ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
Д.Серікбаев атындағы ШЫҒЫС ҚАЗАҚСТАН МЕМЛЕКЕТТІК ТЕХНИКАЛЫҚ УНИВЕРСИТЕТІ

В.Л. Никифоров, Л.П. Латкина, С.Ж. Рахметуллина, Г. Жомартқызы

Бағдарламалу технологиясы

5В070300 «Ақпараттық жүйелер», 5В070400 «Есептеу техникасы және бағдарламалық қамтамасыз ету» мамандықтары студенттеріне арналған оқу құралы

Өскемен
2011

УДК 681.3

В.Л. Никифоров Бағдарламалау технологиясы: 5В070300 «Ақпараттық жүйелер», 5В070400 «Есептеу техникасы және бағдарламалық қамтамасыз ету» мамандықтары студенттеріне арналған оқу құралы: Оқу құралы /В.Л.Никифоров, Л.П. Латкина, С.Ж. Рахметуллина, Г. Жомартқызы / ШҚМТУ.-Өскемен, 2011.-345 б.

Оқу құралында консольді қосымшаларды бағдарламалауға арналған, C# тілінде бағдарламалау технологиясымен қатар Visual Studio.NET ортасында бағдарламалау технологиясына арналған сұрақтар, пән бойынша өзін-өзі тексеруге арналған сұрақтар мен мысалдар жан-жақты қарастырылған.

5В070300 «Ақпараттық жүйелер», 5В070400 «Есептеу техникасы және бағдарламалық қамтамасыз ету» мамандықтары студенттеріне арналған оқу құралы.

Ақпараттық технологиялар және энергетика факультеті оқу-әдістемелік кеңесімен бекітілген.

№ хаттама

МАЗМҰНЫ

АЛҒЫ СӨЗ 10
КІРІСПЕ 11
1 C# ТІЛІНЕ КІРІСПЕ 12
1.1 Алгоритм туралы түсінік 12
1.2 Есепті шешу алгоритмін құру кезеңі 13
1.3 Алгоритмді жүзеге асыру кезеңі 14
1.4 .NET платформасының құрылымы 14
1.5 Visual Studio.NET ортасы 15
1.6 Жобаны жасау. Ортаның негізгі терезелері 16
1.7 C# тілінің символикасы 18
1.8 Айнымалыларды сипаттау 19
1.9 C# тілінің деректер типі 19
1.10 C# тілінің тұрақтылары 20
1.11 C# тілінің атаулар кеңістігінің ұғымы 21
1.12 Өзін-өзі тексеру сұрақтары 23
2 C# ТІЛІНІҢ ҚАРАПАЙЫМ ОПЕРАТОРЛАРЫ 24
2.1 C# ТІЛІНДЕГІ БАҒДАРЛАМА ҚҰРЫЛЫМЫ 24
2.2 Бағдарлама мысалы 24
2.3 Монитор экранына ақпаратты шығару 27
2.4 Пернетақтадан деректерді енгізу 28
2.5 Меншіктеу операторы 30
2.6 C# тілінің стандартты математикалық функциялары 32
2.7 Мәтіндік файлдармен жұмыс 35
2.8 Өзін-өзі тексеру сұрақтары 37
3 C# ТІЛІНІҢ КҮРДЕЛІ ОПЕРАТОРЛАРЫ 36
3.1 Шартты өту операторы if 36
3.2 For циклінің операторы 41
3.3 Өзін-өзі тексеру сұрақтары 46
4 ШАРТТЫ ЦИКЛ ОПЕРАТОРЛАРЫ 47
4.1 While циклінің операторы 47
4.2 do – while циклінің операторы 51
4.3 Циклді пайдаланып есепті шешу мысалы 54
4.4 Өзін-өзі тексеру сұрақтары 56
5 МАССИВТЕР 58
5.1 Сілтемелік тип туралы мағлұмат 58
5.2 Массив ұғымы 58
5.3 Әр түрлі есептерде массивтерді қолдану мысалдары 59
5.4 Динамикалық массивтер 66
5.5 Өзін-өзі тексеру сұрақтары 67
6 МАССИВТЕРДІ ӨҢДЕУ АЛГОРИТМДЕРІ 69
6.1 Алгоритмдер мен массивтер 69
6.2 Массив элементтерін сұрыптау 69

6.3 Массивте мәні бойынша элементтерді іздестіру 72
6.4 Өзін-өзі тексеру сұрақтары 80
7 КЛАСС ӘДІСТЕРІ 81
7.1 Класс әдістері мен деректер туралы мағлұмат 81
7.2 Бағдарламада класс әдістерін қолдану 83
7.3 Әдістерді қайта жүктеу 89
7.4 Рекурсия 90
7.5 Өзін-өзі тексеру сұрақтары 94
8 ЖОЛДЫҚ АЙНЫМАЛЫЛАР 95
8.1 Символдық тип 95
8.2 Жолдық айнымалы ұғымы 99
8.3 Өзін-өзі тексеру сұрақтары 107
9 КӨПӨЛШЕМДІ МАССИВТЕР 108
9.1 Екі өлшемді массивтер 108
9.2 Көп өлшемді массивтер 111
9.3 Array массив класы 113
9.4 Өзін-өзі тексеру сұрақтары 118
10 ҚҰРЫЛЫМДАР 120
10.1 Құрылымдар туралы мағлұмат 120
10.2 Құрылым массивін қолдану 122
10.3 C# тіліндегі тізім 126
10.4 C# тіліндегі файлдар 128
10.5 Файлдармен жұмыс жасағанда сериализация мен десериализацияны қолдану мысалы 128
10.6 Өзін-өзі тексеру сұрақтары 133
11 СТЕКТЕР, КЕЗЕКТЕР ЖӘНЕ ТІЗІМДЕР 134
11.1 Тізімдік құрылымдар 134
11.2 Стек типіндегі құрылыммен жұмысты ұйымдасыру 135
11.3 Кезек типіндегі құрылыммен жұмыстарды ұйымдастыру 138
11.4 Тізім типіндегі құрылыммен жұмыстарды ұйымдастыру 141
11.5 Кеңейтілген іздеу мүмкіндіктері бар тізімдер 145
11.6 Өзін-өзі тексеру сұрақтары 146
12 ГРАФТАР 148
12.1 Графтар теориясының негізгі анықтамалары 148
12.2 Компьютер жадысында графты көретудің тәсілдері 150
12.3 Флойд алгоритмі 154
12.4 Дейкстр алгоритмі 156
12.5 Өзін–өзі тексеру сұрақтары 161
13 ГРАФТАРҒА АРНАЛҒАН АЛГОРИТМДЕР 162
13.1 Графтарды жүріп өту алгоритмдері 162
13.2 Берілген екі графтың арасында барлық маршруттарды іздеу алгоритмі 167
13.3 Өзін-өзі тексеру сұрақтары 173
14 ЕРЕКШЕ ЖАҒДАЙЛАРДЫ АЛДЫН АЛУ 175

14.1 Ерекше жағдайларды алдын алу туралы түсініктер 175
14.2 Ерекше жағдайларды алдын алу бойынша мысал 177
14.3 Өзін-өзі тексеру сұрақтары 178
15 VISUAL STUDIO.NET ВИЗУАЛДЫ БАҒДАРЛАМАЛАУ ОРТАСЫ 180
15.1 Объекті-бағытталған бағдарламалауға кіріспе 180
15.2 Windows жүйесінің оқиғаларды басқару ұғымы 181
15.3 Visual Studio.Net ортасының негізгі терезелері 182
15.4 C# жобасын құру бойынша негізгі құрылымдық элементтер 184
15.5 Бірінші бағдарламаны құру мысалы 185
15.6 Өзін-өзі тексеру сұрақтары 190
16 БАСҚАРУ ЭЛЕМЕНТТЕРІ 191
16.1 Формаларды визуалды жобалау технологиясы 191
16.2 Toolbox панеліндегі басқару элементтері 193
16.3 Басқару элементтерін қолдану мысалы 195
16.4 Өзін-өзі тексеру сұрақтары 197
17 C# ТІЛІНІҢ ГРАФИКАЛЫҚ ИНТЕРФЕЙСІ 199
17.1 System.Drawing атаулар кеңістігі 199
17.2 Graphics класы 199
17.3 Бағдарламалық жүзеге асыру мысалы 202
17.4 Өзін-өзі тексеру сұрақтары 207
18 ҚОСЫМШАДА МЕНЮДІ ҚОЛДАНУ 208
18.1 Бағдарлама менюі 208
18.2 Қосымшаның инструментті панелін құру 211
18.3. Өзін-өзі тексеру сұрақтары 214
19 ДИАЛОГТЫҚ МЕНЮДІ ҚОЛДАНУ 216
19.1 Матрицамен жұмыс жасауға арналған оқиғалар өңдеуіштері 216
19.2 Файлды ашуға арналған оқиғалардың өңдеуіштері 217
19.3 Файлға жазу бойынша оқиғалардың өңдеуіштері 219
19.4 Мәтінмен жұмыс істеуге арналған оқиғаларды өңдеуіші 220
19.5 Өзін-өзі тексеру сұрақтары 223
20 КӨПТЕРЕЗЕЛІ ҚОСЫМШАЛАР 224
20.1 Негізгі «батырмалық» форманы құру 224
20.2 Қосымшаның жаңа түрлерін қосу 226
20.3 Негізгі форма оқиғаларының өңдеуіштері 227
20.4 Мәндерді кесте түрінде көрсету және редакциялау 229
20.5 Тіктөртбұрыштарды графикалық формада көрсету 233
20.6 Өзін-өзі тексеру сұрақтары 236
21. КЛАСС ҰҒЫМЫ 237
21.1 Класс ұғымы 237
21.2 Класс құрамы 238
21.3 Класс әдістері 240
21.4 Объект құрылымы 242
21.5 Оқу бағдарламасының мысалы 242

21.6 Өрістерге қол жеткізу 244
21.7 Өзін-өзі тексеру сұрақтары 245
22 КЛАСС ЭЛЕМЕНТТЕРІ 247
22.1 Конструкторлар 247
22.2 Деструкторлар 250
22.3 Қасиеттер 250
22.4 This сілтемесі бойынша параметр 252
22.5 Класс оқиғалары 252
22.6 Класта қолданылатын операцияларды қайта анықтау 253
22.7 Өзін-өзі тексеру сұрақтары 256
23 ОББ ПРИНЦИПТЕРІ 257
23.1 Инкапсуляция ұғымы 257
23.2 Мұрагерлік ұғымы 260
23.3 Өзін-өзі тексеру сұрақтары 265
24 ПОЛИМОРФИЗМ ПРИНЦИПІ 267
24.1 Полиморфизм ұғымы 267
24.2 Әдістерді статикалық мұралану мысалы 269
24.3 Әдістердің динамикалық мұралануының мысалы 272
24.4 Өзін-өзі тексеру сұрақтары 276
25 ИНТЕРФЕЙСТЕРДІ ПАЙДАЛАНУ 277
25.1 Интерфейс ұғымы 277
25.2 Интерфейс синтаксисі 278
25.3 IEnumerable стандартты интерфейсін қолдану 281
25.4 Өзін-өзі тексеру сұрақтары 286
26 КЛАСТАРДЫҢ КОМПОЗИЦИЯСЫ ЖӘНЕ КОЛЛЕКЦИЯСЫ 288
26.1 Кластардың композициясы және коллекциясы ұғымы 288
26.2 Кластың композициясы мен коллекциясын қолдану мысалы 289
26.3 Framework-тың кейбір коллекциялары 291
26.4 ArrayList коллекциясы 291
26.5 Өзін-өзі тексеру сұрақтары 298
27 ДЕЛЕГАТТАР 300
27.1 Делегат ұғымы 300
27.2 Делегаттың сипаттамасы 300
27.3 Делегатты қолдану мысалы 301
27.4 Делегаттардың үйлесімділігі 303
27.5 Делегаттардың базалық кластарының әдістері 303
27.6 Өзін-өзі тексеру сұрақтары 305
28 ОҚИҒАЛАР 306
28.1 Оқиғалар ұғымы 306
28.2 Visual Studio.NET ортасының жиі қолданылатын кейбір оқиғалары 307
28.3 Кластардің стандартты оқиғаларын қолдану мысалы 309
28.4 Кластардың стандартты емес оқиғалары 311

28.5 Өзін-өзі тексеру сұрақтары 317
А қосымшасы 318
ПӘНДІК КӨРПСЕТКІШ 345
ӘДЕБИЕТТЕР ТІЗІМІ 346

АЛҒЫ СӨЗ

Оқу құралы дегеніміз – белгілі мағлұматты ұғынықты түрде мазмұндайтын басылымның ерекше түрі. Әдетте оқулықтар біршама уақыт бойына сынақтамадан өткен, қайта жазылған дәрістер конспектісі.

«Бағдарламалау технологиясы» пәні түрлі курстарда бағдарламалауды оқытылатын пәндермен тығыз байланысты, бірінші курста оқытылатын «Алгоритмизация және бағдарламалау» пәнінен бастап соңғы курста оқытылатын күрделі «Ақпараттық жүйелермен» аяқталады.

Сондықтан жоғары оқу орындарының бірінші курс студенттеріне арналған бағдарламалау технологиясы бойынша оқу құралының мазмұнында осы пән бойынша соңғы курста оқитын студенттерге арналған оқулық мазмұнынан өзгешеліктері бар.

Бағдарламалау технологиясы бойынша ұсынылып отырған оқу құралы бірінші курс студенттеріне арналады.

Оқу құралының екінші маңызды ерекшелігі – бағдарламалау саласында ешбір мектептік дайындығы жоқ студенттерге арналуында.

Оқу құралының мазмұны .NET платформасы үшін әзірленген C# бағдарламалау тілінің негіздерінен, C# тілінің бағдарламалау технологиясынан және визуальды бағдарламалау ортасы ұсынатын бағдарламалау технологиясынан тұрады.

Оқу құралын жазу барысында авторлар көптеген материалдарды, басқа оқытушылардың кітаптарын қолданды.

Әдетте басылымның жазылуына көмек ұсынған жақын адамдарға алғыс айту қалыптасқан. Оқу құралын жазу барысында бізге ең жақын болып кеткен оқулық авторлары: В.В. Фаронов, Т.А Павловская, А.Л. Фукс, А.В. Фролов және Г.В. Фролов. Аталған авторлардың материалдық әдістемесі мен баяндау реті осы оқу құралының жазылуына үлкен көмегін тигізді.

Оқу құралын жазу қажеттілігінің бірнеше себептері бар, олардың ішінде ең негізгісі – мамандық, сол мамандыққа арналып әзірленуде.

Оқу құралын жазу үшін қолданылған дәрістер конспекттері «Ақпараттық жүйелер» мамандығына арналған, бірақ оқулық материалын кез келген мамандыққа арналған «Бағдарламалау технологиясы» пәнін оқыту барысында пайдалануға болады.

КІРІСПЕ

Ақпараттық технологиялардың дамуы бағдарламалау тілдерінің жоғары сатыға көтерілуіне тікелей байланысты. Олар, өз кезегінде – ақпараттық технологиялардың дамуында әзірлеушіге жаңа мүмкіндіктер ұсынады. Осы үдерістер өзара байланысты және бір-бірін толықтырады.

Жылдам қарқынмен ақпараттық технологиялардың дамуы, әсіресе, Интернет технологияларының, бағдарламалау технологиясында түбегейлі өзгерістерге деген қажеттілікті туындатты. Ол визуальді бағдарламалау ортасының орнына келген бағдарламалау платформасының пайда болуында көрініс тапты.

Осы жаңа бағдарламалау технологиясына алғашқы болып жаңа бағдарламаларды құруға және орындауға арналған .NET («дот-нет» деп айтылады) платформасы бар .NET Framework өнімін 2000 ж. құрған Microsoft корпорациясы кірісті.

.NET платформасы бірнеше бағдарламалау тілдеріне арналған Visual Studio.NET әзірлеу ортасын ғана емес, сонымен қатар басқа да түрлі құралдарды өзіне қосады. Олар әр түрлі типтегі компьютерлерге бағдарламаларды көшіру мәселелерін және рұқсат етілмеген әрекеттерді болдырмау, әр түрлі бағдарламалау тілдерінде деректер типтерінің үйлесімділік, яғни қауіпсіздік мәселелерін шешуге көмектеседі, т.б.

.NET платформасының құрылымы нақты бағдарламалау тіліне байланысты емес және ол Visual Studio.NET құру ортасының құрамына кіретін кез келген тілде қосымшаны құруға мүмкіндік береді. Сонымен қатар, жобаның жеке бөліктері әр түрлі тілдерде жазылып, бір қосымшаға біріктірілуі мүмкін.

.NET платформасы ашық типтегі құрылым болып табылады – түрлі жаңа тілдерді және бағдарламалау орталарын жеңіл қосуға болады. Мұндай мүмкіндіктер бағдарламалау технологияларының эволюциялық даму жолын іске асыруға көмектеседі.

.NET платформасы нақты бір бағдарламалау тіліне байланысты болмаса да, Visual Studio.NET құру ортасының құрамына жаңа бағдарламалау тілі - C# (си-шарп) енгізілді.

C# тілінің құрастырушылар тобын басқаратын жетекші бағдарламашысы – Borland фирмасында бағдарламалау тілдерін құруды ұзақ уақыт басқарған Хейлсберг болып табылады. Бұл жағдай негізі C, C++ тілдері мен Delphi ортасының визуальді компоненттер технологиясы болатын C# тілінің мүмкіндіктеріне өз таңбасын қалдырды.

C# бағдарламалау тілі толық объекті-бағытталған тіл және .NET платформасы ұсынатын мүмкіндіктерді барынша пайдаланады.

Бұл оқу құралы C# тілі және Visual Studio.NET құру ортасының бағдарламалау технологияларын қарастыратын екі бөлімнен тұрады.

1 C# ТІЛІНЕ КІРІСПЕ

1.1 Алгоритм туралы түсінік

Кез келген бағдарламаны жазу үдерісін қарастырсақ, екі негізгі кезеңді бөліп алуға болады – есепті шешу алгоритмін құру және алгоритмді іске асыру кезеңдері.

Есепті шешу алгоритмін құру кезеңін түсіну үшін алгоритм ұғымын анықтап алу керек.

«Қазіргі кездегі алгоритм сөзінің мағынасы рецепт, үдеріс, әдіс, тәсіл, рәсім, бағдарлама сөздерінің мағынасына өте ұқсас, бірақ «алгоритм» сөзінің қосымша сипаты бар. Алгоритм белгілі бір есепті шығаруда операциялардың орындалу ретін анықтайтын ережелер жиыны ғана емес, оның басқа ең негізгі бес қасиеті бар:» [4, 29 бет].

Алгоритм ұғымының ең негізгі бес қасиеті бар, олардың әрқайсысы оның ерекшелігін сипаттап, әрекеттер ретін анықтайтын басқа сөздердің мағынасынан бөледі.

Алгоритмнің негізгі қасиеттері:

– алгоритмнің аяқталуы;
– анықтылық;
– тиімділік;
– деректерді енгізу;
– деректерді шығару.

Алгоритм қасиеттерін анықтаған кезде авторлардың көпшілігі бірінші үш қасиеттерімен ғана шектеледі. Осы бөлімде біз алгоритмнің барлық бес қасиеттерін қарастыратын боламыз. Алгоритмнің бірінші қасиеті - алгоритмнің аяқталуы, яғни алгоритм қадамдардың соңғы санында аяқталуы тиіс. Сонымен бірге қадамдар саны өте көп болуы мүмкін, бірақ олар аяқталуы тиіс.

«Алгоритмнің аяқталу қасиетінен басқа алгоритмнің барлық қасиеттеріне ие процедураны есептеуіш әдіс деп атауға болады» [4, 30 бет].

Алгоритмнің екінші қасиеті - анықтылық, яғни алгоритмнің әрбір қадамы нақты анықталуы тиіс. Мысалы, рецепте «Бір салым тұзды қосыңыз» [32 бет, 4], нұсқауы саусақтардың өлшемі мен азық-түліктің көлемін ескермейді.

Алгоритмнің үшінші қасиеті - тиімділік, яғни шешім қолданылатын ресурстардың ең аз шығынын жұмсап, тез және дұрыс орындалуы тиіс. Алгоритмнің тиімділігін бағалайтын көптеген әдістемелері бар, оның өзі - тұтас бір ғылым, бірақ, әзірше тек қана анықтамасымен ғана шектелейік.

Деректерді енгізуді ұйымдастыру мен бағдарламаны басқару процесі алгоритмнің қазіргі кездегі ұғымында алгоритмнің жеке қасиеті болып бөлінді. Бұл қасиет көбінесе «Пайдаланушының деректерді енгізу

интерфейсін» анықтайды, яғни бағдарламаны пайдалану барысында деректерді енгізу мүмкіндігі.

Алгоритмнің соңғы қасиеті бағдарлама жұмысының нәтижелерін шығаруды анықтайды. Олар технологиялық процестер бойынша суреттер, кесте, график, кейбір цифрлық немесе аналогты мәндер, апаттық жағдайлардың дыбыстық сигналдары болуы мүмкін, т.б..

1.2 Есепті шешу алгоритмін құру кезеңі

Есепті шешу алгоритмін құру есепті талдаудан басталады, яғни есептің мәнін анық түсіну керек. Есепті талдаудан кейін ғана алгоритмді дайындауды – есепті шешетін ретті қадамдарды құруды бастауға болады.

Есепті шешу алгоритмін құру бойынша соңғы жұмыс – алгоритмді тексеру.

Есепті шешу алгоритмін құру кезеңінде алгоритм айқын болмаса, онда есептің шешімін бір-бірімен байланысқан жеке бағыныңқы есептерге немесе блогтарға бөлу керек. Әрбір блог үзінділерге бөлінуі мүмкін және ол әрбір үзіндінің орындау алгоритмі айқындалғанға дейін қайталанады. Есепті шешу кезінде жеке үзінділердің алгоритмдері есептің жалпы шешімін сипаттайтын бір алгоритмге біріктіріледі.

Алгоритмді құрудың бұл кезеңі ең маңызды болып табылады, өйткені осы кезеңде есепті шешудің барлық мүмкін нұсқалары қарастырылуы керек. Әсіресе ол шешмнің алгоритмі анық емес, ауқымды немесе логикалық күрделі есептерді шешу кезінде керек.

Қарапайым, бағдарламаларды құру кезінде есепті шешу алгоритмі интуитивті түрде анықталады немесе сіз оның алгоритм екенін білмей-ақ «есіңізде ұстап жүресіз». Осындай есептерді қарастырған кезде біз оның орындалуын ғана қарастырамыз.

Алгоритмді құру кезінде есеп шешімінің түрлі нұсқалары болуы мүмкін, сондықтан шешімнің ең дұрыс нұсқасын таңдау үшін «алгоритмнің талдауын» жүргізе білу керек [33 бет, 4] .

Алгоритмді құру нәтижесі болып алгоритмнің сипаттамасы немесе оның құрылымдық схемасы есептеледі.

Алгоритм жұмысының дұрыстығын тексеретін бірнеше әдістемелер бар, мысалы, деректердің белгіленген мәндері беріледі және есептің шешімі қолмен қаралады.

Алгоритмдерді тексерудің ең жақсы нұсқасы - әлдебір бағдарламалау тілінің көмегімен орындау, яғни дайын алгоритмді бағдарламалау тілінің бағдарламалық коды арқылы сипаттау (бағдарлама мәтіні) және оның жұмысын компьютерде тексеру. Алгоритмді құру кезеңіне көшейік.

1.3 Алгоритмді жүзеге асыру кезеңі

Алгоритмді құру кезеңі мыналардан тұрады

– бағдарламалау тілдерінің бірінде құрылған алгоритм бойынша бағдарлама кодын жазу;

– бағдарламаны тестілеу – компьютерде бағдарламаны іске қосу және оның жұмысының нәтижелерін белгілі бір бақылау мәліметтері бойынша немесе логикалық түрде тексеру.

Алгоритмді құру кезеңдерін орындау үшін кем дегенде бір бағдарламалау тілін білу керек.

Алгоритмдеу негіздерін білмей бағдарламалауды үйрену мүмкін емес және алгоритмдеу негіздерін үйрену бағдарламалаусыз мүмкін емес.

Барлық бағдарламалаудың алгоритм тілдері, әдетте, құрылымдық немесе процедуралық бағдарламалау технологиясын қолдануға негізделген.

Қазіргі кезде объекті-бағытталған бағдарламалау (ОББ) технологиясы негізге алынып отыр, ол үшін арнайы визуалды бағдарламалау орталары дайындалған, мысалы, Delphi, VISUAL C++, түрлі VISUAL STUDIO және т.б..

Бұл оқу құралында Visual Studio.NET визуалды бағдарламалау ортасының объекті-бағытталған C# тілін қолданып бағдарламалау сұрақтары жан-жақты қарастырылған.

Оқу құралының бірінші бөлімінде C# тілінде бағдарламалаудың негіздері қарастырылған, онсыз Visual Studio.NET визуалды бағдарламалау ортасы ұсынатын бағдарламалау технологиялары мен объекті-бағытталған бағдарламалау технологияларын меңгеру мүмкін емес.

Әлбетте, бағдарламалау тілін меңгерудің бастапқы кезеңінде алгоритмдеудің кейбір сұрақтарын жан-жақты қарастырамыз, өйткені есеп шешімінің алгоритмдерін құру үдерісі мен бағдарлама кодын жазу бір-бірімен байланысты. Алайда, бұдан былай есеп шешімі алгоритмінің идеясы мен бағдарламалық жүзеге асырылуы ғана қарастырылады.

1.4 .NET платформасының құрылымы

.NET платформасының идеясы - кез келген тілде жазылатын қосымшаларды дайындауға және орындауға арналған бірыңғай жүйені жасау.

Қосымшаларды жазу үшін NET платформасына бірнеше бағдарламалау тілдеріне арналған, Visual Studio.NET деп аталатын құру ортасы қосылды. Оның ішінде қосымша жобасының кодын енгізіп, түзету жасауға арналған мәтіндік редактор және жобаны жөндеу мен іске қосу құралдары, анықтама жүйесі, т.б. элементтері бар.

Әр түрлі бағдарламалау тілдері бойынша деректер типтерінің үйлесімділігі үшін .NET платформасы бағдарламалау тілінің әрбіріне типтердің ортақ жүйесін (Common Type System – CTS) – компьютер жадысында деректерді сақтаудың бірыңғай түрін талап етеді.

Қосымшалардың түрлі типті компьютерлерге тасымалдауды қамтамасыз ету үшін .NET платформасында біріңғай аралық компиляция тілі (Common Intermediate Language – CIL) қарастырылған. Оған платформаның кез келген тілінде жазылған қосымшалар түрлендіріледі.

Осы тілдің командалары нақты операциялық жүйеге, компьютер типіне, қосымша кодына тәуелді емес. CIL тіліндегі бағдарлама өз бетімен орындалмайды, ол кез келген компьютерге, кез келген операциялық жүйеге орнатуға болатын, жалпытілді орындау ортасы (Common Language Runtime,– CLR) деп аталатын жүйенің бақылауымен орындалады. Жалпытілді орындау ортасының CIL тіліндегі кодты нақты бір процессордың машиналық командасына аударатын JIT компиляторы бар.

JIT компилятордың атауы өз жұмысының принципін сипаттайды, яғни қосымшаның осы сәтте орындауды қажет ететін бөлігін ғана компиляциялау (just in time – дер кезінде).

.NET платформасында қауіпсіздікті қамтамасыз ету үшін жүйелік көзқарас қолданылады - қосымша компиляциясы кезеңінде exe немесе dll кеңейтуі бар арнайы файл құрылады, яғни CIL тілінде коды және метадеректері бар құрылым. Пайдаланушы компьютерінде қауіпсіздік пен қосымшаны орнату мен өрбітудің жеңілдігін қамтамасыз ететін метадеректер құрастырудың аты мен нұсқасын, қосымшада қолданылатын объект туралы мәліметтер мен деректер типін, құрастыру тәуелді файлдар т.б. өзіне қосады.

Кез келген .NET. тілдерінде бағдарламалауда қолдануға болатын .NET платформасының үлкен кітапхана класы (Framework Class Library – қысқаша .NET Framework) бар.

Материал мазмұнында қосымша, жоба, бағдарлама терминдері жиі қолданылады. «Қосымша» терминін «бағдарлама» терминінің синонимі сияқты қабылдануы мүмкін. «Консоль» үшін құрылатын қосымшаларды бағдарламалар деп атаймыз. «Windows» (Windows- қосымшалар) үшін құрылатын бағдарламаларды қосымшалар деп атаймыз.

Зерттеу сатысындағы қосымша жоба деп аталады.

1.5 Visual Studio.NET ортасы

Кез келген визуалды бағдарламалау ортасы сияқты Visual Studio.NET құру ортасы .NET үйлесімді тілдер қолданатын қосымшаларды жазу, түзету, компиляциялау, дұрыстау және іске қосу құралдарын

ұсынады..NET платформасының Visual Studio.NET құру ортасы төрт тілмен жұмыс істеуге арналған: C#, VB.NET, C++ және J#, бірақ қазіргі уақытта басқа да бағдарламалау тілдерін .NET платформасына қосу үшін құрулар белгілі.

Visual Studio.NET ортасы әр түрлі типті жобаларды жасауға мүмкіндік береді, мысалы:

– Windows-қосымшалар Windows интерфейсінің дәстүрлі басқару элементтерін қолданады;

– консольді қосымша шығаруды командалық процессор терезесінде орындайды;

– басқа қосымшаларда пайдалану үшін қолданылатын кластар кітапханасы (Dll модульдері) ;

– веб-қосымшалар, яғни әдістерін Интернет арқылы шақыруға болатын қосымшалар, т.б.

Visual Studio.NET ортасы технологиясының басым бөлігі Windows және веб-қосымшаларды құруға арналған, бірақ әзірлеушілер консольді қосымшалармен жұмысты да ескерген.

Консольді қосымшамен жұмысты орындау барысында пайдаланушы Windows операциялық жүйесінің командалық жолының терезесіне немесе Турбо Паскаль ортасының мәтіндік тәртіптегі терезесіне ұқсас мәтіндік терезеде жұмыс істейді.

«Консольді қосымшалар тілді оқу үшін ең қолайлы болып келеді, өйткені оларда графикалық интерфейсті дайындау үшін көптеген стандартты объекттер қолданылмайды» [Павл. 15 бет].

Сондықтан Оқу құралының бірінші бөлімінде деректерді өңдеудің түрлі алгоритмдерін зерттеуге көбірек уақыт бөлу мақсатында және C# тілінің базалық қасиеттерін қарастыру үшін біз консольді косымшаларды ғана құраймыз.

Оқу құралының екінші бөлімінде Windows-қосымшаларын және объекті-бағытталған бағдарламалау негіздерін қолданып, бағдарламалау технологиясы қарастырылатын болады.

Visual Studio.NET ортасында жұмыс жасау үшін жобаларды жасау, оларды магнитті дискте сақтау, жобаны іске қосу, т.б. бойынша білім керек, сондықтан C# тілін үйренуді Visual Studio.NET ортасын оқудан бастайық.

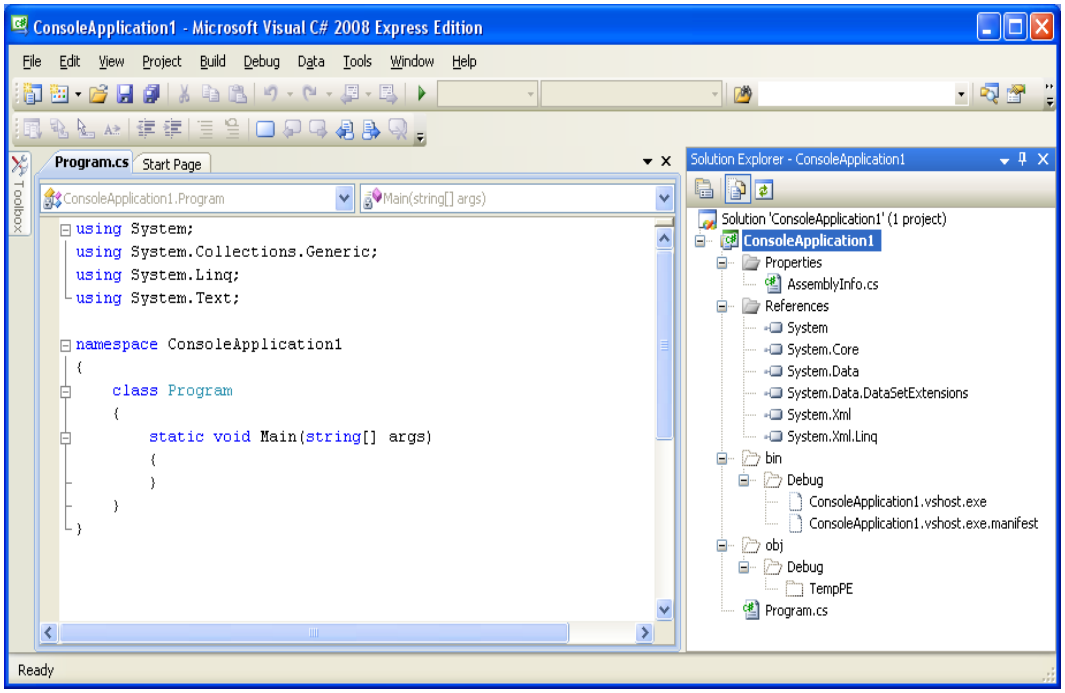
1.6 Жобаны жасау. Ортаның негізгі терезелері

Жобаны жасау үшін Visual Studio.NET ортасы іске қосылғаннан кейін, басты менюде File ->New -> Project командасын таңдау керек.

Ашылған диалогтық терезенің сол жағында Visual C# Windows тармағын, ал оң жағында Console Application тармағын таңдау керек.

Name өрісіне жоба атын жазуға болады, бірақ жобаның берілген атын қалдыруды ұсынамыз. Жобаның орналасу орны ретінде Location өрісінде (жобаның дискіде сақтау орны) компьютердің жұмыс орнын жазыңыз.

Берілген мәндер расталғаннан кейін ортада жоба берілген атаумен құрылады. Бағдарлама кодының үлгісі 1.1-суретінде көрсетілген.



1.1-сурет – Консольді қосымша жобасы құрылғанынан кейінгі Microsoft Visual Studio ортасының үлгісі

1.1-суреттің жоғарғы бөлігінде бас меню (File, Edit, View режимдерімен, т.б.) және орта режимдерінің командаларын жылдам таңдауға арналған құрал-саймандар панелі орналасқан.

Экранның жоғарғы оң жағында жобаны басқару терезесі - Solution Explorer орналасқан (егер ол көрінбей тұрса, бас менюдің View -> Solution Explorer командасын пайдалану керек).

Терезеде жобаға қатысты барлық ресурстар жазылған, мысалы, System, System.Data, System.Xml атаулар кеңістігіне сілтемелер және т.б.

Экранның негізгі бөлігін редактор терезесі алып тұр, онда ортада автоматты түрде құрылған бағдарлама коды (мәтін) орналасқан. Бағдарлама коды үлгі ретінде құрылады, оған Сіз керекті жағдайда өз кодыңызды жаза аласыз.

Бағдарлама коды көрсетілген кезде оның әртүрлі құрамдас бөліктерінің түрлі-түсті түстері қолданылады. Қызметтік сөздерге (кодтың резервте сақтаулы сөздері) көк түс арналған. Түсініктеме сұр немесе қара жасыл түсте, ал қалған мәтін қара түсте көрсетіледі.

Мәтіннің сол жағында құрылым символдары орналасқан, ішінде минус немесе плюсы бар вертикаль сызықтар мен квадраттар қосылған. Егер минусы бар кез келген квадратты басса, онда сәйкес код фрагментін (блогын) «жинауға» болады, бұл ретте минус плюске ауысады. Егер плюс квадраты басылса, онда сәйкес код фрагменті (блогы) экранның сәйкес бөлігіне «жайылады». Ақпаратты көрсетудің осындай құрылымдары Windows жүйесінің бумаларын көрсету кезінде қолданылады және бағдарлама кодының керекті фрагментіне назар аударуға мүмкіндік береді.

Магнитті дискіде жобаны жазу үшін File->Save All команда тізбегін қолдану керек (көптеген дискеттер салынған құрал-саймандар панелінде). Редакцияланатын файлда өзгерістерді ғана сақтау үшін File->Save Program.cs командасын таңдауға немесе аспаптық панелінде бір дискета салынған батырманы басу керек.

Жобаны ашу үшін келесі File->Open Project команда тізбегін қолдану керек.

1.7 C# тілінің символикасы

C# бағдарламалау тілі жаңа .NET технологияларын қолдайтын объекті-бағытталған бағдарламалау тілі болып табылады. Кез келген тілде сияқты C# тілінде алфавит– символдар жиыны бар, олардың көмегімен бағдарлама коды (мәтін) жазылады. C# тілінде бағдарламаны жазу үшін келесі символдар қолданылады:

– кез келген әріптерді, C# тілі бас және кіші әріптерді ажыратады;
– 0 - 9 дейінгі цифрлар;
– қызметтік белгілер;
– қызметтік сөздер;
– бағдарлама мәтіні бойынша түсініктемелер.

Әріптер мен цифрлар бойынша барлығы түсінікті, ал қызметтік белгілер мен сөздерді (олардың саны өте көп) тілді оқу барысында меңгереміз.

Бағдарлама кодын жазу барысында оның әр түрлі конструкциясының (операторлар, функциялар, объекттер, жазулар, айнымалылар, т.б.) аттары, яғни идентификаторлары анықталуы тиіс. Олар әріптерден (латын алфавитін қолдану қажет), цифрлардан (0 – 9), астын сызу ‘_’ символынан тұруы мүмкін, бірақ идентификатордың бірінші символы әріп немесе астын сызу символы болуы керек, сан болмауы тиіс.

Мысалы:

– дұрыс – `A_9, B34, TYTI, Max;`
– дұрыс емес – `A-9, B 34, 13FIG.`

Егер қызметтік сөздердің алдында ‘@’ символы тұрса, оны идентификатор ретінде де қолдануға болады. Мысалы , @for, бірақ

бағдарламалауда қызметтік сөздерді қолдану жағымсыз әдеттің нышаны деп есептеледі. ‘@’ символын қызметтік сөздерден өзге сөздермен бірге қолдануға болмайды.

1.8 Айнымалыларды сипаттау

Бағдарламалау тілінің «айнымалы» ұғымының алуан түрлі анықтамасы бар, бірақ компьютердің аппараттық қамтамасыз ету тұрғысынан ең дұрыс анықтамасы мынандай – бағдарламаны орындау уақытының әр түрлі мезетінде түрлі мәндерді болуы мүмкін компьютер жадысының аймағы. Осы жадының аймағына сәйкес қолданылатын символикалық ат айнымалы аты немесе айнымалы идентификаторы деп аталады.

Айнымалы - идентификатормен белгіленген компьютер жадысының аймағы, онда бағдарламаның жұмысы кезінде өзгеретін деректер сақталады.

Бағдарламада айнымалыны қолданбас бұрын оны жариялау керек – оның атын және типін көрсету керек.

Айнымалы типі оған қандай мәндерді жазуға болатынын көрсетеді – бүтін, нақты, символдық, т.б.

Айнымалы аты идентификатор болып келеді.

Мысалы:

```
int TYT;
float cym, kol;
```

Мұнда, `TYT` айнымалысы – бүтін, ал `cym`, `kol` айнымалылары – нақты (бөлшек) типте екені көрсетілген.

Бір типті айнымалылар нүктелі үтірмен аяқталуы тиіс.

Айнымалыны жариялау кезінде жариялау операторы қолданылады - нүктелі үтірмен аяқталатын әрекет. Бірақ әдебиеттерде бұл амалды жариялау операторы деп атаудың орнына жай ғана бағдарлама айнымалысын жариялау деп атайды.

Айнымалыны бірінші рет қолдану жариялау болып есептелетін BASIC бағдарламалау тілінен немесе айнымалылары бағдарлама басында, арнайы бөлімде жарияланатын PASCAL бағдарламалау тілінен өзгеше **C#** бағдарламалау тілінде айнымалы бағдарламаның кез келген орнында жариялана алады. Сонымен қатар айнымалыны PASCAL тіліне ұқсас бағдарламаның басында немесе BASIC тіліне ұқсас бірінші рет қолдану барысында жариялауға болады.

1.9 C# тілінің деректер типі

.NET бағдарламалау технологиясының жаңа бір міндеті CTS (Common Type System) – жалпы типтер жүйесін қолдану болып табылады.

Ол осы технологиямен жұмыс істейтін кез келген бағдарламау тілі үшін компьютер жадында деректер көрсетімін стандарттауға мүмкіндік береді. Шартты түрде барлық CTS деректер типі мәнді (бүтін, нақты, т.б.) және сілтемелік (массивтер, кластар, т.б.).болып бөлінеді. Компьютер жадысы мәнді типтегі айнымалыларға бағдарлама компиляциясы кезінде, ал сілтемелік типтегі айнымалыларға `new` операторының көмегімен бағдарлама орындалу барысында бөлінеді.

.NET технологиясында деректер типін белгілеу күрделі иерархиялық құрылымға ие, мысалы, System.Int32. Сондықтан C# тілінде кейбір жиі қолданылатын типтердің жазуын жеңілдету үшін қарапайым типтер ұғымы енгізілді. Қарапайым типтер бұл – кейбір мәнді және сілтемелік типтердің қысқартылып жазылған түрі. 1.1-кестесінде C# тілінің қарапайым типтері және осы типтерге сәйкес CTS типтері көрсетілген.

1.1-кестесі – C# тілінің қарапайым типтері

C# тілінің қарапайым типтері	.NET платформасының CTS типтері
byte	System.Byte
sbyte	System.SByte
short	System.Int16
int	System.Int32
long	System.Int64
usort	System.UInt16
uint	System.UInt32
ulong	System.UInt64
float	System.Single
double	System.Double
object	System.Object
char	System.Char
string	System.String
decimal	System.Decimal
bool	System.Boolean

Бағдарламада деректер типін таңдауда есептеу дәлдігінің талаптары және компьютердің айнымалыларға бөлінетін жады көлемі ескеріледі.

Бағдарламалауды меңгерудің бірінші кезеңінде біз тек қана қарапайым типтерді ғана қолданамыз, ал қалған басқа деректер типтерін материалды меңгеру барысында қарастырамыз.

1.10 C# тілінің тұрақтылары

C# тілінде тұрақтыларды жариялау айнымалыларды жариялаумен бірдей, бірақ. `const` қызметтік сөзі қосылып жазылады.

Мысалы:

```
const char CIMV = 'y';
const int MAX = 640;
```

Тұрақты бұл – идентификатормен белгіленетін, бағдарлама жұмысының барысында өзгермейтін деректердің мәні сақталатын компьютер жадысының аймағы.

Тұрақтыны жариялаған кезде оның типі ғана емес, сонымен бірге оның мәні де көрсетілуі керек.

1.11 C# тілінің атаулар кеңістігінің ұғымы

C# тіліндегі кез келген бағдарлама `using` операторының көмегімен бағдарлама кодына кейбір атаулар кеңістігін көрсетуден басталады. Мысалы:

```
using System;
using System.Windows.Forms; и т.д.
```

Бұл ретте бір `using` операторына тек бір атаулар кеңістігі тиісті бола алады.

Әрбір атаулар кеңістігі .NET платформасына тиісті кластардың белгілі бір тобына сәйкестенеді (әрбір класс - белгілі бір тип). Сонымен, .NET платформасының барлық типтер жиынтығы (CTS-те 4000 аса түрлі типтер белгілі) өзінің функционалды міндеті бойынша логикалық байланысқан топтарға біріктірілді, олар атаулар кеңістігі деп аталады. Кейбір міндеттері шешу үшін кейбір кластар, әдістер, функциялар немесе деректер керек болса, онда сіз бағдарламаға тиісті атаулар кеңістігін қосуыңыз керек**.** Сонымен, **C#** тілінде NET платформасының бұрында жазылған кластар кітапханасын қолдану мүмкіндігі орындалады.

Ескере кететін бір жағдай, **C#** тілі бойынша оқулықтардың көптеген авторлары кітапхана терминін қолданбай, атаулар кеңістігі ұғымымен шектеледі. 1.2-кестесінде ең жиі қолданылатын атаулар кеңістігі көрсетілген

Бағдарламалауға жаңадан келген бағдарламашылардың келелі мәселелерінің бірі - керекті атаулар кеңістігінің атын және оны қосу жолдарын анықтау. Атаулар кеңістігімен жұмыс жасау технологиясын C# тілін менгеру барысына қарай қарастырамыз.

Айта кететін жәйт, түрлі Visual Studio орталарының «қосымшаларды дайындау шеберлері» керекті атаулар кеңістігін автоматты түрде таңдап алады, қажеттілік болмаса оларды өзгертпеген дұрыс. Мысалы, егер Visual Studio 2008 ортасында консольде орындалатын қосымша (Fail->New->Project->Console Application) таңдалса, онда қосымшаны дайындау шебері 1.1-суретінде көрсетілген атаулар кеңістігін автоматты түрде қосады.

1.2-кестесі – **.**NET платформасының кейбір атаулар кеңістігі

.NET платформасының кейбір атаулар кеңістігі	Тағайындалуы
System	Object класы бар түпкі атаулар кеңістігі және қарапайым типтегі деректермен, математикалық функция жинағымен, деректерді енгізу–шығарумен, қоқысты жинау операциясымен жұмыс жасауға арналған кластар жиынтығы, т.б.
System.Data System.data.SqlClient және т.б..	Бұл атаулар кеңістігі деректер базасымен жұмыс жасауға арналған
System.IO	Бұл атаулар кеңістігі файлға деректерді енгізу–шығаруға жауап береді, т.б.
System.Drawing System.Drawing.Drawing2D	Бұл атаулар кеңістігінің кластарында графикалық қарапайым құралдары, қаріптер жиыны, сызықтар түрлері, монитор экранында графикалық ақпаратты көрсету құралдарының жиындары бар.
System.Net	Кластар жиыны желілер бойынша деректерді табыстауға жауап береді.
System. Security	Кластар жиыны желілер бойынша деректерді табыстау қауіпсіздігін артыру үшін қолданылады.
System.Web	Кластар жиыны web-қосымшаларда жұмыс жасауға арналады.
System.Windows.Form	Бұл атаулар кеңістігінің кластары Windows интерфейсінің элементтерімен - терезелер, батырмалар, басқа да басқару элементтерімен жұмыс жасауға арналған .

Бағдарламаның атаулар кеңістігін анықтағаннан кейін Visual Studio 2008 ортасында консольді қосымшаны дайындау шебері `namespace Console Application { . . .}` арнайы нұсқауымен бағдарлама кодының аумағын анықтады, онда құрылған қосымшаның деректер типін қолдануға болады.

Айта кететін жәйт, ортаның бас терезесінің сыртқы көрінісі терезені күйге келтіруге байланысты, оны бағдарламашы «өзіне керекті нұсқада» орындайды. Ортаны меңгеру кезеңінде күйге келтіру сипаттамаларын өзгертпеуді ұсынамыз.

1.12 Өзін-өзі тексеру сұрақтары

1 Алгоритм ұғымы.

2 Есептерді шешу алгоритмін құру кезеңдері неден тұрады?

3 Алгоритмді құру қандай кезеңдерден тұрады?

4. NET платформасы не үшін арналған?

5 Visual Studio.NET қосымшаларын дайындау ортасы дегеніміз ...

6 Құрастыру (сборка) файлы неден тұрады?

7 Түрлі тілдерде жазылған бағдарламаларды тасымалдау .NET платформасында қалай қамтамасыз етіледі?

8 .NET платформасының .NET Framework неден тұрады?

9 .NET платформасының CTS неден тұрады?

10 C# тілінде айнымалы ұғымы

2 C# ТІЛІНІҢ ҚАРАПАЙЫМ ОПЕРАТОРЛАРЫ

2.1 C# ТІЛІНДЕГІ БАҒДАРЛАМА ҚҰРЫЛЫМЫ

Есепті шешуге арналған атаулар кеңістігін анықтап алғаннан кейін бағдарлама класын сипаттау басталады.

Класс дегеніміз бұл деректерді (өрістерді) және функцияларды (әдістерді) біріктіретін, осы деректерді біртұтас өңдейтін – код үзіндісі (тип).

Анықтама бойынша кез келген бағдарлама деректер мен оларды өңдеу әдістері бар код болып есептеледі. Сондықтан C# тілінде түрлі есептерді шешетін бағдарлама кодтары бағдарлама кластарына орналастырылады. Класс сипаттамасы `class` қызметтік сөзінен кейін басталады, қызметтік сөзден кейін класс атауы, одан кейін фигуралық жақшаның ішіне деректер мен оларды өңдеу әдістері орналастырылады.

C# тіліндегі бағдарлама класының сипаттамасының ішінде `Main()` атауы бар әдіс орналасуы керек. Осы әдістен бастап бағдарламаның орындалуы басталады. Назар аударыңыз, C# тілінде бас және кіші әріптер ажыратылады.

Егер есеп қарапайым болса, онда барлық әдістер `Main()` әдісінде ғана жазыла алады. Егер есепті бөлшектеу қажет болса (яғни басқа кластармен немесе әдістермен орындалатын бөлек үзінділерге бөлу), онда бағдарлама класында осы үзінділерді тиісті түрде сипаттау керек, ал `Main()` әдісінде оларды іске қосу ретін көрсетуіңіз керек.

2.2 Бағдарлама мысалы

Бағдарлама кодын дайындау үшін компьютерде C# бағдарламау тілінің компиляторын орнату керек. Қазіргі уақытта C# тілінің бірнеше компиляторлары белгілі, мысалы, Visual Studio 2008 визуалды бағдарламалау ортасы, Turbo C# Explorer, т.б.

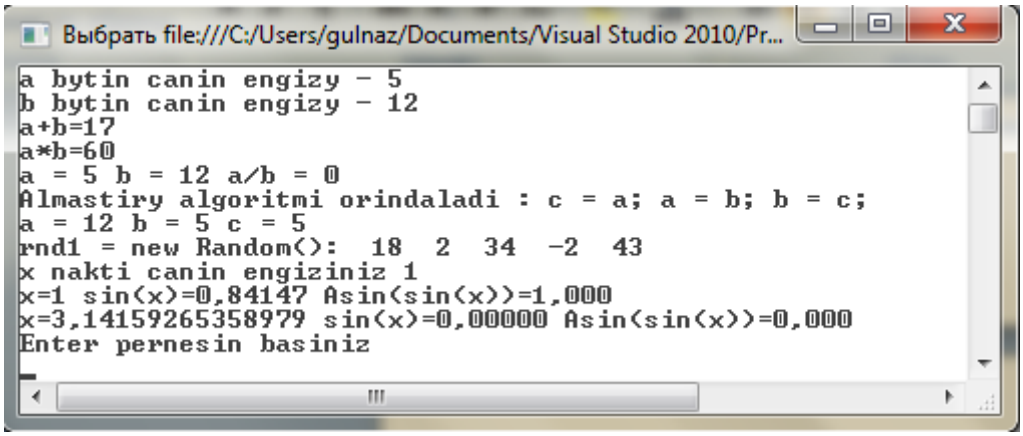
Әрбір компилятордың бағдарламаны дайындаған кезде ескеретін өз ерекшеліктері бар. Оқу құралында Visual Studio 2008 визуалды бағдарламалау ортасында C# компиляторын қолдануға бағытталған.

Бағдарлама кодын қарастырайық. Бағдарлама `a` және `b` бүтін айнымалылардың мәндерін енгізуге (диалогтық тәртіпте) және осы айнымалыларды пайдаланып, арифметикалық амалдарды орындауға мүмкіндік береді. Бағдарламада толық сипаттамасы оқу құралының келесі бөлімдерінде қарастырылатын кейбір операторлар бар.

```
using System;
namespace ConsoleApplication1
{
```

```
      class Program
      {
      static void Main(string[] args)
      {
       int a, b, c;
       double x, y, z;
       string buf;
  Console.Write("a bytin canin engizy - ");
       buf = Console.ReadLine();
       a = Convert.ToInt32(buf);
  Console.Write("b bytin canin engizy - ");
       buf = Console.ReadLine();
       b = Convert.ToInt32(buf);
       c = a + b;
       Console.WriteLine("a+b={0}", c);
       c = a * b;
       Console.WriteLine("a*b={0}", c);
       c = a / b;
       Console.WriteLine("a = {0} b = {1} a/b = {2}", a, b, c);
       Console.WriteLine("Almastiry algoritmi orindaladi : c =
a; a = b; b = c;");
       c = a; a = b; b = c;
       Console.WriteLine("a = {0} b = {1} c = {2}", a, b, c);
       Random rnd = new Random();
       Console.Write("rnd1 = new Random():");
       for (int i = 1; i <= 5; i++)
       {
       a = rnd.Next() % 101 - 50;
       Console.Write(" " + a.ToString());
       }
       Console.WriteLine();
       Console.Write("x nakti canin engiziniz ");
       buf = Console.ReadLine();
       x = Convert.ToDouble(buf);
       y = Math.Sin(x);
       z = Math.Asin(y);
       Console.WriteLine("x={0}                    sin(x)={1:F5}
Asin(sin(x))={2:F3}", x, y, z);
       x = Math.PI;
       y = Math.Sin(x);
       z = Math.Asin(y);
       Console.WriteLine("x={0}                    sin(x)={1:F5}
Asin(sin(x))={2:F3}", x, y, z);
       // Задержка рабочего экрана монитора
       Console.WriteLine("Enter pernesin basiniz");
       Console.ReadLine();
      }
      }
     }
```

1.2-суретінде бағдарлама жұмысы көрсетілген.



1.2-суреті– Бағдарлама жұмысының нәтижесі «1-мысал»

Қалған мысалдарда бағдарлама жұмысын көрсету жұмыс терезесінің көшірмесімен ауысады, мысалы:

```
      a bytin canin engizy - 5
      b bytin canin engizy - 12
      a+b=17
      a*b=60
      a = 5 b = 12 a/b = 0
      Almastiry algoritmi orindaladi : c = a; a = b;
b = c;
      a = 12 b = 5 c = 5
      rnd1 = new Random(): 18 2 34 -2 43
      x nakti canin engiziniz 1
      x=1 sin(x)=0,84147 Asin(sin(x))=1,000
      x=3,14159265358979                  sin(x)=0,00000
Asin(sin(x))=0,000
   Enter pernesin basiniz
```

`Main()` әдісі `public` және `static` деген екі модификаторымен анықталуы мүмкін (көбінесе қол жеткізу спецификаторы деп аталады).

`public` - ашық модификаторы әдістің бағдарлама ішінде немесе сыртында басқа әдістерге қол жеткізімді екенін көрсетеді.

`Main()` әдісі `Program` класында орналасады және әдетте кластың әдістеріне класс типті айнымалыларды, яғни объектті құрғаннан кейін ғана қол жеткізуге мүмкіндік бар. Бірақ, егер әдіс `static` модификаторымен жарияланса, онда оны «класс деңгейінде» қолдануға болады, яғни класс объектісін құрмай.

`Main()` әдісінің дөңгелек жақшаларының ішінде жолдар массивы түрінде берілген кіріс параметрлері анықталған, олар арқылы нұсқаулар бағдарламаның командалық жолынан әдіс іске қосылғанда жеткізіле алады. Біздің бағдарламада бұл параметрлер қолданылмайды, сондықтан

оларға назар аудармауға немесе өшіріп тастауға болады және `Main()` әдісінің қысқартылған түрде жазылуы қалтырылады.

Бағдарламада монитор экранындағы кідіріс бойынша түсініктеме берілген. Бағдарлама түсініктемесін `//` белгісімен бір жолға немесе `/* . . . */` белгілері арқылы бірнеше жолдарға жазуға болады.

`Console.ReadLine();` нұсқауы (бағдарлама соңында) бағдарлама жұмысының нәтижесін көру үшін «бағдарламаның жұмыс терезесін тоқтату» амалын орындайды, сонымен қатар `Console.WriteLine("Для продолжения нажмите клавишу Enter");` нұсқауы «Для продолжения нажмите клавишу Enter» хабарламасын шығарады.

Қарастырылып отырған мысалды `Main()` функциясы үш негізгі әрекетті орындайды:

– монитор экранына әзіржауапты және бағдарлама жұмысының нәтижесін шығару;

– диалогтық режімде перне арқылы айнымалылардың мәндерін енгізу;

– кейбір арифметикалық өрнектерді есептеу және `c, x, y, z` айнымалыларына есептеулер нәтижелерін меншіктеу.

Әрбір әрекетті жеке қарастырайық.

2.3 Монитор экранына ақпаратты шығару

C# тілінде консольді қосымшалармен жұмыс жасау барысында монитор экранына шығару `Console.WriteLine()` және `Console.Write()` статикалық әдістері арқылы орындалады (дәлірек айтқанда `Console` класының `WriteLine()` және `Write()` статикалық әдістері).

Ақпаратты шығарғаннан кейін `Console.WriteLine()` әдісі терезе курсорын келесі жолдың басына көшіруді орындайды.

Ақпаратты шығарғаннан кейін `Console.Write()` әдісі көрсетілген соңғы символдан кейінгі орында терезе курсорын бірден қалдырады.

Ескеру керек, екі әдіс те монитор экранына ақпаратты символды түрде ғана шығара алады, яғни бүтін немесе нақты типтегі мәндерді монитор экранына шығармас бұрын жолдық типке түрлендіру керек. Айнымалылардың сандық мәнін жолдық мәнге түрлендіру осы әдістердің әрқайсысында орындаған, сонымен қатар түрлендіруді берілген форматта жүргізуге болады.

Әдетте әдістердің бірінші параметрі арқылы түсініктеме мәтіні беріледі, онда фигуралы жақша арқылы «толтырғыш» жазылады. «Толтырғыштар» санына шек жоқ және есептің талаптарымен ғана

анықталады. Барлық «толтырғыштар» нөлден бастап нөмірленеді. Толтырғыштың нөмірінен кейін қос нүкте арқылы шығару форматының спецификаторын көрсетуге болады.

Шығару бойынша әдістерде бірінші параметрден кейін және формат арқылы айнымалылардың атаулары көрсетіледі. Олардың мәндері шығарылатын мәтінде «толтырғыштардың» орнына тиісті форматта орналастырылады. Әрине, айнымалылыр мен «толтырғыштардың» саны бірдей болуы керек. Мысалы,

```
Console.WriteLine("a = {0} b = {1} a/b = {2}", a, b, c);
Console.WriteLine("x={0}  sin(x)={1:F5}  Asin(x)={2:F3}",x,  y,
z);
```

Бірінші жағдайда шығарылатын мәтінде `{0}, {1}` және `{2}` орнына `a, b` және `c` айнымалыларының мәндері пішімдеусіз жазылады. Екінші жағдайда `sin(x)` және `Asin(x)` өрнектерін шығару сандарды пішімдейтін `F` спецификаторы арқылы анықталады және ол үтірден кейін 5 және 3 дейінгі дәлдікпен сәйкесінше орындалады. Кейбір пішімдеу спецификаторлары 2.1 кестесінде көрсетілген.

2.1- кестесі – Сандарды пішімдеу спецификаторлары

Спецификатор	Қызметі
C немесе c	Сандарды ақшалай пішімде шығару үшін қолданылады
D немесе d	Ондық сандарды шығару үшін қолданылады. Символдан кейін үтірден кейін шығарылатын символдар санын көрсетуге болады.
E немесе e	Сандарды экспоненциалдық пішімде шығару үшін қолданылады
F немесе f	Нақты санды «белгіленген нүктесімен» шығарады. Символдан кейін үтірден кейін шығарылатын символдар санын көрсетуге болады.
G немесе g	Ортақ (general) пішім. Шығарылатын санның өлшеміне байланысты F немесе E пішімі қолданады.
N немесе n	Санды шығаруда мыңдықты бөлектеу үшін қолданады. Бөлектегіш ретінде үтір қолданылады.
X немесе x	Бүтін санды он алтылық пішімде шығарады. Егер пішім үлкен символмен анықталса, онда он алтылық пішімдегі барлық әріптер үлкен болады.

Монитор экранына ақпаратты шығарудың қалған жолдары материалды ары қарай оқу барысында қарастырылады.

2.4 Пернетақтадан деректерді енгізу

C# тілінде пернетақтадан деректерді енгізу үшін (диалог режимі немесе интерактивті режим) консольді, статистикалық `Console.ReadLine()` және `Console.Read()` әдістері қолданылады. Осы әдістер бағдарламаның орындалуын тоқтатады және компьютер пернетақтасынан деректерді енгізуді күтеді. Ескеру керек, `Console.ReadLine()` әдісі `string` типті айнымалыны, ал `Console.Read()` әдісі `int` типті айнымалыны қайтарады. Екі әдісте де жаңғырық функциясы бар, енгізілген ақпаратты монитор экранына қайталайды.

Пернетақтадан енгізілген ақпарат жадының арнайы аймағына - буферіне жазылады. Буферге енгізудің аяқталуы Enter пернесін (клавиша) басу арқылы орындалады (сонымен бірге осы перне кодтары буферге жазылады).

`Console.ReadLine()` әдісінің нәтижесі жолдық айнымалының мәні болады (осы жағдайда buf айнымалысы). Енгізу аяқталғаннан кейін әдіс жады буферінің ішіндегісін өшіреді.

`Console.Read()`әдісі Консольді енгізуге бөлінетін жады буферінен тек бір ғана символды алады және оны бүтін санға түрлендіреді. Енгізу аяқталғаннан кейін әдіс жады буферінің ішіндегісін өшірмейді. Мысалы, бірінші бағдарлама үзіндісі.

```
Console.Write("a bytin canin engizy - ");
buf = Console.ReadLine();
a = Convert.ToInt32(buf);
Console.Write("b bytin canin engizy - ");
buf = Console.ReadLine();
b = Convert.ToInt32(buf);
c = a + b;
Console.WriteLine("a+b={0}", c);
```

мына кодқа ауыстырылса:

```
Console.Write("a bytin canin engizy - ");
a = Console.Read();
Console.Write("b bytin canin engizy - ");
buf = Console.ReadLine();
b = Convert.ToInt32(buf);
c = a + b;
```

«a bytin canin engizy - » сөйлемінен кейін 47 саны енгізілсе, `a` айнымалысына 52 мәні меншіктеледі (символдың коды - 4), `b` айнымалысына 7 саны, ал `c` қосындысының нәтижесі 59 болады. `b` айнымалысы үшін монитор экранында кідіріс болмайды, `b` айнымалысына консольда енгізуге арналған буфердің қалған бөлігі меншіктеледі.

C# тілінде бағдарламалауды меңгеру барысында `Console.Read()` әдісін қолдануға кеңес берілмейді (жолдық айнымалыларды оқуға дейін).

2.5 Меншіктеу операторы

C# бағдарламалау тілінде меншіктеу операторы ‘=’ символымен белгіленеді. Меншіктеу операторының жазылу пішімі мына сипатта болады:

```
айнымалы = өрнек;
```

Мысалы:

```
c = a + b;
```

‘=’ белгісінің сол жағында айнымалы орналасуы керек, ал ‘=’ белгісінің оң жағында айнымалылар, тұрақтылар, нақты мәндер немесе өрнектер жазылуы мүмкін. Өрнектер ретінде арифметикалық операциялар, әртүрлі функциялар, т.б. бола алады. Өрнектегі әрекеттердің орындалу реті солдан оңға қарай, бірақ жақшалар арқылы оны өзгертуге болады.

C# тілінде келесі арифметикалық операциялар қолданылады:

+ – қосу операциясы;

- – азайту операциясы;

* – көбейту операциясы;

/ – нақты типтегі айнымалылар үшін бөлу операциясы;

/ – бүтін типтегі айнымалылар үшін бүтін санды бөлу операциясы (қалдығы жойылады – бөлудің нәтижесін мысалдан көріңіз);

% – бүтін санды бөлу операциясынан қалдықты табу операциясы бүтін типтегі айнымалылар үшін қолданылады.

Мысалы,

```
7/2=3;        7.0/2.0=3.5;      6%2=0;   7%2=1;
```

C# тілінде аталған операциялардың кез келгені меншіктеу операциясының қысқартылған түрде жазылу мүмкіндігін береді:

```
айнымалы операция = өрнек;
```

Мысалы, егер `a` айнымалысы 5 тең болса, онда `a += 6` жазбасы `a` айнымалысына 11 мәнін меншіктейді. ‘+=’ операциясы ‘=’ символының оң жағына жазылған өрнектің мәнін ‘+’ символының сол жағында орналасқан айнымалы мәніне қосуды орындайды. Қосу опрациясының нәтижесі ‘+’ символының сол жағында орналасқан айнымалыға меншіктеледі.

‘–’ , ‘ *’, ‘ /’, ‘ %’ операциялары үшін меншіктеу операторы жоғарыда сипатталған түрде орындалады.

Мысалы, егер `a` айнымалысы 5 тең болса, онда `a %= 3;` `меншіктеу операторы орындалғаннан кейін a` айнымалысы 2 тең болады .

Алгебралық өрнектің нәтижесін кейбір айнымалыға меншіктеу операциясын орындаған кезде типтердің айқын немесе айқындалмаған түрде түрленуі орындалады.

Егер алгебралық өрнектің операндтары әртүрлі типте болса, онда есептеулердің алдына автоматты түрде типтерді түрлендіру орындалады. Түрлендіру мәні мен дәлдігін сақтай отырып, қысқа типтерді ұзын типтерге ауыстыруды қамтамасыз ететін ереже бойынша орындалады, бірақ керісінше емес.

Автоматты түрде (айқындалмаған) түрлендіру ылғи орындалмауы мүмкін, мәні жоғалмаған жағдайда ғана орындалады.

Арифметикалық операциялар `int` типінен қысқа типтерде анықталмаған. Бұл дегеніміз, егер өрнекте `sbyte, byte, short` және `ushort` типтеріндегі ғана шамалар болса, онда операцияның орындалуының алдында олар `int` типіне түрлендіріледі. Сонымен, кез келген арифметикалық операциялардың нәтижесі `int` типінен кіші емес типте болады.

Бір типтен екінші типке айқындалмаған түрлендіру орындалмаса, онда бағдарламашы мына операция арқылы айқын түрлендіруді орындай алады.

`(тип) өрнек.`

Мысалы, бүтін типтегі айнымалыға нақты типтегі мән меншіктелсе, онда осы мәннің бөлшек бөлігі жойылады. Типтерге айқын түрлендіруді орындау үшін `float` немесе `int` нұсқауларын пайдалануға болады. Мысалы:

```
x = (float)(31*c - 16);
Console.WriteLine("x={0}", x);
//или
a = (int)(18.6/3.6 + 3.7);
Console.WriteLine("a={0}", a);
```

Типтерді айқын түрлендіруді бағдарламаға айқындылықты кіргізу үшін әр түрлі типтегі айнымалылары бар есептеулерде қолдану орынды, бірақ айқын түрлендіру есептеудің дәлдігін жоғалтумен байланысты екенін ескеру керек.

C# тілінде инкремент (айнымалыны бір шамасына өсіру) және декремент (айнымалыны бір шамасына кеміту) операциялары үшін қысқартылған жазбалар қолданылады, яғни меншіктеу операциялары.

`айнымалы = айнымалы + 1;`

мына жазбамен ауыстырылады

`айнымалы ++; ,`

меншіктеу операциясы

`айнымалы = айнымалы - 1;`

мына жазбамен ауыстырылады

`айнымалы --; .`

Мысалы, егер `a` айнымалысының мәні 5-ке тең болса, онда меншіктеу `a++;` операторын орындағаннан кейін `a` айнымалысының

мәні 6-ға тең болады. Егер енді `a--;` меншіктеу операторы орындалса, онда `a` айнымалысының мәні 5-ке тең болады.

C# тілінде инкремент және декремент операциялар жазбасының префиксті пішіні рұқсат етілген, мысалы `++a` немесе `--a`, `бұл жазбалар` жоғарыда қарастрылған постфиксті пішімдегі жазбалардан өзгеше.

Жазбаның префиксті пішімі біріншіден инкремент немесе декремент операцияларының орындалуын, ал содан кейін айнымалыны пайдалануды талап етеді. Постфиксті пішімдегі жазулар біріншіден айнымалыны, одан кейін инкремент немесе декремент операцияларының орындалуына рұқсат береді. Мысалы, `a = b++;` өрнегін келесі екі оператормен ауыстыруға болады: `a = b; b = b + 1;`. C# тілінде `a = ++b;` өрнегі келесі екі операторға балама болады: `b = b + 1; a = b;`.

C# тілінің осындай ерекшеліктерін бағдарлама кодын жазу кезінде ескеру керек.

C# тілінің басымдылықтары бойынша реттелген негізгі операцияларының тізімі ұсынылған әдебиетте келтірілген.

Қарастырылған бірінші бағдарламаның жолдарының бірінде

```
c = a; a = b; b = c;
```

өте маңызды алгоритм жазылған - алмасу алгоритмі . Бұл алгоритм `a, b` және `c` айнымалыларының мәндерін алмастыруға мүмкіндік береді. `Осы` алгоритм басқа да күрделі алгоритмдерде қолданылады, мысалы сандарды сұрыптау немесе тізімді алфавиттік ретте пішімдеу, т.б және осы алгоритмді түсіну өте маңызды.

2.6 C# тілінің стандартты математикалық функциялары

Кез келген бағдарламалау тілдері сияқты C# тілінде `Math` класына тиісті (`System атаулар кеңістігінде`), бағдарлама кодын жазғанда пайдалануға болатын стандартты математикалық функциялар (әдістер) жиыны бар.

Барлық әдістер `public` және `static` модификаторларымен жарияланады, сондықтан олар `Math` класының объектісін алдын ала құрмай-ақ бағдарламаның кез келген орнынан қол жетімді. 2.2-кестеде `Math` класының негізгі әдістері көрсетілген.

Әдістерді қайта анықтау дегеніміз – әр түрлі деректер типі үшін атауы бойынша бірдей бірнеше әдістің қолданылуы. Мысалы, `Max()` әдісіне нақты немесе бүтін сандарды беруге болады.

2.2-кесте – Math класының негізгі әдістері

Әдіс	Сипаттамасы
double Abs(double d);	Аргумент модулін қайтарады.

double Acos(double d);	Арккосинус бойынша радианда бұрышты қайтарады.
double Asin(double d);	Арксинус бойынша радианда бұрышты қайтарады.
double Atan(double d);	Арктангенс бойынша радианда бұрышты қайтарады.
Long BigMul(int x, int y);	Екі 32-разрядты санның көбейтіндісін қайтарады.
double Ceiling(double d);	Аргументке тең немесе одан үлкен ең кіші бүтін санды қайтарады.
double Cos(double d);	D бұрышының косинусын радианмен қайтарады.
double Cosh(double d);	D бұрышының гиперболалық косинусын радианмен қайтарады.
int DivRen(int a, int b, out int R);	Екі бүтін сандарды бөлудің нәтижесін және шығу параметрі ретіндегі R қалдығын қайтарады.
E	2,71828182845905
double Exp(double d);	d дәрежелі E қайтарылады.
double Floor(double d);	Берілген санға тең немесе одан кіші ең үлкен бүтін санды қайтарады.
double IEEERemainder (double a, double b);	a санын b санына бөлу нәтижесінің қалдығын қайтарады.
double Log(double d);	d санының натурал логарифмін қайтарады. Артық жүктелген әдісте екінші параметр болып логарифм негізі жіберіледі.
double Log10(double d);	d санының ондық логарифмін қайтарады.
double Max(double a, double b));	Екі санның ең үлкенін қайтарады. Артық жүктелген әдіс.
double Min(double a, double b);	Екі санның ең кішісін қайтарады. Артық жүктелген әдіс.
PI	3,14159265358979
double Pow(double a, double b);	a санының b дәрежесін қайтарады.
double Round(double a);	A санын бүтін санға — дөңгелектеу. Артық жүктелген әдіс.
int Sign(double a);	a санының нөлден кіші, нөлге тең немесе одан үлкен болуына байланысты -1, 0 немесе +1 қайтарады. Артық жүктелген әдіс.
double Sin(double a);	Радианда a бұрышының синусын қайтарады
double Sinh(double a);	Радианда a бұрышының гиперболалық синусын қайтарады

double Sqrt(double a);	a-ның квадрат түбірін қайтарады.
double Tan(double a);	Радианда a бұрышының тангенсін қайтарады.
double Tanh(double a);	Радианда a бұрышының гиперболалық тангенсін қайтарады

Бағдарламаны ретке келтіру процесінде деректерді енгізуді немесе кейбір процестерді үлгілеуді жеңілдету үшін көптеген бағдарламашылар белгілі бір диапазонда тең қалыпты бөлінген, бүтін немесе нақты псевдокездейсоқ сандар тізбектілігінің генераторын қолданады. Осындай тізбектілікті құратын әдістер `Random` класында орналасқан.

Класс негізгі мән ретінде кейбір бастапқы санды қолданады, оған разрядтарды араластыру алгоритмы қолданылады және осындай жолмен алынған санды кезекті кездейсоқ сан ретінде қайтарады. Сонымен қатар келесі кездейсоқ санды өндіру үшін осы сан негізгі сан болып есептеледі. Сонымен, кезекті санның араластыру алгоритмі және бастапқы мәні толық анықталатындықтан, псевдокездейсоқ сандар тізбектілігі өндіріліп шығады.

2.3-кесте –Random класының кейбір әдістері

Әдіс	Сипаттамасы
Public virtual int Next();	Кезекті псевдокездейсоқ санды қайтарады. Әдістің артық жүктелген нұсқаларында тудырылатын сандардың ең жоғарғы мәнін немесе олардың мәндерінің диапазонын көрсетуге болады.
Public virtual void NextBytes (byte[] buffer);	buffer айнымалысын псевдокездейсоқ мәндері бар байттармен толтырады.
Public virtual double NextDouble();	0.0-ден 1.0 дейінгі диапазондағы нақты псевдокездейсоқ санды қайтарады.

Класта екі конструктор бар: параметрсіз және `int` параметрлі. Біріншісі бастапқы мән ретінде ағымдағы күн мен уақытты, екіншісі бастапқы, негізгі мәнді қабылдайды. Сонымен, бірінші конструктор қайталанбайтын сандар сериясын, екіншісі - бірдей сандар сериясын шығара алады.

Ескерту, кластың әдісі тұрақты емес, сондықтан оларды пайдалану үшін `rnd` класының объектісі міндетті түрде құрылады. Мысалы:

```
Random rnd = new Random();
Console.Write(" " + (rnd.Next()%101).ToString());
Console.WriteLine();
```

Бұл үзінді 0-ден 100 дейінгі аралықта псевдокездейсоқ бүтін санды қалыптастырады. Бұл мүмкін, өйткені `rnd.Next()` әдісі 0-ден ең жоғарғы деңгейге дейін псевдокездейсоқ бүтін санды қалыптастырады.

101-ге бөлуден қалатын қалдық 0-ден 100 дейінгі аралықта болады. `a = rnd.Next() % 101 - 50;` - жазбасы -50-ден 50 дейінгі аралықта псевдокездейсоқ бүтін санды қалыптастырады.

Талдау жасау үшін жалған кездейсоқ бүтін санды қалыптастырудың барлық жағдайлары қарастырылады, 0 – бұл жағдайда `a` айнымалысының мәні минус 50, ал 100 болғанда - `a` айнымалысының мәні 50-ге тең.

`a` айнымалысының барлық қалған мәндері осы шеткі мәндер арасында болады.

Стандартты математикалық функцияларды қолдану кез келген күрделі алгебралық өрнектерді бағдарламалауға мүмкіндік береді.

2.7 Мәтіндік файлдармен жұмыс

Бағдарламаны тестілеу кезеңінде бағдарлама жұмысын әр түрлі кірістік деректердің мәндері бойынша тексеруге тура келеді. Осындай «тестілеу» мәндер арнайы «кірістік» деректердің мәтіндік файлына жазылады және бағдарламаның орындалуы үшін кезекпен ұсынылады. Кейіннен тексеру мен талдау жұмыстарын жүргізу үшін бағдарлама жұмысының нәтижесі көбінесе «шығыстық» файлға жазылады.

Осы мақсаттарды жүзеге асыру үшін C# тілінде мәтіндік файлдармен жұмыс жасау қарастырылған - `input.txt` атты «кірістік» файл және `output.txt` атты «шығыстық» файл. Файлдар бағдарлама файлдары орналасқан каталогта құрылуы тиіс, бастапқы орны — ...\ConsoleApplication1\bin\Debug..

Мысал ретінде бағдарламаны тестілеу процесін қарастырайық, онда нақты санды бөлу нәтижелері қарастырылады.

Бағдарлама коды:

```
using System;
using System.IO;
using System.Text;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
 {
  double z,y,x;
  StreamReader f1 = new StreamReader("input.txt");
  StreamWriter f = new StreamWriter("output.txt");
  string buf = f1.ReadLine();
  z = Convert.ToDouble(buf);
  buf = f1.ReadLine();
  x = Convert.ToDouble(buf);
  f.WriteLine("z = {0} x = {1} ", z, x);
  y = z / x;
```

```
        f.WriteLine(" z / x = {0} ", y);
        y = x / z;
        f.WriteLine(" x / z = {0} ", y);
        y = -z / 0;
        f.WriteLine(" - z / 0 = {0} ", y);
        y = -0.0 / 0.0;
        f.WriteLine("-0.0 / 0.0 = {0} ", y);
        y = 0.0 / 0.0;
        f.WriteLine(" 0.0 / 0.0 = {0} ", y);
        f1.Close();
        f.Close();
      }
      }
    }
```

`input.txt` «кірістік» файлында 123 саны жазылған, осы сан `z` айнымалысына меншіктеледі. Екінші сан берілмегендіктен `x` айнымалысына нөл меншіктеледі.

Бағдарлама жұмысының нәтижесі `output.txt` шығыстық, мәтіндік файлда орналасады, нәтижені «блокнот» арқылы көруге болады:

```
 z = 123 x = 0
 z / x = бесконечность
 x / z = 0
 - z / 0 = -бесконечность
-0.0 / 0.0 = NaN
 0.0 / 0.0 = NaN
```

Бағдарлама жұмысының нәтижесін толығырақ қарастырайық. Біз нөлге бөлу жағдайын жасадық, бұл жағдай көптеген бағдарламалау тілдерінде бағдарлама жұмысын үзеді.

C# тілінде нақты сандар үшін үш нұсқа қарастырылған (бағдарламаның есептеу нәтижелері үшін) – `Infinity`, `NegativeInfinity` және `NaN`. Алғашқы екеуі математикадан белгілі - шексіздік және теріс шексіздік. Үшінші `NaN` (Not a Number) мәні нәтиже нақты сан болмағанда немесе бағдарлама нәтижені анықтай алмаған жағдайда орын алады.

Осы мәндердің пайда болу жағдайларын қарастырайық. Егер көбейту немесе бөлу операциясының орындалуы барысында модуль бойынша нәтиже ең жоғарғы мүмкін саннан артық болса, онда нәтиженің таңбасына қарай мәні шексіздікке немесе теріс шексіздікке ие болады. Осы мәндерді анықтайтын `double` және `float` типтерінің тұрақтылары бар. Нақты сандар мен шексіздік арасында қосу, алу және көбейту операцияларын орындағанда нәтиже шексіздік мәніне ие болады, нәтиже теріс таңбада болуы мүмкін. Нақты санды шексіздікке бөлгенде нәтиже нөлге тең болады.

Егер шексіздік шексіздікке бөлінсе немесе нөл шексіздікке көбейтілсе, онда нәтиже `NaN` болады. Операция орындалу нәтижесі нақты сан болмаса, мысалы, теріс сан квадратының түбірін шығару кезінде, онда

нәтиже жоғарыдағыдай `NaN` болады. Егер операциялар ішінде `NaN қолданылса, онда` нәтиже `NaN` болады.

Қарастырылған мысалдың екінші мақсаты - мәтіндік файлдармен жұмыс. Осы жұмыстың негізгі кезеңдерін қарастырайық.

Біріншіден, арнайы атаулар кеңістігін қосу керек:

```
using System.IO,
```

файлдың енгізу-шығару жұмысына жауап береді.

Екіншіден, файлдың айнымалыларын жариялау керек – файлмен жұмыс жасау үшін объекттерді құру:

```
(StreamReader f1 = new StreamReader("input.txt");
StreamWriter f = new StreamWriter("output.txt");),
```

Оларға магнитті дискідегі файл аты сәйкес келеді

Үшіншіден, магнитті дисктегі мәтіндік ақпаратты жазу мен оқу операцияларын орындау (енгізу және шығару).

Төртіншіден, алынған мәтіндік ақпаратты сәйкес типтердің мәндеріне түрлендіру операцияларын орындау.

Бесіншіден, мәтіндік файлмен жұмысты орындап болғаннан кейін оны жабу керек.

```
f1.Close(); f.Close();
```

2.8 Өзін-өзі тексеру сұрақтары

1 `using` қызметтік сөзі не үшін қолданылады?

2 C# тіліндегі бағдарламаның негізгі әдісі қалай аталады?

3 `Math.Pow(x,y)` функциясы нені есептейді?

4 Компьютер пернетақтасынан x айнымалысының мәнін енгізу үшін қандай әдістерді қолдануға болады?

5 Монитор экранына x айнымалысының мәнін шығару үшін қандай әдістерді қолдануға болады?

6 Бағдарламада түсініктемелер қалай жазылады?

7 `10 % 3` өрнегі неге тең?

8 `10 / 3` өрнегі неге тең?

9 Сандарды шығару пішімі қалай анықталады?

10 Бағдарламада стандартты математикалық функцияларды қолданбас бұрын нені орындау керек?

3 C# ТІЛІНІҢ КҮРДЕЛІ ОПЕРАТОРЛАРЫ

3.1 Шартты өту операторы if

Бағдарламалау тілдерінде барлық операторлар қарапайым және күрделі операторлар болып бөлінеді. Сонымен бірге күрделі оператор деп өз қызметінде басқа операторларды қолданатын операторды жатқызуға болады.

C# тілінде күрделі операторлардың бірнеше операторы болса, онда ол фигуралы жақшаға алынады.

Кең таралған күрделі операторларға шартты өту операторы және цикл операторы жатады. Әлбетте, C# тілінде басқа да күрделі операторлар бар, бірақ оқу құралының осы бөлімінде біз тек осы екі операторларды ғана қарастырамыз.

if шартты өту операторы бағдарламада есепті шешу алгоритмінің кейбір логикалық шартын тексеруді және тексерудің нәтижесіне байланысты бағдарламаның жұмысын мүмкін екі жолдың бірімен жалғастыру керек болған жағдайда қолданылады. Бағдарламаның «тармақталған» учаскесі термині де бар, оларға ауысу үшін шартты өту операторы қолданылады.

`if` операторының жазылу пішімі:

```
if ( шарт) { операторлар; } else { операторлар;}
```

Егер шарт «ақиқат» болса, онда шарттан кейін орналасқан, фигуралық жақшалар ішіндегі операторлар орындалады, әйтпесе else қызметтік сөзінен кейін орналасқан, фигуралық жақшалар ішіндегі операторлар орындалады. Мысалы,

```
if (a > b) { x = a; y = b;} else { x = b; y = a;}
```

Көрсетілген үзіндіде азаймалы ретте екі айнымалының мәндерін ретке келтіру алгоритмі қарастырылған. `a` және `b` айнымалының мәндері салыстырылады және ең жоғарғы мәні бар айнымалы `x` айнымалысына, ал ең кіші мәні бар айнымалы `y` айнымалысына меншіктеледі.

Есептің шарты бойынша екі айнымалының арасындағы ең үлкен мәнін табу және оны `x` айнымалысына меншіктеу керек болса, онда осы есептің шешімінің алгоритмін шартты өту операторы арқылы жазуға болады:

```
if (a > b) x = a; else x = b;
```

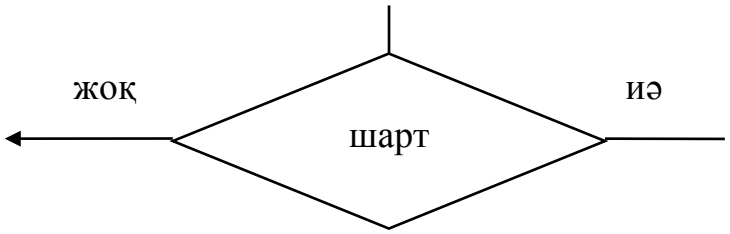
Осы жазбада оператордың іс-әрекет облысын ерекшелеу үшін фигуралық жақшалар қолданылмайды, өткені оператордың іс-әрекет облысы шарттың ақиқат және жалған мәндерінде бір ғана меншіктеу операторынан тұрады.

`if` операторының жазбасында else қызметтік сөзі болмауы мүмкін. Онда, шарт орындалмаған жағдайда бағдарлама `if` операторының сыртындағы операторды орындауға көшеді.

`Егер if` операторының іс-әрекет облысында басқа шартты өту операторлары болса, онда түсінбестікті болдырмау үшін `else` сөзі мен оның іс-әрекет облысы ең жақын сипаталған `if` операторына тиісті болады.

Есеп шішімі алгоритмінің құрылымдық схемаларын дайындағанда шарт ромб түрінде бейнеленеді, шартты жалғастырудың екі нұсқасы бар.

Шарттың графикалық көрінісі мына түрде болады:



3.1-сурет – Есеп шешімі алгоритмінің құрылымдық схемаларында шарттың графикалық көрінісі.

`if` операторының шарты алгебралық өрнектерден, салыстыру және логикалық операциялардан тұрады.

C# тілінде келесі салыстыру операциялары қолданылады:

`A == B` – егер А В-ға тең болса, нәтиже true мәніне, әйтпесе false мәніне тең болады;

`A != B` – егер А В-ға тең болмаса, нәтиже true мәніне, әйтпесе false мәніне тең болады;

`A < B` – егер А В-дан кіші болса, онда нәтиже true мәніне, әйтпесе false мәніне тең болады;

`A > B` – егер А В-дан үлкен болса, онда нәтиже true мәніне, әйтпесе false мәніне тең болады;

`A <= B` – егер А В-дан кіші немесе тең болса, онда нәтиже true мәніне, әйтпесе false мәніне тең болады;

`A >= B` – егер А В-дан үлкен немесе тең болса, онда нәтиже true мәніне, әйтпесе false мәніне тең болады;

C# тілінде логикалық операциялардың екі түрін айырады – разрядтық (поразрядные) логикалық операциялар және шартты логикалық операциялар. Разрядтық логикалық операциялар бүтін сандық типтерге қолданылады екілік (двоичном) түрде ұсынылған. Осы логикалық операцияларды қолданған кезде екілік нәтиже «0» немесе «1» болады.

Шартты логикалық операциялар әр түрлі операторлар «шарттарында» қолданылады және оның жұмысының нәтижесінің мәндері true немесе false болады.

Разрядтық логикалық операциялар:

`&` – разрядтық логикалық көбейту «ЖӘНЕ» операциясы;

`|` – разрядтық логикалық қосу «НЕМЕСЕ» операциясы;

^ – разрядтық логикалық алып тастау «НЕМЕСЕ» операциясы.

Шартты логикалық операциялар:

&& – логикалық көбейту «ЖӘНЕ» операциясы;

|| – логикалық қосу «НЕМЕСЕ» операциясы;

! – логикалық терістеу «ЖОҚ» операциясы.

Логикалық көбейту «ЖӘНЕ» операциясы true (ақиқат) мәнін қабылдайды, егер осы операцияның барлық көбейткіштері true мәніне тең болса.

Логикалық қосу «НЕМЕСЕ» операциясы true мәнін қабылдайды, егер осы операцияның кем дегенде бір қосылғышы true мәніне тең болса.

Өрнектің логикалық терістеу операциясы true мәнін қабылдайды, егер өрнек false мәніне тең болса. Өрнектің логикалық терістеу операциясы false мәнін қабылдайды, егер өрнек true мәніне тең болса.

Ескерту, логикалық операциялар салыстыру операцияларына қарағанда басымдығы төмен. Осыны кейбір логикалық өрнектерді жазуда ескеру керек. Мысалы, өрнегінің мәні «ақиқат» шартты өту операторын жазу керек болса, егер кейбір x айнымалы 0-ден үлкен, ал 10-нан кіші болса, онда бұл өрнек мына түрде жазылады:

```
if ( x > 0 && x < 10 ).
```

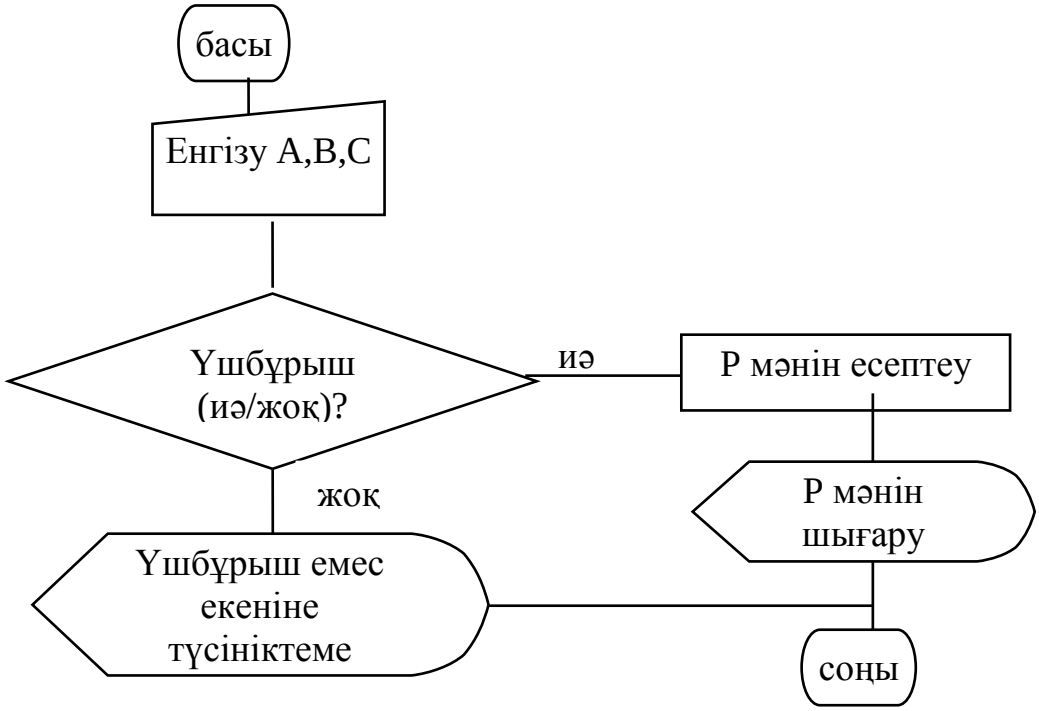
Шартты қолданатын есептің мысалын қарастырайық.

3.1 есебі. Диалог режимінде үшбұрыш жақтарын беру және оның периметрін есептеу керек. Үшбұрыш жақтарының мәндерін енгізгеннен кейін келесі тексерулерді орындау керек: үшбұрыштың барлық жақтары нөлден және кез келген екі жақтарының қосындысы үшіншісінен үлкен болуы керек. Бағдарлама жұмысына тиісті түсініктеме беру керек.

Есепті шешуге арналған құрылымдық схемасын дайындайық. Біріншіден, диалог режимінде үшбұрыш жақтарын енгізуді ұйымдастырамыз.

Одан кейін енгізілген сандармен үшбұрышты тұрғызу мүмкіндігі бойынша шарттарды тексеру керек . Егер «Иә» болса, онда үшбұрыштың периметрін есептеу және шығару керек, әйтпесе тиісті түсініктемені шығарыңыз. 3.1 есепті шешуге арналған алгоритмнің құрылымдық схемасы 3.2. суретінде көрсетілген.

Құрылымдық схемаларды бейнелегенде «төмен» және «жоғары» сызықтарының тілдері болмайды. Бірақ «солға» немесе «оңға» сызықтары болса, онда ортақ сызықтың тілі бар болуы керек.



3.2-суреті – Есепті шешуге арналған алгоритмнің құрылымдық схемасы

Есепті шешуге арналған алгоритмнің құрылымдық схемасына сәйкес бағдарлама кодын дайындаймыз.

