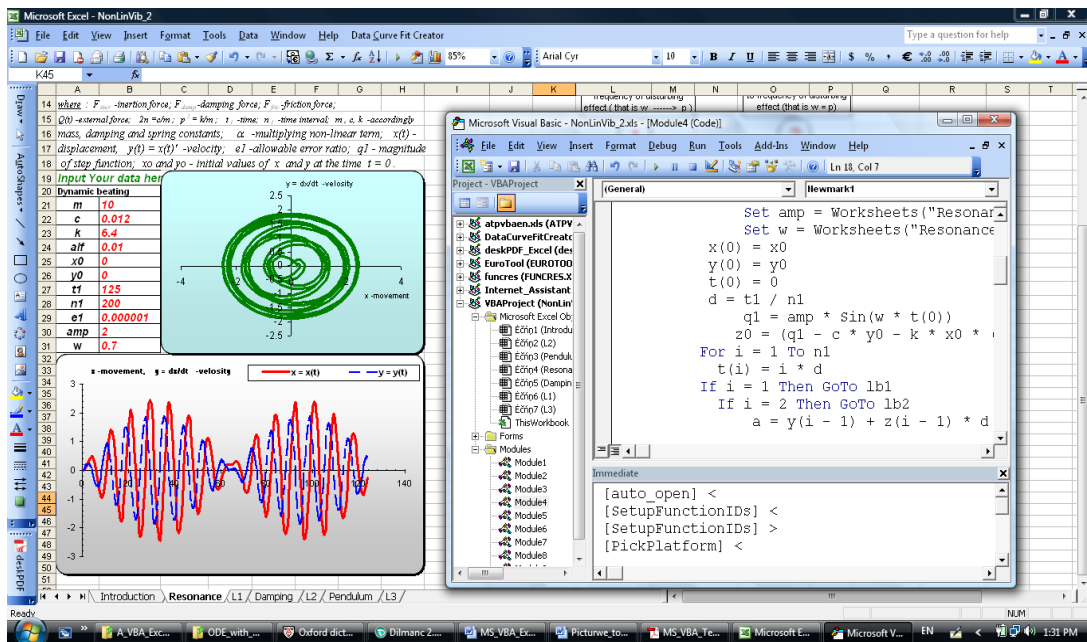


Sahib Hüseyinov

OBJEKTİYÖNLÜ

PROQRAMLADIRMA :



GƏNCƏ 2014

Müəllif - Sahib Tofiq oğlu Hüseynov: 24.03.1059 il tarixində Gəncə şəhərində müəllim ailəsində anadan olmuşdur. 1976-1982-ci illərdə Bauman adına Moskva Dövlət Texniki Universitetinin Konstruktorluq Mexanikası fakültəsində Maşınların və Aparatların Dinamikası və Möhkəmliyi sahəsində ixtisaslaşmışdır. 1986-cı ildə Azərbaycan EA-nın Riyaziyyat Mexanika İnstitutunun qiyabi aspiranturasına qəbul olmuşdur. Elmi işini SSRİ Akademiyasının Hesablama Mərkəzində aparmışdır. 1990-cı ildə mexanikanın qeyri xətti məsələləri sahəsində namizədlik dissertasiyasını müdafiə etmişdir. 2009-cu ildə 6 aylıq müddətdə Avropa Birliyinin Erasmus Mundus Elm və Təhsil sahəsində Əməkdaşlıq və İntegrasiya proqramı çərçivəsində Yunanistanın Saloniki şəhərindəki Aleksandr Texniki Universitetinə Post-Doktor qrant qalibi kimi GDU-dan göndərilmişdir. 30 ilə yaxındır ki, o, Azərbaycanda ali təhsil ocaqlarında müəllim kimi fəaliyyət göstərir. 2008-ci ildən bəri Gəncə Dövlət Universitetinin İnformatika kafedrasının dosent vəzifəsində çalışır. Gəncə şəhərindəki bir sıra istehsalat təşkilatlarında da mühəndislik fəaliyyəti ilə uğurla məşğul olmuşdur. Əsas elmi fəaliyyəti riyazi modelləşdirmə metodları və kompüter hesablamaları köməyi ilə texnikada, mühəndislik sahəsində və təbiət elmlərindəki məsələlərin araşdırılması ilə bağlıdır. Məşğul olduğu elmi sahə ilə bağlı müxtəlif beynəlxalq elmi konfranslarda çıxış edib. GDU-da işlədiyi müddətdə informatika sahəsi üzrə bir neçə sayda dərslük, metodik vəsait və elmi məqalələrin müəllifi olmuşdur (40-dan çox sayda). Otuz ildən artıq pedaqoji fəaliyyətlə məşğul olan S.Hüseynov texniki fənlərdən, riyazi fənlərdən və informatikaya aid fənlərdən dərslər deyib: -riyazi modelləşdirmə, -texnoloji proseslərin optimallaşdırılması, -sistem analizi, -riyaziyyatın xüsusi bölmələri, -ali riyaziyyatın bölmələri, -ümumi informatika, -əməliyyat sistemləri, -kompüter riyaziyyatı, -kompyuter modelləşdirilməsi, -kompüter qrafikası. -obyektyonlu proqramlaşdırma, -elektron cədvəllərin tətbiqi, -tətbiqi proqramlar paketləri v.s.



Elmi redaktor: Sakit Q.Verdiyev,
Azərbaycan Texnologiya Universitetinin İnformatika və telekommunikasiya kafedrasının müdiri, t.e.d., professor.

Rəy verənlər: Süleyman A.Məmmədov,
Azərbaycan Dövlət Aqrar Universitetinin İnformasiya texnologiyaları və sistemləri kafedrasının müdiri, t.e.n, dosent.

Nizami Ə.Allahverdiyev,
Azərbaycan Müəllimlər İnstitutu Gəncə filialının Riyaziyyat, İnformatika və onların tədrisi metodikası kafedrasının müdiri, f.r.e.n.

Dərs vəsaitində praktiki məsləhətlər formasında Microsoft Office əlavələrinin yaradılması üçün Microsoft Visual Basic for Applications (MS VBA) mühitində obyektönlü proqramlaşdırmanın əsasları verilib. Materiallar ayrıca fəsillər və fəsillərin bölmələri görkəmində oxucuya təqdim edilir. Hər bir fəsil və bölmədə qısa, yığcam və eyni zamanda oxucu üçün aydın şəkildə konkret mövzu ilə bağlı nəzəriyyənin əsasları və çox sayda nümunəvi misallar verilib. VBA-nın obyektönlü mühitində Office proqramlaşdırması praktiki tərəfdən mənimsənilsin deyərək hər mövzuya aid sərbəst öz üzərində işləmək üçün tapşırıqlar təklif edilir.

Dərs vəsaiti daha çox informatika, kompyuter elmi kimi ixtisaslarda təhsil alan bakalavr və magistrlər üçün nəzərdə tutulub. Bununla belə, digər ixtisaslarda təhsil alan bakalavr və magistrlər üçün də dərs vəsaiti faydalı ola bilər. Həmçinin Visual Basic for Applications mühitində Microsoft Office proqramlaşdırmasının əsaslarına yiyələnmək istəyən müxtəlif sahələrdə çalışan mütəxəssislər də kitabdan istifadə edə bilər.

Dərs vəsaitinə Azərbaycan Respublikası Təhsil Naziri Mikayıl Cabbarovun əmri ilə (əmr № 202, 18 fevral 2014 il tarixində) nəşr hüququ (qrif) verilmişdir.

UOT 681.3.07

Sahib Tofiq oğlu Hüseynov. **OBJEKTYÖNLÜ PROQRAMLASHDIRMA: Microsoft Visual Basic for Applications.** Dərs vəsaiti. Nəşriyyatı: müəllifin qərarı və razılığı ilə dərs vəsaiti PDF formatına çevrilərək İnternet-də (uyğun olan saytlarda) sərbəst və pulsuz olaraq yüklənib oxucular tərəfindən istifadə edilməsi üçün nəzərdə tutulub. Gəncə, 2014 il. 408 säh. Şəkilli.

ÖN SÖZ

Dünya üzrə 20 ildən artıqdır ki, ofis proqramlaşdırması (**Office programming**) adı ilə tanınan aktual və dinamik inkişaf edən istiqamət mövcuddur. Həmin sahədə Microsoft Office proqramlar paketi ilə birgə işləyən Microsoft Visual Basic for Applications (MS VBA) əsas vasitədir. VBA - obyekt yönlü proqramlaşdırma dillərinə aiddir və hal-hazırda ən müasir tələblərə cavab verən tətbiqi proqramların yaradılmasında istifadə olunan qabaqcıl texnologiyadır. Məşhur MS VB-dən (Microsoft Visual Basic nəzərdə tutulur) bu proqram təminatı əsas amillərə görə çox fərqlənir, ancaq bununla belə VBA-nın tətbiqi sahədə daha çox imkanları vardır. Yuxarıda qeyd etdiyimiz kimi - VBA yüksək səviyyəli, obyekt yönlü proqramlaşdırma dilidir və çox sayda müxtəlif proqramlar və proqramlar paketlərinin tərkibinə daxil edilmiş proqramlaşdırma dilidir. Microsoft Office proqramlar paketindən başlayaraq (Excel, Word, Access, FrontPage, PowerPoint v.s.) hətta Microsoft korporasiyasına aid olmayan belə AutoCAD, CorelDraw, Adobe Creative Suite proqramlarına qədər bu proqram təminatı öz geniş imkanları ilə proqramlaşdırma vasitəsi kimi istifadəçiyə təqdim edilir. Nəzərə alınmalıdır ki, həmçinin istehsal sahəsindəki proseslərin idarə etməsinə aid çoxsaylı xüsusiləşmiş proqramlarda da (ən sadə halda, məsələn, maliyyə resurslarının uçotu, yaxud müştərilərin məlumatlandırma vasitələri proqramlarında) bu proqram təminatı əsas proqramlaşdırma vasitəsi kimi standart paketə yüklənmiş olur. Office proqramlaşdırması son zamanlar böyük miqyasda işgüzarlıq idarə etməsində (business management), iqtisadiyyat və maliyyədə, elmi araşdırma sahələrində, ali və orta təhsildə böyük uğurlar qazanmışdır. Əsas səbəb kimi aşağıdakıları qeyd etmək olar:

- alqoritmik dilin müasir proqramlaşdırma dillərinin standartlarına cavab verməsi (VBA - obyekt yönlü struktura malikdir);
- dilin strukturu, sintaksisi və tərkibindəki standart elementlərinə görə heç də Pascal tipli dillərdən geri qalmır (əslində, bir çox hallarda, üstünlüyü də var);
- Windows və həmçinin MS Office tətbiqi proqramlar paketilə təbii bağlıdır - bununla da VBA yüksək dərəcədə çevik və geniş funksionallığa malikdir;
- İnternet mühitində müxtəlif lazımlı tətbiqi xassələri və metodları mövcuddur – beləliklə, İnternet mühitində də tam funksional proqramlar yaratmaq mümkün olur.

Yuxarıda göstərilən səbəblərə görə MS VBA dilinin öyrənilməsi (bu sahədə professional təlim kursunu keçmək) hal-hazırda vacib və aktual istiqamətdir. MS VBA proqramlaşdırmasının tədrisi ali (ola bilsin hətta orta) təhsildə perspektiv və olduqca zəruri sayıla bilər. Nəzərə alınmalıdır ki, ABŞ-da ali və orta təhsildə artıq 20 ildən çoxdur ki, VBA geniş miqyasda istifadə edilir. Rusiya federasiyasının bəzi aparıcı (hətta periferiyasında olan şəhərlərdə də) orta məktəblərində MS Office proqramlaşdırılması 10 ilə yaxındır ki, tədris planlarına salınıb və bu sahədə nəzərə çarpan uğurları var. MS VBA-nın praktiki biznes sahəsində olan aktuallığı isə şübhə doğurmamalıdır. Dərs vəsaitində Microsoft Office proqram paketinə aid çox sayda VBA dilinin sintaksisinə aid proqram fraqmentləri və praktiki dəyəri olan nümunələr gətirilib. Əsas vurğu VBA mühitində MS Excel, MS Word, MS Access v.s. tətbiqi proqramların yaradılmasına edilmişdir. Dərs vəsaitinin son fəslində Office paketinin əsas sayılan proqramlarında VBA ilə proqram tərtib etmək üçün sərbəst işləmək məqsədi ilə oxuculara tapşırıqlar təklif edilir. İnanmaq istədim ki, bu cür sistemli yaxınlaşma Visual Basic Applications proqram təminatının oxucu tərəfindən yaxşı mənimsəməsinə və onun Office paketinin proqramlarında VBA ilə obyektiv proqramlaşdırmada kömək edəcək.

Müəllif kimi oxucunun nəzərinə çatdırmaq istədim ki, təxminən 1996-cı ildən başlayaraq Microsoft Office proqramlar paketində MS VBA əsasında proqramlaşdırma məsələləri ilə məşğul olmağa başlamışam. Arzu edərdim ki, tərtib etdiyim dərs vəsaitim müxtəlif ixtisaslarda oxuyan bakalavr, magistr və Office proqramlaşdırması ilə maraqlanan mütəxəssislərə kömək etsin.

Dərs vəsaitinin Azərbaycan dilinin qaydalarına uyğun gəlməsinə, pedaqoji tərəfdən düzgün tərtib edilməsinə və mənəvi yardım etdiyinə görə xüsusi minnətdarlığımı Gəncə Dövlət Universitetinin Riyazi Analiz kafedrasının dosenti təcrübəli müəllim, pedaqoq və alim Əliqulu U.Quliyevə bildirirəm.

Müəllif: S.T.Hüseynov

Gəncə 2014

GİRİŞ

Bu kitabın Microsoft Visual Basic for Applications (MS VBA) alqoritmik dilində proqramlaşdırmanın əsas prinsiplərinin öyrənilməsinə kömək edəcək. Ən əsası budur ki, bu dildə istifadəçi öz proqramlarını yaratdıqda ən zəruri vərdişləri əldə etmiş olacaq. Dərs vəsaitinin əsas məqsədi – istifadəçinin MS Office proqramlar paketində VBA dili ilə proqramlaşdırmağın öyrənməsinə kömək etməkdən ibarətdir. Kitabdakı materialların anlanması üçün istifadəçinin başqa alqoritmik dillər ilə tanış olması və ya proqramçılıq təcrübəsinin olması vacib deyil. Həç bir zaman proqramlar və proqramlaşdırma ilə rastlaşmayan oxucular bu sahədə qısa vaxtda biliklərini və bacarıqlarını artırmaqla biləcəklər. Əgər oxucu artıq proqramlaşdırma sahəsində müəyyən təcrübəyə malikdirsə (yəni xüsusilə Visual Basic dili ilə tanışdırsa) onda o, kitabın bir çox bölmələrini daha asan mənimsəyəcək.

Kitabın asan və tez anlanması üçün burada kifayət qədər əyani misallar vardır ki, onlar çətin görünən materialların mənimsənməsində kömək edəcək və alınan biliklərin möhkəmlənməsini təmin edəcək. Dərs vəsaiti elə üslubda yazılıb ki, MS Office 2003 proqramlar paketi istifadəçiləri (həmçinin daha son versiya olan MS Office 2010 istifadəçiləri də) öz layihələrini VBA mühitində yaratdıqda kitabdən həmçinin sorğu vəsaiti kimi də istifadə edə bilsinlər. Beləliklə, VBA dilini mənimsəmiş istifadəçi MS Windows mühitində xüsusiləşmiş layihələrin yaradılması üçün çox güclü proqram tərtib etmə imkanını əldə etmiş olacaq.

KİTAB HAQQINDA ÜMUMİ MƏLUMAT

Bu kitabda VBA mühitində müəyyən dərəcədə vacib və zəruri olan proqramlaşdırma aspektləri işıqlandırılıb. Burada aşağıdakı mövzulara aid kifayət qədər məlumat əldə etmək mümkündür:

- makrosların yazılması və redaktə edilməsi;
- VBA redaktoru pəncərəsinin komponentlərinin təyinatı;
- VBA redaktorunda vizual proqramlaşdırma vəsaitlərindən istifadə;
- dialoq pəncərələri və istifadəçi interfeysinin digər elementlərinin yaradılması;
- VBA proqramlarının yaradılmasında mühüm rol oynayan obyektlərlə işləmə metodları;
- başqa proqramlardan VBA-proqramlarının işə salınması.

KİTAB KİMLƏR ÜÇÜN NƏZƏRDƏ TUTULUB

Dərs vəsaiti birinci növbədə universitetlərin müxtəlif ixtisaslarında təhsil alan bakalavr və magistrlər üçündür. Çünki MS Office proqramlar paketinin özü geniş sahələrdə çalışan müxtəlif mütəxəssislər üçün yaradılmışdır: iqtisadiyyat, maliyyə, biznes idarə etməsində və digər tərəfdən dəqiq elmlər və texniki disiplinlərdə belə VBA istifadəçisi geniş imkanlara malikdir.

İnformatika sahəsində dərs deyən müəllimlər üçün də (universitet, kolec, lisey və orta məktəbdə dərs deyən müəllimlər üçün) bu dərs vəsaiti çox faydalı ola bilər.

Həmçinin VBA-nın obyektiv mühitində Office proqramlaşdırmasını öyrənmək istəyən istənilən sahədə çalışan mütəxəssis bu kitabdən faydalana bilər. Öz əməyinin faydalı olmasını dəfələrlə artırmaq istəyən oxucu kitabda verilən nəzəri materialların və VBA-da tərtib edilən əyani misalların

köməyilə qısa vaxtda öz proqramlarını MS Office mühitində tərtib edib işə sala biləcək. Əlbəttə, dərs vəsaiti MS Office proqramlaşdırmasının (MS Office 2003-dən başlayaraq, 2010-cu versiyasına qədər istifadə edilməsi mümkündür) bütün aspektlərini əhatə edə bilməz. Bununla belə VBA dilinin özünün öyrənilməsi, onun qrafik redaktorunda işləmə vərdişlərinin mənimsənilməsi, müxtəlif tətbiqi məsələlərin Office proqramlaşdırması ilə həlli dərs vəsaitində kifayət qədər hərtərəfli verilib.

KİTABIN STRUKTURU

Kitab ön söz, giriş hissəsindən, 15 fəsildən və istifadə olunan ədəbiyyat siyahısından ibarətdir.

Giriş hissəsində dərs vəsaitinin tərtib edilməsinin məqsədi haqqında danışılır. Girişin “kitab haqqında ümumi məlumat” adlı başlığı altında Microsoft Visual Basic for Applications dilinin hansı dərəcədə oxucunun maraqlarının təmin edəcəyi haqqında məlumat verilir. Girişin digər hissələrində kitabın kimin üçün nəzərdə tutulduğu və hal-hazırkı oxunulan bölmədə dərs vəsaitinin strukturu ətraflı izah edilib.

Birinci fəsildəki ilk bölmələr Microsoft Visual Basic proqramlaşdırma dilinin və MS Visual Basic for Applications (VBA) dilinin yaranma və inkişaf tarixi, bu dillərin oxşar cəhətləri, fərqləri, Office proqramlaşdırma mühitinin tətbiqi sahələri haqqında məlumatlardan başlayır. Həmin fəslin bir bölməsi obyekt yönü proqramlaşdırmanın ümumi xassələri və VBA dilinin Office proqramlaşdırmasında olan tətbiqlərinə həsr edilib. Birinci fəsildə **makrorekorder** ilə Office proqramlar paketində, avtomatik olaraq, VBA proqramlarının yaradılması və yaradılmış makrosu işə salma haqqında təfərrüatlı materiallar verilib.

İkinci fəsil VBA-nın redaktoru ilə tanış olmağa həsr olunub. Burada oxucu VBA-nın qrafik interfeysindəki dialoq pəncərələri ilə tanış olur: VBA kodunun ümumi strukturu, onun necə və harada tərtib etmə qaydaları, əyani olaraq, göstərilir. İş əsnasında VBA-nın sorğu siteminin də ümumi strukturu ətraflı izah edilir.

Üçüncü fəsildə oxucu VBA dilinin strukturu, əsas elementləri, sintaksisi, proqramlaşdırma konstruksiyası ilə tanış olur. Burada VBA alqoritmik dilinin əsas tipləri, standart riyazi funksiyaları və dilin idarəetmə əməlləri əyani misallar ilə izah edilib.

Dördüncü fəsil VBA-da obyektlərlə və obyekt modelləri ilə işləmə qaydalarının öyrənilməsinə həsr olunub. Burada siniflər və obyektlərin xassələri haqqında danışılır. Digər tərəfdən əyani misalların köməyi ilə obyekt metodları haqqında geniş məlumatlar verilir.

Beşinci fəslin bölmələrində formalar, qrafik idarəetmə elementləri və hadisələr haqqında danışılır. Bu fəsildə oxucu çox vacib olan mövzu ilə tanış olacaq: VBA proqramını yaratdıqda onu necə interaktiv üsullarla idarə etməyin mümkünlüyü göstərilir.

Altıncı fəsildə VBA proqramı istifadəçi tərəfindən tərtib edildikdə yarana biləcək səhvləri və xətalərin aradan qaldırılma üsulları hərtərəfli izah edilir.

Yeddinci fəslin hissələrində Assistant və Balloon vasitələrinin VBA proqramının tərtibi və işlənməsi vəziyyətlərində onların interaktiv kömək rolu haqqında danışılır.

Səkkizinci fəsildə proqram səviyyəsində Office paketinin proqramlarında alətlər və menyu paneli ilə işləmə qaydaları verilib (əyani misallar daha çox MS Excel proqramına aiddir).

Doqquzuncu fəslin bölmələrində VBA köməyi ilə verilənlər bazasının yaradılması qaydaları öyrədilir. Verilənlər bazasının tərtib edilməsində ActiveX Data Objects (ADO) obyektlərinin əsas metodları haqqında danışılır.

Onuncu fəsil MS PowerPoint proqramında avtomatlaşdırılmış təqdimatlarda VBA proqramlaşdırması necə həyata keçirilməsindən bəhs edir.

On birinci fəsil tam olaraq MS Word mətn redaktorunda VBA Office proqramlaşdırmasına həsr olunub. Oxucu bu hissənin bölmələrində real yarana biləcək hallarda Word-da tətbiqi məsələlərin VBA ilə avtomatlaşdırılmasını çox saylı misallarda öyrənə biləcək.

On ikinci fəsildə MS Excel elektron cədvəllər sisteminə hesablamaların tərtib edilməsi VBA proqramlaşdırması səviyyəsində öyrədilir. Burada oxucu Excel-də sadə misallardan başlayaraq, diferensial tənliklərin ədədi həllinə qədər, proqramların tərtib etmə qaydalarını öyrənə bilər. Daha çox diqqət verilənlər bazalarının yaradılmasına yönəldilib.

On üçüncü fəsildə əyani misalların köməyi ilə MS Access proqramında mürəkkəb strukturlu verilənlər bazalarının avtomatlaşdırılması VBA proqramlaşdırması səviyyəsində öyrədilir.

On dördüncü fəsil MS Outlook proqramında İnternetlə bağlı biznes məsələlərinin avtomatlaşdırılmasını VBA proqramlaşdırması ilə həyata keçirilməsinə həsr edilib. Xüsusi diqqət istehsalat və biznes proseslərində yaranan *Personal Information Manager* (PIM) texnologiyasının tətbiqinə verilib.

On beşinci fəsildə oxucunun sərbəst işləməsi üçün verilmiş tapşırıqlar dərs vəsaitinin bütün hissələrindəki bəzi əsas sayılan tətbiqi məsələlərin praktiki həllinə aiddir. Tapşırıqlar ətraflı göstərişlər və mümkün olan cavablarla təmin edilib.

1. MS OFFICE PROQRAMLAŞDIRMASININ ƏSASLARI

1.1 Visual Basic salnaməsi

Keçən əsrin 90-cı illərinin sonunda Microsoft kompaniyası yeni tərtib etdiyi Windows əməliyyat sisteminin geniş yayılması üçün çox aktiv mübarizəyə başlamışdı. Məlum olduğu kimi, əməliyyat sistemindən də çox istifadəçilərə (əlbəttə, yüksək və orta səviyyəli istifadəçilər nəzərdə tutulur) lazımı səviyyədə professional işlərin yerinə yetirməsi üçün onlara xüsusi proqram mühiti lazım olur. Belə proqram mühitləri istifadəçi üçün son dərəcə əlverişli olmalıdır [2]. Kompüter texnologiyasındakı proqramlar təminatında hətta belə bir termin mövcuddur: *“istifadəçi ilə dostluq mühiti”*. Digər tərəfdən əməliyyat sisteminin dəyişməsi bəzən proqramçılar (professional istifadəçilər) üçün çox ciddi problem kimi meydana çıxır. Çünki onlar məcbur olurlar ki, proqramların yeni tərtib etmə texnologiyasını mənimsəsinlər. Microsoft kompaniyası isə çox düzgün bir şüar seçdi: “yeni başlayan naşı proqramçılar da, nəhayət, Windows tətbiqi proqramlarında proqramlaşdırmağı bacarmalıdır”. Bununla da Microsoft kompaniyası Visual Basic (MS VB) alqoritmik dilinin instrumental strukturunun birinci versiyasını geniş kütləyə çıxara bildi. Microsoft kompaniyası bu məhsulu əsasən yeni başlayan və qeyri professional proqramçılar kateqoriyası üçün nəzərdə tutmuşdur. Həmin dövrdə kompaniyanın əsas məqsədi geniş bazara bu cür universal aləti çıxarmaqdan ibarət idi. Çünki hətta təcrübəli proqramçılar yeni yaradılmış Windows mühitində proqram tərtib etməkdə həqiqətən çətinlik çəkirdilər. Həmin vaxtlardakı MS Visual Basic 1.0 işçi alətdən də çox gələcək tərtib etmə mühitinin maketinə bənzəyirdi. Bu məhsulun prinsiplial yeni cəhəti ənənəvi proqram tərtib etmə texnologiyasından köklü olaraq fərqlənirdi: Windows mühitində özünə məxsus vizual strukturlu idarəetmə prinsiplərinə əsaslanırdı. Beynəlxalq razılışmaya görə MS VB ilk olaraq proqramların qısa zamanda və tez tərtib etmə (*Rapid Application Development, RAD*) vasitələri alətləri nəslinin əcdadı sayılır. Visual Basic dilinin BASIC dili ilə bir o qədər də uyğunluğu yoxdur (VB obyektiv və struktur xassəli proqramlaşdırma dilidir - Object Pascal-dan, demək olar ki, çox az fərqlənir [3, 10, 11]). Bu dildə Basic-in əsas konstruksiyaları müəyyən qədər saxlanılmışdı və dilin strukturu (proqramın yazılma tərzini) ingilis dilinin strukturuna (qrammatikasına) yaxınlaşdırılmışdı. Bu dildə yazılmış proqramları ingilis dilində yazılmış mətn kimi oxumaq əslində olduqca asandır. Hal-hazırda bu ideologiya çox təbii və adi qəbul edilir. Həmin dövrlərdə “keçmiş vaxtın” proqramçıları üçün bu cür sistemlər tamamilən qeyri adi bir mühit kimi görünürdü (hətta sırf psixoloji tərəfdən belə) və onlar üçün ciddi problemlər meydana çıxırdı. Bütün bunlara baxmayaraq, VB istifadəçilərinin sayı selə bənzər kimi artırdı. Digər tərəfdən VB ildən ilə daha da mükəmməlləşirdi: həm dilin professional elementlərini modernləşdirir həm də tam obyektiv vasitələrinin xassələrini yüksək səviyyədə inkişaf etdirilirdi. Və, nəhayət, 1995-ci ildə VB 4.0 proqram təminatı satışa çıxarıldıqda - artıq bu alqoritmik dil müxtəlif tətbiqi proqramların yaradılmasında geniş tətbiq sahəsi olan bir alət kimi beynəlxalq şöhrət qazanmışdır. Hal-hazırda VB 6.0 versiyası hələ də geniş istifadə edilir. Hal-hazırda VB-nin bir sıra yeni versiyaları da mövcuddur: MS VB 7.0, NET MS VB.NET və son məhsul MS VB 2010 (MS VB 2011 məhsulu üzərində işlər tamamlanmaq mərhələsindədir).

1.2 MS Visual Basic for Applications (VBA) Office proqramlaşdırma mühitinin tarixi inkişaf yolu, yaradılmasının zərurəti və inkişafı

Yenə 90-cı illərin əvvəllərində (MS VB yaranan periodda) belə bir tendensiya kəskin olaraq özünü büruzə verdi: istifadəçinin daha mükəmməl işləməsi üçün proqram təminatına aid olan proqramlara daxili proqramlaşdırma vasitələrinin birləşməsi (calanması) zərurəti yarandı [1]. Bununla istənilən səviyyəli istifadəçi (yeni proqramlaşdırma üsullarını bilən və bacaran mütəxəssis) daha əlverişli üsullarla həmin proqram paketlərini, konkret şəraitə uyğun olaraq, sazlanmasını və tətbiq şərtlərini hərtərəfli nəzərə alaraq, təkmilləşməsinə həyata keçirə bilərdi. Buna görə 1993-cü ildə Microsoft kompaniyası VB əsasında kompaniyanın tətbiqi proqramları (MS Office proqramlar paketi nəzərdə tutulmuşdu) üçün yeni universal proqramlaşdırma sisteminin yaradacağını elan etdi. Həmin sistemin adı Visual Basic for Applications (texniki tərəfdən izahlı tərcümə etsək - tətbiqi proqramlarda istifadə edilən VB). Bu sistemin birinci versiyası MS Office 4.0 tərkibində MS VBA 1.0 adı ilə meydana çıxdı. Lakin bu proqramlaşdırma dili MS Office proqramları paketindən yalnız Excel 4.0 və Project 6.0 proqramlarına daxil edilmişdi. Həmin vaxt MS Word 6.0 və MS Access 2.0 proqramlarına bu sistem əlavə edilməmişdir. Digər tətbiqi proqramlarda kompaniyanın tərtib etdiyi məxsusi xarakterli Microsoft Basic versiyaları əlavə edilmişdir. Keyfiyyət xarakterli dönüş 1996-cı ildə MS Office 97 proqramlar paketinin istismara buraxılması ilə yarandı. Bu versiyada VBA 5.0 proqramlaşdırma mühiti artıq demək olar ki, bütün əsas MS Office proqramlarına əlavə edilmişdi: Word, Access və PowerPoint proqramları kimi MS Office proqramlar paketinə aid proqramlarda. Bundan başqa, VBA 5.0 mühitinin proqramlaşdırma dil mexanizmasının strukturu VB 5.0-dən fərqlənmirdi. MS Office 2000 paketinə VBA 6.0 daxil edilmişdi ki, artıq bu sistem hətta Outlook, Frontpage proqramlarına da əlavə edilmişdi. Bunun nəticəsi kimi hal-hazırda Microsoft indi öz MS Office paketini sadəcə bir tətbiqi proqramlar toplusu kimi yox, daha çox istifadəçilərin xüsusi təmayüllü geniş əhatəli biznes-əlavələri kateqoriyasına aid məsələsinin kompleks platforması kimi təqdim edir. Buna görə MS Office paketinin Developer Edition və Business Application versiyaları satışa buraxılır. Digər tərəfdən proqramlaşdırılan tətbiqi proqramların idarə edilməsi sahəsində VBA standart kimi artıq qəbul edilib. Son zamanlar yüzdən artıq dünyadakı aparıcı kompaniyalar MS VBA proqramının lisenziyasını əvvəlcədən əldə edərək, onu öz proqram məhsullarının tərkibinə əlavə edirlər. MS VBA mühitində proqramlaşdırma əslində təhsil və tədrisdə də böyük uğurlar qazanmışdır. Çünki bu sistemdə proqramlaşdırmanı bacaran istənilən ixtisasın tələbəsi gələcəkdə bir mütəxəssis kimi çox güclü və çevik bir mühiti mənimsəmiş olur. Gələcəkdə heç bir çətinliksiz yüzlərlə başqa daha mürəkkəb sistemlərdə işləmə təcrübəsini qazanmış olur. VBA ilə sadə mikrokomandaların tərtibindən başlayaraq, gələcəkdə həmin alətlər sisteminə istənilən mürəkkəbliyə proqramları tərtib edən çox yüksək səviyyəli peşəkar proqramçı olmaq mümkündür. İyirmi il bundan əvvəl bütün dünyada təxminən iki milyon proqramçı tapmaq mümkün idi. Bu gün isə dünyada təxminən 15 milyon proqramçı saymaq olar ki, onların içərisində orta hesabda 70% VB-dən yaxud, daha çox, VBA-dan istifadə edirlər.

1.3 Müasir proqramlaşdırma texnologiyalarının əsas xassələri

Artıq 20 ilə yaxındır ki, EHM-in tətbiqi proqramları ilə paralel onun proqramlaşdırılması üçün istifadə olunan alqoritmik dillərin ümumi bir xassəsi əmələ gəlmişdir [2]. Bu cür müvafiq proqramlaşdırma mühitləri də yeni mərhələyə keçmişdir və keyfiyyətcə yeni proqramlaşdırma psixologiyası yaranmışdır (MS WINDOWS-un yaradılması məhz bu dillərin meydana çıxması ilə bağlıdır). Şübhəsiz ki, kompyuterlərin texniki təminatının durmadan sürətlə inkişafı (işləmə tezliyinin və yaddaşının kəskin artması) buna zərurət imkanları vermişdir. Proqramlaşdırmanın bu yeni texnologiyası, proqramları ayrı-ayrı obyektlərdən quraşdırdığına görə, obyektistifadəli (və ya vizual, yeni görüntülü) proqramlaşdırma adlandırıldı və bu texnologiya struktur proqramlaşdırmaya nisbətən yeni mərhələdir. Maşın kodlarında yazılmış proqramla müqayisədə universal alqoritmik dillərin birində yazılmış proqram nə qədər irəliyə atılmış böyük addım idi isə - müasir **obyektistifadəli proqramlaşdırma** da həmin 60-90-cı illərdə mövcud olan yüksək səviyyəli alqoritmik dillərə nisbətən irəliyə daha böyük addımdır və proqramın yaradılması üçün sərf olunan vaxtı dəfələrlə qısaltmağa imkan verirdi.

Bu texnologiya ilk olaraq, yuxarıda qeyd etdiyimiz kimi, MS Windows ƏS-i və MS Office paketinin proqramlarında tətbiq olunmağa başlandı. Müəyyən ardıcılıqla yerinə yetirilən əməliyyatları bir yeni əmrdə (yeni "düymədə") cəmləşdirmək üçün makroslardan (makrokomanda strukturlu proqramlardan) istifadə olunmağa başlandı. Makroslar dedikdə - müvafiq proqram mühitinin müəyyən ardıcılıqla yerinə yetirilən müxtəlif əməliyyatların (müəyyən ardıcılıqla basılan "düymələrini") bir-birinə bağlayan kiçik proqramlar toplusu kimi də anlamaq olar.

Obyektyönlü proqramlaşdırmada (OYP) üç əsas baza elementindən istifadə olunur: -obyektlər; -əlaqələndirici məlumatlar; -siniflər. Özünəməxsus xassələrə (əlamətlərə), dəyişənlərə və metodlara malik proqram fraqmenti (daha doğrusu bu proqram fraqmentinin qrafik təsviri) **obyekt** adlanır. WINDOWS mühitini özündə və onun tətbiqi proqram paketlərində rast gəldiyimiz bütün idarəedici elementlər (məhz düymələrin, menyuların, daxil etmə, xaric etmə sahələrinin v.s.. piktoqramları) bu mühitlərin obyektləridir. Təbii dildə hər bir cümləni (və yaxud hadisəni) bir işarə ilə göstərə bilsə idik (məsələn, heroqliflər kimi), onda onları da obyekt adlandırma bilərdik (insan yaddaşı üçün bu qədər işarələri yadda saxlamaq qeyri-mümkün olsa da, müasir EHM-lərin belə «problemi» yoxdur). **Obyekt dəyişənləri** - adi proqramlaşdırma dillərindəki kimi sadə verilənlər (ədədlər, massivlər, mətn və s.), mürəkkəb strukturlu informasiya (qrafika, audio, video və s.) ola bilər. Obyekt dəyişənində verilən kimi başqa obyektədən istifadə olunursa, onda belə obyektlər mürəkkəb obyekt adlanır. **Metodlar** - obyektin iş alqoritmını təyin edən proseduralar və funksiyalar toplusudur. Bu funksiyalar və proseduralar obyektin yerinə yetirəcəyi işləri təyin etdiyi üçün - metodu obyektə verilən tapşırıq kimi qəbul etmək olar. Obyektlərdə üç əsas prinsipə riayət olunur:

- **irsilik** – törəmə obyekt öz «valideyninin» xassələrini (əlamətlərini) saxlayır;
- **inkapsulyasiya** – obyektin verilənləri və metodları xarici mühitdən izolə olunur (başqa istifadəçi müdaxilə edə bilmir);
- **poliformizm** – obyektin formasını müxtəlif cür dəyişmək olar;

Əlaqələndirici məlumatlar, obyektləri bir-biri ilə əlaqələndirən vasitələrdir və onlar üç hissədən ibarətdir: **çağırılan obyektin ünvanı** - identifikatoru, çağırılan obyektə yerinə yetirilən **metodun adı** və verilmiş metodun yerinə yetirilməsi üçün vacib **əlavə informasiya**.

Siniflər - eyni tipli obyektləri birləşdirən şablonlardır. Böyük proqramlarda istifadə olunan eyni tipli obyektlərin hər dəfə istifadəsi zamanı, onların metodlarının və dəyişənlərinin müəyyən olunması proqramın işinin effektivliyini azaldır. Buna görə də belə sinif-şablonlardan (baza sinfi) istifadə edirlər.

Hər bir hazır obyekt (belə obyektlər minlərlədir) müəyyən proqram kodundan – altproqramdan ibarətdir. Onun istifadəsi zamanı proqramçı yalnız obyektin qrafik təsviri ilə işləyir (işin nəticəsini gözü ilə görür). Buna görə də OYP bəzən - **vizual proqramlaşdırma** da adlanır (ancaq “*visual programming*” terminin hal-hazırda bir qədər başqa müasir mənası əmələ gəlmişdir).

Obyektyönlü proqramlaşdırma mühitləri müasir kompyuter istifadəçisinin qarşısına çıxan bütün məsələlərin optimal həll yollarını tapmağa və proqramın müasir istifadəçi interfeysini (WINDOWS-la işləyən proqramlarda rast gəldiyimiz alətlərlə və vasitələrlə təchiz olunmuş mühitlər) yaratmağa imkan verir. Bu mühitlər bütün populyar tətbiqi proqramların formatları ilə işləyə bilər, burada İnternet şəbəkəsi üçün sənədlər hazırlamaq mümkündür, müxtəlif verilənlər bazaları ilə işləmək olur v.s.

Obyektyönlü proqramlaşdırma mühitləri.

İDE mühitləri – *inteqrasiyalı layihələndirmə mühitidir* (Integrated Development Environment – IDE). Proqramlaşdırmaya tamamilə başqa cür yanaşma olduğuna görə bu mühitdə yaradılan proqram da layihə (**proyekt**) adlanır və, müxtəlif standart və xüsusi obyektlərdən istifadə olunaraq, qurulur. Proqramın yaradılması (qurulması) üçün olan bu sahə (layihənin çəkilməsi üçün istifadə etdiyimiz müəyyən formatlı kağızla müqayisə etmək olar) **forma** adlanır. Deməli **forma** bu mühitin ilk və əsas obyektidir. Bu mühitlərdə işin tam mənimsənilməsi (müxtəlif obyektlərin xassələrini və onların metodlarını öyrənmək) istifadə olunan proqramlaşdırma dilinin (proqram kodunun) mənimsənilməsi qədər vacibdir.

1.4 VBA ilə Office-də proqramlaşdırmaq nə üçün lazımdır?

İstehsalatda, biznesdə, elmi-tədqiqat mərkəzlərində Office-də proqramlaşdırmaq dedikdə birinci növbədə çox sayda təkrarlanan əməliyyatların (“əl işlərinin”) azaldılması ilə bağlı hesablamaların avtomatlaşdırılmasına yönələn tədbirlər nəzərdə tutulur. Aşağıda bu növ proqramlaşdırmada yaranan tipik hallar göstərilib [17]:

- müəyyən olan periodla bir-birinə oxşar sənədlərin tərtib edilməsi lazım olduqda: mühasibata aid əmrlər, sərəncamlar, razılaşmalar, müqavilələr, hesabatlar v.s. Çox hallarda məlumatları verilənlər bazasından götürülməsi daha əlverişli olur, onda proqramlaşdırmağın faydası zamana qənaət baxımından çox yüksək ola bilər. Hətta, bəzən məlumatları “əl ilə” daxil etdikdə belə avtomatlaşdırma tədbirləri (zamana görə) uduşu və yarana biləcək səhvlərin minimuma endirilməsi baxımından mənfəətli olacaq;
- həmin halın başqa forması: eyni verilənlərdən dəfələrlə istifadə etmək lazım olduqda. Məsələn, sifarişçi ilə müqavilə bağlanır. Eyni verilənlər (adlanmalar, ünvanlar, haqq-

hesab hesabatı, müqavilənin nömrəsi, imzalama tarixi, məbləğ v.s.) çox sayda olan sənədlər formalarında tələb ola bilər. Müqavilənin özündə, haqq-hesab fakturasında, işlərin təlim-təslim aktında v.s. Daha məntiqə uyğun olar ki, həmin verilənləri bir dəfə daxil edib (verilənlər bazasına) sonra isə avtomatlaşdırılmış halda tələb olan sənədləri hazırlamaq olar (məsələn, Word-də və ya Excel-də);

- istifadəçi tərəfindən “əl ilə” daxil etdiyi verilənlərin təkrarlanması avtomatlaşdırılması lazım olduqda. “Əl ilə” verilənlərin daxil edilməsi prosesində səhvlərin ehtimalı bir çox faktorlardan asılıdır. Bir sıra araşdırmalar göstərir ki, cəmi 2% təşkil edə bilər. O biri tərəfdən isə bu cür səhvlərin müəyyən edilməsi - “ağır zəhmətli işlər” kateqoriyasına aiddir. Buna görə belə səhvlərin, ümumiyyətlə, yaranmaması üçün daha əlverişli yol bəri başdan ağır “əl zəhməti” işlərinin proqramlaşdırma üsulları ilə avtomatlaşdırmaqdır.

Ümumiyyətlə, istənilən əməliyyatın çox sayda təkrar edilməsi lazım olduqda həmin işlərin avtomatlaşdırılması zəruridir. Məsələn, yüzlərlə kontaktların MS Outlook proqramına daxil edilməsi, yaxud MS Project-də layihələrin resurslarının dəyişdirilməsi, və ya MS Excel-də müxtəlif zaman periodunda verilənlər bazasındakı məlumatların analizi. Office proqramlarının obyekt modellərindən peşəkarcasına istifadə edilməsi böyük və kiçik təşkilatları “lazımsız ağır əl işlərindən” azad etmiş olur.

Əlbəttə, cavan və kvalifikasiyası aşağı olan əməkdaşların əmək resursu da həmişə istənilən qədər olur. Digər tərəfdən proqramlaşdırmanın tətbiqinin özü, işində avtomatlaşdırılmış hesablamaları istifadə edən, əməkdaş üçün başqa üstünlüklər gətirə bilər:

- müdiriyyət qarşısında və başqa işçilərin nəzərində nüfuzu arta bilər;
- müəssisədə və ya təşkilatda həmin istifadəçinin proqramları, fəal olaraq, istifadə olunurlarsa (onun özü yaxud işçilər tərəfindən), onda o özünü ixtisarlardan, maaşın azalmasından v.s. xoşa gəlməz hadisələrdən sığorta etmiş olur – çünki həmin tərtib edilmiş proqramları dəyişdirmək lazım olduqda başqa mütəxəssis tapılmaya bilər.

1.5 VBA DİLİ NƏ DEMƏKDİR?

VBA-nın tərif, üstünlüyü, imkanları və tətbiqi

MS VBA (Microsoft Visual Basic for Applications) — Visual Basic alqoritmik dilinin imkanlarını genişləndirən, Microsoft Office proqramlar paketi, Microsoft-un başqa proqramları və digər şirkətlərin proqram təminatı ilə birgə işləmək üçün təyin edilmiş, Visual Basic dilinin dialektidir [13, 14].

Office-də proqramlaşdırarkən, tam olaraq, VBA dilindən imtina etmək də olar. Bunun üçün istənilən Component Object Model (COM) proqram təminatı kateqoriyasına uyğun gələn dil tətbiq edilə bilər, məsələn, adi Visual Basic, VBScript, JScript, C++, Delphi, Java v.s. Hətta .NET kateqoriyalı proqramlaşdırma dilləri tətbiq edilə bilər - VB.NET, C# v.s. Beləliklə, Office proqramları obyektlərinin bütün imkanlarından faydalanmaq mümkün olacaq. Məsələn, aşağıdakı kod *.vbs genişlənməsi ilə saxlanılsa və onun işləməyinə start verilərsə, onda birincisi MS Word işə

salınacaq, sonra dərhal onun içərisindən yeni sənəd açılacaq və həmin sənəddə “VBScript-dən salamlar!” mətni çap ediləcək:

```
Dim oWord
Set oWord=CreateObject("Word.Application")
oWord.Visible=true
oWord.Documents.Add
oWord.Selection.TypeText("VBScript-dən salamlar!")
```

Məhz yalnız VBA bütün hallarda Office proqramları ilə işləmək üçün ən rahat və ən əlverişli proqram təminatıdır. Əsas səbəb çox sadədir – VBA dili Office proqramlarının tərkibinə Microsoft şirkəti tərəfindən əvvəldən quraşdırılıb, VBA dilindəki kodu Office proqramları sənədlərinin birinin tərkibində yaxud həmin proqramın bütün sənədlərində saxlamaq olar – Word sənədlərində, Excel kitablarında, Access fayllarında, PowerPoint təqdimatlarında v.s. Əlbəttə, həmin kodu adı çəkilən sənədlərdən işə salmaq olar, çünki VBA kodunun işləmə mühiti (proqramçı slenqində - “quyruq”) proqramların tərkibinə quraşdırılıb.

Hal-hazırda VBA aşağıdakı tətbiqi proqamlara quraşdırılıb:

- MS Office-in bütün əsas proqramlarına - Word, Excel, Access, PowerPoint, Outlook, FrontPage, InfoPath;
- Microsoft-un digər proqramlarına, məsələn, MS Visio və MS Project;
- lisenziya ilə digər başqa (məşhur və ya məşhur olmayan) şirkətlərin 100-dən artıq proqram təminatına quraşdırılıb: məsələn, CorelDraw və CorelWordPerfect Office 2000, AutoCAD v.s.

VBA-nın başqa üstünlükləri də var: VBA – universal dildir. Onu mənimsəyən nəinki Office paketinin proqramlarında və yuxarıda adı çəkilən proqram təminatlarının bütün imkanlarına açar tapacaq, üstəlik həmin istifadəçi aşağıda sadalanan hər bir işi peşəkarcasına yerinə yetirməyə hazır olacaq:

- Visual Basic-in özündə mükəmməl tətbiqi proqramlar yaratmaq, çünki bu dillər həddindən artıq bir-birinə yaxındır;
- VBScript-in bütün imkanlarından istifadə etməyi bacarmaq (çünki o kəsilmiş, qısaldılmış VBA dilidir). Nəticə etibarlı ilə, Windows inzibatçılıq (administratorluq) skriptləri yaratmaq istifadəçinin imkanları daxilində olacaq, məsələn, Web-səhifə yaratmaq üçün (VBScript Internet Explorer-də) Active Server Pages (ASP) üçün Web-əlavələri yaratmaq, Data Transformation Services (DTS) paketlərində tətbiq etmək və MS SQL Server tapşırıqlarını yerinə yetirmək və digər, və digər başqa buna oxşar işlərdə.

VBA, ilkin olaraq, peşəkar proqramçılar üçün yox (hərçənd ki, ən çox elə onlar bu mühitdən istifadə edirlər), adi istifadəçi üçün və istifadəçi yönümlü proqram təminatı kimi yaradılmışdır. Buna görə VBA-da proqramların tərtibatı asandır və qısa zamana başa gəlir. Bundan əlavə Office-ə istifadəçinin işini daha da asanlaşdırmaq üçün çox güclü vasitələr quraşdırılmışdır: obyektlerle və sintaksislə bağlı köməkçi göstərişlər, makrorekorder v.s.

VBA-da tətbiqi proqram yaratdıqda istifadəçiyə, çox güman ki, xüsusi olaraq, proqramlaşdırma mühitinin tənzimlənməsini qurmaq istifadəçinin kompyuterində lazım kitabxanaların (*“library”*) olması haqqında narahatlıq keçirmək lazım olmayacaq – MS Office istənilən kompyuterdə yüklənmiş olur.

Buna baxmayaraq ki, adətən VBA proqramları istənilən qədər sürətlə yerinə yetirilmir, ancaq onlar kompyuterin resurslarından minimal səviyyədə istifadə edir və çox stabil işləyir, məsələn, terminalların serverlərində. VBA-da məhsuldarlıq və səmərəlilik parametrlərinə xüsusi tələblər yoxdur: kompyuter oyunları, drayverlər, server məhsulları kimi VBA tətbiqi proqramları istifadə edilmir. Təcrübəli, peşəkar proqramistlərin rəyinə görə VBA-nın səmərəlilik və məhsuldarlıq göstəriciləri – VBA problemləri deyil, əslində verilənlər bazalarına müraciət etdikdə orada yaranan problemləridir. Bununla belə, əgər, həqiqətən, VBA ilə problemlər yaranırsa (adətən istifadəçiyə mürəkkəb riyaziyyat lazım olduqda), onda çox sadə çıxış yolu var: C++ dilində vacib olan kodu yazıb ona adi DLL kitabxanasına yaxud (ADD-in) Word, Excel, Access üçün qurulmuş əlavə kimi müraciət etmək.

VBA proqramları standart halda kompilyasiya edilmir və buna görə proqrama dəyişiklik etmək olduqca asandır və rahatdır [3]. Başlanğıc (mənbə) kodlarını axtarmaq lazım gəlmir və proqramı yenidən kompilyasiya etmək lazım deyil.

Peşəkar-proqramçılar icması belə hesab edir ki, proqramlaşdırmanın öyrənilməsi (*“sıfırdan”* və *“Hello, World”* tipli proqramlardan peşəkar səviyyəli proqrama qədər) məhz Office-VBA bağlantısında başlamaq daha əlverişlidir (nəinki C++, Java yaxud Delphi kimi dillərin mühitində).

1.6 Macrorecorder-in tətbiqi: bir sətir belə kod yazmadan proqramın yaradılması haqqında

MS Word və Excel-də Macrorecorder: işə salınması, imkanları

Microsoft Office proqramların əksəriyyətində, Excel, Word, PowerPoint v.s., diqqətəlayiq vasitə quraşdırılıb, bu vasitə ilə istifadəçi, proqramlaşdırma haqqında hətta heç bir şey bilməsə də, proqram tərtib edə bilər [18]. Bu vasitənin adı **Macrorecorder**-dir.

Adından məlum olur ki, Macrorecorder – əslində makrosları yazmaq üçün bir vasitədir. Macros isə VBA-proqramının daha bir adlanmasıdır. Macrorecorder əslində həmin proqramın avtomatik yaradılma vasitəsidir [1, 4, 6, ..., 18].

Macrorecorder-in işləmə prinsipi daha çox maqnitofonun işləmə prinsipinə oxşayır: *“yazılma”* düyməsini basdıqda - VBA mühitində istifadəçinin yerinə yetirdiyi əməliyyatlar yazılmağa başlayır. Maqnitofonun *“canlandırma”* düyməsini basdıqda isə - VBA mühitində yazılma dayanır və istifadəçi əvvəl yerinə yetirdiyi əməliyyatları həmin ardıcılıqla yenidən canlandırma bilər.

Əlbəttə, macrorecorder yalnız ən sadə VBA-proqramlarının yazılmasına imkan verir. Hətta belə bir şəraitdə də maksimum fayda qazanmaq olar. Məsələn, qaynar klavişlərə elə sözlər və nitqdə sözlərin sintaktik əlaqəsini *“qoymaq”* olar ki, hansıları istifadəçi tez-tez yaddaşa daxil edir (insanın yaşı, vəzifəsi, şirkətin adı, məhsulların adı v.s. – bununla istifadəçi xeyli zamana qənaət etmiş olar. Təcrübə göstərir ki, adi istifadəçilərin əksəriyyəti Macrorecorder haqqında heç bir xəbəri olmur,

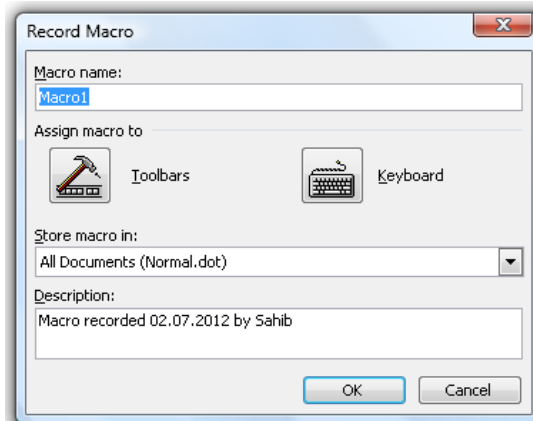
buna baxmayaraq ki, macrorecorder-in tətbiqi, yuxarıda göstərilən kimi, rutin işlərdə səmərəlilik yaradır və çox faydalı zamana qənaət edə bilər.

Macrorecorder-də Macros yaratdıqda aşağıdakı qaydalara riayət etmək lazımdır:

- yaradılması makrosu çox diqqətlə planlaşdırmaq lazımdır, əvvəldən hərtərəfli düşünərək, hansı iş yerinə yetiriləcək və hansı ardıcılıqda. Mümkün qədər hazırlıq işləri də qabaqcadan düşünülməlidir. Məsələn əgər cari tarix sənədin əvvəlində yerləşdiriləcəksə, onda, bəlkə, Makrosun birinci əmri kimi sənədin əvvəlinə keçirilmə əmrinin (<Ctrl>+<Home>) olması daha əlverişli olar;
- baxmaq lazımdır ki, makrosun hazırlanması olmadan, dərhal klavişə yaxud alətlər panelindəki düyməyə təyin ediləsi hazır əmr var mı (**Tools**→**Customize** meniyusu köməyilə). **Commands** qoşmasından istənilən əmri çəkib apararaq istənilən idarə etmə panelinə yerləşdirmək olar və həmin qoşmada **Keyboard** düyməsi basıldıqda – əmr üçün klavişlərin istənilən kombinasiyasını təyin etmək lazımdır;
- əgər makrosun köməyilə mətnin formalaşdırılması lazım olacaqsə, onda əvvəlcədən formalaşdırmada yeni stili yaratmaq lazımdır və sonra isə onu mətnə tətbiq edilməsi daha düzgün olar. Bu halda da makrossuz yararlanmaq olar, sadəcə klavişlərin kombinasiyasını təyin etməklə kifayətdir.

Macrorecorder-də makrosun yaradılması qaydası (MS Office paketində bunun üçün təyin edilmiş vasitə olan proqramlar üçün - Word, Excel, PowerPoint, Project):

Diqqət! Microsoft Office 2003 (və daha son versiyalarda) paketindəki proqramlarda standart hal elə qurulmuşdur ki, makrosları işə salmaq olmur. Bunun üçün makrosları yaratmaqdan əvvəl gerek istifadəçi (Office-in versiyasından asılı olmayaraq) macros təhlükəsizliyini “**Low**” (azərbaycanca aşağı) vəziyyətə keçirmək lazımdır. Office 2003 paketlərində **Tools**→**Macros**→**Security** meniyusundan istifadə edilir. Sonrakı Office versiyalarında həmin parametri **Office**→**Parameters**→**Main** meniyusundan **Developer** seçilməlidir ki, həmin quraşdırma Office proqramının meniyusunda əmələ gəlsin. Sonra əmələ gəlmiş **Developer** quraşdırmasında yenə də **Security** parametrini “**Low**” vəziyyətə keçirmək lazımdır. Bu əməliyyat yalnız bütün Office paketlərindəki proqramlarda yalnız bir dəfə dəyişdirilir. Yalnız bu halda istifadəçi VBA redaktorunu çağırıb öz makroslarını tərtib etməyə başlaya bilər. **Tools**→**Macros** meniyusunda **Record New Macros** əmri seçilməli. Açılan pəncərədə (şək.1.1) istifadəçi aşağıdakıları təyin etməlidir:



Şəkil 1.1 Macrorecorder-in yaradılması üçün dialoq pəncərəsi

- Makrosun adı (Macro name). Burada əvvəlcədən bu qaydaları bilmək vacibdir: Macros adı ədədlərdən başlanmamalı, boşluqlar olmamalı, punktuasiya işarələri olmamalı, yalnız aşağı haşiyələmə işarəsinə “_” ixtiyar var (məsələn, belə: **Fizikadan_Laboratoriya_proqram_1**), hətta azərbaycanca məna daşıyan sözləri ingilis dili rejimində yazılması məqsədə uyğundur, Excel-də adın maksimal uzunluğu 64 simvol, Word-də 80 simvoldan artıq ola bilməz;
- Makrosun çağırılması seçimi (**Assign macro to**) menyudakı düyməyə təyin ola bilər (Toolbars düyməsi) yaxud da klaviaturanın klaviş kombinasiyasına. Həmin seçimi sonradan da etmək olar – bu haqqında sonrakı fəsildə danışılacaq;
- Macros saxlanma yerinin seçimi (**Store macro in**). Word-də bütün yeni yaradılan sənədləri istifadəçi cari fayl və şablonu ilə seçə bilər – **normal.dot** şablonu ilə. Excel-də isə istifadəçinin sərəncamında cari Excel kitabıdır, makrosu yaratdıqda yeni Excel kitabının yaradılması imkanı və **personal.xls** şablonu ilə makrosların şəxsi kitabı (bu gizli kitabdakı makroslar bütün kitablarda əlçatan halda olacaq). Proqram kodunun harada saxlanması haqqında daha ətraflı VBA layihələrinin strukturu haqqında olan hissədə danışılacaq;
- Proqram haqqında məlumat (**Description**). Bu seçim həmin an bir o qədər də vacib olmasa da, lakin, hər halda, doldurulması vacibdir (istifadəçinin özü üçün bir neçə ay və ya il keçdikdə həmin məlumat faydalı ola bilər).

OK düyməsi basıldıqda və ya düymə/klaviatura kombinasiyası təyin edilsə makrosun yazılması avtomatik rejimdə başlanır. Mausun görkəmi həqiqətən maqnitofon kaseti formasını alır və kiçik görüntülü panel əmələ gəlir: **Stop recording** – yazını dayandırmaq və **Pause recording** – yazıya tənəffüs vermək. Təsadüfən bu paneli istifadəçi bağlasa, onda **Tools**→**Macro**→**Stop recording** menyusu ilə yazını dayandırmaq olar. Düymə/klaviatura kombinasiyası təyin edilməyən makrosun işə salınmasının ən sadə üsulu budur: **Tools**→**Macro**→**Macroses** menyusu (yaxud <Alt>+<F8> basıldıqda) seçildikdə əmələ gələn dialoq pəncərəsindəki siyahıdan lazım olan makrosu seçmək və **Run** (işlətmək) düyməsini basaraq proqramı işə salmaq. Həmin pəncərədən proqrama baxmaq, redaktə etmək, silmək, yeni makrosu yaratmaq və makrosların yerlərini siyahıda dəyişmək olar.

Əgər yaradılmış makrosların sayı çoxdursa, onda klavişlərin bütün təyinatlarının siyahısını almaq üçün (hətta Word-ün quraşdırılmış makroslarının da qatqısı ilə) **Tools**→**Macro**→**Macroses** menyusunda yaranan həmin dialoq pəncərəsində **Macros in** seçimində **Word comand** seçmək və **Macro name** seçimində **List Commands** seçilsə, onda **Step info** düyməsi basılmalıdır və bununla iki seçim üçün panel əmələ gələcək: **Current menu and keyboard settings** və **All word commands**. Birincisi seçilsə cəmi 10 səhifəlik məlumat olan yeni Word sənədi yaradılacaq, ikincisində isə tam siyahılı 26 səhifəlik məlumat olan yeni Word sənədi yaradılacaq.

Əgər istifadəçi tərəfindən xeyli sayda Macrorecorder-lə yaradılmış Macros varsa, onda VBA dilini mənimsəyəndən sonra həmin avtomatik rejimdə yaradılan Macros-larda lazımı təkmilləşdirmə aparmaq olar. Bu səpkidə aşağıdakı işləri yerinə yetirmək məqsədə uyğundur:

- istifadəçi makrosunda hansısa əməliyyatlar təkrarlanırsa, onda dövr hesablamasının təşkili zəruridir;
- macrosun işləməsi zamanı müəyyən məlumatı **InputBox** yaxud idarəetmə elementlərindən istifadə edərək alınmasını təşkil etmək;
- makrosun işlədiyi zamanda səhvlər əmələ gəlməsin deyə cari şərtlərin yoxlanmasını təşkil etmək.

Sonrakı fəsillərdə bu haqda daha mükəmməl danışılacaq.

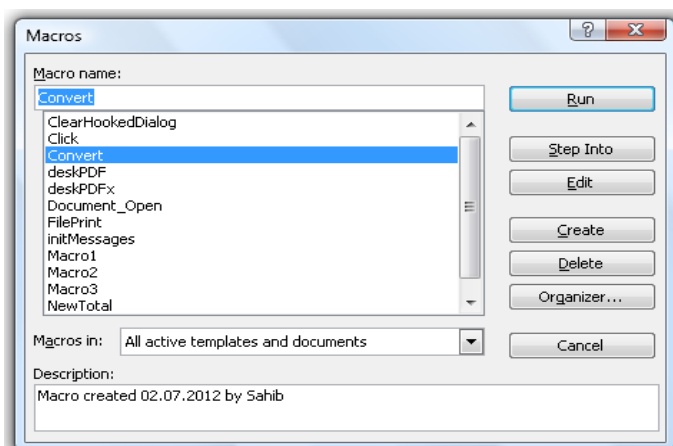
Macrorecorder-lə əlaqəli bir vacib iş də var. Heç bir xüsusi müəllif tərtibatı olmadan, şəxsi istifadə üçün ən sadə proqramların yaradılmasından əlavə olaraq, bu rejimdə peşəkar istifadəçilər Office proqramlarının obyekt modellərini ətraflı tədqiq edirlər. Bacarıqlı istifadəçilər bununla Office-dəki obyekt modellərinin hansı qurulacaq əməliyyatlarda daha əlverişli istifadəsini öyrənirlər. Konkret hal bu cür yarana bilər: Excel-də diaqramların yaradılmasını avtomatlaşdırmaq lazımdır. Adı qaydada **Insert**→**Diagramm** menyusundan istifadə edilir. VBA sorğu materiallarında **Diagram object** adı ilə məlumat tapmaq olar, lakin bu məlumat axtarılan mövzuya aid deyil, çünki əslində **Chart object** axtarılmalı idi. Peşəkar istifadəçi isə Macrorecorder-lə yazaraq diaqramı yaradır və sonra yaradılmış koda baxıb hərəkətin doğru yolunu asanlıqla tapacaq.

1.7 Makrosun dialog pəncərəsi, alətlər paneli, əmrlər sətiri: macros yaradıldıqdan sonra işə salınma qaydaları.

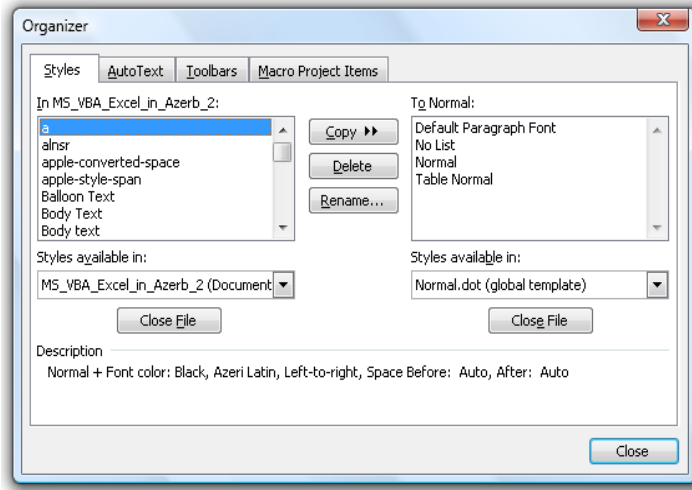
Office proqramlarında macros menyusu və düymələr, makrosların işə salınmasının müxtəlif üsulları

Tutaq ki, macros artıq yaradılıb (Macrorecorder-də yaxud VBA redaktorunda - sonuncunun öyrənilməsi növbəti hissələrdə olacaq) və istifadəçi işə həmin proqramı ya bir dəfə işə salmaq və ya onu daimi olaraq işə salınması imkanını tənzimləmək istəyir. Belə halda istifadəçinin sərəncamında bir neçə üsul vardır [12].

Ən sadə (o biri tərəfdən də narahat) üsul budur - əvvəlcə **Tools**→**Macro**→**Macros** menyusu ilə Macros dialog pəncərəsi gətirilir (şək. 1.2).



Şəkil 1.2 **Macros** dialog pəncərəsi



Şəkil 1.3 **Organizer** dialog pəncərəsi

Bu pəncərədə aşağıdakı işləri yerinə yetirmək olar (Cədvəl 1.1):

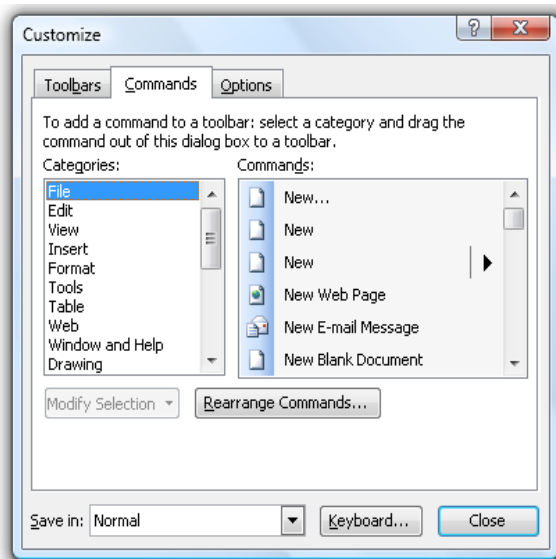
Cəd. 1.1 Macros dialog pəncərəsindəki əməliyyat düymələrinin izahı

Pəncərənin əməliyyat düymələri	Azərbaycanca əməliyyatın izahı
Run	Macrosun işə salınması
Step Info	VBA redaktorunun açılması və addım-addım yerinə yetirilməsi
Edit	VBA redaktorunda proqramın dəyişdirilməsi
Create	Yeni yaradılan makrosun adını vermək lazımdır və VBA redaktorunda, avtomatik olaraq, həmin adla redaktə etmək üçün yeni prosedura əmələ gələcək.
Delete	Makrosun silinməsi.
Organizer	Makrosun parametrlərini, yazı stilindən başlayaraq isarəetmə panellərinə qədər, dəyişmək imkanını yaradır (Organizer dialog pəncərəsi, şəkl.1.3).
Cancel	Bütün əməliyyatlardan imtina və Macros dialog pəncərəsinin bağlanması.

Buna görə hər dəfə bu pəncərənin açılması, lazımı makrosun tapılması (makrosların sayı isə çox güman ki, 30-dan yuxarı da ola bilər) və, seçilərək, **Run** düyməsinin basılması ilə işə salınması, əlbəttə, ən tez və əlverişli üsullardan sayıla bilməz. Daha rahat və əlverişli üsul (əgər makros daimi iş görülürsə onun daha tez çağırılması üçün) budur: - klaviatura kombinasiyasını təyin etmək. Məsələn, tutaq ki, hətta bu kitab yazıldıqda "**Microsoft Visual Basic Applications**" sözü çox sayda təkrarlanır. Onda macrorecorder ilə klaviatura kombinasiyasını əvvəldən, məsələn, **<Alt>+<V>** kimi seçilsə və dayandırmamış macrorecorderdə həmin söz yığılsa, onda artıq bu əməl üçün macros avtomatik rejimdə yazılmış olur. İndi istənilən vaxt həmin sənəddə (söhbət Word sənədindən gedir) oxşar əməliyyatları digər Office proqramlarında da icra etmək olar. Macrorecorder-ə daxil olmadan yalnız **<Alt>+<V>** klaviş kombinasiyası basıldıqda, yuxarıda hər hansı uzunluqda olan söz, avtomatik olaraq, istifadəçi istədiyi sayda mətn sətirində çap olunacaq. Bu üsul çox əlverişli və rahatdır: oxucu öz misalında bu üsulun üstünlüyünü yoxlaya bilər. İş şəraitində klaviatura kombinasiyasından istifadə edilməsi (məsələn, müdir haqqında, təşkilatın tam adı haqqında olan məlumatın daxil edilməsində) təkrarlamalara rast gəldikdə çox faydalı ola bilər.

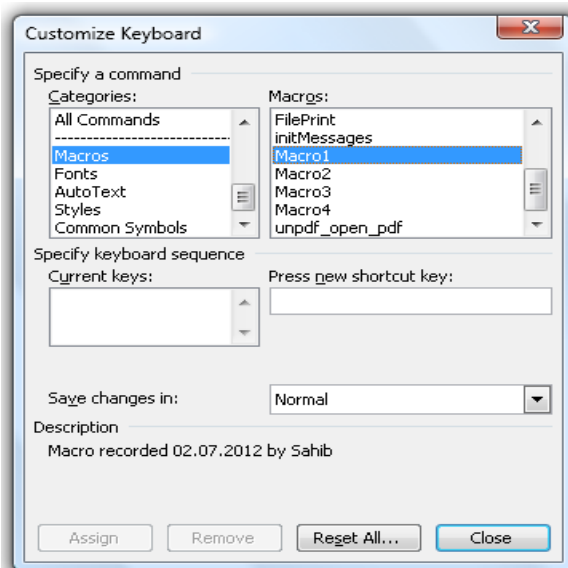
Əsas budur ki, həmin makrosu istifadəçi həqiqətən tez-tez işə salsın, çünki digər halda edilən əməliyyat müəyyən vaxtdan sonra istifadəçinin özünün yadından çıxa bilər.

Makrosu klaviatura kombinasiyasını çox asan yolla (məsələn, Word-də) bu cür vermək olar: **Tools** menyusundan **Customize** seçilir və **Commands** quraşdırmasına keçmək lazımdır (şək. 1.4). Yaranmış pəncərədə **Keyboard** düyməsi basılan kimi dərhal Şəkil 1.5-dəki pəncərə əmələ gəlir.



Şəkil 1.4 Word-ün **Customize** dialog pəncərəsi.

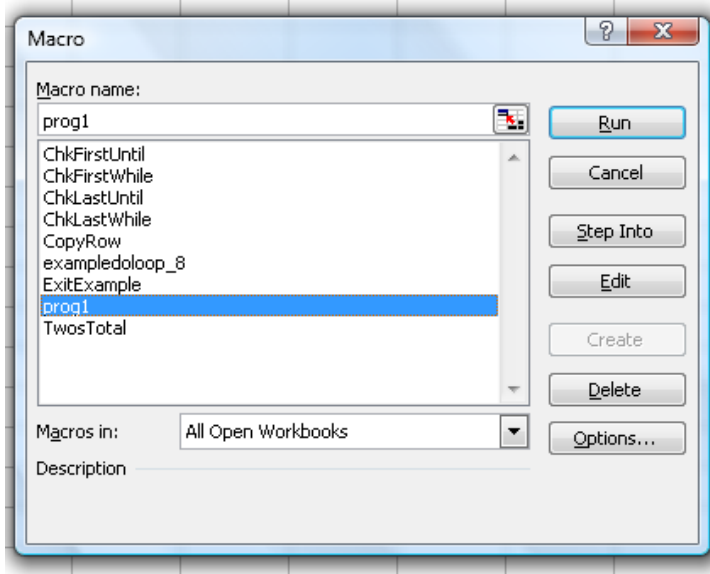
Şəkil 1.5-dəki pəncərədə makroslar üçün klaviatura kombinasiyasını seçmək fərdi istəkdən asılı ola bilər. Burada **Assign** düyməsi klaviatura kombinasiyasını təyin edir. **Remove** düyməsi həmin təyinatı yaxud digərini silə bilər. **Reset All** düyməsi isə bütün təyinatların yenidən təyin edilməsinə səbəb ola bilər.



Şəkil 1.5 Word-də makrosların klaviatura kombinasiyasının seçilməsini təyin edən **Customize Keyboard** dialog pəncərəsi.

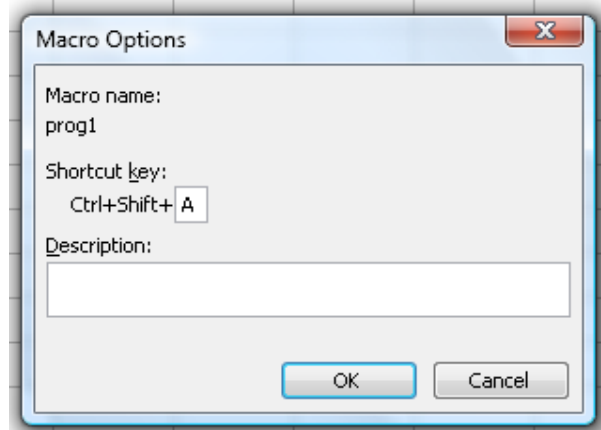
Diqqət! **Current keys** seçimində ehtiyatlı olmaq lazımdır, çünki əgər istifadəçi təyin etdiyi klaviatura kombinasiyasına başqa makrosda varsa, onda xoşa gəlməz vəziyyətlər yarana bilər.

Excel-də isə bir az başqa əməliyyatlar ardıcılığı lazımdır: **Tools**→**Macro**→**Macros**. Dərhal Şəkil 1.6-dakı dialoq pəncərəsi əmələ gələcək



Şəkil 1.6 Makrosların dialoq pəncərəsi

Lazım olan makrosu bu pəncərədən seçib, **Options** düyməsi basılmalıdır. Nəticədə dərhal şəkil 1.7-dəki dialoq pəncərəsi əmələ gələcək. Məsələn, həmin pəncərədə **prog1** makrosuna **<Ctrl>+<Shift>+<A>** klaviatura kombinasiyası təyin edilmişdir. **Discriptions** (İzahat) hissəsində istifadəçi öz ifadələrini klaviatura kombinasiyasının təyin edilməsi haqqında öz qısa ifadəsini ingilis dilində (və ya ingilis hərfləri ilə başqa dildə) daxil edə bilər.



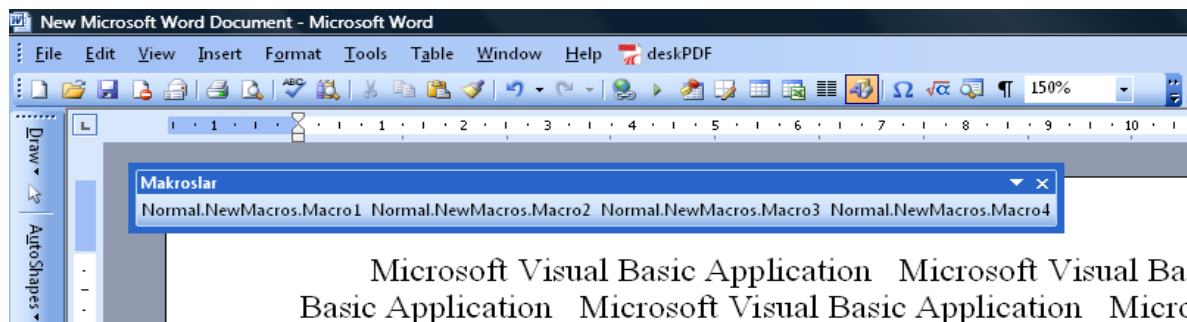
Şəkil 1.7 Excel makroslarının işə salınması üçün klaviatura kombinasiyasını təyin edən **Macro Options** dialoq pəncərəsi

Praktiki işlərdə Excel makrosuna istənilən klaviatura kombinasiyasını təyin etmək olar, lakin sadə üsulla bunu etmək olmaz - həmin tətbiqi proqramdakı hadisələri idarə edən proqram kodunun yazılması lazım olacaq.

Yuxarıda dediyimiz kimi klaviatura kombinasiyası üçün yalnız o makrosları təyin etmək lazımdır ki, ondan hər gün istifadə edilir. Bəs əgər makrosla yalnız ayda bir dəfə, hesabat dövründə, istifadə edirlərsə? Əksər istifadəçilər həmin dövrdə bütün təyin edilən klaviatura kombinasiyalarını yaddan çıxarırlar və, ola bilsin, hətta həmin məlumatı yazdıqları kağız parçalarını da itirmiş olurlar. Hətta makrosu tərtib edən peşəkar proqramçının da özü yaratdığı proqramların işə salınması üçün hansı

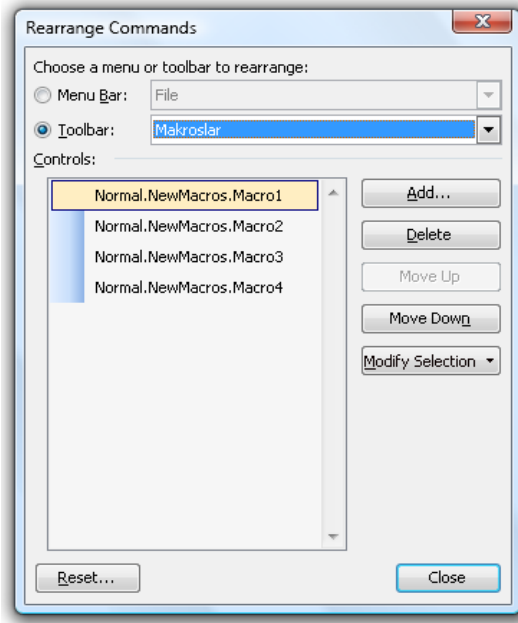
düymələrin basılmasını yaddan çıxarmağı təbii şey ola bilər. Ən yaxşı çıxış yolu – menyu punktuna təyinat verməkdir. Yeni orada anlanmağı asan olan adlanmalarla makrosları nizamlamaq lazım gələcək. Çünki makrosu işə salmaq üçün alətlər panelində düyməni basmaq daha tez zamana başa gəlir və daha rahatdır. Word-də makrosu çağırmaq üçün yeni alətlər panelinin yaradılması və tənzimlənməsi bu cür həyata keçirilə bilər:

- **Tools** menyusunda **Customize** seçərək **Toolbars** guraşdırmasına keçmək lazımdır;
- **New** düyməsi basılmalı və əmələ gələn **New Toolbar** pəncərəsində yeni yaradılacaq panelin adı təyin edilir (məsələn, tutaq ki, **Makros** adını seçmək olar) və paneli yaradılan sənədi də həmin pəncərədən seçməli. Əgər **normal.dot** şablonu seçilsə, onda yaradılan menyu bütün sənədlər üçün əlçatan vəziyyətdə olacaq (tez-tez bu seçim daha lazımlı olur). Digər halda yaradılan panel yalnız həmin sənəddə əlçatan vəziyyətdə olacaq;
- **OK** düyməsi basıldıqda isə yeni boş menyu yaranmış olacaq (sənədin üst hissəsində). Daha əlverişli rahat istifadə üçün onu standart alətlər panelinə çəkib atmaq lazımdır.
- Həmin **Tools**→**Customize** menyusundan **Comands** seçilir və, kateqoriyalarda **Macros** seçilərək, **Comands**-dan seçilən makrosları sadəcə çəkib alətlər panelinə atmaq lazımdır;
- Əgər alətlər panelində birdən artıq makros yerləşdirmək lazımdırsa, onda həmin pəncərədə **Rearrange Comands** düyməsi basılmalıdır. Nəticədə aşağıda Şəkil 1.9-dəki pəncərə alınacaq və həmin pəncərədə nizamlamanı aparmaq olduqca asan olacaq.

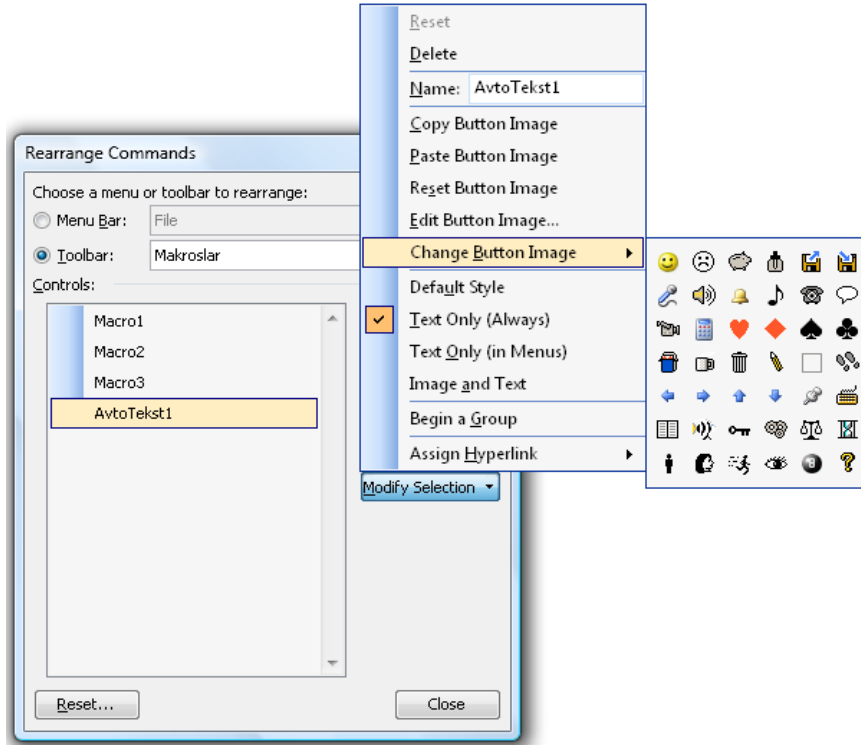


Şəkil 1.8 **Makroslar** adı verilmiş yeni alətlər paneli (onu çəkərək, standart panel və ya onlardan aşağı (yuxarı) yerləşdirmək olar.

Tutaq ki, alətlər panelində lazım olan düymələri əlavə etdik, lakin avtomatik rejimdə o bir qədər əlverişli görünmür (məsələn, adları: Normal.NewMacros.Macro4 v.s.). Bunun üçün **Customize** menyusunda mausun sağ düyməsini alətlər panelində şıqqıldadıb (**Customize** pəncərəsi hökmən açıq olmalıdır!) dərhal xüsusi bir kontekst menyusunun açılmasını görürük (Şəkil 1.10).



Şəkil 1.9 **Rearrange Comands** makrosları alətlər panelində nizamlama imkanını yaradan dialog pəncərəsi.



Şəkil 1.10 **Rearrange Comands** dialog pəncərəsi ilə makrosları alətlər panelində nizamlama imkanını yaradan (şəkil 1.9-dan fərqli olaraq, artıq makrosların adları dəyişdirilmişdir, məsələn, Macro4 olub AvtoTekst1).

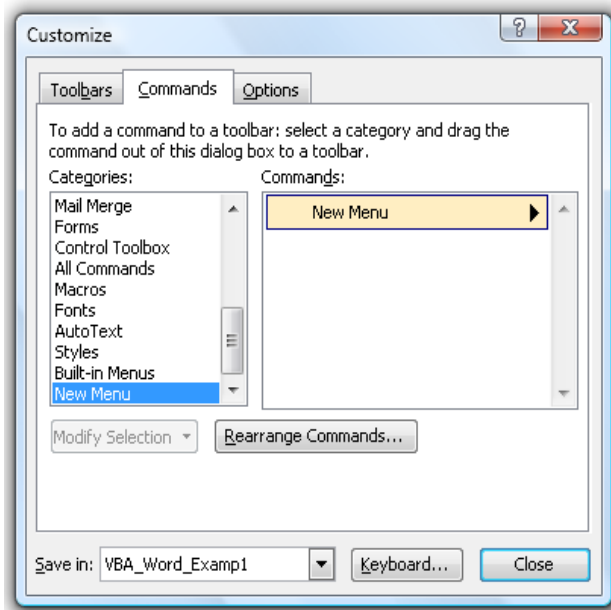
Bu açılmış kontekst menyusu ilə aşağıdakı əməllərin yeritilməsi mümkündür:

- **Reset** – yenidən dəyişikləri standart əvvəlki vəziyyətə qaytarmaq;
- **Delete** – sadəcə seçilmiş düyməni silmək (başqa yol ilə həmin seçilmiş düyməni silmək üçün **Customize** pəncərəsinə çəkib-atmaq kifayətdir);
- **Name** – düymədə adın dəyişdirilməsi;

- **Copy Button İmmage** və **Paste Button İmmage** – başqa düymənin görüntüsü xoşa gəldikdə ondan seçilmiş düymə üçün istifadə etmə imkanını yaradırlar;
- **Edit Button İmmage** – sadə bir qrafik redaktor açılacaq və həmin redaktorda istifadəçi öz xoşuna gəldiyi görüntünü tərtib edə biləcək;
- **Change Button İmmage** – 42 sayda olan görüntülərdən birini düyməni işarə etmək üçün imkan yaradır (əslində yalnız Word-də 1000-dən artıq nişan görüntüsü vardır, bu haqda kitabın alətlər panelinin programlaşdırma ilə yaradılmasına həsr edilmiş fəslində daha ətraflı danışılacaq);
- **Default Style** – standart stil (bu halda düymədə yalnız şəkil əmələ gəlir və heç bir yazı olmur);
- **Text Only (Always), Text Only (in Menus)** və **Image and Text** – bu rejimlərdə istifadəçi düymənin hansı formada olmasını təyin edir: yazı/şəkil);
- **Begin a Group** – bu imkan haqqında kitabın alətlər panelinin programlaşdırma ilə yaradılmasına həsr edilmiş hissəsində daha ətraflı danışılacaq;
- **Assign Hyperlink** – düymə ilə hansısa tətbiqi proqram ilə hiper əlaqəni yaratma imkanını verir.

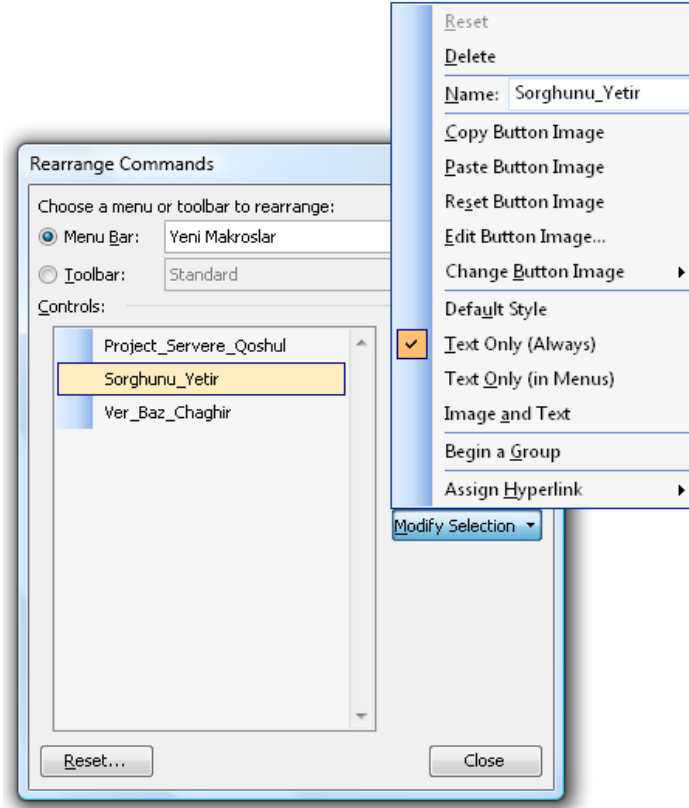
Menyunun yaradılması isə başqa tərzdə həyata keçirilir:

- **Tools**→**Customize** menyusu istiqamətində **Customize** pəncərəsi açılmalıdır;
- **Comands** quraşdırmasında **New Menu** seçilməlidir;
- **New Menu** əmrini siyahıdan mausla çəkib-atmaqla baş menyunun istənilən yerinə qoymaq lazımdır (şək. 1.11);



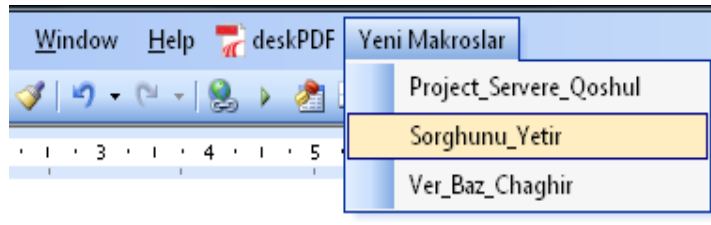
Şəkil 1.11 **Customize** dialog pəncərəsində makroslar üçün yeni menyunun (**New Menu**) yaradılması.

- sonraki addımda açıq **Customize** pəncərəsi vəziyyətində yaradılan menyü kontekstində, mausun sağ düyməsini şıqqıldadaraq, onun adını istifadəçinin istəyilə dəyişdirmək olar (məsələn, **Makroslar** kimi);
- növbəti addımda **Rearrange Comands** dialoq pəncərəsi ilə istifadəçi əvvəlcədən adını dəyişdirdiyi **Yeni Makroslar** adlı menyuda tərkibindəki makrosların **Modify Selection** əmri ilə (şək.1.12) yerləşdirilməsi, adlarının dəyişdirilməsi v.s. əməllərini yerinə yetirə bilər.



Şəkil 1.12 **Rearrange Comands** dialoq pəncərəsi ilə istifadəçi tərəfindən Yeni Makroslar adı ilə adlandırılan menyunun tərkibindəki makrosların yerləşdirilməsi, adlarının dəyişdirilməsi v.s. həyata keçirilməsi.

Nəticədə istifadəçi özü üçün yaratdığı makrosların idarə edilməsindən ötrü çox nəfis, rahat və real iş prosesini yaratmaq məqsədi ilə olduqca əlverişli şəxsi menyusunu yaratmış olacaq (şək. 1.13).

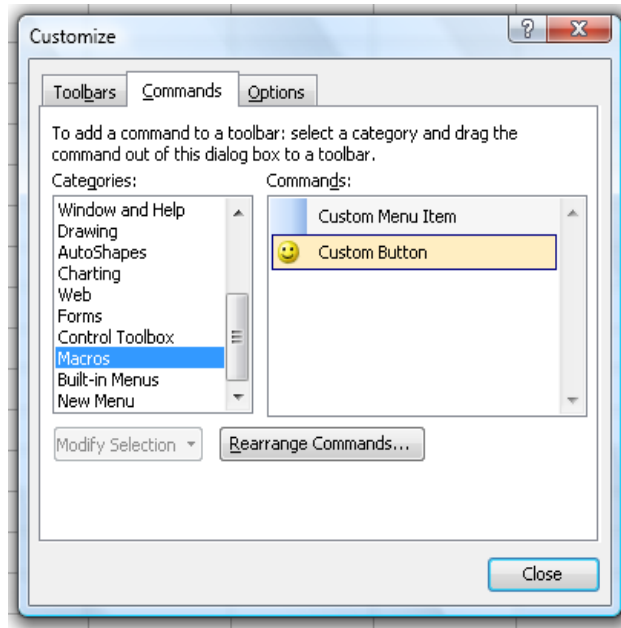


Şəkil 1.13 İstifadəçi tərəfindən yaradılmış makrosların idarə edilməsi üçün şəxsi menyunun görüntüsü.

Excel-də həmin nəticənin alınmasında bir çox oxşarlıq olsa da, hər halda, bir az fərq var:

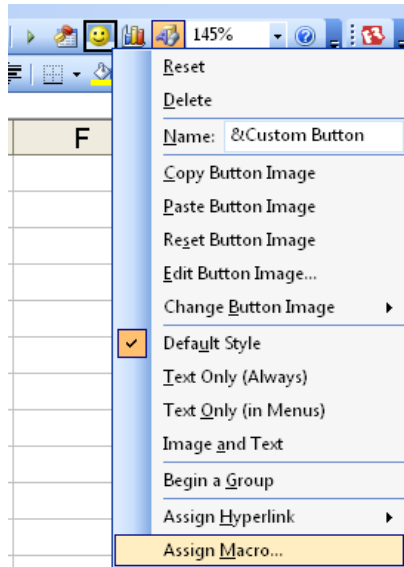
- Excel-də **Customize** dialoq pəncərəsi açılmalıdır (**Tools**→**Customize**);
- həmin pəncərənin **Categories** siyahısında **Macros** seçilməlidir;

- pəncərənin sağ tərəfindəki **Comands** siyahısında makroslar siyahısı əvəzinə iki seçim görsənəcək: **Castom Menu İtem** və **Custom Button** (şək. 1.14).



Şəkil 1.14 Excel-də makrosu işə salmaq üçün düymənin yaradılması.

- **Castom Button** – hazır vəziyyətdə olan düymədir, onu istifadəçi alətlər panelinin istədiyi yerinə çəkib ata bilər;
- həmin düymə üçün mausun sağ düyməsilə kontekst menyusu açılmalıdır (əlbəttə, Excel-də və Word-də düymənin şəkilciyini seçmək üçün digər üsullar da vardır, üstəlik şəkilciyin formasında istifadəçi öz istəyi ilə dəyişdirə bilər) və oradan **Assign Macros** (makrosun həmin düyməyə təyin edilməsi) əmri seçilməlidir (şək. 1.15).



Şəkil 1.15 Excel-də makrosu işə salmaq üçün düymənin yaradılması.

Excel-də yeni menyunun yaradılması eynilə, yuxarıda göstərilən, Word proseduralarından fərqlənir. MS Office-in digər proqramlarında (Power Point, Project Outlook v.s.) makrosların qrafik interfeysdən işə salınmasının qurulması Word-dəkindən fərqlənir. Bununla belə daha

universal (və daha çox zəhmət tələb edən) üsul da var: xüsusi qrafik formanı VBA-da yaradaraq, orada bütün makrosların siyahısının menyusunu yaratmaq. Həmin üsul üçün bir çox mürəkkəb idarəetmə funksiyalarını da əlavə etmək olar. Bu halda VBA-da xüsusi proqram kodu istifadəçi tərəfindən yazılmalıdır və formanın koduna əlavə edilməlidir. Bu kitabın 5-ci fəslində VBA formaları və idarəetmə elementləri ilə işləmək üçün daha ətraflı məlumatlar veriləcək. Həmin fəslin nəzəri məlumatlarını və 15-ci fəsildəki praktiki tapşırıqları oxucu mənimsəməyi bacarsa, onda oxşar formaların və hətta onlardan da mürəkkəb olan qrafik interfeyslərin yaradılmasını asanlıqla tərtib etməyi bacaracaq.

Burada qeyd edilməlidir ki, makrosun işə salınmasının daha bir vacib üsulu da var: makrosların müəyyən hadisənin baş verdiyi zamanı işə salınması və ya dayandırılması. Belə hadisələrdən, məsələn, Excel ya Word sənədlərinin sənəflərində hansısa dəyişikliyin yaratdığı hadisəni misal gətirmək olar. Kitabın müvafiq hissələrində bu haqda daha mükəmməl məlumatlar veriləcək.

Nəhayət, başqa bir mühüm üsul budur ki, proqramlaşdırmasız da makrosun avtomatik işə salınmasını təşkil etmək olar: sadəcə makrosa VBA-da təyin olmuş xüsusi ad vermək lazımdır. Word üçün həmin adların siyahısı aşağıdakı Cədv. 1.2-də verilib.

Cədvəl 1.1 Macrosların VBA-da təyin edilmiş xüsusi adları.

PROSEDURANIN ADI	NƏ VAXT İŞƏ SALINIR
AutoExec	Word işə salındıqda (bu makros <code>normal.dot</code> şablonu ilə yaddaşda saxlanmalıdır)
AutoNew	Yeni sənəd yaradıldıqda
AutoOpen	İstənilən sənədin açıldığında (əgər <code>normal.dot</code> şablonu ilə yaddaşda saxlanmışdırsa) həmin sənəddə AutoOpen adı ilə mövcud olan makros varsa.
AutoClose	Sənəd bağlandıqda.
AutoExit	Word-dən çıxdıqda

Excel-də işçi kitabı üçün xüsusi adlar nəzərdə tutulub: **Auto_Open**, **Auto_Close**, **Auto_Activate** və **Auto_Deactivate**. Microsoft isə xəbərdarlıq edir ki, bu imkanlar yalnız əks əlaqələndirmə üçündür və hadisə baş verəcək proseduralarda istifadə edilməsi məsləhət görülür. **Auto** makrosları ilə bağlı başqa bir vacib moment: əvvəllər viruslar həmin imkandan geniş istifadə edirdilər deyə (Office 2003-dən başlayaraq) standart vəziyyətdə belə makroslar işə salınmır. Bunun üçün gerek VBA-da təyin edilmiş təhlükəsizlik səviyyəsi ən aşağı vəziyyətə keçirilsin (**Tools**→**Macros**→**Security**).

Nəhayət, sonuncu məsləhət budur: makrosları əmr sətrindən, Word və ya Excel işə salındıqda, makrosun adı gerek əmr sətrinin parametri kimi göstərilsin. Məsələn, Word-ün

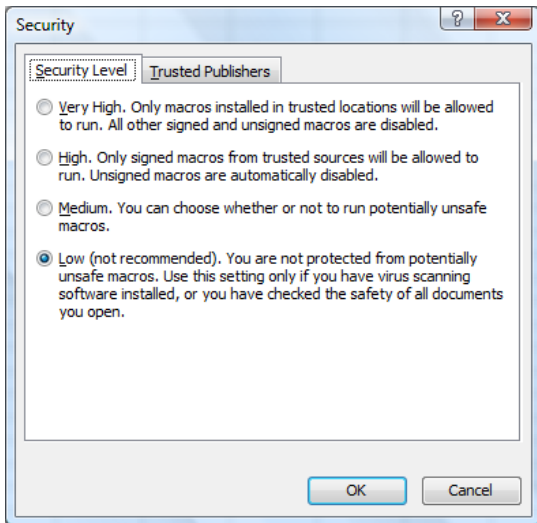
açılması anında tutaq ki, MyMacros adlı makrosun (hökmən **normal.dot**-dan) işə salınsın deyə **winword.exe/mMyMacros** əmrindən istifadə etmək lazımdır. Bu imkandan daha əlverişli belə də istifadə etmək olar:

- Office proqramlarının işə salınma yarıqlarını, məsələn, **Desktop**-da bir neçəsini yaratmaq;
- onlarda əmr sətirini dəyişmək;
- Office proqramını işə salmaq üçün eyni zamanda, avtomatik olaraq, makroslar işə salınacaq.

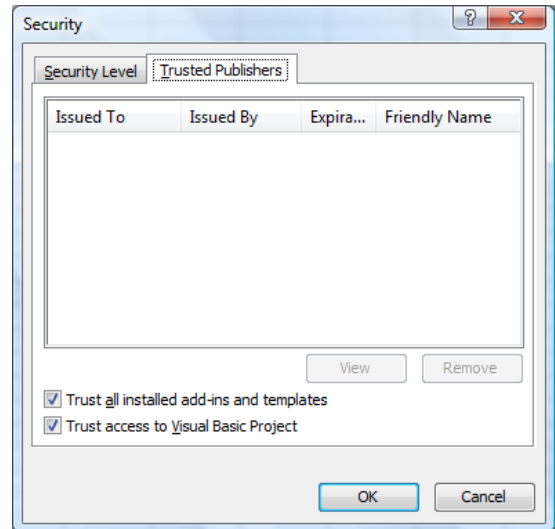
2 VBA REDAKTORU İLƏ TANIŞLIQ

2.1 VBA redaktorunda işləməkdən öncə təhlükəsizlik parametrlərinin seçilməsi. VBA-nın qrafik interfeysi ilə ümumi tanışlıq.

Hər şeydən əvvəl: MS Office proqramındakı Microsoft Visual Basic for Applications (məsələn, MS Excel proqramında) redaktoru ilə işləmədən yaxud hansısa VBA makrosunu işlətməkdən əvvəl gərək Office proqramının **Tools**→**Macro** menyusunda **Security** təlimatı seçilməlidir və təhlükəsizliyi (**Security**) təyin etmək lazımdır. Əgər kompyuterdə antivirus proqramı yüklənmişsə, onda tam rahat işləmək üçün aşağıdakı seçimi etmək lazımdır:



Şəkil 2.1 MS VBA Excel sisteminin **Security** diaqoq pəncərəsində təhlükəsizlik səviyyəsinin seçilməsi.



Şəkil 2.2 MS VBA Excel sisteminin **Security** diaqoq pəncərəsində makros müəllifinə etibar səviyyəsinin seçilməsi.

Digər halda, antivirus proqramının olmadığı, yaxud onun güclü olmadığı halda, onda Microsoft kompaniyası məsləhət görür ki şək. 2.1-də **Medium** səviyyəsi seçilsin və şək. 2.2-dəki kimi seçim dəyişməz qalsın. Hətta daha çox təhlükəsizliyi təmin etmək üçün, Microsoft işləmədən sonra şək. 2.1-də **Very High** yaxud **High** seçimini etməyi məsləhət görür.

Obyektistifadəli proqramlaşdırma mühitləri (OİP) demək olar ki, bütün "köhnə" universal proqramlaşdırma dilləri üçün yaradılmışdı və dilin adının əvvəlinə **Visual** və ya **Object** əlavə artırılmaqla adlandırılırlar (**VisualFoxPro**, **Visual dBase**, **Visual C++**, **Visual Fortran** və s.). Bu mühitlərdən ən çox istifadə olunanları **Microsoft Visual Basic** və **Microsoft Visual Basic for Applications**, yəni **VBA**-dır. Proqramlaşdırma dili kimi, müvafiq olaraq, **Visual Basic (VB)** dillindən istifadə olunur. Mühitlərin özləri kimi bu proqramlaşdırma dilləri arasında (yəni VB və VBA ilə) bir-birinə uyğunluq çoxdur və bunların birində işi mənimsəməklə çox asanlıqla digərində də proqramlaşdırmaq xeyli asanlaşır [18].

Əlbəttə, 1.6 fəslində ətraflı ifadə edilmiş Macrorecorder (birinci baxımda) avtomatlaşdırılmış proqramlaşdırma mühiti kimi çox əlverişli və rahat vasitədir. Lakin real iş vəziyyətində bütün yarana biləcək hallar üçün bu üsul yararlı sayıla bilməz. Macrorecorder proqramlaşdırılmasının peşəkarcasına baxımından bir çox qüsurları vardır:


- əgər hansısa hesablanan kəmiyyətin qiymətindən gözlənilən bir hadisə baş verməlidirsə, onda həmin qiyməti istifadəçi yoxlaya bilmir;
- dövrü hesablama proseduralarının qurulması mövcud deyil;
- səhvlərin təyin edilib, aradan qaldırılması imkanları yoxdur;
- istifadəçi bu mühitdə məhdud sayda obyektlər siyahısından faydalana bilir (və bu siyahı bir o qədər əlverişli deyil, məsələn, Word-də mətni daxil etdikdə, etibarlı olan **Range** obyektinə əvəzinə yalnız istifadəçinin əməllərinə həssas olan **Selection** obyektinə vardır);
- macrorecorder-də yaradılmış makroslar funksionallıq baxımından çox məhduddur.

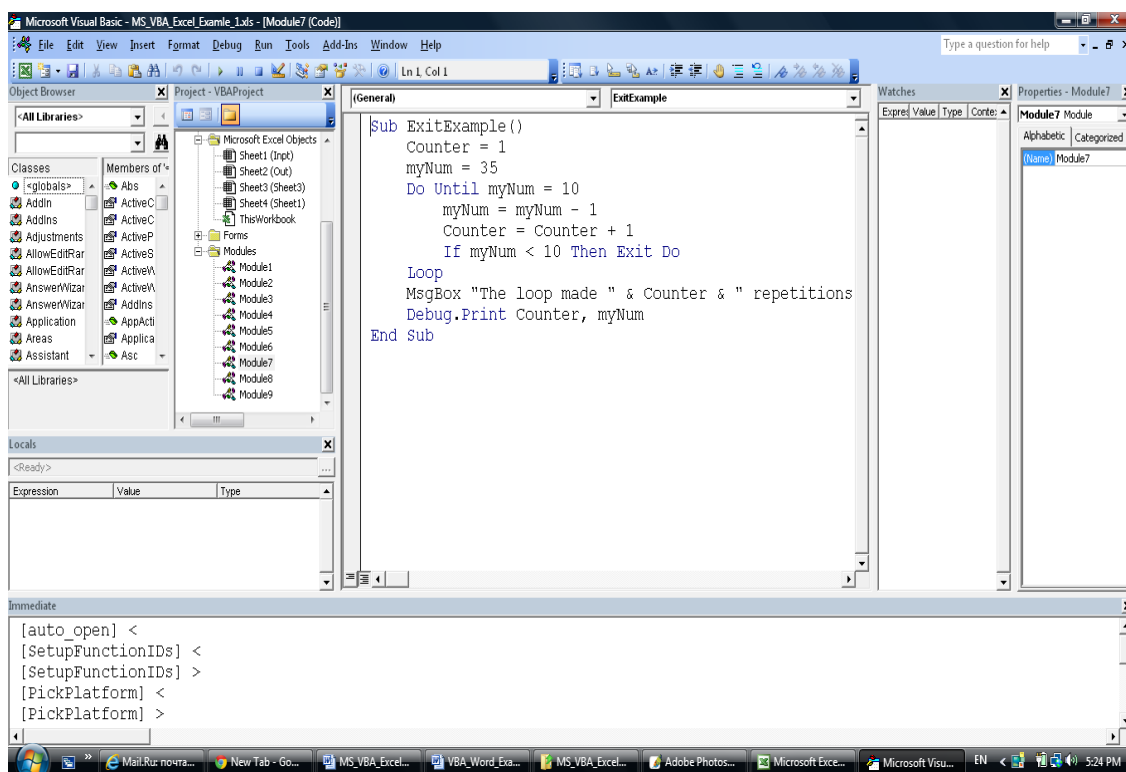
Office-də proqramlaşdırmanın tam imkanları yalnız Visual Basic for Applications (VBA) redaktorunun tətbiqi prosesində aşkar olur. Ciddi işlərdə VBA-sız işləmək mümkün deyil. Bununla belə, məlumdur ki, əksər hallarda orta səviyyəli MS Office proqramlar paketinin istifadəçiləri üçün həmin proqramlaşdırma mühitinin, müstəqil olaraq, bütün üsullarını anlaması çox çətin olur. Bunun üçün isə ilk addımdan istifadəçi VBA redaktorunun imkanlarını mükəmməl bilməlidir və onun dialog pəncərələrində hansı əməllərin aparılmasını yaxşı mənimsəməlidir.

Hal hazırkı fəsildə biz Microsoft Visual Basic for Applications proqramlaşdırma dilinin redaktoru ilə tanış olacağıq (bu redaktorun çoxsaylı dialog pəncərələrində hansı işlərin görülməsində mövcud olan imkanlar haqqında danışılacaq).

Diqqət! MS Office proqramlarında tərtib edilən tətbiqi proqramlarda istifadə edilən obyekt modelləri üçün VBA dili kifayətdir və Visual Basic (VB) redaktoru heç bir ehtiyac yoxdur. Əlbəttə, peşəkar istifadəçi istənilən Component Object Model (COM) proqram təminatı kateqoriyasına uyğun gələn dili tətbiq edilə bilər: məsələn, adi Visual Basic, VBScript, JScript, C++, C#, Delphi, Java, Visual Basic.NET v.s. Təkrarlayırıq ki, VBA dilinin və onun redaktorunun yaradılmasının əsas məqsədi məhz Microsoft Office proqramlarında hesablamaları avtomatlaşdırılması və MS Office obyekt modelləri ilə effektiv işləməkdən ibarət olub. Həqiqətən də bu proqramlaşdırma mühitində işləmək çox asan və rahatdır: sırada gələn fəsilər və hissələrdə oxucu əyani olaraq bunun şahidi olacaqdır.

VBA redaktoru ilə işləməyi başlamaqdan öncə onu MS Office əlavəsində açmaq lazımdır. Bütün Office proqramlarında bu eyni qaydada yerinə yetirilir, nəticədə VBA redaktorunun pəncərəsi açılır (şək. 2.3):

- ən asan üsulda **Tools** menyusunda **Macros**→**Visual Basic Editor** əmrləri seçilməli;
- ən qısa zamana başa gələn üsul isə klaviatürada **<Alt>+<F11>** düymələrinin basılmasıdır;
- başqa üsulda Office əlavəsinin alətlər panelində  **Visual Basic** düyməsini basmaq olar (bu halda əvvəlcədən həmin düyməni alətlər panelinə çıxarmaq lazımdır);
- makrosda səhv olduqda VBA redaktoru çağırılı bilər (mausun sağ düyməsində yaranan kontekst menyü pəncərəsindən);
- **Macros** dialoq pəncərəsində hazır makrosu redaktə etməyə açaraq.

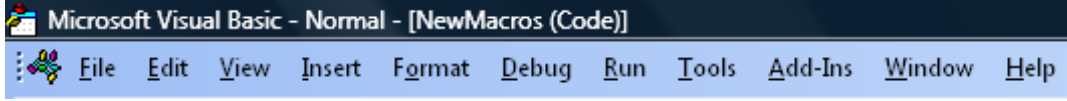


Şəkil 2.3 MS Excel kitabında Visual Basic redaktorunun pəncərəsi (redaktorun əlavə pəncərələri birləşmiş halda qurulub) .

Diqqət! VBA redaktorunda işlərkən eyni zamanda həmin Office sənədində də işləmək imkanı vardır. VBA proqramından keçid üçün klaviatürada **<Alt>+<Tab>** düymələri basılmalıdır (redaktordan “sıçramaq” üçün isə klaviatürada **<Alt>+<F11>** düymələri basılır).

Bütün obyekt istifadəli IDE (Integrated development environment) mühitlərindəki kimi, VBA-da da işləmək üçün onun qrafik interfeysində aşağıdakı komponentlər vardır (cəmi VBA-da 9 dialoq pəncərəsi vardır):

- **Əsas menyu** – WINDOWS-un başqa proqramlarındakı menyulara analogidir. OİP (obyektyönlü proqramlaşdırma) mühitinin proqramlarında istifadə olunan standart obyektlərin çoxu da formasına və iş funksiyalarına görə WINDOWS “obyektlərinə” uyğundur (şək. 2.4).



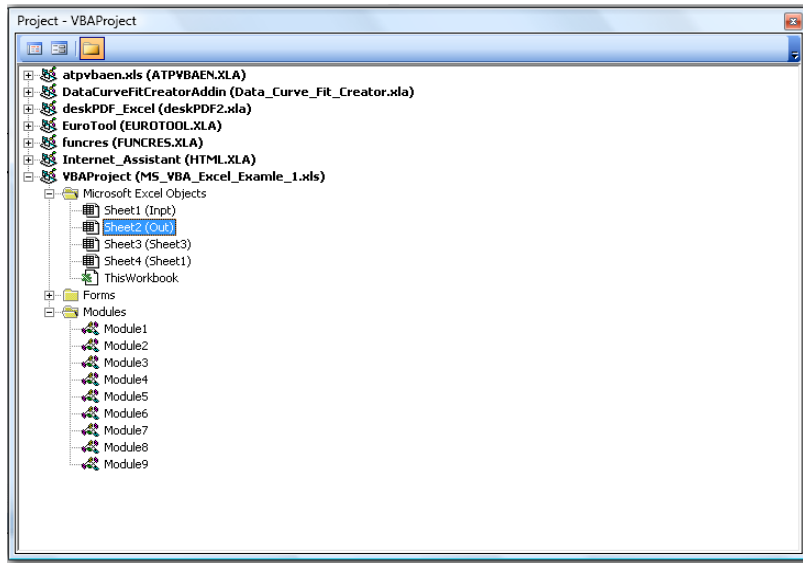
Şəkil 2.4 VBA-nın əsas menyusu.

- **Toolbox** (Alətlər menyusu) – əsas menyuda olan çoxişlənən əməllərin piktoqramlarıdır (qrafik təsvirləri) və bu menyu istifadəçinin arzusuna görə sazlanı bilər. İstənilən ActiveX idarəetməni və standart VB idarəetməni göstərir (şək. 2.5);



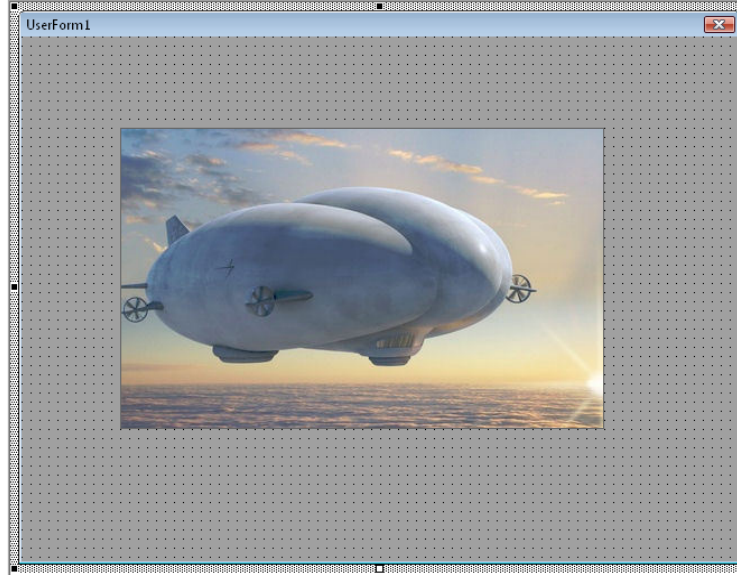
Şəkil 2.5 VBA-nın əsas menyusu (Word sənədində).

- **Project Explorer Window** – (VBA layihəsinin pəncərəsi) VBA layihəsinin bələdçisi funksiyasını yerinə yetirir. Standart formada o açıq vəziyyətdə VBA redaktorunun sol tərəfində yerləşir. Onun tərkib hissələri olan formalar, siniflər və modullar haqqında məlumatları burada görmək olur. İşlərin əsas hissəsi bu mühitdə yerinə yetirilir. Layihələrin iyerarxik siyahısı və layihənin elementlərini nümayiş edir (şək. 2.6);



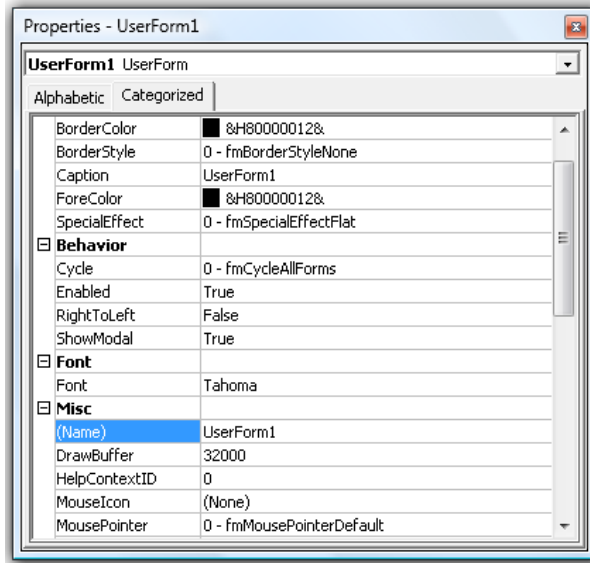
Şəkil 2.6 VBA redaktorunun **Project Explorer Window** pəncərəsi – (VBA layihəsinin pəncərəsi).

- **UserForm Window** – (istifadəçi tərəfindən tərtib edilən qrafik formalar maketlərinin pəncərəsi): VBA layihəsində yerləşdirilən formanın ekranda görünüşüdür və gələcək proqramda aparılan bütün dəyişiklikləri özündə əks etdirən cari vəziyyət monitorudur. Layihədəki idarəetmə pəncərələrinin və elementlərinin tərtib edilib göstərilməsi üçün nəzərdə tutulub (şək. 2.7);



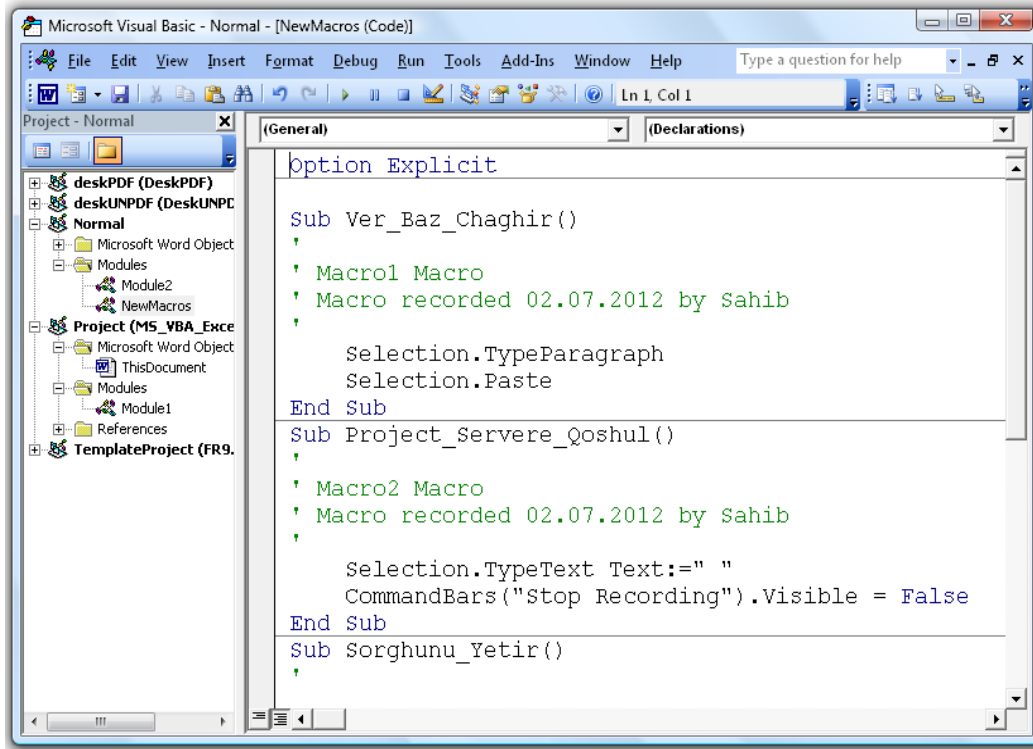
Şəkil 2.7 VBA redaktorunun **UserForm Window** pəncərəsi – (İstifadəçi tərəfindən tərtib edilən qrafik formalar maketlərinin pəncərəsi)

- **Properties Window** – (Xassələr pəncərəsi) layihədə iştirak edən aktiv obyektlərin (formaların və müxtəlif elementlərin) müxtəlif atributlarını (əlamətlərini) göstərən və yaxud yenidən təyin olunmasına imkan verən menyuları özündə saxlayır. Seçilən obyektlər (məqsədlər) üçün siyahıları və onların cari qaydaya salınmasını göstərir (şək. 2.8).



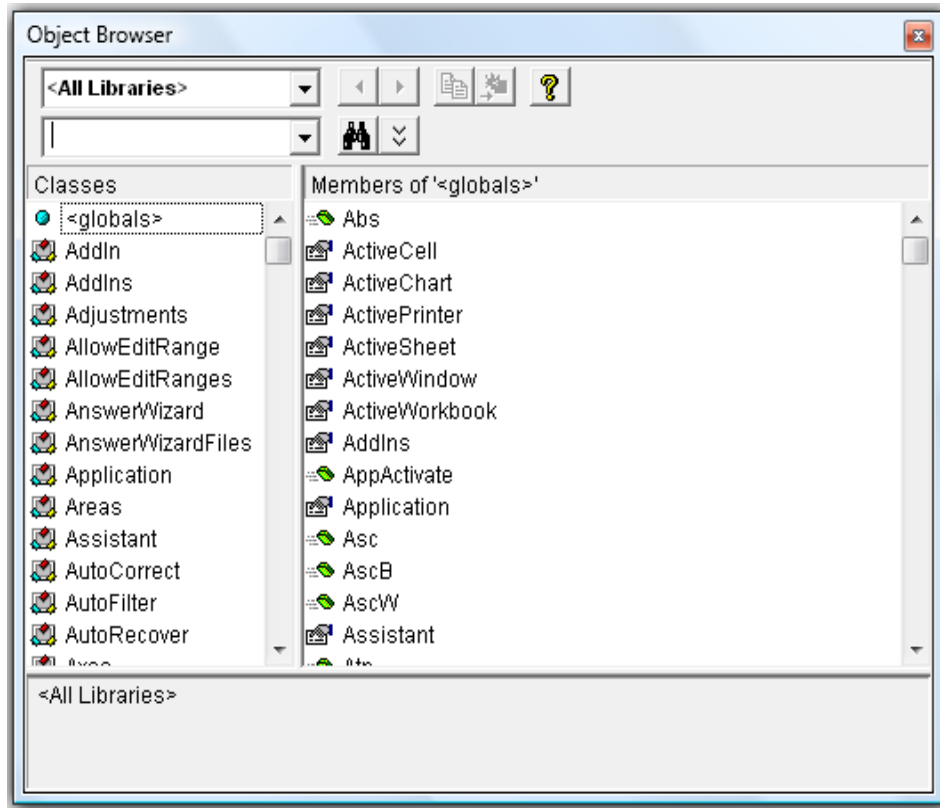
Şəkil 2.8 VBA redaktorunun **Properties Window** pəncərəsi – (Xassələr pəncərəsi)

- **Code Window** – (Proqramlaşdırma pəncərəsi) VBA-nın proqram koduna baxılmasını, makrosun yazılmasını, lazım olduqda dəyişdirilməsini təmin edən dialoq pəncərəsidir (şək. 2.9);



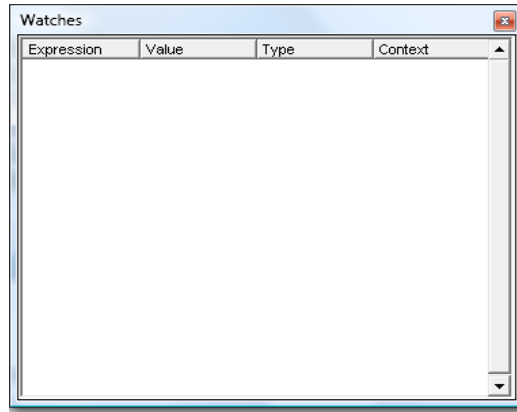
Şəkil 2.9 VBA redaktorunun **Code Window** pəncərəsi– (Proqramlaşdırma pəncərəsi)

- **Object Browser Window** – (Obyektlərə baxış pəncərəsi) layihəni təşkil edən obyektlərə işləmək üçün nəzərdə tutulmuş dialog pəncərəsidir. Layihədəki obyektlərə, onların siniflərinə, xassələrinə, əlaqələrinə baxmaq üçündür. Obyekt kitabxanasındaki sinifləri, obyektləri, metodları, hadisələr, sabitləri və layihədəki proseduraları nümayiş edir (şək. 2.10);



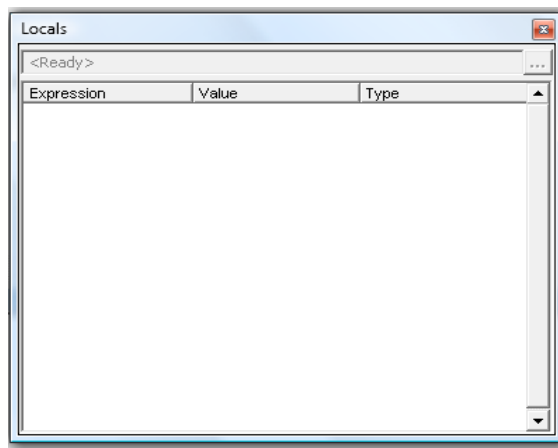
Şəkil 2.10 VBA redaktorunun **Object Browser Window** pəncərəsi – (Obyektlərə baxış pəncərəsi)

- **Watch Window** – (Dəyişənlər pəncərəsi) proqramda idarə edilə bilən dəyişənlərin, ifadələrin aldığı qiymətlərin təyin edilməsi və lazım olduqda redaktə edilməsi üçün dialoq pəncərəsidir. Saat ifadələri (zamanı kontrol etmə) təyin edildikdə avtomatik əmələ gəlir (şək. 2.11);



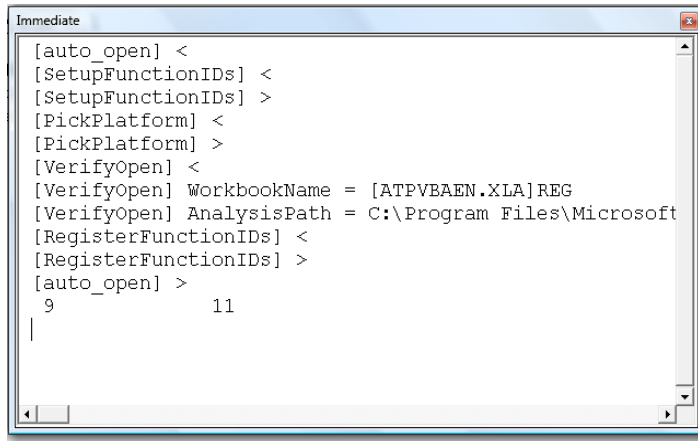
Şəkil 2.11 VBA redaktorunun **Watch Window** pəncərəsi – (Dəyişənlər pəncərəsi)

- **Locals Window** – (Lokal dəyişənlər pəncərəsi) Cari prosedurada bəyan edilmiş dəyişənlərin və onların hamısının qiymətlərini avtomatik göstərir. Proqramın test edilməsi prosesində cari proseduradakı dəyişənlərin qiymətlərinə nəzarət edilməsi (baxılması) üçün nəzərdə tutulmuş qrafik pəncərədir (şək. 2.12);



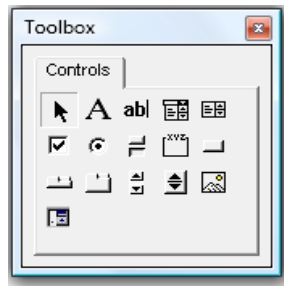
Şəkil 2.12 VBA redaktorunun **Locals Window** pəncərəsi – (Lokal dəyişənlər pəncərəsi)

- **Immediate Window** – (Dərhal nəticələr pəncərəsi) bu dialoq pəncərəsində istifadəçi alına biləcək bütün nəticələri dərhal görmək üçün çıxara bilər. Dərhal proqramın işləməsindən alınan nəticələrin vizual izlənməsi pəncərəsidir. Bunun üçün istifadəçi proqramlaşdırma üsulları ilə alına biləcək nəticələrin çapa çıxarılmasını tərtib etməlidir. Proqramın test etmə mərhələsində çox vacib alətdir. Əslində proqramın sonrakı (istifadə etmə) mərhələsində də mühüm dialoq pəncərəsidir (şək. 2.13);



Şəkil 2.13 VBA redaktorunun **Immediate Window** pəncərəsi – (Dərhal nəticələr pəncərəsi)

- **Formaların elementlər (komponentlər) paneli** - Forma üzərinə yerləşdirilə bilən idarəedici elementlər toplusudur ki, bunlar da proqramın əsas obyektlərindəndir. Bu elementlər mühitin standart obyektləri müxtəlif proqramçılar tərəfindən hazırlanmış və xüsusi «kitabxanalarda» (*library*) saxlanan obyektlər və ya öz yaratdığımız obyektlər ola bilər. Bu panel istifadəçi tərəfindən sazlanıla bilər (şək. 2.14);



Şəkil 2.14 VBA redaktorunun **Toolbox** formalar yaratmaq üçün alətlər paneli – Formaların elementlər (komponentlər) paneli

2.2 Project Explorer – VBA layihəsinin pəncərəsi və VBA layihəsinin strukturu

VBA layihəsinin pəncərəsi (**Project Explorer**) adətən VBA-nın ilkin aktivləşməsi anında açıq olur. Təsadüfən belə olmadıqda, onu aşağıdakı variantlarda çağırılması mümkündür:

- klaviaturanın **<Ctrl>+<R>** düymələri basıldıqda;
- **Standard** idarəetmə panelində **Project Explorer** düyməsi basıldıqda;
- **View**→**Project Explorer** menyusundan istifadə edərək.

Layihə (**Project**) – Word sənədi, Excel iş kitabı, PowerPoint təqdimatı v.s. Office əlavələri üçün ən yüksək səviyyədir [6]. Məsələn, əgər istifadəçi VBA-nı Word-də açsa, onda **Project Explorer**-də Word sənədlərinin cari zamanda açılmış bütün faylları və **Normal.dot** şablonu görünəcəkdir. Əgər VBA Excel-dən açılırsa, onda **Project Explorer**-də Excel iş kitablarının cari zamanda açılmış bütün faylları və xüsusi statuslu olan gizli kitab **PERSONAL.XLS** görünəcəkdir.

Office sənədlərinin tərkibində olan bütün obyektlərdən başqa (mətnlər, şəkillər, düsturlar v.s.) hər bir layihə (əslində onun özü sənəddir) – eyni zamanda standart modullar, siniflər modulları və istifadəçi formalarını öz tərkibində saxlayan konteynerdir. Onların hər birini ayrı-ayrılıqda layihəyə **Insert** menyusuna ilə **Project Explorer**-in kontekst menyusuna əlavə etmək olar. *Standart modullar* – əslində onlar VBA əmrlərini mətn formasında əks etdirən bloklardır. Burada yalnız iki hissə ola bilər:

- modulun səviyyəsini elan edən hissə (modul səviyyəsinin dəyişənlərinin və sabitlərinin elanı);
- modulun üsullarını (metodlarını) əks etdirən hissə (proseduralar və funksiyaların yerləşməsi).

Word-də macrorecorder işlədikdə **Normal.dot** layihəsində və ya cari sənəddə (makrosun saxlanma yerindən asılı olaraq) avtomatik olaraq, **NewMacros** (Excel-də isə **Module1**) yaranır. Həmin fayla macrorecorder-lə yaradılan bütün makroslar yazılır. VBA-nın əksər layihələrində bütün kodun yazıldığı, yalnız bir standart modulundan istifadə edilir. Yeni modulların yazılmağı yalnız aşağıda qeyd olunan hallarda əlverişlidir:

- import-eksportun rahatlığı üçün (**Project Explorer**-in kontekst menyusundan). Bu cür VBA tətbiqi proqramları arasındakı bloklarla (və adi VB ilə) mübadilə etmək rahat olur;
- məhsuldarlıq və səmərəliyi artırmaq üçün. İstənilən prosedura çağırılarda bütün modulun kompilyasiyası baş verir. Buna görə, bəzən, modulları müxtəlif yerlərdə yerləşdirmək sərfəlidir. Bununla həmin anda yalnız lazım olan kod kompilyasiya edilir;
- modulun daha yaxşı oxunma qabiliyyətini artırmaq üçün. Əgər istifadəçinin proqramı müxtəlif qrup məsələləri yerinə yetirirsə, onda hər qrupa aid olan kodu öz moduluna yerləşdirmək daha məsləhətlidir.

Siniflər modulu istifadəçinin özünün məxsusi siniflərinin yaradılmasına imkan verir – məxsusi obyektləri yaratmaq üçün cizgilər, sxemlər. Adətən çox mürəkkəb tətbiqi proqramlarda istifadə edilir və adi VBA əlavələrində tətbiqi olduqca azdır. Buna görə burada daha geniş baxılmasına ehtiyac yoxdur.

İstifadəçi formaları eyni zamanda idarəetmə elementlərinin və proqram kodunun “anbarı” funksiyasını daşıyır və özü də onlara və onlarla bağlı hadisələrə aiddir (bu haqda daha geniş 5-ci fəsildə danışılacaq).

Project Explorer-də daha bir vacib konteyner - **References** (Excel-də o yoxdur) istinadlar konteyneridir. Onun köməyi ilə başqa layihələrə olan istinadları görmək olur (Word sənədləri) və hansı başqa layihəyə aid proqram modullardan istifadə etmək olduğunu bilmək olur. Standart halda hər bir Word layihəsində **Normal** genişlənməsinə istinad yerləşdirilir (yəni **Normal.dot** şablonuna) – istifadəçi buradakı istənilən fayldakı makrosları elə oradanca işə sala bilər. Diqqət edilməlidir ki, həmin konteynerdə yalnız başqa sənədlərə olan istinadlar

vardır. Kitabın 4-cü fəslində başqa obyekt kitabxanalarına istinadların əlavə edilməsindən də danışılacaq.

Project Explorer-in başqa bir xeyirli imkanı budur ki, layihənin xassələrini burada tənzimləmək olur. Bunun üçün görüntüdə **Project** düyümünə mausun sağ düyməsilə şıqqıldadmaqla (Excel-də **VBAProject**) açılan kontekst menyusundan **Project Properties** seçilməli (həmin nəticəyə bu cür də çatmaq olar: **Tools**→**Project Properties**). Həmin pəncərədə bu əməllərin edilməsi də mümkündür:

- layihənin adını dəyişmək (əgər eyni adlı layihələrə istinad varsa);
- layihənin təsvir edilməsinin yaradılması, fayl haqqında məlumatlar və kompilyator istifadə edəcəyi sorğu parametrlərinin təyin edilməsi;
- parolu yaratmaqla, layihəni mühafizə etmək. Bu parolu bilmədən layihəyə baxmaq və redaktə etmək mümkün deyil.

Aşağıda **Project Explorer** pəncərəsinin köməyi ilə digər vacib olan işlərin və yarana biləcək halların təsnifatı verilir:

- öz əli ilə istifadəçi tərəfindən VBA redaktorunda makrosu yaratmaq lazım olduqda və başqa makrosların sənəddə olmadığına istifadə edilir. Bu halda layihənin düyümünün görüntüsündə mausun sağ düyməsilə şıqqıldadaraq, yaranan kontekst menyusundan **Insert**→**Module** əmri seçilməlidir. Bununla layihədə yeni modul yaranacaq və dərhal kodlaşdırma redaktorunda proqramlaşdırmaq üçün açılacaq. Sonrakı addımlar bu fəslin növbəti bölmələrində daha ətraflı açıqlanacaq;
- əgər istifadəçi tərəfindən həmin layihədə artıq makros yaradılıbsa (macrorecorder-lə yaxud öz əli ilə VBA redaktorunda), onda modul, avtomatik olaraq, yaradılmış olacaq. Onu **Modules** konteynerinin altında görmək olar və onu açmaq üçün görüntüsünün üstündə mausun sol düyməsilə iki dəfə şıqqılatmaq kifayətdir. Orada istifadəçi yaratdığı (macrorecorder-lə və ya macrorecorder-siz) makrosları görə bilər;
- əgər istifadəçi idarəetmə elementləri olan qrafik forma yaratmaq istəyirsə (düymələr, mətn sahələri, açılan siyahılar, keçirici açarlar və bayraqcılar v.s.) bunun üçün layihənin düyümünün görüntüsündə mausun sağ düyməsilə şıqqıldadaraq, yaranan kontekst menyusundan **Insert**→**UserForm** əmri seçilməlidir. Yeni forma formalar dizayneri pəncərəsində açılacaq (daha ətraflı bu redaktorla işləmə qaydaları kitabın 5-ci fəslində yazılacaq).

Diqqət! İstifadəçi əvvəlcədən düşünməlidir ki, yaratmaq istədiyi kod harada yaradılmalıdır: yalnız həmin sənəddəmi yaxud həmin Office proqramının bütün sənədlərindəmi. Əgər yalnız həmin sənəddə yaradılmalıdırsa, onda həmin sənədin standart proqram modulundan istifadə edilməlidir. Digər halda, bütün sənədlərdə kodun tətbiq edilməsi lazım olduqda, onda layihənin proqram modullarından istifadə edilməlidir (**Normal Word**-də və **PERSONAL.XLS** Excel-də).

Növbəti fəsildə əvvəldən yaradılmış proqram modulunda VBA kodu redaktoru ilə işləmək qaydaları haqqında danışılacaq.

2.3 Code Editor Window - VBA kodunun redaktoru ilə işləmə

VBA-nın kod redaktorunda proqramlaşdırma işinin əsas hissəsi yerinə yetirilir, buna görə onunla işləmə üsullarını və metodlarını çox yaxşı bilmək lazımdır.

2.3.1 Pəncərənin açılması və onun strukturu

VBA kod redaktorunun işə salınması və yazılası mətnin redaktə etmə üsulları

VBA kod redaktorunun bir neçə açılma üsulu var:

- lazımı elementi seçmək (**Project Explorer**-də, formalar dizaynerində v.s.) və kontekst menyusunda **View**→**Code** əmrini seçmək;
- <F7> düyməsini basmaq;
- menyudan **View**→**Code** əmrini seçmək;
- **Project Explorer**-də modul obyektinin görüntüsünə mausun sol düyməsilə iki dəfə şıqqıldatmaq (və ya onun görüntüsünü seçərək klaviaturanın <Enter> düyməsini basmaq).

Ümumiyyətlə, VBA kodunun redaktoru əslində mətn redaktorudur və burada istifadəçi kodun bir hissəsini kəsib sonra da müəyyən edilmiş yerə qoya bilər [1, 4, ..., 18]. Mətnin bir hissəsini çəkib-atmaq, klaviaturanın <Ctrl> basılmış düyməsi ilə mətni kopyalayıb mətnin seçilmiş yerinə yerləşdirmək, tamamilə Word mətn redaktorundakı kimidir. Hər halda bu redaktorun başlıca məqsədi istifadəçinin ən əlverişli və rahat şəraitdə VBA kodunun yaradılması, testdən keçirilməsi və hazır proqram məhsulunun işlədilməsi ilə bağlıdır. Bu redaktorun xüsusi imkanları haqqında növbəti fəsildə danışılacaq.

2.3.2 Obyektlərin və hadisələrin siyahısı

Code Editor Window - VBA kodunun redaktoru pəncərəsində obyektlərin və hadisələrin siyahısı, Declarations hissəsi.

Kod redaktorun yuxarı hissəsində iki siyahı vardır (şək. 2.9): soldakı proqramdakı obyektlərin siyahısıdır, sağdakı proqramdakı proseduralar/hadisələr siyahısıdır.

Soldakı (**General**) obyektlər siyahısında istifadəçi modulun proqram kodu şəklində olan obyektini seçə bilər. Əgər istifadəçi yaratdığı formanın proqram kodunu açmış olsa, onda həmin siyahıda formanın özünü və ya onun tərkibində olan istənilən idarəetmə elementi seçərək, istifadəçi onun üçün də xüsusi proqram kodunu yazıya bilər. Sağdakı proseduralar/hadisələr siyahısında (**Declarations**) hissəsi var. Bu hissədə bütün modula aid olan məlumatlar yerləşir:


- əgər standart moduldursa, onda onun səviyyəsinin elanı verilir;
- əgər formaya aid proqram kodudursa, onda bütün proseduraların (makrosların) siyahısı verilir.

İstifadəçi tərəfindən lazımı hadisə seçildikdə, avtomatik olaraq, həmin hadisəni emal edən uyğun prosedura yaradılacaqdır.

2.3.3 Proqram kodunun pəncərəsinin bölünməsi və quraşdırmaları

VBA-nın kod redaktorunda quraşdırmaların tətbiqi, bölünmə xətti

Bəzən, proqram kodu bir yerdə yazıldıqda, kodun tamam başqa hissəsinə aid istifadəçinin ağılına yeni bir ideya gələ bilər. Belə hal yarandıqda istifadəçi proqram kodunun bir hissəsindən digər hissəsinə “sıçramaq” imkanından istifadə edəsi olur. Sonra isə, əgər tərtib edilən proqram kodu “uzundursa”, o xeyli vaxt proqram kodunun lazım olan yerini axtararı ola bilər. Bu cür üsul proqramlaşdırma prosesini məhsuldar və səmərəli edə bilməz. Belə hallarda peşəkar proqramçılar quraşdırmalardan istifadə edirlər. VBA-nın kod redaktorunda bunun üçün (adi kitablarda oxucular tərəfindən qoyulan xüsusi səhifələrin arasına yer nişanlanmasına oxşayır) quraşdırma imkanı vardır. Bunun üçün VBA redaktorunun əsas idarəetmə panelində **Edit**→

Bookmarks təlimatı seçilməlidir, yaxud alətlər panelindən (**Toolbox**)  **Edit** alətlər panelindəki düymələr seçilməlidir. Yer nişanlanmasını aktivləşdirmək və ya yox etmək lazım olduqda **Toggle Bookmark** düyməsi seçilməlidir. Bəzən də proqramlaşdırma prosesində redaktor pəncərəsinin iki yerə bölünməsinə böyük ehtiyac yaranır: modulun müxtəlif hissələrinə baxmaq üçün, həmin hissələri seçib kopyalamaq üçün v.s. Bunun üçün redaktorun pəncərəsindəki kod yazılma yerində ayırıcı xətt yerləşdirilir. Həmin əməliyyatı həyata keçirmək üçün pəncərənin diyirlənmə zolağının tam yuxarı hissəsində yerləşən kiçik məkikdən istifadə edərək (onu mausla aktivləşdirərək), yaranan ayırıcı xətti kodun lazımı yerinə çəkib-atmaq lazımdır.

2.3.4 VBA redaktorunda proqramının yazılması

VBA redaktoru: xassələr, metodlar və parametrlərin siyahısının və sözlərin avtomatlaşdırılmış əlavəsinin alınması

Kod redaktorunda proqram tərtibatçısının işinin rahatlaşdırılması üçün bir sıra vasitələr quraşdırılmışdı. Onlardan istifadəçi üçün ən vacibləri aşağıda göstərilib:

- ən xeyirli vasitə - xassələr və metodlar (**Properties/Methods**) siyahısının alınmasıdır. Əksər VBA proqramlarında müxtəlif VBA obyektlərinin xassələrindən və metodlarından istifadə edilir (daha ətraflı bu haqda kitabın 4-cü fəsilə obyektlər və obyekt modellərinə həsr edilmiş bölmələrdə danışılacaq), bununla yanaşı bir çox metodlar parametrləri

qəbul edir. Hər bir xassə və metodun adını, parametrlərin ötürülmə ardıcılığını yadda saxlamaq istifadəçi üçün çox müşkül iş sayıla bilər. Üstəlik əgər istifadəçi həmin obyekt üçün hər dəfə sorğu məlumatına əl atmalı olursa, bu əməliyyatlara çox vaxt sərf oluna bilər. Nəticədə proqramlaşdırmanın məhsuldarlığı və səmərəliliyi haqqında danışmaq mümkün olmaz. Bu çətinliyi aradan qaldırmaq üçün asan bir yol var: əgər avtomatik göstərilmə işə salınıbdırsa (standart halda o həmişə işlək vəziyyətdə olur), onda obyektin adını yazıb sonunda nöqtə qoymaq kifayətdir. Əgər avtogöstəri keçirilmiş vəziyyətdədirsə, onda **Edit** menyusunda **List**→**Properties/Methods** təlimatı seçilməlidir və ya klaviaturada **<Ctrl>+<J>** düymələri basılmalıdır. Lazımı xassə/metod seçildikdə (bir neçə hərf çap etmək və ya mausdan istifadə etmək olar), klaviaturanın **<Tab>** düyməsi basılmalıdır. Həmin vasitə istifadəçinin siniflər/dəyişənlər hissəsində də işləyir. Əgər işləməsə, onda **Options** dialoq pəncərəsində (**Tools**→**Options** menyusunda) **Auto List Members** parametri sazlanmalıdır;

- *metod üçün arqumentlərin siyahısının və onlar haqqında məlumatın* alınması – bunun üçün parametrləri qəbul edən metodun adını çap edərək həmin məlumatları, avtomatik olaraq, görmək olar. Əl ilə həmin nəticəyə bu cür çatmaq olar: klaviaturada **<Ctrl>+<I>** düymələri basılmalı, **işə salınma/keçirilmə** isə **Tools**→**Options**→**Auto QuickInfo** təlimatı ilə həyata keçirilə bilər. Klaviaturanın **<Ctrl>+<Shift>+<I>** düymələri basıldıqda parametrlər haqqında və arqumentlərin siyahısı haqqında məlumat göstərilir;
- *sabitlər siyahısı* daha bir vacib vasitədir – bununla istifadəçi cari xassə üçün buraxılabilən qiymətlər haqqında məlumat ala bilər. Bu vasitə bərabərlik işarəsi (=) çap edildikdə avtomatik olaraq əmələ gəlir. Başqa üsulla klaviaturada **<Ctrl>+<Shift>+<J>** kombinasiyası yığılmalıdır. Sabitlər haqqında daha geniş növbəti hissələrdə danışılacaq.

VBA-nın “açar” sözlərini və hal-hazırda əlçatan olan siniflərin adının sözlərin avtomatik əlavə etmə vasitəsindən (**Complete Word**) istifadə etmək məqsədə uyğundur. Bunun üçün kifayətdir ki, klaviaturada **<Ctrl>+<Space>** kombinasiyası yığılsın. Əslində heç-bir söz yazmamaq da olar, yaxud bir-iki hərfin yazılması kifayət edə bilər.

VBA kod redaktoruna aid daha bir neçə qeyd:

- əgər istifadəçi kod yazıldıqda ilk sətirdə bir neçə abzas yer buraxmışdırsa, onda həmin sayda abzas, avtomatik olaraq, növbəti sətirlərə də təyin olacaqdır. Həmin xassəni **Options** dialoq pəncərəsində **Auto Indent** parametri sazlandıqda rədd etmək olar;
- əgər kod redaktoru açar sözünü tanıyarsa, onda, avtomatik olaraq, açar sözünün birinci hərfini böyüdərk göy rəngə boyayır;
- bəzən bir neçə sətiri şərh (kommentari) halına salmaq və ya, əksinə şərh halından çıxarmaq lazım olur. Bunun üçün alətlər panelində **Edit** seçilir, sonra isə **Comment Block** (şərh etmə) və **Uncomment Block** (şərhdən çıxarma) seçilə bilər;

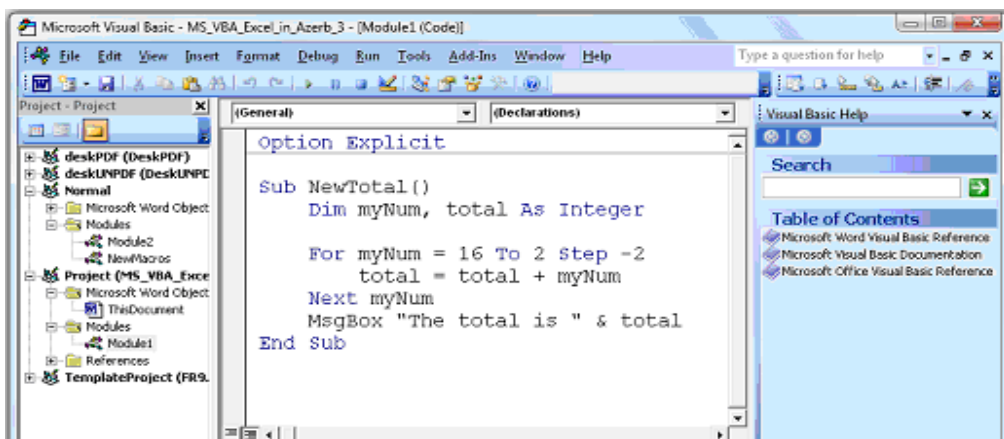
- əgər prosedura yaradıldıqda istifadəçi **Sub** yaxud **Function** “açar” sözünü yazırsa, onda redaktor, avtomatik olaraq, **End Sub** və ya **End Function** operatoru ilə proseduranın sonunu tamamlayır. Proseduralar arasında avtomatik olaraq ayırıcı xətt əmələ gəlir;
- əgər yeni sətərə keçdikdə redaktor sintaktik səhvi təyin edibsə, onda istifadəçiyə dərhal bu haqda məlumat verilir. Bəzi peşəkar proqramçılar bu xassə əsəbləşdirir. Belə hallarda həmin **Options** pəncərəsində **Auto Syntax Check** parametrindəki bayraqçı çıxarılmalıdır. Lakin bu xassənin olmaması əslində işin aparılmasına mane olmur, çünki istənilən halda sintaktik səhvlər dərhal qırmızı rənglə boyalanır.
- VBA redaktorunda proqram tərtibatı üzərində işlərkən istifadəçinin bir neçə pəncərədə eyni vaxtda işləməsi mümkündür. Pəncərələr arasına keçid klaviaturada **<Ctrl>+<Tab>** və ya **<Ctrl>+<F6>** düymələr kombinasiyası ilə həyata keçirilə bilər;
- standart halda VBA redaktoru **Full Module View** rejimində işləyir – modulun tərkibindəkiləri bütövlükdə göstərmə mənasını daşıyır. Proseduraların ayrı-ayrılıqda göstərilməsi lazım olduqda isə **Procedure View** rejiminə keçmək olar. Buna nail olmaq üçün redaktorun iş pəncərəsinin sol tərəfində ən aşağı küncündəki düymələrdən istifadə etmək lazımdır.

2.4 VBA redaktorun sorğu materialları ilə işləmə qaydaları

VBA-nın sorğu materiallarının hissələri, lazımı məlumatın axtarılıb tapılma üsulları

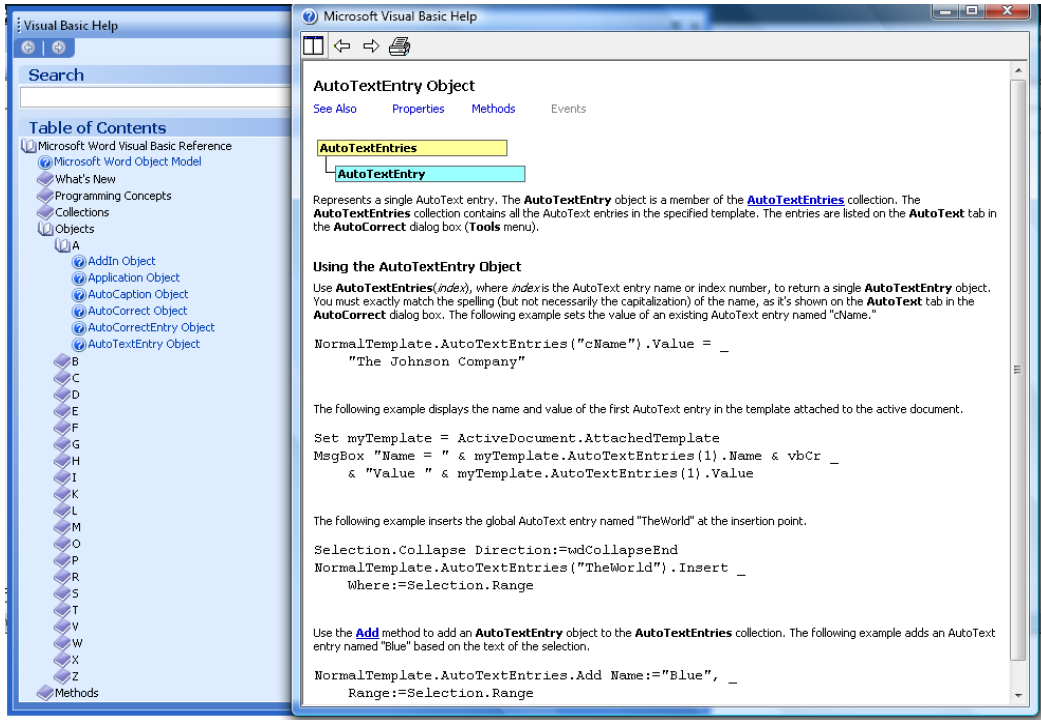
Office proqramlaşdırmasında sorğu materialları ilə işləmə - bir o qədər də, birinci baxımdan, görünən kimi aşkar deyil.

VBA redaktorunda işlərkən, sorğunun pəncərəsi klaviaturanın **<F1>** düyməsi basıldıqda əmələ gəlir. Həmin nəticə **Standart** panelində **Help** düyməsi basıldıqda da alınır. Nəticədə (şək. 2.15 və şək. 2.16) VBA-nın Office-in açılmış sənədinə uyğun sorğu pəncərəsi açılacaq. Bəzən istifadəçi sorğu pəncərəsini redaktor pəncərəsinə birləşmiş vəziyyətdə istifadə edilməsinə üstünlük verir (şək. 2.16).



Şəkil 2.16 MS Word Office sənədində VBA redaktorunun sorğu altpəncərəsi (sağ tərəfdə, əsas pəncərəyə birləşmiş vəziyyətdə)

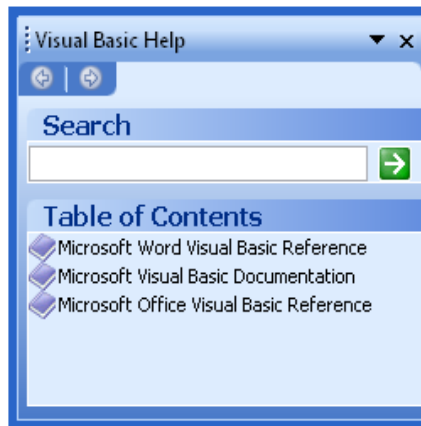
Bəzən isə sorğunun pəncərəsi ayrı olduqda istifadəçi üçün daha rahat vəziyyət yaranır (şək. 2.17).



Şəkil 2.17 Office proqramında VBA redaktorunun sorğu pəncərələri (əsas pəncərədən ayrılmış vəziyyətdə).

Sorğu pəncərəsinin çağırılmasının daha bir üsulu da belə ola bilər – kod redaktorunda mausun göstəricisini lazımı yerə (məsələn, çağırılacaq metodun üstünə və ya istifadə edilən xassəyə) gətirib klaviaturanın <F1> düyməsini basmaq lazımdır. Fərq budur ki, sonuncu üsulda bir neçə sayda cavabın gözlənməsi ehtimalında (məsələn obyekt **Range** yaxud xassə **Range**), avtomatik olaraq, yalnız istifadəçinin axtardığı düzgün cavab veriləcək.

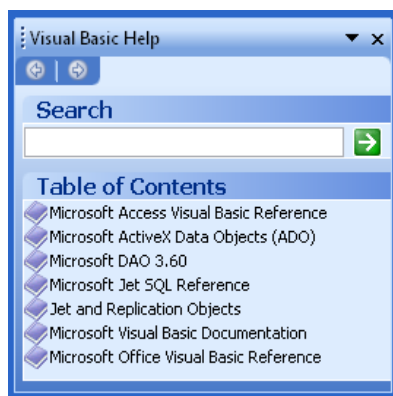
MS Office sənədində proqramlaşdırmağa aid sorğu adətən üç hissədən ibarət olur (şək. 2.17):



Şəkil 2.17 MS Office sənədində proqramlaşdırmağa aid sorğunun üç əsas hissəsi

- birinci hissə (**Microsoft Excel Visual Basic Reference, Microsoft Word Visual Basic Reference** v.s.) — Office proqramının özünün obyekt modelləri üzrə sorğusudur;
- ikinci hissə (**Microsoft Visual Basic Documentation**, bütün Office proqramlarında eynidir) — bu, əslində Visual Basic for Applications dilinin özünün sintaksisi və quraşdırılmış funksiyaları haqqında olan sorğudur;
- üçüncü hissə (**Microsoft Office Visual Basic Reference** - bu da bütün Office proqramlarında eynidir) — Office proqramının ümumi imkanlarının sorğusudur: alətlər paneli və standart menyu ilə proqramlaşdırma işlərinin aparılması, bələdçi ilə işləmək qaydaları, **Windows SharePoint Services** ilə qarşılıqlı işləmənin təşkili v.s. haqqında məlumatlar var.

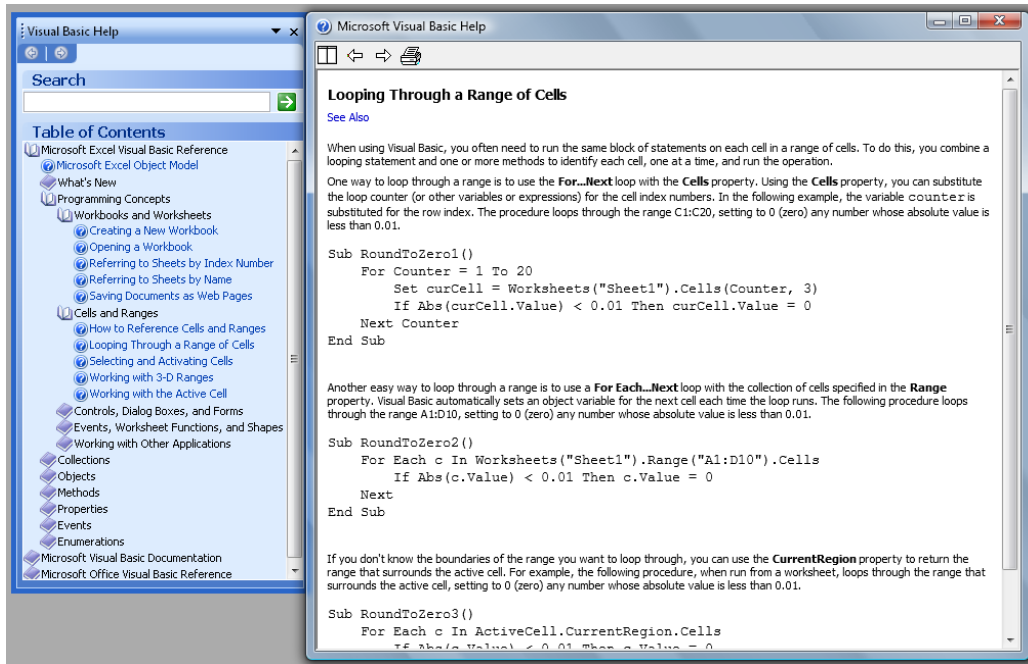
Office proqramlarının bəzilərində (məsələn, Microsoft Access) sorğuya əlavə hissələrdə qatılıb (şək. 2.18) – DAO-nun obyekt modellərinə aid və SQL dilinə aid v.s.



Şəkil 2.18 MS Access sənədində proqramlaşdırmağa aid sorğunun hissələri.

Adətən ən vacib hissə kimi istifadəçilər Office-in konkret proqramının imkanlarına həsr edilmiş sorğu materiallarına önəm verirlər (bu əslində düzgün sayılmalıdır). Həmin hissəni isə şərti olaraq iki hissəyə ayırmaq olar (şək. 2.19):

- **Programming Concepts** (*proqramlaşdırma konsepsiyaları*) — bu hissədə ən geniş yayılmış əməliyyatları proqramlaşdırma ilə necə yerinə yetirilməsi haqqında ətraflı məlumatlar verilir. Məsələn, Excel-də iş kitabının açılması yaxud yaradılması imkanları, lazımı səhifələrin tapılması, məlumatın hücrəyə (xanaya) yazılması və ya oradan oxunması v.s.
- *Office proqramının obyekt modelləri haqqında sorğunun alınması*: kolleksiyalar, obyektlər, xassələr, metodlar v.s. Bununla belə, əsas anlayışlar haqqında olan məlumatlar (onlar daha çox konsepsiyalar kateqoriyasına aiddir: məsələn, Excel-də hansı üsullarla **Range** obyektinin yaradılması haqqında) sorğunun uyğun olan obyektinə aid verilir. Konkret obyektin bütün funksional imkanları haqqında təsəvvür yalnız onunla bağlı bütün xassələri və metodları ardıcıl olaraq öyrəndikdən sonra formalaşsa bilər.



Şəkil 2.20 MS Excel kitabında proqramlaşdırmaya aid VBA Excel-in imkanlarına həsr edilmiş sorğu materialları (analoji spesifik olan materialları Office-in digər proqramlarında da tapmaq olar)

Lazımı istiqaməti tapmaq (yəni obyekt və onun xassələrlə metodlarını) üç üsul ilə həyata keçirilə bilər:

- **Programming Concepts** hissəsinə baxılmalıdır (proqramlaşdırma konsepsiyaları) — istifadəçi rast gəldiyi hadisə orada təsvir edildiyini tapa bilər;
- Bütün obyektlərlə ardıcıl olaraq tanış olmaq üçün (xassələri və metodları ilə birləşdirmə), bu cəhdlə ki, alınan məlumat yaxın zamanda xeyirli ola bilər. Bu üsul ən səmərəsiz üsul sayıla bilər. Çünki Office proqramında yüzlərlə obyekt ola bilər (tez-tez istifadə edilən obyektlərin sayı əslində bir o qədər də çox olmur: bu kitabda onlar haqqında məlumat kifayət qədər verilib). Digər tərəfdən, əgər istifadəçi uzun zaman Office-in hansısa sənədində proqramlaşdırma ilə məşğul olacaqsın, onda bu üsul ilə bir neçə gün ərzində bütün obyektlər haqqında hər tərəfli məlumatın toplanması (hətta, lazım olduqda, bəzi vacib sayılan məlumatların konspektlənməsi) məqsədə uyğun sayılmalıdır;
- lakin ən məntiqli və səmərəli üsul — macrorecorder-də lazımı əməliyyatların aparılmasından və sonra da yaranmış kodu analiz etməkdən ibarətdir (macrorecorder-in düzgün yol göstərməsi əksər hallarda özünü doğrultmaya da bilər).

Bu hissədə, son olaraq, bir vacib olan amili yada salmaq istəyirik (bu haqda biz artıq danışmışdıq): təəssüflər olsun ki, nə Azərbaycan nə də, dilimizə yaxın olan, Türk dilində (hətta çox larimizin mükəmməl bildiyimiz Rus dilində belə) həmin sorğular Microsoft tərəfindən verilməyib. Yalnız İngilis dilini bilərək, istifadəçi daha dolğun məlumatlara yiyələnməyə bilər. Əlbəttə,

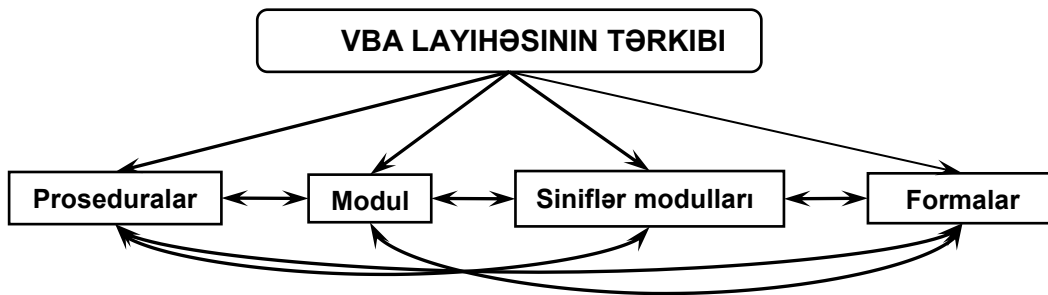
Azərbaycanca yazılan bu kitab VBA haqqında oxucu üçün xeyli faydalı ola bilər. Hər halda oxucu həmin sorğu materiallarını orijinal formada oxusa daha yaxşı nəticələr alınə bilər.

3. MS VBA SINTAKSİNİN ƏSASLARI

3.1 VBA sintaksisinin əsas strukturu və əsas elementləri

MS VBA dili bütün müasir obyektönlü alqoritmik dilləri kimi müxtəlif tip verilənlərlə əməliyyatlar apara bilər. MS VBA dilinin son versiyaları ilə müqayisədə bu dilin verilənlərinin tipləri arasında müəyyən fərqlər ola bilər, ancaq bu fərqlər təsir edici qədər böyük deyil. Buna görə hətta MS Office 97 üçün tərtib edilən VBA proqramları, adətən, heç bir qüursuz yeni versiyalarda da çalışmaq qabiliyyətini itirmir. Yüksək səviyyəli alqoritmik dillərdə qəbul edilmiş proqramlaşdırma standartları MS VBA-da mövcuddur [1, 2, 3, 4, 6, ..., 18]: məsələn, siniflər kateqoriyası, verilənlərin tipləri istifadə olunmazdan əvvəl mütləq onların elan edilməsi v.s. VBA-nın obyekt kitabxanasında 100-dən çox müxtəlif iyerarxiyalı obyektlər vardır.

Burada biz yenidən VBA-nın ümumi strukturunu yada salsaq yerinə düşər. Çünki bu obyekt yönlü proqram təminatının sintaksisini anlamaq və lazımı səviyyədə mənimsəmək üçün onun tərkib strukturunu yaxşı bilmək lazımdır. Aşağıda, şəkil 3.1-də VBA-nın əsas tərkib strukturu, onun elementləri və elementlər arasında olan qarşılıqlı əlaqəli sxem formasında göstərilib.



Şəkil 3.1 VBA-nın əsas tərkib strukturu

Burada:

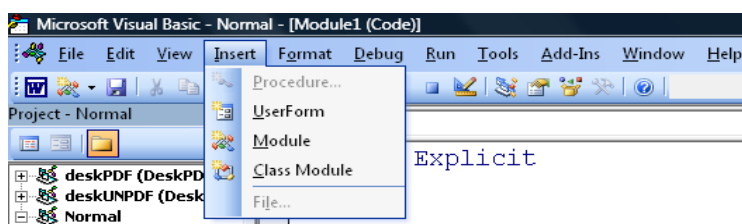
- Modullar - istifadəçi tərəfindən yaxud başqa proqramçılar tərəfindən tərtib edilən makrosalar (proqramlar, funksiyalar, ola bilsin, altproqramlar və proseduralar) ola bilər;
- Proseduralar - istifadəçi tərəfindən yaxud VBA-nın standart proseduralar kitabxanasına aid olan proqramlardır. Başqa sözlə, adi hesablamalar ardıcılıqlarından ibarət hesablama təlimatlarının toplusuna hesablama prosedurası demək olar. Məsələn, proseduraların tərkibində olan tiplər bunlardır: **Function**, **Property**, və **Sub**. Proseduranın adı həmişə modul səviyyəsində elan edilir;

- Siniflər modulları - qeyri standart obyektləri idarə etmək üçün yaradılan proqramlara siniflər modulları deyirlər. Qeyri standart modullar deyəndə, obyekt modeli obyektlər kimi anlamaq lazımdır (məsələn, Excel-in tərkibində olan səhifələr, diaqramlar, şriftlər v.s., cəmi 200-dən artıq qeyri standart modul kateqoriyasına aiddir);
- Formalar - qeyri standart, istifadəçi tərəfindən yaradılan dialoq pəncərələridir (yaradılmasında standart vizual formalardan istifadə edilir - lazım gələrsə, xüsusi proqram da tərtib edilə bilər).

3.2 MS VBA-da istifadəçi tərəfindən yaradılan proqramların və funksiyaların strukturu

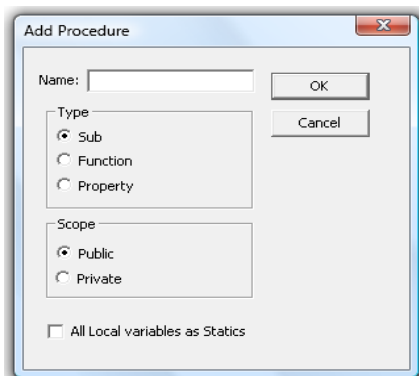
MS VBA-da proqramlaşdırma zamanı istifadə olunan əsas proqram vahidləri altproqramlardır. Onlar iki qrupa ayrılır: *proseduralar* (**Sub**) və *funksiyalar* (**Function**). Proseduranın funksiyadan fərqi ondadır ki, funksiyanın adı ilə hər hansı bir kəmiyyət bağlıdır və buna görə də funksiyanın adını müəyyən əməllərdə göstərmək olar. Aşağıda MS VBA prosedura və funksiyaların strukturu verilmişdi. Hər şeydən əvvəl prosedura və ya, lazım olduqda, funksiyaları tərtib etmək üçün istifadəçi gerek müəyyən ardıcılıqla əməlləri yerinə yetirsin:

- yeni modul yaradılmalıdır (VBA-nın idarəetmə panelində **Insert**→**Module** təlimatı seçilməlidir, şəkl. 3.2);



Şəkil 3.2 VBA-nın idarəetmə panelində yeni modulun yaradılması

- yaradılmış modulun qrafik görüntüsündə (**Project Explorer Window**-da) VBA-nın idarəetmə panelində yenidən **Insert**→**Module** təlimatı seçilməlidir (kontekst menyuda **Procedure** təlimatı aktiv formada görünəcək) və açılan kontekst menyudan **Procedure** əmri seçilməlidir, şəkl. 3.3;



Şəkil 3.3 VBA redaktorunda yeni prosedura (və ya funksiya) yaradılmasını təmin edən **Add Procedure** dialoq pəncərəsi.

