ЎЗБЕКИСТОН РЕСПУБЛИКАСИ ОЛИЙ ВА
ЎРТА МАХСУС ТАЪЛИМ ВАЗИРЛИГИ

ТОШКЕНТ АХБОРОТ ТЕХНОЛОГИЯЛАРИ УНИВЕРСИТЕТИ

М.С. Шарипов, Р.Я. Мамажанов,
Х.Ж. Раҳимбоев
К.А. Ибрагимова

DELPHI 7 ТИЛИДА ДАСТУРЛАШ АСОСЛАРИ

Касб-ҳунар коллежлари учун ўқув қўлланма

Тошкент – 2008

Олий ва ўрта махсус, касб-хунар таълими илмий бирлашмаларининг фаолиятини мувофиклаштирувчи Кенгаш томонидан нашрга тавсия этилган.

ТАТУ «Информатика ва КГ» кафедраси мудири, т.ф.д., проф. **Б.Ш.Раджабов** тахрири остида
М.С.Шарипов - ТАТУ Урганч филиали «Информатика ва АТ» кафедраси мудири т.ф. н., доцент
Р.Я.Мамажанов - ТАТУ «Информатика ва КГ» кафедраси, доценти т.ф.н.
Х.Ж.Рахимбоев - ТАТУ Урганч филиали «Информатика ва АТ» кафедраси ўкитувчиси
К.А. Ибрагимова – ТАТУ «Информатика ва КГ» кафедраси, ассистенти

Такризчи:

А.К.Каримов – Тошкент Юридик институти «Хукукий информатика» кафедраси мудири т.ф.н., доцент.

Сўнгги пайтларда ўкувчиларнинг дастурлашга бўлган кизикишлари ошиб бормокда. Бу ахборот коммуникация тизимларининг ривожланиши ва кундалик хаётимизда кенг кўлланилиши билан боғлик. Агар бирор киши компьютер билан иш кўрадиган бўлса, у холда эртами ёки кечми унда дастур яратиш истаги, баъзида зарурати пайдо булади.
Ҳозирги вақтда кўпчилик компютерларда Windows операцион тизими кўп кўланилади. Шунинг учун дастур яратувчилар ушбу операцион тизимида ишлайдиган дастурлар яратадилар.
Ушбу ўкув кўлланмада Delphi дастурлаш воситасида визуал дастурлаш технологиялари хакида ўкувчилар тўлик тасаввурга эга бўладиган маълумотлар берилган. Ўкув кўлланма академик лицей, коллеж ўкувчилари, олий техника ўкув юртлари талабалари, ўкитувчилари ва курсни мустакил урганувчилар учун мўлжалланган.

СЎЗ БОШИ

DELPHI – БУ НИМА?

Охирги йилларда дастурлашга бўлган қизикиш тоборо ортиб бормокда. Бу компьютер технологиясининг кун сайин ривожланиб бориши билан боғликдир. Айникса визуал дастурлаш технологияларидан фойдаланиб дастурлар яратиш компьютер технологиясининг ривожланишига катта таъсир этмокда.
Бир неча йил олдин оддий дастурловчилар Windows мухитида ишлайдиган дастур яратишни орзу қилардилар. Чунки бу вактда Windows учун дастур яратувчи восита сифатида фақат Borland C++ for Windows дастурлаш тили мавжуд бўлиб. бу тил етарлича билим ва малакага эга бўлган дастурчилар учун мўлжалланган эди.
Ҳисоблаш техникасининг жадал суръатлар билан ривожланиши ва дастурий таъминот яратишнинг унумли воситаларига бўлган талаб "тезкор яратувчи" деб аталувчи дастурлаш тизимлари пайдо бўлишига сабабчи бўлди. Бу тизим ичида Borland Delphi ва Microsoft Visual Basic дастурлаш тиллари хам мавжуд. Тезкор яратиш тизимлари (RAD –тизим, Rapaid Application Development – иловаларини тез яратиш мухити) асосида визуал лойиҳалаш ва ходисавий дастурлаш технологияси ётади. Дастурлашнинг бу технологияси дастурлаш жараёнидаги зерикарли ишларнинг катта кисмини ўз зиммасига олади ва натижада дастурловчига мулокат ойналари ва ходисалар учун функциялар яратиш ишлари колади. RAD–тизимидан фойдаланганда дастурловчининг иш унумдорлиги фантастик даражада бўлади.
Delphi-бу дастур яратишнинг тезкор мухити бўлиб, унда дастурлаш тили сифатида Delphi тили қўлланилади. Delphi тили қатъий типиклашган, объектга мўлжалланган тил бўлиб, унинг асосини дастурловчиларга маълум бўлган Object Pascal ташкил этади.
Ҳозирги кунда Delphi пакетининг янги версияларидан бири бу - Borland Delphi 7 Studio ҳисобланади. Ҳулди олдинги версияларидаги каби Borland Delphi 7 Studio мухитида хам турли дастурлар яратиш мумкин. Delphi мухитида оддий бир ойнали иловалардан тортиб то таксимланган маълумотлар базасини бошқарувчи иловаларгача яратиш мумкин. Delphi пакети таркибида маълумотлар базаси билан ишлашни таъминловчи, XML-ҳужжатлар, справка тизимини яратиш ва бошка масалаларни ечиш имконини таъминловчи утилитлар мавжуд. Delphi 7 ни асосий хусусияти NET технологияни қўллаб - кувватлашидир.
Borland Delphi 7 Studio пакети Windows 98 дан Windows XP гача бўлган барча операцион тизимларда ишлай олади. Delphi 7 ўрнатилиши учун компьютер ресурсларига қатъий талаблар қўйилмайди. Бунинг учун

компьютер процессори частотаси 166 МГц дан кам бўлмаган Pentium ёки Celeron типли бўлиши (Pentium II 400 МГц тавсия килинади) зарур. Тезкор хотирадан 128 Мбайт (тавсия килинади 256 Мбайт) хотира ва етарлича микдордаги диск бўш соҳаси (Enterprise версиясини тўлик ўрнатиш учун 475 Мбайт жой зарур бўлади) талаб этилади.

Дастурлашни ўрганиш учун маълум масалаларни ечиш дастурларини яратиш лозим. Дастурлашда эришиладиган муваффакиятлар маълум даражада малакага боғлик бўлади. Келтирилган мисолларни фақат ўқиш билан чекланмасдан, уларни компьютерга киритиб бажариш максадга мувофик ҳисобланади. Китобда келтирилган мисолларни албатта компьютерда бажариб кўринг. Вазифалар канчалик кўп даражада мустақил равишда бажарилса шунчалик кўп ўрганишингиз кафолатланади.

КИРИШ

Киришда Delphi тилини ўрнатиш жараёни кискача келтирилган. Келтирилган мисолда ютурувчи босиб ўтган масофадаги тезлиги хисобланади. Бу мисол оркали визуал лойихалаш ва ходисавий дастурлаш технологияси намойиш килинади. Яъни асосий тушунча ва атамалар ўрганилади.

DELPHI НИ ЎРНАТИШ

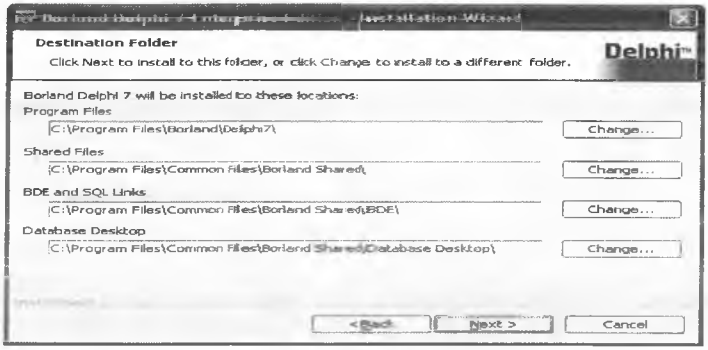
Borland Delphi 7 Studio пакетини тўртта варианти мавжуд: Personal, Professional, Enterprise ва Architect. Ушбу вариантларнинг барчасида турли вазифаларни бажарувчи, яъни юкори эффектли дастурлар яратишни таъминлаб берувчи стандарт воситалар мажмуи мавжуд. Шу билан бирга комплект даражаси канчалик юкори бўлса (Professional дан Architect га караб) дастурчи учун шунчалик кўп имкониятлар мавжуд бўлади. Масалан. Enterprise комплектида масофадаги маълумотлар базаси билан (масалан, Interbase) ишловчи иловалар яратиш имкониятлари мавжуд. Personal комплектида бундай имконият йўк. Borland Delphi 7 Studio пакетларининг тузилиши, таркиби ва имкониятлари хакидаги тўлик маълумотларини Borland Web–сайтидан (www.borland.com/delphi) олиш мумкин.

Ушбу китобдаги материал Delphi ни маълум комплектига боғлик эмас. Мисол сифатида келтирилган барча масалаларни Personal комплектида амалга ошириш мумкин. Компьютерга Delphi 7 ни ўрнатиш CD-ROM дан амалга оширилади. CD-ROM да Delphi 7 ни ўрнатувчи барча файллар ва дастурлар мавжуд (Delphi Setup Launcher). Ўрнатувчи диск CD юритувчига кўйилиши билан ўрнатувчи дастур автоматик ишга тушади. Ўрнатувчи дастур ишга тушиши натижасида экранда (Delphi Setup Launcher) ойнаси (1-расм) хосил бўлади. Бу ойнада CD-ROM дан ўрнатилиши мумкин бўлган дастур махсулотлари рўйхати келтирилган. Бу рўйхатда биринчи бўлиб Delphi 7, маълумотлар базаси сервери Interbase 6.5, маълумотлар базаси локал сервери Interbase 6.5, масофавий отладка инструменти Remote Debugger Server, ModelMaker утилити ва юкланувчи CD-ROM яратувчи InstallShield Express утилити жойлаштирилган.

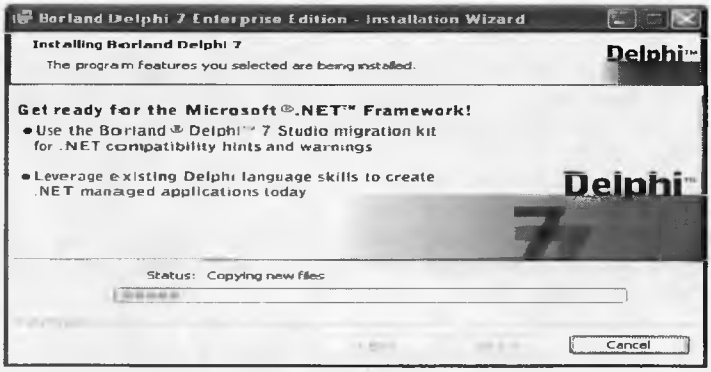
Delphi 7 ни ўрнатиш жараёнини бошлаш учун ойнадаги Delphi 7 сатри танланади. Delphi 7 ни ўрнатиш жараёни жуда оддий жараён хисобланади.

Custom (Танланувчи) вариантда дастурчи Delphi ни фақат зарур бўладиган инструмент ва компонентларини ўрнатиш учун танлаб кўрсатади. Одатда бу вариантни малакали дастурчилар танлайди. Танланувчи вариант компьютер дискида Delphi ни тўлик ўрнатиш учун етарлича бўш жой бўлмаган ҳолатларда ҳам танланади.

Ўрнатиш варианти танлангандан кейин Next тугмаси босилади. Агар Custom варианти танланган бўлса у ҳолда **Custom Setup** (3-расм) мулокат ойнаси ҳосил бўлади.

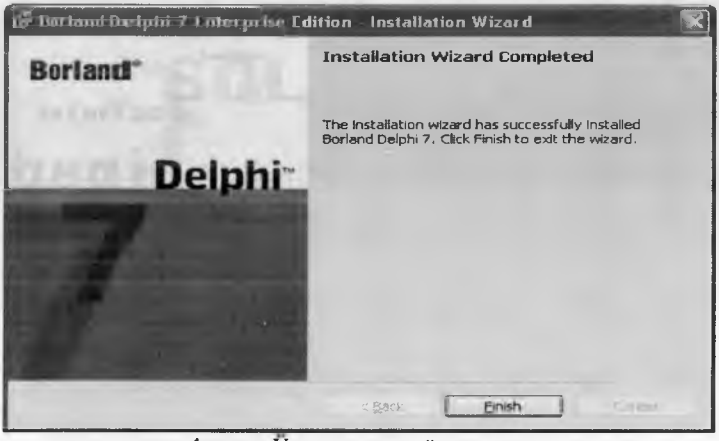


Бу ойнада ўрнатиладиган компонентлар танланади ёки бошқача айтганда, ўрнатилиши керак бўлмаган компонентлар кўрсатилади. Компонентни ўрнатишни тақиқлаш учун номидан чап томондаги диск тасвири сичқонча ёрдамида танланади ва ҳосил бўлган менюдаги **Do Not Insall** танланади.



3-расм. Компонентни ўрнатишни тақиқлаш.

Агар **Typical** ўрнатиш варианти танланса, у ҳолда Next тугмаси босилиши натижасида Delphi пакети ва унинг компонентлари ўрнатиладиган каталоглар кўрсатилган Destination Folder ойнаси ҳосил бўлади.

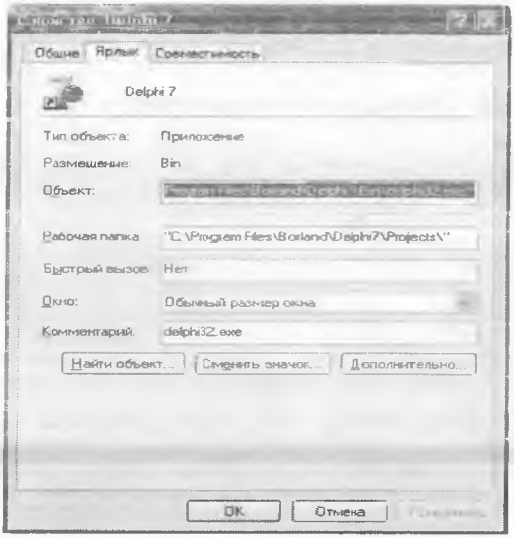


4-расм. Ўрнатиш жараёни тугади.

Next тугмасини навбатдаги босиш натижасида **Save Insallation Database** ойнаси ҳосил бўлади. Бу ойнада фойдаланувчига каттиқ дискка ўрнатиш жараёни ҳақидаги маълумотларни сақлаш таклиф этилади.

Бу Delphi ни кейинчалик ўрнатувчи CD– ROM дан фойдаланмасдан ўчириш (деинсталляция) имконини беради. Шу билан Delphi ни ўрнатиш

жараёнини тайёргарлиги тугади. Экранда **Ready To Insall the Program** ойнаси ҳосил бўлади.
Ойнадаги Install тугмасини босиш ўрнатиш жараёнининг фаоллаштиради. Ўрнатиш жараёнининг тугаши билан экранда ўрнатиш тугаганлиги ҳақидаги хабар келтирилган ойна (4-расм) ҳосил бўлади. Ойнадаги Finish тугмаси босиб бу ойна ёпилади.
Энди Delphi ни ишга тушириш мумкин. Бирок бундан олдин ишчи каталогни, яъни яратилган лойиҳалар каталогини кўрсатиш лозим. Бунинг учун сичқонча кўрсаткичини Delphi ни ишга тушириш буйруги (**Пуск\Программы\Borland Delphi7\Delphi7**) устида ўнг тугмасини босиб, ҳосил бўлган контекст менюсида **Свойства** сатри танланади.
Натижада ҳосил бўлган **Свойства: Delphi** ойнасидаги **Рабочая папка** майдонига Delphi лойиҳалари сақланадиган папка номи киритилади (5-расм).

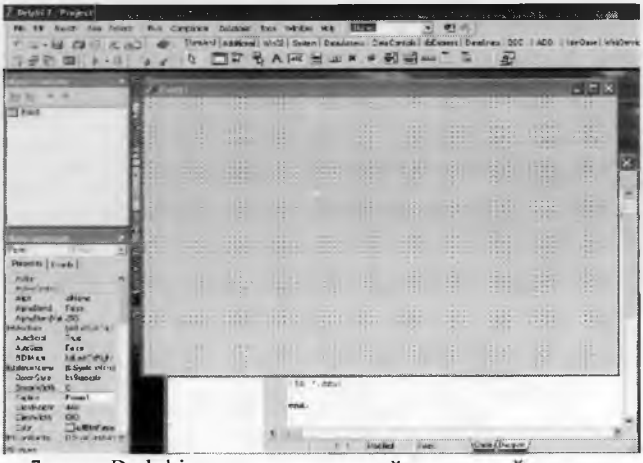


5-расм. Яратиладиган лойиҳалар папкасини кўрсатиш.

ИШНИ БОШЛАШ

Delphi дастурлаш мухити одатдаги гарзда, яъни **Borland Delphi 7** менюсидан **Delphi 7** командасини танлаш оркали ишга туширилади (6-расм).

6-расм. Delphi ни ишга тушириш.



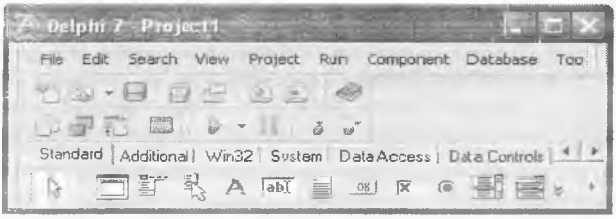
7-расм. Delphi ишга тушгандан кейин экран кўриниши.

Delphi ни ишга туширгандан кейин экран кўриниши одатдагидан бошқарок кўринишда бўлади (7-расм). Экранда битта ойна ўрнига 5 та ойна хосил бўлади. Булар

асосий ойна –**Delphi;**
бошлангич форма ойнаси – **Form 1**;
объектлар хоссаларини тахрирлаш ойнаси **- Object Inspector**;
объектлар рўйхатини кўриш ойнаси- **Object TreeView**;
кодни тахрирлаш ойнаси **- Unit1.pas**.

Кодни тахрирлаш ойнаси деярли тўлалигича бошланғич форма ойнаси билан тўсилган бўлади.

Асосий ойнада (8-расм) буйруқлар менюси инсрументлар панели ва компонентлар палитраси жойлашади.



8-расм. Асосий ойна.

Бошлангич форма ойнаси (**Form1**) яратилаётган *илованинг* ярим тайёр холдаги асосий ойнасидан иборат бўлади.

Дастурий таъминотлар тизимли ва амалий дастурий таъминотларга бўлинади.

Тизимли дасурий таъминот бу-операцион тизимни ташкил килувчи дастурлар.

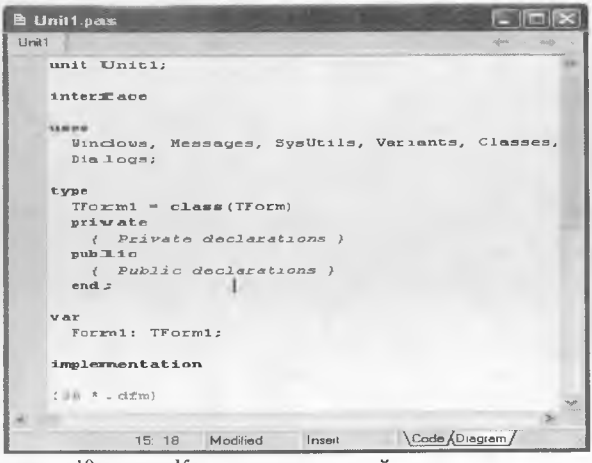
Колган дасурлар амалий дасурий таъминот деб кабул килинади.

Амалий дастурлар кискача килиб иловалар деб аталади.

9-расм. Properties бўлимида объект хоссалари рўйхати ва хоссаларнинг кийматлари кўрсатилган.

Object Inspector ойнаси (9-расм) -бу объектлар хоссаларини таҳририлаш ойнаси бўлиб, объект хоссалари қийматини таҳрирлаш учун мўлжалланган. Визуал дасурлаш атамасида *объектлар* - бу мулокат ойналари ва бошкариш элементларидир (киритиш ва чикариш майдонлари, буйрук тугмалари ва ўлчамлари ва ҳаказо). *Объект ҳоссаси* - бу объектнинг кўринишини, ўрнини ва ўзини тутишни аниқловчи характеристикалардир. Масалан, Weight ва Height форманинг ўлчамини (кенглигини ва баландлигини), Top ва Left хоссаси формани экрандаги ўрнини, caption хоссаси форманинг сарлавҳасини аниклайди.

Код таҳрирлаш ойнасини (10-расм) кўриш учун форма ойнасини бирор томонга, яъни экран четроғига олиб туриш лозим. Кодни таҳририлаш ойнасида дастур матни таҳрирланади. Янги лойиҳа устида иш бошлашдан олдин кодни таҳрирлаш ойнасида Delphi томонидан автоматик яратилган дастур шаблони (матни) жойлашади.

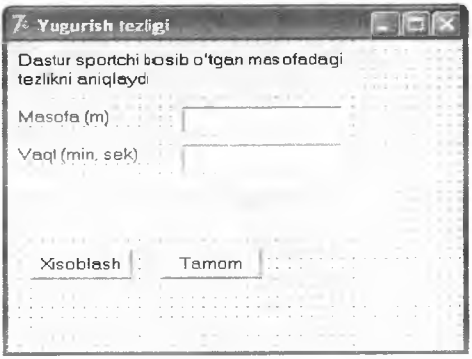


10-расм. Кодни таҳрирлаш ойнаси.

БИРИНЧИ ЛОЙИҲА

Delphi имконятларини ва Visual лойиҳалаш технологиясини номойиш этиш максадида янги илова яратамиз. Бу илова спортчи босиб ўтган масофадаги спортчининг тезлигини аниклайди.

Янги дастур яратишни бошлаш учун Delphi ни ишга туширинг. Агар Delphi мухити ишга туширилган ва бошка лойиҳа юкланган бўлса, у холда янги лойиҳани бошлаш учун File(Файл) менюсидан New Application сатрини танланг.



11-расм. Югириш тезлигини хисоблаш дастури ойнаси.

ФОРМА

Delphida яратилаётган илова *янги лойиха* деб аталади. Янги лойиха устида иш бошланганда форма яратишдан бошланади. Дастурни яратиш жараёнида бу ойна мулокот ойнаси деб аталади.

Бошланғич форма Form1 хоссалари қийматларини ўзгартириш ва формага янги компонентларни (матиларни киритиш ва чикариш майдонлар, буйрук тугмалари) қўшиш оркали яратилади.

Форма хоссаси (1-жадвал) унинг ташки кўринишини, яъни ўлчамини, экрандаги ўрнини, сарлавха сатрини ва рамка шаклини аниклайди. Форма хоссаларини кўриш ва ўзгартириш учун **Object Inspector** (9-расм) ойнасидан фойдаланилади. Жорий вақтда хоссаларининг кийматлари тасвирланаётган объектнинг номи **Object Inspector** ойнасининг юкори кисмида кўрсатилган бўлади. Properties (хосса) катламининг чап устунида объект хоссаларининг номи жойлашган, ўнгда эса хоссаларнинг киймати кўрсатилган.

Форма (form объекти) хоссалари

1-жадвал

Хосса	Таърифи
Name	Форма номи. Дастурда форма номи формани бошкариш ва унда жойлашган компонентларга мурожат этиш учун ишлатилади
Caption	Сарлавха матни
Width	Форма кенглиги
Height	Форма баландлиги
Top	Форманинг юкори чегарасидан экраннинг юкори чегарасигача бўлган масофа
Left	Форманинг чап чегарасидан экраннинг чап чегарасигача бўлган масофа

BorderStyle	Чегара кўриниши. Чегара оддий (bsSizeable), нозик (bsSingle) ёки бўлмаслиги (bsNone) мумкин. Агар ойна одатдаги чегарага эга бўлса, у ҳолда дастур ишлаши жараёнида фойдаланувчи унинг ўлчамини ўзгартириши мумкин. Нозик чегарали ойна ўлчамини ўзгартириш имкони йўқ. Агар чегара мавжуд бўлмаса, у ҳолда дастур ойнаси экранда сарлавҳасиз тасвирланади. Бундай ойна ўрнини ва ўлчамини ўзгартириш мумкин эмас.
derIcons	Ойнани бошқариш тугмалари. Бу ҳоссанинг қийматлари дастур ишлаши жараёнида фойдаланувчи ойнани бошқариш учун қайси тугмалардан фойдаланиши мумкинлигини аниқлайди. Бу ҳосса қиймати biSystemMenu, biMinimaze, biMaximaze ва biHelp аиклаштирувчи ҳоссалар қийматларини ўрнатиш орқали аникланади. biSystem Меню ҳоссаси ойнада Свернуть тугмаси ва тизимли меню мавжуд бўлишини, biMinimaze—Свернуть тугмаси, biMaximaze—Развернуть тугмаси, biHelp — справка маълумотларини чиқариш тугмалари мавжуд бўлишини билдиради.
Icon	Мулокат ойнасидаги тизимли менюни чикариш тугмасини билдирувчи белги
Color	Ойнанинг фон ранги. Рангнинг номини кўрсатиб ёки операцион тизимнинг жорий ранг схемасига боғлаш орқали кўрсатиш мумкин.
Font	Шрифт. Ойна юзасида жойлашган компонентларда "жимликка кўра" қўлланиладиган шрифт. Форманинг Font ҳоссасининг ўзгартирилиши форманинг юзасида жойлашган компонентнинг Font ҳоссасининг автоматик ўзгаришига сабаб бўлади. Яъни компонентлар форманинг Font ҳоссасига ворислик қилади (ворислик қилишни тақиклаш ҳам мумкин).

Форма яратишда биринчи навбатда унинг caption ҳоссасининг қиймати ўзгаритирилади. Бизнинг мисолда Form1 матни "югуриш тезлиги" матнига ўзгартирилади. Буни амалга ошириш учун **Object Inspector** ойнасидаги caption сатри сичконча билан танланади. Натижада бу ҳоссанинг жорий қиймати белгилаб олинади ва сатрда курсор пайдо бўлади. Энди "югуриш тезлиги" матнини киритиш мумкин (12-расм).

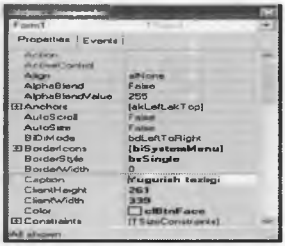
Худди шундай тарзда форма баландлиги ва кенглигини аникловчи Height ва width ҳоссаларининг қиймати ўрнатилади. Форма ўлчами ва унинг экрандаги ўрни ҳамда бошка бошқариш элементларининг ўлчами ва уларнинг формада жойлашиш ўрни пикселларда, яъни экран нукталари сони билан кўрсатилади. Height ва width ҳоссаларига мос ҳолда 250 ва 330 қийматлари ўзлаштирилади. Форма - бу одатдаги ойнадир. Шунинг учун унинг ўлчамини худди бошка ойналардаги каби ўлчамини (сичконча ёрдамида) ўзгартириш мумкин. Сичконча ёрдамида ойна чегараси ушлаб кўчирилганда Height ва width ҳоссаларининг қиймати автоматик ўзгартирилади. Бу кийматлар форманинг ўзгартирилган ўлчамига мос ўзгаради.

BorderStyle	Чегара кўриниши. Чегара оддий (bsSizeable), нозик (bsSingle) ёки бўлмаслиги (bsNone) мумкин. Агар ойна одатдаги чегарага эга бўлса, у холда дастур ишлаши жараёнида фойдаланувчи унинг ўлчамини ўзгартириши мумкин. Нозик чегарали ойна ўлчамини ўзгартириш имкони йўк. Агар чегара мавжуд бўлмаса, у холда дастур ойнаси экранда сарлавхасиз тасвирланади. Бундай ойна ўрнини ва ўлчамини ўзгартириш мумкин эмас.
derIcons	Ойнани бошкариш тугмалари. Бу хоссанинг кийматлари дастур ишлаши жараёнида фойдаланувчи ойнани бошкариш учун кайси тугмалардан фойдаланиши мумкинлигини аниклайди. Бу хосса киймати biSystemMenu, biMinimaze, biMaximaze ва biHelp аниклаштирувчи хоссалар кийматларини ўрнатиш орқали аникланади. biSystem Меню хоссаси ойнада Свернуть тугмаси ва тизимли меню мавжуд бўлишини, biMinimaze—Свернуть тугмаси, biMaximaze—Развернуть тугмаси, biHelp — справка маълумотларини чикариш тугмалари мавжуд бўлишини билдиради.
Icon	Мулокат ойнасидаги тизимли менюни чикариш тугмасини билдирувчи белги
Color	Ойнанинг фон ранги. Рангнинг номини кўрсатиб ёки операцион тизимнинг жорий ранг схемасига боғлаш орқали кўрсатиш мумкин.
Font	Шрифт. Ойна юзасида жойлашган компонентларда "жимликка кўра" кўлланиладиган шрифт. Форманинг Font хоссасининг ўзгартирилиши форманинг юзасида жойлашган компонентнинг Font хоссасининг автоматик узгаришига сабаб бўлади. Яъни компонентлар форманинг Font хоссасига ворислик килади (ворислик килишни такиклаш хам мумкин).