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
 {
  int a, b, c;
  string buf;
 vvod:
  Console.Write("a bytin canin engiziniz ");
  buf = Console.ReadLine();
  a = Convert.ToInt32(buf);
  Console.Write("b bytin canin engiziniz ");
  buf = Console.ReadLine();
  b = Convert.ToInt32(buf);
  Console.Write("c bytin canin engiziniz ");
  buf = Console.ReadLine();
  c = Convert.ToInt32(buf);
  if (a > 0 && b > 0 && c > 0)
  if (a + b > c && a + c > b && b + c > a)
   Console.WriteLine("PERIMETR = {0}", a + b + c);
  else
  {
```

```
        Console.WriteLine("Yshbyrishtin bir kabirgasi kalgan
eki kabirganin kocindisinan ylken nemese ten"); goto vvod;
       }
       else
       {
       Console.WriteLine("Bir kabirgasi <= 0 !");
       goto vvod;
       }
       Console.WriteLine("Enter pernesin basiniz");
       Console.ReadLine();
      }
      }
     }
```

Бағдарлама жұмысы:

```
a bytin canin engiziniz 1
b bytin canin engiziniz 2
c bytin canin engiziniz 3
Yshbyrishtin bir kabirgasi kalgan eki kabirganin
kocindisinan ylken nemese ten
a bytin canin engiziniz -4
b bytin canin engiziniz 5
c bytin canin engiziniz 6
Bir kabirgasi <= 0 !
a bytin canin engiziniz 7
b bytin canin engiziniz 8
c bytin canin engiziniz 9
PERIMETR = 24
Enter pernesin basiniz
```

Есепті шешуге арналған алгоритмнің «Шарты» бірнеше тексеруді жүргізеді. Тексеруді бір шартты өту операторымен немесе бірнеше шартты өту операторларымен орындауға болады, мысалы, екі.

Біздің бағдарламамызда екі `if` операторы қолданылған.

Бірінші `if` операторы үшбұрыштың барлық жақтары 0-ден үлкен болу шартын тексереді. Егер шарт орындалса, онда екінші `if` операторына көшу орындалады, әйтпесе монитор экранына “`Bir kabirgasi <= 0 !`” хабарламасы шығады және бағдарлама `goto` операторы арқылы үшбұрыш жақтарының мәндерін қайталап енгізу үшін `vvod` таңбасына (тамғасына) көшеді.

Екінші `if` операторының шартты үшбұрыш жақтарының арақатынасын тексереді. Егер шарт орындалса (бұл екі `if` операторлардың шарттары орындалуымен сайма-сай), онда монитор экранына үшбұрыш периметрінің мәні шығады және бағдарлама аяқталады. Әйтпесе, монитор экранына «`Yshbyrishtin bir kabirgasi kalgan eki kabirganin kocindisinan ylken nemese ten`” хабарламасы шығады және `goto` оператор арқылы үшбұрыш жақтарының мәндерін қайталап енгізу орындалады.

goto операторы C# тілінің күрделі операторы болып табылмайды, бірақ әдетте бұл шартсыз өту операторы if шартты өту операторын оқығанда қарастырылады.

Шартсыз өту операторы пішімінің жазбасы: goto таңба; .

Шартсыз өту операторын қолдану алдында таңбаны жазу керек (біздің мысалда vvod идентификаторы:), оған біз бағдарламаны бағыттаймыз. Таңба атауынан кейін қос нүкте символын қою керек. Егер бағдарламада шартсыз өту операторы кездессе, онда бағдарлама тиісті таңбадан кейін тұрған операторды орындауға көшеді.

Бір таңбаға кем дегенде бір шартсыз өту операторы сәйкес келуі тиіс.

Біздің бағдарламада есеп шарттары орындалмаған жағдайында шартсыз өту операторының көмегімен үшбұрыш жақтарының мәндерін қайталап енгізу ұйымдастырылды («дұрыс» мәндер берілгенге дейін).

Есеп шешімінің алгоритмінде ол жоқ, бірақ бағдарлама кодын жазғанда есептің шешіміне жалпы шешімді жетілдіретін кейбір үзінділерді әрдайым толықтыруға болады.

Есептің шарты бойынша бағдарламаның орындалуы түсініктемелермен толықтырылуы тиіс. Мысалда үшбұрышты құрастыру шарттары орындалмаған жағдайда түсініктемелер қолданылады.

C# бағдарламалау тілінде шартты операторы бар, ол функция рөлін атқарады және оның жұмысының нәтижесін кейбір айнымалыға меншіктеу немесе кейбір өрнекте, басқа операторда қолдану керек. Қысқаша белгіленуі – ?:.

Осы операторлың пішімі келесі түрде болады:

```
өрнек1 ? өрнек2 : өрнек3;
```

мұндағы өрнек1 – шарт болады. Егер шарт ақиқат болса, онда ?: операторының мәні өрнек2 сәйкес, әйтпесе өрнек3. Мысалы, бағдарламаның келесі үзіндісінде осы оператор арқылы a, b және c үш айнымалысының ішінен ең жоғарғы мән анықталады:

```
x= a>b ? a : b;
x= x>c ? x : c;
```

?: операторын қысқартылған if операторы деп жиі атайды, өйткені оны кейбір есептерде шартты өту операторының орнына қолдануға болады. Алайда if операторы C# тілінің басқа операторларын өзіне қосуы мүмкін, ал ?: операторының ондай мүмкіндігі жоқ.

C# тілінде бағдарлама барысының бірнеше нұсқасының ішінен бірін таңдауды ұсынатын тағы да бір switch – шартты операторы бар.

switch операторының жұмысы мен жазу пішімін массивтер тақырыбында бағдарлама менюін дайындағанда қарастырамыз.

3.2 For циклінің операторы

For циклінің операторы циклдық операциялар саны алдын-ала белгілі болған жағдайда қолданылады.

`for` операторын жазу пішімі:

```
for(өрнек1;өрнек2;өрнек3)
     { операторлар;},
```

мұнда;

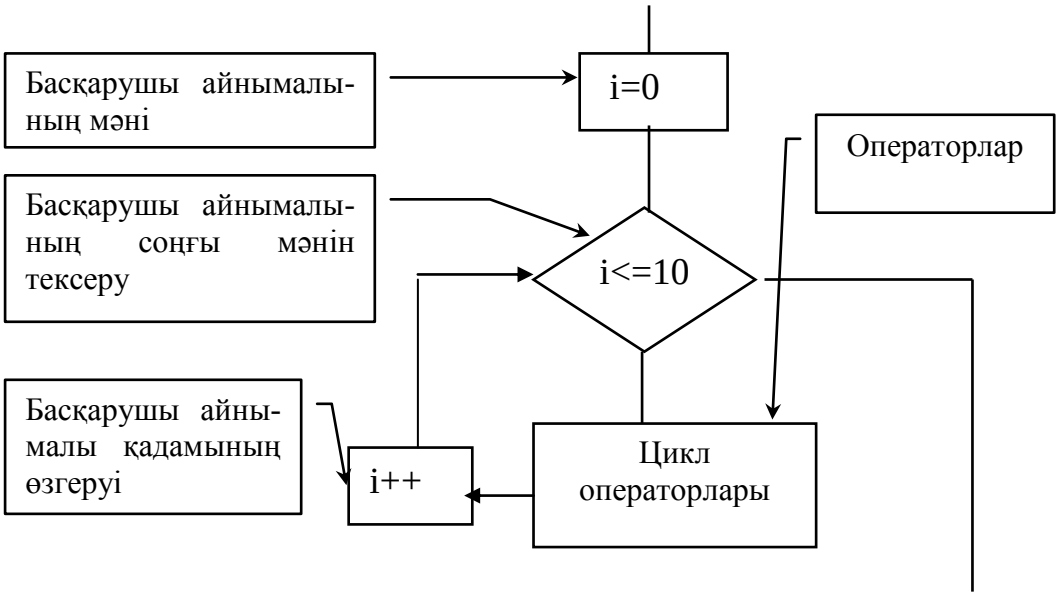
`өрнек1` – циклдің басқарушы айнымалысының бастапқы мәнін анықтайды;

`өрнек2` – циклдің басқарушы айнымалысының ақырғы мәнін анықтайды;

`өрнек3` – циклдің басқарушы айнымалысының өзгеріс қадамын анықтайды.

Мысалы: `for (i = 0; i <= 10; i++) {цикл операторлары}`

3.3 суретінде `for` циклінің графикалық көрінісі келтірілген.



3.3 суреті – For операторының графикалық көрінісі.

`for` циклін жазудың басқа мысалдарын қарастырайық:

1) `for (int i = 0; i <= 10; i++) {цикл операторлары}`

Бұл мысалда `for` циклінің операторы жазылған, онда цикл ішінде бүтін типті басқарушы айнымалы жарияланады және ол 0-ден 10-ға (қоса санағанда) дейін 1 қадамымен өзгереді.

2) `for ( i = 10; i >= 0; i--) {цикл операторлары }`

Бұл мысалда `i` – бұрынырақ жарияланған айнымалы басқарушы айнымалы ретінде қолданылады, ол 0-ден 10-ға (қоса санағанда) дейін минус 1 қадамымен өзгереді.

3) `for ( x = 0; x <= 1; x = x + 0.1) {цикл операторлары}`

Бұл мысалда басқарушы айнымалы ретінде `x` - нақты типтегі айнымалы қолданылады, ол 0-ден 1-ге дейін 0.1 қадамымен өзгереді.

3.2-есеп. минус 50-ден 50-ге дейінгі ауқымда 10 кездейсоқ бүтін сандарды басып шығару.

Есеп шешімінің алгоритмін құру жұмысын оның құрылымдық схемасымен аяқтау міндетті емес. Алгоритм жазбаша түрде, яғни есептің шешу қадамдарын толық сипаттау түрінде көрсетілуі мүмкін.

Алдымен `max` және `min` айнымалыларын жариялау керек. Оларға сәйкесінше 10 кездейсоқ сандардың ішінен ең үлкенін және ең кішісін жазамыз.

Кездейсоқ сандардың ауқымын біз білеміз, сондықтан `max` айнымалысына сандар ауқымынан кіші немесе оның ең кіші мәніне тең санды меншіктеу керек, мысалы минус 100, ал `min` айнымалысына сандар ауқымынан үлкен немесе оның ең үлкен мәніне тең санды меншіктеу керек.

Әлбетте, `max` және `min айнымалыларына` бірінші рет құрылған кездейсоқ санды меншіктеу дұрыс болады, бірақ есептің шарты бойынша олардың 10-ын құру керек, оны циклде орындаған абзал және бізге кейбір мәндерді циклге дейін меншіктеу керек. Егер меншіктеу операциясын қолданбаса, онда айнымалыларға нөл мәндері меншіктеледі, бірақ 0 саны сандар аралығының бір бөлігі, сондықтан нөлдік мәндер бағдарлама жұмысының нәтижесін бұзуы мүмкін (барлық құрылған 10 сан 0-ден үлкен немесе кіші).

Сонымен, 10 кездейсоқ санды құру керек., сондықтан `for` циклінің қадамы бірге тең, 1-ден 100-ге дейінгі аралығындағы, бүтін типті басқарушы айнымалымен бірге қолдану орынды.

Цикл денесінде кездейсоқ сандарды құруды, ал берілген аралықта оны шығаруды және осы санды бір-бірден `max` және `min` мәндерімен салыстыруды орындау керек. Егер `a` саны `max үлкен болса, онда max-ге a санын жазу керек.`

Егер `a` саны `min` кіші болса, онда `min-ге a` санын жазу керек (бірінші құрылған кездейсоқ сан `max-нан` үлкен және `min-нен` кіші болады).

10 циклдік операция аяқталғаннан кейін `max` және `min` айнымалыларының мәндерін басып шығару керек.

Бағдарламаның бастапқы коды:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
```

```
 {
  int i, j, k, a, max, min;
  max = -100; min = 100; j = 0; k = 0;
  Random rnd = new Random();
  for(i=1;i<=10;i++)
  {
  a=rnd.Next() % 101 - 50;
  Console.Write(" {0}",a.ToString());
  if (a>max) {max = a; j = i;}
  if (a<min) {min = a; k = i;}
  }
  Console.WriteLine();
  Console.WriteLine("max={0} Nomer={1}",max,j);
  Console.WriteLine("min={0} Nomer ={1}",min,k);
  Console.WriteLine("Enter pernesin basiniz");
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
 -33 -26 -10 -16 -44 40 -50 -35 -45 -38
max=40 Nomer=6
min=-50 Nomer=7
Enter pernesin basiniz
```

Құрылған алгоритмге біз тағы екі қосымша `j` және `k` айнымалыларын қостық, оларға құрылған сандар тізбектілігі бойынша ең үлкен және ең кіші сандарды сәйкес меншіктедік.

Кейбір бағдарламалау тілдерінде басқарушы айнымалының мәнін цикл денесінде өзгертуге үзілді-кесілді рұқсат берілмеген. C# тілінде оған рұқсат берілмеген. Мысалы, баңдарлама үзіндісі:

```
  for(i=0;i<=10;i++)
  {
  Console.Write(" {0}",i.ToString());
  if (i == 3) i = 6;
  }
 Console.WriteLine();
```

шығарады

```
0 1 2 3 7 8 9 10
```

`for` цикл операторының жазбасында басқарушы айнымалыны жариялауға болады. Мысалы:

```
for (int j=1;j<=10;j++) { операторлар; }
```

`for` циклінің ішінде `j` айнымалысын жариялағаннан кейін біз автоматты түрде осы айнымалының «өмірлік уақытын» анықтаймыз, ол уақыт циклдің жұмыс істеу уақытына тең. Яғни `for` циклі жұмысқа қосылғаннан кейін компьютердің жадынан `j` басқарушы айнымалының

мәніне орын бөлінеді, ал цикл аяқталғаннан кейін жадының ол орны босатылады, яғни бос деп белгіленеді. Жергілікті айнымалыны жариялаудың осы әдісі бағдарламаны құруда қолданылады, оған қойылатын талаптардың бірі - компьютердің жадынан ең аз көлемді алуы керек.

`for` циклінің операторы бірнеше басқарушы айнымалыларын қолдануға мүмкіндік береді, мысалы:

```
for ( i=1, j=10; i<j; i + +, j - -)
```

`for` циклін шартты циклге айналдыруға болады, мысалы:

```
for ( i=0; (char) Console.Read() != 'n'; i++)
```

`for` циклінің осы жазбасы цикл жұмысының аяқталуын басқарушы айнымалымен байланыстырмай `'n'` символын енгізумен байланстырады.

C# тілінде `for` циклінің ерекше жазу пішімі болуы мүмкін, мысалы,

`for (i=0; i<=10) { . . . ; i++}` – басқарушы айнымалының өзгеру қадамы цикл тақырыбынан денесіне көшірілді;

`for (; ;)` – шексіз цикл және т.б.

Қарастырылатын оқу бағдарламаларына қойылатын негізгі талап - бағдарлама жұмысы бойынша алгоритмнің көрнектілігі, ол алгоритмнің көлеміне қойылатын талапқа сәйкес бола бермейді, сондықтан `for` циклінің барлық ерекше жазу пішімдерін біз қарастырмаймыз.

Егер кейбір шарттарға байланысты цикл тақырыбындағы шарттардың орындалуына дейін циклдің орындалуын үзу (тоқтану) керек болса, онда `break` операторы қолданылады. Мысалы, `for` циклінің денесінде циклдің орындалу «шексіздігін» тоқтату мен бақылау керек болған жағдайларда `break` операторын пайдалануға болады.

C# тілінде `break` операторынан басқа цикл денесінің орындалуын басқаратын тағы екі оператор қарастырылған – `continue` және `return`.

`continue` операторы цикл денесінің соңына дейінгі барлық басқа қалған операторларды орындамай, өткізіп жібереді және циклдің келесі итерациясын іске қосады.

`return` - қайтару операторы күрделі операторлардың немесе функцияның орындалуын аяқтайды және бағдарлама басқаруын функцияны шақыру нүктесінде немесе `return` операторын шақырған күрделі операторлардан кейінгі келесі операторға табыстайды. `Return` операторы мына пішімде болады:

```
return [өрнек];,
```

мұндағы өрнектің және `return` функциясын шақырған операторлардың типі бірдей болуы тиіс.

`Егер return` операторы C# тілінің күрделі операторын немесе `void` типті функциясын шақырса, онда өрнек болмау керек.

C# тілінде массивтер, коллекциялар деректерімен және деректер жиынын біріктіру формаларымен жұмыс істеуге арналған арнайы

(foreach) циклі бар. Осы циклдің жұмысы мен жазылу пішімі массивтер тақырыбында қарастырылатын болады.

3.3 Өзін-өзі тексеру сұрақтары

1 if операторында 0<A және B>0 шарттарын қалай дұрыс жазуға болады, A B-ға тең емес?

2 A && (! B) өрнегі неге тең?

3 A==B жазбасы нені білдіреді?

4 Шартты өту операторында келесі шарттарды қандай жазба дұрыс көрсетеді:

y = $\sqrt{(x+z)}$ егер z < x және x > 4, әйтпесе y = $\sqrt{(x-z)}$?

5 C# тілінің күрделі операторларында '{' . . . . '}' символдары не үшін қолданылады?

6 Бағдарламада for циклін қашан қолдану орынды?

7 Қандай оператордың көмегімен цикл жұмысын «уақытынан бұрын» аяқтауға болады?

8 Қандай оператордың көмегімен цикл денесінің бөлігінен «өтіп кетуге» болады?

9 Бағдарламаның келесі үзіндісі нені шығарады?

```
int k=5;
int i;
for (i = k; i <= 0; i --) k = k+1;
 Console.WriteLine("i = {0} k = {1} ", i, k);
```

10 for циклі аяқталғаннан кейін I басқарушы айнымалының мәні қандай болады?

```
int I;
 int k = 5;
 for (I = 0; I <= k; I++) k = k - 1;
 Console.WriteLine("I = {0} ", I);
```

4 ШАРТТЫ ЦИКЛ ОПЕРАТОРЛАРЫ

4.1 While циклінің операторы

While циклінің операторы «шартты цикл» деп жиі аталады, өйткені цикл жұмысының аяқталуы кейбір шарттардың орындалмауымен байланысты. Циклді жазу пішімі:

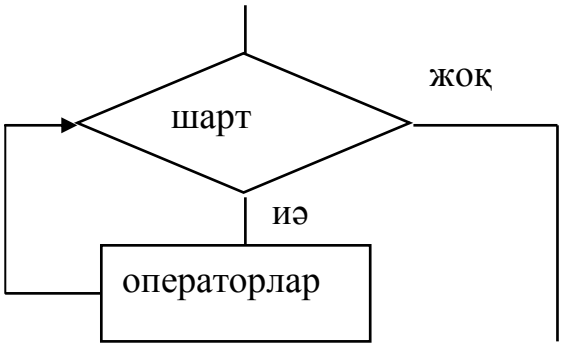
```
while (шарт)
{ операторлар; },
```

мұндағы,

`шарт` – логикалық өрнек, екі мәнді қабылдай алады: жалған немесе ақиқат.

Егер «шарт» ақиқат болса, онда while циклінің «операторлары» орындалады, әйтпесе цикл орындалмайды (аяқталады) және бағдарламаны басқару циклден кейінгі операторға көшеді.

Есептердің шешімдері бойынша алгоритмдердің құрылымдық схемасындағы `while` циклі келесі суретте көрсетілген.



4.1-суреті – `while` циклінің графикалық көрінісі

`while` циклі операторларына арналған мысалдары:

1) `while (a/x >= 0.0001) { операторлар;}`

Бұл мысалда цикл `a/x` қатынасы 0.0001 шамасынан кіші болғанға дейін орындалады.

2) `while ((x > 0) && (x <= 100)) { операторлар;}`

Бұл мысалда цикл `x` айнымалысының мәні 0-ден 100 дейінгі аралықта болғанға дейін орындалады.

4.1-есеп. Емтихан алушының столында 50 емтихан сұрақтары бар. Студент жетінші билетті тәуекелдеп алуға тырысады (алынған билет нөмері 1-ден 50 дейінгі аралықта кездейсоқ түрде құрылуы керек). Сәтсіз алынған билет емтихан алушының столына қайта қайтарылатын болса, студент билетті неше рет алады? Осыдан кейін билеттерді араластырады.

Есепте шартты циклдік процесті ұйымдастыру керек. Цикл жұмысының шарты – жетіге тең емес санды кездейсоқ түрде құру. Цикл

жұмысын іске қосу алдында 1-ден 50 дейінгі аралықта орналасқан, кездейсоқ бірінші а санын құру және талпыныстар санауышына бір санын жазу керек (c=1). Цикл денесінде а айнымалысы үшін кездейсоқ санның функциясы және талпыныстар санауышына (c айнымалысы) арналған инкремент операциясы болуы керек.

Цикл жұмысы аяқталғаннан кейін экран мониторына санауыш мәнін - талпыныстар санын шығаруды ұйымдастыру керек.

Бағдарлама коды мынандай:

```
using System;
namespace ConsoleCiklWhile
{
 class Program
 {
 static void Main()
 {
  int a, c;
  Random rnd = new Random();
 vvod:
  a = rnd.Next() % 50 + 1;
  c = 1;
  while (a != 7)
  {
  a = rnd.Next() % 50 + 1;
  c++;
  }
  Console.WriteLine("Билет {0}-rette tabildi", c);
  Console.WriteLine("Zhalgastiry(Y/N)?");
  char cim = (char)Console.Read();
  Console.ReadLine();
  if (cim == 'Y') goto vvod;
  Console.WriteLine("Enter pernesin basiniz");
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

Билет 34-rette tabildi
Zhalgastiry(Y/N)?
Y
Билет 35-rette tabildi
Zhalgastiry(Y/N)?
Y
Билет 54-rette tabildi
Zhalgastiry(Y/N)?
N
Enter pernesin basiniz

`while` циклінің операторы алғыш артты цикл операторы деп аталады. Яғни, циклдің жұмыс шартын оны іске қосқанға дейін тексеру керек. Сондықтан, қарастырылған есепті шешу алгоритмінде шартты тексеруді ұйымдастыру үшін, цикл іске қосылғанға дейін бірінші сан құрылуы керек. Бұл цикл денесінің екі рет жазылуына алып келеді – цикл іске қосылғанға дейін және оның ішінде. Егер цикл денесінде екі оператор болса, онда бұл қолайсыздық онша білінбейді, бірақ көптеген операторлардың қайталануы дұрыс емес және бағдарламаның көрнекілігін бұзады (оқу құралының келесі бөлімін қараңыз).

Әдетте шартты циклдер шексіз қатарлармен берілген өрнектерді есептеу үшін қолданылады, мысалы `sin(x)` функциясын (мысал [5] алынған)

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots = \sum_{i=1}^{\infty} (-1)^i \frac{x^{2i+1}}{(2i+1)!} \qquad (1)$$

Шеткі және шексіз қосындыларды есептеу алгоритмі әдетте жоғарғы деңгейдегі тілдегі бағдарламалауды оқытатын көптеген пәндерде қарастырылады. Сондықтан дәстүр бойынша `sin(x)` функциясын есептеу бағдарламасының коды мен алгоритімін қарастырайық.

`sin(x)` функциясын есептеу формуласы шексіз қосындылардан тұрады.

Математика үшін шексіздік ұғымын қолдану - әдеттегі құбылыс – бүтін және нақты сандар жиынтығы шексіз. Бағдарламалауда барлық бүтін және нақты сандардың мәндері компьютер мүмкіндігімен шектелген. Сондықтан бағдарламалауда шексіз қосындыларды есептеулер орындалмайды. Алайда қосындыны берілген дәлдікте есептеуге болады. Сонымен, шексіз қосындылар берілген дәлдіктегі қосындыларға айналдырылады.

Берілген дәлдіктегі ақырғы қосындыны есептеу алгоритмін функционалды түрде келесі әрекеттер ретімен көрсетуге болады:

бастапқы шарттарды беру;

```
while( кезекті қосылғыш модулі берілген дәлдіктен үлкен )
      {
қосынды = қосынды + кезекті қосылғыш;
```

жаңа мәндерді қолданып кезекті қосылғышты есептеу;

```
      }
```

Бастапқы шарт мыналардан тұруға тиісті: цикл денесінің жұмысын қамтамасыз ететін қосынды және кейбір айнымалыларды инициализациялау, бірінші қосылғышты есептеу. Мысалы, кезекті қосылғыштың нөміріне жауап беретін айнымалы.

Рекуррентті формулалар арқылы кезекті қосылғышты есептеуге кеңес беріледі. Кезекті қосылғышты есептеуді алдынғы мәнді қолданып және $a_{k+1}=f(a_k)$ рекуррентті формуласын құрып жеңілдетуге болады.

Осы тәсіл 1.1 формуласы бойынша `sin(x)` функциясын және басқа да көптеген математикалық функцияларды есептегенде қолданылады.

1.1 формуласы бойынша керекті рекуррентті қатынастардың құрылуын көрсетейік.

$$a_0 = \frac{(-1)^0 x}{1!} = x; \quad \ldots \quad a_k = \frac{(-1)^k x^{2k+1}}{(2k+1)!}; \quad a_{k+1} = \frac{(-1)^{k+1} x^{2k+3}}{(2k+3)!} \quad \ldots \qquad (2)$$

$\frac{a_{k+1}}{a_k}$ қатынасын есептеп керекті рекуррентті арақатынасын аламыз:

$$a_{k+1} = a_k \frac{-x^2}{(2k+2)(2k+3)} \qquad (3)$$

Рекуррентті арақатынасты енгізу нәтижесінде циклдің әрбір қадамында санның дәрежесін және факториалдарды табудың қажеті жоқ.

4.2-есеп. sin(x)-тің мәнін есепте, x диалог режімінде беріледі. Есептеуді екі жолмен орындау керек: стандартты sin() функциясымен және берілген t дәлдігіне сәйкес (1.1) қатар көмегімен, t мәні диалог режімінде беріледі.

Шыққан рекуррентті арақатынастарды ескере отырып келесі бағдарлама кодын жазуға болады:

```
using System;
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
 {
  double sym, x, t, y;
  string buf;
  int i;
 vvod:
  Console.WriteLine("sin(x) ornegin eseptey");
  Console.Write("x= ");
  buf = Console.ReadLine();
  x = Convert.ToDouble(buf);
  Console.Write("Eseptey daldigin engizy, micali, 0,0001
");
  buf = Console.ReadLine();
  t = Convert.ToDouble(buf);
  sym = 0; y = x; i = 0;
  while (Math.Abs(y) > t)
  {
  sym = sym + y;
  y = y * (-x * x) / ((2 * i + 2) * (2 * i + 3));
```

```
  i++;
  }
  Console.WriteLine("sin(x) = {0} ", Math.Sin(x));
  Console.WriteLine("sym = {0} ", sym);
  Console.WriteLine("Zhalgastiry (Y/N)?");
  char cim = (char)Console.Read();
  Console.ReadLine();
  if (cim == 'Y') goto vvod;
  Console.WriteLine("Enter pernesin basiniz");
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:
sin(x) ornegin eseptey
x= 0,35
Eseptey laldigin engizy, micali, 0,0001 0,00001
sin(x) = 0,342897807455451
sym = 0,342897934895833
Zhalgastiry (Y/N)?
Y
sin(x) ornegin eseptey
x= 1
Eseptey laldigin engizy, micali, 0,0001 0,000001
sin(x) = 0,841470984807897
sym = 0,841471009700176
Zhalgastiry (Y/N)?

Диалог режимінде әр есептеу үшін әр түрлі дәлдіктер берілген, ол есептеу нәтижелерінің ұқсас разрядтар санын анықтайды.

Қарастырылған алгоритмді құру әдістемесін көптеген есептердің (рекуррентті арақатынастарын есептеуге болатын) ақырғы қосындысын есептеуде қолдануға болады.

Кейбір математикалық есептерде күрделі алгебралық өрнектер бірнеше қарапайым өрнектерге бөлінеді. Әрбір қарапайым өрнектек үшін өз рекуррентті арақатынасын есептеуге болады. Алгебралық өрнектерді бөліктерге бөлу (декомпозиция) және рекуррентті қатынастарды қолдану күрделі математикалық есептерді шешуге арналған алгоритмдерді құруға мүмкіндік береді.

4.2 do – while циклінің операторы

do – while циклін тексеруді цикл денесі орындалғаннан кейін жүргізу керек болған жағдайда қолдану орынды, мысалы 4.1 есебінің алгоритмін құру үшін. Циклді жазу пішімі:

```
    do
{ операторлар; }
while (шарт),
```

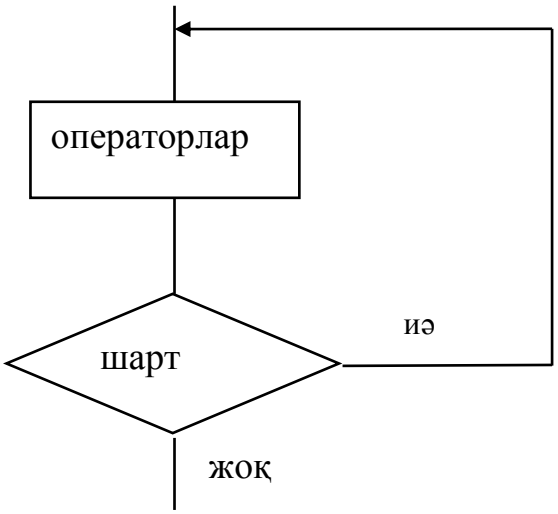
мұнда,

`шарт` – логикалық өрнек, оның екі мәні болуы мүмкін: жалған немесе ақиқат.

егер «шарт» ақиқат болса, онда do – while циклінің «операторлары», әйтпесе цикл орындалмайды (аяқталады).

do – while циклі соңғы шарты цикл деп аталады - шарт цикл денесі орындалғаннан кейін тексеріледі.

Келесі суретте есептің шешімі бойынша алгоритмнің құрылымдық схемасында `do – while` циклі былай көрсетіледі.



4.2-суреті – do – while циклінің графикалық көрінісі

`do – while` циклінің ерекшелігі - цикл денесі кем дегенде бір рет орындалады.

Келесі есепті шығару үшін осы оператордың жұмысын қарастырайық.

4.3-есеп. Қатар қосындысын есептейтін бағдарлама жазу:

$$S = 1 + \frac{1}{x} + \frac{1}{x^2} + \frac{1}{x^3} + \ldots + \frac{1}{x^n} \qquad (4)$$

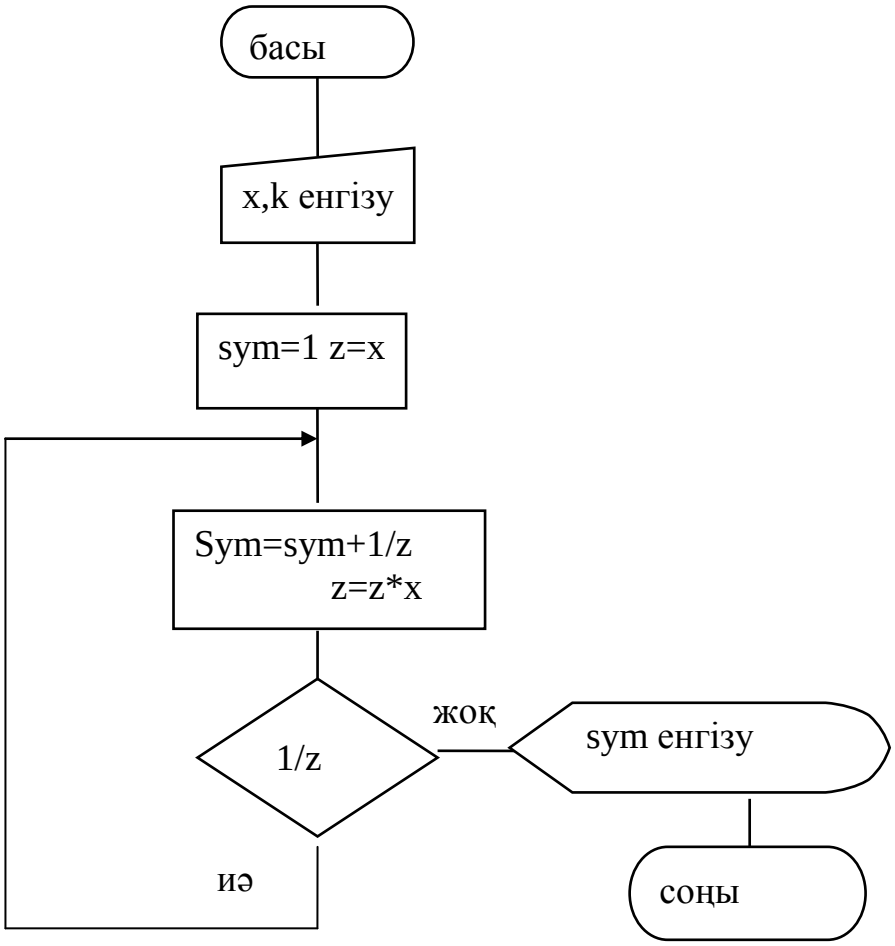
Есептеу n дәрежесіндегі `X`-ке бөлінген бір K санынан кіші болғанға дейін жалғасады, K саны ЭВМ-де диалог режімінде беріледі. `X` мәні де ЭВМ-де диалог режімінде беріледі және ол бірден үлкен болуы керек.

Күрделі математикалық өрнектердің есептеу алгоритмін құру барысында оны жеке бөліктерге бөлу (декомпозиция жасау) дұрыс, мысалы, алымы мен бөлімі. Оны орындау айтарлықтай барлық алгоритмнің құру процесін жеңілдетеді және болатын қателіктер санын азайтады.

Үлкен математикалық өрнекті бір жолмен жазуға тырыспаңыз. Оны жеке үзінділерге бөліңіз. Ол бағдарламаны дұрыстау (отладка) процесін

едәуір қысқартады. Оқыту үлгісі ретінде есептің математикалық өрнегін бөліктерге бөлу (декомпозиция) процесін қарастырайық.

Мысалы, өрнектің бөлімі `X-ке тең Z (Z = X)` айнымалысымен белгіленсе, онда әрбір келесі қосылғыштың бөлімі қатардың алдыңғы бөлімін `X`-ке көбейтіндісіне тең, яғни `Z = Z * X.`



4.5– суреті – 4.3-есептің шешімі бойынша құрылымдық алгоритм

Қатардың қосылғыштарының алымы әрқашан бірге тең, сондықтан ол үшін қосымша айнымалыны қосудың қажеті жоқ.

`do while` циклін «іске қоспас» бұрын өрнекті анықтап алу керек, ол өрнек циклде шарт ретінде қолданылады. Біздің есепте қолданылатын өрнек: $\frac{1}{Z}$`>K` - шарт «ақиқат» «болғанша» цикл орындала береді.

Цикл денесінде қосындыны жинау керек және жаңа `Z` бөлгішін есептеу керек.

Бағдарлама коды мынандай:

```
using System;
namespace ConsoleDoWhile
{
```

```
class Program
{
static void Main()
{
 int n = 0;
 double s = 1, z = 0, y = 0;
vvod:
 Console.WriteLine("x>1 manin engizy");
 string buf = Console.ReadLine();
 double x = double.Parse(buf);
          Console.WriteLine("k < 1 manin engizy");
 string buf = Console.ReadLine();
 double k = double.Parse(buf);

 if (x <= 1 && k >= 0) goto vvod;
 z = x;
 s = 1; n = 0;
 do
 {
 y = 1 / z;
 s = s + y;
 z = z * x;
 n = n + 1;
 }
 while (y > k);
 Console.WriteLine("Kosindi = {0}, cikldin iske kosily
sany = {1}", s, n);
 Console.WriteLine("Enter pernesin basiniz");
 Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
x>1 manin engizy
2,2
k < 1 manin engizy
0,0001
Kosindi = 1,83326850772752, cikldin iske kosily sany = 12
Enter pernesin basiniz
```

4.3 Циклді пайдаланып есепті шешу мысалы

Осы бөлімде тағы бір есептің шешімін қарастырамыз, ол `for` циклімен қатар `while` циклін де қолдануды қажет етеді. Осы есепті шешкеннен кейін, басқа мысалдарда есептің толық шешімін егжей-тегжейлі қарастырмайтын боламыз, бірақ әр түрлі үзінділердің алгоритмдерін оқуды жалғастырамыз, мысалы, сұрыптау, т.б.

4.4-есеп. Үш жүз гектарлы бақшадан картоп өнімін жинау қажет болсын. Жеке тәжірибеге сүйенсек, 840 шұңқыр қазу керек. Бір шұңқырда 0-ден 5 түйнекке дейін (кездейсоқ сан) болуы мүмкін. Түйнектердің көлеміне қарай шелекте оның 70-тен 100 дейін саны болады (кездейсоқ сан). Бір қапқа төрт шелек картоп салынады. Картоп өнімін жинауға бос қаптардың қаншасын алу керек?

Есепті шешу алгоритмі шынайы алгоритмге - картопты қазу кезіндегі адамдардың әрекетіне сәйкес келеді. Біріншіден,барлық картоп (барлық 840 шұңқыр) қазып алынады және кептіру үшін бір үйіндіге жиналады. Ал содан кейін, үйіндідегі картоп біткенге дейін (`while` циклі) оны шелекке толтырады және қапқа төгеді.

Есепті шешу алгоритмін толығырақ сипаттайық. Бірінші кезеңде `for` циклін 840 циклдік операцияларға қосу керек (шұңқырлар саны бойынша), оның ішінде екі әрекетті орындаймыз: бір шұңқырда түйнектердің кездейсоқ санын құрамыз және бақшада қазып алынған түйнектердің жалпы санын жинақтаймыз.

Осылай, `for` циклінің аяғында белгілі бір айнымалыда, мысалы c, біз қазып алынған түйнектердің жалпы санын анықтаймыз.

Шелектердегі өнімді есептеу үшін «әзірше» шартты циклін (үйіндіде картоп таусылғанға дейін) қолдану керек. Цикл ішінде шелектегі түйіндердің кездейсоқ санын құру керек және осы санға түйіндердің жалпы санын азайтуды орындау мен жиналған шелектер санауышын көбейтіп отыру керек.

Осылайша шартты цикл аяқталғаннан кейін жиналған картоп салынған шелектер санын анықтаймыз. Қаптар санын есептеу үшін жиналған шелектер санын төртке бөліп, жауапты дөңгелектеу керек.

Есептелген қаптар саны есептің шешімі болады.

Бағдарлама коды мынандай:

```
using System;
namespace ConsoleFor_While
{
 class Program
 {
 static void Main()
 {
  int i, c, b, m;
  c = 0;
  //int i, j, k, a, max, min;
  //max = -100; min = 100; j = 0; k = 0;
  Random rnd = new Random();
  for (i = 1; i <= 840; i++)
  c = c + rnd.Next() % 6;
  Console.WriteLine("Barligi    {0}    kartoptin    tyinegi
zhinaldi", c);
  b = 0;
  while (c > 0)
  {
```

```
  c = c - (rnd.Next() % 31 + 70);
  b++;
  }
  if (b % 4 == 0)
  m = (int)(b / 4);
  else m = (int)(b / 4) + 1;
  Console.WriteLine("Barligi {0} shelek zhinaldi", b);
  Console.WriteLine("{0} kap kerek", m);
  Console.WriteLine("Enter pernesin basiniz");
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:
Barligi 2039 kartoptin tyinegi zhinaldi
Barligi 24 shelek zhinaldi
6 kap kerek
Enter pernesin basiniz

4.4 Өзін-өзі тексеру сұрақтары

1 Қандай жағдайда бағдарламада `while` циклін қолдану орынды?
2 `while` циклі қандай типке жатады?
3 `do {…}while` циклі қандай типке жатады?
4 "өрнектің" қандай мәнінде мына цикл тоқтатылады?

```
while ("выражение") { . . . }
```

5 "өрнектің" қандай мәнінде мына цикл тоқтатылады?

```
do { . . . } while ("выражение")?
```

6 Бағдарламаның келесі үзіндісі нені шығарады?

```
  k = 5; I = 0;
  while (I < k)
  {
  I = I + 2;
  k++;
  }
Console.WriteLine("I = {0} k = {1} ",I,k);?
```

7 Цикл аяқталғаннан кейін `I` айнымалысының мәні қандай?

```
  I = 0;
  while (I != 10)
  {
  I++;
  . . .
  }
```

```
Console.WriteLine("I = {0} ", I);?
```

8 Бағдарламаның келесі үзіндісі экранға нені шығарады?

```
  I = 0;
  while (I != 10)
  for (I= 1;I<=9;I++) k++;
Console.WriteLine("I = {0}",I); ?
```

9 Бағдарламаның келесі үзіндісі экранға нені шығарады?

```
  I = 0;
  while (I != 10)
  {
   j = 0;
   for (I = 1; I<=9; I++) j++;
  }
Console.WriteLine("j = {0} ", j); ?
```

10 Бағдарламаның келесі үзіндісі экранға нені шығарады?

```
  j = 1; I = 0;
  while (I < 5)
  {
  I++;
  j = j*2;
  }
Console.WriteLine("j = {0} ", j); ?
```

5 МАССИВТЕР

5.1 Сілтемелік тип туралы мағлұмат

C# тілінде айнымалылардың екі типі бар – мәнді және сілтемелік. Мәнді типтегі айнымалылар сәйкес типте деректер мәнін сақтауға қабілетті. Сілтемелік айнымалылар компьютер жадының физикалық адресін сақтайды, олар арқылы компьютер жадында деректердің сәйкес мәндері орналасады.

Деректерді сақтау үшін мәнді типтегі айнымалыларға жады әдетте бағдарлама компиляциясы кезінде, ал сілтемелік типті айнымалыларға `new` операторының көмегімен бағдарлама орындалу барысында бөлінеді.

Біз бұдан бұрынғы бағдарламаларда сілтемелік типтегі айнымалыны - кездейсоқ сандарды дайындайтын `Random` класының (`rnd` объектісі) айнымалысын қолданғанбыз.

Айта кету керек сілтемелік тип әдетте құрылымы күрделі деректерді, яғни массивтер, жолдар, кластар типіндегі деректерді ұйымдастыру үшін қолданылады. Барлық осы деректер үйінді (куча) деп аталатын компьютердің арнайы жадында сақталады. Мәнді типтегі барлық айнымалылар бағдарламалық стекте орналасады, ал сілтемелік типті айнымалылар - операциялық жүйе басқаратын үйіндіде.

Сондықтан сілтемелік типті айнымалыларға қарағанда мәнді типтегі айнымалыларға қатынау (қол жеткізу) үшін біраз уақыт жұмсалады, бірақ мәнді типті айнымалыларға қарағанда сілтемелік типті айнымалыларға жады көлемі бір деңгейге көбірек бөлінеді.

Кейде мәнді типтегі айнымалылар үшін компьютер жадын үлестіруді жадының статикадық үлестіруі деп атайды, яғни компьютер жадын бөлу үдерісі бағдарлама компиляциясы кезінде орындалады. Ал сілтемелік типті айнымалыларға компьютер жадын үлестіруді жадының динамикалық үлестіру деп атайды, яғни компьютер жадын бөлу үдерісі бағдарламаның орындалуы барысында орын алады.

Ескере кету керек, сілтемелік типті айнымалыларға компьютер жадын бөлу `new` операторы арқылы орындалады.

5.2 Массив ұғымы

Массив дегеніміз – бір айнымалы арқылы бір типтегі айнымалылар тобын сипаттау формасы. Массивке біріктірілетін барлық айнымалылар 0-ден `n`-ге дейін нөмірленеді және әрбір айнымалыға өз нөмірі - индекс сәйкес қойылған. Нақты бір массив айнымалысын қолдану үшін массив

аты мен айнымалы индексін көрсету керек. Сондықтан массив айнымалысын индексті айнымалы деп жиі атайды.

Бір өлшемді массивтер (векторлар) болады, мысалы, атаулар массиві, бір емтихан бағаларының массиві, туған күндер массиві және т.б.

Екі өлшемді массивтер (матрицалар немесе кестелер) болады, мысалы, футбол бойынша чемпионат ойынының нәтижелері, студенттердің емтихандар нәтижелері бойынша қорытынды кестесі және т.б.

Көп өлшемді массивтерге барлық қалған массивтер жатады.

Бағдарламада массивті сипаттау екі кезеңнен тұрады: массивті жариялау және массивтің жұмысын бастау.

массивті жариялауда айнымалы типі мен аты анықталады, мысалы:

int[ ] masi;

double[ ] masf; ,

мұнда бірінші жолда біз аты `masi` болатын бүтін айнымалылар массивін қолданамыз, ал екінші жолда аты `masf` болатын нақты айнымалылар массивін жариялаймыз.

Бұл орайда бағдарлама компиляциясы барысында бағдарлама стегіне сілтемелік типті массивтердің аттары жазылады. Массив айнымалыларына компьютер жадысы массив жұмысын сипаттағаннан бастап бөлінеді, мысалы,

masi = new int[10];

masf = new double[20] ; ,

мұнда квадрат жақша ішіндегі мәндер массив өлшемін анықтайды (0 . элементтер саны -1 – массив элементтерінде қолдануға болатын индекстер мәндерінің ауқымы).

Массивті инициализациялау кезеңінде үйіндіде массив объектісі құрылады және оның барлық айнымалыларына «нөлдік» мәндер меншіктеледі (оның инициализациясы орындалады).

Айнымалылардың «нөлдік» мәндері сәйкес:

– сандық айнымалылар үшін нөл болады;

– жолдық айнымалылар үшін бос жолдар болады;

– символдық айнымалыларда символ болмайды.

Массивті инициализациялаудан кейін оның айнымалыларын бағдарламада қолдануға болады.

Массивтерді инициализациялау мен жариялаудың әр түрлі нұсқаларын түрлі есептерді шешу мысалдарында қарастырайық.

5.3 Әр түрлі есептерде массивтерді қолдану мысалдары

Массивті қолдануды қажет ететін көптеген есептер бар. Мәтін символдар массиві болғандықтан есептердің бір бөлігі мәтінді өңдеуге байланысты болады.

Математикалық немесе физикалық есептер, мысалы, эксперименттік мәндер массивімен көрсетілген функцияның орта мәнін есептеу.

Информатикада, мысалы, аттардың тізімін алфавит бойынша реттеу немесе массивте іздеу есептері, т.б.

Массивті қолдануды қажет ететін кейбір есептердің шешімін қарастырайық.

5.1-есеп. Эксперименттік функция 12-кестедегі мәндер арқылы берілген. Осы функцияның барлық максимумын табу керек. Есептеулерді жүргізу үшін функцияның келесі мәндерін қолдану керек: 1.05, 3.17, 5.24, 4.38, 6.42, 4.93, 6.59, 7.84, 5.73, 5.14, 4.87 және 3.18.

Есепті шешу алгоритмі келесі қадамдардан тұрады:

– функцияның бірінші мәнін максимумға тексеру – егер функцияның екінші мәні бірінші мәнінен кіші болса, онда функцияның бірінші мәні максимум болып есептеледі;

– максимумдарды іздестіру - егер функцияның алдыңғы және кейінгі мәндері ағымдағы мәнінен кіші болса, онда функцияның ағымдағы мәні максимум болып есептеледі;

– функцияның соңғы мәнін максимумға тексеру - егер функцияның соңғы мәні соңғының алдында тұрған мәнінен үлкен болса, онда функцияның соңғы мәні максимум болып есептеледі;

Бұл алгоритмде функцияның бірнеше бірдей мәндерінің болу жағдайын қарастырмаймыз. Өмірде мұндай жағдайдың болу мүмкіндігі аз. Есепте осындай жағдайлар орын алатын болса, шешімнің алгоритмін өз бетімен құра аласыз.

Бағдарламаның коды:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
 {
  int i;
  double j, n;
  double[] mas = {1.05, 3.17, 5.24, 4.38, 6.42, 4.93,
  6.59, 7.84, 5.73, 7.14, 3.87, 2.18 };
  //әдістемелік тұрғыдан мына түрде жазу дұрыс
  //float[] mas;
  //mas = new float[12] { 1.05, 3.17, 5.24, 4.38, 6.42,
  // 4.93, 6.59, 7.84, 5.73, 6.14, 4.87, 3.18 };
  //массивті жариялауды және инициализациялауды бірге
қатар қолдануға болады, мысалы:
  int[] max = new int[7];
  //максимумдар индекстерінің массиві
  // массивті "графикалық" түрде көрсетейік
  for (i = 0; i <= 11; i++)
  {
  Console.Write(i);
```

```
  n = Math.Round(mas[i]);
  j = 0;
  while (j <= n) { Console.Write(" "); j++; }
  Console.WriteLine('*');
  }
  int k; // максимум индексі
  k = 0;
  // функцияның бірінші мәнін максимумге тексеру
  if (mas[0] > mas[1]) { max[k] = 0; k++; }
  //максимумдерді іздеу
  for (i = 1; i < 11; i++)
  if ((mas[i - 1] < mas[i]) && (mas[i] > mas[i + 1]))
  { max[k] = i; k++; }
  //функцияның соңғы мәнін максимумге тексеру
  if (mas[10] < mas[11]) { max[k] = 11; k++; }
  //функция максимумдерін экранға шығару
  Console.WriteLine("Funkzia Maksimymi:");
  for (i = 0; i < k; i++)
  Console.WriteLine("{0} - {1}", max[i], mas[max[i]]);
  Console.ReadLine();
 }
 }
}
```

Бағдарламаның жұмысы:

```
0 *
1 *
2 *
3 *
4 *
5 *
6 *
7  *
8 *
9 *
10 *
11 *
Funkzia Maksimymi:
2 - 5,24
4 - 6,42
7 - 7,84
9 - 7,14
```

Келтірілген бағдарламада массивтерді жариялау мен инициализациялаудың бірнеше нұсқасы қарастырылған.

Айнымалылардың аз саны үшін массивтерді жариялау мен инициализациялауды қатарынан орындауға болады, әсіресе егер массив элементтерінің мәндері бағдарламада анықталмаса - массив элементтері массив константаларымен ұсынылса. Мысалы:

double[] mas = { 1.05, 3.17, 5.24, 4.38, 6.42, 4.93, 6.59, 7.84, 5.73, 6.14, 4.87, 3.18 };

Массивті жариялау мен инициализациялаудың екінші нұсқасы бірінші массив типіндегі айнымалыны жариялауды, ал содан кейін инициализациялауды талап етеді.

Мысалы:

float[] mas;

mas = new float[12] { 1.05, 3.17, 5.24, 4.38, 6.42, 4.93, 6.59, 7.84, 5.73, 6.14, 4.87, 3.18 };

Үшінші нұсқада массивті жариялау мен инициализациялау бір әрекетте (амалда) орындалады. Мысалы:

int[] max = new int[7];

Келесі есеп ақпарттық есептер тобына жатады.

5.2-есеп. 20 студенттен тұратын топ емтихан тапсырды. Бағалар 2-ден 5-ке дейін кездейсоқ түрде құрылады. 5, 4, 3 және 2 бағаларын алған студенттердің санын анықтау керек.

Есепті шешу алгоритмі келесі қадамдардан тұрады:

–циклде 1-ден 20-ға дейін кездейсоқ түрде 20 студентке емтихан бағаларының массивін құрамыз;

– бағалар массивін қайта қарау барысында 5, 4, 3 және 2 санын есептейміз;

– монитор экранына бағдарлама жұмысының нәтижесін шығарамыз.

Бағдарламаның коды:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
 {
int i,j;
  int[] a = new int[20];
  int[] b = new int[6];
  Random rnd = new Random();
 // 20 студенттің бағасын кездейсоқ түрде құрамыз
 // монитор экранына шығарамыз
  Console.Write("Studentterdin bagalari: ");
  for (i=0;i<=19;i++)
  {
  a[i] = rnd.Next()%4 + 2;
  Console.Write(" {0}", a[i]);
  }
  Console.WriteLine();
//5, 4, 3 және 2 сандарын есептейміз, нәтижелерді b массивіне жазамыз
  for (i=0;i<=19;i++)
  {
```

```
        if (a[i]==2) b[2]++;
        if (a[i]==3) b[3]++;
        if (a[i]==4) b[4]++;
        if (a[i]==5) b[5]++;
        }
      // бағдарлама жұмысын экранға шығарамыз
        for (i=2;i<=5;i++)
        Console.WriteLine(" {0} bagasin {1} student aldi",i,
b[i]);
        Console.ReadLine();
       }
       }
      }
```

Бағдарлама жұмысы:

Studentterdin bagalari: 5 2 2 5 2 5 4 2 2 3 5 4 4 3 5 3 4 3 2 2
2 bagasin 7 student aldi
3 bagasin 4 student aldi
4 bagasin 4 student aldi
5 bagasin 5 student aldi

`шарттары тексерілетін for` циклінің операторын келесі үзіндімен ауыстыруға болады:

```
for (i=0;i<=19;i++)
  for (j = 2; j <= 5; j++)
  if (a[i] == j) b[j]++;
```

немесе

```
for (i=0;i<=19;i++)
  b[a[i]]++;
```

Кездейсоқ санды құрғаннан кейін бірден барлақ тексерулерді бір циклде жазуға болады:

```
for (i=0;i<=19;i++)
  {
  a[i] = rnd.Next()%4 + 2;
  b[a[i]]++;
  Console.Write(" {0}", a[i]);
 }
```

Осы есептің шешімі бойынша алгоритмді орындаудың бірнеше нұсқа қарастырылды, бірақ алгоритмді орындаудың соңғы нұсқалары айқын емес.

Осы есептің берілген кодын қысқартып жазу өте қызықты, бірақ осы пәннің дәрістерінде ол қарастылмайды.

Оқу құралында біз есепті шешу алгоритмінің бағдарламалық кодтарының ішінен ең қысқа нұсқасының орнына алгоритмнің жүзеге асырылуы айқын нұсқасын қарастырамыз.

Бағдарлама кодының соңғы нұсқасында b массивінің индекс мәні студенттің емтиханда алған бағасымен анықталады.

Егер массив индексі күрделі алгоритм немесе кездейсоқ құрылатын болса, онда массивті осы түрдегі қолдану ассоциативті қолдану деп аталады.

Математикада массивтер полином коэффициенттерін сипаттағанда қолданылады.

Мысалы, $P_n(x)$ полиномының n-і дәрежелі коэффициенті әдетте мына функция түрінде жазылады

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 \qquad (5)$$

a[n+1] массивімен көрсетуге болады, онда массив индексі 0-ден n-ге дейін ауысады.

математиканың міндеттерінің бірі – x айнымалысының мәндерінің белгілі бір үзіндісінде n-і дәрежелі полином түрінде көрсетілген функция өзгерісін зерттеуі.

5.3-есеп. 10-дәрежелі полином коэффициенттерінің массиві берілген. (n = 10) – 1.05,3.17,5.24,4.38,6.42,4.93,6.59,7.84,5.73,7.14,3.87,2.18.

Индексі 0 болатын массивтің мәні полиномның нөлінші коэффициентіне тең және т.б.

Полиномның тәртібін 0-ден 1-ге дейін 0.1 қадамымен зерттеу (x айнымалысы 0-ден 1-ге дейін 0.1 қадамымен өзгереді).

Монитор экранына полином өзгерісін кесте және график түрлерінде шығарыңыз.

Бағдарламаның коды:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main(string[] args)
 {
  double j, n, x, y, p;
  int i, k;
  double[] mas = {1.05, 3.17, 5.24, 4.38, 6.42, 4.93,
 6.59, 7.84, 5.73, 7.14, 3.87, 2.18 };
  double[] masf = new double[11]; //полином мәндерінің
массиві
  Console.WriteLine("Polinomdi grafikalik tyrde
korsety:");
  k = 0;
  for (x = 0; x < 1; x = x + 0.1)
  {
  y = x; p = mas[0];
  for (i = 1; i <= 11; i++)
  {
   p = p + y * mas[i];
   y = y * x;
```

```
  }
  masf[k] = p; k++;
  Console.Write(x);
  n = Math.Round(p);
  j = 0;
  while (j <= n) { Console.Write(" "); j++; }
  Console.WriteLine('*');
  }
  Console.WriteLine("Polinomdi keste tyrinde korsety:");
  i = 0;
  for (x = 0; x < 1; x = x + 0.1)
  {
  Console.WriteLine("x = {0} P = {1}", x, masf[i]);
  i++;
  }
  Console.ReadLine();
 }
 }
}
```

Бағдарламаның жұмысы:

```
Polinomdi grafikalik tyrde korsety:
0 *
0,1 *
0,2 *
0,3 *
0,4 *
0,5 *
0,6  *
0,7  *
0,8   *
0,9     *
1        *
Polinomdi keste tyrinde korsety:
x = 0 P = 1,05
x = 0,1 P = 1,4244787388488
x = 0,2 P = 1,9410304774144
x = 0,3 P = 2,6619038136876
x = 0,4 P = 3,6975170516992
x = 0,5 P = 5,253203125
x = 0,6 P = 7,7263077961728
x = 0,7 P = 11,9008032176404
x = 0,8 P = 19,3213982427136
x = 0,9 P = 32,9819936534232
x = 1 P = 58,54
```

Полиномды есептеу алгоритмінде `n` дәрежедегі `x` айнымалысын рекуррентті есептеу қолданылады. Ол үшін `y` айнымалысына `x` мәні меншіктелген және 1 индексті қосылғыш полиномды есептеу (вычисление слагаемого полинома с индексом 1) орындалған. Одан кейін `y` `x`-ке көбейтіліп (x^2 мәнін аламыз), екінші қосылғыштың мәні есептеледі, т.б..

5.4 Динамикалық массивтер

C# тілінде массивтер элементтерінің мәндері динамикалық болып табылады, яғни оларға жады бағдарлама жұмысының үдерісінде new операциясының көмегімен «үйіндіде» бөлінеді. C# тілінің осындай ерекшелігі бағдарламаларда динамикалық массивтер құруға мүмкіндік берді, яғни олардың элементтер саны бағдарлама жұмысының үдерісінде анықталатын массивтер. Көптеген есептердің тобы бар, онда массив өлшемі бағдарлама жұмысының үдерісінде анықталады немесе пайдаланушы диалогында берілуі тиіс, мысалы, каталогтағы файлдар немесе қоймадағы тауарлар саны.

Таза синтаксис жағынан статикалық және динамикалық массивтерді жариялауда маңызды айырмашылық жоқ. Шынында массивтерді жариялауда сандық сипаттамалар болмайды, мысалы, `double[] mas;`, ал массивті инициализациялау бағдарлама жұмысы уақытында new операциясы арқылы орындалады. Егер диалогта массив элементтер саны берілсе, динамикалық массивті жариялау мен инициализациялауды бірге орындауға болады. Мысалы:

```
Console.WriteLine("masi massivinin elementterin engiz ");
int size = int.Parse(Console.ReadLine());
int[] masi = new int[size];
```

C# тілінде динамикалық массивтер болып бір өлшемді массивтер ғана болады. Динамикалық массивтер элементтерінің санына сай келетін айнымалылардың мәні оларды инициализациялауға дейін анықталуы керек.

5.4-есеп. Топ студенттері «C# тілінде бағдарламалау» пәні бойынша емтихан тапсырады. Топтағы студенттер саны диалог режімінде беріледі (20 көп емес). Бағалар (балл түрінде анықталады) 20-дан 100 дейінгі аралықта кездейсоқ түрде құрылады. Топтың емтихан бағасын динамикалық массивте ұйымдастыру керек. Емтихан нәтижесін шығарыңыз.

Бағдарламаның коды:

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
 {
  Console.WriteLine("20-dan aspaitin noptagi studentterdin
sanin engiziniz");
  int size = int.Parse(Console.ReadLine());
  int[] masi = new int[size];
```

```
  int i;
  Random rnd = new Random();
  for (i = 0; i < size; i++)
  Console.Write(" {0,3}", i + 1);
  Console.WriteLine();
  for (i = 0; i < size; i++)
  {
  masi[i] = rnd.Next() % 81 + 20;
  Console.Write(" {0,3}", masi[i]);
  }
  Console.WriteLine();
  Console.WriteLine("Enter pernesin basiniz");
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

20-dan aspaitin noptagi studentterdin sanin engiziniz

15

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

76 72 28 63 72 90 95 58 39 38 60 56 87 92 91

Enter pernesin basiniz

Массивтің әрбір мәнін шығарғанда оның позициясының саны көрсетіледі – `{0,3}`, ол бағдарлама жұмысының нәтижесін ыңғайлы түрде көрсетуге мүмкіндік береді.

5.5 Өзін-өзі тексеру сұрақтары

1 Мәнді типтегі айнымалы туралы түсінік.

2 Сілтемелік типтегі айнымалы туралы түсінік.

3 Массивте қандай айнымалыларды орналастыруға болады ?

4 Бүтін типті айнымалылар үшін бір өлшемді массивті жариялаудың дұрыс нұсқасын қалай жазуға болады?

5 Бүтін типті 10 айнымалы үшін бір өлшемді массивті инициализациялаудың дұрыс нұсқасын қалай жазуға болады?

6 Бағдарламаның келесі үзіндісі нені орындайды:

for (I = 0; I <= 10; I++)

Console.Write(" {0} \t", a[I]); ?

7 `a[11]` массиві үшін бағдарламаның келесі үзіндісі нені орындайды **:**

for (I = 1; I <= 10; I++)

{k = a[I]; a[I] = a[11 - I]; a[11 - I] = k;} ?

8 Бағдарламаның келесі үзіндісі нені орындайды **:**

k = a[0];

```
for (I = 0;I <= 10; I++)
if (a[I] > k) k = a[I];
Console.WriteLine(" {0} ", k); ?
```

9 Бағдарламаның келесі үзіндісі нені орындайды:

```
for (I = 0; I <= 9; I++)
{ k = a[I]; a[I] = a[I+1]; a[I+1] = k;} ?
```

10 Бағдарламаның келесі үзіндісі нені орындайды:

```
I = 0;
while (I <= 10)
{
a[I] = a[I] * (-1);
I++;
}?
```

6 МАССИВТЕРДІ ӨҢДЕУ АЛГОРИТМДЕРІ

6.1 Алгоритмдер мен массивтер

Көптеген есептерде деректерді массив түрінде көрсету оның шешімін едәуір жеңілдетеді. Кейбір есептер «Жоғарғы математика» курсынан сізге белгілі, мысалы, полиномды есептеу.

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 \qquad (6)$$

функциясы n-і дәрежелі $P_n(x)$ полином деп аталады. Мұндағы a_k мәні полином коэффициенті деп аталады және `n+1` элементтері бар бір өлшемді массив түрінде көрсетіле алады.

$P_n(x)$ полиномының коэффициенттер массивін біле тұра полином мәнін кез келген `x` нүктесінде есептеуге болады. Әдетте `x` нүктесінде полиномды есептеу есебі кейбір алгоритмдерді жүзеге асырумен байланысты, мысалы, Горнер алгоритмі, кесіндіні тең бөлу әдісі (егер кесіндінің ұштарында полином мәні түрлі таңбада болса), итерация әдісі, Ньютона, Лагранж әдістері және т.б. Барлық осы алгоритмдер полином коэффициентінің массивін өңдеумен байланысты.

Информатиканың өзінің есептері бар, олар деректерді массив түрінде көрсетуді талап етеді. Осы деректерді өңдеудің өз алгоритмдері бар. Деректерді өңдеудің көптеген алгоритмдерінің ішінен сұрыптау (деректерді кейбір сипаты бойынша реттеу) және іздеу (деректерде берілген сипат бойынша элементті анықтау) алгоритмдері ерекшеленеді.

Математикалық есептерді шешу алгоритмі (әдісі) «Жоғарғы математика» курсында қарастырылады, сондықтан оқу құралының осы бөлімінде кейбір деректерді сұрыптау мен іздеу алгоритмдері ғана қарастырылады.

6.2 Массив элементтерін сұрыптау

Біз осы бөлімде массив элементтерін сұрыптаудың тек екі әдісінің алгоритмдерін және бағдарламалақ жүзеге асырылуын қарастырамыз.

6.2.1 Таңдау әдісімен сұрыптау алгоритмі. Массив элементтерін таңдау әдісімен, кему тәртібінде сұрыптау алгоритмі келесі амалдардың орындалуын болжайды.

Массивтің бірінші элементін қарастырамыз, оның мәнін массивтің қалған басқа элементтерімен кезекпен салыстырамыз (2-ден соңғы элементке дейін). Егер мәні үлкен элемент кезіксе, онда ол бірінші элементпен орын ауыстырады. Массивті бірінші рет тексерудің нәтижесінде бірінші элемент ең жоғары мәніне ие болады.

Содан кейін екінші элементті қарастырамыз, оның мәнін массивтің қалған басқа элементтерімен кезекпен салыстырамыз (3-ден соңғы элементке дейін). Массивті екінші рет тексерудің нәтижесінде біз сұрыпталған массивтің екінші элементінің мәнін анықтаймыз. Тексеру массивтегі соңғының алдында тұрған элементіне дейін осылай қайталанады, соңғы тексеруде массивтің алдынғы элементі массивтің соңғы элементімен ғана салыстырылады.

Алынған массив элементтерінің мәні кему тәртібінде реттеледі.

Массив элементтерінің саны N-ге тең болса, алгоритмде орындалатын салыстыру операцияларының саны (О.С.) келесі түрде есептеледі:

К.О. = (N-1)+(N-2)+(N-3)+ . . . + 2 +1=N·(N-1)/2 (7)

Осы өрнек алгоритмнің есептеу тиімділігін сипаттайды.

6.1-есеп. 11 кездейсоқ бүтін сандардан тұратын, минус 50-ден 50-ге дейінгі аралықта a массивін құр. Оны шығару керек. Массив элементтерін кему тәртібінде сұрыптауды орындау және жаңа массивті шығару.

Массив элементтерін кему тәртібінде сұрыптау алгоритмінің бағдарламалық іске асырылуын қарастырайық.

Бағдарламаның коды:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
 { int i, j, b;
  int[] a = new int[11];
  Random rnd = new Random();
  // массивті құру және экранға шығару
  Console.Write("Massivti syriptaganga dein: ");
  for (i = 0; i <= 10; i++)
  {
  a[i] = rnd.Next() % 101 - 50;
  Console.Write(" {0}", a[i]);
  }
  Console.WriteLine();
  //таңдау әдісі арқылы массив элементтерін сұрыптау
  for (i = 0; i <= 9; i++)
  for (j = i + 1; j <= 10; j++)
   if (a[i] < a[j])
   { b = a[i]; a[i] = a[j]; a[j] = b; }
  //сұрыптаудан кейін массивті экранға шығару
  Console.Write("Massivti syriptagannan kein: ");
  for (i = 0; i <= 10; i++)
  Console.Write(" {0}", a[i]);
  Console.WriteLine();
  Console.ReadLine();
 }
 }
```

```
    }
```

```
    Бағдарлама жұмысы:
    Massivti syriptaganga dein: 11 16 47 9 5 14 41 18
-2 -31 49
    Massivti syriptagannan kein: 49 47 41 18 16 14 11
9 5 -2 -31
```

6.2.2 «Көпіршікті» әдіспен сұрыптау алгоритмі. Массив элементтерін өсу тәртібінде «көпіршікті» әдіспен сұрыптау алгоритмі келесі әрекеттердің орындалуын қарастырады.

Массивтің бірінші элементін екіншісімен салыстырамыз және егер біріншісі үлкен болса, онда олар орындарын алмастырады. Одан кейін массивтің екінші элементі үшіншісімен салыстырылады, егер екінші элемент үлкен болса, онда олар орындарын алмастырады және т.с.

Массивті бірінші рет «қарап шығудың» нәтижесінде массив элементінің ең үлкен мәні ең соңына жазылады.

Массивтің бірінші элементін қайтадан аламыз және оны екінші элементпен салыстырамыз – барлық процесті массивтің соңғы тұрған элементтің алдындағы элементке дейін қайталаймыз - оның мәнін құрамыз.

Соңғы салыстыру операциясына дейін осылай қайталанады.

Сонымен массив элементерінің мәні өсу ретімен сұрыпталады.

«Көпіршікті» және таңдау әдістері алгоритмдерінің есептеу тиімділіктері бірдей.

Мысал ретінде алдынғы 6.1-есебін қарастырайық.

Массив элементтерін кему тәртібінде «Көпіршікті» әдіспен сұрыптау алгоритмінің бағдарламалық жүзеге асырылуын қарастырайық.

Бағдарламаның коды:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
 {
  int i,j,b;
  int[] a = new int[11];
  Random rnd = new Random();
  // массивті құру және экранға шығару
  Console.Write("Massivti syriptaganga dein: ");
  for (i = 0; i <= 10; i++)
  {
  a[i] = rnd.Next()%101 - 50;
  Console.Write(" {0}", a[i]);
  }
  Console.WriteLine();
```

```
       //  «көпіршікті»  әдісі  арқылы  массив  элементтерін
сұрыптау
       for (i=0; i<=9; i++)
       for (j=0; j<=9-i; j++)
        if (a[j]<a[j+1])
        { b=a[j];a[j]=a[j+1];a[j+1]=b;}
       // сұрыптаудан кейін массивті экранға шығару
       Console.Write("Massivti syriptagannan kein: ");
       for (i=0; i<=10; i++)
       Console.Write(" {0}", a[i]);
       Console.WriteLine();
       Console.ReadLine();
      }
      }
     }
```

Бағдарлама жұмысы:
Massivti syriptaganga dein: -26 -19 32 35 -19 -44 -35 36 10 -49 -5
Massivti syriptagannan kein: 36 35 32 10 -5 -19 -19 -26 -35 -44 -49

Әлбетте әрбір қарастырылған алгоритмді массив элементтерін өсу немесе кему бойынша сұрыптау үшін қолдануға болады.

Егер басқа сұрыптау алгоритмдерімен танысқыңыз келсе, Д.Кнуттың «Искусство программирования для ЭВМ» кітабын ұсынамыз, «сұрыптау мен іздеу» сұрақтарына арналған 3-том, көлемі 843 бет.

6.3 Массивте мәні бойынша элементтерді іздестіру

Д.Кнут «Искусство программирования для ЭВМ» кітабының жартысын («сұрыптау мен іздеу» сұрақтарына 3-том арналған) кейбір бір типті элементтер жиынтығында (массивте, файлда, т.б.) мән бойынша элементтерді іздестіру сұрақтарына арналған. Бұл бағдарламалауды оқу барысында іздеу алгоритмін білу қаншалықты маңызды екенін көрсетеді.

Түрлі іздеу әдістерінің ішінен олардың тек үшеуінің алгоритмдері мен бағдарламалық жүзеге асырылуын қарастырамыз – массивте элемент мәні бойынша екілік, блоктық және тізбектеп іздеу әдістері. Араластыру арқылы іздеу әдісін алгоритм деңгейінде ғана қарастырамыз.

Бізге массив элементтерінің мәндері емес іздеу алгоритмі ғана қажет, сондықтан бүтін сандардан тұратын элементтер үшін барлық алгоритмдерді қарастырамыз. Біз «іздестіру массивінде», яғни кездейсоқ түрде құрылатын бүтін сандар массивінде ізделетін бүтін санды - «ізделетін кілтті» ұсынамыз.

6.3.1 Тізбектеп іздеу алгоритмі. Элементтің тізбектеп іздеу алгоритмі ізделетін кілтті массивтің барлық элементімен, яғни бірінші элементінен

бастап соңғы элементіне дейін тізбекті түрде салыстыруға негізделген. Егер массивтің соңғы элементі қаралып қойса, онда іздеу аяқталады.

Осы алгоритмді көрсету үшін оны кітаптың керекті бетін іздеу алгоритмімен салыстырады, мысалы, үш жүзінші бет. Әрбір бет бірінші беттен үш жүзінші бетке дейін тізбекті түрде парақталады.

Осы алгоритмнің кемшілігі – массив элементтерінің саны көп болған жағдайда іздеуге айтарлықтай көп уақыт жұмсалады. Іздеу уақыты N санына пропорционал, мұндағы N - массив элементтерінің саны.

Алгоритм артықшылығы - массивте элементтерді кез келген тәртіпте және тәртіпсіз түрде орналастыруға рұқсат беру.

6.2-есеп. Іздестіру массиві 0-ден 99-дейінгі аралықтағы 20 кездейсоқ бүтін сандарынан тұрады. Диалог режімінде кез келген бүтін сан – ізделінетін кілт беріледі. Осы сан (индекстері қандай) іздестіру массивінде неше рет кездесетінін табу керек. Тізбектеп іздеу алгоритмін қолдану.

Алгоритм белгілі болғандықтан бағдарлама кодын құруға кірісеміз.

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
 {
  int i,k,n;
  int[] a = new int[20];
  int[] p = new int[20];
  Random rnd = new Random();
  string buf;
  // массивті құру және экранға шығару
  Console.WriteLine("Izdestiry massivi: ");
  for (i=0;i<10;i++)
  Console.Write(" {0}", i);
  for (i=10;i<20;i++)
  Console.Write(" {0}", i);
  Console.WriteLine();
  for (i = 0; i < 20; i++)
  {
  a[i] = rnd.Next()%100;
  if (a[i]>9) Console.Write(" {0}", a[i]);
  else Console.Write(" {0}", a[i]);
  }
  Console.WriteLine();
  Console.WriteLine("Izdey kiltin engiziniz ");
  buf = Console.ReadLine();
  k = Convert.ToInt32(buf);
  n=0;
 // іздеу кілтіне сәйкес массив элементтерін іздестіру
  for (i=0;i<20;i++)
  if (k == a[i]) {p[n]=i; n++;}
  if (n==0)
Console.WriteLine("Izdey kiltine saikes element jok!!");
```

```
       else
       {
               Console.WriteLine("Izdey kiltine saikes keletin
elementter sani = {0}",n);
               Console.WriteLine("Tabilgan indexter: ");
       for (i=0;i<n;i++)
       Console.Write(" {0}",p[i]);
       Console.WriteLine();
       }
          Console.ReadLine();
      }
      }
     }
```

Бағдарлама жұмысы:

```
Izdestiry massivi:
 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
 43 68 85 72 14 44 45 55 57 30 94 83 52 72 77 92 0 91 29
97
Izdey kiltin engiziniz
91
Izdey kiltine saikes keletin elementter sani = 1
Tabilgan indexter:
 17
```

6.3.2 Блоктық іздеу алгоритмі. Блоктық іздеу алгоритмі іздестіру массивінің элементтерін реттелген түрде орналасыруды талап етеді, мысалы, түйінді элементтерді өсу немесе кему ретімен, алфавиттік тәртіпте, т.б.

Іздестіруді іздестіру массивінің түйінді элементі арқылы ғана орындауға болады, мұндағы түйінді элемент іздестіру массивін реттейді.

Іздестіру массивінің элементтері мәндері бойынша өсу ретімен орналасқан деп жорамалдайық.

Бүкіл іздестіру массиві шартты түрде блоктарға бөлінеді, мысалы, бір блокта жүз элементтен. Сұратудың ізделетін кілті ретті түрде бірінші блоктан бастап әрбір блоктың соңғы элементімен салыстырылады.

Егер сұратудың ізделетін кілті кезекті массив блогының соңғы элементінен үлкен болса, онда келесі блоктың соңғы элементіне көшу керек, әйтпесе соңғы салыстыру орындалған блокта тізбекті іздеуді орындау керек.

0-ден 99-ға дейінгі аралықта жүз кездейсоқ құрылған бүтін сандарға арналған алгоритмді орындайтын бағдарламаның коды мынандай:

```
using System;
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
```

```
        {
         int i,j,b,k,n,f,m;
         int[] a = new int[101];
         int[] p = new int[21];
         Random rnd = new Random();
         string buf;
         f = 0; m = 0;
         // іздестіру массивіу құру және экранға шығару
         Console.WriteLine("Izdestiry massivi: ");
         for (i = 0; i < 100; i++)
         {
         a[i] = rnd.Next() % 100;
         if (a[i] < 10) Console.Write(" {0}", a[i]);
         else Console.Write(" {0}", a[i]);
         if ((i + 1) % 20 == 0) Console.WriteLine();
         }
         Console.WriteLine();
         // массивті сұрыптау
         for (i=0;i<99;i++)
         for (j=i+1;j<100;j++)
          if (a[i]>a[j])
          {b=a[i];a[i]=a[j];a[j]=b;}
      Console.WriteLine("Sandardi syriptalgannan kein izdestiry
massivi :");
         for (i = 0; i < 100; i++)
         {
         if (a[i] < 10) Console.Write(" {0}", a[i]);
         else Console.Write(" {0}", a[i]);
         if ((i + 1) % 20 == 0) Console.WriteLine();
         }
         Console.WriteLine();
         Console.WriteLine("Izdey kiltin engiziniz");
         buf = Console.ReadLine();
         k = Convert.ToInt32(buf);
         Console.WriteLine("Blok olshemin engiziniz");
         buf = Console.ReadLine();
         b = Convert.ToInt32(buf);
         // Блоктық іздеу алгоритмі
         for (i=b-1;i<100;i=i+b)
         if(k<=a[i])
         {
         j=i-b+1;n=i;
         while (j<=n || k==a[j])
         {
          if (k==a[j])
          {
          f=1;i=100;
          Console.WriteLine("Nomer:{0}", j);
          }
          j++;
         }
         }
```

```
      if (f==0)
    Console.Write("Izdey kiltine saikes element jok!");
      Console.ReadLine();
     }
     }
    }
```

```
    Бағдарлама жұмысы:
    Izdestiry massivi:
     26 3 28 95 59 6 49 15 46 21 26 61 91 72 37 60 14
34 36 88
     48 65 91 14 48 47 46 52 45 75 18 43 29 15 55 56
35 64 37 56
     56 73 8 57 10 68 22 64 45 78 54 41 6 42 0 8 12
14 47 12
     7 44 70 18 50 30 32 23 64 81 72 50 97 27 79 75
57 81 51 80
     72 82 7 14 21 77 84 87 15 47 44 68 17 25 49 12
50 37 54 80

    Sandardi syriptalgannan kein izdestiry massivi :
     0 3 6 6 7 7 8 8 10 12 12 12 14 14 14 14 15 15 15
17
     18 18 21 21 22 23 25 26 26 27 28 29 30 32 34 35
36 37 37 37
     41 42 43 44 44 45 45 46 46 47 47 47 48 48 49 49
50 50 50 51
     52 54 54 55 56 56 56 57 57 59 60 61 64 64 64 65
68 68 70 72
     72 72 73 75 75 77 78 79 80 80 81 81 82 84 87 88
91 91 95 97

    Izdey kiltin engiziniz
    44
    Blok olshemin engiziniz
    10
    Nomer:43
    Nomer:44
```

Бұл алгоритмнің артықшылығы алдыңғы алгоритмге қарағанда іздестіру уақытының аз жұмсалуы болып табылады. Іздестіру уақыты іздестіру массивіндегі блоктар қосындысына және бір блоктағы элементтер санына пропорционал. Блок өлшемін өзгерте отырып белгілі бір деңгейде іздестіру жылдамдағын реттеуге болады.

Алгоритмнің кемшіліктері: іздестіру уақытының көп жұмсалуы, іздестіру массивін дайындау, яғни түйінді элемент бойынша сұрыптау.

6.3.3 Екілік іздеу алгоритмі. Екілік іздеу алгоритмі іздестіру массивінің элементтерін реттелген тәртіпте орналасуын талап етеді.

Іздестіру массивінің элементтері өсу ретімен орналасқан болсын. Сұратудың ізделетін кілті іздестіру массивінің орта элементімен салыстырылады.

Егер сұратудың ізделетін кілті іздестіру массивінің орта элементінен кіші болса, онда массивтің ортасы мен соңына дейінгі аралықтағы барлық элементтер кейінгі іздестіруде қолданылмайды, ал іздестіру массиві болып басынан ортасына дейінгі аралықта орналасқан барлық элементтер есептеледі.

Егер сұратудың ізделетін кілті іздестіру массивінің орта элементінен үлкен болса, онда массивтің басынан ортасына дейінгі аралықта орналасқан барлық элементтері кейінгі іздестіруде қолданылмайды, ал іздестіру массиві болып ортаншыдан кейін тұрған элементтен соңғы элементке дейін орналасқан барлық элементтер есептеледі.

Сонымен салыстыру операциясынан кейін іздестіру массивінің жартысы кейінгі іздестіруден шығарылды, сондықтан осы алгоритмді жартылай бөлу әдісі деп атайды .

Ары қарай бөлу процессі жаңа іздестіру массивімен қайталанады. Сұратуға сай элемент табылса немесе іздестіру массиві сұратуға сәйкес емес бір элементімен берілсе, онда іздестіру аяқталады.

Алпыс кездейсоқ, өсу тәртібінде құрылған бүтін сандарға арналған екілік іздеу алгоритмін орындайтын бағдарламаның коды мынандай:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
 { int i, n, k, x, f, m, c, p;
  f = 0; m = 0; c = 0;
  int[] a = new int[61];
  Random rnd = new Random();
  string buf;
  // іздестіру массивін құру және экранға шығару
  Console.WriteLine("Izdestiry massivi: ");
  a[0] = rnd.Next() % 2 + 1;
  for (i = 1; i < 60; i++)
  {
  a[i] = a[i - 1] + rnd.Next() % 2 + 1;
  if (a[i-1] < 10) Console.Write(" {0}", a[i-1]);
  else Console.Write(" {0}", a[i-1]);
  if ((i) % 20 == 0) Console.WriteLine();
  }
  if (a[i-1] < 10) Console.Write(" {0}", a[i-1]);
  else Console.Write(" {0}", a[i-1]);
```

```
  Console.WriteLine();
  Console.WriteLine("Izdey kiltin engiziniz");
  buf = Console.ReadLine();
  p = Convert.ToInt32(buf);
  // екілік іздеу алгоритмі
  n = 0; k = 59;
  while (n <= k)
  {
  x = (n + k) / 2;
  if (p == a[x]) { f = 1; m = x; n = k + 10; }
  if (p <= a[x]) k = x - 1; else n = x + 1;
  c++;
  }
  // Іздеудің нәтижесін шығару
 if (f == 0)
Console.Write("Izdey kiltine saikes element jok!");
 else Console.WriteLine("Nomer:{0}", m+1);
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы, бірінші рет іске қосылуы:
Izdestiry massivi:
1 2 4 6 8 10 12 14 15 16 18 20 22 23 24 26 28 29 31 33
34 35 36 38 39 40 42 43 45 46 48 49 50 51 53 54 55 57 58 60
62 64 65 67 68 69 70 72 73 75 77 79 81 83 85 86 87 89 90 92
Izdey kiltin engiziniz
67
Nomer:44

Бағдарлама жұмысы, екінші рет іске қосылуы:
Izdestiry massivi:
1 2 4 5 7 8 10 11 13 15 17 18 20 21 22 23 24 25 27 29
31 33 34 36 38 39 41 43 45 46 48 49 50 51 53 55 57 58 59 60
61 62 64 66 67 68 69 70 72 73 75 77 78 79 81 82 84 86 88 89
Izdey kiltin engiziniz
63
Izdey kiltine saikes element jok!

Басқа алгоритмдерден қарағанда қарастырылған алгоритмнің артықшылығы - іздестіру уақытының көп жұмсалмауында. Іздестіру уақыты іздестіру массиві элементтерінің санының екілік логарифміне пропорционал. Егер іздестіру массиві 1000 элементтен тұрса, онда ең көп дегенде 10 салыстыру операцияларын қолдана отырып іздестірудің нәтижесін алуға болады. Егер іздестіру массиві 65000 элементтен тұрса, онда барлығы 16 салыстыру операциялары керек болады. Осы алгоритмнің

артықшылығы іздестіру массивінің элементтерінің саны үлкен болғанда ерекше байқалады.

Алгоритмнің кемшілігі – массивті реттеу талабы. Егер іздестіру массивінде бірдей элементтер болса, онда алгоритмді «жетілдіру» керек. Мысалы, тауарлардың атаулары немесе қызметкерлердің тегі.

6.3.4 Кілтті адреске түрлендіретін іздеу алгоритмдері хештеу (хеширование) деп аталады. Іздестіру операцияларын орындау тұрғысынан деректерді ұйымдстырудың ең жақсы әдісі деректерді массив түрінде ұйымдастыру болып табылады, ал іздеу кілті ретінде индекс мәні қолданылады. Осы жағдайда керекті элементті табу үшін деректер массивіне бір рет жүгіну жеткілікті.

Әрине осындай жүйелер «таза» түрінде сирек кездеседі. Әдетте олар шағын жүйелер, мысалы, студент нөмірі немесе оқытушы коды бойынша керекті жазбаны табуға болады.

Деректер массивінің индексін табу үшін кілттерге арнайы хеш-функция (HF) арқылы кейбір түрлендірулерді орындау керек.

Хеш-функция ізделетін элементтің кілтін 0 мен n-1 аралығындағы сандық мәнге (индекске) түрлендіреді. Әлбетте элементтер мәнін сақтау үшін N ұяшығы (хеш-кесте) бар массив керек.

Егер мүмкін болатын кілттер n-ге тең немесе одан кіші болса, онда жүйе адреске баламалы кілтті жүйеге түрленеді.

Егер мүмкін болатын кілттер n-нен үлкен болса, онда хеш-функцияны қолдану керек. Ол 0 мен n-1 аралығында кілттерді біркелкі «таратуды» қамтамасыз етеді.

Әлбетте түрлі кілттерге бірдей индекстер тағайындалуы мүмкін, өйткені хеш-функция «көпшілігі біреуге» қағидасымен жұмыс істейді. Осындай жағдайды қақтығыс деп атайды.

Берілген мәнде ықтимал қақтығыстарды шектеуге мүмкіндік беретін хеш-функцияларды ұйымдастырудың түрлі әдістері бар.

Хеш-функцияны дайындау кезінде әдетте бөлу әдісін қолданады.

Ол бастапқы кезеңде кілтті бүтін санға түрлендіреді, ал одан кейін осы сан 0 мен n-1 аралығына қосылады.

Жолдық кілттер үшін сандық мәнді алуға арналған белгілі алгоритмнің бірі былай орындалады: әрбір жолдың байтына алдынғы мән мен белгілі бір тиянақталған көбейткіштің (хэш) көбейтіндісі қосылады.

Тәжірибе түрінде анықталған, 31 мен 37 мәндері хеш-функциясының ASCII жолдары үшін тиімді жиын болып келеді. Атап өту керек, бір деректерге арналған жақсы жұмыс істеп тұрған «хеш-функция» басқа түрге арналғандарға «жаман» жұмыс істеуі мүмкін. Мысалы, бір топ адамдардың (мысалы, студенттердің) туған жылдарына байланысты кілттер бойынша хеш-функциясын пайдалану өте қиын. Коллизияны алудың нұсқасының бірі әр хеш-кесте торына арналған тізімдік құрылымын пайдалану. Хеш-кесте мен жай тізімдерді пайдалануды көптеген авторлар информатиканың ең негізгі жаңалығы деп санайды.

Мысалы, «автомобильдер» анықтама жүйесін құру барысында автомобильдер нөмірінің сандық бөлігі бойынша хеш-функциясын пайдалануға болады.

Сандық бөлігі бойынша сәйкес келетін автомобилдерді (олар айырмашылығы әріптік бөлігінде) жай тізімге ұйымдастыру.

Осындай жүйеде іздеу екі кезеңде орындалады: біріншіде – нөмірдің сандық бөлігі бойынша, ал екіншіде нөмірдің әріптік бөлігінің тізімдегі реттілік іздеуінде.

6.4 Өзін-өзі тексеру сұрақтары

1 Бағдарламаның келесі үзіндісінде сұрыптаудың қандай алгоритмі қолданылған:

```
for (I = 0; I < 10; I++)
for (j = I + 1; j <= 10;j++)
if (a[I] > a[j])
{b = a[I]; a[I] = a[j]; a[j] = b;} ?
```

2 Бағдарламаның келесі үзіндісінде сұрыптаудың қандай алгоритмі қолданылған:

```
 for (i=0; i<=9; i++)
for (j=0; j<=9-i; j++)
if (a[j]<a[j+1])
{ b=a[j];a[j]=a[j+1];a[j+1]=b;} ?
```

3 «Таңдау» әдісін қолданатын сұрыптау алгоритмінің есептеу тиімділігі қандай болады?

4 Ең үлкен орташа іздеу уақыты қандай іздеу алгоритмінде жұмсалады?

5 Массивте элементтерді блоктық іздеу алгоритмі туралы мағлұмат беру.

6 Блоктық іздеу алгоритмінде орташа іздеу уақытын қалай азайтуға болады?

7 Кілт бойынша өсу ретімен сұрыпталған массив жазбаларын екілік іздеу алгоритмі туралы мағлұмат беру.

8 Іздестіру массивінің «кілті» дегеніміз не?

9 Жазба өрісі. Іздестіру массиві жазба өрісі арқылы реттелген.

10 Хеш-функцияның маңызы қандай?

11 Араластыру кезінде қақтығысты болдырмаудың қандай нұсқасы жиі қолданылады?

7 КЛАСС ӘДІСТЕРІ

7.1 Класс әдістері мен деректер туралы мағлұмат

Оқу құралының бірінші бөлімінде C# тілі бағдарламалаудың объекті-бағытталған тілі екенін және барлық әрекеттерді (бағдарламалардың кодтары) `Program` класының `Main` әдісіне жазу керектігін біз ескертіп өткенбіз.

Класс дегеніміз – бір құрылымда деректер мен деректерді өңдеу әдістерін біріктіретін жаңа тип түрі.

Класс деректері ретінде кластың тұрақтылары мен айнымалылары бола алады. Класта деректерді жариялаған кезде әдетте оған қол жеткізу спецификаторлары көрсетіледі, мысалы, `private int a;`.

Класс деректерін жариялаудың жалпы пішімі:

[спецификаторлар] [const] типі атауы

[= бастапқы_мәні].

Деректерге өзгермейтін мәндерді сақтау үшін арналған класс тұрақтылары мен класс өрістері жатады (класс айнымалыларының типтері мен атаулары).

Егер деректер алдында `public` спецификаторы қолданылса, онда олар «бағдарламаға» ашық болады, біз әрқашан `public` спецификаторын қолданамыз.

Объекті-бағытталған бағдарламалу технологиясында класс деректері әдетте «бағдарлама үшін жабылады» - `private` спецификаторы қолданылады және қалыпты жағдайда деректер мен әдістер үшін `private` спецификаторы қолданылады.

Класс әдістері дегеніміз – осы класпен жұмыс жасауға арналған, атауларға ие код үзіндісі.

Әдістер класта қолдануға болатын әрекеттер жиынтығын анықтайды (олар кластың тәртібін анықтайды).

Әдіс бір рет сипатталады, ал әртүрлі деректер үшін қанша керекті болса, сонша рет шақыртуға болады.

Класс әдістерінің жалпы жазылу пішімі мынандай түрде болады:

```
[ спецификаторлар ] әдіс типі әдіс атауы ( [ параметрлер ] )
      {әдіс денесі}
```

Мысалы:

```
 static void Main(string[] args)
                  { }
```

Ең жиі кездесетін спецификаторлар: `private`, `public` және `static`.

`private` спецификаторымен жарияланған кластың кез келген әдістері осы кластың әдістерінде ғана қолжетімді болады.

public спецификаторы арқылы әдіс бағдарламаның кез келген жерінде қолжетімді болады.

static спецификаторы әдісті класс объектісін дайындамай-ақ, оны «класс деңгейінде» қолдануға болатындығын сипаттайды. Бұл өте маңызды, өйткені осы пәнде біз статикалық әдістерді жиі қолданамыз.

Әдіс типі кез келген бағдарламада анықталған типте немесе C# тілінің стандарты типінде немесе void - типсіз түрде беріле алады. Мысалы:

```
int kol(int a) { … }
public double sym(out float r) { … }
public void poisk(ref float s) { … }
public int funkcij(int a,out int b,params int[] c){ … }
```

Егер әдіс типі берілсе (void-тан бөлек), онда әдіс денесінде соңғы оператор ретінде әдіс жұмысының нәтижесін қайтаратын return операторы болуы керек. Сонымен қатар әдісті айнымалыға меншіктеу керек немесе C# тілінің операторларында өрнек сияқты қолдану керек. Әдетте осындай әдістерді функция деп атайды.

Егер әдістің алдында void типі жазылса, онда әдіс өзінің жұмысын return операторы арқылы қайтармауы керек (осы жағдайда return операторы әдістің денесінде болмайды). Әдетте осы әдіс процедура деп аталады, оны айнымалыға меншіктеу керек немесе жеке ішкі бағдарлама - процедура түрінде жазуға болады (дөңгелек жақшаларында параметрлері бар әдіс атауы).

Әдіс атауы – бағдарламашы анықтайтын идентификатор. Әдіс атауының мағынасы болғаны дұрыс, мысалы, sym, max, poisk және т.б.

Әдіс параметрлері (формалды параметрлер) бағдарлама мен әдіс арасында деректермен алмасу үшін арналған. Әдіс параметрлері әдетте керекті алгоритмді орындайтын «күйге келтіру» құралы деп аталады.

C# тілінде әдістердің келесі параметрлері бар:

– мәндерді анықтайтын параметрлер (мәндік параметрлер, яғни әдіс қабылдайтын кіріс параметрлер);

– шығыстық параметрлер (out қызметтік сөзімен белгіленеді);

– сілтемелік параметрлер (ref қызметтік сөзімен белгіленеді);

– массивті параметр (params қызметтік сөзімен белгіленеді).

Мәндерді анықтайтын параметрлерде қызметтік сөз қолданылмайды.

Класс әдістерінің параметрлері үтірлер арқылы бөлінеді. Әдісте массив параметрі біреу және параметрлер тізімінде соңғы болуы керек.

Егер әдісте мәндерді анықтайтын параметрлер жарияланса, онда бұл әдістің кейбір айнымалылардың көшірмелерін өз құзырына алғандығын көрсетеді. Әдіс осы көшірмелердің мәнін өзгерте алады, бірақ олардың түпнұсқасы (бағдарламада) өзгермеген қалыпта қалады. Әдістің жұмысы аяқталғаннан кейін мәндерді анықтайтын параметрлер компьютер жадысынан жойылады.

Әдістің шығыстық параметрлері бағдарламаға нәтижелерді жеткізу үшін арналған. Әдістің денесіндегі шығыстық параметрлерге кейбір мәндер меншіктелуі тиіс, әйтпесе бағдарлама компиляциясы кезінде қате кеткен туралы хабар шығады.

Егер әдісте сілтемелік параметрлер жарияланған болса, онда әдіс сәйкес айнымалылардың адресін өз құзырына алады және оларды өз алгоритмі бойынша қолдана алады (жаңа мәндерді жаза және оқи алады).

Әдістегі жарияланған массивті параметрі нақты айнымалылардың кез келген санымен жұмыс жасауға арналған. Сонымен қатар `params` қызметтік сөзінен кейін тұрған формалды параметр кез келген өлшемді деректер массивімен сәйкестікке келтіріледі.

Сонымен, әдіске өзінің параметрлері арқылы керекті мәліметтерді (мәндерді анықтайтын параметрлер және сілтемелік параметрлер) алуына немесе өз жұмысының нәтижелерін қайтаруына болады (шығыстық параметрлер және сілтемелік параметрлер).

Әдіс денесінде кейбір алгоритмді орындайтын бағдарлама кодының үзіндісі бар. Бұл ретте әдіс формалды параметрлермен бірге әрекеттер үлгісі ретінде қолданылады. Бағдарламада формалды параметрлердің орнына нақты айнымалылар қолданылуы керек – нақты параметрлер мен әдістің әрекеттер үлгісі нақты айнымалылар үшін қолданылады.

Бағдарламада әдістің формалды параметрлермен бірге жұмыс жасауы мүмкін емес.

7.2 Бағдарламада класс әдістерін қолдану

Әртүрлі есептерді шешу мысалдары арқылы бағдарламаларда әдістерді қолдануды қарастырайық.

7.1-есеп. Диалог режимінде үш бүтін сандар енгізіледі. Ең үлкен санды тауып, оны шығару керек. Бүтін типтегі функция (әдіс) қолданылады.

Есепті шешу алгоритмі келесі амалдардан тұрады:

– Диалог режимінде үш бүтін сандарды енгізу керек, мысалы, a,b,c;.

– бірінші айнымалыда ең үлкен сан жазылғанын жариялаймыз (m айнымалысына a мәні меншіктеледі);

– m айнымалысының мәнін b айнымалысының мәнімен салыстырамыз, ең үлкен мәнді m айнымалысына меншіктейміз;

– m айнымалысының мәнін c айнымалысының мәнімен салыстырамыз, ең үлкен мәнді m айнымалысына меншіктейміз;

– монитор экранына m айнымалысының мәнін шығарамыз.

Салыстыру процесін келесі бүтін типті функция түрінде жазайық:

```
static int maxc(int aa, int bb, int cc)
 {
  int mm;
  mm = aa;
```

```
  if (mm < bb) mm = bb;
  if (mm < cc) mm = cc;
  return mm;
 }
```

Бағдарламаның толық коды:

```
using System;
namespace ConsoleApplication1
{
class Program
{
static int maxc(int aa, int bb, int cc)
         {
  int mm;
  mm = aa;
  if (mm < bb) mm = bb;
  if (mm < cc) mm = cc;
  return mm;
 }
 static void Main()
 {
  int a,b,c,m;
  string buf;
  Console.Write("a bytin canin engiziniz ");
  buf = Console.ReadLine();
  a = Convert.ToInt32(buf);
  Console.Write("b bytin canin engiziniz ");
  buf = Console.ReadLine();
  b = Convert.ToInt32(buf);
  Console.Write("c bytin canin engiziniz ");
  buf = Console.ReadLine();
  c = Convert.ToInt32(buf);
  m = maxc(a,b,c);
Console.WriteLine("Algashki sandar: {0}, {1}, {2}", a, b,
c);
Console.WriteLine("Maksimal san = {0}", m);
Console.WriteLine("Enter pernesin basiniz");
Console.ReadLine();
}
}
}
```

Бағдарлама жұмысы:

```
a bytin canin engiziniz 21
b bytin canin engiziniz 34
c bytin canin engiziniz 17
Algashki sandar: 21, 34, 17
```

```
Maksimal san = 34
Enter pernesin basiniz
```

maxc() функциясының денесінде бүтін типті mm айнымалысын жарияладық.

Бағдарламалауда жергілікті және ауқымды айнымалылар ұғымы және олармен жұмыс жасау ережелері бар.

Кез келген функция ішінде жарияланған айнымалылар жергілікті айнымалылар деп аталады.

Класс ішінде, бірақ оның кез келген функциясының сыртында жарияланған айнымалылар осы кластың ауқымды айнымалылары деп аталады.

C# тілінде атаулары бірдей жергілікті және ауқымды айнымалыларды қолдануға рұқсат етілмейді, бағдарламаның барлық айнымалыларында жеке аттары болуы керек.

Әдіс терминологиясында «көз көрерлік аймағы» ұғымы болады. Кластың кез келген әдісі өз класының барлық ауқымды айнымалыларын «көреді» және оларды қолдана алады.

Жергілікті айнымалының «өмір» уақытын әдіс жұмысының уақыты анықтайды.

7.2–есеп. Диалог режимінде үш бүтін сандар енгізіледі. Оларды кему тәртібінде шығару керек. Класс әдісін қолдану керек.

Есеп түсінікті, бірақ алгоритм анық емес. Үш айнымалының орнына екі айнымалы болса, онда есепті шешу оңай болар еді.

A және B айнымалылары бар деп есептейік. Олардың мәнін X және Y айнымалыларына жазу керек, ең үлкен мәнді X айнымалысына, ал ең кіші мәнді Y жазамыз.

Ол үшін A және B айнымалыларының мәнін салыстыру керек, егер A B-дан үлкен болса, онда X айнымалысына A мәнін меншіктейміз, ал B мәнін Y айнымалысына меншіктейміз. Әйтпесе B мәнін X айнымалысына, ал A мәнін Y айнымалысына меншіктейміз.

Екі айнымалы үшін құрылған алгоритмді біз үш айнымалы үшін де былай қолдана аламыз:

Алдымен A және B салыстырамыз, нәтижелерін X және Y жазамыз.

X және C салыстырамыз, нәтижелерін X және Z жазамыз. Ең үлкен мәнді X айнымалысына жазылады.

Z және Y салыстырамыз, нәтижелерін Y және Z жазамыз. Кему тәртібінде табылған сандарды Y және Z айнымалыларына жазылады.

Сонымен, бір алгоритмді – екі санды салыстыруды және ең үлкен және ең кіші санды табуды біз үш рет орындадық. Нәтижесінде X, Y және Z айнымалылар тізбегін аламыз, онда сандар кему тәртібінде орналасады.

Екі айнымалыға арналған алгоритмді орындау үшін `static void maxmin(int aa, int bb, out int xx, out int yy)` әдісін қолданайық.

Өзіміз үшін `maxmin` әдісінің атауында мынаны ескереміз: қайтарылатын параметрлер ең үлкен, ең кіші тәртібінде орналасады

Бағдарламаның коды:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static void maxmin(int aa, int bb, out int xx, out
int yy)
 {
  if (aa>bb) {xx = aa; yy = bb;}
   else { xx = bb; yy = aa;}
 }
 static void Main(string[] args)
 {
  int a, b, c, x, y, z;
  string buf;
  Console.Write("a bytin canin engiziniz ");
  buf = Console.ReadLine();
  a = Convert.ToInt32(buf);
  Console.Write("b bytin canin engiziniz ");
  buf = Console.ReadLine();
  b = Convert.ToInt32(buf);
  Console.Write("c bytin canin engiziniz ");
  buf = Console.ReadLine();
  c = Convert.ToInt32(buf);
  Console.WriteLine("Algashki rettilik: {0}, {1},
{2}", a, b, c);
  maxmin(a,b,out x,out y);
  maxmin(c,x,out x,out z);
  maxmin(y,z,out y,out z);
  Console.WriteLine("Alingan rettilik: {0}, {1},
{2}", x, y, z);
  Console.WriteLine("Enter pernesin basiniz");
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
a bytin canin engiziniz 5
b bytin canin engiziniz -3
c bytin canin engiziniz 8
Algashki rettilik: 5, -3, 8
Alingan rettilik: 8, 5, -3
Enter pernesin basiniz
```

Келтірілген мысалда static void maxmin(int aa, int bb, out int xx, out int yy) әдісі бағдарламадан екі (int aa, int bb) параметрді алады және оған екі (out int xx, out int yy) параметрді қайтарады.