- nəhayət yaranmış **Add Procedure** dialoq pəncərəsindən istifadəçi lazım olan proseduranı (funksiyanı) seçib, onun adını təyin edib və statusunu müəyyən edib (**Public** və ya

Private), VBA redaktorunda proqramı (prosedura və ya funksiya formasında) yazmağa başlamaq olar.

VBA prosedurasının strukturu aşağıda göstərilib. Burada göstərilən nümunədə və bütün kitabda VBA kodunu göstərdikdə, proqramın aydın anlanması üçün, bu cür qaydalardan istifadə edilib (nəzərə alınmalıdır ki, Microsoft VBA redaktorunun standart vəziyyətində təyin etdiyinə görə idarəetmə operatorlarını göy rənglə seçilir, proqramdakı şərhlər (kommentariyalar) nazik və yaşıl rənglə seçilir, digər identifikatorlar, məsələn, istifadəçi tərəfindən təyin edilən identifikatorlar, nazik və qara rənglə seçilir):

- kodun şrifti VBA redaktorunda olan kimi - Unicod-un Courier New şrifti seçilib;
- idarəetmə operatorları qalın şriftlə seçilir;
- istifadəçi tərəfindən təyin edilən identifikatorlar nazik şriftlə seçilir;
- müxtəlif xassələr, metodlar və hadisələr identifikatorları (standart olub olmamasına baxmayaraq) nazik şriftlə seçilir;
- proqramdakı şərhlər (kommentariyalar) nazik şriftlə seçilir – şərhlər bir qat dırnaq arası “'” işarəsindən sonra başlayır.

```
Sub prog1()      '{Bu sətirdə proqramın adı Sub operatorundan sonar
'yazılır (yeni prosedura tərtib edilməsi üçün açıldıqda həmin Sub
'operatoru, avtomatik olaraq, kod sətirində çap edilmiş olur)
'@@@@@@@@@@ Bu sətir VBA şərhinə aiddir @@@@
'{sabitlərin elanı başlayır}
'*****

Const pi=3.14

.....
.
.
.....
'*****

Dim a As Integer
Dim b As Variant
Dim x, y, z As String
.....
.
.
.....
Dim z As Range

'##### Bu yenə də VBA şərhidir #####
'***** Bu sətir də VBA şərhidir *****
```

Burada istifadəçinin təyin etdiyi prosedurdakı bütün identifikatorlar (dəyişənlər və sabitlər) elan edilir.

```

.....
.
.
.....
} ← Burada tərtib edilən
    program yerləşdirilir

'***** Həmçinin bu da VBA şərhidir *****
Debug.Print cavab_1, cavab_2, ...,
.....
.
.
.....
} ← Burada alınan nəticələri
    Immediate Window
    pəncərəsinə çıxarmaq
    olar.

Debug.Print cavab_n
'*****

End Sub '{Bu sətirdəki operatorlar (End Sub) VBA proqramının
'tamamlanmasını (sonunu) göstərir (yeni prosedura tərtib
'edilməyə açıldıqda həmin operatorlar, avtomatik olaraq, kod
'sətrində çap edilmiş olur)
'*****

'Aşağıdakı sətirlərdə Function prosedurası (əslində bu tip proseduradan
'proqramda bir neçə sayda ola bilər) və ya altproqramlar kimi
'yeni Sub proseduraları yerləşdirilə bilər.

Function func1(Parametrlərin siyahısı)
    Const          '{sabitlərin elanı}
.....
'*****

Dim ... As ...    '{tiplərin və dəyişənlərin elanı}
.....
'*****

..... '{burada funksiyanın hesablaması
        '{prosedurası yerləşir}

'*****

End Function

```

3.3 VBA sintaksisinin əsas qaydaları

VBA sintaksisi yeni öyrənenlər üçün həddən artıq vacib mövzu sayılmalıdır desək - səhv etmərik. Necə ki, vurma cədvəlini bilmədən riyazi hesablamalara başlamaq mümkün olmayan kimi, istənilən alqoritmik dilinin (o cümlədən VBA-nın) sintaksisini bilmədən proqramlaşdırmağa başlamaq ağılaşılmaz iş sayılmalıdır. MS VB (Microsoft Visual Basic) proqram təminatı ilə artıq tanış olmuş (bəlkə də peşəkar səviyyəsində) oxucu isə bu bölməni oxumadan da üstündən rahatca keçə bilər. Çünki VB və VBA sintaksisi tamamilə eynidir. Bununla belə hər bir oxucu

bilməlidir ki, VBA əsasən adi istifadəçilər (proqramlaşdırma sahəsində peşəkar olmayanlar) üçün nəzərdə tutulmuşdur. Buna görə bu dilin sintaksisi istifadəçi üçün olduqca “dost” şərtlərinə uyğundur: VBA-nın “çox ciddi” olan sintaksis qaydalarından qorxmaq lazım deyil. Lakin onu da qeyd etməliyik ki, VBA sintaksisi ən müasir sayılan obyektönlü dillərin standartlarına uyğundur. Həmin oxucular ki, başqa müasir proqramlaşdırma dilləri ilə işləmə təcrübəsinə malikdir (C++, Java, Python, Delphi, VBScript və JavaScript, Perl v.s.) bu bölmədə təsvir edilən materialı asanlıqla mənimsəyəcəklər. Proqramlaşdırma sahəsində heç bir təcrübəsi olmayan oxucu isə maksimum diqqətlə təqdim olan materialın öyrənməsi (ola bilsin konspektləşdirilməsi) və müəyyən müddət ərzində mənimsənilmiş bilikləri praktikada tətbiq etməsi məqsədə uyğun sayılmalıdır. Sərf olan güc sonda real praktiki işlərin yerinə yetirilməsində, VBA proqramlaşdırmasında, özünü hökmən göstərəcək. Material bir o qədər də böyük həcmdə deyil – çünki VBA dili əslində olduqca sadə dildir.

İndi isə VBA dilinin sintaksisinin ümumi cəhətlərini qeyd edək. Yuxarıda qeyd etdiyimiz kimi, VBA sintaksisi VB sintaksisindən fərqlənir (üst-üstə düşür desək haqlı olarıq). Bu dilin əsas sintaktik prinsipləri budur:

- VBA registr keçidinə həssas deyil (böyük yaxud kiçik işarələrə);
- bir kod sətirini sonuna qədər şərh (kommentari) şəklində salmaq üçün (') işarəsi sətirin əvvəlində ən birinci işarədən öncə qoyulmalıdır və ya həmin yerdə (hökmən şərh ediləsi yazıdan öncə) ənənəvi əski Basic-də olan kimi **Rem** əmri yazılmalıdır (avtomatik olaraq şərh edilmiş hissə standart sazlanmada yaşıl rəngə boyanır);
- simvolik (işarələr, ədədi olmayan kəmiyyətlər) qiymətlər ikiqat dırnaq arasına işarələrlə (məsələn, belə: "**simbolik_value**") haşiyələnmişdir;
- VBA-da istənilən ad (dəyişənin, sabitin, proseduranın və ya funksiyanın v.s.) 255 işarədən artıq ola bilməz;
- yeni operatorun başlanğıcı – C++, Java, JavaScript kimi dillərdən fərqlənir (bu dillərdə yeni sətərə keçid etmək üçün kod sətirinin sonunda (;) işarəsi yazılmalıdır). VBA-da bunun üçün sadəcə VBA kod sətirinin sonunda heç bir işarənin qoyulmaması və klaviaturanın **ENTER** düyməsinin basılması kifayətdir;
- kod sətirinin uzunluğuna heç bir limit qoyulmur (nəzərə alınmalıdır ki, VBA redaktorunun kod yazma pəncərəsində yalnız 308 işarənin yazılması mümkündür);
- bir neçə operator bir sətirdə yazılırsa, onda onlar bir-birilə ikiqat nöqtə işarəsi ilə (:) ayırırlar, məsələn, bu cür:

```
MsgBox "Yoxlama 1" : MsgBox "Yoxlama 2");
```

- proqramlaşdırma kodunun oxunmasını asanlaşdırmaq üçün bir neçə sətiri məntiqi olaraq bir sətir kimi birləşdirmək olar: bunun üçün növbəti sətirin sonunda boşluq işarəsi (_) qoyulmalıdır. Məsələn, aşağıdakı nümunədəki kimi:

`MsgBox "İstifadəçinin adı" _ vUserName`

3.4 VBA operatorları: hesabi, məntiqi, müqayisə və mənimsəmə

Operator – VBA kodunun ən kiçik icra edilə bilən vahididir. Operator aşağıdakı işləri yerinə yetirə bilər:

- dəyişənin elan edilməsini;
- dəyişənin təyin edilməsini;
- VBA kompilyatorunun qurulmasını;
- proqramda hansısa əməliyyatın icra edilməsini.

VBA-da olan operatorlar cədv. 3.1-də verilib:

Cədvəl 3.1 VBA operatorları

Operatorun tipi	İşarəsi	Yerinə yetirdiyi əməl
Hesabi	+	Toplama
	-	Çıxma
	*	Vurma
	/	Bölmə
	\	Tam bölmə
	^	Qüvvətə qaldırma
	Mod	Bölmədən qalan qismət
Sətir	&	Birləşmə
Məntiqi	AND	VƏ
	OR	VƏ YA
	NOT	YOX
Müqayisə	=	Bərabərdir
	<	Kiçikdir
	>	Böyükdür
	< =	Kiçikdir bərabərdir
	> =	Böyükdür bərabərdir
	< >	Bərabər deyil

VBA-da cəmi 7 hesabi operator vardır. Onlardan dördü standart kateqoriyalıdır: toplama (+), çıxma (-), vurma (*), bölmə (/). Və daha üçü bunlardır:

qüvvətə yüksəltmə (^), məsələn, $2^3=8$;

tam ədədli bölmə (\), kəsr hissəni yuvarlaşdıraraq (atmayaraq) birinci ədədi ikinciye bölür, məsələn, $5 \setminus 2=2$;

modula görə bölmə (Mod), bölmədən alınan ədəddə yalnız qalığı saxlayaraq, birinci ədədi ikinciye bölür, məsələn, $5 \text{ Mod } 2=1$.

VBA-da mənimsemə operatoru – bərabərlik işarəsi ilə (=) həyata keçirilir, məsələn:

```
Let nVar=10
```

yaxud daha sadə formada

```
nVar=10
```

İkinci halda isə bərabərlik işarəsini bərabərlik operatoru ilə qarışdırmaq olmaz, çünki `nVar=10` ifadəsi bunu göstərir ki, “`nVar` dəyişəninə 10 qiyməti verilir” (yaxud başqa sözlə “`nVar` dəyişəni 10 qiymətini mənimləyir”. Məsələn, başqa halda

```
If (nVar=10)
```

bu deməkdir ki, “əgər `nVar` dəyişənin qiyməti 10-a bərabədirsə”.

Əgər dəyişən üçün bir obyekt təyin edilsə, onda bu başqa üsullarla həyata keçirilir (kitabın sonrakı hissələrində bu haqda daha ətraflı danışılacaq).

VBA-da cəmi 8 müqayisə operatoru vardır:

- bərabərlik (=), məsələn, `If (nVar=10)`;
- böyükdür (>), kiçikdir (<), məsələn, `If (nVar>10)` və ya `If (nVar<10)`;
- bərabər deyil (<>), məsələn, `If (nVar<>10)`;
- obyektlərin müqayisəsi (`Is`), obyekt dəyişənlərinin həmin obyektlərə yaxud müxtəlif obyektlərə istinad etməyini müəyyən edir. Məsələn, `If (obj1 is obj2)`;
- oxşarlıq (`Like`), sətir obyektini şablonla müqayisə edir və şablonun uyğun gəlib gəlməməsini müəyyən edir.

Müqayisə operatorları həmişə bu məntiqi qiymətləri qaytarırlar: `True` (doğru) əgər iddia gerçəkdirsə, və ya `False` (yalan), əgər iddia yalandırsa.

Sətir dəyişənlərinin müqayisəsi üçün bunları bilmək vacibdir:

- registr nəzərə alınır (böyük və ya kiçik işarələr);
- sətirlərdə olan boşluqlar nəzərə alınır;
- sətir dəyişənlərinin böyük/kiçik müqayisəsində standart halda sadəcə işarələrin ikili kodları müqayisə edilir – hansılar böyükdür və ya kiçikdir. Əgər əlifbadakı ardıcılıqdan istifadə edilsə, onda `OptionCompareText` əmrindən istifadə etmək olar.

Daha bir vacib olan operator, yuxarıda adı çəkilən, `Like` operatorudur. Onun ümumi sintaksisi belədir:

```
İfadə1 Like İfadə2
```

Bununla yanaşı `İfadə1` – istənilən mətn ifadəsidir, `İfadə2` ifadəsi isə `Like` operatoruna ötürülən şablondur. Bu şablonda xüsusi əvəz etmə işarələrindən istifadə etmək olar (cə. 3.2).

Cədvəl 3.2 **LIKE** operatorunun xüsusi əvəz etmə işarələri

ƏVƏZ ETMƏ İŞARƏLƏRİ	QIYMƏTLƏRİ
#	İstənilən ədəd (0, ..., 9 sırasından yalnız biri)
*	İstənilən işarələrin istənilən sayı (sıfır qiymətliləri daxil edərək)
?	İstənilən işarə (yalnız biri)
[a , b , c]	Verilən siyahıdan istənilən işarə (yalnız biri)
[! a , b , c]	Verilən siyahıdan başqa istənilən işarə (yalnız biri)

Məntiqi operatorlar (daha çox birdən artıq olan şərtlərin yoxlanmasında istifadə edilir):

- **AND** – məntiqi VƏ (hər iki şərt doğru olmalıdır);
 - **OR** – məntiqi YAXUD (hansısa şərtin birisi doğru olmalıdır);
 - **NOT** - məntiqi inkar (şərt yalan olduqda məntiqi **TRUE** qiymətini qaytarır);
 - **XOR** - məntiqi istisna ($E1 \text{ XOR } E2$ ifadəsində əgər yalnız $E1=\text{TRUE}$, və ya yalnız $E2=\text{TRUE}$ doğrudursa, onda **TRUE** məntiqi qiyməti qaytar – digər halda **FALSE** məntiqi qiyməti qaytarır);
 - **EQV** – iki ifadənin ekvivalentliyi (əgər onların qiymətləri eynidirsə, onda **TRUE** məntiqi qiyməti qaytarır);
 - **IMP** – implikasiya (əgər $E1=\text{TRUE}$ və $E2=\text{FALSE}$ doğrudursa, onda **FALSE**, məntiqi qiyməti qaytarır – digər halda **TRUE** məntiqi qiyməti qaytarır);
 - **AND** – məntiqi VƏ;
 - **OR** – məntiqi YAXUD;
 - **NOT** – məntiqi YOX;
 - **→+ və ya &** - konkatenasiya operatoru (demək olar ki, hər bir VBA proqramında istifadə edilir). Məsləhət görülür ki, daha çox **&** işarəsi istifadə edilsin, çünki bu halda, avtomatik olaraq, ədədi qiymətlər sətir qiymətlərə çevrilir – səf etmənin ehtimalı yoxdur. **+** operatoru istifadə edildikdə sətir qiymətinin **Null** qiymətilə cəmi **Null** nəticəsini verir). Nümunə: `MsgBox "Message to user" & vUserName.`

3.5 VBA dəyişənləri və verilənlərin tipi

VBA dəyişənləri, dəyişənlərin elanı, Option Explicit, adlandırma qaydaları, VBA verilənlərinin tipi, VBA dəyişənlərinin ilkin qiymətləri

Dəyişənlər – dəyişən qiymətlərin konteyneridir (“daşıyıcısıdır”). Praktiki olaraq, dəyişənlərsiz heç bir proqram qurula bilməz. Sadə anlam üçün – dəyişənlər kiminsə öz dəyərli əşyasını (ola bilsin pulunu) bankda saxlanması üçün təhvil verməsinə oxşayır: müştəri öz əşyasını (verilənləri) verir, əvəzinə etibar ediləsi nömrələnmiş çekini alır. Geriyə əşyasını (verilənlərini) almaq lazım olduqda isə müştəri həmin çeki banka təqdim edir və əvəzinə öz əşyasını (pulunu) geri alır. Həmin prinsipə də uyğun qaydalar proqramlaşdırmada verilənlər anlayışında istifadə edilir.

VBA alqoritmik dilinin əsas tipləri və standart riyazi funksiyaları

Aşağıda, cədv. 3.3-də MS VBA verilənlərinin tipləri haqqında ətraflı qısa məlumat verilib.

Cədvəl 3.3. VBA verilənlərinin tipi ölçüsü və qiymətləri

Tip	İzahı	Ölçüsü [bayt]	Qiymətlər diapazonu
Integer	Tam ədəd	2	- 32768, ..., 32767
Long Integer	Uzun tam ədəd	4	2147483648, ..., 2147483647
Single-division Floating point	Vahid dəqiqliklə və dəyişən onluq nöqtə ilə	4	- 3.402823E38, ..., 3.4402823E38
Double-division Floating point	İkiqat dəqiqliklə və dəyişən onluq nöqtə ilə	8	-1.79769313486232D308, ..., 1.79769313486232 D308
Currency	Pul vahidi	8	-922337203685477,5808, ..., 922337203685477.5807
String	Sətir vahidi	bir işarə	Dəyişənin uzunluğu: 2 billion; dəyişənin fiksə edilmiş uzunluğu: 0, ..., 65535 işarə
Single	Yeganə dəyişən (vahid dəqiqlikli üzən nöqtəli)	4	IEEE-də saxlanılır dəyişmə diapazonu: mənfi qiymətlər üçün -3.402823E38-dan - 1.401298E-ə qədər, müsbət qiymətlər üçün: 1.401298E -45-dan 3.402823E38-ə qədər.
Boolean	Məntiqi	2	True (doğru) yaxud False (yalan)
Date	Tarix	8	January (Yanvar) 1,100, December (Dekabr) 31,9999
Variant	Dəyişən	16 (ədədlər üçün); 22+1 bir simvola (sətirlər üçün)	Bütün tiplər üçün
Object	VBA obyektlərinin tipi (obyektə olan istənilən istinad)	4	
User-Defined	İstifadəçi tərəfindən tərtib edilir	Müxtəlif	İstənilən diapazonda (VBA qaydalarına uyğun)

Dəyişənlərin elan edilməsinin ümumi sintaksisi VBA-da bu cürdür:

```
Dim Dəyişənin_adı As Dəyişənin_tipi
```

VBA dəyişənlərinin işləməsinə aid sadə VBA fraqmentinin nümunəsi aşağıdakı kimi ola bilər:

```
nMyAge=nMyAge+10
```

```
MsgBox nMyAge
```

Dəyişənlərlə işləməkdən öncə onların elan edilməsi vacib mərhələdir. Yuxarıda gətirilən misal üçün dəyişənin elanı bütün kod sətirlərindən əvvəl bu cür yazıla bilər:

```
Dim nMyAge As Integer
```

Gəlin həmin elan sətirini analiz edək:

Dim – dəyişənin “görükmə” sahəsidir. Bu operator ingilis sözü “*dimension*”, yəni azərbaycanca ölçü (nəyinsə ölçüsü, bizim halda dəyişənin ölçüsü) sözünün qısaltmasından əmələ gəlib. VBA-da dəyişənlərin ölçüsünü bildirən 4 açar kateqoriyalı söz vardır:

Dim - bütün hallarda istifadə edilir. Əgər dəyişən proqramın elanlar hissəsində **Dim** kimi elan edilibsə, onda o bütün modulda əlçatan olacaqdır, prosedurada elan edilibsə, yalnız prosedura işlədiyi zaman müddətində əlçatan olacaqdır;

Private — VBA dəyişənləri elan edildikdə **Dim** kimi məna daşıyır (İngilis dilində “*private*” - xüsusi deməkdir, deməli həmin dəyişən əgər bu cür elan edilsə, onda başqa VBA layihələrində görünməz olacaq);

Public - dəyişən bu təlimatla elan edilsə o layihənin bütün prosedura və modullarında “görünən”, əlçatan olacaq (İngilis dilində “*public*” – ictimai deməkdir, yəni bu cür dəyişən elanı dəyişənləri əlçatan vəziyyətə gətirir). Əgər dəyişəni **Public** statusu ilə proseduranın içərisində etmişdilsə, onda o özünü tamamilə **Dim/Private** elanına uyğun aparacaq;

Static - bu təlimatla elan edilən dəyişən yalnız proseduranın içərisində (elan edildiyi prosedurada) istifadə edilə bilər. Bununla belə həmin dəyişənlər, proseduranın müxtəlif zamanda çağırılmasından asılı olmayaraq, öz qiymətlərini saxlayırlar (İngilis dilində “*static*” sözü dəyişməz mənasını daşıyır). Adətən dəyişənlər hansısa qiymətlərin yığılıb saxlanılmasında istifadə edildikdə bu təlimatla elan edilir, məsələn:

```
Static nVar1 As Integer
```

```
nVar1=nVar1+1
```

```
MsgBox nVar1
```

Əgər proqramlaşdırmada dəyişənlərin xassələrinə aid xüsusi tələblər yoxdursa, onda **Dim** elan təlimatından istifadə edilməsi, praktiki olaraq, hər vaxt məqsədə uyğun sayılmalıdır.

Yuxarıda gətirilən sadə VBA nümunələrində `nMyAge` - identifikatordur (sadə dillə desək dəyişənin adıdır). VBA-da dəyişənlərin (müxtəlif elementlər üçün: dəyişənlər, sabitlər, funksiyalar, proseduralar v.s.) adının seçilməsində vahid bir qaydalar toplusu vardır:

- ad işarədən (hərfdən) başlanmalıdır, ədəddən başlamaq qadağandır;
- adda boşluqlar (İngiliscə “space”) və punktuasiya işarələri olmamalıdır. Yalnız (`_`) işarəsinin qoyulmasına icazə verilir, məsələn: `Natiqin_proseduru_5` və ya `diskret_param_32` ;
- adın maksimal uzunluğu 255 işarədən çox olmamalıdır;
- cari “görükmə” sahəsində unikal olmalıdır (bu haqda aşağıda daha ətraflı danışılacaq);
- VBA-nın rezerv (“açar”) sözlərindən adlandırmada istifadə etmək qadağandır.

VBA proqramlarını yaratdıqda təkidlə məsləhət görülür ki, obyektlərə ad vermə qaydalarından istifadə edilsin – adlandırma sazişi (razılaşması). Adətən əksər hallarda, “macar razılaşması” kimi tarixdə qalan adlandırma sazişindən istifadə edirlər (Microsoft-un *Charles Simonyi* adlı macar mənşəli proqramçının şərəfinə):

- dəyişənin adı prefiksdən sətir hərfləri ilə (kiçik hərflərlə) başlanmalıdır. Prefiks həmin dəyişənin hansı qiymətləri saxladığına işarə etməlidir (cə. 3.4 və cə. 3.5);

Cədvəl 3.4 VBA verilənlərin tipi ölçüsü və qiymətləri

Adlandırma işarələr kombinasiyası	Mənası
<code>str</code> (yaxud <code>s</code>)	String - işarə qiymətləri
<code>fn</code> (yaxud <code>f</code>)	Funksiya
<code>c</code> (yaxud bütün hərfləri böyük etmək)	Sabit
<code>b</code>	Boolean , məntiqi qiymət (True və ya False)
<code>d</code>	Tarix
<code>obj</code> (yaxud <code>o</code>)	Obyektə istinad
<code>n</code>	Ədədi qiymət

Cədvəl 3.5 VBA-da dəyişənlərin tipini ifadə edən işarələr.

İşarə	Dəyişənin tipi (VBA terminologiyası)	Mənası
<code>%</code>	Integer	Tam ədəd
<code>&</code>	Long	Uzun tam ədəd
<code>!</code>	Single	Yeganə (vahid dəqiqlikli üzən nöqtəli) dəyişən
<code>#</code>	Double	İkiqat dəqiqliklə və dəyişən onluq nöqtə ilə
<code>@</code>	Currency	Pul vahidi ilə ölçülən dəyişən
<code>\$</code>	String	Sətir dəyişəni

- funksiyaların, metodların və mürəkkəb tərkibli sözlərin hər bir sözü böyük hərfdən başlanmalıdır, məsələn:

MsgBox objMyDocument.Name

Sub CheckDateSub ()

Bundan əlavə VBA-da istifadəçi tərəfindən yaradılan tiplərdən faydalanmaq mümkündür. Bu halda əvvəlcədən onların **Type** təlimatı ilə dəyərləndirilməsi vacibdir. Adətən “istifadəçi tipləri” istifadəçi tərəfindən daxil ediləsi qiymətlərin yoxlanılmasında lazım olan əlavə vasitə kimi tətbiq edilir (klassik misal kimi poçt indeksini gətirmək olar).

Dəyişənlərin tipinin seçilməsi ilə bağlı ən vacib olan məqamları qeyd edək:

- ümumi prinsip – nəzərdə tutulan qiymətləri özündə saxlayan verilənlərin mümkün qədər ən kiçik tipinin seçilməsi sərfəlidir. Şübhəli hallarda, səhvlərin qarşısını almaq üçün, verilənlərin ən böyük tipinin seçilməsi də məqsədə uyğundur;
- mümkün qədər verilənləri üzən nöqtə tipindən istifadə etməsindən imtina edilməlidir (**Single** və **Double**). Bu cür verilənlərlə işləyən proseduralar adətən daha yavaş işləyir (bundan əlavə müqayisə etmələrdə yuvarlaşdırma ilə bağlı ciddi problemlər meydana çıxıb bilər);
- əgər imkan varsa **Variant** tipindən daha az istifadə edilməsi məsləhətlidir, çünki VBA-da dəyişən tamam başqa tipə gətirilə bilər, o biri tərəfdən bu növ tiplər üçün daha böyük yaddaş tələb olur (həm də dəyişənlər bu cür, “qeyri aşkar”, elan edildikdə dolaşmalar və xətalər qaçılmazdır);
- dəyişənləri təyin etdikdə tipin təyin etmə işarələrindən istifadə etmək olar (% - **Integer**, \$ - **String** v.s.), məsələn, gətirilən nümunədə birinci sətiri **Dim nVar1 As Integer** şərh formasına gətirərək, ikinci sətiri aşağıdakı kimi yazmaq olar:
- `nVar1%=nVar1%+1`

Bu cür yaxınlaşma artıq köhnəlib və VBA-nın yeni versiyalarında qadağandır.

Dəyişənlər elan ediləndə onların tipləri göstərilməyə də bilər. Məsələn, bizim nümunədə elan bu cür ola bilər:

Dim nVar1

Bu halda dəyişən, avtomatik olaraq, **Variant** tipi ilə elan ediləcəkdir. Prinsip etibarı ilə VBA-da dəyişənləri elan etmədən də proqramı qurmaq olar. Məsələn, aşağıdakı misaldakı VBA koduna nəzər salmaq:

```
nVar1=nVar1+1
```

MsgBox nVar1

Bu proqram fraqmenti də, tam olaraq, işlək vəziyyətdə olacaq. Əgər istifadəçi dəyişənini elansız işlədirsə, onda, avtomatik olaraq, ona **Variant** tipi verilir. **Bununla belə təkidlə məsləhət edirik ki, VBA kodunun bütün dəyişənlərini əvvəlcədən elan etmək vacibdir!**

Həmçinin tiplərin aşkar elanı daha vacibdir:

- səhvlərin sayı azalır: proqram işləməyə başlayan andan dəyişənin tərkibinə tipi uyğun olmayan qiymətləri qəbul etməyəcək (məsələn, ədədi tip əvəzinə sətir tipini);
- obyektlə işlədikdə xassələr və metodlarla bağlı olan məsləhətlər yalnız obyekt dəyişəninin tipi elan edildikdə fəaliyyət göstərir. Məsələn, Excel-də iki variant eyni işləyəcək:

birinci variant:

```
Dim oWbk As Workbook
Set oWbk=Workbooks.Add()
```

ikinci variant:

```
Set oWbk=Workbooks.Add()
```

Burada nəzərə alınmalıdır ki, `oWbk` obyektinə üzrə xassələr və metodlar ilə bağlı kömək edən bələdçi yalnız ikinci variantda işləyəcək.

Ümumiyyətlə, oxucu bu faktı bilməlidir ki, bütün peşəkar proqram tərtibatçıları dəyişənləri aşkar olaraq elan edilməsinə qadağa qoyurlar. Bunun üçün kompilyatorun xüsusi əmrindən faydalanmaq olar (həmin əmr yalnız modulun elanlar hissəsində yerləşir):

Option Explicit

Bu təlimat, modulların yaradılmasına başlarkən, artıq kodlaşdırma sətirində, avtomatik olaraq çap edilmiş halda olur – bunun üçün kod redaktoru pəncərəsində **Require Variable Declarations** qarşısında bayraqcığı qurmaq lazımdır (**Editor** quraşdırmasında **Tools** → **Options** menyusunu). Bu əməliyyatı peşəkar proqramçılar nəyə görə etdiyini aşağıdakı nümunədən anlamaq çətin deyil:

```
Dim n
n=n+1
MsgBox n
```

Birinci baxımda yuxarıda verilən VBA kodu heç bir problem yaratmamalıdır: dialog pəncərəsində `n` dəyişənin cari qiymətlərini göstərməlidir. Əslində proqram boş (məlumatsız olan) məlumat pəncərəsini canlandırır. Səbəbi isə çox məkrlidir: kodun üçüncü sətirində yerləşən `n` - əslində İngilis `N` yox Rus `П` hərfinin işarəsidir. Redaktor pəncərəsində onları bir birindən seçmək olduqca çətin deyil. VBA kompilyatoru isə bu cür koda rast gəldikdə, avtomatik olaraq, boş qiymətli və `variant` tipli yeni dəyişən yaradacaq. Bu cür səhvin təyin edilib tapılması isə əslində çox vaxt tələb edə bilər.

Ən yaxşı qayda budur: dəyişənlərin tipini lazım olduqda yox, proqramlaşdırmanın başlanğıc zamanında elan etmək lazımdır. Bu cür qaydaya əməl edilməsi istifadəçi proqramının planlaşdırılmış və yüksək dərəcədə oxunulma qabiliyyətinə malik olmasını təmin edir.

Bir neçə dəyişənin bir sətirdə elan olması VBA-da mümkündür:


```
Dim n1 As Integer, s1 As String
```

Dəyişənlərin qiymətləri mənimsəmələri bu cür həyata keçir:

```
nVar1=30
```

əgər dəyişənin cari qiyməti hansısa qiymət səviyyəsində artmalıdırsa, onda kodlaşma əmri belə ola bilər:

```
nVar1=nVar1+635
```

Hər iki nümunədə bərabərlik işarəsi “bərabər” deyil mənimsəmə mənasını daşıyır.

VBA-da verilənlərin obyekt tipi

VBA-da verilənlərin əlavə tipləri də var ki, onlar obyekt dəyişənləri tipinə aiddir. Onların daha geniş siyahısını oxucu kitabın Excel (Word, Access v.s.) ilə bağlı fəsilərində yaxud daha da mükəmməl bilmək üçün VBA-nın Office proqramlarındakı VBA sənədləşməsində tapa biləcək. Buna baxmayaraq, obyekt dəyişənlərinin ən vacib olan tipləri haqqında bəzi məlumatlar aşağıda verilib:

Diagramlarla bağlı obyekt tipləri (Excel): Axis, AxisTitle, Chart, ChartArea, ChartColorFormat, ChartTitle, DataLabel, DataTable, Floor, GridLines, Legend, LegendEntry, LegendKey, PlotArea, Point, Series, SeriesCollection, TickLabels, Walls

Yığılma (PivotTables) ilə bağlı obyekt tipləri (Excel): PivotCache, PivotField, PivotFormula, PivotItem, PivotTable

Ümumi təyinatlı obyekt tipləri (bütün Office proqramlarında): Comment, Font, FillFormat, Outline, Filter PageSetup, Range, Sheets, Window, Workbook, Worksheet, WorksheetFunction

Obyekt dəyişənlərinin tipinin elan edilməsinə aid aşağıda bir neçə nümunə verilib:

```
Dim wb As Workbook
Dim wks As Worksheet
Dim chrt As Chart
Dim ax As Axis
Dim pf As PivotField
```

Dəyişənlərə ad verdikdə aşağıdakı qaydalara riayət edilməsi olduqca vacibdir:

- sətir qiymətləri hökmən ikiqat dırnaq arası işarələrlə haşiyələnir, məsələn:

```
sVar1="Hello";
```

- tarix/zaman qiymətləri hər iki tərəfdən (#) işarəsi ilə haşiyələnməlidir, məsələn:

```
dVar1=#05/06/2004#
```

- diqqət verilməlidir ki, tarix/zaman qiymətləri bu cür “aşkarcasına” verildikdə istifadəçi gerek ABŞ-da qəbul olan standartda riayət etsin: yuxarıdakı nümunədə 05 ayı göstərir,

06 isə günü. Həmin qiymətin dialoq pəncərələrində göstərilməsi isə istifadəçinin kompyuterində regional sazlanmadan asılı olacaqdır;

- onaltılıq qiymətlərin ötürülməsi lazım olduqda, işarənin qarşısında **&H** simvolları qoyulmalıdır:

```
nVar1=&HFF00
```

- dəyişənlərin tərkibində mənimsəmədən öncə hansı qiymətlərin olduğuna nəzarət edilməlidir;
- dəyişən uzunluğu olan sətir dəyişənlərində "" (sıfır uzunluqlu sətir) nəzarət edilməlidir;
- fiksə olmuş sətir dəyişənlərində - **ASCII** simvolu kateqoriyasına aid və verilən uzunluqlu sətirin qiyməti 0 olmalıdır (bu simvollar ekrana çıxarılmır);
- **B Variant** - boş qiymət;
- **B Object** - obyektlər bir-birinə istinad etmir.

3.6 VBA-da sabitlər

Sabitlər, elan edilməsi, Const açar sözü, qurulmuş standart sabitlər, vbCrLf

Sabitlər – VBA-da verilənlərin saxlanması üçün nəzərdə tutulmuş daha bir konteyner nümunəsidir. Dəyişənlərdən fərqli olaraq, VBA proqramı işlədikdə onların qiyməti dəyişməz qalır. Sabitlərin əsasən aşağıda göstərilən amilləri tərtib edilən proqramlarda olduqca vacibdir:

- proqram kodunu daha yaxşı oxunma qabiliyyətinə çatdırır (potensial səhvlərin aradan qaldırılması təmin olur);
- proqramda dəfələrlə istifadə edilən hansısa kəmiyyətin (məsələn, vergi səviyyəsi və ya fiziki modeldə sabit əmsal) qiymətinin dəyişməsi yalnız bir dəfə lazım olduqda (proqramın işləməsindən qabaq bunu etmək olar).

VBA-da sabitlər **Const** açar sözü ilə, tipi göstərilərək, elan və təyin edilir:

```
Const COMP_NAME As String = "Microsoft"
```

Başqa daha sadə forma ilə də sabitlər elan edilə bilər (tipi göstərilmədən), məsələn:

```
Const pi=3.14
```

Diqqət! Sabitin qiymətini proqramın bədənində dəyişdirmək cəhdi olduqda, dərhal VBA kompilyatoru səhv haqqında cavab verəcək. Sabit olan qiyməti iki vəziyyətdə dəyişmək olar: proqram işləmədən redaktə rejimində, yaxud proqram işləməyə başlarkən (əgər verilənlərin qiymətləri avtomatik rejimdə istifadəçi tərəfindən daxil edilirsə).

Sabitlər adlandırılmış elementlər qrupu ilə (həftə günlərinin adları, rənglər, klavişlər, avtomobillərin markaları v.s.). Proqramın kodunda olan çətin yadda qalan ədədi kodlar əvəzinə sabitlərin köməyi ilə asanlıqla oxunan işarələr işi xeyli sərfəli və məhsuldar edir. Məsələn, aşağıdakı koda nəzər edin:

```
UserForm1.BackColor=vbGreen
```

və

```
UserForm1.BackColor=65280
```

Funksional olaraq, onlar eynidir - birinci sətirdə yazılan kodun mənasını dərhal anlamaq olur (çünki birinci sətirdəki kodda `vbGreen` adlı verilən sabitdir).

VBA-da bir çox xidməti (qurulmuş) sabitlər vardır: kalendar tipli, fayllarla, rənglərlə, formalarla, disklər növləri v.s. ilə işləmək üçün. Onlarla tam müfəssəl VBA-nın sorğu sistemindən məlumat almaq olar: **Microsoft Visual Basic Documentation** → **Visual Basic Reference** → **Constants**. Onların biri haqqında xüsusi demək lazımdır (o sorğu materialların **Miscellaneous** hissəsində də yerləşir): `vbCrLf` sabiti yeni sətərə keçməni təmin edir. Məsələn:

```
MsgBox("First line" + vbCrLf + "Second line")
```

Bir çox sabitlər isə obyekt modellərində qurulmuş halda olur (daha ətraflı oxucu kitabın son fəsilərində onlarla tanış olacaq).

3.7 VBA-da çoxluqlar (arrays)

Çoxluq (İngilis terminologiyasında - array) dedikdə VBA-da bir tipə aid olan elementlərdən ibarət (elementləri tutduqları yerə görə nömrələnməsi vacib amildir) qrupa deyirlər.

Proqramlaşdırma sahəsində İngilis ədəbiyyatında çoxluqları kompakt formada ifadə edən kəmiyyətə *Array* deyirlər (Rus ədəbiyyatında - *Массив*). Azərbaycan dilində çoxluq dedikdə oxucu daha çox riyaziyyat kursunda olan *matris* terminini sinonim söz kimi götürsə daha yaxşı anlayış yarana bilər.

Çoxluqlar və onların elementləri kompyuterin yaddaşında saxlamaq və lazım olduqda üzərində əməllər aparmaq üçün istifadə edilir. Eyni xassəli elementləri bir-bir elan etmək və ya üzərində əməllər aparmaq sərfəli və məhsuldar iş sayıla bilməz. Buna görə çoxluqların (matrislərin) proqramlaşdırmada tətbiqi çox genişdir.

VBA sorğu materiallarında çoxluq (*array*) üçün gətirilən tərif belə səslənir: “Ardıcıl indekslənməmiş və eyni xassəli verilənlər tipinə məxsus olan elementlər sırasına çoxluq (*array*) deyirlər. Həmin çoxluğun hər bir elementi yeganə identifikasiya nömrəsinə malikdir. Onun üzərində hər hansı əməl aparılsa başqa elementlərə təsir olmur.”

Çoxluqların elan etmə sintaksisi belədir:

Sintaksis:

Array(arq_sırası)

Burada arq_sırası dəyişən (**Variant**) tipinə aid olan və bir biri ilə vergüllə ayrılan çoxluğun elementlərinin sırasıdır.

Onların adları indeksə (nömrələnməyə) görə fərqlənir. Məsələn, Office proqramının birinci səhifədəki 100-cü hücrəni (Excel-də görüntülü olduğuna görə hücrə termini kimi xana termini də istifadə edilə bilər) saxlayan, tutaq ki, `Cell1d` adlı dəyişən aşağıdakı nümunədə göstərilən kimi elan edilə bilər:

```
Dim Cell1d To 100) As Range
```

Excel obyekt modelində `Cell` adlı obyekt yoxdur. Bunun üçün `Range` obyektindən çoxluğun indekslərinin aşağı limiti 1 bərabərdir, yuxarısı isə 100. Buna görə aşağıdakı dəyişənlər `Range` dəyişəninin elementləridir: `Cell(1)`, `Cell(2)`, ..., `Cell(100)`

Elan etdikdə çoxluğun ölçüsünün verilməsi

Əvvəlki nümunədə təsvir edilən, `Cell` çoxluğu bir ölçülüdür. Əslində daha böyük ölçülü çoxluqlara rast gəlmək olur, məsələn:

```
Dim Cell(1 To 10, 1 To 100) As Range
```

`Cell` çoxluğu iki ölçülüdür. Onun birinci indeksi 1-dən ... 10-a qədər dəyişir, ikincisi isə 1-dən ... 100-ə qədər dəyişir. Buna görə çoxluğun elementlərinin sayı $10 \times 100 = 1000$ olacaq.

VBA-da çoxluqlar 60 ölçüyə malik ola bilərlər. Praktiki məsələlərdə bu qiymətdən artıq ölçülü çoxluqlar istifadə edilmir, çünki müasir maşınların operativ yaddaşı üçün belə halda daha çox yer lazım ola bilər.

Dinamik çoxluqların elan edilmə qaydası

Əgər çoxluq aşağıdakı kimi elan edilərsə:

```
Dim FileName(1 To 10) As String
```

Onda onun ölçüsü fiksə (müəyyən) edilmişdir, çünki indekslərin yuxarı və aşağı sərhədləri verilib. Office proqramlaşdırmasında tez-tez elə vəziyyət yaranır ki, çoxluqlar elan edildikdə onların ölçülərini əvvəldən müəyyən etmək mümkün olmur. Bunun üçün VBA-da dinamik dəyişən çoxluqlar tipi vardır və bunun üçün `ReDim` operatorundan istifadə edirlər. Dinamik çoxluqlar (arrays) işarəsi boş olan dairəvi mötərizələrlə verilir:

```
Dim FileName() As String
```

Sonra proqramda yenidən həmin çoxluq yenidən elan edilə bilər:

```
ReDim FileName(1 To 10)
```

Nəzərə alınmalıdır ki, ölçünün bu üsul ilə dəyişdirildiyində onun tərkibində olan elementlər itirilə bilər. Bunun qarşısını almaq üçün `Preserve` operatorundan istifadə etmək lazımdır:

```
ReDim Preserve FileName(1 To 200)
```

Belə olduqda istifadəçi yalnız çoxluğun yuxarı ölçüsünü və çoxölçülü çoxluğun son ölçüsünü dəyişdirə bilər.

Bəzən də dinamik çoxluqlar lazım olur – yeni elə çoxluqlar ki, proqramın işlədiyi zaman onların ölçüləri dəyişdirilə bilər. Onların elanı aşağıdakı nümunədəki sintaksis əsasında verilir:

```
Dim MyArray()  
ReDim MyArray(4)
```

Birinci sətirdə `MyArray()` adlı çoxluq ölçüsünün yuxarı limiti verilmədən elan edilir. İkinci sətirdə isə çoxluğun ölçüsü dəyişdirilir. `ReDim` əmri çoxluğun ölçüsünün dəyişdirilməsindən əlavə üstəlik həmin çoxluqdakı köhnə qiymətləri silir. Köhnə qiymətləri saxlamaq üçün `Preserve` açar sözü istifadə edilir:

```
ReDim Preserve MyArray(7)
```

Əgər çoxluğun yeni ölçüsü tərkibində olan elementlərin sayından az olsa, `Preserve` açar sözü kömək edə bilməz – verilənlərin bir hissəsi itiriləcək. Aşağıdakı nümunədə üç elementdən ibarət A çoxluğunun elan edilməsi və sonra B çoxluğunun A çoxluğunun 2 elementli mənimsəməsi verilib:

```
Dim A As Variant  
A=Array(10,20,30)  
B=A(2)
```

Digər nümunədə tam ədəd verilənlər tipinə aid iki elementli `MyArray` çoxluğunun elanı verilib:

```
Dim MyArray(2) As Integer
```

Bu cür çoxluq üç sayda tam ədədli elementi tərkibində saxlaya bilər. 2 – çoxluğun ölçüsünün (elementlərinin sayının yuxarı limitini (*upper bound*) göstərir. Standart halda VBA-da çoxluqların saxladığı elementlərin sayı - 0-dan yuxarı limitə (*upper bound*) qədərdir. İstifadəçi istədikdə bu xassəni dəyişə bilər, məsələn, çoxluğun elementlərinin nömrələnməsi 1-dən başlanması əlverişli olsa, onda bunun üçün o gərəkdir.

```
Option Base 1
```

əmrini proqramda kod sətirlərinin çoxluqlar elanından əvvəl yazsın.

Prinsip etibarı ilə çoxluğun tipi VBA-da elan edilməyə də bilər:

```
Dim MyArray(2)
```

Bu halda çoxluğun elementlərinə avtomatik olaraq, `variant` tipi verilir. Belə çoxluqlar müxtəlif tipli elementləri tərkibində saxlaya bilər - bu halda yaddaşa tələb daha yüksək olacaq və proqram buna görə bir az yavaş işləyə bilər. Çoxluğun hansısa elementinə müəyyən qiyməti aşağıda göstərilən nümunədəki üsul ilə verilməsi VBA-da mümkündür:

```
MyArray(0)=100
```

Sonra isə həmin qiyməti bu cür çıxarmaq olar:

```
MsgBox MyArray(0)
```

Çoxluqların ölçüsü çox qatlı ola bilər (yada salırıq ki, VBA-da çoxqat ölçülü çoxluqların yuxarı limiti 60-dan böyük ola bilməz). Aşağıda matris anlayışına uyğun olaraq, iki qat ölçülü çoxluq 4 sayda sətir elementləri və 9 sayda sütun elementlərindən ibarətdir:

```
Dim MyArray(4,9)
```

Çoxqatlı çoxluqların hər sətirində yalnız bir obyektə aid verilənlərin saxlanması daha əlverişlidir (məsələn, əməkdaşın adı, unikal nömrəsi, telefon nömrəsi v.s.) - adətən 2 və ya 3-dən artıq ölçülü çoxqatlı çoxluqlar rast gəlmir. Çoxluqların yaradılması və daxil edilməsi (doldurulması) eyni vaxtda qurulmuş standart VBA funksiyası **Array()** ilə yerinə yetirilə bilər:

```
Dim MyArray  
MyArray=Array(100, 200, 300, 400, 500)
```

Aşağıdakı nümunədə 2 qat ölçülü çoxluq 5 sətir və 10 sütundan ibarətdir:

```
Dim sngMulti(1 To 5, 1 To 10) As Single
```

Bəzən də bir birinin içərisində yerləşdirilən **For...Next** dövrü hesablama operatorlar strukturundan istifadə etmək lazım olur. Aşağıdakı nümunədə (VBA sorğu materiallarından götürülüb və oxucu bu proqramı işlədə bilər) 2 ölçülü çoxluğun elementləri **single** tipi ilə daxil edilir:

```
Sub FillArrayMulti()  
    Dim intI As Integer, intJ As Integer  
    Dim sngMulti(1 To 5, 1 To 10) As Single  
    ' Fill array with values.  
    For intI=1 To 5  
        For intJ=1 To 10  
            sngMulti(intI, intJ)=intI*intJ  
            Debug.Print sngMulti(intI, intJ)  
        Next intJ  
    Next intI  
End Sub
```

VBA-da çoxluğun ölçüsü göstərilməyə də bilər – o, avtomatik olaraq, ötürülən elementlərin sayı ilə təyin olur. Çoxluğu **Erase** əmri ilə təmizləmək (silmək) olar:

```
Erase MyArray
```

Fiksə edilmiş uzunluğu olan çoxluq daha asan təmizlənir (elementləri silinir), dinamik çoxluq deinizializasiya edilir – onun ölçüsü yenidən inisializasiya edilməlidir (ölçüsü yenidən təyin edilməlidir). Dinamik çoxluqlarda elementlərin sayı adətən müəyyən edilə bilmir. Belə hallarda **UBound()** standart funksiyasından istifadə etmək lazımdır (əgər çoxluq bir ölçülüdürsə və ya istifadəçini birinci qatın ölçüsü maraqlandırarsa, onda ölçünü ötürmək lazım gəlmir):

UBound(NameArray[, ölçü])

Bunun əvəzinə daha effektiv yol var: MS Office proqramlarının obyekt modellərindəki kolleksiyalar istifadə edilir. Kolleksiyalar – eyni xassəli elementləri tərkibində saxlayan xüsusi konteynerlərdir. Məsələn, Word-də elementləri saxlamaq üçün **Documents** kolleksiyası vardır, Excel-də **Workbooks** (açıq kitablar) və **Worksheets** (kitabdakı səhifələr) v.s. Qeyd etməliyik ki, kolleksiyaların istifadəsi çoxluqların (Arrays) istifadəsindən daha əlverişlidir:

- onların ölçüsü daha böyükdür;
- kolleksiyalarda standart xassələr və metodlar toplusu nəzərdə tutulub (məsələn, yeni elementi əlavə etmək üçün **Add()** metodu);
- elementlərin sayı haqqında məlumat almaq üçün **Count** xassəsinin olması;
- lazımı elementə istinad almaq üçün **Item()** metodunun olması.

Kolleksiyalar ilə işləmək haqqında daha ətraflı məlumatı oxucu kitabın 4-cü fəslində tapacaq.

3.8 VBA-da hesablama proseduralarının idarəetmə operatorları

Proqramlaşdırmada hesablama prosedurasının alqoritmi mürəkkəb və ya sadə strukturlu ola bilər. Bu və ya digər halda proqramlaşdırma sahəsində idarəetmə operatorlarından istifadə edilməsi çox önəmlidir. Çünki hətta sadə strukturlu hesablamalarda belə (müəyyən hallarda), avtomatik olaraq, müdaxilə etmək və idarə etmək lazım olur. Mürəkkəb strukturlu proqramlarda isə həmin amillərin təsiri olduqca yüksək dərəcədə artır. Beləliklə, istifadəçi VBA dilində istənilən çətinliyə malik olan proqramları tərtib etdikdə hökmən bu dilin idarəetmə operatorlarını mükəmməl bilməlidir.

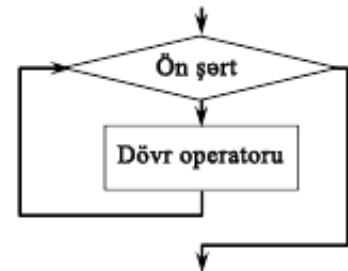
Dövrü hesablamalar – hansısa əməli (ümumi baxımda riyazi hesablamanı) bir neçə dəfə yerinə yetirilməsi lazım olduqda istifadə edilir. Ümumi yaxınlaşmada bu növ hesablamalarda iki hal ola bilər: dövrü hesablamaların sayı əvvəlcədən məlum olduqda və dövrü hesablamaların (dövrələrin) sayı əvvəlcədən məlum olmadıqda.

3.8.1 DO...LOOP dövrü hesablamalar operatorları strukturundan istifadə etmə qaydası

Sintaksisi:

```
Do [{While | Until} şərt]  
[təlimatlar]  
[Exit Do]  
[təlimatlar]
```

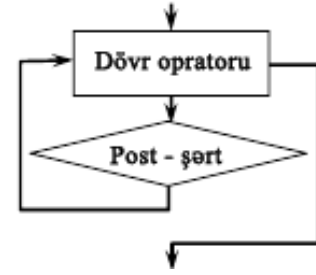
Loop



Başqa sintaksis VBA-da bu cürdür:

```
Do  
[təlimatlar]  
[Exit Do]  
[təlimatlar]
```

```
Loop [{While | Until} şərt]
```



İndi isə dövrü hesablamaların sayının əvvəlcədən müəyyən edilməyən hallara baxaq. Bu cür hesablamalarda VBA-da **Do While ... Loop** və **Do Until ... Loop** dövrü hesablamalar operatorlar strukturu tətbiq edilir.

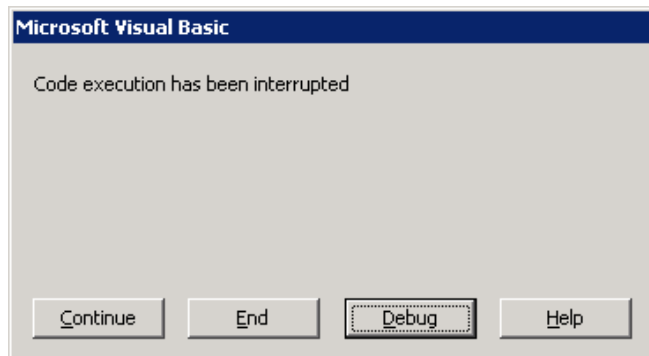
Tərtib edilən VBA prosedurasında **Do...Loop** idarəetmə komandasından istənilən sayda prosedura blokunda dövrü hesablamaların aparılması mümkündür. Komanda təyin edilən şərtə görə həmin şərtin **True** (doğru) qiyməti alana qədər yerinə yetirəcək.

Do While operatorlar strukturu hansısa əməlin (ümumi anlayışda riyazi hesablamaların) hansısa şərtin *doğru* olduğuna qədər yerinə yetirilməsi tələb olunduqda lazım olur, məsələn:

```
Do While MyVar < 10  
MyVar=MyVar+1  
MsgBox "MyVar=" & MyVar  
Loop
```

Praktiki hallarda tətbiqi – çox genişdir: bütün yazıları, tam qurtarana qədər, bir-bir seçmək və ayırmaq, istifadəçi verilənləri daxil etdikdə hansısa verilənin uyğun olan qiymətini daxil edənə qədər v.s.

Diqqət! Əgər, təsadüfən istifadəçi proqramında sonsuz dövrü işə salıbsa, onda hesablamaları dərhal dayandırmaq üçün klaviatürada **<Ctrl>+<Break>** klaviatura düymələri kombinasiyası basılmalıdır. Şək. 3.4-də göstərilən pəncərə açılacaq və oradan istifadəçi dərhal hesablamaları dayandırmaq seçimini edə bilər.



Səkil 3.4 **<Ctrl>+<Break>** klaviatura düymələri kombinasiyası ilə əmələ gələn makrosun işləməsini dayandıran dialog pəncərəsi.

Şərtin gerçək (True) olmasına qədər, Until komanda sözündən istifadə edərək, DO...LOOP dövrü operatorlar strukturundan istifadə etmə qaydaları

Birinci metodda `Until` (qədər) komanda sözündən istifadə edərək, hələ dövrü hesablamağa başlamadan öncə şərti yoxlamaq mümkündür və şərt ödənildikdə dərhal dövrü hesablama prosedurası yerinə yetirilməyə başlanır (bax aşağıdakı `ChkFirstUntil` adlı VBA proqram koduna):

```
Sub ChkFirstUntil()  
    counter=0  
    myNum=20  
    Do Until myNum=10  
        myNum=myNum-1  
        counter=counter+1  
    Loop  
    MsgBox "The loop made" & counter & "repetitions."  
End Sub
```

Yaxud (məsələnin qoyuluşuna görə) dövrü hesablama şərtin yoxlamasından öncə heç olmasa bir dəfə yerinə yetirilməlidirsə, onda `Until` (qədər) komanda sözündən aşağıdakı `ChkLastUntil` adlı VBA proqramı kodundakı metoddan istifadə etmək lazımdır. Həmin proqramda dövrü hesablamlar şərtəki yalan (False) qiymət ödənilməsinə qədər hesablamlar davam edəcək (oxucu bu proqramı diqqətlə analiz etsə çox faydalı nəticəyə nail olar).

```
Sub ChkLastUntil()  
    counter=0  
    myNum=1  
    Do  
        myNum=myNum+1  
        counter=counter+1  
    Loop Until myNum=10  
    MsgBox "The loop made" & counter & "repetitions."  
End Sub
```

`DO Until` strukturunun `Do While` tipli strukturundan fərqi ondan ibarətdir ki, bu halda hansısa əməlin (ümumi anlayışda riyazi hesablamların) hansısa şərtin *yalan* olduğuna qədər yerinə yetirilməsi təmin olur, məsələn:

```
Do Until MyVar>=10  
    MyVar=MyVar+1  
    MsgBox "MyVar = " & MyVar  
Loop
```

Həmin kodu bu cür dəyişərək də yazmaq olar:

```
Do  
    MyVar = MyVar + 1  
    WScript.Echo "MyVar = " & MyVar  
Loop While MyVar < 10
```

Bu halda hasablama dövrü heç olmasa bir dəfə yerinə yrtiriləcək.

Bü növ dövrü hesablama operatorlar strukturundan **Exit Do** açar sözləri əlavə edildikdə dərhal dövrədən çıxmaq mümkün olur.

Şərtin hələ ki (yaxud nə vaxt ki, ...) gerçək (True) olması while əmr sözündən istifadə edərək, DO...LOOP dövrü hesablamalar yerinə yetirən operatorlar strukturundan istifadə etmə qaydaları

Əgər tərtib edilən şərtin yoxlanması dövrü hesablamalardan öncə başlanmalıdırsa, onda **while** (hələ ki, nə vaxt ki) komanda sözü aşağıdakı `ChkFirstWhile` adlı VBA proqram misalındakı metoddan istifadə edilməlidir (nəzərə alaq ki, bu proqramda `myNum` identifikatorunun qiyməti 20 əvəzinə 9 olsa, onda **DO...Loop** dövrü hesablamalar yerinə yetirən operatorlar strukturu ilə dövrü hesablamalar başlamayacaqdır).

```
Sub ChkFirstWhile()  
    counter=0  
    myNum=20  
    Do While myNum > 10  
        myNum=myNum-1  
        counter=counter+1  
    Loop  
    MsgBox "The loop made " & counter & " repetitions."  
End Sub
```

Digər hal üçün (məsələnin qoyuluşuna görə) dövrü hesablama şərtin yoxlamasından öncə heç olmasa bir dəfə hesablama yerinə yetirilməlidir, onda **while** (hələ ki, nə vaxt ki) komanda sözündən aşağıdakı `ChkLastWhile` adlı VBA proqram misalında göstərilən metoddan istifadə etmək lazımdır (bu proqramda dövrü hesablamalar şərtin **False** (yalan) qiymətini ödəyəənə qədər yerinə yetiriləcəkdir):

```
Sub ChkLastWhile()  
    counter=0  
    myNum=9  
    Do  
        myNum=myNum-1  
        counter=counter+1  
    Loop While myNum > 10  
    MsgBox "The loop made " & counter & " repetitions."  
End Sub
```

DO...LOOP dövrü hesablamalar yerinə yetirən operatorlar strukturu əsasında işləyən dövrü hesablama prosedurasının içərisindən çıxmaq lazım olduqda çıxma metodu

Bu cür hallarda **DO...LOOP** komandası əsasında işləyən dövrü hesablama prosedurasının içərisindən çıxmaq üçün **Exit Do** komanda sözündən istifadə etmək mümkündür, və burada iki yol var:

1. sonsuz dövrü hesablamalardan çıxmaq üçün **Exit Do** komanda sözü **If...Then...Else** (yaxud da **Select Case**) şərti idarəetmə yoxlama blokunda doğru (**True**) olan qədər istifadə etmək olar;

2. sonsuz dövrü hesablamalardan çıxmaq üçün **Exit Do** komanda sözü və **If...Then...Else** (yaxud da **Select Case**) şərti idarəetmə yoxlama blokundan istifadə etmək olar.

Aşağıdakı `Exit_by_IF_Then_Else` adlı VBA proqram misalında 1-ci metoddan necə istifadə edilməsi qaydası göstərilib:

```
Sub Exit_by_IF_Then_Else ()
    counter=0
    myNum=9
    Do Until myNum=10
        myNum=myNum-1
        counter=counter+1
        If myNum < 10 Then Exit Do
    Loop
    MsgBox "The loop made" & counter & "repetitions."
End Sub
```

Diqqət! Sonsuz dövrü hesablamasının dayandırması klaviaturadakı **ESC** yaxud **<CTRL+BREAK>** düymələrinin basılması ilə mümkündür.

While ... Wend VBA-da başqa bir dövrü hesablama operatorlar strukturudur. Bu strukturda da əvvəlcədən dövrün təkrarlanması məlum olmur. Bu strukturun funksional imkanları **Do...While** strukturundan fərqlənir:

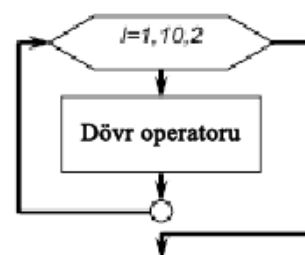
```
While MyVar < 10
    MyVar=MyVar+1
    WScript.Echo "MyVar = " & MyVar
Wend
```

3.8.2 For...Next dövrü hesablama operatorlar strukturundan istifadə etmə qaydası

Tərtib edilən VBA prosedurasında **For...Next** dövrü hesablamalar operatorlar strukturundan təyin edilmiş sayda prosedura blokunun idarə edilməsi mümkündür. Bu operatorlar strukturunda dövrü hesablamasının say indeksini istifadəçi özü təyin etməlidir. İndeksin qiyməti hər dövrdə arta yaxud azala bilər.

Sintaksis:

```
For say = start To son [Step artım]
[təlimatlar]
[Exit For]
[təlimatlar]
Next [say]
```



Aşağıdakı nümunə həmin sintaksisi əyani izah edir (VBA prosedurası işlədikdə 50 dəfə compyuter “beep” səsini icra edir):

```
Sub Beeps ()
  For x=1 To 50
    Beep
  Next x
End Sub
```

For ... Next dövrü hesablama operatorlar strukturundan istifadə etmə qaydası, Next əmr sözündən istifadə etmə qaydası.

For ... Next dövrü hesablama komandasında **Step** əmr sözündən istifadə edərək dövrü hesablamanın say indeksinin artmasına yaxud azalmasına nail olmaq olar. Məsələn, aşağıdakı VBA proqram misalında say indeksi hər dövrdə 2 qiyməti ilə artır (2, 4, 6, 8 və 10) və hesablama dövrü bitdikdə indeksin qiymətlərinin ümumi qiyməti 30-a bərabər olacaq..

```
Sub TwosTotal ()
  For j=2 To 10 Step 2
    total=total+j
  Next j
  MsgBox "The total is" & total
End Sub
```

Digər misalda isə **Step** komanda sözündən istifadə edərək, qiyməti mənfi təyin edildikdə dövrün say indeksi hər hesablama dövründə azaltmaq olur (16, 14, 12, 10, 8, 6, 4, və 2) və indeksin ümumi cəm qiyməti bərabərdir $total=76$.

```
Sub NewTotal ()
  For myNum=16 To 2 Step -2
    total=total+myNum
  Next myNum
  MsgBox "The total is" & total
End Sub
```

Diqqət! **Next** əmr sözündən sonra dövrü hesablama indeksini əlavə etməmək mümkündür.

VBA-da **For...Next** dövrü hesablama operatorlar strukturuna aid başqa bir sadə misala baxaq:

```
For iCounter=1 To 10
  MsgBox "Dövrün Sayı:" & iCounter
Next
```

Həmin dövrü hesablama operatorlar strukturuna **Step** operatoru da əlavə edilsə, onda yuxarıdakı proqramda dövrün sayının hər addımda nə qədər artdığını (dövrün artımını) müəyyən etmək olur, məsələn:

```
For iCounter = 1 To 10 Step 2
  MsgBox "Dövrün Sayı:" & iCounter
Next
```

Dövrün artımını azaltmaq üçün onun ədədi qiyməti mənfi götürülməlidir, məsələn:

```
For iCounter = 10 To 1 Step -2
  MsgBox "Dövrün Sayı:" & iCounter
Next
```

For...Next dövrü hesablama operatorlar strukturundan şərtsiz çıxmaq lazım olduqda həmin struktura **Exit For** operatorları əlavə edilir, məsələn:

```
VStop=InputBox("Dayanma qiymətini daxil edin")
VInput=CInt(VStop)
For iCounter=1 To 10
  MsgBox "Sayqac:" & iCounter
  If iCounter=VInput Then Exit For
Next
```

For ... Next dövrü hesablama operatorlar strukturunun içərisindən çıxmaq lazım olduqda mövcud olan qaydalar

Bu cür hallarda **For ... Next** komandası əsasında işləyən dövrü hesablama prosedurasının içərisindən çıxmaq üçün **Exit For** komanda sözündən istifadə etmək mümkündür, və burada iki yol var:

1. dövrü hesablamalardan çıxmaq üçün **Exit For** komanda sözü **If...Then...Else** yaxud (**Select Case**) şərti idarəetmə yoxlama blokunun doğru (**True**) olan hissəsində istifadə edilir;
2. dövrü hesablamalardan çıxmaq üçün **Exit For** komanda sözü **If...Then...Else** yaxud (**Select Case**) şərti idarəetmə yoxlama blokunda yalan qiymət (**False**) olduqda istifadə edilir;

For Each...Next hesablama operatorlar strukturundan istifadə etmə qaydası

Sintaksis:

```
For Each element In grupda
  [təlimatlar]
  [Exit For]
  [təlimatlar]
```

```
Next [element]
```

For Each...Next komandası prosedurada ki hesablama blokunda hər bir obyektə və ya matrisin hər bir elementi üçün dövrü hesabları müəyyən edilmiş qədər təkrar edir. Visual Basic hər hesablama dövründə dəyişənə növbəti qiymət verir. Məsələn, aşağıdakı VBA prosedurasında yalnız proseduranın öz formasından başqa bütün başqa formaları bağlayır:

```
Sub CloseForms ()
```

```

For Each frm In Application.Forms
    If frm.Caption <> Screen.ActiveForm.Caption Then frm.Close
Next
End Sub

```

Bir çox hallarda VBA-da hansısa kolleksiyanın bütün elementləri ilə müəyyən qaydada əməllər yerinə yetirilməlidir – bütün açıq olan sənədləri, bütün Excel səhifələrini və ya hansısa diapazonda bütün hücrələri (cells) bir-bir seçib ayırmaq v.s. Belə hallar üçün VBA-da çox əlverişli üsul var: **Each** sözü həmin dövrü operatorlar strukturuna əlavə edilir - **For Each ... Next** (burada İngiliscə **Each** açar sözünün mənası Azərbaycanca *hər bir* mənasına gəlir), məsələn :

```

For Each oWbk In Workbooks
    MsgBox oWbk.Name
Next

```

Bu üsuldən istifadə etdikdə istənilən obyektə istinadı asanlıqla təyin edib almaq mümkündür:

```

For Each oWbk In Workbooks
    If oWbk.Name="Informasiya.xls" Then
        Set oMyWorkBook=oWbk
Exit For
    End If
Next

```

Yuxarıdakı son nümunədə dövrlərlə `Workbooks` kolleksiyasının bütün elementləri bir-bir seçilir (Excel-in açıq olan kitabları), hər bir kitabın adı yoxlanılır, və əgər `Informasiya.xls` adlı kitab tapılırsa, onda ona olan istinad təyin edilir və proqramdan çıxma mümkün olur. İş kitablarının kolleksiyası – xüsusi olan kolleksiyadır: özü özündə elementlərin adına görə axtarış apara bilir. Buna görə aşağıdakı bir sətir VBA kodu eyni əməli yerinə yetirə bilər:

```

Set oMyWorkBook=Workbooks("Informasiya.xls")

```

Başqa hallarda **Each** açar sözü olmadan həmin axtarış yerinə yetirilə bilməz.

Digər aşağıdakı proqramda matrisin hər elementi seçilərək onu `I` adlı indeks dəyişəninə qiymətinə bərabər edir.

```

Dim TestArray(10) As Integer, I As Variant
For Each I In TestArray
    TestArray(I)=I
Next I

```

`For Each ... Next` dövrü hesablama operatorlar strukturundan lazım olduqda çıxma qaydaları

`For ...Next` operatorlar strukturuna analogi olaraq, `For Each...Next` dövrü hesablama operatorlar strukturu blokundan aşağıdakı qaydalar əsasında çıxmaq mümkündür:

1. dövrü hesablamalardan çıxmaq üçün **Exit For** komanda sözü **If...Then...Else** yaxud (**Select Case**) şərti idarəetmə yoxlama blokunda doğru (**True**) olan hissəsində istifadə edilir;
2. dövrü hesablamalardan çıxmaq üçün **Exit For** komanda sözü **If...Then...Else** yaxud (**Select Case**) şərti idarəetmə yoxlama blokunda yalan qiymət (**False**) olduqda istifadə edilir;

Aşağıdakı VBA proqram misalında birinci hücrədən başlayaraq, A1:B5 diapazonunun bütün hücrələri ədədsiz şərtə görə yoxlanılır. Əgər belə hücrə (cell) tapılırsa dərhal bu haqda xəbər verilir və dövrü hesablamadan çıxma mümkün olur.

```
Sub TestForNumbers()
  For Each myObject In MyCollection
    If IsNumeric(myObject.Value)=False Then
      MsgBox "Object contains a non-numeric value."
      Exit For
    End If
  Next myObject
End Sub
```

3.8.3 VBA-da şərti idarəetmə operatorlar strukturları. **If...Then...Else** şərti idarəetmə operatorlar strukturundan istifadə etmə qaydası

Bütün alqoritmik dillərin tərkibində (həmçinin VBA-da da) proqramlaşdırmaq üçün tez-tez istifadə edilən ən vacib element - *şərti idarəetmə operatorlar strukturlarıdır*. Onların ümumi işləmə prinsipi olduqca sadədir: kəmiyyətin hansısa şərtlərə uyğun qəlib-gəlməməyi yoxlanılır (hansısa ifadələrin doğru yaxud yalan olması) və bu yoxlamanın nəticəsindən asılı olaraq, proqramdakı hesablamaların yerinə yetirilməsi bu və ya digər budaqlanan istiqamətə yönəlir. VBA-da iki şərti idarəetmə operatorlar strukturu vardır: **If... Then... Else** və **Select Case**.

Sintaksis:

```
If şərt Then [təlimatlar] [Else else_təlimatları]
```

Yaxud istifadəçi başqa sintaksisdən istifadə edə bilər:

```
If şərt Then
[təlimatlar]
[ElseIf şərt_n Then
[Else_təlimatları] ...
[Else
[else_təlimatları]]
End If
```



Proqramçılar öz işlərində daha çox **If... Then... Else** şərti idarəetmə operatorlar strukturundan istifadə edirlər. Onun tam sintaksisini belə ifadə etmək olar:

```

If Şert Then
    Əmr_1
  [ElseIf Şərtlər_N Then
    Əmrlər_N]
  [Else
    Əmr_2]
End If

```

Burada:

- Şert - doğruluğu yoxlanılan ifadədir. Əgər ifadə doğrudursa, onda Əmr_1 yerinə yetirilir, yalan olduqda isə - Əmr_2 yerinə yetirilir;
- Şərtlər_N — yoxlanıla bilən əlavə şərtlərdir. Onların yoxlamada doğru olduğu halda Əmrlər_N əmrləri yerinə yetirilir;

Şerti idarəetmə operatorlar strukturu **If...Then... Else** aşağıdakı hallarda tətbiq edilir:

- Hansısa şertə uyğun olmanın yoxlamasında hansısa əməli yerinə yetirmək lazım olduqda, məsələn:

```

If nTemperature < 10 Then
  MsgBox "Gödəkçəni geyinmək"
End If

```

- əgər əvvəlki nümunədəki əməl yerinə yetirilməlidirsə və uyğunluq olmadığı halda (yoxlamanın nəticəsi yalan olduqda), onda başqa əməlin yerinə yetirilməyi lazım olur, məsələn:

```

If nTemperature < 10 Then
  MsgBox "İsti gödəkçəni geyinmək"
Else
  MsgBox "Nazik gödəkçəni geyinmək"
End If

```

- əgər bir neçə şertə uyğun olmanın yoxlanması lazımdırsa (aşağıdakı nümunədə məntiqi operatorların istifadəsinə diqqət verin):

```

If (nTemperature < 10) And (bYaghmur=True) Then
  MsgBox "Gödəkçəni geyinmək və çətiri götürmək"
End If

```

- əgər birinci yoxlamadan **False** (Yalan) nəticəsi alınmışdır və əlavə şərtlərin yoxlanması tələb olunur, (bu halda **ElseIf** (digər halda) operatoru həmin şərti idarəetmə operatorlar strukturunda istifadə edilməlidir)), məsələn:

```

If (bIGoInCar=True) Then
  MsgBox "Avtomobil üçün geyinmək lazımdır"
ElseIf nTemperature < 10 Then
  MsgBox "Gödəkçəni geyinmək"
Else
  MsgBox "Bir köynəkdə də getmək olar"
End If

```

Son nümunədə `bIGoInCar` - **Boolean** tipli dəyişəndir və o, iki qiyməti ala bilər: **True** və ya **False**. Buna görə həmin VBA cod sətiri bu cür də yazıla bilər:

```
If bIGoInCar Then ...
```

If...Then...Else şərti idarəetmə operatorları strukturuna aid bəzi vacib olan qeydlər:

- **Then** (onda) açar sözü həmişə **If** (əgər) açar sözü və verilən şərtlə bir sətirdə olmalıdır (digər halda VBA kompilyatoru səhv haqqında xəbər verəcək);
- əgər şərtin yoxlanmasından doğru nəticə olduqda yerinə yetiriləsi əmr **If** və **Then** operatorları ilə bir sətirdə yazılıbsa, onda **End If** yoxlamayı tamamlayan operatorlar yazılmaya da bilər, məsələn:

```
If nTemperature<10 Then MsgBox "Gödəkçəni geyinmək"
```

- əgər istifadəçi **Else/ElseIf** operatorlarını yaxud bir neçə əmri yazıbsa, onda **End If** yoxlamayı tamamlayan operator sonda yazılmalıdır, digər halda VBA kompilyatoru səhv haqqında xəbər verəcək;
- **If...Then** yoxlama strukturu tərtib edildikdə, əmrlər blokunu nəzərə gətirmək üçün, bu operatorlar bir neçə boş yerdən sonra yazılmalıdır. Digər halda həmin yoxlama strukturunun sonra həmin istifadəçi yaxud digər istifadəçilərin oxunması üçün çətinlik yarana bilər;
- **If...Then** operatorları bir birinin içərisində praktiki olaraq çox sayda (limitsiz dəfə) təkrarlana bilərlər, məsələn:

```
If MyVar=5 Then  
MsgBox "MyVar=5"  
  If MyVar=10 Then  
    MsgBox "MyVar=10"  
  End If  
End If
```

- istifadəçi proqramındakı riyazi şərtlərin tələblərinə görə **If ... Then ... Else (Əgər ... Sonra ... Digər halda)** VBA məntiqi şərti idarəetmə operatorları strukturundan istifadə edərək, istənilən mürəkkəblikdə alqoritm tərtib edə bilər. Bu komandanı istənilən sayda (təkrarlama) prosedura blokunda vermək olar. Proqramın aydın oxunulması şərtinə görə daha yaxşı olar ki, çox sayda **If...Then...Else** blokunun istifadə edilməsindənə **Select Case (Hadisələrin Seçimi)** komandası tətbiq edilsin.

Aşağıda **If...Then...Else** blokunun istifadə edilməsi ilə bağlı sadə VBA proqramlaşdırma kodları və uyğun olan izahatların oxucu tərəfindən diqqətlə öyrənilməsinə vacib sayırıq.

IF...THEN...ELSE şərti idarəetmə operatorlar strukturundan şərtin doğru (True) olduğu halda istifadə etmə qaydası.

Əgər şərt doğrudursa (True) onda If...Then...Else idarəetmə komandasının If...Then strukturu tətbiq edilə bilər. Aşağıdakı VBA proqram misalında həmin idarəetmə komandasının bir sətirlik strukturu göstərilib:

```
Sub FixDate()  
    myDate = #2/13/95#  
    If myDate < Now Then myDate=Now  
End Sub
```

Əgər proqramın strukturuna görə bir sətir kifayət etmirsə (prosedura bloku həmin idarəetmə komandasının tərkibinə əlavə edildikdə), onda çox saylı sətir sintaksisi tətbiq edilir və bu sintaksisdə həmin idarəetmə blokunu End If idarəetmə sözü tamamlayır (bax aşağıdakı AlertUser1 adlı VBA proqramına):

```
Sub AlertUser1(value as Long)  
    If value = 0 Then  
        AlertLabel.ForeColor = "Red"  
        AlertLabel.Font.Bold = True  
        AlertLabel.Font.Italic = True  
    End If  
End Sub
```

If ... Then ... Else şərti idarəetmə operatorlar strukturunun şərtin doğru (True) və digər hissəsinin yalan (False) olduğu halda istifadə etmə qaydası.

Belə hallarda If ... Then ... Else idarəetmə komandasını iki bloka ayırmaq lazımdır: birinci blokda (Then idarəetmə sözündən sonra) doğru (True) olan şərtə aid blok hesablanır, ikinci blokda isə (Then idarəetmə sözündən sonra) yalan olan şərtə uyğun blok hesablanır və idarəetmə komandası blokunun sonunda End If idarəetmə sözü yerləşdirilir (bax aşağıdakı AlertUser2 adlı VBA proqramına):

```
Sub AlertUser2(value as Long)  
    If value=0 Then  
        AlertLabel.ForeColor=vbRed  
        AlertLabel.Font.Bold=True  
        AlertLabel.Font.Italic=True  
    Else  
        AlertLabel.Forecolor=vbBlack  
        AlertLabel.Font.Bold=False  
        AlertLabel.Font.Italic=False  
    End If  
End Sub
```

Daha mürəkkəb halda, əgər **IF ... THEN ... ELSE** şərti idarəetmə operatorlar strukturundan birinci şərt bloku yalan (**False**) olduqda ikinci şərt blokunda **ELSEIF** əmr sözünün tərtib etmə qaydası.

ElseIf idarəetmə komandasının **If...Then...Else** idarəetmə operatorlar strukturunun ikinci şərt blokuna əlavə etmək olar və yalnız bir halda: birinci şərt blok doğru deyil (**False**). Aşağıdakı VBA **Bonus** adlı funksiya prosedura proqramı həmin idarəetmə strukturunun necə həyata keçirilməsini göstərir:

```
Function Bonus(performance, salary)
  If performance=1 Then
    Bonus=salary*0.1
  ElseIf performance=2 Then
    Bonus=salary*0.09
  ElseIf performance=3 Then
    Bonus=salary*0.07
  Else
    Bonus=0
  End If
End Function
```

3.8.4 **SELECT CASE** şərti idarəetmə operatorlar strukturundan istifadə etmə qaydası

Select Case VBA şərti idarəetmə operatorlar strukturu ilə, istifadəçinin təlimatına uyğun, bir neçə qrup hesablamaları (ifadənin qiymətindən asılı olaraq) yerinə yetirmək olar.

Sintaksis:

```
Select Case yoxlama_ifadəsi
[Case ifadələr_sayahisi_n
[təlimatlar_n]] ...
[Case Else[digər_halda_təlimatlar]]
End Select
```

Cədvəl 3.6 **Select Case** şərti idarəetmə operatorlar strukturunun sintaksisinin hissələri

Hissə	İzahı
yoxlama_ifadəsi	Tələb olunur. İstənilən ədədi yaxud ədədi ifadə, yaxud simvol ifadəsi.
ifadələr_sayahisi_n	Case komandası əmələ gəldikdə tələb edilir.
təlimatlar_n	Seçimdən aslıdır. Əgər yoxlama_ifadəsi hissəsi bir neçə dəfə ifadələr_sayahisi_n uyğun gəlibse.
digər_halda_təlimatlar	Seçimdən aslıdır. Bir neçə təlimat icra edilmişdir, lakin yoxlama_ifadəsi hər hansı Case icrasına uyğun gəlməmişdir.

Lazım olduqda **Case** komandasında çoxsaylı ifadələr istifadə edilə bilər, məsələn, aşağıdakı misaldakı kimi:

Case 1 To 4, 7 To 9, 11, 13, Is > MaxNumber

Yaxud digər VBA misalındakı kimi (**Case** komandasında simvollar ardıcılığı verilib):

Case "everything", "nuts" **To** "soup", TestItem

Select Case şərti idarəetmə operatorlar strukturunun istifadəsi aşağıdakı VBA proqram misalında daha aydın göstərilib:

```
Dim Number
Number=8 'Dəyişənin nömrələnməsi.
Select Case Number 'Number kəmiyyətinin yoxlanması.
Case 1 To 5 'Number 1 və 5 arasındadır.
    Debug.Print "1 və 5 intervalında"
'Aşağıdakılar yalnız True doğru qiyməti yoxlayan tamamlayıcı
'Case şərti idarəetmə operatorlar strukturudur.
Case 6, 7, 8 'Number kəmiyyəti 6 and 8 intervalındadır.
    Debug.Print " 6 və 8 intervalında"
Case 9 To 10 'Number 9-dur və ya 10-dur.
Debug.Print "8-dən çoxdur"
Case Else 'Başqa qiymət.
    Debug.Print "1 və 10 intervalında deyil"
End Select
```

Peşəkar VBA proqramçıları **Select Case** şərti idarəetmə operatorlar strukturunu eyni qiymətin bir neçə dəfə (müəyyən olmuş sayda) müxtəlif ifadələrlə müqayisə etmək lazım olduqda ən ideal alət kimi istifadə edirlər. Onu strukturunu daha aydın anlamaq üçün aşağıdakı VBA kod nümunəsinə nəzər salaq:

```
Select Case sDayOfWeek
Case "I gün"
    MsgBox "Riyaziyyat mühazirəsi"
Case "II gün"
    MsgBox "Kompyuter elmləri mühazirəsi"
...
Case Else
    MsgBox "Bu gün heç bir mühazirə"
End Select
```

Select Case şərti idarəetmə operatorlar strukturuna aid bəzi vacib qeydlər:

- yuxarıdakı nümunədə **Case** "I gün" VBA cod sətiri əslində

Case sDayOfWeek="I gün" mənasına gəlir. Başqa operator və ya operatorlar sırası da istifadə edilə bilər, məsələn,

```
Case 0 To 5, 15, Is > 55
    MsgBox "Bayramı yada sal"
```

Yuxarıdakı nümunənin bəzi vacib hissələrini xüsusi izah etməyimiz oxucu üçün çox faydalı ola bilər:

- Nəzərə alınmalıdır ki, **Is** açar sözü əslində yazılmaya də bilər – VBA kompilyatoru bunu istifadəçinin əvəzinə avtomatik edəcək.
- Bir neçə **Case** yoxlamaları **OR** operatoru kimi birləşə bilər – yeni yoxlama həmin budaq istiqamətində yalnız bir şərtə görə gedə bilər: *əgər yoxlanan qiymət heç olmasa hansısa bir dəyəri ödəyirsə*. Həmin yoxlama dəyərləri bir birilə vergüllə aralanır; (0 **To** 5) diapazonu istifadə edildikdə həmin diapazonun sərhədləri də daxil edilir (baxılan nümunədə 0 və 5).

3.8.5 VBA-da **GoTo** idarəetmə operatoru, onun tətbiq etmə halları

GoTo idarəetmə operatoru VBA-da şərtsiz keçid operatorudur. Proqramdakı hesablamada ardıcılıq heç bir şərtə görə yoxlamadan başqa və əvvəlcədən təyin edilmiş kod sətirindəki (proqramın işləməsindən əvvəl təyin edilmiş yerə) əmrlərin hesablanması lazım olduqda bu operator tətbiq edilir.

Sintaksisi: **GoTo** Label_1

Burada Label_1 işarə edilən VBA kod sətirinin işarələnməsini göstərir (bu işarədən sonra (:)) işarəsi qoyulduqda VBA kompilyatoru sonra gələn kod işarələri keçid üçün nəzərdə tutulan sətir kimi anlayır.

Aşağıda **GoTo** idarəetmə operatoru ilə bağlı sadə VBA kod nümunəsi gətirilib:

```
GoTo EngineNotStarted
...
EngineNotStarted:
MsgBox "avtobusla gedirik"
...
```

Bu nümunədə EngineNotStarted: - sətir işarəsidir (Label), onun adını istifadəçi öz istəyi ilə seçə bilər. İşarənin adı yazıldıqdan sonra isə (:) işarəsi hökmən qoyulmalıdır.

Bəzən həqiqətən də **GoTo** çox faydalı ola bilər – məsələn, istifadəçinin məlum olmayan sayda verilənləri düzgün olaraq daxil edilməsi tələb olanda istifadə edilir.

Aşağıda VBA sorğu materiallarında verilən bir az çətin görükən nümunənin burada oxucunun sərbəst işləməsi üçün nümayiş edilməsini vacib sayırıq.

```
Sub GotoStatementDemo()
Dim Number, MyString
    Number=1      ' Dəyişən nömrələnir.
    ' Number qiyməti qiymətləndirilərək nişanlara göndərilir.
    If Number = 1 Then GoTo Line1 Else GoTo Line2

Line1:
    MyString="Number bərabərdir 1"
    GoTo LastLine      ' LastLine nişanına gedmək.
Line2:
    'Aşağıdakı təlimat heç vaxt icra edilməyəcək.
```



```

MyString="Number bərabərdir 2"
LastLine:
  Debug.Print MyString 'Immediate Window Print pəncərəsində
  "Number bərabərdir 1" sözünün çap edilməsi.
End Sub

```

Diqqət! Bütün ən müasir proqramlaşdırma dillərində **GoTo** idarəetmə operatoru çoxdan ləğv edilib və olduqca ziyanlı operator sayılır. Məsələn, Python dilində bu operator ümumiyyətlə, yoxdur, çünki hesab edilir ki, **GoTo** operatoru struktur proqramlaşdırma prinsiplərinə zidd üsuldur. Görünür ki, Microsoft kompaniyasının **GoTo** operatorunu saxlaması köhnə (hətta ziyanlı olsa da) ən-ənəyə sadıqlıq nümunəsi kimi izah edilə bilər. Çünki VBA-da olan struktur idarəetmə operatorları imkan verir ki, tamam ilə bu operatorsuz effektiv proqramlar istifadəçilər tərəfindən tərtib edilsin. Bu operator haqqında ətraflı məlumat verməyimizə baxmayaraq, biz də, öz tərəfimizdən, VBA istifadəçilərini həmin operatorlardan mümkün qədər istifadə etməməyə məsləhət görürük:

- proqramlaşdırma kodu çox çətin oxunan hala gəlir;
- **GoTo** operatoru ən ağır hallarda sonsuz təkrarlanan hesablamalara çıxara bilər (VBA kompilyatoru belə səhvi tapmaya da bilər).

Qeyd: **GoTo** operatorunu çox böyük effektlə və müvəffəqiyyətlə VBA-da yaxşı səviyyədə təmsil olan strukturlu idarəetmə operatorları ilə əvəz etmək olur (**Do...Loop**, **For...Next**, **If...Then...Else**, **Select Case**).

3.9 VBA-da PROSEDURALAR VƏ FUNKSIYALAR

Proseduralar – VBA dilinin ən vacib funksional bloklarıdır. VBA-da istifadəçi yalnız proqram kodunu icra edə bilər o, kod isə hansısa proseduranın tərkibini təşkil edir (adətən elementinin hadisə ilə bağlı standart modulda yerləşən idarəetmə formasında v.s.). Bəzən təcrübəsiz istifadəçilər əmrilər və təlimatları birbaşa standart modulun elan etmə sahəsinə yazmağa təşəbbüs edirlər, və sonra proqramın işləməməsini anlaya bilmirlər (səhv haqqında heç bir məlumat verilmir – sadəcə həmin kod kompilyator üçün "görünməz" olur). Səbəb sadədir - modul üzrə elanlar bölməsində (siyahılarının üst hissəsində **General** və **Declarations** qiymətləri göstərilir yalnız dəyişənlər elan edilir və kompilyator üçün modul səviyyədə bəzi xüsusi təlimatlar göstərilə bilər. Proqram kodunun bütün qalan hissəsi isə proseduranın içərisində yerləşməlidir.

VBA-da aşağıdakı prosedura tipləri nəzərdə tutulub (sonrakı hissələrdə oxucu daha ətraflı məlumatlarla tanış olacaq):

- **Sub**;
- **Function**;

- **Property.**

Sub tipli prosedura (alt-proqram) – hansısa səviyyədə əməllərin icrası üçün ən universal prosedura növüdür, məsələn, aşağıdakı VBA-da tərtib edilmiş sadə **Sub** prosedurasına bax:

```
Sub Farewell()  
  MsgBox "Sağ ol!"  
End Sub
```

VBA-da makros - əslində parametri olmayan **sub** tipli proseduradır. VBA redaktorundan yalnız makrosların adları ilə yaxud MS Office proqramından onları çağırmaq mümkündür. Başqa proseduraları yalnız digər proseduralardan və ya xüsusi üsullar ilə çağırmaq mümkündür - onlar haqqında növbəti hissələrdə danışılacaq.

Function tipli prosedura (funksiya) – icra ediləsi VBA əmrləri (təlimatları) toplusudur. **sub** tipli proseduralardan **Function** tipli proseduranın prinsipial fərqi ondan ibarətdir ki, funksiyanı çağıran proqrama o, sonra proqramda istifadə ediləsi, hansısa hesablanmış qiyməti qaytarır. Misal üçün aşağıdakı **Function** tipli prosedura nümunəsinə baxaq:

```
Function Tomorrow()  
  Tomorrow=DateAdd("d", 1, Date())  
End Function
```

və onun, mümkün olan, çağırılma nümunəsi (tutaq ki, **Test1()** adlı **sub** tipli proseduradan) belə ola bilər:

```
Private Sub Test1()  
  Dim dDate  
  dDate=Tomorrow  
  MsgBox dDate  
End Sub
```

VBA kodunun mətnində **Function** tipli prosedura hansısa qiymətin mənimsəməsini təmin edən operatoru təyin etmək lazımdır. Yuxarıdakı misalda bu cür operator kimi **Tomorrow=DateAdd(" d", 1, Date())** kod sətirindəki **Tomorrow** dəyişənidir.

Prinsip etibarı ilə **sub** tipli prosedura da hansısa bir və ya bir neçə sayda hesablanmış qiymətləri qaytara bilər – dəyişənlərin köməyi ilə (bu haqda aşağıda daha ətraflı məlumat veriləcək). Bəs onda funksiyalar (**Function** tipli proseduralar) nə üçün lazımdır? Hər şey çox sadədir: funksiyanı proqram kodunun istənilən yerində, istənilən sayda təkrarlamaq lazım gəlir. Belə olduqda **Function** tipli prosedura **sub** tipli prosedurada müəyyən edilmiş qayda əsasında təkrarlandığında məhsuldarlıq xeyli artır. Məsələn, yuxarıdakı **Function** prosedurasının **sub** prosedurasında istifadə edilməsi aşağıdakı nümunədə göstərilib (**Function** tipli **Tomorrow()** prosedurası, **Test_1()** adlı **sub** prosedurasından icra edilməsi üçün çağırılır):

```
Private Sub Test_1()  
    MsgBox Tomorrow()  
End Sub
```

VBA-da yüzlerle daxili (qurulmuş) standart funksiyalar da var - daha çox sayda bu cür funksiyalar MS Office proqramlarının obyekt modellərində nəzərdə tutulub. Hətta yuxarıda nümunə kimi gətirilən sadə VBA nümunələrində də iki qurulmuş funksiya vardır: `Date()` – kompyuterin saatına uyğun olaraq zamanın cari qiymətini qaytarır və `DateAdd()` – cari tarixin qiymətinə müəyyən qiymətdə günlərin, ayların, illərin v.s. qiymətlərini əlavə edir. Qurulmuş standart funksiyalar haqqında daha ətraflı növbəti hissələrdə danışılacaq.

VBA-da hadisələrin emal edilməsini təmin edən proseduralar (event procedure) da vardır – müəyyən hadisələrin baş tutmasında, avtomatik olaraq, işə salınan xüsusi təyinatlı `Sub` tipli proseduralar var. Bu cür proseduralar haqqında daha ətraflı məlumatı oxucu kitabın VBA formaları və hadisələri haqqında olan hissələrində tapacaq.

VBA-da `Property` tipli (xassələr proseduraları) proseduralar da nəzərdə tutulub. Onlar istifadəçinin özünün yaratdığı siniflərin xassələrini təyin etmək üçün lazımdır. Bu mövzu isə kitabın ahəməsindən çıxır. Buna baxmayaraq, `Property` tipli proseduralar haqqında da aşağıda müəyyən qədər praktiki məlumat var.

3.9.1 PROSEDURALARIN NÖVÜ (`Sub`, `Function`, `Property` və standart riyazi funksiyalar)

VBA proseduraları (`Sub`) və funksiyalar (`Function`), prosedura və funksiyaların elan edilməsi, VBA proseduralarının xüsusi tipi

VBA-da proseduranın (PROCEDURE) tərfi: *verilmiş adı olan və vahid hissə kimi icra edilən hesablama təlimatları sırasına prosedura deyirlər. Məsələn, `Function`, `Property`, və `Sub` VBA-da proseduralarının tipidir. Proseduranın adı həmişə modul səviyyəsində təyin edilir. İstənilən icra ediləsi proqram kodu proseduranın içərisində saxlanılır. Proseduralar bir-birinin içərisinə yerləşə bilməz - bir prosedura başqasını içərisindən icra edilməsi üçün çağırma bilər.*

PROSEDURANIN TAPILMASI

Proqramlaşdırma pəncərəsində (**Code Window**) proseduranın tapılması:

- cari proseduranı görmək üçün `Object` dialoq zolağının `Code` pəncərəsində (`General`) və sonra `Procedure` dialoq zolağında `procedure` seçilməlidir.
- hadisə prosedurasını görmək üçün `Code` pəncərəsində `Object` dialoq zolağının `Procedure` dialoq zolağında uyğun olan obyekt seçilməlidir.

Qeyd: vizual olaraq, proseduraları `Code` pəncərəsində ayırmaq üçün `Options` dialoq zolağının (`Tools` menyusu) `Editor` sırasındakı `Procedure Separator` seçilməlidir. İstifadəçi

Procedure görüntüsündən **Full Module** görüntüsünə keçmək üçün gerek **Code** pəncərəsindəki aşağı-sol tərəfdə yerləşən düymələri basaraq seçim edilsin.

Proseduranı başqa modulda axtarib tapmaq üçün:

1. **View** menyusunda **Object Browser** seçilməli, yaxud klaviaturla **F2** düyməsi basılmalıdır.
2. **Project/Library** dialoq zolağında layihə (**Project**) seçilməlidir.
3. **Classes** siyahısında modul seçilməlidir.
4. **Members of** siyahısında proseduranın adı mausun ikiqat basılması ilə seçilməlidir.
5. **Code** pəncərəsində görsənən prosedura seçilməlidir.

İstifadəçi aşağıdakı klaviatura kombinasiyasını seçə bilər, cədv. 3.7:

Cədvəl 3.7 Proseduraların idarə edilməsində işlənən klaviatura kombinasiyaları

Basılan klavişlər	Baş verən hadisə
CTRL+DOWN ARROW	Növbəti proseduranı göstərir.
CTRL+UP ARROW	Əvvəlki proseduranı göstərir.
PAGE DOWN	İstifadəçi kodundakı proseduralarda Page down (bir səhifə aşağı) edir.
PAGE UP	İstifadəçi kodundakı proseduralarda Page up (bir səhifə yuxarı) edir.
F2	Object Browser (obyekt bələdçisi) dialoq pəncərəsini göstərir.

sub PROSEDURASI: sub prosedurasının adı, arqumentləri və proqramın bədənini formalaşdıran kodu tərtib edilir.

Sintaksisi:

```
[Private | Public | Friend] [Static] Sub adı [(argsiyahsı)]
[təlimatlar]
[Exit Sub]
[təlimatlar]
```

End Sub

sub prosedurasının sintaksisinin hissələri (Cədv 3.8):

Cədvəl 3.8 Proseduraların idarə edilməsində işlənən klaviatura kombinasiyaları

Hissə	Təsviri
Public	Seçimlidir. sub prosedurasının bütün modullardakı proseduraları üçün əlçatan olduğunu bildirir. Əgər modulda Option Private rejimində saxlanılırsa, layihədən kənar yerdən əlçatan olmur.
Private	Seçimlidir. sub prosedurasının yalnız modulda elan edildiyi yerdə proseduralar üçün əlçatan olduğunu bildirir.
Friend	Seçimlidir. Yalnız sinif modulda istifadə edilir. sub prosedurası layihədə

	görunür - obyekt nümunəsi kontrollerində isə görünür.
Static	Seçimlidir. Sub prosedurasının lokal dəyişənləri çağırıldığı zaman arasında yaddaşda saxlanılır. Static atributu dəyişənlərin Sub -dan kənarında elan edildiyindən heç bir hadisə baş vermir, hətta onlar prosedurada istifadə edilsə də.
name	Məcburidir. Function proseduranın adı; dəyişənlərin standart adlandırma qaydasına uyğundur.
arqsiyahısı	Seçimlidir. Sub prosedurası çağırıldıqda onun proseduraya ötürüldüyü arqumentlərini təmsil edən dəyişənlərdən ibarət siyahıdır. Dəyişənlər çox olduqda, bir biri ilə vergül ilə ayrılırlar.
statements	Seçimlidir. Sub prosedurası içərisində istənilən icra ediləsi təlimatlar qrupu.

arqsiyahısı arqumenti aşağıdakı sintaksisə tabedir və hissələrdən ibarətdir, cədv. 3.9:

[**Optional**] [**ByVal** | **ByRef**] [**ParamArray**] varadı [()] [**As tip**] [= standartqiymət]

arqsiyahısı arqumenti aşağıdakı hissələrdən ibarətdir, cədv. 3.7:

Cədvəl 3.9 Proseduraların idarə edilməsində işlənən klaviatura kombinasiyaları

Hissə	Təsviri
Optional	Seçimlidir. Arqumentin tələb olmadığını göstərən açar sözdür. Əgər ParamArray istifadə edilirsə, onda Optional heç bir arqumentdə istifadə edilə bilməz.
ByVal	Seçimlidir. Arqumentin qiymətə verilməsini göstərir.
ByRef	Seçimlidir. Arqumentin istinadla ötürülməsini göstərir. Visual Basic-də standart hissədir.
ParamArray	Seçimlidir. Arqumentlər siyahısının sonunda göstərilir və sonuncu arqumentin variant elementlərinin seçimli çoxluğuna aid olduğuna işarə edir. ParamArray açar sözü istənilən sayda arqumentin təmin olmasını əmin edir. ParamArray açar sözü ByVal , ByRef , yaxud Optional ilə birgə istifadə edilə bilməz.
varadı	Məcburidir. Arqumenti təmsil edən dəyişənin adıdır, standart adlandırma qaydalarına tabedir.
tip	Seçimlidir. Sub prosedurasının arqumentinin verilənlər tipini göstərir. Həmin verilənlər tipi bunlardan ola bilər: Byte , Boolean , Integer , Long , Currency , Single , Double , Decimal , Date , String , Object , Variant , yaxud istifadəçi tərəfindən təyin edilmiş digər başqa tiplər.
standartqiymət	Seçimlidir. İstənilən sabit və ya sabit olan ifadə ola bilər. Yalnız seçimli parametrlər üçün doğru sayıla bilər. Əgər tip obyekt tipinə aiddirsə, onda aşkar standart qiyməti Nothing (heç nə) olmalıdır.

Sub PROSEDURASINA AİD SADƏ NÜMUNƏLƏR (VBA sorğu materiallarından götürülüb):

NÜMUNƏ 1:

```
' Sub procedure definition.
' Sub procedure with two arguments.
Sub SubComputeArea(Length, TheWidth)
    Dim Area As Double ' Declare local variable.
    If Length=0 Or TheWidth=0 Then
```

```

' If either argument = 0.
  Exit Sub ' Exit Sub immediately.
End If
Area=Length*TheWidth 'Calculate area of rectangle.
Debug.Print Area 'Print Area to Debug window.
End Sub

```

NÜMUNƏ 2:

```

Sub Main()
  temp = Application.InputBox(Prompt:= _
  "Please enter the temperature in degrees F.", Type:=1)
  MsgBox "The temperature is " & Celsius(temp) & "degrees C."
End Sub

Function Celsius(fDegrees)
  Celsius=(fDegrees-32)*5/9
End Function

```

Function PROSEDURASI

VBA-da Function prosedurasının tərifi: verilmiş ad və arqumentləri (sabitlər, dəyişənlər, yaxud prosedurada olan ifadələr) olan, və vahid hissə kimi icra edilən hesablama təlimatları sırasına **Function** tipli prosedura deyirlər.

Sintaksisi:

```

[Public | Private | Friend] [Static] Function adı [(arqsiyahısı)] [As
tip]
[təlimatlar]
[ad = ifadə]
[Exit Function]
[təlimatlar]
[ad = ifadə]

End Function

```

Function təlimatının sintaksisi üç əsas hissədən ibarətdir, cədv. 3.10:

Cədvəl 3.10 **Function** təlimatının sintaksisindəki hissələrin təsviri.

Hissə	Təsviri
Public	Seçimlidir. Function prosedurasının bütün modullardakı proseduralar üçün əlçatan olduğunu bildirir. Əgər modulda Option Private rejimində saxlanılırsa, layihədən kənar yerdən əlçatan olmur.
Private	Seçimlidir. Function prosedurasının yalnız modulda elan edildiyi yerdə proseduralar üçün əlçatan olduğunu bildirir.
Friend	Seçimlidir. Yalnız sinif modulda istifadə edilir. Function prosedurasının layihədə görünür - obyekt nümunəsi kontrollerində görünür.
Statik	Seçimlidir. Function prosedurasının lokal dəyişənləri çağırıldığı zaman arasında yaddaşda saxlanılır. Static atributu dəyişənlərin Function -dan kənarında elan edildiyində heç bir hadisə baş vermir, hətta onlar prosedurada istifadə edilsə də.

adı	Məcburidir. Function prosedurasının adı; dəyişənlərin standart adlandırma qaydasına uyğundur.
arqsiyahısı	Seçimlidir. Function prosedurası çağırıldıqda onun proseduraya ötürüldüyü arqumentlərini təmsil edən dəyişənlərdən ibarət siyahı. Dəyişənlər çox olduqda, bir biri ilə vergül ilə aralanırlar.
tip	Seçimlidir. Function prosedurasının qaytardığı kəmiyyətin verilənlər tipini göstərir. Həmin verilənlər tipi bunlardan ola bilər: Byte, Boolean, Integer, Long, Currency, Single, Double, Decimal, Date, String, Object, Variant , yaxud istifadəçi tərəfindən təyin edilmiş digər başqa tiplər.
təlimatlar	Seçimlidir. Function prosedurası içərisində istənilən icra ediləsi təlimatlar qrupu.
ifadə	Seçimlidir. Function qiymətini qaytarır.

QEYD:

- Əgər aşkar olaraq VBA prosedurası **Public, Private, yaxud Friend**, elan edilməyibsə, onda standart vəziyyətdə prosedura **Public** tipinə aiddir. əgər lokal dəyişənlər **Static** elan edilməyibsə, onda aralıq çağırışlarda onların qiymətləri yaddaşda saxlanılır. **Friend** açar sözü yalnız sinif modullarında istifadə edilə bilər. Digər tərəfdən **Friend** proseduraları layihənin istənilən modulundan əlçatan olur. **Friend** proseduraları yaxud onların oxşar olan sinifləri tiplər kitabxanasında yerləşdirmək qadağandır.
- **Caution Function** proseduraları rekursiv ola bilər; yeni onlar özləri özlərini çağırır bilər. Rekursiya isə sonsuz sayda dövrlərə gətirib çıxara bilər. **Static** açar sözü adətən rekursiv olan **Function** proseduraları ilə birgə istifadə edilmir.
- Bütün icra edilən VBA kodu proseduranın daxilində olmalıdır. **Function** prosedurasını istifadəçi başqa **Function, Sub** və ya **Property** proseduralarında təyin edə bilməz.
- **Exit Function** təlimatı **Function** prosedurasından dərhal çıxılmasını təmin edir. Proqramın icrası **Function** prosedurasının çağırıldığı yerdən sonrakı təlimatların yerinə yetirilməyinə davam edəcək. **Function** prosedurasında istənilən sayda və istənilən yerdə **Exit Function** təlimatı yerləşdirilə bilər.
- **Sub** prosedurasına oxşar olaraq, **Function** prosedurası ayrı (təcrid) prosedurasıdır, təlimatlar sırasını yerinə yetirir və arqumentlərin qiymətlərini dəyişir. **Sub** prosedurasından fərqli olaraq, **Function** prosedurasını ifadənin sağ tərəfində istifadə etmək olar, məsələn, daxili standart funksiyalar **Sqr, Cos** yaxud **Chr** kimi.
- **Function** prosedurasını istifadəçi funksiyanın adı mötərizələr arasında arqumentlər siyahısı ilə ifadədə verə bilər (bunun üçün **Call** təlimatı verilir). Aşağıdakı VBA nümunəsində **BinarySearch** adı olan funksiyanın necə qaytarılan qiymətin təyin edilməsini göstərir (**False** adlı qiymət burada hansısa qiymətin tapılmadığını göstərir):


```

Function BinarySearch(. . .) As Boolean
. . .
' Value not found. Return a value of False.
If lower > upper Then
    BinarySearch=False
    Exit Function
End If
. . .
End Function

```

- **Function** prosedurasındaki dəyişənlər iki kateqoriyaya ayrılır: prosedurada aşkar olaraq, elan edilən və aşkar olaraq edilməyən. Prosedurada aşkar elan edilən dəyişənlər (**Dim**, və ya ona oxşarla) prosedurada həmişə lokaldır. Prosedurada aşkar edilməyən dəyişənlər hansısa daha yüksək səviyyədə kənar proseduradan elan edilməyənə qədər həmçinin lokal statuslu olur.

DİQQƏT: Prosedura aşkar edilməyən dəyişəni istifadə edə bilər, lakin, əgər modul səviyyəsində eyni adlandırma varsa, onda “adlandırma” konfliktli başlaya bilər. Məsələn, əgər istifadəçinin kodundakı dəyişənin adı başqa proseduranın adı, həmin proseduranın sabiti və ya dəyişəni ilə üst-üstə düşürsə, onda istifadəçinin prosedurası modul səviyyəli istinad etmiş olur. Belə konfliktlərin qarşısını almaq üçün, dəyişənlərin aşkar elan edilməsini gücləndirmək üçün, istifadəçinin **Option Explicit** təlimatından istifadə etməsi məqsədə uyğundur.

DİQQƏT: Effektivliyi artırmaq üçün VBA avtomatik rejimdə hesabi ifadələri dəyişdirə bilər. Əgər funksiya hansısa ifadədə dəyişənin qiymətini dəyişirsə, onda həmin ifadədə **Function** prosedurasının istifadə edilməsi məqsədə uyğun deyil.

Function PROSEDURASINA AİD NÜMUNƏLƏR (VBA sorğu materiallarından götürülüb):

NÜMUNƏ 1:

```

' The following user-defined function returns the square 'root of the
argument passed to it.
Function CalculateSquareRoot(NumberArg As Double) As Double
    If NumberArg < 0 Then      ' Evaluate argument.
        Exit Function      ' Exit to calling procedure.
    Else
        CalculateSquareRoot=Sqr(NumberArg) 'Return square root.
    End If
End Function

```

NÜMUNƏ 2:

```

' Əgər funksiyanın arqumentləri aşağıdakı kimi təyin edilirsə:
Function MyFunc(MyStr As String, Optional MyArg1 As _
Integer=5, Optional MyArg2="Dolly")
Dim RetVal
' Onda funksiya bu cür reaksiya verə bilər:
    RetVal=MyFunc("Hello",2,"World") 'All 3 arguments supplied.
    RetVal=MyFunc("Test", , 5)      '2-ci arqument buraxılıb.

```

```
'1-ci və 2-ci argumentdə adlardan istifadə edilib.  
RetVal=MyFunc(MyStr:="Hello ", MyArg1:=7)
```

Property PROSEDURASI

Property prosedurası proqramçıya özü tərtib etdiyi proqramdakı siniflərlə bağlı xassələri yaratmaq və idarə etmək imkanlarını yaradan Visual Basic təlimatlarından ibarət olan kodlar sırasına deyirlər.

SİNTAKSİSİ:

```
[Public | Private] [Static] Property {Get | Let | Set} _propertyadı  
[(argumentlər)] [As tip]
```

təlimatlar

End Property

Burada:

- **Property** proseduraları formaların, standart modulların və sinif modulların yalnız oxunma xassəli edilməsində istifadə edilir.
- Proqramdakı dəyişənlər sıra formasında olduqda, **Property** proseduraları **Public** dəyişənlərini əvəz etməlidir.
- **Public** dəyişənlərindən fərqli olaraq, **Property** proseduralarında **Object Browser**-ində təyin edilmiş **Help** sətirlərini təyin edir.

Property prosedurası yaradıldıqda o proseduranı özündə saxlayan modulun xassəsi kimi meydana çıxır. **Property** prosedurasında üç tipin təyin edilməsinə imkan var (cədv. 3.11) :

Cədvəl 3.11 **Property** prosedurasının tipləri.

Prosedura	Təsviri
Property Let	Bir xassənin dəyərini müəyyən edir.
Property Get	Xassənin dəyərini qaytarır.
Property Set	Obyektə istinad edilməsini təyin edir.

Property proseduraları adətən cüt-cüt istifadə edilir: kombinasiyalarda bunlar iştirak edir **Property Let**, **Property Get** və **Property Set**. **Property Get** təcrid halda elan edilməsi yalnız oxunma xassənin elanına oxşardır. Bütün **Property** proseduralarının birgə istifadəsi **Variant** tipli dəyişənlərinə çox faydalı ola bilər (əgər **Variant** tipi özündə obyekt və digər tipləri saxlayırsa). **Property Set** obyektlər ilə istifadə üçün nəzərdə tutulub; **Property Let** isə yox. **Property** prosedurasında tələb olan argumentlərin elan etmə sintaksisi cədv. 3.12-də göstərilib. Bu cədvəldə nəzərə alınmalıdır ki, birinci argumentin sonuncu argumentə qədər (1, ..., n) adı eyni olmalıdır, həmçinin verilənlər tipi eyni adlıdırsa, onda bütün **Property** proseduralarında eyni olmalıdır.

Cədvəl 3.12 **Property** prosedurasında tələb olan arqumentlərin elan edilmə qaydaları.

Prosedura	Elan edilmənin sintaksisi
Property Get	Property Get <i>propname</i> (1, ..., <i>n</i>) As <i>type</i>
Property Let	Property Let <i>propname</i> (1, ..., , , , <i>n</i> , <i>n</i> +1)
Property Set	Property Set <i>propname</i> (1, ..., <i>n</i> , <i>n</i> +1)

Property PROSEDURASINA AİD NÜMUNƏLƏR (VBA sorğu materiallarından götürülüb):

NÜMUNƏ 1:

```
Property Let Names(intX As Integer, intY As Integer, varZ_ As Variant)
    `Elan burada olur.
End Property
```

NÜMUNƏ 2:

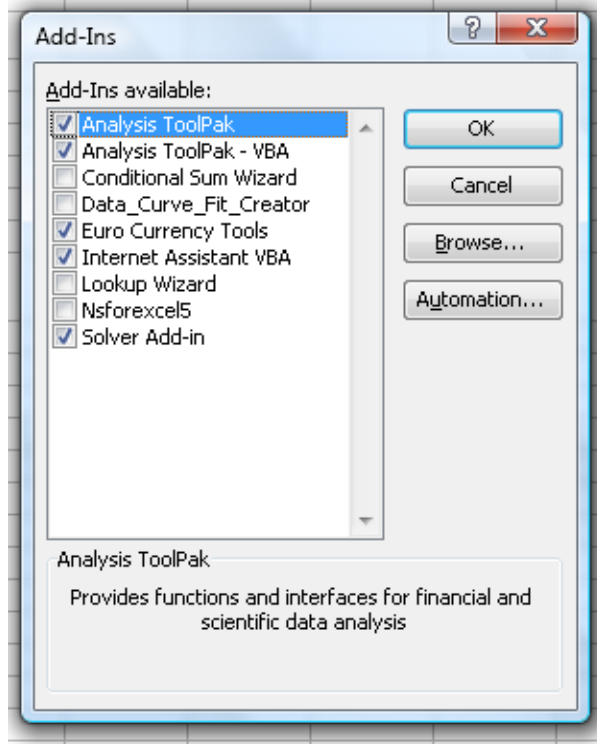
```
Property Get Names(intX As Integer, intY As Integer) As_ Variant
    `Elan burada olur.
End Property
```

ƏDƏDDİ QIYMƏTLƏRLƏ İŞLƏMƏK ÜÇÜN NƏZƏRDƏ TUTULAN QURULMUŞ STANDART VBA FUNKSIYALARI

VBA-nın adi (elmi-texniki) hesablamalarda istifadə edilən standart riyazi funksiyaları:
Abs () , Atn () , Sin () , ... , Int () , Fix () , Rnd () V.S.

Ədədi qiymətlərlə işləmək üçün VBA-da çox sayda qurulmuş standart funksiyalar var. Onlar praktiki məsələlərin proqramlaşdırılmasında tez-tez istifadə edilməyə də - bir çox tətbiqi elmi-mühəndis məsələlərin proqramlaşdırılmasında onlar ən vacib olan funksiyalardır.

Daha bir vacib amil budur ki, istifadəçi VBA proqramlaşdırılması ilə məşğul olmaq istəsə, onda istifadəçinin kompyuterində MS Office Excel proqramının yüklənməsi adi hal kimi nəzərə alınmalıdır. Bu proqramda isə, başqalarından fərqli olaraq, ədədlər üzərində əməllər aparmaq üçün çox güclü standart qurulmuş funksiyalar nəzərdə tutulmuşdur. Digər vacib amil bundan ibarətdir ki. həmin funksiyalar VBA-dan əlçatan vəziyyətdə olur. Aşağıda bir çox daha tez-tez işlənən funksiyalar cədv. 3.11-də verilib. Bundan əlavə, əgər Excel-də **Tools**→**Add-Insert** menyusunda **Analysis ToolPak** və **Analysis ToolPak-VBA** seçilsə (şək. 3.4), onda Excel-də əlavə olan maliyyə, elmi, texniki sahəsində istifadə edilən standart funksiyalar əlavə ediləcək. Üstəlik həmin proqramda (yəni Excel-də) yeni qurulmuş funksiyalar VBA-dan da əlçatan olacaq.



Şəkil 3.4 Excel-də istifadəçi tərəfindən, elmi-mühəndis sahəsində istifadə edilən standart funksiyaların əlavə edilməsini təmin edən dialoq pəncərəsi.

Aşağıda cədv.3.13-də verilən universal funksiyalar bütün Office proqramlarından əlçatandır:

Cədvəl 3.13 VBA-nın adi (maliyyə və elmi-texniki) hesablamalarda istifadə edilən standart riyazi funksiyalar

STANDART RIYAZI FUNKSIYALAR	ƏMƏLIYYAT
Abs (n)	n üçün mütləq qiymət qaytarır
Atn (n)	n üçün arktanqensin qiymətini radianla qaytarır.
Cos (n)	n üçün kosinusun qiymətini radianla qaytarır.
Exp (n)	e sabitinin n dərəcədə qiymətini qaytarır.
Fix (n)	n ədədini yuvarladaraq, ədədin kəsr hissəsini kəsib-atır
Int (n)	n ədədini yuvarladaraq, ona uyğun gələn ən kiçik qiyməti qaytarır.
Sgn (n)	n üçün əgər $n < 0$ onda -1 qaytarır, əgər $n = 0$, onda 0 qaytarır və əgər $n > 0$, onda $+1$ qaytarır.
Rnd (n)	n üçün $0 \dots 1$ intervalında təsadüfi ədədləri generasiyasını edir.
Round (n)	n ədədini, vergüldən sonra təyin edilmiş qiymətə qədər yuvarladaraq, qaytarır.
Sin (n)	n üçün sinusun qiymətini radianla qaytarır.
Sqr (n)	n üçün kvadrat kökün qiymətini qaytarır.
Str (n)	ədədi qiyməti sətərə çevirir.
Tan (n)	n üçün tangensin qiymətini radianla qaytarır.
Val (n)	sətiri ədədi qiymətə çevirir.

Cəd. 3.14-də isə əsas standart funksiyalar verilmişdir. Əslində bu siyahı daha böyük ola bilər, xüsusilə Office proqramlarına aid VBA-nın standart funksiyaları nəzərə alınsa. Onlardan VBA proqramlaşdırmasında daha intensiv istifadə edilənlər haqqında ətraflı məlumatı oxucu kitabın sonrakı hissələrində tapacaq. Yaddan çıxarmayaq ki, VBA-da standart qurulmuş funksiyaların sayı yüzlərlə hesablanır. Lazım olduqda VBA istifadəçisi daha müfəssəl məlumatı VBA sorğu materiallarında tapa bilər.

Cədvəl 3.14 MS Office VBA-da istifadə edilən standart qurulmuş funksiyaların siyahısı.

Abs	DDB	IsError	RGB
Array	Dir	IsMissing	Right
Asc	DoEvents	IsNull	RightB
AscB	Environ	IsNumeric	Rnd
AscW	EOF	IsObject	RTrim
Atn	Error	Lbound	Second
Cbool	Exp	Lease	Seek
Cbyte	FileAttr	Left	Sgn
Ccur	FileDateTime	LeftB	Shell
Cdate	FileLen	Len	Sin
CDbl	Fix	LenB	SLN
Cdec	Format	LoadPicture	Space
Choose	FreeFile	Loc	Spc
Chr	FV	LOF	Sqr
ChrB	GetAllSettings	Log	Str
ChrW	GetAttr	Ltrim	StrComp
Cint	GetAutoServerSettings	Mid	StrConv
CLng	GetObject	MidB	String
Command	GetSettings	Minute	Switch
Cos	Hex	MIRR	SYD
CreateObject	Hour	Month	Tab
CSng	Iif	MonthName*	Tan
CStr	IMEStatus	MsgBox	Time
CurDir	Input	Now	Timer
Cvar	InputB	Nper	TimeSerial
CVDate	InputBox	NPV	TimeValue
CVErr	InStr	Oct	Trim
Date	InStrB	Partition	TypeName
DateAdd	Int	Pmt	UBound
DateDiff	Ipmt	PPmt	UCase
DatePart	IRR	PV	Val
DateSerial	IsArray	QBColor	varType
DateValue	Isdate	Rate	Year

3.9.2 Proseduralarin görünmə sahəsi

VBA proseduralarının görünmə sahəsi, **Public**, **Private**, **Static xassələri** və **Option Private Module əmri**

Standart vəziyyətdə bütün VBA proseduraları (hadisələrin emalından başqa) *açıq* (**Public**) kimi təyin edilir. Deməli onları proqramın istənilən hissələrindən çağırmaq olar – həmin

moduldan, başqa moduldan, başqa layihədən. Proseduranı **Public** kimi bu cür elan etmək olar (nümunədə **Sub** prosedurasının adı `Farewell()` kimi verilmişdir):

```
Public Sub Farewell()
```

Digər tərəfdən, standart vəziyyətdə, proseduralar **Public** kimi təyin edilirlər deyə, onda onlar bu cür də təyin edilə bilərlər:

```
Sub Farewell()
```

Proseduranı lokal xassəli də vermək olar:

```
Private Sub Farewell()
```

Bu halda proseduranı yalnız həmin moduldan (onun yerləşdiyi yerdən) çağırmaq olar. Belə həll yolu, bəzən, başqa modullardan çağırılmaq üçün nəzərdə tutulmamış, proseduraların çağırılması ilə bağlı səhvlərin qarşısını ala bilər.

Bir layihənin çərçivəsində hansısa modulun açıq olan proseduralarının görünmə sahəsinə məhdudiyət qoymaq olar (istifadəçi tərəfindən **Public** tipi ilə təyin edilən proseduralara). Bunun üçün həmin modulun elanlar etmə hissəsində bir sətirin yazılması kifayətdir:

```
Option Private Module
```

Əgər proseduranın elanında **Static** açar sözündən istifadə edilsə, onda həmin proseduradakı bütün dəyişənlər, avtomatik olaraq, statik olacaq və prosedura işini bitirdikdə öz qiymətlərini saxlayacaq, məsələn, əgər aşağıdakı nümunə göstərilən kimi elan edilsə:

```
Private Static Sub Farewell()
```

3.9.3 Proseduraların elan edilməsinin avtomatlaşdırılması

VBA-da proseduraların elanı

Proseduranı “əl ilə” də elan etmək olar, məsələn, aşağıdakı nümunədə proqram kodunda `Farewell()` adlı prosedura üçün bir sətir yazmaqla:

```
Private Sub Farewell()
```

Bu üsulun üstünlüyü ondan ibarətdir ki, avtomatik olaraq, kodun sonunda `End Sub` sətiri əlavə ediləcək (əlbəttə, qrafik ekranda **Insert**→**Procedure** menyusundan da istifadə etmək olar, nəticə eyni olmalıdır).

3.9.4 Proseduralarda parametrlərin ötürülməsi

VBA-da prosedura və funksiyaların parametrlərin ötürülməsi (Optional), parametrlərin ötürülmə üsulları: istinadla (ByRef) və qiymətə görə (ByVal), parametrlərin ötürülməsində istinadların tətbiqi

Parametrlər – bir proseduradan başqasına ötürülən qiymətlərdir. Prinsip etibarı ilə yalnız modul səviyyəsində, dəyişənlərdən istifadə edərək, parametrlərsiz də proseduralar qurula bilər. Bu halda proqramın “oxunma qabiliyyəti” xeyli aşağı səviyyəyə düşəcək. Proseduranın parametrləri qəbul etmə imkanına malik olması üçün onu əvvəldən ötürüləsi parametrlərlə elan

etmək lazımdır. Məsələn, aşağıdakı misalda sadə funksiya prosedurası iki ədədi (parametr kimi almalıdır) toplayaraq, alınan nəticəni çıxarmalıdır (parametr kimi ötürməlidir).

```
Function fSum(nItem1 As Integer, nItem2 As Integer)
    fSum=nItem1+nItem2
End Function
```

Həmin funksiyanın başqa bir hansısa proseduradan çağırılması bu cür ola bilər:

```
MsgBox (fSum(3, 2))
```

Baxılan nümunədə hər iki daxil olunası parametrlər mütləq məcburi hal kimi elan edildi. Buna görə funksiya parametrlərinin ötürülməsi olmadan funksiyanın çağırılması (məsələn, bu cür: **MsgBox** (fSum(3))) hökmən səhvə gətirəcək: "Argument not optional" — "Parametr qeyri məcburi deyil". Hansısa parametrləri sərbəst buraxmaq üçün onları qeyri məcburi xassəli etmək lazımdır. Bunun üçün **Optional** açar sözündən istifadə edirlər:

```
Function fSum(nItem1 As Integer, Optional nItem2 As Integer)
```

VBA sorğu materiallarında qeyri məcburi olan parametrlər kvadrat formalı mötərizələrə alınır.

Qeyri məcburi parametrin ötürülməsi haqqında məlumat öyrənmək üçün **IsMissing** funksiyası (əgər bu parametr üçün **Variant** tipi istifadə edilibsə) yaxud onun qiyməti standart vəziyyətə görə olan dəyişənlərin qiyməti ilə müqayisə edilir (sıfır ədədi qiymələr üçün, boş sətir dəyişənləri üçün v.s.).

Ötürülən parametrlərlə olan funksiyanın çağırılması yuxarıdakı nümunə üçün bu cür ola bilər:

```
nResult=fSum(3, 2)
```

Burada biz bir neçə vacib olan hallara daha ətraflı baxmalıyıq.

Yuxarıda baxdığımız nümunədə parametrlər pozisiya ilə ötürülür, yəni 3 qiyməti birinci parametr (**nItem1**), 2 qiyməti isə ikinci (**nItem2**) mənimsəyir. Lakin parametrləri adları ilə də mənimsətmək olar, məsələn, aşağıdakı nümunədəki kimi:

```
nResult=fSum(nItem1:=3, nItem2:=2)
```

Bu nümunədə oxucu diqqət etməlidir ki, mənimsəmə əməliyyatı C++ dilinin sintaksisinə uyğun işarə ilə aparılmışdı (:=). Adi bərabərlik işarəsi qoyulsa (=) səhv haqqında xəbər verilmiş olacaq. Əlbəttə, aşkar olaraq qiymələrin mənimsənməsi (yuxarıdakı misalda 3 və 2 qiymətlərinin mənimsənməsi) əvəzinə dəyişənlərin özünü istifadə etmək olar. Bəs bu dəyişənlər funksiyada "olduqlarından" sonra dəyişənlər hansı qiymətləri alacaq? Funksiyadan kənar vəziyyətdə həmin dəyişənlərin qiymətləri əvvəlki kimimi olacaq, yaxud dəyişəcək? Hər şey ondan asılıdır ki, parametrlər *istinadla* necə ötürülür – standart vəziyyətdə **ByRef**, qiymətə görə isə **ByVal** açar sözündən istifadə etmək olar. Əgər parametrlər istinadla ötürülürsə, onda, faktiki olaraq, maşının operativ yaddaşında çağırılan prosedura üçün həmin dəyişənlərə istinadlar verilir. Əgər həmin dəyişənin qiyməti çağırılan prosedurada dəyişə, onda çağırılan prosedurada da həmin parametrin qiyməti dəyişəcək. Bu isə VB-da qəbul edilmiş standart vəziyyətdə olan davranışdır.

Əgər parametrlər qiymətə görə ötürülsə, onda maşının operativ yaddaşında həmin dəyişənin əksi (kopiyası) yaranır və o, çağırılan proseduraya ötürülür. Həmin kopya ilə nə edilsə - ilkin dəyişənə heç bir təsir olmayacaq və çağırılan prosedurada bununla bağlı heç bir dəyişiklik olmayacaq. Sadə misalda fərqi nümayiş etmək olar:

```
Private Sub TestProc()  
'nPar1 adlı dəyişəni elan edərək, ona, məsələn, 5 qiymətini mənimsədək  
Dim nPar1 As Integer  
    nPar1=5  
'Onu nItem1 parametri kimi fSum funksiyasına ötürək  
    MsgBox(fSum(nItem1:=nPar1, nItem2:=2))  
'İndi isə yoxlayaq görəək nPar1 dəyişəni fSum funksiyasında olduğdan  
'sonra onunla nə baş verdi:  
    MsgBox nPar1  
End Sub  
Function fSum(nItem1 As Integer, nItem2 As Integer)  
'Dəyişənin qiymətini istifadə edirik  
    fSum=nItem1+nItem2  
'İndi isə onun qiymətini dəyişirik!  
    nItem1=10  
End Function
```

Funksiyada elan etmə sətirini dəyişdirdikdə nə baş verəcəyini oxucu müstəqil olaraq yoxlasa bu işdən çox böyük fayda qazanmış olar:

```
Function fSum(nItem1 As Integer, nItem2 As Integer)
```

sətirini aşağıda göstərilən sətirlə əvəz etsə:

```
Function fSum(ByVal nItem1 As Integer, nItem2 As _ Integer)
```

Başqa maraqlı məqam bundan ibarətdir ki, VBA kompilyatoruna funksiyanın nəyi qaytardığını istifadəçinin marağında olmadığını nümayiş etmək olar. Bunun üçün onun parametrlərini dairəvi mötərizələr arasında yazmaq lazımdır. Məsələn, **MsgBox** qurulmuş standart funksiyası üçün həmin effekti bu cür almaq olar:

```
MsgBox "Test"
```

Yuxarıda gətirilən funksiya üçün isə:

```
fSum 3, 2
```

Bu cür kod tamamilə normal işləyəcək. İstifadəçiyə **MsgBox** funksiyasının hansı qiyməti qaytardığını bilmək lazım olduqda, onda funksiya ötürülən parametrlər gerek dairəvi mötərizələrdə verilsin. Məsələn, aşağıdakı misaldakı kimi:

```
nTest=MsgBox ("Test")
```

QEYD: Əksər hallarda VBA kompilyatoru qurulmuş standart funksiyalar üçün qaytarılan qiymətləri inkar etməyə imkan vermir: parametrləri dairəvi mötərizələrdə verilməsini məcbur edir.

3.9.5 Proseduraların işə salınması və tamamlanması

VBA-da proseduraların işə salınması və tamamlanması üsulları, End Sub konstruksiyası, Exit Sub konstruksiyası.

Proseduraların və funksiyaların koddan çağırılması nümunələri ilə biz artıq rast gəlmişik, məsələn:

```
nResult=fSum(3, 2)
```

Prosedura/funksiya-nın adının yazılması kifayətdir: ona tələb olan parametrləri ötürmək və funksiya olduqda işə tələb olan qiymətləri qəbul etmək. Bəzi proseduraları digərlərindən işə salmaq olar. Bəs baş proqramı (özül proqramı) istifadəçi hansı üsul ilə işə salmalıdır?

Bunun üçün VBA-da aşağıdakı imkanlar var:

- makrosu yaratmaq (yəni **NewMacros** modulunda heç parametr qəbul etməyən xüsusi prosedura yaratmaq) və onu öz adı ilə işə salmaq. Başqa üsul budur: düymə ilə yaxud klaviatura düymələrinin kombinasiyası ilə (bax 1.7-ci bölməyə) işə salmaq. Sonra isə makros başqa proseduraları çağıra bilər;
- VBA formasını yaratmaq və həmin forma ilə bağlı hadisələr toplusundan və idarəetmə elementlərindən istifadə etmək, yaxud Excel səhifəsinin və ya Word sənədinin idarəetmə elementlərindən istifadə etmək;
- proseduraya xüsusi ad vermək (`AutoExec()`, `AutoNew()` v.s.). Bu cür adların tam siyahısını VBA sənədləşməsindən tapmaq olar. Nəzərə alınmalıdır ki, bu cür imkanlardan çox geniş virus tərtib edənlər istifadə edir. Lakin Office 2003-də standart vəziyyətdə belə növ makroslar öz-özünə işə salına bilməz. Bütün makrosların işə salınmasına icazə vermək üçün istifadəçi gərək **Tools**→**Macros**→**Security** menyusunda dəyişiklik etməlidir (bax 1.7-ci bölməyə);
- makrosa xüsusi ad vermək əvəzinə ona xüsusi hadisədən istifadə etmək olar: məsələn, proqramın işə salınması, hansısa əməliyyatın tamamlanması v.s. Proqram kodunun avtomatik işə salınması üçün Microsoft özü bu cür üsulu məsləhət görür.

Proseduranı `/m` parametri ilə və makrosun adı ilə əmr sətirindən də işə salmaq olar, məsələn, bu cür:

```
winword.exe/mMyMacros
```

VBA-da belə vəziyyət də yarana bilər ki, prosedura özü-özünü işə salır. Bu cür vəziyyətlərdən istifadəçi mümkün qədər kənar durmalıdır. Problemin kökü – proseduranın oxunma qabiliyyətindədir və proseduranın özü-özünün sonsuz dəfə çağırıldığında maşının operativ yaddaşının tükənməsi, ardınca yaranan sistem səhvləri ilə rast gəlmə ehtimalı böyük ola bilər.

VBA prosedurasının tamamlanması üçün **End** və **Exit End** konstruksiyalarından istifadə etmək lazımdır. Sintaksis olduqca sadədir:

Exit Sub

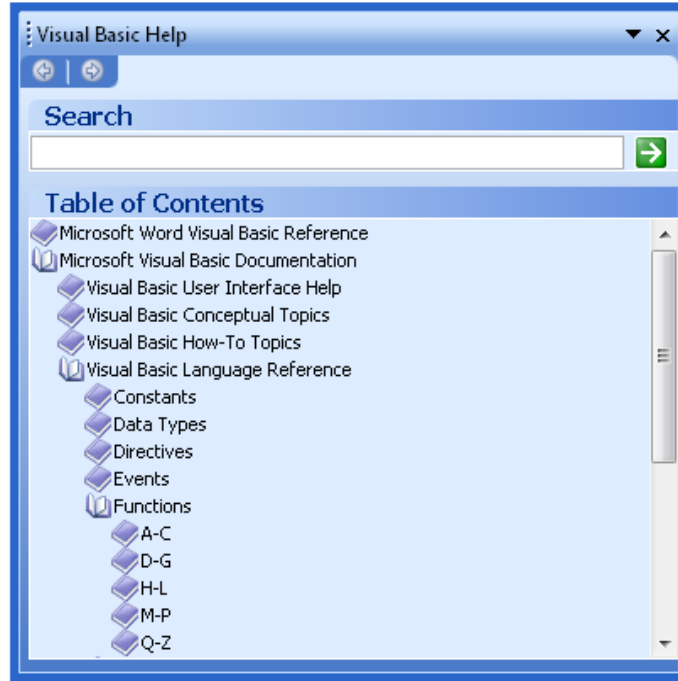
End Sub

Bu konstruksiyalar eyni işi icra edir – cari proseduranın işini dayandırır (tamamlayır). Onlar ayrı-ayrı hallarda istifadə edilir:

- **End** operatoru – hər bir nəzərdə tutulmuş əməl qurtardıqda prosedura işini qurtarır, dayandır. **End** operatoru VBA kodunun son sətirində yerləşdirilir;
- **Exit** operatoru – proseduranın işlədiyi zaman qəflətən, dərhal dayandırılmasını icra edir. Adətən bu operatoru şərti keçid idarəetmə blokunda yerləşdirirlər, məsələn, məlum olsa ki, həmin yoxlama blokunda prosedura hansısa səbəbə görə öz işini davam edə bilməz (məsələn, sonrakı addımda səhvləri emal edən kod işə salınmalıdır).

3.9.6 VBA proqramlaşdırmasında daha önəmli olan və tez-tez istifadə edilən standart funksiyalar

VBA proqramlaşdırma dilində, yuxarıda (3.9 bölmənin funksiyalarla bağlı hissəsində), deyilmişdir ki, VBA-da onlarla *qurulmuş standart funksiyalar* vardır və bəziləri haqqında qısa məlumat verilmişdir (bax cə. 3.11 və cə. 3.12). Praktiki proqramlaşdırma işlərində peşəkar proqramçılar ən önəmli sayılan funksiyalardan daha çox istifadə edirlər. Bu hissədə biz daha tez-tez istifadə edilən funksiyalarla tanış olacağıq. Əgər istifadəçi öz işində digər VBA funksiyalarından istifadə etməyə məcbur olsa, onda yuxarıda qeyd etdiyimiz kimi, VBA sənədləşməsində (sorğu materiallarında) həmin funksiyalar haqqında daha müfəssəl məlumat tapa bilər. Yada salırıq ki, həmin sorğu materialları yalnız İngilis dilindədir, başqa dildə (xüsusilə Azərbaycan, Türk və ya Rus dilində) VBA sənədləşməsində heç bir sorğu materialları yoxdur. Nəzərə alınmalıdır ki, bu materialları istifadəçi istənilən Office proqramlarının VBA sənədləşməsində də tapa bilər: Excel, Word, Access yaxud, məsələn, hətta AutoCAD-da. Adı çəkilən funksiyaların növü həqiqətən çox aktiv istifadə edilir və onlarsız bəzən proqramın qurulması mümkün olmur. Peşəkar VBA proqramçıları onları tamamilə avtomatik olaraq istifadə edirlər. Aydınır ki, yeni başlayan proqramçılar həmin funksiyaların xassələrini ətraflı öyrənmək istəyirlər – bu işə çox təbiidir, çünki funksiyaların xassələrini mükəmməl bilmədən VBA-da effektiv proqram tərtib etmək mümkün deyil. Həmin funksiyaların (oxucu sonrakı hissələrdə onlarla ətraflı tanış olacaq) ətraflı öyrənilməsinin daha bir vacib səbəbi ondan ibarətdir ki, onlar adi Visual Basic-də, VBScript-də və hətta eyni adlarla və sintaksislə didər proqramlaşdırma dillərində də rast gəlir, məsələn: C++, Delphi, Java, JavaScript v.s. VBA sənədləşməsində qurulmuş standart funksiyalar əlifba qaydası ilə qruplaşdırılıb (şək. 3.5).



Şek. 3.5 VBA sənədləşməsinə qurulmuş standart funksiyalar haqqında məlumat almaq üçün sorğu materiallarının dialog pəncərəsi.

Aşağıda VBA-nın daha aktiv istifadə edilən funksiyaları haqqında məlumat veriləcək. Riyazi funksiyalar olan - sinus, kosinus və ya tangens haqqında əlavə məlumat verilməyəcək, çünki praktiki işlərdə (elm-mühəndislik sahəsi ilə bağlı olmayan işlərdə) bu funksiyalar demək olar ki, tamamilə istifadə edilmir. Tam sintaksisi də burada verməyəcəyik - oxucu sorğu materiallarına istinad edə bilər. Əsas məqsədimiz budur ki, oxucu həmin funksiyaların nə etdiyini və hansı hallarda onlardan istifadə edilməsinin vacib olduğunu aydın anlasın. Aşağıdakı bölmələrdə qurulmuş standart VBA funksiyalarını funksionallıq xassələrinə görə qruplaşdırdıq.

3.9.7 Çevirmənin və tiplərin yoxlanmasını təmin edən funksiyalar

Qurulmuş standart çevirmə və tiplərin yoxlanmasını təmin edən funksiyalar:
`CBool()`, `CByte()`, `CCur()`, `CDate()`, `Cdbl()`, `CDec()`, `CInt()`,
`CLng()`, `CSng()`, `CStr()`, `CVar()`, `CVDate()`, `CVErr()`, `Str()`, `Val()`,
`IsNumeric()`, `IsDate()`, `IsEmpty()`, `IsError()`, `IsMissing()`,
`IsNull()`, `IsObject()`, `IsArray()`, `Hex()`, `Oct()`

VBA proqramlaşdırılmasında bir tipin qiymətini başqa tipə çevirmək əməliyyatları tez-tez rast gəlir. Bir neçə tipik olan vəziyyətləri göstərik:

- `InputBox()` vasitəsilə istifadəçidən alınan sətir olan qiyməti ədədi qiymətə keçirilməsi;
- İstifadəçilərin kompyuterindəki regionların xassələrinin qurulmasından asılı olmayaraq, tarix/zaman qiymətlərini istifadəçi sətir qiyməti kimi keçirmək istədikdə;
- əksinə, sətir qiymətlərini tarix/zaman qiymətləri kimi keçirmək lazım olduqda.

Verilənlər tipini çevirmək üçün istifadə edilən funksiyaların əksəriyyətinin adı **C** hərfi ilə başlayır (Convert sözündən əmələ gəlib + verilənlər tipinin adı): **CBool()**, **CByte()**, **CCur()**, **CDate()**, **Cdbl()**, **CDec()**, **CInt()**, **CLng()**, **CSng()**, **CStr()**, **CVar()**, **CVDate()**, **CVErr()**.

TypeName() funksiyası ilə nə alındığını sonra yoxlamaq olar:

```
nVar1=CInt(InputBox("Qiyəti daxil edin"))  
MsgBox TypeName(nVar1)
```

Funksiyaların tipinin çevirməsinə aid bundan əlavə bir neçə xeyirli məsləhət də vermək olar:

- **Str()** - ədədi qiyməti sətir qiymətinə keçirir. Tamamilə **CStr()** etdiyini edir (müsbət ədədlər üçün boş yer qoyur);
- **Val()** - ədəd və işarələr qatışığından yalnız ədədi qiymətləri "çıxarır". Bununla belə funksiya verilənləri soldan sağa oxuyur və birinci rast gəldiyi qeyri ədədi qiymətdə dayanır (yeganə qeyri ədədi qiymətə imkan verilir – tam hissəni kəsrdən ayıran nöqtəyə). Bu funksiyanın istifadəsi ədədi qiymətlərlə birgə ölçü vahidləri və valyuta qiymətləri verildikdə çox faydalı olur.
- çevirmədə səhv əmələ gəlməsin deyə qiymətləri əvvəldən çevirməyə imkan olmalarını **IsNumeric()** və **IsDate()** funksiyaları ilə yoxlamaq olar;
- qiymətlərin xüsusi təyin edilmiş qiymətlərə uyğun gəlməyini **IsArray()**, **IsEmpty()**, **IsError()**, **IsMissing()**, **IsNull()** və **IsObject()** funksiyaları ilə yoxlamaq olar: **True** və ya **False** qiymətlərini qaytarır;
- onluq hesabla verilən qiymətləri sətir ifadədə on altı və səkkizlik hesabdakı qiymətlər tipinə keçirmək üçün **Hex()** və **Oct()** funksiyalarından istifadə edilməlidir. Geriyə çevirmək üçün xüsusi funksiyalar yoxdur - istifadəçi VBA kompilyatoruna işarə edə bilər ki, həmin ədədlər on altılıq və ya səkkizlik hesab sistemində yazılıb, məsələn, onları bu cür yazmaq olar: **&O12** və **&HA**.

3.9.8 Sətir funksiyaları

VBA-nın sətir funksiyaları: **Asc()**, **Chr()**, **InStr()**, **Len()**, **LCase()**, **UCase()**, **Replace()**, **Trim()**

Bunlar daha tez-tez istifadə edilən funksiyalardır. Onlar həmişə tələb edilir, və onları yaxşı bilmək lazımdır.

- **ASC()** - bu funksiya ötürülən işarəyə ədədi kodu qaytara bilər. Məsələn, **ASC("D")** 68 qiymətini qaytarır. Bu funksiya əvvəlki yaxud sonrakı işarənin təyin edilməsində istifadə etmək çox sərfəli olur. Adətən o, **Chr()** funksiyası ilə birgə istifadə edilir (o işarənin ötürülən koduna görə onun özünü qaytarır). Məsələn, bu cür kod Excel-də

A1-dən A20 hücrələrinə İngilis əlifbasının hərflərini, ardıcıl olaraq, A-dan Z-ə qədər, yazılmasını təmin edir:

```
Dim n, nCharCode As Integer
n=1
nCharCode=Asc("A")
Do While n<=26
ActiveWorkbook.ActiveSheet.Range("A"&n).Value=_
Chr(nCharCode)
n=n+1
nCharCode=nCharCode+1
Loop
```

Bu funksiyanın variantları bunlardır - **AscB()** və **AscW()**. Həmin funksiyalar birincisi işarənin ədədi kodunun yalnız birinci baytını qaytarır, ikincisi isə Unicode kodlaşmasında işarənin ədədi kodunu qaytarır.

- **Chr()** - çox vacib funksiya. Ədədi koda görə işarənin özünü qaytarır. Asc() funksiyası ilə birgə işləməsindən əlavə başqa bir vəziyyətdə də onsuz keçinmək mümkün deyil: xidməti işarə daxil edilməsi vacib olduqda, məsələn, Word-də tutaq ki, "AzEnerqo" sözünün çap edilməsi lazım olduqda (burada dırnaq arası işarə xidməti işarədir). `Selection.Text=""AzEnerqo""` sətirinin istifadə edilməsi dərhal sintaktik səhv edilməsi haqqında xəbər verəcək. Əgər bu cür yazılsa, onda hər şey öz qaydasında olacaq:

```
Selection.Text=Chr(34) & "AzEnerqo" & Chr(34)
```

Həmin funksiyanın **ChrB()** və **ChrW()** variantları da var. Onlar da eyni qaydada **Asc()** funksiyası üçün işləyirlər.

- **InStr()** və **InStrRev()** - ən populyar olan funksiya. Sətir dəyişəninin bədənində işarələr ardıcılığını təyin edib dərhal onun pozisiyasını qaytarır. Əgər ardıcılıq təyin edilməsə, onda 0 qiyməti qaytarılır.
- **Left()**, **Right()**, **Mid()** - mövcud olan sətir dəyişənindən sol tərəfdən, sağdan və ya ortadan təyin edilən sayda işarələrin götürülməsini təmin edir.
- **Len()** - sətirdəki işarələrin sayını təyin edir. Bir çox hallarda dövrü idarəetmə operatorları ilə birgə istifadə edilir.
- **LCase()** və **UCase()** - sətiri yuxarı və ya aşağı registrə keçirilməsini təmin edirlər. Tez-tez qiymətlərin müqayisə edilməsi üçün hazırlıq işlərində istifadə edirlər (registrin hansı vəziyyətdə olmağı vacib olmayanda, məsələn, soy adları, firmalar və şirkətlərin adlarında, şəhərlərin adında v.s).
- **LSet()** və **RSet()** - bir dəyişəni işarələrlə dolduranda o birisinin isə uzunluğunu dəyişməz saxladıqda (soldan və ya sağdan) bu funksiyalardan istifadə edirlər. Ariq olan işarələr kəsilərək çalışmayanların yerinə boşluqlar qoyulur.

- **LTrim()**, **RTrim()**, **Trim()** - bu funksiyalar boşluq olan yerlərin, uyğun olaraq, soldan, sağdan və ya həm soldan, həm sağdan götürülməsini təmin edirlər.
- **Replace()** - sətirdə bir işarə ardıcılığını başqası ilə əvəz edilməsini təmin edir.
- **Space()** - təyin edilmiş boşluqlar sayından sətirin alınmasını təmin edir;
- **String()** - təyin edilmiş işarələr sayından sətirin alınmasını təmin edir. Adətən çıxarışın formatlanmasında **Len()** funksiyası ilə birgə istifadə edilir. Daha bir oxşar funksiya - **SpC()** funksiyasıdır: çıxarış konsola getdikdə formatlaşmanı təmin edir (əmr sətirinin uzunluğuna müvafiq olaraq, boşluqları çoxaldır).
- **StrComp()** - iki sərin müqayisə edilməsini təmin edir.
- **StrConv()** - sətiri çevirməyə imkan verir (Unicode-da və geriye, yuxarı və aşağı registre, birinci işarənin baş işarə etmək v.s.).
- **StrReverse()** - sətiri "geriyə çevirmək" əməlini yerinə yetirir (yeni işarələrini əks qaydada yerləşdirmək).
- **Tab()** - konsola çıxarış olanda formatlamaq üçün nəzərdə tutulmuşdur. İstifadəçi təyin etdiyi sayda tabulyasiya işarələrini çoxaldır (əgər heç bir say göstərilərsə, onda sadəcə tabulyasiya işarəsi yerləşdirir). Sətir qiymətinə tabulyasiya işarəsini qoymaq üçün daha bir üsul var: `vbTab` sabitini istifadə etmək.

3.9.9 Tarix və zamanla işləməyi təmin edən funksiyalar

Tarix/zaman verilənləri ilə işləyən VBA funksiyaları: `Date()`, `Time()`, `DateAdd()`, `DateDiff()`, `DatePart()`, `DateSerial()`, `Timer()`

Praktiki işlərdə tarix və zamanla bağlı funksiyalardan istifadə edilməsə heç bir proqramlaşdırma dilində effektiv və tam yararlı proqram tərtib etmək mümkün deyil. Buna görə VBA istifadəçisi həmin qurulmuş funksiyaları yaxşı bilməlidir. Onlardan ən vaciblərini icra etdikləri əməllərin izahatı aşağıda verilib:

- **Date()** - cari sistem tarixini qaytarır. Onu eyni adlı operator ilə quraşdırmaq olar:
`Date=#5/12/2006#`
- **Time()** - cari sistem vaxtını qaytarır;
- **Now()** - cari sistem tarixini və zamanını birgə qaytarır;
- **DateAdd()** - hansısa tarix üçün nəzərdə tutulmuş illərin sayını, rübləri və ayları əlavə etmək imkanını yaradır (saniyəyə qədər);
- **DateDiff()** – tarixlər arasında olan fərqi almağa imkan yaradır (tarix il ilə verildikdə, onda saniyə ölçü vahidi ilə fərqi qiymətləndirir);

- **DatePart()** - çox mühüm funksiyadır, istifadəçi təyin etdiyi tarixin müəyyən bir hissəsini qaytarır (məsələn, yalnız ili, yalnız həftəni, yalnız ayı və ya yalnız günü);
- **DateSerial()** – işarə qiymətləri əsasında tarix yaratmaq imkanını verir. Həmin əməliyyatı **DateValue()** funksiyası da yerinə yetirir – fərq yalnız qəbul olan qiymətlərin formatındadır. Analoji qaydada (zaman qiymətləri üçün) **TimeSerial()** və **TimeValue()** funksiyaları fəaliyyət göstərir;
- **Day()** (həmçinin **Year()**, **Month()**, **Weekday()**, **Hour()**, **Minute()**, **Second()**) – bu funksiyalar tarixin hansısa istənilən hissəsini qaytarırlar (**DatePart()** xüsusi funksiyasını əvəz edirlər);
- **MonthName()** - sözlər ilə ay adını həmin ayın sıra nömrəsinə görə qaytarır. Qaytarılan qiymət kompyuterdəki regional sazlanmadan asılıdır (Azərbaycan, Rus, İngilis, Türk v.s. dilində);
- **Timer()** – gecənin yarısından keçən saniyələrin sayını qaytarır;

Tarix/vaxt qiymətləri ilə işləmək üçün əlavə imkanlar almaq üçün istifadəçi gerek Outlook obyektini modellərindən istifadə etsin. Məsələn, bu cür istifadəçi dünyanın bir çox ölkələrində bayram günləri və ya iş/qeyri-ış günləri haqqında lazımı məlumat əldə edə bilər. Müvafiq bölmədə oxucu bu haqda daha çox məlumat əldə edəcək.

3.9.10 Verilənləri formatlayan funksiyalar

VBA-nın formatlayan funksiyalar, Format() funksiyası.

Verilənləri formatlamaq üçün istifadəçinin ixtiyarında bu funksiya var: **Format()** və **Format** adı ilə başlayan bir sıra digər funksiyalar (**FormatNumber()**, **FormatCurrency()**, **FormatDateTime()**, və s.). Onların sintaksisi bu cür ifadə edilir:

Format (ifadə, "format")

Aşağıda buna aid bir neçə nümunə göstərilib:

```
Format(15/20, "Percent")
Format(Date, "Long Date")
Format(1, "On/Off")
Format(334.9, "###0.00")
Format("Sadə mətn",>)
```

Xüsusi vəziyyət belə yarana bilər - regional parametrlərdən asılı olmayaraq bütün kompyuterlərdə verilənlər eyni formatda olmalıdır. Bunun üçün həll yolu kimi **DatePart()** funksiyasından istifadə etmək olar: tarixi hissələrdən ibarət halda mətn formatına çevirmək və sonra lazımı qaydada formatı quraşdırmaq.

3.9.11 İstifadəçi ilə qarşılıqlı əlaqələr yaradan funksiyalar

VBA-da istifadəçi ilə yaradılması: MsgBox() və InputBox() funksiyaları.

Bir çox VBA proqramlarında istifadəçi ilə qarşılıqlı əlaqənin (dialogun) yaradılması vacib məsələ kimi meydana çıxır - onu hansısa hadisənin başlanması (qurtarması) haqqında məlumatlandırmaq və ondan müvafiq reaksiyanın bu və ya digər formada almaq. İstifadəçi üçün sadəcə Office proqramının pəncərəsində məlumatı çap etmək olar (misal üçün, mövcud Word sənədində) yaxud forma və idarəetmə elementlərində istifadə etmək olar. Bu haqda daha ətraflı məlumatı oxucu müvafiq fəsilərdə tapacaq. Bu bölmədə isə yalnız həmin əməliyyatları yerinə yetirən qurulmuş standart VBA funksiyalarına baxacağıq.

İstifadəçini məlumatlandırmaq üçün ən sadə üsul - **MsgBox()** qurulmuş standart funksiyasından istifadə etmək. Oxucu artıq bir neçə dəfə verilən VBA kodu nümunələrində bu funksiya ilə kitabda rast gəlib - funksiyanın tam sintaksisi isə aşağıda verilir:

```
MsgBox(Mətn[,düymələr] [,pəncərənin başlığı] [,sorgu_ faylı, sorgu faylında işarə etmə])
```

MsgBox() funksiyasının imkanları çox genişdir:

- Müxtəlif sayda düymələri dialog pəncərəsində göstərmək olar (OK, Cancel, Abort, Retry, Ignore, Yes, No);
- **Critical, Warning, Question, Information** işarələrini göstərmək olur;
- standart vəziyyətə görə düymə seçilə bilər;
- pəncərəni adi və ya modal etmək olur.

MsgBox() - dan qaytarılan qiymətlərə aid bir nümunəyə baxaq:

```
Dim nVar As Integer  
nVar=MsgBox("Başlamaq olar?",65,"Nümayiş məlumat_ pəncərəsi")
```

Əgər nVar=1, onda istifadəçi OK düyməsini basıbmiş, digər halda nVar=2 olduqda, onda istifadəçi **Cancel** düyməsini basıbmiş.

Bəzən (məsələn, paket rejimində) bir müddətdən sonra məlumat pəncərəsinin özünün örtülməsi məqsədə uyğun olur. Bunu **Wscript.Shell** obyektinin **Popup()** üsulu ilə etmək olar. Həmin məqsədlə **References** menyusunda **Windows Script Host Object Model** ilə layihəyə istinad etmək lazımdır (faylın ünvanı əksər hallarda burada yerləşir **C:\WINNT\system32\wshom.ocx**), sonra isə aşağıdakı koddan istifadə etmək lazımdır:

```
Dim oShell As New WshShell  
oShell.Popup "Test", 5
```

Başqa alınan pəncərənin xassələri **MsgBox()** -la eynidir. İstifadəçi heç bir düyməyə basmasa qayıtma kodu 1-ə bərabərdir. Başqa üsulda **InputBox()** funksiyasından istifadə edərək istifadəçidən məlumat alınır. Bu üsul çox sadədir:


```

Dim Input
Input=InputBox("Adınızı daxil edin:")
MsgBox(" Siz daxil edibsınız:"&Input )

```

InputBox() -da dəvət mətnini, pəncərənin başlığını, standart vəziyyətə görə qiyməti, pəncərənin yerləşdiyi yeri və sorğu faylı göstərmək mümkündür. Yaddan çıxarmamaq lazımdır ki, istifadəçinin bütün daxil etdiyi verilənləri **InputBox()** funksiyası sətir tipinə çevirir – bu halda geriye əks çevirmə lazım ola bilər.

İstifadəçinin diqqətini audio siqnalla da cəlb etmək olar. Bu məqsədlə **Beep** operatoru istifadə edilir:

```

Dim I
For I=1 To 3
    Beep
Next I

```

Əgər istifadəçi ilə daha mürəkkəb əlaqələrin qurulması lazımdırsa, onda formadan istifadə etmək olar, sənədin özünü və ya köməkçidən (**Office Assistant**) istifadə etmək olar. Bu sahədə Internet **Explorer**-in obyekt modelləri daha geniş imkanlar yaradır.

3.9.12 Sintaktik konstruksiyaları əvəz edən funksiyalar

VBA funksiyaları: Choose(), IIF(), Switch()

VBA-da bir neçə funksiya vardır ki, onların köməyi ilə **IF ... THEN ... ELSE** yaxud **SELECT ... CASE** sintaktik konstruksiyalarını əvəz etmək olur. Bu funksiyaların tətbiqi hər hansı böyük üstünlük vermir. Ancaq bəzən peşəkar proqramçılar həmin funksiyalardan istifadə edirlər. Lakin başlayan istifadəçilər adi sintaktik konstruksiyalardan istifadə etsələr daha yaxşı nəticə alınar. Bəzən başqalarının proqramlarını sərbəst oxumaq üçün həmin funksiyaları bilmək lazım olur. Aşağıda onların izahlı siyahısı verilib:

- **Choose()** - ədədi (qiymətin nömrəsini) və bir neçə qiyməti qəbul edir. Ötürülən ədədə uyğun gələn sıra nömrəsini qaytarır. Məsələn, **Choose(2, "Birinci", İkinci", "Üçüncü")** qaytardığı qiymət olacaq "İkinci".
- **IIF()** - funksiyası "*Immediate IF*" kimi yozulur, yəni "Dərhal **IF**. **IF**-in sadə variantıdır – şərt yoxlanılaraq, iki qiymətdən birisi qaytarılır. Məsələn:

```
IIf(n>10,"Ondan böyükdür","Ondan kiçik yaxud bərabərdir")
```

- **Switch()** - ifadə/qiymət tipli sonsuz sayda cütü qəbul edir (hər ifadə doğruluğa yoxlanılır) və birinci doğru olan ifadənin qiymətini qaytarır, məsələn:

```

Function Language (CityName As String)
    Language=Switch(CityName="Bakı", "Azərbaycan", CityName=_
        "Paris", "Fransuz", CityName="London", "İngilis")
End Function

```

3.9.13 Çoxluqlarla (Arrays) işləmək üçün funksiyalar

VBA-da çoxluqlarla işləməyi təmin edən funksiyalar: `Array()`, `Filter()`, `LBound()`, `UBound()`, `Join()`, `Split()`

Qeyd edildiyi kimi, Microsoft Office proqramlaşdırmasında çoxluqlar az tətbiq edilir. Onların əvəzinə kolleksiyalardan istifadə edirlər. Buna baxmayaraq, VBA-da çoxluqlarla işləmək üçün bir neçə vacib funksiya vardır:

- **Array()** - avtomatik olaraq lazımı ölçülü və tipli çoxluğu yaratmağa və dərhal ona ötürülən qiymətləri yükləməyə imkan verir, məsələn:

```
Dim myArray As Variant
myArray=Array(10,20,30)
MsgBox(A(2))
```

- **Filter()** - bir çoxluqdan başqasını almağa (əvvəlcədən ilkin çoxluqda tələb olan elementləri filtrasiya edərək);
- **LBound()** - çoxluğun aşağı sərhədi haqqında xəbər verir (yəni onun tərkibindəki birinci elementin nömrəsini qaytarır);
- **UBound()** - çoxluğun yuxarı sərhədi haqqında xəbər verir (yəni onun tərkibindəki sonuncu elementin nömrəsini qaytarır);
- **Join()** - çoxluğun sətirlər sıralarını hansısa bir sətir dəyişəni ilə əvəz etməyə imkan yaradır. Ayrıca işarə standart vəziyyətdə boşluqdur - istifadəçi isə öz işarəsini təyin edə bilər;
- **Split()** - yuxarıdakı funksiyanın əks əməllərini yerinə yetirir: bir sətirdən çoxluq yaradır. Bu funksiya (həmçinin `Join()` funksiyası) verilənlər bazasından, elektron cədvəldən və ya maket fayldan alınan qiymətlərin emalında olduqca faydalıdır.

3.9.14 Fayl sistemi ilə işləmək üçün olan funksiyalar

VBA-nın fayl funksiyaları: `Input()`, `FileLen()`, `EOF()`, `LOF()`, `Loc()`

VBA-da faylları, kataloqlar, diskler və başqa fayl sisteminin digər obyektləri ilə işləmək üçün qurulmuş standart funksiyalar nəzərdə tutulub. Bu funksiyalar haqqında məlumat aşağıda verilib. Lakin unutmamaq lazımdır ki, bu imkanlardan əlavə, bizdə həmin tətbiqi proqram üçün spesifik olan imkanlar lazımdır (məsələn, Word sənəd obyekt modeli vasitəsilə Word faylının açılması və yaddaşda saxlanması kimi). İkincisi - hər bir kompyuterdə **Microsoft Scripting Runtime** adlı Windows nəzarəti altında obyektli kitabxana vardır - fayllar, kataloqlar və disklerle müxtəlif əməliyyatları yerinə yetirmək üçün çox sadə və asan sistemdir. Bu sistem haqqında ətraflı məlumatı oxucu İnternetdə www.microsoft.com/scripting ünvanında olan Microsoft-un rəsmi saytında tapa bilər.

VBA-da fayl sistemi ilə işləməyi təmin edən funksiyalar izahatı ilə aşağıda verilib:

- **CurDir()** - tətbiqi proqramın faylları standart vəziyyətdə saxlanılması yerin yolunu qaytarır.
- **Dir()** - diskdə təyin edilən yolla faylın yaxud kataloqun axtarılmasını təmin edir;
- **EOF()** - faylın sonunda olduqda diskdəki fayla yazılma əməliyyatı zamanı **True** qiymətini qaytarır. Öz formatını fayla yazdıqda daha çox istifadə edilir. Word sənədi, Excel kitabı v.s. proqramlarında yaddaşda saxlandıqda - daha yaxşı olar ki, bu sənədlərin obyektlərində olan standart metodlarından istifadə edilsin: **Save** və **SaveAs()**.
- **Error()** - səhvin nömrəsinə görə onun tam izahını qaytarır;
- **FileAttr()** - fayl sisteminde istifadəçi tərəfindən faylın necə açılmasını təyin edir: oxumağa, yazmağa, mətn və ya ikili kod rejimində v.s.
- **FileDateTime()** - əgər fayl yaradıldıqdan sonra ona heç bir istinad edilməyibsə, onda onun yaranma zamanını həmin vaxtdan başlayaraq təyin edir.
- **FileLen()** - təyin edilmiş faylın baytla uzunluğunu qaytarır;
- **FreeFile()** - sırada olan sərbəst ədədi təyin etməyə imkan yaradır (fayl açılarkən həmin ədədi faylın nömrəsi kimi istifadə etmək olur);
- **GetAttr()** - fayla, fayl sistemi ilə əlaqə yaratmaq və onun atributları (gizli, yalnız oxumaq statusu ilə, arxiv, və s.) haqqında məlumatı almaq üçün imkan yaradır;
- **Input()** - açıq fayldan məlumatı oxumağa imkanı yaradır. Məsələn, açılmış olan **C:\text1.txt** faylından məlumatı oxumaq və **Immediate** pəncərəsinə həmin məlumatı çıxarmağı aşağıda göstərilən VBA nümunəsindəki kimi həyata keçirmək olar:

```

Dim MyChar
'Open() funksiyası ilə faylı oxumaq üçün açırıq
Open "c:\text1.txt" For Input As #1
  Do While Not EOF(1) 'nə qədər ki, fayl qurtarmayıb
  'hər dəfə bir işarə alaraq, onu əvvəlkinə əlavə edirik
  MyChar=MyChar&Input(1, #1)
  Loop
Close #1 'Faylı bağlayırıq
'Immediate pəncərəsinə onun 'tərkibini çıxarıyıq
  Debug.Print MyChar

```

- **InputB()** - fayldan yüklənəsi baytların sayını göstərir;
- **Loc()** - açılmış faylda yerləşdirmə və ya oxunma cari yeri təyin edir və həmin yeri müəyyən edən ədədi qaytarır. Buna oxşar **Seek()** funksiyası işləyir - o növbəti cari oxunma və ya yerləşdirmə yerinin nömrəsini təyin edir;
- **LOF()** - açılmış faylın baytla uzunluğunu təyin edir.

3.9.15 VBA-nın digər faydalı funksiyaları

VBA funksiyaları: `DoEvents()`, `Environ()`, `GetAllSettings()`, `Partition()`, `QBColor()`, `RGB()`, `Shell()`, `TypeName()`, `VarType()`

Bu bölmədəki funksiyalar hansısa bir kateqoriyaya aid olmasalar da Office proqramlaşdırmasında vacib rol oynayırlar. Aşağıda onların izahlı siyahısı verilib:

- **DoEvents()** - çox vacib funksiyadır. Müəyyən zaman çərçivəsində hər hansı VBA əməliyyatından alınaraq, idarəetməni əməliyyat sisteminə verilməsini təmin edir (bununla müəyyən yığılmış hadisələrin icrası yerinə yetirilə bilər, məsələn, istifadəçi tərəfindən basılmış klaviatura düyməsinin icrası v.s.). Bundan sonra VBA əməliyyatları davam edəcək. Xüsusi ilə bu funksiya belə bir vəziyyətdə çox faydalı ola bilər: istifadəçi olduqca zaman aparan əməliyyatı aparır (disklərdə axtarış, böyük həcmdə olan verilənlərin emalı v.s.) – istifadəçi isə dərhal işin müvəqqəti dayandırılmasını istəyir. Bunun üçün həmin funksiyanın köməyindən istifadə edərək, işin müəyyən porsiyası yerinə yetirildikdə “arakəsmələri” yaratmaq olur.
- **Environ()** - kompyuter mühitinin dəyişənlərinin mütləq yolunu qaytarır (dəyişənlərin tam siyahısı əmr sətirində SET əmri verildikdə alınır).
- **GetAllSettings()** - istifadəçi təyin etdiyi əlavəyə (tətbiqi proqrama) aid bütün parametrləri (iki ölçülü matris (array) şəkilində) alınmasını təmin edir. **SaveSetting()** həmin məlumatı reyestrə yazılmasına imkan yaradır, **DeleteSetting()** isə silir. **GetSetting()** – hansısa parametr haqqında məlumat almağa imkan verir. Alternativ yol budur: daha effektiv olan **Windows Script Host Object Model** obyekt kitabxanasından istifadə etmək lazımdır (istənilən Windows əməliyyat sistemində bu utilit vardır).
- **Partition()** - ötürülən qiymətin qiymətlər toplusunda hansı diapazona aid olduğunu və həmin diapazonun sətir formasında təsvirini qaytarır. Adətən bu funksiyadan verilənlər bazasına sorğular həyata keçirilən zaman istifadə edirlər.
- **QBColor()** - rənglərin əski olan nömrəli işarə etməsindən (16-lı hesab sistemində) müasir RGB koduna çevirməsini təmin edir. Adətən köhnəlmiş, lakin dəyərli sayılan proqramların yenidən bərpa etməsində istifadə edilir.
- **RGB()** - rəng kodunun bərpa edilməsini təmin edir, rəng ilə işləmək üçün üç sayda rəng istifadə edilir: qırmızı (**Red**), yaşıl (**Green**) və göy (**Blue**). Hər bir əsas rəng üçün qiymətlər diapazonu 0-dan 255-ə qədər dəyişə bilər. Misal üçün, daha çox yaşıl (tünd yaşıl) rəngi almaq üçün funksiyanın parametrləri (argumentləri) belə olmalıdır: **RGB (0, 255, 0);**
- **Shell()** - VBA-dan kənar yerləşən proqram faylının işə salınmasını (icra edilməsini) təmin edir, əməliyyat sistemində proqramın ID haqqında məlumatın qaytarılmasına imkan verir. Adətən peşəkar proqramçılar bu funksiyanı Windows API kateqoriyalı

sistemlərin imkanlarını proqramlarda yoxladıqdan sonra tətbiq edirlər. Praktiki tərəfdən bu funksiyanın köməyi ilə istifadəçi işlədiyi Office proqramından istənilən kənar proqramı işə sala bilər. Microsoft məsləhət görür ki, daha effektiv olan **Windows Script Host Object Model** obyekt kitabxanasının **WshExec** və **WshShell** obyektlərindən istifadə edilsin. Bu haqda daha ətraflı məlumat Microsoft-un saytında www.microsoft.com/scripting ünvanında tapmaq olar.

- **TypeName ()** - bu funksiya ona ötürülən dəyişənin tipinin adını qaytarır. Verilənlər bazasından alınan və ya hansısa obyektin metodunu çağırdıqda müvafiq qiymətin tipinin təyin edilməsinə imkan yaradır.
- **VarType ()** – yuxarıdakı funksiya ilə eyni əməlləri yerinə yetirir, lakin tipin adı əvəzinə tipi işarə edən ədədi kodu qaytarır. Proqramçılar bu funksiyanı proqramlardakı dəyişənlərin tipini yoxladıqda istifadə edirlər.

4. OBYEKT MODELLƏRİ VƏ OBYEKT LƏRLƏ İŞLƏMƏ QAYDALARI

Kitabın giriş hissəsində qeyd etmişdik ki, VBA obyekt yönü proqramlaşdırma (OYP) dilləri kateqoriyasına aiddir. Bəs OYP nədir? Obyektlərin köməyi ilə tətbiqi proqramların yaradılmasını, layihələndirilməsini və proqram tərtibatını, tətbiqi proqramın müxtəlif vəziyyətlərdə davranışını analiz edən metodikanı həyata keçirən kompleks məzmunlu üsula OYP demək olar. VBA-da obyekt dedikdə nəyi birinci növbədə anlamaq lazımdır? Obyekt - verilənləri emal edəsi kodla (proqramla) birgə baxılmasına imkan yaradan bir vasitədir. Yəni, əslində obyekt verilənləri və proqram kodunu birləşdirərək, ona hansısa bir vahid mühit kimi baxılmasına imkan verir və həmin mühitin özünü də elə obyekt adlandırırlar. Beləliklə, VBA dili tam ciddi olaraq, obyekt yönü proqramlaşdırma dilidir - başqa müasir dillərdən fərqli olaraq, obyekt üsulları bu dildə böyük rol oynayır. Məsələn, Excel-dəki bütün görünən obyektlər **İş kitabı (workBook)**, **İş səhifəsi (worksheet)**, **Aktiv hücrə (ActiveCell)**.

