II. Əməliyyat sistemləri
2.1. Proqram təminatının təsnifatı. Baza (sistem) proqram təminatı

Yeni informasiya texnologiyalarının sürətli inkişafı və onun tətbiq olunma sahəsinin genişlənməsi proqram təminatının intensiv inkişaf etməsində mühüm rol oynayır. Onu geyd etmək kifayətdir ki, 1996-cı ildə dünya ictimaiyyəti proqram təminatına 110 milyard dollar sərf etmişdir. Proqram təminatının inkişaf tendensiyası onu göstərir ki, hər il ona qoyulan xərc 20% artır.

İnformasiya sistemlərinin proqram təminatı hesablama texnikası vasitələri ilə verilənləri emal edən sistemlərin yaradılması və istismarı üçün olan proqramlar və sənədlər çoxluğudur.

Proqram təminatının yerinə yetirdiyi funksiyalardan asılı olaraq onu 2 qrupa bölmək olar: baza (və ya sistem) proqram təminatı və tətbiqi proqram təminatı.

Baza (sistem) proqram təminatı informasiyanın kompüterdə emal olunmasını və tətbiqi proqramlar üçün normal iş şəratinin yaradılmasını təmin edir. Baza proqram təminatı aparat vasitələri ilə sıx əlaqədardır və çox zaman onu kompüterin bir hissəsi hesab edirlər.

Tətbiqi proqram təminatı istifadəçinin konkret məsələsinin həllini və bütövlükdə informasiya sistemində hesablama prosesinin təşkilini təmin edir.

Baza (sistem) proqram təminatının tərkibinə daxildir:

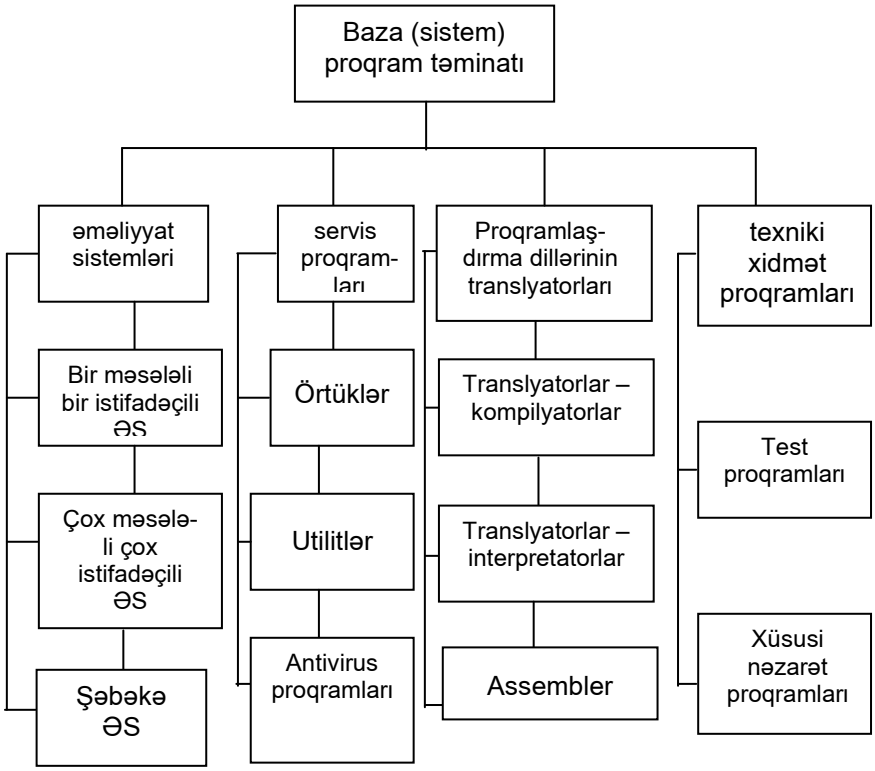
- əməliyyat sistemləri;
- servis proqramları;
- proqramlaşdırma dillərinin translyatorları;
- texniki xidmət proqramları.

Tətbiqi proqram təminatı baza (sistem) proqram təminatının xüsusilə əməliyyat sistemi proqramlarının idarəçiliyi altında işləyir. Tətbiqi proqram təminatının tərkibinə daxildir:

- müxtəlif təyinatlı tətbiqi proqramlar paketi;
- informasiya sistemlərinin və istifadəçinin işçi proqramları.

Baza (sistem) proqram təminatı

Baza (sistem) proqram təminatının tərkibi şəkil 2.1-də göstərilmişdir.



Şəkil 2.1. Baza (sistem) proqram təminatı

Baza proqram təminatına daxil olan əməliyyat sistemləri haqqında məlumat növbəti paraqraflarda veriləcəkdir.

Servis proqram təminatı istifadəçiyə kompüterdə işləmək üçün əlavə xidmətlər göstərən və əməliyyat sisteminin imkanlarını genişləndirən proqramlar çoxluğudur.

Xidmət vasitələrini funksional imkanlarına görə aşağıdakı qruplara bölmək olar:

- istifadəçi ilə olan interfeysi yaxşılaşdıran vasitələr;
- verilənləri korlanmadan və kənar müraciətlərdən qoruyan vasitələr;
- verilənləri bərpa edən vasitələr;
- disk və operativ yaddaş qurğusu arasında verilənlərin ötürülməsini sürətləndirən vasitələr;
- arxivləşdirən vasitələr;
- antivirus vasitələri.

Antivirus proqram vasitələri virusların tapılmasına və onların neytrallaşdırılmasına xidmət edir.

Proqramlaşdırma dillərinin translyatorları proqramlaşdırma dillərində yazılan proqram mətnini maşın koduna çevirir. İlkin proqramlaşdırma dilini, translyatoru, maşın dilini, standart proqramlar kitabxanasını, translyasiya olunmuş proqramın otlatka vasitələrini və onların vahid şəkildə əlaqələndirilməsini təmin edən proqram vasitələri proqramlaşdırma sistemi adlanır. İlkin proqramın maşın koduna çevrilməsi üsulundan asılı olaraq translyatorlar kompilyator və ya interpretator adlanır. Kompilyasiya prosesi zamanı ilkin proqram əvvəlcə obyekt koda çevrilir və sonra yerinə yetirilməyə hazır olan və diskdə fayl şəklində saxlanıla bilən vahid maşın proqramı şəklində hazırlanır. Bu proqram yenidən translyasiya olunmadan təkrar olaraq yerinə yetirilə bilər.

İnterpretator ilkin proqramın operatorlarını addım-addım translyasiya edir və diskdə saxlamadan yerinə yetirir. Ona görə də hər dəfə proqramı yerinə yetirmək lazım gəldikdə o, yenidən translyasiya olunur.

Proqramlaşdırma sistemində Assembler xüsusi rol oynayır və o, özündə Assembler dilini və onun Assembler - kompilyatorunu birləşdirir.

Texniki xidmət proqramları kompüterin və ya ümumiyyətlə hesablama sistemlərinin işi zamanı səhvlərin diaqnostikası və tapılması üçün olan aparat-proqram vasitələri çoxluğudur. Bura kompüterin və ya onun ayrı-ayrı hissələrinin düzgün işləməsini yoxlayan test proqramları, informasiya sistemlərinin işlək vəziyyətdə olmasını yoxlayan xüsusi nəzarət proqramları daxildir.

2.2. Tətbiqi proqram təminatı

Tətbiqi proqram təminatının tərkibi şəkil 2.2.-də göstərilmişdir.

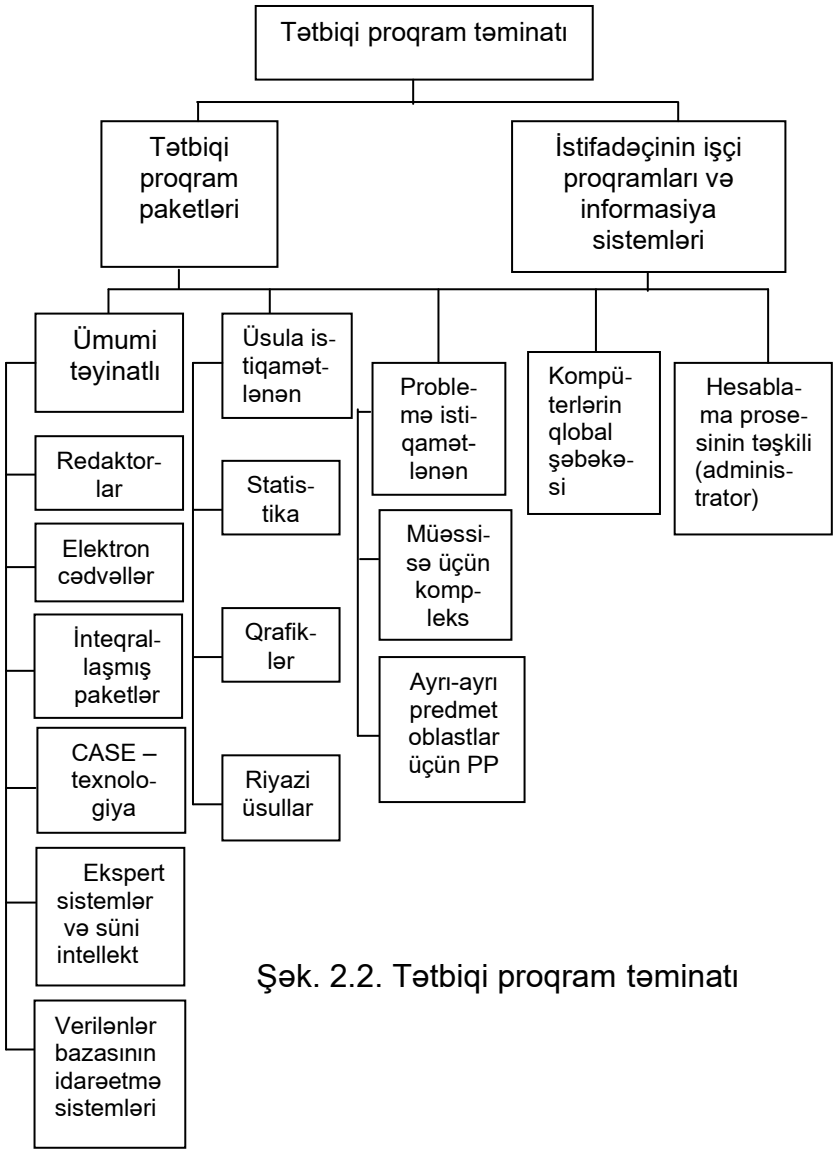
Tətbiqi proqramlar paketi (TPP) müəyyən sinifdən olan məsələlərin həlli üçün istifadə olunan kompleks proqramlardır.

TPP-nin aşağıdakı növləri vardır:

- ümumi təyinatlı;
- üsula istiqamətli;
- problemə istiqamətli;
- qlobal şəbəkə üçün;
- hesablama prosesinin təşkili üçün.

Ümumi təyinatlı TPP-yə daxildir.

- mətn və qrafik redaktorlar;
- elektron cədvəllər;
- verilənlər bazasının idarə edilməsi sistemləri (VBİS);
- inteqrallaşmış paketlər;
- CASE texnologiya;
- ekspert sistem və süni intellekt sistemlərinin örtüyü.



Şək. 2.2. Tətbiqi proqram təminatı

Mətn və qrafik redaktorlar mətnlərin, sənədlərin, qrafik verilənlərin və şəkillərin hazırlanması və ya dəyişdirilməsi üçün istifadə edilir. Mətn redaktorlarına Microsoft Word, Word Perfect, Multi Edit redaktorlarını misal göstərmək olar. Qrafik redaktorlara Paint, Corel Draw, Adobe Photoshop və s. misal göstərmək olar.

Elektron cədvəllər paketi əsasən cədvəllərlə işləmək üçündür. Bu sinfə daxil olan paketlərə Microsoft Excel, Lotus 1-2-3 və s. misal göstərmək olar.

Verilənlər bazasının yaradılması və işlənməsi üçün verilənlər bazasının idarə olunması sistemindən (VBİS) istifadə edilir. Müasir VBİS-lərə Microsoft Access, Microsoft FoxPro, Oracle, Paradox və s. misal göstərmək olar.

İnteqrallaşmış paketlər müxtəlif təyinatlı TPP-nin proqram komponentlərini özündə birləşdirir. Müasir inteqrallaşmış TPP özündə aşağıdakıları birləşdirə bilər:

- mətn redaktorlarını;
- elektron cədvəlləri;
- qrafik redaktorları;
- VBİS-ləri;
- Kommunikasiya modullarını.

Belə paketlərə misal olaraq Microsoft Office, Framework, Statnave-ni göstərmək olar.

Case – texnologiya mürəkkəb informasiya sistemlərinin hazırlanması zamanı tətbiq olunur. Bu informasiya sistemlərinin layihələşməsində müxtəlif mütəxəssislər, sistem analitikləri, layihəçilər, proqramçılar iştirak edirlər.

Ekspert sistemlər müəyyən oblasta aid olan biliklərin emal olunmasını və professional ekspert səviyyəsində həllin qəbul edilməsinin hazırlanmasını təmin edir.

Üsula istiqamətlənən TPP-nin alqoritmik əsasını hər hansı iqtisadi-riyazi üsul təşkil edir. Bura daxildir:

- riyazi proqramlaşdırma;
- şəbəkə planlaşdırılması və idarə edilməsi;
- kütləvi xidmət nəzəriyyəsi;
- riyazi statistika.