Сонымен бірге, бағдарламада қолданылатын нақты параметрлерде `xx, yy` формалды параметрлердің орнында мәндер болуы тиіс.

Параметрлерді қайтаруды `ref сілтемесімен ұйымдастыруға болады, бірақ бұл үшін` қайтарылатын нақты параметрлер сілтемелік типте болуы керек.

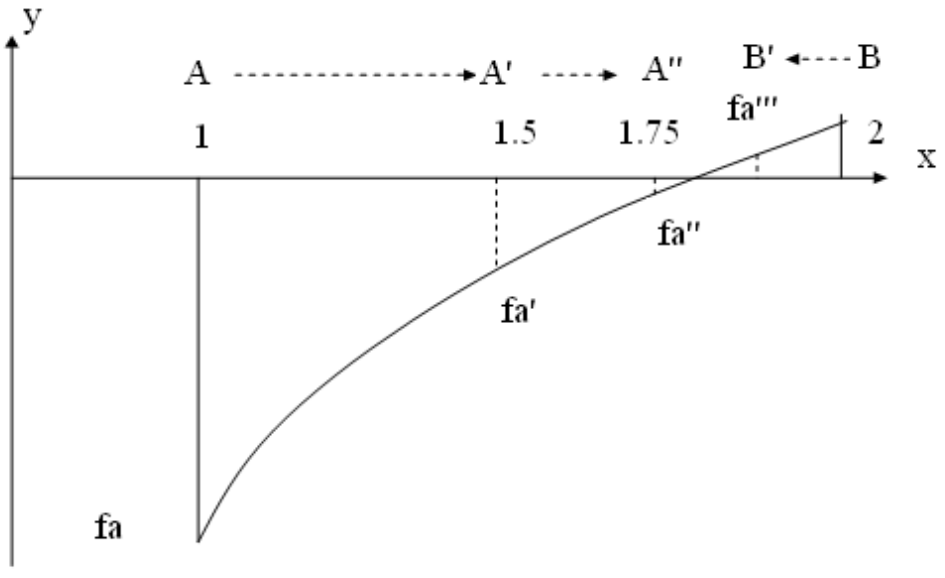
Математикалық есептерде қолданылатын функцияны пайдалану мысалы ретінде теңдеу түбірлерін табатын ортасынан бөлу әдісінің бағдарламалық жүзеге асыруын қарастырайық.

7.3-есебі. Ортасынан бөлу әдісін пайдаланып, 1 мен 2 аралығындағы кесіндіде t = 0.0001 дәлдігімен теңдеу түбірлерін табу:

$$\cos(\frac{2}{x}) - 2\cdot\sin(\frac{1}{x}) + \frac{1}{x} = 0 \qquad (8)$$

Ортасынан бөлу әдісі берілген кесіндінің ұштарында функцияның мәні әртүрлі таңбада болады деп көздейді, мысалы , «-» и «+».

Демек, осы кесіндіде функция нөлге тең болатын кем дегенде бір х бар.



7.1-суреті – Ортасынан бөлу әдісін қолдану

Х мәнін табу алгоритмі берілген кесіндіні тең бірдей бөліктерге бөлуге және функцияның мәні әртүрлі таңбада болатын бөлікті таңдауға негізделген. Осы процесс функцияның нөлге тең болатын шешім табылғанша немесе кесінді ұзындығы берілген дәлдіктен кем болғанша қайталанады (осы жағдайда теңдеудің түбірі табылды деп есептеледі).

Алгоритмде аргументтің түрлі мәндерінде функция шешімін бірнеше рет есептеу орындалады. Сондықтан осы процессті жеке әдісте бөліп жазу керек, мысалы f атауы бар әдіс.

Бағдарлама коды:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static double f(double a)
  { return (Math.Cos(2/a)-2*Math.Sin(1/a)+1/a); }
 static void Main()
 {
  double a, b, x, ya, yb, t;
  a = 1;
  b = 2;
  t = 0.0001;
  ya=f(a);
  x=(a+b)/2;
  yb=f(x);
  while (b-a>t && yb!=0)
  {
  if (ya*yb<0)
   {b=x; x=(a+b)/2; yb=f(x);}
       else
   { a=x; ya=yb; x=(a+b)/2; yb=f(x);}
  }
  Console.WriteLine("Tendeydin tybiri = {0}", x);
  Console.WriteLine("Enter pernesin basiniz");
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
Tendeydin tybiri = 1,87564086914063
Enter pernesin basiniz
```

f() функциясын қолдану бағдарлама көрнекілігін өсіреді және оның жазылуын жеңілдетеді.

7.3 Әдістерді қайта анықтау

C# тілінде бірнеше әдісті бір атаумен анықтауға болады, сонымен қатар оларда формалды параметрлердің әр түрлі жиыны болуы мүмкін (немесе параметрлердің түрлі типтері). C# тілінде осы тәсілді әдістерді қайта анықтау деп аталайды. Осындай әдісті шақырған кезде бағдарлама сәйкес әдісті аргументтердің санын, типін, тәртібін талдау арқылы анықтайды.

Әдістерді қайта анықтау мысалын қарастырайық:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
 class Program
 {
 public static int funk(int x) { return x*x;}
 public static double funk(double x) { return x+x; }
 public static char funk(char x) { return x; }
 static void Main(string[] args)
 {
 Console.WriteLine("x = 5 funk(5) = {0} ", funk(5));
 Console.WriteLine("x = 5.5 funk(5.5) = {0} ",
funk(5.5));
 Console.WriteLine("x = 'g' funk('g') = {0} ",
funk('g'));
 Console.WriteLine("Enter pernesin basiniz");
 Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
x = 5 funk(5) = 25
x = 5.5 funk(5.5) = 11
x = 'g' funk('g') = g
Enter pernesin basiniz
```

Келтірілген мысалда аргументтерінің типтері әртүрлі, қайта жүктелген `funk` функциясын қолданады. Бірақ осындай функциялар аргументтер санымен де өзгеше болуы мүмкін.

Әдетте функцияларды қайта анықтау түрлі типтегі деректерді «түсінетін» әдістерді жазу үшін қолданылады. Мысалы, ағымдағы күнді түрлі жолдармен енгізуге болады. Мысалы:

Бүтін сандармен (ай, күн, жыл – 23 12 07);

мәтінмен (23 желтоқсан 2007 жыл);

қосымша символдарды қолданған жиын (20.10.07 немесе 17/09/2007).

Бағдарлама осы деректерді дұрыс «түсінуі» үшін әрбір тип бойынша әдістерді қайта жүктеуді қолдану қажет.

7.4 Рекурсия

Рекурсия дегеніміз – әдісте операторларды орындау барысында осы әдістің өзіне қайта жүгінуді орындайтын есептеу процесін ұйымдастыру тәсілі.

Рекурсиялық есептеулерді сипаттайтын классикалық мысалы - факториалды есептеу.

N! - есептеу керек болсын

$$N ! = 1 \cdot 2 \cdot \ldots \cdot (N-1) \cdot N = (N - I) ! \cdot N . \quad (9)$$

Сонымен, N! есептеу үшін (N-1)! білу керек.

Өз кезегінде (N - 1) ! = 1·2· ...· (N - 2) !· (N - 1) , және т.б. 2 ! = 1 ! · 2 , 1 ! = 1.

N факториалын қарапайым FOR циклі арқылы есептеуге болады, мысалы:

```
pr = 1;
for (i=1; i<=N;i++) pr = pr*i ;
```

Цикл жұмысының нәтижесі pr айнымалысында жазылады.

Рекурсияны қолдану бір немесе бірнеше шартты циклдік операциялары бар күрделі есептерді оңай шешуге көмектеседі.

Сонымен қатар, шартты цикл денесі рекурсивті функцияға жазылады, ал функцияны қолдану бағдалама кодын қысқартады және оны түсінуді жеңілдетеді.

Рекурсивті функцияны қолданатын бағдарламаның мысалы ретінде диалог режімінде енгізілген бүтін санның факториалын есептеу бағдарламасын дайындаймыз.

Бағдарлама коды:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static int fakt(int n)
 {
  int fu, k;
  fu = n;
  if (n != 1) { k = n - 1; fu = fu * fakt(k); }
  return fu;
 }
 static void Main()
 {
```

```
      int a, n;
      string buf;
Console.Write("n bytin canin engiziniz ");
      buf = Console.ReadLine();
      n = Convert.ToInt32(buf);
      a = fakt(n);
      Console.WriteLine("n! = {0}", a);
      Console.WriteLine("Enter pernesin basiniz");
      Console.ReadLine();
      }
      }
     }
```

Бағдарлама жұмысы:

```
n bytin canin engiziniz 5
n! = 120
Enter pernesin basiniz
```

Көрсетілген бағдарлама кодында циклдар жоқ, бағдарлама алгоритмі жеңіл қабылданады: `n` айнымалының мәнін енгіземіз, факториалды есептейміз, монитор экранына нәтижені шығарамыз

Бағдарламаның орындалу үдерісінде `fakt()` рекурсивті функцияның жұмысын қарастырайық. Бағдарламаның жұмысы барысында 5 санын енгіздік деп есептейік.

5 саны үшін факториалды есептеу `fakt(5)` функциясы есептеледі. Бірақ `fakt(5)` функциясын есептеу үшін `fakt(4)` функциясын есептеу керек. Сондықтан `fakt(5)` функциясы уақытша тоқтатылады, оның параметрлері бағдарламалық стекке орналастырылады және 4 саны үшін факториалды есептеу `fakt(4)` функциясы іске қосылады.

Бірақ `fakt(4)` функциясын есептеу үшін `fakt(3)` функциясын есептеу керек. Сондықтан `fakt(4)` функциясы уақытша тоқтатылады, оның параметрлері бағдарламалық стекке орналастырылады және 3 саны үшін факториалды есептеу `fakt(3)` функциясы іске қосылады. Одан кейін стекке `fakt(3)`, `fakt(2)` функцияларының параметрлері орналастырылады, тек содан кейін ғана `fakt(1)` функциясы есептеледі.

`fakt(1)` функциясының мәні болса, онда бағдарлама `fakt(2)` функциясының мәнін есептей алады. Ол үшін `fakt(2)` параметрлерін стектен шақыру керек және `fakt(1)` функциясының нәтижесін пайдаланып есептеу жұмысын жалғастыру керек. `fakt(2)` функциясының нәтижесін `fakt(3)` функциясын есептеу үшін қолдануға болады. Ол үшін `fakt(3)` параметрлерін стектен шақыру керек және `fakt(2)` функциясының нәтижесін пайдаланып есептеу жұмысын жалғастыру керек. Осыдан кейін `fakt(4)` және fakt(5) функциялары жоғарыда сипатталған үлгіде есептеледі.

`fakt(5)` есептелгеннен кейін оның нәтижесі a айнымалысына меншіктеледі және монитор экранына шығарылады.

Рекурсивті есептеулерді қолдану бағдарлама көрнектілігін арттырады, бірақ бағдарлама жұмысына әдеттегі циклді оператор көмегімен ұйымдастырылған бағдарламаларға қарағанда уақыт көбірек жұмсалады.

7.4-есеп. Келесі өрнекті есепте.

$$y = \cfrac{1}{1+\cfrac{1}{3+\cfrac{1}{5+\cfrac{1}{\cdots}}}} \qquad (10)$$

$$(N-2)+\frac{1}{N}$$

Мұнда N 5555 тең, бүтін тақ сан.

7.4-есептің шешімін екі жолмен қарастырайық. `while` шартты циклінің көмегімен және рекурсия арқылы.

Шартты цикл көмегімен есепті шешу алгоритмін $\frac{1}{N}$ шамасынан бастап $\frac{1}{(1+\ldots)}$ шамасына дейін есептеу керек, әрбір циклда N 2-ге азайтылып отырады. Есептеу N>1 шарты орындалғанға дейін жүргізіледі.

Бағдарлама коды:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
 {
  double s, k, n;
  n=5555;
  k=1;
  s = 1 / n;
  while (n > 1)
  {
  n = n - 2;
  s = 1 / (n + s);
  }
 Console.WriteLine("Natijesi = {0} ", s);
 Console.WriteLine("Enter pernesin basiniz");
  Console.ReadLine();
 }
```

```
 }
}
```

Бағдарлама жұмысының нәтижесі:

```
Natijesi = 0,761594155955765
Enter pernesin basiniz
```

Рекурсия алгоритмі өрнектің формуласымен анықталады, $\frac{1}{(1+\ldots)}$ есептеу үшін $\frac{1}{(3+\ldots)}$ есептеп алу керек, ары қарай $\frac{1}{N}$ дейін қайталанады.

Бағдарлама коды – 2:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
 class Program
 {
 static double rec(double k, double n)
 {
  double rez;
  if (k< n) { k = k + 2; rez = (k - 2) + 1 / rec(k, n); }
  else rez = n;
  return rez;
 }
  static void Main()
 {
  double s, k, n;
  n=5555;
  k=1;
  s = 1 / rec(k, n);
 Console.WriteLine("Natijesi = {0} ", s);
 Console.WriteLine("Enter pernesin basiniz ");
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысының нәтижесі:

```
Natijesi = 0,761594155955765
Enter pernesin basiniz
```

Бағдарлама жұмысының нәтижесі бірдей.

Рекурсия қолданылған бағдарлама кодын түсіну жеңіл болып келеді.

Егер `n` айнымалысының мәнін ұлғайтсақ, мысалы `n = 55555555`, онда рекурсиясы бар бағдарламада стекті өзгертуді қажет етеді (рекурсия

саны бағдарлама стегінің көлеміне байланысты), `while` циклді бағдарлама барлық ауқымдағы сандармен жұмыс істейді.

Бағдарламалық стекті пайдаланғандықтан бағдарламаның рекурсиямен жұмыс істеу уақыты циклдармен жұмыс істеу уақытынан көбірек (`n` шамасының мәні үлкен болғанда).

7.5 Өзін-өзі тексеру сұрақтары

1 C# тілінде әдістің типі анықталған болса, әдіс қалай аяқталуы керек?

2 C# тілінде әдістің қандай формалды параметрлері сілтеме-параметрлер деп аталады?

3 C# тілінде әдістің алдында `out` қызметтік сөзі жазылатын формалды параметрлері қалай аталады?

4 Егер функцияның қайтарылатын мәні `double` типінде болса, бағдарламада функцияны қалай қолдану керек?

5 Функция ішінде басқа функцияны жариялауға бола ма?

6 Қандай функцияны рекурсивті функция деп атайды?

7 Атаулары бірдей бірнеше әдісті анықтау процесі қалай аталады?

8 Кейбір әдістердің алдында `static` модификаторы неге қолданылады?

9 Келесі бағдарлама үзіндісі экранға нені шығарады:

```
public static void ttt(int a, int b, out int x, out int y)
{ if (a>b) {x = a;y = b;}else {x = b; y = a;} }
static void Main(string[] args)
 {
 int a, b, c = 0, d = 0;
 a = 5;
 b = 8;
 ttt(a, b, out c,out d);
 Console.WriteLine("c = {0} d = {1} ", c,d);
 }?
```

10 Келесі бағдарлама үзіндісі экранға нені шығарады:

```
public static void ttt(int a, int b, int x, int y)
 { x = a; a = b; y = a; }
static void Main(string[] args)
 {
  int a, b, c = 0, d = 0;
  a = 5;
  b = 8;
  ttt(a, b, c, d);
  Console.WriteLine(" {0} {1} ", c,d);
 }?
```

8 ЖОЛДЫҚ АЙНЫМАЛЫЛАР

8.1 Символдық тип

Бағдарламалауда символдарды, сөздерді, сөйлемдерді, мәтіндерді өңдеумен байланысты көптеген есептер класы бар.

Осы мақсатта көптеген бағдарламалау тілдеріне символдармен және мәтіндермен, яғни `char` және `string` типтерімен жұмыс істейтін арнайы құралдар қосылған.

C# тілінде осы типтердің екеуі де бар. Олардың арасында тығыз байланыстың барын айта кету керек.

Кейбір авторлар `char` типіндегі айнымалы бір символдан тұратын жол деп есептейді. Басқа авторлар жолдық типтегі айнымалыларды `char` типіндегі айнымалылар массиві ретінде қарастырады.

C# тілінде символдармен жұмыс жасауға арналған арнайы `char` класы қолданылады, онда символдарды көрсетудің екі байттық жүйесі және осы символдармен жұмыс жасаудың көптеген әдістер жиыны қолданылады.

Символдарды көрсетудің екі байттық жүйесі Unicode кодтамасы деген атқа ие болды. Windows тобындағы ОЖ-де 2 байтты UTF-16LE кодтамасы қабылданған.

Барлық символдар топтастырылған және реттелген. Мысалы, 0 мен 9 арасындағы сандардың екілік кодтары өсу тәртібінде, қатарынан орналасқан. Алфавиттің a және z арасындағы үлкен және кіші әріптері де өсу тәртібінде, қатарынан орналасқан. Кириллицада қолданылатын "Ё" әрпі ғана өзгеше сипатталады.

Символдарды кодтаудың осы ерекшелігі мәтіндерді өңдеудің түрлі алгоритмдерін оңай құруға мүмкіндік береді, мысалы, алфавит бойынша сөздерді реттеу.

`Unicode` кодтамасымен берілетін алфавитте 65000 аса символдар бар болғандықтан, кодтардың көбі резервте сақталады және қолданылмайды.

`Char` класында апострофқа алынған символдармен берілген символды тұрақтылар және басқарушы символдар мен `Unicode`-тізбегі анықталған.

Бағдарламада символдарды қолдану үшін олар апострофқа алынады. `Escape`-тізбегінің коды ақпаратты экранға, принтерге, басқа да құрылғыларға шығару үшін пішімдеуді қолданады. Ол бір немесе бірнеше символдан тұрады, оның алдында кері слеш (\) жазылады.

Символды көрсету `Unicode`–тізбегінде кодтың көмегімен орындалады.

8.1-кестеде кейбір escape-тізбегі көрсетілген.

8.1-кесте – C# тіліндегі кейбір `escape`-тізбегінің мысалдары

\n	Жаңа жол символы
\t	Көлденең табуляция
\v	Вертикаль табуляция
\b	Бір позицияға шегіну символы
\r	Күймешені қайтару
\f	Бетті ауыстыру
\a	Дыбыстық сигнал (ескерту)
\\	Кері слеш символы
\'	Бір тырнақша символы
\"	Қос тырнақша символы
\?	Сұрақ белгісі символы
\0	Нөлдік символ (барлық биттер нөлге тең)

Ескерту, мәтін ішінде `escape`-тізбегін қарапайым символдар сияқты қолдануға болады.

`Char` класында символдармен жұмыс жасау үшін әдістердің үлкен жиыны бар. Көптеген әдістер статикалық болып келеді және объекттерді құруды қажет етпейді. Кейбір әдістер 8.2-кестеде көрсетілген.

8.2-кесте – кейбір char класының статикалық әдістері

Әдістер	Сипаттамасы
Double GetNumericValue(char c);	Символдың сандық кодын қайтарады. Егер символ болмаса, онда -1 қайтарылады.
UnicodeCategory GetUnicodeCategory (char c);	Барлық символдар категорияға бөлінген, сондықтан әдіс символ категориясын қайтарады.
bool IsControl(char c);	Егер c символы басқарушы болса, онда ол true мәнін қайтарады.
bool IsDigit(char c);	Егер c символы цифр болса, онда ол true мәнін қайтарады.
bool IsLetter(char c);	Егер c символы әріп болса, онда ол true мәнін қайтарады.
bool IsLetterOrDigit(char c);	Егер c символы цифр немесе әріп болса, онда ол true мәнін қайтарады.
bool IsLower(char c);	Егер c символы кіші әріп болса, онда ол true мәнін қайтарады.
bool IsNumber(char c);	Егер c символы сан болса, онда ол true мәнін қайтарады.
bool IsPunctuation(char c);	Егер c символы тыныс белгілерінің символы болса, онда ол true мәнін қайтарады.
bool IsSeparator(char c);	Егер c символы бөлгіштерге жатса, онда

	true мәні қайтарылады.
bool IsSurrogate(char c);	Егер c символы құрастырылған болса, онда true мәні қайтарылады.
bool IsSymbol(char c);	Егер c символ болса, онда true мәні қайтарылады.
bool IsUpper(char c);	Егер c символы бас әріп болса, онда true мәні қайтарылады.
bool IsWhiteSpace(char c);	Егер c символы бос орын болса, онда true мәні қайтарылады.
char ToLower(char c);	c символын кіші әріпке ауыстырады
String ToString(char c);	c символын жолға ауыстырады
char ToUpper(char c);	c символын бас әріпке ауыстырады

`GetUnicodeCategory()` әдісі `char` класына сәйкес (`UnicodeCategory`) категориясының бір мәнін қайтарады. Кейбір категориялардың сипаттамасы 8.3-кестеде көрсетілген.

8.3-кесте – `Char` класының символдар категориялары (UnicodeCategory).

Категория атауы	Символдың сипаттамасы
ClosePunctuation	Фигуралық, квадрат, дөңгелек жақша сияқты жұп символдарға арналған жабушы символ
Control	Басқарушы символ
CurrencySymbol	Ақша бірлігінің символы
DashPuntuation	Сызықшалар (тире, дефис)
DecimalDigitNumber	ондық сан
EnclosingMark	жабушы маркер
FinalQuotePunctuation	Дәйексөздің жабушы маркері (мысалы, жабушы тырнақша)
Format	Пішімдеуші символ
InitialQuotePunctuation	Дәйексөздің ашушы маркері
LetterNumber	Әріптік сан (мысалы, рим цифры)
LineSeparator	Жолдарды бөлгіш
LowerCaseLetter	Кіші әріптер
MathSymbol	Математикалық символ
ModifierLetter	Символ алдынғы символдың модификаторы болып келеді
NonSpacingMark	Негізгі символды өзгертетін, бос орыннан өзге символ.
OpenPunctuation	Ашылатын жақшаны көрсететін символ (фигуралық, квадратты, дөңгелек)
OtherLetter	Әріп, бірақ кіші, үлкен әріптер емес.

OtherNotAssigned	Бұл символ Unicode-символы емес
OtherNumber	Ондық немесе әріптік емес сандық символ (мысалы, бөлшектің символы)
OtherPunctuation	Фигуралық, квадрат, дөңгелек жақша сияқты жұп символдарға арналған ашушы символ
OtherSymbol	Математикалық емес символ, ағымдағы символ немесе өзгертуші символы
ParagraphSeparator	Азат жолды бөлушісі
PrivateUse	Жиі қолданылатын символ (private-use)
SpaceSeparator	Пішімдеуші немесе басқарушы символ емес бөлуші
SpacingCombiningMark	Символды таңбалау орнының енін өзгертуші символ
TitlecaseLetter	Мәтіннің ең бірінші бас әрпі (азат жол маркері)
UpercaseLetter	Бас әріп

Айнымалылар мен `char` класына тиісті кейбір әдістер қолданылған мысалда алфавит символдарының `Unicode` кодына және керісінше түрлендірудің нұсқалары қарастырылған. `Char` класының кейбір әдістерін бас әріптерді кіші әріптерге ауыстыру үшін түрлендіру. Символ бойынша оның категориясының тұрақтысы.

Бағдарлама код:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
 { char i;
 int a1, a2;
 for (i = 'a'; i <= 'z'; i++)
 { Console.Write("{0}-", i);
  Console.Write("{0} ", (int)i);
 }
 Console.WriteLine();
 Console.WriteLine();
 a1 = (int)'A';
 a2 = (int)'Я';
 for (int j = a1; j <= a2; j++)
 {
  Console.Write("{0}-", j);
  Console.Write("{0} ", (char)j);
 }
 Console.WriteLine();
 Console.WriteLine();
 for (i = 'A'; i <= 'Я'; i++)
```

```
      {
       Console.Write("{0}-", char.ToLower(i));
       Console.Write("{0} ", (int)char.ToLower(i));
      }
      Console.WriteLine("\n\n");
      System.Globalization.UnicodeCategory cat1, cat2;
      cat1 = char.GetUnicodeCategory('A');
      cat2 = char.GetUnicodeCategory(';');
      Console.WriteLine("'A' - category {0}", cat1);
      Console.WriteLine("';' - category {0}", cat2);
      Console.WriteLine("'4'       -       category       {0}",
char.GetUnicodeCategory('4'));
     Console.WriteLine("'+'       -       category       {0}",
char.GetUnicodeCategory('+'));
      Console.WriteLine();
      Console.WriteLine("Enter pernesin basiniz");
      Console.ReadLine();
      }
      }
     }
```

Бағдарлама жұмысы:

```
     a-97 b-98 c-99 d-100 e-101 f-102 g-103 h-104 i-105 j-106
k-107 l-108 m-109 n-110 o-111 p-112 q-113 r-114 s-115 t-116 u-
117 v-118 w-119 x-120 y-121 z-122

     1040-А 1041-Б 1042-В 1043-Г 1044-Д 1045-Е 1046-Ж 1047-З
1048-И 1049-Й 1050-К 1051-Л 1052-М 1053-Н 1054-О 1055-П 1056-Р
1057-С 1058-Т 1059-У 1060-Ф 1061-Х 1062-Ц 1063-Ч 1064-Ш 1065-Щ
1066-Ъ 1067-Ы 1068-Ь 1069-Э 1070-Ю 1071-Я

     а-1072 б-1073 в-1074 г-1075 д-1076 е-1077 ж-1078 з-1079 и-
1080 й-1081 к-1082 л-1083 м-1084 н-1085 о-1086 п-1087 р-1088
с-1089 т-1090 у-1091 ф-1092 х-1093 ц-1094 ч-1095 ш-1096 щ-1097
ъ-1098 ы-1099 ь-1100 э-1101 ю-1102 я-1103

     'A' - category UppercaseLetter
     ';' - category OtherPunctuation
     '4' - category DecimalDigitNumber
     '+' - category MathSymbol

     Enter pernesin basiniz
```

8.2 Жолдық айнымалы ұғымы

C# тілінде жолдармен жұмыс істеу үшін (мәтіндермен, символдардың кез келген бірігуі) сәйкес класымен анықталатын арнайы `string` типі қолданылады. `String` класы сілтемелік типке жатады

және кез келген ұзындықтағы жолдық айнымалыларды анықтауға көмектеседі.

Барлық сілтемелік типтердегідей айнымалылардың мәні үйіндіде сақталады. Бірақ `string` типіндегі айнымалылардың ерекшелігі – олардың болжап болмайтын «еркін» ұзындығы. Сондықтан `string` типіндегі айнымалының мәнін өзгерту үйіндіде жаңа өзгертілген мәнді құру жолымен орындалады. Бұл орайда басқа айнымалылардың мәндері өзгермейді. Жолдық айнымалылармен жұмыс жасауда осы ерекшелікті ескеру керек.

C# тілінде өзгермейтін (immutable) класс ұғымы да бар. Осы кластар үшін объект мәнін өзгертуге болмайды. Бар объект негізінде әдістер жаңа объект құра алады, бірақ алдынғы объект мәні өзгертілмейді. Осындай өзгермейтін класқа `string` класы да жатады. Осы кластың бір де бір әдісі басқа объекттердің мәнін өзгерте алмайды. `String` класының әдістері мен қасиеттері 8.4 және 8.4-кестесінде көрсетілген.

8.4-кесте – String класы

Қасиет	Сипаттама
public char this[int Index] {get:};	Бұл қасиет жолдан керекті символды алуға көмектеседі (тек оқу үшін)
public int Length(){get:};	Жолдың ұзындығын қайтарады (өзгертілмейді)

`Length` қасиеті `S` жолдық айнымалының «ұзындығын», яғни жолдық айнымалының символдар санын анықтауға көмектеседі, сондықтан бағдарламада жиі қолданылады.

Жол символына оның индексі бойынша қол жеткізу әрбір символ бойынша талдау жүргізген кезде жиі қолданылады.

Мәтін сөздерін сұрыптау немесе жеке жолдық айнымалыларды салыстыру үшін `Compare()` әдісін қолдану керек, мысалы, `s1` және `s2` айнымалыларын салыстыру былай орындалады:

```
if (Compare (s1,s2)==0)
```

Класс әдістерін қолдануды оқу құралының басқа бөлмдерінде қарастырамыз.

Жолдық айнымалыларды қолданудан бұрын оларды бағдарламада жариялауды үйрену керек.

`string` класының объекттері сілтемелік типтегі айнымалылар сияқты жарияланады - класс конструкторы айқын немесе айқын емес шақырылады.

8.5-кесте – String класының кейбір әдістері

Әдіс	Сипаттама
public static int	Екі жолды салыстырады және егер S1 < S2 болса,

Compare (string S1, string S2);	теріс сан қайтарылады. Егер S1 == S2 болса, нөл саны, егер S1 > S2 болса, оң сан қайтарылады. Әдісті орындау барысында символдар регистрінің айырмашылығы, мезгіл мен ақша бірлігінің ұлттық жазылуы , т.б. ескерілмейді.
public static int CompareOrdinal(string S1, string S2);	Екі жолды символдарының сандық мәндерін салыстыру арқылы салыстырады
public static string Concat (params object[]);	Жолдарды біріктіреді
public void CopyTo (int SourceInd, char[] Dest, int DestInd, int Count);	Dest массивінде DestInd символынан бастап мынаны көшіреміз: жолдың SourceInd символынан бастап Count санынан аспайтын символдарды
public bool EndsWith (string S);	Егер жол S ішкі жолымен аяқталса, true қайтарылады.
public static string Format (string S, params object[]);	S жолын құрады
public int InfexOf(string S);	S ішкі жолы немесе символы табылса, оның бірінші индексі қайтарылады, егер табылмаса, теріс сан қайтарылады.
public static string Join (string S, params string[] ss);	ss жолдарын S бөлгіші арқылы бір ұзын жолға біріктіреді.
public static int LastIndexOf (string S);	S ішкі жолы немесе символы табылса, оның соңғы табылған индексі қайтарылады, егер табылмаса, теріс сан қайтарылады.
public string PaddLeft (int TotalWidth, char C);	Жолдың ұзындығы TotalWidth тең болғанша оны сол жағынан C символдарымен толықтырады.
public static string Remove (int StartIndex, int count);	Ағымдағы жолдан StartIndex символынан бастап count аспайтын символдарды жояды.
public string Replace (string S, string ss);	Ағымдағы жолдың S ішкі жолын ss түрлендіреді.
public string[] Split(params char[] delimiters);	Жолды delimiters шектеулеріне сәйкес сөздерге бөледі.
public bool StartsWith (string S);	Ағымдағы жол S ішкі жолынан басталса, true қайтарылады.
public string SubString (int BegInd, int	Ағымдағы жолдан BegInd және EndInd символдарымен шектелген ішкі жолды қайтарады.

EndInt);	
public string TpLower();	Ағымдағы жолды кіші әріптермен жазып қайтарады.
public string Insert (int Ind. string S);	Ind индексінен бастап S ішкі жолын орналастырады.
public string Trim();	Бос орындары өшірілген жол қайтарылады.
public string TrimEnd();	Бос орындар жойылған алғашқы жолды қайтарады
public string TrimStart();	Бос орындар жойылған алғашқы жолды қайтарады
public string ToUpper();	Ағымдағы жолды үлкен әріптермен жазып қайтарады.

Жолдық айнымалыны жариялаған кезде конструктор айқын шақырылмайды. Мысалы:

string st1 = "Жол";

Жолдық айнымалыны `string` класының конструкторларын айқын шақырып жариялауға болады, мысалы: бірінші ұзындығы нөл болатын жолдық айнымалыны жариялау, ал одан кейін оған мәнді меншіктеу:

string st2 = new string(' ', 0);

st2 = "Жол 2";.

Жолдық айнымалыны жариялаудан кейін оны бағдарламада түрлі операцияларда қолдануға болады. Негізгі операциялар: меншіктеу, баламалылығын тексеру (== немесе !=), жолдарды тіркестіру (қосу).

`Егер st1 жолдық` айнымалыға `басқа st2 жолдық` айнымалы меншіктелсе, онда `st1` айнымалысы `st2` айнымалының мәні сақталған үйінді (жады аумағындағы адрес) сілтемесіне ие болады. Сонымен жолдық айнымалыларды меншіктеуде олардың мәндерінің орнына сілтемелер «көбейтіледі».

Басқа сілтемелік айнымалылардан өзге жолдық айнымалылардың баламалылығын тексеру кезінде олардың сілтемелері емес, мәні салыстырылады.

Тіркестіру операциясын орындау барысында (`Concat() әдісі – +` операциясы) бір жолдық айнымалының мәніне басқа жолдық айнымалының мәні қосылады. Сондықтан алынған мән бір топта жаңа сілтеме бойынша орналасады.

Жолдық айнымалылар қолданылатын мысалды қарастырайық:

```
using System;
namespace ConsoleApplication1
{ class Program
 {
 static void Main()
 {
  string a = "Tabigat ";
```

```
  Console.WriteLine("a sozi = {0}", a);
  string b = new string(' ', 0);
  Console.WriteLine("b sozinin yzindigi = {0} символов",
b.Length);
  Console.WriteLine("b sozinin engiziniz");
  b = Console.ReadLine();
  Console.WriteLine("b sozi = {0}", b);
  string c = a + b;
  Console.WriteLine("Operazia zhymisi c = a + b; c = {0}", c);
  Console.WriteLine("a sozinin yzindigi = {0} символов",
a.Length);
  Console.WriteLine("b sozinin yzindigi = {0} символов",
b.Length);
  Console.WriteLine("Olardin ortak yzindigi = {0} символов",
c.Length);
  Console.WriteLine("Operazia zhymisi if (string.Compare(a, b)
>= 0) вывод a > b;");
  Console.WriteLine("   else (вывод b >= a);");
  if (string.Compare(a, b) >= 0) Console.WriteLine(" a > b");
  else Console.WriteLine(" b >= a");
  c = a.Remove(1, 2);
  Console.WriteLine("c = a.Remove(1,2); c = {0}", c);
  Console.WriteLine("Enter pernesin basiniz");
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
a sozi = Tabigat
b sozinin yzindigi = 0 символов
b sozinin engiziniz
Tabigat
b sozi = Tabigat
Operazia zhymisi c = a + b; c = Tabigat Tabigat
a sozinin yzindigi = 8 символов
b sozinin yzindigi = 7 символов
Olardin ortak yzindigi = 15 символов
Operazia zhymisi if (string.Compare(a, b) >= 0) вывод a >
b;
   else (вывод b >= a);
 a > b
c = a.Remove(1,2); c = Tigat
Enter pernesin basiniz
```

Бағдарламада жолдық айнымалыларды жариялаудың бірнеше жолдары, қосу операциясы арқылы айнымалыларды біріктіру әдісі, айнымалының «ұзындығын» анықтау үшін пайдаланылатын `Length` қасиеті қолданылады.

Бағдарламада `Compare` бүтін санды әдісі жолдық айнымалыларды салыстыру үшін қолданылған. Диалог режімінде `b` айнымалыға түрлі мәндерді беріп `Compare` әдісінің жұмысын тексеруге болады.

Ағымдағы айнымалыларды қолданатын әдістері a.Remove мысалында қарастырылған.

String класының қасиеті жолдық айнымалыны символдар массиві ретінде қарастырады, және әрбір символды оның индексі арқылы шақырады. Осы char типіндегі қасиет символды өзгертпей, тек қарап шығуға мүмкіндік береді.

Жолдық айнымалылармен жұмыс жасайтын әдістерді қолданатын бірнеше мысалдарды қарастырайық. Мәтіндермен жұмыс жасау барысында міндеттердің бірі мәтінді сөздер массиві түрінде көрсету болып табылады. Мәтіндермен жұмыс ітеу барысындағы мәселенің бірі мәтінді сөздер массив түрінде ұсыну. Осы мәселені шешудің екі жолын қарастырайық – мәтіннің әрбір символын талдау арқылы сөздерді бөлу және Join, Split арнайы әдістерін қолдану арқылы. Split динамикалық әдісі мәтінді элементтерге бөлуді орындайды. Join статикалық әдісі кері операцияны орындайды – элементтерден жолды құрайды.

8.1-есебі. Диалог режімінде 20 сөзден аспайтын мәтін енгізіледі. Барлық сөздерді жеке массивке орналастыру және шығару керек. Бір немесе бірнеше «бос орын» символымен бөлінген кез келген символдар жиыны сөз болып есептеледі.

Бағдарлама коды:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
 {
  int j, k;
  string[] clova;
  clova = new string[20];
  string str1 = new string(' ', 0);
  string b = new string(' ', 0);
  string str2 = new string(' ', 0);
  Console.WriteLine("Matindi engiziniz");
  str1 = Console.ReadLine();
  j=0;
  for (int i = 0; i < str1.Length; i++)
  {
  if (str1[i]!=' ')
  {
  b = str1[i].ToString();
  str2 = str2 + b;

  }
  else
   if (str2[0] != ' ')
   {
```

```
   clova[j] = str2;
   str2 = "";
   j++;
   }
  }
  if (str2[0] != ' ')
  {
  clova[j] = str2;
  str2 = "";
  j++;
  }
  for (int i=0;i<j;i++)
  Console.WriteLine(" {0} soz = {1} ", i+1, clova[i]);
  Console.WriteLine("Enter pernesin basiniz");
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
Matindi engiziniz
20 sozden aspaitin matin engizildi
 1 soz = 20
 2 soz = sozden
 3 soz = aspaitin
 4 soz = matin
 5 soz = engizildi
Enter pernesin basiniz
```

Осы бағдарламада мына үзінді

```
  if (str2[0] != ' ')
  {
  clova[j] = str2;
  str2 = "";
  j++;
  }
```

екі рет қолданылады. Мәтіннің соңында «бос орын» символы болмауы мүмкін, сондықтан `str2` жолдық айнымалының «бос орын» символына теңдігі тексеріледі. Егер `str2` айнымалы бос орынға тең болмаса, онда `clova[j]` массивіне соңғы сөз қосылады.

Бос орынды мәтінді тергеннен кейін арнайы жолмен қосуға болады, онда тексеруді орындамауға болады.

8.2-есебі. Жолда алгебралық өрнек жақшаларсыз жазылған. Мысалы, A·B+A·C·D+C·X. Өрнек диалог режімінде енгізіледі. Алгебралық өрнектегі көбейткіштер санын анықтау керек.

Есепті шешудің бір жолы – өрнекті қосылғыштар түрінде көрсету және әрбір қосылғышта көбейту символдарын есептеу. Егер қосылғышта көбейту символы бар болса, онда осы қосылғыштың көбейткіштер саны бірге артады, әйтпесе қосылғышта көбейткіштер жоқ. Барлық

қосылғыштардың көбейткіштерін қосып, өрнектегі жалпы көбейткіштер санын табамыз. Split динамикалық әдісі мәтінді элементтерге – қосылғыштарға бөледі, бөлгіш ретінде ‘+’ символы қолданылады. Қосылғыштардан тұратын алгебралық өрнекті құрастыру Join әдісі арқылы орындалады. Split және Join әдістері жолдағы жеке сөздермен жұмыс жасауды жеңілдетеді.

Мысалы, c:\Program Files\Microsoft Visual Studio .NET\readmy.html. Мұнда жеке сөз болып диск, барлық каталогтар мен файл аттары есептеледі, ал сөзді анықтауда кері көлбеу сызық қолданылады. (жеке сөздер: c:, Program Files, Microsoft Visual Studio .NET, readmy.html).

Бағдарлама коды:

```
using System;
namespace ConsoleApplication1
{
class Program
 {
 static void Main()
 {
 int k, n;
 string[] kosilgish;
 string str1 = new string(' ', 0);
 Console.WriteLine("Algebralik ornekti engiziniz");
 str1 = Console.ReadLine();
 // clagaemoe массивінің өлшемі Split әдісі қайтаратын
 // массивтің өлшеміне сәйкес автоматты түрде анықталады
 kosilgish = str1.Split('+');
 n = 0;
 for (int i = 0; i < kosilgish.Length; i++)
 {
 Console.WriteLine("kosilgish[{0}]= {1}", i + 1,
kosilgish[i]);
 k = 0;
 for (int j = 0; j < kosilgish[i].Length; j++)
 if (kosilgish[i][j] == '*') k++;
 if (k != 0) n = n + k + 1;
 }
 Console.WriteLine("Kobeitkishter sani ={0}",n);
 //алгебралық өрнекті "құрастыруды" орындайық.
 string algebr = string.Join("+", kosilgish);
 Console.WriteLine("Jana ornek = {0}", algebr);
 Console.WriteLine("Enter pernesin basiniz");
 Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
Algebralik ornekti engiziniz
a * b * c + ab * d + abc + b * b
kosilgish[1]= a * b * c
kosilgish[2]= ab * d
```

```
kosilgish[3]= abc
kosilgish[4]= b * b
Kobeitkishter sani =7
Jana ornek = a * b * c + ab * d + abc + b * b
Enter pernesin basiniz
```

`Split` және `Join` әдістері тек бір бөлгішті қолданған кезде жақсы жұмыс істейді.

Егер талдау барысында бірнеше бөлгіштер берілсе, онда жолды құрастыру барысында қиындықтар туындауы мүмкін – жолды бастапқы күйдегі қалпына келтіруге болмайды, өйткені қолданылған бөлгіш түрі белгісіз болады.

8.3 Өзін-өзі тексеру сұрақтары

1 C# тілінде Unicode жазуы нені білдіреді?

2 C# тілінде escape-тізбегі не үшін қолданылады?

3 Көлденең табуляцияны қандай escape-тізбегі орындайды?

4 string st1 = new string(' ', 0); жазбасы нені білдіреді?

5 Жолдық айнымалылармен жұмыс жасағанда `Split` динамикалық әдісі не үшін қолданылады?

6 C# тілінде `str.Length қасиеті` нені орындайды? Мұндағы `str` – жолдық айнымалы.

7 C# тілінде `string.Compare(str1, str2)` әдісі нені орындайды? Мұндағы `str1,str2` - жолдық айнымалылар.

8 C# тілінде str1 = string.Concat(str2, str3); әдісі нені орындайды? Мұндағы `str1, str2, str3` – жолдық айнымалылар.

9 Бағдарламаның келесі үзіндісі нені шығарады:

strimg clovo = "abcdefg";

Console.WriteLine(" {0}", clovo.Length); ?

10 Бағдарламаның келесі үзіндісі нені шығарады?

```
char b;
 int i, n;
 string clovo = "abcadeafg";
 b = clovo[0]; n = 0;
 for (i = 0; i < clovo.Length; i++)
  if (clovo[i] == b) n++;
 Console.WriteLine(" {0}", n); ?
```

9 КӨПӨЛШЕМДІ МАССИВТЕР

9.1 Екі өлшемді массивтер

C# тілінде тек бір өлшемді массивтерді (векторлар) ғана емес, сонымен қатар элементтердің саны тек компьютер жадысының көлемімен шектелген екі өлшемді және көп өлшемді массивтерді анықтауға болады .

Үш өлшемді және көп өлшемді ақпараттар әдетте математикалық түрде сипатталып ұсынылады. Кәсіпорын иесінің 5 сауда орыны болсын. Бір жыл ішінде ай сайын 100 тауардың сатылымы жөнінде ақпарат келесі үш өлшемді массив арқылы көрсетіле алады:

int[, ,] tovar new int[5,12,100];,

мұнда

0..4 индексі тауарды сату орындарын анықтайды;

0..11 индексі – сатылым айы;

0..99 индексі – сатылатын тауар нөмері.

Тауарды сату туралы ақпаратты көрген кезде кәсіпкер оны монитор экранында кесте немесе матрица түрінде көруі керек. Мысалы, бір жыл ішінде бірінші сауда орынында тауардың сатылымы немесе ай сайын барлық тауарлардың сатылымы, т.б. Сонымен, көп өлшемді массивті екі өлшемді массивке айналдыру керек.

Көп өлшемді және екі өлшемді массивтермен жұмыс істеу технологиясы бірдей. Сондықтан екі өлшемді массивтерді қарастырамыз.

9.1-есеп. Минус 50-ден 100-ге дейінгі аралықтағы кездейсоқ бүтін сандардан тұратын 6×6 матрицасын құру керек. Барлық оң және теріс сандардың қосындысын есептеп, оны шығару керек. Матрицаның ең үлкен және ең кіші элементтерін тауып, экранға шығарыңыз.

Бағдарлама коды:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 public static int sp, so;

 public static void sozd(int[,] ma)
 {
  Random rnd = new Random();
  Console.WriteLine("Matrisa kyrildi!!");
  for (int i = 0; i < 6; i++)
  {
  for (int j = 0; j < 6; j++)
  {
   ma[i,j] = rnd.Next() % 101 - 50;
   Console.Write(ma[i, j]+"\t");
  }
```

```
  Console.WriteLine();
  }
 }
 public static void polotr(int[,] ma)
 {
  sp = 0; so = 0;
  for (int i = 0; i < 6; i++)
  for (int j = 0; j < 6; j++)
   if (ma[i,j] < 0)
   so = so + ma[i,j];
   else sp = sp + ma[i,j];
 Console.WriteLine("On sandardin kosindisi = {0}", sp);
 Console.WriteLine("Teris sandardin kosindisi = {0}", so);
 }
 public static void maxmin(int[,] ma)
 {
  int maxi = -100, mini = 150;
  for (int i = 0; i < 6; i++)
  for (int j = 0; j < 6; j++)
  {
   if (maxi < ma[i,j]) maxi = ma[i,j];
   if (mini > ma[i,j]) mini = ma[i,j];
  }
  Console.WriteLine("Maksimal san = {0}", maxi);
  Console.WriteLine("Minimal san = {0}", mini);
 }
 static void Main()
 {
  int[,] a = new int[6, 6];
  int k = 0;
  string buf;
  while (k < 4)
  {
 Console.WriteLine("1 - Matrisanin kuru jane shigaru 6x6");
 Console.WriteLine("2 - On jane teris sandardin kosindisin
tabu");
 Console.WriteLine("3 - Matrisanin maksimal, minimal sandarin
tabu");
 Console.WriteLine("4 - Bagdarlamadan shigu");
 Console.WriteLine("Menu punktin tandaniz");
  buf = Console.ReadLine();
  k = Convert.ToInt32(buf);
  switch (k)
  {
  case 1: sozd(a); break;
  case 2: polotr(a); break;
  case 3: maxmin(a); break;
  default: break;
  }
  }
 }
 }
```

```
}
```

Бағдарлама жұмысы:

```
1 - Matrisanin kuru jane shigaru 6x6
2 - On jane teris sandardin kosindisin tabu
3 - Matrisanin maksimal, minimal sandarin tabu
4 - Bagdarlamadan shigu
Menu punktin tandaniz
1
Matrisa kyrildi!!
-43 50 31 42 39 5
33 -9 40 25 -16 45
-43 19 26 -27 -17 -12
24 26 21 -13 -4 2
-11 15 1 -43 40 50
8 13 -41 -37 -21 13
1 - Matrisanin kuru jane shigaru 6x6
2 - On jane teris sandardin kosindisin tabu
3 - Matrisanin maksimal, minimal sandarin tabu
4 - Bagdarlamadan shigu
Menu punktin tandaniz
2
On sandardin kosindisi = 568
Teris sandardin kosindisi = -337
1 - Matrisanin kuru jane shigaru 6x6
2 - On jane teris sandardin kosindisin tabu
3 - Matrisanin maksimal, minimal sandarin tabu
4 - Bagdarlamadan shigu
Menu punktin tandaniz
3
Maksimal san = 50
Minimal san = -43
1 - Matrisanin kuru jane shigaru 6x6
2 - On jane teris sandardin kosindisin tabu
3 - Matrisanin maksimal, minimal sandarin tabu
4 - Bagdarlamadan shigu
Menu punktin tandaniz
```

Есепті шешу алгоритмін дайындау барысында есеп жеке бөліктерге бөлінді: матрицаны құру мен оны шығару процессі, барлық оң және теріс элементтердің қосындысын есептеу, матрицаның ең үлкен және ең кіші элементтерін табу. Аталған процесстерді `static void Main()` әдісінде жеке статистикалық әдістер арқылы орындау бағдарламада менюді ұйымдастырады, бағдарлама кодының жеңіл және түсінікті болуына мүмкіндік береді.

Бағдарламада менюге арналған цикл меню тармағы таңдалғаннан кейін аяқталады (меню тармағы 4-ке тең).

Меню тармағының жұмысы `switch` операторы арқылы орындалады.

`switch` операторы мына пішімде жазылады:

switch (өрнек)

{
case таңдау тұрақтысы_1 : [операторлар тізімі;]
case таңдау тұрақтысы_2 : [операторлар тізімі;]
. . .
[default операторлар тізімі;]
}

`Switch` операторы екі бөлімнен тұрады. Оператордың бірінші бөлімінде `switch` қызметтік сөзі, ал одан кейін дөңгелек жақшаларда өрнек жазылады. Өрнек кез келген типте бола алады. Оны айқын емес түрде бүтін санды типтерге немесе `char типіне` түрлендіруге болады.

Екінші бөлімде case таңдау операторы арқылы бағдарламаны жалғастыру мүмкін нұсқалары анықталады. Егер `switch` сөзінен кейін өрнек 2-ге тең болса, онда бағдарламада тұрақтысы 2-ге тең case таңдау операторы ізделінеді және осы таңбадан (меткадан) кейін тұрған барлық операторлар орындалады.

Егер case таңдау операторының тұрақтысы мен `switch` операторынан кейін тұрған өрнектің мәні тең болмаса, онда бағдарлама `default` қызметтік сөзінен кейін тұрған операторларды немесе `switch` операторынан кейін орналасқан операторды орындауға көшеді.

«Операторлар тізімі» өту операторларымен аяқталуы тиіс, мысалы, `break, goto` немесе `return` операторлары.

Әдетте ең соңында `break` операторы қолданылады, ол `switch` операторын аяқтайды. Ол келесі таңдау операторының орындалуына жол бермейді.

Бірнеше таңдау операторларын ретті орындау үшін әдетте `goto` операторы қолданылады.

9.2 Көп өлшемді массивтер

Анықтама бойынша массивтер бір типтегі айнымалыларды және бір өлшемді массивтерді біріктіреді. Біріктірілген массивтер көп өлшемді массивтер деп аталады.

Егер біріктірілген массивтердің өлшемдері бірдей болса, онда ондай көп өлшемді массивтер «тіктөртбұрышты» массивтер деп аталады.

Егер біріктірілген массивтердің өлшемдері бірдей болмаса, онда ондай көп өлшемді массивтер «сынық» (jagged) массивтер деп аталады.

Тіктөртбұрышты массивтермен жұмыс жасауға арналған кейбір алгоритмдер алдыңғы есепте қарастырылды.

«Сынық» массивтер элементтерін құру және өңдеу процестерін қарастырайық.

9.2-есеп. Кәсіпорын иесінің 5 сауда орыны бар. Әрбір сауда орынында түрлі тауарлар саны 10 мен 30 арасындағы кездейсоқ сан

болады. Әрбір тауардың құны 10 мен 70 ш.б. аралығындағы кездейсоқ сан болады. 5 сауда орынында өткізілетін тауар санын және олардың жалпы құнын табу керек.

Бағдарлама коды:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
 {
 int[][] mtovar = new int[5][];
 int k = 0;
 int j,kol,cto;
 string buf;
 Random rnd = new Random();
 //Сыныќ массивті жариялау
 for (int i = 0; i < 5; i++)
 {
 j = rnd.Next() % 21 + 10;
 mtovar[i] = new int[j];
 }
 Console.WriteLine("Matrisa kurildi!!");
 kol = 0; cto = 0;
 // Сыныќ массивті ќуру және экранға шығару
for (int i = 0; i < 5; i++)
 {
 kol = kol + mtovar[i].Length;
 for ( j = 0; j < mtovar[i].Length; j++)
 {
 mtovar[i][j] = rnd.Next() % 61 + 10;
 cto = cto + mtovar[i][j];
 Console.Write(mtovar[i][j]+" ");
 }
 Console.WriteLine();
 }
 Console.WriteLine();
 Console.WriteLine("Tayar sani = {0}", kol);
 Console.WriteLine("Olardin bagasi = {0}", cto);
 Console.WriteLine("Enter pernesin basiniz");
 Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
     Matrisa kurildi!!
     16 26 20 29 15 25 67 68 12 12 34 61 14 36 52 58 24 50 26
38 40 58 34 20 54 64 27
      57 18
     27 36 14 29 36 18 52 50 23 41 35 65 24 70 46 32 18 13 41
59
```

```
     40 33 33 19 39 60 29 60 68 46 19 69 20 60 56 47
     27 42 49 28 68 18 11 16 62 44 23 59 59 59 43 58 57 69 46
62 10 66
     61 44 57 63 53 45 32 37 52 32 21 67 12 47 35 20

     Tayar sani = 103
     Olardin bagasi = 4136
     Enter pernesin basiniz
```

Сынық массивті жариялаған кезде екінші индекс өлшемі көрсетілмейді, индекстер квадрат жақшаға алынады (үтір қолданылмайды), мысалы:

```
int[][] mtovar = new int[5][];
```

«Бағаналар» саны әр сынық массив «жолдарды» үшін динамикалық түрде анықталады:

```
j = rnd.Next() % 21 + 10;
mtovar[i] = new int[j];
```

Сынық массив элементтеріне әрбір индексті квадрат жақшаға орналастыру керек, мысалы:

```
mtovar[i][j] = rnd.Next() % 61 + 10;
```

9.3 Array массивтер класы

Айнымалыларды массивке біріктіру түрлі алгоритмдерде және деректерді өңдеу жүйелерінде қолданылады, мысалы, реляциялық деректер базасында немесе тізімдермен берілген массивтерде сұрыптау мен іздеу алгоритмдерінде. Сондықтан көптеген визуалды бағдарламалау орталарында массивтермен жұмыс істеуге арналған арнайы кластар бар

C# тілінде массивтермен жұмыс жасау үшін `System.Array` класы қолданылады. Осы класта массивтермен жұмыс істеуге арналған статикалық қасиеттер мен әдістер бар. Ең жиі қолданылатын қасиеттердің бірі - массивте элементтер санын есептейтін `Length` типіндегі қасиет. Мысалы, `masi` массивінің элементтер саны `masi.Length` арқылы анықталады. Бүтін типтегі `Rank` қасиет – массив өлшемін анықтауға мүмкіндік береді. `System.Array` класының ең жиі қолданылатын әдістері 9.1-кестеде көрсетілген.

9.1-кестесі – System.Array класының әдістері

Әдіс	Сипаттама
Static int BinarySearch (Array, object, IComparer);	Екілік іздеу әдісі. Сұрыпталған бір өлшемді Array массивінде IComparer интерфейсі арқылы object элементін іздейді және элемент индексін қайтарады, егер элемент табылмаса теріс санды қайтарады.
public static void	Ағымдағы бір өлшемді массивтен барлық

CopyTo (Array, Index);	элементтерді Array массивіне Index индексінен бастап көшіреміз.
public static Array CreateInstance(Type ElementsType, int[] Lengths, int[] LowerBounds);	Әрбір өлшемі бойынша элементтер саны Lengths және Индекстердің төменгі шегі LowerBounds болатын, ElementsType типіндегі элементтерден тұратын көп өлшемді массивті құрайды. Қайта жүктелетін әдістер индекстері 0-ден басталады бір өлшемді және екі өлшемді массивтерді құруға мүмкіндік береді.
public int GetLowerBound (Dimension);	Dimension өлшемі бойынша индекстің ең кіші мәнін қайтарады.
public int GetUpperBound (Dimension);	Dimension өлшемінде индекстің ең үлен мәнін қайтарады.
public static void Reverse (Array);	Бір өлшемді Array массивінің элементтер тәртібін кері орналастырады.
public static void Sort (Array);	Бір өлшемді Array массивін сұрыптайды
public static void Clear (Array, Index, Length);	Массивті тазарту. Бір өлшемді Array массивінде Length элементтерін орналастырамыз. Элементтер типіне қарай Index элементінен бастап мәндері 0, false немесе null болады.
public int GetLenght (Dimension);	Dimension өлшемі бойынша массив элементтерінің санын қайтарады.

Есепті шығару үшін `System.Array` класын қолданып, бағдарламаны жазу процесін қарастырайық. Сонымен қатар класс әдістерінің массиві құру процестері және осы массив пен әдеттегі жолмен құрылған массив элементтерін өңдеу үшін класс әдістерін қолдану қарастырылады.

9.3-есеп. Минус 40 пен 40 аралығындағы кездейсоқ бүтін сандардан тұратын A массивін құру. Массивті шығару. Массивтің барлық элементтерін сұрыптау. Құрылған массив элементтерін өлшемі жағынан бірдей басқа массивке көшіру. Онда диалог режимінде берілген кілт бойынша элементтің бинарлық іздестірілуін орындау. Бағдарламаны жазғанда `System.Array` класының әдістерін қолдану.

Бағдарлама коды:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 public static int[] ctl = { 15 };
 public static int[] niz = { 1 };
```

```
 public static Array masi = Array.CreateInstance(typeof(int),
ctl, niz);
 public static void sozd()
 {
 Random rnd = new Random();
 Console.WriteLine("Massiv kurildi!!: ");
 for (int i = masi.GetLowerBound(0); i <= masi.Length; i++)
 masi.SetValue((rnd.Next(81) - 40), i);
 for (int i = masi.GetLowerBound(0); i <= masi.Length; i++)
 Console.Write(masi.GetValue(i) + " ");
 Console.WriteLine();
 }
 static void Main()
 {
 int[] a = new int[15];
 int k = 0;
 string buf;

 sozd();
 Console.WriteLine("Massivti sorttaimiz jane shigaramiz: ");
 Array.Sort(masi);
 foreach (int i in masi)
 Console.Write(i + " ");
 Console.WriteLine();
 Console.WriteLine("Massivti bytin sandardan tyratin massivke
koshiremiz: ");
 Array.Copy(masi, a, masi.Length);
 foreach (int i in a)
 Console.Write(i + " ");
 Console.WriteLine();

 //массивті реверстейміз
 Array.Reverse(a);
 Console.WriteLine("Shana massivti reverstey jane shigary: ");
 foreach (int i in a)
 Console.Write(i + " ");
 Console.WriteLine();
 //санды кілт бойынша бинарлы іздеу
 Console.WriteLine("Kilt boinsha massivte sandi binarli
izdey");
 Console.WriteLine("Izdey kiltin engiziniz");
 buf = Console.ReadLine();
 k = Convert.ToInt32(buf);
 int idx = Array.BinarySearch(masi, k);
 Console.WriteLine("Sannin indeksi = " + idx);
 Console.WriteLine("jok kiltti engiziniz");
 buf = Console.ReadLine();
 k = Convert.ToInt32(buf);
 idx = Array.BinarySearch(masi, k);
 Console.WriteLine("Sannin indeksi = " + idx);
 Console.ReadLine();
 }
```

```
 }
}
```

Бағдарлама жұмысы:

```
Massiv kurildi!!:
-15 -31 -2 -31 20 13 -30 12 19 25 -24 -32 -22 3 19
Massivti sorttaimiz jane shigaramiz:
-32 -31 -31 -30 -24 -22 -15 -2 3 12 13 19 19 20 25
Massivti bytin sandardan tyratin massivke koshiremiz:
-32 -31 -31 -30 -24 -22 -15 -2 3 12 13 19 19 20 25
Shana massivti reverstey jane shigary:
25 20 19 19 13 12 3 -2 -15 -22 -24 -30 -31 -31 -32
Kilt boinsha massivte sandi binarli izdey
Izdey kiltin engiziniz
12
Sannin indeksi = 10
jok kiltti engiziniz
18
Sannin indeksi = -13
```

`Array.CreateInstance(typeof( int),ctl,niz);` әдісі арқылы 15 элементтен тұратын бір өлшемді `masi` массивін құраймыз `(Lengths` қасиеті `ctl` айнымалысына тең), бірінші `LowerBounds` индексі `1`-ге тең (индекс мәні 1-ден үлкен бола алады, бірақ 0-ге тең емес).

`For` циклінде `masi.GetLowerBound(0)` қолданылады. Айнымалының бастапқы мәні 1-ге, ал соңғы мәні `– masi.Length 15`-ке тең.

```
for (int i = masi.GetLowerBound(0); i <= masi.Length; i++)
masi.SetValue((rnd.Next(81) - 40), i);
```

Массив элементтерін қолдану `System.Array` класының әдістері арқылы ғана мүмкін: жазуда `- masi.SetValue` және оқуды орындағанда `- masi.GetValue.`

Массивті сұрыптауды `Array.Sort(masi);` статикалық әдісімен орындаймыз және оны `foreach` циклі арқылы монитор экранына шығарамыз:

```
foreach (int i in masi)
Console.Write(i + " ");
Console.WriteLine();
```

`System.Array` класының әдістері арқылы құрылған массивті `a` массивіне көшіреміз:

Array.Copy(masi, a, masi.Length);

Ары қарай `a` массиві үшін `System.Array` класының реверс әдісі мен бинарлық іздестіруді қолдану қарастырылған.

9.4-есебі. 0 мен 20 аралығындағы кездейсоқ бүтін сандардан тұратын 5x5 матрицасын құрып, оны шығару керек. Матрицаны 25 элементтен тұратын массивке қайта жазып, осы массивтің барлық элементтерін

сұрыптау керек. Массивті матрицаға кері жазуды орындаңыз, сұрыпталған матрицаны шығарыңыз.

Бағдарлама коды:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 public static Array masi = Array.CreateInstance(typeof(int),
5, 5);
 public static void sozd()
 {
  int kk;
  Random rnd = new Random();
  Console.WriteLine("Matriza kyrildi: ");
  for (int i = 0; i <= 4; i++)
  for (int j = 0; j <= 4; j++)
  {
   kk = rnd.Next(21);
   masi.SetValue(kk, i, j);
  }
  for (int i = 0; i <= 4; i++)
  {
  for (int j = 0; j <= 4; j++)
   Console.Write(masi.GetValue(i, j) + "\t");
  Console.WriteLine();
  }
 }
 static void Main()
 {
  int[] a = new int[25];
  int k = 0;
  string buf;
  sozd();
  Console.WriteLine("Matrizani massivke koshiry, ekranga
shigary: ");
  int j = 0;
  foreach (int i in masi)
  {
  Console.Write(i + " "); a[j] = i; j++;
  }
  Console.WriteLine();
  Console.WriteLine("Massivti syriptay, ekranga shigary: ");
  Array.Sort(a);
  foreach (int i in a)
  Console.Write(i + " ");
  Console.WriteLine();
  Console.WriteLine("Massivti matrizaga koshiry: ");
  k = 0;
  for (int i = 0; i <= 4; i++)
  {
  for (j = 0; j <= 4; j++)
```

```
  {
   masi.SetValue(a[k], i, j); k++;
   Console.Write(masi.GetValue(i, j) + "\t");
  }
  Console.WriteLine();
  }
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
Matriza kyrildi:
2 9 3 17 8
4 14 12 17 11
9 18 8 7 5
16 7 15 6 12
4 2 15 6 16
Matrizani massivke koshiry, ekranga shigary:
2 9 3 17 8 4 14 12 17 11 9 18 8 7 5 16 7 15 6 12 4 2 15 6
16
Massivti syriptay, ekranga shigary:
2 2 3 4 4 5 6 6 7 7 8 8 9 9 11 12 12 14 15 15 16 16 17 17
18
Massivti matrizaga koshiry:
2 2 3 4 4
5 6 6 7 7
8 8 9 9 11
12 12 14 15 15
16 16 17 17 18
```

Екі өлшемді массив элементтерін құру және өңдеу үшін бағдарламада `System.Array` класының әдістері қолданылған. `System.Array` класының кейбір әдістерін тек бір өлшемді массивтер элементтеріне ғана қолдануға болады, мысалы, элементтерді сұрыптау. Сондықтан бағдарламада кең таралған жолы қарастырылады:, өңдеуді таңдау әдісімен орындау, екі өлшемді массив элементтерін бір өлшемді массивке қайтадан жазу және керісінше бір өлшемді массив элементтерін екі өлшемді массивке жазу.

9.4 Өзін-өзі тексеру сұрақтары

1 `Switch` операторы не үшін қолданылады?

2 `Switch<өрнек>` операторында <өрнек> не үшін қолданылады?

3 Неге `Switch` операторында case таңбасының (метка) әрекеті `break` операторымен аяқталуы керек?

4 a матрицасына неше сан жазуға болады, егер `int[,] a = new int[3,4];`?

5 Бағдарламаның келесі үзіндісі нені орындайды?

```
for (int i = 0; i < 6; i++)
{
for (int j = 0; j < 6; j++)
 Console.Write(ma[i, j]+"\t");
Console.WriteLine();
}
```

6 Бағдарламаның келесі үзіндісі нені орындайды?

```
for ( i = 0; i < 9; i++)
for ( j = i+1; j < 10; j++)
{ b = a[i,j]; a[i,j] = a[j,i]; a[j,i] = b; }
```

7 Бағдарламаның келесі үзіндісі нені орындайды?

```
 n = 0;
for (j = 0; j <= 10; j++)
for (I = 0; I <= 10; I++)
if (A[I, j] >= 0) n++;
```

8 Бағдарламаның келесі үзіндісі нені орындайды?

```
for (I = 0; I <= 10; I++)
if (A[I,10-I] > 0) A[I,10-I] = 0;
```

9 Бағдарламаның келесі үзіндісі нені орындайды?

```
for (I = 0; I <= 10; I++)
for (j = 0; j < 10; j++)
for (k = j +1; k <= 10; k++)
if (a[I,j] > a[I,k])
{ b = a[I,j]; a[I,j] = a[I,k]; a[I,k] = b;}
```

10 Бағдарламаның келесі үзіндісі нені орындайды?

```
for (I = 0; I <= 10; I++)
if (A[I,I] > 0) A[I,I] = 0;
```

10 ҚҰРЫЛЫМДАР

10.1 Құрылымдар туралы мағлұмат

Көптеген бағдарламалау тілдерінде массивтерден кейін деректерді сипаттаудың келесі түрі - құрылымдар қолданылады, ол әр түрлі типтегі айнымалыларды біріктіретін деректер типі ретінде қарастырылады.

C# тілінде құрылым дегеніміз - класс типіндегі деректер сипаттамасының форма, бірақ кейбір шектеулері бар.

Сонымен қатар құрылымда класс сияқты әр түрлі типтегі деректер өрісімен қатар осы өрістермен жұмыс жасайтын әр түрлі әдістер, конструкторлар бола алады.

Құрылымда класқа қарағанда өзі бағынатын немесе өзіне бағынышты құрылым болмайды, бірақ Object класының әдістеріне ие.

Құрылымда біріктірілген айнымалылардың саны мен құрамы компьютер жадысының көлемімен ғана шектеледі. Әдетте құрылымда айнымалылар объект немесе процесске сәйкес біріктіріледі. Мысалы, тауар сипаттамасы, адам, технологиялық процесс, биржадағы ставкалар динамикасы туралы деректер, т.б.

Құрылым келесі пішімде анықталады:

[атрибуттар][спецификаторлар] struct құрылым_аты [:интерфейстер]
{ құрылым_денесі }

мұнда,

`атрибуттар` – құрылым туралы қосымша ақпаратты береді;

`спецификаторлар` – әдетте құрылымды қолдану шарттарын анықтайды;

`интерфейстер` – құрылым пайдалана алатын базалық кластар (өрістері жоқ):

`құрылым денесі` – құрылым элементтерінің құрамын анықтайды.

Құрылымды жариялау класты жариялауға өте ұқсас, бірақ екі айырмашылық бар: `class` сөзінің орнына `struct` сөзі қолданылып, міндетті емес интерфейстер тізімі көрсетіледі.

Құрылым денесіне айнымалылар және олардың типтері жазылады. Айнымалы әдетте құрылым өрісі болып аталады. Мысалы, студент туралы жазбаны дайындағанда бойы, салмағы, аяқ киім өлшемі, көзінің түсі сияқты сипаттамалар керек болуы мүмкін. Осы сипаттамаларды құрылымда жазайық:

```
struct stydent
{
 public string name, cvet_gl;
 public int rost, ves, god_roj;
 public float raz_ob;
};
stydent styd = new stydent();
```

Осы мысалда жазба (styd) типіндегі бір айнымалы және жазба өрістеріне арналған түрлі типтегі алты айнымалы жарияланған:

styd.name – жолдық айнымалы;

styd. rost, ctyd.ves, styd.god_roj – бүтін типтегі айнымалы;

styd.raz_ob – нақты типтегі айнымалы;

styd.cvet_gl – жолдық айнымалы.

Бағдарламалауда нақты жазба өрістерін қолданғанда осылай құралған атауды (айнымалы атауы, нүкте символы, өріс атауы) қолдану керек. Мысалы, оларға мәндерді меншіктеу:

styd.name = " Makanov";
styd.cvet_gl = "kara";
styd.rost = 173;
styd.ves = 68;
styd.raz_ob = 27.5;
styd.god_roj = 1986;

Келесі мысалды қарастырайық. Студент типіндегі жазбаны анықтау, мәндерді енгізу, студент туралы мәліметтерді шығару керек.

Бағдарлама коды:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 struct stydent
 {
  public string name, cvet_gl;
  public int rost, ves, god_roj;
  public double raz_ob;
 };

 static void Main()
 {
  stydent styd = new stydent();
  Console.WriteLine("Stydenttin ati {0}", styd.name);
  Console.WriteLine("Kozinin tysi {0}", styd.cvet_gl);
  Console.WriteLine("boi yzindigi {0}", styd.rost);
  Console.WriteLine("tygan jili {0}", styd.god_roj);
  Console.WriteLine("sakmagi {0}", styd.ves);
  Console.WriteLine("Kiim olshemi {0}", styd.raz_ob);
  Console.WriteLine();
  styd.name = "Makanov";
  styd.cvet_gl = "kara";
  styd.rost = 173;
  styd.ves = 68;
  styd.raz_ob = 27.5;
  styd.god_roj = 1986;
  Console.WriteLine("Stydenttin ati {0}", styd.name);
  Console.WriteLine("Kozinin tysi {0}", styd.cvet_gl);
  Console.WriteLine("boi yzindigi {0}", styd.rost);
```

```
  Console.WriteLine("tygan jili {0}", styd.god_roj);
  Console.WriteLine("sakmagi {0}", styd.ves);
  Console.WriteLine("Kiim olshemi {0}", styd.raz_ob);
  Console.WriteLine();
  Console.WriteLine("Enter pernesin basiniz");
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
Stydenttin ati
Kozinin tysi
boi yzindigi 0
tygan jili 0
sakmagi 0
Kiim olshemi 0

Stydenttin ati Makanov
Kozinin tysi kara
boi yzindigi 173
tygan jili 1986
sakmagi 68
Kiim olshemi 27,5

Enter pernesin basiniz
```

Жазба өрістерін экранға шығару өрістерге «нөлдік» мәндер меншіктелетінін көрсетеді.

Деректерді жазбаға біріктіру оларды деректердің басқа құрылымдарында қолдануға мүмкіндік (мысалы, массивте, файлдарда) береді.

10.2 Құрылым массивін қолдану

Компьютер жадысында ақпаратты көрсетудің жиі қолданылатын формаларының бірі - жазбалар массиві түрінде деректерді ұйымдастыру. 10.1 есебін мысалға ала отырып деректерді өңдейтін, жазбалар массиві түрінде көрсетілген кейбір алгоритмдерді қарастырайық.

10.1-есеп. Студенттер туралы кездейсоқ 20 жазбалар массивін құру. Әрбір жазба студенттің тегі (алты кездейсоқ әріп, біріншісі бас әріп) мен "Физика. Тарих. Матем. Бағдарл. Информ." пәндерінің бес кездейсоқ бағаларынан тұрады.

Құрылған жазбалар массивін шығару. Әрбір студент бойынша бағалар қосындысын есептейтін және шығаратын өңдеу жұмысын орындау керек.

Емтиханнан ең үлкен және ең кіші бағалар алған студенттерді монитор экранына шығару керек.

Кодтар кестесінде бас әріптер (кириллица) 1040 – 1071 нөмерлерінде, ал кіші әріптер 1072 – 1102 аралығында орналасады. Сондықтан студент тегін жазғанда бірінші әріп бірінші диапазоннан, ал қалған бес әріп екінші диапазоннан алынады.

Бағдарлама коды:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 public struct stydent
 {
  public string fio;
  public int[] ocen;
 };
 public static stydent[] styd = new stydent[20];
 public static void sozd()
 {
  char c;
  string s;
  int k, j = 0;
  char[] buka = new char[6];

  Random rd = new Random();
  for (int i = 0; i < 20; i++)
  {
  styd[i].ocen = new int[6];
  j = rd.Next(32) + 1040;
  buka[0] = (char)(j);
  s = buka[0].ToString();
  for (int m = 1; m < 6; m++)
  {
   j = rd.Next(32) + 1072;
   buka[m] = (char)(j);
   s = s + buka[m].ToString();
   k = rd.Next(4) + 2;
   styd[i].ocen[m - 1] = k;
  }
  styd[i].fio = s;
  }
  Console.WriteLine("Studentter tizimi kyrildi");
 }
 public static void printz()
 {
  Console.Write("Nomer \t");
  Console.Write("Ati-zhoni \t");
  Console.Write("Fizika \t");
  Console.Write("Tarih. \t");
  Console.Write("Matem. \t");
  Console.Write("Bagdarl. \t");
  Console.Write("Informat. \t");
```

```
 Console.WriteLine();
 for (int i = 0; i < 20; i++)
 {
 Console.Write("{0}\t{1}\t", i + 1, styd[i].fio);
 for (int m = 0; m < 5; m++)
  Console.Write("{0}\t", styd[i].ocen[m]);
 Console.WriteLine();
 }
 Console.WriteLine();
}
public static void stat()
{
 int k, j;
 Console.Write("ФИО \t");
 Console.WriteLine("Ball kosindisi");
 for (int i = 0; i < 20; i++)
 {
 k = 0;
 for (j = 0; j < 5; j++)
  k = k + styd[i].ocen[j];
 Console.WriteLine("{0} {1}", styd[i].fio, k);
 styd[i].ocen[5] = k;
 }
 Console.WriteLine();
}

public static void maxmin()
{
 int k = 0, j = 0;
 int maxi = -100;
 int mini = 150;
 for (int i = 0; i < 20; i++)
 {
 if (maxi < styd[i].ocen[5])
 { maxi = styd[i].ocen[5]; j = i; }
 if (mini > styd[i].ocen[5])
 { mini = styd[i].ocen[5]; k = i; }
 }
 Console.WriteLine();
 Console.WriteLine("{0} - Maksimal baldi {1} aldi", maxi,
styd[j].fio);
 Console.WriteLine("{0} - Manimal baldi {1} aldi", mini,
styd[k].fio);
}
static void Main(string[] args)
{ sozd();
 printz();
 stat();
 maxmin();
 Console.ReadLine();
}
}
```

```
}
```

Бағдарлама жұмысы:

```
Studentter tizimi kyrildi
Nomer Ati-zhoni Fizika Tarih. Matem. Bagdarl. Informat.

1 Юаьоюю 2 2 4 2 4
2 Тфютьз 3 3 3 2 3
3 Прилюс 5 5 3 4 4
4 Ьлшвйъ 3 4 4 4 5
5 Алъпам 5 3 4 3 5
6 Фсбфтз 4 5 2 2 2
7 Яшфылф 2 4 3 4 3
8 Чппюов 3 5 5 4 2
9 Ьчзждц 5 4 5 2 4
10 Алъщчх 2 5 4 2 3
11 Дътфцс 2 3 2 4 5
12 Щыыжоу 3 3 2 4 5
13 Жондщю 2 2 2 5 4
14 Стеъюа 5 2 3 3 5
15 Гтслхг 2 5 3 4 4
16 Лфппфу 4 5 2 5 5
17 Йкэджк 4 3 2 4 5
18 Злзжмй 4 4 4 3 3
19 Щзтььд 3 4 5 4 3
20 Дйъвпр 5 2 3 3 3

ФИО Ball kosindisi
Юаьоюю 14
Тфютьз 14
Прилюс 21
Ьлшвйъ 20
Алъпам 20
Фсбфтз 15
Яшфылф 16
Чппюов 19
Ьчзждц 20
Алъщчх 16
Дътфцс 16
Щыыжоу 17
Жондщю 15
Стеъюа 18
Гтслхг 18
Лфппфу 21
Йкэджк 18
Злзжмй 18
Щзтььд 19
Дйъвпр 16

21 - Maksimal baldi Прилюс aldi
14 - Manimal baldi Юаьоюю aldi
```

Бағдарламада әрбір студент бағасының қосындысын сақтау үшін массивте қосымша бағана қолданылған.

10.3 C# тіліндегі тізім

Бағдарламаларды жазғанда әдетте бір бірімен байланысқан атаулары бар бірнеше тұрақтыларды анықтау қажет болады.

Ол үшін деректердің түгенделген типін қолдану ыңғайлы, оның барлық мүмкін мәндері бүтін санды тұрақтылар тізімімен анықталады, мысалы:

```
enum Кемпірқосақ { Қызыл, Тоқсары, Сары, Жасыл, Көк, Күлгін };
```

Жоғарыда келтірілген мысалда әрбір тұрақтыға еш әрекетсіз `int` типіндегі, 0-ден басталатын реттілік мәндер меншіктеледі, бірақ басқа да мәндерді меншіктеуге болады, мысалы:

```
enum Numer { Қызыл = 2, Тоқсары, Сары, Жасыл = 10, Көк };
```

Оранжевый және Сары тұрақтыларына 3 және 4 мәндері, Көк тұрақтысына 11 мәні меншіктеледі.

Әрбір тізім ішіндегі тұрақтылардың атаулары бірегей болуы керек, ал мәндері бірдей бола алады.

Тізімнің тұрақтылардан артықшылығы - байланыстырылған тұрақтылардың көрнекі болуында, сонымен қатар компилятор типтерді тексеруді орындайды, ал құру ортасы тұрақтылардың мүмкін мәндерін көрсетеді, тізім ретінде көрсетеді

Тізімді жазу пішімі:

```
[атрибуттар] [спецификаторлар] enum тізім_атауы [:базалық _тип ]
     тізім_ денесі [ ; ]
```

Тізім спецификаторларының мағынасы класс үшін қолданылатын спецификаторлардың мағынасымен бірдей, сонымен қатар тек `new`, `public`, `protected`, `internal` және `private` спецификаторлары ғана рұқсат етілген.

Базалық тип – тізімді құрайтын элементтер типі. Еш әрекетсіз `int` типі қолданылады, бірақ типті бүтінсанды типтердің (`char` типінен өзге) ішінен таңдап, айқын түрде анықтауға болады: `byte`, `sbyte`, `short`, `ushort`, `int`, `uint`, `long`, `ulong`.

Тізім денесі тұрақтылар атауларынан тұрады, оның әрбіріне мәнді меншіктеуге болады. Егер мән көрсетілмесе, онда ол мән алдында тұрған тұрақтының мәніне бір қосылып есептеледі. Тұрақтылар еш әрекетсіз `public` спецификаторында болады.

Тізімдік типтегі айнымалылармен арифметикалық операцияларды (+, -, ++, - -), разряд бойынша логикалық операцияларды (^, &, |, ~) орындауға, оларды қатынас операциялары (<, <=, >, >=, ==, !=) арқылы салыстыруға және байтпен (sizeof) көлемін есептеуге болады.

Бүтін санды өрнектерде және меншіктеу операцияларында тізімдік типтегі айнымалыларды қолданғанда типті айқын түрлендіру керек.

Мысал (тізім туралы мысал [2] алынған):

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication1
{
 struct Боец
 {
 public enum Воинское_Звание
 {
 Рядовой, Сержант, Лейтенант, Майор, Полковник, Генерал
 }
 public string Фамилия;
 public Воинское_Звание Звание;
 }
 class Program
 {
 static void Main()
 {
 Боец x;
 x.Фамилия = "Иванов";
 x.Звание = Боец.Воинское_Звание.Рядовой;
 for (int i = 1980; i < 2011; i = i + 5)
 {
 if (x.Звание <= Боец.Воинское_Звание.Генерал)
 {
 Console.WriteLine("Shili: {0} {1} {2}", i, x.Звание,
x.Фамилия);
 x.Звание++;
 }
 }
 Console.WriteLine("Enter pernesin basiniz");
 Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
Shili: 1980 Рядовой Иванов
Shili: 1985 Сержант Иванов
Shili: 1990 Лейтенант Иванов
Shili: 1995 Майор Иванов
Shili: 2000 Полковник Иванов
Shili: 2005 Генерал Иванов
```

```
Enter pernesin basiniz
```

10.4 C# тіліндегі файлдар

Кез келген бағдарламалау тілінде файлдармен жұмыс жасау құралдары бар. Көптеген бағдарламалар жұмысты файлдардан деректерді оқудан бастайды және нәтижелерді файлға жазумен аяқтайды

C# тілінде файлдармен жұмыс жасауға арналған бірнеше класс бар.

Белгілі бір кластар тобы (олар `System.IO` атаулар кеңістігінде орналасқан) файлдармен деректер бірлігі сияқты жұмыс істейді. Олардың міндетіне каталогтармен (файлдар папкасы) жұмыс жасау, файлдардың орнын ауыстыру, олардың атауларын өзгерту, көшіру, файл атрибуттары, файлдарды ұсыну кестесімен (FAT) жұмыс жасау кіреді. Осы топты шартты түрде файлдарды өңдеу класы деп атауға болады.

Windows жүйесінде (сонымен қоса C# тілінде) түрлі құрылғылар (оперативтік жад, монитор, дискілік жад, желілер) арасында деректерді жеткізу үшін деректер ағыны (stream) тұжырымдамасы қолданылады.

Деректердің екілік, символдық, жолдық ағынымен анықталған файлдармен жұмыс жасауға көмектесетін арнайы кластар бар. `Осы кластар да System.IO` кеңістігінде анықталған.

C# тілінде құрылымдар, кластар түрінде көрсетілген деректермен жұмыс жасау үшін сериализация ұғымына топтастырылған арнайы класс жиынтығы бар.

Сериализация ұғымын объекттерді, жазбаларды байттар тізбектілігіне түрлендіру процесімен түсіндіруге болады (файлға жазылатын ақпарат байттар тізбектілігімен берілуі керек). Кері процесс - байттар тізбектілігінен объекттерді, жазбаларды қалпына келтіру десериализация деп аталады.

Кейде сериализация дегеніміз – объекттің ағымдағы жағдайын жадыда немесе ақпарат құрылғысында сақтау, ал кері үдеріс - ақпарат құрылғысынан деректерді оқу десериализация деп аталады.

Сериализация мен десериализация процесстерін басқаратын класс `System.Runtime.Serialization` атаулар кеңістігінде орналасады.

10.5 Файлдармен жұмыс жасағанда сериализация мен десериализацияны қолдану мысалы

Студенттер туралы есепте сериализация мен десериализация процесстерін қолдану мысалын қарастырайық.

10.2-есеп. Диалог режімінде студенттердің саны 20-дан аспайтын массивті құру керек. Әрбір жазбада студенттің тегі, топ атауы, емтихан

балы, жасы жазылады. Монитор экранында жазбалар массивін қарап шығуды алдын ала қарастыру керек.

Жазбалар массивін монитор экранында көрсетуді, файлға жазбалар массивін сериализациялау және файлдан жазбалар массивін десериализациялау процесстерін қолдануды, файлды қайтадан қарап шығуды қарастыру керек. Бағдарламада менюді қолдану керек.

Есептің шартына сәйкес 4 өрісі бар құрылымды құрайық: студенттің аты, топ атауы, емтихан балы, студент жасы.