Форма яратишда биринчи навбатда унинг caption хоссасининг киймати ўзгаритирилади. Бизнинг мисолда Form1 матни "югуриш тезлиги" матнига ўзгартирилади. Буни амалга ошириш учун **Object Inspector** ойнасидаги caption сатри сичконча билан танланади. Натижада бу хоссанинг жорий киймати белгилаб олинади ва сатрда курсор пайдо бўлади. Энди "югуриш тезлиги" матнини киритиш мумкин (12-расм).

Худди шундай тарзда форма баландлиги ва кенглигини аникловчи Height ва width хоссаларининг киймати ўрнатилади. Форма ўлчами ва унинг экрандаги ўрни хамда бошка бошкариш элементларининг ўлчами ва уларнинг формада жойлашиш ўрни пикселларда, яъни экран нукталари сони билан кўрсатилади. Height ва width хоссаларига мос холда 250 ва 330 кийматлари ўзлаштирилади.

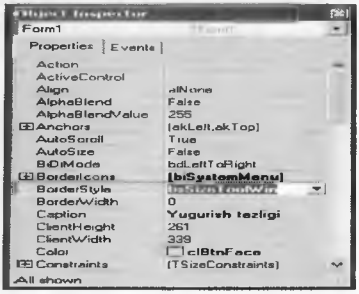
Форма - бу одатдаги ойнадир. Шунинг учун унинг ўлчамини худди бошка ойналардаги каби ўлчамини (сичконча ёрдамида) ўзгартириш мумкин. Сичконча ёрдамида ойна чегараси ушлаб кўчирилганда Height ва width хоссаларининг киймати автоматик ўзгартирилади. Бу кийматлар форманинг ўзгартирилган ўлчамига мос ўзгаради.



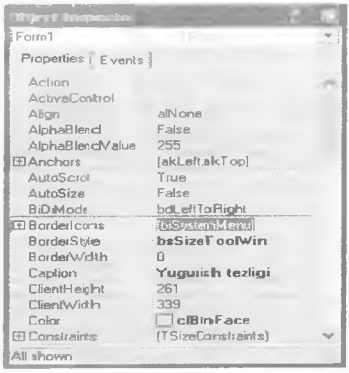
12-расм. Қийматни киритиш орқали хосса кийматини ўрнатиш.

Дастур ишга тушганидан кейин мулоқат ойнасининг ўрни уни яратиш жараёнида форманинг Top (экран юқорисигача бўлган масофа) ва Left (экранни чап чегарасигача бўлган масофа) хоссаларида ўрнатилган кийматларига мос келади. Бу хоссалар кийматини формани сичқонча ёрдамида кўчириш орқали ҳам ўрнатиш мумкин.

Баъзи бир хоссаларни танлаганда, масалан, borderstyle танланганда бу хоссанинг жорий
кийматидан ўнг томонда очилувчи рўйхат белгиси пайдо бўлади. Бундай хосса кийматини рўйхатдан танлаб ўрнатиш мумкин (13-расм). Баъзи хоссалар эса мураккаб ҳисобланади. Яъни унинг киймати бошқа (аниклаштирувчи) хоссалар кийматлари тўплами орқали белгиланади. Мураккаб хоссалар номидан чап томонда "+" белгиси жойлашади. "+" белгиси сичқонча билан танланганда аниклаштирувчи хоссалар рўйхати очилади (14-расм). Масалан, BorderIcons хоссаси дастур ишлаши вақтида ойнани бошқарувчи кайси тугмалар тасвирланишини аниклайди. Агар ундаги biMaximize хоссасига False киймати ўзлаштирилса, у ҳолда дастур ишлаш вақтида ойнанинг сарлавҳасида Развернуть тугмаси йўк бўлади.



13-расм. Рўйхатдан танлаш орқали хосса кийматини ўрнатиш



14-расм. Мураккаб тузилишга эга бўлган BorderIcons хоссаси

Баъзи хоссаларнинг қиймати ёнида учта нуқтадан иборат буйрукли тугма жойлашади. Бу тугма жорий хосса қийматини ўрнатиш учун қўшимча мулокат ойнасидан фойдаланиш имконини беради. Масалан, Font мураккаб хосса кийматини бевосита уни ташкил қилувчи қўшимча хосса қийматларини киритиш орқали, ҳамда шрифт танлашни стандарт мулокат ойнасидан фойдаланиб киритиш мумкин.
2-жадвалда яратилаётган дастур формасининг ўзгартирилиши керак бўлган хоссаларининг рўйхати келтирилган. Қолган хоссалари ўзгартирилмайди ва жадвалда келтирилмаган.

Бошланғич форма хоссаларининг кийматлари 2-жадвал.

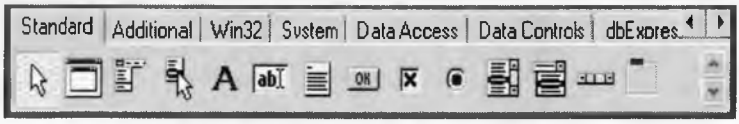
Хосса	Қиймат
Caption	Югуриш тезлиги
Height	250
Width	330
BorderStyle	bsSingle
BorderIcons.biMinimize	False
BorderIcons.biMaximize	False
Font. Size	10

Келтирилган жадвалда баъзи хосса номларида нуқта ишлатилган. Бу ушбу хоссани аниқлаштирувчи хосса кийматини ўрнатиш кераклигини билдиради. Асосий форманинг барча хоссаларининг қиймати ўрнатилганидан кейин унинг кўриниши 15-расмда кўрсатилган ҳолатга келади.

15-расм. Хосса кийматлари ўрнатилгандан кейинги форма шакли.

КОМПОНЕНТАЛАР

Югуриш тезлигини ҳисоблаш дастурида фойдаланувчи спортчининг югурган масофа узунлиги ва кетган вақт кийматларини киритиши керак. Маълумотлар клавиатурадан киритилиши керак бўлган дастурларда, коидага кўра, маълумотлар таҳрирлаш майдонига киритилади. Шунинг учун формага таҳрирлаш майдони – Edit компоненти қўйилади. Энг кўп ишлатиладиган компонентлар **Standart** катламида жойлашган (16-расм).
Формага бирор компонентни қўйиш учун компонентлар палитрасидаги зарур компонентнинг пиктограммаси сичконча билан танланади. Кейин сичконча кўрсатгичи формада компонентнинг юкори чап бурчаги жойлашиши керак бўлган жойга олиб борилади ва яна сичконча тугмаси босилади. Натижада формада стандарт ўлчамдаги компонент хосил бўлади.



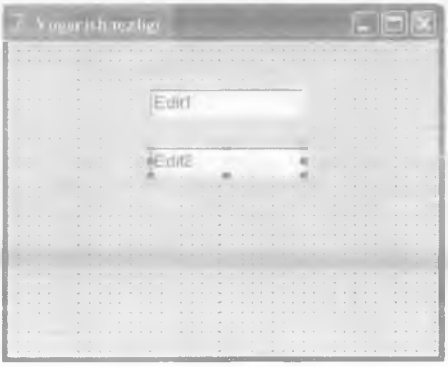
16-расм. Standard катламида энг кўп кўлланиладиган компонентлар жойлашади.

Компонент ўлчамини уни формага қўйиш жараёнида белгилаш мумкин. Бунинг учун компонентлар палитрасидан компонент танлагандан кейин сичқонча кўрсатгичи формада компонентнинг юқори чап бурчаги жойлашиши керак бўлган жойга олиб борилади, чап тугмаси босилган ҳолатда кўрсатгич компонентни пастки ўнг бурчаги жойлашиши керак бўлган нуқтага олиб борилади ва юборилади. Натижада формада зарур ўлчамдаги компонент ҳосил бўлади.

Ҳар бир компонентга Delphi унинг номи ва тартиб номеридан иборат ном беради. Масалан, агар формага иккита Edit компоненти қўшилса, у ҳолда уларни номи Edit1 ва Edit2 бўлади (17-расм). Дастурчи Name ҳоссаси қийматини ўзгартириш орқали компонент номини ўзгартириши мумкин. Оддий дастурларда, қоидага кўра, компонент номини ўзгартирмайдилар.

Бошланғич маълумотларни киритиш учун мўлжалланган иккита Edit таҳрирлаш майдони қўшилгандан кейинги форма кўриниши келтирилган.

Компонентлардан бири белгилаб олинган. Белгилаб олинган компонент ҳоссалари Object Inspector ойнасида тасвирланиб туради. Бошқа компонент ҳоссасини кўриш учун формадаги зарур компонент сичқонча билан танланади. Шунингдек, бу компонент номини Object TreeView ойнасининг ёки Object Inspector ойнасининг юқорисидаги очилувчи рўйхатдан компонент номини танлаш мумкин.



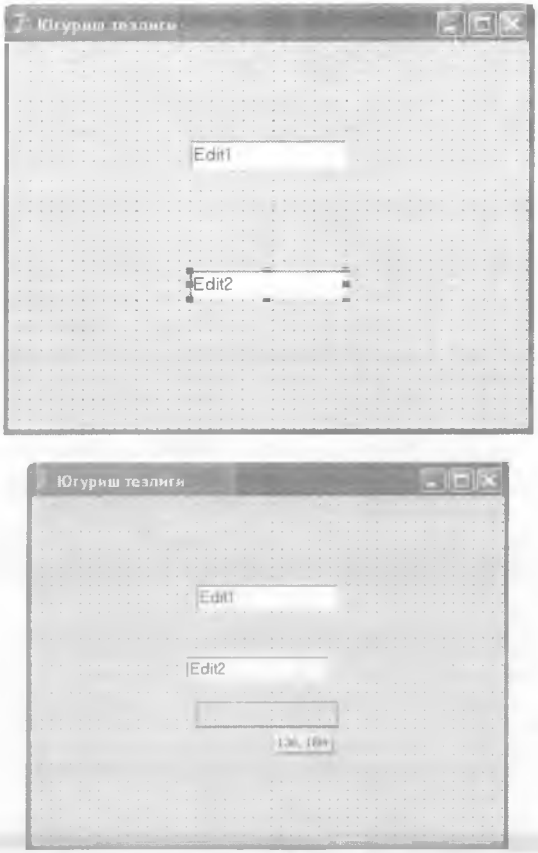
17-расм. Edit компонентлари қўшилгандан кейин форма кўриниши.

3-жадвалда матн-таҳрирлаш майдони-Edit компонентнининг асосий ҳоссалари келтирилган.

Edit (матн-таҳрирлаш майдони) компонентилари 3 – жадвал

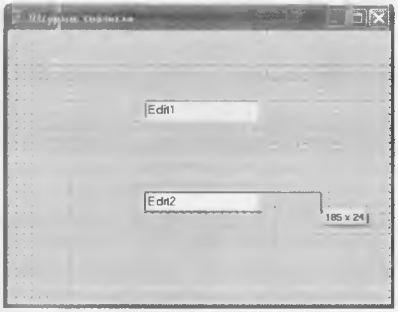
Хосса	Таърифи
Name	Компонент номи. Дастурда компонент ва унинг ҳоссаларига, хусусан, таҳрирлаш майдонига киритилган матнга мурожат этиш учун қўлланилади.
Text	Матн таҳрирлаш майдонидаги матн
Left	Компонентнинг чап чегарасидан форманинг чап чеграсигача бўлган масофа
Top	Компонентнинг юқори чегарасидан форманинг юқори чегарасигача бўлган масофа
Height	Майдон баландлиги
Width	Майдон кенглиги
Font	Матнни тасвирлаш шрифти
ParentFont	Компонент ўзи жойлашган форма шрифти характеристкаларига ворислик килиш аломати. Агар бу ҳосса киймати True булса, у ҳолда форманинг Font ҳоссаси ўзгартирилганда компонентнинг ҳам Font ҳоссаси кийматлари шунга мос ҳолда автоматик ўзгаради.

Delphi компонент ўлчамини ва жойлашиш ўрнини сичконча ёрдамида ўзгартириш имконини беради.
Компонентни жойлашиш ўрнини ўзгартириш учун сичконча кўрсатгичи компонент тасвири устида жойлаштирилади, чап тугмаси босилади ва босилган ҳолатда ушлаб кўрсатгич форманинг керакли жойига олиб борилиб, сичконча тугмаси юборилади. Бу усулда компонентни кўчиришда компонентнинг чап юкори бурчагининг жорий кийматлари (Left ва Top хоссаларининг кийматлари) тасвирланиб борилади (18-расм). Компонент ўлчамини ўзгартириш учун у белгилаб олинади.
Белгилаб олиш натижасида ҳосил бўлган компонент чегарасини кўрсатувчи маркерлардан бири устида сичконча кўрсатгичи жойлаштирилади ва чап тугмаси босилган ҳолатда ҳаракатлантирилиб компонентнинг керакли ўлчами ўрнатилади. Компонент
ўлчамини ўзгартиришда Height ва Width ҳоссасларининг жорий кийматлари тасвирланиб борилади (19-расм).



18-расм. Компонентнинг жойлашиш ўрнини ўзгартиришда Left ва Top хоссалари қийматларини тасвирланиши.

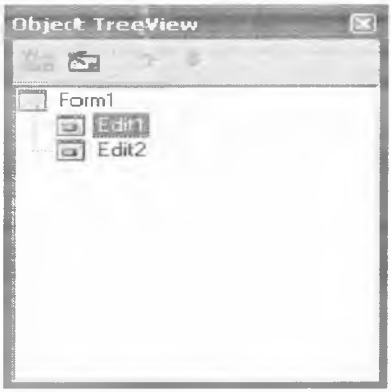
Компонент хоссасини ҳам форма хоссаси каби Object Inspector ёрдамида ўзгартириш мумкин. Зарур компонент хоссаси қийматини Object Inspector ойнаси орқали киритиш учун бу компонентни белгилаб олиш (сичқонча билан унинг тасвирини босиш) зарур.



19-расм. Компонентнинг жойлашиш ўрнини ўзгартиришда Height ва Width хоссалари қийматларини тасвирланиши.

Компонентни Object Inspector ойнасининг юқорисида жойлашган очилувчи рўйхатдан компонент номини танлаш (20-расм.) ёки ObjectTree View ойнасидаги рўйхатдан (21-расм) танлаб ҳам белгилаб олиш мумкин.

20-расм. Компонентни танлаш.



21-расм. Компонентни Object Inspector ойнасидаги рўйҳатдан танлаш

ObjectTree View ойнасидаги рўйҳатдан (21-расм) номини танлаш орқали ҳам белгилаб олиш мумкин.
4-жадвалда Edit1 ва Edit2 таҳрирлаш майдонлари хоссаларининг қиймати келтирилган. Edit1 компоненти масофа узунлигини киритиш учун мўлжалланган. Edit2 вақтни киритиш учун мўлжалланган.
Бу иккила компонентни ҳам Text хоссасининг қиймати буш сатрга тенг.

Edit компоненти хоссаларининг кийматлари 4 – жадвал

Хосса	Компонент	
	Edit1	Edit2
Text		
Top	56	88
Left	128	128
Height	21	21
Width	121	121

Дастур ойнасидаги таҳрирлаш майдонлари ёнида дастур ва маълумот киритиш майдонларининг вазифалари ҳақидаги қисқача маълумот жойлашиши керак. Формага матн чиқариш учун матн чиқариш майдонларидан фойдаланилади. Матн чиқариш майдонлари (статик матн майдонлари) – бу

Label компонентидир. Label компоненти белгиси Standart (22-расм) катламида жойлашган. Label компоненти хам формага худди тахрирлаш майдонлари каби кўшилади.

22-расм. Label компоненти – матн чикариш майдони.

Яратилаётган илова формасига тўртта Label компоненти керак. Улардан биринчиси дастур хакидаги маълумотларини, иккинчи ва учунчиси киритиш майдонларини вазифалари хакидаги маълумотларни, тўртинчиси эса хисоблаш натижаларини (тезликни) чикариш учун мўлжалланган. Label компонентининг хоссалари 5- жадвалда келтирилган.

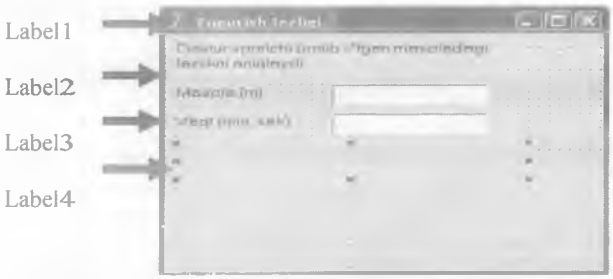
Label (матн чикариш майдони) компоненти хоссалари 5-жадвал

Хосса	Таърифи
Name	Компонент номи. Дастурда жорий компонент ва унинг хоссаларига мурожат этиш учун кўлланилади
Caption	Тасвирланадиган матн
Font	Матнни тасвирланиш шрифти
ParentFont	Жорий компонент ўзи жойлашган форманинг шрифт характеристикаларига ворислик килишини билдирувчи аломат. Агар бу хосса киймати True га тенг бўлса, у холда матн форма учун ўрнатилган шрифт билан тасвирланади
Autosize	Майдон ўлчами майдонда тасвирланадиган матнга боглик аникланишни билдирувчи хосса
Left	Чикариш майдонининг чап чегарасидан форманинг чап чегарасигача бўлган масофа
Top	Тасвирлаш майдонинг юкори чегарасидан форманинг юкори чегарасигача бўлган масофа
Height	Чикариш майдонинг узунлиги
Width	Чикариш майдонинг кенглиги
Wordwarp	Жорий сатрга сигмаган сўз автоматик холда навбатдаги сатрга ўтказилишини билдирувчи белги

Бу ердаги Autosize ва Wordwarp хосасаларига эътибор қилинг. Бу хоссалар чиқариш майдонида бир неча сатрдан иборат матн чиқариладиган бўлса керак бўлади.

Label компоненти формага қўшилганида унинг Auto size хоссаси қийматига True га тенг бўлади. Яъни майдон ўлчами caption хоссаси қиймаги ўзгартирилиш жараёнида автоматик аниқланади. Агар чиқариш майдонидаги матн бир неча сатрдан иборат бўлса, у ҳолда формага Label компоненти қўшилгандан кейин унинг Autosize хосса қийматини False, Wordwarp хосса қийматини True га тенг қилиб ўрнатиш лозим. Кейин width ва height хоссалари қийматини ўзгартириш орқали майдоннинг зарур ўлчамларини ўрнатиш зарур. Фақат шундан кейингина майдонда тасвирланувчи caption хоссаси матнини киритиш мумкин.

Формага тасвирлаш майдонларини (4 та label компонентини) қўшиш ва уларнинг хоссалари қийматларини 6-жадвалга мос ўрнатгандан кейин дастур формасининг умумий кўриниши (23-расм.) тасвирдагидек бўлади.



23-расм. Матнларни тасвирлаш майдонларини қўшгандан кейин форма кўриниши.

Label 1. Label2, Label3 ва Label4 компонентлари хоссаларининг қийматлари

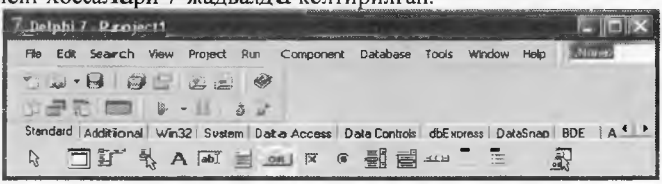
6-жадвал

Компанент	Хоссаси	Қиймати
Label1	Auto size	False
	Word wap	True

	Caption	Дастур спортчи босиб ўтган масофадаги тезликни аниқлайди
	Top	8
	Left	8
	Height	33
	Width	209
Label2	Top	56
	Left	8
	Caption	Масофа (метр)
Label3	Top	88
	Left	8
	Caption	Вақт (минутлар, секундлар)
Label4	Auto size	False
	Word wap	True
	Top	120
	Left	8
	Height	41
	Width	273

Форма яратиш жараёнидаги охирги иш-бу фомага иккита: ҳисоблаш ва тамом матнли иккита командали тугма қўшишдан иборат. Бу тугманинг вазифаси ўз - ўзидан маълум.

Button командали тугмалар ҳам формага худди бошка компонентлар каби қўшилади. Button компонент белгиси Satandart қатламида (24-расм) жойлашган. Бу компонент хоссалари 7-жадвалда келтирилган.



24-расм. Командали тугма – Button компоненти.

Button (командали тугма) компоненти хоссалари 7-жадвал

Хосса	Таърифи
Name	Компонент номи. Дастурда жорий компонент ва унинг хоссаларига мурожат этиш учун қўлланилади
Caption	Тугмада жойлашган матн
Enabled	Тугмани фаоллик белгиси. Агар хосса қиймати True бўлса тугмадан фойдаланиш мумкин. False бўлса мумкин эмас
Left	Тугмани чап чегарасидан форманинг чап чегарасигача бўлган масофа
Top	Тугмани юқори чегарасида форманинг юқори чегарасигача бўлган масофа
Height	Тугма баландлиги
Width	Тугма кенглиги

Форма иккита командали тугма киритгандан кейин уларнинг хоссалари қийматларини 8-жадвалга мос ўрнатилади .

Button1 ва Button2 тугмалари хоссалари қийматлари 8-жадвал

Хосса	Компонент	
	Button1	Button2
Caption	Хисоблаш	Тамом
Top	176	176
Left	16	112
Height	25	25
Width	75	75

Яратилаётган илова формасининг охирги шакли 25-расмда тасвирланган. Илова формасини яратиш ишлари тугагандан кейин дастур матнини ёзишга киришиш мумкин. Бироқ бундан олдин Windows да дастурлашдаги муҳим тушунчаларга таъриф бериш зарур. Булар:

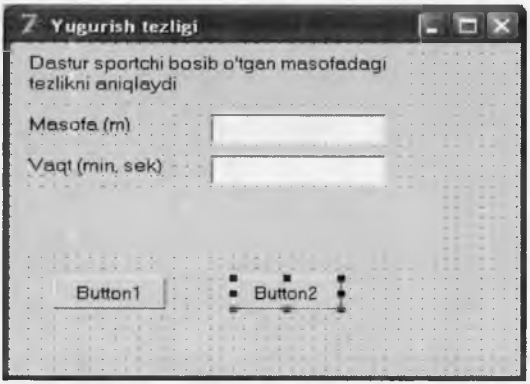
ҳодиса:

ҳодисани қайта ишлаш процедураси.

ҲОДИСА ВА ҲОДИСАНИ ҚАЙТА ИШЛАШ ПРОЦЕДУРАСИ

Яратилган форма кўриниши дастур бажарадиган вазифани англатади. Фойдаланувчи таҳрирлаш майдонига бошланғич маълумотларни киритади ва ҳисоблаш тугмасини босади. Командали тугмада сичқонча тугмасини босилиши -бу Windows даги ҳодисага мисолдир.

Ҳодиса (Event)-бу дастур ишлаши жараёнида юз беради. Delphi да ҳар бир ҳодисага ном берилган. Масалан тугмани сичқонча билан босиш-бу onclick ҳодисаси. икки марта босиш ondbclick ҳодисаси ҳисобланади.



25-расм. Югириш тезлиги дастурининг формаси.

9-жадвалда Windows даги баъзи ҳодисалар рўйхати келтирилган.

Ҳодисалар

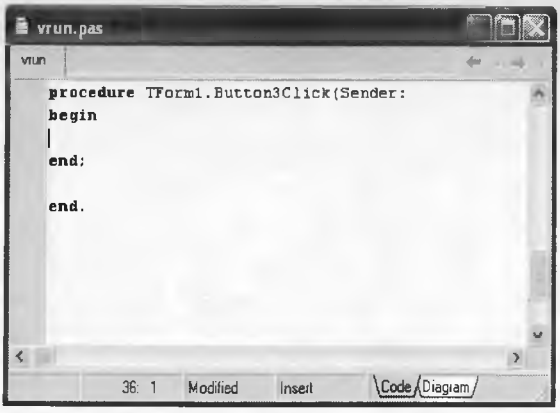
9 – жадвал

Ҳодиса	Юз беради
OnClick	Сичқонча тугмаси босилганда юз беради
OnDbClick	Сичқонча тугмаси 2 марта босилганда юз беради
OnMouseDown	Сичқонча тугмаси босилганда юз беради
OnMouseUp	Сичқонча тугмаси юборилганда юз беради
OnMouseMove	Сичқонча кўрсаткичи ҳаракатланганда юз беради
OnKeyPress	Клавиатура тугмаси босилганда юз беради.
OnKeyDown	Клавиатура тугмаси босилганда юз беради. OnkeyDown ва OnKeyPress ҳодисалари – бу кетма-кет ва такрорланувчи ҳодисалар бўлиб, улар босилган тугма юборилгунча (бу моментда OnKeyPress ҳодисаси юз беради) содир бўлади
OnKeyUp	Клавиатура тугмаси юборилганда содир бўлади.
OnCreate	Объект (формалар, бошкариш элементлари) яратилганда юз беради. Бу ҳодисаларни қайта ишлаш процедуралари одатда ўзгаурвчиларни яратиш, тайёр амалларни бажариш учун қўлланилади.
OnPaint	Дастур иши бошланишида экранда ойна ҳосил бўлганда, ойнанинг бошқа ойналар билан тўсилган кисми пайдо бўлганда ва бошқа ҳолларда юз беради
OnEnter	Элемент бошқарув фокусига эга бўлганда юз беради
OnExit	Элемент бошқарув фокусини йўқотганда юз беради

Ҳодисага таъсир сифатида бирор амал бажарилади. Delphi да ҳодисага акс жавоб ҳодисани қайта ишлаш процедураси шаклида бўлади. Шунинг учун дастур фойдаланувчи амалларига жавоб сифатида бирор ишни бажариши учун дастурчи ҳодисага мос қайта ишлаш процедурасини ёзиши керак.
Ҳодисани қайта ишлаш процедурасини тузиш ва ёзиш ишларининг катта қисмини компонент ўз зиммасига олади. Шунинг учун ҳодисага жавоб стандарт бўлмаган ва аниқланмаган ҳолатларда дастурчи ҳодисани қайта ишлаш процедурасини яратиши керак.
Масалан, масала шартига кўра Edit майдонига киритиладиган символар учун чекланиш йўк бўлса, у ҳолда OnKeyPress ҳодисасини қайта ишлаш процедурасини яратиш (ёки ёзиш) керак эмас. Яъни дастурни ишлаш жараёнида бу ҳодисани қайта ишлаш учун (дастурчига яширин бўлган) стандарт процедура қўлланилади.
Ҳодисани қайта ишлаш процедурасини яратиш услубини ҳисоблаш буйруқли тугмасининг OnClick ходисасини қайта ишлаш процедурасининг яратилиш мисолида кўриб чиқамиз.
Events қатламининг чап устунида (26-расм) танланган компонент (объект) қабул қиладиган ҳодисалар рўйхати жойлашади.
Агар бирор ҳодиса учун процедура аниқланган (ёзилган) бўлса, у ҳолда иккинчи устундаги жорий ҳодиса номи тўғрисида сатрда жорий процедура номи тасвирланади.

26-расм. Events қатламида компонент (жорий мисолда командали тугма) қабул қиладиган ҳодисалар рўйхати келтирилади.

Бирор ҳодисани қайта ишлаш функциясини яратиш учун мос ҳодисани қайта ишлаш процедураси номи жойлашган майдонда сичқонча икки марта босилади. Натижада код редактори ойнаси очилади ва унда ҳодисани қайта ишлаш процедураси шаблони жойлашади.
Object Inspector ойнасида ҳодиса номи ёнида уни қайта ишлаш ишлаш функцияси номи ҳосил бўлади (27-расм).
Delphi ҳодисасини қайта ишлаш функциясига икки қисмдан иборат ном беради. Биринчи қисми объект жойлаган формани билдиради. Иккинчи қисми объектни (компонентни) ўзини ва ҳодисани билдиради.
Бизнинг мисолда форма номи – Form1, командали тугма номи – Button1, ҳодиса номи – Click.