Problemə istiqamətlənən TPP-lər konkret funksional oblasta aid olan məsələlərin həlli üçün tətbiq edilir. Bu sinfə daxil olan TPP-lər sənaye sferalarına, bank sahələrinə, mühasibat sahələrinə, maliyyə sahələrinə və s. aid olan məsələlərin həlli üçün tətbiq edilir.

Qlobal şəbəkə üçün olan TPP-lər istifadəçiyə şəbəkənin resurslarına, verilənlər bazasına və s. əlverişli və etibarlı bir şəkildə müraciət etməyə imkan verir. Elektron poçtu, telekonferensiyanı təşkil etmək üçün və ötürülən informasiyanın məxfiliyini saxlamaq üçün bu sinfə daxil olan paketlərdən istifadə edilir. İnternet qlobal şəbəkə paketlərinə misal olaraq Netscape Novigator, Microsoft İnternet Explorer, Mail və s. göstərmək olar.

Hesablama proseslərinin təşkilinə aid olan TPP-lər verilənlərin administrasiyasını, komutatorları, marşrutizatorları, məlumatların trafiklərini idarə edir.

2.3. Əməliyyat sistemi haqqında anlayış. Onun funksiyaları

Əməliyyat sistemi kompüterin resurslarının idarə olunmasını, tətbiqi proqramların yerinə yetirilməsini və onların xarici qurğularla əlaqəsini, istifadəçinin kompüterlə dialoqunu təmin edən proqram vasitələri çoxluğudur.

Kompüterin resurslarını mərkəzi prosessor, operativ və xarici yaddaş, xarici qurğular, proqramlar və s. aiddir. Əməliyyat sistemi kompüter işə başlayan zaman yüklənir və onun əsas funksiyası istifadəçiyə hesablama sistemi ilə əlverişli interfeys üsulları hazırlamaqdır. Bu üsullar proqram və istifadəçi interfeysləri ola bilər.

Proqram interfeysi hesablama sistemi çərçivəsi daxilində qurğu və proqramların qarşılıqlı əlaqəsini təmin edən proqram vasitələri çoxluğudur.

İstifadəçi interfeysi – istifadəçinin proqram və ya kompüterlə əlaqəsini təmin edən proqram və aparat vasitələri çoxluğudur.

İstifadəçi interfeysi öz növbəsində əmrlə və ya obyekt yönümlü ola bilər. Əmrlə olan interfeysdə istifadəçi klaviaturadan əmrləri daxil etməklə kompüterin resurslarını idarə edir. Obyekt yönümlü interfeysdə hesablama sisteminin resursları obyektlər üzərində əməliyyatlar aparmaqla yerinə yetirilir. Obyekt olaraq fayllar, kataloqlar (qovluqlar), diskovodlar, proqramlar, sənədlər və s. götürülür.

Hər bir kompüter əməliyyat sistemləri ilə təmin olunur və bunların hər biri özünün tətbiqi proqramlar çoxluğunu yaradır. Əməliyyat sistemləri yerinə yetirdikləri funksiyalardan asılı olaraq aşağıdakı üç qrupa bölünürlər:

- bir məsələli (bir istifadəçili);
- çox məsələli (çox istifadəçili);
- şəbəkə əməliyyat sistemləri.

Bir məsələli əməliyyat sistemləri hər vaxt anında bir konkret məsələ ilə işləyən bir istifadəçi üçündür. Bu əməliyyat sistemlərinin tipik nümunəsi MS-DOS-dur.

Çox məsələli əməliyyat sistemləri multiproqram rejimində hesablama sistemindən kollektiv istifadəni təmin edir. Bu halda kompüterin yaddaşında bir neçə proqram – məsələ yerləşir və prosessor resursları bu məsələlər arasında paylayır. Bu sinfə daxil olan əməliyyat sistemlərinə UNIX, OS 2, Microsoft Windows 95, 98, 2000, 2002, XP, NT və s. misal göstərmək olar.

Şəbəkə əməliyyat sistemləri lokal və qlobal şəbəkələrin yaranması ilə əlaqədardır və istifadəçiyə hesablama şəbəkəsinin bütün resurslarından istifadə etməyə imkan verir. Şəbəkə əməliyyat sistemlərinin tipik nümayəndələrinə Novell Solaris (Sun firması) və s. misal göstərmək olar.

Adətən şəbəkə əməliyyat sistemi şəbəkəyə xidmət və resurslardan birgə istifadə üçün daha güclü kompüterə (serverə) qoyulur. Digər əməliyyat sistemləri şəbəkəyə qoşulan fərdi kompüterlərə qoyulur və lokal əməliyyat sistemləri hesab olunurlar. Şəbəkəyə qoşulan kompüterlər iş stansiyaları və ya klient adlanır.

Əməliyyat sistemlərinin əlavə funksiyalarına aşağıdakıları daxil etmək olar:

- xüsusi proqram təminatı olmadan lokal kompüter şəbəkəsinə fəaliyyət göstərmək imkanı vermək;
- İnternetin əsas xidmətlərinə müraciət etməyi təmin etmək;
- İnternet serverinin sistem vasitələrini yaratmaq, ona xidmət etmək və onu idarə etmək imkanına malik olmaq;
- verilənlərin qorunması, baxılması və dəyişdirilməsi vasitələrinə malik olmaq;
- əməliyyat sisteminin multimedia tipli iş mühitini yaratmaq:
- verilmiş cədvələ görə uzaqda yerləşən serverin idarəsi ilə kompüterə və əməliyyat sisteminə avtomatik xidmət əməliyyatlarını yerinə yetirmək.

Burada göstərilənlərdən əlavə müasir əməliyyat sistemləri minimal tətbiqi proqramlar çoxluğuna malikdir ki, bu tətbiqi proqramların köməyi ilə aşağıdakı əməliyyatları yerinə yetirmək olar:

- mətn sənədlərin oxunması, redaktə olunması və çapı;
- sadə şəkillərin hazırlanması və redaktə olunması;
- riyazi hesablamaların yerinə yetirilməsi;
- işçi bloknotunun və gündəliyin hazırlanması;
- elektron poçtun məlumatlarının hazırlanması, göndərilməsi və qəbulu;
- faks məlumatların hazırlanması və redaktə edilməsi;
- səs yazılışının və video yazılışın hazırlanması və redaktə edilməsi;
- mətn, qrafika, səs və video yazını özündə birləşdirən elektron sənədlərin hazırlanması.

Bununla əməliyyat sisteminin funksiyaları qurtarmır. Hesablama texnikasının aparat və proqram vasitələri inkişaf etdikcə əməliyyat sisteminin funksiyaları da kəsilməz olaraq genişlənir və təkmilləşir.

2.4. Əməliyyat sistemlərinə aid misallar

DOS ailəsinə daxil olan əməliyyat sistemləri. Bu ailənin birinci nümayəndəsi 1981-ci ildə hazırlanan MS-DOS (Microsoft Disk Operating System) əməliyyat sistemidir. MS-DOS əməliyyat sistemi bir məsələli əməliyyat sistemidir və aşağidakı xarakterik cəhətləri vardır:

- kompüterlə interfeys istifadəçi tərəfindən daxil edilən əmrlərin vasitəsi ilə olur;
- modul strukturuna malikdir və digər kompüterlərə köçürülməsi sadədir;
- mümkün olan operativ yaddaşın tutumu çox deyildir (640 k bayt).

DOS ailəsindən olan əməliyyat sisteminin çatışmayan cəhəti fərdi kompüterlərin resurslarına qadağan olunmuş müraciətlərdən qoruya bilməməsidir. MS DOS 6.22 əməliyyat sistemi hazırda daha geniş yayılmışdır.

OS/2 ailəsinə daxil olan əməliyyat sistemləri

OS/2 əməliyyat sistemi 1987-ci ildə IBM firması tərəfindən işlənib hazırlanmışdır. OS/2 əməliyyat sistemi ikinci nəslə aid olan çox məsələli 32 mərtəbəli və qrafik əməliyyat sistemidir. IBM PC tipli kompüterlərə uyğundur, bir neçə tətbiqi proqramların paralel işlənməsini təmin etmək və bu halda bir proqramın digər proqramdan və eləcə də əməliyyat sisteminin digər proqramlardan qorumaq imkanı vardır.

OS/2 əməliyyat sistemi əlverişli qrafik interfeys vasitələrinə malikdir və DOS sisteminin fayl sisteminə uyğundur. Bu isə verilənləri həm DOS, həm də OS/2 sistemində istifadə etmək imkanını verir.

OS/2 əməliyyat sisteminin aşağıdakı modifikasiyaları vardır:

- OS/2 Warp 3.0 – yaddaşdan və qrafik interfeysdən istifadə etmək imkanlarını yaxşılaşdırır;
- OS/2 Warp Connect – şəbəkə ilə işləməyi yaxşılaşdırır;
- OS/2 Warp Server – server əməliyyat sistemi kimi işləmək üçün təyin olunmuşdur .

OS/2 əməliyyat sisteminin əsas çatmamazlığı onun əmrlərinin az olmasıdır ki, bu isə onun MS DOS və Windows əməliyyat sisteminə nəzərən populyarlığını azaldır.

UNIX ailəsinin əməliyyat sistemləri

Unıx ailəsinə daxil olan əməliyyat sistemləri də 32 mərtəbəli çox məsələli, çox istifadəçili əməliyyat sistemləridir. Unıx əməliyyat sisteminin üstün cəhəti odur ki, bu əməliyyat sistemi müxtəlif kompüterlərdə – super kompüterlərdən fərdi kompüterlərə qədər – istifadə oluna bilər. Bu əməliyyat sistemi kiçik itgi ilə bir maşın arxitekturasından digərinə köçürülə bilər.

Unıx əməliyyat sistemində paylanan verilənlər bazasına, lokal şəbəkələrə müraciət etmək, uzaq məsafədən əlaqə və adi modemdən istifadə etməklə qlobal şəbəkəyə çıxış imkanları vardır. Unıx əməliyyat sistemində DOS-un və Windows-un əsas əmrləri istifadə olunur. Bu ailəyə daxil olan bir neçə əməliyyat sistemləri vardır ki, onlarda Unıx baza əməliyyat sistemnin xüsusiyyətləri təkrarlanır. Məsələn, şəbəkə üçün olan Unıx Ware 2.0 – 32 mərtəbəli çox məsələli çox istifadəçili əməliyyat sistemidir.

Windows ailəsinin əməliyyat sistemləri

Windows ailəsinə məxsus olan əməliyyat sistemləri Microsoft firması tərəfindən hazırlanmışdır. Bu əməliyyat sistemləri çox məsələli əməliyyat sistemləri olub istifadəçiyə əlverişli qrafik interfeysdən istifadə etmək imkanını verir. Bu ailəyə daxil olan əməliyyat sistemlərinə Windows 95, Windows 98, Windows 2000, Windows XP, Windows NT və s. misal göstərmək olar.

Windows NT daha geniş yayılan 32 mərtəbəli şəbəkə əməliyyat sistemidir. Windows NT-nin iki modifikasiyası vardır: Windows NT Server və Windows NT Work Station. Windows NT Server şəbəkə resurslarını idarə etmək üçün istifadə olunur. Bu əməliyyat sistemində informasiyanın sürətli axtarışı üçün, qlobal şəbəkənin resurslarına müraciət üçün vasitələr vardır.

Windows NT Work Station əməliyyat sistemi Windows NT-nin versiyası olub, lokal kompüter şəbəkələri ilə və iş stansiyaları ilə işləmək üçündür. 32 mərtəbəli çox məsələli və etibarlı əməliyyat sistemidir.

Real vaxtla olan əməliyyat sistemləri. Real vaxt termini informasiyanın işlənməsi sistemlərinə o vaxt tətbiq olunur ki, sistem təminatlı vaxt reaksiyasına malik olsun, yəni cavabın gecikməsi müəyyən olunmuş vaxtı aşmamalıdır.

Real vaxtla olan əməliyyat sistemləri müəyyən vaxt ərzində sistemin reaksiyasına təminat verir. Bu vaxt milli saniyələrlə ölçülür. Real vaxtla olan əməliyyat sistemləri əsasən neft və qazın çıxarılması və nəqlinin avtomatlaşdırılması, kimyəvi proseslərin, maşınqayırma və metallurgiya proseslərinin, bank, energetika və robototexnika məsələlərinin idarə olunmasına tətbiq olunur. Bu əməliyyat sistemlərinə RTMX, AMX, OS-9000, FLEX OS, QNX və s. misal göstərmək olar.

2.6. Proseslərin planlaşdırılması alqoritmi

Proseslərin planlaşdırılması aşağıdakı məsələlərin həllindən ibarətdir:

- yerinə yetirilən prosesin əvəz olunması üçün vaxt anının müəyyən edilməsi;
- hazır proseslərdən yerinə yetirilmək üçün olan prosesin seçilməsi;
- «köhnə» proseslər kontekstindən «yeni» proseslər kontekstinə keçilməsi.

Proseslərin planlaşdırılmasının bu məsələləri üçün müxtəlif alqoritmlər mövcuddur. Bu alqoritmlər çoxluğundan ən çox istifadə olunan aşağıdakı iki qrup alqoritmdir:

- kvantlaşmaya əsaslanan alqoritm;
- üstünlük dərəcəsinə (prioritetə) əsaslanan alqoritm.

Kvantlaşmaya əsaslanan alqoritmdə aktiv prosesin əvəz edilməsi aşağıdakı hallarda baş verir:

- proses yekunlaşır və sistemi tərk edir;
- səhv əmələ gəlir;
- proses GÖZLƏMƏ vəziyyətinə keçir;
- verilmiş proses üçün ayrılan prosessor vaxtının kvantı qurtarır.