Qeyd: Bəzən informatika ilə bağlı Azərbaycanca olan mənbələrdə xana və ya dama terminləri işlədilir. Türk elmi terminologiyasındakı isə “hücrə” termini eyni ilə İngilis dilindəki “cell” sözünün mənasına uyğun gəlir və həmçinin Azərbaycan dilində də eynigüclüdür – məhz buna görə biz də İngilis dilində “Cell” termininə daha çox uyğun olan hücrə sözündən istifadə edəcəyik.

Başqa vacib olan obyektlər bunlardır: **Diapazon (Range)**, **Diagram (Chart)**, **Forma (UserForm)** - ən çox işlənən obyektlərdir. VBA-da 100-dən artıq qurulmuş obyekt vardır (hər bir Office proqramında özünə məxsus sayda və funksionallıq imkanlarında).

Bir neçə sayda obyekt (adətən eyni tipli olan) öz tərkibində saxlayan obyektə **Toplum** obyekt (Collection obyekt) deyirlər. Məsələn, **workBooks** obyekt (İş kitabları) bütün **workBook (İş kitabı)** obyektlərini öz tərkibində saxlayır. Hər bir toplum elementi nömrələnir və adı ilə yaxud

nömrəsi ilə audentifikasiya edilə bilər. Məsələn, **worksheets (1)** aktiv kitabın birinci iş kitabı deməkdir, a **worksheets (Sheet1!)** - **Sheet1** adlı iş səhifəsi deməkdir.

OLE və **ProgID** (Programming Identificators - *programlaşdırma identifikatorları* mənasına gəlir)

VBA-da **OLE** (Object Linking and Embedding – bağlama və obyektlerin yerləşdirilməsi) mexanizmi istifadə edilir [1, 2, 3, 4, 6, ..., 18]: bununla **OLE** xassəli bütün proqramlarla əlaqə saxlanması mümkün olur. **OLE** mexanizmi ilə inteqrə olan elementlər nümunəsi kimi WordArt, ClipArt v.s. proqramların köməyi ilə **OLEObject** obyektlərinin yerləşdirilməsini gətirmək olar. İş kitabının bütün **OLE**-obyektleri **OLEObjects** toplusunu yaradır. “Əl ilə” olan üsulla **OLE**-obyektleri iş səhifəsində **Insert Object** dialoq pəncərəsində yaranan **Object** əmrini seçərək istifadəçi təyin edilmiş yerə obyekt yerləşdirə bilər. **OLE**-obyektinin əsas xassəsi budur ki, istifadəçi onun görüntüsünü aktivləşdirdikdə (bunun üçün mausun göstəricisi ilə obyektin üstündə bir dəfə sağ düymə ilə şıqqıltı etmək kifayətdir) əslində bu obyektə bağlı olan proqram aktivləşir və dərhal proqramın menyusu yaradılmış tətbiqi proqramın menyusu ilə əvəz olur. Sonrakı mərhələdə istifadəçi əsas Office proqramından çıxmayaaraq, obyekt yaradan proqramın vasitələri ilə obyekt redaktə etməyə və görkəmini dəyişməyə imkan alır. Bundan əlavə **OLE**-texnologiyası **Automation** (avtomatlaşdırma) xassəsinə malikdir. Bu üsulla yerləşdirilmiş obyekt üçün, adi əlavə obyekt kimi, xassələrini təyin etmək, üsulları tətbiq etmək və hadisələri emal etmək imkanları yaranır.

1996-cı ildən başlayaraq, Microsoft yeni terminologiyayı istifadə etməyə başladı: **OLE**-obyektini indi **ActiveX** obyektini və **OLE Automation** indi **ActiveX Automation** adlanır. Obyektlərin avtomatlaşdırılmasını seçdikdə **OLE** programlaşdırma identifikatorları (**ProgID**) elementlərindən istifadə etmək lazım gəlir. **ActiveX** idarə etməsi üçün aşağıdakı komponentlərdən istifadə edilir: **ActiveX Controls**; **Microsoft Excel**; **Microsoft Graph**; **Microsoft Word**; **Microsoft PowerPoint**; **Microsoft Access**; **Microsoft Outlook**; **Microsoft Web Components**. **ActiveX** idarəetməsi üçün cədv. 4.1-dəki **OLE** identifikatorları istifadə edilir:

Cədvəl 4.1 VBA-dakı **ActiveX** idarəetməsində **OLE** programlaşdırma identifikatorları

ActiveX VASITƏLƏRİ	İSTİFADƏ EDİLƏN İDENTİFİKATORLAR
CheckBox	Forms.CheckBox.1
ComboBox	Forms.ComboBox.1
CommandButton	Forms.CommandButton.1
Frame	Forms.Frame.1
Image	Forms.Image.1
Label	Forms.Label.1
ListBox	Forms.ListBox.1
MultiPage	Forms.MultiPage.1
OptionButton	Forms.OptionButton.1
ScrollBar	Forms.ScrollBar.1
SpinButton	Forms.SpinButton.1
TabStrip	Forms.TabStrip.1

Əlbəttə, yüksək obyektönlü proqramlaşdırma dili kimi MS VBA-da çox sayda (minlərlə) hazır obyektlərdən - komponentlərdən istifadə edilir. Proqram koduna (mətninə) baxanda bu komponentlərin hər birinin öz adının olduğu və bu ad ilə ona müraciət olunduğunu görməmək mümkün deyil. MS VBA ilə Office proqramlaşdırılması mühitində işləyən proqramçı həmin zaman bu obyektlərin adları ilə yox onların piktoqramları (şəkilləri) ilə daha çox işləyir.

VBA sinifləri. OYP-in ən vacib anlayışlarından biri siniflər anlayışıdır. VBA-da sinif dedikdə adətən müəyyən bir obyektə yaradan layihə kimi təsəvvür etmək daha düzgün olar. Beləliklə, sinif obyektin adını, xassələrini və onun icra etdiyi hərəkətləri müəyyən edir. Öz növbəsində, hər bir obyekt, yuxarıda göstəriləndi kimi, hansısa sinfin nümunəsidir (eyni tipli hissəsidir və ya elementidir).

Microsoft Office proqramlarında VBA proqramlarının funksionallığı bütövlükdə siniflər və obyektlərlə işləmə üzərində qurulub. Formal olaraq, siniflərin bu cür anlanması daha dürüst olardı: *siniflər VBA proqramlarında istifadə edilən funksionallıq bloklarıdır*. Obyektlər yaradıldıqda onlar bir növ sxemləmə, cizgiləmə (Rus terminologiyasında "çertyoj") rolunu oynayır. Belə sxemlər əsasında siniflər nümunəsi olan - obyektlər yaradırlar. Həmin sxemlərin (siniflərin) toplusuna isə VBA terminologiyasında tiplər kitabxanası (**Type Library**) deyirlər. Windows-da onları **.dll** yaxud **.ocx** fayllarında saxlayırlar (bəzən də başqa tip fayllarda saxlayırlar, məsələn, **.exe** yaxud **.tlb**).

VBA proqramlarında daha tez-tez rast gəlinən hal **müəyyən sinifə aid olan obyektin yaradılması** (İngiliscə **instantiation**, yəni bizim dildə nümunə yaradılması mənasına gəlir) ilə bağlı olur. Bir proqramda bir sinifə aid bir neçə sayda obyektin yaradılması mümkündür. Bu kitabda biz Office proqramlaşdırmasında daha çox işlənən sxemlər əsasında yaradılmış standart qurulmuş (Microsoft-un istifadəçi üçün hazırladığı siniflər) obyektlərdən danışacağıq. Çünki istifadəçinin özünün siniflər yaratması mövzusu əslində ayrıca olan çox böyük mövzudur və bu dərs vəsaitinin kursunun çərçivəsindən çıxır.

Obyektlərin iyerarxiyası. VBA kitabxanasında, əvvəl qeyd etdiyimiz kimi, 100-dən artıq obyekt var. Onların hər birinin iyerarxiyik yeri və xassəsi var. İyerarxiya obyektlər arasında olan əlaqəni və onlara əlçatma yollarını müəyyən edir, şəx. 4.1. Obyektə tam istinad bir neçə bir birinin içərisinə yerləşdirilmiş adlar sırasından ibarətdir, arakəsmələr nöqtələr yerinə yetirir. Sıra **Application** obyektə adından başlayır və obyektin öz adı ilə qurtarır. Məsələn, aşağıdakı nümunədə **Kafedra** adlı Excel iş kitabının **Shet1** səhifəsinin **A1** hücrəsinə tam istinad verilib:

```
Application.Workbooks ("Kafedra").Worksheets ("Sheet1").Range ("A1")
```

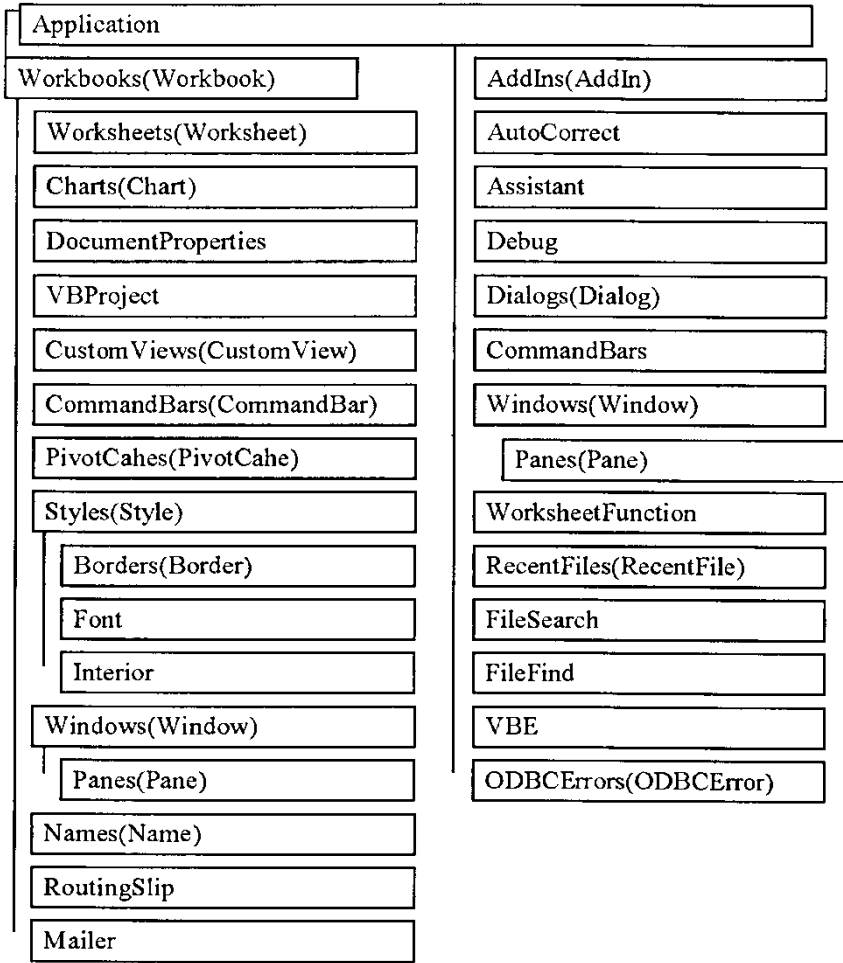
Bunu bilmək lazımdır ki, hər dəfə obyektə tam istinad vermək məcburi deyil. Adətən obyektə qeyri aşkar istinadla kifayətlənmək olar. Qeyri aşkar istinadda, tam istinaddan fərqli olaraq, həmin anda aktiv olan obyektlərin adı verilməyə də bilər. Məsələn, yuxarıdakı tam istinadı qeyri aşkar istinadla bu cür vermək olar:

```
Workbooks ("Kafedra").Worksheets ("Sheet1").Range ("A1")
```

Əgər həmin anda iş kitabı **Kafedra** aktivdirsə, onda qeyri aşkar istinad daha da qısa ola bilər:

Worksheets("Sheet1").Range("A1")

Üstəlik, əgər həmin anda iş səhifəsi Sheet1 də aktivdirsə, onda qeyri aşkar istinad ən qısa formada yazıla bilər: Range("A1")



Şəkil 4.1 VBA-da qurulmuş obyektlərinin iyerarxiyası.

4.1 Obyektlərin yaradılması və yox edilməsi

VBA-da obyektlərin yaradılması, erkən və gec bağlanma (early/late binding), CreateObject strukturu və New açar sözü, obyektlərin yox edilməsi, Nothing açar sözü

VBA-da obyektlər müxtəlif üsulla yaradıla bilər. Əvvəlcə ən sadə üsulu öyrənək, sadə nümunə aşağıda verilib:

```
Dim oApp As Object
Set oApp=CreateObject("Word.Application")
MsgBox oApp.UserName
```

Bu nümunə gec bağlanmaya aiddir (**late binding**). Əvvəlcə biz istənilən obyektə istinad etmə imkanı olan oApp dəyişənini elan edirik. Sonra isə bizim tərəfimizdən yaradılan Word.Application obyektinə istinad etməni həmin oApp dəyişəninə mənimsədirik. Burada, nəzərə alınmalıdır ki, həmin komponent obyektlərin adındakı sintaksisdə nöqtə işarəsi əslində iki obyektə bir birindən ayıran funksiyayı yerinə yetirir: kompleks obyekt iki elementdən ibarətdir

– Word və Application. Sintaksisdə həmçinin nəzərə alınmalıdır ki, hər bir ayrıca element böyük hərfdən başlanmalıdır. Bu cür gec bağlanma operativ yaddaşın məhsuldarlığı və səmərəliliyi baxımından daha pis üsuldür. Üstəlik, bu halda VBA redaktoru da bizə heç bir məsləhət vermir: məsələn, bu obyektin hansı xassələri və metodları vardır. Buna görə gec bağlanmanı yalnız o halda tətbiq etmək lazımdır ki, proqramın məntiqinə uyğun olaraq, həmin dəyişənin içərisində (tərkibində) bir neçə müxtəlif tipli obyekt saxlanmalıdır. Lakin digər başqa hallarda - erkən bağlanma (**early binding**) üsulu daha üstün sayılmalıdır. Buna aid sadə nümunə aşağıdakı kodda verilib:

```
Dim oApp As Word.Application
Set oApp=CreateObject("Word.Application")
MsgBox oApp.UserName
```

Bu halda isə biz oApp dəyişəninə həmin an (əvvəlcədən) **Word.Application** tipini mənimsədirik. Sonra isə yaratdığımız obyektə istinadı mənimsədirik. Bu halda (ikinci nümunədə) kodun ikinci sətirini yazmamaq da olar, məsələn, bu cür:

```
Dim oApp As New Word.Application
MsgBox oApp.UserName
```

Nəzərə alınmalıdır ki, VBA sintaksisinə görə **New** və **WithEvents** açar sözü qurulmuş tiplərin (**String**, **Int** v.s.) və asılı olan obyektlərin yaradılmasında birgə olaraq istifadə edilə bilməz. Buna görə hərdən **Set** operatoru ilə elan etmə lazım olacaq. Bundan əlavə, nəzərə alınmalıdır ki, **VBScript** sintaksisində **New** operatoru yoxdur. Əgər siz hər iki proqramlaşdırma dilini istifadə etmək qərarına gəlimsə, onda bəri başdan **CreateObject()** strukturu ilə işləməyi öyrənmək lazımdır.

Başqa bir üsul da budur: - *digər obyektin metodundan istifadə edilməsi ilə yeni obyektin yaradılması*. Bu halda üsul özü yaradılmış obyektə istinadı bir başa verir, yaxud kolleksiyanın içərisindən qaytarır, məsələn, aşağıdakı nümunədə olan kimi:

```
Dim oApp As New Word.Application
oApp.Documents.Add
Dim oDoc As Word.Document
Set oDoc=oApp.Documents(1)
oDoc.SaveAs"C:\docvbal.doc"
```

Yuxarıdakı nümunədə biz əvvəlcə **Word.Application** obyektini yaradıırıq, sonra isə **Documents** kolleksiyasının **Add()** metodu ilə həmin kolleksiyada yeni sənədi yaradıırıq. Növbəti addımda həmin sənədə istinadı **oDoc** adlı dəyişəninə təyin edirik. Nəhayət, son addımda, yırdığımız obyektin **SaveAs()** metodunu çağırırıq.

Obyektlərin yox edilməsi (silinməsi) çox asan yolla həyata keçirilir, aşağıdakı sintaksisə uyğun olaraq:

```
Set Obyekt dəyişəni=Nothing
```

Məsələn, real vəziyyətdə, bu cür öp adlı obyektini yox etmək olar:

```
Set öp=Nothing
```

Prinsip etibarı ilə obyektı yox etməmək də (silməmək də) olar – həmin obyektə istinad edən sonuncu obyekt dəyişəni görükmə sahəsindən kənara çıxdığı an (yeni dəyişəni istifadə edən prosedura işini bitirdiyi andan başlayaraq), avtomatik olaraq, yaradılmış obyekt, faktiki olaraq, yox edilmiş (silinmiş) vəziyyətdə olacaq. Obyektlərin, lazım olduqda, yox edilməsi proqramlaşdırma mədəniyyətində təqdire layiq əməliyyat sayılır. Ciddi tətbiqi proqramlar yaradılarda bununla təcrübəli istifadəçi resursların israfı və adların konfliktli kimi xoşa gəlməz hallardan özünü sığortalamış olacaq. Obyektlərin silinməsi (yox edilməsi) ilə bağlı daha bir vacib qeyd bundan ibarətdir:

Qeyd: bütün obyektləri aşağıdakı sintaksis ilə *yox etmək mümkün olmur*.

Set Obyekt_dəyişəni=**Nothing**

Bəzən bunun üçün başqa xüsusi üsulları tətbiq etmək lazım gəlir. Məsələn, yuxarıdakı nümunələrdəki yaradılmış `Word` obyektini silmək üçün mütləq bu obyektin `Quit()` adlı xüsusi metodunu tətbiq etmək lazımdır – çünki digər halda VBA kompilyatoru səhv haqqında xəbər verə bilər.

4.2 Obyektin metodları

VBA obyektlərinin metodları, metodların çağırılma üsulları, parametrlərlə işləmə qaydaları

Obyekt öz-özlüyündə böyük məna kəsb etmir. Daha böyük məna bundan ibarətdir: *obyekt üzərində hansı əməlləri aparmaq olar və həmin obyektin hansı xassələri vardır*. Obyektin metodu terminini işlətdikdə isə həmin obyektin hansı xassəyə malik olduğunu və onun üzərində hansı əməllərin aparılmasının mümkünlüyünü demiş oluruq.

VBA-da obyektin metodunun tətbiqinin ən sadə sintaksisi:

Obyekt.Metod

Məsələn, aşağıdakı nümunədə `Quit` (*bağlamaq, çıxmaq* mənasına gəlir) metodu ilə proqram (`Application` obyektini) bağlanır, işini dayandırır, proqramdan çıxma mümkün olur:

Application.Quit

Qruplaşmanın bütün obyektlərinə *obyekt metodunu* tətbiq etmək olar. Məsələn, aşağıdakı misalda `Sheet1` iş səhifəsinin `chartobjects` (Diaqramlar) qruplaşmasına `Delete` (silmək, yox etmək) metodunun tətbiqi nəticəsində `Sheet1` iş səhifəsindəki bütün diaqramların silinməsi mümkün olur:

Worksheets ("Sheet1").ChartObjects.Delete

Yadda saxlamaq lazımdır ki, bir qayda olaraq, proqramlaşdırmada obyektı bizə yalnız ondan ötrü lazımdır ki, onun metodlarından, xassələrindən və hadisələrindən istifadə etmək mümkün

olsun. Beləliklə, metod – verilmiş obyektin yerinə yetirə biləsi əməllərin adlandırılmış toplusuna deyirlər. Metod hansısa əməlləri yerinə yetirərək, qiymətləri qəbul edib qaytara bilir.

VBA-da obyektin metodunu icra üçün çağırılmasının 3 əsas sintaksisi vardır:

- Yuxarıdakı, ən sadə sintaksis kimi gətirilən, üsula birinci üsul deyək:

```
Obyekt.Metod
```

məsələn:

```
oDoc.Activate
```

Bu sintaksisdə heç bir parametr qəbul edilmir və geriye qaytarılmır.

- İkinci üsulda parametrlər ötürülür:

```
Obyekt.Metod Parametr1[,Parametr 2, ... , Parametr]
```

Parametrlər sayılaraq, bir-birilə vergüllə aralanır və moduldan modula ötürülür. Məsələn, aşağıdakı misaldakı kimi:

```
oDoc.SaveAs "D:\doc12.doc"
```

bu halda metodun nəyi qaytardığı nəzərə alınmır, buna görə dairəvi mötərizələr lazım olmur.

- Üçüncü üsul - parametrlər ötürülür və metodun qaytardığı qiyməti dəyişən mənimşeyir:

```
Mənim_dəyişənim=Obyekt.Metod(Parametr1[,Parametr 2_ , ... ,  
Parametr])
```

Aşağıdakı nümunədə həmin üsuldən istifadə edilib:

```
Dim nCent  
nCent=oApp.CentimetersToPoints(10)  
MsgBox nCent
```

Bu halda ötürülən parametrlər üçün dairəvi mötərizələr məcburi qoyulmalıdır. Heç bir parametr ötürülmədikdə belə dairəvi mötərizələr boş qalmalıdır, məsələn, bu cür:

```
Mənim_dəyişənim=Obyekt.Metod()
```

4.3 Obyektin xassələri

VBA obyektlərinin xassəsi, xassələrə qiymətlərin təyin edilməsi, xassələrin tipləri

Xassə obyektin atributunu təmsil edir: əsas parametrlərini, məsələn, ölçüsünü, rəngini, ekranda tutduğu vəziyyətini və obyektin ümumi vəziyyətini (əlçatanlığı və görülmə qabiliyyəti). Obyektin parametrlərini dəyişmək lazım olduqda, onun xassələrinin qiymətlərinin dəyişdirilməsi kifayətdir. Beləliklə, obyektin xassəsinin tərfi bu cür səsənə bilər: **obyektə saxlanılan**

məlumatın alınmasını, obyektin parametrlərinin dəyişdirilməsini təmin edən struktura obyektin xassəsi deyirlər.

Obyekt haqqında olan məlumatın çıxarılmasını təmin edən VBA sintaksisi:

Dəyişən=Obyekt.Xassə

məsələn,

sName=oApp.UserName

Obyekt haqqında olan məlumatın dəyişdirilməsini təmin edən VBA sintaksisi:

Obyekt.Xassə=Qiymət

məsələn:

oApp.ActivePrinter="HP LaserJet 4"

Qiymət adi sabit ola bilər (10 və ya "HP LaserJet 4"), sadə bir hesabi ifadə ola bilər (10+5), başqa obyektin xassəsi ola bilər:

Obyekt1_1.Xassə=Obyekt_2.Xassə

hansısa metodun qaytardığı qiymət ola bilər:

Obyekt.Xassə=Obyekt_1.Metod().

Əlbəttə, bütün xassələrin qiymətini dəyişmək olmaz: VBA-da bəzi xassələr yalnız oxunma və ya yalnız yazılma, digərləri həm yazılma, həm də oxunma üçün əlçatan olur.

Obyektin xassəsinin qiymətini qurmaq üçün nəzərdə tutulan VBA sintaksisi:

Obyekt.Xassə=Xassənin_qiyməti

Məsələn, aşağıdakı nümunədə **Application** obyektinin **Caption** xassəsinə qiymət verərək, Excel pəncərəsindəki başlığın adı dəyişdirilir:

Application.Caption="Verilənlər bazası"

Qruplaşmanın bütün obyektlərinin xassəsinə eyni zamanda da dəyişmək olar. Məsələn, Excel proqramında **Worksheets** obyektlər qruplaşmasının **Visible** (Görünmə) xassəsinin qiymətini **False** (Yalan) qurulduqda, aktiv kitabın bütün iş kitabları gizlənilir:

Worksheets.Visible=False

Xassələr arasında obyektin özünü qaytaranlar daha vacib yer tutur. Məsələn, aşağıdakı cədv. 4.2-də həmin xassələrə aid Excel proqramında tez-tez istifadə edilənlərin izahlı siyahısı verilib:

Cədvəl 4.2 Obyektin özünü qaytaran Excel obyektlərinin xassələri

XASSƏLƏR	EDİLƏN ƏMƏL
ActiveWindow	Excel-in aktiv pəncərəsini qaytarır
ActiveWorkbook	Excel pəncərəsinin aktiv iş kitabını qaytarır

ActiveSheet	Aktiv iş kitabının aktiv iş səhifəsini qaytarır
ActiveDialog	Aktiv iş səhifəsinin iş kitabının aktiv dialog pəncərəsini qaytarır
ActiveChart	Aktiv iş səhifəsinin aktiv diaqramını qaytarır
ActiveCell	Aktiv iş səhifəsinin aktiv hücrəsini qaytarır

Aşağıdakı üç nümunədə eyni aktiv hücrə (**Cell**) qaytarılır. Birincisinə daha çox diqqət verin: obyekt qaytaran xassənin qiyməti təlimatda həmin obyektə yazılır. Bu cür üsul VBA-da buraxıla bilər və obyektləri qaytaran çox geniş olan xassələr sinfinə tətbiq edilir.

```
ActiveCell
ActiveWindow.ActiveCell
Application.ActiveWindow.ActiveCell
```

4.4 Obyektin hadisəsi və **WithEvents** elanı

VBA obyektlərinin hadisələri, WithEvents elanını etmə qaydaları

Obyektin müəyyən etdiyi (tanıdığı) fəaliyyətə (hərəkətə) obyektin hadisəsi demək olar. Məsələn: mausun şıqqıldaması yaxud klaviaturanın düyməsinin basılması, proqramın işə salınması (bağlanması), hansısa digər obyektin üzərində əməllərin başlanması (qurtarması) və bu cür əməllərlə proqramlaşdırılan cavab əməllər v.s. Hadisələr istifadəçinin işinin aparılması ilə yaxud avtomatik rejimə sistemin özü ilə bağlı ola bilər.

Əslində VBA ilə Office proqramlaşdırması iki əsas anlayışla bağlıdır:

1. hadisənin yaranması;
2. həmin hadisələrə cavab olan əks əməllər.

Əgər istifadəçinin hansısa əməli yerinə yetirməsi nəticəsində sistemə müəyyən təsir edilirsə (ən sadə halda klaviaturanın hansısa düyməsi basılır), onda, məsələn, əks cavab kimi istifadəçinin özünün yaratdığı proseduranın kodu işə salınır. Əgər bu cür əks cavab yaradılmayıbsa, yeni müvafiq prosedura tərtib edilməyibsə, onda sistem verilmiş hadisəyə heç bir reaksiya vermir və təsir cavabsız qalacaq. Beləliklə, aydın oldu ki, sistemdə cərəyan edən hərəkətlər (müxtəlif fəaliyyətlər) əslində hadisələr kimi baxıla bilər. Həmin hadisələrə yaranan əks cavablar isə tərtib edilmiş proseduralardır. Bu cür hadisələrə reaksiya (əks təsir) verən (əks təsiri generasiya edən) proseduralar xüsusi tip proseduralar kateqoriyasına aiddir – *hadisələri emal edən proseduralar*. Bütövlükdə VBA-da proqramlaşdırmanın əsas məqsədi yaranası hadisələrə bir başa və ya dolayısı yolla əks təsiri (əks cavabi) generasiya edən proseduraların tərtib edilməsi ilə bağlıdır.

Hadisələr obyektin dərinliyində gizlənir və bacarıqlı istifadəçi gerek həmin hadisələri istifadə etməyi bacarsın, yuxarıda sadalanan üsullarla: VBA redaktorunun pəncərəsində lazımı obyektə və onun hadisələrini seçərək. Bəzi hallarda obyektlərin hadisələri VBA redaktorunun dialog pəncərəsində əmələ gəlmir: məsələn, çox vacib olan **Application** obyektə üçün. Belə hallarda həmin obyektin hadisələri ilə birgə **WithEvents** açar sözü ilə elan etmək lazımdır (aşağıdakı nümunə Word sənədinə aiddir):

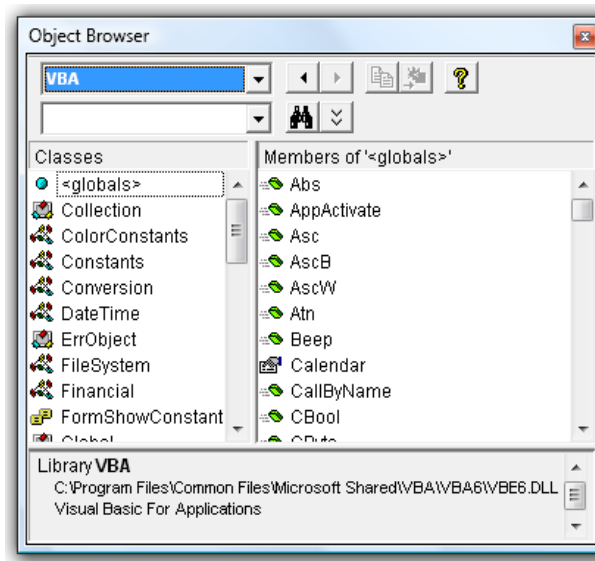
```
Public WithEvents App As Word.Application
```

Bu cür elanlar VBA modulunun elanlar (**declarations**) hissəsində yerləşdirmək lazımdır. Nəticədə Visual Basic kod redaktorunda bütün tələb olan hadisələri ilə birgə yeni **App** obyektini əmələ gəlir. Lakin, biz burada qeyd etməliyik ki, oxucu hadisələrlə işləmək qaydaları ilə daha ətraflı növbəti bölmədə tanış olacaq: müxtəlif formalar və qrafik idarəetmə elementlərinə (düymələr, bayraqcılar, keçiricilər v.s.) həsr edilmiş bölmədə. Hələlik, bu səviyyədə, oxucu VBA dialoq pəncərəsinin kod redaktorunda (yuxarıda sol siyahıda) lazımı qrafik elementin seçib sonra hadisələr siyahısında (obyektlər siyahısının sağındadır) uyğun olan hadisənin seçilməsini bacarsa qənaətbəxş və kifayət saymaq olar. Bu səviyyədə istifadəçi kod redaktorunda avtomatik rejimdə həmin seçdiyi hadisəni generasiya edən prosedurası yaratmış olacaq. Həmin generasiya edilmiş proseduranı istifadəçi tərtib edilən modulun lazım olan yerinə yazsa, onda hansısa hadisə başlandıqda (məsələn, düymə üzərində mausun şıqqıldaması hadisəsi), avtomatik rejimdə ona qarşı hazırlanmış əks təsir yerinə yetiriləcək.

4.5 Obyektlərə baxma və nəzərdən keçirmə

VBA siniflər kitabxanasını nəzərdən keçirmə, Object Browser-in tətbiqi

Obyektlərin yaradılması və onların xassələrindən, metodlarından və hadisələrindən istifadə etmək üçün oxucuda artıq elementar biliklər var. Lakin addım başı VBA Office proqramlaşdırmasında bu cür vəziyyət yaranır: lazımı obyekt tez tapmaq və onun tərkibində olan xassələr, metodlar və hadisələri qısa vaxta təyin etmək lazımdır. Bunun üçün VBA redaktorunda **Object Browser** utiliti vardır (VBA redaktoruna inteqrə edilib). Onu aktivləşdirmək üçün redaktorun pəncərəsində klaviaturanın **<F2>** düyməsini basmaq və açılan qrafik diaoq pəncərəsində tələb olan siniflər kitabxanasını seçmək lazımdır. Həmin qrafik rejimdə siniflər rəng-rəng "kərpicikli" dördbucaqlılar kimi, metodlar isə uçan yaşıl əşya kimi, xassələr əl göstərdiyi yazı kimi görünür və hadisələr şimşək işarəsinə bənzəyir, şəkl. 4.2.



Şəkil 4.2 **Object Browser** dialoq pəncərəsi ilə qruplaşma siniflərinin obyektlərinin metodları, xassələri və hadisələrə baxma imkanı.

Object Browser dialoq pəncərəsində tiplər kitabxanasına baxmaq lazım olduqda (adətən əvvəlcədən o, siyahıda olmur), gərək **Tools**→**References** menyusundan ona istinad verilməlidir və ya **Object Browser**-in özündə **References** kontekst menyusunda istinad verilməlidir. Bununla belə (bu səviyyədə) axtarış aparmaq üçün oxucunun aşağıdakıları bilməyi çox vacibdir:

- **Object Browser**-lə tam dəyərli işləmək üçün istifadəçi obyektönlü proqramlaşdırmada müəyyən qədər təcrübəsi olmalıdır.
- **Object Browser**-dən istifadəçi yalnız metodların, xassələrin, hadisələrin adını və qəbul edilən, qaytarılan qiymətlər haqqında məlumat ala bilər. Hansısa metodun nə etdiyi haqqında olan məlumatı yalnız sənədləşmənin sorğu materiallarında tapmaq olar (**Object Browser**-lə heç bir buna oxşar məlumat əldə etmək mümkün deyil). Bəzən ada görə müəyyən doğru-dürüst fərziyə qurmaq olar.

4.6 Obyekt modelləri

Windows-un qurulmuş obyekt modelləri, Windows Script Host (WSH), Scripting Runtime, ADO, SQLDMO, CDO, WMI, ADSI, Windows Explorer, Internet Explorer utilitlərinin obyekt modelləri

Eyni sahəyə aid məsələlərin həlli üçün nəzərdə tutulmuş obyektlər yığımına **obyekt modelləri** deyirlər. Məsələn, Excel-in obyekt modelində Excel kitabının özünü təmsil edən obyektlər nəzərdə tutulub (iş kitabı, iş səhifələri, hücrələr yığımları, diaqramlar v.s. Kitabın sonunda gələn hissədə Microsoft Office-in proqramları Word, Excel, Access, PowerPoint, Project, Outlook üçün hər tərəfli izahlar verilib. Əgər istifadəçi VBA dilində özünün məxsusi tətbiqi proqramını yaradırsa, onda yalnız Office-in obyekt modelləri ilə kifayətlənmək məcburi deyil. Windows əməliyyat sistemində bir çox başqa obyekt modelləri də qurulmuşdur ki, bunların köməyi ilə istifadəçinin yaratdığı tətbiqi proqramların imkanları xeyli genişləne bilər. Aşağıda Windows-a və Microsoft-un digər məhsullarına qurulmuş obyekt modellərinin siyahısı verilib (bunlarla peşəkar VBA proqramçıları çox aktiv istifadə edirlər):

- **Windows Script Host Object Model (wshom.exe)** - bu kitabxana administratorun işinin avtomatlaşdırılması üçün nəzərdə tutulub. Onun köməyi ilə şəbəkə ilə, printerlərlə, reyestrlərlə, yarlıqlarla, hadisələr jurnalı ilə proqram səviyyəsində iş görməyə imkan yaradılır. Üstəlik həmin kitabxana onlara konsol düyməsinin basılmasını. ötürməsinə v.s təmin edir (Windows-un bütün versiyalarında var);
- **Microsoft Scripting Runtime (scrrun.dll)** - Administrator üçün nəzərdə tutulmuş daha bir kitabxanadır. Əsas üstünlüyü – fayl sistemi ilə işləmək üçün çox əlverişli və sadə siniflər yığıdır (disklərlə, kataloqlarla, fayllarla, mətn fayllarının məzmunu ilə v.s. işləməyə imkan yaradır). **Windows Script Host Object Model** ilə bir komplektdə qurulur;

- **Microsoft ADO** (**msado** adı ilə başlayan fayllar yığımı) – verilənlər bazası ilə işləmək üçün siniflər yığımıdır. Bütün Windows versiyalarında qurulmuş halda olur;
- **Microsoft SQLDMO Object Library** (**sqldma.dll** faylı) - Server Microsoft SQL üzrə tam nəzarəti ələ almaq üçün siniflər yığımıdır (istənilən administrator əməliyyatlarını həyata keçirmək, sorğuları vermək v.s.) yalnız SQL Server kompyutərə yüklənsə əlçatan olur;
- **Microsoft CDO** (**olemsg.dll**, **cdonts.dll**, **cdosys.dll** faylları) – elektron poçtla tam dəyərli şəraitdə işləmək üçün siniflər yığımıdır (məxsusi xəbərlərin yaradılması və elektron poçtla göndərilməsi, poçt yeşiyində yeni xəbərlərə baxılması v.s.). Windows-un bütün versiyalarında var, hətta daha köhnə əməliyyat sistemi olan kompyuterlərdə də var, çünki MS Office ilə birgə yüklənir;
- **Microsoft WMI Scripting v1.1** (**wbemdisp.tlb**) - **WMI (Windows Management Instrumentarium)** proqram interfeysi ilə imkanların genişlənməsini təmin edir. İmkanlar doğrudan da çox böyük görünür: kompyuterdəki ventilyatorun fırlanma sürətinin idarə edilməsindən (ümumiyyətlə, əməliyyat sisteminin bütün əlaqəsi olan qurğulara aiddir) proqram təminatının yüklənməsi, uzaqda olan kompyuterdə proseslərin işə salınması, xidmətlərin idarə edilməsi v.s. qədər. Məsələn, kataloqda hansısa faylın silinməsi (yazılması), hadisələr jurnalında hansısa yazının əmələ gəlməsi (silinməsi), uzaqda olan kompyuterdə müəyyən adlı proqramın işinin bitirilməsi v.s. ilə bağlı əks cavab olaraq, hansısa proqram kodunun işə salınmasını yerinə yetirmək olar. Bütün kompyuterlərdə Windows-un bütün versiyalarında qurulmuş olur.
- **Active Directory Scripting Interface, ADSI** (**adslap.dll**, **wldap32.dll**, **adsnt.dll**, **adsnds.dll**, **adsnw.dll**) – bu obyekt modelləri Active Directory, NT, NetWare kataloqlarındakı obyektləri ilə qarşılıqlı əlaqə yaradır, yəni istifadəçilərin uçot yazılarında, qruplarla, kompyuter obyektləri, printerlərlə v.s. işi təmin edir. Bütün kompyuterlərdə Windows-un bütün versiyalarında qurulmuş olur.;
- **Windows Explorer**-in obyekt modeli – diskdəki fayllarla müxtəlif əməllərin aparılmasında çox əlverişli alətdir;
- **Internet Explorer**-in obyekt modeli – istifadəçi ilə işi təşkil etmək üçün çox güclü və əlverişli alətdir. İstifadəçiyə Web-səhifələri, ardıcıl onları dəyişərək, göstərilməsinə imkan yaradır. Bununla video, audio-kiplərin v.s. nümayişini təşkil etmək olur. Skriptlər və HTML formalarının köməyi ilə istifadəçidən məlumat alınmasında obyekt modeli kimi olduqca əlverişlidir (Microsoft bunun üçün adı qrafik formalardan istifadə edir).

Bu obyektlər haqqında daha ətraflı məlumatı oxucu MSDN sorğu materiallarında tapa bilər. Həmin obyekt modellərinin istifadəçinin proqramında istifadə edilməsi üçün, gərək onun adına yaradılmış layihədə istinad əlavə edilsin. Bu çox sadə üsulla həyata keçirilir:

- Visual Basic redaktorunda **Tools**→**References** meniyusu seçilir;
- lazımı, tələb olan, kitabxana seçilir.

Həmin obyekt modelləri ilə əvvəllər işləməyən istifadəçi üçün onların tətbiqi birinci baxımda çox “qorxunc” görünə bilər. Praktiki məsələlərin həllində lazımı praktiki vərdişlər əldə edilsə, onda bu cür modellərin tətbiqi olduqca asan və eyni zaman lazımlı vasitələr kimi meydana çıxar. Çünki Microsoft onları peşəkar olmayan istifadəçilər üçün yaratmışdır (adi istifadəçilərin daha rahat halda işləməsinə geniş imkanlar yaratmaq üçün). Diqqətli oxucu (VBA istifadəçisi) həmin modellərin praktiki mənimsənilməsində çətinlik çəkməyəcək (kitabın sonunda hər hissəyə aid, həmçinin bu hissəyə də aid, oxucunun sərbəst Office proqramlaşdırma mühitində VBA ilə işləməsi üçün xüsusi tapşırıqlar verilib – bu tapşırıqlar hökmən, müstəqil olaraq, yerinə yetirilməlidir).

5. FORMALAR, IDARƏETMƏ ELEMENTLƏRİ VƏ HADISƏLƏR

VBA proqramlarında olan formalar, formaların tətbiqi, MS VBA-da idarəetmə elementləri

VBA-da **UserForm** (tərkibində idarəetmə elementləri olan xüsusi dialoq pəncərəsi rejimi) formalar vasitəsindən istifadə etməklə Office proqramlaşdırması sahəsində işləmək üçün ən əlverişli mühit yarana bilər [1, 2, 3, 4, 6, ..., 18]. VBA istifadəçiləri həmin formaların proqramların tərtib edilməsində (həmçinin istismarında da) olan faydasını aşağıdakı əməllərin yerinə yetirilməsində hiss edirlər :

- verilənlərin daxil edilməsində;
- istənilən tipdə alınan hesablaşma nəticələrinin çıxarılmasında;
- proqram işlədikdə, müəyyən hadisələrə uyğun olaraq, istifadəçi üçün idarəetmə funksiyalarının həyata keçirilməsində.

Bu vasitə heç də MS VB-dən fərqlənmir, xüsusilə MS Office paketinin 2007-ci və 2010-cu versiyasında. **Show** metodu formanı ekranda göstərir, **Hide** metodu isə onu bağlayır. Müxtəlif tərtibata uyğun olaraq, VBA-da çox geniş OLE (hal-hazırda, 1996-cı ildən bəri, Microsoft OLE-nin yenilənmiş versiyasını **ActiveX** adlandırmışdır) obyektləri və metodları mövcuddur. Buna görə istifadəçi istənilən mürəkkəblikdə tərtib etdiyi proqramın interfeysini Windows mühitinə uyğun yarada bilər. VBA-da istifadəçi ilə təmas saxlamağın ən sadə imkanları ilə (**MsgBox** () və **InputBox** ()) qurulmuş funksiyaların tətbiqi) biz artıq tanış olmuşuq. Bununla belə **MsgBox** () və **InputBox** () standart funksiyaların imkanları həmişə kifayət qədər geniş səviyyədə olmur. Bu fəsilə oxucu VBA formaları ilə bağlı bunları öyrənəcək:

- idarəetmə elementlərindən istifadə edərək, öz şəxsi proqramlarını yaratmaq;
- tətbiqi proqrama proqram məhsulu kimi baxaraq, onun qrafik interfeysini tərtib etmək.

Qrafik interfeys tertib edildikdə, VBA standart formaları çox geniş imkanlar yaradan və olduqca sadə strukturlu alətlər sistemi kimi meydana çıxır. Bir çox idarəetmə elementlərini bir başa Office proqramının özünə yerləşdirmək olar. Klassik üsulda isə formalar VBA-nın prosedura səviyyəsində tətbiq edilir. Bəs VBA proqramında VBA formalarının tətbiqi hansı qaydalar əsasında tətbiq edilir? Əlbəttə, birinci növbədə istifadəçini maraqlandıran məsələ budur: formalar necə işə salınır? Adətən istifadəçi tərəfindən sənədin açıldığı an, formalar dərhal işə salınır. Həmin formalarda istifadəçi hansısa müəyyən əməlləri yerinə yetirməlidir: məsələn, açılan siyahıdan tələb olan qiyməti seçməlidir, bayraqcıqlar və keçiricilər üçün lazımı vəziyyətləri seçməlidir v.s. Son addımda istifadəçi həmin formada “qərar verici” düyməni basaraq hansısa əməlin (hadisənin) yerinə yetirilməsinə əmr verir – onun tərəfindən forma vasitəsi ilə daxil edilmiş verilənlər verilənlər bazasına yazılır və (və ya) verilənlər üzərində əməllər aparılır, verilənlər elektron poçtla müəyyən ünvana göndərilir, hansısa fayla yazılır (sonra çap edilsin deyə) v.s.

5.1 Formaların yaradılması və onların ən vacib olan xassələri və metodları

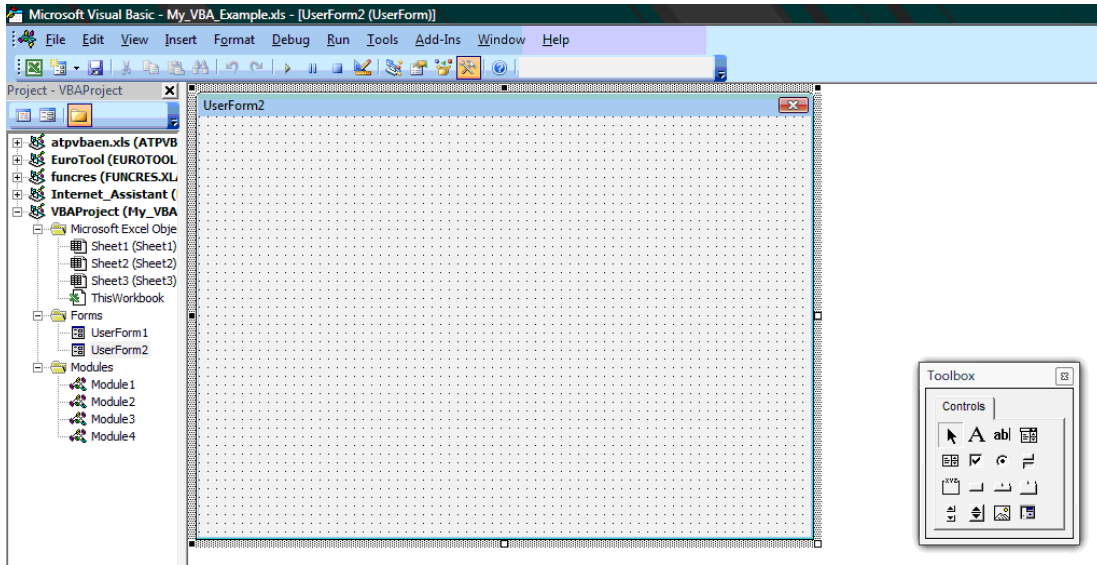
VBA formalarının yaradılması, formaların xassələri, Show() və Hide() metodları, Unload əmri, Initialize() hadisəsi

Formanın yaradılması əslində çox sadədir:

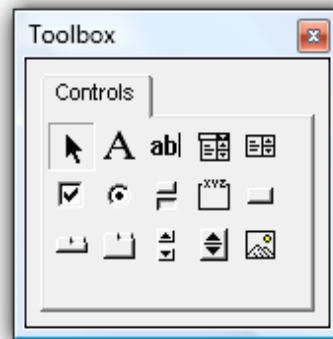
- Visual Basic redaktorunda mausun sağ düyməsi ilə Project Explorer pəncərəsində layihə üzərində şıqqılatmaq, nəticədə kontekst menyusuna əmələ gələcək;
- əmələ gəlmiş kontekst menyusunda **Insert**→**User Form** seçilməli, nəticədə formalar dizayneri pəncərəsi (**Form designer**) açılacaq;
- formalar dizayneri pəncərəsində (**Form designer**) formanın boş pəncərəsi (standart vəziyyətdə onun adı **UserForm1** olacaq) yanında isə, tərkibində idarəetmə elementləri olan, **Toolbox** alətlər paneli olacaq (bax şəx. 5.1 və Şəxş 5.2).

Əgər həmin an xassələr pəncərəsi (klaviaturanın **<F4>** düyməsi ilə canlanır) aktivdirsə, onda bu pəncərədə cari formanın xassələri görsənəcək. Həmin formanın ekvivalent VBA koduna keçmək lazım olduqda (standart vəziyyətdə Click hadisəsi açılır), klaviaturanın **<F7>** düyməsini basmaq lazımdır. Geriyə (forma dizaynerinə) keçid klaviaturanın **<Shift>+<F7>** düymələr kombinasiyası ilə həyata keçirilir.

Başqa çox vacib olan imkan budur ki, formalar və idarəetmə elementləri üçün xassələri xassələr pəncərəsinin qrafik interfeysi ilə seçmək olur – nəticədə əl ilə yazılacaq proqram kodunun həcmi kəskin olaraq azalır.



Şəkil 5.1 formalarla işləmək üçün (yeni forma tərtib etmək üçün) forma dizayneri pəncərəsi.



Şəkil 5.2 formalarla işləmək üçün **ToolBox** alətlər pəncərəsi.

Formaların bir neçə daha vacib olan xassələri aşağıda verilib (**ShowModal** xassəsindən başqa bütün bunlar digər idarəetmə elementlərinə də tətbiq edilə bilər):

- **Name** – bu xassə formanın adını təyin edir. Kənar istifadəçi adətən tərtib edilmiş formanın adını heç bir zaman bilməyəcək (standart vəziyyətdə onun adı **UserForm1** kimi avtomatik təyin edilir). Formanın adını yalnız VBA proqramçısı proqram kodunda istifadə edir. Forma yaradıldıqdan sonra (proqramda ona istinad edilməsi asanlaşsın deyə) onun adını daha mənalı adla əvəz etmək məsləhət görülür (bu məsləhət bütün idarəetmə elementlərinə aiddir);
- **Caption** – bu xassə formanın başlığını təyin edir (standart vəziyyətdə formanın adı ilə üst-üstə düşür). Məsləhət görülür ki, formanın təyinatı haqqında olan qısa məlumat formada mətn sətiri kimi yaradılsın (məsələn, tutaq ki, “Aylıq raportun forması”);
- **Enabled** xassəsi - əgər **False** vəziyyətində qurulubsa, istifadəçi forma ilə heç bir vaxt işləyə bilməyəcək. Formanın müvəqqəti keçirilməsi (işlək halda olmaması) üçün istifadə edilir, məsələn, istifadəçi onun işləməsi üçün bütün şərtləri təmin edəne qədər.

- **ShowModal** xassəsi - əgər **True** qiyməti ilə qurulursa (standart vəziyyətdə belədir), onda istifadəçi bu formanı bağlamayana qədər, başqa formalara keçə bilmir və sənədə də yenidən qayıtmağı bacarmır.

Xassələrin əksəriyyəti pəncərələrin xarici görünüşü, harada yerləşməsi və ölçülərinə aiddir. Formaların yaradılmasında ən vacib üsullar bunlardır:

- **Show()** – bu metod, Visual Basic-in redaktorunun pəncərəsində formanı redakte etdikdə, formanı işə salmağa kömək edir (həmçinin klaviaturanın **<F5>** düyməsi basıldıqda forma işə salına bilər). Aşağıdakı nümunədə bu metodun köməyi ilə **UserForm1** formasının işə salınması həyata keçirir:

```
UserForm1.Show
```

Əgər forma artıq yaddaşa yüklənmişdirsə, onda o, artıq görünən olacaq (əgər belə olmasa, onda forma, avtomatik olaraq, yüklənəcək – **Load** hadisəsi baş verəcək).

Həmin metodun özünü bu cür çağırmaq olar:

- düymə ilə bağlı olan adi makrosla və ya klaviaturanın düymələri kombinasiyası ilə;
- avtomatik olaraq işə salınan makrosdan (məsələn, Word üçün **AutoExec** adı ilə olan makrosdan);
- sənədin özündə və ya başqa formada yerləşmiş (formadan formaya keçmək üçün) idarəetmə elementinin kodundan (məsələn, **CommandButton** ilə);
- əmri hadisələri emal edən **Open** metoduna (Word və ya Excel sənədləri üçün) – bu halda sənəd açıldıqda forma da avtomatik rejimdə açılır.

- İstifadəçi lazım olan verilənləri formada daxil edib “qərar edici” düyməyə basdıqdan sonra forma, təbii olaraq, aradan götürülməlidir (görünməz olaraq, deaktivləşməlidir). Bunun üçün iki üsul var:

- formanı gizlətmək (**Hide()** metodunu istifadə etmək), məsələn:

```
UserForm1.Hide
```

bu halda forma ekrandan götürüləcək - yalnız yaddaşa saxlanacaq. Sonra **Show()** metodu ilə yenə də onu, “gizlənmə” anındakı vəziyyətə uyğun olaraq, çağırmaq olar. Gizləndiyi zaman müddətində formanı redakte etmək olar – onun üzərində olan idarəetmə elementlərini dəyişmək, proqram səviyyəsində kodunu dəyişmək. Sənəd bağlandıqda, forma operativ yaddaşdan tamam silinəcək;

- əgər dəqiq məlumdur ki, proqramın sonrakı fəaliyyətində forma lazım olmayacaq, onda onu **Unload** əmri ilə operativ yaddaşdan silmək olar:

Qalan metodlar ya dəyişmə buferi ilə həyata keçirilən verilənlərin dəyişməsi ilə bağlıdır (**Copy()**, **Cut()**, **Paste()**), ya da formanın xidməti imkanları ilə bağlıdır (**PrintForm()**, **Repaint()**, **Scroll()**).

Yuxarıda qeyd etdiyimiz kimi, VBA-nın ən vacib konsepsiyası – hadisələrdir. VBA-da hadisə (**event**) dedikdə bunu anlamaq lazımdır – proqramda nəyinsə baş verməsi (hadisənin) və proqramın özünün həmin dəyişmənin müəyyən edə bildiyi vəziyyətə hadisə demək olar. Məsələn, dəfələrlə, yuxarıda qeyd edilmiş misallarda, mausun sağ və ya sol düyməsinin şıqqıldadılması, klaviaturanın düymələrinin basılması, proqramların, formaların bağlanması (işə salınması), formanın ekranda hərəkət etməsi, qəflətən qrafik obyektin əmələ gəlməsi (silinməsi), elektron poçtla hansısa məlumatın poçt qutusuna gəlməsi v.s – bunların hamısı hadisədir. VBA özəyində elə qurulmuşdur ki, onun mühitində yaradılan proqramlar hadisələri idarə edə bilirlər (və ya hadisələr tərəfindən idarə edilə bilirlər).

Aşağıda formaların ən vacib hadisələri verilib:

- **Initialize** – formanın açılmasına yaxın və ya formanın hazırlanması zaman baş verir (formanın istifadəçi qarşısında əmələ gələne qədər). Adətən bu hadisəni emal edən prosedura verilənlər bazasının açılması/birləşməsi, formadakı idarəetmə elementlərinin sazlanması v.s. ilə bağlı olanda proqram koduna quraşdırılır;
- **Click** (bu hadisə standart vəziyyətdə artıq seçilmiş olur) və **Db1Click** – uyğun olaraq, mausun sol düyməsinin bir yaxud ikiqat şıqqıldamasına olan reaksiyanı bildirir. Forma üçün bu hadisə növləri bir o qədər tez-tez istifadə edilmir. Adətən şıqqıldama hadisəsini emal edən funksiyalar formanın düymələri (məsələn, idarəetmə elementi **CommandButton** üçün);
- **Error** – bu hadisə formada səhv baş verdikdə istifadə edilir. Peşəkar proqramçılar bu hadisəni ondan ötrü istifadə edirlər ki, yarana bilən forma səhvlərini interaktiv şəkildə, qısa zaman müddətində, aradan qaldırmaq mümkün olsun. Bu haqda VBA kodlarında yarana bilən səhvlər haqqında olan hissədən oxucu daha ətraflı məlumat alacaq;
- **Terminate** – bu hadisə formanın normal işini tamamladığında və operativ yaddaşdan silindikdə tətbiq edirlər (məsələn, **Unload** əmrində). Adətən verilənlər bazası ilə açılmış birləşmələrin qırılmasında, resursların artırılmasında, protokollaşmada v.s. istifadə edilir. əgər formanın işi avariya ilə bitirilibsə, onda bu hadisə baş vermir;

Qalan hadisələr ya pəncərənin ölçülərinin dəyişdirilməsi, ya konsol düyməsinin basılması və ya aktivləşmə/deaktivləşmə ilə bağlıdır (fokusun alınması/itirilməsi).

Formaların yaradılması və redaktə edilməsi ilə bağlı bəzi vacib qeydlər:

- Microsoft Access-də yaradılan formalar standart deyil (Office-in digər proqramlarındakı formalardan fərqlənir). Bu formaların xassələr və metodlar yığımı Office-in başqa proqramlarından fərqlidir - funksionallıq baxımından isə onlar eynidir;
- bəzən formanı daha yaxşı anlamaq üçün onu çap etmək daha əlverişli sayıla bilər. Bunun üçün xüsusi dialoq pəncərəsi nəzərdə tutulub, klaviaturanın **<Ctrl>+<P>** düymələri basıldıqda aktivləşir (forma dizayneri pəncərəsinin seçilmiş forması ilə);
- əgər bütün lazım olan idarəetmə düymələri formaya yerləşməmiş (hətta böyük ölçülü formada belə), onda istifadəçinin əlində iki variant var:
 - iki formadan istifadə etmək (onlar arasında keçidi idarəetmə elementlərinə bağlı olan **Show()** və **Hide()** metodu ilə həyata keçirərək);
 - formada bir neçə sayda quraşdırmadan istifadə etmək. Bunun üçün istifadəçinin sərəncamında **MultiPage** idarəetmə elementi var.

5.2 İdarəetmə elementləri

5.2.1 İdarə etmə elementləri haqqında əsas məlumat

VBA-nın idarəetmə elementləri, formaya idarəetmə elementlərinin əlavə edilməsi

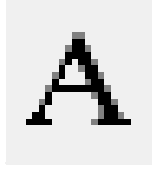
İdarə etmə elementləri – istifadəçi ilə təması təşkil etmək üçün nəzərdə tutulmuş və formada yerləşdirilə bilən xüsusilaşmış obyektlər növüdür. VBA-da standart tipli (**CommandButton**, **CheckBox**, **OptionButton**), həmçinin qeyri standart tipli (məsələn, **Internet Explorer**, **Calendar** v.s.) idarəetmə elementlərindən istifadə edilməsi mümkündür. İdarə etmə elementləri, istifadəçi tərəfindən generasiya edilən (düymənin basılması, qiymətin daxil edilməsi v.s.), hadisələrə reaksiya verir.

Formaya idarəetmə elementlərini adətən formalar dizayneri pəncərəsindən **Toolbox** ilə əlavə edirlər. Bunun üçün **Toolbox**-da idarəetmə elementini seçərək, onu çəkib formanın üstünə, istifadəçi tərəfindən təyin edilmiş yerə, atıb yerləşdirmək lazımdır. Sonra isə həmin idarəetmə elementini redaktə etmək olar (qrafik formasının ölçülərini, ekrandakı vəziyyətini, rəngini, idarəetmə funksiyalarının dəyişdirilməsini v.s.). İdarə etmə elementlərini proqram üsulu ilə də əlavə etmək olar: **Controls** kolleksiyasının **Add()** metodu ilə. Proqram üsulunda isə proqramçı VBA kodunda xeyli əlavə iş görəsi olacaq: VBA kodunda idarəetmə kodunun çox sayda xassələrini əlavə etməlidir, bu isə proqramlaşdırmada məhsuldar iş sayıla bilməz.

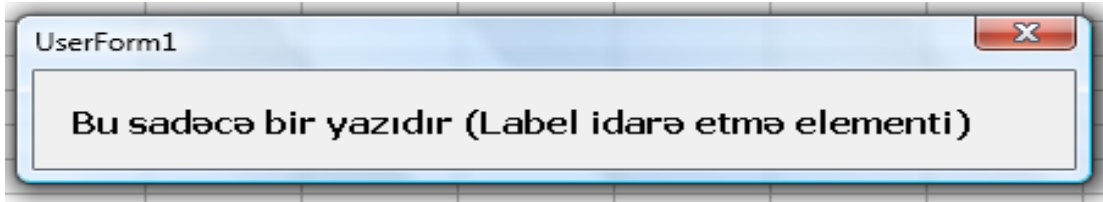
5.2.2 **Label** (yazı) idarəetmə elementi

Label (yazı) idarəetmə elementi və Caption xassəsi

Label - ən sadə idarəetmə elementidir. İstifadəçi bu elementlə formanın təyin edilən yerində hansısa yazının verilməsini icra edə bilər, şəkl. 5.2. və şəkl. 5.4.



Şəkil 5.3 **Toolbox**-da **Label** idarəetmə elementinin piktoqramı.



Şəkil 5.4 Formada **Label** idarəetmə elementinin nümunəsi.

İstifadəçi formadakı idarəetmə elementinin yazısını dəyişdirə bilmir. Adətən **Label** idarəetmə elementi vəziyyət sətiri kimi fəaliyyət göstərir (hal-hazırda nə baş verdi/verəcək/nə etmək lazımdır fəaliyyətini istifadəçiyə bildirmək üçün). Bir çox hallarda bu elementi digər idarəetmə elementləri haqqında izahlar yazmaq üçün istifadə edirlər. **Label** elementinin əsas xassəsi **Caption** xassəsidir (yəni görüntüyə çıxarılan məndir). Digər xassələr bu elementin formatlaşması ilə bağlıdır. Hər halda istifadəçilər bilməlidir ki, bu elementin də bəzi hərəkətlərlə bağlı hadisələr yığılı vardır: məsələn, **Click**, **Error** v.s. (sadəcə bir çox istifadəçi bilmir ki, yazının da üstündən mausla şıqqıldatmaq olar).

5.2.3 **TextBox (mətn qutusu) idarəetmə elementi**

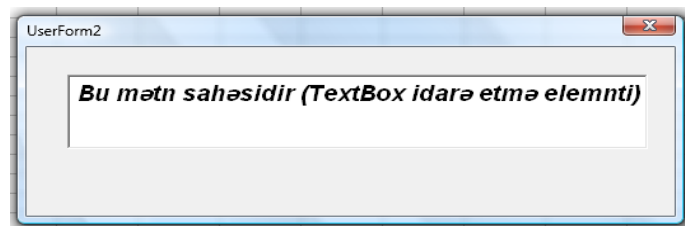
TextBox idarəetmə elementi, **Value**, **Text xassələri**

TextBox - mətn qutusudur və ən çox istifadə edilən idarəetmə elementlərindən biridir, şəkil 5.5-də piktoqramı göstərilib.



Şəkil 5.5 **Toolbox**-dan **TextBox** (mətn qutusu) idarəetmə elementinin piktoqramı.

Aşağıdakı şəkil 5.6-də mətn qutusuna aid nümunə göstərilib, burada mətnin standart vəziyyətdəki **Font** xassəsi (Tahoma, Size: 8, Normal) dəyişdirilib (Arial, Size: 12, Bold-İtalik).



Şəkil 5.6 Formada mətn qutusu (**TextBox** idarəetmə elementi).

Mətn qutusunda aşağıdakı hallarda istifadə edirlər:

- hansısa mətn verilənlərinin (istifadəçi tərəfindən daxil edilən) qəbul edilməsində və sonra, məsələn, elektron poçtla göndərmək, verilənlər bazasına yazmaq (silmək) v.s. əməllərin yerinə yetirilməsində;
- istifadəçiyə mətn verilənlərinin ötürülməsində (məsələn, sonradan redaktə etmək üçün, Excel səhifəsindən);
- istifadəçiyə mətn verilənlərinin ötürülməsində, sonradan kopya edilib çap etmək üçün (əlbəttə, sonradan redaktə etmək olmaması şərti ilə);
- həmin idarəetmə elementinin bəzi xassələrindən istifadə etmək üçün;

Aşağıda həmin idarəetmə elementinin əsas xassələri verilib:

- **Value** və ya **Text**, hər iki xassə mətn qutusu üçün eynidir. Bu xassənin köməyi ilə mətn qutusuna istənilən mətn yazılır;
- **AutoSize** – avtomatik olaraq mətn qutusunun ölçüsünün mətnin uzunluğuna görə tənzimlənməsini təmin edir. Bu xassədən istifadə etmək məsləhət görünür, çünki formanın bütün dizaynı sonradan pozula bilər
- **ControlSource** – mətnin gəldiyi mənbəyə işarə edir (məsələn, Excel hücrəsinə, Word sənədinin elementinə). Əgər mənbədə mətn dəyişsə, onda avtomatik olaraq mətn qutusunda da həmin dəyişiklik olacaq.
- **ControlTipText** – üzərək-çıxan köməkçi məsləhətin mətni (istifadəçi mausun göstəricisini elementə yaxınlaşdırdıqda əmələ gəlir). Bütün idarə elementlərinin doldurulması üçün məsləhət görülür (formanın özü üçün nəzərdə tutulmayıb).
- **Enabled** – əgər **False** qurulsay, onda sahədəki mətn boz rəngdə olacaq, və sahənin içərisindəki ilə heç bir şey dəyişdirilə bilməz. Adətən bütün idarəetmə elementlərində istifadə edilir.
- **Locked** – qutunun sahəsi adi qaydadakı kimi görünsə də (seçib kopya etmək olur) sahədəki mətni redaktə etmək olmayacaq. Adətən dəyişdirilə bilməyən verilənləri istifadəçiyə göstərmək üçün istifadə edilir.
- **MaxLength** – qutuya yerləşdirilən mətnin maksimal uzunluğunu bildirir.
- **MultiLine** – qutuda bir neçə sətirin (yaxud birinin) olmasını təyin edir.
- **PasswordChar** – istifadəçinin daxil etdiyi verilənlərin hansı işarəsi ilə qiymətlər “gizlənilir” (parollar qoyulduqda istifadə edilir).
- **ScrollBars** – şaquli və ya üfüqi keçid zolaqlarının olub-olmamasını təyin edir (mətn böyük olduqda bu xassəsiz keçinmək olmur).

- **WordWrap** - əgər **MultiLine** xassəsi **True** vəziyyətindədirsə, onda bu xassə hökmən qurulmalıdır (avtomatik olaraq mətnin yazısı sərhədə çatdıqda yeni sətərə keçəcək).

Digər xassələr mətn qutusunun formatlaşmasına və redaktənin sazlanmasına aiddir. Mətn qutusunun əsas hadisəsi – **Change** hadisəsidir (qutudakı mətnin dəyişdirilməsi). Adətən bu hadisəyə istifadəçinin verdiyi qiymətlərin yoxlanması da bitişdirilir (calanır).

5.2.4 ComboBox (kombinəli siyahı qutusu) idarəetmə elementi

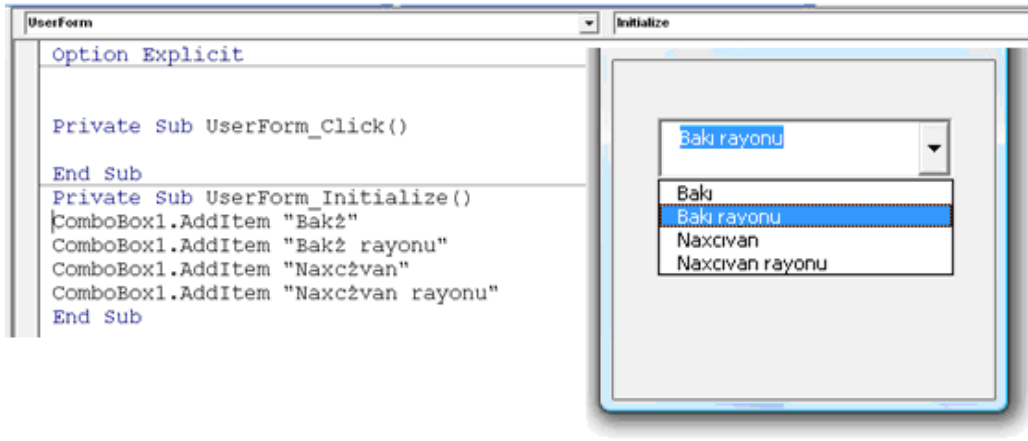
ComboBox idarəetmə elementi, AddItem() metodu, Value xassəsi

Kombinəli siyahı qutusu idarəetmə elementi kimi tez-tez istifadə edilir. Bu siyahı istifadəçi üçün hazır olan qiymətlərin (verilən siyahıda artıq hazır olanda) daxil edilməsinə kömək edir. Şək. 5.7-da bu idarəetmə elementinin **Toolbox**-dakı piktoqramı göstərilib.



Şəkil 5.7 **Toolbox**-dan **ComboBox** (kombinəli siyahı qutusu) idarəetmə elementinin seçilən piktoqramı.

ComboBox (kombinəli siyahı qutusu) idarəetmə elementinin formadakı nümunəsi və bu formanı generasiya edən VBA kodu aşağıdakı şək. 5.7-də göstərilib.



Şəkil 5.8 **ComboBox** (kombinəli siyahı qutusu) idarəetmə elementinin formadakı nümunəsi.

Adətən **ComboBox** idarəetmə elementi iki halda istifadə edilir:

- əgər istifadəçi 4-dən bir neçə onluq yuxarı olan siyahıdan bir və ya bir neçə qiymət seçməlidirsə;
- mənbədən gələn (verilənlər bazasından, Excel səhifəsindən v.s.) verilənlər əsasında pozisiyaların siyahısını dinamik tərzdə formalaşdırmaq lazım olduqda.

Təəssüf ki, xassələr pəncərəsindən istifadə edərək, siyahını pozisiyalarla doldurmaq mümkün deyil – bunun üçün `AddItem()` metodundan istifadə edilməsi lazım olacaq. Adətən VBA kodunda bu metodu hadisələri emal edən hissəyə yerləşdirirlər. Şək. 5.8-də göstərilən nümunənin VBA kodu da həmin formanın yanında VBA redaktorunun pəncərəsində əks edilib.

ComboBox idarəetmə elementinin vacib xassələri:

- **ColumnCount**, **ColumnWidth**, **BoundColumn**, **ColumnHeads**, **RowSource** – bir neçə sayda olan sütunlardan ibarət siyahılarla işlədikdə istifadə edilir. İstifadəçilər bu xassələrdən adətən istifadə etmirlər (daha sadə və əlverişli yol – bir neçə kombinəli siyahı qutusunu yaratmaq);
- **MatchEntry** - istifadəçi tərəfindən qiymətin birinci işarələrinin daxil edilməsində siyahıdan uyğun olan pozisiyaların seçilməsini təyin edir. Bu xassə olduqca vacibdir (məsləhət görülür ki, istifadəçi onu standart vəziyyətdə olan kimi qursun).
- **MatchRequired** – istifadəçinin siyahıda olmayan qiymətin daxil edilməsinə icazə verilməsini (və ya verilməməsi) təyin edir. Standart vəziyyətdə **False**, yəni icazə verilir;
- **Value** (yaxud **Text**) – proqram üsulu ilə seçilmiş qiymətin qurulmasına imkan yaradır.

Digər xassələr:

AutoSize, **Enabled**, **Locked**, **ControlText**, **ControlTipText**, **MaxLength**) - **TextBox** idarəetmə elementində istifadə edilən kimi burada da, analogi olaraq, istifadə edilir.

Analogi olaraq, **TextBox**-da olan kimi, **ComboBox** idarəetmə elementinin vacib hadisəsi - **Change** hadisəsidir. Adətən bu hadisənin icrasında istifadəçinin daxil etdiyi qiymətlər yoxlanılır (qiymətlər mətn sahəsinə yaxud **ListBox** ötürülür).

5.2.5 **ListBox** (siyahı qutusu) idarəetmə elementi

VBA formalarında ListBox (siyahı qutusu) idarəetmə elementi və onun tətbiqi

Bu idarəetmə elementi daha çox yuxarıdakı kombinəli siyahı qutusuna oxşayır, ancaq müqayisədə daha az tətbiq edilir - əsasən iki səbəbə görə:

- burada açılıb-tökülən formada siyahını açmaq olmur. Bütün qiymətlər dərhal mətn sahəsində görünür, buna görə daha böyük pozisiya vermək olmur;
- istifadəçi öz qiymətlərini daxil edə bilmir, yalnız hazırlardan seçim edə bilər.

Şək. 5.9-da **ListBox** (siyahı qutusu) idarəetmə elementinin **Toolbox**-dakı piktoqramı göstərilib. **ListBox** (siyahı qutusu) idarəetmə elementinin formadakı nümunəsi və bu formanı generasiya edən VBA kodu aşağıdakı şək. 5.10-da göstərilib. Bununla belə, bu idarəetmə elementinin bəzi üstünlükləri də var: burada istifadəçi bir dəfəyə, **ComboBox**-dan fərqli olaraq, birini yox, bir neçə qiymət seçə bilər.

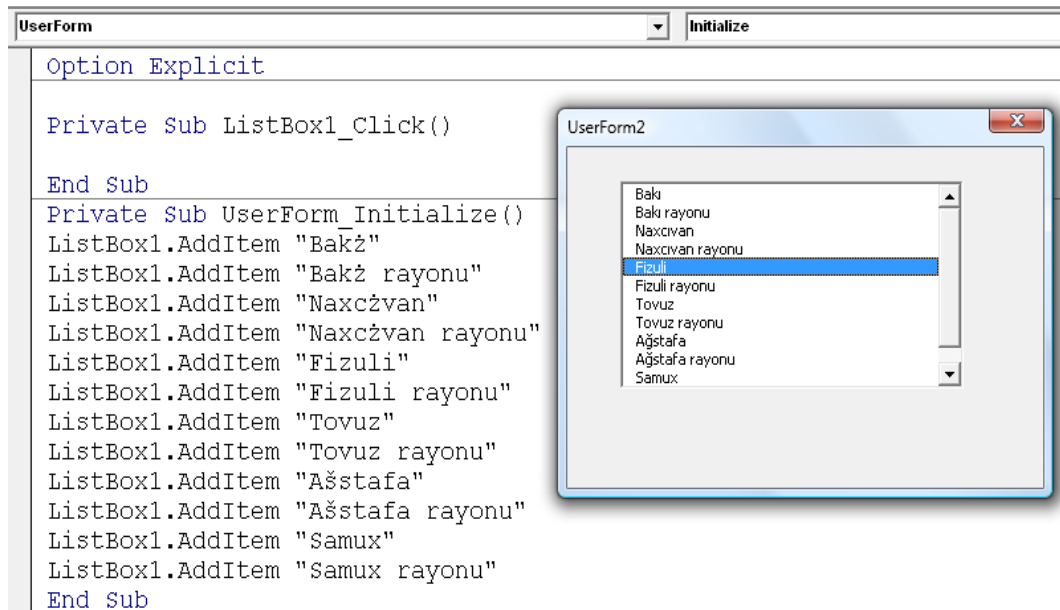
Adətən **ListBox** aşağıdakı hallarda istifadə edilir:

- **ComboBox**-dan istifadəçinin daxil etdiyi qiymətlər üçün redaktə edən aralıq vasitəsi lazım olduqda;
- təkrarən redaktə edən aralıq vasitəsi lazım olduqda, yalnız verilənlər bazasından alındıqda (bunun üçün formada **ListBox** idarəetmə elementinin yanında **Silmək** və **Dəyişmək** idarəetmə düymələrini yerləşdirmək olar).

ListBox-un əsas xassələri, metodları və hadisələri **ComboBox**-dan fərqlənmir. Lakin bir əsas fərq – **MultiSelect** xassəsindədir (bu vasitə ilə istifadəçi bir dəfəyə bir neçə qiyməti seçə bilər). Standart halda bu xassə keçirilmiş vəziyyətdə olur (yəni **Properties Window** (xassələr pəncərəsində) **False** vəziyyətində olur).



Şəkil 5.9 **Toobox**-dan **ListBox** (siyahı qutusu) idarəetmə elementinin seçilən piktoqramı.



Şəkil 5.10 **ListBox** (siyahı qutusu) idarəetmə elementinin formadakı nümunəsi və bu formanı generasiya edən VBA kodu.

5.2.6 **CheckBox** (yoxlama qutucuğu) və **ToggleButton** (fiksə edən düymə) idarəetmə elementləri

CheckBox (yoxlama qutucuğu) və **ToggleButton** (fiksə edən düymə) idarəetmə elementləri, **Caption** və **Value** xassələri, **Change** hadisəsi

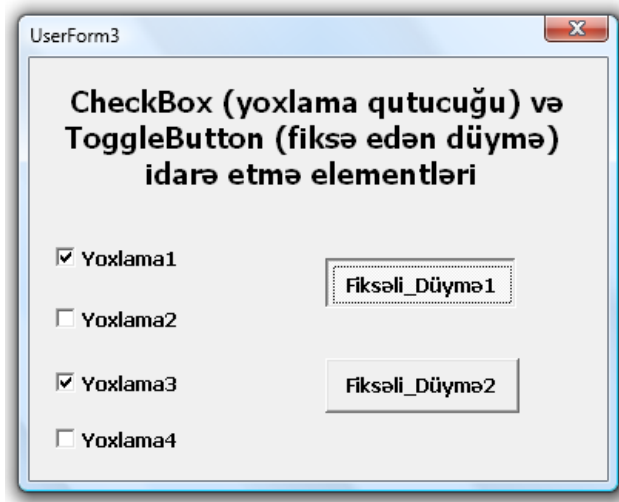
Yoxlama qutucuqları (bəzən istifadəçilər onlara "quş" deyirlər) və *fiksə edən düymələr* formalarda ən çox istifadə edilən idarəetmə elementləridir. Şək. 5.11-də **CheckBox** (yoxlama

qutucuğu, şəkildə soldadır) və **ToggleButton** (fiksə edən düymə, şəkildə sağdadır) idarəetmə elementlərinin **Toolbox**-dakı piktoqramları göstərilib.



Şəkil 5.11 **Toolbox**-dan **CheckBox** (yoxlama qutucuğu, piktoqramı soldakı şəkildir) və **ToggleButton** (fiksə edən düymə, piktoqramı sağdakı şəkildir) idarəetmə elementlərinin piktoqramları.

CheckBox (yoxlama qutucuğu, şəkildə solda 4 sayda **CheckBox** elementi alt-alta qurulub) və **ToggleButton** (fiksə edən düymə, şəkildə sağda iki sayda **ToggleButton** elementi alt-alta qurulub) idarəetmə elementlərinin formadakı nümunəsi aşağıdakı şək. 5.12-də göstərilib.



Şəkil 5.12 **CheckBox** (yoxlama qutucuğu) və **ToggleButton** (fiksə edən düymə) idarəetmə elementlərinin formadakı nümunəsi.

CheckBox-un üç əsas xassəsi var:

- **Caption** – bayraqcıqın sağ tərəfində yerləşən izah edici yazını təyin edir;
- **TriState** - əgər **False** vəziyyətindədirsə (standart vəziyyətdə belədir), onda yoxlama qutucuğu yalnız iki halda ola bilər: qurulub və yox. Digər vəziyyətdə (**True** olduqda), onda üçüncü hal olacaq: **Null**, yəni “boz rəngli quş” qurulur. Sonuncu hal proqram yükləndikdə onun komponentləri seçilərkən istifadə edilir;
- **Value** – yoxlama qutucuğunun öz vəziyyətini təyin edir. **True** qiymətində yoxlama qutucuğu qurulub, **False** olduqda – çıxarılıb və **Null** “boz qutucuq” (**TriState** xassəsi **True** vəziyyətində olduqda).

Bu idarəetmə elementinin də əsas hadisəsi - **Change** hadisəsidir.

ToggleButton düyməyə bənzəyir, basıldıqda “basılı vəziyyət” alınır, ikinci dəfə basıldıqda isə keçirilir. **CheckBox**-da olan kimi burada da iki (və ya üç, **TriState** xassəsindən asılı olaraq) hal ola bilər. Xassələr və metodlar da eynidir. Yeganə fərq istifadəçinin həmin qrafik obyektin görüntü kimi necə qəbul etməsidir: adətən istifadəçilər **ToggleButton** idarəetmə elementini uzun müddətli prosesin başlanmasına işarə edən keçid kimi baxırlar.

5.2.7 **ToggleButton** (keçirici) və **Frame** (çərçivə) idarəetmə elementləri

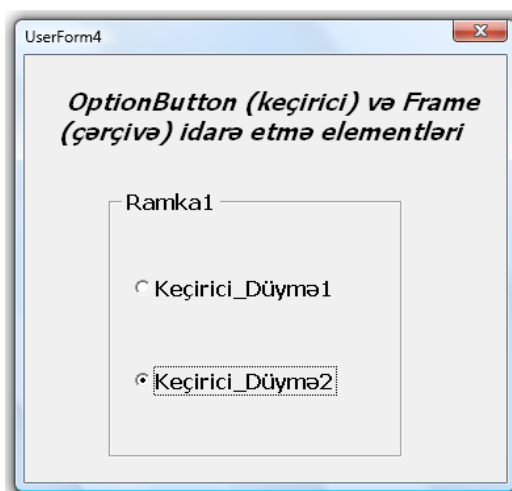
ToggleButton (keçirici) və **Frame** (çərçivə) idarəetmə, **Caption** və **Value** xassəsi, **Change** hadisəsi

Əgər **CheckBox** idarəetmə elementi bir-birini istisna edən variantlardan ötrü nəzərdə tutulmuşdursa, **ToggleButton** seçmək etmək üçün nəzərdə tutulub. Şək. 5.13-də **ToggleButton** (keçirici) və **Frame** (çərçivə) idarəetmə elementlərinin **ToolBox**-dakı piktoqramları göstərilib.



Şəkil 5.13 **Toolbox**-dan **ToggleButton** (keçirici, piktoqramı soldakı şəkildir) və **Frame** (çərçivə, piktoqramı sağdakı şəkildir) idarəetmə elementlərinin seçilən piktoqramları.

ToggleButton (keçirici, şəkildə 2 sayda **ToggleButton** elementi alt-alta qurulub) və **Frame** (çərçivə, şəkildə bir sayda Çərçivə adı ilə **Frame** elementi hər iki **ToggleButton** elementlərini əhatə edir) idarəetmə elementlərinin formadakı nümunəsi aşağıdakı şək. 5.14-də göstərilib.



Şəkil 5.14 **ToggleButton** (keçirici) və **Frame** (çərçivə) idarəetmə elementlərinin formadakı nümunəsi.

Çox populyar nümunə kimi **OptionButton** idarəetmə elementini bu cür illüstrə etmək olar – radioqəbuledicidə radiostansiyanın seçimi: eyni zamanda hər iki radiostansiyaya qulaq asmaq olmaz (buna görə, bəzən bir çox alqoritmik dillərdə bu idarəetmə elementi **RadioButton** kimi tanınır).

Bu elementin iki əsas xassəsi var:

- **Caption** - keçirici üçün yazı;
- **Value** – idarəetmə elementi qurulduqda vəziyyəti göstərir (yalnız iki vəziyyət var - **True** və **False**).

Analoji olaraq, bu elementdə də **Change** əsas xassədir.

Əlbəttə, yalnız bir sayda keçiricinin istifadə edilməsi olduqca mənasız işdir. Heç olmasa ən azı 2 sayda variant olması məsləhət edilir: məntiq əsasında biri seçildikdə o birisi avtomatik olaraq, tənzimlənir. Bir çox hallarda daha çox sayda variantdan birisi seçilməli olur (məsələn, aylıq/kvartal/illik raportunun seçilməsi, raportun tipi, çap forması v.s.). Həlli çox asandır - keçiriciləri qruplaşdırmaq lazımdır. Qruplaşmanın ən sadə variantı – formada forma/əlavə istifadə etmək. Əgər keçiricilər eyni formada (və ya eyni əlavə qurmada) yerləşibse, onda onlar bir birini qarşılıqlı olaraq istisna edir. Digər halda, əgər qruplar daha dəqiq seçilibse, onda gərək **Frame** idarəetmə elementi seçilsin.

Frame – sadəcə çərçivədir, onun köməyi ilə digər idarəetmə elementlərini formada qruplaşdırmaq mümkün olur, bax şəkl. 5.13 (sağdakı piktoqram). Çərçivənin içərisinə yerləşdirilən keçiricilər, avtomatik olaraq, bir birini qarşılıqlı istisna edən statusuna keçirlər. Bəzən bu elementi VBA proqramı işlədikdə (yəni forma aktivləşdikdə) görünməz edirlər, bunun üçün **Properties Window** (xassələr pəncərəsi) dialoq pəncərəsində **BorderStyle** xassəndə 1 qiymətini qurmaq və **Caption** xassəsində qiyməti aradan götürmək (silmək) lazımdır.

5.2.8 **CommandButton** (düymə) idarəetmə elementi

CommandButton (düymə) idarəetmə elementi, **Click hadisəsi**, **Caption** **Cancel** və **Default** xassələri

CommandButton (düymə) elementi formaların tərtib edilməsində - ən çox istifadə edilən idarəetmə elementlərindəndir. Şəkl. 5.15-də **CommandButton** (düymə) idarəetmə elementinin **Toolbox**-dakı piktoqramı göstərilib. Aşağıdakı şəkl. 5.16-də **CommandButton** (düymə) idarəetmə elementinə aid forma nümunəsi göstərilib, burada **Font** xassələri, standart vəziyyətdən fərqli olaraq, dəyişdirilib, (nümunədəki formada 2 sayda **CommandButton** elementi qurulub). Formaların əksəriyyətində mütləq heç olmasa ən azı 2 sayda düymə olur: İmtina (**Cancel**) və İcra (**OK**). İmtina düyməsi basıldıqda forma bağlanmalıdır, əksinə İcra

düyməsi basıldıqda, formanın yaradılması hansı məqsədlə düşünülmüşdürsə, həmin hərəkət (hadisə) baş verməlidir.

Əsas hadisə - **Click** hadisəsidir. Bir qayda olaraq, bu hadisə program koduna birləşmiş olur.

Düymənin ən vacib xassələri:

- **Cancel** - əgər **True** qiyməti ilə qurulsa, onda düymə klaviaturadakı **<Esc>** düyməsi basıldıqda o, da basılacaq. Adətən bu cür düymələrə “*İmtina*”, “*Çıxış*” və ya “*Programın pəncərəsinə qayıt*” adları verilir. **Click** adlı hadisələr emalçısına bu cür kod əlavə etmək lazımdır: **Unload Me**
- **Caption** - düymədə yazılan mətni yaradır;
- **Default** – klaviaturanın **<Enter>** düyməsi basıldıqda formadakı düymə də basılır. Adətən bu cür düymələr əsas düymə kimi təyin edilir (hansısa hadisə icra edilir): raportun çap edilməsi, verilənlər bazasına qiymətlərin yazılması, elektron poçtla xəbərin göndərilməsi v.s;
- **Picture** – bəzi hallarda düymənin üstündə müəyyən şəklın çəkilməsi lazım olduqda istifadə edilir;
- **TakeFocusOnClick** – idarəetmə düyməsi basıldıqda ona idarəetmənin verilib verilməməsini qurur (standart vəziyyətdə **True** qiyməti yazılmış olur, yeni düyməyə idarəetmə verilir).



Şəkil 5.15 **Toobox**-dan **CommandButton** (düymə) idarəetmə elementinin seçilən piktoqramı.



Şəkil 5.16 Formada **CommandButton** (düymə) idarəetmə elementləri.

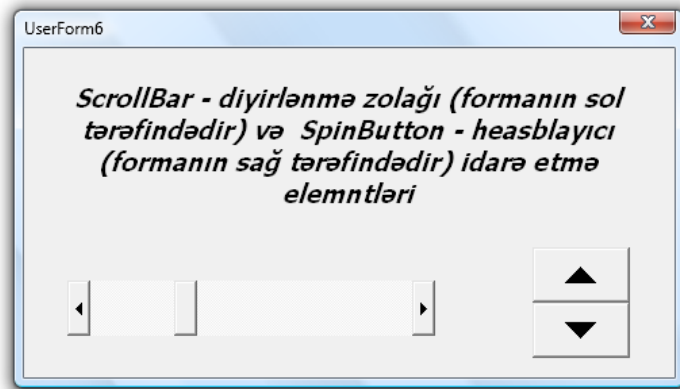
5.2.9 ScrollBar və SpinButton idarəetmə elementləri

ScrollBar (diyirlənmə zolağı) və SpinButton (hesablayıcı) idarəetmə elementləri, onların xassəsi və tətbiqi

Diyirlənmə zolaqları (ScrollBars) daha çox mətn sahələrində rast gəlinir (daxil edilən mətn tam olaraq sahəyə yerləşməyəndə). Adətən VBA Office proqramlaşdırmasında bu idarəetmə elementi kəskin dəyişməyən qiymətlərin seçilməsində istifadə edilir – məsələn, səs, parlaqlığın, hansısa rəngin, qrafik obyektin sıxılma dərəcəsinin və fəzada tutduğu vəziyyətin səviyyəsini tarazladıqda v.s. Şək. 5.17-də ScrollBar (diyirlənmə zolağı) və SpinButton (hesablayıcı) idarəetmə elementlərinin Toolbox-dan seçiləsi piktoqramları göstərilib. ScrollBar (diyirlənmə zolağı, formanın sol tərəfində yerləşdirilib) və SpinButton (hesablayıcı, formanın sağ tərəfində yerləşdirilib) idarəetmə elementlərinin formadakı nümunəsi aşağıdakı şək. 5.18-də göstərilib.



Şəkil 5.17 Toolbox-dan seçilən ScrollBar (diyirlənmə zolağı, piktoqramı soldakı şəkildir) və SpinButton (hesablayıcı, piktoqramı sağdakı şəkildir)



Şəkil 5.18 ScrollBar (diyirlənmə zolağı) və SpinButton (hesablayıcı) idarəetmə elementlərinin formadakı nümunəsi.

Bu elementin də baş hadisəsi – **Change** hadisəsidir.

Əsas xassələr aşağıda verilib:

- **Max** və **Min** – maksimal və minimal qiymətin verilməsini təyin edir (dəyişmə diapazonu: -32 767, ..., +32 767). Maksimal qiymət minimaldan kiçik də ola bilər (diyircəyi geriye çəkmək lazım gələcək);
- **LargeChange** və **SmallChange** — mausla zolağın düyməsi basıldıqda diyircəyin hərəkətinin addımını (əslində sürətini) təyin edir;

- **Orientation** – diyircəyin formadakı şaquli və ya üfüqi vəziyyətini təyin edir. Standart vəziyyətdə bu xassədə 1 qiyməti qurulub, bu deməkdir ki, standart halda diyirlənmə zolağının vəziyyəti avtomatik olaraq təyin edilir (istifadəçinin əlinin hərəkətindən asılı olaraq (həmin xassə ilə diyirlənmə zolağının vəziyyətini əvvəlcədən də vermək olar);
- **ProportionalThumb** – diyirlənmə zolağının hündəsi ölçülərini təyin edir (standart halda düymənin ölçüləri zolağın qalınlığına mütənasib olur);
- **Value** – idarəetmə elementinin əsas xassəsidir (diyirlənmə zolağının vəziyyətini və proqrama qaytardığı qiyməti təyin edir).

Adətən diyirlənmə zolağının köməyi ilə seçilən məlumatı əks etmədən bu elementin tətbiq edilməsi istifadəçilər tərəfindən bir mənalı qarşılınmır. Ən sadə variantda, diyirlənmə zolağı ilə seçilən hər nə varsa sadəcə mətn yazısında əks edilir, məsələn, aşağıdakı VBA kodu ilə yazılmış nümunədəki kimi:

```
Private Sub ScrollBar1_Change()  
    Label1.Caption=ScrollBar1.Value  
End Sub
```

Daha mürəkkəb variantda istifadəçi gerek seçsin: diyirlənmə zolağını istifadə etsin və ya qiymətləri mətn sahəsinə versin. Bu hal üçün **Change** hadisəsi üçün gerek mətn sahəsində istifadəçinin daxil etdiyi qiymətlərin yoxlanmasının təmin edilməsi nəzərə alınmalıdır və o biri tərəfdən diyirlənmə zolağı ilə əks əlaqə yaradılmalıdır. **SpinButton** idarəetmə elementinin, bu hissədə tanış olduğumuz, diyirlənmə zolağından böyük fərqi yoxdur. Bu element əsasən o halda istifadə edilir ki, seçiləsi qiymətlər diapazonu böyük deyil: məsələn, çap etmək üçün raportun kopyaları təyin edilməlidir. **SpinButton** idarəetmə elementinin xassələri həmin sayda və keyfiyyətdə **ScrollBar** elementinin xassələri ilə üst-üstə düşür.

5.2.10 **TabStrip (əlavə qurmalar yığımı) və MultiPage (səhifələr yığımı) idarəetmə elementləri**

TabStrip və MultiPage idarəetmə elementləri, formada bir neçə əlavə qurmanın olması, MultiRow, TabOrientation, Value xassələri

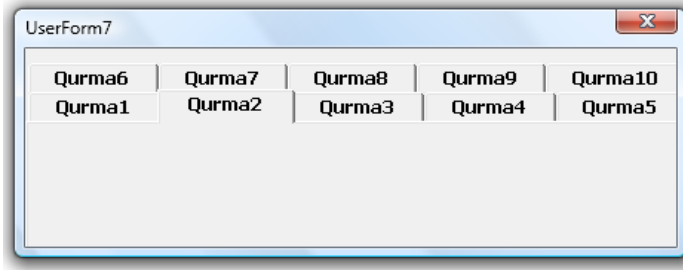
Hər iki idarəetmə elementi eyni halda istifadə edilir – tərtib edilən formada idarə elementlərinin sayı çox olduqda və buna görə onları həmin formada hamısını yerləşdirmək mümkün olmadıqda. Bu elementlərin köməyi ilə bir formada bir neçə sayda əlavə qurma (**Page**, yəni səhifə) yerləşdirmək mümkün olur. Bununla da istifadəçi formada idarəetmə elementlərini seçdikdə, bir qurmadan (səhifədən) diqqətinə asanlıqla keçə bilər. Bu elementlərin arasında olan prinsipial fərq bundan ibarətdir: **TabStrip** obyektlərində həmişə eyni idarəetmə elementləri yerləşir, ancaq **MultiPage** ilə yaradılan əlavə qurmalarda (səhifələrdə) müxtəlif idarəetmə elementləri yerləşə bilər. Buna oxşar əlavə qurmaları adi istifadəçi (ola bilsin oxucunun özü də) bir çox proqramlarda artıq görüb, məsələn, Word-də **Tools**→**Customize** və **Tools**→**Options** menyularının qurma əlavələri.

Şək. 5.19-da **TabStrip** (əlavə qurmalar yığımı) və **MultiPage** (səhifələr yığımı) idarəetmə elementlərinin **ToolBox**-dakı piktoqramları göstərilib.



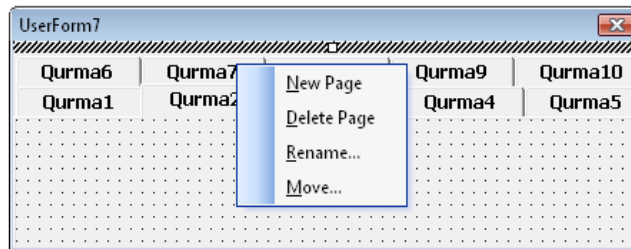
Şəkil 5.19 **Toolbox**-dan seçilən **TabStrip** (əlavə qurmalar yığımı, piktoqramın soldakı şəkildir) və **MultiPage** (səhifələr yığımı, piktoqramın sağdakı şəkildir) idarəetmə elementləri

MultiPage (səhifələr yığımı) idarəetmə elementinin formadakı nümunəsi aşağıdakı şək. 5.20-də göstərilib (bu formada 10 sayda qurma əlavə, yəni əlavə forma səhifələri yaradılıb, 2 nömrəli qurma isə şəkildəki formada aktivləşmiş vəziyyətdədir).



Şəkil 5.20 **MultiPage** (səhifələr yığımı) idarəetmə elementinin formadakı nümunəsi.

MultiPage (səhifələr yığımı) idarəetmə elementinin əlavə səhifələrini VBA redaktorunun **Form Designer Window** (forma dizaynı pəncərəsi) dialoq pəncərəsində artırmaq (silmək) və ya adını dəyişmək istədikdə elementin üstündən mausun sağ düyməsi ilə bir dəfə şıqqıldatmaq lazımdır. Dərhal şək. 5.21-də göstərilən kontekst menyusu açılır. Həmin menyudan lazım olan əməli istifadəçi seçə bilər.



Şəkil 5.21 **MultiPage** (səhifələr yığımı) idarəetmə elementinin VBA redaktorunun forma dizaynı pəncərəsində əlavə qurmaların (səhifələrin) üzərində əməllər aparmaq üçün kontekst menyusu.

TabStrip adətən daha az istifadə edilir. Məsələn, bu element verilənlər bazasının hansısa şirkətin işçilərinin (əgər işçilərin sayı həddən artıq çoxdursa) eyni şablonla verilənlərini daxil edilməsində istifadə edilə bilər. Bu elementin də xassələri və hadisələri **MultiPage** elementindəkilərdən fərqlənir.

Ən vacib olan xassələr bunlardır:

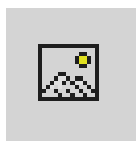
- **MultiRow** – bir neçə sayda sıralardan ibarət qurmalardan istifadə edilməsini təyin edir;
- **TabOrientation** – qurmaların formadakı vəziyyətini təyin edir (yuxarıda, aşağıda);
- **Value** – cari açılan qurmanın nömrəsini müəyyən edir (nömrələnmə 0-dan başlayır).

Change (yəni qurmalar arasında keçid) - əsas hadisə sayılır. Bu hadisəyə, məsələn, istifadəçinin daxil etdiyi qiymətlərin yoxlanmasını və ya hansısa xəbərdarlıq məlumatlarının istifadəçiyə çatdırılmasını icra edən VBA kodunu birləşdirmək olar.

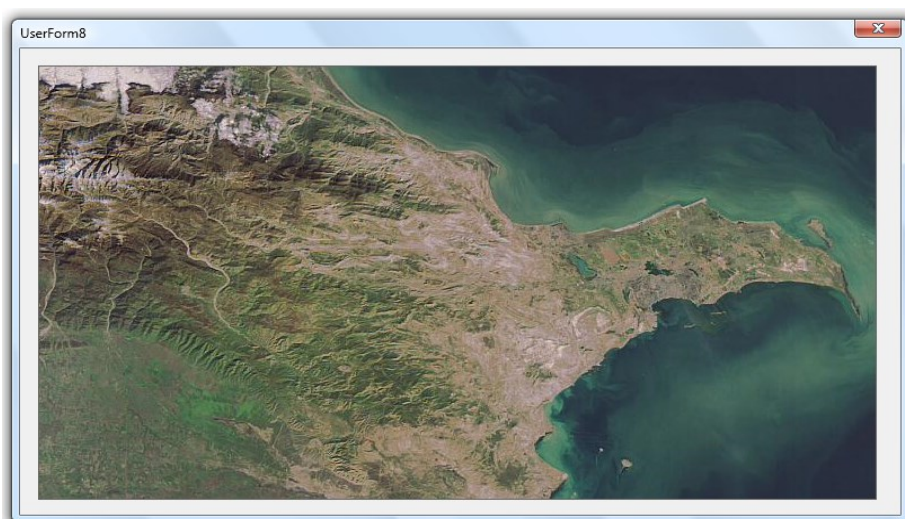
5.2.11 Image (şəkil) idarəetmə elementi

Image idarəetmə elementi, Picture xassəsi

Bu element **Toolbox**-da ən asan idarəetmə elementidir. Formada şəkli VBA-da tanınan ən populyar olan formatların birində əks edilməsinə imkan yaradır. Ola bilsin həmin şəkil hansısa hadisəyə reaksiya versin, məsələn, mausun şıqqıldamasına. Lakin ola bilsin ki, şəkil yalnız dekorativ, yaraşlıq funksiyasını daşısın. Şək. 5.22-də **Image** (şəkil) idarəetmə elementinin **Toolbox**-dakı piktoqramı göstərilib və şək. 5.23-də isə həmin idarəetmə elementinə aid forma nümunəsi gətirilib.



Şəkil 5.22 **Toolbox**-dan **Image** (şəkil) idarəetmə elementinin seçilən piktoqramı.



Şəkil 5.23 Formada **Image** (şəkil) idarəetmə elementinə aid nümunə.

Image idarəetmə elementinin istifadəsinə aid bəzi vacib qeydlər:

- alternativ kimi forma üçün **Picture** xassəsini istifadə etmək olar (xüsusilə əgər formada fon şəkli lazımdırsa);

- başqa iki alternativ - **Picture** xassəsinin **Label** yaxud **CommandButton** idarə elementlərində istifadə etmək (funksionallıq eyni səviyyədə alınır);
- bu element istifadə edildikdə görüntünün özü sənədin içərisinə kopyalanır (xarici fayl artıq lazım olmur).

Image idarə elementinin əsas hadisəsi, başqa elementlərdə olan kimi, **Change** hadisəsidir.

Əsas xassələr:

- **Picture** – formaya yerləşdiriləsi şəklin seçilib təyin edilməsini təmin edir;
- **PictureAlignment** – şəklin formadakı yerini seçməyə imkan yaradır (standart halda – şəkil mərkəzdə yerləşdirilir);
- **PictureSizeMode** - əgər element təyin edilən sahəyə uyğun gəlmirsə, onda dartılma/kiçilmə rejimlərinin seçilməsinə imkan yaradır;
- **PictureTiling** – şəkil çox kiçik olduqda şəkli bütün forma sahəsində çoxaltmağa imkan verir.

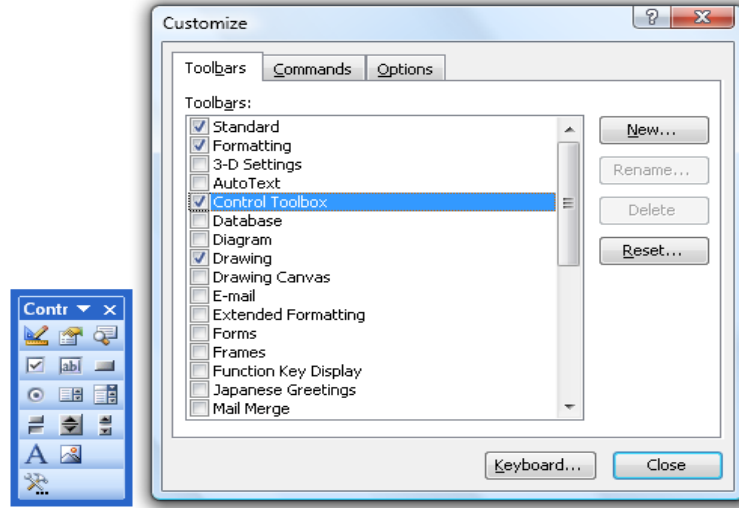
5.2.12 Əlavə idarə elementlərinin tətbiqi. **Microsoft Web Browser, Calendar, RefEdit** idarəetmə elementləri.

VBA-nın əlavə idarəetmə elementləri: **Microsoft Web Browser, Calendar, RefEdit**


Yuxarıdakı bölmələrdə oxucu **ToolBox**-da əlçatan olan standart idarəetmə elementləri ilə tanış oldu (həmin idarəetmə elementləri əvvəlcədən **ToolBox**-da qurulmuş halda olur). Nəzərə alınmalıdır ki, yalnız bu formalarla VBA-da formalar imkanları məhdudlaşmır. İstifadəçinin sərəncamında Windows-da əvvəlcədən qurulmuş olan yüzlərlə (əslində minlərlə) idarəetmə elementləri vardır (başqa məhsullarda da ola bilər və ya başqa şirkətlər tərəfindən təmin edilə bilər). Onları formada yerləşdirmək üçün mausun sağ düyməsi ilə **ToolBox** panelinin üzərində bir dəfə şıqqılatmaq lazımdır və açılan kontekst menyusundan **Additional Controls** seçilməlidir. Sonra isə lazımı elementi seçmək olar. Lakin qeyri standart elementlər tətbiq edildikdə, bir şeyi bilmək lazımdır ki, Office proqramı başqa kompyutərə köçürüldükdə həmin kompyuterdə lazım olan kitabxanaların olması vacibdir.

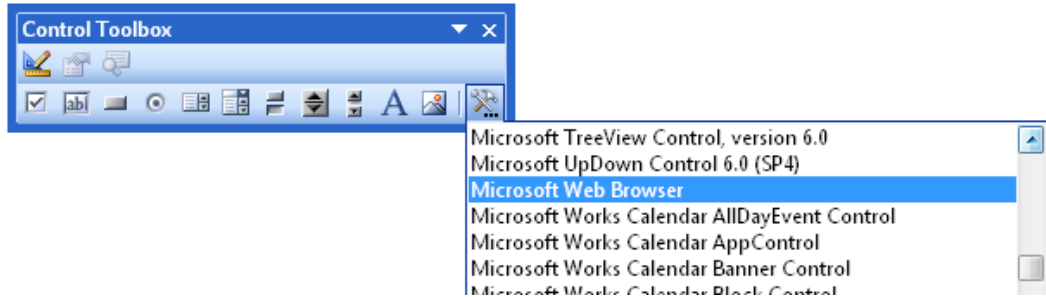
Daha çox istifadə edilən qeyri standart idarəetmə elementləri Internet Explorer, Acrobat Reader, Calendar, audio və video faylların pleyeri v.s. ola bilər. Məsələn, Internet Explorer (Microsoft Web Browser elementi) idarəetmə elementini formada yerləşdirmək üçün aşağıdakı eməlləri yerinə yetirmək lazımdır:

- **Tools**→**Customize** menyusunda **Control Toolbox** üçün seçim edilməlidir, sonra isə dialog pəncərəsi bağlanmalıdır (şək. 5.24);



Şəkil 5.24 Control ToolBox alətlər panelinin seçilməsi

- Şək. 5.24-də sol tərəfdə göstərilən açılan paneldən  piktogramı seçilməlidir (**More Controls** - əlavə idarəetmə elementləri);
- açılan kontekst menyusundakı siyahıdan (çəmi 324 idarəetmə elementi) **Microsoft Web Browser** seçilməlidir (şək. 5.25);



Şəkil 5.25 Kontekst menyusundakı siyahıdan **Microsoft Web Browser**-in seçilməsi.

- Mausun kursoru ilə Excel-də (və ya Word-də) idarə elementinə sənəddə (səhifədə) yer təyin etmək lazımdır.
- elementin üzərində mausun sağ düyməsi ilə şıqqıldadaraq, açılan kontekst menyusundan **Primary Text** seçilməlidir;

Misal üçün, **WebBrowser1**-dan ötrü **GotFocus()** hadisəsi seçilərək, həmin hadisəyə aşağıdakı kodu təyin etmək olar:

```
WebBrowser1.Navigate "http://localhost"
```

Bu halda yaradılmış idarəetmə elementi Web-saytın ev səhifəsini (standart halda istifadəçinin kompyuterində) açacaq.

Bu idarəetmə elementinin tətbiq edilməsinin üstünlüyü aşkardır - Web-səhifələrlə (məsələn, Web-formalarla) istifadəçi öz formasının funksionallığını artırabilir. Internet Explorer adətən

Windows əməliyyat sisteminin nəzarəti altında bütün kompyuterlərdə əvvəlcədən yüklənmiş olur. Buna görə bir kompyuterdən başqasına Office proqramı köçürüldükdə heç bir problem yaranmır. Bu idarəetmə elementi haqqında daha ətraflı məlumatı oxucu MSDN-dən (Microsoft Developer Network –Microsoft tərtibatçı şəbəkəsi) ala bilər. Daha bir tez-tez istifadə edilən qeyri standart idarəetmə elementi, hansı ki, hər bir kompyuterdə yüklənmiş olur - **Calendar** (kalendar) Windows elementidir. Bu elementin köməyi ilə istifadəçi lazım olan tarixi asanlıqla seçə bilir, şəkl. 5.26.

Aug 2012						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

Şəkil 5.26 **Calendar** idarəetmə elementi ilə istifadəçi hansısa qiyməti seçsə formada dərhal yazı ilə çıxarılır.

Bu idarəetmə elementin ən vacib xassəsi **Value** xassəsidir, yəni istifadəçi tərəfindən seçilmiş tarix. Başqa xassələr kalendarin xarici görüntüsü ilə bağlıdır. Excel-də daha bir maraqlı idarəetmə elementi vardır - **ReFEdit**. O, sağ tərəfində düyməsi olan, mətn qutusunda daha çox oxşayır. Elementin üstündəki düymə basıldıqda, forma “gizlənəcək”, istifadəçi isə tələb olan hücrələrin seçilməsi üçün imkan qazanacaq. İstifadəçi işini qurtaran kimi (yəni lazımı diapazon seçildikdə), dərhal forma yenidən geriyyə qayıdacaq. **ReFEdit**-də seçilmiş diapazonun koordinatları əks ediləcək (həmin qiymətləri əl ilə də daxil etmək olar. Bu elementin də əsas xassəsi **Value** xassəsidir. MS Access üçün də çox sayda əlavə idarəetmə elementləri nəzərdə tutulub, onlar bu proqram üçün spesifik xarakter daşıyır (onlar haqqında müvafiq bölmədə danışılacaq).

6. PROQRAMDAKI SƏHVLƏRİN DÜZƏLDİLMƏSİ VƏ EMALI

6.1 Səhvlərin tipi

VBA proqramlarının səhvləri: sintaktik, məntiqi və icra etmə zamanı (run-time errors)

Əgər oxucu indiyə qədər kitabın 15-ci fəsildəki sərbəst işləmək üçün nəzərdə tutulmuş tapşırıqları VBA redaktorunda müəyyən qədər yerinə yetirmişsə, onda proqram kodunun yazıldığında səhvlərin əmələ gəlməsi tez-tez rast gəlməli hal kimi oxucunun diqqətini cəlb etməli idi. Ümumiyyətlə, insan tərəfindən yerinə yetirilən hər bir işdə səhvlərin yaranması labüddür, həmçinin və xüsusilə də proqramlaşdırma sahəsində. Proqram tərtibatçısının əsas məqsədlərindən biri – vaxtında bütün səhvləri tapıb düzəltməkdən ibarətdir.

Bütün yarana biləcək səhvləri üç böyük qrupa bölmək olar:

- *sintaktik səhvlər* - məsələn, proqramdakı operator, dəyişənin adı və ya tipi düz yazılmayıb. Belə səhvlərin tapılıb düzəldilməsi çox zaman və güc tələb etmir. Çünki bir çox sintaktik səhvlər kodun daxil etmə prosesində VBA kompilyatoru tərəfindən “tutulur”. Digər səhvlər haqqında isə kompilyator proqramın işə salınması vaxtında xəbər verə bilər. VBA kompilyatoru bu cür səhvlərin proqram kodunun hansı sətirində və sütununda təyin edildiyini və bu səhvin mahiyyəti haqqında xəbər verir, üstəlik təyin edilən səhvi seçərək, həmin sətiri sarı fonda göstərir, işarələri isə qırmızı rəngə boyayır (standart halda bu rənglər qurulmuş olur). İstifadəçini dərhal həmin sətirdəki səhvi yoxlayıb (ola bilsin VBA-nın rəsmi sənədləşmə bazasına sorğu edərək) düzəltməsi məsləhət görülür;
- *məntiqi səhvlər* – proqram icra edildikdə özünü, istifadəçi əvvəldən planlaşdırdığı kimi aparmır. Əsas iş – həmin səbəbi aşkar edilib, anlanmaqdan ibarətdir. Bu cür səhvlərin aradan qaldırılması üçün, bir qayda olaraq, VBA-da sazlanma üsulları nəzərdə tutulmuşdur (növbəti bölmədə oxucu bu üsullar ilə daha yaxından tanış olacaqdır);
- *icra etmə zamanında yaranan səhvlər (run-time errors)* – proqramın icra etmə prosesində proqram elə növ səhvlə rast gələ bilər ki, onun həllinin tapılması proqram üçün (VBA kompilyatoru üçün) çox müşkül məsələyə çevrilir. Məsələn, həmin adla olan fayl artıq vardır, verilənlər bazasında yazıların bir-biri ilə konfliktli yaranıb, tam dolu olan diske yazı etmə cəhdi edilir v.s. Əvvəlcədən bu cür səhvlərin yaranma bilməsini gözləmək çox çətinidir. Proqramçının peşəkarlığı, bəzən, həmin proqramda icra etmə zamanında yaranacaq səhvlərin ehtimal edib, əvvəlcədən həmin səhvlərin aradan qaldırma bilməsi ilə təyin edilir.

Əgər istifadəçi proqramı özü üçün yazırsa, onda səhvlərin təyin edilib aradan qaldırılması adətən çox çətin məsələyə oxşamır: səhv əmələ gəldikdə proqram sazlayıcısı açılır və səhvin səbəbi təyin edilərək tez bir zamanda aradan qaldırılır. Əgər tərtib edilən proqram başqa bir istifadəçi və ya istifadəçilər (ola bilsin, peşəkarlıq səviyyələri xeyli aşağı olan) üçün nəzərdə tutulursa, onda proqramın icra zamanı yaranan səhvlərinin (**run-time errors**) aradan qaldırılmasına sərf edilən vaxt proqramın məntiqinə sərf edilən vaxtdan da böyük ola bilər.

6.2 Proqramın sazlama üsulları. Immediate, Locals və Watch pəncərələri

6.2.1 Proqramların testlənməsi

VBA-da proqramların sazlanması

İstənilən proqram təminatında işlədikdə tərtib edilən proqramın səhvsiz işləməsini təmin edən əsas üsul – proqramın testlənməsidir (yoxlanması və ya sınaqdan keçirilməsi də demək olar). İri miqyaslı proqram tərtib edildikdə, onun testlənməsinə (yoxlanmasına) sərf edilən zaman bəzən proqramın tərtib edilməsinə sərf olan vaxtdan daha böyük ola bilər. Adətən yaradılmış proqramın test edilməsini kənardan olan peşəkar “proqramçı-tester” etmir – deməli yaradılan proqramın testini elə VBA istifadəçisinin özü aparmalı olacaq. Buna görə oxucuya onun

gələcəkdə, müstəqil olaraq, tərtib ediləsi proqramlarının lazımı səviyyədə testlənməsini aparmaq üçün aşağıda bir sıra mühüm məsləhətlər veririk:

- tərtib edilmiş proqramı mümkün qədər çox sayda açılmış sənədlər olduqda və ya heç bir sənəd açılmayan zaman işə salın;
- sənədin pəncərəsi açıq/örtülü və ya ölçüsü dəyişdirilmiş olduqda proqramın necə işləməsini müşahidə edin;
- müxtəlif elementlər (və ya elementlər qrupu) seçildikdə (aktivləşmiş vəziyyətə çatdırıldıqda) proqramın necə işləməsini yoxlayın;
- əgər informasiya daxil edilərsə, onda proqrama, düşünülmüş olaraq, səhv verilənləri daxil edin. Proqramın bu verilənlərə qarşı reaksiyasını analiz etmək lazımdır. Məsələn, ən trivial halda - proqram ədədi qiymətləri gözləyir. Bu halda, məsələn, sətir qiymətlərini vermək olar və əksinə, tarix tipinə aid qiymət verilə bilər (hətta sətiri boş da qoymaq olar);
- proqramın işləməsini, ən əlverişsiz anda, qəflətən dayandıraraq sonradan yenidən işə salmaq;
- qəflətən internet əlaqəsi kəsildikdə, diskdə boş olan yer və ya printerdə çap edərkən kağız qurtardıqda və bunlara oxşar hadisələr qəflətən başlandıqda, proqramın özünü necə apardığını yoxlamaq lazımdır;
- proqramın işləməsini Office-in və əməliyyat sisteminin müxtəlif versiyalarında (həmçinin ingilis dilli və lokal dil versiyalarında) yoxlamaq lazımdır;
- proqramı işə salmamışdan əvvəl və ya işləmə prosesində, heç bir ehtimalı olmayan qiymətləri verərək, sistem tarixini və vaxtını dəyişdirmək lazımdır;
- imkan olduqda, proqramı istismar edəcəyi istifadəçinin əvəzinə proqramla bir müddət işləmək lazımdır.

Bəzən peşəkar proqramçılar proqramın testlənmə mərhələsində “diversant” üsulundan istifadə edirlər - yeganə məqsəd mümkün olan qədər proqramı sıradan çıxarmaqdan ibarətdir. Bunun üçün ən ağıla sığmaz üsullar tapmaq olar (əlbəttə, əvvəlcədən hər ehtimal üçün, proqramın kopyasını yaratmaq vacibdir). Əgər tapılan üsul “dəyərli” olsa, onda həmin üsula qarşı proqramda qabaqlayıcı müdafiə vasitələrinin yaradılması çox aktual iş sayılmalıdır. Çünki, bu heç də zarafat deyil: yenicə tərtib edilən proqramların real istismarı aparıldıqda adi sayılan istifadəçilərin işi “diversant” işindən az fərqlənir.

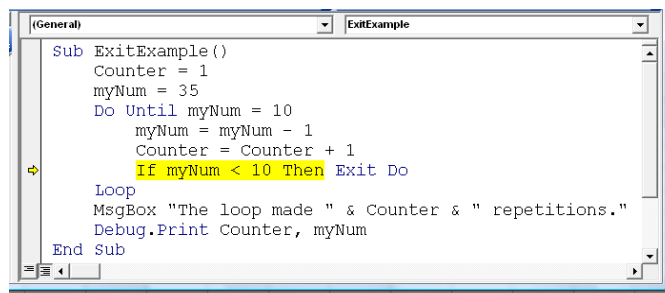
6.2.2 Fasilə rejiminə keçmə

VBA proqramlarının sazlanması və proqramdakı səhvlərin müəyyən edilməsi, proqram işlədikdə fasilə rejiminə keçmə üsulları.

Proqramın sazlanması mərhələsində ən vacib üsullardan biri – proqram işlədikdə lazımı anda proqramın işini müvəqqəti dayandırmaq (işinə fasilə vermək). Nəticədə test edən proqramçı proqramın işləməsinə, kənardan müdaxilə edərək, proqramın dayandırılan anda dəyişənlərin aldığı qiymətə baxa bilər, operatorun, funksiyaların v.s. həmin an qaytardığı qiymətlərinə baxa bilər. Proqramın işlərkən fasilə verilməsi üçün aşağıda göstərilən üsullarla diqqət yetirin:

- proqramın işləməsini addımlı icra etmə rejiminə keçirmək (**Debug**→**Step Into** menyusu ilə və ya klaviaturanın **<F8>** düyməsinin basılması ilə). Bu rejimdə, proqramdakı hər bir operator icra edildikdə, proqram fasilə rejiminə keçir (istifadəçi yenidən növbəti operatorun icra edilməsinə icazə versə - yalnız bu halda proqram işini davam edəcək);
- proqramda (**breakpoint**) kəsilmə nöqtəsi (işin dayandırılması nöqtəsi) verilir. Buna nail olmaq üçün mausla istiqamətləndiricini proqram kodunun dayandırılma təyin edilən sətirdə yerləşdirmək lazımdır və **Debug** menyusunda **Toggle Breakpoint** seçilməlidir (yaxud da klaviaturanın **<F9>** düyməsi basılmalıdır). Təsvir edilən əməli bu cür də etmək olar: həmin seçilmiş sətirin sol tərəfdəki pəncərə çərçivəsində mausun sağ düyməsi ilə bir dəfə şıqqıldadaraq, açılan kontekst menyusunda görünən əməlləri seçmək. Normal vəziyyətə qayıtmaq üçün həmin əməlləri yenidən təkrar etmək lazımdır;
- əgər kəsilmə nöqtəsinin yeri və funksiyası yaddaşda saxlanmalıdırsa (sənəd bağlandıqda, breakpoint nöqtəsi yadda qalmır), onda həmin sətirdən ya əvvəl, ya sonra **stop** təlimatı yazılmalıdır. **stop** operatoru, istifadəçi tərəfindən silinə qədər, hər iş seansında, avtomatik olaraq, proqram seçilmiş sətirin icrasından əvvəl (və ya sonra) fasilə rejiminə keçəcək;
- əgər proqram işini dayandırmırsa (məsələn, sonsuz hesablama dövrü olduqda), onda proqram işlərkən, məcburi olaraq, **Run** menyusundan **Break** əmri seçilə bilər (və ya klaviaturanın **<Ctrl>+<Break>** düymələri basıla bilər);
- daha bir imkan - **Watches** pəncərəsində **Watch** əmri seçilməklə həyata keçirilə bilər.

Hər bir halda proqramın icrası istifadəçinin seçdiyi yerdə dayandırılacaq və növbəti icra ediləsi operator proqram kodunda sarı rənglə haşiyələnəcək, şəkl. 6.1.



```

Sub ExitExample()
    Counter = 1
    myNum = 35
    Do Until myNum = 10
        myNum = myNum - 1
        Counter = Counter + 1
        If myNum < 10 Then Exit Do
    Loop
    MsgBox "The loop made " & Counter & " repetitions."
    Debug.Print Counter, myNum
End Sub

```

Şəkil. 6.1 Kəsilmə nöqtəsi seçilib (sarı rənglə haşiyələnib) və proqram fasilə rejimindədir.

6.2.3 Fasilə rejimindəki əməllər

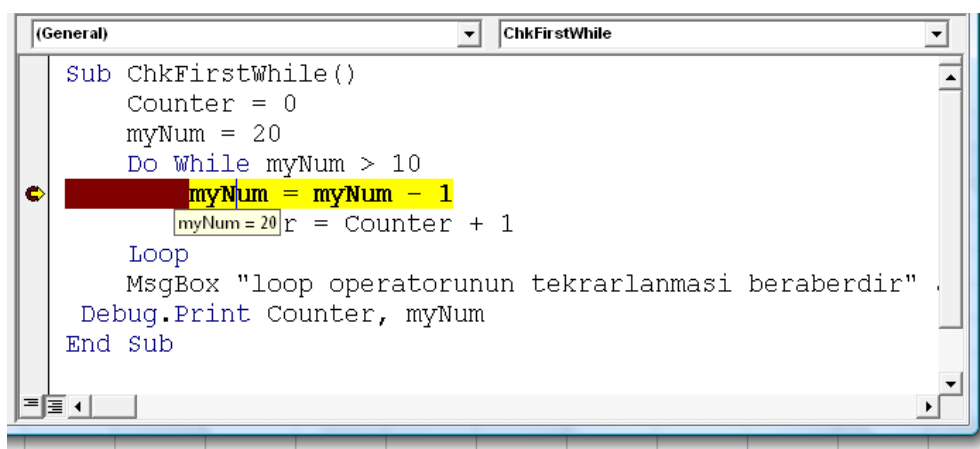
VBA proqramlarının sazlanması və proqramdakı səhvlərin müəyyən edilməsi, fasilə rejimindəki əməllər, addımlı icra, proqram kodundakı sətərə keçid, dəyişənlərin qiymətinə baxma imkanı.

Fasilə rejiminə ona görə daxil olurlar ki, hansısa əməllərə başlamaq mümkün olsun. Fasilə rejimində adətən daha çox aşağıdakı əməlləri yerinə yetirirlər:

- proqramın icrasını addımlı rejimdə davam etmək üçün. Bunun üçün **Debug**→**Step Into** menyusundan istifadə etmək lazımdır (və ya klaviaturanın **<F8>** düyməsi basılmalıdır);
- əgər proqram kodunun sətirindən hansısa proseduranın (funksiyanın) çağırılması icra edilirsə və o, çağırılan prosedura artıq sazlanıbsa (yeni həmin prosedura və ya funksiya ilə heç bir problem yaranmırsa), onda istifadəçi proseduranı dayandırmadan proqram kodunun başqa operatorun icrasına keçə bilər. Bunun üçün **Debug**→**Step Over** menyusu seçilir (və ya klaviaturada **<Shift>+<F8>** düymələr kombinasiyası basılır);
- əgər buna baxmayaraq istifadəçi proseduraya daxil olsa, orada bezi dəyişiklik etsə və oradan dərhal çıxaraq işi başa çatdırmaq istəyirsə, onda gerek **Debug**→**Step Out** menyusu seçilsin (və ya klaviaturada **<Ctrl>+<Shift>+<F8>** düymələr kombinasiyası basılsın);
- əgər, məsələn, istifadəçi proqram kodunun addımlı rejimdə icrasını yox, hissə-hissə icra edilməsini istəyir (məsələn, böyük hesablama dövrlərini addımla icra etməmək üçün). Bunun üçün mausun sağ düyməsi ilə proqram kodunun lazımı hissəsində bir dəfə şıqqıldadaraq, yaranan kontekst menyusundan **Run to Cursor** təlimatını seçmək lazımdır (və ya **Debug** menyusundan həmin təlimatı seçmək, yaxud da klaviaturada **<Ctrl>+<F8>** düymələr kombinasiyasını basmaq). Nəticədə proqramın icrası istifadəçi seçdiyi hissəyə irəli gedəcək və kursor olduğu yerdə dayanacaq;
- əgər yaranan sahəyə görə, proqram kodunun hansısa hissəsindən “sıçrayıb” çıxmaq lazımdırsa, onda **Debug**→**Set Next Statement** menyusu təlimatı seçilməlidir (və ya klaviaturada **<Ctrl>+<F9>** düymələr kombinasiyası basılmalıdır). Sonra isə sarı rəngdə olan nişanlamaları sol sınırla aşağı (yuxarı) çəkmək lazımdır. Əməllərin buraxılmasının (nəzərə alınmamasının) bir üsulu da budur – “buraxılması” olan hissəni mausla seçmək. Sonra **Edit** alətlər paneli ilə onu şərh etmək (**comment**). Yoxlamadan sonra həmin sətiri gerek yenidən şərh halından çıxarmaq (**uncomment**) lazımdır;
- əgər proqram koduna baxıldıqda xeyli irəli gedilibsə, onda dayanma nöqtəsinin axtarışına çox vaxt itirməmək üçün - **Debug** alətlər panelindən **Show Next Statement** təlimatı seçilməlidir;
- dayanmadan və sonra proqramı yalnız yenidən icra etməyini başlamaq üçün - **Run** menyusundan **Continue** təlimatı seçilməlidir (**Run** menyusunda **Run** sözünün əvəzinə **Continue** sözü əmələ gəlir) və ya klaviaturanın **<F5>** düyməsi basılmalıdır. Proqramın

dayandırılması lazım olduqda, həmin menyudan **Reset** əmri seçilməlidir (və ya klaviatürada **<Alt>+<F4>** düymələr kombinasiyası basılır). Bundan əlavə istifadəçinin sərəncamında eyni adlı düymələr **Standard** alətlər panelində vardır;

Daha çox hallarda fasilə rejiminə keçmək bu hallarda vacib olur: dəyişənlərin cari qiymətlərini bilmək lazım olduqda; proqram kodunun düzəlişində (redaktəsində). Dəyişənlərin cari qiymətlərini **Immediate**, **Locals** və **Watch** dialoq pəncərələrində görmək olur (onlar haqqında aşağıda daha geniş məlumat verəcəyik). Başqa (daha asan) üsul da bundan ibarətdir: kodun özündə mausun kursoru lazımı dəyişənin üzərinə gətirilməlidir. Dərhal həmin işarənin üstündə alınan cari qiymət göstərilir (bunun üçün standart vəziyyətdəki kimi **Tools**→**Options**→**Auto** menyusundan **Data Tips** parametri seçilir). Şək. 6.2 görmək olar ki, nümunədə proqramın işləməsinin dayandırılan anında **myNum** adlı dəyişəninin cari qiyməti bərabərdir 20).



```
(General) ChkFirstWhile
Sub ChkFirstWhile()
    Counter = 0
    myNum = 20
    Do While myNum > 10
        myNum = myNum - 1
        myNum = 20
        Counter = Counter + 1
    Loop
    MsgBox "loop operatorunun tekrarlanmasi beraberdir"
    Debug.Print Counter, myNum
End Sub
```

Şəkil 6.2 VBA-da dəyişənlərin fasilə rejimində cari qiymətlərinə mausun kursorunu lazımı dəyişənin üzərinə gətirməklə cari qiymətin göstərilməsi mümkündür.

Dəyişənin görünmə sahəsi və tipi haqqında olan məlumatı almaq üçün daha asan üsul budur – mausun kursorunu dəyişənin üstündə qoyaraq, **Edit** menyusundan **Quick Info** seçmək (və ya klaviatürada **<Ctrl>+<I>** düymələr kombinasiyasını seçmək).

Sazlama rejimində proqram kodunun bütün redaktə etmə imkanları qalmaqda olur. Əgər istifadəçi anlayıbsa ki, hansısa dəyişənə (obyektin xassəsinə) düzgün olmayan qiymət mənimsədilib, onda həmin an səhv düzəldilə bilər və geriye **Set Next Statement** (və ya **<Ctrl>+<F9>**) təlimatı ilə qayıdaraq, proqramın icrasını düzəldilmiş qiymətlərlə davam etmək olar. Nəzərə alınmalıdır ki, şəkl. 6.2-dəki üsul ilə proqramın sazlanması bəzən kifayət etmir, məsələn, dəyişənlərin sayı həddən artıq çox olduqda və ya xassənin qiyməti təyin ediləndən fərqli alınırsa. Belə hallarda yalnız **Immediate**, **Locals** və **Watch** dialoq pəncərələrindən istifadə edilməsi yaxşı nəticə verir.

6.2.4 Immediate pəncərəsi

VBA proqramlarının sazlanması və proqramdakı səhvlərin müəyyən edilməsi, Immediate pəncərəsi, dəyişənlərin qiymətlərinin görmək imkanı, Immediate pəncərəsində əməllərin icrası.

VBA-da **Immediate** pəncərəsi proqramın sazlanması mərhələsində ən güclü dialog pəncərəsidir.

Immediate pəncərəsində (onu **View** menyusundan yaxud klaviaturanın **<Ctrl>+<G>** düymələr kombinasiyası ilə çağırmaq olar) dəyişənlə və obyektlərə aid xassələrin qiymətlərini görmək və, lazım olduqda, dəyişmək mümkündür. Həmin əməlləri bu cür təşkil etmək olar:

```
Print nResult  
Print oDoc.FullName
```

Və ya daha sadə formada:

```
?nResult  
?oDoc.FullName
```

Baxdığımız halda **Print** operatoru **Debug** obyektinin metodudur. Başqa üsul ilə **Immediate** pəncərəsinə çıxışı bu obyektə və ya sadəcə proqram kodundan həyata keçirmək olar:

```
Debug.Print nResult
```

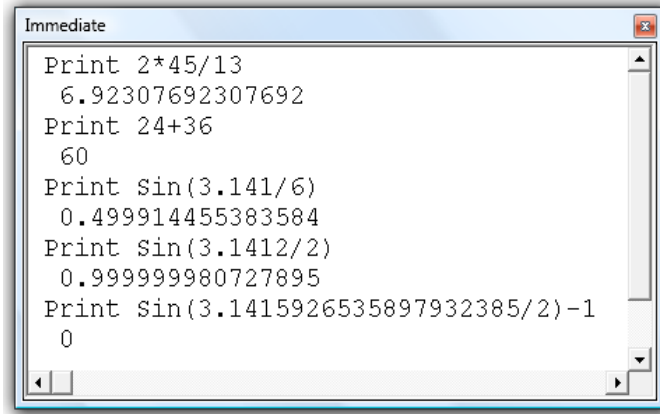
Adi **MsgBox()** metodundan bu metodun fərqi ondan ibarətdir ki, sazlanma mühitindən kənar halda işlədikdə (yəni tərtib edilən proqramın artıq hansısa istifadəçi tərəfindən istismar edildiyi şəraitdə), **Debug** obyektinin bütün çağırışları cavabsız qalır (nəzərə alınmır). Bu obyektin bir metodu da var: **Assert()** – şərtə görə keçid.

Immediate pəncərəsində dəyişənlərə və xassələrə aid olan qiymətlərin redaktə edilməsi (dəyişdirilməsi) tamamilə proqram kodunun dəyişdirilməsinə oxşayır.

Eyniliklə proqram kodundan çağırılan kimi, həmçinin **Immediate** pəncərəsində tərtib edilmiş proqramdakı proseduraları, funksiyaları və ya obyektlərin metodlarını çağırmaq mümkündür. Microsoft məsləhət görür ki, istifadəçilər potensial təhlükəli olan proqramların (məsələn, sistemi sıradan çıxarda bilən proqramların) yoxlanmasını məhz həmin pəncərədə keçirsinlər.

Həmin pəncərəni adi kalkulyator kimi də istifadə etmək mümkündür (məsələn, şəkl. 6.3-də göstərilən, nümunələrdə olan kimi).

Immediate pəncərəsində proqram kodundakı dəyişənlərin adını yenidən çap etməmək üçün proqram kodundan (proqram kodu pəncərəsindən) kodun hissələrini və ya dəyişənin özünü (klaviaturanın **<Ctrl>** düyməsini basılı saxlamaq şərti ilə - kopyalamaq həyata keçirilsin deyə), sürükləyərək çəkib-atmaq üsulu (**drag and drop**) ilə **Immediate** pəncərəsində yerləşdirmək olar.



Şəkil 6.3 **Immediate** pəncərəsinin adi kalkulyator kimi işlədilməsinə aid nümunələr.

6.2.5 **Locals** pəncərəsi

VBA proqramlarının sazlanması və proqramdakı səhvlərin müəyyən edilməsi, Locals pəncərəsi, Locals pəncərəsində dəyişənlərin və obyektlərin qiymətlərinin görmək və dəyişdirmək imkanı

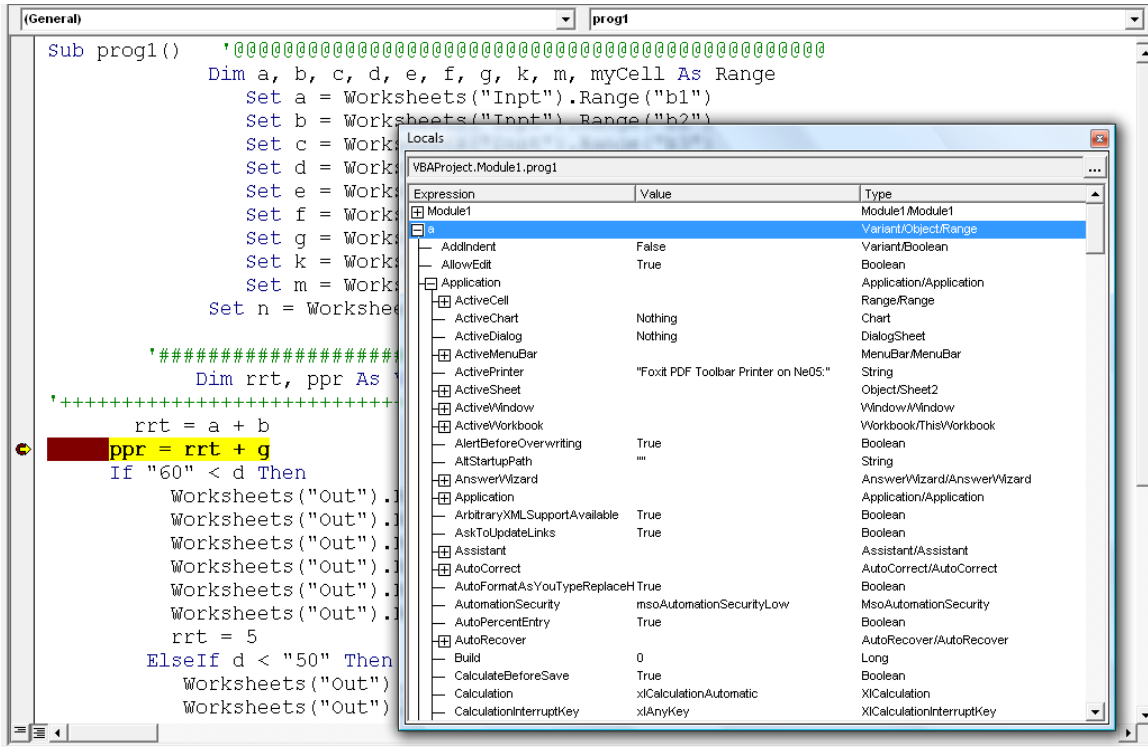
Proqramın sazlanma mərhələsində tez-tez bu cür hal yaranır: proqramı tərtib edən istifadəçi proqramdakı bütün dəyişənlərin və obyektlər xassələrinin qiymətlərini görüb nəzərdən keçirmək istəyir. Məqsəd – VBA qaydaları ilə yolverilməz qiymətləri düzəltməkdən ibarətdir. Belə olduqda, **Immediate** pəncərəsi artıq əlverişsiz olur. Çünki hər bir xassə və ya dəyişən ilə pəncərədə “əlləşmək və qurdalanmaq” əməlləri aparılan sazlanma işinin məhsuldarlığını aşağı salmış olacaq. Bu cür hallarda daha əlverişli mühiti **Locals** pəncərəsi yaradır.

Locals dialoq pəncərəsində proqramdakı hal hazırda əl çatan olan bütün dəyişənlər və obyektlər xassələrinin qiymətlərini görmək və nəzarət etmək mümkündür.

Birinci növbədə nəzərə alınmalıdır ki, **Local** pəncərəsi yalnız proqramın addım-addım işləmə rejimində və ya kəsilmə nöqtəsi (**BreakPoint**) təyin edildikdə aktivləşir. Dəyişənin (və ya xassənin) qiymətini dəyişmək (redaktə etmək) üçün kod proqramının lazım olan sətirini **Locals** pəncərəsində seçərək, sonra həmin sətirdə mausun köməyi ilə (səliqəli) **Value** sütununda qiyməti seçmək lazımdır. Növbəti addımda köhnə qiymətin üstündən yeni qiyməti çap etmək olar. Əgər qiymət sətir tipinə aiddirsə, onda çap etdikdə onu kodda olan kimi dırnaqlar arasında çap etmək lazımdır. Digər halda - qiyməti tarix tipinə aid olan (#) işarəsi ilə əhatə etmək lazımdır.

Locals pəncərəsindən çoxluqlar (arrays) və kolleksiyalar elementlərinin də qiymətlərini dəyişdirmək mümkündür.

Aşağıdakı şəkil 6.4-də **Locals** pəncərəsinə aid proqram sazlanması zamanı yaranan nümunə göstərilib (**Locals** pəncərəsi bu nümunədə VBA kodunun redaktə pəncərəsinin fonunda (proqram kodunun üstündən) yerləşdirilib. Əslində **Locals** pəncərəsini VBA redaktor pəncərəsinin, müvafiq olan, istənilən yerində yerləşdirmək olar.



Şəkil 6.4 Locals pəncərəsinin VBA proqramının sazlanması mərhələsində istifadə edilməsinə aid nümunə.

6.2.6 Watches pəncərəsi

VBA proqramlarının sazlanması və proqramdakı səhvlərin müəyyən edilməsi, Watches pəncərəsi, Watches pəncərəsində müşahidə edilən ifadələrin yaradılması.

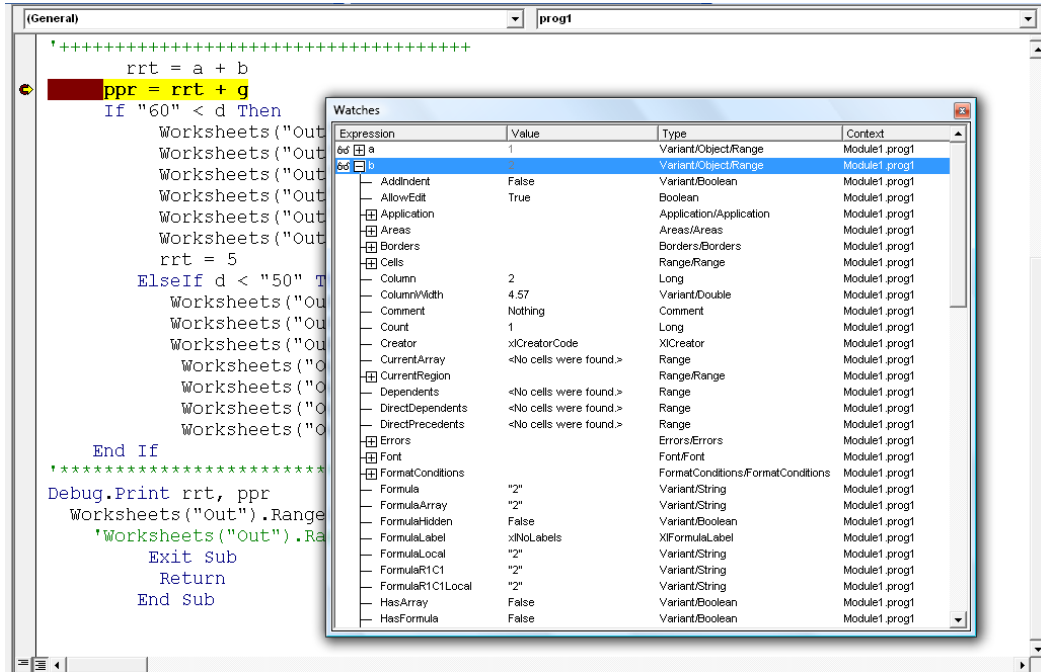
Watches dialoq pəncərəsi (şək. 6.5), istifadəçi tərəfindən təyin edilən qaydada, proqramın icra edilməsinə nəzarət etməyə imkan yaradır. Bu pəncərə ilə işləməkdən əvvəl istifadəçiyə hansısa qiyməti bilmək lazım olmalıdır – bununla proqrama müdaxilə edilməsi üçün siqnal olacaq. Bu qiymət “nəzarət edilən ifadə” adlanır. Həmin ifadə çox sadə də ola bilər, məsələn, `nResult=10`, ancaq daha mürəkkəb də ola bilər, məsələn:

```
InStr(oDoc.FullName, "Document") <> 0
```

Nəzarət edilən ifadəni yaratmaq üçün aşağıdakı qaydalardan istifadə etmək olar:

- redaktor pəncərəsində dəyişən/xassə/ifadə üzərində mausun sağ düyməsi ilə bir dəfə şıqqıldadaraq əmələ gələn kontekst menyusundan **Add Watch** təlimatını seçmək lazımdır. Əmələ gəlmiş **Add Watch** dialoq pəncərəsində avtomatik olaraq həmin ifadə yerləşdiriləcək.
- **Debug** menyusundan **Add Watch** təlimatından istifadə etmək;
- **Debug** menyusundan **Quick Watch** təlimatından istifadə etmək. Bu halda **Watch** pəncərəsində hazır ifadə yerləşdirilmiş olacaq. “İşləmə” şərti kimi orada dəyişənin/xassənin cari qiyməti yerləşdiriləcək. Həmin qiyməti lazım olduqda redaktə etmək mümkündür;

- ifadəni mausla çəkib atmaq üsulu ilə **Watches** pəncərəsində yerləşdirmək (bu pəncərəni o biriləri kimi **View** menyusundan açmaq olar);
- istənilən halda **Add Watch** pəncərəsi açılacaq. Bu pəncərədə istifadəçi nəzarət edilən ifadəni yazmalıdır və ya yazıb qurtarmalıdır) – nəticədə həmin ifadə **IF** konstruksiyasına oxşar olaraq, **True** və ya **False** qiymətini qaytara biləcək.



Şəkil 6.5 **Watches** pəncərəsinin VBA proqramının sazlanması mərhələsində istifadə edilməsinə aid nümunə.

Növbəti addımda baxılan *nəzarət ediləsi ifadə* üçün “təsir sahəsi” seçilir (prosedura və ya modul formasında). Sonra isə əsas qərarı qəbul etmək lazımdır: müşahidə etdikdə nə etmək olar? Burada istifadəçinin sərəncamında üç variantda seçim ola bilər:

- heç bir əməl etməmək (sadəcə **Watches** pəncərəsində **Value** sütununda qiyməti dəyişmək);
- proqramı fasilə rejiminə keçirmək (əgər *nəzarət edilən ifadə* “işlədisə”, onda onun qiyməti **True** olacaq);
- proqramı fasilə rejiminə keçirmək (əgər *nəzarət edilən ifadənin* qiyməti dəyişdisə);

Watches pəncərəsi proqramda baş verdiyi hadisələri müşahidə etməyə imkan verir (hətta ən ağır olan hadisələrdə belə: məsələn, təyin etmək olmur ki, nəyə görə proqram məhz bu cür işlədi, başqa cür yox).

Aşağıdakı şəkil 6.5-də **Watches** pəncərəsinə aid proqram sazlanması zamanı yaranan nümunə göstərilir (**Watches** pəncərəsi bu nümunədə VBA kodunun redaktə pəncərəsinin fonunda (proqram kodunun üstündən) yerləşdirilib. Əslində **Watches** pəncərəsini VBA redaktor pəncərəsinin istənilən yerində yerləşdirmək olar.

6.3 Proqramın icra etmə zamanı səhvlərinin təyin edilməsi və aradan qaldırılması

VBA proqramlarının sazlanması və proqramdakı səhvlərin müəyyən edilməsi, icra zamanı səhvləri (runtime errors), icra səhvlərinin aradan qaldırılması, On Error GoTo əmri, Err obyektı

Proqram tərtibatçısı üçün ən ağır səhvlər - icra etmə zamanı kateqoriyalı səhvləridir (**runtime errors**). Onların yaranma ehtimalları müxtəlifdir [18]: istifadəçi buraxıla bilməyən qiyməti daxil edib, eyni adla olan fayl artıq vardır, istifadəçi daxil etdiyi verilənləri verilənlər bazasının serveri qəbul etməyə imtina edir, şəbəkə əlaqəsi qopmuşdur v.s. *İcra etmə zamanı səhvləri* yarandıqda adətən tətbiqi proqramın işləməsi avariya xassəli şəkildə kəsilir. İstifadəçiyə isə qurulmuş olan elə bir xəbər verilir ki, ondan baş açmaq, onu deşifrə etmək, mümkün olmur. Buna görə VBA proqramı yarandıqda ən çətin mərhələ budur - istifadəçi proqramı istismar etdikdə hansı səhvlər əmələ gələ bilər və həmin səhvlərin təyin edib hansı üsul ilə aradan qaldırmaq olar.

Səhvin emal edilməsinin ümumi prinsipi aşağıdakı kimi ifadə edilə bilər:

- təhlükəli proqramın qarşısında (faylın saxlanması/açılması, sifira bölünmə ehtimalı v.s.)

```
On Error GoTo səhvin_emalçısının_işarəsi
```

əmrini yerləşdirilir, məsələn, bu cür:

```
Dim a As Integer, b As Integer, c As Integer
```

```
On Error GoTo ErrorHandlerDivision
```

```
c=a/b
```

- sonra kod proqramında səhvin emalçısının işarəsi və emalı proqram kodu yerləşdirilir:

```
ErrorHandlerDivision:
```

```
MsgBox "Bölmədə yaranan səhv"
```

Nəzərə alsaq ki, belə hallarda səhv emalçısının kodu istənilən halda işə salınacaq və icra ediləcək (hətta səhv olmasa belə), onda səhv emalçısının işarəsi qarşısında **Exit Sub** əmrinin yerləşdirilməsi məqsədə uyğun olar (əgər o, proseduradırsa) və ya **Exit Function** (əgər o, funksiyadırsa). Həmin mini-proqramın VBA kodu aşağıdakı kim ola bilər:

```
Private Sub UserForm_Click()
```

```
Dim a As Integer, b As Integer, c As Integer
```

```
On Error GoTo ErrorHandlerDivision
```

```
c=a/b
```

```
Exit Sub
```

```
ErrorHandlerDivision:
```

```
MsgBox "Bölmədə yaranan səhv"
```

```
End Sub
```


Bir qayda olaraq, əgər səhvi düzəltmək üçün imkan varsa – bu işi səhv emalçısında düzəldirlər (və ya həmin imkanı istifadəçiyə verirlər). Əgər belə imkan yaranmırsa – istifadəçiyə izahlı məlumat verilir və proqram işini bitirir.

Səhvin emalçı kodu icra edildikdən sonra istifadəçi seçim qarşısında qalır: səhv yaranan proseduranın icrasını davam etdirmək, yaxud həmin proseduranın işini dayandıraraq icra etməni onu çağıran (əsas) proseduraya ötürmək. Beləliklə, istifadəçinin sərəncamında üç seçim variantı yaranacaq:

- səhvi yaradan operatoru bir daha icra etmək (əgər səhv emalçısı yaranan problemləri aradan qaldırırsa). Bunun üçün səhv emalçısına **Resume** əmrini yerləşdirmək kifayətdir;
- səhvi yaradan operatoru icra etməyərək “buraxmaq”. Bunun üçün səhv emalçısına **Resume Next** əmrini yerləşdirmək olar;
- proqram kodunun müəyyən edilmiş yerindən başlayaraq icra edilməsini davam etdirmək. Bunun üçün səhv emalçısına **Resume nişan** əmrini yerləşdirmək lazımdır (nişanla işləmə sintaksisi **GoTo** əmrinin sintaksisinə bənzəyir).

Səhvlərin emalı ilə daha bir neçə vacib qeydlər bunlardır:

- kodun təhlükəli hissəsindən keçdikdən sonra normal iş rejiminə keçmək üçün

On Error GoTo 0

əmrəndən istifadə etmək olar (səhvlərin emalından imtina etmək üçün);

- istifadəçinin sərəncamında

On Error Resume Next

əmri də var. Sadəcə bu əmr kompilyatora təlimat verir ki, bütün yaranan səhvləri nəzərə almasın və növbəti operatorun icrasına keçsin. Praktiki işlərdə çox tez-tez təhlükəli operatorun icrasından əvvəl

On Error Resume Next

yazılır və sonra

Select ... Case

yoxlama strukturu ilə (**Err** obyektinin xassəsinin əsasında) yaranan səhvin nömrəsi təyin edilir. Bundan asılı olaraq, proqramın sonrakı icrası təşkil edilir.

Err obyektinin haqqında bir qədər ətraflı məlumat verək. Bu obyektin iki əsas xassəsi və iki əsas metodu vardır:

- **Number** – bu xassə səhvin nömrəsini bildirir. Adətən səhv emalçısında elə həmin ədəd yoxlanılır (hansı səhvin yaranması yoxlanılır). Əgər səhvin nömrəsi 0 bərabərdirsə, onda hər şey normal vəziyyətdədir – səhv olmamışdır

- **Description** – yaranan səhvin mətn ifadəsidir. Həmin bu izahlar da birinci növbədə istifadəçiyə qaytarılır (adətən proqramı istismar edən istifadəçi bu mətndən heç bir şey anlaya bilmir). Belə məlumatlar daha çox proqram tərtibçisi üçün nəzərdə tutulur;
- **Clear** - bu metod **Err** obyektini səhvlər haqqında olan köhnəlmiş məlumatlardan təmizləyir. Həmin əməli **On Error GoTo 0** əmri də yerinə yetirir;
- **Raise** – proqramda səhvi generasiya edərək, ona səhvin nömrəsi və təsvirini ötürmək imkanlarını yaradır. Əgər real səhvin modelləşdirilməsi çətindir, onda bu imkan proqramın davranışının yoxlanması üçün çox xeyirli imkandır.

Qeyd etməliyik ki, səhvlərin emalı bir tərəfdən olduqca etibarlı, o biri tərəfdən isə xeyli resurs tutumlu iş metodudur. Əgər tərtib edilən VBA proqramında səhvin tutulması və generasiya edilməsi imkanından vaz keçmək mümkündürsə, onda elə bu cür də etmək daha əlverişlidir (məsələn, istifadəçi tərəfindən daxil edilən qiymətləri qurulmuş standart funksiyalarla yoxlamaq). Bununla belə, tərtib edilən proqramda real yarana biləcək avariya tipli səhvlərin qarşısını almaq üçün səhvlərin emalçı proseduralarının proqramda olması – tərtib edilən VBA proqramının böyük müsbət cəhətidir. Bəzən də bu imkan böyük ehtiyacdən yararılır və proqramın etibarlı işləməsi üçün yeganə çıxış yolu olur.

7. YARDIMÇI İLƏ İŞLƏMƏ METODLARI

İstifadəçi ilə qarşılıqlı təmas yaratmaq üçün Office-də yardımçının tətbiqi, VBA-da Assistant və Balloon obyektləri.

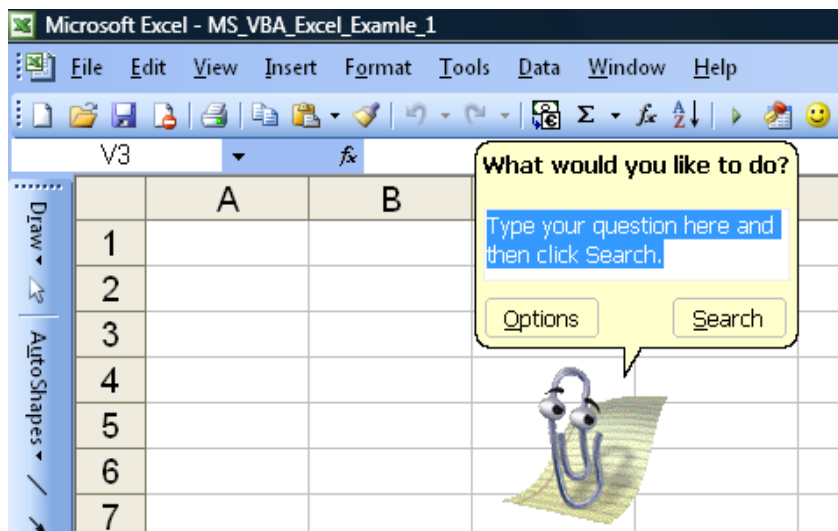
Office-də hansısa məlumatı istifadəçiyə çıxarmaq üçün VBA formalarının idarəetmə elementlərini və ya **MsgBox()** funksiyasını tətbiq etmək məcburi deyil. Bunun üçün Office proqramlarında daha sadə və əlverişli üsullar var: Office proqramlarına yardımçı (məsləhətçi) qurulub – mehriban görüntülü köpək və ya pişik balası, skrepka v.s. Bu yardımçının tətbiqi çox faydalı və əlverişli ola bilər, xüsusilə onun pəncərəsinin qeyri modallığı nəzərə alınsa (yəni yardımçı istifadəçiyə hansısa məsləhətini verdikdə, istifadəçi eyni zamanda işini rahatca davam edə bilər).

Yardımçı ilə işləmək üçün iki obyektəndən istifadə edilir [3]: **Assistant** və **Balloon**. **Assistant** – yardımçının özüdür, **Balloon** – onun yanında olan mətn pəncərəsidir (istifadəçi burada ona xeyirli olan məsləhətləri oxumaqla bərabər öz sorğusunu da sistemə verə bilər), şəkl. 7.1.

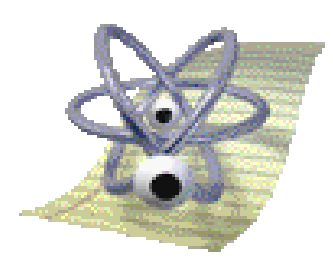
VBA proqram kodundan **Assistant**-in işə salınması bu cür həyata keçirilə bilər:

```
Sub assistant_animation()
  Assistant.On=True
  Assistant.Visible=True
  Assistant.Animation=msoAnimationThinking
End Sub
```

Yuxarıdakı kodun işləməsi nəticəsində yardımçının aşağıdakı animativ görüntüsü yaranacaq, (şək. 7.2).



Şəkil 7.1 İngiliscə "Paper clip" adlı yardımçı (Office Assistant).



Şəkil 7.2 VBA kodun işləməsi nəticəsində əmələ gələn yardımçının animativ görüntüsü

Assistant obyektinin ən vacib olan xassələri və metodları bunlardır:

- **Animation** – adından məlumdur ki, bu xassə hərəkətli görüntü yaradır. Animasiya qurtardıqda həmin xassənin təyin etdiyi vəziyyətdə dəyişməz vəziyyətdə qalır, şək. 7.2-dəki kimi.
- **FeatureTips** – bu xassə yardımçını avtomatik vəziyyətə keçirərək, istifadəçiyə müxtəlif əməliyyatların icrası üçün daha effektiv məsləhətlər verir (hər halda Office tərbiyəçiləri belə düşündür);
- **FileName** – bu xassə ilə istifadəçi uyğun olan yardımçını seçə bilər (Office-də cəmi 7 sayda yardımçı görüntüsü qurulmuş olur). İstifadəçi internetdən yardımçıya yüzlerle başqa görüntü tapa bilər (lakin, onda gerek onları əvvəlcədən yardımçılar kataloquna *.acs genişlənməsi ilə kopyalamaq lazımdır). Məsələn, əgər yardımçı pişik balası görüntüsündə çağırılmalıdırsa, onda aşağıdakı koddan istifadə edilməlidir:

```
Assistant.FileName="Offcat.acs"
```

- **GuessHelp** – bu xassə işə salındıqda yardımçı (onun iddiası ilə istifadəçiyə xeyirli olan) sorğunun mövzuları çıxarılır;
- **KeyboardShortcutTips** – yardımçı müxtəlif əməlləri canlandırmaq üçün klaviatura kombinasiyaları haqqında məlumatları verəcək;
- **Left** – yardımçı sol tərəfə “sıçrayacaq”;
- **MoveWhenInTheWay** – əgər istifadəçi müəyyən yerdə iş görəcəksə, yardımçı avtomatik olaraq, kənara gedəcək;
- **NewBalloon** – bu xassə avtomatik olaraq, yeni **Ballon** obyektini qaytarır (yəni həmin yeni pəncərədə proqram tərtibatçısının proqram istismarçısına verdiyi məlumat mətn kimi göstəriləcək). Əslində bu əməli başqa cür də yerinə yetirmək olar:

Assistant.NewBalloon.Heading="Bu əməkdaşın soy adı yoxdur mu?"

Lakin bu cür də etmək olar:

```
Dim Ball
Set Ball=Assistant.NewBalloon
```

İstənilən halda **Balloon** (hərfi olaraq - köpük) görsənsin deyə, **Show()** metodu çağrılmalıdır:

```
Ball.Show
```

- **On** – bu xassə yardımçını işə salır (əgər o, həmin vaxt keçirilmiş vəziyyətdədirsə);
- **Reduced** – yardımçının ölçüsünü kiçildir;
- **SearchWhenProgramming** – VBA redaktoru işlədikdə yardımçının aktiv olmasını qurur;
- **Sounds** – səs effektlərinin qurulması/keçirilməsi;
- **TipOfDay** – Office-in hər vaxt işə salındığında, yardımçı xüsusi yığım toplusundan hansısa məsləhəti verəcək;
- **Top** – yardımçının yerləşdiyi yerdən ekranın ən yuxarı küncünə olan məsafəni təyin edir;
- **Visible** – yardımçını gizlətmək/göstərmək funksiyasını yerinə yetirir;

Yardımçının metodları ilə istifadəçi sorğu ilə bağlı aşağıdakı əməlləri yerinə yetirə bilər:

1. cavabların alınması ustasını işə sala bilər (**StartWizard()**);
2. standart pəncərəni çağırır (**Help()**);
3. yardımçını başqa yerə köçürə bilər (**Move()**) v.s.

Yardımçını yaradıb onu tənzimləyəndən sonra növbəti mərhələ - **Balloon** obyektini ilə olan işdir (yəni tərtibatçının istifadəçiyə göndərmək istədiyi mətn məzmunlarının formalaşdırılması).

Ballon obyektinin əsas xassələri bunlardır:

- **BalloonType** – köpüyün tipini təyin edir (üç sabitdən birini seçmək olar: ballonda düymələr yığımının nömrələnmiş siyahının və ya markalanmış siyahının olmasını təyin edir);
- **Button** – köpüyün aşağı hissəsindən düymələr yığımını seçməyə imkan yaradır;
- **Callback** – köpüyə hansısa proseduranın bağlanmasına imkan yaradır (sonra həmin proseduranı oradan işə sala bilər);
- **Checkboxes** – köpükdə keçirici idarəetmə elementlərinin yerləşdirilməsinə imkan yaradır;
- **Heading** – köpüyün başlığını təyin edir (burada başlığın işarələrinin ölçüsünü, rəngini və ya əlavə görüntüsünü təyin etmək olar);
- **Icon** - köpüyün sol yuxarı küncündə piktoqramı yerləşdirməyə imkan yaradır;
- **Labels** - nişanları düymələrə və ya siyahılar elementlərinə bağlamaq üçün imkan verir;
- **Mode** - balonun işləmə rejimidir. İstifadəçinin sərəncamında aşağıdakı imkanlar var:
 1. **msoModeModal** (standart halda) – tətbiqi proqramla ilə işləməyi davam etmək üçün köpüyü bağlamaq lazım olacaq;
 2. **msoModeModeless** – köpüyə məhəl qoymayaraq, proqramda işləmək olar;
 3. **msoModeAutoDown** – köpüyün yox olması üçün proqramın pəncərəsinin istənilən yerində mausla bir dəfə şıqqıldatmaq lazımdır.
- **Text** – köpüyün istifadəçiyə hansı məlumatın çıxarılmasını təyin edir. Adi mətnlə birgə müxtəlif görüntüləri çıxarmaq və şriftləri dəyişdirməyə imkan verir.

Əgər **Button** düyməsi ilə xüsusi düymələr yığını təyin edilirsə, onda istifadəçinin hansı düyməyə basdığını öyrənmək üçün aşağıdakı VBA kodundan istifadə etmək olar:

```
intButton=Ball.Show  
və ya bir başa  
Select Case Ball.Show
```

8. ALƏTLƏR PANELLƏRİ VƏ MENYU İLƏ İŞLƏMƏ QAYDALARI

VBA-da alətlər panelləri və menyular ilə işləmə qaydaları, CommandBar, CommandBarControl, CommandBarPopup obyektləri, kontekst menyusu

VBA proqramlarında tez-tez belə hallar yaranır [18]: tərtib edilən proqram üçün standart alətlər panelləri və menyular (proqramda bunlar qurulmuş olur) əvəzinə şəxsi (proqramın müəllifi tərəfindən hazırlanmış, yeni tərtib edilmiş), orijinal xassəli, alətlər panelləri və menyular tələb olur. Alətlər panelləri və menyular ilə işləməyi **CommandBars** kolleksiyası təmin edir (**Application**

obyektində yerləşir). Növbəti hissələrdə bu vacib obyekt haqqında daha ətraflı danışacağıq. **CommandBars** kolleksiyası **CommandBar** obyektlərini öz tərkibində saxlayır. Üstəlik həmin obyektlərin hər biri **CommandBarControls** kolleksiyasını öz tərkibində saxlayır (panel/menyu strukturunu təşkil edən elementləri). Əslində menyü həmin o, elementlər yığımından qurulur. Həmin elementlər bunlardır:

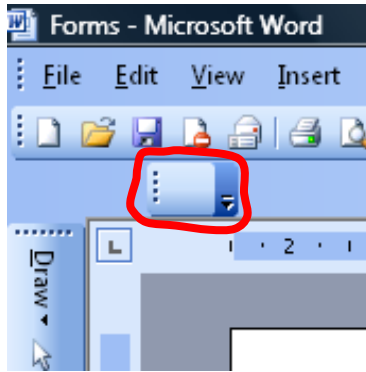
- **CommandBarButton** – proqramın (altproqramın) icrası üçün istifadə edilən düymə və ya menyü elementi;
- **CommandBarComboBox** – menyü/panel strukturunun mürəkkəb elementidir (verilənləri daxil etmə sahəsi, açılan siyahı, siyahılı sahə);
- **CommandBarPopup** – menyü və ya iç menyü;

Özül alətlər paneli yaratmaq istədikdə aşağıdakı VBA proqram kodundan istifadə etmək olar:

```
Dim CBar1 As CommandBar
```

```
Set CBar1=CommandBars.Add("Dokument", msoBarTop)  
CBar1.Enabled=True  
CBar1.Visible=True
```

Bu kod əsasında, məsələn, Word sənədindəki VBA redaktorunun köməyi ilə kiçik bir proqram tərtib edildikdə, nəticədə aşağıdakı şəkl. 8.1-də göstərilən standart Word paneli altında, ölçüsü böyük olmayan, bir yeni, hələlik adsız, alətlər paneli əmələ gələcək (şəkl. 8.1-də əmələ gələn panel qırmızı rəngli çərçivəyə alınıb).



Şəkil 8.1 Word sənədindəki VBA prosedurası ilə Word sənədində yeni tərtib edilən özül boş olan alətlər panelinin əmələ gəlməsi.

Biz, yuxarıdakı VBA kodunda təyin edilən *Dokument* adı ilə, yeni alətlər paneli yaratdıq. Bu paneli, məsələn, **Tools**→**Customize** menyusu ilə aradan götürmək olar. Yaranmış panel tamamilə xeyirsizdir: tərkibində heç bir idarəetmə elementi yoxdur. İdarə etmə elementlərinin əmələ gəlməsi üçün gerek həmin menyunun **CommandBarControls** kolleksiyasına **CommandBarControl** tipli yeni element (yuxarıdakı siyahıda adları çəkilənlərdən) əlavə edilsin.

Bu nümunə üzərində müəyyən əməllər etməmişdən əvvəl, **CommandBar** obyektinin əsas xassələri və metodlarını qeyd edək:

- **BuiltIn** – bu xassə verilmiş panel/menyu strukturunun həmin proqramda qurulmuş olub-olmamasını yoxlayır (yəni Microsoft-un standart elementidirmi) Bu xassənin qiymətini dəyişdirmək məsləhət görülmür: çünki lazım olduqda standart menyuları (və ya özül menyuları) aradan götürmək olar;
- **Context** – tərtib edilmiş özül menyuda VBA proqram kodunun yerini təyin edir (**Normal.dot**-da, sənəd faylında v.s.). Eyni adlı müxtəlif menyuların çağırılması ehtimalı olduqda, yoxlama kimi istifadə edilə bilər;
- **Controls** – bu xassənin sayəsində **CommandBarControls** idarəetmə elementləri kolleksiyasını almaq olur (baxdığımız nümunədə bu xassə bizə çox lazım olacaq);
- **Enabled** - panelin “qoşulması/keçirilməsi”;
- **Height, Left, Top və Width** – bu xassələr proqramdakı pəncərədə menyunun yerləşməsinə təyin edirlər;
- **Index, Name və NameLocal** – xassələri **CommandBars** kolleksiyasında axtarılan panelin tapılmasını təmin edirlər. **Name** – obyektin proqram adı, **NameLocal** – istifadəçi görəsi addır, **Index** - baxılan panelin nömrəsidir;
- **Protection** – tərtib edilən pənədə köhnə düymələrin silinməsi (və ya yenilərinin əlavə edilməsini) qadağa edir;
- **Type** – olduqca vacib olan xassədir: menyupanel strukturunun funksiyası nədən ibarət olacaq (alətlər paneli tipli olacaq, adi panel olacaq və ya mausun sağ düyməsinin bir dəfəlik şıqqıltı ilə açılan kontekst tipli menyupanel olacaq). Bu xassə redaktə üçün əlçatan deyil. **CommandBar** obyektinin tipi **CommandBars** kolleksiyasının **Add()** metodunun icrasında müəyyən edilir: metodun ikinci parametrinin sayəsində. Baxdığımız nümunədə həmin metodun çağırışı belə olacaq:

```
Set CBar1=CommandBars.Add("Dokument",msoBarTop, True, False)
```

Metodun birinci parametri panelin adıdır - "Dokument". İkinci parametr isə **msoBarTop** – panelin vəziyyətini müəyyən edir: birləşəndir (**msoBarTop**, **msoBarBottom**, **msoBarLeft**, **msoBarRight**) və ya göstərir ki, panel birləşməyib (**msoBarFloating**), yaxud da, mausun sağ düyməsinin şıqqıldamasından əvvəl müəyyən edilə bilməyən, kontekst menyusu olacaq (**msoBarPopup**). **Visible** xassəsi kontekst menyusu üçün buraxıla bilən deyil.

