilmə üstünlüyündən asılı olmayaraq birinci növbədə hesablanırlar. Riyazi mənası olmayan ifadələr, məsələn, sıfıra bölmə, mənfi ədədin loqarifmi və s. kimi ifadələr yazmaq olmaz. Məsələn, aşağıdakı ifadə üzərindəki rəqəmlər əməliyyatların yerinə yetirilmə ardıcıllığını bildirir:

$$\overset{1}{(}1+y\overset{7}{)}*\overset{4}{(}2*x+\overset{5}{sqrt}\overset{3}{(}y\overset{6}{)}-\overset{2}{(}x+y\overset{12}{)}\overset{11}{)}/\overset{10}{(}y+1/\overset{8}{(}sqr\overset{9}{(}x\overset{}{)}-4\overset{}{)}\overset{}{)}$$

Turbo Pascal dilində ədədin ixtiyari tərtibindən qüvvətə yüksəltmə əməli və ya standart funksiyası nəzərdə tutulmayıb. x^y-in hesablanması üçün əgər y tam ədəddirsə, qüvvət vurma əməli ilə təyin edilir, məsələn, $x^3 \to sqr(x)*x$ daha böyük qüvvətlər isə dövr daxilində vurma ilə tapılır. Burada **y** – həqiqi ədəd olduqda isə $x^y = \exp(y \cdot \ln(x))$ riyazi düsturundan istifadə edilir, Pascal dilində bu $\exp(y*\ln(x))$ şəklində ifadə edilir.

Məntiqi ifadələr məntiqi əməliyyatlar və yumru mötərizələrlə əlaqələndirilən məntiqi operandlardan ibarətdir. Məntiqi ifadənin yerinə yetirilmə nəticəsi **false** və ya **true** məntiqi sabitidir. Məntiqi operandlar, məntiqi sabitlər, dəyişənlər, funksiyaya müraciətlər, münasibət əməliyyatları ola bilər. *Məsələn,*

1) **x<2*y**; 2) **true**; 3) **d**; 4) **odd(k)**; 5) **not not d**; 6) **not(x>y/2)**;

7) **d and (x<>y) and b**; 8) **(c or d) and (x=y) or not b**.

Burada **d=true**; **b=false**; **c=true**; **x=3.0**; **y=0.5**; **k=5** olarsa, nəticədə alarıq:

1) **false**; 2) **true**; 3) **true**; 4) **true**; 5) **true**;
6) **false**; 7) **false**; 8) **true**.

5.4. Mənimsətmə operatoru, qurma operator və boş operator

Turbo Pascal-ın əsas operatorlarından biri mənimsətmə operatorudur. Operatorun sol tərəfində dəyişən adı verilir, sağ tərəf isə dəyişən adı ilə eyni tipli olan ifadədən ibarətdir. Operatorun sol və sağ tərəfləri mənimsətmə işarəsi adlanan «**:=**» simvollar cütü ilə əlaqələndirilir. *Məsələn*:

```
x:=x+1; A:=5; z:=-637.225;
D:=(x>y) and (k<>0); k:='PASCAL'.
```

Qeyd edək ki, mənimsətmə operatorunda həmişə «**:=**» simvollar cütündən istifadə edilir, sabitlərin təsvirində isə «**=**» simvolu tətbiq edilir. Mənimsətmə operatorunda dəyişən adı ilə ifadə arasındakı tip uyğunluğunun mümkün variantları yuxarıda verilmişdir.

Qurma operator – **begin – end** operator mötərizəsi daxilinə alınmış proqramın ixtiyari operatorlar ardıcıllığıdır. Qurma operatorlar proqramları struktur proqramlaşdırmanın müasir texnologiyaları ilə hazırlımağa imkan verən vacib alətlərdən biridir. Turbo Pascal dili qurma operatora daxil olan operatorlara heç bir məhdudiyyətlər qoymur. Bu operatorların daxilində digər qurma operatorlar da ola bilər. Burada **begin – end** sözləri ilə məhdudlaşdırılmış operatorlar ardıcıllığı bir qurma operator deməkdir. Burada **end** sözü bağlayıcı operator mötərizəsi olduğundan, o həm də ondan əvvəlki operatorun sonunu bildirir, buna görə də ondan əvvəl «**;**» işarəsini qoymaq məcburi deyildir. **End** sözündən əvvəl «**;**» işarəsinin qoyulması, axırıncı operator və **end** sözləri arasında boş operatorun verildiyini bildirir. Boş operator heç bir əməliyyat yerinə yetirmir və onun daxil edilməsi üçün proqrama əlavə «**;**» işarəsini əlavə etmək kifayətdir. Boş operator əsasən idarəetməni qurma operatorun sonuna verilməsi üçün istifadə olunur.

5.5. Daxiletmə və xaricetmə operatorları

Daxiletmə operatoru (daxiletmə standart prorseduruna müraciət) aşağıdakı formadadır:

read (<daxiletmə siyahısı>) ;

burada <daxiletmə siyahısı> vergüllə bir-birindən ayrılan dəyişən adları ardıcıllığıdır. Məsələn, **read(a,b,c,d) ;**. Bu operatorun yerinə yetirilməsi zamanı proqramın işi dayandırılır və istifadəçi klaviaturada **a,b,c,d** dəyişənlərinin qiymətlərini, bir-birindən probellə ayırmaq şərtilə daxil etməlidir. Bu zaman daxil edilən qiymətlər ekranda görünür. Sonda *Enter* düyməsi sıxılır. *Məsələn*,

```
var t:real; i:integer; k:char;
begin read(t,i,k) end;
```

Klaviaturada məsələn, yığmaq olar: **123.41 10 G (Enter)**.

Əgər proqramda bir neçə **read** operatoru varsa, onda onlar üçün verilənlər bir-birinin ardınca daxil edilir. *Məsələn*,

```
var a,b:integer; c,d:real;
begin read(a,b);
      read(c,d);
end;
```

Onda verilənlər klaviaturadan aşağıdakı kimi daxil edilər:

187 34 (Enter) 2.17E-02 1.5E+01 (Enter)

Bu operatorun digər variantı aşağıdakı formaya malikdir:

readln (<daxiletmə siyahısı>) ;

Bu operatorun **read** operatorundan fərqi yalnız ondadır ki, bir **readln** operatorunun siyahısındakı axırıncı verilən daxil edildikdən sonra ondan sonra gələn digər daxiletmə operatorunun siyasındakı verilənlər yeni sətirdən başlayaraq daxil ediləcəkdir. Məsələn, yuxarıdakı misalda

```
readln(a,b);
readln(c,d);
```

əvəzləməsi aparsaq, klaviaturadan daxiletmə aşağıdakı şəkildə olacaqdır:

```
187  34 (Enter)
2.17E-02  1.5E+01 (Enter).
```


Qeyd edək ki, bu operatorun <daxiletmə siyahısı> boş da ola bilər. Bu operatorla klaviaturadan daxil edilən verilənlər arasında sadəcə boş sətir daxil etmək olur.

Xaricetmə operatoru (standart xaricetmə proseduruna müraciət) aşağıdakı formaya malikdir:

write (<xaricetmə siyahısı>)

Burada < xaricetmə siyahısı>nın elementləri müxtəlif tip ifadələr (xüsusi halda sabit və dəyişənlər) ola bilər. *Məsələn*,

write(132) ; write(a+b+2) ; write(x,y,z) ;

və s. Bu operatorla ekrana çıxarılan bir neçə ədəd bir-birindən probellə ayrılmırlar, buna görə də siyahıda bir neçə qiymət olduqda, bunu nəzərə almaq lazımdır. *Məsələn*,

```
write(a,' ',b,' ',c);
```

Nəticədə ekranda alarıq:

1 2 3

Burada sonuncu qiymət çıxarıldıqdan sonra kursor elə bu sətirdə qalır.

Operatorun digər variantı aşağıdakı formadadır:

writeln (< xaricetmə siyahısı >) ;

Bu operatorun **write** operatorundan fərqi yalnız ondadır ki, bu operatorun siyahısındakı sonunucu qiymət ekrana çıxarıldıqdan sonra kursor növbəti sətrə keçirilir və yeni qiymətlər bu sətirdən başlayaraq çıxarılır. Bu operatorun siyahısı boş olduqda, ekrana boş sətir çıxarılır.

Operatorun siyahısında çıxış formatını təyin edən göstəricilər verilə bilər. Format xaric edilən qiymətin ekrandakı ifadəsini təyin edir. Format aid olduğu elementdən «:» işarəsi ilə ayrılır. Əgər format verilməyibsə, onda kompüter qiyməti, susmaqla nəzərdə tutulmuş qaydada çıxarır.

Müxtəlif tip verilənlərin formatlı və formatsız xaricetmə qaydalarına baxaq. Siyahıdakı verilənləri göstərmək üçün aşağıdakı işarələmələrdən istifadə edək: **i,p,q** – tam tipli ifadələr, **r** – həqiqi tipli ifadə, **m** – məntiqi tipli ifadə, **c** – simvol tipli kəmiyyət, **s** – sətir tipli ifadə, **#** - ədəd, ***** - "+"və ya «-» işarələri, **_** isə probel işarəsidir.

1) Formatsız çıxarışla **i** kəmiyyətinin kursorun durduğu mövqedən başlayaraq onluq ifadəsi çıxarılır:

i:=134; write(i) ; nəticə: **134;**

```
i:=287; write (i,i,i); nəticə: 287287287;
```

2) Formatlı çıxarışla **i** kəmiyyətinin **i:p** formatı ilə onluq ifadəsi **p** eni olan sahənin kənar sağ mövqeyinə çıxarılır:

```
i:=134;write(i:6);nəticə:_ _ _134;
i:=123; write ((i+i):7); nəticə:_ _ _ _ 246;
```

3) Formatsız çıxarışla **r** kəmiyyətinin eksponensial formada, 18 simvol eni olan sahəyə onluq ifadəsi çıxarılır. Burada əgər **r**≥0,0 olarsa, **_#.##########E*##** formatından, əks halda **_-#.##########E*##** formatından istifadə edilir:

```
r:=666.787; write(r);
nəticə: _6.6678700000E+02;
r:=-1.999E+01; write(r);
nəticə:_-1.999000000E+01;
```

4) Formatlı çıxarışla **r** kəmiyyətinin **r:p** formatı ilə eksponensial formada onluq ifadəsi **p** eni olan sahənin kənar sağ mövqeyinə çıxarılır. Burada müsbət ədədlər üçün çıxarış sahəsinin minimal uzunluğu 7 simvol, mənfi ədədlər üçün isə 8 simvoldur:

```
r:=555.04; write(r:15); nəticə:
5.550400000E+02;
r:=46.78; write(-r:12); nəticə: -4.67800E+01;
```

5) Formatlı çıxarışla **r** kəmiyyətinin **r:p:q** formatı ilə qeyd olunmuş onluq nöqtə ilə onluq ifadəsi **p** eni olan sahənin kənar sağ mövqeyinə çıxarılır. Burada onluq nöqtədən sonra **q** (**0≤q≤24**) sayda rəqəm, ədədin kəsr hissəsini ifadə edir. Əgər **q=0** olarsa, nə onluq nöqtə, nə də kəsr hissə çıxarılmır və əgər q > 24 olarsa isə ədəd eksponensial formada çıxarılır:

```
r:=511.04; write(r:8:4); nəticə: 511.0400;
r:=-46.78; write (r:7:2); nəticə: _-46.78;
```

6) Formatlı çıxarışla **c** kəmiyyəti **c:p** formatı ilə **p** eni olan sahənin kənar sağ mövqeyinə çıxarılır: **c:='x'; write(c:3);** nəticə: **__x;**

7) Formatsız çıxırışla **s** kəmiyyəti kursorun durduğu mövqedən başlayaraq çıxarılır: **s:='PASCAL'; write(s);** nəticə: **PASCAL;**

8) Formatlı çıxarışla **s** kəmiyyəti **s:p** formatı ilə **p** eni olan sahənin kənar sağ mövqeyinə çıxarılır:

```
s:='PASCAL'; write(s:10);
```


nəticə: **_____PASCAL;**

9) Formatsız çıxarışla **m** kəmiyyəti kursorun durduğu mövqedən başlayaraq çıxarılır: **m:= true; write (m);** nəticə: **true;**

10) Formatlı çıxarışla **m** kəmiyyəti **m:p** formatı ilə p eni olan sahənin kənar sağ mövqeyinə çıxarılır:

```
m:=false; write(m:6, not m:7);
```

nəticə: **_false___true.**

5.6. Nişanlar və keçid operatorları. Şərt operatoru. Seçki operatoru

Turbo Pascal-da nişan – proqramın ixtiyari opretorunu adlandırmağa imkan verən və beləliklə, ona müraciəti təmin edən ixtiyari identifikatordur. Standart Pascal dili ilə uyğunluq yaratmaq üçün Turbo Pascal dilində də nişan kimi işarəsiz tam ədədlərdən isitfadə edilə bilər. Nişan işarələdiyi operatorun bilavasitə qarşısında yerləşdirdirilir və ondan «:» işarəsi ilə ayrılır. Operatoru bir neçə nişanla işarə etmək olar, bu halda bu nişanlar birbirindən «:» işarəsi ilə ayrılır. Proqramda nişandan istifadə etməməzdən əvvəl onu təsvir etmək lazımdır. Nişanlar təsvirlər bölməsinin **label** (nişan) bölməsində elan edilir. Məsələn, **label 1,2,11,12;** və s.

Proqramda idarəetməni şərtsiz olaraq proqramın bu və ya digər hissəsinə vermək üçün şərtsiz keçid operatorundan istifadə olunur. Bu operatorun ümumi şəkli aşağıdakı kimidir:

goto <nişan>;

Burada **goto** operatorun işçi sözü, <nişan> isə proqramda istifadə olunmuş hər hansı nişandır. Bu operator yerinə yetirilərkən idarəetmə nişanı verilmiş operatora verilir. Bu operatordan istifadə edərkən aşağıdakıları nəzərə almaq lazımdır:

1) **goto** operatorunda istifadə edilən nişan təsvirlər bölməsində elan edilməlidir və bu nişan proqramda istifadə edilməlidir;

2) Prosedurda (funksiyada) təsvir olunmuş nişanlar burada lokallaşdırılır, buna görə də prosedurdan (funksiyadan) kənardan bu nişanlara müraciət etmək olmaz.

Ümumiyyətlə, müasir proqramlaşdırma texnologiyası «**goto** operatoru olmadan proqramlaşdırma» prinsipinə əsas-

lanıb. Hesab olunur ki, bu operator proqramı mürəkkəbləşdirir, lakin buna baxmayaraq bəzi hallarda həmin keçid operatorundan istifadə əlverişli olur.

Şərt operatoru müəyyən şərti yoxlayaraq, yoxlamanın nəticəsindən asılı olaraq bu və ya digər əməliyyatı yerinə yetirməyə imkan verir. Beləliklə, şərt operatoru hesablama prosesinin budaqlanmasını təmin edən vasitədir. Şərt operatorunun strukturu aşağıdakı şəkildədir:

if <şərt> **then** <operator 1> **else** <operator 2>;

Burada **if**, **then**, **else** operatorun işçi sözləridir, <şərt> - məntiqi tipli ixtiyari ifadədir, <operator 1> və <operator 2> isə Turbo Pascal dilinin ixtiyari operatorlarıdır.

Bu operator yerinə yetirilərkən, əvvəlcə <şərt> məntiqi ifadəsi hesablanır. Əgər ifadənin nəticəsi **true** (doğru) qiymətini alarsa, <operator1> yerinə yetirilir, <operator2> isə nəzərə alınmır, qiymət **false** (yalan) olarsa, əksinə <operator1> nəzərə alınmır və <operator2> yerinə yetirilir. *Məsələn*,

```
if x<0 then y:=x+1 else y:=2*x;
if (n>15) and (n<25) then a:=n+4 else
b:= n-5;
```

Misal, x və y ədədlərindən ən böyüyünü tapmalı.

```
program p1;
var x,y,max:integer;
begin read(x,y);
if x>y then max:=x else max:=y;
write(max)
end.
```

Şərt operatorunun **else** < operator 2 > hissəsi verilməyə də bilər, yəni operator aşağıdakı şəklə düşər:

if <şərt> **then** < operator 1>;

Bu zaman <şərt> **true** qiymətini aldıqda <operator 1> yerinə yetirilir, əks halda isə operator yerinə yetirilmədən proqramın növbəti sətrinə keçid yerinə yetirilir.

Misal. x, y, z ədədlərindən ən kiçiyini tapmalı.

```
program p2;
var x,y,z,min:integer;
begin read(x,y,z);min:=x;
```



```
if min>y then min:=y;
if min>z then min:=z;
write(min)
end.
```

Buradakı <operator 1> və <operator 2> - dən ixtiyari biri, ixtiyari tipli, o cümlədən digər şərt operatoru da ola bilər və operatorda **else** <operator 2> hissəsi verilməyə də bilər. Bu zaman yarana bilən qeyri-müəyyənlik Turbo Pascal dilində aşağıdakı qaydada həll olunur: proqram mətnində rast gəlinən ixtiyari **else** hissəsi ona proqramda ən yaxın olan **THEN** hissəsinə uyğun gətirilir.

Misal. Kvadrat tənliyin həllini tapmalı.

```
program kv;
var a,b,c,d,x1,x2:real;
begin read(a,b,c);d:=sqr(b)-4*a*c;
if d>=0 then begin x1:=(-b+sqrt(d))/(2*a);
                   x2:=(-b-sqrt(d))/(2*a);
                write('x1=',x1,'x2=',x2)
          end
     else write('helli yoxdur');
end.
```

Seçki operatoru proqramın bir neçə mümkün davam variantlarından hər hansı birini seçməyə imkan verir. Seçkinin aparıldığı parametr <seçki açarı> **real** və **string** tipləri istisna olmaqla ixtiyari tip ifadədir. Seçki operatorunun ümumi strukturu aşağıdakı kimidir:

case <seçki açarı> **of** < seçki siyahısı> **else** <operatorlar> **end;**

burada **case**, **of**, **else**, **end** işçi sözlərdir, < seçki açarı > - seçki açarı, <seçki siyahısı> - <seçki sabiti> : <operator> formalı bir və ya daha çox konstruksiyalar; <seçki sabiti> - isə <seçki açarı> ifadəsi ilə eyni tipli sabitdir, <operatorlar> - Turbo Pascal dilinin ixtiyari operatorlarıdır.

Seçki operatoru aşağıdakı qaydada işləyir: əvvəlcə <seçki açarı> ifadəsi hesablanır, sonra <seçki siyahısı> operatorları ardıcıllığında qarşısında hesablanmış ifadənin aldığı qiymətə bərabər olan sabit gələn operator seçilir. Tapılmış operator yerinə yetirilir

və seçki operatorunun işi sona çatır. Əgər seçki siyahısında seçki açarının hesablanmış qiymətinə uyğun gələn sabit tapılmazsa, onda idarəetmə **else** sözündən sonra gələn operatorlara verilir. Qeyd edək ki, operatorda **else** <operatorlar > hissəsini verməmək də olar. Bu halda seçki siyahısında lazım olan sabit tapılmadıqda heç bir hadisə baş verməyəcəkdir və seçki operatoru sadəcə olaraq öz işini sona çatdıracaqdır.

Misal. Toplama, çıxma, vurma və ya bölmə əməliyyatlarından hər hansı birinin seçilməsi ilə ixtiyari **x, y** ədədləri üçün bu əməlin yerinə yetirilməsi.

```
program p3;
var c:char; x,y,z:real; s:boolean;
begin s:=false;
repeat read(x,' ', y); read(c);
case c of
'+': z:=x+y;
'-': z:=x-y;
'*': z:=x*y;
'/': z:=x/y;
else s:=true;
end;
if not s then writeln('z =',z)
until s
end.
```

Burada seçki siyahısındakı ixtiyari operatorun qarşısında bir deyil, bir-birindən vergüllə ayrılan bir neçə seçki sabiti dura bilər. *Məsələn,*

```
var c:char;
begin read(c);
case c of
'n','N': wrieln ('No');
'y','Y': writeln ('Yes')
end
end.
```


5.7. Dövr operatorları

Turbo Pascal dilində üç müxtəlif dövr operatoru mövcuddur:

1) Hesabi dövr operatoru **FOR** aşağıdakı struktura malikdir:

for <dövr parametri>**:=** <başlanğıc qiymət> **to** <son qiymət> **do** <operator>**;**

Burada **for, to, do** – operatorun işçi sözləridir, <dövr parametri> – **integer** (daha doğrusu ixtiyari nizam tipli) tipli dəyişəndir, <başlanğıc qiymət> - dövr parametrinin başlanğıc qiymətidir və onunla eyni tiplidir, <son qiymət> – dövr parametrinin aldığı son qiymətdir və onunla eyni tiplidir, < operator > – Turbo Pascal dilinin ixtiyari operatorudur.

Bu operator yerinə yetirilərkən əvvəlcə <başlanğıc qiymət> ifadəsi hesablanır və <dövr parametri>**:=**<başlanğıc qiymət> mənimsədilməsi yerinə yetirilir. Sonra <dövr parametri> **<=** <son qiymət> şərti yoxlanılır, əgər şərt ödənilmirsə **for** operatorunun işi sona çatır, şərt ödənildikdə isə <operator> yerinə yetirilir və<dövr parametri> dəyişəni bir vahid artırılır: <dövr parametri>**:=** <dövr parametri>**+1** Bundan sonra təsvir etdiyimiz proses şərt ödənilməyənədək təkrarlanır.

Misal. Birinci N natural ədədin cəmini tapmalı.

```
program p4;
var i,n,s:integer;
begin readln(n);s:=0;
for i:=1 to n do s:=s+i;
    writeln(s)
end.
```

Qeyd edək ki, **for** operatorunun işini idarə edən şərt <operator> - un yerinə yetirilməsindən əvvəl yoxlanılır, əgər şərt **for** operatorunun işinin əvvəlində yerinə yetirilmirsə, onda operator bir dəfə də olsun yerinə yetirilməyəcəkdir. **for** operatorunda dövr parametrinin artım addımı sabitdir və (+1)-ə bərabərdir. Lakin **for** operatorunun aşağıdakı forması da mövcuddur:

for <dövr parametri>**:=** <başlanğıc qiymət> **downto** <son qiymət> **do** <operator>

Burada **to** sözünün **downto** sözü ilə əvəz edilməsi dövr parametrinin artım addımının (-1)-ə bərabər olduğunu və idarəedici şərtin <dövr parametri> **>=** <son qiymət> olduğunu göstərir. Məsələn, yuxarıda baxdığımız misal üçün proqramı elə dəyişdirmək olar ki, proqram həm müsbət, həm də mənfi ədədlərin cəmini tapa bilsin:

```
program p5;
var i,n,s:integer;
begin readln(n); s:=0;
if n>=0 then for i:=1 to n do s:=s+i
        else for i:=-1 downto n do s:=s+i;
writeln(s)
end.
```

2) Şərt qabaqcadan yoxlanılan **while** dövr operatorunun ümumi şəkli aşağıdakı kimidir:

while <şərt> **do** <operator>

Burada **while, do** – operatorun işçi sözləridir, <şərt> - məntiqi tipli ifadə, <operator> isə Turbo Pascal dilinin ixtiyari operatorudur.

Bu operator yerinə yetirilərkən əvvəlcə <şərt> ifadəsi hesablanır, əgər o, **true** qiymətini alırsa, yəni şərt ödənilərsə <operator> yerinə yetirilir və <şərt> ifadəsinin hesablanıb, yoxlanılması davam etdirilir. Bu <şərt> **false** qiyməti alana qədər təkrarlanır, <şərt> bu qiyməti alan kimi **while** operatoru işini sona çatdırır. Əgər elə birinci yoxlamada < şərt > **false** qiymətini alırsa <operator> bir dəfə də yerinə yetirilmədən **while** operatoru işini başa çatdırır.

Misal. $f(x)=\cos 2x$ funksiyasının [0,1] parçasında $h=0{,}05$ addımı ilə qiymətlər cədvəlini almalı.

```
program p6;
const x0=0; h=0.05; xs=1;
var x,y:real;
begin x:=x0;
  while x<xs+h do
  begin y:=cos(2*x); writeln(x:5:2,y:6:4);
        x:=x+h
  end
end.
```


3) Şərt sonradan yoxlanılan **repeat-until** dövr operatorunun ümumi şəkli aşağıdakı kimidir:

repeat <dövrün gövdəsi> **until** <şərt>; burada **repeat, until**-dövr operatorunun işçi sözləridir, <dövr gövdəsi> - Turbo Pascal dilinin operatorlarının ixtiyari ardıcıllığıdır, <şərt> - məntiqi ifadədir.

Operator yerinə yetirilərkən, <dövr gövdəsi> operatorları ən azı bir dəfə yerinə yetirilir və bundan sonra <şərt> məntiqi ifadəsi hesablanır. Əgər bu ifadə **false** qiymətini alırsa, yəni şərt ödənilmirsə, <dövrün gövdəsi> operatorları təkrarən yerinə yetirilir və bu prosses <şərt> **true** qiymətini alana qədər davam etdirilir. Bu halda dövr operatorunun işi sona çatır.

Misal. Müsbət **a** tam ədədi verilib (**a>1**). **n!>a** şərtini ödəyən ən kiçik **n** ədədini tapmalı.

```
program p7;
var p,n,a:integer;
begin read(a); p:=1; n:=1;
  repeat n:=n+1; p:=p*n
  until p>a;
  writeln(n)
end.
```

Burada gördüyümüz kimi **repeat – until** cütü **begin – end** operator mötərizələrinə oxşardır, ona görə də **until**-dən əvvəl «;» qoymaq məcburi deyildir.

Qeyd edək ki, dövrlər bir-birinin daxilində də verilə bilər. Bu zaman daxildəki dövr xaricdəki dövrün hər bir qiyməti üçün tam yerinə yetirilir.

Misal. **y=2k+n**-nin **n=1,2,3** və **k=2,4,6** hallarında qiymətlərini tapmalı.

```
program p8;
var n,k,y:integer;
begin for n:=1 to 3 do
        begin k:=2;
          while k<8 do
            begin y:=2*k+n;
                  write(n,k,y);
```

```
                k:=k+2
            end
        end
end.
```

Qeyd edək ki, **for**, **while** və **repeat** dövr operatorlarının idarə edilməsini sadələşdirmək üçün Turbo Pascal dilinə aşağıdakı iki prosedur daxil edilib:

1) **break** – dövrdən dərhal çıxışı təmin edir, bu prosedur idarəetməni dövr operatorundan dərhal sonra gələn operatora verir;

2) **continue** – dövrün növbəti təkrarlanmasının vaxtından əvvəl sona çatdırılmasını təmin edir, yəni idarəetmənin dövr operatorunun sonuna ötürülməsi prosesinə ekvivalentdir.

Misal. $1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{i}+...$ harmonik sırasının **eps** dəqiqliyi ilə qiymətini hesablayaq.

```
program p8;
var s,eps:real; i:integer;
begin read(eps); s:=0; i:=1;
    while (1/i>=eps) and (i<maxint) do
       begin s:=s+1/i; i:=i:+1 end;
    write (s)
end.
```

```
program p9;
var s, eps: read; i:integer;
begin read(eps); s:=0; i:=1;
  repeat
    s:=s+1/i; i:=i+1
  until (1/i<eps) or (i>=maxint);
  write(s)
end.
```


5.8. Massivlər

Massiv – eyni bir identifikatorlarla işarələnən eyni tipli elementlər ardıcıllığıdır. Massivlərin təsvirinin ümumi şəkli aşağıdakı kimidir:

< tipin adı > **=array**[< indeks tiplərinin siyahısı >] **of** < tip >;

burada <tipin adı> - düzgün identifikatordur, **array**, **of** – operatorun işçi sözləridir, <indeks tiplərinin siyahısı> – kvadrat mötərizələr daxilində bir-birindən vergüllə ayrılmaqla gətirilən bir və ya bir neçə indeks tiplərinin siyahısıdır, <tip> – Turbo Pascal dilinin ixtiyari tipidir. İndeks tipi kimi Turbo Pascal-da **longint** tipindən və **longint** baza tipi olan diapazon tiplərindən başqa ixtiyari nizam tiplərindən istifadə etmək olar. Dəyişəni massiv kimi, onun bilavasitə təsviri zamanı, massiv tipinin əvvəlcədən təsviri olmadan təyin etmək olar. *Məsələn*,

```
var a,b:array[1..10] of real;
```

İndeks tipi kimi adətən indekslərin dəyişmə sərhədləri verilən diapazon tipindən istifadə edilir. Burada **OF** sözündən sonra gələn <tip> Turbo Pascal-ın ixtiyari tipi olduğundan, bu tip xüsusi halda digər massiv də ola bilər. *Məsələn*,

```
type mat=array[0..5]of array[-2..2] of
array[char] of byte;,
```

bu yazılışı aşağıdakı kimi də vermək olar:

```
type mat=array [0..5,-2..2,char] of byte;
```

İndeks tiplərinin siyahısındakı elementlərin sayı məhdudlaşdırılmır, lakin ixtiyari massivin daxili ifadəsinin uzunluğu 65520 baytı aşmamalıdır. Kompüter yaddaşında massiv elementləri bir-birinin ardınca gəlir və burada kiçik ünvanlardan daha yüksəklərinə keçid zamanı daha tez massivin ən sağda duran indeksi dəyişir. *Məsələn*,

```
var a:array[1..2,1..2] of byte;
begin a[1,1]:=1; a[2,1]:=2; a[1,2]:=3;
a[2,2]:=4;
end.
```

Bu halda kompüter yaddaşında ardıcıl olaraq bir-birinin ardınca 1, 2, 3, 4 qiymətlərinə uyğun baytlar düzüləcəkdir.

Turbo Pascal dilində bir mənimsətmə operatoru ilə bir massivin bütün elementlərini onunla eyni tipli olan başqa massivə mənimsətmək olar.

Məsələn,

```
var a,b:array[1..5] of single;
begin
......
  a:=b;
......
end.
```

Nəticədə, **A** massivinin bütün beş elementi, **B** massivinin elementlərinin aldığı qiymətləri alacaqdır. Lakin massivlər üzərində münasibət əməliyyatları nəzərdə tutulmayıbdır. Massivlərin yalnız elementlərini müqayisə etmək olar. Aşağıdakı qayda ilə massivin maksimal elementini və onun massivdəki mövqeyini təyin etmək olar:

```
program p9;
var t:array[1..12] of integer;
    i,max,p:integer;
begin
  for i:=1 to 12 do read(t[i]);
  max:=t[1]; p:=1;
  for i:=2 to 12 do
    if t[i]>max then begin max:=t[i]; p:=i
end;
  write(max,p)
end.
```

Massivdəki elementləri artım sırası ilə düzmək üçün aşağıdakı qayda ilə çeşidləmə aparmaq olar:

```
......
for i:=1 to n-1 do
for k:=1 to n-i do
if t[k]>t[k+1] then
begin  a:=x[k];  x[k]:=x[k+1];  x[k+1]:=a
end;
```


5.9. Yazılışlar

Yazılış – yazılış sahələri adlanan, qeyd olunmuş sayda elementdən ibarət verilənlər strukturudur. Massivdən fərqli olaraq, yazılışın elementləri (sahələri) müxtəlif tiplidir. Yazılışın bu və ya digər elementinə müraciət etmək mümkünlüyünü təmin etmək üçün sahələr adlandırılırlar. Yazılışın elanının ümumi strukturu aşağıdakı kimidir:

<tip adı> **=record** <sahələr siyahısı> **end;**

burada <tip adı> – düzgün identifikator, **record, end** – operatorun işçi sözləridir, <sahələr siyahısı> – ayrıcı kimi aralarında «**;**» işarəsi qoyulan yazılış bölmələrinin ardıcıllığından ibarət sahələr siyahısıdır.

Yazılışın hər bir bölməsi, bir-birindən vergüllə ayrılan bir və ya bir neçə sahə identifikatorlarından ibarətdir. İdentifikatordan sonra «**:**» işarəsi və sahə tipinin təsviri verilir. *Məsələn,*

```
type birthday=record
                day, month: byte;
                year: word;
              end;
var a,b:brithday;
```

Massivlərdə olduğu kimi yazılış tipinin dəyişənlərinin qiymətini, eyni tipli başqa dəyişənlərə mənimsətmək olur, məsələn, **a:=b;**

Yazılışın hər bir elementinə müraciət üçün elementin qurma adından istifadə edilir:

<dəyişən adı>.<sahə adı>;

Məsələn, **a.day:=21; b.year:=1966;**

Yazılışda sahələr bir-birinin daxilində də verilə bilər. *Məsələn,*

```
type brithday=record
                day,month:byte;
                year:Word
              end;
var c:record
        name:string;
```

```
        bd:brithday
      end;
begin
.....
If c.bd.year=1966 then ...
end.
```

Yazılış sahələrinə müraciəti sadələşdirmək üçün **with** birləşdirmə operatorundan istifadə edilir:

with <dəyişən> **do** <operator>;

burada **with, do** – operatorun işçi sözləridir, <dəyişən> – ardınca daxilində ola biləcək sahələrin siyahısı olan yazılış tipli dəyişən adıdır, <operator> – Turbo Pascal dilinin ixtiyari operatorudur. *Məsələn,*

```
whith c.bd do month:=4;
```

və ya ona ekvivalent olan aşağıdakı operatorlar

```
with c do with bd do month:=4;
```

və ya

```
with c,bd do month:=4;
```

və ya

```
c.bd.month:=4;
```

Misal. Tələbələrin proqramlaşdırma fənni üzrə imtahan cədvəli verilib. Əla qiymətlər almış tələbələrin sayını tapmalı.

```
program imt;
type t=record
         saa:string[30];
         qiy:2..5
       end;
var s:t; i,k,n:byte;
begin k:=0; readln(n);
  for i:=1 to n do
    begin
      read(s.saa); read(s.qiy);
      if s.qiy=5 then k:=k+1
    end;
   write (k)
end.
```


5.10. Çoxluqlar

Çoxluqlar – bir-birilə məntiqi əlaqələndirilən eyni tipli obyektlər külliyatıdır. Obyektlər arasındakı əlaqə istifadəçi tərəfindən müəyyən olunur və Turbo Pascal tərəfindən heç bir qayda ilə nəzarət edilmir. Çoxluqlar sonlu olur və buradakı elementlərin sayı 0-dan 256-ya qədər dəyişə bilər. Elementləri olmayan çoxluq boş çoxluq adlanır. Massiv elementlərindən fərqli olaraq çoxluq elementləri nömrələnməyib və müəyyən qayda ilə düzülməyib. Əməliyyatlar yalnız bütövlükdə çoxluq üzərində aparıla bilər. Çoxluğun konkret qiymətləri çoxluq konstruktorunun köməyi ilə daxil edilir. Konstruktor, kvadrat mötərizələrə alınan, bir-birindən vergüllə ayrılmış elementlər ardıcıllığıdır. Elementlər sabit və ya baza tipinin ifadələri ola bilər. Məsələn, **[3,4,7,9,12], [1..100], ['a','b','c'], ['A'..'Z']** və s. Burada **[ ]** simvolu boş çoxluğu, yəni heç bir elementi olmayan çoxluğu bildirir. İki çoxluq o vaxt ekvivalent hesab edilir ki, onların bütün elementləri eyni olsun və burada elementlərin çoxluq daxilində verilmə qaydasının əhəmiyyəti yoxdur. Əgər bir çoxluğun bütün elementləri, digər çoxluğa da daxildirsə, onda birinci çoxluğun ikinci çoxluğa daxil olunduğu qəbul olunur. Boş çoxluq ixtiyari çoxluğa daxildir.

Çoxluq tipi aşağıdakı formada təsvir olunur:

< tip adı > **=set of** < baza tipi >;

burada < tip adı > - düzgün identifikatordur, **SET, OF**-operatorun işçi sözləridir, < baza tipi > çoxluq elementlərinin baza tipidir. Baza tipi **WORD, INTEGER, LONGINT** tiplərindən başqa ixtiyari nizam tipi ola bilər.

Məsələn,

```
type D1=set of '0'..'9'; D2=set of 0..9;
var S1,S2,S3:D1;S4,S5,S6:D2;
begin
S1:=['1','2','3']; S2:=['3','2','1'];
S3:=['2','3'];S4:=[0..3,6];S5:=[4,5];
S6:=[3..9];
end.
```

Burada **S1** və **S2** çoxluqları ekvivalentdir, **S3** isə **S2**-yə daxildir, lakin onlar ekvivalent deyillər.

Çoxluqlar üzərində aşağıdakı əməliyyatlar nəzərdə tutulub:

1) Çoxluqların kəsişməsi: **A*B**. İki **A** və **B** çoxluqlarının kəsişməsi, eyni zamanda A və B çoxluqlarına aid olan elementlərdən təşkil olunmuş çoxluqdur. **S4*S6**-nın nəticəsi [3], **S4*S5**-nin nəticəsi isə boş çoxluqdur.

2) Çoxluqların birləşməsi: **A+B**. İki A və B çoxluqlarının birləşməsi, **A** və ya **B** çoxluqlarından heç olmasa birinə aid olan elementlərdən təşkil olunmuş çoxluqdur. Məsələn, **S4+S5**-in nəticəsi [0,1,2,3,4,5,6], **S5+S6**-nın nəticəsi [3,4,5,6,7,8,9].

3) Çoxluqların fərqi: **A-B**. **A** və **B** çoxluqlarının fərqi, **A** çoxluğunun B çoxluğuna aid olmayan elementlərindən təşkil olunmuş çoxluqdur. Məsələn, **S6-S5**-in nəticəsi [3,6,7,8,9], **S4-S5**-in nəticəsi [0,1,2,3,6].