Kvantı qurtaran proses HAZIRLIQ vəziyyətinə keçir və ona yeni prosessor vaxtının kvantı verilənə qədər gözləyir. Bu halda yerinə yetirilmək üçün müəyyən qayda ilə yeni proses seçilir. Bu halda heç bir proses prosessorun vaxtını çox almır. Proseslərə ayrılan kvantlar bütün proseslər üçün eyni və ya müxtəlif ola bilər. Hazır proseslərin növbəsi müxtəlif qaydada təşkil oluna bilər. Məsələ, LIFO və ya FIFO prinsipi ilə.

Alqoritmlərin digər qrupu proseslərin üstünlük dərəcəsindən istifadə edir. Üstünlük dərəcəsi – proseslərin hesablama maşınlarının resurslarından, xüsusi halda prosessor vaxtından istifadə edən zaman onun üstünlüyünü xarakterizə edən ədəddir. Üstünlük dərəcəsi çox olan proses növbədə də az gözləyir. Növbədə üstünlük dərəcəsi çox olan proses yerinə yetirilmək üçün seçilir.

Bir çox əməliyyat sistemlərində planlaşdırma alqoritmi həm kvantlaşma, həm də üstünlük dərəcəsinə görə təyin edilir. Məsələ, planlaşdırmanın əsasında kvantlaşma və ya kvant ədədi durur, hazır proseslərin növbədən seçilməsi isə proseslərin üstünlük dərəcəsinə əsaslanır.

Proseslərin planlaşdırılmasının iki tip proseduru vardır:

- kənarlaşdırıla bilməyən çox məsələlik (Non – preemptive multitasking);
- kənarlaşdırıla bilən çox məsələlik (Preemptive multitasking).

Birinci üsulda aktiv prosesin yerinə yetirilməsi onun özü idarəni əməliyyatlar sisteminin planlaşdırıcısına verənə qədər davam edir. Bu zaman əməliyyat sisteminin planlaşdırıcısı növbədən yerinə yetirilməyə hazır olan digər bir prosesi seçir.

İkinci üsulda yerinə yetirilən prosesin digər proseslə əvəz edilməsi əməliyyat sisteminin planlaşdırıcısı tərəfindən yerinə yetirilir.

Preemptive və non-Preemptive variantları bir-birindən məsələlərin planlaşdırılması mexanizminin mərkəzləşdirilməsi dərəcəsinə görə fərqlənirlər. Kənarlaşdırıla bilən çoxməsələlikdə məsələlərin planlaşdırılması mexanizmini tamamilə əməliyyat sistemi müəyyən edir. Bu halda proqramçı öz proqramını yazır və onun digər proqramlarla paralel olaraq necə yerinə yetirilməsi haqqında düşünmür. Bu zaman əməliyyat sistemi aşağıdakı funksiyaları yerinə yetirir: aktiv məsələlərin dayandırılması anını müəyyən edir, onun kontekstini yadda saxlayır, hazır məsələləri növbədən götürür və onun kontekstini yükləyərək yerinə yetirir.

Kənarlaşdırıla bilməyən çox məsələlikdə planlaşdırma mexanizmi sistem və tətbiqi proqramlar arasında paylanır. Tətbiqi proqram əməliyyat sistemindən idarəni aldıqdan sonra özü növbəti iterasiyanın yerinə yetirilməsini müəyyən edir və hər hansı sistem müraciəti zamanı idarəni əməliyyat sisteminə verir. Əməliyyat sistemi növbəni formalaşdırır və müəyyən bir alqoritmlə (məsələ, üstünlük dərəcəsinə görə) növbəti məsələni yerinə yetirmək üçün seçir.

Kənarlaşdırıla bilməyən çoxməsələlik NetWare fayl serverində effektiv olaraq istifadə olunur və bunun nəticəsində fayl əməliyyatları yüksək sürətlə yerinə yetirilir.

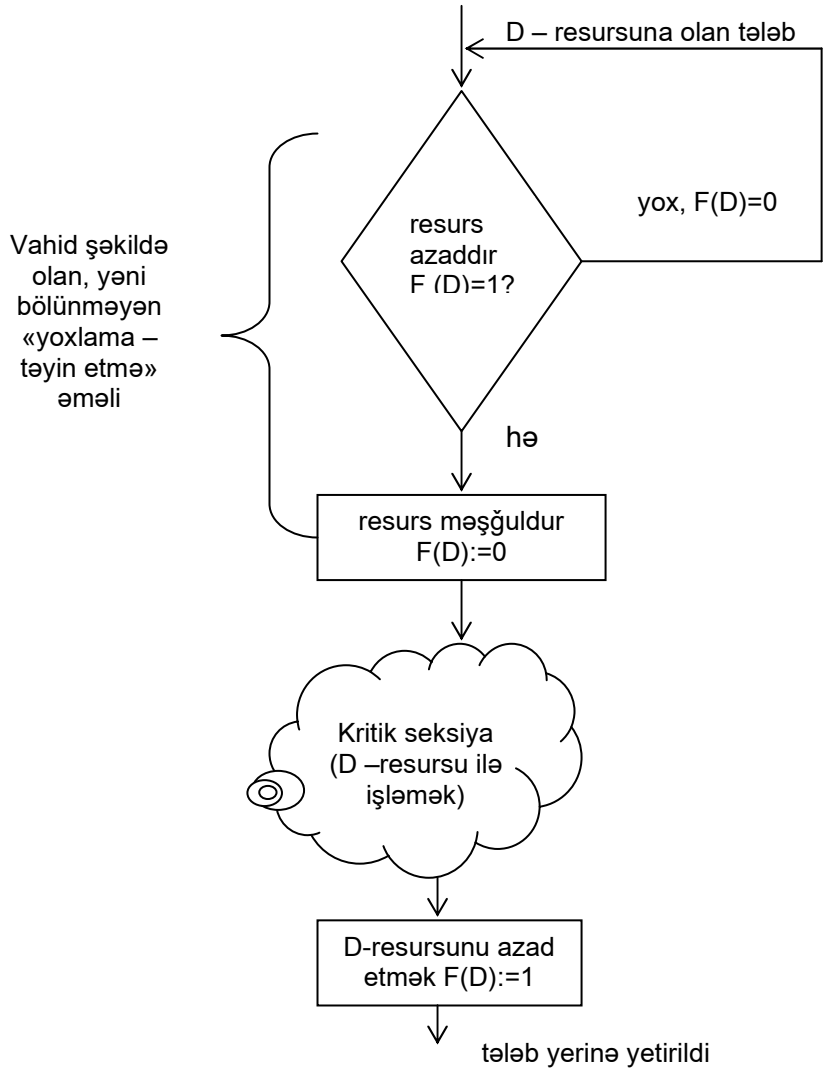
Proqramların yüksək məhsuldarlıqla yerinə yetirilməsinə istiqamətlənən müasir əməliyyat sistemlərinin çoxu (Unix, Windows NT, OS/2, VAX/V MS) kənarlaşdırıla bilən çox məsələlik rejimində realizasiya olunurlar.

Proseslərin çox zaman bir-biri ilə qarşılıqlı əlaqəsi lazım gəlir. Belə ki, bir proses verilənləri digər prosesə verə bilər və ya bir neçə proses ümumi faylda yerləşən verilənlərdən istifadə edə bilər. Bütün bu hallarda proseslərin sinxronlaşması problemi əmələ gəlir.

Sinxronlaşdırmanın köməyi ilə prosesləri aktivləşdirmək və ya dayandırmaq, növbələri təşkil etmək, resursları bloklaşdırmaq və ya azad etmək olar.

2.7. Kritik seksiya

Proseslərin sinxronlaşdırılmasının əsas anlayışı proqramın «kritik seksiya» anlayışıdır. Kritik seksiya – proqramın bir hissəsidir ki, bu hissədən müəyyən verilənlərə müraciət etmək mümkündür. Hər hansı bir resursa müraciət çox olarsa, onda bu resursla əlaqədar olan kritik seksiyada ancaq bir prosesin olması təmin olunur. Bu üsul qarşılıqlı yoxetmə üsulu adlanır.



Şəkil 2.4. Bloklaşdıran dəyişənlərdən istifadə etməklə kritik seksiyaların reallaşdırılması

Qarşılıqlı yoxetməni təmin edən ən sadə üsullardan biri kritik seksiyada yerləşən proqramda kəsilmə halının qadağan olunmasıdır. Bu üsul o qədər də əlverişli deyildir. Çünki sistemin idarə olunmasını istifadəçinin prosesinə etibar etmək qorxuludur. Kritik

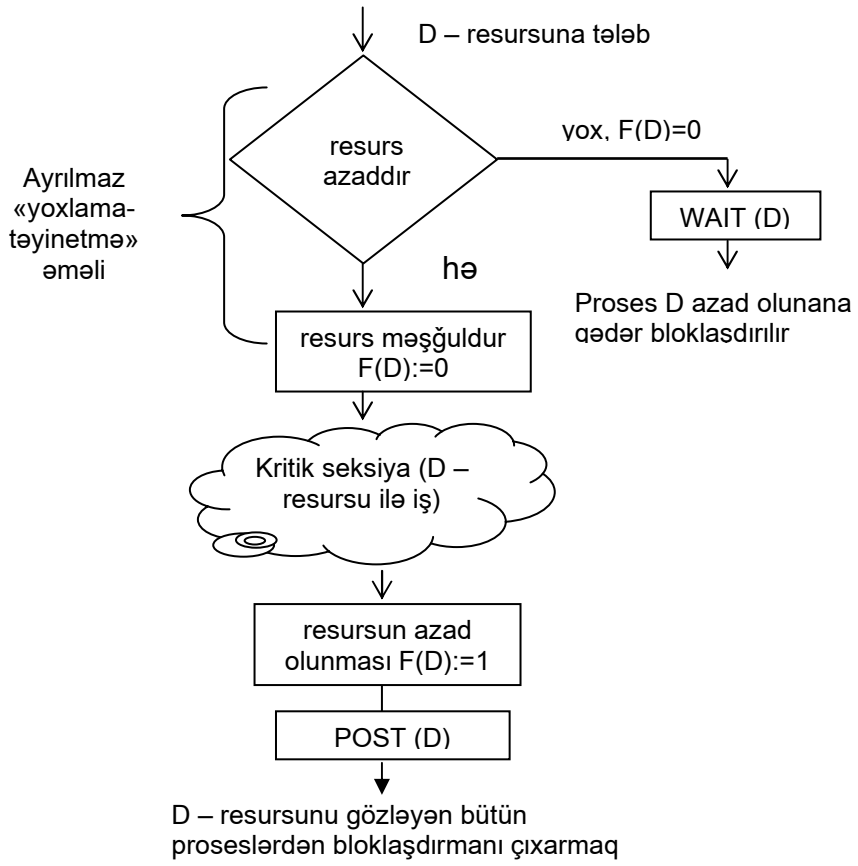
oblastda yerləşən prosesdəki hər hansı bir nasazlıq kəsilmə halına icazə verilmədiyi üçün bütün sistemə təsir edə bilər.

Qarşılıqlı yoxetməni təmin edən üsullardan biri də bloklaşdırılan dəyişənlərdən istifadə etməkdən ibarətdir. Hər bir paylanan resursla bir ikilik dəyişən əlaqəli olur. Əgər resurs azaddırsa (yəni bu proseslə əlaqədar olan heç bir proses verilmiş anda kritik seksiyada yerləşmirsə) bu dəyişən 1, resurs məşğuldursa $\varnothing$ qiymətini alır. Şəkil 2.4-də bloklaşdırılan F(D) dəyişən ilə olan D paylanan resursuna qarşılıqlı yoxetmə ilə müraciəti reallaşdıran alqoritmin bir fraqmenti göstərilmişdir. Kritik seksiyaya girişdən əvvəl proses D resursunun azad və ya boş olmasını yoxlayır. Əgər resurs məşğuldursa yoxlama dövrü olaraq təkrarlanır. Əgər resurs azaddırsa F(D) dəyişəni 0 – qiyməti alır və proses kritik seksiyaya daxil olur. Proses D – paylanan resursu ilə bütün əməliyyatları yerinə yetirdikdən sonra F(D) – dəyişəni yenidən 1 – qiyməti alır.

Qeyd etmək lazımdır ki, bloklaşdırılan dəyişənin yoxlanması və təyin edilməsi əməli ayrılmaz olmalıdır. Bunu izah edək. Tutaq ki, yoxlamanın nəticəsində proses resursun azad olmasını müəyyən etdi və dəyişənə 0 – qiymətini verməyə imkan tapana qədər proses kəsildi. Bu prosesin dayanması vaxtı digər proses resursu məşğul edə bilər və öz kritik seksiyasına daxil ola bilər. Bu proses də resursla olan işini qurtarmamış kəsilə bilər. Bu halda resursun azad olması hesab edilərək idarə birinci prosesə verilərək öz kritik seksiyasında yerinə yetirilməyə başlayır. Beləliklə, qarşılıqlı yoxetmə prinsipi pozulur və bu halda arzuolunmaz nəticələr alına bilər. Belə halların qarşısını almaq üçün maşının əmrlər sistemində vahid «yoxlama – təyin etmə» əmrinin olması vacibdir. Bloklaşdırılan dəyişənlərdən istifadə etməklə kritik seksiyaların reallaşdırılmasının çatışmayan cəhətləri vardır. Belə ki, bir proses kritik seksiyada yerləşən vaxt ərzində bu resursu tələb edən digər prosesə bloklaşdırılan dəyişənin sorğusu üçün lazım olan mürəkkəb əməliyyatların yerinə yetirilməsinə əhəmiyyətsiz prosessor vaxtı sərf olunur. Belə halın qarşısını almaq üçün hadisələr aparatından istifadə edilir. Bu vasitələrin köməyi ilə nəinki qarşılıqlı yoxetmə problemi həll edilir, həm də sinxronlaşma proseslərinin ümumi məsələləri həll edilir. Müxtəlif əməliyyat sistemlərində hadisələr aparatı özünəməxsus reallaşdırılır. Bunun üçün oxşar sistem funksiyalarından istifadə edilir. Bu sistem funksiyalarını şərti olaraq $\mathrm{WAIT}\ (\mathrm{x})$ və $\mathrm{Post}\ (\mathrm{x})$ – işarə edək. Burada x – müəyyən bir hadisənin identifikatorudur. Şəkil 2.5-də bu funksiyalardan istifadə edən alqoritmdən bir fraqment göstərilmişdir.