```
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;
namespace ConsoleApplication1
{
 [Serializable]
 struct Stydent
 {
 public String Name; //студенттің аты
 public String Grup; // тобы
 public int Ball; // балл
 public int Let; // жасы
 }
 class Program
 {
 public static Stydent[] Styds = new Stydent[20];
 public static int kol = 0;
 public static void sozd()
 {
  int b;
  string buf;
  Console.WriteLine("Toptagi stydentter sani <= 20 !!");
  buf = Console.ReadLine();
  kol = Convert.ToInt32(buf);
  for (int i = 0; i < kol; i++)
  {
  Console.WriteLine("{0}-stydenttin atin engiziniz", i + 1);
  buf = Console.ReadLine();
  Styds[i].Name = buf;
  Console.WriteLine("{0}-stydenttin tobin engiziniz", i + 1);
  buf = Console.ReadLine();
  Styds[i].Grup = buf;
  Console.WriteLine("{0}-stydenttin balin engiziniz", i + 1);
  buf = Console.ReadLine();
  b = Convert.ToInt32(buf);
  Styds[i].Ball = b;
  Console.WriteLine("{0}-stydenttin shasin engiziniz", i + 1);
  buf = Console.ReadLine();
  b = Convert.ToInt32(buf);
  Styds[i].Let = b;
  }
```

```
 }
 public static void zapf()
 {
  //сериализация үшін ағынды құрамыз:
  FileStream StreamOut = new FileStream("Sytds.dat",
   FileMode.Create, FileAccess.Write);
  //екілік пішімді қолданамыз:
  BinaryFormatter fmt = new BinaryFormatter();
  fmt.Serialize(StreamOut, Styds); // объекттерге
сериализациялау
  StreamOut.Close();   // ағынды жабамыз
 }
 public static void printm()
 {
  Console.WriteLine("{0,20}, {1, 20}, {2, 10}, {3, 10}",
   "Ati", "Tobi", "Ball", "Shasi");
  foreach (Stydent T in Styds)
  {
  if (T.Ball > 0)
   Console.WriteLine("{0,20}, {1, 20}, {2, 10}, {3, 10}",
    T.Name, T.Grup, T.Ball, T.Let);
  }
 }
 public static void vvod()
 {
  int i = 0;
  //десериализация үшін ағынды құрамыз:
  FileStream StreamIn = new FileStream("Sytds.dat",
  FileMode.Open, FileAccess.Read);
  // екілік пішімді қолданамыз:
  BinaryFormatter fmt = new BinaryFormatter();
  Styds =
(Stydent[])fmt.Deserialize(StreamIn);//Десериализациялау
  kol = 0;
  while (Styds[i].Ball > 0) { kol++; i++; }
  StreamIn.Close();  // ағынды жабамыз
 }
 public static void Main()
 {
  int k = 0;
  string buf;
  while (k < 6)
  {
  Console.WriteLine("1 - Massivti kyry");
  Console.WriteLine("2 - Massivti failga shazy");
  Console.WriteLine("3 - Massivti shigary");
  Console.WriteLine("4 - faildan massivti oky");
  Console.WriteLine("5 - Massivti shigary");
  Console.WriteLine("6 - Shigy");
  Console.WriteLine("Menu punktterin tandaniz");
  buf = Console.ReadLine();
  k = Convert.ToInt32(buf);
```

```
  switch (k)
  {
   case 1: sozd(); break;
   case 2: zapf(); break;
   case 3: printm(); break;
   case 4: vvod(); break;
   case 5: printm(); break;
   default: break;
  }
  }
 }
 }
}
```

Бағдарлама жұмысы:

```
1 - Massivti kyry
2 - Massivti failga shazy
3 - Massivti shigary
4 - faildan massivti oky
5 - Massivti shigary
6 - Shigy
Menu punktterin tandaniz
1
Toptagi stydentter sani <= 20 !!
3
1-stydenttin atin engiziniz
Бедаш Дмитрий
1-stydenttin tobin engiziniz
09-ВТ-1
1-stydenttin balin engiziniz
90
1-stydenttin shasin engiziniz
18
2-stydenttin atin engiziniz
Жумашев Ержегит
2-stydenttin tobin engiziniz
09-ВТ-1
2-stydenttin balin engiziniz
85
2-stydenttin shasin engiziniz
19
3-stydenttin atin engiziniz
Аскарова Дина
3-stydenttin tobin engiziniz
09-ИС-1
3-stydenttin balin engiziniz
89
3-stydenttin shasin engiziniz
18
1 - Massivti kyry
2 - Massivti failga shazy
3 - Massivti shigary
4 - faildan massivti oky
```

```
5 - Massivti shigary
6 - Shigy
Menu punktterin tandaniz
2
1 - Massivti kyry
2 - Massivti failga shazy
3 - Massivti shigary
4 - faildan massivti oky
5 - Massivti shigary
6 - Shigy
Menu punktterin tandaniz
3
 Ati,  Tobi, Ball, Shasi
 Бедаш Дмитрий, 09-ВТ-1, 90,  18
 Жумашев Ержегит, 09-ВТ-1, 85,  19
 Аскарова Дина, 09-ИС-1, 89,  18
1 - Massivti kyry
2 - Massivti failga shazy
3 - Massivti shigary
4 - faildan massivti oky
5 - Massivti shigary
6 - Shigy
Menu punktterin tandaniz
6
```

Бағдарламаны екінші рет іске қосқанда:

```
1 - Massivti kyry
2 - Massivti failga shazy
3 - Massivti shigary
4 - faildan massivti oky
5 - Massivti shigary
6 - Shigy
Menu punktterin tandaniz
4
1 - Massivti kyry
2 - Massivti failga shazy
3 - Massivti shigary
4 - faildan massivti oky
5 - Massivti shigary
6 - Shigy
Menu punktterin tandaniz
5
Ati,  Tobi, Ball, Shasi
Бедаш Дмитрий, 09-ВТ-1, 90,  18
Жумашев Ержегит, 09-ВТ-1, 85,  19
Аскарова Дина, 09-ИС-1, 89,  18
1 - Massivti kyry
2 - Massivti failga shazy
3 - Massivti shigary
4 - faildan massivti oky
5 - Massivti shigary
6 - Shigy
```

```
Menu punktterin tandaniz
```

Бағдарламада есептің шартына сәйкес төрт әдіс қолданылған. (студенттер массивін шығару әдісі екі рет қолданылған).

10.6 Өзін-өзі тексеру сұрақтары

1 C# тілінде құрылымды жариялағанда қандай қызметтік сөз қолданылады?

2 struct типіндегі айнымалылар әдетте қалай аталады?

3 Жазбада қандай айнымалылар біріктіріле алады?

4 Жазбаның ішінде айнымалылар қалай аталады?

5 Құрылымның қандай мәндік және сілтемелік типі бар?

6 Жазба өрісінің атауы қалай анықталады?

7 C# тілінде тізімдік типті жариялағанда қандай қызметтік сөз қолданылады?

8 Тізімдік тип нені біріктіреді?

9 C# тілінде құрылымдармен, кластармен жұмыс істегенде сериализацияны қалай түсінесіз?

10 C# тілінде құрылымдармен, кластармен жұмыс істегенде десериализацияны қалай түсінесіз?

11 СТЕКТЕР, КЕЗЕКТЕР ЖӘНЕ ТІЗІМДЕР

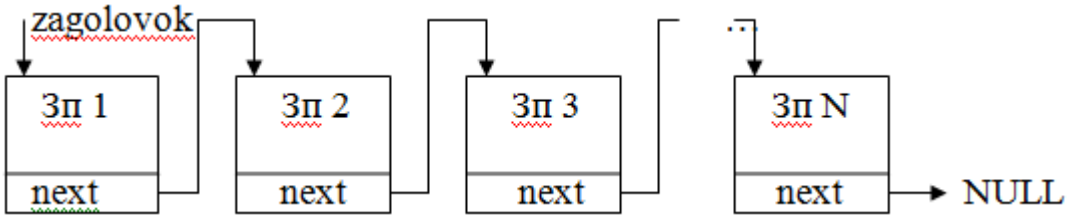
11.1 Тізімдік құрылымдар

Кез келген ақпараттық анықтама жүйесімен жұмыс жасау деректер массивін қолдануды талап етеді. Әдетте деректер магнитті дискте файлдар түрінде сақталады. Бірақ егер файл деректерін үнемі қатынасу және түрлі іздестіру операцияларын орындау керек болса, онда осы деректер файлдарын компьютердің жадына орналастыру дұрыIrimshikақ болады, өйткені магнитті дискте орналасқан файлда деректерді іздестіру уақыты компьютер жадысында орналасқан файлда іздестіру уақытынан мың есе көп.

Компьютердің жадына үлкен анықтамалық ақпарат файлдарын орналастыру мәселесі туындайды.

Деректерді файлдан компьютер жадысына, магнитті дискке енгізудің көптеген алгоритмдері бар. Мысалы, деректер файлын түрлі кесте, матрица түрінде көрсетуге болады.

Компьютердің жадына деректерді орналастырудың бір нұсқасы - түрлі тізімдік құрылымдар. Компьютердің жадына жазбалар тізімін ұйымдастыруда әрбір элемент (жазба) құрылымына қосымша өріс, яғни тізімнің келесі элементіне нұсқағыш (әдетте осы өріс next деп аталады) қосылады. Осындай құрылымның негізінде тізім элементтері біртұтас болып біріктіріледі. Құрылымда тізімге кіріс нүктесі – тізімнің басы немесе тақырыбы және тізімнің аяқталу сипаты (тізімдегі соңғы элементтің next өрісінің мәні NULL тең) немесе тізімдегі элементтер санын анықтайтын айнымалы нөлге тең болуы керек.



11.1-сурет – Қарапайым тізімнің құрылымы

Тізімдік құрылымдар әдетте мына әрекеттерді (алгоритмдерді) орындайды:

– жазбалар файлынан тізімді құруды;
– тізімді қарап шығуды;
– тізімге жаңа элементті қосуды (тізімнің басына, ортасына, соңына);
– тізімнен элементті жою (тізімнің басынан, ортасынан, соңынан);
– тізімді файлға жазу;
– бірнеше тізімдерді біріктіру;

– бір тізімді екіге немесе бірнеше бөліктерге бөлу;

– тізімдегі элементтерді сұрыптау, т.б.

Қарапайым тізімнен элементті өшіру мен қосу жолдары бойынша барлық тізімдік құрылымдарды төрт топқа бөлуге болады:

– стектер;

– кезектер;

– дектер;

– қарапайым тізімдер.

Стек – қарапайым тізім, онда элементті өшіру мен қосу бір жақ бөлігіндегі тізімнің соңында орындалады. Әдетте стекте орындалатын жұмыс принципі «Бірінші кірген соңында шығады, соңында кірген бірінші болып шығады» сөйлемімен түсіндіріледі.

Кезек – қарапайым тізім, онда элементті қосу тізімнің бір жақ бөлігінде (кезек соңы), ал өшіру немесе элементпен жұмыс істеу тізімнің екінші бөлігінде (кезектің басы) орындалады.

Дек – қарапайым тізім, онда элементті өшіру мен қосуды тізімнің кез келген соңында (ұшында) орындауға болады.

Қарапайым тізімде элементті өшіру мен қосу тізімнің кез келген жерінде орындауға болады, сонымен қатар тізімнің ортасында.

11.2 Стек типіндегі құрылыммен жұмысты ұйымдастыру

Стек типіндегі құрылым әртүрлі есептеу үдерісінде кең қолданылады. Компьютер процессоры үзуді өңдеу барысында стекті қолданады. Алгоритмдердің көпшілігі алгоритмді алдыңғы қалпына «қайтаруды» орындау үшін стекттерді қолданады, мысалы, лабиринтті, графты немесе бұтақтарды жүріп өткен кезде. Көптеген көліктер туралы есептерді шешу алгоритмдерінде оңтайлы маршруттарды табу үшін стек пайдаланылады, т.б.

Әлбетте, «Алгоритмизация және бағдарламалау» пәнінде стек сияқты құрылымды қарастырмай өте алмаймыз, C# тілінде оған әдістер жиыны бар арнайы класс (`Stack`) бөлінген.

Стектермен жұмысты орындау барысында негізгі алгоритмдер: стекке элементті қосу алгоритмі, стектен элементті жою алгоритмі, стек мазмұнын қарау.

Стек типіндегі құрылымға қол жеткізу әдетте стек төбесі деп аталатын оның тек бір жақ шетінде ғана мүмкін.

Стек класы динамикалық коллекция – бір типтегі деректердің бірігуі түрінде бола алады. Стекті кеңейткен кезде («толтырылған» стекке элементті қосу барысында) стек сыйымдылығы динамикалық түрде екі есе үлкейеді.

`Stack` класында келесі үш конструктор анықталған:

Public Stack();

Public Stack(int capacity);

Public Stack(ICollection n);

Бірінші конструктор 10 элементтен тұратын «бос» стекті құрады. Екінші конструктор сыйымдылығы `capacity` болатын «бос» стекті құрады. Үшінші конструктор n элементті стекті құрады.

11.1-кестеде `Stack` класының негізгі әдістері ұсынылған.

11.1-кесте – `Stack` класының негізгі әдістері

Әдіс	Сипатама
public virtual bool Contains(object v)	True мәнін қайтарылады, егер v объектісі стекте болса, әйтпесе false мәні қайтарылады.
public virtual void Clear()	Стекті тазартады (Count қасиеті – стек элементтерінің саны нөлге теңестіріледі).
public virtual object Peek()	Стек төбесінің элементін қайтарады, бірақ оны жоймайды.
public virtual object Pop()	Стек төбесінің элементін қайтарады және оны жояды.
public virtual void Push(object v)	v элементін стекке қосады - стек төбесіне

Стек ішіндегісін көру үшін `in` операциясы жиі қолданылады, мысалы:

```
Console.Write("Стек ішіндегісі = ");
foreach (int i in vst)
Console.Write(i + " ");
Console.WriteLine();
```

Кодтың көрсетілген жолдары vst төбесінен және бүтін сандардан тұратын стекті қарап шығуға мүмкінді береді.

Стектермен жұмыс жасайтын алгоритмнің мысалы ретінде келесі есепті қарастырайық:

11.1-есебі. 5 бүтін саннан тұратын стек үшін санды қосу және шығару ықтималдығы кездейсоқ құрылады. Бұл орайда санды стекке қосу ықтималдығы 70% тең, ал стектен шығару 30% тең. Сандар кездейсоқ құрылады және минус 50 мен 50 аралығында болады.

Стекті толтыру барасында және стекке жаңа санды қосу кезінде, шығару ықтималдықтары қарама-қарсы мәндерге өзгереді. Бағдарлама жұмысының соңы санды «бос» стектен шығару мүмкіндігімен анықталады.

Бағдарламаның коды:

```
using System;
using System.Collections;
namespace ConsoleApplication1
{
```

```
class Program
{
static void vkl(Stack vst, int n)
{
 vst.Push(n);
 Console.Write("Stekke element zhazildi - {0}", n);
 Console.Write("Stekte zhazildi = ");
 foreach (int i in vst)
 Console.Write(i + " ");
 Console.WriteLine();
}
static void iskl(Stack vst)
{
 if (vst == null) Console.WriteLine("Stek bos!");
 else
 {
 int n = (int)vst.Pop();
 Console.Write("Stekten element zhoildi {0}", n);
 Console.Write("Stekte zhazildi = ");
 foreach (int i in vst)
  Console.Write(i + " ");
 Console.WriteLine();
 }
}
static void Main()
{
 Stack vstek = new Stack();
 int i, k, n;
 Random rnd = new Random();
 i = 0;
 while (i < 5)
 {
 k = rnd.Next() % 101;
 if (k <= 70)
 {
  i++;
  n = rnd.Next() % 101 - 50;
  vkl(vstek, n);
 }
 else
  if (i > 0)
  {
  i--;
  iskl(vstek);
  }
 }
 Console.WriteLine("Stek tolik!");
 while (i > 0)
 {
 k = rnd.Next() % 101;
 if (k <= 30)
 {
```

```
   i++;
   n = rnd.Next() % 101 - 50;
   vkl(vstek, n);
  }
  else
   if (i > 0)
   {
   i--;
   iskl(vstek);
   }
  }
  Console.WriteLine("Stek bos!");
  Console.WriteLine("Enter pernesin basiniz");
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
     Stekke element zhazildi - -6Stekte zhazildi = -6
     Stekke element zhazildi - 12Stekte zhazildi = 12 -6
     Stekke element zhazildi - 11Stekte zhazildi = 11 12 -6
     Stekten element zhazildi 11Stekte zhazildi = 12 -6
     Stekke element zhazildi - -8Stekte zhazildi = -8 12 -6
     Stekke element zhazildi - -24Stekte zhazildi = -24 -8 12 -
6
     Stekke element zhazildi - 14Stekte zhazildi = 14 -24 -8 12
-6
     Stek tolik!
     Stekten element zhoildi 14Stekte zhazildi = -24 -8 12 -6
     Stekten element zhoildi -24Stekte zhazildi = -8 12 -6
     Stekten element zhoildi -8Stekte zhazildi = 12 -6
     Stekten element zhoildi 12Stekte zhazildi = -6
     Stekke element zhazildi - -27Stekte zhazildi = -27 -6
     Stekke element zhazildi - -13Stekte zhazildi = -13 -27 -6
     Stekten element zhoildi -13Stekte zhazildi = -27 -6
     Stekten element zhoildi -27Stekte zhazildi = -6
     Stekten element zhoildi -6Stekte zhazildi =
     Stek bos!
     Enter pernesin basiniz
```

Бағдарламада стекке элементті қосу және одан элементті шығару операцияларының әрқайсысына түсініктеме қосылған.

11.3 Кезек типіндегі құрылыммен жұмыстарды ұйымдастыру

Кезек типіндегі құрылымдар «қайтаруы бар іздеу» алгоритмдерінде кең қолданылады. Графтардың, бұтақтардың төбелерінен өту алгоритмдері «ішіне қарай жүру» алгоритімін жүзеге aIrimshiky кезінде – стектерді немесе «көлденеңінен жүру» алгоритмін жүзеге aIrimshiky кезінде –

кезектерді қолданады. Жаппай қызмет көрсету теориясының көптеген есептері оңтайлы нұсқаларды табу үшін кезектерді қолданады.

Кезек типіндегі құрылымның екі тақырыбы болады: ол арқылы кезек элементімен жұмысты орындайтын (элементті жою) кезектің басы және кезекке жаңа элементтерді қосу үшін қолданылатын кезектің соңы. Кезектермен жұмыс жасау үшін C# тілінде арнайы класс (`Queue`) бөлінген. Бұл кластың әдістер жиыны бар

`Queue` класында келесі үш конструктор бар:

Public Queue ();

Public Queue (int capacity);

Public Queue (ICollection c);

Бірінші конструктор 10 элементке «бос» кезекті құрайды. Екінші конструктор сиымдылығы `capacity` элементіне тең «бос» кезекті құрайды. Үшінші конструктор ICollection элементтері арқылы жұмыс жасайтын, n элементке арналған жаңа кезекті құрайды.

`Queue` класының негізгі әдістері 11.2-кестесінде ұсынылған.

11.2-кесте – `Queue` класының негізгі әдістері.

Әдіс	Сипатама
public virtual bool Contains(object v)	True мәнін қайтарылады, егер v объектісі кезекте болса, әйтпесе false мәні қайтарылады.
public virtual void Clear()	Кезекті тазартады (Count қасиеті – кезек элементтерінің саны нөлге теңестіріледі).
public virtual object Peek()	Объектті кезектің басынан қайтарады, бірақ оны жоймайды.
public virtual object Dequeue()	Объектті кезектің басынан қайтарады және оны жояды.
public virtual void Enqueue(object v)	v объектісін кезек соңына қосады.

Кезек әдістерінің жұмысын түсіндіретін мысал келесі есепте қарастырылады:

11.2-есебі. Дүкенге бір күнде 250 адам келеді. Сатып алушыларға қызмет көрсету ретінде мыналар сатылады: нан – ықтималдығы 25%; ірімшік – ықтималдығы 20%; печенье– ықтималдығы 20%; сыра – ықтималдығы 10% және балмұздақ – ықтималдығы 25%. Әрбір сатып алушы тек бір өнімді ғана сатып алады деп есептейік. Барлық оқиғалардың ықтималдықтары 0 – 100 аралығына тең. Сатып алушылар кезегін ұйымдастыру керек, ал оларға қызмет көрсету кезінде әртүрдегі сатылған өнімдердің санын және жиі сатып алынатын өнімді анықтап басып шығару керек.

Есептің шарты бойынша барлық оқиғалардың ықтималдықтары бірдей, сондықтан, егер 0 мен 100 аралығында кездейсоқ санды құрсақ,

онда оқиғаның орындалу ықтималдығы берілген диапозон арқылы анықталады. Мысалы, нанды сатып алу ықтималдылығы 1-ден 25-ке дейінгі аралықта анықталады, ал ірімшікті – 26 мен 45, печеньені – 46 мен 65, сыраны – 66 мен 75 және балмұздақты –76 мен 100 аралықтарында анықтайды.

Өнімнің әрбір түрін 0 мен 4-ке дейінгі бүтін сандармен анықтауға болады. Сонымен, сатып алушылар кезегін олар сатып алатын өнімдер кезегімен алмастыруға болады, олар бүтін санмен белгіленеді.

Бағдарлама коды:

```
using System;
using System.Collections;
using System.Text;
namespace ConsoleApplication1
{
 class Program
 {
 public static int n;
 static void vkl(Queue ocer, int n)
 {
  ocer.Enqueue(n);
 }
 static void iskl(Queue ocer)
 {
  n = (int)ocer.Dequeue();
 }
 public static void Main()
 {
  Queue zagol = new Queue();
  int[] masi = new int[5];
  char[] masc = new char[5] { 'N', 'I', 'P', 'S', 'B' };
  string[] mass = new string[5] { "Nan", "Irimshik",
"Печенье", "Sira", "Balmuzdak" };
  int i, j, p, k;
  Random rnd = new Random();

  for (i = 0; i < 4; i++) masi[i] = 0;
  for (i = 0; i < 250; i++)
  {
  p = rnd.Next() % 100 + 1;
  if (p <= 25) vkl(zagol, 0);
  if (25 < p && p <= 45) vkl(zagol, 1);
  if (45 < p && p <= 65) vkl(zagol, 2);
  if (65 < p && p <= 75) vkl(zagol, 3);
  if (p > 75) vkl(zagol, 4);
  };
  for (i = 0; i < 250; i++)
  {
  iskl(zagol);
  masi[n]++;
  Console.Write(masc[n]);
  if ((i + 1) % 25 == 0) Console.WriteLine();
```

```
  };
  k = 0; j = 0;
  for (i = 0; i < 5; i++)
  {
  Console.WriteLine(" {0} = {1}", mass[i], masi[i]);
  if (k < masi[i]) { k = masi[i]; j = i; }
  }
  Console.WriteLine("En kop satilgan {0} = {1}", mass[j],
masi[j]);
  Console.WriteLine("Enter pernesin basiniz");
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
IBIINSBIINNINPINPPNPBSIIP
INNPBPSNNNNNPBSPINSBPSNPP
BBBNNNNIBNNNBINISPBSPPNNB
PPNPNPBSBNNININPBSINIPIPB
IPBBPBBBNBNBINPNPBNPNPBSS
NBBPNBNPSINNBBSNBPPPPPNBB
BPNPBSIPPINPBPBBBPIBNNPBI
PPISPSIPPNBIIIPNNBBSNNBSP
PBSINIPSSPNIIPSBISPINNSSN
NSNIIPNPNBBNIININSIPINSNI
 Nan = 68
 Irimshik = 45
 Pechenie= 58
 Sira = 29
 Balmuzdak = 50
En kop satilgan Nan = 68
Enter pernesin basiniz
```

11.4 Тізім типіндегі құрылыммен жұмыстарды ұйымдастыру

Тізім типіндегі әр түрлі, құрылымдар болады. C# тілінде олар үшін арнайы `List<T>`, `ArrayList`, `LinkedList<T>` , т.б. кластар дайындалған. Объекттердің динамикалық массивтері болып келетін `List<T>` және `ArrayList` кластары жиі қолданылады. (11.3, 11.4 кестелері).

11.3-кесте –ArrayList класының негізгі қасиеттері

Кластың қасиеті	Сипаттамасы
public virtual int Capacity {get;}	Осы қасиет оқу үшін ғана арналған, коллекция ағымдағы сыйымдылығын сақтайды.
public virtual int Count {get;};	Осы қасиет оқу үшін ғана арналған, коллекцияның ағымдағы ұзындығын сақтайды.

11.4-кесте – ArrayList класының негізгі әдістері

Кластың әдістері	Сипаттамасы
public static ArrayList (IList: List);	List коллекциясы негізінде ArrayList объектісі құрылады
public virtual int Add(Oblect Value);	Тізім соңына жаңа объектті қосады, оның индексін қайтарады.
public virtual viod AddRange (ICollection coll);	Тізім соңына бірнеше объекттерді қосады
public virtual int BinsrySearch(Qbject Value);	Сұрыпталған тізімде Value объектісін табады және оның индексін қайтарады. Егер табылмаса, теріс санды қайтарады.
public virtual bool Containse (Object Value);	Егер коллекцияда Value элементі бар болса, true қайтарылады.
public static ArrayList FixedSize(ArrayList AL);	Әдіс элементтерін өзгертуге болатын, бірақ қосуға немесе жоюға болмайтын объекттерді қайтарады,
public virtual IEnumerator GetEnumerator();	Объект үшін итераторды қайтарады.
public virtual ArrayList GetRange(int Indx, int Count);	Элементтер диапазонын қайтарады.
public virtual int IndexOf(Object Value);	Value мәні бар элемент индексін қайтарады.
public virtual void Insert (int Indx, Object Value);	Коллекцияның керекті Indx орнына Value элементін қосады.
public virtual void InsertRange(int Indx, IColliction col);	Элементтер диапазонын қосады
public virtual bool IsFixedSize{get;}	Егер объекттің белгіленген өлшемі болса, true қайтарылады.
public virtual bool IsReadOnly{get;}	Объектте оқуға ғана арналған элементтер болса , true қайтарылады.
public virtual int LastIndexOf(Object Value);	Коллекцияда соңғы рет кездесетін Valiue мәнінің индексін қайтарады.
public static ArrayList ReadOnly(ArrayList AL);	Коллекция элементтеріне оқуға ғана рұқсат беретін режімді орнатады.
public virtual void Remove (Object Value);	Коллекциядан мәні Value болатын бірінші элементті жояды.
public virtual void RemoveAt (int Indx);	Коллекциядан индексі Indx болатын элементті жояды.
public virtual void RemoveRange(int Indx, int count);	Коллекциядан индексі Indx болатын элементтен бастап count элементтерін жояды.

11.4-кестесінің жалғасы

Кластың әдістері	Сипаттамасы
public virtual Object this[int Indx]{set; get;}	Осы қасиет элементтерге индексі бойынша қатынасуға мүмкіндік береді
public static ArrayList Repeat(Object Value, int count);	Value элементі count рет қайталанатын коллекцияны қайтарады.
public virtual void Reverse();	Элементтердің ретін кері бағытқа өзгертеді.
public virtual void SetRange (int Indx, ICollection col);	Indx индексінен бастап col коллекциясын қосады.
public virtual void Sort();	Коллекцияны сұрыптайды.
public virtual void TrimToSize();	Коллекцияға элементтерінің санына тең болатын сыйымдылықты орнатады.
public virtual void Clear();	Коллекцияның барлық элементтерін жояды.

`ArrayList` коллекциясының алғашқы сыйымдылығы 16-ты элементке тең. Коллекцияны кеңейту кезінде оның сыйымдылығы екі есе үлкейеді және 32, 64, 128, т.б. құрайды. `TrimToSize()` әдісі қолданылмайтын элементтерді өшіреді.

Бағдарлама кодының мысалы:

```
using System;
using System.Collections;
using System.Text;
namespace ConsoleApplication1
{
 struct Avto
 {
 public String Marka; //автомобильдің үлгісі
 public String Voditel; // жүргізушінің аты-жөні
 public int Stoim; // автомобильдің бағасы
 public int God;  // шығарылған жылы
 }
 class Program
 {
 public static int kol = 0;
 public static Avto avto = new Avto();
 static ArrayList Garaj = new ArrayList();
 public static void addcpi()
 {
  int b;
  int kol;
  string buf;
  Console.WriteLine("Avtomobil sani");
  buf = Console.ReadLine();
  kol = Convert.ToInt32(buf);
  for (int i = 0; i < kol; i++)
  {
```

```
  Console.WriteLine("{0}-Avtomobildin ylgisin engiziniz", i +
1);
  buf = Console.ReadLine();
  avto.Marka = buf;
  Console.WriteLine("{0}-Avtomobil zhyrgizyshinin ati-zhoni",
i + 1);
  buf = Console.ReadLine();
  avto.Voditel = buf;
  Console.WriteLine("{0}-Avtomobildin bagasi", i + 1);
  buf = Console.ReadLine();
  b = Convert.ToInt32(buf);
  avto.Stoim = b;
  Console.WriteLine("{0}-Avtomobildin shikkan zhili", i + 1);
  buf = Console.ReadLine();
  b = Convert.ToInt32(buf);
  avto.God = b;
  Garaj.Add(avto);
  }
 }
 public static void printcpi()
 {
  Console.WriteLine("{0,20}, {1, 20}, {2, 15}, {3, 15}",
  "Ylgisi", "Zhyrgizyshi ati-zhoni", "Avtom. bagasi", "Avtom.
shikkan zhili");
  foreach (Avto T in Garaj)
  Console.WriteLine("{0,20}, {1, 20}, {2, 15}, {3, 15}",
   T.Marka, T.Voditel, T.Stoim, T.God);
 }
 public static void ydalelemcpi()
 {
  int n;
  string buf;
  Console.WriteLine("Zhoilatin zhazba nomeri?");
  buf = Console.ReadLine();
  n = Convert.ToInt32(buf);
  // элементті номері бойынша жоямыз
  Garaj.RemoveAt(n);
 }
 static void Main()
 {
  // тізімді құрамыз және тоолтырамыз
  addcpi();
  // тізімді шығарамыз
  printcpi();
  // элементті номері бойынша жоямыз
  ydalelemcpi();
  // тізімді шығарамыз
  printcpi();
  Console.ReadLine();
 }
 }
}
```

```
Бағдарлама жұмысы:
Avtomobil sani
3
1-Avtomobildin ylgisin engiziniz
Лада
1-Avtomobil zhyrgizyshinin ati-zhoni
Петров
1-Avtomobildin bagasi
2000
1-Avtomobildin shikkan zhili
2007
2-Avtomobildin ylgisin engiziniz
Фиат
2-Avtomobil zhyrgizyshinin ati-zhoni
Иванов
2-Avtomobildin bagasi
3000
2-Avtomobildin shikkan zhili
2008
3-Avtomobildin ylgisin engiziniz
БМВ
3-Avtomobil zhyrgizyshinin ati-zhoni
Сидоров
3-Avtomobildin bagasi
4000
3-Avtomobildin shikkan zhili
2006
 Ylgisi, Zhyrgizyshi ati-zhoni, Avtom. bagasi, Avtom. shikkan zhili
 Лада, Петров, 2000, 2007
 Фиат, Иванов, 3000, 2008
 БМВ, Сидоров, 4000, 2006
Zhoilatin zhazba nomeri?
1
 Ylgisi, Zhyrgizyshi ati-zhoni, Avtom. bagasi, Avtom. shikkan zhili
 Лада, Петров, 2000, 2007
 БМВ, Сидоров, 4000, 2006
```

11.5 Кеңейтілген іздеу мүмкіндіктері бар тізімдер

C# тілінде бір бағыттық тізімдердің іздестіру мүмкіндіктерін кеңейту үшін қос бағыттық тізімдер қосылды, мысалы, `LinkedList<T>`.

Қос бағыттық тізімдер түйіндердің тізбектерінен тұрады, онда келесі (әдете `Next` өрісі) және алдыңғы (әдете `Old` өрісі) түйіндерге сілтемелер болады.

Кеңейтілген іздестіру мүмкіндіктері тізім бойынша кез келген бағытта көшу мүмкіндігімен байланысты. Қос бағыттық тізімдердің қосымша артықшылығы - тізім элементерінің қосу, алу операцияларының қарапайымдылығы. Осы тәртіп араларында операция орындалатын тізім элементтерінің бірнеше сілтемелерін жаңартады.

Қос бағыттық тізімнің келесі модификациясы циклдік тізімдерді құру болып табылады. Циклдік тізімде соңғы элемент `Next` нұсқағышы арқылы бірінші элементпен, ал тізімнің бірінші элементі `Old` нұсқағышы көмегімен соңғы элементпен байланыстырылады.

Циклдік тізімде тақырып кез келген элементте орналаса алады. Мысалы, іздеу операциясын орындағаннан кейін біз тақырыпты табылған элементке орната аламыз. Келесі іздеу операциясы кезінде тақырып келесі табылған элементке көшірілуі мүмкін.

Тізімде екі бағытта орнын ауыстыру мүмкіндігін ескере отырып іздеу алгоритмін жақсартуға болады.

Алайда стандартты іздеу алгоритмдерінің көпшілігі тізімдерді өңдеуге бағытталған. Бұл орайда тізім тақырыбы белгілі жерде орналасуы керек. Осы мақсатта тақырыпты өрістерінде мағыналық мәні жоқ элементке орналастыру ұсынылды, ал элемент циклдік тізімде бастапқы нүкте ретінде керек.

Тізімді осылай ұйымдастыру тақырыбы бөлінген тізім деген атқа ие болды.

Екілік іздеу алгоритмін орындау үшін бұтақтар тәрізді құрылым қолданылады, элементтерінде екі нұсқағыштары бар (оң және сол).

11.6 Өзін-өзі тексеру сұрақтары

1 Элементтерді қосу мен жою тізімнің бір жақ ұшында орындалады. Осы қарапайым тізім қалай аталады?

2 Стектің басы қалай аталады?

3 Public Stack(ICollection n); жазбасы нені білдіреді?

4 Бағдарламаның келесі үзіндісі нені орындайды

```
 int n = (int) vst.Pop();,
```

мұнда `vst` – стек төбесі?

5 Бағдарламаның келесі үзіндісі нені орындайды:

```
i = 0;
while (i < 5)
  {
   i++;
   n = rnd.Next() % 101 - 50;
   vstek.Push(n);
```

```
  } ,
```

мұнда `vstek` – стек төбесі?

6 элементтерді қосу тізімнің бір жақ ұшында, ал элементті жою екінші жақ ұшында орындалатын қарапайым тізім қалай аталады?

7 `Public Queue()` жазбасы нені білдіреді?

8 `object queue.Peek()` жазбасы нені білдіреді?

9 Бағдарламаның келесі үзіндісі нені орындайды:

```
foreach (int i in ocer)
 Console.Write(i + " ");
 Console.WriteLine();,
```

мұнда `ocer` – кезектің тақырыбы?

10 Бағдарламаның келесі үзіндісі нені орындайды:

```
ocer.Clear();,
```

мұнда `ocer` – кезектің тақырыбы?

12 ГРАФТАР

12.1 Графтар теориясының негізгі анықтамалары

Графтар теориясын білуді қажет ететін белгілі бір есептердің класы болады, мысалы, көлік есептері, желілерде ағындарды тиімдеу есептері, иерархиялық бұтақтар тәрізді құрылымдарда іздестестіру есептері, т.б.

Бұтақтар мен графтар бір біріне ұқсас болып келеді. Мысалы, бағдарлама саласындағы белгілі теорияшыл Д.Кнут графтарды бұтақтар бөлімінде түсіндіріп кеткен. Паскаль тілін жасаушы Н.Вирт “Алгоритмы и структуры данных” кітабында графты мүлдем қарастырмайды, бірақ күрделілігі әртүрлі бұтақтарды қарастырады. Ресей авторлары күрделі құрылымдарды сипаттағанда біріншіден графтарды қарастырады (12.1-суретті қара).

Сонымен қатар граф қос (V,E) жиынтық ретінде анықталады.

мұнда V – төбелердің шекті жиыны;

E – төбелерді байланыстыратын қабырғалардың шекті жиыны.

S=<U,W> жазбасы мынаны сипаттайды: S қабырғасы U және W төбелерін байланыстырады.

Бір қабырғамен байланысатын төбелер сыбайлас төбелер деп аталады.

S қабырғасы және U төбесі (және S қабырғасы мен W төбесі) түйісті (инцидентті) деп аталады.

Бір төбеге түйісті (инцидентті) қабырғалар сыбайлас қабырғалар деп аталады.

Төбенің дәрежесі оған түйісті (инцидентті) қабырғалардың санына тең.

U және K төбелерін байланыстыратын жол – $W_0,W_1, \ldots, W_n$ ($n>0$), төбелерінің тізбегі, мұнда $W_0=U$, $W_n=K$, кез келген i ($0\le i\le n-1$) үшін W_i және W_{i+1} төбелері қабырғалармен байланысқан.

$W_0, \ldots, W_n$ жолының ұзындығы оның қабырғаларының санына – n тең.

Егер жолдың барлық төбелері әртүрлі болса, онда ондай жол қарапайым жол деп аталады.

Егер $W_0=W_n$ болса, онда ондай жол тұйық жол деп аталады.

Барлық қабырғалары әр түрлі тұйық жолды цикл деп атайды.

Барлық төбелері әр түрлі тұйық жолды контур деп атайды.

Екі төбе арасындағы қашықтық – осы төбелерді байланыстыратын ең қысқа жолдың ұзындығы.

Егер кез келген төбелер жұбын байланыстыратын жол бар болса, онда ондай граф байланысты граф деп аталады.

Егер кез келген екі төбелер қабырғалар шекті байланыстырылса, онда ондай граф толық граф деп аталады.

Бағытталған граф немесе орграфтың қарапайым графтан айырмашылығы - төбелер, қабырғалармен емес, доғалармен (доға дегеніміз - бағыты берілген қабырға) байланысуы.

Орграф бойымен орын ауыстыруды тек доға бағытымен ғана орындауға болады.

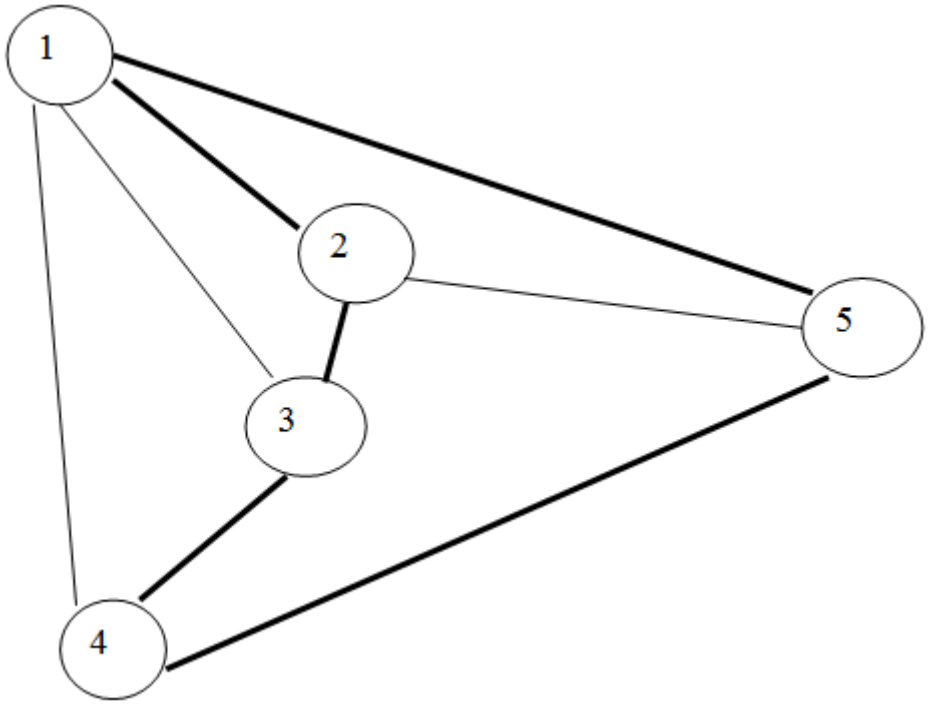
Графтың кез келген төбесінен белгілі төбеге дейін жол бар болса, онда осы төбе орграфтың құйылысы деп аталады.

Бір төбеден графтың кез келген төбесіне дейін жол бар болса, онда ондай төбені орграфтың көзі деп атайды.

U төбесінің графтың қалған төбелеріне дейінгі ең үлкен қашықтық U төбесінің эксцентриситеті деп аталады.

Төбелердің арасындағы ең үлкен эксцентриситетті графтың диаметрі деп атайды. Төбелердің арасындағы ең кіші эксцентриситетті графтың радиусы деп атайды.

Эксцентриситеті графтың радиусына тең төбені графтың медианасы немесе оның центрлік төбесі деп атайды.



12.1-суреті – Графтың көрінісі

Егер барлық төбелер (бірақ барлық қабырғалардың кіруі міндетті емес) циклге кіретін болса және осы цикл графта бар болатын болса, онда осы граф гамильтонды граф деп аталады.

Егер графтың барлық қабырғалары циклге бір рет қана кіретін болса және осы цикл графта бар болатын болса, онда осы граф Эйлер графы деп аталады.

Графтың барлық қабырғаларының бойымен тек бір рет қана жүріп өтетін жолды Эйлер жолы деп атайды.

Графты сурет түріндегі қабылданған форма бойынша көрсетуге болады, онда төбелерге шеңберлер, ал қабырғаларға осы шеңберлерді байланыстыратын сызықтар сәйкес келеді.

12.1-суретінде Гамильт циклі жалпақ сызықпен белгіленген.

Бұтақтар тәрізді құрылымның көптеген анықтамалары бар.

Байланысқан граф арқылы бұтақтардың екі анықтамасын келтірейік.

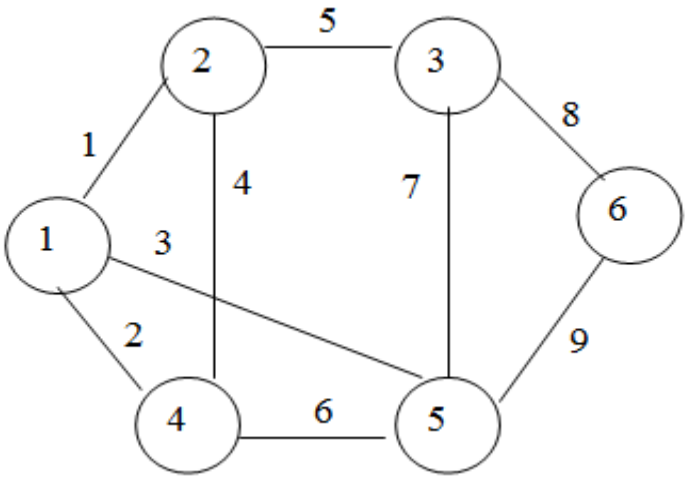
Бұтақтар дегеніміз – доғаларының саны төбелерінің санынан бірге кем болатын байланыстырылған граф.

Бұтақтар дегеніміз – циклі жоқ байланыстырылған граф.

«Графтық» есептерді шығару барысында алгоритмнің күрделі болуы оның «машиналық» көрсетіміне байланысты, яғни «графтық» есептің типіне байланысты ЭЕМ жадысында графты көрсетудің белгілі бір формасы қолданылады.

12.2 Компьютер жадысында графты көретудің тәсілдері

12.2.1 Графты түйістілік матрицасы (матрица инцидентностей) түрінде көрсету. 12.2-суретте көрсетілген қарапайым бағытталмаған граф берілсін.



12.2-сурет – Қарапайым бағытталмаған граф

Графтар теориясы бойынша ЭЕМ жадысында графты көрсетудің классикалық әдісі ретінде түйістілік матрицасы қолданылады. Осындай матрицада жолдардың саны төбелер санына, ал бағаналар саны граф қабырғаларының санына сәйкес болады. Егер төбе мен қабырға түйістілі болса, онда бағана мен жол қиылысында 1 жазылады, әйтпесе 0 жазылады.

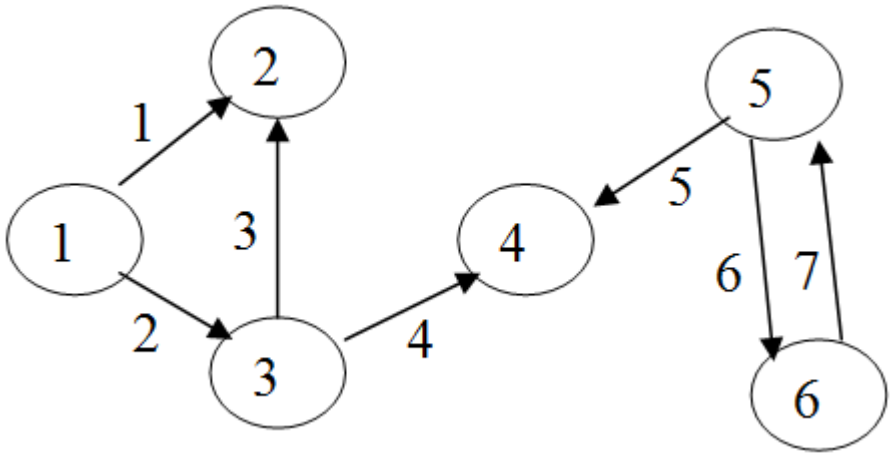
12.2-суретте көрсетілген қарапайым бағытталмаған граф үшін 12.1-кестесінде көрсетілген түйістілік матрицасын құруға болады.

12.1-кесте – Түйістілік матрицасы

	1	2	3	4	5	6	7	8	9
1	1	1	1	0	0	0	0	0	0
2	1	0	0	1	1	0	0	0	0
3	0	0	0	0	1	0	1	1	0
4	0	1	0	1	0	1	0	0	0
5	0	0	1	0	0	1	1	0	1
6	0	0	0	0	0	0	0	1	1

Егер доға сәйкес төбеден «шықса», онда бағытталған графта түйістілік матрицасындағы доға бағыты 1-ге тең, егер доға сәйкес төбеге «кіретін» болса, онда минус 1-ге тең.

12.3-суретте көрсетілген бағытталған граф үшін түйістілік матрицасын қарастырайық



12.3-сурет – Бағытталған граф

Бағытталған графтың түйістілік матрицасы 12.2-кестеде көрсетілген.

12.2-кесте – Бағытталған графтың түйістілік матрицасы

	1	2	3	4	5	6	7
1	1	1	0	0	0	0	0
2	-1	0	-1	0	0	0	0
3	0	-1	1	1	0	0	0
4	0	0	0	-1	-1	0	0
5	0	0	0	0	1	1	-1
6	0	0	0	0	0	-1	1

Айта кететін жәйт, түйістілік матрицалары ЭЕМ жадысында графтарды сақтаудың қолайсыз тәсілдердің бірі болып есептеледі, өйткені оның мәндері анық мәліметті бермейді. Мысалы, матрицаның өзінде 1-ші мен 2-ші төбелерінің байланысқанын анықтау мүмкін емес.

12.2.2 Графты сыбайлас матрица арқылы көрсету. Сыбайлас матрицада жолдар мен бағаналар саны төбелер санына тең.

Қиылыста осы төбелердің байланысын сипаттайтын мән орналасады, мысалы, егер қабырға болса, онда 1-ге, егер қабырға жоқ болса, онда 0-ге тең.

Бағытталған графтар үшін сыбайлас матрицада жолдар мен бағаналар саны төбелер санына тең.

Қиылыста осы төбелердің байланысын сипаттайтын мән орналасады, мысалы, егер сәйкес төбелер арасында байланыс бар болса, онда 1-ге, егер байланыс жоқ болса, онда 0-ге тең.

12.2-суретте көрсетілген граф үшін сыбайлас матрица 12.3-кестеде көрсетілген.

12.3-кесте – Сыбайлас матрица

	1	2	3	4	5	6
1	0	1	0	1	1	0
2	1	0	1	1	0	0
3	0	1	0	0	1	1
4	1	1	0	0	1	0
5	1	0	1	1	0	1
6	0	0	1	0	1	0

12.3-суретте көрсетілген бағытталған граф үшін сыбайлас матрица 12.4-кестеге сәйкес келеді.

12.4-кесте – Бағытталған графтың сыбайлас матрицасы

	1	2	3	4	5	6
1	0	1	1	0	0	0
2	0	0	0	0	0	0
3	0	1	0	1	0	0
4	0	0	0	0	0	0
5	0	0	0	1	0	1
6	0	0	0	0	1	0

12.2.3 Графты жұптар тізімі түрінде көрсету. Графты төбелердің сыбайлас жұптарының тізімі түрінде көрсету. Мысалы, жоғарыда ұсынылған графтарды төбелердің сыбайлас жұптарының тізімімен көрсетуге болады.

1-граф	2-граф
1) 1 – 2	1 – 2
2) 1 – 4	1 – 3
3) 1 – 5	3 – 2
4) 2 – 3	3 – 4
5) 2 – 4	5 – 4
6) 3 – 5	5 – 6
7) 3 – 6	6 – 5
8) 4 – 5	1– 2
9) 5 – 6	1– 2

Осы тізімдерді ЭЕМ жадында екі өлшемді массив түрінде көрсетуге болады. ЭЕМ жадында графтарды осы жолмен сақтау ең тиімді болып есептеледі, бірақ жұмыс жасауға үнемі қолайлы бола бермейді.

12.2.4 Сыбайлас төбелердің тізімі түрінде графты көрсету. Графтарға арналған кейбір есептерде граф төбелерін сыбайлас төбелердің тізімдік құрылымы түрінде көрсетуді талап етеді. Мысалы, стек, кезек, қарапайым тізім. Осы жағдайда әдетте сыбайлас төбелер тізімдерінің тақырыптары бойынша массив құрылады. Мысалы, 12.2-суретте көрсетілген графты тақырыптар массиві, графтың көршілес төбелерінің тізімі түрінде, төменде көрсетілгендей жазуға болады:

M[1]->1->2->4->5->NULL;
M[2]->2->1->3->4->NULL;
M[3]->3->2->5->6->NULL;
M[4]->4->1->2->5->NULL;
M[5]->5->3->4->6->NULL;
M[6]->6->3->5->NULL;

12.3-суретте көрсетілген графты тақырыптар массиві, графтың көршілес төбелерінің тізімі түрінде, төменде көрсетілгендей жазуға болады:

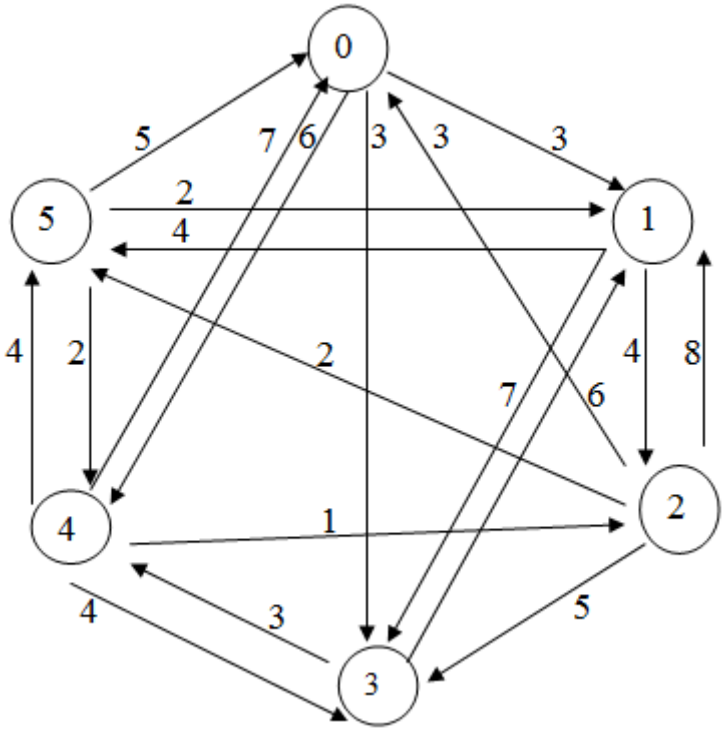
M[1]->1->2->3->NULL;
M[2]->2->NULL;
M[3]->3->2->4->NULL;
M[4]->4->NULL;
M[5]->5->4->6->NULL;
M[6]->6->5->NULL;

Графты көрсетудің осы формасы түрлі графты жүріп өту алгоритмдерін орындау үшін қолайлы.

12.3 Флойд алгоритмі

Көлік туралы есептердің көптеген түрлері бар, есепте графтың әр түрлі төбелерінің арасындағы арақашықтықты анықтау немесе оны қолдану керек және осы арақашықтық ең қысқа арақашықтық болуы керек.

Графтар теориясында граф төбелерінің арасындағы ең қысқа арақашықтықты анықтайтын көптеген алгоритмдер белгілі, бірақ оның барлығы Флойд ұсынған алгоритмді негізге алады. Осы алгоритмнің кең қолданылуы төбелер арасындағы ең қысқа арақашықтықты табу идеясында, бірақ есептеу бойынша Флойд алгоритмінің тиімділігі өте төмен. Флойд алгоритмін қарастырайық. 6 төбесі бар бағытталған граф бар болсын, 12.4-суретте көрсетілген.



12.4-сурет – Бағытталған граф

Осы графтың сыбайлас матрицасы 12.5-кестесінде көрсетілген:

12.5-кесте –12.4-суреттегі графтың сыбайлас матрицасы

	0	1	2	3	4	5
0	0	3	--	3	6	--
1	--	0	4	7	--	4
2	3	8	0	5	--	2
3	--	6	--	0	3	--
4	7	--	1	4	0	4
5	5	2	--	--	2	0

Флойд алгоритмінің идеясы бойынша әрбір a[i,j] – төбелер жұбына k (0 – 5) төбелерінің барлық мүмкін нұсқалары мына түрде тексеріледі: a[i,j]>a[i,k]+a[k,j].

Егер a[i,j] жолы үлкен болса, онда сыбайлас матрицада i және j төбелерінің арасындағы жолдың мәні a[i,k]+a[k,j] жолына тең жаңа мәнмен ауыстырылады. Сонымен, ең қысқа қашықтықты анықтайтын матрица дайын болды.

Мысалы, сыбайлас матрицада 2-шіден 1-г дейін жол 8-ге тең. Бірақ, егер біз бірінші 0-ге, ал содан кейін 1-ге орын ауыстыратын болсақ, онда жол 6-ға тең болады: a[2,0]+a[0,1]= 3+3=6. Барлық нұсқаларды тексере отырып, біз мынаны байқадық: 2-5-1 жолы қысқарық, a[2,5]+a[5,1]=2+2=4.

Сонымен, 2-ші мен 1-ші төбелердің арасындағы ең қысқа жол табылды.

Бағдарлама коды:

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
 {
  int[,] a = new int[6, 6]
  {{0,3,1000,3,6,1000},
  {1000,0,4,7,1000,4},
     {3,8,0,5,1000,2},
  {1000,6,1000,0,3,1000},
     {7,1000,1,4,0,4},
  {5,2,1000,1000,2,0}};
  int i, j, w, n, k;
  Console.WriteLine("Matrizani shigary: ");
  for (i = 0; i < 6; i++)
  {
  for (j = 0; j < 6; j++)
   Console.Write("\t" + a[i, j]);
  Console.WriteLine();
  }
  Console.WriteLine();
  for (k = 0; k < 6; k++)
  for (i = 0; i < 6; i++)
   for (j = 0; j < 6; j++)
   if (a[i, j] > a[i, k] + a[k, j])
    a[i, j] = a[i, k] + a[k, j];
  Console.WriteLine("Zhana matrizani shigary: ");
  for (i = 0; i < 6; i++)
  {
  for (j = 0; j < 6; j++)
```

```
   Console.Write("\t" + a[i, j]);
  Console.WriteLine();
  }
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысының нәтижесі:

```
Matrizani shigary:
 0 3 1000 3 6 1000
 1000 0 4 7 1000 4
 3 8 0 5 1000 2
 1000 6 1000 0 3 1000
 7 1000 1 4 0 4
 5 2 1000 1000 2 0

Zhana matrizani shigary:
 0 3 7 3 6 7
 7 0 4 7 6 4
 3 4 0 5 4 2
 7 6 4 0 3 6
 4 5 1 4 0 3
 5 2 3 6 2 0
```

Бағдарламаның (алгоритмнің) кемшіліктері:

– ең қысқы арақышықтықтың маршруты анықталмайды;

– алгоритмнің есептеу тиімділігі n-нің 3-ші дәрежесіне пропорционал (мұнда n – граф төбелерінің саны), ол n-нің үлкен мәндерінде тиімсіз болып келеді.

12.4 Дейкстр алгоритмі

Көптеген авторлардың пікірінше Дейкстр алгоритмі графта берілген екі төбе арасындағы ең қысқа маршрутты табу үшін ең тиімді алгоритм болып есептеледі.

Бағытталмаған граф берілген болсын, ол 12.5-суретінде көрсетілген. Дейкстр алгоритмінде үш массивті қолдану керек, олардың өлшемі граф төбелерінің санына сәйкес болады.

Бірінші массив - «тұрақты» төбелер массиві графтың берілген төбесінен барлық қалған төбелеріне дейінгі ең қысқа маршруттарды сақтайды. Массивке бірінші болып бірінші төбенің нөмері жазылады (осы төбелер арқылы графтың басқа төбелеріне үшін ең қысқа ара қашықтық анықталады). Массивтің атауы таңдап алынған төбелердің атауларына сәйкес болады. Дейкстр алгоритмі бойынша графтың барлық төбелері «уақытша» болып жарияланады, ал таңдап алынған төбелер «тұрақты» деп аталады. Бірінші төбе «тұрақты» болып жарияланады.

Екінші массив графтың берілген төбесінен барлық қалған төбелеріне дейінгі ең қысқа арақашықтықты сақтайды. Бірінші болып оған таңдап алынған төбе нөмеріне сәйкес сыбайлас матрицасының жолы көшіріліп жазылады. Графтың сыбайлас матрицасы 12.6-кестеде көрсетілген.

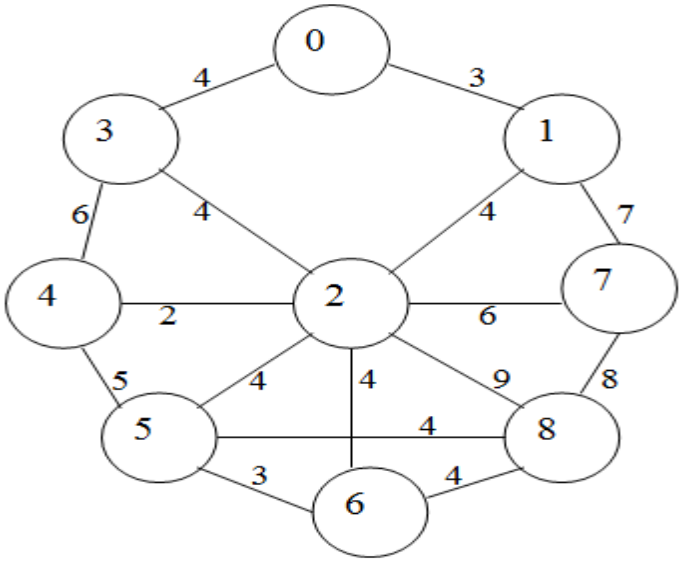
Үшінші логикалық типтегі массивте графтың таңдап алынған (тұрақты) төбелер жазылады. Алғашында графтың «тұрақты» төбесі ретінде бірінші төбе ғана бола алады.

Одан кейін «уақытша» төбе таңдалады, бірінші (тұрақты) төбеден осы төбеге дейінгі аралық ең қысқа болуы керек. Осы төбе `k` төбесі болып жарияланады.

Графтың «тұрақты» төбесінен барлық «уақытша» төбеге дейінгі барлық маршруттар тікелей және `k` төбесі арқылы тексеріледі. ең қысқа аралықтар екінші массивке жазылады. Ал, егер ең қысқа арақашықтықтың маршруты `k` төбесі арқылы өтетін болса, онда бірінші массивке сәйкес позицияда k жазылады.

12.6-кесте – 12.5-суреттегі графтың сыбайлас матрицасы

	0	1	2	3	4	5	6	7	8
0	0	3	--	4	--	--	--	--	--
1	3	0	4	--	--	--	--	7	--
2	--	4	0	5	2	4	4	6	9
3	4	--	5	0	6	--	--	--	--
4	--	--	2	6	0	5	--	--	--
5	--	--	4	--	5	0	3	--	4
6	--	--	4	--	--	3	0	--	4
7	--	7	6	--	--	--	--	0	8
8	--	--	9	--	--	4	4	8	0



12.5–сурет – Қарапайым бағытталмаған граф

Ары қарай k төбесі «тұрақты» болып жарияланады (ол үшінші массивте орындалады). Графтың келесі төбесін іздеу алгоритмі жаңа «тұрақты» төбеге қатысты қайталанады.

Бірінші төбенің нөмері 0-ге тең деп жорамалдайық. Онда post[…]– бірінші массив нөлдерден тұрады. Екінші массивте, яғни ең қысқа қышықтықтар массивінде сыбайлас матрицаның нөлінші жолының мәндері жазылады:

0 1 2 3 4 5 6 7 8

d[min] – 0 3 1000 4 1000 1000 1000 1000 1000

Үшінші масссив, яғни таңдап алынған (выбранный) төбелер массивінің мәндері true болады, тек нөлінші элементте ғана t[0]=false;.

0-ші төбеден басқа бір төбеге дейінгі ең қысқа аралықты таңдаймыз. Біздің жағдайымызда ол 1-ші төбе болады, оған дейінгі қашықтық 3-ке тең болады. Таңдап алынған төбе k деп аталсын. Флойд алгоритмін қолданып, алғашқы төбеден “k” төбесі арқылы өтетін қалған басқа төбелерге дейінгі барлық ең қысқа маршруттарды табамыз.

```
     for (j=0;j<9;j++)
      if ((t[j]==true)&&(d[j]>d[k]+a[k,j]))
       {
  d[j]=d[k]+a[k,j];
  post[j]=k;
   }
```

нәтижесінде d және post массивтерінің жаңа мәнін аламыз:

```
         0 1 2 3 4 5 6 7 8
     d[min] –    0 3 7 4 1000 1000 1000 10 1000
0 1 2 3 4 5 6 7 8
     post[…] – 0 0 1 0 0 0 0 1 0
```

Бұл 0-2 төбелерінің арасындағы ең қысқа арақашықтық 7-ге тең болатынын білдіреді.

Таңдап алынған k төбесін t[k]=false «тұрақтысымен» жариялаймыз және осы төбеге қатысты процесс қайталанып отырады.

Бағдарлама коды:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
 class Program
 {
 public static int[,] a = new int[9, 9]
 {{ 0, 3,1000, 4,1000,1000,1000,1000,1000},
 { 3, 0, 4,1000,1000,1000,1000, 7,1000},
 {1000, 4, 0, 5, 2, 4, 4, 6, 9},
 { 4,1000, 5, 0, 6,1000,1000,1000,1000},
```

```
 {1000,1000, 2, 6, 0, 5,1000,1000,1000},
 {1000,1000, 4,1000, 5, 0, 3,1000, 4},
 {1000,1000, 4,1000,1000, 3, 0,1000, 4},
 {1000, 7, 6,1000,1000,1000,1000, 0, 8},
 {1000,1000, 9,1000,1000, 4, 4, 8, 0}};
 public static int[] d = new int[10];
 public static int[] post = new int[10];
 public static bool[] t = new bool[10];
 public static int i, j, p, k, minras;
 public static void poisk()
 {
  //алғышарттар
  minras = 0;
  for (i = 0; i < 9; i++)
  {
  post[i] = 0;
  t[i] = true;
  d[i] = a[0, i];
  }
  t[0] = false;
  post[0] = 0;
  for (i = 0; i < 8; i++)
  {
  // k төбесін іздеу
  minras = 1000;
  for (j = 0; j < 9; j++)
   if ((t[j] == true) && (minras > d[j]))
   {
   minras = d[j]; k = j;
   }
  // k төбесі аркылы маршруттарды іздеу және минимальді
қашықтықты табу
  t[k] = false;
  for (j = 0; j < 9; j++)
   if ((t[j] == true) && (d[j] > d[k] + a[k, j]))
   {
   d[j] = d[k] + a[k, j];
   post[j] = k;
   }
  }
 }
 public static void print()
 {
  // сыбайлас матрицаны шығару
  Console.WriteLine("Sibailas matrisani shigary: ");
  for (i = 0; i < 9; i++)
  {
  for (j = 0; j < 9; j++)
   Console.Write("\t" + a[i, j]);
  Console.WriteLine();
  }
  Console.WriteLine();
```

```
  // минимальді қашықтықтарды сақтайтын массивті шығару
  for (i = 0; i < 9; i++) Console.Write("\t" + i);
  Console.WriteLine();
  for (i = 0; i < 9; i++) Console.Write("\t" + d[i]);
  Console.WriteLine();
  //маршруттар массивін шығару
  for (i = 0; i < 9; i++) Console.Write("\t" + post[i]);
  Console.WriteLine();
  // графтың 0-і төбесінен бастап 8-і төбесіне дейін маршрутты
шығару
  p = 8;

  Console.Write("Songi tobe = 8 ");
  do
  {
  p = post[p];
  Console.Write("\t" + p);
  }
  while (p != 0);
  Console.WriteLine();
  Console.WriteLine("Bastapki tobe = 0 ");
 }
 static void Main(string[] args)
 {
  poisk();
  print();
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
Sibailas matrisani shigary:
0 3 1000 4 1000 1000 1000 1000 1000
3 0 4 1000 1000 1000 1000 7 1000
1000 4 0 5 2 4 4 6 9
4 1000 5 0 6 1000 1000 1000 1000
1000 1000 2 6 0 5 1000 1000 1000
1000 1000 4 1000 5 0 3 1000 4
1000 1000 4 1000 1000 3 0 1000 4
1000 7 6 1000 1000 1000 1000 0 8
1000 1000 9 1000 1000 4 4 8 0

0 1 2 3 4 5 6 7 8
0 3 7 4 9 11 11 10 15
0 0 1 0 2 2 2 1 5
Songi tobe = 8 5 2 1 0
Bastapki tobe = 0
```

Бағдарлама жұмысының нәтижесі – екі массивті шығару, яғни графтың берілген төбесінен қалған төбелеріне дейінгі ең қысқа аралықтар мен ең қысқа маршруттардың массивтері. Мысал ретінде графтың 0-ші

төбесінен 8-ші төбесіне дейінгі ең қысқа маршрут қарастырылады 8-ші мен 0-ші төбелерінің арасындағы ең қысқа аралық 15-ке тең. Орын ауыстыру маршруты: 8 – 5 – 2 – 1 – 0.

12.5 Өзін–өзі тексеру сұрақтары

1 Қабырға арқылы байланысатын төбелерді қалай атайды?

2 Графтың жолы туралы ұғым.

3 Графтың қандай жолы қарапайым жол деп аталады?

4 Егер кез келген төбеден басқа төбеге жол бар болса, онда ондай граф қалай аталады?

5 Циклдері жоқ, байланыстаралған граф қалай аталады?

6 Графта оның барлық төбелерін өзіне қосатын цикл бар болса (бірақ барлық қабырғалары міндетті емес), онда ондай графты қалай атайды?

7 Сыбайлас матрицада бағана мен жол қиылысында не жазылады?

8 Егер берілген төбеден графтың кез келген төбесіне жол бар болса, онда осы төбе қалай аталады?

9 Флойд алгоритмі нені анықтайды?

10 Дейсктр алгоритмі нені анықтайды?

13 ГРАФТАРҒА АРНАЛҒАН АЛГОРИТМДЕР

13.1 Графтарды жүріп өту алгоритмдері

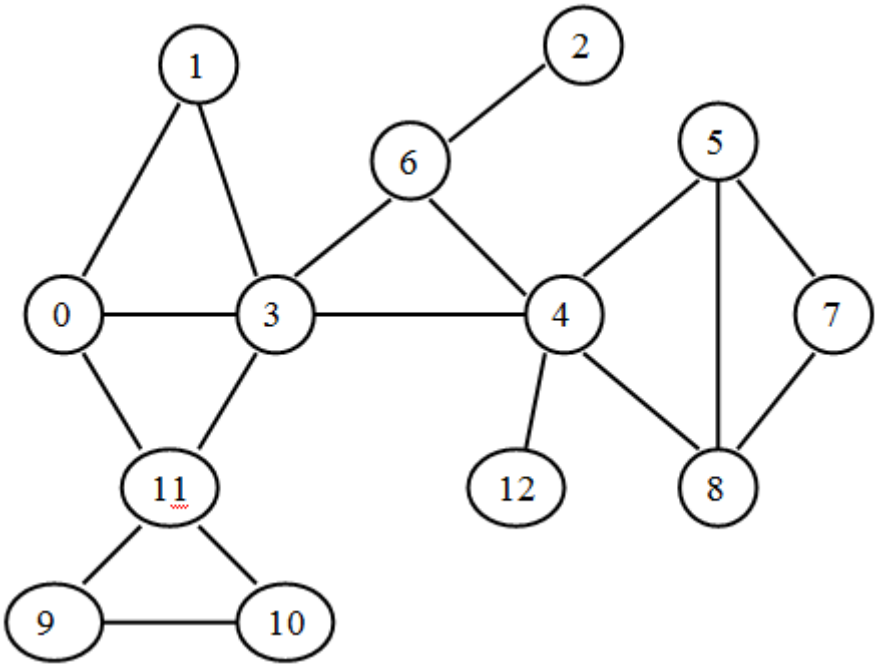
Графтың барлық төбелерін «қарап шығуды» қажет ететін көлік есептерінің тобы бар. Сонымен қатар әрбір төбе бір рет қана «қаралуы» керек.

Графтар теориясында граф төбелерін жүріп өтудің екі негізгі алгоритмі белгілі: графты «тереңдігі» бойынша және «көлденеңі» бойынша жүріп өту.

Бірінші жағдайда алгоритмде стек типіндегі құрылым, ал екінші жағдайда – кезек типіндегі құрылым қолданылады.

«Алгоритмдеу» тұрғысынан бірінші жағдайға көп мән беруге болады, өйткені онда күрделі құрылымдарда қолданылатын «қайтару» алгоритмі жазылады.

Графты «тереңдігі» бойынша жүріп өту алгоритмінің жұмысын 13.1-суретте көрсетілген, күрделілігі орташа мысалда қарастырайық.



13.1-сурет – Күрделілігі орташа граф

Графты «тереңдігі» бойынша жүріп өту алгоритмі графтың «жаңа» төбелері, яғни «қаралмаған» төбелері ұғымын қолданады. Оның жұмысын қарастырайық:

– графтың барлық төбелері «жаңа» деп жарияланады (Дейкстр алгоритмінде «уақытша» термині қолданылады). Графты қарап өтуді белгілі бір төбеден бастаймыз, мысалы 0-ші төбеден оны қарап өткеннен кейін оны «жаңа емес» деп белгілейміз;

– берілген төбемен қосылған бүкіл жаңа сыбайлас тізімнен нөмірі ең төмен төбені таңдаймыз. Қарастырылған мысалда ол 1-төбе;

– оны алғашқы деп жариялаймыз, оны қарап шыққаннан кейін оны «жаңа емес» деп белгілейміз;

– егер қарап өтілген төбенің жаңа сыбайлас төбелері болмаса, онда алдыңғы төбеге қайта ораламыз (стек жұмысы);

– процесс «жаңа» төбелер бар болғанша қайталанады.

13.1-суретте көрсетілген графқа қарастырылған алгоритмді қолданып төбелерді қарап шығу тізбегін аламыз:.

0 – 1 – 3 – 4 – 5 – 7 – 8 – 6 – 2 – 12 – 11 – 9 – 10.

Графты «көлденеңі» бойынша қарастырғанда да графтың «жаңа» төбелері деген ұғымды қолданады. Оның жұмысын қарастырайық:

– графтың барлық төбелері «жаңа» деп жарияланады. Графты жүріп өтуді белгілі бір алғашқы төбеден бастайық, мысалы, 0-төбеден. Ол төбені қарап шыққаннан (төбе нөмерін басамыз) кейін оны «жаңа емес» деп белгілейміз;

– қарап шыққан төбемен байланысты барлық жаңа сыбайлас төбелер тізімі өсу ретімен кезекке тұрғызамыз: 1 – 3 – 11;

– кезектен төбені таңдаймыз (осы мысалды ол 1-төбе), оны қарап өтеміз, «жаңа емес» деп белгілейміз. Қаралып өткен тізіммен байланысы бар сыбайлас төбелердің бүкіл тізімі өсу ретімен кезекке тұрғызамыз: 3 – 11;

– процесс графта төбелер бар болғанша қайталанады.

Графтың қаралған төбелерінің:

0 – 1 – 3 – 11 – 4 – 6 – 9 – 10 – 5 – 8 – 12 – 2 – 7.

13.1-суретінде көрсетілген граф мысалында Графты «тереңдігі» бойынша жүріп өту алгоритмінің бағдарламалақ орындалуын қарастырайық.

Бағдарлама коды:

```
using System;
using System.Collections;
using System.Text;
namespace ConsoleApplication1
{
 class Program
 {
 public static int i, j, k, n,kol;
 static void vkl(Stack vst, int n)
 {
  vst.Push(n);
 }
 static void iskl(Stack vst)
 {
  if (vst == null) Console.WriteLine("Stek bos!");
```

```
 else
 n = (int)vst.Pop();
 }
 public static void Main()
 {
 Stack vstek = new Stack();
 //int i, j, k, n;
 bool[] nov = new bool[13];
 int[,] p = new int[13, 13];
int[,] a = new int[13, 13]
{{0,1,1000,1,1000,1000,1000,1000,1000,1000,1000, 1, 1000 },
 {1,0,1000,1,1000,1000,1000,1000,1000,1000,1000,1000, 1000},
 {1000,1000,0,1000,1000,1000,1,1000,1000,1000,1000,1000,1000},
 {1,1,1000,0,1,1000,1,1000,1000,1000,1000,1,1000},
 {1000,1000,1000,1,0,1,1,1000,1,1000,1000,1000, 1},
 {1000,1000,1000,1000,1,0,1000,1,1,1000,1000,1000,1000},
 {1000,1000,1,1,1,1000, 0,1000,1000,1000,1000,1000,1000},
 {1000,1000,1000,1000,1000,1,1000,0,1,1000,1000,1000,1000},
 {1000,1000,1000,1000,1,1,1000,1,0,1000,1000,1000,1000},
 {1000,1000,1000,1000,1000,1000,1000,1000,1000,0,1,1,1000},
 {1000,1000,1000,1000,1000,1000,1000,1000,1000,1,0,1,1000},
 {1,1000,1000,1,1000,1000,1000,1000,1000,1,1,0,1000},
 {1000,1000,1000,1000,1,1000,1000,1000,1000,1000,1000,1000,0}, };
 for (i = 0; i < 13; i++)
 {
 p[i,0] = i;k=1;
 for (j = 0; j < 13; j++)
 if ((a[i, j] != 1000) && (a[i, j] != 0))
 {
  p[i,k]=j;
  k++;
 }
 p[i,k]=1000;
 }
 for (i = 0; i < 13; i++)
 {
 k = 0;
 while (p[i,k] != 1000)
 {
  Console.Write(" {0}", p[i,k]);
  k++;
 }
 Console.WriteLine();
 }
```

```
 // графты жүріп өту
 bool b;
 // Бастапқы шарттарды беру
 for (i = 0; i < 13; i++) nov[i] = true;
 vkl(vstek,p[0,0]);
 kol = 1;
 Console.Write(" {0}", p[0, 0]);
 nov[0] = false;
 // графты «тереңдігі» бойынша жүріп өту циклі
 while (kol != 0)
 { i = (int)vstek.Peek();
 if (p[i,0] == 1000) b = false;
 else b = !nov[p[i,0]];
 // графтың жаңа төбесін іздеу
 k = 0;
 while (b == true)
 {
  k++;
  if (p[i,k] == 1000) b = false;
  else
 {
 b = !nov[p[i,k]];
 if (nov[p[i, k]]) { vkl(vstek, p[i, k]); kol++; }
 }
 }
 if (p[i,k] != 1000)
 // егер графтың жаңа төбесі табылса
 {
  i = p[i,k];
  Console.Write(" {0}", i);
  nov[i] = false;
 }
 else
// тізімде жаңа төбе жоқ болса, алдынғы төбеге оралу керек
 {
  iskl(vstek); i = n; kol--;
 }
 }
Console.WriteLine();
Console.WriteLine("Enter pernesin basiniz ");
Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

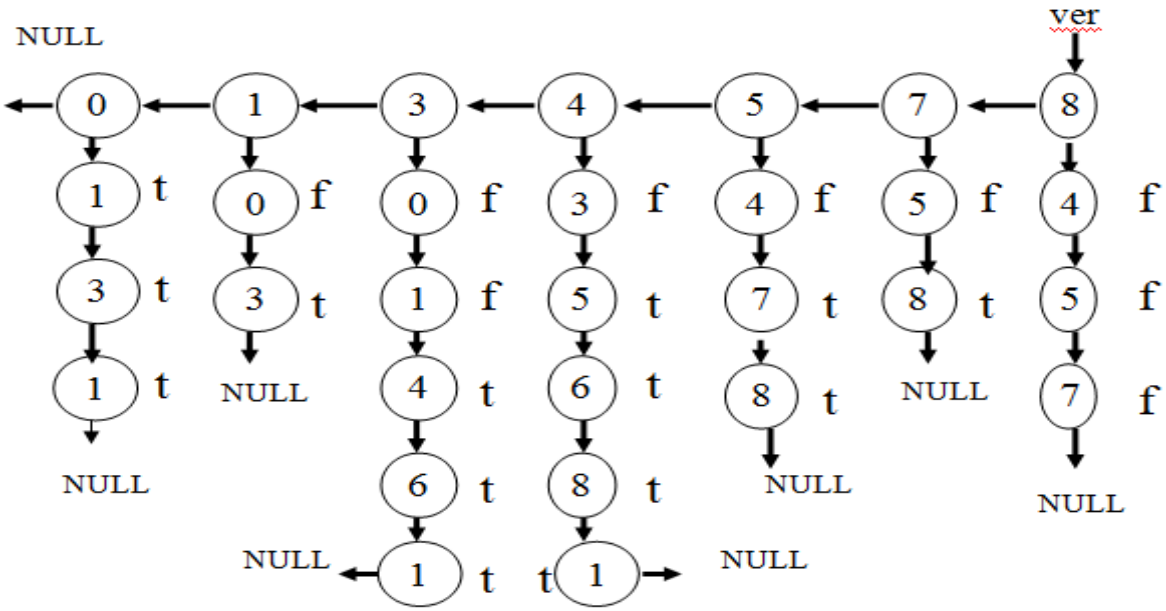
```
 0  1  3  11
 1  0  3
 2  6
 3  0  1  4  6  11
 4  3  5  6  8  12
 5  4  7  8
 6  2  3  4
 7  5  8
 8  4  5  7
 9  10  11
 10  9  11
 11  0  3  9  10
 12  4
 0  1  3  4  5  7  8  6  2  12  11  9  10
Enter pernesin basiniz
```

Бағдарлама басында p[13,13] матрицасы құрылады, онда барлық сыбайлас төбелер тізімі жазылады. Осы тізімдердің мәндерін бақылау үшін олар манитор экранынан шығарылады.

Бағдарламада стек қолданылған, онда біріншіден графтың алғашқы төбе тізімінің тақырыбы жазылады. Графты «тереңдігі» бойынша жүріп өту циклі жұмысының барысында стекке графтың ең кіші жаңа сыбайлас төбелерінің тізім тақырыптары жазылады, олар тақырыбы стек төбесінде орналасқан тізімнен таңдап алынады. Егер осы тізімде жаңа сыбайлас төбе жоқ болса, онда сәйкес тізімнің тақырыбы стектен жойылады, стек төбесі алдыңғы тізімге көшеді.

Графтың сыбайлас төбелер тізімдерінің тақырыптары стекте жоқ болса алгоритм жұмысы аяқталады.

Стектің графты «тереңдігі» бойынша жүріп өту жұмысының бастапқы үзіндісі 13.2-суретінде көрсетілген.



13.2-сурет – Графты «тереңдігі» бойынша жүріп өтудегі стектің жұмысы

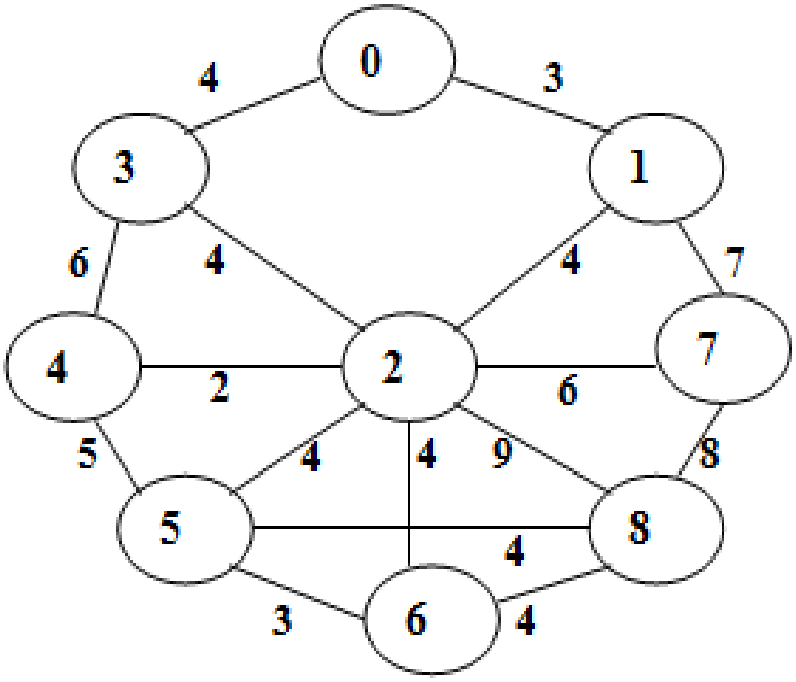
13.2-суретте графтың сыбайлас төбелерінің бірінші тізімінен 1-төбе, екінші тізімінен 3-төбе, үшінші тізімінен - 4-төбе, т.б. таңдап алынғаны көрсетілген. Стектің толуына қарай таңдап алынған төбелердің мәндері true (t) мәнінен false (f) мәніне ауысады. 8-төбені қарап шыққаннан кейін стектен 8, 7, 5 төбелері (оларда жаңа төбелер жоқ) жойылады. 4-төбенің тізімінде жаңа 6-шы нөмерлі төбе бар, оның тізімінің тақырыбы стекке қосылады, т.б.

13.2 Берілген екі төбе арасындағы барлық маршруттарды іздеу алгоритмі

Берілген екі төбе арасындағы барлық маршруттарды іздеу алгоритмі графты «тереңдігі» бойынша жүріп өтуге негізделген. Бірінші айырмашылық – стекті граф төбелерінің нөмірлерін ғана емес, сонымен қатар сыбайлас төбелер тізімін және тізімдегі төбе позициясын сақтау үшін қолдану.

Екінші айырмашылық – граф төбелеріне «жаңа» мәндерді қайтару. Егер граф төбесінің тізімі аяқталса, яғни «жаңа» төбелер жоқ болса, онда стектен төбе жойылады және осы төбені басқа маршруттарда қолдану үшін оған «жаңа»деген мән қайтарылады.

13.1-есеп. Сыбайлас матрицамен (13.1-кесте) берілген 13.3-суреттегі граф үшін диалог режімінде берілетін екі төбе арасындағы барлық маршрутты табатын бағдарламаны құру керек.



13.3-сурет – Бағытталған граф

13.1-кесте – 13.3-суреттегі графтың сыбайлас матрицасы

	0	1	2	3	4	5	6	7	8
0	0	3	1000	4	1000	1000	1000	1000	1000
1	3	0	4	1000	1000	1000	1000	7	1000
2	1000	4	0	5	2	4	4	6	9
3	4	1000	5	0	6	1000	1000	1000	1000
4	1000	1000	2	6	0	5	1000	1000	1000
5	1000	1000	4	1000	5	0	3	1000	4
6	1000	1000	4	1000	1000	3	0	1000	4
7	1000	7	6	1000	1000	1000	1000	0	8
8	1000	1000	9	1000	1000	4	4	8	0

Бағдарлама коды:

```
using System;
using System.Collections;
using System.Text;
namespace ConsoleApplication1
{
 class Program
 {
 public static int i, j, k, kol;
 public struct uzel
 {
  public int nom;
  public int ki;
  public int kj;
 };
 public static uzel n = new uzel();
 static void vkl(Stack vst, uzel n)
 {
  vst.Push(n);
 }
 static void iskl(Stack vst)
 {
  if (vst == null) Console.WriteLine("Стек бос !");
  else n = (uzel)vst.Pop();
 }
 public static void Main()
 {
  Stack vstek = new Stack();
  string buf;
  int beg, en, x, i1, i2;
  uzel y1 = new uzel();
  bool[] nov = new bool[10];
  int[,] p = new int[9, 9];
  int[] m = new int[10];
  //графтың сыбайлас матрицасы
```

```
 int[,] a = new int[9, 9]
{{ 0, 6,1000, 4,1000,1000,1000,1000,1000},
{ 6, 0, 5,1000,1000,1000,1000, 4,1000},
{1000, 5, 0, 3, 4,1000, 5, 6,1000},
{ 4,1000, 3, 0, 7,1000,1000,1000,1000},
{1000,1000, 4, 7, 0, 5,1000,1000,1000},
{1000,1000,1000,1000, 5, 0, 4,1000, 9},
{1000,1000, 5,1000,1000, 4, 0,1000, 6},
{1000, 4, 6,1000,1000,1000,1000, 0, 8},
{1000,1000,1000,1000,1000, 9, 6, 8, 0}};
 //a матрицасы бойынша сыбайлас төбелер тізімін құру
 for (i = 0; i < 9; i++)
 {
 p[i, 0] = i; k = 1;
 for (j = 0; j < 9; j++)
  if ((a[i, j] != 1000) && (a[i, j] != 0))
  {
  p[i, k] = j;
  k++;
  }
 p[i, k] = 1000;
 }
 //сыбайлас төбелер тізімін экранға шығару
 for (i = 0; i < 9; i++)
 {
 k = 0;
 while (p[i, k] != 1000)
 {
  Console.Write(" {0}", p[i, k]);
  k++;
 }
 Console.WriteLine();
 }
 Console.Write("Graftin bastapki tobesin engiziniz? ");
 buf = Console.ReadLine();
 beg = Convert.ToInt32(buf);
 Console.Write("Graftin akirgi tobesin engiziniz? ");
 buf = Console.ReadLine();
 en = Convert.ToInt32(buf);
 for (i = 0; i < 9; i++)
 {
 nov[i] = true;
 m[i] = 0;
 }
 nov[9] = false;
 x = 1;  // кезекті маршруттың нөмері
 m[1] = beg; i1 = 2;
 y1.nom = beg; //y1 түйіннінің nom өрісін анықтаймыз
 y1.ki = beg; //сыбайлас төбелер тізімінің нөмерін сақтаймыз
 y1.kj = 0; // тізімдегі ағымдағы орнын сақтаймыз
 vkl(vstek, y1);
 kol = 1; //стектегі элементтер саны
```

```
 nov[beg] = false;
 nov[en] = false;
 i = beg; j = 0;
 // графтағы барлық маршруттарды іздеу циклі
 while (kol != 0)
 {
 do
 {
  j++;
  if (p[i, j] == en)
  {
  Console.Write(" Жол - {0} -", x); x++;
  for (i2 = 1; i2 < i1; i2++)
   Console.Write(" {0}", m[i2]);
  Console.WriteLine(" {0}", en);
  }
 }
 while ((p[i, j] != 1000) && (!nov[p[i, j]]));
 if (p[i, j] != 1000)
  if (nov[p[i, j]])
  {
  y1.ki = i;
  y1.kj = j;
  i = p[i, j];
  y1.nom = i;
  vkl(vstek, y1);
  j = 0;
  kol++;
  nov[i] = false;
  m[i1] = i;
  i1++;
  };

 if (p[i, j] == 1000)
 {
  kol--;
  if (kol != 0)
  {
  iskl(vstek);
  i = n.ki;
  j = n.kj;
  i1--;
  m[i1] = 0;
  nov[n.nom] = true;
  }
 };
 }
 Console.WriteLine();
 Console.WriteLine("Enter pernesin basiniz");
 Console.ReadLine();
}
}
```

```
}
```

Бағдарлама жұмысы:

```
0 1 3
 1 0 2 7
 2 1 3 4 6 7
 3 0 2 4
 4 2 3 5
 5 4 6 8
 6 2 5 8
 7 1 2 8
 8 5 6 7
Graftin bastapki tobesin engiziniz? 1
Graftin akirgi tobesin engiziniz? 4
 Жол - 1 - 1 0 3 2 4
 Жол - 2 - 1 0 3 2 6 5 4
 Жол - 3 - 1 0 3 2 6 8 5 4
 Жол - 4 - 1 0 3 2 7 8 5 4
 Жол - 5 - 1 0 3 2 7 8 6 5 4
 Жол - 6 - 1 0 3 4
 Жол - 7 - 1 2 3 4
 Жол - 8 - 1 2 4
 Жол - 9 - 1 2 6 5 4
 Жол - 10 - 1 2 6 8 5 4
 Жол - 11 - 1 2 7 8 5 4
 Жол - 12 - 1 2 7 8 6 5 4
 Жол - 13 - 1 7 2 3 4
 Жол - 14 - 1 7 2 4
 Жол - 15 - 1 7 2 6 5 4
 Жол - 16 - 1 7 2 6 8 5 4
 Жол - 17 - 1 7 8 5 4
 Жол - 18 - 1 7 8 5 6 2 3 4
 Жол - 19 - 1 7 8 5 6 2 4
 Жол - 20 - 1 7 8 6 2 3 4
 Жол - 21 - 1 7 8 6 2 4
 Жол - 22 - 1 7 8 6 5 4

Enter pernesin basiniz
```

Егер екі бірдей төбелер берілсе, онда графтың барлық циклдарын табуға болады, олар берілген төбеде басталып, осы төбеде аяқталады. Мысалы:

```
0 1 3
 1 0 2 7
 2 1 3 4 6 7
```

3 0 2 4
4 2 3 5
5 4 6 8
6 2 5 8
7 1 2 8
8 5 6 7
Graftin bastapki tobesin engiziniz? 6
Graftin akirgi tobesin engiziniz? 6
Жол - 1 - 6 2 1 0 3 4 5 6
Жол - 2 - 6 2 1 0 3 4 5 8 6
Жол - 3 - 6 2 1 7 8 5 6
Жол - 4 - 6 2 1 7 8 6
Жол - 5 - 6 2 3 0 1 7 8 5 6
Жол - 6 - 6 2 3 0 1 7 8 6
Жол - 7 - 6 2 3 4 5 6
Жол - 8 - 6 2 3 4 5 8 6
Жол - 9 - 6 2 4 3 0 1 7 8 5 6
Жол - 10 - 6 2 4 3 0 1 7 8 6
Жол - 11 - 6 2 4 5 6
Жол - 12 - 6 2 4 5 8 6
Жол - 13 - 6 2 6
Жол - 14 - 6 2 7 1 0 3 4 5 6
Жол - 15 - 6 2 7 1 0 3 4 5 8 6
Жол - 16 - 6 2 7 8 5 6
Жол - 17 - 6 2 7 8 6
Жол - 18 - 6 5 4 2 1 7 8 6
Жол - 19 - 6 5 4 2 3 0 1 7 8 6
Жол - 20 - 6 5 4 2 6
Жол - 21 - 6 5 4 2 7 8 6
Жол - 22 - 6 5 4 3 0 1 2 6
Жол - 23 - 6 5 4 3 0 1 2 7 8 6
Жол - 24 - 6 5 4 3 0 1 7 2 6
Жол - 25 - 6 5 4 3 0 1 7 8 6
Жол - 26 - 6 5 4 3 2 1 7 8 6
Жол - 27 - 6 5 4 3 2 6
Жол - 28 - 6 5 4 3 2 7 8 6
Жол - 29 - 6 5 6
Жол - 30 - 6 5 8 6
Жол - 31 - 6 5 8 7 1 0 3 2 6
Жол - 32 - 6 5 8 7 1 0 3 4 2 6
Жол - 33 - 6 5 8 7 1 2 6
Жол - 34 - 6 5 8 7 2 6
Жол - 35 - 6 8 5 4 2 6
Жол - 36 - 6 8 5 4 3 0 1 2 6

```
Жол - 37 - 6 8 5 4 3 0 1 7 2 6
Жол - 38 - 6 8 5 4 3 2 6
Жол - 39 - 6 8 5 6
Жол - 40 - 6 8 6
Жол - 41 - 6 8 7 1 0 3 2 4 5 6
Жол - 42 - 6 8 7 1 0 3 2 6
Жол - 43 - 6 8 7 1 0 3 4 2 6
Жол - 44 - 6 8 7 1 0 3 4 5 6
Жол - 45 - 6 8 7 1 2 3 4 5 6
Жол - 46 - 6 8 7 1 2 4 5 6
Жол - 47 - 6 8 7 1 2 6
Жол - 48 - 6 8 7 2 1 0 3 4 5 6
Жол - 49 - 6 8 7 2 3 4 5 6
Жол - 50 - 6 8 7 2 4 5 6
Жол - 51 - 6 8 7 2 6