4) Ekvivalentliyin yoxlanılması: **A=B**.Əgər çoxluqlar ekvivalentdirsə, nəticə **TRUE** olacaqdır, əks halda cavab **FALSE** olacaqdır.

5) Ekvivalent olmamasının yoxlanılması: **A<>B**. Əgər çoxluqlar ekvivalent deyillərsə, nəticə **TRUE** qiyməti olacaqdır, əks halda nəticə **FALSE** olacaqdır.

6) Daxil olmanın yoxlanılması:

a) **A<=B**. Əgər birinci çoxluq ikinciyə daxildirsə, nəticə **TRUE** əks halda **FALSE** olacaqdır.

b) **A>=B**. Əgər ikinci çoxluq birinciyə daxildirsə, nəticə **TRUE**, əks halda **FALSE** olacaqdır. Məsələn, tutaq ki,

```
var M:set of Byte;
 M:=[3,4,7,9];
```

onda `M=[4,7,3,3,9] → true; M<>[7,4,3,9] → false;`

`[3,4]<=M → true; []<=M → true;`

`M>=[1..10] → false; M<=[3..9] → true.`

7) Aid olmasının yoxlanması: **X IN A**. Bu əməliyyat çoxluğun baza tipi ilə üst-üstə düşən skalyar kəmiyyətlə, çoxluq arasındakı əlaqəni müəyyən edir. Əgər **X** qiyməti **A** çoxluğuna daxildirsə, əməliyyatın nəticəsi **TRUE**, əks halda isə **FALSE** olacaqdır. *Məsələn*, **3 in S6** –nın nəticəsi **TRUE**, **2*2 in S1**-in isə **FALSE** olacaqdır.

Bu əməliyyatlardan əlavə olaraq çoxluqlara aşağıdakı iki proseduru da tətbiq etmək olar:

1) **INCLUDE** – çoxluğa yeni element daxil edir. Prosedura müraciətin ümumi şəkli aşağıdakı kimidir:

INCLUDE (S, I);

Burada S – **TSetBase** baza tipli elementlərdən ibarət çoxluq, I-isə bu tipə aid olan və çoxluqda daxil edilməsi tələb olunan elementdir.

2) **EXCLUDE** – çoxluqdan elementi çıxarır. Ümumi şəkli aşağıdakı kimidir:

EXCLUDE (S, I)

burada S və I parametrləri **INCLUDE** prosedurunda olduğu kimidir:

Misal 1. 2-dən N-ə qədər **(1<N≤255)** natural ədədlər arasında 6-ya qalıqsız bölünən ədədlər çoxluğunu və 2-yə və ya 3-ə qalıqsız bölünən ədədlər çoxluğunu ayırmalı.

```
program p13;
const n=20;
var S2,S3,S6,S23:set of 2..n;
      k:byte;
begin S2:=[];S3:=[];
for k:=1 to n do
if k mod 2=0 then S2:=S2+[k];
if k mod 3=0 then S3:=S3+[k]
end;
S6:=N2*N3; S23:=S2+S3;
For k:=1 to n do
If k in S6 then write(k);
For k:=1 to n do
If k in S23 then write(k)
end.
```

Misal 2. 2-dən N-ə qədər **(1<N≤255)** natural ədədlər ardıcıllığından bütün sadə ədədləri tapmalı (Eratosfen xəlbiri).

```
program Eratosfen;
const n=201;
var A,B:set of 2..n;k,p:integer;
begin A:=[2..n];B:=[];p:=2;
repeat
```

```
while not(p in A)do p:=p+1;
B:=B+[p];k:=p;
while k<=n do
begin A:=A-[k];k:=k+p end
until A=[];
for p:=2 to n do if p in B then writeln(p)
end.
```

5.11. Sətirlər

String (sətir) tipi apostrof işarələri arasına alınan simvollar ardıcıllığından ibarətdir. Bu tip birölçülü simvol tipli **ARRAY [0..N] OF CHAR** massivinə oxşardır, lakin ondan fərqli olaraq sətir dəyişəndəki simvolların sayı 0-dan N-ə qədər dəyişə bilir, burada N-sətirdəki simvolların maksimal sayıdır. N-in qiyməti **STRING[N]** tip elanı ilə təyin edilir və 255-dən çox olmamaq şərti ilə nizam tipli ixtiyari sabit ola bilər. Turbo Pascal dilində N-in qiymətini göstərməmək olar, bu sətrin uzunluğu maksimal mümkün, yəni **N=255** qəbul edilir. Sətir tipi ümumi şəkildə aşağıdakı kimi təsvir edilir:

var <identifikator>:string[<sətrin maksimal uzunluğu>]

Məsələn,

```
var Name:String[20];S:String;
```

Turbo Pascal dilində sətrə simvollar ardıcıllığı kimi baxılır. Sətir daxilindəki simvollar birdən başlayaraq nömrələnirlər, sətrin hər bir elementi sətrin adı və kvadrat mötərizədəki indeksi ilə göstərilə bilər. İndeks tam tipli müsbət sabit, dəyişən və ya ifadə ola bilər. İndeksin qiyməti sərhəddindən kənara çıxmamalıdır. *Məsələn*, **Name[5], Name[i], Name[k+1]** və s.

Sətirlərə birləşdirmə "+" əməliyyatını tətbiq etmək olur. *Məsələn,*

```
st:='a'+'b';
st:=st+'c';
```

Nəticədə alırıq **st→abc.** Əgər birləşdirilmiş sətrin uzunluğu maksimal uzunluğu aşırsa, onda " artıq " simvollar atılır.

Məsələn,

```
var st:string[1];
begin st:='123'; writeln(st)end.
```

proqramı nəticədə 1 simvolunu çap edəcək.

İki sətir arasındakı =, <, >, >, <, >=, <= münasibət əməliyyatları, sətirdəki simvolların daxili kodlaşdırma nömrələrinin müqayisəsi ilə aparılır. Əgər bir sətir o birindən qısadırsa, qısa sətirdə çatmayan simvollar **CHR(0)** qiymətləri ilə əvəz edilir. Məsələn, aşağıdakı münasibət əməliyyatları **TRUE** qiymətini verəcək:

```
'A'>'1';'Turbo'<'TurboPascal';'cosm1'<'cos
m2'; 'pascal'>'PASCAL';'MSDOS'='MSDOS'.
```

Sətirlər və simvollar üzərindəki digər əməliyyatlar aşağıdakı standart prosedur və funksiyaların köməyi ilə həyata keçirilir:

1) **CONCAT(S1,S2,...,SN)–String** tipli funksiya olub, **S1,S2,-...,SN** sətirlərini bir sətirdə birləşdirir. *Məsələn*, **Concat('AA', 'BB','C');** nəticədə **'AABBC'** verir.

2) **COPY(ST,P,N)** – **String** tipli funksiya olub, **ST** sətrinin **N** sayda simvolunun **P** nömrəli simvoldan başlayaraq təkrarını alır. *Məsələn*, **ST:='ABCDEFG';Copy(ST,2,3);** nəticədə **'BCD'** verər.

3) **DELETE(ST,P,N)** – proseduru, **ST** sətrinin **N** sayda simvolunu, **P** nömrəli simvoldan başlayaraq ləğv edir. Məsələn, **ST:='abcdefg'; Delete(St,3,2);** nəticədə **'abefg'** alınır.

4) **INSERT(SUBST,ST,P)** – proseduru, **SUBST** alt sətrini, **ST** sətrinə, onun **P** nömrəli simvolundan başlayaraq daxil edir. Məsələn, **ST:=TURBO; Insert('PASCAL',ST,6);** nəticə: **'TURBO PASCAL'.**

5) **LENGTH(ST)** – **INTEGER** tipli funksiya olub, **ST** sətrinin uzunluğunu təyin edir. Məsələn, **ST:='ALGORITM'; Length (ST);** nəticə: 8.

6) **POS(SUBST, ST)** – **INTEGER** tipli funksiya olub, **ST** sətrinə **SUBST** alt sətrinin birinci daxil olduğu mövqeyin nömrəsini təyin edir, əgər **SUBST** alt sətri **ST** sətrində tapılmazsa, funksiyanın nəticəsi sıfır olar. Məsələn,

```
ST:='abcdef';POS('cd',ST); nəticə: 3;
ST:='abcdef'; POS('k',ST); nəticə: 0.
```

7) **STR(X[:N[:M]],ST)** –proseduru, ixtiyari həqiqi və ya tam tipli **X** ədədini, **WRITELN** prosedurunun etdiyi qaydada **ST** simvollar sətrinə çevirir. Burada məcburi olmayan **:N** və **:M** parametrləri çevirmənin formatını təyin edirlər, belə ki, **:N,X** ədədinin simvol ifadəsi üçün ayrılan ümumi sahəni, **:M** isə **X** ədədinin kəsr hissəsindəki (əgər **X** həqiqi tiplidirsə) simvolların sayını təyin edir.

8) **VAL(ST,X,CODE)** – proseduru, **ST** simvollar sətrini, tam və ya həqiqi tipli **X** dəyişəninin onun tipi ilə təyin edilən daxili ifadəsinə çevirir. Burada əgər çevirmə tam yerinə yetirilibsə, **CODE** parametri sıfır qiymətini alır və **X** dəyişəninə çevirmə nəticəsi mənimsədilir, əks halda **CODE** parametri, **ST** sətrində səhv simvolun tapıldığı mövqeyin nömrəsini alır və **X**-ın qiyməti dəyişməz qalır.

9) **UPCASE(CH)** – char tipli funksiya olub, latın əlifbasının kiçik hərfini bildirən **CH** simvol ifadəsini, onun əlifbadakı uyğun böyük hərfinə çevirir. Burada **CH** ixtiyari digər bir simvol olduqda, funksiya onu olduğu kimi saxlayır.

Misal.

```
var x:real;y:integer;st,st1:string;
begin
st:=concat('12','345');{12345}
st1:=copy(st,3,Length(st)-2);{345}
insert('-',st1,2);{3-45}
delete(st,pos('2',st),3);{15}
 str(pi:6:2,st);{3.14}
val('3.1415',x,y);{y=2,x="3.1415"}
end.
```

5.12. Alt proqramlar

Alt proqramlar proqramı bir-birindən müəyyən mənada müstəqil olan hissələrə bölməyə imkan verir. Bu birinci növbədə yaddaşa qənaət etməyə imkan verir, alt proqram burada bir dəfə yazılır, lakin ona ixtiyari sayda müraciət etmək olur. Alt proq-

ramların daxilində öz növbəsində başqa alt proqramlar verilə bilər. Alt proqramlara müraciət üçün prosedurun çağırış operatorunda prosedurun çağırış adını və ya ifadədə funksiyanın adını vermək kifayətdir. Turbo Pascal dilində iki cür alt proqram – prosedur və funksiyalardan istifadə edilir. Burada funksiya prosedurdan yalnız onunla fərqlənir ki, funksiyanı təşkil edən operatorların yerinə yetirilməsinin nəticəsi yeganə qiymət və ya göstərici olur. Alt proqramı təsvir etmək üçün onun başlığını və gövdəsini vermək kifayətdir. Başlıqda alt proqramın adı və formal parametrlər elan olunur. Funksiya üçün başlıqda alınan nəticənin tipi də göstərilir. Başlığın ardınca alt proqramın gövdəsi gəlir, o da əsas proqram kimi təsvirlər və operatorlar bölməsindən ibarətdir. Alt proqramın təsvirlər bölməsində daha aşağı səviyyəli alt proqramların təsviri, onlarda isə digər alt proqramların təsviri və s. ola bilər. İxtiyari alt proqramdan istifadə edilməmişdən qabaq o, təsvir olunmalıdır. Buna görə də aşağı səviyyədə olan alt proqramdan yuxarıdakı alt proqrama müraciət etmək olur, lakin əksinə bunu etmək mümkün deyildir. Yəni alt proqrama bu alt proqramın təsvirindən əvvəl təsvir olunmuş yuxarı səviyyə obyektlərə müraciət etmək olur. Bu cür obyektlər alt proqrama nəzərən qlobal adlandırılır. Turbo Pascalda təsvirlər bölməsində sabitlərin, dəyişənlərin, tiplərin, nişan və alt proqramların təsvirlərinin verilmə ardıcıllığı ixtiyari ola bilər.

Alt proqramın təsviri başlıqdan və alt proqramın gövdəsindən ibarətdir. Prosedurun başlığı aşağıdakı şəkildədir:

Procedure <ad>(formal parametrlərin siyahısı);

Funksiyanın başlığı isə aşağıdakı şəkildədir:

Function <ad>(formal parametrlərin siyahısı)**:<tip>**;

Burada <ad> – alt proqramın adını bildirən identifikatordur, <formal parametrlərin siyahısı> – alt proqramın formal parametrlərinin siyahısını təşkil edir, <tip> – funksiyanın nəticəsinin tipini bildirir. Alt proqramın başlığının ardınca aşağıdakı standart direktivalardan ixtiyari biri gələ bilər: **assembler**, **external**, **far**, **forward**, **inline**, **interrupt**, **near**. Bu direktivalar kompilyatorun işini dəqiqləşdirir və yalnız bu alt proqrama aid olur, ondan sonra gələ bilən alt proqramlara aid olmur. **Assembler** direktivası prosedura giriş və çıxış zamanı yaranan

maşın göstərişlərinin standart ardıcıllığını ləğv edir. Bu zaman alt proqramın gövdəsi daxili **assembler** əmrlərinin köməyilə yerinə yetirilməlidir. **External** direktivası ilə xarici alt proqram elan edilir. **Far** – çağırışın uzaq modelinə əsaslanan alt proqram kodu yaratmaq haqqında kompilyatora göstəriş verir. **Near** – direktivası isə çağırışın yaxın modelinə əsaslanan alt proqram kodu yaratmaq haqqında kompilyatora göstəriş verir. Proqramlarda iki yaddaş modelindən: yaxın və uzaq modellərindən istifadə edilə bilər. Yaddaş modeli proqramın müxtəlif yerlərindən proseduraların çağırış imkanlarını müəyyən edir. Əgər yaxın modeldən istifadə edilirsə, çağırış yalnız 64 *kbayt* ətrafında mümkündür, uzaq model halında isə çağırış ixtiyari hissədən mümkün olur. **forward** – direktivası istifadə edilən alt proqramın təsvirinin proqram mətnində sonradan veriləcəyini kompilyatora xəbər verir. **inline** – göstərir ki, alt proqramın gövdəsi daxili maşın göstərişlərinin köməyi ilə idarə olunur. **interrupt** – kəsilmələrin emalı prosedurlarının yaradılması üçün istifadə olunur.

Başlıqdakı formal parametrlər siyahısı verilməyə də bilər. Əgər bu siyahı verilirsə, onda burada formal parametrlərin adları və onların tipləri göstərilməlidir. *Məsələn*,

```
Procedure S(a:real;b:integer;c:char);
Function F(a,b:real):real;
```

Siyahıdakı parametrlər bir-birindən " ; " ilə ayrılır və eyni tipli parametrlər alt siyahılarda birləşdirilə bilər. Alt proqramın gövdə operatorları formal parametrlər siyahısını təsvirlər bölməsinin genişlənməsi kimi nəzərdən keçirir, belə ki, bu siyahıdakı bütün dəyişənlər alt proqram daxilində ixtiyari ifadələrdə istifadə edilə bilər. Alt proqrama müraciət onun adı üzrə yerinə yetirilir, addan sonra mötərizədə formal parametrlərin yerinə qoyulacaq faktiki parametrlər verilir. Turbo Pascal dilində formal parametrlərin sayı və tipi, alt proqrama müraciətdəki faktiki parametrlərin sayı və tipinə uyğun gəlməlidir. İstifadə olunan faktiki parametrlərin mənası, alt proqrama müraciətdə onların hansı ardıcıllıqla verilməsindən asılıdır. Alt proqramın ixtiyari formal parametri, ya parametr-qiymət, ya parametr-dəyişən və ya nəhayət parametr-sabit ola bilər. Parametr dəyişənlərin qarşısında **VAR** işçi

sözünü, parametr-sabitlərin qarşısında isə **CONST** sözünü vermək lazımdır. *Məsələn*,

```
procedure k(var a:real;b:real;
                          const c:string);
```

burada **a** – parametr-dəyişən, **b** – parametr-qiymət, **c** – isə parametr-sabitdir.

Formal parametr, parametr-dəyişən kimi elan edilibsə, onda alt proqrama müraciətdə ona eyni tipli dəyişən formasında faktiki parametr uyğun gəlməlidir. Formal parametr parametr-qiymət və ya parametr-sabit kimi elan edilibsə, onda alt proqrama müraciətdə ona ixtiyari ifadə uyğun gətirilə bilər. Bu qaydalara riayət olunması Turbo Pascal-ın kompilyatoru tərəfidən nəzarətdə saxlanılır. Əgər parametr, parametr-qiymət kimi təyin edilibsə, onda alt proqrama müraciətdən əvvəl, bu qiymət hesablanır, alınan nəticə müvəqqəti yaddaşa köçürülür və alt proqrama ötürülür. Parametr, parametr-dəyişən kimi təyin edildikdə, alt proqrama müraciət zamanı alt proqrama dəyişənin özü (təkrarı deyil) ötürülür və parametr-dəyişənin dəyişməsi, çağıran proqramdakı faktiki parametrin dəyişməsinə gətirir.

Parametr-sabit halında da alt proqrama dəyişənin və ya hesablanmış qiymətin yerləşdiyi yaddaş oblastının ünvanı verilir. Lakin parametr-sabitə alt proqram daxilində yeni qiymətin mənimsədilməsinin qarşısı kompilyator tərəfindən alınır.

Misal 1. Birinci **n** natural ədədin cəmini və hasilini tapmalı.

```
program p20;
var n,s,p:integer;
procedure prim(k:integer;var x,y:integer);
var i:integer;
begin x:=0; y:=1;
for i:=1 to k do begin x:=x+i; y:=y*i end
end;
begin read(n); prim(n,s,p);
writeln('s=',s); write('p=',p)
end.
```

Misal 2. ƏBOB-un tapılması üçün Evklid alqoritmi.

```
program p21;
var a,b,c:integer;
procedure Evklid(m,n:integer;var
k:integer);
begin while m<>n do
              if  m>n then m:=m-n else
                                  n:=n-m; k:=m
end;
begin write('a='); readln(a); write('b=');
readln(b);
Evklid(a+b,abs(a-b),c);
Evklid(c,a*b,c); writeln('c=',c)
end.
```

Misal 3. Misal 2-ni Function alt proqram ilə həll edək.

```
program  p22;
var a,b,c:integer;
function Evklid(m,n:integer):integer;
begin while m<>n do
if m>n then m:=m-n else n:=n-m;
Evklid:=m
end;
begin  write('a=');readln(a);  write('b=');
readln(b);
c:=Evklid(Evklid(a+b,abs(a-b)),a*b);
writeln('c=',c)
end.
```

Formal parametrlər siyahısındakı ixtiyari parametrin tipi yalnız standart və ya əvvəlcədən elan edilmiş tip ola bilər. Buna görə də məsələn, aşağıdakı proseduru elan etmək olmaz:

```
procedure s(a:array[1..10]of real);
```

Alt proqrama massiv ötürülürsə, bu massivləri əvvəlcədən təsvir etmək tələb olunur. Məsələn,

```
type mas=array[1..10]of real;
procedure s(a:mas);
```


Sətir də faktiki olaraq massiv olduğundan, onun da alt proqrama ötürülməsi analoji qaydada aparılır:

```
type st=string[15];st1=string[30];
function s(a: st):st1;
```

Funksiya və prosedurlardan, digər prosedur və funksiyalara müraciət zamanı faktiki parametr kimi istifadə edilməsini təmin etmək üçün prosedur tiplərdən istifadə edilir. Prosedur tipi elan etmək üçün adı göstərilməyən prosedur (funksiya) başlıqlarından istifadə edilir:

```
type p1=procedure(a,b,c:real;d:real);
     p2=procedure(var a,b);p3=procedure;
        f1=function:string;
        f2=function(var s:string):real;
```

Proqramda prosedur tiplərin dəyişənləri də elan edilə bilər. *Məsələn,*

```
var k:p1;L1,L2:f2;
   ap:array[1..N]of p1;
```

Prosedur tip dəyişənlərə, qiymət kimi uyğun alt proqramların adları mənimsədilə bilər.

Məsələn,

```
type Proc=Procedure(n:word; var a:byte);
var p1:proc;x,y:byte;
Procedure Proc1 (x:word; var y:byte);far;
begin if  x>255  then  y:=x  mod  255  else
y:=Byte(x)
end;
begin p1:=proc1;
for x:=150 to 180 do
begin p1(x+100,y); write(y:8)
end
end.
```

Turbo Pascal dilində qeyri-tip parametrlərdən də istifadə edilə bilir. Parametr o vaxt qeyri-tip sayılır ki, alt proqramın başlığında formal parametr-dəyişənin tipi göstərilməsin, ona uyğun faktiki parametr ixtiyari tipli dəyişən ola bilər. Qeyd edək ki, qeyri-tip yalnız parametr-dəyişənlər ola bilər. Qeyri-tip parametr-

lərdən, verilənlərin tipi vacib olmadığı halda istifadə edilir.

Rekursiya – alt proqramın onu təşkil edən operatorların yerinə yetirilməsi prosesində özü-özünə müraciətdir. *Məsələn,*

```
program Factorial;
var n:integer;
Function Fac(n:integer):extended;
var F:extended;
begin
if n=0 then Fac:=1 else begin F:=Fac(n-1);
Fac:=F*n end
end;
begin readln(n);Writeln('n!=',Fac(n))end.
```

Rekursiv müraciət dolayısı yolla verilə bilər. Bu halda alt proqram özünə, ona müraciət olan digər alt proqramı çağırmaqla müraciət edilir. *Məsələn,*

```
Procedure A(i:byte);
begin
.....
B(i);
.....
end;
Procedure B(j:byte);
.....
begin
.....
A(j);
.....
end;
```

Lakin ixtiyari identifikatordan istifadə etməmişdən əvvəl onu təsvir etmək tələb olunduğundan bu cür proqram konstruksiyasından istifadə etmək olmaz. Bu cür müraciətin mumkun olması üçün qabaqlayıcı təsvirdən istifadə edilir:


```
Procedure B(j:byte); forward;
Procedure A(i:byte);
begin
.....
B(i);
.....
end;
Procedure B;
begin
.....
A(j);
.....
end;
```

5.13. Fayllar

Fayl dedikdə biz, kompüterin xarici yaddaşının adlandırılmış oblastını yaxud informasiya qaynağı və ya qəbuledicisi olan məntiqi qurğu başa düşürük. İxtiyari faylın üç xarakter əlaməti var. Birinci addır ki, proqrama eyni zamanda bir neçə faylla işləməyə imkan verir. İkincisi, hər bir fayl yalnız eyni bir tip elementlərdən ibarət olur və bu tip fayllardan başqa Turbo Pascal-ın ixtiyari tipi ola bilər. Üçüncüsü isə odur ki, yeni yaradılan faylın uzunluğu qabaqcadan elan edilmir və yalnız xarici yaddaş qurğularının həcmi ilə məhdudlaşdırılır.

Fayl tipini və ya fayl tipli dəyişəni aşağıdakı üç üsuldan biri ilə vermək olar:

```
<ad>= file of <tip>;
<ad >=text;
<ad >=file;
```

Burada < ad > faylın tipinin adı, **file, of**-operatorun işçi sözləri, **TEXT**-mətn fayllarının standart tipinin adı, < tip > - fayl tipindən başqa Turbo Pascal dilinin ixtiyari tipidir. *Məsələn,*

```
type p=record
        name:string;
        code:word;
        cost:comp
end;
t1= file of string[80];
var f1:file of char:f2:text;f3:file;
      f4:t1;f5:file of p;
```

Elan üsulundan asılı olaraq faylların aşağıdakı üç formasını göstərmək olar:

1) tip fayllar (**file of** ilə verilir);

2) mətn faylları (**text** tipi ilə təyin edilir);

3) qeyri-tip fayllar (**file** tipi ilə təyin edilir).

Misalda **f1**, **f4**, **f5**-tip fayllar, **f2**-mətn faylı, **f3** isə qeyri-tip fayldır. Fayl forması ümumiyyətlə informasiyanın faylda saxlanılması qaydasını təyin edir. İxtiyari proqram qabaqcadan elan edilmiş standart fayl dəyişənli iki fayl: **input** (verilənlərin klaviaturadan oxunması üçün) və **output** (verilənlərin ekrana çıxarılması üçün) fayllarından istifadə edə bilir. Yerdə qalan ixtiyari fayllardan, eləcə də məntiqi qurğulardan proqramda, onların açılmasını təmin edən xüsusi prosedurların yerinə yetirilməsindən sonra isitfadə etmək olar. Bu prosedura əvvəlcədən elan edilmiş fayl dəyişənin mövcud və ya yaradılan faylın adı ilə əlaqələndirilir, eləcə də informasiya mübadiləsinin istiqamətini: fayldan oxunmaq və ya fayla yazılış olacağını təyin edir.

Fayl dəyişəni, fayl adı ilə **assign** standart proseduru vasitəsilə əlaqə yaradır:

assign (<fayl dəyişəni>, < fayl adı və ya məntiqi qurğu>);

burada <fayl dəyişəni> - proqramda fayl tipli dəyişən kimi elan edilmiş identifikatordur, < fayl adı və ya məntiqi qurğu> - fayl və ya məntiqi qurğu adından ibarət məntiqi ifadədir. Əgər fayl adı boş sətir şəklində verilirsə, məsələn, **assign(f,' ')**, onda informasiya mübadiləsinin istiqamətindən asılı olaraq fayl dəyişəni **input** və ya **output** standart faylı ilə əlaqə yaradır.

Fayl adı aşağıdakı şərtləri ödəyən ixtiyari sətir tipli ifadədir:

1) ad – böyük və kiçik latın hərfləri, rəqəmləri və **!,@,#,$,%,^, &,(),`,~,-, _** işarələrinin 8-ə qədər ixtiyari ardıcıllığıdır, ad bu simvollardan ixtiyari biri ilə başlaya bilər, addan sonra faylın üçdən çox olmayan sayda ad genişlənməsi verilə bilər, ad genişlənməsi addan nöqtə ilə ayrılır. Addan əvvəl isə fayla gedən yol verilə bilər. Bu yol disk adından və (və ya) cari kataloq adından və daha yüksək səviyyəli kataloq adından ibarət olur. Disk adı A...Z simvollarından ixtiyari biri ola bilər və ondan sonra " : " işarəsi qoyulur. Burada A:, B: adları elastik disklərə, C:, D:, E: və s. isə bərk disklərə aiddir. Disk adı verilmədikdə cari disk qəbul edilir. Disk adından sonra faylın yerləşdiyi kataloq adı verilə bilər, əgər kataloq adından əvvəl "\" işarəsi varsa, deməli fayla yol diskin baş kataloqundan başlayır, bu işarə yoxdursa, onda yol cari kataloqdan başlayır. Kataloq adından sonra aşağı səviyyəli bir və ya bir neçə kataloq adı verilə bilər, bütün bu adlar bir-birindən "\" işarəsi ilə ayrılır. Adın yol ilə maksimal uzunluğu 79 simvoldur. *Məsələn,*

```
var f1: text;f2:file of string;
const name='c:\k1\k2\t3.txt';
begin assign(f1,'d1.dat');
          assign(f2,name)
end.
```

Kompüterin klaviatura, displey ekranı, printer və giriş-çıxış kanalları kimi standart aparat qurğuları Turbo Pascal-da məntiqi qurğular adlanan xüsusi adlarla təyin edilirlər. **con** – klaviatura və ya displey ekranını təyin edən məntiqi addır. Turbo Pascal-da bu fiziki qurğular arasında verilənlərin ötürülməsi istiqaməti üzrə fərq qoyulur: verilənləri yalnız klaviaturadan oxumaq və verilənləri yalnız ekrana çıxarmaq olur. **prn** – printerin məntiqi adıdır. Əgər kompüterə bir neçə printer qoşulubsa, onda onlarla əlaqə **lpt1**, **lpt2** və **lpt3** məntiqi adları üzrə yerinə yetirilir. **turbo.tpl** kitabxanasına aid olan **printer** standart kitabxana modulu **lst** fayl dəyişəninin adını elan edir və onu **lpt1** məntiqi qurğusu ilə əlaqələndirir. Bu isə printerə sadə müraciət təşkil etməyə imkan verir. *Məsələn,*

```
Uses Printer;
 begin writeln (lst,'TURBO PASCAL')
end.
```

aux – kompüterin digər maşınlarla əlaqəsinin yaradılması üçün istifadə olunan məntiqi addır. Bu ada uyğun kanal verilənlərin həm qəbulunu, həm də verilməsini təmin edə bilir. Lakin proqramda o, zamanın bir anında bu funksiyalardan yalnız birini icra edə bilir. Adətən, kompüterdə bu cür iki kanal olur və onlara **COM 1** və **COM 2** məntiqi qurğu adları verilir. **NUL** – «boş» qurğunun məntiqi adıdır. Bu qurğu qeyri-məhdud həcmli informasiyanın qəbuledici qurğusu kimi istifadə edilir. **NUL** qurğusuna informasiya mənbəyi kimi müraciət edildikdə faylın sonu əlaməti **EOF** qiyməti alınır.

Məntiqi qurğu ilə faylın dəyişəni **assign** proseduru ilə əlaqələndirilir. *Məsələn*,

```
var f1,f2:text;
 begin assign(f1,'AUX');
       assign(f2,'LPT2')
       end.
```

Turbo Pascal dilində faylı oxumaq üçün, fayla informasiya yazmaq üçün, eləcə də eyni zamanda həm informasiya oxumaq və yazmaq üçün açmaq olur. Fayldakı informasiyanı oxumaq üçün **reset** standart prosedurundan istifadə olunur:

reset (<fayl dəyişəni>);

burada <fayl dəyişəni> - əvvəlcədən **assign** prosedurunun köməyi ilə artıq mövcud fayl və ya informasiya qəbuledici məntiqi qurğusu ilə əlaqələndirilmiş fayl dəyişənidir. Bu prosedur yerinə yetirilərkən, disk faylını və ya məntiqi qurğunu informasiyanın oxunmasına hazırlaşır. Nəticədə xüsusi göstərici dəyişən faylın başlanğıcını, yəni sıfır sıra nömrəli elementi göstərəcəkdir.

Turbo Pascal-da **reset** proseduru ilə informasiyanın yalnız oxunması üçün açılmış tip fayllara **write** (yəni informasiyanın yazılması üçün) proseduru ilə müraciət etməyə icazə verilir. Bu isə əvvəlcədən yaradılmış tip faylları asanlıqla yeniləşdirməyə və ehtiyac olduqda onları genişləndirməyə imkan verir. **reset** prose-

duru ilə açılmış mətn fayllarına **write** və ya **writeln** prosedurlarını tətbiq etmək olmaz.

rewrite (<fayl dəyişəni>);

standart proseduru <fayl dəyişəni> ilə əvvəlcədən əlaqələndirilmiş fayl və ya məntiqi qurğuya informasiyanın yazılışını təmin edir. Bu prosedura ilə əvvəlcədən mövcud olan disk faylına informasiya yazmaq olmaz, belə ki, bu prosedurun yerinə yetirilməsi zamanı köhnə fayl ləğv edilir, yeni fayl informasiya qəbuluna hazırlaşdırılır və onun göstərici-dəyişəni sıfır qiymətini alır.

append (<fayl dəyişəni>);

standart proseduru əvvəlcədən mövcud olan mətni faylın genişləndirilməsi üçün ona informasiya yazılışını təmin edir və bu zaman göstərici faylın sonuna yerləşdirilir. Bu prosedurla tip və ya qeyri-tip fayllara informasiya yazmaq olmaz. Əgər mətni fayl əvvəlcədən **reset** və ya **rewrite** prosedurları ilə açılmışdırsa, onda **append** prosedurunun tətbiqi bu faylın bağlanmasına gətirəcək və bundan sonra fayl onun genişləndirilməsi üçün yenidən açılacaqdır.

İxtiyari tip fayllara tətbiq edilə bilən prosedur və funksiyalara baxaq:

1) **close** proseduru faylı bağlayır, bu zaman əvvəlcədən **assign** proseduru ilə fayl adı ilə fayl dəyişəni arasında yaradılmış əlaqə saxlanılır. Prosedurun formatı **close** (<fayl dəyişəni>); şəklindədir. Yeni faylın yaradılması və köhnə faylın genişləndirilməsi zamanı bu prosedura fayldakı bütün yeni yazılışları saxlayır və faylı kataloqda registrasiya edir. Bu prosedur yerinə yetirildikdən sonra faylla, fayl dəyişəni arasındakı əlaqə saxlandığından, həmin faylı əlavə **assign** prosedurundan istifadə etmədən yenidən açmaq olar.

2) **rename** proseduru faylın adını dəyişdirir. Prosedurun formatı **rename** (<fayl dəyişəni>, <yeni ad>); şəklindədir. Burada <yeni ad> -faylın yeni adını bildirən sətir ifadəsidir. Əgər fayl qabaqcadan **reset, rewrite** və ya **append** prosedurları ilə açılıbsa, onda **rename** prosedurunun yerinə yetirilməsindən əvvəl faylı bağlamaq lazımdır.

3) **erase** proseduru faylı ləğv edir. Prosedurun formatı aşağıdakı kimidir:

erase (<fayl dəyişəni>);

Əgər fayl qabaqcadan **reset, rewrite, append** prosedurları ilə açılıbsa, onda **erase** prosedurunun yerinə yetirilməsindən əvvəl faylı bağlamaq lazımdır.

4) **flush** proseduru faylın daxili buferini təmizləyir və beləliklə, diskdəki fayldan yerinə yetirilmiş axırıncı dəyişikliklərin saxlanılmasını təmin edir. Prosedurun formatı **flush** (<fayl dəyişəni>) ; formasındadır.

5) **eof** (<fayl dəyişəni>) : **Boolean** funksiyası faylın sona çatıb-çatmamasını müəyyən edən məntiqi funksiyadır. Fayl göstəricisi faylın sonunda olduqda **EOF** funksiyası **TRUE** qiymətini alır. Yazılış zamanı bu növbəti elementin faylın sonuna əlavə ediləcəyini, oxunuş zamanı isə faylın sonuna çatdığını bildirir.

6) **chdir** proseduru cari kataloqu dəyişdirir. Formatı: CHDIR (<yol>); burada <yol>-yeni kataloqa gedən yolu göstərən sətir ifadəsidir.

7) **getdir** proseduru cari kataloqun adını təyin edir. Formatı: **GETDIR** (<qurğu>, <kataloq>); burada <qurğu>-qurğunun nömrəsini: **0**- susmaqla qurğu, **1**-A diski, **2**-B diski və s. bildirən **WORD** tipli ifadə, <kataloq>-göstərilən diskdəki cari kataloqa gedən yolu bildirən **STRING** tipli dəyişəndir.

8) **mkdir** proseduru göstərilən diskdə yeni kataloq yaradır. Formatı: **mkdir** (<kataloq>); burada <kataloq>-kataloqa gedən yolu təyin edən, **string** tipli ifadədir.

9) **rmdir** proseduru kataloqu ləğv edir. Formatı: **rmdir** (<kataloq>); burada <kataloq> - ləğv edilən kataloqa gedən yolu təyin edən, **string** tipli ifadədir. Qeyd edək ki, ləğv edilən kataloq boş olmalıdır.

10) **ioresult:word** funksiyası axırıncı giriş-çıxış əməliyyatının şərti əlamətini verir. Əməliyyat müvəffəqiyyətlə sona çatarsa, funksiya sıfır qiymətini alar.

11) **diskfree** (<disk>): **longint** funksiyası göstərilən

diskdə azad sahənin həcmini baytlarla göstərir.

12) **disksize** (<disk>):**longint** funksiyası göstərilən diskin tam həcmini baytlarla göstərir.

13) **findfirst** proseduru göstərilən kataloqda qeydiyyatdan keçmiş fayllardan birincisinin atributlarını müəyyən edir. Formatı: **findfirst** (<maska>, <atributlar>, <ad>); burada <maska> faylın maskasını ifadə edən sətir ifadə, <atributlar> maskanı dəqiqiləşdirən **byte** tipli ifadə, <ad> isə faylın adı qaytarılan **searchrec** tipli dəyişəndir. Fayl maskasının formalaşdırılmasında aşağıdakı simvollardan istifadə olunur: *-bu işarə onun durduğu yerdə fayl adı və ya ad genişləndirilməsinə aid ixtiyari sayda simvolun yerləşdirilə biləcəyini göstərir, ?- işarəsi bildirir ki, bu yerdə icazə verilmiş bir simvol yerləşdirilə bilər. Məsələn, *.* maskası kataloqdakı bütün faylları seçir, C*.* isə C ilə başlayan bütün faylları seçir və s. Maskadan qabaq fayla gedən yol gətirilir. Burada <atributlar>-**findfirst** proseduruna müraciət zamanı hansı fayllardan istifadə etməyin mümkün olduğunu təyin edir. DOS.TPU modulunda fayl atributları aşağıdakı kimi elan edilir:

```
Const readonly=$01; - yalnız oxumaq üçün,
      hidden=$02; - gizlədilmiş fayl,
      sysfile=$04; - sistem faylı,
      volumeid=$08; - cild identifikatoru,
      directory=$10; - alt kataloqun adı,
      archive=$20; - arxiv faylı,
      anyfile=$3F; - ixtiyari fayl.
```

findfirst prosedurunun nəticəsi **searchrec** tip dəyişən qaytarır. Bu tip **DOS.TPU** modulunda aşağıdakı kimi təyin edilir:

```
type Searchrec=record
       fill:array[1..21]of Byte;
       attr:Byte;
       time:Longint;
       size:Longint;
       name:String[12]
end;
```

14) **findnext** proseduru kataloqdakı növbəti faylın adını təyin edir. Formatı: **findnext** (<fayl adı>); burada <fayl adı> - **SEARCHREC** tipli yazılışdır. Yazılışda fayl haqqında informasiya verilir.