27-расм. Delphi яратган ҳодисани қайта ишлаш процедураси шаблони.

Редактор ойнасидаги Begin ва End сўзлари орасига ҳодисани қайта ишлаш функциясини амалга оширувчи инструкциялар ёзилади.
1-листингда ҳисоблаш номли тугманинг OnClick ҳодисасини қайта ишлаш функцияси матни келтирилган. Дастур қандай ёзилганлигига эътибор қилинг. Унда калит сўзлар қалин шрифт, изоҳлар-курсив шрифт билан тасвирланган. Бундан ташқари дастур инструкциялари сатрда бўш жой қолдирилиб қабул қилинган қоидалар асосида жойлашган.

1-листинг. Button1 (ҳисоблаш) тугмасининг OnClick ҳодисасини қайта ишлаш процедураси.

```
// ҳисоблаш тугмасини босиш
procedure TForm1.Button1Click(Sender: TObject);
var dist : integer; // масофа, метр
t: real; // вақт – каср сон
min : integer; //вақт, минут
sek : integer; //вақт, секунд
v: real; // тезлик
begin
// киритиш майдонидан бошланғич маълумотларни олиш
dist:= StrToint(Edit1.Text); t:=StrToFloat(Edit2.Text);
// дастлабки ўзгартиришлар
min := Trunc(t); // минутлар миқдори - бу t сонининг бутун қисми
sek := Trunc(t*100) mod 100;
// секунд миқдори - бу t сонининг каср қисми
// ҳисоблаш
v := (dist/1000) / ((min*60 + sek)/3600);
// натижани чиқариш
label4.Caption := 'Масофа: '+ Edit1.Text
+ ' м' + #13 + 'Время: ' + IntToStr(min)
+ ' мин ' + IntToStr(sek) + ' сек ' + #13 +
'Тезлик: ' + FloatToStrF(v,ffFixed,4,2) + ' км/с';end;
```

ButtonClick функцияси тезликни ҳисоблайди ва ҳисоблаш натижасини Label4 майдонига чиқаради. Бошланғич маълумотлар Edit1 ва Edit2 таҳрирлаш майдонларига, яъни уларнинг Text хоссасига мурожаат этиш орқали киритилади. Text хоссаси символлар сатридан иборат бўлиб, бу символларни дастур ишлаши жарёнида фойдаланувчи киритади. Дастур тўғри ишлаши учун сатр фақат рақамлардан иборат бўлиши керак. Сатрларни сонга айлантириш учун дастурда StrToInt ва StrToFloat функциялари ишлатилади. StrToInt функцияси унга параметр сифатида берилган сатр символларини (бу Edit1.Text -Edit1 майдонидаги матн) текширади ва барча символлар тўғри бўлса, мос сонни кайтаради. Бу сон Dist ўзгарувчисига ўзлаштирилади. StrToFloat функцияси ҳам ҳудди шундай

тарзда иш кўради. Яъни Edit2 майдонидаги матнга мос каср сон олинади ва бу сон t ўзгарувчисига ўзлаштирилади.
Бошлағич маълумотлар dist ва t ўзгарувчиларига берилганидан кейин тайёрланган амаллар ва ҳисоблаш бажарилади. Дастлаб Trunc функцияси ёрдамида сонни каср қисми ташлаб юборилади ва t ўзгарувчининг бутун қисми ажратиб олинади – бу минутлар миқдори ҳисобланади. Trunc (t*100) mod 100 ифодаси қиймати секундлар миқдорини билдиради.
Бу ифода қуйидагича ҳисобланади. Аввал t сони 100 га кўпайтирилади. Олинган нагижа Trunc функциясига берилади. Бу функция t ни 100 га кўпайтириш натижасида ҳосил бўлган сонни бутун қисмини олади. Олинган натижа модул бўйича 100 га бўлинади. Модул бўйича бўлиш - бу бўлишдаги қолдикни аниқлашдир. Барча маълумотлар тайёр бўлганидан кейин- ҳисоблаш бажарилади. Тезлик км/соат да ифодаланиши керак. Бироқ масофа ва вақт метр ва секундларда ифодаланган. Улар километр ва соатга ўзгартирилади.
Ҳисобланган қиймат Label4 майдонига, яъни қийматни бу компонетнинг Caption хоссасига ўзлаштириш орқали чиқарилади. Сонни сатрга айлантириш учун IntToStr ва FloatToStr функцияларидан фойдаланилади.
Тамом тугмаси босилганда дастур ишини тугатиши керак. Бу содир бўлиши учун дастур асосий ойнасини ёпиш ва экрандан йўқотиш керак. Бу Close методи ёрдамида амалга оширилади. **Тамом** тугмасининг OnClick ҳодисасини қисм процедураси 2-листингда келтирилган.
2-листинг. Button2 (**Тамом**) тугмасининг OnClick ҳодисасини қайта ишлаш процедураси.

```
// Тамом тугмасини босиш
procedure TForm1.Button2Click(Sender: TObject);
begin
Form1.Close; // дастурнинг асосий ойнасини ёпиш
end;
```

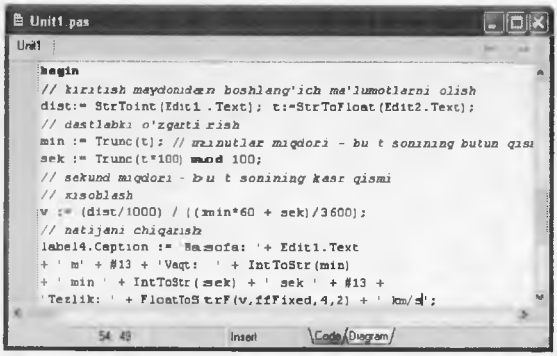
КОД РЕДАКТОРИ

Код редактори дастурлаш тилининг калит сўзларини (procedure, var, begin, end, if ва ҳ.к) ажратиб кўрсатади. Бу дастур матнини тушунарли ва ўқишга осон шаклга келтиради.
Калит сўзлар билан бирга код редактори изоҳларни курсив шаклда ажратиб кўрсатади.
Дастурни яратиш жараёнида тез-тез код редактори ойнасидан форма ойнасига ва аксинча ўтиш зарурати пайдо бўлади. Бу ишни View итнструментлар панелидаги **Toggle Form/Unit** тугмаси ёки клавиатурадаги <F12> тугмаси ёрдамида амалга

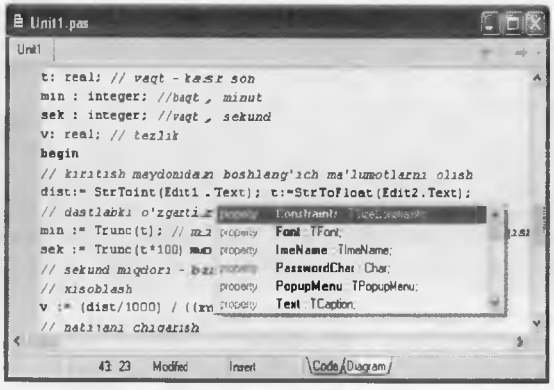
ошириш мумкин (28-расм). Агар лойиҳа бир неча модул ва формалардан иборат бўлса. юқорида номи зикр этилган инструментлар панелидаги View Unit ва View Form тугмалари ёрдамида зарур модул ёки форма ойнасига ўтиш мумкин.

28-расм . View инструментлар панели.

Дастур матнини киритиш жараёнида код редактори процедура ва функцияларнинг параметрлари, ҳамда объектларнинг хоссалари ва методлари ҳақидаги справка маълумотларини тасвирлаб беради.
Масалан, код редактори ойнасида MessageDig (Экранга хабар ойнасини чиқариш формаси) матни ва очилувчи қавс киритилса, у ҳолда экранда хабар ойнаси пайдо бўлиб, бу ойнада MessageDig функциясининг типлари кўрсатилган ҳолдаги параметрлар рўйҳати жойлашади (29-расм). Шу тарзда редактор дастурчига қайси параметр киритиш кереклигини кўрсатиб беради. Бунда бирор параметр киритилиб, вергул қўйилса. навбатдаги параметр белгилаб кўрсатилади. Барча параметрлар киритилиб бўлингунча шундай давом этади.
Код редактори объектларнинг барча хоссалари ва методларининг рўйҳатини кўрсатади. Дастурчи объект (компонент) номини ва нуктани киритиш билан дарров хабар ойнаси пайдо бўлиб, унда жорий объектнинг хоссалари ва методлари рўйхати кўрсатилади (30-расм). Бу рўйҳатнинг керакли элементига ўтиш учун курсорни юритиш клавишларидан фойдаланиш ёки зарур хосса ва метод номидаги дастлабки ҳарфларни териш керак. Ушбу усуллардан бири ёрдамида рўйхатдан зарур хосса ёки метод танланиб <Enter> тугмаси босилса, танланган хосса ёки метод номи дастур матнига қўйилади.
Хабарлар системаси дастур матнини киритиш жараёни учун катта ёрдам беради. Бундан ташқари, дастур матнини киритиш жараёнида хабар ҳосил бўлмаса. бу дастурчи хатога йўл қўйганини билдиради.



29-расм. Хабар.



30-расм . Код редактори объект(компонент) хоссалари ва методлари рўйхатини автоматик чикариш.

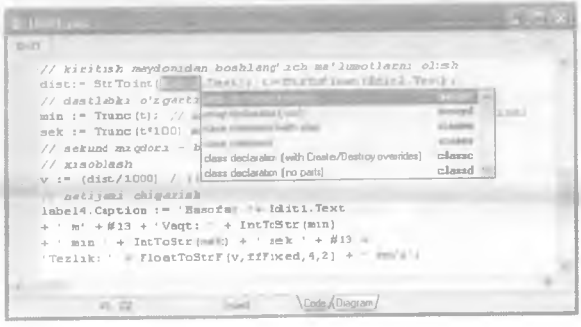
КОД ШАБЛОНЛАРИ

Дастур матнини киритишда *код шаблонларидан* (Code Template) фойдаланиш қулай. Код шаблони - бу умумий кўринишдаги дастур инструкцияларидир. Масалан, *case* инструкцияси шаблони қуйидагича:

```
Case of : ;
;;
Else;
End;
```

Код редактори дастурчини кўп миқдордаги шаблонлар билан таъминлаб беради. Булар: массивлар, класслар, функциялар, процедураларни эълон килиш, танлаш (if, case), циклллар (For, while) инструкцияларидир. Баъзи инструкцияларни, маслан, if ва while инструкцияларининг бир неча хил вариантдаги шаблонлари мавжуд.

Дастур матнини киритиш жараёнида код шаблонлари ва уларни дастур матнига қуйиб фойдаланиш учун <Ctrl>+<J> клавишлар комбинацияси босилади. Натижада хосил бўлган рўйхатдан керакли шаблон танланади (31-расм). Зарур шаблон одатдаги усул билан, яъни рўйхатни юкори ёки пастга ўтказиш ёки шаблон номидаги дастлабки харфларни киритиб танлаш мумкин (рўйхатдаги шаблон номлари калин шрифт билан ажратилган). Рўйхатдан керакли шаблон танланиб <Enter> босилади ва шаблон дастур матнига қўйилади.



31-расм . Код шаблонлари рўйхати <Ctrl>+<J> клавишлари босилганда хосил бўлади.

МАЪЛУМОТНОМА ТИЗИМИ

Дастурни киритиш жараёнида справка маълумотларини, масалан, тил конструкцияси ёки функция ҳақидаги маълумотларини олиш мумкин. Бунинг учун код редакторида справка олиши керак бўлган сўз (дастурлаш тили инструкцияси, процедура ёки функция номи ва ҳ.к.) киритилади ва <F1> босилади.
Справка маълумотларини **Help** менюсидан **Delphi Help** буйругини танлаб олиш мумкин. Бунда экранда справка сиситемасининг стандарт ойнаси ҳосил бўлади. Ушбу ойнадаги **Предментный указатель** майдонига зарур калит сўз киритилади. Қоидага кўра калит сўз сифатида функциялар, процедуралар, хоссалар ва методлар номларини дастлабки бир неча ҳарфлари киритилади.

ЛОЙИҲА ТУЗИЛИШИ

Delphi лойиҳаси дастур бирликлари - модуллардан иборат бўлади. Модуллардан бири асосий модул бўлиб, унда дастурни ишга тушурувчи инструкциялар жойлашади. Илованинг асосий модули тўлалигича Delphi томонидан яратилади.
Асосий модул .dpr кенгайтмали файлда жойлашади. Илованинг асосий модулининг матнини кўриш учун **Project** менюсидан **View Source** командаси танланади.
3-листингда югуриш тезлигини ҳисоблаш дастурининг асосий модули келтирилган.

3-листининг. Югуриш тезлиги иловасининг асосий модули.

```
uses
Forms,vrun1 in 'vrun1.pas' {Form1};
{$R *.res}
begin
Application.Initialize;
Application.CreateForm(TForm1, Form1);
Application.Run;
end.
```

Асосий модул program сўзи билан бошланади ва ундан кейин лойиҳа номига мос бўлган дастур номи жойлашади. Лойиҳа номи лойиҳани сақлашда кўрсатилади ва у орқали компилятор яратадиган дастурнинг ишчи файлнинг номи аниқланади. Uses сўзидан кейин қўлланиладиган модуллар номи жойлашади. Булар Forms библиотека модуллари ва vrun1.pas форма модулларидир.

Изоҳга ўхшаш бўлган {$R *.RES} сатри-бу компиляторга ресурслар файлини улашни билдирувчи дерективадир. Ресурслар файлида илова ресурслари: пиктограмалар, курсорлар, бит шакллари ва бошқалар жойлашади. Юлдузча белгиси ресурслар файлини номи лойиҳа файли номи билан бир хил. бирок .res кенгайтмали бўлишини билдиради.
Ресурслар файли матн файли бўлмаганлиги учун уни матн редактори ёрдамида кўриш имкони йўк. Ресурслар файли билан ишлаш учун махсус дастурлардан, масалан. Resourse Workshop фойдаланилади. Шунингдек. Delphi таркибиги кирувчи ImageEditor дастуридан фойдаланиш мумкин. Бу дастур Tools менюсидаги ImageEditor командасини танлаш оркали ишга тушурилади.
Асосий қисмнинг бошловчи қисми begin ва end инструкциялари орасида жойлашади. Бошқарувчи қисмдаги инструкциялар дастурни ишга тушириши ва бошланғич ойнани экранга чиқариш жараёнини бошқаради.
Ҳар - бир дастур асосий модул билан бирга камида яна битта форма модулига эга бўлади. Бу модулда илованинг бошланғич формаси тавсифи ва уни қўллаб - кувватловчи процедуралар жойлашади. Delphi да ҳар бир форма уз модулига эга бўлади.

4-листингда югуриш тезлигини хисоблаш дастури модулининг магни келтирилган. 4-листининг "Югуриш тезлиги" дастурининг модули.

```
unit vrun1;
interface
uses
Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms. Dialogs, StdCtrls:
type
TForm1 = class(TForm) Edit1: TEdit;
Edit2: TEdit; Label1: TLabel;
Label2: TLabel; Label3: TLabel;
Label4: TLabel;
Button1: TButton;
Button2: TButton;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject):
private
{ Private declarations } public
{ Public declarations } end:
var
Form1: TForm1:
implementation
{$R *.dfm}
```

```
// Ҳисоблаш тугмасини босиш
procedure TForm1. Button1Click'(Sender: TObject);
var dist : integer;  // масофа, метр
t: real; // вақт каср сон
min : integer; // вақт, минут
sek : integer; // вақт, секунд
v: real;
// тезлик
begin
// киритиш майдонларидан бошланғич маълумотларни олиш
dist := StrToInt(Edit1.Text); t := StrToFloat(Edit2.Text);
// дастлабки ўзгартириш
min := Trunc(t);  // минутлар миқдори - бу t сонининг бутун қисми
sek := Trunc(t*100) mod 100; // секунд миқдори - бу t сонининг каср
// қисми
// ҳисоблаш
v := (dist/1000) /  ((min*60 + sek)/3600);
// натижани чиқариш
label4.Caption := 'Масофа: '+ Edit1.Text + ' м' + #13
+ 'Вақт: ' + IntToStr(min) + ' мин '
+ IntToStr(sek) + ' сек ' + #13 +
'Тезлик: ' + FloatToStrF(v,ffFixed,4,2) + км/с';
end;
// Тамом тугмасини босиш
procedure TForm1.Button2Click(Sender: TObject)
begin
Form1.Close;
end;
end.
```

Хабар	Мумкин бўлган сабаби
Missing operator or semicolon (Опреатор ёки нукта вергул мавжуд эмас)	Операторлан кейин нукта вергул қўйилмаган

Модул unit сўзи билан бошланади ва ундан кейин модул номи жойлашади. Матни 3-листингда келтирилган илованинг асосий модулидаги uses инструкциясидан кейин фойдаланилган модуллар рўйхатида айнан юқоридаги ном кўрсатилади.

Модул қуйидаги бўлимлардан иборат бўлади.

Интерфейс:

Тадбиқ қилиш:
Инициализация
Интерфейс бўлими (interface сўзи билан бошланади) компиляторга модулининг қайси қисмларига дастурнинг бошқа модуллари мурожат этиши мумкинлигини ҳабар қилади. Шунингдек, бу ерда type сўзидан кейин Delphi томонидан яратилган форма тафсилотлари жойлашади.
Тадбиқ бўлими implementation сўзи билан очилади ва у ерда форма ишини қўллаб-қувватловчи локал ўзгарувчилар, процедуралар ва функциялар эълон қилинади.
Тадбиқ бўлими {$R *.DFM} директиваси билан бошланади. Бу директива компиляторга ишчи файлини яратиш жараёнида форма тавсилотларидан фойдаланиш кераклигини кўрсатади. Форма тавсилотлари номи модул номи билан мос бўлган .dfm кенгайтмали файлда жойлашади. Форма тавсилотлари файли форманинг ташки кўриниши асосида Delphi мухити томонидан яратилади.
{SR*.DFM} директивасидан кейин форма ва ундаги компонеталар билан боғлик ҳодисаларни қайта ишлаш процедуралари жойлашади. Бу ерга дастурчи бошка процедура ва функцияларни ҳам жойлаши мумкин.

ЛОЙИҲАНИ САҚЛАШ

Лойиҳа бу дастур ишчи файлини (exe-файл) яратиш учун компилятор томонидан фойдаланиладиган файллар тўпламидир. Оддий ҳолатларда лойиҳани тавсифлаш файлида (DOR-файл), асосий модул файлидан (DPR – файл), ресурслар файлидан (RES-файл), форма тавсифи файлидан (DFM – файл) дастурнинг асосий коди ва формадаги компонентлар ҳодисаларни қайта ишлаш функцияларининг коди жойлашган форма модули файлидан (PAS-файли), конфигурация файлидан (CFG-файл) иборат бўлади.
Лойиҳани сақлаш учун **File** менюсидаги **Save Project As** танланади. Агар лойиҳа биринчи марта сақланаётган бўлса, у ҳолда Delphi модулни (код редактори ойнасидаги матн) сақлашни таклиф этади. Шунинг учун экранда **Save Unit As** ойнаси ҳосил бўлади. Бу ойнада лойиҳа файллари сақланадиган папка танланади ва модул номи киритилади. Ойнадаги Сохранить тугмаси босилгандан кейин лойиҳа файли номини киритиш керак бўлган навбатдаги ойна ҳосил бўлади. Модул (pas-файл) ва лойиҳа (dpr-файл) файллари номи ҳар ҳил бўлиши керак. Компилятор томонидан яратилган ишчи файл номи лойиҳа номи билан мос бўлади. Шунинг учун лойиҳа файлига шундай ном бериш керак, у яратиладиган ишчи файл номига муносиб бўлсин.

КОМПИЛЯЦИЯ

Компиляция-бу бошланғич дастурни ишчи дастурга ўзгартириш жараёнидир. Компиляция жараёни иккита босқичдан иборат. Биринчи босқичда дастур матнидаги хатолар текширилади, иккинчи босқичда – ишчи дастур (exe - файл) ҳосил қилинади.
Ҳодисаларни қайта ишлаш функциялари матнини киритиш ва лойиҳани саклагандан кейин Project менюсидаги Compile буйруғини танлаб компиляцияни амалга ошириш мумкин. Компиляция жараёни ва натижалари **Compiling** ойнасида тасвирланиб боради. Компилятор бу ойнага ҳатоликлар ҳақидаги хабарлар (Errors), огоҳлантиришлар (warnings) ва маълумотномаларни (Hints) чиқаради. Хатоликлар хақидаги хабарлар, огоҳлантиришлар ва маълумотномаларнинг ўзи код редакторининг пастки қисмида тасвирланади.

ХАТОЛИКЛАР

Компилятор дастурнинг бошлангич матнида синтактик хатоликлар йўк бўлгандагина ишчи дастурни яратади. Кўпчилик ҳолларда қўлда киритилган дастур кодида хатоликлар мавжуд бўлади. Дастурчи уларни тўғирлаши керак. Хатолик мавжуд бўлган код матнига ўтиш учун курсорни хатолик жойлашган сатрга жойланади ва конматнли менюдан **EditSource** командаси танланади. Хатоликларни тўғирлаш итерацион характерга эга. Одатда аввал нисбатан оддий хатоликлар, масалан, эълон қилинмаган ўзгарувчилар эълон қилинади. Дастур матнидаги навбатдаги хатолик тўғирлангандан кейин такрорий равишда компиляция бажарилади. Компилятор доимо хатоликларни тўғри аниклай олмайди. Шунинг учун компилятор хатолик бор деб топган ва курсорни жойлаштирган дастур қисми таҳлил қилинади. Шу билан бирга хатолик мавжуд бўлган сатрдан олдинги сатрга хам эътибор қилиниши лозим.
10-жадвалда энг кўп йўл қўйиладиган хатоликлар ва уларга мос компилятор хабарлари келтирилган.
Компиляторни хатолик ҳақидаги хабари 10 – жадвал.
Агар компилятор жуда кўп ҳатоликларни аникласа, у ҳолда барча хатоликлар кўриб чиқилади. Дастлаб нисбатан соддарок бўлган хатоликлар тузатилади ва такроран компиляция амали бажарилади. Бундан кейин хатоликлар микдори тузатилган хатоликлар микдорига нисбатан кўпрок микдорда камаяди. Бу тил

синтаксис хусусиятлари билан боғлиқ. Яъни, унчалик аҳамиятсиз бўлган кичик хатолик кўп миқдордаги жиддий хатоликларга сабаб бўлади.
Агар дастур кодида синтактик хатоликлар мавжуд бўлса компилятор ишчи дастурни яратади. Ишчи файл номи лойиҳа номи билан бир хил, кенгайтмаси эса .exe бўлади. Delphi ишчи файлни лойиҳа файли жойлашган каталогда жойлайди.

ОГОҲЛАНТИРИШ ВА МАЪЛУМОТНОМАЛАР

Компилятор дастур кодида хатолик ҳисобланмайдиган камчиликларни аниқлаганда компилятор маълумотнома (Hints) ва огоҳлантиришлар (Warning) ҳосил қилади. Масалан. энг кўп ҳосил бўладиган маълумотномалардан бири эълон қилинган бироқ фойдаланилмаган ўзгарувчи ҳақидаги маълумотномадир: Variable ... is declared but never used in ... Ҳақиқатдан ҳам, фойдаланилмайдиган ўзгарувчини эълон қилишдан маъно йўқ. 11-жадвалда компилятор томонидан энг кўп бериладиган огоҳлантиришлар келтирилган.

Компилятор огоҳлантиришлари 11 – жадвал.

Огоҳлантириш	Сабаби
Variable... is declared but never used in ...	Ўзгарувчидан фойдаланилмаган
Variable . . . might not have been initialized.	Дастурда ўзгарувчига бошланғич қийматини ўзлаштирувчи инструкция мавжуд эмас

ДАСТУРНИ ИШГА ТУШИРИШ

Яратилган дастурни синаб кўриш учун Delphi муҳитида ишга тушириш мумкин. Бунинг учун **Run** менюсидаги **Run** сатри ёки **Debug** инструментлар панелидаги мос тугма танланади.

ДАСТУРГА ЎЗГАРТИРИШЛАР КИРИТИШ

Югуриш тезлиги дастурини бир неча бор ишга туширгандан кейин унга ўзгартириш киритиш эҳтиёжи туғилади. Масалан, масофа киритилиб <Enter> тугмаси босилганда курсор вақт майдонига утсин ёки **масофа** ва **вақт** майдонларига фойдаланувчи фақат рақамларни киритиш имконига эга бўлсин. Дастурга ўзгаритиш киритиш учун Delhi ишга туширилади ва мос лойиҳа очилади. Буни **File** менюсидаги **Open Project** командасини танлаб

амалга ошириш мумкин. Шунингдек File менюсидаги **Reopen** командасидан фойдаланиш мумкин. **Reopen** командаси танланганда охирги яратилган лойиҳалар рўйхати очилади.
5-листингда югуриш тезлиги дастуридаги Edit1 ва Edit2 компонентларининг OnKeyPress ҳодисасини қайта ишлаш процедураларини қўшилгандан кейинги ҳолати келтирилган.
Дастурга ҳодисани қайта ишлаш процедурасини қўшиш учун **Object Inspector** ойнасидан зарур компонент танланади, кейин ойнадаги **Events** қатламидаги ҳодиса танланади. Delphi ҳодисани қайта ишлаш процедурасини яратади. Кейин қайта ишлаш прцедурасини инструкцияларини киритиш мумкин.

5-листинг. Югуриш тезлиги дастурига ўзгартиришлар киритилганидан кейинги ҳолати.

```
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls;
type
TForm1 = class(TForm) Edit1: TEdit;
Edit2: TEdit; Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
Label4: TLabel;
Button1: TButton;
Button2: TButton;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Edit1KeyPress(Sender: TObject;
var Key: Char);
private
{ Private declarations } public
{ Public declarations } end;
var
Form1: TForm1;
implementation
{$R *.dfm)
// Ҳисоблаш тугмасини босиш
procedure TForm1.Button1Click(Sender: TObject);
var
dist : integer; // масофа, метр
t: real; // вакт, каср сон
```

```
min : integer; //вакт. минут
sek : integer; //вакт. секунд
v: real; // тезлиги
begin
// бошланғич маълумотларни киритиш майдонидан олиш
dist := StrToInt(Edit1.Text);
t := StrToFloat(Edit2.Text);
// бошланғич ўзгартириш
min := Trunc(t); // минут миқдори — бу t сонининг бутун қисми
sek := Trunc(t*100) mod 100; // секунд миқдори — бу t сонининг
//каср қисми
// ҳисоблаш
v := (dist/1000) / ( (min*60 + sek)/3600);
// натижани чиқариш
label4.Caption := 'Масофа: '+ Edit1.Text +
' м' + #13 + 'Вақт: ' + IntToStr(min) +
' мин ' + IntToStr(sek) + ' сек ' + #13 +
'Тезлик: ' + FloatToStrF(v,ffFixed,4,2) + ' км/с';
end;
// Тамом тугмасини босиш
procedure TForm1.Button2Click(Sender: TObject);
begin
Form1.Close; end;
// Масофа майдонида тугмани босиш
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
// Key — символ, босилган клавишга мос.
// Агар символ қаноатлантирмаса, у ҳолда процедура уни
// 0 кодли символга алмаштиради. Бунинг натижасида символ
//майдонда пайдо бўлмайди.
case Key of
'0'..'9': ; // рақам
#8 : ; // <Back Space> клавиши
#13 : Edit2.SetFocus ; // <Enter> тугмаси
// қолган символлар — тақиқланган
else Key :=Chr(0); // символ тасвирланмасин
end;
end;
end.
```

Ўзгартишлар киритилганидан кейин лойиҳа сақланади. Бунинг учун File менюсидаги Save All командаси танланади.