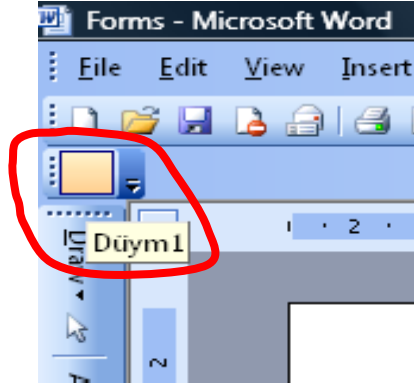
Nəticədə tərtib edilən panelin hansı forma alacağı onun tərkibinə yerləşdirilən idarəetmə elementlərdən asılı olacaq. Baxdığımız nümunədə panelin yaradılması tamamlandıqda, onun üzərinə (tərkibinə) idarəetmə elementlərini yerləşdirmək lazımdır. Məsələn, tutaq ki, alətlər paneli bu formada yaradılır:

```
Dim But1 As CommandBarControl
Set But1=CBar1.Controls.Add(msoControlButton)
```



```
But1.Caption="Düym1"
```

Proqrama bu fraqment əlavə edirik və proqramı yenidən VBA redaktorunda işə salırıq. Alınan görüntü, birinci baxımda, əvvəlki menyudan heç bir şeylə fərqlənməyəcək - mausun kursoru ilə panelin başlanğıcına apardıqda, dərhal izah edici yazı "Düym1" ilə boş düymə əmələ gəlir, şəkl. 8.2.



Şəkil 8.2 Tərtib edilən **Dokument** panelində VBA kodu sayəsində **Düym1** adlı, görünməyən, düymə tipli idarəetmə elementinin əmələ gəlməsi.

Tədricən dəyişdirilmiş VBA kodundan görünür ki, idarəetmə elementinin (panelin üstündə düymələr) yaradılması üçün biz **CommandBar** obyektinin (biz onun adını `CBar1` kimi kodda yazmışıq) **Controls** kolleksiyasının **Add()** metodundan istifadə etdik. Bu kolleksiyanın da metodları, bir çoxlarında olan kimi, standartdır:

- **Application** xassəsi – Office proqramının (Word, Excel v.s.) obyektinə istinadı qaytarır;
- **Count** xassəsi – kolleksiyada neçə sayda idarəetmə elementinin yerləşdiyini müəyyən edir;
- **Item** xassəsi – kolleksiyada **CommandBarControl** idarəetmə elementinə istinadı indeksə (nömrəyə) görə alınmasını təmin edir;
- **Add()** – metodu kolleksiyaya idarəetmə elementinin əlavə edilməsi üçün imkan yaradır (biz təqdim edilən nümunədə ondan istifadə etmişik);
- **Delete()** – metodu kolleksiyadan idarəetmə elementinin silinməsini təmin edir.

Çox vacib olduğuna görə **Add()** metoduna aid bir qədər ətraflı məlumat verək: bu metod 5 sayda parametr qəbul edir ki, bunlardan ikisi (siyahıda birinci sırada duran) çox vacibdir. Birinci parametr – **Type** (kolleksiyaya ötürülən idarə elementinin tipini təyin edir). Tiplərin sayı isə beşdir:

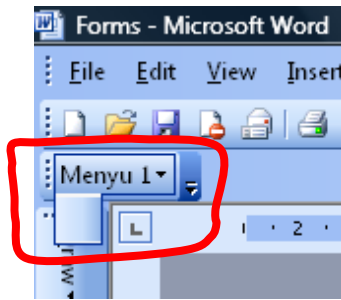
- **msoControlButton** - idarəetmə düyməsidir (alətlər panelini yaratmaq üçün);
- **msoControlEdit** - mətni daxil etmək üçün sahə yaradır;

- **msoControlDropDown** – açılan siyahını yaradır;
- **msoControlComboBox** - kombinəli siyahı yaradır;
- **msoControlPopup** – menyuda punkt yaradır.

Məsələn, biz baxdığımız nümunədə düyməni açılan siyahının punktuna çevirmək üçün gerek aşağıdakı VBA fraqmentindəki kimi - yalnız həmin bir parametrlə dəyişdirilsin, şəkl. 8.3:

```
Dim But1 As CommandBarControl
Set But1=CBar1.Controls.Add(msoControlPopup)
But1.Caption="Menyu 1"
```

Add() metodunun ikinci vacib parametri – **ID** parametridir. Bu parametrlə sayəsində idarəetmə elementini artıq sistemdə mövcud olan qurulmuş idarəetmə elementinə bağlamaq olur. Məsələn, nümunəmizdə çap düyməsini əlavə etmək üçün bu parametrlə qiyməti gerek 4-ə bərabər olsun, sənədin qabaqcadan baxım funksiyasını yerinə yetirən düyməni əlavə etmək üçünse parametrlə qiyməti 5-ə bərabər olmalıdır. Əgər **ID** parametrlənin qiyməti boş qalsa və ya 1 bərabər olsa, onda elə bir istifadəçi elementi yaradılacaq ki, o, heç bir qurulmuş sistem elementinə bağlı olmayacaq.



Şəkil 8.3 Tərtib edilən **Dokument** adlı paneldə, VBA kodunda dəyişiklik etməklə, yeni menyü punktunun əmələ gəlməsi.

Add() metodunun digər parametrləri idarəetmə elementinin identifikatorunun yaradılmasına, elementin başqa elementlərə görə vəziyyətinin təyin edilməsinə və elementin daimi (yaxud da mehvəqqəti) qalmasına aiddir. Əlbəttə, alətlər/menyü paneli ilə iş bununla qurtarmır. Əslində idarəetmə elementlərini axıra qədər sazlamaq lazım olacaq. Məsələn, birinci nümunədəki düymə üçün ən azı biz gerek onun üzərindəki görüntünü müəyyən edək və düymə basıldıqda üstündə yazını əmələ gətirən proseduranı tərtib edib VBA proqramına əlavə etməliyik. Buna görə, aşağıda verilən, **CommandBarButton** obyektinə aid vacib olan xassələri və metodları bilmək lazımdır:

- **Caption** – idarəetmə elementinin üzərində yazını əmələ gətirir (düymə üçün “üzərək çıxan” məsləhət, menyü punktu üçün isə punktun adı kimi əmələ gəlir);
- **Enabled** – idarəetmə elementinin qurulmuş və ya keçirilmiş halında olduğunu bildirir. Adətən istifadəçinin səhvini qabaqlamaq üçün istifadə edilir;

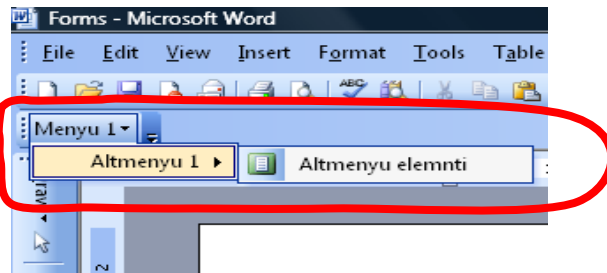
- **FaceId** (yalnız düymələr üçündür) – düymə üçün müvafiq funksiya təyin etməyərək, üzərində sistem şəkillərinin birinin (sistem piktoqramlarında birinin) yerləşdirilməsini təmin edir. Məsələn, 4 qiyməti verildikdə düymədə printerin piktoqramı yaranacaq. Word və Excel-də minlərlə hazır piktoqram şəkilciklər quraşdırılıb, buna görə hazır piktoqramın seçilməsi üçün geniş imkanlar var;
- **Id** – düyməyə təyin edilmiş funksiyanın identifikatorudur. Əgər düyməyə istifadəçi proseduranı təyin etsə, onda **Id** qiyməti həmişə 1 olacaq;
- **Index - Controls** kolleksiyasında idarəetmə elementinin nömrəsidir. İdarəetmə elementləri ilə müxtəlif xidməti əməlləri yerinə yetirmək üçün istifadə edilir;
- **Mask** – şəkil obyektinin yalnız bir hissəsini göstərmək üçün şəkil üzərinə maska qoyulmasını təmin edir. Maska ayrıca şəkil kimi görünür (şəffaf hissələr ağ, şəffaf olmayan hissələr isə qara olmalıdır);
- **OnAction** – idarəetmə elementinin ən vacib xassəsidir: onun qiyməti idarəetmə elementinin aktivləşməsində istifadə edilir (mausun düymə və ya menyu üzərində şıqqılması, mətn sahəsində mətnin daxil edilməsinin qurtarması, siyahıdan yeni qiymətin seçilməsi). Əsas məqsədi tətbiqi proqrama (**COM**, **Add**, **In**) və ya işə salınan proseduraya işarə etməkdən ibarətdir. Məsələn: `But1.OnAction="MySub"`.
- **Parameter** – bu xassə çağırılan proseduraya parametrlərin ötürülməsində və ya sadəcə istifadəçinin özünün verilənlərinin saxlanılmasında istifadə edilir **string** tipli verilənlərlə işləyir.
- **Picture** – bu xassə düyməyə hansısa piktoqramı (şəkilciyi) təyin edilməsinə xidmət edir. Adətən bunun əvəzinə yuxarıdakı **FaceId** xassəsindən istifadə edirlər. Ehtiyac yarandıqda lazımı piktoqramları geniş çeşiddə MS Visual Studio-dan və ya sərbəst olaraq yüklənən şəkillər generatorundan seçmək olar;
- **ShortcutText** – düyməyə və ya menyuya təyin edilmiş klaviatura düymələri kombinasiyaları haqqında məlumatı mətn formasında saxlanmasını təmin edir;
- **State** – düymənin görkəmini təyin edir (adi, basılmış və ya ətrafında çərçivə olan);
- **Style** – düymənin görkəmi ilə bağlı xassədir (piktoqram formasında, yalnız üstündə yazı ilə və ya hər ikisi birdən – hamısı bir yerdə və müxtəlif variantda);
- **ToolTip** – “üzərək çıxan” məsləhətin mətnini müəyyən edir (standart halda **Caption** xassəsinin qiyməti “üzərək çıxır” təyin edilmiş olur);
- **Type** – idarəetmə elementinin tipini qaytarır (tipin dəyişməsinə imkan vermir!);
- **Visible** – idarə elementinin görünən olub olmasını təyin edir;
- **Delete ()** – metodu düymələr kolleksiyasından düymənin silinməsinə təmin edir;

- **Execute()** – metodu **OnAction** hadisəsi ilə təyin edilən hər nə varsa işə salınmasını təmin edir;
- **Reset()** – metodu düymənin parametrləri dəyişdirildikdə, başlanğıc vəziyyətə qaytarılmağa imkan yaradır (burada yeganə **Click** hadisəsi var – düymənin basılmasına reaksiyanı təyin edir).

Yuxarıdakı xassələr və metodları bilərək, alətlər panelini yaratmaq kifayətdir. Açılan və kontekst menyularınının tərtibatı bir qədər çətinidir. Açılan menyuda elementlərin içəriyə yerləşdirmə parametrləri təyin edilməlidir və kontekst menyusunda isə gərək onun hansı obyektə bağlanması təyin edilsin. İç-içə qurulmuş menyularla işləmək üçün isə, nümunədə biz artıq istifadə etdiyimiz, **Controls** kolleksiyası lazım olacaq. Yeganə fərq: **Controls** kolleksiyası bu halda **CommandBar** obyektinə yox, **CommandBarPopup** obyektinə aid olacaq (başqa menyuya aid olacaq). Oxucunun izlədiyi nümunədə bu hal üçün (iç-içə qurulmuş menyu üçün) VBA kodu bu cür yazılacaq:

```
'CommandBar standart obyektini yaradırıq
Dim CBar1 As CommandBar
Set CBar1=CommandBars.Add("Dokument", msoBarTop)
CBar1.Enabled=True
CBar1.Visible=True
Dim Menul As CommandBarPopup
Dim SubMenul As CommandBarPopup
Dim SubMenulItem As CommandBarButton
Set Menul=CBar1.Controls.Add(msoControlPopup)
'Yuxarıdakı menyunu yaradırıq
Menul.Caption="Menyu 1"
Set SubMenul=Menul.Controls.Add(msoControlPopup)
'İç-içə qurulmuş menyunu yaradırıq
SubMenul.Caption="Altmenyu 1"
'İç-içə qurulmuş altmenyuda element yaradaraq ona Procl
'prosedurasını təyin edirik
Set SubMenulItem=SubMenul.Controls.Add(msoControlButton)
SubMenulItem.FaceId=5
SubMenulItem.Caption="Altmenyu elementi"
SubMenulItem.OnAction="Procl"
```

Yuxarıdakı VBA fraqmentindən yeni modul təşkil edib (məsələn, proqramın adını `Dokument_2` kimi yazsaq) və sonra VBA redaktorunda kompilyasiya edib işə salsaq, aşağıda şəkl. 8.4-dəki nəticəni alacağıq.



Şəkil 8.4 Tərtib edilən `Dokument_2` adlı paneldə, VBA kodunda tərtib edilmiş prosedurasının işləməsi nəticəsində iç-içə qurulmuş yeni menyunun əmələ gəlməsi.

Əlbəttə, idarəetmə elementlərini yalnız öz menyularına yox, sistemin qurulmuş menyularına da əlavə etmək olar. Əlavə etmə qaydası eynidir - lazımı qurulmuş menyunu **For ... Each** dövrü hesablama strukturu və ya **Name** xassəsinin qiymətinin yoxlanması ilə tapmaq olar.

Kontekst menyusu (VBA sorğu sənədində - **shortcut menus**) – mausun sağ düyməsinin bir dəfə şıqqıltısı ilə açılan menyudur. Onunla iş bu cür VBA kodu əsasında yaradıla bilər:

```
Set CBar1=CommandBars.Add("Mənim kontekst menyum", msoBarPopup, True)
Set MenuItem1=CBar1.Controls.Add
MenuItem1.FaceId=3
MenuItem1.Caption="Menyu elementi 1"
Set MenuItem2=CBar1.Controls.Add
MenuItem2.FaceId=5
MenuItem2.Caption="Menyu elementi 2"
```

Gördüyünüz kimi, VBA proqramlaşdırılması ilə özül alətlər paneli və menyuların yaradılması bir o qədər də çətin iş deyil - standart əməlləri bilmək vacibdir. Əgər yuxarıdakı son VBA fraqmenti işə salınsa, onda kontekst menyusu əmələ gəlməyəcək. Çünki aşağıdakı **ShowPopup()** metodunun çağırışı əlavə edilməmişdir:

```
CBar1.ShowPopup
```

Bu halda kontekst menyusu həmin an masunun markeri olduğu yerdə yaranacaq (bu metod üçün əmələ gəlmə yerinin koordinatlarını da vermək olar). Əlbəttə, bu metodu hadisələr emalçısına yerləşdirmək olar - nəzərə alınmalıdır ki, məsələn, Excel-də səhifə üçün **BeforeRightClick** hadisəsi varsa da, Word-də bu hadisə yoxdur. Lakin bu halda da başqa bir imkandan istifadə etmək olar: kontekst menyusuna istifadəçi tərəfindən tərtib edilən punktlar əlavə etməklə.

9. VERİLƏNLƏR BAZASI İLƏ İŞLƏMƏ QAYDALARI VƏ ADO OBYEKT MODELİNİN TƏTBİQİ

9.1 Verilənlər bazası ilə işləmək nə üçün vacibdir

Office proqramlarında verilənlər bazasının elementləri. VBA və verilənlər bazaları.

Verilənlərin sayı çox olduqda, təbii olaraq, avtomatlaşdırmaya ehtiyac yaranır. Bu da aydındır ki, verilənlərin sayı çox olduqda onları bu və ya digər formada verilənlər bazasında saxlanması zəruri üsullardandır - indiyə qədər daha əlverişli üsul hələ kəşf edilməmişdir. Deyilənlər istənilən verilənlərə aiddir (həmçinin sənədlərə, qrafik verilənlərə, arxivlərə, video/audio fayllara v.s.).

Office proqramlarında və onların əsasında yaradılan proqramlar öz-özlüyündə çox faydalı ola bilərlər. Onlar verilənlər bazaları ilə birgə işə qoşulduqda, faydalıq əmsalı dəfələrlə artmış olacaq. Real işlərdə Word sənədləri daha çox Excel-də yaradılmış verilənlər bazasından götürülən informasiyanın müxtəlif raportların və işgüzar hesabatların generasiyasında istifadə edilir. Access-in özü elə verilənlər bazasını idarə edən mükəmməl bir sistemdir. Praktiki işlərdə daha çox onu SQL Server və Oracle müştəri-servis verilənlər bazasına, müştəri interfeysini quraraq, informasiyanı daxil etmək üçün istifadə edirlər. Kitabın müvafiq hissələrində oxucu

Excel-in verilənlər bazası sahəsində yalnız elektron cədvəlləri kimi yox, daha geniş və hərtərəfli sistem olduğu haqqında kifayət qədər məlumat alacaq. Oxucu artıq hiss etməli və anlamalıdır ki, işlərin bu cür səviyyədə yerinə yetirilməsi Visual Basic for Applications proqramlaşdırma dilinin tətbiqi sayəsində mümkün ola bilər.

Office proqramlarından verilənlər bazasına istinad edilməsi, praktiki olaraq, istənilən müəssisə və şirkətdə (hətta fundamental elmlərlə məşğul olan elmi tədqiqat institutlarında belə, məsələn, experimental laboratoriya və rəsədxanalarda v.s.) gündəlik və vacib məsələ kimi yaranır. Bir çox hallarda, başlanğıcda verilənlərlə işləmək üçün nəzərdə tutulmuş Office proqramları (məsələn, Excel səhifəsi və ya Word cədvəli), işin həcmi artdıqca həmin proqramların müştəri-server mənbələri ilə uyğunlaşdırmaq lazım gəlir. Buna görə kitabın hal-hazırkı fəslində biz VBA köməyi ilə aşağıdakı işlərin hansı qaydada yerinə yetirilməsi ilə tanış olacağıq:

- verilənlər bazasına qoşulmaq qaydaları;
- müxtəlif informasiyanın verilənlər bazasından yüklənməsi və, istinad edərək, lazım olan informasiyanın təsvir edilməsi;
- yeni yazıların verilənlər bazasına əlavə edilməsi;
- verilənlər bazasında mövcud olan verilənləri silmək və ya dəyişmək.

Əgər oxucunun verilənlər bazası ilə indiyə qədər heç bir tanışlığı və təcrübəsi olmamışdırsa – qorxmaq lazım deyil. Praktiki təcrübə göstərir ki, bu sahədə müəyyən işlərə başlamaq üçün verilənlər bazası üzrə təcrübəli administrator qədər bilmək məcburi deyil. Bu kitabın diqqətli oxucusu bir neçə günə verilənlər bazası ilə işləmə üsullarını mənimsəyə bilər. Bunun üçün kitabla bərabər (paralel olaraq) kompyuterdə də müvafiq iş getməlidir. Yəni “*bu kitab oxucunun əlinin altında olmalıdır, kompyuter qarşısında*” (əslində kitabın bu üslubla öyrənilməsi kitabın bütün fəsillərinə və bölmələrinə aiddir).

Bu fəsildə verilənlər bazası ilə işləməyin ümumi (universal) üsullarından bəhs ediləcək [12, 15, 18]. Əgər həmin üsullar yaxşı mənimsənilsə, onda başlayan VBA istifadəçisi istənilən verilənlər bazası ilə Office proqramlarından (hətta digər proqram təminatlarından belə) işləyə biləcək:

- müştəri-server prinsipli verilənlər bazası ilə (məsələn, Microsoft SQL Server, Oracle, IBM DB2);
- bir qədər sadə olan verilənlər bazası ilə (məsələn, Access, FoxPro, Dbase və ya Paradox);
- hətta daha sadə olan Excel-ə verilənlər bazası ilə.

9.2 Verilənlər bazası və ADO

VBA proqramlarında ADO obyektleri, MDAC, ADO haqqında sorğu sənədləri, Connection, Command və Recordset obyektleri

ADO termininin açılması budur: **ActiveX Data Objects**. ADO strukturu, əslində **ActiveX (COM)** texnologiyasına əsaslanan, müxtəlif mənbələrdən verilənlərin alınmasını və idarə edilməsini təmin edən, proqram obyektleri toplusuna deyirlər. Yaxın keçmişdə Office-də bu məqsədlə başqa proqram obyektlərindən də istifadə edilə bilər, məsələn: **DAO** və **RDO**. Həmin proqram obyektleri artıq köhnəlib və buna görə Microsoft müasir Office proqramlarında onlardan istifadə etməyə məsləhət görmür. Hal-hazırda, **ADO**-dan kəskin olaraq fərqlənən, **ADO**-nun yeni versiyası olan **ADO.NET** yaradılıb (**.NET Framework**-da işləmək üçün nəzərdə tutulub). Həmin versiyanın bir sıra cəhətlərinə görə:

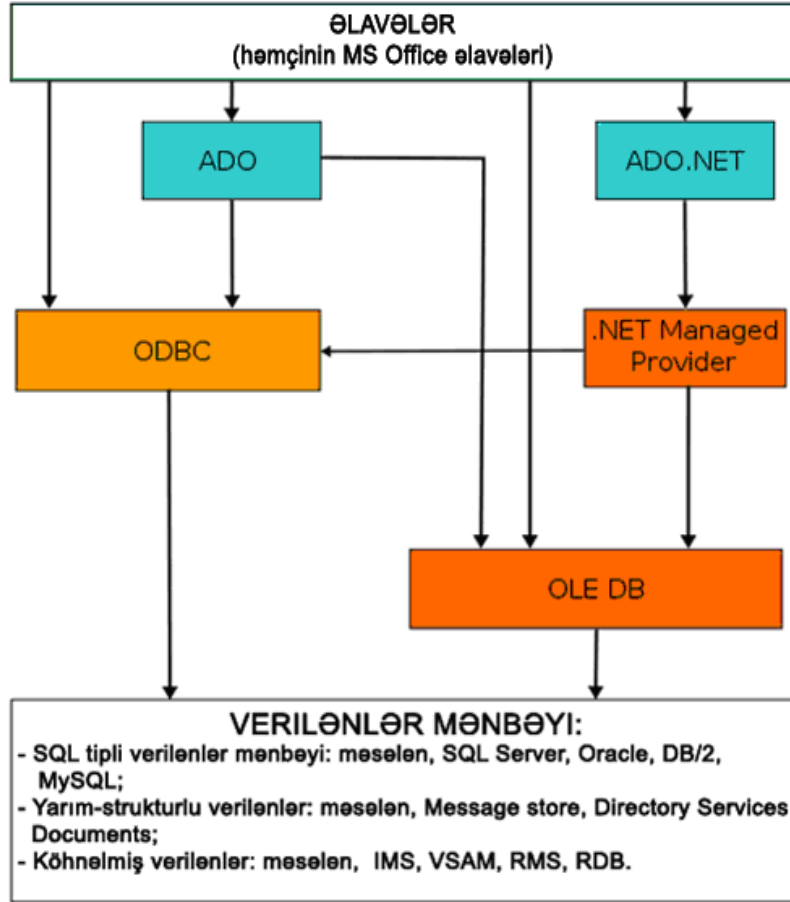
- hökmən **.NET Framework** qurulmasını tələb edir;
- **Visual Basic** redaktorundan adi vasitələr ilə **ADO.NET**-lə işləmək olmur (hökmən **Visual Studio** yüklənməlidir);
- resurs tələbləri yüksəkdir.

ADO.NET-ə bu kitabda baxılmayacaq (kitabın əsas məqsədlərdən kənar olduğuna görə).

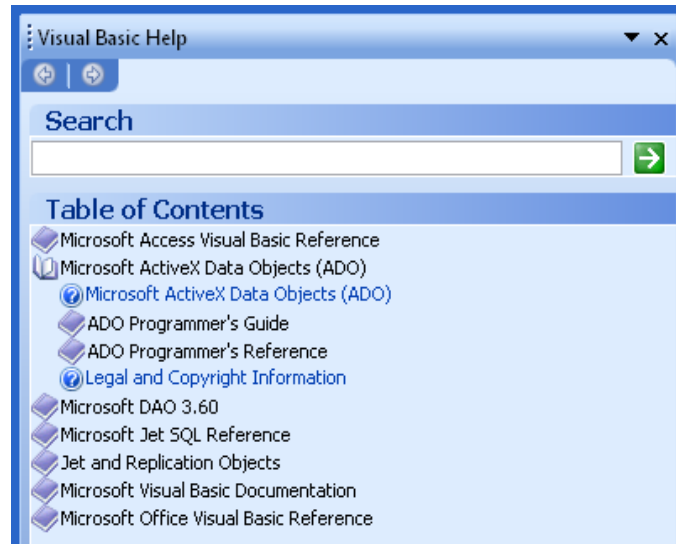
ADO müxtəlif drayverlərlə işləməyi bacarır: məsələn, OLE DB və ODBC drayverləri ilə. ADO əslində COM (Component Object Model, yəni azərbaycanca komponentlərin obyekt modeli) texnologiyası əsasında qurulub. COM - Microsoft-un texnoloji standartıdır. COM eyni zamanda bir neçə proqramda birgə istifadə edilən və bir-biri ilə qarşılıqlı əlaqəsi olan komponentlər əsasında proqram təminatlarının yaradılması üçün nəzərdə tutulmuşdu. Məhz buna görə ADO obyektleri bütün COM uyğunluqlu proqramlaşdırma dillərində istifadə edilə bilər: Visual C++, Visual Basic, Delphi, VBA, VBScript, JScript, ActivePerl, Python v.s.).

Proqram obyektlərinin özü drayverlər komplektində verilir və verilənlər bazaları ilə bağlanmaq üçün nəzərdə tutulub. Bu komponentlər MDAC (Microsoft Data Access Components) adı ilə tanınır. Onlar Windows 2000, Windows XP, Windows 2003, Windows Vista və, daha müasir əməliyyat sistemləri ilə idarə edilən, hər bir müasir kompyuterdə var. İstifadəçilər MDAC-in ən yeni versiyalarını Microsoft Web-saytından tamamilə pulsuz yükləyə bilər. Aşağıdakı şəkl. 9.1-də ADO-nun MS Office proqramlarının müxtəlif verilənlər mənbəyi ilə olan əlaqələri göstərilib.

ADO haqqında ətraflı məlumatı daha asan Microsoft Access proqramında almaq olar. Bunun üçün Microsoft Access-də yeni boş verilənlər bazası yaradılmalıdır və VBA redaktoru koduna keçmək lazımdır (klaviaturanın **<Alt>+<F11>** düymələr kombinasiyası ilə) və sonra **<F1>** düyməsi basılmalıdır. Nəticədə şəkl. 9.2-də göstərilən sorğu sənədi əmələ gələcək. Burada yuxarıdan ikinci sətir – həmin mövzuya aiddir: **ADO (ActiveX Data Objects)**. **ADO Programmer's Guide** – VBA proqramçısı üçün ADO obyektleri ilə edilən əməllər haqqında məlumat verilir. **ADO Programmer's Reference**-də ADO obyektleri, onların xassələri, metodları və hadisələri haqqında məlumat verilir.



Şəkil 9.1 Verilənlərin mənbəyi (Office proqramları) və ADO-nun digər proqram obyektləri sistemləri ilə olan əlaqələrini göstərən sxem.



Şəkil 9.2 İstifadəçinin praktiki işində vacib olan Microsoft Access-in VBA redaktorunda ADO haqqında sorğu sənədlərinin çıxarılması.

Öz-özlüyündə ADO obyektı çox sadədir və anlanması da olduqca asandır. Onun tərkibində cəmi 3 obyekt var:

- **Connection** obyektı – verilənlər mənbəyi ilə bağlantı yaradılmasını və onun idarə edilməsini təmin edir. Bağlantı işi zamanı yaranan bütün səhvlər həmişə yanaşı olan **Errors** kolleksiyasına yerləşdirilir;
- **Command** obyektı – verilənlər mənbəyində müəyyən əməliyyatı yerinə yetirmək üçün əmri təmin edir (sorgunun icrası, proseduranın icrası, obyektin yaradılması və ya verilənlərin dəyişdirilməsi v.s.). əgər verilənlər mənbəyi SQL uyğunudursa, onda **Command**, böyük ehtimalla SQL-i əmrlə təmin edəcək. **Command** obyektı ilə **Parameters** kolleksiyası həmişə yanaşıdır (bu kolleksiyanın parametrləri sorguya və ya saxlanılan proseduraya ötürülür);
- **Recordset** obyektı – verilənlər mənbəyindən alınan və ya başqa üsulla generasiya edilmiş yazılar yığımını təqdim edir. **Fields** kolleksiyası onunla yanaşı olur (həmin yazılar yığımında olan verilənlərin özünü və yazılarda olan sütunlar haqqında məlumatı təqdim edir: məsələn, verilənlərin adını, tipini, ölçüsünü v.s.)

Hər üç obyekt üçün **Properties** kolleksiyası nəzərdə tutulub (bu kolleksiya bağlantının xassəsini, əmrləri və yazılar yığımını təyin edir).

Bütün obyektləri aşkar olaraq yaratmaq məcburi deyil – məsələn, **Recordset** obyektı yaradıldıqda, avtomatik rejimdə **Connection** obyektı yaradıla bilər.

ADO obyektı yaradılmasından əvvəl istifadəçi gərək özü layihəsində lazımı kitabxanaya istinad yaratsın. Bunun üçün **Tools** menyusunda **References** seçilməlidir və **Microsoft ActiveX Data Objects** sətrinin lazımı nömrəsi qarşısında bayraqcıq qurulmalıdır (tərtib edilən əlavə müştəri kompüterlərindəki kitabxanaların versiyasından asılıdır).

Aktiv işləyən oxucuya məsləhət görürük ki, MDAC-ın daha stabil sayılan versiyasından (Microsoft tərəfindən 2005-ci ildə istehsal olan versiyasını) istifadə etsin, çünki bu versiyanın üstün olan cəhətləri var. Ən əsası budur ki, istifadəçi öz VBA proqramını heç bir dəyişdirməzsiz bir kompüterdən o birisinə köçürə bilər (heç bir yeni əlavə obyektin əlavə edilib yüklənməsinə ehtiyac yoxdur). Praktiki işlərdə hətta daha erkən olan versiyalardan istifadə etməyə məsləhət görürlər, çünki onlar Windows-un bütün versiyalarında qurulmuş olur (Windows 2000-dən başlayaraq yuxarı). Müxtəlif versiyaların funksionallığı bir birindən az fərqlənir, ancaq ilkin versiyalar daha stabil işləyir və etibarlı sayılır.

9.3 Connection obyektı və Errors kolleksiyası

ADO Obyektı. VBA-da Connection obyektı, ConnectionString xassəsi, qoşulma sətrinin generasiyası, verilənlər bazası ilə bağlantının açılması və qoşulması, ADOError obyektı və Errors kolleksiyası.

Connection obyektinin yaradılması olduqca asandır. Məsələn, SQL Server serverində LONDON adlı Northwind verilənlər bazasına qoşulmaq (bağlanmaq) üçün Aşağıdakı VBA kodundan istifadə etmək olar:

```

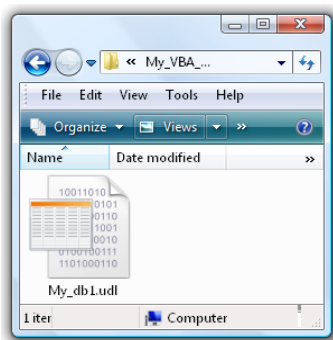
Dim cn As New ADODB.Connection
cn.ConnectionString="Provider=SQLOLEDB.1;Integrated_
Security=SSPI;" & "Initial Catalog=Northwind; Data Source=LONDON"
cn.Open

```

Prinsip etibarı ilə verilənlər bazasına bağlanmaq (qoşulmaq) və bağlantı obyektinin yaradılması üçün yuxarıdakı VBA kodu tamamilə kifayətdir. İstifadəçidə belə bir sual əmələ gələ bilər: **ConnectionString** xassəsində nə yazılıb və bu xassənin qiymətini özü də (istifadəçi kimi) yazı bilərmi?

Ən sadə həll budur: istifadəçinin özü heç bir əlavə yazmasın. O biri tərəfdən bu xassənin qiymətini avtomatik rejimdə də generasiya etmək olar. Həmin nəticəni reallaşdırmaq üçün aşağıdakı sadə əməllər addım-addım yerinə yetirilməlidir:

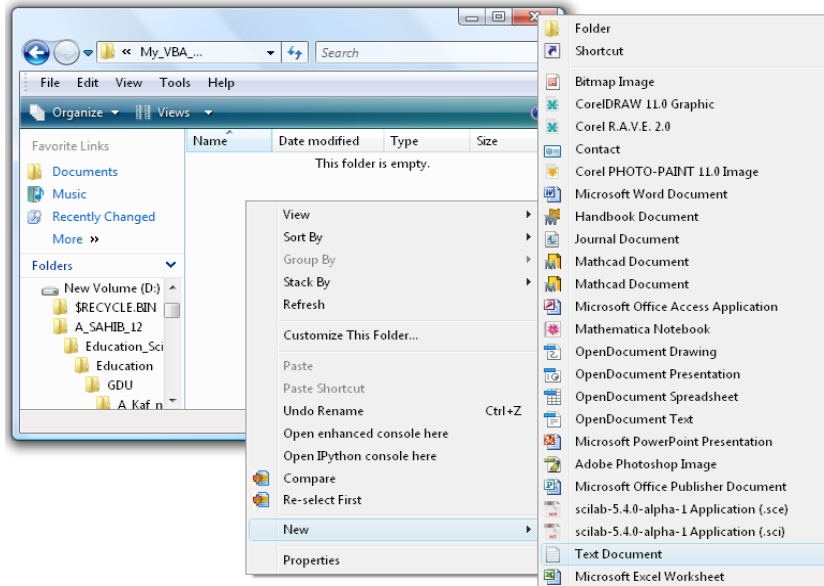
- Yaranmış faylın adını dəyişmək lazımdır: genişlənməsi UDL (ingiliscə User Data Link) olmalıdır. Adı dəyişdikdə baxmaq lazımdır ki, faylın piktoqramı da dəyişibmi (bax şəkl. 9.3)?



Şəkil 9.3 My_db1 adlı yeni yaradılmış TextDocument faylının genişlənməsini dəyişdirərək UDL genişlənməsinin seçilməsi.

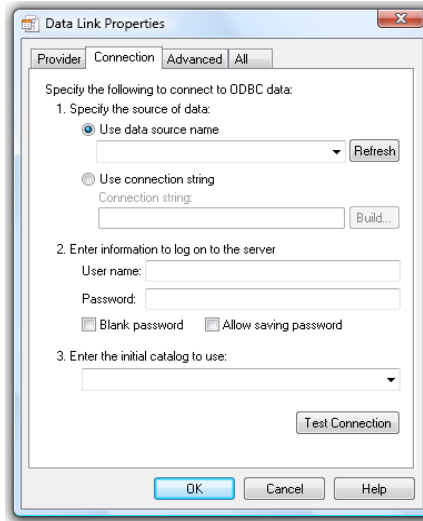
Əgər qovluda həmin faylın görünən piktoqramı dəyişmədisə (yeni əvvəlki TextDocument faylının piktoqramıdır), onda bu hal ona işarə edir ki, faylın real genişlənməsi əvvəlki kimi **.txt** formasındadır (dəyişdirilmiş **.udl** genişlənməsi deyil). Bu halda **Windows Explorer** pəncərəsində **Tools** seçilərək, yaranan menyudan **FolderOptions** (qovluqların xassəsi) seçilməlidir və açılan qurmada, **View** formasına keçərək, orada **Hide file extensions for known file types** (Azərbaycanca – məlum olan fayl tiplərinin genişlənməsini gizlətmək) sətrində bayraqcıq götürülməlidir. Bunu etdikdən sonra yenidən faylın adı (bizim nümunədə fayla My_db1 adını vermişik) dəyişdirilərək, **.udl** genişlənməsi seçilməlidir, bax şəkl. 9.3.

- Qovluqda yeni boş olan istənilən proqrama aid fayl yaradılmalıdır (məsələn, mətn faylı olan ən sadə **TextDocument** faylı), şəkl. 9.4.



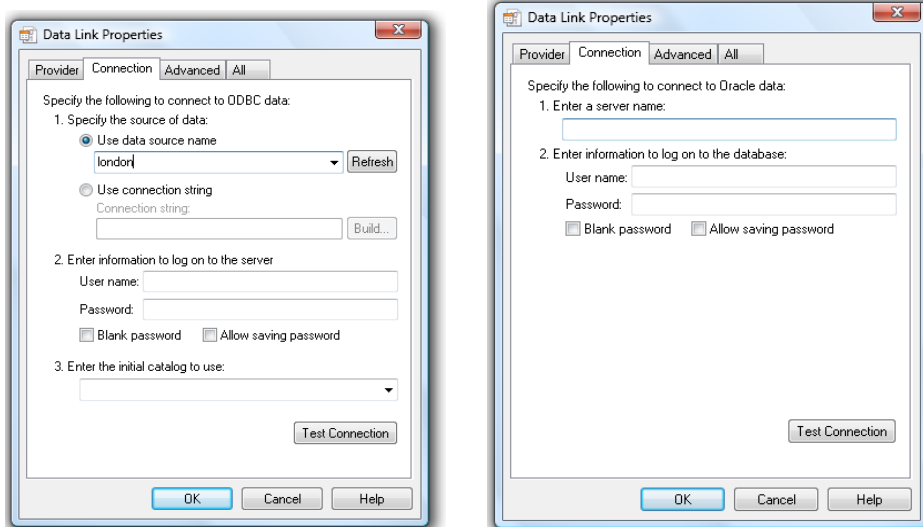
Şəkil 9.4 Qovluqda yeni boş TextDocument faylının yaradılması.

- UDL yaradılan kimi sadəcə onun piktoqramının üstündən iki dəfə mausun sol düyməsi ilə şıqqıldatmaq lazımdır. Dərhal qurmalar strukturlu **Data Link Properties** (Azərbaycanca - verilənlər bağlantısının xassələri) pəncərəsi açılacaq, şək. 9.5.



Şəkil 9.5 Yaradılmış UDL genişlənməsi olan faylın qurmalar strukturlu **Data Link Properties** pəncərəsi.

- Həmin açılmış **Data Link Properties** pəncərəsinin **Provider** adlı birinci əlavə qurmasında verilənlər bazasının lazım olan tipi seçilməlidir (məsələn, **Microsoft OLE DB Provider for SQL Server**). **Oracle** verilənlər bazasına bağlanmaq üçün həmin əlavə qurmada **Microsoft OLE DB Provider for Oracle** seçilməlidir. Access verilənlər bazasına bağlanmaq üçün **Microsoft JET 4.0 OLE DB Provider** seçilməlidir. Excel səhifəsinə bağlanmaq haqqında ayrıca və daha ətraflı müvafiq hissədə danışacağıq.
- Sonra **Connection** əlavə qurmaya keçmək lazımdır. Hər tip verilənlər bazasının həmin əlavə qurmasında fərqli görkəmi var: məsələn, SQL Server üçün onun görüntüsü şək. 9.6.a-dəki kimidir, Oracle üçün isə şək. 9.6.b-dəki kimidir.



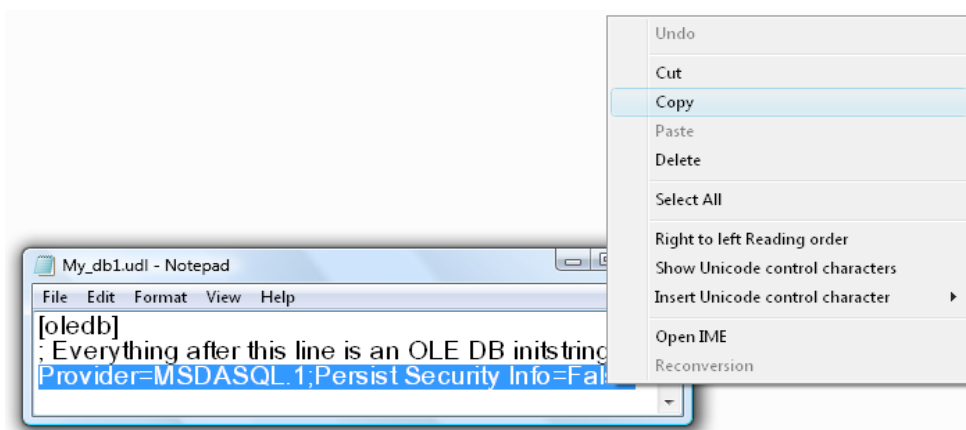
a)

b)

Şəkil 9.6 a) SQL Server bağlanmaq üçün xassələr pəncərəsi və b) Oracle verilənlər bazası ilə bağlanmaq üçün xassələr pəncərəsi.

Əgər istifadəçi bu pəncərələrdə hansı parametrlərin verilməsini bilmirsə, onda lazımı məlumatı verilənlər bazasının administratorundan soruşmaq olar.

- Bütün parametrlər daxil ediləndən sonra, **Test Connection** düyməsinin basılmalıdır (verilənlər bazasına bağlanmanı yoxlamaq üçün). Test müvəffəqiyyətlə qurtaran kimi, pəncərənin **OK** düyməsinə basaraq, pəncərənin qapanmasına nail olmaq lazımdır.
- Tamamlayıcı mərhələ kimi, mausun sağ düyməsi ilə yaradılmış faylın piktoqramı üzərində bir dəfə şıqqıldadaraq kontekst menyusunu açmaq lazımdır. Açılan kontekst menyusundan **Open With** (köməyilə açmaq) → **Choose Program** (proqramı seçməli) seçərək, yenidən açılan fayllar tipləri siyahısının pəncərəsində **Notepad** (bloknöt) seçilməlidir və **OK** düyməsi basılmalıdır.
- Yaradılmış fayl bloknötə açılacaq, orada **Format** menyusunda **Word wrap** (sətirlərdən keçid) sətirində bayraqçıyı götürmək lazımdır və bu faylın son sətirini dəyişmə buferinə (maşının operativ yaddaşına) kopyalamaq lazımdır, şəkl. 9.7.



Şəkil 9.7 Avtomatik rejimdə generasiya edilmiş bağlanma sətirinin maşının operativ yaddaşına yazılması.

DIQQƏT!

Əgər həmin sətirdə Azərbaycan hərfləri varsa (məsələn, kataloqun Azərbaycanca olan adı *Excel faylının yolunda yazılıb*), onda məsləhət görülür ki, maşının operativ yaddaşına kopyalama əməliyyatı klaviaturanın Azərbaycan rejimində olan halda yerinə yetirilsin, digər halda Azərbaycan hərfləri sual işarələri ilə əvəz olacaq.

- Nəhayət, sonuncu mərhələdə, maşının operativ yaddaşında olan qiyməti VBA kod redaktoruna **ConnectionString** xassəsinin qiyməti kimi yazaraq (**Paste** əməliyyatı ilə), həmin qiymətin hər iki tərəfdən dırnaq arasına işarəsi ilə haşiyələmək lazımdır. Əlbəttə, istifadəçi **connection string** sözünü əllə yığaraq, Azərbaycanca da yazıb bilər. Sadəcə bu söz bir-biri ilə nöqtə vergül işarəsi (;) ilə ayrılan parametrlər yığımıdır: "xassə=qiymət" (bağlantı sətirindəki qiymət üçün dırnaq arasına işarəsi istifadə edilmir).

İndi isə hər vacib olan parametrlərin mənasını izah edək:

- **Provider** – verilənlər mənbəyinə bağlanmaq üçün drayverdir. Şək. 9.6-da göstərilən kimi, hər verilənlər mənbəyinin tipindən asılı olaraq, pəncərəsinin görüntüsü (müvafiq olaraq daxil edilən parametrlər də) fərqlidir. Məsələn, şək. 9.6.a-da biz **OLE DB** drayverindən istifadə etmişik (nəzərə alınmalıdır ki, **ODBC** drayverlərinin yavaş işləməsinə baxmayaraq, məhz onları Excel səhifəsindəki cədvələ bağlanma yaratmaq üçün istifadə edirlər). Bağlanma sətirində **Provider** üçün qiymətin göstərilməsi məcburi deyil. Əgər belədirsə, onda gərək onun qiyməti **Connection** obyektinin qiyməti kimi ayrıca təyin edilməlidir.
- **IntegratedSecurity** – baxdığımız halda bu xassə istifadə edilir, çünki biz SQL Server ilə bağlantı yaratmaq üçün Windows autentifikasiyasını istifadə edirik. Əgər biz SQL Server autentifikasiyasını istifadə etsə idik, onda bu xassə bizə lazım olmayacaq idi. Əslində biz onun əvəzinə gərək iki başqa xassədən istifadə etməli idik: **User ID** (istifadəçi identifikatoru) və **Password** (parol). Buna görə sadə istifadəçi (yeni başlayan VBA proqramçısı) daha sadə üsuldən istifadə etsə sərfəli olar: *sadəcə verilənlər bazası administratorundan lazımı qiymətləri öyrənmək*.
- **DataSource** – verilənlər mənbəyinin adı. Bizim baxdığımız hal üçün bu ad - SQL Server əməliyyat apardığı kompyuterin adıdır. Başqa hallarda bu ad, məsələn, Oracle nümunəsi və ya MS Access ola bilərdi – hər şey hansı verilənlər bazasına qoşulmaqdan asılıdır.
- **InitialCatalog** – həmin serverdə verilənlər bazasının adını təyin edir (baxdığımız nümunədə bu ad - **Northwind**).

Nəzərə alınmalıdır ki, verilənlər bazasına qoşulanda, bağlantı sətirini UDL faylı ilə avtomatik rejimdə generasiya etdikdə, birinci dəfə sərf olan vaxt uzun ola bilər. Əslində 1 dəqiqədən az

vaxt gedir. Bundan əlavə istifadəçi sığortalanır ki, bağlantı sətrində səhvlər olmayacaq (**Test Connection** düyməsi ilə UDL faylından verilənlər bazası ilə bağlanmanı test etməyə imkan yaranır). Faktiki olaraq, verilənlər bazası ilə bağlanmaq üçün hər nə lazımdırsa edildi: **Connection** obyektı yarandı, bu obyekt üçün **ConnectionString** xassəsi sazlandı və **Open()** metodu çağırıldı. Buna baxmayaraq obyektin metodları və xassələrini bir qədər də mükəmməl bilmək lazımdır - buna görə aşağıda həmin məlumat ətraflı verilir (çünki bir çoxları ingilis dilini yaxşı bilmir və bu haqda xüsusi ədəbiyyat mövcud deyil):

- **Provider** xassəsi, yuxarıda qeyd edilən kimi, verilənlər bazasına bağlanmaq üçün istifadə edilən drayverin təyin edilməsini təmin edir. Və biz həmin draveyri **ConnectionString** qiymətinin içərisində təyin etdik. Başqa əlverişli üsullar da var: eyni nəticəyə ayrıca xassədən də istifadə etmək olar. Müxtəlif verilənlər mənbəyinə bağlanmaq üçün **Provider** xassələrinin qiymətləri bu cür yazıla bilər:

- "Microsoft.Jet.OLEDB.4.0" – Access, Excel faylları və Jet əsaslı başqa verilənlər mənbəyinə qoşulmaq üçün;
- "SQLOLEDB" - SQL Server-ə qoşulmaq üçün (nümunədəki kimi);
- "MSDAORA.1" - Oracle serverinə qoşulmaq üçün;
- "ADsDSOObject" - Windows kataloq xidməti verilənlər bazasına qoşulmaq üçün.

- **ConnectionString** – xassəsi **Connection** obyektinin baş xassəsidir. Bu xassə mənbəyə qoşulmanın parametrlərini təyin edir. Yuxarıda oxucu onunla necə işləməyi əyani olaraq gördü.

- **Open()** metodu verilənlər bazası ilə bağlantını açmağa imkan yaradır. Qeyd etməliyə ki, qoşulma sətirini obyektin xassəsi kimi ayrıca sazlamaq da olar (**Connection** obyektinin xassəsini ayrıca sazlayan kimi). Bunun üçün qoşulma sətirini metoda parametr kimi ötürmək olar.

- **Close()** metodu bağlantının qapanmasına imkan yaradır. Nəzərə alınmalıdır ki, bağlanma obyektı bu halda operativ yaddaşdan silinmir. Bu obyektı tamam silmək üçün aşağıdakı VBA kodu istifadə edilə bilər:

```
cn.Close  
Set cn=Nothing
```

və ya daha sadə kodla

```
Set cn=Nothing
```

bu kodla bağlantının pozulması avtomatik olacaq.

Bilmək lazımdır ki, həmin obyekt üçün Microsoft daha çox üsul və metod yaradıb (əlavə məlumat üçün bu mövzu ilə bağlı Microsoft-un xüsusi "online" sənədləri və kurslarına da müraciət etmək olar).

Daha ətraflı öyrənilməsi vacib olan xassə: **Errors** xassəsidir (**Error** obyektləri kolleksiyasını (səhvləri) qaytarır). Verilənlər bazası ilə bağlantı yaratdıqda səhvlərin sayı və çeşidi həqiqətən də çox ola bilər: parol və ya ad düzgün verilməyib, istifadəçinin verilənlər bazasına bağlanmaq üçün kifayət qədər haqqı yoxdur, kompyutərə şəbəkə ilə istinad etmək olmur v.s. Buna görə təkidlə məsləhət görülür ki, VBA proqram kodunda səhvləri emal edən proseduralar əlavə edilsin. Baxdığımız nümunəyə uyğun olan səhvləri emal edən VBA prosedurasının ən sadə variantı aşağıda verilib:

```

Dim cn As ADODB.Connection
Set cn=CreateObject("ADODB.Connection")
cn.Provider="SQLOLEDB"
cn.ConnectionString="User ID=SA; Password=password;_
Data Source=LONDON1; & "Initial Catalog=Northwind"
On Error GoTo CnErrorHandler
cn.Open
Exit Sub
CnErrorHandler:
For Each ADOErr In cn.Errors
    Debug.Print ADOErr.Number
    Debug.Print ADOErr.Description
Next
End Sub

```

Praktiki işlərdə hansısa bağlanma üçün səhvlər adətən tutulur: məsələn, Access faylı yoxdur, SQL Server-ə qoşulduqda istifadəçinin adı və ya parolunda səhv var, qoşulmaq üçün istifadəçinin kifayət qədər haqqı yoxdur, fayl fəvqəladə rejimdə açılıb v.s. Nəticədə istifadəçiyə səhvləri düzəltmək üçün təkliflər verilir.

Praktiki işlərdə **ADOError** obyektinin ən vacib olan xassələrini qeyd edək:

- **Description** – səhvin izahlı təsviri. Adətən ən vacib olan məlumat bu xassənin köməyi ilə yaradılır;
- **Number** – səhvin nömrəsi. Nömrəyə görə biliklər bazasında və İnternetdə axtarış aparılması xeyli asanlaşır.
- **Source** – səhvin mənbəyi. Bu məlumat yalnız o vaxt xeyirli olur ki, **Errors** kolleksiyasında müxtəlif mənbələrdən gələn səhvlər var;
- **SQLState** və **NativeError** – SQL-uyğunluğu olan verilənlər mənbəyindən gələn səhv haqqında məlumat.

9.4 Excel səhifəsindəki cədvələ qoşulma

ADO vasitələri ilə Excel cədvəli ilə bağlantı yaradılması, adlandırılmış diapazonun yaradılması, ODBC verilənlər mənbəyinin yaradılması

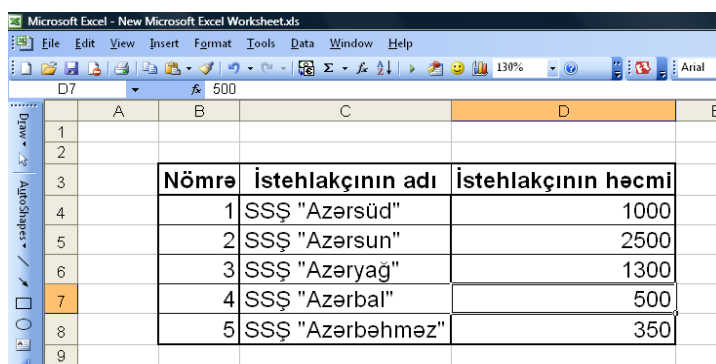
Praktiki işlərdə daha çox belə hal yaranır: verilənlər bazası kimi Excel cədvəlinə qoşulmaq (bağlantı yaratmaq) ehtiyacı var. Əlbəttə, Excel-in obyekt modelləri vasitələri ilə də bu problemi

həll etmək olar (bax 11-ci fəsilə "Excel-də proqramlaşdırma"), ancaq ADO obyektlərinin istifadəsi daha çox üstünlüklər yaradır:

- yazıların tapılması, yeni yazıların cədvələ yerləşdirilməsi, olan yazıların redaktə edilməsi daha asandır. ADO obyektləri Microsoft tərəfindən əzəldən bu məqsədlərlə yaradılmışdır.
- Excel obyekt modelini yalnız Excel-də istifadə etmək olar, ADO obyektləri isə universaldır (istənilən verilənlər mənbəyinə qoşulmaqda istifadə etmək mümkündür). ADO obyektləri ilə eyni proqramı Excel verilənləri ilə birgə "böyük" verilənlər bazası ilə işləmək olur (məsələn, SQL Server və ya Oracle ilə).

Excel səhifəsindəki cədvələ qoşulmaq bir o qədər də çətin iş deyil. Lakin burada da müəyyən əməliyyatlar ardıcılığına tabe olmaq lazımdır. Buna görə aşağıda əməllərin addım-addım yerinə yetirilməsinin qaydası verilib:

- Tutaq ki, istifadəçinin, məsələn, hansısa ad ilə Excel kitabı var və bu fayl kompyuterin D diskinin əsas kataloqunda yerləşib. Fərz edək ki, həmin Excel kitabının birinci səhifəsində, şəkl. 9.8-də göstərilən kimi, tamam ilə sadə strukturlu bir cədvəl var.

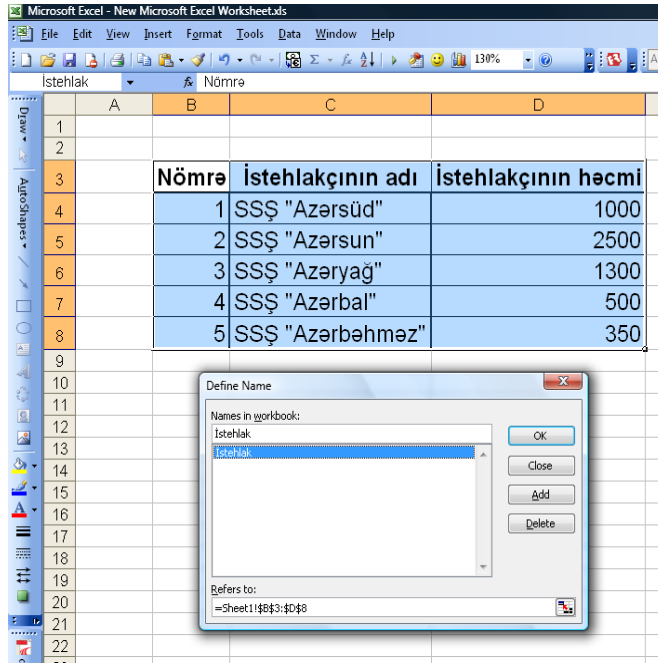


Nömrə	İstehlakçının adı	İstehlakçının həcmi
1	SŞŞ "Azərsüd"	1000
2	SŞŞ "Azərsun"	2500
3	SŞŞ "Azəryağ"	1300
4	SŞŞ "Azərbal"	500
5	SŞŞ "Azərbəhməz"	350

Şəkil 9.8 ADO obyektləri vasitələri ilə istinad ediləsi Excel cədvəli.

Bizim qarşımızda duran məsələ budur: həmin cədvəllə verilənlər bazasına qoşulan kimi, necə qoşulmaq olar? Bunun üçün nə etmək lazımdır?

- **Birinci etap** - hazırlıq işləri. Bəzən bu mərhələsiz də keçinmək olar - əgər Excel səhifəsi bütöv bir cədvəldirsə. Praktikada isə bu cür alınır ki, bir Excel səhifəsində bir neçə cədvəl yerləşdirilir (yaxud bir səhifədə cədvəl şərhilərlə verilir və ya cədvəlin aşağı hissəsində, məsələn, cədvəlin aşağısında yekun hesablamalar var v.s.). Excel-də mürəkkəblik yaratmamaq üçün daha yaxşı olar ki, Excel-də həmin cədvəl aşkar formada nişanlansın. Bunu etmək çox asandır: cədvəli bütövlükdə seçmək (şəkl. 9.8-ə görə - B3-dən başlayaraq D8-ci hücrəyə qədər olan diapazon seçilməlidir) və seçilmiş cədvəl diapazonuna ad təyin etmək lazımdır. Bunun üçün Excel-də **Insert** menyusunda **Name**→**Define** təlimatını seçərək diapazonun adı, açılan **Define Name** pəncərəsində nəzərdə tutulmuş ad verilməlidir. Bizim halda (şəkl. 9.9) diapazonun adı **İstehlak** kimi verildi.

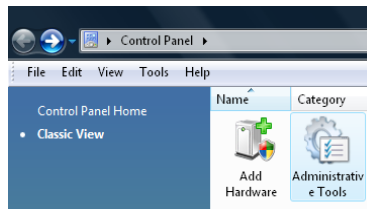


Şəkil 9.9 Seçilmiş Excel cədvəl diapazonuna ad verilməsi.

Diqqət verin ki, diapazon seçildikdə cədvəlın sütunlarının adları da seçilməlidir (bizim halda sütunların adı: **Nömrə**, **İstehlakçının adı** və **İstehlakçının həcmi**).

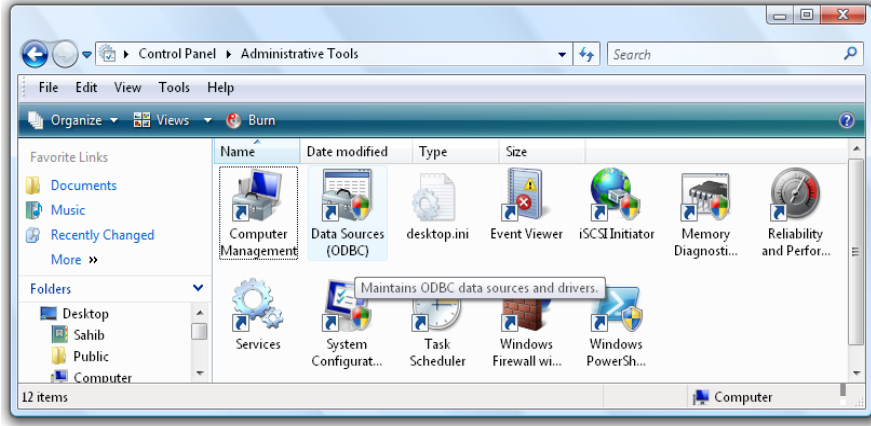
Seçilmiş diapazona ad verilib qurtardıqda, Excel kitabını artıq bağlamaq olar (çünki işin bu mərhələsində o, bizə daha lazım deyil).

- **İkinci etap** – daha bir hazırlıq işlərinə aid mərhələ (ODBC tipli verilənlər mənbəyinin yaradılması). Plan ilə, əslində, *.UDL tipli fayl yaradıla bilər və Excel faylına **D:\.....\Excel_DB1.xls**. qoşulmanın (bağlantının) sazlanmasını aparmaq olardı. Məlumdur ki, UDL faylından bir başa yalnız OLE DB drayveri ilə işləmək olar. Təəssüf ki, lazımı drayver yoxdur (**Microsoft JET 4.0 OLE DB Provider** yalnız **MDB** faylları ilə işləyə bilər). Buna görə növbəti etapda (addımda) ODBC tipli verilənlər mənbəyini yaradıq, çünki qoşulmaq üçün ODBC drayveri Excel-də var. **Control panel**-ində (İdarə etmə panelində) **Administrative Tools** (administrasiya alətləri) piktoqramı üzərində mausun sol düyməsi ilə iki dəfə şıqqıldadıq, şəkl. 9.10.



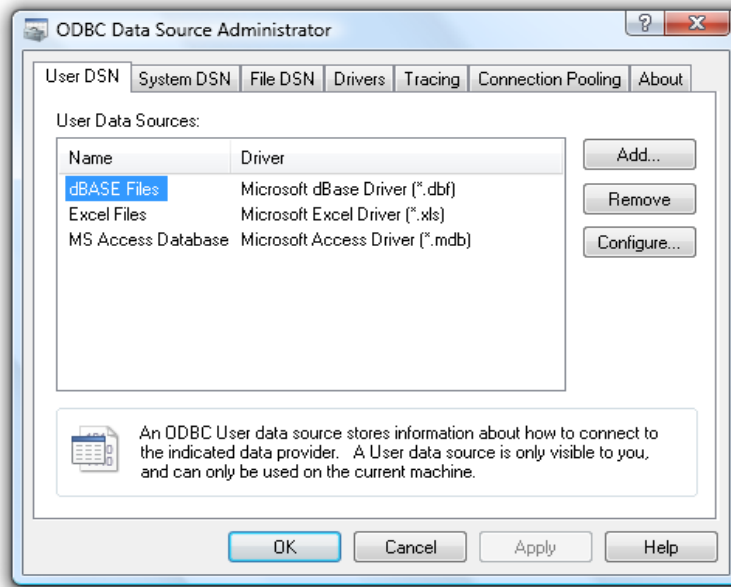
Şəkil 9.10 Control panel-ində Administrative Tools-un aktivləşdirilməsi.

Dərhal **Control panel**-ində **Administrative Tools** tərkibində olan vasitələrin piktoqramları görünür, həmçinin bizə lazım olan **Data Sources (ODBC)** (ODBC verilənlər mənbələri piktoqramı. Bu piktoqramın üzərində mausun sol düyməsi ilə iki dəfə şıqqıldadıq (aktivləşdirilməsi lazımdır), şəkl. 9.11.



Şəkil 9.11 Control panel-ində Data Sources (ODBC) vasitəsinin aktivləşdirilməsi.

Data Sources (ODBC verilənlər mənbələri) piktoqramı aktivləşdirildikdə dərhal bizə lazım olan **ODBC Data Sources Administrator** dialoq pəncərəsi açılır, şəkl. 9.12.

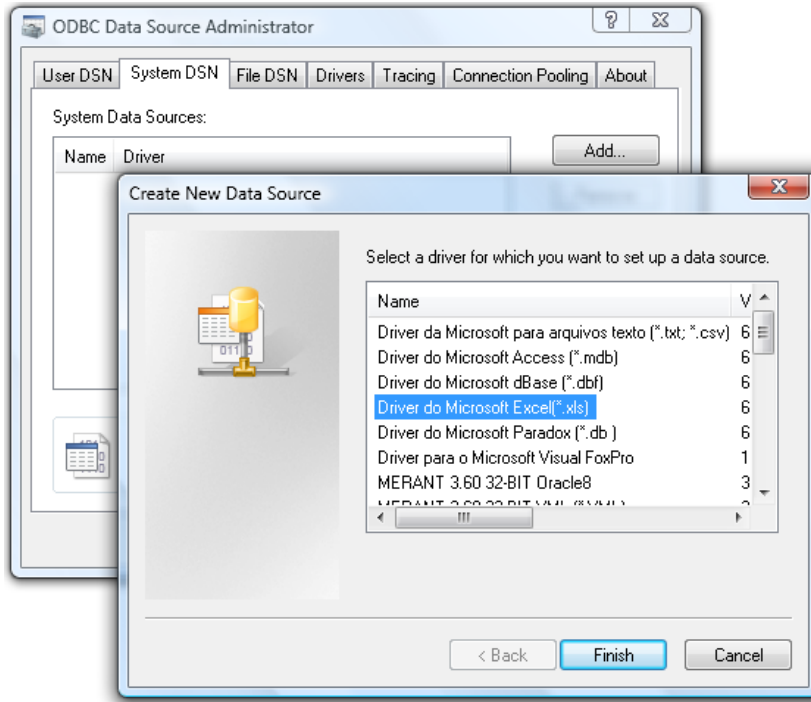


Şəkil 9.12 ODBC Data Sources Administrator vasitəsinin idarəetmə dialoq pəncərəsi.

Bizim sərəncamımızda üç tip DSN (**Data Source Name**, yəni ODBC verilənlər mənbəyi) vardır:

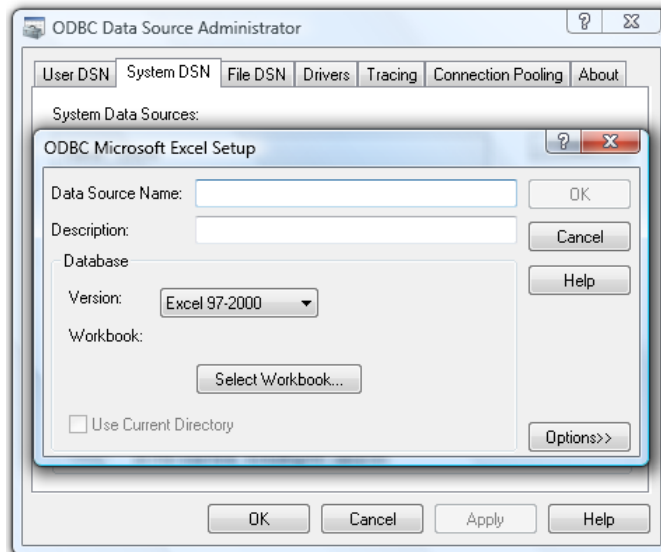
1. **User DSN** – bu verilənlər mənbəyi haqqında olan məlumat reyestrin istifadəçi üçün məxsusi olan hissəsində saxlanılır. Buna görə həmin verilənlər mənbələri yalnız onları yaradan istifadəçi üçün əlçatandır;
2. **System DSN** - bu verilənlər mənbəyi haqqında olan məlumat reyestrin ümumi hissəsində saxlanılır və həmin kompyuterin bütün istifadəçiləri üçün əlçatandır.
3. **File DSN** - bu verilənlər mənbəyi haqqında olan məlumat fayl sisteminin faylına yazılır.

Adətən **System DSN** daha çox istifadə edilir, buna görə **ODBC Data Sources Administrator** vasitəsinin idarəetmə dialoq pəncərəsinin **System DSN** əlavə qurmasını açırıq və **Add** (əlavə et) düyməsini basırıq, şəkl. 9.13.



Şəkil 9.13 **ODBC Data Sources Administrator** vasitəsinin idarəetmə dialoq pəncərəsindəki **System DSN** əlavə qurmasını açaraq **Add** düyməsini basılması.

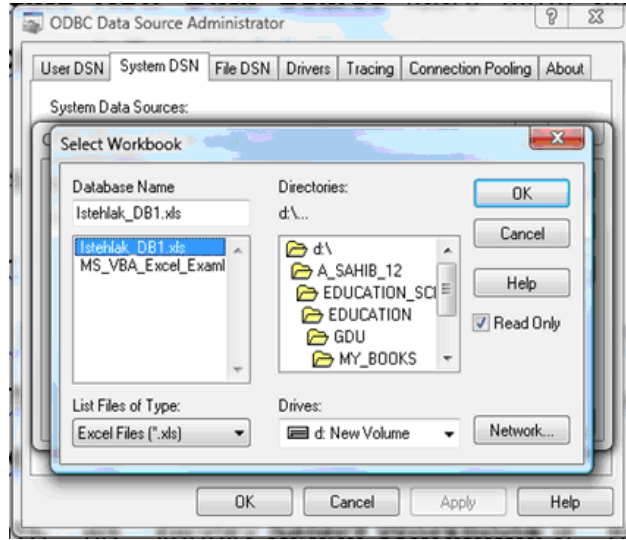
Açılan yeni **Create New Data Source** idarəetmə panelində bizə lazım olan **Driver do Microsoft Excel(*.xls)** drayveri seçirik və həmin pəncərənin **Finish** (tamamlanma) düyməsini basırıq. Bununla verilənlər mənbəyinin yaradılması prosedurası hələ bitmir, çünki yeni **ODBC Microsoft Excel Setup** idarəetmə pəncərəsi əmələ gəlir, şəkl. 9.14.



Şəkil 9.14 **ODBC Microsoft Excel Setup** idarəetmə pəncərəsi.

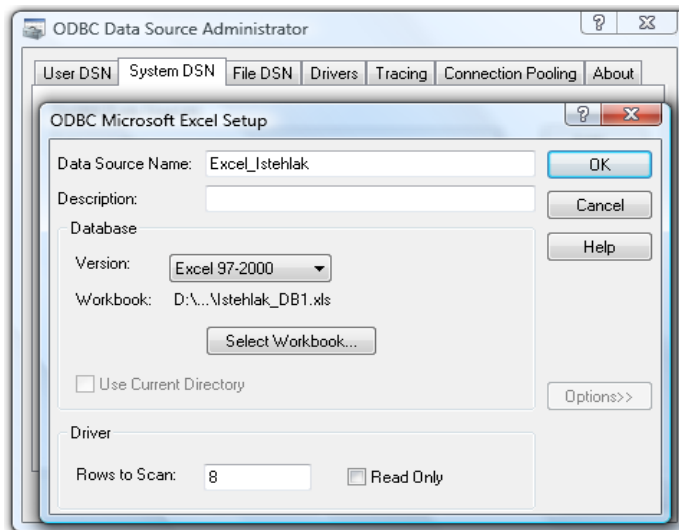
■ **Üçüncü etap** – tamamlayıcı mərhələ:

- **Data Source Name** sahəsində, şək. 9.14, verilənlər mənbəyinin adını yazırıq (əsas budur ki, istifadəçi özü həmin adı yaddan çıxarmasın), məsələn: **Excel_İstehlak**;
- **Select Workbook** düyməsinə basaraq, yeni yaranan **Select Workbook** pəncərəsinin köməyi ilə lazım olan Excel kitabını seçirik (şək. 9.15), bizim nümunədə: **D:\Education\GDU\My_Books\.....\İstehlak_DB1.xls**;



Şəkil 9.15 **Select Workbook** pəncərəsinin köməyi ilə lazım olan Excel kitabını seçilməsi.

- Yenidən **ODBC Microsoft Excel Setup** idarəetmə pəncərəsinə qayıdaraq, **Options** düyməsinə basırıq. Pəncərənin yenidən dəyişməsi sayəsində seçim edirik: əgər yaradılan proqramdan cədvəl dəyişdiriləcəksə, onda **Read Only** sətirindəki təlimatdan bayraqçıyı çıxardırıq. Son nəticədə idarəetmə pəncərəsi şək. 9.16-da olan kimi görəcəyik;



Şəkil. 9.16 ODBC mənbəyinin Excel faylına qoşulması üçün **ODBC Microsoft Excel Setup** idarəetmə pəncərəsinin sazlanma əməllərinin tamamlanması.

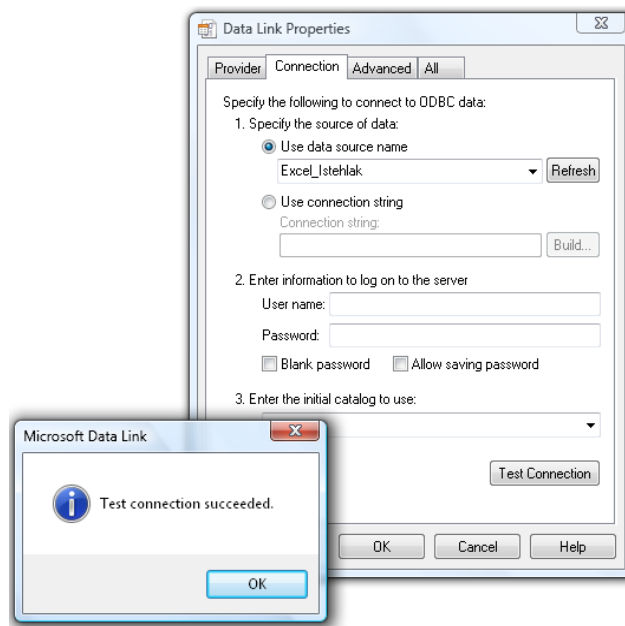
- Nəhayət birçə əməl qaldı - **ODBC Microsoft Excel Setup** idarəetmə pəncərəsində **OK** düyməsi basılmalıdır və bu pəncərə yox olan kimi, hələ ekranda qalan **ODBC Data Sources Administrator** vasitəsinin idarəetmə dialoq pəncərəsindəki **OK** düyməsi də basılır və bu pəncərəni də bağlayırıq. Bununla bütün əməliyyatlar bitir.

ADO sənədləşməsindən istifadə edərək, proqram kodunda **ConnectionString** xassəsinin qiymətini “əl ilə” VBA redaktorunda da yazmaq olardı. Misal üçün, müvafiq VBA sətiri aşağıdakı kod kimi görsənə bilər:

```
cn.ConnectionString="Provider=MSDASQL.1;DSN=FactExcel;DBQ=_
C:\Fact.xls;"
```

“Əl ilə” VBA kodunun yazılması əvəzinə lazımı qiyməti avtomatik rejimdə generasiyasını etmək olar. Çox sadə üsullar ilə həmin əməli aşağıdakı qayda ilə aparmaq mümkündür (əvvəlki bölmədə yazılan qaydalar əsasında):

- UDL faylını yaradıırıq (artıq hazır olan fayldan istifadə etmək olar);
- Həmin faylın piktoqramı üzərində mausun sağ düyməsi ilə iki dəfə şıqqıldadaraq **Provider** qurmasına keçirik və oradan **Microsoft OLE DB Provider for ODBC Drivers** seçirik;
- **Connection** qurmasına keçirik və **Use Data Source Name** siyahısından bizim özümüzün yaratdığımız **Excel_İstehlak** verilənlər mənbəyini seçirik. Həmin pəncərədəki digər sahələri doldurmamaq da olar, şəkl. 9.17. Verilənlər mənbəyi ilə bağlantının yaranmasını yoxlamaq üçün həmin **Connection** qurmasında **Test Connection** düyməsini basaraq testi keçirmək olar. Sonra isə pəncərəni bağlamaq üçün **OK** düyməsi basılmalıdır.



Şəkil 9.17. Yaradılmış ODBC mənbəyi ilə bağlantının yoxlanması (şəkildəki **Microsoft Data Link** pəncərəsində testin müvəffəqiyyətlə nəticələnməsi haqqında xəbər verilib).

- Sonuncu addımda – bizim yaratdığımız UDL faylını bloknotta açıyıq, oradan bağlantı (qoşulma) sətirini kopyalayırıq və VBA proqramında istifadə edirik.

Excel-ə bağlanmaq üçün VBA prosedura kodunun yekun görkəmi belə ola bilər:

```
Public Sub ConnectToExcel()
    Dim cn As New ADODB.Connection
    cn.ConnectionString="Provider=MSDASQL.1;Data Source=_
Excel_İstehlak"
    cn.Open
    'Recordset obyektini haqqında kitabın növbəti hissəsində məlumat
    'veriləcək.
    'Bu prosedura kodu əyani yoxlama üçün yerləşdirilib.
    Dim rs As New ADODB.Recordset
    rs.Open "Excel_İstehlak", cn
    MsgBox rs.GetString
End Sub
```

Bu hissədə, verilən nümunələr əsasında yekunlaşdıraraq, Excel faylının səhifəsindəki cədvəllə bağlantı yaradılması üçün çox vacib olan əsas əməliyyatları ardıcılıqları ilə aşağıda verirək:

1. **Excel** kitabındaki cədvəl üçün adlandırılmış diapazonu yaratmaq;
2. **Excel_İstehlak** adı ilə (verilmiş nümunədən fərqli olaraq, istifadəçi öz istədiyi adı verə bilər) **ODBC** verilənlər mənbəyini yaratmaq.
3. **Connection** obyektinin yaradılmasından başlayaraq, **Open** metodunun çağırışına qədər üç sətirlik VBA kodunu yazmaq.

9.5 Recordset OBYEKTİ VƏ FIELDS KOLLEKSIYASI

9.5.1 Recordset obyektinin işə salınması

Recordset obyektinin işə salınması, Open() metodu, Recordset açıldıqda sorğunun ötürülməsi

Adətən verilənlər bazası ilə bağlantı yaradıldıqdan sonrakı etap - **Recordset** obyektinin yaradılması və onunla işləmək kimi xarakterizə edilə bilər.

Recordset obyektini nədir? Bu sözün İngiliscədən deşifrəsi bu cür ola bilər: **Set of Records**, yəni Azərbaycanca – **yazılar yığıcı (və ya toplusu)**. Əyani olaraq, bu obyektin təsəvvür edilməsi - *kompyuterin operativ yaddaşında saxlanılan, böyük miqyasda olmayan "Excel cədvəlciyi", modeli* kimi anlanması daha çox məqsədə uyğundur. Nəzərə alınmalıdır ki, **Recordset** obyektinin Excel cədvəllərindən prinsipial olan fərqləri vardır:

- Praktiki işlərdə Excel-dəki cədvəllərin "ciddiliyinə" aid böyük tələblər yoxdur: məsələn, bəzən şirkətlərin və müəssisələrin gündəlik hesabat işlərində qruplara görə aralıq yekun hesablamalar Excel cədvəllərin ortasına yerləşdirilir (bəzən həmin yerlərə şərhlər də yerləşdirilə bilər). **Recordset** obyektini, belə demək olsa, "çox ciddi" cədvəldir. Bu cədvəllərdə sütunlar və sətirlər olduqca səlissə təyin edilir və onların

arasında parçalanma, boş yerlər, qadağandır (hərçənd ki, sətirlər və sütunlar kəsişməsində hansısa qiymətlər boş qala bilər);

- Excel cədvəlinin bir sütununda heç bir problem yaranmadan müxtəlif qiymətlər verile bilər - ədədi, düstur, tarix, zaman, sətir tipli v.s. **Recordset**-də hər bir sütun üçün verilənlər tipi əvvəldən təyin edilir və həmin sütundakı qiymətlər yalnız təyin edilmiş tipe aid olmalıdır.

Recordset obyektı adətən mənbədən alınan verilənlər əsasında yaradılır (bununla belə onu “əllə” də doldurub yaratmaq mümkündür). Hələ mənbədə olarkən, həmin verilənlər üçün sütunlar (**Fields**) və sətirlər (**Rows**) kimi obyektlər nəzərdə tutulmuş olur. **Recordset** obyektinin yaradılması və onu mənbədən gələn verilənlərlə doldurulması ən sadə halda bu cür VBA kodu ilə təsvir edilə bilər (fərz edilir ki, məsələn, **SQL Server**-də `rs` adlı verilənlər bazası ilə bağlantı `cn` obyektı ilə açılır):

```
Dim rs As New ADODB.Recordset
rs.Open "Customers", cn
```

Recordset-in gerçəkdən yaradılmış olduğunu və ya mövcud olmamasını (yuxarıdakı nümunəyə müvafiq olaraq) aşağıda verilmiş VBA kodu ilə yoxlamaq olar:

```
MsgBox rs.GetString
```

Recordset açıldıqda səhvlərin yaranma ehtimalı böyük olur, buna görə məsləhət görülür ki, adi hallarda olan kimi, səhvlər emalçısından istifadə edilsin. Bu baxımdan **Recordset**-də xüsusi **Errors** kolleksiyası nəzərdə tutulmamışdır, və ona görə yalnız standart **Err** obyektı istifadə edilir. Yuxarıdakı nümunədə biz Customers cədvəlini bütövlükdə açdıq. Mənbədən verilənləri çuxarmaq üçün bu yeganə (və çox yaxşı) olan üsul deyil. Məsələn, **SQL** dilindən istifadə edərək də `Open()` metoduna sorğu yaratmaq mümkündür. Yuxarıda baxdığımız nümunə üçün bu cür **SQL** kodu yazıla bilərdi:

```
rs.Open "select*from dbo.customers", cn
```

Sorğunun istifadə edilməsinin əsas səbəbləri budur:

- **Where** (şərt – məcburi olaraq, dırnaq arası verilir) filtrini işarə edilməsi və **Recordset**-ə bütün yazıları yox, yalnız istifadəçi istədiklərinin yüklənməsi mümkündür;
- qaytarılan sütunların sayını məhdudlaşdırmaq da mümkündür – nəticə etibarlı ilə ötürülən verilənlərin həcmi və maşının operativ yaddaşının sərfi azalmış olur;
- **SQL** funksiyaları mənbədə verilənləri sortlaşdırmaq və digər xeyirli imkanlara yol verir.

Real yaradılmış tətbiqi proqramlarda daha çox bu hala rast gəlmək olur: sorğu mətni müxtəlif yerlərdən gələn hissələrdən, faktiki olaraq, “yapışdırılır”, məsələn, ardıcıl olaraq aşağıdakı bir-biri ilə zəncirvari asılı olan prosesdə:

1. istifadəçi iç-içə açılan siyahıda sifarişçinin adını seçir;
2. dərhal iç-içə açılan siyahı üçün **Change** prosedurası işə salınır;
3. həmin prosedura öz növbəsində SQL Server-ə sorğu edir;
4. sonra sifarişçiyə aid verilənləri alır;
5. nəhayət alınan qiymətləri digər idarəetmə elementlərinə ötürür.

Biz baxdığımız nümunə üçün SQL proqram kodunun müvafiq sətiri bu cür ola bilər:

```
rs.Open "select*from dbo.customers Where CompanyName=" _ & _
      "' ' & ComboBox1.Value & "' ' , cn
```

"' ' " simvollar yığımı – bir qat dırnaq arası işarəsi iki dəfə ikiqat dırnaq arası işarəsinin içərisindədir. Yuxarıdakı konstruksiya sorğunun, məsələn, aşağıdakı strukturunun reallaşmasında istifadə edilir:

```
select*from dbo.customers where CompanyName='Qasım İsmayılov'
```

Səbəb çox sadədir – SQL dilində sətir dəyişənlərini bir qat dırnaq arası işarələrinin içərisində yazmaq lazımdır. Başqa praktiki məsləhət də bundan ibarətdir: əgər istifadəçi yalnız müştəri tətbiqi proqramı üçün yox, bütün verilənlər bazasının layihələndirilməsinə cavabdehdirsə, onda sorğunu yalnız təqdim əməllərdən təşkil edilməsi daha əlverişlidir. Bunun sayəsində təhlükəsizlik sisteminin çevik idarə edilməsi mümkün olur: verilənlər bazasının “yenidən biçilməsində” (məsələn, əsas cədvəlin cari və arxiv hissələrə ayrılmağı lazım olduqda) daha çox zamanın qənaət edilməsinə imkan yaranır. Daha bir vacib praktiki məsləhət budur: əlbəttə, verilənlər bazası ilə peşəkar səviyyəsində işləmək üçün SQL-in sorğular dilini bilmək çox xeyirlidir və vacibdir (bu mövzuya aid ədəbiyyat həddən artıq çoxdur – öyrənilməsi də heç bir çətinliklə bağlı deyil). Əgər verilənlər bazaları proqramçılığı sahəsində istifadəçi yeni başlayan səviyyəsindədirsə, onda SQL dilinin tətbiqi olmadan da effektiv proseduralar qurmaq olar. Sadəcə cədvəllər tamam açılmalıdır (heç bir sorğu etmədən) – qalan mərhələləri VBA dilinin vasitələri ilə yerinə yetirilməsi mümkündür.

9.5.2 Kursorun sazlanması və Recordset-in digər açılma parametrləri

Recordset obyektini açdıqda onun bir neçə vacib xassəsini təyin etmək olar (onları bir başa da təyin etmək olar, açılmadan əvvəl, o biri tərəfdən onları əlavə parametrlər kimi **Open** () metoduna ötürmək olar). **Recordset** obyektinin proqramlaşdırmada ən çox istifadə edilən xassələri aşağıda verilib:

- **CursorType** - birinci xassədir. Bu xassə yalnız **Recordset**-in açılmasından əvvəl təyin edilir (açıldıqdan sonra o, yalnız oxunma üçün əlçatan olur). Kursoru **Recordset**-də yazılan nişanlanma kimi təsəvvür etmək olar. Kursorun tipindən asılı olaraq, **Recordset** ilə işləmə imkanları və icra edilən əməliyyatların məhsuldarlığı təyin edilir (daha çox imkanlar

olduqda daha az məhsuldarlıq alınır və bunun əksinə). Aşağıdakı qiymətlərin verilməsi mümkündür:

- **adOpenForwardOnly** – bu qiymət standart vəziyyətdə istifadə edilir: maksimal məhsuldarlığa çatmaq üçün optimallaşdırılır (onun imkanları minimal olacaq). Kursor yalnız irəli gedəcək, başqa istifadəçilərin etdiyi dəyişiklər görünməyəcək;
- **adOpenStatic** – yuxarıdakı qiymətə analojidir (yeganə fərq budur ki, kursor bütün istiqamətdə hərəkət edə bilər);
- **adOpenKeyset** – kursurun istənilən istiqamətdə hərəkətinə imkan yaradır, başqa istifadəçilərin yalnız mövcud olan yazılarda etdikləri dəyişikliklər görünür (köhnəmiş yazıların silinməsi və yeni yazıların əlavə edilməsi görünmür);
- **adOpenDynamic** – maksimal imkanları təmin edir: istənilən istiqamətdə hərəkət etməyə imkan verir, başqa istifadəçilər etdiyi yazılarda istənilən dəyişiklər görünür. Təəssüf ki, **Microsoft.Jet.OLEDB.4.0** provayderi bu tip kursoru himayə etmir, buna görə Access və Excel-də onu istifadə etmək olmur. **Recordset.RecordCount** xassəsi normal tərzdə yalnız **adOpenStatic** və **adOpenKeyset** tip kursorlarla funksional əlaqə saxlayır.

■ **CursorLocation** – ikinci vacib xassədir (kursorun harada olduğunu təyin edir: serverdə yaxud müştəridə) Standart vəziyyətdə **adUseServer** qiyməti qurulmuş olur (serverdə yaratmaq). Xassənin ikinci qiyməti **adUseClient** (müştəridə yaratmaq). Bütün hallarda ən əlverişli və məhsuldar üsul – kursoru serverdə yaratmaqdır. Yalnız bircə halda – müxtəlif mənbələrlə işlədikdə kursurun müştəridə yaradılması daha yaxşı nəticələr verir.

■ **LockType** – üçüncü vacib xassədir (sonradan **Recordset**-də yerləşdirən, mənbədəki yazılara bloklanmanın tipini təyin edir). Bu xassənin qiymətləri aşağıda verilir:

- **adLockReadOnly** - qiyməti **Recordset**-dəki yazılar yalnız oxumaq üçün əlçatan olacaq (yazıları redaktə etmək mümkün olmur). Standart halda bu qiymət qurulmuş olur;
- **adLockPessimistic** – qiyməti verilənlərin salamat saxlanmasının etibarlılığına görə ən əlverişli qiymətdir. **Recordset**-də yazıları dəyişdirmək olur. Əgər redaktə etmək lazımdırsa, onda o, mənbədə elə bloklanma yaradır ki, **Update()** və ya **CancelUpdate()** metodları çağırılmasa, başqa istifadəçilər yazıları nə oxumaq nə yazmaq üçün istifadə edə bilməyəcəklər;
- **adLockOptimistic** – bu qiymət məhsuldarlıqda qazanmağa, o biri tərəfdən verilənlərin salamat saxlanmasının etibarlılığına görə uduzmaqla digər tərəfdən yaxşı imkanlar yaradır. Mənbədəki yazı yalnız **Update()** metodunun icrasında bloklanır. Digər istifadəçilərin həmin zaman verilənləri oxumaq və redaktə etməyə imkanları olur;

- **adLockBatchOptimistic** – qiyməti eynilə **adLockOptimistic** qiymətinin funksiyasını yerinə yetirir. Yeganə fərqi: yalnız bir yazıya görə dərhal yeniləşmə əvəzinə bu qiymət paket yeniləşməsini istifadə edir. Çox sayda yazılar redaktə edildikdə bu qiymət məhsuldar nəticələr verir.

Yuxarıda təqdim etdiyimiz nümunələrdə **Open** () metodunun birinci parametri cədvəlin adı olaraq, həmçinin **SQL** əmrinin funksiyasını yerinə yetirirdi (əslində başqa variantlardan da istifadə etmək olar). **OLE DB** drayverinin başlanğıcdan ötürülən mətnin nə olduğunu təyin edə bilmir. Lakin verilənlər bazası serveri ilə qarşılıqlı əməliyyat görür. Praktiki işlərdə bu cür proseslər tətbiqi proqramın yerinə yetirilməsini xeyli gecikdirə bilər. Buna görə **Recordset** açıldıqdan əvvəl ona ötürülən parametrləri işarələmək lazımdır. Bu əməliyyatı **Options** parametri ilə həyata keçirirlər (qiymətlər bu vasitə ilə metoda ötürülür).

Ən çox istifadə edilən qiymətlər isə bunlardır:

- **adCmdText** – SQL əmrinin ötürülməsi;
- **adCmdTable** – cədvəlin adı ötürülür (cədvəldən bütün yazıları qaytaran SQL əmrinin generasiyası ilə eyni güclüdür);
- **adCmdTableDirect** – bir başa yazıların alınması üçün cədvəlin adının ötürülməsini təmin edir (SQL-sorğusu icra edilmədən, əgər mənbə həmin əməliyyata şərait yaradırsa);
- **adCmdStoredProc** – yaddaşda saxlanılan prosedurasının icrası üçün onun adını ötürür, qaytarılan məlumat isə **Recordset**-i doldurmaq üçün istifadə edilir.

Praktiki nöqteyi nəzərdən bunu bilmək çox vacibdir ki, əgər **Recordset**-dəki yazılarda istənilən istiqamətdə hərəkət etmək və yazıları redaktə etmək lazımdırsa (yuxarıda verilmiş nümunələrə müvafiq şəkildə), onda **Recordset**-in işə salmaq üçün aşağıda verilmiş VBA kodundan istifadə etmək lazımdır:

```
Dim rs As New ADODB.Recordset
rs.CursorType=adOpenStatic
rs.LockType=adLockOptimistic
rs.Open ... 'Burada nəyin açılması yazılır
```

Praktiki hallarda peşəkar proqramçılar aşağıdakı VBA kodundan istifadə edirlər (nəzərinizə yetiririk ki, proqram kodları təqdim edilən nümunələrə uyğun tərtib edilib):

```
rs.CursorType=adOpenStatic
rs.LockType=adLockOptimistic
```

Yuxarıdakı xassələrin qiymətlərinin digər varianta dəyişdirilməsi xüsusi hallardan asılı ola bilər (məsələn, məhsuldarlıq meyarı önəmli olduqda).

9.5.3 Recordset-də hərəkət etmə imkanları

ADO/VBA mühitində Recordset-də hərəkət etmə imkanları, BOF, EOF və Bookmark xassələri, Move(), MoveNext(), MovePrevious(), MoveFirst(), MoveLast(), Find(), Seek() metodları.

Recordset obyektini yaradıqdan sonra onunla müxtəlif əməllərin aparılması zamanı gəlir. Ən sadə əməl isə budur – **Recordset** obyektini üzərində hərəkət etmə əməllərinin yerinə yetirilməsi. **Recordset**-də həmişə məhdud sayda yazılar olur (mənbədən alınan sayda). Əvvəldən kursor **Recordset**-in birinci yazısına təyin edilir (buna əmin olmaq üçün **AbsolutePosition** xassəsindən istifadə edirlər). Əgər bir dəfə **MovePrevious()** əmri verilsə, onda səhv olmayacaq (həmin əmr ikinci dəfə verilsə - səhv əmələ gələcək). **AbsolutePosition** xassəsi həmin hal üçün (-2) qiymətini qaytarır. Bu onunla bağlıdır ki, **Recordset**-də mənbədən alınmış birinci yazıdan əvvəl xüsusi **BOF** yazısı yerləşdirilir (İngiliscə “Begin Of File” sözündən əmələ gəlib, hərçənd ki, əslində heç bir fayl yoxdur). Hal-hazırda həmin xüsusi yazıda olub-olmamağı **BOF** xassəsi ilə yoxlayırlar. Məsələn, aşağıdakı kod:

```
Debug.Print rs.BOF
rs.MovePrevious
Debug.Print rs.BOF
```

əvvəlcə bizə **False** qiymətini sonra isə **True** qiymətini qaytaracaq.

Eynən həmin qayda ilə sonuncu yazıdan sonra **Recordset**-də xüsusi **EOF** yazısı olur (İngiliscə “End Of File” sözündən əmələ gəlib). Kursorun həmin yazıda olub olmamasını eyni adlı **EOF** xassəsi ilə yoxlayırlar.

Bəzən də belə alınır ki, **Recordset** açıldıqdan dərhal sonra **BOF** və həmçinin **EOF** eyni zamanda **True** qiymətini qaytarır. Bu hadisənin izahı çox asandır: **Recordset**-də hansısa məlum olmayan səbəbə görə heç bir yazı qayıtmamışdır. Belə hallarda, gözlənilməz hadisələrin qarşısını almaq üçün, məsləhət görülür ki, **Recordset** açılan kimi dərhal onda yazıların olması yoxlanılsın.

Recordset-də cari pozisiyanı müəyyən etdikdən sonra bu obyektə hərəkət etmə imkanları təyin edilməlidir. Bunu etmək üçün ən asan üsul **Move...()** metodlarından istifadə edilməsidir. Həmin metodların qısa təsvirini aşağıda veririk:

- **Move()** – metod iki parametri qəbul edir: **NumRecords** – hansı sayda yazı yerdəyişməsi imkanını təyin edir (bu ədəd mənfi də ola bilər – yəni neçə yazıya geri keçmək imkanını bildirir), ikinci parametr – məcburi deyil (yerdəyişmənin başladığı əlavə qurmanın adını təyin edir). Məsələn, üç əlavə qurmadan istifadə etmək olar: cari, birinci və sonuncu yazı üçün. Əgər əlavə qurmanın adı işarə edilməyibsə, onda yerdəyişməni cari pozisiyadan başlamaq olar;
- **MoveFirst()**, **MoveLast()**, **MoveNext()** və **MovePrevious()** - bu metodların təyin edilməsi adından da məlum ola bilər: müvafiq olaraq, birinci, sonuncu, sonrakı və əvvəlki yazıya keçidi təyin edir;

Qeyd etməliyik ki, **Forward-only** metodundan istifadə edərək, açılmış kursurun köməyi ilə geriye keçid (**MovePrevious()** və ya **Move()** metodlarının mənfi qiymətlərini təyin edərək) gözlənilməz nəticəyə gətirə bilər. Verilənlər mənbəyindən asılı olaraq, səhvdən başlayaraq, təsadüfi yazıya keçməyə qədər.

Bütün yazılara keçid üçün daha çox aşağıda göstərilən sadə VBA kodunda yazılmış alqoritmindən istifadə edirlər (yuxarıdakı hissədə verilmiş nümunələrə uyğun olaraq, **rs** adı verilmişdir):

```
rs.MoveFirst
Do Until rs.EOF
'Yazı ilə nə isə edirik, məsələn, lazımı sahənin qiymətini alırıq.
    rs.MoveNext
Loop
```

Əgər bir başa lazımı yazıya keçmək lazımdırsa, onda **Find()** və **Seek()** metodlarını istifadə edirlər. **Find()** metodu bir sütunun qiymətinə görə axtarış aparmaq üçün nəzərdə tutulub. O, parametr kimi axtarış meyarını qəbul edir – axtarışın haradan başlanmasını, başlanğıc pozisiyadan neçə sayda geridən başlamasını və axtarışın istiqamətini təyin edir. Burada ən əlverişlisi odur ki, axtarış meyarı təyin edildikdə, yerinə qoyma simvolları olan **Like** operatorundan istifadə etmək olar. Lazımı yazı tapıldıqda, **Find()** metodu kursuru tapılmış yazıya qoyur. Digər halda, yazı tapılmadıqda, kursur **EOF** və ya **BOF** (axtarış əks istiqamətdə aparılıbsa) yazısına qoyulur. Məsələn, tutaq ki, bizim nümunədəki **Customers** cədvəlində bütün İrland şirkətlərini **Recordset**-də tapmaq lazımdır. Onda aşağıdakı VBA kodundan istifadə etmək olar:

```
rs.Find "country='Ireland'"
Do While Not rs.EOF
    Debug.Print "Şirkətin adı: "; rs.Fields("CompanyName")
    mark=rs.Bookmark
    rs.Find "country='Ireland'",1, adSearchForward, mark
Loop
```

Bu nümunədə hələ bizə məlum olmayan **Fields** və **Bookmark** obyektləri istifadə edildi. Onların təyinatı aydındır: **Field** obyektini şirkətin adını çıxarmaq və **Bookmark** obyektini +1 pozisiyasının axtarışını sonuncu tapılmış yazıdan davam etmək üçün lazımdır. **Seek()** metodu **Find()** metodundan onunla fərqlənir ki, bu metodda qiymət indeksə görə axtarılır (**Recordset** üçün **Index** obyektini proqram üsulu ilə və ya avtomatik rejimdə yaradılır - əgər cədvəl əsasında yaradılmış **Recordset**-ə, məsələn, **Primary Key** məhdudiyəti qoyulmuşdursa). Bu metod yalnız **TableDirect** tipli əmrlə server kursuru üçün işləyir, buna görə də istifadə üçün məsləhət görülmür.

İndi isə **Recordset**-də yerdəyişməni yerinə yetirmək üçün (nümunələrdə artıq ona rast gəlmişik) çox böyük xeyir gətirə bilən daha bir xassə **Bookmark** haqqında danışaq. Bu xassə çox asandır – kursur **Recordset**-in lazımı yerində durduqda, onun qiymətini **Variable** tipinə

təyin edilməsi kifayətdir. Dərhal yenidən onun üstünə qaydılması mümkün olacaq (bizim təqdim etdiyimiz nümunədə olan kimi). Ümumiyyətlə, bu xassə qaytardığı qiymət əvvəldən **Recordset**-dəki yazının nömrəsi ilə üst-üstə düşür və Microsoft xəbərdar edir ki, bu üsul ilə istifadə edilməsi məsləhət görülmür. Çünki, əgər kursor eyni yerdə durursa, onda **Bookmark** xassəsi müxtəlif qiymətləri qaytara bilər.

9.5.4 **Fields** kolleksiyası və **Field** obyektləri

ADO/VBA mühitində Recordset obyektinin sahələri ilə işləmə qaydaları, Fields kolleksiyası və Field obyektləri, Name və Value xassələri

Recordset-in əsas məzmunu – sətirlərin və sütunların kəsişməsində yerləşən hücrələrdəki qiymətlər ilə müəyyənləşir. **Recordset**-də sətirlər dedikdə yazılar (**Records**) anlanmalıdır və onlar müvafiq **Record** obyektləri ilə təmsil edirlər. **Recordset**-də sütunlar sahə adlanırlar – **Fields** kolleksiyasında yerləşən **Field** obyektləri. **Record** obyektləri tez-tez istifadə olunmur – çünki onların adı olmur. Yazılar arasındakı keçidləri isə **Recordset** obyektinin öz metodları və xassələrinin (**AbsolutePosition**, **Find()**, **Move()** v.s.) köməyi ilə həyata keçirmək daha asandır. **Fields** kolleksiyası və **Field** obyektləri praktiki olaraq, hər proqramda istifadə edilir.

Hər bir **Collection** obyektində olan kimi, **Fields** kolleksiyasının bütün xassələri standartdır:

- **Count** – xassəsi **Recordset** obyektindəki sütunların sayını təyin edir;
- **Item** – adına və nömrəsinə görə lazımı sütunun (**Field** obyektini) qaytarılması üçün imkan yaradır. Bu xassə standart hala aiddir deyə bizim nümunəyə bənzər koddan istifadə etmək olar: `rs.Fields("CompanyName")`. Bu xassəyə müraciət etmək üçün daha bir sintaksis variantı var: `rs!CompanyName`

Standart kateqoriyaya aid olan metodlar və xüsusi təyinatlı metodlar da var:

- **Append()** və **Delete()** – metodları **Recordset**-ə, müvafiq olaraq, yeni sütunun əlavə edilməsinə (silinməsinə) kömək edir. **Open()** metodunun çağırılmasından və ya **ActiveConnection** xassəsinin qurulmasından əvvəl hər iki metodları yalnız **Recordset**-in qapanmış vəziyyətində icra etmək mümkündür.
- **Update()** – metodu **Recordset**-dəki dəyişiklərin saxlanmasına qulluq edir (verilənlər mənbəyində yeni sütunun yaradılmasına cəhd edilir). Əgər verilənlər mənbəyi hansısa səbəbə görə həmin dəyişiklikləri qəbul etməkdən imtina edərsə, onda səhv yaranacaq.
- **CancelUpdate()** – metodu, yuxarıdakı **Update()** metodunun əksinə olaraq, **Recordset**-də edilən dəyişiklikləri ləğv edir.
- **Resync()** – yalnız **Record** obyektinin (**Recordset** ilə səhv salmayı) **Fields** kolleksiyaları üçün fəaliyyət göstərir: sətirdəki qiymətləri yeniləşdirir.

Field obyektinin xassələri isə daha maraqlıdır:

- **ActualSize** – cari yazı üçün verilənlərin real ölçüsünü təyin edir.
- **DefinedSize** – verilənlər mənbəyindən alınan verilənlərin sütundakı nominal ölçüsünü təyin edir.
- **Attributes** – sütunlar atributları üçün bit maskasını təyin edir (boş olan qiymətlərə icazə olduğu, mənfi qiymətlərdən istifadə edilməsi, yeniləşdirməyin olması, fiksə olmuş uzunluqda verilənlər tipinin olmaması v.s. haqqında məlumatı qaytarır).
- **Name** – sadəcə sütunun sətir adıdır. Mənbədən alınan sütunlar yalnız oxumaq üçün əlçatan olur.
- **NumericScale** və **Precision** – qiymətin təqdim edilməsi üçün istifadə edilən qiymətləri təyin edir (müvafiq olaraq, vergüldən sonra mənası olan ədədlərin, rəqəmlərin ümumi maksimal sayını təyin edir).
- **Value** – xassəsi **Field** obyektinin ən vacib xassəsidir. Sütunda olan qiyməti təyin edir (əgər **Record** obyektini **Fields** kolleksiyasının içərisindən keçibsə, onda yalnız həmin yazı üçün aktiv olacaq, digər halda, əgər **Recordset** obyektini **Fields** kolleksiyasının içərisindən keçməyibsə, onda yalnız cari yazı üçün aktiv olacaq). Kursurun tipindən asılı olaraq, oxumağa və ya yazmağa əlçatan ola bilər. ADO böyük ölçülü ikili kod formasında ötürülən verilənlərlə işləməyə imkan verir (müxtəlif görüntülər, sənədlər və arxivlər). **OriginalValue** – qiyməti dəyişiklikdən əvvəl sütunda olan qiyməti təyin edir, **UnderlyingValue** – qiyməti verilənin mənbəyində olan qiyməti təyin edir (**Recordset**-də işlərkən həmin qiymət dəyişə bilər. Buna görə **OriginalValue** və **UnderlyingValue** üst-üstə düşməyə bilər). **Value** xassəsi – standart xassədir, ona görə aşağıdakı VBA kod sətiri eyni güclüdür:

```
Debug.Print rs.Fields("CompanyName")
```

və aşağıdakı kod sətiri

```
Debug.Print rs.Fields("CompanyName").Value
```

- **Status** - qiyməti **adFieldOK**-dən fərqlidir (0 qiyməti olduqda - sahə yaxın zamanda **Recordset**-ə əlavə edilib və ya əlavə edildikdə verilənlər mənbəyində səhv yaranıb).
- **Type** – sənədləşmədə verilən cədvələ uyğun olaraq, verilənlər tipini təyin edir. Məsələn, **nvarchar** verilənlər tipi üçün xassə 202 qiymətini qaytarır.

Field obyektinin yalnız iki metodu var: **AppendChunk()** və **GetChunk()**. **Value** xassəsi ilə işləmək mümkün olmayanda bu iki üsuldən istifadə edirlər (böyük ölçülü ikili kodla ötürülən

verilənlər tipi ilə işləmək üçün tətbiq edilir: müxtəlif formatlı görüntülər, sənədlər, arxivlər v.s. üçün).

9.5.5 Verilənlərin sortlaşması və filtrasiyası

Recordset-də verilənlərin sortlaşması və filtrasiyası, Sort və Filter xassələri, sortlaşma və filtrasiyada məhdudiyyətlər

Recordset-də verilənlər mənbədən gəldikləri qaydada yerləşdirilir. Əgər sorğuda sortlaşmanın xüsusi qaydası göstəlmirsə, onda verilənlər mənbədəki parametrlərə uyğun qaytarılır (məsələn, SQL Server-də onlar standart halda başlanğıc açar üçün yaradılan klaster indeksinə görə səhmanlaşdırılır). Sorğuda **ORDER BY** ifadəsini göstərərək, sortlaşmanı serverdə də aparmaq olar. Başqa yolla da sortlaşmanı aparmaq mümkündür - müştərinin mühitində: **Recordset** obyektinin **Sort** xassəsi ilə. Ümumi qaydanı bu cür ifadə etmək olar: imkan varsa, birinci növbədə sortlaşmanı serverdə aparmaq lazımdır. Çünki server bu cür əməliyyatların aparılması üçün daha yaxşı optimallaşdırılıb: serverin operativ yaddaşı və prosessor resursları müştəri mühitindən daha böyük olur. Müştəri mühitində sortlaşmanı yalnız o halda aparırlar ki, hansısa səbəbə görə həmin əməliyyatı serverdə aparmaq mümkün olmur (məsələn, sortlaşma nəzərdə tutulmamış proseduradan verilənlər alınmalıdır). Bununla belə, **Recordset**-dəki sortlaşma daha az resurs tutumludur (verilənlər, fiziki olaraq, yerlərini dəyişmir, yalnız sortlaşma aparılması sahə üçün yeni indeks yaradılır). Buna görə böyük həcmli verilənlərin sortlaşmasını aparmaq üçün **Recordset**-dəki sortlaşma üsullarını proqramlarda da istifadə etmək olar.

Sort xassəsinin tətbiqində iki çox ciddi məhdudiyyət var:

- onu yalnız kursorun müştəridə açıldığı zaman tətbiq etmək mümkündür (bu deməkdir ki, **CursorLocation** xassəsi **adUseClient** qiymətində qurulmadır – standart halda o, serverdə açılır);
- onu bəzi drayverlər ilə işlətmək mümkün deyil (məsələn, SQL Server və ya Oracle drayverlərinə qoşulduqda, sortlaşmanı aparmaq mümkün olmur);

Sort xassəsinin tətbiqi çox sadədir:

```
rs.Sort="CompanyName"
```

Həmin zaman bu xassəyə ötürülən hər hansı parametr **Recordset**-dəki sütunun (yəni **Field** obyektinin) adı olmalıdır.

Sortlaşma qaydasına uyğun olaraq, aşağıdakı nümunədə VBA kodu ilə bir neçə (iki) sütunun adının ötürülməsi göstərilib:

```
rs.Sort="Country ASC, CompanyName DESC"
```

Bu nümunədə **Recordset** əvvəlcə ölkəyə görə sortlaşacaq. Sonra isə şirkətin adına görə, azalan qayda əsasında, sortlaşma davam edəcək (standart halda artan qaydanı təmin edən **ASC** əzəldən qurulmuş olur). Sortlaşmadan imtina edilməsi və dərhal yazılara (onlar mənbədən qələn qaydada qurulur) qayıtmaq üçün **Sort** xassəsinə boş qiymətin verilməsi kifayətdir:

```
rs.Sort=""
```

Recordset-də yazıları filtrasiya etmək olar. Filtre olmuş yazılar **Recordset**-də qalır və aşağıdakı xassələr əmələ gəlir:

- yerdəyişmə və axtarış əməliyyatlarında yazılar görünmür;
- filtre edilmiş yazılarda kursoru qurmaq olmur.

Prinsipial nöqteyi nəzərdən bütün bu çətinlikləri mənbəyə göndərilən sorğudakı yazıların filtrasiyası ilə həll etmək olur, yaxud da yerdəyişmə əməliyyatlarında əlavə yoxlamaların qurulması ilə (bəzi hallarda, sintaktik nöqteyi nəzərdən, filtrasiyanın tətbiqi daha əlverişli olur).

Recordset obyektində filtrasiyanı aparmaq üçün **Filter** xassəsindən istifadə edirlər. Bu xassə üç qiyməti ala bilər:

- sintaksisi **Find()** metoduna oxşayan sətir qiymətini:

```
rs.Filter="LastName='Smith' AND FirstName='John'"
```

yer dəyişmə simvolları (yalnız “*” və “%”) ilə **Like** operatorundan istifadə etmək olar. **Find()** metodundan fərq bundan ibarətdir ki, **Find()**-da kursor sadəcə birinci tapılan uyğun olan yazının üstünə qurulur, **Filter**-də isə hər nə uyğun gəlmirsə görünmür olur.

- əlavə qurmalar çoxluğu (array);
- 5 sayda xüsusi qiymətlər: bütün konflikt edən yazılar, mənbədə saxlanmalarını gözləyən yazılar v.s.

Filtrasiyadan imtina edilməsi sortlaşmadan imtina edilməsinə bənzəyir:

```
rs.Filter=""
```

9.5.6 **Recordset** obyektini ilə mənbədəki yazıların dəyişdirilməsi

Recordset-də *verilənlərin əlavə edilməsi, dəyişdirilməsi, yazıların silinməsi, AddNew(), Update() və Delete() metodları.*

Verilənlər bazasının yaradılması və istismarında ən vacib iş: *tətbiqi proqramdan idarə edərək, mənbədəki yazılarda dəyişiklik etmək və lazım olduqda, istənilən məlumatı həmin mənbədən almaq.* Bu kompleks məsələnin həllində çox sayda problemlər yarana bilər, məsələn:

- Cədvəllərin sayı çox olduqda - hansı cədvəldəki verilənlər dəyişilməlidir (götürülməlidir)?

- Cədvəllərdə (cədvəldə) bloklama varmı (əgər varsa, əks əməliyyat necə aparılmalı)?
- Məhsuldarlığa, icazələrə, sil-silə tipli yeniləşməyə məhdudiyət varmı?
- İstifadəçi etdiyi dəyişikləri ləvğ edərək, yazıları bərpa etmək olurmu v.s.?

Problemlərin əksəriyyətini çox sadə və etibarlı üsulla bu qaydaya riayət edərək həll etmək mümkündür: istənilən dəyişiklikləri yalnız saxlanılan proseduraların köməyi ilə aparmaq (və ya, ehtiyac olduqda, **Command** obyektinin köməyi ilə). Aşağıda, ən sadə hallarda tətbiq edilməsi məsləhət görülən, **Recordset** obyektinin köməyi ilə dəyişikliklər etmə üsulları haqqında məlumat verəcəyik. Yadda saxlanılmalıdır ki, **Recordset** obyektini açıldıqda, standart halda **LockType** xassəsi **adLockReadOnly** qiyməti ilə qurulmuş olur. Nəticədə **Recordset**-də dəyişikliklər edilməsi mümkün olmur. Buna görə **Recordset** açılmamışdan əvvəl **LockType** xassəsinin qiyməti dəyişdirilməlidir.

Recordset-dən aparılan dəyişikliklərin əlavə edilməsinin ümumi sxemini bu cür təsvir etmək olar:

1. əvvəlcə lazım olan metodlardan birisi çağırılmalıdır (**AddNew()**, **Edit()** və ya **Delete()**);
2. sonra müştəri mühitində operativ yaddaşda dəyişikliklər əlavə edilməlidir;
3. növbəti əməliyyat - **Recordset** üçün **Update()** metodu çağırılmalıdır, həmin metod əlavə edilən dəyişiklikləri verilənlər mənbəyinə yazır (**Delete()** çağırıldıqdan sonra **Update** metodunun çağırılmasına ehtiyac qalmır).

Həmin metodlar haqqında ətraflı məlumat verək:

- **AddNew()** - yeni dəyişiklikləri əlavə edilməsinə xidmət edir və bu metod əslinə üç hissədən ibarət olan əməliyyatdır:
 1. əvvəlcə bu metod çağırılmalıdır (yeni boş olan yazını yaratmaq üçün – cursor, avtomatik olaraq, həmin yazının üstünə qurulmuş olacaq);
 2. sonra, **Fields** kolleksiyasının **Value** qiymətini istifadə edərək, qiymətləri sütunlara əlavə etmək lazımdır;
 3. mənbədə əlavə edilmiş dəyişikləri yazmaq üçün **Update()** metodu çağırılmalıdır.

Verilənlər bazaları ilə, kitabın bu fəslində verilən nümunələrə müvafiq olaraq, **Recordset**-ə bu metodun tətbiqi VBA proqram kodu ilə aşağıdakı kimi yazıla bilər:

```
Dim rs As ADODB.Recordset
Set rs=CreateObject("ADODB.Recordset")
rs.CursorType=adOpenKeyset
rs.LockType=adLockOptimistic
rs.Open "select*from dbo.customers", cn
rs.AddNew
```

```
rs.Fields("CompanyName")="Test rs company"
rs.Fields("Country")="Ireland"
rs.Fields("CustomerId")="TESTR"
rs.Update
```

Prinsipial nöqteyi nəzərdən yeni qiymətləri **AddNew()** metodunun özündə də etmək olardı - sintaktik tərəfdən bunun həyata keçirilməsi çox çətin məsələ çevrilə bilər.

- **Edit()** – mövcud olan yazının redaktə edilməsini təmin edir. Bununla belə, müvafiq olan VBA proqram kodunun görkəmi çox sadədir:

```
rs.Find "CustomerID='ALFKI'"
rs.Fields("ContactName")="Ibrahim"
rs.Update
```

- **Delete()** – yazının mənbədən silinməsini təmin edir. Bu metodun (müvafiq nümunə ilə bağlı) VBA proqram kodu daha da sadədir:

```
rs.Find "CustomerID='TESTR'"
rs.Delete
```

Diqqət edin ki, bu halda **Update()** metodunun çağırılmasına ehtiyac qalmır!

Əlbəttə, mənbədə dəyişikliklər əlavə edildikdə, səhvlər çox saylı səbəblərə görə əmələ gələ bilər. Buna görə məsləhət görülür ki, əvvəlcədən səhvlərin emalçısı proqramda qurulmuş halda olsun və standart **Err** obyektinin xassələri analiz edilsin.

9.5.7 Recordset obyektinin digər xassələri və metodları

Recordset obyektinin **Close()**, **GetRows()**, **GetString()**, **Save()** **metodları**.

Recordset obyektin digər bəzi vacib olan metodları və xassələri haqqında aşağıda ətraflı məlumat verilir (bu obyektinin, yuxarıda baxılan, əsas metodlarından və xassələrindən fərqləri yalnız ondan ibarətdir ki, adi, peşəkar olmayan, istifadəçilər onlardan az istifadə edirlər, adətən onların bu xassələr haqqında kifayət qədər xəbərləri olmur):

- **AbsolutePage**, **PageSize** və **PageCount** – xassələri yazılar qrupunu istifadə etməyə imkan verir (**Recordset**-də hərəkət etmə üçün səhifələr). Standart halda səhifənin ölçüsü 10 yazıya bərabərdir;
- **ActiveCommand** – xassəsi **Command** obyektini qaytarır (bu obyekt **Recordset**-in yaradılması və onun yazılarla doldurulmasında istifadə edilmişdir). **Command** obyektini haqqında aşağıda danışılacaq;
- **ActiveConnection** – xassəsi **Connection** obyektini qaytarır (bu obyekt **Recordset**-in yaradılmasında istifadə edilmişdir). **Connection** obyektini yaratmaq üçün **Source** xassəsindən (sətir qiymətinin alınmasını/ötürülməsini təmin edir) istifadə edirlər;

- **CacheSize** – xassəsi müştəridəki operativ yaddaşda yerləşən yazıların sayını təyin edir (qalan yazılar tədricən, tələb olduqda, yüklənəcək). Bu xassədən **Recordset**-də çox böyük sayda yazı olduqda və ya yazının həcmi böyük olduqda istifadə edirlər (məsələn, ikili coddə verilmiş obyektlərlə işlədikdə);
- **DataSource** və **DataMember** – xassələri yalnız **Data Environment** tətbiq edildikdə istifadə edilir ;
- **EditMode** - xassəsi cari yazının vəziyyətini təyin etməkdə kömək edir (dəyişilib, dəyişilməyib, dəyişikliklər hələ mənbəyə ötürülməyib, silinib v.s.).
- **InsertCommand**, **DeleteCommand**, **UpdateColumn** - xassələri **Command** obyektlərini təyin edir (**Command** obyektləri mənbədə istifadə edilir: **Recordset**-də, müvafiq olaraq, yazıların yaradılması, dəyişdirilməsi və silinməsinə təmin edən əmrlərdir).
- **MarshalOptions** – xassəsi **Recordset** dəyişdirildikdə müştəridən serverə qayıdan yazıları təyin edir: standart halda bu qiymət qurulmuş olur;
- **MaxRecords** – xassəsini **Recordset** obyektlərinin açılmasında çox sayda yazıların alınması ehtimalı olduqda istifadə edirlər (müştəridəki operativ yaddaşın çatışmaması ilə rast gəlməmək üçün). Bu xassə **Recordset**-ə yüklənən yazıların maksimal hüdud sayını təyin edir. Bu məqsədə çatmaq üçün **CacheSize** xassəsindən də istifadə etmək olar.
- **State** – xassəsi **Recordset**-də hal-hazırda nə baş verdiyini müəyyən edir. 5 sayda qiymətdən biri istifadə edilir: açıqdır, bağlıdır, birləşir, mənbədə əmri yerinə yetirir və ya mənbədən verilənləri alır.
- **Status** – xassəsi verilənlərin yeniləşməsindəki son əməliyyatın nəticəsini təyin edir;
- **StayInSync** – xassəsi yalnız iyerarxik olan **Recordset**-lərdə istifadə edilir. Baş elementlərin köçürülməsində də elementlərin iyerarxiyada köçürülməsini və ya yerində qalmasını təyin edir.

Recordset obyektinin metodları aşağıdakı siyahıda təmsil edilir:

- **Cancel ()** – metodu **Recordset**-in açılmasının qarşısını alır (məsələn, əgər açılma uzun zaman çəkirsə);
- **CancelBatch ()** və **CancelUpdate ()** – metodları **Recordset**-ə əlavə edilmiş dəyişiklikləri ləvğ edir (əlbəttə, **Update ()** əmrinin icrasından əvvəl);
- **Clone ()** - metodu hansısa bir **Recordset** obyektinin digərinin içərisinə kopya etməyə imkan verir (adətən birdən artıq cari yazının olmasına ehtiyac yarananda istifadə edilir);
- **Close ()** – metodu **Recordset** verilənlərinin operativ yaddaşda tutduğu yeri azad etməyə imkan yaradır (lakin obyektin özünü silmir). Lazım olduqda, əvvəl təyin edilmiş

parametrlərlə obyekt yenidən bərpa etmək ehtiyacı yaranarsa, onda yenidən `Open()` metodu çağırıla bilər. Nəzərə alınmalıdır ki, `Recordset` obyektinin xassələrinin qiymətləri `Close()` metodu çağırıldıqda dəyişmir;

- `CompareBookmarks()` – metodu iki nişanı bir biri ilə müqayisə edib, yoxlamanın nəticəsini qaytarır (həmin yazıya işarə edir, birincisi yuxarıdadır, birincisi aşağıdadır v.s.);
- `GetRows()` – metodu `Recordset`-dən `Variant` tiplərinin 2-ölçülü çoxluğunu (array) qaytarır. Məcburi olmayan parametrlər kimi start pozisiyasını qəbul edir (çoxluğa yerləşdirilən sətirlərin sayını və `Recordset`-dən götürülən sütunları). Bu metodun tətbiqini aşağıdakı VBA proqram kodu ilə yazmaq olar (yuxarıda təqdim etdiyimiz nümunəyə müvafiq olaraq):

```
arr=rs.GetRows
Debug.Print arr(2,3)
```

- `GetString()` – metodu `Recordset`-dən sətir qiymətinin alınması üçün ən sadə vasitədir. Standart halda sütunlar arasında olan ayrıcılar (tabulyasiya işarələrinin) rolunu oynayır. Çox hallarda bu xassədən peşəkar VBA proqramçıları proqramın sazlanma mərhələsində `Recordset`-ə, əyani olaraq, baxmaq üçün istifadə edirlər. Məsələn, aşağıdakı nümunə kodda olan kimi:

```
Debug.Print rs.GetString
```

- `NextRecordSet()` - metodu ilə cari `Recordset`-i təmizləyərək, `Open()` metodunda verilən əmri icra edir. Adətən bu metoddan eyni tipli çox sayda olan cədvəllər olduqda istifadə edirlər;
- `Requery()` – metodu, `Open()` metodunda istifadə edilən, sorğunu təkrarən açıb, yenidən `Recordset`-i doldurmağa imkan verir;
- `Resync()` – artıq alınmış yazılardakı qiymətlərin, mənbədən onları yenidən yükləyərək, yeniləşdirməyi üçün imkan yaradır. Bu halda yeni yazılar görünməyəcək (`Requery()` metodundan fərqli olaraq);
- `Save()` – metodu `Recordset`-i diskdəki faylda saxlanmasına qulluq edir. Bu halda **ADTG (Microsoft Advanced Data TableGram)**, **XML** formatları və ya provayderin öz formatı istifadə edilə bilər. Məsələn:

```
rs.Save "C:\rscustomers.xml", adPersistXML
```

Lazım olduqda, `Open()` metodunun müvafiq parametrlərini göstərərək, yaddaşda saxlanılan `Recordset`-i faylda da bərpa etmək olar;

- `SetAllRowsStatus()` – bu metod `Recordset`-in bütün sətirləri üçün `Status` xassəsinin qiymətlərini dəyişməyə imkan verir;

- **Supports ()** – cari **Recordset**-in nəyi dəstəklədiyini yoxlayır (hansı istiqamətlərdə hərəkət etmək olar, axtarış vasitələri və nişanlar dəstəklənirmi v.s.). Metodun yoxladığı imkanlar provayderin (mənbəyə qoşulmağı təmin edən drayverin) xassələri ilə müəyyən olur.

Əlbəttə, **Recordset** obyektinin həmçinin hadisələr yığımı (toplusu) da nəzərdə tutulub: **EndOfRecordset**, **FetchComplete** və **MoveComplete**. Praktiki işlərdə onlardan az istifadə edirlər deyər - burada həmin xassələrə baxmayacağıq.

9.6 Command obyektı və Parameters kolleksiyası

ADODB.Command *obyektı, VBA-da saxlanılan proseduraların işə salınması, Parameter obyektı və Parameters kolleksiyası, Execute() metodu, Recordset-ə qiymətlərin qayıtması.*

Ən sadə hallarda, məsələn, verilənləri bir başa cədvəldən almaq və dəyişmək imkanı olduqda, **Recordset** obyektindən istifadə etməmək də olar (hətta belə etmək məsləhətdir). Əksər hallarda hətta **Recordset** obyektinin geniş imkanları belə kifayət qədər kömək etmir. Əvvəl qeyd etdiyimiz kimi, daha üstün, etibarlı və əlverişli üsul: verilənlər mənbəyindəki dəyişiklikləri yaddaşda saxlanılan proseduraların köməyi ilə həyata keçirmək. Bəzən serverdə müvəqqəti cədvəllər və digər obyektlərin yaradılmasına ehtiyac yaranır. Bir çox tətbiqi proqramların biznes-məntiqi (məsələn: faizlərin artması, abonent ödənişləri, xüsusi raportların və hesabatların yaradılması v.s.) də həmçinin yaddaşda saxlanılan proseduralarla həyata keçirilir. Buna görə real, praktiki tətbiqi proqramlarda yalnız bir **Recordset** obyektinin tətbiqi ilə bütün problemləri həll etmək mümkün deyil.

SQL əmrlərini serverdə icra etmək üçün (həmçinin saxlanılan proseduraların işə salınmasını, obyektlər yaratmaq üçün DDL əmrlərini icra etmək üçün, rezervuar kopyaları yaratmaq üçün v.s.) **Command** obyektindən istifadə edirlər.

Bu obyektin yaradılması olduqca asandır:

```
Dim cmd As ADODB.Command
Set cmd=CreateObject("ADODB.Command")
```

Növbəti mərhələdə **Command** obyektinə **Connection** qoşulma (bağlanma) obyektini təyin etmək lazımdır. Bu məqsəd üçün **Command.ActiveConnection** xassəsi nəzərdə tutulub. Ona hazır olmuş **Connection** obyektini ötürmək olar. Başqa üsul ilə bu obyektı qeyri aşkar yaratmaq olar: **ActiveConnection** xassəsinin qiyməti kimi qoşulma (bağlanma) sətrini istifadə edərək. Məsləhət görülür ki, həmişə hazır olmuş bağlanma otürülsün: birincisi, bu cür bağlanma üçün daha çox parametrləri qurmaq olur; ikincisi, əgər tətbiqi proqramda bir neçə **Command** obyektı istifadə edilirsə, onda onların hər birisi üçün yalnız bir bağlanmadan istifadə etmək olar (bununla operativ yaddaşın resursuna xeyli qənaət etmək olur). Biz baxdığımız nümunədə əvvəl yaratdığımız **Connection** obyektindən istifadə edirik:

```
cmd.ActiveConnection=cn
```

Qarşımızda duran növbəti məsələ - əmrin tipinin seçilməsidir. Prinsip etibarı ilə, onu seçməmək də olar – ADO modulları özləri verilənlər mənbəyindən əmrin nə olduğunu aydınlaşdıracaq (yaddaşda saxlanılan prosedurasını, yaxud SQL-sorğudurmu v.s.). Daha üstün üsul - əmrin tipinin seçilməsidir: çünki bu halda zamana və sistem resurslarına qənaət edilir və səhvlərin yaranma ehtimalı azalmış olur. Əmrin tipinin seçilməsini **CommandType** ilə təyin edirlər. Bu xassəyə yuxarıda baxmışıq (ona verilən qiymətlər **Recordset** obyektinin **Open()** metodunun **Options** parametrinin qiymətlərinə oxşayır). Məsələn, əgər biz yaddaşda saxlanılan proseduranı işə salmaq üçün əmr veririksə, onda müvafiq əmri aşağıdakı VBA proqram kodu ilə həyata keçirmək olar:

```
cmd.CommandType=adCmdStoredProc
```

Növbəti addım – icra edilən əmrin mətninin təyin edilməsidir. Bu əməliyyatı **CommandText** xassəsi ilə edirlər. Məsələn, əgər yaddaşda saxlanılan **CustOrderHist** prosedurasını işə salmaq istəyiriksə, onda müvafiq VBA proqram kodu aşağıdakı formada yazılmalıdır:

```
cmd.CommandText="CustOrderHist"
```

Daha çox hallarda saxlanılan proseduralar onlara bir və ya bir neçə parametrin ötürülməsini tələb edir. Bu əməliyyat **Parameters** kolleksiyası və **Parameter** obyektləri ilə həyata keçirilir. Parametrləri təyin etmək üçün iki üsuldən istifadə etmək olar:

- serverə sorğu edərək, **Parameter** obyektlərini avtomatik rejimdə yaratmaq (**Parameters** kolleksiyasının **Refresh()** metodu istifadə edilir), sonra isə onlara ad təyin etmək, məsələn, aşağıdakı VBA proqram kodunda olan kimi:

```
cmd.Parameters.Refresh  
cmd.Parameters(1)="ASTEROID"
```

- **Parameter** obyektlərini “əl ilə” yaradaraq, “əl ilə” də onları **Parameters** kolleksiyasına əlavə etmək lazımdır. Qərribə görsənsə də, bu üsul daha qənaətcildir (serverə artıq dəfə müraciət edilməsinə ehtiyac yoxdur). Əvvəlcədən parametrin xassələrinin qiymətlərinin dəqiq aydınlaşdırılması tələb edilir:

```
Dim Prm As ADODB.Parameter  
Set Prm=cmd.CreateParameter("CustomerID", adVarChar, _  
adParamInput, 5, "ASTEROID")  
cmd.Parameters.Append Prm
```

Nəhayət, bu mərhələdən də keçdikdən sonra, əmrin icra edilməsini işə salmaq lazımdır. Bunun üçün **Execute()** metodu istifadə edirlər. Onun çağırılmasının ən sadə üsulu bu cür yazılır:

```
cmd.Execute
```

Bu metod da üç sayda məcburi olmayan parametr qəbul edir (həmin parametrlərin köməyi ilə əlavə parametrlər, çağırılan əmrin tipini vəs. təyin etmək olur).

Bəzi yaddaşa saxlanılan proseduralar və ötürülən əmrlər hansısa başqa qiymətlərin qaytarılmasını tələb etmir (səhv kodundan başqa). Belə hallar əslində az rast gəlir. Buna görə əsas məsələlərdən biri budur – icra edilən əmrdən qaytarılan qiymətlər necə qəbul edilməlidir?

Əgər qaytarılan qiymət qaytarılan parametr kimi registrasiya edilibsə (məsələn, saxlanılan proseduranın təyinatında `OUT` açar sözü ilə nişan edilib), onda bu qiymət `Command` obyektinin parametrinə təyin ediləcək. Ona çatmaq üçün isə `Value` xassəsindən istifadə edirlər.

Digər halda (məsələn, baxdığımız `CustOrderHist` ilə olan nümunədəki kimi), qaytarılan qiymət sadəcə çıxışın axırına atılırsa (baxdığımız halda yazılar yığılı qaytarılır), onda aşağıdakı iki üsuldən istifadə etmək olar:

- **birinci üsul** - bu üsulda `Execute()` metodunun qaytardığı `Recordset` obyektindən istifadə edilməlidir. Bu halda `Recordset`-in yazılarla doldurulması aşağıda verilmiş VBA proqram kodunun əmri ilə yerinə yetirilir:

```
Dim rs2 As ADODB.Recordset
Set rs2=cmd.Execute()
Debug.Print rs2.GetString
```

- **ikinci üsul** – bu üsulda isə `Recordset` obyektinin `Open()` metodundan istifadə etmək lazımdır. Əsas səbəb budur ki, `Open()` metodu parametr kimi `Command` obyektini qəbul edir (bu halda `Connection` obyektini həmin metoda ötürmək olmaz). Aşağıdakı nümunədə VBA proqram kodu ilə həmin üsulun reallaşdırılması göstərilib:

```
Dim rs2 As ADODB.Recordset
Set rs2=CreateObject("ADODB.Recordset")
rs2.Open cmd
Debug.Print rs2.GetString
```

Praktiki tərəfdən dəyərli olan `Command` obyektinin bəzi xassələri və metodları ilə oxucunun tanış olması vacibdir:

- **CommandStream** – xassəsi əmr mətninin bir başa təyin edilməsi əvəzinə (`CommandText` xassəsinin köməyi ilə) qiyməti çıxarış axınından götürülməsini təmin edir (məsələn, mətn faylından və ya başqa proqramdan). Axının buraxıla bilən formatı provayderdən asılıdır (yəni həmin qoşulmanın drayverindən). `CommandStream` və `CommandText` xassələri birgə istifadə edilə bilməz (ikinci xassə, avtomatik olaraq, boş olacaq);
- **CommandTimeout** – xassəsi, səhv haqqında məlumat qaydına qədər mənbədəki əmr icrasında yaranan nəticənin gözləmə müddətini saniyə ilə bildirir;

- **Dialect** – bu xassə provayderdən əmr mətninin araşdırılmasının (İngiliscə “*parsing*”) xüsusiyyətlərini göstərməyə imkan verir;
- **NamedParameters** – xassəsi provayderə parametrlərin ötürülməsində seçim etməyə qulluq edir: 1) parametrlərin adlarının ötürülməsi; 2) qiymətlərin sadəcə sıra ardıcılığına görə parametrlərin ötürülməsi. Bu xassədə iki qiymət seçilə bilər: **True** və ya **False** (standart halda **False** qiyməti qurulmuş olur);
- **Prepared** – bu xassə məhsuldarlığa təsir edə bilər. Əgər o, **True** qiyməti ilə qurulsun, onda əmrin birinci icrasında provayder onun kompilyasiya edilmiş kopyasını yaradacaq. Sonra isə həmin kopya edilmiş versiyanı hər icrada istifadə edəcək. Birinci dəfə əmr daha yavaş icra edilsə də - sonrakı icralarda sürət xeyli arta bilər. Nəzərə alınmalıdır ki, bu cür “*əmrin hazırlanması prosedurasını*” hər provayder dəstəkləmir;
- **State** – xassəsi **Recordset** obyektində istifadə edilmə qaydalarını saxlayaraq, həmin qiymətləri qaytarır və həmin hallarda istifadə edilir (izahı yuxarıda verilib);
- **Cancel ()** - bu xassə ilə hansısa əmrin icrası xeyli uzandıqda, onun icrasını ləğv etmək olur (əgər provayder həmin imkanı dəstəkləyirsə).

10. Microsoft PowerPoint PROQRAMINDA VBA PROQRAMLAŞDIRMASI

Təqdimatların yaradılmasında və istismarında proqram səviyyəli avtomatlaşdırma – VBA-dan PowerPoint ilə işləməyin xüsusiyyətləri. MS PowerPoint VBA proqramlaşdırma mühitinin əsas obyektləri: PowerPoint.Application, PowerPoint.Presentation, PowerPoint.Slide və PowerPoint.Shape

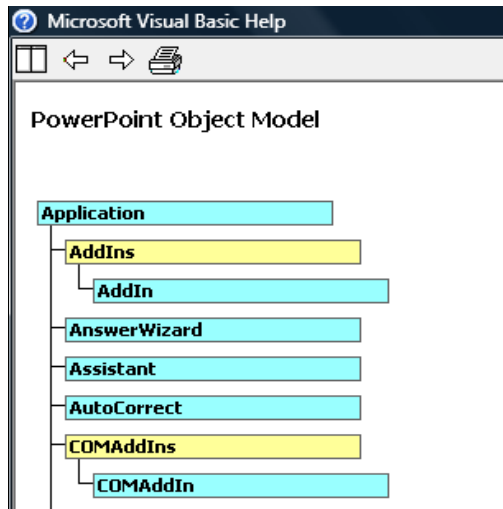
Microsoft Office-in məhz PowerPoint proqramındaki VBA-nın tətbiqindən başlamağımız təsadüfi deyil: çünki hər bir öyrənmə prosesinin əsası - *asandan mürəkkəbə doğru* prinsipidir. MS Office proqramları ailəsində isə ən sadəsi - elektron təqdimatların (İngiliscə *presentation*) yaradılmasına xidmət edən PowerPoint proqramıdır.

Adətən PowerPoint təqdimatlarını mütəxəssislər rəsmi toplantılardakı (toplantılarda, konfranslarda, seminarlarda v.s.) çıxışlarını əyani formada təqdim etmək üçün (məhsulların və xidmətlərin, müxtəlif elmi, texniki ixtiraların və nailiyyətlərin, siyasi mövzularla bağlı raport və hesabatların nümayişində) istifadə edirlər. Slaydların nümayişi səsli və görüntülü məlumatların müşahidəsi ilə aparılması mümkündür deyə, PowerPoint, əsas fəal alət kimi, həmçinin interaktiv dərslərin hazırlanması və aparılmasında istifadə edirlər. PowerPoint-la səsle müşahidə edilən fotoalbomların, səsli diafilmlərin və uşaq oyunlarının yaradılması mümkündür. Son vaxtlar müxtəlif sahələrdə daha bir imkandan istifadə edirlər: PowerPoint-un köməyi ilə, şəkil və video materiallarla müşayiət edilən, səsli kitabları yaradırlar (uşaqlar üçün, fiziki çatışmazlığı olan insanlar üçün v.s.). Bununla belə müəssisələrdə, müxtəlif şirkət və təşkilatlardakı (o cümlədən elm və tədrislə bağlı olanlarda da) gündəlik praktiki işlərdə

PowerPoint-da VBA imkanlarından istifadə edilməsi Word və ya Excel-də olandan xeyli az rast gəlir. Lakin bəzi istisna hallarda, bu cür tələblərin PowerPoint-da yaranması istisna deyil: təqdimatlarda olan verilənlərin sayı və həcmi artdıqca (məsələn, rəqəmsal fotoşəkillərin sayı həmişə həddən artıq çox olanda) həmin an təqdimatın avtomatlaşdırılması məsələsi meydana çıxır. Oxucuya artıq aydındır ki, Visual Basic for Application (VBA) proqram təminatının tətbiqi olmadan bu məsələnin tam miqyaslı həlli ola bilməz. Həmin nöqteyi nəzərdən (yəni VBA ilə PowerPoint-da proqramlaşdırma işlərinin aparılması) praktiki işlərdə daha çox rast gələn halların təsnifatı aşağıda verilib [3]:

- *təqdimatların avtomatlaşdırılmış rejimdə yaradılmasında* (məsələn, hansısa kataloqda verilmiş görüntülər yığılı əsasında);
- *çəkilməmiş sxemlərin nömrələnməsinin avtomatlaşdırılmasında;*
- *təqdimatların emal edilməsində* - görüntülərin formatlarının dəyişdirilməsində, audio-müşəyiətin dəyişdirilməsində v.s.;
- *verilənlər bazası mənbəyindən gələn verilənlər əsasında təqdimatların yaradılmasında;*
- *onlayn mühitində İnternetdə onlayn toplantılar, Web əsaslı seminarlar (webinar) v.s.) təqdimatların yaradılmasında.*

PowerPoint-də obyektlər sisteminin strukturu Şek. 10-da göstərilib (əsas obyektlərin sayı şəkildəkində göstəriləndən daha çoxdur: 50-dən bir qədər artıqdır) :



Şəkil 10 PowerPoint-də obyektlər sisteminin strukturu.

Beləliklə, PowerPoint VBA mühitində proqramlaşdırmanı lazımı səviyyədə aparmaq üçün istifadəçiyə praktiki işlərdə daha tez-tez lazım olan obyektlərlə bir qədər yaxından tanış olaq:

- **Application** - ən yüksək səviyyəli obyektidir (Word və Excel-dəki obyektlərin xassələr və metodlar yığılına çox bənzəyir);

- **Presentations** - kolleksiyası tərkibindəki **Presentation** obyektleri ilə yuxarıdakı **Application** obyektindən bir pillə aşağıda durur (bu obyektler, iyerarxiyada tutduqları yerə görə, Excel-in **Workbook** obyektinə oxşayır);
- **Slides** - kolleksiyası tərkibindəki **Slide** obyektleri ilə **Presentation** obyektinin tərkibində qurulub (nümunəvi analoq kimi Excel-in **Worksheets** obyektlerini gətirmək olar);
- **Shapes** - kolleksiyası tərkibindəki **Shape** obyektleri ilə **Slide** obyektinin tərkibində qurulub. **Shape** obyektı slaydın bütün elementlərini təmsil edir – görüntü, yazı, diaqram, başlıq, cədvəl, avtofiqur v.s. (cəmi 22 element tipi vardır).

Bununla oxucunun nəzərinə çox vacib cəhəti çatdırmaq istəyirik: həmin yuxarıda adları çəkilən obyektlerin ətrafında (**Application, Presentation, Slide** və **Shape**) - PowerPoint-un bütün obyekt modelləri qurulur.

Bu fəsildə PowerPoint-un müxtəlif obyektlerinin xassələri və metodları haqqında ətraflı məlumat verməyimiz məqsədə uyğun sayıla bilməz (çünki buna böyük ehtiyac yoxdur: lazım olduqda, oxucu (istifadəçi) onlar haqqında daha ətraflı məlumatı makrorekorderin köməyi ilə və ya VBA PowerPoint sorğu materiallarından tapa bilər). Bunun əvəzinə, PowerPoint-da VBA ilə proqramlaşdırmanın real həyatda yarana biləcək hallarını, nümunələr gətirərək, illüstrə edəcəyik. Paralel olaraq, kitabdakı həmin addımları oxucu özü də təkrar etsə, onda Office proqramlaşdırılmasının fəal öyrənilməsi uğurlu olacaqdır. Tutaq ki, JPG-şəkillər yığımı əsasında PowerPoint təqdimatı yaradılmalıdır (tutaq ki, bu şəkillər rəqəmsal fotoapparatdan alınıb) və C:\Slides kataloqunda həmin slaydlar yerləşdirilməlidir. Tutaq ki, JPG faylları sıra nömrəsi ilə yerləşdirilib, məsələn, DSCN2440.JPG başlayaraq, DSCN2480.JPG adlı faylda qurtarır. Kataloqdakı faylların sayı dəyişən ola bilər, buna görə kataloqdakı bütün fayllar götürülməlidir. Məqsəd – onlar təqdimatda sıra ilə düzəlməlidir. Çətinlik bundan ibarətdir ki, JPG-faylları müxtəlif ölçüdədir (hündürlük və enlərinə görə) – tələb olunur ki, onlar eyni ölçüdə olsun. Əvvəlcədən bir maraqlı məqamı qeyd edək: necə təəccüblü görünsə də PowerPoint üçün tərtib edilmiş VBA kodunu PowerPoint proqramından deyil, əslində digər Office proqramından, məsələn, Word və ya Excel-dən, işə salınması daha əlverişli olur. Səbəb çox sadədir: VBA kodunun işə salınma anında heç bir başqa aktivləşmiş təqdimata təminat olmayacaq və bununla da “yerinə qoyma” əməliyyatında heç bir səhvə yol verilməyəcək.

Məsələnin həlli addım-addım irəliləmək metodikası ilə bu cür təsvir oluna bilər:

1. Word və ya Excel-də yeni sənəd yaradırıq.
2. Yaradılmış sənədin qrafik interfeysində düymə yaradırıq, və ya zövqdən və seçimdən asılı olaraq, başqa qrafik interfeyslə təmin edirik;
3. Hökmən yaradılan VBA layihəsində iki aşağıda göstərilən obyekt kitabxanasına istinad yaradırıq:

- **Microsoft PowerPoint 11.0 Obyekt Liraya** (C:\Program Filisə\Microsoft Office\Office11\msppt.olb) - PowerPoint proqramının öz obyektləri üçün;
- **Microsoft Scripting Runtime** (C:\Windows\System32\ScrRun.dll) - **FileSystemObject** obyektini və digər fayl sistemi ilə işləməyə qulluq edən obyektlərlə işləmək üçün lazımdır. Bu kitabxana fayl sistemində əsas əməllərin aparılması üçün ən əlverişli alət sayılır və hər bir kompyuterdə vardır (ən köhnə sayılan Windows 2000 əməliyyat sistemində belə).

4. Bu addımda VBA kodunun tərtibinə başlamaq olar - bunun üçün birinci növbədə PowerPoint proqramının özü işə salınmalıdır (digər proqramlarda icra edilən qaydaya uyğun yerinə yetirilir, məsələn, Word, Excel, Access v.s. olan kimi):

```
Dim oApp As New PowerPoint.Application
oApp.Activate
oApp.Visible=msoTrue
```

5. Növbəti addımda – yeni boş olan təqdimat (PowerPoint faylı) yaradırıq:

```
Dim oPresent As PowerPoint.Presentation
Set oPresent=oApp.Presentations.Add()
```

Hələlik hər şey adi qayda ilə gedir (yeni Word sənədinin yaradılmasına bənzəyir). Lakin sonrakı addımlarda PowerPoint proqramının və bizim həll etmək istədiyimiz məsələnin spesifikasiyasına uyğun olan məqamlar başlayacaq.

6. Növbəti əməliyyatda slaydlar yaradılmalıdır (nəzərə alınmalıdır ki, C:\Slides kataloqunda neçə sayda fayl varsa o qədər də slayd yaradılmalıdır). Əlbəttə, VBA kodu yaradıldıqda slaydların yaradılması kodun dövrü əməliyyatları aparılan hissəsində yerləşdirilməlidir. Əvvəlcə **Scripting Runtime** kitabxanasının vasitəsi ilə kataloqdakı faylların kolleksiyasını alırıq (bunun üçün Office vasitələri də kifayət edə bilərdi - bu üsul daha asandır):

```
Dim oFSO As New Scripting.FileSystemObject
Dim oFolder As Scripting.Folder
Dim oFile As Scripting.File
Set oFolder=oFSO.GetFolder("C:\Slides")
For Each oFile In oFolder.Files
'Əgər bu sətirdə, məsələn, bu cür kod sətirini yazsaq:
'MsgBox oFile.Name, onda fayllar yığımının düzülməsinin
'düzgün olmasını görmək olar.
Next
```

7. Nəhayət slaydları yaradırıq. Bunun üçün **Slides** kolleksiyasının **Add()** metodundan istifadə edəcəyik. Bu metod iki mütləq parametri qəbul edir: - təqdimat slaydının nömrəsini (1-dən başlayır) - slaydın şablonunu müəyyən edən **ppSlideLayout** siyahısının (bir neçə onluqdan çox ola bilər) bir qiymətini. Slaydın nömrəsini proqramdakı hesablayıcı ilə təmin etmək olar (bizim üçün ən yaxşı variant – boş şablondur):

```
Dim nCounter As Integer
nCounter=1
```



```

For Each oFile In oFolder.Files
    Set oSlide=oApp.ActivePresentation.
    Slides.Add(nCounter, ppLayoutBlank)
    ...
    nCounter=nCounter+1
Next

```

8. Son addımda isə ən əsas işin icrasına başlayırıq: slayda görüntünü yerləşdiririk və onun ölçülərini tarazlaşdırırıq. Buna görə hər slayd üçün **Shapes** kolleksiyasının **AddPicture ()** metodundan istifadə edəcəyik:

```

oSlide.Shapes.AddPicture FileName:="C:\Slides\" & _
oFile.Name, LinkToFile:=msoFalse, SaveWithDocument:=msoTrue, _
Left:=10, Top:=10, Width:=700, Height:=520

```

Yuxarıdakı VBA kodunda **FileName** parametri – ötürülən faylın adıdır. Dövrü hesablamalarda bu parametr dəyişdiriləcək. **LinkToFile** – parametri görüntü faylının təqdimatın içərisində (**msoFalse**) yerləşməsinə, və ya təqdimatda həmin fayla istinadın yerləşdirilməsini təyin edir. Nəzərə alınmalıdır ki, yerləşdirilən faylların həcmi böyük deyilsə, onda onları təqdimatın (**PPT** faylında) içərisində yerləşdirilməsi əlverişlilik və məhsuldarlıq baxımından daha sərfəlidir. **SaveWithDocument** parametri bizim görüntülərimizi təqdimatla birgə saxlanıb-saxlanmamasını təyin edir (bizim halda – saxlanmalıdır). **Left**, **Top**, **Width** və **Height** parametrləri görüntülərin eyni ölçüdə dəyişdirilməsini təyin edir (bu parametrlər, uyğun olaraq, görüntünün slayd çərçivəsindəki *sol* və *yuxarı* küncünün ölçüsünü, *enini* və *hündürlük* ölçülərini təyin edir). Təbii olaraq, bu punktda tərtib edilən VBA kodu, əsas proqramın dövrü hesablamalar hissəsindəki ardıcıl nöqtələr qoyulan sətirdə yerləşdirilməlidir. Əmal edilən faylların silinməsinə bir dəfəlik icra etmək üçün həmin kodda bir sətiri də əlavə etmək olar:

```
oFile.Delete
```

Beləliklə, yekün VBA kodu aşağıdakı formada yazılacaq:

```

Dim oApp As New PowerPoint.Application
oApp.Activate
oApp.Visible=msoTrue
Dim oPresent As PowerPoint.Presentation
Set oPresent=oApp.Presentations.Add()
Dim oFSO As New Scripting.FileSystemObject
Dim oFolder As Scripting.Folder
Dim oFile As Scripting.File
Set oFolder=oFSO.GetFolder("C:\Slides")
For Each oFile In oFolder.Files
Set oSlide=oApp.ActivePresentation.Slides.
Add(nCounter, ppLayoutBlank)
oSlide.Shapes.AddPicture FileName:=Slides\" & _
oFile.Name, LinkToFile:=msoFalse, SaveWithDocument:=
msoTrue, Left:=10, Top:=10, Width:=700, Height:=520
oFile.Delete
Next

```

Hər bir tərtib edilən proqramın əsas məqsədlərindən biri nəyinsə aydın anlanmasından ibarətdir. Bizim nümunədən bir vacib nəticə çıxarmaq olar: görüntülərin kopyalanması və

yerləşdirilməsi (üstəlik ölçülərinin tənzimlənməsi) kimi rutin xassəli, yorucu “əllə işləmə” metodunu Office proqramlaşdırmasının köməyi ilə və bir neçə sətir VBA kodu yazmaqla, əvəz etmək mümkündür. Nəzərə alınmalıdır ki, PowerPoint-la praktiki işlərdə, məsələn, animasiya, səs müşayiəti, fiqur diapazonları effektləri ilə işlədikdə, böyük çətinliklərin yaranması təbii haldır. VBA sənədlərində lazımı məlumatın tapılması bəzən asan iş olur. Belə hallarda “istiqləndirici məsləhətlərin” alınması üçün makrorekorderdən, fəal olaraq, istifadə etmək lazımdır. Sonra isə avtomatik rejimdə yaradılan VBA kodunu analiz etmək olar. Bəzən burada da trivallıqdan kənar (yeni daha çətin olan) üsullarla rast gəlmək olur. Məsələn, həmin görüntünün yerləşdirilməsi üçün makrokoder aşağıdakı VBA kodunu avtomatik generasiya edir:

```
ActiveWindow.Selection.SlideRange.Shapes.AddPicture ...
```

Belə olduqda məsələnin icrası çətinləşir. Məhz buna görə Microsoft məsləhət görür ki, makrokoderin generasiya etdiyi VBA kodu sonradan yoxlanılsın və düzəlişlər aparılsın.

MS VBA PowerPoint Makrosuna aid bir praktiki nümunə

Bəzən MS VBA PowerPoint proqramında böyük sxemlər çəkildikdə, əvvəlcədən elementlər (bloklar) nömrələnibsə, onda hansı-sa blokun nömrəsi yaddan çıxdıqda onun axtarılması xeyli çətinləşir. Aşağıdakı MS VBA PowerPoint makrosu (proqramı) həmin çətinliyi aradan qaldıra bilər:

```
Sub Change()  
Dim foundText As Object  
Dim oTmpRng As TextRange  
'Dəyişiləsi mətn sahələrinin nömrələri  
For i=66 To 24 Step -1  
    For Each sld In _  
        Application.ActivePresentation.Slides  
        For Each shp In sld.Shapes  
            If shp.HasTextFrame Then  
                Set oTxtRng=shp.TextFrame.TextRange  
                Set oTmpRng=oTxtRng.Replace(FindWhat:=CStr(i), _  
                    ReplaceWhat:=CStr(i + 2), WholeWords:=False)  
            End If  
        Next  
    Next  
Next i  
End Sub
```

11. Microsoft Word PROQRAMINDA VBA PROQRAMLAŞDIRMASI

11.1 WORD-də proqramlaşdırma hansı hallarda lazım olur

Word-də proqramlaşdırma yarana biləcək hallar və VBA proqramlarının Word-də tətbiqi haqqında

Word – təxminən 25 ildən artıqdır ki, MS Office proqramı kimi, dünya miqyasında, istənilən təşkilatda mətn sənədləşməsində istifadə edilən ən populyar proqramdır. Məlumdur ki, ən tanınmış elmi təşkilatlar da Word-də öz elmi hesabatlarını və məqalələrini tərtib edirlər. Dünyada məşhur olan elmi (və qeyri elmi) nəşriyyatlar, əksər hallarda, jurnalların və kitabların hazırlanmasında məhz bu redaktorda yazılmış ilkin materialı müəlliflərdən qəbul edir.

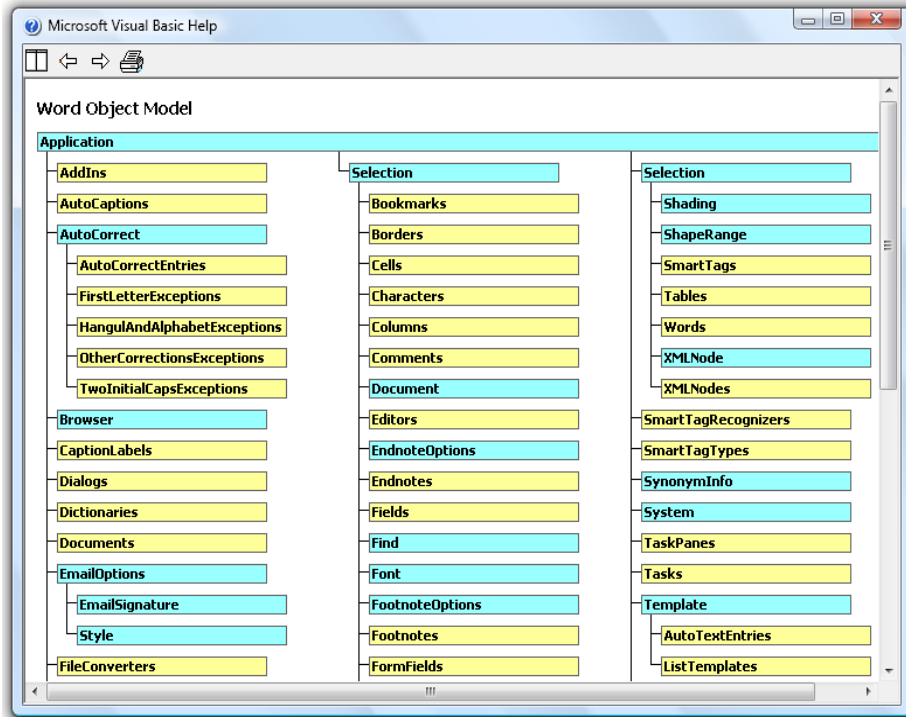
Əgər Word prosessoruna (redaktoruna) proqramlaşdırma baxımından yanaşsaq, onda aşkar olacaq ki, Word – birinci növbədə verilənlər bazası üçün hazırlanan hesabatların yaradılmasını icra edən bir vasitədir [1, 6, 7, 8, 10, 11, 12, 16, 18]. Bu nöqteyi nəzərdən hər hansı hesabat - verilənlər bazasındakı informasiyadan formalaşdırılan istənilən bir sənəddir (məsələn, müqavilə, təlim-təslim aktı, pul vəsaitlərinin daxil olmaları, nağd pul yatırımı haqqında elan, mühasibat üçün sərəncam, faktura v.s.). Əlbəttə, Word-də formalaşdırılan hesabat dedikdə, buraya yekun verilənlər haqqında olan sənədlər də aiddir - periodik hesabatlar (illik, kvartal, aylıq v.s.) və müxtəlif xarakterli məlumatlar (siyahılar və cədvəllər). Bir vacib cəhət də ondan ibarətdir ki, əgər yaradılmış tətbiqi proqram verilənlər bazasına generasiya etdiyi hesabat Microsoft Word-dədirsə, onda bir şeyi bilmək lazımdır ki, son mənzilli istifadəçi üçün həmin hesabatlar olduqca *dostluq* şəraitində (yəni son dərəcədə rahat və əlverişli mühidə) qulluq edəcək. Ola bilsin, başqa proqramlarla fərq bunlar ola bilər: hesabatların formalaşdırılmasının (yaradılmasının) sürətinin bir qədər aşağı olması və proqramlaşdırmanın nisbi çətinliyi. Bəs Word-ün Office proqramlaşdırılmasında *dostluq mühiti* termini nə mənə daşıyır? İş burasındadır ki, bir çox digər analogi məhsullarla işlədikdə, məsələn, **Crystal Reports** və ya **Microsoft Reporting Services** kimi mətn prosessorlarında, sadə bir dəyişiklik aparmaq üçün (məsələn, “Direktor” sözü, tutaq ki, “Direktor Əvəzi” ilə dəyişdirilməsi lazım olduqda) dərhal proqram tərtibatçılarına müraciət olmalıdır. Üstəlik bir müddətdən sonra həmin şəxsi (və ya digərini) direktor vəzifəsində təsdiq etsələr, onda proqram tərtibatçısı yenidən hesabatda düzəlişlər etməlidir. Əgər hesabat, ilkin olaraq, Word redaktorunda yaradılsa, onda bu cür problemlərə rast gəlmək olmur və proqram tərtibatçısına müraciət etmək lazım gəlmir: çünki şirkətlər və müəssisələrdə işləyən istifadəçilərin əksəriyyəti Word-də işləmə qaydalarını bilir. Word-də hesabat hazırlayanların başqa üstünlükləri də var: əgər hesabatın formatı mürəkkəbdirsə (yəni böyük sayda spesifik dizaynı formatlamalar varsa, məsələn, nağd pul yatırımının elanı sənədləşməsində), onda hesabatın şablonunun məhz Word-də hazırlanması daha asan və əlverişli mühidə yaradılır, nəinki, adları yuxarıda çəkilən, **Crystal Reports** və ya **Microsoft Reporting Services** proqramlarında. Səbəbi çox sadədir: bir qayda olaraq, Word-də hesabatlar hazırlandıqda, verilənlər bazasındakı qiymətlər hesabatın şablonuna yerləşdirilir, o isə verilənlər bazasında və ya **.dot** genişlənməsi olan faylda saxlanılır. Proqramlaşdırma ilə bağlı Word-ün başqa bir olduqca vacib cəhəti – müxtəlif formatlı sənədlərlə işləmə mühitinin olmasıdır. Word-ün bu cür cəhətlərinə görə sənədlərin kütləvi emalında (burada proqramlaşdırma meyarı ön plana çıxır) onu əvəz edilməz vasitə saymaq olar. Əksər hallarda böyük müəssisə və şirkətlərdə belə hallar yaranır: şirkətdə sənədləşmə emalı **SharePoint Portal Server** əsasında yaradılıb və buna görə bütün sənədlər eyni formata gətirilməli (Word 2007 və ya, ola bilsin, Word 2003 formatına) və sonra onları həmin SharePoint Portal Server sisteminə yükləmək lazım olacaq. Əsas çətinlik isə bundan ibarətdir ki, server diskindəki kataloqda olan yüzlərlə sənədlər ayrı-ayrı istifadəçilər tərəfindən müxtəlif formatda hazırlanmışdır: məsələn, bir hissəsi Word-ün müxtəlif versiyalarındadır, başqa hissəsi sadəcə mətn fayllarıdır, digərləri isə HTML, XML və ya EML (elektron poçt məlumatları) formatındadır. Əlbəttə, belə hallar yarandıqda, iri miqyaslı şirkətlərdə həmin məsələnin avtomatlaşdırılması (yəni söhbət VBA proqramlaşdırmasının tətbiqindən gedir)

olmadan işin tam miqyaslı yerinə yetirilməsindən söhbət gedə bilməz. Word-də VBA proqramlaşdırmasının daha bir tətbiqi budur: sənədlərin formatlaşdırılması, məsələn, stillərin proqram səviyyəsində tətbiqi, bir çox sənəddə eyni zamanda mətnlərdəki hissələrin axtarılması və əvəz edilməsi, sənədin strukturu ilə işləmə v.s. Adətən buna oxşar məsələlərlə böyük nəşriyyatlarda əlyazmaların hazırlanmasında rast gəlmək olur.

11.2 Word-də proqramlaşdırmağa giriş, Word-dün obyekt modelinə (Word Object Model) ümumi baxış

Microsoft Word-ün obyekt modeli, Application, Document, Selection, Range, Bookmark obyektləri

Word obyektlərinin ümumi strukturu şəkl. 11.1-də göstərilib (MS VBA Word sorğu sənədlərindən götürülüb, şəkildə obyektlərin çox az hissəsi görsənir, əslində obyektlərin sayı yüzlərlədir). Əlbəttə, şəkl. 10-dakı PowerPoint proqramı obyektlərinin sayı ilə müqayisədə buradakı obyektlərin sayı dəfələrlə çoxdur (obyektlərin sayı yüzlərlədir). Oxucu bundan qorxmamalıdır – yüzlərlə sayda olan obyektlərin əksəriyyəti bizim praktiki işimizdə lazım olmayacaq (lazım olduqda isə, əvvəl qeyd etdiyimiz kimi, MS VBA Word sorğu matriallarından lazımı məlumatın İngilis dilində toplanması bir o qədər də çətin iş deyil). Praktiki işlərdə (proqramlaşdırma məsələlərinin həllində) həmin yüzlərlə Word obyektlərin içərisindən yalnız beşinin mükəmməl bilməsi (əlbəttə, bütün xassələri, metodları və kolleksiyaları ilə birgə) adi yeni başlayan VBA tərbiətçisi üçün vacibdir və kifayətdir: **Application** obyekt; **Document** obyekt (Documents kolleksiyası ilə); **Selection** obyekt; **Range** obyekt; **Bookmark** obyekt (Bookmarks kolleksiyası ilə).



Şəkil 11.1 Word obyektlərinin ümumi strukturu (Word sənədinin əsas obyektləri).

Aşağıda bu adları sadalanan ən vacib obyektlər haqqında ətraflı məlumat verəcəyik. Hər bir obyekt üçün əvvəlcə ümumi hallara baxılacaq, məsələn, onların hansı hallarda tətbiq olunması və onların köməyi ilə bu və ya digər əməliyyatı necə yerinə yetirmək olar. Nəzərə alınsa ki, Word-də ən çox rast gələn məsələ - sənədin yaradılması (hansısa şablon əsasında) və sənədin təyin olmuş yerinə lazımı məlumatın yazılması kimi məsələlərdən ibarətdir, onda məhz həmin məsələlərin həllində uyğun olan obyektlərin istifadəsinə daha çox diqqət verəcəyik. Bundan əlavə, hər bir obyekt üçün onların ən vacib xassələri, metodları və hadisələrinin qısa təhlilini verəcəyik. Yada salırıq ki, Office proqramlaşdırmasını mükəmməl öyrənmək istəyən oxucular üçün aşağıda verilən məlumatların fəal formada öyrənilməsi daha yaxşı nəticələrin alınmasının qarantıdır (yəni, lazım olduqda, kompyuter qarşılarında olmalıdır).

11.3 Application obyekti

11.3.1 Application Obyekti ilə işləmə qaydaları

Word.Application obyekti, Word nüsxəsinin işə salınması, Word.Application obyektinin hadisələri

Application obyekti - Microsoft Word-ün sənədinin özüdür. Word-ün digər obyektləri həmin obyektin içərisində "yerləşdirilib". Bu obyektin yaradılması – Word-ün istifadəçi kompyuterində işə salınması deməkdir. Adətən əksər hallarda, elə bu hadisə daha çox lazım olur (əgər istifadəçi, məsələn, Access-in içərisindən öz sənədini Word formatında yaratmaq istəyirsə). Yaddan çıxarılmamalıdır ki, istifadəçi Word-ü Office-in digər proqramından işə salırsa, onda hökmən yaradılan layihəyə **Microsoft Word 11.0 Object Library** kitabxanasına istinadı əlavə etməlidir. Word sənədinin işə salınma kodu olduqca sadədir:

```
Dim oWord As New Word.Application
```

Bu kod başqa Office proqramından icra edildikdə nəyinsə baş verdiyini istifadəçi hətta hiss etməyəcək (görməyəcək). Səbəblər çox sadədir:

- Standart halda Word gizli pəncərədən işə salınır;
- əgər həmin pəncərədə heç bir sənəd açılmayıbsa, onda sənəd dərhal, onu yaradan prosedura işini bitirən kimi, bağlanır.

Word-ü görünən etmək üçün sadə bir kod yığılmalıdır:

```
oWord.Visible=True
```

Bəzən istifadəçilər belə bir sual qarşısında dururlar: həmin Word sənədini gözə görünən etmək lazımdır mı? Bəzi peşəkar VBA proqramçıları bu sənədin görünməz olmağına önəm verirlər: lazımı sənədin yaradılması üçün Word-ün gizli pəncərədən işləməsi daha əlverişli hal kimi baxılır (istifadəçiyə lazım olduqda istifadəçinin özü həmin sənədi açacaqdır). Lakin istifadəçilərin əksər hissəsi Word-ün görünən olmasını daha rahat və əlverişli sayırlar: birincisi – sənəd yaradıldıqda, hər bir yarana biləcək problem dərhal gözə çarpır, ikincisi – psixoloji tərəfdən istifadəçilər Word-ün

göz qarşısında açılıb işə salınması və avtomatik rejimdə sətirlərin çap edilməsi daha çox xoşa gəlir və etibarlılıq (güvenc) mühiti yaranmış olur.

Word-lə gizli pəncərədən işləyərkən, tələb olan əməliyyatlar bitdikdə, sənəd bağlanmalıdır (digər halda o, yalnız **Task Manager** pəncərəsində görünən olaraq, maşının operativ yaddaşında qalacaq). Word-ü bağlamaq üçün onun **Quit()** metodu çağırılmalıdır.

Word-ün öz-özünə bağlanmasının qarşısını almaq üçün onun içərisində yeni sənədin yaradılması kifayətdir. Bu haqda daha ətraflı növbəti hissədə danışılacaq. Yeni Word sənədinin ən sadə yaradılma üsulu isə aşağıda verilən VBA kodunun icra edilməsi ilə həyata keçirilə bilər:

```
Dim oWord As New Word.Application
oWord.Visible=True
oWord.Documents.Add
```

Əgər Word artıq açılıbsa, onda ona olan istinadı aşağıdakı kod sətiri ilə icra etmək olar:

```
Set oWord=GetObject(,"Word.Application")
```

Praktiki işlərdə xüsusi halların istisna olmağı ilə (məsələn, OLE obyektlərinin aktivləşdirilməsi) bu cür yaxınlaşmanın böyük üstünlüyü olmur. Əksinə, istifadəçinin mövcud olan nüsxədə yaratdığı sənədin qəfildən (bilmədən) xarablanması və ya mövcud olan Word nüsxəsinin istifadəçi sənədlərinin yaddaşda saxlamadan bağlanması riski əmələ gələ bilər. Buna görə ən yaxşı variant – yeni Word nüsxəsinin yaradılmasıdır.

Digər halda, əgər VBA kodu Word-də icra edilirsə (yəni Word artıq işə salınıb), onda **Application** obyektinin yaradılmasına ehtiyac qalmır. Bu halda o, istənilən anda əlçatan olacaq (buna əmin olmaq üçün VBA kod redaktorunda **Application** sözünün yazıb sonunda bir nöqtəni əlavə etmək kifayətdir). Bundan əlavə, əgər bu və ya digər xassənin (və ya metodun) hansı obyektə aid olmağı göstərilməyibsə, onda Word-əki VBA kompilyatoru, avtomatik olaraq, həmin xassənin (və ya metodun) **Application** obyektinə aid olduğunu təyin edir. Buna görə aşağıdakı iki VBA kodları eyni güclüdür:

```
Application.Selection.TypeText "Mənim mətnim"
```

və

```
Selection.TypeText "Mənim mətnim"
```

Word-ə **Application** obyektinin daha bir vacib xassəsi budur ki, bu obyekt üçün çox sayda istifadə də əlverişli olan hadisələr nəzərdə tutulub (sənədin açılması, Word-dən çıxış, mausun sağ düyməsi ilə şıqqıltının icra edilməsi, sənədin dəyişdirilməsi, sənədin çapı, sənədin yaddaşda saxlanması v.s.). Standart vəziyyətdə bütün bu hadisələr görünməz olur. Onların görünən olması üçün formaya aid (yalnız formanın – modulun yox!) VBA proqram kodunun **Declaration** hissəsində gərək **Application** obyektini **WithEvents** açar sözü ilə elan edilsin, məsələn, aşağıdakı nümunədəki kimi:

```
Public WithEvents App As Word.Application
```

Belə olduqda, obyektlər siyahısında yeni **App** (yəni, əslində **Application**) obyektini əmələ gələcək. Həmin obyekt üçün işə hadisələrin seçilməsini və VBA koduna hadisə yönli proseduraların əlavə edilməsi adi qaydada formalarda və idarəetmə elementlərində edilən qaydaya uyğun aparılacaq.

11.3.2 Application obyektinin xassələri, metodları və hadisələri

Application *obyektini, onun xassələri metodları və hadisələri*

Aşağıda, arayış üçün, **Application** obyektinin ən vacib olan xassələri, metodları və hadisələri haqqında məlumat verilib.

- **ActiveDocument** – fəal sənədin obyektini qaytarır (baxılan halda Word nüsxəsini). Çox fəal olaraq istifadə edilir, adətən **Application** obyektinin adı çəkilmədən məsələn:

`ActiveDocument.Save`

Bu xassə yalnız oxuma üçün əlçatandır, buna görə hansısa sənədi fəal etməkdən ötrü onun obyektini üçün **Activate** () metodu çağırılmalıdır.