15) **getftime** proseduru faylın yaradıldığı və ya axırıncı yeniləşdirildiyi tarixi təyin edir. Formatı: **getftime** (<fayl dəyişəni>, <tarix>); burada <tarix> - **longint** tipli dəyişəndir.

16) **setftime** proseduru faylın yaradılmasının və ya təzələnməsinin yeni tarixini təyin edir. Formatı: **setftime** (<fayl dəyişəni>, <tarix>); burada <tarix> -zaman və tarixi bildirir.

17) **getfattr** proseduru fayl atributlarını almağa imkan verir. Formatı: **getfattr** (<fayl dəyişəni>, <atributlar>); burada <atributlar>- Word tipli dəyişəndir.

18) **setfattr** proseduru fayl atributlarını təyin etməyə imkan verir. Formatı: **setfattr** (<fayl dəyişəni>, <atributlar>);

19) **fsearch:pathstr** funksiyası kataloqlar siyahısında fayl axtarır. Formatı: **fsearch** (<ad>, <kataloqlar siyahısı>); burada <ad>-axtarılan faylın adı, <kataloqlar siyahısı>- faylın axtarıldığı kataloqların siyahısı.

20) **fsplit** proseduru fayl adını hissələrinə ayırır, yəni fayla gedən yolu, onun adını, ad genişlənməsini ayrı-ayrı parametrlər şəklində qaytarır. Formatı: **fsplit** (<fayl>, <yol>, <ad>, <ad genişlənməsi>);

21) **fexpand:pathstr** funksiyası fayl adına ona gedən yolu və qurğunun adını əlavə edir. Formatı: **fexpand** (<fayl>);

Mətn faylları, **text** standart tipinə aid olan fayl dəyişənləri ilə əlaqə yaradır. Mətni fayllar, mətni informasiyanın saxlanılması üçün nəzərdə tutulub. Məsələn, proqram mətnləri, məhz bu tip fayllarda saxlanılır. Mətn fayl elementləri (yazılışları) dəyişən uzunluqlu ola bilər. Turbo Pascal-da mətni fayl dəyişən uzunluqlu sətirlər külliyatı kimi qəbul olunur. Hər bir sətrə birincidən başlayaraq ardıcıl keçmək olar.

Mətni fayl yaradılarkən hər bir sətrin sonunda xüsusi **eoln** (End of Line – sətir sonu) əlaməti, faylın sonunda isə **eof** (End of File – faylın sonu) əlaməti qoyulur. Fayldakı sətirlərə (yazılışlara) **Read**, **Readln**, **Write**, **Writeln** prosedurları ilə keçmək olar. Bu prosedurlar fayllara dəyişən saylı faktiki parametrlərlə müraciət edə bilir və parametr kimi simvol, sətir və ədədlərdən istifadə edilə bilər. Bu prosedurlarda birinci parametr fayl dəyişəni ola bilər. Bu halda disk faylına və ya məntiqi qurğuya müraciət yerinə yetirilir. Əgər fayl dəyişəni göstərilməyibsə, onda **input** və **output** standart fayllarına müraciət yerinə yetirilir.

READ proseduru simvol, sətir və ədədlərin daxil edilməsini təmin edir və aşağıdakı formata malikdir:

READ (<fayl dəyişəni>, <daxiletmə siyahısı>);

və ya

READ (<daxil etmə siyahısı>);

burada <daxil etmə siyahısı>-**char**, **string** tipli bir və ya daha artıq dəyişənlər, eləcə də ixtiyari tam və ya həqiqi tipli dəyişənlər ardıcıllığıdır.

READLN proseduru da simvol, sətir və ədədlərin daxil edilməsini təmin edir və **READ** porseduru ilə eynigüclüdür. Fərq yalnız ondadır ki, burada sətirdəki axırıncı dəyişən oxunduqdan sonra sətrin **EOLN** əlamatinə qədər qalan hissəsi buraxılır və buna görə də **READLN** və ya **READ** prosedurlarına növbəti müraciət yeni sətrin birinci simvolundan başlanır. Bundan əlavə bu proseduru <daxiletmə siyahısı> parametrini vermədən də tətbiq etmək olar. Bu isə cari sətirdəki **EOLN** əlamətinə qədər olan bütün simvolların buraxılmasına gətirib çıxarır.

WRITE proseduru mətni informasiyanın mətni fayla və ya məntiqi qurğuya ötürülməsini təşkil edir və aşağıdakı formata malikdir:

WRITE (<fayl dəyişəni>, <xaricetmə siyahısı>);

və ya

WRITE (<xaricetmə siyahısı>);

burada <xaricetmə siyahısı>-**CHAR**, **STRING**, **BOOLEAN** tipli bir və ya daha artıq dəyişənlər, eləcə də ixtiyari tam və ya həqiqi tipli dəyişənlər ardıcıllığıdır. Burada <fayl dəyişəni> varsa,

o əvvəlcədən **TEXT** tipli dəyişən kimi təsvir olunmalı və **ASSIGN** proseduru ilə fayl adı və ya məntiqi qurğu ilə əlaqələndirilməlidir.

WRITELN proseduru, **WRITE** proseduru ilə üst-üstə düşür. Yeganə fərq ondadır ki, xaric olunan simvollar sətri **CR** və **LF** kodları ilə qurtarır. Bu prosedurda <xaricetmə siyahısı>nı verməmək də olar. Bu halda kursor növbəti sətrin əvvəlinə keçirilir.

EOLN məntiqi funksiyası mətni faylda sətrin sonuna çatdıqda **TRUE** qiymətini verir və aşağıdakı formata malikdir:

EOLN (<fayl dəyişəni>);

Əgər burada <fayl dəyişəni> parametri verilmirsə, onda funksiya standart **INPUT** faylını yoxlayacaqdır.

SEEKEOLN məntiqi funksiyası sətir sonluğu əlaməti **EOLN** və ya birinci əhəmiyyətli işarəyə rast gəlinən bütün probel və tabulyasiya işarələrini buraxır və **EOLN** əlamətinə rast gəldikdə **TRUE** qiymətini alır. Fomatı aşağıdakı şəkildədir:

SEEKEOLN (<fayl dəyişəni>);

SEEKEOF məntiqi funsiyası fayl sonluğu əlaməti **EOF** və ya birinci əhəmiyyətli işarəyə qədər rast gəlinən bütün probel, tabulyasiya və sətir sonluğu **EOLN** işarələrini buraxır və **EOF** əlamətinə rast gəldikdə **TRUE** qiymətini alır. Formatı aşağıdakı kimidir:

SEEKEOF (<fayl dəyişəni>);

Misal 1. Note.txt mətni faylındakı mətnin neçə sətirdən ibarət olduğunu təyin etməli.

```
var Note:text; k:integer;
begin assign(Note,'Note.txt');
reset(Note); k:=0;
while not eof(Note) do
begin readln(Note); k:=k+1 end;
writeln(k); close(Note)
end.
```

Misal 2. Note.txt mətni faylındakı mətndə ən uzun sətri təyin etməli.

```
var Note: text; max, k: integer; c:char;
begin assign(Note,'Note.txt');
```



```
reset (Note); max:=0;
while not eof(Note) do
begin k:=0;
while not eoln(Note) do
begin read(Note,c); k:=k+1 end;
if k>max then max:=k; readln(Note)
end;
writeln(max); close(Note)
end.
```

Tip fayllarının ixtiyari elementinin uzunluğu ciddi sabitdir, bu isə onların hər birinə birbaşa müraciəti təşkil etməyə imkan verir (yəni elementə onun sıra nömrəsinə görə müraciət etmək olur). Giriş-çıxış prosedurlarına birinci müraciətdən əvvəl fayl göstəricisi faylın əvvəlində durur və sıfır nömrəli birinci elementi göstərir. Hər bir oxunma və ya yazılışdan sonra göstərici faylın növbəti elementinə doğru hərəkət edir. Giriş-çıxış siyahılarındakı dəyişənlər, fayl elementləri ilə eyni tipə aid olmalıdır.

READ proseduru tip faylının növbəti elementlərinin oxunmasını təmin edir və formatı aşağıdakı kimidir:

READ (<fayl dəyişəni>, <daxil etmə siyahısı>); burada <daxil etmə siyahısı>sı fayl elementləri ilə eyni tipli olan bir və ya bir neçə dəyişənlərdən ibarət siyahıdır. Burada <fayl dəyişəni> əvvəlcədən **FILE OF** ...cümləsi ilə elan edilməli və **ASSIGN** proseduru ilə faylın adı ilə əlaqələndirilməlidir. Fayl **RESET** proseduru ilə açılmalıdır.

WRITE proseduru tip faylına verilənlərin yazılışı üçün istifadə edilir və aşağıdakı formata malikdir:

WRITE (<fayl dəyişəni>, <xaric etmə siyahısı>); burada <xaric etmə siyahısı> -fayl elementləri ilə eyni tipli olan bir və ya bir neçə ifadədən ibarət siyahıdır.

SEEK proseduru fayl göstəricisini tələb olunan elementə doğru sürüşdürür və aşağıdakı formata malikdir:

SEEK (<fayl dəyişəni>, <N-ci element>); burada <N-ci element>-fayl elementini bildirən **LONGINT** tipli ifadədir. Faylın birinci elementi sıfır sıra nömrəsinə malikdir. Bu proseduru mətni fayllara tətbiq etmək olmaz.

FILESIZE funksiyası fayldakı elementlərin sayını göstərən **LONGINT** tipli qiymət alır və aşağıdakı formata malikdir:

FILESIZE (<fayl dəyişəni>);

Bu funksiyanı mətni fayllara tətbiq etmək olmaz.

FILEPOS funksiyası növbəti giriş-çıxış əməliyyatı ilə emal ediləcək fayl elementinin sıra nömrəsini ifadə edən **LONGINT** tipli qiymət qaytarır. Formatı aşağıdakı formadadır:

FILEPOS (<fayl dəyişəni>);

Bu funksiyanı mətni fayllara tətbiq etmək olmaz.

Misal 1. k1. dat faylına 20 həqiqi ədəd daxil etməli:

```
program H1;
var f:file of real; i:byte; x:real;
begin assign(f,'k1.dat'); rewrite(f);
for i:=1 to 20 do begin readln(x);
write(f,x)
end; close(f)
end.
```

Misal 2. k2.dat faylındakı ədədlərin cəmini tapmalı.

```
program H2;
var f1:file of real; s,x:real;
begin assign(f1,'k2.dat'); reset(f1);
s:=0;
 while not eof(f1) do
begin read(f1,x); s:=s+x end;
close(f1)
end.
```

Qeyri-tip faylları **FILE** tipli fayl dəyişənləri kimi elan olunurlar və digər fayllardan elementlərinin tipinin göstərilməsi ilə fərqlənirlər. Tipin göstərilməməsi bu faylları ixtiyari digər fayllara uyğunlaşdırılmağa imkan verir və disklə yaddaş arasında verilənlər mübadiləsinin sürətini artırmağa imkan verir. Qeyri-tip faylları **RESET** və ya **REWRITE** prosedurları ilə açarkən, bu faylların baytlarla uzunluğunu göstərmək olar. *Məsələn,*

```
var f:file;
begin assign(f,'k3.dat'); reset(f,512);
end.
```


Qeyri-tip faylın uzunluğu, **RESET** və **REWRITE** prosedurlarında ikinci parametr kimi verilir və bu parametr **WORD** tipli ifadə də ola bilər. Əgər uzunluq göstərilmirsə, o, 128 *bayta* bərabər qəbul edilir. Burada maksimal uzunluq 65535 *bayt* (**Word** tam tipinin tutumu) ola bilər. Qeyri-tip fayllarla iş zamanı, tip fayllara tətbiq olunan bütün prosedur və funksiyalardan istifadə olunur. Lakin **READ** və **WRITE** prosedurları daha sürətli **BLOCKREAD** və **BLOCKWRITE** prosedurları ilə əvəz olunur:

BLOCKREAD (<fayl dəyişəni>, <bufer>, < [, <NN>]);

BLOCKWRITE (<fayl dəyişəni>, <bufer>, < [, <NN>]);

burada <bufer> - disklərdə verilənlərin mübadiləsində istifadə olunan dəyişən adı, <NN> verilməsi məcburi olmayan parametrdir və prosedurdan çıxış zamanı faktiki emal edilmiş yazılışların sayını göstərir.

5.14. Göstəricilər və dinamik yaddaş

Dinamik yaddaş – verilənlər seqmenti (64 kbayt), stek (adətən 16 kbayt) və bilavasitə proqram gövdəsini çıxmaqla proqramın işi üçün kompüterin ayırdığı operativ yaddaşdır. Bu yaddaşın ölçüləri müxtəlif ola bilir. Susmaqla bu ölçü kompüterin bütün mümkün yaddaşı ilə dəstəklənir və adətən ən azı 200-300 *kbayt* olur. Dinamik yaddaş faktiki olaraq böyük ölçülü verilənlər massivlərinin emalı üçün yeganə vasitədir. Bir çox praktiki məsələləri dinamik yaddaş olmadan həll etmək çətinlik törədir və ya heç mümkün olmur. Eləcə də kompüterin qrafik və səs imkanları ilə iş zamanı verilənlərin müvəqqəti yadda saxlanılması üçün də dinamik yaddaşdan geniş istifadə olunur. Kompüterin operativ yaddaşı, hər birinin ünvanı olan bir bayt həcmli oyuqlar ardıcıllığından ibarətdir. Turbo Pascal dilində dinamik yaddaşın idarə edilməsi üçün göstəricilərdən istifadə edilir. Göstərici-qiymət yaddaş oyuğunun ünvanı olan dəyişəndir. Kompüterdə ünvanlar seqment və yerdəyişmə adlanan iki onaltı mərtəbəli sözlərin kulliyatı kimi verilir. Seqment-uzunluğu 65536 bayt (64 kbayt) olan və 16-nın vuruğu (yəni, 0,16, 32, 48 və s.) olan fiziki ünvanla başlayan yaddaş hissəsidir. Yerdəyişmə isə lazımlı ünvana müraciət üçün

seqmentin əvvəlindən neçə bayt buraxmaq lazım olduğunu göstərir. Kompüterin ünvan fəzası 1 Mbaytdır. Bir *Mbayt* hüdüdlarında ünvanlaşmaq üçün 20 dənə ikilik mərtəbə lazımdır ki, bu da 16-lıq sözdən (seqment və yerdəyişmə) aşağıdakı qayda ilə alınır: seqmentin tərkibi 4 mərtəbə sola yerdəyişmə edir, boşalmış sağ mərtəbələr sıfırlarla doldurulur və nəticə yerdəyişmənin tərkibi ilə toplanır. Burada 16 baytlıq yaddaş fraqmenti paraqraf adlanır, buna görə də seqment yaddaşı paraqraf dəqiqliyi ilə ünvanlayır, yerdəyişmə isə bayt dəqiqliyi ilə ünvanlaşma aparır. Hər bir seqmentə kəsilməz və ayrıca ünvanlanan yaddaş oblastı uyğun gəlir. Seqmentlər yaddaşda bir-birinin ardınca intervalsız və ya intervala gələ bilər və ya nəhayət, üst-üstə düşə bilər. Beləliklə, öz daxili strukturuna görə ixtiyari göstərici, seqment və yerdəyişmə kimi qəbul edilən iki sözün (**WORD** tipli verilənlər) külliyatıdır. Göstəricilərin köməyi ilə dinamik yaddaşda Turbo Pascal-ın ixtiyari tip verilənlərini yerləşdirmək olar. Adətən, Turbo Pascal-da göstərici müəyyən tip verilənlərlə əlaqələndirilir. Bu cür göstəricilər tip-göstəricilər adlanır. Bu göstəricilərin elanı üçün uyğun tip qarşısında ^ işarəsi qoyulur. *Məsələn*,

```
var p1:^integer;p2:^real;
type p3=^pr;
pr=record
n:string;
m:string;
next:pr
end;
```

Qeyd edək ki, göstəricilər, proqramda hələ elan edilməmiş verilənlər tipinə müraciət edə bilir.

Turbo Pascal-da göstəricini hansısa konkret verilənlər tipi ilə əlaqələndirmədən də elan etmək olar. Bunun üçün **POINTER** standart tipindən istifadə edilir. *Məsələn*,

```
var p:pointer;
```

Bu növ göstəricilər, qeyri-tip göstəricilər adlanır. Qeyri-tip göstəricilər hər hansı konkret tiplə əlaqəli olmadığından, onların köməyi ilə proqramın işi boyunca strukturu və tipi dəyişən verilənləri dinamik yerləşdirmək əlverişli olur. Göstəricilərin

qiyməti dəyişənlərin yaddaşdakı ünvanları olduğundan fərz etmək olardı ki, bir göstəricinin qiymətini digərinə vermək olar. Lakin bu belə deyil, Turbo Pascal-da eyni verilənlər tipi ilə əlaqəli olan göstəricilər arasında qiymətlər vermək olar. Məsələn, əgər

```
var p1,p2:^integer;p3:^real;
    p4:pointer;
```

onda p1:=p2; mənimsətməsi mümkündür, lakin p1:=p3; isə mümkün deyil, çünki p1 və p3 ayrı-ayrı verilənlər tipinə aiddir. Lakin bu qadağa qeyri-tip göstəricilərə şamil olunmur, buna görə də yazmaq olar:

```
p4:=p3;
p1:=p4;
```

Turbo Pascal-da dinamik yaddaş qalaq adlanan baytlar massivi kimi nəzərdən keçirilir. Fiziki olaraq, qalaq proqram gövdəsinin yerləşdiyi oblastdan sonra gələn böyük ünvanlarda yerləşir. Qalağın əvvəli **HEAPORG** standart dəyişənində, sonu isə **HEAPEND** dəyişənində yerləşir. Dinamik yaddaşın hələ tutulmamış hissəsinin cari sərhəddini **HEAPPTR** göstəricisi bildirir. Dinamik yerləşdirilən ixtiyari dəyişən üçün yaddaş hissəsi **NEW** proseduru ilə ayrılır. Bu prosedura müraciət parametri tip göstəricisidir. Müraciət nəticəsində göstərici, verilənlərin yerləşdirilməsini başlamaq mümkün olan dinamik ünvana uyğun qiymət alır. Göstərici müəyyən qiymət aldıqdan sonra, yəni yaddaşın konkret fiziki baytını göstərdikdə, bu ünvan üzrə uyğun tip ixtiyari qiymət yerləşdirmək olar. Bunun üçün göstəricidən sonra heç bir probel qoymadan ^ işarəsi verilir. *Məsələn*,

```
i^:=2;  r^:=2*pi;   və s.
```

Beləliklə, göstəricinin göstərdiyi qiymət, yəni qalaqda yerləşdirilmiş verilənlər, göstəricidən sonra gələn ^ işarəsi ilə işarələnir. Əgər göstəricidən sonra ^ işarəsi yoxdursa, onda verilənlərin yerləşdirildiyi ünvan nəzərdə tutulur. Dinamik yerləşdirilmiş verilənlərdən proqramın ixtiyari yerində istifadə etmək olar, məsələn, r^:=sqr (r^)+ i^-17; Eyni zamanda r:=sqr (r^)+i^; və ya r^=sqr (r); kimi yazılışlar düzgün deyil.

Dinamik yaddaşı qalaqdan götürməklə yanaşı, onu qalağa qaytarmaq da mümkündür. Bunun üçün DISPOSE prosedurun-

dan istifadə edilir. *Məsələn,*

```
var i,j:^integer;r:^real;
begin new(i);new(r);
.....
dispose(i);dispose(r);
.....
end.
```

Burada dispose prosedurları, new prosedurlarının i və r göstəriciləri üçün ayırdıqları 8 baytı, qalağa qaytarır. Qeyd edək ki, **DISPOSE (X)** proseduru **X** göstəricisinin qiymətini dəyişmir, yalnız bu göstərici ilə əvvəldən əlaqəli olan yaddaşı qalağa qaytarır. Boşalan göstəricini **NIL** sözü ilə qeyd etmək olar. Hər hansı göstəricinin qeyd olub-olmamasını aşağıdakı kimi yoxlamaq olar:

```
const p:^real= NIL;
begin if p=NIL then new(p);
.....
dispose(p);p:=NIL;
end.
```

Göstəricilər üzərində digər müqayisə əməliyyatları aparmaq qadağan olub.

Qalaqda olan bütün əməliyyatlar qalaq adminstratoru adlanan xüsusi alt proqramın vasitəsilə aparılır. **NEW** proseduruna müraciət zamanı həmin alt proqram tələb olunan dəyişənin yerləşdirilməsi üçün ən kiçik azad fraqment axtarıb tapır. Tapılan fraqmentin başlanğıcının ünvanı göstəricidə qaytarılır, fraqmentin özü və ya onun lazımi uzunluqlu hissəsi qalağın tutulmuş hissəsi kimi qeyd olunur. Digər imkan isə qalağın tam fraqmentinin azad olunmasından ibarətdir. Bu məqsədlə dinamik yaddaşın ayrılmasından əvvəl **HEAPPTR** göstəricisinin cari qiyməti **MARK** prosedurunun köməyi ilə dəyişən-göstəricidə yadda saxlanılır. İndi ixtiyari anda **MARK** prosedurunun yadda saxladığı ünvandan başlayaraq, dinamik yaddaşın sonuna qədər olan qalaq fraqmentini azad etmək olur. Bunun üçün RELEASE prosedurundan istifadə olunur. *Məsələn,*

```
var p,p1,p2,p3,p4,p5:^integer;
begin new(p1);new(p2);mark(p);
          new(p3);new(p4);new(p5);
```



```
.....
release(p);
end.
```

Qeyd edək ki, **NEW** prosedurunun parametri yalnız tip göstərici ola bilər. Qeyri-tip göstəricilərlə iş üçün aşağıdakı prosedurlardan istifadə olunur:

GETMEM(p,SIZE) – yaddaşın ayrılması;

FREEMEM(p,SIZE) – yaddaşın azad edilməsi;

Burada p – qeyri-tip göstərici, size – isə tələb olunan və ya azad edilən qalaq hissəsinin baytlarla ölçüsüdür.

Dinamik yadaşla iş üçün istifadə olunan prosedur və funksiyaları qeyd edək:

1) **ADDR** funksiyası, arqumentin ünvanından ibarət **POINTER** tipli nəticə verir. Ümumi forması: **ADDR (x)** kimidir, burada **X**-proqramın ixtiyari obyektidir (ixtiyari dəyişən, prosedur, funksiya adıdır).Qaytarılan ünvan ixtiyari göstərici ilə uyğunlaşır. Qeyd edək ki, analoji nəticəni @ əməliyyatı da verir.

2) **CSEG** funksiyası mikroprosessorun **CS** registrində saxlanılan qiyməti qaytarır. Ümumi forması: **CSEG**. Nəticə **WORD** tipli sözdə qaytarılır.

3) **DISPOSE** proseduru, əvvəlcədən tip göstərici üçün ayrılmış dinamik yaddaş fraqmentini qalağa qaytarır. Ümumi forması: **DISPOSE (TP)**, burada **TP**-tip göstəricidir.

4) **DSEG** funksiyası mikroprsessorun **DS** registrində saxlanılan qiyməti qaytarır. Ümumi forması: **DSEG**. Nəticə **WORD** tipli sözdə qaytarılır.

5) **FREEMEM** proseduru, əvvəlcədən qeyri-tip göstərici üçün ayrılmış dinamik yaddaş fraqmentini qalağa qaytarır. Ümumi forması: **FREEMEM (P, SIZE)**, burada P-qeyri-tip göstəricidir, **SIZE**-isə həmin fraqmentin baytlarla uzunluğudur.

6) **GETMEM** proseduru qeyri-tip göstərici üçün tələb olunan ölçülü dinamik yaddaş fraqmenti ayrır. Ümumi forması: **GETMEM(P,SIZE)**, burada P-qeyri-tip göstərici, **SIZE**-isə fraqmentin baytlarla uzunluğudur.

7) **MARK** proseduru **HEAPPTR** qalaq göstəricisinin cari qiymətini yadda saxlayır. Ümumi forması: **MARK(P)**, burada P-ixtiyari tip göstəricidir, bu göstəricidə **HEAPPTR**-ın cari qiyməti qaytarılacaq.

8) **MAXAVAIL** funksiyası qalağın ən böyük kəsilməz hissəsinin baytlarla ölçüsünü təyin edir. Ümumi forması: **MAXAVAIL**. Nəticə **LONGINT** tiplidir.

9) **MEMAVAIL** funksiyası qalağın ümumi boş sahəsinin baytlarla ölçüsünü təyin edir. Ümumi forması: **MEMAVAIL**. Nəticə **LONGINT** tiplidir.

10) **NEW** proseduru dəyişənin yerləşdirilməsi üçün qalaq fraqmenti ayırır. Ümumi forması: **NEW(TP)**, burada **TP**-tip göstəricidir.

11) **OFS** funksiyası göstərilən obyektin ünvanının yerdəyişməsini bildirən **WORD** tipli qiymət qaytarır. Ümumi forması: **OFS(x)**, burada X-ixtiyari tipli ifadə və ya prosedur adıdır.

12) **PTR** funksiyası verilmiş **SEG** seqmenti və **OFS** yerdəyişməsinə görə **POINTER** tipli qiymət qaytarır. Ümumi forması: **PTR(SEG,OFS)**, burada **SEG**-seqmenti bildirən **WORD** tipli ifadə, **OFS**-isə yerdəyişməni bildirən **WORD** tipli ifadədir. Funksiyanın qaytardığı qiymət, ixtiyari tip göstərici ilə uyğunlaşır.

13) **RELEASE** proseduru qalaq hissəsini azad edir. Ümumi forması: **RELEASE(PTR)**, burada **PTR**-əvvəlcədən **MARK** proseduru ilə yadda saxlanılan qalaq göstəricisinin qiymətini saxlayan ixtiyari tipli göstəricidir.

14) **SEG** funksiyası göstərilən obyektin ünvan seqmentini bildirən **WORD** tipli qiymət qaytarır. Ümumi forması: **SEG(x)**, burada **x**-ixtiyari tip ifadə və ya prosedur adıdır.

15) **SIZEOF** funksiyası göstərilən obyektin daxili ifadəsinin baytlarla uzunluğunu təyin edir. Ümumi forması: **SIZEOF(x)**, **x** – burada dəyişən, funksiya və ya tip adıdır.

İstifadəçi proqramı ilə qalaq arasında əlaqə yaradan işçi alt proqram – qalaq adminstratoru adlanır. Qalaq adminstratoru new, getmem, dispose, freemem və s. prosedurların işini təmin edir

və **HEAPPTR**, **FREELIST** göstəricilərinin qiymətini dəyişdirir. Burada **HEAPPTR** göstəricisi qalağın azad hissəsinin aşağı sərhəddinin ünvanını verir, **FREELIST** göstəricisi isə birinci azad blokun ünvanını verir.

5.15 Tip sabitlər

Turbo Pascal dilində tip sabitlərdən istifadə edilə bilir. Onlar sabitlərin elanı bölməsində aşağıdakı şəkildə verilirlər:

<identifikator>: <tip>= <qiymət>;

burada <identifikator> - sabitin identifikatoru, <tip>-sabitin tipi,<qiymət> - isə sabitin qiymətidir. Proqramın yerinə yetirildiyi müddət ərzində tip sabitlərə digər qiymətlər mənimsətmək olur, buna görə də onlar faktiki olaraq başlanğıc qiymətləri olan dəyişənlərdir. Tip sabitlər, fayldan başqa ixtiyari tipə aid ola bilərlər.

Sadə tiplərin və **STRING** tipinin sabitlərinin elanı çətinlik törətmir, çünki onların qiymətləri kimi qeyri-tip sabitlərdən və ya onların identifikatorlarından istifadə olunur. *Məsələn,*

```
type colors =(white,red,black);
const
c1:colors=red;name:string='Aliyev A.';
year:word=1966;x:real=4.1;min:integer =0;
max:integer=10;days:1..31=21;
```

Tip sabit – massivlər üçün başlanğıc qiymət kimi yumru mötərizələr daxilində, bir-birindən vergüllə ayrılan sabitlər siyahısından istifadə edilir. Məsələn,

```
type colors=(white,red,black);
const c1:array[colors] of string[5]
=('white','red','black');
c2:array[1..5]of byte=(1,2,3,4,5);
```

CHAR simvol tipli sabit-massivlər kimi uyğun uzunluqlu simvol sətrini vermək olar. Məsələn, aşağıdakı elanlar eynigüclüdür:

```
    const p1:array[0..5]of char=('0','1','2',
'3','4','5');
    p2:array[0..5]of char='012345';
```

Çoxölçülü sabit-massivlərin elanı zamanı, hər bir ölçüyə uyğun sabitlər əlavə yumru mötərizələrə alınır və bir-birindən vergüllə ayrılır. *Məsələn*,

```
   var i,j,k:byte;
   const mat:array[1..3,1..3]of
byte=((0,1,2),(3,4,5), (6,7,8));
    cube:array[0..1,0..1,0..2]of
integer=(((0,1,2), (3,4,5)),((6,7,8),
(9,10,11)));
    begin for i:=1 to 3 do
              for j:=1 to 3 do
write(mat[i,j]:3);
    writeln;
              for i:=0 to 1 do
              for j:=0 to 1 do
              for k:=0 to 2 do
write(cube[i,j,k]:3);
    writeln
    end.
```

Sabit-yazılışlar aşağıdakı kimi təyin edilir:

<identifikator>: <tip>= (<sahə qiymətlərinin siyahısı>);

burada <identifikator>-sabit identifikatoru, <tip>-yazılışın tipi, <sahə qiymətlərinin siyahısı>-sahə qiymətlərinin siyahısıdır. Qeyd edək ki, sahə qiymətlərinin siyahısı, <sahə adı:sabit> formalı ardıcıllıqlar siyahısıdır. *Məsələn*,

```
    type point=record x,y:real end;
         vec=array[0..1] of point;
    month=(Jan,Feb,Mar,Apr,May,Jun,Jly,Aug,
Sep, Oct,Nov,Dec);
    date=record
         d:1..31;
         m:month;
          y:1966..1999
            end;
```



```
    const p1:point=(x:0;y:-1);
         p2:vec=((x:1.1;y:1.2),(x:4.5;y:7.5));
         p3:date=(d:21;m:Apr;y:1966);
```

Sabit-çoxluqların qiymətləri düzgün çoxluq konstruktoru kimi verilir, məsələn,

```
    type days=set of 1..31;
         d1=set of '0'..'9';
         d2=set of 1..24;
    const workdays:days=[1..5, 8..12, 15..19,
22..26, 29, 30];
          k1:d1=['0','2','4','6','8'];
          k2:d2=[];
```

Tip göstəricinin yeganə qiyməti NIL ola bilər. *Məsələn*,

```
Const p:^real=NIL;
```

5.16. Modullar

Modullar aşağıdakı struktura malikdir:

unit <ad>;
interface
<interfeys hissəsi>
implementation
<yerinə yetirilən hissə>
begin
<operatorlar bölməsi>
end.

Burada **unit**-modulun başlığını bildirən işçi söz, <ad>-isə modulun adını bildirən identifikatordur. **interface**-modulun interfeys hissəsini başlayan işçi söz, **implementation**-isə modulun yerinə yetirilən hissəsini başlayan işçi sözdür. **begin**-modulun operatorlar bölməsini başlayan işçi sözdür, burada **begin**<operatorlar bölməsi> verilməyə də bilər. Nəhayət **end**

modulun sonunu bildirən işçi sözdür. Beləliklə, modul başlıqdan və ixtiyari boş ola biləcək üç tərkib hissədən ibarətdir.

Modulun başlığı **Unit** işçi sözündən və onun ardınca gələn modul adından ibarətdir. Bu ad modul mətninin yerləşdirildiyi disk faylının adı ilə üst-üstə düşməlidir. Məsələn, əgər **Unit** Global; başlığı verilibsə, onda uyğun modulun mətni Global.pas faylında yerləşməlidir. Modul adı onun digər modullarla və əsas proqramla əlaqə yaradılması üçün nəzərdə tutulub. Bu əlaqə USES <modullar siyahısı>; bölməsi ilə yaradılır. Burada **Uses**-işçi söz, <modullar siyahısı> isə əlaqə yaradılan modullar siyahısıdır. Siyahıda bir-birindən vergüllə ayrılan modul adları gətirilir. Məsələn, **Uses CRT, Graph, Global;**. Əgər proqramda **Uses** elan bölməsindən isitfadə edilirsə, bu bölmə əsas proqramın təsvirlər bölməsində birinci gəlməlidir. Bir modulda digər moduldan istifadə edilə bilər. Belə ki, **Uses** bölməsi modullarda **Interface** və ya **Implementation** sözlərindən sonra və ya nəhayət eyni zamanda hər iki sözdən sonra dərhal verilə bilər (yəni eyni zamanda iki **Uses** bölməsi verilə bilər).

Modulun interfeys hissəsi **Interface** işçi sözü ilə açılır. Bu hissədə modulun əsas proqramda və (və ya) digər modullarda istifadəsi mümkün olan bütün qlobal obyektlərinin (tiplərin, sabitlərin, dəyişənlərin və alt proqramların) elanı verilir. Qlobal alt proqramların interfeys hissəsində elanı zamanı, onların yalnız başlıqları verilir. *Məsələn*,

```
Unit Cmplx;
Interface
type complex=record
                re,im:real
                  end;
procedure Addc(x,y:complex;var z:complex);
procedure Mulc(x,y:complex;var z:complex);
```

İndi əgər əsas proqrama **Uses Cmplx;** bölməsini daxil etsək, proqramda Cmplx modulunun Complex tipindən və **Addc, Mulc** prosedurlarından istifadə etmək olar.

Yerinə yetirilən hissə **IMPLEMENTATION** işçi sözü ilə başlayır və interfeys hissədə elan edilmiş alt proqramların təsvirindən ibarətdir. Burada eyni zamanda modul üçün lokal olan obyektlər – köməkçi tiplər, sabitlər, dəyişənlər və operatorlar bölməsində istifadə edildiyi halda nişanlar elan edilir. Modulun interfeys hissəsində elan olunmuş alt proqramın təsvirindən əvvəl, yerinə yetirilən hissədə onun başlığı verilir. Bu başlıqda formal dəyişənlərin siyahısını və funksiya üçün nəticənin tipini verməmək də olar, çünki onlar artıq interfeys hissəsində təsvir edilmişdi. Lakin əgər alt proqramın başlığı tam şəkildə, yəni formal parametrlərin siyahısı və nəticənin elanı ilə gətirilirsə, onda o, interfeys hissədə elan edilmiş başlıqla üst-üstə düşməlidir. Məsələn,


```
Unit Cmplx;
Interface
Type complex=record
              re,im:real;
                end;
procedure Addc(x,y:complex;var z:complex);
Implementation
procedure Addc;
begin z.re:=x.re+y.re;z.im:=x.im+y.im
end;
end.
```

Operatorlar bölməsi modulu sona çatdıran bölmədir. Bu bölmə onu başlayan **BEGIN** sözü ilə birlikdə modulda verilməyə də bilər və ya boş ola bilər. Bölmə boş olduqda **BEGIN** sözündən sonra modulun sonu əlaməti (**END** sözü və nöqtə işarəsi) gəlir. Bu bölmədə proqramın müəyyən fraqmenti olan operatorlar yerləşdirilir. Bu operatorlar əsas proqrama idarəetmə verilənə qədər yerinə yetirilirlər. Məsələn, burada lazımi fayllar açılır, digər kompüterlərlə əlaqə yaradılır və s.

Misal. Kompleks ədədlər üçün hesab əməllərini realizə edən modul quraq.

```
Unit cmplx;
Interface
type complex =record
                re,im:real;
                    end;
```

```
procedure Addc(x,y:complex;var z:complex);
procedure Subc(x,y:complex;var z:complex);
procedure Mulc(x,y:complex;var z:complex);
procedure Divc(x,y:complex;var z:complex);
const c;complex=(re: 0.1; im:-1);
Implementation
procedure Addc;
begin z.re:=x.re+y.re;z.im:=x.im+y.im end;
procedure Subc;
begin z.re:=x.re-y.re;z.im:=x.im-y.im end;
procedure Mulc;
begin z.re:=x.re*y.re-x.im*y.im;
z.im:=x.re*y.im+x.im*y.re end;
procedure Divc;
var zz:real;
begin zz:=sqr(y.re)+sqr(y.im);
z.re:=(x.re*y.re+x.im*y.im)/zz;
z.im:=(x.re*y.im-x.im*y.re)/zz end;
end.
```

Bu modulun mətnini cmplx.pas faylında yerləşdirib, proqramda bu moduldan istifadə etmək olar. *Məsələn,*

```
Uses cmplx;
var a,b,c:complex;
begin a.re:=1;a.im:=1;b.re:=1;b.im:=2;
            addc(a,b,c);
writeln(c.re:5:1,c.im:5:1,'i');
            subc(a,b,c);
writeln(c.re:5:1,c.im:5:1,'i');
            mulc(a,b,c);
writeln(c.re:5:1,c.im:5:1,'i');
            divc(a,b,c);
writeln(c.re:5:1,c.im:5:1,'i');
end.
```

Turbo Pascal dilində tərkibində çoxlu sayda müxtəlif tiplər, sabitlər, prosedur və funksiyalar olan səkkiz standart modul var. Bu modullar: system, dos, crt, printer, graph, overlay, turbo3 və graph3 modullarıdır. Bunlardan graph, turbo3 və graph3 modulları

ayrıca TPU tipli fayllarda yerləşdirilib, qalan modullar isə turbo.tpl faylına daxildir. Modullardan yalnız system modulu ixtiyari proqrama avtomatik olaraq qoşulur, yerdə qalan modullardan isə onların adları uses bölməsində verildikdən sonra istifadə etmək olur.