Enter pernesin basiniz
```

Әдетте барлық маршруттарды іздестірудің `while` циклінде `do_while` циклі мен екі `if` операторы орналасады.

`do_while` циклінде графты «тереңдігі» бойынша жүріп өту жұмысының алгоритмінің көмегімен графтың жаңа төбесі ізделінеді. Егер кезекті төбе оның «соңғы» төбесіне тең болса, онда стек мәліметтері экранға шығады. Стекте маршрут төбелерінің тізімі бар, олар графтың «бастапқы» мен «ақырғы» төбелері арасында болады.

Егер графтың табылған төбесі «жаңа» болса, онда бірінші `if` операторында алгоритм әрекеттері сипатталады.

Егер қаралған сыбайлас төбелер тізімінде «жаңа» төбе болмаса, онда екінші `if` операторында алгоритмнің әрекеттері сипатталады.

13.3 Өзін-өзі тексеру сұрақтары

1 Өз жұмысына стекті пайдаланатын графты жүріп өту алгоритмі қалай аталады?

2 Графты жүріп өту алгоритмінің стегінде не сақталады? Алгоритм өз жұмысында стекті пайдаланады?

3 Өз жұмысында кезекті пайдаланатын графты жүріп өту алгоритмі қалай аталады?

4 Графты жүріп өту алгоритм кезегі нені сақтайды (графты жүріп өту алгоритмі кезекті пайдаланады)?

5 Графты «тереңдігі» бойынша жүріп өту алгоритмінің аяқталу шарты?

6 Графты «көлденеңі» бойынша жүріп өту алгоритмінің аяқталу шарты?

7 Графтың барлық циклдарын іздеу алгоритмінде графтың қаралып кеткен төбелеріне жаңа төбе қасиеті неге «қайтарылады»?

8 Екі берілген төбе арасында барлық маршруттарды іздеу алгоритмінің стегінде не сақталады?

9 Екі берілген төбе арасында барлық маршруттарды іздеу алгоритмінде `do_while` циклі неге қолданылады?

10 Екі берілген төбе арасында барлық маршруттарды іздеу алгоритмінде неге екі шартты `if операторы` қолданылады?

14 ЕРЕКШЕ ЖАҒДАЙЛАРДЫ АЛДЫН АЛУ

14.1 Ерекше жағдайларды алдын алу туралы түсініктер

Алдынғы бөлімде көрсетілген бағдарлама операторларларында мәндерді диалог режімінде дұрыс енгізу керек, яғни сандық мәндерді символдық мәндермен, ал нақты сандарда үтірді нүктемен шатастыруға болмайды.

Ал нақты жағдайларда қате енгізілген мәндер үшін «ерекше жағдайлар» туындайды, ол бағдарламаның жұмысын уақытынан бұрын тоқтатады (бағдарлама тоқтаталады немесе «тұрып қалады»).

Күрделі жобаларды дайындау барысында қателерді болдырмау мүмкін емес, олар әрқашан көріне бермейді, бірақ нәтижелердің қате болуына немесе бағдарламаның жаңылысуына әкеледі. Осындай қателіктер «ерекше жағдайлардың» себебінен болады, мысалы: санды нөлге бөлу, жоқ файлды ашу, массив индексінің белгіленген аралықтан асып кетуі.

Ары қарай жұмысты дұрыс орындай алмайтын болғандықтан бағдарлама өз жұмысын үзетін болса, онда ондай жағдайды ерекше жағдай деп атайды.

C# тілінде ерекше жағдайларды өңдейтін арнайы механизм қарастырылған. Ерекше жағдай орын алған уақытта `Exception` класының немесе оған тиісті кластың арнайы объектісі құрылады (ол ерекше жағдайларды алдын алу деп аталады). `Exception` класы пайдаланушыға орын алған мәселе бойынша хабар береді немесе ол туралы толығырақ мәлімет береді (T типіндегі айнымалының мәнін құрады немесе мәтіндік хабарламаны шығарада).

Ерекше жағдайларды алдын алу бағдарлама жұмысындағы қателіктер туралы хабарлаудың қалыпты тәсілі болып табылады. Бағдарлама жұмысындағы орын алған қателіктер сәйкес типтегі ерекше жағдайды тудырады, осының салдарынан бағдарлама орындалуының қалыпты барысы тоқтатылады және басқару стандартты ерекше жағдай өңдеуішісіне немесе бағдарламаның өзінде қарастырылған ерекше жағдай өңдеуішісіне беріледі.

Ерекше жағдайлардың стандартты өңдеуішісі операциялық жүйеде қарастырылған , ол сәйкес хабарламаны бере отырып бағдарлама жұмысын аяқтайды. Туындаған ерекше жағдай бойынша стандартты сипаттама пайдаланушыға түсініксіз болуы мүмкін.

C# тілінде ерекше жағдайларды өңдеу құралдары бар, яғни бағдарламаның өзінде қарастырылған құралдар. Осындай өңдеу жұмыстарының мақсаты: пайдаланушыға нәтиженің дәйексіздігі туралы хабарлау, қателіктер салдарын жою әрекеті, анық нәтижеге қол жеткізу.

Ерекше жағдайды өңдеуді орындау үшін ерекше жағдай туындайтын мүмкін бөліктің орнына қорғалған блок жазылады. Қорғалған блокта ерекше жағдай туындаған кезде басқару белгілі бір ерекше жағдайды алдын алу өңдеуішісіне тапсырылады, ол пайдаланушыға бағдарламаның қате жұмысын және қателіктер салдарын жоюға әрекет етеді.

Қорғалған блокты жазу пішімі мына түрде жазылады:

try

{ // Қорғалған блок }

catch (T1 e1)

{// Ерекше жағдайдың өңдеуіші. Егер ерекше жағдайдың типі T1 болса іске қосылады.}

...

catch(Tk ek)

{// Ерекше жағдайдың өңдеуіші. Егер ерекше жағдайдың типі Тк болса іске қосылады. }

finally

{// Аяқталу блогы. Кез келген жағдайды іске қосылады. }

мұнда `try, catch, finally` — қызметтік сөздер.

Try кілттік сөзі алдында жазылған блок бағдарлама кодының қорғалған блогы немесе try-блогы деп аталады. Бағдарламашы осы блокқа бағдарлама жұмысының қате орындалуына себеп болатын, яғни ерекше жағдайды тудыратын кодты орналастырады.

Exception ұрпақтары нақты ерекше жағдайларды өңдеуге арналған. Мысалы, System атаулар кеңістігінде ArgumentOutOfRangeException, IndexOutOfRangeException, StackOverlowException және т.б. маңызды кластар анықталған.

Ерекше жағдайды не туындатады? Оны көбінесе операциялық жүйенің ядросы орындайды, бірақ C#-та оны бағдарламаның өзі throw қызметтік сөзімен орындай алады.

Сонымен, try-блогында T типіндегі ерекше жағдай орын алуы мүмкін.

C# тілінде ерекше жағдайды өңдеу үшін арнайы `catch`-блоктары қарастырылған. Әрбір `catch`-блогында `Exception` класының немесе оның ұрпағының формалды параметрі бар және ол ерекше жағдайда «дұрыс» әрекет жасауға арналған.

Бірінші, формалды параметрі T типімен келісілген (T типінде немесе оның ұрпағынан) `catch`-блогында ерекше жағдайды өзі өңдейді, сондықтан `catch`-блоктарының жазылу тәртібі өте маңызды. Алдымен арнайы өңдеушілер жазылуы тиіс. Әмбебап өңдеуіш болып `Exception` класының формалды параметрлері бар `catch`-блогы табылады және ол кез келген T типімен сәйкестендірілген. Әмбебап өңдеуіш ең соңында орналастырылады, өйткені ол кез келген типтегі ерекше жағдайды

қарастыра алады. Catch-блогы switch операторы сияқты орындалады, ал одан кейін басқару қорғалған бөлікке арналған ортақ іс-әрекеттер орындалатын finally блогына беріледі. Finally блогына қорғалған бөлікте ерекше жағдай туындамаса да басқару беріледі, яғни осы блок кез келген жағдайда орындалады. Онда, мысалы, ашық файлдар жабылуы немесе қорғалған бөлікке бөлінген ресурстардың босатылуы мүмкін.

Catch() өңдеушісінің кез келген санын жазуға рұқсат берілген, сонымен қатар finally блогын қолданбауға болады. Осы кластың ең маңызды қасиеттері 14.1-кестеде көрсетілген.

14.1-кесте - Exception класының қасиеттері

Қасиеттер	Сипаты
HilpLink	Қатені сипаттайтын анықтамалық URI файлын қайтарады
Message	Қатенің мәтіндік сипаттамасын қайтарады
Source	Ерекше жағдайды тудыратын объектті немесе қосымшаның атауын қайтарады
StackTrace	Ерекше жағдай туындаған уақытта стек күйін қайтарады
InnerExcepton	Ағымдағы ерекше жағдай бойынша мәліметтерді сақтау үшін қолданады

14.2 Ерекше жағдайларды алдын алу бойынша мысал

5 кездейсоқ бүтін саннан тұратын массивті құру керек, сандар 0-ден 100 дейінгі аралықта болады. Массив индекстері 0-ден 8-ге дейінгі аралықта кездейсоқ құрылады. Егер массив индексі берілген аралықтан асып кетсе, онда ерекше жағдайды өңдеуді қарастыру керек.

Сонымен, индекстің рұқсат етілген мәні 0-ден бастап 4-ке дейінгі аралықта болады. Индекстің басқа мәндерінде бағдарлама жұмысын тоқтатуға әкелетін ерекше жағдай туындайды. Бағдарламаның қорғалған блогында индекс құрылады және ол рұқсат етілген аралықтан асып кетсе, онда сәйкес хабарлама шығады, бірақ бағдарлама жұмысы тоқтатылмайды.

Бағдарлама коды:

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main(string[] args)
 {
  int i, j;
  int[] a = new int[5];
  Random rnd = new Random();
  Console.WriteLine("Kezdeisok bytin sandar massivi: ");
```

```
  i = 0;
  while (i < 5)
  {
  j = 0;
  try
  {
   i++;
   j = rnd.Next(8);
   a[j] = rnd.Next(100);
   Console.WriteLine("a[{0}] = {1} ", j, a[j]);
  }
  catch (IndexOutOfRangeException)
  {
   i--;
   Console.WriteLine("Massivtin indeksi diapozon sheginen
tiskari {0}", j);
  }
  }
  Console.WriteLine("Bagdarlamanin zhymisinin soni");
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

Kezdeisok bytin sandar massivi:

a[1] = 31

Massivtin indeksi diapozon sheginen tiskari 6

a[2] = 36

a[2] = 38

Massivtin indeksi diapozon sheginen tiskari 7

Massivtin indeksi diapozon sheginen tiskari 6

a[3] = 68

a[1] = 63

Bagdarlamanin zhymisinin soni

Қарастырылған мысалда тек бір ғана ерекше жағдай қарастырылған - массив индексінің рұқсат етілген аралықтан асып кетуі. Берілген мысалда бұл бір ғана мүмкін ерекше жағдай, сондықтан бір ғана `catch` блогы қолданылған.

14.3 Өзін-өзі тексеру сұрақтары

1 Ерекше жағдай ұғымы.

2 Ерекше жағдайды алдын алу ұғымы.

3 Бағдарламада ерекше жағдайларды өңдеу механизмі (әдісі) неге негізделген?

4 Операциялық жүйеде қарастырылған ерекше жағдай бойынша стандартты өңдеуіш қалай жауап береді?

5 Бағдарламаның өзінде алдын ала қарастырылған ерекше жағдай бойынша өңдеуіш не үшін қолданылады?

6 Бағдарламаны қорғалған блогы не үшін құрылады?

7 Бағдарламаның қорғалған блогы қандай қызметтік сөзден басталады?

8 Бағдарламада қарастырылған ерекше жағдай бойынша өңдеуіш ерекше жағдайға қалай жауап береді?

9 `Catch`-блоктар не үшін қолданылады?

10 Бір қорғалған блокқа қанша `catch`-блок сәйкес бола алады?

15 VISUAL STUDIO.NET ВИЗУАЛДЫ БАҒДАРЛАМАЛАУ ОРТАСЫ

15.1 Объекті-бағытталған бағдарламалауға кіріспе

Оқу құралының алдыңғы бөлімінде деректер, әдістер, сонымен қатар C# тілінде жазылған бағдарламаның өзі де кластарда орналасуы керектігі жөнінде ескертілген.

Кластардың пайда болуы бағдарламалау технологиясын өзгертті. Егер бұрын құрылымдалған бағдарламалаудың негізгі бірлігі функциялар мен процедуралар болса, кластардың пайда болуы деректерді және сонымен қатар әдістерді біріктіретін бағдарламаның функцияналды аяқталған модульдерін құруға мүмкіндік берді. Осындай бағдарламалардың негізгі бірлігі - кластар (объекттер), кластар арқылы бағдарламалау технологиясы объекті-бағытталған технология деп аталады.

Кітаптың осы бөлімінде Windows қосымшаларды (күрделі бағдарламалық жүйелер) жобалау кезінде объекті-бағытталған технологияны қолдануға байланысты сұрақтар қарастырылады. Объекті-бағытталған бағдарламалау (ОББ) технологияларында кластардың қолданылуы олардың екі қызметті орындай алатынын көрсетеді: бағдарлама модулі немесе деректер типі ретінде қолданылуы.

Құрылымның модульдігі – Windows қосымшаларының негізгі қасиеті. Үлкен бағдарламалық жүйені модульдерге бөлмей дайындау кезінде бағдарламалаудың модульдік технологиясы арқылы дайындаған жүйеге қарағанда уақыт едәуір көп жұмсалады. ОББ-да Windows қосымшалар модульдік принципте әзірленеді, олар модульдің негізі болатын кластардан тұрады. Құрылымның модульдігі – күрделі бағдарламалық жүйелерді әзірлеу процессін жылдамдату бойынша негізгі құрал.

Екінші жағынан класс деректер типі болып келеді. Windows қосымшаларды объекті-бағытталған жолмен әзірлеу деректерге сүйеніп жобалау деп аталатын стильге негізделген. «Жүйелерді жобалау нақты бір есепке сәйкес келетін деректердің абстракцияларын табуға негізделеді. Осындай әрбір абстракциялар класс түрінде құрылады, класс - бағдарламалық жүйе құрылымының архитектуралық бірлігі болатын модуль» [5].

Windows қосымшалардың көпшілігінде кластар екі қызметті де орындайды, сондықтан бағдарламалық жүйенің әрбір модулінің өзінің нақты міндеті бар. C# тілінде деректер типі болатын және модуль қызметіндегі кластар қолданылады. Модуль кластарына, мысалы, Console, Math кластары жатады. Модуль ретіндегі кластар объекттерді құрай алмайды, ал нақтырақ айтқанда осындай кластың бір ғана объектісі болады. Осы модульдің өрістері мен әдістері басқа кластардың әдістеріне қолжетімді болады.

Үлкен бағдарламалық жүйелерді дайындағанда әзірлеу ортасы маңызды рөл атқарады. Бағдарламалау орталары ұсынатын бағдарламалау технологиялары күрделі бағдарламалық жүйелерді дайындау уақытын едәуір қысқартады.

Сондықтан ОББ бағдарламалау технологиясымен Visual Studio.NET бағдарламалау ортасында Windows (Windows Forms Application) қосымшаларын құра отырып танысатын боламыз.

15.2 Windows жүйесінің оқиғаларды басқару ұғымы

Консольді қосымшаларда бағдарлама жұмысы іске қосылғаннан кейін Main() әдісінің операторлары орындала бастайды. Windows үшін жазылған бағдарламардың ерекшелігі – бағдарлама іске қосылғаннан кейін ол Windows-тан келетін хабарларды күтудің шексіз цикліне ауысады.

Хабар дегеніміз - Windows операциялық жүйесінің жүйеде өтіп жатқан оқиғаларға жауабы. Оқиға ретінде компьютер жұмысында кез келген «бейстандарт» жағдайды есептеуге болады, мысалы, пернетақтада пернені басу, тышқан курсорының орнын ауыстыруы, нөлге бөлу, т.б.

Windows жүйесі жауап бере алатын барлық оқиғалар нөмірленген және әрбір нөмерге – «үзу векторына» сәйкес оқиғаға дұрыс жауап беретін арнайы бағдарлама сәйкестендірілген. Мысалы, компьютердің шалғай құрылғыларының драйверлері (пернетақта, тышқан, таймер).

Оқиға пайда болғанда Windows жүйесі оқиға «нөмерін» анықтайды және сәйкес драйверді іске қосады. Драйвер оқиғаны «өңдеп», Windows жүйесіне хабарлама жібереді.

Windows жүйесі жұмысының негізінде оқиғаларды басқару принципі жатыр. Сонымен, жүйе және Windows үшін жазылған барлық қосымшалар іске қосылғаннан кейін пайдаланушы іс-әрекеттерін немесе операциялық жүйенің оқиғаларын күтеді және оларға белгілі тәртіпте жауап қайтарады.

Windows хабары болған оқиға туралы жазба болып табылады. Мысалы, кейбір хабардың құрылымында мыналар болуы мүмкін: бағдарлама терезесінің дескрипторы, хабарлама коды (идентификаторы), анықтаушы параметрлер (мысалы, тышқан меңзерінің x пен y координаттары), хабарламаның құрылу уақыты.

Windows жүйесі қабылдайтын барлық хабарлар бір ғана данада болатын хабарлардың жүйелік кезегіне орналастырылады. Одан кейін жүйелік кезектен хабарлар жеке Windows қосымшаларының хабарлар кезегіне үйлестіріледі. Сонымен қатар, әрбір қосымша үшін өзінің хабарлар кезегі құрылады. Қосымшалардың хабарлар кезегі тек қана жүйелік хабарлардан толықтырылмайды. Кез келген қосымша хабарды кез келген басқа хабарға, сонымен қатар өзіне жібере алады. Әрбір Windows қосымшаның Windows-тан келетін, хабарларды өңдейтін үздіксіз циклі болады. Осы циклдің көмегімен қосымшалар «өзінің» хабарларын алады

және қосымшаның тиісті хабарлар өңдеуішіне жібереді. Қосымшаның әрбір терезесінде хабарларды өңдейтін өз циклі және терезе функциясы (оған қосымша кезегінен алынатын хабарлар жіберіледі) болады.

Әдетте Windows қосымшасының негізгі терезесі болады, онда негізгі элементтер орналасады – меню, батырма, жалаушалар, т.б. Қосымшамен жұмыс істеу барысында пайдаланушылар менюді таңдайды, батырмаларды басады немесе басқа басқару элементтерін қолданады.

Әрбір басқару элементінің өз идентификаторы болады. Мысалы, батырманы басқанда пайда болатын хабар Windows қосымшасының хабарлар кезегіне орналастырылады. Қолданылған басқару элементінен келетін хабарды Windows операциялық жүйесі осы басқару элементінің қосымшасының кезегіне жібереді.

Windows-та құрылатын қосымшаларда (File -> New -> Project -> Windows Forms Application) екі негізгі тип қолданылады: Form, Application.

Application класы қосымшаны басқарады: хабарларды өңдеу циклі (Application.Run();) бар Main() әдісін іске қосады, хабарды алғанда тиісті әрекеттерді орындайды және қосымша жұмысын дұрыс аяқтайды (Program.cs файлы).

Form класы пайдаланушы интерфейсін анықтайды: форма терезесін инициализациялайды, қосымшаны жұмысқа дайындайды (Form1.cs файлы).

Қарапайым Windows қосымшасын құру барысында әрекеттер ретін қарастырайық.

15.3 Visual Studio.Net ортасының негізгі терезелері

Шартты түрде Windows қосымшаларын құру процессі екі кезеңнен тұрады.

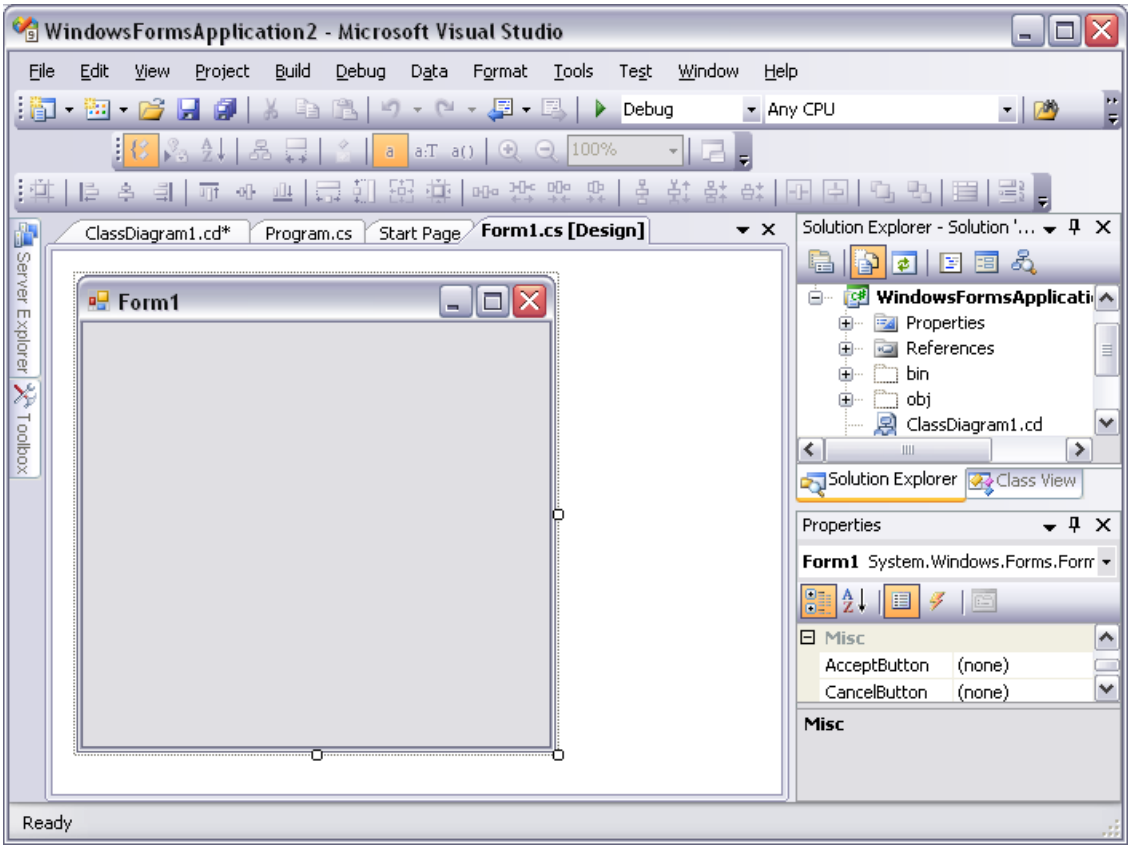
Бірінші кезең – пайдаланушы интерфейсіне арналған визуалды бағдарламалау кезеңі.

Екінші кезеңде қосымшаның хабарлар өңдеуішінің кодын құру керек, яғни Windows жіберетін хабарды алған кездегі қосымшаның жұмысын анықтау.

Windows қосымшаларын құру бойынша бірінші кезеңін орындау үшін Visual Studio .Net ортасын ашу керек (File -> New -> Project -> Windows Forms Application), 15.1-сурет. Жобаны дайындаған кезде жұмыс үстелінде бума атауын көрсету керек, онда барлық файлдар сақталады.

15.1-суретінің ортасында System.Windows.Forms атау кеңістігіне тиісті Form 1 терезесі орналасқан. Жобаны іске қосу үшін F5 пернесін басу керек немесе ортаның Debug жұмыс режімінің Start командасын таңдау керек. Іске қосылған қосымшаның терезесі 15.2-суретте көрсетілген.

Visual Studio .Net ортасының жеке терезелерінің қызметін толығырақ қарап шығайық.

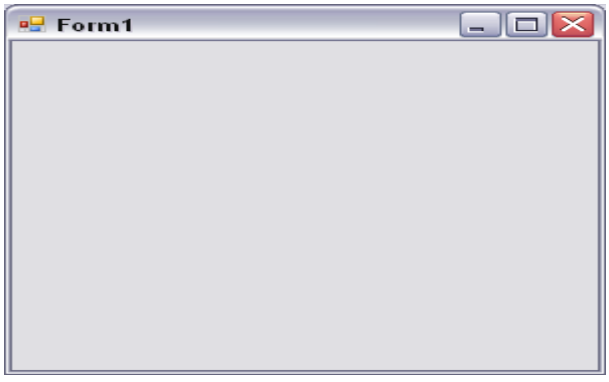


15.1-сурет – Visual Studio .Net ортасы

Form1 терезесі 4 беттен тұрады, олар жобаны редакциялаудың әртүрлі режімінде қолданылады, мысалы, Form1.cs[Design] элементтерді формада орналастыру үшін қолданылады, ал Program.cs терезесі бағдарлама кодын редакциялауда қолданылады.

Properties терезесінде басқару элементтерінің қасиеттері анықталады. Осы терезенің беттері категориялар, қасиеттер немесе оқиғалар бойынша топтастырылған.

Server Explorer терезесі (15.1-сурет, сол жақта, жиналған түрде көрсетілген) – сіздің компьютеріңіздегі, сервердегі деректер көздеріне қол жеткізу үшін қолданылады.



15.2-сурет – бағдарламаның жұмыс терезесі

Toolbox терезесінде (15.1-сурет, сол жақта, жиналған түрде) түрлі басқару элементтері болады.

Solution Explorer терезесі (15.1-сурет, оң жақта) жоба файлдарын қарауға және редакциялауға мүмкіндік береді. Онда жобаны файлдар және кластар бойынша қарауға болады.

Әдетте бағдарлама іске қосылғаннан кейін қателер терезесі пайда болады, онда бағдарлама кодында қате кеткен жолдың нөмірі шығады.

Visual Studio .Net ортасымен жұмыс жасаудың толық сипатттамасы А.В. Фролов, Г.В. Фролов «Визуальное проектирование приложений C#» кітабында берілген [3].

15.4 C# жобасын құру бойынша негізгі құрылымдық элементтер

C# тілі бағдарламалаудың объекті-бағытталған тіл деп аталады, онда класс негізгі ұғым болып табылады.

Класс дегеніміз – тип – көптеген объекттерді сипаттау үлгісі. Объект дегеніміз – класс типіндегі айнымалы, ол бағдарлама орындалу барысында динамикалық түрде құрылады, компьютер жадысында сақталады және бағдарлама аяқталғаннан кейін компьютер жадысынан өшіріледі.

Жобаны дайындаған кезде әдетте бірнеше класс дайындалады, бірақ жобаның жұмысы барысында бір-бірімен күрделі байланысқан жүздеген объекттер динамикалық түрде пайда болады.

FCL кітапханасының негізгі класы Object класы болып табылады, ол барлық кітапханалық және әзірлеуші дайындайтын кластардың түп тегі болып келеді.

Атаулар кеңістігі – бір тақырыптағы немесе әзірлеуші құрған белгілі бір кластар жиынтығының бірлесуі. Атаулар кеңістігіндегі кластардың атауы бірегей болуы керек. Әр түрлі атаулар кеңістігіндегі кластар атауы бірдей болуы мүмкін. Кластың толық атауы атаулар кеңістігінің атауынан, нүкте символынан және класс атауынан тұрады.

Атаулар кеңістігі FCL кітапханасының құрылымын жүйелейді. Егер барлық атаулар кеңістігін иерархиялық бұтақтар класы ретінде қарастырсақ, онда System кеңістігі, оның ішіндегі Object класы осы бұтақтың түпкі класы болады.

Жобаны құрастыру кезеңіндегі қосымша және компиляция бірлігі ретінде қарастыруға болады. Жоба компиляциясының нәтижесі құрастыру болады. Әрбір жобада бір немесе бірнеше атаулар кеңістіктері болады. Жобаны дайындайтын алғашқы кезеңінде белгіленген тип бойынша автоматты түрде қосымша қаңқасы құрылады, ол FCL кітапханасының құрамына кіретін кластардан тұрады. Егер "Windows Forms Application" типіндегі жоба құрылса, онда қосымша қаңқасына Form1 класы – кітапханалық Form класының мұрагері кіреді.

Жобаға автоматты түрде құрылған және жоба әзірлеушісі құрған барлық кластары бар файлдар кіреді. Сонымен қатар, жобада бағдарламаның жұмысы барысында қолданылатын FCL кітапханасындағы атаулар кеңістігіне сілтеме болады. Жобада өзіне қосылатын барлық DLL-ге және басқа жобаларға сілтемелер болады.

«Жобаның жұмыс жасауы үшін оған керекті ресурстар мен орнатулар қосылады. Жобаны құрастыру сипаттамасын сақтайтын файл жобаның бір бөлігі болып келеді.

Таңдалған типіне байлысты жоба орындалатын және орындалмайтын болып бөлінеді. Орындалатын жобаларға, мысалы, Console немесе Windows типіндегі жобалар жатады. Орындалатын жоба қаңқасын құрғанда оған Main атаулы тұрақты әдісі бар класс қосылады. Осындай жобаны компиляциялау нәтижесінде PE-файл (Portable Executable file) – exe-ні анықтайтын, орындалатын файл құрылады. Ескерту, PE-файл компьютерде тек Framework .Net орнатылса ғана орындалады.» [5]

Орындалмайтын жобаларға, мысалы, Dll типіндегі жобалар жатады.

Құрастыру (сборка) – жоба компиляциясының нәтижесі. Ол нұсқаның нөмерімен белгіленген бір немесе бірнеше файлдар коллекциясынан тұрады. Әрбір құрастыру компьютерде біртұтас болады. Бағдарламашы жобамен жұмыс істесе, ал CLR құрастырулармен жұмыс істейді. Құрастыру қауіпсіздік сұрақтарын шешеді, өйткені онда өзіне керекті ресурстардың сипаттамасы мен элементтерді пайдалану құқықтары бар. Әрбір құрастыруда манифест болады, манифест құрастырудан және оның элементтерінің толық сипаттамасынан, қажетті ресурстардан, басқа құрастыруларға сілтемелерден тұрады. CLR-дің осы сипаттамасының арқасында құрастыруды өрбіту, аралық код трансляциясы мен оның орындалуы үшін басқа қосымша ақпарат қажет емес. Манифест құрастыруды идентификациялайды, құрастыруды орындау үшін керекті файлдарды спецификациялайды, құрастыруды құрайтын типтер мен ресурстарды спецификациялайды.

Visual Studio.NET 2008 ортасында дайындалатын әрбір жоба Шешім – Solution деп аталатын белгілі бір қоршамға орналастырылады. Шешімде, әдетте, ортақ тақырып бойыша байланыстырылған бірнеше жоба болуы мүмкін. Мысалы, бір Шешімге үш жобаны орналастыруға болады: әзірленген кластары бар DLL, консольді жоба, Windows-та басқарылатын жоба.

Жаңа жоба құрылу барысында оны бар Шешімге орналастыруға болады немесе ол үшін жаңа Шешім құрылады. [5]

15.5 Бірінші бағдарламаны құру мысалы

Windows жүйесінде жұмыс істейтін қосымшаның формасын қолдану мысалын қарастырайық. Мысал ретінде үшбұрыш периметрін есептеу есебі қарастырылады.

15.1-есеп. диалог режімінде үшбұрыш қабырғаларын беру және оның периметрін есептеу керек. Үшбұрыш қабырғаларын берілгеннен кейін мына тексерістерді орындау керек: үшбұрыш қабырғалары нөлден үлкен болуы керек және үшбұрыштың кез келген екі қабырғаларының қосындысы үшінші қабырғадан үлкен болуы керек. Қосымша кодына түсініктемелерді қолдану керек.

Біз Toolbox терезесінен стандартты үш басқару элементін пайдаланамыз: тұрақты мәтін (Labe1) элементі, мәтінді енгізу-шығару өрісі (TextBox**)**, командалық батырма (Button).

– тұрақты мәтін;

– мәтінді енгізу-шығару өрісі;

– командалық батырма.

Түсіндіретін сөздерді жазу үшін төрт тұрақты мәтін қолданылады.

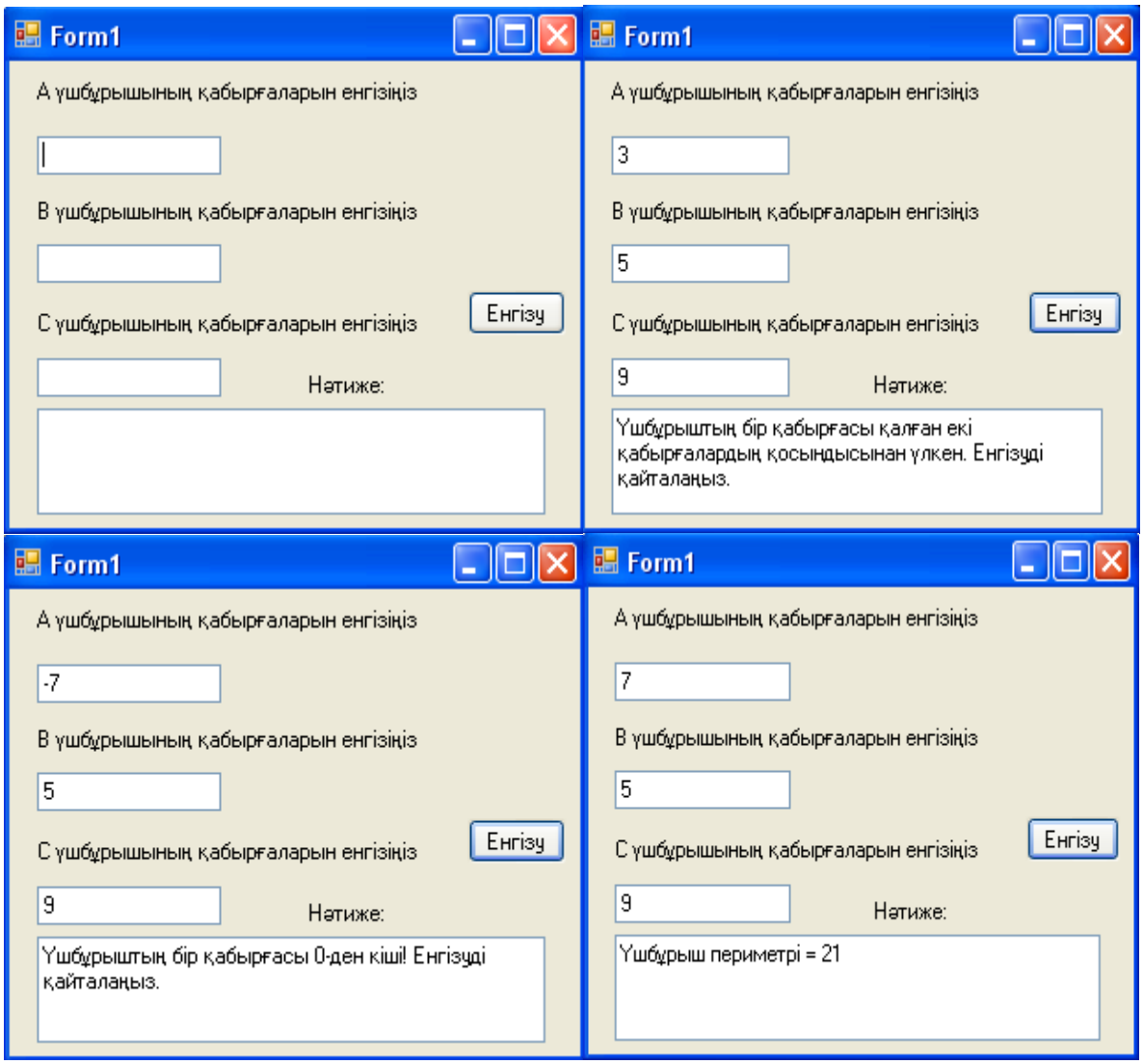
Үш мәтінді енгізу, нәтижені шығарудың бір өрісі және бір командалық батырма қолданылады.

Визуалды бағдарламалау процессінде формаға Toolbox терезесінен керекті басқару элементі көшіріледі және белгілі бір орынға орналастырылады. Әдетте Toolbox терезесі «жиналған» күйде болады. Оны «ашу» үшін тышқанның оң жақ пернесімен Toolbox панелін басу керек,  элементі (оны басу керек) арқылы экранның белгілі бір орнына орнықтыруға болады. Жұмыс аяқталғаннан кейін  элементінің көмегімен Toolbox терезесін «жинауға» болады.

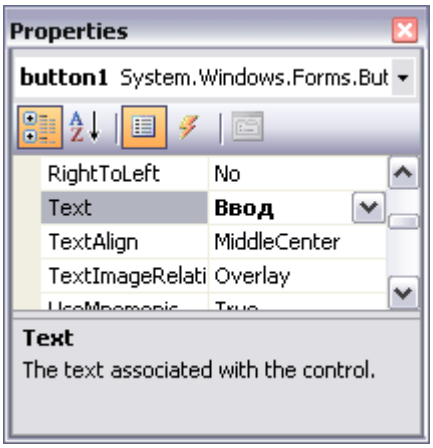
Визуалды бағдарламалау процесінде басқару элементтерінің кейбір қасиеттерін өзгерту керек, мысалы, тұрақты мәтін мен батырманың Text қасиеті өзгертілді (15.3-сурет). Ол үшін Properties терезесін пайдалану керек (15.4-сурет).

Енгізу өрістері мен шығару өрісінің айырмашылығы бар, шығару өрісінің `Multiline` қасиеті `Multiline = true` тең. Барлық басқару элементтерінде `Text` қасиеті қолданылды.

«Ввод» Батырмасын басу бойынша хабарды өңдеуші әдісін құру үшін визуалды бағдарламалау кезеңінде осы батырманы екі рет басса жеткілікті. Бос `private void button1_Click(object sender, EventArgs e)` хабар өңдеушісіне кодты жазамыз: үшбұрыш қабырғаларын мәндерін диалог режімінде беру және олардың үшбұрыш шарттарына сай келуі.



15.3-сурет – «Үшбұрыш» қосымшасының терезесі



15.4-сурет – button1 элементінің Properties терезесі

Program.cs файлының коды:

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
 static class Program
 {
 /// <summary>
 /// The main entry point for the application.
 /// </summary>
 [STAThread]
 static void Main()
 {
 Application.EnableVisualStyles();
 Application.SetCompatibleTextRenderingDefault(false);
 Application.Run(new Form1());
 }
 }
}
```

Form1.cs файлының коды:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsFormsApplication1 //Zadacha15_1
{
 public partial class Form1 : Form
 {
 public Form1()
 {
  InitializeComponent();
 }
 private void button1_Click(object sender, EventArgs e)
 {
  int a, b, c, p;
  a = Convert.ToInt32(textBox1.Text);
  b = Convert.ToInt32(textBox2.Text);
  c = Convert.ToInt32(textBox3.Text);
  p = a + b + c;
  if (a > 0 && b > 0 && c > 0)
  if (a + b > c && a + c > b && b + c > a)
   textBox4.Text = "Үшбұрыш периметрі = " + p.ToString();
  else
  {
   textBox4.Text = "Үшбұрыштың бір қабырғасы қалған екі
қабырғалардың қосындысынан үлкен. Енгізуді қайталаңыз.";
  }
  else
```

```
        {
        textBox4.Text = "Үшбұрыштың бір қабырғасы 0-ден кіші!
Енгізуді қайталаңыз.";
        }
       }
       private void Form1_Load(object sender, EventArgs e)
       {
       }
       }
      }
```

Орта автоматты түрде атаулар кеңістігін құрады. Оларды толығырақ қарастырайық.

System атаулар кеңісті базалық және іргелі анықтамалардан тұрады, олар: деректер типі, оқиғалар, оқиғалар өңдеуіштері, т.б.

System.Collections атаулар кеңістігінде кластар анықталған, олар массивтерді, тізімдерді, сөздіктерді, хэштерді анықтайтын контейнерлер қызметін атқарады.

System.ComponentModel кеңістігіндегі кластар қосымшаның компоненттері мен басқару элементтерінің белгілі бір тәртіптегі қызметін орындау үшін қолданылады.

System.Data класы ADO.NET интерфейсі арқылы деректер базасымен жұмыс істейтін қосымшалар үшін керек.

System**.**Drawing кеңістігі графикалық құрылғылар интерфейсіне (Graphics Device Interface, GDI) қол жеткізу үшін керек, нақтырақ айтсақ, оның толықтырылған GDI+ версиясы үшін. Осы кеңістіктегі кластар қосымша терезесінде сызықтарды, екі өлшемді пішіндерді, кескіндерді, басқа да графикалық объекттерді салу үшін керек.

System.Windows.Forms кеңістігінде формалардың жұмыс тәртібін орындайтын кластар анықталған.

Қосымшаға нақтысында System және System.Windows.Forms екі кеңістігі қажет, ал қалған атаулар кеңістігі қосымшаға қажетінше қосылады.

Қосымша ретке келтірілгеннен (после отладки) кейін барлық файлдарды сақтау керек (менюде File->Save All командасын таңдау керек).

Бағдарламалаудың визуалды ортасы кішігірім қосымша үшін де 10-нан аса файлдар мен бумаларды дайындайды. Жұмыс үстеліндегі бірінші бағдарламаның 1_1_treygolnik бумасында WindowsFormsAplication1 бумасы бар. Оның ішінде WindowsFormsAplication1 атты бума мен WindowsFormsAplication1.csproj-ды редакциялау үшін шақырылатын жоба файлы бар. Ал WindowsFormsAplication1 бумасында тағы bin, obj, Properties бумалары мен бірнеше файлдар (бағдарлама коды – Program.cs, форма коды – Form1.cs) бар. Осы бумада Form1 формасы бойынша файлдың сырт пішінін сақтайтын ресурстық файл және форма мен онда орналасқан барлық басқару элементтер «қасиеттерінің» мәндерін сақтайтын Form1.Designer.cs файлы орналасады.

Visual Studio .Net ортасымен жұмыс жасау үшін әзірше бізге тек форма кодының файлы – Form1.cs қажет және осы файлды ғана өзгертуге болады, басқа файлдардың атауы мен қасиеттерін берілген күйде қалдыру керек.

15.6 Өзін-өзі тексеру сұрақтары

1 Компьютер жұмысындағы оқиғалар ұғымы?

2 Windows жүйесі оқиғаларды қалай «ажыратады»?

3 Windows жүйесі оқиғаның пайда болуы туралы ақпаратты алғанннан кейін не істейді?

4 Драйвер ұғымы.

5 Хабар ұғымы .

6 Хабар неден тұрады?

7 Windows жүйесі драйверден қабылдап алатын хабарларды қайда жібереді?

8 Әрбір қосымшада Windows-тан келетін хабарларды өңдеу циклі не үшін қолданылады?

9 Қосымшаның қандай әдісі Windows-тан келетін хабарларды өңдеу циклін жүзеге асырады?

10 Form класы қандай мақсатпен қолданылады?

16 БАСҚАРУ ЭЛЕМЕНТТЕРІ

16.1 Формаларды визуалды жобалау технологиясы

Визуалды бағдарламалау ортасында Windows.Forms.Designer (форма құрастырушысы немесе дизайнері) – интерактивті режімде басқару элементтерін орналастыра отырып форманы визуалды жобалауға мүмкіндік беретін құрал. Windows.Forms.Designer артықшылығы мынада: сіз формада басқару элементтерін керекті орынға орналастыра аласыз, сонымен қоса оның қасиеттерінің нақты мәндерін анықтаудың қажеті жоқ, мысалы, өлшемнің, орналасудың сандық мәні. Көптеген қасиеттердің мәндерін автоматты түрде форма құрастырушысы арқылы беруге болады, бірақ форма құрастырушысы Сізге белгілі дәрежеде ғана көмектесе алады.

Windows қосымшасында (консольді қосымшаға қарағанда) форма құрастырушысы автоматты түрде бірнеше .cs кеңейтуі бар кластарды құрайды, мысалы, Form1.cs және Program.cs.

C#-та кластар синтакстік тұрғыдан бірнеше бөліктен тұра алады, олардың әрқайсысы “partal” (бөліктік) кілттік сөзінен басталады. Бір кластың сипаттамасын бөліктерге бөлу мүмкіндігі үлкен класпен жұмыс істеуді жеңілдетеді. Кластың әрбір бөлігі жеке файлдарда, өз атауымен сақталады. Мысалы, алдыңғы бөлімдегі мысалда автоматты түрде Form1-дың .cs кеңейтуі бар екі файл – Form1.cs және Form1.Designer.cs құрылады. "Form1.cs" файлында сақталатын, әзірлеушіге арналған Form1 класының бірінші бөлігінде автоматты түрде құрылатын басқару элементтерінің оқиғалар өңдеуішісі орналасады. Формалармен жұмыс істеуге негізделген бағдарламалаудың осындай технологиясы визуалды, оқиғалармен басқарылатын бағдарламалау технологиясы деп аталады. Мысалы, алдыңғы бөлімде қарастырылған Form1.cs файлынан үзінді:

```
namespace WindowsFormsApplication1
{
 public partial class Form1 : Form
 {
 public Form1()
 {
  InitializeComponent();
 }
 private void button1_Click(object sender, EventArgs e)
 {
  int a,b,c,p;
  a = Convert.ToInt32(textBox1.Text);
  b = Convert.ToInt32(textBox2.Text);
  c = Convert.ToInt32(textBox3.Text);
 p = a + b + c;
 . . .
```

Form1 класының екінші бөлігі "Form1.Designer.cs" файлында орналасады. Кластың осы бөлігін автоматты түрде форма құрастырушысы

толтырады. Визуалды жобалау барысында формаға әр түрлі басқару элементтерін орналастырып, олардың қасиеттерін өзгертіп, оқиғалар өңдеуішілерін анықтағанда форма құрастырушысы осы әрекеттерді сәйкес кластағы объекттер әрекеттеріне трансляциялайды, сәйкес кодты құрайды және осы кодты Form1 класына орналастырады.

Форма құрастырушысының жұмысына араласудың, Form1 класының коды түзеуді қажет етпейдә жоқ. Алайда оның жұмысы мен оның құрған коды туралы білу керек.

Төменде алдыңғы бөлімде қарастырылған Form1.Designer.cs файлынан үзінді көрсетілген:

```
namespace WindowsFormsApplication1
{
 partial class Form1
 {
 /// <summary>
 /// Required designer variable.
 /// </summary>
 private System.ComponentModel.IContainer components = null;
            . . .
private void InitializeComponent()
 {
  this.label1 = new System.Windows.Forms.Label();
  this.label2 = new System.Windows.Forms.Label();
  this.label3 = new System.Windows.Forms.Label();
  this.label4 = new System.Windows.Forms.Label();
 this.button1 = new System.Windows.Forms.Button();
            . . .
     // label1
  //
  this.label1.AutoSize = true;
  this.label1.Location = new System.Drawing.Point(12, 9);
  this.label1.Name = "label1";
  this.label1.Size = new System.Drawing.Size(174, 13);
  this.label1.TabIndex = 0;
  this.label1.Text = "A ұшбұрышының қабырғаларын енгізіңіз";
. . . и т.б.
```

Жобада автоматты түрде құрылатын Program.cs класында статикалық Main() әдісі болады. Қосымша іске қосылғанда Windows жүйесі Main() әдісін іздейді және ондағы нұсқауларды орындай бастайды. Main() әдісін әдетте бағдарламаға кіру нүктесі деп атайды.

Төменде алдыңғы бөлімде қарастырылған мысалдың Program.cs файлы көрсетілген:

```
namespace WindowsFormsApplication1
{
 static class Program
 {
```

```
 /// <summary>
 /// The main entry point for the application.
 /// </summary>
 [STAThread]
 static void Main()
 {
  Application.EnableVisualStyles();
  Application.SetCompatibleTextRenderingDefault(false);
  Application.Run(new Form1());
 }
 }
}
```

Консольді қосымшаларда Main() әдісінің денесі бос болады және онда кодты жоба әзірлеушісі жазу керек. Ал Windows қосымшаларында Main() әдісі керекті нұсқаулармен толықтырылып қойылады және әдетте әзірлеуші оны өзгертпейді. Автоматты құрылған Main() әдісі нені орындайды? Ол FCL кітапханасына тиісті Application класының үш статикалық әдістерін кезек-кезек шақыра отырып жұмыс істейді.

Application.EnableVisualStyles(); әдісі Windows XP стилінде құрылатын қосымша компоненттерін ұсынады.

Application.SetCompatibleTextRenderingDefault(false); әдісі true – GDI+, false – GDI мәтінін экранға шығару механизмін анықтайщды.

Application.Run(new Form1()); әдісі `Program` класының негізгі әдісі және Main() әдісінде ол жалғыз болып келеді.

Негізгі жұмысты Run әдісі орындайды, оны шақыру процесінде Form1 класының объектісі құрылады және форма – объекттің визуалды көрінісі ашылады. Формада орналасқан барлық басқару элементтерінің инициализациясы аяқталғаннан кейін Windows хабарларын өңдеу циклі қосылады, қосымша Windows-тан хабарларды күтеді. Ал пайдаланушы болса, мысалы, мына әрекеттерді орындай алады: форма өрістеріне деректерді енгізу, батырманы басу.

Визуалды бағдарламалауды орындау үшін Toolbox панелінде орналасқан басқару элементтерінің қызметін қысқаша қарастырып өту керек.

16.2 Toolbox панеліндегі басқару элементтері

Басқару элементтер немесе компоненттер формаға ToolBox панелінен орналастырылады (View ▸ ToolBox). Оқу құралының осы бөлімінде ToolBox панеліндегі қарапайым басқару элементтері қысқаша сипатталған.

Басқару элементтерімен танысуды формада үнемі қолданылатын Label элементінен бастайық.

16.2.1 Label – таңба (метка). Таңба формада мәтінді оналастыру үшін керек, мәтін элементтің Text қасиетінде сақталады. Мәтіннің қарпін (Font

қасиеті), түсін (BackColor қасиеті), !!!!(ForeColor) мәтінді туралауды (TextAlign) анықтауға болады. Таңба өз өлшемін автоматты түрде өзгерте алады (AutoSize = True қасиеті). Онда суретті (Image қасиеті) орналастыруға, оның мөлдірлігін (BackColor қасиетіне Color.Transparent мәнін беру керек) анықтауға болады .

Label басқару элементі енгізу фокусын қабылдамайды (тышқан пернесін, пернетақтаны басқанда оқиға өңдеуішісін құрай алмайды).

16.2.2 Button – батырма. Button басқару элементі енгізу фокусын қабылдай алады, сонымен қатар тышқанды шерту – негізгі оқиға болып есептеледі (Click).

Егер форманың Accept Button қасиетіне батырма атауын көрсетсек, онда Enter пернесін басқанда Click оқиғасы шақырылады, ондай батырманың жиектемесі болады. Ал егер форманың Cancel Button қасиетіне батырма атауын көрсетсек, онда Esc пернесін басқанда батырманың Click оқиғасы шақырылады. Батырма мәтінінің өлшемін өзгертуге, фонның түсі мен суретін анықтауға болады. Батырмада суретті Image немесе ImageList, ImageIndex қасиеттері анықтауға өзгертуге болады.

Батырмалар көбінесе диалогтық терезелерде қолданылады. Диалогтық терезенің модальдық қасиеті болады, яғни осы терезе жабылғанша қосымшамен ары қарай жұмыс жасау тоқтатылады. Ақпаратты OK (Yes) батырмасын басып растауға немесе терезені жабу батырмасы арқылы оның күшін жоюға болады, нәтижесінде терезе жабылады. Терезенің жабылғаны туралы мәліметті сақтау үшін батырмада DialogResult қасиеті анықталады, бұл қасиет стандартты мәндерді System.Windows.Forms кеңістігінде анықталған DialogResult тізімінен қабылдай алады. DialogResult тізімінің мәндері 16.1-кестеде көрсетілген.

16.1-кесте – DialogResult тізімінің мәндері

Мәні	Сипаттамасы	Мәні	Сипаттамасы
None	Терезе жабылмайды	Ignore	Ignore батырмасы басылды
OK	OK батырмасы басылды	Yes	Yes батырмасы басылды
Cancel	Cancel батырмасы басылды	No	No батырмасы басылды
Abort	Abort батырмасы басылды	Retry	Retry батырмасы басылды

16.2.3 TextBox енгізу өрісі. TextBox Компоненті мәтінді енгізуге, редакциялауға мүмкіндік береді, мәтін Text қасиетінде сақталады. Жолдардың санына шек қойылмайды (шамамен 32 000 символға дейін), осы компонент арқылы масканы анықтап парольді енгізуге болады (PasswordChar қасиеті).

Text қасиеті тек бір жолды ғана енгізуге, ал Lines қасиеті бірнеше жолды енгізуге мүмкіндік береді. Соңғы қасиет бойынша элементтергі жолдар массив түрінде сақталады, сондықтан оларға индекс түрінде қол жеткізуді ұйымдастыруға болады.

Бірнеше жолды енгізу және шығару үшін Multiline, ScrollBars, WordWrap қасиеттері қолданылады. ReadOnly қасиеті тек оқу рұқсатын орнатады.

Элементтің тазарту (Clear), ерекшелеу (Select), буферге көшіру (Copy), буферден кірістіру (Paste), т.б. әдістері бар. Компонент көптеген оқиғаларды өңдей алады, олардың негізгілері - KeyPress және KeyDown.

16.2.4 ListBox — тізім. ListBox компоненті бір немесе бірнеше пункттерді таңдау мүмкіндігін ұсынады. SelectMode қасиетінде келесі бірнеше мәндер болуы мүмкін: None — пункттерді таңдауға рұқсат жоқ; One — бір пунктті ғана таңдауға болады; MultiSimple — бірнеше пункттерді таңдауға болады; MultiExtended — Shift және Ctrl пернелерін басып тұрып бірнеше пункттерді таңдауға болады: егер Chift пернесі басылып тұрса, пункттердің шексіз диапазоны таңдалып алынады; егер Ctrl пернесі басылып тұрса, пункттердің еркін (үздіксіз болуы шарт емес) диапазоны таңдалып алынады. Егер MultiColumn қасиетіне True мәні меншіктелсе, тізім пункттері бірнеше бағанада орналаса алады, ал ColumnWidth қасиеті бағана жалпақтығын анықтайды. Егер бағана компоненттің ені бойынша шығып кететін болса, онда автоматты түрде көлденең айналдыру жолағы (полоса прокрутки) қойылады.

16.2.5 RadioButton ауыстырып-қосқышы. Ауыстырып-қосқыш пайдаланушыға бірнеше нұсқалардың ішінен бірін таңдауға мүмкіндік береді, сондықтан әдетте ауыстырып-қосқыштар топқа біріктіріледі. Егер олардың бірі белгіленсе (Checked қасиеті), қалғандары автоматты түрде босатылады. Бағдарламашы мәтіннің стилін, түсін өзгерте алады. ауыстырып-қосқышы үшін фон түсі мен фонның суретін анықтауға болады. Ауыстырып-қосқыштарды тікелей формаға орналастыруға болады. Егер формада ауыстырып-қосқыштардың бірнеше тобын орналастыру керек болса, онда оларды Group немесе Panel компонентінің ішіне орналастырады. Appearance қасиеті ауыстырып-қосқыштың бейнесін анықтайды: батырма (Button) немесе қалыпты (Normal) түрде бейнеленеді.

16.3 Басқару элементтерін қолдану мысалы

Осы бөлім үшін синус, косинус және тангенс тригонометриялық функцияларын есептеу мысалы таңдап алынды. Айнымалылардың мәндері (бұрыш радианмен берілген) бағдарламаның диалог режімінде беріледі. Сонымен қатар диалог режімінде есептелетін функцияның аты және монитор экранына шығару форматының разрядтар саны – есептеу дәлдігі

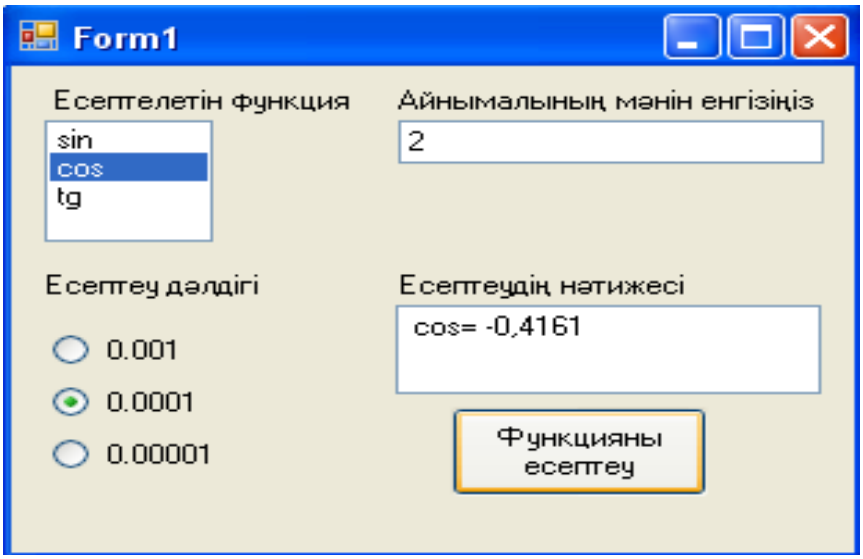
беріледі. Осы есепті дайындау үшін жобада келесі басқару элементтері қолдынылған: Label, Button, Panel, RadioButton, ListBox және TextBox.

Form1.cs файлының коды:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
 public partial class Form1 : Form
 {
 public Form1()
 {
 InitializeComponent();
 }
 private void button1_Click(object sender, EventArgs e)
 {
 string ff = "F3";
 string fu = "sin";
 Double x=0;
 // бұрыштың мәнін анықтаймыз
 x = Convert.ToDouble(textBox2.Text);
 //Выбор функции
 if (listBox1.SelectedIndex == 1) fu = "cos";
 if (listBox1.SelectedIndex == 2) fu = "tn";
 // есептеу ділдігі
 if (radioButton1.Checked)
 {
 ff = "F3";
 }
 else
 if (radioButton2.Checked)
 {
 ff = "F4";
 } else
 if (radioButton3.Checked)
 {
 ff = "F5";
 };
 switch (fu)
 {
case "sin": textBox1.Text = " sin= " +
                     Math.Sin(x).ToString(ff); break;
case "cos": textBox1.Text = " cos= " +
                     Math.Cos(x).ToString(ff); break;
case "tn": textBox1.Text = " tn= " +
     (Math.Sin(x) / Math.Cos(x)).ToString(ff); break;
```

```
    }
    }
    }
}
```

Қосымшаның жұмысы 16.1-суретте көрсетілген.



16.1-сурет – Функцияны есептеу бойынша қосымшаның жұмысы

Бағдарлама жұмысына қосымша түсіндірме берудің қажеті жоқ. Басқа басқару элементтері, яғни меню, диалогтық терезелер, суреттер, т.б. оқу құралының келесі бөлімдерінде қарастырылатын болады.

16.4 Өзін-өзі тексеру сұрақтары

1 NET Платформасының Windows.Forms.Designer не үшін қолданылады?

2 Форма класының сипаттамасында “partal” қызметтік сөзі нені білдіреді?

3 Хабарламалар өңдеуіштері форма класына арналған сипаттаманың қандай бөлігінде орналасады?

4 Windows қосымшасының орындалуы қандай әдістен басталады?

5 Бағдарлама іске қосылғанда қандай әдіс Form1 класының объектісін құрайды?

6 Label басқару элементінің қызметі?

7 Label басқару элементінің қандай қасиеті форма терезесіне ақпаратты шығара алады?

8 Label басқару элементінің қандай қасиеті арқылы оның «мөлдірлігін» анықтауға болады?

9 Батырманың қандай қасиеті арқылы оған суретті орналастыруға болады?

10 Диалогтық терезе жабылғанша қосымшамен ары қарай жұмыс жасауға кедергі келтіретін диалогтық терезені қалай атайды?

17 C# ТІЛІНІҢ ГРАФИКАЛЫҚ ИНТЕРФЕЙСІ

17.1 System.Drawing атаулар кеңістігі

Microsoft .NET Framework шеңберінде жұмыс жасауға арналған C# (GDI+) қосымшасының графикалық интерфейсі атаулар кеңістігінде біріктірілетін кластар жиынынан тұрады. C# тілінің GDI+ негізгі атаулар кеңістігі болып System.Drawing есептеледі.

Осы атаулар кеңістігіндегі кластар «сурет салуға» арналған объекттер мен құралдар тізбегін анықтайды.

System.Drawing атаулар кеңістігіндегі ең жиі қолданылатын класс Brush класы болып есептеледі (Brushes, SolidBrush, т.б.). Brush объектісі (бояу жаққыш) геометриялық фигураның ішкі аумағын толтыру үшін қолданылады.

Brush типі — абстрактілі базалық класс, қалған типтер Brush типінен туындайды және басқа көптеген мүмкіндіктерді ұсынады.

Pen (Pens, SystemPens). Pen (қаламұш) — класс объектісі, оның көмегімен түзу және қисық сызықтарды сызуға болады. Pen класында статикалық қасиеттер жиыны анықталған (мысалы, түсін орнату)

Font (FontFamily). Font типіндегі объекттер қаріп сипаттамаларын анықтайды (атауы, өлшемі, жазылуы, т.б.). FontFamily бір топқа жататын, бірақ өзгешеліктері бар қаріптер жиынын ұсынады.

Graphics. Осы класс монитор экранына геометриялық фигураларды, суреттерді, мәтіндерді шығару үшін қасиеттер мен әдістер жиынын анықтайды.

Region. Бұл класс геометриялық фигураның алатын орнын анықтайды.

Point (PointF). Осы құрылымдар нүкте координаттарымен жұмыс жасауды қамтамасыз етеді. Point int типіндегі мәндермен, ал PointF – float типіндегі мәндермен жұмыс істейді.

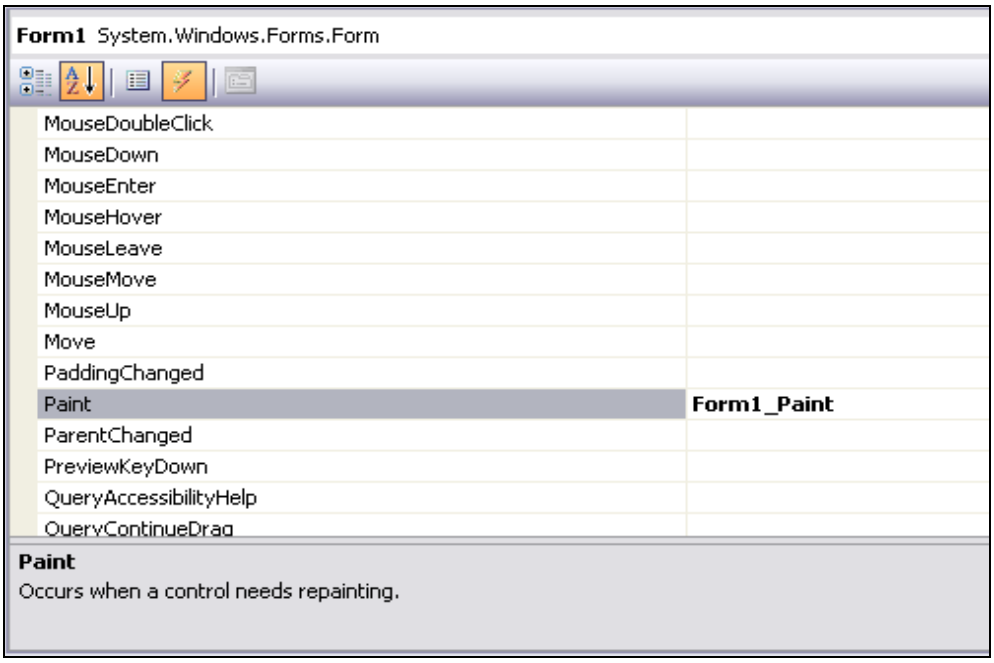
System.Drawing атаулар кеңістігінде Icon, Image, Color, Bitmap кластары және басқа да кластар бар.

17.2 Graphics класы

C# тілінде «сурет салудың» негізгі класы Graphics класы болып есептеледі. Ол бағдарламаның формасында графикалық ақпаратты шығаруға арналған. Қосымша терезеде бір бейнені салу үшін Graphics класының объектісін қолдануы немесе құруы керек. Одан кейін осы объекттің әдістері мен қасиеттерін пайдалана отырып, қосымша терезесінде түрлі фигураларды, мәтіндерді салу мүмкін болады.

Қосымшада Graphics класының объектісін құрамас бұрын «сурет салу» бойынша оқиғалар өңдеуішін анықтап алу керек.

Windows жүйесінде терезенің орын ауыстыруы мен өлшемінің өзгеруін арнайы WM_PAINT хабары «бақылайды». Ол қосымшаны керекті уақытта терезені қайта салу қажеттігі жөнінде хабардар етіп отырады. Тереземен кез келген жұмыс – оның монитор экранында орын ауыстыруы, өлшемдерінің өзгеруі, т.б. Windows жүйесінің терезе «суретін қайта салу» талабымен қатар жүреді. Қосымшада WM_PAINT оқиғалар өңдеуіші осындай хабарды алғаннан кейін терезе емесе оның бөлігінің суретін қайта салуы тиіс. Форманың WM_PAINT оқиғалар өңдеуішін алдын ала дайындау үшін форма терезесінің қасиеттер терезесінде PAINT пунктін тышқанмен екі рет басу керек (17.1-сурет).



17.1-сурет – WM_PAINT өңдеуішін құру

Нәтижесінде WM_PAINT оқиғалар өңдеуіші құрылады (Form1_Paint – 17.1-сурет). Бұл өңдеуіш терезе бейнесін қайта салу керек болған жағдайларда әрдайым орындалады.

Paint оқиғасының өңдеуіші мына түрде құрылады:

```
private void Form1_Paint(object sender, PaintEventArgs e)
 {
 }
```

Form1_Paint өңдеуішіне екі параметр жіберіледі. Бірінші параметр арқылы оқиғаны туындатқан объектке сілтеме жіберіледі, біздің жағдайымызда сілтеме Form1 формасына болады (қай жерде сурет салу керек).

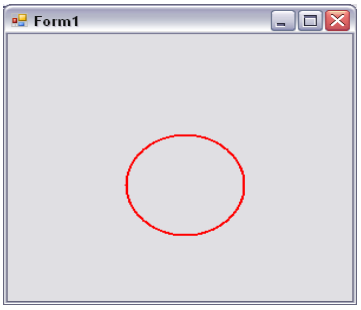
Екінші параметрді қарастыратын болсақ, осы параметр арқылы PaintEventArgs класының объектісіне сілтеме жіберіледі. Ол объект оқуға

ғана арналған ClipRectangle қасиетіне ие. ClipRectangle қасиеті арқылы облыстың шекаралары беріледі және оны Paint оқиғасының өңдеуіші қайта салуы керек. Осы шекаралар Rectangle класының объекттері ретінде беріледі. Осы кластың Left, Right, Width, Height қасиеттері, облысьның орналасу орны мен өлшемін анықтауға көмектеседі. Paint оқиғасының өңдеуіші ClipRectangle қасиетін ескермей, терезені толық қайтадан салады. Мысалы,

```
    private void Form1_Paint(object sender, PaintEventArgs e)
     {   Pen myPen = new Pen(Color.Red, 2);
         Graphics g = e.Graphics;
     g.DrawEllipse(myPen, 100, 100, 100, 100);
}
```

Осы мысалда қызыл түсте, 2 пиксельдегі қалыңдықта салу үшін myPen объектісі – «перо» құрылады. Қайта салынатын облыс (бүкіл форма) үшін Graphics типіндегі g объектісі құрылады. Назар аударыңыз, new қолданылмайды. Одан кейін g объектісі үшін эллипсті салу әдісі іске қосылады.

Graphics типіндегі g объектісінің ерекшелігі – объектің монитор құрылғысының контекстімен нұсқауыш болып келуі (компьютер бейнекартасының драйверімен қосымшаны байланыстыратын Windows жүйесінің арнайы бағдарламалары). Құрылғылар контекстерінің көмегімен Windows жүйесі қосымшалар мен компьютер құрылғыларының драйверлер үйлесімділігін қамтамасыз етеді, мысалы бейнекарта типіне тәуелсіз бағдарлама коды өзгеріссіз қалады, ал бейнекартаны басқару бойынша барлық мәселелерді монитор құрылғысының контексі шешеді. Бағдарламаның жұмысы 17.2-суретте көрсетілген.



17.2-сурет – Эллипсті шығару

C# тіліндегі басқа кластар сияқты Graphics класының қасиеттері мен әдістері бар. Олардың кейбірін қарастырып өтейік.

Clear() – бұл әдіс Graphics объектісін пайдаланушы таңдап алған түске бояйды.

Кейбір геометриялық фигураларды «салуға» арналған әдістердің үлкен тобы: DrawArc(), DrawBezler(), DrawBeziers(), DrawCurve(),

DrawEllipse(), DrawIcon(), DrawLine(), DrawLines(), DrawPie(), DrawPath(), DrawRectange(), DrawRectangles(), DrawString().

Геометриялық фигуралардың ішкі облыстарын толтыру үшін алдында Fill сөзі бар әдістер қолданылады, мысалы, FillPie(), FillElllpse() немесе FillRectangle().

Кейбір әдістердің жұмысын «Визуальное проектирование приложений C#» кітабынан (авторлары А.В. Фролов, Г.В. Фролов. 10-бөлім) оқи аласыз [3]. Ескеру керек, облыстарды қайта салуға арналған Graphics типіндегі объектті қолдану үшін `Form1_Paint` оқиғалар өңдеушісіне тиісті PaintEventArgs класының объектісінен бөлек форма мен басқару элементтерінің кластарында сипатталатын `CreateGraphics` әдісін қолдануға болады (). Мысалы,

```
Graphics g = this.CreateGraphics();   немесе
Graphics g = Graphics.FromHwnd(this.Handle);
```

Графикалық объектті `Image` объектісі арқылы құруға болады. Осы тәсіл берілген суретті (кескінді) өзгерту үшін қолданылады, мысалы,

```
Bitmap bm = new Bitmap( "d:\\picture.bmp" );
Graphics g = Graphics. FromImage( bm );
```

17.3 Бағдарламаны жүзеге асыру мысалы

Мысал ретінде монитор экранына граф типіндегі құрылымды шығару бағдарламасын қарастырайық.

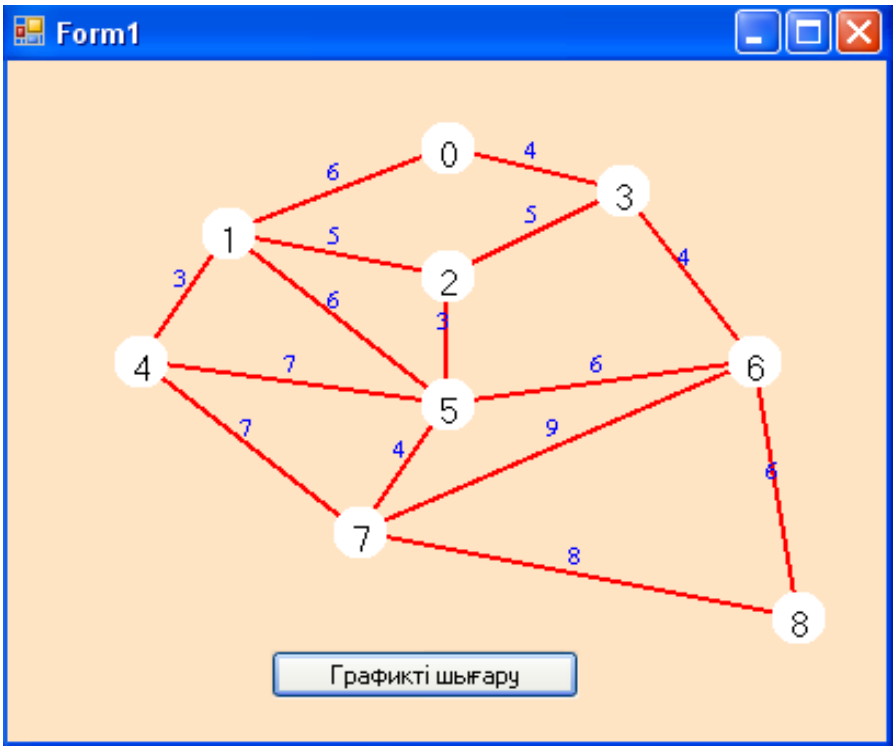
Form1.cs файлының коды:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
 public partial class Form1 : Form
 {
 public Form1()
 {
 InitializeComponent();
 }
 public static int p = 0;
 private void Form1_Paint(object sender, PaintEventArgs e)
 {
 Graphics g = e.Graphics;
 if (p == 0)
```

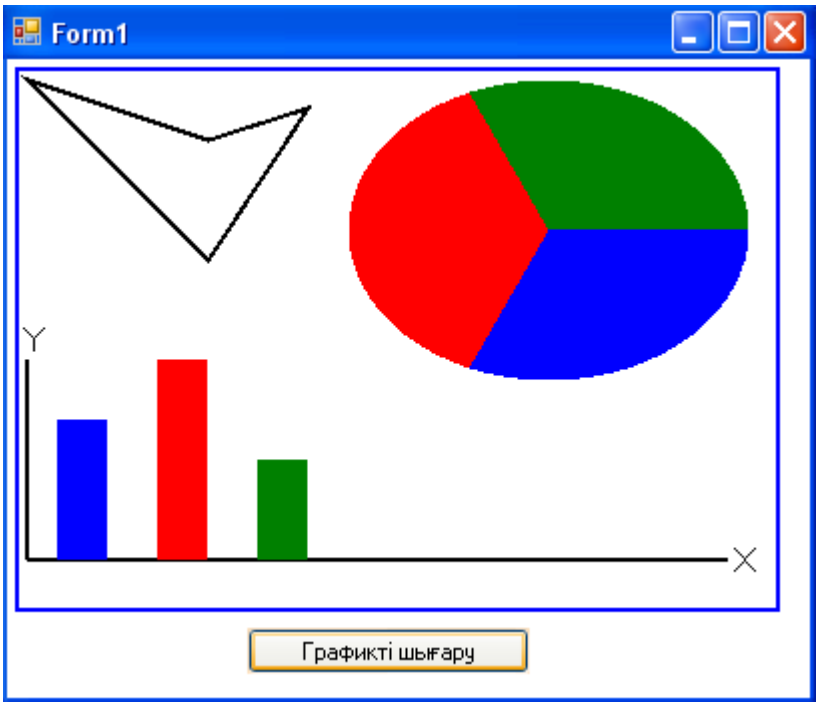
```
      {
      String st;
      int i, j;
      int xc, yc;
      //g.Clear(Color.White);
      int[,] a = new int[9, 9]
      {{0, 6, 1000, 4, 1000, 1000, 1000, 1000, 1000},
       {6, 0, 5,1000,3,6,1000,1000,1000},
       {1000,5,0,5,1000,3,1000,1000,1000},
       {4,1000,5,0,1000,1000,4,1000,1000},
       {1000,3,1000,1000,0,7,1000,7,1000},
       {1000,6,3,1000,7,0,6,4,1000},
       {1000,1000,1000,4,1000,6,0,9,6},
       {1000,1000,1000,1000,7,4,9,0,8},
       {1000,1000,1000,1000,1000,1000,6,8,0}};
      int[,] V = new int[9, 2]
       {{200, 40},
       {100, 80},
       {200, 100},
       {280, 60},
       { 60, 140},
       {200, 160},
       {340, 140},
       {160, 220},
       {360, 260}};
      e.Graphics.Clear(Color.Bisque);
      Pen myPen = new Pen(Color.Red, 2);
      for (i = 0; i < 9; i++)
      for (j = 0; j < 9; j++)
      if ((a[i, j] != 0) && (a[i, j] != 1000))
      {
      g.DrawLine(myPen, V[i, 0], V[i, 1], V[j, 0], V[j, 1]);
      xc = (int)(V[i, 0] + V[j, 0]) / 2;
      yc = (int)(V[i, 1] + V[j, 1]) / 2;
      st = Convert.ToString(a[i, j]);
      g.DrawString(st,   new   Font("Times   new   Roman",8),
Brushes.Blue, xc - 6, yc - 15);
      }
      for (i = 0; i < 9; i++)
      {
      g.FillEllipse(Brushes.White, V[i, 0] - 12, V[i, 1] - 12,
25, 25);
      st = Convert.ToString(i);
      g.DrawString(st, new Font("10_IC_1", 12), Brushes.Black,
V[i, 0] - 6, V[i, 1] - 6);
      }
      }
      if (p == 1)
      {
      g.Clear(Color.White);
      g.DrawRectangle(new  Pen(Brushes.Blue,  2),  5,  5,  380,
270);
```

```
      Pen myPen = new Pen(Color.Black, 2);
      Point[] myPoints =
      {
      new Point(10, 10),
      new Point(100, 40),
      new Point(150, 24),
      new Point(100, 100),
      };
      g.DrawPolygon(myPen, myPoints);
      g.DrawLine(myPen, 10, 250, 360, 250);
      g.DrawLine(myPen, 10, 250, 10, 150);
      g.DrawString("Y", new Font("10_IC_1", 12), Brushes.Black,
5,130);
      g.DrawString("X", new Font("10_BT_1", 12), Brushes.Black,
360,240);
      g.FillRectangle(Brushes.Blue, 25, 180, 25, 70);
      g.FillRectangle(Brushes.Red, 75, 150, 25, 100);
      g.FillRectangle(Brushes.Green, 125, 200, 25, 50);
      g.FillPie(Brushes.Blue, 170, 10, 200, 150, 0, 120);
      g.FillPie(Brushes.Red, 170, 10, 200, 150, 120, 120);
      g.FillPie(Brushes.Green, 170, 10, 200, 150, 240, 120);
      }
      }
      private void button2_Click(object sender, EventArgs e)
      {
      if (p == 0) p = 1; else p = 0;
      this.Invalidate();
      }
      }
     }
```

Бағдарлама жұмысы 17.3 және 17.4 суреттерінде көрсетілген.



17.3-сурет – Графты шығару



17.4-сурет – Түрлі графикалық фигураларды шығару

Бұл мысал «Визуальное проектирование приложений C#» кітабынан алынған (авторлары - А.В. Фролов, Г.В. Фролов) [3].

Бағдарламада ақпаратты шығаруды басқару глобальді p айнымалысы арқылы орындалады. Егер p = 0 (бағдарлама іске қосылғанда) болса, онда форма терезесіне графтың суреті шығады. Форма терезесіне «Графикті

шығару» батырмасын басқанда p мәні қарама-қарсы мәнге ауысады. Егер p 0-ге тең болса, онда ол 1-ге тең болады және керісінше.

Батырманы басу оқиғасының өңдеуішінде ағымдағы объектке арналған әдіс бар - this.Invalidate(); (this нұсқағышы бағдарлама жұмысында қолданылатын ағымдағы объект адресін сақтайды,). Қарастырылған мысалда ағымдағы объект болып форма есептеледі. this.Invalidate(); әдісі Windows операциялық жүйесінен форма үшін WM_PAINT хабарламасын дайындауды талап етеді. Бағдарлама осы хабарды алғаннан кейін private void Form1_Paint(object sender, PaintEventArgs e) оқиғасының өңдеуішін іске қосады, яғни P айнымаласының мәні жаңа болса форма терезесі қайтадан салынады. Сонымен, Windows операциялық жүйесінің WM_PAINT хабарын қосымшадан «алуға» («формировать») болады.

Form1.cs файлының кодын қарастырайық. Бірінші бөлім граф суретінен тұрады. Қосымша граф төбелерінің координаттар массиві – V қосылған. Алдын ала граф қағаз бетінде салынып, координаттардың мәндері суреттен алынған. Форма терезесін тазартамыз – `e.Graphics.Clear(Color.Bisque);`. - `Color.Bisque` түсімен бояймыз. Қызыл түсті таңдаймыз, сызықтың қалыңдығы 2 пиксельге тең болады - `Pen myPen = new Pen(Color.Red, 2);`. Граф төбелерінің арасында сызықтарды салу циклін іске қосамыз. Граф төбелерінің арасындағы орташа нүкте үшін бір уақытта xc және yc координаттары есептеледі. Осы координаттарды қолдана отырып, сыбайлас матрицалардан алынған оның мәнін қабырға сызығының үстіне шығарамыз. `g.DrawString` жазбасының пішімі бойынша мәтінді шығарған кезде қаріп типін, өлшемін көрсету керек. Келесі циклде граф төбелері салынады және жазылады. `g.FillEllipse` әдісі тіктөртбұрышты аймаққа іштей сызылған эллипсті бояйды, оған оның орналасу орны мен өлшемдері параметрлер арқылы беріледі.

Форманың «екінші» терезесінде терезені ақ түспен «тазарту» орындалды. Мысал ретінде көрсету мақсатында көпбұрыш, `X` пен `Y` координаттарының осінің сызығы, тіктөрбұрыштар (гистограмма) жиыны және шеңберлі диаграммалар «салынды».

`g.DrawPolygon(myPen, myPoints);` әдісі көпбұрышты салуға көмектеседі, олардың төбелерінің координаттары массив арқылы анықталады. Әдістің бірінші параметрі сызықтың түсін анықтайды.

`g.FillRectangle` әдісі боялған тіктөрбұрыштарды салуға мүмкіндік береді, тіктөрбұрыштың сол жақ төбесіндегі координаттары, жалпақтығы, ұзындығы берілген. Бірінші параметр ретінде тіктөрбұрыштың ішкі аумағын бояу түсі беріледі.

Шеңберлі диаграмма `g.FillPie` әдісі арқылы салынады, ол «боялған» сегменттерді көрсетуге арналған. Бірінші параметр ретінде әдіске сегменттің ішкі аумағын «бояу» түсін жіберу керек. Келесі төрт параметрлер іштей сызылған эллипсі бар тіктөрбұрыштың орналасуын және өлшемін анықтайды. Соңғы екі параметр эллипс сегментін шектейтін бұрыштарды анықтайды: бастапқы бұрышты және оның сағат тілі бағытындағы өсімшесін.

17.4 Өзін-өзі тексеру сұрақтары

1 C# тілінде GDI+ нені білдіреді?

2 Brush класының объектісі нені анықтайды?

3 Pen класының объектісі нені анықтайды?

4 Graphics класы нені анықтайды?

5 Windows жүйесінде қандай хабарлама терезе өлшемінің өзгеруін және терезе орынының ауысқанын «бақылайды»?

6 Қосымша формасының терезесін қайта салу өңдеуіш қалай аталады?

7 Form1_Paint өндеуішінің бірінші формалды параметр нені анықтайды?

8 Form1_Paint өндеуішінің екінші формалды параметрі нені анықтайды?

9 Монитор құрылғысының контекст ұғымы.

10 Қандай әдіс Windows операцилық жүйесінен форма үшін WM_PAINT хабарын дайындауды талап етеді?

18 ҚОСЫМШАДА МЕНЮДІ ҚОЛДАНУ

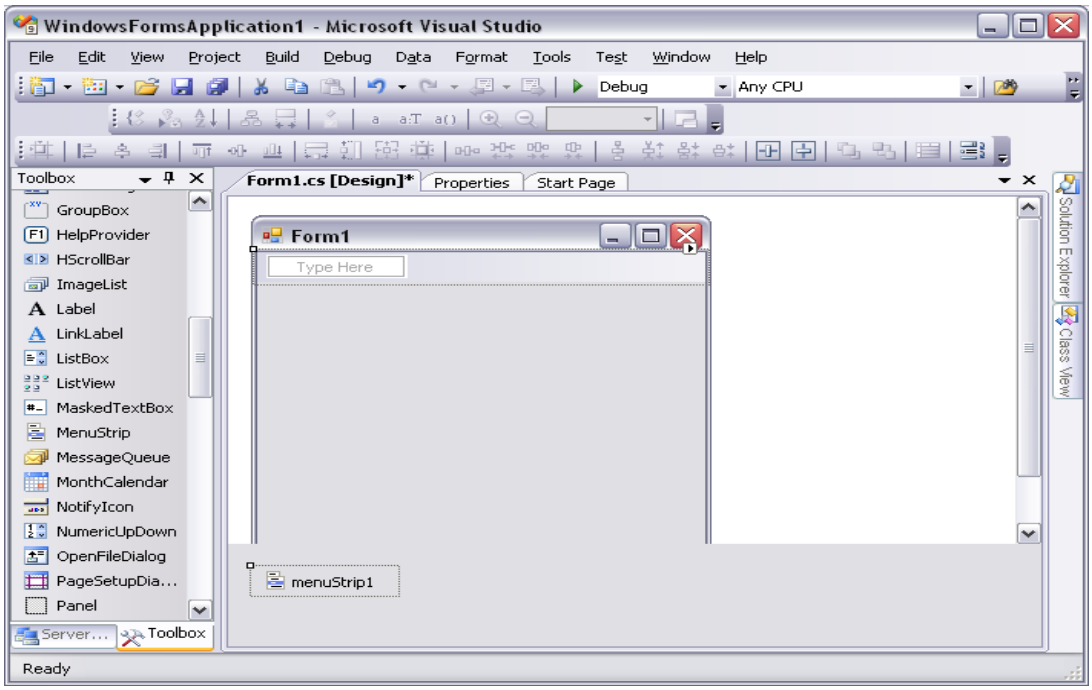
18.1 Бағдарлама менюі

Бағдарлама менюі бағдарлама жұмысының негізгі режіміне сәйкес болуы керек. Бағдарламада менюді пайдалану технологиясын түсіну үшін келесі мысалды қарастырайық.

18.1-есеп. Қосымшада 10-нан 100-ге дейінгі аралықта 6*6 матрицасын құру керек. richTextBox қолданып матрицаны экранға шығару керек. Терезедегі нәтижені мәтіндік файлға жазып, мәтіндік файлды оқып оны терезеге көрсету керек. Барлық әрекеттерді меню арқылы орындаңыз.

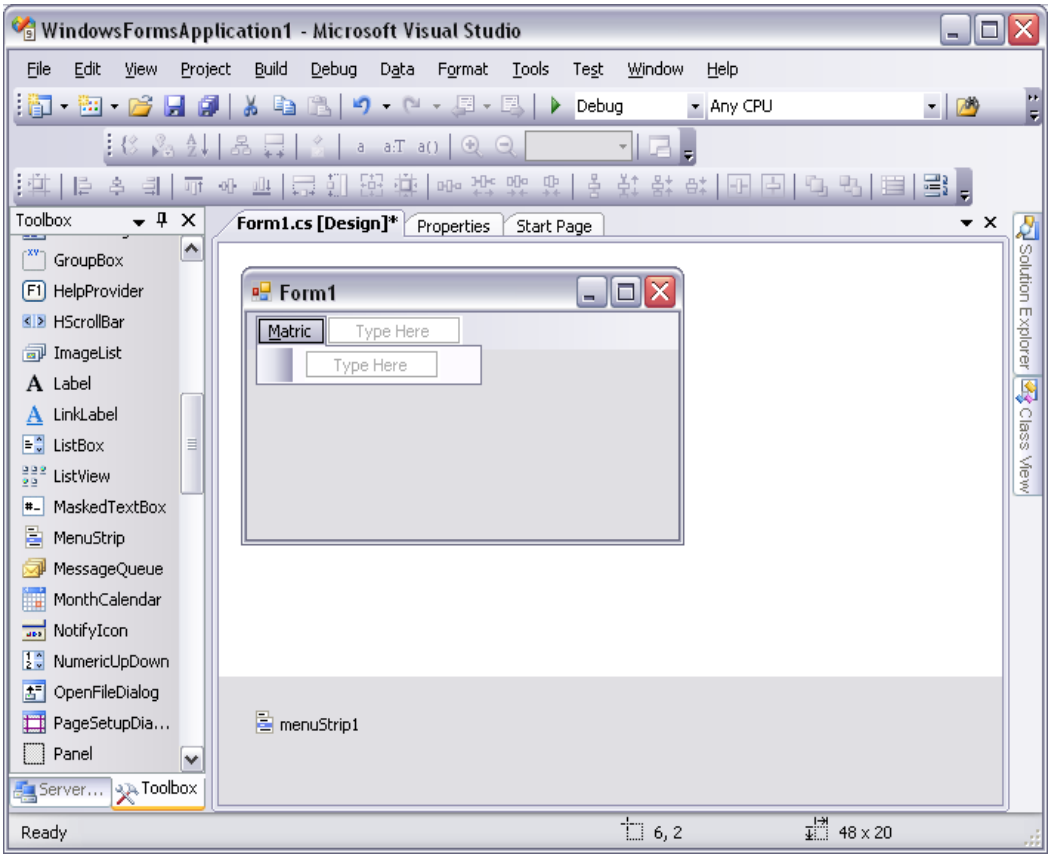
Бағдарлама жұмысын үш режімге бөлуге болады: матрица (құру және шығару), файл (жазу және оқу), мәтін (тазарту және түрлендіру).

Windows жүйесінде жұмыс жасау үшін қарапайым қосымшаны әзірлейік. Негізгі терезеге менюді қосу үшін Toolbox терезесінен MenuStrip деп аталатын менюді жылжыту керек, менюді орналастырғаннан кейін форма терезесі келесі түрде болады (18.1-сурет).



18.1-сурет – Менюді қосу

Осы терезенің төменгі бөлігінде менюдің – menuStrip1 басқару элементінің белгісі (таңбашасын) пайда болады, менюде Type Here жазуы жазылып тұрады («осында жаз»). Осы өрісте «&Matric» жолын жазамыз. Сонымен қосымша терезесінде Matric менюі дайын болды (18.2-сурет).



18.2-сурет –Матрицамен жұмыс істеуге арналған менюді құру

& символын пайдаланып команданы перне арқылы енгізу үшін пернелік акселераторды дайындауға болады (Alt+P). Акселераторларды қолдану тышқан арқылы таңдау командаларының қосымша түрі болып келеді. Alt пернесі әріптің пернесімен (амперсанд символынан кейін тұрады) бірге қолданады. Менюдегі мәтінде акселераторға қосылған әріптердің асты сызылады.

Жұмыс істеу барысында Type Here енгізу өрісі төмен түседі және ол берілген режімдегі команданы қосуға мүмкіндік береді. &SozM және &PrintM екі командасын қосайық. Одан кейін Type Here енгізу өрісінің оң жағына көшеміз (файлдармен жұмыс істеу үшін). Менюдің үшінші пункті арқылы мәтінмен жұмыс істеу командасының режімін қосамыз.

Егер енгізу кезінде қателіктер кетсе, онда оны түзеуге болады. Ол үшін керекті жолды тышқанның оң жақ батырмасымен шертіп, контексті меню арқылы қажетті редакциялау режімін таңдау керек.

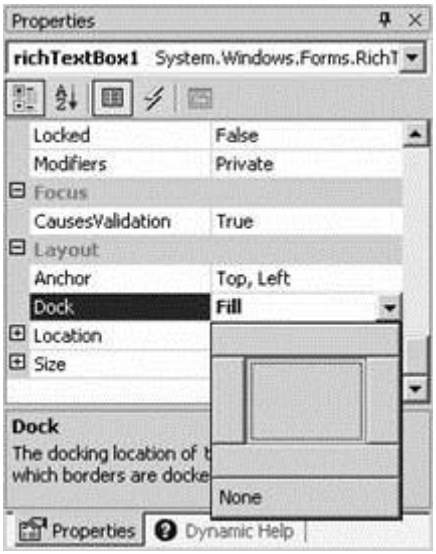
Insert New жолы арқылы Сіз менюдегі жолдардың арасына жаңа жолды қоса аласыз. Insert Separator меню жолдары арасына бөлетін сызықты қосу үшін қолданылады. Edit Names жолы арқылы меню мен жол идентификаторын редакциялауға болады.

Осыдан кейін қосымшаны трансляциялауға және F5 батырмасы арқылы оны іске қосуға болады. Менюді ашып, оның жолдарын тексеріп

көріңіз. Егер барлығы дұрыс орындалса, онда қосымшаны дайындаудың келесі кезеңіне өтуге болады.

Қосымшада мәтінмен жұмыс жасау шін RichTextBox элементін қолданамыз. Ол үшін Toolbox терезесінен формаға RichTextBox элементін көшіреміз. RichTextBox элементі әр түрлі типтегі файлдармен жұмыс істеуге мүмкіндік беретін

Қосымша терезесінде түгелдей орын алуы үшін компоненттің қасиетін күйге келтіріңіз. Одан кейін Dock қасиетін редакциялаңыз. Бұл қасиет компоненттің форманың ішінде орналасу орнын анықтайды (ортасында шерту керек, 18.3-сурет).



18.3-сурет – Dock қасиетін редакциялау

Қосымша іске қосылғаннан кейін нәтижесінде 18.4-суретінде көрсетілгендей терезе пайда болады.

18.4-сурет – Қосымшаның жұмыс терезесі

Меню командаларының оқиғалар өңдеуіштері батырмалардың оқиғалар өңдеуіштері сияқты құрылады. Меню командасы бойынша оқиға өңдеуішін құру үшін оны тышқанның сол жақ пернесімен екі рет шерту керек.

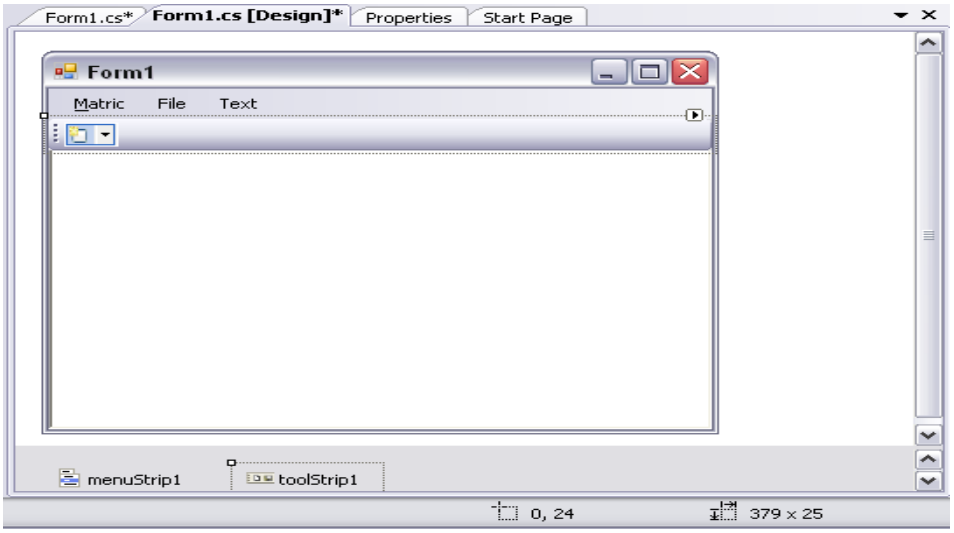
Барлық командалар үшін оқиғалар өңдеуіштерін дайындайық, бірақ бағдарлама режімдеріне емес. Оқиғалар өңдеуіштерінің денесі бос болады, мысалы:

```
    private void sozMToolStripMenuItem_Click(object
sender, EventArgs e)
    { }
```

Қосымшалардың көбінде менюдің негізгі командалары инструменттік панельдегі «иконмен» қайталанады. Қосымшаның командалары үшін инструменттік панельді құрайық.

18.2 Қосымшаның инструментті панелін құру

Қосымшаның терезесіне инструментті панелді қосу үшін Visual Studio .NET ортасының басқару элементтерінің терезесінен ToolStrip белгісін (таңбашасын) форманы жобалау терезесіне көшіреміз.

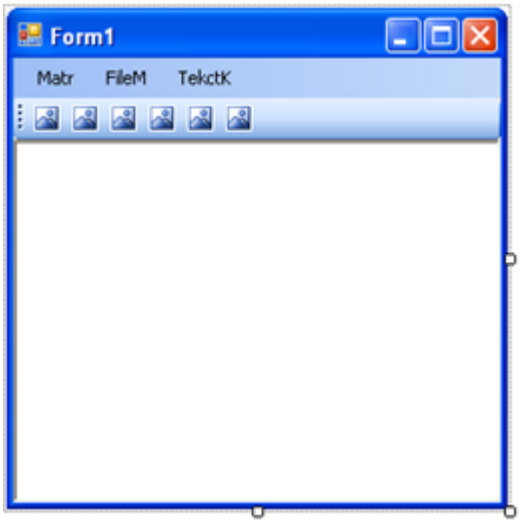


18.5-сурет – Қосымшада инструментті панелді құрудың бірінші кезеңі

Инструментті панель терезесі форманың жоғарғы бөлігінде орналасады және онда батырма болмайды. 18.5-суретте инструментті панелді құрудың бірінші кезеңі көрсетілген.

18.5-суретте көрсетілгендей инструментті панель терезесі мәтін редакторының ішінде орналасқаны көрініп тұр (оның жоғарғы бөлігінде). Оны дұрыстау үшін тышқанның оң жағымен мәтін редакторының терезесін шертіңіз, Bring to Front жолын таңдаңыз. Нәтижесінде терезелер өзара дұрыс орналасады.

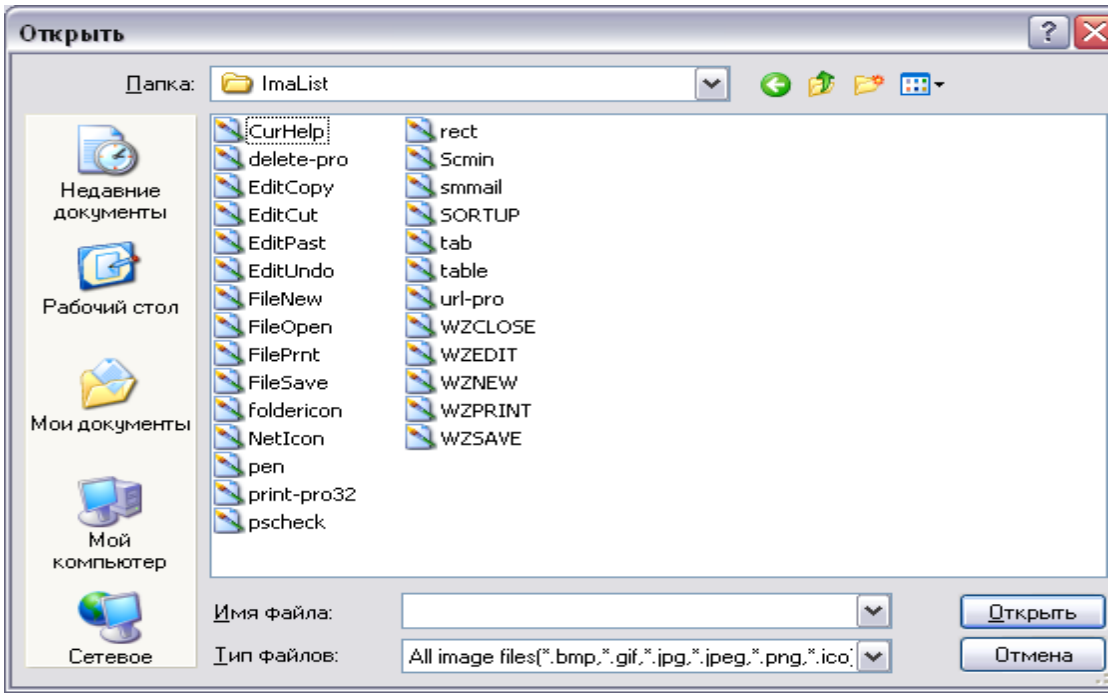
Инструментті панельде ToolStrip элементінің батырмасына курсорды жақындатсақ, онда Add ToolStripButton сөзі көрінеді. ToolStrip элементінің батырмасын тышқанның сол жақ батырмасымен басқанда инструментті панельдің оң жағында дәл осы сияқты батырма пайда болады. Барлық батырмалардың суреттері бірдей.



18.6-сурет – Инструментті панель батырмаларын құру

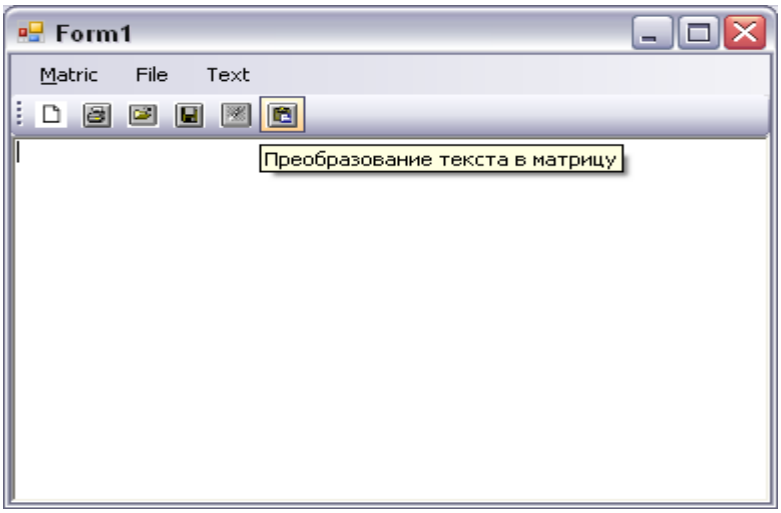
Сондықтан инструментті панельдегі батырмалардың суреттерін өзгерту керек. Тізімді суреттермен (кескіндермен) толтырмас бұрын, Сізге кез келген графикалық редактордың көмегімен оларды дайындап алу керек немесе суреттерді (кескіндерді) тауып, жобадағы жеке бумаға көшіру керек (осы жобада ImaList бумасы қолданылады). С дискісінде *.bmp үлгісі бойынша іздестіруді қосуға болады және табылған суреттердің (кескіндердің) ішінен мағынасы бойынша келетін суреттерді (кескіндерді) таңдап алуға болады.

Осыдан кейін бірінші батырмадан бастап барлық батырмалардың қасиеттерінде Imaje командасын таңдап, Select Resource терезесін ашу керек, оның ішінде Import командасын таңдаймыз. Ашылатын Open диалогтық терезесінде ImaList бумасына көшеміз, қажетті суреттерді таңдаймыз.



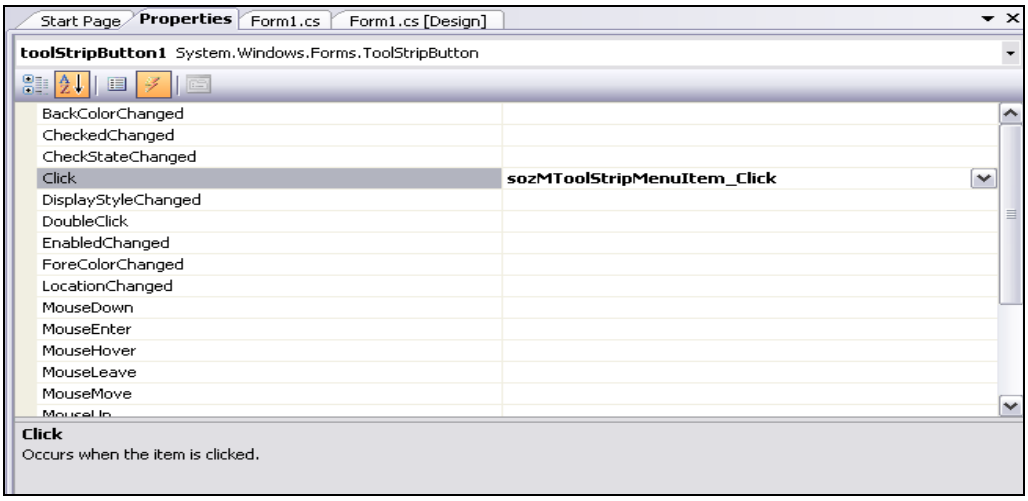
18.7-сурет – Инструменттік панелдің батырмаларына арналған суреттер файлын таңдау

Инструменттік панельдегі әрбір батырманың ToolTipText қасиетін редакциялау бізге әрбір батырмаға тышқан курсорын жақындатқанда түсіндірме хабарды көру мүмкіндігін береді.



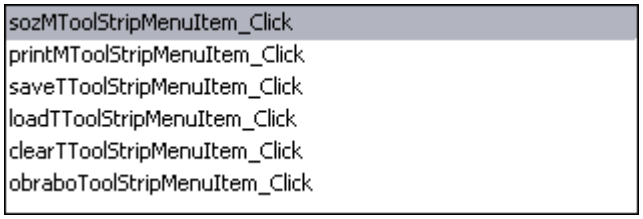
18.8-сурет – Қосымшаның жұмыс терезесі

Инструменттік панель батырмаларының оқиғалар өңдеуіштерін меню командаларының оқиғалар өңдеуіштерімен байланыстыру керек. Ол үшін әрбір батырманың қасиеттерінде «оқиғалар» бетін ашамыз, Click оқиғасын таңдаймыз (18.9-сурет).



18.9-сурет - Оқиғалар өңдеуішін таңдау

Оқиғалар кестесіндегі Click оқиғасын екі рет шертеміз, сонымен қатар меню командалары үшін құрылған оқиғалар өңдеуіштерінің диалогтық терезі ашылады (18.10-сурет). Керекті оқиғаның өңдеуішін таңдаймыз, біздің мысалымызда ол инструменттік панельдің бірінші батырмасы және оған sozMToolStripMenuItem_Click оқиғасын сәйкес көрсету керек.



18.10-сурет – Меню командаларының оқиғалар өңдеуіштерінің тізімі

18.3. Өзін-өзі тексеру сұрақтары

1 Қосымшада меню не үшін құрылады?

2 Қосымшада меню командалары неге сәйкес келуі керек?

3 ToolBox терезесінің қандай басқару элементі қосымшада менюді құруға мүмкіндік береді?

4 Меню редакторының қандай өрісі команданы енгізу үшін қолданылады?

5 Меню редакторының қандай командасы арқылы менюге жаңа жолды жолдардың арасына қосуға мүмкіндік береді?

6 Қосымшада менюді құрған кезде пернетақта акселераторларын не себепті қолдану керек?

7 Меню редакторының Insert Separator командасы не үшін қолданылады?

8 RichTextBox және TextBox басқару элементтерінің арасында қандай ерекше айырмашылықтар бар?

9 Қосымшадағы менюдің негізгі командаларын қайталайтын қандай басқару элементі суреттерді сақтау үшін қолданылады?

10 Инструменттік панель батырмасының қанда қасиеті әр батырманы түсіндірме терезесімен қамтамасыз етеді, түсіндірме терезесі батырмаға тышқан меңзерін «жақындатқанда» пайда болады?

19 ДИАЛОГТЫҚ МЕНЮДІ ҚОЛДАНУ

19.1 Матрицамен жұмыс жасауға арналған оқиғалар өңдеуіштері

Қосымшада менюмен және инструменттік панельмен жұмыс жасауды жалғастыра отырып бізге 6*6 матрицасын құруға және оны қосымша терезесіне шығаруға арналған хабарламалар өңдеуіштерінің кодын жазу керек.

Матрица құрылғаннан кейін өңдеуіш жұмысының сәтті аяқталғаны туралы хабарламаны экранға шығару керек, мысалы, «Матрица құрылды».

«Матрицаны шығару» әдісінің нәтижесінде матрица экранға шағады, сондықтан қосымша хабарламаны шығарудың қажеті жоқ.

Form1.cs файлының коды үзінділер бойынша қарастырылады.

Form1.cs файлының коды:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
 public partial class Form1 : Form
 {
 public static int[,] a = new int[6, 6];
 public Form1()
 {
 InitializeComponent();
 }
 private void sozMToolStripMenuItem_Click(object sender,
     EventArgs e)
 {
 Random rnd = new Random();
 for (int i = 0; i < 6; i++)
 for (int j = 0; j < 6; j++)
 a[i, j] = rnd.Next() % 90 + 10;
 richTextBox1.AppendText("Matrisa kyrildi \n");
 }
 private void printMToolStripMenuItem_Click(object sender,
EventArgs e)
 {
 string ss;
 richTextBox1.AppendText("\n");
 for (int i = 0; i < 6; i++)
 {
 ss = "";
 for (int j = 0; j < 6; j++)
 ss = ss + Convert.ToString(a[i, j]) + "\t";
```

```
richTextBox1.AppendText(ss + "\n");
}
}
    .    .    .
```

Басқа оқиғалар өңдеуіштерінің коды әзірше бос.

19.2 Файлды ашуға арналған оқиғалардың өңдеуіштері

Инструменттік панельде файлды ашу батырмасы – LoadT командасы бар. Осы оқиға өңдеуішін жазу бойынша әрекеттердің ретін қарастырайық.

Қосымшада файлды ашу бойынша оқиға өңдеуішін жазу үшін бізге OpenFileDialog элементі керек болады. Осы элементтің таңбашасын Toolbox терезесінен көшірейік. OpenFileDialog элементінің таңбашасы форма астындағы панельде пайда болады. (19.1-сурет).

19.1-сурет –OpenFileDialog элементін қосу

OpenFileDialog элементін қолдану бағдарламалаудың бүтін бір технологиясы болып келеді, ол компьютердегі бумаларды, дисктерді, файлдарды қолдануға мүмкіндік береді. Осы элемент Toolbox терезесіндегі кез келген элемент сияқты файлдармен жұмыс жасаудың көптеген әдістері бар класс ретінде берілген, мысалы, openFileDialog1.ShowDialog әдісі экранға файлды таңдаудың стандартты диалогтық терезесін көрсетеді (19.2-сурет).

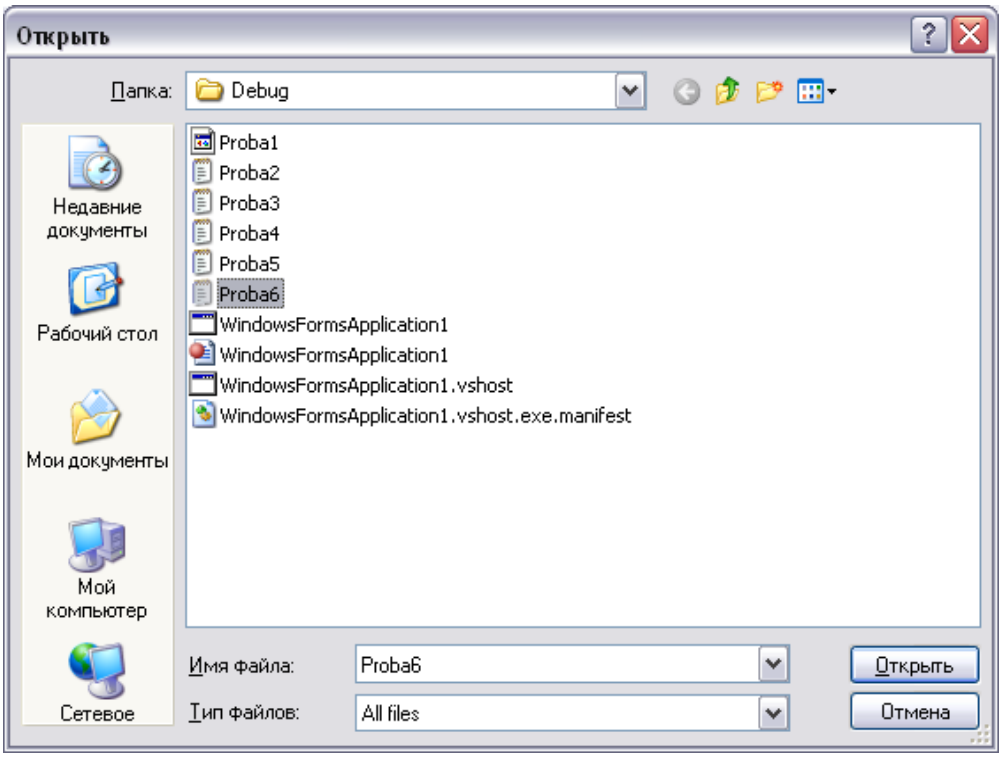
openFileDialog1 элементінің қасиеттерін күйге келтіре отырып ашылатын файлдардың атауларына фильтрді анықтауға болады. Ол үшін openFileDialog1 элементінің Properties терезесінде Filter қасиетін өзгерту керек. Осы қасиетке келесі мәтіндік жолды меншіктеңіз:

```
Text files|*.txt|RTF files|*.rtf| All files|*.*
```

Фильтрдің жолы «|» символымен бөлінген бөліктерден тұрады. Бірінші бөлік файл типінің атауын Text files деп белгілейді, қалған бөліктерде файл атауларынын қалқасы (маскасы) анықталған. Мәтіндік файлдар үшін *.txt қалқасы (маскасы) қолданылады.

Одан кейін RTF files форматы орналасқан. RTF файлдарна *.rtf қалқасы (маскасы) қолданылады.

Қосымшада барлық типтегі файлдар ашылу үшін (All files) *.* маскасы (қалқа) қолданылады (мысал 19.2-суретте көрсетілген).