- **ActivePrinter** – proqramın işləmə məqamında fəal printerin alınmasını və ya sazlanmasını təyin edir. Həmçinin, bir çox hallarda tətbiqi proqramın işinin nəticəsini müəyyən bir şəbəkə printerində çap edilməsi lazım olanda istifadə edilir.
- **AutomationSecurity** - faylların proqram səviyyəsində açılmasında təhlükəsizliyi təyin edir. Standart halda **msoAutomationSecurityLow** qiyməti qurulmuş olur – yəni proqram işə salınmış makroslarla açılır. Həmçinin **msoAutomationSecurityForceDisable** qiyməti istifadə edilə bilər – makrosların icrasını dayandırmaq və **msoAutomationSecurityByUI** qiyməti onda qrafik interfeysdə qurulmuş olur.
- **BackgroundPrintingStatus** – növbədə çapa neçə Word tapşırığının olmasını təyin edir.
- **Browser** – eyni adlı obyektin qaytardığı xassədir (şaquli diyirlənmə zolağının altındakı üç sayda düymə yığımını qaytarır). Proqramçılıq nöqtəyi nəzərdən, onun **Target** xassəsi daha maraqlıdır 12 sayda qiymət qəbul edə bilər: şərh, qeydlər, cədvəl, şəkil, başlıq, səhifə v.s.). Bu obyektin **Next** () və **Previous** () metodları ilə həmin elementlər arasında hərəkət etmək olur.
- **Build** - Word sənədinin versiyası haqqında tam məlumatı qaytarır. Əgər tərtib edilən tətbiqi proqram Word-ün yalnız müəyyən versiyaları üçün nəzərdə tutulubsa, onda yoxlama işlərinin aparılmasında bu xassənin istifadəsi çox xeyirli ola bilər.
- **CapsLock** – klaviaturada **CapsLock** rejiminin qurulmasını yoxlayır. Həmin rejimi bu xassə ilə dəyişmək olmaz – bunun üçün başqa vasitələr vardır. Analoji olaraq, **NumLock** xassəsi işləyir.

- **Caption** – pəncərənin başlığında standart **Microsoft Word** sözünü başqa mətnlə dəyişdirilməyini təmin edir, məsələn: “**Mənim Tətbiqi proqramım**” mətninə.
- **CheckLanguage** – avtomatik rejimdə Word-də yığılan mətnin dilini təyin etməsini yoxlayır. Standart halda yoxlama işini icra edir (bu xassə ilə dil rejimini də dəyişdirmək olar).
- **COMAddIns** – bu xassə ilə **COM (Komponent Object Model** – yeni komponentlərin obyekt modelləri) texnologiyası əsasında Word-ə qurulmuş **COM Add ins** kolleksiyasına istinad almaq olur. Hansısa verilmiş proqrama müraciət etmədən qabaq, yoxlamaların aparılmasında çox faydalı xassədir.
- **CustomizationContext** – menyuda, alətlər panelində və klaviatura kombinasiyalarında daxil edilən dəyişikliklərin hansı şablona və ya sənədə əlçatan olduğunu işarə edir (göstərir). Məsələn aşağıdakı kodda olan kimi:

```
CustomizationContext=NormalTemplate
```

işarə edir ki. həmin andan başlayaraq, bütün daxil edilən dəyişiklər **Normal.Dot** şablonunda saxlanacaq (buna görə də bütün sənədlərdə əlçatan olacaq).

- **Dialogs** – xassəsi **Dialogs** kolleksiyasını qaytarır (Word-ün bütün formada olan dialoq pəncərələrini). Word-ün bu obyekt modelini (“budağı” ilə birgə) istifadə edərək, istifadəçi Word-ün yüzlərlə sayda olan istənilən dialoq pəncərəsini açmağa bilər və həmin pəncərədəki parametrləri təyin edə bilər. Nəzərə alınmalıdır ki, bu “budaq” çox pis sənədləşib: dialoq pəncərələrinin obyektlərindən istifadə etmək istədikdə, əslində biz lazım olan xassəni və onun qiymətini təyin edə bilərik. Bunun üçün eksperimentlər aparmaqdan başqa bir imkan qalmır (makrorekorder və **Locals** pəncərəsinin köməyi ilə). Peşəkar VBA proqramçıları bu cür “tədqiqat işinin” aparılmasını məhsuldar saymayaraq, öz VBA formasını yaradırlar və lazımı əməliyyatın aparılmasına nail olurlar. Məsələn, aşağıdakı VBA kodunda faylın açılmasını icra edən dialoq pəncərəsinin istifadəsi yerinə yetirilir:

```
Dim oDlg As Dialog
Set oDlg=Application.Dialogs(wdDialogFileOpen)
If oDlg.Display=-1 Then
MsgBox "Siz faylı seçdiniz: " & _
Application.Options.DefaultFilePath(wdCurrentFolderPath) & _
"\ " & oDlg.Name
End If
```

Fayllarla işləmək üçün nəzərdə tutulan dialoq pəncərələri üçün **Application** obyektində ayrıca **FileDialog** xassəsi vardır (eyni adlı obyektə qaytarır).

- **DefaultSaveFormat** – Word fayllarının standart halda saxlanma formatını təyin edir (istifadəçinin dialoq pəncərələrində **Save As** kimi seçdiyi format).
- **DisplayAlerts** – çox vacib xassədir. VBA tətbiqi proqramları və makroslar işlədikdə, səhvlərin və dialoq pəncərələrin çıxarılmasının qarşısını alır. Bir çox hallarda onsuz

keçinmək olmur: xüsusilə, proqram işlədikdə, nəyinsə silinməsi və ya proqramın, yaddaşda saxlanmadan, bağlanması lazım olduqda.

- **DisplayAutoCompleteTips** – mətnin avtomatik tamamlanması üçün qoşmaq və ya söndürmək dialoq məsləhətlərini qaytarır (daha çox söndürmək halları lazım olur).
- **Documents** – sənədlər kolleksiyasını qaytarır (çox vacib xassələrdəndir). Bu kolleksiya haqqında daha ətraflı aşağıda məlumat veriləcək.
- **EmailOptions** – bu xassə mürəkkəb strukturlu **EmailOptions** obyektini qaytarır (Word-ün Outlook-da poçt redaktoru kimi sazlanmasında istifadə edilir).
- **EnableCancelKey** – makrosun klaviaturadakı **<Ctrl>+<Break>** düymələr kombinasiyası ilə icrasının dayandırılmasını təyin edir. Bu xassə üçün **WdCancelDisabled** qiyməti qurulsa, onda sonsuz hesablama dövrünü icra edən makrosu yalnız Word-lə birgə bağlamaq olar (**Task Manager** ilə).
- **FeatureInstall** – Office-in yüklənməmiş komponentlərinin, məcburi olaraq, yüklənməsinin qarşısını alır (bunun üçün **msoFeatureInstallNone** qurulmalıdır).
- **FileDialog** – bu xassə **FileDialog** obyektini qaytarır (yəni faylın kataloqun seçilməsini, faylın açılmasını və ya yaddaşda saxlanmasını təmin edən pəncərəni). Bu pəncərəni açmaq üçün həmin obyektin **Show()** metodundan istifadə edilməlidir.
- **FileSearch** – təyin edilmiş parametrlərə görə faylların axtarılmasında istifadə edilən **FileSearch** obyektini qaytarır.
- **International** – daha bir vacib olan xassədir: regional sazlanma haqqında məlumat qaytarır (tarix, zaman, valyuta, rəqəmlərin formatı, Word-ün lokal versiyası v.s.).
- **IsValidObject** – müxtəlif yoxlamalar üçün olduqca əlverişli vasitədir (məsələn, müraciət ediləsi obyektin mövcud olmağını yoxlayır – istifadəçi tərəfindən obyektin sənəddən silinməsi nəticəsində yaranan səhvlərdən qorunmağa imkan yaradır).
- **KeyBindings** – bir çox hallarda olduqca əlverişli xassədir: **KeyBindings** kolleksiyasının (klaviatura bağlantısının qaytarılmasını təmin edir). Sadə dillə desək, bu obyektin (və onun alt obyektlərinin) köməyi ilə istifadəçi Word-ün istənilən əmrini və ya istənilən makrosunu klaviaturanın istənilən düymələr kombinasiyasına (o cümlədən xidməti təyinatlı düymələr kombinasiyasına belə, məsələn, **<Alt> + <F4>**) təyin edə bilər. Əməllərin ümumi ardıcılıq qaydası aşağıdakı kimidir:
 1. **CustomizationContext** parametrini təyin edirik – yeni dəyişdirmələrin saxlanılması təyin edilir. Variantlar: şablon **normal.dot**, cari sənəd və ya cari sənədə bağlanmış şablon;
 2. **Application.BuildKeyCode()** metodu ilə klaviatura kombinasiyasının rəqəmsal kodunu təyin edirik;

3. **KeyBindings.Add()** metodu ilə yeni təyinatı əlavə edirik (bununla belə bütün tələb olan parametrləri təyin edirik). Məsələn, **<Alt> + <D>** düymələri basıldıqda, bütün sənədlərdə **DataLoad()** makrosu işə salınsın deyə, aşağıdakı VBA proqram kodunu istifadə etmək olar:

```
CustomizationContext=NormalTemplate  
Application.KeyBindings.Add wdKeyCategoryMacro, _  
"Normal.NewMacros.DataLoad", BuildKeyCode(wdKeyAlt, wdKeyD)
```

- **Language** - Word-ün hansı lokal versiyasının istifadəçi kompyuterində yüklənməsini təyin edir (daha dəqiq desək, istifadəçi interfeysinin dilini təyin edir). Daha dəqiq məlumatı **LanguageSettings** xassəsi verir (köməkçinin dilinin yüklənmə proqramı v.s. haqqında).
- **MacroContainer** – VBA proqramçıları üçün çox vacib xassədir: proqram kodunun icrası zamanında onun haradan işə salınmasını təyin edir (adətən – iki variant yoxlanılır - **normal.dot** şablonundan və ya cari sənəddən).
- **NewDocument** – yeni Word sənədini yaradılması üçün daha bir vasitəsidir: **NewDocument** obyektini qaytarır. Yeni sənədin yaradılması üçün **Application.NewDocument.Add()** metodundan istifadə edirlər.
- **NormalTemplate** - ən vacib xassədir! **normal.dot** şablonunu təmsil edən **Template** obyektinə (onda dəyişiklik aparmaq üçün) istinadın alınmasını təmin edir.
- **Option** - böyük sayda xassələri olan **Option** obyektini qaytarır. Bu obyektə istifadə edərək, proqram üsullarının köməyi ilə qrafik ekranda **Tools** → **Customize** menyusundan əlçatan olan bütün əlavə qurmaların qiymətlərini sazlamaq olur.
- **Path** – diskdəki Word proqram fayllarına olan yol haqqında məlumatı qaytarır.
- **PrintPreview** – cari sənədin əvvəlcədən baxma rejiminə keçməyi təmin edir və ya bu rejimdə olmağı yoxlayır. İstifadəçiyə sənədin nümayişində və ya özəl çap prosedurasının reallaşmasında kömək edir.
- **ScreenUpdating** – bu xassə ilə ekranın yenidən çəkilməsinin qarşısı alınır (və ya onun qiymətini **False** qurmaq olur). Adətən ekrana hansısa məlumatı (və ya nəyi isə) çıxaran proseduraların işini sürətləndirir.
- **Selection** – çox vacib olan xassədir! **Selection** obyektini qaytarır – daha sadə desək, *yerləşdirmə işarəsinin yerini qaytarır* (bu xassə haqqında daha ətraflı növbəti hissələrdə danışacağıq).
- **ShowStartupDialog** – Word işə salındıqda, **Task Pane** obyektinin göstərilib-göstərilməməyini təyin edir. Daha çox göstərilənin söndürülməsində istifadə edilir. Burada **Show**, **Startup** və **Dialog** xassələri də vardır (onların qiymətləri adlarından çox aydın bilinir).

- **SpecialMode** - Word-ün xüsusi kopyalama və yerləşdirmə rejimində olmasını yoxlayır (bu rejimə keçmək üçün mətn seçilərək, <F2> və ya <Shift> + <F2> düymələri basılır, sonra isə kursurun yerini dəyişib <Enter> düyməsi basılmalıdır).
- **StartUpPath** – avtomatik işə salma kataloquna yol ilə bağlı baxmaq və ya təyin etmək əməllərinin yerinə yetirilməsini təmin edir. Həmin kataloqda olan şablonları və qurulmuş əlavələri Word özü açıldıqda, avtomatik olaraq, onları da açır. Standart halda avtomatik işə salma kataloqu istifadəçinin profilində yerləşdirilmiş olur. Onun tərəfinə gedən yol bu cür görünür: `application\data\microsoft\word\startup`
- **StatusBar** – daha bir xeyirli olan xassədir: mətni vəziyyət sətrinə (**Status Bar**) çıxarmağa imkan verir, yəni tətbiqi proqramın aşağı tərəfində yerləşən sətir (burada səhifələr, sütunlar, dil, işləmə rejimləri v.s. haqqında məlumat yerləşir).
- **System** – xassəsi, eyni adlı olan, **System** obyektini qaytarır (əməliyyat sistemi haqqında məlumat almaq üçün: regional sazlanmalar, maus kursurunun tipi, ekran imkanları, prosessorun tipi v.s. Həmçinin şəbəkə disklerini qoşmağa və **Microsoft System Information** proqramının işə salınmasını təmin edir.
- **Tasks** - – xassəsi, eyni adlı olan, **Tasks** kolleksiyasını obyektləri ilə birgə qaytarır (sistemdə işləyən bütün prosessorlar haqqında məlumat verir). Həmçinin həmin obyektlərlə sistemdə işləyən proqramları tapıb onlar üzərində bəzi əməllər aparılmasına imkan yaradır (görünən və ya görünməyən etmək, aktivləşdirmək, bağlamaq, onun pəncərəsinə Windows xəbərini ötürmək - **Windows API** ilə işləməyə oxşar qaydada v.s.). Təcrübəli tətibatçılar, xaricdə yerləşən proqramlarla işlədikdə, həmin obyektlərdən fəal olaraq, istifadə edirlər. Xaricdə yerləşən proqramları xüsusi obyekt `shell` ilə işə salınması daha əlverişlidir (bu haqda aşağıda daha ətraflı məlumat veriləcək).
- **UserControl** – çox vacib olan xassələrdəndir (bu xassə Excel-də də var): Word-ün necə işə salınmasını təyin etməyə imkan yaradır – istifadəçi proqramı “əl ilə” işə salıb yaxud əlavə proqram üsulu ilə işə salınıb. Bunun əsasında isə qərar vermək olar: proqram üsulu ilə proqramı bağlamaq.
- **UserInitials** və **UserName** – istifadəçinin adı və inisialları haqqında məlumatı qaytarır (inisiallar düzəlişlərdə, ad isə sənədin xassələrində istifadə edilir).
- **VBE** – xassəsi hərfi mənada **Visual Basic Editor** sözlərinin qısalmasıdır və buna görə **VBA** redaktorunu **VBE** adlı obyekt kimi qaytarır.
- **Version** – xassəsi Word-ün versiyası haqqında məlumat qaytarır (yuxarıdakı **Build** xassəsi ilə müqayisədə daha az məlumat qaytarır, məsələn, Word 2003 versiyasının qiyməti 11.0 kimi qaytarılır).
- **Visible** - bu xassə Microsoft Word-ün pəncərəsini gizlədir (Word çox keyfiyyətli halda yox olur: iş masasından və həmçinin məsələlər panelindən).

- **Windows** - eyni adlı **Windows** kolleksiyası haqqında məlumat qaytarır (əslində Word sənədlərinin pəncərə obyektləri haqqında olan məlumatı qaytarır). Bu kolleksiya da həmçinin proqramlaşdırmada çox istifadə edilir.
- **WindowState** - xassəsi Word pəncərələri üzərində bağlamaq, açmaq və bərpa etmək kimi əməllərin yerinə yetirilməsini təmin edir.

Application obyektinin ən vacib metodları aşağıdakı izahlı siyahıda verilib:

- **Activate()** – cari sənədi olan Word pəncərəsini aktivləşdirir. Adətən müəyyən sənədin aktivləşdirilməsi tez-tez lazım olur, buna görə bu metod **Document** obyektini üçün istifadə edilir.
- **BuildKeyCode()** – Word-də klaviatura kombinasiyasının unikal nömrəsinin bilinməsinə qulluq edir (bu metodun tətbiqi haqqında olan nümunə bir az yuxarıda **Application.KeyBindings** xassəsinə baxıldıqda verilib).
- **ChangeFileOpenDirectory()** – bu metod standart halda Word-ün sənədlərlə işləmək üçün açdığı kataloqun dəyişdirilməsinə imkan yaradır (standart halda – “**My documents**” kataloqudur).
- **CheckGrammar()** və **CheckSpelling()** – metodu ilə ötürülən işarəli qiymətlərin qrammatika və orfoqrafiyasının yoxlanmasına qulluq edir (daha çox bu metod **Document** və **Range** obyektlərində istifadə edilir).
- **CleanString()** – olduqca xeyirli metoddur: ötürülən işarəli qiymətləri Word-ün xüsusi işarələrindən “təmizləməyə” və sonra onları mətnə çevirməyə imkan yaradır (**Bloknot**-da yığılmış formaya bənzəyir).
- **DefaultWebOptions()** – metodu eyni adlı obyektini qaytarır (bu obyektin köməyi ilə Word sənədinin HTML formatında yaddaşda saxlanması bir çox xassələrini təyin etmək olur: kodlaşmanı, görüntülərlə işləməni, CSS (Cascading Style Sheets), hansı brauzerlərlə uyğunlaşmasının təmin edilməsini v.s).
- **GoBack()** – bu metod sənəddə redaktə etmənin axırını yerinə keçirilməyini təmin edir. Word sənədi ilə üç son redaktə nöqtəsini saxlayır, buna görə Word-də son sənədin açılmasını və redaktə zamanı istifadəçinin dayandığı nöqtəyə aşağıdakı proqram kodu ilə çox asan nail olmaq olar:

```
RecentFiles(1).Open
Application.GoBack
```

- **GoForward()** - yadda saxlanılan nöqtələrə doğru keçid edilməsini təmin edir.
- **Keyboard()** – olduqca xeyirli metoddur: proqram üsulu ilə Word-də klaviatura paylaşmasının (dilə keçmə kombinasiyalarını) keçidini avtomatlaşdırmağa imkan verir. Məsələn, proqram üsulu ilə Rus dilinə keçid aşağıda verilən kodla həyata keçirilə bilər:

Application.Keyboard 1049

İngilis dilinə isə -

Application.Keyboard 1033

əgər bu metoda heç nə ötürülməsə, onda o, klaviaturanın cari dil vəziyyətini (paylanmasını) qaytaracaq.

- **KeyString()** – metodu yuxarıdakı **BuildKeyCode** metodunun (klaviatura kombinasiyasının unikal identifikatorunu qaytaran) əksidir: verilmiş unikal identifikatorun klaviatura kombinasiyasını qaytarır.
- **ListCommands()** – yeni sənəd yaradaraq, orada Word-dəki əmrlər və klaviatura kombinasiyaları (standart və istifadəçi tərəfindən təyin edilmiş) haqqında cədvəl formasında sorğu məlumatını verir.
- **OnTime()** – çox xeyirli metoddur: istifadəçi tərəfindən təyin edilmiş vaxtda makrosun Word-də icrasını yerinə yetirir. Word-də eyni zamanda yalnız bir taymer işləyə bilər. Bu metodun köməyi ilə avtomatik rejimdə resurs tutumlu əməliyyatları aparmaq olur.
- **OrganizerCopy()** – daha bir xeyirli metoddur: makrosun, alətlər panelinin, avtomətnin (autotext) və ya sənədlərin stilinin yazılmasını həyata keçirir. **Application** obyektini üçün həmçinin **OrganizerDelete()** və **OrganizerRename()** kimi metodlar (adlarından nə etdikləri məlumdur) də nəzərdə tutulub.
- **PrintOut()** – bu metod böyük sayda parametrlər qəbul edir və bütün sənədin və ya onun bir hissəsinin çap edilməsinə imkan yaradır.
- **Quit()** – metodu daha çox istifadə edilən metodlardandır: Word-ün bağlanmasını sənədlərin yaddaşda saxlanma və ya saxlanmama şərti ilə icra edir.
- **Repeat()** – istifadəçi təyin etdiyi sayda son dəfə icra edilən əmri təkrar edir.
- **ResetIgnoreAll()** – mətnin orfoqrafik yoxlamada “yoxlamasız” nişan edilən bütün hissələrdən nişanın çıxarılmasını təmin edir.
- **Run()** – daha bir vacib olan metoddur: proseduranın (və ya makrosun) açılmış şablondan (sənəddən) işə salınmasını və ona parametrlərin ötürülməsini yerinə yetirir.
- **ScreenRefresh()** - proqramın pəncərəsini yeniləşdirir. Adətən **ScreenUpdating** xassəsi tərəfindən avtomatik yeniləşmə söndürüldükdən sonra istifadə edilir.
- **ShowClipboard()** - Word-ün dəyişdirmə buferinin göstərilməsini təmin edir (istifadəçinin bir neçə buferlə işlədiyində tətbiq edilir).

Digər metodlar DDE (Dynamic Data Exchange) protokolları ilə işləmək üçün nəzərdə tutulub yaxud müxtəlif ölçü vahidlərinin çevirməsi v.s. ilə bağlıdır (praktiki işlərdə bir o qədər də vacib rol oynamırlar).

Application obyektinin proqramlaşdırma baxımından bir sıra vacib hadisələri vardır: sənədlərdə (açılmaq, bağlanmaq, yaddaşda saxlanılmaq, çap edilmək, proqramdan çıxmaq v.s.), mausun düymələrinin şıqqıldaması, aktivləşdirilmə v.s. Bir çox hadisələrin təyinatını adlarından apaydın anlamaq asan olur, buna görə onların burada ətraflı baxılması məqsədə uyğun deyil. Lakin bir məqamı yenidən qeyd etməyimizi vacib sayırıq: **Application** obyektinin hadisələri VBA-nın redaktorunda göstərilir. Onların görünməsi üçün gerek proqram kodunun **Declaration** hissəsində aşağıdakı VBA kodu yerləşdirilsin:

```
Public WithEvents App As Word.Application
```

Yalnız bu halda VBA kod redaktorun pəncərəsindəki obyektlər siyahısında bütün tələb olan hadisələri ilə **App** obyektinə əmələ gələcək.

11.4 Documents kolleksiyası və Document obyektləri

11.4.1 Documents kolleksiyası ilə işləmə qaydaları

Documents kolleksiyası, Word.Document obyektinə, VBA vasitələri ilə sənədlərin şablonlardan yaradılması, sənədlər tipinin dəyişdirilməsi

Word-ün obyekt modelində **Documents** kolleksiyası (həmçinin proqramlarda istifadə etmə məntiqinə görə də) **Application** obyektindən bir pillə aşağıda durur. Bununla belə Word proqramlaşdırılmasında **Documents** kolleksiyası və **Document** obyektləri olamadan işləmək mümkün deyil. Adətən proqramlarda əsas tələb edilən əməllər bunlar olur:

- Word-ü işə salmaq;
- sənədi açmaq və ya yenisini yaratmaq;
- həmin sənədlərlə nə işə etmək (məsələn, verilənlər bazasından və ya istifadəçidən alınan qiymətləri həmin sənədin lazım olan yerlərinə çap etmək).

Word-ün işə salınması, artıq biz tanış olduğumuz, **Application** obyektinin köməyi ilə icra edilir. Sənədlə müxtəlif əməllərin aparılması **Selection**, **Range** və **Bookmark** obyektlərinin köməyi ilə həyata keçirilir (onlar haqqında aşağıda danışılacaq). Əsas əməllərdəki ikinci punkt budur - sənədin açılması və ya yenisini yaradılması (sənədin açıq olub-olmamasının yoxlanılması, sənədin yaddaşda saxlanması v.s.) üçün **Documents** kolleksiyası və **Document** obyektlərindən istifadə edirlər. Sənədin yaradılmasının ən sadə proqram üsulu aşağıdakı VBA proqram kodu ilə reallaşdırıla bilər:

```
Dim oDoc As Word.Document  
Set oDoc=Application.Documents.Add()
```

Bununla biz adi boş sənədi yaratmış olduq (**normal.dot** şablonu əsasında) və ona istinadı **oDoc** obyekt dəyişənində alırıq. Sonra sənədə proqram üsulu ilə tələb olan məlumatı daxil etmək lazımdır. Boş sənədin yaradılması və həmin sənədə proqram üsulu ilə hansısa məlumatın yazılması yalnız sənədin çox sadə olduğunda əlverişli üsul kimi sayıla bilər. Məsələn, daha

mürəkkəb sayılan nağd pul köçürmələrinin elanı və ya böyük həcmli sazişlərin sənədləşməsi işləri proqramlaşdırma üsulu ilə olduqca çətin məsələyə çevrilə bilər. Hətta belə VBA Word proqramı yaradılrsa da, sərf edilən vaxt müqayisədə çox olacaq. Daha asan üsul isə budur – mürəkkəb sayılan sənədi adi Word sənədi kimi yığılıb və formaya salındıqdan sonra, onda doldurulması nəzərdə tutulan yerləri proqram üsulu ilə doldurulması üçün boş qoymaq. Bunu daha asan Word şablonu ilə etmək olar. Məsələn, biz baxdığımız halda, gərək dəyişiləsi verilənlərin yerini boş qoyaraq, əvvəlcədən müqavilənin mətni yığılsın və sənəd .dot genişlənməsi ilə yaddaşda saxlanılsın (tutaq ki, o, C:\ diskində muq_blank.dot adı ilə yaddaşda saxlanılmışdır). Onda proqram üsulu ilə həmin şablon əsasında sənədin yaradılması bu cür həyata keçirilə bilər:

```
Dim oDoc As Word.Document
Set oDoc=Application.Documents.Add("C:\muq_blank.dot")
```

Sonra isə **Bookmark** və **Range** obyektləri ilə (onlar haqqında sonrakı hissələrdə danışılacaq) boş qoyulmuş yerlər mətnlə doldurulur.

Sənədlərin şablonlarının harada saxlanması haqqında bəzi vacib olan qeydləri oxucunun nəzərinə çatdıraraq:

- **birinci variant** – sadəcə *istifadəçinin kompyuterinin lokal diskindəki faylda*. Bu ən sadə variantdır – bu üsulun nöqsanları da var: birincisi – istifadəçi onu təsadüfən dəyişdirə bilər, ikincisi – şablonların hər istifadəçinin kompyuterinə yüklənməsi əlverişli üsul deyil (daha yaxşı üsul – müəssisədəki istifadəçilər üçün şablonların mərkəzləşmiş toplusundan istifadə edilməsidir);
- **ikinci variant** – *fayl-serverdəki, yalnız oxumaq üçün əlçatan olan, gizli şəbəkə kataloqunda* şablonların saxlanması. Bu halda istifadəçi kompyuterində lazım olan şablonun olub-olmaması haqqında narahat olmağa dəyməz. Bununla belə, burada da problemlər yaranır: - fayl-serverdəki xarici fayllardan asılı olan proqram qeyri avtonom olur - məsələn, onun filiallara köçürülməsi müəyyən çətinliklərlə müşahidə olunacaq;
- **üçüncü variant** – şablonu *verilənlər bazasında yerləşdirmək*. Bu variant üçün ən əlverişli üsul – şablonu OLE obyekt kimi Access verilənlər bazasına yerləşdirməkdir. Həmin Access verilənlər bazasında tətbiqi proqramın kodunun, istifadəçini qrafik formalarının v.s. yerləşdirilməsi sərfəlidir. Word-ün işə salınması bu halda proqram səviyyəsində Access-dən icra ediləcək. Əlbəttə, bu cür proqram yalnız Access verilənlər bazasına yox – bütün xarici mənbələrdəki verilənlərə də istinad edə biləcək (SQL Server, Oracle v.s).

Oxucu özü də təyin edə bilər ki, üçüncü üsul daha əlverişlidir – çünki istifadəçinin proqramı (şablonları, proqram kodu v.s. ilə birgə) yeganə olan MDB faylından ibarət olacaq (yəni sənədlər adi Word faylları kimi yaradılacaq). Bundan əlavə, Access öz-özündə - olduqca güclü proqramdır (tərkibində çox sayda imkanlar var). Yeganə yarana biləcək problem - `Documents.Add()` metodu yaradılan şablonu yalnız diskdəki fayl kimi qəbul edir (verilənlər bazasından heç bir məlumat olmur). Bu halda əvvəlcədən verilənlər bazasından sənədin şablonu çıxarılır və sonra

müvəqqəti kataloqda saxlanılır. Daha yaxşı üsul bundan ibarətdir: verilənlər bazasında `OLEActivate()` metodu ilə şablon obyektini aktivləşdirmək. Bu halda, avtomatik olaraq, Word işə salınacaq və onun tərkibində, verilənlər bazasındakı şablon əsasında, yeni sənəd yaradılacaq. Bu haqda daha ətraflı məlumat Access VBA proqramlaşdırması haqqında olan fəslində danışacağıq.

Bəzən də bu cür vəziyyət yaranır: proqram üsulu ilə sənəd yaratmaq əvəzinə mövcud olanının açılması və onun üzərində hansısa əməliyyatın aparılması daha əlverişli variant kimi meydana çıxır. Sənədin ən asan üsulla açılması - `Documents` kolleksiyasının `Open()` metodu ilə bu əməliyyatı həyata keçirməkdir. Bu metodunun ən sadə reallaşdırılması aşağıda verilmiş VBA kodu ilə mümkün ola bilər:

```
Dim oDoc 1 As Document  
Set oDoc1=Documents.Open("c:\doc1.doc")
```

Əgər bu sənəd artıq açıqdırsa, onda, standart hala görə, sadəcə artıq açılmış sənədə istinad yaradılır (sənədin yenidən açılmasının əvəzinə).

`Open()` metodunun vacib (və bununla belə aşkar olmayan) xassəsi budur ki, onun istifadəsi zamanı faylın açılmasında "**Security Alert**" (təhlükəsizlik xəbərdarlığı) dialog pəncərəsi açılmır (həmin pəncərə istifadəçi makrosların işləməsi dayandıra bilər). Bu imkana görə proqram tərtibatçısı yaratdığı tətbiqi proqramın etibarlı işləməsini sığortalaya bilər və ya müdafiə sistemini reallaşdırma bilər (çapdan, müxtəlif funksiyalara əlçatmadan v.s.). Sənədlərin yadda saxlanmasının ən yaxşı üsulu - `Document` obyektinin `Save()` və `SaveAs()` metodlarından istifadə edilməsidir. `Documents` kolleksiyasında həmçinin məxsusi `Save()` metodu var – bu metodla bütün açıq olan Word sənədlərini həmin anda yadda saxlanması mümkündür (adətən bu üsul praktiki işlərdə əlverişli olur). Qeyd edək ki, Word-də, Word-ün başqa versiyalarının açılmasından əlavə, digər formatlarda olan sənədlərin açılması da mümkündür: məsələn, TXT, HTML, XML v.s. Yadda saxlanması da, qurulmuş olan, onlarla başqa-başqa formatlarda mümkün olur. Digər tərəfdən istifadəçi öz istifadəçi formatından da istifadə edə bilər: `FileConvertor` obyektinin köməyi ilə. Bu üsulla, proqram səviyyəsində, makrosların köməyi ilə çox sayda fayl serverlərinin müxtəlif kataloqlarında və **Exchange Server**-in ümumi qovluqlarında, və ya **SharePoint Portal Server** verilənlər bazasında, yerləşən faylları çevirmək olar. Burada kataloq ağacından keçmək üçün daha əlverişli üsul - `Scripting.Runtime` kitabxanasının (bu kitabxana hər bir Windows əməliyyat sistemi ilə işləyən kompyuterdə mövcuddur – 4-cü hissəyə bax) istifadə edilməsi ola bilər.

11.4.2 `Documents` kolleksiyasının xassələri və metodları

Word.Documents kolleksiyası, `Add()`, `Open()`, `Item()` metodları, VBA-da proqram səviyyəsində Word-də sənədlərin yaradılması və açılması

Yuxarıda qeyd etdiyimiz kimi, `Documents` kolleksiyası, həmin an açılmış, bütün Word sənədlərini təmsil edir. Kolleksiyada sənədlərin nömrələnməsi 1-dən başlayır. Bu kolleksiyanın, proqramlaşdırmada olan praktiki baxımdan, `Count` xassəsi ən vacib xassəsidir – açılmış

sənədlərin sayını təyin edir. Həmin baxımdan bu kolleksiyanın metodları dəyərli olduğuna görə, onlarla daha ətraflı tanış olmağımız çox vacibdir. Bu metodların bəziləri haqqında biz artıq əvvəlki hissədə bəzi qeydlər etmişik - aşağıda onların izahlı siyahısı verilib:

- **Add()** – bu metod yaradılmış yeni sənədin həmin anda açılmasını təmin edir (üstəlik onun obyektinə istinadı da qaytarır). Bu metod yeni yaradılan Word sənədləri üçün ən geniş yayılmış üsuldur. Metodun tam sintaksisi bu cürdür:

```
Add(Template, NewTemplate, DocumentType, Visible)
```

Bu kodda **Template** - yeni yaradılan sənədin şablonudur, **NewTemplate** - yeni şablonun yaradılmasını təyin edir (qiymətləri: **True/False**), **DocumentType** - sənəd tipinin variantlarını təyin edir (qiymətləri: **wdNewBlankDocument**, **wdNewEmailMessage**, **wdNewFrameset** və ya **wdNewWebPage** (standart halda – yeni boş sənəd qurulmuş olur), **Visible** – sənədin görünən və görünməyən olduğunu təyin edir (standart halda – sənəd görünən olur).

- **Open()** – bu metod **Documents** kolleksiyasının daha bir vacib olan metodlarından sayılır: diskdə olan sənədi açaraq, onu kolleksiyaya daxil edir. Metod çox sayda parametr qəbul edir, onların içində ən vacibi yalnız birisidir – sənədin adı (həmçinin ona gedən yolun ünvanı). Aşağıdakı nümunədə metodun ən sadə tətbiqi göstərilib:

```
Dim oDoc1 As Document
Set oDoc1=Documents.Open("c:\doc1.doc")
```

- **Item()** – metodu ilə kolleksiyadakı sənədi onun indeksinə görə tapmaq mümkün olur. Adətən lazım olan sənədə istinad almaq üçün **For...Next** idarəetmə strukturundan istifadə edirlər (hansısa xassəsə yoxlanılanda isə həmin strukturun içərisində **If** şərti idarəetmə operatoru tətbiq edilir). Əksər hallarda yoxlamaları **Name** xassəsinə görə aparılırlar, aşağıdakı nümunədəki kimi:

```
Dim oDoc1 As Document
For i=1 To Documents.Count
Set oDoc1=Documents.Item(i)
If oDoc1.Name="doc1.doc" Then
Exit For
End If
Set oDoc1=Nothing
Next
```

Bu konstruksiya **oDoc1** dəyişəni formasında olan istinadı **doc1.doc** sənədinə qaytarır (əlbəttə, əgər o həqiqətən də kolleksiyada varsa). Əgər o, kolleksiyada yoxdursa, onda səhv olmasın deyə, əlavə yoxlamaların aparılması lazımdır. Praktiki işlərdə, məsələn, **doc1.doc** sənədinin adının yoxlamadan əvvəl aşağı registrə keçirilməsi (və ya keçirilməməsi) vacib element kimi meydana çıxa bilər. **Item** xassəsinə görə sənədin obyektinə bir başa yol tapmaq olar. Məsələn, aşağıdakı nümunədə biz **Documents** kolleksiyasının bir nömrəli sənədinin adını ala bilərik:

MsgBox Documents.Item(1).Name

- **Save()** və **Close()** - metodlarının köməyi ilə, uyğun olaraq, kolleksiyadakı bütün sənədlərin yaddaşda saxlanması və bağlanması həyata keçirilir.
- **CanCheckOut()** və **CheckOut()** - metodları, uyğun olaraq, birincisi sənədi monopol halda əlçatmasını yoxlayır və ikincisi isə sənədin monopol halda əlçatmasını təyin edir. Bu metodları yalnız o halda tətbiq etmək olar ki, sənəd **SharePoint Portal Server** verilənlər bazasının sənəd kitabxanasında yerləşib.

11.4.3 Document obyektini ilə işləmə qaydaları, obyektin xassələri və metodları

Word.Document obyektini, VBA-da sənədin xassələri və metodları ilə proqramlaşdırma

Application obyektini ilə Word-ü işə salandan sonra və **Documents** kolleksiyası ilə yeni sənəd açıldıqdan (və ya açılmışların içərisindən tapıldıqdan və yaxud, ümumi halda, lazımı sənədə istinad alındıqdan) sonra **Document** obyektinin xassələri, metodları və hadisələri ilə sənədin üzərində müxtəlif əməllər aparıla bilər. Bu obyektin onlarla xassə və metodları vardır, biz onlardan proqramlaşdırma üçün ən vacib olanlarla tanış olacağıq: müstəqil öyrənmə üçün onlardan yalnız, aşkar olan və az istifadə edilənlər qalacaq.

Əsas budur ki, **Document** obyektinə istinad etdikdə xüsusi obyekt dəyişənini yaratmamaq da olar. **Document** obyektinə yol tapmaq üçün ən azı üç üsul vardır:

- sənədlə **Documents** kolleksiyasının elementi kimi işləmək. İstinad sintaksisi, məsələn, bu cür görünə bilər:

```
Documents.Item(1)
```

- xüsusi **ThisDocument** açar sözündən istifadə etmək. Bu üsulla sənədin obyektinə istinad almaq olar (icra edilən proqram modulu da həmin obyektə aiddir), məsələn:

```
MsgBox ThisDocument.Name
```

- **Application** obyektinin **ActiveDocument** xassəsindən istifadə etmək. Bu xassə bizə aktiv sənədin obyektini qaytaracaq:

```
MsgBox Application.ActiveDocument.Name
```

və ya daha sadə formada

```
MsgBox ActiveDocument.Name
```

Document obyektinin ən vacib xassələrinin izahlı siyahısı aşağıda verilib:

- **ActiveWritingStyle** – cari aktiv stili təyin edir (məyən səviyyənin başlığı, adi mətn, hiperistinad v.s.). Mətn daxil edildikdə yoxlanması məsləhət görülür.

- **AttachedTemplate** – şablonun qoşulmasını təyin edir (bütün makrosları, stilləri, avtomətn yazıları v.s. ilə birgə) və ya hansı şablonun qoşulmasını yoxlaya bilər (“əl ilə”, məsələn, **Tools**→**Templates and Add-Ins** menyusundan).
- **Background** - fon şəkilini təmsil edən, **Shape** obyektini qaytarır (fon şəkilləri yalnız Web-sənəd rejimində görünən olur);
- **BuiltInDocumentProperties** – xassəsi **DocumentProperties** kolleksiyasına istinad alınmasına imkan yaradır. Həmçinin eyni adlı obyektlərə də istinad alınmış olur (ad, müəllif, kateqoriya, şərhlər v.s.);
- **Characters** – bu xassə isə **Range** obyektlərinin kolleksiyasını qaytarır (hər biri bir işarəni təmsil edir). Xassə yalnız **Document** obyektində deyil, **Selection** və **Range** obyektlərində də vardır. Bu xassə, məsələn, axtarış və dəyişdirilmə əməliyyatı aparıldıqda və ya statistik hesablamalarda (məsələn, tərcüməçiyə tərcümə etdiyi məndəki işarələrin sayına görə pul ödənişi olacaqsə);
- **Content** – xassəsi **Range** obyektini qaytarır, həmin obyekt isə sənədin əsasını (`main document story`) təşkil edir. Sadə dillə izah etsək – həmin obyekt sənədin mətnidir (kolontitulsuz, qeydlərsiz, şərhərsiz v.s.);
- **CustomDocumentsProperty** – sənədin istifadəçi xassələrini təmsil edən, **DocumentProperties** obyektlərini qaytarır. Sənədlə birgə dəyişənlərin istənilən qiymətlərinin yaddaşda saxlanılmasında istifadə edilə bilər. Məsələn, açıq olan sənədlərin sayını hesablayanda (çap edildi və ya çap edilmədi, bayraqçıqların sayılmasında, bu və ya digər funksiyanın hansı sayda çağırılmasında, hansı kompyuterlərdə və hansı istifadəçilər tərəfindən açılmasını dəqiqləşdirdikdə v.s.) çox əlverişli proqram səviyyəli vasitədir;
- **DefaultTabStop** – bu xassə tabulyasiya işarəsi istifadə edildikdə, geriyyə yer buiraxılmasını təyin edir. Standart halda – 35 punkt qurulmuş olur (təxminən 1,25 sm. bərabərdir);
- **DisableFeatures** - Word-ün yalnız son versiyalarının imkanlarını söndürür (kompyuterlərdə Windows-un köhnəlmiş versiyaları olan istifadəçilər üçün nəzərdə tutulub). Adətən **DisableFeatures** xassəsi sadəcə bu rejimi qurur, uyğunlaşmanın konkret olan səviyyəsi isə **DisableFeaturesIntroducedAfter** xassəsi ilə təyin edilir;
- **DoNotEmbedSystemFonts** – sistem şriftlərinin sənədə daxil edilməsinin qarşısını alır (standart halda – Rus, ..., Yapon, Çin v.s. dilləri üçün latin dilindən fərqli işarələr yerləşdirilir). Sənədin həcmnin qısalmasına imkan verir – bu halda həmin dillərdə olmayan sistemlərdə işləyən istifadəçilər sənədi oxuya bilmir;
- **EmbedTrueTypeFonts** – çox vacib olan xassədir, xüsusi ilə istifadəçi işində ekzotik şriftlərdən istifadə edirsə (məsələn, müxtəlif nəşriyyatlarda). **true-type** şriftlərinin əlavə

edilməsi zəmanət verir ki, sənədin alındığı yerdə onu eyni ilə, həmin sənədi göndərən istifadəçi görən kimi, görəcəklər;

- **Envelope** – xüsusi obyekt olan **Envelope** üçün istinad alınmasına imkan yaradır (poçt konvertlərinin yaradılmasında istifadə edilir);
- **Fields** – eyni adlı **Fields** kolleksiyasının obyektlərinə istinad alınmasını təmin edir. Sahələrlə işlədikdə çox xeyirli olur (Word-ə sahələr anlayışı - sənəddə dəyişilən verilənlər üçün xüsusi təyin edilən yerə deyirlər): düsturlar, tarixlər, müəllif haqqında məlumat, sənədin ölçüsü haqqında məlumat v.s. Word-ün adi qrafik interfeysin köməyi ilə yeni sahəni bu menyudan əlavə etmək olar: **Insert**→**Filed**;
- **Footnotes** – qeydlərin qaytarılmasını təmin edir;
- **Formatting...** - prefiksi ilə başlayan xassələr Word-ün **Formating** alətlər panelindəki stillər siyahısında nəyin göstərilməsini təyin edir;
- **FormFields** – analogi olaraq, **Fields** xassəsinə oxşayır. Əlavə olaraq bu halda formalardakı sahələrə istinad alınır;
- **FulName** – obyektin tam adıdır (fayl sistemi və ya Web-də ona gedən yol ilə birgə). Yalnız oxumaq üçün əlçatandır;
- **GrammarChecked** – qrammatika nöqtəyi nəzərdən, yoxlanmış bütün sənədin işarələnməsini təmin edir (faktiki olaraq, cari sənədin qrammatika yoxlanışını söndürmək əməlini yerinə yetirir). Eyni xassə **Range** obyektində də var. Yoxlamada tapılan səhvlər kolleksiyasını **GrammaticalErrors** xassəsi ilə almaq olar, səhvlərin yaşıl rəngli dalğavari xətlə seçilməsini isə **ShowGrammaticalErrors** xassəsi ilə əldə etmək olar (əgər onlar hələ seçilməyibsə). Orfoqrafik səhvlərin tapılmasında analogi xassələr istifadə edilə bilər - **SpellingChecked**, **SpellingErrors** və **ShowSpellingErrors** xassələri ilə;
- **HasPassword** – cari sənəddə parolun qoyulmasını yoxlayır. Digər oxşar xassə parolu təyin edir – **Password**. Bütün Office proqramlarının parol sisteminin zəif olduğunu nəzərə alsaq, onda bu xassədən yalnız istisna hallarda istifadə etməyə məsləhət görürlər;
- **Indexes** – sənəd üçün indekslər kolleksiyasını qaytarır (yəni obyekt göstəricilərini);
- **Name** – sənədin adını qaytarır (ona olan yolu göstərmədən);
- **OpenEncoding** – sənədin açılmasında istifadə edilən kod səhifəsini qaytarır (məsələn, Rus dili üçün bu kod səhifəsini qiyməti bərabərdir 1251);
- **PageSetup** – eyni adlı obyektə istinadın alınmasını təmin edir (əsasən, çap əməllərində istifadə edilir);
- **Paragraphs** – cari sənəddəki abzaslar kolleksiyasına istinadı qaytarır;

- **Path** – fayl sistemində cari sənəd üçün yolu qaytarır (adı göstərmədən). Həmin yolda yerləşən digər yeni faylın yaradılmasında istifadə edilə bilər.
- **Permission** – bu xassə eyni adlı **Permission** obyektinə (bu obyekt Word sənədinin daxili icazələr sisteminin idarə edilməsində istifadə edilir). Bu obyekt fayl sisteminin icazələrinin idarə edilməsində tətbiq edilə bilməz;
- **PrintRevisions** – sənədlə birgə redaktorun qeyd işarələrinin çap edilib-edilməməsini təyin edir (standart halda – çapa icazə verilir);
- **ProtectionType** – cari sənədin təhlükəsizliyini və ya mühafizəsini yoxlayır (məsələn, hər şeyə icazə var və ya yoxdur, yaxud yalnız oxunmağa, formalar sahəsində dəyişikliyə icazə var v.s.). Mühafizənin özü **Protect ()** metodu ilə qurulur;
- **ReadOnly** – yalnız oxumağı təyin edir (adından məlumdur). Bu xassə yalnız oxumağa əlçatan olur – çünki uyğun olan atribut fayl sistemində qurulur;
- **RemoveDateAndTime** və **RemovePersonalInformation** – sənəddə olan məlumatların (tarix, zaman, icra edilən dəyişdirilmələr, istifadəçi haqqında olan bütün məlumatlar) silinməsini təyin edir. Nümunə-fayl yaradılmasında çox faydalı xassə kimi istifadə edilir;
- **Saved** – son dəfə hansı vaxt cari sənədin dəyişdirilməsini təyin edir, çox vacib xassələrdəndir;
- **SaveEncoding** – sənədin yaddaşda saxlanılmasında istifadə edilən kodlaşmanın göstərilməsini və ya alınmasını təmin edir;
- **SaveFormat** – sənədin formatı (DOC, RTF, TXT, HTML v.s.) haqqında olan məlumatın alınmasına xidmət edir – yalnız oxunma üçün əlçatandır;
- **Sections** – sənəddəki bölmələrin kolleksiyasını qaytarır. Ona oxşar əməli, yalnız sənəddəki cümlələr üçün - **Sentences** xassəsi yerinə yetirir. Analoji qaydada işləyən, **Shapes**, **Styles**, **Subdocuments**, **Tables**, **Windows** və **Words** xassələrini qeyd etmək olar;
- **Type** – sənədin tipini qaytarır (addırmı, şablondurmu və ya freymləri olan Web səhifələridirmi);
- **Variables** – proqramlaşdırmada vacib xassələr kateqoriyasına aiddir: sənədlə birgə istifadəçinin öz xidməti verilənlərinin və istifadəçi atributlarının (**custom attributes**) yaddaşda saxlanılmasında istifadə edilir. Sənədin istifadəçi atributlarından fərqli olaraq, istifadəçilər digər xassələri görmür.

İndi isə **Document** obyektinin ən vacib olan metodlarının izahlı siyahısı ilə tanış olaq:

- **Activate ()** – hansısa, lazım olan, sənədin aktivləşməsini həyata keçirir (məsələn, mətnin daxil edilməsini yerinə yetirmək üçün).

- **AddToFavorites()** – “Seçilmiş” kataloqunda sənədə istinad əlavə edilməsinə xidmət edir.
- **CheckSpelling()** və **CheckGrammar()** – uyğun olaraq, orfoqrafiyanın və qrammatikanın işə salınmasını təmin edir;
- **Close()** – sənədin bağlanmasını icra edir. Standart halda yaddaşda saxlama ilə həyata keçirir;
- **Compare()** – sənədin digəri ilə müqayisə edib, fərqlər tapılan yerdə, redaktor şərhlərinin generasiyasını həyata keçirir;
- **DataForm()** – verilənlər formalarını redaktə etməyə imkan verir (o, verilənləri ki, hansılar sənəddə sətirlər və sahələr ayrıcıları ilə ayrılıblar). Ümumiyyətlə, Word-də verilənlər formalarını yalnız fövqəladə hallarda istifadə etməyə məsləhət görürlər – səhmanlaşmış verilənlərlə işləmək üçün Excel və Access-də daha çox imkanlar var;
- **DetectLanguage()** – mətnin dilinin təyin edilməsinə xidmət edir. Yoxlama cümlələrlə icra edilir (lüğətlərdə olanlarla söz bə söz yoxlayaraq). Bu cür yoxlama, avtomatik rejimdə keçirilir, mətn yığıldıqda və ya yeni sənəd açıldıqda. Yenidən dillər yoxlansın deyərək **LanguageDetected** xassəsi **False** qiymətinə keçirilsin;
- **FitToPages()** – bu olduqca faydalı xassə, avtomatlaşdırılmış rejimdə, mətnin şriftlərinin ölçüsünü elə dəyişdirir ki, sənəddə səhifələrin sayı əvvəlkindən yalnız bir səhifə az olsun. Əslində bu xassə praktiki işlərdə çox faydalı rol oynaya bilər, xüsusilə nəşriyyat sahəsində;
- **FollowHyperlink()** – istənilən proqramda göstərilən sənədin açılmasını yerinə yetirir (məsələn, əgər HTML formatlı sənəd seçilibsə, onu Internet Explorer-də açır);
- **GoTo()** – çox güclü metoddur: **Document**, **Range** və **Selection** obyektləri üçün mövcuddur. Birinci iki hal üçün o, **Range** obyektini qaytarır, üçüncüdə sadəcə mətn yerləşməsi göstəricisini lazımı yerə köçürür. Qaytarılan obyektlər, onlara ötürülən parametrlərdən asılı olaraq, səhifənin sətirin əvvəlinə, aralıq nişanlamaya, cədvələ, bölməyə, sahəyə, qeydə, düstura v.s. işarə edə bilər. Sənəddə həmin obyektin müəyyən nömrəsinə keçməyə imkan verir. Mətnin avtomatik rejimdə daxil edilməsində, mətnin lazım olan yerdə daxil edilməsi üçün, həmin yerin nişanının qurulmasını təmin edir;
- **Merge()** – iki sənədin birləşməsinə həyata keçirir: olduqca güclü, həmçinin mürəkkəb metoddur (redaktor qeydlərinə əsaslanır);
- **PresentIt()** - cari Word sənədinin PowerPoint-da açılmasını təmin edir;
- **PrintOut()** – bütün sənədin və ya onun bir hissəsinin çapa çıxarılmasına xidmət edir. Əslində çox mürəkkəb metoddur: iyirmidən artıq parametr qəbul edir (bir çoxu məcburi deyil). Daha çox **Application**, **Document** və **Window** obyektlərində istifadə edilir;
- **PrintPreview()** – sənədi qabaqcadan baxım rejiminə keçirir;

- **Protect()** – parol və ya İRM (Information Rights Management) ilə sənədə əlavə edilən dəyişikliklərə məhdudiyət qoyur;
- **Range()** – çox vacib olan rejimdir: **Range** obyektini qaytarır (ona daha ətraflı aşağıda baxacağıq). Qəbul edilən parametr kimi diapazonun başlanğıc işarəsinin nömrəsini və son işarənin ötürülməsini təmin edir;
- **Redo()** – son əməlin təkrarını yerinə yetirir. Parametr kimi son əməllərin sayını qəbul edir, təkrar müvəffəqiyyətli olduqda, **True** qaytarır;
- **Repaginate()** – sənədin səhifələrə bölünməsini icra edir. Adətən avtomatik səhifələrə bölünmə əvvəlcədən keçirildikdə, istifadə edilir;
- **Save()** – bu metodun mənası adından tam məlumdur: əgər sənəd hələ yaddaşda saxlanmayıbsa, onda **Save As** dialoq pəncərəsi açılır;
- **SaveAs()** – çox güclü və eyni zaman mürəkkəb metoddur: yaddaşda saxlanılan sənədin yolunu, formatını, kodlaşmasını, parollarını (açılmağa və dəyişdirilməyə), şriftlərin daxil edilməsini və digər əməlləri təyin etmək olar. Bu metoddan sənədlərin avtomatik rejimdə çevirməsində daha çox istifadə edirlər;
- **Select()** – bütün sənədin seçilməsini həyata keçirir. Bu metod bir çox obyektlər üçün nəzərdə tutulub, məsələn, həmçinin, **Selection** və **Range** obyektləri üçün;
- **TransformDocument()** – olduqca güclü metoddur, yalnız XML və XSLT-də yaxşı təcrübəsi olan proqramçılar üçün: sənədə stillər cədvəlinin tətbiq edilməsinə imkan verir (Extensible Stylesheet Language Transformation, XSLT), məlum olduğu bu imkanın köməyi ilə hər şey dəyişdirilə bilər;
- **Undo()** – son zaman edilən bütün (və ya qismən olan sayda) əməllərin imtina edilməsinə xidmət edir. Sintaksis və işləmə prinsipinə görə tamamilə **Redo()** metodunun analoqudur;
- **UndoClear()** – dəyişiklər imtinası buferinin təmizlənməsini təmin edir (bəzi hallarda, məsələn, istismarçının tərtib edilmiş proqramın istifadəçiyə verildiyi zamanda çox faydalı olur – proqramın qəfildən dəyişdirilib, xarab olmasının qarşısını alır);
- **UnProtect()** – sənəddən, **Protect()** metodu və ya qrafik interfeyslə qoyulmuş mühafizənin götürülməsini həyata keçirir. Mühafizə edilmiş sənəddə proqram üsulları ilə dəyişiklik edilməsindən əvvəl çox faydalı rol oynaya bilər.

Qeyd etməliyik ki, **Document** obyektinin üç sayda daha çox işlənən hadisələri vardır: **New()**, **Open()** və **Close()**. Bu metodlar tam aydın məna daşıyır və **Visual Basic** redaktorunun kod pəncərəsində əlçatandır.

11.5 Selection, Range və Bookmark obyektləri

11.5.1 Selection obyektini ilə işləmə qaydaları

Word.Selection obyektini, mətnin seçilmiş hissəsi ilə işləmə qaydaları – üstünlüklər və nöqsanlar

Tətbiqi proqramı işə saldıqdan, lazımı faylı tapılıb aktivləşdirilməsindən, sonra ən çox sayda yerinə yetirilən əməl – lazımı yerdə mətnin daxil edilməsi və redaktə edilməsi əməlləridir. Bunun üçün **Selection, Range və Bookmark** obyektlərindən istifadə edirlər. Onların hər biri öz təyinatına uyğun və nəzərdə tutulmuş məsələlərdə istifadə edilir. Birinci olaraq, **Selection** obyektinin öyrənməsindən başlayaq.

Adətən Word sənədinin pəncərəsində nə işə etməkdən əvvəl, istifadəçi mətnin lazımı hissəsini seçir və ya mətnin yerləşdiriləsi yerin göstəricisini lazımı yerə keçirir. **Selection** obyektini məhz həmin seçilmiş mətn hissəsini təmsil edir (əgər heç nə seçilməyibsə, onda həmin yer yerləşdirmə göstəricisinin yerindədir). Məhz həmin obyektini adətən makrorekorder öz işində istifadə edir.

Selection obyektini yaratmaq və ona istinad almaq məcburi deyil (adətən hətta mümkün olmur). İş burasındadır ki, **Selection** obyektini sənəddə yalnız bir sayda ola bilər. Word işə salındıqda o, avtomatik olaraq yaradılır və həmişə əlçatan olur. Ona bu cür istinad etmək olar:

```
Application.Selection.Text="Yerləşdirilən mətn"
```

və ya daha sadə

```
Selection.Text="Yerləşdirilən mətn"
```

Adətən istifadəçiyə **Selection** obyektinin işarə etdiyi yeri düzgün təyin edilməsi lazım olur (mətnin lazımı hissəsini və ya mətnin yerləşdirilmə nöqtəsini seçmək üçün).

Word sənədində seçilməni aşağıdakı qaydalar əsasında sazlamaq olur:

- ən sadə üsul – sadəcə istifadəçi tərəfindən lazım olan mətn hissəsinin seçilməsini əsas tutmaq. Adətən bu cür üsuldan mətnin hissələrinin mürekkəb redaktə/formatlaşdırma və ya istifadəçi tərəfindən sənədin təyin edilmiş yerində məlumatın yerləşdirilməsi əməlləri aparıldıqda (avtomatik rejimdə lazım olan yerin tapılması mümkün olmayanda) istifadə edirlər;
- **Select ()** metodundan istifadə etmək, hansı ki, bir çox obyektlər üçün nəzərdə tutulub (**Document, Range, Bookmark, Table** sətirlər və sütunlar tipli bütün alt-obyektləri ilə, **PageNumber, Field** v.s. üçün). Bu metod bütün sənədi seçir, nişanlamayı, cədvəli v.s.;
- **Selection** obyektinin çoxsaylı metodlarından istifadə etmək (mövcud olan seçilməni dəyişdirmək üçün);
- Mətnin lazımı hissəsini tapmaq üçün **Find** obyektindən istifadə etmək. Bu obyekt haqqında daha ətraflı - xüsusi bölmədə danışılacaq;

- əgər informasiya sənədin ən əvvəlində yerləşdiriləcəksə, onda ümumiyyətlə, heç nə etmək lazım gəlmir. Çünki standart halda yerləşdirmə göstəricisi həmişə sənədin başlanğıcında qurulmuş olur. Bu halda sənədin aktiv edilməsi gərək yaddan çıxarılmasın.

Burada bir incəlik var: əgər lazımı mətn yerinin istifadəçi tərəfindən seçilməsi əsas tutulursa, onda yadda saxlamaq lazımdır ki, bəzi istifadəçilər sənəddə bir-biri ilə qarışıq əlaqəsi olmayan iki mətn hissəsini <Ctrl> düyməsinin köməyi ilə seçə bilər və ya mətnin hissəsini yox, cədvəlin hissəsini, şəkli və ya digər qeyri standart obyektini seçə bilər. Daha çox **Selection** obyektini işləyən proqramın davranışı bu halda əvvəlcədən gözlənilən kimi olmaya da bilər. Buna görə əlavə yoxlamaları aparmaq üçün **Selection** obyektinin **Type** və **Information** obyektlərindən istifadə edilməsi məsləhət görülür.

Buna baxmayaraq ki, **Selection** obyektinin tətbiqi mətn redaktəsində ən sadə və əyani metod sayılır (makrorekorder daha çox məhz onu istifadə edir), praktikada proqramçılar ondan az istifadə edirlər. Bunun çox sadə izahı var: bu obyekt istifadə edildikdə həddən artıq istifadəçidən asılılıq olur. Əgər tərtibatçı tərəfindən hazırlanmış proqram kodu icra edildikdə istismar edən istifadəçi, təşəbbüs edərək, sənəd üzərində mausla şıqqıldama etsə - nəticə tamam ilə gözlənilməz ola bilər. Bu cür istifadəçilərin “əməliyyatlarından” qorunmaq üçün iki üsul vardır:

- gizli sənədlə (yəni görünməz olan sənədlə) və ya, məsələn, Word-ün gizli olan nüsxəsi ilə işləmək. Görünməzliyi qurmaq/söndürmək üçün **Document** və **Application** obyektlərinin **Visible** xassəsindən istifadə etmək olar;
- daha əlverişli üsul - **Selection** obyektini əvəzinə **Range** və **Bookmark** obyektlərini istifadə etmək (onlar haqqında sonrakı bölmələrdə məlumat verilecək).

11.5.2 Selection obyektinin xassələri və metodları

Word.Selection obyektinin xassələri və metodları, VBA-da seçilmiş mətn hissələri ilə proqramlaşdırma işləri

Əvvəlcə - **Selection** obyektinin daha tez-tez istifadə edilən xassələri haqqında

- **Bookmarks** – xassəsi **Bookmarks** kolleksiyasını (yəni, əslində, mətnin seçilmiş hissəsində olan bütün əlavə qurmaları) qaytarır. Word istifadəsi ilə birgə VBA proqramlarında əlavə qurmalar - ən çox istifadə edilən obyektlərdəndir. Bu haqda xüsusi bölmədə daha ətraflı məlumat verəcəyik.
- **Start** və **End** – bu xassələr, uyğun olaraq, seçilmədəki birinci və sonuncu işarəni təyin edir (sənədin mətni ilə bağlı və ya onun digər hissələri ilə bağlı, məsələn, şərhilərlə). **document story**-də birinci pozisiya – həmişə 0 olur. Dəyişilməz şablondan sənəd yaradıldıqda, bu xassələrdən istifadə etmək olar (sənəddə mətnin yerləşdirilməsi üçün yeri tapmaq üçün). Bu üsuldən proqramlaşdırmada az istifadə edirlər – çünki şablonda dəyişiklik etdikdə çox sayda proqram kodunun yenidən dəyişdirilməsi lazım olur.

- **ExtendMode** - istifadəçini mətnin seçilməsi rejiminə keçirir (bu xassə yalnız o, zaman tətbiq edilir ki, **<Home>** və **<End>** klavişlərinin basılmasında yerləşdirmə göstəricisinin yerini dəyişməsinə gətirmək əvəzinə - seçilmiş hissənin dəyişdirilməsi ilə nəticələnir).
- **Find** — çox vacib xassədir: eyni adlı **Find** obyektini qaytarır. Bu obyekt və ona qurulmuş **Replace** obyektini haqqında daha ətraflı mətnin axtarılması və dəyişdirilməsinə həsr edilmiş hissədə verəcəyik.
- **Flags** – bu xassə seçilmədə olan bəzi məqamların yoxlanılması və ya dəyişdirilməsi ilə bağlıdır: seçilən hissə aktivdirmi, sətirin sonundadırmı v.s. Bit maskası ilə eyni vaxtda beş sayda parametri tənzimləyə bilər.
- **Font** – eyni adlı **Font** obyektini qaytararaq, seçilmiş mətn hissəsində formalaşdırmanı idarə etməyə imkan yaradır. **Format**→**Font** menyusundakı qrafik interfeysdə olan bütün imkanlar burada da əlçatandır. Məsələn, seçilmiş mətn hissəsinə 10 ölçülü Arial şriftini təyin etdikdə aşağıdakı VBA kodundan istifadə etmək olar:

```
Selection.Font.Name="Arial"
Selection.Font.Size=10
```

- **Information** – məqsədli yoxlamalar üçün ən vacib sayılan xassələrdəndir: seçilmiş mətn hissəsi haqqında dolğun məlumat qaytarır (sənədin hansı hissəsindədir, cədvəlin içərisindədirmi, **<CapsLock>** və **<NumLock>** düymələri qoşulmuşdurmu, seçilmə hansı səhifədədir, seçilmədə nə qədər səhifə var v.s.).
- **IPAtEndOfLine** - əgər mətni daxil etmə kursoru ("*insertion point*" - IP) sətirin sonundadırsa (düzəlişdə sağ tərəfin ən kənarındadırsa) – **True** qiymətini qaytarır.
- **LanguageId** – müəyyən dildə yazılan mətn kimi, seçilmiş mətn hissəsini nişanlamağa imkan verir: dilin düzgün təyin edilməsi orfoqrafiyanın yoxlanılmasında çox faydalı olur.
- **NoProofing** – seçilmiş mətn hissəsi üçün orfoqrafiya və qrammatikanın yoxlanmasını söndürmək üçün imkan verir. Bu cür nişanlamalardan həmin hissədə, məsələn, proqram kodu, soy adlarının siyahısı, şirkətlər adlarının siyahısı, spesifik terminlər v.s., olduqda istifadə etmək məsləhət görülür.
- **Range** – seçilmədən **Range** obyektinin yaradılmasına xidmət edir.
- **StoryType** – yoxlama aparmaq üçün daha bir vasitədir: seçmə yerləşdiyi **document story** üçün tipi təyin edir.
- **Text** – demək olar ki, bu xassə **Selection** obyektinin ən vacibidir: seçilmə yerində (və ya göstərici durduğu yerdə) mətnin daxil edilməsinə imkan yaradır. Məsələn, "Salam!" sözünün 100 dəfə yazılması üçün aşağıdakı VBA kodu tətbiq edilə bilər:

```
For i=0 To 100
    Selection.Text="Salam!"
Selection.EndOf
```

Next

EndOf() metodu burada mətnin 100 sayda yenidən yazılmaması üçün istifadə edilib: çünki mətn daxil edildikdən sonra mətn seçilmiş halda qalır.

- **Type** – daha bir yoxlama vasitəsidir (əgər istifadəçi hansısa qadağa edilmiş hissəni səhvən seçmişdirsə - səhvin qarşısını almaqda kömək edir). Məsələn, adi seçilmədə bu xassənin qiyməti 1 olur, bir biri ilə qarışığı olmayan hissələr seçildikdə isə qiymət -2 olur.
- **Words** – bu xassə eyni adlı **Words** kolleksiyasını qaytarır (həmin kolleksiya **Range** obyektləri yığımından ibarətdir və hər birinə seçilmiş mətn hissəsinə aid sözlər uyğundur).

Müqayisədə **Selection** obyektinin metodlarının sayı xassələrinin sayından daha çoxdur:

- **Calculate()** – mətn yığıldığı zaman həmin anda riyazi hesablamaları icra edərək nəticənin qaytarılmasını təmin edir (yalnız **Single** tipli verilənlər istifadə edilir).
- **ClearFormatting()** – formatlaşmanın ləvğini icra edir (həmçinin mətn və paraqraf səviyyəsində). Bu xassə yalnız **Selection** obyektini üçün tətbiq edilmir - tez-tez **Find** və **Replace** obyektlərində də istifadə edilir.
- **Collapse()** – seçilməni yerləşdirmə nöqtəsinə çevirir. Burada iki variant var: yerləşdirmə göstəricisi seçilmənin əvvəlinə və ya axırına qoyulur. Çox əlverişli metoddur – xüsusilə, əgər köhnə mətn silinmədən yenisi əlavə edilir.
- **Copy()**, **CopyAsPicture()**, **Cut()**, **Paste()** və **Delete()** – bu metodları eyni adlarından anlanılması oxucu üçün çətin olmayacaq.
- **EndKey()** – bu metod, funksionallığına görə, **<End>** düyməsinin basılmasına oxşardır. Ötürülən parametrlərdən asılı olaraq, **DocumentStory** obyektinin, sətirin, sütunun və ya cədvəldəki yazının sonuna keçməyə imkan verir (standart halda sətirin axırına qurulmuş vəziyyətdə olur). Həmçinin həmin yerdə yerləşdirmə nöqtəsinə də qurur. Yerləşdirmə kursurunun sənəddəki mətnin axırına keçirmək üçün aşağıdakı proqram kodundan istifadə etmək olar:

```
Selection.EndKey Unit:=wdStory,Extend:=wdMove
```

Lakin, əgər, kursuru obyektin əvvəlinə keçirmək lazım olsa, onda, analoji olan, **HomeKey()** metodundan istifadə edirlər.

- **EndOf()** – eynən **EndKey()** metodunda olan kimi, həmin rejimdə işarənin, sözün, cümlənin, abzasın, bölmənin, sənəd mətninin, cədvəlin v.s. axırına keçməyi icra edir. Bunun əksinə, əvvələ keçmək lazım olduqda isə **StartOf()** metodundan istifadə edirlər.
- **Expand()** – ötürülən parametrdən asılı olaraq, seçmənin bir söz, cümlə, abzas v.s. genişlənməsinə imkan verir. Buna oxşar **Extend()** metodu da seçməni genişləndirir (yalnız söz əvəzinə - cümlə, cümlə əvəzinə - abzas v.s.). **Expand()** metodunun əksi **Shrink()** metodudur.

- **GoTo ()** – eynən **Document** obyektinin eyni adlı metoduna oxşayır.
- **GotoNext ()** – növbəti sətərə, səhifəyə, əlavə qurmaya v.s. keçmək kimi əməlləri yerinə yetirir. Buna analogi **GotoPrevious ()** metodu işləyir (əvvəlki elementə keçidi icra edir).
- **Insert ()** – bu metodun icra etdiyi adından məlumdur. Daha çox **InsertBefore ()** və **InsertAfter ()** metodları istifadə edilir (uyğun olaraq, seçmədən əvvəl və seçmədən sonra yerləşdirmənin icrası üçün).
- **Move . . .** – prefiksli metodlar, Word-də mətnin yerləşdirilməsi ilə bağlı olan, hər proqramda rast gəlir. Bu kateqoriyada ən vacibləri və əlverişli olanları aşağıda verilib:
 - **MoveLeft ()**, **MoveRight ()**, **MoveUp ()**, **MoveDown ()**, **MoveEnd ()**, **MoveStart ()** – bu metodların təyinatı aşkardır. Onların hər biri əlavə parametrləri qəbul edir (bu parametrlər göstəricinin neçə pozisiya hərəkət etdiyini, seçilmənin hərəkət edib etməməyini v.s. təyin edir).
 - **MoveStartUntil ()**, **MoveStartWhile ()**, **MoveEndUntil ()**, **MoveStartWhile ()** bir biri ilə yalnız onunla fərqlənirlər ki, yerləşdirmə kursoru müəyyən edilmiş sayda pozisiya hərəkət etmir – kursoru müəyyən işarələr ardıcılığı rast gələndə həmin ardıcılığın yanına keçirir.
 - **Move ()** – daha çevik olan metoddur: kursoru, müəyyən işarələr sayından əlavə, sözlər, cümlələr və abzaslar da sayaraq, lazımı pozisiyaya keçirə bilir. İlk olan şablon dəyişdikdə isə proqram kodunda minimum dəyişiklik etmək lazım olur.
- **Next ()** – kursoru müəyyən sayda işarə, söz, cümlə, abzas, bölmə, cədvəl sətiri və sütunu irəliyə keçirir. Bunun əksi olan **Previous ()** metodu isə həmin obyektlər üzrə geriye köçürməni icra edir.
- **NextField ()** – metodu formada növbəti sahəyə keçməyi yerinə yetirir və ya sahələrin qurtardığını yoxlayır (bu halda **Field** obyektini əvəzinə **Nothing** qaytarır). Buna oxşar **PreviousField ()** metodu da mövcuddur.
- **SelectColumn ()**, **SelectRow ()**, **SelectCell ()** – Word cədvəllərində müxtəlif əməllər aparılması üçün çox əlverişli vasitələrdir.
- **SelectCurrentAlignment ()**, **SelectCurrentFont ()**, **SelectCurrentIndent ()**, **SelectCurrentColor ()** v.s. – mətni dəyişdirilənə qədər (düzəlişdirmə, şriftlər, rəng, ölçü v.s.) mətni seçirlər. Bu metodlardan adətən mətnin formatlaşdırılmasında istifadə edirlər.
- **SetRange ()** – seçilmənin saxlanması üçün ən sadə üsuldur: seçilən mətn hissəsinin birinci və axıncı işarəsinin nömrəsi verilir. Nömrələnmə 0-dan başlayır, gizli olan xidməti işarələr də sayılır. Həmin metod **Range** obyektində də mövcuddur.

- **Sort()**, **SortAscending()**, **SortDescending()** - əlifba, tarix v.s. görə yerləşdirməni həyata keçirirlər (həmçinin cədvəllərdəki sətir və sütunlara görə də yerdəyişməni qurmaq olur). Bu metodlara görə xeyli dəyərli zamana qənaət etmək mümkündür.

- **ToggleCharacterCode()** – imkan verir ki, xidməti işarə kodunu yerləşdirərək, həmin an onu Unicoda çevirsin. Məsələn, Euro sözünü daxil etmək üçün aşağıdakı koddan istifadə etmək lazımdır:

```
Selection.TypeText Text:="20ac"  
Selection.ToggleCharacterCode
```

- **TypeText()** – mətnin ən sadə və etibarlı daxil etmə metodudur (ən çox istifadə edilən metodlardandır). Yeganə parametri qəbul edir – daxil edilən mətni. Cari mətnin yenidən yazılması **Option** obyektinin **ReplaceSelection** xassəsindən asılı olur.

- **WholeStory()** – sənədin cari hissəsini seçir (document story). Adətən sənəddəki mətnin qeydlərsiz, redaktor düzəlişi olmadan, kolontitulsuz v.s. seçilməsində istifadə edilir.

11.5.3 Range obyektini ilə işləmə, obyektin xassələri və metodları

Word.Range *obyektini, sənəddə diapazonlarla proqram səviyyəsində işləmək qaydaları, Range obyektinin xassələri və metodları, Selection obyektini ilə müqayisədə olan üstünlüklər*

Yuxarıda qeyd etdiyimiz kimi, mətnin sənəddə yerləşdirilməsi və naviqasiyası üçün daha çox proqram tərtibatçıları **Selection** obyektindən istifadə edirlər. Bu məqsədlə **Range** obyektindən də istifadə etmək olar. Obyektlər arasında olan əsas fərq bundan ibarətdir ki, **Selection** obyektini istifadəçi də təyin edə bilər (sadəcə mausla mətni seçərək), ancaq **Range** obyektini yalnız proqramlaşdırma səviyyəsində təyin etmək olur. Nə göstəricinin (kursorun) cari vəziyyətindən nə də istifadəçinin əməllərindən asılı deyil.

Əgər **Selection** və **Range** obyektinin tətbiqi arasında seçim yaranırsa, təcrübəli VBA proqramçıları, imkan olduqda, birinci növbədə, həmişə **Range** obyektindən istifadə etməyə çalışırlar. Bununla onlar ehtimal olan istifadəçi əməlləri ilə bağlı səhvlərdən (məsələn, proqram səviyyəsində mətn yerləşdirildikdə istifadəçi mausu ilə sənədin müəyyən yerlərindən şıqqıldadır) özlərini sığortalamış olurlar.

Formal olaraq, **Range** obyektinin tərfi bu cürdür: *o, sənəddə kəsilməz olan mətn hissəsini təmsil edən proqram obyektidir*. Bu obyekt **Selection** obyektindən asılı deyil – istifadəçi **Range** obyektini ilə işləyərək, cari mətn seçilməsini dəyişdirməyə də bilər. O, öz tərkibinə heç bir işarə daxil etməyə də bilər - daxil etmə kursorunu təmsil edə bilər. Hər anda **Range** obyektini istənilən sayda ola bilər, **Selection** obyektini isə yalnız bir sayda təmsil edilə bilər.

Range obyektinin yaradılması qaydaları:

- **birinci üsul** - **Document** obyektinin **Range ()** metodundan istifadə etmək. Bu halda gərək diapazonun başlanğıc və sonuc işarəsi ötürülsün, həmçinin həmin işarələrin sayılması həyata keçirən **document story** ötürülməlidir. Məsələn, sənədin birinci 10 işarəsini özündə saxlayan diapazonun seçlməsi üçün aşağıdakı VBA kodu istifadə edilə bilər:

```
Dim rngDoc As Range
Set rngDoc=ActiveDocument.Range(Start:=0,End:=10)
```

- **ikinci üsul** – **Range** xassəsindən istifadə etmək (böyük sayda obyektlər üçün həmin xassə nəzərdə tutulub, məsələn: **Bookmark, Selection, Table-Row-Cell, Paragraph** v.s.). Bu halda həmin xassə ilə **Range** obyektinin özü yaranmış olur;
- **üçüncü üsul** – çox sayda olan köməkçi xassələrdən istifadə etmək (**Characters, Words, Sentences** v.s.). Onların köməyi ilə mətn parçalara bölünür – **Range** obyektlərinə çevrilirlər. Əlbəttə, bu halda **Range** obyektlərinin kolleksiyası yaranmış olur: böyük sənədin hər bir işarəsini təmsil etmiş olurlar. Müqayisədə, məhsuldarlıq baxımından, bu üsul bir o qədər də əlverişli sayıla bilməz.
- **dördüncü üsul** – mövcud olan **Range** obyektini yenidən təyin etmək. Adətən bu məqsədlə **Range.SetRange ()** metodunu tətbiq edirlər;
- **beşinci üsul** – real tətbiqi proqramların tərtibində ən əlverişli üsul sayılır: 1-ci addımda əvvəlcə lazım olan sənədin şablonu yaradılır (müqavilə, kredit vəkalətnaməsi, illik/aylıq hesabat v.s.), 2-ci addımda, həmin şablon yaradıldıqda, onun içərisində, verilənlər daxil ediləsi yerdə, nişanlanmalar əlavə edilir, 3-cü addımda proqram üsulu ilə hər bir nişanlama üçün **Range** obyektini yaradılır, 4-cü addımda həmin obyektlərin köməyi ilə informasiyanın daxil edilməsi icra edilir (məsələn, sifarişçi haqqında məlumat, kassa sifarişi v.s.).

Sazlanma üçün (planlaşdırılmış mətn hissəsinin həqiqətən **Range** obyektinin tərkibində olmasına əmin olmaq üçün) **Range** obyektini əsasında **Selection** obyektini yaratmaq olar (yeni, əslində, bu cür diapazonu seçmək olur). Bu məqsədlə çatmaq üçün **Range** obyektində **Select ()** metodu mövcuddur.

Range obyektinin bir çox xassə və metodları eyni ilə **Selection** obyektinin xassə və metodu (bir çoxu ilə artıq tanış olmuşuq) ilə üst-üstə düşür, buna görə eyni funksiyalı metod və xassələrə burada yenidən baxmayacağıq. Eynilə **Range** obyektində də lazımı mətn hissəsini sənədin müəyyən yerində yerləşdirmək üçün yeganə **Text** xassəsi istifadə edilir.

Aşağıda **Range** obyektinin unikal sayılan metodları haqqında məlumat verilir:

- **InsertDatabase ()** – verilənlər bazasındakı Word faylına sorğunun nəticəsinin yerləşdirilməsi üçün, ola bilsin ki, ən sadə üsuldur. Sorğunun nəticəsi əsasında cədvəl generasiyasını icra edir və **Range** obyektini təyin etdiyi yerə onu qoyur. **ODBC, DDE** üzrə

işləməyi bacarır, **MS Access** ilə işləmək üçün qurulmuş vasitələri var. Bu metodun imkanları məhduddur. Buna görə onun yalnız sadə hallarda tətbiq edilməsini məsləhət görürlər. Başqa hallarda daha yaxşı olar ki, **ADO** obyekt kitabxanası tətbiq edilsin (əvvəlki fəsilə bax). Bu metodun vasitəsi ilə **Access** faylına istinad aşağıdakı VBA proqram kodu ilə icra edilə bilər:

With Selection

```
.Collapse Direction:=wdCollapseEnd
.Range.InsertDatabase _
Format:=wdTableFormatSimple2, Style:=191, _
LinkToSource:=False, _
Connection:="Table.Distributors", DataSource:="C:\Program _
Files\Microsoft Office\OFFICE11\SAMPLES\Northwind.mdb"
End With
```

- **IsEqual()** – iki **Range** obyektini (və ya **Selection** obyektini ilə **Range** obyektini) müqayisə etməyə imkan verir. Əgər sənəddə başlanğıc sonuc pozisiya və **document story** üst-üstə düşürsə, onda **True** qiyməti qaytarılır.
- **FoneticGuide()** – sənəddə mətnin üstündən transkripsiyanın yerləşdirilməsini həyata keçirir.
- **Relocate()** – diapazondakı abzasların yerlərini dəyişdirməyə imkan yaradır.
- **Select()** - əvvəl qeyd etdiyimiz kimi, **Range** obyektini əsasında **Selection** obyektini yaradılır (yəni həmin obyektə bütün mətn seçilmiş olur). Səzləmə işlərində çox əlverişli üsuldur.
- **SetRange()** – bu metod **Range** obyektinin ən vacib sayılan metodlarından biridir: həmin obyektin dəyişdirilməsinə imkan verir. Məsələn, sənədin başlanğıcından kursurun cari vəziyyətinə qədər (və ya seçmənin sonuna qədər) olan mətn tərkibində olan **Range** obyektinin alınmasını təmin etmək üçün aşağıdakı VBA kod proqramını istifadə etmək olar:

Dim MyRange As Range

```
Set MyRange=ActiveDocument.Range(Start:=0, End:=0)
MyRange.SetRange Start:=MyRange.Start, _
End:=Selection.End
MyRange.Select
```

11.5.4 Bookmark obyektini

Word.Bookmark obyektini, şablonda nişanlamanın tətbiqi, Bookmark obyektlərindən Selection və Range obyektlərinin alınması

Bookmark obyektinin mənasını adından da anlamaq asandır: ingiliscə *bookmark*, hərfi olaraq, Azərbaycancaya tərcümədə *kitab arasına yaddaş üçün qoyulan nişanlama və ya işarələmə* mənasına gəlir. Deməli, **Bookmark** obyektini - sadəcə nəyinsə (baxdığımız halda mətn sənədinin) işarələməsidir. Praktikada, şablonlar əsasında yaradılan (məsələn, hesabatlar əsasında), sənədin naviqasiyasında ən əlverişli üsul **Bookmark** obyektinin istifadəsi ilə bağlıdır. **Selection** və **Range**

obyektlərindən prinsipial fərqi – hətta sənəd bağlıqda belə, **Bookmark** obyektini ilə icra edilən bütün seçmələr və diapazonlar sənədlə birgə yaddaşda saxlanılır (adı çəkilən obyektlərdə isə seçmələr sənəd bağlanan an silinir). **Bookmark** obyektini tətbiq edildikdə, əgər sənəd şablon əsasında yaradılıbsa, onda şablonda təyin edilmiş bütün işarələmələr həmin yaradılmış sənəddə də təyin edilmiş olacaq. İşarələməni yaratmaq (**Insert**→**Bookmark** menyusunda) daha asandır, nəinki **Range** obyektini üçün sənədin, abzasın, cümlənin, və ya işarələrin sayını hesablamaq və ya **Selection** obyektini üçün **Move()** (**MoveDown()**, **MoveRight()**, **MoveNext()**) əməllərini yerinə yetirmək. Bundan əlavə, əgər şablonun düzəldilməsi lazımdırsa (praktikada bu əməliyyat tez-tez lazım olur), proqram tərtibatçısı yenidən **Selection** və **Range** obyektlərinin təyini üçün VBA kodunda heç bir dəyişiklik edəsi olmayacaq. **Bookmark** obyektinin funksionallığı geniş deyil. **Range** və **Selection** obyektləri ilə müqayisədə bu obyektin xassələrinin və metodlarının sayı azdır. Məlum səbəblərə görə adi istifadəçilər bu obyektini mətnlə bir başa işləmək üçün istifadə etmirlər. **Bookmark** obyektindən (bütün işarələmələr, sənədə məxsus olan, **Bookmarks** kolleksiyasına yığılmış olur) asanlıqla **Selection** obyektini almaq olur (**Select()** metodunun köməyiylə). **Range** obyektini isə **Bookmark** obyektindən asanlıqla **Range** xassəsi əsasında yaratmaq olur. Sonra həmin yaradılmış obyektlərin xassə və metodlarından istifadə etmək adi məsələyə çevrilir, məsələn, aşağıdakı nümunədəki kimi:

```
ThisDocument.Bookmarks("Bookmark1").Select
MsgBox Selection.Text:
```

Bookmark obyektlərinin proqram səviyyəsində yaradılması məcburi deyil - ehtiyac yarandıqda, onda **Bookmark** obyektinin **Add()** metodundan istifadə edilməsi məsləhət görülür:

```
ThisDocument.Bookmarks.Add Name:="temp", Range:=Selection.Range
```

Add() metodunun cəmi iki parametri vardır (hər ikisi yuxarıdakı nümunədə istifadə edilir).

Bookmark obyektinin bəzi vacib xassələrini qeyd edək:

- **Empty** - əgər bu xassə **True** qiymətini qaytarırsa, onda işarələmə yerləşdirmə nöqtəsini göstərir, mətni yox;
- **Name** – işarələmənin adıdır. Olduqca rahat vasitədir: lazımı işarələməni yalnız indekslə (nömrə ilə) yox, adı ilə də tapmaq olur;
- **Range** – həmin işarələmənin yerində **Range** obyektini yaradır;
- **Start**, **End** və **StoryType** - bu xassələr eynilə (eyni funksionallıq səviyyəsində) **Selection** obyektində də var.

Bookmark obyektinin cəmi üç sayda metodu vardır: **Copy()**, **Delete()** və **Select()**.

- **Copy()** – mövcud olan işarələmə əsasında yenisini yaradır;
- **Delete()** - mövcud olan işarələməni silir (yox edir);

- **Select** () – işarələmə istinad etdiyi hər nə varsa onu seçir.

11.6 Word-ün digər obyektləri

Word-ün obyekt modelində, 11.2-ci fəsildə qeyd etdiyimiz kimi, yüzlərlə obyektlər vardır – bu kitabın çərçivəsində onların hamısı haqqında məlumat vermək mümkün deyil. Bunu da qeyd etməliyik ki, ola bilsin, praktiki işlərdə onlardan bir çoxu yeni başlayan istifadəçiyə (və hətta peşəkar VBA proqramçısına belə) uzun müddət lazım olmasın. Yada salırıq ki, istənilən obyekt haqqında əlavə məlumatın ən asan alınması yolu – makrorekorderdən istifadə edilməsidir. Aşağıda, oxucunun hələ tanış olmadığı, Word-ün ən vacib obyektləri nəzərdən keçiriləcək. Onlardan bir çoxu **Application** obyektindəki eyni adlı xassələrindən əlçatandır və bu xassələr nəzərdən keçiriləndə onlar haqqında bəzi məlumatlar vermişdik.

11.6.1 AddIns kolleksiyası və AddIn obyektı

Word.AddIn obyektı, VBA-dan Word şablonları və qurmaları ilə işləmə qaydaları

AddIns kolleksiyası **AddIn** obyektlərindən ibarətdir, onlar isə, öz növbəsində, Word-ün qlobal şablonlarını və qurulmuş əlavələrini təmsil edir. Bu kolleksiyanın çox vacib imkanı ondan ibarətdir ki, **Add**() metodu ilə, avtomatik rejimdə, şablonları və əlavə qurmaları sazlamaq olur (adi üsulla, qrafik interfeys rejimdə, bu əməli **Tools**→**Templates and Add-Ins** menyusu ilə həyata keçirmək mümkündür). Əgər bu vasitədən proqramlarda fəal istifadə edilirsə, onda lazım olan şablon və ya əlavə qurmanın mövcudluğunun yoxlanmasının reallaşması haqqında düşünməyə dəyər. **Şablonlar (Templates)** - **.dot** genişlənməsi olan fayllardır və Word sənədlərinin yaradılmasında köməkçi vasitələr sayılırlar. Adətən daha tez-tez onları standartlaşdırması tələb olan mürəkkəb sənədlərin (məsələn, nəşriyyatlarda əlyazmalar və orijinal-maketlər hazırlanmasında) istismarçı-istifadəçidən qorumaq üçün ilkin hazırlanmış hesabatların və ya stillərin, makrosların, parametrlərin v.s. saxlanılan “deposu” kimi istifadə edirlər. Əlavə qurmalar – kompilyasiya edilmiş **DLL** (Dynamic-link library, **WLL** isə **Word Add-In Library**) modullarıdır. Onlar artıq kompilyasiya edilmişdirlər və **C++** dilinə (istənilən başqa **COM**-uyğunluqlu, yeni **Component Object Model** tipli, dildən də istifadə etmək olar, məsələn: **Visual Basic**, **Delphi** və ya **Python**-dan) yazıla bilirlər deyər, onlar daha tez işləyirlər, nəinki doğma olan **VBA** proqramları – makroslar. Buna görə əlavə qurmaların istifadə edilməsi üçün ciddi ehtiyac yaranır.

11.6.2 AutoCorrect obyektı

AutoCorrect obyektı, VBA-dan Word avto əvəz etmə parametrləri ilə işləmə qaydası

Bu obyektinin köməyi ilə, qrafik interfeys vasitəsilə **Tools**→**AutoCorrect Options** menyusunda təyin edilən parametrləri, proqram səviyyəsində tənzimləmək olur. Həmin əlavə qurmalar, istifadəçi

tərəfindən (“əl ilə”) daxil edilən, mətnin avtomatlaşdırılmış düzəlişlərin yerinə yetirilməsinə təsir edir. İş prosesində daha çox **Autocorrect** obyektini avto əvəzləməni söndürmək üçün istifadə edirlər – istifadəçinin işində bu rejim bəzən maneçilik törədir. Məsələn, qrafik pəncərədəki **Replace as you type** (tərcümədə - "daxil edildikdə - dəyişdirilməlidir") təlimatını bütün siyahısını təmizləmək üçün aşağıdakı VBA kodundan istifadə etmək olar:

```
For Each Item In AutoCorrect.Entries
    Item.Delete
Next
```

11.6.3 Languages kolleksiyası və Language obyektı

Language obyektı, VBA-da Word-ün qurulmuş dilləri ilə işləmə qaydası

Bu kolleksiyada kompyuterdəki Word versiyasında olan bütün dilləri təmsil edir və onlarla işləməyə imkan yaradır (ən qiymətli isə - orfoqrafiya və qrammatikanın yoxlanması üçün nəzərdə tutulan lüğətləri sonradan da yükləmək olar və ya mövcud olanlardan istifadə etmək mümkündür). Word-ün işləyə bildiyi, cari dillər yığımına baxmaq üçün aşağıdakı VBA kodundan istifadə etmək lazımdır:

```
For Each lan In Languages
    Debug.Print lan.Name
    i=i+1
Next lan
Debug.Print i
```

11.6.4 Options obyektı

Options obyektı və VBA-dan Word əlavə qurmaları ilə işləmə

Bu obyektin Word proqramının özünün və cari Word sənədinin tənzimlənməsi üçün olduqca çox sayda xassəsi vardır. Qrafik interfeysdəki **Tools→Options** menyusunda hər nə varsa burada əlçatandır və üstəlik əlavə parametrlərin olması mövcuddur. Bu obyektin alt obyektləri yoxdur – tərkibində hər nə varsa – xassələrindən əlçatandır. Məsələn, Word-də ağ rəngdə hərflərlə göy rəngli fonda yazı rejiminə keçmək üçün (bəzi istifadəçilər, peşəkarlar, bu rejimə daha çox üstünlük verirlər) aşağıdakı VBA proqram kodundan istifadə etmək lazımdır:

```
Options.BlueScreen=True
```

11.6.5 Find və Replacement obyektləri

Word.Find və Word.Replacement obyektləri, VBA vasitələri ilə Word-də mətnin axtarılması və əvəz edilməsi

Find və **Repalcement** obyektləri, adlarından da başa düşmək olar ki, axtarış və əvəz etmə əməlləri üçün nəzərdə tutulub. **Find** obyektı – proqram obyektinə “bükülmüş”, axtarış şərtləridir. Çox sayda xassəsi (**Text, Style, Font, Forward, MathCase, LanguageID** v.s.) var ki, həmin axtarış şərtlərini təyin edə bilir. Axtarışı işə salmaq üçün çox sayda məcburi olmayan parametrləri

var (onlar əksər hallarda həmin obyektin xassələrini dubl edirlər: **ReplaceWith** parametrini ötürüb, hətta mətnin əvəz edilməsinə nail olmaq olar). Hər şeyi dəyişmək və ya sadəcə axtarışın nəticəsini yoxlamaq üçün **Execute** () metodunun qaytardığı qiymət istifadə edilir. Əgər qiymət tapılıbsa, onda qaytarılan qiymət **True** olur - axtarış müvəffəqiyyətli olmuşdur, digər halda **False** qiyməti qaytarılır.

Find obyektinin necə işləməyi bu obyektin hansı obyekt əsasında yaradıldığından asılı olacaq. Məsələn, əgər obyekt **Selection** obyektinin **Find** xassəsi ilə yaradılmışdırsa, onda lazımı fraqment tapıldıqda elə onun özü seçilir. Əgər həmin obyekt **Range** obyektinin **Find** xassəsi əsasında yaradılmışdırsa, onda diapazonu tapılmış mətn ilə yenidən təyin edilir. Məsələn, "2004" mətninin növbəti girişini tapıb seçmək üçün bu VBA kodu tətbiq edilə bilər:

```
Selection.Find.Text="2004"  
Selection.Find.Execute
```

Replacement obyektini eynən dəyişdirilmənin əlavə qurmalarını saxlayır. Məsələn, sənədin sonuna qədər "2004" mətnini "2005" mətninə dəyişdirmək üçün aşağıdakı VBA kodu istifadə edilməlidir:

```
Selection.Find.Text="2004"  
Selection.Find.Replacement.Text="2005"  
Selection.Find.Execute Replace:=wdReplaceAll
```

Vacib məqamı qeyd edək: **Find** və **Replacement** obyektləri standart halda axtarışı və dəyişdirilməni, formatlaşmanı nəzərə alaraq, icra edirlər. Buna görə əgər sənəddə bir söz müxtəlif rənglə formalaşsa, onda həmin söz müxtəlif sözlər kimi sayılacaq. Məhz buna görə bəzi proqramçılar **ClearFormatting** () metodunu tətbiq edirlər. Bir çox VBA proqramçıları axtarış və əvəz etmə zamanı formatlaşdırma fərqi görə əmələ gələcək səhvlərdən özlərini sığortalamaq üçün həmçinin **ClearFormatting** () metodundan istifadə edirlər. Bu metod **Find** və **Replacement** obyektlərinin içərisində formatlaşmanı təmizləyir (sənədin özünə bununla heç bir təsir edilmir) və formatlaşma nəzərə alınmadan axtarışı icra etməyə imkan verir. Məsələn, yuxarıda gətirilən VBA nümunə kodunda bu metod tətbiq edilsə, aşağıdakı formada yazıla bilər:

```
Selection.Find.Text="2004"  
Selection.Find.ClearFormatting  
Selection.Find.Replacement.Text="2005"  
Selection.Find.Replacement.ClearFormatting  
Selection.Find.Execute Replace:=wdReplaceAll
```

11.6.6 Font və ParagraphFormat obyektləri

Word.Font və ***Word.ParagraphFormat*** obyektləri, VBA vasitələri ilə proqram səviyyəsində Word sənədinin formatlaşmasının dəyişdirilməsi

Bu obyektlər mətn hissələrinin və abzaslarının formatlaşmasına cavabdehdir. **Font** obyektinin xassələri qrafik interfeysin **Format**→**Font** menyusunda əlçatan olan bütün parametrləri təyin etməyə imkan yaradır, **ParagraphFormat** obyektinin xassələri isə qrafik interfeysin

Format→**Tabs** menyusundakı əlçatan bütün parametrləri təyin etməyə kömək edir. **Font** obyektini **Font** xassəsindən almaq olur. Bu xassə isə bir çox obyektlərdə mövcuddur, məsələn: **Selection**, **Range** və **Find**. **ParagraphFormat** obyektini isə - **Format** xassəsindən almaq mümkündür (**Paragraph** obyektində və **Paragraphs** kolleksiyasında). **Format** xassəsi **ParagraphFormat** obyektini qaytarır və həmçinin **Find** obyektinin də tərkibində vardır. **Font** və **ParagraphFormat** obyektlərinin xassələri çoxdur - onların hamısı aşkardır. Məsələn, seçilmiş mətnə Arial şriftini təyin etmək və işarələri yarım-qalın etmək üçün aşağıdakı VBA kodunu istifadə etmək olar:

```
Selection.Font.Name="Arial"  
Selection.Font.Bold=True
```

Başqa bir tərəfdən, mətnin birinci abzasının eninə görə düzəlişmə təyin etmək lazım olduqda, onda VBA kodu bu cür yazılmalıdır:

```
Paragraphs(1).Format.Alignment=wdAlignParagraphJustify
```

11.6.6. və 11.6.7 bölmələrə aid praktiki tərəfdən olduqca maraqlı olan bir tətbiqi məsələni aşağıda oxucuya təqdim edirik:

Word sənədlərinin standart kodlaşmaya keçirilməsi üçün təyin olunmuş makrosun nümunəsi (MS Windows-da əsgü Azərbaycan şriftlərini Unicod formatına çevirən VBA proqram kodu)

Windows-da əski (Unicod olmayan) şriftlərin (hərflərin) Unicod-a aid olan TimesNewRoman şriftinə avtomatik çevrilməsini təmin edən proqramın nümunəsi aşağıda verilib (N.Həsənov adında azərbaycanlı proqramçı bu Word VBA kodunu 05/03/2001 il tarixində tərtib edib – sərbəst olaraq, bir müddət əvvəl TQDK saytından bu kodu yükləmək olurdu). Oxucu aşağıda nümunə kimi gətirilən VBA kodunu Word-də uğurla istifadə edə bilər - proqram yazıldıqda onu **normal.dot** genişlənməsində yaddaşda təyin etmək lazımdır (əvvəlki hissələrdə dediyimiz kimi: belə olduqda bu makros bütün Word sənədlərində görünən olacaq və hər bir sənəddən işə salına biləcək).

Makrosun tərkibi:

- **Targets** - dəyişəni ardıcılıqla düzölmüş Azərbaycan əlifbasının latın qrafikalı hərflərinin Unikod kodlarını təsvir edir (əvvəlcə baş hərflər, sonra isə sətir hərfləri göstərilmişdir);
- **Source** - dəyişəni isə Arial Az Cyr (və ya Arial Az Lat) şriftinin müvafiq hərflərinin kodlarını göstərir.

Bu makros seçilmiş sahədə və ya kursurun mövqeyindən aşağı hissədə olan cari Word sənədinin mətnini çevirir. Makrosu işlətmək etmək üçün əsas menyudan (**Tools**→**Macro**→**Macros**→**Convert**→**Run**) və ya digər alternativ üsullardan istifadə edilməlidir. Bu prosesi asanlaşdırmaq üçün makrosa xüsusi düymə və ya nişan təyin etmək olar. Nümunə kimi təqdim olunmuş makrosda düzəlişlər edərək Word sənədlərinin çevrilməsi üçün öz makrosunuzu yarada bilərsiniz. **Source** dəyişəninə aid olan kodları dəyişdirməklə digər qeyri-standart şriftlərə müvafiq olan sənədlərin çevrilməsi üçün makros hazırlamaq mümkündür.

```
Sub Convert()  
' Created 05/03/2001 by Nasreddin Hassanov  
Dim Target As String, Source As String, SelText As Selection
```

```

Target=ChrW(65)+ChrW(66)+ChrW(67)+ChrW(199)+ChrW(68)+ChrW(69)+_
ChrW(399)+ChrW(70)+ChrW(71)+ChrW(286)+ChrW(72)+ChrW(88)+_
ChrW(73)+ChrW(304)+ChrW(74)+ChrW(75)+ChrW(81)+ChrW(76)+_
ChrW(77)+ChrW(78)+ChrW(79)+ChrW(214)+ChrW(80)+ChrW(82)+_
ChrW(83)+ChrW(350)+ChrW(84)+ChrW(85)+ChrW(220)+ChrW(86)+_
ChrW(89)+ChrW(90)+ChrW(97)+ChrW(98)+ChrW(99)+ChrW(231)+_
ChrW(100)+ChrW(101)+ChrW(601)+ChrW(102)+ChrW(103)+ChrW(287)+_
ChrW(104)+ChrW(120)+ChrW(305)+ChrW(105)+ChrW(106)+ChrW(107)+_
ChrW(113)+ChrW(108)+ChrW(109)+ChrW(110)+ChrW(111)+ChrW(246)+_
ChrW(112)+ChrW(114)+ ChrW(115)+ChrW(351)+ChrW(116)+ChrW(117)+_
ChrW(252)+ChrW(118)+ChrW(121)+ChrW(122)

```

```

Source=ChrW(1040)+ChrW(1041)+ChrW(1066)+ChrW(1063)+ ChrW(1044)+_
ChrW(1045)+ChrW(1071)+ChrW(1060)+ChrW(1069)+ChrW(1068)+_
ChrW(1065)+ChrW(1061)+ChrW(1067)+ChrW(1048)+ ChrW(1046)+_
ChrW(1050)+ChrW(1043)+ChrW(1051)+ChrW(1052)+ ChrW(1053)+_
ChrW(1054)+ChrW(1070)+ChrW(1055)+ChrW(1056)+_
ChrW(1057)+ChrW(1064)+ChrW(1058)+ChrW(1059)+ChrW(1062)+_
ChrW(1042)+ChrW(1049)+ChrW(1047)+ChrW(1072)+ChrW(1073)+_
ChrW(1098)+ChrW(1095)+_
ChrW(1076)+ChrW(1077)+ChrW(1103)+ChrW(1092)+ChrW(1101)+_
ChrW(1100)+ChrW(1097)+ChrW(1093)+ChrW(1099)+ChrW(1080)+_
ChrW(1078)+ChrW(1082)+ChrW(1075)+ChrW(1083)+ChrW(1084)+_
ChrW(1085)+ChrW(1086)+ChrW(1102)+ChrW(1087)+ChrW(1088)+_
ChrW(1089)+ChrW(1096)+ChrW(1090)+ChrW(1091)+ChrW(1094)+_
ChrW(1074)+ChrW(1081)+ChrW(1079)

```

```

Set SelText = Application.Selection
With SelText.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Forward = True
    .Wrap = wdFindStop
    .format=False
    .MatchCase=True
    .MatchWholeWord=False
    .MatchWildcards=False
    .MatchSoundsLike=False
    .MatchAllWordForms=False
End With
For i=1 To Len(Source)
    SelText.Find.Text=Mid(Source, i, 1)
    SelText.Find.Replacement.Text=Mid(Target, i, 1)
    SelText.Find.Execute Replace:=wdReplaceAll
Next
SelText.Font.Name="Times New Roman"
End Sub

```

11.6.7 PageSetup obyektı

Word.PageSetup obyektı, VBA vasitələri ilə proqram səviyyəsində çapdan qabaq səhifənin tənzimlənməsi

Əgər tərtib edilən tətbiqi proqramda çap etmə əməli lazımdırsa (qeyd edək ki, bu əməl əksər hallarda bütün proqramlarda lazım olur), onda **PageSetup** obyektı hökmən lazım olacaq. O, qrafik interfeysin **File→Page Setup** menyusundakı tənzimlənen hər nə varsa, proqramlaşdırma üsulları ilə təyin edir. **PageSetup** obyektı **Document**, **Selection** və **Range** obyektlərinin içərisində qurulmuş olur və ona olan istinadlar həmin obyektlər vasitəsi ilə həyata keçirilir. Məsələn, sənəd

çap edildikdə onu albom oriyentasiyasında çıxarmaq üçün bu VBA komandasından istifadə etmək olar:

```
ThisDocument.PageSetup.Orientation=wdOrientLandscape
```

11.6.8 Table, Column, Row və Cell obyektleri

Word.Table obyektini, VBA proqramlaşdırması ilə Word sənədində cədvəllərlə işləmə, avtocəmləmə əməlinin proqramlaşdırılması

Əgər Word proqramında hansısa standart sənədi təsdiq edilmiş formada (məsələn, ödəmələr, kassa orderi, ezamiyyət hesabatı v.s.) yerləşdirmək lazımdırsa, onda, ehtimalı çoxdur ki, bunun üçün cədvəl formaları tətbiq edilməlidir. Çünki cədvəl forması əksər hallarda, bir-biri ilə asılılıqda olan, verilənlərin daha düzgün və daha sahmanlı yerləşdirilməsinə imkan yaradır. Buna görə bir çox proqramçılar əvvəlcədən hətta forma cədvəllərindən istifadə edirlər (əvvəlcədən bilərək ki, onlar cədvələ bir o qədər də oxşarırlar. Bununla belə – cədvəlin xətlərini hər vaxt gizlətmək və hücrələri (xanaları) isə bir-biri ilə birləşdirmək olur. Cədvəlin yaradılması ondan başlayır ki, **Tables** kolleksiyasına (o isə **Document**, **Selection** və **Range** obyektlərində nəzərdə tutulub) yeni obyekt **Table** əlavə edilir (aşağıda baxılacaq VBA nümunəsi işlədikdə bu obyekt - üç sətirli və dörd sütunlu olacaq):

```
Set Range1=ThisDocument.Range(Start:=0, End:=0)
Dim Table1 As Table
Set Table1=ThisDocument.Tables.Add(Range1, 3, 4)
```

Sonra isə cədvəlin xassələrini qurmaq olar, məsələn, **AutoFormat()** metodundan istifadə edərək (onun imkanları qrafik interfeysin **Table**→**AutoFit** menyusundan əlçatan olanlarla eynidir):

```
Table1.AutoFormat wdTableFormatGrid 5
```

Nəticə etibarlı ilə, əsas əməllər kimi, cədvəlin hücrələrinə hansısa verilənlərin daxil edilməsi lazım olur. Lazım olan hücrəyə **Columns** və **Rows**, o biri tərəfdən **Selection** və **Range** obyektinin istifadəsi ilə çatmaq olar. Təcrübəli proqramçılar daha əlverişli üsuldən istifadə edirlər, aşağıdakı nümunədə göstərilmiş VBA proqram kodu ilə:

```
Table1.Cell(1,1).Range.InsertAfter "10"
Table1.Cell(2,1).Range.InsertAfter "15"
Table1.Cell(3,1).AutoSum
```

Yuxarıdakı VBA kodunu işlədikdə bu əməllər icra edilir: cədvəlin birinci sütununun birinci sətirinə 10 qiyməti daxil edilir, birinci sütunun ikinci sətirinə 15 qiyməti daxil edilir, üçüncü sətirdə isə birinci sütundakılar toplanır. Nəzərə alınmalıdır ki, Word cədvəlləri, əlbəttə, Excel cədvəlləri deyil. **Formula()** metodunun tətbiq edilməsi isə **Cell** obyektini üçün cədvələ istənilən çətinlikdə hesablanan qiymətləri daxil etmək olur. Əgər Word yalnız ən geniş məqsəddən ötrü istifadə edilirsə (məsələn, başqa proqramlardan və ya verilənlər bazasından alınmış verilənlərin daxil edilməsi), onda ehtimal çoxdur ki, bu halda VBA Word proqramlaşdırılması, ümumiyyətlə, gərək olmasın. Bu halda – qrafik ekranda cədvəlin şablonunda yaradılması və sonra verilənlərin daxil ediləsi yerləri işarələmələrlə qeyd edilməsi kifayət edəcək və bu üsul ən əlverişli üsul sayıla bilər.

11.6.9 System obyektı

Word.System obyektı, proqram səviyyəsində VBA vasitəsilə əməliyyat sistemindən məlumatın alınması

System obyektı ilə, əlavə işlədiyi kompyuterin (serverin) sistemi haqqında, böyük həcmdə məlumat almaq olur. Bu obyektə alternativ olan **Windows Script Host** və **WMI (Windows Management Instrumentation)** obyekt modelləri də istifadə edilə bilər - sistem haqqında daha böyük həcmdə məlumat alınması bu obyekt modellərinin tətbiqi ilə əldə etmək olar. Bir çox hallarda həmin məqsədlə məhz **System** obyektindən proqramçılar istifadə edirlər – çünki bu obyekt bütün əməliyyat sistemləri üçün tətbiq edilə bilər (yuxarıda adı çəkilən alternativ olan obyektlər yalnız Windows-un ən yeni versiyalarına tətbiq edilə bilər (məsələn, Windows-Vista, Windows-7 və daha yeni versiyalarla).

System obyektinin ən vacib xassələri:

- **CountryRegion** – əməliyyat sisteminin cari regional qurmalarını qaytarır. Məsələn, Rusiya ilə bağlı qurmalar üçün 7 qiyməti qaytarılır, ABŞ üçün isə 1 qiyməti qaytarılır;
- **FreeDisk** – cari diskin əlçatan sahəsinin həcmnin qiymətini qaytarır (cari diski dəyişmək də olar). Əgər sənədin (sənədlərin) həcmi çox böyükdürsə, onda diskdə boş olan yerin yoxlanmasını icra etmək olar, məsələn, yaddaşda saxlama əməli yerinə yetirildikdə;
- **HorizontalResolution** və **VerticalResolution** – istifadəçi kompyuterinin ekranının qrafik imkanları haqqında məlumat almağa imkan yaradır (böyük miqyasda olan formaların əks edilməsini təyin edilməsində faydalı olur);
- **LanguageDesignation** - əməliyyat sisteminin qrafik interfeysinin dilinin təyin edilməsində istifadə edilir. Sətir qiymətilə qaytarılır, məsələn:

```
Debug.Print System.LanguageDesignation
```
- **OperationSystem** - əməliyyat sistemi haqqında məlumat qaytarır;

System obyektinin cəmi iki sayda metodu var:

- **Connect ()** – şəbəkə diskini qoşmaq;
- **MsInfo ()** – sistem məlumatı pəncərəsinin göstərilməsi.

11.6.10 Tasks kolleksiyası və Task obyektı

Word.Task obyektı, VBA vasitəsi ilə proqram səviyyəsində Word-dən tətbiqi proqramların işə salınması

Daha çox Word-ü Excel, Access və digər Office proqramlarından işə salırlar. Bəzən əks hallara da rast gəlmək olur – Word-dən başqa əlavə işə salınmalı və ona keçid olmalıdır. Ən sadə üsul –

başqa proqramı Word-dən işə salmaqdır (**VBA Shell** standart obyektindən istifadə etməklə). Məsələn, ən sadə halda bloknotu Word-dən işə salmaq üçün bu komandadan istifadə etmək olar:

```
Shell ("notepad.exe")
```

Başqa imkanlar da var, məsələn, Word-ün digər sənədlərində olan **Application** obyektindən istifadə etmək və ya WSH (Windows Script Host) vasitələrindən (xüsusilə konsol tipli proqramlar üçün) və ya əlavə başqa kompyuterdən işə salınacaqsa, onda WMI (Windows Management Instrumentation) vasitələrindən istifadə etmək lazım olacaq. Proqram işə salındıqdan sonra, bütün işləyən proqramlar toplusu Word-də **Tasks** kolleksiyası ilə təmsil olur, hər bir proqramı işə uyğun olan **Task** obyektini təmsil edir. **Tasks** kolleksiyasında iki maraqlı metod var:

- **Exists()** – lazım olan tətbiqi proqramın işə salınmasını yoxlayır, məsələn, bizim bloknotumuzun yoxlama ilə işə salınması bu cür VBA kodu ilə həyata keçirilə bilər:

```
If Tasks.Exists("Notepad")=False Then
    Shell "notepad.exe"
Else
    Tasks("Notepad").Activate
End If
Tasks("Notepad").WindowState=wdWindowStateMaximize
```

- **ExitWindows()** – metodu **Log Off** əməliyyatını həyata keçirir, yəni Windows-da iş seansını bitirmək. Yadda saxlanılmayan Word sənədləri bu halda yadda saxlanılmamış da bağlanacaq (istifadəçiyə yadda saxlamaq haqqında heç bir dialoq xəbəri gəlmədən). Digər proqramların yadda saxlanması üçün istifadəçiyə dialoq pəncərəsində yadda saxlamaq haqqında təklif gələcək.

Task obyektinin maraqlı olan (vaciblik baxımından) xassəsi və metodları daha çoxdur:

- **Height, Width, Top, Left** – seçilmiş proqramın pəncərəsinin düzgün təyin edilməsinə kömək edir;
- **Visible** - proqramın gizlədilməsinə imkan yaradır;
- **WindowState** – pəncərənin açılması, bağlanması və bərpa edilməsi üçün imkan yaradır;
- **Activate(), Close(), Move(), Resize()** – metodların təyinatları aşkardır;
- **SendMessage()** - ən maraqlı metoddur: proqramın pəncərəsinə Windows məlumatlarının verilməsinə imkan verir (mausun şıqqılması, klavişin basılması v.s.). "Proqramların pəncərəsinə hansı əlavələri göndərmək olar" sualının cavabını tez tapmaq üçün **Microsoft Platform Software Development Kit** ilə tapmaq mümkündür. Məsələn, bizim bloknotda "Bloknot haqqında" pəncərəsini yaratmaq üçün bu əmrdən istifadə etmək olar:

```
Tasks("Notepad").SendMessage &H111, 11, 0
```

11.6.11 Windows kolleksiyası və window obyektı

Word.Window obyektı, VBA vasitəsilə proqram səviyyəsində Word-də sənədlərin pəncərəsi ilə işləmə

Windows kolleksiyası və Window obyektı açıq olan Word sənədlərinin pəncərələrini təmsil edirlər və əsasən həmin pəncərələrin xarici görüntüsünü qurmaq və pəncərə üzrə naviqasiya etmək üçün nəzərdə tutulublar. Lazımı sənədin pəncərəsinə yol tapmaq üçün aşağıdakı VBA kodundan istifadə etmək mümkündür:

```
Dim Window1 As Window  
Set Window1=Windows ("doc2.doc")
```

və ya bu cür :

```
Set Window1=ThisDocument.ActiveWindow
```

Bundan sonra Window obyektinin xassələri və metodlarından istifadə etmək olar. Onlar olduqca sadədir, məsələn, pəncərənin başlığını dəyişmək üçün bu VBA kodundan istifadə etmək olar:

```
Window1.Caption="Mənim proqramım"
```

12. MS Excel-də PROQRAMLAŞDIRMA

12.1 Excel-də proqramlaşdırma hansı hallarda lazım olur

VBA tətbiqi proqramlarının Excel-də yaradılması, praktikada tipik yarana bilən hallar haqqında

Excel - əslində, proqramlaşdırma nöqtəyi nəzərdən, ən çox istifadə edilən Microsoft Office toplusuna aid olan proqramdır [1, 3, 4, 5, 8, 9, 13, 16, 17]. Bir çox hallarda (şəxsi təcrübəmizə əsaslanaraq təsdiqliyə bilirik ki, əksər hallarda) müəssisə və şirkət işçilərini məhz Excel-də aparılan işlərin avtomatlaşdırılması xüsusilə maraqlandırır. Müəssisələrdə (özəl və ya dövlət) isə daha çox rast gələn hallar bunlardır:

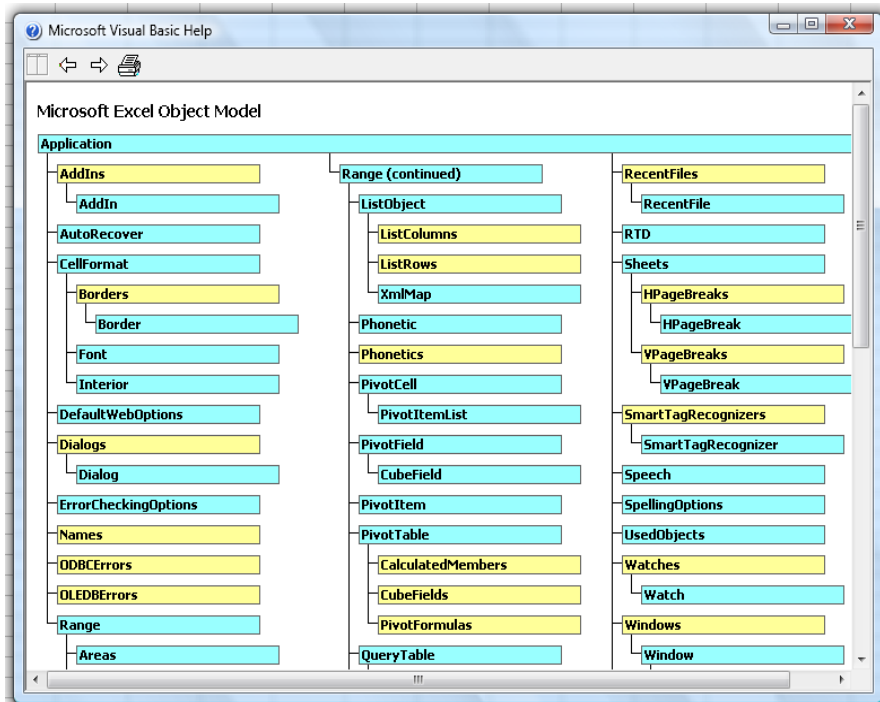
- verilənlər bazasından Excel cədvəlinə verilənlərin yüklənməsini avtomatlaşdırmaq lazımdır. Sonra isə həmin cədvəli avtomatik rejimdə emal etmək tələb olur (müxtəlif çətinlik səviyyəsində hesablamalar: ola bilsin modelləşdirmə (ola bilsin hətta real zaman miqyasında) ilə bağlı məsələlərin həlli, adi hal kimi, məsələn, verilənlər bazası ilə aparılan əməllərin icrası v.s. Son mərhələdə, avtomatlaşdırılmış rejimdə, alınan hansısa nəticələri (məlumatı) standart raport formasında təqdim etmək (ola bilsin, "online" rejimində, İnternet vasitələri ilə) lazımdır. Adi hallarda istifadəçilərin bunun üçün tələb olan bilik və bacarıqları olmur və ya server verilənlər bazası ilə işləmək üçün ixtiyarları çatmır. Buna görə belə hallarda Excel proqramlaşdırması yeganə və əvəz edilməz vasitə olur;
- birinci situasiyanın başqa variantı – verilənlər bazası ilə birgə çalışan proqram, avtomatik rejimdə hesabatları və raportları Excel faylları formatında generasiya edə bilir. Vaxt

keçdikcə, hesabatlara olan tələblər dəyişir – yeni hesabatlara və ya köhnələrin dəyişdirilməsinə ehtiyac yaranır. Bu cür qaçılmaz işlərdə hökmən rutin, çox sayda təkrar olunan və olduqca yorucu məqamlar yaranır. Buna görə işlərin avtomatlaşdırılması zəruriyyəti yaranır - istifadəçi üçün Excel VBA proqramlaşdırması yeganə çıxış yolu olur;

- çox hallarda, professional proqramçılara müraciət etməyə imkan tapmayaraq, istifadəçi, müstəqil olaraq, Excel cədvəllərində öz proqramlarını reallaşdırır. Əksər təşkilatlarda, məsələn, maliyyə planlaşdırmaları və büdcə tərtibi sadəcə Excel faylları çoxluğu səviyyəsində icra edilir. Bu halda Excel bir neçə rolu icra edəsi olur: verilənlər bazasının icraçısı kimi, müştəri proqramı kimi və müxtəlif hesabatlar və raportlar generatoru kimi. Bu cür hallarda da zamana qənaət mənasında hesablama proseduralarının avtomatlaşdırılması olduqca kəskin məsələ kimi meydana çıxır;
- adətən müəssisələrin filliallarındakı, bölmələrdəki, şöbələrdəki məlumatlar, işçilər haqqında olan məlumatlar v.s. Excel faylları formatında yığılır. Müəyyən vaxt keçdikdə isə belə vacib məsələ qalxır – verilənlər bazasına Excel-də yığılmış məlumatların yüklənməsini necə avtomatlaşdırmaq olar. Excel fayllarının formatları yalnız məlumatın verilənlər bazasından çıxarılmasında əlverişli və rahat vasitə deyil, üstəlik “əl ilə” verilənlər bazasına məlumatın yüklənməsində də olduqca əlverişli vasitə sayılır;
- təcrübədən məlumdur ki, digər çox rast gəlinən hal budur: müəssisələrdə Excel faylları və verilənlər bazası (və ya başqa Excel faylları yaxud DBF faylları v.s.) arasında sinxronlaşdırmanın yaradılmasına ehtiyac yaranır. Məsələn, istifadəçi məlumatını Excel faylına daxil edərkən, necə etmək olar ki, həmin məlumat avtomatlaşdırılmış rejimdə həmçinin dərhal verilənlər bazasına əlavə olunsun.
- təhsil sahəsində və elmi təşkilatlarda Excel VBA ilə proqramlaşdırmaq üçün böyük ehtiyac yarana bilər. Məlumdur ki, informatikanın (və ya kompyuter elminin) ilkin kursunun universitet tədrisində, proqramlaşdırma ilə bağlı hissəsini, ənənəvi prosedura proqram təminatları əsasında keçməyə üstünlük verirlər: məsələn, Pascal dilində (xüsusilə Azərbaycan universitetlərində). Ən yaxşı halda Delphi (Object Pascal) və ya C++ dillərindən tədrisdə istifadə edilə bilər. Nəzərə alınsa ki, VBA Excel riyazi məsələlərin həlli üçün də çox güclü vasitədir – bu vasitənin tədrisdə öyrənilməsi və elmi araşdırmalarda tətbiq edilməsi qaçılmaz olacaq. Xüsusilə nəzərə alınsa ki, VBA proqramlaşdırma mühiti bütün kompyuterlərdə əlçatandır, MS Office proqramlarında (Excel-də) inteqrə olmuş haldadır – bu işə icra edilmiş istənilən riyazi hesablamaların sonradan qısa zamanda nəşr edilməsini olduqca rahat və əlverişli edir. ABŞ-ın bir çox aparıcı universitetlərində Excel VBA proqramlaşdırılması nəinki iqtisadiyyat təməyüllü ixtisaslarda, hətta Aviasiya yönü ixtisaslarda belə tələbələrə tədris edilir (məsələn, turbo-reaktiv mühərrikin modelləşdirilməsini icra etmək üçün, aerodinamikaya aid mürəkkəb qeyri xətti xüsusi törəməli və adi diferensial tənliklərin ədədi həllini VBA Excel mühitində almaq üçün). Həmin səbəbə görə, kitabın bu hissəsindəki bir bölmə adi diferensial tənliklər sistemlərin ədədi

həllinin alınmasında və bir sıra başqa ənənəvi riyazi məsələlərin həllində VBA Excel proqramlaşdırmasının tətbiqinə həsr edilib. Hesab edirik ki, informatika üzrə universitet (və hətta orta məktəb) müəllimləri və ən əsas universitet təmayüllü oxucu üçün də həmin bölmədəki biliklərin alınması gələcək təhsil və elmi işlərində çox faydalı olacaq (müxtəlif ixtisaslı tələbə, müəllim və magistrələr üçün nəzərdə tutulub).

Yuxarıda sadalanan hallarda yaranan məsələlərin həll üsulları bu fəsilə oxucunun nəzərinə çatdırılacaq. Ümid edirik ki, praktiki işlərdə həmin məsələlərin həllində bu kitabın oxucusu (əlbəttə, əgər aktiv işləsə) çətinlik çəkməyəcək. Proqramlaşdırma nöqteyi nəzərdən Excel, Word-dən fərqli olaraq, məlumatın yalnız daxil edilib redaktə edilməsi üçün yox, daha geniş mənada istifadə edilir – müxtəlif çətinlik səviyyəsində olan hesablamaların aparılması və onların nəticəsinin xüsusi formatlarda əks edilməsi (müxtəlif qrafiklər, yekun cədvəllər formasında v.s.). Əgər verilənlərin həcmi çox böyükdürsə (məsələn, sifarişçilər/müqavilələr/çatdırılmalar haqqında olan məlumatların saxlanması lazımdırsa), onda Excel + verilənlər bazası bağlantısı (bu cür bağlantı çox əlverişli, rahat və məhsuldar ola bilər) haqqında qərar qəbul etməyə ciddi səbəblər yaranır. Word-dən fərqli olaraq, Excel kitabları və səhifələri üzrə naviqasiya aparılması daha rahatdır – çünki hər bir hücrənin (xananın) öz ünvanı var (hətta iki ünvanı var – A1 formatında və R1C1 formatında). Bundan əlavə Excel-də hücrələr diapazonuna ad verilməsi imkanı var – bu isə çox əlverişli və rahat vasitədir. Excel-də standart obyektlər iyerarxiyası daha böyükdür, şək. 12.1 (şəkildə obyektlər modelinin yalnız bir hissəsi görsənir - əslində daha çox sayda obyektlər mövcuddur). Əgər, məsələn, Word-də obyekt modeli, əsasən üç sayda obyekt üzərində qurulubsa: **Application - Document - Range**, Excel-də isə üstəlik səhifə obyekt var. Buna görə Excelin obyekt modeli iyerarxiyası bu cür görünür: **Application - Workbook (kitab) - Worksheet (səhifə) - Range (diapazon)**.



Şəkil 12.1 VBA proqramlaşdırmasında istifadə edilən MS Excel obyekt modelləri.

Excel-də çox geniş və zəngin miqyasda qurulmuş funksiyalar nəzərdə tutulub (statistik, maliyyə, riyazi, mühəndis v.s. sahələrini əhatə edən) və onları həqiqətən ciddi işlər üçün olan proqramlarda istifadə etmək olur. Excel-də rahat və əlverişli mühitdə işləmək üçün Microsoft onu xaricdə olan bir neçə proqramlarla və xüsusi obyektlərlə təchiz edib: məsələn, **PivotTable** obyekt (yekun cədvəl), **OLAP**-müşəri əlaqəsi (Online analytical processing, **Panorama Software** şirkətindən alınıb və Excel-ə inteqrə edilmişdir), **QueryTable** – informasiya bazaları ilə birgə işləmək imkanını yaradan xüsusi obyekt, **Chart** – diaqramlar və qrafiklərlə işləmək üçün nəzərdə tutulmuş vasitədir (obyekt) v.s.

12.2 Application obyekt

Excel.Application obyekt, **VBA-dan yeni Excel nüsxəsinin işə salınması, Excel.Application** obyektinin hadisələri

Word-də olan kimi, **Application** obyekt Microsoft Excel-də bütöv Excel kitabını təmsil edir və, iyerarxiya baxımından, Excel-in obyekt modelində ən yuxarı səviyyədə yerləşir (baş yeri tutur). Word-də olan kimi, əgər Excel-i başqa proqramdan çağırmaq lazımdırsa, onda **Excel.Application** obyektini yaratmaq lazım olacaq (yənə də yada salırıq ki, **Tools**→**References** menyusu ilə **Microsoft Excel 11.0 Object Library** kitabxanasına istinad yaradılmalıdır). Bu obyektin yaradılması aşağıda gətirilən VBA kodu ilə reallaşdırıla bilər:

```
Dim oExcel As New Excel.Application
oExcel.Workbooks.Add
oExcel.Visible=True
```

Eynilə Word-dəki kimi, əgər işə salınmış Excel-də iş aparılırsa, onda **Application** obyektinin yaradılmasına ehtiyac olmayacaq: o, həmişə əlçatan vəziyyətdədir. Əgər hansısa xassəyə yuxarıda duran obyektə istinad edilirsə, onda Excel-dəki Visual Basic redaktoru hesab edəcək ki, **Application** obyektinə istinad edilibdir. Buna görə aşağıdakı iki VBA kod sətiri eyni güclüdür:

```
Application.Workbooks.Add
```

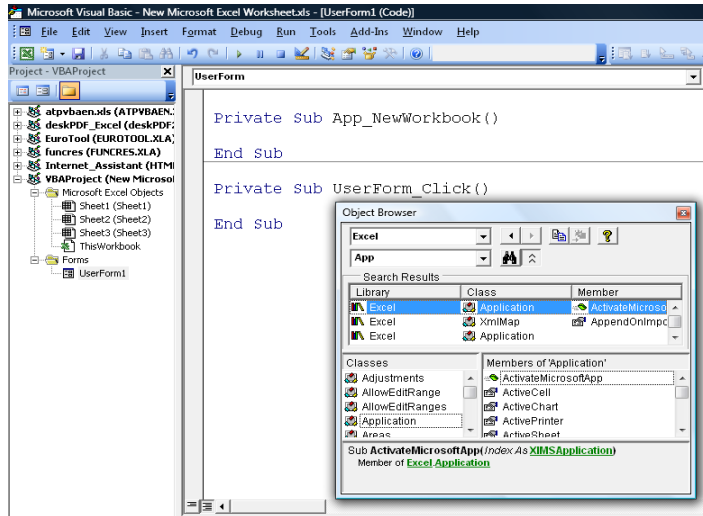
və

```
Workbooks.Add
```

Ümumiyyətlə, MS Office-in bir çox proqramlarında **Application** obyektləri bir-birinə çox bənzəyir və onlara **Word.Application** obyekt ilə bağlı tədbirlər tətbiq edilir. Eynilə Word-də olan kimi, tərbiatçıların əksəriyyəti belə hesab edirlər ki, Excel-in gizli pəncərəsi ilə işləmək daha etibarlı və rahat üsuldur. Analoji olaraq, onlar hesab edirlər ki, yeni Excel nüsxəsinin açılması, istifadəçi tərəfindən artıq açılmış nüsxənin axtarılmasından daha rahatdır. Eyni qaydada, VBA formaları üçün kod redaktoru pəncərəsində **Application** obyektinin əmələ gəlməsindən ötrü, gerek forma kodunun **Declaration** hissəsində **Application** obyekt **WithEvents** açar sözü ilə elan edilsin. Məsələn, aşağıdakı nümunədə olan kimi:

Public WithEvents **App** As Excel.Application

Bu halda formaların kod redaktoru pəncərəsində yeni **App** obyektini əmələ gələcək və istifadəçi **Application** obyektinin hadisə yönlü proseduraları ilə işləmək üçün imkan tapacaq (şək. 12.2).



Şəkil 12.2 Obyektlər siyahısında, öz hadisələr yığımı ilə birgə, yeni **App** obyektinin əmələ qəlməsi.

12.3 Application obyektinin xassələri və metodları

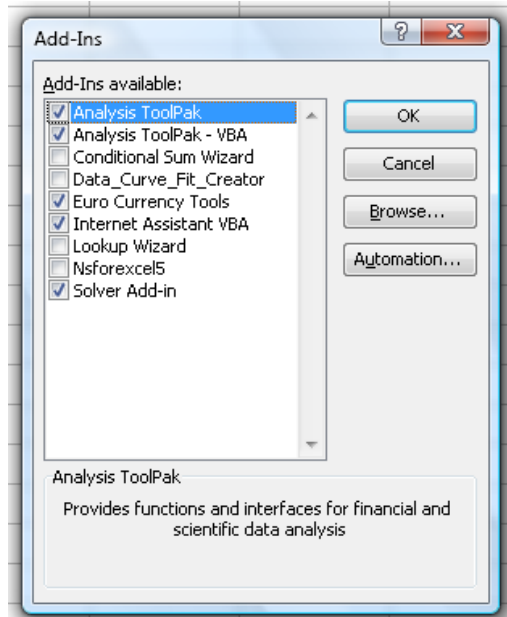
Excel.Application *obyektini, metodları və xassələri*

Excel.Application obyektinin bir çox xassələri və metodları, analogi olaraq, **Word.Application**-də olanlarla üst-üstə düşür. İngilis dilində sərbəst oxuyub anlama bilməyən oxucuları nəzərə alaraq, burada ən vacib olan və Excel-də daha çox istifadə edilən **Application** obyektinin xassələri və metodları haqqında ətraflı məlumat verəcəyik (oxucunun həmin xassələr və obyektlərlə Word-də rast qəlib-qəlməməyinə baxmayaraq). Həmçinin nəzərə alınmalıdır ki, obyektlərin xassə və metodlarında analogiya olsa da, hər halda, Excel proqramı Word-dən kəskin olaraq fərqlənir: bu isə obyektlərin xassələri və metodlarında öz əksini tapmaya bilməz.

Əvvəlcə - **Application** obyektinin xassələri haqqında:

- **Active...** - adı ilə başlayan xassələr, uyğun olaraq, aktiv hücrəni (həminə ki, verilənləri daxil edən kursoru göstərir), aktiv diaqramı, aktiv səhifəni, aktiv kitabı, aktiv pəncərəni qaytarır. Bütün bu xassələr yalnız oxuma üçün əlçatan olur. O biri tərəfdən, onların obyekt yaratmaq üçün istifadə edilməsi məcburi deyil - **ActiveCell**, **ActiveSheet** v.s. obyektləri, əlavə işlədiyi andan başlayaraq, avtomatik olaraq yaranır və hər vaxt əlçatan olur. Yalnız **ActivePrinter** xassəsinin bir qədər fərqi vardır – o, yalnız aktiv printeri obyekt kimi qaytarmır, üstəlik bu obyektini saxlamaq üçün imkan yaradır.
- **AddIns** - xassəsi eyni adlı əlavə qurmalar kolleksiyasını qaytarır (**AddIn** obyektini). Word-dən fərqli olaraq (Word-də əksər hallarda əlavə qurmaların istifadəsi yalnız peşəkar VBA proqramçıları üçün nəzərdə tutulub), Excel-də bu obyektə işləmə imkanı hər bir istifadəçi üçün vacibdir. Bir neçə çox xeyirli əlavə qurma Excel tərkibində qurulmuş olur və onları

qrafik interfeysin **Tools**→**Add-Ins** menyusundan təyin etmək olar. Məsələn, **Analysis ToolPak**, **Analysis ToolPak-VBA**, **Solver Add-in**, **Internet Assistant VBA** v.s. , şəkl. 12.3.



Şəkil 12.3 Excel tərkibinə təchiz edilmiş çox xeyirli olan əlavə qurmaların seçilməsini təyin edən **Add-Ins** dialoq pəncərəsi.

Bu kolleksiyanın köməyi ilə istifadəçi tərəfindən bu və ya digər lazım olan əlavə qurmanın qurulmuş olduğunu yoxlamaq (əgər o tərtib edilmiş VBA proqramında istifadə ediləcəksə) və lazım olduqda onu qoşmaq olar.

- **AutoRecover** – eyni adlı obyektı qaytarır: Excel-in avtoyaddaşda saxlama parametrlərini təyin etmək olar. Məsələn, açılmış Excel sənədlərini hər 5 dəqiqədən bir avtoyaddaşda saxlamaq üçün bu VBA kodu tətbiq edilə bilər:

```
Application.AutoRecover.Time=2
```

Zaman dəqiqələrlə verilir və 1, ..., 120 intervalındakı qiymətlər içərisindən təyin edilə bilər. Həmin əməli qrafik ekranda **Tools**→**Options**→**Save** menyusunu ilə də təyin etmək olar.

- **Calculation** – xassəsi Excel iş kitabının elementlərinin hesabatını (sayılmasını) icra etməyə, təyin etməyə və ya görməyə imkan verir (standart halda – avtomatik rejimdə, “əl ilə” hesabatda və ya yarım avtomatik rejimdə, cədvəllərdən başqa, hər şeyin sayı təyin edilir). Burada bir incəlik var – hər dəfə hücrədə qiymət dəyişdirildikdə sayılmalar çox vaxt aparır və verilənlərin daxil edilməsinə maneçilik törədə bilər. Buna görə avtomatik sayılmanı söndürmək məqsədə uyğundur. Həmin imkanı qrafik interfeysin **Tools**→**Options** menyusunun **Calculation** əlavə qurması ilə də sazlamaq olar.
- **CalculationState** – Excel-in verilənləri sayması ilə məşğul olduğunu və ya bu əməli bitirdiyini bilməyə (yoxlamağa) imkan verir.
- **Cells** – bu xassə **Application** obyektinin ən vacib xassələrindəndir: aktiv kitabın aktiv səhifəsinin bütün hücrələrini təmsil edən **Range** obyektini qaytarır. Standart halda **Range**

obyektinin xassəsi **Item**-dir deyə, aktiv səhifənin hücrələrinə istinadı aşağıdakı nümunədəki VBA kodu ilə etmək olar:

```
Application.Cells(1,2).Font.Bold=True
```

Bu halda birinci sətirin və ikinci sütunun kəsişməsində olan hücrədəki qiyməti biz yarım qalın formatda seçmiş olacağıq. Buna oxşar **Columns** və **Rows** xassələri fəaliyyət göstərir. Məsələn, həmin əməli bütöv sütunla (ikinci sütun ilə) etmək üçün bu VBA kodu tətbiq edilə bilər:

```
Application.Columns(2).Font.Bold=True
```

Sətir (ikinci sətir üçün) üçün isə oxşar əmr bu cür yığıla bilər:

```
Application.Rows(2).Font.Bold=True
```

Yada salaq ki, bir çox istifadəçilər hesab etdiyi kimi, **Cells**, **Columns** və **Rows** xassələri əslində **Cell**, **Column** və **Row** obyektləri toplusunu qaytarmır – yalnız **Range** obyektləri toplusunu qaytarır. **Range** obyektinə buna görə vacibdir ki, Excel-də hücrələrlə olan bütün işlər və əməllər bu obyektin üzərində qurulub (bu fəsilə **Range** obyektinə xüsusi bölmə həsr edilib).

- **Cursor** – maus göstəricisinin Excel-də xarici görkəmini dəyişməyə imkan verir (həmin xassə Word-dəki **Application** obyektində yoxdur). Adətən uzun müddətli hesablamalarda kursora “qum saatının” görkəmi verilir (**xlWait**), sonra isə əvvəlki görkəm geri qaytarılır. Makros işini bitirdikdə isə, avtomatik olaraq, görkəm geri qaytarmır – bunun üçün uyğun olan VBA kodu nəzərdə tutulmalıdır.
- **DataEntryMode** – olduqca maraqlı xassədir: onun köməyi ilə istifadəçini çox sayda olan səhvlərdən və xətalardan qorumaq olur. İstifadəçiyə yalnız seçilmiş diapazonun bloklanması açılmış olan hücrələrinə qiymətlərin daxil edilməsi icazə verildikdə - bu xassə ilə **data entry** model rejimində verilənlərin qiymətlərini daxil edilməsinə imkan verir. Burada üç seçim vardır: **xlOn** – rejimin qoşulması, **xlStrict** – istifadəçinin **<Escape>** düyməsi ilə həmin rejimdən çıxmasının qarşısının alır, **xloff** – rejimin söndürülməsi.
- **DecimalSeparator** və **ThousandsSeparator** – xassələri istifadəçinin kompyuterində regional qurmalara baxmayaraq, aşkar olaraq, kəsr şəklində olan ədədi qiymətin kəsr hissəsini tam hissəsindən ayırmağa imkan vardır. Bu xassələr istifadə ediləndə, məsləhət görülür ki, **UseSystemSeparators** xassəsi ilə sistem təyinatları söndürülsün:

```
Application.UseSystemSeparators=False
```

- **Dialogs** - eyni adlı **Dialog** kolleksiyasını qaytarır: Excel-in dialog pəncərələrini əks etdirmək üçün istifadə etmək olur (əslində bir neçə yüz belə pəncərə var). Həmçinin onlara istifadəçilərin təsirindən yaranan reaksiyanı təyin etmək olur. **FileDialog** obyektinə bu obyektin alt yığımını təşkil edir (fayllar ilə işləmək üçün nəzərdə tutulmuş pəncərələri təmsil

edir, məsələn, faylın açılması pəncərəsini). Bu obyektə işləmə qaydası Word-dəki qaydadan fərqlənir.

- **DisplayAlerts** – bu xassə haqqında Word modulunda biz artıq tanış olmuşuq. Vacib olduğuna görə təkrar edirik: bu xassə müxtəlif xəbərdarlıqları keçirir, proqramla ilə işlədikdə istifadəçi onları görməməlidir (məsələn, heç bir dəyişiklik edilməmiş lazım olmayan faylın bağlanması xəbərdarlıq edən pəncərə).
- **EnableEvents** – bir müddət **Application** obyektini üçün hadisələri söndürməyə imkan verir. Adətən hansısa əməlin yerinə yetirilməsindən əvvəl istifadə edilir: məsələn, faylın açılması, faylın yaddaşda saxlanması v.s.
- **ErrorCheckingOptions** – eyni adlı obyektə istinad alır: Excel-in avto yoxlamasını sazlamaq olur (istifadəçiyə sintaktik səhvi olan düsturlar, boş olan hücrələrə istinadlar v.s. haqqında xəbərin verilib-verilməməsini təyin etmək üçün). Standart halda yoxlamaların bir çoxu qurulmuş halda olur. Buna görə obyektə o vaxt müraciət etmək lazımdır ki, yoxlamaların keçirilməsi üçün zərurət yaranmışdır.
- **FileDialog** – faylların açılma və yaddaşda saxlanma dialoq pəncərələrinə müraciət etməyə imkan yaradır (həmin əməli daha ümumi **Dialogs** xassə ilə etmək olar). Məsələn, istifadəçiyə açılma pəncərəsində hansısa bir faylın seçilməsinə (həmçinin ona doğru yolun alınmasına) imkan vermək üçün aşağıdakı VBA kodundan istifadə etmək olar:

```
Application.FileDialog(msoFileDialogOpen).AllowMulti_
Select=False
Application.FileDialog(msoFileDialogOpen).Show
Debug.Print Application.FileDialog(msoFileDialogOpen)._
SelectedItems(1)
```

Buna oxşar misalı bu xassə haqqında olan sorğu materiallarından tapmaq olar (eyni vaxtda bir neçə faylın seçilməsini təmin edir).

- **FileSearch** – bu xassə göstərilən kataloqda axtarışı icra edərək, nəticəni qaytarır.
- **Interactive** – istifadəçinin Excel kitabına daxil etməsinə qadağa qoyur (mausla və klaviatura ilə birgə). Adətən istifadəçinin proqramın işinə müdaxilə etməsinin qarşısını almaq üçün tətbiq edilir: məsələn, əgər Excel ilə əlaqəsi olan proqramdan istifadəçi müdaxilə etmək istəyirsə.
- **International** və **LanguageSettings** – xassələri eynilə Word-də olan kimi işləyirlər.
- **LibraryPath** - Excel-XLA qurumları faylları olan kataloqa yolu qaytarır. Standart halda qiyməti: `\Office11\Library`.
- **MoveAfterReturn** – xassəsi verilənlərin daxil edilməsi qurtardıqda və **<Enter>** düyməsi basıldıqda, növbəti hücrəyə keçidin qoşulmasını/söndürülməsini icra edir. **MoveAfterReturnDirection** xassəsi isə keçidin istiqamətini təyin etməyə imkan yaradır. Bir çox hallarda bu xassələr istifadəçinin verilənləri daxil etmə əməllərini xeyli

sadələşdirə bilər, məsələn, keçidin sağ tərəfdəki növbəti hücrəyə olmasını aşağıdakı VBA kodu ilə həyata keçirmək olar:

```
Application.MoveAfterReturnDirection=xlToRight
```

- **Names** – aktiv iş kitabında bütün adlandırılmış diapazonları təmsil edən, eyni adlı **Names** kolleksiyasını qaytarır. **Names** kolleksiyasının **Add()** metodu ilə istifadəçi öz adlandırılmış diapazonlarını da təyin edə bilər. Praktikada adlandırılmış diapazonlar Word-ün əlavə qurmaları kimi işləyir: onların köməyi ilə mürəkkəb Excel cədvəllərində verilənlər yığımının təyin edilməsini yerinə yetirmək olur. Alternativ yol ilə Excel-in qrafik interfeysinin **Insert**→**Name** menyusu ilə də diapazonları adlandırmaq olur.
- **ODBCErrors** və **OleDbErrors** – xassələri, uyğun olaraq, **ODBC** və **OleDb** verilənlər bazasına qoşulduqda, yaranan səhvlər haqqında məlumat almağa imkan yaradır. Real olaraq, onlar uyğun olan **ODBCError** və **OleDbError** obyektlərindən (məhz onlar səhvlər haqqında məlumatı saxlayır) ibarət olan eyni adlı kolleksiyaları qaytarır.
- **OnWindow** – xassəsi daha çox hadisəyə bənzəyir. Onun qiyməti kimi, kitabın səviyyə modulunda yerləşən, proseduranın adı göstərilir (standart halda belə modul yaradılmır – onu “əl ilə” yaradırlar). İstifadəçi hər dəfə Excel pəncərəsinə keçdikdə, həmin prosedura çağırılacaq (vacib deyil hansı kitabın hansı səhifəsidir). Bu xassə əvəzinə xüsusi adlı makroslardan istifadə etmək olar: **Auto_Activate** və **Auto_Deactivate** (**OnWindow**-da təyin edilmiş proseduranın icrasından sonra işə salınacaq).
- **Range** – olduqca vacib xassədir: hücrələr diapazonunu təmsil edən və Excel-də hücrələrlə həyata keçirilən hər bir əməldə istifadə edilən **Range** obyektini qaytarır.
- **ReferenceStyle** – hücrələrin əks edilmə rejimini **A1** (hərflər - sütunlar, ədədlər – sətirlər) və **R1C1** (sətirlər və sütunlar rəqəmlə göstərilir) arasında seçilməsinin təyin edilməyə xidmət edir. Excel-in qrafik interfeysində **Tools**→**Options** menyusunda **General** qurmasındakı bayraqcıqı **R1C1** qiymətinə keçirməklə seçim etmək olar. Adi istifadəçilərə **A1** rejimi daha rahat olduğu kimi, peşəkar proqramçılara da **R1C1** rejimi daha rahatdır (xüsusilə sütunların sayı çox olduqda: məsələn, AA, AB v.s.). Buna görə proqramçılar işlərini bitirdikdən sonra, yenidən **R1C1** rejimindən **A1** rejiminə keçirlər.
- **Selection** – həmin an istifadəçi seçdiyini qaytarır. Əgər o, adi cədvəldəki hücrələri seçibsə, onda adi **Range** obyektini qaytarılacaq. Əgər diaqramda nə isə seçilmişdirsə, onda diaqrama uyğun olan obyektlər qaytarıla bilər. Word-dən fərqli olaraq, Excel proqramlaşdırmasında **Selection** obyektini çox az istifadə edilir.
- **Sheets** – bu xassə **Sheets** kolleksiyasını qaytarır: kitabın səhifələr toplusunu (əgər proqram üçün çağırılırsa, onda aktiv kitab üçün qaytarılır) və ayrı səhifələrdə olan diaqramlar toplusunu. Əgər **Worksheets** xassəsi istifadə edilirsə, onda **Sheets** kolleksiyası qaydacaq (yalnız **Worksheet** obyektlərindən ibarət – adi, diaqramsız,

səhifələr). Bu xassə haqqında daha ətraflı məlumatı Excel kitablarına aid olan bölmədə verəcəyik.

- **TemplatesPath** – Excel şablonları olan kataloqu haqqında məlumatı qaytarır (yalnız oxumaq üçün nəzərdə tutulub). Məsələn, həmin kataloqda şəxsi şablonun yerləşdirilməsi və ya oradan hansısa şablonun açılması üçün xidmət edir. Standart halda istifadəçi profilində olan **Application Data\Microsoft\Templates** kataloqu istifadə edilir.
- **ThisCell** və **ThisWorkbook** – işləmək üçün olduqca rahat xassələrdir: obyekt dəyişənlərini yaratmadan, cari hücrəyə və kitaba müraciət etməyə imkan verir. Bu obyektləri yaratmaq lazım gəlmir – işləyən Exceld-də onlar artıq mövcud olur.
- **Windows, Workbooks** və **Sheets** – xassələri, uyğun olaraq, Excel-in bütün açıq olan pəncərələrini, kitablarını və səhifələrini qaytarır. İş kitabının və səhifənin obyektləri haqqında aşağıda əlavə məlumat verəcəyik.
- **WorksheetFunction** – VBA proqramında Excel funksiyalarını bir başa, əvvəlcədən onları hansısa hücrəyə yazmadan, istifadə etməyə imkan yaradır.

Excel.Application obyektinin ən vacib metodlarının izahlı siyahısı aşağıda verilib:

- **ActivateMicrosoftApp()** – Office proqramlarının (Word, Access, PowerPoint və digərlərinin: Project, FoxPro, Schedule Plus) işə salınması və aktivləşdirilməsi (yaxud yalnız aktivləşdirilməsi, əgər proqram artıq işə salınıbsa) üçün nəzərdə tutulmuş xüsusi metoddur.
- **AddCustomList()** – yeni istifadəçi siyahısının yaradılması üçün xidmət göstərir. Parametr kimi **Range** obyektini (bu halda diapazonda olan qiymətlər siyahının elementləri olacaq) və ya standart **Array** obyektini qəbul edir. Bu xassəni **DeleteCustomList()** ilə silmək olar.
- **Calculate...** - adı ilə başlayan metodlar hücrələrdə olan qiymətlərin sayılmasına imkan yaradır. Sadəcə **Calculate()** metodunun üzü (heç bir prefikssiz və sufiksiz) – qiymətlərin adi sayılmasını icra edir (daha çox avto sayılma keçirildikdə lazım olur. **CalculateFull()** – bütün açıq olan kitablardakı qiymətlərin sayılmasını icra edir. **CalculateFullRebuild()** - üstəlik düsturların yenidən qurulmasına imkan yaradır (bütün düsturların yenidən daxil edilməsinə bənzəyir). Sayılma əməliyyatını **CheckAbort()** metodu ilə dayandırmaq olar.
- **ConvertFormula()** – düsturun iki üsulla dəyişdirilməsinə imkan verir: 1) hücrələrin yeni ünvanlaşdırılmasını başqa rejimə keçirmək (məsələn, A1 rejimindən R1C1 rejiminə keçmək), 2) mütləq istinadları nisbiyə keçirmək (yaxud bunun əksini etmək). Parametr kimi düstur mətni ilə birgə (həmçinin çevirmə bayraqcıqları ilə də) sətir dəyişənini qəbul edir: “=” simvolu ilə başlanmalıdır.

- **DoubleClick()** – aktiv hücrə üzrə mausun iki dəfəlik şıqqılıtısına bənzəyir, yəni həmin hücrəyə verilənlərin daxil etmə rejiminə.
- **Evaluate()** – olduqca xeyirli və tez-tez istifadə edilən metoddur: Excel kitabı obyektinin adına görə tapılmasını və sonradan istifadə etmək üçün onu hansısa obyektə və ya qiymətə çevirməyə imkan yaradır. Bu metod aşağıdakı obyektlərin adını qəbul edir:
 - A1 stilində olan hücrələrin adını (**Cell** obyektini qaydır);
 - diapazonların adını (**Range** obyektini qaydır);
 - makrosda təyin edilən adları (daha çox – dəyişənlərin adını);
 - xaricdə olan kitabların adını (məsələn, **Evaluate**(" [Book1.xls] Sheet1!A5"))).

Bu funksiyaları qeyri aşkar da çağırmaq olar – sadəcə obyektin adını kvadrat mötərizələr içində yazaraq. Məsələn, aşağıdakı iki VBA kod sətiri eyni güclüdür:

```
[A1].Value=25
```

və

```
Evaluate("A1").Value=25
```

Kvadrat mötərizələrin sintaksisi daha kiçikdir deyər, adətən bu üsul daha çox istifadə edilir.

Beləliklə, **Evaluate()** metodu – başqa və ya öz Excel kitabındakı hücrəyə və ya diapazona müraciət etmək üçün ən sadə və təbii metoddur.

- **GetOpenFilename()** – faylların açılması pəncərəsi ilə işləmək üçün ən sadələşmiş üsuldur: **Open** dialog pəncərəsinin açılması və istifadəçinin nəyi seçdiyi haqqında (sətir dəyişəni şəklində faylın adı və tam yolu ilə) məlumat alınmasında ən sadə üsuldür. Aşağıdakı VBA proqram kodu həmin əməllərin icra edilməsində istifadə edilə bilər:

```
Filename=Application.GetOpenFilename()  
If Filename < > False Then  
    Debug.Print Filename  
End If
```

- **GetSaveAsFilename()** - **Save As** pəncərəsi ilə birgə işləyən metodla eyni güclüdür.
- **GoTo()** – çox vacib olan və tez-tez istifadə edilən metoddur: parametr kimi **Range** obyektini qəbul edir, hücrəyə istinadı olan sətiri (R1C1 formatında) və ya VBA prosedurasının adını (bu halda diapazon/hücrə ilə təchiz edir və proseduranı işə salır). **Select()** metodundan fərqli olaraq, üstəlik diapazon və ya hücrə yerləşdiyi səhifəni aktivləşdirir.
- **Help()** – bu metod sorğu faylında (həmçinin istifadəçi faylında) olan məlumatı açıb göstərməyə imkan verir. Parametr kimi sorğu faylının adını və mövzunun işarələnməsini qəbul edir.

- **Intersect()** – iki (və daha çox) diapazon kəsişməsindən alınan diapazonu qaytarır. Parametr kimi ötürülən diapazonlar kəsişmədiyi halda **Nothing** qiymətini qaytarır.
- **OnKey()** – VBA prosedurasını müəyyən klaviatura kombinasiyası rejimində işləməsinə imkan verir. Məsələn, **Msg()** prosedurası **Sheet1** səhifəsindəki moduldan **<Alt>+<M>** klaviatura kombinasiyasına keçirtmək üçün aşağıdakı VBA proqram kodu tətbiq edilməlidir:


```
Application.OnKey "%{m}", "Sheet1.Msg"
```
- **OnRepeat()** - **Edit** menyusunda **Undo** əmrini proqram səviyyəsində icra edir: həmin əmrin seçilməsində lazım olan proseduranın icra edilməsini təyin etməyə imkan verir.
- **RegisterXLL()** – Excel-in əlavə qurmalar faylının qoşulması imkanını XLL genişlənməsi ilə yaradır və onun prosedura və funksiyalarını registrə edir. Bu metod özül proqramın yüklənməsində istifadə edilir.
- **Repeat()** – xassəsi son dəfə işləyən istifadəçinin proqram üsulu ilə icra edilməyən əməlinin təkrar edilməsinə imkan yaradır. Excel qrafik interfeysinin **Edit** menyusundakı **Redo** əmrinə bənzəyir.
- **Run()** – çox universal metoddur: imkan verir ki, VBA prosedurası (və ya makrosu), Excel makrosu və ya XLL modulunda olan funksiya (ona 30-a yaxın parametri ötürmək şərtilə) icra edilsin.
- **SendKeys()** – metodu klavişlərin basılmasını emulyasiya edir və onları aktiv pəncərəyə göndərir. Adətən istifadəçiyə nəyi isə nümayiş etmək lazım olduqda (bu halda pauzalar tətbiq edilir) və ya menyu vasitəsi ilə nə isə etmək lazım olduqda (məsələn, tapılan şey tapılmadı və onu VBA kodu ilə icra etmək lazımdır) istifadə edilir.
- **Union()** – çox vacib metoddur: proqram üsulu ilə bir neçə diapazonu birləşdirir.
- **Volatile()** – yalnız hücrə qiymətini hesablayan istifadəçi funksiyasından çağırıla bilər. Əgər o, çağırıldıqda və ona **True** qiyməti ötürülərsə, onda həmin hücrə həssas olur ("*volatile*") və o, səhifədəki hər bir dəyişiklikdə (hətta onunla bağlı olmayanlarda belə) yenidən sayılacaq.
- **Wait()** – prosessordan yüklənməni götürərək, Excel-in işini təyin edilmiş vaxtda dayandırır. Nümayişlər edildikdə istifadə edilir (istifadəçinin hansısa əməlin qurtardığı zaman alınan nəticəni görmək üçün). Bu halda istifadəçi üçün məlumatın daxil edilməsi bloklanır, mausun göstəricisi isə "qum saati" şəklini alır. Excel-in bəzi fon əməliyyatları isə (çap etmə, qiymətlərin yenidən sayılması) işini davam edir: məsələn, 5 saniyəlik fasiləni təyin etmək üçün, aşağıdakı VBA kodundan istifadə etmək olar:


```
If Application.Wait(Now+TimeValue("0:00:5")) Then
  MsgBox "Beş saniyyə artıq keçdi"
End If
```
- **Quit()**, **OnTime()**, **FindFile()** – metodları eynilə Word-dəkilərə bənzəyir.

12.4 Workbooks kolleksiyası və Workbook obyektı, onların xassələri və metodları

Excel.Workbook obyektı, VBA-dan Excel kitabları ilə proqram səviyyəsində işləmək qaydaları

Excel modelində **Application** obyektindən sonra, iyerarxiya qaydası ilə, Excel kitabının özünü təmsil edən obyekt gəlir: **Workbook** obyektı. Müqayisədə, funksionallıq nöqtəyi nəzərdən, Word-də **Document** obyektinin tutduğu yeri Excel-də **Workbook** tutur. Onun əsas məqsədi - açıq olan Excel kitabları arasında lazımı kitaba istinad almaqdan ibarətdir və həmçinin kitabın bütün səhifələri ilə ümumi olan əməllərin aparılması üçün əlavə qurmaların tənzimlənməsini təmin edir. **Workbook** obyektinin yaradılması (alınması desək daha dəqiq olar) çox sadə yollarla həyata keçirilə bilər:

- **birinci üsul** - **Workbooks** kolleksiyasından istifadə etmək (**Application** obyektinin **Workbooks** xassəsindən əlçatandır). Bununla belə bu üsulu tətbiq etmək məcburi deyil - **Workbooks** kolleksiyası Excel-də onsuz da həmişə əlçatandır. Lazımı kitabı adına və ya kolleksiyadakı nömrəsinə görə kolleksiyadan asanlıqla tapıla bilər, məsələn:

```
Debug.Print Workbooks("Say_hesab.xls").FullName
```

- **ikinci üsul** - **Application.ActiveWorkbook** xassəsindən istifadə etmək: hal-hazırda aktiv olan kitaba müraciət etməklə:

```
Debug.Print ActiveWorkbook.Name
```

- **üçüncü üsul** - **Application.ThisWorkbook** xassəsindən istifadə etmək: baxılan proqram modulun mənsub olduğu kitaba müraciət etməklə:

```
Debug.Print ThisWorkbook.Name
```

Praktiki işlərdə əsasən iki əməl daha çox lazım olur: Excel-də yeni kitabın yaradılması və ya mövcud olan kitabın açılması (yaxud da Excel qəbul etdiyi digər formatda olan faylın açılması, məsələn, DBF formatında olan faylı). Bu məqsədlə, uyğun olaraq, **Add()** və **Open()** metodlarından istifadə edirlər. Məsələn, yeni Excel kitabı VBA kodu ilə bu cür yaradıla bilər:

```
Dim oWbk As Workbook  
Set oWbk=Workbooks.Add()
```

Yeganə qəbul etdiyi məcburi olmayan parametr – yeni iş kitabının yaradıldığı əsas rolunu oynayan şablondur. Mövcud olan Excel kitabının yaradılması proqram səviyyəsində bu cür həyata keçirilə bilər:

```
Dim oWbk As Workbook  
Set oWbk=WorkBooks.Open("C:\mybook1.xls")
```

Standart olan metodlardan əlavə **Workbooks** kolleksiyasında üç sayda xüsusi metod nəzərdə tutulub:

- **OpenDatabase ()** – verilənlər bazasının açılması və ona olan sorğunun icra edilməsi (və ya cədvəl/təqdimatın bir başa açılması), sonra isə sorğunun nəticəsinin xaricdən import edilən verilənlər kimi, yeni avtomatik rejimdə yaradılan Excel kitabına yerləşdirmək əməllərini yerinə yetirilməsi üçün;
- **OpenText ()** – funksionallığı yuxarıdakı metoda bənzəyir, amma mənbə kimi burada mətn faylı istifadə edilir. Əlavə parametrlər onun formatını təyin etməkdə kömək edir;
- **OpenXML ()** – bu metod da funksionallığı ilə fərqlənir, yeganə fərq budur ki, mənbə rolunu burada XML formatında olan fayl oynayır.

Word-dəki **InsertDatabase ()** metodu kimi, bu metodlar da ən sadə hallarda istifadə edilə bilər: Microsoft məsləhət görür ki, imkan olduqda, daha güclü standart ADO obyekt modelinin vasitələrindən istifadə edilsin.

İndi isə **Workbook** obyektinin ən vacib olan xassələri haqqında:

- **Name** - ən sadə addır: bu ad kitab faylının adı ilə üst-üstə düşür. Kitabın kod adı **Name** sətirində göstərilir;
- **CodeName** – kodda kitabın adını bildirir: onu **Project Explorer** pəncərəsində və ya kitab xassələri aktivləşdirilibsə, **Properties** pəncərəsində;
- **FullName** – əməliyyat sistemində Excel kitabına gedən yolun tam ünvanı ilə onun tam adını qaytarır. Hər üç kitab yalnız oxumaq üçün əlçatandır – onları başqa üsulla dəyişmək mümkündür (məsələn, faylı başqa adla yaddaşda saxlamaqla və ya bir başa **Properties** pəncərəsində). Burada qeyd etməliyik ki, **Path** xassəsi də müəyyənliklə adlarla bağlılığı vardır (həmçinin Excel faylına doğru gedən yol bu xassədən əlçatandır).
- **Charts, Sheets, ActiveChart, ActiveSheet, CustomViews, BuiltinDocumentProperties** və **CustomDocumentProperties, Windows, WebOptions** - xassələri, uyğun olan, eyni adlı obyektlərin kolleksiyasını qaytarır (onlardan bəzilərinə aşağıda baxacağıq).
- **ConflictResolution** – xassəsi bir neçə istifadəçi eyni vaxtda açıqda ("*shared workbook*"), verilənlərin dəyişdirilməsi konfliktini həll edir. İmkanlar var ki, lokal istifadəçi, avtomatik olaraq, əlverişli/əlverişsiz vəziyyətdə olsun və ya konflikti dialoq pəncərəsində "əl üsulu" ilə həlli tapılsın. Excel kitabı ilə birgə işləmək parametrlərini sazlamaq üçün çox sayda xassə vardır. Nəzərə alınsa ki, bu cür iş üslubu (birgə əlçatan olan verilənlər gərək verilənlər bazasına köçürülsün) qətiyyətlə qəbul edilməz sayılır, ona görə burada həmin xassələrə baxılmayacaq, yalnız bir neçə sayda, istisna edilən, hallardan başqa:
 - **SaveAs ()** və ya **ExclusiveAccess ()** - metodları qadağa/icazə statusunu ümumi əlçatanlıq üçün iş kitabına qoyulması üçün istifadə edilir;

- **MultiUserEditing** – standart halda kitabın birgə redaktə edilməsi rejiminin söndürülməsi/qoşulmasını yoxlamaq üçün istifadə edilir;
- **UserStatus** – bütün istifadəçilərin siyahısının alınmasını təmin edir (həmçinin onlar faylı nə vaxt və hansı rejimdə açdıqları məlumatla birgə).
- **FileFormat** – kitabın formatlı (birbaşa yalnız oxumaq üçün əlçatandır, yalnız yaddaşda saxlandıqda dəyişdirmək olar). Formatların sayı olduqca çoxdur: Excel versiyaları, DBF, Lotus 1-2-3, TXT, CSV, XML formatları – ümumilikdə iyirmiye yaxındır.
- **Names** - xassəsi cari iş kitabının adlandırılmış diapazonlarının kolleksiyasını qaytarır. Cari kitabın adlandırılmış diapazonları haqqında məlumat almaq üçün aşağıdakı VBA kodundan istifadə etmək olar:

```
For Each Item In ThisWorkbook.Names
    Debug.Print Item.Name
Next
```

Bu xassənin tətbiqi xüsusilə icra etmə zamanının yaratdığı potensial səhvlərinin qabaqcadan yoxlanmasında proqramçılar üçün olduqca rahat şərait yaradır.

Workbook obyektinin metodlarının sayı həmçinin böyükdür, ancaq ən çox istifadə edilənləri bunlardır: **Activate()**, **Close()**, **Save()**, **SaveAs()**, **PrintOut()**, **Protect()** və **Unprotect()** – adlarından da aşkardır və Word-ün **Document** obyektinin eyni adlı metodları ilə, funksionallıq baxımından, üst-üstə düşürlər.

12.5 Sheets kolleksiyası və Worksheet obyektı, onların xassələri və metodları

Excel.Worksheet obyektı, VBA vasitələri ilə Excel səhifəsinin proqram səviyyəsində yaradılması, tapılıb təyin edilməsi və yox edilməsi (silinməsi), Excel.Worksheet obyektinin xassələri və metodları

Word-də **Application** və **Document** obyektindən sonra mətnlə işləmək üçün nəzərdə tutulan obyektlər iyerarxiya pilləsində dururlar: **Selection**, **Range** v.s. Excel-də isə iş kitabının obyektı və hücrələr arasında bir keçici obyekt də var - **Worksheet** obyektı (faktiki olaraq, Excel səhifəsi). **Worksheet** obyektləri Excel kitabında **Sheets** kolleksiyasında birlik təşkil edir.

Çox hallarda Excel-ə verilənləri daxil etmək üçün (bir başa və ya verilənlər bazasından) birinci növbədə verilənlər daxil edildiyi səhifə ilə müəyyən qərara gəlmək lazım olur: sadəcə onu seçmək və ya əvvəlcədən onu yaradaraq sonradan seçmək.

Excel səhifəsinin VBA vasitəsi ilə yaradılması olduqca sadə proses kimi görünür:

```
Dim oExcel As New Excel.Application 'Excel-i işə salırıq
oExcel.Visible=True 'Onu görünən edirik
Dim oWbk As Excel.Workbook
Set oWbk=oExcel.Workbooks.Add() 'Yeni kitabı yaradırıq
Dim oSheet As Excel.Worksheet
Set oSheet=oWbk.Worksheets.Add() 'Yeni səhifə yaradırıq
```

```
oSheet.Name="Yeni səhifə"      'Ona yeni ad veririk -
                                'Yeni səhifə
```

Worksheets kolleksiyasının **Add()** metodu bir neçə sayda məcburi olmayan parametr qəbul edir: əsas məqsədi – hansı mövcud olan səhifələr arasında yeni səhifə yerləşdirmək. Əgər heç nə göstərilməsə, onda yeni səhifə birinci yerdə yaradılacaq.

Tez-tez başqa növ məsələyə də rast gəlmək olur – cari kitab açıldıqda, kitabın səhifələrinin arasında lazım olanının tapılması. Bunu etmək çətin deyil – çünki **Worksheets** kolleksiyası səhifələrin adı ilə işləməyi bacırır. Aşağıdakı nümunədə, Excel kitabı işə salınıraq, yeni kitab yaradılır və, bununla belə, “Sheet1” adlı səhifə tapılır onun adı “Yeni səhifə” adı ilə dəyişdirilir:

```
Dim oExcel As New Excel.Application      'Excel-i işə salırıq
    oExcel.Visible = True                'Onu görünən edirik
Dim oWbk As Excel.Workbook
    Set oWbk=oExcel.Workbooks.Add()     'Yeni kitabı yaradırıq
Dim oSheet As Excel.Worksheet
    Set oSheet=oWbk.Worksheets.Item("Sheet1") 'Sheet1 tapılır
    Item("Sheet1")
    oSheet.Name="Yeni səhifə"           'Ona yeni ad təyin
                                        'edilir -Yeni səhifə
```

Diqqət verilməlidir ki, yuxarıdakı VBA proqram kodu Excel-in yalnız İngilis dilli versiyasında işləyəcək: çünki səhifə sözünə ingiliscə “Sheet” deyirlər (biz də belə etdik). Əgər, məsələn, “Sheet” sözü əvəzinə Azərbaycanca Səhifə və ya digər söz (Azərbaycanca) yazılsa, onda həmin proqram çalışmayacaq. Buna görə məsləhət edirik ki, bu üsuldən istifadə edəsiniz: əgər səhifələrin adları standart halda olan kimi istifadə edilirsə (müxtəlif dilli Excel proqramlarında proqram işləyəcəksə), onda mütləq əlavə yoxlamalar tərtib edilməlidir və ya sadəcə səhifələrin adı əvəzinə onların nömrələrini istifadə etmək lazımdır.

Sheets kolleksiyasında ənənəvi xassələr və metodlardan (məsələn, **Count**, **Item**, **Add()**, **Delete()**) və **Worksheet** obyektini ilə əlverişli işləyən digər (**Visible()**, **Copy()**, **Move()**, **PrintOut()**, **PrintPreview()**, **Select()**) metod və xassələrdən əlavə daha bir vacib metodu vardır - **FillAcrossSheets()**. Bu metod **Range** diapazon obyektini kopyalayaraq (variantlar: tam, məzmununu və ya yalnız formasını) cari kitabın bütün səhifələrinə yerləşdirir.

