VISUAL BASIC 6.0

ՀԱՄԱՌՈՏ ԻՆՔՆՈՒՍՈՒՅՑ

ՊԱՏՐԱՍՏՎԵԼ Է
ՀԱՅ ԹԻՄԻ ԵՎ GRAPHICS INFORMATION TECHNOLOGIES-Ի ԿՈՂՄԻՑ

ՀԵՂԻՆԱԿ՝ ՎԻԿՏՈՐ ՄԱՄՅԱՆ

ՏԵԽՆԻԿԱԿԱՆ ՀՈՎԱՆԱՎՈՐ՝ GRAPHICS INFORMATION TECHNOLOGIES

ԻՆՖՈՐՄԱՑԻՈՆ ՀՈՎԱՆԱՎՈՐ՝ www.hacker.am

ԵՐԵՎԱՆ – 2007

ՑԱՆԿ

ՆԱԽԱԲԱՆ

Սկսնակ ծրագրավորողների առաջ միշտ էլ հարց է առաջանում, թե ինչպիսի ծրագրավորման լեզու ընտրել: Կասեմ, որ կարելի է սկսել հեշտ և միաժամանակ շատ հզոր լեզվից՝ **Visual Basic 6.0**-ից:

Առաջին 16 կարգանի **BASICA**-ն, **IBM PC**-ի համար ստեղծվել է **IBM**-ի կողմից: Որից հետո մշակվել են **GW-BASIC** և **QUICK-BASIC** լեզուները: 1992-ին **Microsoft**-ը թողարկեց **VB 1.0**-ն, որը նախատեսված էր **Windows 3.1**-ի համար:

Visual Basic 6.0-ով գրված ծրագրերը աշխատելու համար պահանջում է **msvbm60.dll** ֆայլը (այդ ֆայլը ավտոմատ տեղադրվում է **Windows XP** ինստալյացիայի ժամանակ):

Visual Basic 6.0 միաժամանակ հանդիսանում է և կոմպիլյատոր և ինտերպրետատոր: Այսինքն կարելի է ստեղծել և գործարկվող ֆայլ (**exe**), որը կաշխատի նաև այն համակարգիչների վրա, որտեղ չկա տեղադրված **VB6** լեզուն, և աշխատում է հենց ծրագրային միջավայրից՝ հնարավորություն տալով թեստավորել ծրագիրը և ուղղել սխալները:

Կան **VB 6**-ի երեք տարբերակներ՝ **Standard Edition, Professional Edition, Enterprise Edition:**

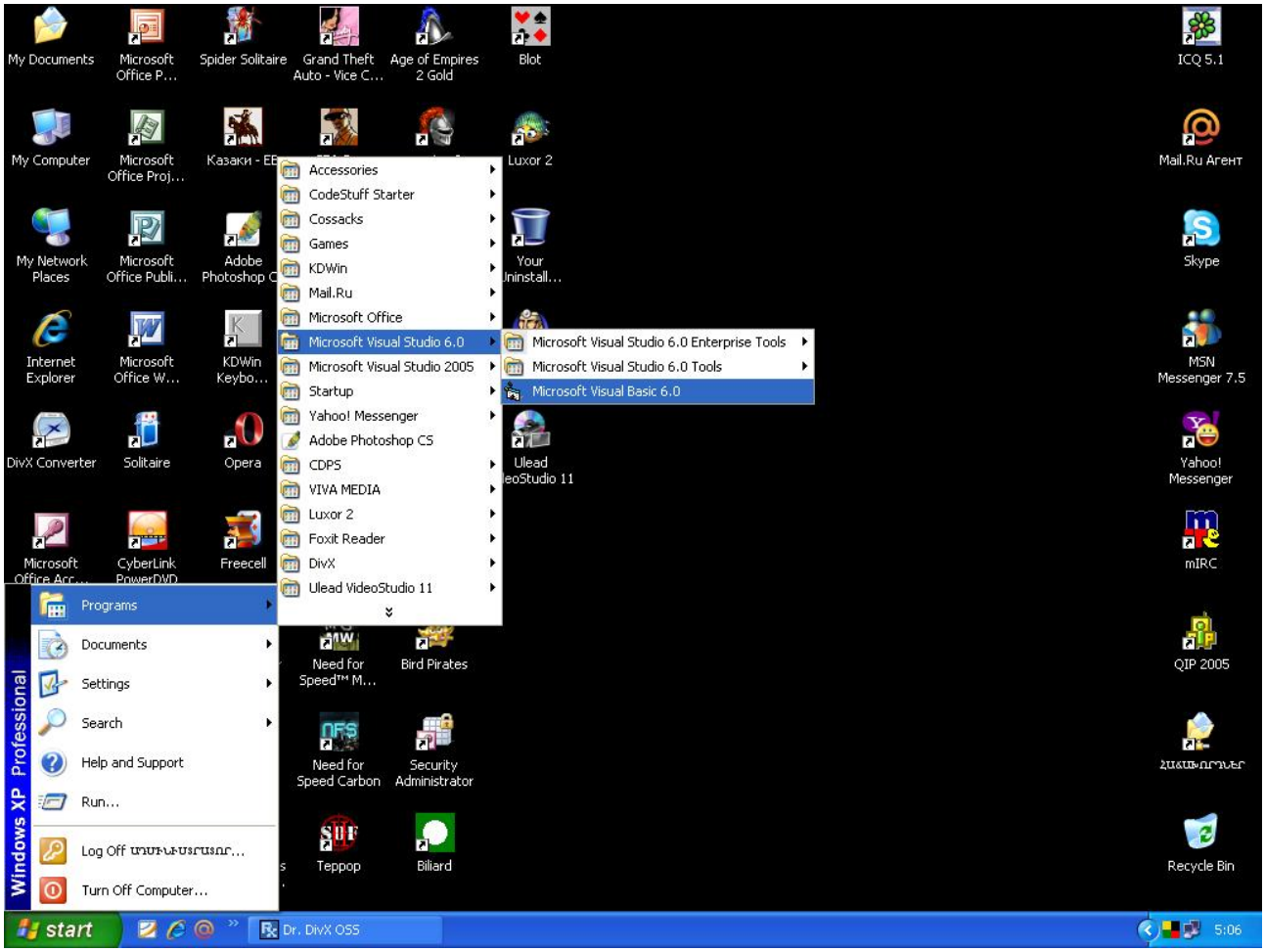
Գլուխ 1. Ծանոթություն Visual Basic 6.0 ծրագրավորման լեզվի հետ

Բաժին 1. Visual Basic 6.0 լեզվի ինտեգրված մշակման միջավայր (IDE)

Visual Basic 6.0-ն գործարկելու համար հարկավոր է կատարել հետևյալ գործողությունները՝

- սեղմել **Start** մենյուն
- ընտրել **Programs** հրամանը
- բացվող մենյուից ընտրել **Microsoft Visual Studio 6.0**
- դրանից հետո ընտրել **Microsoft Visual Basic 6**

Տես նկար 1.1-ը:

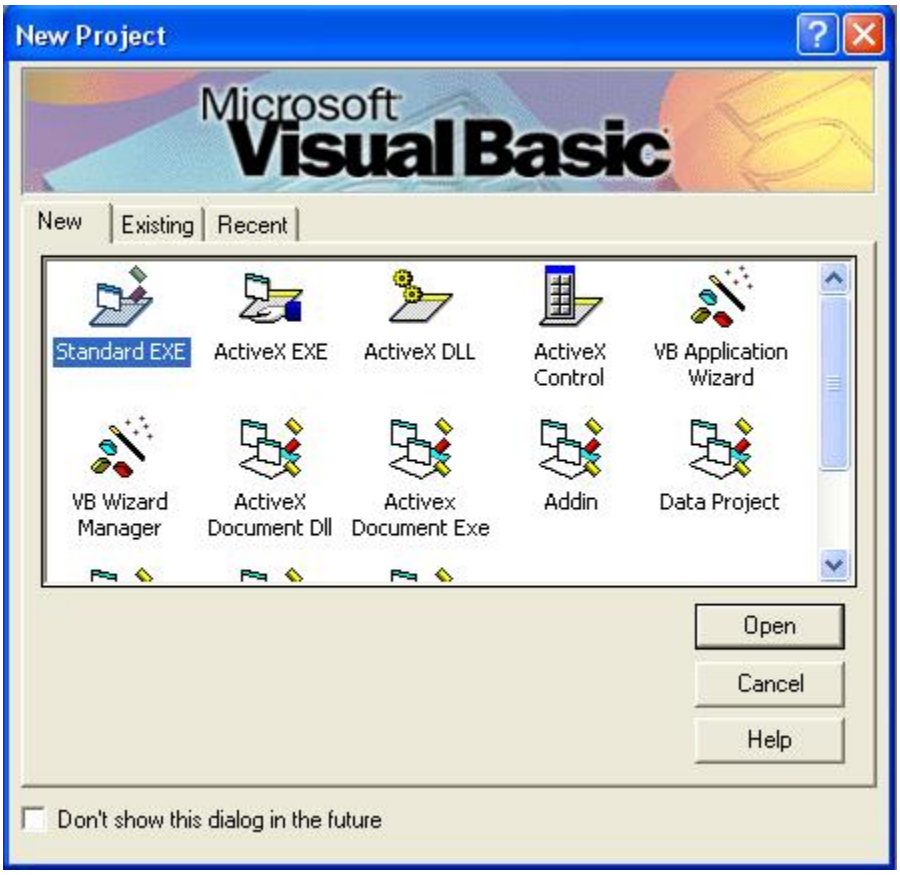


Նկար 1.1

Ավելի արագ գործարկելու համար կարելի է **Desktop**-ին (աշխատանքային սեղան) ստեղծել ծրագրի կանչագիրը:

Visual Basic 6-ը գործարկելուց հետո էկրանին հայտնվում է **New Project** (Նոր Պրոյեկտ) երկխոսական պատուհանը (նկար 1.2): Այդ պատուհանը պարունակում է երեք ներդիրներ՝ հետևյալ պարունակությամբ.

- **New** (նոր) – ստեղծում է նոր պրոյեկտ
- **Existing** (գոյություն ունեցող) – հնարավորություն է տալիս ընտրել պրոյեկտ նախօրոք ստեղծված պրոեկտներից
- **Recent** (վերջին գործարկված) – պարունակում է վերջին բացված մի քանի պրոյեկտների ցանկերը:

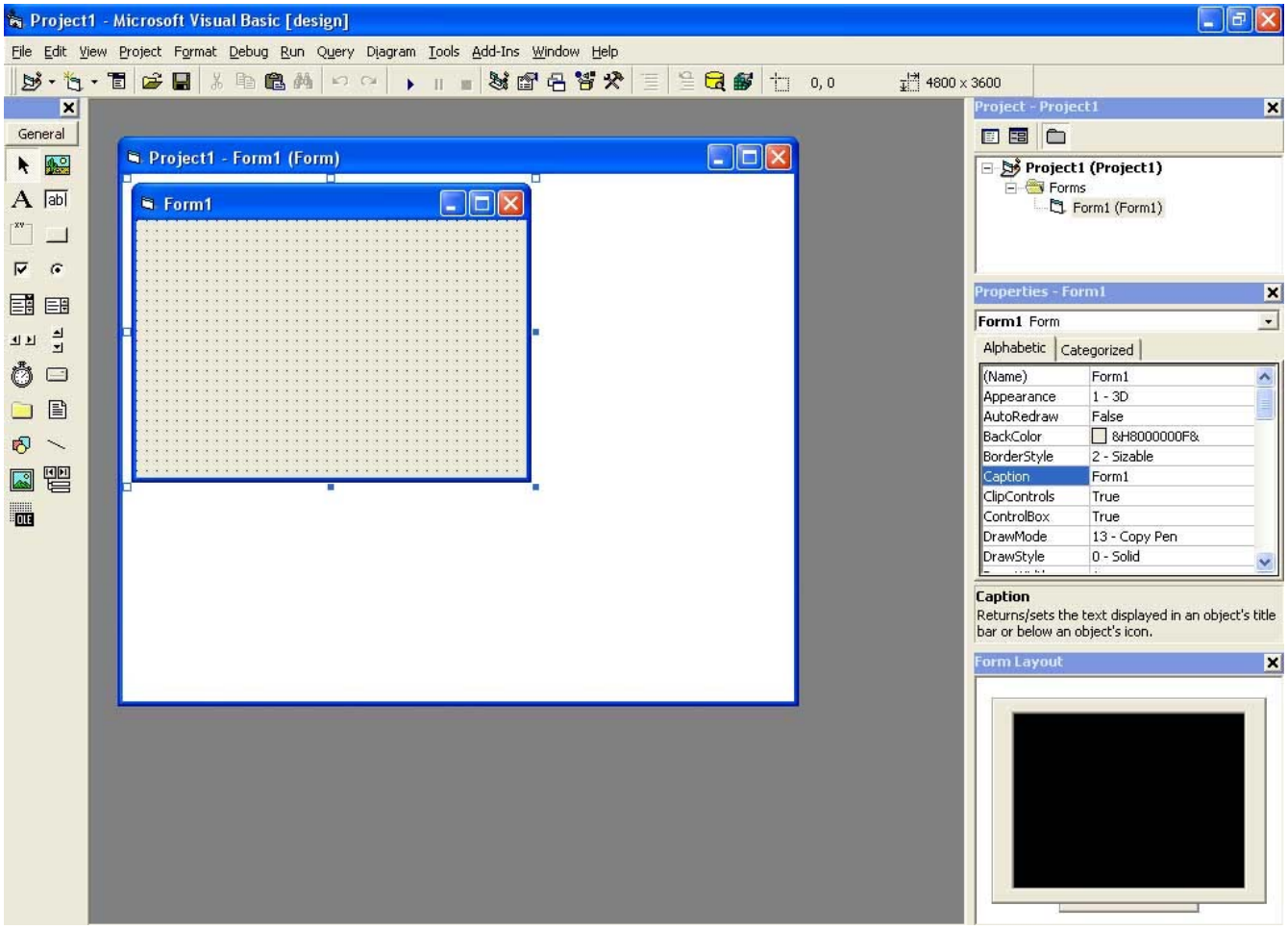


Նկար 1.2

Նոր պրոյեկտ ստեղծելու համար հարկավոր է նշել **New** (նոր) ներդիրը: Այդ պատուհանից ընտրում ենք **Standart.EXE** և սեղմում **Open** (բացել) ստեղնը: **Visual Basic 6.0** ինտեգրված մշակման միջավայրը պատկերված է նկար 1.3 –ում և պարունակում է հետևյալ հիմնական ղեկավարման գործիքները՝

- գլխավոր մենյու
- ստանդարտ գործիքների գոտի
- պրոյեկտի ուղեցույցի պատուհան
- ֆորմայի կոնստրուկտոր
- մենյուի խմբագրիչ
- հատկությունների պատուհան

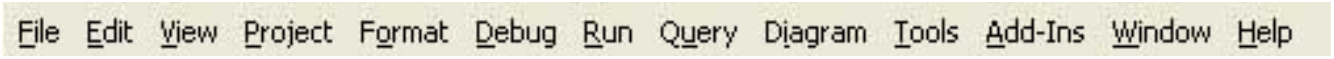
- ֆորմայի դիտարկման պատուհան
- օբյեկտների դիտարկման պատուհան
- կոդի խմբագրման պատուհան



Նկար 1.3

Բաժին 2. Գլխավոր մենյու

Ինչպես **Windows**-ի մնացած հավելվածներում, **Visual Basic 6**-ում նույնպես գլխավոր մենյուն ունի ստանդարտ ձև: Գլխավոր մենյուն պարունակում է հետևյալ հիմնական հրամանները՝ **File**, **Edit**, **View**, **Project**, **Format**, **Debug**, **Run**, **Query**, **Diagram**, **Tools**, **Add-Ins**, **Window**, **Help**: Սակայն մենյուի որոշ հրամաններ տարբերվում են ստանդարտ հրամաններից, քանի որ դրանք նախատեսված են կոնկրետ **Visual Basic 6**-ի համար: Մենյուի պատուհանի նկարը բերված է նկար 1.4 –ում:



Նկար 1.4

Ստորև կներկայացվի հիմնականում ամենաշատ կիրառվող հրամանները իրենց բացատրություններով:

Մենյու **File**.

- **New Project** – ստեղծում է նոր պրոյեկտ
- **Open Project** – բացում է գոյություն ունեցող պրոյեկտ
- **Save Project** – պահպանում է պրոյեկտը
- **Print** – տպում է
- **Make** պրոյեկտի անուն **.exe** – ստեղծում է պրոյեկտի գործարկվող ֆայլը
- **Exit** – ելք պրոյեկտից

Մենյու **Edit**.

- **Undo** – վերջին կատարած գործողությունից մեկ քայլ հետ է կատարում
- **Cut** – կտրում է նշված տեքստը
- **Copy** – պատճենում է նշված տեքստը
- **Paste** – տեղադրում է հիշողությունում առկա ինֆորմացիան
- **Delete** – ջնջում է նշված տեքստը
- **Select All** – նշում է ամբողջ տեքստը
- **Find** – փնտրում է որևէ արտահայտություն

Մենյու **View**.