Назорат саволлари

1. Делфи дастурлаш мухити қайси ойналардан ташкил топган?
2. Яратиладиган лойиҳа файллари сақланадиган папка қандай кўрсатилади?
3. Лойиҳа формаси нима ва унда қандай компанентлар жойлашади?
4. Илова нима?
5. Объект ва объект хоссаси нима?
6. Делфи муҳитидаги Object Inspektor ойнасининг вазифаси нимадан иборат?
7. Дастур формаси қандай ҳоссаларга эга.
8. Объект ҳоссаси қийматлари қандай ўрнатилади?
9. Edit компоненти қандай ҳоссаларга эга?
10. Формадаги компонентларни танлашни қандай усуллари мавжуд?
11. Label компоненти қандай ҳоссаларга эга?
12. Ҳодиса нима?
13. Windows да юз берадиган қандай ҳодисалар мавжуд?
14. Код редакторининг вазифаси нима?
15. Маълумотнома тизимининг вазифаси нимадан иборат?
16. Лойиҳа қандай тузилишга эга?
17. Delphi муҳитида дастур қандай ишга туширилади?

I БОБ. ДАСТУРЛАШ АСОСЛАРИ

ДАСТУР

Дастурдан фойдаланаётган фойдаланувчи бошланғич маълумотларни "компьютерга киритади". Компьютер эса экранга, принтерга ёки файлга "натижаларни чиқаради". Аслида бошланғич маълумотларни компьютер процессори натижаларга айлантиради. Процессор бошланғич маълумотларни дастур деб аталувчи ва махсус тилда ёзилган алгоритм асосида натижаларга ўзгартиради. Шундай қилиб, компьютер бирор ишни бажариши учун бу ишни амалга оширувчи командалар кетма-кетлигини яратиш ёки бошқача айтганда дастур яратиш зарур.

ДАСТУР ЯРАТИШ БОСҚИЧЛАРИ

"Дастур яратиш" ибораси компьютер учун дастур яратиш жараёнининг бир босқичини билдиради. Дастурлаш-бу дастур яратиш (ишлаб чиқиш) жараёни бўлиб, қуйидаги босқичлардан иборат бўлади:

1. Дастурга қўйилган талабларни аниқлаш.
2. Алгоритмни ишлаб чиқиш.
3. Кодлаш (алгоритмни дастурлаш тилида ёзиш).
4. Отладка.
5. Тестлаш.
6. Маълумотнома тизимини яратиш.
7. Ўрнатувчи дискни (CD/DVD-ROM) яратиш.

ДАСТУРГА ТАЛАБ

Дастурга қўйилган талабларни аниқлаш-бу муҳим босқичлардан бири бўлиб, бунда бошланғич маълумотлар тавсифланади, натижаларга бўлган талаблар аниқланади, айрим ҳолатларда дастурни ишлаш тартиби аниқланади (масалан, нотўғри маълумотлар киритилганда), фойдаланувчи ва дастур ўртасидаги мулоқатни олиб бориш учун мулоқат ойналари ярагилади.

АЛГОРИТМНИ ИШЛАБ ЧИҚИШ

Алгоритмни ишлаб чиқиш босқичида натижаларга эриши учун бажариладиган амаллар кетма-кетлиги аниқланади. Агар масала бир неча усул билан ечиладиган бўлса, у ҳолда масалани ечишнинг бир неча вариантлари мавжуд бўлади. Бундай ҳолда дастурчи бирор мезондан

фойдаланиб (масалан, дастурни ишлаш тезлиги) энг маъкул ечимни танлайди. Алгоритмни яратиш босқичининг натижаси сўзлар билан ифодаланган алгоритм ёки блок-схемадан иборат бўлади.

КОДЛАШ

Дастурга қўйилган талаблар аниқланган ва ечиш алгоритми тузилгандан кейин алгоритм танланган дастурлаш тилида ёзилади. Натижада *бошланғич* дастур ҳосил бўлади.

ОТЛАДКА

Отладка-бу хатоларни излаш ва тузатиш босқичидир. Дастурдаги хатолар иккита группага ажратилади: синтактик (матндаги хатолар) ва алгоритмик хатолар. Синтактик хатолар-нисбатан осон тузатиладиган хатолардир. Алгоритмик хатоликларни аниқлаш нисбатан қийинроқ. Агар дастур бошлангич маълумотларни битта ёки иккита тўплами учун тўғри ишлаши отладка босқичини амалга ошган деб ҳисобласа ҳам бўлади.

ТЕСТЛАШ

Агар яратилган дастурдан бошқалар фойдаланиши учун мўлжалланган бўлса, у ҳолда тестлаш босқичи жуда муҳим ҳисобланади. Бу босқичда дастур бошланғич маълумотлар билан жуда кўп текшириб кўрилади. Шунингдек, нотўғри бошланғич маълумотлар учун ҳам текширилади.

МАЪЛУМОТНОМА ТИЗИМИНИ ЯРАТИШ

Агар дастур бошқалар фойдаланиши учун мўлжалланган бўлса, у ҳолда дастурчи албатта маълумотнома тизимини яратиши ва фойдаланувчи дастур билан ишлаши жараёнида унга қулай мурожат этиш имкониятини яратиши лозим. Замонавий дастурларда маълумотнома маълумотлари CHM ёки HLP-файллар шаклида тасвирланади. Маълумотнома тизими таркибига дастурни ўрнатиш (инсталлация) бўйича кўрсатмалар ҳам киритилади ва у Readme файли кўринишида TXT, DOC ёки HTM форматда яратилган бўлади.

ЎРНАТУВЧИ ДИСКНИ ЯРАТИШ

Ўранувчи диск (CD/DVD-ROM) фойдаланувчи дастурни ишлаб чиқувчи ёрдамисиз ўзи мустақил равишда компьютерига ўрнатиши учун яратилади. Одатда ўрнатувчи дискда дастур билан бирга ёрдамчи маълумотлар ва дастурни

ўрнатиш бўйича кўрсатмалар (Readme – файл) файллари ҳам жойлаштирилади. Замонавий дастурлар, шу қаторда Delphi тилида яратилган дастурлар ҳам (оддий дастурлардан ташқари) кўпчилик ҳолларда компьютерга оддийгина нусхалаш усули билан ўрнатилмайди. Чунки бу дастурлар ишлаши учун аниқ бир фойдаланувчи компьютерида йўқ бўлган махсус библиотека ва компонентлар зарур бўлиши мумкин. Шунинг учун компьютерга дастурни ўрнатиш ишини ўрнатувчи дискка ёзилган махсус дастур амалга ошириши керак. Қоидага кўра, ўрнатувчи дастур ўрнатилаётган дастур учун алоҳида папка яратади ва унга зарур файлларни нусхалайди. Агар зарур бўлса реестрга қўшимча ва ўзгартишлар киритиш орқали операцион системани созлайди.

АЛГОРИТМ ВА ДАСТУР

Дастур яратишнинг биринчи босқичида дастурчи қўйилган масалани ечиш учун бажарилиши керак бўлган амаллар кетма-кетмалигини аниқлаб олиши керак. Яъни алгоритмни ишлаб чиқиши керак. *Алгоритм*- бу бошлангич маълумотлардан натижаларга олиб борувчи жараённинг аниқ тавсифидир.

Масалани ечиш алгоритми сўзлар билан ёки график кўринишдаги блок-схема шаклида берилади.

Алгоритмни блок-схема шаклида тасвирлаш дастурчига масалани ечиш учун бажариладиган амаллар кетма-кетлигини аниқлашга, қўйилган масалани тўғри тушуниб олишга ёрдам беради.

Delphi тилида дастурлашда масалани ечиш алгоритми *ҳодисаларни қайта ишлаш процедуралари* тўпламидан иборат бўлади.

Мулокат ойналари ва ҳодисаларни кайта ишлаш алгоритмларини яратгандан кейин дастур матнини ёзиш мумкин. Мисолдаги дастур матни 1.1- листингда келтирилган.

1.1-листинг. Харид нархи дастури.

```
unit xarid_1;
interface uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls;
type
TForm1 = class(TForra)
Edit1: TEdit;
Edit2: TEdit;
Label1: TLabel;
Label2: TLabel;
Button1: TButton;
Label3: TLabel;
```

```
procedure Button1Click(Sender: TObject);
procedure Edit2KeyPress(Sender: TObject;
var Key: Char);
procedure Edit1KeyPress(Sender: TObject;
var Key: Char); private
{ Private declarations } public
{ Public declarations }
end;
var
Form1: TForm1;
implementation
{$R *.dfm}
// кисм дастур
procedure Summa;
var
baho: real; // баҳо
miqdor: integer; // миқдор
s: real; // сумма
mes: string[255]; // хабар
begin
baho := StrToFloat(Form1.Edit1.Text);
miqdor := StrToInt(Form1.Edit2.Text);
s := baho * miqdor;
if s > 500 then
begin
s := s * 0.9;
mes := ' 10% чигирмаланган' + #13;
end;
mes := mes+ 'харид нархи: '
+ FloatToStrF(s,ffFixed,4,2) +'сўм.';
Form1.Label3.Caption := mes;
end;
// Нарх тугмасини танлаш
procedure TForm1.Button1Click(Sender: TObject);
begin
Summa; // харид суммасини ҳисоблаш
end;
// Миқдор майдонида клавиш ни босиш
procedure TForm1.Edit2KeyPress(Sender: TObject;
var Key: Char);
begin
case Key of
```

```
'0' .. '9',#8 : : // рақамлар ва <Backspace> клавиши
#13: Summa; // харид нархини ҳисоблаш
else Key := Chr(O); // символ тасвирланмасин
end;
end;
// баҳо майдонида клавишни босиш
procedure TForm1.Edit1KeyPress(Sender: TObject;
var Key: Char);
begin
case Key of
'0' .. '9', #8 : ; // рақамлар ва <Backspace> клавиши
#13: Form1.Edit2.SetFocus; // <Enter> клавиши
'.',',' :
begin
if Key = '.'
then Key:=', if Pos(',',Edit1.Text) <> 0
then Key:= Chr(0);
end;
else // қолган символлар тақиқланади
Key := Chr(0);
end;
end;
end.
```

1.1-расм. Харид нархи дастурининг ойнаси (формаси).

КОМПИЛЯЦИЯ

Дастурлаш тилининг кўрсатмалари оркали ёзилган дастур бошланғич дастур деб аталади. Бу дастур инсонлар учун тушунарли бўлган, бирок компьютер процессори учун тушунарсиз бўлган кўрсатмалардан иборат бўлади. Процессор бошланғич дастурнинг кўрсатмаларига мос ҳолда амалларни бажариши учун бошланғич дастур машина тилига, яъни процессор кўрсатмалари тилига ўзгартирилиши керак. Бошланғич дастурни машина тилига ўгириш вазифасини махсус дастур-компилятор амалга оширади.

Компилятор асосан куйидаги иккита вазифани бажаради:

1. Бошланғич дастур матнидаги синтактик хатоларни текширади.
2. Ишчи дастур — машина кодини яратади.

Ишчи дастур бошланғич дастурда синтактик хатоликлар йўк бўлганда яратилади. Компилятор томонидан машина кодини яратилиши дастур матнида синтактик хатоликлар йўклигидан дарак беради. Дастур тўғри ишлаётганлигини дастурни тестлаш, яъни синаб кўриш ва олинган натижаларни тахлил килиш оркали текширилади. Масалан, квадрат тенгламани илдизларини аниклашда дискриминантни ҳисоблашда (масалан, дискриминант формуласида) хатолик мавжуд бўлса ва бу ифода синтактик жихатдан тўғри бўлсада дастур нотўғри ечимларни ҳосил қилади.

DELPHI ДАСТУРЛАШ ТИЛИ

Delphi дастурлаш мухитида дастурни ёзиш учун Delphi дастурлаш тили кўлланилади. Delphi тилидаги дастур одатда *операторлар* деб аталувчи *кўрсатмалар* кетма-кетлигидан иборат бўлади. Бир кўрсатма иккинчисидан нукта вергул (;) билан ажратилади.

Ҳар бир кўрсатма *идентификаторлардан* иборат бўлади. Идентификаторлар куйидагиларни билдиради:

Тил буйрукини (:=, if, while, for);
Ўзгарувчини;
ўзгармасни (бутун ёки каср сон);
арифметик (+, -,*,/) ёки мантикий (and, or, not) операцияни;
кисм дастурни (процедура ёки функцияни);
кисм дастур (procedure, function) бошиланиш ёки охири (end), блок бошланиши ва охирини (begin, end).

МАЪЛУМОТЛАР ТИПЛАРИ

Дастур турли типдаги маълумотлар устида амаллар бажаради: бутун ва каср сонлар, символлар, символлар сатри, мантиқий катталиклар.

БУТУН ТИП

Delphi тили бутун сонли маълумотларнинг еттита типини қўллаб - кувватлайди: shortint, smallint, Longint, Int64, Byte, word ва Longword. Бу типларнинг тавсифлари 1.1 жадвалда келтирилган.

Бутун типлар жадвал 1.1.

Тип	Диапазон	Формати
Shortint	-128-127	8 бит
Smallint	-32 768 - 32 767	16 бит
Longint	-2 147 483 648 - 2 147 483 647	32 бит
Int64	-2^{63} - 2^{63} - 1	64 бит
Byte	0-255	8 бит, ишорасиз
Word	0-65 535	16 бит, ишорасиз
Longword	0 - 4 294 967 295	32 бит, ишорасиз

Object Pascal тили Longint типига эквивалент бўлган Integer универсал бутун типни ҳам қўллаб-кувватлайди.

ҲАҚИҚИЙ ТИП

Delphi тили олтита ҳақиқий типни қўллаб-кувватлайди: Real48, single, Double, Extended, comp, Currency. Бу типлар бир-биридан қабул қиладиган қийматлар диапазони, рақамлар миқдори ва компьютер хотирасида сақлаш учун зарур бўладиган байт миқдори билан фарқ қилади. (1.2-жадвал).

Ҳақиқий (каср) типлар 1.2 жадвал.

Тип	Диапазон	Рақамлар миқдори	Байт
Real48	2.9×10^{-39}-1.7×10^{38}	11-12	06
Single	1.5×10^{-45}-3.4×10^{38}	7-8	04
Double	5.0×10^{-324} -1.7×10^{308}	15-16	08
Extended	3.6×10^{-4951} -1.1×10^{4932}	19-20	10
Comp	$2^{63}+1$ - 2^{63}-1	19-20	08

Currency	-922 337 203 685 477.5808 --922 337 203 685 477.5807	19-20	08

Delphi тили Double типига эквивалент бўлган ва нисбатан универсал бўлган Real хакиқий типни ҳам қўллаб-кувватлайди.

СИМВОЛЛИ ТИП

Delphi тили иккита символли типни қўллаб - кувватлайди: Ansichar ва Widechar:

Ansichar типи — бу ANSI кодлашдаги символлар бўлиб, уларга 0 ... 255 диапозондаги сонлар мос келади;

widechar типи — бу Unicode кодлашдаги символлар, уларга 0 ... 65 535 диапазондаги сонлар мос келади.

Object Pascal тили Ansichar га эквивалент бўлган Char универсал симолли типни ҳам қўллаб – кувватлайди.

САТР ТИПИ

Delphi тили учта сатрли типни қўллаб - кувватлайди: shortstring, Longstring

shortstring типи компьютер хотирасида статик жойлашувчи ва узунлиги 0 дан 255 та символгача бўлган сатрдан иборат бўлади;

Longstring типи хотирада динамик жойлашувчи сатр бўлиб, унинг узунлиги бўш хотира ҳажми билан чегаралади;

WideString типи хотирада динамик жойлашувчи сатр бўлиб, унинг узунлиги бўш хотира ҳажми билан чегаралади. WideString типли сатрнинг ҳар бир символи Unicode-символ хисобланади.

Delphi тилида сатр типини кўрсатиш учун string идентификатори қўлланилади. String типи shortstring типига эквивалент.

МАНТИҚИЙ ТИП

Мантиқий катталик иккита: True (чин) ёки False (ёлғон) кийматлардан биттасини кабул килади. Delphi тилида мантикий катталиклар Boolean типига тегишли бўлади.

ЎЗГАРУВЧИ

Ўзгарувчи-бу хотиранинг соҳаси бўлиб, унда маълумотлар жойлашади ва бу маълумотлар устида дастур амаллар бажаради. Аслида дастур

маълумотлар устида амаллар бажарганида хотира ячейкасидаги маълумотлар. яъни ўзгарувчилар билан ишлайди.
Дастур ўзгарувчига (хотира соҳасига) мурожаат этиши учун, масалан, формула бўйича ҳисоблашда боланғич маълумотларни олиши ёки натижаларни сақлаши учун ўзгарувчи ўз номига эга бўлиши керак. Ўзгарувчи номини дастурчининг ўзи белгилайди.
Ўзгарувчи номи сифатида лотин алфавити ҳарфлари, рақамлар ва баъзи махсус символлар кетма-кетлиги қўлланилади. Ўзгарувчи номидаги биринчи символ ҳарф бўлиши шарт. Ўзгарувчи номида пробел ишлатилмайди.
Delphi тили компилятори ўзгарувчи номидаги бош ва кичик ҳарфларни фарқламайди. Шунинг учун SUMMA, Summa ва summa номлари битта ўзгарувчини билдиради.
Ўзгарувчи номини унинг вазифасига мантиқан боғлиқ ҳолда белгилаш мақсадга мувофиқ. Масалан, анъанавий равишда $ax^2 + bx + c = 0$ шаклида ёзиладиган квадрат тенгламани коэффициентларини ва илдизларини сақлаш учун мўлжалланган ўзгарувчиларга a, b, c, x1 ва x2 номларини бериш мантиқан тўғри бўлади.
Бошқа мисол. Агар дастурда харид суммаси ва чегирма қийматини сақлаш зарур бўлса, у ҳолда бу ўзгарувчиларга қуйидаги номларни бериш мумкин:
JamiSumm ва Chegirma ёки UmumSumma ва Skidka.
Delphi тилида ҳар бир ўзгарувчи фойдаланилиши олдидан эълон қилиниши лозим. Эълон қилиш орқали фақат ўзгарувчининг мавжудлилиги эмас, балки унинг типи ва қабул қиладиган қийматлар диапозони ҳам аниқланади. Умумий ҳолда ўзгарувчини эълон қилиш кўрсатмаси қуйидагича:

Ном : тип;

Бу ерда:
ном -ўзгарувчи номи;
тип - ўзгарувчида сақланадиган маълумотнинг типи.
Мисол:

```
a : Real; b : Real; i : Integer;
```

Келтирилган мисолда иккита Real типли ва битта Integer типли ўзгарувчи эълон қилинган.
Дастур матнида, қоидага кўра, ҳар бир ўзгарувчини эълон қилиш алоҳида сатрда жойлашади.
Агар дастурда битта типга тегишли бир неча ўзгарувчи мавжуд бўлса, у ҳолда бу ўзгарувчиларнинг номларини битта сатрга вергуллар билан ажратган ҳолда жойлаб ва охирги ўзгарувчи номидан кейин тип кўрсатилади, масалан:

```
a,b,c : Real; x1,x2 : Real;
```

ЎЗГАРМАСЛАР

Delphi тилида иккита турдаги ўзгармас мавжуд: *оддий* ва *номли.*
Оддий ўзгармас-бу бутун ёки каср сон, символлар сатри ёки алоҳида символ, мантикий кийматдан иборат бўлади.

СОНЛИ ЎЗГАРМАСЛАР

Дастур матнида оддий константлар одатдаги кўринишда, яъни худди сонлар, масалан, математикада мисол ечишдаги каби ёзилади. Каср сонларни ёзишда бутун ва каср кисмини ажратиш учун нукта (.) ишлатилади. Агар ўзгармас манфий бўлса, у холда бевосита биринчи ракамдан олдин "минус" белгиси қўйилади.
Қуйида сонли ўзгармасларга мисол келтирилган:
1230.0
-524.03 0
Каср ўзгармаслар нуктаси ўзгарувчан сон кўринишида тасвирланиши ҳам мумкин. Нуктаси ўзгарувчан сон шаклида тасвирлаш ҳар бир сонни алгебраик формада тасвирлаш усулига асосланади. Бунда сон 10 дан кичик бўлган сон (мантисса) билан ўнинг даражаси (тартиби) кўпайтмаси шаклида тасвирланади. 1.3-жадвалда сонни одатдаги кўринишда, алгебраик кўринишда ва нуктаси ўзгарувчан шаклда ёзиш мисоли келтирилган.

Каср сонни ёзиш намуналари 1.3- жадвал.

Сон	Алгебраик шакли	Нуктаси ўзгарувчан шакли
1 000 000	$1x10^6$	1 .0000000000E+06
-123.452	$-1.23452x\ 10^2$	-1 .2345200000E+02
0,0056712	$5,6712x10^{-3}$	5,6712000000E-03

САТРЛИ ВА СИМВОЛЛИ ЎЗГАРМАСЛАР

Сатрли ва символли ўзгармаслар қўштирнок ичига олинади. Қуйида сатрли ўзгармасларга мисоллар келтирилган:
'Delphi дастурлаш тили', 'Delphi 7'
'2.4'
'Д'

Бу ерда, '2.4' ўзгармасига эътибор қилинг. Бу символли ўзгармас ҳисобланади. Яъни 2,4 сони эмас, балки "икки бутун ундан турт" сони тасвирланган сатрли ўзгармас ҳисобланади.

МАНТИҚИЙ ЎЗГАРМАСЛАР

Мантиқий ифода ёки чин ёки ёлғон қийматли бўлади. Чинга True ўзгармас, ёлғонга эса False ўзгармас мос келади.

НОМЛАНГАН ЎЗГАРМАС

Номга эга бўлган ўзгармас-бу ном (идентификатор) дастурда ўзгармаснинг ўрнига ишлатилади.
Номланган ўзгармас худди ўзгарувчилар каби фойдаланишдан олдин эълон қилиниши керак. Умумий ҳолда эълон қилиш инструкцияси қуйидагича кўринишга эга:
Ўзгармас = қиймат;
Бу ерда:
ўзгармас-ўзгармас номи;
қиймат-ўзгармас қиймати.
Номланган ўзгармаслар дастурда const сўзи билан бошланувчи ўзгармасларни эълон қилиш бўлимида эълон қилинади. Қуйида номланган ўзгармасларни (бутун, сатрли ва каср) эълон қилиш мисоли келтирилган:

```
const
Bound = 10;
Title = 'Югуриш тезлиги';
pi = 3.1415926;
```

Номланган ўзгармасни эълон қилгандан кейин дастурда ўзгармаснинг ўрнига унинг номи кўрсатилиши мумкин.
Ўзгарувчилардан фарқли равишда ўзгармасларни эълон қилишда типи аниқ кўрсатилмайди. Ўзгармас типи унинг кўриниши орқали аниқланади, масалан:

125 — бутун типли ўзгармас;
0.0 — ҳақиқий типли ўзгармас;
' бажариш' — сатрли ўзгармас;
' \' — символли ўзгармас.

ЎЗЛАШТИРИШ АМАЛИ

Ўзлаштириш амали асосий ҳисоблаш амалларидан бири ҳисобланади. Агар дастурда хисоблашни бажариш зарур бўлса, у ҳолда ўзлаштириш амали қўлланилади.
Ўзлаштириш амали бажарилиши натижасида ўзгарувчи киймати ўзгаради, унга киймат берилади.
Умумий ҳолда ўзлаштириш амали қуйидаги кўринишга эга:
ном : = ифода;
Бу ерда:

Ном - ўзгарувчи. Унинг қиймати ўзлаштириш амали бажарилиши натижасида ўзгаради;
: = - ўзлаштириш амали символи;
Ифода — бу ифода киймати ўзлаштириш амали символидан чап томонда номи кўрсталиган ўзгарувчига ўзлаштирилади.

Мисол:
Sur := Baho * Miqdor; Chegirma := 10; Found := False;

ИФОДА

Ифода операндлар ва операторлардан иборат бўлади. Операторлар операндлар ўртасида жойлашади ва операндлар устида бажариладиган амалларни билдиради. Ифода операндлари сифатида ўзгарувчилар, ўзгармаслар, функция ва бошқа ифодалар қўлланилади. Асосий алгебраик операторлар 1.4-жадвалда келтирилган.

Алгебраик операторлар 1.4 -жадвал.

Оператор	Амал
+	Қўшиш
-	Айириш
*	Кўпайтириш
/	Бўлиш
DIV	Бутун сонли бўлиш
MOD	Бўлишдаги қолдик

Ифодаларни ёзишда DIV ва MOD операторларидан ташқари барча операторлар ва операндлар ўртасида пробел қўймаса ҳам бўлади.

+, -, * ва / операторларини қўллаш натижаси ўз-ўзидан маълум. DIV оператори бир сонни бошқасига бўлишдаги бўлинманинг бутун қисмини аниқлашга имкон беради. Масалан, is DIV *i* ифодасининг киймати 2 га тенг.
MOD оператори, модул бўйича бўлиш, бир сонни бошқасига бўлишдаги қолдикни аниқлашга имкон беради. Масалан, 15 MOD 7 ифоданинг киймати 1 га тенг. Оддий ҳолларда ифода ўзгармас ёки ўзгарувчини ифодалайди.
Ифодага мисол:
123 0.001 i+1
A + B/C Summa*0.75 (B1+B3+B3)/3 Baho MOD 100
Ифода кийматини ҳисоблашда операгорлар турлича устиворлик (приоритет)га эга бўлишини инобатга олиш керак. *, /, DIV, MOD операторлари + ва – операторларига нисбатан юқори устиворликка эга.
Операторлар устиворлиги уларни бажарилиш тартибига таъсир қилади. Ифода кийматини ҳисоблашда биринчи навбатда юқори устиворликка эга бўлган операторлар бажарилади. Агар операторларнинг устиворлиги бир хил бўлса, у ҳолда дастлаб энг чапдаги оператор бажарилади.
Амалларни зарур бажарилиш кетма-кетлигини белгилаш учун қавслардан фойдаланилади, масалан:

(r1+r2+r3)/(r1*r2*r3)

Қавс ичига олинган ифода худди битта операнд сифатида қаралади. Яъни қавс ичидаги операндлар устида операциялар одатдаги усулда, бироқ қавсдан ташқаридаги операцияларга нисбатан олдин бажарилади. Қавслар қатнашган ифодаларни ёзишда қавсларни жуфтлиги назорат қилиниши керак. Яъни очилувчи қавслар сони ёпилувчи қавслар сонига тенг бўлиши керак. Қавслар жуфтлигининг бузилиши-ифодаларни ёзишдаги энг кўп содир бўладиган хатолик ҳисобланади.

ИФОДА ТИПИ

Ифода типи ифодада қатнашган операндлар типи ва улар устида бажариладиган амаллар турига боғлиқ. Масалан, иккита бутун сон устида қўшиш амали бажарилаётган бўлса натижа типи бутун сонли бўлиши маълум. Агар бунда операндлардан биттаси каср типли бўлса, у ҳолда хатто каср қисм 0 га тенг бўлса ҳам натижа каср типли бўлади.
Ифода типини аниқлаш жуда мухим. Ифода типини аниқлашда ўзгармас типи унинг кўриниши билан, ўзгарувчи типи эса уни эълон қилишда кўрсатилишини инобатга олиш зарур. Масалан, 1 ва 512- бутун (integer) типли, 1.0, 0.0 ва 3.2E-0.5 ўзгармаслар -ҳақиқий (real) типли.
1.5-жадвалда операндлар типи ва операторлар кўринишига боғлиқ равишда ифода типини аниқлаш қоидалари келтирилган.

Ифода типини аниқлаш қоидалари 1.5- жадвал.

*, +, -	Ақалли операндлардан биттаси real типли	Real
*, +, -	Иккала операнд ҳам integer типли	integer
/	real ёки integer	Доимо real
DIV, MOD	Доимо integer	Доимо integer

ЎЗЛАШТИРИШ АМАЛИНИНГ БАЖАРИЛИШИ

Ўзлаштириш амали қуйидагича бажарилади:

1. Аввал ўзлаштириш амали символидан ўнг томондаги ифода қиймати ҳисобланади.

2. Кейин ҳисобланган қиймат номи ўзлаштириш амали символидан чап томонда кўрсатилган ўзгарувчига ўзлаштирилади.

Масалан, қуйидаги амаллар бажарилиши натижасида:

i:=0; - i ўзгарувчи қиймати 0 га тенг бўлади;

a:=b+c; - a ўзгарувчи қиймати b ва c ўзгарувчилар қийматларининг йиғиндисига тенг бўлади.;

j :=j+1; - j ўзгарувчининг қиймати биттага оширилади.

Агар ифода типи мос келса ёки қабул қилувчи ўзгарувчи типига ўзгартирилса ифода тўғри деб ҳисобланади. Масалан, real типли ўзгарувчига типи real ёки integer бўлган ифода қийматини, integer типли ўзгарувчига эса типи фақат integer бўлган ифода қийматини ўзлаштириш мумкин.