Əgər resurs məşğuldursa, onda proses dövrü olaraq sorğunu yerinə yetirmir və WAIT (D) sistem funksiyasına müraciət edir. Burada D – resursun azad olması hadisəsidir. WAIT (D) funksiyası aktiv prosesi GÖZLƏMƏ vəziyyətinə keçirir və onun deskriptorunda prosesin D – hadisəsini gözləməsi qeyd olunur. Bu vaxt ərzində D – resursundan istifadə edən proses kritik seksiyadan çıxdıqda Post (D) sistem funksiyası yerinə yetirilir və nəticədə əməliyyat sistemi gözləyən proseslərin növbəsindən D – hadisəsini gözləyən prosesi HAZIRLIQ vəziyyətinə keçirir.

Deykstr proseslərin sinxronlaşdırılması üçün ümumi bir vasitə olan semafor mexanizmini təklif etdi. Semaforlar – qiymətləri müsbət tam ədəd olan dəyişənlərdir və onlar üzərində ancaq iki əməliyyat – P və V əməliyyatları aparmaq olar. S – semaforu üçün V əməliyyatı S:= S+1 ilə eynigüclüdür. Yəni S – dəyişəni bir bölünməz əməliyyat qədər artır. Bu əməliyyat seçmə, yadda saxlama, vahid qədər artırma və ya kəsilmə ola bilər. Bu əməliyyat yerinə yetirildikdə S-ə digər proseslər tərəfindən müraciət oluna bilməz. P – əməliyyatı ℓ *: if* $s > 0$ *then* $S := S - 1$ *else go to* ℓ əməliyyatı ilə eynigüclüdür. Bu halda P və V əməliyyatları ayrılmazdır, yəni istənilən anda birdən çox belə əməliyyat yerinə yetirilə bilməz.



Şəkil 2.5. WAIT (D) və Post (D) sistem funksiyalarından istifadə etməklə kritik seksiyaların realizasiyası

Xüsusi halda S semaforu 0 və 1 qiymətləri alarsa, o bloklaşdırılan dəyişənə çevrilir. P – əməliyyatı yerinə yetirilən prosesi gözləmə vəziyyətinə keçirir, bu zaman V əməliyyatı P əməliyyatı vasitəsi ilə dayandırılan digər prosesi aktivləşdirir. Semaforlar əməliyyat sistemlərində proseslərin idarə olunmasında geniş tətbiq olunur.

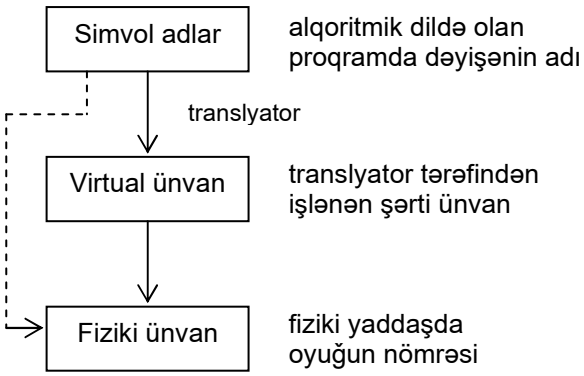
2.8. Yaddaşın idarə olunması

2.8.1. Ünvanların tipi

Yaddaş əməliyyat sistemi tərəfindən dəqiq idarə olunması tələb olunan çox vacib resursdur. Yaddaşın idarə olunmasına görə əməliyyat sisteminin funksiyaları aşağıdakılardır: azad və azad olmayan yaddaş sahələrini izləmək, proseslərə yaddaş sahəsini ayırmaq və ya proses yekunlaşdıqdan sonra yaddaş sahəsini azad etmək, operativ yaddaş kifayət etmədikdə operativ yaddaşdan diskə proseslərin köçürülməsi, operativ yaddaşda boş yer olduqda onların operativ yaddaşa qaytarılması və həmçinin fiziki yaddaşın konkret oblastında ünvanların sazlanması.

Dəyişənlərin və əmrlərin identifikasiyası üçün simvol adlardan (nişanlar), virtual və fiziki ünvanlardan istifadə edilir. Ünvanların tipi şəkil 2.6.-da göstərilmişdir.

Fiziki ünvan dəyişən və əmrlərin operativ yaddaşda yerləşdiyi oyuğun nömrəsidir. Virtual ünvandan fiziki ünvana keçid iki üsulla ola bilər. Birinci üsulda virtual ünvanın fiziki ünvana çevrilməsi xüsusi sistem proqramının – yerini dəyişən yükləyicinin köməyi ilə yerinə yetirilir. Yükləyicidə fiziki yaddaşın başlanğıc ünvanı və ünvandan asılı olan sabitlər haqqında informasiya olur. Yükləyici virtual ünvanı fiziki ünvana çevirərək proqramı fiziki yaddaşa yükləyir.



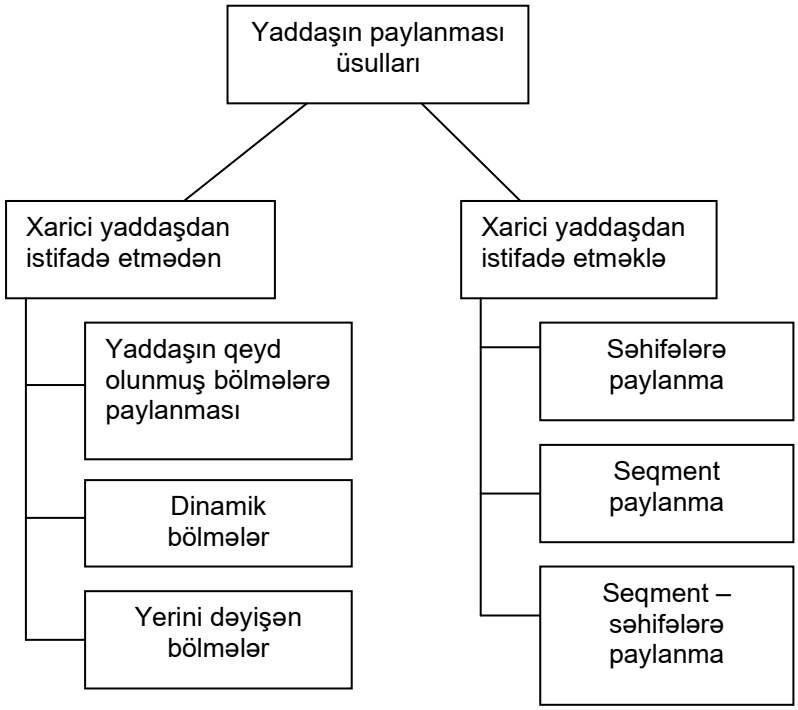
Şəkil 2.6. Ünvanların tipi

İkinci üsulda proqram yaddaşa dəyişməz olaraq virtual ünvanlarla yüklənir. Əməliyyat sistemi virtual ünvanlar fəzasına nəzərən proqramın kodunun yerləşdiyi yeri qeyd edir. Proqramın yerinə yetirilməsi zamanı operativ yaddaşa hər bir müraciətdə virtual ünvan fiziki ünvana çevrilir. İkinci üsul daha çevikdir. Bu üsulda yerinə yetirilmə zamanı proqramın yerini dəyişməsinə imkan verir. Ancaq birinci üsulda hər bir proqram üçün ilkin olaraq konkret yaddaş sahəsi ayrılır. Qeyd edək ki, birinci üsulda hər bir virtual ünvanın fiziki ünvana çevrilməsi ancaq bir dəfə – yükləmə zamanı baş verir, ikinci üsulda isə yaddaşa hər müraciət zamanı virtual ünvan fiziki ünvana çevrilir. Ona görə də yerini dəyişən yükləyici çəkilən xərcləri azaldır.

Bəzi hallarda əvvəlcə proqramın operativ yaddaşın hansı hissəsində yerinə yetirilməsi məlum olarsa, onda translyator yerinə yetirilən kodu birbaş fiziki ünvanda verir.

2.8.2. Disk fəzasından istifadə etmədən yaddaşın paylanması üsulları

Yaddaşı idarəedən üsullar iki sinfə bölünür: operativ yaddaşla disk arasında proseslərin yerdəyişməsi ilə olan üsullar və belə yerdəyişmədən istifadə etməyən üsullar. Yaddaşın paylanması üsullarının təsnifatı şəkil 2.7-də göstərilmişdir.



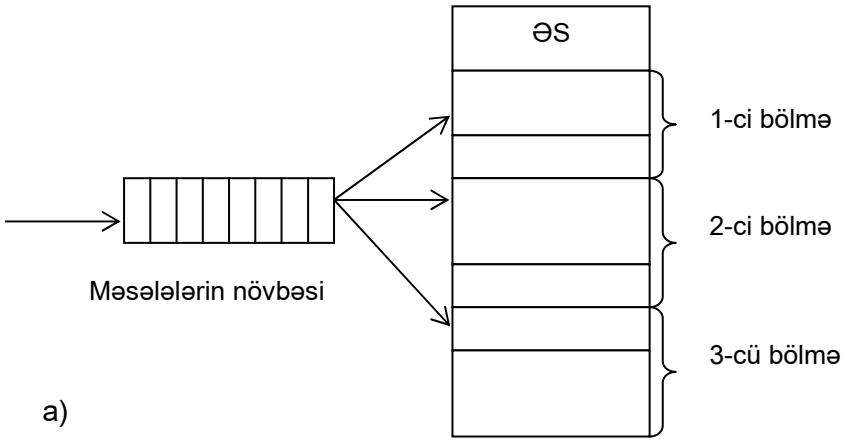
Şəkil 2.7. Yaddaşın paylanması üsullarının təsnifatı

2.8.3. Yaddaşın qeyd olunmuş bölmələrlə paylanması

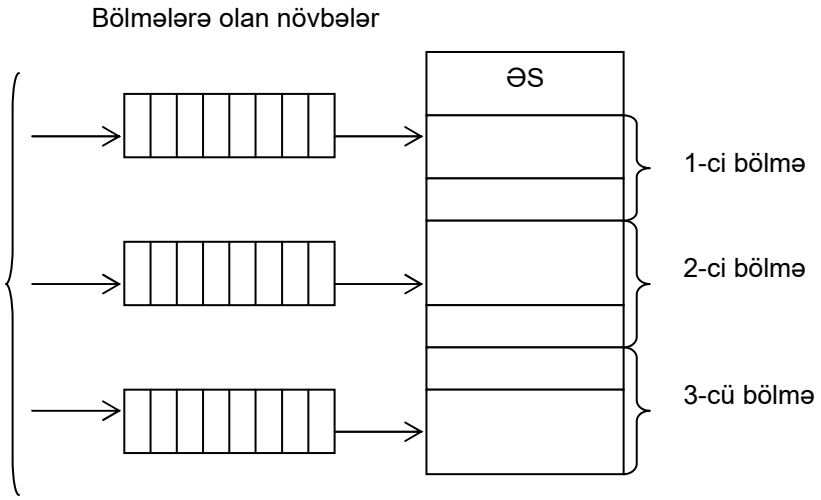
Operativ yaddaşın idarə olunmasının ən sadə üsulu onun bir neçə qeyd olunmuş bölmələrə bölünməsidir. Bu sistemin generasiyası zamanı və ya işə başlayanda operator tərəfindən əl ilə yerinə yetirilə bilər. Yerinə yetirilməyə qəbul olunan növbəti məsələ ya ümumi növbədə və ya müəyyən bir bölmədə növbədə yerləşir (şəkil 2.8. a) və şəkil 2.8. b)).

Bu halda yaddaşı idarəedən alt sistem aşağıdakı vəzifələri yerinə yetirir:
- yerinə yetirilməyə qəbul olunan proqramın ölçüsü boş olan bölmələrin ölçüləri ilə müqayisə edilir və uyğun bölmə seçilir;
- proqramın yüklənməsi və ünvanların sazlanması yerinə yetirilir.