- **Code** – բացում է ծրագրային կոդի խմբագրման պատուհանը
- **Object** – բացում է ֆորմայի կոնստրուկտորի պատուհանը
- **Object Browser** – բացում է օբյեկտի դիտարկիչի պատուհանը
- **Project Explorer** – բացում է պրոյեկտի դիտարկիչի պատուհանը
- **Toolbox** – բացում է ղեկավարման էլեմենտների գոտու պատուհանը

Մենյու **Project**.

- **Add Form** – պրոյեկտի մեջ ավելացնում է ֆորմա
- **Add MDI Form** - ավելացնում է **MDI** ֆորմա
- **Add Module** - ավելացնում է ծրագրային մոդուլ
- **Add Class Module** - ավելացնում է շահագործողական մոդուլ
- **Add User Control** - ավելացնում է շահագործողական ղեկավարման էլեմենտ
- **Add Property Page** – ավելացնում է հատկությունների կարգաբերման ֆորմա
- **Add WebClass** – ավելացնում է **Web** դաս
- **Add Data Report** – ավելացնում է հաշվետվության պրոյեկտ
- **Add DHTML Page** – ավելացնում է **DHTML** էջ

- **Add Data Environment** – ավելացնում է տվյալների բազայի միջավայր
- **Add File** – ավելացնում է ֆայլ
- **Remove** – ջնջում է պրոյեկտի միջից
- **References** – բացում է պատուհան, որտեղից կարելի է հղում կատարել այլ գրադարանների
- **Components** – բացում է **Components** երկխոսական պատուհանը, որտեղից կարելի է ավելացնել նոր ղեկավարման էլեմենտներ
- **Project Properties** – բացում է պրոյեկտի հատկությունների պատուհանը

Մենյու **Format**.

- **Align** – բացում է մենյու, որտեղից կարելի է կարգաբերել օբյեկտների հավասարությունները
- **Make Same Size** – բացում է մենյու, որտեղից կարելի է կարգաբերել օբյեկտների չափերը
- **Center in Form** – օբյեկտին համաչափեցնում է ֆորմայի կենտրոնի հետ

Մենյու **Debug**.

- **Step Into** – իրականացնում է պրոցեդուրայի քայլային կատարում՝ ներառյալ այն պրոցեդուրաները, որոնք կանչվում են տվյալ պրոցեդուրայի միջից
- **Step Over** - իրականացնում է պրոցեդուրայի քայլային կատարում՝ առանց այն պրոցեդուրաների, որոնք կանչվում են տվյալ պրոցեդուրայի միջից
- **Step Out** – ծրագրի կատարումը տևում է մինչև ընթացիկ պրոցեդուրայի ավարտը

Մենյու **Run**.

- **Start** – գործարկում է ծրագիրը կատարման
- **Start with Full Compile** – գործարկում է ծրագիրը կատարման՝ ամբողջական կոմպիլյացիայով
- **Break** - ժամանակավորապես դադարեցնում է ծրագրի գործարկումը
- **End** – դադարեցնում է ծրագրի աշխատանքը
- **Reatart** – վերագործարկում է ծրագիրը

Մենյու **Query**.

- **Run** – իրականացնում է հարցում
- **Clear Results** – մաքրում է հարցման արդյունքները
- **Verify SQL Syntax** – ստուգում է հարցման համապատասխանությունը
- **Remove Filter** – ջնջում է ֆիլտրը

Մենյու **Diagram**.

- **New Text Annotation** – ավելացնում է բացատրական տեքստ
- **Set Text Font** – տեղադրում է ֆոնտը
- **Add Related Tables** – ավելացնում է կապնված աղյուսակ
- **Arrange Selection** – հավասարեցնում է նշված աղյուսակները
- **Arrange Tables** – հավասարեցնում է բոլոր աղյուսակները

Մենյու **Tools**.

- **Add Procedure** – ավելացնում է պրոցեդուրա
- **Procedure Attributes** – տեղադրում է պրոցեդուրայի ատրիբուտները
- **Menu Editor** – բացում է մենյուի խմբագրման պատուհանը
- **Options** – բացում է **Visual Basic** –ի կարգաբերման երկխոսական պատուհանը

Մենյու **Add-Ins**.

- **Visual Data Manager** – գործարկում է տվյալների բազայի ղեկավարման պատուհանը
- **Add-In Manager** – բացում է **Add-In Manager** երկխոսական պատուհանը, որը հնարավորություն է տալիս կատարել փոփոխություններ **Visual Basic**-ի գործիքների հետ

Մենյու **Window**.

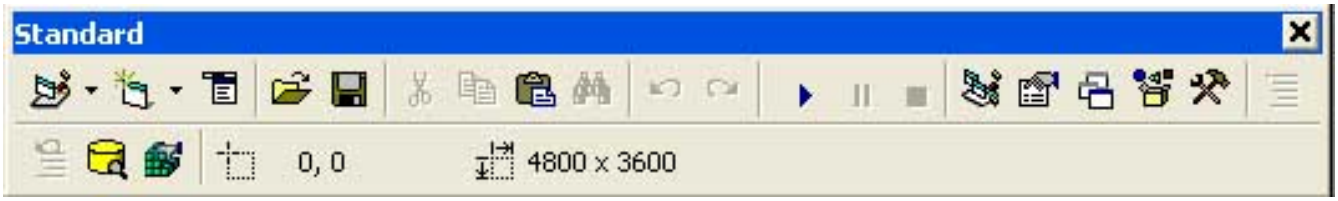
- **Split** – կիսում է պատուհանները
- **Title Horizontally** – պատուհանները դասավորում է հորիզոնական կարգով
- **Title Vertically** - պատուհանները դասավորում է ուղղահայաց կարգով

Մենյու **Help**.

- **Context** – բացում է օգնության պատուհանը
- **Microsoft on the Web** – բացում է **Microsoft**-ը ինտերնետում պատուհանը
- **About Visual Basic** – ցուցադրում է տեղեկատու ինֆորմացիա **Visual Basic 6**-ի վերաբերյալ

Բաժին 3. Ստանդարտ գործիքների գոտի

Ստանդարտ գործիքների գոտին լռելյայն տեղակայված է լինում գլխավոր մենյուի տակը: Եթե այդ գոտին բացակայում է, ապա այն կարելի է ակտիվացնել կատարելով հետևյալը՝ **View** մենյուից ընտրում ենք **Toolbars** ենթամենյուն, որից հետո գործարկում ենք **Standart** հրամանը: Նկար 1.5 –ում պատկերված է ստանդարտ գործիքների գոտին:



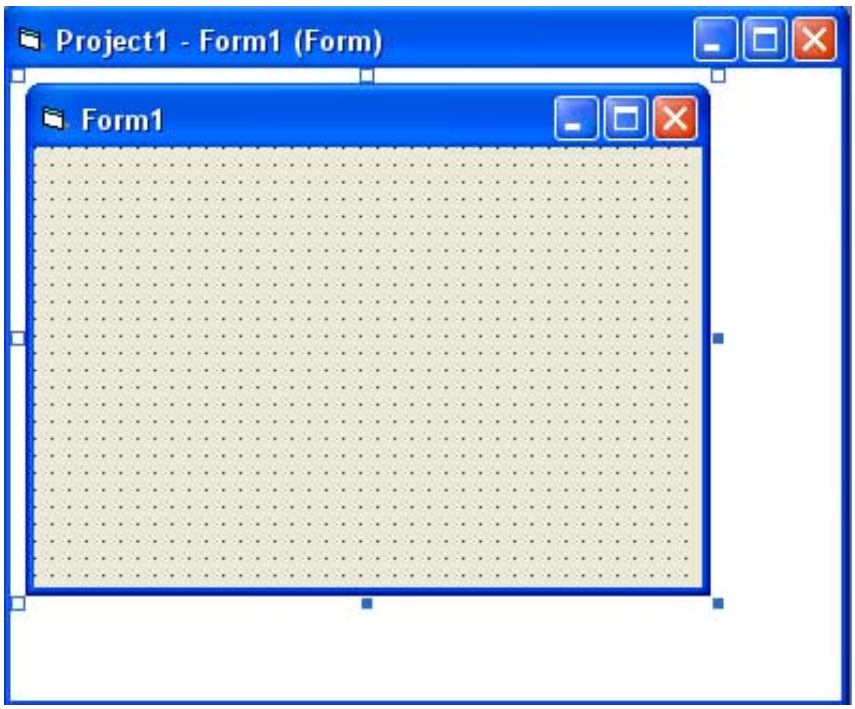
Նկար 1.5

Այժմ ավելի մանրամասն դիտարկենք ստանդարտ գործիքների գոտին:

Standard EXE Project	ավելացնում է ստանդարտ **exe** պրոյեկտ
Add Form	պրոյեկտի մեջ ավելացնում է ֆորմա
Menu Editor	բացում է մենյուի խմբագրիչը
Open Project	բացում է պրոյեկտ
Save Project	պահպանում է պրոյեկտը
Cut	կտրում է նշված տեքստը
Copy	պատճենում է նշված տեքստը
Paste	տեղադրում է տեքստը
Find	փնտրում է տեքստ
Start	գործարկում է ծրագիրը
End	դադարեցնում է ծրագրի կատարումը
Break	ժամանակավոր դադարեցնում է ծրագրի կատարումը
Project Explorer	բացում է պրոյեկտի դիտարկիչը
Properties Window	բացում է հատկությունների պատուհանը
Form Layout Window	բացում է ֆորմայի դիտարկիչի պատուհանը
Object Browser	բացում է օբյեկտների դիտման պատուհանը
Toolbox	բացում է ղեկավարման գործիքների գոտին
Data View Window	բացում է տվյալների դիտման պատուհանը

Բաժին 4. Ֆորմայի կառուցման պատուհան

Ֆորմայի կառուցման պատուհանը հանդիսանում է հիմնական գլխավոր աշխատանքային պատուհանը՝ ծրագրի վիզուալ պրոյեկտավորման համար: Այս պատուհանը կարելի է բացել **View** մենյուից՝ նշելով **Object** հրամանը: Ֆորմայի կառուցման պատուհանի տեսքը բերված է նկար 1.6-ում:



Նկար 1.6

Բաժին 5. Ղեկավարման գործիքների գոտի

Պրոյեկտի վիզուալ կառուցման ժամանակ որպես հիմնական գործիք հանդիսանում է ղեկավարման գործիքների գոտին: Ղեկավարման գործիքների գոտու տեսքը բերված է նկար 1.7 –ում:

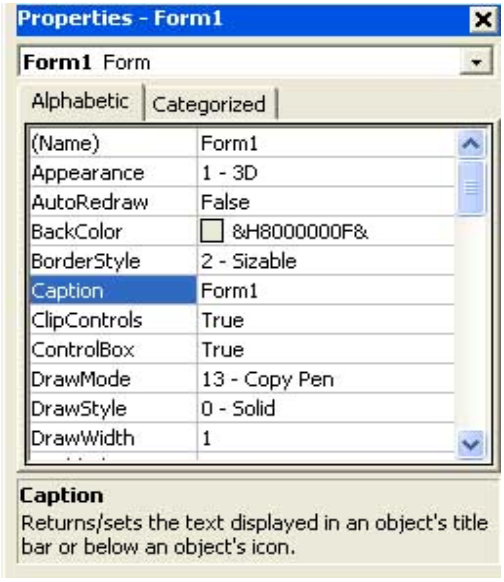
Նկար 1.7

Ղեկավարման գործիքների գոտին կարելի է ակտիվացնել **View** մենյուից՝ ընտրելով **Toolbox** հրամանը: Ղեկավարման գործիքների գոտին պարունակում է ստանդարտ ղեկավարման գործիքներ, սակայն կա այդ գործիքները ավելացնելու հնարավորություն: Ստորև բերվում է ստանդարտ ղեկավարման գործիքների ցանկը.

Pointer	ցուցադրում է մկնիկի նշիչը
PictureBox	հանդիսանում է որպես ֆորմայի մեջ ղեկավարման էլեմենտների կոնտեյներ և գրաֆիկական պատկերներ տեղադրելու համար
Label	նախատեսված է ֆորմայի վրա տեքստ արտապատկերելու համար
TextBox	տեքստային դաշտ է, որտեղ կարելի է հավաքել տեքստ, թվեր և դրանք խմբագրել
Frame	նախատեսված է ղեկավարման գործիքները խմբավորելու համար
CommandButton	Ֆորմայի վրա տեղադրում է ղեկավարման ստեղնը
CheckBox	ֆորմայի վրա տեղադրում է նշիչ, որը աշխատում է այո-ոչ սկզբունքով
OptionButton	ֆորմայի վրա տեղադրում է անջատիչներ
ComboBox	պարունակում է տեքստային դաշտ և բացվող ցանկ
ListBox	ֆորմայի վրա տեղադրում է ցանկ
HScrollBar	հորիզոնական տեղաշարժման ուղի
VScrollBar	ուղղահայաց տեղաշարժման ուղի
Timer	ֆորմայի վրա տեղադրում է ժամացույց
DriveListBox	պարունակում է սկավառակների ցանկը
DirListBox	պարունակում է թղթապանակների ցանկը
FileListBox	պարունակում է ֆայլերի ցանկը
Shape	ֆորմայի վրա տեղադրում է տարբեր գրաֆիկական պատկերներ
Line	տեղադրում է գիծ
Image	նկարներ տեղադրելու համար
Data	տվյալների բազայի հետ աշխատելու համար նախատեսված ղեկավարման գործիք

Բաժին 6. Հատկությունների պատուհան

Հատկությունների պատուհանը նախատեսված է ղեկավարման էլեմենտի հատկությունների ցուցադրման և կարգաբերման համար: Այդ երկխոսական պատուհանը կարելի է արտաբերել **View** մենյուից՝ ընտրելով **Properties** հրամանը: Հատկությունների պատուհանը պատկերված է նկար 1.8 –ում:



Նկար 1.8

Հատկությունների պատուհանում էլեմենտների հատկությունների կարելի է դասավորել երկու ձևով՝ **Alphabetic** (այբբենական) և **Categorized** (ըստ կատեգորիաների): Պատուհանի ներքևի մասում կա բացատրական տող, որը տալիս է հակիրճ ինֆորմացիա նշված հատկության վերաբերյալ:

Այժմ դիտարկենք մի քանի հատկություններ, որոնք բնութագրում են օբյեկտներին:

- **Caption** – տալիս է օբյեկտի վերնագրի տեքստը
- **BorderStyle** – տալիս է օբյեկտի եզրերի տեսքը
- **Enabled** – թույլատրում կամ արգելում է օբյեկտին դիմումը
- **Visible** – տալիս է օբյեկտի տեսանելիության ձևը
- **Name** – տալիս է օբյեկտի անունը
- **Index** – օբյեկտին զանգված սարքելու համար
- **Left** – հեռավորությունը ձախից
- **Top** – հեռավորությունը վերևից
- **Width** – օբյեկտի լայնությունը
- **Height** – օբյեկտի բարձրությունը

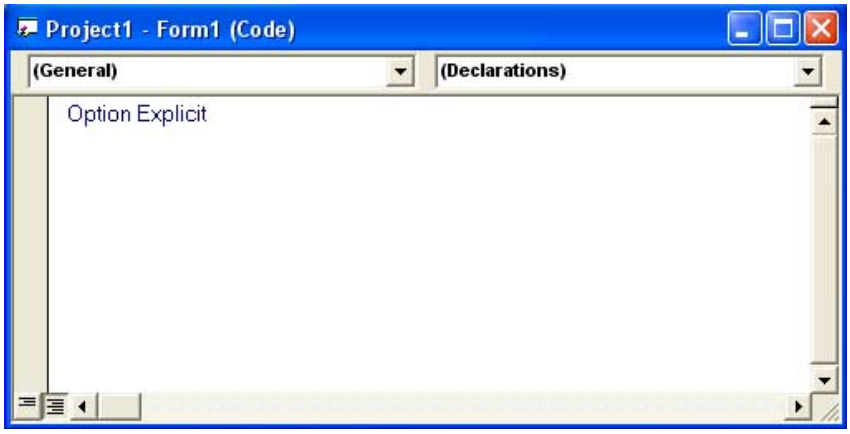
Բաժին 7. Օբյեկտների դիտարկման պատուհան

Օբյեկտների դիտարկման պատուհանի միջոցով կարելի է դիտել պրոյեկտի մեջ մտնող բոլոր օբյեկտների վերաբերյալ անհրաժեշտ ինորմացիան: Բացի այդ, այն մանրամասն ինֆորմացիա է տալիս այդ օբյեկտների հատկությունների վերաբերյալ: Օբյեկտների դիտարկման պատուհանը կարելի է ակտիվացնել **View** մենյուից՝ ընտրելով **Object Browser** հրամանը: Օբյեկտների դիտարկման պատուհանի նկարը բերված է նկար 1.9-ում:

Նկար 1.9

Բաժին 8. Ծրագրային կոդի խմբագրման պատուհան

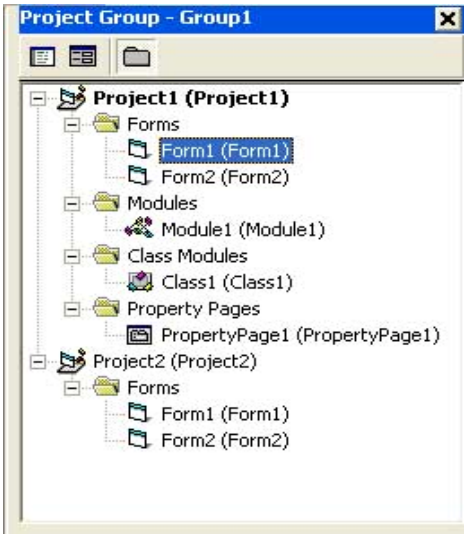
Ծրագրային կոդի խմբագրման պատուհանը պարունակում է շատ լուրջ խմբագրից, որով կարելի է խմբագրել մուտքագրվող կամ մուտքագրված ծրագրային կոդը: Այդ պատուհանը կարելի է ակտիվացնել **View** մենյուից՝ ընտրելով **Code** հրամանը: Ծրագրային կոդի խմբագրման պատուհանի տեսքը բերված է նկար 1.10-ում:



Նկար 1.10

Բաժին 9. Պրոյեկտի դիտարկիչի պատուհան

Պրոյեկտի դիտարկիչի պատուհանը շատ նման է **Windows**-ի դիտարկիչի պատուհանին և թույլ է տալիս շատ արագ տեսնել պրոյեկտի մեջ մտնող բոլոր օբյեկտներին: Պրոյետի դիտարկիչի պատուհանը կարելի է ակտիվացնել **View** մենյուից՝ ընտրելով **Project Explorer** հրամանը: Պրոյեկտի դիտարկիչի տեսքը բերված է նկար 1.11-ում:



Նկար 1.11

Գլուխ 2. Հավելվածի ստեղծումը Visual Basic 6.0-ի միջոցով

Բաժին 1. Աշխատանք պրոյեկտի հետ

Visual Basic 6.0-ի միջավայրում ստեղծվող հավելվածների մեծ մասն աշխատում է ինտերակտիվ ռեժիմում: Էկրանին հայտնվում է ինֆորմացիա, որը նախատեսված է ծրագրի շահագործողի համար և սպասում է նրա պատասխան ռեակցիային՝ ինֆորմացիայի ներմուծման կամ հրամանների տեսքով: **Visual Basic**-ում ինտերակտիվ հավելված ստեղծվում է ֆորմայի հիման վրա, որը ինտերֆեյսի հիմնական մասն է կազմում՝ ղեկավարման գործիքների հետ միասին:

Visual Basic 6.0-ի միջավայրում ցանկացած հավելվածի ստեղծումը սկսվում է պրոյեկտի ստեղծումով: Որպեսզի ստեղծենք նոր պրոյեկտ, հարկավոր է **File** մենյուից ընտրել **New Project** հրամանը և ընտրել համապատասխան պրոյեկտի տեսակը:

Պրոյեկտի հետ աշխատելու ժամանակ հարկավոր է, որ պարբերաբար պահպանվի ինֆորմացիան: Պրոյեկտը պահպանելու համար հարկավոր է կատարել հետևյալ քայլերը՝

1. **File** մեյուից ընտրել **Save Project** հրամանը
2. Բացվող **Save File As** երկխոսական պատուհանում գրել ֆայլի անունը և սեղմել **Save** ստեղնը

Պրոյեկտը բացելու համար հարկավոր է **File** մենյուից ընտրել **Open Project** հրամանը: Բացվող երկխոսական պատուհանից նշել պրոյեկտը և սեղմել **Open** ստեղնը:

Բաժին 2. Իրադարձություն և մեթոդ

Visual Basic 6.0 հանդիսանում է օբյեկտակողմնորոշված ծրագրավորման լեզու: Օգտագործելով օբյեկտի համար նախատեսված մեթոդները՝ կարելի է մինիմումի հասցնել ծրագրավորման կոդի ծավալը: Օրինակ, ֆորմայի տեսքը տպելու համար կարելի է գրել հետևյալը՝ Form1.PrintForm : Որտեղ **Form1**-ը ֆորման է, **PrintForm**-ը՝ մեթոդը:

Կան մեթոդներ, որոնք օգտագործվում են բոլոր օբյեկտների համար: Օրինակ՝ **move** (տեղաշարժել օբյեկտը) և այլն: Բացի հատկություններից ու մեթոդներից օգտագործվում է

նաև ծրագրային կոդ: Օրինակ՝ հրամանային ստեղնին սեղմելու դեպքում տեղի է ունենում **Click** իրադարձությունը:

Որպեսզի բացենք ծրագրային կոդի խմբագրման պատուհանը, հարկավոր է կատարել հետևյալը՝

1. կրկնակ հարված ենք կատարում մկնիկով օբյեկտի վրա
2. նշում ենք օբյեկտը և **View** մենյուից ընտրում **Code** հրամանը

Ծրագրային կոդի խմբագրման պատուհանի տեսքը բերված է նկար 2.1 –ում:

Նկար 2.1

Ծրագրային կոդի խմբագրման պատուհանը վերևի մասում պարունակում է երկու բացվող ցանկեր՝ **Object** (օբյեկտ) և **Procedure** (պրոցեդուրա): **Object** ցանկում պարունակում է ֆորմայի բոլոր օբյեկտների անունները: **Procedure** ցանկում պարունակում է իրադարձությունների անունները, որոնց համար կարելի է ստեղծել պրոցեսուրա:

Ծրագրային կոդի խմբագրման հատվածում գրված է հետևյալը՝

```
Private Sub Command1_Click()
End Sub
```

Այստեղ **Command1_Click** պրոցեդուրայի անունն է:

Գլուխ 3. Ծրագրավորման հիմնական էլեմենտները

Բաժին 1. Փոփոխականներ

Փոփոխականը դա ռեզերվացված տարածություն է օպերատիվ հիշողությունում, որը նախատեսված է ժամանակավոր տվյալների պահպանման համար: Ցանկացած փոփոխական ունի անուն: Այդ անունները շնորհելիս պետք է հետևել մի շարք կանոնների, որը չի կարելի խախտել: Այդ կանոններն են՝

- փոփոխականի անունի սիմվոլների երկարությունը չի կարող գերազանցել 255 սիմվոլ
- փոփոխականի անունը կարող է պարունակել ցանկացած տառ և թիվ
- անվան առաջին սիմվոլը պետք է լինի տառ
- անվան մեջ պետք է բացակայի պրոբել
- անունը պետք է լինի ունիկալ իր տեսանելիության սահմաններում

Բաժին 2. Տվյալների տեսակները

Visual Basic-ում կարող են օգտագործվել հետևյալ տեսակի տվյալների տեսակներ.

- Թվային (**Integer**, **Long**, **Single**, **Double**, **Currency**)
- Տեքստային (**String**)
- Ամսաթվային (**Date**)
- Բայթային (**Byte**)
- Տրամաբանական (**Boolean**)
- Կամայական (**Variant**)
- Օբյեկտային (**Object**)

Ամբողջ թվերի հետ աշխատելու համար օգտագործվում են **Integer** և **Long** տիպի տվյալի տեսակները: Որոնց դիապազոններն են՝

Integer – (-32768) – (32767)

Long – (-2147483648) – (2147483648)

Single և **Double** տեսակները օգտագործվում են սահող ստորակետով թվերի պահպանման համար: Դիապազոններն են՝

Single – $(-3,4^{38} \dots -1,4^{-45}) – (1,4^{-45} \dots 3,4^{38})$

Double – $(-1,7^{-308} \dots -4,9^{-324}) – (4,9^{-324} \dots 1,7^{+308})$

Ֆիքսված ստորակետով թվերի համար օգտագործվում է **Currency** տեսակը (ամբողջ մասը կարող է պարունակել մինչև 15 սիմվոլ, ստորակետից հետո՝ 4):

Բաժին 3. Փոփոխականների հայտարարումը

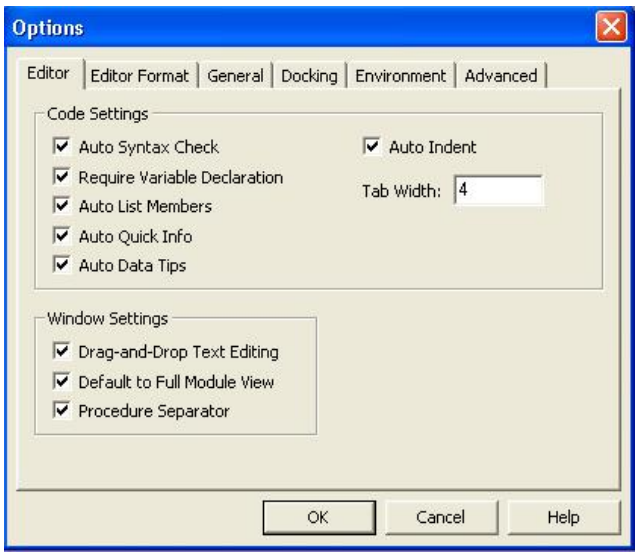
Visual Basic 6.0-ում օգտագործվում է փոփոխականների հայտարարման բացահայտ և ոչ բացահայտ ձևերը: Բացահայտ հայտարարումը իրականացվում է **dim**, **private**, **static** և **public** օպերատորների միջոցով: Հայտարարման ձևը հետևյալն է՝

```
Dim փոփոխականի_անուն  [As տվյալի_տեսակ]
```

Մեկ օպերատորի միջոցով կարելի է հայտարարել բազմաթիվ փոփոխականներ՝ դրանք բաժանելով ստորակետով: Օրինակ՝ `dim strText as String, I as Long` :

Visual Basic-ում հիմնականում շատ է կիրառվում String տիպը, որը կարող է ունենալ մինչև 2 ԳԲ երկարություն, սակայն մենք կարող ենք սահմանափակել այդ երկարությունը՝ օգտվելով աստղանիշից: Օրինակ՝ `dim strText as String * երկարություն`

Ցանկալի է, որ փոփոխականները հայտարարվեն բացահայտ ձևով: Դրա համար հարկավոր է մոդուլի սկզբից ավելացնել **Option Explicit** օպերատորը: Որպեսզի այն ավտոմատ ավելացվի, հարկավոր է **Tools** մենյուից ընտրել **Options** հրամանը: Բացվող **Options** պատուհանի **Editor** ներդիրի մեջ նշել **Require Variable Declaration** նշիչը, նկար 3.1:



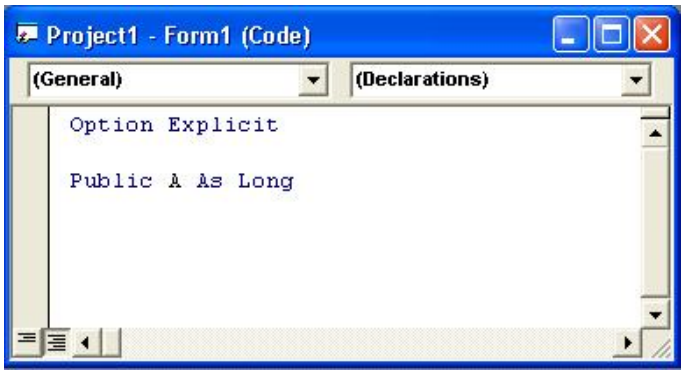
Նկար 3.1

Բաժին 4. Փոփոխականների տեսանելիության գոտին

Ծրագրի ճիշտ աշխատանքի համար շատ կարևոր է ճիշտ ընտրել փոփոխականների տեսանելիության գոտին: **VB**-ում կարելի է օգտագործել գլոբալ և լոկալ փոփոխականներ: Գլոբալ փոփոխականները հասանելի են ծրագրի ցանկացած մասից: Լոկալի ժամանակ կարելի է հասանելիությունը սահմանել կամ կոնկրետ մոդուլի շրջանակներում կամ պրոցեդուրայի: Որպեսզի փոփոխականը հայտարարել որպես գլոբալ, հարկավոր է, որ մոդուլի **General Declaration** հատվածում այն հայտարարել որպես **Public**: Օրինակ՝

```
Public A as Long
```

Տես նկար 3.2



Նկար 3.2

Իսկ պրոցեդուրայի սահմաններում հայտարարվում է հետևյալ կերպ՝

```
Sub Calculate()
        Dim A as Long
End Sub
```

Բաժին 5. Հաստատուններ

Հաստատունը այնպիսի արտահայտություն է, որի արժեքը չի փոփոխվում ծրագրի կատարման ամբողջ ընթացում: Հաստատունների օրինակներ են՝

105,267 – թվային հաստատուն

“Տվյալները պահպանվել են” – տեքստային հաստատուն

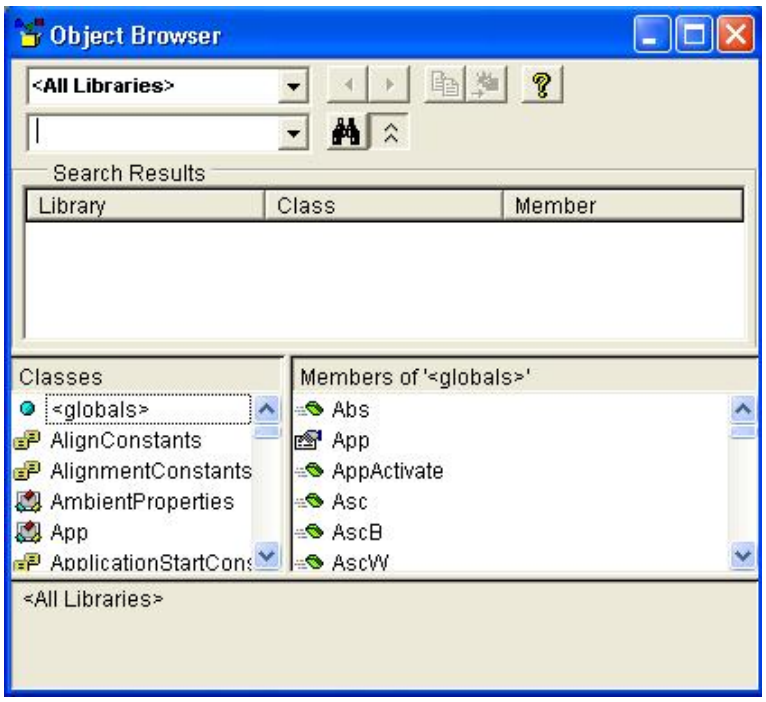
#7/7/2007# - ամսաթվային հաստատուն

True – տրամաբանական հաստատուն

Հաստատունները հայտարարվում են այնպես, ինչպես փոփոխականները և տեսանելիության գոտին նույնպես նույնն է: Հաստատունի հայտարարման օրինակ.

```
Dim Const A As Long = 272545
```

VB-ն ունի ներկառուցված հաստատուններ: Այդ հաստատունների ցանկը կարելի է տեսնել **Object Browser**-ի միջոցով: Տես նկար 3.3:



Նկար 3.3

Բաժին 6. Զանգվածներ

Տվյալները կարելի է պահպանել ոչ միայն փոփոխականների մեջ, այլ նաև զանգվածների: Զանգվածը փոփոխականների հավաքածու է, որը ունի մեկ անուն և այդ փոփոխականները իրարից տարբերվում են ինդեքսով: Այդպիսի ամեն մի փոփոխական կոչվում է զանգվածի տարր: Այդ տարրերի քանակը կոչվում է զանգվածի չափ: Զանգվածի չափը սահմանափակվում է օպերատիվ հիշողությունով և տվյալների տեսակով: Զանգվածի բոլոր տարրերը ունեն տվյալների նույն տիպը: Սակայն, եթե զանգվածը **Variant** տիպի է, ապա զանգվածի տարրերը կարող են պարունակել տարբեր տիպի տվյալներ: Զանգվածի տարրերի ինդեքսը նշվում է փակագծերի մեջ՝ զանգվածի անունից հետո: Օրինակ՝ strNames(1), strNames(2), strNames(25) և այլն:

VB-ում գոյություն ունեն ֆիքսված չափով և դինամիկ զանգվածներ: Ֆիքսված զանգվածի չափը նշվում է հայտարարման ժամանակ, իսկ դինամիկ զանգվածի չափը կարող է փոփոխվել ծրագրի կատարման ժամանակ:

Ֆիքսված չափով զանգվածների հայտարարումը կատարվում է հետևյալ կերպ՝ կախված տեսանելիության գոտուց.