Масалан, i ва n ўзгарувчилар integer типли, d ўзгарувчи эса real типли бўлса, у ҳолда қуйидаги ифодалар:

i:=n/10; i:=1.0;

нотўғри, қуйидаги ифода эса

d:=i+1;

тўғри ҳисобланади.

Компиляция вақтида ифода типининг қабул қилувчи ўзгарувчи типига мослиги текширилади. Агар ифода типи ўзгарувчи типига мос келмаса, у ҳолда компилятор қуйидаги хатолик ҳақидаги хабарни чиқаради:

Incompatible types ... and ...

Бу ерда кўп нуқта ўрнига ифода ва ўзгарувчи типи жойлашади. Масалан, агар n бутун типли ўзгарувчи бўлса, у ҳолда n: = m/2 ифода нотўғри ҳисобланади. Шунинг учун компиляция вақтида қуйидаги хабар чиқарилади:

Incompatible types 'Integer' and 'Extended'.

СТАНДАРТ ФУНКЦИЯЛАР

Кўп марта қўлланиладиган ҳисоблаш ва ўзгартиришларни бажариш учун Delphi тили дастурчини бир қатор стандарт функциялар билан таъминлайди.

Функция қиймати унинг номи билан боғлиқ. Шунинг учун функцияни ифода операнди сифатида қўллаш мумкин. Масалан. квадрат илдизни ҳисоблаш учун k:=Sqrt(n) ифодани ёзиш етарли. Бу ерда Sqrt квадрат илдизни ҳисоблаш функцияси, n-квадрат илидизи аниқланиши керак бўлган сон сақланадиган ўзгарувчи.
Функциялар параметрларининг типи ва қийматининг типи билан характерланади. Функция қиймати ўзлаштириладиган ўзгарувчи типи функция типига мос бўлиши керак. Худди шунингдек, функция параметрининг амалдаги қиймати, яъни функцияга мурожат этишда курсатиладиган параметр қиймати расмий параметр типига мос келиши керак.

МАТЕМАТИК ФУНКЦИЯЛАР

Математик функциялар (1.6-жадвал) турли математик ҳисоблашларни амалга оширишга имкон яратади.

Математик функциялар 1.6-жадвал.

Функция	Значение
Abs (n)	n сонининг абсолют қиймати
Sqrt (n)	n сонининг квадрат илдизи
Sqr (n)	n квадрати
Sin (n)	Синус n
Cos (n)	Косинус n
Arctan (n)	Арктангенс n
Exp(n)	Экспонента n
Ln(n)	Натурал логарифм n
Rardom(n)	0 ... n-1 диапазондаги тасодифий сон

Тригонометрик функциялардаги бурчак катталиги радианларда курсатилиши керак. Бурчак катталигини градусдан радианга ўзгартириш учун қуйидаги формула ишлатилади: (a*3.1415256) 180. бу ерда: а- бурчакни градусдаги ўлчами; 3.1415926 - логарифмик сон. 3.1415926 каср ўзгармас ўрнига стандарт бўлган PI номланган ўзгармасни курсатиш мумкин. Бунда

бурчак катталигини градусдан радианга ўтказиш ифодаси қуйидаги кўринишга келади: a*Pi/180.

ЎЗГАРТИРИШ ФУНКЦИЯЛАРИ

Ўзгартириш функциялари (1.7-жадвал) кўпинча маълумотларни киритиш ва чиқаришни таъминловчи жараёнларда қўлланилади. Масалан, мулоқат ойнасининг чиқариш майдонига (Label компоненти) real типли ўзгарувчи кийматини чиқариш учун бу сонни символлар сатрига ўзгартириш керак. Буни FloatToStr функцияси амалга оширади. Бу функция параметри сифатида кўрсатилган ифода кийматини сатрли кўринишга ўтказади.
Масалан. Label1.caption := FloatToStr(x) амали x ўзгарувчи кийматини Label1 майдонига чиқаради.

Ўзгартириш функциялари

1.7- жадвал.

Функция	Функция киймати
Chr(n)	Коди n га тенг бўлган символ
IntToStr (k)	Бутун k сонининг сатрли кўриниши
FloatToStr (n)	Ҳақиқий n сонининг тасвиридан иборат сатр
FloatToStrF(n, f , k,m)	Ҳақиқий n сонининг тасвиридан иборат сатр. Функциядан фойдаланишда: f — формат (тасвирлаш усули); k — аниқлик (рақамларнинг зарур умумий миқдори); m — ўнлик нуқтадан кейинги рақамлар миқдори
StrToInt (s)	Сатрли кўриниши s дан ибоарт бутун сони
StrToFloat (s)	Сатрли кўриниши s дан иборат ҳақиқий сон
Round (n)	Маълум қоидалар асосида n сонни яхлитлашдаги бутун сон
Trunc (n)	n сонининг каср қисмини ташлаб юборишда ҳосил бўлган бутун сон
Frac(n)	n сонининг каср қисмини билдирувчи каср сон
Int (n)	n ҳақиқий сонининг бутун қисмини ифодаловчи каср сон

ФУНКЦИЯЛАРНИ ҚЎЛЛАШ

Одатда функциялар ифода операнди сифатида қўлланилади. Функция параметри мос типли ўзгармас. ўзгарувчи ёки ифода бўлиши мумкин. Қуйида стандарт функциялар ва ўзгартириш функцияларни қўллашга мисоллар келтирилган.

```
n := Round((x2-x1)/dx);
x1:= (-b + Sqrt(d)) / (2*a);
m := Random(10);
baho := StrToInt(Edit1.Text);
Edit2.Text := IntToStr(100);
mes := 'x1=' + FloatToStr(xl);
```

МАЪЛУМОТЛАРНИ КИРИТИШ

Кўпинча дастурлар боланғич маълумотларни киритиш ойнасидан ёки таҳрирлаш майдонидан (Edit компоненти) кабул қилиб олади.

МАЪЛУМОТЛАРНИ КИРИТИШ ОЙНАСИДАН КИРИТИШ

Киритиш ойнаси-бу стандарт мулокот ойнаси бўлиб, InputBox функциясини қўллаш натижасида пайдо бўлади. InputBox функциясининг киймати - фойдаланувчи киритган сатрга тенг бўлади.

Умумий ҳолда inputBox функциясидан фойдаланиб маълумотларни киритиш амали қуйидагича кўринишга эга:

Ўзгарувчи := InputBox(*Сарлавҳа*, *Хабар*, *Қиймат*);

Бу ерда:

Ўзгарувчи–кийматини фойдаланувчи киритиши керак бўлган сатр типли ўзгарувчи;

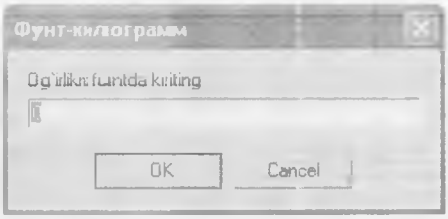
Сарлавҳа – киритиш ойнаси сарлавҳаси;

Хабар - тушунтирувчи хабар матни;

Қиймат-киритиш ойнаси экранда пайдо бўлганида киритиш майдонида тасвирланадиган матн.

Қуйида мисол сифатида оғирликни фунтдан килограммларга ўзгартирувчи дастур учун бошланғич маълумотларни олиш амали келтирилган. Бу амалга мос киритш ойнаси (1.5-расмда) тасвирланган.

```
s:=InputBox('Фунт-килограмм','Оғирликни фунтда киритинг','0');
```



1 5-расм. Киритиш ойнасига мисол.

Агар дастур ишлаш вақтида фойдаланувчи бирор сатрни киритиб, **OK** тугмасини танласа, у ҳолда inputBox функциясининг қиймати киритилган сатрга тенг бўлади. Агар **Cancel** тугмаси танланса, у ҳолда функция қиймати функцияга параметр сифатида берилган *қийматга* тенг бўлади.

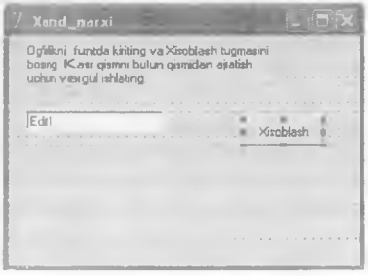
inputBox функциясининг қиймати сатр типли (string) бўлади. Шунинг учун, агар дастурга сонли маълумот киритилиши керак бўлса, у ҳолда бу функция қийматини мос ўзгартириш функцияси ёрдамида типини ўзгартириш керак. Масалан, оғирликни фунтдан килограммга ўтказувчи дастурнинг киритиш ойнасидан бошланғич маълумотларни қабул қилиб олишни таъминловчи қисми қуйидагича шаклда бўлиши мумкин:

```
s := InputBox('Фунт-килограмм','Оғирликни фунтда киритинг',''); funt :=
StrToFloat(s);
```

МАЪЛУМОТЛАРНИ ТАҲРИРЛАШ МАЙДОНИДАН КИРИТИШ

Таҳрирлаш майдони-бу Edit компонентидир. Таҳрирлаш майдонидан матн киритиш майдоннинг Text хоссасига мурожаат этиш орқали амалга оширилади. 1.6-расмда оғирлик ўлчовини фунтдан килограммга ўтказувчи дастурнинг мулоқат ойнаси кўриниши келтирилган. Бунда маълумоларни киритиш амали қуйидаги кўринишда бўлади:

```
Funt := StrToFloat(Edit1.Text);
```



1.6-расм. Edit1 компоненти маълумотларни киритиш учун қўлланилмоқда.

НАТИЖАЛАРНИ ЧИҚАРИШ

Кўпинча дастурларда натижалар хабар ойнасига ёки мулокат ойнасининг чикариш майдонига (компонент Label) чикарилади.

НАТИЖАЛАРНИ ХАБАР ОЙНАСИДА ЧИҚАРИШ

Хабар ойнаси фойдаланувчи диккатини тортиш учун кўлланилади. Хабар ойнаси ёрдамида дастур бошланғич маълумотлардаги хатоликларни хабар килиш ёки кайтариб бўлмас амалларни, масалан, файлни ўчиришни бажарилишини тасдиклаш учун кўлланилади.

Хабар ойнасини экранга ShowMessage процедураси ёки MessageDlg функцияси ёрдамида чикариш мумкин.

ShowMessage процедураси экранга матн ва **Ok** командали тугма жойлашган ойнани чикаради.

У мумий холда ShowMessage процедурасини чакириш инструкцияси куйидаги кўринишга эга:

ShowMessage(Хабар);

Бу ерда, *хабар*- ойнада тасвирланадиган матн.

1.7-расмда навбатдаги кўрсатма бажарилиши натижасида хосил бўлган хабар ойнаси тасвирланган:

ShowMessage(' Оғирликни фунтда киритинг.');

1.7-расм. Хабар ойнасига мисол.

ShowMessage процедураси ёрдамида чикарилган хабар ойнаси сарлавхасида **Project Options** ойнасининг **Application** катламида кўрсатилган илова номи жойлашади. Агар илова номи кўрсатилмаган бўлса, у холда хабар ойнаси сарлавхасида ишчи файл номи жойлашади.

MessageDig функцияси нисбатан универсал хисобланади. У хабар ойнасига стандарт белгилардан бирини, масалан, "Внимание" белгисини

жойлаштиришга. буйруқ тугмалар сонини ва типини кўрсатишга ҳамда фойдаланувчи қайси тугмани босганлигини аниқлашга имкон беради.

1.8-расмда навбатда келтирилган ифода бажарилиши натижасида ҳосил бўлган ойна тасвирланган:

```
r:=MessageDlg('Файл '+ FName + ' ўчирилади.', mtWarning, [mbOk,mbCancel] , 0)
;
```

1.8-расм. Хабар ойнасига мисол.

MessageDlg функциясининг қиймати-сондан иборат бўлиб, бу соннинг қийматини текшириб мулоқот қайси буйруқ тугмасини танлаб тугаганлиги аниқланади. Умумий ҳолда MessageDlg функциясига мурожаат қуйидагича кўринишга эга:

Танлов:= MessageDlg(Хабар, Тип, Тугмалар, Хабар матни)

бу ерда:

Хабар - хабар матни;

Тип - хабар типи. Хабар маълумот берувчи, огоҳлантирувчи ва критик хатолик ҳақидаги хабар бўлиши мумкин. Ҳар бир хабар типига маълум белги мос келади. Хабар типи номланган ўзгармас ёрдамида кўрсатилади (1.8-жадвал).

MessageDlg функцияси ўзгармаслари 1.8-жадвал.

mtWarning	Диққат	
mtError	Хато	
mtInformation	Маълумот	
mtConfirmation	Тасдиқлаш	
mtCustom	Одатдаги	Белгисиз

Тугмалар-хабар ойнасида тасвирланадиган тугмалар рўйхати. Бу рўйхат бир-биридан вергул билан ажратилган номланган ўзгармаслардан иборат бўлиши мумкин (1.9-жадвал). Тўла рўйхат квадрат қавс ичига олинади.

MessageDlg функцияси ўзгармаслари 1.9-жадвал.

Ўзгармас	Тугма	Ўзгармас	Тугма
mbYes	Yes	mb Abort	Abort
mbNo	No	mbRetry	Retry
mbOK	OK	mblgnore	Ignore
mbCancel	Cancel	mbAll	All
mbHelp	Help		

Масалан, хабар ойнасида **OK ва Cancel** тугмалари ҳосил бўлиши учун *Тугмалар* рўйхати қуйидагича бўлиши керак.

[mbOK,mbCancel]

Келтирилган ўзгармаслардан ташқари mbokcancel, mbYesNoCancel ва mbAbortRetrylgnore ўзгармасларини ҳам қўллаш мумкин. Бу ўзгармаслар хабар ойналарида кўп қўлланиладиган буйруқ тугмалар комбинацияларини ҳосил килади.

ХабарМатни-маълумотнома тизимининг бўлимини аниқловчи параметр. Агар фойдаланувчи <F1> клавишини танласа, экранда маълумотнома тизимининг жорий бўлими пайдо бўлади. Агар маълумотнома тизими маълумотларини чиқариш мўлжалланмаган бўлса, у ҳолда *ХабарМатн* параметри киймати 0 га тенг бўлади. MessageDig функцияси қайтарадиган қийматлар (1.10-жадвал) фойдаланувчи хабар ойнасидаги кайси тугмани танлаганлигини аниклашга имкон беради.

MessageDlg функцияси қийматлари. 1.10-жадвал.

MessageDig функцияси қийматлари	Мулокатни тугатган тугма
mrAbort	Abort
mrYes	Yes
mrOk	Ok
mrRetry	Retry
mrNo	No
mrCancel	Cancel
mrIgnore	Ignore
mrAll	All

МУЛОҚАТ ОЙНАСИ МАЙДОНИГА ЧИҚАРИШ

Мулоқат ойнасининг хабар чиқариш учун мўлжалланган қисмларидан бири чиқариш майдони ёки белги майдони деб аталади. Чиқариш майдони — Label компонентидир.
Чиқариш майдонидаги матн Caption хоссасининг қиймати орқали аниқланади. Caption хоссаси қийматини худди бошқа компонентлардаги каби илова формасини яратиш жараёнида ёки илова ишлаши вақтида кўрсатиш мумкин. Дастур ишлаши вақтида чиқариш майдонидаги матнни ўзгартириш, масалан, дастур ишининг натижасини чиқариш учун caption хоссасига янги қиймат ўзлаштириш зарур.
Caption хоссаси символ типли. Шунинг учун дастур ишлаши жараёнида бу майдонга сонли қийматларни чиқариш учун сонни сатрли типга ўзгартириш керак. Бунинг учун, масалан, FloatToStr ёки IntToStr функциясидан фойдаланиш мумкин.
Қуйида мисол сифатида оғирлик ўлчовини фунтдан килограммга ўтказувчи дастурдаги натижани чиқариш ифодаси келтирилган.

```
Label2.Caption:= FloatToStr(kg)+' кг';
```

ПРОЦЕДУРАЛАР ВА ФУНКЦИЯЛАР

Delphi тилида дастурлашда дастурчи иши асосан, ҳодисаларни қайта ишлаш процедураларини (қисм дастурларини) яратишдан иборат бўлади.
Ҳодиса рўй берганида жорий ҳодиса қайта ишлаш процедураси автоматик ишга тушади. Бу процедурани дастурчи ёзиши керак. Ҳодиса рўй берганида ҳодисага мос процедурани ишга тушириш вазифасини Delphi ўз зиммасига олади.
Object Pascal тилида асосий дастур бирилиги *қисм дастур* ҳисобланади. Қисм дастурни иккита тури фарқланади: *процедурлар* ва *функциялар*. Процедура ҳам, функция ҳам бирор ишни амалга оширувчи кўрсатмалар кетма- кетлигидан иборат бўлади. Қисм дастур кўрсатмаларини бажариш учун ушбу қисм дастурни чакириш зарур. Функциянинг процедурадан фарқли томони шундаки, функция номи бирор қиймат билан богланган. Шунинг учун функцияларни ифодаларда ишлатиш мумкин.

ПРОЦЕДУРАЛАР ТУЗИЛИШИ

Процедура ўз сарлавхаси билан бошланади ва ундан кейин қуйидагилар жойлашади:

Ўзгармасларни эълон қилиш бўлими;

янги типларни эълон қилиш бўлими;
ўзгарувчиларни эълон қилиш бўлими;
кўрсатмалар бўлими.

процедура қуйидаги умумий кўринишга эга:

```
procedure Номи (Параметрлар рўйхати);
const          // бу ерда ўзгармаслар эълон қилинади
type           // бу ерда var типлари эълон қилинади
               // бу ерда ўзгарувчилар эълон қилинади
begin
// бу ерда дастур кўрсатмалари жойлашади
end;
```

Процедурлар сарлавҳаси procedure сўзи ва ундан кейин жойлашадиган процедура номидан иборат бўлади. Процедура номи процедурани чақириш ва уни бажарилишини фаоллаштириш учун қўлланилади. Агар процедурада параметр мавжуд бўлса, у ҳолда улар процедура номидан кейин қавс ичида кўрсатилади. Процедурлар сарлавҳаси нуқта вергул (;) билан тугайди.

Агар процедурада номланган ўзгармаслар қўлланиладиган бўлса, у ҳолда улар **const** сўзи билан бошланувчи ўзгармасларни эълон қилиш бўлимида эълон қилинади.

Ўзгармаслар бўлимидан кейин **Type** сўзи билан бошланувчи типларни эълон қилиш бўлими бошланади.

Типларни эълон қилиш бўлимидан кейин ўзгарувчиларни эълон қилиш бўлими бошланади. Бу бўлимда дастурда зарур бўладиган барча ўзгарувчилар эълон қилинади. Ўзгарувчиларни эълон қилиш бўлими **var** сўзи билан бошланади.

Ўзгарувчиларни эълон қилиш бўлимидан кейин кўрсатмалар бўлими жойлашади. Кўрсатмалар бўлими **begin** сўзи билан бошланади ва **end** сўзи билан тамом бўлади. **End** сўзидан кейин нуқта вергул (;) жойлашади.

Кўрсатмалар бўлимида процедуранинг бажарилувчи кўрсатмалари жойлашади. Қуйида мисол сифатида харид нархини ҳисобловчи дастурнинг **Summa** процедураси келтирилган.

```
procedure Summa;

var     baho: real; // баҳо
miqdor: integer; // миқдор
s: real; // сумма
mes: string[255]; // хабар
begin
baho := StrToFloat(Form1.Edit1.Text);
miqdor := StrToInt(Form1.Edit2.Text);
```

```
s := baho * miqdor;  if s > 500 then
begin s := s * 0.9;
mes := 'Чегирма 10%'
+ #13; end: mes := mes+ 'Харид нархи: '
+ FloatToStrF(s,ffFixed,4,2) +' сум.';
Form1.Label3.Caption := mes; end:
```

ФУНКЦИЯ ТУЗИЛИШИ

Функциялар ҳам сарлавҳа билан бошланади. Сарлавҳадан кейин ўзгармасларни, типларни ва ўзгарувчиларни эълон қилиш бўлими ҳамда кўрсатмалар бўлими жойлашади. Функцияларни эълон қилишнинг умумий кўриниши:
function Номи (ПараметрларРуйхати) : Тип;
const // ўзгармасларни эълон қилиш бўлими
type // типларни эълон қилиш бўлими
var // ўзгарувчиларни эълон қилиш бўлими
begin // кўрсатмалар бўлими
result :=Қиймат; // функция номи билан қийматни боғлаш end;
Функция сарлавҳаси function сўзи билан бошланади ва бу сўздан кейин функция номи жойлашади. Функция номидан кейин қавс ичида параметрлар рўйхати келтирилади. Параметрлар рўйхатидан кейин икки нуқта (:) орқали функция қайтарадиган қиймат типи (функция типи) кўрсатилади. Функция сарлавҳаси нуқта кергул (;) билан тугайди.
Функция сарлавҳасидан кейин ўзгармаслар, типлар ва ўзгарувчиларни эълон қилиш бўлимлари жойлашади.
Кўрсатмалар бўлимида ўзгарувчилар бўлимида эълон қилинган ўзгарувчилар билан биргаликда result ўзгарувчисидан ҳам фойдаланиш мумкин. Функция кўрсатмалари бажарилиши тугаганидан кейин бу ўзгарувчи қиймати функция киймати ҳисобланади. Шунинг учун функция кўрсатмалари ичида result ўзгарувчисига қиймат ўзлаштирувчи кўрсатма бўлиши албатта зарур. Қоидага кўра, бу кўрсатма функция кўрсатмалари ичида энг охирда бажарилувчи кўрсатма ҳисобланади.
Қуйида мисол сифатида фунтни килограммга ўтказувчи FuntToKg функцияси келтирилган:

```
// оғирликни фунтдан килограммга ўтказиш
function FuntToKg( f:real):real;
const       // 1 фунт 409,5 граммга тенг.
K=0.4095; // ўзгартириш коэф.
begin
result:=f*K; end;
```

ДАСТУР КЎРСАТМАЛАРИНИ ЁЗИШ

Бир кўрсатма бошқасидан нукта вергул (;) билан ажратилади ёки бошқача айтса, ҳар бир кўрсатма охирида нукта вергул қўйилади.
Дастурнинг битта сатрида бир неча кўрсатмаларни жойлаштириши имкони бўлсада, қоидага кўра дастурнинг ҳар бир кўрсатмаси алоҳида сатрда жойлашади.
Баъзи кўрсатмалар (if, case, repeat, while ва бошқалар) бир неча сатрда жойлашади ва уларнинг қисмларини ажратиш учун сатрда бўш жой ташлаб ёзилади. Қуйида бир неча сатрда жойлашган кўрсатмага мисол келтирилган:

```
if d >= 0 then begin
x1:=(-b+Sqrt(d))/(2*a);
x2:=(-b-Sqrt(d))/(2*a);
ShowMessage('x1='+FloatToStr(x1) +
'x2='+FloatToStr(x2)) ;
End else ShowMessage('Тенглама ечимга эга эмас.');
```

Then ва else сўзлари бир-бирининг остида (сатр бошида бир хилда бўш жой қолдирилиб) қолдирилган бўш жой if сўзига боғлик. End сўзи begin сўзи остида жойлашади. Begin ва end орасидаги кўрсатмалар эса бир-бирининг пастида ва сатр бошидаги бўш жой Begin сўзига нисбатан қолдирилади.
Юқоридаги кўрсатмани қуйидагича ёзиш мумкин:

```
  if d >= 0 then begin
  x1:=(-b+Sqrt(d))/(2*a);
  x2:=(-b-Sqrt(d))/(2*a);
  ShowMessage('x1='+FloatToStr(x1)+'x2='+FloatToStr(x2));
end else ShowMessage('Тенглама ечимга эга эмас.');
```

Бироқ биринчи келтирилган вариант яхшироқ ҳисобланади, чунки у кўрсатма амалга оширувчи алгоритмни акс эттиради. Унда бир қарашда d >= 0 шарт қаноатлантирилганда бажариладиган кўрсатмалар (бунда x1 ва x2 ўзгарувчилар қиймати аниқланади) гуруҳини ва d >= 0 шарт қаноатлантирилмаганда бажариладиган кўрсатмани дарров кўриш мумкин.
Узун ифодалар ҳам бир неча сатрда жойлаштирилиши мумкин. Сатрнинг ихтиёрий қисмида ифодани бўлиб қолган қисмини янги сатрга ўтказиш мумкин. Бироқ бунда ўзгарувчи номларини, сонли ва сатрли ўзгармасларни, ҳамда мураккаб операторларни бўлиш мумкин эмас.
Қуйида ифодани бир неча сатрга ёзиш намунаси келтирилган:

```
  st:= 'Тенглама илдизлари'+ #13
    +'x1=' + FloatToStr(x1)+ #13 +'x2=' + FloatToStr(x2);
```

Компилятор "ортикча" пробеллар ва бўш сатрларни инобатга олмайди. Шунинг учун сатр бошидаги барча пробелларни инобатга олмайди.

Жумладан, бу кўрсатмаларни ёзишда сатр бошида бўш жой қолдиришга имкон беради. Арифметик ва мантиқий ифодаларни (шартларни), параметрлар рўйхатини ёзишда пробелларни қўйиш талаб этилмайди. Бироқ пробелдан фойдаланилса, дастур матнини осон қабул қилинади. Қуйидаги иккита ўзлаштириш кўрсатмаси вариантларини солиштиринг:

```
x1:=(-b+Sqrt(d))/(2*a);
```

ва

```
x1 := (-b + Sqrt(d)) / (2 * a);
```

Бунда иккинчи вариант осон қабул қилинади.

Дастурнинг ишлаш мантиқини тушуниб олишни осонлаштириш учун тушунтирувчи матн — *изоҳлардан* фойдаланиш керак. Умумий ҳолда изоҳлар фигурали қавсга олинади. Очилувчи фигурали қавс изоҳ бошини, ёпилувчи қавс охирини билдиради. Агар изоҳ бир сатрли ёки кўрсатмадан кейин жойлашса, у ҳолда изоҳдан олдин иккита ётиқ чизиқ белгиси (//) қўйилади.

Қуйида ўзгарувчиларни эълон қилиш бўлими келтирилган бўлиб, унда изоҳларни ёзишни иккала варианти ҳам қўлланилган:

```
var
{ тенглама коэффициентлари }
a:real; // иккинчи даражали ноъмалум
b:real; // биринчи даражали номаълум
c:real; // нолинчи даражали номаълум
{ тенглама илдизлари  } x1,x2:real;
```

НАЗОРАТ САВОЛЛАРИ

1. Дастур яратиш жараёни қандай босқичлардан иборат?
2. Алгоритмни яратиш босқичини тавсифланг.
3. Компиляция қандай жараён?
4. Дастур алгоритми блок-схемасида қандай геомгерик шакллар қўлланилади?
5. Делфи дастурлаш тилида маълумотларнинг қандай типлари мавжуд?
6. Ўзгарувчи нима?
7. Ўзгарувчи қандай номланади?
8. Ифодалар қандай тузилишга эга?
9. Ўзлаштириш амали қандай амалга ошади?
10. Қайси функциялар стандарт функция хисобланади?
11. Бошлангич маълумотларни киритишни қандай усуллари мавжуд?
12. Натижаларни чиқаришни қандай усуллари мавжуд?

II БОБ. МАССИВЛАР

Массив бу битта ном остида бир турга мансуб ўзгарувчилар тўпламини ўзида сақловчи ўзгарувчидир. Массивларни маълумот таркибига қараб битта турдаги маълумотларни, масалан, жадвал ва рўйхатларга нисбатан қўллаш жуда қулайдир.

МАССИВНИ ЭЪЛОН ҚИЛИШ

Массив бошқа ўзгарувчилар каби ўзгарувчиларни эълон қилиш бўлимида эълон қилинади. Умумий кўриниши қуйидагича бўлади:

Ном: array[пастки индекс.. юкори индекс] of тур

Бу ерда:

Ном- массив номи;

Array- эълон қилинаётган ном массив эканлигини англатувчи калит сўз;

пастки индекс, юқори индекс - массив элементлари сони, яъни массив элементлари индексининг ўзгариш диапазонини билдирувчи бутун сонлар;

тур- массив элементлари тури.