Bu üsul realizasiyasına görə çox sadədir, lakin yaddaşdan istifadə olunması nöqteyi-nəzərindən əlverişli deyildir. Hər bir bölmədə ancaq bir proqram yerinə yetirildiyinə görə və bölmələrin sayı məhdud olduğuna görə hətta kiçik həcmə malik olan proqram bir tam bölməni məşğul edir. Bu isə yaddaşdan səmərəli istifadə olunmamasına səbəb olur. Digər tərəfdən operativ yaddaşın həcmi hər hansı proqramın yerinə yetirilməsinə imkan versə də yaddaşın bölmələrə bölünməsi buna mane olur.



a)



b)

Şəkil 2.8. Yaddaşın qeyd olunmuş bölmələrə paylanması
a) ümumi növbə ilə
b) ayrı-ayrı növbələr ilə

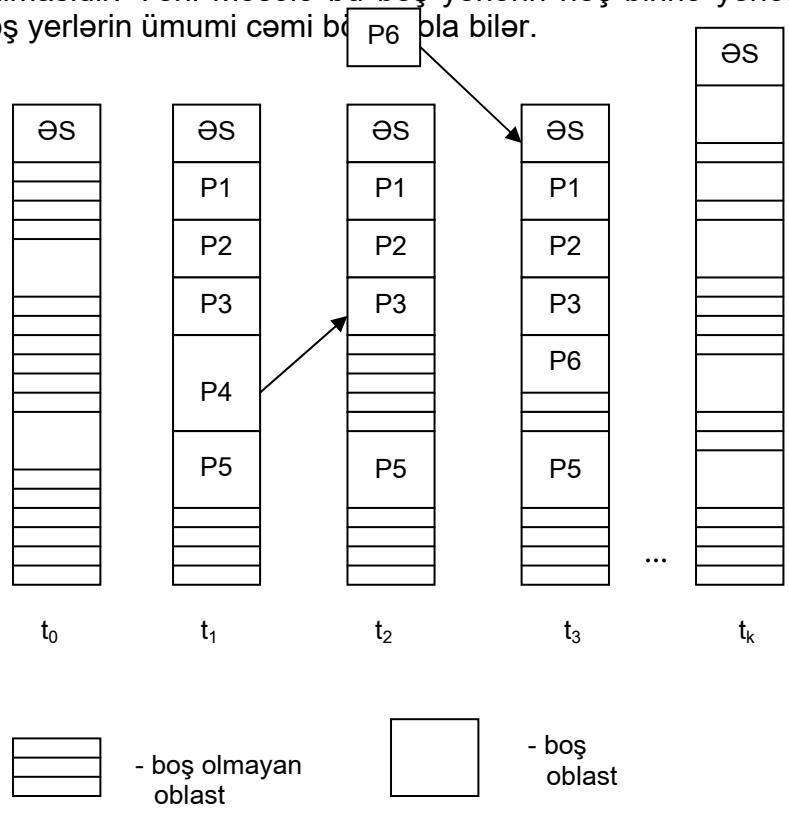
2.8.4. Yaddaşın dinamik bölmələrə paylanması

Bu halda maşının yaddaşı əvvəlcədən bölmələrə bölünmür. İlkin halda bütün yaddaş boş olur. Hər yeni qəbul olunan məsələ üçün lazım olan qədər yaddaş sahəsi ayrılır. Əgər ayrılan yaddaş sahəsi məsələ üçün kifayət deyilsə, onda məsələ yerinə yetirilmək üçün qəbul olunmur və növbədə durur. Yaddaş azad olduqda bu yerə digər proqram yüklənə bilər. Beləliklə, istənilən vaxt anında operativ yaddaş təsadüfi olaraq boş olan və ya boş olmayan bölmələr ardıcıllığından ibarətdir. Şəkil 2.9.-da dinamik paylanmada istənilən vaxt anında yaddaşın vəziyyəti göstərilmişdir. t0 – anında yaddaşda ancaq əməliyyat sistemi (ƏS) yerləşir, t1 – anında yaddaş 5 məsələ arasında bölünür. Bu halda P4 məsələsi yekunlaşaraq yaddaşı tərk edir. P4-dən sonra azad olunan yaddaşa t3 anında P6 məsələsi yüklənir.

Bu üsulun realizasiyası zamanı əməliyyat sisteminin funksiyaları aşağıdakılardır:

- başlanğıc ünvanı və ölçüsü göstərilən boş və boş olmayan oblastların cədvəlini vermək;
- yeni məsələ daxil olduqda onu analiz etmək, cədvəldən boş olan oblastları müəyyən etmək, yeni məsələnin yerləşdirilməsinə kifayət edən bölməni müəyyən etmək;
- yeni məsələni müəyyən olunan bölməyə yükləmək, boş və boş olmayan oblastlar cədvəlini korrektə etmək;
- məsələ yekunlaşdıqdan sonra boş və boş olmayan oblastlar cədvəlini korrektə etmək.

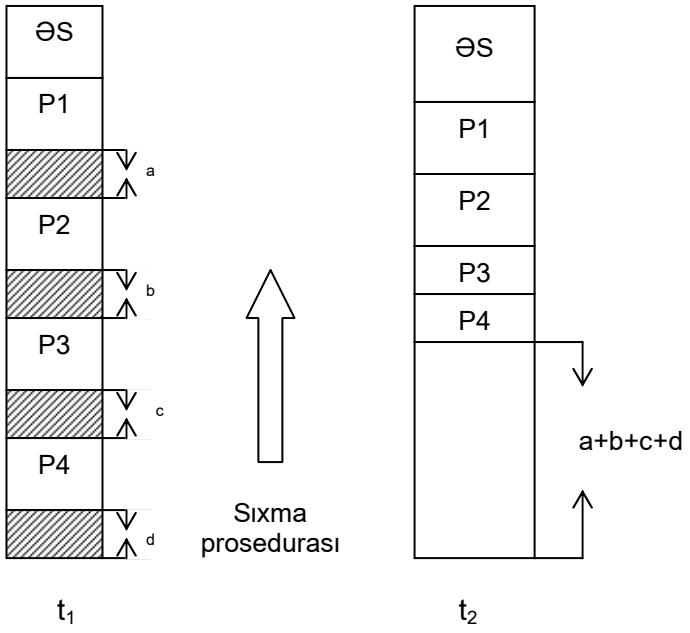
Bu üsulun çatışmayan cəhəti yaddaşda çox kiçik ölçülü boş yerlərin (fraqmentlərin) qalmasıdır. Yeni məsələ bu boş yerlərin heç birinə yerləşməyə bilər. Ayrı-ayrılıqda isə boş yerlərin ümumi cəmi bö[illegible]ola bilər.



Şəkil. 2.9. Yaddaşın dinamik bölmələrə paylanması

2.8.5. Yaddaşın yerini dəyişən bölmələrlə paylanması

Fraqmentləşmə ilə mübarizə üsullarından biri boş olmayan sahələrin yerini dəyişərək ya böyük ünvanlarda, və ya kiçik ünvanlarda yerləşdirilməsidir. Bu halda yaddaşın boş qalan sahələri vahid oblast əmələ gətirirlər (şəkil 2.10-a bax).



Şəkil 2.10. Yaddaşın yerini dəyişən bölmələrlə paylanması

Əməliyyat sisteminin dinamik bölmələrlə yaddaşın paylanması zamanı olan funksiyalarına əlavə olaraq, burada əməliyyat sistemi həm də hər bir anda yaddaşın boş və boş olmayan oblastlar cədvəlini korrektə etməklə bölmələrin məzmununu yaddaşın bir hissəsindən digər hissəsinə köçürməlidir. Bu proses «sıxma» adlanır. Sıxma prosesi ya məsələlər yekunlaşdıqdan sonra və ya yeni daxil olmuş məsələ üçün boş yer çatmadıqda yerinə yetirilir. Bu üsulda proqram yerinə yetirildiyi zaman operativ yaddaş da öz yerini dəyişdiyi üçün virtual ünvanların fiziki ünvanlara çevrilməsi dinamik üsulla yerinə yetirilir.

Sıxma proseduru yaddaşdan çox səmərəli istifadə etməyə imkan verir. Lakin bu prosedurun yerinə yetirilməsinə xeyli əlavə vaxt tələb edilir.

2.9. Xarici yaddaşdan istifadə etməklə yaddaşın paylanması üsulları

Proqramın həcmi boş olan yaddaş sahəsinin həcmindən çox olarsa belə proqramın yaddaşda yerləşdirilməsi problemi ilə istifadəçilər həmişə qarşılaşmışlar. Bu problemlərin həlli xarici yaddaşdan istifadə etməklə yaddaşın paylanması üsullarının köməyi ilə olur. Xarici yaddaşdan istifadə etməklə hesablama proseslərinin təşkili üsullarının inkişafı virtual yaddaş üsulunun meydana gəlməsinə səbəb oldu. Virtual yaddaş haqqında 1.6. paraqrafında ətraflı məlumat verilmişdir.

Virtual yaddaşın ən geniş yayılan növləri yaddaşın səhifələrlə, seqmentlərlə və səhifə – seqmentlərlə təşkilidir.

2.9.1. Yaddaşın səhifələrə paylanması

Hər bir prosesin virtual ünvan fəzası qeyd olunmuş eyni ölçülü hissələrə bölünür. Bu hissələrə virtual səhifələr deyilir. Virtual ünvanlar fəzasının ölçüsü səhifənin ölçüsünə bölünməyə bilər. Bu halda hər prosesin axırıncı səhifəsi fiktiv oblastla tamamlanır. Bütün operativ yaddaş da eyni ölçülü hissələrə bölünür. Bü hissələrə fiziki səhifələr deyilir.

Adətən səhifələrin ölçüləri 2-nin qüvvətləri şəklində götürülür. 512, 1024 və s. Bu ünvanların çevrilməsi mexanizmini sadələşdirir.

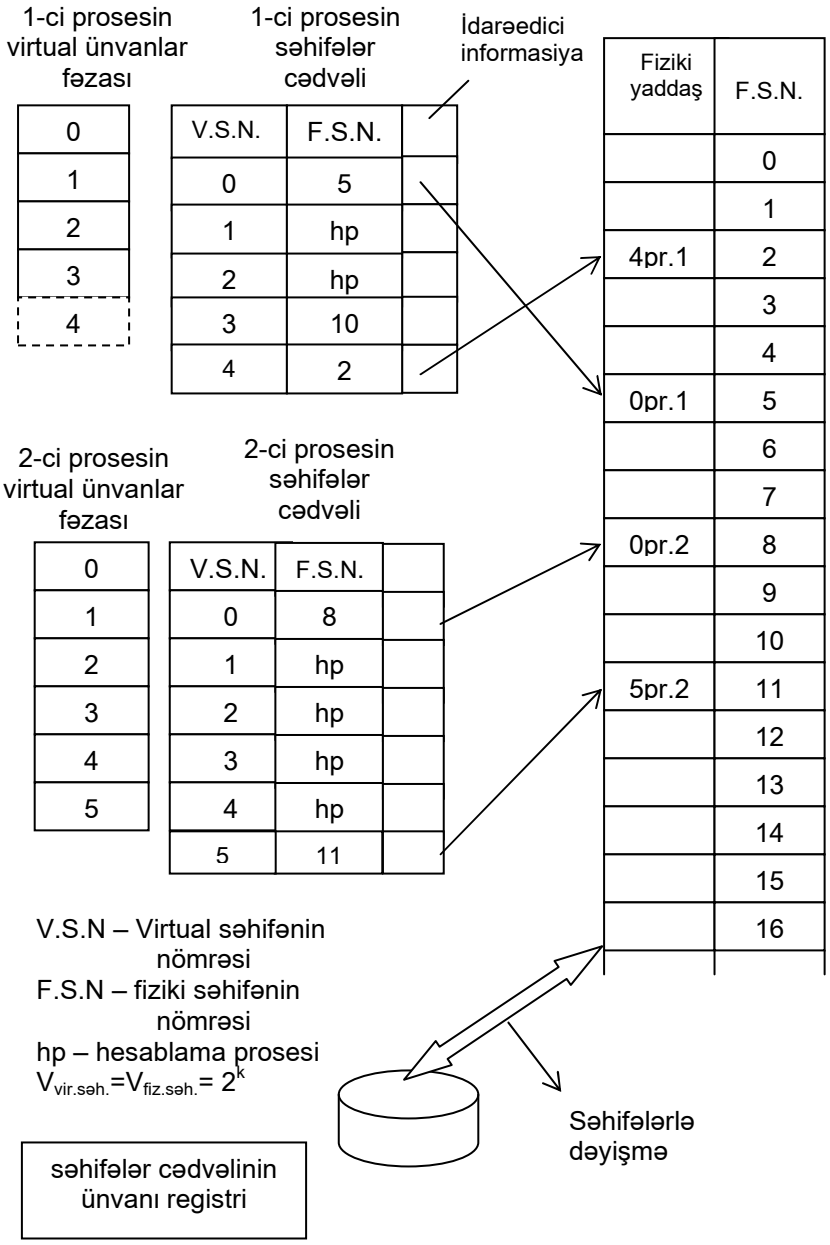
Prosesin yüklənməsi zamanı onun bir hissəsi operativ yaddaşda, bir hissəsi isə diskdə yerləşə bilər. Prosesin yüklənməsi zamanı əməliyyat sistemi hər proses üçün səhifənin cədvəlini tərtib edir. Bu cədvəldə operativ yaddaşa yüklənən fiziki səhifələrin və virtual səhifələrinin nömrələri haqqında məlumat və həmçinin idarəedici informasiyalar yerləşir. Bu idarəedici informasiyalar səhifələrin modifikasiyası əlaməti, səhifələrə müraciət əlaməti (müəyyən vaxt ərzində səhifələrə olan müraciətlərin sayı), səhifənin yüklənə bilməməsi əlaməti və s. ola bilər. Bu informasiyalardan virtual yaddaşın formalaşdırılması və istifadə olunması üçün istifadə edilir. Şəkil 2.11-də yaddaşın səhifələrlə paylanması sxemi göstərilmişdir.

Növbəti proses aktivləşən zaman prosessorun xüsusi registrinə bu prosesin səhifələr cədvəlinin ünvanı yüklənir.