- Գլոբալ զանգվածը հայտարարվում է մոդուլի **Declaration** հատվածում՝ **Public** օպերատորի միջոցով

- Մոդուլի մակարդակով զանգվածը հայտարարվում է **Declaration** հատվածում՝ **Private** կամ **Dim** օպերատորների միջոցով
- Լոկալ զանգվածը՝ **Private** կամ **Dim** օպերատորների միջոցով՝ պրոցեդուրայի ներսում

Զանգվածի հայտարարման ժամանակ նրա անունից հետո փակագծերի մեջ նշվում է զանգվածի չափը: Լռելյայն ներքին սահման է ընդունվում 0: Օրինակ՝ եթե զանգվածը պետք է պարունակի 5 տարր, ապա կգրվի՝ Dim A(4) As String :

Կարելի է բացահայտ տալ զանգվածի ներքին սահմանը **To** բանալիային բառի միջոցով: Օրինակ՝ Dim A(1 To 20) As Long :

VB-ն հնարավորություն է տալիս օգտագործել բազմաչափ զանգվածներ: Օրինակ՝ Dim A (20,20) As Long: Զանգվածի չափողականությունը կլինի 21x21:

Այլ օրինակներ՝

```
Dim m (1 To 20, 5 To 10) As Long
Dim k (30, 25 To 30) As Long
```

Այն դեպքում, երբ զանգվածի չափողականությունը նախօրոք հայտնի չէ, **VB**-ն հնարավորություն է տալիս օգտագործել դինամիկ զանգվածներ, որոնց չափողականությունը կարելի է փոփոխել ծրագրի կատարման ժամանակ:

Դինամիկ զանգված ստեղծելու համար կատարվում են հետևյալ քայլերը՝

- Հայտարարվում է զանգված՝ Dim A() As Long
- **Redim** օպերատորի միջոցով պրոցեդուրայի մեջ վերահայտարարվում է զանգվածը՝

```
Redim A(x)
Redim A(20)
```

Redim A (1 To 20) և այլն:

Եթե ցանկանում եք փոփոխել զանգվածի չափը, առանց կորցնելու զանգվածի տվյալները, ապա **Redim** օպերատորը օգտագործվում է **Preserve** բառի հետ միասին՝ Redim Preserve A(x+1):

Բաժին 7. Մեկնաբանություն

Բացի հրամաններից, ծրագրային կոդի գրման ժամանակ կարող եք կոդի դաշտում գրել տարբեր տեքստեր ու մեկնաբանություններ, որոնք հետագայում կհեշտացնեն այդ կոդի կարդալն ու հասկանալը: Որպեսզի այդ տեքստերը չհասկացվեն որպես ծրագրային կոդ, օգտագործվում է ապաթարցի նշանը (‘):

Օրինակ՝ Debug.Print “Որևէ տեքստ” ‘ արտատպում է տեքստը

Բաժին 8. Պրոցեդուրաներ

Ծրագրավորման ժամանակ մեծ նշանակություն ունեն պրոցեդուրաները, որոնք ծրագրային կոդը բաժանում են տրամաբանական մասերի: **VB**-ում կան հետևյալ պրոցեդուրաները՝ **Sub**, **Function**, **Property**:

Sub պրոցեդուրան որևէ արժեք չի վերադարձնում և ամենաշատ օգտագործվող պրոցոդուրան է: Սինտակսիսը հետըյալն է՝

```
[Private] [Public] [Static] Sub պրոցեդուրա (արգումենտներ)
        ծրագրային կոդ
End Sub
```

Sub-ի և End Sub-ի միջև գրվում է ծրագրային կոդը, որը պետք է կատարվի այդ պրոցեդուրան կանչելու դեպքում:

Ի տարբերություն Sub պրոցեդուրայի, Function պրոցեդուրան վերադարձնում է արժեք: Սինտակսիսը հետևյալն է՝

```
[Private] [Public] [Static] Function ֆունկցիա (արգումենտներ) [As Type]
        ծրագրային կոդ
End Function
```

Sub տիպի պրոցեդուրան կարող է կանչվել երկու ձևով: Առաջին դեպքում օգտագործվում է **Call** բանալիային բառը: Օրինակ՝ `call MyProc (արգումենտ1,… արգումենտN)`

Երկրորդ դեպքում գրվում է միայն պրոցեդուրայի անունը:

```
MyProc արգումենտ1,… արգումենտN
```

Եթե պրոցեդուրան կանչվում է այլ մոդուլից, ապա պետք է բացահայտ նշվի մոդուլի անունը: Օրինակ՝ `call Form1.MyProc (արգումենտ1,… արգումենտN)`

Function պրոցեդուրայի կանչը նման է **Sub**-ի կանչին: Սակայն կա նաև մի փոքր տարբերություն: Եթե պետք է մշակվի վերադարձվող արժեքը, ապա կարելի է գրել հետևյալ կերպ՝ `S=MyFunction ()`:

Գլուխ 4. Ղեկավարման կառուցվածքներ և ցիկլեր

Բաժին 1. Ղեկավարման կառուցվածքները VB-ում

VB-ում, ինչպես և մնացած ծրագրավորման լեզուներում, գոյություն ունեն ղեկավարող կառուցվածքներ, որոնք կարգավորում են հրամանների կատարման կողը: Տարբերվում են երկու հիմնական ղեկավարման օպերատորներ՝ **If** և **Select Case**: Իր հերթին **If** օպերատորը կարող է լինել երկու ձևի՝ If … Then և If … Then … Else:

Բաժին 2. Պայմանի ստուգման արտահայտություններ

Պայմանի ստուգման արտահայտությունը կարող է ունենալ երկու արժեքներից միայն մեկը՝ **True**(ճիշտ) և **False**(սխալ): **VB**-ում համեմատման օպերատորները հետևյալներն են՝

=	հավասարության նշան
>	մեծի նշան
<	փոքրի նշան
<>	անհավասարության նշան (հավասար չէ)
>=	մած կամ հավասար
<=	փոքր կամ հավասար

Պայմանի ստուգման օպերատորների հետ կարելի է կատարել տրամաբանական մաթեմատիկայի գործողություններ, մասնավորապես՝

AND - և

OR – կամ

XOR – կամի բացառում

NOT – ոչ

Բաժին 3. If…Then կառուցվածքը

If…Then կառուցվածքը օգտագործվում է այն ժամանակ, երբ անհրաժեշտ է կատարել մեկ կամ մի քանի օպերացիաներ՝ կախված կոնկրետ պայմաններից (այսինքն երբ պայմանը **True** է): Դիտարկենք օրինակ:

```
If պայման Then
        գործողություն
End if
```

կամ

```
If y>20 then y=2  ‘ մեկ տողանի օպերատոր
If y>20 then
        y=2
End If    ‘ բազմատողային օպերատոր
```

կամ

```
If պայման Then
        գործողություն 1
Else
        գործողություն 2
End If
```

կամ

```
If պայման then
        գործողություն 1
ElseIf մեկ_այլ_պայման Then
        գործողություն 2
ElseIf մեկ_այլ_պայման Then
        գործողություն 3
……
Else
        վերջնական գործողություն
End If
```

Բաժին 4. Select Case կառուցվածքը

Select Case կառուցվածքը նման է **If…Then** կառուցվածքին իր բնույթով: Սինտակսիսը հետևյալն է՝

```
Select Case համեմատվող_արտահայտություն
        Case արժեք1
                գործողություն1
        Case արժեք2
                Գործողություն2
        Case արժեքX-ից արժեքA
                ԳործողությունM
        …..
        Case Else
                գործողությունN
        End Select
```

Բաժին 5. Ցիկլեր

For…Next ցիկլը նախատեսված է նույն գործողությունը բազմաթիվ անգամ կրկնելու համար: Սինտակսիսը հետևյալն է՝

```
For հաշվիչ=սկզբնական_արժեք To վերջնական_արժեք [Step քայլ]
        գործողություն
Next հաշվիչ
```

Հաշվիչը փոփոխականի անունն է, որը հաշվում է կատարած ցիկլի քանակը: Դիտարկենք օրինակ՝

```
For i=0 to 5
        MsgBox “Բաժանորդ N” & i+1
```

```
Next i
```

Step-ը ցույց է տալիս քայլերի քանակը. եթե չի գրվում, ընդունում է 1: Այստեղ օգտագործված **&** նշանը միացնում է երկու արտահայտություններ իրար:

Do…Loop ցիկլը կատարում է այնքան ժամանակ, քանի դեռ չի հասել պայմանյում նշված արժեքին: Այն ունի երկու ձև՝ պայմանը ստուգում է սկզբում կամ վերջում: Եթե պայմանը ստուգում է վերջում, ապա ցիկլը կատարվում է գոնե մեկ անգամ: Սինտակսիսը հետևյալն է՝

```
Do While պայման
        գործողություն
Loop
Do Until պայման
        գործողություն
Loop
```

կամ

```
Do
        գործողություն
Loop While պայման
Do
        գործողություն
Loop Until պայման
```

Օրինակ՝

```
i=0
Do While i<=5
        MsgBox “Բաժանորդ N” & i+1
Loop
```

Գլուխ 5. Շահագործողի ինտերֆեյսի մշակում

Բաժին 1. Ինտերֆեյս

Ինտերֆեյսը նեղ առումով ծրագրի արտաքին տեսքն է՝ շահագործողից թաքցված իր ներքին կառուցվածքի հետ միասին: Ցանկացած ծրագրի համար մեծ նշանակություն ունի ինտերֆեհսի հետ աշխատանքի էֆեկտիվությունը՝ ինֆորմացիան մշակելու համար: Այդ առումով ինտերֆեյսը ամեն մի ծրագրի կարևոր մասն է հանդիսանում:

Բաժին 2. Ընդհանուր խորհուրդներ ինտերֆեյսի մշակման համար

Ինտերֆեյս մշակելիս կարևոր է առաջնորդվել հետևյալ սկզբունքներով՝

- Ստանդարտացում: Խորհուրդ է տրվում մշակել ստանդարտ ինտերֆեյս, որն արդեն փորձված է բազմաթիվ ծրագրավորողներիի ու շահագործողների կողմից: Օրինակ կարող է հանդիսանալ **MS Word** ծրագրի ինտերֆեյսը:
- Հարմարություն և աշխատանքի հեշտություն: Ինտերֆեյսը պետք է լինի պարզ և հասկանալի: Նրանում կատարվող գործողությունը պետք է լինի հեշտ հիշվող:
- Արտաքին դիզայն: Ինտերֆեյսը պետք է նախատեսված լինի երկար աշխատանքի համար, որը չպետք է հոգնեցնի շահագործողի աչքերը:
- Ֆորմաների քիչ բեռնում: Հարկավոր է հնարավորինս քից քանակի ֆորմաներ բեռնել: Անհրաժեշտության դեպքում կարելի է օգտագործել ներդիրներ:
- Խմբավորում: Ղեկավարման էլեմենտները հարկավոր է խմբավորել ըստ իրենց նշանակության և իմաստի:
- Ղեկավարման էլեմենտների դիրքը: Ղեկավարման էլեմենտները պետք է լինեն իրարից որոշակի հեռավորության վրա, այլ ոչ իրար վրա կամ շատ մոտ իրար:

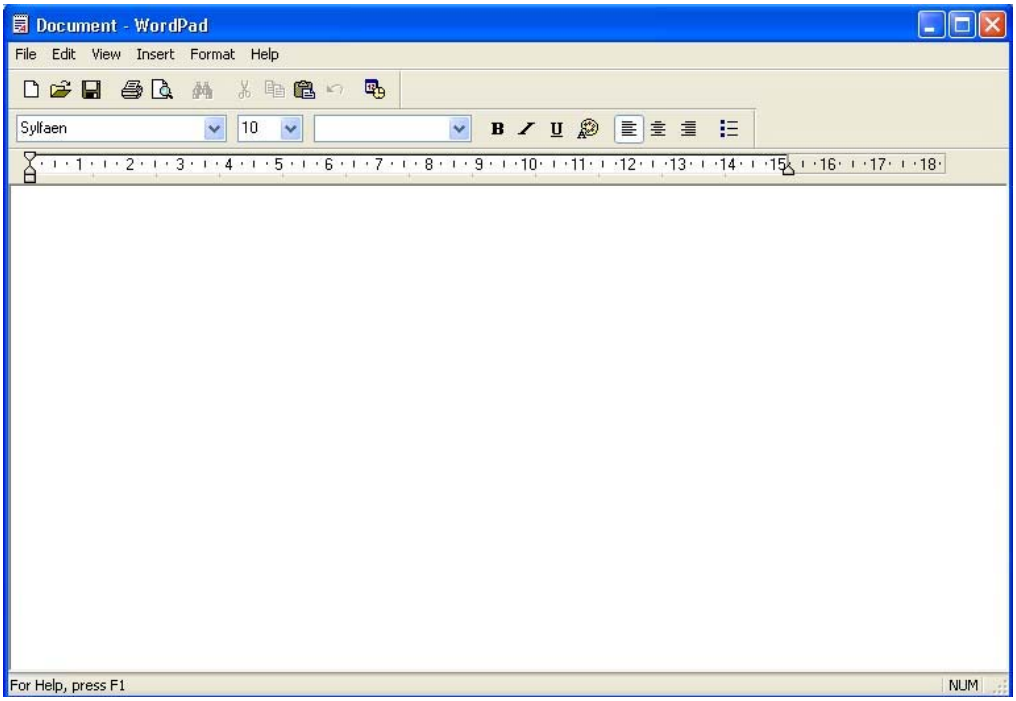
Սրանք հիմնական սկզբունքներն են, բայց դա չի նշանակում, որ բոլոր ծրագերը պետք է պարտադիր ենթարկվեն այդ սկզբունքներին: Կախված ծրագրի նպատակներից, այն կարող է ունենալ ոչ ստանդարտ ինտեֆեյս:

Բաժին 3. Ինտերֆեյսի տեսակները

Visual Basic 6.0-ում օգտագործվում են երեք տեսակի ինտերֆեյսներ՝ մեկ փաստաթղթային (**SDI**), բազմափաստաթղթային (**MDI**) և դիտարկիչ տեսակի (**Explorer**):

SDI ինտերֆեյս

SDI տիպի ինտերֆեյսի օրինակ պատկերված է նկար 5.1-ում:



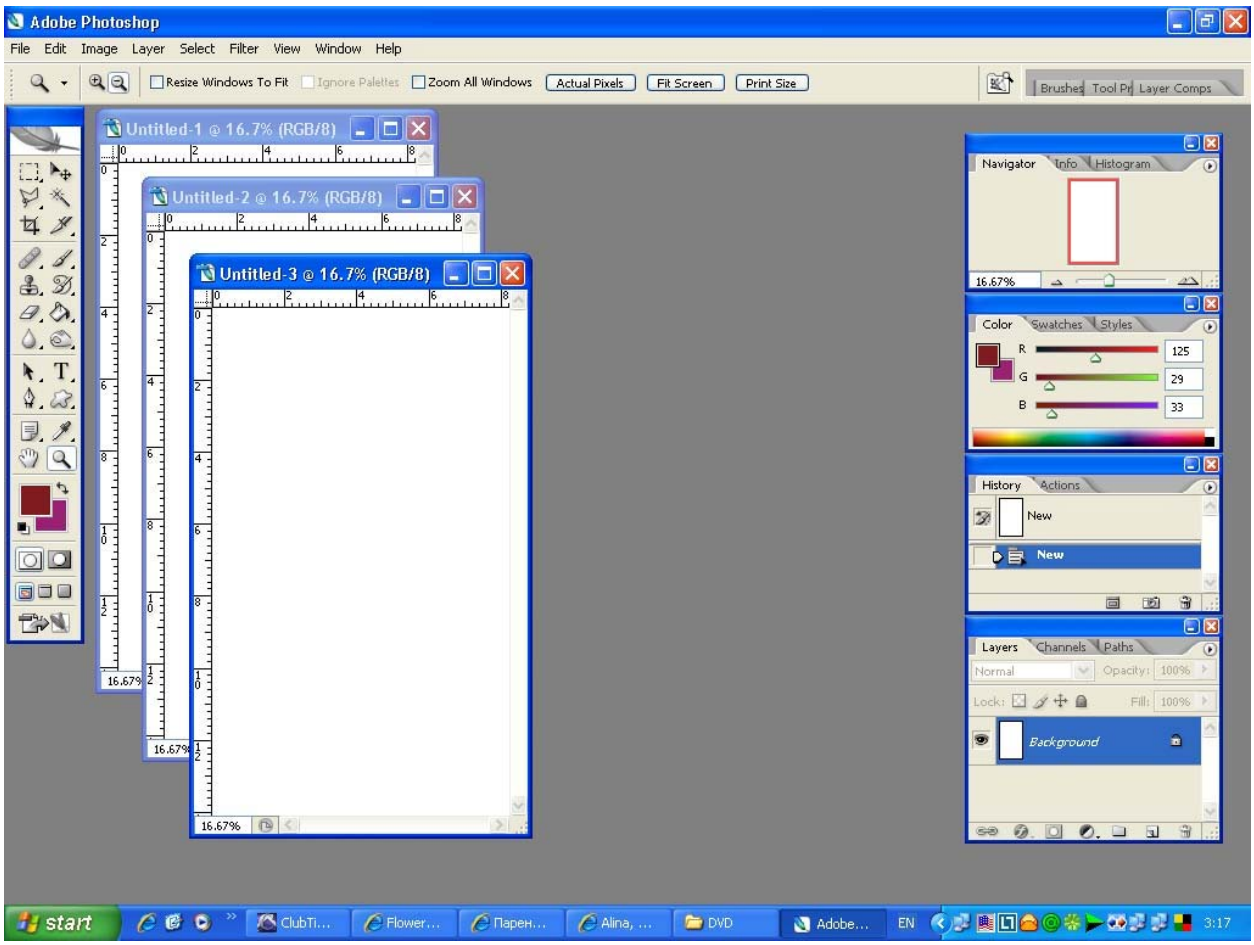
Նկար 5.1

SDI ինտերֆեյսը կազմված է հետևյալ էլեմենտներից.