Массив эълон қилишга мисоллар:

```
temper:array[1..31] of real;
koef:array[0. .2] of integer;
name:array[1..30] of string[25];
```

Массив эълон қилганда ўзгармаслардан фойдаланиш жуда ҳам қулайдир. Номланган ўзгармас ўзгарувчиларни эълон қилиш бўлимидан олдин ўзгармасларни эълон қилиш бўлимида тавсифланади. Ўзгармасларни эълон қилиш **const** сўзи билан бошланади. Номли ўзгармасни эълон қилиш учун дастлаб ўзгармас номи ва тенг (=) белгисидан сўнг унга бериладиган қиймат ёзилади. Масалан, қиймати 10 га тенг nt номли ўзгармасни эълон қилиш учун const бўлимида nt=10 деб ёзиш керак. Ўзгармасни эълон қилгандан сўнг уни дастурнинг ихтиёрий қисмида ундан фойдаланиш мумкин. Қуйида келтирилган мисолда футбол чемпионатининг гуруҳ-қатнашувчи массивини эълон қилишда ўзгармасдан фойдаланилган:

```
const
NT = 18; // гуруҳ сони
SN = 25; // номлар узунлиги чегараси
    team: array[1..NT] of string[SN];
```

Дастурда массив элементидан фойдаланиш учун унинг номи ва элемент индексини квадрат қавсга олиш керак. Индекс сифатида бутун типли ўзгармас ёки ифода қўлланилиши мумкин. Масалан:

```
team [ 1] :='Пахтакор';
```

```
d := koef[l]*koef[l]-4*koef[2]*koef[1];
ShowMessage(name[n+1]);
temper[i] := StrToFloat(Edit1.text);
```

Массивни эълон қилаётган пайтнинг ўзида унинг қийматларини бериб ўтиш мумкин. Массивни эълон килиш жараёнида қийматларини беришнинг умумий кўриниши қуйидагича:

```
Ном:array[пастки индекс..юкори индекс] of тур=(рўйхат);
```

Бу ерда: рўйхат – вергул билан ажратилган массив элементлари. Масалан:

```
a: array[10] of integer = (0,0,0,0,0,0,0,0,0,0);
Team: array[1..5] of String[10]=
('Пахтакор', 'Навбаҳор','Нефтчи','Хоразм','Термиз');
```

Массив элементлари ўлчами билан рўйхатдаги элементлар сони бир хил бўлишига эътибор беринг. Акс ҳолда компилятор хатолик хақида хабар беради: Number of elements differs from declaration (элементлар сони эълон қилингандагига тўғри келмайди).

МАССИВЛАР УСТИДА АМАЛЛАР

Массивлар устида оддий амалларга қуйидагилар киради:

- массивни чиқариш;
- массивни киритиш;
- массивнинг энг кичик энг катта элементини қидириш;
- массивнинг берилган элементини қидириш;
- массивни тартиблаш.

МАССИВНИ ЧИҚАРИШ

Массивни чиқариш деганда унинг элементлари кийматини экранга (мулокат ойнасига) чиқариш тушунилади.

Агар дастурда массивнинг барча элементларини чиқариш керак бўлса, у ҳолда for буйруғидан фойдаланган қулайдир ва бу буйрукнинг санагичи сифатида массив элементининг индекси олинади. Мисол тариқасида массив элементлари қийматларини белги майдонида чиқаришни кўрсатувчи дастурнинг кўриниши 2.1-расмда келтирилган.

Дастур номерланган гуруҳлари номларини номерланган рўйхат қилиб чиқариб беради. Дастур матни 2.1- листингда келтирилган.

2.1- расм. Массивни чикариш дастурининг мулокат ойнаси.

2.1-листинг. Массивни чикариш дастури.

```
unit outar_;
interface
uses
Windows, Messages, SysUtils, Variants,
Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls;
Type TForm1 = class(TForm)
Button1: TButton;
Label1: TLabel;
procedure Button1Click(Sender: TObject);
private
```

```
{ Private declarations } public
{ Public declarations } end;
Var Form1: TForm1;
implementation
($R *.dfm}
const
NT = 5;
var
team: array[1..NT] of string[10] =('Paxtakor', 'Neftchi', 'Navbaxor', 'Xorazm',
'Guliston');
procedure TForm1.Button1Click(Sender: TObject);
var
st:string; // командалар рўйхати
i:integer; // индекс, массив элементи номери
begin
// формада тасвирлаш учун рўйхатни расмийлаштириш
for i:=1 to NT do st := st + IntToStr(i)+ ' '
+ team[i] + #13; // рўйхатни чиқариш
Label1.Caption := st;
end;
end.
```

МАССИВНИ КИРИТИШ

Массивни киритиш деганда дастурнинг ишлаш мобайнида фойдаланувчидан (ёки файлдан) массив элементларини қабул қилиш жараёни тушунилади. Массивнинг ҳар бир элементини киритиш учун киритиш майдонини яратиш зарур. Агар массив элементлари етарли даражада катта бўлса, у ҳолда бу ечим тўғри келмайди. Масалан, 15 та киритиш майдонли формани кўз олдингизга келтириб кўринг. Кўриниб турибдики, сонлар кетма-кетлигини ҳар бир сон алоҳида катакчада жойлашадиган жадвал кўринишида киритган маъқул. Қуйида массив элементларини компонентлар ёрдамида киритишни ташкил этишнинг 2 та усули кўриб чиқилади. StringGrid ва Memo ёрдамида.

StringGrid КОМПОНЕНТИДАН ФОЙДАЛАНИШ

Массив элементларини киритишда StringGrid компонентидан фойдаланиш жуда ҳам қулай бўлиб, бу компонент Additional саҳифасида жойлашган (2.2- расм).

2.2- расм StringGrid компоненти.

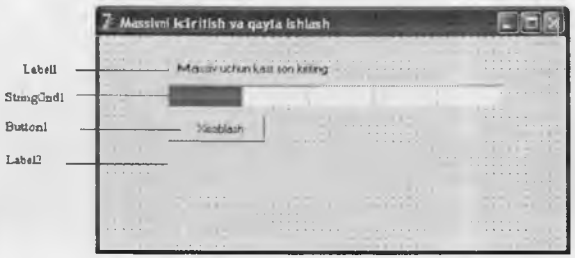
StringGrid компоненти ўзида хар бир катаги символлар сатридан иборат жадвални намоён қилади. 2.1-жадвалда StringGrid компонентнининг баъзи хусусиятлари келтирилган:

StringGrid компоненти хусусиятлари 2.1- жадвал.

Хусусият	Вазифаси
Name	Компонент номи. Компонент хусусиятларига эгалик қилиш учун ишлатилади.
ColCount	Жадвал устунлари сони.
RowCount	Жадвал қаторлари сони.
Cells	Жадвалга мос 2 ўлчамли массив. Col номерли устун ва row номерли қаторда жойлашган жадвалнинг катак элементи cells[row,col,]
FixedCols	Жадвалнинг чап томонидаги устунлар сони бўлиб, уни горизонт бўйлаб харакатлантирган пайтда ўз жойида колади.
FixedRows	Жадвалнинг юкори қисмидаги қаторлар сони бўлиб, уни вертикал бўйлаб харакатлантирган пайтда ўз жойида колади.
Options.goEditing	Жадвал элементини ўзгартириш мумкинлигини билдирувчи хусусият. Агар True бўлса ўзгартиришга рухсат этилган, False бўлса ман қилинган.
Options . goTab	Курсорни жадвал катаклари бўйлаб харакат килдириш учун <Tab> тугмачасидан фойдаланишга рухсат (True) ёки рухсат бермайди (False) .
Options. GoAlways-ShowEditor	Компонентни ўзгартириш жараёни мобайнида топа олиш хусусияти. Агар хусусият False бўлса катакчада курсор пайдо бўлиши учун F2 тугмасини босиб ёки сичконча билан битта туртиб матнни ёзиб бошлаш керак.
DefaultColWidth	Жадвал устунлари кенглиги.
DefaultRowHeight	Жадвал қаторлари баланлиги.

GridLineWidth	Жадвал катаги чегаралари чизигининг калинлиги.
Left	форма чап чегарасидан жадвалнинг чап чегарасигача бўлган масофа.
Top	форма юкори чегарасидан жадвалнинг юкори чегарасигача бўлган масофа.
Height	Жадвал майдони баландлиги.
Width	Жадвал майдони кенглиги.
Font	Жадвал элементини тасвирлаш учун ишлатиладиган шрифт.
ParentFont	Форма шрифтининг меросликбелгиси.

StringGrid компонентини ишлатишга мисол тариқасида массив элементлари кийматларини ўрта арифметигини ҳисобловчи дастурини кўриб чиқамиз. 2.3-расмда мулоқат ойнаси кўрсатилган. Бунда StringGrid компонентида массивни киритиш учун, Label1 ва Label2 компонентлари ҳисоблаш натижасини ва тушунтирувчи матн чиқариш учун, Button компоненти натижани ҳисоблаш учун ишлатилган.



2.3-расм. Массивни киритиш ва қайта ишлаш дастурининг ойнаси

StringGrid компоненти формага бошқа компонентлар каби қўшилади. Ундан сўнг бу компонентни 2.2-жадвалда кўрсатилган каби созлаш керак. Дастур матни 2.2-листингда келтирилган.

StringGrid компоненти хусусиятлари 2.2-жадвал

Хусусият	Қиймат
ColCount	5
FixedCols	0
RowCount	1

DefaultRowHeight	24
Height	24
DefaultColWidth	64
Width	328
Options.goEditing	True
Options.AlwaysShowEditing	True
Options.goTabs	True

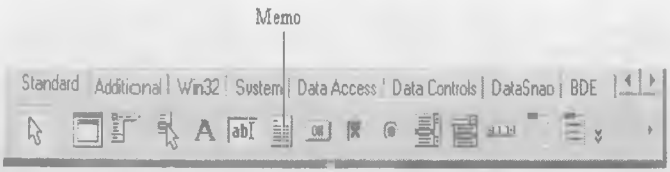
2.2-листинг. Массив бутун сонларини киритиш ва кайта ишлаш.

```
unit getar_;
interface
uses
Windows, Messages, SysUtils, Variants,
Classes, Graphics, Controls, Forms, Dialogs, Grids, StdCtrls;
type
TForm1 = class(TForm)
Label1: TLabel;
StringGrid1: TStringGrid;
Button1: TButton;
Label2: TLabel;
procedure Button1Click(Sender: TObject); private
{ Private declarations }
public
{ Public declarations }
end;
var
Form1: TForm1 ;
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject); var
a : array[1..5] of integer; // массив
summ: integer; // элементлар йиғиндиси
sr: real; // ўрта арифметик
i: integer; // индекс
begin
// массивни киритиш
// агар массив элементи бўш бўлса, уни нол деб оламиз
for i:= 1 to 5 do
if Length(StringGrid1.Cells[i-1. 0]) <>0
then a[i] := StrToInt(StringGrid1.Cells[i-1,0])
else a[i] := 0;
```

```
// массивни қайта ишлаш
summ := 0;
for i :=1 to 5 do
summ := summ + a[i]; sr := summ / 5;
// натижани чиқариш
Label2.Caption :='Элементлар йиғиндиси: ' + IntToStr(summ)+
+ #13+ 'Ўрта арифметик: ' + FloatToStr(sr);
end;end.
```

Memo КОМПОНЕНТИДАН ФОЙДАЛАНИШ

Айрим ҳолларда массивни киритиш учун Memo компонентидан фойдаланиш мумкин. Бу компонент катта сондаги сатрлардан иборат матнни киритишга имкон беради ва уни символли массив кўринишида ишлатиш жуда қулайдир. Memo компоненти формага одатдагидай қўшилади. Memo компоненти белгиси Standard саҳифасида жойлашган (2.4-расм).



2.4 расм. Memo компоненти.

Қуйидаги 2.3-жадвалда Memo компонентининг бир қанча хусусиятлари санаб ўтилган.

2.3-жадвал.

Хусусият	Вазифаси
Name	Компонент номи. Дастурда ушбу компонент хусусиятларига эгалик қилиш учун ишлатилади.
Text	Memo майдонида жойлашган матн. Битта бутун деб қаралади.
Lines	Memo майдонида жойлашган матн. Сатрлар тўплами деб қаралади ва ҳар бир сатрга мурожаат унинг номери буйлаб амалга оширилади.
Lines .Count	Memo майдонидаги сатрлар сони.
Left	Форманинг чап чегарасидан компонентнинг чап чегарасигача бўлган масофа.
Top	Форманинг юқори чегарасидан компонентнинг юқори чегарасигача бўлган масофа.
Height	Майдон баландлиги.

Width	Майдон кенглиги.
Font	Матннинг кўринадиган қисми учун ишлатиладиган шрифт.
ParentFont	Форма шрифтининг мерослик белгиси.

Массив элементларини киритишда Memo компонентидан фойдаланган пайтда ҳар бир элемент киймати алохида сатрда киритилиши керак, яъни ҳар бир массив элементи киритилгандан сўнг <Enter> тугмасини босиш керак. Memo майдонида жойлашган ҳар бир сатрга мурожаатни Lines хусусиятига квадрат қавс билан сатр номерини киритган ҳолда амалга ошириш мумкин (сатрлар 0 дан бошлаб номерланади).

2.3-листингда келтирилган дастур символли массивни Memo компоненти оркали киритишни намойиш қилади.

Memo компонентидан массив элементи кийматларини киритишнинг асосий цикли куйидагича амалга оширилади:

```
for i:=1 to SIZE do
a [ i ]:= Memo1.Lines[i];
```

бу ерда,

Size- массив ўлчамини аникловчи номланган ўзгармас;

а- массив

Memo1- Memo компоненти номи

Lines- Memo компоненти хусусияти булиб,ўзида Memo майдонидаги матннинг ҳар бир сатрни саклайди.

Дастур формаси 2.5-расмда келтирилган. Memo майдони ёнидаги тугмача босилиши билан Memo майдонидан массив кийматини ўзлаштирадиган буйрук тугмаси (Button1) мавжуд.

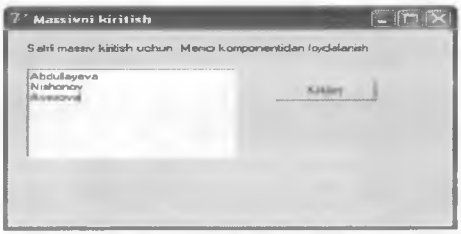
2.5-расм. Массивни киритиш дастурининг мулокат ойнаси.

2.3-листинг. Memo компоненти сатрларидан массивни киритиш.

```
unit fr_memo_; interface
uses
Windows, Messages, SysUtils, Classes,
Graphics, Controls, Forms, Dialogs, Menus, StdCtrls;
type
TForm1 = class(TForm)
Memo1: TMemo;
Button1: TButton;
Label1: TLabel:
procedure Button1Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;
var
Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1 .Button1Click(Sender: TObject);
const
SIZE=5;
// массив ўлчови
a:array[1..SIZE]of string[30];
//массив
n: integer;
 // Memo майдони сатрлари сони
i:integer;
// массив элементлари индекси
st:string;
begin
n:=Memo1.Lines.Count;
if n = 0 then begin
ShowMessage('Дастлабки маълумотлар киритилмаган!');
Exit;
// Ходисани қайта ишлаш жараёнидан чиқиш
end:
// Memo майдонида матн мавжуд
if n > SIZE then begin
ShowMessage('Сатрлар сони массив ўлчовидан ортик');
n:=SIZE;
```

```
/Дастлабки  SIZE та сатрни киритамиз
end;
for i:=1 to n do
a[i]:=Form1.Memo1.Lines[i-1];
// Memo сатри нолдан бошлаб номерланади
// Массив элементларини хабар дарчасида чикариш
if n  > 0 then begin
st:='Киритилган  массив:'+#13;
for i: =1 to n do
st:=st+IntToStr(i)+' '+ a[i]+f13; ShowMessage(st);
end;
end;end.
```

2.6-расмда массивни киритиш дастурининг кўриниши келтрилган. Киритиш тугмасини босиш билан Мемо майдон оркали киритилган массив элементлари кийматни намойиш этувчи ойна пайдо бўлади (2.7- расм).



2.6-расм. Массивни киритиш дастури ойнаси.

2.7-расм. Мемо майдонидан киритилган массив.

КЎП ЎЛЧОВЛИ МАССИВЛАР

Кундалик турмушда жадвал кўринишдаги маълумотлар билан тез-тез ишлашга тўғри келади. Масалан, автомобил савдоси билан шуғулланувчи бирор бир фирманинг иш фаолияти 2.4-жадвалда кўрсатилгандай тасвирланиши мумкин.

2.4-жадвал.

	Январь	Февраль	Март	...	Ноябрь	Декабрь
ВАЗ 2106						
ВАЗ 2107						
ВАЗ 2108						
ВАЗ 2109						

Жадвалнинг устунлари ва сатрлари қоида бўйича бир турдаги маълумотлардан иборат. Шунинг учун жадвал билан иш кўрадиган дастурларда массивлардан фойдаланиш жуда қулай. Демак, юқорида келтирилган жадвални қуйидаги бир ўлчовли масиивлар тўғлами ёрдамида тасвирлаш мумкин:

```
vaz2106: array [1..12] of integer;
vaz2107: array [1..12] of integer;
vaz2108: array [1..12] of integer;
vaz2109: array [1..12] of integer;
```

Ҳар бир массив ўзида мос ойда мос маркадаги автомобилдан сотилганларининг сонини сақлайди.

Бу жадвални қуйидаги кўринишда ҳам массив қилиб яратиш мумкин:

```
jan: array [1..6] of integer;
feb: array [1..6] of integer;
mar: array [1..6] of integer;
dec: array [1..6] of integer;
```

Бу ҳолда ҳар бир массив ҳар бир ойда сотилган автомобиллар сонини сақлашга мўлжалланган. яъни ҳар бир массив элементи битта маркадан қанча автомобил сотилганлиги ҳақида маълумотни сақлайди.

Агар жадвал битта турдаги маълумотни масалан, бугун сонларни сақласа у ҳолда бундай жадвални икки ўлчовли массив кўринишида тасвирлаш мумкин.

Икки ўлчовли массивни эълон қилишнинг умумий кўриниши қуйидаги куринишда бўлади:

Ном: array[пастка чегар1..юкори чегара1, пастки чегара2..юкори чегара2] of тур

Бу ерда:

Ном- массив номи;

Array - эълон килинаётган ном массив эканлигини англатувчи Delphi нинг калит сўзи;

Пастки чегара1..юқори чегара1, пастки чегара2..юқори чегара2- массив индекси оралиғи ва мос ҳолда массив элементлари сонини англатувчи бутун ўзгармаслар;

Тур- массив элементи тури

2.4-жадвал. Икки ўлчовли массив кўринишида куйидагича тасвирланиши мумкин:

itog: array [1..12, 1..6] of integer

2 ўлчовли массив элементлари сонини қуйидаги формула оркали ҳисоблаш мумкин:

(ЮЧ1-ПЧ1+1)х(ЮЧ2-ПЧ2+1)

Бунда:

ЮЧ1 ва ЮЧ2 – массив биринчи ва иккинчи индексларнинг юкори чегараси;

ПЧ1 ва ЧП2 – массив биринчи ва иккинчи индексларнинг пастки чегараси. Шундай экан itog массиви integer турдаги 60 элементдан иборат экан.

Массив элементини ишлатиш учун шу элементнинг индекслари кўрсатилиши керак. Одатда биринчи индекс жадвалнинг қатор номерига, иккинчи индекс эса устун номерига мос келади. Демак. itog[2,3] элементида март ойида (3 чи ой) ВАЗ 2107 маркали автомобилдан (ВАЗ 2107 автомобилидан сотилган автомобиллар ҳақидаги маълумот 2-қаторда жойлашади) қанча сотилганлиги ҳақида маълумот сақланади.

Жадваллар (массивлар) билан ишлаганда for буйругидан фойдаланиш қулай. Масалан, битта маркадаги автомобилдан йил бўйича қанча автомобил сотилганлигини аниқлайдиган дастур кисми қуйидагича:

```
s := 0;
for j := 1 to 12 do
s := s + itog[2,j];
```

Кейинги дастур кисми эса йил буйича сотилган барча автомобиллар сонини ҳисоблаб беради:

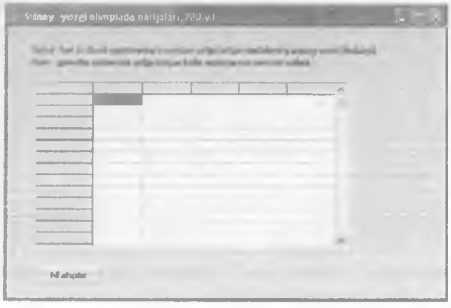
```
s:=0;
for i := 1 to 6 do // автомобилнинг 6 та маркаси
for j := 1 to 12 do //12 ой
 s := s + itog[i,j];
```

Мисол тариқасида 2000 йил Сиднейда бўлиб ўтган ёзги олимпиада мусобақаларининг натижаларини қайта ишловчи дастурни кўриб чиқамиз. Дастлабки маълумотлар 2.5-жадвалда келтирилган.

2000 йил Сидней олимпиадаси натижалари 2.5-жадвал.

Давлат	Олтин	Кумуш	Бронза
Австралия	16	25	17
Беларусь	3	3	11
Буюк Британия	11	10	7
Германия	14	17	26
Италия	13	8	13
Китай	28	16	15
Корея	8	9	11
Куба	11	11	7
Нидерландия	12	9	4
Россия	32	28	28
Руминия	11	6	9
США	39	25	33
Франция	13	14	11
Япония	5	8	5

Дастур ҳар бир давлат томонидан қўлга киритилган медаллар сонининг умумий сонини медалга мос баллга кўра яъни ҳар бир олтин медал учун 7 балл, кумуш учун 6 балл ва бронза учун 5 балл билан ҳисоблаши керак. Дастурнинг мулоқат ойнасининг кўриниши 2.8 - расмда келтирилган.



2.8-расм Олимпиада натижалари дастури мулоқат ойнаси.

Дастлабки маълумотларни киритиш ва тасвирлаш учун StringGrid компонентидан фойдаланилган ва унинг хусусиятлари 2.6-жадвалда келтирилган.

StringGrid компоненти хусусиятларининг қиймати 2.6-жадвал.

Хусусият	Қиймат
Name	Tab1
ColCount	6
RowCount	14
FixedCols	0
FixedRows	1
Options . goEditing	TRUE
DefaultColWidth	65
DefaultRowHeight	14
GridLineWidth	1

Жадвалнинг биринчи қатори жадвал устунларини номлаш учун хизмат қилади. Формани яратаётган пайтда cells массивининг элементлари қийматини ўрнатиш мумкин эмас, фақат дастур бажарилаётган вақтдагина массив элементларига мурожаат қилиш мумкин. Шунинг учун Cells массиви элементларини яъни жадвалнинг биринчи қаторини Форма фаоллашган пайтда содир бўладиган OnActivate ходисани қайта ишловчи процедурасида ўрнатиш лозим. Бундан ташқари бу процедурада жадвалнинг биринчи устуни яъни олимпиадада қатнашган давлатлар номи ҳам киритилади.

2.4-листинг. Жадвални ишга тушириш.

```
procedure TForm1.FormActivate(Sender: TObject); begin
tab 1.Cells[0,0] ='Давлат';
tab 1.Cells[1,0] ='Олтин';
tab 1.Cells[2,0] ='Кумуш';
tab 1.Cells[3,0] ='Бронза';
tab 1.Cells[4,0] ='Жами';
tab 1.Cells[5,0] ='Балл';
tab 1.Cells[0,1] ='Австралия';
tab 1.Cells[0,2] ='Белоруссия';
tab 1.Cells[0,3] ='Буюк Британия';
tab 1.Cells[0,4] ='Германия';
tab 1.Cells[0,5] ='Италия';
tab 1.Cells[0,6] ='Китай';
tab 1.Cells[0,7] ='Корея';
tab 1.Cells[0,8] ='Куба';
```

```
tab1.Cells[0,9] :='Нидерландия';
tab1.Cells[0,10]:='Россия';
tab1.Cells[0,11]:='АҚШ';
tab1.Cells[0,12]:='Франция';
tab1.Cells[0,13]:='Япония'; end;
```

Берилган жадвал маълумотларини қайта ишлаш буйруқ тугмаси "Натижалар" ни босгандан кейин чиқарилади (2.5-листинг).

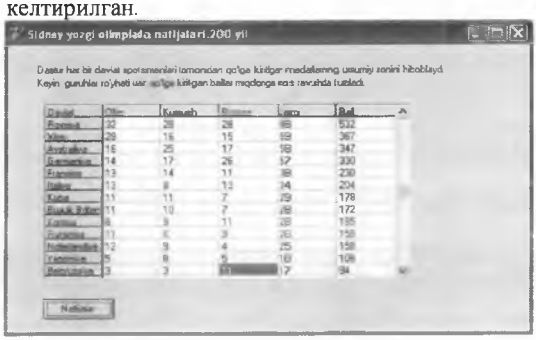
2.5-листинг. Икки улчовли массивни қайта ишлаш.

```
procedure TForm1.Button1Click(Sender: TObject);
var
c,r:integer; //жадвалнинг устун ва катор номерлари
s:integer; // гуруҳнинг медаллар сони
p:integer; // гуруҳ балли
m:integer; // энг куп балли гуруҳ қатор номери
buf:array[0..5] of string; // каторни алмаштириш буфери
i:integer; // катор номери. Тартиблашда фойдаланилади
begin
for r:=1 to tab1.rowcount do // барча каторни қайта ишлаш
begin s:=0;
// жами балларни хисоблаймиз
for c:=1 to 3 do
if tab1.cells[c,r] <>''
then s:=s+StrToInt(tab1.cells[c,r])
else tab1.cells[c,r]:='0'; // балларни хисоблаймиз
p:=7*StrToInt(tab1.cells[1,r])+
6*StrToInt(tab1.cells[2, r] )
+ 5*StrToInt(tab1.cells[3,r]};
// натижаларни чикариш
tab1.cells[4,r]:=IntToStr(s); // жами медаллар
tab1.cells[5,r]:=IntToStr(p); // баллар
end;
// жадвални баллар сонига кура камайиш тартибида тартиблаш
for r:=1 to tab1.rowcount-1 do
begin
m:=r; // r- каторда максимал элемент
for i:=r to tab1.rowcount-1 do
if StrToInt(tab1.cells[5,i])>StrToInt(tab1.cells[5,m])
then m:=i;
if r <> m then
begin // r –каторни m-катор билан алмаштириш
for c:=0 to 5 do begin
buf[c]:=tab1.Cells[c,r];
```

```
        tab1.Cells[c,r]:=tab1.Cells[c,m];
        tab1.Cells[c,m]:=buf[c];
        end;end;end; end;
```

Дастур дастлаб хар бир давлатнинг умумий медаллар сонини ва унга мос балларни хисоблайди. Кейин, хисобланган балларга кўра жадвални камайиш буйича тартиблайди. Тартиблаш мобайнида каторларни алмаштиришда индекси жадвал устунлари индекси билан бир хил buf сатрли массивидан фойлаланилади. Бу усул билан буфердаги каторни жадвалга кўчириш ва жорий жадвал каторини буферга олиш имконини беради.

Массивни кайта ишлаш дастурини тамомланишининг мулокат ойнаси 2.9-расмда келтирилган.



2.9-расм. Олимпиада Натижалари дастури ойнаси.

МАССИВДАН ФОЙДАЛАНИШДАГИ ХАТОЛАР

Массивдан фойдаланишда йўлга қўйиладиган энг кўп хатолардан биттаси бу массив индексининг уни эълон қилган вақтдаги индекс чегарасидан чиқиб кетишидир. Агар индекс сифатида ўзгармасдан фойдаланилса ва унинг қиймати чегарадан чиқиб кетса, бундай гурдаги хато дастурни компиляция қилиш мобайнида аниқланади. Масалан, дастурда массив қуйидагича эълон қилинган бўлса:

```
        Day: array[0..6] of string[11];
```

Компиляция мобайнида дастурда

```
day [7] := 'Якшанба';
```

қисми хато деб белгиланади.

Агар массив элементига мурожаат қилишда индекс сифатида ўзгарувчи ёки ифода ишлатилса, у холда хатолар дастур бажарилиш мобайнида рўй беради. Масалан дастурда массив қуйидагича эълон килинган бўлса,

tab1: array [1..N] of integer;

ва

for i:=0 to N do tab1[i] := 5; катори формал жиҳатдан тўғри ва компиляциядан муваффақиятли ўтади. Бирок дастур бажарилиши мобайнида tab массивига мавжуд бўлмаган нолинчи элемент ўзлаштирилса экранга хатолик ҳакида хабар чикарилади. Хабар дарчаси кўриниши дастурни кесрдан туриб бажарилаётганлигига боғлик.

