

Կ. Ա. ԽԱՉԱՏՐՅԱՆ

ԾՐԱԳՐԱՎՈՐՄԱՆ
ՏԵԽՆՈԼՈԳԻԱՆԵՐ

ՈՒՍՈՒՄՆԱԿԱՆ ԶԵՂՈՆԱՐԿ



«ՏԱՏԵՄԱԳԵՏ»
ԵՐԵՎԱՆ 2004

ԴՏԴ 681.3/. 5(07)
ԳՄԴ 32.973g7
Խ 282

Դրատարակության է Երաշխավորել ԵՊՏԻ
գիտական խորհուրդը

Սասմ. խմբագիր՝ տեխ. գ. դ., պրոֆեսոր Թ. Ա. Նալչաջյան
Գրախոսներ՝ տեխ. գ. թ., դոցենտ Ս. Ս. Ավետիսյան
Մնտ. գ. թ., դոցենտ Վ. Ա. Սարգսյան

Խաչատրյան Կ. Ա.

Խ282 Ծրագրավորման տեխնոլոգիաներ: Ուսումնական
ծեռնարկ.- Եր.: «Տնտեսագետ», 2004.- 208 էջ:

Ներկայացվող աշխատանքը նվիրված է ինֆորմացիոն տեխնոլո-
գիաների զարգացման արդի փուլին համապատասխան ծրագրավորման
նոր մեթոդների օրիենտակողմնորոշված ծրագրավորման սկզբունքների
ներկայացման, ծրագրային արտադրանքի ստեղծման ժամանակակից
լեզուների դասավանդմանը:

Բացի գործնական խնդիրների ծրագրավորման բաժնից, այն ընդ-
գրկում է կուրսային և լաբորատոր աշխատանքների կատարման խնդիր-
ների շունչարան.

Զերկարկը նախատեսված է տնտեսագիտական բուհերի ուսանողների,
ասպիրանտների, ծրագրավորման տեխնոլոգիաները գործնականում
կիրառող մասնագետների համար, ինչպես նաև կարող է ուղեցույց լինել
ծրագրավորման մասնագետների համար:

ՀՊՏՀ գրադարան



000000636

Խ 2404000000 2004
719(01)2004

13749(93)

ռան. 2

ISBN 99930-77-79-8

ԳՄԴ 32.973g7

Բովանդակություն

Նախաբան	5
Գլուխ 1. Ալգորիթմ և ծրագրավորում հասկացությունները	6
1.1. Ալգորիթմ հասկացությունը և նկարագրման եղանակները	6
1.2. Ալգորիթմների տեսակները և հաշվողական պրոցեսները	9
1.3. Ծրագրային արտադրանքի նախագծման մեթոդները	12
1.4. Ծրագրային արտադրանքի ստեղծման փուլերը	15
1.5. Ծրագրային արտադրանքի կառուցվածքը	16
1.6. Օրյեկտակողմնորոշված ծրագրավորում	17
Գլուխ 2. Visual Basic	21
2.1. «Visual Basic» ծրագրավորման լեզուն	21
2.2. Դաշտավածների (լրացումներ) ստեղծումը «Visual Basic» -ով	22
2.3. Օրյեկտ հասկացությունը և նրա բնութագրերը	23
2.4. Visual Basic 6.0 –ի աշխատանքային միջավայրը	25
2.5. Ղեկավարման լլեմենտների գույնի	26
2.6. Նախագծի պատուհան	32
2.7. Կողի պատուհան	32
2.8. Օրագրավորումը «Visual Basic»-ում	33
2.9. “Պատահարադիկավարվող ծրագրավորում	37
2.10. “Պրոցեդուրաներ և ֆունկցիաներ”	38
2.11. VB-ում հաճախ օգտագործվող ֆունկցիաները	39
2.12. «Visual Basic» - ում ներդրված ֆինանսական ֆունկցիաները	42
2.13. Դրամանների կատարման կարգի ղեկավարման օպերատորները	52
2.14. Ստևանաշարի և «մկնիկի» պատահարները	54
2.15. Շահագործողական դասերի ստեղծումը	58
2.16. Ֆայլերի հետ աշխատանքը	60
2.17. Տպող սարքի (Printer) հետ աշխատանքը VB-ի միջավայրից	65

2.18.	Օբյեկտակողմնորոշված ծրագրավորումը VBA-ում	67
2.19.	Զարգացման ինտեգրացված միջավայր	73
2.20.	Սենյուների ստեղծումը Visual Basic-ում	76
2.21.	Աշխատանքը տվյալների բազաների հետ	87
2.22.	Visual Basic-ը և Windows API- ին	114
2.23.	Հավելվածի տեղեկատու համակարգի ստեղծումը	117
2.24.	Հավելվածի կարգավորման պարամետրերը	117
	Գործնական և լաբորատոր աշխատանքներ	119
	Լաբորատոր աշխատանքներ	119
	Գործնական և լաբորատոր աշխատանքների համար առաջարկվող խնդիրներ	127
Գլուխ 3.	C + + որպես ծրագրավորման գործիքային միջոց	136
3.1.	C լեզվի բազային հասկացությունները	136
3.2.	Ծրագրավորման C++ լեզուն	148
Գլուխ 4.	Windows համակարգում աշխատող Microsoft Visual C++ -ի գործիքային միջոցները	168
4.1.	Windows օպերացիոն համակարգում ծրագրավորումը	168
4.2.	Windows-ի միջավայրում մշակվող հավելվածների ինտեգրման մեթոդները	169
4.3.	Ծրագրերի մշակումը Visual C++ -ում	181
4.4.	Նախագծերի ստեղծման վարպետները	183
4.5.	Visual C++-ում երկխոսության պատուհանը և դեկավարման պարզագույն էլեմենտների օգտագործումը	189
	Լաբորատոր և գործնական աշխատանքներ	192
	Գործնական և լաբորատոր աշխատանքների համար առաջարկվող խնդիրներ	202
	Կուրսային աշխատանքի համար առաջարկվող խնդիրներ	205
	Օգտագործված գրականություն	207

Նախաբան

Չուկայական հարաբերությունների զարգացման ներկա փուլում գիտատեխնիկական առաջընթացն արտահայտվում է արտադրության գործիքների որակական փոփոխություններով, տեխնոլոգիական և կառավարման ֆունկցիաների կատարելագործումով, որին եապես նպաստում է ժամանակակից հաշվողական տեխնիկայի և ծրագրավորման միջոցների, նաբենատիկական մեթոդների, նոր ինֆորմացիոն տեխնոլոգիաների կիրառությունը:

Ժամանակակից ինֆորմացիոն տեխնոլոգիաների կիրառությունը օժանդակում է արտադրության էքստենսիվ աճը ինտենսիվով փոխարինման գործընթացին, որն իրականացվում է ցանկացած ոլորտում ինֆորմացիոն տեխնոլոգիաների ներդրման, ինֆորմացիոն հոսքերի արդյունավետ օգտագործման, որոշումների ընդունման օպտիմալ տարբերակների մշակմամբ:

Դաշվողական տեխնիկայի զարգացմանը համապատասխան առաջացել են ծրագրավորման տարբեր մեթոդներ: Յուրաքանչյուր փուլում ստեղծվել են նոր մեթոդներ, որոնք օգնել են ծրագրավորողներին հաղթահարել ծրագրերի աճող բարդությունները: Ժամանակակից պահանջներին բավարարող ծրագրեր կազմելու համար անհրաժեշտ եղավ ծրագրավորման նոր մոտեցում: Արդյունքում մշակվեցին օբյեկտակողմնորոշված ծրագրավորման սկզբունքները, որոնք թույլ են տալիս օպտիմալ ձևով կազմակերպել ծրագրերը պրոբլեմը տրոհելով իրար հետ կապված խնդիրների: Յուրաքանչյուր պրոբլեմ դառնում է ինքնուրույն օբյեկտ, ամբողջ գործողությունը պարզեցվում է և ինարավոր է դառնում գործ ունենալ ավելի մեծ ծավալով ծրագրերի հետ:

Գիրքը ծանոթացնում է այսօրվա գործող ինֆորմացիոն տեխնոլոգիաների ստեղծման նվազագույն գործիքներին:

Առանձնակի ուշալրություն է դարձված ծրագրային արտադրանքի նախագծման մեթոդների, ստեղծման փուլերի, կառուցվածքի, աշխատանքային միջավայրի և ծրագրավորման ժամանակակից լեզուների հիմնական տարրերին և հնարքներին:

Գործնական և մեթոդական առումով ծեռնարկը արժեքավոր է գործնական աշխատանքների կատարման համար անհրաժեշտ բազմաթիվ ծրագրերի առկայությամբ, ինչպես նաև կուրսային և լաբորատոր աշխատանքների համար առաջարկվող խնդիրներով:

Գլուխ 1

ԱԼԳՈՐԻԹՄ և ԾՐԱԳՐԱՎՈՐՈՒՄ ԻԱՍԿԱԳՈՒԹՅՈՒՆՆԵՐԸ

1.1. ԱԼԳՈՐԻԹՄ ԻԱՍԿԱԳՈՒԹՅՈՒՆԸ և ԱԿԱՐԱԳՐՄԱՆ Եղանակները

Իրագործման նպատակով համակարգիչ ներմուծված հրամանների հաջորդականությունը կոչվում է ծրագիր: Խսկ երբ խոսվում է որևէ նպատակ հետապնդող քայլերի հաջորդականության մասին, ապա օգտագործվում է ալգորիթմ տերմինը: «Ալգորիթմ» բառը առաջացել է լատիներեն «*algorithmi*» բառից, որը IX դարի նշանավոր մաթեմատիկոս Ալ-Խորեզմի արաբական անվան գրառումն է: Ալգորիթմ հասկացությունը ինֆորմատիկայում նույնքան հիմնային է, որքան կետը, գիծը՝ երկրաչափության մեջ: Որպեսզի ալգորիթմը դառնա իրագործելի կոնկրետ ծրագիր, անհրաժեշտ է դրա մեջ մտնող գործողությունները վերածել համակարգչի համար հասկանալի իրամանների համակարգի:

Ալգորիթմը տվյալ իրագործողի համար թույլատրելի այն գործողությունների կարգավորված հաջորդականությունն է, որը վերջավոր քայլերի ընթացքում հանգեցնում է կոնկրետ արդյունքի, իսկ ծրագիրը տվյալ մեքենայի լեզվով ալգորիթմի գրառումն է:

ԱԼԳՈՐԻԹՄԻ ՀԻՄՆԱԿԱՆ ԻԱՏԿՈՒԹՅՈՒՆՆԵՐԸ

1. **Ալգորիթմի որոշակիություն:** Ալգորիթմը պետք է բաղկացած լինի իրագործելի քայլերից: Դրա յուրաքանչյուր գործողություն պետք է հասկանալի լինի և միարժեցրող մեկնաբանվի օգտագործողի կողմից: Միևնույն նախնական տվյալներով կազմված նույն ալգորիթմը պետք է աշխատի ճիշտ միևնույն կերպ և հանգի նույն արդյունքին: Ալգորիթմը պետք է կառուցված լինի այնպես, որ բացառի ցանկացած կամայականություն:

2. **Ալգորիթմի մասսայականություն:** Ալգորիթմը պետք է պիտանի լինի նույն դասի բազմաթիվ խնդիրների լուծման համար՝ նախնական տվյալների թույլատրելի ցանկացած արժեքների դեպքում:

3. Ալգորիթմի արդյունավետություն: Վերջավոր թվով պարզ գործողությունների կատարումից հետո ալգորիթմը պետք է հանգի որոշակի արդյունքի:

4. Ալգորիթմի դիսկրետություն: Ալգորիթմը պետք է բաժանված լինի իրար հաջորդող առանձին պարզ գործողությունների (քայլերի):

Գոյություն ունեն ալգորիթմի ներկայացման տարրեր եղանակներ: Գործնականում առավել լայն կիրառում են ստացել երկուսը:

1. Ալգորիթմի բառաբանածևային նկարագրություն: Սա խոսակցական լեզվի և բանածևրի օգնությամբ գրված ալգորիթմի նկարագրությունն է:

2. Ալգորիթմի գրաֆիկական ներկայացում: Գրաֆիկական ներկայացումը բառաբանածևային ներկայացման համեմատ ավելի սեղմ է և դիտողական: Այս դեպքում ալգորիթմը ներկայացվում է հատուկ բլոկների միջոցով, որոնցից յուրաքանչյուրը ներկայացնում է խնդրի լուծման որոշակի գործողություն: Այդպիսի գրաֆիկական ներկայացումը անվանում են ալգորիթմի սխեմա կամ բլոկսխեմա:

Այսուսակ 1-ում բերված են առավել հաճախ կիրառվող բլոկների անունները, դրանց համար ընտրված երկրաչափական պատկերները և իրականացվող գործողությունները: Յուրաքանչյուր բլոկի մեջ գրվում է կատարվող գործողության մասին, ամեն բլոկ կարող է ունենալ իր հերթական համարը, որը դրվում է բլոկի եզրագծի վերին ծախս ընդհատված անկյունում: Բլոկները իրար միացվում են հաշվողական պրոցեսի ընթացքը ցույց տվող կապի գծերով, որոնց ուղղությունը սլաքով պարտադիր պետք է ցույց տրվի այն դեպքերում, երբ կատարման ընթացքը ուղղված է աջից ձախ կամ ներքևից վերև: Եթե կատարման ընթացքն ուղղված է աջից ձախից աջ կամ վերից վար, սլաքները կարելի է չդնել:

Բլոկի անվանումը	Գրաֆիկական պատկերը	Կատարվող ֆունկցիան
Պրոցես		Հաշվարկների կատարում և վերագրման գործողություն
Կանխորոշված պրոցես		Նախապես ստեղծված և առանձին նկարագրված այգորիթմի օգտագործում
Պայման		Պայմանի ստուգում և հաշվողական պրոցեսի այլընտրանքային շարունակման հնարավորություն
Մոդիֆիկացիա		Ցիկլային պրոցեսի կազմակերպում
Մուտք-Ելք		Տվյալների ներմուծում և արտածում
Սկիզբ		Ալգորիթմի սկիզբ
Ավարտ		Ալգորիթմի ավարտ
Միացուցիչ		Դոսքի ընդհատված գծերի միջև կապի ցուցիչ

1.2. Ալգորիթմների տեսակները և հաշվողական պրոցեսները

Ցանկացած ալգորիթմ կարելի է ներկայացնել հետևյալ երեք կառուցվածքների միջոցով՝ գծային, ճյուղավորված և ցիկլային: Այն ալգորիթմները, որոնք իրականացվում են այդ կառուցվածքների միջոցով, համապատասխանաբար անվանում են գծային, ճյուղավորված և ցիկլային ալգորիթմներ կամ պրոցեսներ:

Գծային ալգորիթմներ: Գծային են կոչվում այն ալգորիթմները, որոնց մեջ գործողությունները կատարվում են միայն մեկ անգամ ըստ գրված հաջորդականության: Սովորաբար գծային ալգորիթմ-ների մեջ գերակշռում են պրոցեսի գործողությունները: Պրոցեսի բլոկը նախատեսված է արտահայտության հաշվման և արդյունքի պահպանման համար:

Ճյուղավորված ալգորիթմներ: ճյուղավորված է կոչվում այն ալգորիթմը, որտեղ, ստուգվող պայմանից ելնելով, հաշվման պրոցեսը շարունակվում է հնարավոր տարբեր ուղիներից որևէ մեկով:

Հնարավոր ուղիները կոչվում են ալգորիթմի ճյուղեր:

Բլոկ-սխեմայի մեջ պայմանի ստուգման նպատակով կիրառվում է շեղանկյան տեսք ունեցող պայմանի բլոկը, որն ունի երկու ելք «այո» և «ոչ»: Եթե բլոկում նշված պայմանը ճիշտ է, ապա գործողությունների հետագա ընթացքը շարունակվում է «այո» ճյուղով, հակառակ դեպքում «ոչ» ճյուղով:

Ցիկլային ալգորիթմներ: Դաճախ խնդիրների լուծման ալգորիթմներ կազմելիս հանդիպում են գործողությունների որոշակի խմբեր, որոնք անհրաժեշտ է լինում կրկնել բազմաթիվ անգամ: Այդպիսի դեպքերում կիրառում են ալգորիթմների հատուկ կառուցվածքներ, որոնք կոչվում են ցիկլային: Կրկնվող գործողությունների խումբը կոչվում է ցիկլի մարմին:

Ըստ իրագործման տեխնիկայի՝ կարելի է առանձնացնել չորս հիմնական ցիկլային կառուցվածքներ.

- ա) պարամետրերով,
- բ) նախապայմանով,
- գ) հետպայմանով,
- դ) ներդրված:

Պարամետրով ցիկլային կառուցվածք: Պարամետրով ցիկլային կառուցվածքը իրականացնում է մեկ կամ մի քանի գործողությունների ցիկլային կրկնությունը՝ նախապես հայտնի քանակությամբ: Պարամետրերով ցիկլային կառուցվածքում պետք է հայտնի լինեն ցիկլի պարամետրի սկզբնական և վերջնական արժեքները: Սկզբում ցիկլի պարամետրին վերագրվում է իր սկզբնական արժեքը, որի դեպքում առաջին անգամ կատարվում է ցիկլի մարմինը: Այնուհետև ցիկլի պարամետրի արժեքը փոխվում է նշված քայլով: Այս գործողությունը կատարվում է այնքան ժամանակ, քանի դեռ ցիկլի պարամետրը մեծ չէ իր վերջնական արժեքից: Պարամետրերով ցիկլային ալգորիթմի այն տեսակը, որի դեպքում պարամետրը սկզբնական արժեքից աճելով հասնում է վերջնական արժեքին, անվանում են աճող պարամետրով ցիկլային ալգորիթմ, իսկ եթե ցիկլի սկզբնական պարամետրը նվազելով հասնում է վերջնական արժեքին, անվանում են նվազող պարամետրով ալգորիթմ: Պարամետրով ցիկլային ալգորիթմների կիրառման ժամանակ հնարավոր է ցիկլի վաղաժամկետ ընդհատում: Այն կարելի է իրականացնել պայմանի ստուգումով:

Նախապայմանով ցիկլային կառուցվածք: Նախապայմանով ցիկլային ալգորիթմը կազմակերպում է մեկ կամ մի քանի գործողությունների ցիկլային կրկնությունը ինչպես նախապես հայտնի, այնպես էլ անհայտ քանակությամբ: Նետպայմանով ցիկլային կառուցվածքներում կրկնումները շարունակվում են այնքան ժամանակ, քանի դեռ գրված պայմանը չի կատարվում (կեղծ է):

Ներդրված ցիկլերով ալգորիթմներ: Ներդրված ցիկլային ալգորիթմը իրականացնում է ցիկլի մարմնի մեջ եղած գործողությունների կրկնությունը ինչպես նախապես հայտնի, այնպես էլ անհայտ քանակությամբ: Նետպայմանով ցիկլային կառուցվածքներում կրկնումները շարունակվում են այնքան ժամանակ, քանի դեռ գրված պայմանը չի կատարվում (կեղծ է):

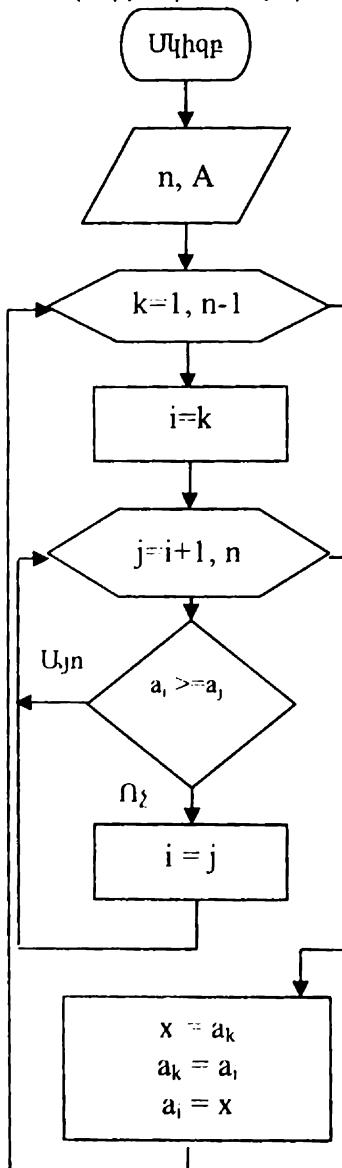
Ներդրված ցիկլերով ալգորիթմներ: Ներդրված ցիկլերով ալգորիթմները կառուցվածքը կազմակերպում են դեպքեր, երբ ցիկլի մարմնում նոր ցիկլի կամ ցիկլերի կազմակերպման անհրաժեշտություն է առաջանում, որոնք միմյանց նկատմամբ կոչվում են ներքին և արտաքին ցիկլեր: Այդպիսի կառուցվածքներն անվանում են ներդրված ցիկլեր:

Բերենք օրինակ՝

Դասակարգել $A(a_1, a_2, a_3, \dots, a_n)$ տարրերը նվազման կարգով.

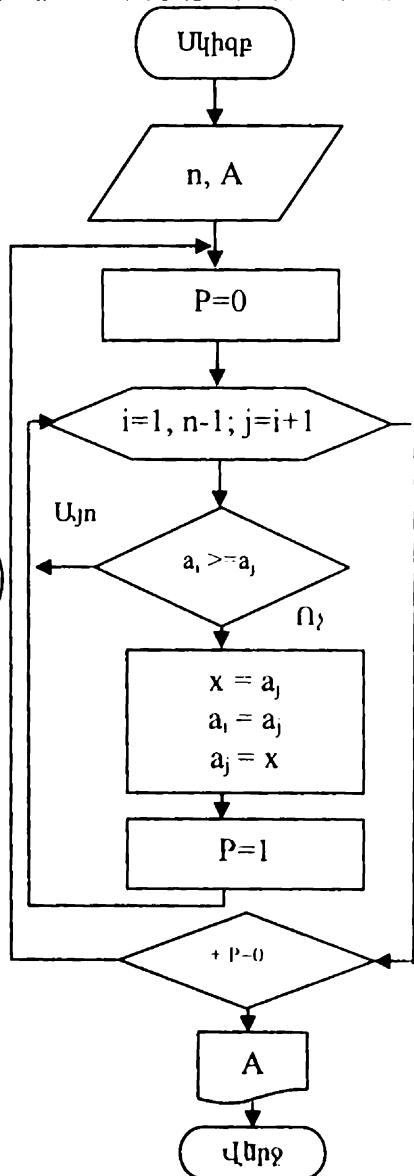
I եղանակ

(հերթական մաքսիմումների մեթոդ)



II եղանակ

(պղպջակների մեթոդ)



1.3. Ծրագրային արտադրանքի նախագծման մեթոդները

Ալգորիթմների և ծրագրերի նախագծման մեթոդները բազմազան են: Դրանք կարելի է դասակարգել ըստ տարրեր չափանիշների, որոնցից կարևորագույններն են համարվում՝

- նախագծային աշխատանքների մեջնայացման աստիճանը,
- մշակման պրոցեսի ընդունված տարրերակը:

Ըստ ավտոմատացման աստիճանի՝ ալգորիթմների և ծրագրերի նախագծումը կարելի է բաժանել՝

- նախագծման ավանդական մեթոդներ (ոչ ավտոմատացված),
- նախագծման ավտոմատացված մեթոդներ (Case տեխնոլոգիաներ):

Ոչ ավտոմատացված նախագծումը նախընտրելի է ոչ շատ աշխատատար և պարզ կառուցվածք ունեցող ծրագրային արտադրանքի համար: Այս ծրագրային արտադրանքը ունի մեծամասամբ կիրառական բնույթ:

Ավտոմատացված նախագծման առաջացման պատճառը նախագծային աշխատանքների ծախսերի և ժամանակի փոփրացմեն է:

Ալգորիթմների և ծրագրերի նախագծումը կարող է հիմնվել տարրեր մոտեցումների վրա, որոնցից առավել տարածված են՝

- ծրագրային արտադրանքի կառուցվածքային նախագծումը,
- առարկայական տիրույթի և նրա լրացումների ինֆորմացիոն մոդելավորումը,
- ծրագրային արտադրանքի օբյեկտակողմնորոշված նախագծումը:

Կառուցվածքային ծրագրավորման հիմքում ընկած է հաջորդական դեկոմպոզիհայի և նրա առանձին բաղկացուցիչների նպատակառուղիված կառուցվածքայնությունը: Կառուցվածքային նախագծման տիպիկ մեթոդ է հանդիսանում՝

- ծրագրային կոդավորումը և տեստավորումը,
- մոդուլային ծրագրավորումը,
- կառուցվածքային ծրագրավորումը:

Կախված կառուցվող օբյեկտից՝ տարրերում են.

- ֆունկցիոնալ-կողմնորոշված մեթոդներ, այսինքն խնդրի կամ պրոբլեմի բաժանումը առանձին, որոշակի ֆունկցիոնալությամբ օժտված պարզ բաղկացուցիչների,
- տվյալների կառուցման մեթոդներ:

Ֆունկցիոնալ-կողմնորոշված մեթոդների համար առաջին հերթին հաշվի են առնվում տվյալների մշակման ֆունկցիաները, որոնց համապատասխան որոշվում է ծրագրի առանձին բաղկացուցիչների կազմը և տրամաբանությունը:

Կառուցվածքային մոտեցման հիմնական շեշտը տվյալների մշակման մոդելավորման գործընթացների բաժանումն է: Կառուցվածքային մոտեցումը օգտագործում է

- տվյալների հոսքի դիագրամները (ինֆորմացիոն-տրամաբանական սխեմաները),
- առարկայական տիրույթի ինտեգրված կառուցվածքը (ինֆոլոգիական մոդել, ER- դիագրամներ),
- դեկոմպոզիցիայի դիագրամները. նպատակի, դեկավարման ֆունկցիաների, լրացումների կառուցվածքը և դեկոմպոզիցիան,
- կառուցվածքային սխեմաներ. ծրագրային արտադրանքի ուրվագիծը ծրագրային մոդուլների հիերարխիայի տեսքով. նրանց միջև կապերի նույնականացումը, բլոկ-սխեմաները.

Ծրագրային արտադրանքի վերաբերյալ ամբողջական պատկերացում կազմելու համար անհրաժեշտ է նաև բնութագրական տեքստային ինֆորմացիա: Տվյալների ինֆորմացիոն մոդելը և կառուցվածքը առավել մեծ նշանակություն ունեն առարկայական տիրույթի ինֆորմացիոն մոդելավորման համար: Այս սկզբունքն ընկած է տվյալների բազաների դեկավարման համակարգի հիմքում:

Ինֆորմացիոն ճարտարագիտության հիմնաղիր Զորջ Սարտինը առանձնացնում է տվյալ մոտեցման հետևյալ հիմնական բաղկացուցիչները

- առարկայական տիրույթի ինֆորմացիոն վերլուծություն /թիզմես տիրույթ/,
- ինֆորմացիոն մոդելավորում - տվյալների փոխադարձ կապի մոդելավորում,
- տվյալների մշակման ֆունկցիաների համակարգային նախագծում,
- տվյալների մշակման գործընթացի մանրակրկիտ կառուցում:

Սկզբնական փուլում կառուցվում են ներկայացման տարրեր մակարդակների ինֆորմացիոն մոդելները ինֆորմացիոն- տրամաբանական մոդելը, որը կախված չէ տվյալների մշակման և պահպանման ծրագրային իրականացումից և արտացոլում է

տվյալների ինտեգրված կառուցվածքը, դատալոգիական մոդելը /տվյալների մշակման և պահպաննան միջավայրի բնութագրիչը/:

Դատալոգիական մոդելը ունի տրամաբանական և ֆիզիկական ներկայացման մակարդակներ: Ֆիզիկական մակարդակում իրականացվում է տվյալների մշակման և պահպաննան գործընթացը: Տրամաբանական մակարդակը կիրառելի է տվյալների բազաների դեկավարման համակարգերում և իրականացվում է հետևյալ տեսքով:

- տվյալների բազաների սկզբունքային մոդելներ. տվյալների բազաների դեկավարման համակարգերի դեկավարումով տվյալների ինտեգրացված կառուցվածքներ,
- տվյալների արտաքին մոդելներ. նախատեսվում են տվյալների լրացումների իրականացման համար:

Ալգորիթմների և ծրագրերի մշակումը Ենթադրում է՝

- առարկայական տիրույթի օբյեկտային-կողմնորոշված վերլուծություն,
- օբյեկտային կողմնորոշված նախագծում:

Օբյեկտային կողմնորոշված վերլուծությունը առարկայական տիրույթի և օբյեկտների առանձնացման վերլուծություն է, դրանց մշակման մեթոդների, հատկությունների որոշում և դրանց միջև փոխադարձ կապի ճշգրտում: Օբյեկտակողմնորոշված նախագծումը միավորում է օբյեկտային դեկոմպոզիցիայի և ներկայացման գործընթացները՝ օգտագործելով տրամաբանական և ֆիզիկական մակարդակում ստատիկ և դինամիկ ձևով նախագծվող համակարգի տվյալների մոդելները:

Ծրագրային արտադրանքի նախագծման համար մշակված են օբյեկտային կողմնորոշված տեխնոլոգիաներ, որոնք ներառում են ծրագրավորման կողմնորոշված տեխնոլոգիաները, այսինքն ծրագրավորման հատուկ լեզուները և շահագործողական ինտերֆեյսի գործիքային միջավայրերը:

Ծրագրային արտադրանքի մշակման ավանդական մոտեցումները միշտ ընդգծել են տվյալների և նրանց մշակման գործընթացների միջև տարբերությունը: Ինֆորմացիոն մոդելավորման տեխնոլոգիաները սկզբից ընտրում են տվյալներն ըստ բնութագրիչների, որից հետո բնութագրում են այդ տվյալների գործընթացները:

Կառուցվածքային տեխնոլոգիաները կողմնորոշված են ճշտելու տվյալների մշակման և դրանց միջև ինֆորմացիոն հոսքերի կապակցող գործընթացները:

1.4. Ծրագրային արտադրանքի ստեղծման փուլերը

1. **Ծրագրավորման տեխնիկական համձնարարագրերի կազմումը պահանջում է՝**

- որոշել մշակվող ծրագրի պլատֆորմը օպերացիոն համակարգի տեսակը /Windows, Unix, OS/2 .../,
- գնահատել ծրագրի ցանցային տարրերակը,
- որոշել ծրագրերի մշակման անհրաժեշտությունը՝ այլ ծրագրային պլատֆորմի տանելու համար,
- հիմնավորել տվյալների բազաների դեկավարման համակարգերի դեկավարման տվյալների բազաների հետ աշխատանքը:

Այս փուլում որոշվում են խնդրի լուծման մեթոդները, մշակվում կոմպլեքս խնդիրների լուծման ընդհանրացված ալգորիթմը, նրա փունկցիոնալ կառուցվածքը կամ օբյեկտների կազմը: Եշգրտվում են ինֆորմացիայի մշակման տեխնիկական պահանջները և շահագործողի ինտերֆեյսը:

2. **Տեխնիկական նախագիծ:** Այս փուլում իրականացվում են այնպիսի կարևոր գործընթացներ, ինչպիսիք են

- ելեկով ծրագրային արտադրանքի նախագծման ընդունված մոտեցումից մշակվում է ալգորիթմը կամ ճշգրտվում է օբյեկտների կազմը, դրանց հատկությունները, մշակման մեթոդները և այլն,
- որոշվում է ընդհանուր համակարգային ծրագրային ապահովումը ներառելով բազային միջոցները /օպերացիոն համակարգը, տվյալների բազաների դեկավարման համակարգերի մոդելը, էլեկտրոնային աշյուսակները/.
- մշակվում է ծրագրային արտադրանքի ներքին կառուցվածքը՝ առանձին ծրագրային մոդուլների տեսքով,
- իրականացվում է ծրագրային մոդուլների գործիքային միջոցների ընտրությունը:

3. **Աշխատանքային փաստաթղթեր /աշխատանքային նախագիծ/:** Այս փուլում իրականացվում է ծրագրային ապահովման բազային միջոցների հարմարեցումը (աղապտացիան): Իրականացվում է ծրագրային մոդուլների մշակումը կամ օբյեկտների մշակման մեթոդը ծրագրավորումը կամ ծրագրային կոդի ստեղծումը: Կատարվում է ծրագրային արտադրանքի ինքնուրույն և համալիր հղկում, բազային ծրագրային միջոցների և մոդուլների

աշխատունակության փորձարկում: Յանալիր հեկման համար նախատեսվում է ստուգողական օրինակ, որը ստուգում է ծրագրային արտադրանքի համապատասխանությունը:

Այս փուլի հիմնական աշխատանքային արդյունքը ծրագրային արտադրանքի շահագործման փաստաթղթերի ստեղծումն է (շահագործման նկարագրությունը, ցուցումներ շահագործողին և ծրագրի տեղադրման և գործարկման յուրահատկությունները):

4. *Սուտքագրումը և աշխատանքը*: Պատրաստի ծրագրային արտադրանքը անցնում է փորձանական շահագործում, որից հետո միայն ներկայացվում է զանգվածային շահագործման:

1.5. Ծրագրային արտադրանքի կառուցվածքը

Ծրագրային արտադրանքը բաղկացած է փոխկապակցված ծրագրային մոդուլներից: Մոդուլը ծրագրի առանձին մասն է, որն ապահովում է որոշակի ֆունկցիաներ ծրագրի հետագա քայլերի իրականացման համար:

Ծրագրային արտադրանքը օժտված է ներքին կառուցվածքայնությամբ, որի նպատակն է բաժանել աշխատանքը ըստ կատարողների, կառուցել աշխատանքային օրացուցային գրաֆիկ և իրականացնել աշխատանքների հսկումը: Որոշ ծրագրային արտադրանքներ օգտագործում են ստանդարտ ենթածրագրեր՝ պատրաստի գրադարանային մոդուլներ: Մոդուլների բազմության մեջ տարբերակում են՝

- գլխավոր մոդուլը, որը ղեկավարում է ծրագրային արտադրանքի գործարկումը,
- ղեկավարող մոդուլը, ապահովում է ենթամոդուլների աշխատանքը,
- աշխատանքային մոդուլները իրականացնում են մշակման ֆունկցիաներ,
- ծառայողական մոդուլներ, գրադարաններ և օժանդակ ծրագրեր, որոնք ունեն սպասարկող ֆունկցիաներ:

Յուրաքանչյուրը կազմավորվում է ինքնուրույն ֆայլի տեսքով:

Կիրառական ծրագրային փաթեթը (application program package) որոշակի դասի խնդիրների լուծման համար նպատակառությամբ ծրագրային համակարգ է: Այս փաթեթների բաղդրիչները միավորված են ընդհանուր տվյալներով /բազաներով/, փոխկապակցված են և օժտված են համակարգային հատկություններով:

1. 6. Օբյեկտակողմնորոշված ծրագրավորում

Հաշվողական տեխնիկայի զարգացմանը համապատասխան առաջացել են ծրագրավորման տարբեր մեթոդներ: Յուրաքանչյուր փուլում ստեղծվել է նոր մոտեցում, որը օգնել է ծրագրավորողներին հաղթահարել ծրագրերի աճող բարդությունները: Առաջին ծրագրերը ստեղծվել են համակարգչի առջևի վահանակի բանալիային փոխանցատիչներով: Ակնհայտ է, որ այդպիսի եղանակը պիտանի է միայն շատ փոքր ծրագրերի համար: Այնուհետև հայտնագործվեց Ասեմբլեր լեզուն, որը թույլ տվեց գրել ավելի երկար ծրագրեր: Հաջորդ քայլը կատարվեց 1950 թ.-ին, երբ ստեղծվեց բարձր մակարդակի առաջին լեզուն Ֆորտրանը: Օգտագործելով բարձր մակարդակի լեզուն ծրագրավորողները կարող էին գրել մինչև մի քանի հազար տող երկարությամբ ծրագրեր: Այդ ժամանակվա համար նճան մոտեցումը ամենահեռանկարայինն էր, սակայն մեծ ծրագրերի ժամանակ դրանք դառնում էին չկարդացվող (անկառավարելի), ուստի այդպիսի ծրագրերին 1960 թ.-ին փոխարինելու եկավ կառուցվածքավորված ծրագրավորումը: Դրանց են վերաբերում Ալգոր, Պասկալ և C լեզուները: Կառուցվածքային ծրագրավորումը հասկանում է ծիշտ նշանակված դեկավարող ստրուկտուրաներ, ծրագրային բլոկներ, ցոտ օպերատորների բացակայություն (կամ ծայրահեղ դեպքում նվազագույն օգտագործում), ինքնուրույն ենթածրագրեր, որտեղ պաշտպանվում են ռեկուրսիոն և լոկալ փոփոխականները: Կառուցվածքային ծրագրավորումը հնարավորություն է տալիս ծրագիրը տրոհել բաղկացուցիչ մասերի: Միջին մակարդակ ունեցող ծրագրավորողը կարող է ստեղծել և պաշտպանել 50000 տող երկարությամբ և ավելի ծրագրեր: Սակայն որոշակի երկարության հասնելու դեպքում կառուցվածքային ծրագրավորումը նույնական անզոր էր դառնում: Ավելի բարդ ծրագիր գրելու համար անհրաժեշտ եղավ ծրագրավորման նոր մոտեցում: Արդյունքում մշակվեցին օբյեկտակողմնորոշված ծրագրավորման սկզբունքները: ՕԿԾ-Ն կուտակել է լավագույն գաղափարները, որոնք կային կառուցվածքային ծրագրավորման մեջ և համարում է այն նոր հզոր գաղափարների հետ, որոնք թույլ են տալիս օպտիմալ ձևով կազմակերպել ծրագրերը: ՕԿԾ-Ն թույլ է տալիս պրոբլեմ տրոհել փոխակապված խնդիրների:

Յուրաքանչյուր պրոբլեմ դառնում է ինքնուրույն օբյեկտ, որը պարունակում է սեփական կողերը և տվյալները, որոնք վերաբերում են այդ օբյեկտին: Այդ դեպքում ամբողջ պրոցեդուրան պարզեցվում է և հնարավոր է դառնում ավելի մեծ ժավալով ծրագրեր կազմել:

ՕԿԾ-ն բոլոր լեզուները, ներառյալ C++, հիմքում ունեն երեք հիմնադիր գաղափարներ՝ ինկապուլյացիա, պոլիմորֆիզմ և ժառանգականություն:

Ինկապուլյացիա (Encapsulation-կաղապարապատում)

Դա մեխանիզմ է, որը միավորում է կողերը և տվյալները, մանիպուլյացիա է անում այդ տվյալների հետ և պաշտպանում է երկուսն էլ արտաքին միջամտությունից և ոչ ճիշտ օգտագործումից: ՕԿԾ-ում կողերը և տվյալները կարող են միավորվել. այդ դեպքում ասում են, որ ստեղծվում է, այսպես կոչված՝ «սև արկղ»: Բոլոր անհրաժեշտ տվյալները և կողերը գտնվում են այդ «արկղ» ներսում: Եթե կողերը և տվյալները միավորվում են այդ եղանակով, ստեղծվում է օբյեկտ: Այլ խոսքով՝ օբյեկտը այն է, որը պաշտպանում է ինկապուլյացիան:

Օբյեկտի ներսում կողը և տվյալները կարող են այդ օբյեկտի համար լինել փակ (Private) կամ բաց (Public): Փակ կողերը կամ տվյալները հասանելի են միայն այդ օբյեկտի այլ մասերի համար: Այդպիսով, փակ կողերը և տվյալները անհասանելի են ծրագրերի այն մասերի համար, որոնք գոյություն ունեն օբյեկտից դուրս: Եթե կողերը և տվյալները բաց են, ապա չնայած այն բանին, որ դրանք տրված են օբյեկտի ներսում, հասանելի են ծրագրի այլ մասերի համար: Բնութագրական է համարվում այն իրավիճակը, եթե օբյեկտի բաց մասը օգտագործվում է այն բանի համար, որպեսզի ապահովի հսկող ինտերֆեյս փակ մասի համար: Իրականում օբյեկտը որոշակի շահագործողական տիպի փոփոխական է: Տվյալների այդ տիպի յուրաքանչյուր էլեմենտ հանդիսանում է բաղադրյալ փոփոխական:

Պոլիմորֆիզմ (Polymorphism-բազմաձևություն /ծանրաբեռնում/. մեկ ինտերֆեյս՝ բազմաթիվ մեթոդներով: Դա հատկություն է, որը թույլ է տալիս նույն անումը օգտագործել երկու կամ ավելի խնդիրների լուծման համար (տեխնիկապես տարրեր իրագործելի խնդիրների համար):

Պոլիմորֆիզմի նպատակը կիրառելի է ՕԿԾ-ում, որը նույն անվան օգտագործումն է դասի համար ընդհանուր գործողություն նշելու համար: Յուրաքանչյուր կոնկրետ գործողության կատարումը պետք է որոշվի տվյալների տիպով: Օրինակ C լեզվի համար, որտեղ պոլիմորֆիզմը ոչ բավարար է պաշտպանած, թվի բացարձակ մեջության որոշումը, պահանջում է երեք տարբեր ֆունկցիաներ՝ abs(), Labs() և fabs(): C++ -ում այդ ֆունկցիաներից յուրաքանչյուրը կարող է անվանվել abs(): C++ -ում կարելի է օգտագործել ֆունկցիայի նույն անունը տարբեր գործողությունների բազմության վրա: Դա կոչվում է ֆունկցիայի ծանրաբեռնում (function overloading): Ավելի ընդհանուր իմաստով պոլիմորֆիզմի գաղափարն է. «Մեկ ինտերֆեյս՝ բազմաթիվ մեթոդներով»: Դա նշանակում է, որ իմաստով մոտ գործողությունների համար կարելի է ստեղծել ընդհանուր ինտերֆեյս: Այդ ժամանակ կոնկրետ գործողության կատարումը կախված է տվյալներից: Պոլիմորֆիզմի առավելությունը այն է, որ օգնում է պարզեցնել ծրագիրը թույլ տալով օգտագործել նույն ինտերֆեյսը գործողությունների միասնական դասի համար: Կախված իրավիճակից կոնկրետ գործողության ընտանիքը դրվում է համակարգչում: Ծրագրավորողին կարիք չկա, որ ինքը ստեղծի ընտանիքը: Պետք է ուղղակի հիշել և օգտագործել ընդհանուր ինտերֆեյսը: Պոլիմորֆիզմը կարող է կիրառվել ֆունկցիաների և օպերացիաների նկատմամբ: Փաստորեն ծրագրավորման թույլը լեզուներում սահմանափակ կիրառվում է պոլիմորֆիզմը, օրինակ թվաբանական օպերացիաներում: Այդ ժամանակ կոմպիլյատորը ավտոմատ որոշում է, թե թվաբանության ինչ տիպ է պահանջվում: C++- ում մենք կարող ենք կիրառել այդ կոնցեպցիան մեր կողմից տրված տվյալների տիպին: Պոլիմորֆիզմի այլպիսի տիպը կոչվում է օպերացիայի ծանրաբեռնում (operator overloading): Պոլիմորֆիզմ հասկացությունում առանցքայինն այն է, ինչը թույլ է տալիս նաև կոնկրետ տարբեր աստիճանի բարորությամբ օբյեկտների հետ նրանց համար ստեղծելով ընդհանուր ինտերֆեյս նման գործողությունների իրականացնան համար:

Ժառանգականություն (inheritance): Դա գործընթաց է, որի միջոցով մեկ օբյեկտ կարող է ծեղոք բերել մյուսի հատկությունները: Ավելի ճիշտ, օբյեկտը կարող է ժառանգել ուրիշ օբյեկտի հիմնական հատկությունները և ավելացնել նաև այնպիսի գծեր,

որոնք բնութագրական են միայն իրեն: Ժառանգականությունը թույլ է տալիս պաշտպանել դասերի հիերարխիայի գաղափարը, որի կիրառումը ինֆորմացիայի մեջ հոսքերը դարձնում են կառավարելի:

Յուրաքանչյուր դեպքում ծնված դասը ժառանգում է բոլոր հատկությունները, որոնք կապված են «ծնող» դասի հետ և ավելացնում է իր սեփական որոշ բնութագրերը:

Առանց աստիճանակարգի դասերի օգտագործման, յուրաքանչյուր օբյեկտի համար հարկ կլինի տալ բոլոր բնութագրերը, սպառիչ նկարագրելու համար: Սակայն ժառանգականության օգտագործման ժամանակ կարելի է նկարագրել օբյեկտը՝ որոշելով այն ընդհանուր դասը, որին նա վերաբերում է և ավելացնելով առանձնահատուկ գծեր՝ այն դարձնելով եզակի:

Գլուխ 2

«Visual Basic»

2.1. «Visual Basic» ծրագրավորման լեզուն

Այսօր չկա ծրագրավորում այդ բառի նախկին իմաստով: Յավելվածը նախագծվում է: Մասնավորապես, ծրագրավորողներին ավելի տեղին կլինի անվանել ծրագրային միջոցների ինժեներնախագծողներ: Visual Basic-ում հավելվածի ստեղծումը նշանակում է նախագծել այն ոչ միայն Windows-ի համար, այլև Windows-ի օգնությամբ ստեղծել նրան այնպիսին, ինչպիսին Windows-ն է:

Արդեն «Visual» բառից կարելի է կռահել, որ Visual Basic-ում իրականացված է ծրագրավորման տեսանելի ոճը: Մենք նույնիսկ չենք ծրագրավորում, այլ նախագծում ենք հավելվածը: Մեր առաջին խնդիրը այդ ժամանակ աշխատանքային միջավայր ստեղծելն է մինչ կողի առաջին տողի հավաքելը: Visual Basic-ը և ինտերպրետատոր է, և կոմպիլյատոր: Visual Basic-ը հնարավորություն է տալիս ստեղծել կատարվող Exec ֆայլեր, ուստի նրան կարելի է դասել կոմպիլյատորների շարքը, թեև չի կարելի անվանել մաքուր կոմպիլյատոր, քանի որ ի տարրերություն, ասենք Visual C++ -ի, Visual Basic-ը անմիջապես մշակման միջավայրից չի ստեղծում կատարվող ֆայլ մեկնարկման ժամանակ: Այդպիսի ֆայլի ստեղծման համար անհրաժեշտ է անել դա ակնհայտորեն (File\Make ***.Exe): Գոյություն ունի Visual Basic-ի երեք տարրերակ: Առաջինը սկսնակների համար է (Learning Edition), որը նախատեսված է ոչ փորձառու ծրագրավորողների համար, երկրորդը պրոֆեսիոնալների համար տարրերակն է (Professional Edition): Այն պարունակում է մի շարք դեկավարման լրացուցիչ տարրեր, ապահովում է տվյալների բազային դիմելու ընդլայնված հնարավորություն և ստեղծելու հավելվածներ OLE սերվերների համար: Երրորդը արդյունաբերական տարրերակը (Enterprise Edition), թույլ է տալիս մշակել բավական բարդ հաճախորդաբարակների հավելվածներ: Այն պարունակում է նաև հատուկ գործիքներ: Օրինակ Visual Source Safe-ը նախատեսված է տարրերակների համեմատման և նախագծերի դեկավարման համար:

2.2. Հավելվածների (լրացումներ) ստեղծումը «Visual Basic»-ով

Visual Basic-ը համդիսանում է ընդհանուր լեզվական հիմք բոլոր լրացումների համար (Excel, Word, PowerPoint): Visual Basic-ը պահպանում է Basic-ի նախորդ տարրերակների հիմնական կառուցվածքը և կանոնները:

Visual Basic-ը (VB, VBA) Microsoft Office միջավայրում կիրառական ծրագրերի ստեղծման վիզուալ ծրագրավորման զարգացած համակարգ է, պատկանում է օբյեկտակողմնորոշված ծրագրավորման դասին :

Visual Basic-ի օգնությամբ կարելի է ստեղծել շահագործողի գրաֆիկական ինտերֆեյսի դեկավարման օբյեկտներ, ընտրել և փոփոխել օբյեկտների հատկությունները՝ կցելով նրանց համապատասխան ծրագրային կոդ: VB-ի գործիքային միջավայրի օգնությամբ ծրագրավորման մեթոդները բերվում են հետևյալին:

- դեկավարման և հսկման օբյեկտների ստեղծում,
- օբյեկտների կանչման համար գործընթացների մշակում:

Visual Basic լեզվով կազմված կիրառական ծրագրերը միաձուլված են հետևյալ հասկացությունների հետ՝

- հսկման և դեկավարման օբյեկտ. Էկրանի ձևապատճերներ (Փորմաներ) /Form/, Փորմայի վրա գրաֆիկական էլեմենտներ, այդ թվում նաև տեքստային պատուհաններ, տեղաշարժի գոտիներ, հրամանի մեղմակներ և այլն,
- հատկություն (բնութագիր). Դեկավարման օբյեկտի բնութագիրը, հատկությունների նշանակությունները,
- պատահար. Դեկավարման օբյեկտի կողմից հասկացվող գործողություն,
- մուտքի մեթոդ. Օբյեկտի վրա ազդող ֆունկցիա կամ օպերատոր,
- գործընթաց. Ենթածրագրեր կամ ֆունկցիաներ, Visual Basic-ի օպերատորների կամայական հաջորդականություն:

Գործընթացները բաժանվում են իրավիճակի (գործարկվում են ընթացիկ իրավիճակի դեպքում) և ընդհանուր նշանակության:

- VB-ով գրված ծրագրերը ստեղծվում են երկու ճանապարհով՝
- մեթնայական ուժիմում՝ ստեղնաշարային միկրոիրամանների կառուցմանք,

- η έκθετη παραγωγή στην αγορά της Ευρώπης είναι σημαντική για την οικονομία της Ελλάδας.