Yaddaşa hər müraciət zamanı müraciət olunan virtual səhifə haqqında informasiya səhifələr cədvəlindən oxunur. Əgər bu virtual səhifə operativ yaddaşda yerləşərsə, onda virtual ünvanın fiziki ünvana çevrilməsi prosesi yerinə yetirilir. Əgər lazım olan virtual səhifə verilmiş anda diskdə yerləşərsə, onda səhifə kəsilməsi adlanan kəsilmə baş verir. Yerinə yetirilən proses gözləmə vəziyyətinə keçir və növbədə hazır olan digər proses aktivləşir. Paralel olaraq səhifə kəsilməsini işləyən proqram diskdə tələb olunan virtual səhifəni tapır və onu operativ yaddaşa yükləməyə cəhd edir. Əgər yaddaşda boş fiziki səhifə varsa, onda virtual səhifənin yüklənməsi yerinə yetirilir. Əgər operativ yaddaşda boş səhifə yoxdursa, onda operativ yaddaşdan hansı səhifəni çıxarmaq məsələsi həll edilir. Bu səhifə aşağıdakı kriteriyalardan biri ilə seçilə bilər:

- çoxdan istifadə olunmayan səhifə;
- birinci yerdə duran səhifə;
- son vaxtlar ən az müraciət olunan səhifə.

Səhifə seçildikdən sonra operativ yaddaşı tərk edir və onun modifikasiya olunma əlaməti (səhifələr cədvəlindən) analiz olunur. Əgər bu səhifə yükləmə anında modifikasiya olunarsa, o diskə yazılır, əks halda ləğv edilir və uyğun fiziki səhifə boş hesab edilir.



Şəkil 2.11. Yaddaşın səhifələrlə paylanması

Yaddaşın səhifələrlə təşkilində virtual ünvanın fiziki ünvana çevrilməsi mexanizminə baxaq (şəkil 2.12.).

Səhifələrlə paylanma zamanı virtual ünvan (P,S) – cütü şəklində göstərilir. Burada P – prosesin virtual səhifəsinin nömrəsi (nömrələmə 0-dan başlayır), S – isə virtual səhifə daxilində yerdəyişmədir. Səhifənin ölçüsünün 2^k olmasını nəzərə alsaq yerdəyişmənin qiyməti virtual ünvanının 2-lik yazılışının k-kiçik mərtəbəsinə uyğundur. Yerdə qalan böyük mərtəbələrdə p - səhifəsinin nömrəsinin 2-lik yazılışı təsvir olunur.

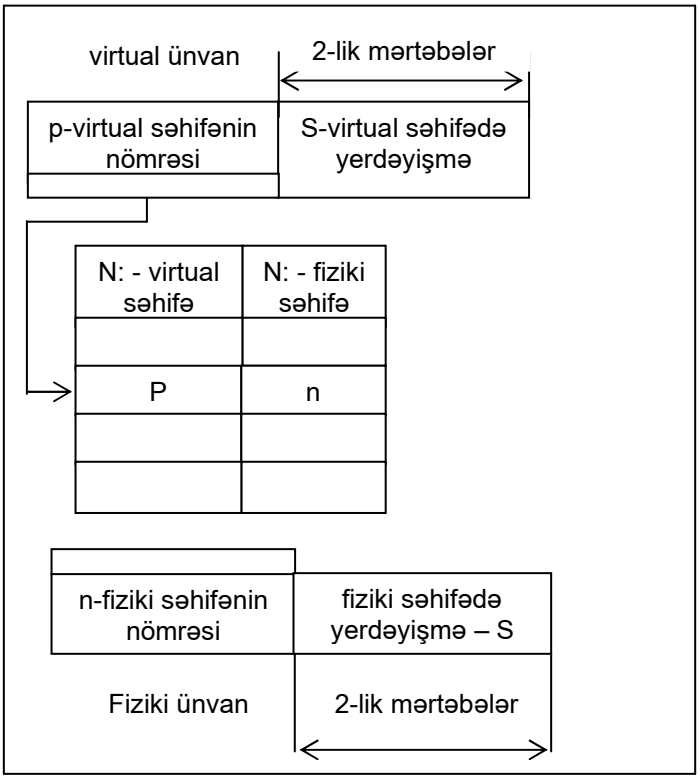
Operativ yaddaşa hər müraciət zamanı aparat vasitələri aşağıdakı əməliyyatları yerinə yetirir:

1. Səhifələr cədvəlinin başlanğıc ünvanına (səhifələr cədvəlinin ünvanı yerləşən registrin məzmunu), virtual səhifənin nömrəsinə (virtual ünvanın böyük mərtəbələri) və cədvəldə yazının uzunluğuna görə cədvəldə lazım olan yazının ünvanı müəyyən edilir.
2. Bu yazıdan fiziki səhifənin nömrəsi müəyyən olunur.
3. Fiziki səhifənin nömrəsinə yerdəyişmənin qiyməti (virtual ünvanın kiçik mərtəbələri) birləşdirilir.

Səhifənin ölçüsü 2^k olduğuna görə 3-cü bənd daha mürəkkəb toplama əməli əvəzinə birləşdirmə əməlini tətbiq etməyə imkan verir. Bunun nəticəsində kompüterin məhsuldarlığı artır.

Yaddaşın səhifələrlə təşkilində sistemin məhsuldarlığına virtual ünvanın fiziki ünvana çevrilməsi ilə və səhifə kəsilmələri ilə əlaqədar olan vaxt itkiləri təsir edə bilər. Vaxt itkisini azaltmaq və məhsuldarlığı artırmaq üçün səhifə kəsilmələrinin sayını azaltmaq və səhifələr cədvəlini daha sürətli yaddaşda yerləşdirmək lazımdır.

Səhifə kəsilmələrinin sayını azaltmaq üçün səhifənin ölçüsünü böyük göstərmək olar.



Şəkil 2.12. Yaddaşın səhifələrlə təşkilində virtual ünvanın fiziki ünvana çevrilməsi

Əlavə 2.

Windows XP sistemində kontekst menyuların bəzi əmrləri ilə iş

Windows XP sistemində hər bir obyektlə müxtəlif sayda əməliyyatlar aparıla bilər. Bu obyektlərə qovluqlar və fayllar da aiddir. Qovluq və fayllar bir-birindən fərqli obyektlər olduğu üçün onlara tətbiq edilən əməliyyatlar da müxtəlif ola bilər. Windows sistemində müxtəlif obyektlərlə işləmək üçün vahid universal əməliyyatlar siyahısı yoxdur. Ona görə də istifadəçilərin obyektlərlə işini asanlaşdırmaq üçün obyekt menyusundan, yəni kontekst menyusundan istifadə olunur. Hər bir obyekt üzərində siçanın sağ düyməsini basmaqla kontekst menyusunu açmaq olar. Bu kontekst menyusunda obyekt üzərində aparıla bilən əməliyyatları göstərən əmrlər siyahısı vardır. İstifadəçi ona lazım olan əməliyyata uyğun əmri siçanın sol düyməsini basmaqla icra edə bilər. Elə standart əmrlər vardır ki, onlar bir sıra obyektlərin kontekst menyusuna daxildir. Məsələn: **Open** (Открыть/Açmaq), **Delete** (Удалить/Ləğv etmək), **Cut** (Вырезать/Kəsmək) və s.

I. Qovluğun kontekst menyusunun bəzi əmrləri:

1) **Open** (Окрыть/Açmaq) – uyğun qovluğu açmaq. Bu zaman qovluğun pəncərəsi açılır və həmin pəncərədə qovluğun məzmunu öz əksini tapır.

2) **Search** (Найти/Axtarış) – uyğun qovluqda olan hər hansı bir faylı və ya qovluğu axtarmaq. Bu zaman dialoq pəncərəsi açılır və orada axtarılan faylın və ya qovluğun adı daxil edilir və search düyməsi basılır.

3) **Cut** (Вырезать/Kəsmək) – uyğun qovluq buferə göndərilir. Bu əmr obyektin özünün yerini başqa yerə dəyişdirmək üçün istifadə edilir.

4) **Copy** (Копировать/Kopiyalamaq) – uyğun qovluğun kopiyası buferə göndərilir. Bu əmr obyektin kopiyasını başqa yerə köçürmək üçün istifadə edilir.

5) **Paste** (Вставить/Yerləşdirmək) – buferdə olan qovluq istənilən yerə qoyulur və ya yerləşdirilir. Bəzi hallarda kontekst menyusunda bu əmr öz əksini tapmır. Bu da onunla əlaqədardır ki, bufer boşdur. Deməli, əgər **Cut** və ya **Copy** əmrlərindən biri yerinə yetirilərsə, onda uyğun obyekt buferə göndərilir və kontekst menyusunda Paste əmri öz əksini tapır.

6) **Delete** (Удалить/Ləğv etmək) – uyğun qovluğu ləğv etmək. Bu zaman ekranda açılan pəncərədə **Yes** (Да/Bəli) düyməsini basmaq lazımdır. Nəticədə qovluq ləğv edilir və səbətdə yerləşdirilir.

7) **Rename** (Переименовать/Yeni ad vermək) – uyğun qovluğun adını dəyişdirmək. Bu zaman qovluğa yeni ad verilir.

8) **Properties** (Свойства/Xüsusiyyətlər) – uyğun qovluğun xüsusiyyətlərinə baxmaq. Bu zaman qovluğun adına, yarandığı tarix və vaxta, qovluğun həcminə və s. baxmaq olar.

II. Faylın kontekst menyusunun bəzi əmrləri:

1) **Open** (Открыть/Açmaq) – mətn faylını açmaq. Bu zaman ekranda pəncərə açılır və bu pəncərədə faylın məzmunu öz əksini tapır.

2) **New** (Создать/Yaratmaq) – faylın tipinə uyğun fayl yaratmaq. Bu zaman bu faylın məzmununu dəyişdirmək olar, onu başqa ad altında yadda saxlamaq olar.

3) **Print** (Печать/Çap) – faylı printerdə çap etmək.

4) Qovluğun kontekst menyusunda olan bir sıra əmrlər, məsələn **Cut, Copy, Paste, Delete, Rename, Properties** və s. faylın da kontekst menyusunda öz əksini tapır və bu əmrlər eyni funksiyaları yerinə yetirir.

III. İş stolunun kontekst menyusunun bəzi əmrləri

İş stolunun kontekst menyusunu açmaq üçün iş stolunda boş yerdə siçanın sağ düyməsini basmaq lazımdır.

1) **New** (Создать/Yaratmaq) – iş stolunda yeni obyekt yaratmaq. Məsələn yeni qovluq yaratmaq olar. Bunun üçün açılmış kontekst menyuda **Folder** (Папку/Qovluq) əmrini icra etmək lazımdır. Bu halda iş stolunda yeni qovluq yaranır. Bu qovluğa ad verilir.

2) **Undo** (Отмена/İmtina etmək) – iş stolunda axırıncı yerinə yetirilmiş əməliyyatdan imtina etmək.

3) **Arrange Icons By** (Упорядочить значки/Nişanları nizamlamaq) – iş stolunda olan obyektləri nizamlamaq. Bu zaman bu obyektləri adlarına, yaranma tarixinə, ölçülərinə və tiplərinə görə nizamlamaq olar. Digər tərəfdən bu obyektləri avtomatik nizamlamaq olar və ya onları istədiyimiz kimi iş stolunda yerləşdirmək olar.

4) **Properties** (Свойства/Xüsusiyyətlər) iş stolunun xüsusiyyətlərinə baxmaq. Bu zaman iş stolunun fonunu vermək olar, yəni işlədiyimiz şəkili ekrana fon kimi vermək olar. Digər tərəfdən ekranın «zastavkasını» vermək olar, yəni hər hansı bir «zastavka» seçilir, vaxt müəyyən olunur, atributları verilir. Qoyulan vaxt bitdikdən sonra ekrana seçilmiş «zastavka» çıxacaq. Qeyd edək ki, bu «zastavkaya» parol da vermək olar.

1) Qovluğu və ya faylı səbətdən bərpa etmək üçün: Səbəti açmaq – uyğun qovluq və ya faylı tapmaq – siçanın sağ düyməsini basmaq – **Recover** (Восстановление/Bərpa) əmrini yerinə yetirmək.

2) Qovluq və ya faylı səbətdən ləğv etmək üçün: səbəti açmaq – uyğun qovluq və ya faylı tapmaq – siçanın sağ düyməsini basmaq – **Delete** (Удалить/Ləğv etmək) əmrini yerinə yetirmək.

3) Səbəti bir dəfəyə boşaltmaq üçün: siçanın oxunu səbətin üzərinə gətirməli – siçanın sağ düyməsini basmaq – … (Очистить корзину/Səbəti boşaltmaq) əmrini yerinə yetirmək.

4) İş stolunda olan obyekti disketə yazmaq üçün: oxu obyektin üzərinə gətirmək – siçanın sağ düyməsini basmaq – **send to** (отправить/göndərmək) – Disk A (Диск A/Disk A) – siçanın sol düyməsini basmaq.

5) İş stolunda hər hansı bir fayl obyektini tələb olunan qovluğa yerləşdirmək üçün: oxu fayl obyektinin üzərinə gətirmək – sağ düyməni basmaq – **Cut** (Вырезать/Kəsmək) əmrini yerinə yetirmək – tələb olunan qovluğun kontekst menyusunu açıb, **Paste** (Вставить/Yerləşdirmək) əmrini yerinə yetirmək.

6) Hər hansı bir qovluq daxilindəki fayl obyektini iş stoluna çıxartmaq üçün: Qovluğu açmaq – fayl obyektinin kontekst menyusunu açmaq – **Cut** (Вырезать/Kəsmək) əmrini yerinə yetirmək – İş stolunun kontekst menyusunu açmaq – **Paste** (Вставить/Yerləşdirmək) əmrini yerinə yetirmək.

7) İş stolunda hər hansı bir fayl obyektinin «yarlığını» yaratmaq üçün: Uyğun obyektin kontekst menyusunu açmaq – **Create Shortcut** (Создать ярлык/Yarlığı yaratmaq) əmrini yerinə yetirmək.