19.2-сурет – Диалогтық терезені ашу бойынша бағдарлама жұмысы

Есептің шарты бойынша біз мәтіндік файлдармен жұмыс істейтін боламыз, бірақ басқа типтегі файлдарды да ашып көруге болады.

Қосымша файлының коды толықтырылды:

```
private void loadTToolStripMenuItem_Click(object sender,
 EventArgs e)
{
 if (openFileDialog1.ShowDialog() == DialogResult.OK &&
 openFileDialog1.FileName.Length > 0)
 {
 try
 {
 richTextBox1.LoadFile(openFileDialog1.FileName,
RichTextBoxStreamType.PlainText);
 }
 catch (System.ArgumentException ex)
 {
 richTextBox1.LoadFile(openFileDialog1.FileName,
RichTextBoxStreamType.RichText);
 }
 this.Text = "Файл [" + openFileDialog1.FileName + "]";
 }
 }
```

Егер файлды таңдау терезесінде пайдаланушы «Открыть» батырмасын басқан болса, онда ShowDialog әдісі DialogResult.OK мәнін қайтарады. Шартқа файл таңдалғаны жөнінде қосымша тексеру енгізілген

(таңдап алынған файлға дейінгі жолдың ұзындығы – openFileDialog1.FileName.Length 0-ден үлкен болуы керек).

Файлды richTextBox1.LoadFile әдісімен ашу барысында оған екі параметр жіберіледі. Бірінші параметр – файлға дейінгі жолды, ал екінші параметрге файл типін анықтаймыз.

Мәтіндік файлдармен жұмыс жасау керек болғандықтан, негізгі тип RichTextBoxStreamType.PlainText болып табылады.

Егер таңдап алынған файлдың форматы LoadFile әдісіндегі PlainText форматынан өзге болса, онда System.ArgumentException ерекше жағдайы (исключение) туындайды. Өңдеуіш фалды RichText типінде файл ретінде қайтадан ашуға әрекеттенеді. Бұл тип RTF форматына сәйкес. Қосымша ақпарат үшін ашылған файлдың толық жолы қосымшаның негізгі терезесінің бас жағында көрсетіледі.

19.3 Файлға жазу бойынша оқиғалардың өңдеуіштері (Файлға жазу үшін қолданылатын оқиғалардың өңдеуіштері)

Инструменттік панельде кезек бойынша келесі болып қосымшаның терезесіндегі жазбаларды файлға жазуға арналған батырма (SaveT командасы) орналасқан. Осы оқиға өңдеуішін жазу бойынша әрекеттердің ретін қарастырайық.

Қосымшада файлға жазуды орындайтын өңдеуішті құру үшін бізге SaveFileDialog элементі қажет болады. Осы элементтің таңбашасын Toolbox терезесінен форма терезесіне көшірейік. SaveFileDialog1 элементінің таңбашасы форманың астында орналасқан панельде пайда болады.

SaveFileDialog1 элементінің қасиеттерін күйге келтіре отырып, оның Filter және FileName қасиеттерін редакциялау керек.

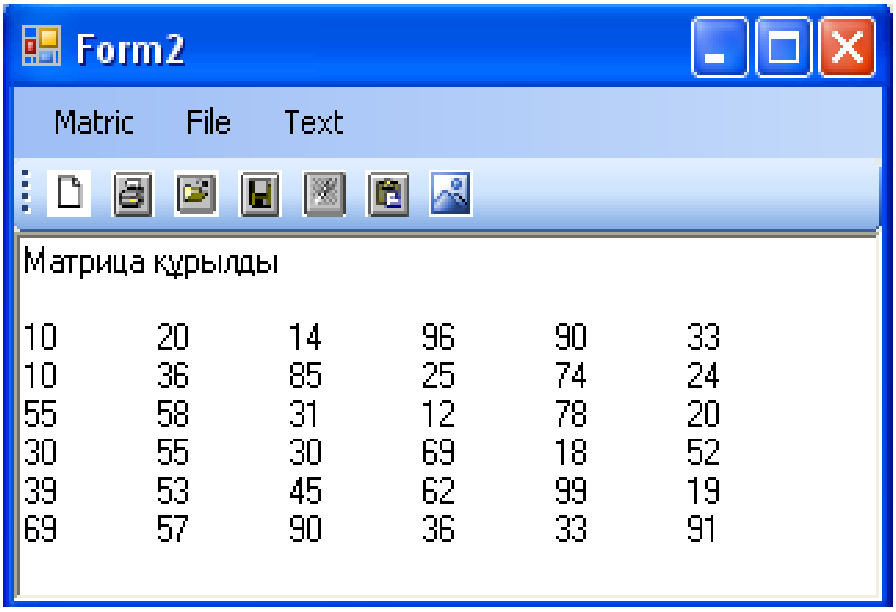
Filter қасиетіне мәтіндік файлдың типін көрсету керек – Text files|*.txt. Ал FileName қасиетіне құжат атауын – doc1.txt мәнін меншіктеу керек. Файлдар осы атаумен сақталады.

Қосымша файлының коды толықтырылды:

```
private void saveTToolStripMenuItem_Click(object sender,
 EventArgs e)
{
 if (saveFileDialog1.ShowDialog() == DialogResult.OK &&
 saveFileDialog1.FileName.Length > 0)
 {
 richTextBox1.SaveFile(saveFileDialog1.FileName,
  RichTextBoxStreamType.PlainText);
 this.Text = "Файл [" + saveFileDialog1.FileName + "]";
 }
}
```

Файлдың толық жолы қосымшаның негізгі терезесінің бас жағында көрсетіледі.

richTextBox1.SaveFile әдісінің екінші параметрі сақталатын файлдың типін анықтайды. Қарастырылған мысалда бағдарлама тек мәтіндік файлдармен жұмыс істеуге бағытталған. Егер осы параметр арқылы RichTextBoxStreamType.RichText мәні жіберілсе, онда құжат RTF форматында сақталады.



19.3-сурет – Қосымшада мәтіндік файлмен жұмыс жасау мысалы

19.4 Мәтінмен жұмыс істеуге арналған оқиғалардың өңдеуіші

Қосымшаның пайдаланушыға арналған аумағында мәтінмен жұмыс жасау үшін менюдің тазарту мен түрлендіру командаларына сәйкес екі оқиғалар өңдеуіші қарастырылған.

Мәтінді тазарту бойынша оқиға өңдеуішінің ішіне барлық жолдарды өшіруді орындайтын richTextBox1 мәтін редакторының бір әдісі жазылған:

```
private void clearTToolStripMenuItem_Click(object sender,
EventArgs e)
 {
  richTextBox1.Clear();
 }
```

Мәтіндегі түрлендіруді жүргізетін оқиға өңдеуішісі пайдаланушыға арналған аумақта орналасқан мәтіннен 6*6 матрицасының мәндерін (әдетте файлдан алынатын) бөліп алу үшін қолданылады.

Мәтінді өңдеу әдісінің коды теменде келтірілген:

```
private void obraboToolStripMenuItem_Click(object sender,
EventArgs e)
```

```
{
 string ss;
 string[] clova;
 int k, n;
 int[] masi = new int[100];
 n = 0;
 ss = richTextBox1.Text;
 clova = ss.Split('\t', ' ');
 for (int i = 0; i < clova.Length; i++)
 {
 k = clova[i].Length;
 if (k == 2 || k == 3)
 {
 masi[n] = Int32.Parse(clova[i]); n++; }
 richTextBox1.AppendText(i.ToString() + " = " + clova[i] +
 " k= " + k.ToString() + "\n");
 }
 int j1, j2; j1 = j2 = 0;
 for (int i = 0; i < 36; i++)
 {
 a[j1, j2] = masi[i];
 j2++;
 if (j2 == 6) { j1++; j2 = 0; }
 if (j1 == 6 && j2 == 6) break;
 }
}
```

Мына жерде кейбір түсініктемелерді келтіріп өту керек.

String типіндегі clova массиві жарияланды, оған пайдаланушыға арналған аумақта орналасқан мәтіннен Split әдісі арқылы табуляция, бос орын немесе қайтару таңбасы арқылы ерекшеленген символдардың кез келген тіркестері жазылады:

```
ss = richTextBox1.Text;
clova = ss.Split('\t',' ','\n');
```

clova.Length - мәтіндегі барлық сөздердің санын анықтайды.

Циклда ұзындығы 2 немесе 3 символға тең барлық сөздер бүтін сандарға түрлендіріледі және masi бүтін сандар массивіне жазылады. Барлық сөздер мен олардың ұзындықтары бір мезгілде экранға шығады (бақылауды жүргізу үшін).

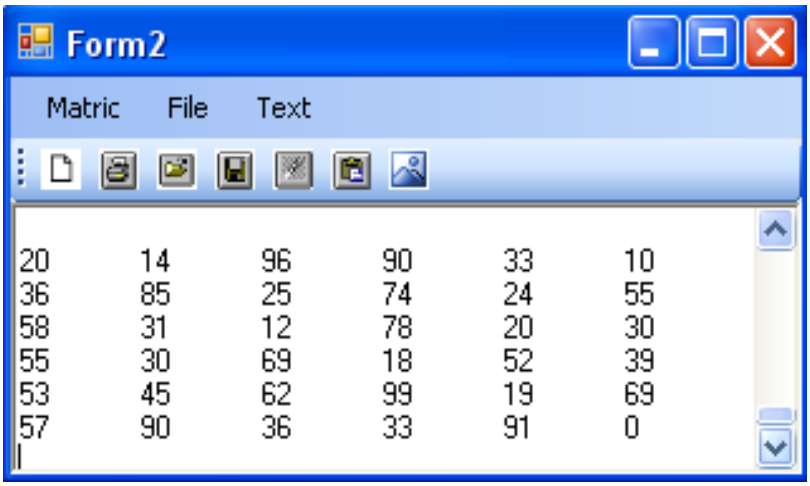
Оқиға өңдеуішінің соңғы бөлігінде массивтегі ерекшеленген бүтін сандар 6*6 матрицасына жазылады.

Пайдаланушы мына нәтижелерді көреді:

```
Матрица құрылды

10   20   14   96   90   33
10   36   85   25   74   24
55   58   31   12   78   20
30   55   30   69   18   52
39   53   45   62   99   19
```

```
69   57   90   36   33   91
0 = Матрица k= 7
1 = құрылды k= 7
2 =

10 k= 4
3 = 20 k= 2
4 = 14 k= 2
5 = 96 k= 2
6 = 90 k= 2
7 = 33 k= 2
8 =
10 k= 3
9 = 36 k= 2
10 = 85 k= 2
11 = 25 k= 2
12 = 74 k= 2
13 = 24 k= 2
14 =
55 k= 3
15 = 58 k= 2
16 = 31 k= 2
17 = 12 k= 2
18 = 78 k= 2
19 = 20 k= 2
20 =
30 k= 3
21 = 55 k= 2
22 = 30 k= 2
23 = 69 k= 2
24 = 18 k= 2
25 = 52 k= 2
26 =
39 k= 3
27 = 53 k= 2
28 = 45 k= 2
29 = 62 k= 2
30 = 99 k= 2
31 = 19 k= 2
32 =
69 k= 3
33 = 57 k= 2
34 = 90 k= 2
35 = 36 k= 2
36 = 33 k= 2
37 = 91 k= 2
38 = k= 1
```

Пайдаланушы аумағын тексергеннен кейін оны тазартуға болады. Одан кейін матрицаны экранға шығару режімін іске қосамыз. (19.4-сурет):



19.4-сурет – Пайдаланушы аумағындағы мәтінді түрлендіргеннен кейін матрицаны эканға шығару

«Матрица құрылды» хабарламасы шығарылған жоқ.

19.5 Өзін-өзі тексеру сұрақтары

1 Диалогтық категорияларға қандай басқару элементтері жатады?

2 Файлды ашу бойынша диалогтық терезені құру үшін қандай басқару элементі қолданылады?

3 Қандай әдіс файлды ашу үшін Windows жүйесінің стандартты диалогтық терезесін экранға шығарады?

4 OpenFileDialog басқару элементі арқылы тек мәтіндік файлдарды ашу үшін осы элементті қалай күйге келтіруге болады?

5 Файлды ашу бойынша диалогтық терезеде пайдаланушы «Открыть» батырмасын басса не болады?

6 if (openFileDialog1.ShowDialog() == DialogResult.OK && openFileDialog1.FileName.Length > 0)

жазбасында openFileDialog1.FileName.Length > 0 шарты нені білдіреді?

7 RichTextBox1 басқару элементінің қандай әдісі файлды ашу үшін қолданылады?

8 SaveFileDialog басқару элементінің қызметі қандай ?

9 SaveFileDialog басқару элементінің FileName қасиеті нені анықтайды?

10 richTextBox1.SaveFile әдісінің RichTextBoxStreamType.PlainText параметрі нені анықтайды?

20 КӨПТЕРЕЗЕЛІ ҚОСЫМШАЛАР

20.1 Негізгі «батырмалық» форманы құру

Оқу құралының алдынғы бөлімдерінде бір терезелік қосымшалар қарастырылған болатын, онда барлық басқару элементтері және бағдарлама жұмысының нәтижелері негізгі форманың бір терезесінде орналасады. Осы бөлімде көптерезелі интерфейсті қосымшаларды жобалау технологиясын (Multiple-document interface, MDI) қарастырамыз.

Көптеген көптерезелі қосымшаларда негізгі «батырмалық» форма қолданылады, қосымшаның негізгі формасында меню түсініктемелері бар «үлкен» батырмалар түрінде орындалған.

Осындай жағдайларда негізгі форманы «батырмалық» негізгі форма түрінде жобалау керек пе? Осы жол кең қолданылады. Құрылатын жобада пайдаланушыға бірнеше әртүрлі сервистер ұсынылатын болса, пайдаланушы жұмысты бастаған уақытта өзіне керекті сервисті таңдайды. Негізгі формада пайдаланушыға керекті сервистері бар меню болуы мүмкін. Егер де әрбір сервис күрделі және осы сервиске жеке интерфейс керек болатын болса, онда бұл жағдайда стандартты менюдің орнына «батырмалық» негізгі форманы қолдану ыңғайлы болады. Меню командалары ретінде формадағы командалық батырмалар қолданылады, меню командасын таңдау және командалық батырманы басу бір-біріне балама болып келеді.

20.1-есеп. Қосымшада тіктөртбұрышты типіндегі 6 геометриялық фигуралардан тұратын массивті құру керек. Тіктөртбұрыштар (тікбұрыштар) екі қарама-қарсы тұрған төбелердің координаттарымен беріледі. Төбелердің координаттары кездейсоқ минус 100-ден 100-ге дейінгі аралығындағы бүтін сандардан құрылады. Пайдаланушы интерфейсінде 3 форма болуы керек: негізгі «батырмалық» форма терезесі: ақпаратты көрсету, редакциялау формасының терезесі; ақпаратты графикалық түрде көрсету терезесі.

Сонымен, негізгі терезеде бағдарлама жұмысының режімдеріне арналған 3 батырма болуы керек:

– тіктөртбұрыштар массивін құру;

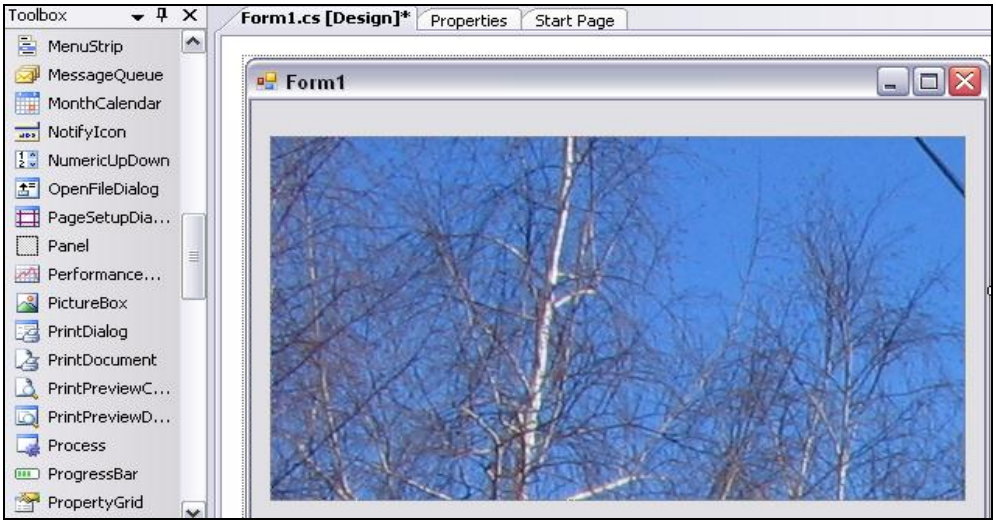
– тіктөртбұрыштар төбелері координаттарының мәндерін көрсету мен редакциялауды кесте түрінде көрсететін терезеге көшу;

– тіктөртбұрыштарды графикалық түрде көрсететін терезеге көшу.

Негізгі формаға қосымшаның авторы туралы ақпаратты көрсететін қосымша батырманы қосуға болады.

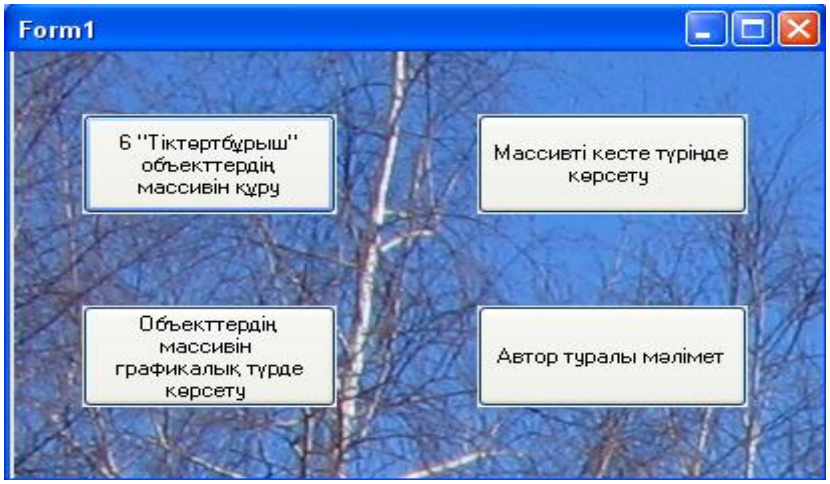
Бағдарлама жұмысының режімдерін менюді пайдалануға болады, бірақ мысал ретінде біз «батырмалық» формаларды ұйымдастыруды қарастырамыз.

Бір ғана негізгі формасы бар жобаны дайындайық. Оған суретті немесе фотосуретті орналастырайық. Ол үшін басқару элементтерінің терезесінде PictureBox элементін таңдап, оны формаға орналастыруымыз керек. PictureBox элементінің қасиеттерінде Image қасиетін таңдап, оның оң жағында кескін файлын таңдау бойынша диалогтық терезені ашу керек. Керекті кескінді тауып, оны таңдау керек. Осыдан кейін қосымшаның негізгі формасының терезесінде сурет пайда болады (20.1-сурет).



20.1-сурет – Форма терезесінде суретті орналастыру

Суреттің үстіне төрт батырманы бағдарлама жұмысының режімдеріне сәйкес орналастырамыз (20.2-сурет). Біздің қосымша үш режімде жұмыс істейді және терезеге автор туралы қосымша батырма енгізілді. Бағдарлама менюінде жұмыстың қосымша режімдері қосылады, мысалы, Help режимі – бағдарламамен жұмыс жасау бойынша нұсқау мен қосымша туралы барлық ақпаратты шығару.

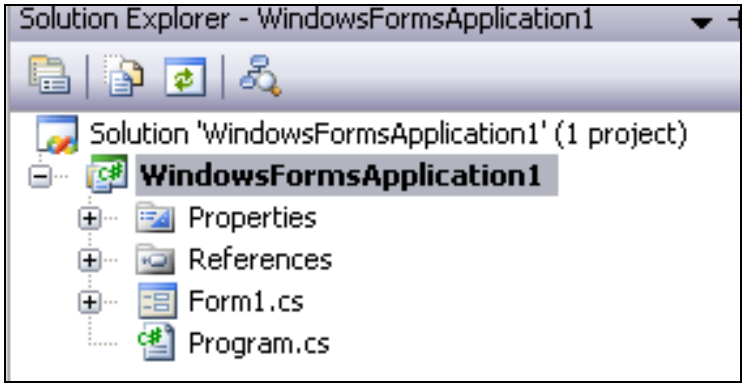


20.2-сурет – Қосымшаның негізгі «батырмалы формасы»

20.2 Қосымшаның жаңа түрлерін қосу

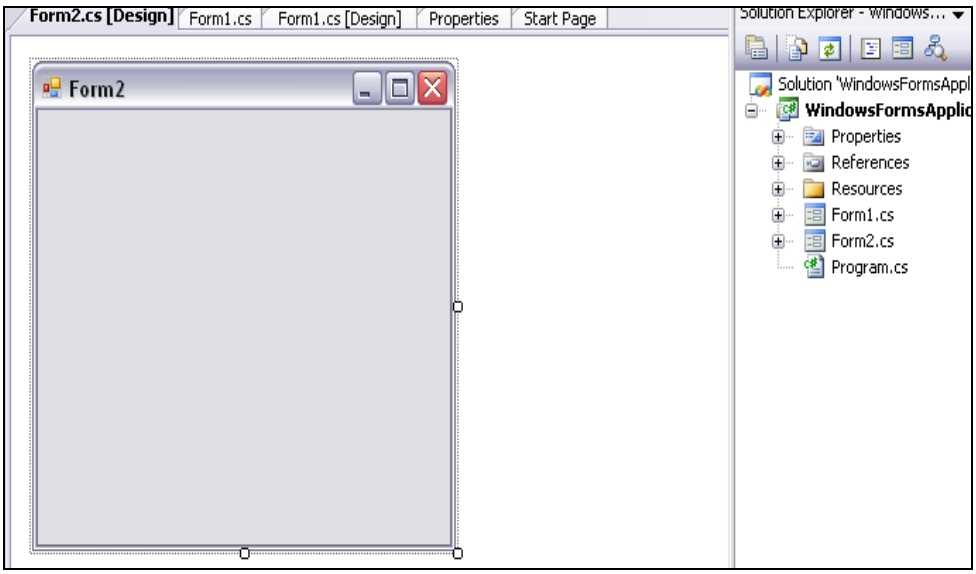
Жобаның қосымшасына жаңа формаларды қосудың бірнеше жолдары бар. Екі негізгі форманы қарастырайық.

Жобаға жаңа форманы қосу үшін Solution Explorer терезесінде жоба атауын білдіретін жолды – WindowsFormsAplication1 тышқанның оң жақ пернесімен шертіңіз (20.3-сурет).



20.3-сурет – Жобаға форманы қосудың бірінші нұсқасы

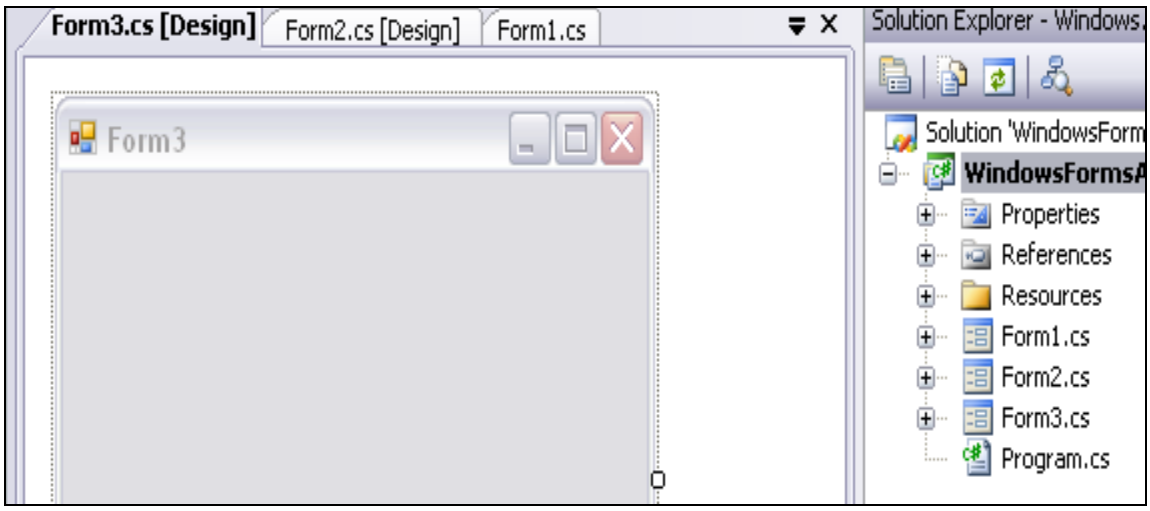
Пайда болған менюде Add режімін, оның ішінде Add Windows Form командасын таңдаңыз (20.4-сурет).



20.4-сурет – Жоба қосымшасына жаңа форманы қосу

WindowsFormsAplication1 жобасының Solution Explorer терезесінде Form2.cs жазуы пайда болды.

Екінші нұсқа бойынша Project режімін, оның ішінде Add Windows Form командасын таңдау керек. Одан кейін форма атын Add батырмасы арқылы растаңыз.



20.5-сурет – Жобаға үшінші форманы қосу

Формалардың қосылуын Solution Explorer терезесінде немесе формаларды редакциялау терезесінде тексеруге болады.

20.3 Негізгі форма оқиғаларының өңдеуіштері

6 тіктөртбұрышты массивті құру режімін негізгі формада орындауға болады, ал осы режім жұмысының нәтижелері 2 және 3-формаларда көрсетілген.

Негізгі форманың коды:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
 public partial class Form1 : Form
 {
 public static int[,] a = new int[6, 6];
 public Form1()
 {
 InitializeComponent();
 }
 private void button1_Click(object sender, EventArgs e)
 {
 Random rnd = new Random();
 for (int i = 0; i < 6; i++)
```

```
        {
        for (int j = 0; j < 4; j++)
        {
        a[i, j] = rnd.Next() % 201 - 100;
        }
        }
        for (int i = 0; i < 6; i++)
        {
        if(Math.Abs(a[i,0]-a[i, 2]) == Math.Abs(a[i, 1] - a[i,
3]))
        a[i, 4] = 1; else a[i, 4] = 0;
        a[i,5]=Math.Abs(a[i,0]-a[i,2])*Math.Abs(a[i,1]-a[i,3]);
        }
        }
        private void button2_Click(object sender, EventArgs e)
        {
        int y;
        Form2 f2 = new Form2();
        f2.ShowDialog();
        }
        private void button3_Click(object sender, EventArgs e)
        {
        Form3 f3 = new Form3();
        f3.ShowDialog();
        }
        private void button4_Click(object sender, EventArgs e)
        {
        Form4 f4 = new Form4();
        f4.Show();
        }
        }
      }
```

Массивті құру кезінде 5-ші бағанаға тіктөртбұрыш, 0- квадрат, 1 – қарапайым тіктөртбұрыш туралы мәлімет, ал 6-шы бағанаға тіктөртбұрыш ауданының мәні жазылады.

Жоба бірнеше формадан тұрады. Сондықтан, «Бірнеше форманы бір уақытта ашуға және бір формадан келесі формаға қалай өтуге болады?» деген сұрақ туындайды. Бұл сұрақтың жауабы терезенің қалай ашылғанына байланысты.

Әрбір терезені (форманы) модальді ("диалогтық терезе") немесе модальді емес (қарапайым терезе) терезе түрінде ашуға болады.

Егер терезе Show() әдісімен ашылатын болса, онда ол қарапайым терезені ашады. Егер терезе ShowDialog() әдісімен ашылатын болса, онда ол диалогтық терезені ашады. Айырмашылығы неде? Диалогтық терезеден диалогты аяқтамай, форманы жаппай шығуға болмайды.

Диалогтық терезені ашып алғаннан кейін, басқа формамен жұмыс жасауға көшуге болмайды. Диалогтық терезені жабудың бірнеше жолы

бар. Форманың жоғарғы оң жақ бұрышында орналасқан кресті немесе арнайы батырманы басуға болады.

Егер форма Show әдісімен ашылса, онда ашық формамен жұмысты аяқтамастан негізгі формаға немесе басқа модальді емес формаға көшіп, керекті мәліметті алғаннан кейін алғашқы формаға қайтып келуге болады.

Біз қосымшада терезені ашудың екі әдісін де қарастыратын боламыз.

Терезені жабу үшін екі әдісті қолдануға болады – Hide() және Close(). Осы әдістердің біріншісі форманы жасырады, екіншісі - жабады. Диалогтық терезе үшін Hide() немесе Close() әдістерін қолдануға болады: екі жағдайда да диалогтық терезе жабылады.

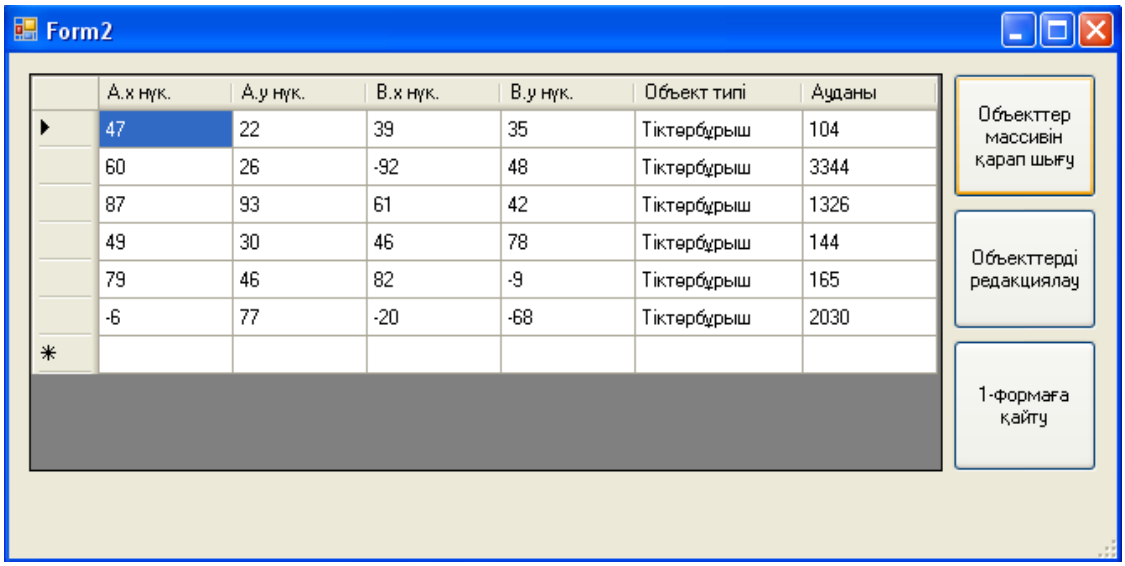
Hide() әдісін Show() әдісімен ашылған, модальді емес формаға да қолдануға болады. Hide() әдісі арқылы диалогтық емес терезені уақытша жасыруға, ал одан кейін Show() әдісін шақырып терезені ашуға болады.

Форманы жабу бойынша ескерту. Негізі форма жабылған кезде ашылған барлық формалар жабылады және қосымша өз жұмысын аяқтайды. Ал кез келген басқа форма жабылған кезде, осы форма ғана жабылады, ал басқа формалар ашық күйінде қалады..

20.4 Мәндерді кесте түрінде көрсету және редакциялау

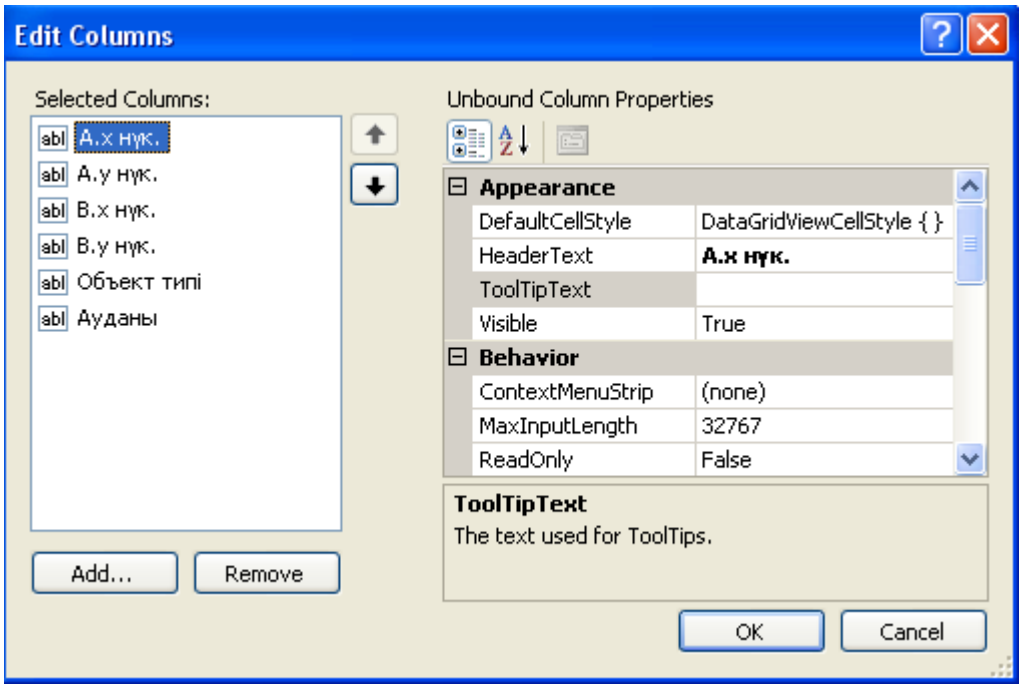
Тіктөртбұрыш төбелері координаттарының мәндерін кесте түрінде көрсетуге және редакциялауға арналған терезе оқиғаларының өңдеуіштерін қарастырайық.

Өңдеуіштер 2-формада жүзеге асырылады (20.6-сурет). Массив мәндерін кесте түрінде көрсету үшін DataGridView элементі қолданылады. Оның қолданылуын қарастырайық, осы элемент пайдаланушыға екіөлшемді массивті енгізуге, көрсетуге мүмкіндік береді.



20.6-сурет – Массив мәндерін кесте түрінде көрсету

DataGridView элементінің қасиеттерінде Columns-ты таңдаймыз, бағаналар параметрінің редакторын іске қосамыз. Редактордың көмегімен біз керекті бағаналар санын қосамыз, бағдарламада қолданылатын бағана атауын және оның формада көрсетілетін атауын анықтаймыз (20.7-сурет).



20.7-сурет – DataGridView бағаналар редакторымен жұмыс жасау

2-форманың коды :

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
 public partial class Form2 : Form
 {
 public static int i, j;
 public static string kop;
 public Form2()
 {
 InitializeComponent();
 }
 private void button1_Click(object sender, EventArgs e)
 {
 for (i = 0; i < 6; i++)
 {
 dataGridView1.Rows.Add();
```

```
      for (j = 0; j < 6; j++)
      {
     dataGridView1.Rows[i].Cells[j].Value=Form1.a[i,j].ToString
();
      }
      if (Form1.a[i,4]==0) dataGridView1.Rows[i].Cells[4].Value
= "Тіктөрбұрыш";
      else dataGridView1.Rows[i].Cells[4].Value = "Квадрат";
      }
      }
      private void button2_Click(object sender, EventArgs e)
      {
      Close();
      }
      private void button3_Click(object sender, EventArgs e)
      {
      string elem = "";
      bool ok;
      int k;
      for (i = 0; i < 6; i++)
      for (j = 0; j < 4; j++)
      {
      do
      {
      ok = true;
      try
      {
      elem = dataGridView1.Rows[i].Cells[j].Value.ToString();
      Form1.a[i, j] = int.Parse(elem);
      }
      catch (Exception any)
      {
      Form5 f5 = new Form5();
      if (f5.ShowDialog() == DialogResult.OK) k = 0;
      dataGridView1.Rows[i].Cells[j].Value = kop;
      ok = false;
      }
      } while (!ok);
      }
      for (i = 0; i < 6; i++)
      {
      if         (Math.Abs(Form1.a[i,0]-Form1.a[i,2])         ==
Math.Abs(Form1.a[i, 1] - Form1.a[i, 3])) Form1.a[i, 4] = 1;
       else Form1.a[i, 4] = 0;
      Form1.a[i, 5] = Math.Abs(Form1.a[i, 0] - Form1.a[i, 2]) *
Math.Abs(Form1.a[i, 1] - Form1.a[i, 3]);
      }
      }
      }
     }
```

2-формада 3 оқиға өңдеуіштері орындалған – DataGridView элементінің кестесіне матрицаны шығару, матрица мәндерін редакциялау және 1-формаға қайту. Кодтың әрбір үзіндісін жеке қарастырайық.

DataGridView элементінің кестесіне матрицаны шығару үшін қолданылатын циклда әрбір жаңа жол бағдарламалық түрде қосылады және онда кесте бағаналарын жекелеп қарап шығу циклі іске қосылады.

Массив 1-формада ауқымды айнымалы түрінде құрылғандықтан оны 2-формада пайдалану үшін Form1.a[i,j] жазуы қолданылады.

Екіөлшемді массивтің 4-ші бағанасында 0-ге тең мәннің болуы тексеріледі – тіктөртбұрыш қабырғалары теңдігінің сипаты (квадрат).

Соңғы бағанаға тіктөртбұрыштың ауданы шығады.

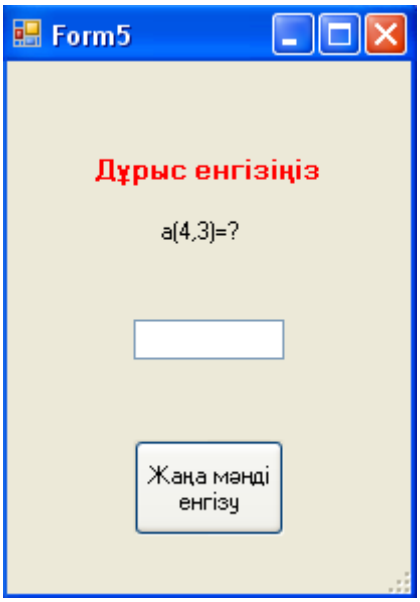
DataGridView элементін редакциялау процессінде 2-формада мәндер тікелей өзгертіледі және «Объекттерді редакциялау» өңдеуіші іске қосылады. Өңдеуіште DataGridView элементінің мәндері 1-форманың массивіне жазылады. Ол қорғалатын блокта орналасады. Егер Сіз сан орнына әріп немесе басқа символды енгізетін болсаңыз, қате орын алған жағдайдың өңдеуіші іске қосылады. Ол мәнді қайта теру үшін 5-ші диалогтық терезе шығарылады (20.8-сурет).

5-форманың коды:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
 public partial class Form5 : Form
 {
 public Form5()
 {
 InitializeComponent();
 label1.Text = "a[" + Form2.i.ToString() + "," + 
Form2.j.ToString() + "]= ?";
 }
 private void button1_Click(object sender, EventArgs e)
 {
 Form2.kop = textBox1.Text;
 Close();
 }
 }
}
```

Қатені түзету дұрыс сандық мән енгізілгенше қайталанады.

Егер редакциялау нәтижесінде тіктөртбұрыштың типі өзгерсе, мысалы, квадрат болса, онда ол 4-бағанада ескеріледі, сонымен қатар тіктөртбұрыштың ауданы қайтадан есептеліп, шығарылады.



20.8-сурет – Қатені түзету

2-форманың кодында DataGridView элементіне кезекті жаңа жол бағдарламалық түрде қосылған. DataGridView элементіне бағдарламалық түрде бағаналарды да құруға болады. Мысалы, егер DataGridView элементіне бағдарламалық түрде 6 бағананы қосу керек болса, онда кодтың келесі үзіндісін қолдануға болады:

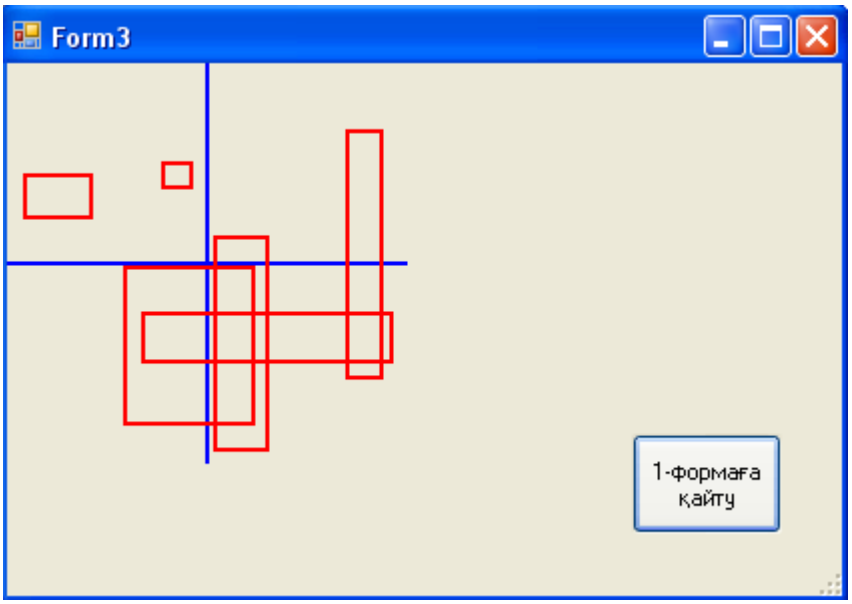
```
dataGridView1.Columns.Clear();
DataGridViewColumn column;
for (int i = 0; i < 6; i++)
{
column = new DataGridViewTextBoxColumn();
column.DataPropertyName = "Column" + i.ToString();
column.Name = "Column" + i.ToString();
dataGridView1.Columns.Add(column);
}
```

Кодтың бірінші жолында DataGridView элементінің бағаналарын жоямыз, column айнымалысын жариялаймыз, одан кейін циклде column объектісін құраймыз. Объект үшін бағдарламада қолданылатын және формаға шығатын атаулар анықталады, осыдан кейін құрылған объект элементке қосылады.

20.5 Тіктөртбұрыштарды графикалық формада көрсету

Тіктөртбұрыштарды шығаратын графикалық форма терезесіндегі оқиғалар өңдеуіштерін қарастырамыз.

Өңдеуіштер 3-формада жүзеге асырылады (20.9-сурет).



20.9-сурет – Тіктөртбұрыштарды формада көрсету

Ескерту, көрсетілген форманың дизайны жоғарғы дәрежеде емес. 3-форманың кодында ешқандай ерекшеліктер жоқ. Төменде форманың коды көрсетілген:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
 public partial class Form3 : Form
 {
 public Form3()
 {
 InitializeComponent();
 }
 private void button1_Click(object sender, EventArgs e)
 {
 Close();
 }
 private void Form3_Paint(object sender, PaintEventArgs e)
 {
 int ax, ay, bx, by;
 Pen myPen = new Pen(Color.Blue, 2);
 Graphics g = e.Graphics;
 g.DrawLine(myPen, 0, 100, 200, 100);
 g.DrawLine(myPen, 100, 0, 100, 200);
 myPen = new Pen(Color.Red, 2);
```

```
 for (int i = 0; i < 6; i++)
 {
 if (Form1.a[i, 0] < Form1.a[i, 2]) ax = Form1.a[i, 0];
  else ax = Form1.a[i, 2];
 if (Form1.a[i, 1] < Form1.a[i, 3]) ay = Form1.a[i, 1];
  else ay = Form1.a[i, 3];
 bx = Math.Abs(Form1.a[i, 0] - Form1.a[i, 2]);
 by = Math.Abs(Form1.a[i, 1] - Form1.a[i, 3]);
 g.DrawRectangle(myPen, ax+100, ay+100, bx, by);
 }
 }
 }
}
```

Жобаға «Автор туралы ақпарат» формасы да қосылған.

20.10-сурет – «Автор туралы ақпарат» режімінің терезесі

4-форманың кодында 1-формаға көшетін батырманың өңдеуіші ғана жазылған. Қалған әрекеттер 4-форма элементтерінің қасиеттері арқылы қосылған.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
 public partial class Form4 : Form
```

```
 {
 public Form4()
 {
  InitializeComponent();
 }

 private void button1_Click(object sender, EventArgs e)
 {
  Close();
 }
 }
}
```

20.6 Өзін-өзі тексеру сұрақтары

1 MDI қысқартуы нені білдіреді?

2 Қандай жағдайларда батырмасы бар форманы жобалау керек?

3 Формада суреттерді орналастыру үшін әдетте қандай басқару элементі қолданылады?

4 Solution Explorer терезесі арқылы жобаға жаңа форманы қалай қосуға болады?

5 Project режімі арқылы жобаға жаңа форманы қалай қосуға болады?

6 Диалогтық (модальді) терезенің форманың қарапайым терезесінен айырмашылығы қандай?

7 Форманың қарапайым (модальді емес) терезесі қандай әдіс арқылы ашылады?

8 Форманың модальді терезесі қандай әдіс арқылы ашылады?

9 Бағдарламаның келесі үзіндісі нені орындайды:

```
private void button3_Click(object sender, EventArgs e)
 {
        Form5 f5 = new Form5();
        if (f5.ShowDialog() == DialogResult.OK) k = 0;
 } ?
```

10 Ақпаратты кестеде шығару үшін қандай басқару элементі жиі қолданылады?

21. КЛАСС ҰҒЫМЫ

21.1 Класс ұғымы

C# тілі объекті-бағытталған бағдарламалау тілі болып табылады. Оның негізгі ұғымы - класс. C# тілін оқу барысында барлық мысалдарда біз класс типіндегі құрылымдарды қолданған болатынбыз. Осы бөлімде тек қана кластар қарастырылады.

C# тілінде бағдарламалауға арналған кітаптарда келтірген анықтамалардан бастайық.

Фаронов В.В. анықтамасы бойынша класс алдында class қызметтік сөзі тұратын код үзіндісі арқылы анықталады [1]: «Класс дегеніміз – деректер типі, яғни кластың нақты даналары - объекттерді дайындалатын «схема».

Павловская Т.А. [2] анықтамасы: «Класс дегеніміз кластың даналары деп аталатын нақты объекттер жиынтығының сипаттамалары мен әрекеттерін анықтайтын жалпылама ұғым».

Ескеретін жағдай, C# тілі пайда болуына дейін ОББ бар болған және класс ұғымы бұрыннан қолданылады. Әдебиетте кездескен ең қысқа анықтаманы келтірейік: «Кластар дегеніміз – бағдарлама жасаушы анықтайтын тип».

Осы анықтамада кластың өте маңызды ерекшелігі көрсетілген – ол – массивтерге, жазуларға немесе құрылымдарға қарағанда жаңа деректер типі. Бірақ, бағдарламашы анықтайтын кез келген тип класс бола бермейді. Класты анықтаған кезде екінші бір маңызды ерекшелігі - класс құрамының болуы, кластың қысқа түрде жазылуы және оның оңай есте сақталуы.

`Класс дегеніміз – өрістерден, әдістерден және оқиғалардан тұратын деректер типі.`

Деректер типі дегеніміз – класс данасы деп аталатын көптеген объекттердің қасиеттері мен әрекеттерін сипаттайтын семантикалық бірлік.

Семантикалық класс класс өрістері, класс әдістері мен оқиғаларының сипаттамасы деп аталатын деректер сипаттамасын ұсынады.

Кейбір авторлар модуль болып келетін кластарды жеке топтарға бөледі, мысалы, басқару элементтерінің класы. Ондай кластардың қосымша қызметтері бар. Олар жоба құрылымының жеке архитектуралық бірлігі болып келеді.

Кластың жазба пішімін қарастырайық. Класс жазбасының пішімінде class қызметті сөзінен кейін оның атауы және одан кейін фигуралы жақшаларда класс денесі жазылады. Бұл класс сипаттамасының ең қысқа құрамы болып келеді. Кластың жалпы сипаттамасы мына пішімде болады (міндетті емес элементтер квадратты жақшаларда көрсетілген):

```
[ атрибуттар ] [ спецификаторлар ]
```

```
class кластың_атауы [ : түп тегі ]
 { кластың_денесі } ,
```

мұнда

атрибуттар – класс туралы қосымша мәліметті береді;

спецификаторлар – класс құрамына қол жеткізу шарттарын анықтайды.

`түп тегі` (родители) – базалық кластар;

класс денесі – класс элементтерінің құрамын анықтайды.

Кластыжариялауда мүмкін спецификаторлар: abstract, sealed және protected. Олар туралы мұрагерлікті қарастырған кезде толығырақ айтылатын болады. Private, public, static және internal спецификаторлары бағдарлама үшін кластың қолжететімділігін анықтайды. Private спецификаторы кластың көрінуін толық жабады, ал public спецификаторы класты бағдарламаның кез келген үзіндісіне көрсетеді (қол жететіндікті анықтайды). Негізінде класта internal қол жеткізу спецификаторы болады. Класс құрылымда анықталған және онда қол жетімді болады. Static спецификаторы осы класқа тиісті айнымалыны (класс объектісін) құрмай-ақ класс және оның элементтерін пайдалануға мүмкіндік береді.

Барлық спецификаторларды класта немесе оның жеке мүшелерінде қолдануға болады, мысалы, өрістерде, әдістерде.

Класты сипаттау пішімінің кейбір міндетті емес элементтерін біз келесі бөлімдерде қарастыратын боламыз.

Класс дегеніміз –белгілі бір мәндермен «толтырылатын» үлгі, яғни класс типіндегі айнымалы – класс данасын әзірлеуге арналған деректер типі.

Бағдарламада түрлі мәндерді бере отырып біз кластың әр түрлі объекттерді көретін боламыз, бірақ кластың типі өзгермейді.

Класты атаулар кеңістігінің ішінде немесе басқа кластың ішінде сипаттауға болады. Соңғы жағдай бойынша класты қабаттасқан класс деп атайды.

C# тілінде класс сілтемелік тип болып келеді және класс объектісін компьютер жадысында орналастыру үшін new операторын қолдану керек.

21.2 Класс құрамы

Класс денесінде деректер, әдістер және оқиғалар өңдеуіштері болуы мүмкін. Кластың осы құрамдас бөліктерін әдетте класс элементтері деп атайды.

Кластың негізгі элементтерін және олардың қызметтерін қарастырайық:

–класс тұрақтыларында өзгермейтін мәндер;

–класс өрістері (класс айнымалыларының типтері мен атаулары);

– класс әдістері, класс деректерімен жұмыс жасауға арналған, белгілі бір атауы бар бағдарлама кодының үзіндісі;

– класс қасиеттері дегеніміз - кластарға өз өрістерінің мәндерін бір-бірімен алмасуға (оқуға немесе жазуға) мүмкіндік беретін әдістердің жиыны;

– класс конструкторы дегеніміз – класс объекттерін құруға және класс өрістеріне мәндерді меншіктеуге арналған кластың арнайы әдістері;

– класс деструкторы дегеніміз – объектке бөлінген ресурстарды босату кезінде әрекеттердің тәртібін анықтайды;

– класс оқиғалары дегеніміз – кластың пайдаланушы әрекеттеріне немесе бағдарламадағы белгілі бір өзгерістерге жауап қайтаруға көмектесетін арнайы әдістер;

– деректер типі, мысалы, тізімдер, құрылымдар, кластар, делегаттар, интерфейстер.

– индексаторлар дегеніміз – класс деректерінің элементтеріне қол жеткізу құралы;

– операциялар дегеніміз – класс объекттеріне арналған, операциялардың белгілері арқылы орындалатын арнайы әрекеттер.

Класс деректері болып константа немесе класс айнымалылары (өрістер) болуы мүмкін. Класта деректерді жариялаған кезде әдетте оған қол жеткізу спецификаторы көрсетіледі, мысалы,

```
private int a;
```

Класс деректерін жариялау кезінде оларды жалпы жазу пішімі мынандай болады:

```
[ атрибуттар ] [ спецификаторлар ]
[ const ] тип атауы [= бастапқы_мәні].
```

Әдетте класс деректері «бағдарлама үшін жабық болады» - private спецификаторы қолданылады. Егер деректердің алдына public спецификатор жазылса, онда олар «бағдарламада» қолжетімді болады.

Еш әрекетсіз деректер мен әдістер үшін private спецификаторы қолданылады.

Объект дегеніміз – класс типіндегі айнымалы, оны құрған кезде компьютер жадында класс элементтерінің мәндері сақталатын жеке аймақ бөлінеді.

Алайда, класта барлық объекттеріне ортақ, бір данадағы статикалық элементтер болуы мүмкін. статикалық деректерді класс деректері деп жиі атайды, ал қалғандарын - класс данасының деректері, яғни объекттер.

Кластың кейбір элементтеріне (әдістеріне) және өрістеріне қол жеткізуге объект құрылғаннан кейін ғана мүмкін болады. Егер рұқсат берілген болса, оларды қолдану үшін нүкте қолданылады, мысалы, stud объектісіндегі name өрісіне қол жеткізу үшін былай жазамыз: stud.name = “Иванов”;.

Объект үшін кластың әдісін шақыруға болады, мысалы, stud.poisk(a);, мұнда poisk(int a) – класс әдісі, stud – құрылған объект.

Синтаксис бойынша класта ішкі класс болуы мүмкін. Осындай жағдай жиі кездеспейді. Ішкі класты өзін туындатқан сыртқы класта және оның ұрпақтарында қолдануға болады. Ішкі кластарда әдетте private немесе protected қол жеткізу модификаторы болады.

21.3 Класс әдістері

Әдіс дегеніміз – кластың деректері және әдістерімен жұмыс істеуге арналған кластың атауы бар функционалдық элемент. Әдістер класқа қолдануға болатын әрекеттер жиынын анықтайды (кластың жұмысын анықтайды). Әдіс тек бір рет сипаталады және ол кластың түрлі объекттері үшін бірнеше рет шақырылуы мүмкін.

Класс әдістерінің жалпы жазылу пішімі мына түрде болады:

```
[ атрибуттар ] [ спецификаторлар ] әдіс типі
әдістің атауы ( [ параметрлер ] )
        {әдістің денесі}
```

Мысалы,

```
static void Main(string[] args)
{ }
```

Ең жиі кездесетін спецификаторлар - private, public және static.

Private спецификаторымен жарияланған кластың кез келген әдісі тек осы класс әдістеріне ғана қолжетімді болады.

Public спецификаторы әдісті бағдарламаның кез келген бөлігінде қолдануға мүмкіндік береді.

Static спецификаторы арқылы әдісті класс объектісін құрмай-ақ «класс деңгейінде» қолдануға болады. Ол өте маңызды, өйткені біз статикалық әдістерді жиі пайдаланатын боламыз.

Қосымшада класс объектісін құрмай-ақ қолжетімді әдісті класс конструкторы қамтамасыз етеді(ол объектті құрады).

Басқа әдістерге қол жеткізу кластың объектісін құрғаннан кейін ғана мүмкін.

Егер спецификатор көрсетілмесе класс әдісінде private спецификаторы қолданылады.

Әдіс типі қосымшада анықталған кез келген типте немесе C# тілінің стандартты типінде немесе void – типсіз болуы мүмкін. Мысалы:

```
int kol(int a) { … }
public double sym(out float r) { … }
public void poisk(ref float s) { … }
public int funkcij( int a, out int b, params int[] c) { … }
```

Егер әдіс типі көрсетілген болса (void типінен өзге), онда әдіс денесінің соңғы операторы болып әдіс жұмысының нәтижесін қайтаратын return операторы болады. Бұл ретте әдісті айнымалыға меншіктеу немесе операторларда өрнек ретінде пайдалану керек. Осындай әдістер функциялар деп аталады.

Егер әдістің алдында void типі көрсетілсе, онда әдіс өз жұмысының нәтижесін return операторы арқылы қайтармауы керек (әдістің денесінде return операторы болмайды). Әдетте бұл әдісті процедура деп атайды. Әдістің атауы – бағдарламашы белгілейтін идентификатор. Әдіс атауының (атының) мағынасы оның жұмысына байланысты болуы керек, мысалы, sym, max, poisk, т.б.

Әдіс және бағдарлама арасында деректермен алмасу үшін әдістің параметрлері (формалды параметрлер) қолданылады. Әдетте әдістің параметрлерін әдісті «күйге келтіру» құралы деп атайды.

C# тілінде әдістердің келесі параметрлері бар:

– мәндерді анықтайтын параметрлер (мәндік параметрлер, яғни әдіс қабылдайтын кіріс параметрлер);

– шығыстық параметрлер (`out` қызметтік сөзімен белгіленеді);

– сілтемелік параметрлер (`ref` қызметтік сөзімен белгіленеді);

– массивті параметр (`params` қызметтік сөзімен белгіленеді).

Мәндерді анықтайтын параметрлерде қызметтік сөз болмайды.

Класс әдістерінің параметрлері үтірлер арқылы бөлінеді. Әдісте массив параметрі біреу және параметрлер тізімінде соңғы болуы керек.

Егер әдісте мәндерді анықтайтын параметрлер жарияланса, онда бұл әдістің кейбір айнымалылар көшірмелерін өз құзырына алғандығын көрсетеді. Әдіс осы көшірмелердің мәнін өзгерте алады, бірақ олардың түпнұсқасы (бағдарламада) өзгермеген қалыпта қалады. Әдістің жұмысы аяқталғаннан кейін мәндерді анықтайтын параметрлер компьютер жадысынан жойылады.

Әдістің шығыстық параметрлері қосымшаға нәтижелерді жеткізу үшін арналған. Әдістің денесіндегі шығыстық параметрлерге кейбір мәндер меншіктелуі тиіс, әйтпесе қосымша компиляциясы кезінде қате кеткені туралы хабар шығады.

Егер әдісте сілтемелік параметрлер жарияланған болса, онда әдіс сәйкес айнымалылардың адресін өз құзырына алады және оларды өз алгоритмі бойынша қолдана алады (жаңа мәндерді жаза және оқи алады).

Әдістегі жарияланған массив параметрі нақты айнымалылардың кез келген санымен жұмыс жасауға арналған. Сонымен қатар `params` қызметтік сөзінен кейін тұрған формалды параметр кез келген өлшемді деректер массивімен сәйкестікке келтіріледі.

Сонымен, әдіске параметрлер арқылы керекті мәліметтерді (мәндерді анықтайтын параметрлер және сілтемелік параметрлер) жіберуге болады және әдіс өз жұмысының нәтижелерін қайтара алады (шығыстық параметрлер және сілтемелік параметрлер).

Әдіс денесінде кейбір алгоритмді орындайтын бағдарлама кодының үзіндісі бар. Бұл ретте әдіс формалды параметрлермен бірге әрекеттер үлгісі ретінде қолданылады. Бағдарламада формалды параметрлердің орнына нақты айнымалылар қолданылуы керек, нақты параметрлер мен әдістің әрекеттер үлгісі нақты айнымалылар үшін қолданылады.

21.4 Объект құрылымы

Класс типіндегі айнымалы объект деп аталады. Объект айнымалы болғандықтан оған компьютер жадысынан орын бөлінеді.

Объект бойынша жадыда нақтылы не сақталатынын қарастырайық.

Класс деректерінің барлық өрістерінің мәндері сақталады.

Объектті дайындаған кезде автоматты түрде құрылатын this арнайы өрісі (сілтеме бойынша параметр) объект адресін сақтайды.

Объект және класс әдістерінің байланысы this параметрі арқылы жүзеге асады. Кластың әрбір әдісі ағымдағы объект элементтерімен жұмыс жасау үшін this параметрін тікелей қолдана алады. this ағымдағы объектке (ағымдағы уақытта бағдарлама жұмысындағы объект) үнемі сәйкес болғандықтан, класс әдістері ағымдағы объект элементтерімен жұмыс жасайды.

This нұсқауышын пайдалану класс әдістерінің көшірмесін әрбір объект үшін жасамауға мүмкіндік береді. Сонымен класс әдістері әрбір объект үшін тираждалмайды.

21.5 Қосымшаны құру мысалы

Ұшбұрыш класын құру бойынша мысалды қарастырайық (бірінші бөлімдегі мысалдың негізінде).

Визуалды бағдарламалау кезеңінде Toolbox терезесінен көшірілген стандартты басқару элементтерін қолданатын боламыз: тұрақты мәтін (Label), енгізу өрісі немесе редакциялау терезесі (TextBox) және командалық батырма (Button).

Form1.cs файлының коды:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsApplication1
{
 public partial class Form1 : Form
```

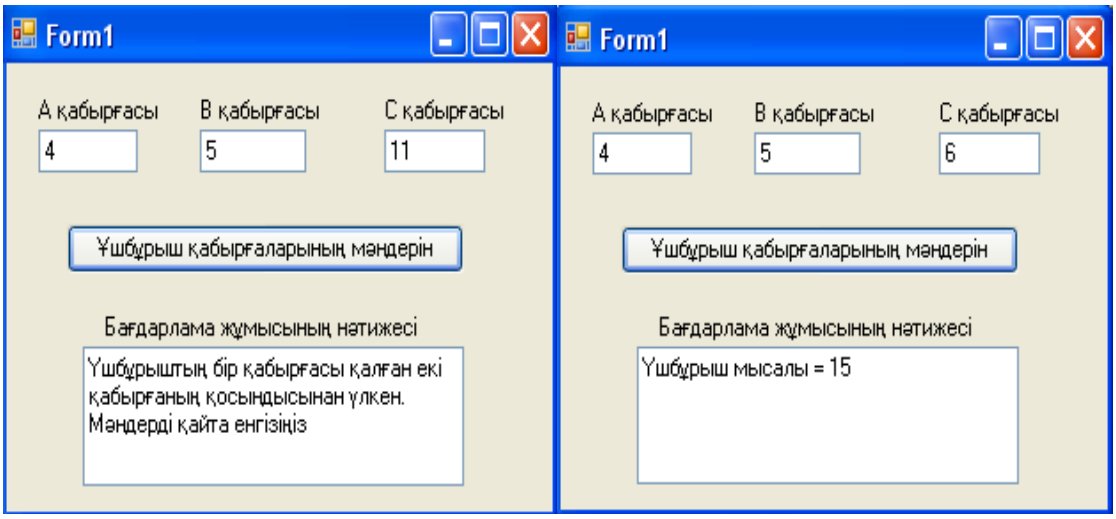
```
      {
      public class treyg
      {
       private int a, b, c, p;
       public string ss;
       public void vvod(int sa, int sb, int sc)
       {
       if (sa > 0 && sb > 0 && sc > 0)
       {
        if (sa + sb > sc && sa + sc > sb && sb + sc > sa)
        {
        a = sa; b = sb; c = sc;
        p = a + b + c;
        ss = "Үшбұрыш мысалы = " + p.ToString();
        }
        else
        ss = "Үшбұрыштың бір қабырғасы қалған екі қабырғаның
қосындысынан үлкен. Мәндерді қайта енгізіңіз";
       }
       else
        ss = "Үшбұрыштың бір қабырғасының ұзындығы 0-ден кіші!
Мәндерді қайта енгізіңіз";
       }
      }
      public Form1()
      { InitializeComponent();
      }
      private void button1_Click(object sender, EventArgs e)
      {
       int A, B, C;
       treyg t = new treyg();
       A = Convert.ToInt32(textBox1.Text);
       B = Convert.ToInt32(textBox2.Text);
       C = Convert.ToInt32(textBox3.Text);
       t.vvod(A, B, C);
       textBox4.Text = t.ss;
      }
      }
     }
```

Кластың кейбір элементтерін және олардың бағдарламада қолданылуын қарастырайық.

class treyg класының деректерімен және әдістерімен жұмыс орындау үшін осы кластың объектісін– t айнымалысын құру керек.

```
treyg t = new treyg();
```

Кластың private int a, b, c, p; деректері жабық болады. Класс деректерінің элементтеріне оның әдістері арқылы ғана қол жеткізуге болады.



21.1-сурет – «Треугольник Класс» қосымшасының терезесі

Мысалы, егер t объектісін құрғаннан кейін деректер элементіне b (t.b = 3;) жаңа мәнді меншіктейтін болсақ, онда бұл әрекет қате кеткені туралы хабарды шығарады, өйткені класс деректерінің элементтеріне тікелей байланысуға private қол жеткізу спецификаторы рұқсат бермейді.

treyg класында екі әдіс қолданылады – конструктор және ұшбұрыш қабырғаларының мәндерін енгізу әдісі.

Конструктор өрістерінің мәндері «нөлдік» болатын объектті құрайды.

treyg класының өрістерінің мәндерін анықтау public void vvod(int sa, int sb, int sc) арқылы орындалады, әдістегі нақтылы параматрлері ретінде диалог режімінде енгізлген A, B, C айнымалыларының мәндері беріледі.

Бағдарламада үшбұрыш қабырғаларының мәндерін дұрыс енгізбеу және сәйкес түсініктемелерді экранға шығару жағдайлары қарастырылған. Бірақ мәтіндік өрістер бос болып, «Үшбұрыш қабырғаларының мәндерін енгізу» батырмасын басуға қатысты қорғаныс қарастырылмаған.

21.6 Өрістерге қол жеткізу

Әрбір өрісте қол жеткізу модификаторы болады, олардың мүмкін мәндері: public, private, protected, internal. protected және internal атрибуттары бірге қолданылуы мүмкін

Private модификаторы. Әдетте егер ешқандай модификатор көрсетілмесе, онда Private модификаторы қолданылады. Ол өрістерді басқа барлық кластардан жабық ұстайды және кластың өз әдістеріне ғана тікелей қол жеткізуге рұқсат береді (оқу, жазу).

Ескерту, барлық өрістер кластың барлық әдістеріне қол жетімді болады. Олар кластың әдістері үшін ауқымды ақпарат болып табылады,

олармен барлық әдістер жұмыс істейді, яғни өрістерден керекті ақпаратты алады және оларды өзгертеді.

Protected модификаторы. Бұл модификатор мұрагер кластарға өрістерді ашық ұстайды. Егер А класы модификаторы protected болатын өрісті жарияласа, онда А класының мұрагері – В класының әдістері А класының өрістерін мұраға алады және осы өрістермен тікелей жұмыс істей алады.

Internal модификаторы. Бұл модификатор бойынша ынтымақтас кластарға өрістер ашық болады. Егер А және В кластары бір құрылымға (сборка) – бір жобаға тиісті болса, онда бұл кластар ынтымақтас кластар деп аталады. Егер А класы белгілі бір өрісі internal модификаторымен жарияласа, онда А класының клиенті болып келетін ынтымақтас В класының әдістері ондай өріспен тікелей жұмыс жасай алады.

Protected және internal атрибуттарының құрамдастырымы (комбинация). Бұл құрамдастырым мұрагер немесе ынтымақтас кластарға өрістерді ашық ұстайды. Егер ынтымақтас кластар болып келетін мұрагерлерге ғана қолжетімді болуы үшін өрістерге қол жеткізуге қатаң шектеу қою керек болса, онда кластың өзін internal модификаторымен, ал сәйкес өрісті protected модификатором жариялау керек. Егер өрістер класс әдістеріне ғана қолжетімді болса, онда олар private қол жеткізу модификаторымен бірге жарияланады. Бұндай өрістер жабық өрістер деп аталады, бірақ әдетте олардың кейбірі басқа кластарға ашық болуы керек болады.

Егер А класының кейбір өрістері А класының мұрагері болып келетін В класының әдістеріне қолжетімді болу керек болса, онда осы өрістерді protected модификаторымен жариялау қажет. Бұндай өрістер қорғалған өрістер деп аталады. Егер кейбір өрістер А класына ынтымақтас В1, В2, т.б. кластарының әдістеріне қол жетімді болу керек болса, онда ол өрістерде `internal` модификаторын пайдалану қажет, ал барлық ынтымақтас кластарын бір жобаға (assembly) орналастыру керек. Ондай өрістер ынтымақтас өрістер деп аталады. Сонымен, егер кейбір өрістер кез келген В класының (В класына А класы қолжетімді) әдістеріне қолжетімді болу керек болса, онда осы өрістерде public модификаторы болуы керек. Мұндай өрістер ашық және көпшілік қолды өрістер деп аталады.

21.7 Өзін-өзі тексеру сұрақтары

1 Класс ұғымы
2 Класс қасиеттері туралы ұғым?
3 Класс конструкторы туралы ұғым?
4 Класс деструкторы туралы ұғым?
5 Класс оқиғасы туралы ұғым?
6 Класс индексаторы туралы ұғым?

7 Объекттің this өрісінің міндеті?

8 Класс типіндегі айнымалы қалай аталады?

9 Класс сипаттамасында static қызметтік сөзі нені білдіреді ?

10 Класс деректерінің сипаттамасында public қызметтік сөзі нені білдіреді?

22 КЛАСС ЭЛЕМЕНТТЕРІ

22.1 Конструкторлар

Конструктордың негізгі қызметі – объектті құру және осы объектің деректер элементтеріне белгілі бір бастапқы мәндерді меншіктеу. Кейде бұл процесс объект инициализациясы деп аталады.

Объектті құру конструктордың негізгі міндеті болып табылады.

Объекттің деректер элементтеріне кейбір мәндерді меншіктеу түрлі жолдармен орындалуы мүмкін. Объекттегі элементтер мәндерінің анықталуына қарай конструкторлардың мына түрлері болады:

– параметрсіз конструкторлар;

– параметрлері берілген конструкторлар;

– жиынтық (множественные) конструкторлар.

Параметрсіз конструкторлар бағдарламада параметрсіз шақырылады. Мысалы,

```
treyg t1 = new treyg();
```

Жоғарыдағы мысалда t1 объектісінің деректер элементтеріне белгіленген мәндер меншіктеледі (әдетте нөлдік мәндер).

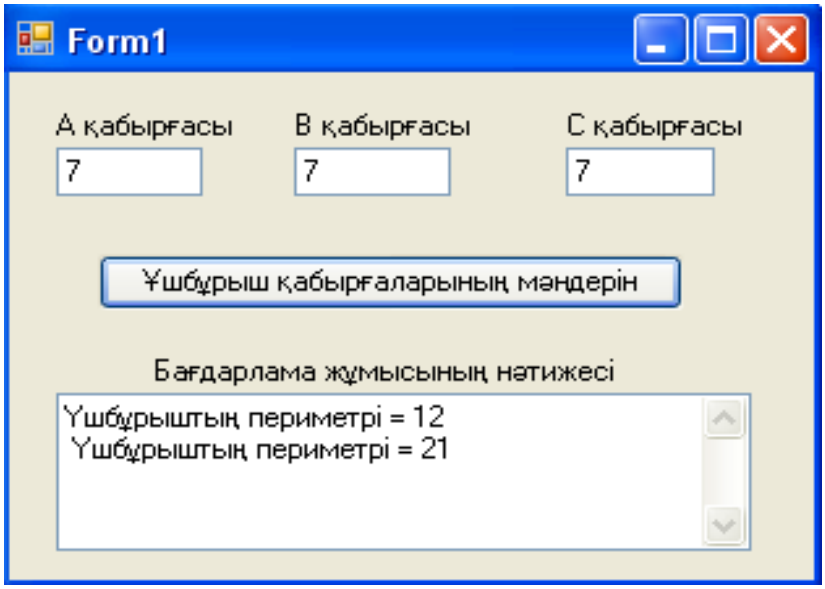
Осындай конструктор жүзеге асырылғанда деректер өрістеріне белгілі бір мәндер жиынын меншіктейтін операторлар қолданылуы мүмкін.

Мысалы, `treyg` класына параметрсіз конструкторды қосуға болады, конструктор үшбұрыш қабырғаларына белгіленген 3, 4 және 5 мәндерін меншіктейді. Осы жағдайды объект периметрін экранға шығарып тексеруге болады:

```
public treyg()
 {
 a = 3; b = 4; c = 5;
 }
 public void print0()
 {
  p = a + b + c;
  ss = "Үшбұрыштың периметрі = " + p.ToString();
 }
```

Объектті құру барысында объекттің деректер элементтерінің бастапқы мәндерін параметрлі конструкторлар анықтауға мүмкіндік береді. Осындай конструкторлардың класта жариялануын мына түрде көрсетуге болады:

```
public treyg(int sa, int sb, int sc)
 {
 a = sa; b = sb; c = sc;
 }
```



22.1-сурет – Қосымшада белгіленген бастапқы мәндері бар конструкторды қолдану

Көптеген бағдарламашылар конструкторды құрған кезде оған объекттердің элементтер мәндерінің дұрыстығын тексеруді қосады, ондай конструкторлар «ақылды» конструкторлар деп аталады. Мысалы, үшбұрыш класының конструкторын дайындаған кезде деректер элементтері бойынша мына тексеру жұмыстарын қосуға болады:

– үшбұрыштың барлық қабырғалары 0-ден үлкен болуы тиіс;
– үшбұрыштың кез келген екі қабырғасының қосындысы үшінші қабырғасынан үлкен болуы керек.

Егер шарттар орындалмайтын болса, хабарламаны шығарып, объект деректерінің элементтеріне белгіленген мәндерді меншіктеу керек немесе мәндерді қайта енгізуді орындау керек.

Осындай «ақылды» конструкторды жариялаудың қарапайым конструкторды жариялаудан айырмашылығы жоқ, ал оны құрған кезде класс деректерін енгізу әдісінде қажетті барлық тексерістер жүргізіледі, мысалы:

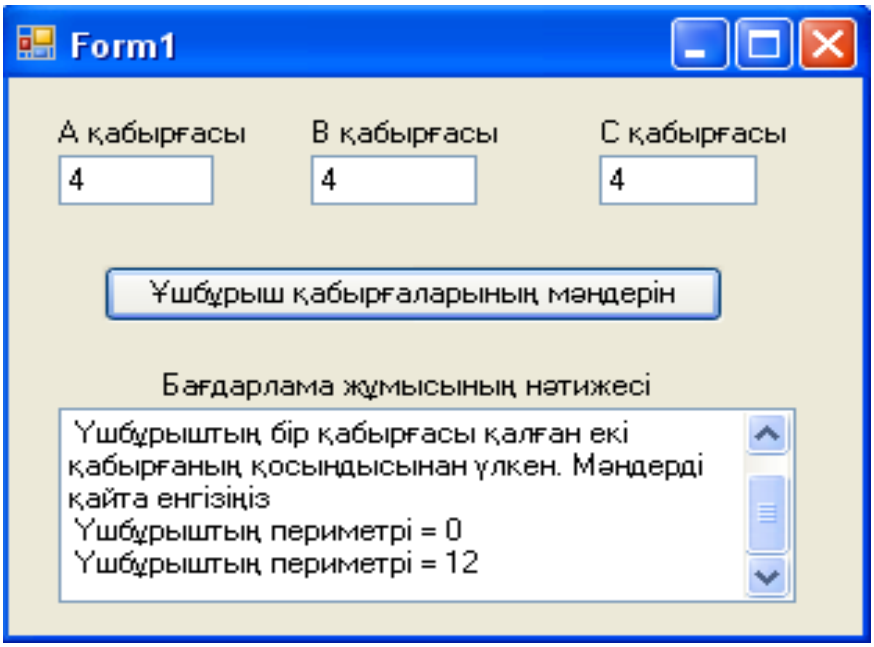
```
public treyg(int sa, int sb, int sc)
 {
 vvod(sa,sb,sc);
}
```

Оқиға өңдеуішінің бағдарлама кодын келесі түрде өзгертейік – конструкторға алдын ала үшбұрыш қабырғаларының қате мәндерін енгізейік:

```
private void button1_Click(object sender, EventArgs e)
```

```
 {
 int A, B, C;
 treyg t = new treyg(3,5,9);
 t.printO();
 A = Convert.ToInt32(textBox1.Text);
 B = Convert.ToInt32(textBox2.Text);
 C = Convert.ToInt32(textBox3.Text);
 t.vvod(A,B,C);
 textBox4.Text = t.ss;
}
```

Бағдарламаның жұмысы 22.2-суретінде көрсетілген.



22.2-сурет – Конструкторға мәндерді алдын-ала қате берген жағдайдағы қосымшаның жұмысы

Жиынтық конструкторлар түрлі типтегі аргументтерді өңдеген кезде қайта жүктелетін функцияны қолданады.

Осындай конструкторлар деректердің мәндері әртүрлі формада берілгенде қолданылады, мысалы, бүтін, нақты сандармен немесе жолмен берілсе.

Әдетте мұндай конструкторлар датаны өңдеу үшін қолданылады.

Қайта анықтау функциясын қолдана отырып класс сипаттамасына түрлі типтегі деректерді «түсінетін» бірнеше конструкторларды қосуға болады.

Мысалы, ағымдағы датаны түрлі тәсілдермен енгізуге болады. Мысалы, датаны енгізудің үш нұсқасы бар:

– бүтін сандармен (айы, күні, жылы – 23 12 07);
– сөздермен (23 желтоқсан 2007 жыл);

– қосымша символдармен қосып жазу (20.10.07 немесе 17/09/2007).

```
class date
 {
 . . .
 date() { . . . }
 date(int mm, int dd, int gg) { . . . }
 date (string tekct) { . . . }
 . . .
};
```

Объектіні инициализациялау кезінде кез келген ұсынылған нұсқаны қолдануға болады.

22.2 Деструкторлар

C# тілінде кластарда арнайы әдістер – деструкторлар болуы мүмкін. Қызметі жағынан қарастырғанда олар конструктордың әрекеттеріне қарама-қарсы әрекеттерді орындауы керек. Бірақ C# тіліндегі деструкторлардың ерекшелігі – олар сәйкес объекттердің конструкторына бөлінген жадыны босатуды орындамайды (оны қоқыс жинаушы бақылайды), бірақ объектке бөлінген ресурстарды босатады, мысалы, деректер базасының сервермен, басқа компьютермен байланысты тоқтатады.

Конструктор сияқты деструктордың атауы кластың атауымен бірдей болады, бірақ деструктор атауының алдында «~» – тильда символы қойылады. Мысалы, егер treyg класында деструктор болса, онда оның жазылуы мына түрде болады:

```
public ~treyg()
 { деструктор_денесі }
```

Деструкторды бағдарламада тікелей шақыруға болмайды, оны үйіндіден объектті жойған кезде автоматты түрде «қоқыс» жинаушысы шақырады. Іс жүзінде объектке бөлінген ресурстардың қажеттілігі жоқ болғаннан кейін ресурстарды босату автоматты түрде орындалады. Сондықтан деструкторлар класс құрылымында конструктивті түрде болады, бірақ әдетте құрылмайды, әдепкідей қолданылады.

22.3 Қасиеттер

Объекті-бағытталған бағдарламалау принциптерінің бірі – инкапсуляция. Инкапсуляция дегеніміз – деректерді қосымшадан тікелей қол жеткізуден қорғау мақсатында класс өрістері мен әдістерінің бірлесуі. Объект өрістері объект интерфейсі - қол жеткізу ережелер жинағы немесе

қасиеттер арқылы қолданылады. Инкапсуляция («капсула» сөзінен) – объект өрістерін жасыру қасиеттер арқылы жүзеге асырылады.

Қасиеттер get() және set() арнайы екі әдістерінен және объект өрісінен тұрады. Get() әдісі объекттің сәйкес өрісінің ағымдағы мәнін қайтарады, ал set() әдісі объект өрісіне жаңа мәнді енгізеді.

Қосымша «қасиеттің» мәнін алғанда немесе өзгерткенде осы әдістер автоматты түрде шақырылады. Синтаксисі бойынша қасиеттердің өрістерден өзгешелігі жоқ, олар меншіктеу операторының сол жағында орналаса алады немесе оператордың оң жағында өрнектің мүшесі ретінде жазыла алады. Мысалы, int a жабық өрісіне бүтін типтегі келесі Aa қасиетін жазуға болады.

```
public class treyg
{
private int a, b, c, p;
public int Aa
{
get { return a;}
set { a = value;}
}
public string ss;
. . .
```

Қосымшада t объектісін құрғаннан кейін a өрісіне Aa қасиетінінің көмегімен жаңа мәнді меншіктеуге болады:

```
int A, B, C;
 treyg t = new treyg();
 A = Convert.ToInt32(textBox1.Text);
 B = Convert.ToInt32(textBox2.Text);
 C = Convert.ToInt32(textBox3.Text);
t.Aa = A; ,
```

немесе A айнымалысына t объектісінің a өрісінің мәнін меншіктеуді мына түрде жазуға болады:

```
A = t.Aa; ,
```

мұнда t.Aa – treyg класының қасиеті.

Бұл мысалда t.Aa қасиеті арқылы біз treyg класының жабық a өрісін ашық өріске айналдырдық. Бұны орындаудың жеңіл жолы – класта private спецификаторының орнына public спецификатор жариялау.

Әлбетте қасиеттер кластың жабық өрістеріне қол жеткізуге арналған.

Қарастырылған мысалда класс өрісіне жаңа мәнді жазу қасиетіне біз мына шарттарды қоса аламыз: мәннің 0-ден үлкен болу және өріске тек бір рет жазуды орандау мүмкіндігі, мысалы:

```
set { if (a == 0 && value > 0) a = value;}
```

Қасиеттің кез келген әдістерінің бірі болмауы мүмкін (бірақ, екеуі емес). Ондай жағдайда бізде тек қана оқуға немесе тек қана жазуға арналған қасиеттер болады.

22.4 Сілтемесі параметрі - This

Арнайы нұсқағыш өрісті (this) қарастырайық, ол объект құрылған кезде автоматты түрде құрылады және осы объекттің адресін сақтайды.

Іс жүзінде объект өрістері мен класс әдістерінің байланысы this параметрі арқылы орнатылады. Кластың әрбір әдісі ағымдағы объект элементтерімен жұмыс жасауы үшін this параметрін тікелей қолдана алады. `This` мәні үнемі ағымдағы объектке (қосымшадағы ағымдағы объект) сәйкес болғандықтан, класс әдістері ағымдағы объект элементтерімен жұмыс жасайды.

Көптеген конструкторларда объект өрістерін инициализациялау үшін `this` параметрі қолданылады. Авторлар класс конструкторын сипаттаған кезде формалды параметрлердің атаулары ретінде класс өрістерінің атауларын қолданады. Сондықтан класс өрістері мен формалды параметрлердің атауларын ажырату үшін класс өрістері атауларының алдына this параметрі жазылады. Мысалы, treyg конструкторы мына түрде жазылады:

```
public treyg(int a, int b, int c)
 {
 this.a = a; this.b = b; this.c = c;
 }
```

Атауларға байланысты түсінбеушіліктерді жоюдың бір жолы – түрлі формалды параметрлерге басқа атауларды беру (бағдарламада осы тәсіл қолданылады).

Кластың статикалық элементтерін қолданған кезде this параметрін пайдалануға болмайды, өйткені олар нақты бір объектке емес жалпы класқа тиісті болады.

C# тілінде base сілтемесі бойынша тағы да бір параметр қоланылады, ол базалық объектпен жұмыс істеу үшін пайдаланылады. Егер Form1.Designer.cs файлының кодын мұқият қарасаңыз, void Dispose(bool disposing) әдісінде жазылған кодтың соңғы жолдарында base.Dispose(disposing); жазбасы бар. Бұл жазба бойынша жадыдан объект жойылған кезде, базалық объект те жойылады.

22.5 Класс оқиғалары

ОББ маңызды құрамдас бөлігі – класта жүзеге асырылған оқиғалар механизмі. Осы механизм арқылы бір объект (оқиға көзі) басқа объектке (оқиғаны қабылдаушы) өз жағдайының өзгергені жөнінде хабарлай алады.

Әдетте оқиға тетігі көпағынды процестердегі синхронизация кезінде қолданылады, бұл дегеніміз - осы ағындар жұмысының кезектілігін реттеу.

Бірақ осы механизмді Windows-қосымшаларда қолдануға болады, онда батырма, жалаушалар, т.б. элементтер пайдаланушының онымен байланысы туралы ақпаратты шығарады. Мысалы, барлық объекттерді – батырмаларды (Button класының) тышқанмен шерткен кезде олар OnClick оқиғасын туындатады.

Формада орналасқан әрбір басқару элементінің белгілі оқиғалар жиыны болады, оқиғалардың «бос» өңдеуіштерін элементтің қасиеттер терезесі арқылы ашуға болады. Алайда бағдарламашылар өздерінің арнайы оқиғалар өңдеуіштерін жаза алады. Оқиғалар механизмін жүзеге асыру үшін, яғни белгілі бір кластың клиенттеріне класс оқиғасының пайда болу жайлы хабарлау үшін оқиға көзі мыналарды орындауы керек:

– оқиғаны класс мүшесі ретінде жариялау (өрістермен, әдістермен, қасиеттермен бірге) – жариялау үшін Event қызметтік сөзі қолданылады;

– керекті мезетте класс клиенттеріне (оқиғаны қабылдаушы) орын алған оқиға туралы ақпаратты жеткізу, қажетті параметрлерді көрсету;

– клиенттен жауапты алу, оны талдау, оқиғаға байланысты әрекеттерді орындау.

Соңғы екі операция (клиентпен ақпарат алмасу) әдетте делегаттар арқылы жүзеге асырылады, оларды біз келесі бөлімдерде қарастырамыз.

22.6 Класс операцияларын қайта анықтау

Кейбір кластарда жазылатын код үзіндісінің (блоктар) тақырыбында класс атауынан (бірақ конструктор емес) кейін, бірақ дөңгелек жақшаларда жазылатын формалды параметрлерге дейін `operator` қызметтік сөзі қолданылады. `Operator` қызметтік сөзі арқылы операцияларды қайта анықтау механизмі жазылады, операцияларды қайта анықтау қарапайым математикалық өрнектерде класс типіндегі айнымалыларды қолдануға мүмкіндік береді. Мысалы, «студент» класы үшін қосу операциясы қайта анықталған болса (operator+) және бағдарламада ct1, ct2 – «студент» типіндегі екі объект құрылған болса, онда ct1 + ct2 өрнегін жазуға болады.

Екі объектіні қосу нені білдіреді? Мүмкін біз ондағы аттарды біріктірген немесе жас шамаларын қосқан болармыз?

Міне, операцияларды қайта анықтау осы мақсатта қолданылады, онда екі объектіні бір-біріне қосқанда нені қосу керектігі анық жазылады.

Operator қызметтік сөзі операцияға қайта анықтау орындалатынын, ал « + » операциясы болса, қосу операциясының қайта анықталатынын көрсетеді.

Operator сөзін пайдаланып кластың меншікті операцияларын анықтау операцияларды қайта анықтау деп аталады. Операцияларды қайта анықтау әдетте математикалық, физикалық ұғымдарды сипаттайтын кластар үшін қолданылады.

Кластың операциясын анықтау арнайы әдістердің (функциялар - операциялар) түрлері арқылы сипатталады.

Операцияны қайта жүктеудің жазылу пішімі мына түрде болады:

```
[спецификаторлар] класс атауы operator операция атауы (формалды параметрлер) {денесі }
```

Спецификатор ретінде әдетте public және static қызметтік сөздері бірге қолданылады. Сонымен қатар, операцияны сыртқы операция түрінде жариялауға болады (extern).

Операцияны өрнектерде қолданғанда орындалатын әрекеттер операция денесінде (басқа әдістердің денесі сияқты блок) анықталады.

Мысалы:

```
public static int operator+ (Stydent S1, Stydent S2)
  { return S1.Ocenka + S2.Ocenka; }
```

Егер біз белгілі бір класс үшін операцияны қайта анықтауды енгізетін болсақ, онда оның әрекеттері тек осы кластың деректеріне ғана қолданылады.

Ескерту, операцияны қайта анықтау жаңа операцияны құрмайды, тек операцияларды класс типіндегі деректерге бейімдейді.

Операцияны қайта анықтауды сипаттаған кезде келесі ережелерді сақтау керек:

– операция кластың ашық статикалық әдісі түрінде сипатталуы керек (public static спецификаторлары);

– формалды параметрлерді операцияға мәндері бойынша жіберілуі тиіс (яғни ref немесе out қызметтік сөздері қолданылмайды);

– кластың барлық операцияларының жазылу форматтары әр түрлі болуы керек.

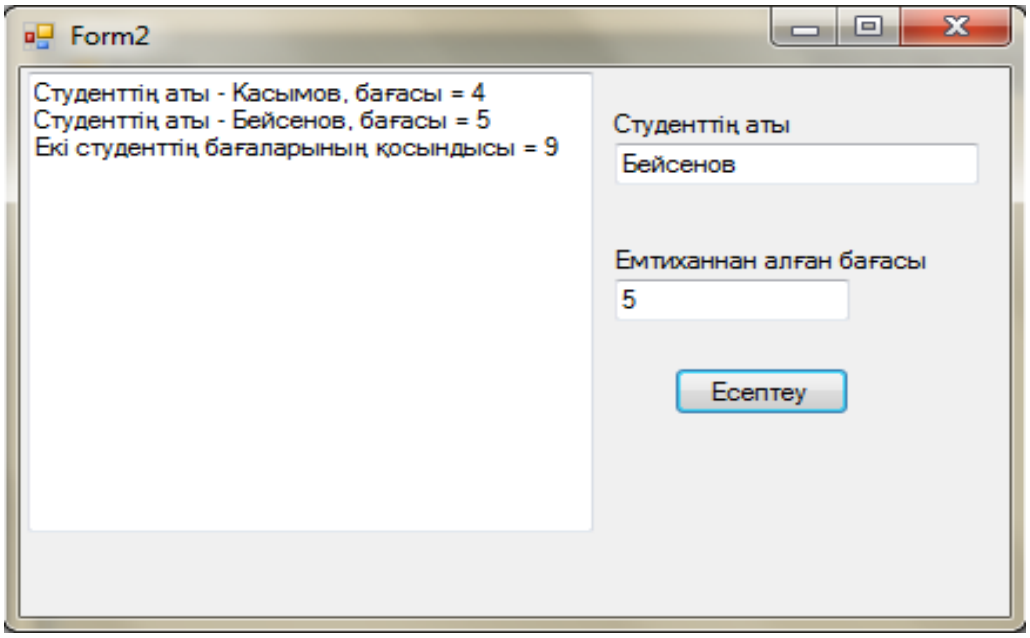
Әдетте операцияны қайта анықтау пайдаланушы анықтайтын типтерге арналған өрнектер синтаксисінің көрнекілігін қамтамасыз ету үшін қолданылады.

Бірінші тұрған екі студенттің бағалар қосындысын есептеу үшін операцияны қайта анықтауды қолдану мысалы:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
```

```
      public partial class Form2 : Form
      {
      public class Stydent
      { int Ocenka;
       string Name;
       public string ss;
       public int Aa
       {
       get { return Ocenka; }
       set { if (value >= 2 && value <= 5) Ocenka = value; else
ss = "Бағаның мәнін дұрыс енгізіңіз \r\n"; }
       }
       public void Vvod(string name)
       {
       Name = name;
       }
      public static int operator +(Stydent S1, Stydent S2)
       {
       return S1.Ocenka + S2.Ocenka;
       }
      }
      public Stydent[] styd = new Stydent[5];
      public static int n = 0;
      public Form2()
      {
       InitializeComponent();
       textBox1.Text = "";
       n = 0;
      }
      private void button1_Click(object sender, EventArgs e)
      { int Oc;
       string fio, In;
       Stydent st = new Stydent();
       fio = textBox3.Text;
       st.Vvod(fio);
       In = textBox2.Text;
       Oc = Convert.ToInt32(In);
       st.Aa = Oc;
       styd[n] = st;
       textBox1.AppendText("Студенттің аты - " + fio + ",
бағасы = " + styd[n].Aa.ToString() + "\r\n");
       if (st.Aa != 0) n++;
       if (n == 2) textBox1.AppendText("Екі студенттің
бағаларының қосындысы = " + (styd[0] + styd[1]).ToString() +
"\r\n");
      }
      }
     }
```

Бағдарлама жұмысы 22.3-суретте көрсетілген.



22.3-сурет – Қосу операциясын қайта анықтау мысалы

22.7 Өзін-өзі тексеру сұрақтары

1 Параметрсіз конструктор не үшін қолданылады?

2 Параметрлері берілген конструктор не үшін қолданылады?

3 Атаулары бірдей болатын әдістерді анықтау процесін қалай атайды?

4 Бір кластың бірнеше конструкторлары қалай аталады?

5 tka класының деструкторы қашан шақырылады?

6 Егер қайтарылатын мәнінің типі void болып жарияланса, әдіс қалай аталады?

7 C# тілінде типі void емес әдіс қалай аяқталуы тиіс?

8 Әдістің қандай формалды параметрлері сілтемелік параметрлер деп аталады?

9 Өрістер мен оларды өңдеу әдістерінің бір құрылымда бірігуі қалай аталады?.

10 Кластың жабық өрістеріне қол жеткізу үшін кластарда қандай механизм қолданылады.

23 ОББ ПРИНЦИПТЕРІ

23.1 Инкапсуляция ұғымы

Объекті-бағытталған бағдарламалау технологиясы үш негізгі принциптерге негізделеді – инкапсуляция, мұрагерлік және полиморфизм.

Инкапсуляция принципі, яғни деректер мен оларды өңдеу әдістерінің бір құрылымға бірігуі класты ұйымдастыру негізін құрайды.

Формалды түрде инкапсуляция дегеніміз – деректерге қосымшадан тікелей қол жеткізуден қорғау мақсатында кластың өрістері мен әдістерінің бірігуі.

Объект өрістері қасиеттер немесе қол жеткізу ережелерінің жиыны – интерфейс арқылы қолданылуы тиіс. Объект өрістерін жасыру – олардың инкапсуляциясы («капсула» сөзінен) қасиеттер арқылы жүзеге асырылады.

Қасиеттер ұғымы алдынғы бөлімде толық қарастырылған, сондықтан қасиетті қосымшада қолдану мысалын ғана қарастырамыз.

Класс қасиеттерінің жұмысын көрсету үшін мысал ретінде treyg класын қолданамыз.

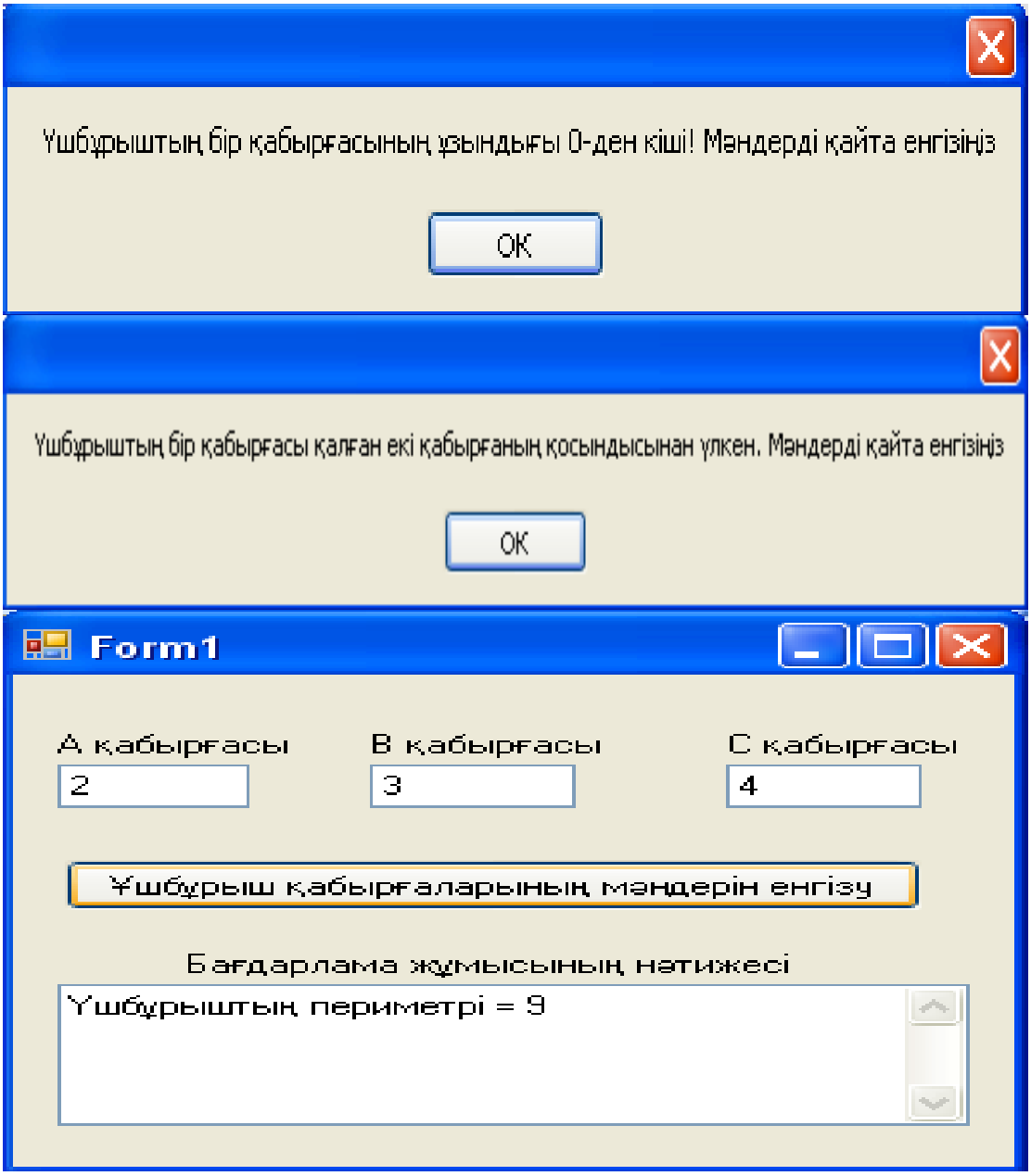
23.1-есеп. treyg класында үшбұрыш қабырғаларына қасиеттер арқылы кездейсоқ бүтін сандар меншіктеледі, сандар минус 5-тен 5-ке дейінгі аралықта болады. Қасиеттер мына шартты тексереді: енгізілген сандар 0-ден үлкен болуы керек. Үшбұрыш қабырғаларының мәндері енгізілгеннен кейін кластың арнайы әдісі мына шартты тексереді: кез келген екі қабырғаның қосындысы үшінші қабырғадан үлкен болуы керек. Қосымшаның жұмысына түсініктемелердің берілуі тиіс.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsApplication1
{
 public partial class Form1 : Form
 {
 public class treyg
 {
  private int a, b, c, p;
  public string ss;
  public treyg()
  {
  a = b = c = 0;
  }
  public int Aa
  {
  get { return a; }
  set { if (value > 0) a = value; else a = 0; }
  }
```

```
        public int Bb
        {
        get { return b; }
        set { if (value > 0) b = value; else b = 0; }
        }
        public int Cc
        {
        get { return c; }
        set { if (value > 0) c = value; else c = 0; }
        }
        public int Pp
        {
        get { return p; }
        }
        public void proverka()
        {
        if (a + b > c && a + c > b && b + c > a)
        {
         p = a + b + c;
         ss = ss + "Үшбұрыштың периметрі = " + p.ToString();
        }
        else
         MessageBox.Show("Үшбұрыштың бір қабырғасы қалған екі
қабырғаның қосындысынан үлкен. Мәндерді қайта енгізіңіз ");
        }
       }
       public Form1()
       {
        InitializeComponent();
       }
       private void button1_Click(object sender, EventArgs e)
       {
        Random rnd = new Random();
        int A = 1, B = 1, C = 1;
        bool ok = true;
        treyg t = new treyg();
        while (ok)
        {
        A = rnd.Next() % 11 - 5; t.Aa = A;
        B = rnd.Next() % 11 - 5; t.Bb = B;
        C = rnd.Next() % 11 - 5; t.Cc = C;
        textBox1.Text = Convert.ToString(t.Aa);
        textBox2.Text = Convert.ToString(t.Bb);
        textBox3.Text = Convert.ToString(t.Cc);
        if (A + B + C == t.Aa + t.Bb + t.Cc)
        {
         t.proverka();
         if (t.Pp != 0) ok = false;
        }
        else
         MessageBox.Show("Үшбұрыштың бір қабырғасының ұзындығы
0-ден кіші! Мәндерді қайта енгізіңіз ");
```

```
    }
    textBox1.Text = Convert.ToString(t.Aa);
    textBox2.Text = Convert.ToString(t.Bb);
    textBox3.Text = Convert.ToString(t.Cc);
    textBox4.Text = t.ss;
  }
  }
}
```

Бағдарлама жұмысы 23.1-суретте көрсетілген.