System moduluna standart Pascal dilinin bütün prosedur və funksiyaları, eləcə də digər standart modullara daxil olmayan prosedur və funksiyalar (məsələn, **inc,dec,getdir** və s.) aiddir.

Printer modulu mətnlərin printerə çıxarılmasını təmin edir. Bu modulda **text** tipli, **lst** fayl dəyişəni müəyyən edilir və **prn** məntiqi qurğusu ilə əlaqələndirilir. Məsələn, bu modul qoşulduqdan sonra aşağıdakı proqram yerinə yetirilə bilər:

```
uses printer;
begin writeln(lst,'Turbo Pascal')end.
```

Crt modulu ekranın mətn rejimini idarə edən prosedur və funksiyalardan ibarətdir. Bu modula daxil olan alt proqramların köməyi ilə kursoru ekranın ixtiyari mövqeyinə çıxarmaq, çıxarılan simvolların rəngini dəyişdirmək, pəncərələr yaratmaq olur.

Graph moduluna ekranın qrafik rejiminin idarə edilməsi üçün çoxsaylı tip, sabit, prosedur və funksiyalar daxildir. **Graph** moduluna daxil olan alt proqramlar vasitəsilə müxtəlif qrafik təsvirlər qurmaq və ekrana standart və ya istifadəçinin yaratdığı şriftlərlə yazıları çıxarmaq olur.

Dos modulunda **MS DOS** əməliyyat sisteminin imkanlarından istifadə üçün şərait yaradan prosedur və funksiyalar cəmlənib.

Overlay modulu mürəkkəb proqramların qurulmasında istifadə edilir.

Turbo3 və **graph3** modulları isə əvvəlki Turbo Pascal **3.0** versiyası ilə uyğunlaşdırılma üçün tətbiq olunur.

5.17. Obyektlər

Obyekt yönümlü proqramlaşdırma obyektlərə yeni xassələr verən üç əsas prinsipə əsaslanır. Bu prinsiplər inkapsulasiya, varislik və polimorfizmdir.

İnkapsulasiya verilənlər və onların emal alqoritmlərinin bir vahid tam şəkildə birləşdirilməsidir. Obyekt yönümlü proqramlaşdırılma (OYP) daxilində verilənlər obyektin sahələri, alqoritm isə obyekt üsulları adlandırılır. İnkapsulyasiya obyekti xarici mühitdən maksimal dərəcədə təcrid etməyə imkan verir. Bu isə yaradılan proqramların etibarlılığını əhəmiyyətli dərəcədə artırır, çünki obyektdə lokallaşdırılmış alqoritmlər proqramla verilənlərin nisbətən kiçik həcmləri ilə mübadilə edirlər, həm də bu verilənlərin tip və sayı adətən ciddi şəkildə nəzarət altında saxlanılır.

Varislik, obyektlərin öz varislərini yaratmaq xassəsidir. Varis-obyekt avtomatik olaraq valideyn-obyektdən bütün sahə və üsullara varis olur. Bundan əlavə obyektləri yeni sahələrlə tamamlayıb, valideyn üsullarını yeniləri ilə əvəz edə (üstələyə) və ya tamamlaya bilir. Varislik prinsipi obyektlərin xassələrinin modifikasiyası problemini həll edir. Obyektlərlə iş zamanı istifadəçi adətən konkret məsələnin həlli üçün öz xassələrinə görə ən uyğun obyekt seçir, sonra isə ondan bu obyektin bacarmadığı işləri yerinə yetirən bir və ya bir neçə varis yaradır.

Polimorfizm – qohum obyektlərin (yəni bir ümumi valideyni olan) mahiyyəti üzrə oxşar problemləri müxtəlif üsullarla həll etmək qabiliyyətidir. Obyekt yönümlü proqramlaşdırmada obyektin xassələri, bu obyektə daxil olan üsularla müəyyən olunur. Obyektin varislərində istifadəçi bu və ya digər üsulun alqoritmlərini dəyişdirməklə, həmin varislərə, onların valideynlərində olmayan xassələr verə bilər. Üsulu dəyişdirmək üçün varisdə eyni adlı üsulu elan edib, onda lazım olan işləri aparmaq lazımdır. Nəticədə valideyn-obyekt və varis-obyektdə müxtəlif alqoritmik əsası olan və deməli, obyektlərə müxtəlif xassələr verən eyni adlı üsullar olacaqdır. Bu obyektlərin poliformizmi adlanır.

Turbo Pascal dilində obyektlərin yaradılması üçün objekt, constructor, destructor işçi sözlərindən və private, public, virtual standart əmrlərindən istiafdə edilir. Bunlardan objekt işçi sözü obyektlərin təsviri üçün istifadə edilir. Obyektin təsviri, tiplərin

təsviri bölməsində verilməlidir. *Məsələn,*

```
type obyekt=object
     <obyektin sahələri>
     <obyektin üsulları>
end;
```

Əgər obyektin hər hansı bir valideyndən törənibsə, valideynin adı object işçi sözündən sonra dərhal yumru mötərizə daxilində verilir:

```
type sonobyekt=object(obyekt)
 <obyektin sahələri>
 <obyektin üsulları>
end;
```

Hər bir obyektin ixtiyari sayda varisi ola bilər, lakin yalnız bir valideyni olur. Obyektlərin qurulmasını, ekranda müxtəlif qrafik təsvirlər (nöqtə, çevrə, xətt, kvadrat) qurub, onları ekran boyunca hərəkət etdirilməsini təmin edən proqramın qurulması misalı üzərində araşdıraq. Bunun üçün əvvəlcə **TGobyekt** adlı valideyn-obyekt yaradaq. Həmin obyektdə yerdə qalan obyektlər üçün ümumi olacaq sahə və üsullar inkapsulyasiyadan keçəcəkdir:

```
type TGobyekt=object
Private
x,y:integer;color:word;
Public
Constructor Init(ax,ay:integer;ac:word);
procedure Draw(ac:word);Virtual;
procedure Show;
procedure Hide;
procedure MoveTo(dx,dy:integer);
end;
```

Sonradan nöqtə, xətt, çevrə və düzbucaqlının xassələrini realizə edən, **TGobyekt** obyektinin varislərini yaratmaq lazımdır. Bu qrafik obyektlərdən hər biri ekrandakı mövqeyi (**x** və **y** sahələri) və rəngi (**color** sahəsi) ilə xarakterizə olunur. Burada **TGobyekt** abstrakt obyekti konkret qrafik fiqurla əlaqəli deyil. Bu obyektin özündə bütün ümumi sahələri və üsulları birləşdirir, o, digər obyektlər üçün valideyn rolunu oynayır. Private əmri ob-

yektin təsvirində gizlədilmiş sahə və üsulların açılmasını təmin edir. **Public** əmri isə **private** əmrini ləğv edir, buna görə də **public** əmrindən sonra gələn obyekt elementləri ixtiyari proqram vahidində istifadə edilə bilər. **Private** və **public** əmrləri bir obyekt daxilində ixtiyari qaydada növbələşə bilər.

Sahələrin təsviri adi dəyişənlərin təsvirindən fərqlənmir. Sahə kimi verilənlərin ixtiyari strukturlarından, o cümlədən digər obyektlərdən də istifadə etmək olar. Obyekt-yönümlü proqramlaşdırmada üsulların təsviri üçün Turbo Pascal dilinin standart prosedur və funksiyalarından, eləcədə xüsusi prosedurlardan-konstruktor və destruktorlardan istifadə edilir. Konstruktorlar konkret obyektin yaradılması üçün istifadə edilir, belə ki, obyekt-verilənlər tipidir, yəni onun əsasında ixtiyari sayda obyekt tipli verilənlər yaratmaq olur. Konstruktorun başlığında procedure sözü əvəzinə **constructor** işçi sözündən istifadə olunur. Onun köməyi ilə virtual adlanan üsullar cədvəli yaradılır. Obyektdə virtual üsullar yoxdursa, burada konstruktor iştirak edə bilməz, əksinə üsullardan heç olmasa biri virtual təsvir edilibsə, obyektin tərkibinə ən azı bir konstruktor daxil olmalıdır və konstruktora müraciət, ixtiyari virtual üsula müraciətdən əvvəl gəlməlidir. Konstruktor obyekt sahələrini konkret qiymətlərlə təmin edir. Qeyd edək ki, eyni bir obyektin ayrı-ayrı nümunələri bir-birindən obyekt sahələrinin tərkibi ilə fərqlənir, lakin onların hamısı eyni bir obyekt üsullarından istifadə edirlər. Obyektin bütün xassələrini təsvir etmək üçün, obyekt üsullarının tərkibini açmaq, yəni uyğun prosedur və funksiyaları təsvir etmək lazımdır. Üsulların təsviri Turbo Pascal dilindəki adi qayda ilə təsvirlər bölməsində obyektin təsvirindən sonra verilir. *Məsələn*,

```
type TGobyekt=object
.....
end;
Constructor TGobyekt.Init;
begin x:=ax;y:=ay;color:=ac end;
procedure TGobyekt.Draw;
begin
end;
Procedure TGobyekt.Show;
begin Draw (color)end;
```



```
procedure TGobyekt.Hide;
begin Draw(GetBkColor)end;
procedure TGobyekt.MoveTo;
begin Hide;x:=x+dx;y:=y+dy;Show end;
```

Qeyd edək ki, üsulların təsviri zamanı üsulun adından əvvəl obyektin adı verilir, yəni üsulun qurma adından istifadə edilir.

Misal. Surət və məxrəcin ən böyük ortaq bölənini tapmaq, ixtisar etmək və natural tərtib üsulları olan adi kəsr obyektini təsvir edək.

```
type natur=1..32767;
frac=record p:integer; q:natur end;
drob=object a:frac;
procedure vvod;
procedure nod(var c:natur);
procedure sokr;
procedure stepen(n:natur; var c:frac);
procedure print;
end;
procedure drob.nod;
var m,n:natur;
begin m:=abs(a.p); n:=a.q;
while m<>n do
if m<>n then if m mod n<>0 then m:=m mod n
   else m:=n else if n mod m<>0 then n:=n
mod m
   else n:=m; c:=m end;
procedure drob.sokr;
var n:natur;
begin if a.p<>0 then begin drob.nod(n);
      a.p:=a.p div n; a.q:=a.q div n end
                  else a.q:=1 end;
procedure drob.stepen;
var i:natur;
begin c.p:=1; c.q:=1;
for i:=1 to n do begin c.p:=c.p*a.p;
                       c.q:=c.q*a.q
                 end; end;
procedure drob.vvod;
```

```
begin write ('Surat:');readln(a.p);
          write ('Maxrac:')readln(a.q);
end;
procedure drob.print;
begin writeln(a.p,'/',a.q)end;
var z:drob;f:frac;
begin z.vvod;z.print;z.sokr;z.print;
z.stepen(4,f);writeln(f.p,'/',f.q)
end.
```

5.18. Turbo Pascal dilinin qrafik imkanları

Turbo Pascal dilinin 4.0 versiyasından başlayaraq, onun tərkibinə GRAPH qrafik alt proqramlar kitabxanası daxil edilib. Burada qrafik ekranın idarə edilməsi üçün 50-dən çox prosedur və funksiyalar var. Kompüter işə salınarkən onun standart vəziyyəti mətn rejiminə uyğundur, buna görə də kompüterin qrafik imkanlarından istifadə edən ixtiyari proqram displey adapterin qrafik iş rejimini aktivləşdirməlidir. Proqram işini sona çatdırdıqda kompüter yenidən mətn rejiminə qayıdır. Qrafik prosedurların konkret adapterlə işi lazımi qrafik drayverin qoşulması ilə tənzimlənir. Drayver kompüterin bu və ya digər texniki vasitələrinin idarə edilməsini təmin edən xüsusi proqramdır. Qrafik drayver displey adapterini qrafik rejimdə idarə edir. Ən geniş yayılmış adapterlərin qrafik iş rejimlərinin xarakteristikasına baxaq:

1) **CGA** (**Color Graphics Adapter** – rəngli qrafik adapter) – 5 qrafik rejimə malikdir. Bunlardan 4 rejim ekranın zəif imkanlarına (320 piksel üfüqi və 200 piksel şaquli istiqamətdə, yəni 320×200) uyğundur və palitra ilə məhdudlaşdırılan rənglər qrupu ilə fərqlənir. Hər bir palitra işıq saçmayan qara rənglə birlikdə 4 rəngdən ibarədir. Bunlar: palitra 0 (açıq yaşıl, al qırmızı, sarı), palitra 1(açıq firuzəyi, açıq qırmızı, ağ), palitra 2 (yaşıl, qırmızı, qəhvəyi) və palitra 3 (firuzəyi, bənövşəyi, açıq boz). Beşinci rejim 640×200 yüksək imkanlara uyğundur, lakin palitrası cəmi iki rəngdən ibarətdir.

2) **EGA** (**Enhanced Graphics Adapter** – gücləndirilmiş qrafik adapter) adapteri CGA adapterinin bütün qrafik rejimlərini də dəstəkləyir. Bundan əlavə burada aşağıdakı rejimlər də

mümkündür: zəif imkanlı rejim (640×200, 16 rəng, 4 səhifə) və yüksək imkanlı rejim (640×350, 16 rəng, 1 səhifə). Bəzi modifikasiyalarda monoxrom rejimdən də (640×350, 1 səhifə, 2 rəng) istifadə edilir.

3) **MCGA** (Multi-**Color Graphics Adapter** – çox rəngli qrafik adapter) adapteri CGA adapteri ilə uyğunlaşıb və bundan əlavə daha bir rejimə (640×480, 2rəng, 1 səhifə) malikdir.

4) **VGA** (**Video Graphics Array** – qrafik video massiv) adapteri **CGA** və **EGA** adapterlərinin rejimlərini dəstəkləyir və onları yüksək imkanlı rejimlə (640×480, 16 rəng, 1səhifə) tamamlayır.

İndi isə qrafik prosedur və funksiyalara baxaq:

1) **InitGraph** proseduru adapterin qrafik rejimini iş vəziyyətinə gətirir. Onun ümumi şəkli aşağıdakı kimidir:

```
procedure InitGraph (var Driver,
Mode:integer; path:string);
```

burada **Driver-integer** tipli dəyişən olub, qrafik drayverin tipini təyin edir, **Mode-integer** tipli dəyişən olub, qrafik adap-terin iş rejimini təyin edir, **Path-string** tipli ifadə olub, drayver faylının adını və ona gedən yolu göstərir. Bu prosedurdan istifadədən əvvəl disk informasiya daşıyıcılarının birində lazımi qrafik drayver olduğu fayl verilməlidir. Prosedur bu drayveri operativ yaddaşa yükləyib, adapteri qrafik iş rejiminə keçirir. Drayverin tipi, qrafik adapterin tipinə uyğun gəlməlidir. Drayver tipini göstərmək üçün modulda aşağıdakı sabitlər təyin edilib:

```
const detect=0;CGA=1; MCGA=2; EGA=3;
EGA64=4; EGAMono=5; IBM8514=6; HercMono=7;
ATT400=8; VGA=9; PC3270=10;
```

Adapterlərin çoxu müxtəlif rejimlərdə işləyə bilir. Adapterə lazım olan iş rejimini vermək üçün Mode dəyişənindən istifadə edilir. Məsələn, **CGA.BGI** drayveri **C** diskindəki **TP\BGI** kataloqunda yerləşirsə və palitra 2 ilə 320×200 iş rejimini daxil edirsə, onda prosedura müraciət aşağıdakı kimi ola bilər:

```
Uses Graph;
var Driver,Mode:integer;
begin Driver:=CGA;Mode:=CGAC2;
```

```
InitGraph(Driver,Mode,'C:\TP\BGI');
```

Əgər kompüterin adapter tipi məlum deyilsə və ya əgər proqram ixtiyari adapterlə iş üçün nəzərdə tutulubsa, prosedura drayver tipini avtomatik təyin edilməsi tələbi olan müraciət yerinə yetirilir:

```
Driver:=Detect;
InitGraph (Driver,Mode,'C:\TP\BGI');
```

2) **GraphResult** funksiyası Integer tipli qiymət qaytarır, bu qiymət qrafik prosedurlara axırıncı müraciətin nəticəsini bildirir. Belə ki, əgər səhv yoxdursa, funksiya sıfır qiymətini, əks halda isə mənfi qiymət qaytaracaqdır.

3) **Function GraphErrorMsg (Code:integer): string;** funksiyası string tipli qiymət qaytarır ki, bu qiymətdə səhv koduna uyğun mətni məlumat verilir. Burada **Code-GraphResult** funksiyasının qaytardığı səhv kodudur.

4) **Procedure CloseGraph;** proseduru adapterin qrafik rejimindəki işini başa çatdırır və ekranın mətni iş rejimini bərpa edir.

5) **Procedure RestoreCRTMode;** proseduru mətn iş rejiminə qısa müddətli qayıdışı təmin edir.

6) **Function GetGraphMode:integer;** funksiyası Integer tipli qiymət qaytarır və bu qiymətdə qrafik adapterin təyin olunmuş iş rejiminin kodu verilir.

7) **Procedure SetGraphMode(Mode:Integer);** proseduru adapterin yeni qrafik rejimini təyin edir. Burada **Mode**-təyin edilən rejimin kodudur.

8) **Procedure DetectGraph (var Driver, Mode: Integer);** proseduru drayverin tipini və onun iş rejimini təyin edir. Burada **Driver**-drayverin tipi, **Mode**-isə iş rejimidir.

9) **Function GetDriverName:String;** funksiyası **string** tipli qiymət qaytarır ki, burada yüklənmiş qrafik drayverin adı olur.

10) **Function GetMaxMode:Integer;** funksiyası adapterin mümkün iş rejimlərinin sayını bildirən **Integer** tipli qiymət qaytarır.

11) **Function GetModeName (ModNumber:integer):string;** funksiyası ekranın imkanlarını və nömrəsinə

görə adapterin iş rejiminin adını bildirən **String** tipli qiymət qaytarır. Burada **ModNumber** rejimin nömrəsidir.

12) **Procedure GetModeRange (Drv:integer;var Min,Max:integer);** proseduru verilmiş qrafik adapterin mümkün iş rejimləri diapazonunu təyin edir. Burada **Drv**-adapter tipi, **Min** və **Max**-uyğun olaraq rejim nömrəsinin mümkün aşağı və yuxarı qiymətlərini bildirən **Integer** tipli dəyişənlərdir.

Bir çox qrafik prosedur və funksiyalar ekranın cari mövqe göstəricisindən isitfadə edirlər. Bu göstərici mətni kursordan fərqli olaraq ekranda görsənmir. Bu göstəricinin koordinatları və ümumiyyətlə qrafik ekrandakı ixtiyari koordinatlar, koordinatları (0,0) olan ekranın yuxarı sol küncünə nəzərən verilirlər. Beləliklə, ekranın üfüqi koordinatı soldan sağa doğru, şaquli koordinat isə yuxarıdan aşağıya doğru artır.

13) **GetMaxX** və **GetMaxY** funksiyaları cari iş rejimində uyğun olaraq üfüqi və şaquli istiqamətlərdə ekranın maksimal koordinatlarını bildirən **Word** tipli qiymətlər verir.

14) **GetX** və **GetY** funksiyaları göstəricinin uyğun olaraq üfüqi və şaquli isitqamətlərdəki cari koordinatlarını bildirən **Integer** tipli qiymətlər verir. Burada koordinatlar pəncərənin, pəncərə verilməyibsə ekranın yuxarı sol küncünə nəzərən təyin edilirlər.

15) **Procedure SetViewPort (x1,y1,x2,y2: integer;ClipOn:boolean);** proseduru qrafik ekranda düzbucaqlı pəncərə təyin edir. Burada **x1,y1,x2,y2**-koordinatları pəncərənin yuxarı sol **(x1,y1)** küncünün və aşağı sağ **(x2,y2)** küncünün koordinatlarıdır, **ClipOn** isə təsvirin pəncərədə yerləşməyən elementlərinin "kəsilməsini" təyin edən **Boolean** tipli ifadədir. Pəncərnin koordinatları həmişə ekranın yuxarı sol küncünə nəzərən verilir. Burada əgər **ClipOn** parametri **True** qiyməti alırsa, pəncərəyə sığmayan təsvir hissəsi kəsilib atılır, əks halda isə nəzərə alınmır.

16) **Procedure GetViewSettings (var ViewInfo: ViewPortType);** proseduru cari qrafik pəncərənin kəsilmə əlamətini və koordinatlarını qaytarır. Burada **ViewInfo, ViewPortType** tipli dəyişəndir. Bu tip **Graph** modulunda aşağıdakı kimi təyin edilib:

```
type ViewPortType=record
x1,y1,x2,y2:integer;
clip:boolean
end;
```

17) **Procedure MoveTo (x,y:integer);** proseduru göstərici üçün yeni cari mövqe təyin edir. Burada **x,y**-uyğun olaraq üfüqi və şaquli istiqamətlərdə göstəricinin yeni koordinatlarını müəyyən edir. Koordinatlar pəncərənin, pəncərə təyin edilməyibsə, ekranın yuxarı sol küncünə nəzərən təyin edilirlər.

18) **Procedure MoveRel (DX,DY:integer);** proseduru göstəricinin yeni vəziyyətini nisbi koordinatlarla təyin edir. Burada **DX,DY** yeni koordinatların uyğun olaraq üfüqi və şaquli istiqamətlərdə artımlarıdır. Artımlar gösətricinin bu prosedura müraciətdən əvvəl mövqeyinə nəzərən verilir.

19) **Procedure ClearDevice;** proseduru qrafik ekranı təmizləyir. Prosedur yerinə yetirildikdən sonra göstərici ekranın yuxarı sol küncündə yerləşdirilir, ekran isə **SetBkColor** prosedurunun verdiyi fon rəngi ilə rənglənir.

20) **Procedure ClearViewPort;** proseduru qrafik pəncərəni təmizləyir, pəncərə müəyyən edilməyibsə bütün ekranı təmizləyir. Göstərici isə pəncərənin yuxarı sol küncünə yerləşdirilir.

21) **Procedure GetAspectRatio(var x,y:Word);** proseduru ekranın tərəflərinin nisbətini qiymətləndirməyə imkan verən iki ədəd qaytarır. Burada **x,y-Word** tipli dəyişənlərdir. Bu dəyişənlərin qaytardığı qiymətlər qrafik ekranın tərəflərinin nisbətini piksellərlə qiymətləndirməyə imkan verir. Bu tapılan əmsallar düzgün həndəsi fiqurların çevrə, kvadrat və s. qurulmasında istifadə edilə bilər.

22) **Procedure SetAspectRatio(x,y:Word);** proseduru qrafik ekranın tərəflərinin nisbətinin miqyas əmsalını təyin edir. Burada **x,y** tərəflərin təyin edilən nisbətləridir.

23) **Procedure SetActivePage(PageNum:Word);** proseduru göstərilən videoyaddaş səhifəsini aktiv edir. Burada **PageNum** səhifə nömrəsidir.

24) **Procedure SetVisualPage(PageNum:Word);**

proseduru göstərilən nömrəli səhifəni əks etdirir. Burada **PageNum** səhifə nömrəsidir.

25) **Procedure PutPixel(x,y:integer;color:Word);** proseduru göstərilən koordinatlarda verilmiş rəngli nöqtə çıxarır. Burada **x,y** – nöqtənin koordinatları, **color** isə nöqtənin rəngidir.

26) **Function GetPixel(x,y:integer):Word;** funksiyası göstərilən koordinatlı pikselin rəngini bildirən **Word** tipli qiymət qaytarır. Burada **x,y**-pikselin koordinatlarıdır.

27) **Procedure Line(x1,y1,x2,y2:integer);** proseduru başlanğıc **(x1,y1)** və son **(x2,y2)** nöqtələrinin koordinatları göstərilmiş xətt çəkir. Burada **(x1,y1)** – xəttin başlanğıcının və **(x2,y2)** isə sonunun koordinatlarıdır.

28) **Procedure LineTo (x,y:integer);** proseduru göstəricinin cari mövqeyindən başlayaraq, prosedurda göstərilən yeni koordinatlarına qədər düz xətt çəkir. Burada **x,y**-göstəricinin mövqeyinin yeni koordinatları və deməli xəttin ikinci üç nöqtəsinin kordinatlarıdır.

29) **Procedure LineRel (DX,DY:integer);** proseduru göstəricinin cari mövqeyindən başlayaraq, onun prosedurda göstərilən koordinat artımlarının təyin etdiyi nöqtəyə qədər düz xətt çəkir. Burada **DX,DY**-göstəricinin yeni mövqeyinin koordinatlarını almaq üçün artımdır. **Line, LineTo, LineRel** prosedurları ilə xətt cari stil və rənglə çəkilir.

30) **Procedure Rectangle(x1,y1,x2,y2:integer);** proseduru yuxarı sol və aşağı sağ təpələrinin koordinatları ilə verilən düzbucaqlı çəkir. Burada **(x1,y1)**-yuxarı sol, **(x2,y2)** isə aşağı sağ təpələrinin koordinatlarıdır. Düzbucaqlı cari xətt stili və rəngi ilə çəkilir.

Misal. Aşağıdakı proqram ekranda bir-birinin daxilində yerləşdirilən 10 düzbucaqlı qurur.

```
uses graph, crt;
var d,r,e,x1,y1,x2,y2,dx,dy:integer;
begin d:=detect; initgraph(d,r,'');
e:=graphresult;
if e<>grok then writeln(grapherrormsg(e))
           else begin dx:=getmaxx div20;
```

```
dy:=getmaxy div 20;
for d:=0 to 9 do
rectangle(d*dx,d*dy,getmaxx-d*dx,
          getmaxy-d*dy);
if readkey=#0 then d:=ord(readkey);
closegraph
end
end.
```

31) **Procedure drawpoly (n:word; var points);** proseduru sınma nöqtələrinin koordinatları ilə verilən ixtiyari sınıq xətt çəkir. Burada **n** – hər iki kənar nöqtələr də daxil olmaqla sınıq xəttin təpə nöqtələrinin sayını göstərir, **Points** isə sınma nöqtələrinin koordinatlarından ibarət **PointType** tipli dəyişəndir. Sınma nöqtələrinin koordinatları **Word** tipli qiymətlər cütü ilə verilir. Onlar üçün modulda təyin edilmiş aşağıdakı tipdən isitfadə edilə bilər:

```
type PointType=record
                 x,y:Word
                       end;
```

32) **Procedure Circle(x,y:integer; r:word);** proseduru çevrə çəkir. Burada **x,y** – çevrənin mərkəzinin koordinatları, **r** – isə onun piksellərlə radiusudur.

33) **Procedure Arc (x,y:integer;bega,enda, r:word);** proseduru çevrə qövsü çəkir. Burada **x,y** –mərkəzin koordinatları, **bega,enda** – qövsün uyğun olaraq başlanğıc və son qövs dərəcələridir, **r** – radiusdur. Qövslər saat əqrəbinin hərəkətinin əksi istiqamətində çəkilir və dərəcələrlə göstərilir.

34) **Procedure GetArcCoords (var Coords:ArcCoordsType);** proseduru qövsün mərkəzinin, başlanğıcının və sonunun koordinatlarını qaytarır. Burada **Coords, ArcCoordsType** tipli dəyişəndir. Bu dəyişəndə prosedur qövsün mərkəzinin, başlanğıcının və sonunun koordinatlarını verir. Bu tip **Graph** modulunda aşağıdakı kimi təyin edilir:

```
type ArcCoordsType=record
          x,y:integer;
          xs,ys:integer;
          xe,ye:integer;
end;
```


35) **ProcedureEllips(x,y:integer; BegA,EndA, rx,ry:word)** proseduru ellips qurur. Burada **x, y** – mərkəzin koordinatları, **BegA, EndA** – uyğun olaraq qövsün başlanğıc və son bucaqlarıdır, **rx, ry** – ellipsin piksellərlə üfüqi və şaquli oxlarıdır.

36) **Procedure SetColor(Color:word);** proseduru çıxarılan xətt və simvollar üçün cari rəngi təyin edir. Burada **Color** – cari rəng nömrəsidir.

37) **Function GetColor:word;** funksiyası cari rəngin kodunu verən **Word** tipli qiymət qaytarır.

38) **Function GetMax Color:word;** funksiyası **SetColor** prosedurasına müraciət üçün istifadə edilə bilən maksimal rəng kodunu təyin edir.

39) **Procedure SetBkColor(Color:word);** proseduru fonun rəngini təyin edir. Burada **Color** fonun rəngidir.

40) **Function GetBkColor:Word;** funksiyası fonun cari rəngini göstərən **Word** tipli qiymət qaytarır.

41) **Procedure SetPalette(n:word; Color: ShortInt);** proseduru palitradakı bir rəngi yeni rənglə əvəz edir. Burada **n** – palitradakı rəngin nömrəsidir, **Color** isə yeni təyin edilən rəngin nömrəsidir.

42) **Procedure FloodFill(x,y:integer;Border: word);** proseduru ixtiyari qapalı fiquru cari rənglə rəngləyir. Burada **x, y** – qapalı fiqur daxilindəki ixtiyari nöqtənin koordinatları, **Border** isə sərhəd xəttinin rənginin nömrəsidir.

43) **Procedure Bar(x1,y1,x2,y2:integer);** proseduru ekranın düzbucaqlı oblastını cari rənglə rəngləyir. Burada **(x1,y1)** – rənglənən oblastın yuxarı sol küncünün, **(x2,y2)** isə aşağı sağ küncünün koordinatlarıdır.

44) **Procedure Bar3D(x1,y1,x2,y2,D:integer; T:boolean);** proseduru paralelepipedin üçölçülü təsvirini verir və onun ön hissəsini rəngləyir. Burada **(x1,y1)** – ön hissənin yuxarı sol küncünün, **(x2,y2)** isə aşağı sağ küncünün koordinatlarıdır, **D** – üçölçülü təsvirin piksellərlə dərinliyini təyin edir, **T** isə fiqurun yuxarı tilinin təsvir üsulunu təyin edir. Belə ki, **T** parametri **True** qiyməti aldıqda bu til əks etdirilir, əks halda isə

gizlədilir.

45) **Procedure FillPoly(N:Word;var Coords);** proseduru qapalı çoxbucaqlını xətlə əhatələyir və rəngləyir. Burada **N**-qapalı çoxbucaqlının təpələrinin sayı, **Coords**-isə **PointType** tipli dəyişən olub, təpə nöqtələrinin kordinatlarını verir.

46) **Procedure FillEllipse(x,y,rx,ry:integer);** proseduru ellipsi xətlə əhatələyib, onu rəngləyir. Burada **x,y**-ellipsin mərkəzinin koordinatları, **rx**, **ry** isə ellipsin piksellərlə üfüqi və şaquli radiuslarıdır.

47) **Procedure Sector(x,y:integer; BegA, EndA,rx,ry:Word);** proseduru ellips sektorunu çəkib, rəngləyir. Burada x,y-sektorun mərkəzinin koordinatları, **rx,ry**-sektorun piksellərlə üfüqi və şaquli radiusları, **BegA**, **EndA** isə elleps sektorunun uyğun olaraq başlanğıc və son bucaqlarıdır.

5.19. Turbo Pascal dilinin proqramlaşdırma mühiti

Turbo Pascal dilinin proqramlaşdırma mühiti TP adlı kataloqda yerləşdirilir. Burada Turbo Pascal-ı çağırmaq üçün **TURBO.EXE** faylını tapıb açmaq lazımdır. Nəticədə ekrana proqramlaşdırma sisteminin işçi stolu çıxarılır. Pəncərənin birinci sətrində mühitin baş menyusu adlanan bölmələr ardıcıllığı verilir. Menyu aşağıdakı bölmələrdən ibarətdir:

1) **File** (fayl) – fayllarla iş üçün və sistemdən çıxış üçün nəzərdə tutlub;

2) **Edit** (redaktə etmək) – sətirlərdə ola biləcək səhvlərin düzəldilməsi və müvəqqəti bufer yaddaşı ilə iş üçün nəzərdə tutulub;

3) **Search** (axtarmaq) – mətnin, prosedurun, funksiyanın və ya səhv olan yerin axtarılmasını təmin edir;

4) **Run** (iş) – proqramın yerinə yetirilməsini təmin edir;

5) **Compile** (kompilyasiya) – proqramın kompilyasiyasını təmin edir;

6) **Debug** (tənzimləmə) – proqramın tənzimlənməsini təmin edir;

7) **Tools** (alətlər) – köməkçi proqramların (utilitlərin) çağırılmasını təmin edir;

8) **Options** (variantlar) – proqramlaşdırma mühitinin qurulması;

9) **Window** (pəncərə) – pəncərələrlə işi təmin edir;

10) **Help** (kömək) – arayışlar bölməsinə müraciəti təmin edir.

Hər bir menyu bölməsi öz növbəsində alt bölmələrdən ibarətdir ki, onların köməyi ilə proqramlaşdırma mühitində proqramlarla işi təmin etmək olur. Proqram mətnlərinin redaktə etmə iş rejimindən, menyu bölmələrindən istifadə rejiminə keçmək üçün F10 düyməsindən, geriyə qayıtmaq üçün isə **ESC** düyməsindən istifadə olunur. Pəncərənin aşağı sətrində əsas funksional düymələrin təyinatı haqqında qısa arayışlar verilir. Ekranın yerdə qalan ikiqat çərçivəyə alınmış hissəsi proqram mətnlərinin daxil edilməsi və üzərində iş aparılmasını təmin edən redaktor pəncərəsidir. Bu pəncərənin yuxarı sətrində proqram mətnin oxunduğu disk faylının adı gətirilir ki, (yeni fayla **NONAMEOO.PAS** adı verilir). Turbo Pascal proqramlaşdırma mühitində eyni zamanda müxtəlif pəncərələrdə verilmək şərti ilə bir neçə proqramla işləmək olur. Mühit eyni zamanda redaktorun 9-a qədər pəncərəsində iş aparılmasına imkan verir. Turbo Pascal proqramlaşdırma mühitində redaktorun pəncərəsi ilə yanaşı tənzimləmə rejiminin pəncərəsi, proqramın yerinə yetirilmə nəticələrinin verilməsi pəncərəsi, arayışlar xidmətinin pəncərəsi də olur. *F1-F10* funksional düymələri, eləcə də onların *Alt, Ctrl və Shift* düymələri ilə kombinasiyaları Turbo Pascal proqramlaşdırma mühitinin idarə edilməsində isitfadə edilir. Bu düymələrdən bəzilərinin təyinatını verək:

1) *F1*-daxili arayışlar xidmətinə arayış üçün müraciəti təmin edir;

2) *F2*-redaktə edilən proqram mətnin disk faylına yazılışını təmin edir;

3) *F3*-disk faylındakı proqram mətnini redaktor pəncərəsinə çıxarır;

4) *F4*-tənzimləmə rejimində istifadə edilir: ondan proqramın yerinə yetirilməsini başlamaq və ya davam etdirmək üçün və kursorun durduğu sətrin yerinə yetirilməsindən əvvəl dayanmaq üçün istifadə olunur;

5) *F5*-aktiv pəncərəni bütün ekran boyunca açır;

6) *F6*-növbəti pəncərəni aktiv etmək üçün;

7) *F7*-tənzimləmə rejimində istifadə edilir: proqramın növbəti sətrinin yerinə yetirilməsi üçün, sətirdə prosedura (funksiyaya) müraciət varsa, bu prosedura girmək və onun birinci operatorunun yerinə yetirilməsindən əvvəl dayanmaq üçün istifadə edilir;

8) *F8*-tənzimləmə rejimində istifadə edilir: proqramın növbəti sətrinin yerinə yetirilməsi üçün, sətirdə prosedura (funksiyaya) müraciət varsa, bu prosedura girib, onu yerinə yetirmək üçün istifadə edilir;

9) *F9*-proqramı kompilyasiyadan keçirmək, lakin yerinə yetirməmək;

10) *F10*-baş menyunun köməyi ilə iş rejiminin dialoq seçiminə keçidi təmin edir;

11) *Ctrl-F9* kombinasiyası ilə proqramın yerinə yetirilməsi: proqramın kompilyasiyasının aparılması, onun operativ yaddaşa yüklənib yerinə yetirilməsi və sonda yenidən Turbo Pascal mühitinə qayıdışı təmin edilir;

12) *Alt-F5* isə redaktor pəncərəsini, proqramın yerinə yetirilmə nəticələrinin çıxarılması pəncərəsi ilə əvəz edir;

13) *Ctrl-Del* – redaktorun buferinin təmizlənməsini təmin edir;

14) *Ctrl-Ins* – ayrılmış blokun redaktorun buferində saxlanılmasını təmin edir;

15) *Alt-X* – Turbo Pascaldan çıxışı təmin edir;

16) *Alt-F3* – aktiv pəncərəni bağlayır.

Beləliklə, birinci növbədə *Ctrl-F9* ilə proqramın işi yoxlanılır və *Alt-X* ilə Turbo Pascal-dan çıxış yerinə yetirilir. *F2* və *F3* istifadəçiyə kataloqu ilə işi təmin edəcək və *Alt-F5* ilə proqramın yerinə yetirilmə nəticələrini ekrana çıxarmaq olur.