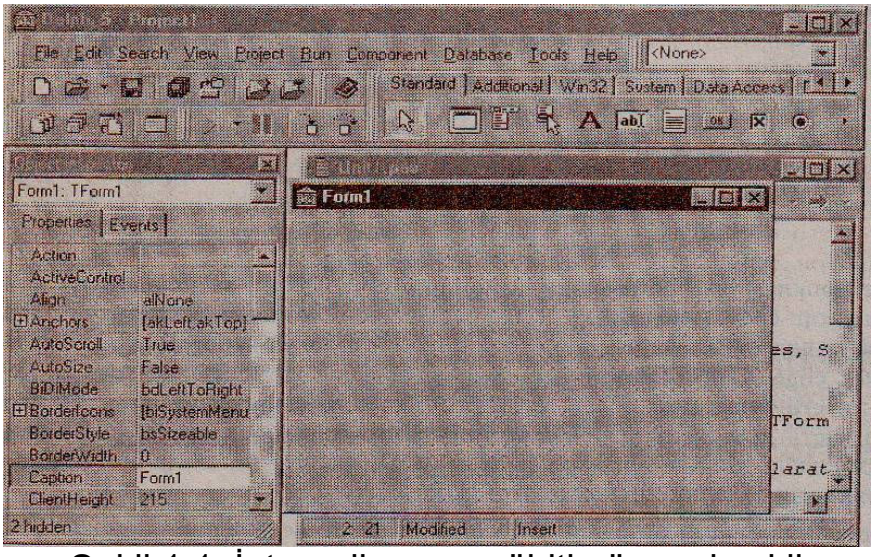
Delphidə proqramlaşdırma

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 Delphi-nin 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 (Интегрированной среде 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 1.1):

- əsas (baş) pəncərə (Delphi-də Project 1);
- obyektlərin inspektoru pəncərəsi (Object İnspector);
- formaların konstruktoru pəncərəsi (Form 1);
- kodların redaktoru pəncərəsi (Unit 1. PAS).



Şəkil 1.1. İnteqrallaşmış mühitin ümumi şəkli

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, Projjet, Run, Component, Database, Tools, Window, Help). Alətlər paneli baş menyunun alt sətrində yerləşir və 15 düymədən ibarətdir. Bunlardan istifadə etməklə baş menyunun tez-tez istifadə edilən əmrlərini yerinə yetirmək olar. Məsələ, File/open və ya Run/Run və s.

Alətlər panelinin 5 növü vardır;

- Standard (standart)
- View (baxış)
- Debug (tənzimləmə)
- Custom (istifadəçi)
- Desktop (iş stolu)

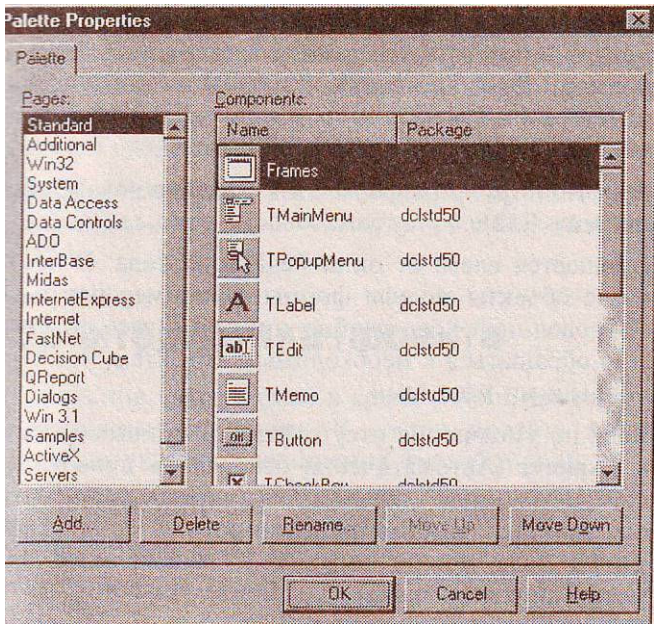
Alətlər panelini və düymələrin tərkibini dəyişmək olar (Siçanın göstəricisini alətlər paneli oblastında yerləşdirməklə və siçanın sol düyməsini basmaqla). Bu qayda ilə komponentlər palitrasının da əks olunmasını dəyişmək olar.

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 1.2).



Şəkil 1.2. Komponentlər palitrasının dialoq pəncərənin ümumi görünüşü

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.

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 proseduralar adlandırırlar.

Hər bir komponent özünə məxsus hadisələr və xassələr yığımına malikdir. Bu barədə növbəti paraqraflarda məlumat veriləcəkdir.

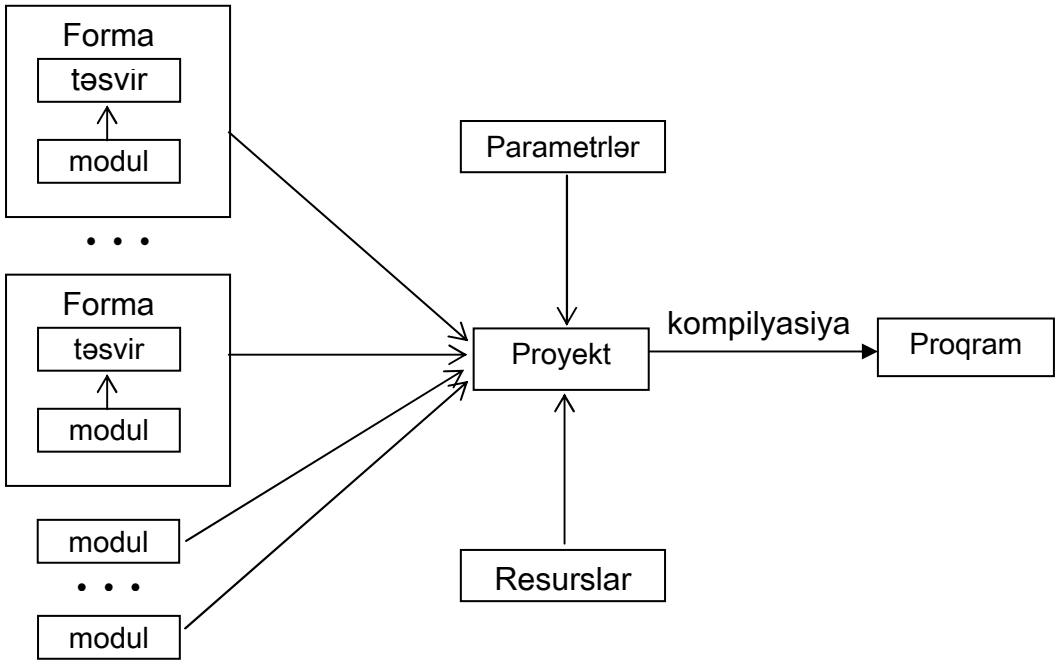
2. Proyektin xarakteristikaları

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 2.1-də proyektin ayrı-ayrı hissələri (fayllar) arasında əlaqə göstərilmişdir:



Şəkil 2.1. Proyektin faylları arasında əlaqə

Göstərilən fayllardan əlavə digər fayllar da yarana bilər. Məsələn, DPR – faylların ehtiyyat surətləri UDP – tipli və ya PAS faylların ehtiyyat surətləri ~PA tipli fayllar.

Delphi işə başlayan zaman avtomatik olaraq Project 1 – adlı yeni proyekt yaranır və bu ad əsas pəncərənin başlığında əks olunur. Bu proyekt Form 1 – 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.

2.1. Proyekt kodu faylı (DPR)

Proyektin kodu faylı proyektin əsas elementidir və bu xüsusi bir proqramdır. Bu proqram aşağıdakı şəkildədir:

```
Proqram Project 1;
USES
Forms,
Unit in ′Unit 1. PAS′ (Form 1);
{$ R· ∗. RES}
begin
Application. İnitialize;
Application. Greate Form (T Form 1, Form 1);
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 Form 1 formasının Unit 1 – moduludur.

{$R..} – direktivi resurslar faylını proyektə qoşur. Resurslar faylının adı proyekt faylının adı ilə (∗ - simvoluna görə) üst-üstə duşur. Bu fayldan əlavə {$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, Form 1-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 (Proyekt / просмотр источника) əmrini vermək lazımdır.

2.2. Formalar faylı (DFM)

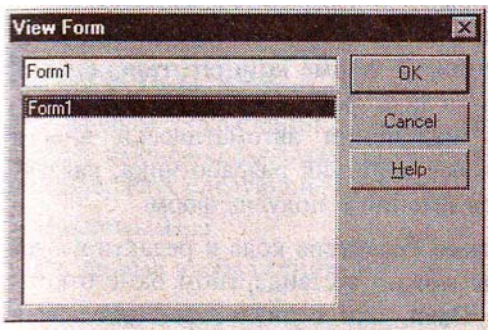
Proyektin tərkibində hər bir forma üçün avtomatik olaraq formanın təsviri faylı (DFM) və modul faylı (PAS) yaranır.

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 (şəkil 2.2.1).



Şəkil 2.2.1. Formanın konstruktoru pəncərəsinin açılması

Eyni zamanda bir neçə formaları ekranda əks etdirmək olar. Dialoq pəncərənin bağlaması da məlum qayda ilə yerinə yetirilir.

2.3. Formanın modulu faylı (PAS)

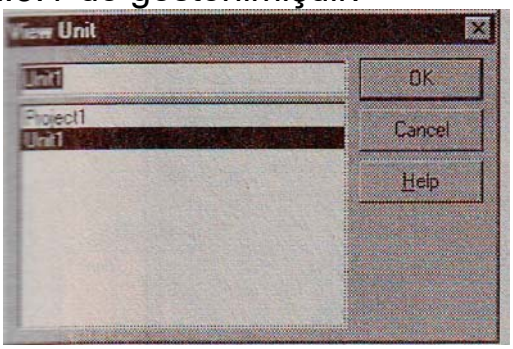
Bu faylda formaların sinfi təsvir edilir. Proyektin boş forması üçün bu modul aşağıdakı kodlardan ibarətdir:

```
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 2.3.1-də göstərilmişdir.



Şəkil 2.3.1. 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.

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).

2.5. Proyektin parametrləri (OPT)

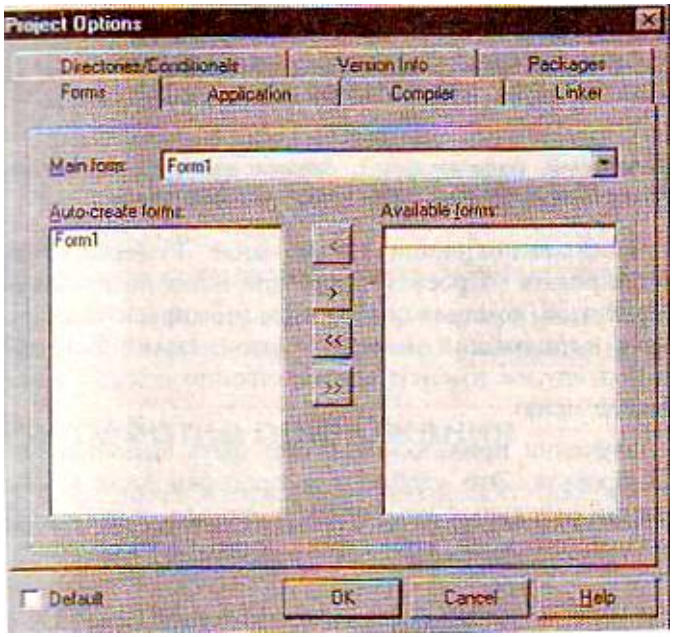
Proyektin parametrlərini təyin etmək üçün proyektin parametrlər pəncərəsindən istifadə etmək lazımdır (Project / Options və ya ctrl + Shift + F11 əmrlərindən istifadə eməklə). Parametrlər qruplara bölünür. Hər bir qrup ayrıca səhifədə yerləşir. Ayrı-ayrı parametrlərin qiymətləri təyin olunduqdan sonra Delphi avtomatik olaraq proyektin uyğun faylında dəyişiklik edir. Belə ki, Forms və Application səhifələrinin parametrləri proyekt və resurs fayllarına, Compiler və Linker səhifələrinin parametrləri isə proyekt parametrləri faylına aid olur.

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 2.5.1.-də göstərilmişdir.



Şəkil 2.5.1. Proyektin parametrləri pəncərəsi

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 <Project 1> <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 / Proqram 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.

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:

- Proqramın interfeysini hazırlamaq;
- 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.

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:

- Komponentlər palitrasından komponentlər seçilir və formada yerləşdirilir;

- 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>

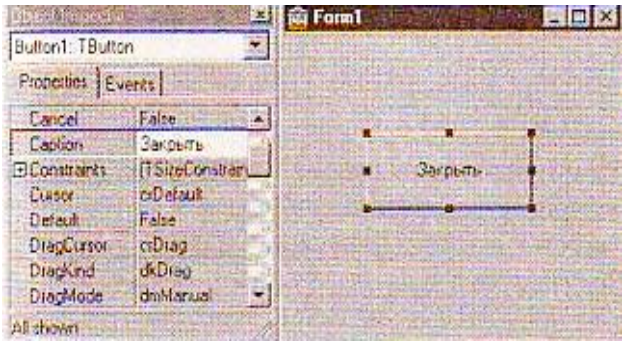
Məsələ: Button1: TButton;
Edit1: TEdit;
Label1: TLabel; - Modullar faylına bu sətirlər əlavə olunarsa, deməli formaya üç komponent gətirilmişdir: Label1, Edit1, Button1.