- Գլխավոր մենյու
- Ղեկավարման էլեմենտների գոտի
- Տվյալների հետ աշխատելու գոտի
- Իրավիճակի նկարագրման գոտի

MDI ինտերֆեյս

MDI ինտերֆեյսի առանձնահատկությունը նրանում է, որ միանման ֆորմա, տարբեր պարունակությամբ, բացվում է մեկ ծնող պատուհանի շրջանակներում: Այդպիսի օրինակ կարող է հանդիսանալ **PhotoShop**-ը:



Նկար 5.2

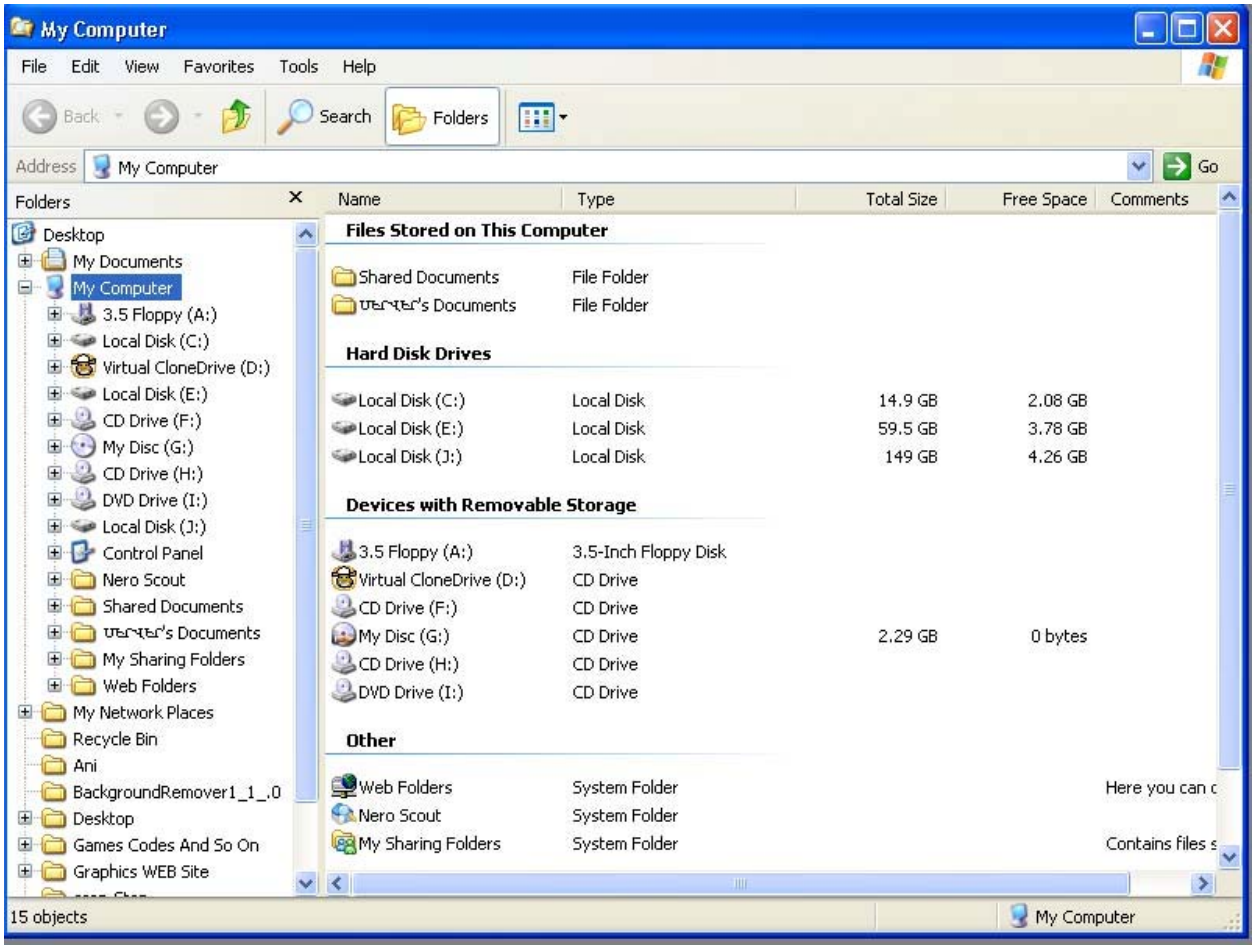
Նշենք, որ ծնող պատուհանը կարող է լինել միայն մի հատ, որը հանդիսանում է կոնտեյներ դուստր պատուհանների համար: Դա նշանակում է, որ, եթե ծնող պատուհանը մինիմիզացվում է, ապա դուստր պատուհանները ևս մինիմիզացվում են, ու եթե փակվում է ծնող պատուհանը, ապա նրա բոլոր դուստր պատուհանները ևս փակվում են: Իրենց հերթին դուստր պատուհանները կարող են գտնվել միայն ծնող պատուհանի ներսում:

MDI ինտերֆեյսը կազմված է հետևյալ բաղկացուցիչներից.

- Գլխավոր մենյու
- Ղեկավարման գործիքների գոտի
- Ծրագրի գլխավոր պատուհան
- Դուստր պատուհաններ
- Ինֆորմացիայի խմբագրման պատուհան
- Իրավիճակի նկարագրման գոտի

Դիտարկիչ տիպի ինտերֆեյս

Այս տիպի ինտերֆեյսը մշակվում է հիերարխիկ կառուցվածք ունեցող ստրուկտուրաների հետ աշխատելու համար: Օրինակ կարող է հանդիսանալ **Windows Explorer**-ի պատուհանը:



Նկար 5.3

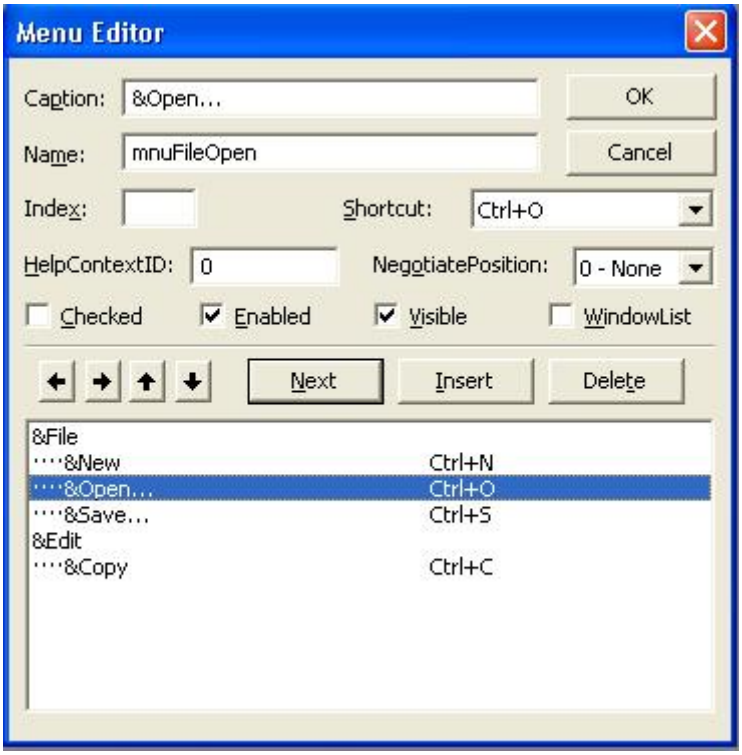
Իր էությամբ դիտարկիչ ինտերֆեյսը նման է **SDI** տիպի ինտերֆեյսին: Դիտարկիչ ինտերֆեյսի հիմնական բաղկացուցիչներն են.

- Գլխավոր մենյու
- Հավելվածի պատուհան, որը նախատեսված է տվյալների ղեկավարման համար
- Հիերարխիկ ստրուկտուրայի ցանկ
- Իրավիճակի նկարագրման գոտի

Բաժին 4. Մենյու

Ցանկացած ծրագիր ստեղծվում է, որպեսզի իրականացնի որևէ կոմպլեքս ֆունկցիաներ, որը ապահովում է դրված խնդրի լուծումը: Ծրագրի ֆունկցիոնալությունը ապահովելու համար կիրառվում են մենյուներ՝ գլխավոր մենյու և կոնտեքստային մենյու: Մենյու ստեղծելու ժամանակ հարկավոր է հետևել որոշակի ստանդարտների: Խորհուրդ է տրվում մենյուի անդամներին անվանել ստանդարտ ձևով: Որպես ստանդարտ կարելի է վերցնել **Windows**-ի մենյուն: Մենյուի անդամները կարող են ունենալ արագ գործարկման ստեղներ: Խորհուրդ է տրվում նույնպես պահպանել ստանդարտ ձևը:

Որպեսզի **Visual Basic**-ում ստեղծենք մենյու, օգտվում ենք **Menu Editor** պատուհանից (նկար 5.4):



Նկար 5.4

Այս պատուհանը կարելի է ակտիվացնել՝ **Tools** մենյուից ընտրելով **Menu Editor** հրամանը:

Menu Editor պատուհանը հիմնականում կազմված է հետևյալ մասերից.

- **Caption** – այստեղ գրվում է այն տեքստը, որը պետք է երևա մենյուում
- **Name** – մենյուի անունը, որին պետք է դիմել ծրագրավորման ժամանակ
- **Shortcut** – արագ գործարկման ստեղնը
- **Visible** – ապահովում է մենյուի տեսանելիությունը
- **Enabled** – ապահովում է մենյուի հասանելիությունը
- **Checked** – մենյուի վրա դնում է նշիչ

Մենյուն խմբագրվում է չորս սլաքների և **Next**, **Insert**, **Delete** ստեղների միջոցով: Տես նկար 5.5:

Նկար 5.5

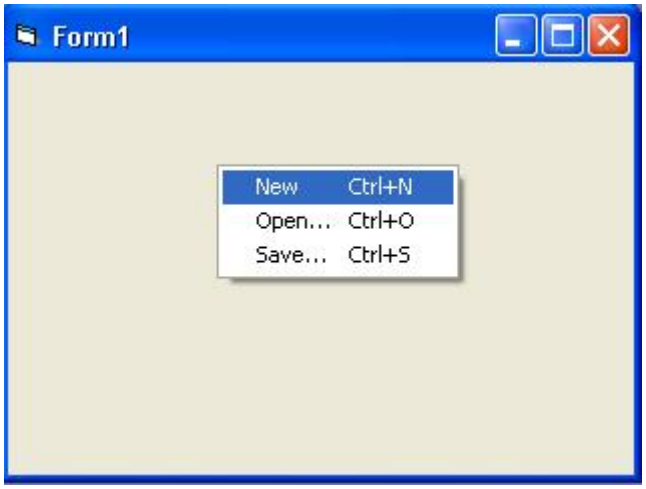
Բաժին 5. Կոնտեքստային մենյու

Կոնտեքստային մենյուն կցվում է որևէ օբյեկտի և հիմնականում հայտնվում է մկնիկի աջ հարվածի դեպքում: Կոնտեքստային մենյուն խմբագրվում է ինչպես սովորական մենյուն՝ **Visible** հատկությունը դրվելով **False**:

Եթե հարկավոր է ֆորմային կցել կոնտեքստային մենյու, ապա կոդը կլինի հետևյալը՝

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
   If Button = vbRightButton Then
      Me.PopupMenu mFile
   End If
End Sub
```

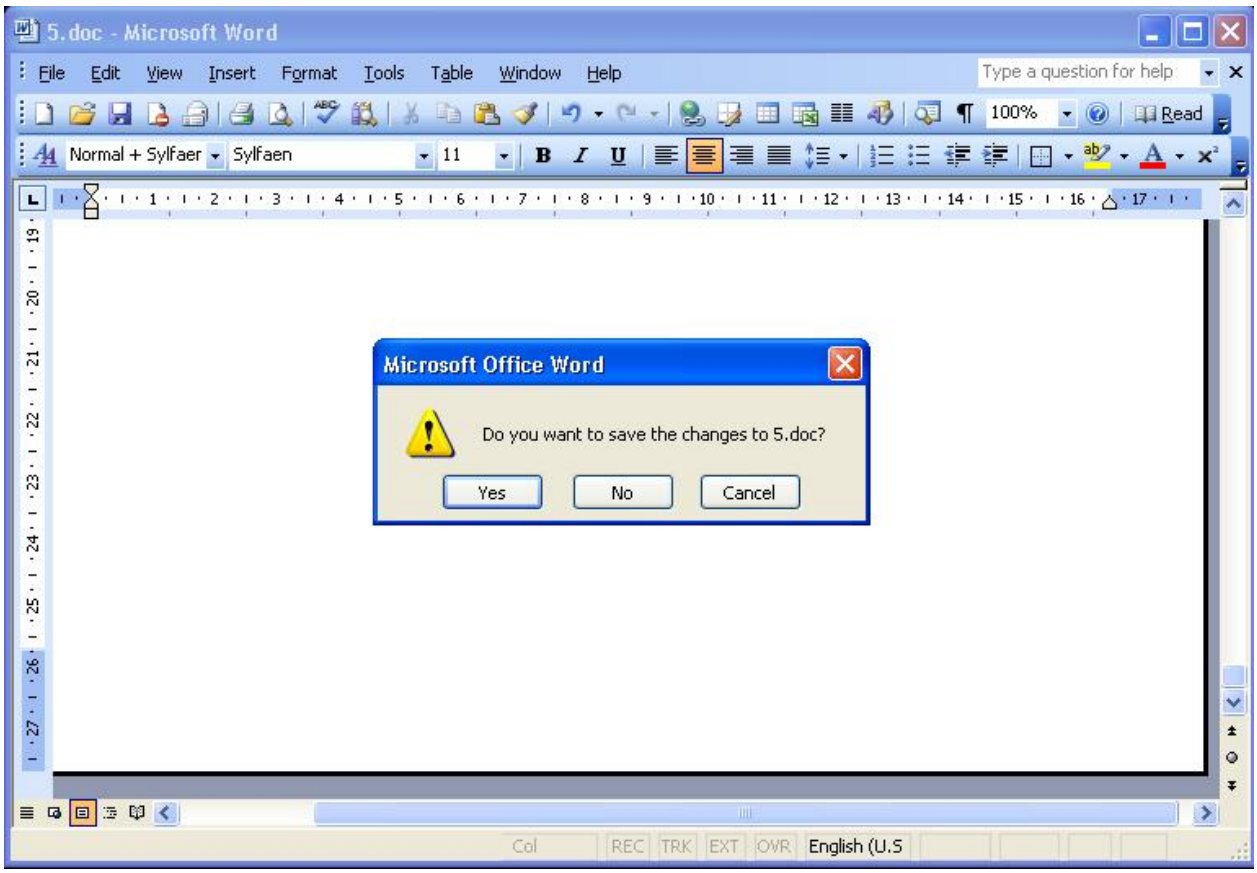
Ստացված արդյունքը պատկերված է նկար 5.6-ում:



Նկար 5.6

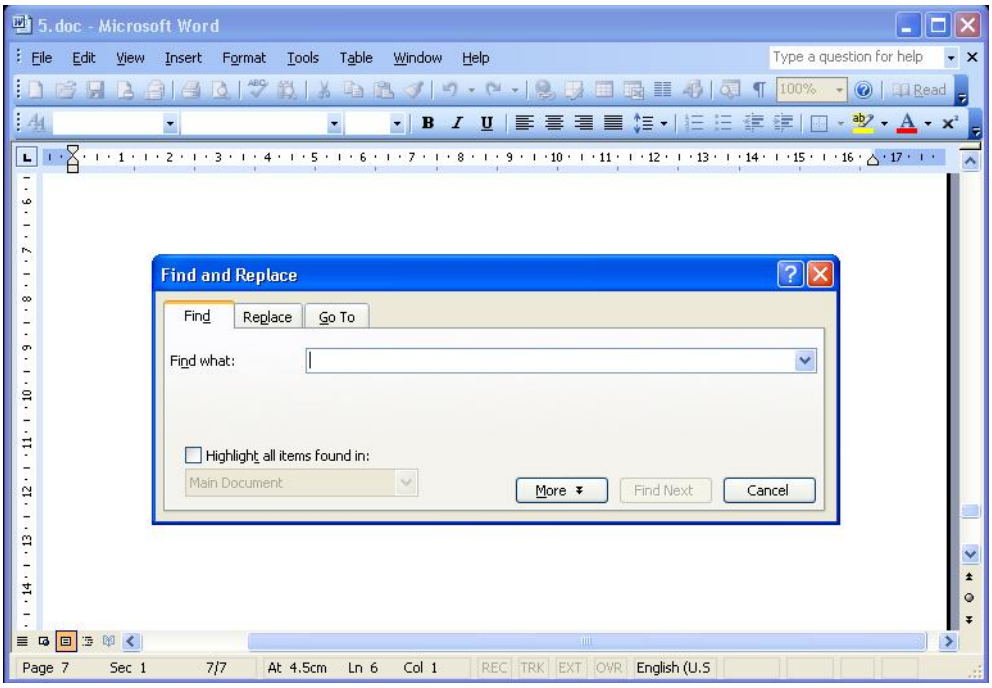
Բաժին 6. Երկխոսական պատուհաններ

Visual Basic 6.0-ում կան հատուկ տեսակի պատուհաններ՝ երկխոսական պատուհաններ: Դրանք լինում են երկու տեսակի՝ մոդալ և ոչ մոդալ: Մոդալի դեպքում հարկավոր է փակել այդ պատուհանը, որպեսզի մյուս պատուհանները դառնան հասնաելի: Օրինակ **MS Word**-ի պատուհանը, երբ նրա մեջ փոփոխություններ են արվել և առանց պահպանելու այդ փոփոխությունները փորձում եք պակել այն: Տես նկար 5.7:



Նկար 5.7

Որպես ոչ մոդալ կարող ենք դիտարկել **Find and Replace** պատուհանը նույնպես **MS Word**-ի մեջ: Տես նկար 8.8:



Նկար 5.8

Բաժին 7. Հաղորդագրությունների MSGBOX պատուհանը

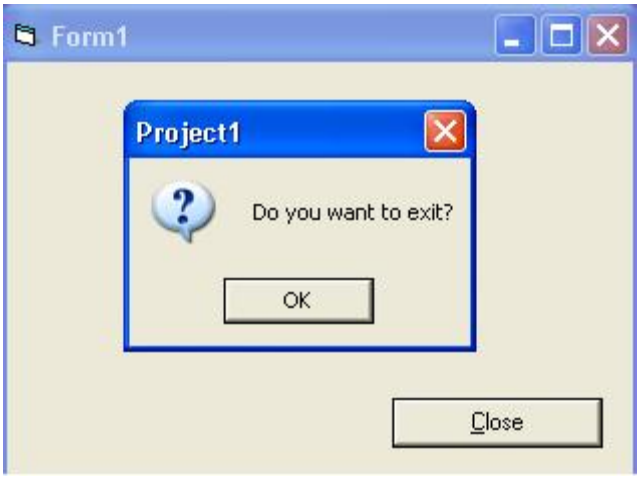
Հաղորդագրությունների պատուհանը կանչվում է msgbox հրամանի կամ msgbox() ֆունկցիայի միջոցով: Սինտակսիսը հետևյալն է.