Turbo Pascal proqramlaşdırma mühitinin mətn redaktoru proqram mətnlərinin yaradılması və redaktə edilməsi üçün geniş imkanlar yaradır. Proqramlaşdırma mühitinin redaktə etmə rejimində olması, redaktor pəncərəsində kursorun olması ilə müəyyən edilir. Redaktə etmə rejimi Turbo Pascal yüklənən kimi avtomatik olaraq işə hazır olur. Bu rejimdən Turbo Pascal-ın ixtiyari digər rejiminə funksional düymələrin köməyi ilə və ya baş menyudan lazım olan rejimin seçilməsi ilə keçmək olur. Menyudan seçimə keçmək üçün F10, çıxış üçün isə *Esc* düyməsindən istifadə edilir. Proqram mətni klaviaturadan daxil edilir, hər sətrin sonunda *Enter* düyməsi sıxmaqla yeni sətrə keçid yerinə yetirilir. Hər bir fayldakı simvolların sayı 64535-i aşmamalıdır və Turbo Pascal-ın kompilyatoru proqram sətrindəki ilk 126 simvolu qəbul edir. Turbo Pascal redaktorunda istifadə edilən əsas əmrlər aşağıdakılardır:

1) *Page Up*– bir səhifə yuxarıya yerdəyişmə
2) *Page Down* – bir səhifə aşağıya yerdəyişmə
3) *Home* – cari sətrin əvvəlinə keçid
4) *End* – cari sətrin sonuna keçid
5) *Ctrl-Page Up* – mətnin əvvəlinə keçid
6) *Ctrl-Page Down* mətnin sonuna keçid
7) *BackSpace* – kursordan soldakı simvolu silir
8) *Delete* - kursordan sağdakı simvolu silir
9) *Ctrl-Y* – kursorun durduğu sətri silir
10) *Enter* – yeni sətirə keçidi, sətrin bölünməsini təmin edir.

Turbo Pascal proqramlaşdırma mühitindəki əsas iş qaydalarına baxaq. Qeyd etdiyimiz kimi Turbo Pascal yüklənən kimi proqramlaşdırma mühiti proqram mühitinin redaktə etmə rejiminə keçir. Bu rejimdə yeni proqram qurmaq və ya mövcud proqramı redaktə etmək olur. Proqram mətnləri mühitdən kənarda fayllarda saxlanılır. Turbo Pascal-da işi başa çatdırdıqdan sonra proqram mətnini növbəti dəfə istifadə etmək üçün disk faylında saxlamaq olur. Disk faylı ilə proqramlaşdırma mühiti arasında verilənlər mübadiləsini həyata keçirmək üçün F2 (fayla verilənləri yazmaq üçün) və *F3* (fayldan verilənləri oxumaq üçün) düymələrindən istifadə edilir. Yeni proqram yaradılarkən, mühit bu proqram mətninin hansı adlı faylda yerləşdiriləcəyini bilmədiyindən ona **NONAME00.PAS** standart adını verir. Proqram mətninin faylda saxlanması üçün *F2* düyməsini sıxmaq lazımdır. Bu anda mühit proqram adını yoxlayır, bu ad standart **Noname** adı olduqda bu adın dəyişdirilməsi haqqında **SAVE FILE AS** formalı sorğu verilir. Bu sorğudan aşağıda faylın adının daxil edilməsi üçün sahə yerləşir ki, burada lazım olan adı yığıb, *Enter* düyməsini sıxmaq lazımdır. Nəticədə proqram mətni həmin adlı faylda saxlanılır. Əgər fayl adında ad genişlənməsi verilməyibsə, mü-

hit fayla standart **Pas** ad genişlənməsi verir. Turbo Pascal-da işi sona çatdıran zaman burada redaktorda fayla yazılmamış proqram mətni qalırsa, mühit onu yaddaşda saxlanılması haqqında sorğu verir:

NONAME.PAS has been modified.Save?

Proqram mətnini faylda saxlamaq lazımdırsa, **Y(Yes** – Hə), lazım deyilsə, **N (No** – Yox) düyməsini sıxmaq lazımdır.

Proqram mətnini qurduqdan sonra onu yerinə yetirmək, yəni proqramı kompilyasiya etmək, ehtiyac varsa onu standart prosedur və funksiyalar kitabxanası ilə əlaqələndirmək, operativ yaddaşa yükləyib idarəetməni proqram mühitinə vermək lazımdır. Bu əməliyyat *Ctrl-F9* əmri ilə yerinə yetirilir. Əgər proqram mətnində hər hansı bir səhv yoxdursa, onda proqram mətni ardıcıl olaraq yerinə yetirilir və bu zaman ekrandakı kiçik bir pəncərədə kompilyasiyadan keçmiş sətirlərin sayı və operativ yaddaşın həcmi haqqında məlumatlar çıxır. Yüklənmiş proqrama idarəetməni verməmişdən əvvəl mühit ekranı təmizləyir, proqramın işləməsi pəncərəsini ekrana çıxarır, proqramın işi sona çatdıqdan sonra isə kompüterin idarə edilməsini yenidən üzərinə götürüb ekranda redaktorun pəncərəsini bərpa edir. Əgər proqramın yerinə yetirilməsinin hər hansı bir mərhələsində mühit səhv taparsa, o işi dayandırır, redaktorun pəncərəsini bərpa edir və kursoru səhv tapılmış proqram sətri üzərinə yerləşdirir. Bu zaman redaktorun yuxarı sətrində səhvin səbəbləri haqqında məlumat verilir. Səhvi düzəldib, proqramın yerinə yetirilməsini davam etdirmək lazımdır.

VI FƏSİL

DELPHİDƏ PROQRAMLAŞDIRMA

6.1. Delphi mühiti

Windows sistemi yarandıqdan sonra Microsoft firması birinci vizual proqramlaşdırma mühiti – Visual Basic sistemi yaratdı və Windows əməliyyat sistemində proqramlaşdırma MS DOS-a nisbətən xeyli asanlaşdı. 1995-ci ildə isə Borland firması Delphinin 1-ci versiyasını buraxdı və bir ildən sonra Delphi-nin 2,3 və 4-cü versiyaları buraxıldı. Məlumdur ki, MS-DOS-da ən geniş yayılan proqramlaşdırma dili (proqramlaşdırma sistemi) Turbo Pascal-dır, Windows-da isə Delphi hesab olunur.

Delphi-də proqramlar inteqrallaşmış mühitdə hazırlanır. Buna IDE (İntegrate Development Environment, rus dilində Интегрированная среда разработки) deyilir. IDE mühiti proqramçı ilə əlaqəni təşkil edir və müxtəlif idarəedici elementlər yerləşən bir sıra pəncərələrdən istifadə olunur. IDE vasitələrindən istifadə etməklə həm proqram kodunu, həm interfeys hissəsini yazmaq olar və onları idarəedici elementlərlə əlaqələndirmək olar. Delphi-də inteqrallaşmış mühit çox pəncərəli sistemdən ibarətdir. Sazlanmadan asılı olaraq IDE müxtəlif şəkillərdə ola bilər. Delphi yükləndikdən sonra IDE 4 pəncərədən ibarət olur (şəkil 6.1):

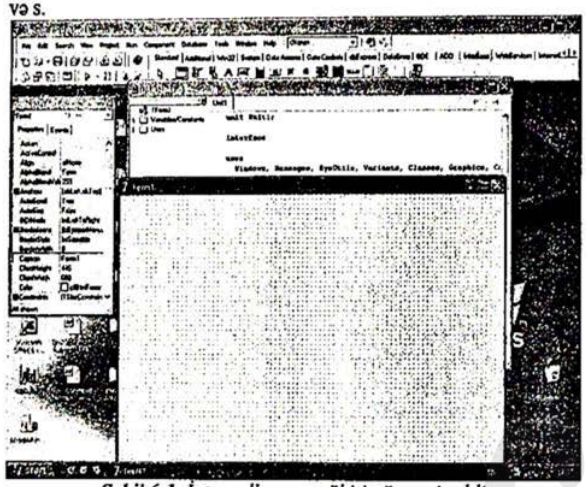
- əsas (baş) pəncərə (Delphi-də Project 1);
- obyektlərin inspektoru pəncərəsi (Object İnspector);
- formaların konstruktoru pəncərəsi (Form 1);
- kodların redaktoru pəncərəsi (Unit 1. PAS).

Bu pəncərələrin yerini dəyişmək, böyüdüb, kiçiltmək və ekrandan silmək olar. Çox pəncərənin olmasına baxmayaraq Delphi birsənədli mühitdir, yəni eyni zamanda bir proqramla işləmək olar. Proqramın proyektinin adı əsas pəncərənin baş hissəsində yazılır. Əsas pəncərə bağlandıqda Delphi ilə işləmək də kəsilir. Əsas pəncərə aşağıdakılardan ibarətdir:

- Baş menyu
- Alətlər paneli
- Komponentlər palitrası

Baş menyuda Delphi-nin funksiyalarına müraciət etmək üçün olan əmrlər yığımı yerləşir (*File*, *Edit*, *Search*, *View*, *Project*, *Run*, *Component*, *Database*, *Tools*, *Window*, *Help*). Alətlər paneli

baş menyunun alt sətrində yerləşir və 15 düymədən iba-rətdir. Bunlardan istifadə etməklə baş menyunun tez-tez istifadə edilən əmrlərini yerinə yetirmək olar. Məsələ, *File / Open* və ya *Run /Run* və s.



Şəkil 6.1. İnteqrallaşmış mühitin ümumi şəkli

Alətlər panelinin 5 növü vardır;
- *Standard* (*Standart*)
- *View* (*Baxış*)
- *Debug* (*Tənzimləmə*)
- *Custom* (*İstifadəçi*)
- *Desktop* (*İş stolu*)

Alətlər panelini və düymələrin tərkibini dəyişmək olar (Siçanın göstəricisini alətlər paneli oblastında yerləşdirməklə və siçanın sol düyməsini basmaqla). Bu qayda ilə komponentlər palitrasının da əks olunmasını dəyişmək olar.

Komponentlər palitrası əsas menyunun alt sətrində, əsas pəncərənin sağ hissəsində yerləşir. Komponentlər palitrasından proqramların formasını layihələşdirmək üçün istifadə edilir. Bütün komponentlər qruplara bölünür və hər qrup ayrıca bir səhifədə yerləşir. Komponentlər palitrasının aşağıdakı səhifələri vardır;

- *Standard* – standart
- *Additional* – əlavə
- *Win 32* - Windows-la 32 mərtəbəli əlaqə
- *System* – sistemin funksiyalarına müraciət
- *Data Access* – verilənlər bazasına müraciət (BDE vasitəsi ilə)
- *Data Control* – verilənləri idarə edən elementlərin yaradılması
- *ADO* – verilənlər bazası ilə Activ X-dən istifadə etməklə əlaqə
- *İnterbase* – eyniadlı verilənlər bazasına bilavasitə müraciətin təşkili
- *Midas* – paylanmış verilənlər bazası üçün proqramların hazırlanması
- *İnter Express* – Web serverin və paylanmış verilənlər bazasının klientinin eyni zamanda proqramı olan eyniadlı proqramların hazırlanması
- *İnternet* – İnternet şəbəkəsi üçün Web serverə proqramların hazırlanması
- *FastNet* – İnternet şəbəkəsinə müraciət protokollarının təmini
- *Decision Cube* – çoxölçülü analiz
- *Q Report* – hesabatların hazırlanması
- *Dialogs* – standart dialoq pəncərənin yaradılması
- *Win 3.1* - Windows 3.x-lə əlaqə
- *Samples* – misalların verilməsi
- *Activ x* – Activ x – komponentləri
- *Server* – ümumi Com serveri üçün VCL – mühiti.

Delphi-nin müxtəlif konfiqurasiyalarında bu komponentlərin heç də hamısı iştirak etmir. Komponentlər palitrasını sazlamaq üçün kontekst menyuda *Palette Properties* pəncərəsindən və ya baş menyuda *Component* dyüməsindən istifadə etmək lazımdır (şəkil 6.2).

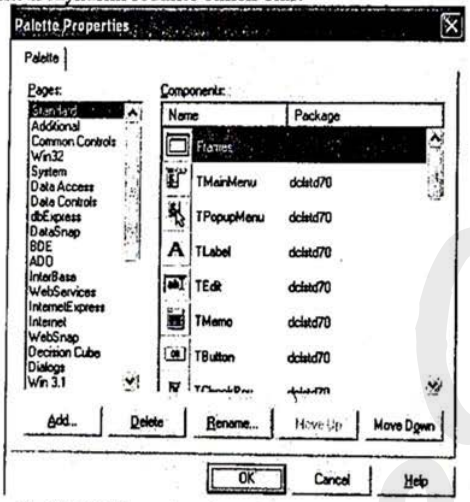
Add – əlavə etmək, *Delete* – silmək, *Rename* – ad vermək, *Move Up* (*Move Down*) – səhifəni və ya komponentləri yuxarıda (aşağıda) yerləşdirmək.

Hide – komponenti gizlətmək. Bu o vaxt olur ki, sağ tərəfdəki komponenti qeyd etdikdə *Delete* sözü *Hide* sözü ilə əvəz olunur.

Bu halda komponent səhifədə görsənmir, ancaq o bərpa oluna bilər.

Formaların konstruktoru pəncərəsi (başlıq – Form 1.) ekranın mərkəzində yerləşir. *Form 1* – pəncərənin adıdır. Burada formaların layihələnməsi yerinə yetirilir və bu formada komponentlər palitrasından zəruri elementlər yerləşdirilir.

Kodların redaktoru pəncərəsi (başlıq *Unit1.pas*) formaların konstruktoru pəncərəsinin arxasında yerləşir. Bu pəncərədə hazırlanan proqramın ilkin modulu yerləşir. Kodların redaktoru adi mətn redaktoru kimidir. Onun köməyi ilə modulun mətnini və digər mətn fayllarını redaktə etmək olar.



Şəkil 6.2. Komponentlər palitrasının dialoq pəncərənin ümumi görünüşü

Kodların bələdçisi pəncərəsi kodların redaktoru pəncərəsindən sağda yerləşir. Bu pəncərədə ağac şəklində modulun bütün obyektləri (dəyişən və proseduraları) əks olunur. Böyük modullarla işlədikdə belə obyektlərə bələdçi pəncərəsindən istifadə etməklə müraciət etmək əlverişli olur. Qeyd edək ki, bu pəncərə ekranda əks olunmaya da bilər (*Options / Environment / Automatically Show Explorer*).

Obyektlər inspektoru pəncərəsi (başlıq – *object İnspektor*) əsas pəncərənin altında ekranın sağ tərəfində yerləşir. Burada *Form1* – cari formasının obyektlərinin xassələri və hadisələri əks olunur. Bu pəncərəni *View / Object Inspector* əmri vasitəsi ilə ekranda əks etdirmək olar. Bu pəncərənin iki səhifəsi vardır: *Properties* (xassələr) və *Events* (hadisələr). *Properties* səhifəsində formaların konstruktoru pəncərəsindəki komponent haqqında informasiyalar yerləşir. Formaları layihələşdirən zaman komponentlərin xassələrini dəyişmək olar.

Events – səhifəsi komponentin yerinə yetirdiyi proseduranı təyin edir. Əgər hər hansı bir hadisəyə uyğun prosedura varsa, onda proqramın yerinə yetirilməsi zamanı bu hadisə baş verdikdə prosedura avtomatik olaraq çağırılır. Bu prosedura hadisənin emalı üçündür. Ona görə də onları hadisələri emal edən prosedurlar adlandırırlar.

Hər bir komponent özünə məxsus hadisələr və xassələr yığımına malikdir. Bu barədə növbəti paraqraflarda məlumat veriləcəkdir.

6.2. Proyektin xarakteristikaları

Delphi-də tərtib edilən hər bir proqram proyekt şəklində birləşdirilən bir neçə elementdən ibarətdir.

Proyektə aşağıdakı elementlər daxildir:

- Proyektin kodu (DPR – tipli)
- Formanın təsviri (DFM)
- Formanın Modulu (PAS)
- Modullar (PAS)
- Proyektin parametrləri (OPT)
- Resursların təsviri (RES)

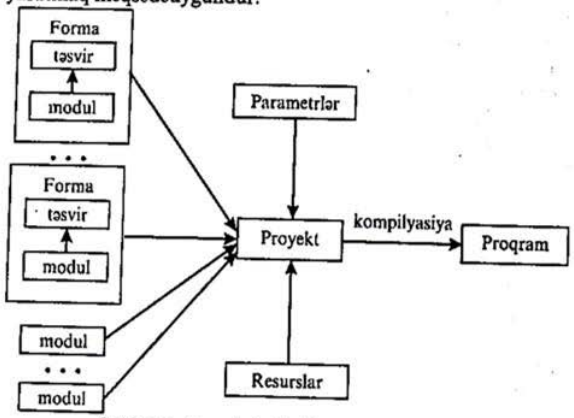
Şəkil 6.3-də proyektin ayrı-ayrı hissələri (fayllar) arasında əlaqə göstərilmişdir:

Göstərilən fayllardan əlavə digər fayllar da yarana bilər. Məsələn, `DPR` – faylların ehtiyyat surətləri `~DP` – tipli və ya `PAS` faylların ehtiyyat surətləri `~PA` tipli fayllar.

Delphi işə başlayan zaman avtomatik olaraq `Project1` – adlı yeni proyekt yaranır və bu ad əsas pəncərənin başlığında əks olunur. Bu proyekt `Form1` – adlı formaya malik olur. İstifadəçi bu proyekti və onun parametrlərini dəyişə bilər.

Adətən proyektin faylları bir kataloqda yerləşir və bu proyektə yeni formalar əlavə etdikdə faylların sayı artır. Buna

görə də hər bir yeni proyektin faylları üçün ayrıca kataloq yaratmaq məqsədəuyğundur.



Şəkil 6.3. Proyektin faylları arasında əlaqə

6.2.1. Proyekt kodu faylı (DPR)

Proyektin kodu faylı proyektin əsas elementidir və bu xüsusi bir proqramdır. Bu proqram aşağıdakı şəkildədir:

```
Program Project1;
USES
Forms,
Unit in 'Unit1.PAS' (Form1);
{ $R*.RES}
begin
Application.Initialize;
Application.Greate Form(T Form1, Form1);
Application.Run;
end.
```

Proyektin (proqramın) adı proyekt faylının adı ilə üst-üstə düşür və bu faylı diskdə saxlamaq lazım olduqda göstərilir.

`USES` bölməsində proqrama qoşulan `Forms` modulu göstərilir. Bu modul bütün proqramlar üçün zəruri və əsas olan bir moduldur. Burada həmçinin proyektin bütün formalarının modulları

göstərilə bilər. İlkin olaraq bu modul `Form1` formasının `Unit 1` moduludur.

`{ $R*.}` – direktivi resurslar faylını proyektə qoşur. Resurslar faylının adı proyekt faylının adı ilə (* - simvoluna görə) üst-üstə düşür. Bu fayldan əlavə `{ $R.}` direktivindən istifadə etməklə digər resurslar faylını da proyektə qoşmaq olar (resurslar faylında müxtəlif piktoqramlar, kursor, rəngli təsvirlər və s. ola bilər). Resurslar faylı kompilyasiya vaxtı proqrama qoşulur və bu vaxt onun həcmi artır. Odur ki, yaddaşa qənaət etmək üçün resursları dinamik olaraq proqrama qoşmaq lazımdır (`Load Form File` – metodundan istifadə etməklə).

Göründüyi kimi, proqramın proyekti üç əsas operatordan ibarətdir: inisializasiyanı yerinə yetirən, `Form1`-i yaradan və proqramı yerinə yetirən. Bunlar haqqında növbəti paraqraflarda məlumat veriləcəkdir.

İstifadəçi müəyyən bir əməliyyat yerinə yetirdikdə proyekt faylı avtomatik olaraq yaranır. Proqramçı bu faylda dəyişiklik edə bilər. Bu halda proyektin tamlığı pozula bilər. Təcrübəli proqramçılar proyektin tamlığını pozmamaq şərti ilə proyekt faylında dəyişiklik edə bilərlər.

Proyekt faylının kodunu redaktə etmək və baxmaq üçün ***Project/View Source*** (***Проект/просмотр источника***) əmrini vermək lazımdır.

6.2.2. Formalar faylı (DFM)

Proyektin tərkibində hər bir forma üçün avtomatik olaraq formanın təsviri faylı (DFM) və modul faylı (PAS) yaranır.

Formanın təsviri faylı – Delphi-nin resursudur. Burada forma və onun komponentlərinin xarakteristikaları yerləşir. Proqramçı formalar konstruktorundan və obyektlər İnspektorundan istifadə etməklə formanın təsviri faylını dəyişə və idarə edə bilər. Ona formalar konstruktorundan müraciət etmək olar. Zəruri halda bu fayla ekranda mətn şəklində baxmaq olar. Bunun üçün bu fayla aid olan formalar konstruktoru pəncərəsini bağlamaq və `File/Open` əmrlərini vermək lazımdır. Bu zaman kodların redaktoru pəncərəsində təsvir faylı görsənir və onun məzmununda dəyişiklik etmək olar. Məsələ, `Button1` düyməsi

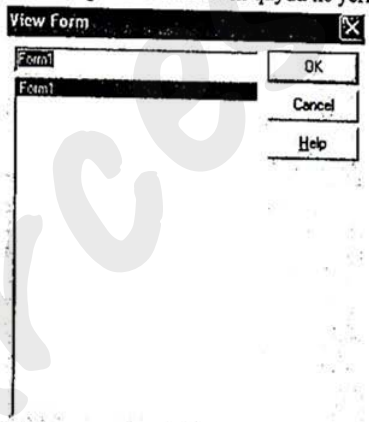
yerləşən formanın təsviri faylı göstərilmişdir. Bu düymə üçün OnClick – hadisəsinin emalçısı da yaradılmışdır:

```
Object Form1:TForm1
Left=192
Top=107
Width=544
Height=375
Caption='Form1'
Color=clBtnFace
Font.Charset=DEFAULT-CHARSET
Font.Color=clWindowText
Font.Height=-11
Font.Name='MS Sans Serif'
Font.Style=[ ]
Old Greate Order=false
Pixels Perinch=96
Text Height=13
object Button1:TButton
left=88
Top=120
Width=75
Height=25
Caption='Button1'
Taborder=0
OnClick=Button1Click
end
end
```

Misaldan göründüyü kimi təsvir faylında formanın özü də daxil olmaqla formanın bütün obyektləri və bu obyektlərin xassələri verilir. Hər bir obyektin tipi göstərilir (yəni sinfi). TForm1 – tipi bu formanın modulunda təsvir edilir. Əgər Caption='Form1' sətrində Caption='Birinci Forma' dəyişikliyi etsək, onda formanın başlığında "Birinci forma" sözü yazılar. Bu əməliyyatı obyektlər inspektoru pəncərəsində də etmək olar.

Formalar konstruktoru pəncərəsini yenidən açmaq üçün File/Close əmri ilə formaya uyğun olan təsvir faylını bağlamaq lazımdır. Sonra isə View əmrini vermək və ya <*Shift*>+<*F12*> – düymələrini sıxmaq lazımdır. *View Form* dialoq pəncərəsindən lazım olan formanı seçmək lazımdır (şək.6.4).

Eyni zamanda bir neçə formaları ekranda əks etdirmək olar. Dialoq pəncərənin bağlaması da məlum qayda ilə yerinə yetirilir.



Şəkil 6.4. Formanın konstruktoru pəncərəsinin açılması

6.2.3. Formanın modulu faylı (PAS)

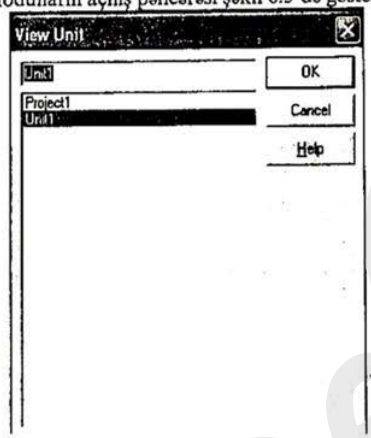
Bu faylda formaların sinfi təsvir edilir. Proyektin boş forması üçün bu modul aşağıdakı kodlardan ibarətdir:

```
Unit Unit 1;
interface
USES
Window,Messages,Sysutils,Classes,
Graphics,Controls,Forms,Dialogs, StdCtrls;
type TForm1=class(TForm)
private
{private declarations}
public
{public declarations}
end;
Var Form1:TForm1;
İmplementation
{$R*.DEM}
end.
```

Yeni komponent əlavə edən zaman Delphi avtomatik olaraq formanın modulu faylını yaradır. Susma prinsipinə görə proyektə TForm tipli yeni forma əlavə olunur ki, formada bu zaman heç bir komponent yerləşmir. Formada komponent yerləşdirən zaman və ya hadisələrin emalçısını yaradan zaman formalar modulunda uyğun dəyişiklik edilir. Bu dəyişikliyin bir hissəsi avtomatik olaraq aparılır, bir hissəsi isə proqramçı tərəfindən yerinə yetirilir. Proqramlaşdırma ilə əlaqədar olan bütün dəyişiklikləri proqramçı ancaq formalar modulunda edir.

Formalar modulunun mətni kodların redaktoru pəncərəsində əks olunur və onun köməyi ilə redaktə olunur.

Formalar modulunu açmaq üçün File/Open və ya dialoq pəncərədə View/Unit və ya <Ctrl>+<F12> əmrlərini vermək lazımdır. Modulların açılış pəncərəsi şəkil 6.5-də göstərilmişdir.



Şəkil 6.5. Modulların açılış pəncərəsi

Modulların açılış pəncərəsində proyekt faylını da seçmək olar. Lazım olan modulu seçdikdən sonra OK – düyməsini sıxdıqda onun mətni kodların redaktoru pəncərəsində ayrıca səhifədə əks olunur.

Modulun kompilyasiyası zamanı DCU genişlənməsinə malik olan fayl avtomatik olaraq yaradılır (modulun kodu faylı). Bu faylı proyektin bütün faylları yerləşən kataloqdan silmək olar.

Çünki növbəti kompilyasiya zamanı Delphi onu avtomatik olaraq yaradır.

Modullar faylı (PAS)

Proqramlaşdırma zamanı forma modullarından əlavə heç bir forma ilə əlaqədar olmayan modullardan da istifadə etmək olar. Onlar Object Pascal-ın qaydaları əsasında yaradılır və ayrıca fayllarda saxlanılır. Bu modulu qoşmaq üçün onun adını USES direktivində göstərmək lazımdır.

6.2.4. Resurslar faylı (RES)

Proyektin birinci saxlanması zamanı avtomatik olaraq adı proyekt faylının adı ilə eyni olan Resurslar faylı (RES) yaranır. Resurslar faylında aşağıdakılar ola bilər:

- Piktoqramlar;
- foto təsvirləri;
- krusorlar.

Resurslar faylı ilə işləmək üçün İmage Editor 3.0 qrafik redaktorundan istifadə edilir ki, redaktor *Tools/Image Editor* (*Средства / Редактор изображения*) əmri vasitəsi ilə çağırılır.

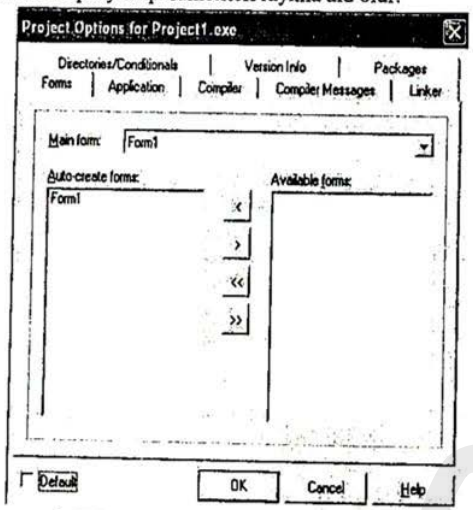
Resurslar faylı ierarxik struktura malikdir, resurslar ayrı-ayrı qruplarda birləşdirilir və hər bir qrupa ad verilir. Məsələ, proyektin piktoqramaları Icon qrupunda yerləşir və susmaya görə MAİNİCON adına malikdir. Yuxarıdakılardan əlavə aşağıdakı resurs fayllarından da istifadə olunur:

- Komponentlərin piktoqramları (DCR)
- Proqramların piktoqramı (ICO)
- Kursorlar (CUR)
- Müxtəlif şəkillər (BMP).

6.2.5. Proyektin parametrləri (OPT)

Proyektin parametrlərini təyin etmək üçün proyektin parametrlər pəncərəsindən istifadə etmək lazımdır (*Project/Options* və ya *Ctrl+Shift+F11* əmrlərindən istifadə emaklə). Parametrlər qruplara bölünür. Hər bir qrup ayrıca səhifədə yerləşir. Ayrı-ayrı parametrlərin qiymətləri təyin olunduqdan sonra Delphi avtomatik olaraq proyektin uyğun faylında dəyişiklik edir. Belə ki, Forms və Application səhifələrinin parametrləri pro-

yekt və resurs fayllarına, Compiler və Linker səhifələrinin parametrləri isə proyekt parametrləri faylına aid olur.



Şəkil 6.6. Proyektin parametrləri pəncərəsi

Proyekt parametrləri faylına aid misal:

```
{ Compiler}
A=1
B=0
C=1
D=1
E=0
...
```

Bu fayl mətn faylıdır və parametrlər və onların qiymətləri sətir-sətir yerləşir.

Proyektin parametrləri pəncərəsi şəkil 6.6-də göstərilmişdir.

6.3. Proyektin kompilyasiyası və yerinə yetirilməsi

Proyektin kompilyasiya prosesi yerinə yetirilməyə hazır olan fayl yaradır (EXE və ya dinamik yüklənən kitabxana – DLL). Kompilyasiyadan sonra alınan faylın adı proyekt faylının adı ilə

eyni olur. Əgər proqramın yerinə yetirilməsi zamanı digər fayllar da tələb olunarsa, onda həmin fayllar diskdə olmalıdır. Kompilyasiya prosesini Project/Compile <Project1> və ya <Ctrl+F9> əmrləri yerinə yetirir. Proyektin kompilyasiyası proyektin işlənməsinin ixtiyari mərhələsində yerinə yetirilə bilər. Bu isə yaranan proqram kodunun və formanın ayrı-ayrı komponentlərinin fəaliyyətlərinin yoxlanması üçün əhəmiyyətlidir.

Proyektin kompilyasiyası zamanı aşağıdakı əməliyyatlar yerinə yetirilir:

- Bütün modullar faylı kompilyasiya edilir və nəticədə hər bir fayl üçün DCU tipli fayl yaranır.
- Əgər modulda dəyişiklik edilərsə, onda bu modulun yazıldığı USES direktivindəki bütün modullar yenidən kompilyasiya edilir.
- Obyekt fayllarda (OBJ) dəyişiklik olunduqda da modullar yenidən kompilyasiya edilir.
- Bütün modullar kompilyasiya olunduqdan sonra proyekt faylı kompilyasiya olunur və proyekt faylının adı ilə eyni olan yerinə yetirilən fayl yaranır.

Kompilyasiyadan əlavə proyektin yığılması (сборка) prosesi də yerinə yetirilə bilər. Bu zaman proyektə daxil olan bütün fayllar onlarda dəyişiklik olmasından asılı olmayaraq kompilyasiya olunur. Proyektin yığılması üçün Project/Build <Project1><ilkin adı> - əmrini vermək lazımdır.

Proyektin yerinə yetirilməsini həm Delphi, həm də Window mühitində təmin etmək olar. Delphi mühitində RUN / RUN və ya <F9> əmrləri vasitəsi ilə. Bu zaman aşağıdakı xüsusiyyətləri nəzərə almaq lazımdır:

- Proqramın (proyektin EXE kodunun) 2-ci ekzemplyarını yerinə yetirmək olmaz.
- Yerinə yetirilmə prosesi qurtardıqdan sonra proyektin işlənməsini davam etdirmək olar.
- Proqramın yerinə yetirilməsi dövrə düşərsə RUN/ Program Reset və ya <Ctrl>+F2> əmrlərini vermək lazımdır.

Delphi mühitində proqramı otladka etmək üçün vasitələr vardır.

Proyektin yerinə yetirilməsi Windows mühitində digər proqramlar kimi bələdçinin (проводник) köməyi ilə olur.

6.4. Proqramın hazırlanması mərhələləri

Delphi vizual proqramlaşdırma sisteminə aiddir və bu sistem RAD (Rapid Application Development – Быстрая разработка приложений) adlanır. Delphi-də proqramın hazırlanması iki mərhələdən ibarətdir:

- o Proqramın interfeysini hazırlamaq;
- o Proqramın fəaliyyət göstərməsini, yəni işləməsini təmin etmək.

Proqramın interfeysində istifadəçi ilə proqram arasında qarşılıqlı əlaqə üsulları müəyyən edilir, formaların xarici görünüşü və proqramçının proqramı necə idarə etməsi təyin edilir. Komponentlərin formada yerləşdirilməsi yolu ilə formalar konstruktorunun interfeysi yaranır. Bu komponentlərə interfeys və ya idarəedici komponentlər deyilir.

Proqramın fəaliyyət göstərməsi proseduraları vasitəsi ilə təmin edilir. Bu proseduralar müəyyən bir hadisə baş verdikdə yerinə yetirilir. Bu hadisəyə uyğun olaraq hadisələrin emalçısı hazırlanır.

Beləliklə, proqramın hazırlanması prosesi komponentlərin formada yerləşdirilməsi və bu komponentlər üçün zəruri olan xassələrin verilməsi, hadisələrin emalçısının hazırlanmasından ibarətdir.

Delphi-də ən sadə proqram boş formaya uyğun olan proqramdır. Bu forma müəyyən mənada boş deyildir. Belə ki, burada formanın başlığı (`Form1`), maksimallaşdırmaq və minimallaşdırmaq, pəncərəni bağlamaq, sistem menyusunu çağırmaq, pəncərənin ölçüsünü dəyişmək üçün düymələr vardır. Delphi-nin birinci yüklənməsi zamanı formaların konstruktoru pəncərəsində məhz bu forma əks olunur. Bu boş formaya uyğun olan proqram avtomatik yaradılır və fəaliyyətsiz görsənir. Əslində bu boş pəncərə çoxlu işlər görür. Məsələ, o istifadəçinin siçan və klaviatura ilə bağlı hərəkətini gözləyir öz ölçüsünü dəyişmək, bağlanmaqla əlaqədar əmrləri yerinə yetirilir və s.

Sadə proqramın hazırlanması ilə əlaqədar olan yerinə yetirilən faylın uzunluğu 275 *Kb*-dir. Əgər `DLL` kitabxanasından istifadə edilərsə 20 dəfə az – 12,5 *Kb* yer tutur.

6.4.1. Proqram interfeysinin hazırlanması

Proqramın interfeysi komponentlər vasitəsi ilə təşkil olunur. Bu komponentləri proqramçı komponentlər palitrasından seçib, formada yerləşdirir. Proqram interfeysinin qurulması WYSIS WYG – prinsipinə əsaslanır.

Komponentlər iki yerə bölünür:

- vizual (görünən);
- vizual olmayan (sistem komponentləri).

Layihələşmə mərhələsində bütün komponentlər görünür. Bu bölgü yerinə yetirilmə mərhələsinə aiddir.

Vizual komponentlərə düymələr, siyahılar, çevricilər, formalar və s. aiddir. Bunlardan proqramı idarə etmək üçün istifadə edilir və ona görə də bu komponentlərə idarəedici komponentlər deyilir. Bu komponentlər proqram interfeysini əmələ gətirir.

Vizual olmayan komponentlər köməkçi əməliyyatları yerinə yetirir. Məsələ, `Timer` (saniyə ölçən), `Table` (verilənlər yığımı).

Proqram interfeysini yaradan zaman hər bir komponent üçün aşağıdakı əməliyyatlar yerinə yetirilir:

- o Komponentlər palitrasından komponentlər seçilir və formada yerləşdirilir;
- o Komponentin xassəsi dəyişdirilir.

Bu əməliyyatlar formalar konstruktoru, obyektlər İnspektoru və komponentlər palitrası pəncərələrindən istifadə etməklə yerinə yetirilir.

Komponentlərə misallar:`Label`, `Button`, `Memo`, `Edit` və s.

Hər bir komponenti formaya qoyduqda modullar faylında və təsvir (`DFM`) faylında Delphi avtomatik olaraq dəyişiklik edir. Hər bir yeni komponent üçün aşağıdakı sətir əlavə olunur:

```
<komponentin adı>:<komponentin tipi>
```

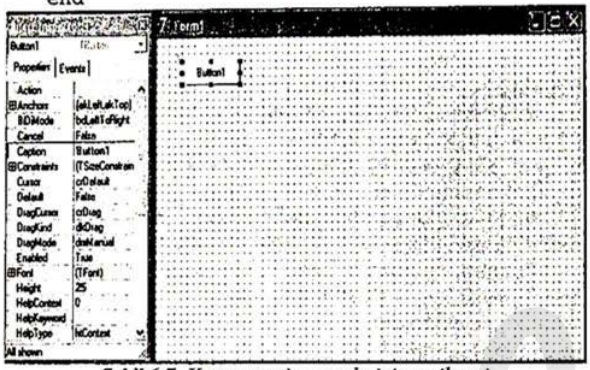
Məsələ: `Button1:TButton;`

`Edit1:TEdit;`

`Label1:TLabel;` – Modullar faylına bu sətirlər əlavə olunarsa, deməli formaya üç komponent gətirilmişdir: `Label1`, `Edit1`, `Button1`.