Komponentlər formaya qoyulduqda hər bir komponentə uyğun formanın təsviri faylında da dəyişiklik olur. Məsələ 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

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.

Məsələ, şəkil 4.1-də Button1 komponentinin xassəsinin verilməsi göstərilmişdir.



Şəkil 4.1. Komponentin xassələrinin verilməsi

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: Məsələ, 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) Button 1 komponentinin üzərində «SES» – yazılacaq.

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 Button 1 düyməsi sıxıldıqda yerinə yetirilməlidir. Məqsəd odur ki, bu düymə sıxıldıqda pəncərə bağlansın. Ona görə də prosedura daxilinə Form 1. 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, Closses, 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:

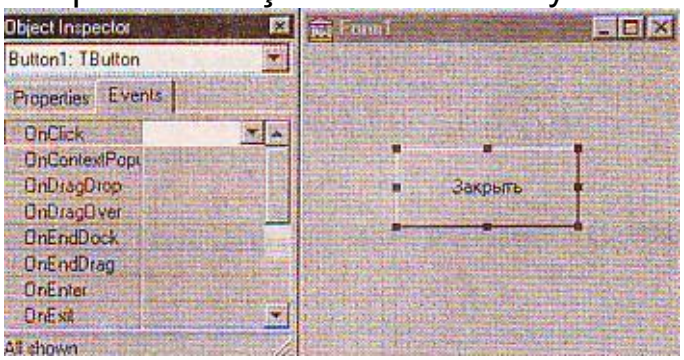
TForm1. Button1Click

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 proseduraşını proqram kodundan (Unit 1. 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 proseduraşı 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 4.2-də komponentlər üçün hadisələrin siyahısı verilmişdir.



Şəkil 4.2. Komponent üçün hadisələrin siyahısı.

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. Otladka vasitələri proqramda səhvlərin tapılması və aradan qaldırılmasına kömək edir. Otladka vasitələrinə müraciət üçün

RUN və View / Debug Windows –

əmrini vermək lazımdır.

Otladka 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.

Otladka vasitələrinin parametrlərini təyin etmək üçün

TOOLS / Debugger Options (Параметры отладки) – əmrini vermək lazımdır.

Açılan pəncərədə İntegrated Debugging-də bayraqcıq (v - belə bir işarə) varsa otladka qovulmuş olur.

v İnteqrated 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 zaman proyektin şərhçisinin Explorinq (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.

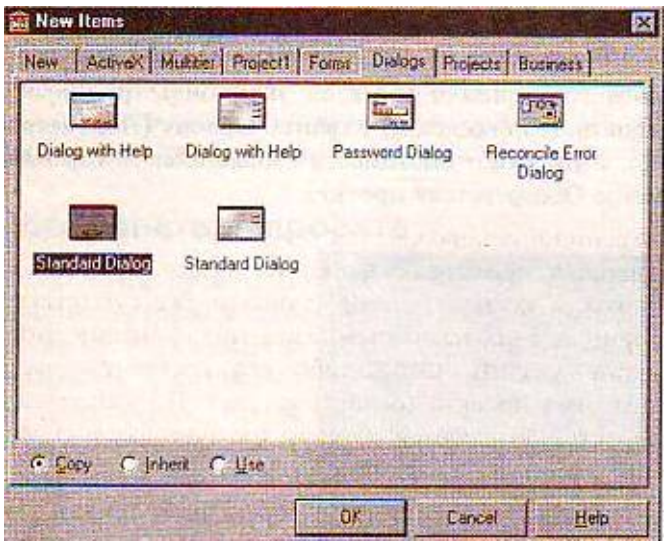
Hər iki halda Explorer Options pəncərəsi açılır ki, modul, sinif və qlobal dəyişənlərin idarə olunması üçün bu pəncərədə yerləşən parametrlərdən istifadə etmək olar.

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 5.1).



Şəkil 5.1. 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
- Proyekt 1 – tərtib olunan proyektin (proqramın) forması
- Forms – formalar
- Dialogs – dialoqlar
- Projects – proyektlər
- Business – formalar masteri

Şəkil 5.1-də Dialogs – pəncərəsi açılmışdır. Project səhifəsinin adı tərtib olunan proyektin adı ilə eyni olur və bu səhifədə Form 1 – 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 5.1-də 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:

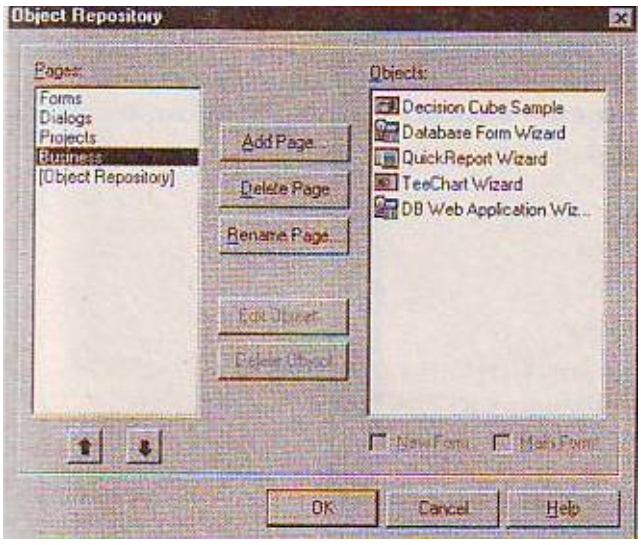
- Copy – proyektə arxivdən obyektin surəti (ekzemplyarı) əlavə olunur. Bu halda proyektdəki obyektdə dəyişiklik etmək olar və bu dəyişikliyin obyektin arxivdəki orijinalına təsiri yoxdur.
- İnherit – arxivdəki obyektdən yeni bir obyekt (törəmə obyekt) əmələ gəlir və proyektə bu törəmə obyekt əlavə olunur. 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 Project 1 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 5.2. 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. 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

6.1. Sadə tiplər

Sadə tiplərin aşağıdakı növləri var:

- tam tiplər
- simvol və ya liter tiplər
- məntiqi tiplər
- həqiqi tiplər

Там типляр: Физики типляр

№	Adı	Diapazonu	Yaddaşda tutduğu yer
1.	Shortint	-128 – 127	1 bayt işarə ilə
2.	Smallint	- 32768 – 32767	2 bayt işarə ilə
3.	Longint	- 2147483648 – 2147483647	4 bayt işarə ilə
4.	İnt 64	$-2^{63} – 2^{63} - 1$	8 bayt işarə ilə
5.	Byte	0 – 255	1 bayt işarəsiz
6.	Word	0 - 65535	2 bayt işarəsiz
7.	LongWord	0 – 4294967295	4 bayt işarəsiz

Ümumi tiplər:

№	Adı	Diapazon	Yaddaşın tutduğu yer
1.	İnteger	- 2147483648 – 2147483647	4 bayt işarə ilə
2.	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

Програмда ясас Боолеан типи истифадя олунур. Йердя галан цч тип диэяр програмлашдырма системляри иля уйьунлуг йаратмаг цчцндцр.

Qeyd: Sadal

Həqiqi tiplər:

№	Adı	Diapazon	Dəqiqlik	Yaddaşın tutduğu yer
1.	Real 48	$2.9.10^{-39} – 1.7.10^{38}$	11-12 rəqəm dəqiqlikdə	6 bayt
2.	Single	$1.7.10^{-45} – 3.4.10^{38}$	7-8 rəqəm dəqiqlikdə	4 bayt
3.	Double	$5.0.10^{-324} – 1.7.10^{308}$	15-16 rəqəm dəqiqlikdə	8 bayt
4.	Extended	$3.6.10^{-4951} – 1.1.10^{4932}$	19-20 rəqəm dəqiqlikdə	10 bayt
5.	Comp	$-2.10^{63} + 1 – 2.10^{63} - 1$	19-20 rəqəm dəqiqlikdə	8 bayt
6.	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.

6.2. 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.
- Strtoİnt (const S: String): İnteger – 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);

- proseduraşının 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);

- proseduraşından 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 fl: 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 proseduraşını (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 təyininдə of object yazmaq olar).

6.3. Variant tipi

Variant tipi müxtəlif üsullarla interpretasiya olunan qiym ətlrin təsviri üçün tətbiq edilir. Variant tipli dəyişənə müxtəlif tipə malik qiymətlər mənimsətmək olar. Bu tiplər o vaxt tətbiq olunur ki, tipin qiyməti kompilyasiya zamanы məlum olmasın və ya proqramын yeriня yetirilmя vaxtы dяyishsin. Bu tipli dяyishяnя tam giymяtli (int 64-dяn bashqa), shяgigi, simvol, sятir vя mяntigi tipli giymяtlяr mяnimsятmяk olar. Bu tiplяr Variant tipi ilя uyьundurlar vя tiplяrin chevrilmяsi avtomatik olaraq yeriня yetirilir.

Mясяlя;

```
Вар в1, в2: Вариант;
к: интеэер;
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.

6.4. 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 Button 1 Click – 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.

6.5. 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.

7. Vizual komponentlər kitabxanası

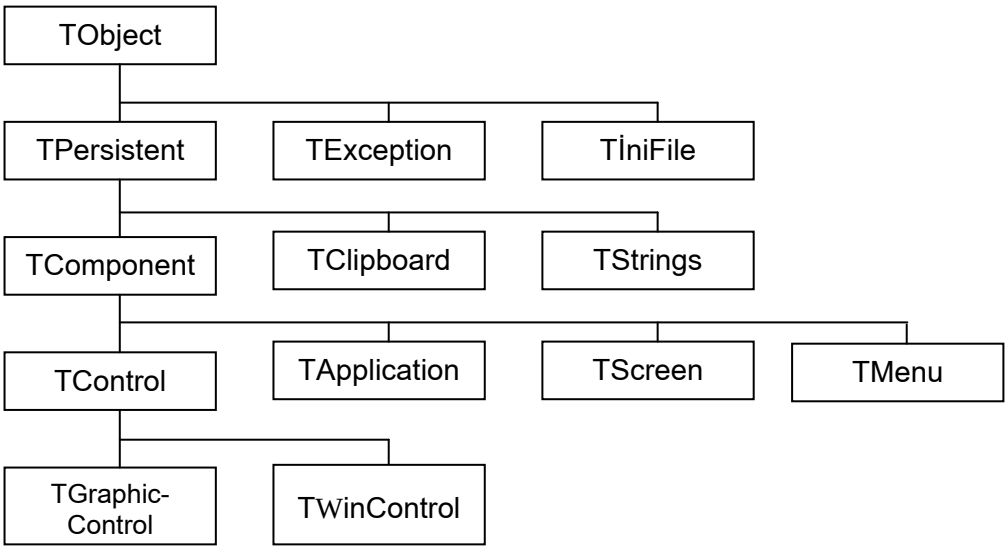
Vizual komponentlər kitabxanasında (VCL – Visual Component Library) çoxlu sayda siniflər yerləşir ki, bu siniflərdən proqramı sürətlə işləmək üçün istifadə olunur. VCL-də həm vizual, həm də vizual olmayan komponentlər yerləşir və onlar Delphi-nin inteqrallaşmış mühiti ilə əlaqədardırlar. Bütün komponentlər 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 7.1-də göstərilmişdir.



Şəkil 7.1. 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 siniflərdir.

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 ❑ Main Menu – baş menyu	❑ Bit Bitn – şəkillərlə olan düymə ❑ Spead Button – sürətli müraciət	❑ Tab Control – idarəedici ❑ Page Control – bloknot

❑ Pupup Menu – dəyişən menyu ❑ Label1 – yazı ❑ Edit 1 – birsətirli redaktor ❑ Memo – çoxsətirli redaktor ❑ Button – Standard düymə ❑ Check Box – bayraq ❑ Radio Button – çevirici ❑ List Box – siyahı ❑ Combo Box – siyahı ilə olan sahə ❑ Scroll Bar – fırlatma zolağı ❑ Group Box – qruppa	❑ Mask Edit – şablonla olan birsətirli redaktor ❑ String Grid – sətirlər cədvəli ❑ Draw Grid – cədvəl ❑ İmage – qrafik obraz ❑ Sape – həndəsi fiqur ❑ Bevel – faska ❑ Scroll Box – fırlatma oblastı ❑ Check List Box – çeviricilər siyahısı ❑ Splitter – ayırıcı ❑ Static Text – Statistik mətn ❑ Control Bar – alətlər paneli üçün konteyner	❑ İmage List – qrafik obrazların siyahısı ❑ Rich Edit – tamfunksional mətn redaktoru ❑ Trac Bar – hərəkət edən ❑ Proqressor Bar – işin gedişini göstərən indikator ❑ Up Down – sayqac ❑ Hotkey – düymələrin kombinasiyasının redaktoru ❑ Animate – video kliplərə baxmaq ❑ Date time Picker – tarixi daxil edən sətir ❑ Month Calendar – təqvim ❑ True View – obyektlər ağacı ❑ List View – siyahı
❑ Radio Group – asılı çevricilər qrupu ❑ Panel – panel ❑ Action List – əməliyyatların siyahısı	❑ Application Evens – hadisələr ❑ Chart – diaqramlar	❑ Header Control – ayrıcı ❑ Status Bar – vəziyyətlər sətri ❑ 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.

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 İnspector 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:

Edit 1. Read Only: = True;

Edit 1. 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.

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. LabelClick (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 koordinatı:′ + 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:

```
Procedure TForm1. 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 TForm1. 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 TForm1. 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.

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.

12. İnformasiyanın daxil edilməsi və redaktəsi

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. Məsələ,

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 (İnput 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
    Form 1. 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.

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.