Кўрилаётган дастур Delphi нинг ўзидан бажарилса, у холда 2.10-расмда келтирилгандай хабар чикарилади.

2.10-расм. Массивнинг мавжуд бўлмаган элементига мурожаат вактида чикариладиган хатолик хабари (Delphi муҳитида бажартирилган дастурда).

Агар дастур Windows дан бажартирилса у ҳолда экранга Range check error (диапазон бошкариш хатолиги) хабари чикарилади. Хабар сарлавҳасида хатолик юз берган дастур номи келтирилади (2.11-расм).

2.11-расм. Массивнинг мавжуд бўлмаган элементига мурожаат вактида чикариладиган хатолик хабари (windows дан бажарилтирилган дастурда).

НАЗОРАТ САВОЛЛАРИ

1. Массив кандай эълон килинади?
2. Массив устида кандай амаллар бажарилади?
3. Массив элементларининг кийматлари кандай киритилади?
4. StringGrid компаненти кандай хоссаларга эга?
5. Memo компаненти кандай хоссаларга эга?
6. Массивлардан фойдаланишда йўл кўйиладиган типик хатоликларга мисол келтиринг.

III боб. ПРОЦЕДУРА ВА ФУНКЦИЯЛАР

Дастур билан ишлаётган пайтда кўпинча дастурчи қандайдир кетма-кетлик дастурнинг турли кисмларида бир нечта марта такрорланаётганига эътибор килиши мумкин. Масалан, 3.1-листингда огирликни фунтдан килограммга ўтказадиган дастури келтирилган.

3.1-листинг. Оғирликни фунтдан килограммга ўтказиш.

```
unit Unit1;
interface
uses
Windows, Messages, SysUtils, Variants,
Classes, Graphics, Controls, Forms, Dialogs, StdCtrls;
type
TForm1 = class(TForm)
Label1: TLabel; // тушунтирувчи матн
Edit1: TEdit; // оғирликни фунтда киритадиган майдон
Button1: TButton; // ҳисоблаш тугмаси
Label2: TLabel; // натижани чиқариш майдони
procedure Button1Click(Sender: TObject);
procedure Edit1KeyPress(Sender: TObject;
var Key: Char); private
{ Private declarations } public
{ Public declarations }
end;
var
Form1: TForm1 ;
implementation
{$R *.dfm}
// ҳисоблаш тугмасининг босилиши
procedure TForm1.Button1Click(Sender: TObject);
var f : real; // фунтдаги оғирлик
kg : real; // килограммдаги оғирлик
begin
f := StrToFloat(Edit1.Text);
kg := f; * 0.4059;
Label2.Caption := Edit1.Text + ' ф. — это ' +
FloatToStrF(kg, ffGeneral, 4, 2} + 'кг.'; end;
// дастлабки маълумотларни киритиш майдонига
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
var
f : real; // фунтдаги оғирлик
 kg : real; // килограммдаги оғирлик
```

```
begin
if Key = Char(VK_RETURN) then
begin
f: = . StrToFloat(Editl.Text) ;
kg := f * 0.4059;
Label2.Caption := Editl.Text + ' ф. - бу ' +
FloatToStrF(kg, ffGeneral, 4, 2)+ 'кг.'.;
end;
end;
end.
```

Дастурда кодни икки марта такрорланишининг олдини олиш мумкин. Бунинг учун дастурнинг бир неча марта такрорланадиган қисмини қисм дастур сифатида лойихалаш ва уни қисм дастур сифатида чақирадиган қилиш керак.

3.2-листингда огирликни фунтдан килограммга ўтказадиган дастурда дастлабки маълумотларни ҳисоблаш ва натижани чиқариш битта қисм дастурга функция сифатида бирлаштирилган.

3.2-листинг. Оғирликни фунтдан килограммга ўтказиш (процедурадан фойдаланиш).

```
unit Onitl; interface
uses
Windows, Messages, SysUtils, Variants,
Classes, Graphics, Controls, Forms, Dialogs, StdCtrls;
type
TForm1 = class(TForm)
Label1: TLabel; // тушунтирувчи матн
Edit1: TEdit; // оғирликни фунтда киритиш майдони
Button1 : TButton; // ҳисоблаш тугмачаси
Label2: TLabel; // натижани чиқариш майдони
procedure Button1Click(Sender: TObject);
procedure EditlKeyPress(Sender: TObject;
var Key: Char); private
{ Private declarations } public
{ Public declarations } end;
var
Form1: TForm1;
implementation
{$R *.dfm}
// дастурчи процедураси
procedure FuntToKg;
var
f : real: // фунтдаги оғирлик
```

```
kg : real; // килограммдаги оғирлик
begin
f := StrToFloat(Form1.Edit1.Text);
kg := f * 0.4059;
Form1.Label2.Caption := Form1.Edit1.Text + ' ф.-бу
' +
FloatToStrF(kg, ffGeneral, 4, 2) + 'кг.';
end;
// щелчок на кнопке Вычислить
procedure TForm1.Button1Click(Sender: TObject);
begin
FuntToKg; // FuntToKg процедурасини чақириш
 end;
// дастлабки маълумотларни киритиш майдонига курсорни босиш
procedure TForm1.Edit1KeyPress(Sender: TObject;
var Key: Char);
begin
if Key = Char(VK_RETURN)
then FuntToKg; // FuntToKg процедурасини чақириш
 end;
end.
```

Қисм дастурдан фойдаланишнинг афзаллиги кўриниб турибди. Биринчидан, дастурда иккиланган коднинг йўқлиги дастурни яратиш қийинлигини камайтиради. отладка жараёни ва ўзгартириш киритишни осонлаштиради. Фараз қилинг, мазкур дастурда тушунтирувчи матнни ўзгартириш керак бўлсин. Қисм дастури йўқ дастурнинг ҳамма қисмини қараб чиқиш ва уларни ўзгартириш керак бўлади. Агар дастурда қисм дастурдан фойдаланган бўлса, у ҳолда фақатгина ушбу қисм дастурдаги матнни ўзгартириш етарли бўлади. Иккинчидан, дастурнинг ишончлилиги ошади. Шунга эътиборингизни қаратиш керакки, қисм дастур фақатгина коднинг иккиланган пайтида эмас, балки катта дастурларни ечганда ҳар бир масалани битта қисм дастур кўринишида ифодалаш қулайдир. Ушбу ҳолда дастурнинг «ўқилувчанлиги» сезиларли даражада ошади ва бунинг натижасида уни отладка қилиш осонлашади.
Қисм дастур-умумий масаланинг бир қисмини ечишга йуналтирилган кичкина дастурдир. Delphi тилида икки турдаги қисм дастур мавжуд-процедура ва функция.
Ҳар бир қисм дастур дастурда чақирилиши учун ўз номига эга бўлади. Функциянинг процедурадан фарқи шундаки, функциянинг номи унинг қиймати билан боғлиқдир ва шунинг учун функцияни ифода операндаси сифатида ишлатиш мумкин, масалан ўзлаштириш буйруғида.

Қоида бўйича қисм дастур параметрларга эга бўлиб улар формал (сохта) ва фактик (ҳақиқий) параметрларга ажратилади.
Функцияни эълон қилаётган пайтда кўрсатиладиган параметрлар сохта параметрлар деб аталади. Уни чақиришда кўрсатиладиган параметрлар ҳақиқий параметрлар дейилади.
Параметрлар маълумотларни қисм дастурга узатиш ва қисм дастурдан натижаларни олиш учун ишлатилади.

ФУНКЦИЯ

Функция - бу қисм дастур яъни номга эга буйруқлар кетма-кетлиги.
Функция буйруқларига ўтиш жараёни функцияни чақириш ёки функцияга мурожаат деб аталади. Чақирилган функция буйруқларидан дастур буйруқларига ўтиш жараёни функциядан қайтиш дейилади.
Функцияга мурожаатнинг умумий кўринишиш қуйидгича:

ўзгарувчи := Функция (Параметрлар) ;

бу ерда.

ўзгарувчи-функция ҳисоблаган қийматни ўзлаштириб олувчи ўзгарувчи номи;
функция-ўзгарувчи ўзлаштириб олиши керак бўлган функция номи;
параметрлар-функция қийматини ҳисоблаш учун керак бўладиган сохта параметрлар рўйхати.

Қуйидагиларга эътибор қилинг:

ҳар бир ўзгарувчи аниқ турдаги қийматни қайтариши керак;
функция қиймаги ўзлаштирилаётган ўзгарувчи тури функция тури билан бир хил бўлиши керак;
ҳар бир функциянинг тури ва параметрлар сони қатъий аниқланган бўлиши керак.

ФУНКЦИЯНИ ЭЪЛОН ҚИЛИШ

Функцияни эълон қилишнинг умумий кўриниши қуйидагича:

```
function Имя (параметр1 : тур1, ..., параметрК : турК) : Тур;
 var
// бу ерда ички ўзгарувчилар эълон қилинади
 begin
// функция буйруқлари
Ном := ифода;
 end;
```

бу ерда:

function-Delphi нинг махсус сўзи бўлиб, дастурчи яратаётган функция буйруғи эканлигини англатади;

ном-функция номи. Даструрдан функция буйрукларига ўтишни амалга оширади;

параметр-бу ўзгарувчи функция кийматини хисоблашда ишлатиладиган кийматдир. Бу параметрнинг оддий ўзгарувчидан фарки шундаки, ўзгарувчиларни эълон килиш кисмида эмас, балки функциянинг сарлавҳасидаги var деб бошланувчи кисмида эълон килинади. Параметр аник кийматни дастур бажарилаётган пайтда функцияни асосий дастурдан туриб чакирганда кабул килиб олади;

тур -дастур чакираётган функция кайтарадиган киймат туридир.

Мисол тарикасида 3.3-листингда isint ва isFloat функциялари келтирилган. Isisnt функцияси тахрирлаш майдонига киритилган сон мос клавиша символига асосан бутунлигини теширади. <Enter> ва <Backspace> клавишалари мумкин деб каралади. isFloat функцияси хам шунга ўхшаш бирок каср сонлар учун. isFloat функциясида иккита параметр мавжуд: босилган клавиша коди ва тахрирлаш майдонига киритилган символлар сатри.

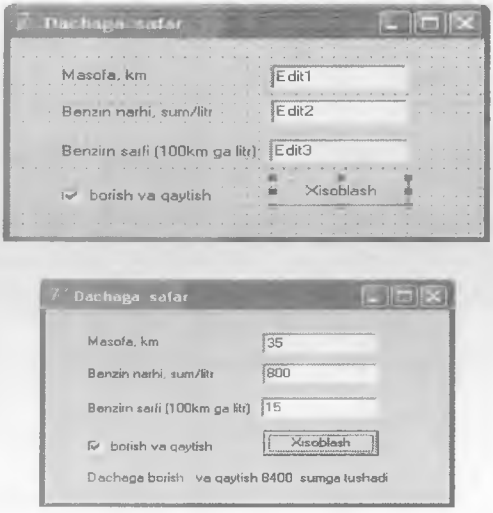
3.3-листинг. Функцияга мисоллар.

```
// бутун сонларни киритиш мобайнида киритилган символ мумкинми йўкми текширади
function Islnt(ch : char) : Boolean;
begin
if (ch >= '0'} and (ch <= '9') // сонлар
or (ch = 113) // <Enter> клавишаси
or (ch = #8) // <Backspace> клавишаси
then Islnt := True // символ мумкин
else Islnt := False; // символ мумкинмас
end;
// каср сонларни киритиш мобайнида киритилган символ мумкинми йўкми текширади
function IsFloat(ch : char; st: string) : Boolean;
begin
if (ch >= '0') and (ch <= '9') // сонлар
or (ch = #13) // клавиша <Enter>
or (ch = #8) // клавиша <Backspace>
then
begin
IsFloat := True; // тўғри символ
Exit; // функциядан чикиш
end;
case ch of
'-': if Length(st) = 0
```

```
        then IsFloat := True;
        if (Pos(',',st) = 0)
        and (st[Length(st)]>= '0') and (st[Length(st)] <= '9')
        then // ажратувчини фақат сондан кейин киритиш мумкин //ва фақат у
хали киритилмаган бўлса
        IsFloat := True; else // қолган символлар таъқиқланган
        IsFloat := False;
        end; end;
```

ФУНКЦИЯДАН ФОЙДАЛАНИШ

Агар сиз дастурда ўз функциянгиздан фойдаланмоқчи бўлсангиз, у ҳолда дастур матнида функцияни ишлатилмоқчи бўлган қисм дастурдан олдин эълон қилинг.



3.1- расм. Дачага саёҳат дастури ойнаси.

Кейинги дастур (дастур матни 3.4-листингда, дастур кўриниши эса 3.1- раемда келтирилган) дарчага саёҳат баҳосини ҳисоблаб беради. Дастурни дастлабки маълумотлари бўлиб масофа, 1 литр бензиннинг баҳоси ва 100 км

га бензин сарфи берилади. Дастлабки маълумотларни киритиш учун edit1, edit2, edit3 майдонларидан фойдаланилади. Ҳодисани қайта ишловчи OnKeyPress функцияси IsFloat киритилган маълумотларни филтрловчи функциядан фойдаланади.

3.4-листинг. Дастурчи функциясидан фойдаланишга мисол.

```
unit fazenda_;
interface
 Windows, Messages, SysUtils. Variants,
Classes, Graphics, Controls, Forms, Dialogs, StdCtrls;
type
TForm1 = class(TForm)
Edit1: TEdit; // масофа
Edit2: TEdit; // 1 литр бензин бахоси
Edit3: TEdit; // 100 кмга бензин сарфи
CheckBox1: TCheckBox; //True–бориш ва қайтиш
Button1: TButton; // ҳисоблаш тугмаси
Label4: TLabel; //ҳисоблаш натижасини чиқариш майдони
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
procedure Edit1KeyPress(Sender: TObject;
var Key: Char);
procedure Edit2KeyPress(Sender: TObject;
var Key: Char);
procedure Edit3KeyPress(Sender: TObject;
var Key: Char);
procedure Button1Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;
var
Form1: TForm1;
implementation
{$R *.dfm}
//символ мумкинми йўқми шуни текширади
function IsFloat(ch: char; st: string): Boolean;
begin
if (ch >= '0') and (ch <= '9') // сонлар
or (ch = #13) // клавиш <Enter>
or (ch = #8) // клавиш <Backspace>
```

```
then
begin
IsFloat := True; // тўғри символ
Exit; // функциядан чикиш
end; case ch of
'-': if Length(st) = 0 then IsFloat := True; ', ':
if (Pos(',',st) = 0)
and (st[Length(st)] >= '0') and (st[Length(st)] <= '9')
then
IsFloat := True/else
IsFloat := False;
end;
end;
// масофа майдонида клавишни босиш
procedure TForm1.EditlKeyPress(Sender: TObject;
var Key: Char);
begin
if Key = Char(VK_RETURN)
then Edit2.SetFocus //баҳо майдонига курсорни ўрнатиш
else
If not IsFloat(Key,Edit2.Text) then Key:= Chr(0); end;
// баҳо майдонида клавишани босиш
procedure TForml.Edit2KeyPress(Sender: TObject;
var Key: Char);
begin
if Key = Char(VK_RETURN)
then Edit3.SetFocus // истеъмол қилиш майдонига курсорни
жойлаштириш
else If not IsFloat(Key,Edit2.Text)
then Key := Chr (0);
end;
// истеъмол майдонида клавишани босиш
procedure TForml.Edit3KeyPress(Sender: TObject;
var Key: Char);
begin
if Key = Char(VK_RETURN)
then Buttonl.SetFocus //
// ҳисоблаш тугмасини
фаоллаштириш
else If not IsFloat(Key,Edit2.Text) then Key := Chr
(0);
end;
```

```
// ҳисоблаш тугмасига босиш
procedure TForm1.Button1Click(Sender: TObject);
var
rast : real; // масофа
cena : real; // бахо
potr : real; // 100 км га талаб
summ : real; // миқдор
mes: string;
begin
rast := StrToFloat(Edit1.Text);
cena := StrToFloat(Edit2.Text);
potr := StrToFloat(Edit3.Text);
summ := rast / 100 * potr * cena;
if CheckBox1.Checked then summ:= summ* 2;
mes := 'дачага бориш';
if CheckBox1.Checked then mes := mes + ' ва қайтиш';
mes:=mes+  FloatToStrF(summ,ffGeneral,4,2)+'сумга тушади.';
Label4.Caption := mes;
end;
end.
```

ПРОЦЕДУРА

Процедура-бу қисм дастурнинг яна бир тури ҳисобланади. Одатда қисм дастур қуйидаги ҳолларда процедура тарзида қўлланилади:

- қисм дастур асосий дастурга ҳеч қандай қиймат қайтармайдиган ҳолларда. Масалан, мулоқат ойнасида бирор бир график чизади;
- қисм дастур ўзи чақиртирилган дастурга бирдан ортиқ қиймат қайтариши керак бўлган ҳолларда. Масалан, квадарат тенгламани ечадиган қисм дастур ўзи чақиртилган дастурга иккита сон-тенглама илдизларини қайтариши керак.

ПРОЦЕДУРАНИ ЭЪЛОН ҚИЛИШ

Процедурани эълон килишнинг умумий кўриниши куйидагича:

```
procedure Ном (var параметр1: тур1; ... var параметрК: турК) ; var
// ички ўзгарувчилар эълон килинади
begin
// процедура буйруклари
end;
```

бу ерда.

Procedure-Delphi тилининг калит сўзи бўлиб, процедура эълон қилинаётганлигини англатади;

ном-процедурани чақиришда ишлатиладиган процедура номи;

параметр К-соҳта параметр бўлиб, процедура буйругида ишлатиладиган ўзгарувчи. Параметрдан олдинги var сўзи шарт эмас, бироқ агар у мавжуд бўлса, процедурани чақираётган пайтда унга мос ҳақиқий параметр бўлиши шарт.

Процедура параметрлари маълумотларни процедурага қабул қилиш учун ҳамда дастурга процедурадан қиймат қайтариш учун хизмат қилади.

Мисол тариқасида 3.5-листингда квадрат тенгламани ечувчи ($ax^2 + bx + c = 0$ тенгламани) процедура келтирилган. Процедурада 6 та параметр мавжуд бўлиб, биринчи 3 таси дастлабки маълумот тенглама коэффициентларини хисоблаб процедурага узатиш учун, x1 ва x2 параметрлар эса натижани қайтариш учун, Ok параметри тенглама ечимга эгалиги ҳақида маълумот бериш учун мўлжалланган.

3.5-листинг. SgRoot процедураси.

```
// квадрат тенгламани ечади
procedure SqRoot(a,b,c : real;
var xl,x2 : real;
var ok : boolean);
{ a,b,c —тенглама коэффициентлари x1,x2 — тенглама илдизлари ok = True — ечимга эга ok = False — ечимга эга эмас }
var
d : real; // дискриминант
begin
d:= Sqr( b) - 4*a*c; if d < 0 then
ok := False // тенглама ечимга эга эмас
else
begin
ok := True;
x1 :=(-b + Sqrt(d)) / (2*a) ; x2 := (b + Sqrt(d)) / (2*a);
end; end;
```

ПРОЦЕДУРАДАН ФОЙДАЛАНИШ

Ишлаб чиқилган процедурани implementation бўлимида, яъни процедурани ишлатувчи қисм дастур олдида жойлаштириш керак. Процедурани чақириш буйруғининг умумий кўриниши қуйидагича:

Ном(параметрлар рўйхати);

бу ерда,

ном-чакирилаётган процедура номи;

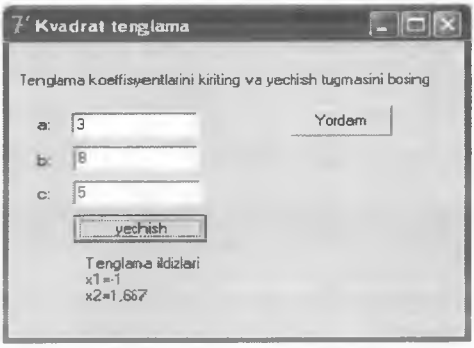
параметрлар рўйхати – вергул билан ажратилган ҳақиқий параметрлар.

Ҳақиқий параметр бу процедурада соҳта параметр сифатида эълон қилинган турга мос ҳолдаги ўзгарувчи, ифода ёки ўзгармас ишлатилиши мумкин.

Масалан, юқорида келтирилган квадрат тенгламани ечувчи процедурани чақириш қуйидагича бўлиши мумкин:

SqRoot(StrToFloat(Edit1.Text), StrToFloat(Edit2.Text), StrToFloat(Edit3.Text), k1,k2,rez);

3.2-листингда SqRoot процедурасидан фойдаланиб квадарти тенгламани ечувчи дастур матни келтирилган. Дастур ойнаси 3.2-расмда келтирилган.



3.2-расм. Квадрат тенгламани ечиш дастури ойнаси.

3.6-листинг. Квадрат тенгламани ечиш (процедурадан фойдаланиб).

```
unit SqRoot_; interface
uses
Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms, Dialogs, StdCtrls;
type
TForm1 = class(TForm)
Edit1: TEdit;
Edit2: TEdit;
Edit3: TEdit;
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
```

```
Label4: TLabe1;
Button1: TButton;
Label5: TLabel;
procedure ButtonlClick(Sender: TObject); private
{ Private declarations }
public
{ Public declarations }
end;
var
Form1: TForm1;
implementation
{$R *.dfm}
// Квадрат тенгламани ечиш
procedure SqRoot(a,b,c : real; var xl, x2 : real; var ok : boolean);
{ a,b,c -тенглама коэффициентлари, x1,x2 -тенглама илдизлари
ok = True - ечим бор ok = False - ечим йўқ }
var
d : real; // дискриминант begin
d:= Sqr(b) - 4*a*c; if d < 0 then
ok := False // тенглама ечимга эга эмас
else
begin
ok := True;
xl := (-b + Sqrt(d)) / (2*a); x2 := (b + Sqrt(d)) / (2*a) ;
end;
end;
procedure TForml.ButtonlClick(Sender: TObject);
var
k1,k2: real; // тенглама илдизлари
rez: boolean; // True —ечим бор, False —ечим йўқ mes:
string; // хабар begin
SqRoot(StrToFloat(Editl.Text), StrToFloat(Edit2.Text) ,
StrToFloat(Edit3.Text) , k1,k2,rez);
if rez then
mes := 'Тенглама ечимлари + #13 +
'x1='+FloatToStrF(kl,ffGeneral,
4,2)+#13+ 'x2='+FloatToStrF(k2,ffGeneral,4,2)+#13 else
mes := 'Тенглама ечимга эга эмас; labels.Caption := mes;
end;
end.
```

ФУНКЦИЯ ВА ПРОЦЕДУРАДАН ҚАЙТА ФОЙДАЛАНИШ

Дастурчи бирор бир функцияни ишлаб чиқиб, унинг матнини дастурнинг implementation бўлимида жойлаштирган ҳолда уни бошқа бир дастурда ишлатиши мумкин. Лекин шуниси ноқулайки, дастурчи функция матнини қайтадан киритиши ёки нусхасини кўчириши талаб қилинади.

МОДУЛ ЯРАТИШ

Delphi тили дастурчига ўз функция ва процедураларини алоҳида модулга жойлаштириш ва кейин ўз дастурларида модуллар рўйхатида модул номи кўрсатган ҳолда ушбу модул функция ва процедураларидан фойдаланиш имконини беради.

Модул яратишни бошлашдан олдин дастлаб форма ойнаси ва модул ойнасини ёпиш керак. Кейин File менюсидан New/Unit буйруғини танлаш керак. Натижада экранда Delphi нинг расмийлаштирилган модул шаблонининг код таҳрирловчи ойнаси пайдо бўлади. Унинг матни 3.7-листингда келтирилган.

3.7 листинг. Модул шаблони.

```
unit Unit1;
interface implementation
end.
```

Модул сарлавҳаси модул номини кўрсатувчи Unit буйруғи билан бошланади. Модулни сақлаш жараёнида дастурчи кўрсатган ном автоматик тарзда модул номига ўзлаштирилади.

Interface сўзи модул интерфейси бўлимини англатади. Ушбу бўлимда дастурчи модулда мавжуд процедура ва функцияларни эълон қилади. Implementation (амалга ошириш) бўлимда эса interface бўлимида эълон қилинган процедура ва функциялар жойлаштирилади. Мисол тариқасида 3.7- листингда олдин кўриб чиқилган IsInt isFloat функциялардан иборат модул келтирилган

3.8 - листинг. Дастурчи модули.

```
unit my__unit;
interface // ушбу модулда мавжуд процедура ва функцияларни эълон қилиш
function IsInt(ch : char) : Boolean;
// Islnt символ бутунми йўқлигини текширувчи функция
function IsFloat(ch : char; st: string) : Boolean;
// IsFloat функцияси киритилган символ мослигини текширади
// ch —кейинга символ
// st —киритилган символлар
```

```
implementation // амалга ошириш
function IsInt(ch : char) : Boolean;
begin
if (ch >= '0') and (ch <= '9') // рақамлар
or (ch = #13) // <Enter> клавишаси
or (ch = #8) // <Backspace> клавишаси
then IsInt := True // символ руҳсат этилган
else IsInt := False; // руҳсат этилмаган символ
end;
function IsFloat(ch : char; st: string) : Boolean;
// ch - навбатдаги символ // st - киритилган символлар
begin
if (ch >= '0') and (ch <= '9') // рақамлар
or (ch = #13) // <Enter> клавиши
or (ch = #8) // <Backspace> клавиши
then
begin
IsFloat := True; // символ тўғри
Exit; // функциядан чиқиш
end; case ch of
'-': if Length(st) = 0 then IsFloat := True; ',':
if (Pos(',',st) = 0)
and (st[Length(st)] >= '0') and (st[Length(st)] <= '9')
then // ажратувчини фақат рақамдан кейин киритиш мумкин
// ва у агар киритилмаган бўлса
IsFloat := True; else // бошқа символлар тақиқланган
IsFloat := False; end
// бу амалга ошириш бўлими // жорий ҳолатда бу бўлимда инструкциялар
мавжуд эмас
end.
```

Модулни сақлаш, оддий, яъни File менюдан Save буйруғи танланади. Модулдан қайта фойдаланиш учун яхшиси Units номли папка яратиб ўша жойга сақлаган маъқул.

МОДУЛДАН ФОЙДАЛАНИШ

Дастурчи ўз дастурида модул функция ва процедураларидан фойдалана олиши учун лойихага фойдаланилаётган модуллар рўйхатига модул номини қўшиши талаб қилинади.

3.9-листингда **дачага бориш** дастурининг варианти келтирилган. OnKeyPress ходисасини қайта ишловчи процедураси my_unit.pas модулида

жойлашган isFloat функциясига мурожаат килади, шунинг учун хам фойдаланиладиган модуллар рўйхатида my_unit кўрсатилган.