Komponentlər formaya qoyulduqda hər bir komponentə uyğun formanın təsviri faylında da dəyişiklik olur. Məsələn, `Button1:TButton` üçün təsvir faylında aşağıdakılar əlavə oluna bilər:

```
Object Button1: TButton
left=88
Top=120
With=75
Height=25
Caption=Button1
Tab Order = 0
end
```



Şəkil 6.7. Komponentin xassələrinin verilməsi

Komponentləri formaya qoyduqdan sonra siçanın köməyi ilə onun ölçüsünü və yerini dəyişmək olar. Formada komponentləri qeyd etdikdən sonra obyektlər inspektorunda onun xassələrini də vermək olar.

Məsələn, şəkil 6.7-də `Button1` komponentinin xassəsinin verilməsi göstərilmişdir.

Obyektlər İnspektorunda hər bir komponentə uyğun xassənin adları və onların susma prinsipinə görə qiymətləri vardır. Bu qiymətləri dəyişmək olar (Məsələ, `LABEL1`-in adını və s.). Obyektlər İnspektorunda xassələrin qiymətlərini vermək üçün xassələr redaktorundan istifadə edilir. Xassələr redaktoru aşağıdakılardan ibarətdir:

- Sadə (mətn) xassə – qiyməti adi sətir simvolları kimi daxil edilir və ya redaktə edilir. Məsələ: `Caption`, `left`, `Top` və s. (`Caption=Close` və s.)

- Sadalanan xassələr – açılan siyahıdan seçilir. Siyahı siçanın köməyi ilə açılır. Məsələ, `FormStyle`, `Visible`, `ModaResult`.

- Çoxluq qiymətli xassələr. Bu xassələrin qiymətləri təklif olunan çoxluqdan qiymətlərin kombinasiyası kimi seçilir. Belə xassənin adının sol tərəfində «+» – işarəsi yerləşir. Bu xassənin adının üzərində siçanın sol düyməsini iki dəfə sıxsaq əlavə siyahı açılar. Bu əlavə siyahıların sağ tərəfinə `True` və ya `False` yazmaqla xassələrin kombinasiyası seçilir. Məsələ, belə xassəyə `Border İcons`-u göstərmək olar.

- Xassə obyekt ola bilər və bu da öz növbəsində başqa alt xassələrə malik ola bilər. Bunların hər birini ayrıca redaktə etmək olar. Belə xassə üçün `Font`, `İtems`, `Lines` – istifadə etmək olar. Obyekt xassələrinin də sol tərəfində «+» işarə vardır. Sağ tərəfdə isə alt xassələrin qiymətləri olur. Bu qiymətlər çoxluq tipli xassələrdə olduğu kimi açılır.

Xassənin qiymətini həm `İnspector Object`-də, həm də proqramın yerinə yetirilməsi zamanı vermək olar. Bu hadisələrin emalçısını yaradan zaman edilir. Məsələ, `Button1`-in adını dəyişmək üçün:

`Button1.Caption:='Close';` yazmaq lazımdır.

Ancaq bu o qədər də əhəmiyyətli hesab edilmir. Çünki proqram yerinə yetirilməzdən əvvəl komponentin adı əvvəlki kimi qalır, yerinə yetirildikdə (`RUN`) komponentin adı dəyişir. Ancaq bəzi komponentlər vardır ki, onların xassələrinin qiymətini yerinə yetirilmə zamanı vermək olar. Məsələ, verilənlər yığımı ilə işlədikdə `Record Count` – yazıların sayı, şəkli çəkmə – `Canvas` və s. Belə xassələrin qiymətləri hadisələr səhifəsində `On Greate` – hadisələr emalçısının köməyi ilə yaradılır. Bu `TForm1. Form Greate` – adlı proseduranın formanın modulu faylında yaranmasına səbəb olur. Bu proseduranın gövdəsində xassənin qiyməti verilir: Məsələn, `Button1` komponenti Formaya qoyulubsa, onun adını yerinə yetirilmə mərhələsində dəyişmək üçün `Unit1. PAS` faylında

```
Procedure TForm1.Form Greate (sender:
                                    Tobject);
begin
Button1.Caption:='SES';
End;
```

proseduranın gövdəsinə bir sətir əlavə etmək lazımdır.

Proqram yerinə yetirildikdən sonra (`RUN`) `Button1` komponentinin üzərində «SES» – yazılacaq.

6.4.2. Proqramın fəaliyyət göstərməsinin və ya işləməsinin təmin edilməsi

Proqramın interfeys hissəsinin hazırlanmasının istənilən mərhələsində onu yerinə yetirmək olar. Kompilyasiyadan sonra da forma əvvəlki şəklində qalır və onu böyütmək, kiçiltmək, yerini dəyişmək olar. Proqramçı formada yerləşən komponentlər üçün bu və ya digər əməliyyatlara cavab verən reaksiya təyin etməlidir. Bu reaksiya proqramın fəaliyyətini təyin edir.

Tutaq ki, formada Button1 komponenti yerləşib. Bu komponent pəncərəni bağlamağa xidmət etməlidir. Əvvəlcə Obyektlər İnspektorundan istifadə edərək komponentə reaksiyaya uyğun ad veririk: `Caption:=Close`

Əgər bu komponenti siçan və ya klaviatura vasitəsi ilə sıxsaq heç bir əməliyyat yerinə yetiliməyəcəkdir. Çünki bu komponent üçün proqramçının hərəkətinə qarşı heç bir reaksiya təyin edilməmişdir. Belə reaksiya təyin etmək üçün hadisəni emal edən prosedura hazırlanmalıdır. Bu proseduraya hadisə baş verdikdə müraciət edilir. Hadisələri emal edən proseduru yaratmaq üçün Obyektlər İnspektorunda hadisələr (`Events`) səhifəsinə keçmək lazımdır. Komponentin üzərində siçanın sol düyməsini sıxan zaman `OnClick` – hadisəsi əmələ gəlir. Bu hadisəni emal edən proseduranı yaratmaq lazımdır. Bunun üçün `Onclick` hadisəsinin üzərində siçanın sol düyməsini iki dəfə sıxmaq lazımdır. Nəticədə *Kodların redaktoru* pəncərəsi ön plana keçir və kursor proseduranın gövdəsində elə yerdə durur ki, bu hissəyə proqramçı proqram kodu yazmalıdır. Bu proqram kodu `Button1` düyməsi sıxıldıqda yerinə yetirilməlidir. Məqsəd odur ki, bu düymə sıxıldıqda pəncərə bağlansın. Ona görə də prosedura daxilinə `Form1.Close` – əlavə edirik (`Beep` əlavə etsək səs (bip) – siqnalı verəcək). Bu halda formalar modulu aşağıdakı şəkildə olar:

```
Unit Unit 1;
İnterface
USES Windows,Messages,Sisutils,
Closes,Graphics,Controls,Forms,
Dialogs,Std Ctrls;
type TForm1=class(TForm)
Button1:TButton;
procedure Button1 Click(Sender: Tobject);
```



```
private
{private declaration}
public
{public declaration}
end;
Var Form1:TForm1;
implementation
{$R*DFM}
procedure TForm1.Button1 Click
(sender:Tobject);
begin
Form1.Close;
end;
end.
```

Bu modulda proqramçı ancaq `Form1.Close` sətrini yazır. Modulun yerdə qalan hissəsini Delphi avtomatik olaraq yaradır. Əgər bu sətrin yerinə `Beep` yazılarsa onda `Button1` düyməsi sıxıldıqda səs (bip) siqnalı verilər. Hadisənin emalçısı olan prosedurun adında `Onclick` – hadisəsinin `On` – söz önü götürülmüşdür: `Obyektin adı. Komponentin adı Click;` yəni:

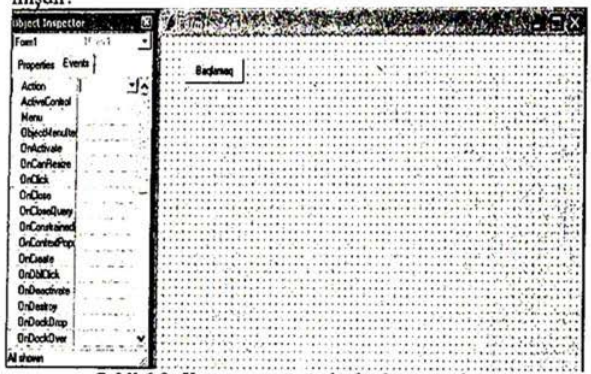
`TForm1.Button1Click`

Komponentin adını dəyişdikdə (`Object İnsPector` – vasitəsi ilə) – uyğun olaraq bütün `DFM` və `PAS` fayllarında dəyişiklik olar. Proyektin hazırlanması qurtardıqdan sonra `RUN` (və ya *F9*) vasitəsi ilə yerinə yetirilir. Yerinə yetirildikdən sonra ekranda yenə də Forma görsənir. Bu formada olan `Close` düyməsinin üzərində siçanın sol düyməsini sıxsaq pəncərə bağlanır (`Form1.Close`-a uyğun) (Əgər `Beep` olarsa səs siqnalı verər).

Digər komponentlərə uyğun olan hadisələr də oxşar olaraq yaradılır. Hadisələrin emalçısı proseduranı proqram kodundan (`Unit1.PAS`) silmək də olar. Bu zaman modulun kompilyasiyası və ya saxlanmasında bu proseduralar proyektin bütün fayllarından avtomatik olaraq silinir.

Lakin Formadan komponenti sildikdə ona uyğun olan hadisələrin emalçısı proseduası moduldan silinmir. Eyni bir proseduranı bir neçə hadisə və ya komponentlə əlaqələndirmək olar. Belə proseduraya ümumi emalçı deyilir və onunla əlaqədar olan hadisələr baş verdikdə çağrılır. Prosedura daxilində hansı komponent və ya hadisə üçün proseduranın çağrılması verilir.

Şəkil 6.8-də komponentlər üçün hadisələrin siyahısı verilmişdir.



Şəkil 6.8. Komponent üçün hadisələrin siyahısı.

6.5. İnteqrallaşmış mühitin vasitələri

Proqramın effektiv və əlverişli şəkildə işləməsini təmin etmək üçün Delphi-nin inteqrallaşmış mühitində müxtəlif vasitələr vardır. İstifadəçi İnteqrallaşmış mühiti idarə edə bilər, onun ayrı-ayrı parametrlərini dəyişə bilər məsələ, kodların redaktoru pəncərəsinin şrift və rəngini, redaktə olunan faylların avtomatik saxlanmasını və s. idarə edə bilər. Parametrlərin verilməsi `Environment Options` – dialoq pəncərəsində yerinə yetirilir (`Tools /Environment Options...`). Parametrlər qruplara görə birləşdirilmiş və ayrı-ayrı səhifələrdə yerləşdirilmişdir. Hər bir proyekt üçün Delphi mühitinin parametrləri `CFG` - tipinə malik olan faylda saxlanılır.

İnteqrallaşmış mühitin əsas vasitələrindən biri Proyektin Meneceridir (`Project Manager`). `Project Manager` – hazırlanan proqramın tərkib hissələrinin idarə olunmasına xidmət edir. Bunu çağırmaq üçün:

`View/Project Manager`.

Bu zaman dialoq pəncərə açılır və proyekt fayllarının və kataloqlarının adları görsənir. `Project Manager`-in köməyi ilə aşağıdakı əməliyyatları yerinə yetirmək olar:

- Proyektin bir hissəsinə baxmaq;
- Proyektin ayrı-ayrı hissələrini ləğv etmək;
- Proyektə yeni hissəni əlavə etmək.

Bununla yanaşı olaraq proyektə yeni formanın əlavə edilməsini və ya proyektdən silinməsini uyğun olaraq

`Project/Add to Project`
`Project/Remove from Project` –

əmrlərindən də istifadə etməklə yerinə yetirmək olar.

İnteqrallaşmış mühitin əsas elementlərindən biri də otladka vasitələridir. Отладка vasitələri proqramda səhvlərin tapılması və aradan qaldırılmasına kömək edir. Отладка vasitələrinə müraciət üçün

`RUN` və `View/Debug Windows` –

əmrini vermək lazımdır.

Отладка vasitələrinin köməyi ilə aşağıdakı əməliyyatları yerinə yetirmək olar:

- göstərilən operatora qədər yerinə yetirmək;
- proqramı addım-addım yerinə yetirmək;
- dayanma nöqtəsinə qədər yerinə yetirmək;
- dayanma nöqtəsinin təyin edilməsi və ləğv edilməsi;
- baxış pəncərəsindən dəyişənlərin və obyektlərin qiymətlərinə baxılması;
- proqramın yerinə yetirilməsi zamanı obyektlərə qiymətlərin verilməsi.

Отладка vasitələrinin parametrlərini təyin etmək üçün

`TOOLS/Debugger Options` (Параметры отладки) – əmrini vermək lazımdır.

Açılan pəncərədə `Integrated Debugging`-də bayraqcıq (v – belə bir işarə) varsa *Отладка* qovulmuş olur.

[v] `Integrated debugging`

Proyekti şərh edən (Project Browser) və obyektlərin arxivi

Proyektin şərhçisinin (***Project Browser***) köməyi ilə ierarxik olaraq siniflərə, modullara, qlobal dəyişənlərə baxmaq olar.

Proyektin şərhçisinin köməyi ilə interface və ya *İmplementation* bölməsində istifadə olunan modulları seçmək olar, simvollara və onların elan olunmasına baxmaq olar və s.

Proyektin şərhçisi *View / Browser* əmri ilə çağırılır. Bu za-

man proyektin şərhçisinin *Exploring* (*Исследование...*) pəncərəsi açılır. Bu pəncərə iki hissədən ibarətdir. Birinci hissədə obyektlərin (sol hissədə) adları (Ağac şəklində), sağ tərəfdə isə seçilən obyektin dəqiq xarakteristikaları göstərilir.

Obyektlərin əks olunmasını idarə edən parametrlərin seçilməsi üçün 2 üsuldan istifadə etmək olar. Birinci üsul:

Proyektin şərhçisinin kontekst menyusunda *Properties* və sonra *Explorer Options* əmri;

İkinci üsul:

Environment Options pəncərəsində *Explorer* səhifəsini tapmaqla.

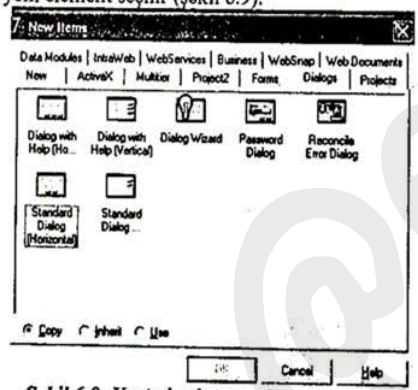
Hər iki halda *Explorer Options* pəncərəsi açılır ki, modul, sinif və qlobal dəyişənlərin idarə olunması üçün bu pəncərədə yerləşən parametrlərdən istifadə etmək olar.

Obyektlərin arxivi (Repository)

Obyektlərin arxivi və ya saxlanma yeri də İnteqrallaşmış mühitin əsas vasitələrindən biridir. Delphi eyni bir obyekti şablon olaraq proqramın hazırlanmasında çoxlu sayda istifadə edə bilər.

Belə obyektləri saxlamaq üçün xüsusi arxivdən istifadə edilir. Bu arxiv *Repository* adlanır.

Proqrama yeni obyekt əlavə etmək üçün *File / New* əmri verilir və bu zaman açılan *New Items* (*Новые Элементы*) pəncərəsindən yeni element seçilir (şəkil 6.9).



Şəkil 6.9. Yeni obyektin seçilməsi pəncərəsi

Arxivdə müxtəlif obyektlər – proqramların şablonları, formalar, hesabatlar və s. saxlanılır. Bütün obyektlər qruplarda birləşir və hər bir qrup ayrıca səhifədə yerləşir:

- New – Baza obyektlər
- Active x – Active x və OLE obyektlər
- Multi tier – çox istifadə olunan proqramların obyektləri
- Proyekt1 – tərtib olunan proyektin (proqramın) forması
- Forms – formalar
- Dialogs – dialoqlar
- Projects – proyektlər
- Business – formalar masteri

Şəkil 6.9-da Dialogs – pəncərəsi açılmışdır. Project səhifəsinin adı tərtib olunan proyektin adı ilə eyni olur və bu səhifədə Form1 – adlı tərtib olunan forma olur. Forma və proyektin adı dəyişən zaman onların adları da arxivdə dəyişir. Proyektə yeni forma əlavə edən zaman onun şablonu avtomatik olaraq proyektin səhifəsinə əlavə olunur. Proyektdən formanı silən zaman onun şablonu da arxivdən silinir. Proyektə yeni obyekti əlavə etmək üçün lazım olan səhifəni seçmək və sonra obyekti göstərmək lazımdır. Şəkil 6.9-da Dialogs səhifəsində Standard Dialog səhifəsi seçilmişdir. «OK» - düyməsini sıxsaq obyekt əlavə olunur.

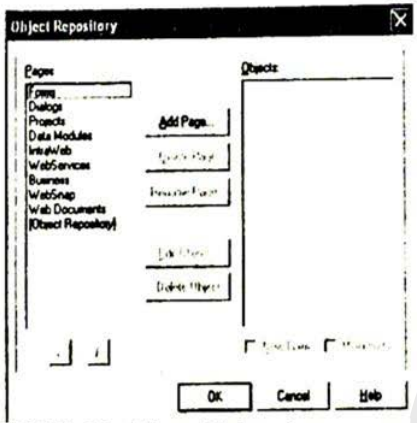
Forms və Dialogs səhifələrinin obyektlərini proyektə müxtəlif üsullarla əlavə etmək olar. Bu üsullar New Item pəncərəsinin aşağı hissəsindəki çeviricilərin vəziyyətindən asılıdır. Bu çeviricilər aşağıdakı vəziyyətlərdə ola bilər:

- Copy – proyektə arxivdən obyektin surəti (ekzemplyarı) əlavə olunur. Bu halda proyektdəki obyektdə dəyişiklik etmək olar və bu dəyişikliyin obyektin arxivdəki orijinalına təsiri yoxdur.
- Inherit – arxivdəki obyektdən yeni bir obyekt (törəmə obyekt) əmələ gəlir və proyektə bu törəmə obyekt əlavə olunur. İstifadəçi obyektə yeni komponentlər əlavə edə bilər və həmçinin onun adı ilə əlaqədar olmayan obyektdəki xassələrini dəyişə bilər. Bu obyektin modifikasiyası zamanı onun əsas komponentlərini silmək olmaz və proyektin adını (Name xassəsi) dəyişmək olmaz. Susma prinsipinə görə yaradılan obyektə bu qayda ilə Project1 səhifəsində yerləşən obyekt (adi formalar) əlavə edilir.
- USE – proyektə obyektin özü bütün faylları ilə birlikdə

əlavə olunur.

Proyektdəki obyektdə edilən dəyişiklik arxivdəki obyektə də aiddir və həmçinin digər proyektdə olan və bu obyekt tərəfindən həmin qaydada istifadə olunan obyektlərə də aiddir (USE kimi).

Obyektlərin arxivini sazlamaq üçün *Tools / Repository* – əmrini vermək lazımdır. Bu zaman *Object Repository* pəncərəsi açılır:



Şəkil 6.10. Obyektlər arxivinin sazlanması pəncərəsi

Obyektlərin arxivinin sazlanması zamanı səhifələri əlavə etmək (*Add Page...*), silmək (*Delete Page...*), yeni ad vermək (*Rename Page...*) olar, həmçinin obyekti redaktə etmək (*Edit Page...*) və obyekti silmək (*Delete Page*) olar. Pəncərənin sol hissəsində səhifələrin adları, sağ hissəsində isə qeyd olunan səhifədəki obyektlərin adları verilir.

İnteqrallaşmış mühitin əsas elementlərindən biri də ***sorğu sistemidir***. Delphi sistemində sorğu sisteminə aşağıdakılar daxildir:

1. Standart sorğu sistemi
2. İnternetin köməyi ilə sorğu
3. Kontekst – asılı sorğu

Sorğunun standart sistemi ***Delphi Help*** və ***Delphi Tools*** əmrlərinin köməyi ilə çağırılır. Bu zaman açılan dialoq pəncərədə məzmuna görə və ya sözə görə, və ya predmet göstəricisinin köməyi ilə axtarılan informasiyanı tapmaq olar.

İnternet vasitəsi ilə sorğu menyunun *Help* əmrinin köməyi ilə, *Microsoft İnternet Explorer*-ə müraciət edilir və uyğun Web-səhifə açılır.

Kontekst asılı sorğu <*F1*> düyməsinin köməyi ilə olur. Belə ki, aktiv obyektdə (dialoq pəncərə, menyu və s.) axtarılan informasiya seçilir və <*F1*> düyməsi sıxılır. Bu zaman həmin informasiya ekrana çıxır.

6.6. Object Pascal-da verilənlərin tipləri

Turbo Pascal-da olduğu kimi Object Pascal-da da verilənlərin hər bir elementi müəyyən bir tipə aid olmalıdır. Object Pascal-da verilənlərin tiplərini aşağıdakı qruplara bölmək olar:

- sadə
- strukturlaşmış
- göstəricilər
- prosedur tiplər
- variant tiplər

6.6.1. Sadə tiplər

Sadə tiplərin aşağıdakı növləri var:

- tam tiplər
- simvol və ya liter tiplər
- məntiqi tiplər
- həqiqi tiplər

Tam tiplər: Fiziki tiplər

Adı	*Diapazonu*	*Yaddaşda tutduğu yer*
Shortint	-128 – 127	1 *bayt* işarə ilə
Smallint	- 32768 – 32767	2 *bayt* işarə ilə
Longint	- 2147483648 – 2147483647	4 *bayt* işarə ilə
Int 64	$-2^{63} - 2^{63} - 1$	8 *bayt* işarə ilə
Byte	0 - 255	1 *bayt* işarəsiz
Word	0 - 65535	2 *bayt* işarəsiz
Long-Word	0 – 4294967295	4 *bayt* işarəsiz

Ümumi tiplər:

Adı	*Diapazon*	*Yaddaşın tutduğu yer*
`İnteger`	- 2147483648 – 2147483647	4 *bayt* işarə ilə
`Cardinal`	0 – 4294967295	4 *bayt* işarəsiz

Ümumi tiplər fiziki tiplərdən birinə uyğundur və onlardan istifadə edilməsi kompilyator üçün daha səmərəlidir. 16-lıq say sistemində də tam ədədləri vermək olar: diapazon - $ 00000000 - $ FFFFFFFF.

Simvol tiplər: Fiziki tiplər:

1. `Ansichar` – 1 bayt yer tutur, ANSİ (Amerikan National Standarts İnstitute) – kodundan istifadə edir.

2. `Wide Char` – 2 bayt yer tutur, Unicode – Beynəlxalq simvollar naborundan istifadə edilir. 60 min elementi var, ANSİ ilə Birinci 256 kodu üst-üstə düşür.

Ümumi tip olaraq Char – götürülür. Bu tip Ansi Char-a ekvivalentdir.

Məntiqi tiplər:

1. Boolean
2. Byte Bool
3. Word Bool
4. Long Bool

} Proqramda əsas `Boolean` tipi istifadə olunur. Yerdə qalan üç tip digər proqramlaşdırma sistemləri ilə uyğunluq yaratmaq üçündür.

Qeyd: Sadalanan və məhdud tiplər Turbo Pascal-da olduğu kimidir.

Həqiqi tiplər:

Adı	*Diapazon*	*Dəqiqlik*	*Yaddaşın tutduğu yer*
`Real 48`	$2.9.10^{-39} – 1.7.10^{38}$	11-12 rəqəm dəqiqlikdə	6 *bayt*
`Single`	$1.7.10^{-45} – 3.4.10^{38}$	7-8 rəqəm dəqiqlikdə	4 *bayt*
`Double`	$5.0.10^{-324} – 1.7.10^{308}$	15-16 rəqəm dəqiqlikdə	8 *bayt*

`Extended`	$3.6.10^{-4951} – 1.1.10^{4932}$	19-20 rəqəm dəqiqlikdə	10 *bayt*
`Comp`	$-2.10^{63} + 1 – 2.10^{63} - 1$	19-20 rəqəm dəqiqlikdə	8 *bayt*
`Currency`	-92337203685477, 5808 – 922337203685477, 5807	19-20 rəqəm dəqiqlikdə	8 *bayt*

Ümumi tip real tipidir və bu tip Double – tipinə uyğundur. Comp və Currency tipləri qeyd olunmuş nöqtəli həqiqi ədədlər üçündür və pul cəmlərini hesablamaq üçündür. Comp tipi faktiki olaraq tam ədəddir, ancaq həqiqi tipə aiddir. Bu tipli dəyişənə həqiqi ədəd mənimsənildikdə o, avtomatik olaraq yaxın tam ədədə qədər yuvarlaqlaşdırılır.

6.6.2. Verilənlərin struktur tipi

Verilənlərin struktur tipinə aşağıdakılar aiddir:
- Sətirlər
- massivlər
- çoxluqlar
- yazılar
- fayllar
- siniflər

Sətir tipi:

Verilənlərin fiziki sətir tipi:

1. `Short String` 255 maksimal uzunluq
2. `Ansi String` $\approx 2*10^{31}$ maksimal uzunluq
3. `Wide String` $\approx 2*10^{30}$ maksimal uzunluq

Verilənlərin ümumi sətir tipi `string` tipidir.

`Short String` – əvvəlki versiyalarla uyğunluq yaratmaq üçündür.

`Ansi String` – Ansi kodda, `Wide String` isə `Unicode`-də kodlaşdırmaq üçün istifadə olunur. `Ansi String` və `Wide String` dinamik massiv kimi qəbul olunur və onun maksimal uzunluğu kompüterin əsas yaddaşı qədərdir.

Bu tiplərdən əlavə `Pchar` tipi də vardır ki, bu tip sətir #0 simvolu ilə qurtardıqda tətbiq olunur. Bu sətrin də maksimal uzunluğu əsas yaddaşın tutumu ilə məhduddur.

Turbo Pascal-dan olan prosedura və funksiyalardan əlavə Object Pascal-da sətirlərlə işləmək üçün çoxlu sayda prosedura və funksiyalar vardır. Bu prosedura və funksiyalar SysUtils – modulunda yerləşirlər. Onlardan bəzilərini göstərək:

- İnttoStr(V:integer):String;

V – tam qiymətli ifadənin qiymətini sətir tipinə çevirir.

- StrtoInt(const S:String):Integer

S – sətrini tam ədədə çevirir.

- FloattoStr(V:Extended):String;

V – həqiqi ifadəsinin qiymətini sətir tipinə çevirir;

- StrtoFloat(const S:String):Extended;

S – sətrini həqiqi ədədə çevirir.

- DatetoStr(Date:TDateTime):String; – tarixin qiymətini sətir tipinə çevirir.

- TimetoStr(Tame:TDateTime):String; – vaxtın qiymətini sətir tipinə çevirir.

- Trim(const S:String):String; S – sətrinin sağındakı və solundakı probelləri və idarəedici simvolları silir.

- Trimleft(const S:String):String;

Trim Right(const S:String):String – sol və sağdan probelləri və idarəedici simvolları silmək üçün istifadə olunur.

Massivlər

Massivlərin statik və dinamik növləri vardır. Statik massivlərin təsviri və onun elementləri ilə işləmək qaydaları Turbo Pascal-da olduğu kimidir. Dinamik massivlərin təsvirində ancaq onun elementlərinin tipi göstərilir, indeksin dəyişmə diapazonu (massivin ölçüsü) göstərilmir. Massivin ölçüsü yerinə yetirilmə zamanı müəyyən edilir.

Dinamik massivin təsvirinin formatı:

Var <massivin adı>:array of <elementin tipi>;

Dinamik massivin ölçülərinin verilməsi yerinə yetirilmə zamanı

```
Set Lenght(Var S;N:integer);
```

- prosedurasının köməyi ilə olur.

Burada S – dinamik massiv, N – isə onun ölçüsüdür. Ölçü

verildikdən sonra onun elementləri üzərində əməliyyatlar aparmaq olar. Dinamik massivin uzunluğunu, indeksin aşağı və yuxarı sərhəddini təyin etmək üçün uyğun olaraq Length( ), Low( ), Hight( ) – funksiyasından istifadə etmək olar.

Elementlərin nömrələnməsi 0-dan başlandığı üçün Low( )=0 olar.

Dinamik massivdən istifadə etməyə aid misal:

```
var n:integer;
    m:array of real;
.....
Set length(m,100);
for n:=0 to 99 do
m[ n] :=n;
Set length(m,200);
.....
```

İki ölçülü dinamik massivlər aşağıdakı kimi təsvir olunurlar:

```
var <massivin adı>: array of array of
        <elementin tipi>;
```

Bu halda massivin ölçüsünü təyin etmək üçün

```
Setlength(var S; N,M:integer);
```

- prosedurasından istifadə edilir. N, M – birinci və 2-ci indeksin dəyişmə diapazonunu müəyyən edir.

Əgər S:=nil; yazılarsa massiv üçün ayrılmış yaddaş sahəsi azad olunur.

Qeyd: Çoxluq və yazı tiplərin verilməsi Turbo Pascal-da olduğu kimidir.

Fayl tiplər də Turbo Pascal-da olduğu kimidir. Elan olunmasına görə üç növə malikdir:

- tipi məlum olan fayllar;
- mətn faylları;
- tipi məlum olmayan fayllar.

Yalnız fərq mətn fayllarının elan olunmasındadır:

```
Var<f.t.d.>:Text File;
```

Məsələ: Var f1:Text File;

və ya Var <f.t.d.>:System.text (System modulunda Text faylı)

Göstərici tiplər də Turbo Pascal-da olduğu kimidir:

```
type <göstərici tip>=^<ünvanlaşan
        verilənlərin tipi>;
```

və ya

```
var <göstərici tip>:pointer;
```

Sysutils və System modullarında çoxlu sayda göstərici tiplər vardır. Məsələ, PString, PVariant və s.

Misal: `Var p:^integer; n,k:integer; ... p:=@n; n=100; k=p^+10;` (nəticədə 110 alınır).

Prosedur tiplər də Turbo Pascal-da olduğu kimidir. Burada prosedur tiplər hadisələrin emalçısını təyin edən zaman daha çox istifadə olunur.

Məsələ, bir çox hadisələrin emalçısı üçün (məsələ OnClick, OnChange) aşağıdakı prosedur tipi təyin etmək olar;

```
type TNotifyEvent=procedure (Sender:Tobject)
of object;
```

Button1 düyməsinin OnClick hadisəsinə Button1Click prosedurasını (hadisələrin emalçısını) təyin etmək olar:

```
Button1.OnClick:=Button1Click;
```

Bu proseduranı əvvəlcədən tərtib etmək lazımdır və o, OnClick hadisəsinin TNotifyEvent tipinə malik olmalıdır (bəzi hallarda prosedur tipin təyinində of object yazmaq olar).

6.6.3. Variant tipi

Variant tipi müxtəlif üsullarla interpretasiya olunan qiymətlərin təsviri üçün tətbiq edilir. Variant tipli dəyişənə müxtəlif tipə malik qiymətlər mənimsətmək olar. Bu tiplər o vaxt tətbiq olunur ki, tipin qiyməti kompilyasiya zamanı məlum olmasın və ya proqramın yerinə yetirilmə vaxtı dəyişsin. Bu tipli dəyişənə tam qiymətli (int 64-dən başqa), həqiqi, simvol, sətir və məntiqi tipli qiymətlər mənimsətmək olar. Bu tiplər Variant tipi ilə uyğundurlar və tiplərin çevrilməsi avtomatik olaraq yerinə yetirilir.

Məsələn,

```
Var v1,v2:Variant;
k:integer;
x:real;
...
k:=10;
v1:=k;
x:=23.7;
```



```
v2:=x;
v1:=x+0.5;
```

Bu tipli dəyişənlərdən əsasən verilənlər bazasından yazıların axtarılmasında istifadə edilir.

6.6.4. Siniflər

Object Pascal dilində siniflər obyektləri təsvir etmək üçün istifadə edilir. Müəyyən bir sinfin tipinə malik olan obyekt bu sinfin ekzemplyarı və ya bu tipdən olan dəyişən adlanır.

Sinif elə tip yazıdır ki, bu yazının tərkibində sahə, xassə və metodlar vardır. Sahə yazılarda olduğu kimidir və obyekt haqqında informasiya saxlamaq üçündür. Metodlar prosedura və funksiyalardan ibarətdir. Xassəni həm sahə kimi istifadə etmək olar, ona qiymətlər mənimsətmək olar, həm də sinfin daxilində xassənin qiymətinə sinfin metodları vasitəsi ilə müraciət etmək olar. Siniflərin təsviri aşağıdakı struktura malikdir:

```
Type <sinfin tipi>=class(<əsas sinfin adı>)
private
<xüsusi təsvirlər>;
protected
<qorunan təsvirlər>;
public
<ümumi müraciət mümkün olan təsvirlər>;
published
<publikasiya olunan təsvirlər>
end;
```

Private və protected bölmələrindəki təsvirə modulun daxilindən və ya varis olan siniflərdən müraciət etmək olar. Public bölməsindəki təsvirlərə proqramın istənilən yerindən müraciət etmək mümkündür. Published bölməsində proqramı layihələşdirən zaman obyektlərin xassələrinə müraciəti təmin edilir. Yerinə yetirilmə vaxtı obyektlərin tipi haqqında məlumata malik olur. Obyektlər inspektorunda publikasiya olunan xassələr görsənir. Published bölməsi göstərilmədikdə susma prinsipinə görə başa düşülür. Sahənin təsvirinə aid misal:

```
type TNewclass=class(Tobject)
private
FCode:integer;
FSign:char;
```

```
FNote:string;
end;
```

Xassə – sahələrə müraciət mexanizmini icra edir. Hər bir xassəyə bir sahə uyğundur. Bu sahədə xassənin qiyməti və 2 metod yerləşir. Bu metodlar sinfin sahələrinə müraciəti təmin edir. Xassənin təsviri property sözü ilə başlayır. Xassənin təsvirinə aid misal;

```
type TNewclass=Class(TObject)
private
FCode:integer;
FSign:Char;
FNote:String;
published
property code:integer read FCode Write FCode;
property Sign:Char read FSign Write FSign;
property Note:String read FNote Write FNote;
end;
```

Qorunan bölmədə təsvir olunan FCode, FSign, FNote sahələrinə müraciət üçün Code, Sign, Note xassələrindən istifadə edilir.

Metodun təsvirinə aid Button1Click – metodunun təsvirini misal kimi göstərək:

```
interface
type TForm1=class(TForm)
Button1:TButton;
procedure Button1Click(Sender:TObject);
end;
.....
implementation
.....
procedure TForm1.Button1Click(Sender:
                                    TObject);
begin
close;
end;
```

Siniflərdə elan olunan metodlar müxtəlif üsullarla çağırıla bilər. Bu metodun növündən asılıdır. Metodların aşağıdakı növləri vardır:

- virtual – virtual metod;
- dinamic – dinamik metod;

- override – overley metod;
- message – məlumatları işləyən metod;
- abstract – absrakt metod.

Bu metodların tərtib olunmasında xüsusi növ metodlardan – construktor və destriktordan istifadə edilir.

6.6.5. Yerinə yetirilmə vaxtı tiplər haqqında informasiya

Obyektdə tiplər haqqında informasiya – RTTİ – (Run Time Type İnformation) saxlanılır. Bu informasiyadan obyektin bu və ya digər tipə aid olması yoxlanılır.

Metodların çoxuna müraciət zamanı TObject tipinə malik olan Sender parametri ötürülür. Aparılan əməliyyatlara uyğun olaraq onun tipi bu əməliyyatlar aparılan obyektin tipinə çevrilir. Tiplərin çevrilməsinin ***aşikar*** və ***qeyri/aşikar*** növləri vardır. Qeyri-aşikar tiplərin çevrilməsi üçün is və as – operatorlarından istifadə edilir:

```
<obyekt> is <sinif>
```

Əgər obyekt göstərilən sinfə daxildirsə True, əks halda False qiymətini alır.

```
<obyekt> as <sinif>
```

Bu halda obyektin tipi sinfin tipinə çevrilir. Qeyri-aşikar çevrilməyə misal:

```
Procedure TForm1.Button1Click(Sender:
                                    TObject);
begin
if (Sender is TButton) then (Sender as
TButton).Caption:=TimeToStr (Now);
end;
```

Button1 – düyməsini sıxan zaman onun adında cari vaxt əks olunacaq

Əgər emalçı ancaq Button1 düyməsinə təyin olunarsa, komponentin adının dəyişməsi aşağıdakı operator vasitəsi ilə daha asan olar:

```
Button1. Caption: = TimeToStr (Now);
```

Tiplərin qeyri-aşikar çevrilməsi o zaman əhəmiyyətli olur ki, emal edən prosedura bir neçə müxtəlif tipli komponentlər üçün ümumi olsun.

Aşikar çevrilmə aşağıdakı kimi olur:

```
<tip>(<obyekt>)
<obyekt>-in tipi göstərilən <tip>-ə çevrilir.
Misal:
Procedure TForm1.Button1Click(Sender:
                                    TObject);
begin
TButton(Sender).Caption:='OK';
end;
```

Bu halda komponentin tipi `TButton` tipinə çevrilir və bu komponentin üzərində siçanın düyməsini sıxdıqda öz adını dəyişərək «OK» adına malik olur.