23.1-сурет – Қосымшаның жұмыс терезесі

MessageBox.Show() терезесі шыққан кезде қосымшаның негізгі терезесінде үшбұрыш қабырғаларын сипаттайтын қасиеттердің мәндері шығады.

23.2 Мұрагерлік ұғымы

Мұрагерлік принципі объекті-бағытталған бағдарламалау тұжырымдамасында іргелі принцип болып табылады. Мұрагерліктің мақсаты – құрылып қойған кластарды қайталап қолдану.

Кейбір авторлар кластардың мұрагерлік идеясын түсіндіргенде жалпыдан жекеге қарай жалғасатын иерархиялық байланыстың мысалын келтіреді:

Жануарлар – мысық тектестер – жолбарыс.

Басқа авторлар кластардың мұрагерлік идеясын кіші объекттен үлкен объектке қарай жалғасатын иерархиялық байланыстың мысалы арқылы түсіндіреді:

Нүкте – кесінді – тіктөртбұрыш.

Материалды түсіндіруде авторлардың осы екі көзқарастары қолданылады.

«Кіші объекттен үлкен объектке қарай» тәсілі мұрагерлік идеясын түсіндіруге мүмкіндік береді – нүкте, одан кейін кесінді, т.б. Осы технологияны мұраланатын кластардың тізбегін «нөлден» дайындаған кезде қолдануға болады.

«Жалпыдан жекеге қарай» немесе «жалпыдан нақтыға қарай» екінші тәсілі бағдарламалауда құрылып қойған кластарды пайдалану арқылы қолданылады, мысалы, DELPHI-де VCL, VISUAL C++ ортасында MFC және басқа да стандартты кітапханалар, C# тілінде атаулар кеңістігі.

Деректері немесе әдістері мұраланатын класты базалық класс деп атайды.

Деректерді немесе әдістерді базалық кластан мұраға алатын класты туынды класс деп атайды.

Мұрагерлік туынды класқа өз қасиеттерін, деректерін, әдістерімен қатар базалық кластың қасиеттерін, деректерін, әдістерін қолдануға мүмкіндігін туғызады.

Мұрагерлік идеясын түсіндіру үшін келесі мысалды қарастырайық. Қосымшада құрылып қойған класқа қарағанда қосымша жаңа қасиеттер мен деректерге ие белгілі бір класты жазу керек болсын.

Оны жүзеге асырудың екі жолы бар.

Бірінші әдіс бойынша бір қосымшадағы класты жаңа қосымшадағы ға көшіріп, оған керекті өзгерістерді жазу керек.

Екінші әдіс бойынша алдынғы класты мұраға алатын (алдыңғы кластан туындайтын) екінші класты құру керек.

Бағдарламаны жазу бойынша қарастырылған екі әдістің ішінен ОББ тұрғысынан екінші әдіс қолайлы болып келеді.

Кластарды мұраға алу бағдарлама кодын айтарлықтай қысқартады және қосымшаны құру уақытын азайтып, оның сенімділігін жоғарлатады.

Мұрагерлік иерархиялық құрылымды құруға мүмкіндік береді, иерархиялық құрылымда туынды кластар базалық кластың өрістерін, қасиетерін, әдістерін өз мүмкіндігіне алады және оларды толықтыра немесе өзгерте алады. Сонымен мұрагерлік кодты бірнеше рет қолдануға мүмкіндік береді. Базалық кластың кодын жазып, оны дұрыстағаннан кейін, осы класты мұралану арқылы оның кодын өзгертпей-ақ түрлі жағдайлар үшін қолдануға болады.

Иерархиялық құрылымның басына жақын орналасқан кластарда құрылымның төменгі жағындағы кластардың жалпы сипаттары біріккен. Иерархиялық құрылымның бойымен төмен жылжыған сайын кластардың айқын ерекшеліктері ұлғаяды.

C# тілінде кластардың иерархиялық тізбегінің базалық класы – System.Object.

Сонымен, мұрагерлік мына өзара байланысты мақсаттарда қолданылады:

– бағдарламада код үзінділерінің қайталануын болдырмау;

– бағдарламаның модификациясын жеңілдету;

– құрылған қосымшаны пайдаланып, жаңа қосымшаны құруды жеңілдету.

Сонымен қатар, мұрагерлікті пайдалану коды қол жетімді емес, бірақ өзгерістерді енгізуді қажет ететін объекттерді қолданудың бір ғана жолы болып келеді.

Кластарды мұралану бойынша жазу пішімінде класс әдеттегідей жарияланады, онда қос нүкте арқылы базалық кластың атауы жазылады:

```
    [ атрибуттар ] [ спецификаторлар ] class
класс_атауы [ : базалық класс ]
    { класс денесі}
```

Егер базалық кластың атауы көрсетілмесе, онда базалық класс болып System.Object класы есептеледі.

Егер бізде базалық класс бар болса, онда оны қос нүкте арқылы көрсету керек, мысалы:

```
public class otr : tka
{ класс денесі }
```

Мына мысалда кесінді класы нүкте класын мұраланады. Әлбетте, егер tka класының деректері privat қол жеткізу спецификаторы арқылы жабық болса, онда олар туынды класс үшін де жабық болады.

Кейбір базалық кластарда деректер protected қол жеткізу спецификаторынан кейін орналасады. Мысалы,

```
protected int x;
protected int y;
```

Қол жеткізу protected спецификаторының қажеттілігі мынада: туынды класс базалық кластың ашық элементтерін (public спецификаторы) қолдана алады.

Екінші жағынан туынды класс базалық кластың жабық элементтерін (private) тікелей қолдана алмайды, ол үшін туынды класс базалық кластың әдістерін қолдануы керек. Сондықтан кластың қорғалған элементтерін анықтайтын protected қол жеткізу спецификаторы пайда болды. Қорғалған элементтер жабық және ашық элементтердің ортасында аралық орынды алады. Егер элемент қорғалған болса, онда туынды класс объекттері оны ашық элемент сияқты қолдана алады. Қосымшанаң қалған бөлігіне қорғалған элементтер жабық болады. Ал егер қосымшада жабық элементтерді қолдану керек болса, онда оны тек кластың сәйкес әдістері арқылы ғана орындауға болады.

Класты мұралану кезіндегі негізгі мәселе – туынды кластың конструкторларын шақыру үшін объекттердің құрылу кезектілігін қарастыру.

Туынды класының әдістеріне базалық кластың деректері мен әдістерін мұралануға мүмкін болғандықтан, туынды класс объектісін дайындаған кезде базалық класта барлық деректері мен әдістері болуы тиіс.

Сондықтан туынды класс конструкторының жұмысы базалық класс конструкторын шақырудан басталады, оның жұмысы аяқталғаннан кейін туынды класс объектісі құрылады.

Туынды класс конструкторы мұраланатын базалық класс деректерінің элементтерін инициализациялауы тиіс.

Туынды класс деструкторын шақырған кезде бірінші болып туынды класс объектісі жойылуы керек, ал одан кейін – базалық класс объектісі.

Мұралану мәселесін қарастырған кезде мынаны ескеру керек: базалық кластың конструкторы мен деструкторы туынды класқа мұраға берілмейді.

23.2-есеп. Мұраланатын кластар тізбегін құру керек: нүкте-кесінді. Класс объекттерінің құрылу кезектілігін көрсететін мүмкіндікті қарастыру керек.

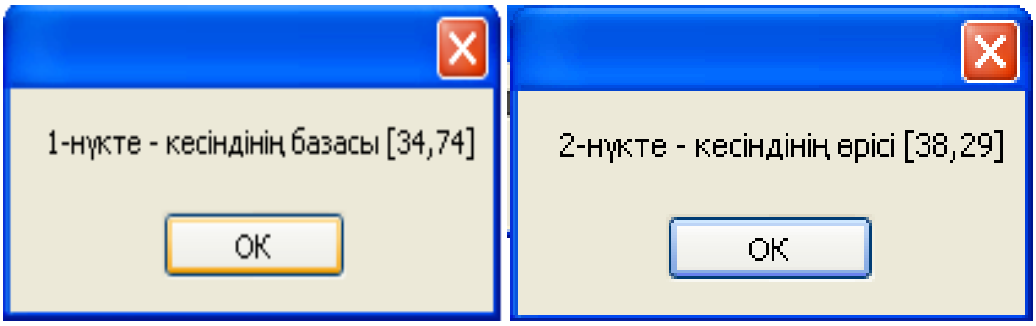
Form1.cs файлының коды:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsApplication1
{
```

```
      public partial class Form1 : Form
      {
      public int ax, ay, bx, by;
      public class tka
      {
       Random rnd = new Random();
       protected int x, y;
       public string ss;
       public tka()
       {
       x = rnd.Next(100);
       y = rnd.Next(100);
       }
       public void gettka(out int ax, out int ay)
       {
       ax = x; ay = y;
       }
       public string printtka()
       {
       return "[" + x.ToString() + "," + y.ToString() + "]";
       }
      }
      public class otr : tka
      {
       public tka b;
       public string s;
       public otr()
       {
       MessageBox.Show("1-нүкте - кесіндінің базасы [" +
x.ToString() + "," + y.ToString() + "]");
       s = "";
       }
       public void getotr(out int ax, out int ay)
       {
       int xx, xy;
       b.gettka(out xx, out xy);
       MessageBox.Show("2-нүкте - кесіндінің өрісі [" +
xx.ToString() + "," + xy.ToString() + "]");
       ax = xx; ay = xy;
       }
       public double dlina()
       {
       double dl;
       int k1, k2, k3, k4;
       k1 = x; k2 = y;
       b.gettka(out k3, out k4);
       dl = Math.Sqrt((k1 - k3) * (k1 - k3) + (k2 - k4) * (k2 -
k4));
       return dl;
       }
       public void printotr()
       {
```

```
        s = "Кесіндідегі нүкте координаты = " + b.printtka();
        s = s + "Кесіндінің базасындағы нүкте координаты = [" +
x.ToString() + "," + y.ToString() + "]";
        s = s + " Кесіндінің ұзындығы = " +
dlina().ToString("###.###");
        }
       }
       public Form1()
       {
        InitializeComponent();
       }
       private void button1_Click(object sender, EventArgs e)
       {
        string str;
        otr c = new otr();
        c.gettka(out bx, out by);
        tka bb = new tka();
        c.b = bb;
        c.getotr(out ax, out ay);
        c.printotr();
        str = c.s;
        textBox1.Text = str;
        Invalidate();
       }
       private void Form1_Paint(object sender, PaintEventArgs e)
       {
        Pen myPen = new Pen(Color.Red, 2);
        Graphics g = e.Graphics;
        g.DrawLine(myPen, ax, ay, bx, by);
       }
       }
      }
```

Қосымшаның жұмысы 23.2-ші және 23.3-суреттерінде көрсетілген.



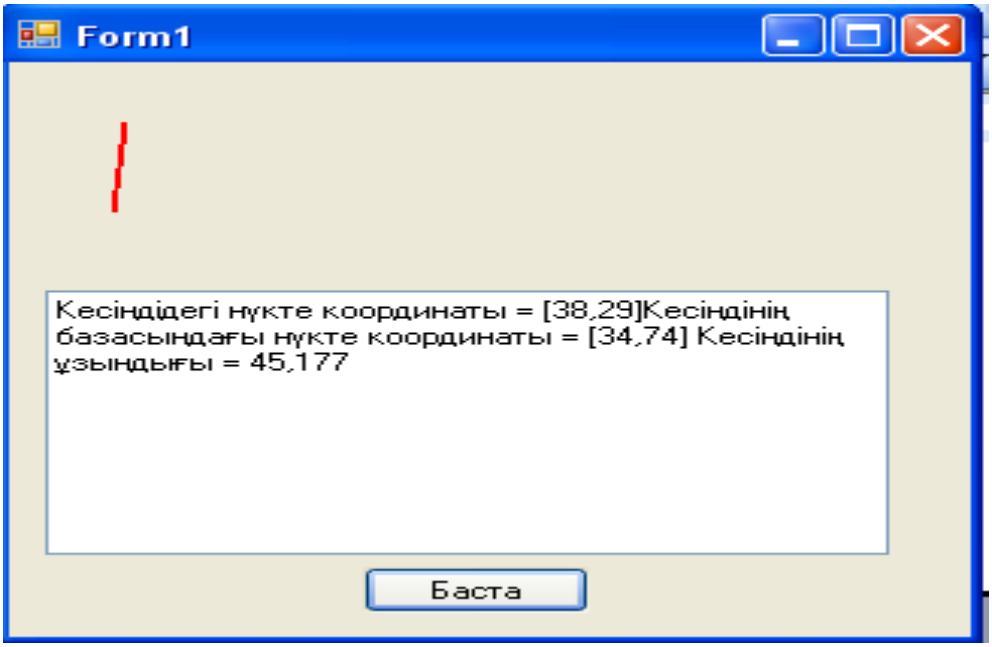
23.2-сурет – Кесінді ұштары координаталарының диалогтық терезелері

Кесінді объектісін құру кезінде базалық класс – нүкте объектісі бірінші болып құрылады (1-диалогтық терезе арқылы тексеру).

Одан кейін нүкте объектісін құраймыз және оны кесінді объектісінің өрісіне меншіктейміз (2-диалогтық терезе арқылы тексеру).

Кесінді ұзындығые есептейміз. Барлық нәтижелерді көпжолды мәтін редакторына шығарамыз.

X,Y координаттар жүйесіне кесіндіні саламыз (координаттар жүйесінің басы – форманың сол жақ жоғарғы бұрышы). Form1_Paint оқиға өңдеуішін құраймыз: форма үшін Properties терезесінде Paint оқиғасын таңдаймыз. Геометриялық фигуралар мен сызықтар үшін объекттерді дайындаймыз және кесінді координаталары бойынша сызықты саламыз.



23.3-сурет – Қосымшаның жұмыс терезесі

Оқиға өңдеуішін іске қосу (форма терезесін қайта салу) Invalidate() әдісі арқылы орындалады.

23.3 Өзін-өзі тексеру сұрақтары

1 Бір объект басқа объектке өз жағдайының өзгергені туралы қалай хабарлай алады?

2 Өз жағдайының өзгергені туралы басқа объектке хабарлайттын объект қалай аталады?

3 Кейбір оқиғалардың «бос» өңдеуіші қалай құрылады?

4 Формада button1 батырмасын екі рет шерткенде қандай оқиға өңдеуіші құрылады?

5 Кластарда операциялардың қайта анықтау не үшін қолданылады?

6 Мұрагерлік ұғымы?

7 Кластарды мұраланудың негізгі мақсаты неде?

8 Қасиеттерін, деректерін, әдістерін басқа класс мұраланған болса, онда ондай класс қалай аталады?

9 Базалық класс қасиеттерін, деректерін, әдістерін мұраға алатын класты қалай атайды?

10 ОББ неге негізделген?

24 ПОЛИМОРФИЗМ ПРИНЦИПІ

24.1 Полиморфизм ұғымы

Алдынғы бөлімде қарастырылған мұрагерлік ұғымы базалық кластың деректері мен әдістерін қолдануға мүмкіндік береді, бірақ сонымен қатар мұрагерлік екі түрге бөлінеді – статикалық және динамикалық мұрагерлік (әдістерді статикалық және динамикалық байланыстыру).

Статикалық мұрагерлік дегеніміз - барлық байланыстары қосымшаны компиляциялау барысында құрылатын және өзі кластарды сипаттау құрылымдарында жазылатын мұрагерлік.

Динамикалық мұрагерлік және онымен байланысты полиморфизм қасиеті бойынша кейбір байланыстар бағдарламаны орындау процесінде қалылтасады.

Полиморфизм дегеніміз – мұраланатын кластар тізбегіндегі аттас әдістердің сан алуан жолмен орындалу түрлері.

Полиморфизм қасиетін жүзеге асыру арнайы виртуальды әдістер және абстрактлы базалық кластар арқылы орындалады.

Абстрактлы базалық кластар ұғымын қарастырайық.

Кластардты мұралану нәтижесінде артықшылықтарға ие болу үшін ОББ әзірлеушілері базалық кластарды құрай бастады. Базалық кластарда белгілі бір объекттер жиынының деректерін өңдейтін барлық мүмкін әдістері болады, бірақ базалық кластарда әдетте, деректер элементтері болмайды.

Мысалы, «геометриялық фигуралар» базалық класын құрғанда оған аудан мен көлемді табу әдістерін қосуға болады. Әлбетте, егер «нүкте» класы немесе «кесінді» класы туынды кластар болса, онда осы объекттер үшін жоғарыда келтірілген әдістердің еш маңызы болмайды.

Базалық кластарда объекттерді құруға болмайтын болса немесе оның маңызы жойылса, онда ондай кластар абстрактлы базалық кластар деп аталады. Абстрактылы базалық кластар тек қана ұрпақты құру үшін ғана қолданылады. Әдетте, оларда тек қана әдістер жиынтығы жазылады және осы әдістерді оның әрбір ұрпағы өзінше жүзеге асыратын болады. Абстрактылы базалық кластардың осындай әдістері қолданыста жоқ немесе виртуалды деректер элементтеріне арналған (яғни, мұрагерлік тізбектегі келешекте құрылатын кластардың деректер элементтеріне арналған).

Мұрагерлік тізбек бойынша келешекте құрылатын кластардың қолданыста жоқ, деректердің виртуальды элементтеріне арналған әдістері виртуальды әдістер деп атала бастады.

Виртуальды әдістерді белгілеу үшін C# тілінде virtual арнайы термині қолданылады. Мысалы:

```
virtual public double ploc() {return 0.0;}
```

virtual сөзі ағылшын тілінен аударғанда «нақты» («фактический») дегенді білдіреді. Әдісті виртуальды деп жариялау мынаны анықтайды: осы әдіске бағытталған барлық сілтемелерге рұқсат әдісті шақыру фактісі бойынша беріледі, яғни компиляция кезеңінде емес, қосымшаның орындалу барысында. Бұл механизм әдістерді динамикалық немесе кейінге қалтырып байланыстыру деп аталады.

Компилятор виртуалды әдістерді кездестіргеннен кейін виртуалды әдістер кестесін (Virtual Method Table, VMT) құрайды, онда виртуальды әдістердің атауы және кіру нүктелерінің адрестері жазылады. Әрбір класс үшін бір виртуальды әдістер кестесі құрылады.

Сонымен қатар, қосымшаның орындалуы барысында әрбір құрылған объектке нұсқауыш қосылады, ол нұсқауыш әзірленген VMT кестесіне бағытталады.

Виртуальды әдісті шықыру былай орындалады: объекттен VMT кестесінің адресі алынады, VMT-нен әдістің адресі алынады, ал одан кейін басқару осы әдіске көшеді. Сонымен, барлық аттас әдістердің ішінен виртуальды әдістерді қолданған кезде объект шақырған нақтылы типке сәйкес виртуальды әдіс таңдап алынады.

Егер туынды класс аттас виртуальды әдісті өз нұсқасында орындаған болса, онда класта осы әдіс override атрибутымен бірге жазылады және орнын басу немесе қалқалау (жабу) әдісі болып жарияланады. Мысалы,

```
override public double ploc() { . . . }
```

Әрбір туында класта виртуальды әдісті қайта анықтау міндетті емес. Егер әдіс туынды класта керекті әрекеттерді орындайтын болса, онда әдіс жай ғана мұраланады.

Қосымшаның орындалуы барысында кез келген базалық класс орнын басу әдісін шақырған кезде виртуальды әдістер кестесінде туынды кластың

сәйкес әдісіне сілтеме жазылады және бұл әдіс аталық кластың әдепкі бөлігіндей қызмет атқарады.

Орнын басу виртуальды әдісінде базалық кластың аттас әдісінде сияқты параметрлер жиынтығы болуы керек.

Полиморфизм принципі абстрактылы базалық кластың виртуальды әдістерін орнын басу әдістері арқылы «қалқалауға» («перекрытии») негізделеді. Сонымен қатар әрбір туынды кластың мұраланған виртуальды немесе орын басу әдістері өзгеше жолмен жүзеге асырылады.

Сонымен бірге полиморфизм қасиеті дегеніміз – әр түрлі мұрагерлік кластардың объекттері үшін базалық кластың аттас виртуальды функциясын қолданған кезде түрлі тәсілмен орындау мүмкіндігі.

Полиморфизм сөзі грек тілінен аударғанда «түрлі тәсілдер» деген мығынаны білдіреді, ал қарастырылып отырған жағдайда - «бір шақыруға — бірнеше әдістер» деген мағынаны қолдануға болады.

Егер әдіс туында кластарда өзгеше орындалуы тиіс болса, онда базалық кластарды сипаттаған кезде осы әдістерді виртуальды әдістер ретінде анықтауға кеңес беріледі. Егер иерархияның барлық кластарында әдіс бірдей орындалатын болса, онда ол әдісті қарапайым әдіс сияқты анықтау керек.

24.2 Әдістерді статикалық мұралану мысалы

Қарапайым геометриялық пішіндердің мұрагерлік кластар тізбегін құрамыз. Базалық класс ретінде өрісі жоқ, ауданды есептеу виртуальды әдісі және басып шығару «таза виртуальды әдісі» бар класты қарастырайық.

Статикалық мұрагерлік мысалы – объекттердің қарапайым құрылуы, өрістерге 0-ден 100-ге дейінгі аралықтағы кездейсоқ мәндерді меншіктеу және оларды экранға басып шығару.

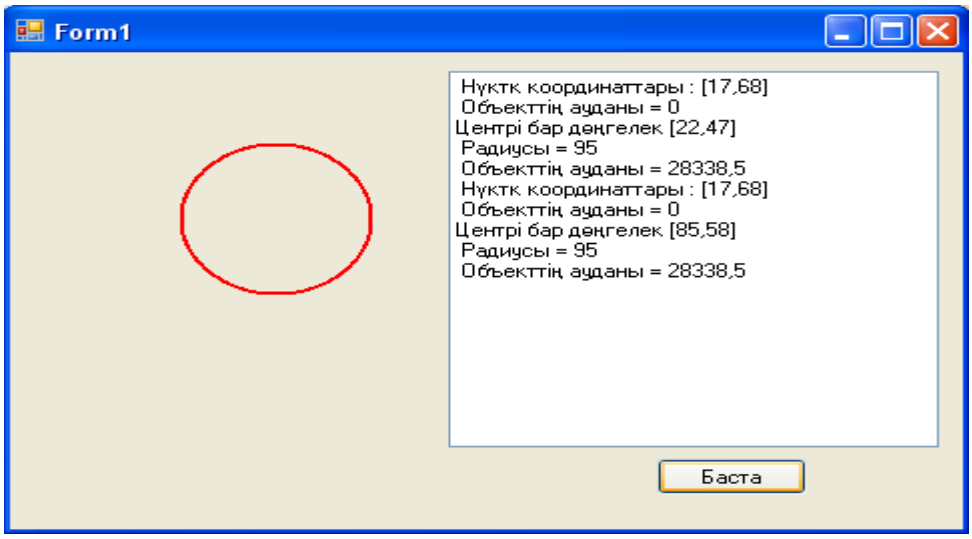
Form1.cs файлының коды:

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsApplication1
{
 public partial class Form1 : Form
 {
 public int x, y, ra;
 public abstract class baseGeo
 {
  public virtual double ploc() { return 0; }
  public abstract string printO();
 }
```

```
public class tka : baseGeo
{
 protected int x, y;
 public string ss;
 public tka()
 {
 Random rnd = new Random();
 x = rnd.Next(100);
 y = rnd.Next(100);
 }
 public int gettkax() { return x; }
 public int gettkay() { return y; }
 public void settka(int xx, int yy)
 {
 x = xx;
 y = yy;
 }
 override public string printO()
 {
 return "[" + x.ToString() + "," + y.ToString() + "]";
 }
}
public class kryg : tka
{
 public kryg()
 {
 Random rnd = new Random(10);
 r = rnd.Next(100);
 }
 public void setkryg(int ax, int ay, int rr)
 {
 settka(ax, ay);
 r = rr;
 }
 public void getkryg(out int ax, out int ay, out int rr)
 {
 ax = x;
 ay = y;
 rr = r;
 }
 public int getkrygr() { return r; }
 override public double ploc()
 // базалық кластың әдісін жабамыз
 {
 return 3.14 * r * r;
 }
 override public string printO()
 // tka класының әдісін жабамыз
 {
 string ss = "";
 ss = "Центрі бар дөңгелек " + base.printO() + "\r\n";
 // tka класының басып шығару функциясы мұраланады
```

```
        ss = ss + " Радиусы = " + r.ToString() + "\r\n";
        return ss;
        }
        private int r;
       }
       public Form1()
       {
        InitializeComponent();
       }
       private void button1_Click(object sender, EventArgs e)
       {
        int i;
        double pl;
        string s;
        Random rnd = new Random(20);
        x = rnd.Next(100);
        y = rnd.Next(100);
        //ra = rnd.Next(100);
        // әдістерді статикалық мұралану мысалы
        tka a = new tka();
        a.settka(x, y);
        s = "Нүкте координаттары : " + a.printO() + "\r\n";
        textBox1.AppendText(s);
        s = " Объекттің ауданы = " + a.ploc().ToString() +
"\r\n";
        textBox1.AppendText(s);
        kryg c = new kryg();
        x = c.gettkax();
        y = c.gettkay();
        ra = c.getkrygr();
        //c.setkryg(x, y, ra);
        textBox1.AppendText(c.printO());
        s = " Объекттің ауданы = " + c.ploc().ToString() +
"\r\n";
        textBox1.AppendText(s);
        Invalidate();
       }
       private void Form1_Paint_1(object sender, PaintEventArgs
e)
       {
        Pen myPen = new Pen(Color.Red, 2);
        Graphics g = e.Graphics;
        g.DrawEllipse(myPen, x, y, ra, ra);
       }
       }
      }
```

Бағдарлама жұмысы 24.1-суретінде көрсетілген.



24.1-сурет – Әдістердің статикалық мұралануы

Әдістердің статикалық мұралануы бойынша келтірілген мысалда әдістердің қалқалау механизмі қолданылады, бірақ әдістер арасындағы байланыстар қосымшаның компиляциясы кезінде нақтылы анықталады.

24.3 Әдістердің динамикалық мұралануының мысалы

Әдістердің динамикалық мұралануының мысалы ретінде алдында қарастырылған мұрагерлік кластар тізбегін қолданамыз, бірақ қосымшаның жұмысы кезінде 5 құрылған объектті стекке орналастырамыз.

Form1.cs файлының коды:

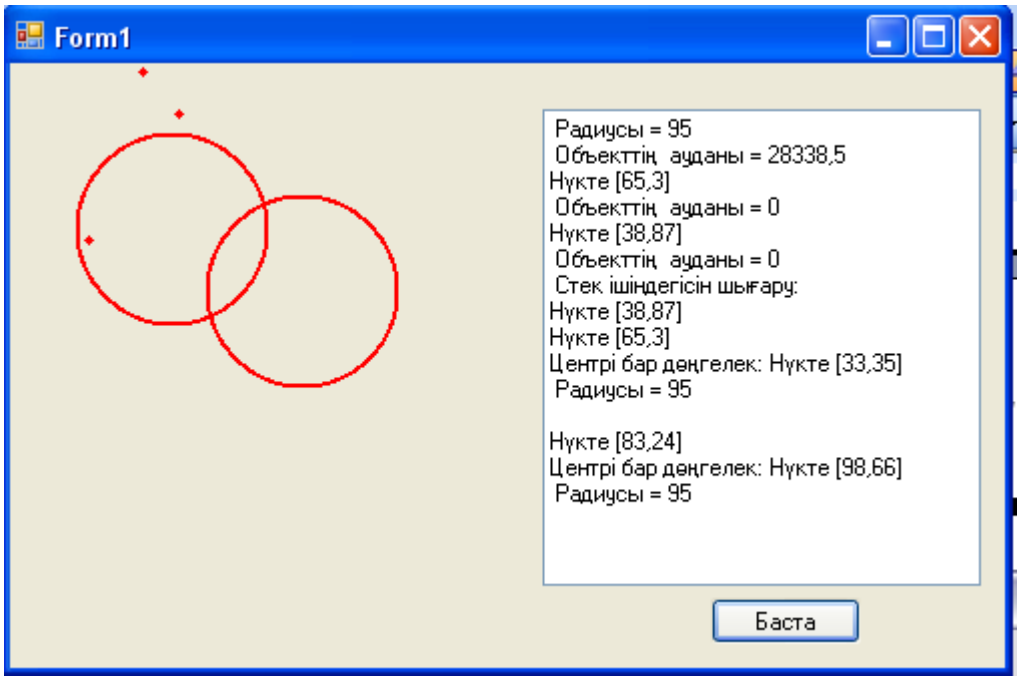
```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsApplication1
{
 public partial class Form1 : Form
 {
 public int x, y, ra;
 public int[,] masi = new int[6,3];
 public Stack vstek = new Stack();
 public abstract class baseGeo
 {
 public virtual double ploc() { return 0; }
 public abstract string print0();
 }
 public class tka : baseGeo
```

```
        {
        protected int x, y;
        public tka()
        {
        Random rnd = new Random();
        x = (int)Math.Round(rnd.NextDouble() * 100);
        y = (int)Math.Round(rnd.NextDouble() * 100);
        }
        public int gettkax() { return x; }
        public int gettkay() { return y; }
        public void settka(int xx, int yy)
        {
        x = xx;
        y = yy;
        }
        override public string printO()
        {
        return "Нүкте [" + x.ToString() + "," + y.ToString() +
"]";
        }
        }
        public class kryg : tka
        {
        public kryg()
        {
        Random rnd = new Random(10);
        r = (int)Math.Round(rnd.NextDouble() * 100);
        }
        public void setkryg(int ax, int ay, int rr)
        {
        settka(ax, ay);
        r = rr;
        }
        public void getkryg(out int ax, out int ay, out int rr)
        {
        ax = x;
        ay = y;
        rr = r;
        }
        public int getkrygr() { return r; }
        override public double ploc()
         // базалық кластың әдісін жабамыз
        {
        return 3.14 * r * r;
        }
        override public string printO()
         // tka класының әдісін жабамыз
        {
        string ss = "";
        ss = "Центрі бар дөңгелек: " + base.printO() + "\r\n";
        // tka класының басып шығару функциясы мұраланады
```

```
      ss = ss + " Радиусы = " + r.ToString() + "\r\n";
      return ss;
      }
      private int r;
      }
      static void vkl(Stack vst, baseGeo n)
      {
      vst.Push(n);
      }
      public Form1()
      {
      InitializeComponent();
      }
      private void button1_Click(object sender, EventArgs e)
      {
      int i,k;
      double pl;
      string s;
      Stack vstek = new Stack();
      Random rnd = new Random(20);
      // әдістерді  динамикалық мұралану мысалы
      for (i = 1; i <= 5; i++)
      {
      x = (int)Math.Round(rnd.NextDouble() * 100);
      y = (int)Math.Round(rnd.NextDouble() * 100);
      k = (int)Math.Round(rnd.NextDouble() * 100);
      if (k % 2 == 0) k = 0; else k = 1;
      if (k == 0)
      {
      tka a = new tka();
      a.settka(x, y);
      s = a.printO() + "\r\n";
      textBox1.AppendText(s);
      s = " Объекттің ауданы = " + a.ploc().ToString() +
"\r\n";
      textBox1.AppendText(s);
      masi[i, 0] = a.gettkax();
      masi[i, 1] = a.gettkay();
      masi[i, 2] = 0;
      vkl(vstek, a);
      }
      else
      {
      kryg c = new kryg();
      x = c.gettkax();
      y = c.gettkay();
      ra = c.getkrygr();
      textBox1.AppendText(c.printO());
      s = " Объекттің ауданы = " + c.ploc().ToString() +
"\r\n";
      textBox1.AppendText(s);
      c.getkryg(out masi[i,0], out masi[i,1], out masi[i,2]);
```

```
vkl(vstek, c);
}
}
s = " Стек ішіндегісін шығару: \r\n";
foreach (baseGeo T in vstek)
s = s + T.printO() + "\r\n";
textBox1.AppendText(s);
Invalidate();
}
private void Form1_Paint(object sender, PaintEventArgs e)
{
Pen myPen = new Pen(Color.Red, 2);
Graphics g = e.Graphics;
for (int i = 1; i <= 5; i++)
{
if (masi[i,2]==0)
{
x = masi[i, 0]; y = masi[i, 1];
g.DrawEllipse(myPen, x, y, 2, 2);
}
else
{
x = masi[i, 0]; y = masi[i, 1]; ra = masi[i, 2];
g.DrawEllipse(myPen, x, y, ra, ra);
}
}
}
}
}
```

Бағдарлама жұмысы 24.2 суретінде көрсетілген.



24.2-сурет – Әдістердің динамикалық мұралануының мысалы

Стек элементтерінің ішіндегісін экранға шығару полиморфизм мысалы болады (әдістердің динамикалық мұралануының). «Статикалық» тұрғыдан стек элементін экранға шығару қарапайым геометриялық пішіндердің элементін экранға шығару болып есептеледі. Бірақ қосымшаның жұмыс істеу процесінде стекке нүкте мен шеңбер кластарының объекттері орналастырылады (әдістердің динамикалық мұралануының мәнісі осында).

24.4 Өзін-өзі тексеру сұрақтары

1 C# тілінде барлық кластардың иерархиялық тізбегінде қандай класс базалық класс болып табылады?

2 Туында класс конструкторын шақырған кезде базалық немесе туында кластардың қандай объектісі ерте (барлығынан бұрын) құрылады?

3 Мұрагерлік кластар тізбегінің артықшылықтары неде?

4 C# тілінде статикалық мұрагерлік нені білдіреді?

5 C# тілінде динамикалық мұрагерлік нені білдіреді?

6 C# тіліндегі полиморфизм ұғымы?

7 Полиморфизм механизмі қалай жүзеге асырылады?

8 Бағдарламаның жұмысы барасында объект анықталған болса, онда оның әдісі қалай аталады?

9 C# тілінде абстрактылы базалық класс деп қандай класс аталады?

10 Виртуальдық әдістер кестесі дегеніміз не?

25 ИНТЕРФЕЙСТЕРДІ ПАЙДАЛАНУ

25.1 Интерфейс ұғымы

"Интерфейс" сөзінің мағынасы көп, түрлі жағдайларда әр түрлі мағынада болуы мүмкін. Бағдарламалық және аппараттық интерфейс ұғымдары бар, бірақ көп жағдайларда интерфейс сөзі объекттер немесе процестер арасындағы байланысты анықтайды. Осы бөлімде интерфейс ұғымы қарастырылатын болады (интерфейс interface сөзінен кейін жазылады). Интерфейс – кластың дербес түрі.

Интерфейс ұғымы толығымен абстрактылы класты сипаттайды, ондағы әдістер толығымен абстрактылы болады.

Интерфейстің абстрактылы кластан айырмашылығы синтаксисіндегі және орындалу тәртібіндегі өзгешеліктерінде болады.

Синтаксистік айырмашылығы мынада: интерфейс әдістері қол жеткізу модификаторынсыз жарияланады.

Орындалу тәртібіндегі өзгешелігі – ұрпақтарға қойылатын талаптардың қатаң болуында. Интерфейсті мұраланатын класс (интерфейсті класс) интерфейстің барлық әдістерін толық жүзеге асыруы тиіс. Жоғарыда сипатталған барлық өзгешеліктер интерфейстің абстрактылы класты мұраланатын кластан айырмашылықтарын көрсетіп тұр. Абстрактылы кластың ұрпағы (класы) аталық абстрактылы кластың кейбір әдістерін ғана орындай алады.

Интерфейстік кластың қарапайым кластан айырмашылығы – бірнеше түпкі интерфейстерді мұралана алуында. Сонымен, C# тілінде интерфейстік кластарда ғана бірнеше рет мұралануға рұқсат берілген.

Класс атауы мен қос нүктеден кейін түпкі интерфейстер тізім бойынша бір бірден көрсетіледі:

```
    public interface INewClass: IInt1, IInt2, ..., 
IIntN
    { . . .}
```

Осындай интерфейстік кластарда түпкі интерфейстердің барлық әдістерінің іске асырылуы болуы керек.

Ескерту, интерфейстік класс интерфейстермен қатар қарапайым бір класты (тек қана бір класты!) мұралануы мүмкін, қарапайым класты мұраланғанда ол әдеттегі мұрагер ретінде оның әдістерін өзгерте алады және өрістерді қоса алады т.б.

Бірнеше рет мұралануды қолданғанда атаулар арасындағы қателіктер немесе ортақ түпкі кластың болуына байланысты қателіктер кетуі мүмкін.

Атаулар арасындағы шиеленіс мына жағдайларда пайда болады: әр түрлі түпкі интерфейстерде синтаксистері бірдей аттас әдістер болуы мүмкін.

Интерфейстік класс түпкі интерфейстердің барлық әдістерін жүзеге асыруы керек болғандықтан коллизия пайда болады, оны шешудің бірнеше жолы бар.

Әдістерді желімдеу. Интерфейстік класс аттас әдістердің орындалуын бірдей деп қарастырады және түпкі кластардың барлық аттас әдістерін іске асыру үшін бір әдісті жариялайды.

Әдістерге басқа атау беру. Егер аттас әдістердің жүзеге асырылуы әр түрлі болуы керек болса, онда әдістердің атаулары өзгертіледі.

Интерфейсті абстрактылы кластан ерекшелендіретін тағы да бір маңызды қызметі бар. Абстрактылы класс класты жобалаудың алғашқы кезеңі болып табылады. Әрбір интерфейс класқа жаңа қосымша қасиеттерді қосады.

25.2 Интерфейс синтаксисі

Міндетті емес элементтері (олар квадрат жақшалармен бөлінген) бар интерфейстің жалпы сипаттамасының жазылу пішімі мына түрде болады:

```
[ атрибуттар ] [ спецификаторлар ] interface
класс_атауы [ : түпкі кластар]
  { класс_денесі } ,
```

мұнда,

атрибуттар – класс туралы қосымша ақпаратты анықтайды;

спецификаторлар – кластың құрамдас бөліктеріне қол жеткізу шарттарын анықтайды;

түпкі кластар – класс мұраланатын аталық түпкі кластар;

класс денесі – интерфейстік кластың құрамын анықтайды.

Егер интерфейстің жазылу пішіміне назар салып қарасақ, онда оның пішімі қарапайым кластың жазылу пішіміне ұқсас екенін байқауға болады. Өйткені интерфейс кластың бір түрі болып табылады.

.NET платформасының кітапханасында интерфейстердің көптеген түрлері бар, оларды класс мұраланған кезде қосымша қасиеттерге ие болады.

Мысалы, IComparable интерфейсін қосқан кезде біз объект деректерінінің өрістерін анықтаймыз (күйге келтіреміз), оларға объекттерді салыстыру әдістерін қосуға болады. Тек IComparable интерфейсін қосқаннан кейін ғана класс объекттерінде белгілі бір өріс бойынша сұрыптауды жүргізуге болады.

IEnumerable және IEnumerator интерфейстерінің іске қосылуы foreach конструкциясының көмегімен объект ішіндегісін қарап шығу мүмкіндігін, ал Icloneable интерфейсі объекттерді клондау мүмкіндігін береді.

Әрбір интерфейс класты белгілі бір жаңа мүмкіншіліктерді қосыды.

Мысалы, интерфейсті сауда үшін, яғни ағымдағы бағамға сәйкес валютаны сату-сатып алу үшін немесе жеңілдіктерді ескере отырып коммуналдық қызметтер бойынша түрлі есептеулер үшін құруға болады.

Мысал ретінде мына интерфейсті құрайық: интерфейс әдістерін жүзеге асырғанда кластарға музыкалық жазбаларды түрленіруге мүмкіндік береді – 7 нота мен дыбыс үзілісін 0 мен 7 аралығындағы цифрлерге түрлендіру және кері түрлендіруді орындау.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsApplication1
{
 public partial class Form1 : Form
 {
 interface ITextNoti
 {
  string Codirovanie();
  string Decodirovanie();
 }
 class MyzikText : ITextNoti
 {
  string text;
  static string[] codeTable =
 {
 "до", "ре", "ми", "фа", "соль", "ля", "си", "пауза"
 };
  //Конструктор
  public MyzikText(string txt)
  {
  text = txt;
  }
  //Интерфейстерді құру
  public string Codirovanie()
  {
  Boolean ok;
  string rez = "";
  string[] noti;
  //Төменгі регистрге түрлендіру
  string text1 = text.ToLower();
  // noti массивінің өлшемі Split әдісі қайтаратын
массивтің өлшеміне сай болады
  noti = text1.Split(' ');
  for (int i = 0; i < noti.Length; i++)
  {
   ok = false;
   for (int j = 0; j < 8; j++)
   if (noti[i] == codeTable[j])
   { ok = true; rez = rez + " " + j.ToString(); }
```

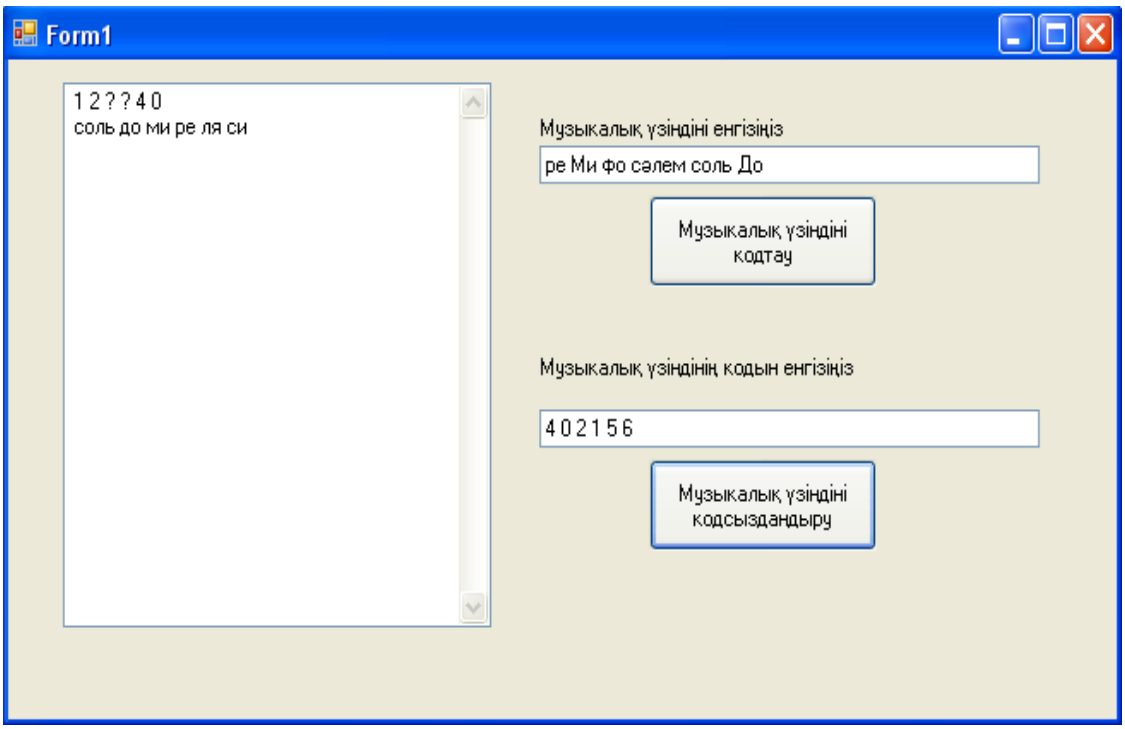
```
    if (ok == false) rez = rez + " ?";
   }
   return rez;
   }
   //нот кестесін қолданып text өрісінің шифрын анықтау
   public string Decodirovanie()
   {
   Boolean ok;
   string rez = "";
   string[] noti;
   // Төменгі регистрге түрлендіру
   string text1 = text.ToLower();
   noti = text1.Split(' ');
   for (int i = 0; i < noti.Length; i++)
   {
    ok = false;
    for (int j = 0; j < 8; j++)
    if (Convert.ToInt32(noti[i]) == j)
    { ok = true; rez = rez + " " + codeTable[j]; }
    if (ok == false) rez = rez + " ?";
   }
   return rez;
   }
  }
  public Form1()
  {
   InitializeComponent();
  }
  private void button1_Click(object sender, EventArgs e)
  {
   string a, b;
   a = textBox1.Text;
   MyzikText IcxodText = new MyzikText(a);
   b = IcxodText.Codirovanie();
   textBox3.AppendText(b + "\r\n");
  }
  private void button2_Click(object sender, EventArgs e)
  {
   string a, b;
   a = textBox2.Text;
   MyzikText IcxodText1 = new MyzikText(a);
   b = IcxodText1.Decodirovanie();
   textBox3.AppendText(b + "\r\n");
  }

  }
 }
```

ItextNoti интерфейсін жариялаймыз, оның екі әдісі бар: кодтау (нот мәтіні 0 мен 7 аралығындағы сандармен алмастырылады) және декодтау (0 мен 7 аралығындағы сандармен берілген мәтін нот мәтінімен алмастырылады).

```
interface ITextNoti
    {
        string Codirovanie();
        string Decodirovanie();
    }
```

Одан кейін интерфейстік класты жариялаймыз, ол интерфейсті мұраланады және оның әдістерін орындайды. Қол жетімді интерфейс әдістерін іске асырамыз. Қосымшаның кодында интерфейс әдістерінің орындалу алгоритміне түсіндірме берілген, сондықтан қосымша түсініктемені беруді қажет етпейді.

Бағдарлама жұмысы 25.1-суретте көрсетілген.



25.1-сурет – Интерфейстік класты қолдану

Қарастырылған мысалда интерфейс пен интерфейстік класты құру және қолдану технологиясы көрсетілген.

25.3 IEnumerable стандартты интерфейсін қолдану

Бір қарағанда интерфейстік класты енгізудің артықшылықтары жоқ болып көрінеді – кодтау, декодтау әдістерін тікелей MyzikText класына орналастыруға болады.

.NET платформасының кітапханасындағы кластарда әр түлі интерфейстердің интерфейстік әдістерінің көптеген саны бар, оларды класс мұраланып қосымша қасиеттерге ие болады (әдістердің орындалуы

арқылы емес, олардың атаулары арқылы). Әдетте әрбір интерфейстік класс интерфейстік әдістерді жүзеге асыруын өзі орындауы керек.

Мысалы, егер класта foreach циклінің көмегімен кейбір массив түрінде берілген тізімдеулі объекттерді қарап шығуды ұйымдастыру керек болса, онда біздің класс IEnumerable (тізімдеулі, перечислимый) интерфейсінің мұрагері болуы керек. Осы интерфейстің бір ғана GetEnumerator() әдісі болады және осы әдіс Enumerator (тізімдеуші, перечислитель) типіндегі объекті қайтарады. GetEnumerator() әдісінің жазылу пішімі келесі түрде болады:

```
IEnumerator GetEnumerator();
```

Сонымен, осы клас IEnumerable және IEnumerator интерфейстерінің мұрагері болуы керек.

IEnumerator интерфейсінің тізімделген кезекті объекті қайтаратын Object Current{get;} деген бір қасиеті және екі әдісі болады. Bool MoveNext() әдісі тізімдеушіні келесі тізімделген объектке жылжытады, void Reset() әдісі тізімдеушіні бірінші тізімделген объектке орнатады.

Жоғарыда келтірілген қасиет пен екі әдіс Foreach циклінің көмегімен массив объекттерін қарап шығу процессін ұйымдастыруға мүмкіндік береді. Осы интерфейстердің әдістері виртуалды коллекциямен жұмыс жасайды және осы жағдай олардың әмбебаптығын анықтайды.

Егер класта объекттерді салыстыру керек болса, мысалы, объекттерді сұрыптау керек болса, онда ондай класты IComparable интерфейсінің мұрагері етіп жариялау керек. Осы интерфейстің бір ғана әдісі болады - CompareTo(object obj), ол әдіс "үлкен", "кіші" немесе "тең" қатынастарының орындалуына байланысты оң, теріс немесе нөлге тең бүтін санды қайтарады.

IEnumerable және IEnumerator интерфейстерімен жұмыс жасауды қарастырайық. Осы мысал бойынша дүкендегі тауарларды қарап шығуды ұйымдастыру керек. Tovar класының екі өрісі болады – тауардың атауы және оның бағасы.

```
class Tovar
 {
  public string Naz; // Тауардың атауы мен бағасы
  public int Cena;
  public Tovar(string n, int c) // Конструкор
  {
  Naz = n; Cena = c;
  }
 }
```

Tovar типіндегі объекттерді сақтау үшін Cklad класын қолданамыз, құрылымы мынандай:

```
class Cklad
{
 public Tovar[] tovar; // тауарлар массиві
 public Cklad() // қойма конструкторы
```

```
 {
 tovar = new Tovar[4];
 }
}
```

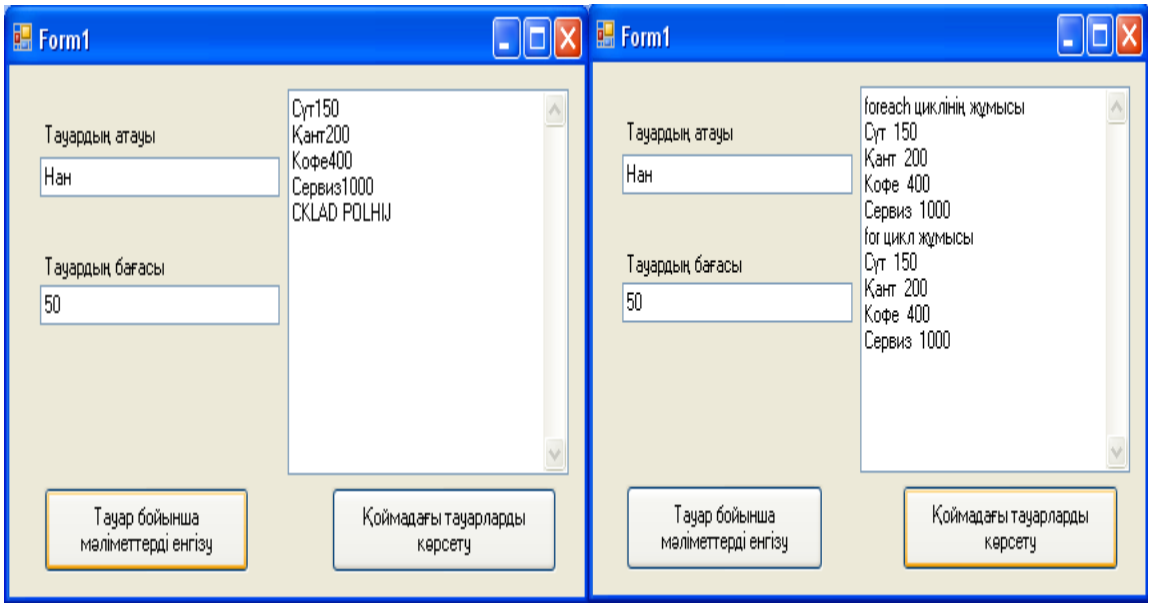
Қоймада сақтауға болатын объекттердің максималь саны 4-ке тең. Қосымшада IEnumerable интерфейсін қолданбаймыз.

Form1.cs файлының коды:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsApplication1
{
 public partial class Form1 : Form
 {
 public static string s;
 public static int kol;
 class Tovar
 {
  public string Naz; // Тауардың атауы мен бағасы
  public int Cena;
  public Tovar(string n, int c) // Конструкор
  {Naz = n; Cena = c;
  }
 }
 class Cklad
 {
  public Tovar[] tovar; // Тауар бойынша массив
  public Cklad() // қойма конструкторы
  {
  tovar = new Tovar[4];
  }
 }
 public Form1()
 { InitializeComponent();
  kol = 0;
  s = "";
 }
 Cklad ckl = new Cklad();
 private void button2_Click(object sender, EventArgs e)
 {
  if (kol < 4)
  {
  ckl.tovar[kol] = new Tovar(textBox1.Text,
Convert.ToInt32(textBox2.Text));
  s = s + textBox1.Text + textBox2.Text + "\r\n";
  }
  else { s = s + "Қойма толды" + "\r\n"; kol--; }
  kol++;
```

```
        textBox3.Text = s;
       }
       private void button1_Click(object sender, EventArgs e)
       {
        s = "";
        s = "foreach циклінің жұмысы" + "\r\n";
        foreach (Tovar t in ckl.tovar)
        {
        s = s + t.Naz + " " + t.Cena.ToString() + "\r\n";
        }
        s = s + "for цикл жұмысы" + "\r\n";
        for (int i = 0; i < kol; i++)
        {
        s = s + ckl.tovar[i].Naz + " " +
ckl.tovar[i].Cena.ToString() + "\r\n";
        }
        textBox3.Text = s;
       }
       }
     }
```

Бағдарлама жұмысы 25.2-суретте көрсетілген.



25.2-сурет – Қосымшаның интерфейстерсіз жұмысы

Айта кететін жәйт, қосымшада foreach циклі массив типіндегі ckl.tovar айнымалысы үшін ғана қолданылады, айнымалының кіріктірілген интерфейсі бар.

Бірақ, егер біз foreach циклін Cklad класс объектісінің tovar массиві үшін емес, Tovar класының объекттері үшін қолданатын болсақ, мысалы

```
foreach (Tovar t in ckl)
 {
 s = s + t.Naz + " " + t.Cena.ToString() +
"\r\n";
 },
```

онда, компилятор қосымшада кеткен қателік туралы хабарлама шығарады:

«foreach statement cannot operate on variables of type ' WindowsFormsApplication1.Form1.Cklad ' because 'Books' does not contain a public definition for 'GetEnumerator'»
(**foreach** операторын 'WindowsFormsApplication1.Form1.Cklad' типіндегі айнымалыларда қолдануға болмайды, өйткені осы кластың айнымалыларында **'GetEnumerator'** әдісінің айқын анықтамасы жоқ).

Бағдарламаға керекті интерфейсті қосайық, ол үшін бағдарламаға қосымша атаулар кеңістігін қосу керек:

```
using System.Collections;
```

Cklad класы IEnumerable интерфейсін мұралануы керек:

```
class Cklad : IEnumerable
```

Cklad класының денесіне GetEnumerator әдісін қосу керек:

```
public IEnumerator GetEnumerator()
 {
 for (int i = 0; i < 4; i++) yield return
tovar[i];
}
```

Т.А.Павловскаяның кітабынан алынған кейбір түсініктемелерді бере кетейік [2].

«Сонымен, егер класс элементтерін бір бірден қарастыру үшін foreach циклі қолданылған болса, онда GetEnumerator, Current, MoveNext, Reset төрт әдісін орындау керек. Мысалы, егер кластың ішкі элементтері массив түрінде ұйымдастырылған болса, онда кластың жабық өрісін сипаттау керек болады, онда ағымдағы индекс сақталады. MoveNext әдісінде индекстің өзгері 1-ге тең, массив шекарасынан асып кету шарты тексеріледі. Current әдісінде массив элементі ағындағы индекс бойынша қайтарылады және т.б. »

Бұл қызықты жұмыс емес, бірақ оны жиі орындауға тура келеді, сондықтан 2.0 версияда объекте бір бірден қарастыруды жеңілдететін құралдар – итераторлар енгізілген.

Итератор дегеніміз – код блогы, онда объект элементтерін бір-бірден қарап шығу реттілігі анықталады. `Foreach` циклінің әрбір қадамында итератордың бір қадамы орындалады, осының нәтижесінде `yield` қызметтік сөзі арқылы кезекті мән қайтарылады.

2.0 версияда бір-бірден қарап шығуды орындау үшін – класта `IEnumerable` интерфейсі орындалатының көрсету және итераторды сипаттау керек. Оған қол жеткізу `IEnumerator` интерфейсінің `MoveNext` және `Current` әдістері арқылы орындалады.

`foreach` циклінің әрбір қадамында итератор үшін «қоршам» - қызметтік объект әзірленеді, ол объект итератордың ағымдағы жағдайын сақтайды. Басқаша айтқанда, итераторды құраушы код үзіліссіз түрде орындалмайды, ол жеке итерацияларға бөлінген және осы аралықтардың арасында итератордың күйі сақталады.

Бағдарламада келтірілген екі өзгеріс `foreach` циклін қолдануға мүмкіндік береді, алдынғы қарастырған жағдайда бұл циклды қолданған кезде қате кеткені туралы хабарламаны шығаратын.

Бағдарламаның «тұрып қалуына» себеп болатын `foreach` циклінің жұмыс ерекшелігін ескере кету керек. Егер біздің бағдарламада тауардың бірнеше объекттерін енгізгеннен кейін, бірақ массив толық толмай тұрып, қоймадағы тауарларды қарап шығу режімін қосатын болсақ, онда бағдарлама жоқ мәндерді шығару кезінде «тұрып қалады», сондықтан foreach цикліндегі мәндерді бақылау керек.

`For` циклінде мұндай кемшіліктер жоқ, өйткені ондағы ақырғы мән ағымдағы ауқымды `kol` айнымалысының мәнімен анықталады.

```
for (int i = 0; i < kol; i++)
{
    s = s + ckl.tovar[i].Naz + " " +
ckl.tovar[i].Cena.ToString() + "\r\n";
}
```

25.4 Өзін-өзі тексеру сұрақтары

1 Интерфейс ұғымы?

2 Интерфейстік класты мұралану және қарапайым класты мұралану арасындағы айырмашылық неде?

3 Интерфейстік класс және қарапайым класс арасындағы айырмашылық неде?

4 Интерфейстік кластарда атаулар арасында қарама-қайшылықтардың туындауы неліктен?

5 Бірнеше рет мұралану кезінде атаулар арасында болатын қарама-қайшылықты қалай шешуге болады?

6 Әдістерді желімдеудің мәнісі неде?

7 Әдістердің атауын өзгертудің мәнісі неде?

8 Интерфейстердің артықшылықтары неде?

9 Интерфейстер қандай типте болуы мүмкін?

10 Интерфейсті жариялағанда қандай қызметтік сөз қолданылады?

26 КЛАСТАРДЫҢ КОМПОЗИЦИЯСЫ ЖӘНЕ КОЛЛЕКЦИЯСЫ

26.1 Кластардың композициясы және коллекциясы ұғымы

Егер класс өз өрістерінде басқа класс объекттерін пайдаланатын болса, онда кластардың осындай бірлестігі композиция деп аталады.

Класс композициясы бұл - бұрын жазылған қосымша кодын қайта пайдалану түрлерінің бірі. Мысалы, «дәріхана» класын «дәрілер» класының композициясы, яғни «дәрілер» класының түрлі объекттерінің массиві ретінде қарастыруға болады.

Композиция мысалы ретінде «гараж» класында «автомобиль» объекттерінің бірлестігін келтіруге болады.

Деректердің бір құылымда бір типті объекттердің бірлестігі коллекция деп аталады.

Коллекция көптеген объекттерді өңдеу функцияларын қамтамасыз етуі керек, яғни объекттерді сақтау мен жою және объекттерді қосу, жаңарту бойынша операцияларды ұсыну.

Әдетте коллекция жеке класпен анықталады.

Бүкіл коллекцияны сипаттайтын класты коллекция класы деп атайды.

Кластар композициясы бойынша қарастырылған мысалдарда екінші класс бірінші кластың объекттер коллекциясы қызметінде болады. Мысалы, «гараж» класы «автомобиль» класының объекттер коллекциясы болып табылады. «Дәріхана» класы – «дәрілер» класы объекттерінің коллекциясы.

Барлық коллекциялар кластарын шартты түрде сызықты және сызықты емес деп бөлуге болады.

Сызықты коллекцияларды келесі коллекциялардың кластары ұсынады:

– индекстелген қол жеткізу, мысалы, сөздіктер мен анықтама кітаптары;

– тікелей қол жеткізу, мысалы, массивтер;

– ретті қол жеткізу, мысалы, стектер, кезектер, тізімдер. Көптеген «тізімдер» динамикалық массивтермен анықталған;

Сызықты емес коллекцияларды келесі коллекциялардың кластарын ұсынады:

– иерархиялық, мысалы, әр түрлі бұтақтар тәрізді құрылым. Классификацияның иерархиялық жүйесі немесе құжаттарды іздестіру массивін ұйымдастыру;

– топтық, мысалы, түрлі жиынтықтар, тораптық құрылымдар, графтар.

Кластар композициясы бұрын құрылған кластарды немесе қосымшалар үзінділерін қолданған кезде бағдарламалаудың қуатты құралы болып табылатының ерекше атап өту керек.

C# бағдарламалау ортасынада басқа кластардың бір типті объекттерін коллекциялау үшін көптеген түрлі кластар қолданылады, мысалы, тізімдер, стектер, кезектер, сөздіктер, бұтақтар, және көптеген басқа да коллекциялар.

26.2 Кластың композициясы мен коллекциясын қолдану мысалы

Сан алуан коллекциялар кластарының ішінен ең қарапайым кластар коллекциясын – стекті қарастырайық.

26.1-есеп. Стек элементіне КІТАП класының объектісі жатады. Кітаптарды жинағанда кітаптарды тек үстіңгі жақтан ғана алуға немесе қосуға болады.

Кластың деректерін ашық деп есептейік (кітап авторы, атауы, бағасы). Класс әдісі ретінде параметрлері берілген конструкторды ғана қолданамыз.

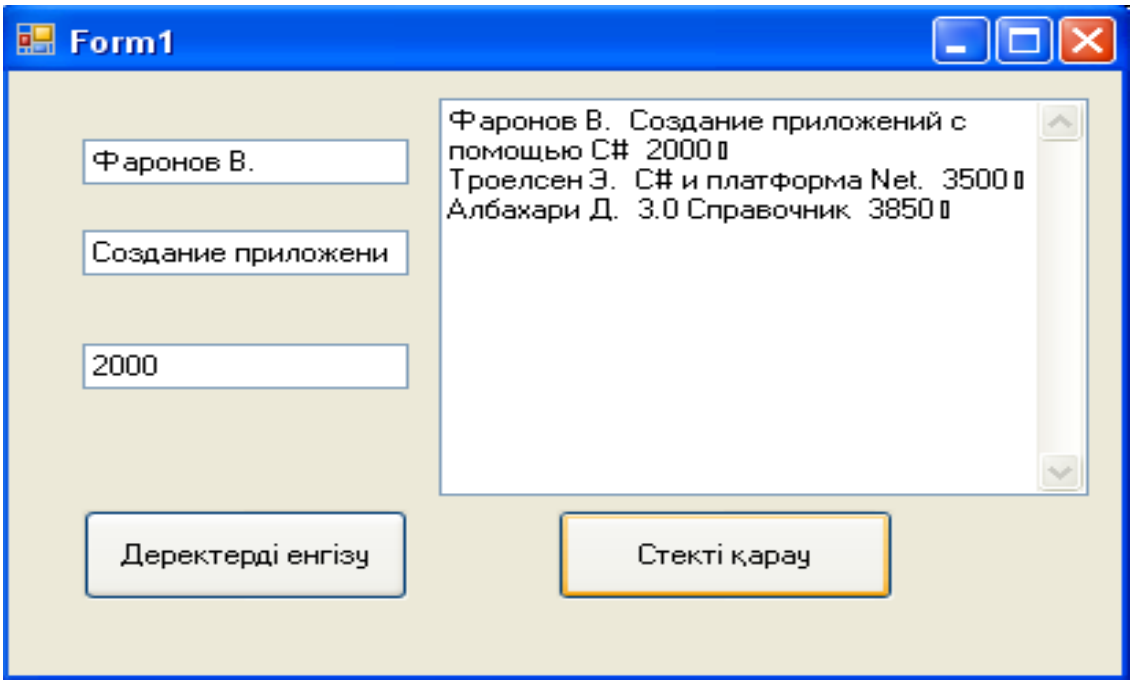
Тізімдік құрылым түрінде коллекцияны ұйымдастыру үшін стандартты Stack құрылымын қолданамыз.

Form1.cs файлының коды:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
 public partial class Form1 : Form
 {
 public Form1()
 {
  InitializeComponent();
 }
 public class Kitap
 {
  public string Atayi;
  public string Avtor;
  public int Bagasi;
  public Kitap(string sa, string sb, int sc)
  {
  Avtor = sa; Atayi = sb; Bagasi = sc;
  }
 };
 public Stack<Kitap> vstek = new Stack<Kitap>();
```

```
public string ss = "";
private void button1_Click(object sender, EventArgs e)
{
 string a, b;
 a = textBox1.Text;
 b = textBox2.Text;
 c = Convert.ToInt32(textBox3.Text);
 Kitap Tom = new Kitap(a, b, c);
 vstek.Push(Tom);
}
private void button2_Click(object sender, EventArgs e)
{
 Kitap Tom = new Kitap("", "", 0);
 foreach (Kitap T in vstek )
 {
 ss = T.Avtor + " " + T.Atayi+ " " +
 Convert.ToString(T.Bagasi) + " \n";
 textBox4.AppendText(ss);
 }
}
}
}
```

Қосымша жұмысы 26.1-суретінде көрсетілген



26.1-сурет – Қосымшаның жұмысы

Айта кететін жәйт, C# тілінде объекттер коллекцияларын жүзеге асыратын бірнеше кластар бар, мысалы, массив, стектер, кезектер, т.б. Массивтер ғана C# тілінінің конструкциясы болып келеді, ал басқа коллекциялар - .NET платформасының кластары (Framework

кітапханалары). Сондықтан осы кітапхана коллекциясының құрамымен танысудың маңызы бар.

26.3 Framework-тың кейбір коллекциялары

Коллекциялар кластарын жүзеге асыру интерфейстер көмеегімен орындалады. .NET платформасы, анығырақ айтқанда Framework кітапханасы коллекцияларды ұйымдастыру, индекс бойынша оның элементтеріне қол жеткізу, коллекцияда іздестіруді, т.б. орындау үшін бағдарламашыға қосымша интерфейстерді ұсынады.

ICollection интерфейсі объекттер коллекциялары үшін стандартты болып келеді. Ол IEnumerable, IEnumerator интерфейстерін қолданып өз элементтерін бір бірден қарап шығуға; коллекция өлшемін анықтауға; күрделі өңдеу жұмыстары үшін коллекцияны массивке көшіруді қамтамасыз етеді.

IList интерфейсі массив типіндегі коллекцияларды құру үшін стандартты болып келеді. Ол IEnumerable, IEnumerator интерфейстерін қолданып өз элементтерін бір бірден қарап шығуды да қамтамасыз етеді. Сонымен қатар интерфейс қайта жүктелетін индексатордың көмегімен элементке тікелей қол жеткізуге мүмкіндік туғызады. IList интерфейсі индекс бойынша коллекция элементін қосуды, жоюды, редакциялауды қамтамасыз етеді.

Стандартты емес коллекциялармен жұмыс жасау үшін көптеген интерфейстер бар, мысалы, IDictioaryEnumerator интерфейсі сөздік типтегі коллекциялармен жұмыс жасауға көмекеседі.

Интерфейстерді қолдану Framework кітапханасында коллекциялардың бірнеше класын құруға мүмкіндік береді, мысалы, Array**,** ArrayList**,** LinkedList**,** Queue**,** Stack және басқалар.

Коллекцияны визуалды көрсету үшін C# тілінде бірнеше басқару элементтері (немесе коллекциялар кластары) бар, мысалы, DataGridView, TreeView, т.б.

Сіздер коллекцияларды «Қолданбалы бағдарламалау» және «Деректер базалары» пәндерінде толық қарастыратын боласыңдар, ал осы пәнде ArrayList коллекциясының мысалында коллекциялар класымен жұмыс жасауды қарастырамыз

26.4 ArrayList коллекциясы

ArrayList Класс System**.**Collections**.**ArrayList атаулар кеңістігіне тиісті және кез келген типтегі объекттерді сақтау үшін қолданылады. ArrayList класының негізгі қасиеттері мен әдістері 26.1-кестеде көрсетілген.

26.1-кесте - ArrayList класының негізгі қасиеттері мен әдістері

Қасиеттер мен әдістер	Сипаттама
public static ArrayList (IList: List);	List коллекциясының негізінде ArrayList объектін құрады
public virtual int Add(Oblect Value);	Тізім соңына жаңа объектті қосады және индексті қайтарады.
public virtual viod AddRange (ICollection coll)l	Тізім соңына бірнеше объекттерді қосады.

26.1-кестенің жалғасы

Қасиеттер мен әдістер	Сипаттама
public virtual int BinsrySearch(Qbject Value);	Value объектін іздестіреді және оның индексін қайтарады, ал егер объект табылмаса, теріс санды шығарады.
public virtuai int Capacity {get;}	Осы қасиет тек қана «оқу» үшін қолданылады, коллекцияның ағымдағы сиымдылығын сақтайды.
public virtual void Clean();	Коллекцияның барлық элементтерін жояды.
public virtual bool Containse (Object Value);	Коллекцияның Value деген элементі бар болса, true мәні қайтарылады.
public virtal int Count {get;};	Осы қасиет тек қана «оқу» үшін қолданылады, коллекцияның ағымдағы ұзындығын сақтайды.
public static ArrayList FixedSize(ArrayList AL);	Әдіс объектті қайтарады. Оның элементтерін өзгертуге болады, бірақ қосуға немесе жоюға болмайды.
public virtual IEnumerator GetEnumerator();	Объект үшін итераторды қайтарады.
public virtual ArrayList GetRange(int Indx, int Count);	Элементтер диапазонын қайтарады.
public virtual int IndexOf(Object Value);	Value деген мәні бар элемент индексін қайтарады.
public virtual void Insert (int Indx, Object Value);	Коллекцияның Indx керекті орнына Value элементін орналастырады.
public virtual void	Элементтер диапазонын

InsertRange)int Indx, IColliction col);	кіргізеді.
public virtual bool IsFixedSize{get;}	Егер объекттің өлшемі тиянақты болса, онда true мәнін қайтарады.
public virtual bool IsReadOnly{get;}	Егер объекттің «оқу» үшін ғана арналған элементтері болса, онда true мәні қайтарылады.
public virtual Object this[int Indx]{set; get;}	Бұл қасиет элементтерді индексі бойынша қолдануға мүмкіндік береді.

26.1-кестенің жалғасы

public virtual void RemoveRange(int Indx, int count);	Индексі Indx тең элементтен бастап коллекциядан count элементті жояды.
public static ArrayList Repeat(Object Value, int count);	Value элементі count рет қайталанатын коллекцияны қайтарады.
public virtual void Reverse();	Элементтердің реттілігін кері бағытқа өзгертеді.
public virtual void SetRange (int Indx, ICollection col);	Indx индексінен бастап col коллекциясын кіргізеді.
public virtual void Sort();	Коллекцияны сұрыптайды
public virtual void TrimToSize();	Коллекцияның сиымдылығын орнатады. Сиымдылық элементтерінің санына тең.
public static ArrayList ReadOnly(ArrayList AL);	Коллекция элементтеріне тек қана «оқу» үшін режімін орнатады
public virtual void Remove (Object Value);	Коллекциядан мәні Value-тең бірінші элементті жояды.
public virtual void RemoveAt (int Indx);	Коллекциядан индексі Indx-тең элементті жояды.

Әдетте ArrayList коллекциясының сиымдылығы 16 элементке тең. Коллекция кеңейген кезде оның сиымдылығы екі еселенеді, 32, 64, 128 элементтерді құрайды. TrimToSize() әдісі қолданылмайтын элементтерді кесіп тастайды.

26.1-есеп. ArrayList класын «Автомобиль» объекттерін біріктіретін «Гараж» коллекциясын ұйымдастыру үшін қолдану. Коллекциямен жұмыс жасау үшін 26.1-кестесінің қасиеттері мен әдістерін қолдану керек. Бұл мысалда «Автомобиль» объектісінің өрістер саны тек екеу болады – үлгісі мен бағасы. Қосымша беттер жиынтығы – TabControl басқару элементі қолданылды, осы элемент арқылы бағдарлама менюі мен «қосымша формалар» қосылады.

Бағдарламада Panel басқару элементі – бағдарламаның нәтижесін шығару немесе жасыру үшін панелі қолданылады: «Қарап шығу» режімі, «Баға графигін қарап шығу» командасы.

ArrayList коллекциясының ерекшелігі – кез келген типтегі объекттерді сақтау үшін қолданылуында, сондықтан 26.1-кестесіндегі кейбір әдістер интерфейстік кластарды қосымша күйге келтіруді талап етеді.

Бағдарламаны тексеру барысында егер коллекция объекттер саны 4-тен аспаса, онда коллекция сиымдылығы 16-ға емес 4-ке тең болады. Коллекцияны кеңейткен кезде сиымдылық екі есе өседі – 8 (егер объекттер саны 4-тен үлкен, бірақ 9-кіші болса), 16 және т.б.

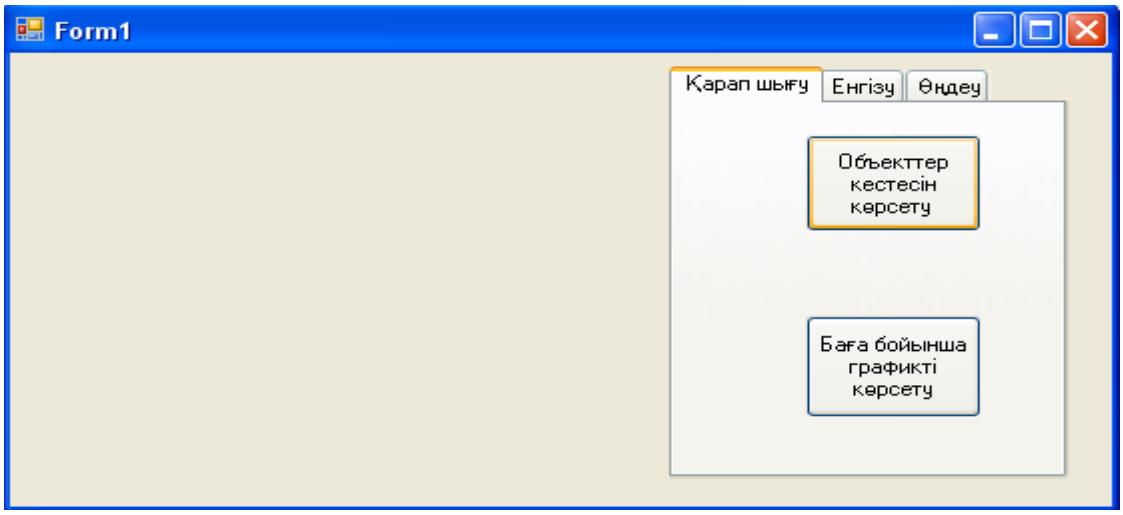
Form1.cs файлының коды:

```
using System;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
 public partial class Form1 : Form
 {
 public Form1()
 {
  InitializeComponent();
  textBox3.Text = "";
  panel2.Visible = true;
  panel3.Visible = false;
  panel1.Visible = false;
 }
 public class Avto
 {
  public string Marka;
  public int Bagasi;
  public Avto(string sm, int sc)
  {
  Marka = sm; Bagasi = sc;
  }
 };
 public ArrayList Garaj = new ArrayList();


 private void button1_Click(object sender, EventArgs e)
 {
  panel2.Visible = true;
  panel3.Visible = false;
  panel1.Visible = false;
  string n;
```

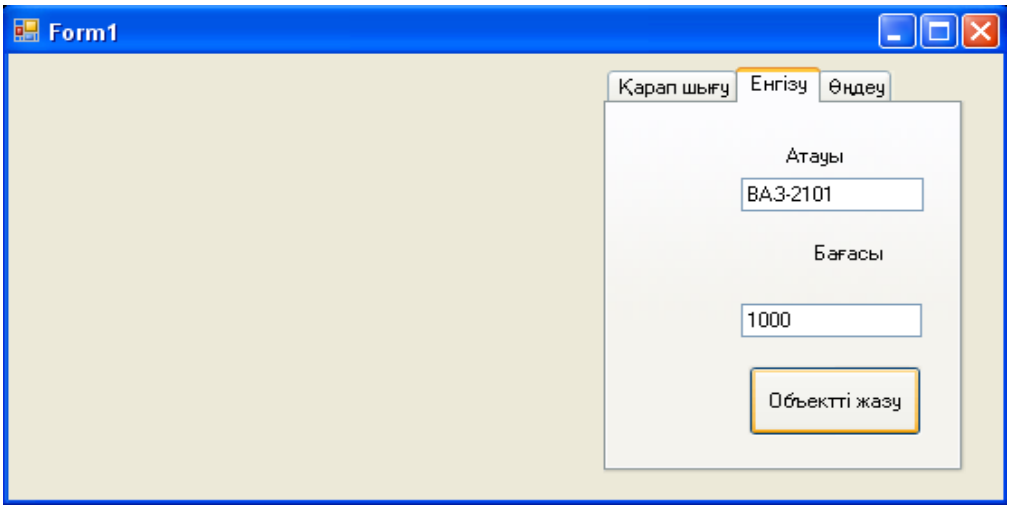
```
        int c;
        n = textBox1.Text;
        c = Convert.ToInt32(textBox2.Text);
        Avto at = new Avto(n, c);
        Garaj.Add(at);
       }

       private void button2_Click(object sender, EventArgs e)
       {
        panel2.Visible = false;
        panel3.Visible = false;
        panel1.Visible = true;
        int i = 0;
        dataGridView1.Rows.Clear();
        foreach (Avto at in Garaj)
        {
        dataGridView1.Rows.Add();
        dataGridView1.Rows[i].Cells[0].Value = at.Marka;
        dataGridView1.Rows[i].Cells[1].Value = at.Bagasi;
        i++;
        }
       }

       private void button3_Click(object sender, EventArgs e)
       {
        panel2.Visible = false;
        panel3.Visible = false;
        panel1.Visible = false;

        this.Invalidate();
       }
       private void button4_Click(object sender, EventArgs e)
       {
        panel2.Visible = false;
        panel3.Visible = true;
        panel1.Visible = false;
        int i = Garaj.Capacity;
        int j = Garaj.Count;
        textBox3.AppendText("Коллекция " + j.ToString()+ "
элементтерден тұрады" + "\r\n");
        textBox3.AppendText("Ағымдағы коллекцияның өлшемі = " +
i.ToString() +" элемент" + "\r\n");
       }

       private void button5_Click(object sender, EventArgs e)
       {
        panel2.Visible = false;
        panel3.Visible = true;
        panel1.Visible = false;
        Avto at1 = new Avto("", 0);
        Avto at2 = new Avto("", 0);
        int i=0;
```

```
        foreach (Avto at in Garaj)
        {
        if (i==0) at1=at;
        at2 = at;
        i++;
        }
        int j = Garaj.Count;
        Garaj.RemoveAt(j-1);
        Garaj.RemoveAt(0);
        Garaj.Insert(0, at2);
        Garaj.Insert(j-1, at1);
        textBox3.AppendText("Алмастыру аяқталды" + "\r\n");
       }

      private void button6_Click(object sender, EventArgs e)
      {
       Garaj.RemoveAt(0);
       panel2.Visible = false;
       panel3.Visible = true;
       panel1.Visible = false;
       textBox3.AppendText("Коллекциядан 0 элемент жойылды" +
"\r\n");
      }
      private void Form1_Paint(object sender, PaintEventArgs e)
      {
       string ss;
       Graphics g = e.Graphics;
       Pen myPen = new Pen(Color.Red, 2);
       g.DrawLine(myPen, 50, 150, 50, 50);
       g.DrawLine(myPen, 50, 150, 250, 150);
       int h,i = 0;
       foreach (Avto at in Garaj)
       {
       h = at.Bagasi/100;
       if (i < 8)
       {
        g.FillRectangle(Brushes.Blue, i * 25 + 55, 150 - h, 20,
h);
        ss = i.ToString();
        g.DrawString(ss, new Font("10_IC_1", 12),
Brushes.Black, (i + 1) * 25 + 30, 160);
       }
       i++;
       }
      }
      }
     }
```

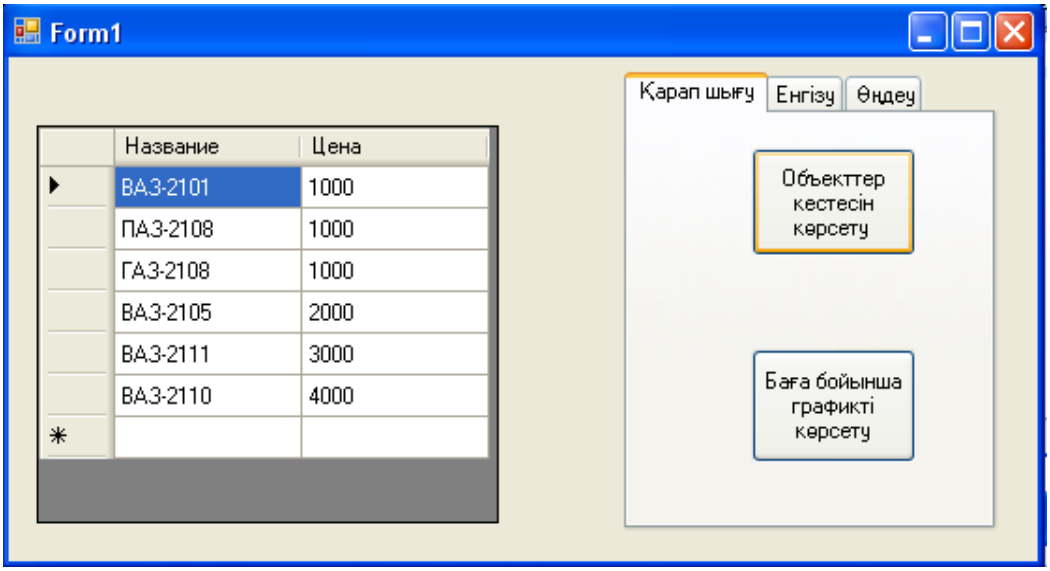
Қосымша жұмысы 26.2 –26.6 суреттерінде көрсетілген.



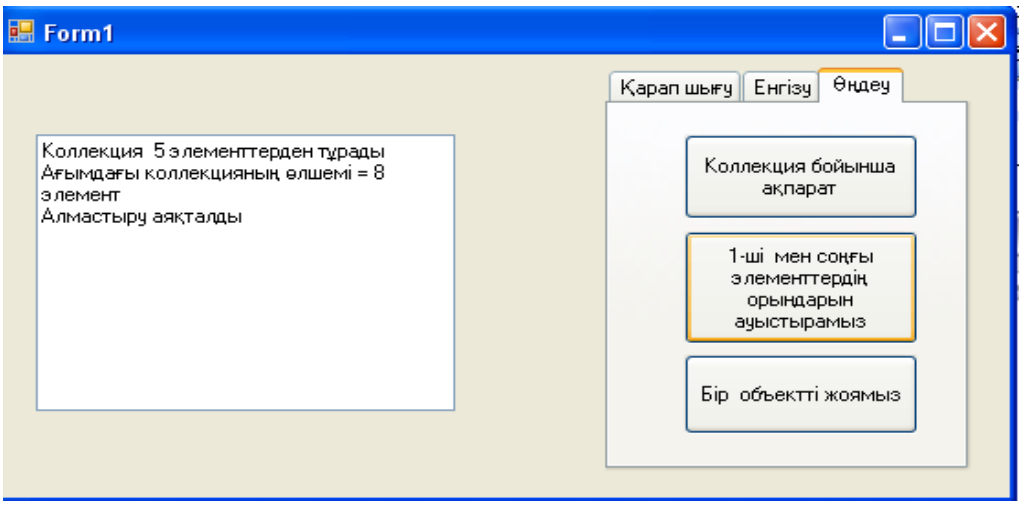
26.2-сурет – Суреті бар қосымша жұмысының терезесі



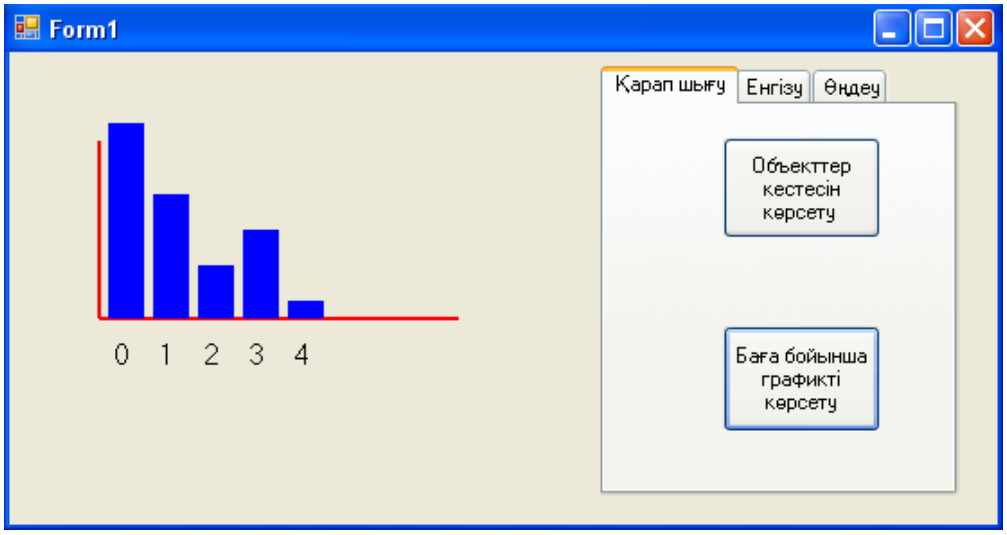
26.3-сурет – Коллекцияның деректерін енгізу режімі



26.4-сурет – Коллекция объекттерінің кестелерін қарау режімі



26.5-сурет – Коллекция туралы ақпаратты қарау режімі



26.6-сурет – Коллекция объекттерінің баға графигін көрсету

Осы мысалда негізгі назар жаңа басқару элементтерін және кластарды қолдану технологиясына аударылған.

26.5 Өзін-өзі тексеру сұрақтары

1 Класс композициясы ұғымы.

2 Класс композициясының негізгі мақсаты неде?

3 Класс коллекцияның ұғымы.

4 Бір типтегі объекттерді коллекциялауда әдетте қандай біріктіретін құрылым қолданылады?

5 Қандай коллекциялар тікелей қол жеткізу коллекцияларына жатады?

6 Қандай коллекциялар тізбекті қол жеткізу коллекцияларына жатады?

7 Қандай коллекциялар иерархиялық коллекцияларға жатады?

8 Қандай типтегі объекттер ArrayList класының коллекциясына қосыла алады?

9 ArrayList класының қандай қасиеті коллекцияның ағымдағы сиымдылығын сақтайды?

10 ArrayList класының қандай қасиеті коллекцияның ағымдағы ұзындығын сақтайды?

11 ArrayList класының қандай әдісі объектті коллекцияның керекті орнына орналастыруға мүмкіндік береді?

27 ДЕЛЕГАТТАР

27.1 Делегат ұғымы

«Біртипті» әдістерді қолдана отырып есептеу жүргізуді қажет ететін көптеген есептер болады, мысалы, нақты типтегі математикалық функциялар (тригонометриялық функциялар, логарифмдер, экспонент, т.б.). Кейде осындай есептерде формалды параметрі ретінде есептелетін функциясының атауы болып келетін әдісті қолдану қажеті туындайды.

C# тілінде әдістер тек класс ішінде ғана орналастырыла алады. Сондықтан C# тілінде «біртипті» әдістерге сілтемелерді сақтауға мүмкіндік беретін арнайы класс қосылған. Ол класс Делегат деп аталады.

Делегат дегеніміз – әдістерге сілтемелерді сақтауға арналған арнайы класс.

Анықтама бойынша класс типіндегі айнымалы объект болып келеді. Әрқайсысы функця болып келетін (немесе функцияға сілтеме) объекттердің жиынтығын сипатауға мүмкіндік беретін класс функционалдық тип деп аталады.

Сонымен, C# тілінде делегаттар функционалдық типтерді сипаттау үшін арналған.

Осындай кластың даналары – функцияларға (әдістерге) сілтемелер болып табылады, айнымалыларға сияқты оларға да компьютер жадысынан орындар бөлінеді (олардың алғашқы адрестері функцияларға кіру «нүктелері» болып келеді және сілтемелермен беріледі).

Делегаттың осындай ерекшелігі оны қолданудың екі жолын анықтады - есепті өз бетімен шешу үшін немесе оқиғаларды қолдау үшін (келесі бөлімде қарастырылады).

Осы бөлімде кейбір есептерді шешу үшін делегеттардың жеке қолданылуын қарастырамыз.

27.2 Делегаттың сипаттамасы

Формат записи делегата фактически задает сигнатуру (описание) методов, которые могут быть вызваны с его помощью:

Делегаттың жазылу пішімінде әдістердің сипаттамасы берілген:

```
[спецификаторлар] delegate <тип> <атауы> (<параметрлер>);,
```

мұнда

спецификаторлар делегатқа қол жеткізу шартын анықтайды;

`delegate` —қызметтік сөз;

`<типі>` — қайтарылатын нәтиженің типі;

`<атауы>` —делегат атауы (бірегей идентификатор);

<параметрлер> — шақырудың формалды параметрлері.

Мысалы, нақты типтегі аргументі бар нақты типтегі барлық функцияларды сипаттау келесі түрде жазылады:

```
public delegate double Funk(double argym);
```

Осы сипаттамаға сәйкес кез-келген функцияны класс конструкторының, яғни делегаттың шақыру параметрі ретінде қолдануға болады. Ол делегаттың нақты данасын - функцияға сілтемені қайтарады. Делегат жұмысының ерекшелігі – сілтемелердің қосымшаны компиляциялау кезеңінде емес, қосымша жұмысының барысында (динамикалық түрде) құрылуында.

27.3 Делегатты қолдану мысалы

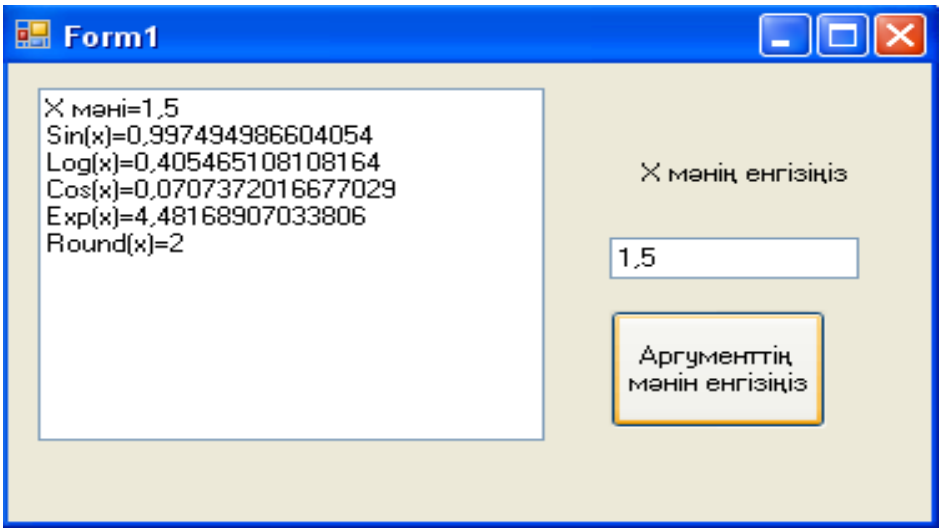
Делегат жұмысының механизмін жақсы түсіну үшін келесі есептің шешімін қарастырайық: нақты типтегі аргументі бар нақты типтегі бес функцияны есептеу үшін қосымшаны дайындау керек: Sin(x), Log(x), Cos(x), Exp(x), Round(x). Есептеу функциясы үшін делегатты қолдану керек.

Form1.cs файлының коды:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
 public partial class Form1 : Form
 {
 public Form1()
 {
  InitializeComponent();
 }
 public delegate double Funk(double x);//делегатты
жариялау
 public double rab(Funk f, double x) { return f(x); }
 // делегатты қолданатын функцияны жариялау
 private void button1_Click(object sender, EventArgs e)
 {
  string st;
  double x, y;
  textBox2.Text = "";
  x = Convert.ToDouble(textBox1.Text);
  st = "x-тің мәні=" + textBox1.Text + "\r\n";
  textBox2.AppendText(st);
  y = rab(Math.Sin, x);  // делегатты қолдану
```

```
        st = "Sin(x)=" + y.ToString() + "\r\n";
        textBox2.AppendText(st);
        y = rab(Math.Log, x);       // делегатты қолдану
        st = "Log(x)=" + y.ToString() + "\r\n";
        textBox2.AppendText(st);
        y = rab(Math.Cos, x);       // делегатты қолдану
        st = "Cos(x)=" + y.ToString() + "\r\n";
        textBox2.AppendText(st);
        y = rab(Math.Exp, x);       // делегатты қолдану
        st = "Exp(x)=" + y.ToString() + "\r\n";
        textBox2.AppendText(st);
        y = rab(Math.Round, x);     // делегатты қолдану
        st = "Round(x)=" + y.ToString() + "\r\n";
        textBox2.AppendText(st);
       }
       }
    }
```

Қосымша жұмысы 27.1-суретте көрсетілген.



27.1-сурет – Делегатты қолдану

Қосымша кодына мына бөлікті қосуға болады, мысалы,

```
    private void button1_Click(object sender, EventArgs e)
    {
     string st;
     Funk[] ff={Math.Sin, Math.Log, Math.Cos,
Math.Exp,Math.Round};
     string[] sfu = { "Sin", "Log", "Cos", "Exp", "Round" };
     double x, y;
     textBox2.Text = "";
     x = Convert.ToDouble(textBox1.Text);
     st="X мәні =" +textBox1.Text+ "\r\n";
     textBox2.AppendText(st);
     for (int i = 0; i < 5; i++)
```

```
 {
 y = rab(ff[i], x);
 st = sfu[i]+"=" + y.ToString() + "\r\n";
 textBox2.AppendText(st);
 }
}
```

Бағдарламада делегаттарды жариялау және қолдану идеясы осы мысал бағдарламасының кодында толық қарастырылған.