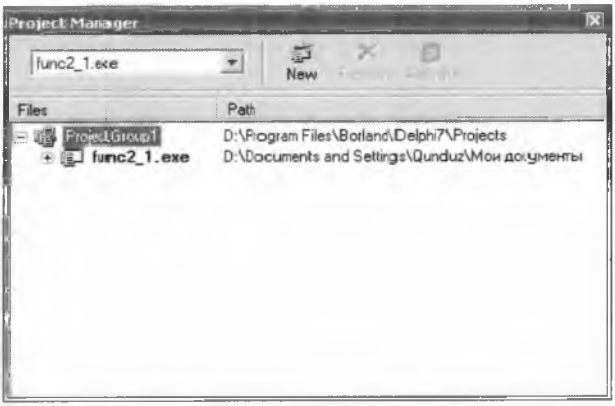
3.9-листинг. Дастурчи модулидан функцияни ишлатиш.

```
unit fazenda_;
interface
uses
Windows, Messages, SysUtils, Variants,
Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, my_unit; // дастурчи модули
type
TForm1 = class(TForm)
Edit1: TEdit; // масофа
Edit2: TEdit; // бир литр бензин нархи
Edit3: TEdit; // 100 км га сарфланадиган бензин
CheckBox1: TCheckBox; // True — бориш ва қайтиш
Button1: TButton; // Ҳисоблаш тугмаси
Label4: TLabel; // Ҳисоблаш натижасини тасвирлаш майдони
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
procedure Edit1KeyPress(Sender: TObject;
var Key: Char);
procedure Edit2KeyPress(Sender: TObject;
var Key: Char);
procedure Edit3KeyPress(Sender: TObject;
var Key: Char);
procedure Button1Click(Sender: TObject);
private
{ Private declarations} public
{ Public declarations } end;
var
Form1: TForm1;
implementation
{$R *.dfm}
// Масофа майдонида клавиш босилиши
procedure TForm1.Edit1KeyPress(Sender: TObject;
var Key: Char);
begin
if Key = Char(VK_RETURN)
then Edit2.SetFocus // курсорни баҳо майдонига жойлаш
else If not IsFloat(Key.Edit2.Text)
```

```
then Key := Chr(0);
end;
// Баҳо майдонида клавиш босиш
procedure TForm1.Edit2KeyPress(Sender: TObject; var Key: Char);
begin
if Key = Char(VK_RETURN)
then Edit3.SetFocus // курсорни Сарф майдонида жойлаш .
else If not IsFloat(Key.Edit2.Text) then Key := Chr(0);
end;
// Сарф майдонида клавиш босиш
procedure TForm1.EditSKeyPress(Sender: TObject:
var Key: Char);
begin
if Key = Char(VK_RETURN)
then Button1.SetFocus // // Ҳисоблаш тугмасини фаоллаштириш
else If not IsFloat(Key.Edit2.Text) then Key := Chr(0);
end;
// Ҳисоблаш тугмасини танланиши
procedure TForml.ButtonlClick(Sender: TObject);
var
rast : real; // масофа
cena : real; // баҳо
potr : real; // 100 км даги сарф
summ : real; // сумма
mes: string;
begin
rast := StrToFloat(Editl.Text) ;
cena := StrToFloat(Edit2.Text);
potr := StrToFloat(Edit3.Text);
summ := rast / 100 * potr * cena;
if CheckBoxl.Checked then summ := summ * 2;
mes := 'Дачага бориш;
if CheckBox1.Checked then
mes : = mes + ' ва аксинча ;
mes := mes + 'га тенг бўлади '
+ FloatToStrF(summ,ffGeneral, 4,2) + ' руб.';
Label4.Caption := mes;
end;
end.
```

Дастурда ишлатиладиган модуллар рўйхатига модул номини кушгандан сунг модулнинг ўзини ҳам лойихага қўшиш керак бўлади. Буни

учун Project менюсидан **Add to Project** буйруғини танлаш ва очилган мулокат ойнасидан модул файли номини кўрсатиш керак.
Лойиха таркибини кўриш **View** менюсининг мос буйруғи оркали **Project Manager** ойнасида амалга оширилади.



3.3-расм. **Project Manager** ойнасида лойиха таркиби акс эттирилади.

Модулни лойиҳага қўшиш ва фойдаланиладиган модуллар рўйҳатига унинг номини киритгандан сўнг дастур компиляция килиниши мумкин.

НАЗОРАТ САВОЛЛАРИ

1. Функция қандай эълон килинади?
2. Дастурда функцияга қандай мурожаат килинади?
3. Процедура нима?
4. Процедура қандай эълон килинади?
5. Модул нима? Модул қандай яратилади?

IV боб. Файллар

ФАЙЛНИ ЭЪЛОН ҚИЛИШ

Файл бу шундай маълумотлар тўпламики у ўзида бир турдаги элементлари сони чекланмаган миқдордаги маълумотларни сақлайди. Ҳозирча биз уни ўлчами чекланмаган массив деб қарашимиз мумкин. Бу маълумот тури бошқа барча маълумот турлари каби (ўзгарувчи, массив) файл ҳам ўзгарувчиларни тавсифлаш бўлимида эълон килиниш керак. Файлни эълон қилишда файл элементи тури кўрсатилади.

Умумий кўриниши қуйидагича:

Ном: file of элемент тури

Масалан.

res: file of char; // символли файл
koef: file of real; // хакикий сонли файл
f:file of integer; // бутун сонли файл

Компонентлари символли турдан иборат файл символли ёки матнли файл деб аталади. Матн файлини эълон қилиш қуйидагича амалга оширилади:

Ном: TextFile;

бу ерда.

ном- файл типли ўзгарувчи номи;
TextFile- ном остидаги файлли ўзгарувчи матн файли эканлигини англатади.

ФАЙЛНИ ФАОЛЛАШТИРИШ

Файлли ўзгарувчини эълон қилиш фақат унга турни аниқлаб беради. Дастур маълумотларни файлдан чиқариб олиш ёки файлга киритиш учун файлли ўзгарувчига аниқ бир файлни боғлаш керак (файл номини бериш).

Файл номини файлли ўзгарувчига ўзлаштириш AssignFile процедурасини чақириш орқали амалга оширилади.

AssignFile процедурасининг умумий кўриниши қуйидагича:

AssignFile (var f, файл_номи:string);

Файл номи Windows қоидаларига кўра берилади. У тўла бўлиши мумкин, яъни фақат файл номи эмас, балки файлга йўлни ҳам камраб олиши мумкин (диск номи, каталог ва осткаталоглар).

Қуйида бир нечта мисоллар келтирилган:

AssignFile(f, 'a:\result.txt');
AssignFile(f, '\students\abdullayev\dastur.txt');

```
fname:=('otchet.txt'); AssignFiie(f.fname);
```

ФАЙЛГА ЁЗИШ

Матн файлига бевосита ёзиш write ёки writeln буйруклари ёрдамида амалга оширилади. Умуман ушбу буйруклар қуйидагича бўлади:

```
Write(файлли_узг, Киритиш_рўйхати);
Writeln(файлли_узг. Киритиш_рўйхати);
```

бу ерда

файлли_ўзгарувчи – ёзилиш керак бўлган файлли ўзгарувчи номи;
киритиш_рўйхати – вергул орқали ажратилган файлга киритилиши керак бўлган ўзгарувчилар. Ўзгарувчилар номи ўрнида сатрли константани ҳам киритиш мумкин.

Масалан, агар f ўзгарувчи TextFile турида бўлса, у ҳолда x1 ва x2 ўзгарувчиларни киритиш қуйидагича бўлади:

```
Write(f,'Tenglama ildizlari=',x1.x2);
```

Write ва writeln буйруклари орасидаги фарк шундан иборатки, writeln киритилаётган кийматларни янги сатрдан бошлайди.

ФАЙЛНИ ҚИЙМАТ КИРИТИШ УЧУН ОЧИШ

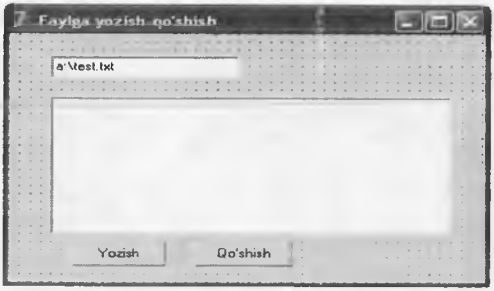
Файлга маълумот киритишдан олдин уни очиш зарур. Файлни очиш ва унга киймат киритишнинг иккита усули мавжуд:

устига ёзиш (мавжуд файл устига ёзиш ёки янги файл яратиш);
мавжуд файл давомига қўшиш.

Янги файл яратган ҳолда ёки мавжуд файлни ўрнига ёзиш учун файлни очишда Rewrite(f) процедурасидан фойдаланилади, бунда f –TextFile туридаги ўзгарувчи.

Мавжуд файлни унинг ичидаги маълумотлар давомига маълумот қўшиш максадида очиш учун Append(f) процедурасидан фойдаланилади, бунда f - TextFile туридаги ўзгарувчи.

4.1-расмда матн файлига маълумот қўшишни амалга оширувчи дастур мулокат ойнаси келтирилган.



4.1- расм. Файлга маълумот ёзиш – қўшиш дастури ойнаси.

4.1-листингда Ёзиш тугмаси босилиш билан бажариладиган процедура келтирилган. У файлни янги файл яратиш ёки эскисини устига янгисини алмаштириш усули билан очади ва Memo1 майдонидаги маълумотларни ушбу файлга ёзади.

Файл номини дастур бажарилиши мобайнида Edit1 майдонига киритиш зарур ёки олдиндан Edit1.text хусусиятига бирор бир файл номини масалан, test.txt ни ўзлаштириш керак.

4.1-листинг. Янги файл ёки мавжуд файлни ўрнига ёзиш.

```
procedure TForm1.Button1Click(Sender: TObject);
var
f: TextFile; // файл
fName: String[80]; // файла номи
i: integer;
begin
fName := Editl.Text;
AssignFile(f, fName);
Rewrite(f); // кайта ёзиш учун очиш
//файлга ёзиш
for i: =0 to Memol.Lines.Count do //сатрлар нолдан бошлаб номерланади
writeln(f, Memol.Lines[i]);
CloseFile(f); // файлни ёпиш
MessageDlg('Маълумотлар файлга ёзилди ',mtlnformation,[mbOk],0);
end;
```

4.2-листингда **Қўшиш** буйрук тугмасини босиш билан бажариладиган процедура матни келтирилган. У Edit1 майдонида кўрсатилган файлни очади а унга memo1 майдонидаги маълумотларни қўшади.

4.2-листинг. Мавжуд файлга маълумот қўшиш.

```
procedure TForm1.Button2Click(Sender: TObject);
var
f: TextFile; // файл
fName: String[80];.// файл номи
```

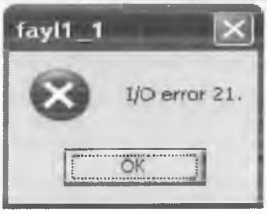
```
i: integer; begin
fName := Editl.Text;
AssignFile(f, fName);
Append(f); // қўшиш учун очиш
//файлга ёзиш
for i:=0 to Memo1.Lines.Count do // сатрлар нолдан бошлаб номерланади
writeln(f, Memo1.Lines[i]);
CloseFile(f); // файлни ёпиш
MessageDlg('Маълумотлар файлга қўшилди ',mtInformation,[mbOk],0);
end;
```

ФАЙЛНИ ОЧИШДАГИ ХАТОЛИКЛАР

Дастурни бажарилиши давомида фаайлни очишга уриниш муваффакиятсиз бўлиб хатоликка олиб келиши мумкин. Файлни очишда хатоликларга бир канча сабаблар бўлиши мумкин. Масалан, дастур ишга тайёр бўлмаган каттик дискдаги файлни очишга уринган пайтда. Бошка сабаб эса очилаётган файлнинг йуклиги (файл йук, хеч каерга қўшиб бўлмайди). Агар дастур Delphi дан туриб бажартирилса, файлни очишдаги хатолик юз берганда экранда куйидагича мулокат ойнаси (4.2- расм) пайдо бўлади:

4.2-расм. Файлни очишда хатолик ҳақидаги хабарга мисол(дастур Delphi дан бажартирилганда).

Агар дастур Windows дан бажартирилса бу хабар бошкача кўринишда бўлади (4.3-расм):



4.3-расм. Файлни очишда хатолик ҳақидаги хабарга мисол (дастур Windows мухитида ишга туширилганида).

Файлни очиш жараёнини бошкаришни дастурнинг ўзи кўлга олиши мумкин. Бунда IOResult (input-output result-киритиш/чикариш натижаси) функциясини кийматини текшириш билан амалга ошириш мумкин. Агар киритиш/чикариш муваффакиятли тамомланган бўлса, бу функция 0 кийматни, акс холда хатолик кодини кайтаради.

Дастур киритиш ва чикариш амаллари натижасини бошкариши учун рухсат бериш керак. Бунинг учун файлни очиш процедураси олдидан киритиш ва чикаришдаги хатоларни автоматик тарзда кайта ишлашни ман килувчи {$I-} директива сатрини киритиш керак. Бу директива билан дастур компиляторга хатоларни бошкарувни ўз кўлига олганлигини англатади. Файлни очиш процедурасидан сўнг {$I+} директивасини кўйиш зарур. бу киритиш/чикаришни автоматик кайта ишлаш холатига ўтилганлигини билдиради. Масалан:

```
AssignFile(f,filename);
{$I-}
Append(f) // кўшиш учун очиш
{$I+}
if IOResult<> 0 // очиш хатолиги
then Rewrite(f); // ёзиш учун очиш
// мавжуд ёки янги файлни очиш
```

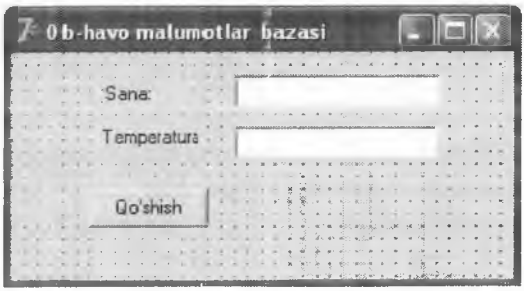
ФАЙЛНИ ЁПИШ

Дастур ёпилишидан олдин барча очилган файллар ёпилиши керак. Бу close процедурасини чакириш оркали амалга оширилади. Close процедураси битта параметрга эга бўлиб у файлни ўзгарувчидир. Процедурани ишлатишга мисол:

```
Close(f);
```

ДАСТУРГА МИСОЛ

Келтириладиган дастур оддий маълумотлар базасини киритишга мисолдир. Дастур ишга тушганда фойдаланувчи сана ва хаво температурасини киритиш мумкин бўлган мулокат ойнаси пайдо бўлади 4.4-расм.



4.4-расм. «Об-ҳаво» маълумотлар базаси дастурининг мулоқат ойнаси.

Сана Edit1 майдонига температура эса Edit2 майдонига киритилади. Дастур матни 4.3-листингда келтирилган.

4.3-Листинг. Оддий маълумотлар базаси (файлга ёзиш).

```
unit pogoda_;
interface
uses
Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms, Dialogs, StdCtrls;
type
TForm1 = class(TForm)
Edit1: TEdit; // сана
Edit2: TEdit; // температура
Button1: TButton; // қўшиш тугмаси
Label1: TLabel;
Label2: TLabel;
procedure FormActivate(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure FormClose(Sender: TObject;
var Action: TCloseAction); private
{ Private declarations } public
{ Public declarations } end;
var
Form1: TForm1;
implementation
{$R *.dfm}
const
DBNAME = 'a:\pogoda. db';
var
db: TextFile; // файл — маълумотлар базаси
procedure TForm1.FormActivate(Sender: TObject);
begin
```

```
AssignFile(db, DBNAME);. {$I-}
Append(db); if IOResult = 0 then
begin
Edit1.Text := DateToStr(Date);  // жорий санани олиш
Edit2.SetFocus;
//  Edit2 майдонига курсорни ўрнатиш
End else begin
Rewrite(db); if IOResult <> 0 then begin
// киритиш майдони ва буйрук тугмасининг фаоллигини тўхтатиш
Edit1.Enabled := False; Edit2.Enabled := False;
Button1.Enabled := False; ShowMessage(DBNAME+'  яратишда хатолик ');
end; end; end;
// Кўшиш тугмасини битта туртганда
procedure TForm1.Button1Click(Sender: TObject);
begin
if (Length(edit1.text)=0) or (Length(edit2.text)=0)
then ShowMessage('Киритишда хатолик'+#13+'Барча майдонлар
тўлдирилиши керак')
else writeln(db, edit1.text,' ',edit2.text);
end;
// OnClose ходисаси форма ёпилаётган вактда рўй беради
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
CloseFile(db);
// МБ файлни ёпиш
end;end.
```

Маълумотлар базаси файлни OnActivate ходисасини кайта ишловчи FormActivate процедураси очади. OnActivate ходисаси форма фаоллашган пайтда юз беради ва натижада ушбу процедура автоматик тарзда ишга тушади. Агар файлни очиш муваффакаятли бажарилса, у холда edit1 майдонига жорий сана ёзилади. Жорий сана маълумоти Date функцияси оркали олинади. Date функцияси кайтарган кийматни (Double турдаги сон) кулай турга утказиш учун Datetostr функциясидан фойдаланилади. OnActivate ходисасини кайта ишловчи процедура Edit1 майдонига санани жойлаштиргандан сўнг setFocus методи оркали курсорни температурани киритиш майдонига урнатади. Агар файлни очиш ва файлни яратиш жараёнида хатолик юз берса, у холда процедура **кўшиш** тугмасини мурожаат килиб бўлмайдиган килиб кўяди.

TForm1.Button1Click процедураси (onclick ходисасини кайта ишловчи процедура) **Кўшиш** тугмасини (button 1) босиш билан ишга туширилади. Натижада киритилган маълумот маълумотлар базаси pogoda.db файлига ёзилади. Дастур ёзишдан олдин барча майдонлар тўлдирилганлигини

текширади ва агар тўлдирилмаган бўлса, хабар чиқарали. Процедура ишининг натижаси сифатида pogoda.db файли охирига сана ва температурадан иборат сатр қўшилади.
Ҳар бир кунги маълумотларни МБ да алоҳида сатрда жойлаштириш учун дастурда write эмас writeln буйруғидан фойдаланилган. Writeln буйруғидаги 3 та элементга эътибор беринг. Файлга санани (edit1.text) киритгандан сўнг файлга битта бўш жой ва ундан сўнг температура (edit2.text) ёзилади.
Агар температурани санадан сўнг ёзилса у ҳолда, бу сана ва температура битта сон деб каралади.
TForm1.Formclose процедураси маълумотлар базасини ёпади. Дастур бир нечта марта бажарилгандан сўнг pogoda.db файли қуйидагича бўлиши мумкин, масалан:
9.05.2001 10 10.05.2001 12 11.05.2001 10 12.05.2001 7

ФАЙЛДАН КИРИТИШ

Дастур дастлабки маълумотларни клавиатурадан эмас, балки матн файлидан ўқиб олиши ҳам мумкин. Бундай имкониятдан фойдаланиш учун TextFile туридаги файлли ўзгарувчи эълон қилиш ва уни AssignFile буйруғи орқали фаоллаштириб файлни ўқиш учун очиш керак ва ундаги маълумотларни read ёки readln буйруғи орқали ўқиш лозим.

ФАЙЛНИ ОЧИШ

Файлни ўқиш учун очиш битта-файл типли параметрига эга бўлган Reset процедураси орқали амалга оширилади. Reset процедурасини чақиришдан олдин AssignFile функцияси аниқ бир файлни фаоллаштирган бўлиши керак. Масалан, қуйидаги буйруқлар файлни ўқиш учун очади:

```
AssignFile(f, 'c:\data.txt'); Reset(f);
```

Файлни ёзиш учун очгандаги каби IOResult функция қийматини текширган ҳолда хатоларни бошқариш мумкин. 4.4-листингда келтирилган дастур қисмида файлни очишда IOResult функцияси қийматини текшириш кўрсатилган:
4.4-листинг.

```
var
fname : string[80]; // файл номи
f : TextFile; // файл
res : integer; // файлни очиш хатолиги коди ( IOResult қиймати)
answ : word; // фойдаланувчи жавоби
begin
fname := 'a:\test.txt'; AssignFile(f,fname);
```

```
repeat
<$I-}
Reset(f); // файлни ўқиш учун очиш
{$!+}
res:=IOResult;
if res <> 0
then answ:=MessageDlg(fname+'ни очиш хатолиги'+#13+'Файлни
очиш такрорлансинми?',mtWarning,
[mbYes, mbNo],0); until (res= 0) OR (answ = mrNo);
if res <> 0
then exit; // процедуранинг тугатилиши
end;
```

МАЪЛУМОТЛАРНИ ФАЙЛДАН ЎҚИШ

Файлдан ўқиш read ва readln буйруклари орқали амалга оширилади ва қуйидаги кўринишда бўлади:

Read(файлли ўзгарувчи, ўзгарувчилар рўйхати)

Readln(файлли ўзгарувчи, ўзгарувчилар рўйхати)

бунда,

файлли ўзгарувчи-TextFile туридаги ўзгарувчи;

ўзгарувчилар рўйхати – вергул билан ажратилган ўзгарувчилар номи.

СОНЛАРНИ ЎҚИШ

Шуни эсда тутиш керакки, матн файлида сонлар эмас уларнинг акси сақланади. read ёки readln буйруклари орқали бажарилган амал дастлаб файлдан белгиларни биринчи ажратувчи (бўш жой ёки сатр охири) гача ўқиб олади, сонга айлантириб қийматни read ёки readln буйруғида келтирилган параметрга ўзгарувчига ўзлаштириб олади.

Масалан, агар матн файли a:\data.txt қуйидаги сатрга эга бўлса:

```
23 15
45 28
56 71
```

У ҳолда,

```
AssignFile(f, 'a:\data.txt');
    Reset(f); // ўқиш учун очиш
    read(f, a); read(f, b, c); read(f, d);
```

ўзгарувчилар қиймати қуйидагича бўлади:

a = 23, b = 15, c = 45, d = 28.

Readln буйруғининг read буйруғидан фарки шундаки, файлдан навбатдаги сонни ўкиб олиб ўзгарувчига ўзлаштиргандан сўнг ва хаттоки ўкилган сондан кейин сон мавжуд бўлса хам файлдаги кўрсаткич кейинги сатр бошига ўтади. Шунинг учун:

AssignFile(f,'a:\data.txt'); Reset(f); readin(f, a); readin(f, b, c); readin(f, d); буйрукдан сўнг ўзгарувчилар киймати куйидагича бўлади:

a = 23, b = 45, c = 28. d = 56

САТРЛАРНИ ЎКИШ

Дастурда сатрли ўзгарувчилар узунлиги кўрсатилиб ёки кўрсатилмасдан хам эълон килиниши мумкин. Масалан.

Stroka1 :string[10];

Stroka2 :string;

Сатрли ўзгарувчи киймати унинг узунлиги эълон килганда канча кўрсатилган бўлса шунда ундан ортик бўлмаган узунликда жорий сатрдан ўкиб олинади. Узунлиги кўрсатилмаган сатрли ўзгарувчи файлдан киймат ўкиган пайтда жорий сатр узунлигига тенг символни ўкийди. Бошкача килиб айтганда, агар файлдан сатрни барча элементларини ўкиш керак бўлса, у холда сатрли ўзгарувчи узунлигини файлнинг энг катта сатридан катта узунлик кўрсатинг. Масалан, бизга friends.txt матн файли берилган бўлсин:

Kariyev Anvar Sobirovich Alisher Rahmatov Abdullayevich

4.1-жадвалда ўзгарувчиларни эълон қилишнинг бир нечта вариантлари ва friends.txt матн файлидан ўкиб олиш буйруғи ва унинг киймати келтирилган.

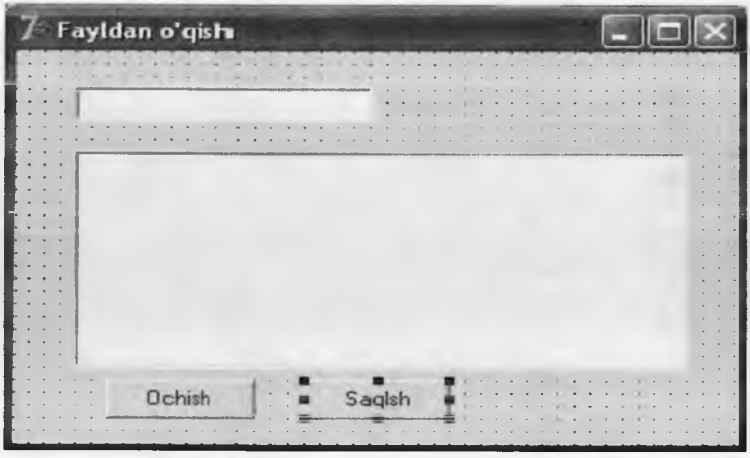
Файлдан сатрларни ўкишга мисол 4.1- жадвал.

Ўзгарувчиларни эълон қилиш ва уни файлдан ўкиш		Файлдан ўкигандан кейинги ўзгарувчи киймати
fam: string[15]	Readin (f, fam, name)	fam= 'Kariyev
Name: string[10]		name= ' Anvar'
fam, name: string;	Readin (f, fam, name)	fam= 'Kariyev Anvar '
		name= ' '
drug: string[80]	Readin (f, drug)	drug =' Kariyev Anvar '

ФАЙЛ ОХИРИ

Фараз киламиз дискда кандайдир матн файли мавжуд бўлсин. Мулокат ойнасига ушбу файлнинг барча элементларини чикариш керак бўлсин. Буни ечими деярли осон: файлни очиш, ундаги биринчи сатрни ва кейин иккинчи,

учинчи ва ҳакоза файлнинг охиригача ўқиш керак. Бирок қандай қилиб охирги сатрга келганлигини ёки файлнинг охирини аниқлаш мумкинми?
Файлнинг охирини аниқлаш учун EOF (end of File-файл охири). EOF функциясида битта параметр бўлиб, у файлли ўзгарувчидир. EOF функция False кийматни олади, агар ўқилган элемент файлда энг охирги бўлмаса, яъни файлдан яна ўқиш имкони мавжуд. Агар ўқилган элемент охирги бўлса, у ҳолда EOF нинг киймати True бўлади.
EOF функцияси кийматини файлни очган заҳотиёк текшириш мумкин. Агар бунда функция True кийматга тенг бўлса, бу файл ҳеч қандай маълумотга эга эмаслигини, яъни бўшлигини англатади. 4.5-листингда қўйилган масалани ечувчи процедура келтирилган. У файлдан фойдаланувчи иш давомида киритган сатрларни ўқиб олади ва уларни memo майдонида акс эттиради. Дастур ойнаси 4.5-расмда келтирилган.



4.5-расм. Файлдан ўқиш дастурининг ойнаси.

4.5-листинг. Файлдан ўқиш.

```
unit rd_;
interface
uses
Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms, Dialogs, StdCtrls, Buttons;
type
TForm1 = class(TForm)
Button2: TButton;
Edit1: TEdit;
```

```
Memo1: TMemo;
Button1: TButton;
procedure Button2Click(Sender: TObject);
procedure Button1Click(Sender: TObject); private
{ Private declarations ) public
{ Public declarations } end;
var
Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
var
f: TextFile; // файл fName: String[80]; // файл номи
buf: String[80]; // файлдан ўкиш учун буфер
begin
fName := Edit1.Text: AssignFile(f, fName); {$!-}
Reset(f); // ўкиш учун очиш{$I+}
if IOResult <> 0 then begin
MessageDlgt( fName+'файлига мурожат этишдаги
хатолик ',mtError,[mbOk],0);
exit; end;
// файлдан ўкиш
while not EOF(f) do begin
readln(f, buf); // файлдан сатрни ўкиш
Memo1.Lines.Add(buf); // сатрни Memo1 майдонига қўшиш
end;
CloseFile(f); // файлни ёпиш
end;
// Saqlash тугмасини танланиши — файлга ёзиш
procedure TForm1.Button2Click(Sender: TObject);
var
f: TextFile; // файл
fName: String[80]; // файл номи
i: integer/; begin
fName := Edit1.Text: AssignFile(f, fName);
Rewrite(f); // кайта ёзиш учун очиш
// файлга ёзиш
for i:=0 to Memo1.Lines.Count do // сатрлар нолдан бошлаб рақамланади
writeln(f, Memo1.Lines[i]);
CloseFile(f): // файлни ёпиш
MessageDlg('Маълумотлар файлга ёзилди ',mtInformation,[mbOk],0);
end;end.
```

Файлни қайта ишлашни ташкил қилиш мақсадида ҳар бир ўқиладиган сатрдан олдин EOF функцияси қийматини текширадиган while буйруғидан фойдаланилган. **Сақлаш** буйруқ тугмаси ва унга мос процедура memo майдонидаги матнни файлга ёзиш имконини беради, яъни файлдан ўқиш дастури матн редакторини ўзида намоён қилади.

Назорат саволлари

1. Файл қандай эълон қилинади?
2. Файл қандай фаоллаштирилади?
3. Файлга маълумот ёзиш қандай амалга оширилади?
4. Файл билан ишлашда қандай хатоликлар юз беради?
5. Файлдан маълумотлар қандай ўқиб олинади?

МУНДАРИЖА

КИРИШ

Масъул муҳаррир:
т.ф.д., проф Б.Ш.. Раджабов

Бичими 60х84 1/16
Босма табоғи- 1. Адади – 200

Ўзбекистон Ногиронлар Жамиятининг
"NORS EFFECT ELEKTRONICS"
шўъба корхонасида чоп этилди
Гувохнома реестри № 10 - 1465
Тошкент ш., 2-Чимбой кўчаси , 115а