6.7. Vizual komponentlər kitabxanası

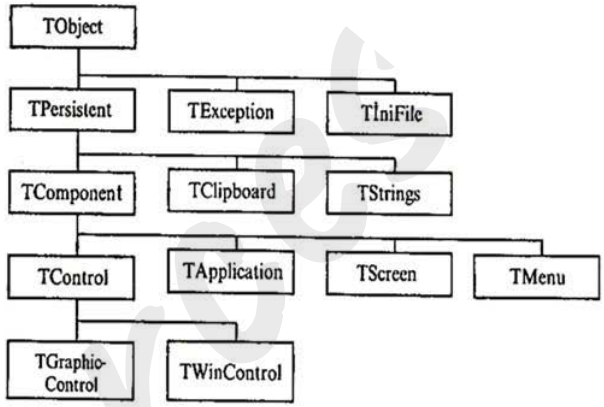
Vizual komponentlər kitabxanasında (VCL – Visual Component Library) çoxlu sayda siniflər yerləşir ki, bu siniflərdən proqramı sürətlə işləmək üçün istifadə olunur. VCL-də həm vizual, həm də vizual olmayan komponentlər yerləşir və onlar Delphi-nin inteqrallaşmış mühiti ilə əlaqədardırlar. Bütün komponentlər sinifdirlər, lakin bütün siniflərin heç də hamısı komponent deyildir.

VCL-nin bütün sinifləri ierarxiyanın müəyyən bir səviyyəsində yerləşirlər və siniflərin ağacını əmələ gətirirlər. Belə struktura görə əsas sinfə aid olan xassələr onun bütün varis siniflərinə də aiddir.

`TObject` – sinfi bütün siniflər üçün ümumidir və ağacın əsasında yerləşmişdir. Bu sinif abstraktdır və bütün varis siniflər üçün ümumi olan metodları icra edir. Bu metodlara aşağıdakıları misal göstərmək olar:

- `Create` – obyekti yaradan;
- `Destroy` – obyektin ləğv edilməsi;
- `Free` – `Create` – ilə yaradılan obyektin ləğv edilməsi, bu zaman `Destroy` – metodu çağrılır.

Siniflərin ierarxiyasından bir fraqment şəkil 6.12-də göstərilmişdir.



Şəkil 6.12. Siniflərin ierarxiyasından bir fraqment

Məsələ, `TPersistent` sinfi də bir çox obyeklər üçün abstraktdır və disk və operativ yaddaşla işləmək üçün istifadə edilir.

`TComponent` sinfi – bütün komponentlər üçün baza sinfidir. Komponentlərin yaradılmasında, ləğv edilməsində əsasən istifadə olunur.

`TControl` sinfi – virtual komponentlər üçün baza sinfidir (idarəedici elementlər). Bu sinif virtual komponentlərin fəaliyyət göstərməsi üçün vasitələrə malikdir. Bütün vizual komponentlər pəncərəli və qrafik komponentlərə bölünürlər ki, bu komponentlər də uyğun olaraq `TWin Control` və `TGraphic – Control` siniflərinə aiddir. Ən çox istifadə olunan siniflər `TPersistent`, `TComponent` və `TControl` sinifləridir.

6.8. Vizual komponentlərin ümumi xarakteristikaları

Proqram interfeysini yaratmaq üçün Delphi-də vizual komponentlər çoxluğundan istifadə edilir ki, bunlar əsasən Standard, Additional və Win 32 – səhifələrində yerləşirlər. Komponentlərin bu bölgüsü şərtidir. Məsələ, Additional-da yerləşən BitBtn – komponenti ilə Standard səhifəsində yerləşən Button komponenti praktik olaraq funksiyalarına görə bir-birindən fərqlənmirlər.

Standard səhifəsindəki komponentlər	*Additional səhifəsindəki komponentlər*	*Win 32 səhifəsindəki komponentlər*
Frames – freymlər	Bit Bitn – şəkillərlə olan düymə	Tab Control – idarəedici
Main Menu – baş menyu	Spead Button – sürətli müraciət	Page Control – bloknot
Pupup Menu – dəyişən menyu	Mask Edit – şablonla olan birsətirli redaktor	İmage List – qrafik obrazların siyahısı
Labell – yazı	String Grid – sətirlər cədvəli	Rich Edit – tam funksional mətn redaktoru
Edit 1 – birsətirli redaktor	Draw Grid – cədvəl	Trac Bar – hərəkət edən
Memo – çoxsətirli redaktor	İmage – qrafik obraz	Proqressor Bar – işin gedişini göstərən indikator
Button – Standard düymə	Sape – həndəsi fiqur	Up Down – sayqac
Check Box – bayraq	Bevel – faska	Hotkey – düymələrin kombinasiyasının redaktoru
Radio Button – çevirici	Scroll Box – fırlatma oblastı	Animate – video kliplərə baxmaq
List Box – siyahı	Check List Box – çeviricilər siyahısı	Date time Picker – tarixi daxil edən sətir
Combo Box – siyahı ilə olan sahə	Splitter – ayırıcı	Month Calendar – təqvim
Scroll Bar – fırlatma zolağı	Static Text – Statistik mətn	True View – obyektlər ağacı
Group Box – qrup	Control Bar – alətlər paneli üçün konteyner	List View – siyahı
Radio Group – asılı çevricilər qrupu	Application Evens – hadisələr	Header Control – ayırıcı

Panel – panel	Chart – diaqramlar	Status Bar – vəziyyətlər sətri
Action List – əməliyyatların siyahısı		Tool Bar – alətlər paneli
		Cool Bar – «əyri» alətlər paneli
		Page Scroller – xəyalların fırladılması

Bu səhifələrdən əlavə System, Data Access, Data Controls, İnter Base, Midas, İnternet Explorer kimi səhifələr də vardır ki, bu səhifələrdə yerləşən komponentlərin köməyi ilə daha mürəkkəb proqramlar yazmaq olar.

VCL – vizual komponentlər kitabxanasında bütün vizual komponentlər üçün baza sinfi TControl sinfidir. Bu sinif elementin yerini, ölçüsünü, onun başlığını, rəngini və digər parametrlərin müəyyən olunmasında əsas rol oynayır. TControl sinfi vizual komponentlərin ümumi xassələrini, hadisələrini və metodlarını əhatə edir.

Vizual komponentlər 2 qrupa bölünür:

1) Pəncərəli
2) Pəncərəsiz

Pəncərəli komponentlər konkret məqsəd üçün istifadə olunan xüsusi pəncərələrdir. Pəncərəli idarəetməyə malik olan elementlərə redaktə etmə komponentlərini misal göstərmək olar. Pəncərəli idarəetmə elementləri üçün TWinControl sinfi baza sinfidir.

İdarəetmənin pəncərəli elementləri giriş üçün fokusa malikdir. Giriş üçün fokus aşağıdakı üsullardan biri ilə verilə bilər:

- redaktə etmə kursoru vasitəsi ilə (məsələ, Edit, Memo)
- düzbucaqlının köməyi ilə (məsələ, Button1).

İdarəetmənin pəncərəli elementləri pəncərənin deskriptoruna malikdir ki, bu deskriptor Windows tərəfindən istifadə olunur. Belə ki, deskriptora görə Windows pəncərəni tanıyır və bu pəncərəyə müraciəti təmin edir.

Pəncərəsiz idarəetmə elementlərinin baza sinfi TGraphic Control-dır. Bu TControl sinfinin alt sinfidir. Bu elementlər giriş fokusuna malik deyillər və digər interfeys elementləri üçün əsas (ana) sinif ola bilərlər. Pəncərəsiz elementlərin üstünlüyü ondan ibarətdir ki, o resurslardan az istifadə edir və deskriptora malik olmur. Məsələ, alətlər paneli yaratmaq lazım olarsa Button standart komponentinə nisbətən daha cəld işləyən Speed Button komponentindən istifadə etmək olar. Vizual komponentlər daha çox ümumi xassələrə, hadisələrə və metodlara malikdirlər.

6.9. Vizual komponentlərin xassələri

Xassə – proqramın tərtibi və yerinə yetirilməsi zamanı komponentlərin xarici görünüşünü və özünü aparmasını idarə etməyə imkan verir. Komponentlərin xassələrinin qiymətləri əsasən proqram tərtib edilən zaman Inspector object vasitəsi ilə verilir. Xassələrə aid aşağıdakıları misal göstərmək olar:

1) Caption xassəsi TCaption tipinə malikdir və komponentin başlığını yazmaq üçün istifadə olunur. TCaption tipi String tipinə oxşardır. Başlıqlarda bəzi simvolların altından xətt çəkilmiş olur ki, bu klavişlərin kombinasiyasından istifadə etməklə, cəld müraciəti təmin edir <Alt> + <xətt çəkilmiş simvol>.

Klavişlərin kombinasiyasını görmək üçün '&' - simvoldan istifadə edilir. Məsələ:

```
Button1.Caption:='& Close';<Alt>+<C>;
Label1.Caption:='Y& OL';<Alt>+<O>.
```

2) Align xassəsi TAlign tipinə malikdir və konteynerdə komponentin düzləndirilməsi variantını təyin edir. Konteyner rolunu Form forması və ya Panel paneli oynayır.

Align – xassəsi çoxlu qiymətlər ala bilər:

- alNone – düzləndirməkdən istifadə olunmur, komponent olduğu yerdə də yerləşir.
- alTop – komponent konteynerin yuxarı hissəsində yerləşir. Hündürlüyü dəyişmir, eni konteynerin eninə bərabər olur.
- alBottom – alTopa oxşardır. Konteynerin aşağı hissəsində yerləşir.

- alLeft – konteynerin sol hissəsində yerləşir. Eni dəyişmir, hündürlüyü konteynerin hündürlüyünə bərabər olur.
- alRight – alLeft-ə oxşardır. Sağ tərəfdə yerləşir.
- alClient – komponent bütün konteyneri tutur.

Misal: Panelin formaya nisbətən düzləndirilməsi. Adətən komponentlər paneli baş menyunun aşağı hissəsində yerləşir. Bu paneli formanın yuxarı küncünə düzləndirmək üçün:

```
Panel1.Align=alTop;
```

yazmaq lazımdır.

3) TColor tipinə malik olan Color xassəsi komponentin fonunun rəngini müəyyən edir. Rəngi müəyyən edən sabitlərə misallar:

clBlack – qara, clBlue – mavi,
clGreen – yaşıl, clRed – qırmızı və s.

4) Ctl3D komponenti Boolean tipinə malikdir və vizual komponentlərin formasını (görünüşünü) müəyyən edir. Komponentin qiyməti False olarsa 2 ölçülü, True olarsa 3 ölçülü (susmaya görə) görünüş olar.

5) Cursor xassəsi TCursor tipinə malikdir və siçanın göstəricisinin formasını müəyyən edir. Kursorun göstəricisinin 20-yə qədər forması ola bilər və əsas istifadə olunanlar aşağıdakılardır:

- CrDefault – susma prinsipinə görə (adətən ox şəklində);
- CrNone – göstərici görünmür;
- CrArrow – ox şəklində görünür;
- CrCross – xaç şəklində görünür;
- CrHovrClass – qum saatı şəklində görünür.

6) Drag Cursor xassəsi TCursor tipinə aiddir və komponentin yerdəyişməsi zamanı kursorun forması müəyyən olunur. Bu xassənin qiymətləri Cursor xassəsinin qiymətlərindən fərqlənmir.

7) Enabled xassəsi Boolean tipinə malikdir və komponentin aktivliyini müəyyən edir. Belə ki, siçan və klaviatura vasitəsi ilə daxil olan məlumatlara reaksiya vermək qabiliyyəti müəyyən edilir.

8) Font xassəsi TFont tipinə malikdir və vizual komponentlərdə əks olunan mətnin şriftlərini müəyyən edir. Bundan əlavə TFont sinfində şriftlərin parametrlərini idarə edən xassələr də vardır. Font xassəsinin qiymətləri aşağıdakılardır:

- Name:TFont Name tipinə malikdir və şriftin adını müəyyən edir;
- Size:integer; – şriftin ölçüsün bildirir;
- Style:TFont Style; - şriftin stilini bildirir;
- Color:TColor; mətnin rəngini bildirir.

Məsələ:

```
Label1.Font.Color:=ClGreen;
```

(mətnin rəngi yaşıl olur)

```
Label1.Color:=ClBlue;
```

(fonun rəngi mavi olur)

9) Hlight və Width, Left və Top xassələri integer tipinə aiddirlər və uyğun olaraq komponentin vertikal, horizontal ölçülərini müəyyən edirlər və komponentin sol yuxarı küncünün onun yerləşdiyi konteynerə görə koordinatlarını təyin edirlər.

10) HelpContext xassəsi THelp - Context tipinə malikdir və sorğu sisteminin kontekst nömrəsini göstərir.

11) Hint xassəsi String tipinə aiddir və kursor komponentin üzərində yerləşdikdə ona aid olan mətni əks etdirir. Mətni (подсказка) görmək üçün ShowHint xassəsinə True mənimsətmək lazımdır. Əks halda False – mənimsətmək lazımdır.

12) PopupMenu xassəsi TPopupMenu tipinə aiddir və siçanın sağ düyməsini sıxdıqda lokal kontekst menyunu almaq üçündür. Bu menyunun görsənməsi üçün AutoPop xassəsinə True mənimsətmək lazımdır. Susma prinsipinə görə False başa düşülür.

13) Text xassəsi TCaption tipinə malikdir və Caption xassəsinə uyğundur. Ondan fərqi odur ki, burada komponentin başlığı deyil, onun məzmunu görsənir. Məsələ, Edit və Memo-da onların daxilində yazılan mətn və xassənin qiyməti olur.

14) Read Only xassəsi Boolean tipinə aiddir və daxil edilən və redaktə edilən informasiyadakı mətnlərdə dəyişiklik edilib-edilməməsinə icazə verilir. Əgər Read Only=True olarsa, onda mətn ancaq oxumaq üçündür. Əgər Read Only=False olarsa, mətni redaktə etmək olar. Ancaq istifadəsi mətni dəyişə bilməz. Proqramçı proqram yolu ilə mətni dəyişə bilər. Məsələ:

```
Edit1.Read Only:=True;
Edit1.Text:='yeni mətn';
```

Bunlardan əlavə TabOrder, TTabOrder komponentlərə baxış ardıcıllığını müəyyən edir. TabOrder 0, – 1, – 2 – qiymətini

alır. Parent TWinControl – dinamik olaraq komponentlərin yaradılmasında istifadə olunur. DragMode, Constraints xassələri də vardır ki, onlardan da yeni komponent və onun ölçülərini təyin etdikdə və siçanın köməyi ilə yerini dəyişdikdə istifadə olunur.

6.10. Vizual komponentlərin hadisələri

Vizual komponentlər çoxlu sayda (onlarla) müxtəlif növlü hadisələri emal etmək imkanına malikdirlər. Hadisələri qruplara bölmək olar. Bu qruplar əsasən aşağıdakılardır:

- idarəedici elementin seçilməsi;
- siçanın göstəricisinin yerinin dəyişməsi;
- klaviaturanın klavişinin sıxılması;
- idarəedici elementin giriş fokusunun alınması və itirilməsi;
- drag-and-drop metodu ilə obyektlərin yerinin dəyişməsi.

Hadisələr də xassələr kimi uyğun bir tipə aid olmalıdır. Hadisələrin böyük bir qismi TNotifyEvent tipinə daxildir. Bu tip aşağıdakı kimi təyin olunur:

```
type TNotifyEvent=procedure(Sender:Tobject)
of object;
```

Bəzi hadisələrdə Sender parametrindən əlavə, digər parametrlər də prosedura daxilinə ötürülür, məsələ, siçanın göstəricisinin dəyişməsi ilə əlaqədar olan hadisədə siçanın göstəricisinin koordinatları da prosedura daxilinə ötülür.

İdarəedici elementin seçilməsində TNotifyEvent tipinə malik olan OnClick hadisəsi baş verir. OnClick hadisəsi proqram tərtibində ən çox işlənən hadisədir. Bu hadisə siçanın düyməsini komponentin üzərində sıxdıqda baş verir. Məsələ, LABEL1 – komponenti üçün bu hadisəni emal edən proseduranı yazaq:

```
procedure TForm1.Label1Click(Sender:Tobject);
begin
Label1.Caption:=Time to Str(Time);
end;
```

(Time – funksiyasının qiyməti cari vaxtdır)

Siçanın ixtiyari düyməsini sıxdıqda aşağıdakı iki hadisə də baş verir:

- OnMouseDown:TMouseEvent tipinə malikdir, siçanın

düyməsini sıxan zaman baş verir;

- `OnMouseUp:TMouseEvent;` - siçanın düyməsini buraxan zaman baş verir.

Bunlardan əlavə siçanın sol düyməsini komponentin üzərində 2 dəfə sıxdıqda `OnDblClick:TNotifyEvent` tipli hadisə baş verir.

Delphi bu hadisələri `Click` metodu vasitəsi ilə imitasiya etməyə imkan verir. Məsələ

`Button 2 Click`

- operatoru `Buton 2` – düyməsinin sıxılmasını imitasiya edir.

Siçanın göstəricisinin vizual komponentlər üzrə yerini dəyişməsi zamanı `OnMouseMove:TMouseMoveEvent` hadisəsi kəsilməz olaraq baş verir.

Burada:

```
TMouseMoveEvent=procedure(Sender:Tobject;
Shift:TShift State; x,y:integer) of object;
```

Burada `Sender` parametri siçanın göstəricisinin hansı obyekt üzərində olduğunu göstərir. `x,y` isə göstəricisinin `Sender` obyektinin koordinat sistemində mövqeyini göstərir. `Shift` parametri isə <*Alt*>, <*Ctrl*>, <*Shift*> və siçanın düyməsinin sıxılıb-sıxılmamasının vəziyyətini göstərir. Bu parametr aşağıdakı qiymətləri ala bilər:

- `ss Shift` – <*Shift*> - düyməsi sıxılmışdır;
- `ss Alt` – <*Alt*> - düyməsi sıxılmışdır;
- `ss Ctrl` – <*Ctrl*> - düyməsi sıxılmışdır;
- `ss Left` – siçanın sol düyməsi sıxılmışdır;
- `ss Midle` – siçanın orta düyməsi sıxılmışdır;
- `ss Double` – siçanın düyməsi 2 dəfə sıxılmışdır;

Məsələ, <*Shift*> və <*Alt*> düyməsi birlikdə sıxıldıqda Shift parametri [ss Shift, ss alt] qiymətlərini alır. Heç biri sıxılmadıqda [ ] – qiymətini alır.

Aşağıdakı misalda siçanın koordinatlarını əks etdirən prosedura yazılmışdır:

```
Procedure TForm1.Form Mouse Move(Sender:
Tobject; Shift:TShift State; x,y:integer);
begin
Form1.Caption:='siçanın göstəricisinin koor-
dinatı:'+İnttoStr(x)+'və'+İnttoStr(Y);
end;
```


Bu halda siçanı formanın üzərində gəzdirdikdə onun koordinatları formanın başlığında əks olunur. Kursor formada boş sahələrdə gəzməlidir. İdarəedici elementlərin üzərində olduqda onun koordinatları əks olunmur. `x,y` koordinatları formanın sol yuxarı küncünə nəzərən piksellərlə hesablanır.

Klaviatura ilə işləyən zaman `OnKeyPress` və `OnKeyDown` hadisələri klavişləri sıxan zaman, `OnKeyUp` – isə klavişi buraxan zaman yaranır. `OnKeyDown` – hadisəsi klavişi kəsilməz olaraq sıxdıqda, `OnKeyUp` isə buraxdıqda yaranır.

`OnKeyPress` – hadisəsi isə `TKeyPressEvent` tipinə malikdir və hərf rəqəm klavişlərinin hər dəfə sıxılması zamanı generasiya olunur. Adətən o, bir klavişin sıxılmasına reaksiya tələb olunduqda işlənilir. `TKeyPressEvent` tipi aşağıdakı kimi təyin olunur:

```
type TKeyPressEvent=procedure(Sender:
Tobject; Var Key: Char) of object;
```

`Key` parametrində sıxılan simvolun ASCİİ kodu olur. Zəruri olan halda bu kodu analiz etmək və dəyişmək olar. Əgər `Key` parametrinə #0 – simvolu mənimsədilərsə, onda klavişin sıxılması qadağan olunur.

`OnKeyPress` hadisəsinin emalçısına aid misal:

```
procedure TForm1.Edit1KeyPress(Sender:
Tobject; Var key:Char);
begin
if key='t' then key:#0;
end;
```

Burada `Edit1` redaktorunun məzmununu redaktə edən zaman 't' simvolunun daxil edilməsi qadağan edilir.

ASCİİ kodu olmayan idarəedici simvollarla işləyən zaman `TKeyEvent` tipinə malik olan `OnKeyDown` və `OnKeyUp` hadisələrindən istifadə etmək olar. `TKeyEvent` tipi aşağıdakı kimi təyin olunur:

```
type TKeyEvent=procedure(Sender:Tobject;
Var key:Word; Shift:TShiftState);
```

Bu hadisədən <*Shift*>, <*Alt*>, <*Ctrl*> və s. idarəedici klavişlərin vəziyyətini analiz etmək üçün istifadə olunur. Burada göründüyü kimi `key:Word` tipinə malikdir və klavişin kodunu almaq üçün `Chr(key)` – funksiyasından istifadə etmək lazımdır.

Hərf – rəqəm və idarəedici simvolların birgə işlənməsinə aid misal:

```
Procedure TForm1.Edit2KeyDown(Sender:
Tobject; Var key:Word; Shift:TShift State);
begin
if (Shift=[ ss Ctrl] ) and (Chr(key)='1') then
MessageDlg('"Ctrl-1"düymələri sıxılmışdır',
mtconfirmation,[ mbok] ,0);
end;
```

Əgər EDİT2 – komponenti giriş foksunda yerləşərsə, onda *'Ctrl+1'* - düymələri sıxıldıqda Confirm dialoq pəncərəsi çağırılır və bu pəncərədə *"'Ctrl+1'* düymələri sıxılmışdır" məlumatı görsənir.

Qeyd edək ki, bəzi klavişləri (məsələ Tab) sıxdıqda OnkeyPress və OnkeyUp hadisələri yaranmır.

Pəncərəli elementlərin foksunu alan zaman TNotifyEvent tipinə malik OnEnter hadisəsi yaranır. Bu hadisə idarəedici elementlərin hər hansı bir üsulla aktivləşməsi zamanı (məsələ, siçanın düyməsini sıxmaqla və ya *Tab* düyməsini sıxmaqla və s.) yaranır. Fokusun itirilməsi zamanı OnExit:TNotifyEvent hadisəsi əmələ gəlir. İdarəedici elementlərin giriş foksunun alınması və itirilməsi (ləğvi)-nə aid aşağıdakı misala baxaq:

```
ProcedureTForm1.Edit1 Enter(Sender:TObject);
begin
Label1.Caption:=(Sender as TControl).Name +
'aktivdir';
end;
procedure TForm1.Edit1 Exit(Sender:Tobject);
begin
Label1.Caption:=TEdit1(Sender).Name+'aktiv
deyil';
end;
```

Label1 komponentinə Edit1-in giriş foksunun aktivliyi və ya aktiv olmaması haqqında məlumat çıxır.

Drag-and-drop (yerini dəyişmək və saxlamaq) texnologiyası müxtəlif obyektləri, məsələ bir siyahıda olan elementləri digər siyahıya qoymağı təmin edir. Bu halda idarəetmədə 2 element istifadə olunur: mənbə, qəbuledən. Mənbədə yerini dəyişən obyekt

yerləşir, qəbuledəndə isə mənbədəki obyektin idarəedici elementləri yerləşir.

Bu metodlarla əlaqəli olan aşağıdakı iki hadisə daha çox istifadə olunur:

1) OnDragOver:TDragOverEvent – tipinə malikdir. Hadisələrin emalçısına aşağıdakı parametrlər ötürülür: Tobject tipinə malik olan Source – mənbə obyekti; Tobject tipinə malik Sender qəbuledici obyekti; Siçanın göstəricisinin cari x,y: integer koordinatı, yerdəyişmənin vəziyyəti State:TDragState və yerdəyişmə əlamətini təsdiq edən Accept:Boolean parametri. Əgər yerdəyişmə əməliyyatı qəbul olunarsa Accept =true – əks halda Accept=false qiymətini alır. OnDragOver – hadisəsində əməliyyatın mümkünlüyü analiz olunur.

2) OnDragDrop: TDragDropEvent – obyekt qəbulediciyə doğru yerini dəyişdikdə bu hadisə qəbuledici tərəfindən çağırılır. Yerdəyişmə əməliyyatının işlənməsi üçün hadisələrin emalçısına aşağıdakı parametrlər ötürülməlidir: ilkin Source: Tobject – obyekti; Sender qəbuledici obyekti və siçanın göstəricisinin cari (x,y) koordinatı.

OnDragDrop – hadisəsində yerdəyişən obyektin qəbulu və işlənməsi məsələsi həll olunur.

Misal 1. Tutaq ki, Label1-də yazılan mətnin Form1 – forması daxilində yerini dəyişmək tələb olunur. Adətən yerdəyişmə əməliyyatı başlamazdan əvvəl DragMode xassəsinə dmAutomatic qiyməti vermək lazımdır ki, yerdəyişmə əməliyyatının başlanması avtomatik olsun. Əks halda proqram yolu ilə BeginDrag metodunu çağırmaq lazımdır.

```
// Label1 yazısı üçün
// Drag Mode xassəsinə dm Automatic
// qiymətini verməli
procedure Form1.FormDragOver(Sender,Source:
Tobject; x,y:integer; State:TDrag State;
var Accept:Boolean);
begin
if Source=Label1 then Accept:=true else
Accept:=false;
end;
procedure Form1.FormDragDrop(Sender,Source:
```

```
Tobject; x,y:integer);
begin
Label1.Left:=x; Label1.Top:=y;
end;
```

Misal 2. Bir siyahının elementlərinin digər siyahıya köçürülməsinə aid misal.

```
// List Box 1 siyahısı üçün DragMode xassəsinə
// dm Automatic qiymətini mənimsətməli.
procedure TForm1.ListBox2DragOver(Sender,
Source:Tobject;x,y:integer;State:TDragState;
var Accept:boolean);
begin
if Source=ListBox1 then Accept:=true else
Accept:=false;
end;
procedure TForm1.ListBox2DragDrop(Sender,
Source:Tobject; x,y:integer);
begin
With Source as TListBox do
ListBox2.İtems.Add(İtems[ İtemİndex] );
İtems.Delete(İtemİndex);
end;
```

Əgər `With` operatoru işlənməsə, onda;

```
ListBox2.İtems.Add(Source as TListBox).
İtems((Source as TListBox).İtemİndex);
(Source as TListBox).İtems.Delete((Source
as TListBox).İtemİndex);
```

Əgər kursor komponentin üzərində müəyyən müddət hərəkətsiz qalarsa, onda `OnHint:TNotifyEvent` hadisəsi yaranar. Bu hadisənin emalçısında komponentə aid olan köməkçi sözlərin yazılmasını və bu sözlərin komponentin yanında görsənməsini təmin etmək olar.

6.11. Vizual komponentlərin metodları

Vizual komponentlərlə əlaqədar olan çoxlu sayda metodlar vardır ki, onların köməyi ilə obyektləri yaratmaq, ləğv etmək, rəngləmək olar, həmçinin onları gizlətmək, əks etdirmək və digər əməliyyatları yerinə yetirmək olar.

Bütün vizual komponentlər üçün ümumi olan metodlara baxaq.

`SetFocus` prosedurası pəncərəli idarəedici elementdə giriş fokusunu təyin edir. Əgər verilmiş vaxt anında idarəedici element giriş fokusunu ala bilmirsə, səhv əmələ gəlir. Ona görə də məqsədəuyğundur ki, komponentin aktivləşməsi imkanı `CanFocus: Boolean;` - funksiyası vasitəsi ilə yoxlanılsın. Əgər idarəedici element giriş fokusunu ala bilirsə bu funksiyanın qiyməti `True`, əks halda `False` olur. İdarəedici element giriş fokusunu ala bilmir o vaxt ki, o aktivləşməmiş vəziyyətdə olsun, yəni `Enabled` xassəsi `False` qiymətini alsın.

Misal. `ListBox3` – siyahısında giriş fokusunu almalı:

```
if ListBox3.CanFocus then ListBox3.SetFocus;
```

Komponentin məzmununu silmək üçün `Clear` metodundan istifadə edilir.

Məsələ, `Memo1` və `ListBox1` komponentlərinin məzmunlarının silinməsi üçün aşağıdakı sətirləri yazmaq lazımdır.

```
LitsBox1.Clear;
Memo1.Clear;
```

Bunlardan əlavə `Refresh` metodu vardır ki, bu metod idarəedici elementləri təzələmək üçün (elementdəki xəyalı silmək üçün) istifadə edilir. Komponentin üzərində şəkillər çəkdikdə proqramçı vizual komponentin oblastında şəkil çəkməni idarə etmək üçün istifadə edə bilər. Bu metod adətən avtomatik çağırılır.

`Perform` metodu məlumatları idarəedici elementə göndərmək üçün istifadə olunur.

`Perform` funksiyasının parametrləri aşağıdakı kimidir:

`Perform(msg:Cardinal;Wparam,Lparam:Longint):` `Longint:Msg` – parametri vasitəsi ilə kodu verilən məlumat göndərilir. `Wparam, Lparam` –parmetrlərində isə əlavə məlumatlar olur.

Misal.

```
Label1.Caption:=İnttoStr(ListBox1.
Perform(LB_GetCount,0,0));
```

`ListBox1` – siyahısına `LB – Count` kodlu məlumatını göndərir.

Nəticə `Label1`-ə çıxarılır.

6.12. İnformasiyanın daxil edilməsi və redaktəsi

6.12.1. Birsətirli redaktorlar

İnformasiyanın daxil edilməsi və redaktəsi xüsusi sahələrdə yerinə yetirilir. Bunun üçün müxtəlif komponentlərdən, məsələ, `Edit`, `MaskEdit`, `Memo`, `RichEdit` komponentlərindən istifadə edilir.

Birsətirli redaktor informasiyanı daxil etmək üçün olan sahədir. Bu sahədə mətnləri əks etdirmək və dəyişdirmək olar. Delphi-də bir neçə birsətirli redaktorlar vardır. Onlardan ən çox istifadə olunan `Edit` – komponentidir. `Edit` komponenti klaviaturadan simvolları daxil etməyə və onları redaktə etməyə imkan verir. Birsətirli redaktor *Enter* və *Esc* idarəedici düymələrinə reaksiya vermir. Simvolların registrini dəyişmək üçün `TEditCharCase` tipinə malik olan `CharCase` xassəsindən istifadə edilir. Bu xassə aşağıdakı qiymətlərdən birini ala bilər:

- `ecLowerCase` – mətnin simvolları aşağı registrin simvollarına çevrilir;
- `ecNormal` – simvolların registrləri dəyişmir (susma prinsipinə görə);
- `ecUpperCase` – mətnin simvolları yuxarı registrin simvollarına çevrilir.

`Edit` komponentinin köməyi ilə parolu daxil etdikdə `Char` tipinə malik olan `PasswordChar` xassəsindən istifadə olunur. Bu xassə giriş sahəsində əks olunacaq simvolu müəyyən edir. Mətni daxil edən zaman faktiki simvolların yerində bu simvol əks olunur. Məsələn,

```
Edit1.PasswordChar:='+';
Edit1.Text:='parol';
```

Bu halda redaktəetmə sətrində +++++ simvolları görünəcək və həqiqətdə isə `Text` xassəsi *parol* qiymətini alır. Susma prinsipinə görə `PasswordChar` xassəsi `#0` qiymətini alır və bu zaman redaktə olunan sətirdə real daxil olunan informasiya əks olunur.

MaksEdit komponenti də birsətirli redaktordur. `Edit` komponentindən fərqi odur ki, daxil olunan informasiyaya şablona (maskaya) görə məhdudiyyət qoyur. Şablonun (maskanın) köməyi ilə istifadəçi tərəfindən daxil edilən simvolların sayına və

tipinə məhdudiyyət qoymaq olar. Bundan əlavə daxil edilən informasiyaya əlavə simvollar (məsələ, vaxtı və tarixi daxil edən zaman ayırıcı işarələr və s.) qoymaq olar. Maskaya görə redaktə etməyin köməyi ilə telefon nömrələrini, vaxtı və tarixi, poçt indeksini və digər əvvəlcədən məlum formata malik olan informasiyaları daxil etmək əlverişlidir.

Maska `String` tipinə malik olan `EditMask` xassəsi vasitəsi ilə verilir. Maskanı tərtib etmək üçün şablonlar redaktorundan istifadə etmək lazımdır (`Input Mask Editor`). `EditMask` xassəsi üzərində siçanın düyməsini iki dəfə sıxmaqla bu redaktoru çağırmaq olar.

Daxil olan informasiyanı saxlamaq üçün `OnKeyPress` – hadisəsinin emalçısını istifadə etmək olar.

Misal. `Edit1` redaktoru üçün ancaq rəqəmlərdən ibarət olan simvolların daxil edilməsinə məhdudiyyətin qoyulması:

```
procedure TForm1.Edit1KeyPress(Sender:
TObject; Var key:Char);
begin
if (key<'0') or (key>'9') then key:=#0;
end;
```

Redaktə olunan sahədə ancaq bir sətir yerləşə bilər. Bu sətrin sonunu göstərən simvol iştirak etmir. Ona görə də <Enter> düyməsini sıxdıqda heç bir əməliyyat yerinə yetirilmir. Zəruri olan halda <Enter> düyməsini sıxmaqla əlaqədar olan əməliyyatı proqramçı özü müəyyən edir. Məsələ, `SetFocus` metodundan istifadə etməklə <Enter> düyməsini sıxdıqda informasiyanın daxil edilməsinin sona çatması əlamətini müəyyən etmək olar və digər idarəedici elementə keçmək olar. Oxşar əməliyyatı `ActiveControl` xassəsinə qiymətlər verməklə də yerinə yetirmək olar.

Misal. Birsətirli redaktorda <Enter> düyməsinin sıxılması ilə əlaqədar olan proseduranı yazaq:

```
procedure TForm1.Edit1KeyPress(Sender:
Tobject; Var Key:Char);
begin
if key=#13 then begin
key:=#0;
Form1.Active Control:=Edit 2;
end; end;
```

```
procedure TForm1.Edit2KeyPress(Sender:
Tobject; Var key:Char);
begin
if key=#13 then begin
key:=#0;
Edit3.SetFocus;
end;
end;
procedure TForm1.Edit3KeyPress(Sender:
Tobject; Var key:Char);
begin
if key=#13 then key:=#0;
end;
```

İnformasiya ardıcıl olaraq üç sahəyə Edit1, Edit2, Edit3 sahələrinə daxil edilir. Birinci və ikinci sahələrə informasiyanın daxil edilməsi qurtardıqdan sonra <Enter> düyməsini sıxdıqda avtomatik olaraq növbəti sahə aktivləşir. Üçüncü sahədən giriş foksu avtomatik olaraq heç yerə verilmir.

Redaktə olunan sahəyə daxiletmə prosesi qurtardıqdan sonra növbəti idarəedici elementə keçidi təmin etməyin əlverişli üsullarından biri <Enter> düyməsinin sıxılması ilə əlaqədar olan aşağıdakı proseduradan istifadə etməkdir:

```
procedure TForm1.AllEditKeyPress(Sender:
Tobject; Var key:Char);
begin
if key=#13 then begin
Form1.Select Next(sender:as TWin Control,
true,true);
Key:=#0;
end;
end;
```

<Enter> düyməsi sıxılan zaman Select Next – metodu yerinə yetirilərək və foksu növbəti idarəedici elementə verir.

```
Select Next (CurControl: TWin Control;
GoForward, CheckTabStop: Boolean)
```

prosedurunun üç parametri vardır. CurControl parametri pəncərəli idarəedici elementi, GoForward – foksun ötürülmə istiqamətini, CheckTabStop parametri TabStop xassəsinin nəzərə alınmasını göstərir.

6.12.2. Çoxsətirli redaktorlar

Çoxsətirli mətnlərlə işləmək üçün *Memo* komponentindən istifadə edilir. Çoxsətirli redaktor birsətirli redaktorun bütün imkanlarına malikdir. Əsas fərqi ondan ibarətdir ki, çoxsətirli redaktorda bir neçə sətir yerləşir. Çoxsətirli redaktorun məzmununa müraciət etmək üçün String tipinə malik olan Text xassəsindən istifadə edilir. Bu halda Memo komponentinin bütün məzmunu bir sətirdə təsvir olunur və <Enter> düyməsini sıxmaqla olan sətrin sonu əlaməti #13#10 kodları ilə müəyyən olunur.

Ayrı-ayrı sətirlərlə işləmək üçün TStrings tipinə malik olan Lines xassəsindən istifadə edilir.

Misal. Çoxsətirli redaktora aid əməliyyatlar:

```
Memo1.Lines[ 3] :='abc';
Memo2.Lines.Clear;
Memo3.Lines.Add('Yeni sətir');
```

Burada Memo1 redaktorunun 4-cü sətrinə yeni 'abc' qiyməti verilir (sətirlər 0-dan başlayaraq nömrələnir). Memo2 redaktorunun məzmunu tamamilə təmizlənir. Memo3 redaktorundakı mətnin sonuna yeni sətir əlavə edilir.

Memo komponentinin məzmununu mətn faylından yükləmək olar və ya mətn faylında saxlamaq olar. Bunun üçün TStrings tipinə malik olan aşağıdakı metodlardan istifadə etmək olar:

```
Load From File(const File Name:String);
SaveToFile(const FileTame:String);
```

File Name parametri oxumaq və yazmaq üçün olan mətn faylını müəyyən edir.