Worksheet obyektinin ən vacib olan xassələri bunlardır:

- **Cells** - ən tez-tez istifadə edilən xassələrdəndir: eynilə, yuxarıda baxılan, **Application** obyektinin eyni adlı xassəsi kimi işləyir – bir fərq ilə: yalnız bir aktiv səhifə ilə məhdudlaşmışır. **Columns** və **Rows** xassələri də analogi qaydada işləyir.
- **EnableCalculation** – Excel kitabının hücrələrində avtomatik sayılmaları söndürə bilir.
- **EnableSelection** – vacib xassədir: səhifədə hər nə varsa onların seçilməsinin qarşısının alınmasını, heç bir şeyə qadağa qoyulmamasını və ya yalnız bloklaşdırılmayan hücrələrin seçilməsinə icazə verilməsini icra edir.

- **Next** – kitabın növbəti səhifəsinə keçməyə imkan yaradır (eyni ilə həmin əməli **Previous** xassəsi əvvəlki səhifə üçün edir).
- **PageSetup** – Word-də olan kimi, səhifənin çap parametrlərini sazlayan **PageSetup** obyektini yaradır (Excel-in qrafik interfeysinin **File**→**PageSetup** menyusundan edildiyi əməlləri üçün).
- **Protection** – xassəsi ilə eyni adlı **Protection** obyektini almaq olur: Excel səhifəsinə istifadəçinin dəyişiklik etməsinin qarşısını almaq üçün istifadə edilir. **Protection** sözü ilə başlayan digər xassələr başqa mühafizə parametrlərinin sazlanmasında kömək edir.
- **QueryTables** – olduqca vacib xassədir: xaricdə olan mənbələrdə olan verilənləri (adətən verilənlər bazasından) təmsil edən **QueryTables** kolleksiyasını qaytarır.
- **Range** – bu xassəsi **Worksheet** obyektinin həqiqətən də ən vacib xassəsidir: **Range** obyektini qaytarır (Excel obyekt modelində tutduğu yer eynilə Word-dəki eyni adlı obyektin yeri ilə bərabər səviyyədədir). Bu obyektin vacib olduğuna görə növbəti bölmə ona həsr olunub.
- **Type** – cari səhifənin tipini müəyyən etməkdə kömək edir (adətən iki tip istifadə edilir: **xlWorksheet** (adi səhifə) və **xlChart** (diaqram səhifəsi)).
- **UsedRange** – bu xassə də **Range** obyektini qaytarır: obyekt, hücrələri boş olmayan düzbucaqlı diapazon sahəsini təmsil edir. Kopyalama və formatlama əməllərində istifadə edilir.
- **Visible** – istifadəçinin nəzərindən səhifənin gizlədilməsinə imkan yaradır (məsələn, əgər həmin səhifə xidməti məqsədlə hansısa işdə istifadə edilirsə).

Worksheet obyektinin vacib olan metodları:

- **Activate ()**, **Calculate ()**, **Copy ()**, **Paste ()**, **Delete ()**, **Move ()**, **Evaluate ()**, **Select ()**, **SaveAs ()**, **PrintOut ()**, **PrintPreview ()**, **Protect ()**, **Unprotect ()** – xassələri artıq oxucuya tanışdır: fərq budur ki, onlar indi **Worksheet** obyektində qulluq edirlər (onları seçilmiş səhifəyə tətbiq etmək olacaq);
- **PivotTables ()** – xassəsi, proqramlaşdırma baxımından, çox maraqlı olan obyektlərdən ibarət olan **PivotTable ()** kolleksiyasını qaytarır (aşağıda vacib olduğuna görə daha ətraflı baxılacaq);
- **Scenarios ()** – eyni adlı **Scenario** obyektlərindən ibarət olan **Scenarios** (ssenarilər) kolleksiyasını qaytarır: bu obyektləri müxtəlif ssenarilərin yoxlanmasında istifadə etmək olar (məsələn, satışın müxtəlif məbləği, vergilər səviyyələri, səflər v.s.);

- **SetBackgroundPicture()** – səhifəyə fon görüntüsünün verilməsini təmin edir (təbii olaraq, bu fonun yarım şəffaf olması daha arzu edilən haldır - digər halda səhifənin fonunda hücrələrdəki məlumatı oxumaq olmayacaq);
- **ShowAllData()** – vacib xassədir: səhifədəki bütün filtrə edilmiş və gizli olan verilənlərin göstərilməsinə imkan verir;

Worksheet obyektinin ən vacib olan hadisəsi - **Change** hadisəsidir. Praktiki məsələlərdə istifadəçi tərəfindən hansısa hücrədəki qiymətin dəyişdirilməsi nəticəsində başqa Excel kitabının və ya həmin Excel kitabının, təyin edilmiş hansısa səhifəsindəki müəyyən hücrələrdə (və ya hətta verilənlər bazasında, bəzi hallarda İnternet resurslarında olan verilənlər bazasında), başqa hücrələrdən asılı olan, dəyişikliklər olması halları tez-tez rast gəlir. Başqa praktiki məsələlərdə həmin hadisə istifadəçi tərəfindən Excel kitabına daxil etdiyi qiymətlərin yoxlanılmasında istifadə edilə bilər. Bu hadisə tipli VBA prosedurası **Target** adlı xüsusi parametrlə birgə işləyir - əslində, dəyişmiş hücrəni təmsil edən, **Range** obyektinə birgə işləyir. **Range** obyektinə isə sətir və sütun dəyişdirilmiş qiymət haqqında məlumat almaq olar.

Worksheet obyektinin daha iki sayda vacib olan hadisəsi vardır - **BeforeRightClick()** və **BeforeDoubleClick()**. Qeyd etməliyik ki, Word-ün **Document** obyektində bu güclü hadisələr yoxdur. Adlarından məlum olur ki, onlardan birincisi səhifədə, mausla istənilən yerdə sağ düymə ilə bir dəfə şıqqıltı etdikdə, şıqqıltının tutulmasını icra edir. İkincisi isə həmin düymənin, analoji əməllərdə, iki qat şıqqıltısını tutur. Bu hadisələrin köməyi ilə, istifadəçiyə, onun psixologiyasına uyğunlaşdıraraq, məxsusi olan reksiyanı təyin etmək olur (məsələn, kontekst menyusunun açılmasında, xəbərdarlıq məlumatları verildikdə, başqa iş rejiminə keçdikdə v.s.).

12.6 Range obyektinə, onun xassələri və metodları

Excel.Range *obyektinə, VBA vasitələri ilə diapazonlar və hücrələrlə proqramlaşdırma işləri, Excel.Range obyektinin yaradılması, Excel.Range obyektinin xassələri, metodları və hadisələri*

Əminliklə demək olar ki, Excel obyekt modeli iyerarxiyasında ən çox istifadə edilən obyekt məhz **Range** obyektidir. Bu obyekt olduqca universaldır: Excel kitabında bir hücrəni, bir neçə hücrəni birlikdə (həmçinin bir-biri ilə kəsişməyən hücrələri və ya yanaşı olmayan hücrələr birliyini) və ya tam bir səhifəni təmsil edə bilər. Əgər Word-də verilənləri daxil etmək üçün **Range** və **Selection** obyektlərini, eyni funksionallıq baxımından, istifadə etmək olursa, Excel-də bunun üçün yalnız **Range** obyektinə istifadə edilir:

- əgər verilənlər hücrəyə daxil edilməlidirsə və ya hücrə özü formatlanmalıdırsa, onda həmin hücrəni təmsil edən **Range** obyektinə alınmalıdır;
- əgər seçilmiş hücrələrlə hansısa əməl aparılmalıdırsa, seçilməni təmsil edən, **Range** obyektinə alınmalıdır;

- əgər bir qrup hücrə ilə sadəcə nə isə etmək lazımdırsa, həmin qrup hücrəni təmsil edən **Range** obyektini alınmalıdır.

Microsoft Knowledge Base-də (Microsoft şirkətinin biliklər bazasında - www.microsoft.com/support) Excel-də **Range** obyektinin 22 üsul ilə alınması (yəni yaradılması) 291308 nömrə altında olan bir məqalədə verilib. Əlbəttə, çox güman ki, bütün bu üsullar real istifadəçiyə lazım olmayacaq. Lakin biz, burada, proqramlaşdırmada ən çox istifadə edilən üsullar ilə tanış olacağıq:

- **ən sadə və aşkar olan üsul** – **Range** obyektinin xassəsindən istifadə etmək. Bu xassə **Application**, **Worksheet** obyektlərinin və **Range** obyektinin özündə də nəzərdə tutulub (yeni diapazonu əvvəl mövcud olanın əsasında yaratmaq lazım olduqda istifadə edilir). Məsələn, A1 hücrəsini təmsil edən **Range** obyektinə istinad VBA proqram kodu ilə bu cür almaq olar:

```
Dim oRange As Range
Set oRange=Worksheets("Sheet1").Range("A1")
```

A1-dən D10 hücrəsinə qədər olan diapazona isə bu cür:

```
Dim oRange As Range
Set oRange=Worksheets("Sheet1").Range("A1:D10")
```

Range obyektinin öz tərkibində olan **Range** xassəsi ilə çox ehtiyatlı olmaq lazımdır: çünki Excel **Range** obyektini əsasında öz məxsusi nömrələnməsi olan virtual səhifə yaradır. Buna görə, məsələn, aşağıdakı VBA kodu:

```
Set oRange1=Worksheets("Sheet1").Range("C1")
Set oRange2=oRange1.Range("B1")
oRange2.Value=20
```

VBA kodundan anlayaraq, gözləndiyi kimi: bu VBA kodu 20 ədədini B1 hücrəsinə yox, D1 hücrəsinə yazacaq (yəni virtual səhifəyə nisbətə C1-dən başlayan B1 hücrəsinə).

- **ikinci üsul** – **Cells** xassəsindən istifadə etmək: bu xassənin imkanları daha azdır (biz bir hücrədən ibarət olan diapazonu qaytara bilərik). Bu xassənin daha əlverişli sintaksisi vardır (dəyişənlərin ötürülməsi, istənilən sayda hücrə və istənilən istiqamətdə yerdəyişmə baxımından v.s.). Məsələn, D1 hücrəsinə istinad alınması üçün aşağıdakı VBA kodu istifadə edilə bilər:

```
Dim oRange As Range
Set oRange=Worksheets("Sheet1").Cells(1,4)
```

Lakin bir neçə hücrədən ibarət diapazonu almaq üçün isə daha əlverişli yol kimi **Range** və **Cells** xassələrindən istifadə etmək lazımdır, məsələn, aşağıdakı nümunədəki kimi:

```
Dim oRange
Set oRange=Range(Cells(1,1), Cells(5,3))
```

- **üçüncü üsul** – cari diapazonun dəyişməsinə və ya yeni diapazonun yaradılmasını icra edən, **Range** obyektinin çox saylı xassələrindən istifadə etmək. Bu vacib xassələrə aşağıda baxacağıq.

Adətən lazımı hücrə tapıldıqdan sonra, ona nə isə yazılmalıdır. Bu əməli yerinə yetirmək üçün **Value** xassəsi istifadə edilir, məsələn:

```
oRange.Value="Mənim qiymətim"
```

Range obyektini, funksionallıq nöqtəyi nəzərdən, çox vacib olduğuna görə, onun xassələri və metodları olduqca çeşidli və çox funksionaldır (Excel-də rahat işləmək üçün onları mükəmməl bilmək lazımdır).

Aşağıda **Range** obyektinin ən vacib olan və proqramlaşdırmada ən çox istifadə edilən xassələri verilib:

- **Address** – cari diapazonun ünvanını qaytarır, məsələn, əvvəlki nümunə üçün **\$A\$1:\$C\$5** diapazonu qaytarılacaq. Əgər diapazonda yalnız bir hücrə varsa, onda **\$A\$1** görkəmində qiymət qaytarılacaq. Bu xassəyə çox sayda parametr ötürülə bilər – istinadın stilini təyin etmək üçün, sütun və sətir üçün mütləq və ya nisbi ünvan üçün, nəyə nisbətən həmin ünvan nisbi olacaq v.s. Nəzərə alınmalıdır ki, bu xassə yalnız oxumaq üçün əlçatandır. **AddressLocal** - əslində həmin xassədir, lakin Excel-in lokallaşmış versiyalarını nəzərə alır.

Praktikada belə hallar çox tez-tez rast gəlir – hücrənin ünvanı hissələrə bölünərək, ona sətirin və ya sütunun adını qaytarmaq lazımdır. Bu əməli daha sadə yolla sətir funksiyalarının tətbiqi ilə icra etmək olar. Məsələn, bir hücrəni təmsil edən, **oRange** obyektini üçün sütunun adını bu cür qaytarmaq olar:

```
sColumnName=Mid(oRange.Address, 2, (InStr(2, oRange.Address, "$")-2))
```

Sətirin nömrəsi üçün isə bu VBA kodu istifadə edilə bilər:

```
sRowNumber=Mid(oRange.Address, (InStr(2, oRange.Address, "$")+1))
```

Ola bilsin ki, birinci baxımdan, yuxarıda göstərilən nümunə VBA kodu çətin görünsün. Əslində belə deyil – sütunun adı üçün biz birinci dollar işarəsi (o isə həmişə birinci simvol kimi durur) və ikinci dollar işarəsi arasında olan “hər nə varsa” götürürük. Sətirin adı üçün isə ikinci dollar işarəsindən sonra gələn “hər nə varsa” onu götürürük. Həmin ikinci dollar işarəsini qurulmuş **InStr()** funksiyası ilə tapmaq olur, lazımı sayda simvolların təyin edilməsini isə (hansıdansa başlayaraq) daha sadə yolla qurulmuş olan **Mid()** funksiyasının köməyi ilə yerinə yetirmək olar.

- **AllowEdit** – bu xassə yalnız oxumaq üçün əlçatandır: istifadəçinin mühafizə edilmiş səhifədə hansısa hücrədə və ya seçilmiş hücrələr diapazonunda düzəlişlərin aparılmasını icra etməyini/etməməyini təyin edir: daha çox yoxlamalarda istifadə edilir.

- **Areas** – olduqca vacib xassələrdəndir: standart olmayan diapazonları, standart olan hissələrə bölərək, alınan **Range** obyektlərini **Areas** kolleksiyasına yerləşdirir. İş burasındadır ki, yuxarıda qeyd etdiyimiz kimi, **Range** obyektı adətən yanaşı durmayan hücrələr toplusundan ibarət olur. Həmin qaydada düzölmüş hücrələr toplusuna tətbiq edilən metodların nəticəsi bəzən gözənilməz effektlər verir (ən sadə hallarda səhvi qaytarır). **Areas** xassəsi isə həmin çətinlikləri aradan qaldırmaq üçün olduqca əlverişli vasitədir. Bu xassə ilə standart olamayan diapazonu yoxlayıb təyin etmək də mümkündür:

```
If Selection.Areas.Count > 1 Then
    Debug.Print "Bir-birindən ayrılmış sahələri olan diapazon"
End If
```

- **Borders** - xassəsi **Borders** kolleksiyasına istinad alır: daha çox diapazonun çərçivəsini idarə etməkdə istifadə edilir.
- **Cells** – bu xassə **Range** obyektində də vardır və həmin qaydada da fəaliyyət göstərir. Fərqi var – burada diapazon əsasında olan məxsusi virtual ünvanlaşma istifadə edilir, məsələn, aşağıdakı nümunədəki kimi:

```
Dim oRange, oRange2 As Range
Set oRange=Range(Cells(2,2), Cells(5,3))
Set oRange2=oRange.Cells(1,1) 'A1 əvəzinə B2-yə istinad alınır.
Debug.Print oRange2.Address 'Yoxlama
```

Eyni xüsusiyyətlər **Row** və **Rows**, **Column** və **Columns** xassələrində də vardır.

- **Characters** - bu xassə çətin olan bir məsələnin öhdəsindən gəlir: hücrədəki başqa verilənlərə dəyməyərək, mətn hissələrinin dəyişdirilməsini (mətni və ya formatı) icra edir. Məsələn, A1 hücrəsinə mətni daxil edərək, bununla belə birinci duran hərfin rəngini dəyişmək üçün, aşağıdakı nümunədən istifadə etmək olar:

```
Dim oRange As Range
Set oRange=Range("A1")
oRange.Value="Mənim mətnim"
oRange.Characters(1, 1).Font.Color=vbRed
```

Əgər sadəcə qiymət dəyişdirilməlidirsə, onda daha yaxşı üsul **Value** xassəsini tətbiq etməkdir (nümunənin üçüncü sətrindəki kimi).

- **Count** - diapazondakı hücrələrin sayını qaytarır: yoxlamada istifadə edilə bilər.
- **CurrentRegion** – çox əlverişli üsuldur: o, boş hücrələrdən ibarət olan diapazon kimi **Range** obyektini qaytarır (yəni əslində, ilkin olan diapazon/hücrə-nin daxil olduğu boş olmayan sahəni). Bu xassə, xaricdəki mənbədən alınan verilənlərin kopyalama/ixrac əməllərindən istifadə edilə bilər (lakin əvvəldən verilənlərin sayı məlum olmadıqda). Məsələn, aktiv olan hücrə ətrafında olan bütün boş olmayan sahənin seçilməsi üçün aşağıdakı nümunədə verilən VBA kodundan istifadə etmək olar:

```
Worksheets("Sheet1").Activate
ActiveCell.CurrentRegion.Select
```

- **Dependents** – ilkin olan diapazondan asılı **Range** obyektini qaytarır (çox güman ki, bir-birilə ayrı duran sahələrlə birləşə). Xəssə yalnız cari səhifə üçün fəaliyyət göstərir – xaricdə olan səhifələrə tətbiq edilmir. Məsələn, aşağıdakı nümunədə, aktiv hücrədən asılı olan, bütün hücrələri seçmək üçün bu VBA kodu tətbiq edilə bilər:

```
Worksheets("Sheet1").Activate
ActiveCell.Dependents.Select
```

Əks əlaqələrə baxmaq üçünsə, **Precedents** xassəsini tətbiq etmək lazımdır. Yalnız birinci səviyyədə olan əlaqələrə baxmaq üçün isə **DirectDependents** və **DirectPrecedents** xassələrindən istifadə etmək olar.

- **End** – daha bir çox istifadə edilən üsullardandır: bu xassənin köməyi ilə, ilkin diapazonun sonuncu hücrəsini təmsil edən, **Range** obyektini almaq mümkün olur (sonuncu hücrənin təyin edilməsi isə ötürülən parametrlə təyin edilir).
- **Errors** – eyni adlı **Errors** kolleksiyası ilə, diapazonda aşkar edilən səhvləri təmsil edən, **Error** obyektlərinə əlçatmaq üçün imkan yaradır.
- **Font** – xəssə, Word-də olan kimi, burada da hücrədəki mətnin formalaşmasını (rəngini, şriftini, hərflərin ölçüsünü v.s.) təmsil edən **Font** obyektini əlçatan edir.
- **FormatConditions** – özü (istifadəçi tərtib edən) obyektini yaratmağa imkan verir: başqa hücrələrə və diapazonlara tətbiq edilə bilən və hücrələrin formalaşmasını təmsil edən obyektlərin yaradılmasında istifadə edilir.
- **Formula** – bu xəssə **Range** obyektinin ən vacib olan xassələrindəndir: yalnız oxumaq üçün rejimdə olanda hücrədə yazılmış düsturun mətnini qaytarır. Redaktə etmə rejimində olduqda isə hücrəyə lazım olan düsturun sonradan avtomatik rejimdə icra üçün yazmağa imkan yaradır. Əgər bu xəssə bir neçə hücrədən ibarət diapazona tətbiq edilirsə, onda düstur diapazonun bütün hücrələrinə yazılacaq. Xəssənin tətbiqinə aid nümunə aşağıda gətirilib:

```
Worksheets("Sheet1").Range("A3").Formula="=$A$1+$A$2"
```

- **FormulaLocal**, və həmçinin **AddressLocal** – xassələri lokal versiyalı Excel-də hücrələrin nömrələnməsi üçün düzəliş verməyə imkan yaradır.
- **FormulaHidden** - diapazondakı düsturların istifadəçi nəzərindən qizlətmək üçün xidmət edir: yalnız mühafizə edilmiş səhifələrdə işləyə bilər.
- **HasFormula** – hansısa diapazonu hesablanan qiymətlərin olmasına (düsturların mövcudluğuna) yoxlayır.
- **Hidden** – diapazonun gizlədilməsini icra edir: yalnız bir şərtə işləyir - diapazonun tərkibində sütun və ya sətir olduqda, digər halda səhv qaytarılır.

- **Interior** – formatlaşma ilə bağlı olan daha bir xassədir: əsasən diapazonun rənglənməsində istifadə edilir.
- **Item** – ilkin olan diapazonun hərəkət etməsi əsasında daha bir **Range** obyektinin alınmasına imkan yaradır.
- **Locked** – səhifənin mühafizəsini qurduqda diapazondakı hücrələri blokladırır.
- **Name** – adlandırılmış diapazonu təmsil edən **Name** obyektinə istinad alınmasına xidmət edir (Excel-in qrafik interfeysində bu obyektin imkanları ilə **Insert**→**Name** menyusunda tanış olmaq olar). Funksionallıqda, Word-dəki qurma obyektinə bənzəyərək, diapazonlara və düsturlara adları ilə müraciət edir.
- **Next** – növbəti hücrəyə keçməyə kömək edir: əgər səhifə mühafizə olunubsa, onda keçid sağ tərəfdəki hücrəyə olur, digər halda növbəti hansısa bloklanmış hücrəyə.
- **NumberFormat** - ədədlər üçün əvvəldən qurulmuş hansısa formatı təyin edir. Excel-in qrafik interfeysindəki **Format**→**Cells**→**Number** menyusundakı imkanlara uyğun gəlir.
- **Offset** – yeni **Range** obyektini, müəyyən yer dəyişməsi ilə, ilkin olan **Range** obyektinin əsasında yaradır. Məsələn, aşağıdakı VBA proqram kodu ilə ilkin vəziyyətdə olan hücrədən üç hücrə yuxarı və üç hücrə sol tərəfə yer dəyişməsi ilə yeni yerə köçürülməni icra etmək olar:


```
Worksheets("Sheet1").Activate
ActiveCell.Offset(rowOffset:=-3, columnOffset:=-3).Activate
```
- **Orientation** – mətnin hücrədə oriyentasiyasını seçməyə imkan verir. Maillik bucağını qradusla göstərir. Məsələn, mətni diaqonalla yerləşdirmək üçün bu koddan istifadə etmək olar:


```
oRange.Orientation=-45
```
- **PageBreak** – bu xassəni, adətən, səhifə parçalanmasını proqram səviyyəsində verdikdə istifadə edirlər. Tətbiqi bu cür ola bilər:


```
Worksheets("Sheet1").Rows(50).PageBreak=xlPageBreakManual
```
- **Pivot...** - bu prefikslə başlayan **PivotTable** (yekun cədvəl) obyektini ilə işləyən xassələrə aiddir. Obyektlə proqram səviyyəsində işləmə haqqında daha ətraflı aşağıda danışacağıq.
- **QueryTable** – bu xassə **QueryTable** obyektinə istinad almaq üçün çox vacibdir. Xassə **Range** obyektini üçün **QueryTable** obyektinə istinad alır. **QueryTable** obyektini ilə də proqram səviyyəsində işləmək qaydaları haqqında daha ətraflı aşağıda danışacağıq.
- **Range** – yuxarıda dediyimiz kimi bu xassə ilkin olan diapazon obyektini əsasında yenisinin yaradılmasına imkan verir. Bu halda hücrələrin nömrələndirilməsindəki xüsusiyyətlər haqqında yaddan çıxarmaq lazım deyil.

- **Resize** – cari diapazonu dəyişdirməyə imkan verir. Məsələn, cari diapazonun bir sütun aşağıya və bir sətir sağa artırılması VBA kodu ilə bu cür yerinə yetirilə bilər:


```
oRange.Resize(oRange.Rows.Count+1, oRange.Columns.Count+1).Select
```
- **ShrinkToFit** – bu xassə avtomatik rejimdə diapazondakı mətni elə qurur ki, sütunun eni ölçüsündə yerləşə bilsin.
- **Style** – cari diapazonun stilini təmsil edən eyni adlı **style** obyektini qaytarır. Excel-in qrafik interfeysində həmin əməlləri **Format**→**Style** menyusunda etmək mümkündür.
- **Text** – xassəsi diapazonun birinci hücrəsinin qiymətini **String** tipli alınması üçün imkan yaradır. **Range** obyektində bu xassə yalnız oxunma rejimi üçün əlçatandır.
- **Validation** – bu xassə diapazona daxil edilən qiymətlərin yoxlanmasını icra edən eyni adlı **Validation** obyektini qaytarır.
- **Value** - ən çox istifadə edilən xassələrdəndir: diapazon hücrələrinə qiymət (ədədi, mətn və ya digər tipli verilənin) təyin edilməsi üçün imkan yaradır. **Value2** xassəsi bu xassəyə çox bənzəyir, fərqi yalnız **Currency** və **Date** verilənlər tipini dəstəkləməməsidir.
- **WrapText** – diapazon hücrələrində mətnin başqa sətirə keçirilməsinin qoşulub (və ya söndürülməsini) icra edir.

İndi isə **Range** obyektinin metodları haqqında:

- **Activate()** – cari diapazonu seçərək, kursoru onun birinci hücrəsinə yerləşdirir.
- **AddComment()** - hücrəyə şərh əlavə edilməsi üçün imkan verir: hücrə qırmızı rəngli bucaqcıqla işarə ediləcək, şərh isə həmin yerdən üzərək yuxarı qalxan izahat formasında görsənəcək. Bu metod yalnız bir hücrədən olan diapazona tətbiq edilə bilər (həmin əməli Excel-in qrafik interfeysində **Insert**→**Comment** menyusunu ilə yerinə yetirmək mümkündür).
- **AutoFill()** – diapazonda avto-doldurmanın istifadə edilməsində tətbiq edilir (Məsələn, əgər birinci duran iki hücrə 1 və 2 qiyməti ilə doldurulmuşdursa, onda, avtomatik rejimdə, hücrələrdə 3, 4, 5, v.s. davam ediləcək).
- **AutoFit()** – mətni hücrələrə düzgün yerləşdirmək üçün diapazondakı bütün sütun və sətirlərin, uyğun olaraq, enini və uzunluğunu avtomatik rejimdə dəyişməyə imkan verir. Yalnız tamamilə hücrələr və ya sütunlar toplusundan ibarət diapazonlara tətbiq edilə bilər, digər halda səhv qaytarılacaq.
- **AutoFormat()** – avto-formatlaşmanın hansısa bir stilini tətbiq edir (qrafik interfeysin **Format**→**AutoFormat** menyusundan icra edilə bilər).
- **BorderAround()** – seçilmiş parametrlərlə çərçivəyə diapazonun yerləşdirilməsi üçün imkan yaradır.

- **Clear...** - prefiksi ilə başlayan metodlar diapazonun içərisini təmizləmək lazım olduqda istifadə edilirlər: məslən diapazonları qiymətlərdən, formatlaşmadan, şərhərdən v.s. təmizləmək tələb olduqda.
- **Consolidate()** – seçilmiş aqreqat funksiyasını istifadə edərək bir neçə diapazonun verilənlərini bir diapazonda birləşdirmək üçün nəzərdə tutulub (o cümlədən səhifələrdə də aiddir).
- **Copy()** – diapazonu başqa yerə kopyalamağa imkan verir. Əgər təyinat yeri göstərilməyibsə, onda məlumat dəyişmə buferində yaddaşda saxlanılır. Analoji olaraq, **Cut()** metodu işləyir (ilkin diapazonun verilənləri kəsilib yaddaşda saxlanılır).
- **CopyFromRecordset()** – olduqca əlverişli metoddur: obyektəki seçilmiş diapazonun yuxarıdakı sol bucağından başlayaraq, **ADO Recordset** obyektindən Excel səhifəsinə verilənləri yerləşdirir.
- **DataSeries()** – tarix və zaman funksiyalarının avtomatik rejimdə təyin edilməsi üçün nəzərdə tutulub: təyin edilmiş zaman intervalında diapazondakı zamanın (tarixin) qiymətini, avtomatik olaraq, artırılmasını icra edir.
- **Delete()** – cari diapazonun verilənlərini yox edir. Məcburi olmayan parametrlər kimi, yox edilənlərin yerinə gələn yeni hücrələrin hansı tərəfdən hərəkət etməsini təyin etmək olur.
- **Dirty()** – ingilis dilindən tərcümədə “kirlilən” mənasına gəlir: diapazondakı hücrələri “kirlilən” kimi qeyd edir: nəticədə yenidən hesablanmada həmin hücrələr də hesablanacaq. Adətən bu metodu o halda tətbiq edirlər ki, Excel özü həmin hücrələrin yenidən hesablanmasını təyin edə bilmir. Diapazon hücrələrinin hesablanmasını məcburi rejimə də keçirmək olar: bunun üçün **Calculate()** metodunu istifadə etmək lazımdır.
- **Fill...** - prefiksli metodlar eyni qiyməti, əvvəldən təyin edilmiş istiqamətdə, diapazonun hücrələrində çoxaltmaq üçün nəzərdə tutulub: məsələn, **FillDown()**, **FillUp()**, **FillLeft()**, **FillRight()** - tətbiq edilməsi adlarından aşkardır).
- **Find()** – metodu, diapazon hücrələrində axtarışdan tapılan lazım olan qiymətə görə, diapazonun birinci hücrəsini təmsil edən, yeni **Range** obyektini qaytarır. Bu metodun bir sıra məcburi olmayan parametrləri var: axtarışın istiqamətini, registrin həssaslığını, hücrənin bütün/qismən qiymətlərinin axtarılmasını v.s. təyin etmək üçün. **FindNext()** və **FindPrevious()** metodları **Find()** metodu tapdığından başlayaraq, müxtəlif istiqamətdə axtarış davam etməyi bacarır.
- **GoalSeek()** – Excel funksiyasının qiymətlərinin təyin edilməsində avto-seçim rejimini proqram səviyyəsində yaratmaqdan ötrü nəzərdə tutulub. Excel-in qrafik interfeysində həmin əməli **Tools**→**Options** menyusunda icra etmək olar.

- **Insert()** – metodu diapazona hücrələri elə yerləşdirir ki, o biriləri kənara hərəkət edir (istiqaət – sağa və ya aşağı təyin edilə bilər).
- **Justify()** – diapazon sahəsində mətni eyni qaydada paylanmasına imkan yaradır. Əgər həmin diapazona mətn sığışmırsa, onda həmin mətn yaxındakı hücrələrə paylanacaq (qiymətlərin yenidən yazılması şərti ilə).
- **Merge()** – diapazonun bütün hücrələrini bir hücrədə birləşdirir: bu halda yalnız bir qiymət qalır – yuxarının sol kənarında duran hücrənin qiyməti. Yenidən həmin hücrənin bir neçə yeni hücrəyə bölünməsi üçün **UnMerge()** metodu istifadə edilməlidir.
- **Parse()** – şablon əsasında bir hücrənin bir neçəsinə bölünməsinə imkan yaradır (məsələn, şəhər kodunu telefon kodundan ayırmaq üçün).
- **PasteSpecial()** – bu metod **Copy()** və **Cut()** metodlarını tamalayır: dəyişmə buferində olan məlumatı yerinə qoyulmasına imkan verir (təyin edilmiş əməllərlə, məsələn, vurmaq, bölmək, toplamaq v.s. - mövcud olan verilənlər üçün əlavə verilənləri yerləşdirir). Həmçinin metod dəyişmə buferində hər nə varsa yerləşdirildikdə, yerləşdirmə parametrlərini də göstərir.
- **PrintOut()** və **PrintPreview()** – uyğun olaraq, diapazonu çapa və çapdan əvvəl baxışa ekrana çıxarmağa imkan yaradır.
- **Replace()** – bu üsul **Find()** metodunu tamamlayır: diapazondakı qiymətlərin axtarılmasını və dəyişdirilməsini icra edir.
- **Select()** – lazım olan diapazonu seçir: Excel-də **selection** obyektini yoxdur – lakin bunun əvəzinə seçilmiş obyektini təmsil edən **Range** obyektinin alınması imkanı vardır.
- **Show()** – ekran skrollingini elə icra edir ki, təyin edilmiş diapazon görsənsin.
- **ShowDependents()** – diapazon hücrələri üçün oxları göstərir (yalnız 1-ci asılılıq səviyyəsi üçün). Bu metoda əks olan - **ShowPrecedents()** metodudur.
- **ShowErrors()** – təyin edilmiş hücrənin səhv mənbəyini göstərir.
- **Sort()** – diapazondakı hücrələri sortlaşdırır: sortlaşdırmanı sazlamaq üçün çox sayda məcburi olmayan parametrlər istifadə edilə bilər. **SortSpecial()** – bu metod isə həmin əməlləri Asiya dillərinin xüsusiyyətləri əsasında aparır.
- **SpecialCells()** – olduqca rahat və əlverişli metoddur: müəyyən tipli hücrələri (boş olan, səhvlərlə olan, sabitlərlə olan, xüsusi formatları olan) təmsil edən, **Range** obyektini qaytarır. Məsələn, iki boş hücrəni təmsil edən **Range** obyektini almaq üçün aşağıdakı VBA kodu istifadə edilə bilər:

```
Set oRange2=oRange.SpecialCells(xlCellTypeBlanks)
oRange2.Select 'Yoxlama - bu həqiqətən belədirmi?
```

- **SubTotal()** – metodu diapazon üçün yekun qiyməti hesablaya bilir: aqreqat funksiyasını və bir çox digər parametrləri seçmək mümkündür.
- **Table()** – bu metodla əslində iki obyekt yaratmaq olur: 1) ötürülən sütun və ya sətir əsasında cədvəlin yaradılması; 2) həmin yaradılmış cədvəlin hücrələrində hesablamalar aparmaq üçün funksiyanın yaradılması. Excel VBA sənədləşməsində bu metoda aid bir nümunə göstərilib ki, həmin VBA kodu ilə, avtomatik olaraq, səhifədə vurma cədvəlini generasiya etmək mümkün olur (bu VBA kodunu sınaqdan keçirin):

```

Set dataTableRange=Worksheets("Sheet1").Range("A1:K11")
Set rowInputCell=Worksheets("Sheet1").Range("A12")
Set columnInputCell=Worksheets("Sheet1").Range("A13")
Worksheets("Sheet1").Range("A1").Formula="=A12*A13"
For i=2 To 11
    Worksheets("Sheet1").Cells(i,1)=i-1
    Worksheets("Sheet1").Cells(1,i)=i-1
Next i
dataTableRange.Table rowInputCell, columnInputCell
With Worksheets("Sheet1").Range("A1").CurrentRegion
    .Rows(1).Font.Bold=True
    .Columns(1).Font.Bold=True
    .Columns.AutoFit
End With

```

- **TextToColumns()** – mürəkkəb metod olsa da, əslində çox vacib əməli yerinə yetirməyə imkan yaradır: təyin edilmiş alqoritm əsasında diapazondakı sütunların bir neçə sütunlara bölünməsinə icra edir (çox sayda məcburi olmayan parametr qəbul edir).

12.7 QueryTables kolleksiyası və QueryTable obyektı

Excel.QueryTable obyektı, VBA vasitəsi əsasında verilənlər bazasından Excel cədvəllərinə qiymətlərin proqram səviyyəsində importu (ixrac edilməsi), Excel.QueryTable obyektinin xassələri, metodları və hadisələri

Bir çox praktiki məsələlərdə **Application**, **Workbook**, **Worksheet** və **Range** obyektlərinin imkanları kifayət edir. Məsələn, hansısa məlumatın verilənlər bazasından yerləşdirilməsi üçün, **ADO.Recordset** obyektinin bütövlüklə üzərindən keçərək, lazım olan hər nə varsa Excel səhifəsinə yerləşdirmək olar və sonra VBA vasitəsilə proqram səviyyəsində aşağıdakı sətirlərdə yerləşdirilən verilənlər haqqında yekun məlumatı yazmaq. Excel-də müxtəlif hallar üçün, vəziyyəti xeyli sadələşdirən, bir neçə başqa çox vacib olan xüsusi obyekt də nəzərdə tutulub. Məsələn, həmin verilənlər bazasından məlumatın yerləşdirilməsini bu bölmədə baxılan, **QueryTable** obyektı ilə də aparmaq mümkündür. Bunlara oxşar daha iki xüsusi **PivotTable** və **Chart** obyektı ilə biz növbəti bölmələrdə tanış olacağıq. **QueryTable** obyektinin təyinatı – verilənlər bazasından qayıdan qiymətlər toplusu ilə işləməkdir. Bu obyekt Excel-in qrafik interfeysinin **Data→Import External Data→Import Data** menyusundan əlçatandır. **QueryTable** obyektləri ilə verilənlər mənbəyindən alınan yazılar toplusunun sonradan emal edilməsi üçün Excel səhifəsinə yerləşdirmək mümkün olur. **QueryTable** verilənlər mənbəyi ilə “bir-tərəfli” iş qaydasında işlədikdə də (yəni mənbədən verilənlər Excel-ə yalnız yüklənir, mənbədəki məlumatın dəyişdirilməsinə

ehtiyac yoxdur, mənbə ilə bağlantı olmadan Excel-də həmin məlumatın emal edilməsi isə sonrakı mərhələyə saxlanılır) əlverişli və rahat obyekt kimi istifadəçiyə kömək edir. Excel-də bu cür dəyişikliklər sinxronlaşmasını, məsələn, **Worksheet** obyektinin **Change** hadisəsini tətbiq etdikdə “ələ keçirmə” üsulu ilə reallaşdırmaq olar. Daha asan (əslində daha düzgün) üsul da var: Access imkanlarından istifadə etmək. Adətən Excel-ə verilənləri sonradan emal etmək (yəni Excel-dəki zəngin funksiyalar kitabxanası və qrafiklərin qurulması üçün olan imkanlardan istifadə edərək, hərtərəfli analizin aparılması nəzərdə tutulur) üçün yerləşdirirlər. Bu bölmədə biz mənbədəki verilənlərin Excel-ə yalnız həmin “bir-tərəfli” ötürülməsinə baxacağıq. Adi halda olan kimi, **QueryTable** obyektini səhifədə yaradıb sonradan yerləşdirmək üçün, xüsusi **QueryTables** kolleksiyası lazım gələcək (o, iş səhifəsinə, yəni **Worksheet** obyektinə, mənsubdur və bu obyektin eyni adlı xassəsindən əlçatan olur). **QueryTables** obyektinin xassələri və metodları, biz baxdığımız əksər kolleksiyalarda olan kimi – standartdır. Daha ətraflı baxılması üçün, əlbəttə, **Add()** metodu layıqdır. Bu metodla **QueryTable** obyektini yaradılır və eyni zamanda onu kolleksiyaya daxil edir. **Add()** metodu üç sayda parametrlə qəbul edir:

- **Connection** – bu parametrlər **QueryTable** üçün verilənlər mənbəyidir (**Variant** tipli obyekt şəklində). Verilənlər mənbəyi kimi aşağıdakıları istifadə etmək olar:
 - **OLE DB** və ya **ODBC** qoşulma sətirini (**ODBC** qoşulma sətiri "**ODBC**" ilə başlamalıdır, digərləri **ADO** haqqında yazılan fəsildekilərdən fərqlənir);
 - **ADO** və ya **DAO** standart vasitələri ilə yaradılmış hazır **Recordset** obyektini. Bununla belə, **QueryTable** istinad etdiyi, **Recordset** obyektini dəyişdirərək **QueryTable** obyektinin özünü yeniləşdirmək olar (bir çox səbəblərə görə, **QueryTable** obyektini işlədikdə, bu üsul ən əlverişli sayılır);
 - başqa bir **QueryTable** obyektini (qoşulma sətiri və sorğu mətni ilə birgə);
 - **Web**-sorgusu və ya **Microsoft Query** sorgusunun nəticəsi (**DQY** və ya **IQY** faylları kimi). Bu cür fayllı Excel-in qrafik interfeysinin **Data→External Data Import→New DataBase→Query** menyusu ilə yaratmaq mümkündür.
- **Destination** – alınmış **QueryTable** obyektinin yerləşdirildiyi yerdir. **Range** obyektini ötürülür və yerinə qoyulmuş həmin hücrənin yuxarı sol bucağından başlayır.
- **SQL** – bu məcburi olmayan parametrlə **ODBC** verilənlər mənbəyində icra edilən, **SQL**-sorgusunu təyin etmək olur. Həmin sorgunu **QueryTable** obyektinin eyni adlı xassəsi ilə də təyin etmək mümkündür.

QueryTable obyektini yaradıdıqda, hazır olan **Recordset** obyektini istifadə edilir. Çünki bu halda əldə olan mənfəətlər bunlardır: 1) qoşulmanın sazlanmasının və cursorun maksimum olan imkanlarının əlçatan olması; 2) dəyişikliklərin əlavə edildiyi **Recordset** obyektində (əslində isə operativ yaddaşda) aralıq vəziyyətdə olan verilənlərin olduqca effektiv saxlanması imkanları; 3) **Recordset** obyektinin bütün əlverişli və rahat olan xassə və metodları. Excel səhifəsində

QueryTable obyektini VBA kodu ilə yaradılması aşağıdakı nümunədəki kimi görünə bilər (biz burada, **ADO** haqqında olan bölmədə yazıldığı kimi, **Northwind.Customers** cədvəlinin əsasında olan **Recordset** obyektindən istifadə edirik):

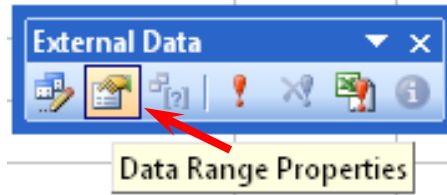
```
Dim cn As ADODB.Connection
Set cn=CreateObject("ADODB.Connection")
cn.Provider="SQLOLEDB"
cn.ConnectionString="User ID=SA;Password=password;Data Source=_
LONDON1;" & "Initial Catalog=Northwind"
cn.Open
Dim rs As ADODB.Recordset
Set rs=CreateObject("ADODB.Recordset")
rs.Open "select*From dbo.customers", cn
Dim QT1 As QueryTable
Set QT1=QueryTables.Add(rs, Range("A1"))
QT1.Refresh
```

Bir başa **QueryTable** obyektinin səhifədə yerləşdirilməsini **QueryTable.Refresh()** metodu ilə icra edirlər. Bu cür əməliyyat yerinə yetirildikdə, **QueryTable** obyektinin yalnız operativ yaddaşa yaradılması mümkündür.

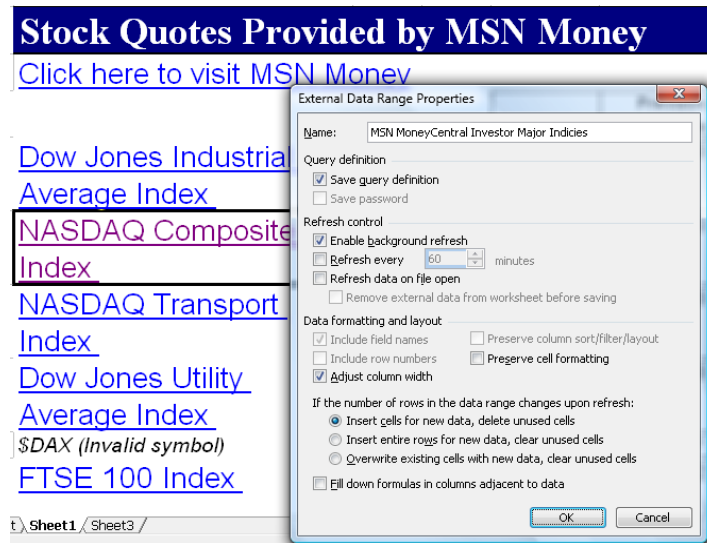
İndi isə **QueryTable** obyektinin ən vacib olan xassələri və metodları haqqında:

- **BackgroundQuery** – istifadəçi Excel-də hansısa başqa işini gördükdə, sorğunun fon rejimdə icra edilməsini təyin edir. Standart halda **True** qiyməti qurulmuş olur - istifadəçi Excel-dəki fəaliyyəti ilə proqramın normal işinə əgər maneçilik törədə bilərsə, onda xassənin **False** qiymətinə keçirmək olar.
- **CommandText** – SQL əmrinin mətnidir, yəni mənbəyə ötürülən sorğu mətnidir. SOL-in analoji xassəsi ilə (o, əks əlaqənin qarşılıqlı əvəz edilməsi üçün saxlanılır) birgə mövcuddur və onun ilə müqayisədə prioriteti daha üstündür. Hazır **Recordset** obyektini **QueryTable** obyektinə göndərdikdə, bu xassə əlçatan olmur.
- **CommandType** – bu xassə **CommandText** obyektinə göndərilən əmrin tipidir (göndərilən əmrin tipləri isə bunlar ola bilər: bütün cədvəl, SQL-sorğusu, kubun adı v.s.).
- **Connection** – qoşulma sətridir: **QueryTables** kolleksiyasının **Add()** metodu çağırılarda onun ötürülməsi mümkün olur. Hazır **Recordset** obyektini ilə işlədikdə xassə əlçatan olmur.
- **Destination** – xassə **QueryTable** obyektinin səhifədə tutduğu diapazonun yuxarıdakı sol hücrəni təmsil edən **Range** obyektini qaytarır: bu xassə **Add()** metodunun ikinci parametridir. **QueryTable** obyektini qurulandan sonra xassə yalnız oxumaq üçün əlçatan olur.
- **EnableEditing** – qrafik ekranda istifadəçinin **QueryTable** obyektinin dəyişdirilməsinin mümkünlüyünü təyin edir (standart halda **True** qiymətində qurulmuş olur). Əgər xassəni **False** qiymətinə keçirsələr, onda istifadəçi **QueryTable** obyektinin yalnız yeniləşməsinə icra edə bilər.

- **EnableRefresh** – yenidən verilənləri **Recordset** mənbəyindən aldıqda, istifadəçinin **QueryTable** obyektinin yeniləşdirmək imkanını təyin edir.
- **FetchedRowOverflow** – mənbədən alınan yazılar Excel səhifəsində yerləşdirilə bilməyəndə (65536 yazıdan çox yazı yükləndikdə), xassə **True** qiymətini alır. Bu halda səhv yaranmır. Məhz buna görə, çox sayda olan yazılar toplusu ilə işlədikdə, uyğun olan yoxlamaların aparılması qaçılmaz tədbir kimi meydana çıxır.
- **FieldNames** – çox vacib xassədir: **QueryTable** obyektinin birinci sətirə mənbədən alınan sütunların adlarının yerləşdirilməsini icra edir. Standart halda **True** qiyməti qurulmuş olur (yeni sütunların adlarının yerləşdirilməsi icra edilir).
- **MaintainConnection** – səhifə bağlanana qədər, mənbə ilə birləşmənin açıq (bağlı) olmasını təyin edir. Standart halda **True** qiyməti qurulmuş olur - zaman-zaman olan yeniləşmələr rejimi optimallaşdırılmış olur. Xassə çox faydalı olan xassələr siyahısındadır: əgər **False** qiyməti qurulsa, onda müştəridəki operativ yaddaşda qənaəti yeniləşmə sürətinin əsasında həyata keçirmək mümkün olur.
- **Name** - xassəsi **QueryTable** obyektinin adıdır (qrafik interfeysində bu xassəyə Excel-in **External Data** idarəetmə panelində **Data Range Properties** düyməsinin basılması nəticəsində baxmaq olar, bax şəkl. 12.4 və şəkl. 12.5).



Şəkil 12.4 Excel-in **External Data** idarəetmə panelində **Data Range Properties** düyməsinin basılması.



Şəkil 12.5 Excel-in **External Data Range Properties** pəncərəsində **Name** xassəsinin görünməsi (**QueryTable** obyektinin adı kimi bu nümunədə məşhur **MSN MoneyCentral Investor Indices** verilənlər bazasının adı görünür – Excel-dəki avtomatik rejim ilə internet mənbəyindən götürülür).

- **Parameters** – xassəsinin köməyi ilə **Parameters** kolleksiyasını (sorgular parametrlərinə) əlçatan edir. Buradakı imkanlar **Recordset** obyektinin parametrləri ilə işləmək imkanları üst-üstə düşür.
- **PreserveColumnInfo** və **PreserveFormatting** – yeniləşmə **QueryTable** obyektində aparıldıqda, sütunlar (sortlaşmada, filtrasiyada v.s.) və formatlama haqqında məlumatın saxlanması (yaxud saxlanmamasını) təyin etmək olur. Standart halda – hər şey yaddaşda saxlanmalıdır.
- **QueryType** - bu xassə ilə **QueryTable** obyektini yaradıldıqda, nədən istifadə edilməsini müəyyən etmək olur (**Recordset** ilə mi, cədvələ bir başa əlçatma əsasında mı, SQL-sorgusu ilə mi v.s.).
- **Recordset** – imkan verir ki, **QueryTable** yaradılmasında istifadə edilən **Recordset** obyektinə istinad alınsın və ya onu həmin **QueryTable** obyektini üçün başqası ilə əvəz edilməsi icra edilsin (dəyişmələr yalnız **Refresh** () metodu çağırıldıqda baş verir).
- **Refreshing** – bu xassə mənbəyə fon sorgusu icra edildiyi anda **True** qiymətini alır. Əgər sorğunun icrası xeyli gecikirsə, onda əməliyyatın kəsilməsi üçün **CancelRefresh** () metodundan istifadə etmək olar.
- **RefreshOnFileOpen** – səhifə açıldıqda verilənləri avtomatik rejimdə yeniləşdirmək və ya yüklənmiş qiymətlərlə kifayətlənmək (standart halda bu vəziyyət qurulmuş olur) vəziyyətini təyin edir.
- **RefreshPeriod** – mənbədən olan məlumatı **QueryTable**-də hansı zaman intervallarında avtomatik rejimdə yeniləşdirməyi təyin edir. Standart halda – **0** qiyməti qurulmuş olur (yəni avtomatik yeniləşmə söndürülmüş olur).
- **RefreshStyle** – yeniləşməni, **QueryTable**-də mövcud olan hücrələrdə apardıqda (yeni əvvəlkiylə yeniləri ilə dəyişdirildikdə), bu xassə ilə mövcud olan hücrələrlə nə edilməsini təyin etmək olur.
- **ResultRange** – bu xassə isə **QueryTable** obyektinin ən vacib xassəsidir: **QueryTable** obyektindən səhifəyə yerləşdirilmiş diapazonu almağa imkan verir (diapazonda bütün yüklənmiş hücrələr təmsil olunur). Məlumdur ki, sonrakı addımlarda verilənlər bazasından yüklənmiş həmin verilənlər üzərində müəyyən əməllər aparılacaq (adətən sütunlar və sətirlər üzərində). Bu metodun işləməsi üçün mütləq **Refresh** metodu ilə **QueryTable** verilənlərinin səhifəyə yerləşdirilməsini icra etmək lazımdır. Yalnız bundan sonra xassənin adı diapazon kimi qaytardığı obyektindən istifadə etmək mümkün olur. Bu metodun işləməsinin nümayişi üçün aşağıdakı ən sadə üsuldən istifadə etmək olar – bir sətirik VBA kodundan istifadə etməklə:

```
QT1.ResultRange.Select
```

Aşağıdakı digər VBA kodu ilə **QueryTable** obyektinin birinci sütunu altında həmin sütunun bütün qiymətlərinin cəmini generasiya edir:

```
Set c1=Sheets("Sheet1").QueryTables(1).ResultRange.Columns(1)
c1.Name="Column1"
c1.End(xlDown).Offset(1, 0).Formula="=SUM(Column1)"
```

- **RowNumbers** – xassəsi **QueryTable** ilə alınmış verilənlərlə aparılan işi xeyli asanlaşdırır: **QueryTable**-də (solda), daha bir sütunu generasiya edir (həmin sütun isə **QueryTable**-dən alınmış yazıların nömrələrindən ibarətdir).
- **SaveData** – Excel kitabı ilə birgə **QueryTable**-dən alınmış verilənlərin saxlanılmasını və ya saxlanılmamasını təyin edir. Standart halda **True** qiyməti qurulmuş olur (yəni yadda saxlanılması qurulmuş olur). Əgər istifadəçi əvvəldən yalnız mənbədən alınmış ən yeni verilənlərlə işləyəcəksə, onda **False** qiyməti qurulmalıdır.
- **SavePassword** – parolun qoşulma sətiri ilə birgə saxlanılmasını təyin edir (bu xassəni yalnız **ODBC** mənbələri üçün istifadə etmək olar). Xassənin qiyməti **False** kimi seçilsə, onda proqramın təhlükəsizliyi artırılmış olacaq.
- **SourceDataFile** – mənbə faylının tam yolu və adını təmsil edir (**Access**, **DBF** və digər kiçik miqyaslı verilənlər bazasının idarəetmə sistemləri üçün). Müştəri-server sistemləri üçün (məsələn, **SQL Server** kimi) - **Null** qiymətini qaytarır.
- **Text...** - prefiksi ilə başlayan xassələr mətn faylının çox saylı parametrlərini təyin edir (əgər bu fayl **QueryTable** üçün mənbə kimi seçilibsə).
- **Web...** - prefiksli başlayan xassələr **Web**-mənbəyə olan sorğulardan alınan verilənlərin parametrlərini təyin etməkdə kömək edir.

QueryTable obyektinin **Refresh()**, **CancelRefresh()**, **Delete()** metodlarının mənası hətta adlarından aşkardır, və buna görə xüsusi izahata ehtiyac qalmır. **ResetTimer()** metodu avtomatik yeniləşmə taymerini sıfırlamağa imkan verir. **SaveAsODC()** metodu isə verilənlər mənbəsinin təyin edilməsini **Microsoft Query** faylı kimi saxlanılmasına imkan verir (əgər verilənlər mənbəyi kimi **Recordset** istifadə edilmişdirsə, onda metod səhvi qaytaracaq). **QueryTable** obyektinin iki sayda hadisəsi vardır: **BeforeRefresh** və **AfterRefresh** – onlar, uyğun olaraq, mənbədən verilənlərin yüklənməsi başladığından əvvəl və qurtaran anda işə salınır.

12.8 Yekun cədvəllər ilə işləmə (**PivotTable** obyektini)

Excel.PivotTable obyektini, VBA ilə Excel-də OLAP kublari və yekun cədvəlləri ilə proqram səviyyəsində işləmə, PivotCache obyektini, yekun cədvəlinin maketinin yaradılması

Bir çox müəssisələrdə iş məqamında fəaliyyət haqqında emal edilməmiş verilənlər (*raw data*) yığılıb qalır. Məsələn, ticarət müəssisəsi üçün malların satışları haqqında verilənlər yığıla bilər

(şəbəkə əlaqəsi müəssisəsində hər bir məhsulun satışı üçün). Müəssisə menecmenti üçün isə əksər hallarda emal edilməmiş informasiya əsasında analitik informasiyanın generasiyası lazım olur – məsələn, hər bir məhsulun müəssisənin gəlirinə təsiri və ya hansısa konkret olan obyektin zonasında olan xidmətin keyfiyyəti. Emal edilməmiş informasiyadan bu məlumatların çıxarılması çox çətin işdir: olduqca mürəkkəb strukturlu SQL-sorğusunu icra etmək lazım olur. Nəticədə çox vaxt tələb olur və buna görə işə maneçilik yaranır. Buna görə son zamanlar bu cür emal edilməmiş verilənləri əvvəl arxiv verilənləri bazasına yerləşdirirlər - **Data Warehouse**, sonra isə **OLAP (online analytical processing)** kublarına (sonuncular isə interaktiv analiz üçün çox rahat vasitədir).

OLAP kublarını daha yaxşı anlamaq üçün çox ölçülü cədvəlləri, necə ki, bir analoq kimi, təsəvvür etmək olar. Həmin cədvəllərdə standart iki ölçülü struktur (adi cədvəllərdəki sütunlar və sətirlər) əvəzinə ölçülərin sayı daha çox ola bilər. Adətən kubdakı ölçüləri ifadə etmək üçün "*hansısa parametrin kəsiyində*" terminindən istifadə edirlər. Məsələn, marketinq şöbəsinin zaman kəsiyində, regional kəsiyində, məhsulların tipləri kəsiyində satışlar kanalları kəsiyinə aid məlumat lazım ola bilər. Kublar köməyiylə (standart SQL-sorğularından fərqli olaraq) "**regional distribyutorlar vasitəsi ilə Şimali-Şərq regionunda keçən ilin dördüncü rübündə hansısa tip məhsul hansı sayda satılmışdır**" sualına cavab almaq çox asan olur. Əlbəttə, adi verilənlər bazasında bu cür kubların yaradılması mümkün deyil. OLAP kubları ilə işləmək üçün xüsusiləşmiş proqram məhsulları lazımdır. Məsələn, Microsoft şirkəti **Analysis Services** adlanan **SQL Server**-lə birgə **OLAP** verilənlər bazasını təqdim edir. Oracle, IBM, Sybase v.s. şirkətləri də OLAP tipli verilənlər bazalarını istifadəçilər üçün təqdim edir.

Excel-də bu cür kublarla işləmək üçün xüsusi müştəri quraşdırılmışdır: **Pivot Table** (Azərbaycanca tərcümədə **Yekun cədvəl**). Excel-in qrafik interfeysində o, **Data→PivotTable and PivotChart Report** menyusundan əlçatandır. Uyğun olaraq, bu müştərini təmsil edən obyekt **PivotTable** adlanır. Qeyd etməliyik ki, o yalnız OLAP kubları ilə işləmir. Əlavə olaraq, Excel-in adi cədvəlləri və digər verilənlər bazaları ilə də işləyə bilər – ancaq bu hallarda bir sıra imkanlar əlçatan olmur. Yekun cədvəl və **PivotTable** obyektini - **Panorama Software** firmasının məhsuludur (Microsoft onu alaraq, Excel-ə inteqrə etmişdir). Buna görə **PivotTable** obyektini ilə işləmə qaydaları Excel VBA-nın başqa obyektləri ilə müqayisədə bir qədər fərqlənir. Məhz buna görə, həmin obyektlə işləməyi bacarmaq üçün, oxucunun diqqətini həmin istiqamətə yönəltmək istərdik. Çünki praktiki cəhətdən **PivotTable** çox vacib obyektidir. Eyni qaydada aparılan işlərin sayı yekun cədvəllərdə olduqca böyük ola bilər. Buna görə işlərin avtomatlaşdırılmasına (yəni VBA Excel tətbiqi proqramlarına) burada böyük ehtiyac yaranır.

Bəs yekun cədvəl ilə proqram işini nə cür təsvir etmək olar?

Birinci növbədə gərəkli olan iş – **PivotCache** obyektinin yaradılmasıdır: sonra həmin obyekt OLAP mənbəyindən alınan yazılar toplusunu təmsil edəcək. Çox şərti olaraq, bu obyektini **QueryTable** obyektini ilə müqayisə etmək olar. Hər bir **PivotTable** obyektini üçün yalnız bir sayda olan **PivotCache** obyektini istifadə edilə bilər. **PivotCache** obyektinin yaradılması **PivotCaches** kolleksiyasındakı **Add()** metodu ilə həyata keçirilir:

```
Dim PC1 As PivotCache
```

```
Set PC1=ActiveWorkbook.PivotCaches.Add(xlExternal)
```

PivotCaches – standart kolleksiyadır, onun metodlarından diqqətə layiq yalnız **Add()** metodu ola bilər. Bu metod isə iki parametri qəbul edir:

- **SourceType** – məcburidir, yekun cədvəl üçün verilənlər mənbəyinin tipini təyin edir. **PivotTable** obyektinin yaradılmasına bir neçə yol ilə nail olmaq olar: Excel diapazonu əsasında, verilənlər bazasından olan verilənlər əsasında, xaricdə olan mənbədəki verilənlər əsasında, başqa bir **PivotTable** ilə v.s. Praktiki məsələlərdə OLAP-ın tətbiqinə ehtiyac yalnız verilənlərin sayı həqiqətən çox olduqda yarana bilər. Çünki bu halda uyğun olan xarici xüsusiləşmiş saxlanma yeri lazım olacaq, məsələn, **Microsoft Analysis Services** kimi. Bu halda **xlExternal** qiyməti təyin edilməlidir.
- **SourceData** – bütün hallarda məcburidir, yalnız bir istisna haldan başqa – birinci parametrin qiyməti **xlExternal** olduqda. **PivotTable** obyektinin yaradılması üçün istifadə edilən verilənlər diapazonunu təyin edir. Adətən **Range** obyektini qəbul edir.

Növbəti məsələ - **PivotCache** obyektinin parametrlərinin sazlanmasından ibarətdir. Əvvəl dediyimiz kimi, bu obyekt bir qədər **QueryTable** obyektini xatırladır: onların xassələr və metodlar toplusu çox oxşardır. Bəzi daha vacib olan xassələri və metodları qeyd edək:

- **ADoConnection** – bu xassə ilə **ADO Connection** obyektini qaytarıla bilər (avtomatik olaraq, xaricdə olan verilənlər mənbəyinə qoşulur). Qoşulmanın əlavə sazlanması üçün istifadə edilir;
- **Connection** – eyni ilə **QueryTable** obyektinin eyni adlı xassəsi kimi işləyir: hazır **ReCorset** obyektini, qoşulma sətirini, mətn faylını, Microsoft Query faylının və Web-sorğunu qəbul edə bilər. OLAP-la işlədikdə, daha çox qoşulma sətiri bir başa yazılır (çünki **Recordset** obyektini ilə verilənlərin dəyişdirilməsi üçün yeniləşmənin mənası olmayanda – OLAP verilənlər mənbəyi yalnız oxumaq üçün əlçatan olur). Məsələn, **Foodmart** adlı verilənlər bazasına (**Analysis Services**-in treninq üçün yaradılmış verilənlər bazasıdır) **LONDON** adlı serverdən qoşulması proqram səviyyəsində bu cür təsvir edilə bilər:

```
PC1.Connection="OLEDB;Provider=MSOLAP.2; Data Source=_  
LONDON1;" & "Initial Catalog=FoodMart 2000"
```

- **CommandType** və **CommandText** – xassələri verilənlər bazası serverinə ötürülən əmr tipini və əmrin özünü təsvir edirlər. Məsələn, **Sales** adlı kuba müraciət etmək və onu bütövlükdə müştəriyə keşə almaq üçün aşağıdakı VBA kodu istifadə edilə bilər:

```
PC1.CommandType=xlCmdCube  
PC1.CommandText=Array("Sales")
```

- **LocalConnection** - xassəsi Excel vasitəsi ilə yaradılmış, lokal kuba (***.cub** faylına) qoşulmağa kömək edir. Microsoft bu faylları “istehsalat” tipli və həcmli verilənlərlə istifadə

edilməsini məsləhət görmür – yalnız müxtəlif maketlərin yaradılmasında fayda qazanmaq olar;

- **MemoryUsed** – bu xassə ilə **PivotCache** obyektinin istifadə etdiyi operativ yaddaşın miqdarını qaytarmaq olur. Əgər **PivotTable** obyektini **PivotCache** ilə yaradılmayıbsa və açılmayıbsa, onda **MemoryUsed** xassəsi “0” qiymətini qaytarır. Əlavə olan “zəif” müştəri ilə işlədikdə xassəni yoxlamalar üçün istifadə etmək mümkün olur;
- **OLAP** – xassəsi əgər **PivotCache** OLAP serverinə birləşibse, onda **True** qiymətini qaytarır;
- **OptimizeCache** – keşin strukturunu optimallaşdırmağa imkan yaradır. Verilənlərin başlanğıc yüklənməsi daha çox vaxt aparır. Sonra işin sürəti arta bilər. Bu xassənin OLE DB mənbəyi ilə işləmək üçün imkanı yoxdur;

PivotCache obyektinin digər xassələri **QueryTable** obyektinin analoji olan xassələri ilə üst-üstə düşür və buna görə burada baxmayacağıq.

PivotCache obyektinin əsas metodu - **CreatePivotTable()** metodudur. Bu metod əsasında üçüncü mərhələ icra edilir – yekun cədvəlin (yəni əslində **PivotTable** obyektinin) yaradılması. Bu metod dörd sayda parametrlə qəbul edir:

- **TableDestination** – yeganə məcburi parametrdir. **Range** obyektini qəbul edir: onun yuxarı sol küncünə yekun cədvəl yerləşdirilir;
- **TableName** – yekun cədvəlin adını təmsil edir: əgər ad verilməyibsə, onda, avtomatik olaraq, “**PivotTable1**” generasiya edilir;
- **ReadData** - əgər **True** qiyməti qurulsa, onda kubun bütün tərkibində olan avtomatik olaraq, keşə qoyulacaq. Bu parametrlə çox ehtiyatlı olmaq lazımdır – ehtiyatsız tətbiq edilməsi müştəriyə olan yükü birdən artırma bilər;
- **DefaultVersion** – bu xassə adətən göstərilmir. Yaradılan yekun cədvəlin tipinin təyin edilməsində kömək edir. Standart halda ən yeni versiya istifadə edilir;

Excel kitabının birinci səhifəsinin birinci hücrəsində yekun cədvəlin proqram səviyyəsində yaradılması VBA kodu ilə bu cür təsvir edilə bilər:

```
PC1.CreatePivotTable Range("A1")
```

Mənbədən sahələri yükləmək üçün onda dörd ərazi nəzərdə tutulubdur:

- **sütunlar** ərazisi – burada hədləri az olan ölçülər yerləşdirilir (verilənlər analiz edilən “kəsik”);
- **sətirlər** ərazisi - hədləri çox olan ölçülər yerləşdirilir;
- **səhifə** ərazisi – yalnız filtrasiya ediləsi ölçülərə aiddir (məsələn, hansısa regiona aid və ya hansısa ilə aid verilənlər göstərmək lazım olduqda);
- **verilənlər** ərazisi – cədvəlin mərkəzi hissəsi: analiz edilən verilənlər (məsələn, satışın yekunu);

İstifadəçi tərəfindən elementlərin hər bir əraziyə düzgün yerləşdirilməsinin ehtimalı çox azdır. Həm də bu əməliyyatlara çox zaman tələb ola bilər. Buna görə daha əlverişli üsul – verilənləri yekun cədvələ proqram üsulları ilə yerləşdirməkdir. Bu əməliyyatlar üçün `CubeField` obyektini lazımdır. Obyektin ən əsas xassəsi – `Orientation` xassəsidir (bu və ya digər sahənin yerini təyin edir). Məsələn, İstehlakçı ölçüsünün sütunlar ərazisinə yerləşdirilməsini bu cür proqramla həyata keçirmək olar:

```
PT1.CubeFields("[İstehlakçı"]).Orientation=xlColumnField
```

Sonra, məsələn, Zaman ölçüsünü sətir ərazisinə yerləşdirmək olar:

```
PT1.CubeFields("[Zaman]").Orientation=xlRowField
```

Sonra isə - məsələn, Məhsul ölçüsünü səhifə ərazisinə yerləşdirmək olar:

```
PT1.CubeFields("[Məhsul]").Orientation=xlPageField
```

Nəhayət, Satış Vahidi ölçüsünü yerləşdirmək olar:

```
PT1.CubeFields("[Measures].[Satış Vahidi]").Orientation=xlDataField
```

Beləliklə, yekun cədvəl yaradılıb, və onunla işləmək olar. Bəzən daha bir əməli də aparmaq lazım olur – ölçü iyerarxiyasının lazımı təbəqəsini açmaq. Məsələn, əgər bizi rüb-rüb aparılan analiz maraqlandırırsa, onda **Zaman** ölçüsünün **Rüb** təbəqəsi açılmalıdır (standart halda yalnız ən yuxarı təbəqə göstərilir). Əlbəttə, bunların hamısını istifadəçi özü də edə bilər – lakin onun mausla haraya şıqqıldatması haqqında əvvəlcədən fərziyə qurmaq çox çətin olur. Məsələn, proqram səviyyəsində **Zaman** ölçüsünün iyerarxiyasını 2010-cu il rübləri təbəqəsində açılmasını `PivotField` və `PivotItem` obyektləri ilə icra etmək olar:

```
PT1.PivotFields("[Zaman].[İl]").PivotItems("[Zaman].[2010]")._
.DrilledDown=True
```

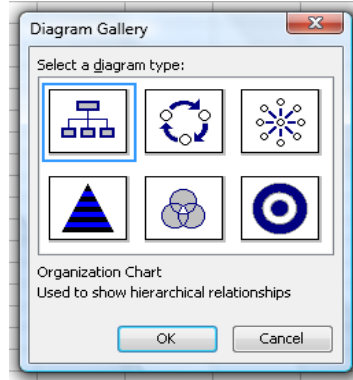
12.9 Chart obyektini: diaqramlarla işləmə

Excel.Chart obyektini, VBA vasitələri ilə diaqramlarla proqram səviyyəli iş, Diaqramın tipinin seçilməsi, cərgələrin əlavə edilməsi

Excel-in ən vacib tətbiqi - verilənlərin analizi ilə bağlıdır. Verilənlərin analizi məlumdur ki, daha əlverişli üsul – xüsusi imkanları olan (məsələn, trendlərin yaradılması imkanı ilə birgə) diaqramların istifadə edilməsidir. Buna görə praktikada bir birinə oxşar diaqramların yaradılmasının avtomatlaşdırılması məsələləri nə tez-tez rast gəlir (adətən diaqramlar verilənlər bazasından alınmış verilənlər əsasında qurulur).

Azərbaycan dilində Excel-dəki diaqramlarla bir qədər terminoloji dolaşıqlıq mövcuddur: biz adi istifadədə diaqram sözünü işlətdikdə, ingilis dilində diaqrama **Chart** deyilir (tələffüzdə "kart" kimi eşidilməlidir). Excel-in qrafik interfeysində diaqramla işləmək üçün **Insert**→**Chart** menyusundan istifadə edilir. VBA Excel-də diaqramlarla işləmək üçün **Chart** obyektini nəzərdə tutulubdur. Bununla belə nəzərə alınmalıdır ki, Excel-in obyekt modelində həmçinin **Diagram** obyektini də nəzərdə

tutulub – və bu obyektı qətiyyən **Chart** obyektı ilə qarışdırmaq lazım deyil, çünki **Diagram** obyektı əslində yalnız əlaqələr sxemini təmsil edir. Excel-in qrafik interfeysində **Insert**→**Diagram** menyusu ilə əlçatan olur (nəticədə sxemlər ilə işləmək üçün şəkl. 12.6-da göstərilən dialoq pəncərəsi əmələ gəlir).



Şəkil 12.6 Excel-in qrafik interfeysindən yaradılan sxemlərlə işləmək üçün Diagram Gallery dialoq pəncərəsinin yaradılması (**Diagram** obyektini **Chart** obyektı ilə qarışdırmaq səhv salmaq olmaz).

Buna görə sonrakı izahatlarımızda diaqram dedikdə Excel-də yaradılan riyazi qrafiklər nəzərdə tutulmalıdır. Excel-də diaqramlar **Chart** obyektı ilə yaradılır. Bu obyektlə proqram səviyyəsində işləmək üçün daha yaxşı olar ki, həmin obyekt elan edilsin:

Dim oChart **As** Chart

Sonra diaqramın yaradılmasına başlamaq olar. Diaqramın yaradılması bizim əvvəllər çox sayda istifadə etdiyimiz üsul ilə həyata keçirilməlidir – **Charts** kolleksiyasının **Add()** metodu çağırılmalıdır:

```
Set oChart=ActiveWorkbook.Charts.Add(,ActiveSheet)
```

Prinsip etibarı ilə diaqram artıq yaradılıbdır. Onun heç bir xassəsi təyin edilməmişdir – buna görə boş səhifə kimi görünəcək. Bu obyekt məzmununu alsın deyə əlavə bir neçə sayda əməllərə ehtiyac var.

Birincisi (və yeganə mütləq əməl) – diaqram üçün verilənlər mənbəyinin təyin edilməsidir ki, bunun üçün VBA Excel-də **SetSourceData()** metodu nəzərdə tutulub. Verilənlər mənbəyi kimi yalnız **Range** obyektı istifadə edilə bilər (o, həmin metodun birinci və yeganə məcburi parametri kimi ötürülür). Metodun ikinci (məcburi olmayan) parametri isə verilənlərin hansı qaydada sayılmasını təyin edir (yəni əvvəl sütunlarla və ya sətirlərlə sayılma icra edilməlidir). Məsələn, bizim yuxarıdakı nümunə üçün həmin proqram kodunu bu cür yazmaq olar:

```
oChart.SetSourceData(Sheets("Sheet1").Range("A1:A10"))
```

Prinsip etibarı ilə yaradılmış kod işə salınsa diaqram artıq yaradılmış olacaq. Seçiləsi olan digər parametrlər isə standart haldakı kimi qurulmuş olur. Lakin praktikada ən azı heç olmasa, diaqramın tipini təyin etmək lazımdır (standart halda o, "**Clastered Column**" adlı histoqram görkəmində

olur). Bunun üçün **ChartType** xassəsi istifadə edilir (Microsoft bu xassə üçün 73 sayda qiymət nəzərdə tutub). Məsələn, diaqramın adı qrafik şəklinə çevirmək üçün aşağıdakı VBA kodu tətbiq edilə bilər:

```
oChart.ChartType=xlLineMarkers
```

Daha bir vacib məsələ - diaqrama əlavə cərgələrin qoşulmasıdır. Bu məqsədlə **Series** obyektinə istinad almaq lazımdır – cərgə, sonra həmin cərgə üçün **Values** xassəsini təyin etmək lazımdır (qiymət kimi ona **Range** obyektini ötürülür):

```
Dim oSeries As Series
Set oSeries=oChart.SeriesCollection.NewSeries
oSeries.Values=Worksheets(1).Range("B1:B10")
```

Bəzən istifadəçilər istəyirlər ki, diaqramlar ayrıca səhifədə yox, məhz verilənlər yerləşən səhifədə yerləşsin. Standart halda diaqram operativ yaddaşda yaradılır və ayrıca səhifəyə yerləşdirilir. Əgər diaqramı verilənlər yerləşən Excel cədvəlinin özündə yerləşdirmək lazımdırsa, onda diaqram əvvəlcə ayrıca səhifədə yaradılmalıdır, sonra isə **Location** metodu ilə lazım olan səhifəyə köçürülməsi icra edilir. Əvvəlcə yaradılmış ayrıca səhifə onda avtomatik olaraq silinir:

```
oChart.Location xlLocationAsObject, "Sheet1"
```

Diqqət verin ki, **Location** metodu öz birinci parametri kimi sabitlərdən birini qəbul edir (**xlLocationAsNewSheet** – xüsusi yaradılmış yeni səhifəyə köçürmək, **xlLocationAsObject** – obyektə, yeni səhifəyə köçürmək), ikinci parametr kimi isə - düşünülməmiş kimi, səhifənin obyektini yox, mütləq səhifənin adı qəbul olunur. Əgər proqram kodunu Excel-in həm İngilis və Azərbaycan versiyalarında istifadə etmək lazım olarsa, onda səhifənin adını proqramlaşdırma vasitəsi ilə alınması daha əlverişli sayılmalıdır.

Location metodu ilə bağlı böyük problem və çətinlik onunla bağlıdır ki, diaqram səhifənin içərisinə yerləşdirildikdən sonra diaqrama olan istinad itir. Buna görə həmin diaqramın obyektini yenidən tapmaq lazım olur. **Chart** obyektinə təkrar müraciət cəhdini etdikdə "**Automation Error**" məlumatı alınır. Buna görə daha əlverişli çıxış yolu – **Location** metodunun çağırışını diaqrama aid proqram kodunun sonunda yerləşdirməsidir. Çünki digər halda yaradılmış diaqram gərək yenidən axtarılacaq və ona yenidən obyekt istinadı alınsın, məsələn, aşağıdakı nümunədəki kimi:

```
Dim oSeries As Series
Set oSeries=Worksheets(1).ChartObjects(1).Chart.SeriesCollection.NewSeries
oSeries.Values=Worksheets(1).Range("B1:B10")
```

Diaqramın çox saylı digər parametrləri **Chart** obyektinin xassələri və metodları ilə sazlanıla bilər:

- **ChartArea** – xassəsi eyni adlı **ChartArea** obyektini qaytarır (diaqramın tutduğu sahəni təmsil edir və diaqramın xarici görünüşünün sazlanması üçün istifadə edilir: məsələn, **Font**, **Interior** xassələri v.s.). Əgər diaqramın bütöv yox, yalnız qrafikin qurulduğu yerdəki hissəsinin xarici görünüşü sazlanmalıdırsa, onda **PlotArea** xassəsi istifadə edilir. Standart halda diaqram səhifənin dəqiq mərkəzində yerləşdirilir. Əgər diaqram səhifənin hansısa təyin edilmiş sahəsində yerləşdirilməlidirsə, onda **ChartArea** obyektinin bizə artıq tanış olan **Top**, **Height**, **Left** və **Width** xassələri istifadə edilir;
- **ChartTitle** - xassəsi eyni adlı obyektini qaytarır: həmin obyektlə diaqramın başlığını **Text**, **Font**, **Border** v.s. xassələri ilə sazlamaq olur;
- **ChartType** - ən vacib xassələrdəndir: diaqramın tipini təyin edir;
- **HasDataTable** – bu xassənin qiyməti **True** ilə qurulsa, onda diaqramın aşağı hissəsində (standart halda) diaqramın qurulması üçün, mənbə kimi götürülmüş ədədlərlə dolu cədvəl əmələ gələcək. Eyni zamanda, avtomatik olaraq, həmin cədvəlin görüntüsünü sazlamağa imkan verən **DataTable** proqram obyektini də yaranacaqdır. Eyni prinsip əsasında **HasLegend**, **HasPivotFields** və **HasTitle** xassələri fəaliyyət göstərir;
- **Name** – diaqramın adının sazlanmasını təmin edir (Excel-in əlavə qurmasında olan kimi). Standart halda qurulan diaqramlar bu cür adlandırılır: “**Chart1**”, “**Chart2**” v.s.;
- **SizeWithWindow** - əgər bu xassədə **True** qiyməti seçilsə, (standart halda **False** qiyməti qurulmuş olur), onda diaqramın ölçüləri, avtomatik olaraq, səhifənin ölçülərinə uyğunlaşdırılacaqdır;
- **Tab** – bu xassə haqqında az adam bilir: eyni adlı obyektə Excel kitabındakı qrafik interfeysin diaqramı üçün əlavə qurmasını (sadəcə səhifəni) sazlamaq olur. Məsələn, əlavə qurmanı yaşıl rəngə boyamaq üçün aşağıdakı kod tətbiq edilə bilər:

```
oChart.Tab.Color=RGB(0,255,0)
```

- **Visible** – diaqramın silinmədən gizlənməsini təmin edir.

Digər xassələr əsasən 3-D diaqramların görünməsinə və diaqramların istifadəçi tərəfdən dəyişdirilməsinin qarşısının alınması ilə bağlıdır.

İndi isə - **Chart** obyektinin ən əsas sayılan metodları haqqında:

- **Activate()** – metodu tez-tez istifadə edilən metodlardandır: diaqramı aktivləşdirir (sadəcə diaqrama keçidi təmin edir);
- **ApplyCustomType()** – istifadəçi tərəfindən tərtib edilmiş istifadəçi xassəli diaqramın yaradılmasını təmin edir (bunun üçün əvvəlcə həmin tip diaqram üçün şablon yaradaraq, onu şablonlar qalereyasına qoymaq lazımdır);

- **ApplyDataLabels** () – metodu ilə diaqramı nişanlama qoyaraq, verilənləri diaqramda göstərmək olur. Bu metodun nişanlama görüntüsü üçün çox sayda parametrləri vardır (məsələn, qiymətlər göstərsin və ya yox v.s.);
- **Axes** () - diaqramın oxlarını təmsil edən obyekt qaytarır. Sonradan həmin obyekt alınmış oxların sazlanması üçün istifadə etmək olar;
- **ChartWizard** () – bu metod ən qısa zaman rejimində yenidən diaqramın formatlaşdırılmasına imkan yaradır (effekt təxminən Excel-in qrafik interfeysinin diaqramların qurulması üçün istifadə edilən **Chart Wizard** alətinin funksionallığına bənzəyir). Bu metodla tərtib edilən bircə sətir VBA kodu başqa üsullarla ən azı üç sətirlik VBA kodu ilə müqayisədə eyni güclüdür;
- **Copy** () – diaqramın kitabın başqa bir yerinə köçürülməsinə imkan verir (məsələn, başqa diaqramın mövcud olanın əsasında qurulması). Mövcud plan diaqramın başqa yerə köçürmək üçün **Location** () və ya **Move** () metodlarından da istifadə etmək olar;
- **CopyPicture** () – çox dəyərli metoddur: diaqramı görüntü kimi dəyişmə buferinə yerləşdirir. Sonra bu görüntünü Word sənədinə və ya digər yerə qoymaq olar. Başqa variant budur – **Export** () metodundan istifadə etmək (diskdə fayl formasında təmsil edilən diaqram əsasında şəkil yaradılmasına imkan verir);
- **Delete** () – sadəcə diaqramı silərək yox edir;
- **Evaluate** () – Excel kitabında lazımı diaqramın adına görə onun tapılmasını təmin edir;
- **PrintOut** () – diaqramın çapa göndərilməsini təmin edir. Çapı tənzimləmək üçün metodun çox sayda parametri vardır;
- **Refresh** () – diaqramın qurulması üçün istifadə edilən qiymətlərlə bağlı yeniləşdirilməyə imkan yaradır;
- **Select** () – diaqramın seçilməsini təmin edir (mausun bir dəfə şıqqıltısına analoji effekt yaradır). Onun əksi olan **Deselect** () metodu isə seçilməni yox edir (klaviaturanın **<Esc>** düyməsinə analoji effekt yaradır);
- **SetBackgroundPicture** () – diaqramı fon şəklinin qoyulmasına imkan yaradır (fon şəkli çox parlaq olmasa daha yaxşı olar);
- **SetSourceData** () – çox vacib üsuldur: diaqramın qurulmasında istifadə edilən verilənləri təyin edir (bu haqda biz artıq məlumat vermişik);

Chart obyekt üçün çox sayda hadisələr də nəzərdə tutulmuşdur: maus şıqqıltısına, seçmələrə, seçmələrin çıxarılmasına, verilənlərin sayılmasına, ölçülərin dəyişməsinə yaranan müxtəlif rekursiyalarla bağlı. Proqram səviyyəsində olan praktiki işlərdə bu cür hadisələrdən çox az sayda istifadə edirlər.

12.10 Excel-in VBA proqramlaşdırılmasında vacib olan başqa obyektleri haqqında

Excel.CellFormat, Excel.ListObject, Excel.Validation, Excel.Watch obyektleri

Excel-də çox sayda başqa obyektler də mövcuddur - onların daha ətraflı baxılması bu kitabın çərçivəsindən kənarıdır. Xüsusilə nəzərə alınmalıdır ki, MS Office paketinin hər yeni versiyasında (məsələn, MS Office 2007 və MS Office 2010), digər proqramlarda olan kimi, Excel VBA proqram təminatında xeyli sayda yeni obyektler əlavə edilir. Kitabın əvvəlində qeyd etdiyimiz kimi, daha çox diqqəti o obyektlərə vermişik ki, onlar bütün Excel versiyalarında vacib rol oynayır. Oxucuya təqdim edilən VBA Excel kod nümunələri heç bir dəyişiklik olmadan Excel-in bütün versiyalarındakı VBA proqramlaşdırılmasında tətbiq edilə bilər (əlbəttə, Excel proqramına aid olan qeydlərimiz digər Office proqramlarına da aiddir).

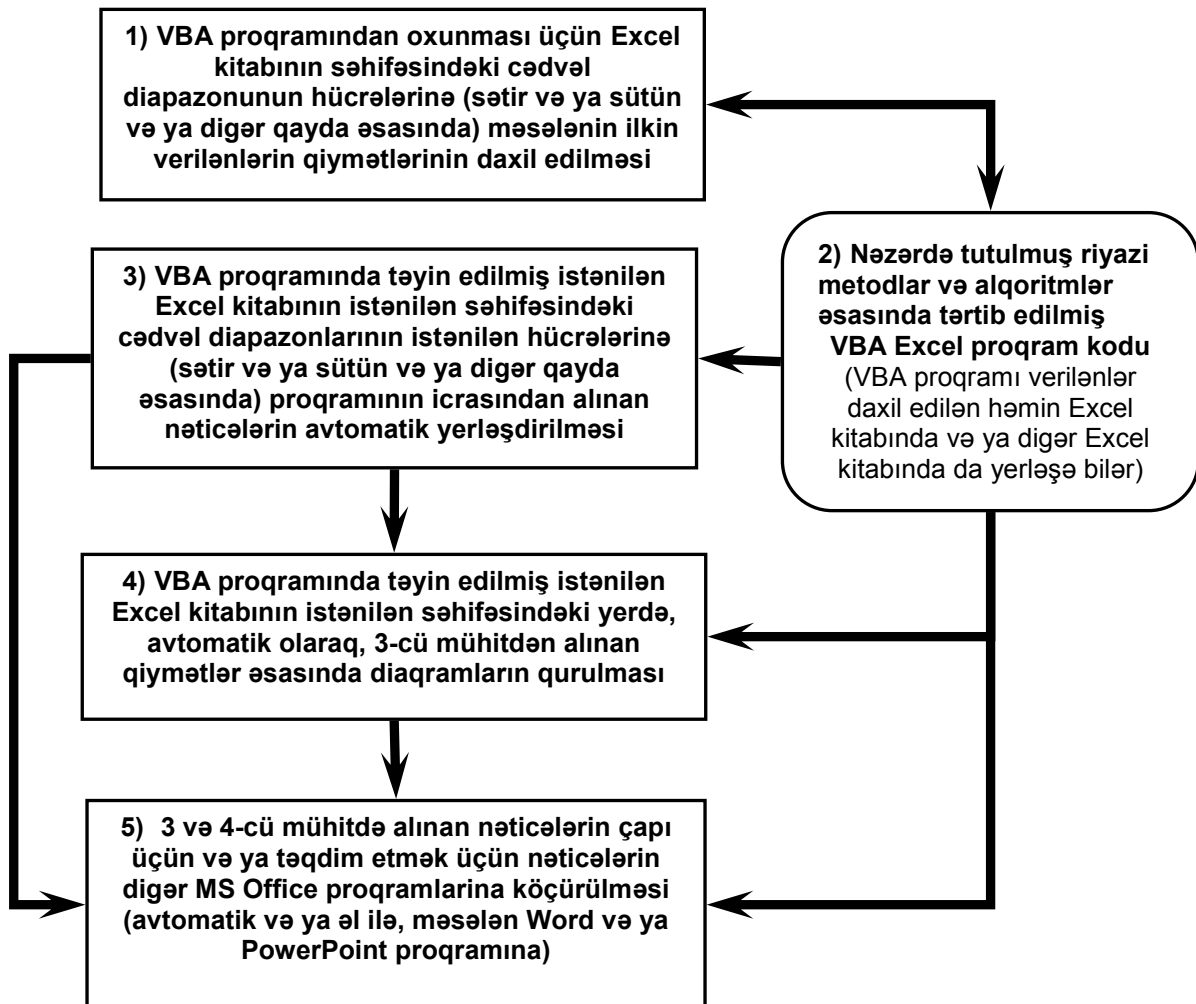
Bu bölmədə Excel-in başqa ən vacib olan obyektleri aşağıda nəzərdən keçirilir. Onlar haqqında daha müfəssəl məlumatı VBA Excel rəsmi sənədləşməsindən və ya makrorekorderdən almaq olar:

- **CellFormat** – obyektini ilə Excel hücrələrin formatlanması sazlanma bilər: şrift, rəng tərtibatı v.s. Bununla belə obyektini başqa məqsədlə də istifadə etmək olar: başqa hücrələrdəki formatlamanın kopyalanması, formatlanmaya görə axtarılma və əvəz edilmə v.s.
- **FormatCondition** – şərti formatlamanın sazlanmasını təmin edir;
- **ListObject** – siyahılarla işləmək üçün nəzərdə tutulub (adətən Excel kitabları ilə **SharePoint Portal Server** verilənlər bazasında işlədikdə istifadə edilir);
- **Validation** – bu obyekt istifadəçi tərəfindən daxil edilən verilənlərin yoxlanmasını təmin edir;
- **Watch** – obyektini düsturlar üçün kontrol qiymətlərin təyin edilməsində kömək edir (kontrol qiymətlərlə ekranın kənarında olan düsturların qiymətlərini izləmək olur – qrafik interfeysdə **Tools→Formula Auditing→Show Watch Window** menyusu ilə eyni güclüdür).

12.11 VBA Excel proqramlaşdırması və kompyuter riyaziyyatı

Məlumdur ki, VBA proqramlaşdırmasından xəbərsiz olan, Excel istifadəçilərinin təxminən 90% bu Office proqramının yalnız işçi səhifələri ilə (istisna hallarda diaqramlarla) işləyirlər. Səbəbi budur: Office istifadəçilərinin bir çoxu VBA haqqında bixəbərdirlər və ya ondan istifadə etməyi bacarmırlar. ABŞ-ın qabaqcıl universitetlərində VBA Excel proqramlaşdırması yüksək texnologiya yönli mühəndislik ixtisaslarında da tədris edilir: <http://www.faculty.virginia.edu/ribando/modules/xls>. Başqa bir tərəfdən riyazi məsələlərin həlli üçün xüsusi proqramlar təminatı yarandı ki, bu kateqoriyalı proqramlara kompyuter riyaziyyatı proqramları deyilər, ancaq onlar universitetlər üçün kütləvi tədrisdə əlçatan deyil (məsələn: Matlab, Maple v.s.). Ənənəvi riyazi məsələlərin həlli üçün istifadə edilən proqram təminatları (məsələn, Basic, Pascal və ya C++ müasir MS Office

proqramları ilə birgə işləmə imkanları güclü olmur. Digər tərəfdən, müasir, yuxarıda adlı çəkilən, xüsusiləşmiş kompyuter riyaziyyatı (və ya kompyuter cəbri proqram təminatları),. Həll əslində çox asandır: bütün kompyuterlərdə MS Office proqram paketi adətən yüklənmiş olur (həmçinin MS Excel VBA proqram təminatı da). Excel VBA proqramlaşdırması ilə ən çətin elmi-texniki, iqtisadi v.s. məsələlərin effektiv həlli mümkündür. Office proqramlaşdırılması sahəsində VBA-nın tətbiqi məhz Excel proqramında daha çox geniş yayılmışdır. Yalnız Excel-də daha çox xüsusi riyazi funksiyalar, proseduralar və xüsusi obyektlər mövcuddur. İstifadəçi üçün birçə şeyə ehtiyac vardır: Excel VBA dilini mükəmməl bilmək və tətbiqi məsələlərin həllində lazımınca istifadə etməyi bacarmaq. Alınan nəticələri (hesablamaları və diaqramları) isə çox asanlıqla, Office-in digər proqramlarına köçürərək yüksək məhsuldarlığa malik elmi işləri tərtib etmək mümkündür. Excel VBA proqramlaşdırılması texnologiyası şəx. 12.7-də verilib. Bu fəsildə təqdim edilən müxtəlif texniki, mühəndis və iqtisad yönü ixtisaslarda təhsil alan tələbələri üçün nəzərdə tutduq. Aşağıdakı bölmələrdə baxılan məsələlər çətinlik dərəcəsi ardıcılığı ilə verilmişdi. VBA proqramları Excel-in 2003 versiyasından başlayaraq, 2010 versiyasına qədər yoxlanılmışdır.



Şəkil 12.7 Excel VBA proqramlaşdırılması texnologiyasını və onun tətbiqini izah edən sxem.

12.11.1 Ən sadə riyazi hesablamalar strukturlu VBA Excel makrosu

Yuxarıda qeyd etdiyimiz kimi təqdim edilən məsələlər “asandan çətinə” doğru istiqamətdə təqdim edildiyinə görə bu bölmədə baxılan VBA proqram kodu ən sadə riyazi hesablamalar strukturuna malikdir. Çünki şəx. 12.7-də göstərilən Excel VBA proqramlaşdırılması texnologiyasını və onun tətbiqini sadə məsələlərin həllindən başlanılması bu sahədə yeni başlayanlar üçün, hesab edirik ki, çox vacibdir.

MƏSƏLƏ 1:

1. Excel kitabının birinci səhifəsində (adını `Sheet1`-dən dəyişdirib, məsələn, `Input` qoyun) `a`, `b`, `c`, `d`, `e`, `f`, `g`, `k`, `l` və `m` identifikatorlarını **Range** tipli verilənlər kimi qiymətlərini, məsələn, aşağıdakı cədvəldəki kimi səhifə diapazonuna (ixtiyari seçmək olar) daxil etmək lazımdır;

a	b	c	d	e	f	g	k	l	m
1	20	-31	60	45.76	32.81	0.765	800	9	10

2. Tərtib edilən VBA makrosunda `a`, `b`, `c`, `d`, `e`, `f`, `g`, `k`, `l` və `m` **Range** tipli verilənlər kimi səhifədən oxunmasını tərtib edin.
3. Makrosda aşağıda verilən düstur əsasında hesablamaları icra edən təlimat kodlarını tərtib edin (dəqiq təlimatlar yetirildikdən sonra oxucu bu düsturu istədiyi kimi dəyişdirə bilər, yalnız bir şərtlə ki, 0 ədədinə bölünmə alınmasın).

$$\text{rezultat} = (a+b+c) / d + \text{Sin}(3.14159 * (e+f) / k) + g * m + l$$

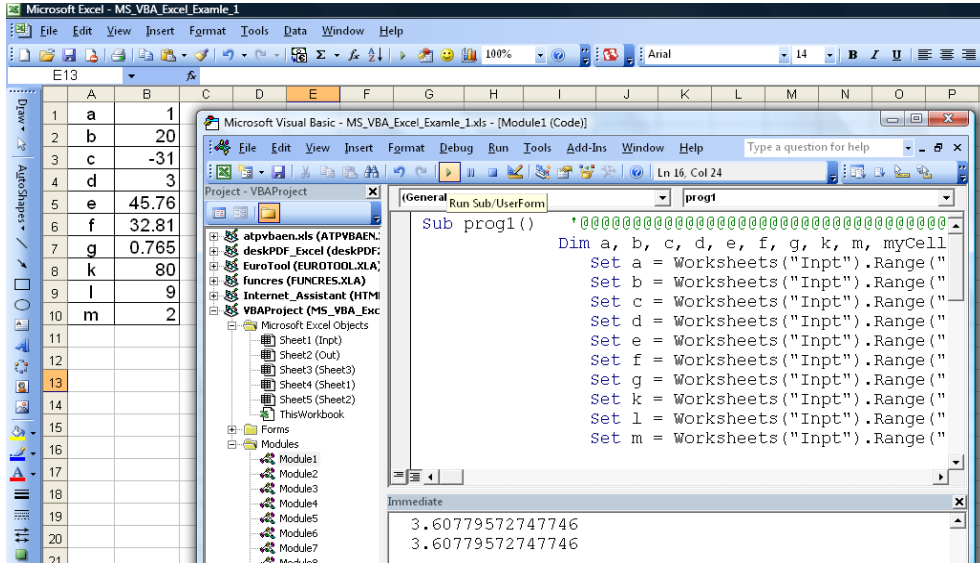
4. `rezultat` adlı dəyişənin qiymətindən asılı olaraq, iki hal üçün həmin Excel kitabının başqa səhifəsinə (məsələn, `Sheet2` səhifəsini dəyişdirib adın `Out` qoyun) hücrələrdə aşağıdakı şərtlərə görə əməlləri icra edən təlimat kodlarını yazın:

əgər rezultat > 5 onda

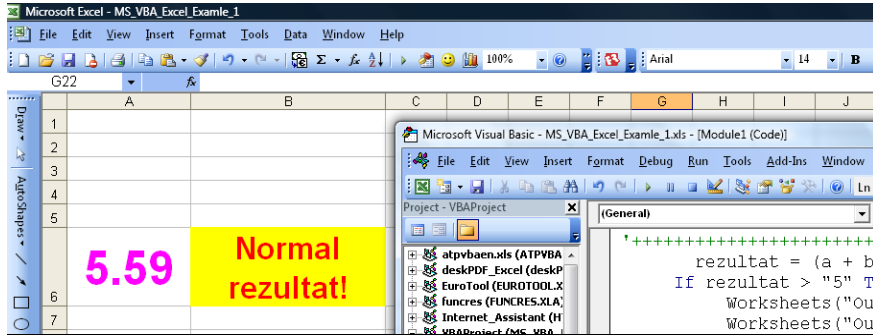
`Out` səhifəsinin B6 hücrəsinə “Normal rezultat!” yazılmalıdır və həmin hücrədə bu formatlama icra edilməlidir: hücrənin rəngi sarı rəngə boyanmalı, şriftin ölçüsü 24, rəngi isə qırmızı olmalıdır. `Result` adlı dəyişən `Out` səhifəsinin A6 hücrəsində əmələ gəlməlidir, əmələ gələn qiymətin şriftinin ölçüsü 36, rəngi narıncı olmalıdır **əgər rezultat < 5 onda**. `Out` səhifəsinin B6 hücrəsinə “Pis rezultat” yazılmalıdır və həmin hücrədə bu formatlama icra edilməlidir: hücrənin rəngi yaşıl rəngə boyanmalı, şriftin ölçüsü 20, rəngi qara və qalın olmalıdır. `Result` adlı dəyişən `Out` səhifəsinin A6 hücrəsində əmələ gəlməlidir, əmələ gələn qiymətin şriftinin ölçüsü 48, rəngi qara və qalın olmalıdır

MƏSƏLƏNİN HƏLLİ ÜÇÜN MƏSLƏHƏTLƏR VƏ KÖMƏK: Nəzərə alınmalıdır ki, oxucu əvvəlki hissələrdə verilən materiallarla tanış olub və lazımınca mənimsəyib. Məsələnin həllini icra edən tərtib edilmiş VBA kodunun mətnini aşağıda göstəririk. Həmin VBA kodu yığılıb və test edilib işləməsinə əmin olduqdan sonra, müxtəlif nəticələr almaq üçün Excel kitabının `Input`

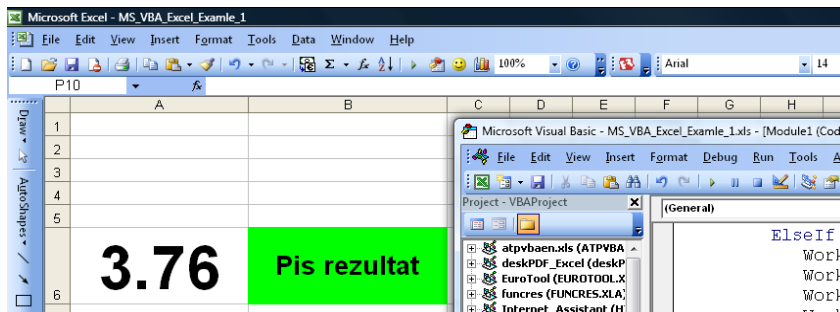
səhifəsindəki a, b, c, d, e, f, g, k, l və m identifikatorlarının qiymətlərini dəyişdirərək, həmin kodu icra etmək lazımdır. Oxucuya kömək üçün Excel-də alınan görüntüləri də burada göstərilməsini vacib sayırıq, şək. 12.8, şək. 12.9 və şək. 12.10.



Şəkil 12.8 Excel kitabının Input səhifəsində a, b, c, d, e, f, g, k, l və m identifikatorlarının daxil edilməsi (identifikatorlar sütun qaydası ilə daxil edilsələr də VBA kodundakı set operatoru ilə istənilən qaydada və diapazonların hücrələrində daxil edilən verilənlərin oxunması mümkündür).



Şəkil 12.9 birinci şertə görə rezultat adlı dəyişənin, hesablanmış qiymətindən asılı olaraq, həmin Excel kitabının Out səhifəsində alınan nəticələr.



Şəkil 12.10 İkinci şertə görə redutla adlı dəyişənin, hesablanmış qiymətindən asılı olaraq, həmin Excel kitabının Otu səhifəsində alınan nəticələr.

Heç bir xüsusi şərhlər və izahat verilmədən (çünki məsələ çox sadədir), məsələnin həlli üçün Excel VBA kodunun tam mətni aşağıda verilib (nəzərə alınıb ki, oxucu artıq VBA proqramlaşdırması ilə tanışdır).

```

Sub progl ()
'*****
  Dim a, b, c, d, e, f, g, k, m, myCell As Range
  Set a=Worksheets("Input").Range("b1")
  Set b=Worksheets("Input").Range("b2")
  Set c=Worksheets("Input").Range("b3")
  Set d=Worksheets("Input").Range("b4")
  Set e=Worksheets("Input").Range("b5")
  Set f=Worksheets("Input").Range("b6")
  Set g=Worksheets("Input").Range("b7")
  Set k=Worksheets("Input").Range("b8")
  Set l=Worksheets("Input").Range("b9")
  Set m=Worksheets("Input").Range("b10")
  Dim rezultat As Variant
'*****
  rezultat=(a+b+c)/d+Sin(3.14159*(e+f)/k)+g*m+l
'*****
  If rezultat > "5" Then
    Worksheets("Out").Range("B6").Value="Normal rezultat!"
    Worksheets("Out").Range("B6").Interior.ColorIndex=6
    Worksheets("Out").Range("B6").Font.Size=24
    Worksheets("Out").Range("B6").Font.ColorIndex=3
    Worksheets("Out").Range("A6").Font.Size=36
    Worksheets("Out").Range("A6").Font.ColorIndex=7
  ElseIf rezultat < "5" Then
    Worksheets("Out").Range("B6").Value="Pis rezultat"
    Worksheets("Out").Range("B6").Interior.ColorIndex=4
    Worksheets("Out").Range("B6").Font.ColorIndex=1
    Worksheets("Out").Range("B6").Font.Bold=True
    Worksheets("Out").Range("B6").Font.Size=20
    Worksheets("Out").Range("A6").Font.ColorIndex=1
    Worksheets("Out").Range("A6").Font.Size=48
    Worksheets("Out").Range("A6").Font.Bold=True
  End If
'*****
Debug.Print rezultat
  Worksheets("Out").Range("A6").Value=rezultat
  Exit Sub
  Return
End Sub

```

Yuxarıdakı VBA kodun normal halda (müvəffəqiyyətlə) icrasına nail olduqdan sonra həmin VBA kodunun əsasında bir qədər sınaqlar aparılması (kodun dəyişdirilməsi və yenidən icra edilməsi) oxucunun Excel VBA proqramlaşdırılması vərdişlərinin daha yaxşı mənimsənilməsinə və növbəti addımda bir qədər də çətin məsələlərin həlli ilə məşğul olmaq üçün yaxşı köməklik edə bilər.

12.11.2 VBA Excel proqramlaşdırması ilə tərs matrisin hesablanması

Bu bölmədə baxılan məsələ əvvəlkindən bir qədər çətin olsa da, hər halda, hesab edirik ki, xətti cəbri tənliklər sisteminin tərs matris ilə həlli səviyyəsində oxucunun lazımı qədər nəzəri hazırlığı vardır. Lakin bizim məqsədimiz başqadır: VBA Excel proqramlaşdırılmasını bəzi riyazi məsələlərin həllində istifadə edilməsini oxucuya mənimsətmək. Əgər oxucu bu sahədə daha mükəmməl olan proqram və prosedura axtarsa, onda <http://www.netlib.org> saytına müraciət edə bilər. Qeyd edirik ki, orijinal proqram Virjiniya Universitetinin (ABŞ) VBA proqramlaşdırılmasının tədrisdə tətbiqinə aid saytıdan götürülüb: www.faculty.virginia.edu/ribando/modules

MƏSƏLƏ 2: VBA Excel proqramlaşdırılması ilə verilmiş kvadrat matrisin tərs matrisini Gauss-Jordan üsulu ilə hesablayın:

$$A = \begin{pmatrix} 3 & 2 & 4 \\ 2 & 5 & 3 \\ 7 & 2 & 2 \end{pmatrix}$$

MƏSƏLƏNİN HƏLLİ ÜÇÜN MƏSLƏHƏTLƏR VƏ KÖMƏK:

Yuxarıda qeyd etdiyimiz kimi, riyaziyyatla bağlı nəzəri məlumatlar bu kitabın çərçivəsindən kənarıdır və buna görə maraqlanan oxucu riyaziyyatın bu bölməsi ilə bağlı xüsusi ədəbiyyatdan istifadə edə bilər. Buna baxmayaraq, məsələni VBA-da proqramlaşdırmaq üçün tərs matrisin Gauss-Jordan metodu əsasında hesablanma alqoritmini aşağıda göstəririk.

Gauss-Jordan üsulu: A matrisini E vahid matris formasına Gauss-Jordan üsulu ilə gətirmək lazımdır. Birinci matrisə hər dəfə eyni əməliyyatı tətbiq etdikdə həmin əməliyyatı ikinci matrisə də tətbiq etmək lazımdır. Birinci matrisin vahid matrisə gətirilməsi qurtaran kimi ikinci matris əslində A matrisinin tərs matrisi A^{-1} olacaq. Alqoritmin riyazi yazısı aşağıda verilib:

$$\Lambda_1 \cdot \dots \cdot \Lambda_i \cdot \dots \cdot \Lambda_n \cdot A = \Lambda A = E \quad \Rightarrow \quad \Lambda = A^{-1}$$

Gauss metodu istifadə edildikdə birinci matris soldan Λ_i elementar matrisinin birinə vurulur (Λ_i - baş diaqonalda vahidlər, digərlərində sıfır və yalnız bir pozisiyada sıfır qiymətlər olmayan diaqonal matrisdir). Beləliklə, bütün əməliyyatlar tətbiq edilib qurtardıqdan sonra ikinci matris Λ , əslində axtardığımız, A^{-1} tərs matrisinə bərabər olacaq. Yuxarıdakı düsturda Λ_i matrisi bu cür hesablanmalıdır:

$$\Lambda_i = \begin{bmatrix} 1 & \dots & 0 & -a_{1i}/a_{ii} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 1 & -a_{i-1i}/a_{ii} & 0 & \dots & 0 \\ 0 & \dots & 0 & 1/a_{ii} & 0 & \dots & 0 \\ 1 & \dots & 0 & -a_{i+1i}/a_{ii} & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & -a_{ni}/a_{ii} & 0 & \dots & 1 \end{bmatrix}$$

MƏSƏLƏNİN VBA Excel MÜHİTİNDƏ HƏLLİ.

Tərtib olunası VBA proqramında aşağıdakı əməllər yerinə yetirilməlidir:

1. Excel səhifəsində A matrisindəki elementlərin qiymətləri müəyyən təyin edilmiş diapazonlara daxil edilməlidir;
2. həmin daxil edilmiş qiymətlərin oxunmasını təmin edən VBA proqram kodu yazılmalıdır;
3. Qauss-Jordan üsulu əsasında tərs matris A^{-1} hesablayan VBA prosedurası tərtib edilməlidir;
4. alınmış həll matrisini Excel səhifəsinə çıxaran VBA kod sətirləri proqrama əlavə edilməlidir;
5. proqramı testdən keçirərək, onun düzgün işləməsinə əmin olmaq lazımdır.

Məsələnin həllini VBA-da icra edən proqram kodunu aşağıda veririk (Excel-də alınan nəticə aşağıdakı şəkl. 12.11-də göstərilib):

Option Explicit

Dim mat() As Double

Dim matinv() As Double

Sub invert(s As Integer)

'Tərs matrisin təyin edilməsi üçün Qauss metodu istifadə edilir.

'matinv alt proqramı tərs matrisi qaytarır. mat və matinv global

'dəyişənlərdir. s argumenti matrisin ölçüsüdür.

Dim i As Integer, j As Integer, p As Integer, k As Integer

Dim c As Double, d As Double

ReDim Preserve mat(1 To s, 1 To s), matinv(1 To s, 1 To s)

'Qauss metodunun tələbi: matinv matrisinin vahid matris kimi təyin edilir.

For i=1 To s

For j=1 To s

If i=j Then

matinv(i,j)=1

Else

matinv(i,j)=0

End If

Next

Next

'Qaus metodunun icrası başlayır. Çıxmaq əməliyyatı ilə

'diaqonql altında sıfır qiymətli sütunlar yaradılır.

```

For p=1 To s-1
  c=mat(p,p)
  If c=0 Then
'İlk sıfır olmayan sətəri tapan flip_non_zero alt proqramı çağırır
  flip_none_zero s, p
  End If
  c=mat(p,p)
  If none_all_zero(s, p) Then
    For i=p+1 To s
      d=mat(i,p)/c
      For k=1 To s
        'Matrisdə çıxma əməliyyatı yerinə yetirilir
        mat(i,k)=mat(i,k)-d*mat(p,k)
        'Həmin çıxmanı tərs matrisdə icra edir
        matinv(i,k)=matinv(i,k)-d*matinv(p,k)
      Next
    Next
  End If
Next
'Bu addımda matris artıq üçbucaqlıdır və diaqonaldan yuxarı hissədə
'sıfır qiymətli sütunlar yaradılır.
For p=s To 2 Step -1
  c=mat(p,p)
  For i=p-1 To 1 Step -1
    d=mat(i,p)/c
    For k=1 To s
      mat(i,k)=mat(i,k)-d*mat(p,k)
      matinv(i,k)=matinv(i,k)-d*matinv(p,k)
    Next
  Next
Next
'İndi matris artıq diaqonal matrisə çevrildi və diaqonaldakı
'ədədlər normallaşır (yeni 1 bərabər edilir)
For k=1 To s
  c=mat(k,k)
  If c<>1 Then
    For i=1 To s
      mat(k,i)=mat(k,i)/c
      matinv(k,i)=matinv(k,i)/c
    Next
  End If
Next
End Sub
Sub flip_none_zero(s As Integer, p As Integer)
'Bu prosedura ələ xətti axtarır ki, onun p sütununda sıfırdan
'fərqli əmsali var. Əgər xətt tapıldısa, onda bu xətti p xətti ilə
'çevirmək lazımdır. Gauss metodunun tələbi ilə həmin əməl tərs
'matrislə də aparılır.
Dim i As Integer, k As Integer
Dim a As Double
i=p
While (mat(i,p)=0) And (i<s)
  i=i+1
Wend
If i<s Or mat(s,p<>0) Then
  For k=p To s
    a=mat(i,k)
    mat(i,k)=mat(p,k)
    mat(p,k)=a
    a=matinv(i,k)
  Next
End If

```

```

        matinv(i,k)=matinv(p,k)
        matinv(p,k)=a
    Next
End If
End Sub
Function none_all_zero(s As Integer, p As Integer) As Boolean
'Doğrudur, eğer en azı bir satırın p sütununda sıfırdan farklı
'amsalları varsa.
Dim i As Integer
none_all_zero=False
For i=p To s
    If mat(i,p)<>0 Then
        none_all_zero=True
    End If
Next
End Function
'Aşağıdaki display_matrix(s As Integer) adlı alt program Excel
'sahifesine tayin edilmiş matrisi yazır
Sub display_matrix(s As Integer)
Dim i As Integer, j As Integer
For i=1 To s
    For j=1 To s
        'Değişmiş satır ve soldaki sütun matrisi:
        ActiveSheet.Cells(i+20,j+1).Value=matinv(i,j)
    Next
Next
End Sub
'Aşağıdaki read_matrix() adlı alt program Excel sayfasındaki A
'matrisinin elementlerini okumak için tertib edilib
Sub read_matrix()
Dim i As Integer, j As Integer
Dim s As Integer
s=ActiveSheet.Cells(20, 1).Value
ReDim Preserve mat(1 To s,1 To s), matinv(1 To s,1 To s)
For i=1 To s
    For j=1 To s
        'Değiştirilmiş satırlar ve sağ taraftaki sütun matrisi:
        mat(i,j)=ActiveSheet.Cells(i+2,j+1).Value
    Next
Next
Next
display_matrix s
invert s
display_matrix s
End Sub

```

$[A]^{-1} =$	-0.05128205	-0.0513	0.17949							
	-0.21794872	0.28205	0.01282							
	0.3974359	-0.1026	-0.141							
NOTE: for large systems you should not invert the matrix!										
See: Netlib Repository at ORNL instead										

Şekil 12.11 Excel sayfasında hollin (yeni tayin edilmiş ters matrisin) gösterilmesi (sol taraftaki "Invert Using VBA" düğmesi ile bir başa hemin sayfeden, tertib edilen VBA programı icra edilmek için çağırılabilir). Ortadaki qeydde yazılıb: "Böyük sistemler için ters matrisi tapma

bilməyəcəksiniz!” – bu isə, yuxarıda qeyd etdiyimiz kimi, nəzəriyyə ilə bağlıdır (əslində baxılan üsul maksimum 20 və ya 30 tərtibli kvadrat matris üçün tətbiq edilə bilər).

12.11.3 MS Excel-də VBA proqramlaşdırması ilə adi diferensial tənliklərə ifadə edilən bəzi sistemlərin modelləşdirilməsi

Bu bölmədə baxılan məsələ əvvəlkindən bir qədər də çətinləşir: adi diferensial tənliklər (ADT) ilə ifadə edilən sistemlərinin ədədi həllini VBA Excel mühitində reallaşdırmaq [5]. Əvvəlcədən qeyd etməliyik ki, VBA Excel mühiti bu cür məsələlərin ədədi həlli üçün çox əlverişli imkanlar yaradır, sadəcə şəkl. 12.7 baxmaq kifayətdir. Yuxarıda qeyd etdiyimiz kimi, ABŞ-nin Virjiniya Universitetində (www.faculty.virginia.edu/ribando/modules), hətta aerodinamikaya aid olan qeyri xətti xüsusi törəməli diferensial tənliklərlə ifadə edilən texniki sistemlərinə VBA köməyi ilə Excel-də ədədi üsullar ilə modelləşdirirlər. Xüsusilə, nəzərə alsaq ki, MS Excel-də 2D və 3D elmi qrafiklərin qurulması üçün güclü imkanlar var - VBA proqramlaşdırması bəzi məqamlarda məşhur kompyuter riyaziyyatı proqram təminatları ilə rəqabət edə bilər (təsadüf deyil ki, həmin proqramların hamısında Excel əlaqələnməsi həmçinin mövcuddur).

MƏSƏLƏ 3: Məşhur Volterra-Lotka ekoloji sisteminin ədədi üsul ilə modelləşdirilməsini VBA Excel proqramlaşdırılması ilə həyata keçirmək lazımdır (dərs vəsaitinin müəllifi kimi qeyd etməliyəm ki, bu məsələnin VBA Excel mühitində həllini 1997 ildə almışdım, nəticələr isə məqalədə nəşr edilib, bax [5]).

Baxılması məsələnin VBA Excel-də reallaşdırılması və ümumiyyətlə, işlərin aparılması ardıcılığı eyni qaydada gedir (bax şəkl. 12.7):

1. Excel səhifəsində məsələnin ədədi qiymətlərini müəyyən edilmiş diapazonun hücrələrinə daxil etmək;
2. adi diferensial tənliklərin ədədi həlli üçün mövcud olan metodların alqoritmi əsasında (bizim baxdığımız məsələlərin həllində standart 4 tərtibli Runge-Kutta metodunun alqoritmi istifadə edilib) VBA proqram kodunu tərtib etmək;
3. tərtib edilmiş VBA kodunu sınaqdan keçirib (test mərhələsi) normal işləməsinə əmin olmaq;
4. alınan nəticələrin həmin Excel kitabının təyin edilmiş səhifələrinin birində təyin edilmiş diapazonlara çıxarılması üçün həmin (artıq 3-cü mərhələdə testdən keçmiş) proqram koduna müvafiq VBA kod sətirlərini əlavə etmək;
5. alınmış ədədi nəticələrin daha əlverişli analizi üçün proqram kodunda və ya Excel-in qrafik interfeysinin köməyi ilə uyğun olan qrafiklərin (diqramların) avtomatlaşdırılmış halda yaradılmasını təmin etmək.

Bu kitabın çərçivəsinə daxil deyil deyə, məşhur Runge-Kutta üsulunun alqoritminin riyazi yazılışını gətirmirik (maraqlanan oxucular bunun üçün tətbiqi riyaziyyata aid xüsusi dərsliklərdən, dərs vəsaitləri və sorğu kitablarından istifadə edə bilərlər - oxucunun bu cür əməl etməyini məsləhət görürük).

Məsələnin VBA Excel mühitində həlli

Kitabın müəllifi S.T.Hüseynov həmin məsələni 1998-ci ildə Excel VBA proqramlaşdırması mühitində həll etmişdir. Bu VBA Excel proqramını müəllif bir məqsədlə yaratmışdır: müxtəlif elm və texnika mütəxəssisləri, xətti və ya qeyri xətti ADT sistemləri ilə ifadə edilən tətbiqi məsələlərin ədədi üsul əsasında həllini sadə və əlverişli mühitdə məhz Microsoft Excel Visual Basic for Application proqramlaşdırması ilə reallaşdırma bilsinlər. Tərtib edilmiş proqram tam testdən müvəffəqiyyətlə keçib və Excel-in bütün versiyalarında (MS Excel 97-dən başlayaraq daha MS Excel 2010 versiyasına qədər) yaxşı nəticələr alınmışdır. Oxucunun nəzərinə gətirilən qaydalara riayət olunsa, proqram ilə müvəffəqiyyətli nəticələrin alınması normal hal sayılmalıdır. Proqramların tətbiq edilməsi nəticəsində ekologiyada və biologiyada (Volterra-Lotka məsələsi), hərbi elmdə (Lançester modeli), müxtəlif texniki-mühəndis elmlərində və iqtisadiyyatda fəaliyyət göstərən tədqiqatçılar uyğun olan modellərinin ədədi həllini tapa bilərlər. Hesablama proqramı universitet tədrisində də tətbiq edilə bilər: ali riyaziyyatın ədədi metodları, riyazi və kompyuter modelləşdirməsi, İnformatika kursunun özündə v.s.

Excel-də VBA ilə yazılmış `RNGEC` adlı hesablama proqramı IV tərtibli məşhur Runge-Kutta metodunun alqoritmi əsasında tərtib edilib. Həmin standart alqoritm başlanğıc şərtləri olan (Koşi məsələsi) normallaşmış (yəni 1-ci tərtibli ADT sistemlərinə gətirilmiş) ADT sisteminin ədədi həlli üçün tətbiq edilir. Tətbiq edilən ədədi alqoritm əsas üstünlükləri bunlardır: ədədi həllin yaxşı səviyyədə olan dayanıqlığı və kifayət dərəcədə olan dəqiqliyi. Tərtib edilmiş proqramın isə üstünlükləri aşağıda göstərilib:

1. avtomatlaşdırılmış qaydada məsələnin ilkin ədədi qiymətlərini Excel kitabının istifadəçi tərəfindən təyin edilmiş səhifəsindən oxunması təmin edilir;
2. avtomatlaşdırılmış qaydada alınan ədədi həlli həmin Excel kitabının istifadəçi tərəfindən təyin edilmiş səhifəsinə çap edilir;
3. alınmış nəticələrin əlverişli analiz edilməsi üçün zamandan asılı qrafiklər və hərəkətin faza portretləri də həmin Excel kitabının istifadəçi tərəfindən təyin edilmiş səhifəsində qurulur.

PROQRAMLA MÜSTƏQİL İŞLƏMƏK ÜÇÜN MƏSLƏHƏTLƏR VƏ KÖMƏK:

1) Məsələnin ilkin ədədi qiymətlərinin, istifadəçi tərəfindən təyin edilmiş, Excel səhifəsindəki diapazon hücrələrindən oxunması - VBA kod sətirlərinin tərtibi.

Birinci növbədə məsələnin ilkin qiymətlərini Excel səhifəsinə daxil etmək lazımdır. Bunu belə etmək lazımdır ki, həmin diapazondakı hücrələrdən proqram sonra həmin ədədi qiymətləri oxuya bilsin. Bunun üçün proqrama, uyğun olaraq, Excel-də `Inpt` adlı səhifə yaradılmalı (və ya mövcud olan `Sheet1` adlı səhifənin adı `Inpt` adı ilə dəyişdirilməlidir). Həmin səhifənin `B3:B13` diapazondakı hücrələrinə ədədi qiymətlər daxil edilməlidir. Yaxşı olar ki, şəkl. 12.12-də göstərilən kimi, `A3:A13` diapazonundakı uyğun olan hücrələrində isə (mətn formatında) ədədi qiymətlərin adları, mümkündürsə, hətta izahat edici şərhlər də verilsin – bununla proqramın (və məsələnin özünün) sonradan, müəyyən vaxt keçdikdən sonra daha asan anlanması mümkün olacaq.

	A	B	C	D	E	F
1	Volterra - Lotka ekoloji modeli (Yirtıcı - Qurban modeli)					
2	Problem № 1			Problem № 1		
3	n	2		n	2	
4	rm	200		rm	200	
5	t0	0		t0	0	
6	t1	2000		t1	2000	
7	y10	350		y10	350	
8	y20	35		y20	35	
9	a	0.041		a	0.041	
10	b	-0.00005		b	-5E-05	
11	c	-0.0002		c	-2E-04	
12	k	-0.041		k	-0.041	
13	L	0.0003		L	0.0003	

Şəkil 12.12 Excel kitabının Inpt səhifəsində B3:B13 diapazonunda yerləşən hücrələrə məsələnin ilkin qiymətlərinin daxil edilməsi.

Məsələnin strukturuna uyğun aşağıdakı işarələmələr qəbul edilib:

n – adi diferensial tənliklərin (ADT-lərin) sayı, B3 hücrəsinə daxil edilir;

rm – inteqrallama addımlarının sayıdır, B4 hücrəsinə daxil edilir;

t_0 – zamanın başlanğıc qiyməti, B5 hücrəsində 0 qiyməti daxil edilir;

t_1 – ədədi həllin zaman diapazonudur, B6 hücrəsinə daxil edilir;

y_{10} və y_{20} – birinci və ikinci axtarılan $y_1(0)$ və $y_2(0)$ funksiyaların başlanğıc qiymətləridir (məsələn, şəkl. 12.12-də Volterra-Lotka modeli üçün $y_1(t)$ – qurbanlar populyasiyasıdır, $y_2(t)$ – yırtıcılar populyasiyasıdır (hərbi elmdə, Lançester modeli üçün, məsələn, qırmızı ordunun canlı qüvvəsi və yaşıl ordunun canlı qüvvəsinin sayı ola bilər) uyğun olaraq, B7 və B8 hücrələrinə daxil edilir;

a, b, c, k və L – məsələnin sabit əmsallarıdır, uyğun olaraq B9, ..., B13 hücrələrinə daxil edilir.

Excel VBA mühitində həmin verilənlərin oxunmasını təmin etmək üçün tərtib edilmiş VBA kodunu tətbiq etmək olar (aşağıdakı fraqmentinə oxucu diqqətlə baxsa hər şey ona aydın olacaq):

```
'** Məsələnin ilkin verilənlərinin Inpt səhifəsindən oxunması **'
```

```
Dim n, rm, t0, t1, y10, y20, a, b, c, k, L As Range
```

```
Set n=Worksheets("Inpt").Range("b3")
```

```
Set rm=Worksheets("Inpt").Range("b4")
```

```
Set t0=Worksheets("Inpt").Range("b5")
```

```
Set t1=Worksheets("Inpt").Range("b6")
```

```
Set y10=Worksheets("Inpt").Range("b7")
```

```
Set y20=Worksheets("Inpt").Range("b8")
```

```
Set a=Worksheets("Inpt").Range("b9")
```

```
Set b=Worksheets("Inpt").Range("b10")
```

```
Set c=Worksheets("Inpt").Range("b11")
```

```
Set k=Worksheets("Inpt").Range("b12")
```

```
Set L=Worksheets("Inpt").Range("b13")
```

2) **Proqramın əsas hissəsi: Runge-Kutta metodunun alqoritmi əsasında VBA proqram kodunun tərtib edilməsi.**

Qeyd edilməlidir ki, istifadəçi istədikdə (məsələn, həmin ədədi üsul hansısa səbəbə görə onu təmin etmir - sərt tipli məsələnin həllinə ədədi üsul uyğun gəlmir) həmin əsas hissəni başqa daha uyğun olan alqoritmə əsasında yeni VBA kodunu tərtib edə bilər.

```
'#####
'## Proqramın əsas hissəsi: Runge-Kutta üsulu ilə ADT-nin
'##### ədədi həllinin icrası
'burada y(1) və y(2) axtarılan funksiyaaların ədədi qiym.-dir
Dim i, r As Integer
Dim t, z(2), f(2,300), y(2,200), kft(4,2), P,Q As Variant
    t=t0
    y(1,1)=y10
    y(2,1)=y20
    h=t1/rm
For i=1 To n
    z(i)=y(i,1)
Next i
    For r=1 To rm
    GoSub Funcs
    For i=1 To n
    kft(1,i)=h*f(i,r)
    y(i,r)=z(i)+0.5*kft(1,i)
    Next i
    t=t+0.5*h
    GoSub Funcs
    For i=1 To n
    kft(2,i)=h*f(i,r)
    y(i,r)=z(i)+0.5*kft(2,i)
    Next i
    t=t+0.5*h
    GoSub Funcs
    For i=1 To n
    kft(3,i)=h*f(i,r)
    y(i,r)=z(i)+kft(3,i)
    Next i
    GoSub Funcs
    For i=1 To n
    kft(4,i)=h*f(i,r)
    y(i,r)=z(i)+(1/6)*(kft(1,i)+2*_
    (kft(2,i)+kft(3,i))+kft(4,i))
    z(i)=y(i,r)
    Next i
```

Proqramdakı `Funcs` adlı istifadəçi funksiyası araşdırılan riyazi modelin ADT-lərini formalaşdırır (bax tərtib edilmiş VBA kodunun aşağıdakı fraqmentinə). Proqramı tam sınaqdan keçirdikdən sonra, həmin fraqmentdə oxucu lazım olan uyğun dəyişiklikləri əlavə edə bilər.

```
'Aşağıdakı funksiya modelin ADT-lərini formalaşdırır
Funcs:
P=0
Q=0
f(1,r)=a*y(1,r)+b*y(1,r)*y(1,r)+c*y(1,r)*y(2,r)+P
f(2,r)=k*y(2,r)+L*y(1,r)*y(2,r)+Q
'*****
Return
Flgl:
End Sub
```

Əslində həmin fraqmentə uyğun olan riyazi model aşağıdakı ADT sistemi ilə ifadə edilir (məşhur Volterra-Lotka ekoloji modeli):

$$\begin{cases} \frac{dy_1(t)}{dt} = ay_1(t) - by_1(t)y_2(t) + P(t), \\ \frac{dy_2(t)}{dt} = -by_2(t) + ky_1(t)y_2(t) + Q(t). \end{cases}$$

$$y_1(0) = y_{10}, \quad y_2(0) = y_{20}.$$

burada $y_1(t)$ və $y_2(t)$ - axtarılan funksiyalardır; $y_1(0)$ və $y_2(0)$ - başlanğıc şərtlərdir; t - zamandır; a , b və k - sistemin daxili strukturu ilə bağlı sabit əmsallardır; $P(t)$ və $Q(t)$ - xaricdən sistmə təsir edən zamandan asılı funksiyalardır.

3) Alınmış ədədi həllin Excel-in təyin edilmiş səhifəsinin təyin edilmiş diapazonunun hücrələrinə çıxarılması (uyğun olan VBA kod sətirlərinin tərtib edilməsi və proqrama əlavə edilməsi).

Nəticələri Excel səhifələrindən birinə çıxarmaq üçün bunları etmək lazımdır: həmin Excel kitabının, məsələn, **Sheet2** adlı səhifəsinin adını proqramdakı koda uyğun Output qoymaq lazımdır. Hesablamanın nəticəsi ədədi qiymətlər formasında həmin Excel kitabının "Output" adlı səhifəsinə çıxarılır, bax şəx: uyğun olaraq, t , $y_1(t)$ və $y_2(t)$ həllinin (identifikatorların) **A:A**, **B:B** və **C:C** sütunlarına çap edilməsi.

	A	B	C	D	E	F
1	10	395.2219	71.50438			
2	20	399.076	136.9537			
3	30	351.6424	248.3695			
4	40	259.7327	376.4719			
5	50	166.4101	452.0965			
6	60	103.0987	448.2643			
7	70	68.51449	394.2315			
8	80	51.25564	325.0703			
9	90	43.30462	259.6527			
10	100	40.6415	204.5939			
11	110	41.47909	160.7938			

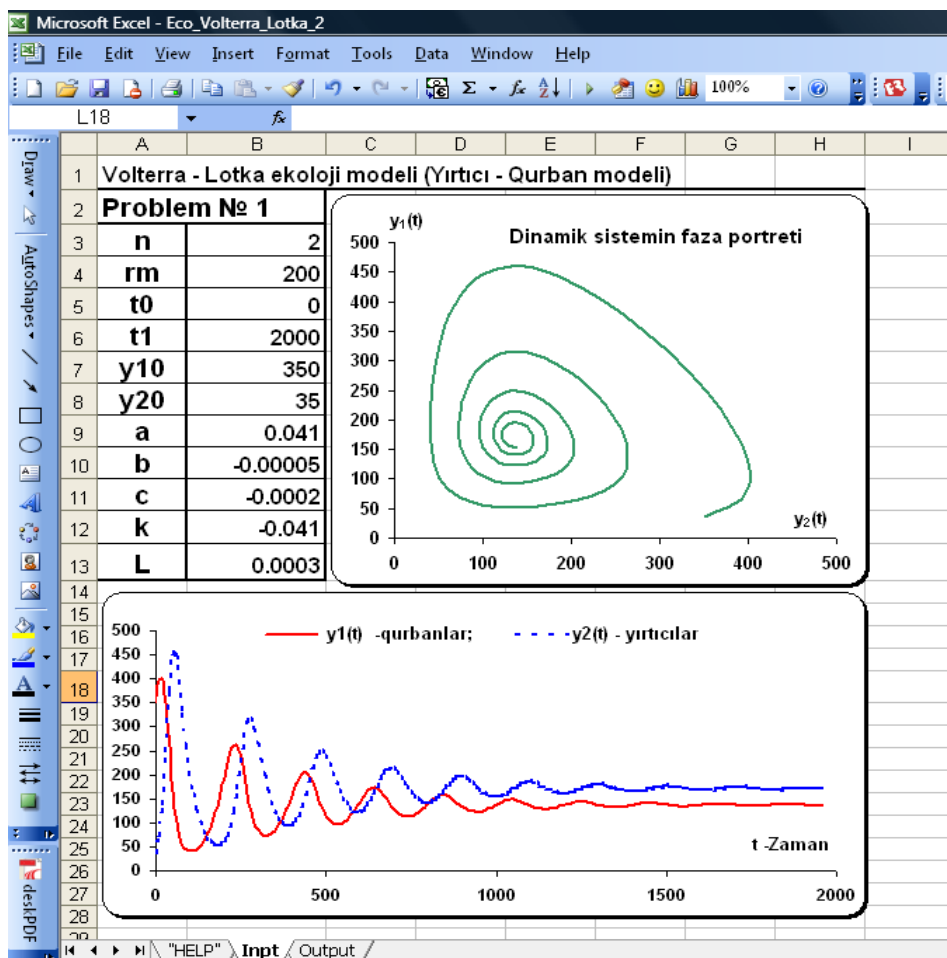
Şəkil 12.13 Alınmış nəticələrin ədədi qiymətlər formasında həmin Excel kitabının **Output** adlı səhifəsinə çıxarılması.

Hesablama nəticələrinin necə VBA Excel mühitində çapa çıxarılmasını oxucu tərtib edilmiş proqram kodunun aşağıdakı fraqmentindən yaxşı anlaya bilər:

```
'Nəticələrin Output səhifəsinə çıxarılması
Set curcell=Worksheets("Output").Cells(r, 1)
curcell.Value=t
Set curcell Worksheets("Output").Cells(r, 2)
curcell.Value=y(1, r)
Set curcell=Worksheets("Output").Cells(r, 3)
curcell.Value=y(2, r)
```

4) Alınmış ədədi həllin əsasında Excel-in təyin edilmiş səhifəsində qrafiklərin qurulması

Beləliklə, alınmış nəticələri iki formada istifadəçinin nəzərinə təbiiq etmək olar: birincisi – yuxarıda izahı verilmiş, ədədi qiymətlərin həmin Excel kitabının təyin edilmiş səhifəsinin (bizim halda **Output** adlı səhifəsinin) təyin edilmiş diapazonlarına (bizim halda **A:A**, **B:B** və **C:C** sütunlarına) çap edilməsi; ikincisi – həmin çap edilmiş nəticələrin əsasında qrafiklərin qurulması. Qeyd etməliyik ki, bütün dünyada araşdırma və analiz aparıldıqda tədqiqatçılar rəqəmlərlə dolu sütunlara baxmaq əvəzinə üstünlüyü daha çox qrafik formada alınan nəticələrə verirlər. Bu isə təəccüblü deyil – qrafik formada olan vizual görüntülər insana daha aydın və əlçatan vasitədir. Təqdim edilən proqramda qrafiklərin (diaqramların) alınması üçün iki üsuldən istifadə etmək olar, və bu üsulların hər ikisi eynigüclüdür. Birinci üsul – VBA Excel proqramlaşdırması ilə (kitabın 12.9 hissəsindəki **Chart** obyektini ilə diaqramlarla işləmə üsulları) lazımı qrafiklərin qurulması. İkinci üsul – Excel-in qrafik interfeysinin **Insert**→**Picture**→**Chart** menyusu ilə **Chart** obyektinin yaradılması və **Output** səhifəsindəki **A:A**, **B:B** və **C:C** sütunlarındakı qiymətlərlə bağlanması. Proqramın müəllifi ikinci üsuldən istifadə edib, çünki ikinci üsul daha rahat və əlverişli mühit yaratmışdır. Nəzərə alınmalıdır ki, praktikada belə hallara da rast gəlmək olur: yalnız birinci üsulun tətbiq edilməsi mümkündür (yəni VBA proqramlaşdırılması ilə **Chart** obyektinin yaradılması və dəyişənlərlə operativ yaddaşda bağlanması). Analiz üçün operativ müdaxilə məqsədi ilə proqramın müəllifi yaradılmış qrafikləri məsələnin ilkin verilənlərinin daxil edildiyi **Inpt** adlı səhifəyə yerləşdirib (yada salırıq ki, oradan VBA proqram kodu məsələnin ilkin verilənlərini oxuyur), bax şəx. 12.14.



Şəkil 12.14 Alınmış nəticələrin əsasında **Chart** obyektinin köməyi ilə qrafiklərin yaradılması

Nəhayət VBA proqram kodu ilə bağlı bütün izahatlar verildikdən sonra həmin VBA kodunun listingini tam şəkildə aşağıda veririk:

```

Sub RNGEC ()
'***** 20/12/1998, author: Sahib T.Huseynov, Ganja *****
Dim n,rm,t0,t1,y10,y20,a,b,c,k,L As Range
'** Məslənin ilkin verilənlərinin Inpt səhifəsindən oxunması **
Set n=Worksheets("Inpt").Range("b3")
Set rm=Worksheets("Inpt").Range("b4")
Set t0=Worksheets("Inpt").Range("b5")
Set t1=Worksheets("Inpt").Range("b6")
Set y10=Worksheets("Inpt").Range("b7")
Set y20=Worksheets("Inpt").Range("b8")
Set a=Worksheets("Inpt").Range("b9")
Set b=Worksheets("Inpt").Range("b10")
Set c=Worksheets("Inpt").Range("b11")
Set k=Worksheets("Inpt").Range("b12")
Set L=Worksheets("Inpt").Range("b13")
'#####
'# Proqramın əsas hissəsi: Runge-Kutta üsulu ilə ADT-nin ədədi #
'# həllinin icrası. Burada y(1) və y(2) axtarılan funksiyaların #
'# ədədi qiym.-dir #
'#####
Dim i, r As Integer
Dim t,z(2),f(2,300),y(2,200),kft(4,2),P,Q As Variant
t=t0
y(1,1)=y10
y(2,1)=y20
h=t1/rm
For i=1 To n
z(i)=y(i,1)
Next i
For r=1 To rm
GoSub Funcs
For i=1 To n
kft(1,i)=h*f(i,r)
y(i,r)=z(i)+0.5*kft(1,i)
Next i
t=t+0.5*h
GoSub Funcs
For i=1 To n
kft(2,i)=h*f(i,r)
y(i,r)=z(i)+0.5*kft(2,i)
Next i
t=t+0.5*h
GoSub Funcs
For i=1 To n
kft(3,i)=h*f(i,r)
y(i,r)=z(i)+kft(3,i)
Next i
GoSub Funcs
For i=1 To n
kft(4,i)=h*f(i,r)
y(i,r)=z(i)+(1/6)*(kft(1,i)+2*_
(kft(2,i)+kft(3,i))+kft(4,i))
z(i)=y(i,r)
Next i

```

```

'*****Aşağıdaki kod sətirləri nəticələri Immediate Window ****
'*****pəncərəsində çap edir ****
    Debug.Print t, y(1, r), y(2, r)
'***** Nəticələrin Excel-in Output səhifəsinə çıxarılması ****
    Set curcell=Worksheets("Output").Cells(r, 1)
        curcell.Value=t
    Set curcell=Worksheets("Output").Cells(r, 2)
        curcell.Value=y(1, r)
    Set curcell=Worksheets("Output").Cells(r, 3)
        curcell.Value=y(2, r)
'#####
    Next r
Exit Sub
'*** Aşağıdaki funksiya modelin ADT-lərini formalaşdırır ****
Funcs:
P=0
Q=0
f(1, r)=a*y(1, r)+b*y(1, r)*y(1, r)+c*y(1, r)*y(2, r)+P
f(2, r)=k*y(2, r)+L*y(1, r)*y(2, r)+Q
'*****
Return
Flgl:
    End Sub

```

13. Microsoft Access-də PROQRAMLAŞDIRMA

13.1 Access-də tətbiqi proqramların yaradılmasındakı xüsusiyyətlər

Access-də VBA proqramlarının yaradılması, tipik hallar

Access-dəki proqramlaşdırma Word, Excel və digər MS Office proqramlarında olan proqramlaşdırmadan kəskin olaraq fərqlənir [8, 10, 11, 12, 15, 16, 18]. Əsas prinsiplial fərq bundan ibarətdir ki, Word, Excel, PowerPoint, FrontPage v.s. kimi əlavələr istifadəçinin bilavasitə onlarla işləməsi üçün nəzərdə tutulub – proqram məhsulunun tərtibatçıları tərəfindən heç bir proqram səviyyəsində olan düzəlişsiz. Əlbəttə, bəzən Access də tamamlanmış bir tətbiqi proqram kimi istifadəçilər tərəfindən istismar edilir. Əksər hallarda o, tərtibatçıların öz proqramlarının yaradılması üçün platforma səviyyəsində istifadə edilir.

İkinci fərq budur ki, Access-də verilənlərlə işləmək üçün onun öz nüvəsi qurulmuşdur. Faktiki olaraq, Access – verilənlər bazasının idarə edilməsində tam dəyərli və mükəmməl bir sistemdir. Buna görə onun hər tərəfli imkanlarından istifadə etmək üçün verilənlər bazası ilə işləməyin prinsiplərini bilmək lazımdır: cədvəllər və onların arasında olan əlaqələr (açarlar sistemi), verilənlərin normallaşdırılması, verilənlər tipi, tamlığın məhdudiyətləri v.s. Əksər hallarda müəssisə və şirkətlərdə işləyən Access istifadəçiləri bu haqda heç bir şey bilmirlər və ya, ən yaxşı halda, bilikləri səthi səviyyədə olur.

Proqramların arxitekturası baxımından Access-in istifadəsində müxtəlif variantlar mövcuddur. Bəzən Access (MDB faylı) sadəcə cədvəllərdə olan verilənləri idarə edən bir nüvə kimi istifadə edilir. İstifadəçilər bu verilənlərlə tərtibatçılar tərəfindən yaradılmış və xaricdə yerləşən tətbiqi proqramlardan işləyirlər: məsələn, Visual Basic, Delphi və ya C++ proqram təminatlarında. Bunun

əksinə olaraq, bəzən Access-də yaradılmış istifadəçi interfeysi səviyyəsində verilənlərlə işləyirlər (verilənlər bu halda fiziki olaraq verilənlər bazası serverində yerləşir, məsələn, SQL Server, Oracle, IBM D2 v.s.).

Access-də qurulmuş sorğular dili kimi **JET SQL** nəzərdə tutulub. Bu dilin aktiv köməyi ilə Access-də verilənlərlə işləmək üçün olduqca əlverişli və rahat mühit yaradılıb.

Access - əslində güclü verilənlər bazası alətidir. Əlbəttə, dərs vəsaitinin bu fəslində baxılan VBA proqramlaşdırması Access-in verilənlər bazası sahəsində olan bütün imkanlarını əhatə edə bilməyəcək. Çünki Access-lə işləməyin daha çox imkanları (məsələn, onun sorğular dili və ya cədvəllərin yaradılması) bu kitabın çərçivəsindən kənar qalır: həmin imkanlar haqqında ətraflı məlumat verən xüsusi ədəbiyyat vardır. Qeyd etməliyə ki, həmin ədəbiyyatda da məhz VBA proqramlaşdırması və Access-in öz obyekt modeli kitabın çərçivəsindən kənar qalır. Buna görə oxucu kitabdakı bu fəslə lazımcasına mənimsəməyi bacarsa, onda Access haqqında olan bilikləri tamamlamış olacaq.

Bizim üçün ən birinci aktual olan sual budur: Access-dəki avtomatlaşdırma vasitələrindən istifadə edərək, müəssisə və şirkətlərdə hansı məsələlər həll edilə bilər? Bu suala bir mənalı cavab budur – Access-in verilənlər bazasının idarəetmə sistemi olduğuna görə əksər hallarda ondan verilənlərin saxlanacağı yeri (konteyneri) kimi istifadə edirlər. Verilənlər bu halda bir birindən tamamilə fərqli ola bilər: bağlanmış müqavilələr haqqında olan adi verilənlər müxtəlif formatlı rəqəmsal fotoqraflar, hesabatların və raportların avtomatlaşdırılmış generasiyasında istifadə edilən, Word və Excel şablonları. Access-in üstünlüyü bundan ibarətdir ki, bunların hamısının qrafik interfeyslə birgə bir MDB faylına “paketləmək” mümkündür. Bu işə proqramı çox kompakt və əlverişli formaya salır: bir kompyuterdən başqasına MDB faylının keçirilməsi rahat mühitdə icra edilir. Access-in başqa vacib təyinatı budur - **MS SQL Server, Oracle, IBM DB2** kimi güclü müştəri-server sistemlərində yerləşən verilənlərlə işləmək üçün müştəri interfeysinin təmin edilməsi. Microsoft-un şərti qradasiyasına görə eyni zamanda verilənlərə müraciət edən istifadəçilərin sayı 10-dan artıq deyilsə, onda Access, FoxPro, Paradox v.s. kimi sadə sistemlərdən istifadə edilməsi məsləhət görülür. Əgər eyni zaman müraciət edən istifadəçilərin sayı daha çox olsa və ya verilənlərin həcmi çox böyük olsa, onda daha mürəkkəb strukturlu və daha funksional müştəri-server sistemlərindən istifadə etmək məsləhət görülür. Lakin verilənlərlə əlçatma təminatı səviyyəsində (müştəri-server sistemləri və ya Access verilənlər bazasında) daha ixtisaslaşmış tətbiqi proqramlar müxtəlif məsələləri həll edir:

- *adi formaların yaradılması* – verilənlər bazasında və Web-formalarda verilənlərin daxil edilməsi/dəyişdirilməsi/baxılması icra etmək üçün proqram interfeyslərinin yaradılması (onlara xüsusi terminologiyada verilənlərə əlçatma səhifələri deyirlər);
- *verilənlər bazası üçün müxtəlif formada hesabatların yaradılması (həmçinin parametrləşmiş hesabatların);*

- *adi üsul ilə tətbiqi proqramın proqram məntiqinin yaradılması* – VBA-da (modullar) və başlayan istifadəçilər üçün makroslar (makrosları istənilən zaman modullara çevirmək mümkündür);
- *köməkçi əməllər* – verilənlərin çapı, eksportu, dəyişdirilməsi (hərçənd ki, verilənlərin dəyişdirilməsinin daha asan üsulu – DTS obyekt modelindən istifadə edilməsidir) verilənlərin yüklənməsi, replikasiya v.s.

13.2 Access tətbiqi proqramlarının yaradılmasının əsas mərhələləri

Access-də verilənlər bazası ilə qarşılıqlı işləyən VBA proqramlarının yaradılması, tətbiqi proqramların layihələndirilməsi və reallaşdırılması

Əgər müəssisə və şirkətdə istifadəçi elə bir fəaliyyətlə məşğul olasıdır ki, yaratdığı proqramlarda olan verilənlər sonra hansısa mənbədə saxlanmalıdırsa, (məsələn, Access verilənlər bazasında və ya müştəri-server sistemində) üstəlik bu işdə o, peşəkar tərtibatçı rolunda fəaliyyət göstərməlidir, onda həmin proqramların layihələndirilməsində və yaradılmasında mövcud olan və məsləhət görülən bütün mərhələləri yaxşı mənimsəyib sonra icra etməyi bacarmalıdır. Bununla yeni başlayan istifadəçi sonradan düzəltməsi çətin olan səhvlərdən özünü sığorta etmiş olar. Həmin işlərin aparılması, şərti olaraq, aşağıda təsvir edilib:

- **Birinci mərhələ** - *müəssisə və şirkət tələbatların, mövcud olan bölmələrinin (departamentlərinin v.s.) və istehlakçılar haqqında olan məlumatın toplanması*: məsələn, irsindən asılı olaraq hansı sistemlər burada işləyir, nə ilə qarşılıqlı uyğunlaşma yaradılmalıdır, informasiya axınları necə təşkil olmalıdır, sistemin dəyişkənliyi nə təhərdir v.s. Əksər hallarda bu işlərdə Microsoft Visio (vektor-qrafik redaktor və ya Windows üçün diaqram və blok-sxem redaktoru) və ya ERW-in (verilənlər bazasının layihələndirməsi və sənədləşdirilməsi üçün proqram vasitəsi) istifadə edilir.
- **İkinci mərhələ** - *tətbiqi proqramın arxitekturasının seçilməsi, uyğun olan verilənlər bazasını idarə edən (VBİS) sistemin seçilməsi*: bu mərhələdə məlumatın harada saxlanması (müştəri server sistemində yoxsa Jet nüvəsində saxlanacaq) təyin edilir, məlumatın saxlanmasını təmin edən cədvəllər və onların arasında olan əlaqələr layihələndirilir. Bundan əlavə, tətbiqi proqramın arxitekturası təyin edilir – neçə sayda səviyyə olacaq, terminal və ya Web-texnologiyaları tətbiq edilməsi/edilməməsi, replikasiyaların olub/olmaması v.s. təyin ediləcək.
- **Üçüncü mərhələ** - *VBİS (verilənlər bazasının idarə sistemi) reallaşması və tətbiqi proqramın biznes-məntiqi*. Bu mərhələdə verilənlər bazasının obyektləri layihələndirilir, yaradılır, sazlanır və verilənlərin ilkin qiymətləri ilə doldurulur – cədvəllər, formalar, hesabatlar, saxlanılan proseduralar, makroslar və proqram modulları v.s. Access-də tətbiqi proqramlar yaradılanda, adları çəkilən əməllərin əksəriyyətini Access-in qrafik əlavəsində aparılır. VBA kodunun istifadə etmə sahələri bunlardır: istifadəçinin daxil etdiyi verilənlərin

yoxlanması; formada, idarəetmə elementləri ilə işləmədə, formalar arasında keçidlərin təşkili; başqa idarəetmə elementlərinin əlaqələnməsi; hesabatların avtomatlaşmış generasiyası; xaricdə olan obyekt modellərinə müraciət etmə v.s. Bu mərhələdə MS Visio və ERWin proqram təminatları kömək edə bilər.

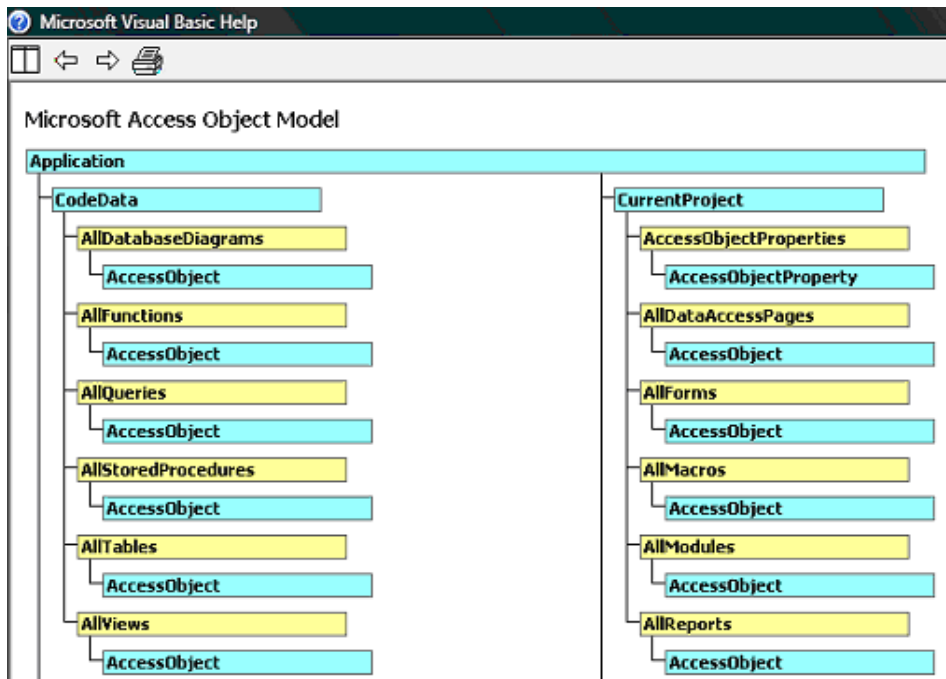
- **Dördüncü mərhələ** - verilənlər bazasının optimallaşdırılması. Bu məsələ kompleks xassəlidir: həmçinin bu mərhələdə VBA proqram kodları da optimallaşdırılmalıdır.
- **Beşinci mərhələ** - *tətbiqi proqramların testlənməsi və sazlanması (kitabın 6-cı bölməsinə bax).*
- **Altıncı mərhələ** - tətbiqi proqramın istismarına başlanması.

Gördüyünüz kimi, bu mərhələlər kifayət qədər kompleks xassəlidir və bu kitabda hər xassəyə baxılması mümkün deyil. Məhz buna görə, dərs vəsaitinin məqsədi və mövzusunə uyğun olaraq, biz yalnız Access-dəki VBA proqramlaşdırılması ilə olan məqamlara baxacağıq.

13.3 Application obyekt, onun xassə və metodları

Access.Application obyekt, xaricdə olan proqramdan Access-in işə salınması, Access.Application obyektinin xassələrə və metodları

Access-in obyekt modeli öz arxitekturasına görə Word və Excel proqramlarından kəskin olaraq fərqlənir, bax şəkl. 13.1 (cəmi 85 sayda obyekt mövcuddur, onların hamısının şəkildə yerləşməsi mümkün deyil – oxucu, lazım olduqda, MS VBA Help sorğu sənədlərindən sxemin tam ölçülü görüntüsünü tapa bilər).



Şəkil 13.1 MS Access obyekt modelinin struktur sxemi (cəmi 85 sayda obyekt mövcuddur, onların hamısı şəkildə yerləşməsi mümkün olmayıb).

Access proqramlaşdırmasının Word və Excel proqramlaşdırması ilə müqayisədə çox sayda fərqlər var. Lakin burada da **Application** obyektı mövcuddur – Access arxitekturasının ən yüksək tərə nöqtəsində durur (şək. 13.1). Analoji olaraq, həmin obyektin köməyi ilə Access-i başqa proqramdan proqram səviyyəsində işə salmaq mümkündür. Həmçinin onun xassələri və metodları kodun istənilən hissəsindən əlçatandır. Məsələn, Access-in başqa proqramından işə salınmasını icra edən VBA kodu aşağıdakı nümunədə olan kimi görünə bilər:

```
Dim appAccess As Object
Set appAccess=CreateObject("Access.Application")
appAccess.Visible=True
MsgBox appAccess.Name
```

Əgər yuxarıdakı **Visible** xassəsi **True** qiyməti ilə qurulmasa, onda standart halda Access görünməyən pəncərədə açılacaq (onu yalnız **Task Manager** proseslər siyahısında görmək olacaq). Əgər bu kod Word və ya Excel-dən işə salınsa, onda standart halda, proqramın icrası bitdikdə, obyekt operativ yaddaşdan çıxarılacaq (məhz buna görə yuxarıdakı VBA kodunda **MsgBox** xəbər pəncərəsi yerləşdirilib – prosesi saxlamaq üçün).

Access-in işə salınması üçün bir çox başqa proqram üsulları da mövcuddur - **Windows Explorer** obyekt modelindən, əməliyyat sisteminin komanda interpretatorundan, *.mad mətn yarlıqından, API-dən v.s. Praktikada proqram üsulları ilə Access-in işə salınması nadir hallarda rast gəlir – çünki proqramın öz örtüyünün istifadəsi daha əlverişlidir (Access içərisindən Word, Excel v.s. proqramlar da işə salına bilər).

MDB faylındakı verilənlər bazasında olan verilənləri əlçatan etmək üçün Access-i açmağa məsləhət görmürlər – bunun üçün ADO obyektlərini istifadə etmək lazımdır (onlar daha sadədir, rahatdır və resurs tutumludur).

İndi isə Access obyekt modelinin **Application** obyektindən danışaq. Oxucu özü də tam əmin ola bilər ki, onların toplusu, analoji olan, Word və Excel-dəkilərə çox az oxşayır. Əvvəlcə - ən vacib xassələr haqqında danışaq:

- **AutomationSecurity** – verilənlər bazası açıldıqda təhlükəsizlik səviyyəsini təyin edir. Standart halda **msoAutomationSecurityByUI** qiyməti istifadə edilir (Access-in qrafik interfeysinin **Macro→Security** menyusunda qurulanların istifadəsi). Makroslarla işləyən faylların ümumiyyətlə, tamamilə açılmamasını qurmaq olar (**msoAutomationSecurityForceDisable**). Əksər hallarda **msoAutomationSecurityLow** qiymətindən daha çox istifadə edirlər (fayllar əlavə xəbərdarlıqsız açılır);
- **BrokenReference** – parçalanmış istinadların olmasını yoxlayır (proqramın dll modulunun və ya başqa verilənlər bazasının tapa bilməməsi halında). Konkret olan istinadı Reference obyektı ilə tapmaq olar;

- **CodeContextObject** – çox faydalı xassədir, onunla modulun/makrosun hansı verilənlər bazasının obyektindən (hesabatdan, formadan v.s.) işə salınmasını təyin etmək olur. Bu xassə ilə səhvin mənbəyinin tapılmasında da istifadə edirlər;
- **CodeData** – bu xassə də vacib kateqoriyaya aiddir: **AllDatabaseDiagrams**, **AllFunctions**, **AllQueries**, **AllStoredProcedures**, **AllTables** və **AllViews** kolleksiyalarını əlçatan edir. Bu kolleksiyalarda **AccessObject** ilə Excel verilənlər bazasında olan cədvəllərin, sorğuların, diaqramların və digər obyektlərin onlarla xassələrini təyin etmək olur. **CodeData** xassəsinin tətbiqi nəticəsində verilənlər bazasında olan bütün cədvəllər haqqında alınan məlumatı aşağıdakı VBA kodu ilə almaq olar:

```
For Each oTable In CodeData.AllTables
    Debug.Print oTable.Name
Next
```

- **CodeProject** - funksionallıq baxımından **CodeData** xassəsinə bənzəyir, yalnız **AllForms**, **AllReports**, **AllMacros**, **AllModules** və **AllDataAccessPages** proqram modullarının kolleksiyalarını əlçatan edir.
- **CommandBars** – xassəsi **CommandBar** obyektlərini qaytarır (yeni Access alətlər panelini). Bu kolleksiyayı Access üzərində qurulmuş olan tətbiqi proqramın istifadəçi interfeysinin sazlanmasında istifadə etmək olar.
- **CurrentData** – analogi olaraq, **CodeData** və **CodeProject** kimi işləyir: həmin kolleksiyaları əlçatan edir. Həmçinin **CurrentProject** xassəsi xaricdə olan verilənlər mənbəyində (SQL Server-də) eyni qaydada işləyir.
- **CurrentObjectName** və **CurrentObjectType** – xassələri hansısa kodun hansı obyektəndən çağırılmasını təyin edir. Bu xassələri yoxlamalarda istifadə etmək əlverişli olur.
- **DataAccessPages** – bütün Web-formalarda olan obyektlərin daxil olduqları kolleksiyaya istinad almağa imkan yaradır.
- **DBEngine** – xassəsi **DBEngine** obyektini əlçatan edir (bunula Access-in “mühərriki” olan Jet nüvəsinə baxmaq və onu sazlamaq olur). Məsələn, bu xassə ilə verilənlər bazasını sıxmaq, parolunu təyin etmək, istifadə edilən kodlaşmanı təyin etmək olur.
- **DoCmd** - ilə çox vacib olan **DoCmd** obyektini əlçatan etmək olur (onunla çox sayda vacib olan əməlləri icra edirlər). Faktiki olaraq, bu obyekt VBA proqramlaşdırması baxımından Access-in ən əsas işçi qüvvəsidir. Daha ətraflı bu obyektə növbəti bölmələrdə baxacağıq. **DoCmd** obyektinin özünü yaratmaq lazım gəlmir – o, onsuz da **Application** obyektindən əlçatandır məsələn, aşağıdakı VBA kodunda olan kimi:

```
Application.DoCmd.RunSQL "Delete from table1"
```

- **Forms** – xassəsi **Form** obyektləri kolleksiyasına istinad qaytarır. Əvvəl baxdığımız **AllForms** kolleksiyasından bu kolleksiya iki məqamla fərqlənir:
 - hal-hazırda açıq olan formalar onun daxilində yerləşir;
 - **AccessObject** obyektləri tərkibində yoxdur, əvəzinə daha çox xassəsi və metodları olan **Form** obyektləri vardır.
- **MenuBar** – xassəsi istifadəçi menyusunu standart olmayan qaydada qurmağa imkan verir (bütün tətbiqi proqramın səviyyəsində - əgər forma üçün öz menyusu nəzərdə tutulubsa, onda onun xüsusi prioriteti olacaq).
- **Modules** – xassəsi **Forms** xassəsinə analojidir: eyni adlı **Modules** obyektlərini əlçatan edir. Bununla standart olmayan modullar və siniflər modulu əlçatan olur;
- **Printers** və **Printer** – xassələri standart halda sistemə qoşulmuş bütün printerləri və onların xassələrini qaytarır. **Printer** xassəsi ilə printerin bütün xassələrini proqram səviyyəsində qurmaq olur;
- **References** - xassəsi **Refernce** obyektinin kolleksiyasına istinad almağa imkan verir (tiplər kitabxanasına istinad alır). İstinadların işlək halda olmasının yoxlanmasında və ya proqram üsulu ilə istinadların əlavə edilməsində istifadə edilə bilər;
- **Reports** – xassəsi **Forms** və **Modules** xassələrinin işləmə prinsipi ilə işləyir, **Report** obyektlərini əlçatan edir;
- **Screen** – xüsusi olan **screen** obyektini əlçatan edir. Bu obyektlə Access-in aktiv elementlərini əlçatan etmək mümkündür: formaları, hesabatları, idarəetmə elementləri v.s. Bu obyektlə həmçinin kursurun, mausun görkəmini dəyişmək olur;
- **UserControl** – xassəsi ilə Access-in necə işə salınmasını təyin etmək olur (əl ilə və ya başqa tətbiqi proqramdan proqram səviyyəsində). Bundan asılı olaraq, məsələn, Access-in avtomatik bağlanması və ya istifadəçinin qərarı ilə bağlanmasını təyin etmək olar;
- **Version** – Access-in versiyasını qaytarır (məsələn proqramların işləmə qabiliyyətinin əlavə yoxlanmasını təmin edir, Access 2003 üçün qaytarılan qiymət: 11.0.).
- **Visible** - xassəsi istifadəçi üçün Access-i görünən edir (və ya əksinə görünməz edir, gizlədir). Standart halda, proqram səviyyəsində Access işə salındıqda, Access görünməz olur, buna görə həmin xassənin qiymətini **True** ilə qurmaq lazımdır;

Application obyektində olan metodların sayı da bir o qədər az deyil (üstəlik metodların bir hissəsi **DoCmd** obyektinə köçürülüb). Aşağıda, onlardan VBA Access proqramlaşdırılmasında ən vacibləri haqqında, məlumat veririk:

- **AccessError()** – səhvləri emal edən vacib metoddur: standart **Err.Description** metodunun qaytardığı, çox məna verməyən, "**Application-defined or object-defined error**", əvəzinə bu metod Access və DAO kitabxanalar səhvləri haqqında ətraflı məlumat qaytarır;
- **BuildCriteria()** – metodu SQL-sorğularda, formalar və hesabatlar filtrində v.s. istifadə edilir: çox qısa zamanda və əlverişli olan mühitdə yazıların seçilmə meyarını quraşdırır – düzgün quraşdırılmış sətir qiymətini qaytarır;
- **CloseCurrentDatabase()** – Access bağlanılmadan cari verilənlər bazasını bağlayır. Adətən Access-in yeni faylı işə salmadan, növbəti verilənlər bazasının açılmasında istifadə edilir;
- **CodeDb()** – **DAO.Database** obyektini qaytarır. Həmin obyekt, hal-hazırda icra edilən VBA kodu ilə bağlı verilənlər bazasını təmsil edir: adətən başqa verilənlər bazası ilə müxtəlif əməllər icra edən, proqram modullu xüsusi verilənlər kitabxana bəzi olanda tətbiq edilir. Cari verilənlər bazası üçün həmin obyektə oxşarına istinad almaq üçün **CurrentDb()** metodu istifadə edilir;
- **CompactRepair()** - Access verilənlər bazasının sıxılması/təmiri əməllərini və səhv haqqında olan kod qiymətinin qaytarılmasını icra edir (protokolun yazılması da mümkündür). Həmin məqamda verilənlər bazası bağlı olmalıdır;
- **ConvertAccessProject()** – Access verilənlər bazası versiyasını çevirir: **acFileFormatAccess2** versiyasından başlayaraq, **acFileFormatAccess2002** versiyasına qədər çevirmələr icra edilə bilər;
- **CreateAccessProject()** – metodu ilə proqram səviyyəsində Access layihəsi yaradıla bilər (SQL Server-i əlçatan etmək üçün proqram interfeysini). Onu həmin an yaradıb açmaq üçün **NewAccessProject()** metodu istifadə edilə bilər. Sadəcə açmaq üçün isə **OpenAccessProject()** metodu mövcuddur;
- **CreateAdditionalData()** – XML faylına doğma olan cədvəlin exportunda **ExportXML()** metodu ilə birgə istifadə edilir: **AdditionalData** obyektini yaradır. Cədvəllər toplusunun exportunu təmin edir;
- **CreateControl()** – formada proqram səviyyəsində idarə elementini yaradır. Çox sayda parametrlər qəbul edir. Hesabatda idarəetmə elementini yaratmaq üçün **CreateReportControl()** metodu istifadə edilir. Elementlərin silinməsi isə uyğun olaraq, **DeleteControl()** və **DeleteReportControl()** metodları ilə icra edilir;
- **CreateForm()** – proqram səviyyəsində Access formasını yaratmağa imkan verir (üstəlik həmin proqram üsulu ilə yaradılmış obyektə istinad alınmasını təmin edir). Sonradan bu formanın xassələrini sazlamaq olar, idarə elementlərini əlavə etmək v.s. Analoji olaraq, bu qayda ilə hesabatı **CreateReport()** metodu ilə yaratmaq olar;

- **CreateGroupLevel()** – proqram səviyyəsində hesabatda qruplaşmanı yaradır (və ya yazıları sortlaşdırmaq olur). Hesabatın adını, sortlaşma aparıldığı sütunu, qruplar üçün yuxarı/aşağı kolontitul yaradan düsturları qəbul edir;
- **CreateNewWorkgroupFile()** – proqram səviyyəsində istifadəçi icazəli olan iş qrupu faylını yaradır (qrafik ekranda həmin əməli **Tools**→**Security**→**Workgroup** menyusundan icra edilə bilər);
- **CurrentUser()** – verilənlər bazasının cari istifadəçisinin adını sətir qiyməti şəklində almağa imkan yaradır. Standart halda iş **Admin** istifadəçisi adından icra edilir;
- **D...** - ilə başlayan metodlar SQL kodundan istifadə etməyərək, müxtəlif əməllərin yerinə yetirilməsini bir başa Access-dən icra edilməsinə imkan yaradırlar.
 - **DAvg()**, **DSum()**, **DCount()**, **DMax()**, **DMin()** v.s. – cədvəldə və ya formada sütuna (və ya yazılar toplusuna) aqreqat funksiyalarının tətbiqi etməyə imkan yaradırlar;
 - **DLookup()** - cədvəldən və ya formadan (həmçinin ikili kodlu obyektlər və Word şablonlarından) lazım olan qiymətin tapılmasını və qaytarılmasını təmin edən son dərəcə vacib olan metoddur. Parametr kimi cədvəlin, formanın, sütunun adını və filtrini qəbul edir. Əgər filtr şərtinə bir neçə qiymət uyğun gəlsə, onda birinci qiymət qaytarılır;
 - **DFirst()** və **DLast()** – adlarından asılı olmayaraq, bu metodlar eyni qaydada işləyir: cədvəlin və formanın sütununun tərkibində olan təsadüfə seçilən qiyməti qaytarırlar;
- **Echo()** – metodu Access ekranının təsvirinin çəkilməsini icra edərək, vəziyyət sətirinə mətni çıxarır;
- **Eval()** – çox hallar üçün əlverişli metoddur: mətn sətiri üzərində VBA kodunda işləmə mühitini yaradır. Metod **Variant** tipini qaytarır – bütün qaytarılan qiymətlər yerləşsin deyə. Məsələn aşağıdakı nümunədəki kimi:


```
Eval("1+1")
```

 2 ədədini qaytaracaq,


```
Eval("Mənim_Funksiyam()")
```

 İkinci kod isə həmin funksiya qaytarası hər nə varsa onu qaytaracaq. **Eval()** metodu tiplərin yoxlanılmasında olan çox yer tutan yoxlamalar və çevirmələri aradan qaldırmaq üçün istifadə edilir, məsələn, istifadəçi tərəfindən qiymətlərin daxil edilməsində, onlara müxtəlif qiymətlər verildikdə daha çox istifadə edilir;
- **ExportXML()** və **ImportXML()** – metodları ilə verilənlərdən ibarət cədvəllərin (həmçinin açarlar, indekslər, kodlaşmalar v.s. ilə daxil olaraq) XML-uyğunluqlu fayllara eksport və

import əməllərində istifadə edirlər. Bununla belə, həmin metodlarla icra edilən Access-dən eksport və import əməllərini yalnız Access verilənlər bazası üçün yox, həmçinin, 6.5 versiyasından başlayaraq, bütün SQL Server verilənlər bazaları üçün də icra etmək mümkündür;

- **GetOptions()** və **SetOptions()** – metodları ilə qrafik interfeysin **Tools**→**Options** menyusundan əlçatandır. Onlarla sazlamaları proqram səviyyəsində qurmaq mümkündür, məsələn, **<Enter>** düyməsi basıldıqda, cədvəldə keçid sağa (standart halda belədir) yox aşağı istiqamətə olsun deyə aşağıdakı kod tətbiq edilə bilər.

```
Application.SetOption "Move After Enter", 2
```

- **hWndAccessApp()** – bu metod Access pəncərəsinə olan göstəricinin qaytarılmasını təmin edir (Windows API ilə işləyənlər üçün əlverişli şərait yaradır);
- **NewCurrentDatabase()** – yeni Access verilənlər bazasının yaradılması və həmin an açılmasını icra edir. Mövcud olan verilənlər bazasının açılması üçün isə **OpenCurrentDatabase()** metodundan istifadə etmək olar;
- **Nz()** - əgər cədvəlin verilmiş sütununda olan qiymət təyin edilməyibsə (**Null**), onda metod boş sətir və ya başqa qiyməti qaytarır. Peşəkar VBA proqramçıları, cədvəldə axtarış aparanda, bu metoddan tez-tez istifadə edirlər – boş qiymətlərə (istənilən, həmçinin **Memo** daxil edilməsi ilə) müraciət etdikdə səhvlərin yaranmasının qarşısını almaq üçün;
- **Quit()** – yaddaşda heç nə saxlamadan və ya istifadəçini sorğu edərək, proqramdan çıxmanı icra edir;
- **RefreshDatabaseWindow()** – verilənlər bazasını yeniləşdirir: adətən formaların, hesabatların v.s. proqram səviyyəsində yaradılmasından sonra istifadə edilir;
- **Run()** – metodu VBA kodundan proseduranı və ya funksiyayı çağıraraq, ona 30 sayə yaxın parametrlər ötürür. Qurulmuş olan funksiyalarının və ya istifadəçi funksiyalarının çağırılmasında istifadə edilə bilər. Onları standart üsullarla da çağırmaq mümkündür - daha çox bu metoddan Access-in başqa kompilyasiya edilmiş proqramdan (məsələn, **DLL** və ya **EXE** genişlənməsi olan proqramlardan) çağırılmasında istifadə edirlər;
- **RunCommand()** – bu metod Access-in onlarla qurulmuş komandalarının birisini icra edir: faktiki olaraq, idarəetmə panelində və qurulmuş menyularda hər nə varsa. Məsələn, Access-in pəncərəsini maksimallaşdırmaq üçün bu VBA kodundan istifadə etmək olar:

```
RunCommand acCmdAppMaximize
```

- **SysCmd()** – bir çox xidməti əməlləri yerinə yetirir: məsələn, ev kataloqu, Access versiyası, işarə edilmiş verilənlər bazasının vəziyyəti haqqında məlumatın alınması, vəziyyət sətrindən bəzi əlavələrin işə salınması v.s.