Visual Basic-ը օգտագործում է p-Code տեխնոլոգիան, այսինքն յուրաքանչյուր ներմուծված տող վերածնավորում է միջանցիկ կողի:

2. 3. Օբյեկտ հասկացությունը և նրա բնութագրերը

Visual Basic 6.0-ն դասվում է օբյեկտակողմնորոշված լեզուների դասին: Օբյեկտակողմնորոշված ծրագրավորումը կարելի է ներկայացնել որպես հավելվածների նախագծման և կառուցման վերլուծման մեթոդ, որտեղ օգտագործվում են օբյեկտներ: Օբյեկտը կարելի է համարել տվյալների և կոդի մշակման համար նախատեսված նրանց որոշակի ամբողջությունը:

Բոլոր տեսանելի տարրերը, ինչպիսիք են ֆորման, սեղմակը, մուտքագրման դաշտը, գրությունը և այլն, օբյեկտներ են: Visual Basic 6.0-ում գոյություն ունեն հարյուրից ավելի ներդրված օբյեկտներ, որոնք գտնվում են աստիճանակարգի տարրեր մակարդակներում: Աստիճանակարգը որոշում է օբյեկտների միջև կապը և ցույց է տալիս օբյեկտներին դիմելու ճանապարհը: Այդ ճանապարհը ներկայացվում է մի շարք անվանումներով, որոնք բաժանված են կետով: Օրինակ

Form.Text1- (հիմնավորման այս ձևը կոչվում է բացահայտ հիմնավորում):

Պարտադիր չէ միշտ նշել օբյեկտ տանող ուղին: Այս դեպքում բավարար է օբյեկտի ոչ բացահայտ հիմնավորումը: Օրինակ, եթե Form1 կողի մոդուլում տալիս ենք հիմնավորում Text1 մուտքագրման դաշտին, ապա կարող ենք գրել Text1: Բացի ներդրված օբյեկտներից, շահագործողը ինքը կարող է ստեղծել իր օբյեկտները:

OLE օբյեկտները : Visual Basic 6.0-ում օգտագործվում է OLE (Object Linking and Embedding-օբյեկտների կապում և ներդրում) մեխանիզմը, որը թույլ է տալիս փոխանագործակցել ցանկացած OLE մեխանիզմ ունեցող ծրագրի հետ: OLE մեխանիզմը գործում է, օրինակ WordArt, ClipArt և այլ ծրագրերով ստեղծված օբյեկտների տեղադրման համար:

Դաս և մեթոդ: Օբյեկտային կողմնորոշված ժրագրավորման կարևորագույն հասկացություն է հանդիսանում կարգը: Կարգը սովորաբար նկարագրվում է որպես նախագիծ, որի հիմքի վրա պետք է ստեղծվի կոնկրետ օբյեկտը: Այլ խոսքով, կարգը բնութագրում է օբյեկտի անվանումը, հատկությունները և նրա հետ իրականացվող գործողությունները: Մեթոդի կառուցվածքը հետևյալն է Օբյեկտ.մեթոդ:

Օրինակ՝ Form1.Cls—Cls (նարքել) մեթոդի օգնությամբ կատարվում է բոլոր էլեմենտների ջնջում:

Դատկություններ: Օբյեկտները ունեն այնպիսի հատկություններ, ինչպիսիք են չափերը, գույնը, էլրանի վրա դիրքը կամ կարգավիճակը (օրինակ՝ հասանելիությունը կամ տեսանելիությունը):

Բնութագրերի փոփոխման կառուցվածքը հետևյալն է:

Օբյեկտ.հատկություն=հատկության արժեք:

Օրինակ՝ Form1.caption="օրինակ":

Այստեղ նույնպես գործում են բացահայտ և ոչ բացահայտ հիմնավորումների սկզբունքները:

Դատկություններից հատկանշական են տրամաբանական հատկությունները (read only): Տրամաբանական հատկությունները ընդունում են երկու նշանակություն՝ իրական կամ 1 և ոչ իրական կամ 0: Ղեկավարման էլեմենտների տրամաբանական հատկություն է Visible-ը: Եթե Visible=True, ապա ղեկավարման էլեմենտը արտացոլվում է Form-ի վրա: Եթե Visible=False, ապա չի արտացոլվում:

Դատահար (իրադարձություն): Իրադարձությունը գործողություն է, որը ընդունվում է օբյեկտի կողմից (օրինակ՝ մկնիկի հարված կամ ստեղնի սեղմում) և որը կարելի է ծրագրավորել: Այլ խոսքով, գործողությունները հանդիսանում են իրադարձություններ, իսկ հետևանքները՝ գործընթացներ:

2.4. Visual Basic 6.0-ի աշխատանքային միջավայրը

Visual Basic 6.0 ծրագրի գործարկումը կատարվում է հիմնականում հետևյալ ժամապարհով՝

1. **Start** սեղմակի գործարկումից հետո՝ ընտրելով **Program** մենյուից **Microsoft Visual Basic 6.0** Ենթամենյուն,
2. Բացված **New Project** երկխոսության պատուհանը կառաջարկի նախագծերի տարբեր նախատիպեր (շաբլոններ), որոնք կիեշտացնեն հավելվածների ստեղծումը։ Այս պատուհանը բարկացած է երեք ներդիրներից
 - **New** ներդիրի օգնությամբ կատարվում է նոր նախագծի /New Project/ գործարկումը,
 - **Existing** ներդիրի օգնությամբ իրականացվում է գոյություն ունեցող նախագծերի գործարկումը /Open Project/,
 - **Recent** ներդիրը առաջարկում է վերջին շրջանում մշակված նախագծերի ցուցակը,
3. New Project երկխոսության պատուհանի New ներդիրի վրա տեղակայված **Standard.Exe** նշանի վրա կատարում ենք մկնիկի կրկնակի սեղմում։

Վերոհիշյալ գործողությունների կատարումից հետո էկրանին կհայտնվի VB6.0 աշխատանքային միջավայրը։

Զևապատկերի (Ֆորմայի) պատուհանը նախատեսված է կառուցելու ստեղծվող ծրագրի աշխատանքային ինտերֆեյսով։ Այս պատուհանը կարող է պարունակել գլխավոր մենյու, իրանանի սեղմակներ, ցուցակների պատուհաններ, տեղաշարժի գոտիներ և այլ էլեմենտներ, որոնք գոյություն ունեն Windows-ի կիրառական ծրագրերում։

VB 6.0-ի աշխատանքային միջավայրի գործարկման հետ միաժամանակ հայտնվում է Form1 պատուհանը ստանդարտ կետապատ կողորդինատական ցանցի տեսքով։ Վերջինս հեշտացնում է ինտերֆեյսի ստեղծման աշխատանքը։ Form1 պատուհանը կարող է գրավել էկրանի մի մասը կամ ամբողջ էկրանը։ Նոր ֆորմա պելացնելու համար անհրաժեշտ է ընտրել Project մենյուի Add Form ենթակետը։

Form պատուհանի գործարկման բնութագրերը կարգավորվում են Form Layout պատուհանի օգնությամբ։

2.5. Ղեկավարման էլեմենտների գոտի

Դաշտավայրը ստեղծումը գործնականում անհնար է առանց ղեկավարման էլեմենտների օգտագործման, քանի որ դա շահագործողի և հավելվածների փոխհամագործակցության միակ միջոցն է: Ղեկավարման էլեմենտների հավաքածուն սահմանափակ չէ, և շահագործողի ցանկությամբ կարելի է ընդլայնել դրանց քանակը Project\Components կամ (Ctrl+t) ճանապարհով:

Ղեկավարման էլեմենտների հետ աշխատելու ժամանակ հարկավոր է իիշել, որ դրանց կարելի է վերաբերել որպես փոփոխականների հատկություններին տալով կամ հանելով որոշակի արժեքներ ինչպես նշակման փուլում, այնպես էլ ծրագրի կատարման ժամանակ:

Ղեկավարման էլեմենտները (Toolbox) նախատեսված են շահագործողական ինտերֆեյսը կառուցելու համար: Այս գոտին տեղակայված է լինում էկրանի ձախ մասում: Բոլոր սեղմակները, բացի առաջինից, նախատեսված են ստեղծելու ղեկավարման էլեմենտները: Առաջին սեղմակը կոչվում է ցուցիչ (Pointer): Կատարելով մկնիկով հարված ցուցիչով նշանի վրա՝ կարելի է ընտրել ցանկացած էլեմենտ, կատարել դրա չափերի, դիրքի և այլ բնութագրերի փոփոխություն Form1 պատուհանի վրա: Այն բանից հետո, եթե ղեկավարման էլեմենտները տեղափոխվում են Form պատուհանի վրա, դրանք դառնում են օբյեկտներ, այլ խոսքով՝ շահագործողական ինտերֆեյսի ծրագրային էլեմենտներ: Յուրաքանչյուր ղեկավարման էլեմենտ նախատեսված է որոշակի փունկցիայի կատարման համար: Եթե մկնիկը տեղադրենք ցանկացած ղեկավարման էլեմենտի վրա, ապա կտրվի հաղորդագրություն տվյալ էլեմենտի անվան մասին: Ֆորման պահպում է տերսուային ֆայլի տեսքով և կարող է ունենալ հետևյալ հատկություններն ու իրադեպերը:

BorderStyle հատկության միջոցով շահագործողը կարող է ձևափոխել ֆորմայի եզրագծերի տեսքը: ControlBox-հատկությունը թույլատրում կամ արգելում է համակարգային մենյուի աշխատանքը: MaxButton-հատկությամբ ամբողջ էկրանը տրամադրվում է ֆորմային: MinButton հատկությամբ այն տեղադրում է status bar գոտում: Load պատահարը տեղի է ունենում ֆորմայի սկզբնական բեռնման ժամանակ: UriLoad պատահարը տեղի է ունենում ֆորմայի աշխատանքի դադարեցման պահին:

Ներկայացնենք հիմնական դեկավարման էլեմենտները:

TextBox *Տեքստային դաշտ /անվանումը՝ Text/* - Հիմնականում նախատեսված է ինֆորմացիայի մուտքագրման համար: Ունի հետևյալ պատահարները Change-գործում է տեքստային դաշտում որևէ փոփոխություն կատարելուց հետո՝ յուրաքանչյուր նիշի մուտքագրման, հեռացման կամ փոփոխման ժամանակ: LostFocus և GotFocus պատահարներից առաջնը գործում է, երբ տեքստային դաշտը դադարում է ակտիվ լինել, երկրորդը երբ տեքստային դաշտը ակտիվանում է: KeyPress պատահարը գործում է, երբ սեղմվում է ստեղնաշարի որևէ ստեղն: Ընդ որում, այս պատահարը վերադարձնում է սեղմված ստեղնի կողը, որը կարելի է օգտագործել ծրագիրը ձևավորելիս: Բոլոր էլեմենտները, որոնք նախատեսված են ինֆորմացիայի մուտքի համար, ունեն text հատկություն, որը պարունակում է տեքստային դաշտի ինֆորմացիան, իսկ այն էլեմենտները, որոնք նախատեսված են դիտելու համար, ունեն Caption հատկություն: Բազմատող դաշտի ներմուծման համար օգտագործվում է MultiLine հատկությունը: Եթե ներմուծվող տեքստն ավելի երկար է, քան նախատեսված տիրույթը, ապա ScrollBar հատկության օգտագործելու դեպքում ավտոմատ առաջանում է տեղաշարժի գոտի: Տեքստային դաշտում տեքստն ընդգծելու համար SelStart հատկությունը նշում է տեքստի սկիզբը, իսկ SelLength-ը նիշերի քանակը: Տեքստային դաշտը անհասանելի կամ փակ սահմանելու համար օգտագործվում են Enabled և Locked հատկությունները: Առաջին պարամետրը False ընտրելու դեպքում, տեքստային դաշտը ընդունում է քաց մոխրագույն գույն և չի ակտիվանում, այսինքն անհասանելի է: Երկրորդի պարամետրը True ընտրելու դեպքում թեև տեքստային դաշտը կարող է ակտիվանալ, սակայն նրանում փոփոխություն անել հնարավոր չէ: PasswdChar-ը հետաքրքիր հատկություններից է: Եթե մուտքագրվող ինֆորմացիան ծածկագրի բնույթ է կրում, ապա կարելի է այս դաշտում հավաքել որևէ նիշ (հաճախ *) և տեքստային դաշտի բոլոր նիշերը կներկայացվեն այդ նիշով: Սակայն այս հատկությունը գործում է, եթե Multiline հատկության պարամետրը ընտրված է False:

Label գրություն /անվանումը՝ Label/ - Սա գրություն է, որը շահագործողը չի կարող ստեղնաշարով փոփոխել: Դիմնական հատկություններից են Alignment-գրության հավասարեցումը եզրերի նկատմամբ, Autosize-գրության չափերից կախված պիտակի

չափերը ավտոմատ փոփոխվում են, WordWrap-տողում չտեղափորվող բառը ավտոմատ տեղափոխվում է հաջորդ տող, Caption-պարունակում է գրության տեքստը:

CommandButton սեղմակ /անվանումը՝ Command/ - Օգտագործվում է այն դեպքերում, եթե անհրաժեշտ է լինում սկսել կամ ավարտել ինչ-որ գործընթաց: Գլխավոր պատահարը Click-ն է: Կանչը կատարվում է տարբեր ձևերով՝ մեջիկով, [Tab]+[Enter] ստեղներով կամ պրոցեդուրայի ներսից Value հատկությանը True արժեք տալով: Այս էլեմենտը ունի երկու հատկություն, որոնք առաջիկ հետաքրքիր են և կապված են Click պատահարի հետ: Առաջինը Default հատկությունն է, որը նշանակում է, որ տվյալ սեղմակը համարվում է ակտիվ՝ լրացնելու կանոնով: Եթե այդ հատկության պարամետրը ընտրված է True, ապա [Enter] ստեղնը սեղմելուց հետո կատարվում է Click պատահարը: Ընդ որում, այս հատկության պարամետրը կարող է լինել True միայն ֆորմայի մեկ էլեմենտի համար (սովորաբար այս հատկությունն ունենում են OK կոճակները): Երկրորդը Cancel հատկությունն է, որը նշան է նախորդ հատկությանը այն տարբերությամբ, որ սեղմակի Click պատահարը կատարվում է [Esc] ստեղնը սեղմելուց հետո (սովորաբար այս հատկությունն ունենում են Cancel-ընդհատել սեղմակները):

ListBox Ցուցակ /անվանումը՝ List/ - Շահագործողին հնարավորություն է տալիս ցուցակից ընտրելու մեկ կամ մի քանի էլեմենտ: Եթե ցուցակին հատկացված դաշտում ոչ բոլոր էլեմենտներն են տեղափորվում, ապա ավտոմատ առաջանում է տեղաշարժի գոտի: Հիմնական պատահարը հանդիսանում է Click-ը: Ցուցակի նոր էլեմենտների ավելացումը կատարվում է AddItem մեթոդի միջոցով՝ ListBox.AddItem էլեմենտ [, Index]: Index պարամետրը ցույց է տալիս, թե որտեղ պետք է ավելացվի նոր էլեմենտը: Ցուցակի էլեմենտի հեռացումը կատարվում է RemoveItem մեթոդի միջոցով: Ցուցակի բոլոր էլեմենտների հեռացումը կատարվում է Clear մեթոդի միջոցով: Text հատկության միջոցով կարելի է ստանալ ինդեքսին հանապատախան ընտրված էլեմենտի տեքստը: Ցուցակից ընտրված էլեմենտի ինդեքսը կարող ենք ստանալ ListIndex հատկությամբ: List() և ListIndex հատկությունների համակցման միջոցով կարելի է ստանալ ցուցակի ընտրված էլեմենտը:

Օրինակ՝ ListItem= ListBox.List (ListBox.ListIndex): Եթե ցուցակի ոչ մի էլեմենտ ընտրված չէ, ապա ListIndex հատկության արժեքը -1 է: Ցուցակի էլեմենտների ընթացիկ քանակությունը պահպանվում է ListCount հատկության մեջ: Sorted հատկությունը որոշում է ցուցակի էլեմենտների դասավորության կարգը: Եթե sorted=True, ապա ցուցակի էլեմենտները կդասավորվեն այբբենական կարգով: Վերջին ավելացված ինդեքսը գրվում է NewIndex հատկության մեջ: Հահագործողը կարող է միաժամանակ ընտրել մի քանի էլեմենտներ օգտագործելով Multiline հատկությունը, որն ընդունում է հետևյալ արժեքները. 0՝ հնարավոր չէ բազմակի ընտրությունը, 1՝ հնարավոր է պարզ բազմակի ընտրություն, 2՝ հնարավոր է ընդլայնված բազմակի ընտրություն, որը իրականացվում է մկնիկի կամ [Shift] և [Ctrl] ստեղների միջոցով:

ComboBox կոմքինացված դաշտ /անվանումը Combo/: Ըստ եռթյան, դա ցուցակի և ներածման դաշտի կոմքինացիան է: Ցուցակով դաշտը օգտագործվում է այն դեպքում, եթե նախօրոք հնարավոր չէ որոշել այն արժեքները, որոնք պետք է ընդգրկվեն ցուցակում, կամ ցուցակը պարունակում է բավականին շատ քանակությամբ էլեմենտներ: Այսպիսի ցուցակում անհրաժեշտ արժեքները կարելի են ոչ միայն ընտրել, այլև անմիջապես մուտքագրել համապատասխան դաշտ: Ներմուծված նոր արժեքն ինքնարերաբար տեղադրվում է ցուցակում: Դիմնական իրադեպը Click-ն է, որն օգտագործվում է էլեմենտի ընտրության համար և change իրադեպը ներմուծվող դաշտի տեքստի փոփոխման համար:

OptionButton անջատիչ /անվանումը Option/: Սա առավելապես օգտագործվում է որոշակի խումբ պայմաններից որևէ մեկը ընտրելու համար: Սովորաբար դրանք խմբավորվում են մեկ Frame-ի մեջ: Ինչպես և CheckBox փոխանցատիչի համար, կարևոր է միայն Click իրադեպը: Կարևորագույն հատկություններից է Value-ն, որը կարող է ընդունել True կամ False արժեք կախված նրանից անջատիչը նշված է, թե ոչ: Style հատկությունը օգտագործվում է, եթե ցանկանում ենք այն ներկայացնել սեղմակի տեսքով: Դրա համար Style հատկության պարամետրը պետք է ընտրել Graphical:

CheckBox դրոշակ (նշիչ) /անվանումը Check/: Ստուգման նշիչը օգտագործվում է տարբեր տեսակի պայմաններ նշելու համար: Այն կարող է ունենալ երկու արժեք՝ նշված և չնշված: Կա նաև երրորդ արժեքը, որը ներկայացվում է ստուգման նշիչը նշված, բայց ոչ հասանելի տեսքով, սակայն այս արժեքը հնարավոր է տալ միայն

ծրագրային մեթոդներով: Կարևորագույն պատահար է հանդիսանում Click-ը: Կարևորագույն հատկություններից է Value-ն՝ կախված նրանից՝ ստուգման նշիչը նշված է, թե՛ ոչ, էլեմենտը կարող է ընդունել՝ Unchecked(0), Checked(1), Grayed(2): Style հատկությունը օգտագործվում է, եթե ցանկանում ենք այն ներկայացնել սեղմակի տեսքով: Դրա համար Style հատկության պարամետրը պետք է ընտրել Graphical:

Frame շղթանակ /անվանումը՝ Frame/: Ծրջանակը հանդիսանում է էլեմենտ-բեռնարկդ: Նա մի քանի էլեմենտներ միավորում է մեկ խմբի մեջ: Խմբի մեջ մտնող էլեմենտները կարելի է դիտարկել որպես մեկ ամբողջություն, այսինքն տեղաշարժել, ակտիվացնել, պահպանել, դարձնել տեսանելի կամ անտեսանելի: Ծրջանակը չունի հատկություն կամ պատահար:

HscrollBar հորիզոնական պտուման գոտի /անվանումը՝ Hscroll/: Յորիգոնական տիպի տեղաշարժի գոտին ունի Change իրադեպ, որն առաջանում է սողնակի տեղաշարժումից կամ Value և Scroll հատկությունների ծրագրային փոփոխություններից: Հարժման միջակայքը որոշվում է Min և Max հատկություններով:

VscrollBar ուղղահայաց տեղաշարժի գոտի /անվանումը՝ Vscroll/: Ուղղահայաց տիպի տեղաշարժի գոտին ունի Change իրադեպ, որն առաջանում է սողնակի տեղաշարժումից կամ Value և Scroll հատկությունների ծրագրային փոփոխություններից: Հարժման միջակայքը որոշվում է Min և Max հատկություններով:

Image պատկեր /անվանումը՝ Image/: Ի տարրերություն Picture-Box էլեմենտի պատկերի դեկավարման էլեմենտը չի հանդիսանում էլեմենտ-բեռնարկդ: Այն թույլ չի տալիս նկարել և խմբավորել օբյեկտներ: Սակայն այն օգտագործում է քիչ ռեսուրսներ, հետևաբար հարմար է նկարների արտացոլման համար: Գլխավոր հատկությունը նույնպես հանդիսանում է Picture-ը: Stretch հատկությունը որոշում է, թե ինչպես արտացոլել նկարը: Եթե հատկության արժեքը հավասար է True-ի, ապա նկարի չափերը փոփոխվում են մինչև Image դեկավարման էլեմենտի չափերը, հակառակ դեպքում նեկավարման էլեմենտն է փոփոխվում մինչև նկարի չափերը:

PictureBox գրաֆիկական դաշտ /անվանումը՝ Picture/ (նկար պատուհանով): Ինչպես երևում է անունից, այս էլեմենտը նախատեսված է նկարների և ուրիշ գրաֆիկական օբյեկտների արտացոլման համար: Այս դեկավարման էլեմենտը հանդիսանում է էլեմենտ-բեռնարկդ, ուստի նրան կարելի է օգտագործել ուրիշ

Էլեմենտներ միավորելու համար: Ֆորմայի վրա էլեմենտի դիրքը տալու համար օգտագործվում է Align հատկությունը: Տարբեր չափերի նկարների արտացոլման համար ղեկավարման էլեմենտի չափերի ավտոմատ փոփոխման համար օգտագործվում է Autosize հատկությունը: Ամենակարևոր հատկությունը Picture-ն է, որը պարունակում է արտացոլվող գրաֆիկական օբյեկտը: Դավելվածի կատարման ժամանակ հատկության փոփոխման համար օգտագործվում է LoadPicture ֆունկցիա: Պատկերի պահպանման համար կարելի է օգտվել SavePicture ֆունկցիայից: PictureBox-ի մեթոդները թույլ են տալիս նկարել կետ, գիծ և շրջանագիծ, ինչպես նաև Print մեթոդով տպել տեքստը:

Timer վայրկյանաշափ /անվանումը՝ Timer/: Վայրկյանաշափ օբյեկտը ծրագրում գործում է ինչպես անտեսանելի վայրկյանաշափ: Նրա օգնությամբ ժամանակի որոշակի պահի կարելի է սկսել կամ ավարտել հավելվածի գործունեությունը: Միակ պատահարը Timer-ն է, որը գործում է նախատեսված ժամանակահատվածը լրանալու պահին: Ժամանակի միջակայք սահմանելու համար օգտագործվում է Interval հատկությունը, որի արժեքը տրվում է միլիվայրկյաններով: Վայրկյանաշափը անջատելու համար Interval հատկությանը պետք է տալ 0 արժեք կամ False արժեք:

DriveListBox սկավառակների ցուցակ /անվանումը՝ Drive/: Այս էլեմենտը նախատեսված է սկավառակների, ցուցակների և ֆայլերի հետ աշխատանքի և արտացոլման համար: Այն հնարավորություն է տալիս բոլոր հասանելի սկավառակների և համակարգչային սարքերի ցուցակի արտացոլման համար և ապահովում նրանց ընտրելու հնարավորություն: Ամենահետաքրքիր պատահարը Change-ն է, որը գործում է տվյալների կրիչը փոխելու ժամանակ: Դատկություններից ամենից հաճախ օգտագործվում է Drive հատկությունը, որը վերադարձնում է ընտրված սկավառակը կամ սարքը:

FileListBox ֆայլերի ցուցակ /անվանումը՝ File/: Օգտագործվում է ֆայլերի ընտրման համար: Նա արտացոլում է ընթացիկ ցուցակի ֆայլերը, որոնցից կարելի է այն ընտրել: Գլխավոր պատահարը հանդիսանում է Click-ը, որը գործում է ցուցակում ֆայլի անունը ընտրելու ժամանակ: Քետաքրքրություն են ներկայացնում նաև Pathchange պատահարը, որը տեղի է ունենում ուղղու փոփոխություններու (path հատկություն) և PatternChange պատահարը ֆայլի տիպային ծեփի ընտրման-փոփոխման ժամանակ (Pattern հատ-

կություն): Այս դեկավարման էլեմենտը ունի շատ ընդհանուր հատկություններ, որոնք համընկնում են *ListBox*-ի հատկությունների հետ: Pattern հատկությունը թույլ է տալիս որոշելու այն ֆայլերի տիպերը, որոնք պետք է արտացոլվեն ցուցակում:

2. 6. Նախագծի պատուհան

Նախագծի պատուհանում արտացոլվում են հավելվածի բոլոր էլեմենտները՝ ֆորմաները, մոդուլները և այլն, որոնք խնդրավորված են ըստ կատեգորիաների: VB6.0-ում մշակվող բոլոր հավելվածները կոչվում են նախագծեր: Նախագիծը պարունակում է մի քանի խումբ բաղկացուցիչներ՝ ֆորմաներ, մոդուլներ և այլն: Բոլոր հավելվածները կառուցվում են մոդուլային սկզբունքով: Այդ պատճառով օբյեկտային կողը բաղկացած է ոչ թե մեկ մեջ ֆայլից, այլ մի քանի մասերից:

Որպեսզի պահպանվի նոր մշակված նախագիծը, անհրաժեշտ է այն նշել նախագծի պատուհանում և գլխավոր մենյուից ընտրել **File\Save Project** կամ **File\Save Project AS** ենթամենյունները: Նախագծում նոր էլեմենտ ավելացնելու համար հարկավոր է ընտրել **Project\Add**, ջնջելու համար՝ **Project\Remove** ենթամենյունները:

Նախագծի պատուհանի պարունակությունը պահպանվում է հատուկ ֆայլում, որն ունի VBP ընդլայնումը և պարունակում է էլեմենտների ցուցակ, որոնք պետք է թերթավորել մշակման միջավայրում: Եթե մի քանի նախագծեր միավորվում են մեկ խմբի մեջ, ապա պահպանվում է VBG ընդլայնումով ֆայլում:

2. 7. Կոդի պատուհան

VB6.0 –ի գործարկումից հետո այս պատուհանը անմիջապես չի երևում: Այստեղ կատարվում է դրված խնդրի ծրագրի /ծրագրային կոդի/ մշակումը: Կոդը VB6.0-ում բաժանված է գործընթացների, որոնք անմիջականորեն կապված են որոշակի դեկավարման էլեմենտների հետ: Եթե կատարենք մկնիկի կրկնակի հարված Form1 պատուհանի կամ որևէ դեկավարման էլեմենտի վրա, ապա կգործարկվի **Project-Form1[Code]** – ի պատուհանը: Ծրագրի մուտքագրումը իրականացվում է որպես սովորական

տեքստ: Այստեղ նույնպես գործում են ֆորմատավորման հրամանները, որոնցից են՝ պատճենման (Ctrl+C), տեղափոխման (Ctrl+X) և տեղադրման (Ctrl+V) գործընթացները: Մուտքագրվող ծրագրի խմբագրման ռեժիմը, օրինակ՝ տեքստի տառատեսակի ձևը և չափը, կարելի է կարգավորել գլխավոր մեջյուի Tools\Options-Editor Format ճանապարհով գործարկված պատուհանի օգնությամբ:

2. 8. Ծրագրավորումը «Visual Basic»-ում

Visual Basic-ում կողք բաղկացած է օպերատորներից: Բարդ օպերատորների գրառումն ու լուսաբանումը մատչելի դարձնելու համար օգտագործվում է ընդգծման նշան (_): Ծրագրի տողի առավելագույն երկարությունը կարող է լինել 1023 նիշ, ընդ որում 10 ընդգծման նշանից ոչ ավելի: Ի տարբերություն փոփոխականի հաստատումը իր արժեքը ստանում է ծրագրի ալգորիթմի նշական ժամանակ, ընդ որում հետազայում այն ենթակա չէ փոփոխման: Դաստատումի հայտարարումն ուղեկցվում է նրա արժեքի գրանցմամբ և ունի հետևյալ շարահյուսությունը (սինտաքսիսը):

[Public/Private/dim] Const Դաստատումի Անվանումը [As Տիպի Անվանումը] = արժեք:

Փոփոխականի հայտարարման կարգը ունի հետևյալ տեսքը.

Public/Private/dim փոփոխականի Անվանումը [As Տիպի Անվանումը]:

Եթե տիպը չի հայտարարվում, ապա վերագրվում է Variant տիպ:

Կախված պարունակությունից լինում են տարրեր տեսակի փոփոխականներ: Visual Basic-ը պաշտպանում է (սատարում է) հետևյալ տիպի փոփոխականները Boolean-տրամաբանական արժեք, Byte-մեկ բայտանոց ամբողջ թիվ, Integer-ամբողջ թիվ, Long- կրկնակի երկարությամբ ամբողջ թիվ, Single-սահող ստորակետով թիվ, Double-կրկնակի ճշտությամբ սահող ստորակետով թիվ, Currency-դրամական մեծություն, Decimal-տասնորդական թիվ, Date-ամսաթիվ/ժամանակ, String-տողային փոփոխական (կարող են լինել և՝ հաստատում, և՝ փոփոխական երկարությամբ), Object-օբյեկտային փոփոխական, Variant- չորոշված տիպով փոփոխական:

Փոփոխականները նախապես հայտարարվում են, այնուհետև որոշվում:

Ծրագրավորման շատ լեզուներում բոլոր օգտագործվող փոփոխականները պետք է հայտարարվեն: Այդ պրոցեդուրայով ծրագրավորման համակարգում հաղորդվում է փոփոխականի անունը և տիպը, որը կարևոր է հիշողության մեջ նրան տեղ հատկացնելու առումով:

Dim փոփոխականի անուն [As տվյալների_տիպ]:

Փոփոխականի անունը կարելի է ընտրել կամայականորեն, բայց այդ ժամանակ պետք է պահպանել հետևյալ կանոնները՝

- Փոփոխականի անունը պետք է սկսվի տառով,
- Անվան առավելագույն երկարությունը՝ 255 նիշ,
- Անունը չի կարող պարունակել %, &, !, #, @, .. \$ նշանները,
- Անունները կարող են պարունակել տառեր, թվեր և ընդգծման սիմվոլներ (),
- Անունը չի կարող լինել բանալիային, ամրակցված (ռեզերվացված) բառ (օրինակ Print, empty, ByRef, ByVal, Private, Time, For, Me, On և այլն):

Հայտարարման ժամանակ տվյալների տիպը նշելը պարտադիր չէ: Տվյալների տիպը հայտարարման ժամանակ կարող է դրվել՝ ուղղակի փոփոխականի անվանը ավելացնելով տիպի նշանը:

Փոփոխականի տիպը	Նշանը	Օրինակ
Integer	%	Counter%
Long	&	NR&
Single	!	Result!
Double	#	Number#
Currency	@	Summa@
String	\$	FirstName\$

Օրինակ՝ Dim FirstName\$, Dim Price@, Dim Counter%:

Ոչ ակնհայտ հայտարարման ժամանակ միշտ փոփոխականի անվան կողքին պետք է նշել համապատասխան տիպի նշանը: Ակնհայտ հայտարարման ժամանակ դա պետք չէ: Օրինակ՝

Dim price As currency

Price=523

Նույնը ոչ ակնհայտ հայտարարման ժամանակ կունենա հետևյալ տեսքը.

price@=523

Variant տիպը քամելեն տիպ է: Նա որոշում է տվյալների տիպը՝ կախված պարունակությունից: Օրինակ, եթե պարունակությունը 5 է, ընդունում է Integer տիպ, եթե 1.2՝ ստանում է Double, եթե տեքստ՝ String: Ծրագրի կատարման ժամանակ այն փոխում է տիպը:

Չնայած այն բանին, որ անվանումների վիճ, ըստ եռթյան, որևէ եական սահմանափակումներ չեն դրվում, ցանկալի է, որ կիրառվի որոշակի դասակարգում, որը կնպաստի ծրագրի ընթեռնելիությանը: Այդ նպատակով օբյեկտի անվանումը սկսվում է կարծ նախամասնիկով, որը տվյալ դասի օբյեկտների համար հանդիսանում է հաստատուն և ցույց է տալիս տեսանելիության դաշտը (scope): Նախամասնիկից հետո դրվում է օբյեկտի անվանումը, ընդ որում առաջին տարը գրվում է մեծատառով: Եթե անվանումը բաղկացած է մի բանի բառից, ապա դրանցից յուրաքանչյուրը պետք է սկսվի մեծատառով: Նշենք փոփոխականների տիպերն արտահայտող նախամասնիկները Boolean-blн, Byte-byт, collection-col, Currency-cur, Date(Time)-dtm, Double-dbl, Error-err, Integer-int, Long-Int, Object-obj, Single-sng, String-str, Variant-vnt, շահագործողի կողմից որոշվող տիպ-սետ: Օրինակ IntNum1 աս integer, strName աս string և այլն:

Ղեկավարման էլեմենտներ: Ղեկավարման էլեմենտները կարելի է ոիտել որպես փոփոխականների անալոգներ: Ղեկավարման էլեմենտների անվանումների համար նույնականացնելու մեջ պահանջվում է օգտագործել նախամասնիկներ (պրեֆիքսներ) ImageList-ils, Image-img, CommandButton-cmd, Label-lbl, PictureBox-pic, PictureClip-clp, FileListBox-fil, DBGrid-dbgrd, Dbcombo-dbcbo, Data-dat, Shape-shp, Form frm, ProgressBar-prg, Hscrollbar-hsb, ComboBox-cbo, CheckBox-chk, DriveListBox-drv, Line-lin, ListView-lvw, ListBox-lst, MAPIMessages-mpp, MAPISession-mps, MCI-mci, MenuItem-mnu, Gauage-gau, Mstab/SSTab-mst, OLE-ole, Frame-fra, TabStrip-tab, RichTextBox-rtf, Slider-sld, CommandDialog-dlg, StatusBar-sta, Control-ctr, TreeView-tre, ToolBar-tlb, Grid-grd, TextBox-txt, MDI-Form-mdi, VscrollBar-vsб, DirListBox-dir, Timer-tmr:

Փոփոխականների բացահայտ նկարագրության դեպքում օգտագործվում է Explicit օպցիան: VB-ում որպեսզի նկարագրված փոփոխականները ավտոմատ կերպով չստանան variant տիպ, կարելի է օգտագործել խմբակային նկարագրություն DefType օպերատորի օգնությամբ հետևյալ ֆորմատով DefType սկզբնատարը [–Վերջնատառը]:

VB -ում գոյություն ունի որոշման երեք տիրույթ.

- լոկալ փոփոխականները հասանելի են միայն տվյալ պրոցեդուրային,
- բեռնարկղ փոփոխականները հասանելի են միայն տվյալ ֆորմային, մոդուլին կամ դասին,
- գլոբալ փոփոխականները հասանելի են ամբողջ նախագծին:

Բեռնարկղեր և գլոբալ փոփոխականները նկարագրվում են (General)(Declaration) բաժնում: Վերջինիս համար նկարագրման մեջ dim բառին փոխարինում է public բառը:

Ստատիկ փոփոխականների եռթյունը կայանում է նրանում, որ պրոցեդուրան ավարտելուց հետո այն պահպանում է իր արժեքը, քանի դեռ պրոցեդուրան գտնվում է իշխողությունում: Ունի հետևյալ ֆորմատը՝ static փոփոխականի անուն [As փոփոխականի տիպ]:

Որպեսզի պրոցեդուրայի բոլոր լոկալ փոփոխականները հայտարարվեն Static, պետք է օգտագործել հետևյալ ֆորմատը՝ Static Sub / Function / Property ([արգումենտներ]):

Զանգվածը որոշակի տիպի էլեմենտների հավաքածու է, որոնցից յուրաքանչյուրն ունի հերթական ինդեքսային համար: Գոյություն ունեն ստատիկ և դինամիկ զանգվածներ: Ստատիկ զանգվածների սահմանները որոշվում են նախագծի կազմնան ժամանակ և փոփոխության ենթակա չեն, իսկ դինամիկ զանգվածների սահմանները կարող են ծրագրի կատարման ընթացքում փոփոխվել: VB-ում զանգվածների ինդեքսավորումը միշտ սկսվում է 0-ից, իսկ եթե ուզում ենք, որ սկսվի 1-ից, ապա (General) (Declaration) բաժնում պետք է կիրառել option base 1 օպերատորը: Դինամիկ զանգվածները ստեղծվում են երկու փուլով: Ակզրում այն սահմանվում է (General)(Declaration) բաժնում, առանց նշելու չափողականությունը, այնուհետև ReDim օպերատորի միջոցով որոշվում է զանգվածի փաստացի չափողականությունը: Չափագրողի կողմից սահմանվող փոփոխականների տիպը ունի հետևյալ ֆորմատը՝

[Private / Public] Type Տիպի_Անվանումը

էլեմենտ_1([[չափողականությունը]]) As տիպ

էլեմենտ_2([[չափողականությունը]]) As տիպ

End Type:

VB-ում կատարվող գործողությունները բաժանվում են ստանդարտ հավաքածուների երեք խմբերի՝ թվաբանական, հարա-

բերության և տրամաբանական և գործում են մաթեմատիկայում գործող կանոնների առավելությունների համապատասխան:

VB-ում միևնույն օբյեկտի անվան բազմակի օգտագործման կրկնությունից խուսափելու համար նախատեսված են With օբյեկտի_անուն, End With օպերատորները, զանգվածի կամ ընտանիքի յորաքանչյուր էլեմենտի համար For Each-Next օպերատորները:

2. 9. Պատահարադեկավարվող ծրագրավորում

Պատահարի վրա կողմնորոշումը Visual Basic-ում Windows լրացումների (հավելվածների) ստեղծման միջուկն է: Microsoft Windows-ը համակարգ է, որը հիմնված է հաղորդումների և իրադարձությունների վրա: Դա նշանակում է, որ Windows-ում յուրաքանչյուր գործողություն առաջ է բերում իրադարձություն, որը հաղորդագրության տեսքով փոխանցվում է հավելվածին: Դա վելվածը վերլուծում է հաղորդագրությունը և կատարում է համապատասխան գործողությունը: Նման գործողությունների մշակման հիմք է հանդիսանում հենց Windows-ի գաղափարը: Windows-ի մշակվայրում շահագործողը կարող է աշխատել միաժամանակ մի քանի հավելվածների հետ: Ոչ մի հավելված չի կարող գործել ինքնուրույն չհամագործակցելով այլ հավելվածների և օպերացիոն համակարգերի հետ: Վերադաս ատյանը (ինստանցիան) պետք է նրան հաղորդի, թե ինչ է կատարվում, և միայն այդ ժամանակ է հավելվածը արձագանքում նրան: Visual Basic-ի օգնությամբ ստեղծվող հավելվածները նույնպես աշխատում են այդ սկզբունքով: Բայց այդ ժամանակ Visual Basic համակարգը իր վրա է վերցնում աշխատանքի մի մասը: Նա «բռնում է» հաղորդագրությունը և փոխանցում այն համապատասխան օբյեկտին (օրինակ սեղմակին), որտեղ այնուհետև «կանչում է» համապատասխան իրադարձություն (օրինակ Click պատահարը):

Ծրագրային կողի կատարման համար միշտ անհրաժեշտ է իրադարձություն: Դա Visual Basic-ում հավելվածների ստեղծման կարևորագույն կանոններից մեկն է: Ոչ մի կող չի կատարվում առանց իրադարձության: Դեկավարման գոտու օբյեկտները կոչվում են դեկավարման էլեմենտներ (controls): Յուրաքանչյուր

օբյեկտ բնութագրվում է որոշակի պարամետրերով, որը կարելի է տրոհել երեք կատեգորիայի՝ պատահար, մերոդ և հատկություն: Օբյեկտները միավորվում են դասերում: Նույն դասին պատկանում են այն օբյեկտները, որոնք ունեն հատկությունների, մերոդների և պատահարների ընդհանրություն: Պատահարները կապված են շահագործողի որոշակի գործողությունների հետ և կարող են կոչվել Visual Basic-ի կող: Մերոդները օբյեկտի աշխատանքային օպերատորներն են: Օրինակ՝ Move մեթոդը թույլ է տալիս տեղափոխել դեկավարնան էլեմենտը նշված դիրքը: Հատկությունը պատասխանում է օբյեկտի արտաքին տեսքի և վարքի համար:

2.10. Պրոցեդուրաներ և ֆունկցիաներ

Ըստ եռթյան պրոցեդուրան ենթածրագիր է: Այն սկսվում է Sub և ավարտվում End օպերատորով, որոնց միջև տեղադրվում է կողը: Պատահարների մշակումները իրականացվում են որպես պրոցեդուրաներ: Պատահարը մշակող պրոցեդուրայի անվանումը բաղկացած է օբյեկտի և պատահարի անվանումների համակցումից: Կարելի է նաև ստեղծել սեփական պրոցեդուրաներ, այսպես կոչված ընդհանուր պրոցեդուրաներ: Դրա համար պետք է նշնել (General)(Declaration) բաժին և մտցնել Sub և պրոցեդուրայի անուն: Ֆունկցիան կառուցվում է ծիշտ այնպես, ինչպես և պրոցեդուրան: Սակայն գոյություն ունի մեկ տարրերություն, ինչպես մաթեմատիկայում, ֆունկցիայի աշխատանքի արդյունքը հանդիսանում է վերադարձվող արժեքը: Ֆունկցիայի որոշնան համար օգտագործվում է Function բանալիհային բառ: Պրոցեդուրայի վերջի End Sub-ի փոխարեն գրվում է End Function:

Օրինակ. Function NDS (Netto as currency, Percent as single)
as currency

```

NDS= Netto*Percent
End Function
Private sub command1_click()
Dim Tax as currency
...
Tax = NDS(100, 0.15)
...
End sub

```

Պատահարների մշակման ստանդարտ պրոցեդուրաներում
արգումենտները Visual Basic-ն ինքն է տեղադրում: Օրինակ՝
Private Sub Command1_MouseMove(Button As Integer, Shift As
Integer, X As Single, Y As Single)
...
End Sub

2. 11. VB-ում հաջախ օգտագործվող ֆունկցիաները

VB-ում գոյություն ունեն բազմաթիվ ներկառուցված
ֆունկցիաներ: Նշենք դրանցից մի քանիսը Abs(), Array(), Asc(),
Atn(), Chr(), Cos(), Date(), Time() և այլն: Գոյություն ունեն նաև
տիպերի ձևափոխման ֆունկցիաներ: Այս ֆունկցիաների միջոցով
կարելի է մի տիպի արտահայտությունը ձևափոխել մեկ այլ տիպի:
Boolean=Cbool (արտահայտություն), Byte=Cbyte (արտահայտու-
թյուն), Currency=Ccur(արտահայտություն), Date=Cdate (արտա-
հայտություն), Double=CDbl (արտահայտություն), Decimal=Cdec
(արտահայտություն), Integer=Cint (արտահայտություն), Long=Clngl
(արտահայտություն), Single=CSng (արտահայտություն),
String=CStr (արտահայտություն), Variant=Cvar (արտահայտու-
թյուն), Variant=CVErr (արտահայտություն).

Fix և Int ֆունկցիաները ընդամենը հատում են արտահայտու-
թյան կոտորակային մասը, իսկ Cint ֆունկցիան կլորացնում է այն:

Date - վերադարձնում է ընթացիկ համակարգային ամսաթիվը:

DateAdd(interval, count, date)- ավելացնում է կամ հանում է
նշված ամսաթվին նշված ինտերվալը:

DateDiff(interval, date1, date2[, firstdayofweek{, firstweekofyear}]) - վերադարձնում է ժամանակի տրված ինտեր-
վալի թիվը երկու նշված ամսաթվերի միջև:

EOF (ֆայլի_համարը) - վերադարձնում է True, հակառակ
դեպքում False:

Error (սխալի_համարը) - վերադարձնում է տվյալ արժեքին
համապատասխան սխալի հայտարարումը:

Exp (թիվ) - վերադարձնում է թիվը բարձրացրած համապա-
տասխան աստիճան:

FileLen (ֆայլի_անունը) - վերադարձնում է տվյալ ֆայլի չափը բայթերով:

Fix (թիվ) - վերադարձնում է թվի ամբողջ մասը: Բացասական թվերի դեպքում վերադարձնում է տվյալ թվից մեծ կամ հավասար թիվ:

FreeFile [(շարքի_համարը)] - վերադարձնում է հաջորդ կանալի ազատ համարը, որը կարելի է օգտագործել:

Inport (թիվ, [#]ֆայլի_համարը) - վերադարձնում է ֆայլից կարդացվող սիմվոլները, որտեղ թիվը կարդացվող սիմվոլների քանակն է, ֆայլի_համարը՝ կանալի համարը:

Int (թիվ) - վերադարձնում է թվի ամբողջ մասը: Բացասական թվերի դեպքում վերադարձնում է արգումենտից փոքր կամ հավասար թիվ:

IsArray (արտահայտություն) - վերադարձնում է True, եթե նշված փոփոխականը զանգված է, հակառակ դեպքում False:

IsMissing (արգումենտ) - ստուգում է, թե արդյոք պրոցեդուրային փոխանցվում է ոչ պարտադիր արգումենտ, թե՝ ոչ:

IsNull (արտահայտություն) - վերադարձնում է True, եթե արգումենտը պարունակում է զրո, False՝ հակառակ դեպքում:

IsNumeric (արտահայտություն) - վերադարձնում է True, եթե արգումենտը թվային մեջություն է, False՝ հակառակ դեպքում:

Lbound (զանգված, չափը) - վերադարձնում է տվյալ զանգվածի ամենափոքր ինդեքսը:

Lcase (տող) - բոլոր տառերը ձևափոխում է փոքրատառերի:

Left (տող, երկարություն) - վերադարձնում է նշված քանակությամբ սիմվոլներ՝ վերցրած տողի ծախս ծայրակետից:

Len (տող, փոփոխականի_անուն) - վերադարձնում է տողի կամ փոփոխականի սիմվոլների քանակը:

Lock (ֆայլի_համարը) - վերադարձնում է բաց ֆայլի կառդալու/գրելու ընթացիկ վիճակը:

Loc (ֆայլի_համարը) - վերադարձնում է բաց ֆայլի չափը բայթերով:

Ltrim, Rtrim, Trim - հեռացնում է բացատմերը ծախսից, աջից և երկու կողմերից:

Mid (տող, սկիզբ, [երկարություն]) - վերադարձնում է տողի նշված մասը, որտեղ տողը տեքստային արտահայտություն է, սկիզբը ցույց է տալիս, թե որտեղից է սկսվելու վերադարձվող

մասը, Երկարությունը պարունակում է վերադարձվող մասի Երկարությունը:

RGBColor (գույն) - վերադարձնում է RGB գունային կոդը 0-15 միջակայքից:

RGB (կարմիր, կանաչ, կապույտ) - վերադարձնում է RGB գունային կոդը 0-255 միջակայքից:

Right (տող, Երկարություն) - վերադարձնում է նշված քանակությամբ սիմվոլներ վերցրած տողի աջ ծայրակետից:

Rnd ([թիվ]) - վերադարձնում է 0-1 միջակայքի պատահական թիվ

Round (արտահայտություն [, տասնորդական_թվի_քանակ]) – վերադարձնում է տրված քանակությամբ տասնորդական թիվ՝ կլորացված տեսքով:

Sgn (թիվ) - վերադարձնում է թվի նշանը:

Space (թիվ) - վերադարձնում նշված քանակությամբ բացատներ:

Spc (թիվ) - #print և print մեթոդի դեպքում դուրս է բերում տրված քանակությամբ բացատներ:

Str (թիվ) – թվային արտահայտությունը ծևափոխում է տեքստայինի:

Tab (սյան_դիրք)- #print և print մեթոդի դեպքում ցույց է տալիս ելքի դիրքը:

Time () - վերադարձնում է համակարգի ընթացիկ ժամանակը:

Timer – վերադարձնում է վայրկյանների քանակը:

TypeName (փոփոխականի_անուն) – վերադարձնում է տվյալ փոփոխականի տիպը:

Ubound (զանգված, չափ) – վերադարձնում է զանգվածի ինդեքսի մեծագույն արժեքը:

Ucase (տող) - բոլոր տառերը ծևափոխում է մեծատառերի:

Val (տող) - վերադարձնում է տողի պարունակության թվային արժեքը:

VarType (փոփոխական) – որոշում է, թե variant տիպը ինչ փոփոխականի տակ է հանդես գալիս:

Օրինակ, օգտվելով DateDiff և DateAdd ֆունկցիաներից որոշել, թե մինչ այդ պահը շահագործողը քանի վայրկյան է ապրել և քանի՞ վայրկյան կապրի և 2 շաբաթվա ընթացքում:

```

Private Sub Command1_Click()
    Dim bd As Date
    Dim y As String
    bd = InputBox("Enter your birth day", "bd")
    Text1.Text = DateDiff("s", bd, Date)
    y = DateAdd("ww", 2, Date)
    Text2.Text = DateDiff("s", Date, y)
End Sub

```

2. 12. «Visual Basic» –ում ներդրված ֆինանսական ֆունկցիաները

Visual Basic 6.0-ն ունի ներդրված ֆինանսական ֆունկցիաների մի ամբողջ խումբ, որոնց օգնությամբ շահագործողը հնարավորություն է ստանում իրականացնելու մի շարք ֆինանսական հաշվարկներ: Ֆինանսական ֆունկցիաները բաժանվում են երեք հիմնական խմբերի:

1. ամորտիզացիայի,
2. կանոնավոր կուտակումների /վճարների հաշվարկման/,
3. ֆինանսական հոսքերի հաշվարկման:

Դիտարկենք այս ֆունկցիաների խմբերը և դրանց հատկությունները:

Ամորտիզացիոն հաշվարկների ֆունկցիաներ

Ամորտիզացիոն հաշվարկների ֆունկցիաները հաշվապահական գործընթացներում օգտագործվում են որոշակի ժամանակահատվածում դրամական արտահայտությամբ հիմնական միջոցների արժեզրկման համար: Օրինակ, երկակի հաշվառնան մեթոդով ամորտիզացիոն հաշվարկը կատարվում է հետևյալ բանաձևով՝

**Ամորտիզացիա /Ժամանակահատված = ((Սկզբնական
արժեք - մնացորդային արժեք)*2) / շահագործման
ժամանակահատված**

Visual Basic-ի գրադարանին պատկանող բոլոր ֆինանսական ֆունկցիաները ունեն անվանական արգումենտներ /named

arguments/: Անվանական արգումենտները պետք է գրվեն առանց սխալների: Դրանց օգնությամբ կարելի է կատարել պարամետրերի փոխանցում կանյալական կարգով՝ օպերատորի օգնությամբ շնորհելով արժեքներ յուրաքանչյուր առանցքային բնութագրին: Անվանական արգումենտների առկայությունը ֆինանսական ֆունկցիաներին տալիս է լրացնելի հնարավորություններ:

Դիտարկենք ամորտիզացիայի հաշվարկման համար նախատեսված ֆունկցիաները և դրանց հատկությունները:

	Ֆունկցիան	Նշանակությունը
1.	DDB (cost, salvage, life, period, [factor])	Ակտիվների ամորտիզացիայի աստիճանը որոշակի ժամանակահատվածի համար
2.	SYD (cost, salvage, life, period)	Գծային ամորտիզացիայի մեթոդով հաշված ամորտիզացիայի աստիճանը տրված ժամանակահատվածի համար
3.	SLN (cost, salvage, life)	Հավասարաչափ ամորտիզացիայի մեթոդով հաշված ամորտիզացիայի աստիճանը միավոր ժամանակահատվածում

Ամորտիզացիայի հաշվարկման համար նախատեսված ֆունկցիաների արգումենտները օժտված են հետևյալ հատկություններով.

	Արգումենտը	Տվյալի տեսակը	Նշանակությունը
1	Cost	Double	Գնահատվող ֆոնդի սկզբնական արժեքը
2	Salvage	Double	Գնահատվող ֆոնդի արժեքը շահագործման ժամանակահատվածի վերջում (մնացորդային արժեքը)
3	Life	Double	Գնահատվող ֆոնդի շահագործման ժամանակահատվածի չափը
4	Period	Double	Ցույց է տալիս այն ժամանակահատվածը, որի համար հաշվարկվում է ամորտիզացիայի աստիճանը
5	Factor	Variant	Ամորտիզացիայի հաշվարկի ցուցանիշ

Life և period արգումենտները պետք է տրվեն նույն չափման միավորներով /օր, ամիս-ամիս/:

Factor-ը ոչ պարտադիր անվանական արգումենտ է: Եթե այս արգումենտը չի գրված, ապա այն ընդունում է 2 արժեք, կիրառվում է գնահատվող ֆոնդի արժեքի իջեցման հաշվարկի երկակի մեթոդը:

Բոլոր արգումենտները պետք է ունենան դրական արժեք: Ներկայացնենք DDB ֆունկցիայի տեսքը իր արգումենտներով:

$$DDB = \text{period} * ((\text{cost} - \text{salvage}) * \text{factor}) / \text{life}$$

Օրինակ 1. 100.000 դրամ արժողությամբ ֆոնդի, որի շահագործման ժամանակահատվածը չի գերազանցում 4 տարին, 2 տարվա ժամանակահատվածի համար տարեկան 20% արժեքի նվազման դեպքում ամորտիզացիոն հաշվարկը կներկայացվի հետևյալ ձևով.

$$D = DDB(100000, 0, 4, 2, 1.2)$$

Օրինակ 2. Գծային կախվածության իջեցման մեթոդով ամորտիզացիայի հաշվարկը՝ SYD ֆունկցիայի կիրառմամբ:

Ընդունենք, որ ֆոնդի սկզբնական արժեքը (Cost) կազմում է 100.000 դրամ, մնացորդային (Salvage) արժեքը՝ 10.000 դրամ, հաշվարկային (Period) ժամանակահատվածը՝ 2 տարի, շահագործման (Life) ժամանակը՝ 12 տարի.

$$D_{per} = SYD(100000, 10000, 12, 2)$$

Օրինակ 3. Գծային կախվածության իջեցման մեթոդով ամորտիզացիայի հաշվարկը՝ SLN ֆունկցիայի կիրառմամբ: Ընդունենք, որ ֆոնդի սկզբնական արժեքը (Cost) կազմում է 100.000 դրամ, մնացորդային (Salvage) արժեքը՝ 10.000 դրամ, շահագործման (Life) ժամանակը՝ 12 տարի.

$$D_{yer} = SLN(100000, 10000, 12)$$

Կանոնավոր կուտակումների (վճարների) հաշվարկման ֆունկցիաներ

Կանոնավոր կուտակումները իրենցից ներկայացնում են ներդրումներից շահույթ կամ վարկի մարում: Ներկայացնենք կանոնավոր կուտակումների (վճարների) հաշվարկման համար նախատեսված ֆունկցիաները և դրանց հատկությունները:

	Ֆունկցիան	Նշանակությունը
1.	<code>FV (rate, nper, pmt[, pv[, type]])</code>	Հաստատված տոկոսադրույթի դեպում տրված վճարային ժամանակահատվածում վճարված գումարը:
2.	<code>Ipmt (rate, per, nper, pv[, fv[, type]])</code>	Հաշվարկված տոկոսադրույթի գումարը, որը վերաբերում է տվյալ վճարային ժամանակահատվածին:
3.	<code>Nper (rate, pmt, pv[, fv[, type]])</code>	Վճարային ժամանակահատվածի քանակի հաշվարկում:
4.	<code>Pmt (rate, nper, pv[, fv[, type]])</code>	Հաշվարկված հաստատուն գումարների վճարումը որոշակի ժամանակահատվածի համար:
5.	<code>PPmt (rate, per, nper, pv[, fv[, type]])</code>	Հաշվարկված վճարումների այն մասը, որը վերաբերում է հիմնական գումարի վճարմանը տվյալ վճարային ժամանակահատվածում:
6.	<code>PV (rate, nper, pmt[, fv[, type]])</code>	Ակզենական /ընթացիկ/ գումարի հաշվարկում:
7.	<code>Rate (nper, pmt, pv[, fv[, type[, guess]]])</code>	Հաշվարկային դրույթի հաշվարկում:

Կանոնավոր կուտակումների (վճարների) հաշվարկման համար նախատեսված ֆունկցիաների արգումենտները օժտված են հետևյալ հատկություններով.