```
Msgbox (Prompt [,buttons][,title][,helpfile,context])
```

Դիտարկեմք օրինակ, որը արտաբերում է հաղորդագրությունների պատուհանը:

```
MsgBox "Do you want to exit?", vbQuestion, App.Title
```

Տեսքը պատկերված է նկար 5.9-ում:



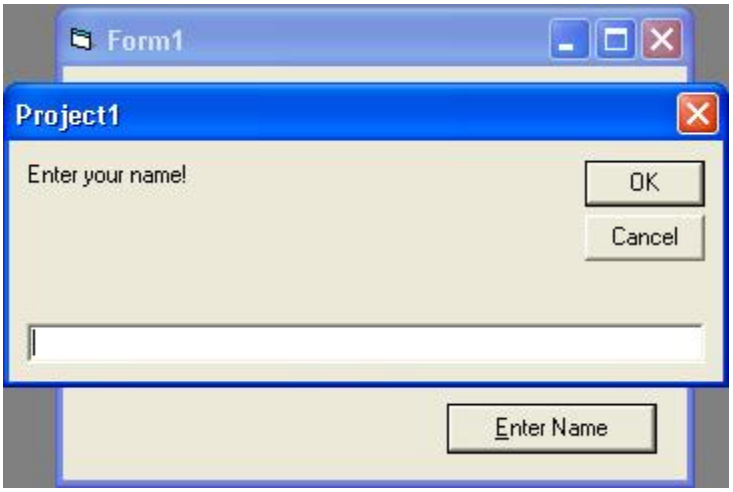
Նկար 5.9

Բաժին 8. Ինֆորմացիայի ներմուծման INPUTBOX պատուհանը

Հաճախակի հարկավոր է լինում, որպեսզի շահագործողը ծրագրին փոխանցի որոշակի ինֆորմացիա, որը պետք է մշակվի ծրագրի կողմից: Դրա համար հաճախ օգտագործում են **InputBox** պատուհանը: Սինտակսիսը հետևյալն է՝

```
InputBox (Prompt [,title][,default][,Xpos][,Ypos][,helphile,context])
```

Տեսքը պատկերված է նկար 5.10-ում:



Նկար 5.10

Գլուխ 6. Գրաֆիկայի օգտագործումը Visual Basic-ում

Բաժին 1. Գրաֆիկան հավելվածի մեջ

Visual Basic-ը ունի շատ մեծ հնարավորություններ գրաֆիկայի հետ աշխատելու համար: Գրաֆիկան հավելվածի մեջ օգտատգործվում է, որպեսզի հավելվածի ինտերֆեյսը լինի ավելի հաճելի և գեղեցիկ, սակայն հարկավոր է, որ այն չափը չանցնի: Ծրագրերի վաճառքի համար շատ մեծ նշանակություն ունի ծրագրի գրաֆիկական տեսքը: Գրաֆիկական միջոցներով **Visual Basic**-ում կարելի է կատարել հետևյալ գործողությունները՝

- Խմբավորել և առանձնացնել ինֆորմացիան ֆորմայի մեջ
- Ինֆորմացիան տրամադրել գրաֆիկական տեսքով
- Ինտերֆեյսի տեսքը լավացնել անիմացիայի միջոցով

Բաժին 2. Visual Basic-ում գրաֆիկայի հետ աշխատելու ղեկավարման էլեմենտները

Visual Basic-ում գրաֆիկայի հետ աշխատելու համար կան ղեկավարման էլեմենտներ: Դրանք են՝ **Line**, **Shape**, **Frame**:

Line-ը գիծ է, որի հաստությունը լռելյայն հավասար է մեկի: Սակայն, հատկություններից **BorderStyle**-ի արժեքը փոփոխելով, կարելի է փոփոխել նրա տեսքը: **BorderStyle**-ը կարող է ունենա հետևյալ արժեքները՝ **Trasparent, Solid, Dash, Dot, Dash-Dot, Dash-Dot-Dot, Inside Solid:**

Shape-ը հանդիսանում է գրաֆիկանան կոնտեյներ, որը խմբավորում է ղեկավարման գործիքները: **Shape** հատկությունը կարող է ընդունել հետևյալ արժեքները՝ **Rectangle, Square, Oval, Circle, Rounded Rectangle, Rounded Square**:

Frame-ը նույնպես հանդիսանում է կոնտեյներ: Ունի **Caption** հատկություն, այսինքն կարող ենք տալ տեքստ: Եթե **Visible** հատկությունը **False** է, ապա նրա մեջ գտնվող բոլոր օբյեկտներն ևս անհայտանում են: **Frame**-ի **BorderStyle** հատկությունը կարող է լինել **None** կամ **Fixed Single**:

Բաժին 3. Նկարների հետ աշխատելու համար նախատեսված գործիքները Visual Basic-ում

Գրաֆիկական պատկերներ տեղադրելու համար ֆորմայի մեջ օգտագործվում են **Image** և **Picture** օբյեկտները: Սակայն ոչ բոլոր ֆայլերի հետ են աշխատում այդ օբյեկտները: Աշխատում են միայն **bmp, dib, ico, cur, wmf, emf, gif, jpg, jpeg** ֆայլերի հետ: Որպեսզի օբյեկտի մեջ բեռնենք պատկեր, հարկավոր է գրել հետևյալ կոդը՝

```
object.Picture=LoadPicture(ֆայլի_հասցե)
```

Իսկ ջնջելու համար հետևյալ կոդը՝

```
object.Picture=LoadPicture(“”):
```

Բաժին 4. Գույների հետ աշխատելու ֆունկցիաներ

Գույների հետ աշխատելու համար ներդրված են **RGB()** և **QBColor()** ֆունկցիաները: **RGB()** ֆունկցիան գույնի համար վերադարձնում է **Long** տիպի թվային արժեք: Կազմի մեջ մտնում են երեք գույներ՝ կարմիր, կանաչ և կապույտ: Թվային արժեքը կարող է փոփոխվել 0-255: Օրինակ կարմիր գույն ստանալու համար հարկավոր է գրել՝ **RGB(255,0,0)**:

QBColor() ֆունկցիան կարող է ընդունել 16 թվային արժեքներ՝ 0-15: Օրինակ կարմիր գույն ստանալու համար գրվում է **QBColor(4)**:

Բաժին 5. Գրաֆիկական մեթոդներ

Visual Basic-ում գոյություն ունեն նաև գրաֆիկական մեթոդներ: Դրանք են՝ **Circle, Cls, Line, PainPicture, Point, Print, Pset**:

Circle մեթդի սինտակսիսը հետևյալն է.

```
Object.Circle [step] (x,y), radius, [color, start, end aspect]
```

Cls օբյեկտի շնորհիվ կարելի է մաքրել ֆորման կամ **Picture** օբյեկտը տեքստից կամ գրաֆիկայից, որը ստեղծվել է ծրագրային կոդով: Սինտակսիսը հետևյալն է՝

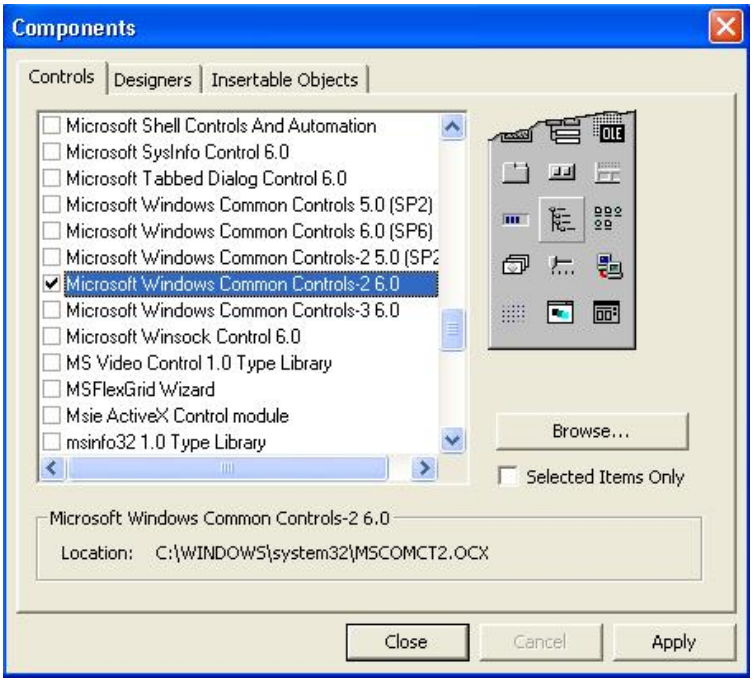
```
Object.cls
```

Line մեթոդը մախատեսված է գծեր գծելու համար: Սինտակսիսը հետևյալն է՝

```
Object.Line [step] [x1,y1][step] – (x2,y2),(color),[B][F]:
```

Բաժին 6. Անիմացիան Visual Basic-ում

Animation գործիքը հնարավորություն է տալիս **VB**-ում կիրառել անիմացիա: Այն հնարավորություն է տալիս ցուցադրել հատուկ տեսակի **avi** ֆայլեր: Դրա համար հարկավոր է ծրագրին կցել **Microsoft Windows Common Controls 2.6.0** գրադարանը: Դա արվում է հետևյալ կերպ՝ **Project** մենյուից ընտրում եք **Components** հրամանը: Բացված պատուհանից ընտրում եք **Microsoft Windows Common Controls 2.6.0** և սեղմում **Close** ստեղնը: Տես նկար 6.1:



Նկար 6.1

Դրանից հետո ֆորմայի մեջ ավելացվում է **Animation** օբյեկտը: **Animation** օբյեկտը ունի հետևյալ հիմնական մեթոդները՝ **Open, Play, Stop, Close: Play**-ը կարող է ունենալ հետևյալ արգումենտները՝ **Repeat, Start, Stop:** Դիտարկենք ծրագրային կոդ, որտեղ օգտագործվում է **Animation** օբյեկտը:

```
Private Sub btnPlay_Click()
   Animation1.Play
End Sub
Private Sub btnStop_Click()
   Animation1.Stop
End Sub
Private Sub Form_Load()
   Animation1.Open (File_Name)
End Sub
Private Sub Form_Unload(Cancel As Integer)
   Animation1.Close
End Sub
```

Գլուխ 7. Աշխատանք ինֆորմացիայի հետ

Բաժին 1. Screen օբյեկտը

Visual Basic-ում **Screen** օբյեկտը իրենից ներկայացնում է **Windows**-ի ամբողջ էկրանը: Քանի որ **Windows**-ը ունի միայն մեկ էկրան, **Screen** օբյեկտը սովորաբար չի հայտարարվում: Սակայն, եթե անհրաժեշտ է հայտարարել, հարկավոր է գրել հետևյալ կերպ՝

```
Dim scr As Screen
Set scr = Screen
```

Screen օբյեկտի հատկություններն են՝

- ActiveControl – ակտիվ ղեկավարման էլեմենտը
- ActiveForm – ակտիվ ֆորման
- FontCount – հասանելի ֆոնտերի քանակը
- Fonts() – վերադարձնում է հասանելի ֆոնտի անունը
- Height – էկրանի բարձրությունը
- MouseIcon – մկնիկի նշիչը
- MousePointer – մկնիկի նշիչի տեսակը
- TwipsPerPixelX – պիկսելում թվիպների քանակը
- TwipsPerPixelX – պիկսելում թվիպների քանակը
- Width – էկրանի լայնությունը

Բաժին 2. TypeOf

TypeOf բանալիային բառի շնորհիվ կարող ենք գործողություններ կատարել ակտիվ օբյեկտների հետ: Դիտարկենք օրինակ՝

```
If TypeOf Screen.ActiveControl Is TextBox Then
   Screen.ActiveControl.SelText = ""
ElseIf TypeOf Screen.ActiveControl Is PictureBox Then
   Screen.ActiveControl.Picture = LoadPicture("")
End If
```

Բաժին 3. Իրադարձություններ կապված ստեղնաշարի հետ

Ստեղնաշարի հետ աշխատելու համար կան հետևյալ իրադարձությունները՝ **KeyDown**, **KeyPress** և **KeyUp**:

KeyPress պրոցեդուրայի համար որպես արգումենտ հանդես է գալիս **KeyAscii** տեսակի փոփոխականը, որը պարունակում է ստեղնի **ANSI** կոդը: **ASCII** և **ANSI** տարբերությունը էական է միայն այն սիմվոլների համար, որի արշեքը մեծ է 127-ից: Դիտարկենք օրինակ՝

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
   MsgBox KeyAscii
End Sub
```

KeyAscii փոփոխականի արժեքը կարելի է ոչ միայն կարդալ, այլև տեղադրել՝

```
KeyAscii = Asc(UCase(Chr(KeyAscii))):
```

Ստեղնաշարի հետ աշխատելիս օգտագործվում է նաև **KeyUp** և **KeyDown** իրադարձությունները: Այստեղ օգտագործվում են **KeyCode** և **Shift** փոփոխականները: **KeyCode**-ը պարունակում է ստեղնի կոդը, **Shift**-ը՝ **CTRL**, **SHIFT**, **ALT** ստեղների կոդը: **Shift**-ի պարամետրերը հետևյալներն են՝

vbShiftMask – SHIFT

vbCtrlMask – CTRL

vbAltMask – ALT

Դիտարկենք օրինակ՝

```
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
   If KeyCode = vbKeyF4 Then
      MsgBox "Pressed F4"
   ElseIf KeyCode = vbKeyF4 And Shift = vbShiftMask Then
      MsgBox "Pressed F4 + Shift"
   End If
End Sub
```

Visual Basic-ը հնարավորություն է տալիս գեներացնել ստեղնի սեղմում: Սինտակսիսը հետևյալն է՝

```
SendKeys Ctrl [,Wait] :
```

Գլուխ 8. Աշխատանք ֆայլերի հետ

Բաժին 1. Ֆայլերի հետ աշխատելու ձևերը Visual Basic-ում

Հավելվածի նախագծման ժամանակ հաճախ անհրաժեշտ է լինում աշխատել ֆայլերի հետ: Դրա համար **Visual Basic**-ը ունի ֆունկցիաներ, որոնք հնարավորություն են տալիս աշխատել ֆայլերի հետ: Ընդունված է տարբերել ֆայլերի հետևյալ ձևերը՝

- Ֆայլեր ըստ հաջորդական դիմման: Սրանք տեքստային ֆայլեր են և իրենցից ներկայացնում են սիմվոլների հաջորդականություն:
- Կամայական դիմման ֆայլեր: Սրանք ստրուկտուրավորված ֆայլեր են, որոնք ինֆորմացիան պահում են գրանցումների տեսքով: Օրինակ տվյալների բազաների ֆայլերը:
- Երկուական դիմման ֆայլեր: Սրա օրինակ կարող են լինեն ծրագրային ֆայլերը:

Բաժին 2. Ֆայլի բացումը

Որպեսզի ֆայլերի հետ աշխատել, հարկավոր է այն նախապես բացել: Դրա համար օգտագործվում է **Open** օպերատորը: Սինտակսիսը հետևյալն է՝

```
Open ֆայլի_հասցե For mod [access][lock] As [#] FileNumber [Len=recLenght]
```