Делегаттардың ерекшелігін тағы да бір рет атап өтейік: шақырылатын әдістерді динамикалық түрде шақыру (бағдарламаның орындалуы барысында). Делегаттардың осы ерекшелігі бағдарламаның базалық конструкцияларын құру кезінде жиі қолданылады, осы конструкцияда әр түрлі бағдарлама үзінділерін жазуға болады, мысалы «біртипті» әдістері бар менюдің фрагменттері.

27.4 Делегаттардың үйлесімділігі

Делегаттардың «үйлесімділігін» қарастыра отырып екі сұрақты талқылау керек – делегаттар типінің үйлесімділігі және делегаттардың объекттер даналарының үйлесімділігі.

Делегаттар типтерінің жазылу пішімдері үйлесімді болғанымен, олар бір-бірімен үнемі үйлесімді бола бермейді, мысалы,

```
public delegate void Funk1();
public delegate void Funk2();
. . .
Funk1 d1 = Metod1;
Funk2 d2 = d1; // типтер үйлесімділігі туралы қате
```

Егер делегатты жариялау үшін бір ғана тип қолданылған болса, онда делегаттардың даналары үйлесімді (тең) болады. Егер екі делегат бір әдіске бағытталған сілтемені қолданатын болса, онда бір типті делегаттардың даналары бір-біріне тең деп есептеледі, мысалы,

```
public delegate void Funk1();
. . .
Funk1 d1 = Metod1;
Funk1 d2 = Metod1;
```

Осы мысалдағы d1 және d2 делегаттарының даналарын бір-біріне тең деп есептеуге болады.

27.5 Делегаттардың базалық кластарының әдістері

Айта кететін жәйт, .NET платформасындағы CTS-те абстрактылы `System.Delegate` және `System.MulticastDelegate` кластары бар, олар арқылы құрылатын делегаттар кейбір әдістерді мұралануы мүмкін.

Құрылатын делегаттар System.MulticastDelegate –тан мұраланатын әдістер мен қасиеттердің тізімі төменде көрсетілген:

public MethodInfo Method {get:}– бұл қасиет делегат нұсқайтын әдістің атауын қайтарады;

public object Target {get:} – егер делегат әдісті – класс мүшесін нұсқайтын болса, онда Target әдіс орналасқан кластың атауын; егер әдіс статикалық болса, онда Target null мәнін қайтарады;

public static Delegate Combine(Delegate[]) – делегат өңдейтін әдістер тобына жаңа әдістерді қосады;

public static Delegate Remove(Delegate source, Delegate value) – source делегатының әдістер тізімінен value әдісін жояды;

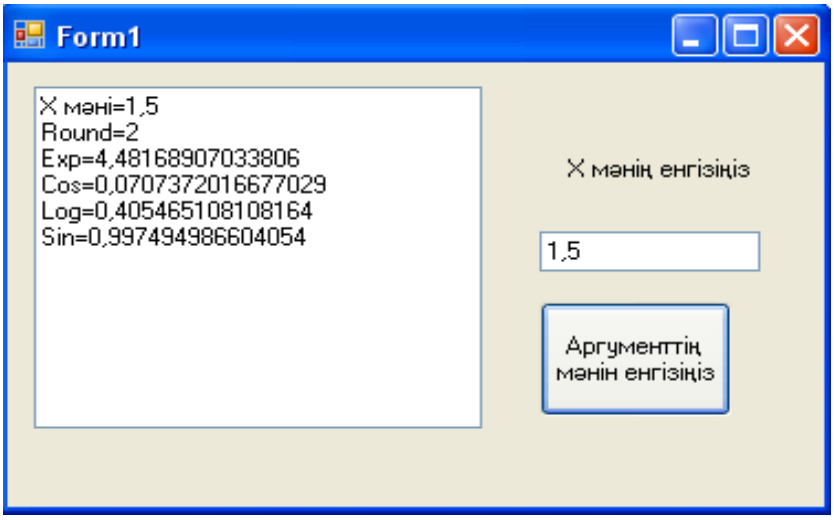
public sealed override Delegate[] GetInvocationList() – делегатпен байланысқан барлық әдістерді қайтарады.

Делегаттар үшін көпадресті делегат деген ұғым бар – саны бойынша кез келген болатын функцияларды нұсқайтын делегат. C# тілінің барлық делегаттары System.MulticastDelegate-тің туынды класы болғандықтан, C# тіліндегі кез келген делегат көп адресті делегат болып келеді.

Нақты типтегі функцияларды есептеуді орындайтын бағдарламаның делегаты үшін «көп адресті» қасиетін қолданайық. Бағдарлама коды батырманы басу өңдеуішінде жазылған, сондықтан кодтың осы үзіндісі ғана көрсетілген:

```
 private void button1_Click(object sender, EventArgs e)
  {
   string st;
   Funk[] ff ={ Math.Sin, Math.Log, Math.Cos, Math.Exp,
Math.Round };
   string[] sfu = { "Sin", "Log", "Cos", "Exp", "Round" };
   double x, y;
   textBox2.Text = "";
   x = Convert.ToDouble(textBox1.Text);
   st = "X мәні=" + textBox1.Text + "\r\n";
   textBox2.AppendText(st);
   Funk df = Math.Sin;
   for (int i = 1; i < 5; i++)
   df += ff[i]; //df = df + ff[i];
   for (int i = 4; i >= 0; i--)
   {
   y = rab(df, x);
   df -= ff[i]; //df = df - ff[i];
   st = sfu[i] + "=" + y.ToString() + "\r\n";
   textBox2.AppendText(st);
   }
  }
```

Қосымша жұмысы 27.2-суретінде көрсетілген.



27.2-сурет –Көпадресті делегатты қолдану

27.2-суретте көрсетілгендей «көп адресті» делегаттың жұмысы стектің принципі бойынша ұйымдастырылған.

Делегаттар әдістерге арналған сілтемелерді сақтау үшін арнайы класс ретінде қолданылады. Айта кететін жәйт, кез келген объект (санымен қатар делегат объектісі) сілтеме түрінде болады және оны кейбір әдістерге параметр ретінде жазуға болады.

Осылайша функционалды параметрлеу қамтамасыз етіледі: әдістерге тек қана деректерді ғана емес, сонымен қатар оларды өңдейтін түрлі әдістерді беруге болады.

27.6 Өзін-өзі тексеру сұрақтары

1 Делегат ұғымы.

2 Қандай класс функционалды тип деп аталады?

3 Не нәрсе делегат типіндегі кластың данасы болып табылады?

4 Делегатты жариялағанда қандай қызметтік сөз қолданылады?

5 Делегатты шақыру параметрі ретінде қандай әдісті немесе функцияны қолдануға болады?

6 Делегатты шақыру параметрлеріне сілтемелер қай уақытта құрылады?

7 Делегаттардың типтері қай уақытта үйлесімді болады?

8 Делегаттардың даналары қай уақытта үйлесімді болады?

9 Делегаттарды құрған кезде .NET платформасының қандай абстрактлы кластарын мұралауға болады?

10 Көпадресті делегат ұғымы.

28 ОҚИҒАЛАР

28.1 Оқиға ұғымы

Windows операциялық жүйесі жұмысының негізінде оқиғаны басқару принципі жатады. Windows жүйесіне арналған барлық қосымшалар іске қосылғаннан кейін пайдаланушының әрекеттерін немесе операциялық жүйенің хабарламаларын күтеді және оларға белгілі бір тәртіпте жауап береді - қосымшаның хабарламалар кезегі арқылы оқиғалар құрылады.

Әрбір басқару элементінің (батырма немесе меню жолы) өз идентификаторы болады. Батырманы басқан кезде немесе меню жолын таңдаған кезде Windows қосымшаның хабарламалар кезегіне хабарламаны кіргізеді. Ол хабарламада қолданылған басқару элементінің идентификаторы болады. Windows операциялық жүйесі басқару элементінен келген хабарламаны өзітиісті қосымшаның (басқару элементі осы қосымшаға тиісті) кезегіне бағыттайды.

Егер форма терезесінде батырма басылған болса, онда батырманы басу факті (хабарламасы) Windows операциялық жүйесі арқылы форма терезесінің хабрламалар кезегіне қайтып келеді ады және осыдан кейін батырманың Click оқиғасы пайда болады.

Сонымен, оқиғаның пайда болу механизмі түсінікті, бірақ оқиға деген не?

Біз 21-бөлімде (класс құрылымы) класс оқиғаларын арнайы әрекеттер ретінде анықтадық, ол оқиғалар кластың пайдаланушы әрекеттеріне немесе бағдарламаның белгілі өзгерістеріне жауап беруіне мүмкіндік береді.

Осы анықтамадан кейін оқиғалар мен әдістер – оқиғалар өңдеуішіне синоним болып табылады.

Осы ұғымдарды анықтайық. Әрбір объект белгілі бір кластың данасы болып келеді. Класс қасиеттері оның жағдайын, ал класс әдістері оның әрекеттерін анықтайды.

Барлық объекттерде қасиеттер жиыны бірдей, бірақ объекттер қасиеттерінің мәні әртүрлі, сондықтан бір кластың объекттері әртүрлі жағдайларда бола алады. Мысалы, «Қызметкер» класының объекттерінде «кіріс» қасиетінің мәні әртүрлі болуы мүмкін және осы объекттер әртүрлі күйде болуы мүмкін. Мысалы, «автомобильді сатып алғым келеді, бірақ оған жағдайым жоқ …».

Барлық объекттердің әдістері бірдей және бір кластың барлық объекттерінің оқиғалар жиыны бірдей болады. Бірақ пайда болған оқиғаларды өңдейтін әдістер әртүрлі болуы мүмкін. Мысалы, форманың екі батырмасының («Деректерді енгізу», «2-формаға көшу») Click оқиғалары бірдей, бірақ оқиғалар өңдеуіштері (олардың кодтары) әр түрлі.

Сонымен, оқиға класс қасиеті ретінде барлық объекттерге ортақ болады және кластың әрбір нақты данасы үшін оның жүзеге асырылуы әртүрлі болады.

Оқиғалар кейбір хабарлама пайда болған кезде объекттің жекеленген әрекетін анықтауға мүмкіндік береді.

Сонымен, класс оқиғалары дегеніміз –пайдаланушы әрекеттеріне немесе бағдарламадағы өзгерістерге класқа жауап беруіне мүмкіндік беретін арнайы әдістер.

Пайдаланушының көптеген әрекеттеріне оқиғалар өңдеуіштерінің үлгілері құрылып қойылған (Properties ->Events).

28.2 Visual Studio.NET ортасының жиі қолданылатын кейбір оқиғалары

Windows–қосымшалардың әрбір басқару элементі үшін оқиғалардың өз жинағы анықталған. Оқиғалар өңдеуіштері бойынша үлгінің дайындамасы мына жолмен құрылады: керекті басқару элементінің Properties терезесіндегі, Events бетіндегі сәйкес оқиға атауының оң жағында орналасқан өрісті екі рет шерту.

Басқару элементтерінде жиі қолданылатын оқиғалар:

Click, DoubleClick – тышқанды бір немесе екі рет шерту;

KeyDown, KeyUp – кез келген пернені (пернелер тіркестері) басу немесе оны жіберу;

KeyPress – пернені басу (ASCII-коды бар);

MouseDown, MouseUp – тышқан батырмасын басу немесе оны жіберу;

MouseMove – тышқан орнын ауыстыру;

Paint – форма кескінін салу керек болған кезде пайда болады.

Мысалы, формада орналасқан «Басу» батырмасын екі рет шертсе, онда осы оқиғаның өңдеуіші құрылады, өңдеуіш тақырыбының жазылуы төменде көрсетілген:

```
    private void button1_Click(object sender,
EventArgs e).
```

Егер бізге батырманы екі рет басу нәтижесінде пайда болған оқиғаны өңдеу керек болса, онда форманың оқиғаларында «KeyDown» оқиғасының оң жағында өрісті басу керек, нәтижесінде осы оқиғаның өңдеуіш үлгісі пайда болады, өңдеуіш тақырыбының жазылуы мына түрде болады: private void Form1_MouseDoubleClick(object sender, MouseEventArgs e).

Мысал ретінде тағы екі оқиға өңдеуішінің үлгілерін көрсетейік:

```
    private void textBox1_KeyDown(object sender,
KeyEventArgs e)
```

```
    private void Form1_MouseClick(object sender,
MouseEventArgs e)
```

Олардың қашан құрылатынын, өзіңіз анықтап көріңіз?

Барлық оқиғаның өңдеуішінің үлгілерінде екі формалды параметр бар, сонымен қатар бірінші параметр (object sender) барлығында бірдей.

C# тілінде бағдарламалау бойынша көптеген оқулықтарда оқиға жұмысының механизмін түсіндірген кезде «жариялау-жазылу» үлгісі қолданылады. Осы үлгі бойынша хабарламаны (sender) жіберуші бір класс хабарламаны жариялайды, ал хабарламаны алушы басқа кластар осы хабарламаларды алуға жазылады.

Жоғарыда жазылған терминологияға сәйкес барлық хабарламалар өңдеуіштерінің object (C# тілінде object класы кез келген класқа базалық класс болып келеді) типіндегі sender бірінші формалды параметрі хабарлама жіберуші болып табылады (хабарлама жіберуші).

Оқиғалар өңдеуішінің үлгілеріндегі екінші формалды параметр объект типіндегі айнымалы (XXXEventArgs класының объектіне сілтеме) болып келеді. Мысалы, тышқан өңдеуіші үшін тышқан меңзерінің e.X e.Y координаттарын алуға болады.

XXXEventArgs класы (XXX – оқиға атауы) барлық стандартты оқиғаларда EventArgs жүйелі класының мұрагері болып табылады.

«Жариялау-жазылу» үлгісін сипаттағанда хабарларды алушылар (receivers) осы хабарларды алуға жазылатынын ескерткен болатынмыз. Яғни басқару элементтері мен форма үшін хабарларды көрсету (жазу) керек. Оқиғалар өңдеуіштерінің әдістері бірыңғай жазылу пішімінде болғандықтан, оларды көпадресті делегаттар – функциялардың кез келген санын көрсете алатын делегаттар арқылы біріктіруге болады. Form1.Designer.cs файлында, инициализация бөлімінде әрбір элемент үшін қасиеттерді анықтаумен қатар, оқиғалардың тізімі анықталады. Мысалы, оқиғалар тізімі анықталған форманың инициализациялау үзіндісі келесі түрде болады:

```
    private void InitializeComponent()
    {
     . . .
    this.MouseDoubleClick += new
System.Windows.Forms.MouseEventHandler
     (this.Form1_MouseDoubleClick);
    this.Paint += new
System.Windows.Forms.PaintEventHandler(this.Form1_Paint);
    this.MouseClick += new
System.Windows.Forms.MouseEventHandler
     (this.Form1_MouseClick);
    this.MouseMove += new
System.Windows.Forms.MouseEventHandler
    (this.Form1_MouseMove);
         . . ..
    }
```

Екскерту, барлық визуалды басқару элементтер Control класынан таралған.

28.3 Кластардың стандартты оқиғаларын қолдану мысалы

Кластардың «стандартты» оқиғаларын қолданатын бағдарламаны қарастырайық

Қосымшада формадағы тышқанның сол жақ батырмасын бір рет басу бойынша пайда болатын оқиға өңдеуіші қолданылған, сонымен қатар қосымшада меңзер тек форманың сол жақ бөлігінде орналасқан жағдайда ғана тышқан батырмасын басу оқиғасына жауап қайтарады:

```
    private void Form1_MouseClick(object sender,
MouseEventArgs e)
     {
    if ((e.X < 20 && e.X > 0 && e.Y < 20 & e.Y > 0)
&& p == 0)
```

Осы тәсілді бағдарламашылар экранның белгілі орнында тышқанның шертілуіне жауап қайтару үшін жиі қолданады. Мысалы, қала картасында, топтағы студентердің фотосуреттерінде – форманың белгілі бөлігіне тышқан меңзерін қойған кезде керекті түсініктемелерді көрсетуге болады.

Форма бойымен тышқаннның орын ауыстыру оқиғасының өңдеуіші тышқан меңзерінің орнын 100 нүктеден тұратын массивке жаза алады (қосымша шарт p==1).

Графикалық режімде массивті қарап шығу тышқанның сол жақ батырмасын екі рет басу нәтижесінде орындалады.

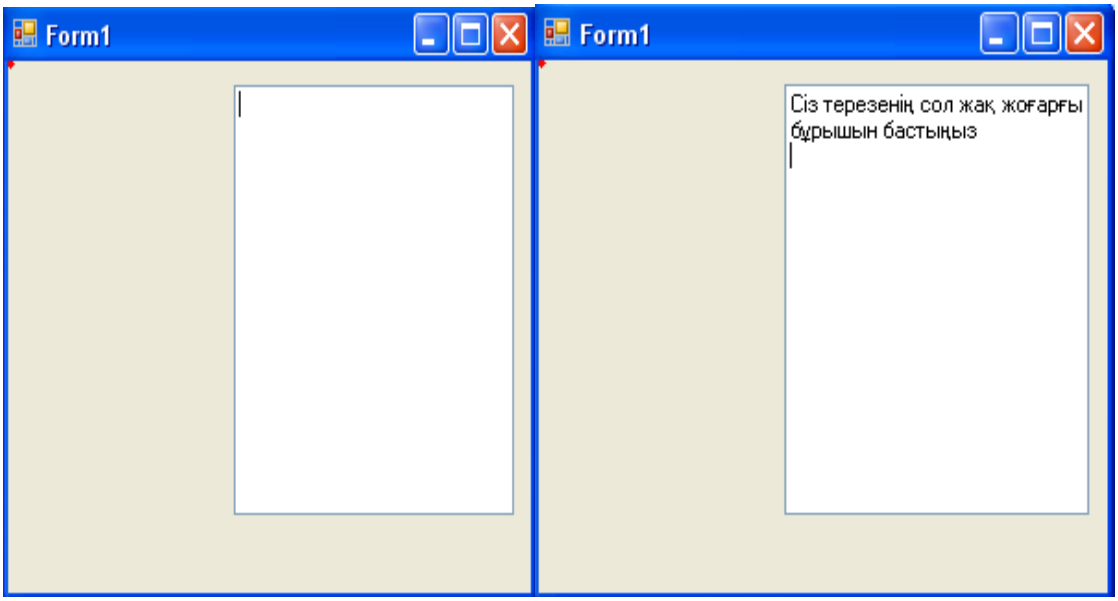
Батырманы басу өңдеуішінің көмегімен редактор терезесінде бағдарламаның алғашқы нұсқауы жүзеге асады.

Form1.cs файлының коды:

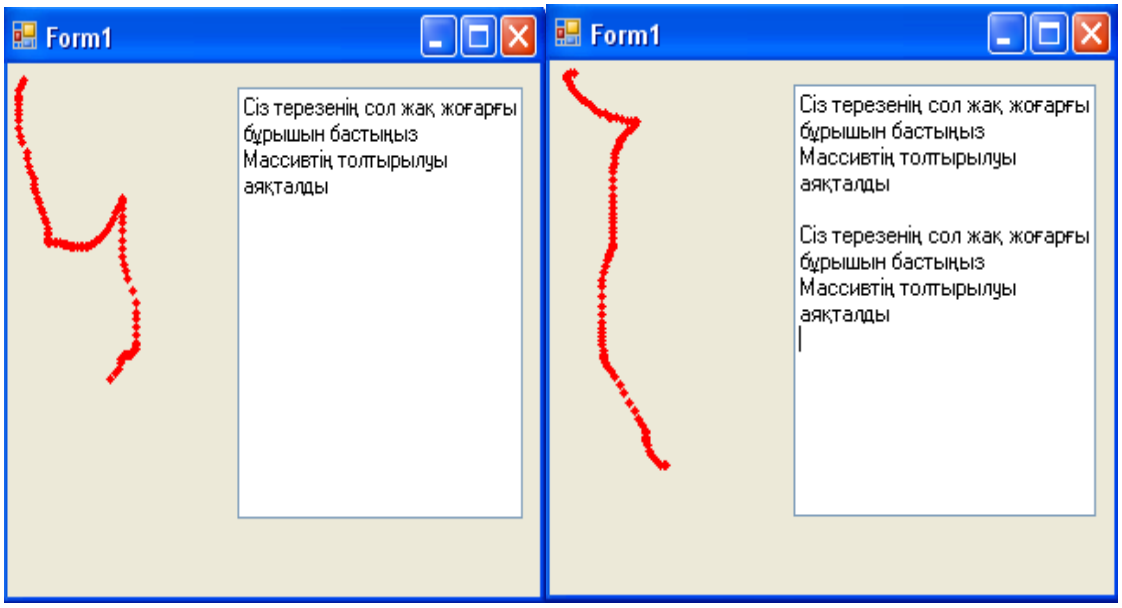
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsApplication1
{
 public partial class Form1 : Form
 {
 public int p,i,j;
 int[,] tk = new int[100, 2];
 public Form1()
 {
 InitializeComponent();
 textBox1.Text = "";
 p = 0; i = 0; j = 0;
```

```
      }
      private void Form1_MouseClick(object sender,
MouseEventArgs e)
      {
      if ((e.X < 20 && e.X > 0 && e.Y < 20 & e.Y > 0) && p ==
0)
      {
      textBox1.AppendText("Сіз терезенің сол жақ жоғарғы
бұрышын бастыңыз \r\n");
      p++;
      }
      }
      private void Form1_MouseMove(object sender,
MouseEventArgs e)
      {
      if (p == 1)
      if (i < 100) { tk[i, 0] = e.X; tk[i, 1] = e.Y; i++; }
      else
      {
      textBox1.AppendText("Массивтің толтырылуы аяқталды
\r\n");
      p++;
      }
      }
      private void Form1_MouseDoubleClick(object sender,
     MouseEventArgs e)
      {
      this.Invalidate();
      }
      private void Form1_Paint(object sender, PaintEventArgs e)
      {
      Pen myPen = new Pen(Color.Red, 2);
      Graphics g = e.Graphics;
      for (int n = 0; n < 100; n++)
      g.DrawEllipse(myPen, tk[n, 0], tk[n, 1], 2, 2);
      }
      private void textBox1_KeyDown(object sender, KeyEventArgs
e)
      {
      p = 0; i = 0;
      }
      }
     }
```

Қосымша жұмысы 28.1 және 28.2 суреттерінде көрсетілген.



28.1-сурет – Қосымшаны іске қосу және массивті толтыру



28.2-сурет – Массивті көрсету, қосымшаны босату және оны қайтадан іске қосу

28.4 Кластардың стандартты емес оқиғалары

Пайдаланушы әрекеттеріне қосымшаның жауап қайтаруы ретіндегі стандартты оқиғалармен қатар Windows-қосымшаның ішінде арнайы әдістер арқылы оқиғалар құрылуы мүмкін.

Қосымшаның өзі құратын оқиғалар түрлі объекттер арасында динамикалық байланысты ұйымдастыруға мүмкіндік береді, мысалы, дүкенде белгілі бір тауардың сатылуы көптеген құжаттардың ішіндегісін

автоматты түрде өзгертуі ерек (Сатудан түсетін жалпы кіріс, нақты тауарды сату бойынша құжат, осы тауардың қоймада болуы, т.б.).

Сонымен, хабарды шығару көзінен пайда болған оқиға туралы қосымшаның кейбір объекттеріне хабар беру қажеттілігі туындайды.

Оқиға көзі және оқиғану алушы (кейде оларды клиент деп атайды) арасындағы өзара әрекеттерінің механизмі делегатты пайдалануға негізделген.

Қосымшада делегат данасы жарияланады, ол оқиғалар өңдеуіштерінің стандартты әдістеріне сәйкес болып келеді.

Одан кейін оқиға көзі (sender) болып келетін класты анықтау керек және сол жерде оқиғаны сипаттайтын әдісті, оқиғаны іске асыратын (инициирующий) әдісті анықтау керек.

Қосымшаның жұмысы барысында оқиғалар өңдеуіштерінің объекттерін (клиенттері) делегат арқылы өңделетін әдістер тізіміне қосу керек. Осы процесс оқиғалар өңдеуіштерін тіркеу деп аталады.

Оқиға пайда болғанда барлық тіркелген әдістер делегат арқылы кезек бойынша орындалуға шақырылады.

Оқиға көзі мен пайдаланушы арасындағы өзара әрекеттесу механизмінің жұмысы келесі мысалда қарастырылған.

28.1-есеп. Минус 5-тен 10-ға дейінгі аралықта болатын 10 кездейсоқ бүтін сандардан тұратын массивті құру керек. Теріс сандарды қолдануға болмайды деп жорамалдайық және олар үшін оқиғаларды құрайық. массивті экранға басып шығару, оның қосындысын және мәндердің өзгеру графигінің суретін салу керек.

Егер массив мәні теріс сан болатын болса, онда оқиғаны құру керек және осы оқиғаға екі өңдеуіш жауап беруі керек.

Біріншісі теріс санның таңбасын ауыстыру керек, екіншісі – массив элементтерінің жаңа мәндеріне сәйкес сандардың қосындысын өзгерту керек.

Көрнекілік үшін массивтің өзгерген мәндерінің графигі шығарылсын.

Сонымен, оқиғаны құру және қолдану үшін алдымен делегатты жариялау керек (делегат арқылы клиент пен оқиға көзі арасында байланыс орнайды).

.NET кітапханасында стандартты делегаттардың көптеген түрі сипатталған, олар оқиғаларды өңдеу механизмін жүзеге асыру үшін арналған. Осы кластардың көпшілігі бір ереже бойынша жазылған:

– делегат атауы оқиға атауымен басталып EventHandler жұрнағымен аяқталады;

– делегаттың екі формалды параметрі бар. Бірінші параметр оқиға көзін анықтайды және object типінде болады. Екінші параметр оқиға аргументін анықтайды және EventArgs типінде немесе одан туындайтын типте болады.

Делегатты жариялағанда осы ережелерді ұстанған жөн, мысалы:

```
    public delegate void ZamenaEventHandler(object
sender, ZamenaEventArgs arge);
```

Осы мысалда біз Zamena оқиғасын өңдеу үшін делегатты құрамыз, оқиға объектте – оқиға көзінде пайда болатын өзгерістермен байланысты. Делегат сипаттамасында екі аргумент көрсетілген: оқиғаны туындатқан sender объекті және оқиғаға байланысты параметрлері бар ZamenaEventArgs типіндегі arge объекті.

Бізге массив индексін ғана беру керек болғандықтан, ZamenaEventArgs класын мына тәртіпте анықтау дұрыс болады:

```
public class ZamenaEventArgs : EventArgs
 {
 public int item;
 }
```

Оқиғаларды өңдеу механизмін жүзеге асырудың келесі кезеңі - оқиға көзі (sender) болатын класты анықтау. Осы әдіс арнайы жазылу пішімінде болады және көбінесе оған сәйкес делегаттың жазылу пішімімен анықталады. Әдіске қол жеткізу спецификаторын анықтағаннан кейін (әдетте public) event қызметтік сөзі жазылады, одан кейін делегат анықтайтын тип және оқиға атауы көрсетіледі.

```
public event ZamenaEventHandler Zamena;
```

Кластың толық сипаттмасы мана түрде болады:

```
class sobit
 {
 public event ZamenaEventHandler Zamena;
 public void prov(ZamenaEventArgs arge)
 {
  if (masi[arge.item] < 0)
  {
  Zamena(this,arge);
  }
 }
 }
```

Оқиғаларды өңдеу механизмін жүзеге асырудың келесі кезеңінде оқиғаларды қабылдаушы кластарды анықтау керек. Осы кластардың ерекшелігі – оларда оқиғалар өңдеуіштерінің әдістері болуы керек. Әдістердің жазылу пішімі сәйкес делегаттың жазылу пішіміне сай болуы керек. Мысалы:

```
 class zam1
 {
 public void OnZam1(object sender,
ZamenaEventArgs e)
 {
  masi[e.item] = masi[e.item] * (-1);
```

```
 }
 }
```

Оқиғалар өңдеуіштерін тіркеу оқиғалар өңдеуіштерінің кластарының (кластар емес) объекттері үшін жүргізу керек. Ол тек қана қосымшаның жұмысы кезінде ғана мүмкін – объекттер (айнымалылыр) қосымшаның жұмысы кезінде ғана құрылады, мысалы, форманың инициализациясы барысында:

```
public Form1()
 {
 InitializeComponent();
 zam1 z1 = new zam1();
 zam2 z2 = new zam2();
 so.Zamena += z1.OnZam1;
 so.Zamena += z2.OnZam2;
 }
```

Қосымша жұмысын көрсету үшін екі батырма қолданылады – «Массивті құрк» және «Массивті тексеру және оқиғаларды іске қосу».

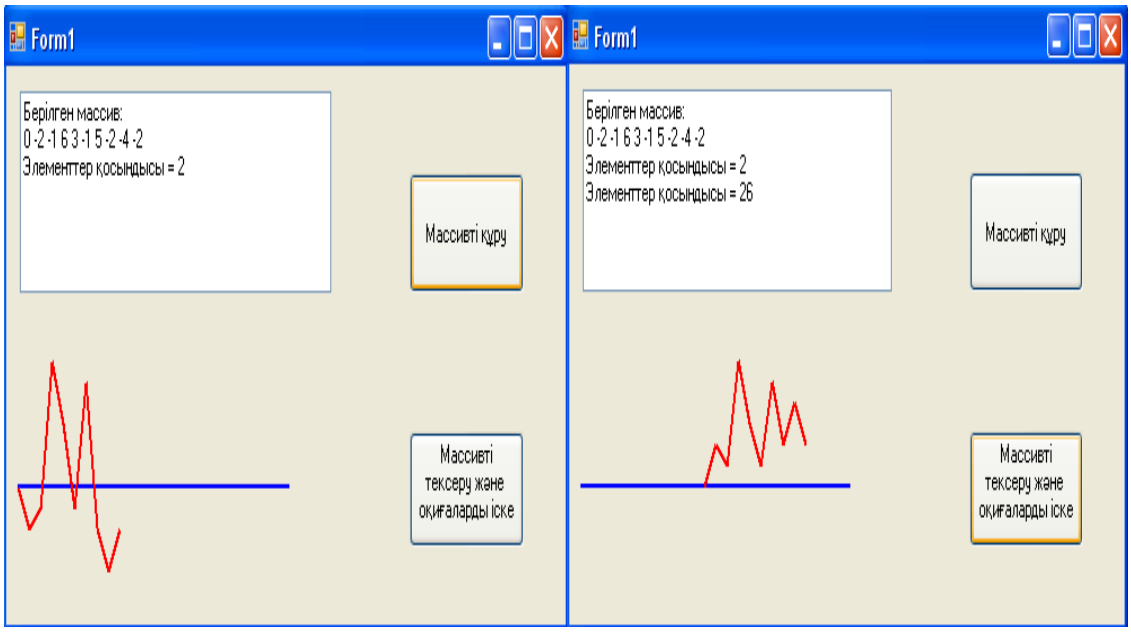
Бағдарлама коды:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{

 public partial class Form1 : Form
 {
 public delegate void ZamenaEventHandler(object sender,
ZamenaEventArgs arge);
 public int cob=0;
 public static int sum = 0;
 public static int[] masi = new int[10];
 public class ZamenaEventArgs : EventArgs
 {
  public int item;

 }
 class sobit
 {
  public event ZamenaEventHandler Zamena;
  public void prov(ZamenaEventArgs arge)
  {
   if (masi[arge.item] < 0)
   {
   Zamena(this,arge);
   }
```

```
        }
       }
       sobit so = new sobit();
       class zam1
       {
        public void OnZam1(object sender, ZamenaEventArgs e)
        {
        masi[e.item] = masi[e.item] * (-1);
        }
       }
       class zam2
       {
        public void OnZam2(object sender, ZamenaEventArgs e)
        {
        //if (masi[e.item] > 0) //оқиғаларды өңдеу кезектілігін
тексеру
         sum = sum + 2 * masi[e.item];
        }
       }
       public Form1()
       {
        InitializeComponent();
        zam1 z1 = new zam1();
        zam2 z2 = new zam2();
        so.Zamena += z1.OnZam1;
        so.Zamena += z2.OnZam2;
       }
       private void button1_Click(object sender, EventArgs e)
       {
        cob = 0; sum = 0;
        string ss="";
        Random rnd = new Random();
        for(int i=0;i<10;i++)
        {
        masi[i] = rnd.Next() % 15 - 5;
        sum = sum + masi[i];
        ss = ss + masi[i].ToString() + " ";
        }
        textBox1.AppendText("Берілген массив: \r\n");
        textBox1.AppendText(ss + "\r\n");
        textBox1.AppendText("Элементтер қосындысы = " +
sum.ToString() + "\r\n");
        this.Invalidate();
       }
       private void button2_Click(object sender, EventArgs e)
       {
        cob=1;
        ZamenaEventArgs zz = new ZamenaEventArgs();
        for (int i = 0; i < 10; i++)
        {
        zz.item = i;
        so.prov(zz);
```

```
        }
        textBox1.AppendText("Элементтер қосындысы = " +
sum.ToString() + "\r\n");
        this.Invalidate();
       }
       private void Form1_Paint(object sender, PaintEventArgs e)
       { Graphics g = e.Graphics;
        g.DrawLine(new Pen(Brushes.Blue, 2), 10, 200, 250, 200);
        Pen myPen = new Pen(Color.Red, 2);
        if (cob == 0)
        {
        for (int i = 1; i < 10; i++)
         g.DrawLine(myPen, i * 10, (200 - masi[i - 1] * 10), (i
+ 1) * 10, (200 - masi[i] * 10));

        }
        if (cob == 1)
        { for (int i = 1; i < 10; i++)
         g.DrawLine(myPen, i * 10 + 110, (200 - masi[i - 1] *
10), (i + 1) * 10 + 110, (200 - masi[i] * 10));
        }
       }
       }
      }
```

Қосымшаның жұмысы 28.3-суретте көрсетілген.



28.3-сурет – Оқиғаларды өңдеу бойынша қосымша жұмысы

28.5 Өзін-өзі тексеру сұрақтары

1 Оқиға ұғымы.

2 Пайдаланушы әрекеттеріне арналған, басқару элементтерінің оқиғалар өңдеуіштерінің дайындамалары қай жерде орналасқан?

3 Кластың оқиғалары әдетте қалай аталады?

4 Формадағы батырманы екі рет шерткенде не орындалады?

5 Барлық хабарлама өңдеуіштерінің бірінші формалды параметрі нені анықтайды?

6 Хабарлама өңдеуішінің екінші формалды параметрі нені анықтайды?

7 Формада орналасқан басқару элементтерінің оқиғалары қай жерде көрсетіледі?

8 Кластың барлық оқиғаларын қалай біріктіруге болады?

9 Форманың көпадресті делегатында оқиғаларды қосу қай жерде орындалады?

10 Форманың көпадресті делегаты қай жерде жарияланады?

А қосымшасы

Өзін-өзі тексеруге арналған сұрақтардың жауаптары

C# ТІЛІНЕ КІРІСПЕ

1 Алгоритм ұғымы.

Алгоритм дегеніміз - ретімен орындалуы нәтижесінде ресурстарды мейілінше аз жұмсап есепті шешуге әкелетін бірқатар ережелердің саны.

2 Есептерді шешу алгоритмін құру кезеңдері неден тұрады?

Есеп анализі, есепті шешу алгоритмін әзірлеу және алгоритмді тестілеу.

3 Алгоритмді жүзеге асыру кезеңі неден тұрады?

Бағдарлама кодын бағдарламалау тілдерінің бірінде жазу және бағдарламаны тестілеу.

4 .NET платформасы не үшін арналған?

.NET платформасы дегеніміз - кез келген бағдарламалау тілінде жазылған қосымшаны әзірлеу мен орындау үшін арналған біртұтас жүйе.

5 Visual Studio.NET қосымшаларын дайындау ортасы дегеніміз ...

Бағдарламашылар бағдарламаларды жазу, түзету, машиналық кодқа түрлендіру, бағдарламаны дұрыстау және іске қосу үшін қолданатын құралдардың жиынтығы.

6 Құрастыру файлы неден тұрады?

CIL тілінде кодтан және метадеректерден

7 Түрлі тілдерде жазылған бағдарламаларды тасымалдау .NET платформасында қалай қамтамасыз етіледі?

CIL аралық тілі арқылы

8.NET платформасында .NET Framework неден тұрады?

Кластар кітапханасынан

9.NET платформасында CTS неден тұрады?

.NET платформасының барлық бағдарламалау тілдері үшін ортақ типтер жүйесінен.

10 C# тілінде айнымалы ұғымы.

Бағдарламаны орындау уақытының әр түрлі кезінде компьютер жадысының аймағы түрлі мәндерде болуы мүмкін.

C# ТІЛІНІҢ ҚАРАПАЙЫМ ОПЕРАТОРЛАРЫ

1 `using` қызметтік сөзі не үшін қолданылады?

Осы қызметтік сөз арқылы бағдарламаға белгілі бір атаулар кеңістігі қосылады.

2 C# тіліндегі бағдарламаның негізгі әдісі қалай аталады?

static void Main()

3 `Math.Pow(x,y)` функциясы нені есептейді?

Math.Pow(x,y) функциясы функциясы y дәрежедегі x есептейді.

А қосымшасының жалғасы

4 Компьютер пернетақтасынан x айнымалысының мәнін енгізу үшін қандай әдістерді қолдануға болады?

C# тілінде пернетақтадан деректерді енгізу үшін (интерактивтік немесе диалогтық режімде) консольді статикалық Console.ReadLine() және Console.Read() әдістері қолданылады.

5 Монитор экранына x айнымалысының мәнін шығару үшін қандай әдістерді қолдануға болады?

Консольді қосымшаларда монитор экранына айнымалылар мәндерін шығару үшін C# тілінде статикалық Console.WriteLine() және Console.Write() әдістері қолданылады.

6 Бағдарламада түсініктемелер қалай жазылады?

Бағдарламада түсініктемелерді // белгісі арқылы бір жолға, сонымен қатар /* . . . . */ белгісі арқылы бірнеше жолға жазуға болады.

7 `10 % 3` өрнегі неге тең?

10-ды 3-ке бөлудің қалдығы 1-ге тең.

8 `10 / 3` өрнегі неге тең?

Бұл бүтін санды бөлу операциясы, нәтижесі 3-ке тең.

9 Сандарды шығару пішімі қалай анықталады?

Сандық мәндерді шығару пішімі толтырғышта толтырғыш нөмерінен кейін қос нүкте символы арқылы анықталады.

10 Бағдарламада стандартты математикалық функцияларды қолданбас бұрын нені орындау керек?

Бағдарламада математикалық функцияларды қолдану үшін Math класының атауын көрсету керек, мысалы, y = Math.Sin(x); немесе z = Math.Asin(y);.

C# ТІЛІНІҢ КҮРДЕЛІ ОПЕРАТОРЛАРЫ

1 `if` операторында `0<A` және `B>0` шарттарын қалай дұрыс жазуға болады, `A B`-ға тең емес?

if (A>0 && B>0 && A!=B)

2 `A && (!B)` өрнегі неге тең?

Егер A айнымалысының мәні TRUE, ал B айнымалысының мәні FALSE болса, онда өрнектің мәні TRUE болады, қалған басқа жағдаларда өрнектің мәні FALSE мәніне тең.

3 `A==B` жазбасы нені білдіреді?

A және B айнымалыларын салыстыру операциясы орындалады.

4 Шартты өту операторында келесі шарттарды қандай жазба дұрыс көрсетеді:

`y` = $\sqrt{(x+z)}$ егер `z < x` және `x > 4`, әйтпесе `y` = $\sqrt{(x-z)}$?

if (x > z && x > 4) y = Math.Sqrt(x + z); else y = Math.Sqrt(x - z);

А қосымшасының жалғасы

5 C# тілінің күрделі операторларында `'{' . . . . '}'` символдары не үшін қолданылады?

Фигуралық жақшалардың көмегімен күрделі операторлардағы әрекеттің басы мен соңын көрсетеді.

6 Бағдарламада `for` циклін қашан қолдану орынды?

Циклдік операциялардың саны алдын-ала белгілі болған жағдайда.

7 Қандай оператордың көмегімен цикл жұмысын «уақытынан бұрын» аяқтауға болады?

break операторының көмегімен.

8 Қандай оператордың көмегімен цикл денесінің бөлігінен «өтіп кетуге» болады?

Continue операторының көмегімен.

9 Бағдарламаның келесі үзіндісі нені шығарады?

```
int k=5;
int i;
for (i = k; i <= 0; i --) k = k+1;
 Console.WriteLine("i = {0} k = {1} ", i, k);
```

5 және 5 - екі саны экранға шығады.

10 `for` циклі аяқталғаннан кейін `I` басқарушы айнымалының мәні қандай болады?

```
 int I;
  int k = 5;
  for (I = 0; I <= k; I++) k = k - 1;
  Console.WriteLine("I = {0} ", I);
```

Экранға 3 саны шығады.

ШАРТТЫ ЦИКЛ ОПЕРАТОРЛАРЫ

1 Қандай жағдайда бағдарламада `while` циклін қолдану орынды?

Циклдің аяқталу шарты анықталған болса

2 `while` циклі қандай типке жатады?

Алғы шартты циклге.

3 `do {…}while` циклі қандай типке жатады?

Соңғы шартты циклге.

4 "Өрнектің" қандай мәнінде мына циклдің орындалуы тоқтатылады?

```
while ("өрнек") { . . . } ?
```

Егер "өрнек" логикалық типте және «жалған» болса.

5 "Өрнектің" қандай мәнінде мына циклдің орындалуы тоқтатылады?

```
do { . . . } while ("өрнек")?
```

Егер "өрнек" логикалық типте және «жалған» болса.

А қосымшасының жалғасы

6 Бағдарламаның келесі үзіндісі нені шығарады?

```
k = 5; I = 0;
while (I < k)
{
I = I + 2;

k++;
}
Console.WriteLine("I = {0} k = {1} ",I,k);
```

Ол келесі мәндерді шығарады: I= 10 10

7 Цикл аяқталғаннан кейін I айнымалысының мәні қандай?

```
I = 0;
while (I != 10)
{
I++;
. . .
}
Console.WriteLine("I = {0} ", I);
```

Цикл аяқталғаннан кейін келесі мән экранға шығады: I= 10

8 Бағдарламаның келесі үзіндісі экранға нені шығарады?

```
I = 0;
while (I != 10)
for (I= 1;I<=9;I++) k++;
Console.WriteLine("I = {0}",I);
```

Экранға мына мән шығарылады: I= 10

9. Бағдарламаның келесі үзіндісі нені шығарады?

```
I = 0;
while (I != 10)
{
 j = 0;
 for (I = 1; I<=9; I++) j++;
}
Console.WriteLine("j = {0} ", j);
```

Экранға мына мән шығарылады: j = 10

10 Бағдарламаның келесі үзіндісі экранға нені шығарады?

```
j = 1; I = 0;
while (I < 5)
{
I++;
j = j*2;
}
Console.WriteLine("j = {0} ", j);
```

А қосымшасының жалғасы

Экранға мына мән шығарылады: j = 32

МАССИВТЕР

1 Мәнді типтегі айнымалы туралы түсінік.

Мәнді типтегі айнымалыға арналған жады бағдарлама компиляциясы барысында бөлінеді, онда айнымалының мәндері жазылады.

2 Сілтемелік типтегі айнымалы туралы түсінік.

Сілтемелік типтегі айнымалыға арналған жады бағдарламаның орындалуы барысында үйіндіде new операторы арқылы бөлінеді.

3 Массивте қандай айнымалыларды орналастыруға болады ?

Массивте кез келген типтегі біртипті айнымалыларды орналастыруға болады.

4 Бүтін типті айнымалылар үшін бір өлшемді массивті жариялаудың дұрыс нұсқасын қалай жазуға болады?

Мысалы, int[ ] masi;

5 Бүтін типті 10 айнымалы үшін бір өлшемді массивті инициализациялаудың дұрыс нұсқасын қалай жазуға болады?

Мысалы, masi = new int[10];

6 Бағдарламаның келесі үзіндісі нені орындайды:

```
for (I = 0; I <= 10; I++)
Console.Write(" {0} \t", a[I]);
```

Монитор экранына A массивінің мәндерін шығарылады (бір жолға)

7 `a[11]` массиві үшін бағдарламаның келесі үзіндісі нені орындайды?

```
for (I = 1; I <= 10; I++)
{k = a[I]; a[I] = a[11 - I]; a[11 - I] = k;}
```

Ештемені орындамайды.

8 Бағдарламаның келесі үзіндісі нені орындайды?

```
k = a[0];
for (I = 0;I <= 10; I++)
        if (a[I] > k) k = a[I];
Console.WriteLine(" {0} ", k);
```

Ең үлкен санды табады.

9 Бағдарламаның келесі үзіндісі нені орындайды?

```
for (I = 0; I <= 9; I++)
{ k = a[I]; a[I] = a[I+1]; a[I+1] = k;}
```

Массивтің барлық элементтері солға қарай бір орынға жылжытылады, ал бірінші элемент массивтің соңына жазылады.

10 Бағдарламаның келесі үзіндісі нені орындайды?

```
I = 0;
while (I <= 10)
{
```

A қосымшасының жалғасы

```
        a[I] = a[I] * (-1);
        I++;
}
```

Массивтің барлық элементтерінің таңбаларын ауыстырады.

МАССИВТЕРДІ ӨҢДЕУ АЛГОРИТМДЕРІ

1 Бағдарламаның келесі үзіндісінде сұрыптаудың қандай алгоритмі қолданылған:

```
for (I = 0; I < 10; I++)
for (j = I + 1; j <= 10;j++)
if (a[I] > a[j])
{b = a[I]; a[I] = a[j]; a[j] = b;} ?
```

«Таңдау» әдісін қолданып өсу тәртібі бойынша сандарды сұрыптау алгоритмі қолданылған.

2 Бағдарламаның келесі үзіндісінде сұрыптаудың қандай алгоритмі қолданылған:

```
 for (i=0; i<=9; i++)
 for (j=0; j<=9-i; j++)
 if (a[j]<a[j+1])
 { b=a[j];a[j]=a[j+1];a[j+1]=b;} ?
```

«Көпіршікті» әдісін қолданып, кему тәртібі бойынша сандарды сұрыптау алгоритмі қолданылған.

3 «Таңдау» әдісін қолданатын, сұрыптау алгоритмінің есептеу тиімділігі қандай болады?

Операциялар саны (N-1)+(N-2)+(N-3)+ . . . + 2 +1=N*(N-1)/2 тең.

4 Ең үлкен орташа іздеу уақыты қандай іздеу алгоритмінде жұмсалады?

Ең үлкен орташа іздеу уақыты тізбектеп іздеу алгоритмінде болады.

5 Массивте элементтерді блоктық іздеу алгоритмі туралы мағлұмат беру.

Массивтің барлық элементтері шартты түрде N жазба бойынша блоктарға бөлінеді. Іздеу әрбір блоктың соңғы жазбалары бойынша орындалады. Егер ізделінетін жазба блоктың кезекті соңғы жазбасынан кіші болса, онда тізбектеп іздеу осы блокта ғана орындалады.

6 Блоктық іздеу алгоритмінде орташа іздеу уақытын қалай азайтуға болады?

Тізімнің блогында жазба санын өзгерту

7 Кілт бойынша өсу ретімен сұрыпталған массив жазбаларын екілік іздеу алгоритмі туралы мағлұмат беру.

Ізделінетін жазба массивтегі «орташа» жазбасымен салыстырылады. Егер ізделінетін жазба үлкен болса, онда салыстыру массивтің бірінші жартысының «орташа» жазбасымен жүргізіледі, т.б. Егер кіші болса, онда

А қосымшасының жалғасы

салыстыру массивтің екінші жартысының «орташа» жазбасымен жүргізіледі, т.б.

8 Іздестіру массивінің «кілті» дегеніміз не?

Іздестіру массиві реттелетін жазба өрісі.

9 Хеш-функцияның маңызы қандай?

Хеш-функция ізделіп жатқан жазбаның «кілтін» сандық мәнге – Хеш-кестенің индексіне түрлендіреді.

10 Хештеу кезінде коллизияларды болдырмаудың қандай нұсқасы жиі қолданылады?

Хеш-кестенің әрбір ұяшығы үшін қарапайым тізімдерді қолдану.

КЛАСС ӘДІСТЕРІ

1 C# тілінде типі анықталған функция қалай аяқталуы керек?

return операторы арқылы.

2 C# тілінде функцияның қандай формалды параметрлері сілтемелік параметрлер деп аталады?

Алдында ref қызметтік сөзі тұратын формалды параметрлер.

3 C# тілінде алдында `out` қызметтік сөзі жазылатын функцияның формалды параметрлері қалай аталады?

Шығу параметрлері деп аталады.

4 Егер функцияның қайтарылатын мәні `double` типінде болса, бағдарламада функцияны қалай қолдану керек?

Оны нақты типтегі айнымалыға меншіктеу немесе өрнекте қолдану керек.

5 Функция ішінде басқа функцияны жариялауға бола ма?

Жоқ.

6 Қандай функцияны рекурсивті функция деп атайды?

Өзін-өзі тікелей немесе жанама түрде шақыратын болса.

7 Атаулары бірдей бірнеше әдісті анықтау процесі қалай аталады?

Әдістерді қайта жүктеу.

8 Кейбір әдістердің алдында `static` модификаторы неге қолданылады?

`Static` модификаторы класс объектісін құрмай-ақ әдістерді қолдануға мүмкіндік береді.

9 Келесі бағдарлама үзіндісі экранға нені шығарады:

```
public static void ttt(int a, int b, out int x, out int y)
 { if (a>b) {x = a; y = b;}else {x = b; y = a;} }
static void Main(string[] args)
 {
 int a, b, c = 0, d = 0;
 a = 5;
 b = 8;
```

А қосымшасының жалғасы

```
ttt(a, b, out c,out d);
 Console.WriteLine("c = {0} d = {1} ", c,d);
     }
```

Экранға келесі мәндер шығады: c = 8 d = 5

10 Келесі бағдарлама үзіндісі экранға нені шығарады:

```
public static void ttt(int a, int b, int x, int y)
     { x = a; a = b; y = a; }
static void Main(string[] args)
 { int a, b, c = 0, d = 0;
 a = 5;
 b = 8;
 ttt(a, b, c, d);
 Console.WriteLine(" {0} {1} ", c,d);
     }
```

Экранға келесі мәндер шығады: 0 0

ЖОЛДЫҚ АЙНЫМАЛЫЛАР

1 C# тілінде Unicode жазуы нені білдіреді?

Бұл Windows жүйесінде символдардың екі байтты кодтамасы.

2 C# тілінде escape-тізбегі не үшін қолданылады?

Монитор экранына ақпаратты шығару үшін ақпаратты пішімдеу.

3 Көлденең табуляцияны қандай escape-тізбегі орындайды?

\t тізбегі.

4 string st1 = new string(' ', 0); жазбасы нені білдіреді?

Нөлдік ұзындықтағы жолдық айнымалы жарияланады және инициализацияланады.

5 Жолдық айнымалылармен жұмыс жасағанда `Split` динамикалық әдісі не үшін қолданылады?

Жолдық айнымалының мәтінін элементтерге бөлуге мүмкіндік береді.

6 C# тілінде `str.Length` қасиеті нені орындайды? Мұндағы `str` – жолдық айнымалы.

Осы қасиет айнымалы жолында символдар санын анықтайды.

7 C# тілінде `string.Compare(str1, str2)` әдісі нені орындайды? Мұндағы `str1,str2` - жолдық айнымалылар.

str1 мен str2 жолдық айнымалылардың мәндерін салыстырады.

8 C# тілінде str1 = string.Concat(str2, str3); әдісі нені орындайды? Мұндағы `str1, str2, str3` – жолдық айнымалылар.

str2 айнымалысының мәніне str3 айнымалысының мәнін қосады және нәтиже str1айнымалысына жазылады.

А қосымшасының жалғасы

9 Бағдарламаның келесі үзіндісі нені шығарады:

```
    strimg clovo = "abcdefg";
```

```
Console.WriteLine(" {0}", clovo.Length);
```

Үзінді clovo жолдық айнымалысындағы символдар санын экранға шығарады, символдар саны 7-ге тең.

10 Бағдарламаның келесі үзіндісі нені шығарады?

```
char b;
int i, n;
string clovo = "abcadeafg";
b = clovo[0]; n = 0;
for (i = 0; i < clovo.Length; i++)
if (clovo[i] == b) n++;
Console.WriteLine(" {0}", n);
```

Үзінді clovo жолдық айнымалысындағы бірінші ‘a’ символының неше рет кездесетінің анықтайды (3 рет).

КӨП ӨЛШЕМДІ МАССИВТЕР

1 `Switch` операторы не үшін қолданылады?

Бағдарламаны ары қарай жалғастыру үшін бірнеше нұсқаның бірін таңдау мүмкіндігін ұсынады.

2 `Switch<өрнек>` операторында <өрнек> не үшін қолданылады?

Ол таңдау тұрақтысының мәнін анықтайды.

3 Неге `Switch` операторында case таңбасының (метка) әрекеті `break` операторымен аяқталуы керек?

Break операторы келесі таңбаға өтуге рұқсат етпейді.

4 a матрицасына неше сан жазуға болады, егер `int[,] a = new int[3,4];`?

Матрицаға 12 санды жазуға болады.

5 Бағдарламаның келесі үзіндісі нені орындайды?

```
for (int i = 0; i < 6; i++)
{
for (int j = 0; j < 6; j++)
Console.Write(ma[i, j]+"\t");
Console.WriteLine();
}
```

Монитор экранына матрица мәндерін жолдар бойынша шығарады.

6 Бағдарламаның келесі үзіндісі нені орындайды?

```
for ( i = 0; i < 9; i++)
for ( j = i+1; j < 10; j++)
{ b = a[i,j]; a[i,j] = a[j,i]; a[j,i] = b; }
```

A қосымшасының жалғасы

A матрицасын аударады.

7 Бағдарламаның келесі үзіндісі нені орындайды?

```
n = 0;
```

```
for (j = 0; j <= 10; j++)
for (I = 0; I <= 10; I++)
if (A[I, j] >= 0) n++;
```

Матрицаның оң сандарының санын анықтайды.

8 Бағдарламаның келесі үзіндісі нені орындайды?

```
for (I = 0; I <= 10; I++)
if (A[I,10-I] > 0) A[I,10-I] = 0;
```

А матрицасының берілген диагоналінің оң элементтері нөлге айналады.

9 Бағдарламаның келесі үзіндісі нені орындайды?

```
for (I = 0; I <= 10; I++)
for (j = 0; j < 10; j++)
for (k = j +1; k <= 10; k++)
if (a[I,j] > a[I,k])
{ b = a[I,j]; a[I,j] = a[I,k]; a[I,k] = b;} ?
```

Өсу реті бойынша а матрицасының сандарын жолдар бойынша сұрыптау орындалады.

10 Бағдарламаның келесі үзіндісі нені орындайды?

```
for (I = 0; I <= 10; I++)
if (A[I,I] > 0) A[I,I] = 0;
```

А матрицасының басты диагоналының оң элементтері нөлденеді.

ҚҰРЫЛЫМДАР

1 C# тілінде құрылымды жариялағанда қандай қызметтік сөз қолданылады?

Құрылымды хабарлаған кезде struct. қызметтік сөзі пайдаланылады.

2 struct типіндегі айнымалылар әдетте қалай аталады?

Struct типіндегі айнымалыны әдетте жазба деп атайды.

3 Жазбада қандай айнымалылар біріктіріле алады?

Жазбада кез келген типтегі айнымалылар біріктіріле алады.

4 Жазбаның ішінде айнымалылар қалай аталады?

Жазбаның ішіндегі айнымалылар әдетте өрістер деп аталады.

5 Құрылымның қандай мәндік және сілтемелік типі бар?

Мәндік типтегі деректерді ұйымдастырудың бір ғана күрделі формасы – құрылым болып келеді.

6 Жазба өрісінің атауы қалай анықталады?

Жазба атауы, нүкте символы және жазба өрісінен тұратын Құрамды атау қолданылады.

7 C# тілінде тізімдік типті жариялағанда қандай қызметтік сөз қолданылады?

Тізімдік типті жариялаған кезде enum қызметтік сөз қолданылады.

8 Тізімдік тип нені біріктіреді?

Тізімдік типте әдетте бір бірімен байланысты тұрақтылар біріктіріледі.

А қосымшасының жалғасы

9 C# тілінде құрылымдармен, кластармен жұмыс істегенде сериализацияны қалай түсінесіз?

Сериализация дегеніміз – объекттерді, жазбаларды файлға жазылатын байттар тізбегіне түрлендіру процессі.

10 C# тілінде құрылымдармен, кластармен жұмыс істегенде десериализацияны қалай түсінесіз?

Десериализация дегеніміз – файлдың байттар тізбегінен объекттерді немесе жазбаларды қалпына келтіру.

СТЕКТЕР, КЕЗЕКТЕР ЖӘНЕ ТІЗІМДЕР

1 Элементтерді қосу мен жою тізімнің бір жақ ұшында орындалады. Осы қарапайым тізім қалай аталады?

Стек.

2 Стектің басы қалай аталады?

Сыртқы стек.

3 Public Stack(ICollection n); жазбасы нені білдіреді?

ICollection типіндегі n элементтеріне арналған, стекті дайындайтын және инициализациялайтын конструктор.

4 Бағдарламаның келесі үзіндісі нені орындайды?

int n = (int) vst.Pop(); , мұнда `vst` – стек төбесі.

n элементін стектен шығарады.

5 Бағдарламаның келесі үзіндісі нені орындайды?

```
i = 0;
while (i < 5)
  { i++;
   n = rnd.Next() % 101 - 50;
   vstek.Push(n);

 }
```

мұнда `vstek` – стек төбесі.

Стекке кез келген 5 бүтін санды қосады.

6 Элементтерді қосу тізімнің бір жақ ұшында, ал элементі жою екінші жақ ұшында орындалатын қарапайым тізім қалай аталады?

Кезек.

7 `Public Queue()` жазбасы нені білдіреді?

10 элементке арналған кезекті дайындайтын конструктор.

8 `object queue.Peek()` жазбасы нені білдіреді?

Бұл - queue тақырыбынан объектті қайтаратын функция, бірақ функция объектті жоймайды.

9 Бағдарламаның келесі үзіндісі нені орындайды?

foreach (int i in ocer)

Console.Write(i + " ");

А қосымшасының жалғасы

Console.WriteLine(); , мұнда `ocer` – кезектің тақырыбы.
Кезектің ішіндегісін экранға, бір жолға шығарады.
10 Бағдарламаның келесі үзіндісі нені орындайды?
ocer.Clear(); , мұнда `ocer` – кезектің тақырыбы.
Кезектен барлық элементтерді алып тастайды.

ГРАФТАР

1 Қабырға арқылы байланысатын төбелерді қалай атайды?

Бір қабырға арқылы байланысатын төбелер сыбайлас төбелер деп аталыды.

2 Графтың жолы туралы ұғым.

U және K төбелерін байланыстыратын жол дегеніміз – W0,W1, . . . , Wn (n>0) төбелерінің тізбектілігі, мұнда W0=U, ал Wn=K және кез келген і (0<=i<=n-1) үшін Wi және Wi+1 төбелері қабырғалар арқылы байланысқан.

3 Графтың қандай жолы қарапайым жол деп аталады?

Егер жолдың барлық төбелері әр түрлі болса, онда ондай жол қарапайым жол деп аталады.

4 Егер кез келген төбеден басқа төбеге жол бар болса, онда ондай граф қалай аталады?

Егер графтың кез келген төбелер жұбын байланыстыратын жол бар болса, онда ондай граф байланысқан граф деп аталады.

5 Циклдері жоқ, байланыстаралған граф қалай аталады?

Циклсіз байланыстыралған граф бұтақтар деп аталады.

6 Графта оның барлық төбелерін өзіне қосатын цикл бар болса (бірақ барлық қабырғалары міндетті емес), онда осындай графты қалай атайды?

Егер графта оның барлық төбелерін өзіне қосатын цикл бар болса (бірақ барлық қабырғалары міндетті емес), онда ондай графты Гамильтон графы деп атайды.

7 Сыбайлас матрицада бағана мен жол қиылысында не жазылады?

Қиылыста төбелерді сипаттайтын мәндер сақталады.

8 Егер берілген төбеден графтың кез келген төбесіне жол бар болса, онда осы төбе қалай аталады?

Егер берілген төбеден графтың кез келген төбесіне жол бар болса, онда осы төбе граф көзі деп аталды.

9 Флойд алгоритмі нені анықтайды?

Флойд алгоритмі граф төбелерінің арасынадығы минимальді ара қашықтықтықты анықтайды.

10 Дейсктр алгоритмі нені анықтайды?

Дейсктр алгоритмі графта берілген екі төбе арасынадағы минимальді маршрутты табу үшін қолданылады.

А қосымшасының жалғасы

ГРАФТАРҒА АРНАЛҒАН АЛГОРИТМДЕР

1 Өз жұмысына стекті пайдаланатын графты жүріп өту алгоритмі қалай аталады?

Граф төбелерін «тереңдігі» бойынша жүріп өту.

2 Графты жүріп өту алгоритмінің стегінде не сақталады? Алгоритм өз жұмысында стекті пайдаланады?

Графты «тереңдігі» бойынша жүріп өту циклінің жұмысы барысында стекке тізімдердің тақырыбы жазылады.

3 Өз жұмысында кезекті пайдаланатын графты жүріп өту алгоритмі қалай аталады?

Граф төбелерін «көлденеңі» бойынша жүріп өту.

4 Графты жүріп өту алгоритм кезегі нені сақтайды (графты жүріп өту алгоритмі кезекті пайдаланады)?

кезекке өсу реті бойынша барлық жаңа сыбайлас төбелер орналастырылады (қаралып өткен төбелерге қосылған).

5 Графты «тереңдігі» бойынша жүріп өту алгоритмінің аяқталу шарты?

Стекте элементтер болмайды.

6 Графты «көлденең» бойынша жүріп өту алгоритмінің аяқталу шарты?

Кезекте элементтер болмайды.

7 Графтың барлық циклдарын іздеу алгоритмінде графтың қаралып кеткен төбелеріне жаңа төбе қасиеті неге «қайтарылады» ?

Бұл төбелерді басқа маршруттарда қолдану үшін орындалады.

8 Екі берілген төбе арасында барлық маршруттарды іздеу алгоритмінің стегінде не сақталады?

Стекте төбе туралы жазба сақталады, онда граф төбесінің нөмері, сыбайлас төбе тізімінің нөмері, төбенің тізімдегі позициясы жазылады.

9 Екі берілген төбе арасында барлық маршруттарды іздеу алгоритмінде `do_while` циклі неге қолданылады?

Оның негізгі қызметі – сыбайлас төбелер тізімінен жаңа төбені іздеу.

10 Екі берілген төбе арасында барлық маршруттарды іздеу алгоритмінде неге екі шартты `if операторы` қолданылады?

Бірінші `if операторында` алгоритмнің әрекеттері сипатталған - егер графтың табылған төбесі «жаңа» төбе болса, екінші `if` операторында алгоритмнің әрекеттері сипатталған - егер қаралып өтетін сыбайлас төбелердің тізімінде «жаңа» төбе болса.

ЕРЕКШЕ ЖАҒДАЙЛАРДЫ АЛДЫН АЛУ

1 Ерекше жағдай ұғымы.

А қосымшасының жалғасы

Бағдарлама жұмысының мерзімінен бұрын аяқталуына себеп болатын бағдарлама жұмысындағы жағдай (бағдарлама «тоқтайды» немесе үзіледі).

2 Ерекше жағдайды алдын алу ұғымы.

Ерекше жағдайды алдын алу – бағдарлама жұмысындағы қателіктер туралы ескертудің тәсілі.

3 Бағдарламада ерекше жағдайларды өңдеу механизмі (әдісі) неге негізделген?

Exception класының немесе оның ұрпағының арнайы объектісін құруға негізделген бағдарламада ерекше жағдайларды өңдеу механизмі.

4 Операциялық жүйеде қарастырылған ерекше жағдай бойынша стандартты өңдеуіш қалай жауап береді?

Орын алған ерекше жағдай бойынша сәйкес ақпаратты шығара отырып бағдарламаның орындалуын аяқтайды.

5 Бағдарламаның өзінде алдын ала қарастырылған ерекше жағдай бойынша өңдеуіш не үшін қолданылады?

Қателіктің салдарларын жою және дұрыс нәтижені алу үшін.

6 Бағдарламаны қорғалған блогы не үшін құрылады?

Онда ерекше жағдай орын алуы мүмкін бағдарламаның үзіндісі орналасады.

7 Бағдарламаның қорғалған блогы қандай қызметтік сөзден басталады?

Бағдарламаның қорғалған блогының алдына try кілттік сөзі жазылады.

8 Бағдарламада қарастырылған ерекше жағдай бойынша өңдеуіш ерекше жағдайға қалай жауап береді?

C# тілінде ерекше жағдайды өңдеу үшін арнайы catch-блоктар қарастырылған.

9 `Catch`-блоктар не үшін қолданылады?

Олар ерекше жағдайларға «дұрыс» жауап қайтаруға арналған.

10 Бір қорғалған блокқа қанша `catch`-блок сәйкес бола алады?

`catch`-блок өңдеуіштерінің кез келген саны болуы мүмкін (біде бірі болмауы мүмкін).

VISUAL STUDIO.NET ВИЗУАЛДЫ БАҒДАРЛАМАЛАУ ОРТАСЫ

1 Компьютер жұмысындағы оқиғалар ұғымы?

Оқиға дегеніміз – компьютер жұмысындағы кез келген «стандартты емес» жағдайдың пайда болуы, мысалы, пернетақтада батырманы басу, тышқан меңзерінің орын ауыстыруы, нөлге бөлу, т.б.

2 Windows жүйесі оқиғаларды қалай «ажыратады»?

Әрбір оқиғаның жеке нөмері – «үзу векторы» болады.

3 Windows жүйесі оқиғаның пайда болуы туралы ақпаратты алғанннан кейін не істейді?

А қосымшасының жалғасы

Оқиға нөмерін анықтағаннан кейін Windows жүйесі сәйкес драйверді «іске қосады».

4 Драйвер ұғымы.

Бұл - сәйкес оқиғаларға дұрыс әрекеттерді орындайтын арнайы бағдарлама.

5 Хабар ұғымы.

Жалпы алғанда хабарлар - жүйеде орын алып жатқан оқиғаларға Windows операциялық жүйесінің әрекеті (жауабы).

6 Хабар неден тұрады?

Windows хабарлары жазба болып келеді, онда қандай оқиғаның болғаны туралы және болған оқиға бойынша қосымша ақпарат (параметрлер) жазылады.

7 Windows жүйесі драйверден қабылдап алатын хабарларды қайда жібереді?

Windows жүйесі қабылдап алған барлық хабарлар хабарлардың жүйелі кезегіне орналастырылады. Одан кейін хабарлар жүйелі кезектен жеке Windows-қосымшалардың хабарлар кезегіне таратылады.

8 Әрбір қосымшада Windows-тан келетін хабарларды өңдеу циклі не үшін қолданылады?

Қосымша осы цикл арқылы «өзіне» арналған хабарларды теріп алады және оларды қосымшаның сәйкес хабарлар өңдеуіштеріне табыстайды.

9 Қосымшаның қандай әдісі Windows-тан келетін хабарларды өңдеу циклін жүзеге асырады?

Application.Run(new Form1());

10 Form класы қандай мақсатпен қолданылады?

Form класы қосымшаның пайдаланушыға арналған интерфейсін анықтайды.

БАСҚАРУ ЭЛЕМЕНТТЕРІ

1 .NET платформасының Windows.Forms.Designer не үшін қолданылады?

.NET платформасының Windows формаларына арналған конструктор немесе дизайнер. Ол интерактивті режімде қосымша формасын визуальді жобалауға мүмкіндік береді.

2 Форма класының сипаттамасында “partal” қызметтік сөзі нені білдіреді?

Бұл қызметтік сөз мынаны көрсетеді: берілген сипаттамада форманың класс сипаттамасына тиісті барлық кодтың тек бір бөлігі ғана орналасады.

3 Хабарламалар өңдеуіштері форма класына арналған сипаттаманың қандай бөлігінде орналасады?

А қосымшасының жалғасы

Хабарламалар өңдеуіштерінің сипаттамасы Form1.cs файлында орналасады.

4 Windows қосымшасының орындалуы қандай әдістен басталады?

Windows қосымшасының жұмысы Main() әдісінің орындалуымен басталады.

5 Бағдарлама іске қосылғанда қандай әдіс Form1 класының объектісін құрайды?

Бағдарлама іске қосылғанда Form1 класының объектісін құру мен инициализациялауды Application.Run() әдісі жүзеге асырады.

6 Label басқару элементінің қызметі?

Бұл басқару элементі мәліметтерді формаға шығару үшін арналған.

7 Label басқару элементінің қандай қасиеті форма терезесіне ақпаратты шығара алады?

Label басқару элементіне мәліметті шығару үшін Text қасиеті қолданылады.

8 Label басқару элементінің қандай қасиеті арқылы оның «мөлдірлігін» анықтауға болады?

Label элементінің «мөлдірлігі» BackColor қасиеті арқылы анықталады.

9 Батырманың қандай қасиеті арқылы оған суретті орналастыруға болады?

Батырмаға суретті Image қасиеті арқылы орналастыруға болады.

10 Диалогтық терезе жабылғанша қосымшамен ары қарай жұмыс жасауға кедергі келтіретін диалогтық терезені қалай атайды?

Осындай диалогтық терезе модальді диалогтық терезе деп аталады.

C# ТІЛІНІҢ ГРАФИКАЛЫҚ ИНТЕРФЕЙСІ

1 C# тілінде GDI+ нені білдіреді?

Бұл – қосымшаның графикалық интерфейсінің қысқаша белгілеу.

2 Brush класының объектісі нені анықтайды?

Brush класының объектісі геометриялық фигуралар ішінде кеңістік қалай және немен толтырылатынын анықтайды.

3 Pen класының объектісі нені анықтайды?

Pen класының объектісі сызықтардың түсі мен типін анықтайды, олар арқылы геометриялық фигуралардың контуры салынады.

4 Graphics класы нені анықтайды?

Ол монитор экранына мәтінді, кескінді, геометриялық фигураларды шығару үшін қолданылатын қасиеттер мен әдістердің жиынын анықтайды.

5 Windows жүйесінде қандай хабарлама терезе өлшемінің өзгеруін және терезе орынының ауысқанын «бақылайды»?

Терезелер өлшемінің өзгеруі мен орындарының ауысуын Windows жүйесінде WM_PAINT хабары «бақылайды».

А қосымшасының жалғасы

6 Қосымша формасының терезесін қайта салу бойынша өңдеуіш қалай аталады?

Форма терезесін қайта салуға байланысты өңдеуіш Form1_Paint деп аталады.

7 Form1_Paint өндеуішінің бірінші формалды параметр нені анықтайды?

Ол оқиғаны туындатқан объектке сілтемені береді.

8 Form1_Paint өндеуішінің екінші формалды параметрі нені анықтайды?

Ол арқылы PaintEventArgs класының объектісіне сілтеме беріледі.

9 Монитор құрылғысының контекст ұғымы.

Бұл – Windows жүйесінің арнайы бағдарламалары, олар қосымшаны компьютер бейнекартасының драйверімен байланыстырады.

10 Қандай әдіс Windows операциялық жүйесінен форма үшін WM_PAINT хабарын дайындауды талап етеді?

Бұл – this.Invalidate(); әдісі.

ҚОСЫМШАДА МЕНЮДІ ҚОЛДАНУ

1 Қосымшада меню не үшін құрылады?

Қосымшада меню қосымша жұмысының негізгі режімдерін ыңғайлы түрде қолдануға мүмкіндік береді.

2 Қосымшада меню командалары неге сәйкес болуы керек?

Қосымшадағы меню командалары қосымша жұмысының негізгі режімдеріне сәйкес болуы керек.

3 ToolBox терезесінің қандай басқару элементі қосымшада менюді құруға мүмкіндік береді?

Қосымшадағы менюді MenuStrip басқару элементі құруға мүмкіндік береді.

4 Меню редакторының қандай өрісі команданы енгізу үшін қолданылады?

Команданы енгізу үшін меню редакторында Type Here өрісін қолдану керек.

5 Меню редакторының қандай командасы арқылы менюге жаңа жолды жолдардың арасына қосуға мүмкіндік береді?

Менюге жаңа жолды жолдардың арасына Insert New командасы арқылы қосуға болады.

6 Қосымшада менюді құрған кезде пернетақта акселераторларын не себепті қолдану керек?

Пернетақта акселераторларын қолдану – командаларды тышқанмен таңдаудың балама жолы.

А қосымшасының жалғасы

7 Меню редакторының Insert Separator командасы не үшін қолданылады?

Меню редактордың осы командасы арқылы жолдар арасында бөлу сызығын қосуға болады.

8 RichTextBox және TextBox басқару элементтерінің арасында қандай ерекше айырмашылықтар бар?

RichTextBox элементі түрлі типтегі файлдармен жұмыс жасауға мүмкіндік береді.

9 Қосымшадағы менюдің негізгі командаларын қайталайтын қандай басқару элементі суреттерді сақтау үшін қолданылады?

Көптеген кескіндерді сақтау үшін ImageList басқару элементі қолданылады.

10 Инструменттік панель батырмасының қандай қасиеті әр батырманы түсіндірме терезесімен қамтамасыз етеді, түсіндірме терезесі батырмаға тышқан меңзерін «жақындатқанда» пайда болады?

Инструменттік панель батырмаларын түсіндірме терезесімен қамтамасыз етеу үшін ToolTipText қасиетін қолдану керек.

ДИАЛОГТЫҚ МЕНЮДІ ҚОЛДАНУ

1 Диалогтық категорияларға қандай басқару элементтері жатады?

Дискті, бумаларды, компьютер файлдарын қолдану үшін Windows жүйесінің диалогтық ресурстарын пайдалануға мүмкіндік беретін элемент.

2 Файлды ашу бойынша диалогтық терезені құру үшін қандай басқару элементі қолданылады?

Файлды ашу бойынша диалогтық терезені құру үшін OpenFileDialog басқару элементі қолдану керек.

3 Қандай әдіс файлды ашу үшін Windows жүйесінің стандартты диалогтық терезесін экранға шығарады?

Бұл – openFileDialog1.ShowDialog әдісі.

4 OpenFileDialog басқару элементі арқылы тек мәтіндік файлдарды ашу үшін осы элементті қалай күйге келтіруге болады?

OpenFileDialog элементінің Filter қасиетіне "Text files|*.txt" мәнін анықтау керек.

5 Файлды ашу бойынша диалогтық терезеде пайдаланушы «Ашу» батырмасын басса не болады?

OpenFileDialog элементінің ShowDialog әдісі DialogResult.OK мәнін қайтарады.

6 if (openFileDialog1.ShowDialog() == DialogResult.OK && openFileDialog1.FileName.Length > 0)

Жазбасында openFileDialog1.FileName.Length > 0 шарты нені білдіреді?

А қосымшасының жалғасы

Шарттың осы бөлігі мынаны білдіреді: «пайдаланушы таңдаған файлдың толық жолының ұзындығы» нөлден үлкен болуы керек.

7 RichTextBox1 басқару элементінің қандай әдісі файлды ашу үшін қолданылады?

richTextBox1.LoadFile әідісі.

8 SaveFileDialog басқару элементінің қызметі қандай?

Ол файлды сақтау бойынша диалогтық терезені құруға арналған.

9 SaveFileDialog басқару элементінің FileName қасиеті нені анықтайды?

Бұл қасиет сақталатын файл атауының үлгісін анықтауға мүмкіндік береді.

10 richTextBox1.SaveFile әдісінің RichTextBoxStreamType.PlainText параметрі нені анықтайды?

Құжат Text форматында сақталады.

КӨПТЕРЕЗЕЛІ ҚОСЫМШАЛАР

1 MDI қысқартуы нені білдіреді?

Көптерезелі интерфейсті.

2 Қандай жағдайларда батырмасы бар форманы жобалау керек?

Дайындалатын жоба қосымшасының әрбір режімі жеке интерфейсті қажет ететін түрлі күрделі сервистермен берілген жағдайда.

3 Формада суреттерді орналастыру үшін әдетте қандай басқару элементі қолданылады?

PictureBox басқару элементі.

4 Solution Explorer терезесі арқылы жобаға жаңа форманы қалай қосуға болады?

Жоба атауының (WindowsFormsAplication1) үстінен тышқанның оң жақ пернесін шерту керек және көрінетін менюден Add режімін, ондағы Add Windows Form командасын таңдау керек.

5 Project режімі арқылы жобаға жаңа форманы қалай қосуға болады?

Project режімін, оның ішінде Add Windows Form командасын таңдау және форманың атауын растап, Add батырмасын басу керек.

6 Диалогтық (модальді) терезенің форманың қарапайым терезесінен айырмашылығы қандай?

Диалогтық (модальді) терезе жабылғанша одан шығуға болмайды.

7 Форманың қарапайым (модальді емес) терезесі қандай әдіс арқылы ашылады?

Show() әдісі арқылы.

8 Форманың модальді терезесі қандай әдіс арқылы ашылады?

ShowDialog() әдісі арқылы.

9 Бағдарламаның келесі үзіндісі нені орындайды:

А қосымшасының жалғасы

```
private void button3_Click(object sender, EventArgs e)
 {
        Form5 f5 = new Form5();
        if (f5.ShowDialog() == DialogResult.OK) k = 0;
 } ?
```

Модальді диалогтық терезе түрінде ашылатын жаңа форма құрылады.

10 Мәліметтерді кестеде шығару үшін қандай басқару элементі жиі қолданылады?

DataGridView басқару элементі.

КЛАСС ҰҒЫМЫ

1 Класс ұғымы.

Класс дегеніміз - өрістерден, әдістерден, оқиғалардан тұратын деректер типі.

2 Класс қасиеттері туралы ұғым?

Бұл - класс өрістерінің мәндерін бағдарламаның басқа кластарымен алмасуға (оқуға немесе жазуға) мүмкіндік беретін әдістер жинағы.

3 Класс конструкторы туралы ұғым?

Бұл – класс объекттерін құруға және класс өрістеріне бастапқы мәндерді меншіктеуге арналған кластың арнайы әдістері.

4 Класс деструкторы туралы ұғым?

Бұл – объектке бөлінген ресурстарды босату кезінде әрекеттердің ретін анықтайтын арнайы әдістер.

5 Класс оқиғасы туралы ұғым?

Бұл – пайдаланушы әрекеттеріне немесе бағдарламадағы белгілі бір өзгерістерге жауап беруіне мүмкіндік беретін кластың арнайы әдістері.

6 Класс индексаторы туралы ұғым?

Класс деректерінің элементтеріне (әдетте массивтерге) реттік нөмері бойынша қол жеткізу құралы.

7 Объекттің this өрісінің міндеті?

Бұл – ол класс әдістеріне ағымдағы объект өрістерімен жұмыс жасауға мүмкіндік беретін ағымдағы объект адресіне сілтеме.

8 Класс типіндегі айнымалыны қалай аталады?

Класс типіндегі айнымалы әдетте объект деп аталады.

9 Класс сипаттамасында static қызметтік сөзі нені білдіреді ?

Бұл нұсқау мынаны көрсетеді: кластың айнымалысын (класс объектісін) құрмай-ақ бағдарлама класты және оның элементтерін қолдана алады.

10 Класс деректерінің сипаттамасында public қызметтік сөзі нені білдіреді?

А қосымшасының жалғасы

Бағдарламада қолжетімді болатын деректердің сипаттамасының басталуын білдіреді.

КЛАСС ЭЛЕМЕНТТЕРІ

1 Параметрсіз конструктор не үшін қолданылады?

Ол объекті құрайды және объект өрістеріне кейбір белгіленген мәндерді меншіктейді.

2 Параметрлері берілген конструктор не үшін қолданылады?

Ол объектті құрады және объект өрістеріне мәндерді меншіктейді.

3 Атаулары бірдей болатын әдістерді анықтау процессін қалай атайды?

Әдістерді қайта жүктеу.

4 Бір кластың бірнеше конструкторлары қалай аталады?

Жиынтық конструкторлар.

5 tka класының деструкторы қашан шақырылады?

Үйіндіден объект жойылған кезде ол қоқыс жинаушысының көмегімен автоматты түрде шақырылады.

6 Егер қайтарылатын мәнінің типі void болып жарияланса, әдіс қалай аталады?

Процедура.

7 C# тілінде типі void емес әдіс қалай аяқталуы тиіс?

Әдістің жұмысы return операторының орындалуымен аяқталуы тиіс.

8 Әдістің қандай формалды параметрлері сілтемелік параметрлер деп аталады?

Алдында ref сөзі тұратын формалды параметрлер.

9 Өрістер мен оларды өңдеу әдістерінің бір құрылымда бірігуі қалай аталады?

Инкапсуляция.

10 Кластың жабық өрістеріне қол жеткізу үшін кластарда қандай механизм қолданылады.

Қасиеттер.

ОББ ПРИНЦИПТЕРІ

1 Бір объект басқа объектке өз жағдайының өзгергені туралы қалай хабарлай алады?

Оқиғалар механизмінің көмегімен.

2 Өз жағдайының өзгергені туралы басқа объектке хабарлайтын объект қалай аталады?

Оқиғаның көзі.

3 Кейбір оқиғалардың «бос» өңдеуіші қалай құрылады?

Оқиғалар бөлімінде қасиеттер терезесінде, сәйкес жолды екі рет шерту керек.

А қосымшасының жалғасы

4 Формада button1 батырмасын екі рет шерткенде қандай оқиға өңдеуіші құрылады?

private void button1_Click(object sender, EventArgs e)

5 Кластарда операциялардың қайта жүктелуі не үшін қолданылады?

Ол әдеттегі математикалық өрнектерде класс типіндегі айнымалыны қолдануға мүмкіндік береді.

6 Мұрагерлік ұғымы?

Мұрагерлік дегеніміз – туынды кластың базалық кластың кейбір қасиеттерін, деректерін, әдістерін қолдануға мүмкіндігі.

7 Кластарды мұраланудың негізгі мақсаты неде?

Мұрагерліктің негізгі мақсаты құрылған кластарды қайта пайдалану болып келеді.

8 Қасиеттерін, деректерін, әдістерін басқа класс мұраланған болса, онда ондай класс қалай аталады?

Базалық класс.

9 Базалық класс қасиеттерін, деректерін, әдістерін мұраға алатын класты қалай атайды?

Туынды класс.

10 ОББ неге негізделген?

Инкапсуляция, мұрагерлік, полиморфизм принциптеріне.

ПОЛИМОРФИЗМ ПРИНЦИПІ

1 C# тілінде барлық кластардың иерархиялық тізбегінде қандай класс базалық класс болып табылады?

System.Object.

2 Туында класс конструкторын шақырған кезде базалық немесе туында кластардың қандай объектісі ерте (барлығынан бұрын) құрылады?

Базалық класс объектісі.

3 Мұрагерлік кластар тізбегінің артықшылықтары неде?

Тізбектегі соңғы класты мұралану тізбектегі барлық кластардың өрістерін, қасиеттерін, әдістерін иелене алуына мүмкіндік береді.

4 C# тілінде статикалық мұрагерлік нені білдіреді?

Объект арасында байланыстар бағдарлама компиляциясы барысында анықталатын мұрагерлік.

5 C# тілінде динамикалық мұрагерлік нені білдіреді?

Объект арасында байланыстар бағдарламаның орындалуы барысында анықталатын мұрагерлік.

6 C# тіліндегі полиморфизм ұғымы?

Полиморфизм дегеніміз – мұраланатын кластар тізбегіндегі атаулары бірдей әдістерді жүзеге асыру түрлерінің сан алуандылығы.

7 Полиморфизм механизмі қалай жүзеге асырылады?

А қосымшасының жалғасы

Полиморфизма механизмі базалық кластың атаулары бірдей әдістерін қалқалау есебінен жүзеге асырылады.

8 Бағдарламаның жұмысы барасында объект анықталған болса, онда оның әдісі қалай аталады?

Виртуальдық әдіс деп аталады.

9 C# тілінде абстрактлы базалық класс деп қандай класс аталады?

Объекттерін құрға мүмкін болмайтын класс.

10 Виртуальдық әдістер кестесі дегеніміз не?

Виртуальды әдістер және оларды ашу нүктесінің адресі сақталатын арнайы кесте.

ИНТЕРФЕЙСТЕРДІ ПАЙДАЛАНУ

1 Интерфейс ұғымы?

Интерфейс дегеніміз – барлық әдістері абстрактылы болатын абстрактылы класс.

2 Интерфейстік класты мұралану және қарапайым класты мұралану арасындағы айырмашылық неде?

Интерфейсті (интерфейстік класс) мұраланатын класс интерфейстің барлық әдістерін іске асыруы міндетті.

3 Интерфейстік класс және қарапайым класс арасындағы айырмашылық неде?

C# тілінде интерфейстік класс үшін көп мәрте мұралануға рұқсат берілген.

4 Интерфейстік кластарда атаулар арасында қарама-қайшылықтардың туындауы неліктен?

Атаулар арасында қарама-қайшылықтар көп мәрте мұралану барысында болуы мүмкін, өйткені әр түрлі аталық интерфейстерде синтаксисі бірдей аттас әдістер болуы мүмкін.

5 Бірнеше рет мұралану кезінде атаулар арасында болатын қарама-қайшылықты қалай шешуге болады?

Бірнеше рет мұралану кезінде атаулар арасында болатын қарама-қайшылықты шешудің бір жолы әдістерді біріктіру болып табылады.

6 Әдістерді біріктірудің мәнісі неде?

Интерфейстік кластағы барлық аттас әдістердің бағдарламалық жүзеге асырылуы бірдей болуы керек.

Интерфестік класта барлық аттас әдістер үшін бірдей жүзеге асыру бағдарламасы болуы қажет, ол өзінің барлық түпкі аттас әдістері үшін жалғыз деп жарияланады.

7 Әдістердің атауын өзгертудің мәнісі неде?

А қосымшасының жалғасы

Көп мәрте мұралану болған жағдайда атаулар арасында болатын қарама-қайшылықтарды шешудің бір жолы, сонымен қатар бірдей түпкі интерфейстер әдістері қайтадан аталады. Әр «жаңа» әдіс үшін өзінің жүзеге асырылуы жазылады.

8 Интерфейстердің артықшылықтары неде?

Интерфейстер класқа қосымша қасиеттерді ұсынады. Әрбір интерфейс класқа белгілі бір жаңа қасиетті береді.

9 Интерфейстер қандай типте болуы мүмкін?

Интерфейстерде тип болмайды.

10 Интерфейсті жариялағанда қандай қызметтік сөз қолданылады?

Интерфейсті жариялаған кезде класс атауының алдына interface қызметтік сөзін қолдану керек.

КЛАСТАРДЫҢ КОМПОЗИЦИЯСЫ ЖӘНЕ КОЛЛЕКЦИЯСЫ

1 Класс композициясы ұғымы.

Егер кейбір класс өз деректерінің өрістерінде басқа кластың объекттерін пайдаланатын болса, онда осындай кластардың бірігуі композиция деп аталады.

2 Класс композициясының негізгі мақсаты неде?

Класс композициясы бұрын жазылған бағдарлама үзіндісін қайта қолданудың тәсілі болып табылады.

3 Класс коллекцияның ұғымы.

Деректердің бір құрылымында біртипті объекттерді біріктіру кластар коллекциясы деп аталады.

4 Бір типтегі объекттерді коллекциялауда әдетте қандай біріктіретін құрылым қолданылады?

Коллекция деп аталатын класс.

5 Қандай коллекциялар тікелей қол жеткізу коллекцияларына жатады?

Элементтеріне элемент нөмірі арқылы қол жеткізуге болатын коллекциялар, мысалы, массивтер.

6 Қандай коллекциялар тізбекті қол жеткізу коллекцияларына жатады?

Элементтеріне қол жеткізу үшін тізбекті іздеуді орындауды қажет ететін коллекциялар.

7 Қандай коллекциялар иерархиялық коллекцияларға жатады?

иерархиялық коллекцияларда деректер түрлі «бұтақтар тәрізді» құрылым түрінде ұйымдастырылады, мысалы, бинарлық бұтақтар немесе иерархиялық жіктеу жүйесі түрінде.

8 Қандай типтегі объекттер ArrayList класының коллекциясына қосыла алады?

А қосымшасының жалғасы

ArrayList класының коллекциясы кез келген типтегі объекттерді сақтау үшін арналған.

9 ArrayList класының қандай қасиеті коллекцияның ағымдағы сиымдылығын сақтайды?

Бүтін типті Capacity қасиеті.

10 ArrayList класының қандай қасиеті коллекцияның ағымдағы ұзындығын сақтайды?

Бүтін типті Count қасиеті.

ДЕЛЕГАТТАР

1 Делегат ұғымы.

Делегат дегеніміз әдістерге сілтемелерді сақтау үшін арналған арнайы класс.

2 Қандай класс функционалды тип деп аталады?

Әрқайсысы функция болып келетін белгілі бір объекттер жиынтығын сипаттауға мүмкіндік беретін класс функционалды тип деп аталады.

3 Не нәрсе делегат типіндегі кластың данасы болып табылады?

Делегат типіндегі кластың данасы болып функцияларға (әдістерге) сілтемелер келеді, оларға айнымалыларға сияқты компьютер жадысында орын бөлінеді, бастапқы адресі – функцияға кіру «нүктелері» болады.

4 Делегатты жариялағанда қандай қызметтік сөз қолданылады?

Интерфейсті жариялағанда класс атауының алдына delegate қызметтік сөзін қолдану керек.

5 Делегатты шақыру параметрі ретінде қандай әдісті немесе функцияны қолдануға болады?

Сипаттамасы делегат сипаттамасына сәйкес келетін кез келген функция немесе әдіс делегатты шақыру параметрі ретінде қолданыла алады.

6 Делегатты шақыру параметрлеріне сілтемелер қай уақытта құрылады?

Делегатты шақыру параметрлеріне сілтеме бағдарлама жұмысы кезінде құрылады (динамикалық).

7 Делегаттардың типтері қай уақытта үйлесімді болады?

Делегаттардың форматтары үнемі ұқсас болса да, олардың типтері әрқашан үйлесімсіз болады.

8 Делегаттардың даналары қай уақытта үйлесімді болады?

Делегаттарды жариялау үшін бір ғана тип қолданылған болса, онда делегаттардың даналары үйлесімді (тең) болады.

9 Делегаттарды құрған кезде .NET платформасының қандай абстрактылы кластарын мұралауға болады?

А қосымшасының жалғасы

.NET платформасының стандартты типтер жүйесінде (CTS) `System.Delegate` және `System.MulticastDelegate` абстрактлы кластар бар, оларды делегаттарды құру кезінде базалық тип ретінде қолдануға болады.

10 Көпадресті делегат ұғымы.

Делегаттар үшін көпадресті делегат, яғни кез келген сандағы функцияларды көрсете алатын делегат деген ұғымы бар. C# тілінің барлық делегаттары System.MulticastDelegate-тен туынды класс болғандықтан, кез келген делегат көпадресті болуы ықтимал.

ОҚИҒАЛАР

1 Оқиға ұғымы.

Класс оиғалары дегеніміз – кластың пайдаланушы әрекеттеріне немесе бағдарламадағы белгілі өзгерістерге жауап беруге мүмкіндік беретін арнайы әдіс.

2 Пайдаланушы әрекеттеріне арналған, басқару элементтерінің оқиғалар өңдеуіштерінің үлгілері қай жерде орналасқан?

Пайдаланушының көптеген әрекеттері үшін оқиғалар өңдеуіштерінің үлгілері әзірленген, оларды Properties терезесінде, Events бетінде көруге болады.

3 Кластың оқиғалары әдетте қалай аталады?

Хабарлар өңдеуіші.

4 Формадағы батырманы екі рет шерткенде не орындалады?

Осы оқиғаның өңдеуіші құрылады:

private void button1_Click(object sender, EventArgs e).

5 Барлық хабарлама өңдеуіштерінің бірінші формалды параметрі нені анықтайды?

Ол хабарды жіберушіні (кейде хабар көзі деп аталады) анықтайды.

6 Хабарлама өңдеуішінің екінші формалды параметрі нені анықтайды?

Оқиға өңдеуішінің екінші формалды параметрі объект типіндегі айнымалы болып келеді, дәлірек айтқанда XXXEventArgs класының объектісіне сілтеме.

7 Формада орналасқан басқару элементтерінің оқиғалары қай жерде көрсетіледі?

Form1.Designer.cs файлында, инициализациялау бөлімінде әрбір элемент үшін басқару элементтерінің қасиеттерімен қатар оқиғалар тізбесі анықталады.

8 Кластың барлық оқиғаларын қалай біріктіруге болады?

А қосымшасының жалғасы

Оқиғалар өңдеуіштерінің барлық әдістерде бірыңғай жазба пішімі болғандықтан, оларды көпадресті делегаттардың көмегімен біріктіруге болады.

9 Форманың көпадресті делегатында оқиғаларды қосу қай жерде орындалады?

Ол меншіктеу операцияларының көмегімен InitializeComponent() әдісінде орындалады, мысалы,

this.MouseClick += new System.Windows.Forms.MouseEventHandler
(this.Form1_MouseClick);

10 Форманың көпадресті делегаты қай жерде жарияланады?

Form1.Designer.cs файлында ол еш әрекетсіз жағдайда тұрады.

ПӘНДІК КӨРПСЕТКІШ

C
CIL, 13
CLR, 13
CTS, 13

J
JIT, 13

N
.NET платформасы, 12
.NET Framework, 13
.NET технологиясы , 18

R
RTF, 216

Ә
әдістерді желімдеуді, 276
әдістерді қайта анықтау, 88
әдіс параметрлері, 82
  массивті, 82, 83
  сілтемелік параметрлер, 82 , 83
  шығыстық параметрлер, 82

Б
біріңғай аралық компиляция тілі, 13

В
виртуальды әдістер, 265
виртуальды әдістер кестесін, 266

Д
делегаттар, 297
  көпадресті , 301
десериализация, 128
деректер ағыны (stream), 128
деструкторлар, 249
динамикалық массив, 66

Е
ерекше жағдайлар, 175
ерекше жағдайларды алдын алу, 175
ерекше жағдайларды өңдеу, 176
ерекше жығдайды туындауы, 176

Ж
жиынтық конструкторлар, 246, 248

И
идентификатор, 17
индексатор, 238
инкапсуляция, 249
интерфейс,
IEnumerable, 279
көптерезелі, 223
интерфейсті класс, 275
итераторлар, 284

К
класс, 81, 236
  Exception, 177
  Random, 32
  деструкторы, 238
  конструкторы, 238
  қасиеттері, 238
  модификаторы, 24, 30
  оқиғалары, 238
  өрісі, 81, 236
  конструкторы, 239
компьютер жадына сақтау, 128, 137
конструкторлар, 246
  параметрсіз, 246

Қ
құрылымдар, 120
қызметтіксөздер, 16

abstract, 237
break, 45, 109
case, 111
class, 22
continue, 45
delegate, 297
else, 36
for, 42
foreach, 46
get, 249
goto, 40
if, 36
new, 18
out, 82
override, 266
private, 81
protected, 243
public, 24
ref, 82
return, 45
set, 249
static, 24
struct, 120
switch, 41
this, 250
throw, 176
virtual, 265, 266
void, 82
while, 47

М
массивтер, 58
динамикалық, 66
көпөлшемді, 108
сынық, 111
меню, 208,
мұрагерлік, 259
объектіге бағытталған бағдарламалау, 12, 81
оқиғалар, 181, 191, 303

П
полиморфизм, 256, 265
процедура, 240
псевдокездейсоқ сандар, 32, 108

Р
рекурсивті функция, 90
рекурсия, 89

С
сериализация, 128,

Т
таңба, 41
тізімдер, 134
тип, 2
char, 95
enum, 126
String,100
қарапайым, 18
мәндік, 17
сілтемелік, 17

Ш
шартты оператор, 41, 47

ӘДЕБИЕТТЕР ТІЗІМІ

1 В.В. Фаронов, Создание приложений с помощью C# Руководство программиста. - М.: “Эксмо”, 2008г.

2 Т.А. Павловская C#, Программирование на языке высокого уровня. Учебник для вузов, СПб,: Питер, 2009г.

3 А.В. Фролов, Г.В. Фролов Визуальное проектирование приложений C#. Книга вышла в издательстве Кудиц-Образ

4 Д. Кнут. Искусство программирования для ЭВМ. Т.1./ Основные алгоритмы / - М.:Мир,1976.

5 А.Л. Фукс Лекции по дисциплинам «Основы алгоритмизации» и «ООП» для студентов Томского государственного университета.