Misal. Memo1 komponentinə mətn faylından informasiyanın oxunması:

```
Memo1.Lines.Load From File('c:\text\infor1.txt');
```

Memo2 komponentindən mətn faylına informasiyanın yazılması:

```
Memo2.Lines.SaveToFile('c:\text\infor2.txt');
```

İnformasiyaya baxmağın əlverişli olması üçün TScrollStyle tipinə malik olan ScrollBars xassəsindən istifadə etməklə baxış zolağı vermək olar. bu xassənin qiymətləri aşağıdakılar ola bilər:

- ss None – baxış zolağı yoxdur (susma prinsipi);
- ss Horizontal – aşağıdan üfüqi baxış zolağına malikdir;
- ss Vertical – sağdan vertikal baxış zolağına malikdir;
- ss Both – hər iki baxış zolağı vardır.

Bunlardan əlavə Memo komponentinin sahəsindəki mətn sağ sərhəddə görə (Alignment xassəsi taRightJustify – qiymətini alır), sol sərhəddə görə (Alignment xassəsi taLeftJustify qiymətini alır) və mərkəzə görə (Alignment xassəsi taCenter qiymətini alır) düzləndirilə bilər.

Birsətirli redaktordan fərqli olaraq Memo komponenti <Enter> və <Tab> düymələrinin sıxılmasına reaksiya verir. Bunun üçün uyğun olaraq

WantReturns və WantTabs xassələrinə true qiyməti verilməlidir.

RichEdit komponenti vasitəsi ilə də mətnləri redaktə etmək olar və o, bütün formatlaşma vasitələrinə malikdir. Bu komponentin sahəsində yerləşən mətn RTF (Rich Text Format) – formatına uyğunlaşandır və Windows mühitində bütün mətn prosessorları tərəfindən istifadə edilə bilər.

6.12.3. Redaktə üçün olan komponentlərin ümumi xassələri, hadisələri və metodları

İnformasiyanı redaktə etmək üçün olan komponentlər birbirinə oxşardır və onların hamısı üçün ümumi olan xassələr, hadisələr və metodlar vardır.

Redaktorun məzmununda ixtiyari dəyişiklik aparan zaman TNotifyEvent tipinə malik olan OnChange hadisəsi yaranır. Bu hadisədən giriş sahəsində yerləşən informasiyanın operativ olaraq düzgünlüyünə nəzarət etmək üçün istifadə edilir.

İnformasiyanı redaktə etmək üçün olan komponentlər aşağıdakı xassələrə malik ola bilərlər:

o Boolean tipinə malik olan Modiefid xassəsi. İnformasiyanın modifikasiyası zamanı true qiymətini alır. Bu xassədən modifikasiya olunan informasiyanın diskdə saxlanmasını yoxlamaq üçün istifadə etmək olar. Məsələn:

```
o if Memo1. Modiefid then <informasiya saxlanılır>;
```


o İnteger tipinə malik olan MaxLength xassəsi.

Redaktə olunan sahəyə daxil edilən simvolların maksimal sayını göstərmək üçün istifadə olunur. Susma prinsipinə görə Maxlength=0 və istifadəçi tərəfindən daxil edilən simvolların sayı məhdud deyildir.

o Auto Select (boolean tipli), SelStart və Sellength (integer tipli) Seltext (String tipli) xassələrindən redaktə olunan sahədən mətn fraqmentləri seçmək üçün istifadə edilir. AutoSelect xassəsi mətnlərin avtomatik olaraq seçildiyini, SelStart və Sellength xassələri sətrin başlanğıcını və uzunluğunu, Seltext xassəsi isə ayrılmış mətn fraqmentini müəyyən edir.

Proqram vasitəsi ilə mətn ayrılan zaman Boolean tipli HideSelection xassəsinə false qiymətini vermək lazımdır. Əks halda HideSelection true qiymətinə malik olar və fokus digər idarəedici elementə keçən zaman ayrılan mətn fraqmenti görünmür.

Misal.

```
Memo1.SelStart:=18;
Memo1.Sellength:=8;
Memo1.Seltext:='abcdefgh';
if pos('xyzt',Edit1.text)< >0 then
begin
Edit1.HideSelection:=false;
Edit1.SelStart:=pos('xyzt',Edit1.text)-1;
Edit1.Sellength:=length('xyzt');
```

Memo1 komponentində 18-cidən başlayaraq 8 simvol abcdefgh – sətri ilə əvəz olunur. Edit1 komponentində xyzt sətri axtarılır və axtarış müsbət nəticə verən zaman həmin sətir qeyd olunur.

Ayrılmış mətn fraqmentləri ilə işləmək üçün aşağıdakı metodlardan istifadə etmək olar:

SelectAll, CopytoClipBoard, CuttoClipBoard, Paste From ClipBoard. SelectAll metodu redaktə olunan elementdə bütün mətn seçir. CopytoClipBoard və CuttoClipBoard metodları uyğun olaraq ayrılmış mətni dəyişmə buferinə köçürür və ya kəsib atır.

Məsələn,

```
Memo1.CuttoClipBoard;
```

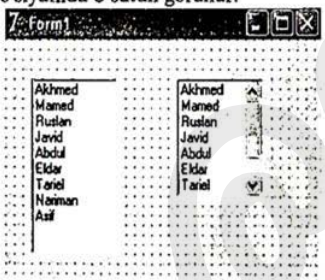
qeyd olunmuş mətni dəyişmə buferində yerləşdirir.
`PasteFromClipBoard` – metodu dəyişmə buferində yerləşən mətni redaktə olunan elementdə kursorun durduğu yerdə yerləşdirir.

6.12.4. Sadə və kombinasiyalı siyahılar

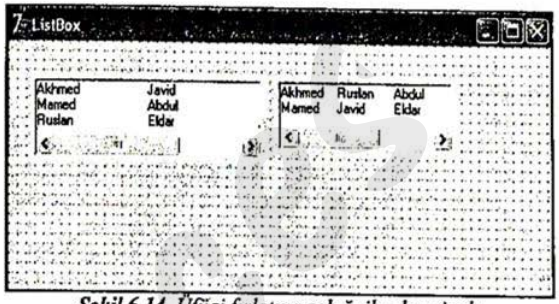
Siyahı bir-biri ilə qarşılıqlı əlaqəli olan nizamlanmış elementlər yığımıdır. Bu elementlər mətnlərdən ibarət olan sətirlərdir. Siyahının ayrı-ayrı sətirlərini silmək və ya yeni sətir əlavə etmək üçün imkanlar vardır.

Sadə siyahı düzbucaqlı oblastdan ibarətdir və siyahının elementləri bu oblastda yerləşir. Sadə siyahı ilə işləmək üçün `ListBox` – komponentindən istifadə olunur. Əgər sətirlərin sayı çox olarsa (siyahı görünən oblasta yerləşməzsə) onda fırlatma zolağından istifadə etmək olar. Fırlatma zolağının istiqaməti sütunları müəyyən edən integer tipli `Columns` xassəsindən də asılıdır. Fırlatma zolağı üfüqi və şaquli istiqamətlərdə ola bilər. Susmaya görə `Columns=0` və bu halda siyahının elementləri bir sütunda yerləşir və zəruri olan halda vertikal fırlatma zolağı görünür və ya görünmür (şəkil 12.1).

Əgər `Columns≥1` isə onda siyahının oblastında həmişə üfüqi fırlatma zolağı iştirak edir və siyahının elementləri elə sütunlara bölünür ki, üfüqi fırlatma zolağının köməyi ilə onun bütün elementlərini görmək mümkün olsun. Şəkil 12.2-də 9 familiyadan ibarət iki siyahı göstərilmişdir. Sol tərəfdəki siyahıda `Columns=2` və siyahıda eyni zamanda 2 sütun görünür, sağ tərəfdəki siyahıda `Columns=3` və siyahıda 3 sütun görünür.



Şəkil 6.13. Sadə siyahıların variantları



Şəkil 6.14. Üfüqi fırlatma zolağı ilə olan siyahı

Sadə siyahının stili Style xassəsi vasitəsi ilə verilə bilər. Bu xassə aşağıdakı qiymətləri ala bilər:

- `LbStandart` – standard stil (susma prinsipinə görə);
- `LbOwnerDrawFixed` – `ItemHeight` xassəsi ilə müəyyən olunan qeyd olunmuş hündürlüyə malik elementlərlə olan siyahı;
- `LbOwnerDrawVariable` – müxtəlif hündürlüyə malik elementlərlə olan siyahı
- Siyahı adi çərçivəyə malik ola da bilər və ya olmaya da bilər. Çərçivənin olmasını `BorderStyle` xassəsi müəyyən edir.
- `bsNone` – çərçivə yoxdur;
- `bsSingle` – çərçivə var (susma prinsipinə görə).

Kombinasiyalı siyahı redaktə olunan sahə ilə siyahını birləşdirir. İstifadəçi siyahıdan qiyməti seçə bilər və ya onu redaktə olunan sahəyə daxil edə bilər. Kombinasiyalı siyahı ilə işləmək üçün `ComboBox` – komponentindən istifadə edilir.

`ComboBox` – komponenti vasitəsi ilə müəyyən olunan siyahı həm sadə siyahı, həm də açılan siyahı şəklində ola bilər. Açılan siyahı bağlanmış vəziyyətdə görünür və ekranda az yer tutur. Şəkil 6.15-də `ComboBox` komponenti həm bağlanmış, həm də açılmış şəkildə göstərilmişdir.

Sadə siyahıdan fərqli olaraq kombinasiyalı siyahı üfüqi fırlatma zolağına malik olmur və bu siyahıdan ancaq bir qiymət seçmək olar. Kombinasiyalı siyahının xarici görünüşü və özünü aparması `TComboBoxStyle` tipinə malik olan `Style` xassəsi

vasitəsi ilə verilir. Bu xassə susma prinsipinə görə CsDropDown qiymətini alır. Bu qiymət redaktə olunan SA-həyə malik açılan siyahının olmasını təmin edir. Style xassəsi sadə siyahıda olduğu kimi digər qiymətlər də ala bilər.

Integer tipinə malik olan DropDownCount xassəsi ekranda açılan siyahıda sətirlərin sayını müəyyən edir. Susma prinsipinə görə bu xassənin qiyməti 8 qəbul edilir. Əgər xassənin qiyməti Items xassəsinin alt xassəsi olan Count xassəsinin (siyahıda sətirlərin sayını müəyyən edir) qiymətindən böyükdürsə, onda açılan siyahıda avtomatik olaraq vertikal fırlatma zolağı görünür. Əgər siyahının ölçüsü DropDownCount xassəsinin qiymətindən kiçikdirsə, onda siyahının əks olunduğu oblast kiçildilir.

Boolean tipinə malik olan DroppedDown xassəsi siyahının açılan olub-olmamasını müəyyən edir. Əgər xassə True qiymətini alarsa, siyahı açılmış şəkildə əks olur, əks halda siyahı bağlanmış şəkildə əks olunur.

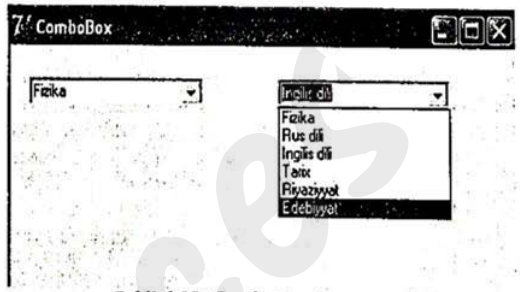
DroppedDown xassəsi Style xassəsinin CsSimple qiymətini almadığı halda fəaliyyət göstərir.

Açılan siyahının əks olunduğu halda OnDropDown hadisəsi (TNotifyEvent tipinə malik olan) baş verir. Proqramçı DroppedDown xassəsinə uyğun qiymətlər verməklə proqramın yerinə yetirilmə prosesində siyahıların açılmasını və bağlanmasını idarə edə bilər.

Misal. Siyahının açılmasının və bağlanmasının idarə edilməsi.

```
Procedure TForm5.btnListopen Click(Sender:
Tobject);
  begin
  ComboBox7.DroppedDown:=true;
  end;
  procedure TForm5.btnListCloseClick(Sender:
                                     Tobject);
  begin
  ComboBox7.DroppedDown:=false;
  end;
```

btnListOpen – düyməsini sıxdıqda ComboBox7 – siyahısı açılır, btnListClose düyməsini sıxdıqda bağlanır.



Şəkil 6.15. ComboBox komponenti

6.12.5. Siyahılar üzərində əməliyyatlar

Siyahılar üzərində əməliyyatlar dedikdə siyahının elementlərinə yeni qiymətin mənimsədilməsi, siyahıya yeni sətrin əlavə edilməsi, siyahıdan elementin silinməsi, siyahının elementlərinin nizamlanması və bu kimi əməliyyatlar başa düşülür. Həm sadə, həm də kombinasiyalı siyahılar bir-birinə çox oxşar olduqları üçün onların ümumi xassə, metod və hadisələri vardır. Siyahılar üzərində göstərilən əməliyyatların yerinə yetirilməsində ***İtems*** xassəsi əsas rol oynayır.

İtems xassəsi TString tipinə malik olub sətirlər massivini, siyahıdakı elementlərin sayını və siyahının məzmununu təyin edir. TStrings tipinin çoxlu sayda törəmə sinifləri vardır, məsələ, TStringList.

Siyahının hər bir elementi sətirdir və sətirlərin hər birinin İtems massivində nömrəsi vardır. Sətirlər İtems massivində sıfırdan başlayaraq nömrələnir. İtems[0] – siyahının birinci elementini, İtems[1] – siyahının 2-ci elementini və s. göstərir.

Integer tipinə malik olan Count xassəsi siyahıda elementlərin sayını göstərir. Birinci elementin nömrəsi sıfıra, axırıncı elementin nömrəsi isə Count 1-ə bərabərdir.

Misal. ListBox1. – siyahısının elementlərinə yeni qiymətlərin verilməsi.

```
Var n:integer;
...
for n:=0 to ListBox1.İtems.Count-1 do
```

```
ListBox1.İtems[n]:='Sətrin nömrəsi'+
                                    İnttoStr(n);
```

Add və İnsert metodları siyahıya sətirlərin əlavə olunmasını təmin edir. Add(const s:string):integer; - funksiyası s – parametri ilə verilən sətiri siyahının sonuna əlavə edir və funksiyanın qiyməti yeni sətrin siyahıda yeri (nömrəsi) olur.

İnsert(İndex:integer; const s:string); – proseduru siyahıya İndex parametri ilə müəyyən olunan yerə s – parametri ilə verilən sətri əlavə edir.

Misal. ComboBox1 siyahısına 'Button1 düyməsi sıxılmışdır' – sətrinin əlavə edilməsi.

```
Prosedure TForm1.Button1Click(Sender:
                                        Tobject);
ComboBox1.İtems.Add('Button1 düyməsi
                                sıxılmışdır');
end;
```

Add Strings(Strings:TStrings) – prosedurası siyahının sonuna strings – parametri ilə təyin olunan bir qrup sətir əlavə edir.

Assign(Source:TPersistent); – prosedurası uyğun tipli obyektlərin birini digərinə mənimsədir. Siyahılara tətbiqi zamanı isə bir siyahıdakı informasiyanı digərinə köçürür. Əgər siyahıdakı elementlərin sayı bərabər deyilsə, onda əvəz olunan siyahıdakı elementlərin sayı köçürülən siyahıdakı elementlərin sayına bərabər götürülür.

Equals(Strings:TStrings):Boolean; – funksiyası iki siyahıda eyni mətn informasiyalardan ibarət olan sətirlərin olmasını yoxlayır. Əgər siyahıların məzmunları bərabər olarsa, funksiyanın qiyməti true, əks halda isə false olur.

Misal. İki siyahının bərabərliyinin yoxlanması.

```
if not(ListBox2.İtems.Equals(ListBox1.İtems)
then begin
ListBox2.Clear;
ListBox2.İtems.Add Strings(ListBox1.İtems);
end;
```

və ya

```
if not (ListBox2.İtems.Equals(ListBox1.İtems)
then ListBox2.İtems.Assign(ListBox1.İtems);
```


Əgər siyahılar bərabər deyilsə, ListBox2-nin məzmunu ListBox1-ə bərabər olur.

Siyahıdan elementləri silmək üçün Delete və Clear metodlarından istifadə edilir.

Delete(İndex:İnteger); - prosedurası nömrəsi İndex parametri ilə verilən elementi silir.

Misal. Siyahıdan elementin silinməsi.

```
procedure TForm1.Button2 Click(Sender:
                                        Tobject);
begin
ComboBox1.İtems.Delete(4);
end;
```

Button2 düyməsini sıxan zaman ComboBox siyahısından 5-ci element silinir.

Clear proseduru siyahının bütün elementlərini silərək onu təmizləyir.

Misal. Siyahının məzmununun silinməsi

```
procedure TForm1.btnpersonalClearClick
(Sender:Tobject);
begin
Lb Personal.İtems.Clear;
end;
```

btnpersonalClear düyməsini sıxan zaman LBPersonal siyahısı silinir.

Move(CurİndexNewİndex:İnteger); – prosedurası CurİndexNömrəsi ilə müəyyən olunan elementi NewİndexNömrəsi ilə müəyyən olunan mövqeyə gətirir. Əgər göstərilən nömrə siyahıdakı elementlərin sayından böyük olarsa səhv haqqında məlumat verilir.

Elementin siyahıda axtarılması

İndex of (const s:string):integer;

– funksiyasının köməyi ilə yerinə yetirilir. Siyahıda s – sətrinin olması yoxlanılır. Əgər s – sətri siyahıya daxildirsə, funk-siyanın qiyməti tapılan sətrin nömrəsinə, əks halda (s - sətri siyahıda yoxdursa) funksiyanın qiyməti 1-ə bərabər olur.

TStrings tipinə malik olan SaveToFile və LoadFromFile metodları mətn faylları ilə işləmək üçün tətbiq olunur.

SaveToFile(const FileName:string);

– prosedurası siyahının elementlərini FileName faylında saxlayır. Əgər bu fayl diskdə yoxdursa, yenidən yaradılır.

`Load From File(const FileName:string);`
– prosedurası göstərilən mətn faylından siyahını oxuyur. Bu zaman əvvəlki siyahı silinir. Əgər fayl diskdə yoxdursa səhv əmələ gəlir.

Misal 1. Siyahının məzmununun saxlanılması

```
ListBox3.SaveToFile ('S:\Name\Fam.txt');
```

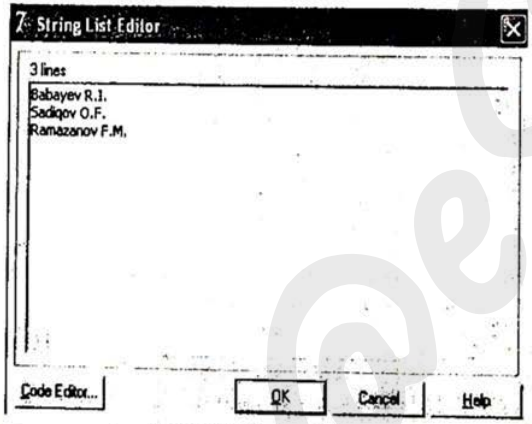
`ListBox3` – siyahısının məzmunu `C:\Name` kataloqunda `Fam.txt` faylına yazılır.

Misal 2. Siyahının məzmununun fayldan yüklənməsi;

```
procedure TForm1.FormGreat(Sender:Tobject);
ComboBox2.İtems.Load From
File('C:\text\personal.txt');
end;
```

`Personal.txt` faylında yerləşən siyahı `ComboBox2` siyahısına yüklənir.

Proqramın tərtibi zamanı siyahıdakı sətirlərin dəyişdirilməsini `StringListEditor` redaktoru vasitəsi ilə yerinə yetirmək olar. Bu redaktoru obyektlər inspektoru pəncərəsində `TString` tipinin qiymətinin üzərində siçanın sol düyməsini iki dəfə sıxmaqla çağırmaq olar (şəkil 12.4). Sətir redaktoru siyahıya yeni sətrin əlavə edilməsinə, sətrin silinməsinə və sətrin məzmununun dəyişməsinə imkan verir.



Şəkil 6.16. Sətir redaktoru

Siyahılar əlifba sırası ilə nizamlana bilərlər. Siyahının *nizamlanması* üçün `Boolean` tipli `Sorted` xassəsindən istifadə edilir. Xassənin qiyməti false olduqda (susma prinsipinə görə `false` qəbul edilir) elementlər siyahıda düşdükləri ardıcıllıqla yerləşirlər. `True` olduqda isə elementlər əlifba sırasına görə avtomatik olaraq nizamlanırlar. `Sorted` xassəsi dinamik deyil, statik olaraq faliyyət göstərir. Bu o deməkdir ki, nizamlanmış siyahıya `Add` və ya `İnsert`-lə əlavə olunan yeni sətirlər siyahının axırına və ya göstərilən mövqeyə əlavə olunurlar. Siyahını nizamlamaq üçün `Sorted` xassəsinə əvvəlcə `False`, sonra isə `True` mənimsətmək lazımdır:

```
ListBox1.Sorted:=False;
ListBox1.Sorted:=True;
```

İstifadəçi siyahıda ayrı-ayrı elementləri siçanın və klaviaturanın köməyi ilə seçə bilər. Siyahıda seçilmiş element `İnteger` tipinə malik olan `İtemİndex` xassəsi ilə müəyyən olunur. *Məsələn,*

```
ListBox2.İtem İndex:=3;
```

Bu operator `ListBox2`-də 4-cü sətri seçir və bu sətir ekranda əks olunur.

`ListBox1`-də seçilmiş sətri ekranda əks etdirmək üçün aşağıdakı operatoru yazmaq olar:

```
Label1.Caption:='siyahıda'+İnttoStr
(ListBox1.İtemİndex)+'sətri seçilmişdir';
```

`İnteger` tipli `SelCount` xassəsi siyahıda seçilən elementlərin sayını müəyyən edir.

Siyahıdan seçilən sətirlərin sayını müəyyən etmək üçün `Boolean` tipli `Selected[Index:integer]` – xassəsindən istifadə etmək olar. Bu xassəyə məntiqi qiymətlərdən ibarət olan massiv kimi baxmaq olar. Əgər `İndex` nömrəli sətir seçilərsə `Selected True` qiymətini, əks halda `False` qiymətini alır.

Misal 1. Siyahının seçilmiş elementlərinin `'sətir seçilmişdir'` mətni ilə əvəz edilməsi:

```
var i:integer;
for i:=0 to ListBox3.İtems.Count-1 do
if ListBox3.Selected[i] then
ListBox3.İtems[i]:='sətir seçilmişdir';
```

Misal. Siyahının elementlərinin proqram yolu ilə seçilməsi

```
ListBox4.Selected[1]:=true;
ListBox4.Selected[3]:=true;
```

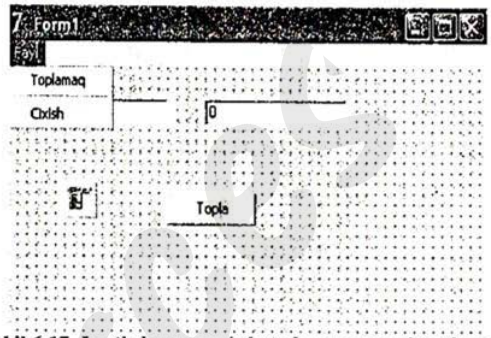
`ListBox4` – siyahısında 2-ci və 4-cü sətir seçilmişdir. Siyahının elementləri seçildikdə `OnClick` hadisəsi baş verir ki, bu hadisədən seçilmiş sətirlər üzərində əməliyyatlar aparmaq üçün istifadə etmək olar:

```
procedure TForm1.ListBox4 Click(Sender:
                                    Tobject);
begin
Label3.Caption:=
        ListBox4.Items[ ListBox4.ItemIndex] ;
end;
```

prosedurası yerinə yetirildikdə `Label3`-də `ListBox4` – siyahısından seçilən element əks olunacaqdır.

6.13. Əsas menyunun və köməkçi menyunun (kontekst menyunun) hazırlanması

Tətbiqi proqramların idarə olunmasını təmin etmək üçün əsas və köməkçi (kontekst) menyulardan istifadə edilir. Delphi sistemində hər iki menyunu hazırlamaq üçün vasitələr vardır. Proqrama əsas menyunu əlavə etmək üçün ***Standard*** səhifəsində yerləşən `TMainMenu` komponentindən istifadə etmək lazımdır. Proqrama menyu əlavə etmək üçün `TMainMenu` komponentini formanın istənilən yerində yerləşdirmək lazımdır. `TMainMenu` vizual olmayan komponentdir. Formalar pəncərəsində `MainMenu1` obyektinin üzərində iki dəfə siçanın düyməsini sıxmaqla menyu redaktorunu çağırmaq olar. İlk anda menyu boş olur. Obyektlər İnspektoru pəncərəsində `LocalLizable` kateqoriyasını açaraq `Caption` xassəsinə menyunun birinci punktunun adını veririk (məsələn, `&Fayl`) və *Enter* düyməsini sıxırıq. Yenidən *Enter* düyməsini sıxdıqda `Caption` xassəsinə yenidən keçid olur və menyunun yeni punktu formada görünür. Bu yeni punkta növbəti adı daxil edirik (məsələ, `TOPLA`) və *Enter* düyməsini sıxırıq. Bu qayda ilə menyunun növbəti punktları hazırlanır. Adətən menyuların axırıncı punktu `&ÇIXIŞ` olur. Menyulara yeni menyu əlavə etmək üçün *İnsert*, silmək üçün isə *Delete* düyməsindən istifadə edilir. Menyu hazır olduqdan sonra redaktoru bağlamaq lazımdır. Bu zaman hazırlanan menyu formada görünür (şəkil 6.17).



Şəkil 6.17. Layihələşmə mərhələsində menyunun hazırlanması

Menyu punktlarının adlarının verilməsində – (defis) işarəsindən istifadə etmək üçün `Caption` xassəsində birinci mövqeyə – (defis) yazmaq lazımdır. Menyunun istənilən punktunun fəaliyyətini təmin etmək üçün həmin punkta uyğun olan hadisələrin emalçısı prosedurasını hazırlamaq lazımdır. Məsələ, tutaq ki, menyunun `ÇIXIŞ` punktunu seçdikdə cari forma bağlanmalıdır. Bunun üçün menyu redaktorunda `ÇIXIŞ` punktunun üzərində düyməni iki dəfə sıxmaq lazımdır. Bu zaman Delphi sistemi avtomatik olaraq aşağıdakı kimi yeni metod yaradır:

```
procedure TMyForm.N4Click(Sender:Tobject);
begin
end;
```

Burada `N4` – proqramın daxilində ***Çıxış*** menyusunun identifikatorudur.

`TMyForm` sinfinin təsvirində o (`N1`, `N2`, `N3` – punktları ilə yanaşı) aşağıdakı kimi təsvir olunur:

```
N4:TMenuItem;
```

`TMenuItem` sinfi menyu punktlarını təsvir etmək üçün istifadə edilir.

Çıxış punktunun seçilməsi zamanı formanı bağlamaq üçün (proqramı dayandırmaq üçün) `TForm` sinfinə aid olan `Close` metoduna müraciət etmək lazımdır:

```
procedure TMyForm.N4Click(Sender:Tobject);
begin
close;
end;
```

Bu metod cari formanın korrekt bağlanmasını təmin edir. Əgər proqramı bağlamaq üçün sistemin bağlayan düymələrindən istifadə edilərsə, proqram korrekt bağlanmaya bilər. Proqramın korrekt bağlanmasını yoxlamaq üçün `Boolean` tipli `Close Qiery` funksiyasından istifadə etmək olar:

```
procedure TMyForm.N4Click(Sender:Tobject);
begin
if Close Query then Close;
end;
```

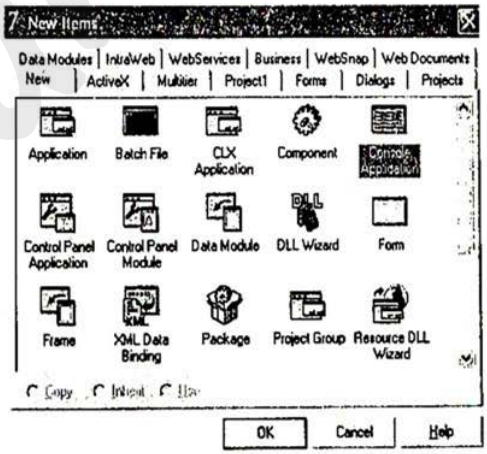
Siçanın sağ düyməsini sıxmaqla alınan kontekst menyu bir çox proqramlar üçün əlverişli imkanlar yaradır. Kontekst menyunu hazırlamaq üçün `TPopupMenu` komponentindən istifadə edilir. Əsas menyudan fərqli olaraq kontekst menyunu istənilən pəncərəli komponent üçün hazırlamaq olar. `TMainMenu` komponentində olduğu kimi `TPopupMenu` komponenti formada yerləşdirilir və menyu sətirləri əsas menyunun sətirlərinə oxşar olaraq hazırlanır. Ümumiyyətlə, kontekst menyunun hazırlanması və xassələri `TMainMenu` sinfindən olan menyuların hazırlanması qaydalarından fərqlənmir. Kontekst menyu punktları hazırlandıqdan sonra onun hansı obyektə aid olması obyektin xassələr siyahısını ekrana gətirməklə və `PopupMenu` xassəsinə yaratdığımız kontekst menyulardan birini qeyd etməklə müəyyən edilir. Menyu sətirlərinin fəaliyyətini təmin etmək üçün əsas menyuda olduğu kimi hadisələri emal edən proseduraları yazmaq lazımdır.

6.14. Konsol proqramların hazırlanması

Pascal dilinin əsas operatorlarından istifadə etməklə Delphi sistemində proqramlar hazırlamaq olar. Belə proqramlar MS DOS əməliyyat sisteminin stilinə uyğundur və onlar **Konsol** proqramlar adlanır. İstifadəçi Konsol proqramlara qrafik interfeysin köməyi ilə müraciət edə bilmir. İstifadəçi ilə informasiya mübadiləsi üçün daha sadə vasitələrdən istifadə edilir. Adətən Konsol proqramlarda `Readln` (verilənləri daxil etmək üçün) və `Writeln` (verilənləri yazmaq üçün) proseduralardan istifadə edilir.

Konsol proqramlar hazırlamaq üçün *File → New → Other* (*Файл→ Создать→ Другое*) əmrini vermək və açılan `New Items` (*Создание программы*) pəncərəsində `Console Application` (*Консольное приложение*) proqramını seçmək lazımdır (şəkil 6.18). Bu zaman Delphi sistemi avtomatik olaraq mətn redaktorunda aşağıdakı kimi ilkin kod generasiya edir:

```
Proqram Project1;
{ $APPTYPE CONSOLE}
USES
SYSUTİLS;
begin
{ TODO - OUSER - c Console Main:İnsert code
here}
end.
```



Şəkil 6.18. Konsol rejiminin seçilməsi

İlkin koda avtomatik olaraq `Project1` adı verilir. `{ $APPTYPE CONSOLE}` direktivi kompilyatora proqramın konsol proqram olması haqqında məlumat verir. `Begin ... end` arasında proqramçı öz ilkin proqram mətnini yazır.

Proqramı yerinə yetirməmişdən əvvəl onun ilkin mətnini

saxlamaq üçün alətlər panelində *Save All* (*Сохранить Все*) düyməsindən istifadə etmək olar. Bu zaman Delphi sistemi `Project1` əsas faylının saxlanma yerini soruşur. İstənilən uyğun qovluğu göstərmək olar.

Konsol proqramı kompilyasiya etmək və yerinə yetirmək üçün *F9* düyməsini sıxmaq lazımdır. Bu zaman ilkin mətn saxlanılan qovluqda `Project1.exe` – yerinə yetirilən faylı görünür. Ekranda konsol proqramın pəncərəsi açılır. Bu proqram Delphi mühitində avtomatik olaraq yerinə yetirilə bilir. *Enter* düyməsini sıxmaqla yenidən Delphi mühitinə qayıtmaq olar. Delphi 7 mühitində Pascal-ın imkanlarını nümayiş etdirmək üçün aşağıdakı misala baxaq.

$y=(a^2+b^2)/2$ – ifadəsinin qiymətini hesablamaq üçün aşağıdakı kimi konsol proqram yazmaq olar:

```
proqram Project1;
{ $APPTYPE CONSOLE }
USES SYSUTİLS;
Var a,b,y:integer;
Begin
Readln(a,b);
y:=(a*a+b*b)/2;
Writeln(y);
Readln
end.
```

Proqramın sonunda yazılan Readln operatoru proqramdakı əməliyyatlar yerinə yetirilib qurtardıqdan sonra pəncərənin bağlanmasına imkan vermir və *Enter* düyməsinin sıxılması gözlənilir. Bu halda proqramın nəticəsinə baxmaq imkanı əldə edirik.

F9 düyməsini sıxdıqdan sonra proqramın pəncərəsi ekranda görünür. Bu pəncərədə, məsələn:

2 2

- ədədlərini daxil edirik.

Əgər indi *Enter* düyməsini sıxsaq nəticəni ekranda görərik:

4

Proqramın işini yekunlaşdırmaq üçün yenidən *Enter* düyməsini sıxmaq lazımdır (Şəkil 6.19).

Şəkil 6.19. Konsol proqramın nəticəsi

Qeyd edək ki, konsol proqram öz işini qurtardıqdan sonra onun pəncərəsi bağlanmaya bilər. Konsol proqramın pəncərəsini avtomatik olaraq bağlamaq üçün proqramın menyusunda *Свойства* punktunu seçib açılan dialoq pəncərədə *Закрывать* sözünə bayraqcıq qoymaq lazımdır.

ƏDƏBİYYAT

1. Əliyev A.Y. İnformatika, hesablama texnikası və proqramlaşdırmanın əsasları. Bakı, Mütərcim, 1998, 216 s.
2. Əliyev A.Y. İnformatika və proqramlaşdırma. Bakı, Mütərcim, 2008, 404 s.
3. Mehdiyeva Q.Y., Əliyev A.Y., Piriverdiyev V.Ə. Proqramlaşdırma üzrə məsələlər. Bakı, Bakı Universiteti, 2004, 106 s.
4. Əliyev Ə.Ə., Kazımov C.K., Kompüterin arxitekturası və əməliyyat sistemləri. Bakı, Mütərcim, 2007, 132 s.
5. Алиев А.А. Распределенные системы обработки информации. Баку, 2003, 130 с.
6. Мехтиева Г.Ю., Алиев А.Ю., Пиривердиев В.А. Практикум по программированию. Баку, Бакинский Университет, 2004, 113 с.
7. Архангельский А.Я. Программирование в Delphi 7. Москва, Бинот, 2003. 1152 с.
8. Бобровский С. Delphi 5. Учебный курс. Питер. 2000, 640 с.
9. Бобровский С. Delphi 7. Учебный курс. Питер, 2007, 736 с.
10. Гофман В., ХомоненкоА. Delphi 5. Санкт-Петербург, 2000, 800 с.
11. Гусева А.И. Технология межсетевых взаимодействий М., 1997, 272 с.
12. Дуглас Э.Камер Компьютерные сети и INTERNET. Разработка приложений для INTERNET: Пер. с англ. М., 2002, 640 с.
13. Дэвис Д., Барбер Д.и др. Вычислительные сети и сетевые протоколы М., 1982, 563с.
14. Ляхович В.Ф., Крамаров С.О. Основы информатики. Ростов, Феникс, 2004, 704 с.
15. Мартин Дж. Вычислительные сети и распределенная обработка данных, Т1-2-М: - 1985. – 525с.
16. Марк Спортак, Френк Паппас и др. Компьютерные сети и сетевые технологии: Пер. с англ. К., 2002, 736 с.
17. Новиков Ю.В., С.В.Кондратенко Локальные сети: архитектура, алгоритмы, проектирование. М., 2001, 312 с.

18. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. 2-е изд. : Питер, 2003, 864 с.
19. Основы современных компьютерных технологий: Учебное пособие /Под ред. Проф.Хомоненко А.Д., С.-П., 1998, 448 с.
20. Петров А.В., Алексеев В.Е., Ваулин А.С. и другие. Вычислительная техника и программирование. М. Высшая школа, 1990, 479 с.
21. Пярнпуу А.А. Програмирование на современных алгоритмических языках. М., Наука, 1990, 384 с.
22. Фаронов В.В. Turbo Pascal 7.0. М. изд. ОМД Групп, 2003, 616 с.
23. Фаронов В.В. Delphi 4. Учебный курс. М.: «Намедни», 1999, 448 с.
24. Фаронов В. Delphi 6. Санкт-Петербург, 2002.
25. Фигурнов В.Э. IBM PC для пользователя краткий курс. М., ИНФРА, 1997, 480с.
26. Федоров А.Г. Создание Windows – приложений в среде Delphi. М.: ТОО фирма «Компьютер Пресс», 1995, 287 с.

MÜNDƏRİCAT

Доктор технических наук, профессор
Алиев Алекпер Али Ага оглы

Кандидат физико-математических наук, доцент
Алиев Айдын Юнус оглы

Кандидат физико-математических наук, доцент
Кязимов Джаваншир Кязим оглы

Основы информатики

Учебное пособие для высших учебных заведений

Баку, Мутарджим, 2008

Çapa imzalanıb: 10.12.09
Format: 60x84 1/16. Qarnitur: Times.
Ofset kağızı: əla növ. Həcmi: 18.75 s.ç.v. Tiraj:150
Sifariş № 93. Qiyməti müqavilə ilə.

"Mütərcim" Nəşriyyat-Poliqrafiya Mərkəzi
Bakı, Rəsul Rza küç., 125
Tel./faks 596 21 44
e-mail: mutarjim@mail.ru