Որտեղ՝

- Mode – ֆայլին դիմման ռեժիմն է: Այն կարող է լինել Append, Binary, Input, Output կամ Random
- Access – ֆայլին դիմման ձևն է: Կարող է լինել Read, Write, Read/Write
- Lock – Այլ պրոցեսների համար ֆայլի հասանելիության ապահովում: Կարող է լինել Shared, Lock Read, Lock Write, Lock Read Write:
- FileNumber – Ֆայլի իդենտիֆիկատոր: Կարող է լինել 1-ից 511 ներառյալ:
- recLenght – Կարդալու կամ գրելու համար անհրաժեշտ չափ: Կարող է լինել մինչև 32.767 բայթ:

Բաժին 3. Ֆայլի փակումը

Ֆայլը փակելու համար օգտագործվում է **Close** օպերատորը: Սինտակսիսը հետևյալն է՝ `Close [FileNumber]:`

Ֆայլը **Open** օպերատորով բացելուց հետո պետք է պարտադիր փակվի: Եթե **FileNumber** պարամետրը նշված չէ, ապա **Close** օպերատորը փակում է բոլոր այն ֆայլերը, որոնք բացվել են **Open** օպերատորի միջոցով:

Գլուխ 9. Սխալների մշակում

Բաժին 1. Սխալների տեսակները

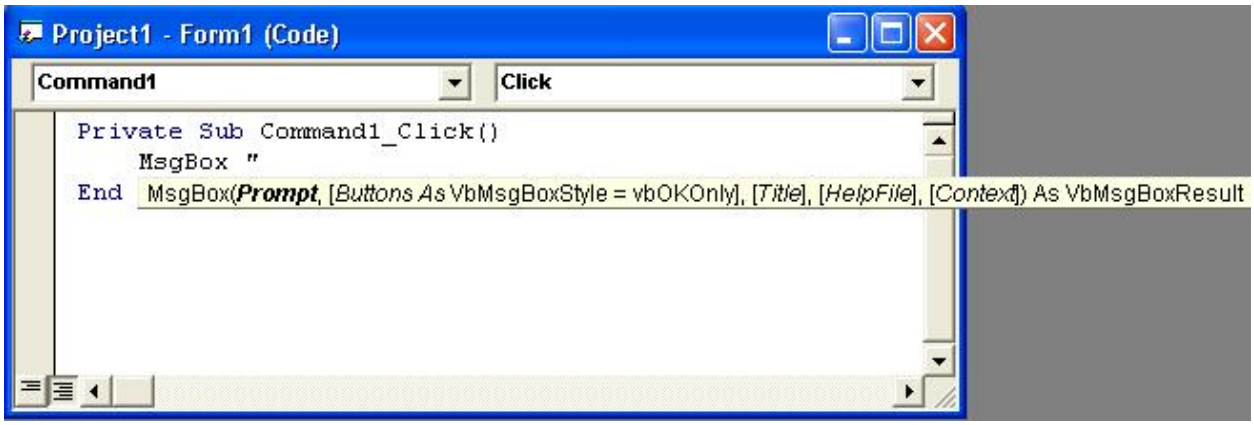
Ծրագրի աշխատանքի ժամանակ կարող են առաջանալ չորս տեսակի սխալներ՝

- Սինտակսիսային սխալ
- Ծրագրի կառուցվածքային սխալ
- Սխալներ, որոնք առաջանում են ծրագրի կատարման ժամանկ
- Տրամաբանական սխալներ

Սինտակսիսային սխալների առաջացման արդյունք կարող են լինել բանալիային բառերի սխալ գրումը կամ օպերատորների անթույլատրելի կոմբինացիաների օգտագործումը: **Visual Basic**-ը սինտակսիսային սխալները ճանաչում է անիմջապես, հենց որ կուրսորը հեռացվում է ընթացիկ տողից: Սխալի հայտնաբերման դեպքում **Visual Basic**-ը տալիս է համապատասխան հաղորդագրություն սխալի վերաբերյալ:

Բաժին 2. Կոնտեքստային հուշում

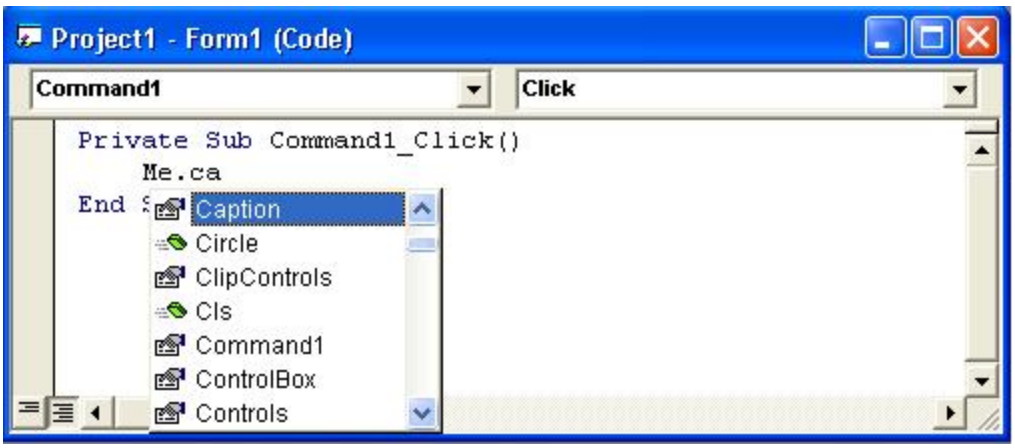
Visual Basic-ում ներդրված են միջոցներ, որոնք հնարավորություն են տալիս ոչ միայն գտնել սխալները, այլև օգնում են, որպեսզի սխալ թույլ չտալ: Այդպիսի միջոց է հանդիսանում կոնտեքստային հուշումը: Ծրագրային կոդը գրելու ժամանակ այն ավտոմատ կերպով ցուցադրում է օպերատորների սինտաքսիսը: Տես նկար 9.1:



Նկար 9.1

Բաժին 3. Էլեմենտների ցանկի ավտոմատ ցուցադրում

Որպեսզի հնարավորինս նվազեցվի սխալների քանակը օբյեկտների անունները, հատկությունները և մեթոդները գրելու ժամանակ, **Visual Basic**-ում ավտոմատ ձևով ցուցադրվում է այդ ցանկերը: Տես նկար 9.2:



Նկար 9.2

Բաժին 4. Սխալների մշակում

Visual Basic-ում սխալների հետ աշխատելու հարամ կա **On Error** օպերատորը (սա աշխատում է միայն ծրագրի գործարկման ընթացքում): Սինտակսիսը հետևյալն է՝

```
On Error {GoTo Label | Resume Next | GoTo 0}
```

On Error օպերատորը ինքը սխալների մշակում չի կատարում, այլ կառավարումը փոխանցում է այն պրոցեդուրային, որը նախատեսված է սխալները մշակելու համար: Սխալների մշակումը պետք է իրականացվի անմիջապես սխալի հայտնաբերումից հետո: Սկզբից հարկավոր է բացահայտել սխալի տեսակը: Դրա համար **Visual Basic**-ում ներդրված է **Err** օբյեկտը, որի **Number** հատկությունը վերադարձնում է սխալի կոդը: Եթե ծրագրային կոդի մեջ հարկավոր է անտեսել սխալը, ապա կարելի է գրել հետևյալ կերպ՝ **On Error Resume Next**: Դիտարկենք սխալի մշակման մի փոքր պրոցեդուրա.

```
Private Sub Command1_Click()
   If Err.Number <> 0 Then Err.Clear
   On Error GoTo err_
      Err.Raise (9)
      Exit Sub
err_:
      MsgBox Err.Number
      MsgBox Err.Description
      Err.Clear
      Resume Next
End Sub
```

Հավելված Ա

WIN32 API ծրագրավորում

Visual Basic 6.0-ն հնարավորություն է տալիս աշխատել **API** ֆունկցիաների հետ, որոնք գտնվում են **dll** գրադարաններում: Նախքան ֆունկիցիային դիմելը, հարկավոր է, որ այն հայտարարվի: Դա արվում է **Declare** օպերատորի միջոցով: Եթե **dll** ֆունկցիան ֆորմայի մոդուլի մեջ է, ապա այն պետք է հայտարարվի որպես **Private**, իսկ ծրագրային մոդուլում՝ որպես **Public**: Որպիսի կոնֆլիկտներ տեղի չունենան ֆունկցիայի անունները հայտարարելու ժամանակ օգտագործվում է **Alias** ռեզերվացված բառը:

Ֆունկցիայի և պրոցեդուրայի հայտարարման օրինակներ՝

[Private/Public] Declare Sub պրոցեդուրայի_անուն Lib "DLL_ֆայլի_անունը" [Alias "անուն"] [([պարամետրերի_ցանկ])]

[Private/Public] Declare Function ֆունկցիայի_անուն Lib "DLL_ֆայլի_անունը" [Alias "անուն"] [([պարամետրերի_ցանկ }]] As տիպի_տեսակը

օրինակ

```
Declare Function BitBlt Lib "gdi32" _
(ByVal hDestDC As Long, ByVal x As Long, ByVal y As Long, _
ByVal nWidth As Long, ByVal nHeight As Long, _
ByVal hSrcDC As Long, ByVal xSrc As Long, ByVal ySrc As Long, _
ByVal dwRop As Long) As Long
```

Որոշ օգտակար API ֆունկցիաների նկարագրություն

BitBlt – պատճենում է պատկերը

```
Declare Function BitBlt Lib "gdi32" Alias "BitBlt" _
(ByVal hDestDC As Long, ByVal x As Long, ByVal y As Long, _
ByVal nWidth As Long, ByVal nHeight As Long, ByVal hSrcDC As Long, _
ByVal xSrc As Long, ByVal ySrc As Long, ByVal dwRop As Long) As Long

Private Const SRCCOPY = &HCC0020

Private Sub Command1_Click()
BitBlt Picture2.hDC, 0, 0, 100, 100, Picture1.hDC, 0, 0, SRCCOPY
End Sub
```

FindWindow – գտնում է պատուհանը

```
Declare Function FindWindow Lib "user32" Alias "FindWindowA" _
(ByVal lpClassName As String, ByVal lpWindowName As String) As Long

Private Sub Command1_Click()
Dim hand As Long
hand = FindWindow(vbNullString, "Calculator")
End Sub
```

GetCursorPos – վերադարձնում է մկնիկի դիրքը

```
Declare Function GetCursorPos Lib "user32" Alias "GetCursorPos" _
(lpPoint As POINTAPI) As Long

Type POINTAPI
x As Long
y As Long
End Type

Private Sub Command1_Click()
Dim a As POINTAPI
GetCursorPos a
Form1.Caption = a.x
End Sub
```

GetDriveType – վերադարձնում է սկավառակի տեսակը

```
Declare Function GetDriveType Lib "kernel32" Alias "GetDriveTypeA" _
(ByVal nDrive As String) As Long

Private Sub Command1_Click()
Form1.Caption = GetDriveType("C:\")
End Sub
```

SetCursorPos – փոփոխում է մկնիկի դիրքը

```
Declare Function SetCursorPos Lib "user32" Alias "SetCursorPos" _
(ByVal x As Long, ByVal y As Long) As Long

Private Sub Command1_Click()
SetCursorPos 100, 100
End Sub
```

ShellExecute – գործարկում է ծրագիր

```
Declare Function ShellExecute Lib "shell32.dll" Alias _
"ShellExecuteA" (ByVal hwnd As Long, ByVal lpOperation As String, _
ByVal lpFile As String, ByVal lpParameters As String, _
ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long

Const SW_MAXIMIZE = 3

Private Sub Command1_Click()
ShellExecute Form1.hwnd, "open", "C:\...\1.bmp", 0, 0, SW_MAXIMIZE
End Sub
```

Հավելված Բ

Ռեեստրի հետ աշխատանքի համառոտ նկարագրություն

Կարդալու ֆունկցիաները

Սիմվոլային ինֆորմացիա կարդալու համար օգտագործվում է **GetRegString** ֆունկցիան:

օրինակ

```
Dim a As String
a = GetRegString(HKEY_LOCAL_MACHINE, _
"Software\Microsoft\DirectX", "Version")
MsgBox = a
```

DWORD տիպի ինֆորմացիա կարդալու համար օգտագործվում է **GetRegDWord** ֆունկցիան:

օրինակ

```
Dim a As Long
a = GetRegDWord(HKEY_LOCAL_MACHINE, _
"Software\Microsoft\Internet Explorer\AboutURLs", "Home")
MsgBox = a
```

Բանալիների ցանկի պարունակությունը ստանալու համար օգտագործվում է **GetRegKeys** ֆունկցիան:

օրինակ

```
Dim a As Variant
Dim c As Long
a = GetRegKeys(HKEY_LOCAL_MACHINE, "Software")
For c = 0 To UBound(a)
List1.AddItem a(c)
Next c
```

Բանալու պարամետրերը ստանալու համար օգտագործվում է **GetRegKeyValues** ֆունկցիան:

օրինակ

```
Dim a As Variant
Dim c As Long
a = GetRegKeyValues(HKEY_LOCAL_MACHINE, _
"Software\Microsoft\Windows\CurrentVersion\Run")
For c = 1 To UBound(a)
List1.AddItem a(c, 0)
List1.AddItem a(c, 1)
List1.AddItem ""
Next c
```

Գրանցելու ֆունկցիաները

Սիմվոլային պարամետր գրանցելու համար օգտագործվում է **SetRegString** ֆունկցիան:

օրինակ

```
SetRegString HKEY_LOCAL_MACHINE, _
"Software\XYZ", "MyProgram", "Options"
```

DWORD տիպի ինֆորմացիա գրանցելու համար օգտագործվում է **SetRegDWord** ֆունկցիան:

օրինակ

```
SetRegDWord HKEY_LOCAL_MACHINE, _
"Software\XYZ", "MyProgram", "444"
```

Ջնջելու ֆունկցիաները

Բանալին ջնջելու համար օգտագործվում է **DeleteRegKey** ֆունկցիան:

օրինակ

```
DeleteRegKey HKEY_LOCAL_MACHINE, "Software", "XYZ", True
```

Պարամետր ջնջելու համար օգտագործվում է **DeleteRegValue** ֆունկցիան:

օրինակ

```
DeleteRegValue HKEY_LOCAL_MACHINE,_
"Software\Microsoft\Internet Explorer\AboutURLs", "mozilla", False
```

Հավելված Գ

Օգտակար խորհուրդներ

1. Եթե ձեզ հարկավոր է փոխել **Boolen** տիպի արտահայտությունը հակառակ, ապա գրեք այսպես՝

```
blnVal = Not(blnVal)
```

2. Որպեսզի արտահայտության մեջ փոփոխեք մի քանի սիմվոլներ, ապա օգտվեք **mid$** ֆունկցիայից՝

```
Dim a As String
a = "Иванов"
Mid$(a, 1, 4) = "Петр" 'կստացվի Петров
```

3. Chr(13)+Chr(10) փոխարեն կարող եք օգտագործել vbNewLine

```
MsgBox "առաջի տող" & vbNewLine & "երկրորդ տող"
```

4. Եթե օգտագործվող ֆայլը գտնվում է ձեր գործարկվող ֆայլի հետ նույն թղթապանակի մեջ,ապա կարող եք օգտվել **App.Path** հատկությունից

```
Picture1.Picture = LoadPicture (App.Path & "\1.bmp")
```

5. Եթե ուզում եք իմանաք ձեր ծրագիրը գործարկված է մեկից ավելի, ապա **Form_Load** իրադարձության մեջ գրեք հետևյալը՝

```
If App.PrevInstance Then
        MsgBox "Ծրագիրը արդեն իսկ գործարկված է"
        End
End If
```

6. Որպիսի ստանաք բոլոր հասանելի էկրանային ֆոնտերը գրեք հետևյալ կոդը՝

```
For c = 1 To Screen.FontCount
    List1.AddItem Screen.Fonts(c)
Next c
```

7. Եթե ուզում եք իմանաք ֆայլը գոյություն ունի թե ոչ, կարող եք օգտագործել հետևյալ ֆունկցիան՝

```
Function FileExist(fileName As String) As Boolean
On Error Resume Next
FileExist = Dir$(fileName) <> ""
If Err.Number <> 0 Then FileExist = False
On Error GoTo 0
End Function
```

Հավելված Դ

Visual Basic 6.0 - ի ներդրված ֆունկցիաների համառոտ նկարագրությունը

Abs(*Number*)- վերադարձնում է թվի բացարձակ արժեքը (մոդուլը)

օրինակ.

```
Dim MyNumber
MyNumber = Abs(50.3)  'վերադարձնում է 50.3.
MyNumber = Abs(-50.3) 'վերադարձնում է 50.3.
```

Array(*Arglist*) - վերադարձնում է **variant** տիպի փոփոխական, որը պարունակում է զանգված

օրինակ.

```
Dim MyWeek, MyDay
MyWeek = Array("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun")
MyDay = MyWeek(2) 'այժմ MyDay պարունակում է "Tue".
MyDay = MyWeek(4) 'այժմ MyDay պարունակում է "Thu".
```