1.	rate	Double	Հաշվարկային ժամանակահատվածում տրված հաշվարկային դրույքաչփը
2.	nper	Integer	Դիմարկվող ժամանակահատվածում վճարումների տարբերության թիվը
3.	pmt	Double	Հաշվարկային ժամանակահատվածում վճարումների չափը
4.	per	Double	Պարբերության թիվը
5.	pv	Variant	Ընթացիկ պահին վճարումների չափը: Եթե արգումենտը բացակայում է, ապա արժեքը ենթադրվում է 0
6.	[Fv]	Variant	Վճարումների վերջնական արդյունքը: Եթե չկա, ապա ընդունվում է 0 արժեքը
7.	[Type]	Variant	Կատարվող վճարումների ռեժիմը: 0 արժեքը նշանակում է, որ վճարումները կատարվում են վերջում, 1 արժեքը՝ ժամանակաշրջանի սկզբում
8.	[Guess]	Variant	Վերջնական արդյունարար հաշվարկի սխալների չափը

Rate և nper արգումենտները հաշվարկներում ներկայացվում են միևնույն ժամանակային միավորով: Բոլոր արգումենտների համար վճարումները (դեպոզիտ ներդրումները) ներկայացվում են բացասական, իսկ ստացվածները (դիվիդենդները)՝ դրական արժեքով:

Օրինակ 1. FV ֆունկցիայի կիրառմամբ հաշվենք 12 ամսվա ընթացքում ամսական 600 դրամ վճարումով՝ առանց ընթացիկ մուտքերի տարեկան 12% հաշվարկաչփով կուտակումները:

$$S(12)=FV(12/(100*12), 12, -600, 0, 1)$$

Օրինակ 2. Ipmt ֆունկցիայի կիրառմամբ հաշվենք 30 տարով, տարեկան 8% տոկոսադրույքով 500.000 դրամ ներդրված գումարի վճարման մասը երրորդ տարվա առաջին ամսում.

$$S(25)=Ipmt(8/(100*12), 25, 360, 500000, 0, 1)$$

Օրինակ 3. NPer ֆունկցիայի կիրառմամբ, հաշվենք հաշվում եղած 100.000 դրամ գումարի 5% տոկոսադրույթով, ամսական 1000 դրամ ժախսով ներդրված գումարի սպառման ժամանակահատվածը:

$$T=Nper(0,05/12,-1000,100000,0,1)$$

Օրինակ 4. Pmt ֆունկցիայի կիրառմամբ հաշվենք 30 տարով, տարեկան 8 տոկոսադրույթով ներդրված 100.000 դրամ գումարի ամսական վճարումները:

$$S(t)=Pmt(8/(100*12),360,100000,0,1)$$

NPV և PV ֆունկցիաների տարրերությունն այն է, որ PV ֆունկցիան թույլ է տալիս գործընթացների գրանցումը ոչ միայն ժամանակաշրջանի վերջում, այլ նաև սկզբում:

Ֆինանսական հոսքերի հաշվարկման ֆունկցիաներ

Ներկայացնենք ֆինանսական հոսքերի հաշվարկման համար նախատեսված ֆունկցիաները և դրանց հատկությունները.

	Ֆունկցիան	Նշանակությունը
1.	IRR (values() [, guess])	Պարբերական հաջորդական գործընթացներում շահույթի նորմայի հաշվարկը
2.	MIRR (values(), finance rate, reinvest rate)	Շահույթի նորմայի հաշվարկը պարբերական ֆինանսական գործընթացների համար, եթե չեն համընկնում շահույթի դրույցը և ներդրումների ժախսերը
3.	NPV (rate, values())	Ֆինանսական գործընթացների պոյեկցիայում ընթացիկ պահի սալդոյի հաշվարկումը

Դիտարկենք ֆինանսական հոսքերի հաշվարկման ֆունկցիաների արգումենտների հատկությունները.

	Արգու-մենտը	Տվյալի տեսակը	Նշանակությունը
1.	values()	Double	Դրամական հոսքերի զանգված, որը պետք է պարունակի մեկ բացասական (ժախս) և մեկ դրական (շահույթ, մուտք) սահմանային արժեք
2.	rate	Double	Ակտիվների և պարտավորությունների վերագնահատման արագությունը
3.	finanse rate	Double	Տրամադրված ֆինանսավորման դիմաց վճարումների տոկոսաչափը
4.	reinvest rate	Double	Գործառնություններից ստացված տոկոսաչափը
5.	[guess]	Variant	Վերջնական արդյունարար հաշվարկի սխալների չափը՝ որոշված IRR ֆունկցիայով: Եթե այս արգումենտը բացակայում է, ապա սխալները ընդունվում են 0.1-10%

Օրինակ 1. $N()$ չափանի զանգվածի միջոցով հինգ օպերացիաների օգնությամբ շահույթի նորմայի գնահատումը MIRR ֆունկցիայի օգնությամբ: Զանգվածի առաջին բացասական էլեմենտը համապատասխանում է ծրագրավորված բիզնեսի համար վերցրած վարկին: Մնացած 4 դրական արժեքը ունեցող էլեմենտները ցույց են տալիս հաջորդ 4 տարիների շահույթը: և և Re=ն ցույց են տալիս ներդրումների ֆինանսավորման չափը և գործառնությունից ստացված շահույթի չափը:

Dim e,Re,Npb

Dim N(5) As Double` հայտարարում է զանգվածը

e=0.1` սրվում է ճշտությունը 10%

Re=0,12

N(0)=-70000` ծախսերը

Values(1)=22000: Values(2)=25000` չորս տարվա շահույթը

Values(3)=28000: Values(4)=31000

Npb=MIRR(N(),e,Re)` շահույթի նորման:

Օրինակ 2. NPV ֆունկցիայի օգնությամբ ընթացիկ պահի օպերացիաների արդյունքների պրոյեկցիաների հաշվարկը: V() զանգվածի էլեմենտների դասավորությունը համապատասխանում է ֆինանսական գործառնությունների (վճարումներ և մուտքեր) հաջորդականությանը:

Dim e,Re,Npb

Dim N(5) As Double հայտարարում է զանգվածը

e=0.0625 շահույթի չափը

V(0)=-80000՝ բիզնեսի կազմակերպման ծախսերը

չորս տարվա շահույթը

Values(1)=24000: Values(2)=26000

Values(3)=28000: Values(4)=30000

Npb=NPV(e,V())՝ ընթացիկ պահի պրոյեկցիան:

Օրինակ 3. N() չափանի զանգվածի միջոցով հինգ օպերացիաների օգնությամբ շահույթի նորմայի գնահատումը IRR ֆունկցիայի օգնությամբ:

Dim e,Re,NorPb

Dim N(5) As Double հայտարարում է զանգվածը

e=0.1 տրվում է ճշտությունը 10%

N(0)=-70000 ծախսերը

չորս տարվա շահույթը

Values(1)=22000: Values(2)=25000

Values(3)=28000: Values(4)=31000

NorpB=IRR(N(),e)*100՝ շահույթի նորման:

IRR ֆունկցիան ստանում է վերջնական արդյունք բազմաթիվ իտերացիոն ցիկլերի շնորհիվ: Այդ ցիկլը սկսվում է guess սկզբնական նշանակությունից և շարունակվում է այնքան, քանի դեռ չի ստացվել արդյունքը 0.00001% ճշտությամբ: Եթե 20 ցիկլից հետո բավարար արդյունք չի ստացվում, ապա ֆունկցիան տալիս է սխալի մասին հաղորդագրություն:

Ֆինանսական ֆունկցիաների օգտագործման օրինակներ

- Ա կազմակերպությունը Բ բանկից 2 տարի ժամկետով, տարեկան 12% տոկոսադրույթով վերցնում է 1000000 դրամ երկարաժամկետ վարկ: Վարկի դիմաց վճարվող տոկոսները պետք է վճարել ամսական: Որոշել և տեքստային դաշտ դուրս բերել Ա կազմակերպության կողմից Բ բանկին վճարվելիք ամսական տոկոսավճարի չափը:

$$F1=Pmt(12/(100*12),24,1000000,0,1)$$

- Ա կազմակերպությունը Բ բանկից վերցնում է 2000000 դրամ՝ 3 տարով երկարաժամկետ վարկ: Վարկի դիմաց Ա կազմակերպությունը Բ բանկին ամսական վճարում է 85000 դրամ տոկոսավճար: Որոշել և տեքստային դաշտ դուրս բերել կազմակերպության կողմից վարկի դիմաց ամսական վճարվող տոկոսադրույթը:

$$F2=Rate(36,85000,2000000)$$

- Ա կազմակերպությունը Բ բանկից 2 տարի ժամկետով, տարեկան 12% տոկոսադրույթով վերցված վարկի ամսական ծախսը կազմում է 72000 դրամ: Որոշել և տեքստային դաշտ դուրս բերել Բ բանկի կողմից Ա կազմակերպությանը տրված վարկի սկզբնական գումարը:

$$F3=PV(12,24,72000)$$

- Ա կազմակերպությունը գնել է 10 տարվա օգտակար ծառայություն ունեցող՝ 1000000 սկզբնական արժեքով հիմնական միջոց: Հիմնական միջոցի շահագործման վերջում դրա մնացորդային արժեքը պետք է կազմի 50000: Որոշել ամրութիւնացիայի գումարը առաջիկա 2 տարիների համար՝ ամորտիզացիայի տարեկան տոկոսադրույթը ընդունելով 13%:

$$F4=DDB(1000000,50000,10,2,1.3)$$

- Կազմակերպությունը գնել է 2000000 սկզբնական արժեքով, 10 տարվա օգտակար ծառայություն ունեցող հիմնական միջոց, որը 3 տարի հետո պետք է վերավաճառվի: Օգտակար ծառայության ժամկետի վերջում հիմնական միջոցի

մնացորդային արժեքը պետք է կազմի 75000: Ծուկայական գնահատումները ցույց են տվել, որ 3 տարի հետո հիմնական միջոցի արժեգրկումից կորուստները կկազմեն 300000: Որոշել և տեքստային դաշտ դուրս բերել 3 տարի հետո (վերավաճառքի պահին) հիմնական միջոցի հաշվեկշռային արժեքը:

$$F5=2000000-SYD(2000000,75000,10,3)-300000$$

6. Գծային կախվածության իշեցման մեթոդով ամորտիզացիայի հաշվարկը SYD ֆունկցիայի կիրառմամբ: Ընդունենք, որ ֆոնդի սկզբնական արժեքը կազմում է 100000 դրամ, մնացորդային արժեքը՝ 10000 դրամ, հաշվարկային ժամանակահատվածը՝ 2 տարի, շահագործման ժամանակը՝ 12 տարի:

$$F6=SYD(100000,10000,12,2)$$

7. Գծային կախվածության իշեցման մեթոդով ամորտիզացիայի հաշվարկը SLN ֆունկցիայի կիրառմամբ: Ընդունենք, որ ֆոնդի սկզբնական արժեքը կազմում է 100000 դրամ, մնացորդային արժեքը՝ 10000 դրամ, շահագործման ժամանակը՝ 12 տարի:

$$F7=SLN(100000,10000,12)$$

8. FV ֆունկցիայի կիրառմամբ հաշվենք 12 ամսվա ընթացքում, ամսական 600 դրամ վճարումով առանց ընթացիկ նուտքերի, տարեկան 12% հաշվարկաչափով կուտակումները:

$$F8=FV(12/(100*12),12,-600,0,1)$$

9. Ipmt ֆունկցիայի կիրառմամբ հաշվենք 30 տարով տարեկան 8% տոկոսադրույցով, 500000 դրամ ներդրված գումարի վճարման մասը երրորդ տարվա առաջին ամսում:

$$F9=Ipmt(8/(100*12),25,360,500000,0,1)$$

10. Nper ֆունկցիայի կիրառմամբ, հաշվենք հաշվում եղած 100000 դրամ գումարի՝ 5% տոկոսադրույցով, ամսական 1000 դրամ ծախսով ներդրված գումարի սպառման ժամանակահատվածը:

$$F10=Nper(0.05/12,-1000,100000,0,1)$$

2. 13. Յրամանների կատարման կարգի դեկավարման օպերատորները

Visual Basic-ը ներկայացնում է մի շարք ֆունկցիաներ և օպերատորներ: Ամենից հաճախ օգտագործվում է IF...THEN օպերատորները, որոնք կարող են ունենալ պարզ՝ մեկտողանի կամ բարկային կառուցվածք:

Մեկ տողանին ունի հետևյալ կառուցվածքը.

IF Պայման Then օպերատոր [Else օպերատոր]:

Եթե IF-ից հետո պայմանը ճշմարիտ է, այսինքն՝ արդյունքը հավասար է True (Ճիշտ), կատարվում է Then-ից հետո նշված օպերատորը: Եթե արդյունքը հավասար է False-ի, ապա կատարվում է Else բանալիային բառից հետո եկող օպերատորը, եթե այդպիսին կա:

Օրինակ՝ IF x<9 Then Print "False!" Else Print "True"

Բլոկային կառուցվածքը ունի հետևյալ տեսքը.

IF պայման THEN

[օպերատորներ]

[ElseIf պայման THEN

[օպերատորներ 2]

ELSE

[օպերատորներ 3]]

EndIf

Սկզբունքորեն բլոկային գրվածքները ներկայացնում են նույնպիսի հնարավորություններ, ինչպիսին և մեկ տողանին: Բայց կախված պայմանից, եթե անհրաժեշտ է կատարել ոչ թե պարզ հրամաններ, այլ օպերատորների խումբ, պետք է օգտագործել բլոկային կառուցվածք (սինտաքսիս): Դա վերաբերում է և Else ճյուղին: Բացի դրանից, բլոկային կառուցվածքը Elseif-ի հետ թույլ է տալիս վերլուծել մի քանի պայմաններ:

Visual Basic-ի և մեկ ճյուղավորման օպերատոր է հանդիսանում Select Case օպերատորը, որը թույլ է տալիս կատարել մի քանի խումբ օպերատորներից մեկը՝ կախված պայմանի արժեքից:

Select Case ստուգվող_արտահայտություն

[case արժեք1

[օպերատորներ 1]]

[case արժեք 2

[օպերատորներ 2]]

[case else

[օպերատորներ 3]

End Select

Մի քանի օպերատորների բազմաթիվ անգամ կատարման համար նախատեսված են ցիկլեր: Visual Basic-ը առաջարկում է երկու կառուցվածքներ: For ... Next ցիկլը հնարավորություն է տալիս տեղադրելու ցիկլի անցումների թիվը, իսկ Do ... Loop - ը ավարտվում է տրված պայմանի կատարման ժամանակ:

For...Next ցիկլը հանդիսանում է ամենահին ու ամենապարզ կառուցվածքը, որն ունի հետևյալ տեսքը:

For հաշվիչ= սկզբնական _արժեք Տո վերջնական_արժեք [step= քայլ]

Օպերատորներ

Next [հաշվիչ]

Ցիկլի կատարման սկզբուն հաշվիչի արժեքը դրվում է սկզբնական արժեքը: Յուրաքանչյուր անցման ժամանակ հաշվիչ փոփոխականը մեծանում է 1-ով կամ էլ քայլի մեծությամբ: Եթե նա հասնում է կամ դառնում է մեծ (փոքր բացասական քայլի դեպքում) վերջնական արժեքից, ապա ցիկլը ավարտվում է: Սկզբնական և վերջնական արժեքների տարբերության հարաբերությունը քայլի մեծությանը կազմում է անցումների քանակը: Ցիկլից առանց պայմանի դուրս գալը իրականացվում է Exit For օպերատորի օգնությամբ:

Եթե անցումների քանակը պետք է կախված լինի պայմանից, օգտագործվում է Do... Loop ցիկլ: Կախված պայմանի դիրքից կիրառվում են օպերատորի երկու տարբերակ: Ցիկլ, որի պայմանը կատարվում է սկզբում և որի պայմանը կատարվում է վերջում: Եթե պայմանը ստուգվում է ցիկլի սկզբում, ապա այն երբեք չի կատարվում պայմանի չկատարման դեպքում: Եթե ստուգվումը տեղի է ունենում վերջում, ցիկլը կատարվում է նվազագույնը մեկ անգամ, անկախ նրանից, պայմանը կատարվում է, թե ոչ: Ցիկլի մարմինը կատարվում է անորոշ քանակով, քանի դեռ պայմանը չի կանչում ցիկլից դուրս գալը:

Ղեկավարման էլեմենտները ունեն հատկություններ, մեթոդներ և իրադարձություններ: Ղեկավարման էլեմենտների գանգվածները ննան են փոփոխականների սովորական զանգվածներին: Ինքը զանգվածը և նրա առանձին էլեմենտները տարբերվում են հնողերսներով: Եթե մենք դեկավարման էլեմենտը պատճենում ենք

փոխանակման բուֆեր, ինքնաբերաբար առաջանում է զանգված: Դեկավարնան էլեմենտների ավելացնան ուրիշ մեթոդ է controls ընտանիքի Add մեթոդի օգտագործումը, որն ունի հետևյալ կառուցվածքը:

`Object.Add(ProgID, name, container)`

որտեղ՝ Add-ը պարտադիր պարամետր է, որը իրենից ներկայացնում է օբյեկտ (controls ընտանիքից), որի մեջ ավելացվում է դեկավարնան էլեմենտ:

ProgID-ն՝ պարտադիր արգումենտ, դեկավարնան էլեմենտի տող-իդենտիֆիկատոր է: Դեկավարնան էլեմենտների մեծամասնության համար ProgID -ի արժեքը կարելի է որոշել Object Browser ուժիհատի օգնությամբ: Այդ արժեքը սովորաբար կազմվում է գրադարանի անունից և կոնկրետ դեկավարնան էլեմենտի դասի անունից:

Name-ը պարտադիր արգումենտ-տող է, որը նույնականացնում է ընտանիքի էլեմենտը:

Container-ը՝ ոչ պարտադիր արգումենտ է, հղում դեկավարնան էլեմենտի համար օբյեկտ-բեռնարկերին: Եթե այդ արգումենտը որոշված չէ կամ հավասար է NULL-ի, ապա կիրառվում է լրելայն այն բնունարկերը, որին պատկանում է controls ընտանիքը:

2. 14. Ստեղնաշարի և «մկնիկի» պատահարները

Տաք ստեղմեր: Windows-ում համարյա դեկավարնան էլեմենտների բոլոր գրություններում գոյություն ունի ընդգծված սիմվոլներ: [ALT] ստեղմի և համապատասխան սիմվոլի օգնությամբ կարելի է սկեռումը տեղափոխել դեկավարնան այդ էլեմենտի վրա կամ էլ կատարել համապատասխան գործողություն: Օրինակ [Alt+F] ստեղմի սեղմումով կարելի է կանչել File մենյուի հրամանը չօգտվելով մկնիկից: VB-ում Caption հատկության օգնությամբ կարելի է տալ “տաք ստեղմ” համապատասխան տարից առաջ տեղադրելով & (ամպերսանդ) սիմվոլ:

Պատահարներ, որոնք կապված են ստեղնաշարի հետ:

Ստեղնաշարից մուտքը կարող է մշակվել ոչ միայն Windows-ի, այլև դեկավարնան էլեմենտի կողմից: Դրա համար անհրաժեշտ է մշակել KeyDown, KeyPress և KeyUp պատահարները: KeyPress պատահարի մշակողի պրոցեդուրայում որպես պարամետր փոխանցվում է KeyAscii փոփոխական, որը պարունակում է այն

սիմվոլի Ascii կողը, որը սեղմվել է ստեղնաշարից: KeyAscii փոփոխականի արժեքը կարելի է ոչ միայն կարդալ, այլև տեղադրել: Յետևաբար հնարավոր է ստեղնաշարից մտցված սիմվոլը փոխարիթել ուրիշով: KeyAscii = Asc(Ucase(chr(keyAscii)))

KeyUp/KeyDown: Keypress պատահարը կանչվում է միայն ANSI (>127) կող ունեցող ստեղնի սեղմելու ժամանակ, իսկ նշիչի դեկավարման ստեղնի կամ էլ ֆունկցիոնալ ստեղների սեղմելու ժամանակ այդ պատահարը չի կանչվում: Այդ ստեղների սեղման մշակման համար պետք է օգտագործել KeyUp և KeyDown պատահարներ: Օրինակ

```
Private sub Text1_KeyDown(keycode As Integer, Shift AS Integer)
```

```
    MsgBox Keycode
```

```
End sub
```

KeyPreview: Սովորաբար ստեղնաշարի պատահարները ստեղծվում են (գեներացվում են) ակտիվ դեկավարման համար: Բայց սեղմելով որոշ ստեղներ, օրինակ ֆունկցիոնալ, պետք է մշակվի ֆորման և ոչ թե ակտիվ դեկավարման էլեմենտը: Յետևաբար ֆորման նույնականացնելու ժամանակ կարող է մշակել KeyDown, KeyPress և KeyUp պատահարներ, բայց դրա համար ձևապատճերի (ֆորմայի) KeyPreview հատկության արժեքը պետք է դնել True: Այդ դեպքում ստեղնաշարի բոլոր պատահարները պետք է մշակվեն ֆորմայի կողմից և հետո միայն ակտիվ դեկավարման տարրի կողմից:

SendKeys VB – ն թույլ է տալիս ոչ միայն ստեղնաշարից իրապես սեղմնել ստեղնը, այլև իմիտացնել այդպիսի սեղմումը: SendKeys օպերատորի օգնությամբ ստեղնաշարի իմիտացվող սեղմանն կողը գրվում է աննիջապես ստեղնաշարի բուհեր: Համակարգը այդ ժամանակ չի տարբերում այդպիսի մուտքը իրական մուտքից: Օրինակ SendKeys Ctrl[, wait]

Visual Basic-ում ֆոկուսի փոխանցման համար գոյություն ունի AppActivate օպերատոր AppActivate Title [, wait]:

Ակնիկ: Windows-ում մուտքի համար երկրորդ կարևոր սարքը մկնիկն է: Բոլոր գործողությունները, որոնք կատարվում են մկնիկի

միջոցով նույնպես կարելի է վերլուծել, որոնց համար առաջացվում են մի շաբթ պատահարներ՝ Click, DoubleClick, MouseDown, MouseUp, MouseMove, DragDrop, DragOver:

Ինֆորմացիայի ներածման համար օգտագործվում է **InputBox** ֆունկցիա, որի աշխատանքն իրականացվում է **InputBox** երկխոսության պատուհանի օգնությամբ: **InputBox** պատուհանը բաղկացած է չորս էլեմենտից՝

- վերնագրի տողից,
- ներմուծման իրավերքից (**Prompt**),
- ներմուծման դաշտն՝ իր լռելյայն առաջարկվող արժեքներով,
- երկու սեղմակ (OK և CANCEL):

InputBox ֆունկցիայի կանչն ունի հետևյալ տեսքը.

Վերադարձվող_Արժեք = **InputBox** (**Prompt** [, **Title**] [, **default**] [, **Xpos**] [, **Ypos**] [, **Helpfile**, **Context**])

Prompt – պարամետրը որոշում է այն տեքստը, որը պետք է գրվի երկխոսության պատուհանում որպես իրավեր,

Title – պարամետրը պատասխանատու է վերնագրի համար,

Default – պարամետրը ներմուծման տողում առաջարկվող լռելյայն արժեքն է,

Xpos, **Ypos** – ցույց են տալիս պատուհանի ծախսին անկյան կոորդինատները:

MessageBox պատուհանը նախատեսված է տարբեր հայտարարությունների և հաղորդումների համար. դրա կազմի մեջ մտնում են՝

- հաղորդվող տեքստը,
- վերնագիրը,
- պիկտոգրամը,
- սեղմակների հավաքածուն:

MessageBox – ը կարող է կանչվել թե՝ որպես ֆունկցիա, թե՝ որպես պրոցեդուրա: Պրոցեդուրան ունի հետևյալ տեսքը՝

MsgBox (**Prompt** [, **Buttons**] [, **Title**] [, **Helpfile**, **Context**])

Ֆունկցիան ունի հետևյալ տեսքը՝

Վերադարձվող_Արժեք = **MsgBox** (**Prompt** [, **Buttons**] [, **Title**] [, **Helpfile**, **Context**]):

Օրինակներ՝

- Խմբագրման սլաքների օգնությամբ պատկերի շարժումների կազմակերպումը:

```
Private Sub Form1_KeyDown(KeyCode As Integer , Shift As Integer)
If KeyCode=VbKeyDown Then
Shape1.Top=Shape1.Top+Shape1.Height
Elseif KeyCode=VbKeyLeft Then
Shape1.Left=Shape1.Left-Shape1.Width
Elseif KeyCode=VbKeyRight
Shape1.Left=Shape1.Left+Shape1.Width
Elseif KeyCode=VbKeyUp Then
Shape1.Top=Shape1.Top-Shape1.Height
Else
EndIf
End Sub
```

- Սկնիկի ծախ սեղմակի սեղմելու ժամանակ Pset մեթոդի օգնությամբ դրվում է կետ ֆորմայի այն մասում, որտեղ տվյալ պահին գտնվում է մկնիկի նշիչը: Այնուհետև նշիչը ստանում է խաչի տեսք և հրամանային սեղմակին հարվածելով գծում է ուղիղ գիծ մկնիկի սեղմած տեղից մինչև քայլ թռնչելու տեղը:

```
Private Sub Form1_Mouse_Down( Button As Integer , Shift As Integer, X As single, Y As single )
If Button =VbLeftButton Then
Form1.MousePointer=VbCrosshair
Form1.Pset(X,Y),Vbred
X0=X
Y0=Y
EndIf
End Sub
Private Sub Form1_Mouse_Up( Button As Integer , Shift As Integer,
X As single, Y As single )
If Button =VbLeftButton Then
Form1.MousePointer=VbCrosshair
X1=X
Y1=Y
EndIf
End Sub
```

```
Private Sub Command1_Click()
Line ( X0,Y0 ) - ( X1,Y1 )
End Sub
```

2. 15. Շահագործողական դասերի ստեղծումը

Նոր դասի նկարագրման համար անհրաժեշտ է սկզբից ստեղծել դասի մոդուլ: Դրա համար պետք է կատարել Insert/Modul, որի արդյունքում կստեղծվի Class1 անունով դասի մոդուլ: Այնուհետև այն կարենի է անվանափոխել Name հատկության արժեքը դարձնելով Ենթադրենք Rectangle: Դիտարկենք պարզ օրինակ, որը ստեղծում է դաս՝ Rectangle օբյեկտի համար:

```
Private IntA As Integer, IntB As Integer
Private sub class _initialize
IntA =1
IntB=1
End sub
Public Function Length() As Integer
Length=IntA
End Function
Public Function Width() As Integer
Width=IntB
End Function
Public Function Sguare As Integer
Sguare =IntA*IntB
End Function
Public sub CreateRectangle(ByVal IntLN As Integer, ByVal IntWd As
Integer)
If (IntLN OR IntWd) < = 0 Then
MsgBox "Ուղանկյան մակերեսը չպետք է հավասարվի 0-ի"
Else
IntA=IntLN
IntB=IntWd
End if
End sub
```

Տվյալ դասն ունի երկու փակ հատկություն՝ IntA և IntB, որոնք իրենցից ներկայացնում են ուղանկյան երկարությունը և լայնությունը և հինգ մեթոդներ:

- Class_Initialize
- Create _Rectangle
- IntLength
- IntWidth
- Sguare

Այժմ դիտարկենք ստեղծված դասի օբյեկտների գործնական օգտագործումը

```
Sub TestSub()
Dim object1 As class1
Dim object2 As class1
Dim IntWd As Integer
Dim IntLn As Integer
Dim intA As Integer
Dim intB As Integer
Dim intC As Integer
IntWd =5
IntLn =5
Set object1=New Class1
intA=object1.length
intB=object1.Width
Set object2=object1
Rect2.Createrectangle intLn, intWd
intC=Rect2.square
End sub
```

2. 16. Ֆայլերի հետ աշխատանքը

Ծրագրավորողին հաճախ հարկ է լինում աշխատել անմիջապես ֆայլերի հետ՝ կատարելով այնպիսի գործողություններ, ինչպիսին ֆայլերի կամ կատալոգների ավելացումը կամ հեռացումն է, տվյալների գրանցումը ֆայլում կամ ել դրանից տվյալների կարդալը: Գոյություն ունեն հետևյալ տիպի ֆայլեր՝

- Դաջորդական դիմելու ֆայլեր. սովորաբար տեքստային ֆայլեր են: Տվյալները ունեն ինչ-որ կառուցվածք՝ կազմակերպված բաժանիչներով: Կառուցվածքային միավորը, որպես օրենք, հանդիսանում է տողը:
- Կամայական ձևով դիմելու ֆայլեր. դրանք կառուցվածքավորված ֆայլեր են, օրինակ՝ տվյալների բազայի ֆայլերը գրվածքների տեսքով:
- Երկուական (բինար) ֆայլեր. դրանք բայթային դիմումով ֆայլեր են:

Ինֆորմացիայի պահպանման համար նախատեսված են ֆայլերի մշակման օպերատորներ, որոնք բույլ են տալիս կարդալ և պահպանել տվյալները տարբեր կրիչների վրա: Ֆայլերը բացելու և պահպանելու գործընթացը բաղկացած է մի քանի փուլերից:

- Ֆայլի դեսկրիպտորի ստացում (handle)
- Ֆայլի բացում
- Կարդալ և գրել տվյալները
- Ֆայլի փակում:

Ֆայլերի հետ աշխատելու համար պետք է հասկանալ, թե ինչպես է համակարգը կամ հավելվածը կապվում ֆայլի հետ: Դրա համար կա մուտք/ելք ուղի: Ֆայլը բացելու ժամանակ այն համապատասխանության մեջ է դրվում որոշակի համարի ուղու հետ: Այդպիսով, յուրաքանչյուր բացված ֆայլ ունի սեփական ուղի, որի օգնությամբ կարդացվում կամ գրանցվում են տվյալները: Նետեաբար տվյալների մուտք/ելքի համար ֆայլում նշանակություն ունի ոչ թե ֆայլի անունը, այլ ուղու համարը: Բացի դրանից, OЗ-ը պետք է տեղեկություն ունենա ազատ ուղիների մասին, որը կարող է օգտագործվել ֆայլի բացման համար:

Ֆայլերում ինֆորմացիայի մուտքի և ելքի կազմակերպումը

Նշենք ֆայլերի ղեկավարման որոշ ֆունկցիաներ և օպերատորներ.

Close	Get#	Name
Close#	GetAttr	Open
Dir	Input	Print #
Eof	Input#	Put#
FileAttr	Kill	Reset
FileCopy	LineInput#	Seek
FileDateTime	Loc	SetAttr
FreeFile	Lock	UnLock
FileLen	Lof	Write #

Ինչպես նշեցինք Visual Basic-ում կիրառվում են ֆայլերին դիմելու երեք տեսակներ:

1. Հաջորդական (sequential) տեքստային ֆայլեր գրելու և կարդալու համար:
2. Պատահական (կամայական -Random) տեքստ գրելու կամ կարդալու համար կամ ֆիքսած երկարությամբ գրվածքների, կառուցվածքավորված երկուական ֆայլեր կարդալու և գրելու համար:
3. Երկուական (Binary) կամայական կառուցվածքավորված ֆայլեր գրելու և կարդալու համար:

Հաղորդակցման ուղիների ստեղծման ժամանակ համակարգը պետք է իմանա, թե յուրաքանչյուր կոնկրետ ֆայլին ինչպիսի դիմում պետք է օգտագործել, և այդ ֆայլի տվյալների կառուցվածքն ինչպիսին է:

1-ի ժամանակ կամայական ինֆորմացիա կարդացվում կամ պահպանվում է տողերով տեքստի տեսքով: Տեքստում կարող են գտնվել տողի փոխանցման սիմվոլ (`vbcrlf` կամ `chr(13)&chr(10)`) կամ տարրույթոր (`vbtabs,chr(9)`): Այդ սիմվոլները օգտագործվում են տեքստի ֆորմատավորման համար: Հաջորդական դիմումի ֆայլերի բացման եղանակը տրվում է «օրেն» օպերատորի օգնությամբ, սակայն մինչ այդ պետք է իմանալ ազատ ուղու համարը:

«FreeFile» ֆունկցիան վերադարձնում է ազատ ուղու համարը, որը կարելի է օգտագործել ֆայլերի հետ աշխատանքի համար: Այն ունի հետևյալ ֆորմատը՝ FreeFile[(range Number)]: Եթե ազատ ուղիներ չկան (բացված են առավելագույն թույլատրելի քանակով ֆայլեր), առաջանում է կատարման սխալ: Ոչ պարտադիր Range Number պարամետրը թույլ է տալիս որոշել արժեքների միջակայքը, որից ընտրվում է հերթական ուղու ազատ համարը: Եթե դրա արժեքը հավասար է 0-ի (լույսայն), ապա վերադարձվում է ուղու համար 1-255 միջակայքից, եթե 1, ապա՝ 256-511 միջակայքից: FreeFile_ը կարելի է և օգտագործել առանց արգումենտի, օրինակ՝ IntFH=FreeFile():

Այժմ, եթե արդեն ունենք ազատ ուղի, կարելի է բացել ֆայլը «open» օպերատորի օգնությամբ: Այն ունի հետևյալ ֆորմատը.

Open ֆայլի_անուն For [Input / Output / Append] As filehandle

Եթե նշված անունով ֆայլը գոյություն ունի, ապա «Output» ռեժիմում նրա պարունակությունը հեռացվում է, իսկ «Append» ռեժիմում ֆայլը բացվում է ավելացնելու համար: Գոյություն ունեն հաջորդաբար դիմելու համար բացված ֆայլից տվյալներ կարդալու մի քանի հնարավորություններ: Ընդհանուր դեպքում դա իրականացվում է «Input» օպերատորի օգնությամբ, որն ունի մի քանի տարատեսակներ.

- Line Input# - կարդում է մեկ տող
- Input# - կարդում է հաջորդաբար «Write#» օպերատորի կողմից գրված սիմվոլները
- Input\$- կարդում է որոշակի քանակությամբ սիմվոլներ:

Գոյություն ունեն ֆայլից ամբողջ ինֆորմացիան կարդալու մի քանի տարրերակներ: Կարդալուց առաջ պետք է բացել ֆայլը «Open ... For » օպերատորի օգնությամբ:

IntFH=FreeFile

Open " C:\Text.txt" For Input As intFH

Օրինակներ.

Առաջին տարրերակ՝

DO Until EOF(intFH)

Line Input #intFH, StrString

StrText=StrText&StrString&vbLf

Loop

Երկրորդ տարբերակ՝

StrText= Input\$(LOF(intFH), intFH)

Close# intFH

Եթե տարբերակն էլ բերում են նույն արդյունքի:

Ֆայլում գրառումը: Visual Basic-ում ֆայլում ինֆորմացիայի գրառման համար օգտագործվում են Print# և Write# օպերատորները:

Print# filehandle, [{Spc(n) | Tab [(n)]}][expression][charpos]

Օրինակ՝ Print# intFH, “Դատված1” ; Tab ; “Դատված2”

Եթե օպերատորում տվյալները բաժանենք ստորակետով, ապա ֆայլում դրանք կրածանվեն տարբույացիայի սինվոլով, իսկ կետստորակետով տվյալների ֆայլում գրանցվում են առանց բաժանիչների:

«Write#» օպերատորը ունի այնպիսի սինտաքսիս, ինչպիսին է Print# -ը: Տարբերությունը միայն ելքի ֆորմատավորման մեջ է: Օրինակ Print# intFH, “ԱՅՆա”, “Երևան”, 17 ֆայլում կլինի ԱՅՆա Երևան 17 իսկ Write# intFH, “ԱՅՆա”, “Երևան” 17 ֆայլում կլինի “ԱՅՆա”, “Երևան”, 17: Տվյալները, որոնք պահպանվել են Write# օպերատորի օգնությամբ, կարելի է կարդալ Input# օպերատորի օգնությամբ:

Պատահական Random ձևը, տբ - ին դիմելու միջոցների ի հայտ գալով, մի քիչ կորցրել է իր նշանակությունը: Ի տարբերություն հաջորդական դիմումի, որի ժամանակ տվյալները ֆայլում պահպանվում են ոչ կառուցվածքավորված տեսքով, պատահական դիմումը ենթադրում է, որ ֆայլը ունի հաստատուն կառուցվածք, որը թույլ է տալիս կարդալ տվյալները կամայական կարգով:

Open Ֆայլի_անուն For Random [Access դիմում] [Բլոկավորում] As [#]handle[Len=Գրվածքի_Երկարություն]:

Եթե դիմելու իրավունքը չի նշված, ապա լրելյան օգտագործվում է Read Write: Օրինակ.

Open,Date,Date For Random AccessReadWrite As intFH

Թանի որ դիմելու այդ տիպը սովորաբար նախատեսված է ֆայլերի հետ աշխատանքի համար, որոնք կարող են օգտագործվել շատ շահագործողների կամ հավելվածների կողմից, ապա պետք է ապահովել կոլեկտիվ օգտագործման ժամանակ տվյալների ամբողջականությունը: Դրա համար պետք է տեղադրել Lock պարամետր, որը որոշում է բացված ֆայլին դիմելու իրավունքը: Այդ պարամետրը կարող է ընդունել հետևյալ արժեքները:

- Shared–ֆայլը կարող է օգտագործվել բոլոր պրոցեսների կողմից գրելու և կարդալու համար
- Lock Read–ոչ մի ուրիշ պրոցես չի կարող կարդալ տվյալները ֆայլից
- Lock Write–ոչ մի ուրիշ պրոցես չի կարող գրել տվյալները ֆայլում
- Lock Read Write–ոչ մի ուրիշ պրոցես չի կարող գրել և կարդալ: Օրինակ՝ Open “Address.Dat” ForRandom Access Write As 1 Len=27

Open “Address.Dat” ForRandom Access Write As 1 Len=Len
(Varname):

Սուտք և ելք: Տվյալների գրառման և կարդալու համար օգտագործվում են համապատասխանաբար Put և Get օպերատորները:

Put # file handler, գրառման_համար, փոփոխական օրինակ Put
#intFH,7, Address պահպանում է 7-րդ գրվածքը:

Get # file handler, գրառման_համար, փոփոխական օրինակ Get # intFH, 2, Address կարդում է 2-րդ գրվածքը:

Որպեսզի մեկ գրվածքում պահպանվեն տարրեր տիպի մի քանի արժեքներ, պետք է օգտագործել տվյալների շահագործման տիպ:

```
(General) (Declaration)
Type Person
First Name As String * 20
Name As String * 20
CustomerN As Integer
End Type
Private Customer As Person
'Իրողեղութա
Private Sub Command1_Click()
intFH = FreeFile
Open "C:\ LORE.DAT" For Random As intFH Len=Len
(Customer)
Get # intFH, 2, Customer
Close # intFH
End Sub
```

Երկուական դիմում: Երկուական դիմումը որոշ չափով տարբերվում է կամայական դիմումից: Տարբերությունը միայն այն է, որ երկուական դիմումը հնարավոր է ոչ թե տվյալների որոշակի հավաքածուին, այլ ցանկացած ֆայլի ներսը՝ առանձին բայթին:

Open ֆայլի _անուն For Binary [Access դիմում] [Բլոկիրովկա] As [#] Handle

Երկուական դիմելու ժամանակ «Open» օպերատորի ֆորմատը ննան է կամայական դիմելու ֆորմատին: Գլխավոր տարբերությունն այն է, որ Random բանալիային բառի փոխարեն նշվում է Binary, իսկ Len պարամետրը բացակայում է, քանի որ գրվածքն ունի ֆիքսված երկարություն՝ 1 բայթ: Open "SPOCK.VUL" For Binary As intFH:

Սուտր և ելքը: Get # intFH, 5, Var 'կարդում է 5-րդ բայթը, եթե տվյալները կարդացվում են:

Put # intFH, 78, Var 'զրում է 78-րդ բայթը տողում, կարդացվող բայթերի քանակը հավասար է տողի երկարությանը:

StrOutput\$=String (14, "") այս օրինակում տողում կարդացվում է 14 բայթ սկսած Get # intFH, 26, StrOutput\$ ֆայլի 26-րդ բայթից:

Դավելվածի աշխատանքի պատշաճ ավարտման համար պետք է բոլոր բացված ֆայլերը փակել: Close # intFH կամ Close intFH # - ի օգտագործումը պարտադիր չէ, իսկ եթե Close, ապա փակում է բոլոր ֆայլերը, որոնք բացվել են հավելվածի կողմից:

Windows-ում օգտագործվում են հետևյալ տիպի տառատեսակներ ռաստրային, վեկտորային, TrueType (կոնտուրային) և փրինթերային:

2.17. Տպող սարքի (Printer) հետ աշխատանքը VB-ի միջավայրից

Տվյալների տպագրման համար VB-ում կան մի շարք հնարավորություններ: Պարզագույններից կարելի է նշել PrintForm մեթոդը, որի օգնությամբ անհրաժեշտ ծևապատկերը (ֆորման) կարելի է ուղարկել տպող սարքին: Բացի դրանից, կարելի է օգտվել նաև Printer օբյեկտից: PrintForm մեթոդը թույլ է տալիս, որ տպագրման ենթակա ֆորման թղթի վրա տպվի այն տեսքով, ինչ տեսքով գոյություն ունի էկրանի վրա: Ազգբում «Load» օպերատորի միջոցով ֆորման բեռնավորվում է առանց այն դուրս բերելու էկրանի վրա, այնուհետև կատարվում է ինիցիալիզացիա և

ուղարկվում տպելու PrintForm մեթոդի միջոցով: Վերջում հիշողությունից հեռացվում է օպերատորի օգնությամբ:

```
Load frmPrint
```

```
frmPrint.lblOutput.caption= "Text"
```

```
frmPrint.lblOutput.Picture=LoadPicture("C:\tatrap.bmp")
```

```
frmPrint.PrintForm
```

```
Unload frmPrint
```

«Printer» օբյեկտը հնարավորություն է տալիս տպելու ոչ թե էկրանի, այլ տպող սարքի համար նախատեսված ֆորմաներ: Այդ իսկ պատճառով այն ապահովում է ավելի բարձր որակ, սակայն պահանջում է մեծ ծավալ: «Printer» օբյեկտի համար առաջնային է Print մեթոդը: Եթի կետերի փոփոխման համար օգտագործվում է CurrentX / CurrentY հատկությունը: Տպելու տիրույթի լայնությունն ու բարձրությունը կարգավորվում է ScaleWidth և ScaleHeight հատկություններով: Տպագրվող տողի փաստացի լայնությունը և բարձրությունը որոշելու համար պետք է օգտվել TextHeight և TextWidth մեթոդից: Օրինակ՝

```
strOutputText="Hello Armenia"
```

```
strOutputText=Printer.TextWidth(strOutputText)
```

```
nPrinterWidth=Printer.ScaleWidth
```

```
Printer.CurrentX=(nPrinterWidth-nTextWidth)/2
```

```
PrinterPrint strOutputText
```

```
Printer.EndDoc
```

Տեքստը դասավորում է կենտրոնի նկատմամբ հավասար հեռավորությամբ:

Եթե «Print» օպերատորում տպվող արտահայտությունները միմյանցից բաժանվում են կետ-ստորակետով, ապա դրանք տպվում են անմիջապես մեկը մյուսից հետո, իսկ եթե բաժանվում են ստորակետով, ապա տպվում են մեկը մյուսից 14 նիշ հեռավորությամբ: Տպվող օբյեկտների միջև եղած հեռավորությունը կարելի է կարգավորել Tab() ֆունկիայի միջոցով:

Գրաֆիկական օբյեկտների տպման համար կարելի է օգտագործել «Print» օբյեկտի Pset, Line և Circle մեթոդները: Ֆորմաների պատրաստի գրաֆիկական պատկերները կարելի է տպագրել PaintPicture մեթոդի միջոցով: Դրա համար պետք է նշել «Picture» հատկությամբ օժտված դեկավարնան տարրը, իսկ անհրաժեշտության դեպքում նաև դիրքն ու պատկերի չափերը:

```

Printer.ScaleMode=vbCentimeters
Print.Picture picPictPrinter.Picture 5,5
    Դեռևս օպերատիվ հիշողությունում գտնվող տպելու ենթակա
պատրաստի տողի մասին տեղեկությունը տրվում է NewPage
մեթոդով, իսկ փաստաթղթի ուղարկումը՝ EndDoc մեթոդով:
    Printer.Print "Page1"
    Printer.NewPage
    Printer.Print "Page2"
    Printer.EndDoc
    Տպելու գործընթացն ընդհատելու համար օգտագործվում է
KillDoc մեթոդը:

```

2. 18. Օբյեկտակողմնորոշված ծրագրավորումը VBA –ում

Դասի (class) ֆունկցիոնալ հնարավորություններից օգտվելու համար անհրաժեշտ է հայտարարել օբյեկտային փոփոխական: Դասը ընդամենը նախատիպ է, որի հետ չի կարելի աշխատել անմիջապես: Պետք է ստեղծել հատուկ փոփոխական, որը պարունակում է դիմում օբյեկտին: Օբյեկտային փոփոխականները բաժանվում են 2 տիպի ունիվերսալ, որոնք կարող են պահել ցանկացած տիպի օբյեկտ, և հատուկ տիպի: Առաջինը հայտարարվում է «Object» տիպի: Օրինակ Dim MyObject AS Object: Օբյեկտների էլեմենտներին դիմելու համար կիրառվում է «control» տիպ Dim MyControl As control: «MyControl» օբյեկտային փոփոխականը կարող է պարունակել դիմում ցանկացած, բայց ոչ ուրիշ տիպի էլեմենտնի: Եթե մեզ հայտնի է օբյեկտի տիպը, որը պահվում է փոփոխականում, դա նշում ենք հայտարարման համար:

```

Օրինակ Dim Tbox AS TextBox
Dim OkCancel As commandbutton
կամ Dim Tbox AS control
Set Tbox=Form3! TxtName
Tbox.Font.Bold=True
Բացի դրանից, կարելի է ստեղծել օբյեկտային փոփոխական,
որը դիմում է անմիջապես«Font» օբյեկտին:
Dim TboxFont AS object
Set TboxFont=Form3!TxtName.Font
TboxFont.Bold=True

```

Ֆորմաները (ձևապատկերները) իրենցից ներկայացնում են դասեր, որոնք կարելի է ստեղծել «Dim» հրամանով: VB-ում ֆորմաների հետ գործողության համար կա 4 մեթոդ՝ Load, Show, UnLoad և Hide: Օրինակ՝ Dim myForm As Form

Set myForm=Form1

myForm.Show

Մի քանի օրինակների ստեղծման համար օգտագործվում է տարրերի զանգված: Զանգվածի բոլոր տարրերը ունեն ընդհանուր անուն և տարրերվում են ինդեքսով: VBA-ն առաջին հերթին ստեղծվել է նրա համար, որպեսզի կարողանանք դիմել հավելվածի ֆունկցիոնալ հնարավորություններին կամ ավտոմատացնել այն (VBA-ի հրամաններից կազմված պրոցեդուրայի՝ մակրոսի օգնությամբ) կամ դեկավարել դրանց այլ հավելվածից: Ինչպես հայտնի է, շատ հավելվածներում այդ նպատակի համար օգտագործվում են օբյեկտներ: Նշենք դրանցից մի քանիսը՝

- IDE Visual Basic-ը պաշտպանում է CommandBars հավաքածուն (կողեկցիան-ընտանիքը), որի օբյեկտները ցանկի (մենյուի) հրամաններ կամ էլ գործիքակազմի սեղմակներ են:
- DAO գրադարանը պաշտպանում է «Database» օբյեկտը, որը ներկայացնում է տվյալների բազան ամբողջությամբ:
- Word-ը պաշտպանում է Active Document (ընթացիկ փաստաթուղթ) օբյեկտները և Selection (ընթացիկ առանձնացված) հատվածը:
- Excel-ը պաշտպանում է հատուկ օբյեկտների հավաքածու՝ լավագույնս համապատասխանեցնելով նրա բնույթին: Օրինակ, Activesheet – ը՝ ActiveDocument –ի վոլյուսը և այլն:

Բոլոր այդ օբյեկտները ունեն հասկություններ, որոնք նույնականացնում են օբյեկտներ կամ օբյեկտների հավաքածուներ: Օր. Documents("Fin4k.doc").Paragraphs(1).Font.Bold:

Եթե Word-ում ինչորմացիայի միավորը հանդիսանում է տեքստի հատվածը (և ոչ թե միմպոլը կամ բառը), ապա Excel-ում նման բազային միավորը բժիշների միջակայքն է ներկայացված «Range» օբյեկտով: Կարելի է հայտարարել օբյեկտային փոփոխականները հետևյալ տեսքով Dim AppWord As Word.Application կամ Dim AppExcel As Excel.Application: Կամ եթե AppExcel-ը հայտարարենք որպես օբյեկտային փոփոխական, ապա կոնկրետացնելու համար պետք է օգտվել CreateObject և GetObject ֆունկցիաներից: Օրինակ, Dim AppExcel As Object, այնուհետև Set

AppExcel=CreateObject("Excel.Application") կամ Set փոխական
= GetObject(C:\MyDocument\Fin4k.xls):

Օրինակ VB-ի միջավայրից դիմել Word-ի փաստաբղյն,
Excel-ի աշխատանքային թերթին, կատարել հաշվարկ Excel-ի
միջավայրում:

```
Option Explicit
Dim AppWord As Word.Application
Dim AppExcel As Excel.Application
Private Sub Command1_Click()
    Screen.MousePointer = vbHourglass
    Set AppExcel = CreateObject("Excel.Application")
    Screen.MousePointer = vbDefault
    Command3.Enabled = True
    Command5.Enabled = True
    Command8.Enabled = True
End Sub
Private Sub Command2_Click()
    Screen.MousePointer = vbHourglass
    Set AppWord = CreateObject("Word.Application")
    Screen.MousePointer = vbDefault
    Command4.Enabled = True
    Command6.Enabled = True
End Sub
Private Sub Command3_Click()
    Dim wSheet As Worksheet
    Dim wBook As Workbook
    If AppExcel.Workbooks.Count = 0 Then
        Debug.Print "Adding a new Workbook"
        Set wBook = AppExcel.Workbooks.Add
    End If
    Set wSheet = AppExcel.Sheets(1)
    wSheet.Cells(2, 1).Value = "1st QUARTER"
    wSheet.Cells(2, 2).Value = "2nd QUARTER"
    wSheet.Cells(2, 3).Value = "3rd QUARTER"
    wSheet.Cells(2, 4).Value = "4th QUARTER"
    wSheet.Cells(3, 1).Value = 123.45
    wSheet.Cells(3, 2).Value = 435.56
    wSheet.Cells(3, 3).Value = 376.25
```

```

wSheet.Cells(3, 4).Value = 425.75
Range("A2: D2").Select
With Selection.Font
    .Name = "Verdana"
    .FontStyle = "Bold"
    .Size = 12
End With
Range("A3: D3").Select
With Selection.Font
    .Name = "Verdana"
    .FontStyle = "Regular"
    .Size = 11
End With
Range("A2: D2").Select
Selection.Columns.AutoFit
Selection.ColumnWidth = Selection.ColumnWidth * 1.25
Range("A2: E2").Select
With Selection
    .HorizontalAlignment = xlCenter
End With
AppExcel.Visible = True
End Sub
Private Sub Command4_Click()
Dim wDoc As Document
Dim tmpText As String
Dim parCount As Long, wordCount As Long, charCount As Long
Dim msg As String
If AppWord.Documents.Count = 0 Then
    AppWord.Documents.Add
End If
AppWord.Documents(1).Range.InsertAfter "Document's Title" &
vbCr AppWord.Documents(1).Range.Font.Bold = True
AppWord.Documents(1).Range.Font.Size = 16
AppWord.Documents(1).Range.Font.Name = "Comic Sans MS"
AppWord.Documents(1).Range.InsertAfter "This the document's
first paragraph, aligned to the left." & vbCr
AppWord.Documents(1).Range.InsertAfter "This the document's
second paragraph. "

```

```
AppWord.Documents(1).Range.InsertAfter "It was inserted with  
the statement "  
AppWord.Documents(1).Range.InsertAfter"AppWord.Documents  
(1).Range.InsertAfter"AppWord.Documents(1).Range.InsertAfter  
"and is also left aligned." & vbCrLf  
parCount = AppWord.Documents(1).Paragraphs.Count  
wordCount = AppWord.Documents(1).Words.Count  
charCount = AppWord.Documents(1).Characters.Count  
msg = "The new document contains " & vbCrLf  
msg = msg & parCount & " paragraphs" & vbCrLf  
msg = msg & wordCount & " words" & vbCrLf  
msg = msg & charCount & " characters"  
MsgBox msg  
AppWord.Visible = True  
AppWord.Documents(1).Paragraphs(1).Alignment=wdAlignPara  
graphCenter  
End Sub  
Private Sub Command5_Click()  
AppExcel.DisplayAlerts = False  
AppExcel.Quit  
Command3.Enabled = False  
Command5.Enabled = False  
Command8.Enabled = False  
End Sub  
Private Sub Command6_Click()  
On Error Resume Next  
AppWord.DisplayAlerts = False  
AppWord.Quit  
Dim wRunning As String  
wRunning = AppWord.Application.Name  
If Error Then  
Command4.Enabled = False  
Command6.Enabled = False  
End If  
End Sub  
Private Sub Command7_Click()  
End  
End Sub
```

```
Private Sub Command8_Click()
Dim wSheet As Worksheet
Dim wBook As Workbook
Dim expression
expression = InputBox("Enter math expression to evaluate (i.e.,
1/cos(3.45)*log(19.004)")
If Trim(expression) <> "" Then
If AppExcel.Workbooks.Count = 0 Then
Debug.Print "Adding a new Workbook"
Set wBook = AppExcel.Workbooks.Add+
End If
Set wSheet = AppExcel.Sheets(1)
On Error GoTo CalcError
wSheet.Cells(1, 1).Value = "=" & expression
wSheet.Calculate
MsgBox "The value of the expression " & expression & vbCrLf &
" is " & wSheet.Cells(1, 1).Value
End If
Exit Sub
CalcError:
MsgBox "Error in evaluating expression"
End Sub
Private Sub Form_Terminate()
Set AppExcel = Nothing
Set AppWord = Nothing
End Sub
```

2. 19. Զարգացման ինտեգրացված միջավայր

IDE (Intagrated Development Environment) – Սա թույլ է տալիս ստեղծել, տեստավորել և կարգավորել ցանկացած տիպի նախագիծ ստանդարտ EXE ֆայլից մինչև ActiveX էլեմենտներ, ինչպես նաև ստեղծել ActiveX փաստաթղթեր: IDE-ն ընդայնվում և կատարելագործվում է Visual Basic-ի յուրաքանչյուր նոր տարրերակի հետ: Մենք նույնպես կարող ենք կատարելագործել, ընդգրկել նրա մեջ նոր հրամաններ և նույնիսկ նոր մենյու: Այդ խնդիրը լուծվում է լրակառույցի օգնությամբ (add-ins): Լրակառույցները իրենցից ներկայացնում են բաղկացուցիչներ, որոնք տեղադրված են համակարգչում նախագծողի (մշակողի) կողմից: Visual Basic-ում ստեղծվող տարրեր տիպի նախագծերի մեջ շարլոն add-in-ը հավելված է, որը կցում է ինտեգրացված միջավայրին նրա հնարավորությունների ընդլայնման համար: Օրինակ, եթե հավելվածներում հաճախ ենք օգտագործում API ֆունկցիաները բեռնավորում ենք API Viewer լրակառույցը նեկնարկնան ժամանակ: Լրակառույցը հանդիսանում է նախագծի (պրոյեկտի) մաս: Նախագծի յուրաքանչյուր բացման ժամանակ ավտոմատ բեռնավորվում է նրան միացված բոլոր լրակառույցները: Սակայն լրակառույցները չեն նտնում նախագծի կատարվող ֆայլի մեջ. դրանք ընդամենը գործիքներ են, որոնք հեշտացնում են մշակման պրոցեսը և պետք չեն կատարման ժամանակ: Որպեսզի ծրագրավորողը կարողանա դեկավարել նախագծման միջավայրը, Visual Basic-ը ներկայացնում է նրա տրամադրության տակ VBADE դաս, որը ապահովում է Visual Basic-ի աշխատանքային միջավայրը: VBADE դասը պաշտպանում է բոլոր օբյեկտները, որոնք անհրաժեշտ են ինտեգրացված միջավայրի բաղկացուցիչների հետ աշխատանքի համար (տարրեր պատուհաններ, մենյու, նախագծի բաղադրիչներ, ծրագրային կոդերով ֆորմայի էլեմենտներ և այլն): VBADE-ի դասի հատկությունները և մեթոդները թույլ են տալիս դիմել ցանկացած օբյեկտի, որի հետ աշխատում ենք նախագծման ռեժիմում և դեկավարել նրա աշխատանքը: VBADE-ի դասի օբյեկտների մասին ինֆորմացիա ստանալու համար պետք է կատարել հետևյալ գործողությունները.

1. սկսում ենք նոր նախագիծ և կատարում References պատուհանում հղում Microsoft Visual Basic 6.0 Extensibility
2. մեկնարկում ենք Object Browser-ը

3. Object Browser-ում ընտրում ենք VBIDE Type Library տողը և պատուիանում ներկայացվում են դասերի թվարկումը և ֆրանց անդամները:

Լրակառույցը պետք է ներկայացնի շահագործողի տրամադրության տակ երկու տիպի օբյեկտներ.

- օբյեկտներ, որոնք նախատեսված են Visual Basic-ի միջավայրի դեկավարման համար,
- օբյեկտներ, որոնք ներկայացնում են տարբեր պատուիաններ (օրինակ՝ ծրագրի պատուիան, հատկությունների պատուիան կամ պալիտրա), նախագծի բաղկացուցիչներ և այլն:

Բոլոր այդ օբյեկտները հանդիսանում են արմատային օբյեկտի հատկություններ, որոնք ներկայացնում են ինտեգրացված միջավայրը ամբողջությամբ (տվյալների բազայի օբյեկտներին դիմելու հնարավորություն օբյեկտային փոփոխականի միջոցով, որը ներկայացնում է տվյալների բաց բազան): Ցանկացած լրակառույցի արմատային բաղադրիչը հանդիսանում է VBInstance օբյեկտային փոփոխականը, որը ներկայացնում է Visual Basic-ի ընթացիկ օրինակը: VBIDE օբյեկտի բոլոր բաղադրիչներին դիմելու համար օգտագործվում է VBInstance փոփոխական, որը հայտարարվում է VBIDE.VBE տիպի:

VBIDE դասը ծրագրավորողին առաջարկում է Windows հավաքածու, որը ներկայացնում է տարբեր պատուիաններ և Window օբյեկտ: IDE-ի բոլոր պատուիանները հանդիսանում են Windows ընտանիքի անդամներ, որոնք բաղկացած են Windows օբյեկտներից: Window օբյեկտը պաշտպանում է ոչ քիչ թվով հատկություններ և մերողներ: Caption, Width, Height, Left և Top օբյեկտները բացատրության կարիք չեն գգում. դրանք թույլ են տալիս պատուիանը տեղադրել անհրաժեշտ տեղում: VBIDE օբյեկտի բոլոր հաստատունները սկսվում են `vbeext` նախամասնիկով: Windows դասը պարունակում է երկու ստանդարտ հատկություն՝ `Count` և `Item`: IDE-ի ընթացիկ ակտիվ պատուիանը որոշվում է VBE օբյեկտի `MainWindow` հատկությամբ: Եթե IDE-ի ընթացիկ օրինակը պարունակում է նախագծերի խումբ, ապա `VBProject` հավաքածուն թույլ է տալիս դիմել այդ խմբի ցանկացած նախագծի: `VBProject` օբյեկտը ներկայացնում է առանձին նախագիծ, սակայն ցանկացած նախագիծ բաղկացած է բաղմաթիվ բաղկացուցիչներից (ֆորմաներ, դասեր, հատկությունների էջեր և այլն):

VBProject օբյեկտի բաղկացուցիչներին դիմելու համար օգտագործվում է VBComponents հավաքածու, որը միավորում է Vbcomponent օբյեկտներ: Ընթացիկ նախագիծը որոշվում է VBInstance.ActiveVBProject-ով: Նման ձևով կարելի է դիմել ակտիվ նախագիծի ընթացիկ բաղադրիչին: Այդ նպատակի համար օգտագործվում են VBInstance նախագիծի SelectedVBcomponent հատկությունը: Ֆորմայի էլեմենտի հետ աշխատանքի համար օգտագործվում են Vbcontrols և ContainedVBcontrols հավաքածուներ (Երկու հավաքածուներն ել բաղկացած են vbccontrol օբյեկտներից): Ծրագրային կոդին դիմելու ժամանակ կիրառվում է CodeModule օբյեկտ: Vbcontrol էլեմենտը պարունակում է որոշ ստանդարտ անդամներ (Add, Remove, Count, Item) և պաշտպանում է ItemAdded, ItemRemoved, ItemRenamed իրադարձություններ:

Լրակառույցի ստեղծման գործընթացը այնքան էլ չի տարբերվում VB-ի սովորական հավելվածի ստեղծումից: Add-in շարլոնը գեներացնում է լրակառույցի կմախք, որի մեջ մենք ստեղծում ենք կողը: Լրակառույցի հետ աշխատանքի ժամանակ պետք է ներկայացնել կող Add-ins մենյուում լրակառույցին կցելու համար, ինչպես նաև կող մենյուում լրակառույցի անվան ակտիվացնան մշակման համար: Ստեղծում ենք նոր նախագիծ և ընտրում New Project պատուհանում Add-In շաբլոն: Visual Basic-ը նախագիծի պատուհանում ստեղծում է երկու ծրարներ (թղթապանակ-Folder).

- Forms ծրարը պարունակում է ֆորմաներ, որոնք արտացոլվում են լրակառույցում (եթե դրանք կան): Լույսայն Forms ծրարը պարունակում է frmAddin ֆորմա:
- Designers ծրարում գտնվում է ActivexDesigner կոնստրուկտոր Connect անունով:

2. 20. Մենյուների ստեղծումը Visual Basic-ում

Ինչպես ցանկացած կիրառական ծրագրային փաթեթներում գոյություն ունի գլխավոր մենյու /օրինակ File, Edit, Insert .../, այնպես էլ Visual Basic-ով ստեղծված հավելվածներում կարելի է ստեղծել շահագործողի համար ցանկալի մենյուների շարք: Form1 պատուհանի ընտրությունից հետո մենյուի ստեղծումը կատարվում է հատուկ խմբագրիչի օգնությամբ /Menu Editor/: Այս խմբագրիչի գործարկումը կատարվում է հետևյալ ճանապարհներից որևէ մեկով:

1. [Ctrl+E] ստեղնաշարային համադրությամբ
2. Tools\menu Editor...Ճանապարհով
- 3.Գործիքների ստանդարտ գոտու վրա համապատասխան սեղմակով:

Մենյուն սովորաբար բաղկացած է լինում մի քանի մակարդակներից: Վերին մակարդակը մենյուի տողն է, որտեղ տեղադրվում են գլխավոր մենյուի էլեմենտները: Դնարավոր է կառուցել մինչև վեց մակարդակի մենյու, սակայն խորհուրդ է տրվում բավարարվել ավելի քչով: Եթե անհրաժեշտ է լինում, որ մենյուի հրամանի կատարումից հետո կատարվի կող, ապա մենյուի տվյալ էլեմենտը պետք է վերջանա (!) նշանով: Եթե մենյուի էլեմենտի կատարումը կանչում է երկխոսության պատուհան, ապա անվանման գրանցմանը պետք է հետևի երեք կետ (...):

Մենյուն կառուցվում է հետևյալ սկզբունքով՝

Էլեմենտ 1. Տողի վերնագիրը (մակարդակ_1)

-----Մակարդակ_2

-----Մակարդակ_3

-----Մակարդակ_4

-----Մակարդակ_5

-----Մակարդակ_6

Էլեմենտ 2. Տողի վերնագիրը (մակարդակ_1)

-----Մակարդակ_2

-----Մակարդակ_3

-----Մակարդակ_4

-----Մակարդակ_5

-----Մակարդակ_6

Մեկնաբանենք խմբագրիչի գործարկումից հետո բացված պատուհանը:

- Caption դաշտում մուտքագրվում է մենյուի ցանկալի անվանումը
- Name դաշտում նախ գրանցվում է հատուկ նախածանցը՝ որու, որից հետո նոր միայն մենյուի անվանումը
- Enabled և Visible նշիչներով որոշվում է մենյուի էլեմենտի տեսանելիությունը և հասանելիությունը
- Shortcut դաշտում կարող ենք տվյալ մենյուի համար ընտրել այն գործարկելու ստեղնաշարային համարությունը “տաք ստեղներ” (Ctrl+A,CTRL+B)
- WindowList. օգտագործվում է MDI ձևի մենյուներ ստեղծելու համար
- HelpContextID. օգտագործվում է Help կոմախյատորի հետ աշխատելու ժամանակ
- Next. կատարվում է անցում հաջորդ մենյուին կամ ենթամենյուին
- Insert. տեղադրում է նոր մենյու գոյություն ունեցող մենյուից առաջ /ավելացնում է նոր տող/
- Delete. հեռացնում է ընթացիկ մենյուն
- Գլխավոր մենյուի տակ ենթամենյուների սփյուման ուղղությունները կարգավորվում է համապատասխան սլաքների ուղղությամբ
- Negotiate Position դաշտի օգնությամբ կատարվում է մենյուի դիրքի տեղաշարժ՝ ընտրելով առաջարկվող տարբերակներից մեկը:

Մենյուի էլեմենտների անվանումների նշանակման ժամանակ հարկավոր է պահպանել որոշակի կանոններ: Օրինակ

File mnuFile

File\Open... mnuFOpen

File\Send\Fax mnuFSFax

Օրինակ, ֆորմայի վրա տեղադրել 1 հրամանային սեղմակ (commandButton), մեկ Picturebox դեկավարման էլեմենտ, մեկ Timer, մեկ commandDialog էլեմենտ, մեկ MMcontrol էլեմենտ և Image էլեմենտի զանգված: Ստեղծել մեջու, որի ենթամենյուներով ընտրվի երաժշտական (առողիո) ֆայլ և անհմացիոն ծրագիր: Start Demo սեղմակի օգնությամբ ընտրված երաժշտության տակ մեկնարկի համապատասխան անհմացիոն ծրագիրը, իսկ սեղմակի գրությունը փոխվի Stop Demo -ի, որի օգնությամբ դադարեցվի աշխատանքը:

Option Explicit

Dim FrameNum

Dim XPos

Dim YPos

Dim DoFlag

Dim Motion

Dim R

Dim G

Dim B

Private Sub CircleDemo()

Dim Radius

R = 255 * Rnd

G = 255 * Rnd

B = 255 * Rnd

XPos = ScaleWidth / 2

YPos = ScaleHeight / 2

Radius = ((YPos * 0.9) + 1) * Rnd

Circle (XPos, YPos), Radius, RGB(R, G, B)

End Sub

Private Sub cmdStartStop_Click()

Dim UnClone

Dim MakeClone

Dim X1

Dim Y1

Select Case DoFlag

Case True

cmdStartStop.Caption = "Start Demo"

DoFlag = False

```
mnuOption.Enabled = True
If mnuCtlMoveDemo.Checked = True Then
    picBall.Visible = False
ElseIf mnulImageDemo.Checked = True Then
    imgMoon(0).Visible = False
ElseIf mnuScaleDemo.Checked = True Then
    Cls
    Scale
ElseIf mnuCircleDemo.Checked = True Then
    Cls
End If
Case False
cmdStartStop.Caption = "Stop Demo"
DoFlag = True
mnuOption.Enabled = False
If mnuCtlMoveDemo.Checked = True Then
    picBall.Visible = True
    Motion = Int(4 * Rnd + 1)
ElseIf mnulImageDemo.Checked = True Then
    imgMoon(0).Visible = True
    FrameNum = 0
    Motion = Int(4 * Rnd + 1)
ElseIf mnuScaleDemo.Checked = True Then
    Randomize
    DrawWidth = 1
    ScaleLeft = 1
    ScaleTop = 10
    ScaleWidth = Int(13 * Rnd + 3)
    ScaleHeight = -10
ElseIf mnuCircleDemo.Checked = True Then
    DrawWidth = 1
    DrawStyle = vbDash
    DrawMode = vbXorPen
End If
End Select
End Sub
```

```

Private Sub CtlMoveDemo()
    Select Case Motion
        Case 1
            picBall.Move picBall.Left - 20, picBall.Top - 20
            If picBall.Left <= 0 Then
                Motion = 2
            ElseIf picBall.Top <= 0 Then
                Motion = 4
            End If
        Case 2
            picBall.Move picBall.Left + 20, picBall.Top - 20
            If picBall.Left >= (DemoForm.Width - picBall.Width) Then
                Motion = 1
            ElseIf picBall.Top <= 0 Then
                Motion = 3
            End If
        Case 3
            picBall.Move picBall.Left + 20, picBall.Top + 20
            If picBall.Left >= (DemoForm.Width - picBall.Width) Then
                Motion = 4
            ElseIf picBall.Top >= (DemoForm.Height - picBall.Height) - 680
            Then
                Motion = 2
            End If
        Case 4
            picBall.Move picBall.Left - 20, picBall.Top + 20
            If picBall.Left <= 0 Then
                Motion = 3
            ElseIf picBall.Top >= (DemoForm.Height - picBall.Height) - 680
            Then
                Motion = 1
            End If
    End Select
End Sub
Private Sub Delay()
    Dim Start
    Dim Check
    Start = Timer

```

```
Do Until Check >= Start + 0.15
    Check = Timer
Loop
End Sub
Private Sub MnuWawmusic_Click()
With CommonDialog1
    .InitDir = "C:\Documents and Settings\User\Desktop"
    .Filter = "wav(*.wav)|*.wav"
    .ShowOpen
End With
With MMControl1
    .Wait = True
    .DeviceType = "WaveAudio"
    .FileName = CommonDialog1.FileName
    .Command = "open"
End With
End Sub
Private Sub Form_Resize()
    If mnuScaleDemo.Checked = True And DemoForm.WindowState
    = 0 Then
        Randomize
        DrawWidth = 1
        ScaleLeft = 1
        ScaleTop = 10
        ScaleWidth = Int(13 * Rnd + 3)
        ScaleHeight = -10
    End If
End Sub
Private Sub Form_Unload(Cancel As Integer)
    End
End Sub
Private Sub ImageDemo()
    Select Case Motion
        Case 1
            imgMoon(0).Move imgMoon(0).Left - 100, imgMoon(0).Top -
100
            IncrFrame
    End Case
End Sub
```

```

If imgMoon(0).Left <= 0 Then
    Motion = 2
Elseif imgMoon(0).Top <= 0 Then
    Motion = 4
End If
Case 2
    imgMoon(0).Move imgMoon(0).Left + 100, imgMoon(0).Top -
100
    IncrFrame
    If imgMoon(0).Left >= (DemoForm.Width - imgMoon(0).Width)
Then
    Motion = 1
    Elseif imgMoon(0).Top <= 0 Then
        Motion = 3
    End If
Case 3
    imgMoon(0).Move imgMoon(0).Left + 100, imgMoon(0).Top +
100
    IncrFrame
    If imgMoon(0).Left >= (DemoForm.Width - imgMoon(0).Width)
Then
    Motion = 4
    Elseif imgMoon(0).Top >= (DemoForm.Height -
imgMoon(0).Height) - 680 Then
        Motion = 2
    End If
Case 4
    imgMoon(0).Move imgMoon(0).Left - 100, imgMoon(0).Top +
100
    IncrFrame
    If imgMoon(0).Left <= 0 Then
        Motion = 3
    Elseif imgMoon(0).Top >= (DemoForm.Height -
imgMoon(0).Height) - 680 Then
        Motion = 1
    End If
End Select
End Sub

```

```
Private Sub IncrFrame()
    FrameNum = FrameNum + 1
    If FrameNum > 8 Then
        FrameNum = 1
    End If
    imgMoon(0).Picture = imgMoon(FrameNum).Picture
    Me.Refresh
    Delay
End Sub

Private Sub LineCtlDemo()
    linLineCtl.X1 = Int(DemoForm.Width * Rnd)
    linLineCtl.Y1 = Int(DemoForm.Height * Rnd)
    linLineCtl.X2 = Int(DemoForm.Width * Rnd)
    linLineCtl.Y2 = Int(DemoForm.Height * Rnd)
    Cls
    Delay
End Sub

Private Sub LineDemo()
    Dim X2
    Dim Y2
    R = 255 * Rnd
    G = 255 * Rnd
    B = 255 * Rnd
    X2 = Int(DemoForm.Width * Rnd + 1)
    Y2 = Int(DemoForm.Height * Rnd + 1)
    Line -(X2, Y2), RGB(R, G, B)
End Sub

Private Sub mnuCircleDemo_Click()
    Cls
    mnuCtlMoveDemo.Checked = False
    mnuImageDemo.Checked = False
    mnuScaleDemo.Checked = False
    mnuCircleDemo.Checked = True
End Sub

Private Sub mnuCtlMoveDemo_Click()
    Cls
    mnuCtlMoveDemo.Checked = True
    mnuImageDemo.Checked = False
```

```
mnuScaleDemo.Checked = False
mnuCircleDemo.Checked = False
End Sub
Private Sub mnuExit_Click()
    End
End Sub

Private Sub mnulmageDemo_Click()
    Cls
    mnuCtlMoveDemo.Checked = False
    mnulmageDemo.Checked = True
    mnuScaleDemo.Checked = False
    mnuCircleDemo.Checked = False
End Sub
Private Sub mnuLineCtlDemo_Click()
    Cls
    mnuCtlMoveDemo.Checked = False
    mnuLineDemo.Checked = False
    mnuShapeDemo.Checked = False
    mnuPSetDemo.Checked = False
    mnuLineCtlDemo.Checked = True
    mnulmageDemo.Checked = False
    mnuScaleDemo.Checked = False
    mnuCircleDemo.Checked = False
End Sub
Private Sub mnuLineDemo_Click()
    Cls
    mnuCtlMoveDemo.Checked = False
    mnuLineDemo.Checked = True
    mnuShapeDemo.Checked = False
    mnuPSetDemo.Checked = False
    mnuLineCtlDemo.Checked = False
    mnulmageDemo.Checked = False
    mnuScaleDemo.Checked = False
    mnuCircleDemo.Checked = False
End Sub
Private Sub mnuPSetDemo_Click()
    Cls
```

```
mnuCtlMoveDemo.Checked = False
mnuLineDemo.Checked = False
mnuShapeDemo.Checked = False
mnuPSetDemo.Checked = True
mnuLineCtlDemo.Checked = False
mnuImageDemo.Checked = False
mnuScaleDemo.Checked = False
mnuCircleDemo.Checked = False
End Sub
Private Sub mnuScaleDemo_Click()
    Cls
    mnuCtlMoveDemo.Checked = False
    mnuImageDemo.Checked = False
    mnuScaleDemo.Checked = True
    mnuCircleDemo.Checked = False
End Sub
Private Sub mnuShapeDemo_Click()
    Cls
    mnuCtlMoveDemo.Checked = False
    mnuLineDemo.Checked = False
    mnuShapeDemo.Checked = True
    mnuPSetDemo.Checked = False
    mnuLineCtlDemo.Checked = False
    mnuImageDemo.Checked = False
    mnuScaleDemo.Checked = False
    mnuCircleDemo.Checked = False
End Sub
Private Sub PSetDemo()
    R = 255 * Rnd
    G = 255 * Rnd
    B = 255 * Rnd
    XPos = Rnd * ScaleWidth
    YPos = Rnd * ScaleHeight
    PSet (XPos, YPos), RGB(R, G, B)
End Sub
Private Sub ScaleDemo()
    Dim Box
    For Box = 1 To ScaleWidth
```

```

R = 255 * Rnd
G = 255 * Rnd
B = 255 * Rnd
Line (Box, 0)-Step(1, (Int(11 * Rnd))), RGB(R, G, B), BF
Next Box
Delay
End Sub
Private Sub ShapeDemo()
Dim ClonelD
R = 255 * Rnd
G = 255 * Rnd
B = 255 * Rnd
DemoForm.BackColor = RGB(R, G, B)
CloneID = Int(20 * Rnd + 1)
XPos = Int(DemoForm.Width * Rnd + 1)
YPos = Int(DemoForm.Height * Rnd + 1)
shpClone(CloneID).Shape = Int(6 * Rnd)
shpClone(CloneID).Height = Int(2501 * Rnd + 500)
shpClone(CloneID).Width = Int(2501 * Rnd + 500)
shpClone(CloneID).BackColor = QBColor(Int(15 * Rnd))
shpClone(CloneID).DrawMode = Int(16 * Rnd + 1)
shpClone(CloneID).Move XPos, YPos
shpClone(CloneID).Visible = True
Delay
End Sub
Private Sub Timer1_Timer()
If mnuCtlMoveDemo.Checked And DoFlag = True Then
    CtlMoveDemo
ElseIf mnuImageDemo.Checked And DoFlag = True Then
    ImageDemo
ElseIf mnuScaleDemo.Checked And DoFlag = True Then
    ScaleDemo
ElseIf mnuCircleDemo.Checked And DoFlag = True Then
    CircleDemo
End If
End Sub

```

Մենյուի ստեղծման օրինակներ դիտարկենք նաև տվյալների բազաների հետ աշխատանքի ժամանակ:

2. 21. Աշխատանքը տվյալների բազաների հետ

Չահագործողի տեսանկյունից տվյալների բազան դինամիկ այլուսակների տեսակ է, որոնց կառուցվածքը որոշվում է նրանց դաշտերով (սյուներով), իսկ պարունակությունը տվյալների հավաքածուով (տողերով): Տվյալների բազայում ինֆորմացիան սովորաբար ներկայացվում է իրար հետ կապված այլուսակների տեսքով: Տվյալների բազայի այլուսակում այլուները համապատասխանում են տվյալների բազայի դաշտերին, իսկ տողերը պարունակում են տվյալների առանձին գրվածքներ: Այլուսակներում գրվածքների փնտրման արագացման համար դաշտերին վերագրվում են ինդեքսներ: Դա նշանակում է, որ այլուսակի որոշակի դաշտի կամ դաշտերի համար ստեղծվում է պատճեն, որը պարունակում է տրված ինդեքսի արժեքը, և դրվում է կարգավորված և չկարգավորված սյուների գրվածքների միջև փոխկապակցվածություն: Այսպիսի ինֆորմացիան պահպում է հատուկ ինդեքսային ֆայլերում կամ այլուսակների հետ ընդհանուր ֆայլում (օրինակ Ms Access-ում.mdb ընդլայնումով ֆայլում): Տվյալների ուլյացիոն բազայի տարբերիչ նշանը հանդիսանում է մի քանի այլուսակների վրա դեկավարվող ինֆորմացիայի բաշխման հնարավորությունը: Այդ ժամանակ այլուսակների միջև կապը իրականացվում է բանալիային դաշտերի օգնությամբ, որոնք պետք է ինդեքսավորվեն և կարող է կազմված լինեն մի քանի դաշտերից: Դաշտերի հետ տարբեր մանիպուլացիաներ (ընտրանք, նորացում, ավելացում, հեռացում և այլն) իրականացվում է SQL (Structured Query Language)-կառուցվածքավորված հարցումների լեզու) լեզվի օգնությամբ: Այն դարձել է ստանդարտ տվյալների բազայի մեծամասնությունում: Ms Access-ի օբյեկտային մոդելը կառուցվածքավորված է այնպիսի ձևով, որպեսզի ընդգրկի իր բոլոր օբյեկտները հիերարխիայի մեջ, որը գլխավորում է Application օբյեկտը: Բոլոր օբյեկտները և Ms Access-ի ընտանիքները հանդիսանում են նրա անդամներ: Ms Access-ի հավելվածը ունի իր օբյեկտների հիերարխիան: Application օբյեկտը ծառայում է նրա համար, որպեսզի կիրառի բոլոր մեթոդները և հատկությունները Ms Access-ի բոլոր հավելվածների նկատմամբ: Application օբյեկտը, ինչպես նաև հիերարխիայի մնացած օբյեկտները կարող են պաշտպանել մեթոդները և հատկությունները: Microsoft Access ՏԲԴ-ն հանդիսանում է 32 կարգանի տվյալների ուլյացիոն

բազայի դեկավարման համակարգի նոր սերունդ: Շահագործողի հավելվածների ստեղծման համար կարելի է օգտվել Visual Basic ծրագրավորման լեզվով գրված մակրոսներից և մոդուլներից: Visual Basic-ում տվյալների բազայի հետ կապելու պարզագույն միջոցը դեկավարման էլեմենտների գոտու Data էլեմենտի օգտագործումն է:

SQL -ի կառուցվածքը և կազմը

SQL-ը բավականին հզոր լեզու է ՏԲԴ-ի հետ փոխազդեցության համար: Սակայն SQL-ը ամբողջական համակարգչային (ծրագրավորման) լեզու չէ, ինչպիսին Cobol, Fortran, Pascal, Java, Visual Basic-ը կամ C++-ը: Ինքնուրույն SQL-ը չի հանդիսանում ո՞չ ՏԲԴ, ո՞չ էլ առանձին ծրագրային արտադրանք: Տվյալների բազայի միջուկը հանդիսանում է ՏԲԴ-ի սիրտը. այն պատասխանում է ֆիզիկական կառուցվածքավորվածության և սկավառակի վրա տվյալների գրանցման, ինչպես նաև սկավառակից տվյալները ֆիզիկապես կարդալու համար: Բացի դրանից, այն ընդունում է SQL հարցումները ՏԲԴ-ի ուրիշ բաղկացուցիչներից, ինչպիսիք են ֆորմաների ստեղծող (գեներատոր), հաշվետվությունների գեներատոր կամ էլ ինտերակտիվ հարցումների ծևավորման մոդուլ շահագործողի հավելվածից և նույնիսկ այլ հաշվողական համակարգերից: SQL-ը կատարում է շատ տարրեր ֆունկցիաներ:

Տվյալների կազմակերպումը: SQL-ը շահագործողին հնարավորություն է տալիս փոփոխելու տվյալների ներկայացման կառուցվածքը, ինչպես նաև նշելու տվյալների բազայի էլեմենտների հարաբերությունը:

Տվյալների կարդալը: SQL-ը շահագործողին կամ հավելվածին հնարավորություն է տալիս կարդալ տվյալների բազայից նրա մեջ պարունակվող տվյալները և օգտվել դրանցից:

Տվյալների մշակումը: SQL-ը շահագործողին կամ հավելվածին հնարավորություն է տալիս փոփոխել տվյալների բազան, ավելացնել նրանում նոր տվյալներ, ինչպես նաև հեռացնել կամ նորացնել արդեն առկա տվյալները:

Դիմելու դեկավարումը: SQL-ի օգնությամբ կարելի է սահմանափակել շահագործողի կարդալու և գրելու հնարավորությունները և պաշտպանել դրանց չարտոնված դիմելուց:

Տվյալների համատեղ օգտագործում: SQL-ը կորոդինացնում է շահագործողների միջև տվյալների համատեղ օգտագործումը զուգահեռ աշխատելու համար, որպեսզի նրանք չխանգարեն մեկը մյուսին:

Տվյալների ամբողջությունը: SQL-ը թույլ է տալիս ապահովել տվյալների բազայի անբողջականությունը պաշտպանելով նրան անհարկի փոփոխությունների պատճառով խաթարումից:

Ժամանակակից ռելյացիոն արտադրանքների մեծամասնությունը պաշտպանում է SQL ստանդարտ լեզվի տարբերակներից մեկը: Այն օգտագործվում է ռելյացիոն գործողությունների նկարագրման համար:

Արտաքին տվյալների հետ աշխատանքի ժամանակ օգտագործվում է հաճախորդ-սերվեր տեխնոլոգիան: Այն հավելվածը բաժանում է երկու մասի օգտագործելով երկու կողմերի լավագույն հատկությունները: Front-End (հաճախորդի մասը) ապահովում է ինտերակտիվ, օգտագործման մեջ պարզ, որպես օրենք գրաֆիկական ինտերֆեյս և տեղադրվում է շահագործողի համակարգչի վրա: Back-End (սերվեր) ապահովում է տվյալների դեկավարումը, ինֆորմացիայի բաժանումը, ծկուն վարչարարությունը, անվտանգությունը և տեղադրում է հատուկ առանձնացված համակարգիչների վրա: Հաճախորդ-սերվեր տեխնոլոգիայում հաճախորդի հավելվածը ծևավորում է հարցում SQL -ի սերվերին, որի վրա կատարվում են բոլոր հրամանները: Հրամանների կատարման արդյունքներն ուղարկվում են հաճախորդին օգտագործելու և դիտելու համար: Microsoft SQL սերվերը ներկայում ՏԲ-ի հարք սերվերներից մեկն է: MS SQL սերվերների հետ կապի և տվյալների դիմելու համար համընդհանուր ճանաչման է արժանացել ODBC (տվյալների անմիջական դիմում) տեխնոլոգիան: ODBC-ն լեզվի ընդհանուր սահմանումն է և կանոնակարգերի հավաքածուն: ODBC-ն թույլ է տալիս հաճախորդի հավելվածին, օրինակ Access-ին կամ Visual Basic-ին, աշխատել սերվերի կողմից պաշտպանվող հրամանների և ֆունկցիաների հետ: Որպես սերվերային կողմ կարող է հանդես գալ SQL-ի ցանկացած սերվեր, որն ունի ODBC (MS SQL Server, Oracle և այլն) դրայվերներ: Visual basic-ն ունի MS SQL Server-ների հետ ամենամեծ թվով փոխադրեցության միջոցները: SQL-ի հրամանների հավաքածուի օգնությամբ կարող ենք փոխել և հեռացնել տվյալները, ինչպես

Նաև ստեղծել և ձևափոխել նոր աղյուսակներ՝ այդ թվում որպես շաբլոն ունենալով նախապես ստեղծած հարցումներ:

SQL (Structured Query Language) հարցումների կառուցվածքավորված լեզուն բաժանվում է 6 խմբի:

1. DDL—Data Definition Language (տվյալների որոշման լեզու)
Create Table (ստեղծել աղյուսակ)
Alter Table (փոփոխել աղյուսակը)
Drop Table (հեռացնել աղյուսակը)
Create Index (ստեղծել ինդեքս)
Drop Index (հեռացնել ինդեքսը)
2. DML—Data Manipulation Language (տվյալների մանիպուլացիայի լեզու)
Insert (նեցնել)
Update (նորացնել)
Delete (հեռացնել)
3. DQL—Data Query Language (տվյալների հարցումների լեզու)
Select (ընտրել)
4. DCL—Data Access Management Language (տվյալների հետ գործելու կառավարման լեզու)
Alter Password (փոխել Ծանաբառը)
Grant (շնորհել)
Revoke (չեղյալ համարել)
5. DAC—Data Administering commands (տվյալների ադմինիստրավորման լեզու)
Start Audit (սկսել վերստուգումը)
Stop Audit (դադարեցնել վերստուգումը)
6. TCC—Transaction Executing Commands (փոխագործողությունների իրականացման լեզու)
Commit (փոխանցել, իրականացնել, հաստատել)
Rollback (հետ վերադառնալ նախկին վիճակին)

Ամենահաճախ կիրառվող SELECT հրամանն է:

Տվյալների բազայի հետ աշխատանքի ուսումնասիրումը դիտարկենք կոնկրետ օրինակով:

Առաջադրանք N1

Ղեղատան համար ստեղծել տեղեկատվական համակարգ՝ որոշակի տարածաշրջանի մասշտաբներում։ Համակարգը հիմնված է տվյալների բազայի վրա, որի հիմնական օբյեկտներն են դեղատները և դեղերը։ Դրանք առանձին օբյեկտներ են, որոնք բնութագրվում են իրենց համապատասխան ռեկվիզիտներով, սակայն որպես առանձին օբյեկտներ՝ անկախ չեն։ Նրանց միջև կա կախվածություն։ Բազան ինֆորմացիա է պարունակում դեղատների և այդ դեղատներում առաջարկվող դեղերի ու դեղամիջոցների մասին։ Համակարգը նախագծվում է այնպես, որ հնարավորություն տա ինֆորմացիան հեշտությամբ մուտքագրել տվյալների բազա ֆորմաների միջոցով, անհրաժեշտության դեպքում խնդրագրել այդ տվյալները, կատարել փնտրում, մասնավորապես ըստ դեղերի և դեղատների անվանումների, ավտոմատ կերպով ստանալ հաշվետվություն, մասնավորապես ժամկետանց դեղերի և պիտանելիության ժամկետի ավարտին մոտեցած դեղերի մասին։ Ընդ որում, փնտրման համակարգը նախատեսված է այնպես, որ պարտադիր չէ, որ հաճախորդը ծշտորեն իմանա դեղամիջոցի կամ դեղատան անվանումը։ Այսինքն, ընտրելով մենյուի համապատասխան կետերը, կարող ենք ստանալ երկու տիպի ցուցակներ ժամկետանց դեղերի ցուցակը և այն դեղերի ցուցակը, որոնց որոշակի կարծ ժամանակահատվածը է մնում մինչև պիտանելիության ժամկետի լրանալը։ Այդ ժամանակահատվածը կարելի է սահմանել օրինակ 1 ամիս, և այդ ցուցակները ստացվում են համակարգչի համակարգային Timer-ի ընթացիկ ամսաթվից հանելով պիտանելիության ժամկետի ամսաթիվը։ Համակարգի նպատակն է որոշակի ինֆորմացիոն ծառայություն մատուցել դեղատան հաճախորդներին, մասնավորապես նրանց տեղեկացնել, թե որ դեղատնից ինչ գնով կարելի է ծեռք բերել անհրաժեշտ դեղը, ինչպես կարելի է կապվել ընտրած դեղատան հետ և իմանալ այդ դեղատան հասցեն։

Բազայի առյուսակները ունեն հետևյալ ռեկվիզիտները։

Դեղատներ

- դեղատան անվանումը
- դեղատան հասցեն
- դեղատան հեռախոսի համարը

- E-mail
- Fax
- տնօրենի ազգանունը, անունը:

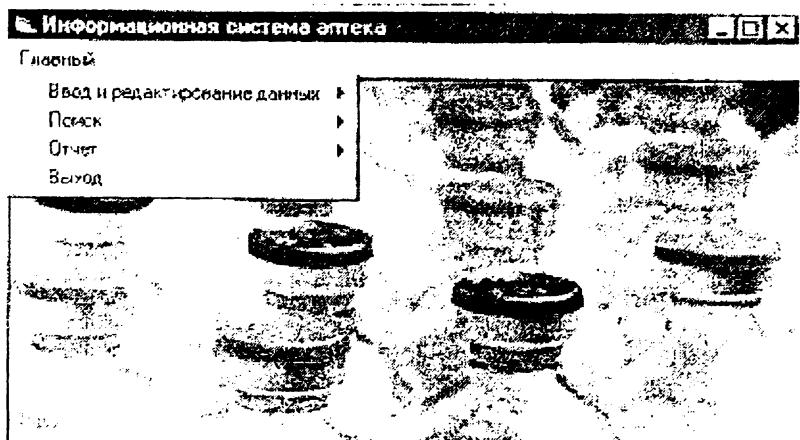
Դեղեր՝

- դեղի կողը
- դեղի անվանումը
- գինը
- արտադրության ամսաթիվը
- պիտանելիության ժամկետը
- ֆիզիկական տեսքը (հար, քսուք...)
- ազդեցության ուղղվածությունը (ցավազրկող, քնաբեր...):

Ուկվիզիտները որոշելուց հետո տվյալների բազան կազմակերպվում է MS Access ՏԲԴ-ով, որը բաղկացած է երկու աղյուսակներից, որոնք նորմալացված են և անհրաժեշտության դեպքում այդ աղյուսակների հիման վրա կարելի է հարցումների օգնությամբ ստանալ նոր աղյուսակներ: Աղյուսակները մինյանց կապված են մեկը բոլորին կապով, կապը իրագործվում է DrugstoreID ռեկվիզիտի միջոցով, որը դեղատների աղյուսակում բանալիային է և միանշանակ բնորոշում է դեղատներին: Դա նշանակում է, որ այն չի կարող կրկնվել այդ աղյուսակում:

Դամակարգի գլխաղաս մենյուն և հիմնական աշխատանքային ռեժիմները

Visual basic-ով ծրագիրն աշխատեցնելիս բացվում է հիմնական ֆորման՝



Ելնելով համակարգի առջև դրված պահանջներից՝ ձևակերպում են չորս հիմնական աշխատանքային ռեժիմներ, որոնցից երեքը իրենց հերթին ունեն ենթառեժիմներ՝

1. տվյալների մուտքագրում և խմբագրում
2. փնտրում
3. հաշվետվություն
4. ելք:

Առաջին աշխատանքային ռեժիմը բաղկացած է երկու ենթառեժիմներից:

Այս ենթառեժիմներում իրականացվում է տվյալների մուտքագրումը տվյալների բազա:

1. դեղատներ. հնարավորություն է ստեղծում իրականացնել տվյալների մուտքագրում Dragstore աղյուսակում կամ փոփոխության ենթարկել եղած տվյալները,
2. դեղեր. հնարավորություն է ստեղծում իրականացնել տվյալների մուտքագրում Medicine աղյուսակում կամ փոփոխության ենթարկել եղած տվյալները:

Երկրորդ աշխատանքային ռեժիմը նույնպես բաղկացած է երկու ենթառեժիմներից:

1. դեղերի որոնում. հնարավորություն է ստեղծում իրականացնել դեղամիջոցների որոնումը տվյալների բազայում,
2. դեղատների փնտրում. հնարավորություն է ստեղծում իրականացնել դեղատների փնտրումը տվյալների բազայում:

Երրորդ աշխատանքային ռեժիմն էլ բաղկացած է երկու ենթառեժիմներից

1. ժամկետանց դեղեր. ներկայացնում է բոլոր ժամկետանց դեղերի ցուցակը
2. պիտանելիության ժամկետի ավարտին մոտ դեղեր. ներկայացնում է բոլոր պիտանելիության ժամկետի ավարտին մոտեցած դեղերի ցուցակը:

Վերջին աշխատանքային ռեժիմը ապահովում է ելքը ծրագրից:

Համակարգի մուտքային և ելքային ինֆորմացիան

Համակարգի մուտքային ինֆորմացիան իրենից ներկայացնում է բազայում նկարագրված ռեկվիզիտներին համապատասխան ինֆորմացիա:

Գլխադաս մենյուում «Տվյալների մուտքագրում և խմբագրում» ռեժիմը և ենթառեժիմները ընտրելիս էկրան են դուրս բերվում մուտքագրման ֆորմաները.

Ա. Առաք

Դեղատեսներ

Անդամական անվանումը	
Անդամական հասցե	
Հեռախոս	
Ֆաք	
E-Mail	
Տեղաբնիք	

[◀ ▶ Ռեզերվացիա] [ՀԱՇԹ ՄՈՋՈՂ] [Ապելացնել] [ԵԼՔ]

Ա. Առաքը

Դեղերի օլիգոն

Դեղատան Կոդը	Դեղի Կոդը
Դեղի անվանումը	
Գինը դրամով	
Արտադրության ամսաթիվը	
Պիտակի և ծինչներ	
Տեսակը	
Ազդեցության ուղղութը	

[◀ ▶ Ռեզերվացիա] [ՀԱՇԹ ՄՈՋՈՂ] [Ապելացնել] [ԵԼՔ]

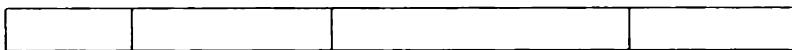
Ելքային փաստաթղթերը երկուսն են՝

- Ժամկետանց դեղերի ցուցակը
- պիտանելիության ժամկետի ավարտին մոտ դեղերի ցուցակը:

Առաջինն ունի հետևյալ տեսքը՝

Ժամկետանց դեղերի ցուցակ

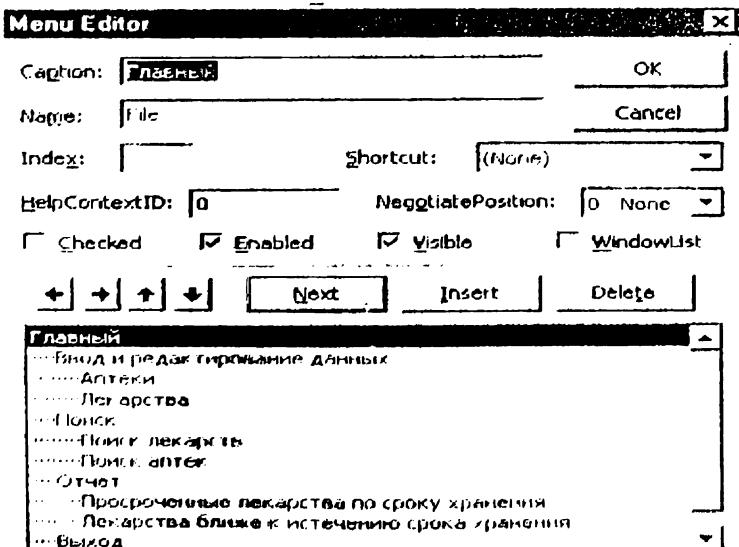
Դեղի կողը, Դեղի անվանումը, Դեղառան անվանումը, Դիտանի է մինչև



Երկրորդն ունի նույն տեսքը, սակայն այլ պարունակությամբ:

Գլխադաս մենյուի և աշխատանքային ռեժիմներից մի քանիսի ծրագրային իրականացումը

Ծրագրի մենյուն կազմակերպելու համար Visual Basic-ը հնարավորություն է ստեղծում կազմակերպել մենյուն Menu Editor-ի օգնությամբ, որը գտնվում է Tools մենյուում։ Այն ընտրելիս բացվում է հետևյալ պատուհանը։



Menu Editor – ի օգնությամբ ֆլեյփրոլում է մենյուի միայն կառուցվածքը: Որպեսզի աշխատեմ մենյուի ռեժիմները, անհրաժեշտ է նրանց համար գրել համապատասխան կոդ:

Կոդը հետևյալն է:

```
Private Sub Open_Drugstore_Click()
Drugstore.Show
End Sub
(Եկրանին դուրս կբերի Դեղատներ մուտքագրման ֆորման)
Private Sub Open_Medicines_Click()
Medicine.Show
End Sub
(Եկրանին դուրս կբերի Դեղերի մուտքագրման ֆորման)

Private Sub View_med_Click()
FrmFind.Show
End Sub
(Եկրանին դուրս կբերի փնտրման համար նախատեսված ֆորման)

Private Sub Exit_Click()
End
End Sub
(Ավարտում է ծրագրի աշխատանքը)
```

Դեղատներ մուտքագրման ռեժիմը

Այս ռեժիմն աշխատեցնելիս էկրանին բացվում է «Դեղատներ» (Drugstore) ֆորման:

Ֆորմայում տեղադրվում են հետևյալ օբյեկտները՝ հետևյալ հատկություններով.

Data

Name	Drugstore
Connect	Access
DatabaseName	C:\My Documents\My Own\Kursajin\Kursajin.mdb
RecordSource	Drugstore

Յոթ հատ TextBox (համապատասխան Label-ներով), որոնցից յուրաքանչյուրը համապատասխանում է բազայում աղյուսակի ռեկվիզիտներին.

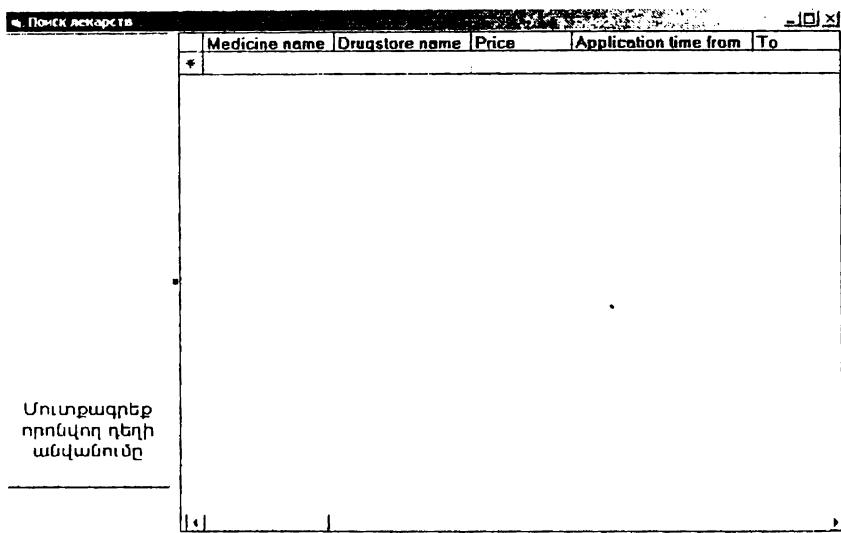
DataSource Drugstore
DataField

* յուրաքանչյուր TextBox-ի համար DataField դաշտում ընտրվում է աղյուսակի համապատասխան ռեկվիզիտը:

Ֆորմայում տեղադրվում է նաև երեք CommandButton՝ հետևյալ կոդերով.

```
Private Sub CmdAdd_Click()
Prompt$="Դուք ցանկանո՞՞մ եք ավելացնել նոր գրառում"
Reply= MsgBox(prompt$,vbYesNo,"Նոր գրառում")
If reply=vbYes Then
Text1.SetFocus
Drugstore.Recordset.AddNew
End If
End Sub
(Նոր գրառում)
Private Sub CmdClose_Click()
Unload Me
End Sub
(Փակում է ֆորման)
Private Sub CmdDelete_Click()
On Error Go To DeleteErr
Reply= MsgBox("Դուք ցանկանո՞՞մ եք ջնջել գրառումը", vbYesNo,
"Գրառման ջնջում")
If reply = vbYesNo Then
With Drugstore.Recordset
.Delete
.MoveNext
If.EOF Then.MoveLast
End With
End If
Exit Sub
DeleteErr:
MsgBox Err.Description
End Sub
(Ջնջում է գրառումը)
```

Որոնման ռեժիմ



Այս ռեժիմը աշխատեցնելիս էկրանին բացվում է որոնման ֆորման:

Այս ֆորմային համապատասխանում է հետևյալ կոդը.

```
Private Sub DataList1_Click()
Dim SQLStr As String
SQLStr = "SELECT Medicine.[Medicine name], Drugstore.[Drugstore name], Medicine.Price, Medicine.[Application time form], Medicine.To FROM Drugstore INNER JOIN Medicine ON Drugstore.DrugstoreId = Medicine.DrugstoreId"
SQLStr = SQLStr & "Where [Medicine name] =" & DataList1.Text
Data1.RecordSource = SQLStr
Data1.Refresh
TsutsakGrid.Visible = True
End Sub
Private Sub Text1_Change()
DataList1.BoundText = Text1.Text
End Sub
```

Առաջադրանք №2

Ստեղծել բանկային ոլորտում դեպոզիտային գործառույթների ավտոմատացման համար ինֆորմացիոն համակարգ:

Խնդրի նպատակն է մուտքագրել և դիտել գործառնությունների և անձերի վերաբերյալ ինֆորմացիա, ստանալ ամփոփ տվյալներ: Խնդրի իրականացման համար անհրաժեշտ է կառուցել տվյալների բազա, որը բաղկացած է երկու՝ deposit և ands աղյուսակներից: Deposit աղյուսակը բաղկացած է հետևյալ ռեկվիզիտներից:

GorcKod - գործառնության կող

GorcTip - գործառնության տիպ (երկարաժամկետ, կարճաժամկետ, ցապահանջ և միջբանկային դեպոզիտներ)

AndcKod - այն անձի կողը, որը ներդրել է որոշակի տիպի դեպոզիտ Hhhamar- անձի հատուկ համարը բանկում

Gumar1000dr-դեպոզիտի գումարը արտահայտված հազար դրամով

Fdate-դեպոզիտի ներդրման սկզբնական ամսաթիվը

Edate-դեպոզիտների վերադարձման ամսաթիվը

Vtip-վերադարձման տիպը միանվագ կամ պարբերաբար

Tokos-պայմանագրային տոկոսը

Երկրորդ Ands աղյուսակը բաղկացած է հետևյալ ռեկվիզիտներից.

AndsKod-Անձի կողը

Առստ-անձի անունը

Status-անձի կարգավիճակը իրավաբանական կամ ֆիզիկական

Erkir-այն երկիրը, որտեղ գրանցված է տվյալ անձը

Qaxaq-քաղաքը

Hasce-հասցեն

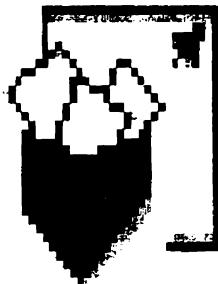
Post-փոստի համարը

Her-հեռախոսի համարը:

Visual Basic-ում գլխադաս մենյուն, որի միջոցով մուտք է գործվում նախազծի տարրեր բլոկներ, ունի հետևյալ տեսքը.

FrmSplash

Կուրսային Աշխատանք



Դեպոզիտային Բանկ

Depositnei
Mutqagrum

Andsi
Mutqagrum

Amporum

Hashvelvutum

Exit

Ներկայացնենք գլխադաս Եջի իրազործման ժրագիրը, որը ընդգրկում է նաև հաշվետվության ներկայացման ժրագիրը:

```
Option Explicit
Private Sub command1_click()
Form1.Show
End Sub
Private Sub command2_click()
Form2.Show
End Sub
Private Sub command3_click()
Form3.Show
End Sub
Private Sub command4_click()
Dim appWord As New Word.Application
Dim DepPath As String
Load Form3
DepPath = App.Path & "\Report.dot"
With appWord
Documents.Add DepPath
.ActiveDocument.ShowSpellingErrors = False
.Selection .GoToWdGoToBookmark, Name: ="ErkarQanak"
.Selection.TypeText Form3!label3
.Selection .GoToWdGoToBookmark, Name: ="KargQanak"
End With
End Sub
```

```

.Selection.TypeText Form3!label4
.Selection .GoToWdGoToBookmark, Name: ="CpQanak"
.Selection.TypeText Form3!label5
.Selection .GoToWdGoToBookmark, Name: ="MbQanak"
.Selection.TypeText Form3!label6
.Selection .GoToWdGoToBookmark, Name: ="ErkarGumar"
.Selection.TypeText Form3!label7
.Selection .GoToWdGoToBookmark, Name: ="KargGumar"
.Selection.TypeText Form3!label8
.Selection .GoToWdGoToBookmark, Name: ="CpGumar"
.Selection.TypeText Form3!label9
.Selection .GoToWdGoToBookmark, Name: ="MbGumar"
.Selection.TypeText Form3!label10
.Selection .GoToWdGoToBookmark, Name: ="SumQanak"
.Selection.TypeText Val(Form3!label3)+Val(Form3!label4)+  

Val(Form3!label5) + Val(Form3!label6)
.Selection .GoToWdGoToBookmark, Name: ="SumGumar"
.Selection.TypeText Val(Form3!label7)+Val(Form3!label8)+  

Val(Form3!label9) + Val(Form3!label10)
.Visible = True
End With
End Sub
Private Sub command5_click()
End
Unload Form1
Unload Form2
Unload Form3
Unload frmSplash
End Sub
Private Sub Form_KeyPress(KeyAscii As Integer)
Unload Me
End Sub
Private Sub Frame1_Click()
Unload Me
End Sub

```

Սեղմելով գլխադաս եցի առաջին սեղմակը եկրանին հայտնվում է դեպոզիտների վերաբերյալ տվյալների մուտ-

քագրման համար նախատեսված մուտքային էկրանային ձևապատկերը ֆորման, որը ներկայացված է ստորև:

Form1

◀ Դեպոզիտային տվյալների մուտքագրում

Գործառնության համար	1
Աճածի կոդ	12
Պորժաօնության տիպ	Ցանկաց.
Դատուկ հաշվի համար	451
Գումար	3000 000դրամ
Սկզբ. աճսարիկ	1210.00
Վերջ. աճսարիկ	
Ժամկետը	
Վերադարձման տիպ	<input checked="" type="radio"/> Մեկ անգամից <input type="radio"/> Պարբերաբար
Տոկոսը	2
<input type="button" value="More>>"/> <input type="button" value="SaveRecord"/> <input type="button" value="Del.Record"/>	
<input type="button" value="First"/> <input type="button" value="Previous"/> <input type="button" value="Next"/> <input type="button" value="Last"/> <input type="button" value="Close"/>	

Դետայալ ծրագիրը իրագործում է առաջին ձևապատկերը (ֆորման).

```
Private Sub command1_Click()
On Error GoTo errhandler
Data1.Recordset.MoveFirst
If Text3 = "" Or Text4 = "" Then
label5 = ""
Else
label5 = DateDiff("d", Text3, Text4)
```

```
End If
errhandler:
If Err = 13 Then
Resume Next
End If
End Sub
Private Sub command2_click()
On Error GoTo errhandler
Data1.Recordset.MovePrevious
If Text3 = "" Or Text4 = "" Then
label5 = ""
Else
label5 = DateDiff("d", Text3, Text4)
End If
If Data1.Recordset.BOF = True Then
Data1.Recordset.MoveFirst
End If
errhandler:
If Err = 13 Then
Resume Next
End If
End Sub
Private Sub command3_click()
On Error GoTo errhandler
Data1.Recordset.MoveNext
If Text3 = "" Or Text4 = "" Then
label5 = ""
Else
label5 = DateDiff("d", Text3, Text4)
End If
If Data1.Recordset.EOF = True Then
Data1.Recordset.AddNew
Data1.Recordset.Update
Data1.Recordset.MoveLast
Command6.Enabled = True
UpDown1.Visible = True
Text5 = 5
Text1.SetFocus
```

```
End If
Exit Sub
errhandler:
If Err = 13 Then
Resume Next
End If
End Sub
Private Sub command4_click()
On Error GoTo errhandler
Data1.Recordset.MoveLast
If Text3 = "" Or Text4 = "" Then
label5 = ""
Else
label5 = DateDiff("d", Text3, Text4)
End If
errhandler:
If Err = 13 Then
Resume Next
End If
End Sub
Private Sub command5_click()
Me.Move 800, 1400
Form2.Show
End Sub
Private Sub command6_click()
Data1.Recordset.Edit
Data1.Recordset("AndcKod") = Val(Text1)
Data1.Recordset("Gumar1000dr") = Val(Text2)
Data1.Recordset("Hhhamar") = Val(text6)
Data1.Recordset("Fdate") = Text3
Data1.Recordset("Edate") = Text4
Data1.Recordset("GorcTip") = Combo1.Text
If option1 Then
Data1.Recordset("Vtip") = "Parberabar"
End If
Data1.Recordset.Update
MsgBox "Your recordis saved!", vbInformation, "Attention"
End Sub
```

```
Private Sub command7_click()
Dim response As Integer
response = MsgBox("Are you sure?", vbQuestion + vbYesNo + vbDefaultButton2, "Attention")
If response = vbNo Then 'if selected "No" button
Exit Sub
Else 'if selected "yes" button
Data1.Recordset.Delete
Data1.Recordset.MovePrevious
End If
End Sub
```

```
Private Sub command8_click()
Form1.Hide
End Sub
```

```
Private Sub form_load()
Data1.DatabaseName = App.Path & ("\DepBank.mdb")
Data1.RecordSource = "Depozit"
Data1.Refresh
Combo1.AddItem "cpahang"
Combo1.AddItem "kargagamket"
Combo1.AddItem "erkaragamket"
Combo1.AddItem "migbankain"
Combo1.Refresh
Combo1.ListIndex = 0
End Sub
Private Sub UpDown1_DownClick()
If Text5 = 1 Then
MsgBox "The number must not be less than 1!", vbInformation,
"Attention"
Exit Sub
End If
Text5 = Text5 - 1
End Sub
Private Sub UpDown1_UpClick()
If Text5 = 10 Then
```

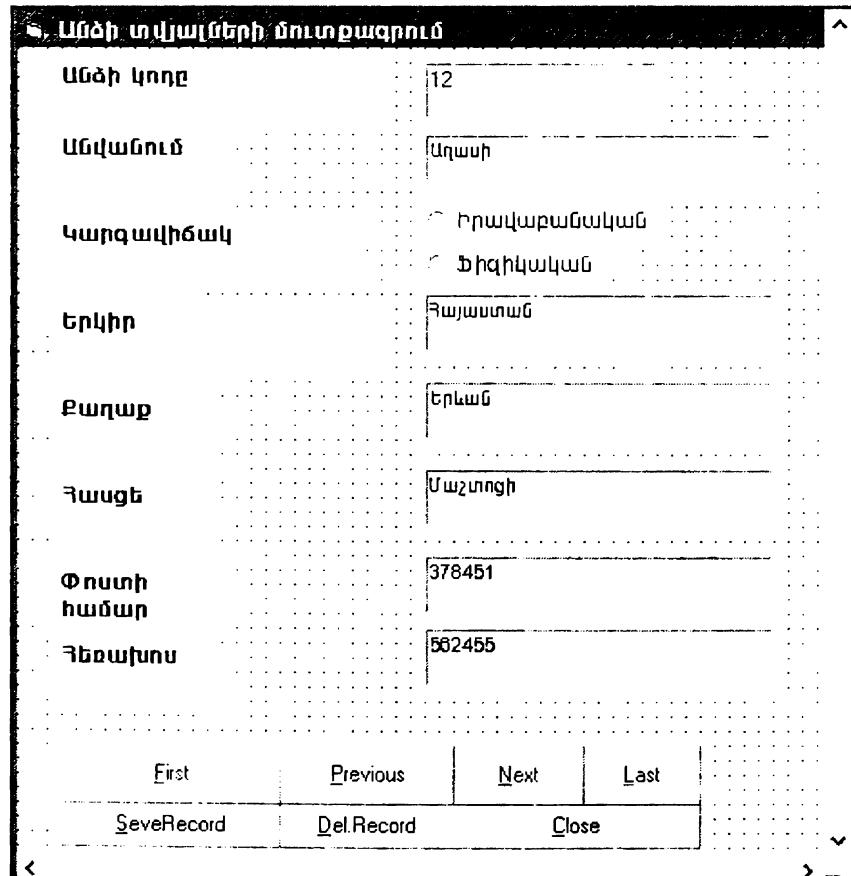
```

MsgBox "The number must not be more than 10!", vbInformation,
"Attention"
Exit Sub
End If
Text5 = Text5 + 1
End Sub

```

Form2

Այս ձևապատկերին (ֆորմային) կարելի է դիմել գլխաղաս էջում՝ սեղմելով երկրորդ ստեղնը: Ներկայացնենք և ֆորման, և համապատասխան ծրագիրը:



```
Private Sub command1_click()
On Error GoTo errhandler
Data1.Recordset.MoveFirst
errhandler:
Resume Next
End Sub
Private Sub command2_click()
On Error GoTo errhandler
Data1.Recordset.MovePrevious
If Data1.Recordset("Status") = "Iravabanakan" Then
option1 = True
Else
option2 = True
End If
If Data1.Recordset.BOF = True Then
Data1.recordset.movefirst
End If
errhandler:
Resume Next
End Sub
Private Sub command3_click()
Dim current As Currency
Dim bigest As Currency
On Error GoTo errhandler
Data1.Recordset.MoveNext
If Data1.Recordset("Status") = "Iravabanakan" Then
option1 = True
Else
option2 = True
End If
If Data1.Recordset.EOF = True Then
Data1.Recordset.AddNew
Data1.Recordset.Update
Data1.Recordset.MoveFirst
current = Data1.Recordset("AndsKod")
bigest = current
Do Until Data1.Recordset.EOF
Data1.Recordset.MoveNext
```

```
If Data1.Recordset("AndsKod")>current Then bigest=Data1.Recordset("AndsKod")
current = bigest
End If
Loop
Data1.Recordset.MoveLast
Label1 = bigest + 1
Text1.SetFocus
End If
Exit Sub
errhandler:
Resume Next
End Sub
Private Sub command4_click()
On Error GoTo errhandler
Data1.Recordset.MoveLast
errhandler:
Resume Next
End Sub
Private Sub command5_click()
Form1.Move 2650, 1400
Form2.Hide
Unload Form2
End Sub
Private Sub command6_click()
Data1.Recordset.Edit
Data1.Recordset("AndsKod") = Val(Label1)
Data1.Recordset("Anun") = Text1
Data1.Recordset("Erkir") = Text2
Data1.Recordset("Qaxaq") = Text3
Data1.Recordset("Hasce") = Text4
Data1.Recordset("Post") = Text5
Data1.Recordset("Her") = text6
If option1 Then
Data1.Recordset("Status") = "Iravabanakan"
Else
Data1.Recordset("Status") = "Fizikakan"
End If
```

```
Data1.Recordset.Update
MsgBox "Your record is saved!", vbInformation, "Attention"
End Sub
Private Sub command7_click()
Dim response As Integer
response = MsgBox("Are you sure?", vbQuestion + vbYesNo +
vbDefaultButton2, "Attention")
If response = vbNo Then 'if selected "No" button
Exit Sub
Else 'if selected "Yes" button
Data1.Recordset.Delete
Data1.Recordset.MovePrevious
End If
End Sub
Private Sub form_load()
Dim target As String
Me.move6700, 1400
Data1.DatabaseName = App.Path & ("\DepBank.mdb")
Data1.RecordSource = "Ands"
Data1.refrsh
target = "AndsKod=" & "" & Form1!Text1 & ""
Data1.Recordset.FindFirst target
Label1 = target
taxt1 = Data1.Recordset("Anun")
If Data1.Recordset("Status") = "Iravabanakan" Then
option1 = True
Else
option2 = True
End If
Text2 = Data1.Recordset("Erkir")
Text3 = Data1.Recordset("Qaxaq")
Text4 = Data1.Recordset("Hasce")
Text5 = Data1.Recordset("Post")
text6 = Data1.Recordset("Her")
End Sub
```

Form3-ով կարող ենք դիտել ֆորման, որի օգնությամբ կարող ենք ինանալ ներկա պահին մուտքագրված բոլոր դեպոզիտները՝ ըստ քանակի և գումարի: Ֆորման թարմացվում է ամեն անգամ այն բացելիս:

Ամփոփում

Դեպոզիտի տիպ	Քանակ	Գումար
Երկարաժամկետ	1	2000000
Լարձավամկետ	0	0
Ցպահանջ	1	3000000
Միջբանկային	0	0

Close

```

Private Sub command1_Click()
Form3.Hide
End Sub
Private Sub form_Load()
Dim target As String
Dim qanak As String
Dim summ As Currency
label2 = Date & " " & Time
Data1.DatabaseName = App.Path & ("\DepBank.mdb")

```

```
Data1.RecordSource = "Depozit"
Data1.Refresh
qanak = 0
summ = 0
target = "GorcTip='Erkarajamket'"
Data1.Recordset.FindFirst target
If Data1.Recordset.NoMatch Then
label3 = "0"
label7 = "0"
Else
qanak = qanak + 1
summ = summ + Data1.Recordset("Gumar1000dr")
Do
Data1.Recordset.FindNext target
If Not Data1.Recordset.NoMatch Then
qanak = qanak + 1
summ = summ + Data1.Recordset("Gumar1000dr")
Else
label3 = qanak
label7 = summ * 1000
qanak = 0
summ = 0
Exit Do
End If
Loop
End If
target = "GorcTip='Cpahang'"
Data1.ecordset.FindFirst target
If Data1.Recordset.NoMatch Then
label5 = "0"
label9 = "0"
Else
qanak = qanak + 1
summ = summ + Data1.Recordset("Gumar1000dr")
Do
Data1.Recordset.FindNext target
If Not Data1.Recordset.NoMatch Then
qanak = qanak + 1
```

```
summ = summ + Data1.Recordset("Gumar1000dr")
Else
label5 = qanak
label9 = summ * 1000
qanak = 0
summ = 0
Exit Do
End If
Loop
End If
target = "GorcTip='Karchajamket'"
Data1.Recordset.FindFirst target
If Data1.Recordset.NoMatch Then
label4 = "0"
label8 = "0"
Else
qanak = qanak + 1
summ = summ + Data1.Recordset("Gumar1000dr")
Do
Data1.Recordset.FindNext target
If Not Data1.Recordset.NoMatch Then
qanak = qanak + 1
summ = summ + Data1.Recordset("Gumar1000dr")
Else
label4 = qanak
label8 = summ * 1000
qanak = 0
summ = 0
Exit Do
End If
Loop
End If
target = "GorcTip='Mijbankain'"
Data1.Recordset.FindFirst target
If Data1.Recordset.NoMatch Then
label6 = "0"
label10 = "0"
Else
```

```

qanak = qanak + 1
summ = summ + Data1.Recordset("Gumar1000dr")
Else
label6 = qanak
label10 = summ * 1000
qanak = 0
summ = 0
Exit Do
End If
Loop
End If
End Sub

```

Հաշվետվությանը կարող ենք դիմել գլխադաս էջից՝ համապատասխանաբար չորրորդ սեղմակին սեղմելով։ Հաշվետվությունը ներկայացնում է առկա դեպոզիտների քանակը և գումարը ըստ նրա տիպի։ Այն պարունակում է նաև բոլոր տեսակի դեպոզիտների ընդհանուր գումարը։ Հաշվետվությունը իրականացված է Word-ի միջավայրում։

Report

Դեպոզիտի տիպ	Քանակ	Գումար
Երկարաժամկետ	4	5000000
Կարճաժամկետ	6	10000000
Ցպահանջ	2	8000000
Միջբանկային	5	6000000
Ընդամենը	17	29000000

2. 22. Visual Basic-ը և Windows API- ին

Եթե մեր հավելվածի պահանջնունքները դուրս են գալիս հիմնական լեզվային միջոցների և ստանդարտ էլեմենտների շրջանակից, օգնության է գալիս API-ն (Application programming Interface-կիրառական ծրագրերի հնտերֆեյսը): DLL (Dynamic Link Library-դինամիկ կապերի գրադարան) գրադարանին դիմումը ապահովում է դիմելու քույլտվություն Win32 մի քանի հազար համակարգային ֆունկցիաներին, ինչպես նաև այն պրոցեդուրաներին, որոնք գրված են այլ լեզուներով։ Ինչպես հուշում է հենց DLL անվանումը, այն իրենից ներկայացնում է պրոցեդուրաների գրադարան, որոնք, ի տարրերություն կոմպիլյացիայի փուլում ստատիկ կապերի, կցվում են հավելվածին նրա կատարման փուլում։ DLL-ի պարունակությունը չի ընդգրկում հավելվածի կատարվող ֆայլի մեջ։ Այդպիսով, DLL-ը կարող է նորացվել անկախ հավելվածից, իսկ գրադարանը կարող է միաժամանակ օգտագործվել միանգամից մի քանի հավելվածներում։ Microsoft Windows համակարգը նույնպես բաղկացած է DLL-ից։ Դավելվածը կանչում է այդ գրադարանների պրոցեդուրաները պատուհանների և գրաֆիկների հետ գործողությունների, հանակարգային ռեսուրսների դեկարտման, ռեեստրին (մատյանին) դիմելու ժամանակ։ Երբեմն այդ պրոցեդուրաները միավորվում են “Windows API” տերմինի տակ։ Visual Basic-ը ծրագրավորողից թաքցնում է Windows-ի համար ծրագրավորման շատ բարդություններ, բայց թույլ է տալիս դիմել Windows-ի միջոցներին։ Գոյություն ունեն ավելի քան 1000 API ֆունկցիաներ, որոնք պայմանականորեն բաժանվում են 4 հիմնական կատեգորիաների։

- աշխատանք հավելվածների հետ։ հավելվածի մեկնարկման և փակման ֆունկցիա, մենյուի հրամանի մշակման ֆունկցիա, պատուհանի չափերի փոփոխման և տեղաշարժման ֆունկցիա
- գրաֆիկա։ մետաֆայլերի ռաստրային ֆայլերի ստեղծման ֆունկցիա
- հանակարգչային ինֆորմացիա։ ընթացիկ դիսկի որոշման, հասանելի և ընդհանուր հիշողության ծավալի, ընթացիկ շահագործողի անունի, հանակարգչի օպերացիոն համակարգի ֆունկցիաներ և այլն

- ռեեստրի հետ աշխատանք. Windows-ի ռեեստրի հետ օպերացիաների համար ֆունկցիաներ, որոնք դուրս են գալիս Visual Basic-ի ներկառուցված GetSettings և SetSettings ֆունկցիաների շրջանակից ստեղծել, կարդալ և հեռացնել բաժինները և պարամետրերը:

DLL պրոցեդուրաները, որոնք կանչվում են Visual Basic-ում, պետք է նախապես հայտարարվեն: Դաստատման ժամանակ նշվում է DLL-ի անունը և API ֆունկցիան, ինչպես նաև փոխանցվում է ինֆորմացիա արգումենտների տիպի և քանակի մասին: DLL պրոցեդուրաները հայտարարվում են Declaration բաժնում ծրագրի պատուհանում Declare հրամանով: Եթե պրոցեդուրան վերադարձնում է արժեք, Declare հրամանը ձևակերպվում է ֆունկցիայի տեսքով:

Declare Function անուն Lib "գրադարան" [Alias "կեղծանում"] ([[[ByVal] փոփոխական [As տիպ] [, ByVal] փոփոխական [As տիպ]]...]) As տիպ:

Եթե արժեքը չի վերադարձվում, Declare հրամանը ձևակերպվում է պրոցեդուրայի տեսքով:

Declare Sub անուն Lib "գրադարան" [Alias "կեղծանում"] ([[[ByVal] փոփոխական [As տիպ] [, ByVal] փոփոխական [As տիպ]]...]) As տիպ:

DLL պրոցեդուրաները, որոնք գտնվում են ստանդարտ մոդուլներում, լրելայն հանդիսանում են բաց և կարող են կանչվել հավելվածի ցանկացած կետից: DLL պրոցեդուրաները, որոնք հայտարարվել են ցանկացած ուրիշ մոդուլում, հանդիսանում են փակ տվյալ մոդուլի համար, և նրանց հայտարարումից առաջ պետք է լինի Private բանալիային բառ:

Declare հրամանի Lib բաժինը հաղորդում է Visual Basic-ը, որը պետք է փնտրել DLL ֆայլում, որը պարունակում է պրոցեդուրան: Յիմնական գրադարանների դիմելու ժամանակ (User32, Kernel32, Gdi32 և այլն) ֆայլի ընդլայնումը ընդգրկելը պարտադիր չէ: Նշենք գրադարանի ֆայլի որոնման ժամանակ կատալոգները դիտելու ստանդարտ հաջորդականության նկարագրությունը, եթե ֆայլի ուղին նշված չէ:

1. Կատալոգ, որը պարունակում է ծրագրի Exe. ֆայլը
2. Ընթացիկ կատալոգ
3. Windows համակարգային կատալոգ (սովորաբար պարտադիր չէ-\windows\system)

4. Windows կատալոգը (սովորաբար \Windows)
5. Փոփոխական շրջապատ Path

Ստանդարտ գրադարանային ֆայլերը.

adnapi.dll – գրադարան, որը պաշտպանում է տարբեր API ոչ տրիվիալ ֆունկցիաներ, որոնց մեջ շատ ֆունկցիաներ կան անվտանգության և ռեսստրի հետ աշխատանքի համար:

comdlg.dll – ստանդարտ երկխոսության պատուհանների գրադարան,

Gdi32.dll – գրաֆիկական ինտերֆեյսի ֆունկցիաների գրադարան Kernel32.dll – Windows-ի միջուկի բազային 32 կարգանի ֆունկցիաների պաշտպանում

Lz32.dll – տվյալների սեղման 32 կարգանի ֆունկցիաներ

Mpr.dll – Multiple Provider Router գրադարան

Netapi32.dll – ցանցային API 32 կարգանի գրադարան

Shell32.dll – Shell API 32 կարգանի գրադարան

User32.dll – շահագործողական ինտերֆեյսի պրոցեդուրաների գրադարան

Version32.dll - վերսիաների հսկման գրադարան

Winmm.dll – Windows – ի մուլտիմեդիա գրադարան

Winspool.dll – տպելու դիսպենտերի ինտերֆեյս

API Viwer հավելվածի օգնությամբ գտնում ենք անհրաժեշտ պրոցեդուրան, պատճենում ենք կոդի հատվածը փոխանակման բոլոր և տեղադրում այն Visual Basic-ի մեր հավելվածում:

Visual Basic-ում բոլոր արգումենտները սովորաբար փոխանցվում են հղումով, այսինքն Byref բանալիային բառով (եթե ակնհայտորեն չի նշված հակառակը):

2.23. Հավելվածի տեղեկատու համակարգի ստեղծումը

Տեղեկատու համակարգի ստեղծումը ընդհանուր տեսքով ընդգրկում է հետևյալ փուլերը.

1. տեղեկատու համակարգի սկզբնական ֆայլերի ստեղծում, որը ընդգրկում է:
 - HTML ֆայլը՝ նկարագրելով առանձին թեմաները
 - պրոյեկտի ֆայլը
 - բովանդակության ֆայլը
 - ինդեքսային ֆայլը, որը օգտագործվում է ինֆորմացիայի արագ որոշման համար
 2. պրոյեկտի ֆայլի կոճակիյացիայի օպերացիան, որի կատարումից հետո տեղեկատու համակարգի բոլոր ֆայլերը կոճակիյացվում են նեկ ֆայլի մեջ
 3. տեղեկատու համակարգի տեստավորում և կարգավորում
 4. հավելվածի ինտերֆեյսի էլեմենտներին տեղեկագրի բաժինների նշանակում:
- Նախագծի ստեղծումը իրականացվում է հետևյալ քայլերով
- a) ստեղծում ենք տեղեկատու համակարգի թեմաները HTML ֆորմատով առանձին-առանձին,
 - b) մեկնարկում ենք HTML Help Workshop ծրագիրը և օգտվում վարպետից:

2.24. Հավելվածի կարգավորման պարամետրերը

Հավելվածների ստեղծման ժամանակ հաճախ հարց է առաջանում, որտեղ պահպանել հավելվածի կարգավորման պարամետրերը կամ որտեղ տեղադրել վերջին անգամ բացված ցուցակը: Windows-ի վերջին տարբերակներում այդ պարամետրերը տեղադրվում են Windows-ի ռեստրում (մատյանում): Visual Basic-ը առաջարկում է ռեստրի հետ աշխատելու չորս հրամաններ:

Savesetting ֆունկցիայի օգնությամբ գրվածքները պահպանվում են ռեստրում:

Savesetting(հավելվածի_անունը, բաժին, բանալի, տեղադրում):
Օրինակ Savesetting "bigmoney", "User", "Name", "Hugo"

Օրինակ ստեղծված է Bigmoney անունով ռեստրի նոր ճյուղ:

Նրա տակ գտնվում է User ծյուղը, իսկ նրա մեջ Name գրվածքը՝ Hugo արժեքով:

Պարամետրերի կարդալը Տեղադրումները կարդալու համար օգտագործվում է Getsetting ֆունկցիան: Getsetting (հավելվածի _անուն, բաժին, բանալի[], լույսայն]): Լույսայն պարամետրը օգնում է ինանալ, թե հաջո՞ղ է կատարվել արդյոք գրվածքների կարդալը: Եթե նշված տիրույթի կարդալը հնարավոր չէ, ֆունկցիան վերադարձնում է արժեք, որը վերագրված է լույսայն պարամետրին: Եթե պրոցեսը ավարտվել է հաջող, ապա որպես վերադարձվող արժեք ստանում ենք ռեեստրում պահպանված գրվածքը:

```
Vreturn=Getsetting("BigMoney", "User", "Name", "Failure");
```

Օրինակ, եթե կարդալը չի կատարվում այն պատճառով, որ գրվածքը գոյություն չունի, ապա որպես վերադարձվող արժեք ստանում ենք "failure", այսինքն լույսայն պարամետրը:

Getsettings ֆունկցիայի օգնությամբ կարդացվում է բաժնից բոլոր գրվածքները: Այդ ժամանակ վերադարձվում է զանգված Variant տիպի բաժնում նշված բոլոր էլեմենտներով:

Պարամետրերի հեռացումը. Դավելվածի դեխնստայացիայի (հեռացման) ժամանակ պետք է հատուկ հոգ տանել ռեեստրում ոչ անհրաժեշտ տեղադրումների հեռացման մասին, որը կանխում է ինֆորմացիոն «աղբի» կուտակումը: Դրա համար Visual Basic-ն ունի DeleteSetting օպերատոր, որը հեռացնում է գրվածքը կամ ամբողջ բաժինը:

```
DeleteSetting (հավելվածի _անունը, բաժինը, [], բանալի])
```

DeleteSetting " BigMoney", "User" շահագործողի մասին ամբողջ ինֆորմացիան հեռացվում է:

Ունեստրի ուրիշ տիրույթների հետ աշխատանքի համար պետք է օգտագործել Windows API -ի ֆունկցիաները:

Գործնական և լաբորատոր աշխատանքներ

Լաբորատոր աշխատանքներ

Լաբորատոր աշխատանք N 1

Աշխատանք գժային պրոցեսների հետ: Ղեկավարման էլեմենտների հատկությունների փոփոխութ նախագծման ռեժիմութ և ծրագրային ռեժիմում:

1. Ֆորմայի վրա տեղադրել երկու գրություն (label) և երկու տեքստային դաշտ: Առաջին գրության օգնությամբ գրել “Անուն”, երկրորդի օգնությամբ “Ազգանուն”: Առաջին տեքստային դաշտում մուտքագրել օգտվողի անունը, երկրորդում ազգանունը: Ծրագրավորել հրամանային սեղմակն այնպես, որ փոփոխ տեքստային դաշտերի տեքստերը տեղերը և գրությունների պարունակությունները:

```
Private Sub Command1_Click()
Dim x As String
Dim y As String
x = Label1.Caption
Label1.Caption = Label2.Caption
Label2.Caption = x
y = Text1.Text
Text1.Text = Text2.Text
Text2.Text = y
Form1.BackColor = vbRed
End Sub
Private Sub Form_Load()
Label1.Caption = "name"
Label2.Caption = "lastname"
Text1.Text = ""
Text2.Text = ""
Command1.Caption = "changing places"
Form1.WindowState = 2
End Sub
```

2. Մուտքագրել եռանկյան գագաթի կոորդինատները և հաշվել բարձրությունները:

Լաբորատոր աշխատանք N 2

Նպատակը: Աշխատանք ճյուղավորվող պրոցեսների հետ
1. Որոշել տրված երեք թվերից մեծագույնը և փոքրագույնը:

```
Private Sub
Command1_Click()
a = CDbl(Text1.Text)
b = CDbl(Text2.Text)
c = CDbl(Text3.Text)
Dim lmin As Double
Dim tmax As Double
If a > b Then
lmin = b
tmax = a
Else
tmax = b
lmin = a
End If
If c > tmax Then
tmax = c
End If
If c < lmin Then
lmin = c
End If
Text4.Text = tmax
Text5.Text = lmin
End Sub
```

```
Private Sub
Command2_Click()
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
End Sub

Private Sub Form_Load()
Label1.Caption = "a = "
Label2.Caption = "b = "
Label3.Caption = "c = "
Label4.Caption = "max = "
Label5.Caption = "min = "
Command1.Caption = "find
min max"
Command2.Caption = "clear"
Form1.WindowState = 2
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
End Sub
```

2. Տրված են եռանկյան գագաթի կոորդինատները: Որոշել, թե տրված կետը ընկած է եռանկյան ներսում, թե՝ ոչ:
3. Կիսամյակի ստացած գնահատականներով որոշել վճարվող կրթաթոշակի չափը:

Հաբորատոր աշխատանք N 3

Նպատակը: Աշխատանք պրոցեդուրաների և ֆունկցիաների, մեթոդների և հատկությունների հետ

- Խմբի առաջադիմությունը տոկոսային արտահայտությամբ ինչ-որ առարկայից կազմել է անբավարար 5, բավարար 17, լավ 36, գերազանց 42 տոկոս: Խմբի առաջադիմությունը հավելվածի գրաֆիկական պատուհանում արտահայտել շրջանագծային դիագրամի տեսքով օգտագործելով Fillstyle հատկությունը:

Dim a As Double: Const pi = 3.14159265358979

```
Private Sub Command1_Click()
Picture1.Scale (-100, -100)-(100, 100)
a = 2 * pi / 100
Picture1.FillStyle = 2
Picture1.Circle (0, 0), 60, vbGreen, -100 * a, -5 * a
Picture1.FillStyle = 3
Picture1.Circle (0, 0), 60, vbRed, -5 * a, -22 * a
Picture1.FillStyle = 4
Picture1.Circle (0, 0), 60, vbBlue, -22 * a, -58 * a
Picture1.FillStyle = 5
Picture1.Circle (0, 0), 60, vbYellow, -58 * a, -100 * a
End Sub
```

```
Private Sub Command2_Click()
End
End Sub
```

```
Private Sub Form_Load()
Form1.WindowState = 2
End Sub
```

2. Կազմել ծրագիր, որը որոշի երկու թվերի ամենափոքր ընդհանուր բազմապատիկը:

```
Function mbj(n As Integer, m  
As Integer)  
Dim p As Integer  
p = n Mod m  
Do While p <> 0  
n = m  
m = p  
p = n Mod m  
Loop  
mbj = m  
End Function
```

```
Function pbk (n As Integer, m  
As Integer)  
pbk = n * m / mbj (n, m)  
End Function
```

```
Private Sub  
Command1_Click()  
Dim nm As Integer  
m = CInt(Text1.Text)  
n = CInt(Text2.Text)  
nm = pbk(n, m)  
Text3.Text = CStr(nm)  
End Sub
```

```
Private Sub Form_Load()  
Form1.WindowState = 2  
Label1.Caption = "n = "  
Label2.Caption = "m = "  
Label3.Caption = "result"  
Text1.Text = Empty  
Text2.Text = Empty  
Text3.Text = Empty  
Command1.Caption = "count"  
End Sub
```

3. Կազմել պարզագույն հաշվիչի ծրագիր:

4. Կազմել ծրագիր, որը հաշվի $C_n^m = n! / m!(n-m)!$ –ը:

Հարորատոր աշխատանք N 4

Նպատակը՝ Աշխատանք ցիկլային պրոցեսների հետ

- Օգտվելով Pset մեթոդից հավելվածի գրաֆիկական պատուհանում տարբեր գույներով նույն կոորդինատային առանցքի վրա կառուցել $Y=ax^2+bx+c$, $Y=k\sin(x)$, $Y=\sin(k\cdot x)$ և $Y=k\cdot x$ ֆունկցիաների գրաֆիկները նախապես մուտքագրելով a, b, c, k գործակիցները:

```
Private Sub Command1_Click()
```

```
Dim a As Double, b As Double, c As Double, k As Double, y1 As  
Double, y2 As Double, y3 As Double, y4 As Double
```

```
a = InputBox(a)
```

```
b = InputBox(b)
```

```
c = InputBox(c)
```

```
k = InputBox(k)
```

```
Scale (-10, 10)-(10, -10)
```

```
Line (-10, 0)-(10, 0)
```

```
Line (0, -10)-(0, 10)
```

```
DrawWidth = 7
```

```
For x = -10 To 10 Step 0.001
```

```
y1 = a * x ^ 2 + b * x + c
```

```
PSet (x, y1), vbRed
```

```
y2 = k * Sin(x)
```

```
PSet (x, y2), vbGreen
```

```
y3 = Sin(k * x)
```

```
PSet (x, y3), vbBlue
```

```
y4 = k * x
```

```
PSet (x, y4), vbYellow
```

```
Next x
```

```
End Sub
```

```
Private Sub Form_Load()
```

```
Form1.WindowState = 2
```

```
End Sub
```

2. Որոշել $a_1, a_2, a_3, \dots, a_n$ միաչափ զանգվածի մեջագույն տարրը:

```
Dim a() As Integer, n As Integer
Private Sub Command1_Click()
n = InputBox("input n", n)
ReDim a(n)
For i = 1 To n
a(i) = InputBox("input the elements of the vector", a(i))
Next i
Max = a(1)
For i = 1 To n
If a(i) > Max Then
Max = a(i)
End If
Next i
For i = 1 To n
Print a(i)
Next i
Print
Print "n=", n
Print
Print "max=", Max
End Sub
Private Sub Command2_Click()
End
End Sub
Private Sub Form_Load()
Form1.WindowState = 2
End Sub
```

3. Հաշվել $A(i,j)$ մատրիցայի դրական անդամների միջին թվաքանականը:

4. Մուտքագրել ո չափանի զանգված: Ընտրել և դուրս բերել միայն կենտ թվերը՝ օգտագործելով InputBox ֆունկցիան: Զանգվածը արտացոլել ListBox –ում:

Հարորատոր աշխատանք N 5

Նպատակը: Աշխատանք խառը պրոցեսների հետ

- Մուտքագրել խմբի ուսանողների քանակը, անուն ազգանունները և “Ծրագրավորման տեխնոլոգիաներ” առարկայից ստացած գնահատականները: List1-ում արտացոլել դրական ստացողների ցուցակը, List2-ում անբավարար ստացողների ցուցակը:

```
Dim aah() As String  
Dim gnh() As Integer  
  
Private Sub Command1_Click()  
dim usqan as integer  
usqan = InputBox("մտցրեք ուսանողների քանակը",usqan)  
ReDim aah(usqan)  
ReDim gnh(usqan)  
For i = 0 To usqan - 1  
aah(i) = InputBox("aah(" & i & ")=",aah(i))  
gnh(i) = InputBox("gnh(" & i & ")=",gnh(i))  
If gnh(i) > 2 Then  
List1.AddItem aah(i)  
Else  
List2.AddItem aah(i)  
End If  
Next i  
End Sub  
  
Private Sub Command2_Click()  
End  
End Sub  
  
Private Sub Form_Load()  
Form1.WindowState = 2  
Label1.Caption = "դրական ստացողներ"  
Label2.Caption = "անբավարարներ"  
Command1.Caption = "մեկնարկել"  
Command2.Caption = "դադարեցնել"  
End Sub
```

2. Կազմել ծրագիր, որը էկրանի տարրեր անկյուններից գծի պատահական գույներով, պատահական երկարությամբ հատվածներ և դադարեցնի աշխատանքը ստեղնաշարի ստեղնը սեղմելու դեպքում:

```
Dim x1 as Boolean  
Private sub Form_Initialize()  
Dim x as double, y as double ,xyyx as integer  
WindowState=vbMaximized  
Scale(0,0)-(1,1)  
X1=False  
Backcolor=vbBlack  
Randomize()  
Do Until x1  
X=Rnd  
Y=Rnd  
Xyyx=int(15*Rnd)  
Line(0,0)-(x,y),QBColor(xyyx)  
Line(1,0)-(x,y),QBColor(xyyx)  
Line(0,1)-(x,y),QBColor(xyyx)  
Line(1,1)-(x,y),QBColor(xyyx)  
DoEvents  
Loop  
End sub  
Private sub Form_KeyDown(KeyCode As integer, shift As integer)  
X1=true  
End  
End sub  
Private sub Form_Load()  
Show  
End sub
```

3. A(i,j) մատրիցայի գլխավոր անկյունագծի տարրերը դասվորել նվազման կարգով:

4. Կազմել ծրագիր, որը հաշվի արտադրամասի աշխատողների աշխատավարձը հետևյալ պայմանների դեպքում. 3% կենսաթոշակային ֆոնդին կատարվող վճարումներ և եկամտա-հարկի հաշվարկ հետևյալ տոկոսադրույթներով մինչև 20000 դրամ աշխատավարձի ժամանակ 0% պահում, 20001-80000-ի դեպքում (աշխատավարձ-20000)*0.1, իսկ 80000-ից բարձրի դեպքում 8000+ (աշխատավարձ-80000)*0.2:

Գործնական և լաբորատոր աշխատանքների համար առաջարկվող խնդիրներ

1. Կազմել բլոկ սխեման և ծրագիրը (կոդը), որը հաշվի ֆունկցիայի արժեքը.

$$y = \begin{cases} |X|, \text{եթե } X < -2 \\ \log_2 X, \text{եթե } X > 3 \\ \sqrt{\sin^2 X + e^X}, \text{եթե } |X| \leq 1 \\ 1, \text{եթե } X = 2 \\ 0 \text{ մնացած դեպքերում} \end{cases}$$

2. Կազմել բլոկ սխեման և ծրագիրը (կոդը), որը հաշվի ֆունկցիայի արժեքը.

$$y = \begin{cases} e^x, \text{եթե } X < 0 \\ \sqrt{\sin X}, \text{եթե } X \in [1; 2] \\ \log_2 |X|, \text{մնացած դեպքերում} \end{cases}$$

3. Կազմել բլոկ սխեման և ծրագիրը (կոդը), որը հաշվի ֆունկցիայի արժեքը.

$$y = \begin{cases} 6\sin(X(aX + b)), \text{եթե } X < 0 \\ 7\cos(X - b), \text{եթե } 0 \leq X \leq 1 \\ e^{\sqrt{1 + X^2}}, \text{եթե } X > 1 \\ a = \text{const}, b = \text{const} \end{cases}$$

4. Կազմել բլոկ սխեման և ծրագիրը (կոդը), որը հաշվի ֆունկցիայի արժեքը.

$$y = \begin{cases} \frac{1}{X + 1} + 3X^5, \text{եթե } X < 0 \\ 5X + 7 + \frac{3}{X - 2}, \text{եթե } X \geq 0 \end{cases}$$

5. Կազմել բլոկ սխեման և ծրագիրը (կոդը), որը հաշվի ֆունկցիայի արժեքը.

$$y = \begin{cases} \ln(X^4 + c^2), \text{եթե } X < -1 \\ e^{\sqrt{1 + \ln^2(X^2 + 1)}}, \text{եթե } X > 1 \\ c * X^2 + 1, \text{եթե } |X| \leq 1 \\ c = \text{const} \end{cases}$$

6. Կազմել բլոկ սխեման և ծրագիրը (կոդը), որը հաշվի ֆունկցիայի արժեքը.

$$y = \begin{cases} \sqrt{x^2 + a^2}, & \text{եթե } 1 \leq x \leq 2.5 \\ \operatorname{tg}x, & \text{եթե } 2.5 \leq x \leq 4 \\ \sin \cos x, & \text{եթե } 6 \leq x \leq 8 \\ 0 & \text{մնացած դեպքերում} \end{cases}$$

7. Կազմել բլոկ սխեման և ծրագիրը (կոդը), որը հաշվի ֆունկցիայի արժեքը.

$$S = \prod_{i=1}^{200} \frac{i^2}{i^2 + 10}$$

8. Կազմել բլոկ սխեման և ծրագիրը (կոդը), որը հաշվի ֆունկցիայի արժեքը.

$$S = \sum_{n=1}^m 2^n$$

9. Կազմել բլոկ սխեման և ծրագիրը (կոդը), որը հաշվի ֆունկցիայի արժեքը.

$$y = \begin{cases} \sin x^2, & \text{եթե } |x| \leq 2 \\ \cos^2 x & \text{մնացած դեպքերում} \end{cases}$$

10. Կազմել բլոկ սխեման և ծրագիրը (կոդը), որը հաշվի ֆունկցիայի արժեքը.

$$y = \sum_{k=1}^n (-1)^k \frac{k^2 + 2}{5}$$

11. Կազմել բլոկ սխեման և ծրագիրը (կոդը), որը հաշվի միջին թվաբանականը.

$$\bar{X} = \frac{\sum_{i=1}^N X(i)}{N}$$

12. Կազմել բլոկ սխեման և ծրագիրը (կոդը), որը հաշվի միջին երկրաչափականը.

$$\bar{Y} = \sqrt[n]{\prod_{I=1}^N X(I)}$$

13. Կազմել բլոկ սխեման և ծրագիրը (կոդը), որը հաշվի միջին հարմոնիկը.

$$\bar{Z} = \frac{N}{\sum_{I=1}^N \frac{1}{X(I)}}$$

14. Կազմել բլոկ սխեման և ծրագիրը (կողը), որը հաշվի կշռված միջին թվաբանականը.

$$\bar{S} = \frac{\sum_{I=1}^N X(I)M(I)}{\sum_{I=1}^N M(I)}$$

15. Կազմել բլոկ սխեման և ծրագիրը (կողը), որը հաշվի կշռված միջին հարմոնիկը.

$$\bar{P} = \frac{\sum_{I=1}^N M(I)}{\sum_{I=1}^N \frac{M(I)}{X(I)}}$$

16. Կազմել բլոկ սխեման և ծրագիրը (կողը), որը հաշվի դիսպերսիան.

$$S^2 = \frac{\sum_{I=1}^N (X(I) - \bar{X})^2 * M(I)}{\sum_{I=1}^N M(I)}$$

17. Կազմել բլոկ սխեման և ծրագիրը (կողը), որը հաշվի միջին գծային շեղումը.

$$\bar{X} = \frac{\sum_{I=1}^N |X(I) - \bar{X}| * M(I)}{\sum_{I=1}^N M(I)}, \text{ որտեղ } \bar{X} = \frac{\sum_{I=1}^N X(I) * M(I)}{\sum_{I=1}^N M(I)}$$

18. Կազմել բլոկ սխեման և ծրագիրը (կողը), որը հաշվի կշռված միջին քառակուսային շեղումը.

$$S = \sqrt{\frac{\sum_{I=1}^N (X(I) - \bar{X})^2 * M(I)}{\sum_{I=1}^N M(I)}}$$

19. Գտնել $a_1, a_2, a_3, \dots, a_n$ զանգվածի մեջիանան, եթե $N=2k+1$ մեջիանան կլինի.

$$a\left(\frac{N}{2}\right) \text{ թ, եթե } N = 2k + 1,$$

$$\text{ապա } a\left(\frac{N}{2}\right) + a\left(\frac{N}{2} + 1\right)$$

20. Կազմել բլոկ սխեման և ծրագիրը, որը տրված $X(x_1, x_2, x_3, \dots, x_n)$ վեկտորի տարրերից որոշի մեծագույն տարրը և ֆիքսի նրա համարը:
21. Կազմել բլոկ սխեման և ծրագիրը, որը տրված $X(x_1, x_2, x_3, \dots, x_n)$ վեկտորի տարրերից որոշի զույգ ինդեքս ունեցող բացասական տարրերի միջին թվաբանականը:
22. Կազմել բլոկ սխեման և ծրագիրը, որը տրված $X(x_1, x_2, x_3, \dots, x_n)$ վեկտորի տարրերից որոշի կենտ ինդեքս ունեցող բացասական տարրերից մեծագույնը:
23. Կազմել բլոկ սխեման և ծրագիրը, որը տրված $X(x_1, x_2, x_3, \dots, x_n)$ վեկտորի տարրերից որոշի զույգ ինդեքս ունեցող դրական տարրերից փոքրագույնը:
24. Կազմել բլոկ սխեման և ծրագիրը, որը տրված $X(x_1, x_2, x_3, \dots, x_n)$ վեկտորի տարրերից որոշի 0-ների քանակը:
25. Կազմել բլոկ սխեման և ծրագիրը, որը տրված $X(x_1, x_2, x_3, \dots, x_n)$ վեկտորի բացասական տարրերը փոխարինի իրենց մոդուլներով:
26. Կազմել բլոկ սխեման և ծրագիրը, որը հաշվի տրված $X(x_1, x_2, x_3, \dots, x_n)$ վեկտորի այն տարրերի քանակը և գումարը, որոնք պատկանում են [a;b] միջակայքին:
27. Կազմել բլոկ սխեման և ծրագիրը, որը տրված $X(x_1, x_2, x_3, \dots, x_n)$ վեկտորի բացասական տարրերը փոխարինի առաջին հանդիպող դրական թվով:
28. Կազմել բլոկ սխեման և ծրագիրը, որը տրված $X(x_1, x_2, x_3, \dots, x_n)$ վեկտորի տարրերը դասավորի նվազման կարգով:
29. Կազմել բլոկ սխեման և ծրագիրը, որը տրված $X(x_1, x_2, x_3, \dots, x_n)$ վեկտորի տարրերից որոշի, թե որ տոկոսն է կազմում 0-ական տարրերի քանակը ամբողջ զանգվածում:
30. Կազմել բլոկ սխեման և ծրագիրը, որը տրված $X(x_1, x_2, x_3, \dots, x_n)$ վեկտորի տարրերից զույգ ինդեքս ունեցող դրական թվերը դասավորի աճման կարգով:
31. Կազմել բլոկ սխեման և ծրագիրը, որը տրված $X(x_1, x_2, x_3, \dots, x_n)$ վեկտորի տարրերից որոշի փոքրագույն և մեծագույն տարրերի միջին թվաբանականը:
32. Տրված են X , Y , Z իրական թվերը: Եթե $X \leq Y \leq Z$, ապա յուրաքանչյուր թիվը փոխարինի դրանցից ամենամեծով, եթե $X > Y > Z$, ապա թվերը չփոխել, հակառակ դեպքում բոլոր թվերը փոխարինել իրենց քառակուսիներով:

33. Կազմել բլոկ սխեման և ծրագիրը, որը որոշի $A(a_1, a_2, a_3, \dots, a_n)$ և $B(b_1, b_2, b_3, \dots, b_n)$ վեկտորների սկալյար արտադրյալը:
34. Կազմել բլոկ սխեման և ծրագիրը, որը հաշվի 7-ի վրա բաժանվող երկնիշ թվերի միջին թվաբանականը:
35. Կազմել բլոկ սխեման և ծրագիրը, որը հաշվի 5-ի վրա բաժանվող երկնիշ թվերի քանակը և գումարը:
36. Տրված է $A=II$ $a_i II, i = 1..n ; j = 1..m$ մատրիցան. կարգավորել մատրիցի սյուները ըստ առաջին տողի տարրերի նվազման:
37. Տրված է $A=II$ $a_i II, i = 1..n ; j = 1..m$ մատրիցան. տեղափոխել P և T -ող սյուների տեղերը:
38. Տրված է $A=II$ $a_i II, i = 1..n ; j = 1..m$ մատրիցան. կազմել վեկտոր, որի բաղադրիչները հանդիսանում են մատրիցի համապատասխան տողերի մեջագույն տարրերը:
39. Տրված է $A=II$ $a_i II, i = 1..n ; j = 1..m$ մատրիցան. կազմել վեկտոր, որի բաղադրիչները հանդիսանում են մատրիցի համապատասխան սյուների փոքրագույն տարրերի հերթական համարները սյուների մեջ:
40. Տրված է $A=II$ $a_i II, i = 1..n ; j = 1..m$ մատրիցան. որոշել մատրիցի բոլոր տողերի մեծագույն տարրերը և նրանց մեջ փոքրագույնը:
41. Տրված է $A=II$ $a_i II, i = 1..n ; j = 1..m$ մատրիցան. հաշվել մատրիցի դրական, բացասական և 0-ական տարրերի քանակների տոկոսային հարաբերությունները ընդհանուր տարրերի քանակի նկատմամբ:
42. Տրված է $A=II$ $a_i II, i = 1..n ; j = 1..m$ մատրիցան. կարգավորել մատրիցի տողերը ըստ առաջին սյան տարրերի աճման:
43. Տրված է $A=II$ $a_i II, i = 1..n ; j = 1..m$ մատրիցան. գտնել մատրիցի գլխավոր անկյունագծից վերև գտնվող տարրերից փոքրագույնը:
44. Տրված է $A=II$ $a_i II, i = 1..n ; j = 1..m$ մատրիցան. գտնել մատրիցի գլխավոր անկյունագծից ներքև գտնվող տարրերից մեծագույնը:
45. Կազմել ծրագիր, որը հաշվի արտադրամասի աշխատողների աշխատավարձը հետևյալ պայմանների դեպքում: 3% կենսաբոշակային ֆոնդին կատարվող վճարումների և եկամտահարկի հաշվարկման հետևյալ տոկոսադրույթների դեպքում: Մինչև 20000 դրամ՝ եկամտահարկ 0%; 20001-

- 80000-ի դեպքում՝ (աշխատավարձ-20000) - ի 10%,
 800001-ից բարձրի դեպքում՝ (աշխատավարձ -
 80000)*20%+8000:
46. Կազմել ծրագիր, որը հաշվարկի անաշխատունակության համար վճարվող գումարը հետևյալ պայմանների դեպքում՝ Z-ը՝ միջին աշխատավարձը նախորդ երկու ամիսների համար, K-ն՝ երկու ամիսների փաստացի աշխատանքային օրերի քանակը, C-ն՝ անընդմեջ աշխատանքային ստաժը, b-ն՝ հիվանդության օրերի թիվը:
 47. Մարենատիկայից անբավարար ստացած ուսանողների ցուցակը (անունները) ելքագրել հայելային արտապատկերման (շրջված) տեսքով:
 48. Ֆորմայի վրա ժամացույցի տեսքով կառուցել մուտքի իրավունքի ծրագիր, որը 15 վայրկյանի ընթացքում պահանջի գաղտնաբառ. Եթե գաղտնաբառը ծիչտ է, հրավիրի համակարգ, իսկ եթե ոչ, տա հաղորդագրություն, որ չի ճանաչում:
 49. Ներկայացնել մարենատիկայից խմբում 5, 4, 3 և 2 ստացողների քանակները դիմումի տեսքով:
 50. Կազմել ծրագիր, որը հաշվի աշխատավարձի վճարումը օգտագործելով ամենաքիչ թվով քանկյանուներ, որոնք բաղկացած լինեն (50000, 20000, 5000, 1000, 500, 200, 100, 50, 25, 10) հայկական թղթադրամներից:
 51. Կառուցել ֆորմա, որը մուտքագրի և արտացոլի խմբի ուսանողների անունները, հասցեները և կիսամյակի քննություններից ստացած գնահատականները:
 52. Կառուցել ֆորմա, որը մուտքագրի երկու մատրիցաներ և արտացոլի նրանց արտադրյալը:
 53. Կառուցել ֆորմա, որը մուտքագրի մատրիցա և արտացոլի տրանսպոնացված մատրիցան:
 54. Կառուցել ֆորմա, որը մուտքագրի քառակուսային մատրիցա և հաշվի որոշիչը Գառւսի մեթոդով:
 55. Կառուցել ֆորմա, որը կետի դեկարտյան կոորդինատները ձևափոխի բևեռային կոորդինատների և հակառակը:
 56. Քառակուսային ինտերպուլացիա - էքստրապուլացիայի մեթոդով մեկ փոփոխականից կախված ֆունկցիայի էքստրեմումի հաշվարկը:
 57. Զույգ կոռելյացիայի գործակիցների հաշվարկը:

58. ա) Փոքրագույն քառակուսիների մեթոդով գծային ֆունկցիայի որոշումը:
բ) Փոքրագույն քառակուսիների մեթոդով քազմանդամի ապրոկսիմացիան:
59. Սահմանային տոկոսադրույթի որոշումը:
Խնդրի լուծման ժամանակ հարկավոր է մուտքագրել հետևյալ բնութագրերի արժեքները վճարումների թիվը, դրամական վարկի չափը, մեկ վճարման չափը, տոկոսադրույթը: Մարգինալ տոկոսաչափը որոշվում է հետևյալ հավասարման արմատից ավանդի ընթացիկ ծավալը = դրամական վարկի չափին:
60. Պարբերական վճարումների դիագրամային վերլուծությունը:
Form-ի վրա տեղադրել հետևյալ դեկավարման էլեմենտները նշված հաջորդականությամբ. ա) Textbox (4 հատ) բ) commandbutton գ) label դ) Mschart: Text1 – Text4 դաշտերում գրանցեք համապատասխան վճարումների թիվը, վարկի չափը, մեկ վճարման չափը, տոկոսաչափը և OK սեղմակը սեղմելով կառուցի դիագրամը:
61. Գրապահարանում կան երկու ինֆորմատիկայի, երեք ծրագրավորման և չորս մաթեմատիկայի գրքեր: Բանի տարբեր եղանակներով կարող ենք 9 գրքերը դասավորել գրապահարանում, եթե ինֆորմատիկայի գրքերը դրված են միասին, ծրագրավորմանը՝ միասին, մաթեմատիկայինը՝ միասին:
62. Կազմել կետ-խաչի (x-երի և 0-ների) խաղի կողը, երբ ո=5-ի հաշվի առնելով նաև անկյունազները:
63. Շախմատային տախտակը ներկայացնել 8X8 չափանի սիմվոլային մատրիցի տեսքով: Տրված են բնական թ և թվեր ($1 \leq p \leq 8$; $1 \leq q \leq 8$), որոնց հատման կետը որոշում է թագուհու տեղը: Մատրիցի համապատասխան էլեմենտը նշել F տառով: Այն դաշտերը, որոնք գտնվում են թագուհու հարկածի տակ, դնել *, իսկ մնացած դաշտերը Օ սիմվոլ: Մատրիցի տողերը ելքագրել մեկը մյուսի տակ: Լուծել նմանատիպ խնդիր ծիու համար:
64. Կառուցել 10 ներդրված քառակուսիներ հերթականությամբ ներկայացնելով կարմիր և կանաչ գույներով:

65. Ստանալ էկրանի կենտրոնում պատկերներ, որը բաղկացած լինի 10 ներդրված քառակուսիներից՝ 10, 20, 30, ...100 երկարությամբ կողմերով:
66. Էկրանի վրա ելքագրել երկու ուղղանկյուններ: Մեկը շրիխավորել ուղղահայաց ուղղությունով, մյուսը՝ հորիզոնական ուղիղներով:
67. Կառուցել ուղղանկյուն՝ 30 և 50 կողմերով, որի կենտրոնը համընկնում է էկրանի կենտրոնի հետ: Ուղղանկյան կողմերը պետք է գուգահեռ լինեն էկրանի կողմերին:
68. Տրված են 6 ամբողջ թվեր, որոնք որոշում են եռանկյան գագաթների դիրքը, տեղադրված էկրանի ծախս կողմուն (կեսում): Կառուցել էկրանի վրա այդ եռանկյունը, ինչպես նաև եռանկյուն, որը սիմետրիկ է այդ եռանկյանը էկրանի կենտրոնով անցնող ուղղահայաց ուղղի նկատմամբ:
69. 4 ամբողջ թվեր տալիս են էկրանի վրա հատվածի ծայրակետերի դիրքը: Ստանալ այդ հատվածի պատկերը և էկրանի կենտրոնի նկատմամբ նրա սիմետրիկ պատկերը:
70. Ստանալ էկրանի կենտրոնում պատկեր, որը բաղկացած լինի 9 ներդրված քառակուսիներից և ներկել դրանք 3 գույներով:
71. Կառուցել 9 խոտացված (կոնցենտրացված) շրջաններ՝ ներկված հերթականությամբ՝ կանաչ, կարմիր և կապույտ գույներով:
72. Գրել ծրագիր, որի կատարման ընթացքում կանաչ գույնի շրջանը, որը հայտնվում է էկրանի կենտրոնում, աստիճանաբար ընդլայնվում է՝ մեծանալով 3 անգամ, այնուհետև սեղմվում մինչև սկզբնական չափերը:
73. Կազմել կոդ, որը որոշի թե քանի հնարավոր եղանակով կարելի է խմբի տղաներից կազմել ֆուտբոլի թիմեր:
74. Կազմել կոդ, որը որոշի թե քանի հնարավոր եղանակով կարելի է խմբի աղջիկներից կազմել բասկետբոլի թիմեր:
75. Էկրանի վերևի անկյուններից դեպի կենտրոն են սլանում 2 ավտոմեքենաներ, իսկ ներքևի աջ անկյունից՝ ավտոտեսչության աշխատակիցը: Էկրանի կենտրոնում մեքենաները բախվում են իրար: Վերևի աջ անկյունից՝ շարժվող մեքենան փախուստի է դիմում դեպի ներքևի ծախս անկյունը: Ավտոտեսչության աշխատակիցը հետապնդում է նրան և կանգնեցնում է ներքևի ծախս անկյունում: Կազմել կոդ, որը ցուցադրի վերը շարադրվածը:

76. Էկրանի անկյուններում տեղադրված են պայմանական ռումբեր: Օբյեկտը գտնվում է էկրանի կենտրոնում: Կազմել կող, որ օբյեկտը շարժի խնբագրման ստեղներով և հանդիպելով ռումբերին առաջացնի ձայնային ազդանշան, իսկ հանդիպման կետում մեծացող և նախկին չափերին վերադարձող կարմիր շրջան:
77. Էկրանի վրա պատահական հայտնվում են 3 ճանճեր և շարժվում ամենահեռու գտնվող անկյան ուղղությամբ: Եթե չեք հարվածում ճանճին, ապա նրա չափերը մեծանում են և ուղեկցվում ձայնային ուժեղացող ազդանշանով: Կազմել ծրագիր, որ յուրաքանչյուր ճանճին հարվածելիս ուղեկցվի ձայնային և գրաֆիկական էֆեկտով և գումարի խաղացողի հաշվին միավոր: Ծրագիրը դադարեցվի որոշակի միավոր հավաքելու դեպքում և ելքագրի ամփոփ տվյալներ:
78. Կազմել կող, որը մնխիկի կրօնակի սեղմումով կատարի նշիշի տեսքի փոփոխում և այդ կետի կոորդինատի ֆիքսում Բաց քողնելու ժամանակ կառուցի շրջանագիծ, որի կենտրոնը հանդիսանում է ֆիքսված կոորդինատը:
79. Pmt ֆունկցիայի կիրառմամբ հաշվել 30 տարով, տարեկան 8% տոկոսադրույթով ներդրված 100000 դրամ գումարի ամսական վճարումները:
80. IPMT ֆունկցիայի կիրառմամբ հաշվել 30 տարով, տարեկան 8% տոկոսադրույթով 500000 դրամ ներդրված գումարի վճարման մասը երրորդ տարվա առաջին կեսին:
81. FV ֆունկցիայի կիրառմամբ հաշվել 12 ամսվա ընթացքում ամսական 600 դրամ վճարումով, առանց ընթացիկ մուտքերի տարեկան 12% հաշվարկաչափով կուտակումները:
82. NPER ֆունկցիայի կիրառմամբ հաշվել հաշվում եղած 100000 դրամ գումարի 5% տոկոսադրույթով ամսական 1000 դրամ ծախսով ներդրված գումարի սպառման ժամանակահատվածը:

ԳԼՈՒԽ 3

C++ ը որպես ծրագրավորման գործիքային միջոց

3.1. C լեզվի բազային հասկացությունները

Ծրագրավորման նոր ալգորիթմական լեզու սովորելու համար անհրաժեշտ է պարզել հետևյալ հարցերը:

1. Ինչ այբուբեն ունի լեզուն և ինչպես ճիշտ գրել նրա լեզվաները (լեկսեմ–ծրագրի տեքստի միավորը, որը կոմպիլյացիայի ժամանակ ընդունվում է որպես միասնական ամբողջ և ինաստով չի կարող տրոհվել ավելի փոքր էլեմենտների):
2. Լեզվում տվյալների ինչ տիպեր են ընդունվում և ինչպես են դրանք որոշվում (նկարագրվում):
3. Լեզվում ինչպիսի օպերացիաներ են թույլատրելի տվյալների վրա, ինչպես է դրանց օգնությամբ կառուցվում արտահայտություն և ինչպես են դրանք կատարվում:
4. Ինչպիսի կառուցվածք ունեն ծրագրերը և ինչպիսի հաջորդականությամբ են տեղադրվում օպերատորները, նկարագրությունը և հայտարարումը:
5. Ինչպես ելքագրել (ներկայացնել) շահագործողին ծրագրի աշխատանքի արդյունքները:
6. Ինչպես են իրականացված վերագրման, պայմանական և անցնան օպերատորները:
7. Ինչպես մտցնել սկզբնական տվյալները ծրագրի համար:
8. Լեզվում ինչպիսի հատուկ կոնստրուկցիաներ կան ցիկլերի կազմակերպման համար:
9. Ինչպիսի ենթածրագրերի (պրոցեդուրա) և ենթագրի-ֆունկցիաների ապարատ գոյություն ունի:

Հեղակի այբուբենը

Այբուբենը լատինական այբուբենի մեծատառ և փոքրատառ տառերն են (A...Z, a...z), թվերը 0, 1...9, հատուկ նշաններ՝ „., .,], (,), +, -, /%, \, !, ;, ?, <, >, =, #, *, ^, ~,, չարտացովող սիմվոլներ, պրոբելներ, նոր տողի անցում:

Իդենտիֆիկատոր. տառերի, թվերի և ընդգծման սիմվոլների հաջորդականություն է, օրինակ `ком_16`, `Time`, `_Min`

Ծառայողական բառեր (բանալի բառեր) – իդենտիֆիկատորներ են, որոնք պահեստավորված են (ռեզերվացված են), այսինքն այնպիսիներ, որոնք չի կարենի օգտագործել որպես ազատ ընտրվող անուններ:

Լեզվում կան 6 դասի լեկսեմներ ազատ ընտրվող և օգտագործվող իդենտիֆիկատորներ, ծառայողական (բանալիային բառեր), հաստատուններ, տողեր (տողային հաստատուններ), գործողություններ (գործողության նշաններ), բաժանիչներ (կետա-դրական նշաններ):

Իդենտիֆիկատորները, որոնք սկսվում են մեկ կամ երկու ընդգծման սիմվոլներով, ռեզերվացված են գրադարաններում և կոմպիլյատորներում օգտագործման համար: Օգտագործվում են կողեր, որոնք չունեն գրաֆիկական ներկայացում դիսփլեյի էկրանին, ստուդիաշարի կամ տպիչի վրա (' \ n' տողի անցում ' \ t ' հորիզոնական տարրույցիա և այլն): Լեզվում օգտագործվում են 3 տևսակի գործողություններ (օպերացիա) ունար (մեկ տեղանի), բինար (երկտեղ), տրինար (եռատեղ): Օրինակ `x<0?x:x`: Նույն նշանը կարող է օգտագործվել և՛ ունար, և՛ բինար, և՛ տրինար հարաբերություններում: Օրինակ `&A` կամ `A&B`; `-A` կամ `A-B`; և այլն:

Բինար օպերացիաները բաժանվում են հետևյալ խմբերի:

- Աղիտիվ (+, -)
- Մուլտիպլիկատիվ (x, /)
- Տեղաշարժեր (-> >)
- Կարգային (&, |, ^)
- Դարաբնություն (>, =)
- Տրամաբանական (&&, ||)
- Վերագրման (P =)
- Կառուցվածքավորված օբյեկտի ընտրություն (., ->)
- Ստորակետ գործողություն (.)
- Փակագիծը որպես օպերացիա ((), [])

Ծրագրավորման էկեմենտար միջոցները

Յուրաքանչյուր կատարվող ֆայլ պարունակում է տայն ֆունկցիա: Պարզագույն դեպքում ծրագրի սկզբնական տեքստը հետևյալն է և այսպիսուն:

```
void main()
{
   օբյեկտների_որոշումը;
   կատարվող_օպերատորներ;
}
```

/* և */- նշանակում է (մեկնաբանություն) բացատրություն, որը անտեսվում է կոմպիլյատորի կողմից:

Include հրահանգը. Հատ ծրագրեր դրվում են մեկ կամ մի քանի ֆայլերով, հաճախ տայն գլխավոր ֆունկցիայի ամենասկզբում:

```
# include <ֆայլ_1>
# include <ֆայլ_2>
# include <ֆայլ-ո>
```

Դրահանգների հայտնվելը բերում է նրան, որ նախապողոցեսորը այդ հրահանգների տեղում տեղադրում է համապատասխանաբար ֆայլ_1, ֆայլ_2,..., ֆայլ_n ֆայլերի տեքստերը:

Ի տարրերություն շատ ուրիշ օպերատորների, *Include հրահանգը* չի ավարտվում կետստորակետով:

Մակրո # define հրահանգի օգնությամբ հնարավոր է դառնում ցույց տալ նախապոցեսորին, որ նրա ցանկացած հայտնվելու ժամանակ սկզբնական ֆայլում տվյալ Մակրո-ի անունը փոխարինի համապատասխան արժեքով: Օրինակ՝

```
# define Pi 3.1415926 կապում է Pi-ին 3.1415926 արժեքի հետ: Մակրո-ի արժեքից հետո չի դրվում (;) կետ-ստորակետ:
```

Նկարագրման ժամանակ փոփոխականից առաջ նշվում է տիպը, ինտո փոփոխականի անունը և հավասարությունից հետո՝ արժեքը:

```
Int height=71;
Float income=26034.12;
```

Printf ֆունկցիան կարելի է օգտագործել սիմվոլների ցանկացած կոմբինացիայի ելքի համար:

Օրինակ `Printf("\n Գուրգենի տարիքը-%d, նրա եկամուտը $%.2f", age, income);`

“*Դուքսի հաջորդական սիմվոլները կուրսորը տեղափոխում են նոր տող:*

“*Գուրգենի տարիքը”* կելքագրվի նոր տողի սկզբից: *%d* սիմվոլները բնութագրում են `age` ամբողջ փոփոխականը, հաջորդ

լիտերային տողը՝ “նրա եկամուտը \$”% 2f. դա ֆորմատի նշումն է, որ տասմորդական ստորակետից հետո ելքագրի միայն երկու թիվ:

Scan օպերատորը հանդիսանում է մուտքի շատ ֆունկցիա-ներից մեկը, որը գոյություն ունի արտաքին գրադարաններում: Փոփոխականի անունից առաջ անհրաժեշտ է դնել & սիմվոլ, որը ցույց է տալիս հասցեն:

Ֆորմատավորող սիմվոլների տիպերը ներկայացնենք աղյուսակի տեսքով:

Ֆորմատի սիմվոլը	Ելքագրվող օբյեկտի տիպը
%c	char (մեկ բայթանոց ամբողջ թիվ)
%s	String (տող)
%d	Int (երկու բայթանոց ամբողջ)
%o	Int (8- ական տեսքով)
%u	unsigned int
%x	Int (տասնվեցական տեսքով)
%ld	Long (տասական տեսքով)
%lo	Long (ուրական տեսքով)
%lu	unsigned long
%lx	Long (տասնվեցական տեսքով)
%f	float/ double (ֆիքսած կետով)
%e	float/ double (երսպննենցիալ տեսքով)
%g	float/ double (f տեսքով կամ e տեսքով)
%lf	long float (ֆիքսած կետով)
%le	long float (երսպննենցիալ ձևով)
%Lg	long float (f կամ e տեսքով կախված արժեքոց)

Օղմակ # include <stdio.h>

```
main()
{
int weight, /*կշիռ*/ height; /*հասակ*/
printf("Նոցրեք Ձեր կշիռը: ");
scanf("%d", &weight);
printf("Նոցրեք Ձեր հասակը: ");
scanf("% d",&height);
printf("\n\n կշիռը=%d, հասակը=%d\n", weight, height);
}
```

Յուրաքանչյուր ֆունկցիայից առաջ պետք է տեղադրել տեղեւկություն ֆունկցիայի վերադարձվող արժեքի տիպի (արդյունքի տիպի) մասին: Եթե ֆունկցիան ոչինչ չի վերադարձնում, ապա նշվում է void տիպը: main() ֆունկցիան հանդիսանում է այն ֆունկցիան, որը մեկնարկվում է կատարելու օպերացիոն համակարգի հրամանով: main() ֆունկցիայի վերադարձվող արժեքը նույնականացնելու համար առաջարկվում է օպերացիոն համակարգին: Եթե ծրագրավորողը չի ենթադրում, որ օպերացիոն համակարգը չի վերլուծելու նրա ծրագրի կատարման արդյունքները, ապա պետք է նշել, որ վերադարձվող արժեքը բացակայում է, այսինքն ունի void տիպը: Եթե արդյունքի տիպի մասին տեղեւկությունը բացակայում է, ապա լրելյայն ընդունվում է, որ main() ֆունկցիան վերադարձնում է int տիպի ամբողջ թվային արժեք:

Օպերատորներ և օպերացիաներ

Ը լեզվի հիմքը կազմում են օպերատորները, որոնց հետևում են կետ-ստորակետը և ծառայում են օպերատորների բաժանման համար: Ընդունված է բոլոր օպերատորները խմբավորել հետևյալ դասերի մեջ՝ վերագրումներ, ֆունկցիաների կանչ, ճյուղավորում և ցիկլեր:

Օպերատորները հաճախ վերաբերում են հետևյալ դասերից մեկին՝
օրինակ՝ if ((c=cube(a*b))>d)

Կազմված է հետևյալ դասերի ներկայացուցիչներից՝ վերագրում, ֆունկցիայի կանչ և ճյուղավորում:

Տարբերում են հետևյալ խմբի օպերացիաները՝ թվաբանական, հարաբերության, վերագրման, տրամաբանական, բիտային, չափի հաշվման և հաջորդականության օպերացիաներ (ստորակետ):

Թվաբանական օպերացիաներից են գումարում(+), հանում (-), բաժանում(/), բազմապատկում (*) և մնացորդ (%):

Բոլոր օպերացիաները (բացառությանը մնացորդի) որոշված են int, char, float տիպի փոփոխականների համար: Մնացորդը որոշված չէ float տիպի փոփոխականների համար: Բոլոր սահող ստորակետով թվաբանական օպերացիաները տեղի են ունենում կրկնակի ճշտությամբ օպերանդների հետ:

Դամեմատության օպերացիաները: Լեզվում որոշված են համեմատման հետևյալ օպերացիաները՝ հավասարության ստուգում (= =), անհավասարության ստուգում (!=), փոքր է (<), փոքր է կամ հավասար (<=), մեծ է (>), մեծ է կամ հավասար (>=): Բոլոր թվար-

կած օպերացիաները մշակում են ոտ տիպի արդյունք: Եթե տվյալ հարաբերությունը օպերանոների միջև ճշմարիտ է, ապա այդ ամբողջի արժեքը մեկ է, եթե կեղծ է, ապա 0:

Սեծ և փոքր տիպի բոլոր օպերացիաները ունեն հավասար առավելություն, ընդ որում բարձր է, քան == և != օպերացիաների առավելությունները: Վերագրման օպերացիայի առավելությունը ցածր է համեմատնան բոլոր օպերացիաների առավելություններից: Դաշվարկման ճիշտ կարգ տալու համար օգտագործվում են փակագծեր:

Տրամաբանական օպերացիաներ: Լեզվում կան երեք տրամաբանական օպերացիաներ՝

&& օպերացիա (and-կ)

|| օպերացիա (or-կամ)

! ժխտում

Տրամաբանական օպերացիաների արգումենտները կարող են լինել ցանկացած թվեր, արդյունքը 0 կամ 1: Տրամաբանական օպերացիաները ունեն ցածր առավելություն: Ուստի այդպիսի օպերացիաներով արտահայտություններում փակագծերը հաճախ են օգտագործվում:

Արտահայտությունների հաշվարկը, որը պարունակում է տրամաբանական օպերացիաներ, տեղի է ունենում ձախից աջ և դադարեցվում է, հենց որ հաջողվուն է որոշել արդյունքը: Օրինակ

if ($i > 50 \ \&\& \ j == 24$)

if ($value1 < value2 \ \&\& (value4 > 50 \ || \ value4 < 20)$)

Վերագրման օպերացիաներ: Վերագրման օպերացիաներից են =, +=, -=, *=, /=, ինչպես նաև պրեֆիքսային (մինչև փոփոխականի օգտագործումը գործողության կատարում) և պոստֆիքսային (փոփոխականի օգտագործումից հետո գործողության կատարում) օպերացիաներ $++$ և $--$: Վերագրման բոլոր օպերացիաները վերագրվում են արտահայտության հաշվման արդյունքի փոփոխականին: Եթե ձախ կողմի տիպը տարբերվում է աջ կողմի տիպից, ապա աջ կողմի տիպը բերվում է ձախ կողմի տիպին: Մեկ օպերատորում վերագրման օպերացիան կարող է հանդիպել մի քանի անգամ: Դաշվարկը կատարվում է աջից ձախ: Օրինակ

$a = (b = c) * d$:

$a += b$ նշանակում է $a = a + b$

$a -= b$ նշանակում է $a = a - b$

$a /= b$ նշանակում է $a = a / b$

Պրեֆիքսային և պոստֆիքսային օպերացիաները ++ և -- օգտագործվում են փոփոխականի արժեքը մեկով մեծացնելու (հնկրեմենտացիա) և փոքրացնելու (դեկրեմենտացիա) համար:
 ++ փոփոխականի արժեքը մեծացնում է մեկով մինչև արտահայտությունում այդ փոփոխականի օգտագործումը:
 +-+ մեծացնում է մեկով փոփոխականի օգտագործումից հետո:
 -- փոքրացնում է մեկով մինչև փոփոխականի օգտագործումը:
 a- փոքրացնում է մեկով փոփոխականի օգտագործումից հետո:

Sizeof(չափ) օպերացիայի արդյունքում ստացվում է բայթերի քանակը, որը զբաղեցնում է օպերանդը:

Օրինակ

Printf ("In Ամբողջի տակ հիշողության չափը %d", sizeof(int));

Printf ("In Սիմվոլի տակ հիշողության չափը %d", sizeof(char));

Նեկավարման օպերատորներ: Ծրագրի աշխատանքի ղեկավարման օպերատորները կոչվում են ծրագրի ղեկավարող կառուցվածքներ: Դրանք են բաղադրյալ օպերատորներ, ընտրման օպերատորներ, ցիկլերի օպերատորներ, անցման օպերատորներ:

Բլոկներ և բաղադրյալ օպերատորներ: Օպերատորների ցանկացած հաջորդականություն, որն ամփոփված է ծևավոր փակագծերում, հանդիսանում է բաղադրյալ օպերատոր (բլոկ): Բաղադրյալ օպերատորը չպետք է ավարտվի (;) կետ-ստորակետով, քանի որ բլոկի սահմանափակիչը ծառայում է հենց ինքը փակագիծը: Բլոկի ներսում յուրաքանչյուր օպերատոր պետք է սահմանափակվի կետստորակետով: Բաղադրյալ օպերատորը կարող է օգտագործվել ամենուրեք, որտեղ լեզվի սինտաքսիսը թույլատրում է կիրառել սովորական օպերատոր:

Դատարկ օպերատոր: Դատարկ օպերատորը ներկայացվում է (;) կետ-ստորակետ սիմվոլով, որից առաջ չկա արտահայտություն: Դատարկ օպերատորը օգտագործվում է այնտեղ, որտեղ լեզվի սինտաքսիսը պահանջում է, սակայն ծրագրի տրամաբանությամբ օպերատորը պետք է բացակայի: Նրա օգտագործման անհրաժեշտությունը հաճախ առաջանում է, եթե գործողությունը, որը կարող է կատարվել ցիկլի մարմնում, ամբողջությամբ տեղադրվում է ցիկլի վերնագրում:

Ընտրման օպերատորները պայմանական օպերատորները (if) և փոխանցատիչներն են (switch):

Եյուղավորման օպերատորներից են if, if else, ?, switch և go to օպերատորները: Եյուղավորման օպերատորների ընդհանուր տեսքը հետևյալն է:

```
If (տրամաբանական արտահայտություն)
օպերատոր_1;
else
օպերատոր_2;
```

? օպերատորն ունի հետևյալ կառուցվածքը
<տրամաբան.արտահայտություն>?<արտ._1>; <արտ._2>;
Եթե տրամաբանական արտահայտության արժեքը ճշմարիտ է,
ապա հաշվարկվում է արտահայտություն_1-ը, հակառակ դեպքում
հաշվում է արտահայտություն_2-ը:

```
Switch օպերատորն ունի հետևյալ կառուցվածքը
switch (անբողջ տիպի արտահայտություն)
{
case արժեք_1;
օպերատոր_1-հաջորդականություն;
break;
case արժեք_2;
օպերատոր_2-հաջորդականություն;
break;
case արժեք_n;
օպերատոր_n-հաջորդականություն;
break;
default;
օպերատոր;
}
```

Հեզկում ցիկլերի օպերատորները լինում են երեք տեսակի նախապայմանով (while), հետպայմանով (do) և պարամետրական (for).

Ցիկլի օպերատորների կառուցվածքները կարելի է նկարագրել հետևյալ կերպ

while(տրամաբանական արտահայտություն) /*ցիկլ նախապայմանի ստուգումով*/
օպերատոր,

Do
օպերատոր;

while (տրամաբանական արտահայտություն); /* ցիկլ հետպայմանի ստուգումով */

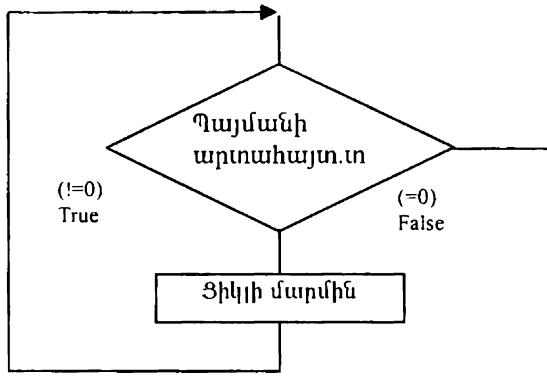
for (արժեքների տեղադրում, ստուգում, նոր_արժեք)
օպերատոր;

բլոկ սխեմայի տեսքով ներկայացնենք նրանց աշխատանքի սկզբունքները

/* Նախապայմանով ցիկլ */

օրինակ:

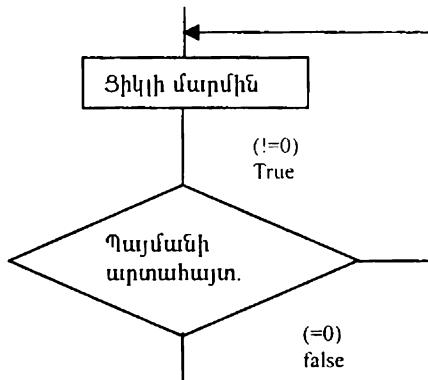
```
i=2  
b=1.0;  
r=x;  
while(r>eps || r<-eps)  
{  
b=b+r;  
r=r*x/i;  
i++;  
}
```



/* Հետպայմանով ցիկլ */

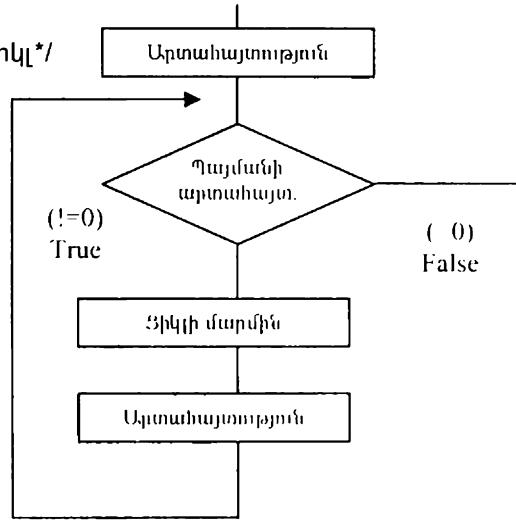
օրինակ:

```
i=1;  
b=0.0;  
r=1.0;  
do {  
b=b+r;  
r=r*x/i;  
i++  
}  
while(r>eps || r<-eps)
```



/*Պարամետրական ցիկլ*/

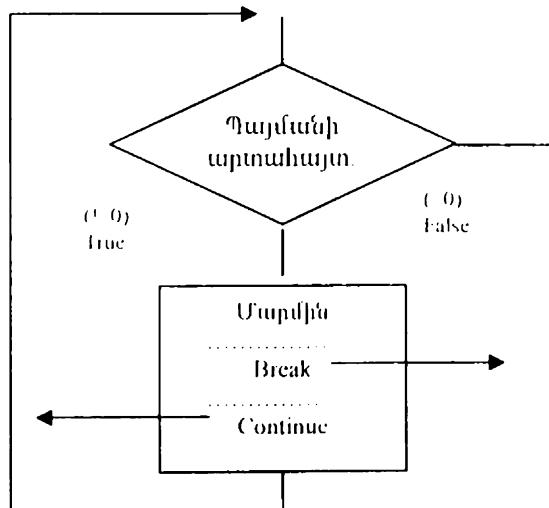
```
օրինակ  
l=2  
b=1.0;  
r=x;  
for ( ; l >eps || r<-eps)  
{  
b=b+r;  
r=r*x/i;  
i=i+1;  
}
```



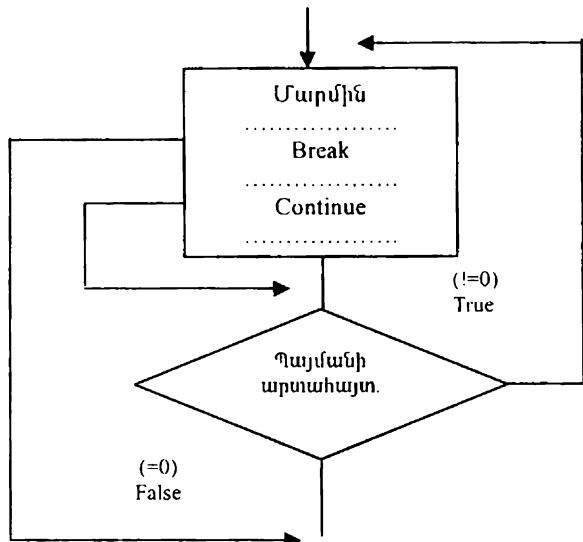
Անցման օպերատորները կատարում են դեկավարման անպայման փոխանցում, goto (անպայման անցում), continue (ցիկլի ընթացիկ իտերացիայի ավարտում), break (ելք ցիկլից կամ փոխանցատիչից), return (Վերադարձ ֆունկցիայից):

Break և Continue օպերատորների աշխատանքի սկզբունքները ներկայացնենք սխեմայի տեսքով:

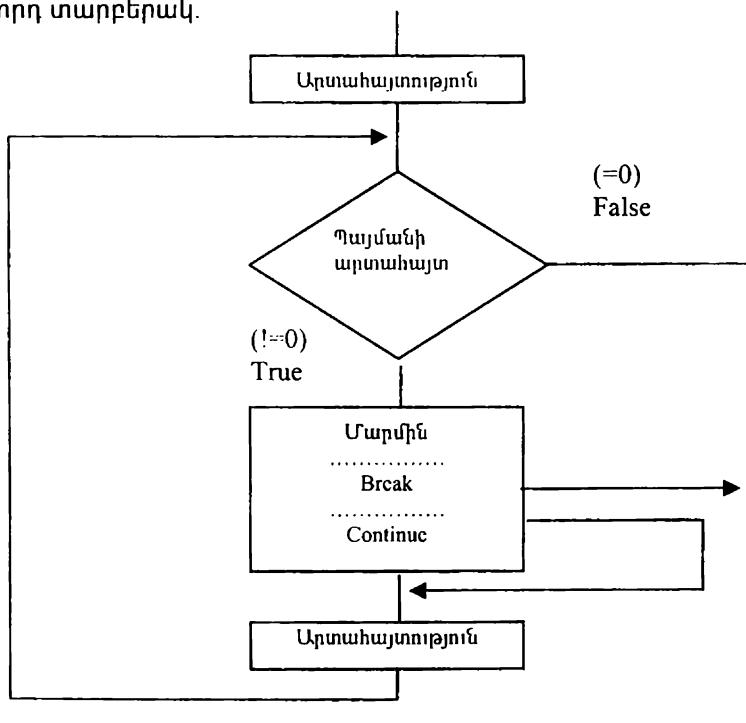
Առաջին տարրերակ .



Երկրորդ տարբերակ.



Երրորդ տարբերակ.



Պարզագույն ծրագրերի տրամաբանական կազմակերպումը ֆունկցիաների և զանգվածների օգտագործմամբ

Ծրագրերի ապահովման մշակման (նախագծման) գործընթացը ենթադրում է բարդ խնդիրները տրոհել ավելի պարզ խնդիրների և առաջադրանքների հավաքածուի: C-ն պաշտպանում է ֆունկցիաները որպես տրամաբանական միավորներ (ծրագրի տեքստի բլոկ), որոնք ծառայում են կոնկրետ առաջարանքի կատարմանը: Ծրագրային ապահովման նախագծման կարենը հայեցակետ է ֆունկցիոնալ դեկոմպոզիցիան:

Ֆունկցիան ունի գրո կամ ավելի ֆորմալ պարամետրեր և վերադարձնում է սկայար տիպի արժեք, void (դատարկ) տիպ կամ նշիչ: Ֆունկցիայի կազմելու ժամանակ արժեքը, որը տրվում է նույտում, պետք է հանապատասխանի ֆունկցիայի նկարագրման մեջ ֆորմալ պարամետրերի տիպին և թվին:

C-ն ներկայացնում է անսովոր բարձր ճկունություն ծրագրերի ֆիզիկական կազմակերպման համար կամ և ծրագրային համակարգերի հանար:

Զանգվածները նույն տիպի օբյեկտների հավաքածու են, որոնց դիմելը իրականացվում է զանգվածում անմիջապես ինդեքսով: Զանգվածներին դիմելը C-ում իրականացվում է նշիչների օգնությամբ: Զանգվածները կարելի է նկարագրել հետևյալ կերպ:

տվյալների_ տիպ զանգվածի_ անուն [զանգվածի չափը]:

Օգտագործելով զանգվածի անունը և ինդեքսը կարելի է դիմել զանգվածի լեմենտին:

Զանգվածի անուն [ինդեքսի արժեքը]

Ինդեքսի արժեքը պետք է ընկած լինի 0-ից մինչև զանգվածի չափից մենք միավոր փորբ միջակայթում, որը նշված է նրա նկարագրման ժամանակ: Նշենք զանգվածների նկարագրման մի քանի օրինակներ:

Char name[20];

Int drades[125];

Float income[30];

Double measurements[1500];

/* Օրագիր, որը (լուսաբանում է) ցուցադրում է զանգվածների օգտագործումը */

/* array.c Փայլը */

#include < stdio.h>

#define size 1000

```

int data [size];
main ()
{
extern float average (int a[ ] , int S);
int I;
for (I = 0 ; I < size ; I + +)
data [ I ] = I ;
print f ("\n data զանգվածի միջին արժեքը=%f \n", average
(data, size));
}
float average (int a [ ], int S)
{
float sum = 0.0 ;
int I ;
for (i=0; I < s ; i++)
sum += a [ I ];
return sum / s ;
}

```

3.2. Ծրագրավորման C++ լեզուն

Ծրագրավորման զարգացման մեջ կարևոր փուլ է հանդիսանում C++ լեզվի ստեղծումը և լայն տարածումը: Այս լեզուն, պահպանելով համակարգային և կիրառական ծրագրեր գրելու համընդհանուր ճանաչման արժանացած է լեզվի միջոցները (որը պրոցեդուրակողմնորոշված լեզու է), ծրագրավորման ոլորտ է ներմուծել ծրագրային ապահովման մշակման նոր տեխնոլոգիական մոտեցման լայն հնարավորություններ, որը ստացել է “օբյեկտակողմնորոշված ծրագրավորում” անվանումը: Օբյեկտակողմնորոշված ծրագրավորման գաղափարների ներդրումը գործնականում տալիս է ինֆորմատիկայի նոր տիրույթների զարգացման հնարավորություն, նշանակալի բարձրացնում է ստեղծվող ծրագրային միջոցների տեխնոլոգիական մակարդակը, մշակման և ուղեկցման վրա ծախսերի կրճատմանը, դրանց կրկնակի օգտագործմանը ներքաշելով նրան ԵՇՍ-ի ինտելեկտուալ հնարավորությունների ընդլայնման պրոցեսի մեջ: C++-ը ընդհանուր նշանակության ծրագրավորման լեզու է: Յայնի է իր արդյունավետությամբ և խնայողությամբ: C++ -ի նշված առավելությունները ապահովում են մշակման լավ որակ համարյա ցանկացած տեսակի ծրագրային

արտադրանքի համար: C++-ի օգտագործումը՝ որպես գործիքային լեզու, թույլ է տալիս ստանալ արագ և կոմպակտ ծրագրեր: Հատ դեպքերում ծրագրերը, որոնք գրված են C++-ով, համեմատելի են իրենց արագությամբ Ասսեմբլեր լեզվով գրված ծրագրերի հետ: Թվարկենք C++-ի որոշ էական առնձնահատկությունները:

- C++-ը ապահովում է կառուցվածքային ծրագրավորման օպերատորների լրիվ հավաքածու:
- C++-ը առաջարկում է անսովոր շատ մեծ քանակությամբ օպերացիաների հավաքածու: C++-ի շատ օպերացիաներ համապատասխանում են մերենայական հրամաններին, ուստի թույլ է տալիս մերենայական կողի ուղիղ տրանսլյացիա:
- C++-ը պաշտպանում է փոփոխականների և ֆունկցիաների վրա նշիշների օգտագործում: Ծրագրի օբյեկտների վրա նշիշները համապատասխանում են այդ օբյեկտի մերենայական հասցեին: Նշիշների խելացի օգտագործման միջոցով կարելի է ստեղծել արդյունավետ ծրագրեր այնպես, ինչպես դա անում է ԷՌՍ-ը: C++-ը պաշտպանում է նշիշների բարանությունը և դրանով թույլ է տալիս իրականացնել անմիջապես հիշողության հասցեների հետ դիմելու թույլտվություն և մանիպուլյացիաներ:

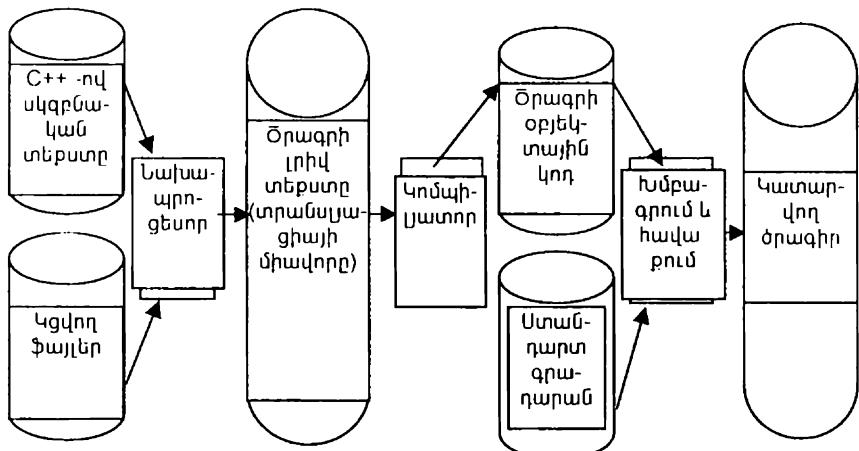
Սակայն պետք է նշել, որ C++ լեզվի առավելությունը դառնում է ակնհայտ մեծ ծրագրային նախագծերի իրականացման ժամանակ:

C++ ուն յուրաքանչյուր ծրագիր իրենից ներկայացնում է նախապրոցեսորային հրահանգների նկարագրություն և գլոբալ օբյեկտների ու ֆունկցիաների հաջորդականություն: Յուրաքանչյուր արտահայտություն դառնում է օպերատոր, եթե արտահայտության վերջում հայտնվում է կետ-ստորակետ: Նախապրոցեսորային հրահանգները (օրինակ `#include`; `#Define`) դեկավարում են ծրագրի տեքստի ծևափոխությունը մինչև նրա կոմպիլյացիան: Օբյեկտները անհրաժեշտ են ծրագրում նշակող տվյալների ներկայացման համար: Ֆունկցիան որոշում է ծրագրի պոտենցիալ հնարավոր գործողությունը: C++ լեզվում ծրագիրը պետք է ծևակերպվի մեկ կամ մի քանի տեքստային ֆայլերի տեսքով: Տեքստային ֆայլերը տրոհվում են տողերի: Յուրաքանչյուր տողի վերջում կա նրա ավարտի մասին նշան, պլյուս դեկավարող սիմվոլ, որը նշում և անցումը նոր տողի: Ակզրնական ծրագիրը տեքստային ֆայլի տեսքով անցնում է նշակման երեք պարտադիր փուլեր

- տեքստի նախապրոցեսորային ծևափոխություն
- կոմպիլյացիա

- դասղասում (կոմպանովելա կապերի խմբագրում և հավաքում):

Կատարվող ծրագրի նախապատրաստման սխեման ունի հետևյալ տեսքը



Ծրագրի տեքստում նախապրոցեսորային հրահանգների կատարումից հետո չի մնում ոչ մի նախապրոցեսորային հրահանգ:

C++ - ում կոնսոլային մուտք և ելքը

Քանի որ C++-ը դա բարելավված C-ն է, հետևաբար C լեզվի բոլոր էլեմենտները պարունակվում են և C++-ում: Դա հասկացվում է, որ բոլոր ծրագրերը, որոնք գրված են C-ով, լույսայն նույնպես հանդիսանում են C++-ի ծրագրեր (որոշ բացառությամբ): C++-ում մուտք/ելքը կատարվում է՝ օգտագործելով օպերացիաներ, այլ ոչ թե նուտք/ելքի ֆունկցիաներ:

Ստեղծաշարից մուտք ու ելքի համար օգտագործվում է `Cin >>` փոփոխական, ելքի համար՝ `Cout<<` արտահայտություն:

Այստեղ արտահայտություն բառը կարող է լինել C++-ի ցանկացած իրական արտահայտություն՝ ներառյալ ելքի ուրիշ արտահայտություն:

C++-ում մուտք/ելքի օպերացիայի ճիշտ օգտագործման համար պետք է ծրագրում ընդգրկել `iostream.h` վերնագրային ֆայլը:

C++ - ում մեկնաբանությունը (կոմենտարիա)

C++-ում ծրագրի մեջ մեկնաբանություն կարելի է ընդգրկել 2 տարբեր եղանակով: Առաջինը ստանդարտ եղանակն է ինչպես և C-ում, այսինքն մեկնաբանությունը սկսվում է /*-ով և ավարտվում */-ով: Այս տիպը չի կարող ներդրված լինել: Երկրորդ տիպը, որով կարող ենք գրել մեկնաբանություն, C++-ծրագրում հանդիսանում է մեկ տողանի մեկնաբանությունը, որը սկսվում է // -ով և ավարտվում է տողի վերջով:

C++ լեզվի բազային հասկացությունները

Կան ցիկլերի կազմակերպման 3 օպերատորներ While, do, և For. Ցիկլի օպերատորը բաղկացած է երկու էլեմենտից վերնագրից և մարմնից: Լեզվում յուրաքանչյուր ծրագիր ֆունկցիաների համախնդրություն է: Ֆունկցիայի որոշման մեջ նշվում է գործողությունների հաջորդականությունը, որը կատարվում է ֆունկցիային դիմելու ժամանակ ֆունկցիայի անունը, արդյունքի տիպը և ֆորմալ պարամետրների համախմբությունը, որոնք փոխարինվում են փաստացի արգումենտներով ֆունկցիային դիմելու ժամանակ: Ֆունկցիայի որոշման կառուցվածքը ունի հետևյալ տեսքը

Արդյունքի_տիպը ֆունկցիայի_անունը (ֆորմալ_պարամ._բնութ.)

{

օբյեկտների_որոշումը;
կատարվող_օպերատորներ;

}

Դասեր

Դավանաբար C++-ի առավել կարևոր հասկացություններից մեկը հանդիսանում է դասը, որը իրենից ներկայացնում է օբյեկտների ստեղծման համար մեխանիզմ: Դասի նկարագրման սինտաքսը նման է կառուցվածքի նկարագրման սինտաքսին:

Բներնք հիմնական ձևը

Class դասի_անուն {

դասի փակ ֆունկցիաները և փոփոխականները

Public:

դասի բաց ֆունկցիաները և փոփոխականները

} օբյեկտների_ցուցակը;

Դասի նկարագրման մեջ օբյեկտների ցուցակը չի հանդիսանում պարտադիր: Ինչպես և կառուցվածքի դեպքում, դասի օբյեկտները կարող ենք հայտարարել ինտու անհրաժեշտության դեպքում: Չնայած դասի _անունը նույնպես պարտադիր չէ գործնական տեսակետից, այն պետք է: Դասի _անունը դառնում է տվյալների տիպի նոր անուն, որը օգտագործվում է դասի օբյեկտների հայտարարման համար: Ֆունկցիաները և փոփոխականները, որոնք հայտարարվում են դասի հայտարարման ներսում, դառնում են այդ դասի անդամներ (members): Դասում հայտարարված բոլոր ֆունկցիաները և փոփոխականները լրելյայն դառնում են այդ դասի համար փակ: Դա նշանակում է, որ դրանք հասանելի են միայն այդ դասի այլ անդամների համար: Դասի բաց անդամների հայտարարման համար օգտագործվում է Public բանալիային բառը, որին հետևում է երկու կետ (վերջակետը): Public բառից հետո հայտարարված բոլոր ֆունկցիաները և փոփոխականները հասանելի են և՝ դասի ուրիշ անդամների համար, և՝ ծրագրի ցանկացած այլ մասի համար, որում պարունակվում է դասը: Օրինակ՝

```
Class myClass {
    // դասի փակ էլեմենտ
    int a;
    Public:
        Void set_a(int num)
        int get_a();
}
```

Այս դասն ունի մեկ փակ ա փոփոխական և երկու բաց ֆունկցիաներ՝ set_a() և get_a(): Ուշադրություն դարձնենք, որ ֆունկցիայի նախատիպերը հայտարարվում են դասի ներսում: Ֆունկցիաները, որոնք հայտարարվում են դասի նկարագրման ներսում, կոչվում են անդամ_ֆունկցիաներ (member functions): Քանի որ ա-ն հանդիսանում է դասի փակ փոփոխական, այն անհասանելի է myClass-ից դուրս ցանկացած ֆունկցիայի համար: Սակայն, քանի որ set_a() և get_a()—ն հանդիսանում են myClass-ի անդամներ, նրանք ունեն ա-ին դիմելու իրավունք: Բացի դա, set_a() և get_a()—ն հայտարարվում են myClass-ի բաց անդամներ և կարող են կանչվել ծրագրի ցանկացած մասից, որոնք օգտագործում են myClass-ը:

Չնայած set_a() և get_a() ֆունկցիաները հայտարարվել են myClass-ում, դրանք դեռ որոշված չեն: Ֆունկցիա անդամի որոշման համար մենք պետք է կապենք դասի անունը, որի մասն է

հանդիսանում ֆունկցիա_անդամը, ֆունկցիայի անվան հետ: Դա ծեռք է բերվում դասի անունից հետո երկու հատ 2 կետից հետո ֆունկցիայի անվան նկարագրման ճամապարհով: Երկու հատ երկու կետը կոչվում են տեսանելիության տիրույթի ընդլայնման օպերացիա (scope resolution operator): Ֆունկցիա_անդամի տակու համար օգտագործվում է հետևյալ ընդհանուր ձևը

դասի_անունի_տիպ::ֆունկցիայի_անուն (պարամետրերիցուցակ)

```
{
    ... // ֆունկցիայի մարմինը
}
```

Դասի հայտարարումը հանդիսանում է տրամաբանական վերացրկում, որը տակսի օբյեկտի նոր տիպ: Օբյեկտի հայտարարումը ստեղծում է այդպիսի տիպի օբյեկտի ֆիզիկական եռթյունը:

Այն բանից հետո, եթե դասի օբյեկտը ստեղծվել է, կարելի է դիմել դասի բաց անդամներին օգտագործելով (.) կետ օպերացիա ինչպես որ իրականացվում է կառուցվածքի անդամին դիմելիս: Յուրաքանչյուր օբյեկտ պարունակում է դասում հայտարարված բոլոր տվյալների սեփական պատճեն:

C և C++ - ի տարրերությունների մասին

- Առաջին, եթե C-ում ֆունկցիան չունի պարամետրեր, նրա նախատիպը պարունակում է ֆունկցիայի պարամետրերի ցուցակում void բառը: Օրինակ, chart f1(void);
C++-ում void բառը չի հանդիսանում պարտադիր: Յետևաբար C++-ում նախատիպը սովորաբար գրվում է այսպես Chart f1();
C++-ում հետևյալ երկու հայտարարումները հանդիսանում են int f1(); int f1(void);
- Մյուս ոչ մեծ տարրերությունը C և C++-ի միջև այն է, որ C++-ի ծրագրերում բոլոր ֆունկցիաները պետք է ունենան նախատիպեր:
- Երրորդ տարրերությունն այն է, որ եթե C++-ում ֆունկցիան ունի void-ից տարրեր վերառարձվող արժեքի տիպ, ապա բարեւ օպերատորը այդ ֆունկցիայի ներսում պետք է պարունակի տվյալ տիպի արժեքը:
- Քաջորդը, որ C++-ի ծրագրերում կարող ենք լոկալ փոփոխականների հայտարարման համար ընտրել տեղը, ի տարրերություն C-ի, որտեղ լոկալ փոփոխականները կարող են հայտարարվել:

րարվել միայն բլոկի սկզբում «գործողություն» ցանկացած օպերատորից առաջ:

Ֆունկցիայի ծանրաբեռնման հասկացությունը

Դասերից հետո հավանաբար հաջորդ և ոչ սովորական հնարավորությունը ֆունկցիայի ծանրաբեռնումն է (function overLoading): Ֆունկցիայի ծանրաբեռնումը ոչ միայն ապահովում է այն մեխանիզմը, որի միջոցով C++-ում ձեռք է բերվում պոլիմորֆիզմի (բազմաձևության) տիպերից մեկը, այլև ձևավորում է այն միջուկը, որի շուրջը զարգանում է C++-ում ծրագրավորման միջավայրը: C++-ում երկու կամ ավելի ֆունկցիաներ կարող են ունենալ նույն անունը՝ տարրերվելով կամ տիպով, կամ արգումենտների թվով, կամ և՛ մեկով, և՛ մյուսով: Եթե երկու կամ ավելի ֆունկցիաներ ունեն նույն անունը, ասում են, որ նրանք ծանրաբեռնված են: Ծանրաբեռնվող ֆունկցիաները հնարավորություն են տալիս պարզեցնել ծրագիրը՝ թույլ տալով դիմել նույն անվանը իմաստով մոտ գործողությունների կատարման համար: Ֆունկցիայի ծանրաբեռնումը շատ պարզ է, ուղղակի հայտարարում և տալիս ենք բոլոր պահանջվող տարրերակները: Կոմպիլյատորը ընտրում է ճիշտ տարրերակը: C++-ում կարելի է նաև ծանրաբեռնել օպերացիաները:

C++ - ի բանալիային բառերը

Ը լեզվում ANSI ստանդարտում 32 բանալիային բառերին C++-ում առաջարկվում է ևս 29-ը: Նշենք այդ լրացուցիչ բանալիային բառերը.

asm	friend	protected	try
bool	inLine	public	typeid
catch	mutable	reinterpret_cast	using
class	namespace	static_cast	virtual
const_cast	new	template	vchar_t
delete	operator	this	
dinamic_cast	overload	throw	
false	private	true	

Կոնստրուկտորներ և դեստրուկտորներ

Եթե գրել ենք շատ երկար ծրագիր, ապա պետք է հիշենք, որ ծրագրի որոշ մասերում սովորաբար պահանջվում է ինիցիալիզացիա (արժեքների տեղադրում): Ինիցիալիզացիայի անհրաժշտությունը ավելի հաճախ հանդես է գալիս օբյեկտների հետ աշխատանքի ժամանակ: C++-ը ներկայացնում է դասի նկարագրման մեջ ընդգրկվող ֆունկցիա_կոնստրուկտոր (constructor function): Դասի կոնստրուկտորը կանչվում է այդ դասի օբյեկտի ստեղծման ժամանակ, ամեն անգամ: Կոնստրուկտորը ունի նույն անունը, ինչ որ դասը, որի մասն է կազմում ինքը և չունի վերադարձվող արժեք: Օրինակ՝

```
# include <iostream.h>
Class myClass {
    int a ;
    Public:
        myClass(); // կոնստրուկտոր
        ~myClass(); // դեստրուկտոր
        void show();
    };
    myClass::myClass()
    {
        cout << "կոնստրուկտորում պարունակությունը \n";
        a = 10;
    }
    myClass:: ~myClass()
    {
        cout << "հեռացում...\n";
    }
    void myclass:: show()
    {
        cout << a << "\n";
    }
    main()
    {
        myClass ob;
        ob.show();
        return 0;
    }
```

Գլոբալ օբյեկտների համար օբյեկտի կոնստրուկտորը կանչվում է այն ժամանակ, երբ սկսվում է կատարվել ծրագիրը: Լոկալ օբյեկտների համար կոնստրուկտորը կանչվում է ամեն անգամ օպերատորի կատարման ժամանակ, որը հայտարարում է փոփոխականը: Կոնստրուկտորի հակադարձ ֆունկցիան դեստրուկտորն է (destructor): Այդ ֆունկցիան կանչվում է օբյեկտի հեռացման ժամանակ: Լոկալ օբյեկտները հեռացվում են այն ժամանակ, երբ դրանք դուրս են գալիս տեսանելիության տիրույթից: Գլոբալ օբյեկտները հեռացվում են ծրագրի ավարտման ժամանակ:

Կոնստրուկտորին կարելի է փոխանցել արգումենտ: Դրա համար ուղղակի որոշման մեջ ավելացնում են անհրաժեշտ պարամետրերը: Այնուհետև օբյեկտի հայտարարման ժամանակ պարամետրերը տրվում են որպես արգումենտ:

Ժառանգականություն հասկացությունը

Ժառանգականությունը մեխանիզմ է, որի միջոցով մի դասը կարող է ժառանգել մյուսի հատկությունները: Ժառանգականությունը բույլ է տալիս կառուցել դասերի հիերարխիա՝ անցնելով ընդհանուրից ավելի մասնակիի: Երբ մի դասը ժառանգում է մյուսին, դասը, որը ժառանգվում է, կոչվում է բազային դաս (base class), ժառանգողը կոչվում է ածանցյալ դաս (derived class): Սովորաբար ժառանգականության պրոցեսը սկսվում է բազային դասը տալուց: Բազային դասը որոշում է բոլոր այն որակները, որը պետք է լինի ընդհանուր բոլոր ածանցյալ դասերի համար: Ըստ եւրյան, բազային դասը իրենից ներկայացնում է մի շարք բնութագրիչ գծերի ավելի ընդհանուր նկարագրություն: Ածանցյալ դասը ժառանգում է այդ ընդհանուր գծերը և ավելացնում է հատկություններ, որոնք բնութագրական են տվյալ դասի համար: Օրինակ.

```
// բազային դասի տալը.
Class B {
    int i;
    Public:
        void set_i(int n);
        int get_i();
    };
}
```

```
// բազային դասի տալը
class D: Public B {
    int j;
    Public:
        void set_j(int n);
        Int mul();
    };
}
```

Օբյեկտների վրա նշիչները

Մինչև այժմ մենք օբյեկտների անդամներին դիմելու համար օգտագործել ենք (.) կետ օպերացիա: Եթե աշխատում ենք օբյեկտների հետ, ապա դա ճիշտ է: Սակայն օբյեկտների անդամներին դիմելը կարելի է իրականացնել և այդ օբյեկտների վրա նշիչի միջոցով: Այդ դեպքում սովորաբար կիրառվում է սլաք (->) օպերացիա: Դայտարարում ենք օբյեկտի վրա նշիչ: Տալիս ենք այդ օբյեկտի դասի անունը, այնուհետև փոփոխականի անունը, որին նախորդում է աստղանիշ: Օբյեկտի հասցեն ստանալու համար նրանից առաջ անհրաժեշտ է & օպերատոր ճիշտ այնպես, ինչպես դա արվում է ցանկացած այլ տիպի փոփոխականի հասցե ստանալու համար: Ինչպես և ցանկացած այլ նշիչի համար, եթե ինկրեմենտացմում ենք նշիչը օբյեկտի վրա, նա ցույց է տալու այդ տիպի հաջորդ օբյեկտը:

Օրինակ.

```
# include <iostream.h>
class myClass {
int a;
public:
myClass(int x); // կոնստրուկտոր
int get();
};
myClass:: myClass(int x)
{
a=x;
}
int myClass:: get()
return a ;
}
main()
{
myClass ob(120); // օբյեկտի ստեղծում
myClass*p; // օբյեկտի վրա նշիչի ստեղծում
p = &ob; // ob-ի հասցեի փոխանցումը թիմ պ-ին
cout << "օբյեկտի օգտագործման ժամանակ արժեքը: " <<ob.get(),
cout << "\n";
cout << "նշիչների օգտագործման ժամանակ արժեք" <<ob->get(),
return 0;
}
```

Դասերը, կառուցվածքները և միավորումները

Միակ տարբերությունը դասի և կառուցվածքի միջև այն է, որ դասի անդամները լրելյան փակ են, իսկ կառուցվածքի անդամները՝ բաց: Ցուցադրենք կառուցվածքի նկարագրման ընդլայնված սինտաքսիսը:

```
Struct տեղի_անունը {  
    // բաց ֆունկցիա_անդամներ և տվյալներ _անդամներ  
    Private:  
        // փակ ֆունկցիա_անդամներ և տվյալներ _անդամներ  
    } օբյեկտների _ ցուցակ
```

Այսպիսով, C++-ի ֆորմալ սինտաքսիսի համապատասխան Struct-ը և Class-ը ստեղծում են տվյալների նոր տիպեր: Ուշադրություն դարձնենք նոր Private բանալիային բառին, այն հաղորդում է կոնպիլյատորին, որ նրանից հետո եկող անդամները հանդիսանում են փակ՝ այդ դասի համար: C++-ում միավորումը նույնպես որոշում է դասի տիպ, որի մեջ ֆունկցիաները և տվյալները կարող են պարունակվել անդամների տեսքով: Միավորումը ննան է կառուցվածքին այն առումով, որ նրանուն լրելյան բոլոր անդամները բաց են այնքան ժամանակ, քանի դեռ չի օգտագործվում Private որոշիչ: Գլխավորը այն է, որ միավորումը C++ ուն բոլոր տվյալներ_անդամները պահպում են հիշողության մեջ տիրույթում: Միավորումը կարող է պարունակել կոնստրուկտորներ և դեստրուկտորներ: Եթե կառուցվածքի և դասերի միջև գոյություն ունեն առաջին հայացքից որոշ բացքողումներ, ապա միավորումների ժամանակ այդ ասել չի կարելի: Օբյեկտակողմնորոշված լեզվում կարենոր է ինկապսուլյացիայի սատարումը: Ուստի ծրագրերը և տվյալները միավորենու կարողությունը բույլ է տալիս ստեղծել տվյալների տիպեր, որտեղ բոլոր տվյալները օգտագործում են ընդհանուր հիշողություն: Դա չի կարելի անել՝ օգտագործելով դասերը: Միավորումը չի կարող ունենալ Static ատրիբուտով անդամ:

Ներդրվող ֆունկցիաներ

C++-ում կարելի է տալ ֆունկցիա, որը փաստորեն չի կանչվում, այլ նրա մարմինը ներդրվում է ծրագրում նրա կանչելու տեղում: Ներդրվող (inline) ֆունկցիայի առավելությունը կայանում է նրանում, որ այն կապված չէ ֆունկցիայի կանչի և վերադարձման մեխանիզմի հետ: Դա նշանակում է, որ ներդրվող ֆունկցիան

կարող է կատարվել սովորականից բավական արագ: Ներդրվող ֆունկցիաների թերությունն այն է, որ եթե դրանք չափից շատ մեծ են և կանչվում են հաճախ, ապա ծրագրի ծավալը շատ է մեծանում, դրա համար սովորաբար սահմանափակվում ենք կարճ ֆունկցիաներով: Ներդրվող ֆունկցիայի հայտարարման համար ուղակի պետք է օգտագործել `inline` բնութագրիչ ֆունկցիայի որոշումից առաջ:

```
Օրինակ // Ներդրվող ֆունկցիայի օրինակ
#include <iostream.h>
inline int even(int x)
{
    return !(x%2);
}
main()
{
    if (even(10)) cout << "10-ը զույգ է \n";
    if (even(11)) cout << "11-ը կենտ է \n";
    return 0;
}
```

Եթե ինչ-որ սահմանափակում ներդրվող ֆունկցիայի օգտագործման վրա խախտվել է, կոմպիլյատորը դրա փոխարեն ստեղծում է (գեներացնում է) սովորական ֆունկցիա: Եթե ֆունկցիա_անդամի որոշումը բավականին կարճ է, ապա կարելի է ընդգրկել դասի հայտարարման մեջ:

Ընկերային (բարեկամական) ֆունկցիաներ

Եթե անհրաժեշտ է ֆունկցիայի դասի փակ անդամներին դիմելու իրավունք ստանալ առանց նրա անդամ լինելու, C++-ը առաջարկում է ընկերային ֆունկցիա (friend functions): Ընկերային ֆունկցիաները չեն հանդիսանում դասի անդամներ, բայց այնուամենայնիվ ունեն նրա փակ էլեմենտներին դիմելու իրավունք: Ընկերային ֆունկցիան տրվում է այնպես, ինչպես սովորական ֆունկցիան, որը չի հանդիսանում դասի անդամ: Սակայն դասի հայտարարման մեջ, որի համար ֆունկցիան պետք է լինի ընկերային, անհրաժեշտ է ընդգրկել նրա նախատիպը, որից առաջ դրվում է `friend` բանալիային բառը: Օրինակ //ընկերային ֆունկցիայի օգտագործման օրինակ

```

# include <iostream.h>
class myclass
{
int n, d;
public:
myclass(int i, int j){n=i;d=j;}
// myClass ֆունկցիայի համար ընկերայինի հայտարարում
friend int isfactor(myClass ob);
};
int isfactor(myClass ob)
{
if (!(ob.n%ob.d)) return 1;
else return0;
}
main()
{
myClass ob1(10,2), ob2(13,3);
if (isfactor(ob1)) cout << " 10-ը առանց մնացորդի բաժանվում է 2-ի \n";
else cout<< " 10-ը առանց մնացորդի չի բաժանվում է 2-ի \n";
if (isfactor(ob2)) cout << " 15-ը առանց մնացորդի բաժանվում է 3-ի \n";
else cout<< " 15-ը առանց մնացորդի չի բաժանվում է 3-ի \n";
return 0;
}

```

Վիրտուալ ֆունկցիաներ

Վիրտուալ ֆունկցիան (Virtual function) հանդիսանում է դասի անդամ ֆունկցիա: Նա հայտարարվում է բազային դասի ներսում և վերառորշվում է ածանցյալ դասում: Վիրտուալ ֆունկցիայի ստեղծման համար ֆունկցիայի հայտարարումից առաջ դրվում է virtual բանալիային բառ: Եթե դասը, որը պարունակում է վիրտուալ ֆունկցիա, ժառանգվում է, ապա ածանցյալ դասում վիրտուալ ֆունկցիան վերառորշվում է: Ըստ Էռլյան վիրտուալ ֆունկցիան իրականացնում է «մեկ ինտերֆեյս, բազմաթիվ մեթոդներ» գաղափարը, որը ընկած է պոլիմորֆիզմի հիմքում: Բազային դասի ներսում վիրտուալ ֆունկցիան տալիս է այդ ֆունկցիայի ինտերֆեյսը: Ածանցյալ դասում վիրտուալ ֆունկցիայի յուրաքանչյուր վերառորշում որոշում է նրա իրականացումը՝ կապված ածանցյալ

դասի յուրահատկության հետ: Այսպիսով, վերառողջումը ստեղծում է կոնկրետ մեթոդ: Ածանցյալ դասում վիրտուալ ֆունկցիայի վերառոշման համար virtual բանալիային բառը չի պահանջվում: Վիրտուալ ֆունկցիան կարող է կանչվել այնպես, ինչպես և ցանկացած այլ ֆունկցիա_անդամ: Սակայն ավելի հետաքրքիր է կանչել վիրտուալ ֆունկցիան նշիչի օգնությամբ, որի շնորհիվ պաշտպանվում է դինամիկ պոլիմորֆիզմը: Ինչպես գիտենք, բազային դասի նշիչը կարելի է օգտագործել որպես ածանցյալ դասի օբյեկտի վրա նշիչ: Եթե երկու կամ ավելի տարբեր դասեր հանդիսանում են բազայինից ածանցյալ, որը պարունակում է վիրտուալ ֆունկցիա, ապա եթե բազային դասի նշիչը ցույց է տալիս ածանցյալ դասերի տարբեր օբյեկտներ, կատարվում է վիրտուալ ֆունկցիայի տարբեր վերսիաներ: Այդ պրոցեսը հանդիսանում է դինամիկ պոլիմորֆիզմի սկզբունքների իրականացում: Փաստորեն, եթե ասում են դասի մասին, որը պարունակում է վիրտուալ ֆունկցիա, ասում են դասի պոլիմորֆիզմի մասին (polimorphic class):

Յաճախ վիրտուալ ֆունկցիաները չեն որոշվում բազային դասում: Այդտեղ ընդգրկվում են միայն այդ ֆունկցիաների նախատիպերը: Սաքուր վիրտուալ ֆունկցիաների համար օգտագործվում է հետևյալ ընդհանուր ձև

Virtuaal տիպը ֆունկցիայի_անուն (պարամետրերի_gուցակ)=0:

Այս հայտարարման բանալիային մասը հանդիսանում է ֆունկցիայի հավասարեցումը 0-ի: Դա հաղորդում է կոնյակիլյատորին, որ բազային դասում գոյություն չունի ֆունկցիայի մարմին: Եթե ֆունկցիան տրվում է որպես մաքուր վիրտուալ ֆունկցիա, ենթադրվում է, որ այն պետք է վերառողվի յուրաքանչյուր ածանցյալ դասում: Եթե դա չկա, ապա կոնյակիլյատորին ժամանակ առաջանում է սխալ՝ Այսպիսով, մաքուր վիրտուալ ֆունկցիայի ստեղծումը ճանապարհ է, որը երաշխափորում է, որ ածանցյալ դասերը ապահովեն նրանց վերառողումը: Եթե դասը պարունակում է թեկուզ մեկ մաքուր վիրտուալ ֆունկցիա, կարող ենք խոսել արստրակտ դասի մասին (abstract class): Քանի որ արստրակտ դասում պարունակվում է ամենաքիչը մեկ ֆունկցիա, որի մոտ բացակայում է մարմինը, ապա տեխնիկապես այդպիսի դասը ամբողջական չէ, և այդ դասի ոչ մի օբյեկտ չի կարելի ստեղծել: Այսպիսով, արստրակտ դասերը կարող են լինել միայն ժառանգվող: Նրանք երբեք չեն լինուն առանձնացված: Կարևոր է հիշել սակայն, որ կարելի է ստեղծել

աբստրակտ դասի նշխներ, որոնց շնորհիվ ձեռք է բերվում դինամիկ պոլիմորֆիզմ:

Օրինակ՝ կառուցել նախագիծ, որը մուտքագրի ըստ ֆակուլտետների ուսանողների ցուցակները, իրականացնի ուսանողների թոշակների հաշվարկը և ելքագրի:

```
# include <iostream.h>
# include <fstream.h>
# include <string.h>
const int N = 20000;
Class mard
{
    // ուսանողների հատկությունները
    char anun [20]; // ուսանողի անուն
    char azganun [20]; // ուսանողի ազգանուն
    char hairanun [20]; // ուսանողի հայրանուն
    int her; // ուսանողի հեռախոս
    char dep [20]; // ուսանողի ֆակուլտետ
    char bajin [20]; // ուսանողի բաժին
    int kurs; // ուսանողի կուրս
    char tip[20];// ուսանողի տիպ (գերազանցիկ, նորմալ, անբավարար)
    int flag; // համեմատությունների համար փոփոխական
    Public: // դասում ներդրված ֆունկցիա, որը տալիս է ուսանողների
    // մասին տեղեկատվությունը ստեղծաշարից մուտքագրելու
    // միջոցով
    Void Set()
    {
        cout << "Enter Anun \ n";
        cin >> anun;
        cout << "Enter Azganun \ n";
        cin >> azganun;
        cout << "Enter Hairanun \ n";
        cin >> hairanun;
        cout << "Enter Her \ n";
        cin >> her;
        cout << "Enter Dep \ n";
        cin >> dep;
        cout << "Enter Bajin \ n";
        cin >> bajin;
```

```

cout << "Enter Kurs \n";
cin >> kurs;
cout << "Enter Tip \n";
cin >> tip;

flag = 1
}

// Get ֆունկցիան հերթականորեն էկրանին արտապատկերում է
ուսանողի մասին հատկություններն ու դրանց պարունակություն-
ները: Դրանց համար օգտագործվում է cout հոսքը:
void get()
{
    cout << "anun," << anun << "\n azganun," << azganun << "\n
    hairanun," << hairanun << "\n her," << her << "\n dep," <<
    dep << "\n bajin," << bajin << "\n kurs," << kurs << "\n tip," <<
    tip << "\n \n \n";
}

// Այս ֆունկցիան վերադարձնում է բույյան հաստատում, որի
արժեքը ճիշտ է, եթե նրան փոխանցված պարամետրը հավասար է
տվյալ ուսանողի հեռախոսահամարին:
bool PPP (int x)
{
    if (her == x) return true;
    else return false;
}

// Այս ֆունկցիան որպես պարամետր ստանում է սիմվոլների
հաջորդականություն, որը համեմատում է տվյալ ուսանողի անվան
հետ (strcmp ներդրված ֆունկցիայի միջոցով): Այն վերադարձնում է
ճիշտ, եթե պարամետրի արժեքը համապատասխանում է անվանը:
bool PPP (char nn [ ])
{
    if (strcmp (anun nn) == 0) return true;
    else return false;
}

// Ֆունկցիան դիտում է ուսանողի մասին տեղեկատվության
ներածվածության փաստը: Եթե ներածումը հանարվում է կատար-
ված, այն վերադարձնում է flag ֆունկտիվանի արժեքը, որը
հավասար է մեկի, այսպիսով միաժամանակ հայտնելով, թե տվյալ
ուսանողը «զրադիմ» այսինքն նրա մեջ չի կարելի նույրագրել
տեղեկատվություն ինֆորմացիայի կորստից խուսափելու համար:
int zanit ()
{

```

```

    return flag;
}

// Այս ֆունկցիան ազատում է տվյալ ուսանողին, այն flag-ի
արժեքը դարձնում է 0, այնպես որ, եթե zanit ֆունկցիայի միջոցով
կատարվում է հարցում տվյալ ուսանողի տեղի գբաղվածության
վերաբերյալ, ապա այն կվերադարձնի 0:
void azat ()
{
    flag=0;
}

// Այստեղ դիտվում է ուսանողի տիպը: Եթե նա գերազանցիկ է,
ապա ֆունկցիան վերադարձնում է 7000՝ գերազանցիկի կրթաթո-
շակ, եթե ուսանողը ստացել է երկու կամ նա ստվորում է ՎՃարովի
համակարգում, ապա վերադարձվում է 0՝ ուսանողը թոշակ չի
ստանում: Դակառակ դեպքում վերադարձվում է թոշակի նորմալ
չափը 5000 դր.:
int tosh ()
{
    if (strcmp (tip, "ger") == 0)
        return 7000;
    elseif (strcmp (tip, "nor") == 0)
        return 5000;
    else return 0;
}

Ուսանողի տիպը վերագրվում է նրան նկարագրող tip
հատկությանը (strcpy ֆունկցիայի միջոցով):
Void gnahat (char tt [ ])
{
    strcpy (tip, tt)
};

// Դայտարարվում է mard տիպի ուսանողների զանգված, որն
ունի m անունը և N էլեմենտների քանակը.
mard m [N];
Void mutq ()
{
    int i;
    char ch;
    for (i = 0; i < N; i++)
    {
        if (m [i].zanit ()) continue;
        m [i].set ();
        cout << "Duq uzum eq sharunakel ?";
    }
}

```

```

        cin >> ch;
        if (ch == 'n') break;
    }
}

```

Յարցման հնարավոր պատասխանները ոլիտվում են յ և ո
ստեղների սեղմումներով:

```

Void tpe1 ()
{
    int i;
    for (i = 0; i < N; i++)
    {
        if (!m [i].zanit ()) continue;
        m [i].get ();
    }
}

```

Այս ֆունկցիան կատարում է որոնում: Այն հարցնում է որոնման
տարրերակը անվան կամ հեռախոսահամարի միջոցով: Ա կամ Ի
ստեղներ:

```

int poisk ()
{
    int x, i = 0
    char ch;
    cout << "(A) nunov sharunakel, te (H)eraxosov ?";
    cin >> ch;
    if (ch == 'H')
    {
        cout << "Enter Tel";
        cin >> x;
        for (i = 0; i < N; i++)
        {
            if (m [i].PPP (x)) break;
        }
    }
    if (ch == 'A')
    {
        char nn [10];
        cout << "Enter Name";
        cin >> nn;
        for (i = 0; i < N; i++)
        {
            if (m [i].PPP (nn)) break;
        }
    }
    if (i < N)
    {
        m [i].get ();
    }
}

```

```

        return i;
    }
    else
    {
        cout << "chka";
        return -1;
    }
}

```

Կախված ընտրությունից՝ ուսանողի տվյալ հատկությունը համեմատվում է մուտքագրված որոնման պարամետրի հետ: Եթե այն համընկնում է, ցիկլը կանգ է առնում, և արտապատկերվում է տվյալ ուսանողի մասին տեղեկատվությունը, վերադարձվում է նրա ինդեքսը զանգվածում, հակառակ դեպքում ելքագրվում է, որ ուսանողը գտնված չէ և որպես ինդեքս վերադարձվում է -1:

Այն կատարում է փունկցիայի միջոցով, որից ստացված ինդեքսի -1 չլինելու դեպքում կատարում է հարցում ջնջել տվյալ ուսանողին, թե ոչ: Այս պատասխանի դեպքում գործում է նրա azat ֆունկցիան, և ելքագրվում է յոյեղ արտահայտությունը:

```

void jnj()
{
inti;
Charch;
i = poisk ();
if (i == -1) return;
Cout << "Jnjenoq \n";
}

```

// Գումար ֆունկցիան int տիպի ցած անունով փոփոխականի մեջ կուտակում է բոլոր այն ուսանողների կրթաթոշակների գումարները, որոնց մասին տեղեկատվությունը առկա է մասիվում: Գումարի հաշվումից հետո ելքագրվում է Lrv Toshak= և բոշակների գումարը:

```

void gumar ()
{
intgum = 0
for (int i = 0; i < N; itt)
{ if (! m[i].zanit ()) Continue;
gum = gum + m[i].tosh ();
}
cout << "Lrv Toshak=" << gum << endl;
}

```

Ֆունկցիան մուտքագրում է ուսանողի տիպը՝ ger-գերազանցիկ, ուշ-նորմալ, Lik-լիկվիդացիայի մասնակցող կամ սովորող վճարովի համակարգում։ Հարցման արդյունքը տեղափոխված է լինուիլականի մեջ։

```

void gnn ()
{ int l;
Char tt [S];
i = poisk ();
if (i == -1) return;
Cout << "Tip ?";
Cin >> tt;
if (!strcmp (tt, "ger") || !strcmp (tt, "nor") || !strcmp (tt, "Lik"))
m [i].gnahat (tt);
else
Cout << "error\n";
}
ofstream out ("arxiv.txt");
out.Write ((Char *) m, Size of (mard [n]));
out.close ();
}

```

Այս ֆունկցիան կոշտ սկավառակի վրա գտնվող arxiv.txt ֆայլից ուսանողների ու մասիկի մեջ բեռնում է ամբողջ ֆայլում հիշված ինֆորմացիան:

```

void Kardal ()
{ if stream in ("arxiv . txt");
in.read ((Char*) m, sizeof (mard [n]));
in.Close();
}
void main ()
{ int x ;
kardal ();
for ( ; )
{ Cout << " 1-mutg; 2-tpel; 3-
poisk; 4-jnj; 5-gumar; 6-
qrunutjun o- exit" << endl;
Cin >> x
if (x == 1) mutg ();
if (x == 2) tpel ();
if (x == 3) poisk ();
if (x == 4) jnj ();
if (x == 5) gumar ();
if (x == 6) gnn ();
if (x == 0) break;
}
}
qret ();

```

Գլուխ 4

Windows համակարգում աշխատող Microsoft Visual C++ -ի գործիքային միջոցները

4.1. Windows օպերացիոն համակարգում ծրագրավորումը

Windows օպերացիոն համակարգն օժտված է հետևյալ երեք հատկություններով, որոնք հատուկ հետաքրքրություն են ներկայացնում Windows հավելվածներ ծրագրավորելու համար:

- Գրաֆիկական ինտերֆեյս, որը շահագործողին ապահովում է հարմար և իրապուրիչ գրաֆիկական արտացոլում: Ծրագրավորողի համար ամենից առաջ դա նշանակում է գոյություն ունեցող ստանդարտի համապատասխան բարձր պահանջներ՝ ստեղծվող ծրագրերի նկատմամբ: Բայց մյուս կողմից հնարավորություն է տալիս ստեղծել ուղղակի փայլուն հավելված՝ համեմատարար ոչ մեծ ծախսերով:
- Windows հավելվածները կատարվում են սեփական պատուհանում: Դավելվածի պատուհանի միջոցով կատարվում է շահագործողի ինֆորմացիայի մուտքն ու ելքը: Դավելվածի գլխավոր պատուհանը շահագործողի տեսանկյունից նույնացվում է սեփական հավելվածի հետ, բայց ծրագրավորողի համար դա միայն վիզուալ ինտերֆեյս է:
- Windows-ի աշխատանքը կողմնորոշված է պատահարների վրա: DOS-ում ժամանակի ցանկացած պահին կարող է կատարվել միայն մեկ ծրագիր: Եթե ծրագիրը մեկ անգամ կանչվել է, ապա կատարվում է այնքան ժամանակ, քանի որ չի ավարտվել կամ ընդհատվել:
- Windows հավելվածը մեծ մասամբ կատարվում է առանձին հատվածներով: Եթե լուծել ենք առանձին ենթախնդիր, դեկավարումը վերադարձվում է Windows-ին, որը անհրաժեշտության դեպքում կարող է կանչվել այլ ծրագրերից: Այսպիսով, Windows-ը անջատվում և միացվում է տարբեր հավելվածների միջև: Շահագործողի կողմից որևէ մուտք առաջացնում է Windows-ի պատահարը: Պատահարը մշակվում է և ծրագրային դեկավարումը փոխանցվում համապատասխան հավելվածին:

Հավելվածի համար դա նշանակում է, որ այն կանչվում է պատահարի մշակման համար, համապատասխանաբար արձագանքում է մուտքին և այնուհետև նորից վերադարձվում է ոչ ակտիվ վիճակի այնքան ժամանակ, քանի դեռ նրա համար չի հայտնվում հաջորդ պատահարը:

4.2. Windows-ի միջավայրում մշակվող հավելվածների ինտեգրման մեթոդները

- Պրոցեսների ղեկավարում
- հաղորդումների մշակում
- հիշողության կառավարում
- բազմառաջադրանքայնություն և բազմահոսքայնություն

Պրոցեսների ղեկավարում

Windows 95, 98, 2000, XP, NT. համդիսանում են բազմառաջադրանք 32 կարգանի օպերացիոն համակարգեր, որոնք ներկայացնում են շահագործողներին ոչ միայն գրաֆիկական ինտերֆեյսի հարմարություններ, այլև ապահովում են մի քանի ծրագրերի միաժամանակյա կվազի- զուգահեռ կատարում:

Windows-ը ղեկավարում է ընթացիկ ծրագրերը շրջանաձև բուֆերի եղանակով: Շրջանաձև բուֆերը մտցված ծրագրերը հերթով Windows-ից ստանում են կենտրոնական պրոցեսորի ղեկավարում և իրենց կողերը կատարելու հնարավորություն:

Յուրաքանչյուր ծրագրի տրվում է ղեկավարման համար բավականին կարծ ժամանակահատված, որը տպավորություն է ստեղծում նրանց զուգահեռ կատարման վերաբերյալ: Այսինքն, աշխատում է բազմառաջադրանքային ռեժիմներ: Նա ոչ միայն բազմառաջադրանքային է, այլև բազմահոսքային: Վերջինս նշանակում է, որ առանձին ծրագրերի կատարումը կարող է կազմակերպվել մի քանի զուգահեռ կատարվող հոսքերի տեսքով:

Յուրաքանչյուր պրոցեսուրայի կանչի ժամանակ Windows-ը կազմակերպում է պրոցես 4 հետաքայթ հիշողության հասցեների տիրույթով: Առանձնացված հասցեների տիրույթ են բեռնավորվում պրոցեսուրայի կատարվող կողերը, և այդ կողերի սկզբնական հասցեն փոխանցվում է ծրագրի մուտքի ֆունկցիային (WinMain()) օբյեկտի դեսկրիպտորի տեսքով: Այնուհետև Windows-ը կազմա-

Կերպում է գլխավոր հոսքը, որը սկսվում է մուտքի ֆունկցիայի կատարումից:

Գլխավոր հոսքը իր կատարման ընթացքում կարող է ստեղծել այլ հոսքեր, որոնք կատարվում են ընդհանուր հասցեների տիրույթում:

Դաղորդումների մշակում

Windows-ը պահանջում է ժրագրավորում, որը կողմնորոշված է հաղորդումների վրա: Դաղորդումը ծառայում է Windows-ի և ընթացիկ հավելվածի միջև որպես կապի միջոց և կարևոր է եմենու բազմաառաջարանքային միջավայրի համար: Բայց հաղորդումը կարող է օգտագործվել նաև հավելվածները մեկը մյուսի հետ կապելու համար:

Windows-ը օգտագործում է հաղորդումների փոխանցման 2 եղանակ՝

- հաղորդումների մշակման ցիկլի օգտագործում (message loop)
- անմիջապես փոխանցում՝ փոխելով հաղորդումների մշակման ցիկլը:

Դաղորդման փոխանցումը՝ օգտագործելով մշակման ցիկլը

Դաղորդագրությունը, որը կանչել է շահագործողի գործողություններով, միշտ փոխանցվում է օգտագործելով հաղորդումների մշակման ցիկլը: Դաղորդումների մշակման ցիկլը իրենից ներկայացնում է while ցիկլ, որում հավելվածը ընդունում է Windows-ից հաղորդագրություն և դրանք բաշխում է հաղորդումների մշակման համապատասխան ֆունկցիաներին:

Դաղորդումների մշակման ցիկլը ստացվող հաղորդումների մշակման միջանկյալ հանգույց է: Նրա նշանակությունը կայանում է ստացվող հաղորդումների ժամանակագրական կարգով մշակման մեջ: Դա հատկապես կարևոր է շահագործողի մուտքի մշակման ժամանակ, օրինակ, եթե շահագործողը տեքստի մուտքի ժամանակ նշում է որոշակի պարագրաֆ և մտցնում նրա մեջ նոր բառ: Դաղորդումը, որը հավելվածին տեղեկացնում է շահագործողի կողմից առաջացրած պատահարի մասին, միշտ փոխանցվում է հաղորդումների մշակման ցիկլի միջոցով: Այն իրականացվում է հետևյալ քայլերով՝

- Առաջանում է պատահար, օրինակ՝ մկնիկի տեղաշարժման հետևանքով:
- Պատահարը գրանցվում է համակարգային հերթի մեջ: Դամակարգային հերթի մեջ են դրվում հաղորդումները բոլոր ծայրանասային սարքերից (մկնիկ, ստեղնաշար, տպող սարք և այլն), այնուհետև որոշվում է հավելվածը (կամ հոսքը), որին վերաբերում է ստացվող հաղորդագրությունը:
- Դամակարգային հերթից պատահարները, որոնք նշվել են նրանց ստացման համաձայն (FIFO սկզբունքով, առաջինը մտավ՝ առաջինը դուրս եկավ, ի տարբերություն FILO-ի առաջինը մտավ վերջինը դուրս եկավ, որը օգտագործվում է ենթաճրագրերի աշխատանքի ժամանակ), մտցվում է հերթի մեջ նրանց հոսքերին համապատասխան: Եթե օրինակ շահագործողը նշել է պատուհանը, ապա որոշվում է հոսք, որը ստեղծել է այդ պատուհանը, և հաղորդագրությունը գրանցվում է նրա հերթի մեջ:
- Դաղորդումը ընդունվում է հավելվածի հաղորդումների մշակման ցիկլում, անհրաժեշտության դեպքում ծևափոխվում է հեշտ կարդացվող պարամետրերի և փոխանցվում է հետագայում պատուհանի մշակող ֆունկցիային:
- Պատուհանի ֆունկցիան ստանում է հաղորդագրություն և համապատասխանաբար արձագանքում է նրան: Յուրաքանչյուր պատուհան որոշում է սեփական ֆունկցիան, որը ընդունում է ստացվող հաղորդագրությունը և իրականացնում է համապատասխան մշակում: Դրա համար պատուհանի յուրաքանչյուր ֆունկցիայում նախատեսված է switch օպերատոր, որի ճյուղերի օգնությամբ դրվում է, թե ինչ հաղորդում է ընդունվել, և կանչվում է ստացվող հաղորդագրության մշակման համապատասխան ֆունկցիան:

Դաղորդագրության անմիջական փոխանցումը

Մեծ քանակությամբ հաղորդումներ միայն անուղակի են կապված շահագործողի գործողություններին և ուղարկվում է Windows-ի «Ներսից» (օրինակ, եթե Windows-ը տեղեկացնում է պատուհանին նրա տեղաբաշխման վերականգնման կամ փակման մասին): Այդ դեպքերում պատահարները չեն տեղադրվում համակարգային հերթում, բացի որոշ բացառիկ դեպքերի, օրինակ WM_PAINT-ի օգտագործման ժամանակ, իսկ Windows-ը

ուղարկում է հաղորդագրություն անմիջապես համապատասխան պատուհանի ֆունկցիային: Windows-ը ճանաչում է ավելի քան երկու հարյուր տարբեր հաղորդագրություններ, որոնք սկսվում են “WM_” նախածանցով: Ծրագրավորողը պարտավոր չէ մշակել բոլոր այդ հաղորդումները: Բավական է ապահովել լույսայն սկզբունքով նախատեսված մշակումը, հաղորդագրությունների մեծամասնության համար: DefWindowProc() API ֆունկցիայի կանչելու ճանապարհով: Մյուս կողմից ծրագրավորողը պետք է ապահովի ծրագրի համար անհրաժեշտ հաղորդագրության զավթումն ու կապի նրանց մշակման համապատասխան ֆունկցիայի կանչի հետ: Դաղորդումների զավթման և մշակման համար ծրագրում պետք է որոշվի հաղորդումների մշակման ցիկլ և յուրաքանչյուր պատուհանի ֆունկցիան: Պատուհանի ֆունկցիան կանչվում է անմիջապես Windows օպերացիոն համակարգի կողմից, այսինքն նա հանդիսանում է callback ֆունկցիա: Պատուհանի ֆունկցիայի վերադարձվող արժեքի տիպը և նրա պարամետրերի ցուցակը խիստ է տրված՝ ի տարբերություն պատուհանի ֆունկցիայի անվան, որը կարող է ընտրվել կամայականորեն: Պատուհանի ֆունկցիայի անունը պետք է գրանցվի Windows-ում: API ծրագրերում դա տեղի է ունենում պատուհանի կառուցվածքի գրանցման ժամանակ: Բոլոր պատուհանները, որոնք այնուհետև ստեղծվում են տրված պատուհանային կառուցվածքի հիման վրա, կանչում են պատուհանի ֆունկցիան նշված անունով: Դաղորդման մշակման ցիկլը կատարվում է մշտապես և գործնականում հանդիսանում է ծրագրի արտաքին ցիկլ:

MFC-ծրագրերում հաղորդագրությունների «զավթումը»

МFC դասերի հետ աշխատանքի ժամանակ հաղորդագրությունների մշակումը կատարվում է շատ դեպքերում վերը նկարագրվածի նման:

1. MFC ծրագիրը տիրապետում է գլխավոր հոսքի համար հաղորդագրությունների մշակման ցիկլի: Բայց մենք չենք տեսնում այդ ցիկլը, ինչպես չենք տեսնում WinMain() մուտքի ֆունկցիան: Դրանք «քարենված են» սկզբնական գրադարանային մոդուլներում, որոնք ավտոմատ միացվում են MFC դասերին (mfc/src/winmain.cpp): AfxWinMain() ֆունկցիան կատարվում է հավելվածի cWinApp դասի օբյեկտի Run() մեթոդի կանչով: Այդ

մերողը իր հերթին կանչում է cWinThread դասի Run() մերող և նրա մեջ սխեմայով կազմակերպում է իր սեփական հաղորդումների մշակնան ցիկլը վերը նկարագրվածի նման:

Տ API ծրագրում անհրաժեշտ է որոշել և գրանցել յուրաքանչյուր պատուհանի համար հատուկ ֆունկցիա: MFC ծրագրերում պատուհանը ներկայացվում է ոչ թե կառուցվածքով, այլ պատուհանի դասերով. որոնք ծնված են cWnd դասից: Այդ ժամանակ անհրաժեշտ է տարբերել վերը նկարագրած հաղորդագրությունների մշակումը Windows-ի ներսում MFC գրադարանի դասերի օրյունական միջև հաղորդագրությունների փոխանցումից, որը հնարավոր է դարձնում պարզ եղանակի օգնությամբ կապ հաստատել մշակնան ֆունկցիայի և հաղորդագրության միջև ընդգրկելով պատահարի մշակնան ֆունկցիան հատուկ նակրոսի օգնությամբ հաղորդագրությունների բարտեցի (Message map) մեջ:

```
class CMyWnd: public CFrameWnd{
public:
    CMyWnd();
    afx_msg void OnLButtonDown(UINT Flags,CPOINT point);
    DDX_CINT MESSAGE_MAP()
};

BEGIN_MESSAGE_MAP(CmyWnd, CframeWnd)
ON_WM_LBUTTONDOWN()
END_MESSAGE_MAP()
```

Կարելի է կատարել հաղորդագրությունների մշակում տարբեր դասերում, ինչպես ծնված CWnd դասից, այնպես էլ ծնված CDocument և CWinApp դասերից:

Windows-ի հաղորդագրությունը միշտ հասցեագրված և պատուհանին: MFC ծրագիրը զարդարում է Windows-ի հաղորդագրությունը և ուղարկում է նրան ներկայացման (View) դասի սկզբին: Եթե այդ դասում ծրագրավորողը չի նախատեսել հաղորդագրության մշակնան ֆունկցիա, ապա հաղորդագրությունը կփոխանցվի փաստարդի դասին: Եթե մշակնան ֆունկցիա չգտնվի փաստարդի դասում, ապա հաղորդագրությունը կփոխանցվի գլխավոր պատուհանի դասին (MDI հավելվածներում դրստր պատուհանի դասի սկզբունք, իսկ այնուհետև, գլխավոր պատուհանի դասում) և, վերջապես, հավելվածի դասին: Դավելվածը կարող է ոչ միայն ընդունել հաղորդագրությունը, այլև ուղարկել նրանց, ինչպես նաև ակտիվացնել իր պատուհանը վերապատկերելով այն

Էկրանի վրա, իրականացնել Windows-ի ղեկավարման էլեմենտների գործունեության եղանակի փոփոխում, կատարել ինֆորմացիայի փոխանակում այլ հավելվածների հետ:

Դաղորդագրության մշակման ցիկլի միջոցով հաղորդագրության փոխանցման համար օգտագործվում է PostMessage() ֆունկցիան, որը տեղադրում է ուղարկվող հաղորդագրությունը համապատասխան հերթի մեջ և վերադարձնում է ղեկավարումը կանչող ֆունկցիային: Սակայն ղեկավարման վերադարձը չի նշանակում, որ ուղարկվող հաղորդագրությունը արդեն մշակել է ընդունող հավելվածը: CWnd դասի PostMessage() մեթոդը որպես արգումենտ պահանջում է կող և հաղորդագրության պարամետրեր:

```
BOOL PostMessage(UINT message, WPARAM wParam=0, LPARAM lParam=0);
```

Տվյալ մեթոդի կատարման արդյունքում Message հաղորդագրությունը տեղադրվում է հաղորդագրությունների մշակման ցիկլում, որը պատկանում է CWind դասի օբյեկտին, և որը կանչում է մեթոդը:

Եթե պահանջվում է այլ պատուհան ուղարկել հաղորդագրությունը, ապա օգտագործում են postMessage()-ի API ֆունկցիա:

```
BOOL PostMessage(HWND hwnd, UINT message, WPARAM wParam=0, LPARAM lParam=0);
```

Տվյալ ֆունկցիան ունի լրացուցիչ պարամետր, որը թույլ է տալիս նշել դեսկրիպտոր, և որը հասկանում է պատուհանը: Կարելի է հաղորդագրությունը ուղարկել անմիջապես հոսքին, և ոչ թե պատուհանին:

```
BOOL PostThreadMessage(DWORD idThread, HWND hwnd, UINT message, WPARAM wParam=0, LPARAM lParam=0);
```

Դաղորդագրությունը անմիջապես պատուհանին ուղարկելու համար, փոխելով հաղորդագրությունների հերթը, օգտագործվում է SendMessag() ֆունկցիա: Այդ ֆունկցիան ուղարկում է հաղորդագրությունը անմիջապես պատուհանի ֆունկցիային: Ի տարբերություն PostMessage() ֆունկցիայի՝ SendMessag() ֆունկցիան սպասում է իր կողմից փոխանցված հաղորդագրության մշակմանը: Այսինքն՝ հոսքը, որը ուղարկել է հաղորդագրությունը, կատարվելու է հետո, եթե վերջին հաղորդագրությունը կմշակվի հասցեատիրոջ հոսքում: Այլ խոսքով, SendMessage() ֆունկցիան կատարում է հաղորդագրությունների սինխրոնիզացված փոխանակում: CWnd

դասի SendMessage() մեթոդը որպես արգումենտ պահանջում է հաղորդագրության կոդ և պարամետրեր:

LRESULT SendMessage(UNIT message, WPARAM wParam=0, LPARAM lParam=0);

Տվյալ մեթոդի կատարման արդյունքում message հաղորդագրությունը ուղարկվում է անմիջապես պատուհանի ֆունկցիային, որը պատկանում է cWnd դասի օբյեկտին, որն էլ կանչել է տվյալ մեթոդը: Եթե պահանջվում է հաղորդագրությունը ուղարկել այլ պատուհանի, ապա օգտագործվում է SendMessage()-ի API ֆունկցիա:

LRESULT SendMessage(HWND hwnd, UNIT message, WPARAM wParam=0, LPARAM lParam=0);

Տվյալ ֆունկցիան ունի լրացուցիչ պարամետր, որը թույլ է տալիս նշել ընդունող պատուհանի դեսկրիպտոր:

Դաղորդագրությունները այլ հավելվածներ ուղարկելու համար անհրաժեշտ է նշել պատուհանի դեսկրիպտոր: Նրա մեծությունը կարելի է ստանալ, օրինակ HWND FindWindowEx(HWND hwndParent, HWND hwndChildAfter, LPCTSTR LpszClass, LPCTSTR LpszWindow):

Ակտիվ պատուհանների մասին ինֆորմացիա ստանալու համար կարելի է օգտվել GetNextWindow() և GetWindowText() API ֆունկցիաններից:

Դաղորդագրությունների մշակումը

Windows-ի API ֆունկցիաների հետ աշխատելիս հավելվածի յուրաքանչյուր պատուհանի համար հաղորդագրությունների մշակումը կանչում է Windows-ի համապատասխան callback ֆունկցիան (պատուհանի ֆունկցիա): Այդ ֆունկցիան սովորաբար բաղկացած է switch օպերատորից, որը ընտրում է ստացվող հաղորդագրությունը և կանչում է նրանցից յուրաքանչյուրի համար մշակման համապատասխան ֆունկցիա:

MFC-ում պատուհանի ֆունկցիայի դերում հանդես է գալիս հաղորդագրությունների քարտեզը (Message Map), որը թույլ է տալիս հաղորդագրությունների մշակումը որոշակի դասերում: Այդ ժամանակ Windows-ը նախկինի նման ուղարկում է հաղորդագրությունը պատուհանի ֆունկցիաներին, որը «բաքնված է» MFC կողում, իսկ MFC-ն վերառուղրությունը է հաղորդագրությունը հաղորդագրության քարտեզին: MFC ծրագրերի հաղորդագրությունների գավթման և մշակման համար, որի հիմքը գեներացվում է

հավելվածի վարպետի կողմից, կարող ենք օգտվել հետևյալ եղանակներով:

- Հաղորդագրությունների մեջամասնության մշակման համար անհրաժեշտ է ընդգրկել հաղորդագրությունների քարտեզի որոշակի մակրոսներ: Այդ դեպքում նախապես պետք է պարզել հաղորդագրության համար տրված է արդյոք սեփական հաստատուն (օրինակ՝ WM_LBUTTONDOWN_DOWN հաստատունը համապատասխանում է պատուհանի աշխատանքային տիրութում մկնիկի ծախս սեղմակը սեղմելու մասին հաղորդագրությանը):

Օգտագործվում է հաստատուն, որը ընդիանուր է որոշակի կատեգորիայի հաղորդագրությունների համար (օրինակ՝ WM_COMMAND մեջում իրամանի ընտրման մասին, գործիքների գոտու սեղմակի և ստեղների համադրման մասին հաղորդագրության համար)

- WM_PAINT հաղորդագրության մշակման համար նախատեսված է առանձին մեխանիզմ
- Կարելի է որոշել սեփական հաղորդագրություն:

Հաղորդագրությունների քարտեզի ծևավորումը

Windows հավելվածը աշխատանքի ժամանակ ստանում է հաղորդագրությունների հոսք, որը տեղեկացնում է կատարվող պրոցեսների մասին: Եթե, օրինակ, շահագործողը անցնում է մի ծրագրից մյուսին, ապա ակտիվացվող հավելվածի գլխավոր պատուհանը ստանում է հաղորդագրություն WM_ACTIVATE (հնարավոր է տարբեր պարամետրերով), որը տեղեկացնում է հավելվածին նրա ակտիվացման մասին: Եթե շահագործողը սեղմել է պատուհանի աշխատանքային տիրություն մկնիկի ծախս սեղմակը, ապա պատուհանը ստանում է WM_LBUTTONDOWN հաղորդագրություն: Եթե հավելվածը պետք է արձագանքի այդպիսի հաղորդագրությանը, պետք է կազմակերպել տվյալ հաղորդագրությունը մշակող համապատասխան ֆունկցիա:

Զեռքով հաղորդագրությունների մշակման կազմակերպումը

1. Պետք է որոշել, թե ինչպիսի հաղորդագրություն ենք ցանկանում մշակել և ինչպիսի հաստատուն է նրան համապատասխանում: Դրա համար օգտվում ենք MSDN գրադարանից

(Նայում ենք այն թեմաները, որոնք սկսվում են WM_ նախածանցով) կամ SPY++ ծրագրերից:

2. Զևակերպում ենք հաղորդագրությունների քարտեզը: Եթե դասը, որի մեջ պետք է կատարվի հաղորդագրությունների մշակումը, չի տիրապետում հաղորդագրությունների քարտեզին, ձևակերպում ենք այն այսպես:

I. Ընդգրկում ենք DECLARE_MESSAGE_MAP մակրոս դասի հայտարարման վերջում (ենթադրենք պատուհանի դասի հայտարարման մեջ):

II. Ավելացնում ենք դասի սկզբնական ֆայլում հաղորդագրության քարտեզը: Քարտեզը ձևավորվում է BEGIN_MESSAGE_MAP() և END_MESSAGE_MAP() մակրոսների օգնությամբ, ընդ որում՝ առաջին մակրոսին անհրաժեշտ է որպես պարամետր փոխանցել դասի անունը և անմիջական բազային դասը:

3. Տեղադրում ենք քարտեզի մեջ հաղորդագրությունների մշակման գրառումը: MFC-ում նախատեսված են մի շարք մակրոսներ տարբեր հաղորդագրությունների համար, որոնք ավտոմատ ձևավորում են հաղորդագրությունների կապը մշակող մեթոդի հետ: Այդ ժամանակ անհրաժեշտ է հիշել, որ յուրաքանչյուր համակարգային հաղորդագրության համար, որը սկսվում է WM_ նախածանցով, MFC-ում տրված են մակրոսի և մշակող մեթոդի անունները:

Հաղորդագրության քարտեզի մակրոսի անունը ձևավորվում է հաղորդագրության հաստատումի անունից ավելացնելով ON_ նախածանց (օրինակ WM_LBUTTONDOWN_DOWN հաղորդագրության համար հաղորդագրության քարտեզի մակրոսի անունը կլինի ON_WM_LBUTTONDOWN()): Սեթող_մշակողի անունը ստացվում է մակրոսի անունից հեռացնելով բոլոր ընդգծման նշանները և WM_ նախածանցը թողնելով մեծատառ յուրաքանչյուր բարի առաջին տառերը (օրինակ OnLButtonDown()): Սեթող_մշակողի պարամետրերի ցուցակը կարելի է իմանալ օգտագործելով MSDN գրադարանը, կատարելով որոնում մակրոսի անունով:

4. Վերնագրային ֆայլ ունի հետևյալ տեսքը:

```
Class CMyFrameWnd : Public CFrameWnd{  
Public:
```

```
CMyFrameWnd();  
DECLARE_MESSAGE_MAP();  
};
```

```
5. Ակզենական տեքստի ֆայլը ունի հետևյալ տեսքը.  
BEGIN_MESSAGE_MAP(CMyFrameWnd, CFrameWnd)  
ON_WM_LBUTTONDOWN()  
END_MESSAGE_MAP();
```

Դիշողության դեկավարումը

32 կարգանի հասցեավորման ժամանակ կորչում է բաղադրյալ հասցեավորման անհրաժեշտությունը: Դրա փոխարեն անմիջապես օգտագործվում է լայն տարածված 32 կարգանի ճարտարապետության առավելությունը, որը թույլ է տալիս կատարել 4 հեգաբայր հիշողության հասցեավորումը: Win32-ում յուրաքանչյուր պրոցես տիրապետում է իր սեփական հասցեների տիրույթներին, այսինքն գործում է ամբողջ 4 հեգաբայր վիրտուալ հիշողությունը: Պրոցեսի հոսքերը բաժանում են իրար միջև այդ հասցեների տիրույթը: Յուրաքանչյուր 32 կարգանի պրոցեսի սեփական հիշողությունը պարունակվում է եջի արտացոլման աղյուսակներում (page mapping tables): Պրոցեսից դրւու հասցեների փոխանցումը հաճախ բերում է սխալների, ուստի ավելի ծիշտ է կատարել փոխանակում դեսկրիպտորներով, որոնք հանդիսանում են գլոբալ և եզակի Windows-ի սահմաններում (օրինակ՝ պատուհանի դեսկրիպտորներ): Գլոբալ են բոլոր դեսկրիպտորները, որոնք վերաբերում են շահագործողի ինտերֆեյսի էլեմենտներին (պատուհան, մենյու, մկնիկի նշիչ և այլն): Լոկալները գործում են միայն սեփական պրոցեսի շրջանակներում և հանդիսանում են միջուկի օբյեկտների դեսկրիպտորներ (հոսքեր, ֆայլերի իդենտիֆիկատորներ և ուղիները և այլն):

○ Բազմահոսքայնություն և բազմառաջադրանքայնություն Win16-ում կատարվող կողմ անվանում են առաջադրանք (task):

Քանի որ Windows 3.x-ում ծրագիրը կարելի է կանչել բազմաթիվ անգամ, ուստի ծրագիրը և առաջադրանքը նույնը չեն: Հետևաբար խոսքը ծրագրի օրինակների մասին է, և յուրաքանչյուր այդպիսի օրինակին համապատասխանում է առաջադրանք: Win32-ում խոսում են պրոցեսների և հոսքերի մասին: Ծրագրի յուրաքանչյուր օրինակ համապատասխանում է միայն մեկ պրոցեսի, և յուրաքանչյուր պրոցես ավտոմատ տնօրինում է հոսքի,

որը նշանակում է սեփական կատարման ենթակա գործողություն-ների շղթա: Win32-ում հաղորդագրությունը ուղարկվում է հոսքով և հոսքերը բաժանում են իրար միջև պրոցեսորի ժամանակը: Չկա եական տարրերություն հոսքերի և առաջադրանքների միջև, բայց հոսքերն ունեն այն առավելությունը, որ կարող են իրենք ստեղծել նոր հոսքեր, ընդ որում ստեղծված և ստեղծվող հոսքերը պատկանում են նույն պրոցեսին: Քանի որ բոլոր ստեղծված հոսքերը մասնակցում են բազմառաջադրանքայնությանը, ապա հավելվածը հնարավորություն է ստանում առանձնացնելու ժրագրերի ժամանակի մեջ ծախսեր պահանջող (օրինակ տերստը տպելը) առանձին հոսք, որպեսզի ինքը հավելվածը, նրա գլխավոր հոսքը երկար կարողանա կատարվել տպելու ժամանակ: Լրացուցիչ հոսքին նրա ստեղծման ժամանակ կարելի է նշանակել առավելություն, որպեսզի այդպիսով ռացիոնալ ձևով դեկավարի պրոցեսորային ժամանակը:

Բազմառաջադրանքայնությունը դեռ հայտնի է Windows 3.x-ից: Սակայն այնտեղ խոսքը գնում է համատեղ չսեղմվող (փոխարինվող) բազմառաջադրանքայնության մասին, որի ժամանակ Windows-ը միաժամանակ դեկավարում է մի քանի բաց հավելվածներ: Windows-ը հսկում է շահագործողի մուտքը և փոխանցում է նրան հերթով որպես հաղորդագրության համապատասխան հավելվածներ: Դավելվածը ստանում է մշակման համար պրոցեսորի դեկավարման հաղորդագրություն և վերադարձնում է նրան, եթե նրանց առաջադրանքը կատարվել է: Դավելվածի կատարման ժամանակ, քանի դեռ չի մտել հաղորդագրությունների մշակման ցիկլ, Windows-ը չի կարող «առանձնացնել» նրան պրոցեսորի դեկավարումից: Վատագույն դեպքում հավելվածը մտնում է անվերջ ցիկլի մեջ կամ էլ «կախվում է», որի արդյունքում կաթվածահար է անում ամբողջ համակարգը:

Win32-ում հոսքերի դեկավարումը իրականացվում է այնպես, որ յուրաքանչյուր հոսքի համար առանձնացվում է ժամանակի հատված, որի ընթացքում հսկում է պրոցեսորին: Եթե պրոցեսը դրա հետ որոշում է բոլոր ընթացիկ հոսքերի համար պրոցեսորի դեկավարման հերթականությունը, ապա ստեղծվում է մի քանի ժրագրերի գուգահեռ մշակման տպավորություն: Իհարկե դա ֆունկցիավորման պարզեցված մոդելն է: Windows-ը ճշտում է նորելը հետևյալ եղանակներով՝

- Հոսքին տրվում է առավելություն (պրիորիտետ)

- Օպերացիոն համակարգը կարող է դիմամիկ ձևով փոխել առավելությունները:
- Չհամագործակցող հոսքերը անցնում են կատարման ֆոնային ռեժիմ:

Յուրաքանչյուր նոր ստեղծվող պրոցեսին օպերացիոն համակարգը նշանակում է առավելության դաս ցածր, նորմալ, բարձր և իրական ժամանակի ռեժիմ:

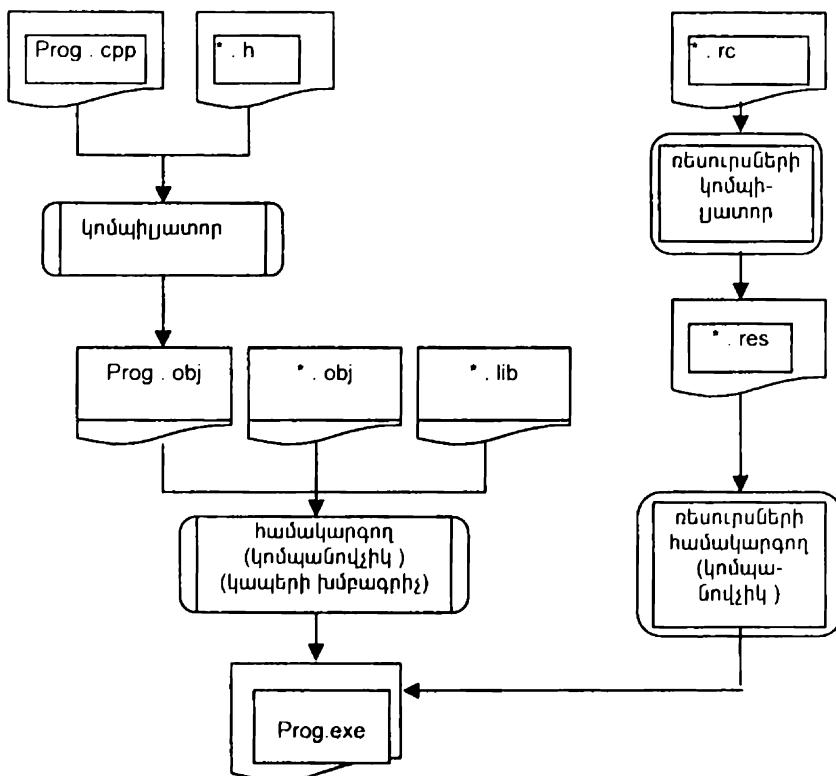
Ավելի բարձր առավելությամբ պրոցեսներն առավելություն ունեն ավելի ցածր առավելությամբ պրոցեսների նկատմամբ: Բայց դա ամենահին չի նշանակում, որ դրանց համար ցանկացած ժամանակ կառանձնացվի ավելի պրոցեսորային ժամանակ: Ավելի բարձր առավելությամբ պրոցեսները , որոնք տվյալ պահին չեն կատարվում, անցնում են ֆոնային ռեժիմ: Բայց եթե այդ պրոցեսը ստանում է մշակման ենթակա հաղորդագրություն, Windows-ը անմիջապես հեռացնում է պրոցեսորից ընթացիկ կատարվող պրոցեսը, որն ունի ավելի ցածր առավելություն և փոխանցում է պրոցեսորային ժամանակը ավելի բարձր առավելությամբ պրոցեսին: Նորմալ առավելությունը այն առավելությունն է, որը նշանակում է լրեւյան Windows-ում ստեղծվող բոլոր պրոցեսների համար: Եթե միաժամանակ կատարվում են նորմալ առավելությամբ մի քանի պրոցեսներ, ապա այն պրոցեսը, որի հետ աշխատում է շահագործողը, անմիջապես նշվում է որպես ակտիվ, և դրա առավելությունը բարձրացվում է: Յուրաքանչյուր ծրագրում կարող է նշանակվել առավելություն՝ SetPriorityClass() API ֆունկցիայի կանչի օգնությամբ: Հոսքերը՝ ստեղծման ժամանակ ստանում են իրենց պրոցեսների առավելություններին համապատասխան առավելություններ: Հոսքի առավելությունը կարող է փոփոխվել օպերացիոն համակարգի կողմից կամ էլ SetThreadPriority() API ֆունկցիայի կանչով: Օպերացիոն համակարգ բազմառաջադրաբարյան ռեժիմում կարող է ընդիատել ցանկացած պրոցեսի կատարումը՝ կատարման ցանկացած փուլում:

4.3. Ծրագրերի մշակումը Visual C++ -ում

MFC (Microsoft Foundation Classes) իրենից ներկայացնում է դասերի գրադարան, որը նախատեսված է Windows-API ֆունկցիաների համար: MFC գրադարանը համակարգում է API ֆունկցիաները, Windows-ի տվյալների տիպերը և կառուցվածքները որոշելով տարբեր էլեմենտները, որոնցից բաղկացած է Windows ծրագիրը որպես օբյեկտներ և կազմակերպում է նրանց համար դասեր, որոնցում հավաքված են օբյեկտների մշակման համար բոլոր անհրաժեշտ տվյալները և մեթոդները:

Նշենք ծրագրերի մշակման ժամանակ գործողությունների հաջորդականությունը

- նախագծի կազմակերպում
- ընթացիկ տեքստի մուտք
- կոմպիլյացիա և համակարգում (կոմպանովկա)
- տեստավորում և կարգաբերում



Սկզբնական ֆայլը բացելու համար ակտիվացնում ենք պրոյեկտի պատուհանի File View ներդիրը, բացում ենք սկզբնական ֆայլի հանգույցը (source Files) և կատարում մեզ հետաքրքրող ֆայլի վրա կրկնակի սեղմում: Դաջորդ քայլը կոնֆիւատորի կանչն է սկզբնական տեքստի օբյեկտային կոդի ծևափոխման համար: Դրա համար կանչում ենք Build հրամանը Compile Ֆայլի_անուն, կամ սեղմում Build գործիքների գոտու վրա compile սեղմակը: Կոնֆիւատորից հետո պետք է կանչել կապերի խմբագրիչը՝ կատարվող .exe ֆայլերի ստեղծման համար:

Մեծ ծրագրերը հաճախ բաղկացած են մի քանի մոդուլներից, այսինքն՝ սկզբնական տեքստերի ֆայլերից, որոնցից յուրաքանչյուրը բարգմանվում է օբյեկտային կոդի առանձին ֆայլի: (#include): Կապերի խմբագրիչը կազմում է տարբեր մոդուլներից բաղկացած կատարվող (.exe) ֆայլ: Պրոյեկտի բոլոր ֆայլերի կոնֆիւացիայի և բեռնավորվող մոդուլի մեջ նրանց համակարգման ռամար կանչում ենք Build/Build բեռնավորվող_ֆայլի_անուն:

- File / New / Projects
- Տալիս ենք պրոյեկտին անուն, նշում ենք պրոյեկտի կատարողը և պրոյեկտի տիպը (օրինակ՝ Win32 console Application): Ընտրում ենք Create new Workspace սեղմակ-փոխանքատիչը, ոնում ենք նշիչը Win32-ի վրա և սեղմում OK սեղմակը:
- Եվրանին հայտնվում է կրնակային հավելվածի վարպետը: Ընտրում ենք նրա մեջ An Empty project (դատարկ պրոյեկտ) և սեղմում Finish սեղմակը:
- Ստեղծում ենք սկզբնանան ֆայլ: Ընտրում ենք File/New/Files: File Name դաշտում նշում ենք ֆայլի անունը, ընտրում ֆայլերի լիպերի ցուցակից C++ Source File (C++ սկզբնական ֆայլ) և OK: Եկրանին հայտնվում է խմբագրման պատուհան՝ բաց դատարկ ֆայլով: Մտցնում ենք ֆայլում սկզբնական տեքստը և պահպանում այն՝ սեղմելով [ctrl] + [s]

```
# include <stdio.h>
void main ()
{
    Printf(" hello World!"\n ");
}
```

- Կոնֆիւացիա և համակարգում: Build/Build բեռնավորվող_ֆայլի_անուն
- Եթե կոնֆիւատորը հայտնաբերել է սխալներ, ապա օբյեկտային կոդը չի ծևափորվում և կապերի խմբագրիչը չի կանչվում:
- Կատարում ենք ծրագիրը (Build/Execute բեռնավորվող_ֆայլի_անունը):

4.4. Նախագծերի ստեղծման վարպետները

Visual C++-ում ծրագրային ֆայլերի դեկավարումը իրականացվում է նախագծի շրջանակներում: Յետևաբար ծրագրերի մշակումը սկսվում է նախագծի կազմակերպումով:

- ATL COM App Wizard (ATL COM հավելվածների վարպետ): Սա թույլ է տալիս ստեղծել ATL պրոյեկտ (Active Template Library), որի մեջ այնուհետև կարելի է ավելացնել com օբյեկտներ:
- Cluster Resource Type Wizard (ռեսուրսների հավաքածուի վարպետ): Սա գեներացնում է 2 տիպի պրոյեկտ Microsoft Cluster Server (MSCS) ռեսուրսների ծեավորման համար: Հահագործողական ռեսուրսների ստեղծումը հեշտացնում է հավելվածի համար MSCS-ի կողմից դեկավարումը և հսկումը: Resourse DLL պրոյեկտը նախատեսված է DLL ստեղծման համար, որը բեռնավորվում է MSCS ռեսուրսների մոնիթորով հավելվածի դեկավարման համար Cluster Administration Extension DLL նախատեսված է COM սերվերի ծեավորման համար, որը ապահովում է շահագործուղին ինտերֆեյսով նոր տիպի ռեսուրսների դեկավարման համար
- Custom AppWizard (հավելվածի կարգավորող վարպետ) Նախատեսված է հավելվածի սեփական վարպետի մշակման համար
- Database Project (տվյալների բազայի պրոյեկտ): Տվյալների բազայի պրոյեկտը տվյալների սիսեմաների հավաքածու է, այսինքն տվյալների բազա և ինֆորմացիա, որը անհրամեշտ է նրան դիմելու համար: Տվյալների բազայի նախագծի ստեղծման ժամանակ կարելի է տեղադրել միացումներ մեկ կամ ավելի տվյալների բազայի ODBC միջոցով և դիտել դրանց բաղկացուցիչները շահագործողի ինտերֆեյսի օգնությամբ, որը ընդգրկում է Database Designer (տվյալների բազայի մշակող)- տվյալների բազայի նախագծման և ստեղծման համար և Query Designer (հարցումների մշակող) ցանկացած ODBC համատեղելի տվյալների բազայի համար SQL օպերատորների համար:
- DevStudio Add-in Wizard (լրակառույցների վարպետ): Ստեղծում է լրակառույցի պրոյեկտի կմախքը (կարկաս), որը նախատեսված է Developer Studio ֆունկցիաների ընդլայնման համար:

- Extended Stored Proc Wizard (Արտաքին պահպանվող պրոցեդուրաների վարպետ): Նախատեսված է SQL սերվերի արտաքին պահպանվող պրոցեդուրաների ստեղծման համար: Արտաքին պահպանվող պրոցեդուրան իրենից ներկայացնում է extenC ֆունկիա, որը արտահանվում է Win32.dll -ից, որը կարող է կանչվել SQL սերվերից:
- ISAPI Extension Wizard (ISAPI ընդլայնման վարպետ): Պաշտպանում է API ընդլայնում և մշակում ինտերնետ սերվերի համար: Microsoft Internet Information Server (WinNT) կամ Microsoft personalWeb Server (դրվում է Microsoft Front Page-ի հետ) և նաև ֆիլտրեր, որոնք օգտագործվում են տվյալների հաղորդման ժամանակ:
- Make File: Պրոյեկտի տիպ է, որը կիրառվում է հրամանների ֆայլի հավելվածի ստեղծման ժամանակ: Հավելվածի արդյունքում ստացվածը կարելի է կանչել մշակման ինտեգրված միջավայրից:
- MFC ActiveX Control Wizard (Activx դեկավարման էլեմենտի վարպետ): Նախատեսված է ActiveX դեկավարման էլեմենտների մշակման համար MFC (Microsoft Foundation Classes) դասերի գրադարանի հիման վրա:
- MFC AppWizard (dll) (dll հավելվածի վարպետ): Ստեղծում է դինամիկ բեռնավորվող գրադարանի պրոյեկտի կմախը (կարկաս) MFC դասերի գրադարանի հիման վրա:
- MFC AppWizard (exe) (exe հավելվածի վարպետ): Ստեղծում է MFC դասերի գրադարանի հիման վրա Windows հավելվածի պրոյեկտի կմախը:
- New Database Wizard (տվյալների բազայի վարպետ): Գոյություն ունեցող SQL սերվերի բազայի հիման վրա ստեղծում է նոր տվյալների բազայի պրոյեկտի կմախը (կարկաս):
- Utility Project (սերվիսային պրոյեկտ): Սերվիսային պրոյեկտի ստեղծման ժամանակ նրա մեջ չի ավելացվում ֆայլեր: Սերվիսային պրոյեկտը չի գեներացնում LIB, DLL կամ EXE տիպի ելքային ֆայլեր: Տվյալ տիպի պրոյեկտը օգտագործվում է որպես բեռնարկղ (կոնտեյներ) ֆայլերի համար, որոնք կարող են կառուցվել առանց կապերի փուլի:
- Win32 Application (Win32 հավելված): Պրոյեկտի տիպը նախատեսված է ինչպես պրոցեդուրային այնպես էլ օբյեկտակողմնորոշված Win32 հավելվածի մշակման համար (հավելվածի կմախքը այդ ժամանակ չի ստեղծվում):

- Win32 console Application (Win32 կոնսոլային հավելված): Պրոյեկտի տիպը թույլ է տալիս ստեղծել հավելված, որը աշխատում է տեքստային պատուհանում (Ms Dos – ի կոնսոլային պատուհանում): Բայց պետք է հաշվի առնել, որ դա կլինի Win32 կոնսոլային հավելված, այսինքն հավելված, որը աշխատում է Ms Dos-ի պատուհանում 32 կարգանի օպերացիոն համակարգում, բայց ոչ Ms Dos օպերացիոն համակարգում: Տվյալ տիպի պրոյեկտի հավելվածի կմախք չի ստեղծվում:
- Win32 Dinamic Link Library (Win32 դինամիկ գրադարան): Պրոյեկտի տիպը նախատեսված է դինամիկ գրադարանների մշակման համար (հավելվածի կմախք չի ստեղծվում):
- Win32 Static Library (Win32 ստատիկ գրադարան): Պրոյեկտի տիպը նախատեսված է ստատիկ գրադարանների մշակման համար (հավելվածի կմախք չի ստեղծվում):

Հավելվածների ստեղծման վարպետների օգտագործումը

Visual Studio ծրագրավորման միջավայրը օգտագործվում է ոչ միայն խմբագրման, կոմպիլյացիայի և ծրագրերի կարգավորման համար, այլև նրանց նախապատրաստման, ստեղծման համար:

Հավելվածի վարպետը թույլ է տալիս ստեղծել այդպիսի նախապատրաստում մի քանի րոպեի ընթացքում, սակայն դրա հիման վրա ամբողջական կոմերցիոն հավելված ստանալը կտևի ոչ միայն շաբաթներ, այլև ամիսներ: MFC գրադարանը Visual C++-ի հիմնական գրադարաններից է և թույլ է տալիս ստեղծել 4 տեսակի հավելված:

- Երկխոսության հավելված
- Միապատուհան հավելված
- Բազմապատուհան հավելված
- Վերին ճակարողակի մի քանի փաստաթղթերով հավելված:

Երկխոսային հավելվածը Microsoft-ի կողմից դիտվում է որպես հավելվածի պարզագույն տիպ, որը չպետք է ունենա ոչ մի մենյու, բացի համակարգայինից, ինչպես նաև չի կարող բացել կամ պահպանել ինֆորմացիան ֆայլում:

Երկխոսության հավելվածը հանդիսանում է Windows-ի ամբողջական հավելված, որի մեջ կարող է օգտագործել ActiveX տեխնոլոգիայի ամբողջական սատարում, որը թույլ է տալիս լուծել տվյալների ուղարկումը և ստացումը հավելվածից:

MFC հավելվածների ստեղծման վարպետի օգնությամբ ինքնուրույն երկխոսության հավելվածի նախապատրաստում, պետք է կատարել հետևյալ քայլերի հաջորդականությունը.

1. Փակում ենք բոլոր բաց պրոյեկտները և ֆայլերը, ընտրում File/New/Project հրամանը կամ սեղմում New Project սեղմակը Standard գործիքների գոտու վրա: Արդյունքում կհայտնվի New Project երկխոսության պատուհան:
2. Project Types հիերարխիկ ցուցակի պատուհանում բացում ենք Visual C++ Projects ծրարը (թղթապանակը):
3. Templates (շարլոն-նախատիպ) ցուցակի պատուհանում առանձնացնում ենք MFC Application նշանը և Name տեքստային դաշտում մտցնում ենք հավելվածի անունը "Dialog" և սեղմում OK սեղմակը: Կհայտնվի MFC Application Wizard-Dialog երկխոսության պատուհանը:
4. Բացում ենք Application Type ներդիրը: Application Type փոխանցատիչը դնում ենք Dialog Based դիրքի վրա և սեղմում Finish:

Windows –ի բազմապատուհան հավելված

Windows –ի բազմապատուհան հավելվածը օգտագործում է MDI (Multiple Document Interface), որը և հանդիսանում է Windows –ում հավելվածի հիմնական տեսակը: Ինքնուրույն Windows –ի բազմապատուհան հավելված ստեղծելու համար, օգտագործելով MFC Application Wizard, պետք է.

1. Փակել բոլոր բաց պրոյեկտներն ու ֆայլերը և ընտրել File/New/Project հրամանը կամ Standard գործիքների գոտու վրա սեղմել New/Project կոճակը:
2. Project Types հիերարխիկ ցուցակի պատուհանում բացում ենք Visual C++ Projects ծրարը (թղթապանակը):
3. Templates (շարլոն-նախատիպ) ցուցակի պատուհանում առանձնացնում ենք MFC Application նշանը և Name տեքստային դաշտում մտցնում ենք հավելվածի անունը "MDI" և սեղմում OK սեղմակը: Կհայտնվի MFC Application Wizard երկխոսության պատուհանը:
4. Բացում ենք Application Type ներդիրը: Application Type փոխանցատիչը դնում ենք Multiple Documents դիրքի վրա և սեղմում Finish:

5. Մեկնարկում ենք հավելվածը Debug/Start-ով (F5) և հայտնվող Երկխոսության պատուհանին պատասխանում Yes:

MFC հավելվածների մյուս տեսակները

MFC Application Wizard Երկխոսության պատուհանում Application Type ներդիրում կարող ենք ընտրել նաև մյուս տիպերը մեկ պատուհանային հավելված (Single Document) կամ վերին մակարդակի մի քանի պատուհանների (Multiple top-level documents) տեսակները: Առաջինը կարող է աշխատել միայն մեկ փաստաթղթի հետ և, ըստ էության, միջանկյալ դիրք է գրավում Երկխոսության հավելվածների և բազմապատուհան հավելվածների միջև:

Սովորաբար մեկ փաստաթղթի մշակումը ենթադրում է միաժամանակ աշխատել դրա հետ կապված փաստաթղթերի հետ: Օրինակ աշխատանքի ընթացքում դրանց մեջ հանապատասխան փոփոխություններ նշումնել կամ դրանցից անհրաժեշտ ինֆորմացիա ստանալ: Ուստի Visual Studio-ում նշումները կապված են հավելվածի նոր տիպ Վերին մակարդակի հավելված մի քանի պատուհաններով: Դավելվածի այս տիպը իրենից ներկայացնում է մի քանի փոփոխապված միապատուհան հավելվածներ: Ստեղծված պատուհաններից առաջինը գլխավորն է, և այն փակելուց փակվում են մնացած պատուհանները (Exit): Մնացած պատուհանները հանդիսանում են ենթակա և դրանց փակումը չի փակում մյուս պատուհանները (close): Մյուս տիպը Console Application-ը կոնսոլային հավելվածը ստեղծվում է որպես օրենք այն դեպքերում, երբ հավելվածի համար պահանջվում է ստեղծել պարզագույն թաղանթ, որը տիրապետում է ամենապարզ ինտերֆեյսի:

Դավելվածների, փաստաթղթերի և ներկայացումների դասերը

Windows միջավայրում ծրագրավորման գաղափարախոսության հիմքում, որտեղ օգտագործվում են MFC դասերի գրադարանները, ընկած է փաստաթուղթ / ներկայացում սկզբունքը:

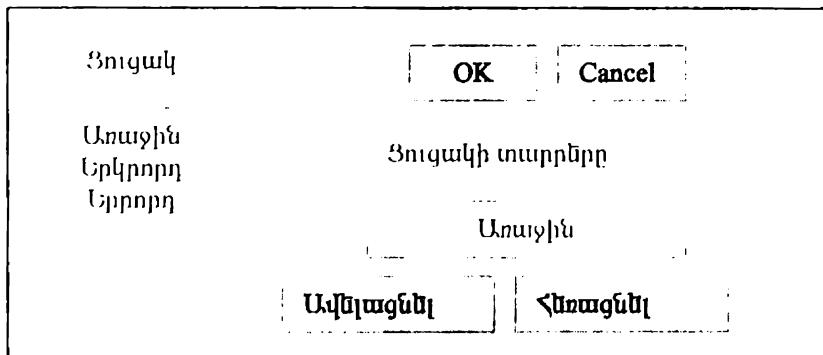
Այն ենթադրում է, որ յուրաքանչյուր հավելվածում պետք է հստակ առանձնացվեն ինֆորմացիայի ծևափոխությունների և նրա արտացոլման ֆունկցիաները: Դեռևս յուրաքանչյուր հավելված, որը կառուցված է տվյալ գաղափարախոսությանը

համապատասխան, պետք է ընդգրկի դասի նկարագրումը, որը ածանցյալ է CDocument դասից, որի վրա է դրվում օպերատիվ հիշողությունում մշակվող ինֆորմացիայի պահպանման խնդիրը, արտաքին հարցումներով այդ ինֆորմացիայի ներկայացումը, սկավառակից այդ ինֆորմացիայի կարդալը և գրելը: Այդ հավելվածը պետք է ընդգրկի դասի նկարագրությունը, որը ծնվել է CView դասից, որի վրա է դրվում էկրանին այդ ինֆորմացիայի արտացոլման խնդիրը և շահագործողի ինտերֆեյսի իրականացումը: Բացի այդ երկու դասերից, հավելվածը կարող է պարունակել և ուրիշ օգտակար դասեր, սակայն հատկապես այդ երկու դասերի փոխհամագործակցումը փաստաթղթի նախատիպի (շաբլոնի) շրջանակներում թույլ է տալիս նշանակալի պարզեցնել շատ ստանդարտ օպերացիաներ՝ ստեղծվող հավելվածի և Windows օպերացիոն համակարգի փոխգործողություններում: Ցանկացած ծրագրի մյուս անբաժանելի քաղկացուցիչը, որը օգտագործում է MFC գրադարան, հանդիսանում է հավելվածի դասը, որը ծնունդ է CWinApp դասի: Տվյալ դասի մեթոդը պատասխանում է ինիցիալիզացման, աշխատանքի և ամբողջ հավելվածի ճիշտ ավարտի համար:

4.5. Visual C++-ում Երկխոսության պատուհանը և ղեկավարման պարզագույն էլեմենտների օգտագործումը

Windows միջավայրում ամբողջական ծրագիրը կարող է ունենալ բազմաթիվ երկխոսության պատուհաններ, որոնցից յուրաքանչյուրը նախատեսվում է փոխանակման կամ ինֆորմացիայի ելքի համար իր առանձնահատուկ դեպքերով։ Յուրաքանչյուր երկխոսության պատուհանի նկարագրություն բաղկացած է երկու մասից, որոնք տեղադրված են տարբեր ֆայլերում երկխոսության պատուհանի ռեսուրսների նկարագրություն, որոնց պահպանման համար օգտագործվում են ռեսուրսների հատուկ ֆայլեր, որոնք, որպես օրենք, ունեն ուղղակի պատուհանի դասի նկարագրություն, որը տեղադրված է իրականացման ֆայլում։ Այդ ֆայլերից յուրաքանչյուրին համապատասխանում է իր վերնագրային ֆայլը։

Դիցուք պետք է ստեղծել երկխոսության հետևյալ պատուհանը։



Տեղադրում ենք ցուցադրված ղեկավարման էլեմենտները։ Երկխոսության պատուհանում ընտրում ենք IDC_List ղեկավարման էլեմենտը և Properties պատուհանում սեղմում ControlEvents սեղմակը։ Արդյունքում կրացվի տվյալ ղեկավարման էլեմենտի կողմից ուղարկվող հաղորդագրությունների ցուցակը։ Բացվող ցուցակում նկնիկի ծախս սեղմակով պետք է նշել LBN_DblClick։ Դամապատասխան տեքստային դաշտում կհայտնվի բացվող ցուցակի նշանը և բացելով այդ ցուցակը կպարունակի միակ տող, որը առաջարկում է ավելացնել ուղղությունուն հաղորդագրության մշակման ֆունկցիան։ Ընտրելով այդ տողը հաղորդագրու-

թյան մշակման ֆունկցիայի անունը կհայտնվի տեքստային դաշտում, խմբագրման պատուհանում կավելանա վերնագրային ֆայլի խմբագրման պատուհանների գոտին և ClistDlg դասի իրականացնան ֆայլը: Բացում ենք երկխոսության պատուհանի ռեսուրսների խմբագրման պատուհանը, առանձնացնում “Ավելացնել” սեղմակը և Properties պատուհանում սեղմում ControlEvents սեղմակը: Նշելով BN_Clicked տողի վրա՝ կրացվի ցուցակ, որը կպարունակի միակ տող և որն էլ առաջարկում է ավելացնել OnBnClickedAdd հաղորդագրության մշակման ֆունկցիան: Ընտրելով այդ տողը՝ ֆունկցիայի անունը կհայտնվի տեքստային դաշտում: Նույնը պետք է կատարել “Հեռացնել” սեղմակի համար:

Բացում ենք ClassView պատուհանը և նրա մեջ List –ի ցուցակը: Նշում ենք մկնիկի աջողվ ClistDlg իրամանը: Կհայտնվի Add Member Variable wisard- List երկխոսության պատուհանը: Դնում ենք Control variable նշիչը (դեկավարման էլեմենտի հետ կապը): Control_ID և Control type –ը կդառնան հասանելի: Control_ID բացվող ցուցակում առանձնացնում ենք IDC_EDIT ռեսուրսի իդենտիֆիկատորը: Category բացվող ցուցակում առանձնացնում ենք Value տողը և Variable name տեքստի դաշտում մտցնում M_List փոփոխականի իդենտիֆիկատորը և սեղմում Finish:

Բացում ենք ListDlg.cpp ֆայլի խմբագրման պատուհանը և մտցնում նրա մեջ փոփոխություններ:

```
void CLisDlg::OnLbnDblclickList()
{
CString Temp;
if (m_List.GetCursel ()!=LB_ERR)
m_List.GetText(m_List.GetCursel (), Temp);
SetDlgItemText(IDC_EDIT, Temp);
}

void ClistDlg::OnBnClickedAdd()
{
UpdateData();
m_List.AddString(m_Edit);
}

void CListDlg::OnBnClickedDelete()
{
```

```

if (m_List.GetCurSel ()!=LB_ERR)
m_List.DeleteString(m_List.GetCurSel());
UpdateData()
}

```

Մտցնում ենք “ցուցակի էլեմենտ” տեքստային դաշտում “Առաջին” և սեղմում “Ավելացնել” սեղմակը. նման ձևով մտցնում ենք “Երկրորդ” և “Երրորդ” տեքստերը: ListBox դեկավարման էլեմենտի ստեղծնան ժամանակ Behaviour բաժնում (Properties պատուհանում) Sort տեքստային դաշտում դրված է True: OnLBnDBLClickList ֆունկցիան ստեղծում է Temp անունով CString դասի օբյեկտ և օգտագործելով CListBox::GetText ֆունկցիա գրվում է նրա մեջ ցուցակի առանձնացված տողի պարունակությունը, որից հետո օգտագործելով Cwnd::SetDlgItemText ֆունկցիան Temp փոփոխականի պարունակությունը գրվում է տեքստային դաշտում:

OnBnClickedAdd ֆունկցիան ընդգրկում է երկու ֆունկցիաների կանչ: UpdateData ֆունկցիան կատարում է գրանցում տեքստային դաշտի պարունակությունը համապատասխան փոփոխականի նեզ, իսկ CListBox::AddString ֆունկցիան ավելացնում է այդ փոփոխականի արժեքը ցուցակում: OnBnClickedDelete ֆունկցիան ստուգում է ցուցակում ընտրված փոփոխականը, որի համար կանչվում է CListBox::GetCurSel ֆունկցիա, որը վերադարձնում է ընտրված տողի ինդեքսը և ստուգում է, թե արդյոք վերադարձվող արժեքը սխալ չի՝ պարունակում: Եթե սխալ չի հայտնաբերվել, ապա այդ ֆունկցիան կանչվում է ևս մեկ անգամ որպես CListBox::DeleteString ֆունկցիայի արգումենտ. ոչնչացնել տող, որի ինդեքսը նշված է որպես արգումենտ: Այն դեպքում, եթե անհրաժեշտ է տեղադրել կամ հանել էլեմենտի առանձնացումը ցուցակից, կարելի է օգտվել CListBox::SetSel ֆունկցիայից, որի առաջին արգումենտը իրենից ներկայացնում է տողի ինդեքս, իսկ երկրորդը տրամաբանական փոփոխական, որին վերադարձվում է True արժեք, եթե անհրաժեշտ է ընտրել ցուցակից տվյալ տողը և False եթե անհրաժեշտ է հանել ընտրվածը:

Աշխատանքի մեկնարկման համար սեղմել <F5> ստեղնը:

Լաբորատոր և գործնական աշխատանքներ

Լաբորատոր աշխատանք N 1

Աշխատանքի նպատակը՝ ճյուղավորման և ցիկլի օպերատորների օգտագործման հնարքների ուսուցումը C-ում:

```
1  /* Իրականացնել էքսպոնենտի մոտավոր հաշվարկի ժրագիրը
   */
2  #include <stdio.h>
3  void main()
4  {
5      Int i;           /* i-ն շարքի անդամների հաշվիչն է */
6      Double eps,b=1.0,r,x; /* eps-ը բացարձակ Ծշտությունը */
7      /* b – ն էքսպոնենտի արժեքը;r-ը հերթական անդամը;x-ը
       ցուցիչը */
8      Printf ("\n մոցրեք արժեքը x=");
9      Scan ("%lf", &x);
10     do {
11         Printf ("\n մոցրեք Ծշտությունը eps=");
12         Scan ("%Lf", &eps);
13     }
14     While (eps<=0.0);
15     For (i=2,r=x;r>eps || r<-eps; i++)
16     { b=b+r
17      R=r*x/i}
18     Printf ("արդյունքը: %f\n",b);
19     Printf ("սխալը: %\n",r);
20     Printf("շարքի անդամների թիվը: %d\n",i);
21 }
```

Դաշվել դրական թվերի գումարը

```
# include <stdio.h>
void main ()
{
double s,x; /* x-ը հերթական թիվն է, s-ը գումարը */
int k; /* k- ն դրական անդամների քանակը */
Printf ("\nՄտցրեք թվերի հաջորդականությունը, վերջում 0\0");
For (x=1.0, s=0.0,k=0;x!=0.0;)
{ scanf ("%lf",%x);
if (x<=0.0) continue
k++;s+=x
printf ("\n Գումարը = %f, դրական թվերի գումարը=%d",s,k);
}
```

Առաջադրանք: Կազմել ծրագիր, որը ելքագրի ստեղնաշարից մուտքագրվող տարբեր տիպի անկետային տվյալները էկրանին:

Լաբորատոր աշխատանք N 2

Աշխատանքի նպատակը: Թվաբանական, տրամաբանական, հարաբերությունների և վերագրումների օպերացիաների ուսուցումը C-ում:

```
1      /* միջինի և դիսպերսիայի հաշվարկը*/
2      #include <stdio.h>
3      void main ()
4      { /* n- ը էլեմենտների քանակն է*/
5      Int i,j,n /* b- ն միջինն է, d- ն դիսպերսիայի
6      գնահատականը;*/
7      float a,b,d,x[100],e; /* a,e- ն օժանդակ փոփոխականներ*/
8      While (1)
9      {
10      Printf ("Ին մոցրեք արժեքը n=");
11      Scanf("n%d",&n);
12      If (n>0 && n<=100) break;
13      Printf ("Ին սխալ է, պետք է լինի 0<n<101");
14      } /* n- իարժեքի մուտքի ցիկլի վերջը*/
15      Printf ("Ին մոցրեք էլեմենտի արժեքները:\n");
16      For (b=0.0,i=0;i<n;i++)
17      {
18      Printf ("x[%d]=",i);
19      Scanf ("%f",&x[i]);
20      b+=x[i]; /* էլեմենտների գումարի հաշվումը*/
21      b/=n; /* միջինի հաշվարկը*/
22      For (j=0,d=0.0;j<n;j++)
23      {
24      a=x[j]-b;
25      D+=a*a
26      } /* դիսպերսիայի գնահատականը*/
27      d/=n
28      Printf ("Ին միջինը=%f, դիսպերսիան=%f,b,d);
29      }
```

Առաջադրանք: Կազմել և տպել ծրագիր, որը տառ և տու ֆունկցիաների (օգնությամբ) օգտագործմամբ լուծում է նվազագույն և առավելագույն արժեքների հաշվման խնդիրը:

Լաբորատոր աշխատանք N 3

Աշխատանքի նպատակը. Ծրագրերի տրամաբանական կազմակերպման և ֆունկցիաների օգտագործման ուսուցումը C-ում:

$$C_n^m = \frac{n!}{m!(n-m)!} \text{ ֆունկցիայի օգնությամբ } \quad \text{բինոմալ գործակցի հաշվարկը}$$

Որտեղ $n \geq m \geq 0; n, m - \text{ը ամբողջ թվեր են:}$

```
# include < stdio.h>
int fact (int k) /*k! ֆակտորիալի հաշվարկը */
{
int j,i; /* օժանդակ փոփոխականներ */
for (i=1,j=1;i<=k;i++) /* հաշվման ցիկլ */
j*=i;
return j;
} /* Փունկցիայի որոշման վերջը */
/* բինոմալ գործակցի հաշվարկը */
void main()
{
int n,m,nmc,nm; /* nm-ը (n-m)-ի արժեքն է */
/* nmc -ն բինոմալ գործակցի արժեքը */
while (1)
{printf ("\n մտցրեք n=");
scanf ("%d",&n);
printf ("մտցրեք m=");
scanf ("%d",&m);
if (m>=0 && n>=m && n<10) break;
printf ("Սխալ կա! պետք է լինի 0<=m<=n<10");}
nm=n-m;
nmc=fact(n)/fact(m)/fact(nm);
printf("n բինալ գործակից=%",nmc);
} /* հիմնական ծրագրի վերջը */
```

Առաջադրանք: Ներկայացվող խնդրում լրացնել բաց թողնված մուտքի և ելքի կազմակերպումը: Դաշվել գումարը.

$$S = \sum_{j=1}^{10} \prod_{i=1}^5 a_{ji}$$

```
1      Double a[10][5];
2      For (j=0,s=0.0;j<10;j++)
3      { For (i=0,p=1.0;i< 5;i++)
4      { If (a[j][i]==0.0) break;
5      P*=a[j][i] ;}
6      If (i< 5) continue;
7      S+=p; }
```

Լաբորատոր աշխատանք N 4

Աշխատանքի նպատակը: Զանգվածների հայտարարման և դիմելու հնարքների ուսուցումը C++-ում:

Գտնել տրված միջակայքում գտնվող 5-ին բազմապատիկ թվերի միջին թվաբանականը

```
#include<iostream.h>
main ()
{
int n,i,j,p;
double a[1000],s;
while (1)
{
cout<<"\n մտցրեք էլեմենտի քանակը";
cin >>&n;
if (n>1 && n<=1000) break;
cout<< "սխալ է, նորից մտցրեք \n";
}
cout <<"մտցրեք զանգվածի էլեմենտների արժեքները \n";
for (i=0;i<n;i++)
{
cout <<"a["<< i+1<<"]=";
cin>>a[i];
}
s=0,j=0;
for (i=5,i<1000;i+=5)
{
s+=i
j+=1
}
s/=j
cout<<s;
}
```

Գտնել $a_1, a_2, a_3, \dots, a_{100}$ տարրերից մեծագույնը:

```
# include< iostream.h>
main ()
{
int n,i,j;
double a[100],Max;
while(1)
{
cout<<"\n Մտցրեք էլեմենտների թիվը";
cin>>&n;
if (n>1 && n<=100) break;
cout<<"սխալ է, նորից մտցրեք:\n"
}
cout<<"Մտցրեք զանգվածի էլեմենտների արժեքները:\n"
for (j=0;j<n;j++)
{
count<<"a["<<j+1<<"]=";
cin>>a[j];
}
Max=a[0];
For (j=1;j<n;j++)
If (a[j]>Max)
{
Max=a[j];
}
cout<<"\n Մեծագույնը="<<Max;
}
```

Առաջադրանքներ:

- Կազմել և ստանալ ծրագրի տեսքը, որը ընտրում է 1-ից 1000 բոլոր զույգ թվերը և կենտ թվերը:
- Կազմել և ստանալ ծրագրի տեսքը, որը ընտրում է 1-ից 1000 բոլոր 3-ով վերջացող թվերը:
- Կազմել և ստանալ ծրագրի տեսքը, որը ընտրում է 1-ից 1000 բոլոր 2-ով վերջացող թվերը:

Լաբորատոր աշխատանք N 5

Աշխատանքի նպատակը: Դասերի ուսուցումը C++ -ում:
Դասերի համար դասի սահմանում

```
#include <iostream.h>
#include <string.h>
class addr {
    char name[40];
    char street[40];
    char city[30];
    char state[3];
    char zip[10];
public:
    void store(char *n, char *s, char *c, char *t, char *z);
    void display();
};
void addr :: store(char *n, char *s, char *c, char *t, char *z)
{
strcpy(name,n);
strcpy(street,s);
strcpy(city,c);
strcpy(state,t);
strcpy(zip,z);
}
void addr :: display()
{
cout<<name<<"\n";
cout<<street<<"\n";
cout<<city<<"\n";
cout<<state<<"\n";
cout<<zip<<"\n\n";
}
main()
{
addr a
a.store("Կ.Խաչատրյան", "Նորքի 6-րդ զանգված", "Երևան",
"Հայաստան", "374");
a.display();
return 0;
}
```

Լարորատոր աշխատանք N 6

Աշխատանքի նպատակը: Ժառանգականության ուսուցումը C++-ում: Դա մի մեխանիզմ է, որի արդյունքում մի դասը կարող է ժառանգել մյուսի հատկությունները: Այն բույլ է տալիս կառուցել դասերի(class) հիերարխիա անցնելով ավելի ընդհանուրից ավելի հատուկի, բազային դասից(base class) ածանցյալ դասի(derived class): Սովորաբար ժառանգման պրոցեսը սկսվում է բազային դասը տալուց: Բազային դասը որոշում է բոլոր այն որակները որ պետք է լինի ընդհանուր բոլոր ածանցյալ դասի համար: Օրինակ՝ որոշենք ընդհանուր բազային fruit (միջոց) դասը, որը բնութագրում է մրգերի որոշ բնութագրեր: Այդ դասը ժառանգվում է երկու ածանցյալ դասերի կողմից՝ Apple և Orange: Այդ դասերը պարունակում են հատուկ ինֆորմացիա կոնկրետ մրգերի խնձորի և նարինջի մասին:

```
// դասի ժառանգման օրինակ  
# include <iostream.h>  
# include <string.h>  
enum yn{no, yes};  
enum color{red, yellow, green,  
orange};  
void out(enum yn x)  
char *c[]={  
"red", "yellow", "green", "orange"  
};  
  
//մրգերի սկզբնական դասը  
  
class fruit {  
//Այս բազային դասում բոլոր  
էլեմենտները բաց են  
Public:  
enum yn annual;  
enum yn perennial;  
enum yn tree;  
enum yn tropical;  
enum color clr;  
char name [40];  
};
```

```
//Ածանցյալ դաս խնձոր  
Class Apple: public fruit {  
enum yn cooking;  
enum yn crunchy;  
enum yn eating;  
Public:  
void seta(char*n, enum color c,  
enum yn ck, enum yn crchy ,  
enum yn e);  
void show ();  
};  
  
//Ածանցյալ դաս նարինջ  
class Orange:public fruit{  
enum yn juice;  
enum yn sour;  
enum yn eating;  
Public:  
void seto(char *n, enum color c, enum yn j, enum yn sr, enum  
yn e);  
void show();  
};
```

```

void Apple::Seta(char*n,
enum color c, enum yn ck,
enum yn crchy, enum yn e)
{
strcpy(name,n);
annual=no;
perennial=yes;
tree=yes;
tropical=no;
clr=c;
cooking=ck;
crunchy=crchy;
eating=e;
}
void orange::seto(char*n,
enum color c, enum yn j,
enum yn sr, enum yn e)
{ strcpy (name,n);
annual=no;
perennial=yes;
tree=yes;
tropical=yes;
clr=c;
juice=j;
sour=sr;
eating=e;
}
void Apple::show()
{
cout<<name<<"խնձորը-
դա"«"\n";
cout<<"միամյա բույս է";
out(annual);
cout<<"բազմամյա բույս է";
out(perennial);
}

```

```

cout<<"ծառ է:"; out(tree);
cout<<"արևադարձային է:";
out(tropical);
cout<<"Գույնը:"c[clr]<<"\n";
cout<<"Հեշտ պատրաստվում
է:"; out(cooking);
cout<<"ճրթճրթացող:";
out(crunchy);
cout<<"ուտելու:";out(eating);
cout<<"\n";
}
void orange::Show()
{
Void out(enum yn x)
{if(x==no)cout<<"ոչ\n";
else cout<<"Այո\n";
}
main()
{
Apple a1,a2;
orange o1,o2;
a1.seta("դեմիրճյան",red,no,y
es,yes);
a2.seta("բելֆլոր",red,yes,no,
yes);
o1.seto("պուպոկ",orange,no,
no,yes);
o2.seto("վալենսիա",orange,y
es,yes,no);
a1.show()
a2.show()
o1.show()
o2.show()
return o;
}

```

Գործնական և լաբորատոր աշխատանքների համար առաջարկվող խնդիրներ

1. Տրված է ամբողջ թվերի միաչափ զանգված: Գրել ծրագիր, որը որոշի նվազագույն էլեմենտների քանակը:
2. Տրված է ամբողջ թվերի միաչափ զանգված: Գրել ծրագիր, որը որոշի 7-ին բազմապատիկ էլեմենտների քանակը:
3. Տրված է ամբողջ թվերի միաչափ զանգված: Գրել ծրագիր, որը որոշի վերջին նվազագույն արժեքի համարը:
4. Գրել սցենար, որը որոշի ամբողջ թվերի միաչափ զանգվածում առաջին մեծագույն արժեքի համարը:
5. Գրել սցենար, որը որոշի թե քանի տարրեր թվեր կան տրված զանգվածում:
6. Ստեղծել ֆունկցիա և դրա օգնությամբ որոշել երկու թվերի ամենամեծ ընդհանուր բաժանարարը:
7. Ստեղծել ֆունկցիա և դրա օգնությամբ որոշել երկու թվերի ամենափոքր ընդհանուր բազմապատիկը:
8. Գտնել տրված բնական թվի մեջ մեծագույն և փոքրագույն նիշերի համարները:
9. Գտնել տրված բնական թվի մեջ փոքրագույն նիշին հավասար թվանշանների քանակը:
10. Որոշել տրված թիվը կատարյա՞լ է, թե՝ ոչ, այսինքն՝ հավասար է իր բաժանարարների գումարին, թե՝ ոչ:
11. Գտնել տրված բնական թվի մեջ 2 մեծագույն ոչ հավասար թվանշանները և դրանց համարները:
12. Որոշել, թե տրված թվերը փոխադարձ պարզ թվե՞ր են, թե՝ ոչ:
13. Տրված բնական թվի թվանշանները ձևափոխել ըստ աճման կարգի:
14. Որոշել, թե տրված թվերը երկվորյա՞կ թվեր են, թե՝ ոչ, այսինքն՝ պարզ են և դրանց տարրերությունը հավասար է 2-ի:
15. Գրել սցենար, որը տրված երկուական համակարգի թիվը ներկայացնի տասսական համակարգով:
16. Որոշել, թե տրված թվերը բարեկամակա՞ն թվեր են, թե՝ ոչ, այսինքն՝ նիրանցից յուրաքանչյուրը հավասար է մյուսի բաժանարարների գումարին, թե՝ ոչ:
17. Գրել սցենար, որը տրված տասսական համակարգի թիվը ներկայացնի երկուական համակարգով:

18. Տրված հոռմեական թիվը ձևափոխել արարականի:
19. Որոշել, թե տրված թվերը ֆիբոնաչիի թվե՞ր են, թե՞ ոչ:
20. Խտացնել տրված զանգվածը, հեռացնելով պարզ թվերը:
21. Որոշել տրված միջակայքում Արմսրոնգի թվերը:
22. Գրել սցենար, որը միավորի երկու միաչափ զանգվածները և դասավորի աճման կարգով:
23. Որոշել տրված բնական թվի մեջ երրորդ անգամ հանդիպող 7 թվանշանի համարը (ինդեքսը), եթե չկա՝ տալ հաղորդագրություն:
24. Որոշել տեքստում հանդիպող այբուբենի տառերի քանակը:
25. Որոշել տեքստում յուրաքանչյուր տառի քանակը տոկոսներով արտահայտված:
26. Որոշել տեքստում բառերի քանակը:
27. Գրել սցենար, որը կարգավորի միաչափ զանգվածի էլեմենտները նվազման կարգով:
28. Որոշել [տ,ո] միջակայքում 7-ին բազմապատիկ թվերի գումարը:
29. Տրված են եռանկյան գագաթի կոորդինատները: Կազմել ծրագիր, որը որոշի, թե տրված կետը ընկած է եռանկյան ներսում, թե՞ ոչ:
30. Հաշվել $S=1^m+2^m+3^m+\dots+n^m$ գումարը:
31. Գտնել 457-ը չգերազանցող բոլոր եռանիշ թվերի գումարը, որոնք 6-ի վրա բաժանելիս տալիս են 2 մնացորդ:
32. Գրել սցենար, որը հատվածի կիսման մեթոդով որոշի հավասարման արմատը:
33. Որոշել կիսամյակի ստացած գնահատականներով որոշվող ուսանողին վճարվող կրթաթոշակի չափը:
34. Անկետայում բերված են յոթ աշխատակիցների մասին տվյալներ ազգանունը և աշխատանքի ընդունվելու ամսաթիվը: Գրել սցենար, որը հաշվի ստաժը և առավելագույն թվով նույն ստաժը ունեցողների քանակը:
35. Անկետայում բերված են կիսամյակի սկզբի և վերջի մասին ամսաթվային տվյալներ: Պահանջվում է գրել սցենար, որը շաբաթվա օրով որոշի:
 - կիսամյակում տվյալ առարկային հատկացված ժամերը
 - պարապմունքների անցկացման ամսաթվերը
 - պարապմունքի համարով որոշի անցկացման ամսաթիվը:

36. Գրել ծրագիր, որը մուտքագրի աշխատողի աշխատաժ ժամերի քանակը և յուրաքանչյուրի աշխատավարձի չափը: Այնուհետև ելքագրի աշխատողի ամփոփ աշխատավարձը:
37. Ստեղծել Card դաս, որը պաշտպանում է գրադարանային քարտերի կատալոգը: Այդ դասը պետք է պահպանի գրքի վերնագիրը, հեղինակի անունը և վերցրած օրինակների քանակը: Օգտագործել Store() բաց ֆունկցիա անդամ, զրքերի մասին ինֆորմացիայի պահպանման համար և Show() բաց ֆունկցիա անդամ էկրանին ինֆորմացիայի ելքագրման համար: Պայմ () ֆունկցիայում կարծ լուսաբանել ստեղծված դասի աշխատանքը:
38. Ստեղծել sleep () ֆունկցիա, որը կանգնեցնում է համակարգչի աշխատանքը տրված վայրկյանների չափով, և որը տրվում է ֆունկցիայի արգումենտում: Ծանրաբեռնել sleep-ը այնպես, որպեսզի նա կանչվի կամ ամբողջ թվով կամ տողով, որը տալիս է ամբողջ թիվ: Օրինակ, այս երկու կանչերն էլ պետք է ստիպեն համակարգչին կանգնեցնել 10 վայրկյան: sleep(10); sleep("10"): Ցուցադրել այս ֆունկցիայի աշխատանքը կարծ ծրագրում:
39. Ստեղծել դաս, որը պարունակում է անվան և հասցեի մասին ինֆորմացիա: Պահպանել այդ բոլոր ինֆորմացիան սիմվոլային տողում դասի փակ մասում: Մտցնել դասում բաց ֆունկցիա՝ անունը և հասցեն հիշելու համար: Ընդգրկել նաև բաց ֆունկցիա, որը ելքագրի այդ անունը և հասցեն էկրանին (այդ ֆունկցիաները անվանել Store () և display ()):
40. Ստեղծել սկզբնական building բազային դաս, շենքում հարկերի, սենյակների և ընդհանուր մակերեսի քանակը պահպանելու համար: Ստեղծել ածանցյալ դաս house, որը ժառանգում է building-ը և պահպանում է հետևյալ ինֆորմացիան՝ լողասենյակների քանակը և ննջասենյակ-ների քանակը: Բացի դրանից ստեղծել office ածանցյալ դաս, որը ժառանգում է building-ը և պահպանում է կրակ-մարիչների և հեռախոսների քանակը:

Կուրսային աշխատանքի համար առաջարկվող խնդիրներ

1. Կառուցել նախագիծ, որը մուտքագրի խմբի համարը, խմբի ուսանողների ցուցակը և կիսամյակում հանձնած առարկաներից նրանց ստացած գնահատականները: Ցուցակը դասակարգել ըստ առարկաներից ստացած գնահատականների նվազման կարգով: Կատարել ամփոփ վերլուծություն:
2. Կառուցել նախագիծ, որը ներկայացնի ըստ կիսամյակի 5, 4, 3, և 2 ստացողների քանակները դիագրամայի տեսրով: Կատարել ամփոփ վերլուծություն:
3. Կառուցել նախագիծ, որը ներկայացնի ինստիտուտի մասնագիտությունները, ամբիոնները և պրոֆեսորադասախոսական անձնակազմը: Կատարել դիագրամային վերլուծություն ըստ գիտական կոչումների:
4. Կառուցել նախագիծ, որը ներկայացնի ինստիտուտի ֆակուլտետներն ըստ իրենց մասնագիտությունների և ուսանողների թվի: Կատարել սեռատարիքային վերլուծություն:
5. Կառուցել նախագիծ, որը ներկայացնի ինստիտուտի աղջինիստրատիվ կառուցվածքը: Կատարել ամփոփ վերլուծությունները:
6. Կառուցել նախագիծ, որը ներկայացնի ինստիտուտի գրադարանային համակարգը և նրա աշխատանքը:
7. Կառուցել նախագիծ, որը ներկայացնի ինստիտուտի կադրերի բաժնի աշխատանքը:
8. Կառուցել նախագիծ, որը ներկայացնի ինստիտուտի ուսումնական մասի աշխատանքը:
9. Կառուցել նախագիծ, որը ներկայացնի ինստիտուտի տնտեսական մասի աշխատանքը:
10. Կառուցել նախագիծ, որը ներկայացնի ինստիտուտի գիտական նախագծերը:
11. Ուսումնական պրոցեսի համար նախագծել ինֆորմացիոն համակարգ:
12. Կառուցել նախագիծ, որը ավտոմատացնի բանկային համակարգում դեպոզիտների հավաքագրման աշխատանքները:
13. Կառուցել նախագիծ դեղատան աշխատանքների տեսլեկատու ինֆորմացիոն համակարգի համար:

- 14.Կառուցել նախագիծ, որը իրականացնի հիմնական միջոցների հաշվառման խնդիրը:
- 15.Կառուցել նախագիծ, որը իրականացնի պահեստում նյութերի շարժի օպերատիվ հաշվառումը:
- 16.Կառուցել նախագիծ, որը իրականացնի ավիաչերթերի հաշվառումը:
- 17.Կառուցել նախագիծ, որը իրականացնի ավիատոմսերի հաշվառումը:
- 18.Կառուցել նախագիծ, որը իրականացնի հիվանդանոցի գործունեության ամփոփ հաշվառումը:
- 19.Կառուցել նախագիծ, որը իրականացնի գյուղատնտեսական մթերքների վերամշակման կոնկրետ հիմնարկի ամփոփ հաշվառումը:
- 20.Կառուցել նախագիծ, որը իրականացնի սպառված էլեկտրաէներգիայի դիմաց կատարված վճարումների ամփոփ հաշվառումը:
- 21.Կառուցել նախագիծ, որը իրականացնի բնակչության կոմունալ ծառայությունների վարձերի վճարման հաշվառումը:
- 22.Կառուցել նախագիծ, որը իրականացնի կենսաբոշակների հաշվարկների և վճարումների հաշվառումը:
- 23.Կառուցել նախագիծ, որը իրականացնի բնակարանների վաճառքով զբաղվող բրոբերային ֆիրմաների գործունեության հաշվառումը:
- 24.Կառուցել նախագիծ, որը իրականացնի հյուրանոցում աղմինիստրատորի աշխատանքի փաստաթղթաշրջանառության ավտոմատացումը:
- 25.Կառուցել նախագիծ, որը ավտոմատացնի բանկի գործառնական օրվա փաստաթղթաշրջանառությունը:
- 26.Կառուցել նախագիծ, որը իրականացնի հիմնարկության աշխատողների աշխատավարձի հաշվարկը:
- 27.Կառուցել նախագիծ, որը իրականացնի ուսանողների թոշակների հաշվարկը:
- 28.Կառուցել նախագիծ, որը իրականացնի միջքաղաքային ավտոտրանսպորտի աշխատանքների հաշվառումը:
- 29.Կառուցել նախագիծ, որը լուսաբանի ֆուտբոլի աշխարհի առաջնությունը:
- 30.Կառուցել նախագիծ «Ծրագրավորման տեխնոլոգիաներ» առարկայից համակարգչի օգնությամբ տեստերով քննություն կազմակերպելու համար:

Օգտագործված գրականություն

1. Ա. Աբրահամեան, Եռլ. Բարոնեան, Գ. Սարկարով, Կ. Մարկարեան, Ինֆորմատիկայի հիմունքներ, Ս. Ղազարի Սխիթարեան տպարան, Վենետիկ, 1991
2. Ո. Վ. Արմենակյան, Ա. Ն. Սարգսյան, Ծրագրային արտադրանքի նախագծման մեթոդները: Մեթոդական ձեռնարկ, Երևան, 2001
3. Михаэль Рейтингер, Геральд Муч, Visual Basic 6.0, "Ирина", ВНВ, Киев, 2000
4. Э. Петруцос, К. Хай, Visual Basic 6.0 и VBA для профессионалов, СПб, 2000
5. А. А. Васильев, Л. Андреев, VBA В OFFICE 2000, Учебный курс, СПб, "Питер", 2001
6. В. В. Полбельский, С. С. Фомин. Программ-и на языке С. учеб. пособие. М.Фис., 1998
7. В. В. Полбельский . С++, учеб. пособие, М. Фис., 2000
8. Герберт Шилдт, Самоучитель С++, ВНВ, СПб, 1997
9. Марк Луис, Visual C++ 6.0, М. Лаборатория базовых знаний, 1999
10. Х. М. Деимпел, П. Дж. Деимпел, Как программировать на С++, М., 2003