13.4 Makrokomandalar və DoCmd obyektı

Access .DoCmd obyektı, makrokomandaların VBA vasitələri ilə işə salınması

DoCmd obyektı – yuxarıda qeyd etdiyimiz kimi, Access VBA proqramlaşdırılmasının əsas “işçi qüvvəsidir”. Bu obyekt proqram səviyyəsində Access makrokomandalarını icra edir. Bu əməllər - Access-də proqram səviyyəsində icra edilən ən geniş yayılmış operasiyalardır.

DoCmd obyektinin xassəsi yoxdur – onun yalnız metodları vardır. Son Access versiyalarında unifikasiya mənası ilə **DoCmd** metodu **Application** obyektinin tərkibinə keçirilib. Microsoft mümkün qədər **Application** obyektinin eyni adlı metodlarından istifadə edilməsini məsləhət görür.

Burada **DoCmd** metodunun tərkibində olan digər metodların şərhli və izahlı siyahısının gətirilməsinə heç bir ehtiyac yoxdur – çünki bu metodlar eyniliklə makrokomanda dizaynerində olanlara uyğundur (makrokomandaların sayı bir qədər çoxdur – çünki onlardan bəziləri üçün xaricdə olan proqramın işə salma, xəbər vermə, klaviş basılma ötürməsi pəncərələri üçün ayrıca VBA vasitələri mövcuddur).

Makrokomandaları da burada ətraflı izahlarının verilməsi aktual deyil – çünki onların hər biri üçün **<F1>** klavişi basıldıqda, rəsmi sənədləşmədən kifayət qədər ətraflı sorğu materialları əmələ gəlir. Onlar üçün əl ilə kodun yazılması da mənasızdır – hər an yaradılmış makrosu VBA moduluna çevirmək və sonradan alınmış koda baxmaq mümkündür. Aşağıda **DoCmd** obyektinin, VBA proqramlaşdırılmasında vacib olan, yalnız makrokomandaların bəzi imkanları haqqında danışılır:

- **OutputTo()** (**OutputInFormat** makrokomandasına uyğundur), **TransferText()** (**TransferText** makrokomandasına uyğundur), **TransferDatabase()** (**TransferDataBase** makrokomandasına uyğundur), **TransferSpreadsheet()** (**TransferSpreadSheet** makrokomandasına uyğundur) – bu makrokomandalar verilənlərin Excel, RTF, SNP TXT, DBF, ODBC verilənləri mənbəyindəki formatlarda import/eksport icrasını təmin edir.
- **RunSQLquery**, **RunMacro**, **RunApplication**, **RunProgram** – adlarından belə məlumdur. Proqram dedikdə VBA prosedurası və ya funksiyası anlanmalıdır, tətbiqi proqram dedikdə isə əməliyyat sisteminin xaricdə yerləşən proqramı kimi anlamaq lazımdır.
- **Open...Table**, **Query**, **Form** v.s. – adlarından oxucu üçün tam aydın ola bilər. Açılma rejimini və digər başqa rejimləri (dizayner, baxma v.s. üçün) seçmək olar. Obyekt açıldıqda onun VBA kodundan və idarəetmə elementlərindən istifadə etmək olar.

Həmçinin **Access.DoCmd** obyektinin makrokomandalar vasitələri VBA mühitində verilənlər bazasını və ayrıca faylları kopyalamaq, yazıları axtarmaq, idarəetmə elementlərini aktivləşdirmək və digər daha çox əməllərin icrasını yerinə yetirmək imkanları vardır. Lazım olduqda, təkrarən yada salırıq ki, əlavə məlumatları oxucu VBA Access sənədləşməsinin (lisenziyalı MS Office paketi

istifadəçi kompyuterinə düzgün yükləndikdə həmin sənədləşmə sistemində heç bir problem yaranmamalıdır) sorğu materiallarından tapa bilər.

13.5 VBA-dan Access formaları ilə işləmə: Form obyektı

Access.Form obyektı, VBA vasitələri ilə Access formaları ilə işləmə, Access formalarının açılması, Access formasında idarəetmə elementləri ilə işləmə

Access-də daha çox istifadə edilən element – formalardır. Access formalarının təyinatı adı VBA formalarının təyinatından heç də fərqlənir – başlıca olaraq, onlar qrafik idarəetmə elementlərinin görüntüsü konteynerləridir (saxlanc yerləridir). Bununla belə Access formalarının quruluşu, funksional imkanları, onlarla işləmə üsulları hətta mövcud olan idarə elementləri Word və Excel-də istifadə edilən VBA-nın bizə tanış olan formatlarından köklü olaraq fərqlənir.

Real, praktiki işlərdə Access formalarının tətbiq etmə halları bunlardır:

- *Axxess verilənlər bazası cədvəllərində və digər xarici mənbələrdə yerləşən verilənlər bazasındakı yazıları redaktə etmək.* Bu cür formaların yaradılmasında proqramlaşdırma ümumiyyətlə, lazım deyil – dizayner rejimində formalar yaradıldıqdan sonra **Toolbox**-dan (formalar yaradan alətlər paleti) istifadə etmək lazımdır. Sonrakı addımda xaricdə yerləşən verilənlər mənbəyinə qoşulmaq lazımdır (məsələn, SQL və ya Oracle verilənlər bazasına): Access-in qrafik interfeysinin **Fayl→Get External Data→(İmport və ya Link Tables)** menyusundan istifadə edilir.
- *İstifadəçi proqramının idarəetmə paneli kimi istifadə edildikdə.* Bir çox hallarda Access əsasında başlanğıc forma yaradılır: proqram işə salındıqda o, avtomatik açılır. Həmin formada başqa formaların, hesabatların, makrosların, sadəcə sistemdən çıxma imkanının v.s. çağırılması (icra edilməsi) üçün xüsusi düymələr və digər idarəetmə elementləri nəzərdə tutulur. Başqa formalar bağlandıqda idarəetmə həmin başlanğıc formaya ötürülür.
- *Sadəcə istifadəçiyə istənilən əməllərin yerinə yetirilməsi üçün imkan yaratmaq.* Məsələn, formalardan hesabatların parametrlərini təyin etmək, verilənlərin digər xaricdə yerləşən proqrama yüklənməsini icra etmək.

Bəs VBA-dan proqram səviyyəsində Access formaları ilə necə işləmək lazımdır?

Birincisi bunu yaxşı bilmək lazımdır – bütün hallarda formalarla işləmək üçün **AccessObject** ümumi obyektinin istifadə edilməsi tələb olur. Formalardan əlavə bu obyekt Access-də cədvəlləri, makrosları, modulları, hesabatları və digər çox sayda elementləri təmsil edir. **AccessObject** obyektinin universal olduğuna görə əvvəlcə VBA redaktorunda oxucunun tanış olduğu köməkçilərdən (**Intellisense**) böyük fayda olmayacaq. Forma obyektinə müraciət edilməsi **AllForms** kolleksiyasından (**CodeProject** və **CurrentProject** obyektlərindən əlçatandır) mümkündür. Məsələn, Access verilənlər bazasında bütün formalar haqqında olan məlumatı aşağıdakı VBA kodu ilə almaq olar:

```

Dim oA As AccessObject
For Each oA In CurrentProject.AllForms
    Debug.Print oA.Name
Next

```

Əgər **AllForms** kolleksiyasında olan formalara indeksləri ilə müraciət etsək, onda diqqət vermək lazımdır ki, həmin kolleksiyadakı nömrələndirmə 0 ilə başlasın. Bu kolleksiyadakı elementlərə adları ilə müraciət etmək olar:

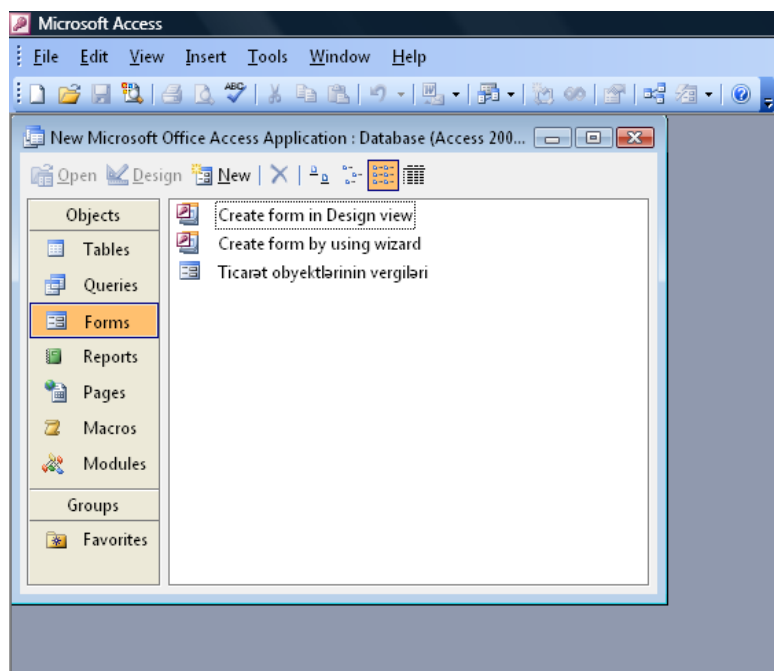
```
Debug.Print CurrentProject.AllForms("Form1").IsLoaded
```

Bu proqram kodunda **IsLoaded** xüsusi xassəsi həmin formanın açıq olub olmamasını (yəni maşının operativ yaddaşına yüklənməsini) təyin edir.

Proqram səviyyəsində formaları başqa üsulla da tapmaq olar. Bütün Access açıq formaları, avtomatik olaraq, **Application.Forms** kolleksiyasına yerləşdirilir və **Form** obyektləri kimi təmsil edirlər. Bu xassəni formanın genişliyinin dəyişdirilməsində də istifadə etmək olar - bu halda aktiv obyektin ölçülərini dəyişən **DoCmd.MoveSize()** metodundan istifadə etməyi məsləhət görürlər. Məsələn, yuxarıda baxdığımız nümunə üçün VBA kodunu bu cür yazmaq olar:

```
DoCmd.MoveSize Width:=10000
```

Formanı necə açmaq olar? Ən birinci bunu bilmək vacibdir - əgər Word və Excel-də formanın açılması üçün proqram səviyyəsində VBA kodunu tərtib edib, sonra icra edilməsi lazımdırsa, Access-də buna ehtiyac yoxdur. Access-in qrafik interfeysdən “əl ilə” verilənlər bazasının pəncərəsindən formanı açmaq mümkündür, bax şəkl. 13.2. Adətən həmin pəncərədən yeni formaların yaradılması və ya mövcud olanların dəyişdirilməsini də icra edirlər.

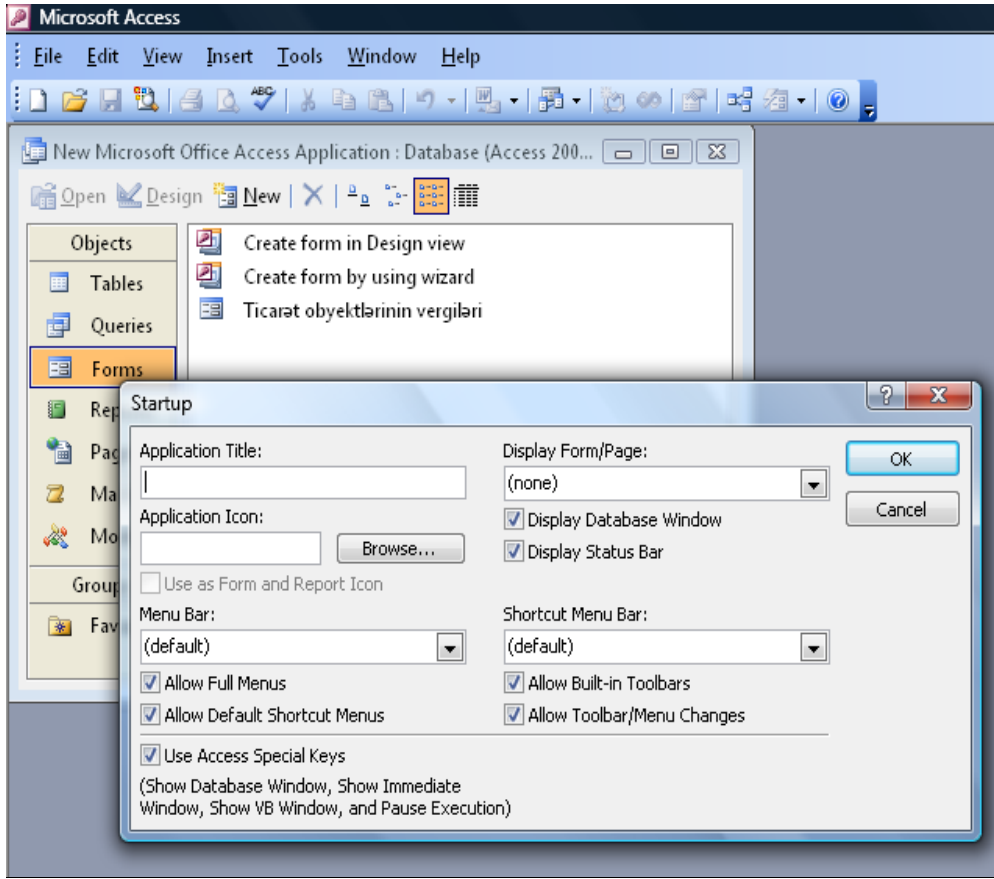


Şəkil 13.2 Access-dəki formalarla işləmə dialoq pəncərəsi.

Daha bir tez-tez istifadə edilən üsul – Access verilənlər bazası açıldıqda, sadəcə forma işə salınır. Bunun üçün Access-in qrafik interfeysinin **Tools→Startup** menyusundan istifadə edilir və

əmələ gələn dialoq pəncərəsində lazım olan obyektin çıxarılması (işə salınması) qurulur, şəx. 13.3. Verilənlər bazasının pəncərəsi bu halda istifadəçidən gizlənmiş halda olur, buna görə istifadəçi qrafik interfeysdən verilənlər bazasında olan cədvəllər, formalar, modullar haqqında heç bir məlumat ala bilməyəcək – verilənlər bazasının bütün xidməti elementləri gizlənmiş halda olur. Əlbəttə, istifadəçi bu halda həmin narahatlığı aradan qaldıra bilər: Access işə salındıqda klaviaturanın **<Shift>** düyməsini basılı vəziyyətdə saxlaya bilər. Bu halda proqram səviyyəsinə keçmək daha rahat mühit yarada bilər. Belə hal yarandıqda həmin məqsəd üçün proqram səviyyəsində **DoCmd.OpenForm()** metodundan istifadə etmək lazımdır. Bu metod ən sadə halda qəbul edilən parametrlər kimi, formanın adını götürür:

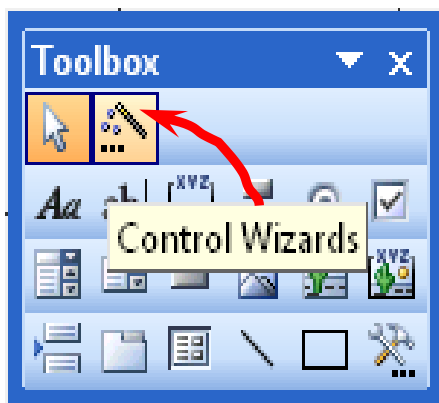
```
DoCmd.OpenForm "Form1"
```



Şəkil 13.3 Access verilənlər bazası açıldıqda sadəcə formanın işə salınması üçün qrafik interfeysinin **Tools**→**Startup** menyusundan əmələ gələn **Startup** dialoq pəncərəsi.

Lakin forma artıq açılıbsa, onda bu metod formanı yenidən açır: sadəcə formanı aktivləşdirir. **DoCmd.OpenForm()** metodu həmçinin bir neçə sayda məcburi olmayan parametrlər də qəbul edir: onların köməyi ilə istifadəçi formadakı yazıların filtrlərini qura bilər, formanın açılma rejimini sazlaya bilər v.s. Formanın bağlanması isə **DoCmd.Close()** metodu həyata keçirir. Lakin əgər istifadəçi sadəcə aktiv olan formanı gizlətmək istəyirsə, (formada istifadəçinin daxil etdiyi qiymətləri yaddaşda saxlamaq və gələn göstəridə onu əks etmək üçün), onda metodun **Visible** xassəsinə **False** qiyməti verilməlidir.

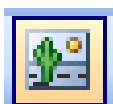
Adətən forma öz-özlüyündə lazım olmur, ondan üzərində (tərkibində) idarəetmə elementləri olan konteyner (saxlanc yeri) kimi istifadə edirlər. Əksər hallarda idarəetmə elementlərini proqram səviyyəsində yaratmağa ehtiyac qalmır – daha sadə və rahat yol formanın dizayner rejimində **Toolbox**-dan lazımı idarəetmə elementlərinin seçilib formanın üzərində yerləşdirməkdir. **Toolbox**-dakı bir çox idarəetmə elementləri ilə oxucu artıq tanışdır: mətn sahələri, üst yazıları, düymələr, bayraqcılar və keçiricilər. Oxucuya bu kitabda rast gəlmədiyi idarəetmə elementləri də (VBA idarəetmə elementlərindən fərqli olanları) vardır: obyektin sərbəst və birləşmiş çərçivələri, səhifələrin parçalanması, tabe olan formalar/hesabatlar v.s. **Toolbox**-un yuxarı sol tərəfində **Control Wizards** (idarəetmə elementləri bələdçisi) adlı düymə var, şəkl. 13.4. Əgər həmin **Control Wizards** adlı düymə basılı vəziyyətdə olsa, onda adı olan (oxucuya tanış olan) idarə elementlərini formaya əlavə etdikdə bələdçinin (ingiliscə “Wizard”) uyğun olan pəncərəsi əmələ gələcək və istifadəçiyə lazım olan VBA kodu həmin idarəetmə elementi üçün avtomatik rejimdə generasiya ediləcək.



Şəkil 13.4 **Toolbox**-un **Control Wizards** (idarə etmə elementləri bələdçisi) düyməsi.

Həmin generasiya edilmiş VBA kodunu isə sonradan başqa proqramlarda makrorekorderin əvəzləyicisi kimi istifadə etmək olar. Qeyd edək ki, digər MS Office-in digər proqramlarından fərqli olaraq, Access-də makrorekorder yoxdur: bu əməliyyatı yalnız bir məqsədlə etmək olar – bu və ya digər əməlin necə yerinə yetirilməsini anlamaq üçün.

İndi isə **Toolbox**-un qeyri standart olan (oxucuya tanış olmadığı, yəni VBA idarəetmə elementlərindən fərqli olan) idarəetmə elementləri ilə tanış olaq:



- **Unbound Object Frame** (Obyektin sərbəst çərçivəsi) – Access verilənlər bazasına (cədvəl ilə yox!) və ya Access verilənlər bazasına nisbətən xaricdə olan faylda OLE obyektinin (məsələn, Word sənədini, Excel səhifəsi/diaqramını, PowerPoint təqdimatını, rəqəmsal şəkl, səs yazısını və ya videoklipi) formada yerləşdirmək imkanını yaradır.



- **Bound Object Frame** (obyektin birləşdirilmiş çərçivəsi) – yuxarıdakı funksionallığı tam saxlayır, yalnız bir fərqlə: istifadə edilən OLE obyektlerini Access verilənlər bazasındakı (və ya xaricdəki verilənlər mənbəyində olan) cədvəllərdə saxlanmasını təmin edir. Word hesabatlarının generasiyası üçün bu üsul əvəzəlməzdir. Məsələn, tutaq ki, Access verilənlər bazasında üç sütunlu cədvəl vardır, bax şəkl. 13.5. Burada, **File** sütununda, avtomatik olaraq, hesabatları generasiya edən, Word şablonları yerləşir. Sonra forma üzərinə **WordTemplate** adlı **Bound Object Frame** (obyektin birləşdirilmiş çərçivəsi) yerləşdirilməlidir. Artıq verilənlər bazasında olan şablon əsasında Word faylının yaradılması üçün hər şey vardır.

	Field Name	Data Type	
	Num	Number	
	Description	Text	
	File	OLE Object	

Şəkil 13.5. Word şablonlarını özündə saxlayan cədvəl.

Həmin Word hesabatının avtomatik generasiyasının icra edilməsi üçün, basıldıqda icraya start verən, formadakı düymənin VBA kod proqramı aşağıdakı nümunədə verilib:

```
'oFrame adlı istinadın formadakı Obyektin Birləşdirilmiş Çərçivəsi
'obyekti üçün alınması
Dim oFrame As BoundObjectFrame
    Set oFrame=oForm.Controls("WordTemplate")
'DLookup() metodu ilə onun içərisinə Templates cədvəlinin
'File sütunundakı qiyməti yükləyirik, nəzərə alınmalıdır ki,
'Templates cədvəlində sətirin nömrəsi (Num sütununun qiyməti)
'bərabərdir 1
'*****
oFrame=Application.DLookup("[File]","Templates","[Num]=1")
'*****
'Obyekti proqramın ayrıca pəncərəsində açırıq - yeni Word 'sənədini,
formadakı obyekt çərçivəsinə yüklənmiş olan 'şablon əsasında
yaradıırıq
'*****
    oFrame.Verb=acOLEVerbOpen
'*****
'Proqram obyektini aktivləşdiririk
'*****
    oFrame.Action=acOLEActivate
'*****
'Word-də oWord dəyişəninə istinad alırıq
'*****
Dim oWord As Word.Application
    Set oWord=GetObject(,"Word.Application")
```



```
'*****
'Yratdığımız sənədə istinad alırıq
'*****
Dim oDoc As Word.Document
Set oDoc=oWord.ActiveDocument
'*****
'Sonra Word vasitələri işləmək lazımdır, məsələn, manşırlama
'ile işarə edilən yerlərə lazım olan mətni yerləşdiririk.
```

Əlbəttə, həmin əməllər yerinə yetirildikdə obyektin birləşdirilmiş çərçivəsi əvvəlcədən görünməz edilməsi daha düz olardı. Çünki bu halda istifadəçi onu öz təşəbbüsü ilə aktivləşdirə bilməz.



- - **Page Break** (Səhifənin parçalanmasının idarəetmə elementi) – formanın yeni ekranının başlanğıcını təyin edir.



- - **Subform/Subreport** (Tabə olan forma/hesabat) – formada tabə olan formalar, cədvəllər və hesabatların yerləşdirilməsində istifadə edilir.

Əvvəl dediyimiz kimi, proqram səviyyəsində Access-də formaları çox nadir hallarda yaradırlar. Əgər formada idarəetmə elementlərin dəyişən yığımı yerləşdirilməlidirsə, onda daha düzgün - başlanğıcdan bütün idarəetmə elementləri yaradılmalıdır və, lazım olduqda, tədricən, onlar görünən edilməlidir. Bu və ya digər halda proqram səviyyəsində də forma üzərində idarəetmə elementlərini yaratmaq mümkündür. Həmin əməliyyat `Application.CreateControl()` metodu ilə icra edilir (metod çox sayda parametrlə qəbul edir: üzərində idarəetmə elementləri yaradılan formanın adını, idarəetmə elementinin tipini, formada idarə elementinin yerini v.s).

Formadakı idarəetmə elementlərinin qiymətlərinə edilən müraciət, idarəetmə elementlərinin adları ilə işləməyi bacaran, `Controls` kolleksiyasından icra edilir, məsələn, aşağıdakı VBA kod nümunəsində olan kimi:

```
cSumInNumber=oForm.Controls("SumNumber").Value
```

13.6 Formaların xassələri, metodları və hadisələri

Access . Form obyektini, onun xassələri, metodları və hadisələri

Aşağıda `Form` obyektinin VBA Access proqramlaşdırmasında ən vacib rol oynayan xassələri, metodları və hadisələri haqqında olan izahlı məlumatları veririk.

`Form` obyektinin ən vacib xassələri bunlardır:

- **ActiveControl** – hal-hazırda formanın hansı idarəetmə elementinin aktiv olmasını təyin edir (adətən yoxlamalarda istifadə edilir). Bu xassə `Report` (hesabat) obyektində və xüsusi

olan **Screen** obyektində də mövcuddur. **Screen.ActiveControl** xassəsi ilə hal-hazırda aktiv olan idarə elementinin adından əlavə, onun hansı formaya və ya hesabata aid olduğunu da təyin etmək olur. Məsələn:

MsgBox Screen.ActiveControl.Parent.Name

- **After...** - bu xassə toplusu ilə adi hadisəvi proseduranın adını hansısa hadisəyə təyin edib, proseduranı başqası ilə əvəz etmək olur.
- **AllowAdditions** – cari formada yeni yazının istifadəçi tərəfindən əlavə edilməsinə icazə və ya qadağan verilməsi. Analoji olaraq, yazıların silinib/silinməməsi **AllowDeletions** xassəsi təyin edir. Redaktəyə isə **AllowEdits** xassəsi ilə təyin edilir. **Allow...** prefiksli digər xassələr isə istifadəçi üçün müxtəlif informasiya təsvirinin icazə alınmasına aiddir.
- **AutoCenter** – formanın əlavə pəncərəsinin dəqiq mərkəzində yerləşməsinə təyin edir.
- **AutoSize** – yazılardan asılı olaraq, formanın ölçülərinin avtomatik dəyişdirilməsini təyin edir. Bu xassədən çox ehtiyatla istifadə etmək lazımdır – çünki avtomatik ölçülər dəyişəndə, tərtibatçı əvvəlcə nəzərdə tutduğunu görməyə də bilər.
- **Before...** - bu prefiksli xassələr (həmçinin **After...** prefiksli xassələr) formaların hadisəvi prosedurlarını əvəz edirlər.
- **Bookmark** – çox vacib xassədir: ADO-da **Recordset** obyektinin eyni adlı xassəsi kimi işləyir (9-cu fəsilə bax). Xassənin köməyi ilə cari yazı haqqında informasiyanı sətir dəyişəndə yadda saxlamaq olur (yazıya işarələmə qoymaqla).
- **BorderStyle** – formadakı çərçivənin təyin edilməsində kömək edir (bir başa, iç-içə qoyulmuş obyektlərsiz).
- **Caption** – xassəsi sadəcə formanın başlığını təyin edir.
- **ChartSpace** – öz tərkibində 64 sayda diaqram saxlaya bilən xüsusi konteyner obyektidir. Proqram səviyyəsində formalar üzərində diaqramların yaradılması və dəyişdirilməsi eməllərində istifadə edilir.
- **CloseButton** – proqram səviyyəsində bir başa formanın xassəsindən istifadəçi üçün **Close** düyməsinin lazım olduqda görünən/görünməz olmağında istifadə edilir (istifadəçinin hansısa əməlin, məsələn, verilənlərin daxil edilməsini, qəflətən dayandırmasının qarşısını almaq üçün istifadə edirlər).
- **ControlBox** – bu xassə **Control** menyusunun (forma başlığının ən yuxarı sol tərəfində yerləşən və **Restore**, **Move**, **Maximize**, **Minimize** və **Close** əməlləri tərkibində olan nişancıq) əlavə edilib edilməməsinin təyin edilməsində istifadə olunur. Əgər o götürülsə, onda formanın başlığının sağ tərəfində olan **Restore**, **Move**, **Maximize**, **Minimize** və **Close** şəkildələri də dərhal yox olacaq. Adətən bu xassə xüsusi olan modal formalarda tətbiq edilir, məsələn, müxtəlif xəbərdarlıq pəncərələrinin rolunu ifa edən formalarda. Təəssüflər olsun ki,

bu xassənin sazlanması yalnız formanın dizayner rejimində mümkündür. Buna oxşar **MinMaxButtons** xassəsi işləyir: formada **Minimize** və ya **Maximize** düymələrinin olub/olmamasını təyin edir (dizayner rejimində sazlanması əlçatan olur).

- **Controls** – olduqca vacib xassədir: cari formada olan bütün idarəetmə elementləri ilə birgə eyni adlı **Controls** kolleksiyasını qaytarır.
- **Count** – formadakı idarəetmə elementlərinin sayını qaytarır. Daha çox formaya yox, **Controls** kolleksiyasına aiddir.
- **CurrentRecord** – bütün formada açıq olan yazılar içərisindən cari yazının nömrəsini qaytarır: xassə yalnız oxumaq üçün əlçatandır.
- **CurrentView** – çox vacib xassədir: cari formanın hansı rejimdə açılmasını təyin edir (dizayner, forma və ya cədvəl rejimində). Xassə yalnız oxumaq üçün əlçatandır. Rejimlər arasında keçidin icra edilməsi, uyğun olan parametrləri ilə, **DoCmd.Close()** və **DoCmd.OpenForm()** metodları ilə mümkündür.
- **Cycle** – bu xassə istifadəçinin keçid sırasına görə son yazıda olduqda **<Tab>** düyməsinin basıldıqda nə baş verdiyini təyin edir: növbəti yazının (standart halda) birinci sayda gedən idarəetmə elementinəmi keçəcək və ya cari yazının birinci sayda edən idarəetmə elementinə qayıdacaq yaxud da formanın (ekranın) cari səhifəsi çərçivəsində çevrə ilə hərəkət edəcək.
- **DataEntry** – bu xassə ilə istifadəçi verilənlərin daxil edilməsində heç bir mövcud olan yazıları görmür. Beləliklə, istifadəçi üçün formada yalnız bir yazı əlçatan olur – boş olan yazı. Bununla istifadəçi yalnız yeni verilənləri formaya daxil etmiş olur (bəzən praktiki işlərdə bu cür rejimin qurulmasına böyük ehtiyac yaranır).
- **DataSheet...** - bu cür prefiksli xassələrlə forma **DataSheet View** rejimində (yəni cədvəl rejimində) olduqda, formanın xarici görüntüsünü təyin etmək olur.
- **DefaultControl** – proqram səviyyəsində forma yaradılarda VBA kodunun sətirlərini, kəskin olaraq, az olmasına imkan verir. Bu halda tərtibatçı istənilən tipli **DefaultControl** virtual idarəetmə elementinin parametrlərini təyin edə bilər. Eyni tipli idarəetmə elementi yaradılarda, ona, avtomatik olaraq, **DefaultControl** üçün təyin edilən sazlamalar tətbiq olacaq. Bu imkan çox əlverişli mühit yarada bilər: məsələn, əgər formada proqram səviyyəsində bir neçə onluq sayında eyni şriftli fonları eyni rəngli v.s., mətn sahələri yaradılmasına böyük ehtiyac var:

```
'Standart hala görə mətn sahəsini yaradırıq
Set oDefaultTextBox=oForm.DefaultControl(acTextBox )
    oDefaultTextBox.FontSize=12
'Foirmada yeni mətn sahəsini yaradırıq - artıq 12 ölçülü 'şriftlə
Set oTextBox1=CreateControl(oForm.Name,acTextBox,500,500)
```

- **DefaultView** – standart halda formanın hansı rejimdə açılmasını təyin edir. İstifadəçi başqa rejimə də keçə bilər, əgər **ViewsAllowed** xassəsi icazə versə.
- **Dirty** - əgər cari yazı hələ verilənlər bazasında, yaddaşda saxlanmayaraq, dəyişdirilibsə, onda **True** qiymətini qəbul edir. Cari yazı, yaddaşda saxlanan andan başlayaraq, **False** qiymətini alır. Adətən istifadəçinin daxil etdiyi verilənləri saxlamaq üçün yoxlamalarda istifadə edilir..
- **DividingLines** — formada ayran xətlərin olmasını təyin edir (formada olan yazıları və formanın sahələrini biri-birilə ayıran xətlər).
- **FetchDefaults** – yeni yazını dolduranda standart halda sütunlarda qiymətlərin çıxarılmasını təyin edir.
- **Filter** – olduqca vacib xassədir. O, formadakı yazıların filtrasiya edilməsini icra edir (məsələn, istifadəçi seçdiyi tarix diapazonuna görə, hansısa sütunun istifadəçi tərəfindən aşağı açılan siyahıdakı qiymətin təyin edilməsi). Xassəyə ötürülən qiymət ifadə şəklində olur: SQL sorğularında **WHERE** açar sözündən sonra gəlir. Bu xassə ilə birgə filtri işə salmaq üçün **FilterOn** xassəsi də istifadə edilir. Məsələn, əgər forma cədvələ *City* sözü ilə bağlanıbsa və formanın həmin sütunda yalnız “Bakı” sözü ilə bağlı yazıların əks edilməsi lazımdırsa, onda aşağıdakı VBA kod nümunəsi istifadə edilə bilər:

```
oForm.Filter="City='Bakı'"
oForm.FilterOn=True
```

Nəzərə alınmalıdır ki, **Filter** xassəsi yalnız formada olan yazıların filtrasiyası üçün tətbiq edilə bilər. Əgər forma ilə SQL Server və ya Oracle verilənlər bazasına müraciət edilməldirsə, onda filtri serverə ötürülən sorğuda tətbiq etmək daha yaxşı nəticə verir. Bununla proqrama lazım olmayan verilənlərin yüklənməsinin qarşısı alınır: **ServerFilter** xassəsi tətbiq edilir (aşağıda axtar).

- **Form** – xassəsi, **Subform/Subreport** idarəetmə elementi tərkibində olan formaya və ya hesabatla istinad alınmasını təmin edir.
- **FrozenColumns** – “bərkişdirilmiş” sütunların (yəni formanı sağ/sol tərəfə diyirlətdikdə ekrandan kənar getməyən sütunların) sayını təyin edir. Bu xassə yalnız oxumaq üçün əlçatandır. Formada parametrləri sazlamaq üçün Access-in qrafik interfeysində **Format→Freeze Columns** menyusu nəzərdə tutulub.
- **HasModule** – bu xassə ilə cari forma və ya hesabatda öz sinif modulunun olması təyin edilir (standart halda – yox). İstifadəçi sinif modulu ilə formaya özəl xassələr və metodlar əlavə edə bilər və ya mövcud olanları yenidən təyin edə bilər. Adətən bu cür imkan (ümmümlükdə VBA-da istifadəçi siniflər ilə iş) çox mürəkkəb strukturu olan və spesifik tələbatları olan layihələrdə istifadə edilir.

- **HelpFile** və **HelpContextId** – cari forma üçün sorğu faylını və ondakı nişanlamaları təyin edir.
- **Hwnd** – cari forma üçün Windows pəncərəsinin deskriptorunun qayıtmasını təmin edir. Windows API ilə işlədikdə tətbiq edilir.
- **InputParameters** – verilənlər mənbəyinə sorğular icra edildikdə, sorğuların ötürülmüş parametrləri haqqında olan və nəticələri formaya yüklənmiş, informasiyanı sətir qiyməti şəklində qaytarılmasına imkan yaradır.
- **KeyPreview** – bu xassə basılmış düymənin nəyə verilməsini (hansı əməlin yerinə yetirilməsinə) təyin edir – yalnız formadakı aktiv idarəetmə elementinə (standart halda), yoxsa əvvəlcə formaya sonra isə aktiv idarəetmə elementinə. Bu xassə ilə formaların hadisəvi proseduralarını emal etmək olar, məsələn, aşağıdakı VBA kodu nümunəsində olan kimi:

```

Private Sub Form_Load()
    Me.KeyPreview=True
End Sub
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
    Select Case KeyCode
        Case vbKeyF5
            MsgBox "Basılı klaviş F5!"
        Case vbKeyF6
            MsgBox "Basılı klaviş F6!"
    End Select
End Sub

```

- **MenuBar** - cari forma və ya hesabat açıldıqda avtomatik əmələ gələn istifadəçi menyusunun adını göstərməyə imkan yaradır. Həmçinin **ShortcutMenuBar** və **ToolBar** xassələrinə bax.
- **Moda** – çox vacib xassədir. Əgər xassədə **True** qiyməti qurulsa, onda forma modal şəklini alacaq. Bu deməkdir ki, başqa forma aktivləşənə qədər cari forma bağlanmalıdır. Əgər, forma aktiv olsa, onda daha bir **Popup** xassəsində **True** qiyməti təyin edilməlidir.
- **Module** – Bu xassə formanın sinif modulunun obyektinə istinad alır (bax **HasModule** xassəsinə). Xassə ilə olduqca ehtiyatlı olmaq lazımdır: dizayner rejimində ona müraciət etdikdə, avtomatik olaraq, forma üçün yeni sinif modulu yaranır.
- **Moveable** – formanı ekranda hərəkət etdirməyə imkan yaradır. Formanın yerdəyişməsi proqram dəyişməsinə gətirib çıxarmır: çünki proqram səviyyəsində hər vaxt formanın yerini dəyişmək mümkündür.
- **Name** – formanın adını təmsil edir.
- **NavigationButtons** – yazılar üzrə istifadəçiyə keçidləri icra etmək üçün düymələrin verilməsini təmin edir (formanın aşağı hissəsində yerləşən düymələrlə yazılar üzrə birinciyə, sonuncuya, yenisinə, növbəti yazıya v.s. əməlləri yerinə yetirməyi təmin edir).

- **NewRecord** – forma üçün cari yazının yeni olmasını təmin edir: yoxlamalarda istifadə edilir.
- **ObjectPalette** və **PaintPalette** – formanın özü üçün və formada yaradılan elementlər üçün Windows-un uyğun olan rəngləmə sxemini yaradır. Windows-dan fərqli olaraq, Access eyni zamanda məhdudiyyətsiz sayda rəngləmə sxemlərini yarada bilər. Həmin rəng sxemi forma və ya hesabatla birgə yaddaşda saxlanacaq. **PaletteSource** xassəsi forma və hesabat üçün yaradılmış rəngləmə sxemini diskdəki fayl şəklində göstərilməsini təmin edir.
- **On...** - prefiksli xassələr (**After...** və **Before...** kimi) formada hadisəvi proseduraları əvəz edir: üstəlik hadisəyə Access makrosunun və ya VBA prosedurasının adını təyin edir.
- **OnTimer** – xassəsi müəyyən zaman intervalında (**TimeInterval** xassəsi ilə təyin edilir) Access-in, avtomatik generasiya etdiyi hadisələrə reaksiyanı təyin edir
- **OpenArgs** – xassəsi **DoCmd.OpenForm()** metodu ilə açılan formaya ötürülən parametri əlçatan edir.
- **OrderBy** – formada yazıların sortlaşmasını sazlayan xassədir. Qəbul etdiyi sətir qiyməti sortlaşma aparılan sütunun adından və sortlaşmanın istiqamətindən ibarətdir. Məsələn:


```
oForm.OrderBy="City DESC"
oForm.OrderByOn=True
```
- **OrderByOn** - sortlaşmanın bilavasitə işə salınmasını icra edir.
- **Page** – formada və ya hesabatda cari səhifənin çap edilməsində səhifənin nömrəsini qaytarır. Adətən mətn sahələrində səhifələrin avtomatik generasiyasında istifadə edilir. Çapdan əvvəl və ya çap zamanı yalnız əvvəlcədən baxma üçün əlçatan olur. Onunla birgə **Pages** xassəsi istifadə edilir (formada və ya hesabatda səhifələrin ümumi sayını qaytarır).
- **Painting** – cari formanın yenidən çəkilməsini qadağan edir: adətən **False** qiyməti ilə qurulur (uzun zamanlı resurs tutumlu işlərdə resursları qorumaq üçün). Lakin əməliyyat bitən kimi xassəni yenidən **True** qiymətində qurmaq lazımdır. Access-in bütün pəncərələrinin yenidən çəkilməsinə qadağan qoymaq üçün **Echo** makrokomandası tətbiq edilə bilər.
- **Picture...** - prefiksli xassələr formaya görüntünü yerləşdirməyə (adətən fon rejimində) və görüntünün parametrlərinin qurmağına imkan verir.
- **PivotTable** – xassəsi **PivotTable** obyektini Access-də əlçatan edir (bu xassə yekun cədvəlin Web-versiyasıdır (ActiveX komponenti), Excel ilə bağlı fəsildə bu haqda danışmışıq).
- **Printer** – çox hallarda forma, hesabat və ya bütün proqram üçün printerin standart hala görə sazlaşmağa və ya bu haqda informasiya almağa imkan yaradır. Standart halda printerin istifadə edilməsini **UseDefaultPrinter** xassəsi ilə təyin etmək olar.

- **Properties** – forma və ya hesabatın bütün qurulmuş olan xassələrini təmsil edən obyektlər kolleksiyasını qaytarır. Həmin obyektlərin iki xassəsi olur: **Name** – onun adını təyin edir və **Value** – onun qiymətini. Adətən test mərhələsində bütün forma və hesabatlar üzrə yoxlamalar aparıldıqda istifadə edilir.
- **Prt...** - bu prefiksli xassələr forma və ya hesabatın çapdan əvvəl printerin sazlanmasında istifadə edilir. Qəbul edilən parametrlər kimi ona mürəkkəb bayt strukturlu qiymətlər ötürülür. Bu xassədən yalnız printer drayveri haqqında mükəmməl informasiya olduğu halda istifadə edilməsi məsləhət görülür.
- **RecordLocks** – bir neçə sayda istifadəçi tərəfindən eyni zamanda cari forma redaktə ediləsi olduqda, formadakı yazıların bloklaşdırılması hadisəsini təyin edir: **0** – yazılar bloklaşdırılmır (bu halda istifadəçi yaddaşda saxlamaq istəsə dərhal səhv haqqında xəbər alacaq); **1** – formanın bütün istinad etdiyi yazılar bloklaşır, başqa istifadəçilər formaya yalnız baxa bilər; **2** – yalnız dəyişdirilən səhifə bloklaşacaq (verilənlər bazasında təxminən 4 Kbayt təşkil edir), başqa istifadəçilər həmin səhifəni yalnız oxuya biləcəkdir.
- **RecordSelectors** – xassə **Form view** rejimində yazının seçilmə sahəsinin əlçatan olmasını təyin edir: bütün yazının seçilməsini təmin edən formanın sol tərəfindəki dördbucaqdır)
- **Recordset** – xassəsi, forma haqqında informasiya mənbəyi olan, **ADO Recordset** obyektini (9-cu fəsil) qaytarmağa və ya sazlamağa imkan verir. Başqa xarici verilənlər mənbəyinə müraciət etdikdə xassənin istifadə edilməsi çox rahat mühit yada bilər. Bundan əlavə, oxucuya artıq yaxşı tanış olan, **Recordset** obyektini (9-cu fəsilə bax) öz rahat və əlverişli xassə və metodları ilə əmələ gəlir.
- **RecordsetType** – formanın əsaslandığı **Recordset** obyektinin tipini təyin edir (onun informasiyasını redaktə etmək məqsədi ilə).
- **RecordSource** – forma və ya hesabat üçün verilənlər mənbəyini təyin etməkdə kömək edir. Mətn qiymətini (cədvəlin və ya SQL dilində sorğusunun adını) qəbul edir. **Recordset** xassəsi dəyişdirildikdə, xassənin qiyməti, avtomatik olaraq, dəyişir.
- **RecordSourceQualifier** – formanın istinad etdiyi cədvəlin adını qaytarır: Access formasının istifadə edilməsi yalnız bir halda ola bilər: verilənlər mənbəyi SQL Server olduqda.
- **ResyncCommand** – istifadəçi tərəfindən forma vasitəsi ilə yazılarda dəyişiklik edildikdə, verilənlər mənbəyinə gedən komandanın proqram səviyyəsində təyin edilməsinə xidmət edir. Bu xassə yalnız o halda istifadə edilir ki, verilənlərin dəyişdirilməsi üçün avtomatik generasiya edilən komanda hansısa səbəbə görə istifadəçini təmin etmir.
- **ScrollBars** – formanın üfüqi və şaquli diyirlənmə zolağının əks edilməsini təyin edir (standart halda həmin zolaqlar göstərilmir).

- **Section** – bu xassə metoda daha çox oxşayır: formanın və ya hesabatın ərazisini (seksiyasını) təmsil edən parametri qəbul edir və həmin ərazinin xassələrinin sazlanmasında istifadə edilir, məsələn:

```
oForm.Section(acPageHeader).Visible=False
```

- **ServerFilter** – forma açıldıqda, verilənlər mənbəyinə (məsələn, SQL Server və ya Oracle) filtrin sorğuya qoyulmasına imkan yaradır: Access, filtr kimi təklif edilən, sətir ifadənin sintaksisini yoxlamır. **ServerFilterByForm** xassəsinin qiymətini **True** qiyməti ilə qursalar, onda server filtri işə salınacaq:

```
oForm.ServerFilter="City='Gəncə'"
oForm.ServerFilterByForm=True
```

Yuxarıdakı kodu formanın hadisəvi prosedurası **OnOpen** yerləşdirməsi daha yaxşı effekt verə bilər.

- **ShortcutMenu** – xassəsi **False** qiyməti ilə qurulsa, onda forma və həmçinin onun üstündəki idarə elementləri üçün kontekst menyusu göstərilməyəcək.
- **ShortcutMenuBar** – forma və hesabat üçün mausun sağ düyməsinin şıqqıltısı ilə istifadəçi kontekst menyusunun göstərilməsini təmin edir (əslində bu xassə bütün idarəetmə elementləri üçün nəzərdə tutulubdur).
- **Tag** – forma üçün “*teq*” rolunu oynayır (İngiliscə *Tag* sözü deskriptor sözünün jarqonudur, deskriptor isə daxili strukturda (fayllar v.s.) verilənlərin axtarılmasını təmin edən identifikator, açar söz və ya nişanlama mənasına gəlir). Bu xassə istifadəçi atributuna malikdir və yalnız proqramdan əlçatan olur və başqa heç nəyə təsir etmir. Adətən xidməti informasiyanın (məsələn, nişanlamaların) saxlanılmasında istifadə edilir. Maksimal uzunluğu 2048 simvolla olan sətir qiymətini qəbul edir.
- **TimerInterval** – formanın taymerinin (standart halda keçirilmiş olur) milli saniyə ölçüsündə qiymətini qurmağa imkan yaradır. Adətən formanın hadisəvi prosedurası **Load**-da sazlanır. Daha çox **Timer** hadisəsi üçün hadisəvi prosedura ilə birgə istifadə edilir. Gözləmələrin təşkilində (məsələn, hansısa xaricdə olan proqramın işinin tamamlanmasını gözlədikdə, onun vəziyyəti hər yarım saniyə yoxlanması lazımdırsa), formanın animasiyasında v.s. istifadə edilir.
- **Toolbar** – forma və ya hesabat açıq olduqda hər dəfə alətlər panelinin təyin edilməsinə imkan verir. Bu xassədən daha çox özəl istifadəçi alətlər panelindən istifadə edirlər.
- **UniqueTable** – forma hansısa başqa forma ilə və ya saxlanılan prosedura ilə bağlı olduqda bu xassə istifadə edilir. İş burasındadır ki, bağlanan prosedura və formalar eyni zamanda bir neçə cədvələ istinad edə bilər. Nəticədə konfliktlərin yaranması qaçılmaz olur. Həmin konfliktlər yarananda xassə dəyişikliklərin icra edilməsini təyin edir – olduqca vacib xassədir.

- **ViewsAllowed** – istifadəçinin forma ilə hansı rejimdə işləməsini təyin edir (yalnız **Form view** rejimindəmi yalnız **DataSheetView** rejimindəmi və yaxud da (standart halda) hər iki rejimdə). Dizayner rejimi istifadəçi üçün hər an əlçatan olur. Bu rejimi yalnız icazələrlə qadağan etmək olar.
- **Visible** – formanı, lazım olduqda, görünən və görünməz etməyə imkan yaradır.
- **Width** – xassəsi formanın genişliyini təyin edir (standart halda ölçü vahidi tipi ilə verilir – 1 ingilis düymünün 1/1440 hissəsidir).
- **WindowHeight, WindowLeft, WindowTop, WindowWidth** - xassələri formanın pəncərəsinin ölçülərini təyin edir.

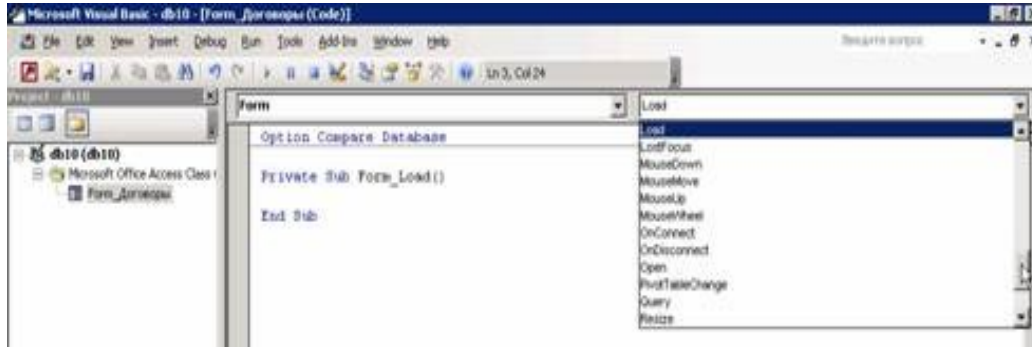
Form obyektinin metodları xassələrlə müqayisədə olduqca azdır. Bir çox halda onların təyinatını bilavasitə adından anlamaq mümkündür:

- **GoTo ()** – çoxsaylı ekran formasında lazımı səhifəyə keçməyə imkan yaradır;
- **Move ()** – formanın ekranda yerdəyişməsini təmin edir;
- **Recalc ()** – formanın hesablanan idarəetmə elementlərinin qiymətlərinin sayılmasını icra edir. Forma aktivdirsə, onda qrafik ekranda sadəcə <F9> düyməsi basıla bilər.
- **Refresh ()** – formadakı verilənlər toplusunda olan dəyişikliklərin əks edilməsində kömək edir. Əgər verilənlər bazasından təkrarən verilənləri yükləmək lazımdırsa (məsələn, onlar başqa istifadəçi tərəfindən artıq dəyişdirilib), onda **Requery ()** metodu tətbiq edilməlidir.
- **Repaint ()** – formanın yenidən çəkilməsini icra edir. Avtomatlaşdırılmış rejimdə bu metod **Painting** xassəsi ilə yenidən çəkilmə prosesində istifadə edilir.
- **SetFocus ()** - formada fokusu qurur (fokus - Maus və ya klaviatura ilə istifadəçinin daxil etdiyini təyin edən vəziyyətdir).
- **Undo ()** – proqram səviyyəsində istifadəçinin daxil etdiyi informasiyanın silinməsinə imkan yaradır (əgər həmin informasiyada səflik varsa). Nəzərə alınmalıdır ki, yeni yazıya keçdikdə, avtomatik olaraq, cari yazı yaddaşda saxlanılır, buna görə belə halda **Undo ()** metodu kömək edə bilməyəcək.

Forma üçün vacib olan hadisələr də nəzərdə tutulub və praktikada onlardan aktiv istifadə edirlər. Onlarla iki üsul ilə işləmək mümkündür:

- formanın xüsusi olan xassələri vasitəsi ilə (onların adı **Before...**, **After...**, **On...** v.s. prefiksləri ilə başlayır);
- adi üsul ilə - yeni hadisəvi proseduraların vasitəsi ilə.

Bu üsulları seçmək üçün **Project Explorer**-də lazımı formanı seçərək, <F7> düyməsini basmaq, sonra isə əmələ gələn pəncərədən lazım olan hadisəvi prosedura seçilməlidir (bax şəkl. 13.6).



Şəkil 13.6 Formanın hadisələr siyahısı.

13.7 VBA-dan hesabatlarla işləmə: Report obyektı

Access.Report obyektı, VBA vasitəsi ilə Access hesabatları ilə işin aparılması, təşkili: hesabatın açılması, çapı, idarəetmə elementləri və ərazilərin yaradılması və istifadə edilməsi

Access-də VBA proqramlaşdırılmasında daha çox istifadə edilən obyektlərdən birisi – hesabatı təmsil edən **Report** obyektidir. Access hesabatları ola bilsin ki, verilənlər bazasında avtomatik generasiya edilən hesabatlar sırasında (məsələn, **Crystal Reports**, **Microsoft Reporting Services** və ya artıq tanış olduğumuz **VBA/ADO/Word** və **VBA/ADO/Excel** bağlantıları kimi), ən sadə və effektiv üsuldur. Access hesabatları ilə yalnız Access-in öz verilənlər bazası üçün yox, hətta xarici verilənlər mənbəyi üçün də (məsələn, **SQL Server** və ya **Oracle** verilənlər bazası üçün) hesabatların avtomatlaşdırılmış icrasını yerinə yetirmək mümkündür. Bu halda həmin hesabatların əlavə funksional imkanları məhz VBA imkanları ilə təmin ola bilər (məsələn, şərti formatlaşma kimi əməllərin yerinə yetirilməsində).

VBA proqramlaşdırması nöqtəyi nəzərdən hesabatlarla aparılan işlər, yuxarıda tanış olduğumuz, formalarla aparılan işə çox bənzəyir. Məsələn, burada da, eyniliklə formalarda olan kimi, bütün hesabatların əlçatan edilməsini **Application.CurrentProject.AllReports** kolleksiyası ilə (burada isə **AccessObject** obyektləri yerləşir) yerinə yetirmək olur. Bütün açıq olan hesabatların əlçatan edilməsi isə, tərkibində ənənəvi **Report** obyektı olan, **Reports** kolleksiyası ilə icra edilir. Nəzərə alınmalıdır ki, hesabatların tərtib edilməsində çox nadir hallarda proqram səviyyəsində işlərin aparılmasında rast gəlmir. Çünki Microsoft şirkəti hesabatların avtomatik generasiyası üçün də adi istifadəçi (yəni proqramlaşdırmanı bilməyən istifadəçi) üçün olduqca rahat mühit yaratmışdır. Əksər hallarda Access-in qrafik interfeysində hesabatların avtomatik generasiyası verilənlər bazası pəncərəsinin **Reports** qurmasından icra edilir. Lakin bununla belə, hesabatların VBA proqramlaşdırılması ilə tərtib edilməsi (və ya iş aparılması) formalarla müqayisədə daha çox rast gəlir. Miqyası böyük olan şirkət və müəssisələrdə hesabatların yaradılması və istismarına xüsusi tələblər qoyulursa, onda VBA proqramlaşdırması həqiqətən də yeganə və ən effektiv üsul kimi meydana çıxır.

Proqram səviyyəsində hesabatı `Application.CreateReport()` metodu ilə yaratmaq olur:

```
Dim oReport As Report
Set oReport=Application.CreateReport()
```

Bu halda, həmin VBA kodu əsasında yaradılan hesabat maşının operativ yaddaşında olacaq. Oradan isə onu yaradan VBA prosedurası işini tamamlayan andan silinməyə başlayacaq. Yaradılmış hesabatı yaddaşda (diskdə Access faylının tərkibində) saxlamaq üçün həmin yuxarıdakı proqram koduna bir sətiri əlavə etmək lazımdır (bu halda, yaddaşda saxlanılan hesabatın adı, standart hala uyğun olaraq, `Report1`, `Report2` v.s. kimi olacaq):

```
DoCmd.Close, acSaveYes
```

Ümumiyyətlə yaddaşda saxlanılan hesabatlarla özəl adların verilməsi praktikada vacib amil kimi meydana çıxır. Bu məqsədlə, `DoCmd.Close()` metodunun çağırılmasından əvvəl başqa metodun çağırılmasını proqramda təşkil etmək lazımdır - `DoCmd.Save()` metodunu:

```
DoCmd.Save, "Satış_haqqında_hesabat"
DoCmd.Close
```

Bu halda `CreateReport()` metodu boş hesabat yaradacaq (əgər parametr kimi biz ona artıq idarəetmə elementləri olan hesabat şablonunu ötürməmişiksə). Hesabatda formalarda mövcud olan idarəetmə elementləri də istifadə edilə bilər. Nəzərə alınmalıdır ki, idarəetmə elementlərini hesabatda yerləşdirməkdən əvvəl, onların dəqiq harada yerləşdirilməsini təyin etmək vacibdir.

Access hesabat doqquz sayda ərazidən ibarət ola bilər (onlardan, standart halda, yalnız üçü görünən olur):

- *yuxarı kolontitul (Page Header)* - səhifənin yuxarı kolontitulu kimi anlamaq lazımdır. Hesabatda hansı sayda yazı varsa, o qədər də bu element təkrarlanacaq. Adətən onun tərkibinə səhifənin nömrəsini, hesabatın özü haqqında əsas olan məlumatı, müəllif haqqında məlumatı v.s. əlavə edirlər;
- *verilənlər ərazisi (Details)* - bəzən ona "details" yəni "ətraflı məlumat" deyirlər. Adətən buraya verilənlər bazasında olan yazılar çıxarılır. Bu ərazi də verilənlər bazasında olan yazıların sayı qədər təkrarlanır;
- *aşağı kolontitul (Page Footer)* – adətən yuxarı kolontitul hansı məqsədlə istifadə edilirsə, bundan da eyni məqsədlərə görə istifadə edirlər.

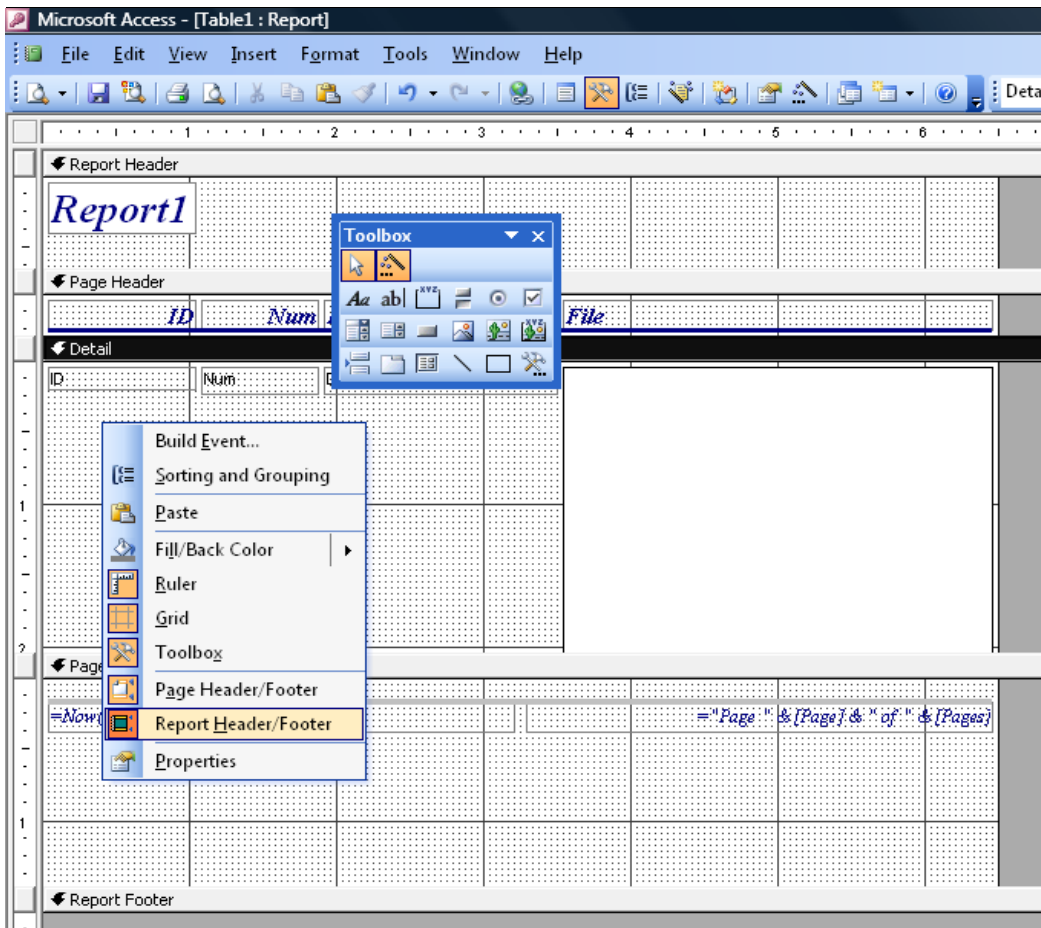
Əgər mausun sağ düyməsi ilə hesabatın boş yerində və sonra əmələ gələn kontekst menyusunda **Reports Header/Footer** bir dəfə şıqqıltı etdikdə, onda hesabatdan əlavə iki sayda ərazisi də görünəcək, bax şəkl. 13.7:

- *hesabatın başlığı (Report Header)* – hesabatın önündə gedən və heç bir yerdə bir daha təkrar edilməyən xüsusi ərazidir. Adətən bura informasiya haqqında olan yekun məlumat yazılır: məsələn, müəllif, səhifələrin sayı, nəticələr, verilənləri təmsil edən diaqramlar v.s. haqqında qısa məlumat yerləşdirilir.

- **hesabatın qeydləri (Report Footer)** – hesabatın sonunda (aşağı hissəsində) yerləşdirilir və həmçinin təkrar olunmur. Bu ərazi də hesabat başlığı kimi istifadə edilə bilər.

İdarə etmə elementinin hesabatda yeri təyin edildikdən sonra həmin elementlərin bilavasitə hesabatda proqram səviyyəsində yerləşdirmək üçün **Application.CreateReportControl()** metodundan istifadə etmək lazımdır. Bu metod parametr kimi hesabatın adını (hesabat hökmən açıq olmalıdır), idarəetmə elementinin tipini, hesabatda elementin yerləşdiriləsi ərazinin yerini, elementin hesabatda olan koordinatlarını və elementlə bağlanan sütunun adını qəbul edir. Məsələn, **Details** ərazisində heç bir obyektə bağlı olmayan, adı təyin edilmiş mətn sahəsinin yaradılması üçün aşağıdakı VBA kodu tətbiq edilə bilər:

```
Set txt1=CreateReportControl(oReport.Name,acTextBox,acDetail)
txt1.Name="txtCustomerID"
```



Şəkil 13.7 Dizayner pəncərəsində hesabatın hazırlanması üçün əlavə imkanların seçilməsinə imkan yaradan kontekst menyusu.

Əgər hesabat adı yolla (yəni yalnız qrafik interfeysin köməyi ilə, proqramlaşdırmasız) yaradılıbsa, onda çox güman ki, həmin hesabat proqram səviyyəsində açılacaq. Bu əməli **DoCmd.OpenReport()** metodu ilə həyata keçirməyi məsləhət görürlər. Burada bir incəlik var: əgər aşağıdakı VBA kodu, məsələn,

```
DoCmd.OpenReport "Satışlar_haqqında_hesabat"
```

yerinə yetirilsə, onda hesabat gözlənilən kimi, açılmaq əvəzinə çapa çıxarılacaq. Lakin həmin kodda aşağıdakını əlavə etsək

```
DoCmd.OpenReport "Satışlar_haqqında_hesabat", acViewPreview
```

onda fayl adı qaydada açılacaq.

İdarə etmə elementlərinə edilən müraciətlər formaya edilənlərdən fərqlənir – **Controls** kolleksiyasından istifadə etmək lazım gəlir.


Hesabatda olan **Report** obyektinin metodları və hadisələri **Form** obyektinin analoji hadisələrindən fərqlənir. Yeganə prinsipial fərq – **Report** obyektində şəkillərin çəkilməsi üçün metodlar nəzərdə tutulub: **Circle()** – dairənin (ellipsin) çəkilməsi; **Line()** – xəttin çəkilməsi; **PSet()** – ayrıca piksel üçün rəngin qurulması v.s. Proqram səviyyəsində mətn yazılarının çıxarılması üçün həmçinin müxtəlif metodlar və xassələr toplusundan istifadə etmək mümkündür. Qeyd etməliyik ki, proqram səviyyəsində şəkillərin çəkilməsi məhsuldarlığı qaldıran amil ola bilməz: çünki Access-in qrafik interfeysindəki dizayner rejimində həmin əməlləri daha effektiv və sadə üsullar ilə yerinə yetirmək mümkündür. Başqa bir tərəfdən, əgər dairələr, xətlər v.s. çəkmək əvəzinə **Image (görüntü)** idarəetmə elementi istifadə edilsə, onda imkanlar bir neçə qat artmış olacaq.

13.8 Access-in başqa obyektləri haqqında

Access.DataAccessPage, Access.GroupLevel, Access.Module, Access.References və Access.SmartTag obyektləri

Access-in obyekt modelində VBA proqramlaşdırmasında istifadə edilən digər obyektlər də var. Onların bu bölmədə seçilməməsinin yalnız bir səbəbi var: onlardan VBA proqramlaşdırmasında, əvvəl baxdığımız **Application, DoCmd, Form** və **Report** obyektləri ilə müqayisədə, nadir hallarda istifadə edirlər. Buna baxmayaraq, onlar haqqında, qısa da olsa, lakin vacib ola biləcək məlumatları aşağıda veririk.

Access-də çox əlverişli bir mühit var ki, onun haqqında bəzən təcrübəli proqramçılar və istifadəçilər bilmirlər. Bu imkanın adı budur: *verilənləri əlçatan edən səhifələr (Data Access Pages)*. İş burasındadır ki, verilənləri əlçatan edən səhifələrin tətbiqi - əslində verilənlər mənbəyinə informasiyanın ötürülməsi üçün ən sadə üsul - Web-formanın tətbiqidir. Fiziki baxımdan, həmin Web-forma **ActiveX** idarəetmə elementləri ilə yaradılır və müxtəlif əməllərin yerinə yetirilməsi üçün verilənlər mənbəyinə qoşulmaqdan ötrü tələb olan funksionallığı təmin edir. Formada,

VBScript-də  (bir qədər az imkanlı, VBA-nın yaxın “qohumu”) yazılmış kod əsasında, adi idarəetmə elementlərindən istifadə etmək olar: məsələn, düymələr, mətn sahələri v.s. üçün. *Verilənləri əlçatan edən səhifələrin* ən vacib üstünlüyü budur ki, onlar brauzer mühitində icra edilmirlər - **ActiveX** elementinin icra etmə mühitində işləyirlər. Məhz buna görə proqramçının sərəncamında Windows-un obyekt modelləri ilə işləmək imkanları qalır. Məsələn, **Click** hadisəsini

emal edən kodundan **ADO Recordset** obyektini yaradıla bilər və istifadəçi kompyuterində Word, Excel, yaxud Access işə salına bilər, yaxud da şəbəkə diskinə qoşulmaq mümkün olar.

Verilənləri əlçatan edən səhifələr müxtəlif yollarla yaradıla bilər:

- Access verilənlər bazasının obyektini kimi (bunun üçün verilənlər bazasının pəncərəsində **Pages** (yeni səhifələr) qurması nəzərdə tutulub);
- verilənlər bazasından kənarında – sadəcə HTML faylları kimi.

Bu imkanları reallaşdırmaq üçün Access-in qrafik interfeysində gerek **File**→**New** menyusu seçilsin. Sonra isə (ekranın sağ hissəsində) – **Blank Data Access Pages** obyektini (yeni boş olan verilənləri əlçatan edən səhifələr) əmələ gələn kontekst menyusundan seçilməlidir.

Access obyekt modelində *verilənləri əlçatan edən səhifələr* **DataAccessPage** obyektini ilə təmsil edilib. Həmin səhifələr əslində yuxarıda qeyd etdiyimiz kimi, Web-səhifədirlər. Buna görə onların xassə və metodlarının toplusu adi formaların xassə və metodlar toplusundan az sayda (kəsilmiş sayda) təmsil edilir. *Verilənləri əlçatan edən səhifələrin* Web-parametrlərini xüsusi olan **WebOptions** obyektini ilə təyin edirlər.

Formalar və hesabatlarda tez-tez şərti formatlamadan istifadə edilməsi tələb olunur: istənilən mətn sahəsinin və ya kombine edilmiş siyahının formatı müxtəlif şərtlərdən asılı olanda (məsələn, cari yazının və ya verilənlər bazasının sütunundakı qiymətdən asılı olur). Belə hallarda, qonşu sütunun qiymətindən asılı olaraq, mətn sahəsi gah görünən, gah da görünməyən ola bilər. Şərti formatlamayı tətbiq etmək üçün **FormatConditions** kolleksiyasının tərkibində olan **FormatCondition** obyektlərindən istifadə etmək məsləhət görülür.

Qruplaşmalarla işlədikdə **GroupLevel** obyektini hesabat və formalarda istifadə edilir. Əgər qruplaşma artıq hesabatda təyin edilibsə (məsələn, 0 qiyməti ilə qurulubsa), onda **ControlSource** xassəsi ilə hadisəvi **Open** prosedurasındakı qruplaşma səviyyəsini dəyişmək olar. Məsələn, aşağıdakı VBA kodu ilə **ControlSource** xassəsinin qiyməti **SortForm** adlı açıq formadakı **txtPromptYou** mətn sahəsində saxlanılan qiymətlə dəyişdirilir:

```
Private Sub Report_Open(Cancel As Integer)
  Me.GroupLevel(0).ControlSource=Forms!SortForm!txtPromptYou
End Sub
```

Module obyektini Access verilənlər bazasında proqram modullarını təmsil edir – standart modulları və ya sinif modullarını. Bu obyektləri **Modules** kolleksiyasının tərkibinə əlavə ediblər, kolleksiyanın özü isə **Application** obyektinin eyni adlı **Modules** xassəsindən əlçatan ola bilər. Layihələrə proqram səviyyəsində proqram kodunun əlavə edilməsi tələb olunanda, onda **Module** obyektindən istifadə edirlər. Bu məqsədlə həmin obyektə xüsusi olan metodlar nəzərdə tutulubdur: **AddFromFile()**, **AddFromString()**, **CreateEventProc()** v.s.

References xüsusi kolleksiyası obyekt kitabxanalarına istinadların avtomatlaşdırılmış rejimdə əlavə edilməsi (və ya dəyişdirilməsi) üçün nəzərdə tutulub. Access VBA-da **Refernce** (istinad) obyektləri toplusu həmin kolleksiyanın tərkibində yerləşir. VBA Access sənədləşməsində nədənsə **References** kolleksiyasında olan xassələr və metodlar haqqında heç bir məlumat yoxdur. Lazım olduqda, həmin məlumatların kod redaktorundan tapılması mümkündür. Layihədə obyekt kitabxanasına olan yeni istinadın əlavə edilməsini isə **References.AddFromFile()** metodu ilə həyata keçirmək lazımdır.

14. MS Outlook-da PROQRAMLAŞDIRMA

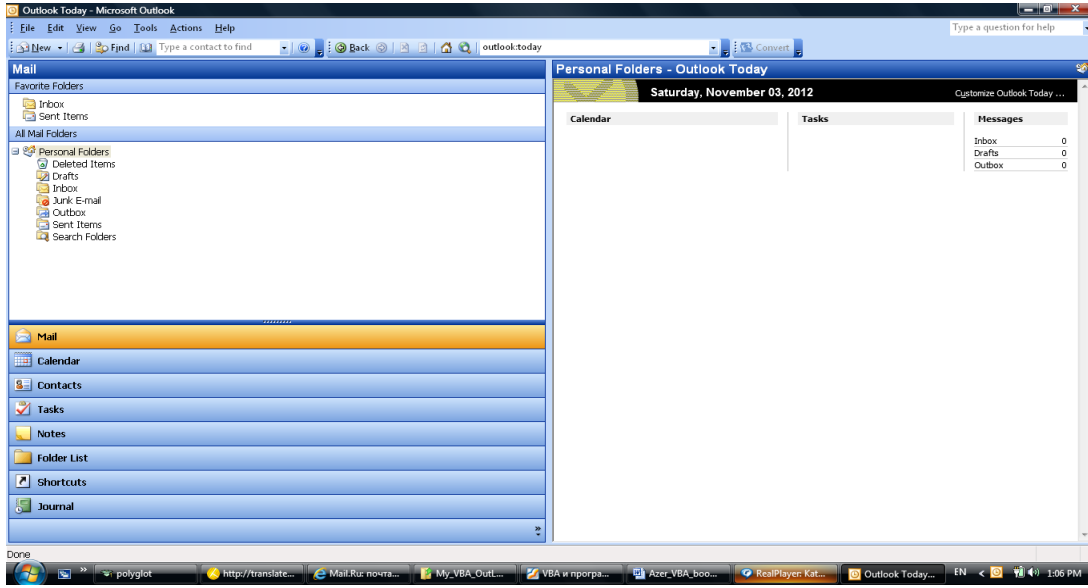
14.1 Outlook- da VBA proqramlaşdırmasının vacibliyi haqqında

Outlook-da VBA proqramlarının tətbiqində yaranan ehtiyaclar və tələblər, tipik hallar, tətbiqlər haqqında nümunələr

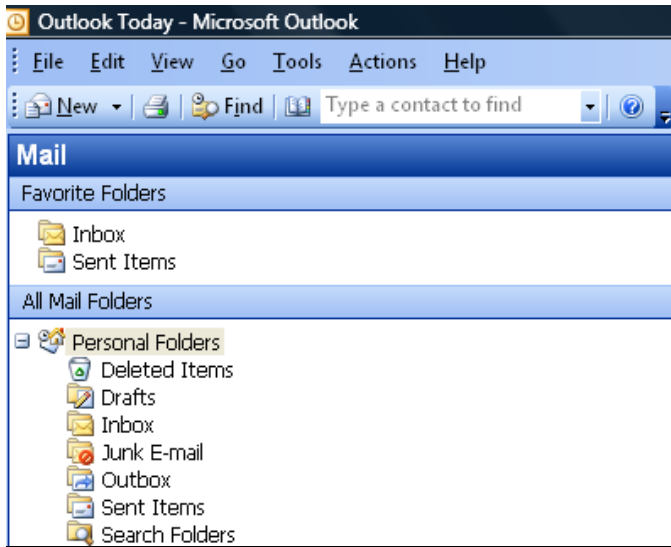
MS Outlook (MS Office paketində proqramın qısaldılmış versiyası Outlook Express adlanır) – qısaca desək, elektron poçtla işləmək üçün ən geniş yayılmış proqramdır. Bu Office proqramının vacibliyi yalnız onun elektron poçtla xəbərlərin göndərilməsi (alınması) ilə bağlı deyil. Praktikada daha çox onun başqa cəhətinin böyük və əvəzedilməz dəyəri vardır: Microsoft özü bu cəhətə **Personal Information Manager (PIM)** - azərbaycanca tərcümə etsək: *şəxsi elektron məlumat müdiri* (direktoru və ya meneceri).

Outlook-la praktiki məsələlərdən ən birincisi budur – kalendarla avtomatlaşdırılmış mühitdə işləmək (yəni, başqa sözlə, istifadəçinin iş vaxtının və işlə əlaqədar olan müxtəlif əməllərin, yarana biləcək hadisələrin, təyin olmuş zaman müddətində təşkili və idarə edilməsi). Əlbəttə, biz Outlook-da edilən adi əməllər haqqında, onun qrafik interfeysi ilə işləmə qaydaları haqqında burada danışmayacağıq. Bu haqda oxucu lazımı məlumatı digər ədəbiyyatda (Office proqramlarına həsr olmuş bilik resurslarında) tapa bilər. Bizi maraqlandıran sahə - Outlook-da VBA proqramlaşdırmasıdır. Nəzərə alınsa ki, bir çox adi istifadəçilər bu proqram ilə tanış deyil, onda aşağıda şəkl. 14.1 (a, b və c)-də MS Outlook-un qrafik interfeysinin görüntülərinin verilməsi vacibdir. Outlook-dakı **Calendar** rejimi Outlook-un digər elementləri (işləmə rejimləri) ilə sıx olan əlaqələrlə bağlıdır (bax şəkl. 14.1 (c), məsələn, **Mail, Contacts, Tasks, Notes, Folder List, Shortcuts**). Digər tərəfdən **Calendar** elementi həmçinin xaricdə olan proqramlarla da əlaqələnmiş qaydada işləyir (məsələn, **Microsoft Project** ilə). Müəssisə və şirkətlərdə Outlook-dan tətbiq edildikdə, məsələlərin təyin edilməsi çox hallarda istifadəçi tərəfindən tərtib edilir, – belə hallarda həmin məsələlər (iş əməliyyatları) **Calendar** rejiminin elementləri kimi, avtomatik olaraq, idarəetmə pəncərəsində əmələ gəlir. Tutaq ki, istifadəçinin poçt qutusu **Exchange Server**-dədir və həmin istifadəçi Outlook-a öz məsələlərini məhz özü daxil edir. Belə olduqda istifadəçinin kalendarına olan əlçatanlığı onun menecerlərinə təhvil verilməsi - ən əlverişli yol sayılır. Bu cür işləmə rejimində işin avtomatlaşdırılması baxımından bir vacib üstünlük vardır - menecerlər hal-hazırda həmin istifadəçidə hansı yerinə yetiriləsi məsələnin (işin) olduğunu dərhal görə biləcəklər. Bununla da

intensiv iş aparılan şöbələrdə və departamentlərdə bu cür xoşa gəlməz halların qarşısı alınmış olar: əməkdaş artıq vacib işlə məşğuldur, anacq ona yeni (növbədən kənar) çətin məsələ tapşırılır.



a)



b)



c)

Şəkil 14.1 (a, b və c) MS Outlook-un qrafik interfeysinin elementləri:

a) – MS Outlook-un ümumi görüntüsü; b) - MS Outlook-un Elektron poçtla bağlı **All Mail Folders** – *bütün poçt qovluqları* rejimində açılan idarəetmə diaqnoz pəncərəsi; c) – Outlook-un müxtəlif işləmə rejimlərinə keçmək üçün idarəetmə düymələri.

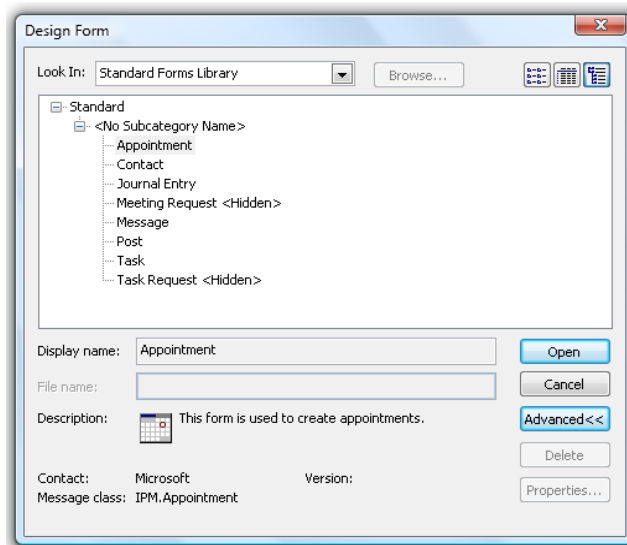
Calendar-in başqa vacib olan imkanı da vardır: əgər **Exchange Server**-də Outlook-un tərkibində **Calendar** tipli elementləri olan ümumi qovluq yaradılırsa, onda istifadəçilərin və ya menecerlərin sərəncamında bir üstünlük də mövcud olacaq: resurslara olan ümumi əlçatanlığın planlaşdırılması əlavə olan aktiv işlərdən ötrü hazır vəziyyətdə olacaq. Məsələn, tutaq ki, şirkətdə belə bir qayda qəbul edilibdir: müştərilərlə bütün şəxsi danışıqlar yalnız xüsusi olan danışıqlar ofis otağında aparılmalıdır, həmin otaqların sayı isə məhduddur. Buna görə bu cür hadisənin əmələ gəlməsi

istisna deyil: müştəri artıq gəlmişdir və hələ onun boş olan danışıq otağı yoxdur. Məhz belə hadisələrin qarşısı alınsın deyə, Outlook-un **Calendar** tipli ümumi qovluğu istifadə edilir. Müştəri ilə təmasda olduğu zaman İstifadəçi həmin ümumi qovluğu Outlook-dan açır və hansı vaxt hansı otağın boş olmasını görə bilir. Müştəri ilə görüşmə vaxtı haqqında razılığa gələndən sonra həmin ümumi qovluqda **Contacts** tipli elementi yaradırlar – bununla başqa əməkdaşlar artıq xəbərdar olacaqlar ki, hansısa tarixin müəyyən saatında hansısa danışıqlar otağı boş olmayacaq. Yuxarıda təsvir edilən işgüzar proseslərin avtomatlaşdırılmasında VBA proqramlaşdırması funksionallığının əlavə edilməsi qaçılmazdır: Outlook-un mühitində kalendarın başqa mənbələrlə sinxronlaşdırılması həmçinin ümumi kalendarların istifadə edilməsi tələb olunacaq. İkinci vacib məsələ - *kontaktlarla işləmədir* (real müştərilərlə qrup və ya personal formada əlaqələrin qurulması tələb olunur). Outlook kontaktı – əslində “ünvan kitabçasının” ən geniş yayılmış və standart olan formatıdır. Müəssisələrdə departamentlər arasında olan əlaqələrin ümumi siyahısı Outlook-un **Contacts** elementləri kimi **Exchange Server**-in ümumi qovluğunda saxlanılır. Ümumiyyətlə, təşkilatın ünvan/telefon kitabçası adətən **Exchange Server**-in ünvan kitabçasının formatında aparılır və həmçinin Outlook-dan əlçatan olur. Beləliklə, proqramlaşdırma səviyyəsində kontaktların yaradılması (məsələn, verilənlər bazasında olan informasiyanın əsasında) və ya onların sinxronlaşdırılması olduqca aktual məsələdir. Üçüncü məsələ budur – *tapşırıqlar və məsələlərlə əlaqəli olan işlər*. Miqyası böyük olan layihələrdə daha əlverişli yol budur - xüsusiləşmiş proqram təminatından istifadə etmək. Sadə olan layihələrdə isə tapşırıqlar və məsələlərlə əlaqəli işlərin avtomatlaşdırılması üçün Outlook proqramının tətbiq edilməsi (Outlook-da **Tasks** rejimi) kifayətdir. Bu cür layihələrdə məsuliyyəti bir, iki və ya maksimum beş menecer daşıyır. Məsələlərin strukturu təyin edildikdən sonra məsələlərin operativ həll edilməsi başqa məsul şəxslərə tapşırıla bilər. Bu cür proseduralarda həmin şəxslər elektron poçtla vaxtaşırı xəbərləşirlər - bununla bütün iş prosesi avtomatlaşdırılmış nəzarət altında keçir. Başqa bir fürsət – *qeydlər* ilə bağlı işlər. Aydınlıq üçün belə bir müqayisə gətirək: yaddaş üçün kiçik yapışqanlı vərəqlər (onları ofis əməkdaşları adətən öz ətrafında olan bütün yerlərə yapışdırırlar) və onların elektron analoqu olan Outlook vasitəsi (**Shortcuts** rejimi). Yuxarıda təsvir edilən, qurulmuş imkanlar ilə Outlook-da aparılan işlərin növü məhdudlaşmır. **Outlook - Exchange Server** bağlantısında bütöv bir proqramlaşdırma sahəsi əsaslanıb – *kollektiv istifadə proqramlarının hazırlanması*. Kollektiv istifadə proqramları ilə həll edilən məsələlərin əsas hissəsi – daxili korporativ informasiyanın yığılması və avtomatlaşdırılmış emal edilməsi kateqoriyasına aiddir. Məsələn, real həyatda bu cür vəziyyət yarana bilər: hər bir sırayı bank ayın sonunda mərkəzi dövlət bankına iqtisadi normativlər haqqında məlumat verməlidir. Əksər hallarda həmin səhnə reallıqda bu cür inkişaf edir: iqtisadi-planlaşdırma departamentinin informasiyanın yığılmasına məsul olan əməkdaşı mühasibat şöbəsinə gedir və oradan ayın sonunda hesablarda olan qalıqlar haqqında olan məlumatları alır. Gəlin baxaq görək Outlook vasitələri ilə həmin əməlləri necə avtomatlaşdırmaq olardı:

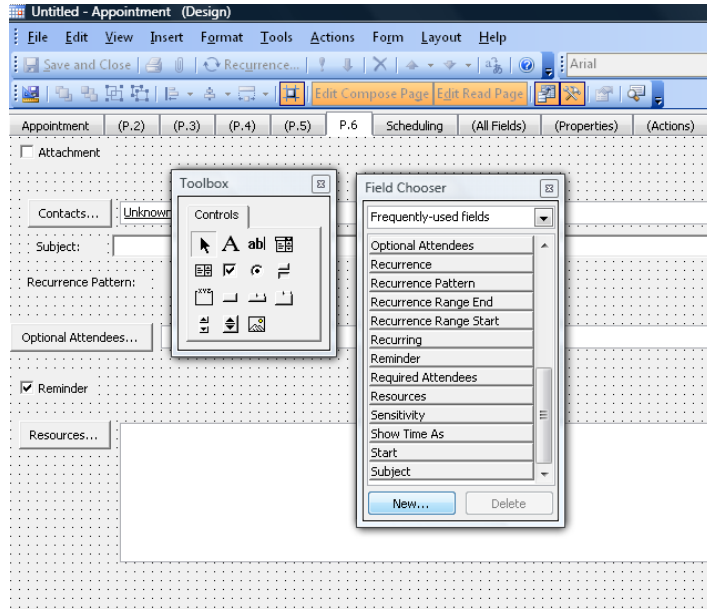
1. Hər ayın başında iqtisadi-planlaşdırma departamentinin informasiyanın yığılmasına məsul olan əməkdaşın Outlook-un **Inbox** qovluğunda avtomatik rejimdə normativlərin məlumatla doldurulması üçün xüsusi forma əmələ gəlir.

2. Əməkdaş, yuxarıdakı mərhələdə adı çəkilən, həmin formadakı **Mühasibata göndər** idarəetmə düyməsini basır. Forma dərhal mühasibatdakı məsul işçiyə göndərilir. Lakin burada başqa variantlarda ola bilər, məsələn, avtomatik olaraq, hesablarda olan qalıqlar haqqında məlumatları, tutaq ki, "Əməliyyat_günü" adlı Outlook Visual Basic for Application mühitində xüsusi tərtib edilmiş, proqramın köməyi ilə alır (əgər tərtibatçı həmin proqramı reallaşdırmağı bacarırsa).
3. Mühasibat əməkdaşı formanın lazım olan sahələrini doldurur və formadakı **Sonra** idarəetmə düyməsini basaraq, yadda saxlanılan verilənlərlə birgə, məlumatı kredit şöbəsinə ötürür.
4. Kredit şöbəsinin əməkdaşı öz növbəsində ona aid olan formadakı sahələri doldurur (avtomatik olaraq, əvvəlki mərhələdə mühasibatda doldurulan sahələr indi yalnız oxunma üçün əlçatandır) və bu da formadakı **Sonra** idarəetmə düyməsini basır.

İstənilən müəssisədə daxili korporativ informasiyanın yığılması və emalı ilə yarana biləcək məsələlərin sayı çoxdur. Məsələn, ezamiyyət haqqında hesabatlar, filiallardan gələn informasiya, müxtəlif səviyyədə məsuliyyətli əməkdaşlar tərəfindən doldurulmuş xüsusi formalar (ofisdə və ya ezamiyyətdə), inventarlaşdırmalar haqqında informasiya v.s. Əlbəttə, bu siyahını sonsuz qədər davam etdirmək olar. Praktika göstərir ki, ən yaxşı üsul - **Outlook/Exchange Server** vasitələrinin tətbiqidir. Bu məqsədlə xüsusi Outlook formaları istifadə edilir (idarəetmə elementləri və proqramlaşdırma imkanları ilə təchiz edilmiş, xüsusi şablonlar). Həmin formaların adı VBA və ya Access formaları ilə heç bir əlaqəsi yoxdur. Bu cür formanın yaradılmasını və ona lazım olan proqram kodunun əlavə edilməsini Outlook-un öz vasitələri ilə həyata keçirmək mümkündür. Bunun üçün Outlook-un qrafik interfeysindəki **Tools**→**Forms**→**Design a Form** menyusundan istifadə etmək lazımdır: nəticədə formaların dizayn edilməsi üçün pəncərə açılacaq (bax. şəkl. 14.2). Bu addımdan sonra isə standart xəbər vermə formasının şablonu yaradıla bilər: onun üzərinə yeni idarəetmə elementlərinin yerləşdirilməsi və onları proqram kodu (**Form**→**View Cod** menyusundan) ilə bağlanması kimi əməlləri həyata keçirmək mümkündür (bax şəkl. 14.3).



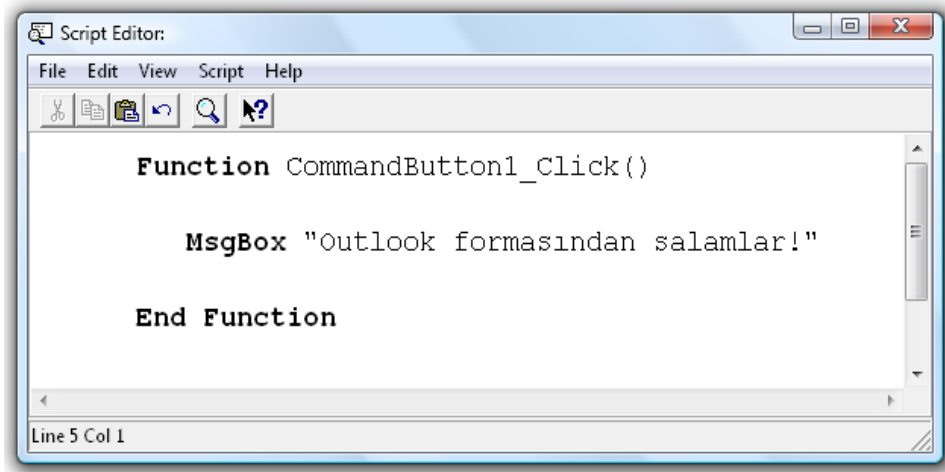
Şəkil 14.2 Outlook-un formaların dizaynının başlanması üçün **Design Form** dialoq pəncərəsi.



Şəkil 14.3 Outlook-un formalar dizaynı pəncərəsi.

Tutaq ki, həmin formada **CommandButton1** adlı düyməyə proqram kodu əlavə edilməlidir. Bunun üçün aşağıdakı əməlləri icra etmək lazımdır:

1. Forma dizayneri pəncərəsindəki **Form**→**View Cod** menyusunu seçərək **Script Editor** pəncərəsini açırıq.
2. Həmin pəncərədə aşağıdakı proqram kodunu yazırıq (şək. 14.4):



Şəkil 14.4 **Script Editor** pəncərəsində kod proqramının yazılması.

Əlbəttə, avtomatik rejimdə hadisəvi proseduranı generasiya etmək olmaz: bizim sərəncamımızda İntel lisensə koməkləri, sintaksis işıqlanması, testlənmə rejimi və digər VBA imkanları burada yoxdur.

3. Həmin düymə formalı idarəetmə elementini işə salmaq üçün **Form** menyusunda **Run This Form** komandası seçilməlidir.

Oxucu yəqin anladı ki, bu proqramlaşdırma mühiti VBA-da öyrəndiyimiz proqramlaşdırma mühitindən bir qədər fərqlənir. Əslində başqa proqramlaşdırma dilidir - **VBScript** (Visual Basic

Script). VBA-dan əlavə olaraq, Outlook formaları üçün özəl proqramlaşdırma mühiti, özəl obyekt və hadisəvi model nəzərdə tutulub. Həmin formalarla işləmə qaydaları **Exchange Server**-in korporativ imkanlarından ayrılmazdır: formalar kitabxanası, server skriptləri, ümumi qovluqlar, marşrutlaşma v.s. Bütün bunların hamısı əslində böyük bir mövzudur və diqqətlə baxılması üçün ayrıca həcmi böyük olan kitab tələb edir. Buna görə Outlook formaları və kollektiv istifadə edilən proqramlar bu kitabda baxılmayacaq. Əgər oxucu bu sahədə öz biliklərini genişləndirmək istəsə, onda Microsoft-un rəsmi sənədləşməsində olan faylından istifadə edə bilər. Bu fayl, standart halda, Office paketi yükləndikdə, C:\ProgramFiles\Microsoft Office\OFFICE11\1049\OLFM10.CHM yolunda yerləşir və ya www.slipstick.com saytında Outlook & Exchange Solutions Center səhifəsində lazım olan məlumatlar oxucu üçün əlçatan ola bilər. Bu kitabda isə, kitabın əsas məqsədlərinə uyğun olaraq, Outlook-la işləmədə ənənəvi VBA vasitələrindən istifadə edilməsinə diqqətimizi yönəldəcəyik. Burada biz VBA-nın standart modulları, formaları ilə proqramlaşdırmanı bizə artıq tanış olan VBA kod redaktorunda aparacağıq.

Qeyd etməliyik ki, VBA vasitələri ilə avtomatlaşdırma məsələləri Outlook-da kifayət qədər çoxdur:

- elektron poçtla məlumatların təyin olmuş vaxtla uyğun göndərilməsi;
- elektron poçtla məlumatın alınması və onun avtomatik emalı (məsələn, əgər məlumat standart formada gəlibsə, onda ondakı verilənləri götürərək, verilənlər bazasına qoymaq). Bu cür məsələlər şirkətlərin filialları ilə işlədikdə, verilənlərin yığılmasında yarana bilər;
- kontaktlara işləmədə avtomatlaşdırmanı tətbiq etmə məsələləri – məsələn, verilənlər bazasından və ya Excel faylından Exchange serverin ümumi qovluğuna import etmək lazım olduqda;
- kalendarla, avtomatik rejimlərdə, yazıların yaradılması və ya dəyişdirilməsinə tələbat olduqda.
- İnternet şəbəkəsində aparılan bir sıra başqa tədbirlərin təşkil edilməsində: məsələn, virtual mitinqlər, iclaslar, seminarlar v.s.

Bu fəslin hissələrində oxucu VBA proqramlaşdırma səviyyəsində Outlook-un həmin imkanları ilə işləmək qaydalarını öyrənəcək.

14.2 Outlook-dakı proqramlaşdırmanın bəzi xüsusiyyətləri haqqında

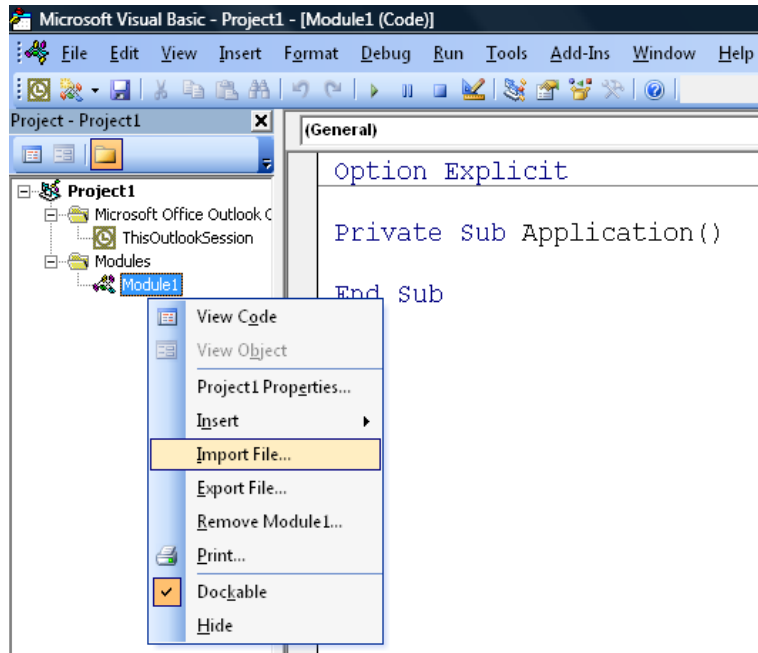
PST faylları və VBA proqram modulları, adlar fəzaları, item elementləri, Outlook Object Model Guard

Outlook-dakı proqramlaşdırmağa başlamaqdan əvvəl burada VBA proqramlaşdırmasında mövcud olan bəzi maraqlı xüsusiyyətlərin oxucununun bilməsi çox vacib məsələdir.

Birinci xüsusiyyət budur ki, yaradılması VBA proqram kodu, Outlook-un proqram modulları kimi, məhz harada yerləşməlidir. Diqqətli oxucununun yadındadır ki, məsələn, Word-də onlar sənədlərlə (və

ya şablonlarla, məsələn, normal.dot kimi) birgə, Excel-də iş kitabları fayllarında, Access-də isə **MDB** verilənlər bazası fayllarında saxlanılır. Outlook-da modulların standart modulları PST şəxsi qovluqlar fayllarında yerləşir - standart halda istifadəçinin profilində yaradılır. Nəticədə, bir tərəfdən, Outlook-da VBA proqram kodu ilə işləmə həmin istifadəçi üçün sadələşir: istifadəçi kompyuterində Outlook-dan həmin proqram həmişə əlçatan olur. Başqa bir tərəfdən isə həmin yaradılmış proqram kodunun digər istifadəçinin sərəncamına keçməsi mümkün olmaz hala çevrilir. Bu halda iki çıxış yolundan istifadə etmək mümkündür:

- birinci yol – *proqram modullarının eksport və import vasitələrindən istifadə etmək* (bunlar isə **Project Explorer**-də mausla yaradılan (bax. şəkl. 14.5) kontekst menyusundan əlçatandır);
- ikinci yol – konteyner əlavəsini (məsələn, Word faylını və ya Excel kitabını yaratmaq): onun vasitəsi ilə proqram səviyyəsində Outlook-u işə salaraq açılmış faylda lazımı əməlləri yerinə yetirmək.



Şəkil 14.5 Proqram kodunun Outlook VBA mühitində import və eksport imkanları.

Outlook VBA mühitində proqramlaşdırmanın ikinci xüsusiyyəti budur ki, Outlook-da adlar fəzası konsepsiyası nəzərdə tutulub. Outlook-da adlar fəzası, formal olaraq, istənilən verilənlər mənbəyinin (məsələn, diskdə **Exchange poçt** qutusunda və ya **PST** faylıdakı qovluqlar kimi) abstrakt olan iyerarxik ağac kökü obyektini kimi təyin edilir. Outlook-un adlar fəzasını, daha yaxşı və sadə yolla anlamaq üçün onu verilənlərə qoşula bilən hansısa drayver kimi təsəvvür etmək lazımdır. Hal-hazırda Outlook yalnız bir sayda adlar fəzasını dəstəkləyir – **MAPI** (yəni Microsoft-un tərifli ilə **Messaging Application Programming Interface**). Son vaxtlar tərtibatçılar tələb edirlər ki, həmin adlar fəzası müxtəlif əməllərin icrasında aşkar bildirilsin. Məsələn, Outlook-un işə salınması üçün və onun içərisində olan **Inbox** (Girənlər) qovluğunun başqa proqramdan açılması aşağıdakı

VBA proqram kodu ilə həyata keçirilə bilər (hökmən yoxlayın, Microsoft Outlook 11.0 Object Library kitabxanasına istinadın qurulmasını yaddan çıxarmayın):

```
Dim oOutlook As New Outlook.Application
    Set oNameSpace=oOutlook.GetNamespace("MAPI")
    Set oInbox=oNameSpace.GetDefaultFolder(olFolderInbox)
oInbox.Display
```

Outlook VBA mühitində proqramlaşdırmanın üçüncü xassəsi isə budur: bəzi terminoloji qarışıqlıqların mövcud olması. Bir qayda olaraq, MS Office proqramlarının obyekt modeli rəsmi sənədləşməsində **item** (element) termini kolleksiyalar elementlərinə aiddir. Outlook-da bu terminin ikinci mənası əmələ gəlir: **item** – Outlook qovluğunda saxlana bilən hər şey (məsələn, poçt xəbəri – **MailItem** obyekt, kontaktlar – **ContactItem** obyekt, görüşlər – **Appointment** obyekt v.s. ola bilər. Qarışdırmaq lazım deyil!

Outlook VBA mühitində proqramlaşdırmanın dördüncü xüsusiyyəti - Outlook proqramı uzun illər müddətində, birinci olaraq, müxtəlif çeşidli virusların qurbanı olmuşdur (troyan proqramları, poçtla gələn, ziyan verici proqram təminatları v.s.). Bəzən də bu cür hücumlar müvəffəqiyyətli olmuşdur: nəticədə Outlook istifadəçisi öz kompyuterindən virusla dolu məktubları (həmçinin VBA proqram kodunun köməyi ilə də) göndərməyə başlayır. Bu cür xoşa gəlməz halların qarşısını almaq üçün Microsoft, bilərəkdən, Outlook-un obyekt modelində məhdudiyyətlər qoydu. Bəzən həmin məhdudiyyətlər (onların öz adı vardır - **Outlook Object Model Guard**) VBA proqramlarının normal işləməsinə əngəllər törədir. Bu fəslin sonunda həmin məhdudiyyətlər haqqında bir qədər ətraflı danışacağıq. Həmin Outlook məhdudiyyətlərinin səbəbindən, bəzən, Outlook-un obyekt modeli əvəzinə **CDO (Collaboration Data Objects)** kitabxanasının istifadə edilməsi daha əlverişli olur. Nəzərə alınmalıdır ki, **CDO** kitabxanası hətta ən müasir Windows əməliyyat sistemi ilə birgə hər bir kompyutərə yüklənmiş olur.

Outlook-un digər maraqlı xassəsi budur ki, başqa Office proqramlarından fərqli olaraq, burada **Application** obyektindən başqa daha heç bir obyekt bir başa (**New** açar sözü ilə və ya **CreateObject** əmri ilə) yaratmaq olmur. Başqa obyektləri yaratmaq üçün digər obyektlərin uyğun olan metodlarından istifadə etmək lazımdır.

Həmçinin bunu da qeyd edək ki, Outlook-da, Access-də olan kimi, proqramlaşmada çox faydalı olan makrorekorder yoxdur. İstifadəçi bütün ona lazım olan məlumatları Outlook-un rəsmi sənədləşməsindən axtarıb tapa bilər.

14.3 Application obyekt, onun xassələri və metodları

Outlook.Application obyekt, onun xassələri və metodları, proqram səviyyəsində xəbərdarlıqların kəsilməsi və kontaktların yaradılması

Hər bir Office proqramında olan kimi, Outlook obyekt modelinin başında **Application** obyekt yerləşir. Bu obyekt xaricdə yerləşən proqramlardan Outlook-un işə salınması üçün istifadə etmək olar (yuxarıdakı bölmədə verilən nümunəyə bax). **Application** obyektinin Outlook-dakı

fərqləndirici xüsusiyyəti budur ki, digər Office proqramları ilə müqayisədə, burada həmin obyekt daha az sayda xassə və metodlarla təmsil edilib. İzahı sadədir - həmin xassələr və metodların çoxu **Namespace** obyektinin tərkibinə keçib.

Gəlin VBA proqramlaşdırmasında daha çox vacibliyi cəhətlərinə görə ən çox istifadə edilən aşağıdakı **Application** obyektinin xassələrini öyrənək:

- **Explorers** - bu xassə eyni adlı **Explorers** kolleksiyasını onun tərkibindəki **Explorer** obyektləri ilə birgə qaytarır (hər bir obyekt öz növbəsində, istifadəçi tərəfindən baxılmağa təqdim edilən, Outlook qovluğunu təmsil edir). Həmin kolleksiyanın və tərkibində olan obyektlərin əsas təyinatı budur: Outlook-da bu və ya digər qovluğun istifadəçi tərəfindən açılıb açılmasının yoxlanmasını icra etmək (nəticədən asılı olaraq, məsələn, həmin pəncərənin aktivləşməsi - **Explorer.Activate()** metodu ilə və ya bağlanması **Explorer.Close()** metodunun tətbiqi ilə). **Application.ActiveExplorer()** metodu ilə hal-hazırda açıq olan pəncərəyə istinad almaq mümkün olur. **Application.GetExplorer()** metodu ilə Outlook-un lazım olan qovluğunun aktivləşdirilməsini icra etmədən, onun **Explorer** obyektini üçün istinad almaq olur.
- **Inspectors** - olduqca **Explorers** xassəsinə bənzəyir. Xassə tərkibində **Inspector** obyektləri olan **Inspectors** kolleksiyasını qaytarır. Əsas fərq budur – **Inspector** obyektləri Outlook-un açıq olan qovluqlarını yox, baxma/redaktə əməllərinin yerinə yetirilməsi üçün açılan elementləri təmsil edir (məsələn, poçt göndərişlərini). **Inspector** obyektini, **Explorer** obyektini kimi, həmin məqsədlərlə istifadə edilir - yoxlamalar üçün. Obyektin xassələri və metodları eyniliklə **Explorer** obyektində olanlarla üst-üstə düşür. Analoji olaraq, **ActiveInspector()** və **GetInspector()** metodları həmin məqsədlərlə istifadə edilir.
- **Reminders** – xassəsi eyni adlı **Reminders** kolleksiyasını və onun daxilində olan **Reminder** obyektlərini qaytarmağa imkan yaradır. Həmin obyektlər isə cari xəbərdarlıqları təmsil edir və proqram səviyyəsində bütün xəbərdarlıqların kəsilməsini təmin edir:

```
Dim oOutlook As New Outlook.Application
Dim oReminder As Outlook.Reminder
    For Each oReminder In oOutlook.Reminders
        oReminder.Dismiss
    Next
```

- **Session** – bu xassə **Namespace** obyektini qaytarır: öz növbəsində həmin obyekt cari seansın adlar fəzasını, yeni əslində **MAPI**-ni (**Messaging Application Programming Interface**) təmsil edir. Bu xassəsinə **GetNamespace()** metodunun əvəzinə tətbiq etmək olar. **Namespace** obyektinin özü üçün və həmçinin bir sıra Outlook obyektləri üçün də analoji **Session** xassəsi nəzərdə tutulub. **Namespace** obyektini haqqında daha ətraflı məlumat növbəti bölmədə verəcəyik.

İndi isə **Outlook.Application** obyektinin metodlarına baxaq (proqramlaşdırma baxımından onlar böyük maraq kəsb edir):

- **Active...()** – metodu sadəcə hal-hazırda aktiv olan **Explorer**, **Inspector** və ya **Window** obyektinə istinad qaytarır.
- **AdvancedSearch()** – çox vacib metoddur: Outlook qovluqları üzrə axtarışı icra edir (praktikada bu əməllərə böyük ehtiyac yaranır). Metod və onunla bağlı **Search** və **Results** obyektləri haqqında aşağıda ətraflı danışacağıq.
- **CopyFile()** - diskdən Outlook qovluğuna hansısa faylı kopyalayır. Məsələn, bu metodu layihə üzrə olan sənədləşmə kataloqdakı bütün faylların **Exchange Serve**-in ümumi qovluğuna və ya **SharePoint Portal Server** sənədlər kitabxanasına köçürülməsində istifadə etmək olar.
- **CreateItem()** – daha tez-tez istifadə edilən metodlardandır: Outlook-da yeni elementlərin yaradılmasına imkan verir. Məsələn, kontakt tipli element yaradılaraq onun xassələri doldurulur. Sonradan, baxmaq üçün açılmasını aşağıdakı VBA proqram kodunun istifadəsi ilə həyata keçirmək olar:

```
Dim oOutlook As New Outlook.Application
Dim oContact As Outlook.ContactItem
Set oContact=oOutlook.CreateItem(olContactItem)
oContact.FirstName="Xüsusi kurslar akademiyası"
oContact.EmailAddress="inform@askme.az"
oContact.Save
oContact.Display
```

İndi isə təsəvvür edin ki, Excel cədvəlinin sətirləri və ya verilənlər bazası yazıları əsasında kontaktlar obyektləri dövrədə yaradılmalıdır. Bu halda kontaktlar sorğu kitabı Outlook-a çox tez və effektiv olaraq yüklənəcək. Yaddan çıxarmaq olmaz – hər kontaktı yaradıb yaddaşda saxladıqdan sonra, onun obyektini operativ yaddaşdan silmək lazımdır – digər halda kompyuterin yaddaşı tükənəcək və səhv haqqında dərhal xəbər veriləcək!

- **CreateItemFromTemplate()** – eyniliklə həmin qaydada Outlook elementini yaradır, yalnız Outlook-un şablonu əsasında fayl sistemində (**.oft.** fayl genişlənməsi formatında).
- **GetNameSpace()** – metodu ilə **MAPI** adlar fəzasının obyektini almaq olur: VBA proqramlarının əksəriyyətində istifadə edilir. Bu obyektlə işləmə qaydası haqqında daha ətraflı növbəti bölmədə danışacağıq.
- **IsSearchSynchronous()** – bu metoddan axtarış rejimində yoxlamaları aparmaq üçün istifadə edirlər (Outlook-da axtarışa həsr edilən bölməyə bax).
- **Quit()** – sadəcə Outlook-dan çıxmağı təmin edir.

14.4 Namespace obyektı

Outlook.Namespace obyektı, onun xassələri və metodları, VBA-dan elektron poçtun alınması və emal edilməsi, proqram səviyyəsində qovluğa qoşulma əməlləri

Yuxarıda dediyimiz kimi, Outlook-da adlar fəzası anlamı istifadə edilir – Microsoft-un niyyəti ilə, müxtəlif istifadəçi verilənləri saxlanclarını (konteynerlərini) əlçatan edən, drayverlərə bənzər bir vasitədir. Bu cür drayverlərin hər birinin, əlbəttə, öz imkanları olmalıdır. Uzun müddətdir ki, Outlook-da yeganə olan **MAPI (Messaging Application Programming Interface)** adlar fəzası istifadə edilir və hələlik bu sahədə olan yeniliklərdən heç bir məlumat yoxdur.

Əgər tərtib edilən VBA proqramında **Namespace** obyektinin xassələri və metodları lazım olacaqsa, onda bu obyektə istinad edilməsinin iki üsulu vardır:

1. **Application** obyektinin **GetNameSpace ()** metodundan istifadə etmək:

```
Set oNameSpace=Application.GetNameSpace("MAPI")
```

2. **Application** obyektinin **Session** metodundan istifadə etmək:

```
Set oNameSpace=Application.Session
```

Yuxarıdakı iki sətir VBA proqram kodu eyni güclüdür - rəsmi sənədləşmədə həmişə daha çox birinci sətirdən istifadə edilir.

Bu obyekt bizə VBA proqramlaşdırmasında nə üçün lazımdır? Cavab birmənalıdır - elektron poçtla müxtəlif, geniş yayılmış, əməllərin icrasını avtomatlaşdırılması üçün, məsələn:

- elektron poçtun serveri ilə qoşulmanı yaratmaq lazım olduqda;
- elektron poçtun göndərilməsi/alınmasında;
- lazımı qovluğun seçilməsində;
- adres kitabları ilə işləmək istədikdə;
- bütövlükdə **Exchange** qovluqları üzrə axtarış apardıqda (əlavə olaraq, **Namespace** obyektı üstəlik bütün **Exchange** qovluqlarının virtual kök strukturunu təmsil edir).

Beləliklə, **Namespace** obyektı praktiki işlərdə tez-tez istifadə edilir. Məsələn, tutaq ki, Outlook-dan bu tapşırıq təcili yerinə yetirilməlidir: elektron poçtun bütün hesabat yazıları üçün poçt xəbərləri yüklənməlidir və onların hərəsi ilə lazım olan əməllər yerinə yetirilməlidir. Gətirdiyimiz misalda biz sadəcə hər bir xəbəri çıxaracağıq. Əslində praktikada biz gərək hər bir xəbəri standart formatda seçə idik və alınan məlumatları verilənlər bazasına qoya idik. Bu cür tapşırıqların yaranma səbəbləri bunlar ola bilər: şirkətin filiallarından, provayderdə yerləşən müəssisənin Web-saytından, ezamiyyətdə olan ticarət nümayəndələrindən v.s. gələn informasiyanın emal edilməsində.

Bəs bu cür xəbər hansı formada ola bilər? Gəlin təsəvvür edək ki, biz xaricdə yerləşən proqramdan işləyəsi olacağıq. Aydındır ki, hadisələrlə işlərin icrasını VBA kod redaktorunda

aparılması üçün ən rahat yol - Word və ya Excel formalarından istifadə etməkdir. Ən birinci edilən əməl budur - Microsoft **Outlook 11.0 Object Library** obyekt kitabxanasına olan istinadı layihəyə qoymaq. Sonra isə biz artıq tanış olduğumuz adi işlə məşğul olmalıyıq - ən adi VBA formasını yaradaraq, standart hala görə onun üzərinə **CommandButton1** idarəetmə düyməsini yerləşdiririk. VBA proqram kodunun **General**→**Declarations** bölməsində hansısa abstrakt olan **Outlook.Items** obyektini üçün hadisələr qeyd olmalıdır. Bu əməli aşağıdakı proqram sətiri ilə icra etmək olar:

```
Public WithEvents oItems As Outlook.Items
```

Bundan sonra, oItems obyektini hadisələri ilə birgə kod redaktoru pəncərəsində əmələ gələcəkdir. İndi isə həmin forma üçün artıq VBA proqram kodunun yazılmasına başlamaq olar. Həmin proqram kodu, məsələn, aşağıdakı VBA proqram kodu (izah edici şərhlərlə birlikdə) kimi yazıla bilər:

```
Private Sub CommandButton1_Click()  
'Outlook-u işə salırıq  
Dim oOutlook As New Outlook.Application  
'Bunu bilmək vacibdir: oItems hadisələri - əslində  
'Outlook-da Inbox qovluğunun hadisəsidir.  
  Set oItems=oOutlook.GetNamespace("MAPI").GetDefaultFolder_  
(olFolderInbox).Items  
'İndi sadəcə bütün qeydləmə yazılartından  
'poçtun qəbul edilməsini işə salırıq.  
Dim oNamespace As Outlook.NameSpace  
  Set oNamespace=oOutlook.GetNamespace("MAPI")  
oNamespace.SyncObjects("Bütün qeydləmə yazıları").Start  
End Sub  
'... və yeni xəbərin əmələ gəlməsi hadisəsini tuturuq.  
Private Sub oItems_ItemAdd(ByVal Item As Object)  
  MsgBox Item.Subject  
End Sub
```

Microsoft sənədləşməsində deyilir ki, əgər qovluğa bir dəfəyə çox sayda elementlər əlavə olunsa, onda **ItemAdd** hadisəsi işləməyə bilər. Bəzi istifadəçilər bu hadisənin etibarlı işləməsini təsdiqləyir.

Əgər şirkətdə **Exchange Server** qurulubsa, onda daxil olan elektron poçtun emalı üçün daha düzgün yol – **Exchange**-in server skriptlərini və onun hadisəvi modellərini istifadə etməkdir. Real həyat əksər hallarda daha mürəkkəb olur. Məsələn, çox hallarda **Exchange Server**-in etibarlı işləməsinə administrator cavab verir. O isə, adi hallarda, tərtibatçılara hansısa skriptlərin sazlanmasına icazə vermir. Başqa variant – şirkətdə (müəssisədə), məsələn, **Unix - SendMail, PostFix, CommunigatePro** və ya digər poçt sistemi işləyir. Onların hadisələrini anlayıb, işə salınması uzun zaman və çox miqdarda resurs tələb edə bilər. Outlook həmişə istifadəçinin əlinin altındadır.

İndi isə, **Namespace** obyektinin proqramlaşdırmada daha vacib olan xassələri və metodları haqqında danışaq:

- **AddressLists** – bu xassə, **AddressList** obyektləri tərkibində olan, **AddressLists** kolleksiyasını qaytarır. **AddressList** obyektləri hal-hazırda əlçatan olan adres kitablarını təmsil edirlər. Məsələn, hal-hazırda istifadəçi üçün əlçatan olan adreslər kitablarının siyahısını proqram səviyyəsində bu cür almaq olar (fərz edilir ki, istifadəçi Outlook-da xaricdə yerləşən proqramdan işləyir):

```
Dim oOutlook As New Outlook.Application
Dim oNameSpace As Outlook.NameSpace
Dim oAddress As Outlook.AddressList
Set oNameSpace=oOutlook.GetNamespase("MAPI")
For Each oAddress In oNameSpace.AddressLists
Debug.Print oAddress.Name
Next
```

AddressList obyektində **AddressEntries** kolleksiyası vardır. Öz növbəsində həmin kolleksiyanın tərkibində, adres kitablarını təmsil edən, **AddressEntry** obyektləri yerləşib. Bu cür obyekt “budağı” ilə istifadəçi proqram səviyyəsində adres kitabına obyektləri əlavə edə bilər (lazım olduqda silə bilər, xassələrini v.s. dəyişə bilər). əgər Outlook heç bir əlavə verilənlər mənbəyinə qoşulmayıbsa, onda yuxarıdakı əməllərə heç bir ehtiyac yaranmamalıdır: istifadəçi üçün əlçatan olan adres kitablarından yalnız **Contacts** olacaq, onlarla isə daha rahat işləmə üsulu – **ContactItem** obyektlərinin tətbiqidir. **Outlook Exchange Server** qoşulubsa, onda **AddressLists** xassəsi çox lazımlı və maraqlı vasitəyə çevrilə bilər.

- **CurrentUser** – daha bir xeyirli xassədir: cari istifadəçi (onun adı ilə Outlook açılmışdı) haqqında məlumatı qaytarır. Qaytarılan obyekt **Recipient** formasında alınır. Həmin obyektlə, məsələn, ünvanlar siyahısında yazılmaq üçün nəzərdə tutulan, cari istifadəçinin elektron poçtunun ünvanını, onun adını və digər atributlarını almaq olar (əgər həmin parametrlər cari istifadəçi üçün təyin edilmişdirsə). Məsələn, cari istifadəçinin elektron poçtunun ünvanı haqqında olan məlumatı əlçatan etmək üçün, aşağıdakı VBA proqram kodunu istifadə etmək olar:

```
Set oNameSpace=Application.GetNamespase("MAPI")
Set oRecipient=oNameSpace.CurrentUser
Debug.Print oRecipient.Address
```

Təəssüflər olsun ki, xaricdə yerləşən proqramdan həmin proqram kodunu icra etdikdə bu cür xəbər əmələ gəlir: *“Do you want to do acceptable the Outlook E-mail addresses?”*, yəni *“Outlook-dakı elektron poçtun ünvanlarını əlçatan etmək istəyirsinizmi?”*. İstifadəçiyə bu xəbər pəncərəsinin heç bir faydası yoxdur – Microsoft pəncərəni əlavə təhlükəsizliyi təmin etmək üçün nəzərdə tutub. Lakin istifadəçinin bu halda əlində iki variantı ola bilər: 1) proqram səviyyəsində həmin pəncərədə **<Shift>+<Tab>** və **<Enter>** düymələrinin basılmasının imitasiyasını icra etmək (Microsoft qayğıkeşliklə, hətta **Yes** qaynar düyməsini həmin pəncərədə söndürür); 2) bu proqram kodunu işləyən Outlook-dan işə salmaq. Həmin tədbirlər edildikdə - xəbərdarlıq yaranmır. Düymələrin basılmasını imitasiya etmək üçün

Windows Script Host obyekt kitabxanasının **WshShell** obyektini tətbiq etmək olar. Bu halda əlavə çətinlik yaranabilir: VBA kodunun icrasında heç bir səhv yaranmasa da, sadəcə **Debug.Print** `oRecipient.Address` sətirinin icrası "sallaq" vəziyyətdə qalır.

- **ExchangeConnectionMode** – praktiki işlərdə çox vacib olan xassədir: hal-hazırda Outlook-un **Exchange Server**-lə işləməsi üçün sazlanmasını və ona qoşulmasını təyin edir.
- **Folders** - xassəsi **Namespace** obyektinin ən əsas xassəsidir: **MAPIFolder** obyektləri tərkibində olan, **Folders** kolleksiyasını qaytarır. **MAPIFolder** obyektləri Outlook-un yuxarı səviyyədə duran qovluqlarını təmsil edir (hər bir qovluğun öz **Folders** xassəsi vardır – buna görə istifadəçi, heç bir şübhə olmadan, Outlook-un bütün qovluqları üzrə keçərək, yoxlamalar apara bilər. Outlook-un qovluqlarının vacib olan özəl xassələri və metodlarından əlavə olaraq, qovluğun **Items** xassəsi istifadəçi qovluğun bütün elementlərini əlçatan edə bilər (xəbərləri, kontaktları, kalendar elementlərini v.s.). **Folders** kolleksiyası və **MAPIFolder** obyektini haqqında növbəti bölmədə daha ətraflı məlumat verəcəyik.
- **Offline** – Outlook-un hal-hazırda elektron poçt serverinə qoşulub/qoşulmamasını təyin edir.
- **SyncObjects** – içərisində **SyncObject** sinxronlaşma obyektləri olan kolleksiyayı qaytarır. Obyektlərin özləri göndəriş qruplarını təmsil edir. **SyncObject** obyektini ilə icra edilən ən vacib iş – elektron poçt serveri ilə proqram səviyyəsində qoşulmanı və ya ayrılmanın imitasiya edilməsidir.

Adətən Outlook obyekt modelini istifadə edən proqramlarda **Namespace** obyektində olan metodlardan daha geniş istifadə edilir. Onlardan proqramlaşdırmada ən vacib olanları aşağıda təsvir edilib:

- **AddStore()** və **AddStoreEx()** – proqram səviyyəsində, diskdə yerləşən, Outlook xəbərləri saxlanıcını təmsil edən faylı (**.PST**) açır. Buna diqqət verin: əgər diskdə həmin fayl yoxdursa, onda metod çağırıldıqda, sadəcə həmin faylı yaradır. **AddStoreEx()** xəbərlər faylı üçün kodlaşmanı göstərməklə fərqlənir. **PST** faylı **RemoveStore()** metodu ilə bağlamaq mümkün olur.
- **CreateRecipient()** – bu metod proqram səviyyəsində **Recipient** obyektini yaradır. Adətən başqasının poçt qutusunda olan **GetSharedDefaultFolder()** metodunun ötürülməsi üçün istifadə edilir.
- **Dial()** – istifadəçinin komutə olan birləşməni qoşmaq üçün **New call** dialoq pəncərəsini açır. Məcburi olmayan parametr kimi, avtomatik olaraq, həmin pəncərəyə yerləşdirilən, kontaktın adını qəbul edir.
- **GetDefaultFolder()** – olduqca vacib metoddur: Outlook-da qurulmuş 12 sayda olan qovluqların hansısa birisi üçün **MAPIFolder** obyektini qaytarır. Məsələn, **Contacts**

qovluğuna program səviyyəsində qoşulmağı və açılmasını, aşağıda verilən VBA program kodu ilə həyata keçirmək olar:

```
Dim oOutlook As New Outlook.Application
Set oNameSpace=oOutlook.GetNamespace("MAPI")
Set oInbox=oNameSpace.GetDefaultFolder(olFolderContacts)
oInbox.Display
```

- **Get...FromID()** – bu strukturda olan metodlardan, adətən yalnız o halda istifadə edilirlər ki, istifadəçi, CDO obyekt kitabxanasından (onun haqqında daha ətraflı məlumatı oxucu son bölmədə tapa bilər) istifadə edərək, yazılan öz kodunu Outlook obyekt modelinin koduna çevirmək istəyir.
- **GetSharedDefaultFolder()** – bu metod eyniliklə **GetDefaultFolder()** metodunun etdiyini icra edir, yalnız bir halda tətbiq edilir: istifadəçinin Outlook-da öz poçt qutusunda əlavə başqalarında poçt qutusu açıq olduqda və ya müəyyən poçt qutusunda **Inbox** qovluğuna istinad alınması lazım olduqda.
- **Logon()** - MAPI protokoluna uyğun olan qoşulmanı təyin edir (yeni əslində **Exchange Server**-lə birləşməni təyin edir). Məcburi olmayan parametrlər kimi, poçt profilinin adını, dialoq pəncərəsinin göstərilib/göstərilməməsini təyin edən qiyməti v.s. qəbul edir. **Exchange Server**-lə ayrılmanı **Logoff()** metodu ilə icra etmək olur.
- **PickFolder()** - **Folder choose** dialoq pəncərəsini açır: əgər istifadəçi həmin pəncərədə qovluğu seçərək **OK** düyməsinə basıbsa, onda metod seçilmiş qovluq üçün **MAPIFolder** obyektini qaytaracaq.

14.5 Folders kolleksiyası və MAPIFolder obyekt

Outlook.MAPIFolder obyekt, onun xassələri və metodları, konkret seçilən qovluğa istinadın alınması, VBA-dan bütün qovluqlar üzrə keçid əməllərinin təşkili

Adətən program səviyyəsində Outlook-da işlədikdə, onun elementləri ilə hansısa əməllərin icra edilməsi tələb olur – poçt xəbərləşməsinin təşkili, kontaktların qurulması, kalendər əsasında görüşlərin təşkili v.s. Bütün mümkün olan əməlləri icra edən elementlər Outlook-un qovluqlarında yerləşir (Microsoft tərəfindən əvvəldən qurulmuş olurlar və ya istifadəçi tərəfindən yaradılırlar). Outlook-un obyekt modelində qovluqlara, **Folders** kolleksiyasının tərkibində olan, **MAPIFolder** obyektləri uyğun gəlir.

Outlook-da qovluqlar iç-içə də yerləşdirilə bilər. İyerarxiya baxımından, ən yuxarı yerdə yuxarı səviyyəli qovluqlar yerləşir. Bunlar, ola bilsin, oxucu düşündüyü kimi, **Inbox**, **Contacts**, **Drafts**, v.s. deyil, əslində daha yüksək səviyyəli qovluqlardır: məsələn, **Personal Folders**, **Common Folders**, **Mailbox – Administrator**. Ən yüksək səviyyəli qovluqları təmsil edən **Folders** kolleksiyasını

Namespace obyektinin **Folders** xassəsi ilə əlçatan etmək mümkündür (yuxarıda bu haqda ətraflı izahlar vermişik). Həmin əməli aşağıdakı VBA proqram kodu ilə həyata keçirmək olar:

```
Dim oOutlook As New Outlook.Application
Dim oNameSpace As Outlook.NameSpace
Dim oFolder As Outlook.MAPIFolder
Set oNameSpace=oOutlook.GetNamespace("MAPI")
For Each oFolder In oNameSpace.Folders
    Debug.Print oFolder.Name
Next
```

Əgər yaranmış situasiyada konkret seçilən qurulmuş qovluq lazımdırsa (məsələn, **Inbox**), onda daha asan yol budur – həmin qovluğu **Namespace** obyektinin **GetDefaultFolder()** metodu ilə tapılmasını icra etmək. Aşağıdakı VBA proqram kodu həmin əməli reallaşdırır:

```
Dim oOutlook As New Outlook.Application
Dim oNameSpace As Outlook.NameSpace
Dim oFolder As Outlook.MAPIFolder
Set oNameSpace=oOutlook.GetNamespace("MAPI")
Set oFolder=oNameSpace.GetDefaultFolder(olFolderInbox)
Debug.Print oFolder.Name
```

Tez-tez praktikada belə hallar da yaranır: Outlook-un poçt qutusunda bəzən bir neçə sayda poçt qutularında) VBA vasitəsi ilə bütün qovluqlar üzrə keçid əməllərini təşkil etmək lazımdır. Məsələn, hər bir qovluqda (həmçinin iç-içə yerləşənlərdə də) bütün xəbərlərə baxmaq və həmin xəbərlərlə nə isə də etmək (məsələn, müəyyən bir göndəricidən gələn məktubları hansısa bir xüsusi qovluğa yığmaq). Aşağıda oxucuya təqdim edilən nümunədə həmin əməllər bu cür icra edilir: sadəcə Outlook-un bütün qovluqları üzrə keçid edərək, onların adları çıxarılır. Əlbəttə, diqqətli oxucu çox zəhmət çəkmədən, təqdim olan nümunəni bu cür əməllərin icrası üçün də modifikasiya edə bilər: hər bir qovluqda olan elementlər yoxlanılır və hansısa verilmiş şərtə görə həmin elementlər üzərində müəyyən əməl görülür. Biz baxdığımız halda daha əlverişli yol - iki sayda proseduradan istifadə etməkdir (onlardan birisi özü özünü çağıracaq – rekursiya effekti yaranır):

```
'*****Birinci prosedura *****
Public Sub StartProcl()
Dim oOutlook As New Outlook.Application
Dim oNameSpace As Outlook.NameSpace
Dim oChildFolder As Outlook.MAPIFolder
Set oNameSpace=oOutlook.GetNamespace("MAPI")
'Yuxarı səviyyənin hər bir qovluğunu yoxlayırıq və hər bir
'qovluq üçün DoFolder() prosedurasını işə salırıq.
For Each oChildFolder In oNameSpace.Folders
    DoFolder oChildFolder
Next
End Sub
'*****İkinci prosedura *****
'Bu prosedura bütün iç-içə olan qovluqların adını çıxarır və onların
'hər birisi üçün yenidən özü özünü icra edilməsi üçün çağırır.
Public Sub DoFolder(ByVal oFolder As MAPIFolder)
Dim oChildFolder As Outlook.MAPIFolder
```

```

For Each oChildFolder In oFolder.Folders
  Debug.Print oChildFolder.Name
'Hər bir iç-içə olan qovluq üçün DoFolder() prosedurasını işə salırıq.
'*****
  DoFolder oChildFolder
'*****
Next
End Sub

```

İndi isə VBA proqramlaşmasında vacib olan **Folders** kolleksiyasının və **MAPIFolder** obyektinin xassələri və metodları haqqında danışaq. **Folders** kolleksiyasının xassələri və metodları standartdır (məsələn, **Count**, **Item()**, **Add()**, **Remove()** v.s. kimi). Bunun əksinə **MAPIFolder** obyektinin çox sayda xassə və metodları vardır. Onlardan VBA proqramlaşmasında ən vacibləri haqqında izahlı siyahı aşağıda verilib:

- **AddressBookName** – istifadəçinin ünvan kitabında əks edilməsi üçün kontaktlar qovluğunun adını dəyişir. Xassəni başqa qovluqlara tətbiq etmək olmaz (səhv haqqında xəbər verilir).
- **CurrentView** – bu xassə **View** obyektini qaytarır (istifadəçi qovluğunun necə əks olmasını təyin edir). **MAPIFolder** obyektini üçün bu xassə yalnız oxumağa əlçatan olur.
- **DefaultItemType** – bu xassə qovluğun yaradıldığı zaman ona təyin edilən element tipini standart hala görə (poçt xəbərləşmələrini, kontaktları v.s.) sabit qiymət kimi qaytarır. Qovluq yaradılanda həmin xassə təyin edilir və bundan sonra dəyişdirilə bilmir. Adətən onu hansısa qovluqların emal edilmədən çıxarılmasında istifadə edirlər (məsələn kontaktlar qovluğunda poçt xəbərlərini axtarmamaq üçün).
- **DefaultMessageClass** – yuxarıdakı **DefaultItemType** xassəsinə eyniliklə bənzəyir, lakin bir fərqlə - informasiya ədəd kimi yox, sətir qiyməti kimi qayıdır. İstifadəçi hər iki xassədən istifadə edə bilər (hər ikisi də eynigüclüdür).
- **Description** – xassə sadəcə qovluğun təsviridir. Outlook-un qrafik interfeysində işlədikdə onu qovluğun xassəsindən əlçatan olur.
- **EntryID** – obyekt modeli **MAPI**-də olan **PR_ENTRYID** xassəsinin analoqudur: kodun qarşılıqlı əlaqələnməsi üçün nəzərdə tutulub. VBA proqramçıları ondan başqa məqsədlə daha çox istifadə edirlər: unikal xəbərlər identifikatoru kimi. Həmin identifikator istənilən **MAPI**-uyğunluqlu saxlancında (konteynerində) xəbər əmələ gəldikdə (məsələn, Outlook faylında və ya **Exchange Server** poçt qutusunda) avtomatik olaraq, yaradılır və yalnız başqa saxlanc yerinə keçirildikdə dəyişir.
- **FolderPath** – Outlook-un saxlanc yeri iyerarxiyasında qovluğa tam yolu təyin edir, məsələn, **\\Personal Folder\Inbox**.

- **Folders** - verilən qovluq üçün iç-içə olan qovluqların kolleksiyasını qaytarır (olduqca vacib xassədir). Yuxarıda dediyimiz kimi, çox hallarda qovluqlar ağacından dövrə tam keçidi təşkil etməkdə istifadə etmək olar.
- **InAppFolderSyncObject** – bu xassə **SyncObject** obyektinin **Application Folders** adlı xüsusi sinxronlaşma qrupunun işləməsi nəticəsində hansısa qovluğun sinxronlaşmasını təyin edir. Həmin sinxronlaşma qrupunun əsas xüsusiyyəti budur ki, yalnız həmin qrupu proqram səviyyəsində dəyişdirmək olar.
- **IsSharePointFolder** – kollektiv iş mühitində, kontakt və kalendar elementləri tərkibində olan, hansısa qovluğun **Windows SharePoint Services** verilənlər mənbəyində olub/olmamasını təyin edir. Bu xassə adətən yoxlamalarda istifadə edilir.
- **Items** – daha bir vacib xassədir: hansısa qovluğun elementləri üçün **Items** kolleksiyasını əlçatan edir. Qovluqlar elementlərinin işləməsi haqqında daha ətraflı gələn bölmədə danışacağıq.
- **Name** – sadə və eyni zaman çox vacib olan xassədir: qovluğun adını təyin edir.
- **ShowAsOutlookAB** – poçt xəbərləşməsi təşkil edildikdə, ünvanın seçimi pəncərəsində **Contacts** tipli elementlər tərkibində olan qovluğun içərisində onların göstərilməsini və ya göstərilməməsini təyin edir. Standart halda **Contacts** tipli qovluqlar üçün bu xassə **True** qiyməti ilə qurulmuş olur. Adətən yalnız o vaxt istifadə edilir ki, **Contacts** tipli elementlər tərkibində olan qovluq xidməti məqsədlə istifadə edilir.
- **ShowItemCount** – Outlook-un qovluq üçün olan xəbərlər sətirində nəyin göstərilməsini təyin edir (heç nə göstərilmir, bütün xəbərlərin ümumi sayı göstərilir və ya hələ oxunmamış xəbərlərin sayı göstərilir).
- **StoreID** – qovluq yerləşən, **MAPI**-uyğunluqlu saxlanılma yerinin unikal identifikatorunu təyin edir. Çox uzun sətir kimi görsənir (516 simvol). Fərqlənməni təyin etməkdə tətbiq etmək olar - məsələn, bir neçə sayda **Exchange** poçt qutusu üçün.
- **UnReadItemCount** – verilmiş qovluq üçün hələ oxunmamış xəbərlərin sayını təyin edir: yalnız oxumaq üçün əlçatan olur.
- **Views** – verilmiş qovluqla bağlı olan **View** obyektlərinin (görüntülər rejimləri) kolleksiyasını qaytarır.
- **WebViewOn** – qovluq üçün HTML səhifəsi formasında olan səhifənin görüntüsünü yaradır. Həmin səhifə tamamilə kənar element kimi, nə həmin qovluqla nə də Outlook elementləri ilə bağlı olmaya bilər (səhifə **WebViewURL** xassəsi ilə verilə bilər).

CopyTo(), **MoveTo()**, **Delete()**, **AddToFavorites()**, **Display()** metodlarının təyinatını sadəcə onların adından anlamaq olar. **GetExplorer()** metodu, qovluğun Outlook-un **Explorer** pəncərəsində göstərilməsini təmin edən, **Explorer** obyektini qaytarır.

14.6 Items kolleksiyası və Outlook elementlərinin obyektləri

Outlook.MailItem, Outlook.ContactItem obyektləri, VBA-dan proqram səviyyəsində xəbərləşmə və kontaktlarla olan işlərin təşkili

Adətən VBA proqramlaşmasında qovluqların elementləri ilə iş (poçt göndərmələri, kontaktlar, kalendar elementləri v.s. ilə) - Outlook obyekt modelini istifadə edən proqramların əsas hissəsidir. Outlook-da proqram səviyyəsində işlədikdə məhz qovluqların obyektləri ilə müxtəlif əməllər aparılmalı olur. Qovluqlar elementlərini əlçatan etmək üçün qovluq obyektinin **Items** xassəsi istifadə edilir (nəticədə **Items** kolleksiyası qaytarılır). Bu kolleksiyada həmin qovluğun bütün elementləri yerləşir. Outlook elementləri üçün bütöv bir obyekt nəzərdə tutulmayıb. Bunun əvəzinə Outlook-da istifadəçinin sərəncamına hər bir element üçün 16 sayda ayrı-ayrı obyektlər verilmişdi:

- **AppointmentItem** – azərbaycancaya tərcümədə “görüşlər + element (və ya hissə)” mənasına gəlir və yerinə yetirdiyi əməllər həmin məna daşıyır: adətən **Calendar** qovluğunda yerləşir.
- **ContactItem** – analogi olaraq, “kontaktlar + element (və ya hissə)”: mənasına gəlir və yerinə yetirdiyi əməllər həmin mənanı daşıyır: proqram səviyyəsində kontaktların yaradılmasında tez-tez istifadə edilir.
- **DistList** – göndərişlərin siyahısını təmsil edir və **Contacts** qovluğunda yerləşir.
- **DocumentItem** – bu ad Outlook-a saxlanmağa qoyulan və Outlook-un heç bir elementi ilə formatı uyğun gəlməyən istənilən fayla verilir. **DocumentItem** obyektini kimi, məsələn, Word sənədi, Excel kitabı, ZIP arxivi, Acrobat Reader PDF faylı, icra edilən EXE faylı təmsil oluna bilər – bir sözlə əməliyyat sisteminin istənilən faylı ola bilər. Outlook saxlancında (PST fayllarında, poçt qutularında və Exchange Server-in ümumi qovluqlarında) faylların yerləşdirilməsinə çox da meyl etmək lazım deyil – çünki buna görə Outlook-un adı elementlərinin əlçatan olması xeyli yavaşla bilər.
- **JournalItem** – bu obyekt gündəlikdəki yazını təmsil edir.
- **MailItem** – elektron poçtun xəbərini təmsil edir: proqramlaşmada tez-tez istifadə edilir.
- **MeetingItem** – elektron xəbərləşmənin xüsusi tipidir: görüşə dəvət əməllərini təmsil edir. Adətən elektron poçtun adı xəbərləri yerləşdiyi yerdə rast gəlir, məsələn, **Inbox** qovluğunda. Proqram səviyyəsində bu elementi yaratmaq olmur – o, yalnız müvafiq xəbər alınmasında avtomatik rejimdə yaranır və onu yalnız **Calendar**-dakı **Appointment** obyektini ilə göndərmək mümkündür.
- **NoteItem** - qeyd obyektidir (**Notes** qovluğunda yerləşir). Başqa elementlərdən öz xassələrinin və metodlarının az sayı ilə fərqlənir.
- **PostItem** – poçt xəbərlərinin daha bir xüsusi olan növüdür: ümumi qovluğa göndərilən xəbəri təmsil edir. Adı **MailItem** obyektindən onu fərqləndirən bunlardır:

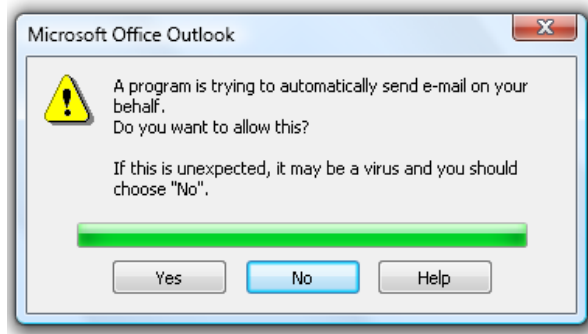
- yalnız ümumi qovluqlarda rast gəlir;
 - onun göndərilməsi üçün **Send()** metodu əvəzinə **Post()** metodu istifadə edilir.
- **RemoteItem** – həmçinin poçt xəbərlərinin xüsusi olan növüdür: bu obyektlər minimal sayda doldurulmuş xassəsi olan poçt xəbərlərini (real olaraq, yalnız xəbəri alan haqqında məlumat doldurulur - alınma tarixi, həcmi v.s.) və 256 simvoldan ibarət olan mətnin özünü təmsil edir. Bu obyektlər də yalnız avtomatik rejimdə yaranır: Outlook-a uzaqdan əlçatma qoşulması olan **Exchange Server** serverinin poçt qutusuna qoşulduqda baş verir. Əgər xəbər OST faylında yerləşibse (yəni MAPI ilə həmin **Exchange Server**-dən yüklənib və ya POP3/IMAP4 ilə alınıbsa), onda bu obyekt xəbər üçün heç bir zaman yaradılmır.
 - **ReportItem** – tamamilə xüsusi olan poçt xəbərini təmsil edir: xüsusi generasiya edilmiş xidməti məktubdur. “Çatdırılması mümkün deyil” tipli (non-delivery report) xəbərdir. Göndərişin gecikməsi və ya səhvin yaranmasında poçt serveri onu özü avtomatik rejimdə yaradır.
 - **TaskItem** – bu obyekt **Tasks** qovluğunda olan tapşırığı və ya məsələni təmsil edir.
 - **TaskRequestAcceptItem, TaskRequestDeclineItem, TaskRequestItem, TaskRequestUpdateItem** – bu obyektlər də xüsusi poçt göndərişlərinə aiddir: məsələnin kimə ötürülməsi haqqında “yazışmanı” təmsil edirlər. Həmçinin bu obyektləri də proqram səviyyəsində yaratmaq olmur – onları poçt serveri avtomatik generasiya edir.

Bununla belə qeyd etməliyik ki, əksər hallarda praktiki işlərdə **MailItem, ContactItem** və hərdən bir **DocumentItem** obyektləri daha çox Outlook VBA proqramlaşdırmasında istifadə edilir. Biz diqqətimizi məhz onların tətbiqinə cəmləyəcəyik. Daha bir vacib olan məqam budur: yuxarıda dediyimiz kimi, Outlook obyekt modelinə xüsusi məhdudiyətlər qoyulub: müxtəlif dərəcədə təhlükəli olan virusların bu obyekt modelində ziyan verici əməllərin qarşısını almaq üçün. Adətən bunlar elektron poçtdakı ünvanlar haqqında olan məlumat, göndərəninin adı, məktublارın mətni v.s. təmsil edən, ən vacib xassələrdir. Həmin məhdudiyətlər bu obyektlərin içərisində qurulur (daha sadə desək, Outlook-un əsas obyektlərinə): **AppointmentItem, ContactItem, MailItem** və **TaskItem**. Həmin obyektlərlə görülən əməllərə də məhdudiyətlər qoyulur: məsələn, məktublارın göndərilməsi ilə bağlı əməllərə: misal üçün Outlook vasitəsi ilə proqram səviyyəsində elektron xəbərin yaradılması və göndərilməsi çox asanlıqla tərtib edilən aşağıdakı VBA proqram kodu ilə icra edilə bilər:

```
Dim oOutlook As New Outlook.Application
Dim oMessage As Outlook.MailItem
'Xəbər obyektini yaradırıq
Set oMessage=oOutlook.CreateItem(olMailItem)
'Kime göndəririk?
oMessage.To="Administrator@nwtraders.msft"
'Xəbərin mövzusu
'*****
oMessage.Subject="VBA-dan salamlar!"
'*****
```

```
'Xəbərin mətni. Body xassəsinin istifadəsi - göstərir ki,
'biz xəbəri adi mətnlə göndəririk.
'Həmçinin HTML və ya RTF xəbərini göndərmək olar.
'*****
oMessage.Body="Xəbərin mətni"
'*****
'Əlavələri daxil edirik.
'*****
oMessage.Attachments.Add ("C:\installlog.txt")
'*****
'Xəbəri göndəririk.
'*****
oMessage.Send
'*****
```

Bu halda xəbərin göndərilməsi əvəzinə (virusların işə salınmasının qarşısı alınsın deyə) şəx. 14.6-da göstərilən xəbər pəncərəsi əmələ gəlir: bu məqsədlə pəncərədəki **YES** düyməsi bir neçə saniyə əlçatan olmayacaq.



Şəkil 14.6 Outlook-un xəbərdarlıq pəncərəsi

Bəs belə hallarda nə etmək lazımdır?

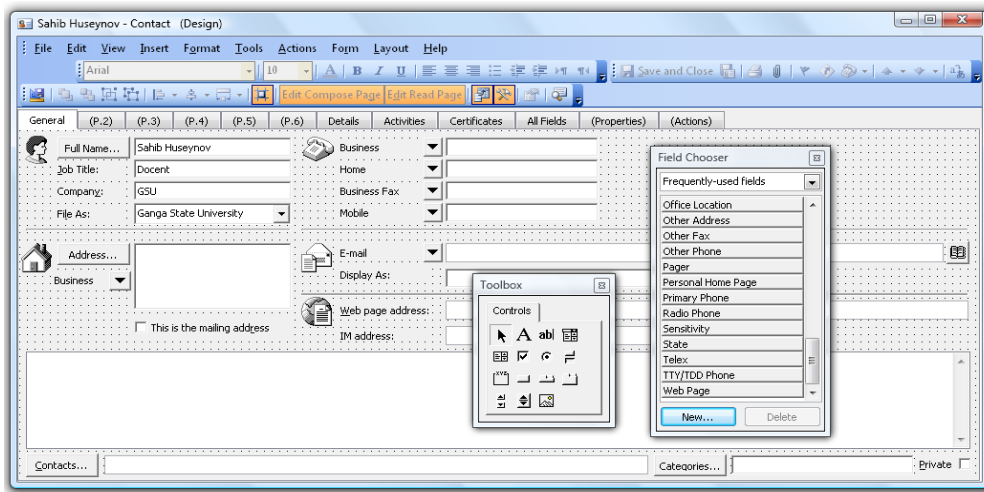
Həll variantları müxtəlif ola bilər:

1. **birinci variant** – Outlook-un proqram səviyyəsində açılması olmadan, Outlook-un öz VBA modulunda olan makrosundan xəbərin göndərilməsi. Bu halda pəncərə əmələ gəlməyəcək. Həll yolu bir qədər narahatdır, xüsusi ilə həmin proqram kodunu köçürdükdə - müxtəlif kompyuterlərdə işə salmaq mümkün olmur;
2. **ikinci variant** – həmin xəbəri avtomatik generasiya etməyən və tərtibatçılara VBA-dan heç bir problemsiz xəbərlərin göndərilməsini təmin edən, Outlook obyektlərinin əvəzləyicilərini istifadə etmək. Bu səmtdə ən geniş yayılmış vasitə - pulsuz olan **Outlook Redemption** alətidir. Onun haqqında informasiyanı <http://www.dimastr.com/redemption> saytıdan almaq və elə oradan da həmin aləti öz kompyuterinizə yükləmək olar. Əgər istifadəçi elektron poçtu bir serverdən göndəirsə - bu variant çox rahatdır. Bəzən bu alətin istifadəçi kompyuterinə yüklənməsində problemlər yaranır;
3. **üçüncü variant** – proqram səviyyəsində xəbərdarlıq pəncərəsinin tapılması və proqram səviyyəsində lazım olan düymənin basılması. Microsoft tərtibatçıları bunun qarşısını alıblar: VBA-dan və ya VBScript-dən bunu etmək mümkün deyil. Prinsip etibarı ilə həmin

əməlləri aşağı səviyyəli Windows API proqram interfeysindən etmək mümkündür. Başqa variant – bu imkanları əlçatan etmək üçün MAPI xəbərləri ilə işləmək mühitini yaradan xüsusi proqram interfeysini istifadə etməkdən ibarətdir. Lakin bu halda istifadəçi gerek VBA əvəzinə, məsələn, C++, Delphi, Java, Python v.s. kimi proqram təminatlarından istifadə etsin. Bundan əlavə, API (Application Programming Interface) ilə işləmək üçün istifadəçi gerek sistem proqramlaşdırması haqqında müəyyən səviyyədə hazırlığa malik olsun.

4. dördüncü variant – *hər bir kompyuterde Windows əməliyyat sisteminin bütün versiyalarında olan CDO, yəni Microsoft-un Collaboration Data Objects, obyekt kitabxanasının köməyi ilə məktublarnın proqram səviyyəsində göndərilməsini və emalını icra etmək.* Qeyd etməliyik ki, həllin bu variantı ən rahat və əlverişlidir. Bu haqda daha ətraflı kitabın 14.8-ci bölməsində danışacağıq.

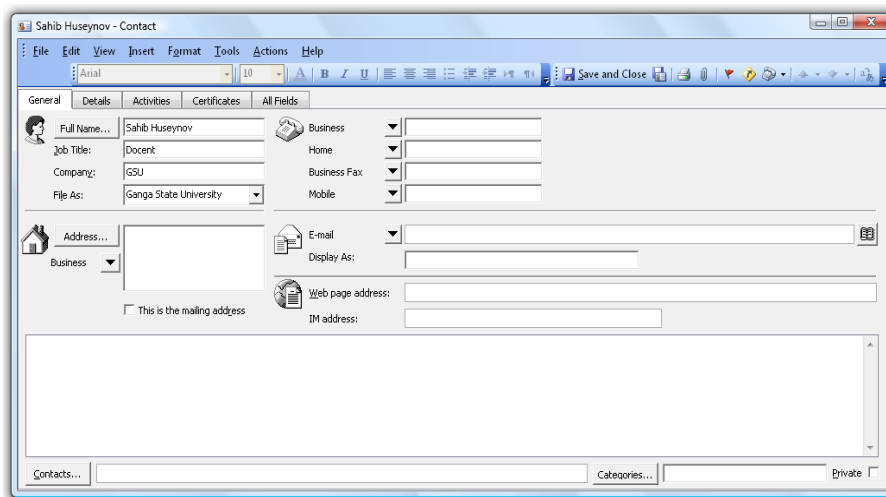
İndi isə Outlook elementləri obyektlərinin özü haqqında danışaq. Bu obyektlərin xassələri və metodları kifayət qədər çoxdur (məsələn, **ContactItem** obyektinin xassəsinin sayı təxminən 100 bərabərdir) və onların hamısının baxılması kitabın həcmindən daha artıq yer tələb edərdir. Bununla belə oxucu artıq obyektlərin xassələri və metodları ilə işləmək vərdişlərini mənimsəyibdir və buna görə burada adı çəkilən xassələrin və metodların proqramlaşdırmadakı tətbiqində hansısa problemlər yaranmamalıdır. Adətən yeganə yaranan sual bu olur: *elementin müəyyən atributuna hansı proqram xassəsi uyğun gəlir?* Təəssüf ki, Outlook-da makrorekorder yoxdur. Bu və ya digər xassənin adını VBA redaktorunun **Locals** pəncərəsində tapmaq olar. Həmin üsulun tətbiqini konkret misalda baxacağıq. Məsələn, təsəvvür edək ki, biz bu məsələni həll etməliyik: *ContactItem obyektinin hansı proqram xassəsinə GSU sahəsi uyğun gələ bilər?* Məsələnin həlli üçün birinci addımda biz gerek həmin xassənin doldurduğu kontaktı yaradaq (bax şəkl. 14.7 və şəkl. 14.8).



Şəkil 14.7 Outlook-da **Contacts** (kontaktlar) formasını yaratmaq və sazlamaq üçün nəzərdə tutulan pəncərə (dizayn rejimi).

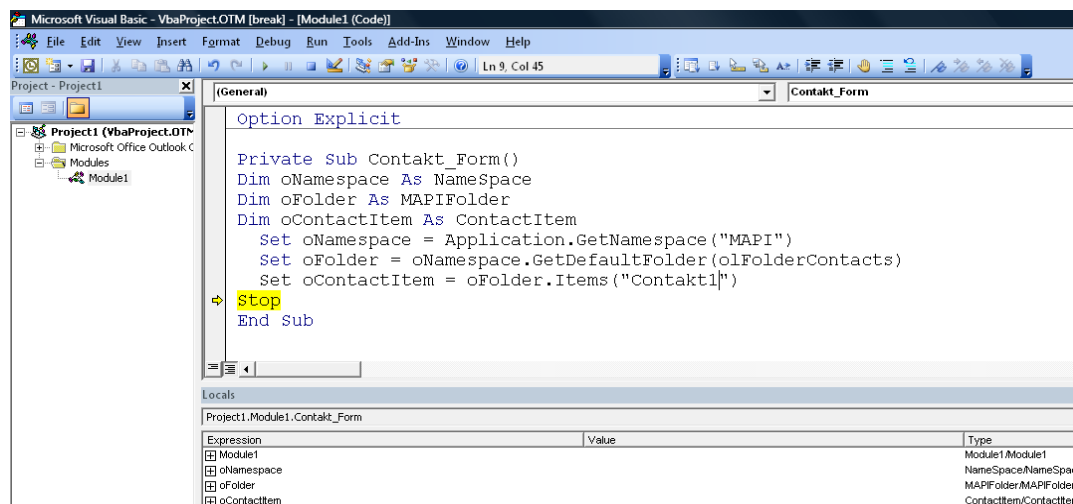
Sonrakı növbəti addımda gərək həmin elementin yaradıldığı proqram kodu yazılsın. Məsələn, əgər, həmin kontakt **Contacts** qovluğundadırsa, onda Outlook-dakı VBA proqram kodu bu cür yazıla bilər:

```
Dim oNamespace As NameSpace
Dim oFolder As MAPIFolder
Dim oContactItem As ContactItem
Set oNamespace=Application.GetNamespace("MAPI")
Set oFolder=oNamespace.GetDefaultFolder(olFolderContacts)
Set oContactItem=oFolder.Items("Contact1")
'Burada Stop əmri lazımı yerdə dayanmaq üçün yazılıb.
Stop
```



Şəkil 14.8 Outlook-da **Contacts** (kontaktlar) formasının istifadə üçün hazırlanmış forması (şək. 14.7-dəki formanın dizaynı pəncərəsinin **Forms**→**Run This Form** menyusu ilə avtomatik generasiya edilir).

Növbəti addımda həmin proqram kodunu Outlook-un VBA redaktorunda işə salmaq lazımdır. Proqramın icrası **stop** sətirində dayandıqda, redaktorun **View** menyusu ilə **Locals** pəncərəsi açılmalıdır. Bizim baxdığımız hal üçün aşağıdakı şək. 14.9-da göstərilən kimi, VBA redaktorunun **Locals** pəncərəsində dörd sayda obyekt görünəcək.



Şəkil 14.9 VBA redaktorunun **Locals** pəncərəsində kontakt obyektlərinin görsənməsi.

Son mərhələdə biz gərək `oContactItem` obyektinin düyümünü açaq və onun **CompanyName** adlı xassəsinin `GSU` qiyməti ilə qurulmuş xassəsinə tapıq, bax şəkl. 14.10.

Locals		
Project1.Module1.Contakt_Form		
Expression	Value	Type
CompanyLastFirstNoSpace	""	String
CompanyLastFirstSpaceOnly	""	String
CompanyMainTelephoneNumber	""	String
CompanyName	"GSU"	String
ComputerNetworkName	""	String

Şəkil 14.9 VBA redaktorunun **Locals** pəncərəsində bizim doldurduğumuz qiymətlə olan xassənin tapılması.

Bu qayda ilə başqa `Item` obyektlərinin lazımı xassələri tapıla bilər.

14.7 Outlook-un başqa obyektləri haqqında

Outlook.Explorer, Outlook.Inspector, Outlook.Search obyektləri, Outlook qovluqlarında axtarış

Biz Outlook obyekt modelinin yalnız əsas budağında yerləşən obyektlərlə tanış olduq: **Application**→**Namespace**→**Folders/MAPIFolder**→**elementlərin obyektləri**. Bu iyerarxik budaqdan kənarında bir neçə sayda əlverişli obyektlər də vardır: onlar haqqında bu bölmədə danışacağıq.

Explorer obyektini Outlook-un qrafik interfeysində qovluğun əks edilməsi üçün nəzərdə tutulub. Bununla siz əks etmədə qurulmuş olan və ya istifadəçi tərəfindən yaradılmış təsvirlərdən istifadə edə bilərsiniz (**View** obyektini ilə birləşdirin). Məsələn, proqram səviyyəsində **Contacts** tipli qovluğu açıb oradakı kontaktlar obyektlərini təşkilatlara görə sortlaşdırmaq üçün aşağıdakı VBA proqram kodundan istifadə etmək olar:

```
Dim oNamespace As NameSpace
Dim oFolder As MAPIFolder
Dim oExplorer As Explorer
Set oNamespace=Application.GetNamespace("MAPI")
Set oFolder=oNamespace.GetDefaultFolder(olFolderContacts)
Set oExplorer=oFolder.GetExplorer()
oExplorer.Display
oExplorer.CurrentView="Təşkilatlara görə"
```

Verilmiş qovluq üçün əlçatan olan bütün təsvirlərə proqram səviyyəsində baxmaq üçün bu proqram kodunu tətbiq etmək olar (təsvirlər üçün **View** obyektləri tətbiq eldir):

```
Dim oNamespace As NameSpace
Dim oFolder As MAPIFolder
Dim oView As View
Set oNamespace=Application.GetNamespace("MAPI")
Set oFolder=oNamespace.GetDefaultFolder(olFolderContacts)
For Each oView In oFolder.Views
Debug.Print oView.Name
```

Next

Explorer obyektinin daha bir vacib imkanı vardır – istifadəçinin hansı elementləri seçdiyinin təyin edilməsi. Məsələn, cari pəncərədə hal-hazırda istifadəçinin seçdiyi məktublارın mövzusu haqqında olan informasiyanı almaq üçün bu VBA proqram kodundan istifadə etmək mümkündür:

```
For Each Item In Application.ActiveExplorer.Selection
  If TypeName(Item)="MailItem" Then Debug.Print Item.Subject
Next
```

Explorer obyektinin eyni gücə malik olan oxşarı **Inspector** obyektidir. O da Outlook pəncərəsini təmsil edir. Fərq burasındadır ki, bu pəncərədə element (poçt xəbəri, kontaktı v.s.) baxılması və redaktə edilməsi üçün açıq olur. Bu obyektə istinadın alınması eyniliklə **Explorer** obyektində olan kimi həyata keçirilir. Müqayisədə bu cür istinadlardan az istifadə edirlər - əsasən yoxlamalarda (məsələn, bu cür yoxlamada: Outlook-da proqram səviyyəsində emal edilən element aktivdirmi?).

Outlook-un obyekt modelinin ayrıca budağında axtarışları icra edən obyektlər vardır. Ümumiyyətlə, Outlook-da axtarışların təşkil edilməsi addım başı rast gələn əməllərdir. Məsələn, adi halda istifadəçi öz şəxsi axtarışını bu cür təşkil edə bilər – sadəcə bütün qovluqlar və onların tərkibində olan bütün elementlərin seçilməsini təşkil etmək (seçim yuxarı səviyyədə olan qovluqlardan başlayır). Buna oxşar misal - **Folders** kolleksiyası və **MAPIFolder** obyektləri ilə işləməyə həsr olmuş nümunə kitabın 14.5-ci bölməsində gətirilib. Lakin daha effektiv və bir qədər mürəkkəb olan üsul budur – axtarış üçün Outlook-un öz qurulmuş obyektlərindən istifadə etmək. Bunun üçün iki obyektəndən istifadə edirlər - **Search** və **Results**, **Application** obyektinin **AdvancedSearch()** metodu və **AdvancedSearchComplete** hadisəsi (çünki axtarış asinxron rejimdə icra edilir – eyni zamanda 100-dən artıq axtarış proqram səviyyəsində və həmçinin qrafik interfeysdən işə salmaq mümkündür). Dediklərimizin reallaşdırılması bu cür ola bilər - başlanğıcda **AdvancedSearch()** metodunu işə salırıq. **AdvancedSearch()** metodu isə dörd sayda parametri qəbul edir:

- **Scope** – axtarış diapazonunu (əslində qovluğun adını) təmsil edir. Əgər axtarış üçün qovluğun harada yerləşdiyi məlum deyilsə, onda qovluğun adını proqram səviyyəsində almaq olar: **Namespace** və **MAPIFolder** obyektlərinin **Folders** xassəsinin tətbiq etmək lazım olacaq. Qovluğun adında xüsusi simvollar olduqda, onda onları bir qat dırnaq arasına işarələrin arasında yazmaq lazımdır.
- **Filter** - ən mürəkkəb parametrdir: hansı şərtlərlə nəyin axtarılmasını təyin edir. Sintaksis filtrlərin sintaksisinə uyğun gəlməlidir: SQL Server-də sorğularda olan kimi (**LIKE** operatoru axtarış üçün istifadə edilə bilər – belə olduqda, axtarış daha əlverişli mühitdə aparılır). Sütunların adlarının təyin edilməsi bir o qədər də asan iş deyil – ehtimal çoxdur ki, əlavə məlumat üçün belə hallarda istifadəçi Microsoft-un ünvanlarla bağlı xüsusi saytlarına gərək müraciət etsin:

1) http://msdn.microsoft.com/library/en-s/cdosys/html/_cdosys_schema_mailheader.asp

2) http://msdn.microsoft.com/library/en-us/cdosys/html/_cdosys_schema_httpmail.asp.

Məsələn, tutaq ki, biz "Hesabat" sözü rast gələn bütün mövcud olan məktublarda axtarış aparmalıyıq. Onda program kodunun filtri ilə bağlı sətiri bu cür yazılmalıdır:

```
"urn:schemas:mailheader:subject LIKE '%Hesabat%'"
```

Digər halda, əgər "Hesabat" mövzusu axtarılırsa, onda həmin sətir bu cür yazılacaq:

```
"urn:schemas:mailheader:subject='Hesabat'"
```

- **SearchSubFolders** - əgər axtarışı üstəlik bütün iç-içə olan qovluqlar üzrə aparmaq lazımdırsa, onda bu parametrin qiyməti **True** ilə qurulmalıdır.
- **Tag** – sadəcə axtarışın sətir identifikatorudur: yalnız bir neçə sayda axtarış işə salındıqda və nəticələrin bir-birindən fərqləndirilməsində istifadə edirlər.

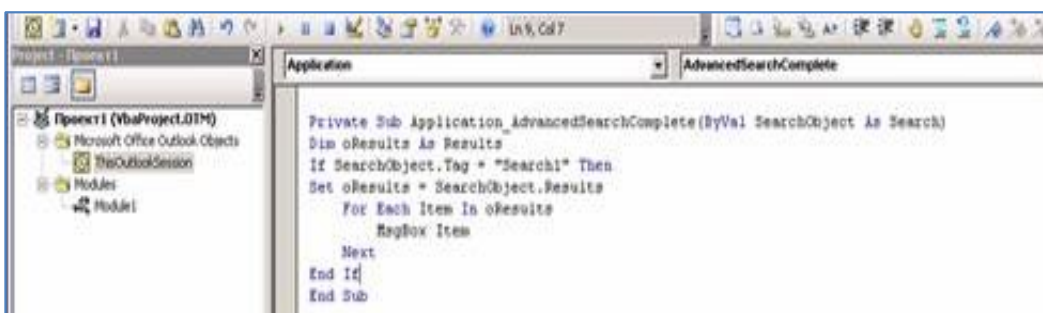
Nəticədə **AdvancedSearch()** metodu bizə **Search** obyektini qaytara bilər:

```
Dim oSearch As Search
```

```
Set oSearch=Application.AdvancedSearch("Inbox",_  
"urn:schemas:mailheader:subject LIKE '%Hesabat%'"_  
",True, "Search1")
```

Axtarış asinxron rejimdə aparılır deyərək **AdvancedSearch()** metodu çağırduğumuz proseduranı, axtarışın qurtarmasını gözləməyərək, dayanmadan icra edəcək. Bu halda biz gərək axtarışın sonunu **AdvancedSearchComplete** hadisəsi ilə izləyək. Bunun üçün:

1. **Project Explorer**-də **Microsoft Office Outlook Objects** konteynerini açırıq;
2. **Application** obyektini və **AdvancedSearchComplete()** metodu seçilərkən (kitabın 14.7-ci bölməsinə bax), VBA redaktorunun yuxarı hissəsində **ThisOutlookSession** sətiri hadisələr və obyektler siyahısının üstündən mausla iki dəfə şıqqıldadıırıq, şəkl. 14.10.



Şəkil 14.10 **AdvancedSearchComplete** xassəsindən istifadə edirik.

Həmin hadisədə **Search** obyektini və ondan törəyən **Results** obyektini istifadə edilir. **Item** – Outlook-un VBA mühitində olan adi obyektlərindəndir (baxdığımız halda **MailItem** obyektləridir) – bu obyektlərin bütün xassələri və metodları istifadəçi üçün əlçatan olur.

15. SƏRBƏST İŞLƏMƏK ÜÇÜN TAPŞIRIQLAR (məsləhətlər və cavablarla birgə): MS OFFICE PROQRAMLARINDA VBA PROQRAMLAŞMASI

15.1 SƏRBƏST İŞLƏMƏK ÜÇÜN TAPŞIRIQ 1:

Word-ə avtomatik rejimdə mətnin daxil edilməsi üçün makrosun yazılması

MƏSƏLƏNİN ŞƏRTİ:

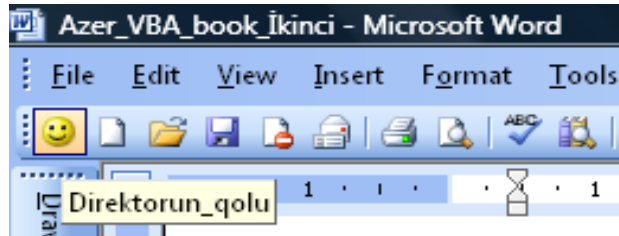
Müəssisədə bir gün ərzində əməkdaş mühasibata sərəncamları aparıb verməlidir. Hər bir sərəncam isə aşağıdakı şək. 15.1-də göstərilən sətirlərə analoji olaraq tamamlanmalıdır.

Müəssisənin baş müdiri:	M.M.Məmmədov
Məsul icraçı: A.T.Qafarova	
E-Mail: aaqəmbərov@simurq.az.com	
Tel: +99422 08534451200	

Şəkil 15.1 Avtomatlaşdırılması nəzərdə tutulmuş mətn sətirləri.

TAPŞIRIQ:

1. Makrorekorder köməyi ilə elə makros yazılmalıdır ki, şək. 15.1-də göstərilən sətirləri, avtomatik olaraq, yaratsın (icraçıya aid hər nə məlumat varsa, dəyişdirib Sizə aid verilənlərlə əvəz edin).
2. Tərtib etdiyiniz makros Sizin yaratdığınız bütün Word sənədlərindən əlçatan olmalıdır.
3. Tərtib edilən makros hər bir yaradılan (və ya istifadədə olan) Word sənədindəki baş menyusunda yerləşdirilən hansısa düymənin (məsələn, “şən düymə” formasında) basılması nəticəsində avtomatik olaraq, işə salınmalıdır.
4. Üstəlik mausu həmin düyməyə yaxınlaşdırdıqda “Direktorun_qolu” adlı köməkçi yazı, üzərək, əmələ gəlməlidir, şək. 15.2.




Şəkil 15.2 İstifadəçinin tapşırıqda olan tələbləri yetirmək üçün yaradılmış düymə və üzərək qalxan köməkçi yazı.

5. Yeni Word sənədini yaradın, sənəddə makrosu tərtib edərək, onu işə salın və düzgün işləməsinə əmin olun.

TAPŞIRIQ 1-in CAVABI: Word-ə avtomatik rejimdə mətnin daxil edilməsi üçün makrosun yazılması

Həlli: (tapşırığın lazımcıca sərbəst yerinə yetirilməsi üçün nəzəri və praktiki cəhətdən kitabın 1.5 və xüsusilə 1.6-cı bölmələrinin yenidən təkrar edilməsi vacibdir).

1. Word-də yeni sənədi açın. Həmin yeni yaradılmış sənədin **Tools** menyusunda **Macros**→**Record New Macro** seçin. Əmələ gələn **Record Macro** pəncərəsinin **Macro Name** sahəsində “Direktorun_qolu” mətnini (dırnaq arası işarəsi olmadan daxil etmək lazımdır) daxil edin. Əmin olun ki, həmin pəncərənin **Store Macro in** sahəsi “All Documents (Normal.dot)” qiyməti ilə qurulub və həmin **Record Macro** pəncərəsində **OK** düyməsini basın.
2. **Tools**→**Customize** menyusundan **Customize** pəncərəsində **Commands** qurmasında Normal.NewMacros. Direktorun_qolu elementini çəkib ataraq, idarəetmə panelini istədiyiniz sahəsində yerləşdirin. Sonra mausun sağ düyməsi ilə həmin Normal.NewMacros.Direktorun_qolu elementi üzərində bir dəfə şıqqılatmaq lazımdır (nəzərə alınmalıdır ki, bu halda **Customize** pəncərəsi, həmin əməllər yerinə yetirilərkən, bağlı olmamalıdır). Əmələ gələn kontekst menyusundan **Change Button Image** əmri seçilməlidir. Əmələ gələn seçim pəncərəsindən, düymə şəklini seçmək lazımdır (məsələn, “Şən düymə” şəklini ). Bir daha da mausun sağ düyməsi ilə həmin element üzərində şıqqıldadaraq, yaranan kontekst menyusundan **Default Style** seçimini edin. İndi **Customize** pəncərəsini bağlamaq olar: bunun üçün həmin pəncərədə **Close** düyməsini basın. Nəticədə, avtomatik olaraq, makros yazılmağa başlayacaq.
3. Tapşırıqda təyin edilmiş mətni Word sənədində daxil edin, sonra isə **Stop Recording** düyməsini basın (və ya **Tools**→**Makros** menyusundan **Stop Recording** əmrini seçin).
4. Əlavə yeni Word sənədini yaradın və orada da yeni yaradılmış idarəetmə düyməsinin düzgün işləməsinə əmin olun.

15.2 SƏRBƏST İŞLƏMƏK ÜÇÜN TAPŞIRIQ 2: *Makrosun redaktə edilməsi*

MƏSƏLƏNİN ŞƏRTİ:

Sərbəst iş 1-də yaratdığınız makrosu elə dəyişdirin ki, o, məsul icraçının soy adını soruşsun.

TAPŞIRIQ:

1. makros kodunun başlanğıcında olan sətərə (birinci sətir olan `Selection.TypeText` üstündən) aşağıdakı sətirləri əlavə edin:

```
Dim sInput As String
```



```
sInPut=InputBox("Məsul icraçının soy adını əlavə_edin", "Verilənlərin sorğu edilməsi")
```

2. bu sətiri isə:

```
Selection.TypeText Text:=("Məsul icraçı:A.T.Qafarova")
```

(soy ad Sizin soy adınız olmalıdır) aşağıdakı sətir ilə dəyişin:

```
Selection.TypeText Text:=("Məsul icraçı:" & sInPut)
```

Dəyişdirilmiş makrosu yaddaşda saxlayın VBA kod redaktorunu bağlayın və əmin olun ki, makros indi yeni qayda ilə işləyir (yəni tapşırıq 1-də tələb olan mətnin avtomatik olaraq çap etməkdən əvvəl icraçının adını xüsusi **InputBox** dialoq pəncərəsinə daxil etmək üçün tələb edir).

Qeyd: oxucu daxil etdiyi VBA kodunun anlanmasının çətin olması üçün narahat olmasın. Oxucu üçün bu tapşırığın əsas məqsədi – VBA proqram kodu redaktorunda işləmək vərdişlərini əldə etmək.

TAPŞIRIQ 2-nin CAVABI (Makrosun redaktə edilməsi):

Həlli: (tapşırığın lazımcına sərbəst yerinə yetirilməsi üçün nəzəri və praktiki cəhətdən kitabın 1.5 , 1.7, 2.1 – 2.4 bölmələrinin yenidən təkrar edilməsi vacibdir).

1. Word-ü açın və klaviaturanın **<Alt>+<F11>** düymələrini basın. Açılmış **Microsoft Visual Basic** redaktoru pəncərəsində **Project Explorer** pəncərəsini tapın, orada **Normal**→**Modules**→**NewMacros** düyümünü açın və klaviaturanın **<F7>** düyməsini basın.
2. Yaratdığınız **Sub** Direktorun_qolu() proseduranı tapın və lazım olan dəyişiklikləri edin. Proseduranın ümumi mətni, məsələn, aşağıdakı kimi ola bilər (nəzərdə tutun ki, makrorekorderin proqram kodu heç də Sizin yaratdığınıza oxşamaya bilər – hər şey ondan asılıdır ki, makrorekorderdə Siz hansı əməlləri yerinə yetirmişdiniz):

```
Sub Direktorun_qolu()  
,  
' Direktorun_qolu proqram kodu makrosdur  
' Makros 11.11.2012 tarixində yazılıb  
,  
Dim sInPut As String  
sInPut=InputBox("Məsul icraçının soy adını əlavə edin",_  
"Verilənlərin sorğu edilməsi")  
Selection.TypeText Text:=" Müəssisənin baş müdiri:" &_  
vbTab & vbTab & vbTab & " M.M.Məmmədov  
Selection.ParagraphFormat.Alignment=wdAlignParagraphCenter  
Selection.TypeParagraph  
Selection.ParagraphFormat.Alignment=wdAlignParagraphLeft  
Selection.TypeText Text:=("Məsul icraçı:" & sInPut)  
Selection.TypeParagraph  
Selection.TypeText Text:="E-Mail: atqafarova@simurq.az.com"  
Selection.TypeText Text:="Tel: +99422 08534451200"  
End Sub
```

3. Dəyişikləri yadda saxlamaq üçün klaviaturanın <Ctrl>+<S> düymələrini basın. Word-ə qayıtmaq üçün indi də klaviaturanın <Alt>+<Q> düymələrini basın. Makrosu işə salın və icra edin. Əmin olun ki, makros tapşırığın şərtinə uyğun işləyir.

15.3 SƏRBƏST İŞLƏMƏK ÜÇÜN TAPŞIRIQ 3:

Dəyişənlər və operatorlarla işləmək verdişləri (Word və Excel-də VBA vasitələri ilə qarşılıqlı əməllərin aparılması)

MƏSƏLƏNİN ŞƏRTİ:

Excel kitabında və Word sənədində qarşılıqlı aparılan əməllərin avtomatlaşdırılması tələb olunur:

1. Excel-də xəbərləşmə pəncərəsinə həmin Excel kitabının hansısa səhifəsindəki, məsələn, A1, A2 və A3 hücrələrində olan qiymətləri çıxarmaq lazımdır;
2. sonra, yeni Word sənədini açaraq, sənədin baş hissəsində "Göndərilən mətn" sətirini çap etmək tələb olur.

TAPŞIRIQ:

- Yeni Excel kitabını yaradın və onu yadda saxlayın, məsələn, bu adla saxlayın: LabVariablesOperators.xls. Həmin kitabın birinci səhifəsində olan A1, A2 və A3 hücrələrinə istənilən (və ya tələb olan) qiymətləri daxil edin.
- Excel-də Visual Basic redaktorunu açaraq, bu kitabda yeni standart modul (məsələn, Modul1) yaradın, bax kitabın 2.2 və 2.3 bölmələrinə.
- VBA redaktorunun **Tools|References** menyusu ilə layihənizə **Microsoft Word Object Library** kitabxanasına istinad alın.
- Yaradığınız standart modulda aşağıdakı VBA proqram kodunu daxil edin:

```
Public Sub FromExcelToWord()  
'bu kod Excel-dəki A1, A2 və A3 hücrələrində olan  
'qiymətləri xəbər pəncərələrinə çıxarır və sonra yeni  
'Word sənədini açaraq, onun baş hissəsində "Göndərilən  
'mətn" sətirini çap edir  
    MsgBox Range("A1").Text  
    MsgBox Range("A2").Text  
    MsgBox Range("A3").Text  
    Set oWord=CreateObject("Word.Application")  
    oWord.Visible=True  
    Set oDoc=oWord.Documents.Add()  
    oDoc.Activate  
    oWord.Selection.TypeText "Göndərilən mətn"  
End Sub
```

- Yığdığınız VBA proqram kodunun etibarlı və düzgün işləməyinə tam əmin olun.
- Yuxarıdakı VBA proqram kodunda elə dəyişiklik edin ki, Word-dün baş hissəsində "Göndərilən mətn" sətiri əvəzinə, Excel-dəki A1, A2 və A3 hücrələrində olan həmin qiymətlər birgə çap edilsin.

TAPŞIRIQ 3-ÜN CAVABI (Dəyişənlər və operatorlarla işləmək verdişləri):

Həlli: (tapşırığın lazımcına sərbəst yerinə yetirilməsi üçün nəzəri və praktiki cəhətdən kitabın 2.2, 3.1 – 3.3 bölmələrinin yenidən təkrar edilməsi vacibdir).

Yekun, dəyişdirilmiş VBA proqram kodu bu cür yazıla bilər:

```
Public Sub FromExcelToWordAnswer()  
'bu kod Excel-dəki A1, A2 və A3 hücrələrində olan  
'qiymətləri xəbər pəncərələrinə çıxarır və sonra yeni  
'Word sənədini açaraq, onun baş hissəsində "Göndərilən  
'mətn" sətirini çap edir  
Dim sA1, sA2, sA3, sText As String  
sA1=ThisWorkbook.Worksheets(1).Range("A1").Text  
sA2=ThisWorkbook.Worksheets(1).Range("A2").Text  
sA3=ThisWorkbook.Worksheets(1).Range("A3").Text  
sText=sA1+" "+sA2+" "+sA3  
Set oWord=CreateObject("Word.Application")  
oWord.Visible=True  
Set oDoc=oWord.Documents.Add()  
oDoc.Activate  
oWord.Selection.TypeText sText  
End Sub
```

15.4 SƏRBƏST İŞLƏMƏK ÜÇÜN TAPŞIRIQ 4:

Şərti keçid operatorlarının tətbiqi

MƏSƏLƏNİN ŞƏRTİ:

Excel kitabının səhifəsindəki hücrələrdə VBA şərti keçid operatorlarının istifadəsi ilə aşağıda verilən əməlləri avtomatlaşdırmaq tələb olunur:

TAPŞIRIQ:

1. Yeni Excel kitabını yaradıb ona (məsələn, LabCondConstructions.xls) ad verin.
2. Həmin Excel kitabında Excel Visual Basic redaktorunu açıb, orada yeni standart modul Modul1 yaradın.
3. Standart Modul1-də aşağıdakı VBA kodunu daxil edin.

```
Public Sub IfThenSub()  
'Bu VBA proqram kodu Excel kitabının birinci səhifəsinin  
'A1 hücrəsinə "Siz düyməni basdınız: 13" mətn qiymətini,  
'xəbər pəncərəsində hansı düymənin basıldığından asılı  
'olaraq, qoyur.  
Dim nResult As Integer  
nResult=MsgBox("Düyməni basın", vbYesNo, _  
"Xəbər pəncərəsi")  
ThisWorkbook.Worksheets(1).Range("A1").  
Value="Siz düyməni basdınız:" & nResult  
ThisWorkbook.Worksheets(1).Range("A1").  
Columns.AutoFit  
End Sub
```

4. Həmin kodu işə salın və onun düzgün işləməyinə əmin olun.

QEYD: Xəbər pəncərəsində hansı düymənin basılması nəticəsində hansı qiymətin qaytarılması haqqında olan məlumatı **MsgBox()** standart funksiyası haqqında olan sorğu materiallarından tapmaq olar.

TAPŞIRIQ 4.a:

Yuxarıdakı proseduranı elə dəyişdirin ki, hücrəyə ədədlər əvəzinə basılmış düymənin sətir qiyməti yazılsın (məsələn, "Siz düyməni basdınız: "). Buna nail olmaq üçün **If...Then...Else** şərti keçid operatorlar konstruksiyasından istifadə edin.

TAPŞIRIQ 4.b:

1. Yığdığınız prosedurada aşağıdakı sətiri

```
nResult=MsgBox("Düyməni basın", vbYesNo, "Xəbər pəncərəsi")  
bu birisi ilə
```

```
nResult=MsgBox("Düyməni basın", vbAbortRetryIgnore, "Xəbər_ pəncərəsi")  
əvəz edin
```

2. Yığdığınız prosedurada elə dəyişiklik edin ki, nəticədə xəbər pəncərəsində hansı düymənin basıldığına görə, A1 hücrəsinə "Buraxmaq", "Təkrar etmək" və ya "İmtina etmək" mətn qiymətləri avtomatik yazılsın. Buna nail olmaq üçün **select...Case** sintaktik konstruksiyasından istifadə edin.

TAPŞIRIQ 4-ün CAVABI (Şərti keçid operatorlarının tətbiqi):

Həlli: (tapşırığın lazımcıca sərbəst yerinə yetirilməsi üçün nəzəri və praktiki cəhətdən kitabın 3.4, 3.5, 3.8.3 və 3.8.4 bölmələrinin yenidən təkrar edilməsi vacibdir).

Yekun, dəyişdirilmiş VBA proqram kodları bu cür yazıla bilər:

Tapşırıq 4.a-nın həlli:

```
Public Sub IfThenSubAnswer()  
Dim nResult As Integer  
nResult=MsgBox("Düyməni basın", vbYesNo, "Xəbər_ pəncərəsi")  
If nResult=6 Then  
sResult="Əlbəttə,"  
ElseIf nResult=7 Then  
sResult="Xeyir"  
Else  
sResult="Naməlum düymə"  
End If  
ThisWorkbook.Worksheets(1).Range("A1")._ Value="Siz düyməni basdınız:" & sResult  
ThisWorkbook.Worksheets(1).Range("A1")._ Columns.AutoFit  
End Sub
```

Tapşırıq 4.b-nin həlli:

```

Private Sub SelectCaseAnswer()
    nResult=MsgBox("Düyməni basın",_
vbAbortRetryIgnore, "Xəbər pəncərəsi")
    Select Case nResult
        Case 3
            sResult="İmtina etmək"
        Case 4
            sResult="Təkrar etmək"
        Case 5
            sResult="Buraxmaq"
        Case Else
            sResult="Naməlum düymə"
    End Select
    ThisWorkbook.Worksheets(1).Range("A1")._
Value="Siz düyməni basdınız: " & sResult
    ThisWorkbook.Worksheets(1).Range("A1")._
Columns.AutoFit
End Sub

```

15.5 SƏRBƏST İŞLƏMƏK ÜÇÜN TAPŞIRIQ 5:

Dövrü hesablamalar operatorlar strukturlarının istifadə edilməsi

MƏSƏLƏNİN ŞƏRTİ:

Word sənədində mətnlə işlədikdə istifadəçi ilə dialoq rejimində bir sıra rutin xarakterli dövrü əməllərin yerinə yetirilməsi lazım olur. Buna görə, vaxta qənaət etmək üçün müdiriyyət həmin əməllərin proqram səviyyəsində avtomatlaşdırılmasını tələb edir.

TAPŞIRIQ:

- Word sənədinin VBA redaktorunda standart modul yaradaraq, həmin modulda **ForNextSub()** adlı elə prosedura yaradın ki, həmin sənəddə o, 1-dən 10 qədər ədədləri əks edən xəbərləri ardıcıl olaraq 10 sayda xəbər pəncərəsində istifadəçiyə göstərsin.
- Həmin proqram modulunda, məsələn, **ForEachSub()** adlı elə prosedura yaradın ki, o, xəbər pəncərələrinə Sizin yaratdığınız Word sənədinizdə də daxil etdiyiniz hər bir sözü ardıcıl olaraq çıxarsın. Nəzərə alın ki, **ThisDocument.Words** konstruksiyası ilə sənəddəki bütün sözlərin kolleksiyasını (yəni əslində toplusunu – ingilis terminologiyası ilə "Array") almaq olur. Hər bir sözün qiymətini isə həmin kolleksiyanın **Text** xassəsinin köməyi ilə almaq mümkündür.

İŞİN YERİNƏ YETİRİLMƏSİ ÜÇÜN MƏSLƏHƏTLƏR:

- Yeni Word sənədini yaradın və ona, məsələn, **LabLoops.doc.** adını verin və sonra həmin sənəddə bir neçə cümlədən ibarət mətn yazın.
- Həmin Word sənədinin Visual Basic redaktorunu açın və orada yeni standart modulu yaradın.

TAPŞIRIQ 5-in CAVABI (Dövrü hesablamalar operatorlar strukturlarının istifadə edilməsi):

Həlli: (tapşırığın lazımınca sərbəst yerinə yetirilməsi üçün nəzəri və praktiki cəhətdən kitabın 3.8.1 və 3.8.2 bölmələrinin yenidən təkrar edilməsi vacibdir).

Yekün, tərrib edilmiş, VBA proqram kodları bu cür yazıla bilər:

Tapşırıq 5.a-nın həlli (iki varintda tərrib edilə bilər):

```
Public Sub ForNextSub()  
    For i=1 To 10  
        MsgBox i  
    Next  
End Sub
```

və ya bu cür :

```
Public Sub ForNextSub2()  
    Dim i As Integer  
    i=1  
    Do While i<=10  
        MsgBox i  
        i=i+1  
    Loop  
End Sub
```

Tapşırıq 5.b-nın həlli:

```
Public Sub ForEachSub()  
    For Each oWord In ThisDocument.Words  
        MsgBox oWord.Text  
    Next  
End Sub
```

15.6 SƏRBƏST İŞLƏMƏK ÜÇÜN TAPŞIRIQ 6:

Prosedura və funksiyaların tərrib edilməsi

MƏSƏLƏNİN ŞƏRTİ:

Word sənədində aşağıda verilən əməllərin avtomatlaşdırılması tələb olur: bunun üçün VBA prosedurası və funksiyası tərrib edilməlidir.

TAPŞIRIQ:

1. **Normal.dot** şablonunun **NewMacros** modulunda elə yeni funksiya (məsələn, adını `fMultiply()` qoya bilərsiniz) yaradın ki, o, aşağıdakı əməllərin avtomatlaşdırılmasını icra edə bilsin:
 - daxil edilən parametrlər kimi iki ədədi qəbul etsin;
 - həmin ədədləri bir birinə vurub alınan nəticəni qaytarsın.
2. **Normal.dot** şablonunun **NewMacros** modulunda yeni prosedurasını (məsələn, adını `AutoNew()` qoyun) aşağıda verilmiş VBA proqram kodu əsasında yazın:

```
Public Sub AutoNew()  
    Dim nMult1 As Integer  
    Dim nMult2 As Integer  
    Dim nResult As Integer
```

```
nMult1=CInt(InputBox("Birinci ədədi daxil edin: "))
nMult2=CInt(InputBox("İkinci ədədi daxil edin: "))
nResult=10
Selection.InsertAfter nResult
Selection.Collapse wdCollapseEnd
```

End Sub

3. Tərtib edilmiş `AutoNew()` prosedurasını elə dəyişdirin ki, `nMult1` və `nMult2` dəyişənlərin qiymətlərini `fMultiply()` funksiyasına ötürmək mümkün olsun və həmin funksiyadan `nResult` dəyişəni üçün qiymət alsın (həmin qiymət 10 qiyməti əvəzinə istifadə edilməlidir).
4. Word-də yeni sənəd yaradın və əmin olun ki, yaratdığınız prosedura və funksiya burada lazımı qaydada və etibarlı işləyir.
5. Yaratdığınız `AutoNew()` adlı prosedura sonrakı işinizə mane olmasın deyə, bütün proseduranın proqram kodunu şərh kimi təyin edin.

İŞİN YERİNƏ YETİRİLMƏSİ ÜÇÜN MƏSLƏHƏTLƏR:

- Yeni Word sənədini yaradın və klaviaturanın **<Alt>+<F11>** düymələrini basın.
- VBA-nın **Project Explorer** pəncərəsində **Normal**→**Modules** düyümünü açın.
- Açılmış düyümdə **NewMacros** sətiri üzərində mausla iki dəfə şıqqıldadın.
- **NewMacros** modulunda `fMultiply()` funksiyası üçün aşağıdakı kod sətirlərini əlavə edin:

```
Public Function fMultiply(nM1 As Integer, nM2 As Integer)
    fMultiply=nM1*nM2
End Function
```

- `AutoNew()` adlı yaradılmış proseduranı (VBA proqram kodunu) tamam şərh kimi təyin etmək üçün, bütün kodu mausla seçin (**Public Sub** `AutoNew()` və proseduranın sonunda yerləşən **End Sub** sətiri ilə birgə) və **Edit** alətlər panelində olan **Comment Block** düyməsini basın.

TAPŞIRIQ 6-nın CAVABI (Prosedura və funksiyaların tərtib edilməsi):

Həlli: (tapşırığın lazımcına sərbəst yerinə yetirilməsi üçün nəzəri və praktiki cəhətdən kitabın 3.9.1

- 3.9.15 bölmələrinin yenidən təkrar edilməsi vacibdir).

Tərtib ediləsi `AutoNew()` adlı proseduranın kodu bu cür yazıla bilər (dəyişdirilmiş kod daha böyük, və maili şriftlə göstərilib):

```
Public Sub AutoNew()
    Dim nMult1 As Integer
    Dim nMult2 As Integer
    Dim nResult As Integer
    nMult1=CInt(InputBox("Birinci ədədi daxil edin: "))
    nMult2=CInt(InputBox("İkinci ədədi daxil edin: "))
```



```

nResult=fMultiply(nMult1, nMult2)
Selection.InsertAfter nResult
Selection.Collapse wdCollapseEnd
End Sub

```

15.7 SƏRBƏST İŞLƏMƏK ÜÇÜN TAPŞIRIQ 7:

VBA proqramlarında Windows Script Host xarici obyekt modelinin tətbiqi

MƏSƏLƏNİN ŞƏRTİ:

Word sənədində aşağıda verilən əməllərin avtomatlaşdırılması tələb olur (bunun üçün VBA-nın **Windows Script Host** xarici obyekt modelini tətbiq edin).

TAPŞIRIQ:

1. 6-cı tapşırıqda (bax kitabın 15.6 bölməsinə) yaradılmış Word sənədinin VBA layihəsinə **Windows Script Host Object Model** kitabxanasına istinadı əlavə edin;
2. **WSH()** adı ilə yeni yaratdığınız prosedurada **WScript.Network** və **WScript.Shell** proqram obyektlərini yaradın, onların xassələri və metodlarına baxıb tanış olun.
3. **WSH()** prosedurasına elə kod əlavə edin ki, o, aşağıdakı əməlləri avtomatlaşdırıa bilə idi:
 - **WScript.Network** obyektinin **ComputerName**, **UserName** və **UserDomain** xassələrinin qiymətlərini mətn dəyişənlərinə qəbul etsin və o qiymətləri həmin Word sənədinə çap etsin.

Qeyd: *Word sənədinə mətn dəyişəninin çapını icra edən VBA kodu aşağıdakı kimi yazıla bilər:*

```

ThisDocument.Activate
Selection.TypeText Mənim_mətn_dəyişənim

```

- **WScript.Shell** obyektinin **Run()** metodunu çağırırsın və ona "calc" adlı yeganə mətn parametrini ötürsün.
- **WScript.Shell** obyektinin **Environment** xassəsini istifadə edərək, mühitdəki dəyişənlər haqqında informasiya daşıyan mətn dəyişənləri kolleksiyasını yaratsın.
- Həmin kolleksiyada olan mətn dəyişənlərinin bütün qiymətlərini Word sənədinə çap etsin.

Qeyd: *yaradılan kolleksiya və onun elementlərində istifadə edilən dəyişənlər üçün Variant tipi təyin olmalıdır.*

İŞİN YERİNƏ YETİRİLMƏSİ ÜÇÜN MƏSLƏHƏTLƏR:

- Word-ü açıb yeni sənəd yaradın (adi üsul ilə, proqram səviyyəsində yox).
- VBA redaktorunda həmin sənədin layihəsini açın.
- Həmin layihədə **WSH()** adlı yeni prosedura yaradın.

TAPŞIRIQ 7-nin CAVABI (VBA proqramlarında Windows Script Host xarici obyekt modelinin tətbiqi):

Həlli: (tapşırığın lazımcına sərbəst yerinə yetirilməsi üçün nəzəri və praktiki cəhətdən kitabın 4.1 - 4.6 bölmələrinin yenidən təkrar edilməsi vacibdir).

WSH () adlı prosedurasının proqram kodunun son görkəmi aşağıda göstərilən kimi ola bilər:

```
Public Sub WSH()
  '** Dəyişənlərin tipini təyin edirik.*****
  Dim oNetwork As WshNetwork
  Dim oShell As WshShell
  Dim sComputer As String
  Dim sDomain As String
  Dim sUser As String
  Dim oColl As Variant
  Dim sEnv As Variant
  '***** Obyektləri yaradıırıq. *****
  Set oNetwork=CreateObject("WScript.Network")
  Set oShell=CreateObject("Wscript.Shell")
  '*****
  '** Wscript.Network obyektinin xassələrinin *****
  '** qiymətlərini alırıq və Word-də çap edirik.*****
  sComputer=oNetwork.ComputerName
  sDomain=oNetwork.UserDomain
  sUser=oNetwork.UserName
  ThisDocument.Activate
  Selection.TypeText sComputer & vbCrLf & sDomain & _
  vbCrLf & sUser & vbCrLf & vbCrLf
  '***** Wscript.Shell obyektinin Run metodunu çağırırıq.
  oShell Run "Calc"
  '*****
  '** Mühit dəyişənlərinin kolleksiyasını alırıq.
  Set oColl=oShell.Environment
  '*****
  '** Kolleksiyanın hər bir elementini çıxarıırıq
  For Each sEnv In oColl
    Selection.TypeText sEnv & vbCrLf
  Next
  '*****
  '** İndi daha yaxşı olar ki, operativ yaddaşdan
  '** yaradılmış obyektləri silək. *****
  Set oNetwork=Nothing
  Set oShell=Nothing
  '***** PROQRAMIN SONU *****
End Sub
```

15.8 SƏRBƏST İŞLƏMƏK ÜÇÜN TAPŞIRIQ 8:

Proqram səviyyəsində idarəetmə elementlərinin istifadə edilməsi

15.8.1 VBA ilə Excel-də sadə formanın yaradılması

MƏSƏLƏNİN ŞƏRTİ:

Excel kitabında aşağıda təsvir olan informasiyanı avtomatlaşdırılmış halda istifadəçinin nəzərinə təqdim etmək lazımdır (bunun üçün VBA forma obyektlərindən istifadə edilməlidir).

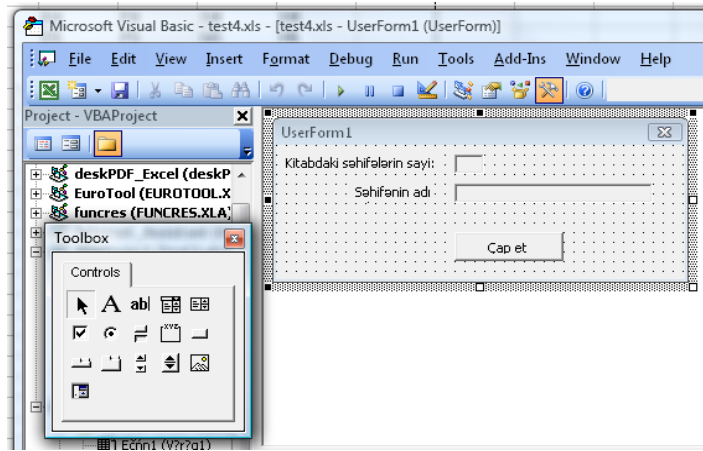
TAPŞIRIQ:

1. Excel kitabında hansı sayda səhifələrin olması haqda istifadəçiyə forma vasitəsi ilə xəbər vermək lazımdır.
2. Yalnız hansı səhifədə verilənlər varsa və o səhifə aktivdirsə, onda onun adını istifadəçiyə göstərmək lazımdır.
3. Həmin səhifədəki verilənləri oxuyub çap etmək lazımdır.

TAPŞIRIĞIN YERİNƏ YETİRİLMƏSİ ÜÇÜN MƏSLƏHƏTLƏR: Excel-i açıb yeni kitab yaradın (adi üsul ilə, proqram səviyyəsində yox); həmin kitabın hansısa səhifəsində və hansısa müəyyən diapazonda olan hücrələrə istənilən qiymətləri daxil edin; kitabdakı VBA redaktorunu açıb orada yeni forma modulunu yaradın; həmin formada tapşırıqın şərtlərinə uyğun idarə elementlərini əlavə edin; yaradılmış formanın VBA kodunu avtomatik generasiya edin; Açılan VBA kod redaktorunda məsələdə qoyulan şərtlərə müvafiq idarəetmə elementlərinin reaksiyasını təyin edin; tərtib edilmiş VBA kodunu sınaqdan keçirin və düzgün işləməsinə əmin olun.

Həlli: (tapşırıqın lazımınca sərbəst yerinə yetirilməsi üçün nəzəri və praktiki cəhətdən kitabın 5.1 - 5.12 bölmələrinin yenidən təkrar edilməsi vacibdir).

1. MS VBA Excel-də **UserForm** dialoq rejiminə daxil olub lazımı formanı tərtib etmək (şək. 15.3);



Şəkil 15.3 VBA Excel-də **UserForm** dialoq rejiminə daxil olunması.

2. Məsələnin tələblərinə uyğun həmin **UserForm** rejimində yaradılan formadan (VBA-nın **View**→**Cod** əsas menyusundan) formanın proqram koduna keçmək lazımdır.
3. Açılmış VBA redaktorunda aşağıdakı proqram kodunu yığın:

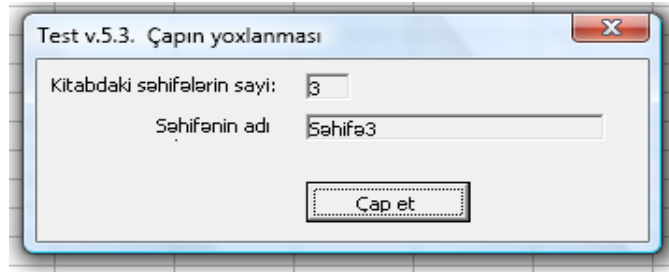
```
Const NBook="test3.xls"  
Private Sub cmdPRN_Click()  
    NameBook=Application.Workbooks(NBook).ActiveSheet.Name  
    lblN.Caption=NameBook  
    Worksheets(3).Range("A1:H10").Select  
    Worksheets(3).PrintOut  
End Sub  
Private Sub Label1_Click()
```

```

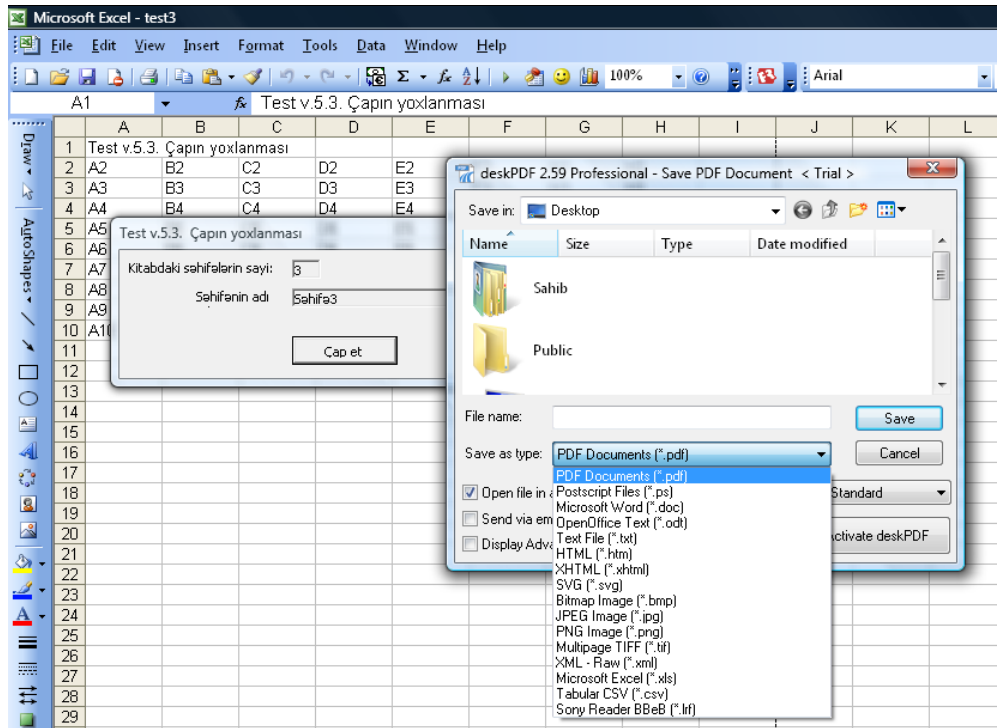
End Sub
Private Sub UserForm_Initialize()
Me.Width=250           'pəncərənin eni
Me.Height=100         'pəncərənin hündürlüyü
Me.Caption="Test v.5.3. Çapın yoxlanması"
With Application.Workbooks(NBook).Activate
say=.Sheets.Count 'Kitabdaki səhifələrinin sayı
'Formaya səhifələrin sayının çıxarılması
lblKoll1.Caption=kol
.Worksheets(3).Activate
NameBook=.Worksheets(3).Name
End With
lblN.Caption=NameBook
End Sub

```

4. Makrosu sınaqdan keçirib işlək şəkildə gətirmək (yeni kodun generasiya etdiyi formanın alınmasını əldə edin, şəkl. 15.4). VBA makrosu diapazonundakı qiymətləri çapa çıxaracaq.



Şəkil 15.4. MS VBA redaktorunda makros işlədikdə yaranmış formanın görüntüsü.



Şəkil 15.5. Excel kitabında tərtib edilmiş VBA makrosu işlədikdə əmələ gələn dialoq pəncərəsi aktiv olan səhifədəki diapazon qiymətlərini çapa çıxarmağa hazırlaşır.

15.8.2 Excel-də VBA ilə tərkibində kompleks təyinatlı idarəetmə elementləri olan formaların yaradılması

MƏSƏLƏNİN ŞƏRTİ:

Müəssisədə işçilərin siyahısını avtomatlaşdırılmış rejimdə emal etmək lazımdır. Excel kitabında həmin verilənlər – kitabın birinci səhifəsinin birinci sütununda yerləşən hücrələrin mətn qiymətlərindən ibarətdir. Avtomatlaşdırılmış emal VBA proqramlaşması ilə icra edilməlidir. Lazım olduqda bəzi əməllər Word proqramına da ötürülə bilər (məsələn, əməllərin avtomatik hazırlanması). Prosesi idarə etmək üçün idarəetmə elementləri ilə birgə xüsusi VBA formasını yaradın.

TAPŞIRIQ:

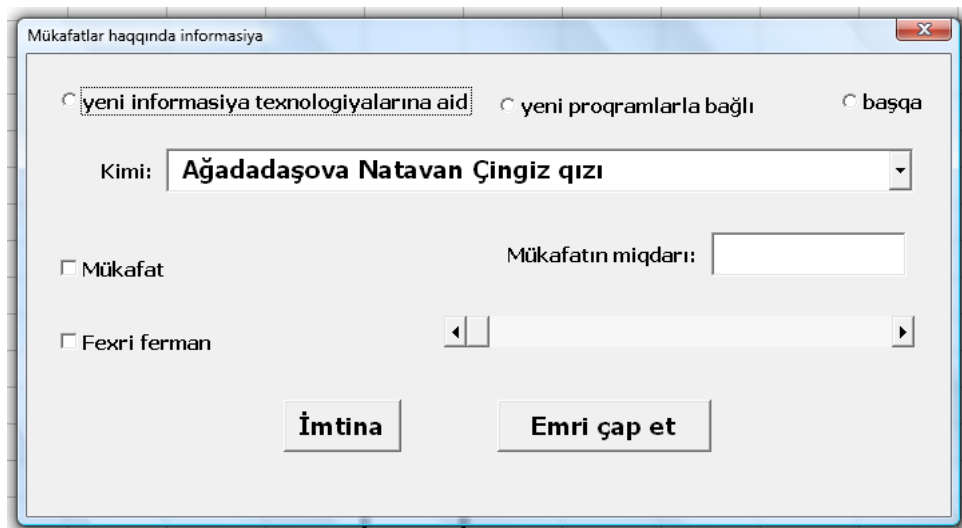
Excel-də verilənləri proqram səviyyəsində emal etmək üçün yaradılan istifadəçidən ötrü tərtib edilən formanı elə qurmaq lazımdır ki, həmin forma şərhələrlə seçilmiş blokda olan dəyişənlərə qabaqcadan təyin edilmiş qiymətlərin verilməsi əvəzinə, istifadəçi ilə dialoq yaratsın: istifadəçinin idarə etməsi nəticəsində lazımı qiymətlərin daxil edilməsi mümkün olsun. Tapşırığın yerinə yetirilməsi aşağıda verilən tələblərə uyğun olmalıdır:

1. Tərtib ediləsi proqramda **sSebeb** adlı dəyişənin qiyməti (məsələn, hansı səbəbə görə əməkdaş mükafatlanır v.s.) üç mümkün olan qiymətlər arasından seçilməlidir: "yeni informasiya texnologiyalarına aid", "yeni proqramlarla bağlı" və istifadəçi özü formanın mətn sahəsindən daxil etdiyi qiymət. Bu əməli icra etmək üçün formada üç sayda keçirici idarəetmə elementindən və mətn sahəsindən istifadə edin. Əgər istifadəçi iki digər keçiricilərindən birisini seçibsə, onda mətn sahəsi gizlənmiş vəziyyətdə olmalıdır. Standart halda "yeni informasiya texnologiyalarına aid" qiyməti götürülməlidir.
2. Proqramdakı digər vacib dəyişən **sSAA** (əməkdaşın soy adı, adı və atasının adını təmsil edir) formada kombinə edilmiş siyahı idarəetmə elementinin köməyi ilə istifadəçi tərəfindən seçilməlidir. Həmin kombinə edilmiş siyahıya avtomatik olaraq Excel səhifəsindəki A sütununun bütün boş olmayan hücrələrin qiymətləri yerləşdirilməlidir. Standart halda sütunun birinci hücrəsində olan "Məhəmmədov Məhəmməd Məhəmməd oğlu" mətn qiyməti yerləşdirilməlidir.
3. Proqramın digər vacib dəyişənləri **bFlagPremia** (əməkdaşa mükafatın verilməsini təmsil edir) və **bFlagFerman** (əməkdaşa fəxri fərmanın verilməsini təmsil edir) gerek formada yerləşən idarəetmə bayraqçıqlarının vəziyyətindən asılı olaraq, qiymətləri təyin edilməlidir: uyğun olaraq "Mükafat" və "Fəxri ferman". Standart halda hər iki bayraqçıq təyin edilmiş halda olmalıdır.

Vacib qeyd: (nəzərə alınmalıdır ki, VBA-nın bütün versiyalarında Azərbaycan dilində olan "Ə" hərfi VBA-da "?" işarəsi kimi generasiya edilir, buna görə proqramda Türk dilinə oxşar kimi, "E" hərfini "Ə" hərfi əvəzinə istifadə bilərsiniz – formada idarə etmək üçün bu çatışmazlıq bir o qədər də vacib amil deyil). Görünür bu məsələ ilə bağlı Azərbaycan mütəxəssisləri Microsoft-a dövlət səviyyəsində öz iradlarını bildirməlidir...

Əgər istifadəçi hər iki bayraqcığı açıq qoyubsa, onda həmin əməkdaşa bu cür xəbərdarlıq verilməlidir: "Nə mükafat, nə də fəxri fərman seçilməyib". Nəticədə heç bir əməl baş verməməlidir.

- İstifadəçi gerek **nMükafatMiqdar** (mükafatın miqdarının pul qiyməti ilə verilməsini təmsil edir və 0, ..., 1000 Azm diapazonundan seçilə bilər) dəyişəninin qiymətini formada diyirlənmə zolağı ilə və ya mətn sahəsi ilə verə bilsin. Əgər formada "Mükafat" adlı bayraqcıq açıqdırsa, onda diyirlənmə zolağı və mətn sahəsi istifadəçinin nəzərindən gizlənmiş vəziyyətdə olmalıdır. Diyirlənmə zolağının gediş imkanının minimal qiyməti 1 Azm olmalıdır. Standart halda mükafatın qiyməti 1 Azm bərabər olmalıdır.
- Formaya daha bir idarəetmə elementini əlavə edin: "İmtina". Bu düymə ilə istifadəçi cari formanı bağlamalıdır. Həmçinin formadakı bu düymə klaviaturanın <Esc> düyməsi basıldıqda işə salınmalıdır.
- Tərtib ediləsi formanın ümumi görkəmi, məsələn, şəkl. 15.6-da olan kimi görünə bilər. Formanın başlığında "Mükafatın haqqında informasiya" mətni görünməlidir:



Şəkil 15.6 Hazır formanın fərz edilən maketi (oxucu öz dizaynı ilə də formanı yarada bilər).

TAPŞIRIĞIN YERİNƏ YETİRİLMƏSİ ÜÇÜN MƏSLƏHƏTLƏR:

- Yeni Excel kitabını yaradın və ona, məsələn, Order.xls adını verin. Kitabın birinci səhifəsində A1 : A5 sütununun hücrələrini (fərz edilən) əməkdaşların soy adı, adı və atasının adı ilə doldurun, şəkl. 15.7.

	A
1	Məhəmmədov Məhəmməd Məhəmməd oğlu
2	Cəfərov Nurməmməd İbrahim oğlu
3	Səfərova Səfura Dilqəm qızı
4	Mirqasımova Şəhla İsmail qızı
5	Gülməmmədov Ağabala Mirmaxmud oğlu
6	Ağadadaşova Natavan Çingiz qızı
7	

Şəkil 15.7 Excel səhifəsində müəssisə şöbəsinin əməkdaşlarının siyahısı.

- Visual Basic redaktorunu açıb **Project Explorer** pəncərəsində mausun sağ düyməsi ilə **"This book"** elementi üzərində bir dəfə şıqqıldadı və açılan kontekst menyusundan **View Code** əmrini seçin.
- Kod redaktorunun pəncərəsində aşağıdakı kodu yığın:

```
'Excel kitabı açılında Uf1 formasını göstəririk.
Private Sub Workbook_Open()
    Uf1.Show
End Sub
'Word-də əmri çap edən xüsusi prosedura.
Public Sub DocWrite(sPovod As String, sFio_
As String, bFlagPremia As Boolean, bFlagGramota_
As Boolean, nSummaPremii As Long, sOtvIsp_
As String)
    Dim oWord As Word.Application
    Dim oDoc As Word.Document
    Set oWord=CreateObject("Word.Application")
    Set oDoc=oWord.Documents.Add()
    oWord.Visible=True
    oDoc.Activate
With oWord.Selection
    .TypeText "Əmr"
    .Style="Sərlövhə 1"
    .ParagraphFormat.Alignment=wdAlignParagraphCenter
    .TypeText vbCrLf
    .Style="Adi"
    .TypeText vbCrLf
    .TypeText "Bakı şəhəri" & Space(90) & Date
    .TypeText vbCrLf
    .TypeText vbCrLf
    .TypeText "Nail olduğu müvəffəqiyyətlərə görə"_
& sPovod & "təltif olunsun" & sFio & ":"
    .TypeText vbCrLf
    If bFlagPremia Then
        .TypeText vbTab & "- pul mükafatı_
məbləğindədə " & nSummaPremii & " manatla"
    End If
    If bFlagGramota Then
        .TypeText ";"
        .TypeText vbCrLf
        .TypeText vbTab & "- fəxri fərmanla."
    Else
        .TypeText "."
```

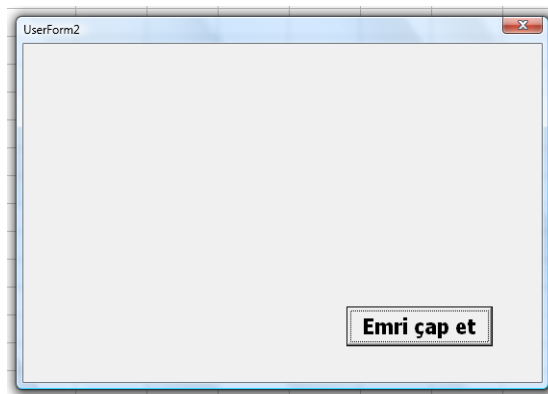


```

End If
.TypeText vbCrLf
.TypeText vbCrLf
.TypeText vbCrLf
.TypeText vbCrLf
.TypeText "Baş direktor" & vbTab & _
vbTab & vbTab & "B.B.Babaşov
.ParagraphFormat.Alignment=wdAlignParagraphCenter
.TypeParagraph
.TypeText vbCrLf
.TypeText vbCrLf
.ParagraphFormat.Alignment=wdAlignParagraphLeft
.TypeText Text:=("Məsul icraçı:" & sOtvIsp)
.TypeParagraph
End With
End Sub

```

4. **Project Explorer** pəncərəsində Order.xls layihəsi üzərində mausun sağ düyməsi ilə bir dəfə şıqqıldadın və əmələ gələn kontekst menyusundan **Insert | UserForm** təlimatını seçin. Yaratdığınız formanın obyektini mausla seçin və klaviaturanın <F4> düyməsini basın. Redaktorun **Properties** pəncərəsində həmin formanın **Name** xassəsi üçün **UF1** qiymətini daxil edin. **Toolbox**-dan seçib formaya bir sayda yeganə - **CommandButton1** idarəetmə elementini yerləşdirin. Bu düymənin **Caption** xassəsi üçün "Emri çap et" qiymətini daxil edin (dırnaq arası işarələrini daxil etməyin). **Font** xassəsində isə yazılası mətnin istənilən parametrini özünüz seçə bilərsiniz (şriftin növünü, onun ölçüsünü, rəngini v.s.). Formada düymənin ölçülərini və yerləşdirildiği koordinatları dəyişdirin, şəkl. 15.8-də göstərilən kimi:



Şəkil 15.8 Forma (hələlik üzərində yeganə idarəetmə elementi - düymə ilə).

5. Formada **CommandButton1** düyməsi üzərində mausla bir dəfə şıqqıldadın və əmələ gələn kontekst menyusundan **View Code** təlimatını seçin. Obyekt üçün yaranan hadisəvi proseduraları tərtib etməkdən ötrü nəzərdə tutulan kod redaktorunda **click** hadisəsi olan yerdə aşağıdakı kodu yığın və yaddaşda saxlayın:

```

Private Sub CommandButton1_Click()
  Dim sPovod As String
  Dim sFio As String
  Dim bFlagPremia As Boolean
  Dim bFlagGramota As Boolean

```

```

Dim nSummaPremii As Long
Dim sOtvIsp As String
'Formadan verilənləri götürürük.
sPovod="yeni informasiya texnologiyalarına aid"
sFio="Məhəmmədov Məhəmməd Məhəmməd oğlu"
bFlagMukafat=True
bFlagFerman=True
nMakfatMiqdar=100000
sOtvIsp="S.D.Səfərova"
'Verilənlərin daxil edilməsinin sonu
Call ThisBook.DocWrite(sPovod, sFio, _
bFlagMukafat, bFlagFerman, nMakfatMiqdar, sOtvIsp)
End Sub

```

6. Formanı VBA-da icra etmək üçün **Ran** təlimatı ilə proseduranı işə salın və əmin olun ki, forma həqiqətən də yeni Word sənədini yaradır və onun baş səhifəsində tərtib edilmiş əmri çap edir.

Həlli: (tapşırığın lazımcına sərbəst yerinə yetirilməsi üçün nəzəri və praktiki cəhətdən kitabın 5.1 - 5.12 bölmələrinin yenidən təkrar edilməsi vacibdir).

Tapşırığın 1-ci punktuna dair (keçiricilər və mətn sahəsi idarəetmə elementlərinin qurulması):

1. **Project Explorer** pəncərəsində formanın **UF1** obyektı üzərində iki dəfə mausla şıqqıldadın. Sonra **ToolBox**-da **Label** obyektı üzərində mausla şıqqıldadıb bu elementi formanın yuxarı hissəsində yerləşdirin. Yaradılmış **Label1** idarəetmə elementi üzərində mausun sağ düyməsi ilə bir dəfə şıqqıldadıb yaranan kontekst menyusunda **Properties** təlimatını seçin. **Caption** xassəsinin qiymətini "Nəyə görə:" mətn qiyməti ilə qurun və **Font** xassəsi ilə lazım olan şrifti seçin (şriftin ölçüsünü, rəngini v.s.).
2. **ToolBox**-da **OptionButton** idarəetmə elementi üzərində mausla şıqqıldadıb, elementi formada lazım olan yerə qoyun. Bu əməliyyatı iki dəfə təkrar edin.
3. Birinci keçirici idarəetmə elementinin xassəsini açın. Həmin xassədə **Name** qiymətini **optNail** ilə əvəz edin. **Caption** xassəsinin qiymətində bu mətn qiymətini əlavə edin: "yeni informasiya texnologiyalarına aid". İkinci keçiricinin həmin xassəsində "yeni proqramlarla bağlı" qiymətini əlavə edin və **Name** xassəsində **optTetbig** qiymətini yazın. Üçüncüsünün **Caption** xassəsində "başqa" mətn qiymətini əlavə edin və **Name** xassəsində isə **optBashga** qiymətini yazın.
4. **ToolBox**-da **TextBox** idarəetmə elementi üzərində mausla şıqqıldadıb elementi formadakı müəyyən bir yerə aparın. Elementin **Name** xassəsində **txtBashga** qiymətini yazın.
5. Formadakı boş bir yerdə mausun sağ düyməsi ilə bir dəfə şıqqıldadıb yaranan kontekst menyusundan **View Code** təlimatını seçin. **UserForm** üçün **Initialize** hadisəsini seçin və onun üçün bu kodu daxil edin:

```
optOsvoenie.Value=True  
txtDrugoe.Visible=False
```

6. **optBashga** keçiricisinin **Change** hadisəsi üçün bu kodu əlavə edin:

```
If optDrugoe.Value=True Then  
    txtDrugoe.Visible=True  
Else  
    txtDrugoe.Visible=False  
End If
```

7. **CommandButton1** idarəetmə elementinin **Click** hadisəsinə keçin və bu kod sətiri

```
sSebeb="yeni texnologiyalarla aid"
```

əvəzinə aşağıdakı kodu onun üçün əlavə edin:

```
If optNail.Value=True Then sSebeb="yeni informasiya_  
texnologiyalarına aid"  
If optTetbig.Value=True Then sSebeb="yeni proqramlarla bağlı"  
If optBashga.Value=True Then sSebeb=txtBashga.Value
```

8. Formanı icra etmək üçün işə salın. Əmin olun ki, əmr doğrudan da Word-də çap olundu və hər şey normal qaydada işləyir.

Tapşırığın 2-ci punktuna dair (kombinəli siyahı idarəetmə elementinin qurulması):

1. Formada daha bir idarəetmə elementini yerləşdirin: "Kimi:" yazısı ilə **Label1** idarəetmə elementini (şriftini də qurun).
2. **Toolbox**-da **ComboBox** idarəetmə elementi üzərində mausla şıqqıldadıb elementi formada yerləşdirin. Yaradılan **ComboBox** idarəetmə elementinə **cbSAA** adını təyin edin.
3. **UserForm** formasının **Initialize** hadisəsi üçün proqram kodunu açın və aşağıdakı kodu onun üçün əlavə edin:

```
Dim oColumn As Range  
Dim oCell As Range  
Set oColumn=Columns("A")  
For Each oCell In oColumn.Cells  
    If oCell.Value < > "" Then  
        cbSAA.AddItem oCell.Value  
    End If  
Next  
cbSAA.Value="Məhəmmədov Məhəmməd Məhəmməd oğlu"
```

4. **CommandButton1** idarəetmə elementinin **Click** hadisəsi üçün aşağıdakı kod sətiri əvəzinə

```
sSAA="Məhəmmədov Məhəmməd Məhəmməd oğlu"
```

bu kodu əlavə edin:

```
sSAA=cbSAA.Value
```

5. Formanı icra etmək üçün işə salın. Əmin olun ki, hər şey normal qaydada işləyir.

Tapşırığın 3-cü punktuna dair (bayraqçıqlar idarəetmə elementlərinin qurulması):

1. **ToolBox** köməyi ilə formada iki **CheckBox** idarəetmə elementini yerləşdirin. Birincisinin **Name** xassəsində **chMukafat** qiymətini və **Caption** xassəsində "Mukafat" mətn qiymətini təyin edin. İkinci idarəetmə elementinin **Name** xassəsində **chFerman** qiymətini və **Caption** xassəsində "Ferman" mətn qiymətini təyin edin.
2. Formanızın **UserForm** obyektini üçün **Initialize** hadisəsini tapın. Onun üçün program kodunu açın və orada aşağıdakı kod sətirlərini əlavə edin:

```
chMukafat.Value=True
```

```
chFerman.Value=True
```

3. **CommandButton1** idarəetmə elementinin **Click** hadisəsi üçün aşağıdakı kod əvəzinə

```
bFlagPremia=True
```

```
bFlagGramota=True
```

bu kodu əlavə edin:

```
bFlagMukafat=chMukafat.Value  
bFlagFerman=chFerman.Value  
If bFlagMukafat=False And bFlagFerman=False Then  
  MsgBox "Nə mükfüt, nə də fərman verilməyib!"  
Exit Sub  
End If
```

4. Formanı icra etmək üçün işə salın. Əmin olun ki, hər şey normal qaydada işləyir.

Tapşırığın 4-cü punktuna dair (diyirlənmə zolağı və əvəzləyici idarəetmə elementlərinin qurulması):

1. Forma üzərinə daha bir idarəetmə elementini yerləşdirin: "Mükafatın miqdarı" yazısı ilə olan **Label** idarəetmə elementini və ona **lb1Cem** adını təyin edin.
2. Yanında daha bir mətn sahəsini yerləşdirin və onun **Name** xassəsinə **txtCem** adını təyin edin.
3. Onların yanında **ScrollBar** idarəetmə elementini yerləşdirin və onun xassələrini aşağıdakı kimi təyin edin:

Xassə	Qiymət
Name	sbSum
Min	0
Max	100000
SmallChange	1

4. **sbCem** idarəetmə elementinin **Change** hadisəsi üçün bu kodu əlavə edin:

```
txtCem.Value=sbCem.Value
```

5. **txtCem** idarəetmə elementinin **Change** hadisəsi üçün bu kodu əlavə edin:

```
sbCem.Value=CInt(txtCem.Value)
```

QEYD: *Bu cür kod potensial təhlükə kəsb edir, çünki istifadəçi tərəfindən mətn sahəsinə daxil edilən qiymət yoxlanılmır. Məsələn, əgər həmin qiyməti ədədi qiymətə çevirmək olmazsa və ya həmin qiymət 100000 ədədindən böyükdürsə, onda icra etmə zamanı ilə bağlı səhv əmələ gələcək. Qabaqdakı modullarda həmin səhvlərin qarşısının alınma yolları göstəriləcək.*

6. **UserForm** formanızın **Initialize** hadisəsi üçün bu kodu əlavə edin:

```
sbCem.Value=1  
txtCem.Value=1
```

7. **chMukafat** idarəetmə elementinin **Change** hadisəsi üçün bu kodu əlavə edin:

```
If chMukafat.Value=False Then  
  lblCem.Visible=False  
  txtCem.Visible=False  
  sbCem.Visible=False  
Else  
  lblCem.Visible=True  
  txtCem.Visible=True  
  sbCem.Visible=True  
End If
```

8. **CommandButton1** düyməsinin **Click** hadisəsi üçün bu kod əvəzinə

```
nMukafatCemi=100000
```

aşağıdakı kodu əlavə edin:

```
nMukafatCemi=sbCem.Value
```

Tapşırığın 5-ci punktuna dair (düymə idarəetmə elementinin qurulması):

1. Formada daha bir idarəetmə düyməsini yerləşdirin və onun qiymətini aşağıdakı kimi qurun:

XASSƏ	QIYMƏT
Name	"btnEscape"
Caption	"İmtina"
Cancel	True

2. Düymənin **Click** hadisəsi üçün bu kodu əlavə edin:

```
Upload Me
```

Tapşırığın 6-cı punktuna dair (formanın başlığının (sərlövhəsinin) dəyişdirilməsi):

1. Formanın boş olan yerində mausun sağ düyməsi ilə bir dəfə şıqqıldadın və əmələ gələn kontekst menyusundan **Properties** təlimatını seçin.
2. Əmələ gələn xassələr pəncərəsində **Caption** xassəsinin qiymətini bu cür dəyişin:
"Mükafatlar haqqında informasiya".

15.9 SƏRBƏST İŞLƏMƏK ÜÇÜN TAPŞIRIQ 9:

Proqramların icra müddəti səhvlərinin təyin edilməsi

MƏSƏLƏNİN ŞƏRTİ:

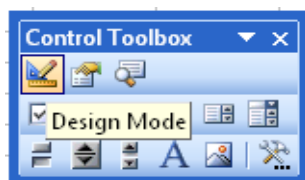
Excel-də VBA ilə yaradılan proqramda icra müddəti səhvini təyin edən prosedura tərtib edin və onu həmin proqramın modulu kimi yoxlayın.

TAPŞIRIQ:

Proqramı elə tərtib edin ki, istifadəçi tərəfindən daxil edilən verilənlər yoxlansın və buraxıla bilməyən qiymətlərin daxil edilməsi avtomatik olaraq mümkün olmasın (məsələn, bölünən və bölənin sətir qiymətləri daxil ediləndə və ya bölənin qiyməti 0 bərabər olduqda). Excel səhifəsində forma obyektlərindən (idarəetmə elementlərindən) istifadə edin.

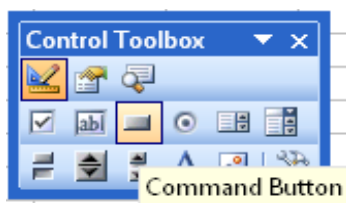
TAPŞIRIĞIN YERİNƏ YETİRİLMƏSİ ÜÇÜN MƏSLƏHƏTLƏR:

1. Yeni Excel faylını yaradın və adını, məsələn, `ErrorHandling.xls` qoyun.
2. Faylın birinci səhifəsinin A1 hücrəsinə bu mətn qiyməti yazın: "Bölmənin nəticəsi".
3. Mausun sağ düyməsi ilə istənilən alətlər panelində və ya menyu üzərində bir dəfə şıqqıldadın və açılan alətlər panelləri siyahısından **Control Toolbox** seçin, şəkl. 15.9.



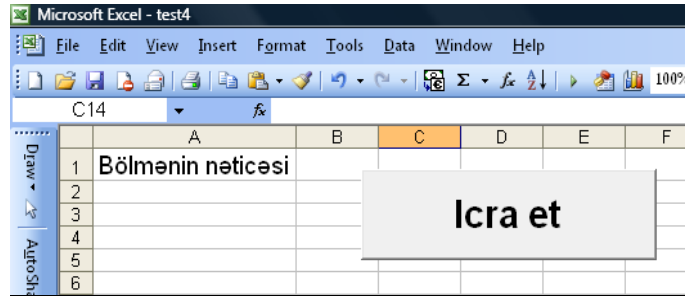
Şəkil 15.9 **Control Toolbox** idarəetmə panelində dizayner rejiminə keçirilməsi.

4. **Control Toolbox**-da Dizayner rejimində (şəkl. 15.10) **Command Button** düymə idarə elementini mausla üstündən basıb sonradan Excel səhifəsində yerləşdirmək lazımdır.



Şəkil 15.10 **Control Toolbox** idarəetmə panelindən Command Button idarəetmə düyməsinin seçilməsi.

5. Excel səhifəsində yerləşdirdiyiniz idarəetmə düyməsi və A1 hücrəsinə daxil etdiyiniz qiymət şək. 15.11-də göstərilən kimi olmalıdır. Düymə üzərində mausun sağ düyməsi ilə bir dəfə şıqqıldadıb kontekst menyusundan **Properties** (xassələr) təlimatını seçin. İdarəetmə elementinin xassəsini istədiyiniz kimi təyin edə bilərsiniz.



Şəkil 15.11 Yaratdığınız proqramın interfeysi

6. Dizayner rejimində mausun sağ düyməsi ilə yaratdığınız idarəetmə düyməsinin üzərində bir dəfə şıqqıldadı və kontekst menyusundan **View Code** təlimatını seçin. Nəticədə VBA kod redaktorunun pəncərəsi açılacaq və kod yazılan pəncərədə həmin düymə üçün **Click** hadisəsinin prosedurasını görəcəksiniz. Həmin proseduraya aşağıdakı kodu əlavə edin:

```
Private Sub CommandButton1_Click()
    Dim nNum1 As Integer
    Dim nNum2 As Integer
    Dim nResult As Variant
    nNum1=InputBox("Birinci ədədi daxil edin")
    nNum2=InputBox("İkinci ədədi daxil edin ")
    nResult=nNum1/nNum2
    Range("B1").Value=nResult
End Sub
```

7. Qaydın həmin Excel səhifəsinə və **Control Toolbox**-da dizayner rejimindən çıxın. Yaratdığınız idarəetmə düyməsini mausla basın. Əmin olun ki, buraxıla bilən verilənləri daxil etdiyiniz halda kod düzgün işləyir və B1 hücrəsinə bölmənin nəticəsini çıxarır.

Həlli: (tapşırığın lazımcına sərbəst yerinə yetirilməsi üçün nəzəri və praktiki cəhətdən kitabın 6.1 - 6.3 bölmələrinin yenidən təkrar edilməsi vacibdir).

Məsələnin əslində çox sayda həlli mövcuddur. Aşağıda bir neçə həll variantları təklif edilir.

1. Çox geniş yayılmış üsullardan biri budur: səhv kodunun emal edilməsi və təkrarən proqramın özü özünün çağırması:

```
Option Explicit
Private Sub CommandButton1_Click()
    Call subPrepare
End Sub
!*****
Public Sub subPrepare()
    Dim nReturnCode As Integer
    Dim nAnswer As Integer
```



```

nReturnCode=fDiv()
Select Case nReturnCode
Case 1
  MsgBox("Sıfıra bölmək olmaz!")
  nAnswer=MsgBox("Təkrarlamaq?", vbYesNo)
If nAnswer=vbYes Then
  Call subPrepare
Else
  Application.Quit
End If
  Case 2
  MsgBox("Ədəd lazımdır!")
  nAnswer=MsgBox("Təkrarlamaq?", vbYesNo)
If nAnswer=vbYes Then
  Call subPrepare
Else
  Application.Quit
End If
  Case 3
  MsgBox ("Naməlum səhv")
  nAnswer=MsgBox("Təkrarlamaq?", vbYesNo)
If nAnswer=vbYes Then
  Call subPrepare
Else
  Application.Quit
End If
End Select
End Sub
!*****
Function fDiv()
On Error Resume Next
  Dim nNum1 As Integer
  Dim nNum2 As Integer
  Dim nResult As Variant
nNum1=InputBox("Birinci ədədi daxil et")
nNum2=InputBox("İkinci ədədi daxil et")
nResult=CInt(nNum1)/CInt(nNum2)
Select Case Err.Number
  Case 0
    Range("B1").Value=nResult
    fDiv=0
  Case 11
    fDiv=1
  Case 13
    fDiv=2
  Case Else
    fDiv=3
End Select
End Function

```

2. Bu üsulda – prosedura səhvin kodunu dövrü hesablama strukturunda icra edir:

```

Private Sub CommandButton1_Click()
  Dim nNum1 As Variant
  Dim nNum2 As Variant
  Dim nResult As Integer
  Dim nError As Integer
Do
  nNum1=InputBox("Birinci ədədi daxil edin:")

```

```

    On Error Resume Next
    nError=CInt(nNum1)
    If Err.Number=13 Then
        MsgBox ("Lazımı ədəd")
        nNum1=""
    End If
    On Error GoTo 0
    Loop While (nNum1="")
    Do
        nNum2=InputBox("İkinci ədədi daxil edin:")
        On Error Resume Next
        nError=CInt(nNum2)
        If Err.Number=13 Then
            MsgBox("Lazımı ədəd")
            nNum2=""
        ElseIf nNum2=0 Then
            MsgBox("Sıfıra bölmək olmaz!")
            nNum2=""
        End If
    On Error GoTo 0
    Loop While (nNum2="")
    nResult=nNum1/nNum2
    Range("B1").Value=nResult
End Sub

```

3. Yalnız bu üsulda ümumiyyətlə, heç bir səhvin yaranmasına yol verilmir:

```

Private Sub CommandButton1_Click()
    Dim nNum1 As Variant
    Dim nNum2 As Variant
    Dim nResult As Integer
    Do
        nNum1=InputBox("Birinci ədədi daxil edin:")
        If IsNumeric(nNum1 & "") Then Exit Do
        MsgBox "Lazımı ədəd"
    Loop
    Do
        nNum2=InputBox ("İkinci ədədi daxil edin:")
        If IsNumeric(nNum2 & "") Then
            If Int(nNum2) <> 0 Then Exit Do
            MsgBox "Sıfıra bölmək olmaz!"
        Else
            MsgBox "Lazımı ədəd"
        End If
    Loop
    nResult=nNum1/nNum2
    Range("B1").Value=nResult
End Sub

```

15.10 SƏRBƏST İŞLƏMƏK ÜÇÜN TAPŞIRIQ 10:

Sorğu ilə verilənlər bazasından yazıların yüklənməsi

MƏSƏLƏNİN ŞƏRTİ:

İstifadəçi Word sənədindən verilənləri əlavə edərək, onların əsasında avtomatik rejimdə verilənlər bazasına sorğu edir və tələb olan məlumatı (verilənlər bazasından çıxarılan verilənləri) proqrama (yəni Word sənədinə) yüklənməlidir.

TAPŞIRIQ:

1. Verilənlər bazası kimi Access-in misallar paketində yerləşən hazır nümunə verilənlər bazasından istifadə etmək lazımdır: həmin Access nümunəsi **Northwind** adı ilə standart halda **C:\Program Files\Microsoft Office\OFFICE11\SAMPLES** ünvanında yerləşir.
2. İstifadəçi Word sənədində verilənləri daxil edərək (məsələn, şirkətdə işləyən əməkdaşlara aid verilənləri), həmin zaman verilənlər bazasına sorğu göndərməlidir.
3. Word-də sorğunun hazırlanmasını avtomatlaşdırmaq üçün xüsusi idarəetmə elementindən (məsələn, düymədən) istifadə edilməlidir.
4. Həmin idarəetmə elementinin (düymənin) hadisələrini elə qurmaq lazımdır ki, sorğu, avtomatik daxil edilən kimi, verilənlər bazasının lazımı cədvəlindən axtarılan qiymətləri tapılsın.
5. Sonradan tapılan (və ya tapılmayan) qiymətlər geriye Word sənədinə avtomatik rejimdə yüklənməlidir (bunun üçün düymənin **click** hadisəsi üçün xüsusi VBA kodu tərtib edilməlidir).
6. Düymənin **click** hadisəsi üçün tərtib olan VBA kodunda dəyişənlərə aşkar təyin olan qiymətləri təyin etmək əvəzinə, onlara **Northwind.mdb** verilənlər bazasındakı **Employees** (əməkdaşlar) cədvəlin qiymətlərini təyin etmək lazımdır, aşağıda göstərilən qayda əsasında:

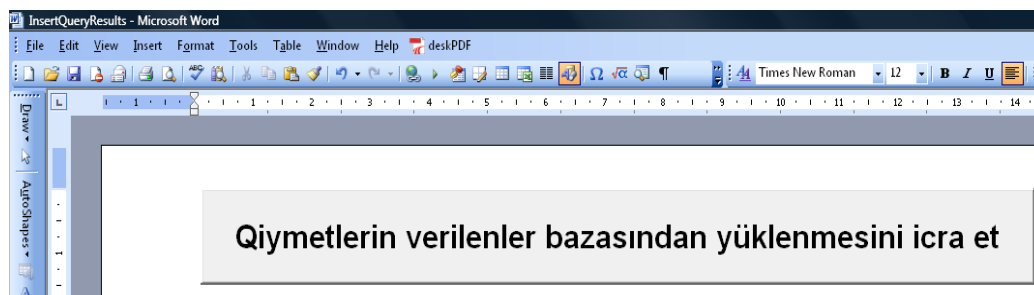
Dəyişənin adı	Access-də Northwind verilənlər bazasının Employees cədvəlindəki yazılar sahəsinin adı (cədvəldəki sütunun adı)
sLastName	Last Name adlı sütunun qiyməti
sFirstName	First Name adlı sütunun qiyməti
sTitle	Title adlı sütunun qiyməti

7. Əməkdaşın nömrəsi (**Employees** cədvəlində **Employee ID** adında olan sütundakı verilənlərin qiymətidir) Word-dən işə salanın VBA kodu ilə **InputBox()** standart funksiyasının köməyi ilə istifadəçi tərəfindən daxil edilməlidir.

TAPŞIRIĞIN YERİNƏ YETİRİLMƏSİ ÜÇÜN MƏSLƏHƏTLƏR:

1. Yeni Word sənədini yaradaraq, məsələn, ona **InsertQueryResults.doc** adını verin.
2. Sənəddə mausun sağ düyməsi ilə istənilən idarəetmə paneli və ya menyu üzərində bir dəfə şıqqıldadıb, açılan kontekst menyusundan **Control Toolbox** (idarəetmə elementlərinin paneli) seçin.
3. **Control Toolbox**-da dizayner rejiminə keçin və oradan Word sənədinin seçdiyiniz bir yere yeni idarəetmə düyməsini qoyun. Düymənin lazım olan parametrlərini sazlayın (düymənin koordinatlarını, üzərində olan yazını, yazının şriftini v.s.).

4. Mausun sağ düyməsi ilə həmin yaradılmış düymənin üzərində bir dəfə şıqqıldadın və açılan kontekst menyusundan **Properties** (xassələr) təlimatını seçin. Xassələri istədiyiniz kimi təyin edin. Word-də həmin idarəetmə düyməsi şəkl. 15.12-də təsvir edilən kimi görsənə bilər.



Şəkil 15.12 Word sənədində idarəetmə düyməsinin fərz edilən maket görüntüsü (düymədə “ə” hərfi əvəzinə “e” hərfinin yazılması yalnız VBA-nın “ə” Azərbaycan hərfini qəbul etməməsi ilə bağlıdır) .

5. Word-ün idarəetmə panelinin **Insert**→**Bookmark** menyusunu ilə bu düymə altında **Bookmark1** adlı qoşmasını yerləşdirin.
6. Dizayner rejimində mausun sağ düyməsi ilə yaradılmış idarəetmə düyməsi üzərində bir dəfə şıqqıldadın və açılan kontekst menyusundan **View Code** təlimatını seçin. Nəticədə həmin düymənin **Click** hadisəsi üçün VBA kod redaktorunun pəncərəsi açılacaq. Pəncərədə artıq **Click** hadisəsi üçün prosedura yazılmış halda görünəcək. Həmin proseduraya aşağıdakı VBA kodu əlavə edilməlidir:

```

Private Sub CommandButton1_Click()
    Dim nEmpId As Integer
    Dim sLastName As String
    Dim sFirstName As String
    Dim sTitle As String
    nEmpId=CInt(InputBox("Əməkdaşın nömrəsini daxil edin:"))
    '***** Dəyişdiriləsi kod *****
    sLastName=" Fuller"
    sFirstName=" Andrew"
    sTitle="Prezident müavini, Ticarət"
    '***** Dəyişdiriləsi kodun sonu. *****
    ThisDocument.Activate
    ThisDocument.Bookmarks("Bookmark1").Select
    Selection.TypeText CStr(nEmpId) & " "
    & sLastName & " " & sFirstName & " "
    & vbTab & sTitle & vbCrLf
End Sub

```

Həlli: (tapşırığın lazımcına sərbəst yerinə yetirilməsi üçün nəzəri və praktiki cəhətdən kitabın 9.1 - 9.6 bölmələrinin yenidən təkrar edilməsi vacibdir).

Word-də yaradılan idarəetmə düyməsinin **Click()** hadisəsi üçün tərtib ediləsi VBA kodu aşağıda göstərilən nümunədəki kimi yazmaq olar:

```

Private Sub CommandButton1_Click()

```

```

Dim nEmpId As Integer
Dim sLastName As String
Dim sFirstName As String
Dim sTitle As String
'*** istifadəçidən əməkdaşın nömrəsini aalırıq.****
nEmpId=CInt(InputBox("Əməkdaşın nömrəsini daxil edin:"))
'** Connection obyektini yaradıırıq və sazlayırıq ***
Dim cn As New ADODB.Connection
'**** Northwind.mdb faylına gedən yol *****
cn.ConnectionString="Provider=Microsoft._
Jet.OLEDB.4.0;" & "Data Source=C:\Program Files_
\Microsoft Office\OFFICE11\SAMPLES\Northwind.mdb"
cn.Open
'**** Recordset obyektini yaradıırıq və sazlayırıq. ***
Dim rs As New ADODB.Recordset
'*** İstənilən tərəfə hərəkətə imkan yaradıırıq. *****
rs.CursorType=adOpenStatic
'***** Recordset-i sorğuya görə açıırıq. *****
rs.Open " SELECT [Employee ID], [First Name],_
[Last Name], [Title] " & " FROM [Employees] _
WHERE [Employee ID]=" & nEmpId, cn
'***** Recordset-in boş olmasını yoxlayırıq. *****
If rs.EOF=True And rs.BOF=True Then
MsgBox "Bu adlı əməkdaş yoxdur"
Exit Sub
End If
'*** Cədvəldə tapılan yazılardakı qiymətləri *****
'*** təyin edirik *****
sLastName=rs.Fields("Last Name")
sFirstName=rs.Fields("First Name")
sTitle=rs.Fields("Title")
ThisDocument.Activate
ThisDocument.Bookmarks("Bookmark1").Select
Selection.TypeText nEmpId & " " & sLastName_
& " " & sFirstName & " " & vbTab & sTitle & vbCrLf
End Sub

```

15.11 SƏRBƏST İŞLƏMƏK ÜÇÜN TAPŞIRIQ 11: VBA ilə Power Point mühitində proqramlaşdırma

15.11.1 Elementlərin slaydlara proqram səviyyəsində əlavə edilməsi

MƏSƏLƏNİN ŞƏRTİ: PowerPoint təqdimatında proqram səviyyəsində hər bir slaydın aşağı sağ kənarında bu yazını əlavə etmək lazımdır: " © Microsoft kompaniyasının Bakıdakı Ofisi", şəh. 15.13.

TAPŞIRIQ: Yuxarıda təsvir edilən məsələni həll etmək üçün PowerPoint mühitində xüsusi VBA prosedurasını yaratmaq lazımdır.

TAPŞIRIĞIN YERİNƏ YETİRİLMƏSİ ÜÇÜN MƏSLƏHƏTLƏR:

Real praktiki işlərdə daha əlverişli həll bundan ibarət ola bilər: həmin makros PowerPoint proqramının xaricində olan proqramında (məsələn, Word sənədində və ya Excel səhifəsində) tərtib edilsin və elə oradan da icra edilməsi üçün işə salınsın. Çünki PowerPoint-dən həmin makros işə salınsa, onda proqram kodu hər bir yeni təqdimata avtomatik kopyalanacaq. Lakin yeni

başlayanlara asan olsun deyə tələb olunan VBA kodunu bir başa PowerPoint-də tərtib edib işə salmaq olar.

İstifadəçilər üçün Microsoft Office proqramlaşdırması

©Microsoft kompaniyasının Bakıdakı Ofisi

Şəkil 15.13 PowerPoint təqdimatının hər bir slaydının sağ kənarına proqram səviyyəsində mətnin əlavə edilməsi (kiçik şriftlə " © Microsoft kompaniyasının Bakıdakı Ofisi" mətni).

Həlli: (tapşırığın lazımcına sərbəst yerinə yetirilməsi üçün nəzəri və praktiki cəhətdən kitabın 10-cu fəslinin yenidən təkrar edilməsi vacibdir).

Tərtib edilən makros aşağıda təqdim olan nümunə kimi görsənə bilər:

```
Public Sub InsertCopyRight()  
    Dim oSlide As Slide  
    Dim oShape As Shape  
    'Təqdimat slaydları üzrə dövr edirik  
    For Each oSlide In Application.ActivePresentation.Slides  
        'Hər slaydda Shape obyektini ilə yazını yaradırıq.  
        'Left , Top , Width və Height parametrlərinin lazım olan  
        'qiymətlərini sınaqlardan və ya makrorekorderdən tapırıq.  
        Set oShape=_  
        oSlide.Shapes.AddTextbox(msoTextOrientation_  
        Horizontal, 500, 510, 210, 40)  
        'TextFrame və TextRange iç-içə qurulmuş  
        'obyektlərlə mətni əlçatan edirik.  
        oShape.TextFrame.TextRange.Text=Chr(169) &_  
        "Microsoft kompaniyasının Bakıdakı Ofisi"  
        'Bəzək vurma işləri ilə məşğul oluruq.  
        oShape.TextFrame.TextRange.Font.Size=12  
        oShape.TextFrame.TextRange.Font.Bold=msoTrue  
    Next  
End Sub
```

15.11.2 VBA PowerPoint Makrosu ilə slaydlardakı sxemlərin avtomatik nömrələnməsi

MƏSƏLƏNİN ŞƏRTİ:

PowerPoint təqdimatında çox sayda sxemlər çəkilib. Əvvəlcədən elementlər (bloklar) nömrələnib və hansısa blokun nömrəsi yaddan çıxdıqda onun axtarılması xeyli çətinləşir. Həmin çətinliyi aradan qaldırmaq üçün işləri proqram səviyyəsində avtomatlaşdırmaq lazımdır.

TAPŞIRIQ:

Yuxarıda təsvir edilən məsələni həll etmək üçün PowerPoint mühitində xüsusi VBA prosedurasını yaratmaq lazımdır.

Həlli: (tapşırıqın lazımınca sərbəst yerinə yetirilməsi üçün nəzəri və praktiki cəhətdən kitabın 10-cu fəslinin yenidən təkrar edilməsi vacibdir).

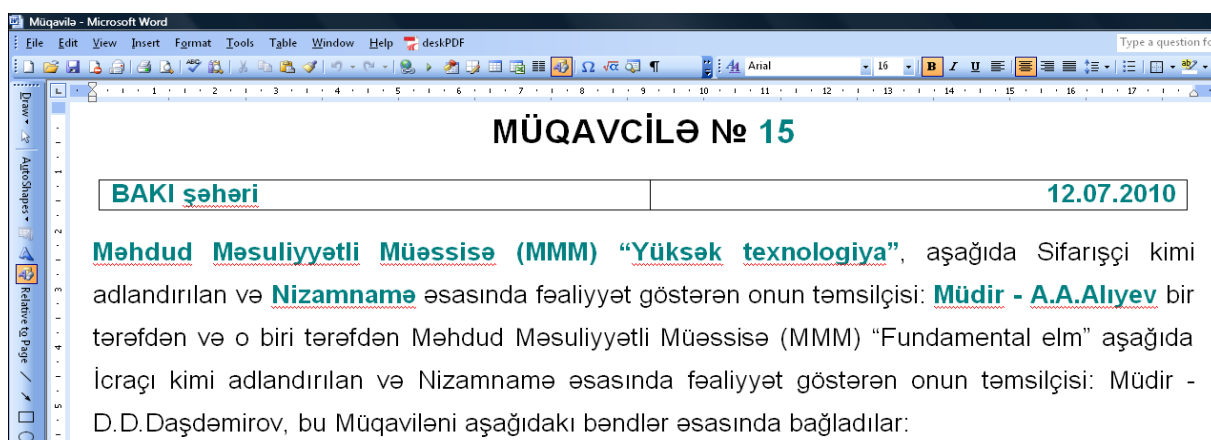
Tərtib edilən makros aşağıdakı nümunəyə bənzəyə bilər:

```
Sub Change ()
  Dim foundText As Object
  Dim oTmpRng As TextRange
  'Dəyişiləsi mətn sahələrinin nömrələri
  For i=66 To 24 Step -1
    For Each sld In Application.ActivePresentation.Slides
      For Each shp In sld.Shapes
        If shp.HasTextFrame Then
          Set oTxtRng=shp.TextFrame.TextRange
          Set oTmpRng=oTxtRng.Replace(FindWhat:=_
            CStr(i), Replacewhat:=CStr(i+2), _
            WholeWords:=False)
        End If
      Next
    Next
  Next i
End Sub
```

15.12 SƏRBƏST İŞLƏMƏK ÜÇÜN TAPŞIRIQ 12: *Proqram səviyyəsində Word sənədinin formalaşdırılması*

MƏSƏLƏNİN ŞƏRTİ:

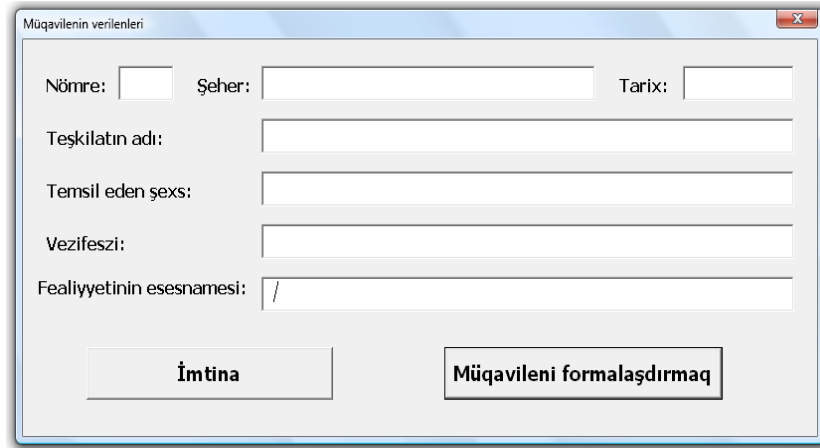
Müəssisədə müqavilənin Word-də formalaşmasını avtomatlaşdırmaq tələb olur. Müqavilənin maketi aşağıdakı şək. 15.14-də göstərilən kimi olmalıdır (məsələni sadələşdirmək üçün yalnız müqavilənin şək. 15.14-də görünən başlanğıc hissəsi formalaşdırılmalıdır). Proqram səviyyəsində dəyişdirilən verilənlər həmin şəkildə mavi rəngli qalın şriftlə seçilib.



Şəkil 15.14 Proqram səviyyəsində yaradılan müqavilənin nümunəsi.

TAPŞIRIQ:

1. **normal.dot** şablonunda Müqavilə adlı istifadəçi formasını yaradın (şək. 15.15-də göstərilən maketə analogi olaraq).



Şəkil 15.15 Müqavilədəki verilənlərin daxil edilməsini təmin edən forma.

2. Formanın avtomatik açılmasını icra edən makros tərtib edin və yaradılmış Word sənədində həmin makrosa Word-ün idarəetmə panelində düymə təyin edin.
3. Kompüterin diskində Word üçün C:\MüqavileTemplate.dot şablonunu (həmin şablona tələb olan verilənlər qoyulacaq) yaradın və lazım olan yerlərə qurmaları əlavə edin.
4. Formadakı **Müqaviləni formalaşdırmaq** adlı idarəetmə düyməsi üçün (şək. 15.15) proqram kodunu tərtib edin. Bu proqram kodu əsasında şablondakı verilənlər əsasında və formadan daxil edilən verilənlər əsasında avtomatik rejimdə tərkibində müqavilə mətni olan yeni sənəd formalaşacaq.

Qeyd: Real tətbiqi proqramlarda formadakı verilənlər bazasında saxlanılmalı idi və sonradan çoxsaylı təkrarlarla onlardan istifadə edilməli idi (məsələn, başqa sənədlərin formalaşdırılmağında). Bu tapşırıqda məsələni çətinləşdirməmək üçün yaradılan sənədin verilənləri bir başa formadan götürülməlidir. Oxşar situasiyada verilənlər bazası ilə işlərin aparılmasına aid nümunələr kitabın Access-ə həsr olan fəslində təsvir edilib. Məhz buna görə bu tapşırıqdakı bütün verilənlər (həmçinin tarix və müqavilənin nömrəsi də) mətn tipinə aid kimi götürülür.

Həlli: (tapşırıqın lazımınca sərbəst yerinə yetirməsi üçün nəzəri və praktiki cəhətdən kitabın bütövlükdə 11-ci fəslin yenidən təkrar edilməsi vacibdir).

1-ci punkta dair – istifadəçi formasının hazırlanması:

1. Yaradılmış Word sənədində VBA redaktorunu açın və **Project Explorer**-də mausun sağ düyməsi ilə **Normal** layihəsi üzərində bir dəfə şıqqıldadın və açılan kontekst menyusundan **Insert**→**UserForm** seçin.

2. Formalar dizaynerində, şəh. 15.15-də göstərilən formaya analoji olan, formanı qurun (kitabın 5-ci fəslində VBA formaları ilə işləmək qaydaları haqqında ətraflı məlumat verilib). Baxılan məsələdə formadakı idarəetmə elementləri bu cür adlandırılacaqdır:
 - `txtCity` - şəhərin adının daxil ediləsi mətn sahəsi;
 - `txtNumber` - müqavilənin nömrəsinin daxil ediləsi mətn sahəsi;
 - `txtDate` - tarixin daxil ediləsi mətn sahəsi;
 - `txtOrg` – təşkilatın adının daxil ediləsi mətn sahəsi;
 - `txtPerson` - təşkilat təmsilçisinin adının daxil ediləsi mətn sahəsi;
 - `txtTitle` - təşkilat təmsilçisinin vəzifəsinin daxil ediləsi mətn sahəsi;
 - `txtLaw` - hüquqi sənəd haqqında məlumatın daxil ediləsi mətn sahəsi;
 - `cmdDog` - müqaviləni formalaşdırmaq üçün icazə düyməsi;
 - `cmdCancel` - ümumiyyətlə, formada işləməkdən imtina edərək, onun bağlanmasını icra edən düymə.
3. Formadakı idarəetmə elementlərinin görüntüsünü istədiyiniz kimi dəyişdirin. Formanın özü üçün onun `Caption` xassəsini “Müqavilənin verilmələri” mətn qiyməti ilə təyin edin. `cmdDog` idarəetmə düyməsi üçün `Default` xassəsinin qiymətini `True` ilə təyin edin. `cmdCancel` idarəetmə düyməsi üçün isə onun `Cancel` xassəsinin qiymətini `True` ilə təyin edin. Formanın öz xassəsi `Name` üçün `FormMuq` qiymətini təyin edin.

2-ci punkta dair – formanı işə salmaq üçün makrosun yaradılması:

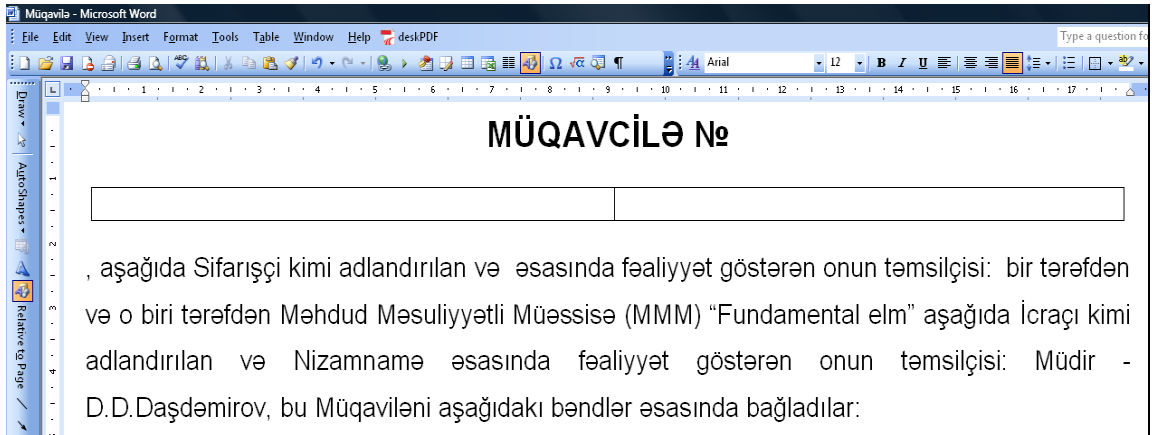
1. **Normal** layihəsinin **NewMacros** modulunda `FormMuqShow()` adlı yeni prosedura yaradın. Proseduranın kodu aşağıdakı nümunədə olan kimi görünə bilər:

```
Public Sub FormMuqShow()  
    FormMuq.Show  
End Sub
```

2. Əmin olun ki, bu prosedura işə salındıqda, yaratdığınız forma açılır.
3. Word-ün **Tools** menyusundan **Customize** seçin və sonra açılan pəncərədə **Commands** qurmasını seçin. **Categories** siyahısından **Macros** seçin və Word-ün istənilən alətlər panelinə **Normal. NewMacros. FormDogShow** adlı makrosun görüntüsünü yerləşdirin. Yaradılmış düymə üçün uyğun olan təsviri sazlayın (kitabın 1-ci fəslinə bax). Bundan sonra **Customize** pəncərəsini və əmin olun ki, düyməyə basdıqda forma açılır.

3-cü punkta dair – Word sənədinin şablonunun yaradılması:

1. şəh. 15.16-da göstərilən yeni Word sənədini yaradın.



Şəkil 15.16 Müqavilənin formalaşdırılan Word şablonu.

2. Həmin sənədin lazım olan yerlərinə nişanları yerləşdirin. şəkl. 15.14-də nişanların yerini görüb təyin etmək olar (mavi rəng və qalın şriftlə yazılan mətn). Bizim məsələdə nişanların adı bu cür adlandırılır:

- bNumber – müqavilənin nömrəsinin çıxarılması üçün nişan;
- bCity – şəhərin adının çıxarılması üçün nişan;
- bDate – tarixin çıxarılması üçün nişan;
- bOrg – təşkilatın adının çıxarılması üçün nişan;
- bPerson – təşkilat təmsilçisinin adının çıxarılması üçün nişan;
- bTitle - təşkilat təmsilçisinin vəzifəsinin çıxarılması üçün nişan;
- bLaw – hüquqi əsasnamə olan sənədin adının çıxarılması üçün nişan.

3. Bu faylı **C:\MuqavileTemplate.dot** adı ilə Microsoft Word şablonu kimi yaddaşda saxlayın.

4-cü punkta dair – formadakı idarəetmə düymələri üçün proqram kodunun yaradılması:

1. **cmdCancel** düyməsinin **Click** hadisəsi üçün aşağıdakı proqram kodunu əlavə edin:

```
Private Sub cmdCancel_Click()
    FormDog.Hide
End Sub
```

2. **cmdDog** idarəetmə düyməsi üçün isə aşağıdakı proqram kodunu əlavə edin:

```
Private Sub cmdDog_Click()
Dim oDoc As Document
Set oDoc=Application.Documents.Add("C:\MuqTemplate.dot")
oDoc.Bookmarks("bNumber").Range.Text=txtNumber.Value
oDoc.Bookmarks("bCity").Range.Text=txtCity.Value
oDoc.Bookmarks("bDate").Range.Text=txtDate.Value
oDoc.Bookmarks("bOrg").Range.Text=txtOrg.Value
oDoc.Bookmarks("bTitle").Range.Text=txtTitle.Value
oDoc.Bookmarks("bPerson").Range.Text=txtPerson.Value
oDoc.Bookmarks("bLaw").Range.Text=txtLaw.Value
FormDog.Hide
oDoc.Activate
End Sub
```

15.13 SƏRBƏST İŞLƏMƏK ÜÇÜN TAPŞIRIQ 13: VBA ilə Excel mühitində proqramlaşdırma

15.13.1 Verilənlər bazasında olan informasiyanın analiz edilməsində Excel-də proqramlaşdırma üsullarının tətbiqi

MƏSƏLƏNİN ŞƏRTİ:

Şirkətin anbarında müəyyən çeşidli və müəyyən miqdarda olan malların qeydiyyatını Access-in standart nümunə verilənlər bazası olan *Northwest* proqramının tərkibində olan *Products* cədvəli əsasında aparırlar. Həmin nümunə proqram MS Office istifadəçi kompyuterinə yükləndikdə C:\Program Files\Microsoft Office\OFFICE11\SAMPLES ünvanında yerləşir. *Products* cədvəlində qeydiyyat üçün vacib olan sütunların adı aşağıdakı cədvəldə verilib. Nəzərə alınmalıdır ki, *Northwest* nümunə proqramı Microsoft tərəfindən İngilis dilində tərtib olmuşdur (MS Access-in Rus versiyasında bu nümunə verilənlər bazası tam Rus dilində tərtib edilib və proqram “Борей” adı ilə çağırılır). Azərbaycan dilində olan nümunə yoxdur – buna görə, məsələnin aydın anlanması üçün, burada adların və identifikatorların lazım olan tərcümələri Azərbaycanca göstərib):

Northwind verilənlər bazasının <i>Products</i> cədvəlindəki ən vacib olan sütunların İngilis dilində adları	Həmin adların Azərbaycan dilində tərcümələri	<i>Products</i> sütunlarında yerləşdirilən verilənlərin hansı mənə daşdığına izahı
Product ID	Malın identifikatoru	Malın kodu (siyahıda olan nömrəsi)
Product Name	Malın adı.	Malın satışa buraxıldıqda ona istehsalçı tərəfindən verilmiş ad, məsələn, “Badamlı” mineral suyu.
Unit Price	Vahidin qiyməti (pul - “\$”)	Bir ədəd malın təyin olmuş satış qiyməti.
Units In Stock	Anbarda olan vahidlər	Yeni bir ədəd mal, hansı ölçü vahidi ilə ölçülürsə ölçülsün – hal-hazırda anbarda hansı saydadır.
Units On Order	Sifarişə gedən vahid	Bu qiymət anbarda olan malın minimal ehtiyatını təmsil edir. Yəni, əgər malın real sayı həmin səviyyədə azdırsa, onda bu maldan, təcili olaraq, (verilmiş səviyyəyə uyğun) anbara gətirmək üçün sifariş etmək lazımdır.
Discontinued	Dayandırılmış (xitam edilmiş)	<i>Products</i> cədvəlində bu qiymətlər bayraqcıq idarəetmə elementi kimi təmsil edilib. Yəni, əgər həmin mal üçün bayraqcıqda “quş” qoyulubsa, onda şirkət həmin malın anbara gətirilməsinə xitam vermişdir (alınıb gətirilməsini dayandırmışdır).

Products cədvəlin digər sütunları qeydlər üçün nəzərə alınmaya da bilər. *Northwest* nümunə verilənlər bazası Access paketində xeyli keçmişdə yaradılmışdır və buna görə *Products*

cədvəlinin doldurulması əski forma ilə icra edilir – daha əlverişli və rahat formalar nəzərdə tutulmayıb (hətta ən yeni Access 2010 versiyasında belə).

TAPŞIRIQ:

Yuxarıda təsvir edilən bəzi çətinlikləri aradan qaldırmaq və qeydiyyat işlərini daha yüksək səviyyədə avtomatlaşdırmaq üçün şirkət istifadəçiyə Excel əsasında elə növ tətbiqi proqramı yaratmağa tapşırılmışdır ki, həmin tətbiqi proqram aşağıda verilmiş əməlləri icra etməyə qadir olsun:

1. Adı çəkilən `Products` cədvəlindəki bütün sətir və sütunlarında olan verilənləri Excel səhifəsinə yükləsin.
2. Excel-ə, aşağıda təsvir edilən məzmunlu əlavə sütunları generasiya etsin:

`Order Goods Units` - dənə ilə malların sifarişi. Bu sütunda olan qiymətlər `Units On Order` (yəni sifarişə gedən vahid və ya anbarda olan malın minimal ehtiyatı) sütunu və `Units In Stock` (yəni Anbarda olan vahidlər və ya hal-hazırda anbarda nə qədər sayda mal var) sütununun fərqidir. Bu sütuna təcili halda sifariş ediləsi malların sayı haqqında informasiya yerləşdirilir. Həmin informasiyanı yalnız o yazılar üçün generasiya etmək lazımdır ki, onlar üçün `Units On Order` sütununda olan qiymət `Units In Stock` sütununda olan qiymətdən böyükdür və `Discontinued` (yəni alınıb anbara gətirilməsinə xitam vermişdir mal) sütununda olan qiymət `False` (yəni – Yalan) qiyməti ilə təyin edilib.

`Order Cost` - sifarişin dəyəri. Bu sütunda olan qiymətlər cədvəlin hər sətiri üçün anbardakı artımın pul dəyərini (yəni sifarişin dəyərini) təyin edir. Sifarişin dəyəri yuxarıdakı `Order Goods In Pieces` sütunun `Unit Price` sütununa hasili kimi hesablanır. Analoji olaraq, bu sütunda olan informasiya yalnız o yazılar üçün generasiya edilməlidir ki, onlar üçün `Units On Order` sütununda olan qiymət `Units In Stock` sütununda olan qiymətdən böyükdür və `Discontinued` (yəni alınıb anbara gətirilməsinə xitam vermişdir mal) sütununda olan qiymət `False` (yəni – Yalan) qiyməti ilə təyin edilib.

Qeyd: Real məsələlərdə daha düzgün (və daha məhsuldar) olardı ki, həmin əlavə generasiya edilən sütunların hesablanması verilənlər bazası serverinə keçirilsin (hesablanan sütunları olan SQL-sorğudan istifadə edərək). Bu tədris təmayüllü tapşırığı yerinə yetirmək üçün həmin sütunların yüklənməsini Excel vasitəsi ilə icra edin (nəzərə alın ki, əgər biz qeyri relyasion verilənlər mənbəyinə müraciət ediriksə (məsələn, mətn fayllarına), onda bu cür həll - mümkün olan yeganə həldir).

3. Verilənlər bazasından alınan yazılar altında bir sətirdə iki yekun sətirləri yükləyə bilsin:
 - "Total cost of goods in stock" – tərcümədə "anbarda olan bütün malların yekun dəyəri". Qiymətin hesablanması bu cür icra

edilir: hər sətir üçün Units In Stock sütununun Unite Price sütununa hasilləri cəmlənir;

- “Total value of the goods to the order” - tərcümədə “Sifarişdə olan malların yekun dəyəri”. Order Cost sütununun yekun qiymətidir (yəni sütunda olan qiymətlərin cəmidir).

Tətbiqi proqramın ümumi görüntüsü şəx. 15.17-də göstərilən kimi tərtib edilə bilər:

Product ID	Product Name	Unit Price	Units In Stock	Reorder Level	Discontinued	Order Goods Units	Order Cost	Units On Order
1	Chai	18.00	39	10	FALSE			0
2	Chang	19.00	17	25	FALSE			40
3	Aniseed Syrup	10.00	13	25	FALSE	8	\$152.00	70
4	Chef Anton's Cajun Seasoning	22.00	53	0	FALSE	12	\$120.00	0
5	Chef Anton's Gumbo Mix	21.35	0	0	TRUE			0

Şəkil 15.17 Excel səhifəsinə import edilmiş verilənlər (həmin import edilmiş sütunların başlığı sarı rəngə boyanıb və qalın şriftlə yazılıb).

Tapşırıqda tələb edilən yekun sətirlər isə şəx. 15.18-də göstərilən kimi görünə bilər:

73	69	Gudbrandsdalsost	36.00	26	15	FALSE		
74	70	Outback Lager	15.00	15	30	FALSE	15	\$225.00
75	71	Fløtemysost	21.50	26	0	FALSE		
76	72	Mozzarella di Giovanni	34.80	14	0	FALSE		
77	73	Röd Kaviar	15.00	101	5	FALSE		
78	74	Longlife Tofu	10.00	4	5	FALSE	1	\$10.00
79	75	Rhönbräu Klosterbier	7.75	125	25	FALSE		
80	76	Lakkalikööri	18.00	57	20	FALSE		
81	77	Original Frankfurter grüne Soße	13.00	32	15	FALSE		
84		Total cost of goods in stock:		\$74 050.85				
85		Total value of the goods to the order:		\$3 633.45				

Şəkil 15.18 Cədvəlin ən aşağıda duran sətirləri və hesablanmış yekun qiymətlər (qalın şriftlə seçilib).

Həlli: (tapşırıqın lazımcına sərbəst yerinə yetirməsi üçün nəzəri və praktiki cəhətdən kitabın 12-ci fəslinin yenidən təkrar edilməsi vacibdir).

1. Yeni Excel faylını yaradın, **Control Toolbox** idarəetmə panelini aktivləşdirin və onun köməyi ilə şəx. 15.17-də göstərilən **Command Button** idarəetmə düyməsini yaradın.
2. Həmin idarəetmə elementinin xassəsində onun adını və yazının formatını sazlayın – düymənin adını “Verilenleri yükle” kimi təyin edin.
3. Düymənin üzərində mausun sağ düyməsi ilə bir dəfə şıqqıldadıq və açılan kontekst menyusundan **View Cod** təlimatını seçin. Dərhal VBA-nın proqram kodunun redaktoru açılacaq və proqram kodunun düyməyə aid **click** hadisəsi üçün olan yerdə cursor aktiv vəziyyətdə duracaq.
4. Kod redaktorunun **Tools** menyusundan **Reference** təlimatını seçin və bayraqcığı **Microsoft ActiveX Data Objects 2.1 Library** sətirinin üstündə təyin edin.

5. İndi isə kod redaktorunda idarəetmə düyməsinin **Click** hadisəsi üçün aşağıdakı VBA kodunu əlavə edin:

```
Private Sub CommandButton1_Click()  
'Başlanğıcda bütün Excel kitabını köhnə verilənlərdən  
'təmizləyirik  
    Cells.Select  
    Selection.Clear  
'Connection obyektini yaradıırıq və sazlayırıq.  
Dim cn As New ADODB.Connection  
cn.ConnectionString=_  
"Provider=Microsoft.Jet.OLEDB.4.0;_  
Data Source=C:\Program Files\Microsoft_  
Office\OFFICE11\SAMPLES\Northwind.mdb"  
cn.Open  
'Recordset obyektini yaradıırıq və sazlayırıq.  
    Dim rs As New ADODB.Recordset  
    rs.Open"SELECT[Product ID],[Product Name],_  
    [Unite Price],[Units In Stock],_  
    [Units On Order]," & "[Discontinued] FROM Products", cn  
'Recordset obyektini əsasında sorğu cədvəlini QueryTable obyektini  
'yaradıırıq və onu 4-cü sətirdən başlayaraq Excel səhifəsinə  
'yükleyirik.  
    Dim QT1 As QueryTable  
    Set QT1=QueryTables.Add(rs,Range("A4"))  
    QT1.Refresh  
'QueryTable obyektində yazıların sayını təyin edirik.  
Dim nRowCount As Integer  
Dim oRange As Range  
Set oRange=QT1.ResultRange  
nRowCount=oRange.Rows.Count  
'Yeni "Order Goods Units" sütununu yaradıırıq.  
Range("G4").Value="Order Goods Units"  
Range("G4").Font.Bold=True  
Range("G4").Columns.AutoFit  
'Yeni "Order Cost" sütununu yaradıırıq.  
Range("H 4").Value="Order Cost"  
Range("H4").Font.Bold=True  
Range("H4").Columns.AutoFit  
'QueryTable obyektini üzərində G sütununu tərkibinə qatan  
'diapazon yaradıırıq.  
    Set oRange=Range("G5","G" & nRowCount+3)  
'Dövrde bizə lazım olan dəyişənləri hazırlayırıq.  
Dim oCell As Range  
Dim sRowNumber As String  
Dim cMoney As Currency  
Dim cItogMoney As Currency  
Dim cItogSklad As Currency  
'Yaradılmış diapazonun bütün hücrələri  
'üzərindən dövr edərək keçirik.  
For Each oCell In oRange.Cells  
'Sətrin mütləq nömrəsini sətir dəyişəni tipində alırıq.  
    sRowNumber=Replace(oCell.Address(True), "$G$", "")  
'Təyin etdiyimiz şərtləri yoxlayırıq.  
    If Range("E" & sRowNumber).Value > Range_  
("D" & sRowNumber) And Range("F" & sRowNumber).Value=False Then  
'G sütünü üçün qiyməti alırıq (sayla ölçülən sifariş).  
    oCell.Value=(CInt(Range("E" & sRowNumber)._  
Value)- CInt(Range("D" & sRowNumber).Value))
```



```

'H sütünü üçün qiyməti alırıq (sifarişin dəyəri).
  cMoney=(CInt(Range("E" & sRowNumber).Value)
  - CInt(Range("D" & sRowNumber).Value))*CCur(Range("C" &
sRowNumber).Value)
'Qiyməti H sütununa yazırıq.
  Range("H" & sRowNumber).Value=cMoney
'Dərhal bu qiyməti yekun hesablamaya cəmləyirik.
  cItogMoney=cItogMoney+cMoney
  End If

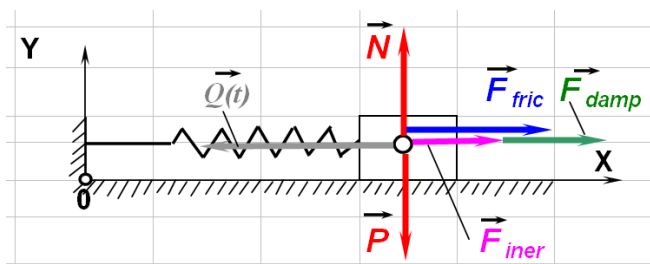
'Həmin dövrdə anbarda olan malların dəyərini cəmləyirik.
cItogSklad=cItogSklad+(Range("C" & sRowNumber).Value*Range("D" &
sRowNumber).Value)
Next
'Yekün hesablamalara aid tələb olan iki sətiri formalaşdırırıq.
Range("B" & nRowCount+6).Value="Total cost of goods in stock:"
Range("B" & nRowCount+6).Font.Bold=True
Range("B" & nRowCount+7).Value=
"Total value of the goods to the order:"
Range("B" & nRowCount+7).Font.Bold=True
Range("D" & nRowCount+6).Value=cItogSklad
Range("D" & nRowCount+6).Font.Bold=True
Range("D" & nRowCount+7).Value=cItogMoney
Range("D" & nRowCount+7).Font.Bold=True
'Yekün hesablamaya aid yazıların şriftini qalın təyin edirik ...
Range("D" & nRowCount+7).Select
'... və skrollamanı icra edirik.
Range("D" & nRowCount+7).Show
End Sub

```

15.13.2 Excel VBA proqramlaşdırma mühitində Newmark ədədi metodu əsasında adi diferensial tənliklərlə ifadə edilən dinamik sistemin modelləşdirilməsi

MƏSƏLƏNİN ŞƏRTİ:

Elmi-araşdırma laboratoriyasında dəqiq mexanikaya aid dinamik sistemin qeyri xətti adi diferensial tənliklərlə ifadə edilən modelinin ədədi həllinin təyin edilməsi tələb edilir (məsələnin ədədi üsul ilə həlli daha üstün yol kimi seçilib). Laboratoriyada olan kompyuterlərdə bu cür məsələləri həll etmək üçün xüsusi təyinatlı kompyuter riyaziyyatı kateqoriyasına aid proqram təminatları yoxdur – bir qədər uyğun gələn yalnız MS Office paketinin tərkibində olan Excel proqramıdır. Aparıcı elmi işçilər qərara gəlir ki, hal-hazırkı vəziyyətə görə daha effektiv yol – Nyumark (*Newmark numerical method*) ədədi metodunun alqoritmindən istifadə etməkdir. Bunun üçün isə VBA Excel mühitində proqram tərtib etmək lazımdır. İstifadəçi qarşısında duran tapşırıqlar aşağıda təsvir edilib. Qısaca məsələnin şərtini bu cür təsvir etmək olar (şək. 15.19 və şək. 15.20).



Şəkil 15.19 Dəqiq mexanikaya aid dinamik sistemin təsviri.

$\Sigma F_{n(x)} = 0$	\longrightarrow	$F_{iner} + F_{damp} + F_{fric} - Q(t) = 0 ;$
		$mx'' + cx' + k(x + ax^3) - Q(t) = 0 ;$
və ya	\longrightarrow	$x'' + 2nx' + p^2(x + ax^3) = (1/m)Q(t) ;$

Şəkil 15.20 Dəqiq mexanikaya aid dinamik sistemin qeyri xətti adi diferensial tənliklərlə ifadə edilməsi.

Burada: F_{iner} – inersiya (ətalət) qüvvəsidir; F_{damp} – dempfer (söndürücü) qüvvəsidir; F_{fric} – sürtünmə (*friction*) qüvvəsidir; $Q(t)$ – xaricdən təsir edən qüvvədir;

Aşağıdakı parametrlər hesablama proqramında istifadə edilə bilər:

$2n = c/m$; $p^2 = k/m$; t_1 - zaman; n_1 – zaman intervalı; m , c və k - uyğun olaraq, kütlə, dempfer əmsalı və yayın elastiklik əmsalıdır; a - qeyri xətliliyi nəzərə alan əmsaldır; $x(t)$ – yerdəyişmədir, $y(t) = x(t)'$ – sürətdir; e_1 – buraxılabilən xəta ədədidir; x_0 və y_0 – zamanın $t = 0$ anında başlanğıc qiymətləridir.

TAPŞIRIQ:

1. Nyumark ədədi metodunun alqoritmini öyrənib mənimsəyin
2. Ədədi metodunun alqoritmini əsasında hesablama proqramının strukturunu tərtib edin.
3. VBA Excel mühitində həmin proqramı reallaşdırmaq və yoxlamalardan keçirib (hesablama proqramının dəqiqlik və dayanıqlıq meyarlarına görə), araşdırılan dinamik sistemin ədədi modeləşdirilməsini yerinə yetirin.

TAPŞIRIĞIN YERİNƏ YETİRİLMƏSİ ÜÇÜN MƏSLƏHƏTLƏR:

1. Tələb olunan Nyumark ədədi metodun alqoritmini riyazi tərəfdən daha yaxşı anlamaq üçün xüsusi ədəbiyyatlardan istifadə etmək lazımdır. Bu kitabın mövzusunə aid olmadığına görə burada metodun alqoritmi haqqında məlumat verilməsinə ehtiyac yoxdur. Oxucu həmin metodun alqoritmini buradan öyrənə bilər: **Newmark N.M. A method of computation for structural dynamics.** Journal Eng. Mech. Div. Proc. Amer. Soc. Civil Engrs. 1959, v. 85, N. 5, pp. 67 – 94. və http://en.wikipedia.org/wiki/Newmark-beta_method.
2. Peşəkar proqramçılar edən kimi: ədədi metodu yaxşı mənimsədikdən sonra proqram kodunu ən birinci psevdokodda (yeni fərz edilən və ya beyində canlandırılan proqram təminatında) tərtib edin.
3. Yeni Excel faylı yaradın və adını, məsələn, **Newmark1.xls** qoyun.
4. Excel kitabının birinci səhifəsinin adını *Resonance* qoyun və məsələnin ədədi verilənlərini təyin edilmiş hücrələrə daxil etmək üçün nəzərdə tutun. Məsələnin şərtinə uyğun olaraq, həmin səhifədə şəkl. 15.21-də göstərilən qiymətləri daxil etmək üçün cədvəl qurun (verilənlərin fiziki ölçüləri göstərilməyib).

	A	B	C
22			
23			
24	Verilənləri bura daxil edin		
25	Resonance effect		
26	m	10	
27	c	0.025	
28	k	3.6	
29	alf	0	
30	x0	0	
31	y0	0	
32	t1	100	
33	n1	200	
34	e1	0.000001	
35	amp	10	
36	w	0.6	
37			

Şəkil 15.21 Excel kitabının birinci səhifəsində məsələnin ədədi verilənlərini təyin edilmiş hücrələrə daxil etmək üçün nəzərdə tutulan cədvəl.

- Excel VBA proqram kodu redaktorunu açın və VBA **Project (NonLinVib)** layihəsində yeni modul yaradın.
- Daxil edilmiş (və ya dəyişdirilmiş) qiymətlərin VBA proqramından oxunmasını təmin etmək üçün proqramın baş hissəsində (verilənlərin, sabitlərin və dəyişənlərin tipini təyin edən sətirlərdən sonra) aşağıdakı proqram kodunu daxil edin:

```

Set m=Worksheets("Resonance").Range("b26")
Set c=Worksheets("Resonance").Range("b27")
Set k=Worksheets("Resonance").Range("b28")
Set alf=Worksheets("Resonance").Range("b29")
Set x0=Worksheets("Resonance").Range("b30")
Set y0=Worksheets("Resonance").Range("b31")
Set t1=Worksheets("Resonance").Range("b32")
Set n1=Worksheets("Resonance").Range("b33")
Set e1=Worksheets("Resonance").Range("b34")
Set amp=Worksheets("Resonance").Range("b35")
Set w=Worksheets("Resonance").Range("b36")

```

- Yuxarıdakı 2-ci punktun məsləhəti ilə kod redaktorunda psevdokodda tərtib etdiyiniz hesablama prosedurasını indi VBA-da yazın (yəni həmin Nyumark metodunun əsasında tərtib edilən əsas hesablama prosedurasını indi VBA proqram kodunda yazın).
- Hesablanmış nəticələrin qiymətlərini Excel kitabının digər səhifəsinə yüklənməsini təmin edən kodu proqrama əlavə edin. Həmin fraqment, məsələn, aşağıdakı VBA proqram koduna bənzəyə bilər:

```

For i=1 To n1
Set curcell=Worksheets("L1").Cells(i, 1)
curcell.Value=i
Set curcell=Worksheets("L1").Cells(i, 2)

```

```

curcell.Value=j(i)
Set curcell=Worksheets("L1").Cells(i, 3)
curcell.Value=tt(i)
Set curcell=Worksheets("L1").Cells(i, 4)
curcell.Value=xx(i)
Set curcell=Worksheets("L1").Cells(i, 5)
curcell.Value=yy(i)
Next i

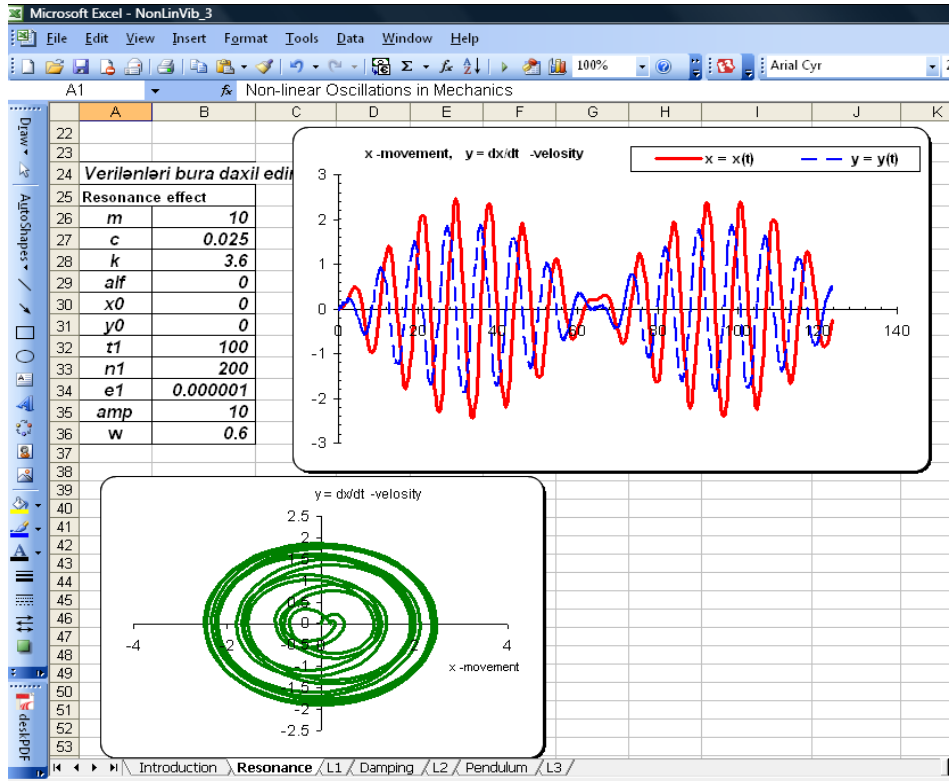
```

Yuxarıdakı kod proqrama əlavə edilsə, onda L1 adlı Excel səhifəsinə hesablanmış qiymətlər çap ediləcək (L1 səhifəsinin görüntüsü şəkl. 15.22-də olan kimi görsənə bilər): i - ədədi metodun zaman intervalında iterasiyanın sayıdır; $j(i)$ - iterativ prosedurada i sayılı zaman iterasiyasında ədədi həllin dəqiqliyinin təmin edilməsi üçün dəqiqlik iterasiyasının sayıdır; $tt(i)$ - zamanın i sayında hesablanmış qiymətidir; $x(t)$ - axtarılan funksiyanın (məsələnin fiziki məzmununa görə dinamik sistemdə yerdəyişmə mənasını daşıyır) qiymətidir; $yy(i)$ - axtarılan $xx(i)$ funksiyanının birinci törəməsidir (dinamik sistemdə sürət mənasını daşıyır).

	A	B	C	D	E	F	G
1	1	8	0	0	0	$i -$	
2	2	7	0.625	0.007785	0.024913	$j(i) -$	
3	3	6	1.25	0.043409	0.089083	$t(i) -$	
4	4	6	1.875	0.122557	0.164189	$x(i) -$	
5	5	6	2.5	0.240529	0.213324	$y(i) -$	
6	6	5	3.125	0.370717	0.203275		
7	7	6	3.75	0.470626	0.116433		
8	8	6	4.375	0.494139	-0.04119		
9	9	6	5	0.406701	-0.23861		
10	10	7	5.625	0.198939	-0.42622		
11	11	6	6.25	-0.10556	-0.54818		
12	12	5	6.875	-0.45106	-0.55742		
13	13	6	7.5	-0.75955	-0.42974		
14	14	6	8.125	-0.94825	-0.17412		
15	15	6	8.75	-0.95132	0.164312		
16	16	6	9.375	-0.7399	0.512219		
17	17	6	10	-0.33429	0.785747		
18	18	6	10.625	0.195965	0.911054		
19	19	5	11.25	0.743574	0.841295		

Şəkil 15.22 Excel kitabının təyin edilmiş başqa səhifəsinə hesablanmış nəticələrin qiymətlərinin çıxarılması.

- Çap edilmiş nəticələrin əsasında həllin qrafiklərini avtomatlaşdırılmış rejimdə qurulmasını təmin edən Chart obyektlərini qurun (proqram səviyyəsində və ya Excel-in qrafik interfeysində). Alınan hesablanmış nəticələrin qrafikləri şəkl. 15.23-də göstərilən kimi görünə bilər.



Şəkil 15.23 Excel-in Chart obyektı ilə hesablanmış nəticələrin qrafiklərinin qurulması.

10. Məsələnin ilkin ədədi qiymətlərini, əlbəttə, sistemin fiziki mahiyyətinə uyğun olaraq, dəyişdirin və bir neçə variantda yenidən proqramı işə salın. Məsələnin fiziki mahiyyətinə uyğun olan digər variantların ilkin ədədi qiymətlərini aşağıda veririk. Oxucu bu variantların ədədi qiymətləri ilə də məsələni yenidən həll edə bilər (hər halda məsələnin fiziki mahiyyətinin anlanması olduqca vacib amildir). Həmin variantların ədədi qiymətləri bunlardır (nəzərinizə çatdırırıq ki, verilənlərin fiziki ölçüləri burada göstərilməyib):

- **Dinamik döyüntü:** $m = 10$; $c = 0.012$; $n = c/m$; $k = 6.4$; $p^2 = k/m$; $a = 0.01$.; $A = 2$; $\omega = 0.7$; $q_1 = A \cdot (1/m) \cdot \sin(\omega \cdot t)$; $x_0 = 0$; $y_0 = 0$;
- **Resonans effekti:** $m = 10$; $c = 0.025$; $n = c/m$; $k = 3.4$; $p^2 = k/m$; $a = 0$; $A = 12$; $\omega = 0.6$; $q_1 = A \cdot (1/m) \cdot \sin(\omega \cdot t)$;
- **Sönən rəqsi hərəkət (dempferin aşağı səviyyəli müqaviməti ilə):**
 $m = 10$; $c = 1$; $n = c/m$; $k = 6.4$; $p^2 = k/m$; $a = 0$.; $A = 0$; $\omega = 0.6$; $q_1 = A \cdot (1/m) \cdot \sin(\omega \cdot t)$; $x_0 = 20$; $y_0 = 100$;
- **Sönən rəqsi hərəkət (dempferin yuxarı səviyyəli müqaviməti ilə):**
 $m = 10$; $c = 15$; $n = c/m$; $k = 6.4$; $p^2 = k/m$; $a = 0$.; $A = 0$; $\omega = 0.6$; $Q(t) = A \cdot (1/m) \cdot \sin(\omega \cdot t)$; $x_0 = 20$; $y_0 = 25$;

11. Alınan ədədi qiymətləri və qrafikləri araşdırın və analiz edin. Proqramın düzgün və etibarlı işləməsinə tam əmin olun.

Həlli: (tapşırığın lazımınca sərbəst yerinə yetirməsi üçün nəzəri və praktiki cəhətdən kitabın 12-ci fəslinin yenidən təkrar edilməsi vacibdir).

VBA-da tertib edilən proqram kodu aşağıda göstərilən nümunəyə bənzəyə bilər. Əgər bu proqramı oxucu eyniliklə VBA redaktorunda yığsa, onda düzgün nəticələr alınması mümkündür. Burada bir incəlik vardır - proqramın ilkin ədədi qiymətləri Excel-in birinci səhifəsindən oxuduğu hücrələrin indekslərini düzgün vermək lazımdır. Həmçinin hesablanmış nəticələrin Excel kitabının başqa səhifəsinə çap edilməsinə lazımınca diqqət vermək vacibdir.

```

Sub Newmark1()
'Proqramdakı sabitlər və dəyişənlərin tipinin təyin edilməsi.
Dim i As Integer
Dim j(1000), t(1000), x(1000), y(1000),_
z(1000), tt(1000), xx(1000), yy(1000) As Variant
Dim m, c, k, s, x0, y0, t1, n1, e1, amp, w As Range
'Excel kitabının "Resonance" adlı səhifəsindəki məsələnin ədədi
'verilənlərinin B26, ..., B36 hücrələrindən oxunmasını təmin edirik.
Set m=Worksheets("Resonance").Range("b26")
Set c=Worksheets("Resonance").Range("b27")
Set k=Worksheets("Resonance").Range("b28")
Set alf=Worksheets("Resonance").Range("b29")
Set x0=Worksheets("Resonance").Range("b30")
Set y0=Worksheets("Resonance").Range("b31")
Set t1=Worksheets("Resonance").Range("b32")
Set n1=Worksheets("Resonance").Range("b33")
Set e1=Worksheets("Resonance").Range("b34")
Set amp=Worksheets("Resonance").Range("b35")
Set w=Worksheets("Resonance").Range("b36")
'Newmark metodunun əsas (bədən) hissəsi başlayır.
'Başlanğıc şərtlər. *****
x(0)=x0
y(0)=y0
t(0)=0
'*****
d=t1/n1
q1=amp*Sin(w*t(0))
z0=(q1-c*y0-k*x0*(1+alf*x0*x0))/m
For i=1 To n1
t(i)=i*d
If i=1 Then GoTo lb1
If i=2 Then GoTo lb2
a=y(i-1)+z(i-1)*d/2
b=x(i-1)+y(i-1)*d/2
Y1=y(i-2)+2*z(i-1)*d
GoTo lb3
lb1:
a=y0+z0*d/2
b=x0+y0*d/2
Y1=y0+z0*d
GoTo lb3
lb2:
a=y(1)+z(1)*d/2
b=x(1)+y(1)*d/2
Y1=y(0)+2*z(1)*d
lb3:
j(i)=1
GoTo lb5
lb4:
j(i)=j(i)+1
x(i)=X1
Y1=a+z1*d/2
lb5:
X1=b+Y1*d/2
q1=amp*Sin(w*t(i))
z1=(q1-c*Y1-k*X1*(1+alf*X1*X1))/m
If j(i)=10 Then GoTo lb6

```

```

        If Abs(X1-x(i))>=e1*Abs(X1) Then GoTo lb4
lb6:      x(i)=X1
          y(i)=Y1
          z(i)=z1
lb7:      tt(i)=t(i-1)
          xx(i)=x(i-1)
          yy(i)=y(i-1)
'Zaman intervalında alınan həllin qiymətlərini
'İmmediate Windows pəncərəsinə çıxardırıq.
        Debug.Print i, j(i), tt(i), xx(i), yy(i)
        Next i
' ***** Burada Nyumark prosedurası qurtarır. *****
'Təyin edilmiş "L1" səhifəsinə hesablanmış
'nəticələrin qiymətlərinin çıxardırıq.
        For i=1 To n1
            Set curcell=Worksheets("L1").Cells(i, 1)
            curcell.Value=i
            Set curcell=Worksheets("L1").Cells(i, 2)
            curcell.Value=j(i)
            Set curcell=Worksheets("L1").Cells(i, 3)
            curcell.Value=tt(i)
            Set curcell=Worksheets("L1").Cells(i, 4)
            curcell.Value=xx(i)
            Set curcell=Worksheets("L1").Cells(i, 5)
            curcell.Value=yy(i)
        Next i
End Sub

```

15.14 SƏRBƏST İŞLƏMƏK ÜÇÜN TAPŞIRIQ 14: Access-də VBA proqramının yaradılması

MƏSƏLƏNİN ŞƏRTİ:

Müəssisənin müdiriyyəti Word-də yaradılmış müqavilələrin avtomatlaşdırılmasını icra edən tətbiqi proqramı (bax 15.12-ci bölmədəki tapşırığa – “Proqram səviyyəsində Word sənədinin formalaşdırılması”) yenidən dəyişdirmək üçün qərar verdi. Çünki həmin proqramda aşağıdakı çatışmamazlıqlar meydana çıxdı:

- İstifadəçi tərəfdən formaya daxil edilən verilənləri proqram yaddaşda saxlamır – buna görə həmin verilənlərdən təkrarən istifadə etmək olmur. Daha əlverişli olardı ki, verilənlər hansısa verilənlər bazasında saxlanılsın, məsələn, Access-də;
- proqramın işləməsi üçün iki komponent lazımdır: proqram kodu (həmçinin VBA forması) fiziki mühit baxımından **Normal.dot** faylında yerləşir və ikinci komponent - şablon **MuqavileTemplate.dot** isə diskin kök kataloqunda yerləşir. Proqramın bu cür struktur qurumu onun bir kompyuterdən başqasına keçirilməsinə əngəllər törədir.

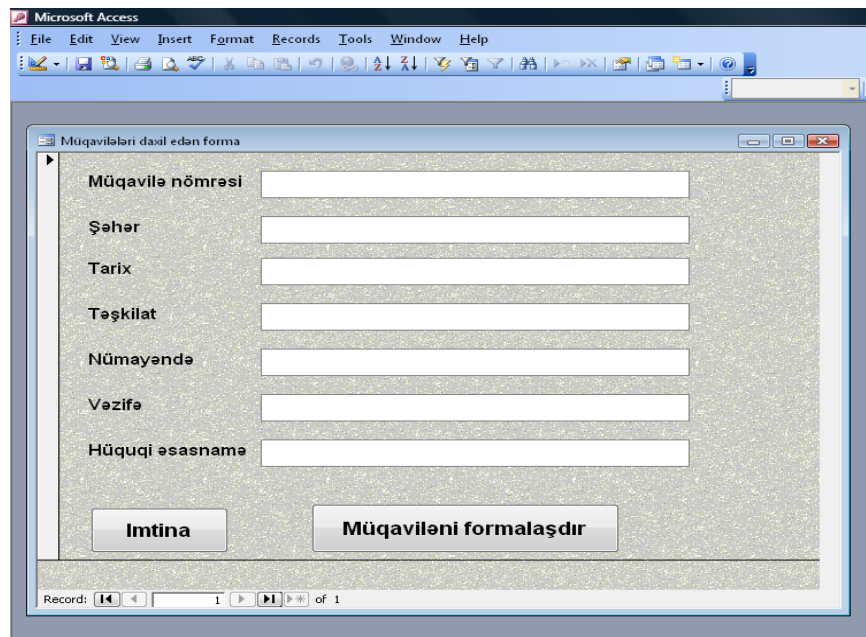
TAPŞIRIQ:

Yuxarıda adı çəkilən Word-də avtomatlaşdırılmış müqavilə yaradan tətbiqi proqramı aşağıdakı tələblərə uyğun dəyişdirmək lazımdır:

- istifadəçi müqavilələr ilə bağlı olan verilənləri daxil edəndə, tətbiqi proqram həmin verilənləri Microsoft Access verilənlər bazasında saxlanmasını təmin etməlidir;
- Word sənədinin şablonu (onun əsasında müqavilə formalaşır) həmin Access verilənlər bazasında yerləşməlidir;
- İstifadəçi tərəfindən edilən hər bir səhv hərəkətlərdən minimal səviyyədə mühafizə yaradılmalıdır: tətbiqi proqram işə salındıqda istifadəçi verilənlərin doldurulması üçün nəzərdə tutulmuş formadan başqa heç bir başqa obyektə görməməlidir.

Bundan ötrü aşağıdakı əməlləri yerinə yetirmək lazımdır:

1. Access-də yeni verilənlər bazasını yaradın və onun daxilində lazım olan cədvəlləri və Word-də tərtib edilən bütün müqavilələr və şablon sənədləri qurun.
2. Həmin verilənlər bazasında şəkl. 15.24-də göstərilən formaya oxşayan forma yaradın. Bu forma ilə istifadəçi müqavilələr haqqında olan informasiyanı verilənlər bazasının cədvəlinə yükləməli və həmçinin Word formatında müqaviləni avtomatik rejimdə formalaşdırmalıdır. Həmin forma üçün VBA proqram kodunu tərtib edin.
3. İstifadəçi tərəfindən edilən hər bir səhv hərəkətlərdən minimal səviyyədə mühafizə yaradılması üçün tətbiqi proqram işə salındıqda istifadəçinin gördüyü forma obyektinə şəkl. 15.24-də göstərilənə oxşamalıdır.



Şəkil 15.24 Aktivləşdirilmiş proqramın pəncərəsi və əmələ gələn müqavilənin verilənlərini daxil etməsini təmin edən forma.

TAPŞIRIĞIN YERİNƏ YETİRİLMƏSİ ÜÇÜN MƏSLƏHƏTLƏR (tapşırığın lazımınca sərbəst yerinə yetirməsi üçün nəzəri və praktiki cəhətdən kitabın bütövlükdə 13-cü fəslin yenidən təkrar edilməsi vacibdir):

Tapşırığın 1-ci punktuna dair – verilənlər bazasının və lazım olan cədvəllərin yaradılması:

1. Microsoft Access-i işə salın və **File** menyusundan **New** əmrini seçin. Yaranan pəncərədə **Blank database** seçin və faylın adını, məsələn, **Muqavileler.mdb** qoyun.

2. Həmin yeni yaradılmış verilənlər bazasını acın və **Tables** qurmasına aid **Create table in Design view** təlimatı üzərində mausun sağ düyməsi ilə iki dəfə şıqqıldadın. Yaradılmış cədvəldə üç sayda şəkl. 15.25-də göstərilən sütunu tərtib edin. Bu cədvəli şablon kimi (yeni ingilis terminologiyasında **Template** kimi) yaddaşa saxlayın və dizayner pəncərəsini bağlayın.

Field Name	Data Type	
ŞablonNömrəsi	Number	
Təsvir	Memo	
Şablon	OLE Object	

Şəkil 15.25 **Template** tipli cədvəlin strukturu.

3. Verilənlər bazasının pəncərəsində **Tables** qurmasına aid **Create table in Design view** təlimatı üzərində mausun sağ düyməsi ilə iki dəfə şıqqıldadın. Yeni cədvəldə olan sütunların strukturu aşağıdakı cədvəl 15.1-də göstərilən struktura bənzəməlidir. Tərtib edilmiş cədvəli yaddaşa **Muqavileler** adı ilə saxlayın və Dizayner pəncərəsini bağlayın.

Cədvəl 15.1

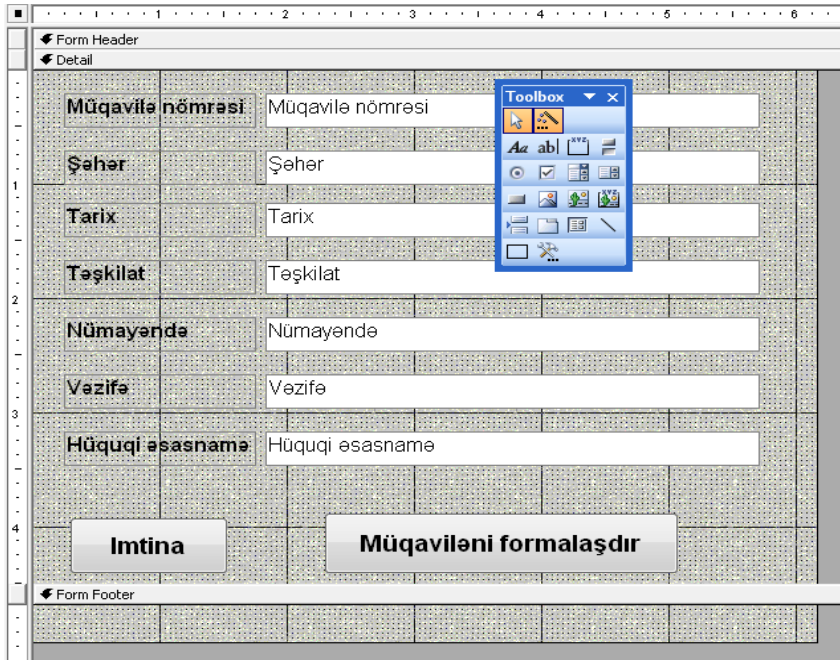
Sütunun adı	Verilənlərin tipi	Ölçüsü
MüqaviləninNömrəsi	Mətn (ilkin açar)	10
Şəhər	Mətn	30
Tarix	Tarix/zaman	50
Təşkilat	Mətn	50
Nümayəndə	Mətn	50
Vəzifəsi	Mətn	50
HüquqiƏsasname	Mətn	100

4. Verilənlər bazası pəncərəsində **Tables** qurmasına aid **Temlates** cədvəli üzərində mausun sağ düyməsi ilə iki dəfə şıqqıldadın. Cədvəl verilənləri daxil etmə rejimində açılacaq. Cədvəlin **Təsvir** adlı sütununun birinci sətirinə "Müqavilə şablonu" yazın. Sonra **Şablon** sütununda hücrəni seçin və Access-in **Insert** menyusundan **Object** təlimatını seçin. Açılan pəncərədə keçiricini **Create from File** seçin. Açılan fayl brauzerindən, 15.12-ci bölmədəki sərbəst işləmək üçün yerinə yetirdiyiniz tapşırıqda yaratdığınız MüqavileTemlates.dot faylını yerləşdirdiyiniz ünvandan tapın və onu seçin. Sonra pəncərədə **OK** düyməsini basaraq faylı həmin verilənlər bazasının tərkibinə yerləşdirin.

5. Yerləşdirdiyiniz faylın doğrudan da verilənlər bazasının tərkibində olmasına əmin olmaq üçün cədvəldəki hücrədə yerləşdirilən şablon üzərində mausun sağ düyməsi ilə iki dəfə şıqqıldadın. Əmin olun ki, **Şablonun nömrəsi** sütununda birinci sətir üçün 1 ədədi avtomatik generasiya olub və sonra həmin cədvəli bağlayın.

Tapşırığın 2-ci punktuna dair – Access forması və müqavilə faylını avtomatik formalaşdıran VBA proqram kodunun yaradılması:

1. Verilənlər bazası pəncərəsində **Forms** qurmasının **Create table by wizard** təlimatını seçin. Dərhal **Table Wizard** (yəni tərcümədə “Usta köməyi ilə cədvəl tərtibatı”) pəncərəsi açılacaq.
2. Həmin pəncərənin birinci ekranından **Contacts** seçin. Sonra pəncərənin ikinci ekranından bütün sahələri seçin və pəncərədəki İngilis dilli terminləri Azərbaycanca dəyişdirin (bunun üçün pəncərədəki **Rename Field** düyməsini basın. Bütün sahələri lazımınca Azərbaycan dilində adlandırdıqdan sonra pəncərənin **Next** düyməsini basın.
3. Pəncərədə növbəti yaranan ekranda formanın bir sütunlu variantını seçin və **Next** düyməsinə basın.
4. Yaranan növbəti ekranda xoşunuza gələn stili seçin və **Next** düyməsini basın.
5. Son yaranan ekranda formanın adını **Müqavilələri daxil edən forma** yazın və **Modify the table design** təlimatını seçin və pəncərədəki **Finish** düyməsini basın. Bununla yeni yaradılmış forma üçün dizayner rejimi açılacaq.
6. Formada yaradılmış elementlərin yerləşdirilməsini və istədiyiniz kimi dəyişdirilməsini dizayner vasitəsi ilə öz zövqünüzə uyğun tərtib edin.
7. **Toolbox** ilə formanın boş olan yerində **Option Group** idarəetmə elementini əlavə edin. Avtomatik generasiya edilən yazını həmin elementdən silin və o elementin xassələrini açın. **Name** xassəsində `OLEObject1` qiymətini qurun və **Visible** xassəsini **Not** qiyməti ilə qurun.
8. Formaya iki düymə əlavə edin və adlarını **İmtina** və **Müqaviləni formalaşdır** qoyun. Birinci düymənin **Name** xassəsində `cmdCancel` ikincisinin isə **Name** xassəsində `cmdDog` qurulmalıdır. Açılan **Wizard forms** pəncərəsində **Cancel** düyməsini basın.
9. Əmin olun ki, formada idarə elementlərinin mətn sahələrində standart vəziyyətə görə MüqaviləNömrəsi, Şəhər, Tarix v.s. adları qalmışdır. Nəticədə dizayner pəncərəsindəki forma şəkl. 15.26-da göstərilən kimi olmalıdır.



Şəkil 15.26 Dizayner pəncərəsində görünən forma.

10. **İmtina** düyməsi üzərində mausun sağ düyməsi ilə iki dəfə şıqqıldadın və kontekst menyusundan **Build Event** təlimatını seçin. Açılan **Creator** pəncərədə **Programs** seçin və **OK** düyməsini basın. Bu düymənin **click** hadisəsi üçün aşağıdakı VBA kodunu əlavə edin:

```
Private Sub cmdCancel_Click()
    Form.Undo
End Sub
```

11. VBA-nın redaktor pəncərəsinin **Tools**→**References** menyusunu ilə **Microsoft Word 11.0 Object Library** obyekt kitabxanasına istinadı əlavə edin.
12. Analoji olaraq, **Müqaviləni formalaşdır** düyməsinin **click** hadisəsi üçün də aşağıdakı VBA kodunu əlavə edin:

```
Private Sub cmdDog_Click()
    Dim Tarix As Date
    Dim MüqaviləNömrəsi, Şəhər, Təşkilat, Nümayəndə, Vəzifə, _
        HüquqiƏsasnamə As String
    'Dəyişənlərə formanın idarəetmə elementlərinin
    'köməyi ilə qiymətləri təyin edirik.
    If Form.Controls("Tarix").Value <> "" Then
        Tarix=Form.Controls("Tarix").Value
    If Form.Controls("MüqaviləNömrəsi").Value <> "" Then
        MüqaviləNömrəsi= Form.Controls("MüqaviləNömrəsi").Value
    If Form.Controls("Şəhər").Value <> "" Then
        Şəhər=Form.Controls("Şəhər").Value
    If Form.Controls("Təşkilat").Value <> "" Then
        Təşkilat=Form.Controls("Təşkilat").Value
    If Form.Controls("Nümayəndə").Value <> "" Then
        Nümayəndə=Form.Controls("Nümayəndə").Value
    If Form.Controls("Vəzifə").Value <> "" Then _
```

```

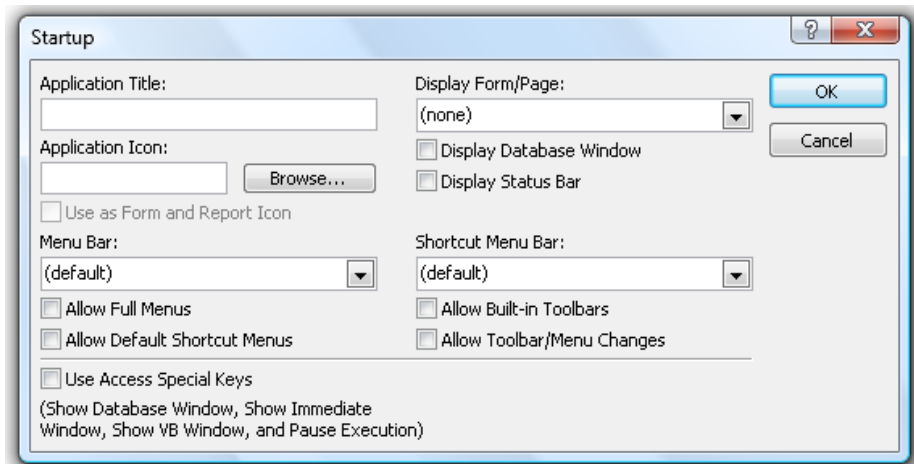
Vəzifə=Form.Controls("Vəzifə").Value
If Form.Controls("HüququƏsasnamə").Value <> "" Then
    HüququƏsasnamə=Form.Controls("HüququƏsasnamə").Value
    'Şablon alırıq - indi verilənlər bazasından.
Dim oBOF As BoundObjectFrame
    Set oBOF=Form.Controls("OLEObject1")
    oBOF=DLookup("[Şablon]", "Şablonlar", "_
    [ŞablonNömrəsi]=1")
    oBOF.Verb=acOLEVerbOpen
    oBOF.Action=acOLEActivate
    'İşə saldıığımız Word faylına və onda açılan sənədə istinadları
    'alırıq.
Dim oWord As Word.Application
    Set oWord=GetObject(, "Word.Application")
Dim oDoc As Word.Document
    Set oDoc=oWord.ActiveDocument
    oWord.Visible=True
    oWord.ActiveWindow.WindowState=wdWindowStateMaximize
    oDoc.Activate
    'Verilənləri nişanlayırıq.
    oDoc.Bookmarks("bNumber").Range.Text=MüqaviləNömrəsi
    oDoc.Bookmarks("bCity").Range.Text=Şəhər
    oDoc.Bookmarks("bDate").Range.Text=Tarix
    oDoc.Bookmarks("bOrg").Range.Text=Təşklət
    oDoc.Bookmarks("bTitle").Range.Text=Vəzifəsi
    oDoc.Bookmarks("bPerson").Range.Text=Nümayəndə
    oDoc.Bookmarks("bLaw").Range.Text=HüquqiƏsasnamə
End Sub

```

13. Yaradığınız program kodunu işə salın və onun düzgün işləməsinə əmin olun.

Tapşırığın 3-cü punktuna dair – istifadəçi əməllərinə qarşı minimal mühafizənin təmin edilməsi:

1. Access verilənlər bazasının pəncərəsindəki **Tools** menyusundan **Startup** təlimatını seçin.
2. Açılmış **Startup** pəncərəsində bütün bayraqcılıqları çıxardın və **Display From/Page** siyahısında “Müqavilələri daxil edən forma” seçin (şək. 15. 27).



Şəkil 15.27 **Startup** pəncərəsində tətbiqi programın parametrlərinin qurulması.

3. **Startup** pəncərəsində **OK** düyməsini basın. Verilənlər bazasını bağlayın və yenidən açın. Əmin olun ki, müqavilələri daxil edən formadan başqa bütün digər obyektlər həqiqətən istifadəçinin nəzərindən gizlənilib.

15.15 SƏRBƏST İŞLƏMƏK ÜÇÜN TAPŞIRIQ 15: *VBA mühitində Outlook kontaktları ilə işlərin aparılması*

MƏSƏLƏNİN ŞƏRTİ:

Müəssisədə belə bir vəziyyət yaranıb: müştərilərlə (təşkilatlarla) işləmək üçün, onların nümayəndələri haqqında olan informasiya (nümayəndələrin adı, tutduqları vəzifə, əlaqə saxlamaq üçün müxtəlif çeşidli ünvanlar v.s.) müəssisənin elektron mühitində xüsusi cədvəldə saxlanılır. Müştərilərlə bir başa əlaqə saxlayan mütəxəssislər xahiş edirlər ki, həmin informasiya onlar üçün MS Outlook-dan əlçatan olsun.

TAPŞIRIQ:

Yaranmış məsələni operativ həll etmək üçün müəssisə rəhbərliyi öz İCT mütəxəssisinə **ImportContacts()** adlı Outlook VBA makrosunu yaratmağı tapşırır. Həmin makros aşağıdakı əməlləri yerinə yetirməlidir:

1. Avtomatik rejimdə Outlook-da **Müştərilərlə əlaqələr** adlı **Contacts** tipli qovluq yaratmalıdır.
2. Həmin qovluqda istifadəçi kompyuterinin C:\Program Files\Microsoft Office\OFFICE11\SAMPLES\Northwind.mdb yolunda yerləşən **Northwind.mdb** verilənlər bazasının **Müştərilər** cədvəlində (əvvəlcədən həmin verilənlər bazasında **Customers** cədvəlinin adı dəyişdirib **Müştərilər** təyin edilir) olan bütün yazılar üçün kontaktlar yaratmalıdır.

Qeyd: Məsələni asanlaşdırmaq üçün yaradılacaq kontaktlarda təşkilatın adı, təşkilat nümayəndəsinin adı, onun telefonu və E-poçt ünvanı haqqında olan informasiyanı qoymaq kifayətdir. Praktiki işlərdə, çox güman ki, daha əlverişli yol bundan ibarət ola bilər: **Exchange Server** serverində ümumi qovluq yaradaraq, proqram səviyyəsində onda kontaktları yaratmaq. Bununla həmin informasiya dərhal müəssisənin bütöv istifadəçi qrupu üçün əlçatan ola bilər. Lakin baxdığımız sadə məsələdə Outlook-un lokal qovluğunda kontaktlar elementləri yaradılmalıdır. Hər bir hal üçün tərtib edilən VBA proqram kodu eyni olacaq.

HƏLL VƏ TAPŞIRIĞIN YERİNƏ YETİRİLMƏSİ ÜÇÜN MƏSLƏHƏTLƏR (tapşırığın lazımınca sərbəst yerinə yetirməsi üçün nəzəri və praktiki cəhətdən kitabın bütövlükdə 14-cü fəslin yenidən təkrar edilməsi vacibdir):

1. Outlook-u işə salın və onun aktiv pəncərəsində klaviaturanın **<Alt>+<F11>** düymələr kombinasiyasını basın. Nəticədə VBA proqram kodu redaktorunun pəncərəsi açılacaq.

2. VBA kod redaktorunun **Tools**→**References** menyusunda **Microsoft ActiveX Data Objects 2.1 Library** obyekt kitabxanasına istinadı əlavə edin.
3. VBA kod redaktorunun **Project Explorer** pəncərəsində mausun sağ düyməsi ilə **Project1** üzərində iki dəfə şıqqıldadın və açılan kontekst menyusundan **Insert** →**Module** təlimatını seçin. Bununla yeni **Module1** standart proqram modulu yaranacaq.
4. Həmin proqram modulunda **ImportContacts()** adlı yeni proseduraya yaradaraq, aşağıdakı VBA kodunu əlavə edin:

```

Public Sub ImportContacts()
  Dim oFolder As MAPIFolder
  Dim oNameSpace As NameSpace
  Dim oContact As ContactItem
  Set oNameSpace=Application.GetNamespace("MAPI")
  'Qovluğa yaradırıq və onun obyektinə istinad alırıq.
  '*****
  Set oFolder=oNameSpace.Folders ("Şəxsi qovluqlar")._
  Folders.Add("Müştəri kontaktları",olFolderContacts)
  '**** Birləşmə obyektini yaradırıq ****.
  Dim cn As New ADODB.Connection
  cn.ConnectionString="Provider=Microsoft.Jet.OLEDB.4.0;"_
  & "Data Source=C:\Program Files\Microsoft_
  Office\OFFICE11\SAMPLES\Nortwind.mdb"
  cn.Open
  '**** Recordset obyektini yaradırıq.****
  Dim rs As New ADODB.Recordset
  rs.Open "Müştərilər", cn
  '**** Recordset üzərində dövr edirik.****
  Do While rs.EOF=False
  '**** Kontakt obyektini yaradırıq.****
  Set oContact=Application.CreateItem(olContactItem)
  'Recordset-in verilənləri ilə onun xassələrini təyin edirik.
  '*****
  oContact.CompanyName=rs.Fields("Adı")
  oContact.FullName=rs.Fields("Müraciət edin")
  oContact.BusinessTelephoneNumber=rs.Fields("Telefon")
  oContact.BusinessEmailAddress=rs.Fields("E-Poçt")
  'Bizim qovluğa köçürürük və yaddaşda saxlanmasını təmin edirik.
  '*****
  oContact.Move oFolder
  oContact.Save
  '**** Yaddaşı təmizləyirik. ****
  Set oContact=Nothing
  '**** Recordset-də bir yazı irəli yeri dəyişirik. ****
  rs.MoveNext
  Loop
End Sub

```


ƏDƏBİYYAT

1. Paul Lomax VB & VBA in a Nutshell: The Language. Published by O'Reilly & Associates, Inc., printed in the USA, 1998.
2. Ə.Aslanov. Müasir proqramlaşdırma texnologiyaları. GDU-nun professor-müəllim heyətinin elmi konfransı. Bakı, 2001.
3. Steve Cummings. VBA for Dummies. 3-rd edition. Published by Hungry Minds. Inc, printed in the USA, 2001.
4. Steven Roman. Writing Excel Macros with VBA. 2-nd edition, O'Reilly Association, Inc., 2002.
5. С.Т.Гусейнов. Численное исследование модели Ланчестера с учетом внутренней конкуренции в одной из противоборствующих сторон. Azərbaycan Milli Elmlər Akademiyası, Gəncə Regional şöbəsi, Xəbərlər Məcmuəsi № 10, Fizika-Riyaziyyat bölməsi, 2003.
6. Л.А. Демидова, А.Н.Пылькин. Программирование в среде Visual Basic for Applications: Практикум. - М.: Горячая линия - Телеком, 2004.
7. Paul McFedries. Absolute Beginner's Guide to VBA. Published by Que Publishing, Printed in the USA, 2004.
8. Richard Mansfield. Mastering VBA for Microsoft Office 2007. Published by Wiley Publishing, Inc. Printed in the USA, Indianapolis, Indiana, 2007.
9. Gail Perry. Excel 2007 Macros. Published by McGraw-Hill Companies, printed in the USA 2009.
10. Л.Д.Слепцова. Программирование на VBA в MS Office 2010. И.Д.Вильямс, Москва, 2010.
11. Petroustos Evangelos. Mastering Microsoft Visual Basic 2010. Published by Wiley Publishing, Inc., Indianapolis, Indiana 2010.
12. John Walkenbach. Excel 2010 Power Programming with VBA. Published by Wiley Publishing, Inc. Printed in the USA, Indianapolis, Indiana, 2010.
13. Офисное программирование в среде MS VBA Excel. www.5ballov.ru.
14. http://en.wikipedia.org/wiki/Visual_Basic_for_Applications
15. Б.Калошин. Быстрый старт в работе с базами данных на VBA. <http://www.vbaworld.ru>.
16. Rod Stephens. Microsoft Office Programming: A Guide for Experienced Developers. <http://www.apress.com>.
17. R.J. Ribando. An Excel/Visual Basic for Applications (VBA) Programming Primer. Revised 6/18/2010. <http://www.faculty.virginia.edu/ribando/modules>.
18. А.Колесов, О.Павлова. Советы тем, кто программирует на Visual Basic и MS Office/VBA. <http://www.twirpx.com/library/comp/office/vba/>

MÜNDƏRİCAT

ÖN SÖZ	3
GİRİŞ (Kitab haqqında ümumi məlumat. Kitab kimin üçün nəzərdə tutulub. Kitabın strukturu)	5
1. MS OFFICE PROQRAMLAŞDIRMASININ ƏSASLARI	8
1.1 Visual Basic salnaməsi (dilinin yaradılmasının zərurəti və inkişafı)	8
1.2 MS Visual Basic for Applications (VBA) Office proqramlaşdırma mühitinin tarixi və inkişaf yolu	9
1.3 Müasir proqramlaşdırma texnologiyalarının əsas xassələri	10
1.4 VBA ilə Office-də proqramlaşdırmaq nə üçün lazımdır?	11
1.5 VBA dili nə deməkdir?	12
1.6 Macrorecorder-in tətbiqi: bir sətir belə kod yazmadan proqramın yaradılması haqqında	14
1.7 Makrosun dialoq pəncərəsi, alətlər paneli, əmrlər sətri: makros yaradıldıqdan sonra işə salınma qaydaları	17
2. VBA REDAKTORU İLƏ TANIŞLIQ	27
2.1 VBA redaktorunda işləmədən öncə təhlükəsizlik parametrlərinin seçilməsi. VBA-nın qrafik interfeysi ilə ümumi tanışlıq	27
2.2 Project Explorer – VBA layihəsinin pəncərəsi və VBA layihəsinin strukturu	34
2.3 Code Editor Window - VBA kodunun redaktoru ilə işləmə	37
2.3.1 Pəncərənin açılması və onun strukturu	37
2.3.2 Obyektlərin və hadisələrin siyahısı	37
2.3.3 Proqram kodunun pəncərəsinin bölünməsi və quraşdırmaları	38
2.3.4 VBA redaktorunda proqramının yazılması	38
2.4 VBA redaktorun sorğu materialları ilə işləmə qaydaları	40
3. MS VBA SINTAKSISİNİN ƏSASLARI	44
3.1 VBA sintaksisinin əsas strukturu və əsas elementləri	44
3.2 VBA-da istifadəçi tərəfindən yaradılan proqramların və funksiyaların strukturu	45
3.3 VBA sintaksisinin əsas qaydaları	47
3.4 VBA operatorları: hesabi, məntiqi, müqayisə və mənimsəmə	49
3.5 VBA dəyişənləri və verilənlərin tipi	52
3.6 VBA-da sabitlər	58
3.7 VBA-da çoxluqlar (arrays)	59
3.8 VBA-da hesablama proseduralarının idarəetmə operatorları	63
3.8.1 DO . . . LOOP dövrü hesablamalar operatorlar strukturundan istifadə etmə qaydası	63
3.8.2 For . . . Next dövrü hesablamalar operatorlar strukturundan istifadə etmə qaydası	67

3.8.3	VBA-da şərti idarəetmə operatorlar strukturları. If...Then...Else şərti idarəetmə operatorlar strukturundan istifadə etmə qaydası	71
3.8.4	SELECT CASE şərti idarəetmə operatorlar strukturundan istifadə etmə qaydası	75
3.8.5	VBA-da GoTo idarəetmə operatoru, onun tətbiq etmə halları	77
3.9	VBA-da proseduralar və funksiyalar	78
3.9.1	Proseduraların növü (Sub , Function , Property və standart riyazi funksiyalar)	80
3.9.2	Proseduraların görünmə sahəsi	89
3.9.3	Proseduraların elan edilməsinin avtomatlaşdırılması	90
3.9.4	Proseduralarda parametrlərin ötürülməsi	90
3.9.5	Proseduraların işə salınması və tamamlanması	93
3.9.6	VBA proqramlaşdırmasında daha önəmli olan və tez-tez istifadə edilən standart funksiyalar	94
3.9.7	Çevirmənin və tiplərin yoxlanmasını təmin edən funksiyalar	95
3.9.8	Sətir funksiyaları	96
3.9.9	Tarix və zamanla işləməyi təmin edən funksiyalar	98
3.9.10	Verilənləri formatlayan funksiyalar	99
3.9.11	İstifadəçi ilə qarşılıqlı əlaqələr yaradan funksiyalar	100
3.9.12	Sintaktik konstruksiyaları əvəz edən funksiyalar	101
3.9.13	Çoxluqlarla (matrislərlə) işləmək üçün funksiyalar	102
3.9.14	Fayl sistemi ilə işləmək üçün olan funksiyalar	102
3.9.15	VBA-nın digər faydalı funksiyaları	104
4.	OBJEKT MODELLƏRİ VƏ OBJEKTLERLƏ İŞLƏMƏ QAYDALARI	105
4.1	Obyektlərin yaradılması və yox edilməsi	108
4.2	Obyektin metodları	110
4.3	Obyektin xassələri	111
4.4	Obyektin hadisəsi və WithEvents elanı	113
4.5	Obyektlərə baxma və nəzərdən keçirmə	114
4.6	Obyekt modelləri	115
5.	FORMALAR, İDARƏETMƏ ELEMENTLƏRİ VƏ HADISƏLƏR	117
5.1	Formaların yaradılması və onların ən vacib olan xassələri və metodları	118
5.2	İdarəetmə elementləri	122
5.2.1	İdarə etmə elementləri haqqında əsas məlumat	122
5.2.2	Label (yazı) idarəetmə elementi	122
5.2.3	TextBox (mətn qutusu) idarəetmə elementi	123
5.2.4	ComboBox (kombinəli siyahı) idarəetmə elementi	125
5.2.5	ListBox (siyahı) idarəetmə elementi	126

5.2.6	CheckBox (yoxlama qutusu) və ToggleButton (fiksə edən düymə) idarəetmə elementləri	127
5.2.7	OptionButton (keçirici) və Frame (çərçivə) idarəetmə elementləri	129
5.2.8	CommandButton (düymə) idarəetmə elementi	130
5.2.9	ScrollBar (diyirlənmə zolağı) və SpinButton (heasblayıcı) idarəetmə elementləri	132
5.2.10	TabStrip (əlavə qurmalar yığımı) və MultiPage (səhifələr yığımı) idarəetmə elementləri	133
5.2.11	Image idarəetmə elementi	135
5.2.12	Əlavə idarə elementlərinin tətbiqi. Microsoft Web Browser , Calendar , RefEdit idarəetmə elementləri	136
6.	PROQRAMDAKI SƏHVLƏRİN DÜZƏLDİLMƏSİ VƏ EMALI	138
6.1	Səhvlərin tipi	138
6.2	Proqramın sazlama üsulları. Immediate , Locals və Watch pəncərələri	139
6.2.1	Proqramların testlənməsi	139
6.2.2	Fasilə rejiminə keçmə	141
6.2.3	Fasilə rejimindəki əməllər	142
6.2.4	Immediate pəncərəsi	144
6.2.5	Locals pəncərəsi	145
6.2.6	Watches pəncərəsi	146
6.3	Proqramın icra etmə zamanı səhvlərinin təyin edilməsi və aradan qaldırılması	148
7.	YARDIMÇI İLƏ İŞLƏMƏ METODLARI	150
8.	ALƏTLƏR PANELLƏRİ VƏ MENYU İLƏ İŞLƏMƏ QAYDALARI	153
9.	VERİLƏNLƏR BAZASI İLƏ İŞLƏMƏ QAYDALARI VƏ ADO OBYEKT MODELİNİN TƏTBİQİ	160
9.1	Verilənlər bazası ilə işləmək nə üçün vacibdir	160
9.2	Verilənlər bazası və ADO	162
9.3	Connection obyektı və Errors kolleksiyası	164
9.4	Excel səhifəsindəki cədvələ qoşulma	170
9.5	Recordset obyektı və Fields kolleksiyası	177
9.5.1	Recordset obyektinin işə salınması	177
9.5.2	Kursorun sazlanması və Recordset -in digər açılma parametrləri	179
9.5.3	Recordset -də hərəkət etmə imkanları	182
9.5.4	Fields kolleksiyası və Field obyektləri	184
9.5.5	Verilənlərin sortlaşması və filtrasiyası	186
9.5.6	Recordset obyektı ilə mənbədəki yazıların dəyişdirilməsi	187
9.5.7	Recordset obyektinin digər xassələri və metodları	189
9.6	Command obyektı və Parameters kolleksiyası	192

10. MS PowerPoint PROQRAMINDA VBA PROQRAMLAŞDIRMASI	195
11. MS Word PROQRAMINDA VBA PROQRAMLAŞDIRMASI	200
11.1 Word-də proqramlaşdırma hansı hallarda lazım olur	200
11.2 Word-də proqramlaşdırmağa giriş, Word-dün obyekt modelinə (Word Object Model) ümumi baxış	202
11.3 Application obyektini	203
11.3.1 Application obyektini ilə işləmə qaydaları	203
11.3.2 Application obyektinin xassələri, metodları və hadisələri	205
11.4 Documents kolleksiyası və Document obyektləri	212
11.4.1 Documents kolleksiyası ilə işləmə qaydaları	212
11.4.2 Documents kolleksiyasının xassələri və metodları	214
11.4.3 Document obyektini ilə işləmə qaydaları, obyektin xassələri və metodları	216
11.5 Selection, Range və Bookmark obyektləri	222
11.5.1 Selection obyektini ilə işləmə qaydaları	222
11.5.2 Selection obyektinin xassələri və metodları	223
11.5.3 Range obyektini ilə işləmə, obyektin xassələri və metodları	227
11.5.4 Bookmark obyektini	229
11.6 Word-ün digər obyektləri	231
11.6.1 AddIns kolleksiyası və AddIn obyektini	231
11.6.2 AutoCorrect obyektini	231
11.6.3 Languages kolleksiyası və Language obyektini	232
11.6.4 Options obyektini	232
11.6.5 Find və Replacement obyektləri	232
11.6.6 Font və ParagraphFormat obyektləri	233
11.6.7 PageSetup obyektini	235
11.6.8 Table, Column, Row və Cell obyektləri	236
11.6.9 System obyektini	237
11.6.10 Tasks kolleksiyası və Task obyektini	237
11.6.11 Windows kolleksiyası və Window obyektini	239
12. MS Excel-də PROQRAMLAŞDIRMA	239
12.1 Excel -də proqramlaşdırma hansı hallarda lazım olur	239
12.2 Application obyektini	242
12.3 Application obyektinin xassələri və metodları	243
12.4 Workbooks kolleksiyası və Workbook obyektini, onların xassələri və metodları	251
12.5 Sheets kolleksiyası və Worksheet obyektini, onların xassələri və metodları ...	253
12.6 Range obyektini, onun xassələri və metodları	256

12.7	QueryTables kolleksiyası və QueryTable obyektı	265
12.8	Yekun cədvəllər ilə işləmə (PivotTable obyektı)	270
12.9	Chart obyektı: diaqramlarla işləmə	274
12.10	Excel-in VBA proqramlaşdırılmasında vacib olan başqa obyektləri haqqında ...	279
12.11	VBA Excel proqramlaşdırması ilə kompyuter riyaziyyatı	279
12.11.1	Ən sadə riyazi hesablamalar strukturlu VBA Excel makrosu	281
12.11.2	VBA Excel proqramlaşdırması ilə tərs matrisin hesablanması	284
12.11.3	MS Excel-də VBA proqramlaşdırması ilə adi diferensial tənliklərlə ifadə edilən bəzi sistemlərin modelləşdirilməsi	288
13.	MS Access-də PROQRAMLAŞDIRMA	295
13.1	MS Access-də tətbiqi proqramların yaradılmasındakı xüsusiyyətlər	295
13.2	Access əlavələri yaradılmasının əsas mərhələləri	297
13.3	Application obyektı, onun xassə və metodları	298
13.4	Makrokomandalar və DocMd obyektı	305
13.5	VBA-dan Access formaları ilə işləmə: Form obyektı	306
13.6	Formaların xassələri, metodları və hadisələri	311
13.7	VBA-dan hesabatlarla işləmə: Report obyektı	320
13.8	Access-in başqa obyektləri haqqında	323
14.	MS Outlook-da PROQRAMLAŞDIRMA	325
14.1	Outlook-da VBA proqramlaşdırmasının vacibliyi haqqında	325
14.2	Outlook-dakı proqramlaşdırmanın bəzi xüsusiyyətləri haqqında	330
14.3	Application obyektı, onun xassələri və metodları	332
14.4	Namespace obyektı	335
14.5	Folders kolleksiyası və MAPIFolder obyektı	339
14.6	Items kolleksiyası və Outlook elementlərinin obyektləri	343
14.7	Outlook-un başqa obyektləri haqqında	348
15.	SƏRBƏST İŞLƏMƏK ÜÇÜN TAPŞIRIQLAR (məsləhətlər və cavablarla birgə): MS OFFICE PROQRAMLARINDA VBA PROQRAMLAŞMASI	351
15.1	Word-ə avtomatik rejimdə mətnin daxil edilməsi üçün makrosun yazılması	351
15.2	Makrosun redaktə edilməsi	352
15.3	Dəyişənlər və operatorlarla işləmək vərdişləri (Word və Excel-də VBA vasitələri ilə qarşılıqlı əməllərin aparılması)	354
15.4	Şerti keçid operatorlarının tətbiqi	355
15.5	Dövrü hesablamalar operatorları strukturlarının istifadə edilməsi	357
15.6	Prosedura və funksiyaların tərtib edilməsi	358
15.7	VBA proqramlarında Windows Script Host xarici obyekt modelinin tətbiqi	360
15.8	Proqram səviyyəsində idarəetmə elementlərinin istifadə edilməsi	361
15.8.1	VBA ilə Excel-də sadə formanın yaradılması	361
15.8.2	Excel-də VBA ilə, tərkibində kompleks təyinatlı idarəetmə elementləri olan, formaların yaradılması	364

15.9	İcra müddəti səhvlərinin təyin edilməsi	372
15.10	Sorğu ilə verilənlər bazasından yazıların yüklənməsi	375
15.11	VBA ilə Power Point mühitində proqramlaşdırma	378
15.11.1	Elementlərin slaydlara proqram səviyyəsində əlavə edilməsi	378
15.11.2	VBA PowerPoint Makrosu ilə slaydlardakı sxemlərin avtomatik nömrələnməsi	379
15.12	Proqram səviyyəsində Word sənədinin formalaşdırılması	380
15.13	VBA ilə Excel mühitində proqramlaşdırma	384
15.13.1	Verilənlər bazasında olan informasiyanın analiz edilməsində Excel- də proqramlaşdırma üsullarının tətbiqi	384
15.13.2	Excel VBA proqramlaşdırma mühitində Wilson ədədi metodu əsasında adi diferensial tənliklərlə ifadə edilən dinamik sistemin modelləşdirilməsi	388
15.14	Access-də VBA proqramının yaradılması	394
15.15	VBA mühitində Outlook kontaktları ilə işlərin aparılması	400
ƏDƏBİYYAT	402