Asc(*String*) – վերադարձնում է պարամետրի առաջին սիմվոլի կոդը

օրինակ

```
Dim MyNumber
MyNumber = Asc("A") 'վերադարձնում է 65.
MyNumber = Asc("a") 'վերադարձնում է 97.
MyNumber = Asc("Apple") 'վերադարձնում է 65.
```

Atn(*Number*)- վերադարձնում է թվի արկտանգեսը

օրինակ

```
Dim pi
pi = 4 * Atn(1) 'հաշվում է pi-ի արժեքը
```

Cos(*number*) – հաշվում է թվի կոսինուսը

օրինակ

```
Dim MyAngle, MySecant
MyAngle = 1.3 'անկյունը տեղադրում ենք ռադիանով
MySecant = 1 / Cos(MyAngle) 'հաշվում ենք սեկանսը
```

Chr(*charcode*) – վերադարձնում է կոդին համապատասխան սիմվոլը, հանդիսանում է **Asc** հակադարձ ֆունկցիան

օրինակ

```
Dim MyChar
MyChar = Chr(65) 'վերադարձնում է A.
MyChar = Chr(97) 'վերադարձնում է a.
MyChar = Chr(62) 'վերադարձնում է >.
MyChar = Chr(37) 'վերադարձնում է %.
```

Choose(*index,choice-1[, choice-2, ... [, choice-n]]*) – արգումենտի ցանկից ընտրում է արտահայտություն

Command – վերադարձնում է հրամանային տողի արտահայտությունը

օրինակ

```
Function GetCommandLine(Optional MaxArgs)
Dim C, CmdLine, CmdLnLen, InArg, I, NumArgs
If IsMissing(MaxArgs) Then MaxArgs = 10
ReDim ArgArray(MaxArgs)
NumArgs = 0: InArg = False
CmdLine = Command()
CmdLnLen = Len(CmdLine)
For I = 1 To CmdLnLen
C = Mid(CmdLine, I, 1)
If (C <> " " And C <> vbTab) Then
If Not InArg Then
```

```
If NumArgs = MaxArgs Then Exit For
NumArgs = NumArgs + 1
InArg = True
End If
ArgArray(NumArgs) = ArgArray(NumArgs) & C
Else
InArg = False
End If
Next I
ReDim Preserve ArgArray(NumArgs)
GetCommandLine = ArgArray()
End Function
```

CurDir [*(drive)*] – վերադարձնում է **String** տիպի արժեք, որի մեջ պարունակում է ընթացիկ թղթապանակի հասցեն

CVErr (*error number*) – վերադարձնում է շահագործողի կողմից որոշված սխալի կոդը

Date – վերադարձնում է ընթացիկ ամսաթիվը

DateDiff (*interval, date1, date2[, firstdayofweek[, firstweekofyear]]*) – վերադարձնում է երկու ամսաթվերի մեջ եղած տարբերությունը

օրինակ

```
Dim TheDate As Date
Dim Msg
TheDate = InputBox("ներմուծել ամսաթիվ")
Msg = "մինչ այսօր եղած օրերի թիվը՝ " & _
DateDiff("d", Now, TheDate)
MsgBox Msg
```

DatePart (*interval, date[,firstdayofweek[, firstweekofyear]]*) – վերադարձնում է ամսաթվի մասը միայն, օրինակ միայն ամիսը կամ օրը

օրինակ

```
Dim TheDate As Date
Dim Msg
TheDate = Date
Msg = "ամսվա համարը՝ " & DatePart("m", TheDate)
MsgBox Msg
```

DateSerial (*year, month, day*) – վերադարձնում է ամսաթիվը առանձին մասերով

օրինակ

```
Dim MyDate
MyDate = DateSerial (2003, 2, 12)
'վերադարձնում է February 12, 2003.
```

Day (*date*) – վերադարձնում է օրը

Dir [*(pathname[, attributes])*] – վերադարձնում է ֆայլի կամ թղթապանակի անունը, որը համապատասխանում է փնտրման չափանիշներին

DoEvents ( ) – օպերացիոն համակարգին հնարավորություն է տալիս արձագանքել իրադարձություններին

Environ (*{envstring | number}*) – վերադարձնում է օպերացիոն համակարգի շրջակայքի ասոցացված հասցեներից վորևէ մեկը

օրինակ

```
Dim c As String
For a = 1 To 20
c = Environ(a)
If c <> "" Then
Text1.Text = Text1.Text & c & vbCrLf
Else
Exit For
End If
Next a
```

EOF (*filenumber*) – վերադարձնում է **true**, եթե հասել է ֆայլի վերջին

օրինակ

```
Dim InputData
Open "MYFILE" For Input As #1 'բացում ենք ֆայլը կարդալու համար
Do While Not EOF(1) 'ստուգում ենք վերջն է թե ոչ
Line Input #1, InputData 'կարդում ենք մեկ տող
Debug.Print InputData 'արտապատկերում ենք տողը
Loop
Close #1 ‘փակում ենք ֆայլը
```

Error [*(errornumber)*] – վերադարձնում է սխալին նկարագրող հաղորդագրություն

օրինակ

```
Dim ErrorNumber
For ErrorNumber = 61 To 64
   Debug.Print Error(ErrorNumber)
Next ErrorNumber
```

Exp(*number*) – վերադարձնում է թվի էքսպոնենցիալը

օրինակ

```
msgbox= Exp(1)
```

FileAttr(*filenumber, returntype*) – վերադարձնում է **open** ֆունկցիայի միջոցով բացված ֆայլի ձևը

FileDateTime(*pathname*) – վերադարձնում է ֆայլի վերջին փոփոխության ամսաթիվը

FileLen(*pathname*) – վերադարձնում է ֆայլի ծավալը

օրինակ

```
Dim MySize
MySize = FileLen("FileName") 'վերադարձնում է ֆայլի ծավալը բայթերով
```

Fix(*number*) – վերադարձնում է թվի ամբողջական մասը

օրինակ

```
MyNumber = Fix(99.2) 'վերադարձնում է 99.
MyNumber = Fix(-99.8) 'վերադարձնում է -99.
MyNumber = Fix(-99.2) 'վերադարձնում է -99.
```

Format(*expression[, format[, firstdayofweek[, firstweekofyear]]]*) – վերադարձնում է ֆորմատավորված տեքստ

FreeFile[*(rangenumber)*] – վերադարձնում է ազատ ֆայլի համարը

GetAllSettings(*appname, section*) – վերադարձնում է **SaveSetting**-ի միջոցով պահպանված պարամետրերը

GetAttr(*pathname*) – վերադարձնում է ֆայլի կամ թղթապանակի ատրիբուտները

GetSetting(*appname, section, key[, default]*) – ռեեստրից վերադարձնում է պահպանված պարամետրերը

Hex (*number*) – վերադարձնում է թվի 16-ական կոդը

օրինակ

```
Dim MyHex
MyHex = Hex(5) 'վերադարձնում է 5.
MyHex = Hex(10) 'վերադարձնում է A.
MyHex = Hex(459) 'վերադարձնում է 1CB.
```

Hour (*time*) – վերադարձնում է ժամը

օրինակ

```
Dim MyTime, MyHour
MyTime = #4:35:17 PM# '
MyHour = Hour(MyTime) 'պարունակում է 16
```

IIf (*expr, truepart, falsepart*) – վերադարձնում է արգումենտի պարամետրերից երկուսից մեկը

օրինակ

```
Function CheckIt (TestMe As Integer)
    CheckIt = IIf(TestMe > 1000, "Large", "Small")
End Function
```

Input (*number, [#]filenumber*) – օգտագործվում է ֆայլում գրանցելու համար

InputBox (*prompt[, title] [, default] [, xpos] [, ypos] [, helpfile, context]*) – բացում է երկխոսական պատուհան ինֆորմացիա մուտքագրելու համար
InStr (*[start,]string1, string2[, compare]*) – վերադարձնում է 2-րդ պարամետրի արտահայտության դիրքը 1-ին պարամետրի մեջ
Int (*number*) – եթե թիվը փոքր է 0-ից ապա կլորացվում է մինչև մոտակա ամբողջ թիվը, հակառակ դեպքում վերադարձնում է ամբողջ մասը

օրինակ

```
Dim MyNumber
MyNumber = Int(99.8) 'վերադարձնում է 99.
MyNumber = Int(-99.8) 'վերադարձնում է -100.
MyNumber = Int(-99.2) 'վերադարձնում է -100
```

IsNumeric(*expression*) – եթե արտահայտությունը թիվ է, ապա վերադարձնում է **true**
LBound(*arrayname[, dimension]*) – վերադարձնում է զանգվածի ներքին սահմանի ինդեքսը
LCase(*string*) – տեքստը փոխում է փոքրատառերի

օրինակ

```
Dim UpperCase, LowerCase
Uppercase = "Hello World 1234"
Lowercase = Lcase(UpperCase) 'վերադարձնում է "hello world 1234"
```

Left(*string, length*) – վերադարձնում է տեքստի ձախ մասի սիմվոլները՝ նշված թվին համապատասխան

օրինակ

```
Dim AnyString, MyStr
AnyString = "Hello World"
MyStr = Left(AnyString, 1) 'վերադարձնում է "H".
MyStr = Left(AnyString, 20) 'վերադարձնում է "Hello World".
```

Len(*string | varname*) – վերադարձնում է արտահայտության երկարությունը
LoadPicture(*[stringexpression]*) – նկարը բեռնում է **PictureBox**-ի մեջ
LOF(*filenumber*) – վերադարձնում է **open** օպերատորով բացված ֆայլի ծավալը

օրինակ

```
Dim FileLength
Open "TESTFILE" For Input As #1
FileLength = LOF(1)
Close #1
```

Log(*number*) – հաշվում է թվի բնական լոգարիթմը
LTrim(*string*) – վերադարձնում է արտահայտությունը՝ ջնջելով ձախ կողմի պրոբելները
Mid(*string, start[, length]*) – վերադարձնում է արտահայտություն, որը վերցվում է պարամետրերին համապատասխան

օրինակ

```
Dim MyString, FirstWord, LastWord, MidWords
MyString = "Mid Function Demo"
FirstWord = Mid(MyString, 1, 3) 'վերադարձնում է "Mid".
LastWord = Mid(MyString, 14, 4) 'վերադարձնում է "Demo".
MidWords = Mid(MyString, 5) 'վերադարձնում է "Function Demo".
```

Minute(*time*) – վերադարձնում է **data** տեսակի արտահայտություն, որը պարունակում է 0-59 թվերից որևէ մեկը՝ կախված րոպեից
Month(*date*) - վերադարձնում է **data** տեսակի արտահայտություն՝ կախված ամսաթվից վերադարձնում է այդ արժեքը
MsgBox(*prompt[, buttons] [, title] [, helpfile, context]*) – արտաբերում է հաղորդագրությունների պատուհանը
Now – վերադարձնում է համակարգային ամսաթիվը և ժամանակը
Oct(*number*) – վերադարձնում է թվի 8-ական կոդը
RGB(*red, green, blue*) – վերադարձնում է **Long**, որը պարունակում է գույնի կոդը

Right(*string, length*) – արտահայտությունից վերադարձնում է արգումենտում նշված թվի չափով սիմվոլներ, աջ մասից
Rnd[*(number)*] – վերադարձնում է 0-1 միջակայքից պատահական թիվ
RTrim(*string*) – ջնջում է արտահայտության աջ մասի պրոբելները
Shell(*pathname[,windowstyle]*) – գործարկում է **windows**-ի հավելված
Sin(*number*) – վերադարձնում է թվի սինուսը
Space(*number*) – վերադարձնում է դատարկ տարածություն՝ կախված արգումենտից
Spc(*n*) – ավելացնում է պրոբել
Sqr(*number*) – վերադարձնում է թվի արմատը
Str(*number*) – թիվը ձևափոխում է սիմվոլի
StrComp(*string1, string2[, compare]*) – համեմատում է երկու արտահայտությունները
StrConv(*string, conversion*) – ձևափոխում է արտահայտության տիպը
Tab[*(n)]* – ստեղծում է դատարկ տարածություն (տաբուլյացիա)
Tan(*number*) – վերադարձնում է թվի տանգեսը
Time – վերադարձնում է ընթացիկ համակարգային ժամանակը
Timer – վերադարձնում է կեսգիշերից հետո անցած ժամանակը վայրիկյաններով
Trim(*string*) –երկու կողմից ջնջում է արտահայտության պրոբելները
TypeName(*varname*) – վերադարձնում է պարամետրի վերաբերյալ ինֆորմացիա
UBound(*arrayname[, dimension]*) – վերադարձնում է զանգվածի վերին ինդեքսը
UCase(*string*) – արտահայտությունը դարձնում է մեծատառ

օրինակ

```
Dim LowerCase, UpperCase
LowerCase = "Hello World 1234"
UpperCase = UCase(LowerCase) 'վերադարձնում է "HELLO WORLD 1234".
```

Val(*string*) – սիմվոլը փոխում է թվի
VarType(*varname*) – վերադարձնում է պարամետրի տեսակը
Year(*date*) – վերադարձնում է տարվա ամսաթիվը

Visual Basic 6.0 - ի օպերատորների համառոտ նկարագրությունը

AppActivate *title[, wait]* – ակտիվացնում է հավելվածի պատուհանը

օրիմակ

```
Dim MyAppID, ReturnValue
AppActivate "Microsoft Word"
MyAppID = Shell("C:\...\WINWORD.EXE", 1) 'գործարկում ենք Microsoft Word-ը
AppActivate MyAppID  'ակտիվացնում ենք Microsoft Word-ը
```

ChDir *path* – փոփոխում է ընթացիկ թղթապանակը
ChDrive *drive* – փոփոխում է ընթացիկ սկավառակը
Close [*number*] – փակում է **open** օպերատորով բացած ֆայլը

օրինակ

```
Open "TESTFILE" For Output As #3
Print #3, "Hello World"
Close #3
```

Date = *date* – փոփոխում է համակարգային ամսաթիվը
DeleteSetting *appname, section[, key]* – ջնջում է ռեստրում գրանցված ինֆորմացիան

օրինակ

```
SaveSetting "MyApp", "Startup", "Top", Form1.Top
SaveSetting "MyApp", "Startup", "Left", Form1.Left
DeleteSetting "MyApp", "Startup"
```

Erase *arraylist* – ջնջում է մասիվի պարունակությունը և դատարկում է հիշողությունը
Error *errornumber* – ակտիվացնում է սխալ՝ պարամետրի արժեքին համապատասխան
FileCopy *source, destination* – պատճենահանում է ֆայլը
օրինակ
```
FileCopy "C:\Windows\Win.ini", "C:\Backups\Win.bak"
```
Kill *pathname* – ջնջում է ֆայլը
օրինակ
```
Kill "TestFile"
Kill "*.TXT"
```
Load *object* – հիշողություն է բեռնում ֆորմա կամ կոմպոնենտ
MkDir *path* – ստեղծում է թղթապանակ
օրինակ
```
MkDir "C:\MYDIR"
```
Open *pathname For mode [Access access] [lock] As [#]filenumber [Len=reclength]* – բացում է ֆայլը
RaiseEvent *eventname [(argumentlist)]* – գործարկում է իրադարձությունը
Randomize [number] – գործարկում է պատահական թվերի գեներատորը
օրինակ
```
Dim MyValue
Randomize
MyValue = Int((6 * Rnd) + 1)
```
Reset – փակում է **open** օպերատորով բացված բոլոր ֆայլերը
RmDir *path* – ջնջում է թղթապանակը
օրինակ
```
RmDir "C:\Temp"
```
SendKeys *string[, wait]* – գեներացնում է ստեղնաշարի սեղմում
օրինակ
```
Dim ReturnValue, I
ReturnValue = Shell("CALC.EXE", 1)
AppActivate ReturnValue
For I = 1 To 100
SendKeys I & "{+}", True
Next I
SendKeys "=", True
SendKeys "%{F4}", True
```
SetAttr *pathname, attributes* – փոփոխում է ֆայլի ատրիբուտները
օրինակ
```
SetAttr "TESTFILE", vbHidden
SetAttr "TESTFILE", vbHidden + vbReadOnly
```
Unload *object* – բեռնաթափում է օբյեկտին

Ուշադրություն.

Սույն ինքնուսույցը նախատեսված է միայն ծանոթացնելու համար **Visual Basic 6.0** ծրագրավորման լեզվի հետ: Այն ունի միայն ուսուցողական բնույթ: Հնարավոր է, որ ինքնուսույցը պարունակի անճշտություններ կամ թերություններ: Այդ տեսակ սխալներ հայտնաբերելու դեպքում խնդրում ենք տեղյակ պահել հեղինակային կազմին՝ գրելով հետևյալ էլ-հասցեին. lambada-center@mail.ru: Ցանկացած անձ, ով կօգտագործի ինքնուսույցի մեջ ներկայացված ինֆորմացիան հակաօրինական նպատակներով, ինքն է պատասխանատվություն կրում դրա համար (ինքնուսույցի հեղինակային կազմը ոչ մի պատասխանատվություն չի կրում):

VISUAL BASIC 6.0 պատրաստի կոդի օրինակներ կարող եք ձեռք բերել www.vb.am կայքից: