ЎЗБЕКИСТОН РЕСПУБЛИКАСИ
ОЛИЙ ВА ЎРТА МАХСУС ТАЪЛИМ ВАЗИРЛИГИ

ГУЛИСТОН ДАВЛАТ УНИВЕРСИТЕТИ

Абдурахимов Д.Б. Негматуллоев З.Т.

Дастурлаш тиллари фанидан

фанидан ўқув–услубий қўлланма

Гулистон - 2018

Абдурахимов Д.Б., Негматуллоев З.Т. «**Дастурлаш тиллари**» фанидан ўқув-услубий қўлланма 2018 **й.- 313 бет.**

Ушбу ўқув-услубий қўлланма амалдаги дастурлар асосида тайёрланган бўлиб, олий ўқув юртларининг “5110700-Информатика ўқитиш методикаси” бакалавр таълим йўналишида таҳсил олаётган талабалар учун мўлжалланган. Унда замонавий педагогик технологиялар тизимига асосланган ҳолда назарий материаллар, амалий машғулотлар топшириқлари, билимларни назорат қилиш учун саволлар мажмуаси кабилар келтирилган.

Ўқув-услубий қўлланма Гулистон давлат университети Ўқув - методик Кенгаши томонидан 29.08.2018 йилдаги, 1- сонли баённома асосида нашрга тавсия этилган.

Тақризчилар С.И.Қулмаматов, “Ахборот технологиялари” кафедраси доценти, педагогика фанлари номзоди.

Д.Тоштемиров , “Ахборот технологиялари” кафедраси доценти, педагогика фанлари номзоди

Данный учебно-методический пособия подготовлен на основе действующей программы и предназначен для студентов бакалавра обучающихся по направлениям 5110700 - «Методика преподавание информатики». Содержатся теоретические материалы, методические требования и задания к выполнению практических и лабораторных работ, комплекс вопросов для контроля знаний. Комплекс создан на основе современной педагогической технологии.

The given uchebno-methodical complex is prepared on the basis of the operating program and intended for students of the bachelor trained on directions 5110700 - «The applied mathematics and computer science». Theoretical materials, methodical requirements and tasks to performance of practical and laboratory works, a complex of questions for the control of knowledge contain. The complex is created on the basis of modern pedagogical technology.

1-БОБ. БОШЛАНҒИЧ МАЪЛУМОТЛАР

Бу ерда Delphi дастурини ўрнатиш қисқа очиб берилади. Спорчининг масофани босиб ўтиш тезлигини хисоблаш мисолида визуал лойихалаш ва ходисаларни дастурлаш технологияси намойиш қилинади. Асосий тушунча ва терминлар аниқланади.

1.1. Delphi ни ўрнатиш

Borland Delphi 7 Studio пакетининг тўртта версияси мавжуд: Personal, Professional, Enterprise ва Architect. Бу пакетларнинг ҳар бири турли мақсадларга қаратилган юқори самарали дастурлар ишлаб чиқиш учун стандарт воситалар тўпламига эга. Пакетларнинг имкониятлари Personal дан Architect га қараб кенгайиб боради. Масалан, Enterprise комплекти масофадаги маълумотлар базаси (мисол учун, InterBase) билан ишлай олади, Personal да эса бундай имкон йўқ.

Ушбу китобдаги материаллар Delphi нинг бирор аниқ бир пакетига боғланмаган. Намуна тариқасида олинган барча масалалар Personal пакети доирасида амалга оширилиши мумкин.

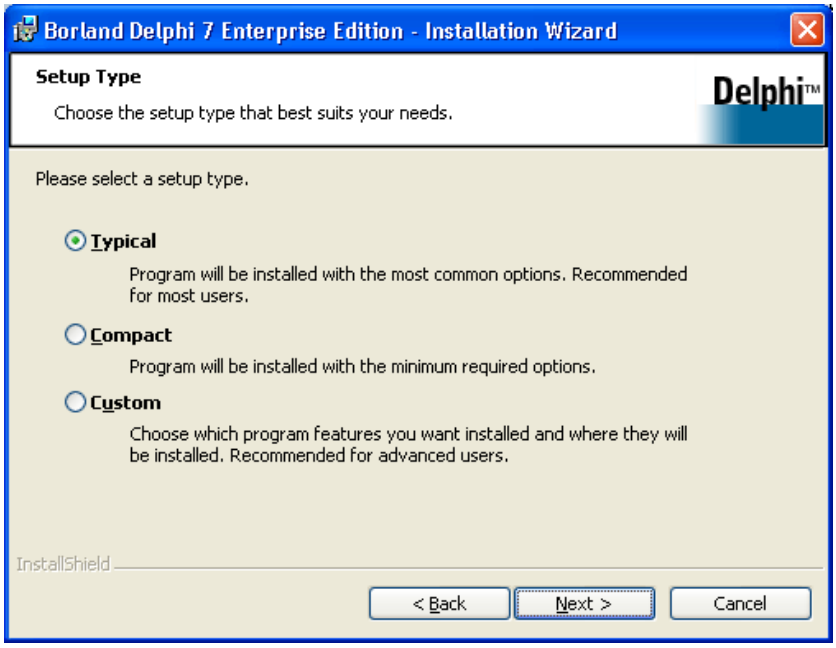
Delphi 7 ни компьютерга ўрнатиш барча зарур файллар ва ўрнатиш программаси (Delphi Setup Launcher) каби материаллар сақланаётган ўрнатувчи диск ёрдамида амалга оширилади. Бу дискни CD-диск юритувчи қурилмага қўйилган заҳоти, ўрнатувчи дастур автоматик тарзида ишга тушади.

1-расм. Delphi 7 ўрнатишни бошлаш

Натижада экранда ўрнатувчи CD-ROM ёрдамида компьютерга ўрнатилиши мумкин бўлган дастурий маҳсулотларнинг рўйхатини ўз ичига олган **Delphi 7 Setup Launcher ойнаси** (1-расм) пайдо бўлади.

Бу рўйхатда аввало Delphi 7, қолаверса InterBase 6.5 маълумотлар базаси сервери, InstallShield Express - ўрнатувчи CD-ROM ларни яратиш утилитлари кабиларни кўриш мукмин.

Delphi ни ўрнатишни бошлаш учун **Delphi** 7 сатри чертилади. Delphi ни ўрнатиш жараёни оддий. Серия номери (Serial Number) ва калит (Authorization Key) киритилганидан сўнг, экранда дастлаб лицензион келишув ойнаси, кейин **Setup Type** (2-расм) ойнаси пайдо бўлади. Бу ойнада ўрнатишнинг мумкин бўлган вариантлари тавсия қилинади: **Typical** (Оддий), **Compact** (қисқартилиган) ёки **Custom** (фойдаланувчи танлаб ўрнатадиган).



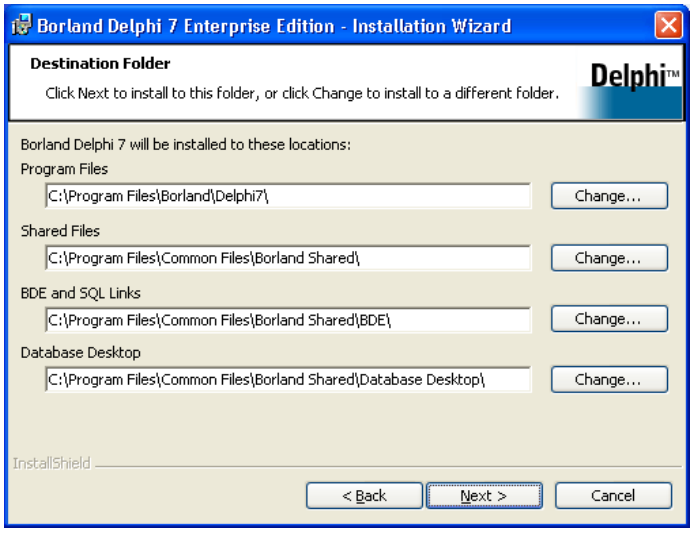
2-расм. **Setup Type** диалог ойнасидан ўрнатиш вариантини танлаш

Оддий вариант ўрнатишда CD-ROM дан қаттиқ дискка Delphi нинг барча компоненталарини тўлиқ кўчирилади. Бунинг учун дискда камида 475 Мбайт (Enterprise пакети учун) атрофида бўш жой бўлиши талаб қилинади. Агар компьютерда етарлича бўш жой бўлса, шу вариантни ўрнатиш тавсия қилинади.

Қисқартириб ўрнатишда эса Delphi нинг энг муҳим компоненталари кўчириб олинади. Бу вариант қаттиқ дискдан энг кам бўш жой талаб қилади. Қисқартириб ўрнатилганда, Delphi нинг айрим имкониятларидан фойдаланиб бўлмайди. Чунки, бу ҳолда қаттиқ дискка ёрдамчи маълумотномалар системасининг файллари, айрим компонента ва утилитлар, намуналар кўчирилмайди.

Фойдаланувчи танлаб ўрнатадиган вариант дастурчига Delphi нинг энг керакли восита ва компоненталарини танлаб ўрнатишга имкон беради. Одатда, бу вариантни тажрибали дастурчилар қўллашади. Delphi ни тўлиқ ўрнатиш учун қаттиқ дискда етарлича бўш бўлмаган ҳолда бу вариантдан фойдаланиш мумкин.

Ўрнатиш вариантини танлагандан сўнг, **Next** тугмасини босинг. Агар **Custom** варианти танланган бўлса, **Custom Setup** ойнаси очилади. Ундан ўрнатилмайдиган компоненталарни аниқланади. Компонентани ўрнатишни таъқиқлаш учун компонента номидан чапдаги диск тасвирини чертиш ва очилган меню пунктларидан **Do Not Install** ни танлаш лозим.



3-расм. Компоненталар ўрнатиладиган каталоглар рўйхати

Агар ўрнатишнинг **Typical** варианти танланган бўлса, у ҳолда Next тугмасини чертиш натижасида **Destination Folder** ойнаси очилади. Унда Delphi пакети ва унинг компоненталари ўрнатиладиган каталоглар рўйхати таклиф қилинади.

Навбатдаги **Next** тугмасини чертиш **Save Installation Database** ойнасини очади. Унда фойдаланувчига Delphi ни қаттиқ дискка ўрнатиш жараёни ҳақидаги ахборотни сақлаб қўйиш тавсия қилинади. Бу ахборот Delphi ни ўрнатувчи дисксиз ўчиришда (деинсталляция қилишда) керак бўлиши мумкин. Шу билан ўрнатишга тайёргарлик босқичи тугайди. Экранда **Ready To Install the Program** ойнаси пайдо бўлади. **Install** тугмасини чертиш ўрнатиш жараёнини фаоллаштиради ва ўрнатиш бошланади. Ўрнатиш жараёни тугаганидан сўнг, бу ҳақида махсус ахборот экранга чиқарилади (4-расм). **Finish** тугмасини чертиш бу ойнани ёпади.

4-расм. Ўрнатиш жараёни тугади

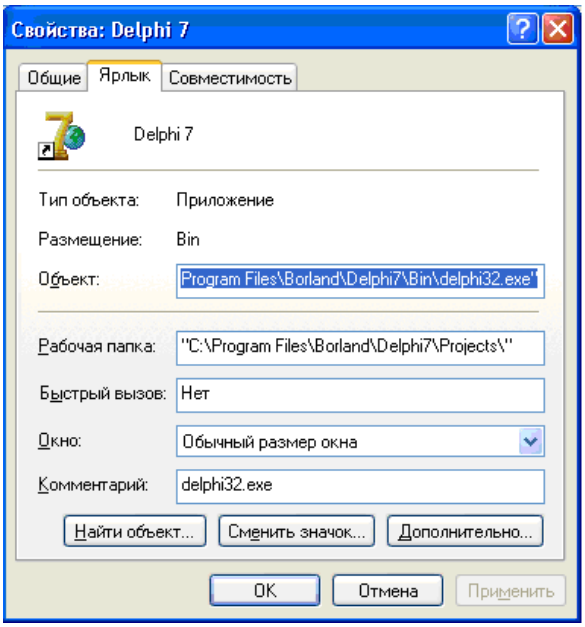
Энди Delphi ни ишга тушириш мумкин. Аммо, бундан олдин ишчи каталог ва лойихалар каталогини кўрсатиб қўйиш лозим. Бунинг учун сичқонча кўрсаткичи Delphi ни ишга тушириш буйруғига келтирилади: **Пуск / Программў / Borland Delphi 7 / Delphi**7. Сўнгра сичқончанинг ўнг томонини босиб, пайдо бўлган контекст менюдан **Свойства** тугмасини танланади. Очилган **Свойства: Delphi**7 ойнасидаги **Рабочая папка** ойнасида Delphi лойихалари учун мўлжалланган папка номи кўрсатилади. (5-расм).

менюга

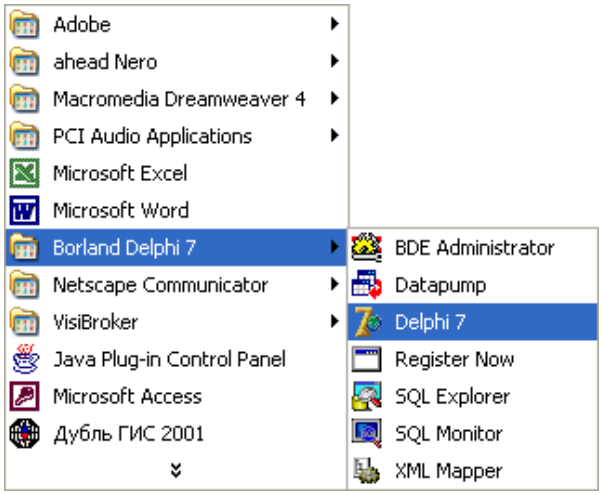
1.2. Ишни бошлаш

Delphi бошқа иловалар каби одди усулда ишга туширилади, яъни **Borland Delphi**7 менюсидан **Delphi**7 буйруғи танланади.

Delphi ишга тушганидан сўнг, экран 7-расмдаги кўринишни



5-расм. Лойихалар папкасини кўрсатиш

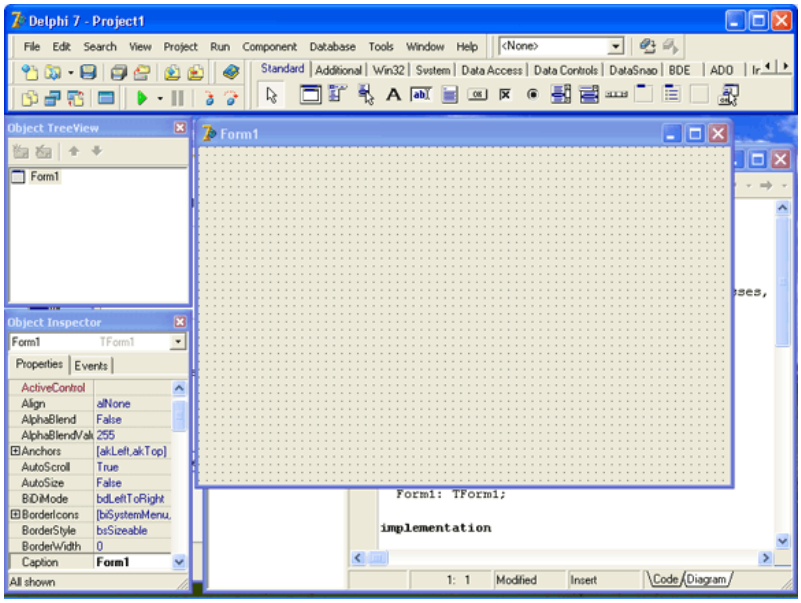


6-расм. Delphi ни ишга тушириш.

олади. Бу ойнада бир вақтнинг ўзида 5 та диалог ойна пайдо бўлади:

- **Delphi7** нинг бош ойнаси**;**
- **Form1 -** бошланғич форма ойна ;
- **Object Inspector -** объектлар хусусиятини таҳрирлаш ойнаси;
- **Object TreeView –** объектлар рўйхатини кўриш ойнаси;
- **Unitl.pas –** кодларни таҳрирлаш ойнаси.

Кодларни таҳрирлаш ойнаси деярли тўлалигича бошланғич форма ойнаси билан тўсиб қўйилган.

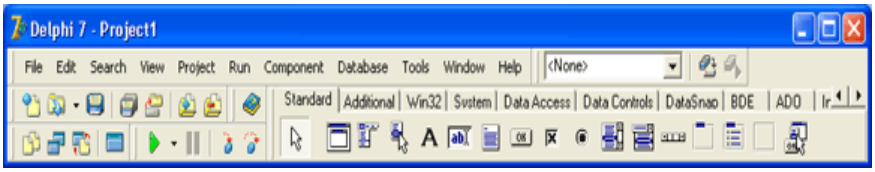


7-расм. Delphi ишга тушганидан кейин экраннинг кўриниши

Бош ойнада (8-расм) буйруқлар менюси, қуроллар панели, компоненталар палитраси жойлаштирилган.

Бошланғич форма ойнаси **(Form1)** яратиладиган лойиха учун олдиндан тайёрланган ишланмадан иборат.

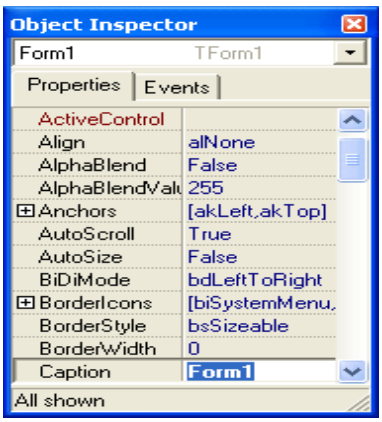
Дастурий таъминотни системали ва амалий гуруҳларга бўлинади. Системали дастурий таъминот – бу операцион системадан иборат. Қолган барча дастурларни амалий дастурий таъминот деб аталади. Амалий дастурларни қисқа қилиб ***иловалар*** деб аташ қабул қилинган.



8-расм. Бош ойна

Object Inspector (9-расм) — объектларнинг хусусиятларини таҳрирлаш ойнаси объектларнинг

ҳусусиятларини ўзгартириш учун мўлжалланган. ***Объект*** деганда диалог ойналари, бошқариш элементлари (киритиш ва чиқариш майдонлари, буйруқли тугмалар, ўчиргичлар ва х.к.) назарда тутилади. ***Объектнинг ҳусусияти*** эса объектнинг кўриниши, ҳолати, ҳулқи каби характеристикаларини билдиради. Масалан, ***Width*** ва ***Height*** ҳусусиятлари форманинг ўлчамларини (баландлиги ва кенглиги) белгиласа, ***Top*** ва ***Left*** форманинг экрандаги ҳолатини аниқлайди, ***Caption*** ҳусусияти сарлавҳа матнини кўрсатади.



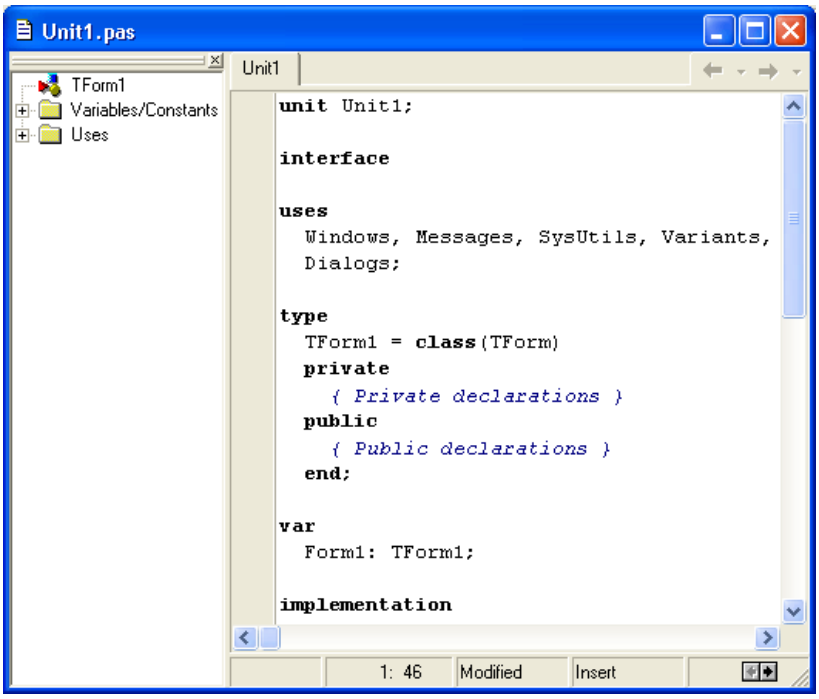
9-расм. **Properties** пунктида объект ҳусусиятлари кўрсатилган.

Кодларни таҳрирлаш ойнасида (10-расм), (уни форма ойнасини бир четга суриб, очиш мумкин) дастур матни ёзилади. Янги лойиха устида иш бошланганда, кодларни таҳрирлаш ойнасида Delphi да яратилган тайёр дастур шаблони жойлашган бўлади.

менюга

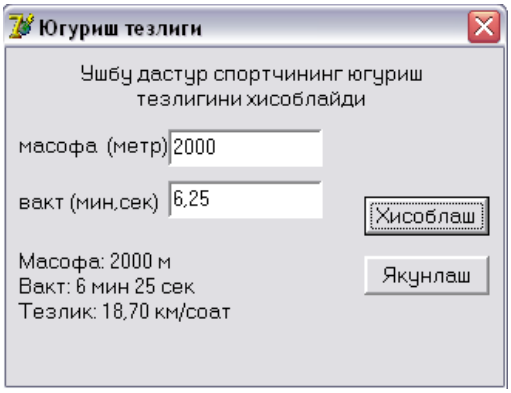
Биринчи лойиха

Delphi нинг имкониятлари ва визуал дастурлаш технологиясини намойиш қилиш учун спортчи белгиланган масофани қандай тезлик билан босиб ўтишини ҳисобловчи лойиха яратамиз. Дастурнинг диалог ойнаси ва ишлаши 11-расмда келтирилган.



10-расм. Кодларни таҳрирлаш ойнаси

Янги дастур устида иш бошлаш учун Delphi ни ишга туширинг. Агар сиз бу муҳитда бошқа лойиха билан ишлаётган бўлсангиз, **File** (Файл) менюсидан **New** / **Application** (Создать / Приложение) буйруғини танланг.



11-расм. Югуриш тезлигини ҳисоблаш ойнаси

Форма

Яратилаётган илова, яъни янги лойиха устида ишлаш диалог ойнасини, яъни бошланғич формани қуришдан бошланади.

Бошланғич форма **Form1** формасининг ҳусусиятларини ўзгартириш ҳамда унга эхтиёжга қараб керакли компоненталарни (киритиш ва чиқариш майдонлар, буйруқли тугмалар) ни ўрнатиш орқали яратилади.

Форманинг ҳусусиятлари (1-жадвал) унинг ташқи кўриниши, ўлчамлари, сарлавҳа матни ҳошиясининг кўринишини белгилайди. Форма ва унинг компоненталари ҳусусиятлари ва қийматларини ўзгартириш учун **Object Inspector** ойнасидан фойдаланилади. Бу ойнанинг юқори қисмида объектнинг номи ҳамда ҳусусиятларининг жорий вақтдаги қийматлари кўрсатилади. **Properties** (ҳусусияти) қистирмасининг чап колонкасида объектнинг ҳусусиятлари, ўнг томонда эса уларнинг қийматлари келтирилади.

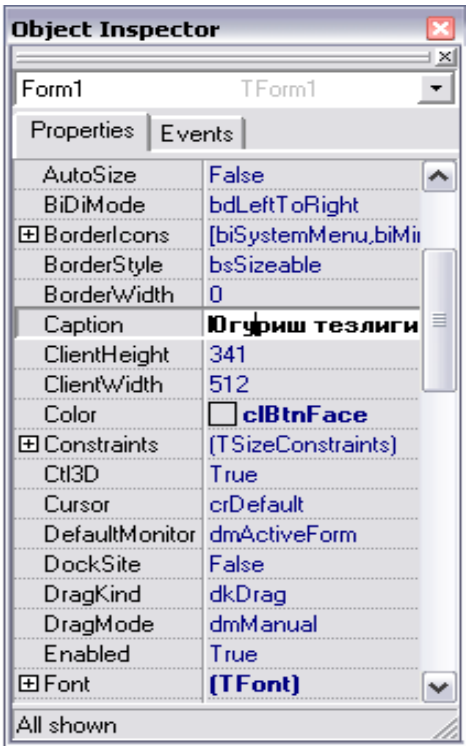
Форманинг (***mform*** объектининг) ҳусусиятлари 1-жадвал

Ҳусусият	Мазмуни
Name	Форманинг номи. Дастурда форманинг номи формани бошқариш ва форма компоненталарига мурожаат қилиш учун фойдаланилади.
Caption	Сарлавҳа матни
Width	Форманинг кенглиги
Height	Форманинг баландлиги
Top	Форманинг юқори чегарасидан экраннинг юқори чегарасигача бўлган масофа
Left	Форманинг чап чегарасидан экраннинг чап чегарасигача бўлган масофа
BorderStyle	Чегаранинг кўриниши. Чегара оддий (bsSizeable), ингичка (bs Single) бўлиши ёки умуман бўлмаслиги (bsNone) мумкин. Агар ойнанинг чегараси оддий бўлса, уни фойдаланувчи сичқончадан фойдаланиб, ўзгартириш мумкин. Ингичка чегарали ойна ўлчамларини ўзгартириб бўлмайди. Агар чегара бўлмаса, у ҳолда экранга сарлавҳасиз ойна чиқарилиши мумкин. Бундай ойнанинг ҳолати ва ўлчамларини дастурнинг иши мобайнида ўзгартириш мумкин эмас.
BorderIcons	Ойнани бошқариш тугмалари. Ҳусусиятиниг қиймати дастурнинг иши давомида фойдаланувчилар қайси тугмалардан фойдаланиш мумкинлигини кўрсатади. Ҳусусиятнинг қиймати biSystemMenu, biMinimaze, biMaximaze ва biHelp ҳусусиятларининг қийматларини аниқлаш орқали берилади. biSystemMenu ҳусусияти ихчамлаш ва система тугмаларига, biMinimaze— ихчамлаш тугмасига, biMaximaze — кенгайтириш тугмасига, biHelp — маълумотномаларни чиқариш тугмаси билан ишлашга рухсат беради.
Icon	Диалог ойнаси сарлавҳасидаги система менюсини чақиришни англатувчи нишон.

Color	Фон ранги. Рангни унинг номини кўрсатиб ёки операцион системанинг ранглари гаммасига боғлаб қўйиш орқали белгилаш мумкин. Иккинчи ҳолда ранглар жорий ранглар схемаси бўйича аниқланади ва операцион системанинг ранглар схемаси ўзгарганда ўзгаради.
Font	Шрифт. Форма сиртида "кўрсатилмаганда ҳам" фойдаланиладиган жорий шрифт. Форманинг ҳусусияти ўзгартирилганда форма сиртида жойлашган компонентаплярнинг Font ҳусусияти автоматик тарзда ўзгаради, яъни компонентаплар формадан Font ҳусусиятини мерос қилиб олади.

Форма яратишда биринчи навбатда caption (сарлавҳа) ҳусусиятининг қийматини ўзгартириш лозим. Бизнинг мисолимизда "Form1" матнинг "югуриш тезлиги" билан алмаштириш керак. Бунинг учун **Object Inspector** ойнасида сичқонча тугмасини **Caption** сатрида чертамиз. Натижада ҳусусиятнинг жорий қиймати ажратиб кўрсатилади ва шу сатрда курсор пайдо бўлади. Шундан кейин "Югуриш тезлиги" матнини киритиш мумкин. (12-расм).

Худди шу усул билан форманинг кенглиги ва баландлигини аниқловчи **Height** ва **width** ҳусусиятларни ўзгартириш мумкин. Форманинг ўлчамлари, унинг ҳолати ҳамда бошқа бошқарув элементларининг ўлчамлари ва уларнинг форма сиртидаги ҳолатлари пикселларда (экрандаги нуқталар) берилади. **Height** ва **width** ҳусусиятларини мос равишда 250 ва 330 қилиб белгиланг.



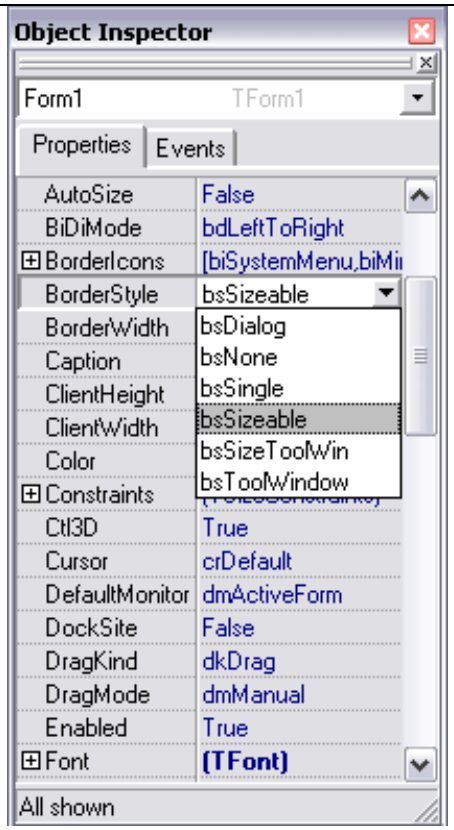
12-расм. Ҳусусият қийматини қийматни киритиш орқали ўзгартириш

Форма — бу оддий ойнадир. Шунинг учун унинг ўлчамларини бошқа ойналар каби сичқонча ёрдамида ўзгартириш мумкин. Бунда **Height** ва **Width** ҳусусиятларининг қийматлари автоматик тарзда ўзгаради.

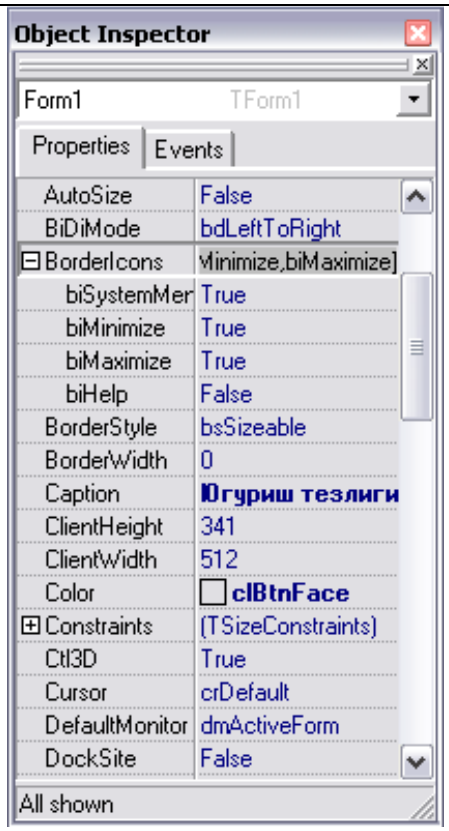
Диалог ойнасининг экрандаги ҳолати формани ташкил қилишдаги ҳолатига мос келади. Бу ҳолатни **Top** (экраннинг юқори чегарасидан чекиниш) ва **Left** (экраннинг чап чегарасидан чекиниш) ҳусусиятларининг қийматлари белгилаб беради. Бу қийматларни ҳам сичқонча ёрдамида ўзгартириш мумкин.

Айрим ҳусусиятларни танлашда, масалан, **Borderstyle**, ҳусусиятнинг жорий қийматини белгилашда, ўнг томонда очиладиган рўйхат таклиф қилинади. Қийматни ана шу рўйхатдан танлаш мумкин. (13-расм)

Айрим ҳусусиятлар мураккаб ҳисобланади, чунки уларнинг қийматлари бошқа ҳусусиятларнинг қийматларидан келиб чиқиб аниқланади. Бундай ҳусусиятларнинг олдида "+" нишони туради. У чертилса, аниқловчи ҳусусиятлар рўйхати таклиф қилинади. (14-расм). Масалан, BorderIcons ҳусусияти ойналарни бошқаришнинг қайси тугмалари билан дастур иши давомида ишлаш мумкинлигини белгилайди. Агар biMaximize ҳусусиятига False қиймати берилса, дастурнинг ишлаши жараёнида **Развернуть** тугмаси ойна сарлавҳасида кўринмайди.



13-расм. Ҳусусият қийматларини рўйхатдан танлаш



14-расм. Мураккаб ҳусусиятларнинг очилган рўйхати

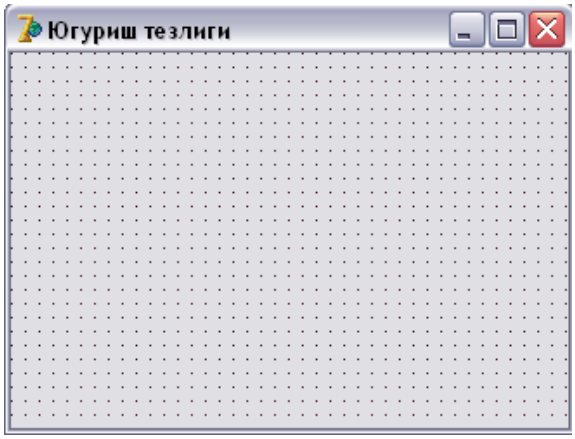
Айрим ҳусусиятларнинг ёнида уч нуқтали тугма жойлашган. Бу ҳусусият қийматини аниқлашда янги диалог ойнасидан фойдаланиш мумкинлигини англатади. Масалан, **Font** мураккаб ҳусусиятининг қийматини белгилашда шрифт танлашнинг стандарт ойнасидан фойдаланиш мумкин.

2-жадвалда яратилаётган форманинг ўзгартириладиган ҳусусиятлари келтирилган. Ўзгармайдиган ҳусусиятлар бу жадвалга киритилмаган.

Бошланғич форманинг ҳусусиятлари 2-жадвал.

Ҳусусият	**қиймат**
Caption	Югуриш тезлиги
Height	250
Width	330
BorderStyle	bsSingle
BorderIcons . biMinimize	False
BorderIcons . biMaximize	False
Font. Size	10

Келтирилган жадвалда айрим ҳусусиятларнинг ёнида нуқта (.) белгиси турибди. Бу ҳусусиятнинг аниқланадиган қийматини белгилашни билдиради. Бошланғич форманинг жадвалдаги ҳусусият лари ўрнатилганидан сўнг, форма 15-расмдаги кўринишни олади.



15-расм. Бошланғич форманинг кўриниши.

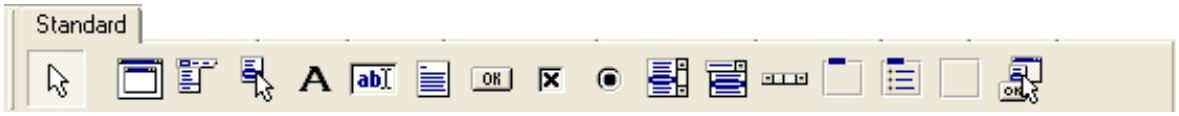
менюга

Компонента

Югуриш тезлигининг дастури фойдаланувчидан бошланғич маълумотлар – масофа ҳамда шу масофани югуриб босиб ўтиш вақтини олиши лозим. Бундай ҳолларда одатда бошланғич маълумотлар киритиш майдонларига клавиатура ёрдамида киритилади. Шунинг учун формага киритиш майдони **Edit** компонеталарини жойлаштириш лозим.

Энг кўп фойдаланиладиган компонента **Standard** қуроллар панелида жойлаштирилган (16-расм).

Формага компонентани қўшиш учун компонента палитрасидан шу компонента пиктограммаси устига сичқончани келтириб, чертилади. Сўнгра курсорни компонентанинг чап юқори бурчаги формада жойлашиши керак бўлган нуқтага ўрнатилади ва чап тугмани яна бир марта чертилади. Натижада формада стандарт ўлчамли компонента пайдо бўлади.

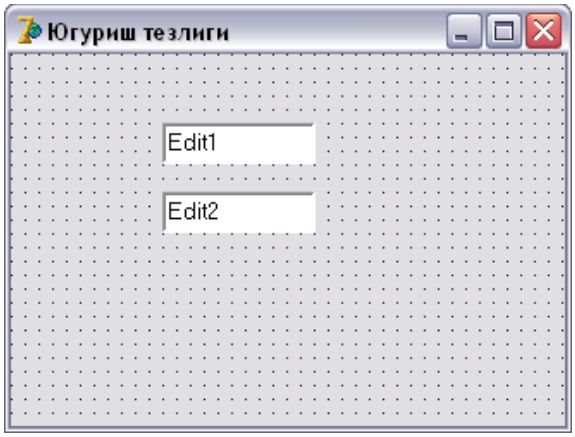


16-расм. Standard қуроллар панелининг компонента рўйхати

Компонентанинг ўлчамларини уларни формага қўшиш жараёнида белгиланиши мумкин. Бунинг учун компонента сичқонча ёрдамида танланганидан сўнг, курсорни форманинг компонента чап юқори бурчаги туриши керак бўлган нуқтасига келтириб, чап тугмаси босилади ва уни қўйиб юбормаган ҳолда курсорни компонентанинг қуйи ўнг бурчаги туриши керак бўлган нуқтага олиб келинади. Шундан сўнг сичқончанинг чап тугмасини қўйиб юбориш мумкин. Формада кўрсатилган ўлчамдаги компонента пайдо бўлади.

Delphi формага қўшилаётган хар бир компонентага ном беради. Бу ном компонента номи ва унинг тартиб номеридан иборат бўлади. Масалан, формага иккита Edit компонентаси қўшилган бўлса, уларнинг номлари мос равишда Edit1 ва Edit2 бўлади. Дастурчи Name ҳусусияти қийматини ўзгартириб, бу номларни бошқасига алмаштириши мумкин. Одатда содда дастурларда компоненталарнинг номлари алмаштирилмайди.

17-расмда форманинг иккита Edit компонентаси, яъни киритиш майдонларини қўшилганидан кейинги ҳолати келтирилган. Компоненталарнинг бири ажратилган. Ажратилган компонентанинг ҳусусиятлари **Object Inspector** ойнасида тасвирланган. Бошқа компонента ҳусусиятларини кўриш учун сичқончанинг чап тугмасини шу компонента устида чертиш лозим. Шунингдек, компонента номини **Object TreeView** ойнасидан ёки **Object Inspector** ойнасининг юқори қисмидаги объектларнинг очиладиган рўйхатидан ҳам танлаш мумкин.



17-расм. Иккита **Edit** компоненталари қўшилган форма

3-жадвалда киритиш майдони - Edit компонентасининг ҳусусиятлари келтирилган.

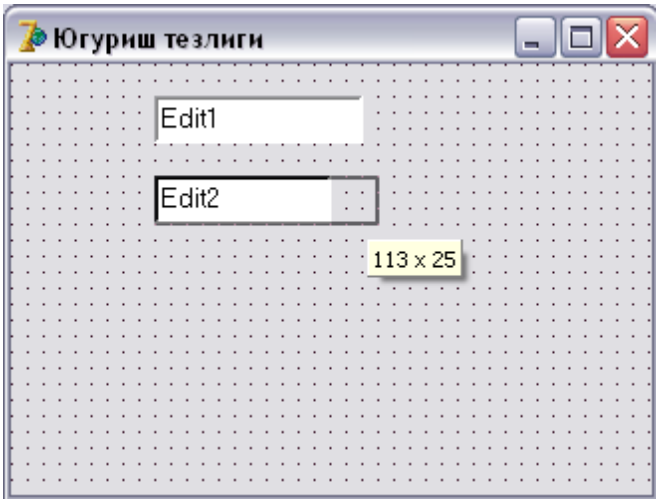
Киритиш майдони Edit компонентаси ҳусусиятлари 3-жадвал

Ҳусусияти	**Мазмуни**
Name	Компонентанинг номи. Дастурда ундан компонента ва унинг ҳусусиятлари билан ишлаш учун фойдаланиш мумкин. Масалан, таҳрирлаш майдонига киритилган матн билан ишлаш
Text	Киритиш ва таҳрирлаш майдонидаги матн
Left	Компонента чап чегарасидан форманинг чап чегарасигача бўлган масофа
Top	Компонента юқори чегарасидан форманинг юқори чегарасигача бўлган масофа
Height	Майдоннинг баландлиги
Width	Майдоннинг кенглиги
Font	Киритилаётган матн учун шрифт
ParentFont	Компонента томонидан форма шрифти аломатларини мерос қилиб олиш белгиси. Агар ҳусусиятнинг қиймати **True** бўлса, у ҳолда форманинг **Font** ҳусусияти ўзгарганда компонентанинг **Font** ҳусусияти ҳам автоматик тарзда ўзгаради.

Delphi компонента ўлчамларини сичқонча ёрдамида ўзгартиришга ҳам имкон беради.

Компонента ҳолатини ўзгартириш учун сичқонча курсорни унинг тасвири устига келтириб, чап тугмаси босилади. Уни қўйиб юбормаган ҳолда компонентани форманинг керакли жойига олиб борилади. Компонентанинг сурилиб бориши билан (18-расм) компонентанинг чап юқори бурчагининг координаталари кўрсатиб борилади (Left ва Top нинг қийматлари).

Компонента ўлчамини ўзгартириш учун стрелкани компонента чегарасини кўрсатувчи маркерлардан бирига ўрнатиб, чап тугмасини босиб турган ҳолда компонента чегарасини эхтиёжга қараб сурилади. Сўнгра чап тугмани қўйиб юбориш мумкин. Ўлчамни ўзгартириш вақтида компонентанинг Height ва Width ҳусусиятларининг жорий қиймати кўрсатиб борилади.



18-расм. Left ва Top ҳусусиятларининг жорий қийматлари

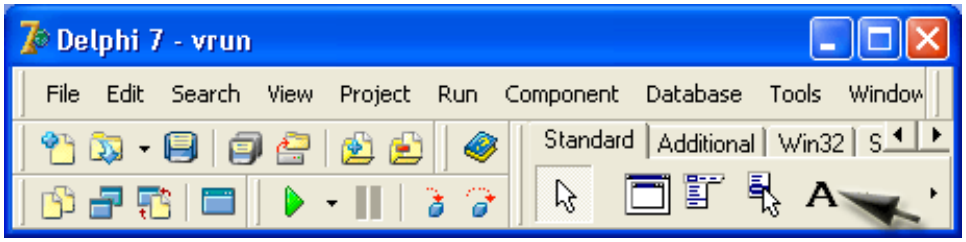
4-жадвалда Edit1 ва Edit2 киритиш майдонларининг ҳусусиятлари келтирилган. Edit1 - компонентаси масофа, Edit2 –эса шу масофани босиб ўтиш вақтини киритиш учун мўлжалланган. Ҳар икки компонентанинг ҳусусиятининг қиймати бўш сатр эканлигига эътибор беринг.

Киритиш майдонларидан ташқари, дастур ойнасида дастур ва киритиш майдонининг нимага мўлжалланганлиги ҳақида қисқа ахборот бўлиши лозим. Бундай ахборотларни формага қўйиш учун матнларни чиқариш майдонидан фойдаланилади. **Матнларни чиқариш майдони** – бу **Label** компонентасидир. **Label** компонентаси

Edit компоненталарининг ҳусусиятлари 4-жадвал

Ҳусусияти	**Компонента**	
	Edit1	**Edit2**
Text		
Top	56	88
Left	128	128
Height	21	21
Width	121	121

Standard қуроллар панелида жойлашган. (19-расм). Label компонентасини формага киритиш майдони каби қўйиш мумкин.



19-расм. Label компонентаси — матнларни чиқариш майдони

Тайёрланаётган илова формасига тўртта Label компонента-сини қўшиш лозим. 1-майдон ахборотнома учун, 2- ва 3 - майдонлар киритиш майдонларининг мақсадини кўрсатиш учун, 4-майдон эса ҳисоб натижасини (тезликни) чиқариш учун мўлжалланган.

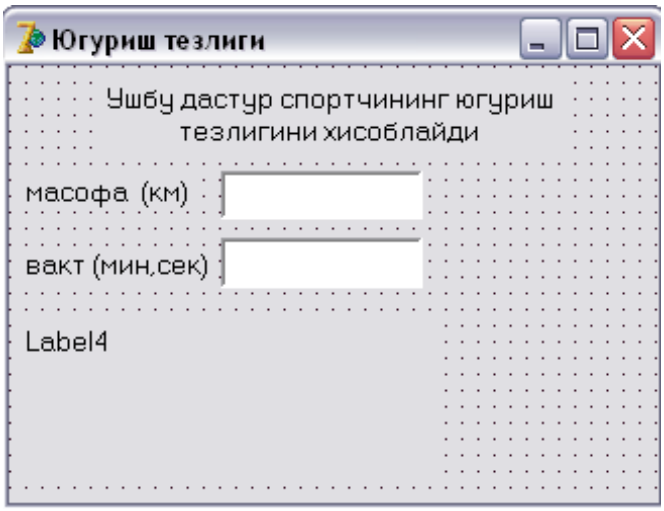
Label компонентасининг ҳусусиятлари 5-жадвал

Ҳусусияти	**Мазмуни**
Name	Компонентанинг номи бўлиб, компонента ва унинг ҳусусиятларига мурожаат қилиш учун хизмат қилади.
Caption	Чиқариладиган матн
Font	Матннинг шрифи

ParentFont	Форма белгиларини компонента томонидан мерос қилиб олиниши. Агар ҳусусиятнинг қиймати True бўлса, матн форма учун белгиланган шрифтда чиқарилади.
AutoSize	Майдоннинг ўлчами унга ёзилган белгилар сонига қараб белгиланади.
Left	Чиқариш майдонининг форманинг чап чегарасидан чекиниш масофаси
Top	Чиқариш майдонининг форманинг юқори чегарасидан чекиниш масофаси
Height	Чиқариш майдонининг баландлиги
Width	Чиқариш майдонининг кенглиги
Wordwrap	Сўзлар жорий сатрга сиғмаганда, уларни автоматик тарзда навбатдаги сатрга ўтказиш аломати.

Autosize ва **Wordwrap** ҳусусиятларига алоҳида эътибор беринг. Бу ҳусусиятлардан чиқариш майдонидаги маълумотлар бир нечта сатрга мўлжалланган ҳолларда фойдаланиш мумкин. Формага **Label** компонентасини қўшилганда **Autosize** ҳусусиятининг қиймати True, яъни, ўлчам caption ҳусусиятининг қийматининг ўзгаришига қараб аниқланади. Агар чиқариш майдонидаги матн бир нечта сартни эгаллаши талаб қилинса, формага **Label** компонентаси қўшилганидан кейин, **Autosize** - га **False**, **wordwrap** - га **True** қийматларини бериш лозим. Сўнгра **Width** ва **Height** ҳусусиятлари қийматларини аниқлаб қўйиш керак. Шундан кейингина caption ҳусусиятига матнни киритиш мумкин.

Формага тўртта киритиш майдони (Label компонентаси) жойлаштирилиб, уларнинг ҳусусият қийматлари кўрсатилганидан сўнг форма 20-расмдаги кўринишни олади.



20-расм. Чиқариш майдонлари қўшилганидан сўнг форманинг кўриниши

Caption сатрига матн битта сатр каби киритилади. Матннинг чиқариш майдонидаги ҳолати **Autosize** ва **Wordwrap** ҳусусиятлари ҳамда матннинг шрифтига боғлиқ бўлади.

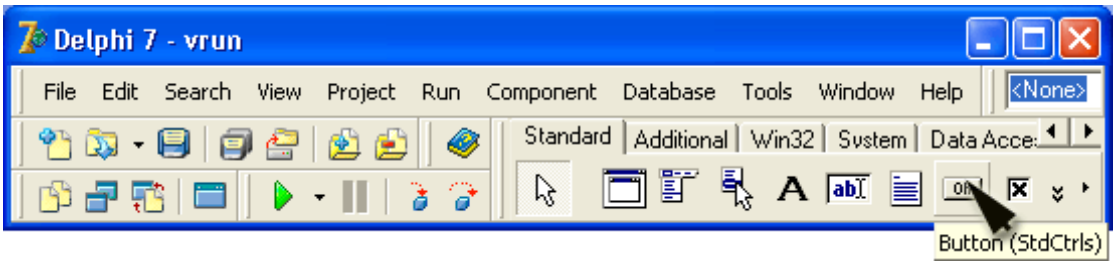
Label1, Label2, Label3 и Label4 компоненталарининг ҳусусиятлари **6-жадвал**

Компонента	**Ҳусусияти**	**Қиймати**
Labell	AutoSize	False
	Wordwrap	True
	Caption	Ушбу дастур спортчининг югуриш тезлигини хисоблайди
	Top	8
	Left	8
	Height	33
	Width	209
Label2	Top	56
	Left	8
	Caption	Масофа (метр)

Label3	Top	88
	Left	8
	Caption	Вақт (мин,сек)
Label4	AutoSize	False
	Wordwrap	True
	Top	120
Label 4	Left	8
	Height	41
	Width	273

Форманинг сўнгги босқичида иккита буйруқли тугма ўрнатилади: **Хисоблаш** ва **Якунлаш.** Уларнинг маъноси равшан.

Буйруқли тугма – **Button** компонентаси формага бошқа компонентаплар каби қўшилади. Унинг Button нишони **Standard** қуроллар панелида жойлашган. (21-расм).



21-расм. Буйруқли тугма — Button компонентаси

Button компонентасининг ҳусусиятлари 7-жадвал

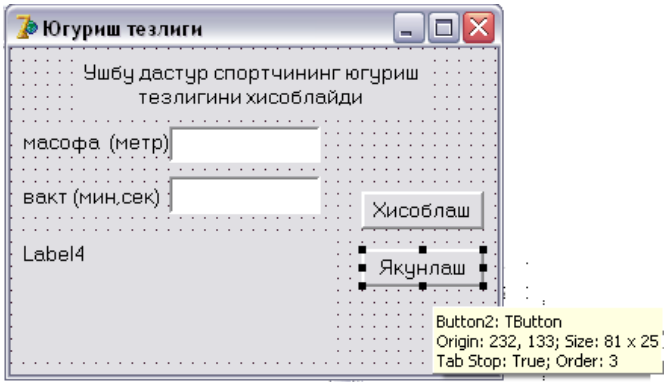
Ҳусусияти	**Мазмуни**
Name	Компонента номи. Дастурда компонента ва унинг ҳусусиятларига мурожаат қилишда ишлатилади.
Caption	Тугма устидаги матн
Enabled	Тугма билан ишлашга рухсат. Унинг қиймати True бўлса тугма билан ишлаш мумкин, акс ҳолда – йўқ.
Left	Чиқариш майдонининг форманинг чап чегарасидан чекиниш масофаси
Top	Чиқариш майдонининг формани юқори чегарасидан чекиниш масофаси
Height	Тугманинг баландлиги
Width	Тугманинг кенглиги

Формага иккита буйруқли тугма ўрнатилганидан сўнг, уларнинг қийматларини 8-жадвалга мос равишда белгилаш лозим.

Button1 ва Button2 компоненталарининг ҳусусиятлари. 8-жадвал

Ҳусусияти	**Компонента**	
	Button1	**Button2**
Caption	**Ҳисоблаш**	**Якунлаш**
Top	176	176
Left	16	112
Height	25	25
Width	75	75

Форманинг якуний кўриниши 22- расмда келтирилган.



22-расм. Югуриш тезлиги дастурининг формаси

Форма яратилганидан сўнг, дастур матнини ёзиш мумкин. Бундан олдин Delphi да дастурлашнинг икки муҳим тушунчасини аниқлаб оламиз: **ходиса** ва **ходисаларни қайта ишлаш процедураси.**

менюга

1.3. Ходиса ва ходисаларни қайта ишлаш процедураси

Форманинг кўриниши дастурнинг қандай ишлашидан дарак бериб турибди. Фойдаланувчи киритиш майдонига бошланғич маълумотларни киритиши ҳамда **Хисоблаш** тугмасини чертиши лозим. Буйруқ тугмасини чертиш Delphi да ***ходиса*** деб аталади.

Ходиса (Event) — бу дастурнинг иши жараёнида содир бўладиган воқеадир. Сичқонча тугмасини бир марта чертиш - ***OnClick***, икки марта чертиш ***OnDblClick*** ходисаси ҳисобланади. 9-жадвалда айрим ходисалар келтирилган.

Ходиса 9-жадвал

Ходиса	**Содир бўлади**
OnClick	Сичқонча тугмаси чертилганда
OnDblClick	Счт икки марта чертилганда
OnMouseDown	Сичқонча тугмаси босиб турилганда
OnMouseUp	Сичқонча тугмаси қўйиб юборилганда
OnMouseMove	Сичқонча сурилганда
OnKeyPress	Клавиатура тугмаси босиб турилганда
OnKeyDown	Клавиатура тугмаси босилганда OnKeyDown ва OnKeyPress ходисалари- то клавиатуранинг ушлаб турилган тугмаси қўйиб юборилмагунча навбати билан такрорланиб турувчи ходисалардир. (Бу вақтда OnKeyUp ходисаси рўй беради)
OnKeyUp	Клавиатура тугмаси қўйиб юборилганда
OnCreate	Объект (форма, бошқариш элементлари) яратилганда. Бу ходисаларни қайта ишлаш процедураси ўзгарувчиларни эълон қилиш ва тайёргарлик кўришда фойдаланилади.
OnPaint	Дастур иш бошлаган вақтда экранда ойна пайдо бўлганда. Бошқа ҳолларда эса ойнанинг бир қисми бошқа ойна билан тўсиб турилганда.
OnEnter	Бошқарув элементи томонидан фокус олинганида
OnExit	Бошқарув элементи фокусни йўқотганда

Ходиса рўй берганда дастур қандайдир реакция билдириши керак. Delphi ходисага ходисаларни қайта ишлаш процедураси орқали жавоб беради. Шундай қилиб, фойдаланувчининг бирор хатти-харакатига (ходисага) дастур бирор вазифани бажариб жавоб бериши учун дастурчи шу ходисага мос қайта ишлаш процедурасини ёзиши лозим. Шуни эътиборга олиш керакки, ходисаларни қайта ишлаш процедурасининг каттагина қисми компонента зиммасига юкланган. Шунинг учун дастурчи ходисага жавоб реакцияси стандарт бўлмаган ёки аниқланмаган ҳоллардагина ходисаларни қайта ишлаш

процедурасини ёзиши керак. Масалан, масала шарти бўйича **Edit** майдонига киритиладиган белгилар учун чекловлар бўлмаса, **OnKeyPress** ходисаларни қайта ишлаш процедурасини ёзишнинг кераги йўқ, чунки дастурнинг иши давомида бу вазифа стандарт равишда (дастурчига кўрсатилмаган ҳолда) бажарилади.

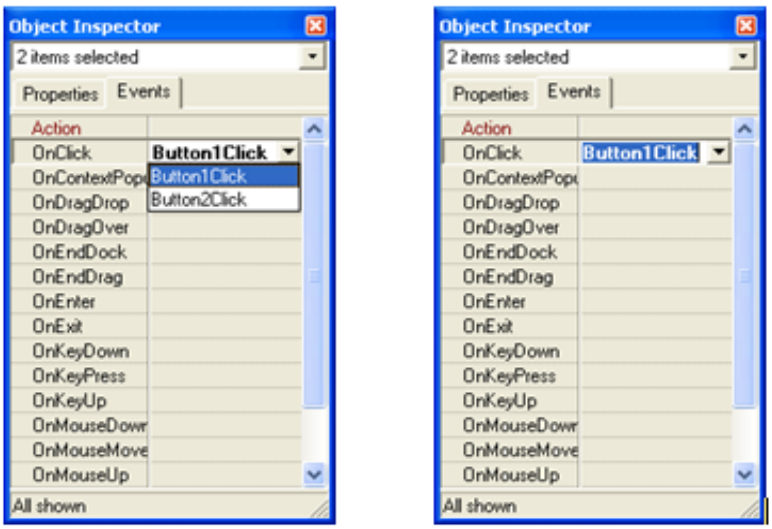
OnClick ходисасининг қайта ишлаш процедурасини **Хисоблаш** буйруқли тугмаси мисолида яратамиз.

Ходисаларни қайта ишлаш процедураси яратишни бошлаш учун дастлаб **Object Inspector** ойнасида ходисаларни қайта ишлаш процедураси ёзиладиган компонентани танлаймиз. Шу ойнанинг ўзида **Events** (Ходиса) пунктини танлаймиз.

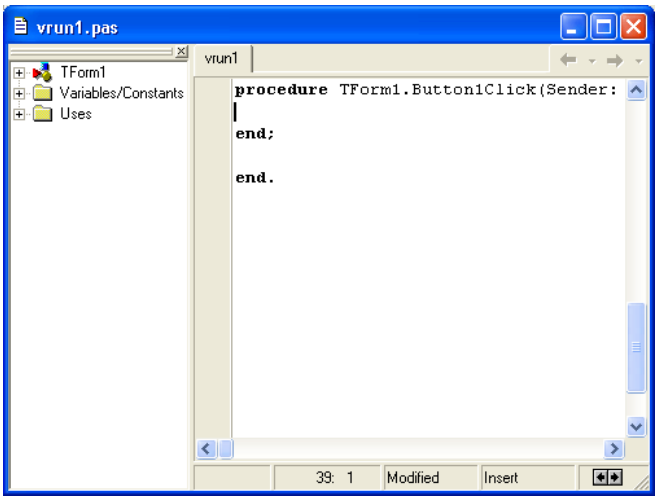
Events нинг чап устунида (23-расм) белгиланган объект учун содир бўлиши мумкин бўлган ходисалар рўйхати келтирилган. Агар ходиса учун ходисаларни қайта ишлаш процедураси аниқланган (ёзилган) бўлса, у ҳолда ўнг томондаги устунда шу процедуранинг номи пайдо бўлади.

Ходисаларни қайта ишлаш функциясини ташкил қилиш учун мос ходисаларни қайта ишлаш процедурасининг керакли номи майдонида сичқонча тугмаси икки марта чертилади. Натижада кодларнинг таҳрирлаш ойнаси очилади. Бу ойнага ходисаларни қайта ишлаш процедурасининг шаблони автоматик тарзда қўшилади, **Object Inspector** ойнасида эса ходисанинг номи билан бир қаторда уни қайта ишлаш функцияси номи пайдо бўлади. (24-расм).

Delphi ходисаларни қайта ишлаш функциясига икки қисмдан иборат ном беради. Номнинг биринчи қисми ходисаларни қайта ишлаш функцияси ёзилаётган компонентани ўз ичига олган формани таниш учун, иккинчи қисми эса объект ва ходисани белгилаш учун хизмат қилади. Бизнинг мисолимизда форманинг номи — Form1, буйруқли тугма номи — Button1, ходисанинг номи эса - Click.



23-расм. Events пунктида компонента (бизнинг мисолимизда - буйруқли тугма) қабул қила оладиган ходисалар рўйхати келтирилган.



24-расм. Delphi ходисаларни қайта ишлаш процедурасининг шаблони

Кодларни таҳрирлаш ойнасида ***begin*** ва ***end*** сўзлари орасида ходисаларни қайта ишлаш процедураси буйруқларини ёзиш мумкин.

1-листингда **Хисоблаш** буйруқли тугмаси учун **onclick** ходисаларни қайта ишлаш функциясининг матни келтирилган. Унинг умумий кодларни таҳрирлаш ойнасидаги кўринишига мос келади: калит сўзлар қорайтириб, изоҳлар курсив шрифтда, буйруқлар эса сатр бошидан чекинтириб (шундай одат мавжуд) ёзилган.

1-листинг. Хисоблаш тугмаси учун ***Onclick*** ходисасини қайта ишлаш процедурасининг матни.

```
// хисоблаш тугмаси босилганда
procedure TForm1.ButtonlClick(Sender: TObject);
var
masofa : integer;// масофа, метрда
t: real;// вақт ҳақиқий сон кўринишида
min : integer;// вақт, минутлар
sek : integer;// вақт, секундлар
v: real;// тезлик
begin
// киритиш майдонидан бошланғич маълумотларни олиш
masofa := StrToint(Edit1.Text); t := StrToFloat(Edit2.Text);
// дастлабки алмаштиришлар
min := Trunc(t);// минутлар — t сонининг бутун қисми
sek := Trunc(t*100) mod 100;
// секундлар — t сонининг каср қисми
// хисоблаш
v := (masofa/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/соат';
end;
```

Button1click функцияси тезликни ҳисоблаб, натижани **Label4** майдонига чиқаради. Бошланғич маълумотлар эса **Editl** ва **Edit2** киритиш-таҳрирлаш майдонларининг **Text** ҳусусиятига мурожаат қилиб олинади. **Text** ҳусусияти дастурнинг ишлаши жараёнида фойдаланувчи киритган белгилар кетма-кетлигидан иборат бўлади. Дастурнинг иши тўғри бўлиши учун бу кетма-кетлик фақат рақамлар ва вергулдан иборат бўлиши лозим. Сатрни сонга айлантириш учун **StrToInt** ва **StrToFloat** функцияларидан фойдаланилган. ***StrToInt*** функцияси берилган сатрдаги (***Edit1.Text*** - майдонидаги) белгиларни текширади. Агар ҳамма белгилар рақамлардан иборат бўлса, улар ифодалайдиган бутун сонни қиймат қилиб олади ва бу қийматни **masofa** ўзгарувчисига беради. ***StrToFioat*** функцияси ҳам худди шу каби ишлайди. У ***Edit2.Text*** майдонидаги рақамлар кетма-кетлигига мос келадиган ҳақиқий сонни ***t*** ўзгарувчига қиймат қилиб беради.

Бошланғич маълумотлар **masofa** ва **v** ўзгарувчиларга берилганидан сўнг, хисоблаш жараёни бошланади. Дастлаб ***Trunc*** функцияси ёрдамида соннинг каср қисми ташлаб юборилади ва *t* — нинг бутун қисми (минутлар) аниқланади. ***Trunc(t*100) mod 100*** ифодасининг қиймати секундлар миқдорини топади. Бу қуйидагича ҳисобланади: дастлаб ***t*** сони 100 га кўпайтирилади, ***Trunc*** функцияси ёрдамида кўпайтманинг каср қисми ташлаб юборилади, ҳосил бўлган сонни 100 га бўлиб, ***mod*** ёрдамида қолдиғи топилади.

Ҳамма маълумотлар тайёр бўлганидан сўнг, хисоблаш бошланади. Тезлик км/соат ларда ифодалангани учун, метрда берилган масофа ҳамда минут ва секундларда берилган вақт километр ва соатларга айлантирилади.

Тезликнинг хисобланган қиймати **Label4** майдонига, унинг **Caption** ҳусусиятини ўзгартириб чиқарилади. Сонларни сатрга айлантириш учун ***IntToStr*** ва ***FloatToStr*** функцияларидан фойдаланиш мумкин.

Якунлаш тугмаси босилганида дастур ўз ишини тугатиши лозим. Бунинг учун экрандан дастурнинг бош ойнасини ёпиб, олиб ташлаш керак. Бу вазифани ***close*** методи ёрдамида бажариш мумкин. Якунлаш тугмаси учун ***Onclick*** ходисаларни қайта ишлаш процедурасининг матни 2-листингда келтирилган.

2-листинг. Якунлаш тугмаси учун ***Onclick*** процедурасининг матни.

```
// Якунлаш тугмаси босилганда
procedure TForm1.Button2Click(Sender: TObject);
begin
Form1.Close; // дастурнинг бош ойнасини ёпиш
end;
```

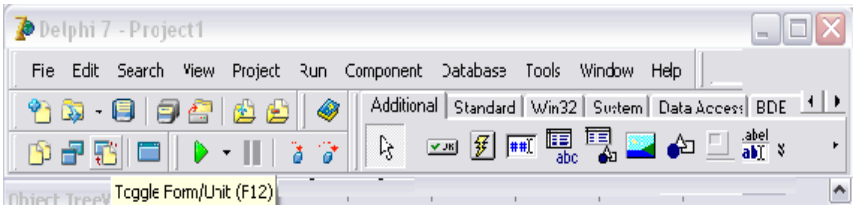
менюга

1.4. Кодлар муҳаррири

Кодлар муҳаррири Delphi нинг калит сўзларини (procedure, var, begin, end, if ва бошқ.) қорайтириб ёзади. Бу дастур матни ифодали қилиб, дастур структурасини тушунишни осонлаштиради.

Изоҳлар курсив (қийшайтрилган) шрифтда ифодаланади.

Дастурни тайёрлаш жараёнида кодлар муҳарриридан форма ойнасига ва аксинча ўтишга тўғри келади. Бу ишни **View** қуроллар панелидаги **Toggle Form / Unit** буйруқли тугмаси ёки **<F12>** клавишаси ёрдамида амалга ошириш мумкин.

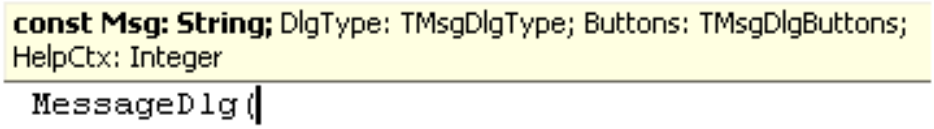


25-расм. View қуроллар панели

Эслатмалар системаси. Дастур матнини киритиш жараёнида кодлар муҳаррири процедура ва функцияларнинг параметрлари, объектларнинг ҳусусиятлари ва методлари ҳақидаги эслатмаларни экранга чиқариб боради.

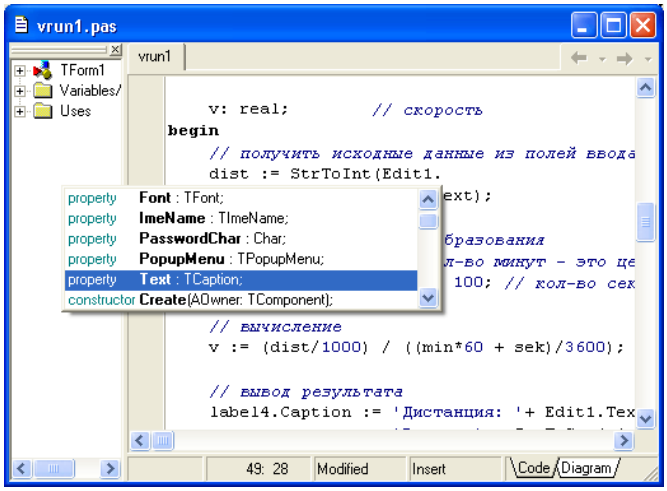
Масалан, кодлар муҳаррири ойнасида ***MessageDlg*** (экранга ахборотларни чиқарувчи функция) ҳамда очилган қавс белгиси ёзилса, экранда ***MessageDlg*** функциясининг параметрлари ҳақидаги эслатмалар ойнаси пайдо бўлади. (26-расм). Параметрлардан бири қорайтириб кўрсатилади. Бунда муҳаррир фойдаланувчига қайси параметрни киритиш кераклиги эслатиб қўяди. Бу параметр ёзилиб, вергул қўйилганидан кейин навбатдаги параметр ҳақидаги эслатма пайдо бўлади. Бу жараён барча параметрлар кўрсатилмагунча давом этади.

Объектлар учун кодлар муҳаррири ҳусусиятлар ва методлар рўйхатини экранга чиқаради. Дастурчи объект (компонента) нинг



26-расм. Кодлар муҳаррири автоматик тарзда объект (компонента) нинг ҳусусиятлари ва методлари рўйхатини чиқаради.

номини ёзиб, нуқта белгисини қўйиши билан экранда шу объектнинг ҳусусиятлари ва методлари рўйхатидан иборат эслатмалар ойнаси пайдо бўлади. (27-расм).

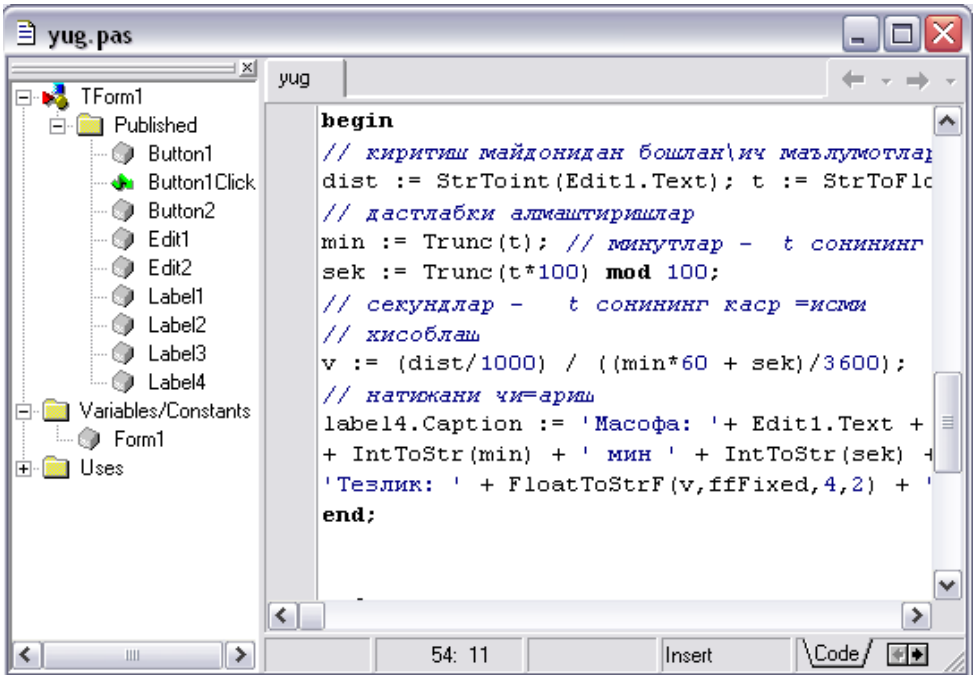


27-расм. Edit компонентасининг эслатмалар ойнаси

Рўйхатдаги зарур бўлган элементга сичқонча курсори ёки клавиатурадан керак бўлган хусусият ёки методнинг дастлабки бир нечта харфларини ёзиш орқали ўтиш мумкин. Рўйхатдан танланган элемент клавиатурадан <ENTER> ёки сичқончанинг чап тугмаси босилса, дастур матнига қўшиб қўйилади.

Эслатмалар системаси дастур матнини тайёрлаш жараёнини анча енгиллаштиради. Бундан ташқари, агар эслатма пайдо бўлмаса, дастурчи ҳатоликка йўл қўйганлигини билдиради. Масалан, процедура ёки функциянинг номи нотўғри ёзилган бўлиши мумкин.

Коднинг навигатори. Кодлар муҳаррири ойнаси икки қисмдан иборат. Ўнг томонга дастур матни ёзилади. Чап томон эса код навигатори (Code Explorer) деб аталади. (31-расм) У дастур матни бўйлаб навигацияни (йўл топиш) осонлаштиради. Структураси тайёрланаётган лойихага боғлиқ бўлган иерархик рўйхатда лойиха формалари, уларнинг компоненталари, ходисаларни қайта ишлаш процедуралари, глобал ўзгарувчилар ва константалар ўз аксини топган. Рўйхатдан керакли элементни топиб, дастур матнидаги код фрагментига осонгина ўтиб олиш мумкин.



28-расм. Code Explorer ойнаси дастур матни бўйича навигацияни енгиллаштиради.

Код навигатори ойнасини одатдаги усуллар билан ёпиш мумкин. Экранга код навигатори ойнасини чақириш учун **View** менюсидан **Code Explorer** тугмасини танлаш етарли.

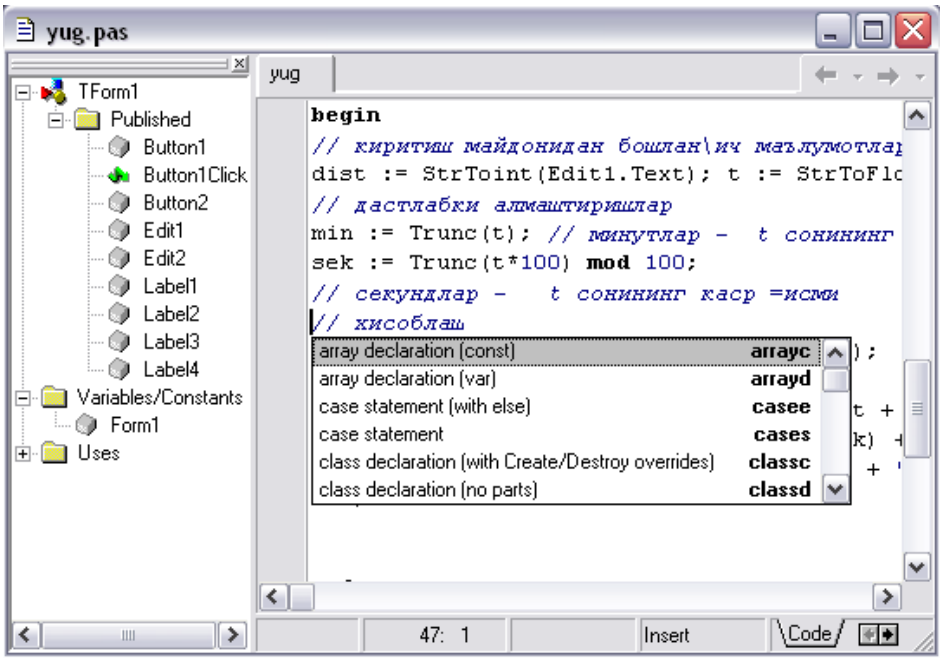
Коднинг шаблонлари. Дастур матнини киритиш жараёнида ***код шаблонлари*** (Code Templates) дан фойдаланиш мақсадга мувофиқ ҳисобланади. Коднинг шаблони – бу дастурнинг Delphi да ёзилган буйруқларининг умумий кўринишидир. Масалан, ***case*** буйруғининг шаблони қўйидагича:

```
case of :;
  :;
  else ;
```

end;

Кодлар муҳаррири дастурчига шаблонларнинг катта тўпламини таклиф қилади: массив, класс, функция ва процедураларни эълон қилиш, танлаш ва тармоқланиш буйруқлари ва х.к. Айрим буйруқлар учун шаблонларнинг бир нечта вариантлари мавжуд.

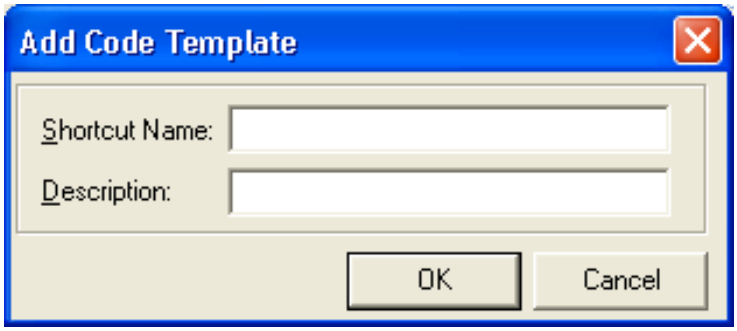
Дастур матнини ёзиш жараёнида код шаблонларидан фойдаланиб, уларни дастур матнига қўшиш учун **Ctrl** + **J** тугмалар комбинациясини босиш керак. Экранда пайдо бўлган рўйхатдан керакли шаблонни танлаш мумкин. (29-расм) Шаблонни одатдаги усуллар билан танланади: рўйхатни айлантирилади ёки унинг дастлабки бир нечта харфларини ёзиб кўрсатилади. Рўйхатдан



29-расм. Код шаблонлари рўйхати Ctrl + J тугмалари ёрдамида чақирилади

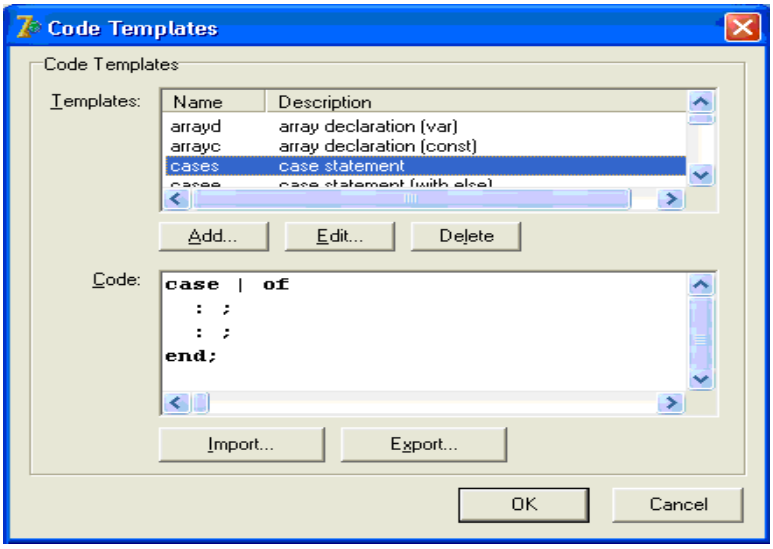
керакли элемент топилганидан сўнг, <Enter> тугмаси босилади ҳамда белгиланган шаблон дастур матнига қўшиб қўйилади.

Дастурчи агар зарурат бўлса, ўзининг код шаблонларини яратиш ва улардан худди стандарт шаблонлар каби фойдаланиши мумкин. Бунинг учун у **Tools** менюсидан **Editor Options** буйруғини танлаши, сўнгра **Source Options** пунктидаги **Edit Code Templates** тугмасини босиб, экранда пайдо бўлган **Code Templates** диалог ойнасидаги **Add** тугмасини чертиб, очилган ойнадан **Add Code Template** пунктини танлаши лозим. (30-расм) Шундан кейин пайдо бўлган ойнада шаблон номи **(Shortcut Name)** ҳамда унинг қисқа **(Description)** характеристикасини кўрсатиши лозим. Сўнгра **ОК**



30-расм. Ойнада шаблон номи ва унинг характеристикаси кўрсатилади.

тугмасини чертиб, **Code Templates** диалог ойнасининг **Code** ойнасига шаблонни киритади (31-расм).



31-расм. Дастурчи яратган шаблонга намуна

менюга

1.5. Лойиха структураси

Delphi да яратилган лойиха дастурий бирлик - модуллар тўпламидан иборат. Модулларнинг бири асосий бўлиб, дастур шу модулдан бошлаб бажарилади.

Асосий модул кенгайтмаси .dpr бўлган файл ҳисобланади. Лойиханинг асосий модул матнини кўриш учун **Project менюсидан View Source** буйруғирини танлаш лозим.

3-листингда югуриш тезлиги дастурининг асосий модули келтирилмоқда.

3-листинг. Югуриш тезлиги дастурининг асосий модули матни.

```
program yugu1;
uses
  Forms,
  yug in 'yug.pas' {Form1};
{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
End.
```

Асосий модул ***program*** сўзи билан бошланади. Ундан кейин номи лойиха номи билан бир ҳил бўлган дастур номи келади. Лойиха номи лойихани сақлаш вақтида берилади ва бу ном компилятор томонидан яратиладиган бажарилувчи дастур номини аниқлайди. Сўнгра, ***uses*** сўзидан кейин фойдаланилган модуллар рўйхати берилади.

{$R *.RES} сатри — бу компиляторга ресурслар файлининг ишга туширишга кўрсатмадир. Ресурслар файли илованинг ресурсларини ўз ичига олади: пиктограммалар, курсорлар, битли тасвирлар ва х.к. Юлдузча нишони ресурслар файлининг номи ҳам лойиха файли билан бир ҳил, аммо .Res кенгайтмали эканлигини англатади.

Ресурслар файли "матнли" файл эмас , шунинг учун уни матнлар муҳаррири ёрдамида кўриб бўлмайди. Ресурслар файли билан ишлаш учун махсус **Resource Workshop** каби дастурлардан фойдаланилади. Шунингдек, Delphi таркибига кирган **Image Editor** утилитини ҳам қўллаш мумкин. Уни **Tools** менюсида жойлашган.

Асосий модулнинг бажариладиган қисми ***begin*** ва ***end*** сўзлари орасида берилади. Бу қисм илована инициализация қилади ва экранга бошланғич ойнани чиқаради.

Асосий модулдан ташқари, ҳар бир дастур ўз ичига ҳеч бўлмаганда битта форма модулини олади.

Унда илова бошланғич формаси ҳамда унга керак бўлган процедуралар рўйхати сақланади. Delphi да ҳар бир формага ўзининг модули мос келади. .

4-листингда югуриш тезлигини хисоблаш дастури модулининг матни келтирилган.

4-листинг. Югуриш тезлиги дастурининг модули.

```
unit yug;
interface
uses indows, 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);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
var
masofa : integer;// масофа, метрда
t: real;// вақт ҳақиқий сон кўринишида
min : integer;// вақт, минутлар
sek : integer;// вақт, секундлар
v: real;// тезлик
begin
// киритиш майдонидан бошланғич маълумотларни олиш
masofa := StrToint(Edit1.Text); t := StrToFloat(Edit2.Text);
// дастлабки алмаштиришлар
min := Trunc(t);// минутлар — t сонининг бутун қисми
sek := Trunc(t*100) mod 100;
// секундлар — t сонининг каср қисми
// хисоблаш
v := (masofa/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/соат';
end;
end.
```

```
// Якунлаш тугмаси босилганда
procedure TForm1.Button2Click(Sender: TObject)
begin
Form1.Close;
end;

end.
```

Дастурни ишга тушириш

Модуль ***unit*** сўзи билан бошланади. Ундан кейин модул номи кўрсатилади. Бу ном матни 3-листингда берилган илованинг асосий модулида эслатиб ўтилади.

Модуль интерфейс, реализации, инициализация каби бўлимлардан иборат бўлади.

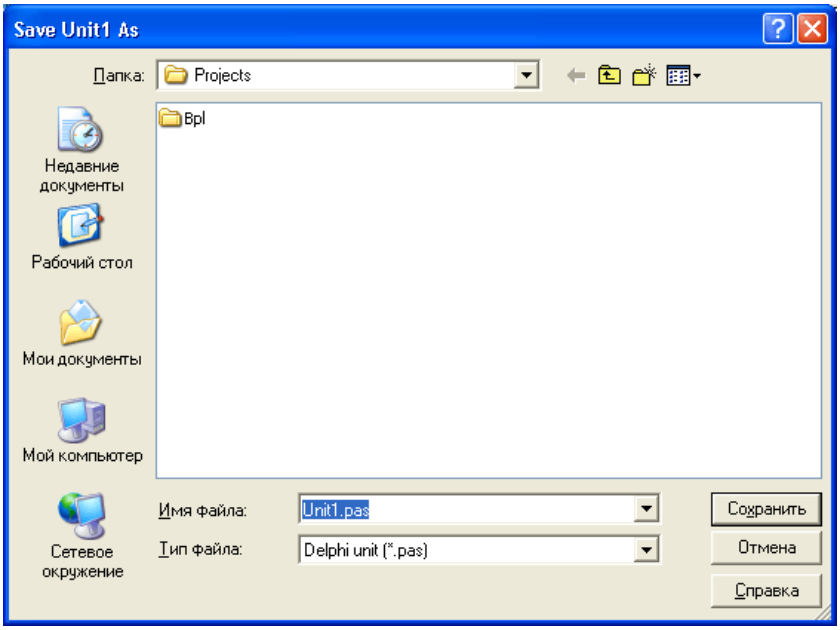
Интерфейс бўлими (***Interface*** сўзи билан бошланади) компиляторга модулнинг қайси қисми дастурнинг бошқа модуллар учун зарур бўлиши ҳақида ахборот беради. Бу бўлимда (***uses*** сўзидан кейин) шу модулда фойдаланиладиган модуллар кутубхоналари рўйхати санаб ўтилади. Шунингдек, бу ерда Delphi да яратилган форма ***type*** сўзидан кейин кўрсатилади.

Реализация бўлими ***Implementation*** билан бошланади ва яратилган форма учун зарур бўлган локал ўзгарувчилар, процедуралар ва функциялар эълон қилинади. Бўлим {$R *.DFM} директиваси билан бошланади. У компиляторга бажариладиган файлни яратишда формадаги маълумотлардан фойдаланиш ҳақида кўрсатма беради. Форма номи модул номи билан бир ҳил, аммо кенгайтмаси ***.dfm*** бўлган файлда сақланади. Бу файл Delphi муҳитида форманинг ташқи кўриниши асосида генерация қилинади.

{$R *.DFM} директивасидан кейин форма ва унинг компоненталари ходисаларни қайта ишлаш процедуралари келтирилади. Бу ерга дастурчи бошқа процедура ва функцияларни ҳам киритиши мумкин. Инициализация бўлими модулдаги ўзгарувчиларни инициализация қилади. Инициализация бўлими реализация (барча процедура ва функцияларни ифодалаш) бўлимидан кейин ***begin*** ва ***end*** лар орасида жойлашади. Агар инициализация бўлими ўз ичига хеч қандай кўрсатмани олмаса, (худди келтирилган мисолдаги каби), у ҳолда ***begin*** сўзи кўрсатилмайди. Шуни таъкидлаш керакки, модулнинг каттагина хажмдаги буйруқларини Delphi нинг ўзи яратади. Масалан, Delphi, дастурчининг форма яратиш бўйича хатти-харакатларини таҳлил қилиб, формадаги объектлар ҳақидаги маълумотларни (***type*** сўзидан кейин) генерация қилади.

Лойихани сақлаш. Лойиха – бу компилятор бажариладиган файлни (EXE-файли) яратиши учун зарур бўлган файллар тўпламидан иборат. Энг оддий мисолда лойиха лойиха ҳақидаги маълумотлардан иборат файл (DOF-файли), асосий модул файли (DPR-файли), ресурслар файли (RES-файли), форма ҳақидаги маълумотлардан иборат файл (DFM-файли), илованинг асосий кодлари жойлашган форма модулининг файли, шунингдек форманинг компоненталари учун ходисаларни қайта ишлаш процедура ва функциялари (PAS-файллар) ҳамда конфигурация файли (CFG-файли) дан иборат бўлади.

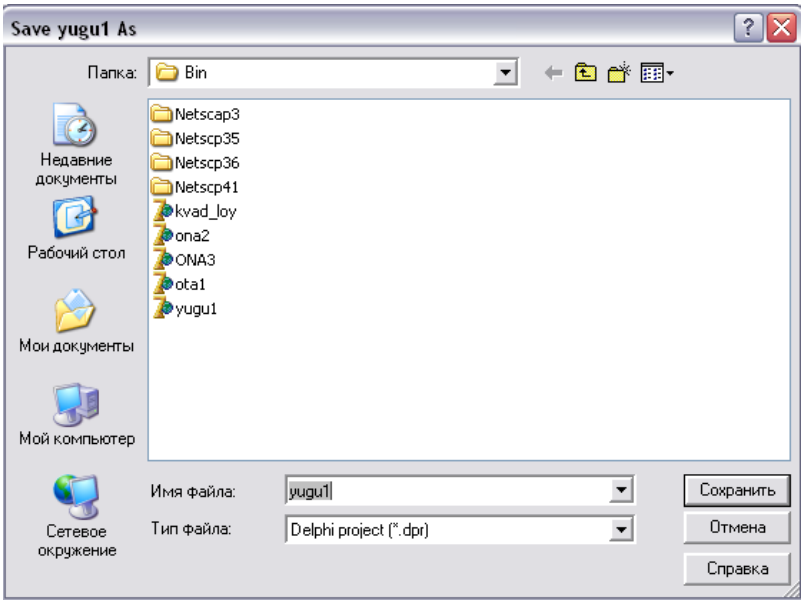
Лойихани сақлаш учун **File** менюсидан **Save Project As** буйруғини танлаш лозим. Агар лойиха бирор марта ҳам сақланмаган бўлса, у ҳолда Delphi дастлаб модулни (кодлар муҳаррири ойнасидаги маълумотларни) сақлаб қўйишни таклиф этади. Шунинг учун экранда **Save Unit1 As** ойнаси пайдо бўлади. Бу ойнада (32-расм) лойиха файллари учун ажратилган папка ва модул номини кўрсатиш лозим. **Сохранить** тугмаси босилганидан кейин экранда навбатдаги ойна (33-расм) пайдо бўлади. Унда лойиха файли номи кўрсатилади.



32-расм. Форма модулини сақлаш

Модул файли (pas-файл) ва лойиха файлининг (dpr-файл) номлари ҳар ҳил бўлишига эътибор беринг. Бажариладиган файл (EXE-файл) номи лойиха файлининг номи билан бир ҳил. Шунинг учун лойиха файлига ном танлаганда бажариладиган файлнинг номини ҳам ҳисобга олиш зарур. Модулга эса номни бошқача, масалан, лойиха файли номига тартиб номерларини қўшиш орқали танлаш мумкин.

Эслатма: Лойиха – бу файллар тўпламидан иборат бўлгани учун, ҳар бир лойихани алоҳида папкада сақлаш тавсия қилинади.



33-расм. Лойихани сақлаш

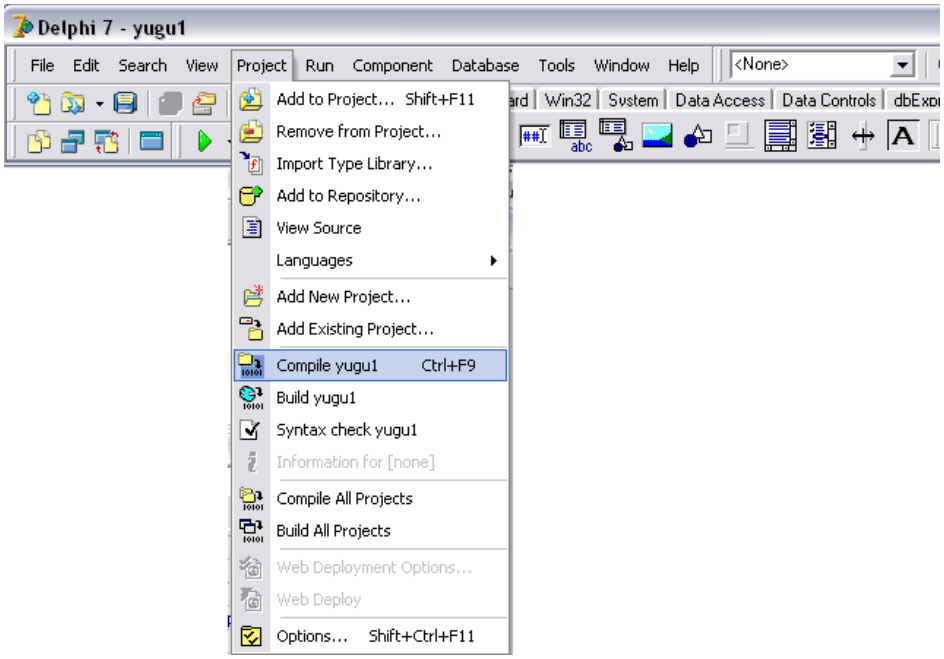
менюга

1.6. Компиляция

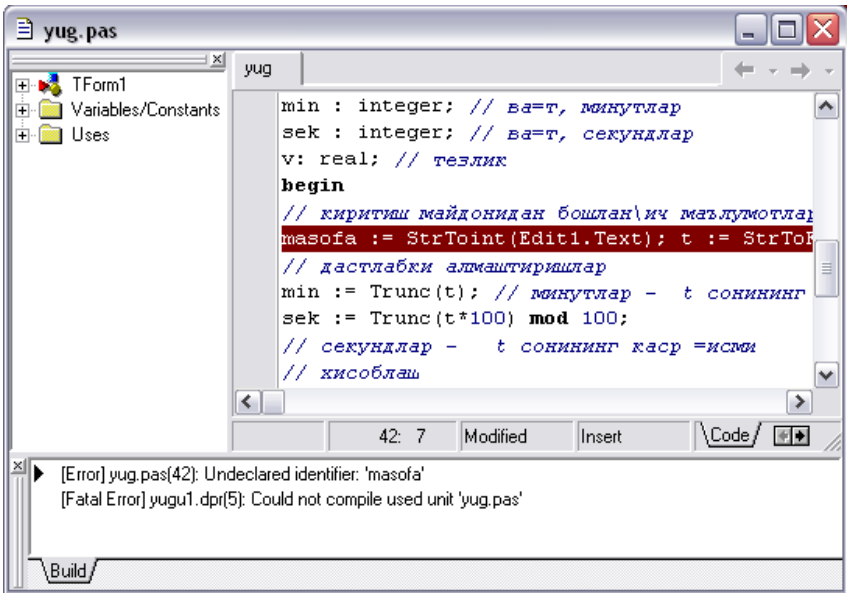
Компиляция — бу бошланғич дастурни бажариладиган файлга айлантириш жараёнидир. Компиляция жараёни икки босқичдан иборат. 1-босқичда дастур матнининг ҳатосиз ёзилганлиги текширилади, иккинчисида эса бажариладиган файл (exe-файл) генерация қилинади.

Ходисаларни қайта ишлаш функцияси яратилиб, сақланганидан сўнг, **Project** менюсидан **Compile** буйруғини танлаб компиляцияни бажариш мумкин.. Компиляция жараёни ва натижаси **Compiling** диалог ойнасида (34-расм) кўрсатилади. Бу ойнага компилятор аниқлаган ҳатоликлар (Errors), огоҳлантириш (warnings) ҳамда эслатмалар (Hints) чиқарилади. Ҳатолик, эслатма ва огоҳлантиришлар кодлар муҳаррири ойнасининг қуйи қисмида берилади. (35-расм).

Эслатма: Агар компиляция вақтида экранда **Compiling** ойнаси кўринмаса, у ҳолда **Tools** менюсидан **Environment options** буйруғини танланг. **Preferences** пунктидаги **Show compiler progress** ўчиргичини ёқилган ҳолатга ўтказинг.



35-расм. Компиляцияни бошлаш

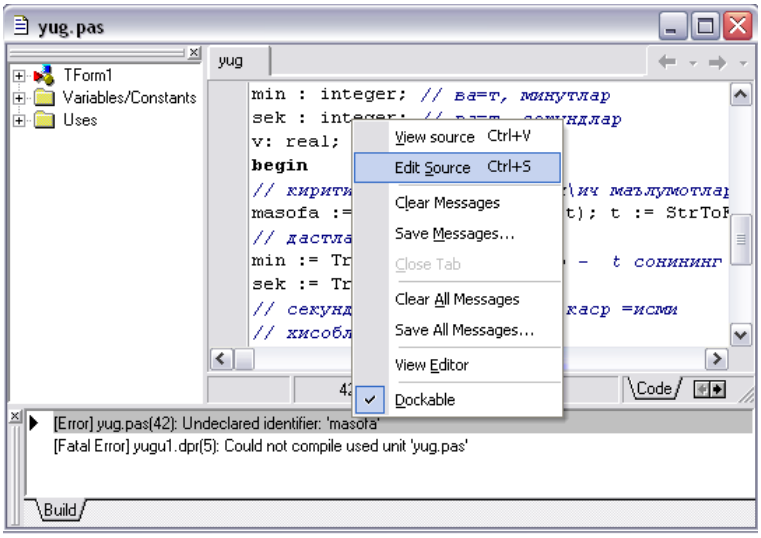


35-расм. Компилятор аниқлаган ҳатоликлар ҳақидаги ахборот

Ҳатоликлар. Компилятор бажариладиган файлни фақат дастур матнида бирорта ҳам синтактик ҳатолик бўлмагандагина яратади. Кўпинча, хозиргина ёзилган дастур матнида ҳатоликлар мавжуд бўлади. Дастурчи уларни бартараф қилиши лозим.

Хатолик мавжуд бўлган дастур парчасига ўтиш учун курсорни ҳатолик ҳақидаги ахборот устига келтириб, контекст менюсидан (36-расм) **Edit source** буйруғини танлаш керак.

Ҳатоликлар бирин-кетин йўқотиб борилади. Ҳар бир ҳатолик йўқотилгандан кейин, такрорий компиляция ўтказилади. Компилятор одатда ҳатолик мавжуд бўлган парчани аниқ кўрсатмаслиги мумкин. Бу ҳолда компилятор кўрсатган парчани таҳлил қилиш билан чегараланиб қолмай, парчадан олдинги сатрга ҳам эътибор қаратиш лозим.



36-расм. Ҳатолик мавжуд бўлган парчага ўтиш

10-жадвалда энг кўп учраши мумкин бўлган ҳатоликлар ва компиляторнинг уларга мос равишда берадиган ахборотлари санаб ўтилади.

Компиляторнинг хатоликлар ҳақидаги ахбороти 10-жадвал

Ахборот	Мумкин бўлган сабаб
Missing operator or semicolon (оператор ёки нуқтали вергул етишмаяпти)	Буйруқдан кейин нуқтали вергул қўйилмаган
Undeclared identifier: '...'	'...' ўзгарувчи эълон қилинмаган

Агар компилятор етарлича кўп ҳатоликларни аниқлаган бўлса, дастлаб энг оддий ҳатоликларни бартараф этинг ва такрорий компиляция ўтказинг. Ҳатоликлар сони анчагина камайиши керак. Чунки, кичик бир ҳатолик ортидан унга боғлиқ бўлган кўплаб ҳатоликлар келиб чиқиши мумкин.

Агар дастур матнида синтактик ҳатоликлар мавжуд бўлмаса, у ҳолда компилятор дастурнинг бажариладиган файлини яратади. Унинг номи лойиха файли номи билан бир хил, кенгайтмаси эса — .exe бўлади. Delphi бажариладиган файлни лойиха файли сақланган папкада сақлайди.

Огоҳлантириш ва эслатмалар. Дастур матнида ҳато бўлмаган ноаниқликлар мавжуд бўлса, компилятор экранга эслатма (Hints) ва огоҳлантиришлар (warnings) чиқарилади. Масалан, дастур матнида эълон қилинган, аммо фойдаланилмаган ўзгарувчилар ҳақидаги эслатма энг кўп учрайди:

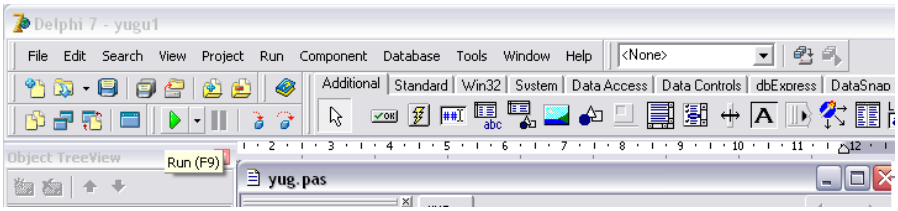
Variable ... is declared but never used in ...

Ҳақиқатдан ҳам, фойдаланилмаган ўзгарувчини эълон қилишнинг нима кераги бор? 11-жадвалда энг кўп учрайдиган огоҳлантиришлар келтирилган.

Компиляторнинг огоҳлантиришлари 11-жадвал

Огоҳлантириш	**Мумкин бўлган сабаби**
Variable... is declared but never used in ...	Ўзгарувчидан фойдаланилмаган
Variable . . . might not have been initialized.	Ўзгарувчига бошланғич қиймат берувчи буйруқ етишмайди. (инициализация қилинмаган ўзгарувчидан фойдаланилмоқда

Дастурни ишга тушириш. Delphi муҳитидан туриб ҳам дастурни ишга тушириш мумкин. Бунинг учун **Run** менюсидан **Run** буйруғини танлаш ёки **Debug** қуроллар панелидаги махсус тугмани лозим (37-расм).



37-расм. Дастурни ишга тушириш

менюга

1.7. Бажариш вақтидаги ҳатоликлар

Дастурнинг ишга туширилганда **бажариш вақтидаги ҳатолиги** (run-time errors) ёки **чиқариш ҳатолиги** (exceptions) деб аталадиган ҳатоликлар юзага келиши мумкин. Бунга кўпинча нотўғри бошланғич маълумотлар сабаб бўлади. Масалан. Югуриш тезлигини ҳисоблаш дастури учун **Вақт** майдонига 3.20 матни, (яъни бутун ва каср қисмини ажратишда вергул ўрнига нуқта қўйилган бўлса) киритилган бўлса, у ҳолда **Хисоблаш** тугмаси босилганда экранда ҳатолик ҳақида ахборот пайдо бўлади (дастур Windows муҳитидан туриб ишга туширилган): (38-расм).

38-расм. Бажариш вақтидаги ҳатолиги

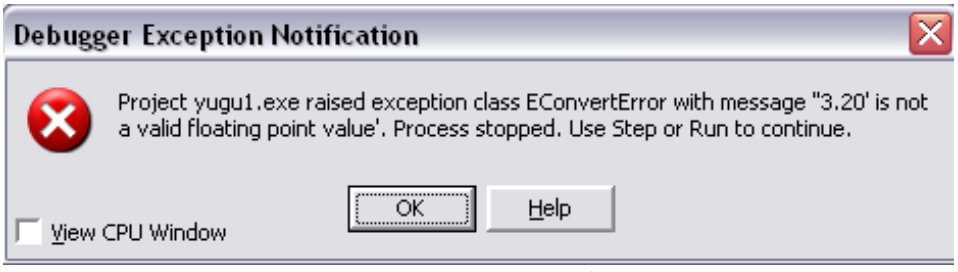
Ҳатоликнинг юзага келишининг асосий сабаби дастур матнида соннинг бутун ва каср қисми нуқта билан, киритиш ойнасида эса одатда вергул (Windows нинг айрим версиялари нуқта билан ажратишга рухсат беради) билан ажратилишидадир.

Агар Windows ни бутун ва каср қисмни вергул билан ажратишга созланган бўлса-ю, фойдаланувчи диалог ойнасида масалан, 3.20 сатрини киритган бўлса, у ҳолда

t: =StrToFloat(Edit2.Text)

буйруғини бажаришда мослик йўқолади, чунки ***strToFloat*** функциясининг қиймати ҳақиқий соннинг ифодаси бўлмай қолади .

Агар дастур Delphi муҳитидан ишга туширилиб, мослик йўқолган бўлса, дастурнинг иши тўхтайди ва экранда хатолик ва унинг характери ҳақидаги ахборот пайдо бўлади. Масалан, 39-расмдаги ахборотда фойдаланувчи киритган соннинг ҳақиқий сон эмаслиги ҳақида маълумот берилмоқда.



39-расм. Ҳатолик ҳақидаги ахборотга мисол

Дастурчи ОК тугмасини босиб, ўз ишини давом эттириши (бунинг учун **Run** менюсидан **Step Over** буйруғини танлайди**) ёки** дастурнинг бажарилишини тўхтатиши (**Run** менюсидан **Program Reset** буйруғи танланади) мумкин.

Дастурни ишлаб чиқишда дастурчи фойдаланувчиларнинг ҳатоликка олиб борувчи барча хатти-харакатларини ҳисобга олиши ва дастурни улардан химоя қилишни таъминлаши зарур.

5-листингдаги югуриш тезлигини хисоблаш дастурида фойдаланувчининг тўғри бўлмаган айрим хатти-харакатларидан дастурни химоя қилиш амалга оширилган. Ҳусусан, масофа (Edit1) майдонига фақат рақамлар киритилиши таъминланган.

Ўзгаришлар киритиш. Югуриш тезлиги дастурини бир неча марта ишга туширилганидан сўнг, унинг матнига ўзгартириш киритишга ҳохиш пайдо бўлиши мумкин. Масалан, дастурни шундай ўзгартириш керакки, масофани киритиб, <Enter> тугмаси босилганда, курсор **Вақт** майдонига ўтсин. Ёки **Масофа** ва **Вақт** майдонларига фойдаланувчи фақат рақамларни кирита олсин.

Дастур матнига ўзгартириш киритиш учун, дастлаб Delphi ни ишга тушириб, ўзгартириладиган лойиха очилади. Буни **File** менюсидан **Open Project** буйруғини танлаш орқали амалга ошириш мумкин. **Reopen** буйруғи танланса, дастурчи ишлаган охирги лойихалар рўйхати очилади.

5-листингдаги Югуриш тезлиги дастури матнига Edit1 ва Edit2 компоненталари учун **OnKeyPress** ходисаларни қайта ишлаш процедураси қўшилган.

Дастур матнига ходисаларни қайта ишлаш процедурасини қўшиш учун **Object Inspector** ойнасидан ходисаларни қайта ишлаш процедураси яратиладиган компонента танланади. Сўнгра **Events** бўлимидан ходисани танлаб, процедура номи майдонида сичқонча икки марта чертилади. Delphi ходисаларни қайта ишлаш процедураси шаблонини яратади. Шундан кейин процедура буйруқларини киритиш мумкин.

5-листинг. **Югуриш тезлиги** дастурининг модули ўзгаришлар киритилганидан кейин қуйидагича бўлади.

```
interface

uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls;

type
TForml = class(TForm) Editl: TEdit;
Edit2: TEdit; Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
Label4: TLabel;
Buttonl: TButton;
Button2: TButton;

procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure EditlKeyPress(Sender: TObject; var Key: Char);

private
{ Private declarations } public
{ Public declarations } end;

var
Form1: TForm1;

implementation
{$R *.dfm)

// Хисоблаш тугмаси босилганидан кейин

procedure TForm1.ButtonlClick(Sender: TObject);
var
masofa : integer // масофа, метрларда
t: real;// вақт ҳақиқий сон кўринишида
min : integer;// вақт, минутлар
sek : integer;// вақт, секундлар
v: real;// тезлик
begin
// киритиш майдонидан бошланғич маълумотларни олиш
masofa := StrToint(Edit1.Text); t := StrToFloat(Edit2.Text);
// дастлабки алмаштиришлар
min := Trunc(t);// минутлар — t сонининг бутун қисми
sek := Trunc(t*100) mod 100;
// секундлар — t сонининг каср қисми
// хисоблаш
```

```
v := (masofa/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/соат';
end;

// Якунлаш тугмаси босилганида
procedure TForml.Button2Click(Sender: TObject);
begin
Form1.Close;
end;

// Масофа майдонида тугма босилганда

procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);

begin

// Key —босилган тугмага мос келувчи белги
// Агар мумкин бўлмаган белги киритилган бўлса, процедура уни
// коди 0 бўлган белги билан алмаштиради. Натижада
// фойдаланувчида дастур айрим тугмаларни босилишига эътибор
// бермас экан деган тасаввур пайдо бўлади
case Key of
'0'..'9': ; // рақамлар
#8 : ; // ўчириш клавишаси <Back Space>
#13 : Edit2.SetFocus ; // <Enter> клавишаси
// қолган белгиларни киритиш таъқиқланади
else Key :=Chr(0); // белгини кўрсатмаслик
end;
end;
end.
```

Дастурни ишга тушириш

Ўзгаришлар киритилганидан сўнг, лойихани сақлаш лозим. Бунинг учун **File** менюсидан **Save all** буйруғи танланади.

менюга

1.8. Иловани якуний созлаш

Дастур ҳамма талабларга жавоб берадиган бўлганидан сўнг, уни якуний созлаш, яъни дастурга ном ва нишон тайинлаш лозим. Бу ном ва нишон папкадаги файллар рўйхати орасида, ишчи столда, дастур ишлаётган бўлса масалалар панелида кўриниб туради.

Иловани созлаш **Project** менюсидан **Options** буйруғи танланганда очиладиган **Project Options** диалог ойнасининг **Application** пункти ёрдамида бажарилади. (40-расм),

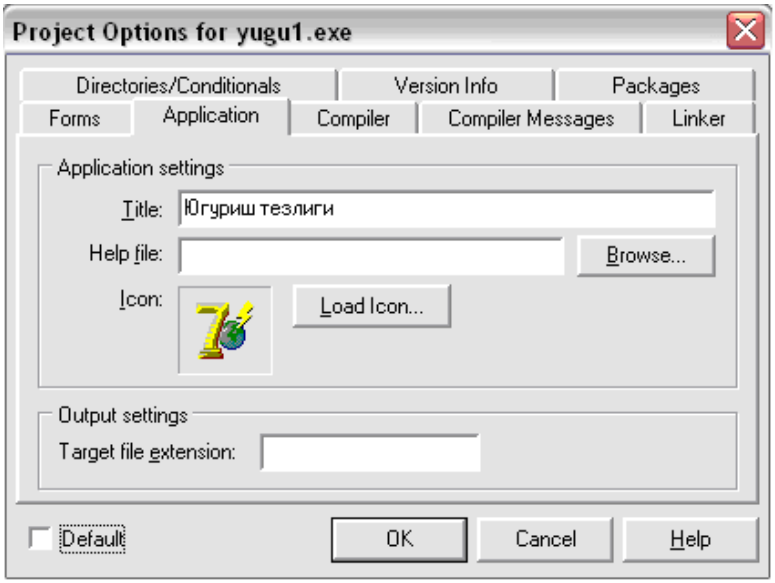
Title майдонига илова номи ёзилади. Бу майдонга киритилган матн Windows масалалр панелида, ишлаётган дастур нишони билан ёнма-ён чиқарилади.

Иловага стандарт бўлмаган нишон танлаш учун **Load Icon** тугмаси босилади. Сўнгра папкалар кўришнинг стандарт ойнаси ёрдамида дастурга мос келадиган нишон қидирилади. (Нишонлар .**ico** кенгайтмали файлларда сақланади).

Илова учун нишон яратиш. Delphi таркибига **Image Editor** (тасвир муҳаррири) кирган. У дастурчига иловалар учун ўзининг нишонини яратишга имкон беради. **Image Editor** дастури **Tools** менюсидан ёки Windows муҳитида — **Пуск / Программы Borland Delphi 7 / Image Editor** буйруқлари билан ишга туширилади.

Янги нишон яратиш учун **File** менюсидан **New** буйруғи очадиган рўйхатдан **Icon File** тугмаси танланади. Яратилаётган файлнинг типи кўрсатилганидан сўнг, **Icon Properties** ойнаси очилади. Унда янги нишоннинг аломатлари белгиланади: size (ўлчами) — 32x32 (Windows нишонларининг стандарт ўлчами) ва Colors (ранглар) — 16 ҳил ранг. **ОК** тугмаси босилгандан кейин **Icon1.ico** ойнаси очилади. Унда стандарт қуроллар ва ранглар ёрдамида керакли нишонни чизиш мумкин.

Image Editor да расм чизиш Microsoft Paint дан фарқ қилмайди. Аммо бу ерда бир "нозик жой" бор. Дастлаб тасвир майдони шаффоф (transparent) ранг билан бўялган бўлади. Агар нишонни шу фонда чизилса, уни кейинчалик экранга чиқарилганда, шаффоф рангга бўялган қисми нишон жойлашадиган фон рангини қабул қилади.



40-расм. Application пункти ёрдамида дастур учун ном ва нишон танланади

Расмни чизиш жараёнида ҳато чизилган элементларни шаффоф рангга бўяш орқали "ўчириш" мумкин. Унга ранглар палитрасининг қуйи қаторидаги чап квадрат тўғри келади. Шаффоф рангдан ташқари, ранглар палитрасида "инверсли" (қарама-қарши) ранг ҳам мавжуд. Бу ранг билан чизилган расмлар экранга чиқарилганда фон рангига нисбатан инверсион рангга бўялади.

Яратилган нишон **File** менюсидаги **Save** буйруғи билан сақлаб қўйилади.

менюга

1.9. Иловаларни бошқа компьютерга ўтказиш

Битта EXE-файлдан иборат бўлиб, фақат стандарт компоненталардан фойдаланадиган унчалик катта бўлмаган иловаларни дискеталар ёрдамида бошқа компьютерларга ўтказиш мумкин. Одатда, бошқа компьютерларда бу иловани муаммоларсиз ишга тушириш мумкин.

Модуллар кутубхоналари, драйверлар ва бошқа дастурий компоненталарни ўз ичига олган иловаларни бошқа компьютерларга ўтказиш мураккаброқ. Бундай иловалар учун ўрнатувчи диск (CD-ROM) яратиш мақсадга мувофиқ бўлади. Бу ишни Delphi таркибига кирган **InstallShield Express** пакети ёрдамида ҳал қилиш мумкин. Ўрнатувчи дисклари яратиш жараёни ҳақида кейинги бобларда тўхталамиз.

2-боб. ДАСТУРЛАШ АСОСЛАРИ

2.1. Дастурларни ишлаб чиқиш босқичлари

Дастурлаш — дастур ишлаб чиқиш (яратиш) жараёни бўлиб, қуйидаги қадамлар кетма-кетлиги орқали ифодаланиши мумкин:

1. Спецификация (дастур ва унга бўладиган талабларни аниқлаш).
2. Алгоритмини қуриш.
3. Кодлаш (алгоритмни дастурлаш тилида ифодалаш).
4. Дастур матнидаги мавжуд хатоларни аниқлаш ва бартараф этиш.
5. Тестдан ўтказиш.
6. Эслатмалар (справка) системасини яратиш.

7. Дастурни ўрнатиш дискини яратиш (CD-ROM).

Спецификация. Дастурга қўйиладиган талабларни аниқлаш дастур ёзишдаги энг муҳим босқичлардан бири бўлиб, унда берилган маълумотлар батафсил ифодаланади, натижага бўлган талаблар аниқланади, айрим ҳолларда дастурнинг ҳулқи белгиланади (масалан, нотўғри маълумотлар киритилганда), фойдаланувчи ва компьютер ўртасидаги мулоқот ойнаси ишлаб чиқилади.

Алгоритмни ишлаб чиқиш. Бу босқичда ечилаётган масаланинг натижасини олиш учун бажариш лозим бўлган амаллар кетма-кетлигини аниқлаш зарур бўлади. Агар масала бир нечта усуллар билан ҳал қилиниши, ёки натижаларнинг бир неча вариантларда олиш мумкин бўлса, дастурчи бирор бир критерияга (масалан, алгоритмнинг бажарилиш тезлигига) асосланган ҳолда, энг мақбул ечим ёки усулни танлайди. Алгоритмни ишлаб чиқиш натижасида масала ечиш йўлининг сўзлар ёки блок-схема орқали ёзилган батафсил ифодаси (алгоритми) ҳосил қилинади.

Кодлаш. Бу алгоритм танлаб олинган бирор дастурлаш тилида қабул қилинган қонун-қоидалар ёрдамида ёзилади. Натижада шу дастурлаш тилидаги дастур юзага келади.

Дастур матнидаги мавжуд хатоликларни аниқлаш ва бартараф этиш. Дастурдаги хатоликлар икки гуруҳга бўлинади: синтактик (матндаги) ва алгоритмик ҳатоликлар. Синтактик ҳатоликлар энг осон тўғриланадиган ҳатоликлар ҳисобланади. Алгоритмик ҳатоликларни аниқлаш эса мураккаброқ. Дастур матнидаги мавжуд хатоликларни аниқлаш ва бартараф этиш жараёни бошланғич киритиладиган маълумотлар учун дастур тўғри натижа берганидан кейингина тугалланган деб ҳисобланиши мумкин.

Тестдан ўтказиш. Агар дастур бошқа фойдаланувчилар учун ёзилган бўлса, тестдан ўтказиш босқичи жуда ҳам муҳим бўлади. Бунда турли ҳил бошланғич маълумотлар учун, шу жумладан нотўғри малумотлар ҳам киритилганда, дастур ўзини қандай тутиши аниқланади.

Маълумотлар системасини яратиш. Агар дастур бошқа фойдаланувчилар учун ёзилган бўлса, дастурдан фойдаланиш бу фойдаланувчиларга қулай бўлиши учун йўриқнома ва эслатмалар ишлаб чиқилиши шарт. Бу эслатмаларга дастур билан ишлаш жараёнида осон мурожаат қилишни ташкил этиш керак. Замонавий дастурий таъминотда бундай маълумотномалар CHM ёки HLP кўринишларидан бирида ифодаланади. Агар дастурни ўрнатиш (инсталляция) талаб қилинадиган бўлса, у ҳолда TXT, DOC ёки HTM форматларидан биридаги йўриқнома ҳам ёрдамчи маълумотномалар системасига кириши зарур.

Ўрнатувчи дискни яратиш. У фойдаланувчи дастурчининг ёрдамсиз ҳам ўз компьютерига мустақил равишда дастурни ўрната олиши учун мўлжалланади. Одатда ўрнатувчи дискда дастурдан ташқари, ёрдамчи маълумотномалар системасининг файли, дастурни ўрнатиш йўриқномаси (**Readme** файли) ҳам мавжуд бўлади. Шуни назарда тутиш керакки, замонавий дастурлаш тилида ёзилган дастурлардан уларни компьютерга тўғридан-тўғри кўчириб олиб, фойдаланиш мумкин эмас. Бунинг сабаби шуки, дастур иши учун зарур бўлган махсус кутубхона ва компоненталарнинг тўлиқ таркиби бу фойдаланувчининг компьютерида бўлмаслиги мумкин. Шунинг учун махсус дастур ёрдамида дастур ва унинг барча компоненталари ўрнатувчи дискдаги махсус дастур ёрдамида фойдаланувчининг компьютерига ўрнатилиши лозим. Одатда, ўрнатувчи дастур алоҳида папка очиб, унга барча керакли файл ва компоненталарни кўчиради. Бундан ташқари, зарур бўлса, реестрларга қўшимчалар ва ўзгартиришлар киритиш орқали операцион тизимга ҳам ўзгартиришлар киритиши ва созлаши мумкин.

менюга

2.2. Алгоритм ва дастур

Дастур яратишнинг биринчи босқичида дастурчи қўйилган масалани тўла ҳал қилиш учун бажарилиши зарур бўлган амаллар кетма-кетлиги ва ундаги тартибни аниқлайди, яъни алгоритм қуради. **Алгоритм** - ечилаётган масала доирасида бошланғич маълумотлардан натижага ўтиш жараёнини ифодаловчи аниқ кўрсатмалардир.

Таъриф: Алгоритм деб қўйилган масалани тўла ҳал учун бажарилиши зарур бўлган амаллар кетма-кетлигининг қатъий тартибига айтилади.

Масаланинг ечиш алгоритми сўзлар орқали, махсус математик формулалар ёки махсус блок-схема деб аталувчи махсус график кўринишда ифодаланиши мумкин.

1-мисол. Кўчани хавфсиз кесиб ўтиш қоидаси.
1. Йўлнинг четига келиб тўхтанг.
2. Йўлнинг чап томонига қаранг.
3. Агар чап томонда транспорт воситалари яқин келиб қолган бўлса, ўтиб кетгунча кутинг.

4. Чап томонингизда транспорт воситалари қолмаган бўлса, йўлнинг ўртасига ўтиб тўхтанг.
5. Йўлнинг ўнг томонига қаранг.
6. Агар ўнг томонда транспорт воситалари яқин келиб қолган бўлса, ўтиб кетгунча кутинг.
7. Ўнг томонингизда транспорт воситалари қолмаган бўлса, йўлнинг қолган қисмини кесиб ўтинг.

Шунингдек, ихтиёрий дориларни тайёрлаш йўллари, овқатларни тайёрлаш усуллари, хаким белгилаган дориларни истеъмол қилиш, банкомётдан пул олиш каби амалларни алгоритм сифатида қабул қилиш мумкин. Алгоритмларга ҳаётий ва турли фан соҳаларидаги масалаларни ечиш йўллари ҳам киради.

Алгоритмларга қуйидаги талаблар қўйилади :

1. Бошланиши ва тугаши кўрсатилиши керак.
2. Ҳар қандай амал буйруқ тарзида ифодаланиши шарт.
3. Ҳар бир амал ижрочига тушунарли бўлган кўринишда ифодаланган бўлиши шарт.
4. Ҳар бир амалда қатнашаётган ўзгарувчиларнинг қийматлари олдиндан аниқланган бўлиши керак.
5. Ҳар қандай амал натижаси бир қийматли бўлиши керак.
6. Бажариладиган амаллар сони чекланган бўлиши керак.
7. Якуний натижаларни ажратиб кўрсатиш ва чиқариш шарт.
8. Масалани тўла ечиш учун берилган ҳамма маълумотлар ва мумкин бўлган барча имкониятлар ҳисобга олинган бўлиши керак.
9. Алгоритм оммавий, яъни битта синфга таалуқли бўлган кўплаб масалаларни ечишга мўлжалланган бўлиши керак.

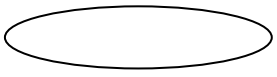
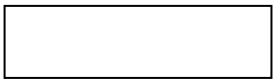
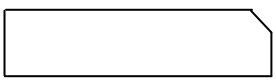
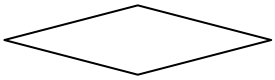
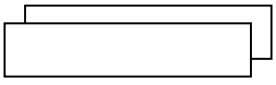
Юқоридаги талабларнинг бирортаси бузилган бўлса, қўйилган масалани ечиш учун қурилган алгоритм тўлақонли бўла олмайди, яъни масаланинг тўла ечимини бера олмайди. Масалан: Агар бирон бир амални бажаришда қатнашаётган ҳар бир ўзгарувчининг қиймати олдиндан аниқланмаган (4-талаб) бўлса, у ҳолда ана шу ўзгарувчининг ўрнига одатда нол қўйиб ҳисобланади. Бу эса ҳар доим ҳам тўғри натижа беравермайди. Фараз қилайлик, К-ўзгарувчининг қиймати олдиндан аниқланмаган бўлсин. У ҳолда $D=(A+B)/K$ ифоданинг қийматини ҳисоблашнинг иложи йўқ, чунки K нинг ўрнига компилятор нол қийматини қўяди. Натижада нолга бўлиниш ҳолати рўй беради. Бундай бўлиши эса мумкин эмас. Энди 6-талабни бузиб кўрайлик.

1. *Ҳисоблансин* $i:=1;$
2. *Ҳисоблансин* $i:=i+1;$
3. *1-га ўтилсин.*

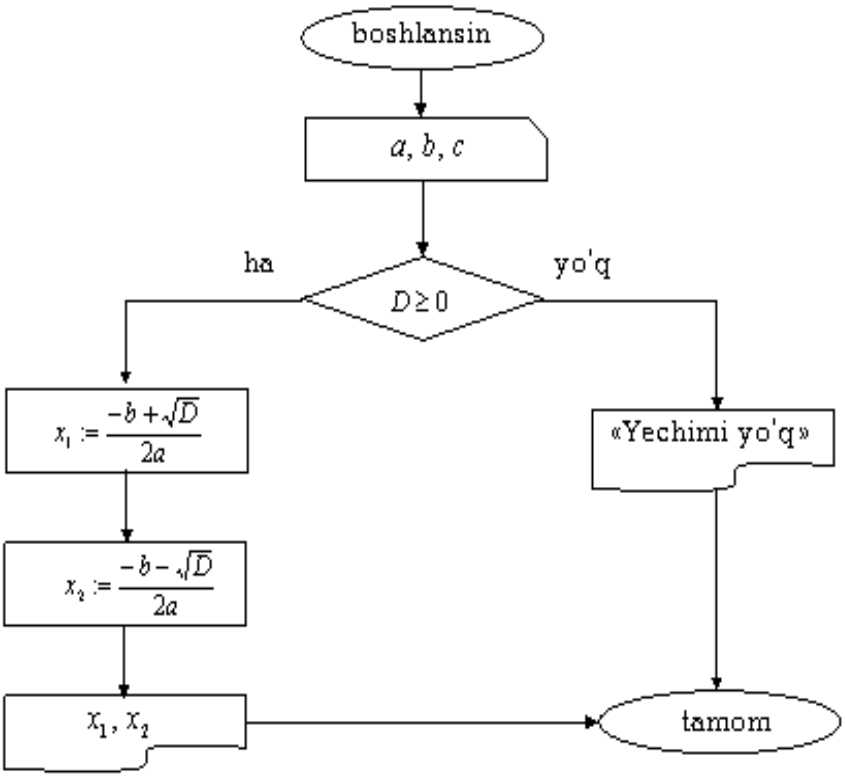
Бу ҳолда қурилган алгоритм «чексиз алгоритм» бўлиб қолади, яни уни «ижрочи» ҳеч қачон тугата олмайди.

Блок-схемалар усулида алгоритмнинг ҳар бир буйруғи махсус геометрик шакллар ёрдамида ифодаланади. Блок-схемаларни қуришда 1-жадвалдаги шакллардан фойдаланиш мумкин.

1-жадвал

Шакл	**Вазифаси**
	алгоритмнинг бошланиши ва охирида қўйилади
	ўзгарувчиларга қиймат бериш
	маълумотларни киритиш
	мантиқий ифодаларни ҳисоблаш
	амалларни бажариш йўналиши
	ёрдамчи алгоритмга мурожаат
	якуний натижаларни чиқариш

2-мисол: $ax^2+bx+c=0$ квадрат тенглама учун блок-схема.

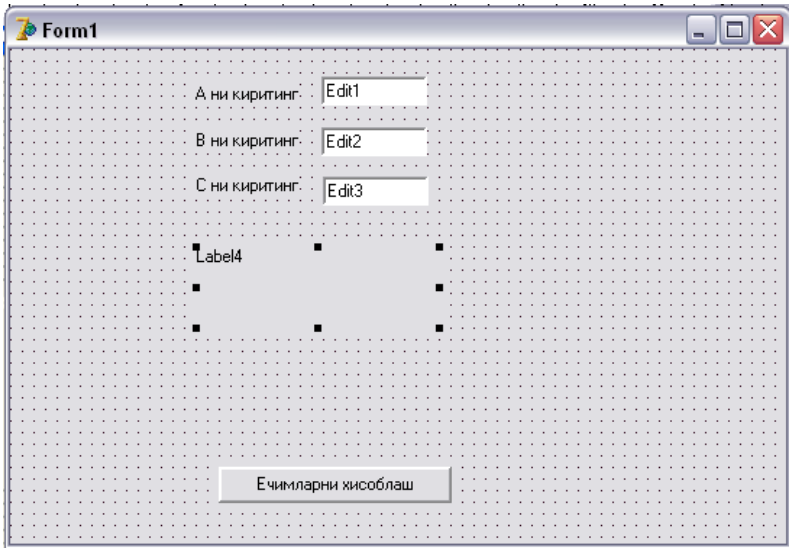


2.1-расм. Квадрат тенгламанинг блок-схемаси

Алгоритмни блок-схема ёрдамида ифодалаш дастурчига амаллар кетма-кетлигини аниқлаш, масалани тўғри тушунганлигига ишонишга имкон беради.

Delphi да ёзилган дастур масаланинг ечиш алгоритмига мос ходисаларни қайта ишлаш процедуралари тўпламидан иборат бўлади.

Мисол тариқасида квадрат тенгламанинг ечиш ташкил қилинган диалог ойнаси ҳамда юқоридаги блок-схемага мос келадиган дастур матнини келтирамиз.



2.2-расм. Квадрат тенглама учун форманинг кўриниши

2.1-листинг. **Квадрат тенгламанинг дастури**

```
unit Unit1;
interface
uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
```

```
    Label2: TLabel;
    Edit2: TEdit;
    Label3: TLabel;
    Edit3: TEdit;
    Label4: TLabel;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
    var a1,b1,c1,d,x1,x2:real;
begin
   a1:=strtofloat(edit1.Text);
   b1:=strtofloat(edit2.Text);
   c1:=strtofloat(edit3.Text);
   d:=b1*b1-4*a1*c1;
   if d>=0 then begin
   x1:=(-b1+sqrt(d))/(2*a1);
   x2:=(-b1-sqrt(d))/(2*a1);
   label4.caption:='x1='+floattostr(x1)+#13+'x2='+floattostr(x2);
   end
   else
   label4.caption:='Тенгламанинг ҳақиқий ечимлари йўқ';
end;
end.
```

Дастурни ишга тушириш

Компиляция. Дастурлаш тилларидан биридаги буйруқлар ёрдамида ёзилган дастур бошланғич дастур деб аталади. Бу буйруқлар инсонга тушунарли, аммо компьютер процессорига тушунарли эмас. Процессор бошланғич дастурни бажара олиши учун уни машина тили – процессор тушунадиган тилга ўтказиш лозим. Бошланғич дастурни машина тилига ўтказишни ***компилятор*** деб аталадиган махсус дастур бажаради.

Ишлаш принципи 2.3-расмда келтирилган компилятор қуйидаги икки вазифани бажаради:

1. Бошланғич дастур матнида синтактик хатоликларнинг бор ёки йўқлигини аниқлайди.

2. Машина кодидаги бажариладиган дастурни хосил (генерация) қилади.

Эслатма: Фақат дастур матнида синтактик хатоликлар мавжуд бўлмагандагина бажариладиган дастур генерация қилинади.

Компилятор томонидан машина кодидаги генерация фақатгина бошланғич дастур матнида синтактик хатолар йўқлигидан далолат беради ҳалос. Дастурнинг тўғри ишлаётганлигига уни синаш

2.3-расм. Компиляторнинг ишлаш схемаси.

учун ишга тушириб, тестдан ўтказиш орқали ишонч ҳосил қилиш мумкин. Масалан, квадрат тенглама илдизини топиш формула-ларидан бирини нотўғри, аммо синтактик ҳатоларсиз ёзилган бўлса, дастур ҳам шунга мос равишда нотўғри натижаларни беради.

менюга

2.3. Delphi дастурлаш тили

Delphi дастурлаш муҳитида дастурларни ёзиш учун Delphi дастурлаш тилида фойдаланилади. Delphi даги дастур операторлар деб аталадиган кўрсатмалар (буйруқлар) кетма-кетлигидан иборат. Бу кўрсатмалар бир-биридан нуқтали вергул (;) белгиси билан ажратилади.

Ҳар бир кўрсатма идентификаторлар комбинациясидан иборат бўлади. Идентификатор қуйидаги маънолардан бирини англатиши мумкин:

- тилнинг кўрсатмалари **(:=,** if, while, for);
- ўзгарувчиларни;
- ўзгармасларни (константалар) (бутун ёки ҳақиқий) ;
- арифметик (+, -, *,/) ёки мантиқий (and, or, not) амалларни;
- қисм дастурни (процедура ёки функцияни);
- процедуранинг бошланиши (procedure, function) ёки тугаши (end) ҳамда блокнинг бошланиши ёки охири (begin, end).

менюга

2.4. Маълумотларнинг типлари

Дастур бутун ва ҳақиқий сонли, белгили, матнли ёки мантиқий типдаги маълумотларни қайта ишлаши мумкин.

Бутун тип. Delphi тили еттита типдаги бутун сонли маълумотларни қабул қила олади: Shortint, Smailint, Longint, Int64, Byte, Word ҳамда Longword

Бутун типдаги сонлар. 2.1-жадвал

типи	диапазони	ўлчами
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 энг универсал ***Integer*** бутун типли маълумотни қабул қилади ҳалос. У ***longint*** типига эквивалент.

Ҳақиқий тип. Delphi тилида олтита ҳақиқий типдаги маълумотлар мавжуд: real48, single, double, extended, comp, currency. Бу типлар бир-биридан қабул қиладиган қийматларининг диапазони, ишончли рақамларининг сони ва компьютер хотирасидан эгаллайдиган хажмлари билан фарқланади.

Ҳақиқий типлар. Жадвал1.2.

Тип	Диапазон	Ишончли рақамлари	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 универсал ҳақиқий тип - Real типини қабул қилади.

Белгили тип. Delphi тилида иккита белгили тип мавжуд: Ansichar и Widechar:

- Ansichar типи — бу ANSI кодидаги белгилар бўлиб, уларга 0 дан 255 гача бўлган сонлар мос келади;
- Widechar типи — Unicode кодидаги белгилар бўлиб, уларга 0 дан 65 535 гача бўлган сонлар мос келади.
- Object Pascal Ansichar белгили типига эквивалент бўлган Char типини ўз ичига олган.

Сатрли тип. Delphi тилига учта сатрли тип киритилган: shortstring, Longstring, WideString:

- Shortstring типи узунлиги 0 дан 255 гача бўлиши мумкин бўлган ва компьютер хотирасидан статистик тарзда жой оладиган сатрдир.
- Longstring типи узунлиги бўш хотира хажми билан чекланадиган ва хотирада динамик тарзда жойлашадиган сатрдан иборат;
- WideString типи узунлиги бўш хотира хажми билан чекланадиган ва хотирада динамик тарзда жойлашадиган сатр кўринишида бўлади. WideString типидаги сатрнинг ҳар бир белгиси Unicode-белгисидан иборат.

Delphi тилида сатрли типларни белгилаш учун String типидан фойдаланилади. У shortstring типига эквивалент.

Мантиқий тип. Мантиқий катталик True (рост) ёки False (ёлғон) қийматларидан бирини қабул қилади. Delphi тилида мантиқий катталиклар Boolean типига мансуб бўлади.

менюга

2.5. Ўзгарувчилар

Ўзгарувчи – бу компьютер хотирасининг бир қисми бўлиб, унда дастур ёрдамида қайта ишланиши талаб қилинган маълумотлар сақланади. Дастур маълумотлар билан иш олиб борар экан, у амалда хотира ячейкасидаги маълумотлар, яъни ўзгарувчилар устида амаллар бажаради.

Дастур ўзгарувчиларга (хотира соҳасига) бирор формула бўйича ҳисоблаш ёки олинган натижаларни сақлаш мақсадида мурожаат қилиши учун, ҳар бир ўзгарувчи ўз номига эга бўлиши керак. Бу номни дастурчи белгилайди.

Ўзгарувчининг номи сифатида лотин харфлари, рақамлар ҳамда айрим махсус белгилар кетма-кетлигидан фойдаланиш мумкин. Номнинг биринчи белгиси харф бўлиши лозим. Ўзгарувчиларнинг

номини белгилашда бўш жой белгисини қўллаш мумкин эмас. Номларни белгилашда Delphi тили учун катта ва кичик харфларнинг фарқи йўқ. *SUMMA, Summa* ва *summa* номлари битта ўзгарувчининг номи сифатида қабул қилинади.

Ўзгарувчини номлашда унинг вазифаси ва номи бир хил бўлиши мақсадга мувофиқ ҳисобланади. Масалан, $ax^2 + bx + c = 0$ кўринишидаги квадрат тенглама коэффициентлари ва илдизларини мос равишда *a, b, c, x1* ва *x2* деб номлаган маъқул. Агар дастурда йиғинди ва умумий йиғиндиларни ҳисоблашга тўғри келса, бу ўзгарувчиларни *um_yigindi* ва *yigindi* тарзида белгилаш тавсия қилинади.

Delphi тилида ҳар бир ўзгарувчидан фойдаланишдан аввал эълон қилиниши лозим. Бу эълон ёрдамида фақат ўзгарувчининг номигина эмас, балки у қабул қиладиган маълумотларнинг типи ҳам кўрсатилиши лозим. Ўзгарувчилар умумий кўринишда қуйидагича эълон қилинади:

Ном : тип;

Бу ерда ***Ном*** – ўзгарувчининг номи, ***тип*** – шу ўзгарувчи қабул қиладиган маълумотларнинг типи. Масалан:

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

Келтирилган мисолда иккита *real* ва битта *integer* типидаги ўзгарувчилар эълон қилинган.

Агар дастурда битта типга мансуб бир нечта ўзгарувчилар қатнашса, уларни битта кўрсатма орқали ҳам эълон қилиш мумкин, масалан:

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

менюга

2.6. Константа (ўзгармас) лар

Delphi тилида икки турдаги константалар мавжуд: оддий ва номланган. Оддий константа деганда бутун ёки ҳақиқий сон, белгилар кетма-кетлиги, алоҳида белги ёки мантиқий қиймат тушунилади.

Бутун константалар дастур матнида ҳаётдаги каби ёзилади:

123 0 -234

Ҳақиқий константаларнинг бутун ва каср қисмини ажратиб кўрсатиш учун одатдаги "вергул" ўрнига "нуқта" белгиси қўлланади:

0.0 23.45 -524.03 -0.124

Ҳақиқий константаларни айрим ҳолларда сузувчи вергуллар орқали ҳам ифодалаш мумкин. Бунда соннинг алгебраик кўриниши, яъни 10 да кичик бўлган ***мантисса*** ҳамда 10 нинг даражасини билдирувчи ***тартиб*** ларнинг кўпайтмасидан фойдаланилади. Масалан:

Сон	**Алгебраик кўриниш**	**Сузувчи вергулли кўриниш**
1 000 000 -123.452 0,0056712	$1x10^6$ $-1,23452x10^2$ $5,6712x10^{-3}$	1 .0000000000E+06 -1 .2345200000E+02 5,6712000000E-03

Матнли ва белгили константалар апострофлар орасида кўрсатилади:

'Delphi дастурлаш тили' 'Delphi 7' '2.4' 'Д'

Бу ердаги '2.4' константаси 2.4 сонини эмас, балки шу сонни ифодаловчи белгилар кетма-кетлигини англатади.

Мантиқий константалар (True) рост ёки (False) бўлиши мумкин.

Номланган константа – бу ном (идентификатор) бўлиб, дастурда бирор константани кўрсатади. Номланган константа ҳам ўзгарувчилар каби фойдаланишдан аввал эълон қилиниши лозим. Бу иш умумий кўринишда қуйидагича амалга оширилади:

константа = қиймат;

Бу ерда ***константа*** — константанинг номи, ***қиймат*** – константанинг қиймати.

Номланган константалар дастурда ***const*** бўлимида эълон қилинади. Масалан:

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

Номланган константа эълон қилинганидан сўнг, бу константа ўрнига унинг номидан фойдаланиш мумкин.

2.7. Қиймат бериш буйруғи

Қиймат бериш буйруғи ёрдамида кўрсатилган формула бўйича ҳисоблаш ишлари бажарилади. Бу буйруқ умумий кўринишда қуйидагича ёзилади:

Ном : = ифода;

Бу ерда ***Ном*** - қиймат бериш буйруғининг бажарилиши натижасида қиймати ўзгарадиган ўзгарувчи; **:=** - қиймат бериш буйруғи белгиси, ***ифода*** – константа, арифметик, матнли ёки мантиқий ифода бўлиб, шу ифода бўйича ҳисоблаш ишлари бижарилади ва олинган натижа **:=** белгисидан чап томонда турган ўзгарувчига қиймат қилиб берилади. Масалан:

```
B := 2.34  D := b*b-4*a*c    Til := 'Delphi 7'    Found := False;
```

Ифода. Ифодалар операнда ва операторлардан ташкил топади. Операторлар операндалар ўртасида кўрсатилади ва улар устида бажариладиган амални билдиради. Операндалар сифатида ўзгарувчилар, константалар, функциялар ва бошқа ифодалардан фойдаланиш мумкин.

Алгебраик операторлар. **1.4.жадвал**

оператор	амал	оператор	амал
=	Қўшиш	/	Бўлиш
-	Айириш	DIV	Бутун сонли бўлиш
*	Кўпайтириш	MOD	Бўлишдаги қолдиқ

DIV оператори бир сонни иккинчисига бўлганда бўлинманинг бутун қисмини англатади. Масалан, 13 DIV 5 амали натижаси 2 га тенг. MOD оператори бир сонни иккинчисига бўлганда пайдо бўладиган қолдиқни англатади. Масалан 13 MOD 5 амали натижаси 3 га тенг.

Ифодаларнинг қийматини ҳисоблашда операторларнинг савиясини ҳисобга олиш лозим. *, /, DIV, MOD операторларининг савиялари + ва - айиришга нисбатан юқори туради. Операторларнинг савияси уларнинг бажарилиш тартибига таъсир кўрсатади. Ифодаларнинг қийматини ҳисоблашда бир нечта операторлар қатнашса, уларнинг бажарилиш тартиби савияларига қараб белгиланади. Агар операторларнинг савиялари бир хил бўлса, дастлаб чап томонда турган оператор бажарилади. Эҳтиёж бўлса. бундай тартибни қавслар ёрдамида ўзгартириш мумкин. Масалан:

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

Қавслар ичидаги ифода битта операнда сифатида қабул қилинади. Қавслар оддий тартибда, аммо қавсдан ташқаридаги операндаларга нисбатан олдинроқ ҳисобланади. Ифодаларни ёзишда Қавслардан фойдаланилганда, қавс жуфтликларининг тўғри бўлишига алоҳида эътибор бериш лозим.

Ифоданинг типи шу ифодага кирган операндалар билан аниқланади. Агар операндалар бутун типда бўлса, қўшиш, айириш, кўпайтириш амалларининг типи ҳам бутун бўлади. Бутун сонлар устида оддий бўлиш амалининг натижаси доимо ҳақиқий бўлади. Агар операндаларнинг бирортаси ҳақиқий бўлса, бу ифоданинг қиймати ҳақиқий ҳисобланади.

Ифодаларнинг типларини аниқлаш 1.5.-жадвал

Оператор	**Операндалар типи**	**Ифоданинг типи**
*, +, -	Агар бирор операнда *real* бўлса	*real*
*, +, -	Ҳар икки операнда *integer*	*integer*
/	*real* ёки *integer*	Доимо *real*

DIV, MOD	Доимо *integer*	Доимо *integer*

Қиймат бериш буйруғи қуйидагича бажарилади:

1. Дастлаб қиймат бериш буйруғидан ўнг томонда турган ифоданинг қиймати ҳисобланади.

2. Сўнгра, бу қиймат қиймат бериш буйруғининг чап томонида турган ўзгарувчига қиймат қилиб берилади. Масалан:

- *i:=0;* — *i* ўзгарувчининг қиймати нолга тенг бўлади;
- *a:=b+c;* — *b* ва *c* ўзгарувчиларининг йиғиндиси *a* га қиймат қилиб берилади;
- *j :=j+1;* — *j* нинг қиймати бирга ортади.

Агар ифоданинг типи қиймат олаётган ўзгарувчининг типига мос бўлса, буйруқ тўғри ёзилган бўлади. *Real* типидаги ўзгарувчи *real* ёки *integer* типидаги ифоданинг қийматини олиши мумкин. *Integer* типидаги ўзгарувчи фақат *integer* типидаги ифода қийматини қабул қила олади. Агар *i* ва *n* ўзгарувчилари i*nteger*, d – эса *real* бўлса, у ҳолда

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

буйруқлари нотўғри,

d:=i+1;

буйруғи эса тўғри ҳисобланади.

Компиляция жараёнида ифоданинг ва қиймат олаётган ўзгарувчининг типлари ўртасидаги мослик текширилади. Агар мослик бўлмаса,

Incompatible types ... and ...

кўринишидаги ахборот экранга чиқарилади. Бу ахборотда кўп нуқта белгиси ўрнига ифода ва ўзгарувчининг типлари кўрсатилади. Масалан :

Incompatible types 'Integer' and.'Extended'.

менюга

2.8. Стандарт функциялар

Дастурчилар ихтиёрига Delphi тили бир қатор стандарт функцияларни таклиф қилади.

Функциянинг типи унинг номи билан боғланган. Шунинг учун операнда сифатида бу функциялардан фойдаланиш мумкин. Функция қийматининг типи ва аргументларининг типлари билан характерланади. Қиймат олаётган ўзгарувчининг типи функция типига мос бўлиши лозим.

Математик функциялардан (1.6.жадвал) турли ҳисоблаш ишларини бажаришда эҳтиёжга қараб фойдаланиш мумкин.

Математик функциялар 1.6. жадвал

функция	**қиймат**
Abs (n)	n нинг абсолют қиймати
Sqrt (n)	n нинг квадрат илдизи
Sqr (n)	n нинг квадрати
Sin (n)	Синус n
Cos (n)	Косинус n
Arctan (n)	Арктангенс n
Exp(n)	Экспонента n
Ln(n)	n нинг натурал логарифми
Rardom(n)	0 дан n-1 гача бўлган тасодифий сон

Тригонометрик функцияларда бурчакларни радианларда ифодаланиши лозим. Бурчакни α-градусдан радианга алмаштириш учун (α*π)/180 формуласидан фойдаланиш мумкин.

Алмаштириш функциялари (1.7. жадвал) маълумотларни киритиш ва чиқаришда жуда кўп фойдаланилади. Масалан, экранга чиқариш майдонига (компонент Label) real типидаги маълумотни чиқариш учун дастлаб бу маълумотни матнли типга маълумотга алмаштириш керак бўлади. Бу ишни

FloatToStr функцияси ёрдамида бажариш мумкин. Масалан,

Label1.caption := FioatTostr(x)

буйруғи Label1 майдонига *x* – нинг қийматини чиқаради.

Алмаштириш функциялари 1.7.жадвал

функция	**функциянинг қиймати**
Chr(n)	Коди *n* га тенг бўлган белги
IntToStr (k)	Бутун *k* сонини ифодаловчи матн
FloatToStr (n)	Ҳақиқий *n* сонини ифодаловчи матн
FloatToStrF(n,f,k,m)	Ҳақиқий *n* сонини ифодаловчи матн. Бу ерда *f* - формат (ифодалаш усули); *k* – аниқлик (рақамларнинг умумий сони); *m* - вергулдан кейинги рақамлар сони.
StrToInt (s)	*s* матни ифодалаётган бутун сон
StrToFloat (s)	*s* матни ифодалаётган ҳақиқий сон
Round (n)	*n* сонини яхлитлаш
Trunc (n)	Ҳақиқий *n* сонини каср қисмини ташлаб юбориб, ҳосил қилинган бутун сон
Frac(n)	Ҳақиқий *n* сонининг каср қисми
Int (n)	Ҳақиқий *n* сонининг бутун қисми

Функциялардан фойдаланиш. Функциялардан операндалар сифатида ҳам фойдаланиш мумкин. Функциянинг параметри константа, ўзгарувчи ёки аргумент типига мос типдаги ифода бўлиши мумкин. Қуйида функциялардан фойдаланишга мисоллар келтирамиз:

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

менюга

2.9. Маълумотларни киритиш

Дастурда бошланғич маълумотларни киритиш ойнаси ёки таҳрирлаш майдони (компонент Edit) орқали ташкил қилиниши мумкин.

Киритиш ойнаси орқали маълумотларни киритиш - бу стандарт диалог ойнаси бўлиб, ***inputBox*** функциясига мурожаат қилиш натижасида юзага келади. *InputBox* функциясининг қиймати – фойдаланувчи киритган матндир.

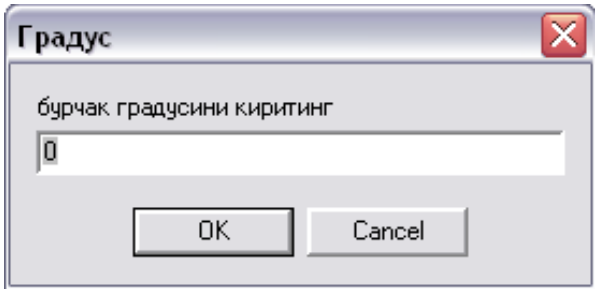
InputBox функцияси умумий кўринишда қуйидагича ёзилади:

ўзгарувчи := InputBox(сарлавҳа, эслатма, қиймат);

бу ерда ***ўзгарувчи*** – сатрли типдаги ўзгарувчи бўлиб, у қийматни фойдаланувчидан олади, ***сарлавҳа*** - киритиш ойнасидаги матн, ***эслатма*** - изоҳловчи матн**,** ***қиймат*** - киритиш ойнаси экранга чиққанда, шу ойнада кўринадиган матн.

Қуйидаги мисолда градусни радианга ўтказиш учун бошланғич маълумотни киритиш ойнаси келтирилган. Бу ойнага мос буйруқ қуйидагича ёзилади:

```
s:=InputBox('Градус','бурчак градусини киритинг','0');
```



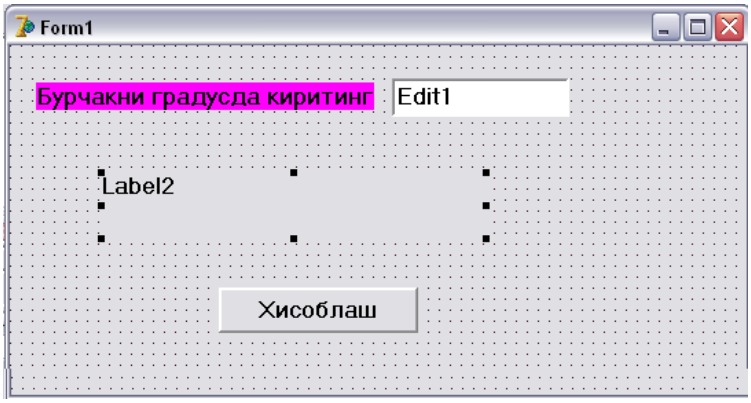
2.5-расм. Киритиш ойнасига мисол

Агар дастур иши давомида фойдаланувчи матнни киритиб, **ОК** тугмасини чертса, *inputBox* функциясининг қиймати киритилган матндан иборат бўлиб қолади. Агар **Cancel** тугмаси чертилса, функциянинг қиймати қилиб қийматнинг кўрсатилган параметри қабул қилинади.

Агар зарурат бўлса, *inputBox* фукциясининг матнли (string) қиймати алмаштириш функциялари ёрдамида сонли типга ўтказилиши керак бўлади. Масалан:

s:=InputBox('Градус','бурчак градусини киритинг','0');
gradus := StrToFloat(s);

Таҳрирлаш ойнасидан киритиш - бу ***Edit*** компонентасидир. Таҳрирлаш ойнасидан киритиш ***Edit*** компонентасининг ***Text*** хусусиятига мурожаат қилиш орқали амалга оширилади..



1.6-расм. Edit1 компонентаси маълумот киритишда фойдаланилмоқда

1.6-расмда градусни радианга ўтказишнинг диалог ойнаси келтирилган. Унда *Edit1* компонентаси маълумотларни киритиш учун қўлланмоқда. Уни сонли маълумотга айлантириш учун кўрсатма қуйидагича ёзилади:

gradus := StrToFloat(Editl.Text);

2.10. Маълумотларни чиқариш

Энг содда дастур ўз иши натижасини маълумотлар ойнасига ёки диалог ойнасининг чиқариш майдонига (компонент Label) маълумотларни чиқариши мумкин.

Маълумотлар ойнасига чиқариш фойдаланувчиларнинг эътиборини жалб қилиш учун фойдаланилади. Бу ойна ёрдамида дастур бошланғич маълумотларнинг ҳатолиги ҳақида ахборот бериш ёки орқага қайтариб бўлмас (масалан, файлларни ўчириш) амалларни бажаришга рухсат (одатда тасдиқ кўринишида) сўраш учун фойдаланилиши мумкин.

Маълумотлар ойнасига ахборотни ***ShowMessage*** процедураси ёки ***MessageDlg*** функцияси ёрдамида чиқариш мумкин.

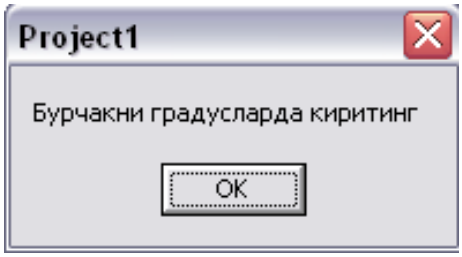
ShowMessage процедураси экранга матнли ойна ҳамда **ОК** тугмасини чиқаради. Умумий ҳолда бу *ShowMessage* процедурси қуйидагича ёзилади:

ShowMessage(маълумот);

Бу ерда ***маълумот*** — ойнага чиқарилиши талаб қилинган матн.

2.7-расмда қуйидаги буйруқнинг натижаси келтирилган:

Showmessage('Бурчакни градусларжда киритинг');



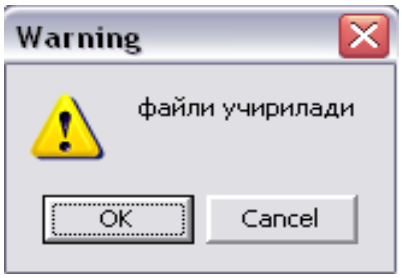
2.7-расм. Маълумотлар ойнасига намуна

Шунга эътибор бериш керакки, *ShowMessage* процедураси ойнасининг сарлавҳасидаги маълумот **Project Options** ойнасининг **Application** қистирмасидан олинади. Агар илованинг номи кўрсатилмаган бўлса, у ҳолда сарлавҳада бажариладиган файлнинг номи чиқарилади.

MessageDig функцияси унга қараганда универсалроқ. У ойнага стандарт нишонли маълумотлардан бирини, масалан, "Warning (диққат)" сўзи, буйруқли тугмаларнинг миқдори ва типи, фойдаланувчи қайси тугмалардан бирини чертиши кераклигини чиқариши мумкин. 2.8-расмда қуйидаги буйруқнинг натижаси ифодаланган:

r:=MessageDlg(FName+'файли учирилади', mtWarning,[mbOk,mbCancel] , 0) ;

MessageDlg функциясининг қиймати — сон. Бу қийматни текшириб, диалог қайси тугманинг чертилиши ёрдамида тугатилганлигини аниқлаш мумкин. *MessageDlg* ойнаси умумий кўринишда қуйидагича ташкил қилинади:



2.8-расм. Маълумотлар ойнасига мисол

танлаш:=MessageDlg(маълумот, Тип, тугмалар, эслатма);

Бу ерда ***маълумот*** – ахборот матни; ***Тип*** — ахборотнинг типи . Ахборот маълумотнома, огоҳлантириш ёки критик ҳатолик ҳақидаги ахборот кўринишларидан бирида бўлиши мумкин. Ҳар бир типдаги маълумотга махсус нишон мос келади. Ахборот типи номланган константа ёрдамида кўрсатилади (2.8-жадвал); ***Тугмалар***- маълумот ойнасига чиқариладиган тугмалар рўйхатини ўз ичига олади. Бу рўйхат бир-биридан нуқтали вергул билан ажратилган бир нечта номланган константалардан иборат бўлиши мумкин (2.9-жадвал). Рўйхат квадрат қавслар ичига олинади.

MessageDlg функциясининг константалари 2.8-жадвал.

Константа	Ахборот типи	Нишон
mtWarning	Диққат	
mtError	Ҳато	
mtInformation	Ахборот	
mtConfirmation	Тасдиқ	
mtCustom	Оддий	Нишони йўқ

MessageDlg функциясининг константалари 2.9-жадвал

Константа	Тугма	Константа	Тугма
mbYes	Yes	mb Abort	Abort

mbNo	No	mbRetry	Retry
mbOK	OK	mbIgnore	Ignore
mbCancel	Cancel	mbAll	All
mbHelp	Help		

Масалан, маълумотлар ойнасида **ОК ва Cancel** тугмаларининг пайдо бўлиши учун ***Тугмалар рўйхати*** қуйидагича бўлиши керак:

[mbOK,mbCancel]

Юқоридаги константалардан ташқари, *mbokcancel, mbYesNoCancel* ва *mbAbortRetryIgnore* константаларидан фойдаланиш мумкин. Бу константалар диалог ойналарида энг кўп қўлланиладиган буйруқли тугмаларнинг комбинацияларини аниқлайди. ***Эслатма*** - агар фойдаланувчи F1 тугмасини чертса, экранга чиқариладиган ёрдамчи маълумотномалар системасининг бўлимини белгилайдиган параметр, агар эслатмаларнинг экранга чиқарилиши кўзда тутилмаган бўлса, у ҳолда ***Эслатма*** параметрининг қиймати нолга тенг бўлади.

MessageDlg функциясининг (2.10-жадвал) қиймати фойдаланувчи томонидан қайси тугма чертилганлигини аниқлашга имкон беради.

MessageDlg функциясининг қийматлари 2.10-жадвал

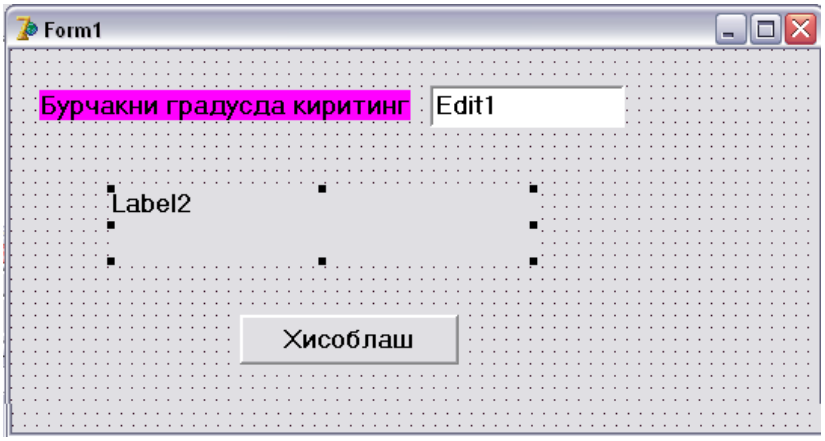
MessageDig функцияси қиймати	Диалог тугмани чертилганда тугайди
mrAbort	Abort
mrYes	Yes
mrOk	Ok
mrRetry	Retry
mrNo	No
mrCancel	Cancel
mrIgnore	Ignore
mrAll	All

Диалог ойна майдонига чиқариш. Диалог ойнасининг маълумотларни чиқариш учун мўлжалланган бир қисми чиқариш майдони ёки тамға майдони деб аталади. ***Label*** компонентаси - чиқариш майдонидир.

Чиқариш майдонидаги маълумот ***Caption*** хусусияти билан аниқланади. ***Caption*** нинг хусусиятини илова формасини ишлаб чиқиш жараёнида ҳамда дастурнинг ишлаши жараёнида ўзгартириш мумкин. Дастур ёрдамида чиқариш майдонига маълумотларни узатишни ташкил қилиш учун ***Caption*** хусусиятига янги қиймат бериш керак.

2.9-расмда бурчакни градусдан радианга ўтказишнинг диалог ойнаси берилган. Бу ойнада иккита *Label*, битта *Edit* ва *Button* компонентлари мавжуд. *Label1* компонентаси эслатма тарзидаги маълумотни берса, *Label2* компонент — дастур натижасини экранга чиқаради.

Caption хусусияти белгили тип ҳисобланади. Шунинг учун дастурнинг ишлаши жараёнида майдонга сонли маълумотларни чиқаришга эҳтиёж пайдо бўлса, бу сонларни *FloatToStr* ёки *IntToStr* функциялари ёрдамида сатрли типга ўтказиш керак. Масалан: *Label2.Caption:=FloatToStr(kg)+'кг';*.



2.9-расм. *Label2* майдони дастур натижаси учун мўлжаланган.

менюга

2.11. Процедура ва функциялар

Delphi тилида дастурлашда дастурчининг асосий вазифаси ходисаларни қайта ишлаш процедураларини (қисм дастурларни) ишлаб чиқишдан иборат бўлади.

Ходиса рўй берганда, дастурчининг шу ходиса учун мўлжаллаган процедураси автоматик тарзда ишга тушиши керак бўлади. Ходисалар рўй берганда, уларга мос келган ходисаларнинг қайта ишлаш процедураларини ишга туширишни *Delphi* ўз зиммасига олади.

Object Pascal тилида дастурнинг асосий бирлиги қисм дастур ҳисобланади. Қисм дастурлар икки турга бўлинади: процедура ва функция. Процедура ҳам, функция ҳам бирор вазифани бажариш учун кўрсатма - буйруқлар кетма-кетлигидан иборат бўлади. Қисм дастурдаги буйруқларни бажариш учун бу қисм дастурга мурожаат қилиш (чақириш) лозим. Функциянинг процедурадан фарқи шундаки, функция номи билан қиймат боғланган бўлади ва бу функциянинг номидан ифодаларда алоҳида операнда сифатида фойдаланиш мумкин.

Процедура структураси. Процедура сарлавҳадан бошланади. Сўнгра константаларни эълон қилиш бўлими, типларни эълон қилиш бўлими, ўзгарувчиларни эълон қилиш бўлими, кўрсатма-буйруқлар бўлими келади. Процедуралар умумий кўринишда қуйидагича ёзилади:

procedure ном (Параметрлар рўйхати);
const
// бу ерда номланган константалар рўйхати берилади;
type
// бу ерда типлар ***var*** ёрдамида эълон қилинади;
// бу ерда ўзгарувчилар ва уларнинг типлари эълон қилинади;
begin
// бу ерда дастурнинг буйруқлари ёзилади;
end;

Процедура сарлавҳаси ***procedure*** дан бошланади. Ундан кейин процедуранинг номи кўрсатилади. Бу ном шу процедурага мурожаат қилиш (ишга тушириш ёки чақириш) учун хизмат қилади. Агар процедурада параметрлар қатнашса, улар процедура номидан кейин қавслар ичида кўрсатилади. Сарлавҳа "нуқтали вергул" белгиси билан тугайди.

Агар дастурда ностандарт, яъни янги типларни яратишга эҳтиёж пайдо бўлса, бу типлар ***type*** сўзидан кейин эълон қилинади.

Ўзгарувчиларни эълон қилиш бўлимида шу процедура учун хос бўлган барча ўзгарувчилар ва уларнинг типлари рўйхати келтирилади. Бу рўйхат ***var*** сўзидан кейин бошланади.

Буйруқлар бўлими ***begin*** сўзи билан бошланади ва ***end*** сўзи билан тугайди. Бу ерда процедуранинг буйруқлари кетма-кетлиги жойлашади. ***End*** сўзидан кейин "нуқтали вергул" белгиси қўйилади.

Қуйидаги процедурада умумий харид суммасини топиш масаласи ҳал қилинган. Агар 5000 сўмдан ортиқ суммага харид қилинса, умумий суммадан 10% чегириб ташланади.

```
procedure Summa;
var
   baho: real;// нархи
   miqdori: integer;// харид қилинган бир ҳил буюмлар сони
   s: real;// сумма
   mes: string[255];// хабарнома

begin
   baho := StrToFloat(Form1.Edit1.Text);
  miqdori := StrToInt(Form1.Edit2.Text);
  s := baho * miqdori;
  if s > 500 then begin
    s := s * 0.9;
```

```
    mes := '10% ли чегирма айриб ташланди.'+#13;
  end;
  mes := mes+ 'Харид нархи : '+ FloatToStrF(s,ffFixed,4,2) +' сум';
  Forml.Label3.Caption := mes;
end;
```

Функция структураси функциянинг сарлавҳаси, константалар, типлар, ўзгарувчиларни эълон қилиш бўлимлари ҳамда буйруқлар бўлимидан иборат бўлади. Функция умумий ҳолда қуйидагича кўринишда ташкил қилинади:

```
function ном (Параметрлар рўйхати) : Тип;
const// константалар рўйхати бўлими
type//типларни эълон қилиш бўлими
var//ўзгарувчиларни эълон қилиш бўлими
begin//буйруқлар бўлими
Result := қиймат;//функция номини қиймат билан боғлаш
end;
```

Функциянинг сардавҳаси ***function*** сўзи билан бошланади, ундан кейин функциянинг номи келади. Сўнгра қавслар ичида функциянинг параметр-аргументлари ва уларнинг типларининг рўйхати ёзилади. Қавсдан кейин икки нуқта (:) қўйиб, функциянинг қабул қиладиган қийматининг типи кўрсатилади. Сарлавҳа "нуқтали вергул" билан тугайди.

Буйруқлар бўлимида ўзгарувчиларни эълон қилиш бўлимда кўрсатилган ўзгарувчилардан ташқари, r***esult*** ўзгарувчисидан ҳам фойдаланиш мумкин. Функциядаги буйруқлар бажариб бўлинганидан сўнг, бу ўзгарувчининг қиймати функциянинг қийматига айланади. Шунинг учун, функциянинг буйруқлари орасида албатта ***result*** ўзгарувчисига қиймат берувчи буйруқнинг бўлиши шарт. Одатда, бу буйруқ функциянинг энг охирги бажариладиган буйруғи бўлади.

Қуйидаги мисолда градусларни радианга айлантириш функцияси келтирилган:

```
function GradToRad(grad:integer):real;
  begin
     result:=(grad*pi)/180;
  end;
```

менюга

2.12. Дастурда буйруқларни ёзиш.

Ҳар бир буйруқ бошқасидан нуқтали вергул билан ажратилади. Бошқача айтганда, ҳар бир буйруқдан кейин нуқтали вергул белгиси қўйилади.

Дастурнинг ҳар бир сатрида бир ёки бир нечта буйруқларни кўрсатиш мумкин.

Айрим буйруқларни (***if, case, repeat, while*** ва х.к.) бир нечта сатрга ёзиш қабул қилинган. Уларнинг структурасини бошқаларидан ажратиш мақсадида сатрнинг чап чегарасидан буйруқларни бир оз чекинтириб ёзиш тавсия этилади. Бу дастур матнини ўқиш ва тушунишни осонлаштиради. Масалан:

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

Then ва ***else*** бир-бирларининг остига ҳамда ***if*** га нисбатан бир ҳил масофада чекинтириб ёзилганига эътибор беринг. ***End*** сўзи ***begin*** остига ёзилган. ***begin*** ва ***end*** лар орасида буйруқлар ***begin*** га нисбатан бир-бирининг остида, бир масофада чекинтириб жойлаштирилган. Юқоридаги буйруқларни қуйидагича ҳам ёзиш мумкин:

```
if d >= 0 then begin x1:=(-b+Sqrt(d))/(2*a); x2:=(-b-Sqrt(d))/(2*a);
```

ShowMessage('x1=' + FloatToStr(xl) + 'x2=' + FloatToStr(x2)) ; **end**
else ShowMessage('Тенглама ҳақиқий ечимларга эга эмас.')

Аммо, биринчи вариант қулайроқ, чунки, унда алгоритм структураси яхшироқ кўринади.

Айрим узун ифодаларни бир нечта сатрга бўлиб ёзиш мумкин. Бундай ифодаларни ихтиёрий белгисидан бошлаб узиш ва қолган қисмини кейинги сатрга ўтказиш мумкин. Ўзгарувчиларнинг номларини, сонли ва матнли константаларни, шунингдек таркибий операторларни узиш, масалан, қиймат бериш операторини мумкин эмас. Қуйида бир нечта сатрга ёзилган буйруққа мисол келтирамиз:

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

Шуни ҳисобга олиш керакки, компилятор ортиқча "бўш жой" белгиси ҳамда бўш сатрларга эътибор бермайди. Шунинг учун у сатр бошидаги барча "бўш жой" белгиларини "кўрмайди". Бу эса сатрлардаги буйруқларини чекинтириб ёзишга имкон беради. Арифметик, мантиқий ифодаларни (шартларни), параметрлар рўхатини ёзишда "бўш жой" белгиларини қўйиш талаб қилинмайди, аммо улардан фойдаланиш дастур матнини дастурчи учун ўқишни осонлаштиради. Қуйидаги икки ифодани солиштиринг:

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

Иккинчи вариантнинг осон ўқилиши кўриниб турибди.

Дастур ишини тушунишни енгиллаштириш учун дастур матнига изоҳларни киритиш мумкин. Умумий ҳолда изоҳларни фигурали қавслар орасида кўрсатилади. Очилаётган қавс изоҳнинг бошланганлигини, ёпилаётгани эса тугаганлигини билдиради. Агар изоҳ бир сатрли ёки бирор буйруқдан кейин келса, ундан аввал // (қўш чизиқча) белгилари қўйилади. Масалан:

```
var
{ квадрат тенгламанинг коэффициентлари }
a:real;//биринчи даражали номаълум олдидаги коэффициент
b:real;//иккинчи даражали номаълум олдидаги коэффициент
c:real;//нолинчи даражали номаълум олдидаги коэффициент
{ тенгламанинг илдизлари } x1,x2:real;
```

менюга

2.13. Дастурлаш усули

Дастур устида ишлар экан, дастурчи ўзи ёзаётган дастур аввало дастурчи учун, қолаверса бошқалар учун ҳам мўлжалланганлигини яхши тасаввур қилиш лозим. Дастур матни биринчи ўринда, дастурчи учун, сўнгра у билан бирга лойиха устида ишлаётган ҳамкасблари учун керак бўлади. Шунинг учун дастурчилар ишининг самарасини кучайтириш мақсадида дастур матнини осон ўқилиши, тушунарли бўлиши ҳамда дастурнинг структураси ечилаётган масала структураси ва алгоритмига мос бўлиши зарур деган критериялар қабул қилинган. Бунга қандай эришиш мумкин? Бунинг учун яхши дастурлаш усулига риоя қилиш лозим. дастурлаш усули – бу дастурчи ўз иши жараёнида (онгли равишда ёки "бошқалар шундай қилгани учун") риоя қиладиган қонун-қоидалар мажмуасидир. Табиийки, яхши дастурчи яхши дастурлаш усулларига риоя қилиши керак. Дастурлашнинг яхши усули қуйидагиларни назарда тутади:

- изоҳлардан фойдаланиш;
- ўзгарувчи, процедура, функцияларни маъносига қараб номлаш ;
- чекинишлардан фойдаланиш;
- бўш сатрлар ва "бўш жой" белгиларидан фойдаланиш .

Дастурлашнинг яхши усули дастур матнини компьютер хотирасига киритишда ҳатоликларни камайтиришга, дастурни ўқиш ҳатоликларни аниқлаш ва тузатмалар киритишни енгиллаштиришда ёрдам беради.

Дастурлаш усулининг яхшилигини баҳолаш учун аниқ бир критерия ўйлаб топилмаган. Аммо, дастур матнига бир марта қарашда дастурнинг яхши ёки ёмон ёзилганлигини кўриш мумкин. +олаверса, яхши дастур фақат матнининг ёзилиши усули билан белгиланмайди. Яхши дастур ишончли, фойдаланувчига нисбатан дўстона муносабат, бажарилиш тезлиги билан фарқланади.

Дастурнинг ишончлилиги деганда дастур фойдаланувчининг онгли эканлигига қарамасдан, бошланғич маълумотларни назорат қилиши, бажарилган амалларнинг натижаларини текшириши (мисол

учун, бирор бир сабаб билан бажарилмаслиги мумкин бўлган амаллар, маслан, файллар устидаги амаллар) назарда тутилади.

Дўстона муносабат деганда эса яхши режалаштирилган диалог ойнаси, ёрдамчи маълумотномалар системасининг мавжудлиги, дастурнинг фойдаланувчи нуқтаи назаридан онгли ва олдиндан айтиш мумкин бўлган ҳулқи тушунилади.

Эслатма: Ушбу электрон қўлланмада биз келтирган дастурларни яхши дастурлаш усулига мисол қилиб олиш мумкин.

3-БОБ. DELPHI ДА БОШҚАРИШ БУЙРУҚЛАРИ

3.1. Шарт ва мантиқий ифодалар

Компьютер одатда дастурдаги буйруқларни кўрсатилган тартибда, кетма-кет бажаради. Бундай дастурларни чизиқли дастурлар деб аталади. Амалда, чизиқли бўлган масалалар камдан-кам учрайди. Энг элементар масала ҳам чизиқли бўлмаслиги мумкин. Масалан, электр занжиридаги токни хисоблаш талаб қилинган бўлсин. Агар фойдаланувчи бошланғич маълумотларни тўғри киритса, ҳақиқатдан ҳам, масала чизиқлига айланади. Аммо, фойдаланувчи бошланғич маълумотларни тўғри киритади деб ким кафолат бера олади? Хисоблаш формуласи $I = U / R$ га кўра қаршилик нолга тенг бўлиши мумкин эмас. Аммо, фойдаланувчи қаршилик учун нол қийматни берса нима бўлади? Дастурни иложи борича бундай ҳатоликлардан химоя қилишни ташкил этиш дастурчининг асосий вазифаларидан бири ҳисобланади. Шунинг учун, фақат тўғри киритилган бошланғич маълумотлар учун хисоблашни амалга ошириш

мақсадида масаланинг ечиш алгоритмига ўзгартириш киритиш (3.1-расм) талаб қилинади.

Дастурнинг кейингши юришлари танланадиган алгоритм нуқталари танлаш нуқталари дейилади. Масалан ечишнинг навбатдаги қадамини танлаш бирор шарт (мантиқий ифода) нинг қийматига боғлиқ равишда амалга оширилади.

Шарт. Кундалик ҳаётда шартлар жавоби Ҳа ёки ЙЎҚ тарзида бўлган савол тариқасида қўйилади. Масалан:

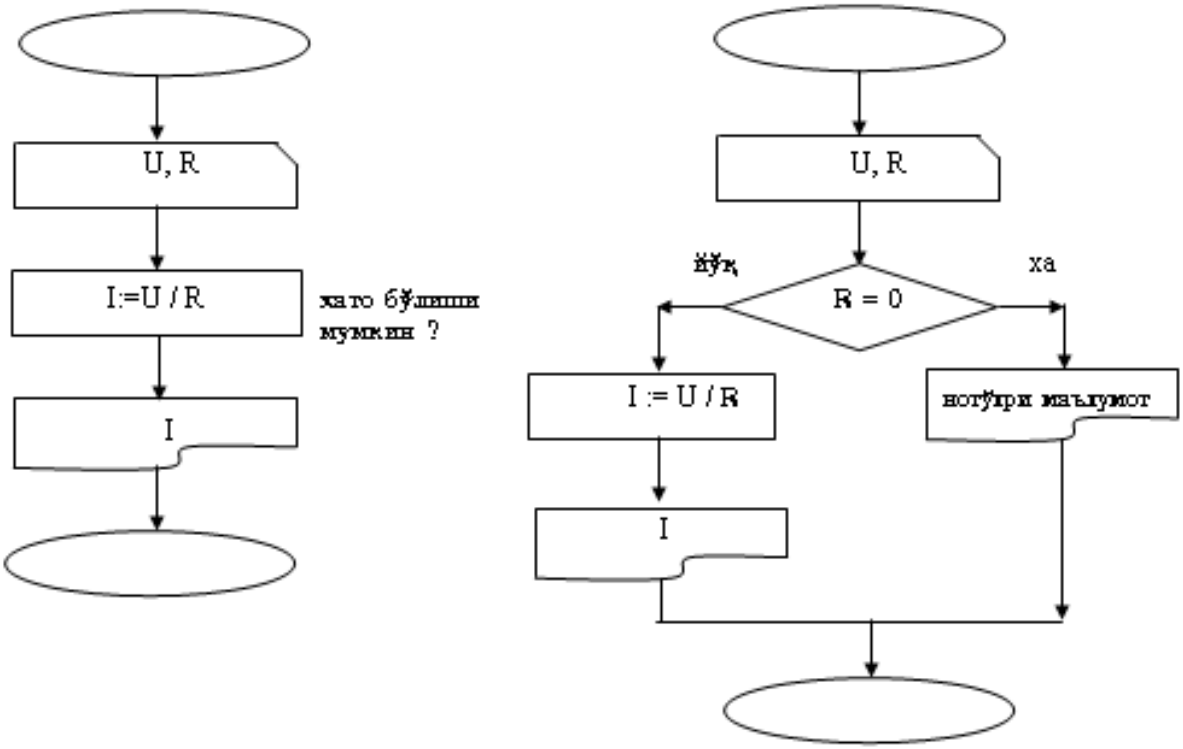
- Қаршилик нолга тенгми?
- Жавоб тўғрими?
- Харид нархи 5000 сўмдан кўпми?

Дастурда шартлар – мантиқий типдаги (Boolean) ифодадан иборат. Бу ифода True (рост) ёки False (ёлғон) қийматларидан бирини қабул қилади.

Энг содда шарт икки операнда ва муносабат белгисидан иборат бўлади. Умумий ҳолда шартлар қуйидагича ёзилади:

On1 Оператор On2

Бу ерда ***On1*** ва ***On2*** — шартнинг икки операндаси бўлиб, уларнинг ўрнида ўзгарувчилар, константалар, фнукциялар ва ифодалар келиши мумкин, ***Оператор*** — таққослаш оператори.



3.1-расм. Бир масала ечимининг иккита алгоритм варианти

Delphi тилида олтита таққослаш оператори мавжуд

Таққослаш операторлари 3.1-жадвал

Оператор	мазмуни	Таққослаш натижаси
>	Катта	Агар биринчи операнда катта бўлса -True, акс ҳолда – False
<	Кичик	Агар биринчи операнда кичик бўлса -True, акс ҳолда – False
=	Тенг	Агар биринчи ва иккинчи операндалар тенг бўлса -True, акс ҳолда – False
<>	Тенг эмас	Агар биринчи ва иккинчи операндалар тенг бўлмаса -True, акс ҳолда – False
>=	Катта ёки тенг	Агар биринчи операнда иккинчисидан катта ёки тенг бўлса -True, акс ҳолда – False

<=	Кичик ёки тенг	Агар биринчи операнда иккинчисидан кичик ёки тенг бўлса- True, акс ҳолда– False

Мисоллар келтирамиз:

Summa < 1000 Score >= HBound Sim = Chr(13)

Дастлабки мисолда шартнинг операндалари ўзгарувчи ва константа бўлиб, бу шартнинг қиймати **Summa** ўзгарувчисининг қийматига боғлиқ. Агар **Summa** нинг қиймати 1000 дан кичик бўлса, бу шартнинг қиймати True, акс ҳолда, яъни **Summa** 1000 дан катта ёки тенг бўлса – False.

Иккинчи мисолда операндалар сифатида ўзгарувчилар қатнашмоқда. Бу шарт агар Score ўзгарувчисининг қиймати **HBound** ўзгарувчисининг қийматидан катта ёки тенг бўлгандагина True қийматини олади.

Учинчи мисолда иккинчи опреранда ўрнида функция келган. Агар **Sim** ўзгарувчисининг қиймати <Enter> клавишасининг коди бўлган 13 га тенг бўлса, бу шарт True бўлади.

Шартларни ёзишда унинг операндаларининг бир ҳил типда бўлишига алоҳида эътибор бериш лозим. Агар операндаларнинг бири бошқа типга мансуб бўлса, уларни бир ҳил типга келтириш зарур. Масалан, Key ўзгарувчиси integer деб эълон қилинган бўлса,

Key = Chr(13)

кўринишдаги шарт синтактик жиҳатдан нотўғри, чунки, Chr функциясининг қиймати char (белгили) типда бўлади.

Дастур матнини трансляция қилишда нотўғри ёзилган шартлар учраб қолса, бу ҳақда

incompatible types (ўзаро мос бўлмаган типлар)

тарзидаги ахборот экранга чиқарилади.

Содда шартлардан фойдаланиб, **And** - "мантиқий ВА", **Or**- "мантиқий ЁКИ" ҳамда **Not** - "ИНКОР" мантиқий амаллари ёрдамида мураккаб шартларни ҳосил қилиш мумкин. Мураккаб шартлар умумий ҳолда қуйидагича ёзилади :

Шарт1 оператор Шарт2

Бу ерда **Шарт1** ва **Шарт2** – содда шартлар, **оператор** —эса **and** ёки **or** лардан бири бўлиши мумкин. Масалан:

(ch >= '0') **and** (ch <= '9')
(day = 7) or (day + 6)
(Forml.Editl.Text <> ' ') or (Forml.Edit2.Text <> ")
Forml.CheckBoxl.Checked **and** (Forml.Editl.Text <> ")

And, **Or** ва **Not** мантиқий амалларининг натижаси 3.2-жадвалда келтирилган.

Мантиқий амалларнинг қийматлари **3.2-жадвал**

A	B	A and B	A or B	not A
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False

Мураккаб шартларни ёзишда мантиқий операторларнинг савияси таққослаш операторларига қараганда юқори эканлигини назарда тутиш лозим. Шунинг учун содда шартларни қавсларда ёзиш талаб қилинади.

Масалан, харид нархидан чегирма қуйидагича бўлсин: "Харид нархи 100 сўмдан кўп ва харид куни – якшанба бўлса 10% ли чегирма берилади". Агар хафта куни бутун типдаги Day ўзгарувчиси бўлиб, унинг 7 га тенг бўлиши якшанбани англатса, чегирма шартини қуйидагича ёзиш мумкин:

(Summa > 100) **and** (Day = 7)

Агар чегирмалар якшанба бўлмаган кундаги 500 сўмдан ортиқ харид учун ҳам берилса, шартни қуйидагича ёзиш мумкин:

((Summa > 100) **and** (Day =7)) **or** (Summa > 500)

менюга

3.2. Тармоқланиш (*if*) буйруғи

Алгоритмнинг тармоқланиш нуқтасидаги навбатдаги қадамни танлаш ***if*** ва ***case*** буйруғларидан бири ёрдамида амалга оширилиши мумкин. ***If*** (тармоқланиш) буйруғи икки имкониятдан бирини, ***case*** эса бир нечта имкониятдан бирини танлайди.

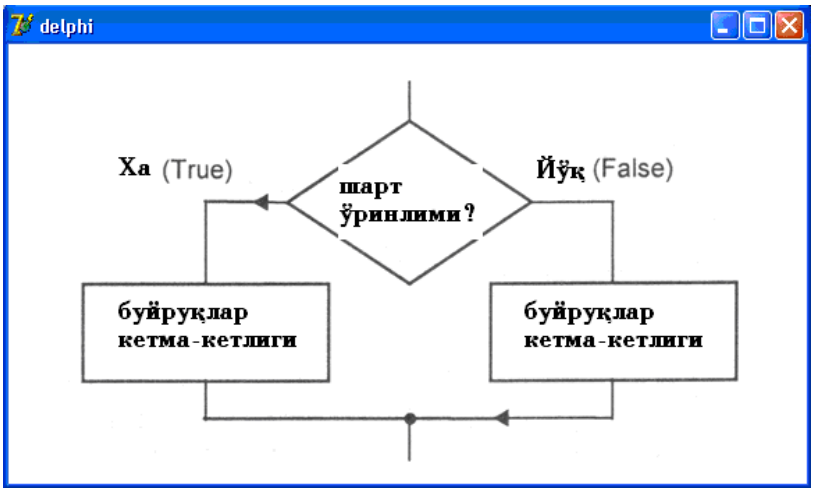
If* буйруғи** навбатдаги мумкин бўлган икки қадамдан бирини танлашга имкон беради, яъни тармоқланади. Танлаш кўрсатилган шартнинг бажарилишига боғлиқ равишда амалга оширилади. Умумий кўринишда ***If буйруғи қуйидагича ёзилади:

if шарт **then**
begin
// бу ерда агар шарт True бўлса бажариладиган буйруқлар ёзилади.
end
else
begin
// бу ерда агар шарт False бўлса бажариладиган буйруқлар ёзилади.
end;

Else дан олдин (**End** дан кейин) нуқтали вергул қўйилмайди.

If буйруғи қуйидагича бажарилади:
1. Шартнинг қиймати ҳисобланади. (Унинг қиймати True ёки False).
2. Агар шарт рост (шартнинг қиймати - True) бўлса, у ҳолда ***then*** сўзидан кейинги (***begin*** ва ***end*** орасидаги) буйруқлар бажарилади. ***If*** буйруғини бажариш шу билан тугайди, яъни ***else*** дан кейинги буйруқлар бажарилмайди. Агар шарт ёлғон (шартнинг қиймати - False) бўлса, у ҳолда ***else*** сўзидан кейинги (***begin*** ва ***end*** орасидаги) буйруқлар бажарилади.

3.2-расмда ***if-tnen-else*** га мос алгоритм келтирилган.



3.2-расм. if-then-else буйруқларининг алгоритми

Масалан, агар ***t*** ўзгарувчи электр занжиридаги қаршиликлар уланиши типини билдирса, (***t=1*** кетма-кет уланиш, ***t=2*** - параллел), ***r1*** ва ***r2*** - қаршиликлар миқдори бўлса, қуйидаги ***if*** буйруғи хисоблаш формуласини танлашга имкон беради.

```
if t=1 then
  begin
        z :=r1 +r2;
  end
    else
  begin
        z :=(r1 +r2)/(r1*r2);
  end;
```

Агар ***if*** буйруғида ***begin*** ва ***end*** орасида фақат битта буйруқ бўлса, у ҳолда ***begin*** ва ***end*** ларни

ёзмаслик ҳам мумкин. Масалан, юқоридаги буйруқларни

```
if t =l then  z :=r1 +r2
     else   z :=(r1 +r2(r1*r2);
```

кўринишида ёзиш мумкин. Буйруқнинг тугаганлигини билдирувчи нуқтали вергул белгисини қаерга қўйилганлигига эътибор беринг.

Агар бирор амал фақат шарт рост (True) бўлган ҳолдагина бажарилиб, бошқа ҳолларда бажарилиши талаб қилинмаса, бундай буйруғларни умумий кўринишда қуйидагича ёзиш мумкин:

```
if шарт then
begin
{ шарт фақат рост бўлгандагина бажариладиган буйруқлар}
end ;
```

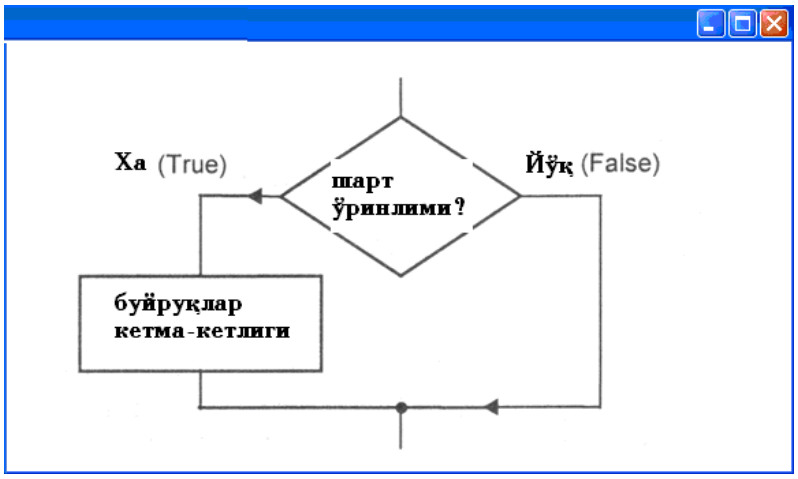
Масалан

```
if n =m
then c :=c +l;
```

буйруғи фақат **n=m** бўлгандагина **c** нинг қийматини бирга оширади.

If буйруғини амалда қўлланишини квадрат тенгламанинг ҳақиқий ечимларини топиш мисолида келтирамиз.

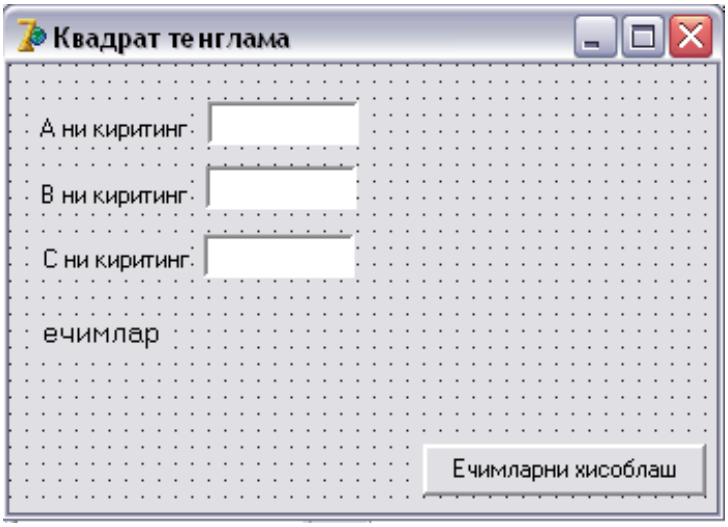
Маълумки, квадрат тенглама ***a, b, c*** коэффициентлари орқали берилади. Унинг ҳақиқий ечимларини топиш учун дастлаб $d = b^2 - 4ac$ формула билан дискриминантини топамиз. Агар



3.3-расм. ***if-then*** буйруғига мос алгоритм.

дискриминант нолдан катта ёки тенг бўлса, квадрат тенглама ҳақиқий ечимларга эга бўлади ва бу ечимлар $x_1 = \frac{-b+\sqrt{D}}{2a}$, $x_2 = \frac{-b-\sqrt{D}}{2a}$ формулалар билан аниқланади. Агар дискриминант нолдан кичик бўлса ечимлар мавжуд эмас.

Илова формасига коэффициентларни киритиш учун 3 та киритиш майдони, изоҳ ва ечимлар учун тўртта чиқариш майдонлари ҳамда битта буйруқли тугма қўйилади. Бу компоненталарнинг қийматлари 3.3-жадвалда берилади.



3.4-расм. Квадрат тенглама дастурининг диалог ойнаси

Эслатма: Бу ерда ва бундан буён формаларни тавсифлашда компоненталарнинг ўзгарадиган ҳусусият қийматлари келтирилади ҳалос. Компонеталарнинг қолган ҳусусиятлари ўзгармайди.

Квадрат тенглама дастурининг компонентлари 3.3-жадвал

компонента	Мазмуни
Edit1	***a*** коэффициент учун
Edit2	***b*** коэффициент учун
Edit3	***c*** коэффициент учун
Label1	A коэффициентни киритиш учун
Label2	B коэффициентни киритиш учун
Label3	C коэффициентни киритиш учун
Label4	Ечимни экранга чиқариш учун
Button1	Хисоблаш жараёнини бошлаш учун

Эслатма: Форма компоненталарининг қийматлари таърифланган жадвалларда компонента номи ва нуқтадан кейин ҳусусияти номи кўрсатилади. Масалан, жадвалдаги **Form1.Caption** сатри илова формасини яратишда форманинг **Caption** ҳусусиятига кўрсатилган матнни киритишни англатади.

Компоненталарнинг ҳусусиятлари жадвали 3.4-жадвал

Компонента	**Мазмуни**
Edit1.text	
Edit2.text	
Edit3.text	
Label1	A ни киритинг
Label2	B ни киритинг
Label3	C ни киритинг
Label4	Ечимлар
Button1	Ечимларни хисоблаш

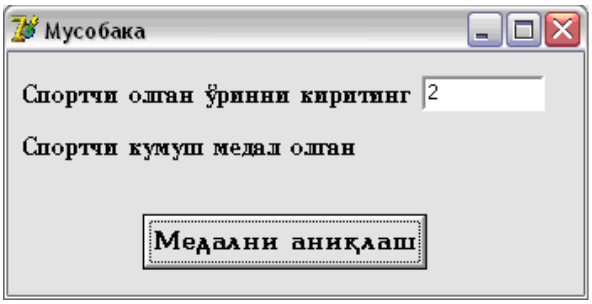
Дастур **Ечимларни хисоблаш** тугмаси босилиши билан хисоблашни бошлайди. Бунда ***onclick*** ходисаси рўй беради. Уни ***TForm1.Button1Click*** процедураси ёрдамида қайта ишланади.

3.1-листинг. Квадрат тенглама ечимларини хисоблаш

```
unit kw_teng;
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: TLabel;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private     { Private declarations }
  public      { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
var a1,b1,c1,d,x1,x2:real;
begin
   a1 := strtofloat(edit1.Text);
   b1 := strtofloat(edit2.Text);
   c1 := strtofloat(edit3.Text);
   d := b1*b1-4*a1*c1;
   if d >= 0 then
      begin
        x1 := (-b1 + sqrt(d))/(2*a1);
        x2 := (-b1 - sqrt(d))/(2*a1);
        label4.caption :='x1='+floattostr(x1)+#13+'x2='+floattostr(x2);
      end
    else
        label4.caption := 'Тенгламанинг ҳақиқий ечимлари йўқ';
end;
end.
```

Кўпинча дастурда иккита эмас, балки ундан кўп вариантлардан бирини танлашга тўғри келади. Масалан, спортчи мусобақада қатнашиб, А ўринни эгаллаган бўлса, у қандай медал олган? Бу масала учун мумкин бўлган 4 та вариантдан бири ўринли бўлишини ҳисобга олиб, хулоса чиқаришга тўғри келади. Ҳулоса эса А нинг қийматига боғлиқ.



3.5-расм. Мусобақа дастурининг диалог ойнаси

Мусобақа дастурининг алгоритми қуйидагича:

1. Бошлансин
2. Киритилсин А
3. Агар А=1 бўлса В := "олтин медал олган" ; 8 га ўтилсин
4. Агар А=2 бўлса В := "кумуш медал олган" ; 8 га ўтилсин
5. Агар А=3 бўлса В := "бронза медал олган" ; 8 га ўтилсин
6. В := "медал олмаган"
7. Чиқарилсин В
8. Ишни тугатилсин

3.2-листинг. Мусобақа дастурида спортчи олган медални аниқлаш.

```
unit Unit1;
```

```
interface
uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,   Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
   Edit1: TEdit;
   Label1: TLabel;
   Label2: TLabel;
   Button1: TButton;
   procedure Button1Click(Sender: TObject);
 private
   { Private declarations }
 public
   { Public declarations }
 end;
var
 Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
  var a:integer;
      b:string;
begin
  a := strtoint(edit1.Text);
  if a = 1
     then b := 'олтин медал олган'
     else if a = 2
            then b := 'кумуш медал олган'
            else if a = 3
                   then b := 'бронза медал олган'
                   else b := 'медал олмаган';
   label2.Caption := 'Спортчи '+b;
end;
end.
```

Дастурни ишга тушириш

Бу мисолда бир нечта ***if*** лар ичма-ич жойлашган. Бундай дастурларни ўқиш ва тушуниш ҳамда компьютер хотирасига киритиш дастурчи учун бир оз қийинроқ.

менюга

3.3. *Case* буйруғи

Аввалги мисолда спортчининг олган медалини аниқлаш учун мумкин бўлган тўртта вариантдан бирини танлаш ичма-ич жойлашган ***if*** буйруқлари орқали амалга оширилган эди. Аммо, кўп вариантлардан бирини танлашда масалага ***if*** ёрдамида ёндашиш ярамайди. Буни айниқса танлаш вариантларининг сони катта бўлганда янада яққол кўриш мумкин.

Delphi тилида вариантлардан бирини осонгина танлашга имкон берадиган ***Case*** буйруғи мавжуд. У умумий кўринишда қуйидагича ёзилади:

```
case Селектор of
     рўйхат1 :  begin  {1-буйруқлар кетма-кетлиги} end;
     рўйхат2 :  begin   {2-буйруқлар кетма-кетлиги} end;
     . . . . . .
     рўйхатN : begin  {N-буйруқлар кетма-кетлиги} end;
   else
```

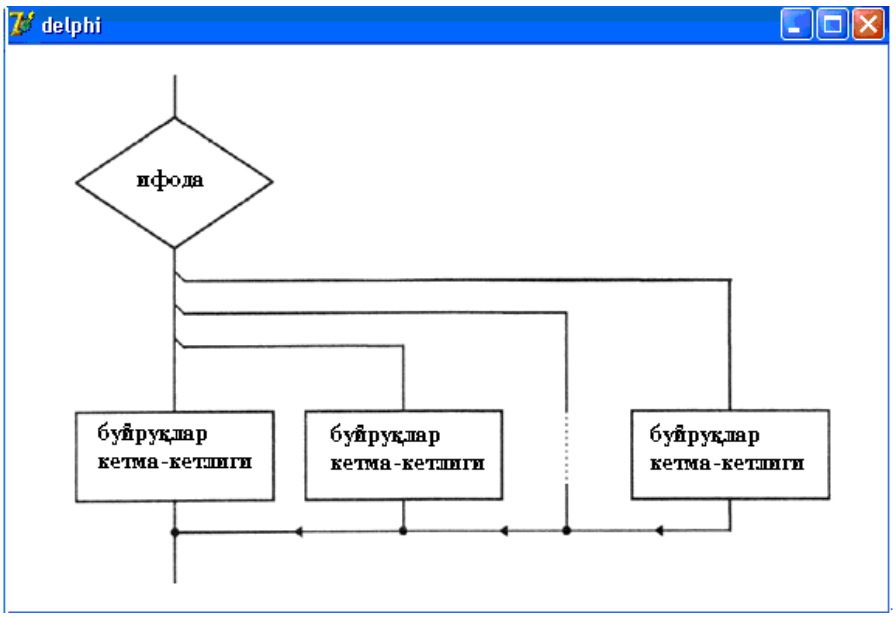
begin { N+1 -буйруқлар кетма-кетлиги) **end;**
end;

Бу ерда ***Селектор*** — қиймати дастурнинг навбатдаги қадамини (яъни, бажарилиши керак бўлган навбатдаги буйруқлар кетма-кетлигини) аниқлайдиган ифода; **рўйхатN** — константалар рўйхати. Агар константалар бирор диапазондаги сонлардан иборат бўлса, улардан биринчиси ва охиргисини икки нуқта билан ажратиб кўрсатиш мумкин. Масалан, 1, 2, 3, 4, 5, 6 константалри ўрнига 1..6 деб ёзса бўлади.

Case буйруғи қуйидагича бажарилади:

1. Дастлаб селектор-ифоданинг қиймати хисобланади.
2. Селектор-ифоданинг қиймати константалар рўйхатидаги константалар билан кетма-кет солиштирилади.
3. Агар селектор-ифоданинг қиймати рўйхатдаги бирор константа билан устма-уст тушса, шу константага мос буйруқлар кетма-кетлиги бажарилади. Шу билан ***case*** буйруғининг бажарилиши тугайди.
4. Агар селектор-ифоданинг қиймати константалар рўйхатидаги бирор константага тенг бўлмаса, у ҳолда ***else*** дан кейин турган буйруқлар кетма-кетлиги бажарилади.

Зарурат бўлмаса, ***else*** ни ёзмаслик ҳам мумкин. Бу ҳолда агар селектор-ифоданинг қиймати рўйхатда кўрсатилган контанталарнинг бирортасига тенг бўлмаса, у ҳолда ***case*** буйруғидан кейинги навбатда турган буйруқ бажарилади.



3.7-расм. *Case* буйруғининг алгоритми

Case буйруғига мисолар келтирамиз.

```
case n_day of
1,2,3,4,5: day:='Иш куни ' ;
6: day := 'Шанба';
7: day := 'Якшанба';
end;
```

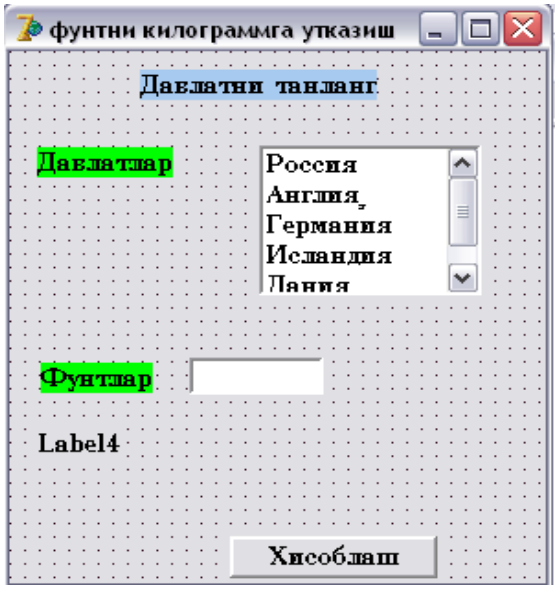
```
case n_day of
1..5: day := 'Иш кун';
6: day := 'Шанба';
7: day := 'Якшанба';
end;
```

```
case n_day of
6: day := 'Шанба';
7: day := 'Якшанба';
else day := 'Иш куни';
end;
```

Мисол тариқасида оғирлик ўлчов бирликларидан фунтни килограммга ўтказувчи дастурни кўрамиз.

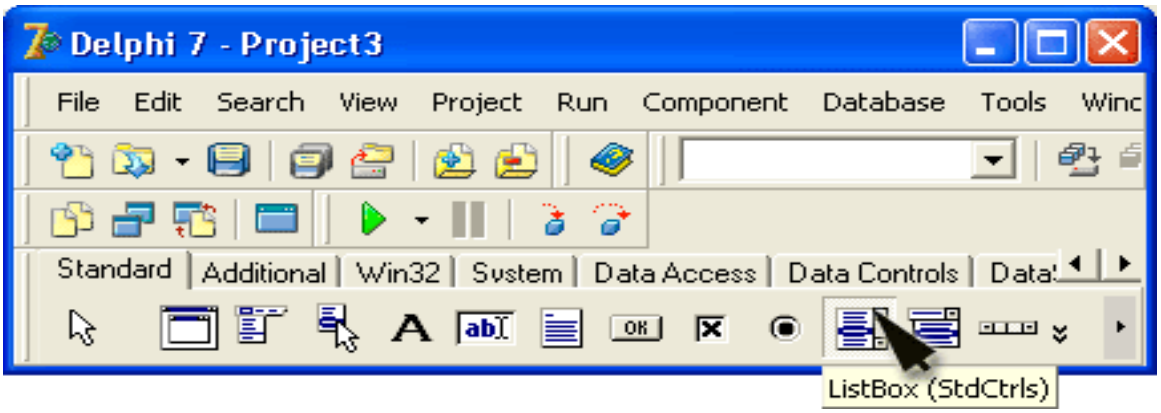
Дастур турли давлатларда фунтни турли белгилашларни ҳам хисобга олади. Масалан, Россияда бир фунт - 409,5 грамм, Англияда — 453,592 грамм, Германия, Дания ва Исландияда эса - 500 граммни ташкил қилади.

3.8-расмдаги диалог ойнасига давлатларни танлаш учун **Давлатлар** рўйхати киритилган.



3.8-расм. ***Case*** дан фойдаланувчи дастурнинг диалог ойнаси

Давлатни танлаш учун рўйхат - ***ListBox*** компонентасидан фойдаланилмоқда. ***ListBox*** компонентасининг нишони **Standart** куроллар панелида (3.9-расм) жойлашган.



3.9-расм. ListBox компонентаси

Рўйхат формага бошқа объектлар каби жойланади. 3.5-жадвалда **ListBox** компонентасининг ҳусусиятлари келтирилган.

ListBox компонентасининг ҳусусиятлари 3.5-жадвал

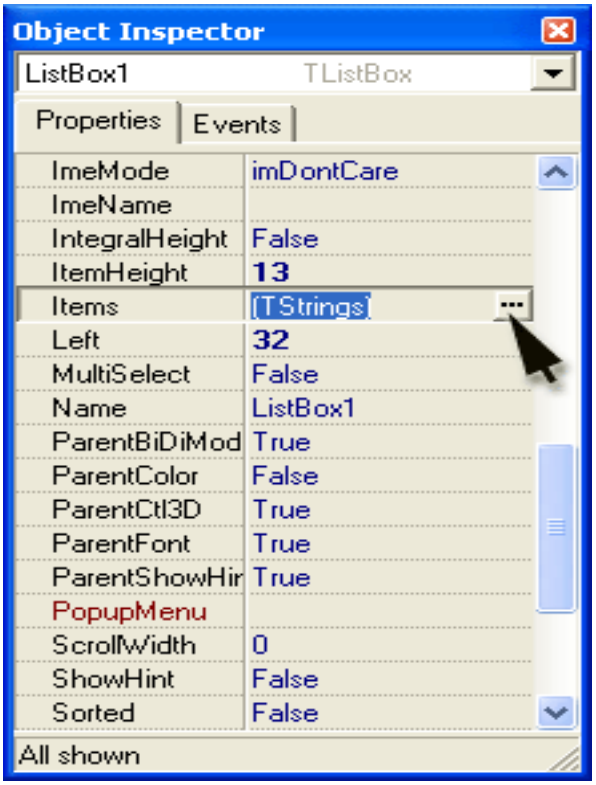
Ҳусусияти	Мазмуни
Name	Компонентанинг номи. Дастурда компонента ҳусусиятларига мурожаат қилиш учун фойдаланилади.
Items	Рўйхат элементлари
Itemindex	Рўйхатдан танланган элемент номери. Рўйхатдаги биринчи элементнинг номери нолга тенг.
Left	Рўйхатнинг чап чегарасидан форманинг чап чегарасигача бўлган масофа.
Top	Рўйхатнинг юқори чегарасидан форманинг юқори чегарасигача бўлган масофа.
Height	Рўйхат майдонининг баландлиги
Width	Рўйхат майдонининг кенглиги
Font	Рўйхат элементларини кўрсатиш учун шрифт
Parent-Font	Она формадан шрифт ҳусусиятларини мерос олиш қилиб белгиси

Items ва **Itemindex** ҳусусиятлари алоҳида эътиборга сазовор. **Items** ҳусусияти рўйхат элементларини сақлайди.

Itemindex рўйхатдан танланган элемент номерини сақлайди. Агар бирорта ҳам элемент танланмаган бўлса, унинг номери минус бирга тенг.

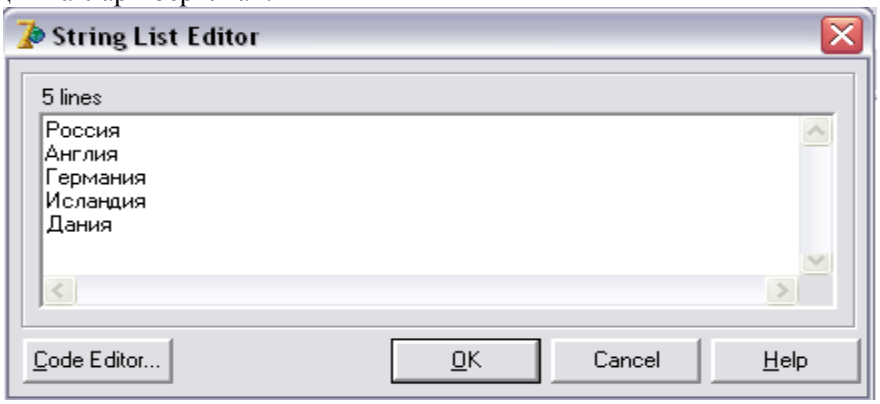
Рўйхат форма яратилаётганда ҳам, дастур ишлаётган вақтда ҳам ҳосил қлиниши мумкин. Форма яратилаётганда рўйхат ҳосил қилиш учун **Object Inspector** ойнасида **Items** ҳусусиятини танлаб, сатрлар рўйхатининг муҳааррири устида сичқонча тугмасини чертиш лозим. (3.10-расм).

Очилган **String List Editor** диалог ойнасида (3.11-расм) ҳар бирини алоҳида сатрга ёзиб, рўйхат элементлари киритилади. Ҳар бир элемент киритилганидан сўнг, янги сатрга ўтиш учун <Enter> тугмаси босилади. Рўйхат тугаганидан сўнг, **ОК** тугмаси чертилади.



3.10-расм. Рўйхат муҳарририни ишга тушириш

3.6-жадвалда формадаги компонеталар рўйхати, 3.7-жадвалда эса компонеталар ҳусусиятларининг қийматлари берилган.



3.11-расм. Рўйхат муҳаррири

Форманинг компонеталари. 3.6-жадвал

Компонента	**Мазмуни**
ListBoxl	Давлатлардан бирини танлаш учун
Editl	Оғирликни фунтларда киритиш учун
Label1, Label2, Label3	Киритиш майдонларини изоҳлаш учун
Label4	Хисоблаш натижасини чиқариш учун

Button1	Хисоблаш процедурасини активлаштириш учун

Компоненталар ҳусусиятларининг қийматлари 3.7-жадвал

ҳусусият	**Қиймати**
Form1 .Caption	***Case*** дан фойдаланишга мисол дастури
Editl. Text	
Label1 . Caption	Давлатни танланг
Label2 .Caption	Давлатлар
Label3 . Caption	Фунтлар
Button1 . Caption	Хисоблаш

Қайта хисоблаш процедураси **Хисоблаш** тугмаси босилганда ишга тушади ва фунтлардаги оғирликни 1 килограммдаги фунтлар қоэффициентига кўпайтиради. Бу коэффициентни рўйхатдан танланган элементга қараб аниқланади.

3.3.-листингда фунтни килограммга ўтказиш дастурининг матни келтирилган.

3.3-листинг. Оғирликни **фунтдан килограммга ўтказиш.**

```
unit funt;
 interface
 uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,   Dialogs, StdCtrls;
 type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Edit1: TEdit;
    Label4: TLabel;
    Button1: TButton;
    ListBox1: TListBox;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
  var  Form1: TForm1;
implementation
{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
begin
{
ListBox1.items.add('Россия');
ListBox1.items.add('Австрия');
ListBox1.iterns.add('Англия');
ListBox1.items.add('Германия');
ListBox1.iterns.add ('Дания');
ListBoxl.iterns.add('Исландия');
ListBox1.iterns.add ('Италия');
ListBox1.items.add ('Нидерландия'); }
ListBox1.itemindex := 0;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
funt:real;//фунтдаги оғирлик
kg:real;//килограммдаги оғирлик
```

```
k:real;//қайта хисоблаш коэффициенти
begin
case ListBox1.Itemindex of
0:      k := 0.4095;//Россия
1:      k := 0.453592;//Англия
2:      k := 0.56001;//Австрия
3..5,7: k := 0.5;//Германия, Дания, Исландия, Нидерландия
6:      k := 0.31762;//Италия
end;
funt := StrToFloat(Edit1.Text);
kg := k*funt;
label4.caption := Edit1.Text + ' ф. - бу '
+ FloatToStrF(kg,ffFixed, 6,3) +  'кг.';
end;
end.
```

Дастурни ишга тушириш

FormCreate ходисаларни қайта ишлаш процедурасига эътибор беринг. Бу ходиса форма очилган заҳоти автоматик тарзда содир бўлади. Бу процедурани ўзгарувчиларга бошланғич қиймат беришда, шу жумладан рўйхатга элементларни қўшиш учун фойдаланиш ҳам мумкин. Келтирилган мисолда рўйхат муҳаррири томонидан дастурнинг иши давомида ташкил қилинган.

менюга

3.4. Цикллар

Кўплаб масалаларнинг ечиш алгоритмлари циклик жараёнларни ўз ичига олади, яъни мақсадга эришиш учун маълум бир буйруқлар кетма-кетлиги бир неча марта такроран бажаришни назарда тутади.

Масалан, билимларни назорат қилиш дастури савол беради, жавобни қабул қилади, жавобнинг баҳосини баллар йиғиндисига қўшади, сўнгра бу амалларни то назорат қилинувчи токи ҳамма саволларга жавоб бериб бўлмагунча яна бир неча марта такроран бажаради. Бошқа мисол. Рўйхатдан бирор фамилияни қидириш талаб қилинган бўлсин. Дастлаб, рўйхатдан биринчи фамилия текширилади, сўнгра иккинчиси, учинчиси ва х.к. Бу жараён токи қидирилаётган фамилия топилгунча ёки рўйхат тугагунча давом этади.

Айрим буйруқлари бир неча марта такроран бажарилиши талаб қилинган алгоритмларни циклли алгоритм, шу буйруқларнинг ўзи эса цикл деб аталади.

Delphi да циклли жараёнларни дастурлаш учун уч ҳил кўринишдаги буйруқлардан фойдаланиш мумкин: ***for, while*** ва ***repeat*** буйруқлари.

менюга

3.5. *For* цикли

Қуйидаги масалани кўрайлик. $y = 3x^2 - 2x + 7$ функциянинг қийматларини в точках —1, 0, 1, 2 ва 3 нуқталардаги қийматларини хисоблаш талаб қилинган бўлсин. (хисобланган қийматлар жадвал кўринишида, форманинг Label майдонига чиқариш керак). Бу масаланинг ечиш дастури қуйидагича бўлиши мумкин:

3.4-листинг.

```
procedure TForm1.ButtonlClick(Sender: TObject);
var y: real;//функциянинг қиймати
x: real;//функциянинг аргументи
dx: real;//аргументнинг орттирмаси
st: string;//жадвални ифодалаш

begin

st := '';  x := -1;  dx := 1;
```

```
y := 3*x*x – 2*x + 7;
st := st + FloatToStr(x) + ' ' + FloatToStr(y) + chr(13);
x := x + dx;

y := 3*x*x – 2*x + 7;
st := st + FloatToStr(x) + ' ' + FloatToStr(y) + chr(13);
x := x + dx;

y := 3*x*x – 2*x + 7;
st := st + FloatToStr(x) + ' ' + FloatToStr(y) + chr(13);
x := x + dx;

y := 3*x*x – 2*x + 7;
st := st + FloatToStr(x) + ' ' + FloatToStr(y) + chr(13);
x := x + dx;

y := 3*x*x – 2*x + 7;
st := st + FloatToStr(x) + ' ' + FloatToStr(y) + chr(13);
x := x + dx;

Label1.Caption := st;

end;
```

Процедура матнидан кўриниб турибдики,

```
y := 3*x*x – 2*x + 7;
st := st + FloatToStr(x) + ' ' + FloatToStr(y) + chr(13);
x := x + dx;
```

буйруқлари кетма-кетлиги беш марта кўрсатилган ва беш марта бажарилади.

Ечилаётган масала етарлича содда бўлишига қарамай, унинг дастурини ёзиш ва компьютер хотирасига киритиш дастурчилар учун бир оз ноқулайликлар пайдо қилади. Биз кўрган масалада x нинг 5 та қийматини хисобга олиш талаб қилинган эди. Агар x нинг 100 та ёки 1000 та нуқтадаги қийматларини топиш талаб қилинса, бу ноқулайлик янада кучайган бўлар эди.

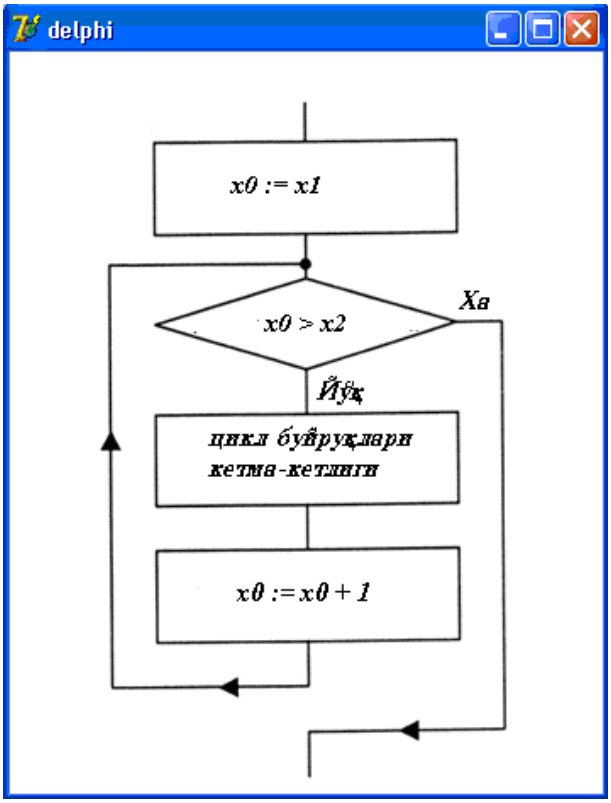
Шунга ўхшаш ноқулайликларнинг олдини олиш учун Delphi дастурлаш тилига ***For*** буйруғини киритилган. У умумий кўринишда қуйидагича ёзилади:

For x0 :=x1 to x2 do
begin цикл буйруқлари кетма-кетлиги ***end;***

Бу ерда ***x0*** **-** цикл жараёнини бошқарадиган ўзгарувчи бўлиб, циклнинг бажарилиш ёки бажарилмаслиги унинг жорий қийматига боғлиқ; ***x1 - x0*** нинг бошланғич қиймати; ***x2 - x0*** нинг охирги қиймати. ***x0, x1, x2*** **–** параметрлар бутун типдаги бўлиши шарт.

For буйруғи қуйидагича бажарилади: Дастлаб ***x0*** – ўзгарувчига ***x1*** **-** бошланғич қиймат берилади. ***x0*** нинг қиймати учун ***x0≥x2*** шарти текширилади. Агар бу шарт ўринли бўлмаса, цикл буйруқлари кетма-кетлиги бир марта тўла бажарилади. Шундан сўнг, ***x0*** - нинг қиймати бирга орттирилади ва яна ***x0≥x2*** шарти текширилади. Бу шарт ўринли бўлмаса, цикл буйруқлари кетма-кетлиги яна бир марта тўла бажарилади. Бу жараён токи ***x0≥x2*** шарти ўринли бўлиб қолгунча давом этаверади. ***x0≥x2*** шарти рост, яъни True бўлганидан кейингина ***for*** дан кейинги навбатда турган буйруқнинг бажаришга ўтилади. Бу фикрларни блок-схемалар (3.12-расм) орқали ифодаланса, ***for*** буйруғининг бажарилиши тартиби янада тушунарлироқ бўлади. Бу блок-схемадан ***x1*** параметри ***x2*** дан катта бўлса, цикл бир марта ҳам бажарилмаслиги ҳамда цикл буйруқлари кетма-кетлиги бир марта бажарилганда, ***x0*** – нинг қиймати автоматик тарзда бирга ортиши кўриниб турибди. Демак, ***For*** буйруғидан такрорланишлар сони олдиндан маълум бўлган ҳолатдагина фойдаланиш мумкин.

Келтирилган фикрларни хисобга олиб, юқоридаги масалани ***for*** буйруғи ёрдамида қуйидагича ёзиш мумкин:



3.12-расм. ***For*** буйруғининг бажарилиш алгоритми

3.5-листинг.

```
procedure TForm1.ButtonlClick(Sender: TObject);
var y: real;// функциянинг қиймати
x: real;// функциянинг аргументи
dx: real;// аргументнинг орттирмаси
st: string;// жадвални ифодалаш
begin
st :='';  x := -1;  dx := 0.5;
for I := l to 5 do begin
y := 3*x*x – 2*x + 7;
st := st + FloatToStr(x) + ' ' + FloatToStr(y) + chr(13);
x := x + dx;    end;
Label1.Caption := st;
End.
```

Ҳар икки дастур вариантини солиштириб кўрилса, иккинчи вариантнинг ҳар томонлама қулайлиги кўриниб турибди. Биринчидан, бу дастур матнини компьютер хотирасига киритиш оз вақт талаб қилади. Иккинчидан, бу дастур матнини ўқиш ва тушуниш аввалгисига қараганда енгил. Учинчидан, зарурат бўлса, ***x*** нинг параметрларини осонгина ўзгартириш мумкин.

For циклининг такрорланишлари сони *x2-x1+1* га тенг бўлади.

Агар циклда такроран бажарилиши керак бўлган буйруқлар кетма-кетлиги фақат битта буйруқдан иборат бўлса, у ҳолда ***do*** хизматчи сўзидан кейинги ***begin*** ва ***end*** ларни ёзмаслик мумкин. Бу ҳолда буйруқнинг умумий кўриниши

For x0 :=x1 to x2 do буйруқ ***;***

тарзида бўлади. Қуйидаги мисолда иккита берилган ***a*** ва ***b*** бутун сонлари орасидаги йиғинди, яъни ***a + (a+1) + ... + b*** ифоданинг қиймати хисобланмоқда. Бу масала учун яратилган формада иккита **Edit**, биттадан **Label** ва **Button** компонеталари қатнашади.

3.6-листинг.

```
procedure TForm1.Button1Click(Sender: TObject);
   var i,s,a,b:integer;
begin
```

```
   s := 0;
   a := strtoint(edit1.Text);
   b := strtoint(edit2.Text);
   for i := a to b do s := s+i;
   label1.Caption := inttostr(s);
end;
```

Дастурни ишга тушириш

Цикл жараёнини бошқарувчи ***x0*** параметрдан циклнинг ичида фойдаланмаслик ҳам мумкин. Бу ҳолда унинг вазифаси талаб қилинган сондаги такрорланишларни таъминлашдан иборат бўлади. Масалан,

$$\sin x + \sin\sin x + \sin\sin\sin x + \ldots + \underbrace{\sin\sin\ldots\sin x}_{n\ марта}$$

ифоданинг қийматини хисоблаш талаб қилинган бўлсин. Бу ерда ***x*** – ўзгарувчидан ташқари яна иккита ўзгарувчи керак бўлади. Биринчиси янги қўшилувчини, иккинчиси эса умумий йиғиндини хисоблаш учун керак. Дастур формасида иккита ***Edit*** (***x*** ва ***n*** учун), биттадан ***Label*** ва ***Button*** компоненталари (зарурат бўлса, ***x*** ва ***n*** ўзгарувчиларни изоҳлаш учун иккита Label ни формага киритиш мумкин) қатнашади.

3.7-листинг.

```
procedure TForm1.Button1Click(Sender: TObject);
  var n,i:integer;
      x,s1,s2:real;
begin
    x := strtofloat(edit1.Text);
    n := strtoint(edit2.Text);
    s1 := x;
    for i := 1 to n do begin
       s1 := sin(s1);
       s2 := s2 + s1;   end;
    label1.caption := floattostr(s2);
end;
```

Юқорида айрилганидек, бу дастурдаги ***i*** ўзгарувчининг вазифаси цикл жараёнини ***n*** марта такрорланишини таъминлашдан иборат.

Юқорида айтиб ўтилдики, ***for*** фойдаланганда ***x0*** –параметрнинг ўзгариш қадами бирга тенг. Бу бууйруқдан ***x0*** – нинг ўзгариш қадами -1 га тенг бўлганда ҳам фойдаланиш мумкин. Бу ҳолда ***for*** буйруғининг умумий кўриниши

For x0 := x1 Downto x2 do
begin цикл буйруқлари кетма-кетлиги ***end;***

тарзида бўлади.

Қуйидаги $\sqrt{3+\sqrt{6+\ldots+\sqrt{3n}}}$ ифоданинг қийматини ҳисоблаш талаб қилинган бўлсин. Кўриниб турибдики, бу ифоданинг қийматини хисоблаш жараёнини энг ички илдиздан бошланиши керак. Агар дастурда йиғиндини хисоблаш учун ***S*** ўзгарувчи киритилган бўлса, унинг бошланғич қиймати 0 га тенг. Навбатдаги йиғинди $S = \sqrt{3i + S}$ формула бўйича топилади. ***i*** нинг қийматлари эса ***n*** дан бошлаб ҳар бир қадамда 1 га қараб камайиб боради. Қўйилган масаланинг дастур матни қуйидагича ёзилади:

3.8-листинг.

```
procedure TForm1.Button1Click(Sender: TObject);
   var n,i:integer;
       s:real;
begin
    n := strtoint(edit1.text);
    s := 0;
```

```
    for i := n downto 1 do s := sqrt(s + 3*i);
    label1.Caption := floattostr(s);
end;
```

менюга

3.6. *While* цикли

While (токи) буйруғи такрорланишлар сони олдиндан маълум бўлмаган ҳолларда цикларни ташкил қилиш учун мўлжалланган.

Одатда ***while*** буйруғи билан берилган аниқликдаги хисоблаш, массив ва файллардан берилган элементни қидириш каби масалаларни дастурлашда фойдаланиш мумкин.

While буйруғи умумий кўринишда қуйидагича ёзилади:

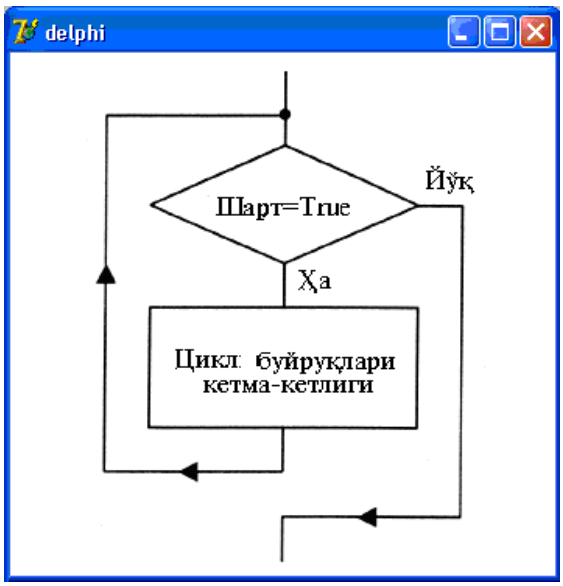
```
while шарт do begin
   // бу ерда циклнинг буйруқлари кетма-кетлиги ёзилади.
end ;
```

Бу ерда ***шарт*** – такрорланиш ёки такрорланмасликни аниқлайдиган мантиқий ифода.

While буйруғи қуйидаги алгоритм асосида бажарилади:

1. Дастлаб ***шарт*** қиймати хисобланади.
2. Агар ***шарт*** нинг қиймати ***False*** (яъни ***шарт*** ўринли бўлмаса), у ҳолда ***while*** буйруғини бажариш шу ерда тугатилади ва ундан кейинги навбатда турган буйруқни бажаришга ўтилади.
3. Агар ***шарт*** нинг қиймати True (яъни, ***шарт*** ўринли) бўлса, у ҳолда ***begin*** ва ***end*** лар ўртасида кўрсатилган циклнинг буйруқлари кетма-кетлиги бир марта тўла бажарилади. Шундан кейин ***шарт*** яна бир марта текширилади. Агар ***шарт*** ўринли бўлса, циклнинг буйруқлари кетма-кетлиги яна бир марта тўла бажарилади. Бу жараён токи ***шарт*** ўринли бўлмай (False) қолгунча давом этаверади.

While буйруғига мос алгоритм қуйидагича ёзилади:



3.13-расм. While буйруғининг ишлаш алгоритми

Алгоритмдан қуйидаги ҳолатлар кўриниб турибди:

1. ***While*** цикли бир марта ҳам бажарилмаслиги мумкин. Бунинг учун унда кўрсатилган шарт биринчи марта текширилгандаёқ ўринли бўлмаслиги етарли.
2. ***While*** циклидаги буйруқлар кетма-кетлиги ҳеч бўлмаганда бир марта бажарилиши учун, ***while*** буйруғидаги шартда қатнашадиган параметрларга аввалдан ***шарт*** ўринли бўладиган қийматларни бериб қўйиш лозим.
3. Циклнинг қачондир тугашини таъминлаш учун, цикл буйруқлари кетма-кетлиги цикл шартида қатнашадиган параметрларнинг қийматларига таъсир этиб бориши (шартда қатнашадиган ўзгарувчиларнинг қийматларини цикл ичида ўзгартириб бориш) керак.

Берилган N натурал сонининг туб ёки туб эмаслигини аниқлаш масаласини кўрайлик. Биз иш бошламасдан аввал бу сонни туб деб фараз қиламиз. Энди қилган фаразимизнинг тўғри ёки нотўғрилигини

аниқлаймиз. Бунинг учун ҳар қандай натурал соннинг 1 дан бошқа бўлувчилари $\lfloor 2, \sqrt{N} \rfloor$ оралиқда ётишини билган ҳолда, N сонини шу интервалдаги барча k бутун сонларга сонларга бўлиб кўрамиз. Агар N сони шу сонлардан бирортасига бўлиниб қолса, у туб эмас, яъни бизнинг фаразимиз нотўғри. Такрорлаш жараёни то k бўлувчининг қиймати $\lfloor \sqrt{N} \rfloor$ сонидан катта бўлиб қолгунча ёки фаразимиз бузилгунча этади. Бу масаланинг дастури қуйидагича ёзилади:

3.9-листинг. N натурал сонининг туб ёки туб эмаслигини аниқлаш.

```
procedure TForm1.Button1Click(Sender: TObject);
   var n,m,k:integer;
       y:string;
begin
    n := strtoint(edit1.Text);
    y := 'Туб';
    m := trunc(sqrt(n));
    k := 1;
    while (k <= m) and (y = 'Туб') do
        begin
          k := k +1;
          if n mod k = 0 then y := 'Туб эмас';
        end;
    label2.caption := 'Берилган сон' + edit1.text + #13 + 'У ' + y;
end;
```

Дастурни ишга тушириш

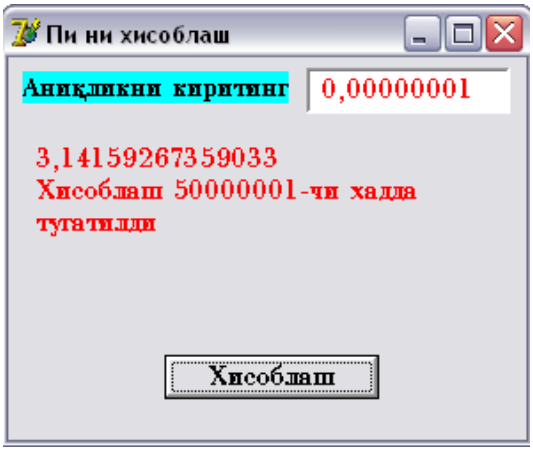
Энди π сонини фойдаланувчи танлаган аниқликда ҳисоблаш дастурини кўрайлик. Бу масаланинг алгоритми асосида

$$1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\ldots+(-1)^{n-1}\frac{1}{2n-1}+\ldots$$

йиғиндининг етарлича катта номерли хадлар учун π⁄4 га яқинлашиши ётади.

Берилган қаторнинг ҳар бир n-чи ҳадини топиш учун $(-1)^{n-1}\frac{1}{2n-1}$ ни ҳисоблаймиз ва йиғиндига қўшамиз. Хисоблаш навбатдаги ҳаднинг қиймати берилган аниқликдан кичик бўлгунча давом эттирилади. -1 нинг даражасини ҳисоблаш учун қуйидагича йўл тутамиз. Маълумки, -1 нинг даражалари ишорасини навбати билан ўзгартириб, бир марта +1, бир марта -1 га тенг бўлади. Шунинг учун, дастурда қиймати -1 га тенг бўлган битта ўзгарувчи оламиз. Ҳар гал цикл бир марта бажарилганда, биз шу ўзгарувчининг қийматини тескарисига ўзгартириб борамиз. Бу бизга -1 нинг даражаларини беради. Олинган ҳадларнинг йиғиндиларини S га йиғиб борамиз. Кўрсатилган аниқликка эришганимиздан сўнг, S нинг қийматини 4 га кўпайтириб қўямиз. Чунки қаторнинг йиғиндиси π⁄4 га тенг эди.

Бу дастурнинг диалог ойнаси қуйидагича: Формага **Edit1** (аниқликни кўрсатиш учун), **Label1** (аниқликни изоҳлаш учун), **Label2** (натижани чиқариш учун) ҳамда **Button1** (хисоблашни бошлаш учун) компоненталарни жойлаймиз.



3.14-расм. π сонини хисоблаш дастурининг диалог ойнаси

3.10-листинг. π сонини хисоблаш

```
procedure TForm1.Button1Click(Sender: TObject);
    var eps,m,s,t:real ;
        n : longint;
begin
    eps := strtofloat(edit1.text);
    m := -1;
    s := 0;
    n := 0;
    t := 1;//цикли камида бир марта бажарилиши учун
    while  abs(t) >= eps do begin
      n := n +1;//янги ҳаднинг номери
      m := -m; // (-1) нинг навбатдаги даражаси
      t := m/(2*n-1);
      S := S + t; end;
      S := S*4;
      label2.caption := floattostr(s) + #13 +
      'Ҳисоблаш ' + inttostr(n) +  '-чи ҳадда ' + #13 + 'тугатилди';
end;
```

Дастурни ишга тушириш

Эслатма: Агар ***while*** буйруғи билан ташкил қилинган циклда такроран бажариладиган буйруқлар кетма-кетлиги битта буйруқдан иборат бўлса, ***do*** хизматчи сўзидан кейин, шу буйруқнинг ўзини ***begin*** ва ***end*** ларсиз ёзиш мумкин. У ҳолда буйруқнинг умумий кўриниши

While шарт do буйруқ;

тарзида ёзилади.

менюга

3.7. Repeat цикли

Repeat буйруғи худди ***while*** каби такрорлашлар сони олдиндан маълум бўлмаган вақтда, дастурнинг айрим буйруқлари кетма-кетлигини такроран бажарилишини таъминлаш мақсадида фойдаланилади. Такрорланиш сони бу усулдаги циклда ҳам дастурнинг бажарилиши давомида аниқланади.

Repeat буйруғи умумий кўринишда қуйидагича ёзилади:

repeat

// буйруқлар кетма-кетлиги

until шарт

Бу ерда ***шарт*** — мантиқий типдаги ифода бўлиб, циклнинг бажариш ёки бажармасликни ҳал қилади.

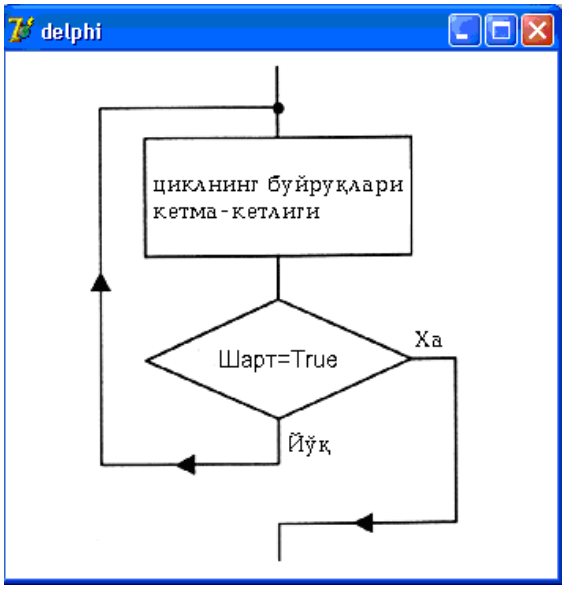
Repeat буйруғи қуйидагича бажарилади:

1. ***Repeat*** ва ***until*** орасидаги буйруқлар кетма-кетлиги бажарилади.

2. Сўнгра мантиқий ифоданинг қиймати хисобланади. Агар мантиқий ифоданинг қиймати ёлғон (False) бўлса, 1-қадамга ўтилади, акс ҳолда 3-қадамга ўтилади.

3. Циклни бажариш тўхтатилади ва навбатда турган буйруқни бажаришга ўтилади.

Шундай қилиб, **repeat** ва ***until*** орасида турган буйруқлар ктема-кетлиги мантиқий ифоданинг қиймати ёлғон (False) бўлса, такрор ва такрор бажарилаверади. Унинг ишлаш алгоритми 4.4-расмда келтирилган.



3.14.-расм. Repeat буйруғининг ишлаш алгоритми

Масала: Фибоначчи сонлари $f_0 = f_1 = 1,\ f_n = f_{n-2} + f_{n-1}$ формулалар билан хисобланади. Олдиндан берилган К сонидан катта бўлган биринчи фибоначчи сонини топинг.

Бу масалани ечиш учун қуйидагича фикр юритамиз. К сони шундай катта бўлиши мумкинки, шунча фибоначчи сонларини сақлаш учун компьютер хотираси кичиклик қилиб қолиши мумкин. Демак, оддий йўл билан бу масалани ҳал қилиш қийин. Бошқача мулоҳаза юритамиз. Фомулалардан кўриниб турибдики, янги ҳадни топишда ундан аввалги икки ҳад ишлайди ҳалос. Қолганларини керак бўлмагани учун ташлаб юбориш ҳам мумкин. Шунинг учун белгилаш киритамиз. Топилаётган янги ҳадни ***f2*** ўзгарувчи билан, ундан битта олдинги ҳадни ***f1*** билан, иккита олдинги ҳадни эса ***f0*** билан белгилаймиз. Нечанчи ҳад топилаётганлигидан қатъий назар, биз шу белгилашга содиқ қоламиз. Ишни бошлаш учун ***f0=f1=1*** қийматдан фойдаланамиз. Янги ***f2*** ҳад топилганидан кейин, у ўзидан кейинги ҳад учун ***f1*** вазифасини ўташи лозим. Уни топишда қатнашган ***f1*** эса ***f0*** вазифасини бажаради.

3.11-листинг. К дан катта бўлган биринчи Фибоначчи сони.

```
procedure TForm1.Button1Click(Sender: TObject);
    var k,f0,f1,f2:longint;
begin
    k := strtoint(edit1.Text);
    f0 := 1; f1 := 1;
    repeat
      f2 := f1 + f0;  f0 := f1;  f1 := f2;
    until f2 >= k;
  label1.caption := inttostr(f2);
end;
```

Дастурни ишга тушириш

Бу масалани ***while*** цикли билан ҳам ҳал қилиш мумкин эди. Аммо, бу ҳолда ***repeat*** ва ***until*** лар орасидаги буйруқларни циклга кирмай туриб ҳам ёзишга тўғри келган бўлар эди, яъни бир ҳил буйруқларни икки марта ёзилади. Биринчи мартасида циклга киришга тайёргарлик учун, иккинчи мартасида эса навбатдаги ҳадларни ҳисоблаш учун. ***repeat*** ва ***until*** циклида эса бундай ноқулайлик йўқ,

ҳамма буйруқлар бир мартадан ёзилган.

Цикллар билан ишлаганда, уларнинг қачондир тугаши лозимлигини назарда тутиш керак. Айрим ҳолларда, бошланғич маълумотларнинг нотўғри киритилганлиги, циклни тугашини ҳал қиладиган мантиқий ифодаларда қатнашадиган ўзгарувчиларнинг қийматларининг ўзгармаслиги ёки бошқа қандайдир сабаблар "чексиз цикл" нинг юзага келтириши мумкин. Бу ҳолда компьютер хеч қачон бу циклни тугата олмайди, демак, масалани охиригача еча олмайди. Бундай ҳатолик синтактик бўлмагани учун, компьютер уни сезмай қолади. Қуйидаги дастур парчаларига эътибор беринг:

```
А) x := 3; y := x mod 2;
   While y <> 0 do begin x := x + 2; y := x mod 2; end;

Б)  x := 3;
   repeat  y := x mod 2;  x := x + 2; until y = 0;
```

Ҳар икки мисолда "чексиз цикл" юзага келганлигини кўриш мумкин.

Диққат! **1. Repeat** ва ***until*** орасидаги буйруқлар кетма-кетлиги ҳеч бўлмаганда бир марта ишлайди. 2. Цикл қачондир тугаши учун **Repeat** ва ***until*** орасидаги буйруқлар кетма-кетлиги мантиқий ифода таркибига кирган ўзгарувчиларнинг қийматларини ўзгартирсин. 3. ***Repeat*** цикли камида бир марта ишлайди, ***For*** ва ***While*** цикллари эса бир марта ҳам ишламаслиги мумкин.

менюга

3.8. *Goto* буйруғи

Биз юқорида ***if*** ва ***case*** буйруқлари навбатдаги буйруқлар кетма-кетлигини бажариш ёки бажармаслиги бирор шартнинг ўринли бўлишига боғлиқлигини кўрган эдик. Шунинг учун у буйруқларни шарти ўтиш буйруқлари деб ҳам юритилади. Амалларнинг бажарилиш тартибини буза оладиган яна бир буйруқ - ***goto*** шартсиз ўтиш буйруғи ҳам мавжуд. У умумий кўринишда қуйидагича ёзилади:

goto тамға;

Бу ерда ***тамға*** – ***goto*** буйруғидан кейин бажарилиши керак бўлган оператордан олдин жойлашган идентификатор.

goto буйруғида фойдаланиладиган тамға тамғаларни эълон қилиш бўлимида эълон қилинган бўлиши лозим. Бу бўлим ***label*** сўзи билан бошланади ва дастур матнида ўзгарувчиларни эълон қилиш бўлимидан олдин жойлаштирилади.

Дастурда тамға ***goto*** буйруғидан кейин бажарилиши керак бўлган буйруқдан олдин қўйилади ва ундан икки нуқта билан ажратиб қўйилади.

3.14-листингда натурал соннинг тублигини текширадиган дастур матни келтирилмоқда. Биз илгари юқорида келтирган натурал соннинг туб ёки туб эмаслигини текшириш дастури айрим камчиликлардан ҳоли эмас эди. Унда киритилган маълумотларни назорат қилиш ҳамда 2 сони учун дастур нотўғри натижа берар эди. ***Goto*** буйруғи ёрдамида ана шу камчиликларни бартараф қилишга уриниб кўрамиз. Бунинг учун 1) агар фойдаланувчи манфий сон киритса, бу ҳақида махсус ахборотни экранга чиқариб, дастур ишини тугатишини ташкил қиламиз; 2) 1 сони киритилган бўлса, уни текширмаймиз; 3) фойдаланувчи 2 сонини киритса, уни тўғридан-тўғри туб деб, дастур ишини тўхтатамиз. 4) Қолган сонларни эса эски алгоритмимиз ёрдамида текширамиз.

3.12-листинг. Туб сон дастури.

```
procedure TForm1.Button1Click(Sender: TObject);
label  Bye, javob;// тамғаларни эълон қилиш бўлими
var n: integer;// текшириладиган сон
k: integer;// бўлувчи
y: string;// натижа
m : integer;// мумкин бўлган энг катта бўлувчи
begin
n := StrToInt(Edit1.text);
if n <= 0 then begin
MessageDlg('Сон нолдан катта бўлиши керак ', mtError, [mbOk], 0);
Edit1.text := '';
goto bye;
```

```
end;
// мусбат сон киритилган бўлса
If n = 1 then begin
y := '1  сони туб ҳам, мураккаб ҳам эмас');
goto javob;
end
else
if  n = 2 then begin
y :='2 сони туб. ';
goto javob;
end
else
// 2 дан катта бутун сони киритилган бўлса
y := 'Туб';
    m := trunc(sqrt(n));
    k := 1;
    while (k <= m) and (y = 'Туб') do
        begin
          k := k +1;
          if n mod k = 0 then y := 'Туб эмас';
        end;
    javob : label2.caption := 'Берилган сон  ' + edit1.text + #13
    + 'Ó ' + y;
Bye :
end;
```

Дастурни ишга тушириш

Дастурлаш бўйича адабиётларда ***goto*** буйруғидан умуман фойдаланмасликка ёки камроқ фойдаланишга чақирилади. Чунки, ундан, айниқса ноўрин фойдаланилганда дастур матни чалкашиб кетади. Аммо, бундай ҳулоса қатъий эмас. ***Goto*** буйруғини ўринли қўллаш имкониятлари мавжуд. Юқоридаги масала бунга мисол бўлиши мумкин.

4-БОБ. БЕЛГИЛАР ВА САТРЛАР

Компьютер фақат сонли маълумотларнигина эмас, балки белгили маълумотларни ҳам қайта ишлаши мумкин. Delphi тили белгили маълумотларни алоҳида олинган белгилар шаклида ҳам, белгилар катма-кетлигидан иборат бўлган сатрлар кўринишидаги маълумотларни ҳам қайта ишлаши мумкин.

4.1. Белгили маълумотлар

Белгили маълумотларни сақлаш ва қайта ишлаш учун ***Ansichar*** ва ***WideChar*** типидаги ўзгарувчилардан фойдаланилади. ***Ansichar*** типи ҳар бири саккиз разрядли иккилик саноқ системасида (байт) кодланадиган ANSI – белгилари тўпламидан иборат. ***WideChar*** типи эса ҳар белгиси икки байт билан кодланадиган Unicode кодлаштириш усулидаги белгилар системасидан иборат.

Windows операцион системасида асосан ANSI кодлаш усулидан фойдаланилади. Рус алифбесини ўз ичига олган ANSI жадвалини Windows-1251 деб аталади. Қуйида шу жадвалдаги айрим белгилар ва уларнинг кодларини келтирамиз.

Айрим хизматчи белгилар **4.1-жадвал**

Код	Белги	Код	Белги
9	Табуляция	27	Enter
11	Янги сатр	32	Бўш жой
13	Абзац охири		

32-127 кодли белгилар **4.2-жадвал**

код	Белги	код	белги	код	белги	код	Белги
32	Бўш жой	56	8	80	P	104	h

33	!	57	9	81	Q	105	i
34	"	58	:	82	R	106	j
35	#	59	;	83	S	107	k
36	$	60	<	84	T	108	l
37	%	61	=	85	U	109	m
38	&	62	>	86	V	110	n
39	'	63	7	87	W	111	o
40	(	64	@	88	X	112	p
41	)	65	A	89	Y	113	q
42	*	66	B	90	Z	114	r
43	+	67	C	91	[	115	s
44	,	68	D	92	ғ	116	t
45	-	69	E	93	]	117	u
46	,	70	F	94	^	118	v
47	⁄	71	G	95	_	119	w
48	0	72	H	96	'	120	x
49	1	73	I	97	a	121	y
50	2	74	J	98	b	122	z
51	3	75	K	99	c	123	[
52	4	76	L	100	d	124	|
53	5	77	M	101	e	125	]
54	6	78	N	102	f	126	~
55	7	79	O	103	g		

192-255 кодли белгилар **4.2-жадвал**

код	белги	код	белги	код	белги	код	белги
192	А	208	Р	224	а	240	р
193	Б	209	С	225	б	241	с
194	В	210	Т	226	в	242	т
195	Г	211	У	227	г	243	у
196	Д	212	Ф	228	д	244	ф
197	Е	213	Х	229	е	245	х
198	Ж	214	Ц	230	ж	246	ц
199	З	215	Ч	231	з	247	ч
200	И	216	Ш	232	и	248	ш
201	Й	217	Ҳ	233	й	249	ҳ
202	К	218	Ъ	234	к	250	ъ
203	Л	219	Ў	235	л	251	ў
204	М	220	Ь	236	м	252	ь
205	Н	221	Э	237	н	253	э
206	О	222	Ю	238	о	254	ю
207	П	223	Я	239	п	255	я

Дастлаб версиялари билан мосликни таъминлаш учун AnsiChar га эквивалент бўлган Char типидан фойдаланиш мумкин.

Белгили типдаги ўзгарувчининг қиймати кўриш мумкин бўлган ихтиёрий белгидан иборат бўла олади. Бу белгилар сифатида лотин ва рус алифбеси харфлари, рақамлар, тиниш белгилари ҳамда махсус белгилар, масалан, "янги сатр", "бўш жой" кабиларни олиш мумкин. (4.1, 4.2, 4.3-жадвалларга қаранг.)

Белгили типдаги ўзгарувчилар ўзгарувчиларни эълон қилиш бўлимида эълон қилинади. Бу эълон умумий кўринишда қуйидагича ёзилади:

Ном: char;

Бу ерда ***ном*** – белгили типдаги ўзгарувчининг номи; ***char*** – белгили типнинг калит сўзи. Масалан:

x: char; ch: char; y9:char;

Дастурнинг бошқа ўзгарувчилари каби char типидаги ўзгарувчилар ҳам қиймат бериш буйруғи ёрдамида қиймат олиши мумкин. Агар char типидаги ўзгарувчи қиймат бериш буйруғи ёрдамида қиймат олс, у ҳолда := белгисидан ўнг томонда char типидаги маълумот (масалан, белги, белгили константа, char типидаги ўзгарувчи ёки ифода) келиши лозим. Белги ва белгили константалар апостроф белгиси билан кўрсатилади.

```
c1 := '*';  c2 := c1;
```

буйруқлари бажарилганидан сўнг, c1 - ўзгарувчи * ни қиймат қилиб олади, c2 – c1 нинг қийматини олади. (Бу ерда ҳар икки ўзгарувчини char типида деб фараз қиламиз).

Char типидаги ўзгарувчиларни бошқа char типидаги ўзгарувчи ёки белгили константа билан таққослаш мумкин. Бу таққослаш шунга асосланадики, ҳар бир белгига қандайдир сон мос қўйилган. (4.1, 4.2, 4.3-жадвалларга қаранг.) Масалан: 'A' белгисига 'a' га қараганда кичикроқ сон мос келади. Шундай қилиб,

'0'<'1'<..<'9'<..<'A'<'B'<..<'Z'<'a'<'b'<..<'z'

деб ёзиш мумкин. Рус алифбеси харфларига лотин харфларига қараганда каттароқ сонлар тўғри келади. Бунда қуйидаги муносабатлар ўринли:

'А'<'Б'<'В'<..<'Ю'<'Я'<'а'<'б'<'в'<...<'э'<'ю'<'я'

Дастур матнида белги ўрнига унинг жадвалдаги кодидан ҳам фойдаланиш мумкин. Фақат бу код олдига # операторини қўйиш лозим бўлади. Масалан, 'в' константаси ўрнига #226 деб ёзиш мумкин. Ёзувнинг бундай шакли одатда хизматчи белгилар ёки дастурни ёзиш вақтида клавиатурадан киритиб бўлмайдиган белгилар учун қўлланади. Масалан, янги сатрга ўтиш учун "абзац охири" белгиси ўрнига #13 тарзидаги ёзув қўлланади.

Белгили маълумотларни қайта ишлашда кўпинча ***Chr*** ва ***Ord*** фунцияларидан фойдаланилади. ***Chr*** функциясининг қиймати бўлиб параметр сифатида кўрсатилган кодга мос келувчи белги хизмат қилади. Масалан,

c:=chr(32) ;

буйруғи бажарилганда, ***c*** ўзгарувчининг қиймати "бўш жой" дан иборат бўлиб қолади. (4.1, 4.2, 4.3-жадвалларга қаранг.)

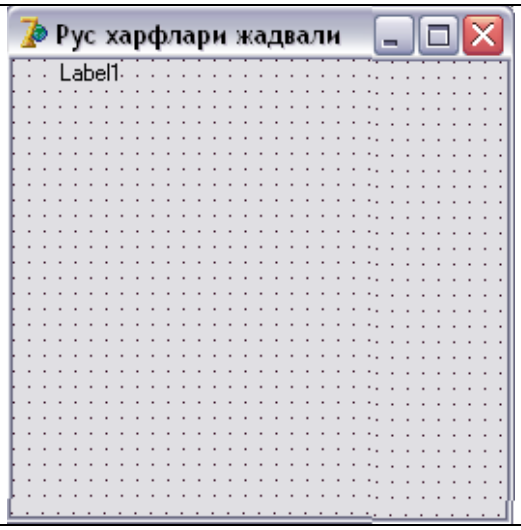
Ord функцияси параметр сифатида кўрсатилган белгининг кодини аниқлайди. Масалан,

k :=ord('*') ;

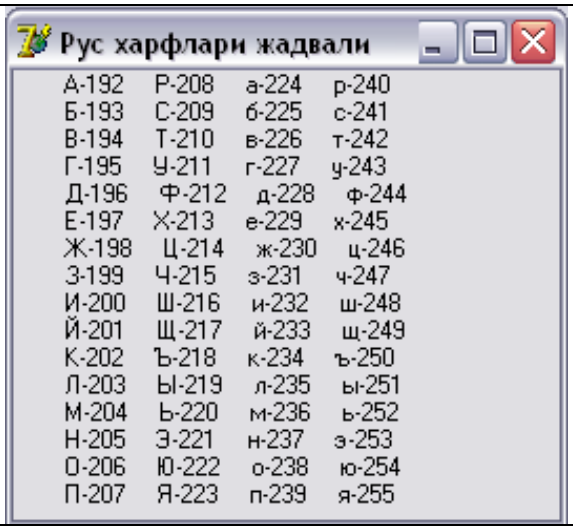
буйруғи бажарилганда ***k*** ўзгарувчининг қиймати "*" белгисининг коди бўлган сонга (42) тенг бўлади.

4.1-листингда берилган дастур матни экранга рус алифбоси харфларини чиқариш учун мўлжалланган. Дастур ойнасининг кўриниши 4.1-расмда келтирилган.

Бу дастурда асосий ишни **OnActivate** ходисаларни қайта ишлаш процедураси бажаради. У илова ойнаси активлаштирилганда ишга тушади. Шунинг учун **TForm1.FormActivate** процедураси форма экранда пайдо бўлган заҳоти автоматик тарзда ишга тушади.



4.1-расм. Илова ойнаси форма ташкил қилинаётганда



4.2-расм. Илова ойнаси иш вақтида

4.1-листинг. Рус харфлари жадвали

```
unit unit1;
interface
uses
Windows, Messages, SysUtils, Classes, Graphics,
Controls, Forms, Dialogs, StdCtrls;
type
TForm1 = class(TForm)
Label1: TLabe1;
procedure FormActivate(Sender: TObject); private
{ Private declarations }
public
{ Public declarations }
end;
var
Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.FormActivate(Sender: TObject);
var
st:string;//жадвал белгиларнинг сатри сифатида ҳосил қилинади
dec: byte;//белгининг коди
i,j:integer;//жадвалдаги сатр ва устун номери
begin
st := '';
dec := 192;
for i := 0 to 15 do//ўн олтита сатр
begin
dec := i + 192;
for j := 1 to 4 do//тўртта устун
begin
st := st + chr(dec) + '-' + IntToStr(dec) + ' ';
dec := dec + 16;
end;
st := st + #13;//экраннинг янги сатрга ўтиш
end;
Label1.caption := st;
end;
end.
```

Дастурни ишга тушириш

Рус харфларининг жадвали иловасининг формасида фақат битта Label1 компонентаси жойлаштирилган. Жадвалнинг устунлари бир ҳил кенгликка эга бўлиши учун Label1.Font.Name ҳусусиятига барча белгилари бир катталикда бўлган шрифт номи, масалан, **Courier New Cyr** қиймати берилади. Дастур ойнасининг ишчи экрани 4.2-расмда берилган.

менюга

4.2. Сатрлар

Сатрлар ***shortstring, Longstring*** ҳамда ***widestring*** типидаги маълумот сифатида бериши мумкин. Бу типлар битта сатрга ёзиш мумкин бўлган белгиларнинг энг кўп сони, ўзгарувчиларни компьютер хитирасида сақлаш учун ажратиладиган жой усуллари ва белгиларнинг корлаш усуллари билан бир-биридан фарқланади.

Shortstring типидага ўзгарувчи учун хотирадан жой статик, яъни дастурнинг бажарилиши бошлангунча ажратилади. Бундай сатрдаги белгиларнинг энг кўп сони 255 тадан ошмайди. ***Longstring*** ва ***widestring*** типидаги ўзгарувчиларга эса хотира динамик усулда, яъни дастурнинг бажарилиши вақтида ажратилади. Шунинг учун бундай сатрларнинг узунлиги чегараланмайди.

Бундан ташқари, дастурларда универсал сатрли тип ***String*** дан ҳам фойдаланиш мумкин. У ***Shortstring*** типига эквивалент.

Сатрли типдаги ўзгарувчилар ўзгарувчиларни эълон қилиш бўлимида эълон қилиниши шарт. Бу эълон умумий кўринишда қуйидагича бўлиши мумкин:

Ном: String;

ёки

Ном: String [узунлик];

Бу ерда ***Ном*** – ўзгарувчининг номи: ***string*** — сатрли типни англатувчи калит сўз; [узунлик] – сатрнинг энг катта узунлигини кўрсатувчи бутун типдаги константа.

Сатрли типдаги ўзгарувчиларни эълон қилишга мисолар:

name: string[30]; buff: string;

Агар старли ўзгарувчини эълон қилишда сатрнинг узунлиги кўрсатилмаган бўлса, у ҳолда унинг узунлигини 255 га тенг деб қабул қилинади, яъни

satr: string [255]; ҳамда satr: string;

эълонлари тенг кучли.

Дастур матнида, сатр ҳисобланган белгилар кетма-кетлиги (сатрли константа) апостроф (' ') белгилари орасида кўрсатилади. Масалан, parol сатрли ўзгарувчисига қиймат бериш ифодасини қуйидагича ёзиш мумкин:

parol := 'Маҳфий сир';

ёки

parol := '2006';

Шуни алоҳида таъкидлаш керакки,

Parol := 2006;

буйруғи нотўғри, чунки ***parol*** сатрли тип, 2006 эса бутун сонли типдаги маълумот ҳисобланади. Бундай буйруқлар компиляция қилиш жараёнида учраб қолса, экранга

incompatible types: 'Char' and 'Integer'

(Char ва Integer типлари бир-бирига мос эмас)

кўринишидаги ахборот чиқарилади.

Сатрли типдаги маълумотларни =, <, >, <=, >= муносабат белгиларидан фойдаланиб string типидаги бошқа сатрлар билан таққослаш мумкин. Сатрлар дастлабки белгиларидан бошлаб белгима-белги таққосланади. Агар таққосланаётган сатрларнинг хамма белгилари бир ҳил бўлса, бундай сатрларни ўзаро тенг деб қаралади. Агар дастлабки белгиларидан бошлаб, бир ҳил позициядаги белгилари ҳар ҳил бўлса, бу сатрларнинг ичида бир ҳил бўлмаган биринчи белги учраган позицияда қайси сатрнинг белгиси каттароқ кодга эга бўлса, шу сатр катта ҳисобланади +уйидаги жадвалда сатрларни таққослашга мисоллар келтирамиз:

Сатрларни таққослаш **4.4-жадвал**

1-сатр	2-сатр	Таққослаш натижаси
Алиев	Алиев	Бу сатрлар тенг
Ворисова	Ворисова	1-сатрдан 2-сатр катта
Амиров	Амирова	1-сатр 2-сатрдан кичик
Basic	Delphi	1-сатр 2-сатрдан катта

Сатрли маълумотлар устида таққослаш амалидан ташқари, қўшиш амалини ҳам бажариш мумкин. Сатрларни қўшганда шу сатрлардан биринчисининг ўнг томонига иккинчисини ёнма-ён ёзиш натижасида янги сатр ҳосил бўлади. Масалан,

```
First_name := 'Алиев' ;
last_name := 'Вали';
ful_name := first_name + ' ' + last_name;
```

амаллари бажарилганда, ful_name ўзгарувчисининг қиймати 'Алиев Вали' га тенг бўлади.

Сатрли маълумотнинг бирор белгисини кўрсатиш керак бўлса, унинг номи ва квадрат қавслар ичида шу белгининг номери ёзилади. Масалан, юқоридаги ***ful_name*** ўзгарувчиси учун ***ful_name[8]*** ёзуви "а"

белгисини билдиради.

менюга

4.3. Сатрлар устида амаллар бажариш

Delphi тилида сатрла билан ишлаш учун бир қатор функция ва процедуралар киритилган. Улар икки гуруҳга - сонли ва сатрли қиймат оладиган функция ва процедураларга бўлинадилар. Дастлаб сонли қиймат оладиган функция ва процедураларни кўрайлик.

Length функцияси. Length функцияси берилган сатрнинг узунлигини аниқлаш учун хизмат қилади. Бу функциянинг аргументи сатр типидаги ифодадан иборат. Length функциясининг қиймати (бутун сон) сатрдаги белгилар сонига тенг бўлади. Масалан,

```
n := length('Алиев');
m := length('Delphi тили');
```

буйруқлари бажарилганда n ва m ўзгарувчиларнинг қийматлари мос равишда 5 ва 11 га тенг бўлади.

Эслатма: "Бўш жой" ҳам битта белги ҳисобланади.

Pos функцияси. Pos функцияси бир сатр иккичисининг таркибида мавжудми ёки йўқми деган саволга жавоб беради. Бу функция умумий кўринишда қуйидагича ёзилади:

pos (1-camp, 2-camp) ;

бу ерда ***1-camp*** — сатр типидаги ўзгарувчи ёки константа бўлиб, уни сатр типидаги ***2-camp*** такрибидан қидирилади. ***2-camp*** нинг таркибида нечанчи позициядан бошлаб ***1-camp*** нинг учраши ***pos*** функциясининг қийматини аниқлайди. Агар учрамаса бу функциянинг қиймати нолга тенг бўлади. Масалан,

```
p :=pos('тил', 'Delphi тили');
q :=pos('Basic', 'Delphi тили');
```

буйруқлари бажарилганда, ***p*** – нинг қиймати 8 га, q - эса 0 тенг бўлади.

Val* процедураси.** ***Val процедураси ҳарфий катталикларни сонли катталикка айлантириш учун хизмат қилади. Бу процедура умумий кўринишда қуйидагича ёзилади:

Val(satr, a, b);

Бу ерда ***satr*** – сонли катталикка айлантирилаётган сатр типидаги маълумот; ***a*** – ***satr*** нинг биринчи рақам бўлмаган белгисигача турган рақамлар кетма-кетлиги ифодалайдиган бутун сон, ***b*** - ***satr*** нинг биринчи рақам бўлмаган белгиси турган позицияни кўрсатувчи бутун сон бўлиб, хатолик кодли деб ҳам аталади.

Val(satr, a, b) процедурасининг бажарилиши намуналари жадвали

satr	*a*	*B*
12345	12345	0
1236,86565	1236	5
0,1276	0	2
-12,23	12	4
+14.57	14	4
%14.445	0	1
Ф12.13	0	1

Энди сатрли қиймат оладиган функция ва процедураларни кўрайлик.

Concat функцияси. Concat функцияси бир нечта сатрларни қўшиб битта сатрга йиғиш учун хизмат қилади. Унинг умумий кўриниши қуйидагича:

Camp :=concat(camp1, ... campN);

Бу ерда ***camp1, ... campN*** – битта сатрга йиғилаётган сатрли ўзгарувчи ёки константалар; ***camp*** – натижавий сатр. Масалан,

```
st1 := 'Delphi '; st2 := ' жуда бой ';  st3 :='тил!';
st := concat(st1, st2, st3);
```

буйруқлари бажарилганидан сўнг, ***st*** сатрли ўзгарувчининг қиймати "'Delphi жуда бой тил!" га тенг бўлади.

Delete процедураси. Delete процедураси сатрнинг маълум бир қисмини ўчиришга имкон беради. Бу процедура умумий кўринишда қуйидагича ёзилади:

delete(camp, n, m)

бу ерда ***camp*** – сатр типидаги ўзгарувчи ёки константа; ***n*** – ўчириш бошланадиган белгининг номери; ***m*** – ўчириладиган белгилар сони. Масалан,

N := 'Тошкент шаҳри гўзал'; delete(n, 9, 6);

буйруқлари бажарилганидан сўнг, n – нинг қиймати 'Тошкент гўзал' га тенг бўлади.

+уйида келтирилаётган ***while*** буйруғи ***st*** сатрининг бошида келган "бўш жой" белгиларини ўчиради:

while(pos(' ', st) =1) do delete (st, 1, 1);

Бўш жойларни ***delete (st, 1, 1)*** процедураси ўчиради. Бу буйруқ циклда ***st*** сатрининг биринчи белгиси "бўш жой" бўлган ҳолларда (бу ҳолда ***pos (' ',st)*** функциясининг қиймати бирга тенг) ишлайди.

Copy функцияси. Copy функцияси сатрнинг маълум бир қисмини ажратиб олиш учун ишлатилади. **Copy** функцияси умумий кўринишда қуйидагича ёзилади:

copy(camp, n, m) ;

бу ерда ***camp*** – парчаси ажратиб олинадиган сатр, ёки сатр типидаги ўзгарувчи; ***n*** —ажратиш бошланадиган биринчи белгининг ўрни; ***m*** – ажратиб олинадиган белгилар сони. Масалан,

st :='Профессор Олимов'; fam :=copy(st,11, 6) ;

бўлса, ***fam*** ўзгарувчининг қиймати "Олимов" га тенг бўлади.

5-БОБ. КОНСОЛ ИЛОВАЛАР

5.1. Кириш

Ушбу электрон қўлланма Windows учун дастурлашга мўлжалланганига қарамай, консол иловаларни ҳам назардан четга чиқариш ярамайди. Консоль — бу монитор ва клавиатурани битта деб қарашдир. Консолли иловалар деганда эса MS DOS операцион системаси (ёки DOS ойнаси) учун мўлжалланган дастурлаш тушунилади. Бу ҳолда киритиш қурилмаси – клавиатура, чиқариш қурилмаси эса – белгили маълумотларни акслантириш режимида ишлайдиган монитордан иборат.

Консолли иловалар дастурлашнинг умумий масалаларини кўришда жуда ҳам қулай ҳисобланади. Чунки, бунда асосий эътибор масалани ҳал қилишга қаратилади.

Консол иловларни яратишдан аввал маълумотларни экранга чиқариш ва клавиатурадан киритиш буйруқлари билан танишамиз.

менюга

5.1. Киритиш ва чиқариш буйруқлари

Кўпинча масалаларни ечиш жараёнида масаланинг шартида берилган маълумотларни клавиатура орқали киритишга тўғри келиб қолади.

Read оператори ўзгарувчиларга қийматларни клавиатура ёрдамида беришни ташкил қилиш учун ишлатилади. Бу оператор умумий кўринишда қуйидагича ёзилади:

read (ўзгарувчилар рўйхати);

Рўйхатдаги ўзгарувчилар бир-бирларидан вергул билан ажратилади. Масалан:

read *(r,k,h)*; .

Read буйруғини бажарган ЭҲМ ишдан тўхтайди ва рўйхатда кўрсатилган барча ўзгарувчилар учун қиймат киритилишини кутади. Клавиатурадан киритилаётган маълумотлар бир-биридан бўш жой белгиси билан ажратилади. Киритилган маълумотлар тартиб рақамларига қараб мос равишда берилган рўйхатдаги ўзгарувчиларга қиймат қилиб берилади. Бошқача айтганда, биринчи киритилган маълумот рўйхатдаги биринчи ўзгарувчига, иккинчи маълумот иккинчисига ва хоказо тартибда берилади. Юқоридаги оператор учун клавиатурадан қуйидаги маълумотлар киритилган бўлсин:

2.34 15 Delphi

У холда ***r*** га 2.34, ***k*** га 15, ***h*** га эса «Delphi» қийматлари берилади ва дастурнинг кейинги буйруқлари ўзгарувчиларнинг ана шу қийматлари учун бажарилади.

+иймат олаётган ўзгарувчи билан унга берилаётган қиймат бир хил типга мансуб бўлиши лозим. **Char** ёки **string** типидаги маълумот киритилаётганда уларни апостроф орасига олиш шарт эмас. **Real**

типида маълумот киритилаётганда эса бутун сонларни хам киритишга рухсат берилади. (Бу ҳолда киритилган 10 сонини 10.00 тарзида қабул қилинади.) Boolean типидаги маълумот сифатида фақат **false** ёки **true** қийматларидан бирини киритиш мумкин халос.

Клавиатурадан киритилган маълумотлар сони **Read** операторида берилган рўхатдаги ўзгарувчилар сонидан кам бўлмаслиги лозим. Акс ҳолда, рўйхатдаги қайсидир ўзгарувчи қиймат олмагани сабабли навбатдаги операторлар бажарилмай тураверади.

Агар киритилган маълумотлар сони **Read** оператори билан кўрсатилган ўзгарувчилар сонидан кўп булса, бунинг зарари йўқ. Чунки ортиқча қийматлар ёки навбатдаги **Read** даги ўзгарувчиларга қиймат қилиб берилади. Масалан, битта дастурда

Read *(a,b,s);* **Read** *(x,y);*

операторларига жавобан клавиатурадан

2.3 -1.5 2.4 22 -0.05 4.125

маълумотлари киритилган бўлса, ***a*** га 2.3, ***b*** га -1.5, ***c*** га 2.4 қийматлари берилса, ***x*** ўзгарувчи 22, ***y*** эса -0.05 қийматлари берилади. Ортиқча киритилган 4.125 дан эса ЭҲМ фойдаланмайди, яъни ташлаб юборади.

Readln опереатори рўйхатда кўрсатилган ўзгарувчиларга қиймат киритилганидан сўнг, курсорни янги сатрнинг бошига ўтказиб қўяди. Бу холда ортиқча маълумотларнинг барчаси ташлаб юборилади, навбатдаги ***Read*** ёки ***Readln*** ёрдамида берилган ўзгарувчиларга эса қиймат қилиб янги сатрнинг бошида киритилган маълумотлар олинади. Масалан:

Readln *(a,b,s);* ***Read*** *(x,y);*

операторлари учун клавиатура орқали

2.3 -1.5 2.4 22 3.75
-0.05 4.125

кўринишидаги маълумотлар киритилган бўлса, ***a, b, c*** ўзгарувчилар мос равишда 2.3, -1.5, 2.4 қийматларини олса, ***x*** билан ***y*** ўзгарувчилар -0.05 ва 4.125 ни олади.

Write оператори турли ҳисоблаш натижаларини, матнларни ҳамда арифметик ифодаларнинг қийматларини ҳисоблаб дисплей экранига чиқариш учун ҳизмат қилади. Бу оператор умумий ҳолда қуйидагича ёзилади:

write(чиқариладиган маълумотлар рўйхати); .

Чиқариладиган маълумотлар бир-биридан вергул билан ажратилади.

Экранга чиқариш керак бўлган матнларни апостроф белгиси билан кўрсатилади. Масалан:

write *('Delphi tili');*

буйруғи бажарилганда экранга

Delphi tili

ёзуви чиқарилади.

write оператори ёрдамида Delphi дастурлаш тили қоидалари билан ёзилган арифметик ифодаларнинг қийматларини ҳам ҳисоблаш мумкин. Масалан:

$$write(3*4+15/3-10.5);$$

буйруғининг натижаси қавслар ичида берилган ифоданинг қиймати бўлган

$$6{,}5000000000E+00$$

кўринишдаги сонни экранга чиқаришдан иборат бўлади.

Агар ***write*** ёрдамида экранга чиқариш талаб қилинган маълумотлар сифатида ўзгарувчилар рўйхати берилган бўлса, у ҳолда бу ўзгарувчиларнинг қийматлари ЭҲМ хотирасидан қидириб топилади ва экранга чиқарилади. Фараз қилайлик, бирор дастурнинг бажарилиши давомида *x, y, z* ўзгарувчилар мос равишда 23, 123.12, 'Delphi' қийматларини олган бўлсин. У ҳолда

write *(x,y,z)*;

операторининг бажарилиши натижасида экранда

$$2.3000000000E+011.2312000000E+02\ \text{Delphi}$$

кўринишидаги маълумотлар чиқарилади. Бу ёзувлардан кўриниб турибдики, экрандаги сонли маълумотлар

ўқиш ва тушуниш учун бир оз ноқулай. Ана шу ноқулайлик олдини олиш учун экранда узатиладиган сонли маълумотларни маълум бир ўлчамга солишга (форматлашга) тўғри келади. Ўлчам одатда икки бутун сондан иборат бўлиб, уларнинг биринчиси умумий рақамлар сонини, иккинчиси эса каср қисмидаги рақамлар сонини белгилайди. Ўлчам форматланаётган ўзгарувчидан кейин икки нуқта (:) орқали аниқланади ва фақат шу ўзгарувчигагина тегишли бўлади. Юқоридаги маълумотларнип форматлаб, экранга узатайлик:

write(*x:4:2, y:6:2, z*);

Бу буйруқ таъсирида маълумотлар қуйидаги кўринишда экранга чиқарилади:

23.00 123.12 Delphi

Бутун сон типидаги маълумотларда ҳақиқий қисм бўлмаслигини ёдда тутиш зарур.

Экранда (кўрсаткич) курсор мавжуд бўлиб, у маълумотларни қайси жойдан бошлаб киритилиши ёки чиқарилиши кераклигини кўрсатади. Навбатдаги маълумот курсор турган жойдан бошлаб киритилади ёки чиқарилади. Юқорида биз ***readln*** ёрдамида ана шу курсорни навбатдаги сатрнинг биринчи позициясига ўтказишни кўрган эдик. Шу ҳолатни ***writeln*** ёрдамида ҳам такрорлаш мумкин.

Writeln оператори талаб қилинган маълумотларни экранга чиқарганидан кейин, курсорни янги сатрнинг бошига ўтказади. Масалан:

write *(x);* ***write*** *(y);* ***write*** *(z);*

буйруқлари маълумотларни

2.3000000000+E01 1.231000000E+02 Delphi

кўринишида экранга чиқарса,

writeln(x); writeln(y); write(z);

буйруқлари маълумотларни экранга

2.3000000000+E01
1.231000000E+02
Delphi

тарзида чиқарилишини таъминлайди.

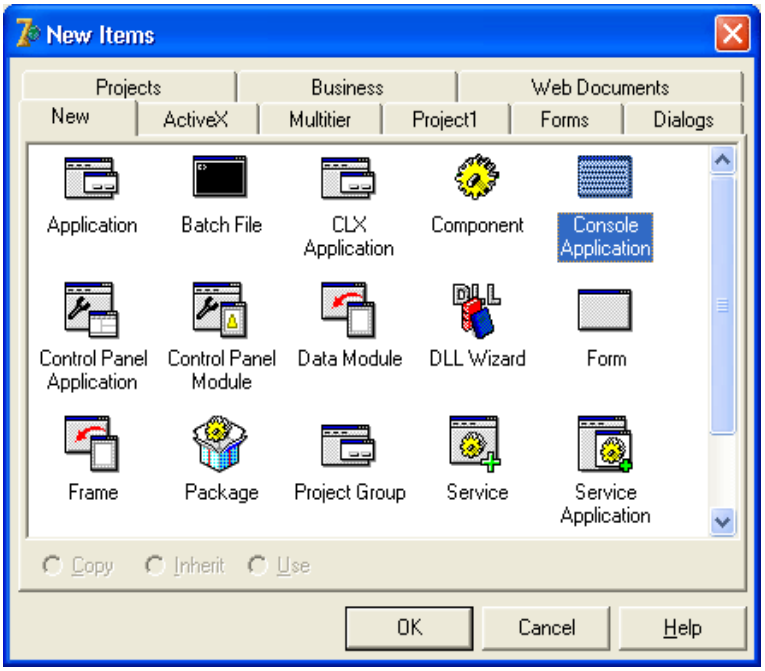
Readln ва ***writeln*** операторларидан ҳеч қандай аргументсиз ҳам фойдаланиш мумкин. ***Readln*** буйруғи ENTER тугмасини босилиши кутади ва курсорни янги сатрнинг бошига ўтказса, ***writeln*** эса шунчаки курсорни янги сатрнинг бошига кўчиради.

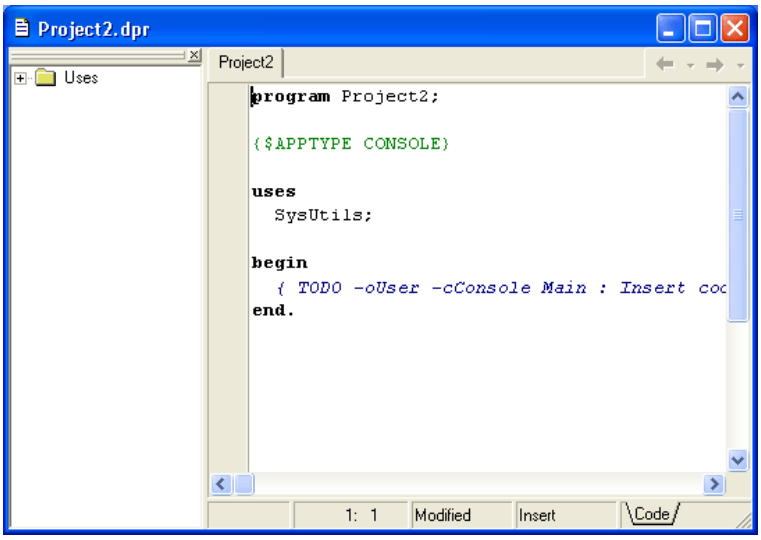
Эслатма: Агар клавиатурадан киритилаётган маълумот билан уни қиймат қилиб оладиган ўзгарувчининг типларига бир-бирига мос бўлмаса, дастур ўз ишини тўҳтатади ва экранга йўл қўйилган ҳатолик ҳақида ахборот чиқарилади.

менюга

5.2. Консолли иловалар яратиш

Консолли иловалар қуйидагича усулда яратилади. Дастлаб **File** менюсидан **New / Other Application,** пункти танланади. Сўнгра **New** тугмасини босиб **New Items** диалог ойнасини очамиз. Ундан **Console Application** тугмасини босамиз. Натижада экранда **Projectl.dpr** ойнаси пайдо бўлади. У ерда консолли иловаларнинг бош процедураси жойлашган. Бу ойнада дастурнинг буйруқларини киритиш мумкин.





5.1-расм. Консолли иловалар бош процедурасининг шаблони

Консолли иловалар ***program*** буйруғи билан бошланади. Сўнгра дастурнинг номи ёзилади. Дастлаб у бошланғич лойиха номи билан бир хил бўлади. Дастур матнини сақлаш вақтида у дастурчи қўйган ном билан автоматик тарзда алмаштириб қўйилади.

Шуни ёдда тутиш кераккки, консолли иловаларни Windows да яратилади, DOS дастури каби ишлатилади. DOS да ASCII кодлаш усули, Windows да эса ANSI усули қўлланади. Уларда рус алифбесининг харфлари турли кодларга эга. Бу консолли иловалардаги рус алифбесида ёзилган изоҳлар ўрнига бошқа матнни чиқаришга олиб келади. Шунинг учун консолли иловаларда маълумотларни лотин алифбесида чиқариш тавсия қилинади.

Агар зарурат бўлса, консолли иловаларда рус алибесидаги ахборотларни экранга чиқариш учун ANSI-сатрни ASCII-сатрига ўтказувчи қайта кодлаш функциясини яратиш ва фойдаланишга тўғри келади. Агар бу функцияни RUS деб атасак, маълумотларни рус алифбесидаги (кирилча) маълумотларни экранга чиқариш буйруқлари қуйидагича кўринишда бўлади:

writeln(Rus('Delphi тили ажойиб тил'));

5.1-листингда намуна сифатида фойдаланувчи киритган оғирликни фунт ўлчов бирлигидан килограммга ўтказиш дастури келтирилмоқда. Маълумотларни экранга чиқариш учун ANSI-сатрни ASCII-сатрига ўтказувчи қайта кодлаш функцияси RUS дан фойдаланилади.

5.1-листинг. Оғирликни фунтдан килограммга ўтказиш (консолли иловалар)

```
program funt_k;
{$APPTYPE CONSOLE}
```

```
 //RUS - ANSI-сатрни ASCII-сатрига қайта кодлаш  функцияси
function Rus(mes: string):string;
 //ANSI да кирилча ҳарфлар 192 дан 255 гача кодланади
 //ASCII да -128 дан 175 гача (А..Яа..п) ва 224 дан 239 гача (р..я)
Var i: integer; //қайта кодланаётган белгининг номери
begin
for i := 1 to length(mes) do
case mes[i] of
'А'..'п' : mes[i] := Chr(Ord(mes[i]) - 64);
'р'..'я' : mes[i] := Chr (Ord(mes [i] ) -16);
end;
rus := mes; end;
 //Асосий дастур
Var f : real; //  фунтдаги оғирлик
      w : real;    //граммдаги оғирлик
       k : integer;  //килограммлар
       g : integer;   //граммлар
  //w = f*0,4095 + k*1000 + g
begin
writeln(Rus('Фунт ва килограммлар'));
writeln(Rus('Оғирликни фунтда киритинг ва  <Enter> ни босинг'));
write('> ');
readln(f);
w := f * 409.5;  //Бир фунт фунт - бу 409,5 гр.
if w > 1000 then begin
k := Trunc(w/1000); g := Round(w - k*1000);
end else
begin k := 0; g := Round(w) ; end;
write(f:4:2, Rus(' ф. - бу '));
if k >= 1 then write(k, Rus(' кг. '));
writeln(g, Rus(' гр.'));
write(Rus('Ишни тугатиш учун  <Enter> ни босинг')); readln;
end.
```

Дастурни ишга тушириш

Дастур матни {$APPTYPE CONSOLE} сатри билан бошланмокда. У бир қараганда изоҳга ўхшайди, аммо бундай эмас. Чунки, очилган қавсдан кейин пул бирлиги белгиси турибди. Бу куррсатма компилятор учун мўлжалланган. Унга риоя қилган компилятор дастурни консолли иловалар каби генерация қилади. .

Консолли иловаларнинг компиляцияси одатдаги усулда амалга оширилади, яъни **Project** менюсидан **Compile** буйруғи танланади.

Муваффақиятлди компиляциядан кейин, дастурни **Run** менюсидан **Run** буйруғи билан **ишга туширилади. К**онсолли иловалар ишга тушганда экранда DOS-дастурларнинг стандарт ойнаси пайдо бўлади. 5.2-рарсмда консолли илова ишлаётган DOS-ойнасининг кўриниши тасвирланган.

Консолли иловаларнинг лойихаси стандарт усулда сақлаб қўйиш мумкин. **File** менюсидан **Save** буйруғини танланганда экранда **Save Project** нинг диалог ойнаси пайдо бўлади ва унга лойиха номини киритиш мумкин.

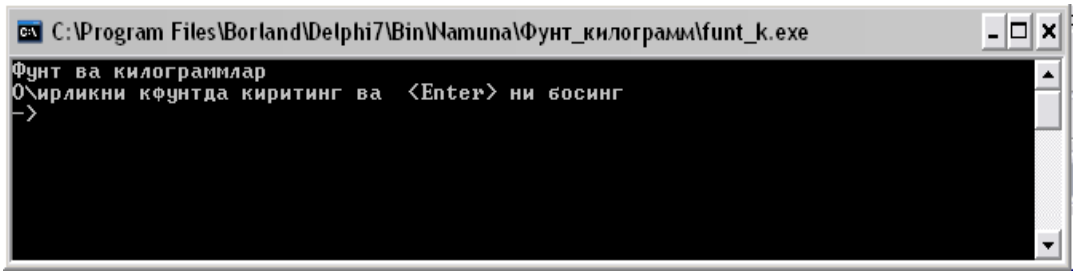


Рис. 5.2. Консолли иловалар ишлаётган DOS-ойна

6-боб. **МАССИВЛАР**

6.1. Кириш

Жадвал катталиклари ёки массивлар бир хил типдаги ва кўплаб сондаги маълумотларни сақлаш ҳамда қайта ишлаш учун мўлжалланган. Масалан: фамилиялар рўйхати, имтихондан талабаларни олган баҳолари, кундалик ўртача ҳарорат ва ҳоказоларни массив сифатида қабул қилиш мумкин. Фараз қилайлик, 5 та баҳони ўқиш, улар ичидаги энг катта баҳони топиш ва қолган ҳамма баҳоларнинг ундан қанчага фарқ қилишини топиш талаб қилинган бўлсин. Бу дастурда 5 та баҳонинг ҳаммасини киритиб бўлмагунча, уларнинг энг каттаси ва бошқа баҳоларни ундан қанчага фарқ, қилишини топиб бўлмайди. Бунинг учун ҳамма баҳоларни ЭҲМ хотирасида сақлашга тўғри келади. Турган гапки, баҳоларнинг ҳаммаси *integer* типида бўлади. Уларни сақлаш учун ***integer*** типидаги 5 та турли ўзгарувчиларни киритиш мумкин. Баҳоларни билдирадиган ўзгарувчиларни бошқалари билан алмаштириб қўймаслик учун BAHO1, BAHO2, BAHO3 ёки шунга ўхшаш қилиб танлаш мақсадга мувофиқ ҳисобланади. Агар баҳолар сони кўп бўлса (айтайлик 100 та бўлса), бу усул ҳам яхши натижа бера олмайди. Ўзгарувчиларни бундай кўринишда танлаш дастурни мураккаблаштириб юборади. Чунки, дастурда қатнашадиган ўзгарувчилар сони қанча кўп бўлса, уни ўқиш ва тушуниш шунча қийин бўлади. Бундай ҳолатлариинг олдини олиш учун Delphi тилида массивлар тушунчаси киритилган.

Массив — бу маълумотларнинг маълум бир структурага эга бўлган, бир ҳил типдаги ва умумий номга эга бўлган туридир. Массивларда мазмуни ва шакли бир ҳил бўлган катта ҳажмдаги жадваллар, рўйхатлар каби маълумотларни сақлаш катта афзалликларга эга.

Жадвал катталиклари ёки массивлар бир хил типдаги ва кўплаб сондаги маълумотларни сақлаш ҳамда қайта ишлаш учун мўлжалланган. Масалан: фамилиялар рўйхати, имтихондан талабаларнинг олган баҳолари, кундалик ўртача ҳарорат ва ҳоказоларни массив сифатида қабул қилиш мумкин.

менюга

6.2. Массивларни эълон қилиш

Массивлар ҳам фойдаланишдан аввал, бошқа ўзгарувчилар каби ўзгарувчиларни эълон қилиш бўлимида эълон қилиниши керак. Умумий ҳолда массивларни қуйидагича эълон қилиш мумкин:

Ном: array *[қуйи индекс .. юқори индекс*] ***of*** тип

Бу ерда ном- массивнинг номи; ***array*** - Delphi да ***ном-*** массив эканлигини англатувчи хизматчи сўз; *қуйи индекс .. юқори индекс* – массив элементларининг ўзгариш диапазонини кўрсатувчи бутун типдаги константа; тип — массив элементларининг типи. Масалан,

```
Temper : array[1..31] of real;
Koef : array[0. .2] of integer;
Name : array[1..30] of string[25];
```

Массивларни эълон қилишда номланган константалардан фойдаланиш мақсадга мувофиқ ҳисобланади. Номланган константалар одатда константаларни эълон қилиш бўлимида эълон қилинади. Шундан кейин ундан оддий сонли ёки белгили константа сифатида фойдаланиш мумкин. Қуйидаги мисолда футбол бўйича чемпионатда қатнашадиган жамоаларнинг рўйхатини киритиш учун массив эълон қилиш масаласи кўрилган:

```
const
    NT = 18;// жамоалар сони
    SN = 25;// жамоаларнинг энг узун номи
Var komanda : array[1..NT] of string [SN];
```

Дастурда массив элементини кўрсатиш учун массивнинг номи ва квадрат қавслар ичида элементнинг номери (массивнинг индексини) кўрсатиш керак. Индекс сифатида константа ёки бутун типдаги ифодадан фойдаланиш мумкин. Масалан:

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

Агар массив локал бўлмаса, яъни модулнинг ўзгарувчилар бўлимида эълон қилинган бўлса, эълон қилиш билан бир вақтда уни инициализация қилиш мумкин, бошқача айтганда унинг элементларига бошланғич қийматлар бериш мумкин. Массивни эълон қилиш вақтида унга қиймат бериш қуйидагича кўринишда бўлади:

Ном : array [қуйи индекс .. юқори индекс] of тип = (рўйхат);

бу ерда рўйхат –массив элементларининг бир-биридан вергул билан ажратилган қийматлари. Масалан :

```
a: array[10] of integer = (0,0,0,0,0,0,0,0,0,0);
Komanda: array[1..5] of String[10]=('Пахтакор', 'Нефтчи', 'Навбаҳор', 'Машъал','Насаф');
```

Инициализация рўйхатидаги элементлар сони массивнинг ўлчамларига мос бўлиши керак. Акс ҳолда компилятор бу ҳақда

Number of elements differs from declaration
(элементлар сони эълонда кўрсатилганига мос эмас)

тарзидаги ахборотни экранга чиқаради.

Агар локал массивни инициализация қилишга уринилган бўлса, бу ҳатолик ҳақида ахборотнинг кўриниши ъуйидагича бўлади:

Cannot initialize local variables
(локал ўзгарувчи инициализация қилиниши мумкин эмас)

Локал массивларни фақат дастурнинг иши давомида инициализация қилиш мумкин. Масалан:

```
for i := 1 to 10 do a[i] := 0;
```

менюга

6.3. Массив элементларини киритиш ва чиқариш (*StringGrid* ва *Memo* компонеталари)

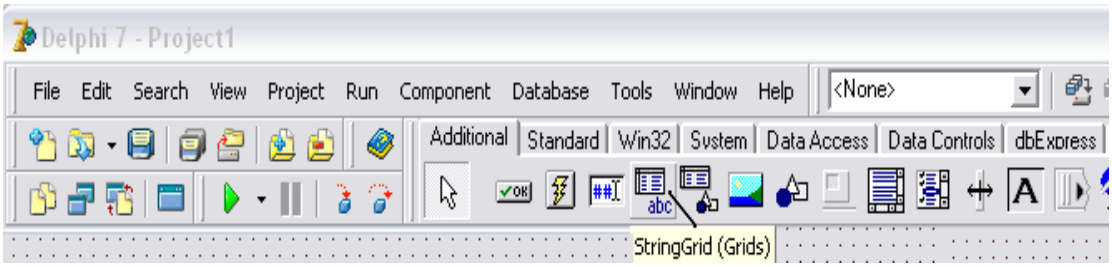
Массивлар билан ишлаганда, унинг элементлари ўртасида қуйидаги амалларни бажариш мумкин: массив элементларини киритиш ва чиқариш, энг катта ва кичик элементларини қидириш, массивдан берилган элементни излаш, массив элементларини тартиблаш ва х.к.

Массив элементларини киритиш деганда массив элементларига қийматни фойдаланувчи томонидан киритилиши ёки дастурнинг иши давомида қиймат берилиши назарда тутилади.

Бу масаланинг энг осон йўли хар бир элемент учун алоҳида киритиш майдонини ташкил қилишдан иборат. Аммо етарлича катта массивни киритишга тўғри келганда бу усул ярамайди. Фараз қилинг, битта форма устида юзлаб киритиш майдонлари жойлашсин.

Турган гапки, массив элементларини жадвалнинг сатри ёки устунига киритиш жуда ҳам қулай. Бунда ҳар бир элемент алоҳида ячейкага жойланади. Қуйида жадвал элементларини киритишнинг ***StringGrid*** ва ***Memo*** усулларини кўрамиз.

StringGrid компонентаси. Массив элементларини киритишда StringGrid компонентаси жуда ҳам қулай. StringGrid компонента-сининг нишони **Additional** қуроллар панелида жойлашган (6.1-расм).



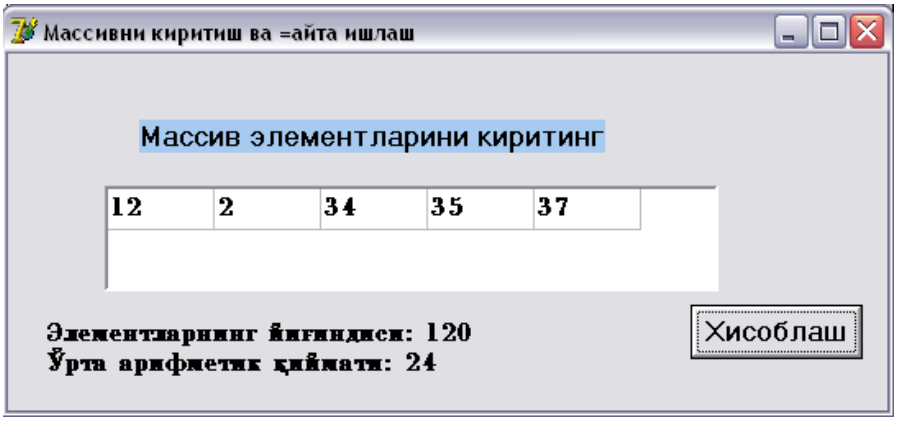
6.1-расм. StringGrid компонентаси

StringGrid компонентаси ячейкалари белгилар билан тўлдирилган жадвалдан иборат. 6.1-жадвалда ***StringGrid*** компонентасининг айрим ҳоссалари келтирилган.

StringGrid компонентасининг ҳусусиятлари 6.1-жадвал

Ҳусусияти	**мазмуни**
Name	Компонентанинг номи. Дастурда компонента ҳусусиятларига мурожаат қилиш учун фойдаланилади.
ColCount	Жадвалнинг устунлари сони
RowCount	Жадвалнинг сатрлари сони
Cells	Жадвалга мос келувчи икки ўлчовли массив. Col номерли устун ва row номерли сатр кесишган ячейкадаги элемент - cells [col, row].
FixedCols	чапдан фиксирланган устунлар сони. Бу устунлар бошқа рангда ажратиб кўрсатилади.
FixedRows	Юқоридан фиксирланган сатрлар сони. Бу сатрлар бошқа рангда ажратиб кўрсатилади.
Options . goEditing	Жадвал ячейкасини таҳрирлашга рухсатнома. True — таҳрирлаш мумкин, False — йўқ.
Options . goTab	<Tab> тугмасидан фойдаланишга рухсатнома. Курсорни кейинги ячейкага ўтказишда (True) – мумкин, (False)-таъқиқланади.
Options . GoAlways-ShowEditor	Компонентанинг таҳрирлаш режимида бўлиши аломати. Агар бу ҳусусиятнинг қиймати False бўлса, у ҳолда ячейкада курсор пайдо бўлиши учун матнни териш бошланиши, <F2> тугмасини босиш ёки сичқонча тугмасини чертиш лозим.
DefaultColWidth	Жадвал устунларнинг кенглиги
DefaultRowHeight	Жадвал сатрларининг баландлиги
GridLineWidth	Жадвал ячейкаларини ажратувчи чизиқ кенглиги
Left	Жадвал майдонининг чап чегарасидан то форманинг чап чегарасигача бўлган масофа
Top	Жадвал майдонининг юқори чегарасидан то форманинг юқори чегарасигача бўлган масофа
Height	Жадвалнинг кенглиги
Width	Жадвалнинг кенглиги
Font	Жадвалдаги матнлар учун шрифт
ParentFont	Формадан шрифт аломатларини ўзига олиш

StringGrid компонентасидан фойдаланишга намуна сифатида массив элементларининг ўрта арифметик қийматини ҳисоблаш дастурини кўрайлик. Дастурнинг диалог ойнаси 6.3-расмда келтирилган. ***StringGrid*** компонентаси массив элементларини киритиш учун, Label1 ва Label2 компоненталари изоҳлаш ва натижани кўрсатиш учун формага қўйилган. Button1 — ҳисоблаш жараёнини бошлайди.



6.3-расм. Массивни киритиш ва қайта ишлаш дастурининг диалог ойнаси

StringGrid компонентаси формага бошқа объектлар каби қўшилади. Шундан кейин уни 6.2-жадвалга мувофиқ созлаш лозим. Height ва width ҳоссаларнинг қийматларини сичқонча ёрдамида компонентанинг ўлчамлари сатр ўлчамига тенг бўладиган қилиб ўрнатилади. Дастурнинг матни 6.2-листингда келтирилган.

StringGrid1 компонентасининг ҳусусиятлари 6.2-жадвал

ҳусусияти	қийматлари
ColCount	5
FixedCols	0
RowCount	1
DefaultRowHeight	24
Height	24
DefaultColWidth	64
Width	328
Options . goEditing	True
Options . AlwaysShowEditing	True
Options .goTabs	True

Листинг 6.2. Бутун сонларни киритиш ва қайта ишлаш

```
unit getar;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, Grids, StdCtrls;
type
TForm1 = class(TForm)
Label1: TLabel;
StringGridl: TStringGrid;
Button1: TButton;
Label2: TLabel;
procedure Button1Click(Sender: TObject); private
{ Private declarations }
public
{ Public declarations }
end;
var
Form1: Tform1 ;
implementation
{$R *.dfm}
procedure Tform1.ButtonlClick(Sender: TObject);
var a : array[1..5] of integer;// массив
    summ: integer;// элементларнинг йиғиндиси
    sr: real;// ўрта арифметик қиймат
    i: integer;// индекс
begin
```

```
//массивни киритиш
//агар ячейка бўш бўлса, массивнинг ундаги элементини нол деймиз
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 := 'Элементларнинг йиғиндиси: ' + IntToStr(summ)
+ #13 + 'Ўрта арифметик қиймати: ' + FloatToStr(sr);
end;
end.
```

Дастурни бир неча марта синагандан сўнг, массив элементларини киритишни ўзгартиришга истак пайдо бўлади. Масалан, ячейкага матн киритилганидан сўнг <Enter> тугмаси босилганида курсор автоматик равишда кеийнги ячейкага ўтсин. Буни ***onKeyPress*** ходисаларни қайта ишлаш процедураси ёрдамида амалга ошириш мумкин. Шу процедурага киритилаётган маълумотларни назорат қилишни ҳам топшириш мумкин.

OnKeyPress ходисаларни қайта ишлаш процедурасининг матни 6.3-листингда келтирилган. ***Col*** нинг ҳусусиятига алоҳида эътибор беринг. Унинг қиймати дастур давомида курсор турган ячейка номерига тенг. Бу ҳусусиятдан курсорни керакли ячейкага ўтказиш учун ҳам фойдаланиш мумкин. Шуни ёдда тутиш керакки, устунлар ва сатрлар нолдан бошлаб номерланади.

6.3-листинг. OnKeyPress ходисаларни қайта ишлаш процедураси

```
procedure TForm1.StringGridlKeyPress(Sender: TObject;
var Key: Char);
begin
case Key of
#8,'0'..'9' : ;//рақамлар ва  <Backspace> клавишаси
#13://  <Enter> клавишаси
if StringGridl.Col < StringGrid1.ColCount — 1
then StringGridl.Col := StringGridl.Col + 1;
else key := Chr(0);//қолган белгилар таъқиқланган
end;
end;
```

Агар каср сонларни киритишга тўғри келса, (a: array [1. .5] of real), у ҳолда OnKeyPress ходисаларни қайта ишлаш процедураси бир оз мураккаблашади. Чунки, рақамлардан ташқари ажратувчи белги (вергул ёки нуқта) ҳамда минус белгиларини ҳам ҳисобга олиш керак бўлади. Бу ерда бир оз айёрлик қилиш лозим, яъни нотўғри киритилган ажратувчи белгини тўғрисига алмаштириидлади. Windows нинг жорий созланиши учун қайси белги ажратувчи эканлигини Decimalseparator глобал ўзгарувчисига мурожаат қилиб аниқлаш мумкин.

6.4-листингда ҳақиқий сонлар массивини киритиш ва қайта ишлаш дастурининг матни келтирилган. ***OnKeyPress*** ходисаларни қайта ишлаш процедураси фақат мумкин бўлган белгиларни киритишни таъминлайди.

6.4-Листинг. Ҳақиқий сонлар массивини киритиш ва қайта ишлаш

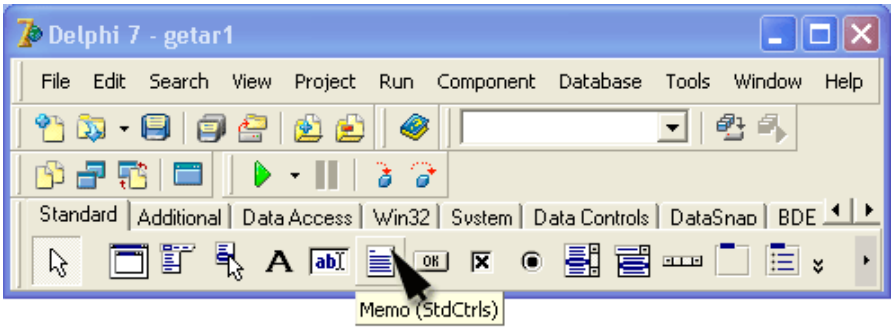
```
unit. getar_1;
 interface
 uses Windows, Messages, SysUtils, Variants, Classes,
 Graphics, Controls, Forms, Dialogs, Grids, StdCtrls;
 type
 Tform1 = class(TForm)
 Label1: TLabel;
 StringGrid1: TStringGrid;
```

```
Button1: TButton;
Label2: TLabel;
procedure Button1ClicktSender: TObject);
procedure StringGrid1KeyPress(Sender: TObject; var Key: Char);
private
{ Private declarations }
public
{ Public declarations }
end;
var
Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
var
a : array[1..5] of real;// массив
suram: real;// элементлар йиғиндиси
sr: real;// ўрта арифметик қиймат
i: integer;// индекс
begin
// массивни киритиш
// Агар ячейка бўш бўлса, унга мос элементни нол деб ҳисоблаймиз
 for i := 1 to 5 do
if Length(StringGrid1.Cells[i-l,0])<>0
then a[i] := StrToFloat(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: ' + FloatToStr(summ)
+ #13 + 'Ўрта арифметиги: ' + FloatToStr(sr);
end;

// Ячейкага фақат мумкин бўлган белги киритилишини таъминлаш
procedure TForm1.StringGrid1KeyPress(Sender: TObject; var Key: Char);
begin
case Key of
#8,'0'..'9' : ;// рақамлар ва  <Backspace>  тугмаси
#13:// <Enter> клавишаси
if StringGrid1.Col < StringGrid1.ColCount - 1
then StringGrid1.Col := StringGrid1.Col + 1;
'.',',': begin// бутун ва каср қисмни ажратувчи белги
if Key <> DecimalSeparator then
Key:=DecimalSeparator;//ажратувчи белгини тўғрисига алмаштириш
if Pos(StringGrid1.Cells[StringGridl.Col,0],DecimalSeparator) <> 0
then Key := Chr(O);// иккинчи ажратувчи белгини таъқиқлаш
 end;
' -' :// минусни фақат биринчи белги қилиб киритиш мумкин
// яъни фақат ячейка бўш бўлганди
if Length(StringGrid1.Cells[StringGrid1.Col, 0]) <>0 then
Key := Chr(0) ;
else// қолган белгилар таъқиқланади
key := Chr(0);
end;
end;
end.
```

менюга

6.4. Memo компонентасидан фойдаланиш

Memo компонентаси етарлича катта бўлган сондаги сатрларни киритишга имкон беради. Шунинг учун, **Memo** компонентасидан белгили массивларни киритишда фойдаланш мумкин. Унинг нишони **Standard** куроллар панелида жойлашган. (6.4-расм.)**Memo** компонентаси формага бошқа компонентлар каби қўйилиши мумкин.



6.4-расм. Memo компонентаси

Memo майдонига матнни киритиш учун **Object Inspector** ойнасидаги **Lines** ҳусусиятининг **[tstrings...]** қиймати чертилади. Натижада **Memo** майдонига матн киритиш муҳаррири ишга тушади. **Memo** компонетасидан фойдаланилиб массив элементларини киритишда массивнинг ҳар бир элементини алоҳида сатрга киритиш ва ҳар бир элемент киритилгандан сўнг **<Enter>** тугмасини босиш лозим.

6.3-жадвалда **Memo** компонентасининг айрим ҳусусиятларини келтирамиз.

Memo компонентаси**нинг ҳусусиятлари.** **6.3-жадвал.**

Ҳусусияти	Мазмуни
Name	Компонентанинг номи. Компонента ҳусусиятларига мурожаат қилишда фойдаланилади.
Text	Memo майдонидаги матн. Матнни битта деб қаралади.
Lines	Memo майдонидаги матн. Матнни сатрлар кетма-кетлиги сифатида қаралади. Сартга унинг номери бўйича мурожаат қилинади.
Lines .Count	Memo майдонидаги сатрлар сони
Left	Майдоннинг чап чегарасидан то форманинг чап чегарасигача бўлган масофа
Top	Майдоннинг ўнг чегарасидан то форманинг ўнг чегарасигача бўлган масофа
Height	Майдоннинг баландлиги
Width	Майдоннинг кенглиги
Font	Киритилаётган матн учун шрифт
ParentFont	Шрифт ҳусусиятларини формадан олиш

Memo майдонида турган матннинг бирор сатрига мурожаат қилиш **Lines** ҳусусияти ёрдамида, квадрат қавслар ичида сатр номерини кўрсатиб амалга оширилади.

Қуйидаги мисол Memo компонентасидан фойдаланишни намойиш этади. Унинг дастури 6.5-листингда келтирилган.

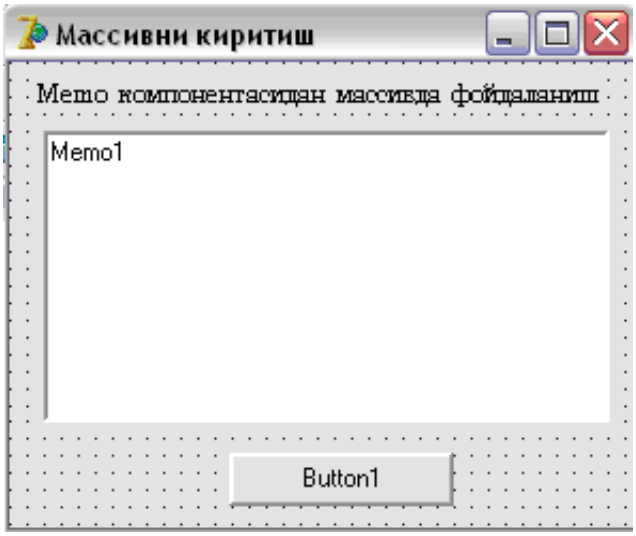
Memo компонентасидан белгили массивни киритишнинг процедурасининг асосий цикли қуйидагича:

```
for i := l to SIZE do
a [ i ] := Memol.Lines[i];
```

бу ерда **SIZE** — массив ўлчамини кўрсатувчи номланган константа; **a** - массив; **Memo1** — Memo-компонентанинг номи; **Lines** —Memo компонентасининг ҳусусияти бўлиб, ҳар бир элементи Memo

майдонидаги матннинг битта сатридан иборат бўлган массив.

Дастурнинг формаси 6.5-расмда келтирилган. Унда Memo майдонидан ташқари, Memo майдонида массив элементларини киритишни бошлаш учун Button1 тугмаси жойлашган.



6.5-расм. Массивни киритиш иловасининг диалог ойнаси

6.5-листинг. Memo компонентасидан массив элементларини киритиш.

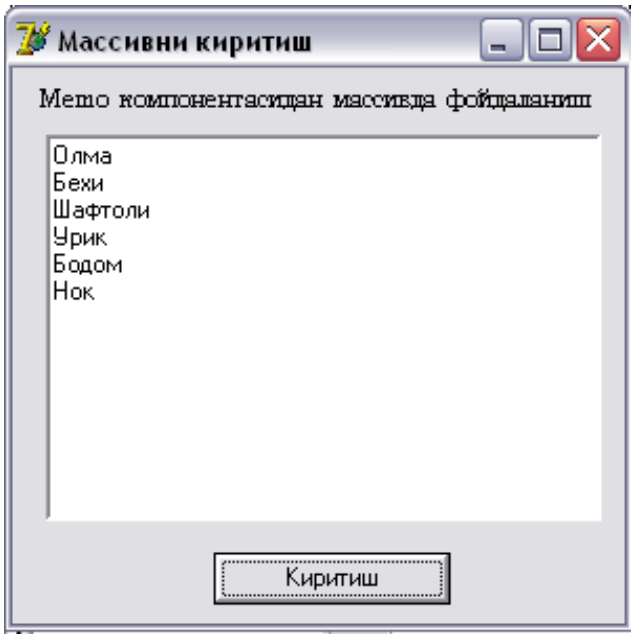
```
unit Unit1;
 interface
 uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,   Dialogs, StdCtrls;
 type
  TForm1 = class(TForm)
    Memo1: TMemo;
    Label1: TLabel;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
 var
  Form1: TForm1;
 implementation
 {$R *.dfm}

 procedure TForm1.Button1Click(Sender: TObject);
 const
 SIZE = 5;// массив ўлчами
 var a : array[1..SIZE] of string[30];//массив
    n: integer;//Memo майдонига киритилган сатрлар сони
    I : integer;//массив элементининг индекси
    st:string;
 begin
 n := Memo1.Lines.Count;
 if n = 0 then begin
 ShowMessage('Бошланғич маълумотлар нотўғри!');
 Exit;// ходисаларни қайта ишлаш процедурасидан чиқиш
 end;
// Memo майдонига матн киритилган
 if n > SIZE then begin
 ShowMessage('Сатрлар сони массив ўлчамидан катта.');
 n := SIZE;//фақат дастлабки SIZE та сатрни оламиз
 end;
```

```
for i := 1 to n do
a[i] := Form1.Memo1.Lines[i-1];
// Memo майдони сатрлари нолдан бошлаб номерланган
// массивни маълумотлар ойнасига чиқариш
if n > 0 then begin
st := 'Киритилган массив:' + #13;
for i := 1 to n do
st := st + IntToStr(i) + ' '+ a[i] + #13;
ShowMessage(st);
end;
end;
end.
```

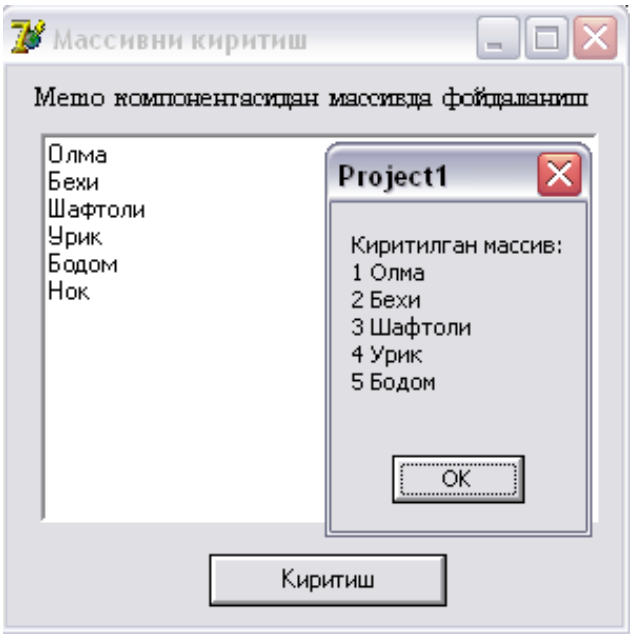
Дастурни ишга тушириш

Бу дастурда асосий ишни TForm1.Button1Click процедураси бажаради. У дастлаб Memo1 майдонида матннинг бор-йўқлигини текширади. Агар матн бўлса, (бу ҳолда Lines.Count ҳусусиятининг қиймати нолдан катта), процедура Memo майдонига киритилган сатрлар сони ва массивнинг ўлчамини таққослайди. Ага бу миқдор массив ўлчамидан катта бўлса, у ҳолда дастур автоматик тарзда ***n*** нинг қийматини ўзгартиради ва дастлабки SIZE та сатрни олади ҳалос.

6.6-расмда **Массивни киритиш** дастурининг диалог ойнаси келтирилган. **Киритиш** тугмаси чертилганидан сўнг, экранда 6.7-расмдаги ойна пайдо бўлади. Ундаги массив ўз элементларини Memo майдонидан олган.



6.6-расм. Массив киритиш иловасининг диалог ойнаси



6.7-расм. Memo майдонидан киритилган массив

менюга

6.5. Массивнинг энг катта (энг кичик) элементини топиш

Массивнинг энг катта (энг кичик) қийматини топиш масаласини бутун сонлар мисолида кўриб чиқамиз.

Массивнинг энг катта (энг кичик) қийматини топиш алгоритми жуда ҳам содда: Дастлаб массивнинг биринчи элементини энг катта (энг кичик) деб фараз қиламиз. Сўнгра массивнинг қолган элементлари энг катта (энг кичик) элемент билан таққосланади. Агар текширилаётган элемент энг катта элементдан катта (энг кичик элементдан кичик) бўлса, шу элемент энг катта (энг кичик) бўлиб қолади. Текшириш қолган элементлар учун, то массивнинг охирги элементигача давом эттирилади.

Бу масала дастурининг диалог ойнаси эхтиёжга қараб созланган stringGrid1 компонентаси,

маълумотларни изоҳлаш ва натижани чиқариш учун Label1 ва Label2 компонеталари ҳамда текширишни бошлаш учун Button1 бууйруқли тугмасини ўз ичига олади. 6.4-жадвалда stringGrid1 компонентаси ҳусусиятининг қийматлари келтирилган.

6.6-листингдаги дастурнинг матни массивнинг энг кичик элементини топиш учун мўлжалланган.

StringGrid1 компонентаси ҳусусиятининг қийматлари 6.4-жадвал

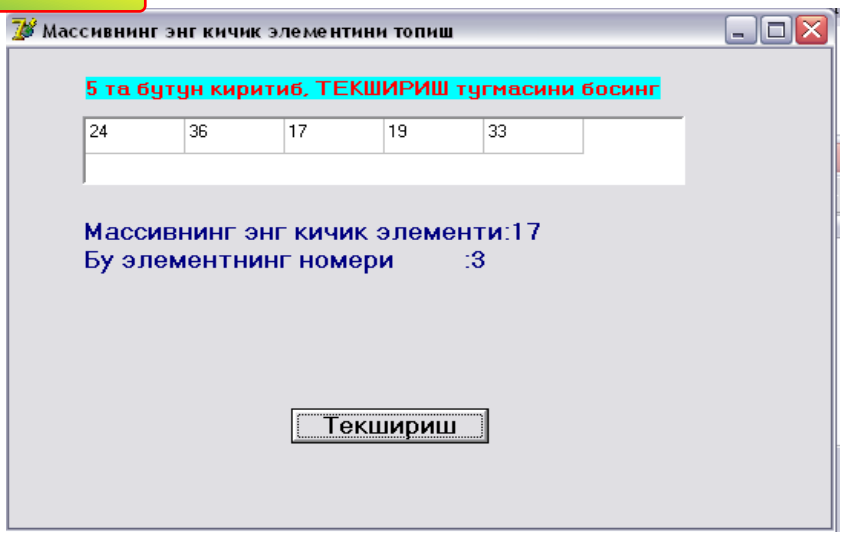
Ҳусусияти	Қиймати
ColCount	5
FixedCols	0
RowCount	1
DefaultRowHeight	24
Height	24
DefaultColWidth	64
Width	328
Options . goEditing	True
Options . AlwaysShowEditing	True
Options .goTabs	True

6.6-листинг. Массивнинг энг кичик элементини топиш

```
unit lookmin_;
interface
Windows, Messages, SysUtils, Classes, Graphics,
Controls, Forms, Dialogs, StdCtrls, Grids;
type
TForm1 = class(TForm)
Label1: TLabel;
Button1: TButton;
Label2: TLabel;
StringGridl: TStringGrid;
procedure ButtonlClick(Sender: TObject);
 private { Private declarations )
public   { Public declarations }
end;
var
Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.ButtonlClick(Sender: TObject);
const
SIZE = 5;
var
a : array[1..SIZE]of integer;//бутун сонлар массиви
min : integer;//массивнинг энг кичик элементининг номери
i : integer;//энг кичиги билан солиштирилаётган элемент номери
begin
//массивни киритиш
for I := 1 to SIZE do
a[i] := StrToInt(StringGrid1.Cells[i-1,0]);
//Энг кичик элементни  излаш
min := 1;//биринчи элемент энг кичик бўлсин
for i := 2 to SIZE do
if a[i]< a[min]then min := i;
// натижани чиқариш
```

```
label2.caption := 'Массивнинг энг кичик элементи:' +
IntToStr(a[min]) +#13 + 'Бу элементнинг номери        :'+ IntToStr(min);
end;
end.
```

Дастурни ишга тушириш



6.8-расм. Массивнинг энг кичик элементини топиш дастурининг диалог ойнаси

менюга

6.6. Массивдан маълумотларни иккига бўлиш усули билан қидириш

Кўпинча массив элементлари орасидан бирор бир маълумотни қидириш билан боғлиқ масалаларни ечишга тўғри келади. Амалиётда бу қидиришни бирор бир усул билан тартибланган массивда олиб борилади. Масалан, алфавит бўйича тартибланган фамилиялар массивидан "Отаханов" фамилияси, кутубхонадаги алфавит тартибида тартиблаган китоблар массивидан "Шум бола" романини қидириб топиш ва х.к. Тартибланган масивдан бирор маълумотни қидириш масаласи учун энг яхши усуллардан бири - бу тенг иккига бўлиш усулидир.

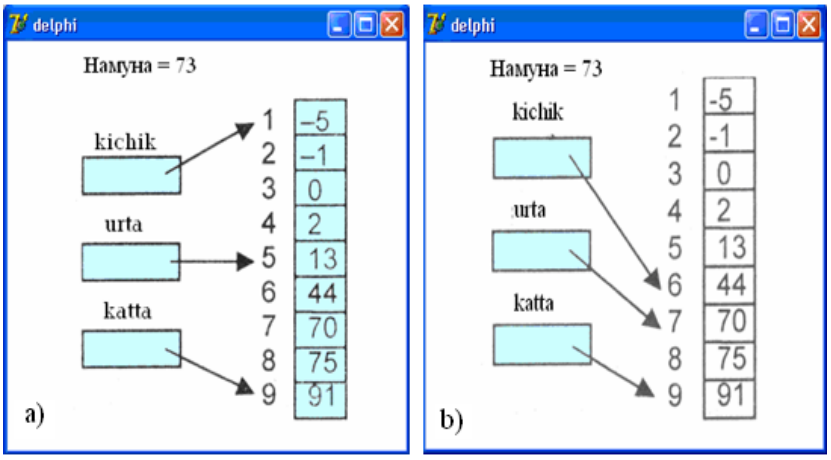
Фараз қилайлик, массив ўсиш тартибида тартибланган бўлсин. Шу массивда берилган сон (намуна) бор ёки йўқлигини аниқлаш талаб қилинади.

Бу усулнинг ғояси намуна изланаётган оралиқни тенг иккига бўлишга асосланади. Текширишни бошлашдан аввал, биз бу массивда изланган намуна элемент мавжуд эмас деб фараз қиламиз. Берилган оралиқ чап чегарасининг индекси ***kichik***, ўнг чегарасининг индекси ***katta,*** уларнинг ўртасида жойлашган элементнинг индекси ***urta*** бўлсин. Дастлаб масивнинг ўртадаги элементининг индекси ***urta*** топилади. Сўнг намуна массивнинг ***urta*** индексли элементи билан таққосланади. Агар улар тенг бўлса, масаланинг ечими топилди деб жараённи тўхтатилади. Акс ҳолда намуна ***urta*** индексли элементга нисбатан қайси оралиқда жойлашганлиги аниқланади. Унга кўра биз намуна изланаётган оралиқнинг нокерак қисмини ташлаб юбориб, керакли қисмини олиб қоламиз. Шу оралиққа боғлиқ равишда ***katta*** ёки ***kichik*** ўзгарувчилардан бири ***urta*** қийматини олади. ***katta*** ёки ***kichik*** нинг янги қийматларини ҳисобга олиб, ***katta*** ёки ***kichik*** нинг қийматидан ***urta*** нинг янги қийматини ҳисобланади ва юқоридаги жараён яна такрорланади. Бу иш токи намунага тенг бўлган элемент топилгунча ёки текширилаётган оралиқ учун ***katta - kichik*** =1 бўлиб қолгунча давом эттирилади. Сўнги ҳолда массивнинг ***katta*** ёки ***kichik*** индексли элементларининг намунага тенглиги текширилади. ***Urta*** нинг қийматини аниқлашда ***katta*** + ***kichik*** миқдорнинг жуфт ёки тоқлигини текширамиз. Агар у жуфт бўлса, ***urta=(katta+kichik)/2***, акс ҳолда ***urta=[katta+kichik]/2+1*** формула билан топилади. Бу ерда [x]-бутун сонни англатади.

Фараз қилайлик, бу массивнинг номи B ва унда n та элемент мавжуд бўлсин. Тенг иккига бўлиш усули қуйидаги алгоритм ёрдамида амалга оширилади.

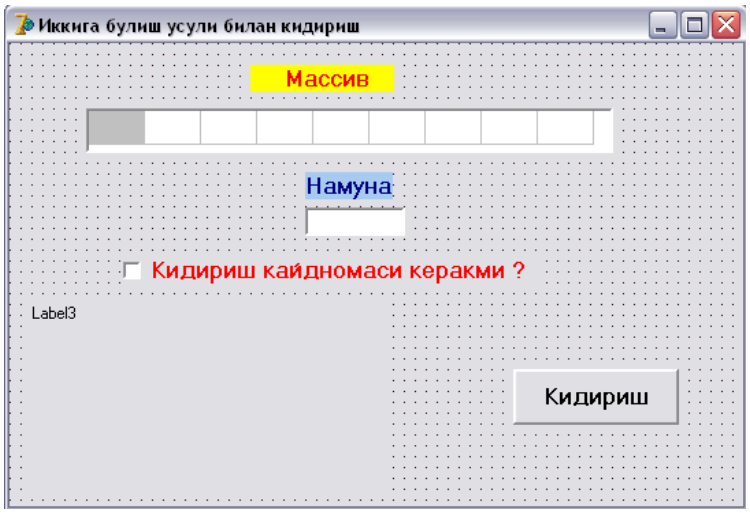
1. Бошлансин
2. Киритилсин ***Намуна***
3. ***y := 'йўқ'***
4. ***kichik := 1; katta := n***

5. агар ***katta-kichik =1*** ёки ***y ≠ 'йўқ'*** бўлса 11 га ўтилсин
6. ***a :=kichik +katta***
7. агар ***a*** жуфт бўлса, ***urta :=(kichik +katta)/2***, акс ҳолда ***urta :=[kichik +katta]/2 +1***
8. агар ***Намуна=B(urta)*** бўлса 12 га ўтилсин
9. агар ***намуна>B(urta)*** бўлса ***kichik :=urta***, акс ҳолда ***katta :=urta***
10. 5 га ўтилсин
11. Агар B(kichik) = Намуна ***ёки*** B(katta)= Намуна бўлса ***y:='Ха'***
12. Чиқарилсин ***y***
13. Тамом



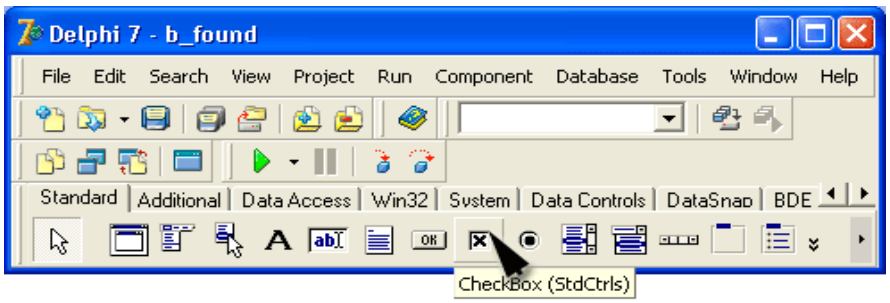
6.10-расм. Массивни иккига бўлиш усулида ўрта элементни танлаш

Массивни иккига бўлиш усули билан қидириш дастурининг диалог ойнаси 6.11-расмда берилган. Формада Label3 майдони қидириш натижаси ҳамда қидириш қайдномасини эълон қилиш учун мўлжалланган. Формадаги **қайдномани чиқарилсин** байроқчаси ўрнатилган бўлса Қайднома, ***kichik, urta*** ва ***katta*** ўзгарувчиларнинг дастур давомида қабул қилган қийматлари экранга чиқарилади. Бу маълумот иккига бўлиш усулининг моҳиятини тушунишга ёрдам беради.



6.11-расм. Иккига бўлиш усули билан қидиришнинг диалог ойнаси

Илова формасида биз янги компонента ***CheckBox*** – байроқчадан фойдаландик. Унинг нишони **Standard** куроллар панелида жойлашган. (6.12-расм.). ***CheckBox*** компонентасини формага бошқа объектлар каби ўрнатиш мумкин. ***CheckBox*** компонентасининг айрим ҳусусиятлар и 6.5-жадвалда берилган.



6.12-расм. CheckBox компонентаси

CheckBox компонентасининг ҳусусиятлари 6.5-жадвал

Ҳусусияти	Мазмуни
Name	Компонентанинг номи. Компонента ҳусусиятларига мурожаат қилиш учун ишлатилади.
Caption	Байроқчанинг маъносини изоҳловчи матн
Checked	Байроқчанинг вазияти: агар ўрнатилган бўлса (√) у ҳолда checked = True; акс ҳолда Checked=False
State	Байроқчанинг ҳолати. Checked ҳусусиятидан фарқли равишда, ўрнатилган, туширилган ҳамда оралиқ ҳолатни ажратиб олишга имкон беради. Байроқчанинг ҳолати: cbChecked (ўрнатилган); cbGrayed (кул-ранг, ноаниқ ҳолат); cbUnChecked (туширилган)
AllowGrayed	Байроқча оралиқ ҳолатда бўла оладими? Агар AllowGrayed + True, бўлса, байроқча оралиқ ҳолатда бўла олади, акс ҳолда – йўқ.
Left	Байроқчанинг чап чегарасидан форманинг чап чегарасигача бўлган масофа
Top	Байроқчанинг юқори чегарасидан форманинг юқори чегарасигача бўлган масофа
Height	Изоҳловчи матннинг баландлиги
Width	Изоҳловчи матннинг кенглиги
Font	Изоҳловчи матн учун шрифт
ParentFont	Формадан шрифт аломатларини ўзига олиш

Формага CheckBox компонентаси ўрнатилганидан кейин, унинг ҳусусиятларини 6.6-жадвалга мувофиқ ўзгартирилади.

CheckBox1 компонентаси ҳусусиятларининг қиймати 6.6-жадвал

Ҳусусияти	қиймати
Caption	Қидириш қайдномаси керакми?
Checked	True

6.8- листингда **ҚИДИРИШ** тугмаси учун Onclick ходисаларни қайта ишлаш процедурасининг матни келтирилган. Бу процедура массив элементлари ҳамда намуна сонни киритади, сўнгра иккига бўлиш усули билан намунага тенг бўлган элементнинг массивда бор ёки йўқлигини аниқлайди.

StringGrid1 ва Edit1 компоненталари учун OnKeyPress ходисаларни қайта ишлаш процедурасига алоҳида эътибор беринг. Улардан биринчиси курсорни кейинги ячейкага ёки Edit1 майдонига ўтказса, иккинчиси - **ҚИДИРИШ** тугмасини активлаштиради. Қидиришни <Enter> тугмаси билан ҳам бошлаш мумкин.

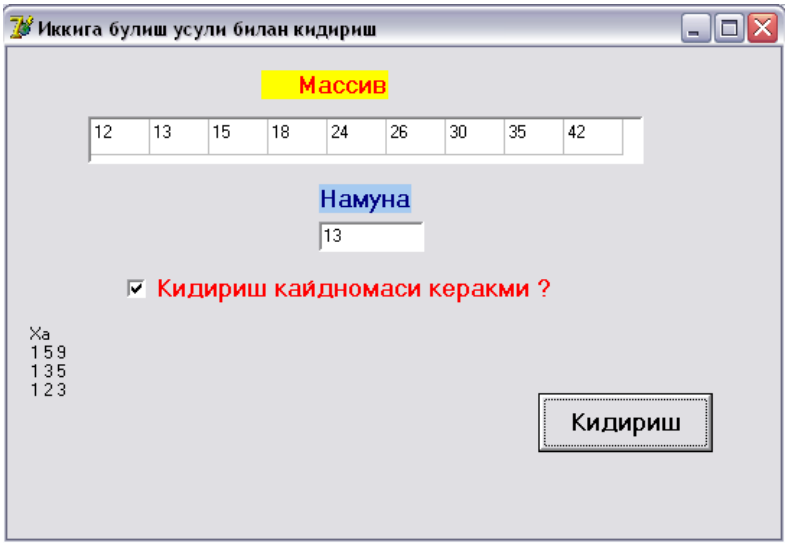
Листинг 5.8. Массивдан иккига бўлиш усули билан берилган маълумотни қидириш

```
interface
uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,   Dialogs, StdCtrls, Grids;
type
  TForm1 = class(TForm)
    StringGrid1: TStringGrid;
    Label1: TLabel;
    Label2: TLabel;
    Edit1: TEdit;
    CheckBox1: TCheckBox;
    Label3: TLabel;
    Button1: TButton;
```

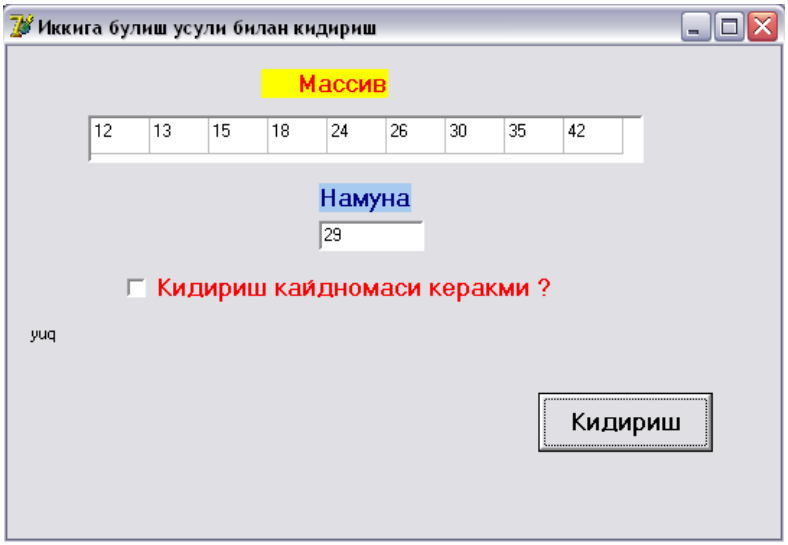
```
    procedure Button1Click(Sender: TObject);
    procedure StringGrid1KeyPress(Sender: TObject; var Key: Char);
    procedure Edit1KeyPress(Sender: TObject; var Key: Char);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
      const n = 9;
   var kichik, katta, urta,a,i:integer;
      b: array[1..9] of integer;
      y,y1,y2 : string;
      namuna : integer;
begin
      namuna := strtoint(edit1.Text);
      for i := 1 to n do
      b[i] := StrToInt(StringGrid1.Cells[i-1,0] ) ;
      y  := 'yuq'; kichik := 1; katta := n ;
      while (katta-kichik>1)and(y = 'yuq') do begin
        a := katta + kichik;
        if a mod 2 = 0 then urta := trunc(a/2)
                       else urta := trunc(a/2) + 1;
        y1 := y1 + inttostr(kichik) + ' ' + inttostr(urta)
              + ' ' + inttostr(katta) + #13;
        if namuna = b[urta] then y := 'Xa'
        else if namuna > b[urta] then kichik := urta
                                 else katta := urta;
       end;
      if (namuna = b[kichik]) or (namuna = b[katta]) then y := 'Xa';
      if CheckBox1.Checked then
      Label3.caption := y + #13 + y1 else Label3.caption := y ;
end;
// клваишани StringGrid ячейкасида босилганда
procedure TForm1.StringGrid1KeyPress(Sender: TObject; var Key: Char);
begin
if Key = #13 then // <Enter> тугмаси босилса
if StringGrid1.Col < StringGrid1.ColCount - 1
then// êóðñîðни массивнинг кейинги ячейкасига ўтказиш
StringGrid1.Col := StringGrid1.Col + 1
else//  êóðñîð Editl, намуна майдонларида
Edit1.SetFocus;
end;
// клавиша Editl майдонида босилса
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
if Key = #13// <Enter> тугмаси босилди
then// буйруқли тугмани активлаштир
Button1.SetFocus;
end; end.
```

Дастурни ишга тушириш

Қуйида **Иккига бўлиш усули билан қидириш** дастурининг иши давомида ҳосил бўлиши мумкин бўлган диалог ойнасининг қайдномали ва қайдномасиз кўринишларига мисоллар келтирамиз. (6.13а-расм ва 6.13б –расмлар)



6.13а-расм. Диалог ойнасининг қайдномали кўриниши



6.13б-расм. Диалог ойнасининг қайдномасиз кўриниши

менюга

6.7. Массив элементларини тартиблаш

Массив элементларини тартиблаш деганда унинг элементларини маълум бир тартибда қайтадан жойлаштириш тушунилади. Масалан, бутун сонларнинг массивни берилган бўлиб, унинг элементларини ўсиш тартибида тартибланса, элементлари учун

$$a[1] < a[2] < .. .< a[n]$$

шарти ўринли бўлади. Бу ерда n- массивнинг юқори индекси.

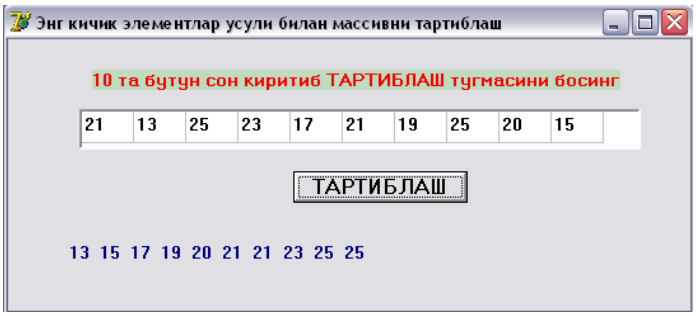
Массив элементларини тартиблаш масаласи информацион системаларда одатда тайёргарлик босиқичи сифатида қўлланади. Масалан, тартибланган массив элементлари орасидан бирор маълумотни излаш тартибланмаган тўпламга нисбатан тезроқ бажарилади. (Иккига бўлиш усули билан қидириш масаласига қаранг.)

Массив элементларини тартиблашнинг усуллари ҳилма-ҳил: энг кичик элементлари орқали тартиблаш, кўпиксимон тартиблаш усули, тартибланган жадвалга янги элементни қўшиш усули, икки тартибланган массивларни қўшиш усули ва х.к. Бу усуллардан бирортасини бошқасига қараганда устун қўйиб бўлмайди. Усуллардан бири элементларнинг маълум бир тартиб билан жойлашганда афзал бўлса, бошқаси юлшқача тартиб учун афзал ҳисобланади.

Энг кичик элементлари орқали тартиблаш усули энг содда кўп қўлланадиган усуллардан бири ҳисобланади. Фараз қилайлик, N та элементли массив элементларини ўсиш тартибида тартиблаш талаб қилинган бўлсин. Унинг амалга ошириш ғояси қуйидагича. Дастлаб массивнинг энг кичик элементи топилади ва биринчи элемент билан ўрин алмашади. Демак, 1-элемент тартибланди. Энди қолган элементлар орасидан энг кичиги топилади. У иккинчи элемент билан ўрин алмашади. Натижада иккинчи элемент ҳам тартибланди. Бу жараён дастлабки N-1 та элемент учун бажарилади. Натижада массивнинг ҳамма элементлари тартибланиб қолади. Бу ғояга асосланиб, қўйилган масаланинг ечиш алгоритмини қуйидагича қуриш мумкин:

1. Бошлансин
2. Киритилсин A массив
3. Киритилсин n;
4. k := 1;
5. min := a(k); ind := k
6. l := k+1
7. агар a(l)<min бўлса, у ҳолда min := l; ind := l
8. l := l + 1
9. агар l ≤ n бўлса 7 га ўтилсин
10. c := a(ind) ; a(ind) := a(k) ; a(l) := c;
11. Чиқарилсин a(k)
12. k := k + 1
13. агар k<=n-1 бўлса, 5 га ўтилсин
14. Ишни тугатилсин.

Қуйида шу алгоритмга мос келадиган дастур матнини келтирамиз. Бу дастурнинг диалог ойнаси 6.14-расмда кўрсатилган.



6.14-расм. Массивни тартиблаш дастурининг диалог ойнаси

Матни 6.9-листингда берилган дастур ТАРТИБЛАШ (Button1) тугмасини босилиши билан ишга тушади. Массивнинг элементлари StringGrid1 майдонидан олинади. Массивнинг навбатдаги энг кичик элементи топилганидан сўнг, уни Label1 майдонига чиқариш учун тайёргарлик қилинади.

6.9-листинг. Массивни оддий усул билан тартиблаш

```
procedure TForm1.Button1Click(Sender: TObject);
   const n = 10;// массивнинг ўлчами
      var i,k,l: word;
       ind: word;// к-чи ҳаддан бошлаб энг кичик элементнинг индекси
       c: integer;// элементлар ўрин алмашганда ёрдам беради
       min:integer;// кғҳчи ҳаддан бошлаб энг кичик элемент
       a:array[1..n] of integer;
       s: string;//тартибланган жадвални йиғиш учун
begin
  s := '';
  for i := 1 to n do
  a[i] := strtoint(stringgrid1.Cells[i-1,0]);
  for k := 1 to n-1 do begin
    min := a[k]; ind := k;
   //k-чи хаддан бошлаб, энг кичик элемент топилмоқда
    for l := k+1 to n do
    if min>a[l] then begin min := a[l]; ind := l; end;
   // энг кичик элемент ва тартибланаётган элемент
```

```
    // ўринлари алмаштирилмоқда
    c := a[ind]; a[ind] := a[k]; a[k] := c;
   //натижани чиқариш учун йиғиб борилмоқда
    s := s + inttostr(a[k]) + '  ';
  end;
 //охирги элемент натижага чиқариш учун олинмоқда
  s := s + inttostr(a[n]);
  label1.Caption := s;
end;
```

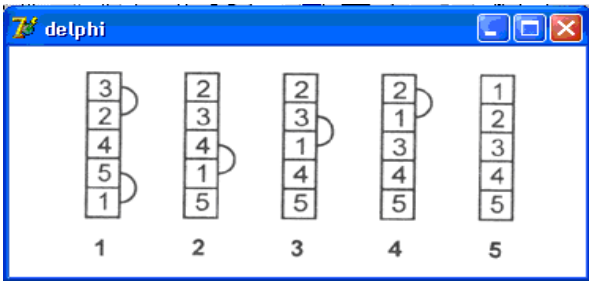
Дастурни ишга тушириш

Ушбу дастурнинг диалог ойнаси 6.14-рсмда келтирилган.

Тартиблашнинг кўпиксимон усули. Биз усулни сатрли катталиклар учун изоҳлаймиз. Фараз қилайлик, бизга N-фамилия қатнашган рўйхат берилган бўлсин. Шу рўйхатдаги фамилияларни алфавит тартибида тартиблаш талаб қилинган бўлсин.

Сатрли маълумотларни ҳам сонли маълумотлар каби таққослаш мумкинлиги ҳақида биз юқорида фикр юритган эдик. (4.1-п.га қаранг)

Тартиблашнинг кўпиксимон усули икки қўшни элементларни таққослаш ётади. Агар қўшни элементлардан чап томондагиси ўнг томондагисидан катта бўлса, уларнинг ўринлари ўзаро алмаштирилади. Текшириш яна бошидан бошланади. Натижада массивнинг кичик элементлари массивнинг бошига қараб сурилади (кўпикка ўхшаб кўтарилади), катталари эса охирига қараб сурилиб боради (чўкади). Бу жараён массивдаги (***элементлар сони-1)*** марта такрорланади. 6.15-расмда 1 рақами билан массивнинг 1-ўтишдан кейинги, 2 рақами билан 2-ўтишдаги ва х.к. ўтишидан кейинги холати тасвирланган.



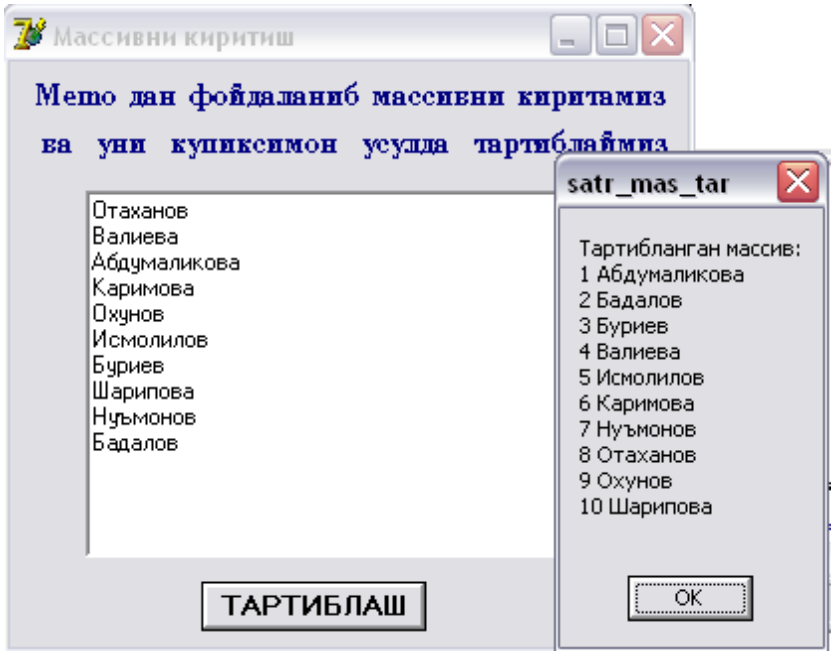
6.15-расм. Массивнинг кўпиксимон тартиблаш жараёни

Бу усул учун ёзилган дастур матни 6.10-листингда, диалог ойнаси 6.16-расмда келтирилган.

6.10-листинг. Кўпиксимон усул билан массивларни тартиблаш

```
procedure TForm1.Button1Click(Sender: TObject);
const n = 10;//массивнинг ўлчами
var a : array[1..n]of string[30];//массив
   m : integer;// Memo майдонидаги сатрлар сони
   I :  integer;//массив элементининг индекси
   St : string;
   k : string;/ўрин алмаштиришда қатнашадиган ёрдамчи ўзгарувчи
begin
m := Memo1.Lines.Count;
if m = 0 then begin
ShowMessage('Бошланғич маълумотлар киритилмаган!');
Exit;//ҳодисаларни қайта ишлаш процедурасидан чиқиш
end;
// Memo майдонига матн киритилган
if m > n then begin
ShowMessage('Сатрлар сони массив ўлчамидан катта ');
```

```
m := n;// Фақат дастлабки N та сатрни киритамиз
end;
for i := 1 to n do
a[i] := Form1.Memo1.Lines[i-1];
// массив элементлари тартибланмоқда
 i := 1;
 while I <= n-1 do begin
if a[i] > a[i+1] then
// Бу элементлар ўрин алмашмоқда
begin k := a[i]; a[i] := a[i+1]; a[i+1] := k;
i := 0;// текширишни яна бошидан бошлаш учун
end;
i := I + 1;
end;
// массивни маълумотлар ойнасидан чиқариш
if n > 0 then begin
st := 'Тартибланган массив:' + #13;
for i := 1 to n do
st := st + IntToStr(i) + ' ' + a[i] + #13; ShowMessage(st);
end;
end;
```



6.16-расм. Кўпиксимон усулнинг диалог ойнаси

менюга

6.8. Кўп ўлчовли массивлар

Биз юқорида чизиқли жадвал деб аталадиган бир ўлчовли массивлар билан ишлашни ўргандик. Массивни бир ўлчовли деб аташнинг сабаби шуки, унинг элементлари битта сатрда ёки битта устунда жойлашган бўлади. Ушбу пунктда биз икки ўлчовли массивлар билан ишлашни ўрганамиз. Унинг элементлар бир нечта сатр ва устанларда жойлашган бўлади. Ихтиёрий матрица, детерминант, ўқувчиларнинг табелидаги баҳоларни икки ўлчовли массивга мисол сифатида олиш мумкин.

Кундалик ҳаётимизда кўпинча жадвал кўринишида ифодаланган маълумотлар билан ишлашимизга тўғри келади. Масалан, DAEWOO фирмасининг машиналари билан савдо қиладиган бир фирманинг 2006 йилдаги фаолияти қуйидаги жадвал билан берилган бўлиши мумкин:

6.7-жадвал.

	Январь	Февраль	Март	...	Ноябрь	Декабрь

Тико						
Дамас						
Нексия						
Матиз						
Лиганза						

Одатда ҳар бир жадвалнинг битта сатри ёки устуни бир ҳил мазмун ва типдаги маълумотлардан иборат бўлади. Шунинг учун дастурда бундай жадвалларни икки ўлчовли массивлар шаклида қайта ишлаш мақсадга мувофиқ ҳисобланади. Икки ўлчовли массивларни бир ўлчовли массивлар тўплами деб ҳам қараш мумкин.

```
Tico: array [1..12] of integer;
Damas: array [1..12] of integer;
Nexia: array [1..12] of integer;
Matiz: array [1..12] of integer;
Liganza: array [1..12] of integer;
```

Келтирилган массивларнинг ҳар бири битта маркадаги машиналарни сотишни англатади, массивларнинг ҳар бир элементи эса маълум бир ойдаги савдони билдиради. Массивларни қуйидагича кўринишда ҳам ифодалаш мумкин:

```
jan: array [1..5] of integer;
feb: array [1..5] of integer;
mar: array [1..5] of integer;
. . . . . .
dec: array [1..5] of integer;
```

Массивларни бундай кўринишда ифодаланса, хар бир массив маълум бир ойда сотилган машиналарни, массивларнинг элементлари ўзларига мос сотилган автомобил маркаларини кўрсатади.

Агар жадвал тўлалигича бир ҳил типдаги маълумотлардан (масалан, бутун сонлардан) иборат бўлса, бундай жадвалларни икки ўлчовли массивлар шаклида ёзиш мумкин. Икки ўлчовли массивлар умумий кўринишда қуйидагича ёзилади:

Ном : array[n .. m, k .. l] of Tun

Бу ерда ***Ном***- массивнинг номи; ***array*** — Delphi даги калит сўз; ***n*** – сатрларнинг қуйи индекси, ***m*** – юқори индекси, ***k*** - устунларнинг қуйи индекси, ***l*** - устунларнинг юқори индекси; ***Tun*** — массив элементларининг типи.

6.7-жадвални қуйидагича икки ўлчовли массив шаклида ёзиш мумкин:

```
yakun: array [1..12, 1..5] of integer
```

Икки ўлчовли массивларнинг элементлари сонини (m-n+1)x(k-l+1) формула билан топиш мумкин. Шундай қилиб, ***yakun*** массиви 60 та Integer типидаги маълумотлардан иборат.

Массив элементига мурожаат қилиш учун унинг номи ва квадрат қавсларда индекс номерларини кўрсатиш лозим. Биринчи индекс массивнинг сатрлари номерини, иккинчи индекс эса устун номерларини англатади. Масалан, ***yakun***[2,3] элементи март ойида сотилган Нексия машиналарини билдиради.

Массивлар билан ишлаганда ***for*** буйруғидан фойдаланиш қулай. Масалан, бир йил лавомида сотилган Дамас машиналарини ҳисоблаш учун дастур парчаси қуйидагича ёзилади:

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

Қуйидаги дастур парчаси массив элементларининг (бир йилда сотилган автомашиналарнинг умумий сони) йиғиндисини топади:

```
S := 0;
for i := 1 to 5 do// автомобилларнинг 5 та модели
for j := 1 to 12 do//12 ойдаги
s := s + itog[i,j];
```

Юқоридаги мисолда ички цикл (j бўйича цикл) ҳар гал бир марта тўла бажарилганда, ташқи циклнинг бошқарувчи ўзгарувчисининг (i-ўзгарувчисининг) қиймати 1 га ортади. Шундан кейин ички цикл яна бир марта тўла бажарилади ва х.к. Шундай қилиб, s ўзгарувчининг қийматига ***yakun*** массивининг элементлари yakun[l,l], yakun[l,2], ..., yakun[l,12], yakun [2,l], yakun [2,2], ..., yakun [2,12] ва х.к. кетма-кет қўшилади.

Мисол тариқасида 2000 йилдаги Сидней олимпиадаси натижаларини қайта ишлаш дастурини

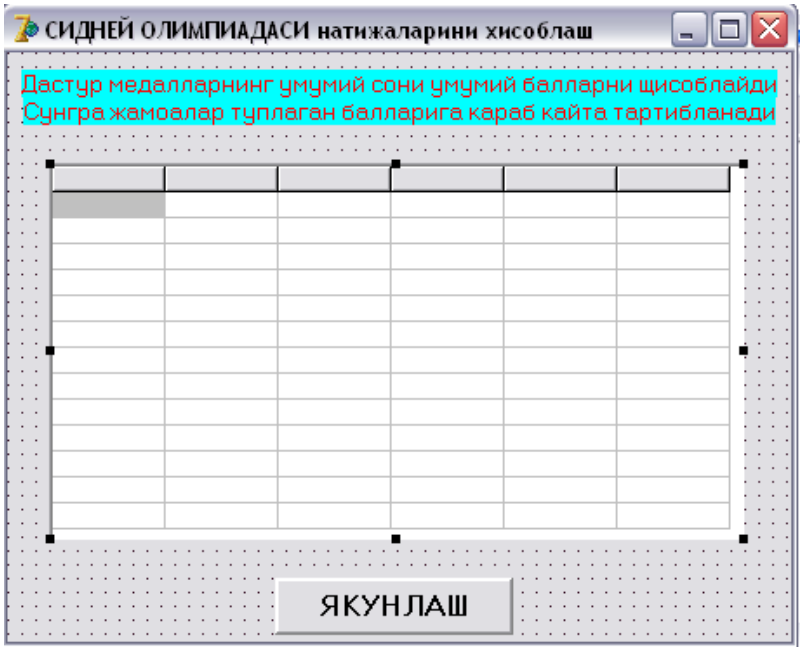
келтирамиз. Бошланғич маълумотлар 6.8-жадвалда берилган.

2000 йилдаги Сидней олипиадаси натижалари **6.8-жадвал**

мамлакат	**олтин**	**кумуш**	**Бронза**
Австралия	16	25	17
Беларусь	3	3	11
Буюк Британия	11	10	7
Германия	14	17	26
Италия	13	8	13
Ҳитой	28	16	15
Корея	8	9	11
Куба	11	11	7
Нидерландия	12	9	4
Россия	32	28	28
Руминия	11	6	9
АҚШ	39	25	33
Франция	13	14	11
Япония	5	8	5

Бу дастур ҳар бир мамлакат вакиллари олган медалларнинг умумий миқдори ҳамда ҳар бир мамлакат тўплаган очколар (баллар) ни ҳисоблайди. Балларни ҳисоблашда ҳар бир олтин медаль учун жамоага – 7 балл, кумуш учун – 6, брона учун – 5 балл берилади.

Дастурнинг диалог ойнаси қуйидагича:



6.18-расм. Олимпиада якунлари дастурининг диалог ойнаси

Массивга маълумотларни киритишда StringGrid компонентасидан фойдаланилади. Унинг қийматлари 6.9-жадвалда берилган.

StringGrid компонентасининг ҳусусияти қийматлари 6.9-жадвал

Ҳусусияти	**Қиймати**
Name	Tab1
ColCount	6
RowCount	14

FixedCols	0
FixedRows	1
Options . goEditing	TRUE
DefaultColWidth	65
DefaultRowHeight	14
GridLineWidth	1

Фиксирланган биринчи сатрнинг ячейкалари жадвал устунларига сарлавҳа қўйиш учун мўлжалланган. Дастурнинг формаси қурилаётган вақтда массивнинг ***cells*** ячейкаларининг қийматларини FormCreate (унинг матни 6.11-листингда берилган) ходисаларни қайта ишлаш процедураси ҳисоблайди. Бу ходисаларни қайта ишлаш процедураси формани фаоллаштирилганда ишга тушади. Бундан ташқари, бу процедура олимпиада қатнашчилари рўйхатини 1-устунга ёзади.

Бошланғич маълумотларни қайта ишлаш дастури **ЯКУНЛАШ** (Button1) тугмаси чертилганда ишга тушади. Унинг матни 6.11-листингда келтирилган.

6.12-листинг. Икки ўлчовли массивни қайта ишлаш

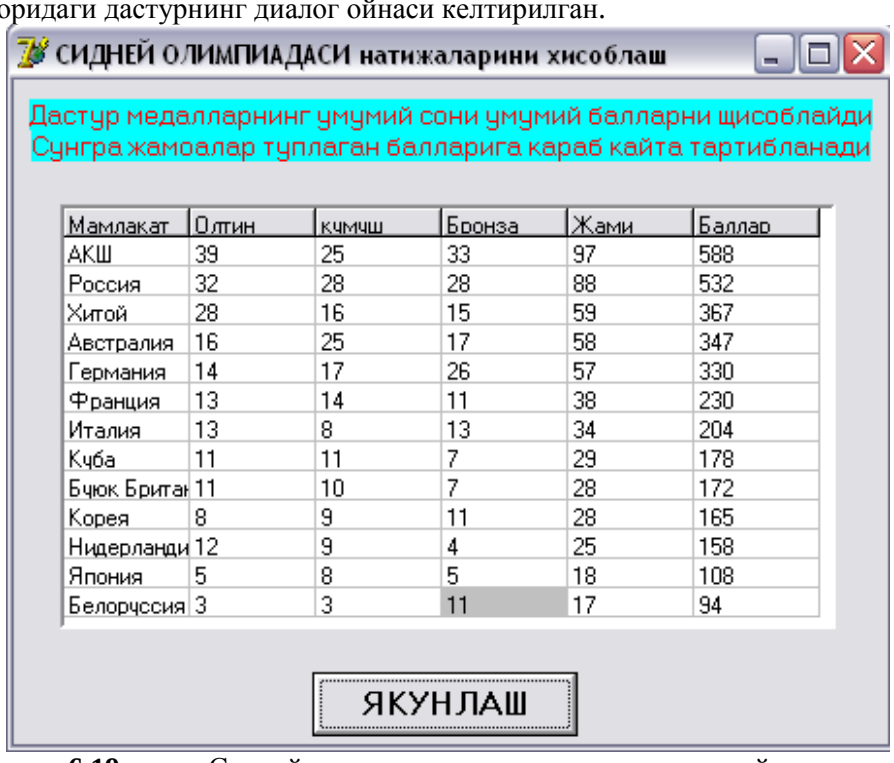
```
unit olim;
 interface
 uses   Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,   Dialogs, StdCtrls, Grids;
 type
  TForm1 = class(TForm)
    tab1: TStringGrid;
    Button1: TButton;
    Label1: TLabel;
    Label2: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
   private  { Private declarations }
  public    { Public declarations }
  end;
 var
  Form1: TForm1;
 implementation
 {$R *.dfm}
 procedure TForm1.Button1Click(Sender: TObject);
 var
 c,r:integer;//жадвалнинг устун ва сатр номерлари
 s:integer;//жамоанинг умумий медаллари
 p:integer;//жамоанинг очкоси
 m:integer;//Энг катта очко олган жамоанинг сатр номери
 i:integer;//Сатр номери. Тартиблаш вақтида фойдаланилади
 s1:string;
 begin
 for r := 1 to tab1.rowcount do //ҳамма сатрларни қайта ишлаш
 begin s := 0;
//медалларнинг умумий сонини аниқлаймиз
 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';//очколарни ҳисобланмоқда
 p := 7*StrToInt(tab1.cells[1,r])+
 6*StrToInt(tab1.cells[2, r] )
 + 5*StrToInt(tab1.cells[3,r]);
//натижаларни чиқариш
 tab1.cells[4,r] := IntToStr(s);//жами медаллар
 tab1.cells[5,r] := IntToStr(p);//очколар
 end;
```

```
// жадвални камайиш тартибида, 5ғустун бўйича тартиблаймиз.
// энг кичик элементлар усули билан тартиблаш
for r := 1 to tab1.rowcount-1 do
begin
m := r;// Энг катта элемент r-сатр
for := 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- ва m – сатрларни алмаштирамиз
for c := 0 to 5 do begin
s1 := tab1.Cells[c,r];
tab1.Cells[c,r] := tab1.Cells[c,m];
tab1.Cells[c,m] := s1;
end; end;  end; end;
```

```
procedure TForm1.FormCreate(Sender: TObject);
begin
tab1.Cells[0,0] := 'Мамлакат'; tab1.Cells[1,0] := 'Олтин';
tab1.Cells[2,0] := 'кумуш';      tab1.Cells[3,0] := 'Бронза';
tab1.Cells[4,0] := 'Жами';                tab1.Cells[5,0] := 'Баллар';
tab1.Cells[0,1] := 'Австралия';          tab1.Cells[0,2] := 'Белоруссия';
tab1.Cells[0,3] := 'Буюк Британия';
tab1.Cells[0,4] := 'Германия'; tab1.Cells[0,5] := 'Италия';
tab1.Cells[0,6] := 'Ҳитой';               tab1.Cells[0,7] := 'Корея';
tab1.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;
end.
```

Дастурни ишга тушириш

6.19-расмда юқоридаги дастурнинг диалог ойнаси келтирилган.



6.19-расм. Сидней олимпиадаси дастурининг диалог ойнаси

6.9. Массивлардан фойдаланишдаги ҳатоликлар

Массивлардан фойдаланганда энг кўп учрайдиган ҳатолик бу – массив индекс ифодасининг кўрсатилган диапазондан четга чиқишидир. Агар индекс сифатида константа келиб унинг қиймати белгиланган чегарадан чиқиб кетса, у ҳолда бундай ҳатоликни компилятор "кўради". Масалан, дастурда

```
day : array[0..6] of string[ll],
```

массиви эълон қилинган бўлса, компиляция жараёнида

```
day [7] := 'Якшанба';
```

буйруғи ҳато деб қўрсатилади.

Агар массив элементига индекс сифатида ўзгарувчи ёки ифода қўлланилган бўлса, у ҳолда дастурнинг бажарилиши давомида ҳатолик юзага келиши мумкин. Масалан, дастурда

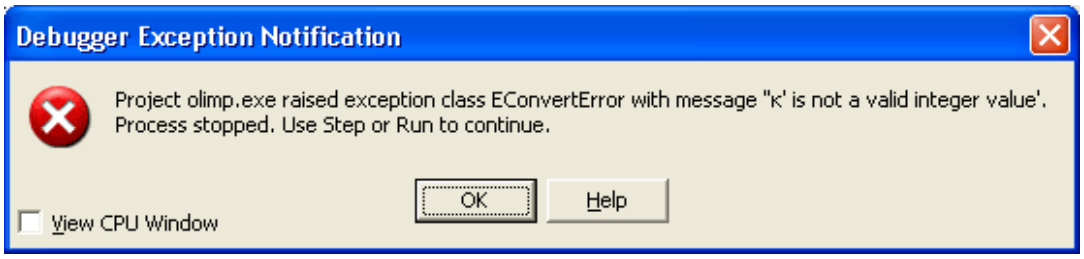
```
tab1: array [1..N] of integer;
```

массиви эълон қилинган бўлса, у ҳолда

```
for i:=0 to N do tab1[i] := 5;
```

буйруғи расман тўғри деб эътироф этилади ва у муваффақиятли компиляция қилинади. Аммо, дастурнинг бажарилиши давомида tab1 массивнинг мавжуд бўлмаган нолинчи элементига мурожаат қилишга уринилганда, экранга ҳатолик ҳақидаги ахборот чиқарилади. Бу ахборотнинг кўриниши ва мазмуни дастурни қаерда ишга туширилганлигига боғлиқ.

Қаралаётган дастур Delphi дан ишга туширилса, 6.20-расмдаги ахборот чиқарилади.



6.20-расм. Мавжуд бўлмаган элементга мурожаат ҳақидаги ахборот
(дастур Delphi дан ишга туширилган)

Агар дастур Windows дан ишга туширилган бўлса, у ҳолда мавжуд бўлмаган элементга мурожаат қилинганлиги ҳақидаги ахборот

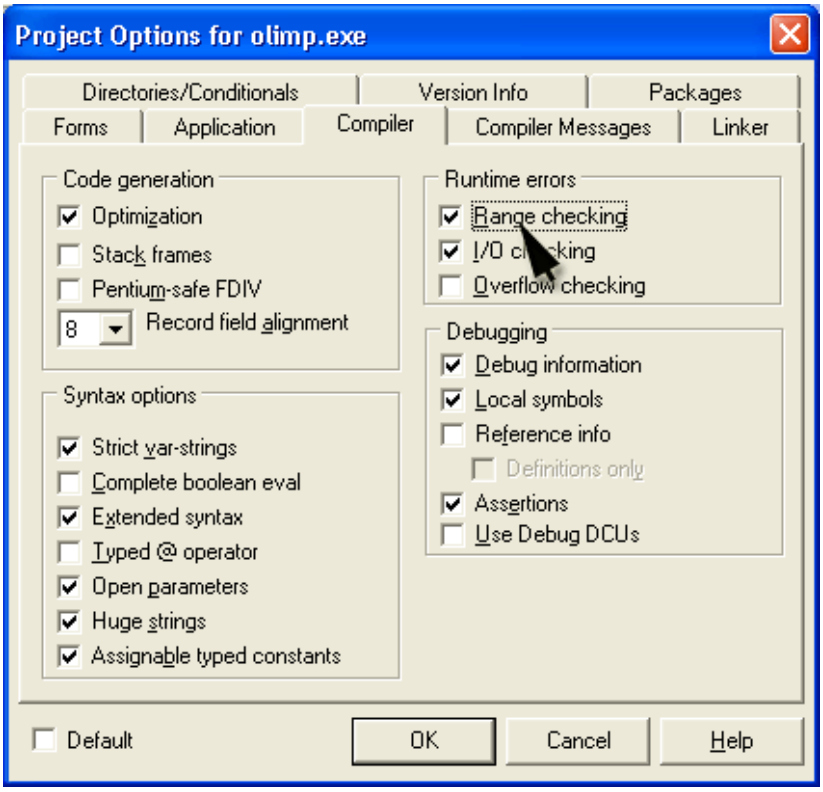
Range check error (диапазон назоратининг ҳатолиги).

тарзида бўлади. Ойнанинг сарлавҳасида ҳатолик учраган илованинг номи кўрсатилади.

Индексдаги ифода қийматининг диапазон чегарасидан четга чиқиши вақтидаги дастурнинг ҳулқи компиляторнинг созланишидаги қийматлар билан аниқланади.

6.21-расм. Мавжуд бўлмаган элементга мурожаат ҳақидаги ахборот
(дастур WINDOWS дан ишга туширилган)

Дастур индексдаги ифодаларнинг қийматларини назорат қилиши учун (бу ҳолда Delphi бажарилаётган дастурга назорат қилишни таъминловчи бажарилувчи дастурий компонентани қўшиб қўяди), Project менюсида очиладиган **Project Options** диалог ойнасининг **Compiler пунктида Range checking** (диапазон назорати) байроқчасини ўрнатиш лозим. У **Runtime errors** (бажариш вақтидаги ҳатоликлар) гуруҳига киради. (6.22-расм).



6.22-расм. Project Options диалог ойнасининг **Compiler** пункти

7-боб. ПРОЦЕДУРАЛАР. ПРОЦЕДУРА-ФУНКЦИЯЛАР

7.1. Формал ва жорий ўзгарувчилар. Локал ва глобал ўзгарувчилар

Формал ва жорий ўзгарувчилар. Одатда кўплаб масалаларни ечиш жараёнида олдиндан масалани қандай қийматлар (ўзгарувчилар) учун берилганини билиб бўлмайди. Лекин уни ечиш учун шартли равишда турли ўзгарувчилар киритилади ва шу ўзгарувчилар учун қўйилган масалани тўла ечишнинг қонун - қоидалари (алгоритми) яратилади. Ана шу ўзгарувчилар формал ўзгарувчилар дейилади. Турли фан соҳаларидаги масалалар, масалан, математик масалаларнинг ечиш йўллари ана шу формал ўзгарувчиларга нисбатан келтирилади. Масалан, умумий кўриниши

$$ax^2 + bx + c = 0$$

бўлган квадрат тенгламанинг ечиш учун $D = b^2 - 4ac$ дискриминант топилади. Агар D≥0 бўлса, квадрат тенгламанинг илдизларини

$$x_1 = \frac{-b + \sqrt{D}}{2a}, \quad x_2 = \frac{-b - \sqrt{D}}{2a}$$

формулалар билан топилади. Бу формулалардаги барча ўзгарувчилар формал ўзгарувчилар бўлади.

Кейин шу синфга тааллуқли бўлган конкрет масалани олинади. Бу масалада одатда ҳамма ўзгарувчилар аниқ кўрсатиб қўйилади. Масалани ана шу ўзгарувчилар учун ҳал қилиш талаб қилинади. Бундай ўзгарувчилар жорий ўзгарувчилар деб аталади. Энди масалани ечиш учун яратилган ҳамма қонун-қоидаларни формал ўзгарувчиларнинг ўрнига масала шартида берилган жорий ўзгарувчиларни қўйиб бажарилади. Масалан, ҳаётдаги бирор масалани ечиш жараёнида кўриниши $px^2 + qx + r = 0$ бўлган квадрат тенгламага дуч келинди. Унинг коэффициентлари p=2, q=12, r=3 бўлсин. Ана шу ўзгарувчилар жорий ўзгарувчилар дейилади. Жорий ўзгарувчиларни формал ўзгарувчилар ўрнига қўйиш, яъни умумий формуладаги a –ўрнига p, b – ўрнига q, c – ўрнига эса r ўзгарувчиларни қўйиш керак. Шундан кейин умумий формуладан келиб чиқиб, квадрат тенгламани осонгина ечиш мумкин.

Локал ва глобал ўзгарувчилар. Масала шартида кўрсатилмаган, лекин масалани ечиш учун ҳисобланиши зарур бўлган ўзгарувчиларни оралиқ ўзгарувчилар дейилади. Масалан, квадрат тенглама учун ***a, b, c*** –коэффициентлар берилади ва x_1, x_2 илдизларни топиш талаб қилинади. Дискриминант D – ўзгарувчиси квадрат тенглама шартида кўрсатилмайди. Лекин унинг қийматини квадрат

тенгламани ечиш учун албатта ҳисоблаш керак бўлади. Шунинг учун бу ўзгарувчини оралиқ ўзгарувчи деб ҳисобланади.

Одатда оралиқ ўзгарувчилар одатда фақат битта процедура учун тааллуқли бўлади. Шунинг учун уларни **локал (маҳаллий) ўзгарувчилар** деб ҳам юритилади.

Глобал ўзгарувчилар деб бир вақтнинг ўзида ҳам асосий дастурга, ҳам процедураларга бирдек тааллуқли бўлган ўзгарувчиларга айтилади.

Бу ўзгарувчилардан Delphi муҳитида фойдаланиш йўл-йўриқлари ҳақида кейинроқ тўхталамиз.

менюга

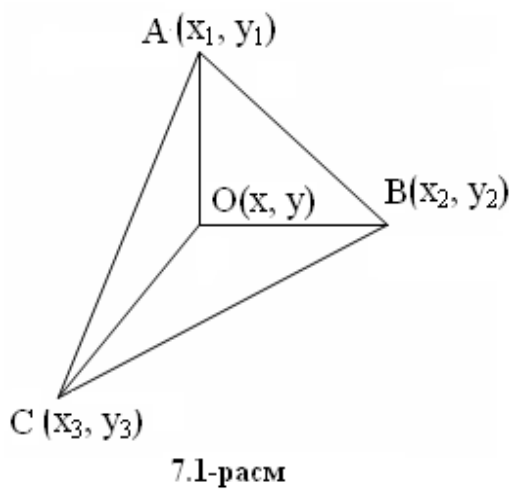
7.2. Қисм дастурлар

Кўпинча, дастурчи дастур устида ишлар экан, ундаги айрим буйруқлар кетма-кетлиги дастурнинг турли қисмларида бир неча марта учрашини сезиб қолади. Масалан, учларининг координаталари берилган учбурчакнинг томонларини ҳисоблашда

$$AB = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

формуласи турлича кўринишларда уч марта учрайди.

Дастур матнида кодларнинг қайта-қайта учрашидан ҳалос бўлиш мумкин. Бунинг учун, қайта-қайта ёзилиши талаб қилинган буйруқлар кетма-кетлигини ажратиб олинади. Бу буйруқлар кетма-кетлигини қисм дастур деб аташ қабул қилинган. Қисм дастур учун формал ўзгарувчилар танланади.



7.1-расм

Асосий масалани ечишга қаратилган, эхтиёжга қараб қисм дастурларнинг ишини бошқарадиган дастурни асосий дастур деб атаймиз. Асосий дастурда қисм дастурдаги формал ўзгарувчилар ўрнига жорий ўзгарувчилар танланиб, уларга мурожаат қилиш ташкил қилинади.

Қуйидаги масалани кўрайлик. Учбурчак учларининг координаталари ***(x₁, y₁), (x₂, y₂), (x₃, y₃)*** берилган бўлсин. Берилган ***(x, y)*** координатали нуқта шу учбурчак ичида ётадими? (7.1-расм)

Бу масалани ечиш учун (***(x, y)*** координатали нуқтани учбурчакнинг учлари билан туташтирамиз. Натижада Биз битта ўрнига тўртта учбурчакка эга бўламиз: Δ_{ABC}, Δ_{AOB}, Δ_{AOC}, Δ_{BOC} . Бу учбурчакларнинг юзалари бўлган S_{ABC}, S_{AOC}, S_{AOB}, S_{BOC} юзаларни ҳисобланади. Энди масала шартида қўйилган саволга жавоб бериш учун

$S_{ABC} = S_{AOC} + S_{AOB} + S_{BOC}$

муносабатни текширамиз. Агар бу муносабат ўринли бўлса, у ҳолда бериган (x, y) координатали нуқта учбурчак ичида ётади, акс ҳолда - йўқ.

Бундан кўриниб турибдики, битта дастур матнида тўрт марта учбурчакнинг юзаларини ҳисоблаш буйруқларини ёзиш керак бўлади. Мана шундай ҳолларда битта учбурчак учун формал параметрлар танлаб (масалан, бу учбурчак учларининг координаталари ***(a₁, b₁), (a₂, b₂), (a₃, b₃)*** бўлсин), шу параметрлар учун учбурчак юзини ҳисоблаш қисм дастурини ёзиш лозим. У қуйидаги буйруқлардан иборат бўлади:

```
begin
    a:=sqrt(sqr(a2-al)+sqr(b2-bl)) ;
    b :=sqrt(sqr(a3-a2)+sqr(b3-b2)) ;
    c:=sqrt(sqr(a3-al) +sqr(b3-bl)) ;
    p:=(a+b+c)/2 ;
    yuza:=sqrt(p*(p-a) *(p-b) *(p-c)) ;
end ;
```

Энди биз қисм дастурлардан фойдаланиш йўл-йўриқларини кўрамиз.

Қисм дастурлардан фойдаланишнинг афзаллиги қуйидагши вазиятларда кўриш мумкин.Биринчидан, энди асосий дастурларда такрорланадиган кодлар бўлмайди. Бу эса дастур матнини яратиш ва компьютер хотирасига киритиш жараёнинини осонлаштиради. Иккинчидан, зарур бўлса, қисм

дастурларга ўзгартириш киритиш осон. Қисм дастурдан фойдаланилмаган ҳолда бутун матнни кўриб чиқиш керак бўлади. Агар дастур қисм дастурдан фойдаланаётган бўлса, у ҳолда фақат қисм дастурни кўриб чиқилади ҳалос. Учинчидан, дастурнинг ишончлилик даражаси ортади. Чунки, қисм дастурлардан фақат кодлар такрорланадиган ҳолларда эмас, балки катта масалаларни кичик масалачаларга бўлиб ечиш ҳам дастур яратиш жараёнини енгиллаштиради. Бунда агар ҳар бир масалача учун қисм дастур яратилса, бундай дастурни ўқиш, тушуниш, таҳлил қилиш, ҳатоликларин аниқлаш каби вазифаларни осонгина ҳал қилиш мумкин.

Қисм дастур – бу унчалик катта бўлмаган дастур бўлиб, умумий масаланинг маълум бир қисмини ечишга мўлжалланади. Delphi да қисм дастурлардан икки ҳил кўринишда фойдаланиш мумкин: процедура ва процедура функция (бундан кейин уни осонгина қилиб функция деб атаймиз). Ҳар бир қисм дастурнинг номи бўлиб, бу ном унга асосий дастурдан туриб мурожаат қилиш учун хизмат қилади.

Одатда, қисм дастурлар қандайдир параметрларга эга бўлади. Бу параметрларни формал ва жорий параметрлар деб аталади.

Қисм дастурни эълон қилишда фойдаланилган параметрларни формал, асосий дастурдан уларга мурожаат қилишда қўлланадаиган параметрларни эса жорий параметрлар деб аташ қабул қилинган. Бу параметрлар қисм дастурга асосий дастурлардан маълумотларни олиб кириш ҳамда қисм дастурдан асосий дастурга маълумотларни олиб чиқиш учун хизмат қилади.

Айрим ҳолларда, жорий параметр сифатида типи шу параметрнинг типи билан ҳил бўлган ифодалардан ҳам фойдаланиш мумкин.

менюга

7.3. Функция

Айрим масалаларни ечиш жараёнида турли ифодаларнинг қийматларини ҳисоблаш учун қандайдир функцияларнинг қийматларидан фойдаланишга тўғри келади. Биз кундалик ҳаётимизга чуқур кириб борган стандарт ёки элементар деб атайдиган шундай функциялар борки, кўпчилик бу функцияларни таниймиз, аммо уларнинг маъносига эътибор бермаймиз. Масалан: $y = \cos x$ ифоданинг қийматини ҳисоблаганда, *cosx* ўрнида бошқа ифода, яъни "тўғри бурчакли учбурчакдаги ***x*** бурчак қаршисидаги катетнинг гипотенузага нисбати" ($\cos x$ нинг таърифи) ётганлигини одатда ҳис қилмаймиз. Зарур бўлиб қолса, шу нисбатнинг ўрнига осонгина $\cos x$ деб ёзиб қўя қоламиз.

Агар масала шартида стандарт бўлмаган функциялар, ёки узундан-узун ифодаларни параметрларининг турли қийматлари учун ёзишга тўғри келиб қолса, дастурчи Delphi да ўз ишини шу мураккаб ифодаларнинг қийматлари ҳисоблаш жараёнини ҳам худди стандарт функциялардан фойдаланишга ўхшаш енгиллаштириши учун имкон яратилган.

Бундай ҳолларда дастурчининг ишини бир мунча соддалаштириш учун Delphi да функциялар ёки фойдаланувчи функциясини тузиш тавсия қилинади.

Процедура-функциялар одатда мураккаб ифодаларни, узун арифметик ифодаларни ёки бирор ифода қийматини аргументларнинг (параметрларнинг) турли қийматлари учун ҳисоблашга тўғри келган ҳолларда ташкил қилиниши мумкин.

Функция — бу ўзининг номига эга бўлган қисм дастур, яъни буйруқлар кетма-кетлигидир.

Функция умумий ҳолда қуйидагича эълон қилинади:

```
function Ном (параметр1 : тип1, ..., параметрК : типК) : Тип;
 var
        // бу ерда орлалиқ параметрлар эълон қилинади
begin
        // функциянинг буйруқлари кетма-кетлиги
   Ном := ифода;
end;
```

Бу ерда ***function*** — Delphi нинг функция яратилаётганлигини билдирувчи хизматчи сўзи; ***Ном*** — функциянинг номи, функцияга мурожаат қилишда фойдаланилади; ***параметр*** — функциянинг қийматини ҳисоблаш учун зарур бўлган формал ўзгарувчи бўлиб, у У ўзига қийматни асосий дастурдан шу функцияга мурожаат қилинаёганда олади; ***тип*** — асосий дастурнинг функциядан оладиган

маълумотининг типи.

Шунга алоҳида эътибор бериш керакки, функциянинг буйруқлари функциянинг номи билан аталган ўзгарувчига қиймат берадиган буйруқ билан тугайди. Функциянинг қийматини аниқлайдиган ифоданинг типи эълонда кўрсатилган функциянинг типи билан бир ҳил бўлиши лозим.

Юқори келтирилган нуқтанинг учбурчак ичида ётиши ҳақидаги масала учун функция ёзамиз.

```
Function Yuza(a1, a2, a3, b1, b2, b3: real):real;
    Var ab, ac,bc,p : real;//Оралиқ ўзгарувчилар
begin
      ab := sqrt(sqr(a2-a1) + sqr(b2-b1);
      ac := sqrt(sqr(a3-a1) + sqr(b3-b1);
      bs := sqrt(sqr(a3-a2) + sqr(b3-b2);
      p := (ab + ac + bc)/2;//ярим периметр
      //Герон формуласи қўлланмоқда
      Yuza := sqrt(p*(p-ab)*(p-ac)*(p-bc);
end;
```

Функциянинг буйруқларига ўтиш функцияга мурожаат қилиш ёки уни чақириш деб аталади. Функция буйруқларидан шу функцияни чақирган буйруққа ўтиш эса функциядан қайтиш деб аталади.

Функцияга мурожаат қилиш умумий кўринишда қуйидагича ёзилади.

ўзгарувчи := Функция (жорий параметрлар) ;

бу ерда ***ўзгарувчи*** – қиймат олаётган ўзгарувчи; ***Функция*** - қиймати ***ўзгарувчи*** га бериладиган функциянинг номи; ***жорий параметрлар*** -

функциянинг қийматини ҳисоблаш учун фойдаланиладиган жорий ўзгарувчилар рўйхати. Бу параметрлар сифатида константалар ёки ўзгарувчилардан фойдаланиш мумкин.

Шуни ёдда тутиш керакки, биринчидан, ҳар бир функция маълум бир типдаги маълумотни олиб келади, шунинг учун бу қийматни оладиган ўзгарувчининг типи ҳам функция типига мос бўлиши лозим; иккинчидан формал ва жорий ўзгарувчилар рўйхати хар бир функция учун қатъий аниқланган бўлиши лозим.

Функциядан фойдаланиш. Дастурда функциядан фойдаланиш учун одатда унинг матнини энг содда ҳолда шу функциядан фойдаланадиган қисм дастурдан олдин кўрсатиш қабул қилинган.

7.1-листингда **нуқта учбурчак ичида ётадими?** масаласининг дастури матни келтирилган. Бу дастурнинг диалог ойнаси 7.2-расмда берилган.

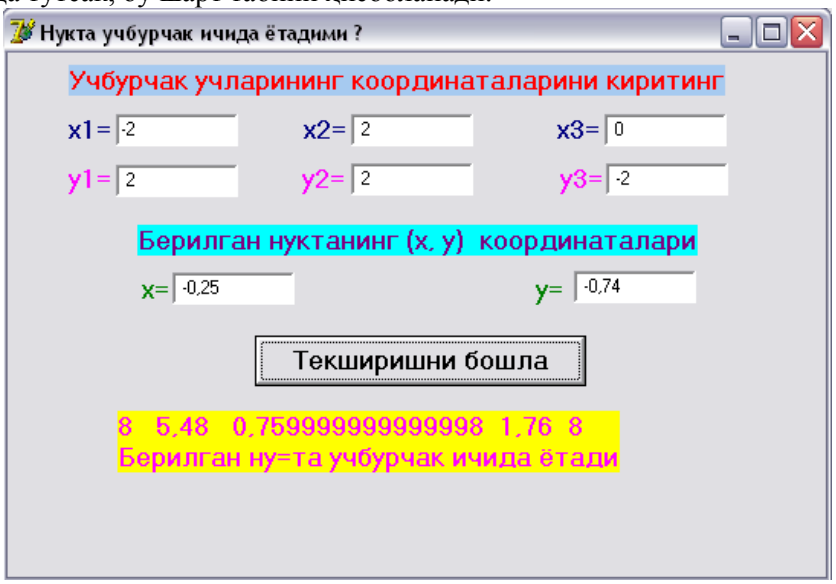
7.1-листинг. Нуқта учбурчак ичида ётадими? дастури матни

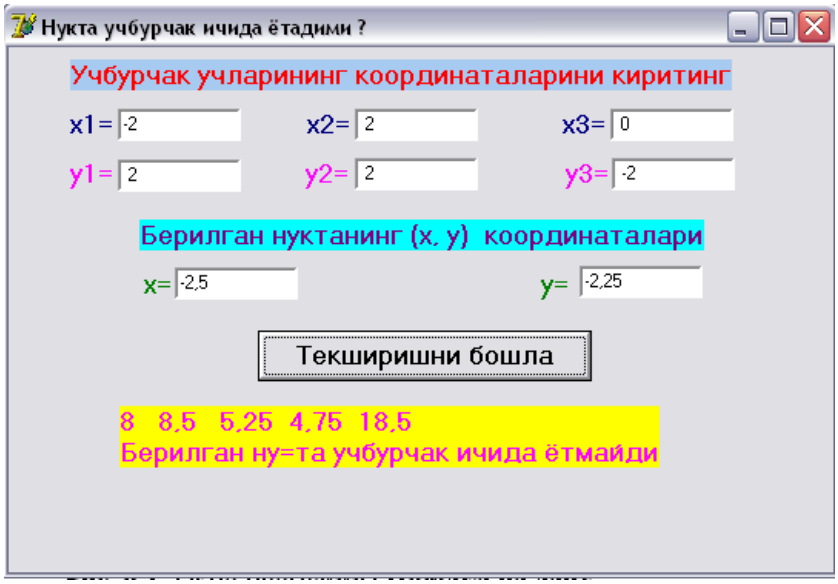
```
unit Unit1;
interface
uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,   Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Edit6: TEdit;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Edit7: TEdit;
```

```
    Edit8: TEdit;
    Button1: TButton;
    Label11: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
    var x,x1,x2,x3,y,y1,y2,y3 : real;//координаталар
        S: real;//ABC учбурчакнинг юзи
        s1,s2, s3: real;//кичик учбурчакларнинг юзалари
        d:string;
    Function Yuza(a1, a2, a3, b1, b2, b3: real):real;
        Var ab, ac,bc,p : real;//Оралиқ ўзгарувчилар
    begin
        ab := sqrt(sqr(a2-a1)+sqr(b2-b1));
        ac := sqrt(sqr(a3-a1)+sqr(b3-b1));
        bc := sqrt(sqr(a3-a2)+sqr(b3-b2));
        p := (ab + ac + bc)/2;//ярим периметр
       //Герон формуласини қўлланмоқда
        Yuza := sqrt(p * (p - ab) * (p - ac) * (p - bc));
     end;
begin
    //Учбурчакнинг координаталари
     x1 := strtofloat(edit1.Text);
     x2 := strtofloat(edit2.Text);
     x3 := strtofloat(edit3.Text);
     y1 := strtofloat(edit4.Text);
     y2 := strtofloat(edit5.Text);
     y3 := strtofloat(edit6.Text);
    //берилган нуқтанинг координаталари
     x := strtofloat(edit7.Text);
     y := strtofloat(edit8.Text);
    //Катта учбурчакнинг юзи
     s := yuza(x1,x2,x3,y1,y2,y3);
    //Кичик учбурчакларнинг юзалари ҳисобланмоқда
     S1 := yuza(x1,x2,x,y1,y2,y);
     s2 := yuza(x1,x,x3,y1,y,y3);
     s3 := yuza(x,x2,x3,y,y2,y3);
    d := floattostr(s)+ '   ' + floattostr(s1) + '   ' +floattostr(s2)
     +'  ' +  floattostr(s3) + '  ' +floattostr(s1 + s2 + s3) + #13;
     if abs(s-(s1 + s2 + s3))<=0.00000001 then
  label11.Caption := d+'Берилган нуқта  учбурчакнинг ичида ётади'
  else
   label11.Caption := d+' Берилган нуқта  учбурчакнинг ичида ётмайди'
 end;
end.
```

Дастурни ишга тушириш

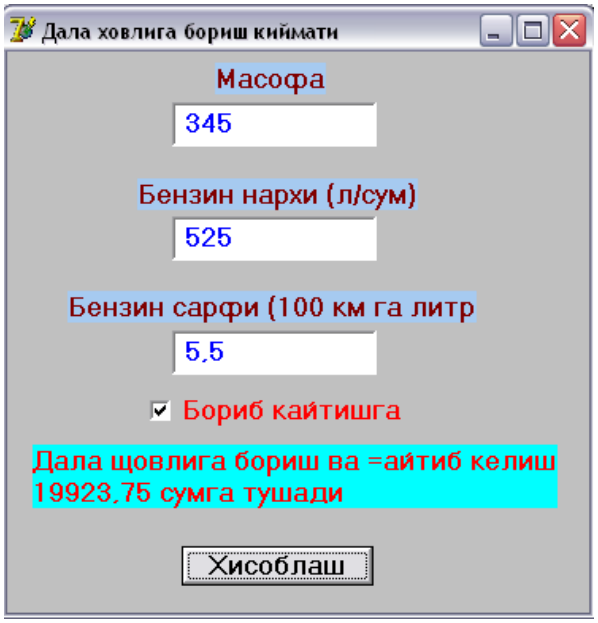
Дастур матнидаги ***abs(s-(s1+s2+s3))<=0.00000001*** шартига эътибор беринг. Биз масалани ечиш ғоясига кўра, S=S1+S2+S3 ёки S-(S1+S2+S3)=0 шартини текширишимиз керак эди. Аммо, бу формулалардаги параметрларнинг ҳақиқий сон эканлиги, Delphi муҳитида аниқликнинг жуда катта эканлиги, ёки Delphi муҳити 0.999999999999999 ёки 1.00000000000001 сонларини ҳақиқий 1 сони деб қабул қилиши мумкинлигини назарда тутсак, бу шарт табиий ҳисобланади.





7.2-расм. Дастур диалог ойнасининг кўринишлари

Қуйидаги дастур (унинг матни 7.2-листингда берилган) дала ҳовлига бориб келиш қийматини ҳисоблайди. Унинг диалог ойнаси 7.3-расмда келтирилган. Бу масала учун бошланғич маълумот сифатида масофа, бир литр бензин нархи, 100 км га сарфланадиган бензин миқдори олинади. Бу маълумотларни киритиш учун Edit1, Edit2 ва Edit3 майдонларидан фойдаланамиз. OnKeyPress ходисаларни қайта ишлаш функцияси IsFloat функциясидан фойдаланади. Бу функция киритилаётган белгиларни назорат қилиш учун мўлжалланган. Бу функция майдонларга фақат рухсат берилган белгилар, яъни, рақамлар, <Enter>, <Backspace> тугмаларини босилганлигини эътиборга олади ҳалос, қолган тугмаларга эътибор бермайди.



7.3-расм. Дала ҳовлига бориб келиш қиймати дастурининг диалог ойнаси

7.2-листинг. Функциядан фойдаланишга намуна

```
unit dala1;
interface
uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,   Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Label2: TLabel;
    Edit2: TEdit;
    Label3: TLabel;
    Edit3: TEdit;
    CheckBox1: TCheckBox;
    Label4: TLabel;
    Button1: TButton;
    procedure EditlKeyPress(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}
//Каср сон учун киритилган белгининг тўғрилигини аниқлайди
function IsFloat(ch : char; st: string) : Boolean;
begin
if (ch >= '0') and (ch <= '9')//рақамлар
or (ch = #13)//<Enter> тугмаси
or (ch = #8)//<Backspace> тугмаси
then
begin
IsFloat := True;//белги тўғри ёзилган
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//ажратувчи белги рақамдан кейин ёзилиши керак
      IsFloat := True else//қолган белгилар таъқиқланган
                  IsFloat := False;
end;
end;

//Тугмани МАСОФА ойнасида босиш
procedure TForm1.EditlKeyPress(Sender: TObject; var Key: Char);
begin
if Key = Char(VK_RETURN)
then Edit2.SetFocus//Курсорни НАРХ ойнасига ўтказиш
else
If not IsFloat(Key,Edit2.Text) then Key := Chr(0);
end;

//НАРХ майдонида тугмани босиш
procedure TForm1.Edit2KeyPress(Sender:TObject; var Key: Char);
begin
if Key = Char(VK_RETURN)
then Edit3.SetFocus//курсорни ҳаражат майдонига ўтказиш
else If not IsFloat(Key,Edit2.Text)
then Key := Chr (0);
end;

  //тугмани ҲАРАЖАТ майдонида босиш
procedure TForm1.Edit3KeyPress(Sender:TObject; var Key: Char);
begin
if Key = Char(VK_RETURN)
then Button1.SetFocus//ҲИСОБЛАШ тугмасини фаоллаштирилмокда
else If not IsFloat(Key,Edit2.Text) then Key := Chr (0);
end;

//ҲИСОБЛАШ тугмаси чертилганда
procedure TForm1.Button1Click(Sender: TObject);
var
masofa : real;//масофа
narx : real;//нарх
sarf : real;//100 км га сарф
summ : real;//сумма
mes: string;
begin
masofa := StrToFloat(Edit1.Text);
narx := StrToFloat(Edit2.Text);
sarf := StrToFloat(Edit3.Text);
summ := masofa/100 * sarf * narx;
if CheckBox1.Checked then summ := summ * 2;
mes := 'Дала ҳовлига бориш';
if CheckBox1.Checked then mes := mes + ' ва қайтиб келиш'+#13;
mes := mes + FloatToStrF(summ,ffGeneral,10,2) + ' сўмга тушади';
Label4.Caption := mes;
end;
end.
```

Дастурни ишга тушириш

менюга

7.4. Процедура

Процедура — бу қисм дастурнинг кўринишларидан бири ҳисобланади. Одатда қисм дастурлар процедура шаклида икки ҳолдан бирида расмийлаштирилади:

- қисм дастур асосий дастурга маълумотларни қайтариши шарт бўлмаган ҳолларда. Масалан, диалог ойнасида графикларни чизиш талаб қилинганда.
- қисм дастур уни чақирган дастурга биттадан ортиқ қийматни қайтаришига тўғри келганда. Масалан, квадрат тенгламани ечаётган қисм дастур иккита илдизни ҳисоблаши ва уларни асосий дастурга олиб чиқиши керак.

Процедурани эълон қилиш. Бу масала умумий кўринишда қуйидагича ҳал қилинади:

```
procedure Ном (var параметр1: тип1; ... var параметрК: типК);
 var
    // Бу ерда локал, яъни оралиқ ўзгарувчилар эълон қилинади
 begin
    // бу ерда процедуранинг буйруқлари ёзилади
 end;
```

Бу ерда ***procedure*** — Delphi тилида процедуранинг бошланишини англатувчи хизматчи сўз; ***Ном*** — процедурага мурожаат қилишда фойдаланиладиган процедуранинг номи; ***параметр*** — формал параметр, у процедура буйруқларида иштирок этади; ***var*** сўзи параметрдан аввал ёзилиши шарт эмас. Аммо, у ёзилган бўлса, бу ҳолат процедурани чақириш буйруғида албатта жорий ўзгарувчи сифатида ўзгарувчи келиши шартлигини билдиради. Бу параметрлар маълумотларни мурожаат қилувчи дастурдан процедурага ҳамда процедурадан унга мурожаат қилган дастурга олиб ўтиш учун хизмат қилади.

Мисол тариқасида берилган учта a, b, c сонлардан учбурчак ясаш мумкинлигини текширайлик. Маълумки, бу сонлардан учбурчак ясаш мумкин бўлиши учун

$$a+b \le c, \quad a+c \le b, \quad b+c \le a$$

шартлар бир вақтда ўринли бўлиши керак. Юзасини ҳисоблаш учун эса Герон формуласидан фойдаланамиз. Шу масалага мос процедурани қуйидагича ёзиш мумкин

7.5-листинг. Учбурчак ясаш процедураси

```
procedure uchburchak(a, b, c : real);
{ a, b, c – берилган учта сон, s- учбурчакнинг юзи,
z- маълумотларни экранга чиқаришга тайёргарлик}
var
  ok : boolean;// ok=True-учбурчак мавжуд, ok=False – мавжуд эмас
p : real;// ярим периметр
begin
yuza := 0;
z := '';
if (a + b >= c) and (a + b >= c) and (b + c >= a)
then begin
          ok := true;
          p := (a + b + c)/2;
          yuza := sqrt(p * (p - a) * (p - b) * (p - c));
        end
       else ok := false;
       z := floattostr(a)+ '    ' + floattostr(b) + '    ' + floattostr(c);
       if ok then
       z := z + '    ' + 'учбурчак мавжуд Унинг юзи =' + floattostr(yuza)
             else z := z + '    ' +'учбурчак мавжуд эмас';
      s2 := s2 + z + #13;
      um_yuza := um_yuza + yuza;
end;
```

Процедуралардан фойдаланиш. Тайёрланган процедурани **implementation** бўлимида ундан

фойдаланадиган қисм дастурдан аввал кўрсатилиши лозим.

Процедурага мурожаат қилиш буйруғи қуйидагича ёзилади:

Ном (Параметрлар рўйхати);

бу ерда ***Ном*** – чақирилаётган процедуранинг номи; (Параметрлар рўйхати)–бир-биридан вергул билан ажратилган жорий ўзгарувчилар рўйхати.

Процедурага мурожаат қилинганда жорий ўзгарувчи сифатида процедуранинг эълонида кўрсатилишига мос равишда типлари ўзаро тўғри келадиган ўзгарувчи, константа ёки ифода келиши мумкин. Масалан, квадрат тенгламага мурожаат қилинганда

```
uchburchak(StrToFloat(Edit1.Text), StrToFloat(Edit2.Text),
StrToFloat(Edit3.Text), k1, k2, nat);
```

ёки

```
uchburchak (a, b, c);
```

ёки

```
uchburchak (2.5, b, 3.45);
```

кўринишидаги ёзувлардан фойдаланиш мумкин.

Бизга қуйидаги масала қўйилган бўлсин. Тўртта p, q, r, t ҳақиқий сонлар берилган бўлсин. Улардан ҳосил қилинган учликлар ичидан учбурчак ясаш мумкинларини топинг. Агар учбурчак ясаш мумкин бўлса, шу учбурчакларнинг умумий юзини топинг.

Берилган тўртликлардан (p,q,r), (p,q,t), (q,r,t) ва (p,r,t) учликларни ҳосил қилиш мумкин. Ихтиёрий учликдан учбурчак ясаш мумкинлигини юқоридаги процедура ёрдамида текширишни ташкил қиламиз.

7.6-л**истинг. Учбурчак ясаш мумкинлиги ҳақида дастур матни**

```
unit uch4;
interface
uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,   Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Button1: TButton;
    Label6: TLabel;
  procedure Button1Click(Sender: TObject);
  procedure FormCreate(Sender: TObject);
  private  { Private declarations }
  public   { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
   var z, s2:string;
       yuza, um_yuza : real;
    procedure TForm1.Button1Click(Sender: TObject);
    var um_yuza, p,q,r,t:real;
        s1,s2 : string;

procedure uchburchak(a, b, c : real);
{ a, b, c – берилган учта сон, s- учбурчакнинг юзи,
z- маълумотларни экранга чиқаришга тайёргарлик}
var
```

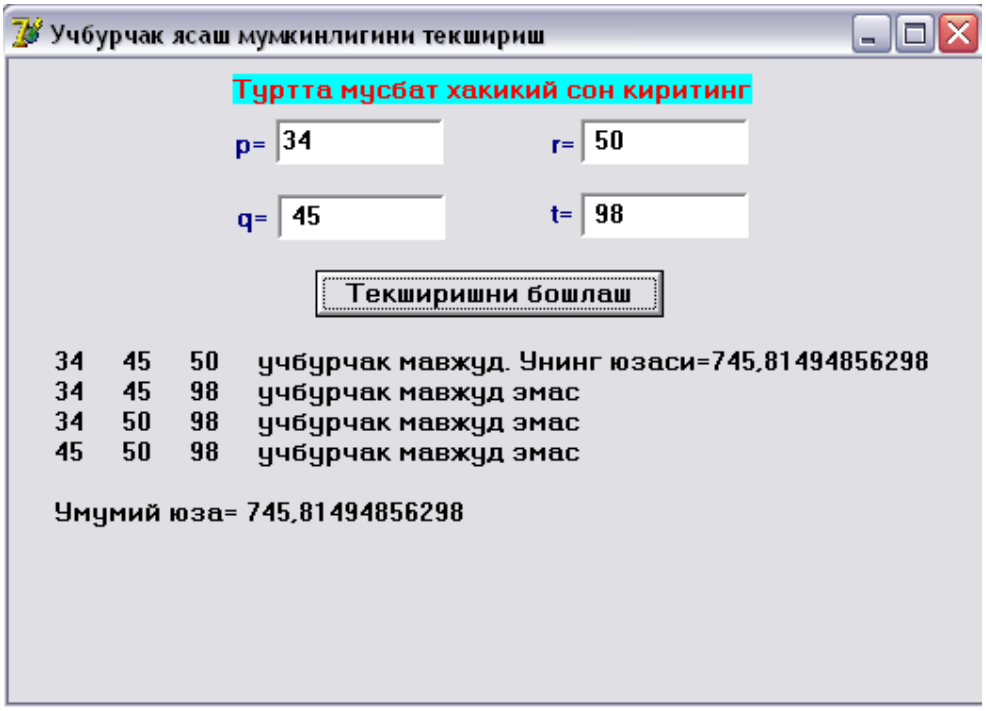
```
  ok : boolean;//ok=True-учбурчак мавжуд, ok=False – мавжуд эмас
p : real;//ярим периметр
begin
yuza := 0;  z := '';
if (a + b >= c) and (a + b >= c) and (b + c >= a)
then begin
          ok := true;
          p := (a + b + c)/2;
          yuza := sqrt(p * (p - a) * (p - b) * (p - c));
       end
      else ok := false;
      z := floattostr(a)+ '     ' + floattostr(b) + '     ' + floattostr(c);
      if ok then
      z := z + '     ' + 'учбурчак мавжуд. Унинг юзи =' + floattostr(yuza)
            else z := z + '     ' +'учбурчак мавжуд эмас';
     s := s + z + #13;
     um_yuza := um_yuza + yuza;
end;
begin
    s2 := '';
    p := strtofloat(edit1.text);
    q := strtofloat(edit2.text);
    r := strtofloat(edit3.text);
    t := strtofloat(edit4.text);
    uchburchak(p,q,r);
    uchburchak(p,q,t);
    uchburchak(p,r,t);
    uchburchak(q,r,t);
    s := s + #13 +'Умумий юза= ' + floattostr(um_yuza);
    label6.caption := s;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
um_yuza := 0;
s2 := '';
end;
end.
```

Дастурни ишга тушириш

Бу дастурининг диалог ойнаси 7.4-расмда келтирилган.

Эътибор берган бўлсангиз, асосий дастурдан процедурага тушадиган формал ўзгарувчилар процедура сарлавҳасида эълон қилинган. Аммо, процедурадан асосий дастурга қайтиши керак бўлган ўзгарувчилар кўрсатилмаган.

Улар дастур матнида, Button1Click ходисаларни қайта ишлаш процедураси матнида аввал кўрсатилган. Бу ўзгарурвчилар у***uza, um_yuza, s2*** ўзгарувчилари асосий дастур учун ҳам, процедура учун ҳам бир ҳил қилиб танланган. Бундай ҳолларда ўзгарувчиларни мисолдаги каби эълон қилиш ва фойдаланиш мумкин. ***um_yuza*** ва ***s2*** ўзгарувчилари бошланғич қийматларини форма фаоллаштирилганда, яъни FormCreate ходисаларни қайта ишлаш процедурасидан олади.



7.4-расм. Учбурчак ясаш ҳақидаги масаланинг диалог ойнаси

менюга

7.5. Модулларни яратиш ва фойдаланиш

Айрим процедуралардан кўплаб масалалар учун фойдаланиш мумкин. Бунинг учун хар доим ана шу процедураларни дастурлар таркибига киритишга тўғри келади. Фараз қилайлик, икки сондан каттаси ёки кичигини топиш учун процедура яратилган бўлса ва бу процедурадан фойдаланиб 10 та масала ечишга тўғри келса, 10 та дастур таркибига уларни киритиш лозим бўлади. Бунинг учун дастурчи ўзи қайта фойдаланмоқчи бўлган шу функция ёки процедура матнини **implementation** бўлимига жойлаши керак. Бунинг учун функция ёки процедура матнини янгидан компьютер хотирасига киритиши ёки бошқа дастур таркибидан унинг матни нусхасини олиб, янги дастур таркибига қўшиши лозим.

Дастурчи ўзи севган, ўзиниг иши учун кўплаб марта қўллашига тўғри келадиган процедуралар билан ишлаганда ана шундай вазият юзага келиши мумкин.

Шундай ҳолатларда дастурчи ўз ишини осонлаштириш учун процедуралардан алоҳида кутубхона ташкил этиши мумкин. Бу кутубхонага кирган функция ва процедуралар зарурат туғилганда қайтадан ёзилмайди, бошқа дастур таркибидан янги дастур матнига қўшилмайди. Бу процедура ва функциялардан тайёр маҳсулот сифатида фойдаланиши мумкин. Бунинг учун у ана шу процедура ва функциялардан модул деб аталадиган янги кутубхона яратиши лозим бўлади.

Модуль – бу процедура ва функцияларнинг алоҳида кутубхонасидир.

Delphi тили дастурчилар ихтиёрига жуда катта миқдордаги процедура ва функцияларни стандарт модулларга бирлаштирган ҳолда бериб қўйган. Дастурчи янги лойиха яратишни бошлаган вақтда, унинг формага қўшган объектлари, уларнинг ҳусусиятлари, формалар билан боғлиқ процедура ва функциялар жойлашган модуллар автоматик тарзда ишга туширади. Бу модуллар дстур матнида **uses** хизматчи сўзидан кейин кўрсатилади. Масалан,

uses Windows, Messages, SysUtils, Variants, Classes, Graphics ва х.к.

Шундан кейин, дастурчи рўйхатдаги модулларга кирган процедура ва функциялардан бемалол фойдаланиши мумкин. Бунинг учун дастурчи ўзига керакли процедура номини кўрсатса етарли.

Delphi дастурчиларга функция ва процедуралардан алоҳида, янги модул яратишига ҳамда кейинчалик эҳтиёжга қараб, улардан фойдаланиши учун имкон яратилган. Бунинг учун дастурчи ўзи фойдаланмоқчи бўлган процедураси жойланган модуль номини **uses** буйруғидан кейин келадиган модуллар рўйхатига қўшиб қўйиши керак бўлади.

Янги модул яратиш. Янги модул яратиш учун, дастурчи форма ва форма модули ойналарини

ёпиши керак. Сўнгра **File** менюсидан **New / Unit** тугмасини босиши лозим. Натижада кодлар муҳаррири ишга тушиб, экранда Delphi тилидаги янги модулни яратиш учун тайёр шаблон пайдо бўлади. Унинг кўриниши қуйидагича:

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

Бу ерда **unit** - хизматчи сўз, **unit1** – яратилаётган янги модулнинг номи. Бу ном модул матнини сақлашга буйруқ берилганда дастурчи қўйган ном билан автоматик тарзда алмаштириб қўйилади.

Interface сўзи модулнинг интерфейси бўлимини белгилаб беради. Бу бўлим ўз ичига янги модул таркибига кираётган процедура ва функцияларнинг сарлавҳалари рўйхатини олади.

Implementation (реализация) бўлимида интерфейс бўлимида кўрсатилган процедура ва функцияларнинг дастурлари матни жойланади.

Биз мисол тариқасида, IsInt ва IsFloat функциялари (бу функциялардан бутун ва каср сонларни киритишда фойдаланиш мумкин. Улар сонларни киритишда фақат рухсат берилган белгиларнинг киритилишини назорат қилади, бошқа белгилардан фойдаланишга рухсат бермайди)

Листинг 7.7. Янги модулга намуна

```
unit my__unit;
interface// процедура ва функциялар рўйхатини эълон қилади
    function IsInt(ch : char) : Boolean;
   // бу функция киритилаётган белги бутун сонларни ёзишда қатнашиши мумкинлигини аниқлайди
function IsFloat(ch : char; st: string) : Boolean;
// бу функция киритилаётган белги каср сонларни ёзишда қатнашиши мумкинлигини аниқлайди
// ch — навбатдаги белги
// st — хозиргача киритилган белгилар

implementation// реализация
function Islnt(ch : char) : Boolean;
begin
if (ch >= '0') and (ch <= '9')// рақамлар
or (ch = #13)// <Enter> клавишаси
or (ch = #8)// <Backspace> клавишаси
then Islnt := True// бу белгига рухсат бор
else Islnt := False;// бутун сонлар ичида бундай белги йўқ
end;

 function IsFloat(ch : char; st: string) : Boolean;
begin
if (ch >= '0') and (ch <= '9')// рақамлар
or (ch = #13)// <Enter> клавишаси
or (ch = #8)// <Backspace> клавишаси
then
begin
IsFloat := True;// бу белги мумкин
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// ажратувчи белгини фақат рақамдан кейин ёзиш мумкин (агар у аввал ёзилмаган бўлса)
IsFloat := True; else// қолган белгилар таъқиқланган
IsFloat := False;
end;
end.
```

Дастур матни одатдаги усул билан сақлаб қўйилади. Модулларни сақлаш учун алоҳида папка

ажратган маъқул. Масалан, унинг номи **Units** бўлсин.

Модуллардан фойдаланиш. Дастурда модулнинг функция ва процедураларидан фойдаланиш мумкин бўлиши учун, бу модул номини лойихага қўшиш ва уни модуллар рўйхатида кўрсатиб қўйиши лозим.

7.9-листингда **Дала ҳовлига бериб келиш қиймати** дастурининг матнининг янги модул қатнашган варианти келтирилган. OnKeyPress ходисаларни қайта ишлаш процедураси бошланғич маълумотлар учун киритиш IsFloat функциясига мурожаат қилади. Бу функция **yangi_modul** модулида жойлашган. Шунинг учун модуллар рўйхатида **yangi_modul** номи қатнашмоқда.

Листинг 7.8. Янги модулдаги функциядан фойдаланиш

```
unit dala1;
interface
uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  Dialogs, StdCtrls,
yangi_modul;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Label2: TLabel;
    Edit2: TEdit;
    Label3: TLabel;
    Edit3: TEdit;
    CheckBox1: TCheckBox;
    Label4: TLabel;
    Button1: TButton;
    procedure EditlKeyPress(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}
//тугмани МАСОФА ойнасида босиш
procedure TForm1.EditlKeyPress(Sender: TObject; var Key: Char);
begin
if Key = Char(VK_RETURN)
then Edit2.SetFocus//курсорни НАРХ ойнасига ўтказиш
else
If not IsFloat(Key,Edit2.Text) then Key := Chr(0);
end;

//НАРХ майдонида тугмани босиш
procedure TForm1.Edit2KeyPress(Sender:TObject; var Key: Char);
begin
if Key = Char(VK_RETURN)
then Edit3.SetFocus//курсорни ҲАРАЖАТ майдонига ўтказиш
else If not IsFloat(Key,Edit2.Text)
then Key := Chr (0);
end;

  //тугмани  ҲАРАЖАТ майдонида босиш
procedure TForm1.Edit3KeyPress(Sender:TObject; var Key: Char);
```

```
begin
if Key = Char(VK_RETURN)
then Button1.SetFocus//ҲИСОБЛАШ тугмасини  фаоллаштирилмоқда
else If not IsFloat(Key,Edit2.Text) then Key := Chr (0);
end;

//ҲИСОБЛАШ тугмаси босилганда
procedure TForm1.Button1Click(Sender: TObject);
var
rast : real;//масофа
cena : real;//нарх
potr : real;//100  км га сарф
summ : real;//сумма
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 := 'Дала ҳовлига бориш';
if CheckBox1.Checked then mes := mes = ' ва қайтиб келиш' + #13;
mes := mes/FloatToStrF(summ,ffGeneral,10,2) + ' сўмга тушади';
Label4.Caption := mes;
end;
end.
```

Дастурни ишга тушириш

Модул номини иловада фойдаланилаётган модуллар рўйхатига қўшилгандан сўнг, модулнинг ўзини лойихага қўшиб қўйиш лозим. Бунинг учун **Project** менюсидан **Add to Project** буйруғини танлаш хамда очилган диалог ойнасидан модул файли номини танланади. натижада муҳаррир ойнасида лойихага қўшилган модул матни пайдо бўлади.

Лойиха структурасини **Project Manager** ойнасида кўриш мумкин. Уни **View** менюсидан **Project Manager** буйруғини ёрдамида экранга чақириш мумкин. Мисол тариқасида **Дала ҳовлига бориб келиш қиймати** масаласининг структураси 7.5-расмда келтирилган.
Модул матнини лойихага қўшиб, унинг номини лойихада фойдаланилаётган модуллар рўйхатига (**uses** сўзидан кейин) қўшилганидан сўнг, дастур матнини компиляция қилиш мумкин.

8-БОБ. ФАЙЛЛАР БИЛАН ИШЛАШ

8.1. Бошланғич маълумотлар

Яқин йилларгача ЭҲМ лар фақат сонли маълумотларни қайта ишлашга мўлжалланган эди. Хозирга келиб, ЭҲМ халқ хўжалигининг деярли ҳамма соҳаларини қамраб олди. Шунинг учун ҳам хар бир соҳадаги маълумотларни ўқиш, сақлаш, қайта ишлаш ҳамда бошқа маълумот истеъмолчиларига узатиш каби энг муҳим ва долзарб масалалар замонавий ЭҲМ лар зиммасига юкланмоқда. Халк хўжалигининг бошқарув системалари, алоқа воситалари каби кўплаб соҳаларида ЭҲМ лар асосан маълумотларни сақлаш воситаси сифатида қўлланмоқда. Узоқ муддат сақлашга тўғри келадиган маълумотлар махсус воситаларда, магнит дискларида, магнит ленталарида файллар кўринишида сақланади.

Файл деб, хотира қурилмаларидан бирида сақланаётган (маълум бир ҳажмни эгаллаган) ва ўзининг номига эга бўлган маълумотлар тўпламига айтилади.

Файл бўш бўлиши ҳам мумкин. Файллар маълумот сақлашнинг энг қулай усули эканлигининг сабаби қуйидагилардан иборат:
1) Одатда дастурни бажариб, олинган натижалар дастур ўз ишиии тугатгандан сўнг, ЭҲМ хотирасидан ўчиб кетади. Бу маълумотлар яна зарур бўлса, дастурни янгидан ишга туширишга тўғри келади. Буни олдини олиш учун олинган

натижаларни файлларга ёзиб қўйилиши мумкин;.

2) Битта файлда сақланаётган маълумотлар кўплаб масалалар (дастурлар) учун асос бўлиш мумкин, яъни битта дастур натижалари сақлаб қўйилса, бу маълумотлардан фойдаланиб бошқа масалаларни ечиш мумкин;

3) Маълумотлар сони ЭҲМ нинг оператив хотирасига сиғмайдиган даражада кўп бўлиши мумкин. Бундай вақтда маълумотларнинг бир қисмини бирор файлда вақтинча сақлаб қўйиш мумкин;

4) Файллардан улардаги маълумотлар доирасидаги ихтиёрий мақсадлар ва масалалар учун фойдаланиш мумкин.

Файллар ўзининг манзили ҳамда номига эга бўлади. Файлнинг номи одатда иккита қисмдан иборат бўлиши мумкин: ном ва кенгайтма. Масалан:

D:/DELPHI/alomat.pas

ёзуви **alomat.pas** файлини англатади. бу ерда **alomat**- файлнинг номи, **.pas**-эса унинг кенгайтмаси. Бу файлнинг манзили - **D** дискдаги **DELPHI** папкаси.

Дастурчи файллар ишини ташкил қилар экан, фақат дастур ва унинг натижаси ҳақида қайғурибгина қолмасдан, балки кўплаб қўшимча дастурлар ёрдамида файлларни яратиш, файлда сақланаётган маълумотларни бошқариш, таҳлил қилиш, тартиблаш, эҳтиёжга қараб дисплей ёки қоғозда акслантириш каби масалаларни ҳам хал қилиши керак. Яна, илгари кўзда тутилмаган янги заруратлар учун қўшимча дастурлар яратиш ҳақида ҳам ўйлаши керак.

Delphi да файллар деб, ЭҲМ да сақланаётган бир хил типга мансуб бўлган маълумотлар (компоненталар) тўпламига айтилади.

Файлдаги маълумотлардан фойдаланиш учун уларни ўқилади ва ўқилган маълумотларни ўзгарувчиларга қиймат қилиб берилади.

Ихтиёрий вақтда файлнинг фақат битта компонентаси билан ишлаш мумкин халос. Бу маълумотни кўрсаткич (курсор) кўрсатиб туради. Кўрсаткич биринчи компонентадан бошлаб, хар бир маълумот ўқилгандан кейин, навбатдаги ўқиш керак бўлган маълумотни кўрсатиб туради. (Бошланғич синфлардаги хатчўпларни эслаб кўринг.)

Файлдаги маълумотлар сони ўзгариб туради, у дастлаб нолга тенг, кейинчалик фақат файлга янги маълумотлар қўшилганда ўсиши ёки ўчирилганда нолтагача камайиши мумкин. Янги маълумотлар доим файлнинг охирига қўшилади.

менюга

8.2 Файлли типлар.

Файл — бу номланган структурали маълумотлар бўлиб, улар бир ҳил типдаги элементлардан (компоненталардан) иборат. Бу маълумотлар сони амалий жиҳатдан компьютер хотирасининг бўш қисми билан чегараланган. Бошқача айтганда, файл – бу чексиз сондаги маълумотлардан иборат массивдир.

Барча структурали маълумотлар (ўзгарувчилар, массивлар) каби файлларни ҳам ўзгарувчиларни эълон қилиш бўлимида эълон қилиниши керак. Бу эълонда файл элементларининг типи кўрсатилади.

Файлли типлар умумий кўринишда қуйидагича эълон қилинади:

Ном : file of тип;

Бу ерда ***Ном***- файлли ўзгарувчининг номи; ***file of*** - файлли ўзгарувчи эълон қилинаётганлигини билдирувчи хизматчи сўз; ***тип*** – шу файлда сақланаётган маълумотларнинг типи. Масалан:

Bel : file of char;// белгли маълумотлар файли
koef: **file of** real;// ҳақиқий сонлар файли
f: file of integer;// бутун сонли файл

Барча компоненталари белгили типда бўлган файлни белгили ёки матнли файл деб аталади. Улар қуйидагича эълон илинади:

Ном : TextFile;

бу ерда T***extFile*** — ***Ном*** ўзгарувчиси белгилаб турган файл матнли файл эканлиги англатади.

Файлли ўзгарувчини эълон қилиш бу шунчаки файл компоненталарининг типини кўрсатади халос. Дастур файлга маълумотларни киритиши ёки файлдан маълумотларни ўқиши учун бу ўзгарувчини бирор файл билан боғлаш лозим. Бунинг учун ана шу файлнинг манзили ва номини кўрсатиш талаб қилинади.

Файлнинг номи AssignFile процедураси билан кўрсатилади. Бу процедура файлли ўзгарувчини аниқ бир файл билан боғлайди. Умумий ҳолда AssignFile процедураси қуйидагича ёзилади:

AssignFile(var f, файл номи : string)

Файлнинг номи Windows учун қабул қилинган қоидалар ёрдамида ёзилади. У ўз ичига файлнинг манзили, яъни диск номи, каталоглар, қисм каталогларни ҳамда файлнинг номини олиши мумкин. Масалан:

AssignFile(f, 'c:/kv_tenglama.Bas');

AssignFile(g, '/DELPHI/DOC/KITOB.doc');

fname:=('Хисоб.txt'); AssignFile(h,fname);

бу ерда 1-эълонда f – файлли ўзгарувчи C дискдаги kv_tenglama.Bas файлини, 2-эълонда g- файлли ўзгарувчи жорий дискнинг DELPHI каталогининг DOC қисм каталогидаги KITOB.doc файлини, 3-эълонда эса h – файлли ўзгарувчи fname – ўзгарувчиси билан белгиланган Хисоб.txt файлини англатмоқда. Бу эълонлардан кейин шу файлли ўзгарувчиларга қандай амал юзасидан мурожаат қилинса, компьютер бу амалларни шу файлли ўзгарувчилар кўрсатиб турган файллар устида бажаради.

менюга

8.3. Файлларни очиш ва ёпиш. Маълумотлар киритиш

Файлни очиш. Файлга маълумотларни ёзиш ёки маълумотларни ўқиш имконига эга бўлиш учун аввал бу файлни очиш керак бўлади.

Агар бошланғич файлни ташкил қилувчи дастурдан илгари фойдаланилган бўлса, дастурнинг иши натижаси бўлган файл хотира қурилмаларидан бирида мавжуд бўлиши мумкин. Шунинг учун дастурчи, бу дастурни янгидан ишга тушириш лозим бўлган ҳолларда эски файлни нима қилиш кераклигини ҳал қилиши зарур: эски файлни ўчириб, унинг ўрнига янги файл яратиш керакми ёки эски файлдаги маълумотларнинг давомига янги маълумотларни қўшиш керакми?

Файлнинг эски вариантидан фойдаланиш усуллари файлни очиш вақтида аниқланади.

Файлларни уч ҳил мақсаддан бири учун очиш мумкин:

- Файлни маълумотларни янгидан ёзиш учун;
- Мавжуд файлнинг давомига янги маълумотларни қўшиш учун:
- Файлдаги маълумотларни ўқиш учун.

Файлни янги файл яратиш режимида очиш учун

Rewrite(f);

кўринишидаги буйруқдан фойдаланилади. бу ерда ***f*** — TextFile типидаги файлли ўзгарувчи. Агар жорий каталогда f – файли кўрсатиб турган номдаги бошқа файл мавжуд бўлса, бу файл ўчирилиб, унинг ўрнига янги файл ташкил қилинади. Буни ҳаётда эски дафтарни ташлаб юбориб, (дафтар ҳам қандайдир маънода файлни англатади) унинг ўрнига янги дафтар тутилганига қиёслаш мумкин.

Файлни ундаги маълумотларнинг давомига янги маълумотларни қўшиш режимида очиш учун

Append (f);

процедурасидан фойдаланилади. бу ерда f — TextFile типидаги файлли ўзгарувчи. Маълумотлар ёзишнинг бу усулида, файл очилганидан сўнг, маълумот ёзиладиган позицияни кўрсатувчи курсор файлдаги охирги маълумот ёзилган жойдан кейинги позицияни кўрсатади ва ёзиладиган маълумот шу позициядан бошлаб файлга киритилади. Бу ерда юқорида айтилган дафтарни келган еридан бошлаб ёзишга тенглаш мумкин.

Файлга одатда, маълумотларни қачондир ва қандайдир мақсадда фойдаланиш учун ёзиб қўйилади. Сақлаб қўйилган бу маълумотлардан фойдаланиш имконига эга бўлиш учун, дастлаб бу маълумотларни ўқиш лозим бўлади. Файлни ундаги маълумотларни ўқиш мақсадида очиш учун

Reset(f);

буйруғидан фойдаланилади. Бу ерда ***f*** – файлли ўзгарувчи. Бу жараённи ҳаётдаги маълумотлари, яъни ёзувларини (мисол учун конспектларни) ўқиш учун очилаётган дафтар деб қараш мумкин.

Файлга маълумотларни ёзиш. Файлга маълумотларни киритиш (ёзиш) учун ***write*** ёки ***writeln***. Бу буйруқ умумий кўринишда қуйидагича ёзилади:

write (Файлли ўзгарувчия, рўйхат) ;
writeln (Файлли ўзгарувчи, рўйхат);

бу ерда ***Файлли ўзгарувчи*** — маълумотлар киритиладиган файлни кўрсатувчи файлли ўзгарувчи; ***Рўйхат*** – бир-биридан вергул билан ажратилган ва қийматлари файлга ёзилиши талаб қилинган ўзгарувчининг рўйхати. Бу рўйхатга ўзгарувчилардан ташқари сатрли константаларни киритиш мумкин..

Масалан, агар ***f*** - ўзгарувчи TextFile типидаги ўзгарувчи бўлса, у ҳолда x1 ва x2 - ўзгарувчиларнинг қийматларини файлга ёзиш буйруғи қуйидагича бўлиши мумкин:

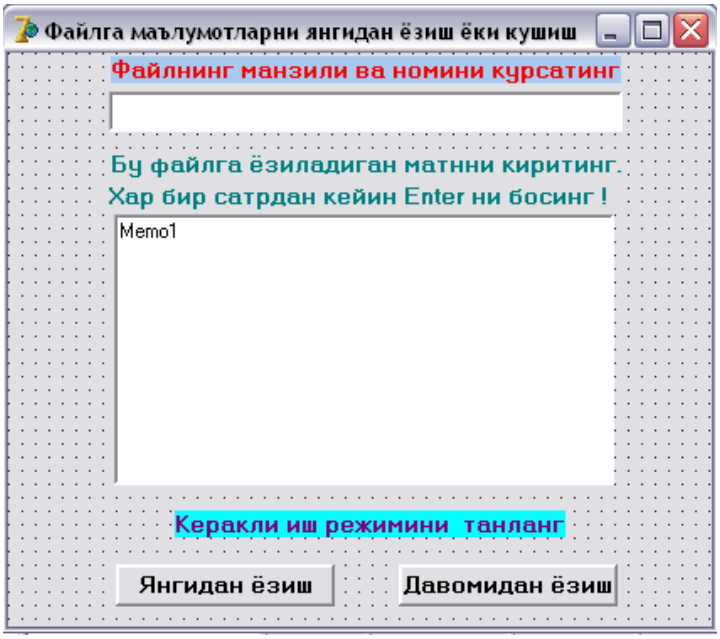
write(f, 'Тенгламанинг илдизлари', xl, x2);

Write ва ***writeln*** буйруқларининг фарқи шу ердаки, ***writeln*** буйруғи файлга рўйхатда кўрсатилган барча қийматларни ёзиб бўлганидлан сўнг, курсорни янги сатрнинг бошига ўтказади. Демак, бу файлга киритиладиган навбатдаги маълумот курсор турган ердан, яъни янги сатрнинг бошидан ёзилади. ***Write*** эса маълумотларни ёзиб, келган ерида курсорни тўхтатиб қўяди. Бу файлга ёзиладиган навбатдаги маълумот шу ерда бошлаб ёзилади.

Файлни ёпиш. Дастур ўз ишини тугатмасдан туриб, жорий вақтда очилган барча файлларни ёпиш лозим. Бу амал ***close*** процедураси ёрдамида бажарилади. ***Close*** процедураси битта параметрга эга. Бу параметр файлли ўзгарувчининг номидан иборат бўлади. Масалан:

Close(f).

8.1-расмда кўрсатиладиган номдаги матнли файлга янги маълумотларни ёзиш ёки қўшиш амалларидан бирини бажарадиган дастур матни келтирилган.



8.1-расм. Файлга янгидан ёзиш ёки қўшиш дастурининг диалог ойнаси

8.1-листингда **Янгидан ёзиш** тугмасининг ходисаларни қайта ишлаш процедураси келтирилган. У файлни янги файл яратиш (агар кўрсатилган номдаги файл жорий каталогда мавжуд бўлса, эскисини ўчириб, ўрнига янгисини яратиш) режимида очади ва Memo1 ойнасидани матнни шу файлга ёзади.

Файлнинг номини Editl майдонидаа кўрсатилади. Илова формасини яратиш жараёнида файлнинг номини қайта аниқлаш имконияти ҳам мавжуд. Бунинг учун

Edit1.text := 'C:/DELPHI/test.txt';

тарзидаги буйруқлардан фойдаланиш мумкин.

8.1-л**истинг. Файлни янгидан яратиш ёки эскисини алмаштириш**

```
procedure TForm1.Button1Click(Sender: TObject);
var
f: TextFile;//файл
fName: String[80];//файлнинг номи
i: integer;
begin
fName := Edit1.Text;
AssignFile(f, fName);
Rewrite(f);//файлни янгидан, ёзиш учун очмоқда
//Файлга ёзиш
```

```
for i := 0 to Memo1.Lines.Count do
//Memo да сатр  номери 0 дан бошланади
writeln(f, Memo1.Lines[i]);
CloseFile(f);//файлни ёпиш
MessageDlg('Маълумотлар  қуйидаги файлга ёзилди: ' +#13 + Edit1.text, mtInformation,[mbOk],0);
end;
```

8.2-листингда **Давомидан ёзиш** тугмаси учун ёзилган процедуранинг матни келтирилмоқда. У номи Edit1 майдонида кўрсатилган файлни очиб, ундаги мавжуд маълумотларнинг давомидан Memol майдонида кўрсатилган маълумотларни ёзади.

8.2-листинг. Мавжуд файлнинг давомида ёзиш.

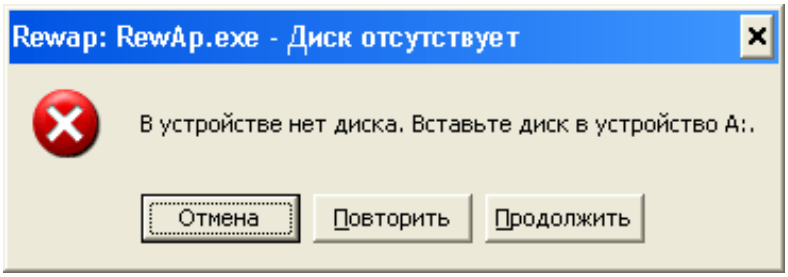
```
procedure TForm1.Button2Click(Sender: TObject);
var
f: TextFile;//файл
fName: String[80];// файлнинг номи
i: integer; begin
fName := Edit1.Text;
AssignFile(f, fName);
Append(f);//файлни давомидан ёзиш учун очиш
//файлга ёзиш
for I := 0 to Memo1.Lines.Count do
//Memo да сатр  номери 0 дан бошланади
writeln(f, Memo1.Lines[i]);
CloseFile(f);//файлни ёпиш
MessageDlg('Маълумотлар қуйидаги файлга қўшилди:  ' + #13 + Edit1.text, mtInformation,[mbOk],0);
end;
```

менюга

8.4. Файлларни очишдаги хатоликлар

Апйрим ҳолларда файлларни очишга уринишда бажариш вақтидаги хатоликлар юзага келиши мумкин. Бунинг сабаблари бир нечта бўлиши мумкин. Масалан, диск юритувчи ҳали ишга тайёр бўлиб улгурмасдан (эҳтимол диск қўйилмагандир ёки диск юритувчи ёпилмагандир), дискдаги файлни очишга ҳаракат қилиниши мумкин. Яна бир кўп учрайдиган ҳатолик – мавжуд бўлмаган файлни очишдир (йўқ файлга маълумотлар қўшимча равишда ёзиш).

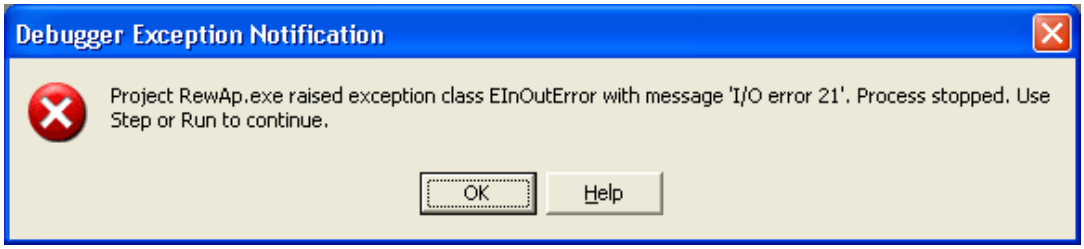
Дастурларни Delphi дан туриб ишга туширишда файлларни очишдаги ҳатоликлар юзага келиб қолса, экранда бу ҳақда ахборот берадиган диалог ойнаси пайдо бўлади (8.2-расм):



8.2-расм. Файлни олишдаги ҳатоликка мисол.
Дастур Delphi дан ишга туширилган)

Агар дастур Windows дан ишга туширилган бўлса, бу ҳатолик ҳақидаги ахборот бошқача кўринишда бўлади (8.3-расм).

Дастур файлларни очиш вақтидаги ҳатоликларни назорат қилишни ўз зиммасига олиши мумкин. Буни IOResult (Input-Output Result — киритиш-чиқариш амалининг натижаси) функциясининг



8.3-расм. Файлни олишдаги ҳатоликка мисол.
Дастур Windows дан ишга туширилган)

қийматини текшириш орқали бажариш мумкин. IOResult функцияси 0 га тенг бўлади, агар киритиш-чиқариш амали тўғри бажарилган бўлса, акс ҳолда бу қиймат ҳатолик кодига тенг бўлади.

Дастур киритиш-чиқариш амали натижасини назорат қила олиши учун файлни очиш буйруғидан олдин компиляторга {$I-} кўрсатмасини бериб қўйиш керак. Бу кўрсатма киритиш-чиқариш амаллари билан боғлиқ ҳатоликларни қайта ишлаш компиляторга таъқиқлайди ва бу соҳадаги бошқаришни дастур зиммасига олганлиги ҳақида ахборот беради. {$I-} кўрсатмаси киритиш-чиқариш амалларини бажаришда юзага келиши мумкин бўлган ҳатоликларни автоматик тарзда кузатиб бориш режимини қайта тиклайди.

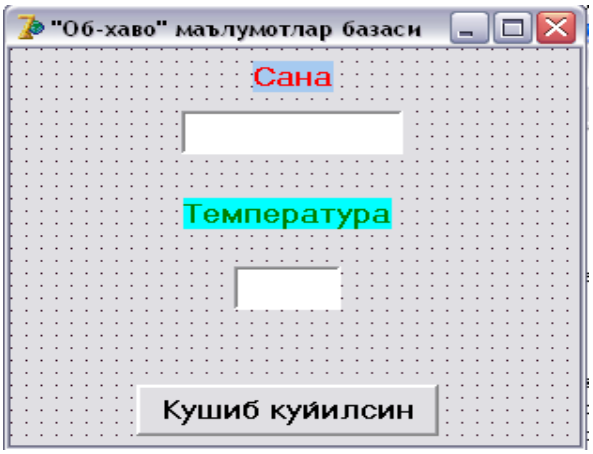
8.4-расмда мавжуд бўлмаган файлни қўшимча маълумотлар ёзиш режимида очишга уриниш бўлганда янги файл яратиш режимида (шу билан ҳатолик бартараф қилинади) очишни таъминлайдиган дастурнинг алгоритми келтирилган.

қадам	Буйруқ
...	... ...
N	Файл давомида ёзиш режимида очилсин (Append)
N+1	Агар бу буйруқ ҳатолик билан бажарилса файлни янги файл (Rewrite) режимида очилсин
...	... ...

Юқоридаги алгоритмга мос келадиган дастур парчаси қуйидагича ёзилади:

```
AssignFile(f,filename);
{$I-}
Append(f)//Файлни давомида ёзиш учун очилмоқда
{$I+}
if IOResult<> 0//Очишда ҳатолик юзага келса
then Rewrite(f);//файлни янги файл режимида очиш
//Натижада мавжуд бўлган ёки янги  файл очилади.
```

Намуна. Қуйидаги дастур содда маълумотлар базаси бўлган файлни яратиш ва тўлдириш учун мўлжалланган. У ҳар гал ишга тушганда экранда унинг диалог ойнаси пайдо бўлади (8.4-расм). Бу диалог ойнасининг махсус ойналарига фойдаланувчи сана ҳамда шу кунги ҳавонинг температурасини киритиши лозим. Бу дастурда Edit1 майдонига сана, Edit2 майдонига температура ёзилади. **Қўшиб қўйилсин** тугмаси босилганда бу маълумот файлга қўшиб қўйилади.



8.4-расм. "Об-ҳаво" маълумотлар базасининг диалог ойнаси

8.3-листинг. Содда маълумотлар базасига мисол. (файлга ёзиш)

```
unit ob_havo;
interface
uses   Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,   Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Label2: TLabel;
    Edit2: TEdit;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;

implementation
{$R *.dfm}
const
DBNAME = 'c:⁄ob_havo. txt';
var
db:  TextFile;⁄⁄файл-маълумотлар базаси
procedure TForm1.Button1Click(Sender: TObject);
begin
if (Length(edit1.text)=0) or (Length(edit2.text)=0)
then ShowMessage('Маълумотларни  киритишда ҳатолик мавжуд. '
+ #13 + 'Ҳамма майдонлар тўлдирилиши шарт.')
else writeln(db, edit1.text,' ',edit2.text);
end;
⁄⁄OnClose ҳодисаси формани ёпишда содир бўлади
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
CloseFile(db);⁄⁄маълумотлар базасини ёпиш
end;
procedure TForm1.Formactivate(Sender: TObject);
begin
AssignFile(db, DBNAME); {$I-}
Append(db);
If  IOResult = 0 then begin
Edit1.Text := DateToStr(Date);⁄⁄Жорий санани киритиш
Edit2.SetFocus;⁄⁄курсорни Edit2 майдонига ўтказиш
end
else begin
Rewrite(db);
if IOResult <> 0 then begin
⁄⁄Киритиш тугмаси ва буйруқли тугмаларни таъқиқлайди
Edit1.Enabled := False; Edit2.Enabled := False;
Button1.Enabled := False;
ShowMessage(DBNAME + ' файлни яратишда ҳатолик мавжуд ');
end;  end;
end;
end.
```

Дастурни ишга тушириш

Маълумотлар базаси файлини FormActivate процедураси очади. Бу ходиса илова формаси активлашганда рўй беради. Шунинг учун процедура автоматик тарзда ишга тушади. Агар файлни очиш амали мувафаққиятили бажарилса, Edit1 майдонига жорий сана ёзилади. Жорий сана ҳақидаги маълумотни Date функцияси аниқлайди. Date функциясининг қийматини (Double типидаги сонни) кўриш учун қулай бўлган кўринишда келтириш учун Datetostr функциясидан фойдаланилади. Санани Edit1 майдонига киритилганидан кейин onActivate ходисаларни қайта ишлаш процедураси setFocus методи ёрдамида курсорни температура майдонига ўтказади. Агар файлни очиш ёки янги файлни яратиш жараёнида ҳатолик юзага келса, у ҳолда дастур **Қўшиб қўйилсин** тугмасини бекор қилади ва бу ҳақдаги ахборотни экранга чиқаради.

TForm1.Button1Click процедураси (onclick ходисани қайта ишлаш процедураси) **Қўшиб қўйилсин** тугмасининг босилиши билан ишга тушади. Натижада киритилган маълумотлар obhavo.txt файлига ёзиб қўяди. Бундан аввал дастур ҳамма майдонлар тўлдирилганлигини текширади. Агар майдонларнинг бирортасига маълумот киритилмаган бўлса, бу хақдаги маҳсус ахборот экранга чиқарилади.

Дастурнинг ишлаши жараёнида obhavo.txt файлининг охирига янги киритилган маълумотлар, яъни сана ва температура ёзиб қўйилади.

Бу дастурда write эмас, балки маълумотлар базасида хар бир санадаги маълумот алоҳида сатрда жойлашиши учун writeln буйруғидан фойдаланилган. Бу буйруқ учта элементдан иборат маълумотни файлга ёзади. Сана, яъни Edit1.text файлга ёзилганидан кейин, "бўш жой" белгиси, сўнгра температура - edit2.txt ёзилади. Агар "бўш жой" белгисини файлга қўшилмаса, у ҳода сана ва температура файлга "ёпиштириб" ёзилган, яъни иккита маълумот бирлашиб, рақамлар кетма-кетлигидан иборат бўлиб қолар эди.

Маълумотлар базасини ёпиш амалини Tform1.Formclose процедураси бажаради. У илова формасини ёпишда юзага келадиган onclose ходисасини қайта ишлайди.

Ушбу дастур бир неча марта ишга туширилганидан кейин, obhavo.txt файли куйидагича бўлиб қолиши мумкин:

9.05.2001 10
10.05.2001 12
11.05.2001 10
12.05.2001 7

менюга

8.5. Маълумотларни файлдан киритиш

Биз аввалги пунктда маълумотларни файлларда сақлаш усуллари билан танишдик. Ушбу пунктда эса ана шу маълумотлардан қандай қилиб фойдаланиш мумкинлигини кўрамиз.

Маълумотларни фақат клавиатурадангина эмас, балки матнли файллардан ҳам киритиш мумкин. Бундай имкониятдан фойдаланиш учун ***TextFile*** типидаги ўзгарувчиларни эълон қилиш лозим. Бу ўзгарувчиларга ***AssignFile*** буйруғи ёрдамида маълумотлар олинадиган файлни бириктирилади. Шундан кейин бу файли маълумотларни ўқиш учун очилади, сўнгра ***read*** ёки ***readln*** процедуралари ёрдамида маълумотларни ўқиб, ўзгарувчиларга қиймат қилиб берилади. Шундан кейингина бу маълумотларни қайта ишлаш мумкин.

Файлни ўқиш учун очиш. Файлларни маълумотларни ўқиш (киритиш) учун очишда ***Reset*** процедурасидан фойдаланилади. Бу процедура битта файлли ўзгарувчили параметрга эга. ***Reset*** процедурасини чақиришдан аввал, ***AssignFile*** процедураси ёрдамида файлли ўзгарувчини аниқ бир файл билан боғлаш лозим. Масалан, қуйидаги буйруқлар файлни киритиш мақсадида очади:

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

Агар файлнинг номи нотўғри кўрсатилган бўлса (кўрсатилган файл дискда мавжуд бўлмаслиги мумкин), у ҳолда бажариш вақтидаги ҳатолик юзага келади. Бундай ҳатолик диск юритувчи ишга тайёр бўлмаган ҳолда хам содир этилиши мумкин.

Шунинг учун дастурда файлни очиш амалини такроран бажаришга рухсат берилса, қайтадан очишга уриниб кўриш мумкин.

Ҳудди файлларни очишдаги каби, дастур хатоликларни қайта ишлаш жараёнини ўз зиммасига олиши мумкин. Бунинг учун IOResult функциясининг қийматини текшириш лозим. .

Матни 8.4-листингда келтирилаётган дастур парчаси IOResult функциясининг қиймати файлни очиш буйруғининг натижасини текшириш учун фойдаланилмоқда. Агар файлни очишга уриниш кутилган натижани бермаса, у ҳолда дастур вужудга келган хатолик ҳақида ахборот беради ва бу буйруқни такроран бажариш учун фойдаланувчидан рухсат сўрайди.

8.4-листинг. Файлни очиш хатолигининг назорати (дастур парчаси)

```
var
fname : string[80];//Файлнинг номи
f : TextFile;//файл
nat : integer;
//файлни очишдаги хатолик коди ( IOResult нинг қиймати)
answ : word;//фойдаланувчининг жавоби
begin
fname := 'C:/test.txt';
AssignFile (f, fname);
repeat
{$I-}
Reset(f);//файлни ўқиш учун очиш
{$I+}
Nat := IOResult;
if Nat <> 0
then answ:+MessageDlg( fname+' файлини очишда хатолик бўлди' +#13 +'Яна бир марта уриниб кўрайми? ',
mtWarning,
[mbYes, mbNo],0);
until (Nat = 0) OR (answ = mrNo);
if res <> 0
then exit;//Процедура ишини якунлаш
//бу ерда файл тўғри очилса, бажарилиши керак бўлган буйруқлар
end;
```

Маълумотларни файлдан ўқиш. Бу амал ***read*** ва ***readln*** буйруқлари ёрдамида бажароилади ва умумий кўринишда қуйидагича ёзилади:

read(Файлли Ўзгарувчи, Ўзгарувчилар рўйхати);
readln(Файлли Ўзгарувчи, Ўзгарувчилар рўйхати) ;

бу ерда ***Файлли Ўзгарувчи*** — TextFile типидаги ўзгарувчи; ***Ўзгарувчилар рўйхати*** — бир-биридан вергул билан ажратилган ўзгарувчиларнинг номлари.

Файлдан сонларни ўқиш. Файл очилган пайтда кўрсаткич ундаги биринчи маълумотни кўрсатиб туради. *Read* буйруғи билан ана шу кўрсаткич кўрсатиб турган маълумот ўқилади, сўнгра кўрсаткич навбатдаги маълумотга ўтади. Файлли ўзгарувчининг типи билан ундаги маълумотларни қиймат қилиб оладиган ўзгарувчиларнинг типлари бир хил бўлиши шарт.

Шуни ёдда тутиш керакки, матнли файлда сонлар эмас, балки уларнинг тасвири сақланади. ***Read*** ёки ***readln*** процедураларининг қиладиган иши амалда иккита қисмдан иборат: дастлаб ажратувчи белгигача ("бўш жой белгиси ёки сатрнинг охири) бўлган белгиларни ўқилади, сўнгра ўқилган сонларнинг тасвири бўлган белгилар кетма-кетлиги сонга айлантирилади ва ҳосил қилинган қиймат ***Read*** ёки ***Readln*** процедураларининг параметри бўлган ўзгарувчиларга берилади. Масалан, агар C:/data. txt матнли файли қуйидаги маълумотларни ўз ичига олган бўлса,

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

У ҳолда

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

буйруқларининг бажарилиши натижасида ўзгарувчилар қуйидаги қийматларни олади:

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

Readln буйруғининг ***read*** дан фарқи шу ердаки, навбатдаги сонни ўқиб, олинган қийматни рўйхатдаги охирги ўзгарувчига қиймат қилиб берилганидан сўнг, жорий сатрда ўқиладиган маълумотлар тугамаган бўлса ҳам, файлдан ўқиш кўрсаткичи автоматик тарзда янги сатрнинг биринчи белгисига ўтказиб қўйилади. Шунинг учун

AssignFile(f,'C:/data.txt'); Reset(f);
readln(f, a); readln(f, b, c); readln(f, d);

буйруқлари бажарилганидан сўнг, ўзгарувчиларнинг қийматлари қуйидагича бўлади:

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

Агар файлдан сонли ўзгарувчилар учун ўқилган маълумотлар ичида рақам бўлмаган белгилар кетма-кетлиги мавжуд бўлса, ҳатолик рўй беради.

Сатрларни ўқиш. Дастурда сатрли ўзгарувчи эълон қилинганида, унинг узунлигини кўрсатилиши ёки кўрсатилмаслиги мумкин. Масалан:

Satr1:**string**[10]; satr2**:string;**

Файлдан белгилари сони аниқ кўрсатилган сатрли ўзгарувчига унинг эълонида қанча бўлса, шунча белги (аммо жорий сатрдагидан кўп бўлмаган) ўқилади.

Файлдан узунлиги очиқ кўрсатилмаган сатрли ўзгарувчиларга қиймат ўқишда, бу ўзгарувчининг қиймати жорий сатрнинг аввалги ўқилганидан қолган қисмига тенг бўлади. Бошқача айтганда, агар файлдан сатрни тўлалигича ўқимоқчи бўлсангиз, узунлиги файлдаги энг узун сатрдан каттароқ бўлган сатрли ўзгарувчи эълон қилинг ҳамда бу ўзгарувчига қийматни ўқинг.

Агар битта ***readln*** билан бир нечта, масалан иккита ўзгарувчига қиймат бермоқчи бўлсангиз, у ҳолда биринчи ўзгарувчи ўзининг эълонида қанча белги кўрсатилган бўлса, шунча белгини қиймат қилиб олади (узунлиги кўрсатилмаса, у ҳолда бутун сатрни тўлалигича олади). Иккинчи ўзгарувчининг қиймати шу сатрнинг қолган белгиларидан иборат бўлади, агар бундай белгилар бўлмаса, битта ҳам белгини олмайди (унинг узунлиги 0 га тенг). Масалан, матнли dustlar.txt файлида

Исмоилова Муқаддас
Абдуллаев Карим
Саматова Гулноза

сатрлари сақланаётган бўлсин.

8.1-жадвалда ўзгарувчиларни эълон қилишнинг бир нечта вариантлари, dustlar.txt файлидан ўқиш буйруқлари ва файлдан ўқиш натижасида ўзгарувчиларнинг олган қийматлари келтирилган.

Файлдан сатрларни ўқиш намуналари 8.1-жадвал

Ўзгарувчини эълон қилиш	Файлдан ўқиш буйруқлари	Файлдан ўқилганидан кейин ўзгарувчилар олган қийматлар
fam: string[15]; name: string[10]	Readln (f, fam, name)	fam= 'Исмоилова' name= 'Муқаддас'
fam, name: string;	Readln (f, fam, name)	Fam= 'Исмоилова Муқаддас' name= ' '
dust: string[80]	Readln (f, drug)	Dust =' Исмоилова Муқаддас'

Файлнинг охири. Файллар билан ишлаган пайтда кўпинча шу файлдаги барча маълумотларни кўриб чиқишга тўғри келади. Бунинг учун аввал биринчи, кейин иккинчи ва хоказо охирги маълумотларни ўқиш керак .

Файллар билан ишлаганда кўрсаткичнинг жорий ҳолатини билиш жуда муҳим ҳисобланади. Чунки, ***Readln*** ва ***read*** буйруқлари кўрсаткич кўрсатиб турган маълумотдан бошлаб бажарилади, яъни ўқилади. (Бошланғич синфлардаги хатчўпни кўз олдингизга келтиринг.)

Файл ўқиш учун очилган вақтда кўрсаткич-курсор ўқиш учун ундаги маълумотларнинг биринчисини кўрсатиб туради. Бу маълумот ўқилгандан сўнг кўрсаткич кейинги маълумотга ўтади. Унинг жорий холатини текшириб туриш учун Турбопаскалда мантиқий ***eof(y)*** (***end of file*** – файлнинг охири) ва ***eoln(y) (end of line*** – сатрнинг охири) функциялари киритилган. ***Eof(y)*** функцияси агар кўрсаткич файлдаги бирор маълумотни кўрсатиб турган бўлса "FALSE" (ёлғон), ўқиладиган

маълумотлар қолмаган, яъни файлдаги ҳамма маълумотлар ўқиб бўлинган бўлса "TRUE" (рост) қийматини олади.

Худди шунингдек, ***eoln(y)*** функциясн сатрдаги маълумотларнинг ҳаммаси ўқилган бўлса "TRUE", акс ҳолда "FALSE" қийматини олади.

Фараз қилайлик, ***y*** - файлида қуйидаги маълумотлар сақланаётган бўлсин:

3 2 5 4 2 3 5 4 3 4 5 4

Файл ўқиш учун очилган бўлса, дастлаб кўрсаткич ҳолати

3 2 5 4 2 3 5 4 3 4 5 4

кўринишида бўлади. Демак, EOF(y) нинг қиймати FALSE бўлади. Учта маълумот ўқилгандан сўнг кўрсаткич тўртинчи маълумотни кўрсатади:

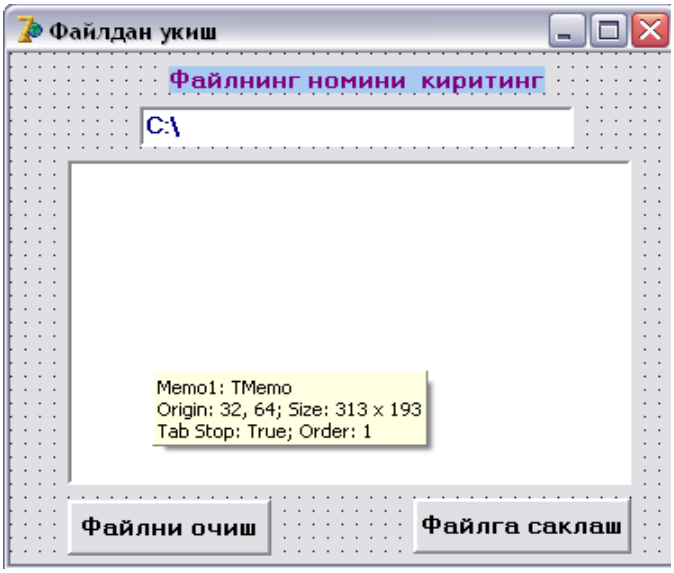
3 2 5 4 2 3 5 4 3 4 5 4

Бу ҳолда ҳам EOF(y) функциясининг қиймати - FALSE. Ҳамма маълумотлар ўқиб бўлингандан сўнг, кўрсаткич файлнинг охирига ўтади

3 2 5 4 2 3 5 4 3 4 5 4

ва eof(y) функцияси TRUE қийматини қабул қилади.

8.5-листинда фойдаланувчи кўрсатган номдаги файлдан сатрларни ўқийдиган дастур матни келтирилган. У ўқиган сатрларини Memo майдонига чиқаради. Бу дастурнинг диалог ойнаси 8.6 –расмда келтирилган.



8.6-расм. Файлдан ўқиш дастурининг диалог ойнаси

8.5-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 ButtonlClick(Sender: TObject);
private { Private declarations }
```

```
public  { Public declarations }
end;
var
Form1: TForml;
implementation
{$R *.dfm}
// Очиш тугмасини чертиш
procedure TForm1.Button1Click(Sender: TObject);
var
f: TextFile;// файл
fName : String[80];// файлнинг номи
buf: String[80];//файлдан ўқиш учун буфер
begin
fName := Edit1.Text;
AssignFile(f, fName);
{$I-}
Reset(f);//ўқиш учун очиш
{$I+}
if IOResult <> 0 then begin
MessageDlg(fName + 'файлига нотўғри мурожаат қилинди ',
mtError,[mbOk],0); exit; end;
// файлдан ўқиш
while not EOF(f) do begin
readln(f, buf);//файлдан навбатдаги сатрни ўқиш
Memo1.Lines.Add(buf);//бу сатрни Memo1 майдонига қўшиш
end;
CloseFile(f);//файлни ёпиш
end;
//Сақлаш тугмасини чертиш  — файлга ёзиб ўйиш
procedure TForml.Button2Click(Sender: TObject);
var  f: TextFile;//файл
fName: String[80];//файлнинг номи
i: integer ;
begin
fName := Edit1.Text; AssignFile(f, fName);
Rewrite(f);//файлни қайта ёзиш учун очиш
// файлга ёзиш
for I := 0 to Memo1.Lines.Count do
//сатр номерлари  0 дан бошланади
writeln(f, Memo1.Lines[i]);
CloseFile(f);//файлни ёпиш
MessageDlg('файлга ёзилди', mtInformation, [mbOk],0);
end;
end.
```

Дастурни ишга тушириш

Файлни қайта ишлашни ташкил қилиш учун while буйруғидан фойдаланилди. У ҳар гал (шу жумладан, биринчи мартада ҳам), навбатдаги маълумотни ўқишдан аввал, EOF функциясининг қийматини текширади. Бу цикл файлда маълумотлар тўла ўқилгунча бажарилаверади.

"Сақлаш" тугмаси ва унга мос процедура Memo майдонига киритилган сатрларни файлда сақлаб қўйиш учун мўлжалланган, яъни ўта содда матн муҳаррири ишини ифодалайди.

Файлдан ўқилган навбатдаси сатрни Memo майдонига ***Add*** методини қўллаган ҳолда қўшиб қўйилади.

Одатда файллар билан ишлаган вақтда, иҳтиёрий масала ечиш жараёни камида икки босқичдан иборат бўлади: а) файлни яратиш; б) файлдан фойдаланиш. Агар талаб қилинган файл илгари

яратилган бўлса, тўғридан тўғри иккинчи босқичга ўгиб, файлдаги маълумотлар доирасида масала шартида қўйилган саволларга жавоб қидириш мумкин. Акс ҳолда аввал биринчи босқични албатта бажариш лозим.

Масала. 1 дан 1000 гача бўлган бутун сонлар ичидаги туб сонларни F-файлига ёзинг.

Ечиш ғояси: Маълумки, агар 1 дан катта бўлган бутун сон фақат 1 га ва ўзига қолдиқсиз бўлинса, бу сонни туб сон деб аталади. Иҳтиёрий N-бутун соннинг 1 дан фарқли бўлувчилари $[2, \sqrt{N}]$ оралиқда бўлади. Қўйилган саволга жавоб бериш учун 1 дан бошлаб 1000 гача бўлган ҳамма сонларни ана шу оралиқдаги сонларга бирма-бир бўлиб чиқилади. Агар сон хеч бўлмаганда шу сонлардан биттасига бўлинса, демак бу сон туб эмас. Уни файлга ёзилмайди ва навбатда турган соннинг бўлувчиларини қидиришга ўтилади. Агар бўлувчилари бўлмаса, бу сон файлга ёзилади ва навбатдаги соннинг туб ёки туб эмаслигини текширишга ўтилади. 2 дан бошқа барча туб сонлар тоқ сон бўлгани учун, ҳисоблаш ишларини тезлаштириш мақсадида 2 ни тўғридан-тўғри файлга ёзилади. Сўнра туб сонларни фақат тоқ сонлар ичидан қидирилади.

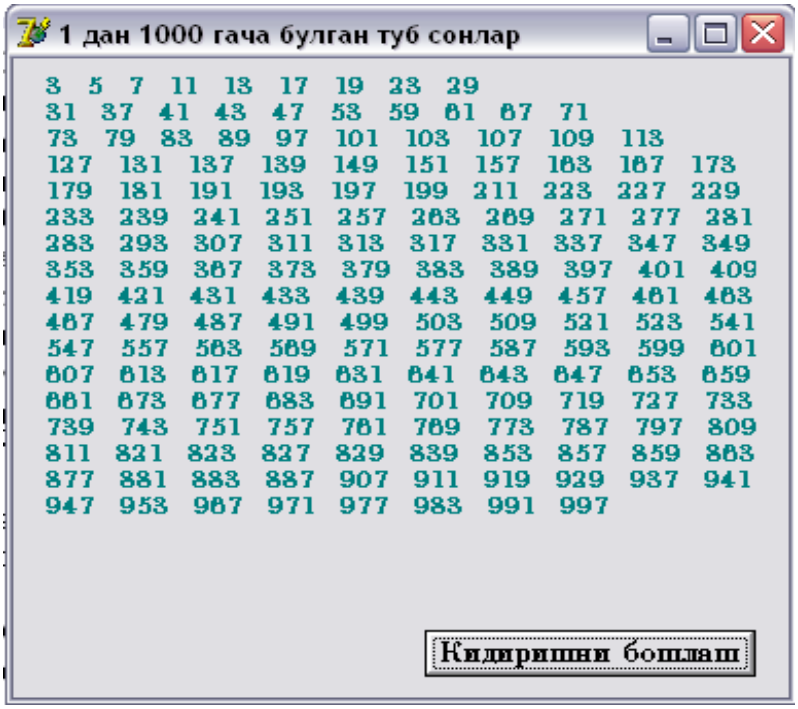
8.6-листинг. Файлга маълумотларни ёзиш

```
procedure TForm1.Button1Click(Sender: TObject);
var f: file of integer ;
    x, n, k,m : integer ;  y : string[3]; s:string;
begin
        assignfile(f,'C:/Tub.son');
        rewrite (f) ;//файл ёзиш учун очилди
     n := 2;  write (f, n) ;
     n := 1 ;   m := 1;
        while n<= 1000 do begin
       n := n+2;  y := 'ha' ;
           k := 2;//          N сонининг туб ёки туб эмаслиги текширилмоқда
           while (k<=sqrt(n))and(y='ha') do//текширилмоқда
           begin
                  if n mod k = 0 then y := 'yuq' ;
           k := k + 1;
       end;
           if y = 'ha' then begin
           write(f,n);       // туб сон файлга ёзилди
           //файлнинг ҳар бир сатрига 10 тадан туб сон ёзиш
        m := m + 1;
       //10 та туб сон ёзилган бўлса, янги сатрга ўтиш
        if m = 10 then s :=s + inttostr(n) +#13
          else s := s + inttostr(n) + '  ' ;
           if m = 11 then m := 1;
        end;
end;
```

Дастурни ишга тушириш

Юқоридаги дастурни ЭҲМ ёрдамида бажарсак, С диск да TUB.CON файли ҳосил қилинади ва унга 1 дан 1000 гача бўлган сонлар ичидаги туб сонлар ёзилади. Топилган бу сонлар диалог ойнасининг label1 ойнасига ҳам чиқарилади. Дастурнинг диалог ойнасида олинган натижалар 8.7-расмда келтирилган. Tub.son файлига ҳам айнан шу маълумотлар ёзиб қўйилди. Энди бу файлдаги маълумотлардан фойдаланиб, кўплаб масалаларни хал килиш мумкин. Келтирилаётган масалалар ана шулар жумласидандир:

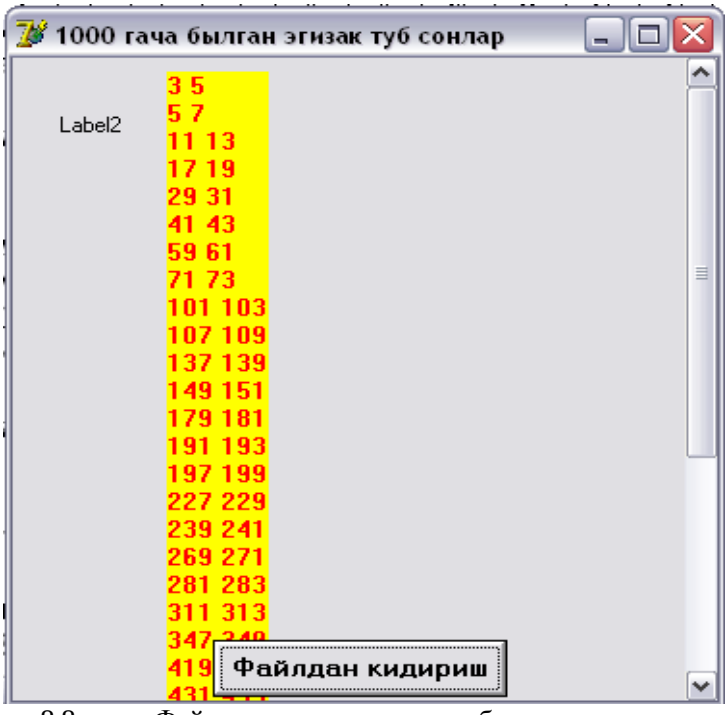
1. 1 дан 1000 гача бўлган туб сонлар файли TUB.CON даги сонларнинг урта арифметигини топинг.
2. 1 дан 1000 гача бўлган туб сонлар файли TUB.CON даги "эгизак" туб сонларни аниқланг.
3. 1 дан 1000 тача бўлган туб сонлар файли TUB.CON даги энг катта соннн топинг. Ва ҳоказо.



8.7-расм. 1000 гач бўлган туб сонлар дастурининг диалог ойнаси

Юқоридаги иккинчи масалани ечиш учун қуйидагича фикр юритиш тавсия қилинади :

Ечиш ғояси: "Эгизак" туб сон деб орасидаги фарқи 2 га тенг бўлган туб сонларга айтилади. Бу масалада туб сонларни қайтадан топилмайди, балки Tub.son файлидаги маълумотлар учун ечилади. Шунинг учун файлдаги кетма-кет келган икки туб соннинг айирмаси олинади. Агар айирма 2 га тенг бўлмаса, навбатдаги туб сонга ўтилади ва ҳоказо. Акс ҳолда, орасидаги фарқ 2 га тенг бўлган туб сонлар экранга чиқарилади ва кейинги сонга ўтилади. Аниқлик учун биринчи сон A, иккинчи сон эса B билан белгиланган бўлсин. Файлдаги маълумотлар сони аввалдан маълум бўлмагани учун *eof()* функциясидан фойдаланилади.



8.8-расм. Файлдан олинган эгизак туб сонлар жадвали

8.7-листинг. Файлдаги маълумотларни ўқиш

```
procedure TForm1.Button1Click(Sender: TObject);
    var f : file of integer ;  x , a, b : integer ;  s :string;
begin
    assignfile(f, 'c:\tub.son');
```

```
    reset(f) ;   //     файл ўқиш учун очилди
  S :='';      //натижаларни чиқариш учун
    read(f,a) ;        //  Биринчи сон ўқилди
    while not (eof(f)) do begin
    read (f,b);        //иккинчи сон ўқилди.
    if b-a=2 then s :=s + inttostr(a)+' ' + inttostr(b) + #13;;
   //"эгизак" туб сонлар чиқарилди ва
    a := b ;           //навбатдагим сонга ўтилди
    end ; label1.caption :=s;
end;
```

Дастурни ишга тушириш

Бу дастурнинг иш фаолиятини 8.8-расмдан кўриш мумкин.

менюга

9-боб. ЯНГИ ТИПЛАР БИЛАН ИШЛАШ

Биз хозиргача фақат стандарт, яъни integer, real, char, string ва boolean типдаги маълумотлар билан ишладик. Delphi тили дастурчиларга ностандарт типларни аниқлаш, эълон қилиш ва фойдаланишга имкон яратади.

Дастурчи аниқлайдиган янги типлар юқорида санаб ўтилган стандарт типлар ёки дастурчи олдинроқ аниқлаган типлар устига қурилади. Дастурчилар қуйидаги типдаги янги типларни эълон қилишлари ва фойдаланишлари мумкин:

- Элементлари саналадиган типлар;
- Элементлари чегараланган типлар;
- Аралаш типли маълумотлар (ёзувлар).

9.1. Элементлари саналадиган типлар

Кўпинча маълум бир оиладаги қийматларни (ҳафтанинг кунлари-душанба, сешанба, чоршанба, пайшанба, жума, шанба, якшанба ёки йилнинг фасллари-қиш, баҳор, ёз, куз каби) қабул қиладиган катталиклар билан ишлашга тўғри келади. Зарур бўлса, янги тип орқали бу қийматларни аниқлаш ва фойдаланиш мумкин. Бунда шу типдаги ўзгарувчилар қабул қила оладиган барча қийматларни бирма-бир санаб кўрсатилади. Бу иш умумий кўринишда

type *тип номи =(элементлар рўйхати) ;*

тарзида ташкил қилинади. Масалан:

```
type TKUN=dushanba, seshanba, chorshanba,
          payshanba, juma, shanba, yakshanba) ;

   TCol = (Red,Yellow,Green);
var X: KUN ; Y:TCol;
```

Энди X-ўзгарувчи фақат TKUN типидаги маълумотларни, Y ўзгарувчи эса Red, Yellow, Green қийматларидан бирини қабул қилади ҳалос. Ўзгарувчиларнинг қийматларини юқоридаги каби олдиндан аниқлаб қўйиш, дастурларнинг бажарилиши давомида янглишиб бошқача қиймат бериб қўйишнинг олдини олади.

Эслатма: Delphi тилида қабул қилинган келишувга кўра янги аниқланаётган типларнинг номлари T (Type – тип сўзидан олинган) ҳарфидан бошланиши шарт.

Элементлари санаб кўрсатиладиган тип эълон қилиниши билан бирга, бу элементларнинг бир-бирига нисбатан муносабати ҳам белгиланади. Рўйхатда кўрсатилган элемент ўзига нисбатан ўнг томонда жойлашган элементга қараганда кичикроқ ва аксинча. Биринчи бўлиб кўрсатилган элемент шу типдаги маълумотларнинг ичида энг кичиги, рўйхатнинг охиридаги берилган элемент эса энг катта ҳисобланади. Масалан, Tkun типидаги маълумотлар учун қуйидаги муносабат ўринли:

Dushanba < seshanba < chorshanba < ..., < yakshanba) ;

Тузилиш ҳусусиятига кўра элементлари саналадиган типдаги ўзгарувчилардан бошқарувчи буйруқларда ҳам фойдаланиш мумкин. Масалан:

```
if (Day >= Dushanba) OR (Day <= Juma) then
begin
{ амаллар кетма-кетлиги }
end;
```

Кўриниб турибдики, дастурчи аниқлаган типдаги маълумотлардан фойдаланиб ёзилган дастурлар кўргазмалироқ ва енгил ўқилади. Бу эса ҳатоликларни юзага келиш эҳтимоллигини камайтиради.

Компиляция вақтида Delphi санаб ўтиладиган типдаги қиймат оладиган ўзгарувчи ва қиймати унга бериладиган ифоданинг типи орасидаги мосликни текширилади. Агар ўртада мослик бўлмаса, экранга вужудга келган ҳатолик ҳақида ахборот берилади. Масалан,

```
Type  TCol =Red, Green, Blue ;
Var X : TCol;
begin
x :=1;
if x = 6  then begin
{ буйруқлар кетма-кетлиги }
end;
```

дастур парчасида x := 1 буйруғи нотўғри ёзилган. Бу ердаги ҳатолик ҳақида

incompatible types: 'Tcol' and 'Integer'

тарзидаги ахборот экранга узатилади. Чунки, ***x*** – ўзгарувчи дастурчи аниқлаган Tcol типига мансуб, унга эса бутун типдаги сон қиймат қилиб берилмоқда. Шунинг учун ***If*** шартида кўрсатилган буйруқ ҳам нотўғри.

Айтиш мумкинки, элементлари санаб ўтиладиган типларни эълон қилиш номланган константаларни эълон қилишнинг қисқартирилган вариантидир. Масалан, юқоридаги Tcol типининг эълони қуйидаги эълон билан тенг кучли:

Const Red = 0; Green = 1; Blue =2;

Эслатма: Элементлари саналадиган типдаги ўзгарувчиларнинг қийматларини дастурдаги буйруқлар натижасида кўрсатилган қийматлардан бошқа қиймат олмаслигини компилятор назорат қилмайди.

менюга

9.2. Элементлари чегараланган тип

Элементлари чегараланган типлар фақат элементлари тартибланган базавий типларнинг маълум бир қисмини чегаралар ёрдамида ажратиб олиш орқали ҳосил қилинади ва умумий кўринишда қуйидагича ёзилади:

type тип номи b_ch..o_ch ;

Бу ерда ***b_ch***-бошланғич чегара, ***o_ch***-охирги чегара. Бу типга мансуб бўлган ўзгарувчилар фақат ***b_ch*** - дан ***o_ch*** – гача бўлган оралиқдаги қийматларни қабул қила олади, бошқа ҳеч қандай қийматларни қабул қилмайди. Базавий тип сифатида одатда ***Integer*** типидан фойдаланилади.

Тип аниқлангандан сўнг, шу типдаги қийматларни қабул қиладиган ўзгарувчилар бошқа (стандарт типдаги ўзгарувчилар каби эълон қилинади. Шундан кейингина бу ўзгарувчилардан фойдаланиш мумкин. Масалан: баҳоларнинг 1 дан 5 гача бўлишини билган ҳолда

type TBaho=1..5 ;

янги типни ҳосил қилинади. Энди *B* типидаги қийматларни қабул қиладиган ўзгарувчиларни эълон қилиш мумкин :

var baxo : TBaho ;

Демак, дастурда қатнашадиган ҳамма *baxo* ўзгарувчилари фақат TBaho типидаги, яъни 1 дан 5 гача бўлган қийматларни қабул қилиши мумкин.

Элементлари чегараланган типларни эълон қилишда номланган константалардан ҳам фойдаланиш мумкин. Қуйидаги мисолда элементлари чегараланган TIndex типининг эълонида номланган константа – ***x***

қатнашмоқда:

Const x = 100;

Type Tindex = l .. X;

Элементлари саналадиган типлардан массивларни эълон қилишда фойдаланиш қулай. Масалан,

Type TIndex = 1 .. 100;

Var tab1 : array[TIndex] **of** integer; i:TIndex;

Бутун сонли типдан ташқари, зарур бўлса, элементлари саналадиган типлардан, умуман олганда элементлари тартибланган ихтиёрий типлардан (масалан, ***real*** типи тартибланмаган ҳисобланади) фойдаланиш ҳам мумкин. қуйидаги дастур парчасига эътибор беринг:

Type TMonth = (Jan, Feb, Mar, Apr, May, Jun,
Jul, Aug, Sep, Oct, Nov, Dec);

TSammer = Jun.. Aug;

Эслатма: Элементлари чегараланган типдаги ўзгарувчиларнинг қийматларини дастурдаги буйруқлар натижасида кўрсатилган диапазондан четга чиқмаслигини компилятор назорат қилмайди. Бу типларни эълон қилиш дастурнинг кўргазмалилигини ошириш учун керак бўлади ҳалос.

менюга

9.3. Аралаш типлар ёки ёзувлар

Мактаб ўқувчиси табелини кўз олдингизга келтиринг. Унда вилоят, туман, мактаб номери, синфи, ўқувчининг фамилияси, исми, отасининг исми, ўқув йили ҳамда ўқувчининг турли фанлар бўйича чораклардаги ва йиллик баҳолари сақланади. Демак, битта синфда 25 та ўқувчи бўлса, улар ҳақидаги ҳамма маълумотларни сақлаш учун 25 дона табел зарур бўлади. Энди ҳар бир табелдаги барча маълумотларни битта сатрга ёзилган ҳолатини кўз олдингизга келтиринг. Демак, 25 та сатрдаги маълумотлар жамғармаси ҳосил бўлади. Бу жамғармадан маълумотларни олиш ёки ёзиш ишларини осонлаштириш мақсадида мазмун жиҳатидан бир ҳил бўлган маълумотларни алоҳида устунларга ёзилади. Масалан: вилоятлар битта устунга, ўқувчиларнинг фамилиялари битта, исмлари бошқа устунга каби ёзилади.

Ҳосил бўлган устунларни майдон деб аталади. Ҳар бир сатрни эса ёзув дейилади. Демак, ёзувлар майдонлар тўпламидан иборат. Ҳар бир майдон мазмун жиҳатидан қандайдир бир ҳил типдаги маълумотларни ўз ичига олади.

Кўриниб турибдики, бир ёзувни битта ўзгарувчи деб қарасак, у ўзгарувчи турли типдаги қийматларни қабул қилади. Бир томондан, ёзувларни битта структура деб қаралса, иккинчи томондан бир нечта элементлардан иборат бўлган маълумот деб қараш мумкин. Бундай маълумотларнинг яна бир ҳусусияти шундаки, унинг ҳар бир элементи ҳар ҳил типларга мансуб бўлиши мумкин. Шунинг учун бу ўзгарувчиларни аралаш типли ўзгарувчилар деб ҳам юритилади. Бундай маълумотлар Delphi да ифодалаш учун ёзув (***Record***) деб аталадиган тузилмадан фойдаланилади.

Шундай қилиб, ёзув ёки аралаш тип деганда майдон деб аталадиган алоҳида номланган компоненталарнинг мураккаб тузилмасидан иборат бўлган маълумотларни тушунилади.

Ёзувлар умумий кўринишда

type *ёзув номи*=***record*** *1-майдон:1-тип;... ; n-майдон:n-тип* ***end;***

тарзида аниқланади. Масалан:

type ***T****tab*=*record vil, tum:string[20]; mak:integer;*
sinf:l..12; fam, ism:string[20]; uquv:integer end;

Энди шу типдаги маълумотларни қабул қиладиган ўзгарувчиларни эълон қилиш мумкин. Бу эълон одатдаги каби амалга оширилиши мумкин. Масалан:

var *uquvchi: Ttab ;*

Юқоридаги эълон ёрдамида ***uquvchi*** номли ўзгарувчининг ***vil, tum*** майдонлари узунлиги 20 тагача бўлган матнли, ***mak*** майдони бутун сонли, ***sinf*** майдони 1 дан 12 гача бўлган сонли, ***fam*** ва ***ism*** майдонлари узунлиги 20 гача бўлган матнли, ***uquv*** майдони эса бутун сонли маълумотларни сақлаш учун мўлжалланганлиги ҳақида ЭҲМ га ахборот берилмоқда. Чунки бу ўзгарувчи ***tab*** типидаги қийматларни қабул қилади. Шу ўзгарувчининг бирор майдонига мурожаат қилиш учун аввал ўзгарувчининг номи, "." белгиси ва майдон номи

uquvchi.vil, uquvchi.sinf, uquvchi.fam

тарзида кўрсатилади. Бу ерда *uquvchi* ўзгарувчисининг *vil, sinf, fam* майдонларига мурожаат қилинмоқда.

Uquvchi. vil:='Namangan'; uquvchi.sinf:=11;
uquvchi.fam:='Aliyev';

Кўриниб турибдики, ёзувнинг ҳамма ўзгарувчи ва майдон номлариини кўрсатишни ташкил қилиш анча мураккаб иш. Бу ишни фойдаланувчи учун қулайроқ ҳолга келтириш учун ***with*** хизматчи сўзидан фойдаланиш мумкин. Унинг умумий кўриниши қуйидагича:

with ёзув номи do begin ... end;

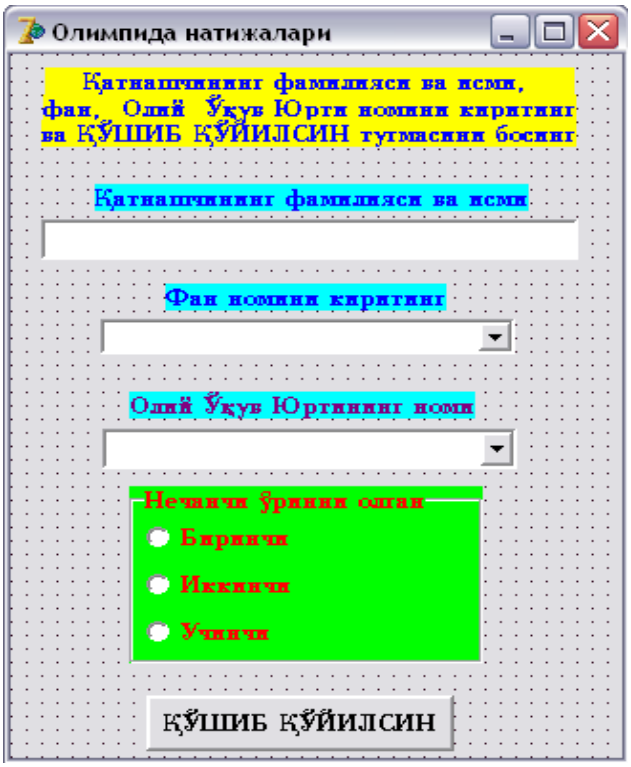
Бу имкониятдан фақат битта ёзувнинг бир нечта майдонлари учун фойдаланиш мумкин. Масалан:

with uquvchi do begin vil:='Namangan' ;
sinf:=ll ; fam:='Otaxanov'; end ;

Ёзувларни файлга киритиш ва файлдан ёзувларни киритиш. Барча маълумотлар каби ёзувларни ҳам файлларда сақлаш мумкин. Дастур ёзув-ўзгарувчининг қийматини файлга киритиш ёки файлдан олиш учун дастлаб компоненталари ёзув типида бўлган файлни эълон қилиш лозим. Масалан,

Type TTalaba = **record** fam, ism: **string**[20]; gurux : st**ring**[10]; **end;**
var f: **file of** TTalaba;

буйруқлари компоненталари TTalaba типида бўлган ***f*** файлини эълон қилади.

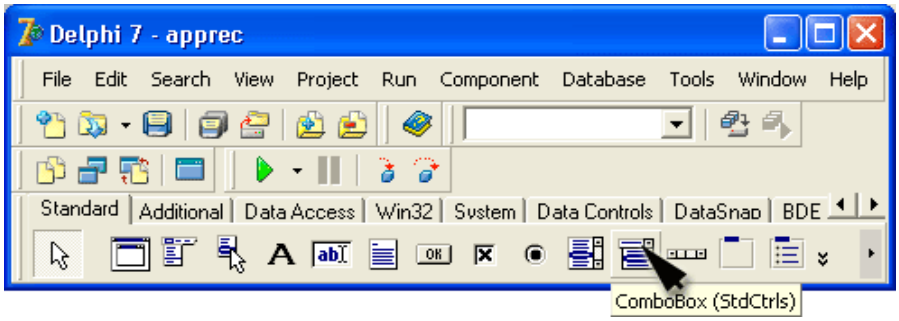


9.1-расм. Файлга ёзувларни қўшиш дастурининг диалог ойнаси

Ёзувли файллар билан ишлаш оддий файллар билан ишлаш жараёни билан бир ҳил. Дастлаб файлли ўзгарувчи эълон қилинади, сўнгра AssignFile буйруғи ёрдамида бу ўзгарувчини аниқ бир файл билан боғланади. Шундан кейин бу файлни очиш (ўқиш ёки ёзиш мақсадида) керак. Энди бу файлдаги маълумотларни ўқиш ёки бу файлга маълумотларни ёзиш мумкин бўлади.

Файлга ёзувларни ёзиш. Фойдаланувчи киритган фанлар бўйича олимпиада натижалари ҳақидаги маълумотларни файлда сақлаш учун ёзиб қўядиган дастурни кўрайлик. Бу дастур содда маълумотлар базасини ҳосил қилади. Бошланғич маълумотлар диалог ойнасининг (9.1-расм) махсус майдонига киритилади. Сўнгра бу маълумот компоненталари TUrin бўлган файлда сақлаб қўйилади.

+атнашчининг фамилиясини киритиш учун Edit1 майдони ташкил қилинган. Фан тури ва Олий ўқув юртининг номини танлаш учун эса ComboBox (аралаш рўйхат) компонентасидан фойдаланилади. Бу компонентанинг нишони **Standard** (9.2-расм)



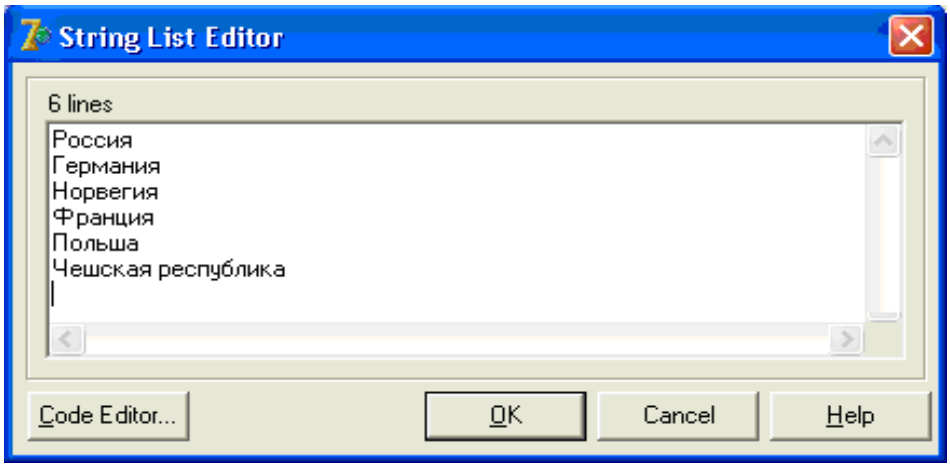
9.2-расм. ComboBox компонентасининг нишони

куроллар панелида жойлашган. ComboBox (аралаш рўйхат) компонентаси маълумотларни бевосита киритиш-таҳрирлаш майдонига киритиш ёки очиладиган рўйхат тугмаси чертилганда очиладиган рўйхатдан танлаш орқали киритиш имконини беради.

ComboBox компонентасининг айрим ҳусусиятлари **8.1-жадвал.**

Ҳусусияти	Мазмуни
Name	Компонентанинг номи. Компонентанинг ҳусусиятларига мурожаат қилиш
Text	Киритиш-таҳрирлаш майдонидаги матн
Items	Очиладиган рўйхат элементлари
DropDownCount	Очилган рўйхатда кўринадиган элементлар сони
Left	Компонентанинг чап чегарасидан форманинг чап чегарасигача бўлган масофа
Top	Компонентанинг юқори чегарасидан форманинг юқори чегарасигача бўлган масофа
Height	Компонентанинг баландлиги (киритиш-таҳрирлаш майдони)
Width	Компонентанинг кенглиги
Font	Рўйхат элементлари учун шрифт
ParentFont	Бош формадан шрифт ҳусусиятларини олиш

Очиладиган рўйхат тугмаси чертилганда экранга чиқариладиган рўйхат илова формасини яратиш жараёнида ҳам, дастурнинг иши жараёнида ҳам ҳосил қилиниши мумкин. Формани яратиш жараёнида рўйхат ҳосил қилиш учун **Object Inspector** ойнасида **Items** ҳусусиятидаги сатрлар рўйхатини таҳрирлаш муҳаррири (учта нуқтали тугма) чертилади. Муҳаррир ишга тушгандан кейин рўйхат элементларини киритиш мумкин. (9.3-расм). Дастурнинг тўла матни 9.1-листингда келтирилган.



9.3-расм. ComboBox2 компонентаси учун рўйхат киритиш

9.1-листинг. Файлга ёзувларни қўшиш.

```
unit fayl;
  interface
```

```
uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,   Dialogs, StdCtrls, ExtCtrls;
type
    TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Edit1: TEdit;
    ComboBox1: TComboBox;
    Label4: TLabel;
    Label5: TLabel;
    ComboBox2: TComboBox;
    Button1: TButton;
   Label6: TLabel;
   RadioGroup1: TRadioGroup;
   procedure Button1Click(Sender: TObject);
   procedure FormActivate(Sender: TObject);
   procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
type    //Ўринлар
TUrin = (Birinchi, Ikkinchi, Uchinchi);
//Ёзув эълон қилинмоқда
TMedal=record
fan: string[20];//фан номи
Vuz: string[40];//Олий Ўқув Юрти номи
fam_ism: string[30];//қатнашчи
urin: Turin;//ўрин
end;
var
  Form1: TForm1;
  f: file of TMedal;//ёзувлар файли
implementation
{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
var qatn: TMedal;
begin
with qatn do begin
Fan := ComboBox1.Text;
Vuz := ComboBox2.Text;
fam_ism := Edit1.Text;
case RadioGroup1.ItemIndex of
0: Urin := Birinchi;
1: Urin := Ikkinchi;
2: urin := Uchinchi;
end;
end;
write(f,qatn);//файлга ёзилмоқда
end;

//формани активлаштириш
procedure TForm1.FormActivate(Sender: TObject);
var
resp : word;//фойдаланувчининг жавоби
begin
AssignFile(f, 'C:/Urinlar.db');
```

```
{$I-}
Reset (f);//файлни очиш
Seek(f, FileSize(f));//кўрсаткични файлнинг охирига ўтказади
{$I+}
if IOResult = 0
then button1.enabled := TRUE
//Энди +ўшиб қўйилсин тугмаси ишлайди
else begin
resp := MessageDlg('Маълумотлар базаси топилмади.'
+ 'Янги МБ яратайми?', mtInformation,[mbYes,mbNo],0);
if resp = mrYes then begin {$I-}
rewrite(f);
{$I+}
if IOResult = 0
then button1.enabled := TRUE
else ShowMessage('МБ файлини яратишда ҳатолик бор!');
end;
end;
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
CloseFile(f);//файлни ёпиш
end;
end.
```

Дастурни ишга тушириш

Келтирилган ушбу дастурда TForm1.FormActivate процедураси файлни давомига ёзиш учун очади. Бу ерга қуйидаги ҳолатга эътибор беринг. Файлнинг охирига янги маълумотларни қўшиш учун AppendFile процедурасидан фойдаланиб бўлмайди, чунки, файлимиз матнли файл эмас. Шунинг учун файлни қайтадан ёзиш режимида, яъни Rewrite процедураси билан очамиз. Сўнгра Seek процедураси ёрдамида кўрсаткични файлнинг охирига ўрнатамиз. Seek процедурасининг параметри бўлиб Filesize функцияси хизмат қилади ва унинг қиймати файлнинг ҳажмига (байтларда) тенг.

TForm1.Button1Click процедураси ёзувларни тўғридан-тўғри файлга ёзади. Fan ва Vuz майдонлари аралаш типдаги рўйхатлар ҳусусиятидан, яъни Фан (comboBox1) ва Олий Ўқув Юрти (ComboBox2) майдонларидан тўлдирилади.

Fam_ism майдони қатнашчи (edit1 компонентасидан) киритиш-таҳрирлаш майдонидан, қатнашчи олган ўрин эса RadioGroupl компонентаси тугмаларидан олинади.

TForm1.FormClose процедураси файлни ёпади. Tmedal типи иккита процедуралар (TForm1.FormActivate ва TForm1.Button1Click) да фойдаланилгани учун, уни форма модули бўлимида эълон қилинган. f – файлли ўзгарувчи ҳам худди шу сабабли форманинг модул бўлимида эълон қилинган.

Дастурнинг келтирилган вариантида фанларнинг номи ҳамда Олий Ўқув Юртларининг рўйхати рўйхат сатрларининг муҳаррири ёрдамида аниқланиши кўзда тутилган. Шу билан бирга, дастурнинг иши жараёнида ҳам бу рўйхатни ҳосил қилиш мумкин. Бунинг учун Add методидан фойдаланилади. Уни дастурга (Tform1.FormActivate процедураси) матнига қуйидаги буйруқлар билан қўшиб қўйилади:

```
Form1.ComboBox1.Item.Add('Россия');
Form1.ComboBox1.Item.Add('Австрия');
Form1.ComboBox1.Item.Add('Германия');
Form1.ComboBox1.Item.Add('Франция');
```

Файлдан ёзувларни ўқиш. Аввалги пунктда яратилган файлдан ёзувларни ўқиш ва қайта ишлаш жаранини намойиш қилувчи дастурни кўрамиз. Унинг диалог ойнаси 9.4-расмда, матни эса 9.2-листингда келтирилган. Бу дастур **Хаммаси** ёки **Танлаш** тугмаларидан бирини танланишига қараб, файлдан ўқилган ёзувларни маълумотларни Memo1 компонентасининг ойнасига чиқаради. 9.2-далвалда дастур формадаги компоненталар ҳусусиятининг қийматлари келтирилган.

Memo1 компонентаси фақат маълумотларни кўриш учун мўлжалланганлиги сабабли, ReadOnly

(фақат ўқиш, кўриш) ҳусусиятига True қиймати берилган. Memo компонентасининг ScrollBars (силжитиш полосаси) ҳусусияти кўринадиган ҳолга келтирилган. Агар қийматьи олдиндан кўрсатилмаса, ScrollBars ҳусусиятига ssNone қиймати берилган, яъни силжитиш полосалари кўринмайди. Қаралаётган мисолда экранга вертикал силжитиш полосаси чиқарилади, шунинг учун ScrollBars ҳусусиятига ssVertical қиймати берилган.

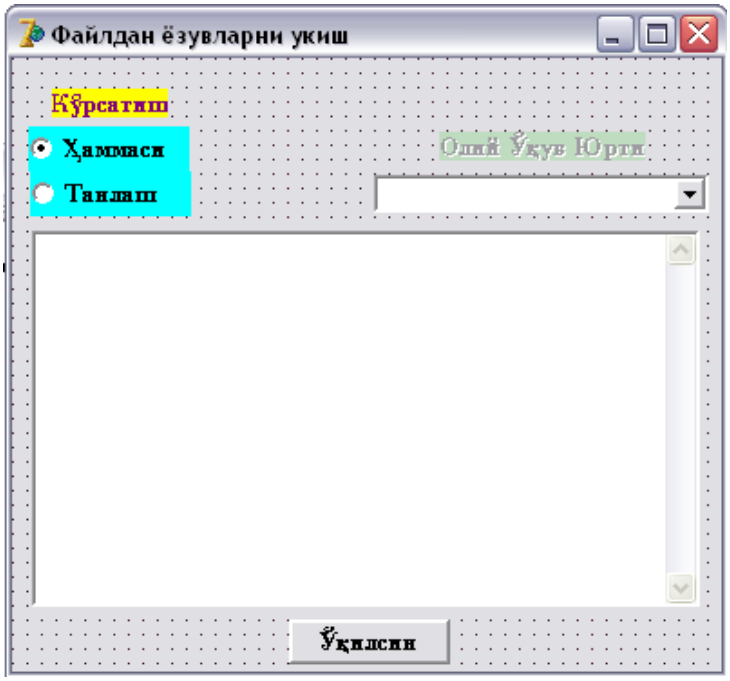
Компоненталар ҳусусиятларининг қийматлари. 9.2-жадвал

Ҳусусияти	қиймати
RadioButton1 . Checked	True
Label1 .Enabled	False
ComboBox1 . Enabled	False
Memo1 . Readonly	True
Memo1. ScrollBars	ssVertical

Олий Ўқув Юртлар рўйхатини чиқариш учун ComboBox1 компонентасидан фойдаланилади. У маълумотни киритиш эмас, балки фойдаланувчига таклиф қилинадиган рўйхатдан танлашга имкон беради. Бу рўйхатни формани яратиш жараёнида Items ҳусусиятига қийматлар бериш орқали ҳосил қилинади.

Дастур ишга тушганидан сўнг, Олий Ўқув Юртлари рўйхатини танлаш режимини ўчириш учун (бу ҳолда **Кўрсатиш** гуруҳининг **Хаммаси** тугмаси танланган деб қабул қилинади) йству ComboBox1 ва Label1 компоненталарининг Enabled ҳусусиятига формани яратиш жараёнида False қийматини бериб қўйилади.

Олий Ўқув Юртини танлаш (ComboBox1) режими дастурнинг ишлаши жараёнида **Танлаш** тугмаси чертилганда ишга тушади. RadioButton2 ўчиргичи учун Onclick ходисаларни қайта ишлаш процедураси ComboBox1 майдонига рухсат беради.



9.4-расм. Файлдан ёзувларни ўқиш дастурининг диалог ойнаси

9.2-листинг. Файлдан ёзувларни ўқиш

unit fayl_uqish;

```
interface
uses   Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,   Dialogs, StdCtrls, ExtCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    ComboBox1: TComboBox;
    Memo1: TMemo;
```

```
    Button1: TButton;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    Label2: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure RadioButton1Click(Sender: TObject);
    procedure RadioButton2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var   Form1: TForm1;
implementation
{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
type TUrin = (Birinchi, Ikkinchi, Uchinchi);
//ёзув
TMedal = record
Fan : string[20]; Vuz:string[20];
Fam_ism : string[30]; urin:TUrin;
end;
var
f: file of TMedal;//ёзувлар файли
rec: TMedal;//файлдан ўқилган ёзув
n: integer;//сўралган ёзувлар сони
st: string[80];
begin
AssignFile(f,'c://Urinlar.db');
{$I-}
Reset (f);//файлни ўқиш учун очиш
{$I-}
if IOResult <> 0 then begin
ShowMessage('МБ файлини очишдаги хатолик.');
Exit;
end;
//МБ ни қайта ишлаш
if RadioButton2.Checked then
Memo1.Lines.Add('*** ' + ComboBox1.Text + ' ***'); n := 0;
Memo1.Clear;//Memo майдонини тозалаш
while not EOF(f) do begin
read(f, rec);//ёзувни файлдан ўқилмоқда
if RadioButton1.Checked or
(rec.Vuz = ComboBox1.Text) then begin
n := n + 1;
st := rec.fam_ism +  ', ' + rec.fan;
if RadioButton1.Checked then
st := st + ', ' + rec.Vuz;
case rec.Urin of
Birinchi : st := st+ ', Олтин';
Ikkinchi : st := st+ ', Кумуш';
Uchinchi : st := st+ ', Бронза';
end;
Memo1.Lines.Add(st); end;
end;
CloseFile(f);
if n = 0 then
ShowMessage('Мб да сўралган маълумот топилмади.');
```

```
end;

procedure TForm1.RadioButton1Click(Sender: TObject);
begin
Label1.Enabled := False;
ComboBox1.Enabled := False;// Энди Vuz майдони ёпилди
end;

procedure TForm1.RadioButton2Click(Sender: TObject);
begin
Label1.Enabled := True;
ComboBox1.Enabled := True;// Vuz майдони очилди
ComboBox1.SetFocus;// курсор Vuz майдонига ўтди
end;
end.
```

Дастурни ишга тушириш

TForm1.Button1Click процедураси кўрсатилган файлни очади ва кетма-кет ундаги ёзувларни ўқийди. Бу маълумотлар Memo1 майдонига қўшилиб боради. Бу амал Memo1.Lines.Add(st) буйруғи билан бажарилади.

менюга

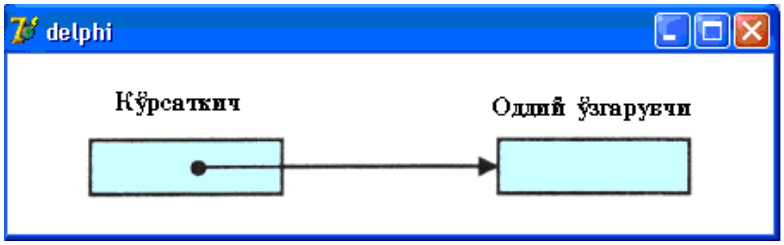
9.4. Динамик структурали маълумотлар

Биз ҳозиргача статик, яъни дастурнинг ишлаши жараёнида ўзгармайдиган структурали маълумотлар билан ишлаб келдик. Дастурнинг ишлаши жараёнида ўзгарувчиларнинг сони ўзгармаган ҳолда фақат ўзгарувчиларнинг қиймати ўзгариши мумкин эди. Шунинг учун ҳам бундай маълумотларни статик маълумотлар деб аталади. Айрим ҳолларда статик маълумотлар билан ишлаш ноқулайликлар келтириб чиқаради. Масалан, синф ўқувчилари ҳақидаги маълумотларни киритиш ва қайта ишлаш талаб қилинган дастурда маълумотларни сақлаш учун массивлардан фойдаланилади. Массивнинг ўлчамларини аниқлашда дастурчи синфдаги ўқувчиларнинг бўлиши мумкин бўлган энг катта ёки ўртача қийматини ҳисобга олиши керак эди. Агар ўқувчиларнинг сони ҳақиқатда бу миқдордан кам бўлса, ЭҲМ хотирасидан ноўрин фойдаланилган бўлади, кўп бўлган ҳолда эса бу дастурни қўллаб бўлмайди. (Дастур матнига ўзгартириш киритиб, компиляция қилишга тўғри келади.)

Бундай характерга эга бўлган масалалар табиатан ўзгарувчан бўлади ва бундай масалаларни динамик структуралар ёрдамида ечиш қулай ҳисобланади.

Кўрсаткичлар. Одатда дастурда қатнашаётган ҳар бир ўзгарувчи учун ЭҲМ хотирасидан биттадан ячейка ажратилади. Дастурчи ўзгарувчининг ЭҲМ хотирасидаги қайси ячейкага ёзилганлиги билан қизиқмаслиги мумкии. Дастурда ўзгарувчилардан қандайдир маълумотларни сақлаш учун фойдаланилган. Бу ўзгарувчиларга керак бўлган вақтда уларнинг номини кўрсатиш орқали мурожаат қилинади. Бу ўзгарувчиларнинг қийматлари сақланаётган ячейкаларнинг адресларини дастурчи билиши шарт эмас эди.

Оддий ўзгарувчилардан ташқари Delphi да бошқа ўзгарувчиларга мурожаат қиладиган ўзгарувчилар ҳам мавжуд. Бундай ўзгарувчиларни кўрсаткичлар деб аталади. Кўрсаткич - бу шундай ўзгарувчики, унинг қиймати бошқа ўзгарувчининг манзили ёки маълумотларнинг структурасига тенг бўлади. Уни график кўринишда қуйидагича тасвирлаш мумкин: (9.5-расм)



9.5-расм. Кўрсаткич-ўзгарувчи

Баъзи масалаларни ечиш жараёнида кўрсаткичлардан фойдаланишга тўғри келиб қолади. Мана

уларнинг айримлари:

- Дастур умумий хажми 64 килобайтдан катта бўлган маълумотлар билан ишлаши зарур бўлса. Дастурдаги маълумотлар тўлалигича ЭҲМ нинг маълумотлар сегментида сақланади. Унинг ҳажми эса 64 килобайт. (Ортиқча маълумотларни иима қилиш керак ?)
- Дастур умумий хажми олдиндан маълум бўлмаган маълумотлар билан ишлашига тўғри келиб қолса. Баъзи ҳолларда, масалан, сатрлар ёки массивларнинг барча элементларини сақлаш учун мумкин бўлган энг катта жойни банд қилиш лозим бўлади. Бу жойнинг ҳаммасига ҳам маълумот ёзилмаслиги мумкин. Демак, хотирадан ноўрин ва самарасиз фойдаланишга йўл қўйилади.
- Маълумотларни сақлаш учун хотира буферларидан фойдаланилса. Кўпинча маълумотларни, масалан, матнларни ЭҲМ хотирасига доимий сақлаш учун ёзишдан аввал, вақтинча буферларда сақлаб туришга тўғри келади. Бунинг учун маълум бир ҳажмдаги буферларни ажратиш лозим. Буфер ҳажми кичик бўлиб, матн катта бўлиши ёки катта ҳажмдаги буферда кичкина матн сақланиши мумкин. Бу ҳолда ҳам хотира нотўғри тақсимланмоқда.

Кўрсаткичлардан фойдаланиш хотирадан унумли фойдаланишга ёрдам беради. Кўрсаткич ЭҲМ хотирасида бор-йўғи 4 байт жойни банд қилган ҳолда, бир неча ўн килобайт жойни банд қилаётган маълумотларни кўрсатиши мумкин.

Кўрсаткич, дастурдаги бошқа ўзгарувчилар каби ўзгарувчиларни эълон қилиш бўлимида эълон қилинади. Умумий кўринишда кўрсаткичлар қуйидагича эълон қилинади:

Ном: ^ Тип;

Бу ерда Ном – кўрсаткичли ўзгарувчининг номи; Тип —кўрсаткичли ўзгарувчи кўрсатадиган маълумотнинг типи; ^ белгиси эълон қилинаётган ўзгарувчининг кўрсаткич эканлигини англатади. Кўрсаткичларни эълон қилишга мисоллар келтирамиз:

```
p1: ^integer;
p2: ^real;
```

Бу мисолларда p1 ўзгарувчи — бу integer типидаги ўзгарувчини кўрсатади, p2 —эса real типидаги ўзгарувининг кўрсаткичи.

Кўрсаткич кўрсатаётган ўзгарувчининг типи кўрсаткичнинг типи ҳисобланади. Масалан, дастурда p: ^integer кўрсаткичи эълон қилинган бўлса, ^p — бутун типдаги кўрсаткич ёки "p — бу бутун кўрсаткич" деб аташ қабул қилинган.

Дастур иш бошлаган вақтда кўрсаткич-ўзгарувчи "ҳеч нарсани кўрсатмайди". Бу ҳолда кўрсаткичнинг қиймати NIL га тенг дейилади. NIL хизматчи сўзи бирор маълумотни кўрсатмаётган кўрсаткичнинг қийматини билдиради. NIL идентификатордан қиймат бериш ва мантиқий ифодаларда фойдаланиш мумкин. Масалан, p1 ва p2 ўзгарувчилар кўрсаткич деб эълон қилинган бўлса,

```
p1 := NIL;
```

буйруғи p1 – ўзгарувчига қиймат беради,

```
if p2 = NIL then ShowMessage('p2 - кўрсаткич аниқланмаган!');
```

буйруғи эса p2- кўрсаткични аниқланганлигини текширади.

Кўрсаткичга ўзига мос типдаги ўзгарувчининг манзилини қиймат қилиб бериш мумкин. (Дастур матнида ўзгарувчининг манзили – бу олдида @ оператори турган ўзгарувчининг номидан иборат бўлади.) +уйидаги буйруқ натижасида p – ўзгарувчининг қиймати n - ўзгарувчининг манзилига тенг бўлади:

```
p := @n;
```

Ўзгарувчининг манзилидан ташқари, кўрсаткичга бошқа кўрсаткичнинг қийматини бериш мумкин. (Бу мулоҳаза ўзаро бир ҳил типли кўрсаткичлар учун ўринли.) Масалан, p1 ва p2 кўрсаткичлар integer типи да бўлса,

```
p2 := p1;
```

буйруғининг бажарилиши натижасида p1 ва p2 кўрсаткичлар битта ўзгарувчини кўрсатади.

Кўрсаткичлардан улар кўрсатиб турган ўзгарувчилар билан ишлаш имконига эга бўлиш мақсадида фойдаланиш мумкин. Масалан, агар ***p*** - кўрсаткич ***i*** – узгарувчини кўрсатиб турган бўлса,

```
p^ := 5;
```

буйруғининг натижасида ***i*** ўзгарувчининг қиймати бешга тенг бўлади. Келтирилган мисолда ^ нишони шуни англатадики, беш қиймати кўрсаткич-ўзгарувчи кўрсатиб турган ўзгарувчига берилади.

менюга

9.5. Динамик ўзгарувчилар

Динамик ўзгарувчилар деб дастурнинг ишлаши жараёнида хотирадан жой ажратиладиган ўзгарувчиларга айтилади.

Динамик ўзгарувчилар учун хотирадан жой ажратиш амали ***new*** процедураси ёрдамида бажарилади. New процедурасида битта параметр — хотира ажратилиши талаб қилинган ўзгарувчининг типга кўрсаткич мавжуд. Масалан, агар p – real типдаги кўрсаткич бўлса, у ҳолда

new(p);

процедурасининг бажарилиши натижасида real типидаги ўзгарувчи учун жой ажратилади (real типидаги ўзгарувчи яратилади), ҳамда ўзгарувчи-кўрсаткич p га шу ўзгарувчи учун ажратилган хотиранинг манзили қиймат қилиб берилади.

Динамик ўзгарувчининг номи йўқ, шунинг учун унга фақат кўрсаткич ёрдамида мурожаат қилиниши мумкин.

Динамик ўзгарувчидан фойдаланадиган процедура ўз ишини якунлашидан аввал шу ўзгарувчилар банд қилган хотира қисмини бўшатиши лозим. Буни дастурчилар тилида динамик ўзгарувчиларни "йўқ қилиш" ҳам деб аталади. Динамик ўзгарувчи банд қилган ҳотирани бўшатиш учун Dispose процерурасидан фойдаланилади. Бу процедуранинг параметри битта - динамик ўзгарувчига кўрсаткич. Масалан, агар p — new(p) процедураси билан хотирадан жой ажратилган динамик ўзгарувчига кўрсаткич бўлса,

dispose(p);

динамик ўзгарувчи банд қилган хотирани бўшатади.

Қуйидаги процедура (унинг матни 9.3-листингда келтирилган) динамик ўзгарувчиларни яратиш, фойдаланиш ва йўқ қилиш жараёнини намойиш қилади.

9.3-листинг. Динамик ўзгарувчиларни яратиш, фойдаланиш ва йўқ қилиш

```
procedure TForm1.Button1Click(Sender: TObject);
  var p1,p2,p3: ^Integer;//integer типидаги ўзгарувчига кўрсаткич
  begin
 //integer типидаги динамик ўзгарувчи учун хотирадан жой ажратамиз.
 //динамик ўзгарувчилар учун хотирадан жой ажратамиз.
  New(p1); New(p2); New(p3);
  p1^ := 5;  p2^ := 3;  p3^ := p1^ + p2^;
  ShowMessage('Берилган сонларнинг йиғиндиси  ' + IntToStr(p3^)
  + ' га тенг');
 //динамик ўзгарувчиларни йўқ қиламиз
 //Динамик ўзгарувчилар банд қилган хотира қисмини бўшатамиз.
  Dispose(p1); Dispose(p2); Dispose(p3);
  end;
```

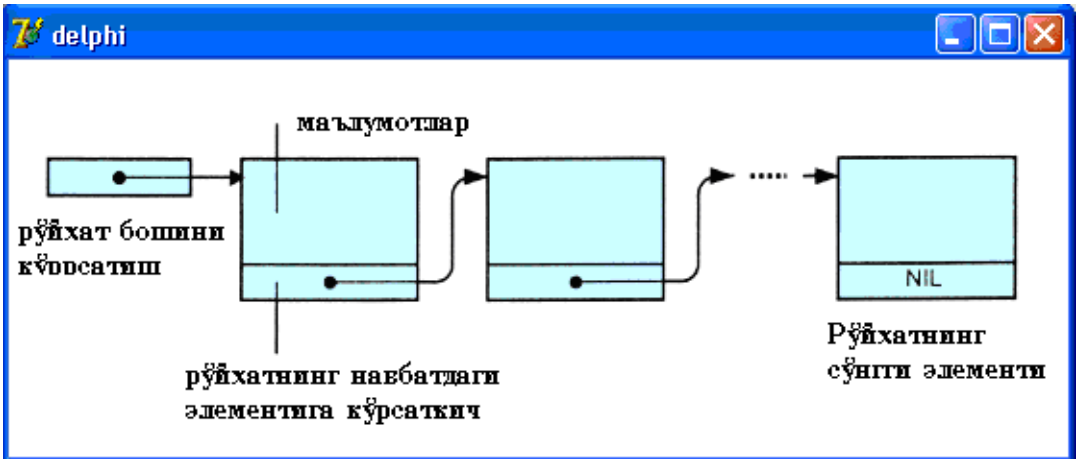
Процедура иш бошлаган вақтда учта динамик ўзгарувчи яратади. p1 ва p2 ўзгарувчилар қиймат бериш буйруғи ёрдамида қийматлар олади, учинчиси эса дастлабки иккитасининг йиғиндисига тенг.

менюга

9.6. Рўйхатлар

Кўрсаткич ва динамик ўзгарувчилар рўйхат, дарахт каби мураккаб динамик структурали маълумотларни яратишга имкон беради.

Рўйхатни қуйидагича тасвирлаш мумкин: (9.6-расм).

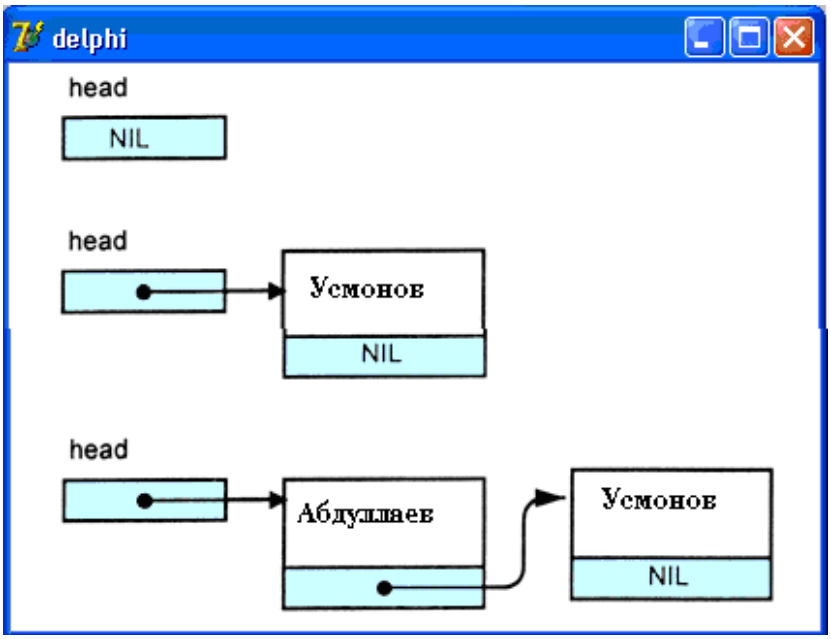


9.6-Расм. Рўйхатнинг тасвирий ифодаси

Рўйхатнинг ҳар бир элементи (тугуни) икки қисмдан иборат бўлган ёзувдан иборат. Ёзувнинг биринчи қисми маълумот характерига эга. Иккинчиси эса рўйхатнинг навбатдаги (олдинги) элементи билан боғланишга жавоб беради. Фақат навбатдаги элемент билан боғланиш усулини таъминлайдиган рўйхат бир боғламли деб аталади.

Дастур рўйхатдан фойдалана олиши учун рўйхатнинг компоненталари типини аниқлаш ва кўрсаткич ўзгарувчи-кўрсаткичга рўйхатнинг биринчи элементини кўрсатиб қўйилади. Қуйида талабалар рўйхатининг компоненталарини эълон қилишга намуна келтирилган

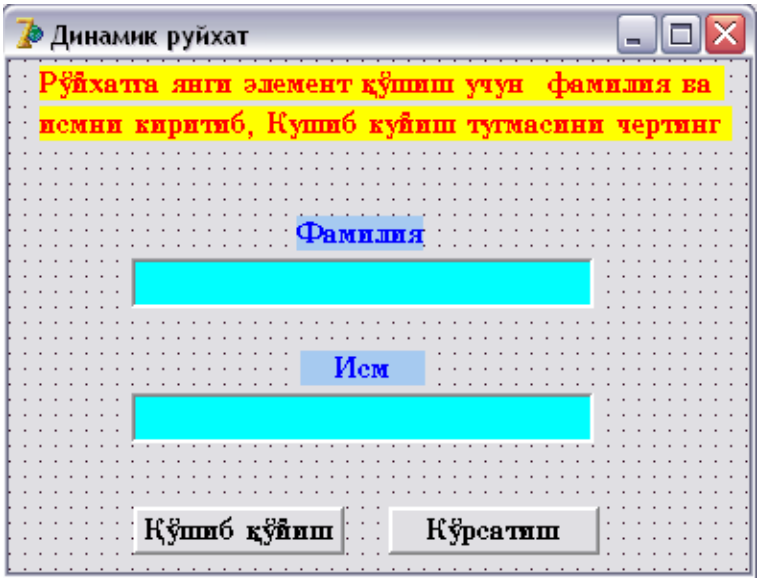
```
type
  TPStudent = ^TStudent;//Tstudent типидаги ўзгарувчига  кўрсаткич
 //рўйхат элементларининг типини ифодалаш
  TStudent = record
  surname, name: string[20];'//фамилияси ва исми
  group: integer;//гуруҳ номери
  address: string[60];//Манзили
  next: TPStudent;//Рўйхатнинг кейинги элементига кўрсаткич
  end;
  var
  head: TPStudent;//Рўйхатнинг биринчи элементига кўрсаткич
```



9.7-Расм. Рўйхатга янги элемент қўшиш

Маълумотларни рўйхатнинг бошига, охирига ёки керакли ерига қўйиш мумкин. Бу ҳолларнинг барчасида кўрсаткич ишини тузатиб борилади. 9.7-расмда рўйхат бошига янги элемент қўшиш кўрсатилган. Унга иккинчи элемент қўшилганидан сўнг head шу элементни кўрсатади.

Қуйидаги дастур рўйхатнинг бошига янги фамилиялар киритиб, талабалар рўйхатини ҳосил қилади. Маълумотлар дастур диалог ойнасининг киритиш-таҳрирлаш майдонига киритилади. Бу маълумот **Қўшиб қўйиш** тугмаси чертилганда рўйхатга қўшилади.



9.8-расм. Динамик рўйхат дастурининг диалог ойнаси

9.4-листинг. Динамик рўйхатга янги элемент қўшиш

```
unit Din_ruy;
interface
uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,   Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Edit1: TEdit;
    Label4: TLabel;
    Edit2: TEdit;
    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}
type
TPStudent = ^TStudent;//TStudent типи даги кўрсаткич
TStudent = record
f_name:string[20];//фамилия
l_name: string[20];//исми
next: TPStudent;//рўйхатнинг кейинги элементи
end;
var
head: TPStudent;//Рўйхатнинг бошланиши

//Рўйхатнинг бошланишига янги элемент қўшиш
```

```
procedure TForm1.Button1Click(Sender: TObject);
var
curr: TPStudent;//рўйхатнинг янги элементи
begin
new(curr);//рўйхат элементи учун жой жратиш
curr^.f_name := Edit1.Text;
curr^.l_name := Edit2.Text;
//рўйхат бошига қўшилмоқда
curr^.next := head;
head := curr;
//Киритиш майдони тозаланмоқда
Edit1.text :=''; Edit2.text := '' ;
end;

//рўйхатни экранга чиқариш
procedure TForm1.Button2Click(Sender: TObject);
var
curr: TPStudent;//рўйхатнинг жорий элементи
n:integer;//рўйхат (элементларининг) сони
st:string;//Рўйхатнинг сатрли кўриниши
begin n := 0; st := '';
curr := head;//рўйхатнинг биринчи элементи
while curr <> NIL do begin
n := n + 1;
st := st + curr^.f_name + ' ' + curr^.l_name + #13;
curr := curr^.next;//рўйхатнинг кейинги элементини кўрсатиш
end;
if n <> 0
then ShowMessage('Рўйхат:' + #13 + st)
else ShowMessage('Рўйхатда элементлар йўқ.');
end;
end.
```

Дастурни ишга тушириш

Рўйхатга элемент қўшишни TForm1.Button1Click процедураси бажаради. У динамик ўзгарувчи-ёзувни яратади, унинг майдонларига диалог ойнасининг киритиш майдонларидаги қийматларни ҳамда кўрсаткичнинг head майдонига тузатмалар киритади.

Рўйхатни экранга TForm1.Button2Click процедураси чиқаради. Бу процедура **Кўрсатиш** тугмаси чертилганда ишга тушади. Рўйхат элементлари билан ишлаш учун curr кўрсаткичидан фойдаланади. Дастлаб у рўйхатнинг биринчи элементи адресига тенг. Рўйхатнинг биринчи элементи қайта ишланганидан сўнг, curr кўрсаткичига curr кўрсатаётган ёзувнинг next майдонининг қиймати берилади. Натижада curr ўзгарувчиси рўйхатнинг иккинчи элементи адресига тенг бўлади. Кўрсаткич шундай усул билан рўйхат элементлари бўйлаб сурилиб боради. Бу жараён жорий элемент next майдонининг қиймати (curr ўзгарувчиси кўрсатаётган элементнинг манзили) NIL бўлиб қолмагунча давом этади.

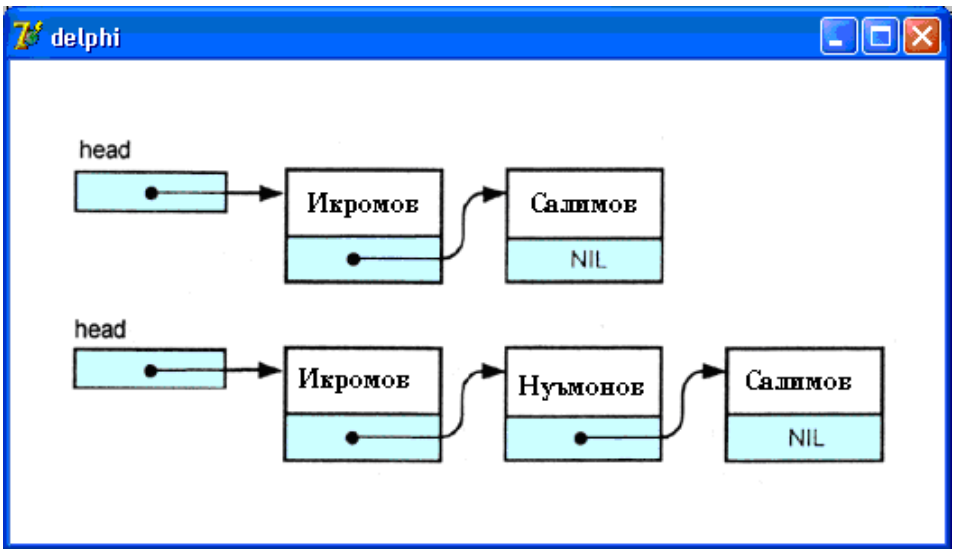
менюга

9.7. Тартибланган рўйхат

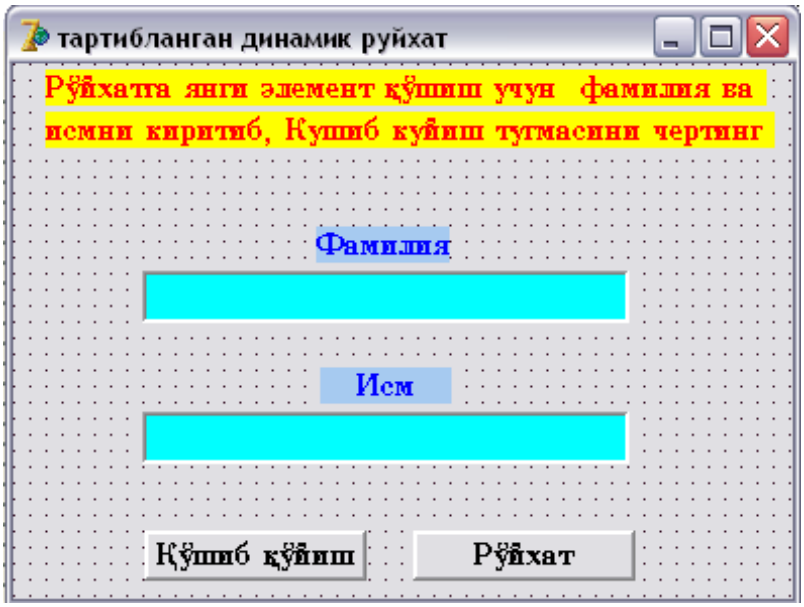
Одатда рўйхатнинг элементлари қандайдир усул билан тартибланган бўлади. Бу тартиблаш бирор майдоннинг элементлари бўйича амалга оширилади. Масалан, одамлар ҳақидаги маълумотлар фамилиялар киритилган майдон бўйича тартибланган бўлиши мумкин.

Элементни рўйхатга қўшиш. Элементларни рўйхатга қўшиш кўрсаткичларга тузатмалар киритиш орқали амалга оширилади. Элементни тартибланган рўйхатга қўшиш учун, тартибланган рўйхатдан янги элемент қайси элементдан кейин туриши кераклиги аниқланади. Шундан кейин кўрсаткичларга тузатмалар киритилади. Янги элементнинг кўрсаткичини тартибланган рўйхатда ундан аввол келадиган элементни

кўрсатаётган элементга қаратилади. Янги элементдан олдин туриши керак бўлган элементнинг кўрсаткичини янги элементга қаратилади. (9.9-расм.).



9.9-Расм. Тартибланган рўйхатга янги элемент қўшиш



9.10-расм. Тартибланган динамик рўйхат2 дастурининг диалог ойнаси

Қуйидаги дастур (унинг матни 9.5-листингда келтирилган, диалог ойнаси эса 9.1- расм) **Фамилия** майдони бўйича тартибланган рўйхат ҳосил қилади. Рўйхатга элементлар Edit1 ва Edit2 киритиш-таҳрирлаш майдонларидан олинади ҳамда қўшиб қўйиш тугмаси чертилганда рўйхатга **Фамилия** майдони бўйича тартибни сақлаган ҳолда қўшиб қўйилади.

9.5-листинг. Тартибланган рўйхатга янги элемент қўшиш

```
unit Tar_Din_ruy;
interface
uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,   Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Edit1: TEdit;
    Label4: TLabel;
    Edit2: TEdit;
    Button1: TButton;
    Button2: TButton;

    procedure Button1Click(Sender: TObject);
```

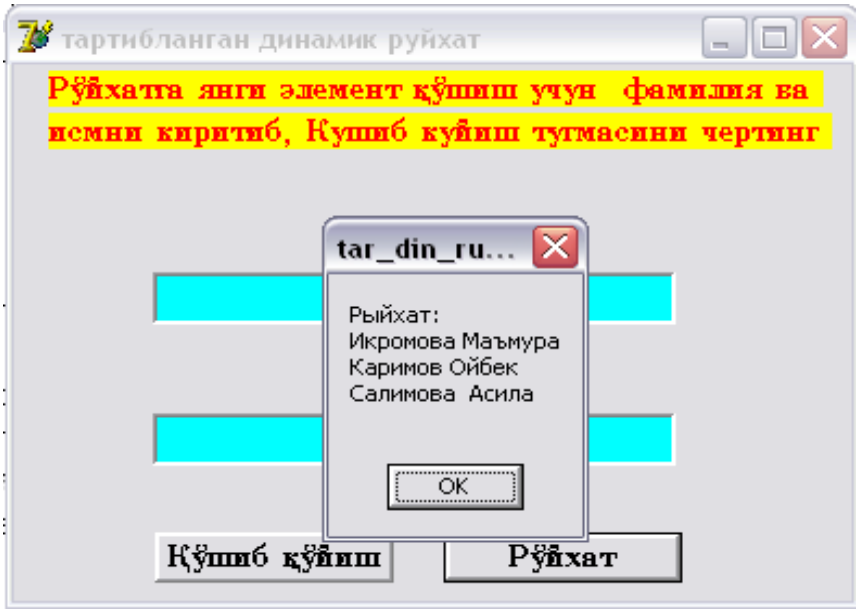
```
    procedure Button2Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  private    { Private declarations }
  public     { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}

type
TPStudent = ^TStudent;//TStudent типидаги кўрсаткич
TStudent = record
f_name:string[20];//фамилия
l_name: string[20];//исми
next: TPStudent;//рўйхатнинг кейинги элементи
end;
var
head: TPStudent;//рўйхатнинг бошланиши

//рўйхатнинг бошига янги элемент қўшиш
procedure TForm1.Button1Click(Sender: TObject);
var  node: TPStudent;//рўйхатнинг янги тугуни
      curr: TPStudent;//рўйхатнинг жорий тугуни
      avv: TPStudent;//curr дан олдин турадиган тугун
begin
new(node);//рўйхатнинг янги элементини ҳосил қилиш
node^.f_name := Edit1.Text;//фамилия
node^.l_name := Edit2.Text;//исм
//тугунни ðûéðàòга қўшиш
//дастлаб рўйхатдан тугунга мос жойни аниқлаймиз
Curr := head;
avv := NIL;

{ Диққат! Агар қуйидаги шартни
(node.f_name > curr".f_name) and (curr <> NIL)

билан алмаштирилса бажарий вақтидаги ҳатолик рўй беради. Чунки,  curr = NIL ва шу сабабли  curr.^name ўзгарувчи мавжуд эмас. Шартнинг фойдаланилган вариантида эса дастлаб қиймати FALSE бўлган  (curr <> NIL) шарти текширилмоқда. Бу ҳолда иккинчи шарт текширилмайди.}

while (curr <> NIL) and (node.f_name > curr^.f_name) do
begin
//киритилган қиймат жорий элементдан катта
avv := curr;
Curr := curr^.next;//кейинги тугунга
end;
if avv = NIL then begin
//янги элементни  рўйхат бошига ўтказиш
node^. Next := head; head := node;
end
else begin
//янги тугун avv дан кейин, аммо curr дан олдин
node^.next := avv^.next;
avv^.next := node;
end;
Edit1.text := '';   Edit2.text := '';
Edit1.SetFocus;
```

```
end;

//рўйхатни экранга чиқариш
procedure TForm1.Button2Click(Sender: TObject);
var  curr: TPStudent;//рўйхатнинг жорий элементи
n:integer;//рўйхат (элементларнинг)  сони
st:string;//рўйхатнинг сатрли кўриниши
begin n := 0; st := '';
curr := head;//рўйхатнинг биринчи элементи
while curr <> NIL do begin
n := n + 1;
st := st + curr^.f_name + ' ' + curr^.l_name + #13;
curr := curr^.next;//рўйхатнинг кейинги элементи
end;
if n <> 0
then ShowMessage('рўйхат:' + #13 + st)
else ShowMessage('рўйхатда элементлар йўқ.');
end;
procedure TForm1.FormActivate(Sender: TObject);
begin
head := NIL;//рўйхат бўш
end;
end.
```

Дастурни ишга тушириш

Tform1.Button1Click процедураси динамик ўзгарувчи-ёзувни ҳосил қилади, унинг майдонларига дастур диалог ойнасидаги маълумотларни қиймат қилиб беради, бу тугун-маълумот учун тартибланган рўйхатдан жой аниқлайди ва янги элементдан аввал турган тугуннинг next кўрсаткичига тузатмалар киритиб рўйхатга қўшади.

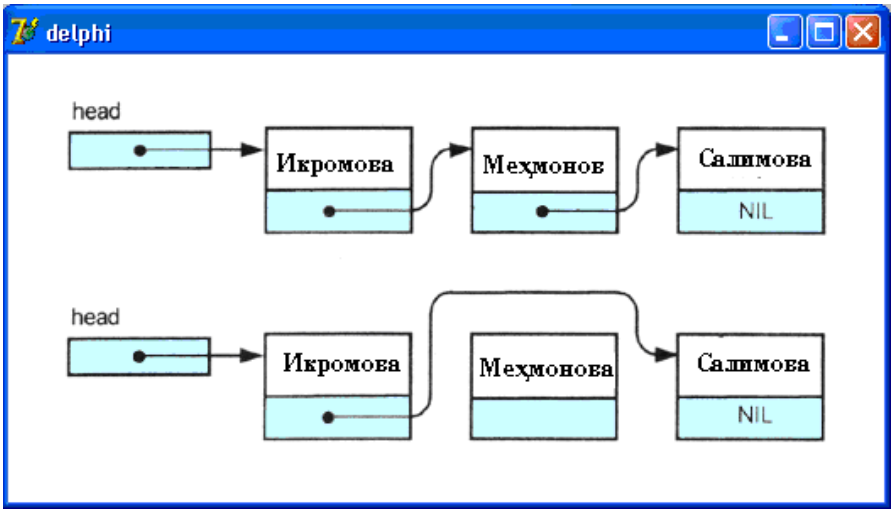
Рўйхатни экранга Tform1.Button2Click процедураси чиқаради. Дастур ишга тушганидан сўнг, фамилиялар, масалан, Каримов, Салимова, Икромова тартибида киритилганидан кейин, рўйхатнинг кўриниши 9.11-расмда келтирилган.



9.11-расм. Дастур тартиблаган рўйхатдан намуна

9.8. Элементларни рўйхатдан ўчириш

Тугунни рўйхатдан ўчириш учун ўчирилиши талаб қилинган тугундан олдин турган тугуннинг кўрсаткичига тузатмалар киритиш лозим. (9.12-расм)



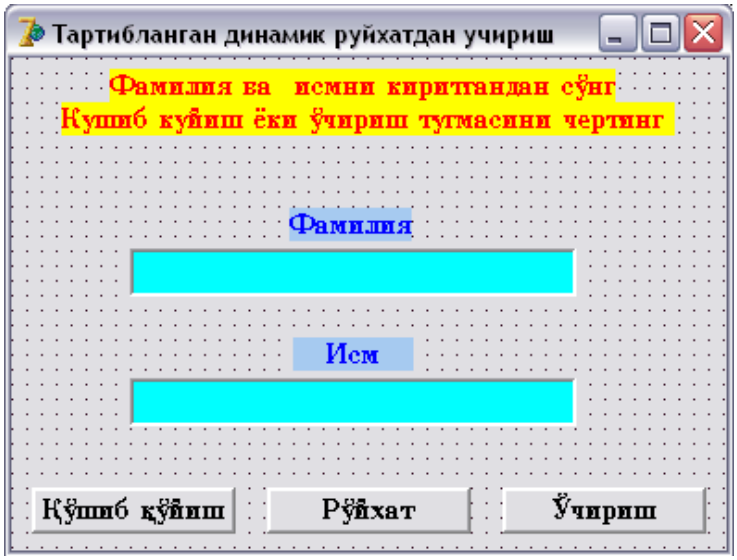
9.12-расм. Элементни рўйхатдан ўчириш

Тугун динамик ўзгарувчи бўлгани учун бу тугунни рўйхатдан ўчирилганидан сўнг, шу тугуннинг хотирадан банд қилган жойи ҳам бўшатилиши шарт. Динамик хотирани бўшатиш (ёки динамик ўзгарувчини йўқотиш) dispose процедураси ёрдамида амалга оширилади. Dispose процедурасида битта параметр — динамик ўзгарувчига кўрсаткич қатнашади ҳалос. Бу динамик ўзгарувчи банд қилган жойни бўшатилиши керак. Масалан,

```
Var  p: ^integer;
begin
new(p);
{ дастурнинг буйруқлари}
dispose(p);
end
```

дастурида p - динамик ўзгарувчи ҳосил қилинади, сўнгра у йўқотилади. Хотиранинг бўшаган қисмидан бошқа ўзгарувчилар учун фойдаланилиши мумкин. Агар динамик ўзгарувчилар банд қилган хотира қисмлари бўшатилмаса, дастур маълум бир муддат ишлаганидан сўнг, навбатдаги динамик ўзгарувчи учун жой етмай қолиши мумкин.

Қуйидаги дастур тартибланган рўйхатга тугунларни қўшиш ва ўчиришга имкон беради. Бу дастурнинг диалог ойнаси 9.13-расмда берилган.



9.13-расм. Динамик рўйхатдан ўчириш дастурининг диалог ойнаси

Тугунни рўйхатга қўшиш, экранга чиқариш, рўйхат тугунларини эълон қилиш процедуралари **Тартибланган динамик рўйхат2** дастуридаги мос процедуралар билан бир хил бўлгани учун, уларни келтириб ўтирмаймиз.

Тугунни рўйхатдан ўчириш амалини TForm1.Button3Click процедураси бажаради. У **Ўчириш** (Button3) тугмасини чертилиши билан ишга тушади. Бу процедуранинг матни 9.6-листингда берилган.

9.6-листинг. Тугунни рўйхатдан ўчириш

```
procedure TForm1.Button3Click(Sender: TObject);
var    curr:TPStudent;//жорий, текширилаётган тугун
       pre: TPStudent;//аввалги тугун
       found:boolean;//TRUE – ўчириладиган тугун рўйхатда бор
begin
  if head = NIL then
    begin
      MessageDlg('Рўйхат бўш!', mtError,[mbOk],0);
      Exit;
    end;
  curr := head;  //жорий тугун - биринчи тугун
  pre:=NIL;    //ундан олдин тугунлар йўқ
  found := FALSE;
 //Ўчирилиши талаб қилинган тугунни топиш
  while (curr <> NIL) and (not found) do
     begin
     if (curr^.f_name = Edit1.Text) and (curr^.l_name = Edit2.Text)
         then found := TRUE //керакли  тугун топилди
          else //навбатдаги тугунга
             begin pre := curr;  curr := curr^.next; end;
     end;
     if found then begin
         //керакли тугун топилди
          if MessageDlg('тугун рўйхатдан ўчирилади!',
                    mtWarning,[mbOk,mbCancel],0) <> mrYes
            then Exit;
         //тугун ўчирилмоқда
           if pre = NIL
                 then  head := curr^.next
                //рўйхатдаги биринчи тугун ўчирилмоқда
            else pre^.next := curr.next;
           Dispose(curr);
           MessageDlg('Тугун' + #13 +
                 'Исми:' + Edit1.Text + #13 +
                 'Фамилияси:' + Edit2.Text + #13 +
                 'рўйхатдан ўчирилди.',
                 mtInformation,[mbOk],0);
       end
     else//ўчириш сўралган тугун рўйхатда йўқ
         MessageDlg('Узел' + #13 +  'Исми:'+ Edit1.Text + #13 +
          'Фамилияси:' + Edit2.Text + #13 + 'рўйхатда топилмади.',
                 mtError,[mbOk],0);
     Edit1.Text := '';  Edit1.Text := '';  Edit1.SetFocus;
end;
```

Дастурни ишга тушириш

Бу процедура рўйхатни бошидан-оёқ кўриб чиқади ва диалог ойнасининг киритиш майдонидаги ўчириладиган тугунни бор ёки йўқлигини текшириб чиқади. Агар мавжуд бўлса, у ҳолда фойдаланувчидан бу тугунни ўчиришга ижозат сўрайди. Агар фойдаланувчи ОК тугмасини босиб, бу амални тасдиқласа, процедура уни ўчиради. Агар ўчириладиган маълумот рўйхатдан топилмаса, дастур бу ҳақида экранга ахборот чиқаради.

10-боб. **ОБЪЕКТЛИ ЙЎНАЛТИРИЛГАН ДАСТУРЛАШГА КИРИШ**

10.1. Кириш

Азалдан дастурлаш процедурали дастурлаш асосида юзага келди ва ривожланди. Бу дастурлашнинг моҳияти алгоритм, процедура ва маълумотларни қайта ишлашдан иборат.

Объектли йўналтирилган дастурлаш (ОЙД) – бу шундай дастурлар ишлаб чиқариш усулики, унинг асосида объект ётади. Объект — бу борлиқ оламнинг бирор объектига мос келадиган бирор тузилмадан иборат. ОЙД усули билан ечиладиган масала объектларнинг атамалари ва уларнинг устида бажариш мумкин бўлган амаллар ёрдамида ифодаланади. Шунинг учун бу усулда ёзилган дастур объектлар ва улар ўртасидаги муносабатларни ўз ичига олади.

Эслатма: Шуни айтиш жоизки. Delphi да компоненталар базаси ёрдамида иловалар ишлаб чиқиш учун ОЙД концепцияларини билиш шарт эмас. Аммо, бу бобдаги материаллар дастур ўз компоненталари билан қандай муносабатда бўлишини ҳамда Delphi нимани ва нима учун дастур матнига қўшишини чуқур тушуниш учун фойдали бўлади.

менюга

10.2. Класс

Классик Pascal тили дастурчиларга ўзларининг мураккаб маълумот типларини – ёзувларни аниқлашга имкон беради. Delphi тили дастурлашнинг ОЙД концепциясига мувофиқ классларни аниқлаши мумкин. Класслар – бу мураккаб структура бўлиб, ўз ичига маълумотларни, процедура ва функцияларни ифодалашдан ташқари, классларнинг вакили бўлмиш объектлар устида бажарилиши мумкин бўлган амалларни ҳам олади. Энг содда классни эълон қилишга намуна қуйидагича :

```
TPerson = class
  private
  fname: string[15]; faddress: string[35];
  public
  procedure Show;
  end;
```

Классдаги маълумотлар майдонлар, процедура ва функциялар эса методлар деб аталади. Юқоридаги мисолда TPerson — класс номи, fname ва faddress – майдон номлари, show — метод.

Эслатма: Delphi да қабул қилинган келишувга биноан майдонларнинг номлари f харфидан (field-майдон сўзидан) бошланиши шарт.

Классларни ифодалаш дастур матнида типларни эълон қилиш бўлимида (type) жойлаштирилади.

менюга

10.3. Объект

Объектлар классларнинг вакили сифатида дастурда var бўлимида эълон қилинади. Масалан:

```
Var student: TPerson; professor: TPerson;
```

Эслатма: Delphi да объект – бу динамик структура. Объект- ўзгарувчининг қиймати маълумотлардан эмас, балки объектдаги маълумотларга мурожаатга тенг бўлади. Шунинг учун дастурчи бундай маълумотлар учун хотирадан жой ажратиш ҳақида қайғуриши лозим.

Хотирадан жой ажратиш класснинг махсус методи – одатда Create (яратилсин) номи бериладиган конструктор ёрдамида амалга оширилади. Конструкторнинг махсус роли ва ҳулқини алоҳида таъкидлаш учун классларни ёзишда одатдаги procedure сўзи ўрнига constructor сўзидан фойдаланилади. Қуйидаги мисолда таркибига конструктор киритилган Tperson классини эълон қилиш келтирилган:

```
TPerson = class private
   fname: string [ 15 ];
   faddress: string[35];
   constructor Create;// конструктор
   public
   procedure show;// метол
   end;
```

Объектнинг маълумотлари учун хотирадан жой ажратиш метод-конструкторни қўллаш натижасининг қийматини объектнинг (класснинг) типига ўзлаштириш йўли билан амалга оширилади. Масалан,

professor := TPerson.Create;

кўрсатмаси бажарилганидан сўнг, professor объектининг маълумотлари учун хотирадан етарли жой ажратилади.

Конструктор хотирадан жой ажратишдан ташқари, объектнинг майдонларига бошланғич қийматлар бериш вазифасини ҳам бажаради, яъни объектни инициализация қилади. Қуйидаги мисолда конструкторни Tperson объектига қўллаш намунаси келтирилган: :

```
constructor TPerson.Create;
  begin
  fname := '';
  faddress := '';
  end;
```

Конструкторни қўллаш бир оз бошқачароқ. Биринчидан, танасида одатдаги динамик хотирадан жой ажратувчи New процедураси йўқ (хотирадан жой ажратиш бўйича ҳамма ишларни компиляторнинг ўзи бажаради). Иккинчидан, дастурда конструкторга метод-функцияга мурожаат каби мурожаат қилинсада, конструктор расман қийматларни қайтармайди.

Объектдан фойдаланиш мумкин бўлиши учун аввал эълон, сўнгра инициализация қилинади. Шундан кейин, объект майдонларининг қийматларини аниқлаш мумкин. Объектнинг майдонига мурожаат қилиш учун объектнинг номи ва майдоннинг номи бир-биридан нуқта билан ажратилган ҳолда ёзилади. Объект кўрсаткич булгани билан, ^ белгисини қўйилмайди. Масалан, professor объектининг fname майдонига мурожаат қилиш учун одатдаги professor^.fname ёзуви ўрнига

professor.fname

деб ёзилади.

Агар дастурда бирор объект кейинчалик фойдаланилмайдиган бўлса, бу объектнинг майдонларини хотирадан банд қилган жойлари бўшатилиши мумкин. Бу масала Free метод-деструктори ёрдамида ҳал қилинади. Масалан, professor объекти майдонларининг хотирадан эгаллаган жойлари

professor.Free;

кўринишидаги кўрсатма ёрдамида бўшатилади.

менюга

10.4. Метод

Класснинг методлари (классни эълон қилишда кўрсатилган процедура ва функциялар) класснинг объектлари устида амаллар бажаради. Метод бажарилиши учун объектнинг номи ва методнинг номлари бир-биридан нуқта билан ажратиб кўрсатилади. Масалан,

professor. Show;

кўрсатмаси show методини professor объектига нисбатан қўлланишини англатади. Амалий жиҳатдан, методни объектга нисбатан қўллаш процедураларга мурожаатнинг махсус ёзиш усулидир.

Класснинг методлари дастурда оддий процедура ва функциялар каби аниқланади. Фарқи шуки, унинг номи икки қисмдан иборат бўлади: метод тегишли бўлган объектнинг номи ҳамда методнинг номи. Улар бир-биридан нуқта билан ажратилади. қуйидаги мисолда TPerson классининг show методини аниқлашга намуна келтирилган:

```
//TPerson классининг show методи
procedure TPerson.Show;
begin
ShowMessage( 'Имя:' + fname + #13 + 'Адрес:' + faddress );
end;
```

Эслатма: Методнинг буйруқларида объектнинг майдонларига мурожаат қилиш объект номини кўрсатмасдан амалга оширилади.

менюга

10.5. Объектнинг ҳусусиятлари ва инкапсуляцияси

Инкапсуляция деганда объект номини фақат класснинг методлари ёрдамида мурожаат қилинишини таъминлаш мақсадида яшириш тушунилади.

Delphi тилида объектнинг майдонларига мурожаат қилишдаги чеклов объект ҳусусиятини кўрсатиш билан амалга оширилади. Объектнинг ҳусусияти ҳусусият қийматини олган майдон ҳамда ҳусусият майдонларига мурожаат қилиш учун иккита метод билан характерланади. Ҳусусият қийматларини аниқлаш ҳусусиятн ёзиш усули (write) деб аталади, ҳусусият қийматини олиш усули эса ҳусусиятни ўқиш усули (read) дейилади.

Классни эълон қилишда ҳусусият номидан аввал property (ҳусусият) сўзи ёзилади. Ҳусусият номидан кейин унинг типи ва ҳусусият қийматларига мурожаат қилиш методларининг номлари кўрсатилади. Read сўзидан кейин ҳусусиятни ўқиш методининг номи, write дан кейин эса ҳусусиятни сақлаш методининг номи ёзилади. Қуйидаги мисолда иккита Name ва Address ҳусусиятларига эга бўлган Tperson классининг эълони келтирилмоқда:

```
type
TName = string[15];
TAddress = string[35];
TPerson = class//класс
private
FName: TName;// Name ҳусусиятининг қиймати
FAddress: TAddress;//Address ҳусусиятининг қиймати
Constructor Create(Name:Tname);
Procedure Show;
Function GetName: TName;
Function GetAddress: TAddress;
Procedure SetAddress(NewAddress:TAddress);
public
Property Name: Tname// Name ни ўқиш
read GetName;//фақат ўқиш мумкин ҳалос
Property Address: TAddress// Address ҳусусияти
read GetAddress//фақат ўқиш мумкин ҳалос
write SetAddress;// фақат ёзиш мумкин ҳалос
end;
```

Дастурда ҳусусият қийматларини кўрсатиш учун объектга нисбатан ҳусусият қийматларини аниқлаш методини ёзиш керак эмас, ҳусусиятга қиймат беришнинг оддий ёзувидан фойдаланиш лозим. Масалан, student объектининг Address ҳусусиятига қиймат бериш учун

```
student.Address := 'Наманган шаҳри, Навоий кўчаси, 25-уй' ;
```

деб ёзиш етарли.

Компилятор ҳусусиятга қиймат бериш буйруғини

```
student.SetAddress('Наманган шаҳри, Навоий кўчаси, 25-уй' ;
```

методига мурожжат тарзида қайта трансляция қилади.

Ташқи кўринишдан, ҳусусиятларни дастурда қўллаш объектнинг майдонларидан фойдаланишдан фарқ қилмайди. Аммо, ҳусусият ва объект майдони ўртасида каттагина фарқ мавжуд: ҳусусиятларга қиймат бериш ва ўқишда кўрсатилган вазифани бажарувчи процедура автоматик тарзда чақирилади.

Дастурда методларга айрим қўшимча вазифаларни ҳам топшириш мумкин. Масалан, Масалан, метод ёрдамида ҳусусиятларга берилаётган қийматларнинг тўғрилигини назорат қилиш, ҳусусият билан мантиқан боғланган бошқа майдонларга қийматлар бериш, ёрдамчи процедураларни чақириш каби масалаларда фойдаланиш мумкин.

Объектнинг маълумотларини ҳусусият кўринишида расмийлаштириш объект ҳусусиятлари қийматларини ўзида сақлаётган майдонларга мурожаат қилишни чеклашга имкон беради, масалан, майдондаги маълумотларни фақат ўқишга рухсат бериш мумкин. Дастурнинг буйруқлари ҳусусият қийматларини ўзгартириб қўймаслиги учун, ҳусусиятни кўрсатишда фақат ўқиш методидан фойдаланиш лозим. Фақат ўқиш учун мўлжалланган ҳусусият қийматини ўзгартиришга уриниш компиляция вақтидаги

ҳатоликни юзага келтиради. TPerson класси учун юқорида келтирилган эълонда Name ҳусусияти фақат ўқиш учун, Address — ҳусусияти эса ўқиш ва ёзиш учун мўлжалланган.

Ёзишдан химояланган ҳусусият қийматини аниқлаш масаласи объект инициализацияси вақтида ҳал қилиниши мумкин. Қуйида TPerson класси объектларини яратиш ва унинг ҳусусиятларига мурожаат қилиш учун TPerson классининг методлари келтирилган:

```
//TPerson объектининг конструктори
Constructor TPerson.Create(Name:TName);
begin
Fname := Name;
end;
// Name ҳусусияти қийматини олиш методи
Function TPerson.GetName;
begin
Result := FName;
end;
// Address ҳусусияти қийматини олиш методи
function TPerson.GetAddress;
begin
Result := FAddress;
end;
// Address ҳусусияти қийматини ўзгартириш методи
Procedure TPerson.SetAddress(NewAddress:TAddress);
begin
if FAddress =' '
then FAddress := NewAddress;
end;
```

TPerson объектининг келтирилган конструктори объект яратади ва Name ҳусусиятининг қийматини аниқловчи Fname майдонини белгилайди. TPerson класси объектларини яратишни таъминловчи ва унинг ҳусусиятларини ифодаловчи дастурнинг буйруқлари қуйидагича бўлиши мумкин:

```
student := TPerson.Create('Абдуллаев');
student.Address := 'Навоий кўчаси , 3-уй, кв.25';
```

менюга

10.6. Ворислик

ОЙД концепцияси янги классларни мавжуд классларга янги майдонлар, ҳусусиятлар ва методларни қўшиш орқали яратиш имконини ҳам беради. Янги классларни ташкил қилишнинг бундай усули юзага келтириш деб аталади. Бу ҳолда янги, юзага келган (насл) класс ўзининг базавий ота классининг ҳусусият ва методларини мерос олади.

Насл-классни эълон қилишда отасининг класси кўрсатилади. Масалан, TEmployee (ходим) класси биз юқорида кўрган TPerson классига FDepartment (бўлим) майдонини қўшиш орқали ҳосил қилиниши мумкин. Бу ҳолда TEmplioyee классини эълон қилиш қуйидагича ёзилиши мумкин:

```
TEmployee = class(TPerson)
  FDepartment: integer;//бўлим номери
  constructor Create(Name:TName; Dep:integer);
  end;
```

Қавслар ичига олинган TPerson номли класс TEmployee класси Tperson классининг ҳоисласи сифатида юзага келганлигини эканлигини англатади. Ўз навбатида TPerson класси TEmployee учун базавий класс ҳисобланади. .

TEmpioyee класси ўз шахсий конструкторига эга бўлиши лозим. У ота-классни ҳамда ўз майдонларининг инициализациясини таъминлаш учун хизмат қилади. Намуна сифатида қуйидаги TEmployee конструктори классини кўриш мумкин:

```
constructor TEmpioyee.Create (Name : Tname; Dep : integer);
begin
inherited Create(Name);
Fdepartment := Dep;
end;
```

Бу келтирилган мисода inherited буйруғи билан ота-классининг конструктори чақирилади. Шундан сўнг, насл-класснинг майдонига қиймат берилиши мумкин.

Янги классни ҳосил қилиш жараёни тугаганидан кейин дастурда ота классининг метод ва майдонларидан фойдаланиш мумкин. Қуйидаги дастур парчасида ана шу имконият кўрсатилган.

```
engineer := TEmployee.Create('Абдуллаев',413);
engineer.address := 'Навоий кўчаси, 3-уй, кв.25';
```

Бу ердаги биринчи кўрсатма Temploуee ти пидаги объектни яратади, иккинчиси эса — ота классига мансуб бўлган ҳусусият қийматини белгилайди.

менюга

10.7. Protected ва private директивалари

Класс элементларини (майдонлар, методлар, ҳусусиятлар) эълон қилишдан ташқари эълон одатда protected (химояланган) ва private (ёпилган) директиваларини ҳам ўз ичига олади. Улар класс элементларини дастурда кўриниш даражаларини белгилаб беради.

Protected секциясида эълон қилинган объектлардан фақатгина улардан ҳосил қилинган класслардагина фойдаланиш мумкин. Бу секциянинг класслари элементларининг кўриниш соҳалари класснинг эълони жойлашган модул билан чегараланиб қолмайди. Одатда protected секциясига класслар методларининг эълонлари жойлашади.

Private секциясида эълон қилинган класснинг элементлари фақат модулнинг ичидагина кўринади. Бу элементлардан модулдан ташқарида, ҳаттоки шу класслардан ҳосил қилинган классларда ҳам фойдаланиб бўлмайди. Одатда private секциясига класснинг майдонларини эълонлари жойлаштирилади, бу майдонлардан фойдаланиш ҳуқуқини берадиган методлар эса protected секциясида жойлаштирилади.

Қуйида фойдаланиш ҳуқуқини бошқарадиган кўрсатмаларни ўз ичига олган Tperson классининг эълони келтирилмоқда:

```
TPerson = class private
FName: TName;// Name ҳусусиятининг қиймати
FAddress: TAddress;// Address ҳусусиятининг қиймати
protected
Constructor Create(Name:TName);
Function GetName: TName;
Function GetAddress: TAddress;
Procedure SetAddress(NewAddress:TAddress);
Property Name: TName
read GetName;
Property Address: TAddress
read GetAddress
write SetAddress;
end;
```

Эслатма: Айрим ҳолларда класс элементларини тўла яширишга тўғри келади. Бундай ҳолларда классни алоҳида модулда эълон қилиниши лозим. Бу класснинг объектларидан фойдаланувчи дастурда эса модулга мурожаат қилинади.

менюга

10.8. Полиморфизм ва виртуал методлар

Полиморфизм — бу турли классларга кирган методлар учун бир ҳил номлардан фойдаланиш имкониятидир. Полиморфизм концепциясида объектга нисбатан метод қўлланганида айнан объектнинг

классига мос келувчи методдан фойдаланишни таъминлайди.

Учта, бири қолган иккиси учун базавий бўлган класслар аниқланган бўлсин:

```
type
  // базавий класс
   TPerson = class
   fname: string;// ном
   constructor Create(name : string);
   function info : string;
   virtual;
   end;
// TPerson дан ҳосил қилинган
   TStud = class(TPerson)
   Fgr : integer;// ўқув гуруҳининг номери
   constructor Create(name : string; gr : integer);
   function info: string;
   override;
   end;
// TPerson дан ҳосил қилинган
   TProf = class(TPerson)
   fdep:string;// кафедранинг номи
   constructor Create(name : string; dep : string);
   function info : string;
   override;
   end;
```

Бу классларнинг ҳар бирида info методи қатнашмоқда. Базавий классда virtual директиваси ёрдамида info методи виртуал деб эълон қилинган. Методни виртуал деб эълон қилиш насл классларда виртуал методларни ўзининг шахсий методлари билан алмаштиришга имкон беради. Ҳар бир насл классда ўзининг info методи аниқланган, ва улар ота классидаги мос методнинг ўрнини босади. (ота класидаги виртуал методларнинг ўрнини боса оладиган насл класснинг методлари override директиваси билан кўрсатилади). Қуйида ҳар бир класс учун info методининг аниқланиши кўрсатилган:

```
function TPerson.info : string;
begin
result := '';
end;
function TStud.info : string;
begin
result := fname + ' гр.' + IntTostr(fgr);
end;
function TProf.info : string;
begin
result := fname + ' каф.' + fdep;
end;
```

Ҳар икки класс битта базавий классдан ҳосил қилингани учун талабалар ва ўқитувчилар рўйхатини қуйидагича эълон қилиш мумкин (объект — бу кўрсаткич эканлигини эсга олинг):

list: **array**[l..SZL] of TPerson;

Delphi тили ота классидаги кўрсаткич учун насл класси кўрсаткичининг қийматини бера олгани учун, рўйхатни юқоридаги тарзда эълон қилишга имкон беради. Шунинг учун list массивининг элементлари Tstud классининг объектлари ҳам, Tprof классидаги объектлар ҳам бўла олади.

Талабалар ва ўқитувчилар рўйхатини экранга массив элементларига нисбатан info методини қўллаган ҳолда чиқарилиши мумкин. Масалан. Мана бундай қилиб:

```
st := '';
for i:= l to SZL do// SZL - массив-рўйхатнинг элементлар сони
if list[i] <> NIL
then st := st + list[i].Info + #13;
ShowMessage (st);
```

Дастурнинг иши давомида массивнинг ҳар бир элементи Tstud типидаги объектларни ҳам, Tprof типидаги объектларни ҳам олиши мумкин. Полиморфизм концепцияси айнан ҳар бир объектнинг типига мос бўлган методни қўлланишини таъминлайди.

Қуйидаги дастур, юқорида эълон қилинган TPerson, TStud ва Tprof классларидан фойдаланиб, талаба ва ўқитувчилар рўйхатини ҳосил қилади ва экранга чиқаради. Дастурнинг матни 10.1-листингда, диалог ойнаси эса 10.1-расмда келтирилган.

10.1-расм. Полиморфизм дастурининг диалог ойнаси

10.1-листинг. Полиморфизм

```
 unit polimor_;
 interface
 uses   Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,   StdCtrls;
 type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    GroupBox1: TGroupBox;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    Label1: TLabel;
    Label2: TLabel;
    Button1: TButton;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  Public
    { Public declarations }
  end;
 type

  //базавий класс
   TPerson = class
         fName: string;//ном
         constructor Create(name:string);
         function info:string;
       virtual;
   end;

  //Талаба класси
```

```
  TStud = class(TPerson)
        fGr : integer;//гуруҳ номери
        constructor Create(name:string;gr:integer);
        function info:string;
      override;
    end;

 //Ўқитувчи класси
  TProf = class(TPerson)
        fdep:string;//кафедра номи
        constructor Create(name:string;dep:string);
        function info:string; override;
  end;
const   SZL = 10;   //рвйхат хажми
var
 Form1: TForm1;
 List: array[1..SZL] of TPerson;//рўйхат
 n:integer = 0;              //рўйхатдаги одамлар сони
implementation
{$R *.DFM}

constructor TPerson.Create(name:string);
   begin
      fName := name;
   end;

constructor TStud.Create(name:string;gr:integer);
   begin
   inherited create(name);//базавий класс конструкторини чақириш
   fGr := gr;
   end;

constructor TProf.create(name:string; dep:string);
  begin
  inherited create(name);//базавий класс конструкторини чақириш
  fDep := dep;
  end;

function TPerson.Info:string;
  begin
   result := fname;
  end;

function TStud.Info:string;
  begin
    result := fname + ' гр.' + IntToStr(fGr);
  end;

function TProf.Info:string;
  begin
    result := fname + ' каф.' + fDep;
  end;

//Қўшиб қўйиш тугмаси чертилганда
procedure TForm1.Button1Click(Sender: TObject);
  begin
    if n < SZL then
      begin     //объектн рўйхатга қўшиш
        N := n + 1;
        if Radiobutton1.Checked
           then//TStud объектини яратамиз
          List[n] := TStud.Create(Edit1.Text,StrToInt(Edit2.Text))
```

```
            else // TProf объектини яратамиз
               List[n] := TProf.Create(Edit1.Text,Edit2.Text);
          // киритиш майдони тозаланмоқда
        Edit1.Text := '';
        Edit2.Text := '';
        Edit1.SetFocus; // курсорни Фамилия майдонига ўтказилмоқда
      end
    else ShowMessage('рўйхат тўлди!');
  end;

  procedure TForm1.Button2Click(Sender: TObject);
  var
    i:integer; // индекс
    st:string; // рўйхат
  begin
    for i := 1 to SZL do
      if list[i] <> NIL then st := st + list[i].info + #13;
    ShowMessage('рўйхат' + #13 + st);
  end;
  end.
```

Дастурни ишга тушириш

Tform1.Button1Click процедураси **Қўшиб қўйиш** (Button1) тугмаси чертилганда ишга тушади ва Tsud ёки Tprof классидаги list[n] объектини ҳосил қилади. Яратилаётган объектнинг класси RadioButton ўчиргичларининг ҳолати билан аниқланади. Ўчиргич **Талаба** (RadioButton1) ҳолатида бўлса Tstud класси, **Ўқитувчи** (RadioButton2) — ҳолатида эса Tprof классини аниқлайди.

TForm1.Button2Click процедураси **Рўйхат** (Button2) тугмаси чертилганда ишга тушиб, рўйхатнинг ҳар бир элементига info методини қўллайди ҳамда умумий рўйхатдан иборат бўлган сатрни ҳосил қилади.

менюга

10.9. Delphi нинг класслари ва объектлари

Delphi интерфейсини қўллаш учун Delphi тили форма ва унинг турли компоненталари (буйруқ тугмалари, киритиш майдонлари ва х.к.) билан ишлаганда қўллаш мумкин бўлган катта сондаги турли туман классларни ўз ичига олган класслар кутубхонасидан фойдаланади.

Илова лойихасини тайёрлаш вақтида Delphi автоматик тарзда дастур матнига керакли барча объектларни қўшиб қўяди. Агар Delphi ишга туширилганидан сўнг, кодлар редактори ойнасини кўрилса, унда қуйидаги сатрларни учратиш мумкин:

```
type
TForm1 = class(TForm)
private
{ Private declarations }
public
{ Public declarations }
end;
var
Form1: Tform1
```

Бу – бошланғич класс, илованинг бўш формаси ва объект-илова формасининг эълонидан иборат.

Дастурчи керакли компоненталарни қўшиб форма яратар экан, Delphi тили форма классини ҳосил қилади. Дастурчи ходисаларни қайта ишлаш процедураси яратаётганда Delphi илова формаси классининг

эълонига методларнинг эълонини ҳам қўшади.
Визуал компоненталарнинг классларидан ташқари, кутубхонага новизуал (кўринмайдиган) компоненталар класслари ҳам киради. Улар мос объектлар яратади ҳамда уларнинг метод ва ҳусусиятларидан фойдаланишга йўл очади. Новизуал компонентага мисол қилиб таймер (Ttimer типини) ва маълумотлар базасини бошқариш компоненталарини олиш мумкин. Яна кўплаб бошқа класслар мавжуд, аммо, уларни кўриб чиқиш ушбу китоб доирасида бизнинг олдимизда турган масалаларга кирмайди.

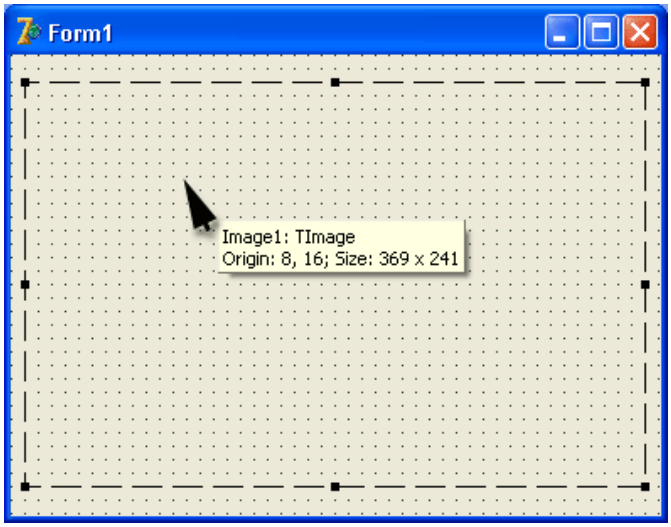
11-боб. **DELPHI НИНГ ГРАФИК ИМКОНИЯТЛАРИ**

Delphi тили дастурчиларга тасвирий воситалар ёрдамида экранга турли чизмалар, расмлар, иллюстрацияларн чиқаришга имкон беради.

Дастур графикани объектнинг (форма ёки Image компонентасининг) сиртига чиқаради. Объектнинг сиртига canvas ҳусусияти мос келади. Объект сиртига бирор график элементни (тўғри чизиқ, айлана, тўғри тўртбурчак ва х.к) чиқариш учун бу объектнинг canvas ҳусусиятига эҳтиёжга қараб метод қўллаш лозим. Масалан, Form1.Canvas.Rectangle (10,10,100,100) буйруғи дастур ойнасида тўғри тўртбурчак чизади.

11.1. Холст

Биз юқорида айтиб ўтганимиздек, дастур графикани чиқариши мумкин бўлган сиртга Canvas ҳусусияти мос келади. Ўз навбатида, canvas ҳусусияти — бу Tcanvas типидаги объект ҳисобланади. Бу типдаги методлар содда график элементларни (нуқта, чизиқлар, айлана, тўғри тўртбурчаклар ва х.к.) ясашга имкон беради. Ҳусусиятлари эса график элементларнинг характеристикаларини (ранг, қалинлиги, чизиқларнинг стиллари, соҳаларга фон бериш усуллари, матнларга шрифтлар ва х.к.) белгилаб беради.



11.1-расм. Холстнинг координаталари

Содда график элементларни чиқариш методлари Canvas ҳусусиятларини қандайдир абстракт холст деб қарайди ва унда чизиш амалларини бажариши мумкин. (canvas – "сирт", "расм чизиш учун холст" деб таржима қилинади.) Холст алоҳида нуқталар-пикеллардан ташкил топган. Пикселнинг ҳолати унинг горизонтал (X) ва вертикал (Y) координаталари билан белгиланади. Чап юқори пикселнинг координаталари (0, 0) га тенг. Координаталар юқоридан пастга қараб, чапдан ўнг томонга қараб ўсади. (11.1-расм). Холст ўнг қуйи нуқтасининг координатаси холстнинг ўлчамларига боғлиқ.

Холстнинг ўлчамларини расмлар (Image) соҳасининг Height ва width ҳусусиятлари ёки форманинг ҳусусиятлари бўлган ClientHeight ва Clientwidth лар ёрдамида аниқлаш мумкин.

менюга

11.2. Қалам ва чўтка

Рассом расмларни одатда қалам ва чўтка ёрдамида чизади. Холст сиртида содда графикани ясаш таъминлайдиган методлар ҳам қалам ва чўткадан фойдаланади. Қалам чизиқлар ва контурларни чизса, чўтка контурлар билан чегараланган соҳаларни бўяш учун қўлланади.

Қалам ва чўткага Pen (қалам) ва Brush (чўтка) ҳусусиятлари мос келади. Улар мос раавишда TPen ва Tbrush типидаги объектлар ҳисобланади. Бу объектларнинг қийматлари экрандаги график элементларнинг кўринишларини белгилаб беради.

Қалам. Қалам нуқта, чизиқлар ва геометрик фигураларни (тўғри тўртбурчак, айлана, эллипслар, ёйлар ва х.к.) чизиш учун ишлатилади. Холст сиртида қалам қолдираётган чизиқнинг кўриниши TPen объектининг ҳусусиятларига боғлиқ. Бу ҳусусиятлар 11.1-жадвалда келтирилган.

TPen объектининг ҳусусиятлари 11.1-жадвал

Ҳусусияти	Мазмуни
Color	Чизиқнинг ранги
Width	Чизиқнинг қалинлиги
Style	Чизиқнинг кўриниши
Mode	Акслантириш режими

Color ҳусусияти қалам билан чизилаётган чизиқ рангини белгилайди. 11.2-жадвалда color ҳусусиятининг қиймати сифатида фойдаланиш мумкин бўлган номланган константалар (Tcolor типи) рўйхати келтирилган.

Color ҳусусиятининг қийматлари 11.2-жадвал

Константа	**Ранг**	**Константа**	**Ранг**
clBlack	Қора	clSilver	Кумуш ранг
clMaroon	Каштан ранг	clRed	қизил
clGreen	Яшил	clLime	Салат ранг
clOlive	Зайтун ранг	clBlue	Мовий
clNavy	Тўқ кўк	clFuchsia	Тўқ қизил
clPurple	Қизил	clAqua	Феруза ранг
clTeal	Кўкимтир яшил	clWhite	Оқ
clGray	Кул ранг		

Width ҳусусияти чизиқ қалинлигини (пикселларда) кўрсатади. Масалан:

Canvas. Pen. Width :=2

буйруғи қалинлиги 2 пиксел бўлган чизиқни англатади.

Style ҳусусияти чизиқнинг кўринишини (стилини) билдиради. Чизиқ узлуксиз ёки узлукли (узуқ-узуқ) кўринишларидан бирида бўлади. 11.3-жадвалда чизиқнинг стилини белгиловчи номланган константалар рўйхати келтирилган. Пунктир чизиғининг қалинлиги 1 пикселдан катта бўла олмайди. Агар Pen.width ҳусусиятининг қиймати 1 дан катта бўлса, у ҳолда пунктир чизиғи узлуксиз кўринишда экранга узатилади.

Pen.width ҳусусиятининг қийматлари 11.3-жадвал

Константа	**Чизиқнинг кўриниши**
psSolid	Узлуксиз чизиқ
psDash	Узун штрихли пунктир чизиқ
psDot	Калта штрихли пунктир чизиқ
psDashDot	Узун ва калта штрихли пунктир чизиқ
psDashDotDot	Битта узун, иккита кисқа штрихли пунктир чизиқ
psClear	Чизиқ кўринмайди. (соҳа чегараси кўринмаслиги керак бўлган ҳолларда қўлланади)

Mode ҳусусияти холстнинг нуқталарининг рангига боғлиқ равишда чизиқнинг нуқталарининг ранги қандай ҳосил қилинишини аниқлайди. Агар кўрсатилмаган бўлса. У ҳолда чизиқлар Pen.Color ҳусусиятининг қийматида кўрсатилган ранг билан чизилади. Аммо, дастурчи фон рангига нисбатан инверсион рангни ҳам белгилаши мумкин. Бундай ҳол фон рангига боғлиқ бўлмаган ҳолда чизиқ тўлалигича кўринишини таъминлайди. 11.4 –жадвалда Pen.Mode ҳусусиятининг айрим қийматлари келтирилган.

Pen.Mode ҳусусияти чизиқ рангига таъсир кўрсатади. 1.4 –жадвал

Константа	Чизиқнинг ранги
pmBlack	Қора, Pen. Color ҳусусияти қийматига боғлиқ эмас
pmWhite	Оқ, Pen. Color ҳусусияти қийматига боғлиқ эмас
pmCopy	Чизиқ ранги Pen. Color ҳусусияти қийматига кўра танланади.
pmNotCopy	Чизиқ ранги Pen. Color ҳусусияти қийматига инверсион рангда бўлади.
pmNot	Чизиқ нуқталарининг ранги чизиқ чизиладиган холст рангига нисбатан инверсион бўлади.

Чўтка. Методлар чўткани (canvas.Brush) ёпиқ соҳаларни (масалан, геометрик фигураларни) чизиш ва бўяш учун ишлатади. Чўтка, объект сифатида иккита ҳусусиятга эга. Улар 11.5-жадвалда берилган.

TBrush (чўтка) ҳусусиятининг қийматлари 11.5-жадвал

Ҳусусияти	маъноси
Color	Ёпиқ соҳани бўяйди
Style	Соҳага фон бериш стили (типи)

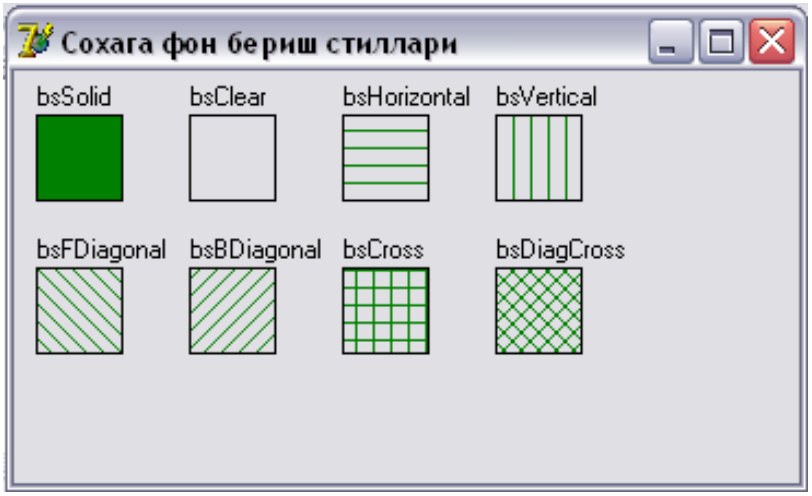
Контурнинг ёпиқ ички соҳаси бўялган ёки фон берилган бўлиши мумкин. 1-ҳолда соҳа тўлалигича фонни бекитади, 2-чисида эса штрихланмаган жойларда фон кўриниб туради. Color ҳусусиятининг қиймати сифатида TColor типидаги ихтиёрий номланган константадан фойдаланиш мумкин. (11.2-жадвалга қаранг)

Соҳага фон бериш стилини белгиловчи номланган константалар 11.6-жадвалда берилган.

Brush.style ҳусусиятининг қийматлари 11.6-жадвал

Константа	Соҳани тўлдириш типи (фони)
bsSolid	Узлуксиз фон билан тўлдириш
bsClear	Соҳа бўялмайди
bsHorizontal	Соҳага горизонтал штрихли фон бериш
bsVertical	Соҳага вертикал штрихли фон бериш
bsFDiagonal	Соҳага олд томонга қийшайган штрихли фон бериш
bsBDiagonal	Соҳага орқага қийшайган штрихли фон бериш
bsCross	Катаксимон штриховкали фон
bsDiagCross	Қийшиқ чизиқли катаксимон фон

Намуна сифатида 11.1-листингда **соҳага фон бериш стиллари** дастури келтирилган. Унинг ёрдамида экранга (11.2-расм) қора рангли ва турли стиллар билан фон берилган саккизта тўғри тўртбурчак тасвири чиқарилади.



11.2-расм. Соҳага фон бериш стиллари дастурининг ойнаси

11.1-листинг. Фон бериш стиллари.

```
unit ctil_;
interface
uses   Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,   Dialogs, ExtCtrls;
type
  TForm1 = class(TForm)
    Image1: TImage;
    procedure Image1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.Image1Click(Sender: TObject);
const
bsName: array[1..8] of string = ('bsSolid','bsClear','bsHorizontal',
'bsVertical','bsFDiagonal','bsBDiagonal', 'bsCross','bsDiagCross');
var
x,y: integer;//тўртбурчак юқори чап бурчагининг координаталари
w,h: integer;//тўғри тўртбурчакнингг кенглиги ва баландлиги
bs: TBrushStyle;//фон бериш стиллари
k: integer;//  фон бериш стили номери
i,j: integer;
begin
w := 40; h := 40;//Тўғри тўртбурчакнинг ўлчамлари
y := 20;
for i := 1 to 2 do
begin
x := 10;
for j := 1 to 4 do
begin
k := j + (i-1)*4;//фон бериш стили номери
case k of
1: bs := bsSolid;
2: bs := bsClear;
3: bs := bsHorizontal;
4: bs := bsVertical;
5: bs := bsFDiagonal;
```

```
 6: bs := bsBDiagonal;
 7: bs := bsCross;
 8: bs := bsDiagCross; end;
 //тўғри тўртбурчакнинг чиқариш
 Canvas.Brush.Color := clGreen;
 //бўяш ранги - яшил
 Canvas.Brush.Style := bs;
 //фон бериш стили
 Canvas . Rectangle (x, y, x+w, y+h) ;
 //стил номерини чиқариш
 Canvas.Brush.Style := bsClear;
 Canvas.TextOut(x, y-15, bsName[k]);
 x := x + w + 30;
 end;
 y := y + h + 30;
 end;
 end;
 end.
```

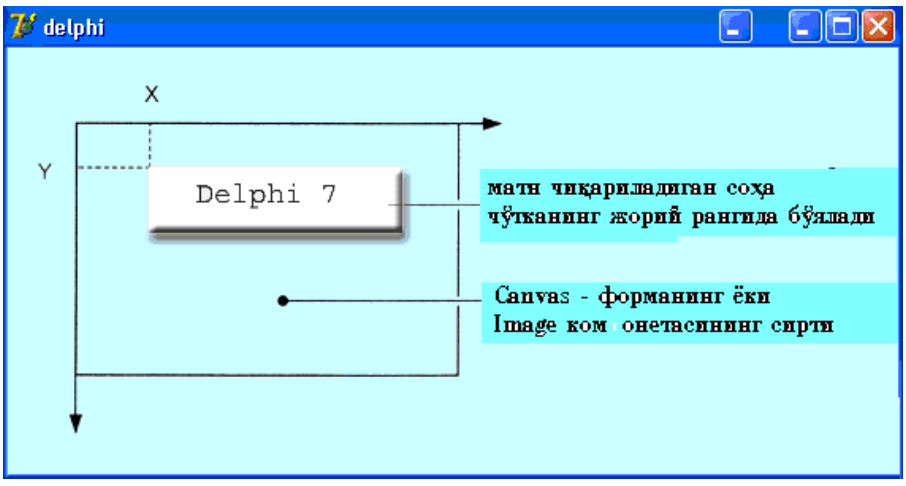
менюга

11.3. Матнларни чиқариш

Матнларни график объектнинг сиртига чиқариш учун TextOut методидан фойдаланилади. Бу методни умумий кўринишда қўйидаги буйруғи билан чақирилши мумкин:

Объект.Canvas.TextOut(x, y, Матн)

Бу ерда ***объект*** — сиртига матн чиқариладиган объектнинг номи; ***x, y*** – график сиртдаги нуқтанинг координаталари (11.3-расм); ***Матн***-белгили типдаги ўзгарувчи ёки константа, унинг қиймати матн сифатида сиртнинг ***(x, y)*** координатали нуқтасидан бошлаб экранга чиқарилади.



11.3-расм. Матн чиқариладиган соҳанинг координаталари

Матнларни экранга чиқариш учун фойдаланиладиган шрифт Canvas объектининг Font ҳусусияти билан белгиланади. 11.7-жадвалда Tfont типидаги объектнинг ҳусусиятлари келтирилган. Улардан TextOut ва TextRect методларида фойдаланиш мумкин.

TFont объектининг ҳусусиятлари 11.7-жадвал

Name	
Size	
Style	Объект. Canvas . Font := [fsBold, fsItalic]
Color	

Диққат!: Матнларнинг чиқариш соҳаси чўтканинг жорий ранги билан бўялган бўлади. Шунинг учун матнни экранга чиқаришдан аввал Brush.Color ҳусусиятига bsClear қийматини бериш ёки чўткага

матн чиқариладиган сиртнинг ранги билан бир ҳил қийматни ўзлаштириш лозим.

Қуйидаги дастур парчаси форма сиртига матнларни чиқариш учун Textout функциясидан фойдаланишга намуна бўла олади.

```
with Form1.Canvas do begin
//шрифтнинг характеристикаларини белгилаш
Font.Name := 'Tahoma';
Font.Size := 20;
Font.Style := [fsltalic, fsBold] ;
Brush.Style := bsClear;//матнни чиқариш соҳаси бўялмайди
TextOut(0, 10, 'Borland Delphi 7');
end;
```

Матнни Textout методи билан сиртга чиқарилганидан сўнг, чиқариш кўрсаткичи (қалам) матн чиқарилган соҳанинг ўнг юқори бурчагига боради.

Айрим ҳолларда дастурни ишилаб чиқиш вақтида узунлиги номаълум бўлган ахборотдан кейин қандайдир матнни чиқаришга тўғри келади. Масалан, бу бирор сонни харфлар орқали ёзувидан кейин чиқарилиши талаб қилинган "сўм" сўзи бўлсин. Бу ҳолда чиқарилган матннинг ўнг томондаги координатасини билиш лозим бўлади. TextOut методи билан чиқарилган матннинг ўнг томондаги координасини PenPos ҳусусияти ёрдамида аниқлаш мумкин. Қуйидаги дастур парчаси матннинг сатрларини иккита TextOut ёрдамида чиқаришни намойиш қилади.

```
with Form1.Canvas do begin
  TextOut(0, 10, 'Borland ') ;
  TextOut(PenPos.X, PenPos.Y, 'Delphi 7');
  end;
```

менюга

11.4. Содда график элементларни чизиш учун методлар

Ихтиёрий расм, чизма ва схемаларни содда график элементларнинг (нуқта, чизиқ, айлана, ёй ва х.к.) тўпламидан иборат деб қараш мумкин. Шунинг учун экранда керакли тасвирни ҳосил қилиш учун дастур бу тасвирни ташкил қилувчи график содда элементларни чизишни (экранга чиқаришни) таъминлаши лозим.

Содда график элементларни компонента сиртида (форма ёки илюстрацияларни чиқариш соҳасида) чизиш масаласи шу компонентанинг Canvas ҳусусиятининг эхтиёжга қараб, мос методларини қўллаш орқали ҳал қилиниши мумкин.

Чизиқлар. Тўғри чизиқларни чизиш учун LineTo методидан фойдаланилади. Бу методни умумий кўринишда қуйидаги усулда чақирилади:

Компонент.Canvas.LineTo(x,y)

LineTo методи қаламнинг жорий позициясидан бошлаб, методга мурожаат қилинганда кўрсатилган координатали нуқтагача бўлган тўғри чизиқни (кесмани) ясайди. Чизиқнинг бошланғич нуқтасини қаламни график сиртнинг керакли позициясига суриш усули билан кўрсатиш мумкин. Бу вазифани ***MoveTo*** методи бажаради. Унинг параметрлари иккита бўлиб, қаламнинг навбатдаги позициясини кўрсатувчи нуқтани англатади. Чизиқнинг кўриниши, (ранги, қалинлиги ва стили) чизиқ чизилиши талаб қилинган график сиртнинг Pen ҳусусиятининг қийматлари билан аниқланади.

Кўпинча, ҳисоблашлар натижаларини графиклар кўринишида ифодалаш қулай ҳисобланади. Тушуниш осон бўлиши ҳамда кўргазмалиликни ошириш учун графикларни координаталар ўқи ва рақамли тўр фонидан фойдаланган ҳолда ясалади. 11.2-листингда форма сиртига координаталар ўқи ва рақамли тўрни чиқарувчи дастурнинг матни келтирилмоқда. (11.4-расм)

11.4-расм. Координата тўри дастурининг ойнаси

11.2-листинг. Координата ўқлари ва рақамланган тўр

```
unit grid_;
interface
uses    Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,   StdCtrls;
type
  TForm1 = class(TForm)
    procedure FormPaint(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.FormPaint(Sender: TObject);
var
 x0,y0:integer;//координата ўқларининг бошланғич нуқтаси
 dx,dy:integer;//координата тўрининг қадами (пикселларда)
 h,w:integer;  // координата тўрининг чиқариш соҳасининг баландлиги ва кенглиги
 x,y:integer;
 lx,ly:real;   //тўр чизиқларини рақамлаш (X ва Y бўйича)
 dlx,dly:real; //тўр чизиқлариниг белгилаш қадами (X ва Y бўйича)
 cross:integer;//рақамланмаган тўр чизиқларининг ҳисоблагич
 dcross:integer;//рақамланганлари орасида рақамланмаган чизиқларнинг миқдори
begin
 x0 := 30; y0 := 220;//ўқлар (40,250) нуқтадан бошланади.
 dx := 40; dy := 40; //координата тўрининг қадами 40 пиксел
 dcross := 1;  //X тўрнинг чизиқларини белгилаш: 1 - ҳар бирини
               //                  2 - биттадан оралатиб
               //                  3 - иккитадан оралатиб
 dlx := 0.5;       //X ўқининг белгилаш қадами
 dly := 1.0;       //У ўқининг белгилаш қадами: 1, 2, 3 ва х.к.
 h := 200;
 w := 300;
 with form1.Canvas do
 begin
   cross := dcross;
   MoveTo(x0,y0); LineTo(x0 , y0-h); //X ўқи
   MoveTo(x0, y0); LineTo(x0 + w, y0); // Y ўқи
```

```
  // X ўқи бўйича бўлаклаш, тўр ва рақамлаш
   x := x0 + dx;
   lx := dlx;
   repeat
     MoveTo(x,y0-3); LineTo(x,y0 + 3);// бўлаклаш
     cross := cross-1;
     if cross = 0 then  // рақамлаш
     begin
       TextOut(x-8, y0 + 5, FloatToStr(lx));
       cross := dcross;
     end;
     Pen.Style := psDot;
     MoveTo(x, y0-3); LineTo(x, y0-h); // тўрнинг чизиқлари
     Pen.Style := psSolid;
     lx := lx + dlx;
     x := x + dx;
   until (x>x0 + w);
  //  Y ўқи бўйича бўлаклаш, тўр ва рақамлаш
   y := y0-dy;
   ly := dly;
   repeat
     MoveTo(x0-3, y); LineTo(x0 + 3,y);   // бўлаклаш
     TextOut(x0-20, y, FloatToStr(ly)); // рақамлаш
     Pen.Style := psDot;
     MoveTo(x0 + 3, y); LineTo(x0 + w, y);   // тўрнинг чизиқлари
     Pen.Style := psSolid;
     y := y-dy;
     ly := ly + dly;
   until (y<y0-h);
  end;
end;
end.
```

Дастурни ишга тушириш

Келтирилган дастурнинг ўзига ҳос томони шундаки, у тўрнинг қадами ва рақамлаш усулини белгилашга имкон беради. Бундан ташқари, дастур X ўқининг тўрининг ҳар бир чизиқларини эмас, балки оралатиб ҳам рақамлай олади. Бунинг сабаби шуки, агар рақамлашда қатнашадиган сонлар бир нечта рақамдан иборат бўлса, рақамлашда қатнашадиган сонларни экранга чиқарганда уларнинг айрим рақамлари бир-бирининг устига тушиб қолиши эҳтимоли олдини олишдан иборат.

Синиқ чизиқ. ***Polyline*** методи синиқ чизиқ чизиш учун хизмат қилади. Бу метод параметр сифатида Tpoint типидаги массивни қабул қилади. Массивнинг ҳар бир элементи ***x*** ва ***y*** майдонли ёзувдан иборат бўлиб, синиқ чизиқнинг синиқ нуқтасини белгилайди. Polyline методи координатаси массивда жойлашган синиқ чизиқнинг синиш нуқталарини кетма-кет кесмалар билан бирлаштиради: биринчини иккинчи билан, иккинчини учинчи билан ва х.к.

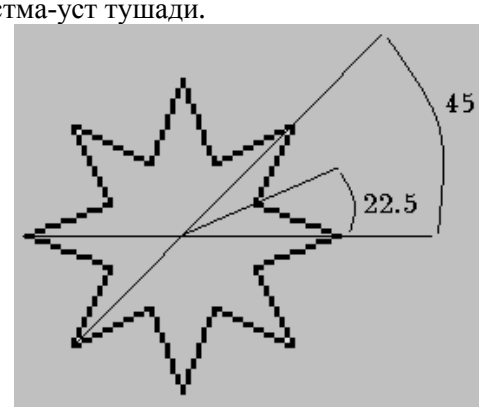
Polyline методидан ёпиқ контурларни чизиш учун ҳам фойдаланиш мумкин. Бунинг учун методнинг параметри сифатида иштирок этаётган массивнинг биринчи ва охирги элементлари бир ҳил бўлиши лозим. Намуна сифатида 11.3-листингда берилган дастурни кўрайлик. Бу дастур диалог ойнаси сиртида сичқонча тугмаси чертилган жойда беш қиррали юлдуз тасвирини (11.5-расм) ясайди. Юлдуз чизилган ранг сичқончанинг қайси тугмаси чертилишига боғлиқ. Сичқонча тугмасини чертиш (MouseDown ходисаси) учун ходисаларни қайта ишлаш процедураси юлдуз чизиш процедураси StarLine ни чақиради ва унга параметр сифатида сичқонча нуқтаси чертилган нуқтанинг координаларини узатади. Юлдузни StarLine процедураси чизади. У параметр сифатида юлдуз марказининг координаталарини ҳамда ҳолстни олади. Дастлаб, юлдузнинг учлари ва чуқурликларининг координаталари ҳисобланади ва бу маълумотлар p массивга ёзилади. Сўнгра бу массив PolyLine методига параметр қилиб узатилади. P массив элементларини ҳисоблашда қутб координаталар системасидан фойдаланамиз. Бунинг учун бурчаклар радианларда ифодаланган бўлиши керак.

11.3-листинг. Ёпиқ контур (юлдуз) чизиш

```
unit Stars_;
interface
uses   Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,   Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    procedure FormMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}

procedure StarLine(x0,y0,r: integer; Canvas: TCanvas);
   //x0,y0 – Юлдуз марказининг координатаси
   //r – юлдузнинг радиуси
var
    p : array[1..17] of TPoint;//учлар чуқурликлар массиви
    a: real;  //Учлар  учун бурчак
     i: integer;
begin
    a := 0;//ўнг учдан ясашни бошлаш учун
    for i := 1 to 16 do
      begin
        if (i mod 2 = 0) then
          begin//чуқурлик координаталари
            p[i].x := x0 + Round(r/2*cos(a*2*pi/360));
            p[i].y := y0-Round(r/2*sin(a*2*pi/360));
          end
        else
          begin//учнинг координаталари
            p[i].x := x0 + Round(r*cos(a*2*pi/360));
            p[i].y := y0 - Round(r*sin(a*2*pi/360));
          end;
        a := a + 22.5;
      end;
    p[17].X := p[1].X;//Юлдуз контурини ёпиш учун
    p[17].Y := p[1].Y;
    Canvas.Polyline(p);//юлдузни чизиш
end;

//сичқонча тугмаларини чертилиши
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  if Button = mbLeft //чап тугма чертилганми ?
     then Form1.Canvas.Pen.Color := clBlack
     else Form1.Canvas.Pen.Color := clRed;
  StarLine(x, y, 30,Form1.Canvas);
end;
end.
```

Дастурни ишга тушириш

Эслатма: P массивнинг ўлчами учлар ва чуқурликлар сонидан биттага кўп ҳамда массивнинг биринчи ва охирги элементлари устма-уст тушади.



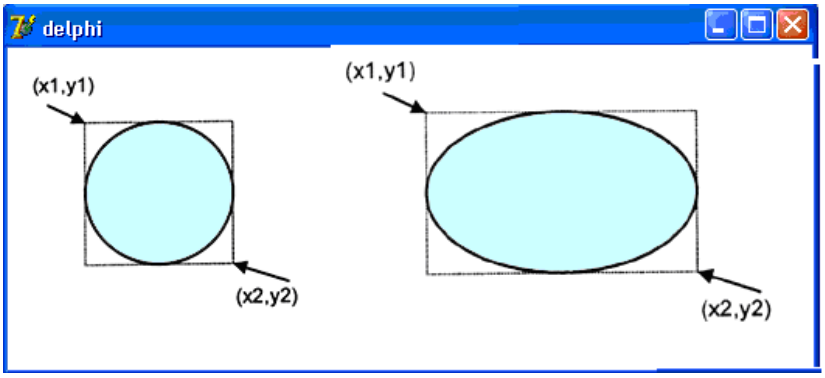
11.5-расм. Юлдуз

Айлана ва эллипс. Эллипс ёки айлана тасвирини **ҳосил** қилиш учун ***Ellipse*** методидан фойдаланилади. Эллипс ёки айлана чизиш бу методнинг параметрларига боғлиқ. Методни ишга туширишнинг умумий кўриниши қуйидагича:

Объект.Canvas.Ellipse(x1,y1, x2,y2]

Бу ерда ***объект*** — сиртида расм чизиш талаб қилинган объектнинг (компонентанинг) номи; ***x1, y1, x2, y2*** — айлана ёки эллипсга ташқи чизилган тўғри тўртбурчакнинг координаталари. Демак, агар тўғри тўртбурчак квадратдан иборат бўлса айлана, акс ҳолда эллипс чизилади. (11.6-расм).

Эллипс чизиқларининг ранги, қалинлиги ва стили Pen ҳусусиятининг қийматлари билан аниқланади, эллипснинг ички ранги ва фон бериш стили расм чизиладиган сиртнинг (canvas нинг) Brush ҳусусияти қийматлари билан белгиланади.



11.6-расм. Чизиладиган фигура Ellipse методининг параметрларига боғлиқ

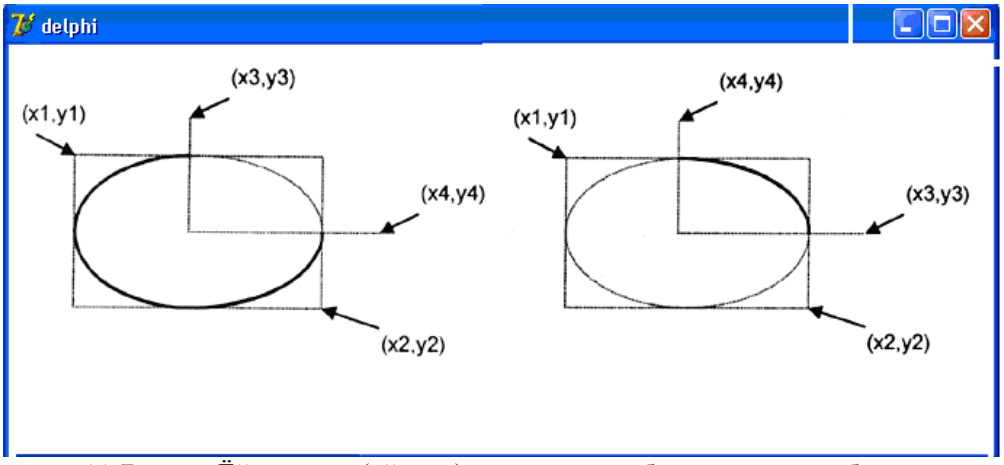
Ёй. Ёйларни ***Arc*** методи билан ясаш мумкин. Бу методни умумий кўринишда қуйидагича ёзиш мумкин:

Объект.Canvas.Arc(x1,y1,x2,y2,x3,y3,x4,y4)

Бу ерда ***x1, y1, x2, y2*** — чизилаётган ёйнинг асоси бўлган эллипснинг (айлананинг) параметрлари; ***x3, y3*** — ёйнинг бошланиш нуқтасини кўрсатувчи параметрлар; ***x4, y4*** —ёйнинг тугаш нуқтасини белгилайдиган параметрлар.

Бошланғич (охирги) нуқта – бу эллипс чегарасининг эллипс марказидан (x3, y3) ҳамда (x4, y4) координатали нуқталарга ўтказилган тўғри чизиқлар орасида ётган қисми. Ёйлар соат милларига тескари йўналишда ясалади. (11.7-расм).

Ёйнинг ранги, қалинлиги ва чизиқларининг стилини расм чизилаётган (canvas) сиртнинг Pen ҳусусияти қийматлари билан кўрсатилади.



11.7-расм. Ёй эллипс (айлана) нинг маълум бир қисмидан иборат.

Тўғри тўртбурчак. Тўғри тўртбурчаклар ***Rectangle*** методи билан чизилади. Бу методнинг умумий кўриниши қуйидагича:

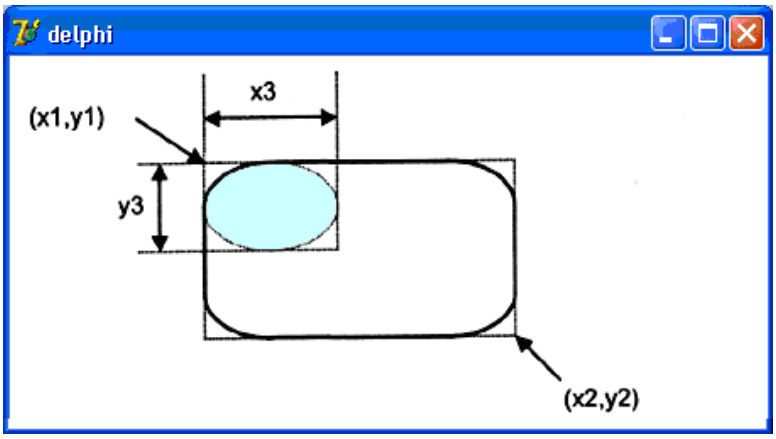
Объект.Canvas.Rectangle(x1, y1,x2, y2)

Бу ерда ***Объект*** — сиртида расм чизиладиган объект (компонент) нинг номи; ***x1, y1*** ва ***x2, y2*** — тўғри тўртбурчакнинг чап юқори ва ўнг қуйи бурчакларнинг координаталари.

RoundRec методи учлари ёйсимон бўлган тўғри тўртбурчак ясаш учун фойдаланилади. Бу методни умумий кўринишда

Объект.Canvas.RoundRec(x1,y1,x2, y2, x3, y3)

тарзида ишга туширилади. Бу ерда ***x1, y1, x2, y2*** – ёйсимон учли тўғри тўртбурчакка ташқи чизилган тўғри тўртбурчакнинг координаталари; ***x3***, ***y3*** — чорак қисми тўғри тўртбурчак учи сифатида чизиладиган эллипснинг ўлчамлари (11.8-расм).



11.8-расм. RoundRec методи учлари ёйсимон тўғри тўртбурчак чизади

Тўғри тўртбурчак чизиқларининг ранги, қалинлиги ва стили Pen ҳусусиятининг қийматлари билан аниқланади. Тўғри тўртбурчакнинг ички ранги ва фон бериш стили расм чизиладиган сиртнинг (canvas нинг) Brush ҳусусияти қийматлари билан белгиланади.

Тўғри тўртбурчак ясаш учун яна иккита метод мавжуд. Бу методлар қалам ўрнига мўйқалам билан чизади. ***FillRect*** методи бўялган тўғри тўртбурчак, ***FrameRect*** методи эса фақат тўғри тўртбурчак контурини чизиш учун фойдаланилади. Бу методларнинг ҳар иккисида фақат битта параметр - TRect типидаги структура қатнашади. TRect структурасининг майдонлари тўғри тўртбурчакли соҳанинг координаталарини сақлайди, улар Rect функцияси ёрдамида аниқланиши мумкин.

Кўпбурчак. ***Polygon*** методи кўпбурчак чизиш учун хизмат қилади. Бу методнинг параметри Tpoint типдаги массивдан иборат. Бу масивнинг ҳар бир элементи ёзув бўлиб, иккита майдон ***x*** ва ***y*** ларни ўз ичига олади. Бу ёзув-нуқта кўпбурчакнинг битта учини ифодалайди. Polygon методи координаталари массивда кўрсатилган нуқталарни кетма-кет кесмалар билан бирлаштиради: биринчисини иккинчи билан, иккинчисини учинчиси билан ва х.к. Сўнгра охирги нуқта ва биринчи нуқталар туташтирилади.

Кўпбурчак чизиқларининг ранги, қалинлиги ва стили Pen ҳусусиятининг қийматлари билан аниқланади, кўпбурчакнинг ички ранги ва фон бериш стили расм чизиладиган сиртнинг (canvas нинг) Brush ҳусусияти қийматлари билан белгиланади, соҳа сўйқаламнинг жорий ранги ва стилида бўялади. Қуйидаги процедурада polygon методи билан учбурчак ясалади.

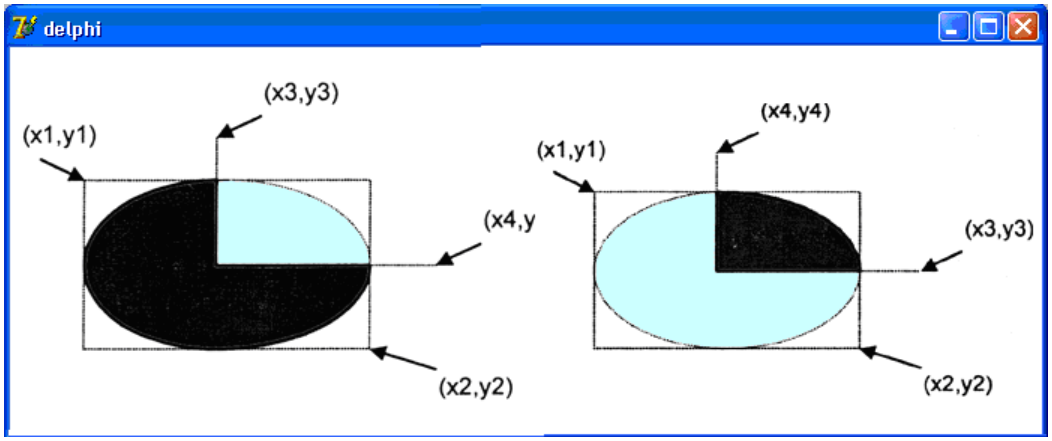
```
procedure TForm1.Button2Click(Sender: TObject);
var
pol: array[1..3] of TPoint;//Учбурчак учларининг координаталари
begin
pol[1].x := 10;  pol[1].y := 50;
pol[2].x := 40;  pol[2].y := 10;
pol[3].x := 70;  pol[3].y := 50;
Form1.Canvas.Polygon(pol);
end;
```

Сектор. Секторларни ***Pie*** методи ёрдамида чизиш мумкин. Бу методни умумий кўринишда қуйидаги бўйруқ билан чақирилади:

Объект. Canvas.Pie(x1,y1,x2,y2,x3,y3,x4,y4)

Бу ерда ***x1, y1, x2, y2*** — секторнинг асоси бўлган эллипсни (айланани) аниқлайдиган параметрлар; ***x3, y3, x4, y4*** - секторнинг чегараси бўлган тўғри чизиқларнинг охирги нуқталарининг координаталари.

Тўғри чизиқларнинг бошланғич нуқталари эллипс (айлана) маркази билан устма-уст тушади. Сектор соат милларига тескари йўналишда кесиб олинади: координаталари (x3, y3) бўлган нуқтадан бошланиб, (x4, y4) координатали нуқтада тугайди. (11.9-расм).



11.9-расм. Pie методининг параметрларига секторнинг боғлиқлиги

Нуқта. Дастур график чизадиган сиртга Canvas объекти мос келади. TColor типидаги икки ўлчовли массивдан иборат бўлган ***Pixels*** хусусияти график сиртнинг ҳар бир нуқтасининг ранги ҳақидаги маълумотни сақлайди. Pixels хусусиятидан фойдаланиб, график сиртнинг ихтиёрий нуқтасига талаб қилинган рангни бериш мумкин. Масалан,

```
Form1.Canvas.Pixels[10,10] := clRed;
```

буйруғи форма сиртидаги (10, 10) координатали нуқтага қизил ранг беради.

Pixels массивининг ўлчамлари график сиртнинг ўлчамлари билан белгиланади. Форманинг график сиртининг (клиентлар соҳаси деб ҳам аталади ишчи соҳанинг) ўлчамлари Clientwidth ва ClientHeight хусусиятларининг қийматлари билан аниқланади, Image компонентасининг ўлчамлари эса Width ва Height хусусиятларининг қийматлари билан белгиланади. Форманинг иши соҳасининг чап юқори бурчагига pixels[0,0], ўнг қуйи бурчакка эса -Pixels[Clientwidth-1,ClientHeight-1] элементлари мос келади.

Pixels хусусиятидан графиклар ясашда фойдаланилади. Одатда графиклар берилган формулалар бўйича ҳисоблашлар натижаси асосида қурилади. Функция аргументининг ўзгариш диапазони бошланғич маълумот деб қабул қилинади. Функция қийматларининг ўзгариш соҳаси ва диапазонини эса ҳисоблаб топиш мумкин. Олинган маълумотлар асосида масштабни топиш мумкин. Масштаб ясалаётган график расм чизиш учун мўлжалланган форма ёки сиртни тўла эгаллаши учун керак бўлади. Масалан, агар бирор *f(x)* функция 0 дан 1000 гача бўлган қийматларни қабул қилса, ҳамда унинг графигини чиқариш учун баландлиги 250 пиксел бўлган форма соҳаси олинган бўлса, у ҳолда Y ўқи бўйича масштабни $t = 250/1000$ формула билан топилади. Шундай қилиб, *f(x)* = 70 нуқтага координатаси Y = 233 бўлган нуқта мос келади. *Y* нинг координаталари

$$Y = h - f(x) \times t = 250 - 70 \times (250/1000)$$

формула билан топилган. Бу ерда h – график қуриладиган соҳанинг баландлиги. Эътибор берган бўлсангиз,

$250-70\times(250/1000)$ ифоданинг аниқ қиймати 232,5 га тенг. Аммо, нуқтани Canvas сиртига чиқариш учун фойдаланиладиган pixels хусусиятининг индекси фақат бутун сон бўлиши мумкин. Шунинг учун 232,5 сони ўзига энг яқин бутун сон, яъни 233 орқали яхлитланди.

Қуйида келтирилаётган дастур матни (11.4-листинг) pixels хусусиятидан фойдаланган ҳолда $y=2\sin xe^{x/5}$ функциясининг графигини чизади. График чизиш учун форманинг мумкин бўлган барча қисми банд қилинади. Агар фойдаланувчи дастурнинг иши давомида ойна ўлчамларини ўзгартирса, график янги масштаб бўйича бошқатдан чизилади.

11.5-листинг. Функциянинг графиги

```
unit grfunc_;
interface
uses   Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;
type
  TForm1 = class(TForm)
    procedure FormPaint(Sender: TObject);
    procedure FormResize(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
// графигини чизиш талаб қилинган функция
 Function f(x:real):real;
 begin
    f := 2*Sin(x)*exp(x/5);
 end;
// функция графигини чизиш
procedure GrOfFunc;
var
 x1,x2:real;   // функция аргументининг ўзгариш чегараси
 y1,y2:real;   // функция қийматининг ўзгариш чегараси
 x:real;       // функциянинг аргументи
 y:real;       // функциияниng х нуқтадаги қиймати
 dx:real;      // аргумент орттирмаси
 l,b:integer;  // график соҳанинг чап қуйи бурчаги
 w,h:integer;  // график соҳанинг кенглиги ва баландлиги
 mx,my:real;  // X ва Y ўқлари бўйича масштаб
 x0,y0:integer;// координаталар боши
begin
 // график соҳа
 l := 10;                // X - чап юқори бурчакнинг координатаси
 b := Form1.ClientHeight-20;// Y-чап юқори бурчак координатаси
 h := Form1.ClientHeight-40; // баландлиги
 w := Form1.Width-40;        // кенглиги
 x1 := 0;    // аргумент диапазонининг қуйи чегараси
 x2 := 25;   // аргумент диапазонининг юқори чегараси
 dx := 0.01; // аргумент қадами
 // функциянинг [x1, x2]кесмадаги энг катта ва энг кичик қийматларини топамиз
 y1 := f(x1);// минимум
 y2 := f(x1);// максимум
 x := x1;
 repeat
   y := f(x);
```

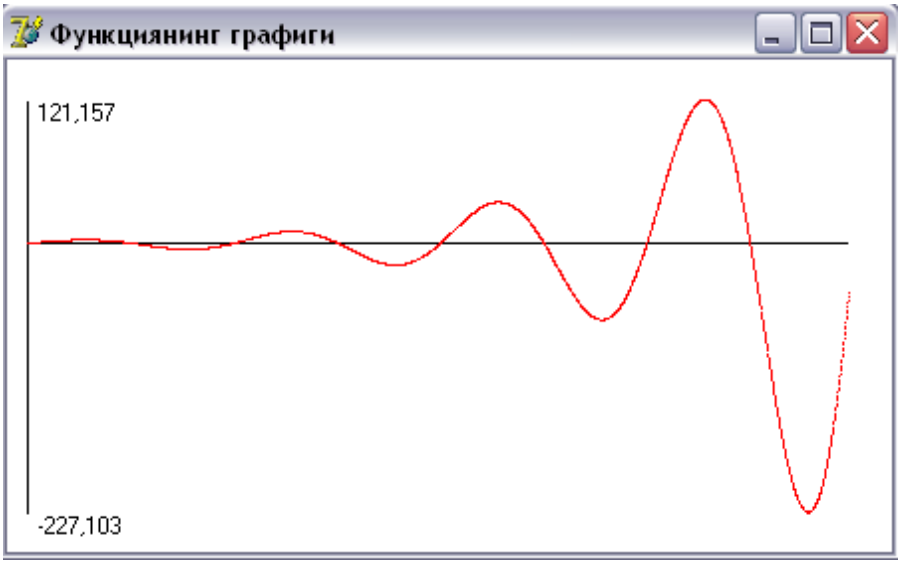
```
    if y < y1 then y1 := y;
    if y > y2 then y2 := y;
    x := x+dx;
  until (x>=x2);
 // масштабни ҳисоблаймиз
  my := h/abs(y2-y1); // Y ўқи бўйича масштаб
  mx := w/abs(x2-x1); // X ўқи бўйича масштаб
 // ўқлар
  x0 := l;
  y0 := b-Abs(Round(y1*my));
  with form1.Canvas do
  begin
    // ўқлар
    MoveTo(l,b);LineTo(l,b-h);
    MoveTo(x0,y0);LineTo(x0+w, y0);
    TextOut(l+5, b-h, FloatToStrF(y2, ffGeneral, 6, 3));
    TextOut(l+5, b, FloatToStrF(y1, ffGeneral, 6, 3));
    // графикни ясаш
    x := x1;
    repeat
      y := f(x);
      Pixels[x0+Round(x*mx),y0-Round(y*my)] := clRed;
      X := x + dx;
    until (x>=x2);
  end;
 end;
 procedure TForm1.FormPaint(Sender: TObject);
 begin
   GrOfFunc;
 end;
 //дастур ойнасининг ўлчами ўзгарди
 procedure TForm1.FormResize(Sender: TObject);
 begin
   // формани тозалаш
   form1.Canvas.FillRect(Rect(0,0,ClientWidth,ClientHeight));
   // графикни ясаш
   GrOfFunc;
 end;
 end.
```

Дастурни ишга тушириш

Ушбу дастурда асосий ишни GrOfFunc процедураси бажаради. У дастлаб функциянинг [x1,x2] ораликдаги максимал (y2) ва минимал (yl) қийматларини ҳисоблайди. Сўнгра форманинг кенглиги (Forml.Clientwidth -40) ва баландлиги (Form1.ClientHeight - 40) каби маълумотларни эътиборга олиб, X ўқи бўйича масштаб (mx) ва Y ўқи бўйича масштабларни (my) топади.

График чиқариладиган сиртнинг баландлиги ва кенглиги форманинг ишчи (клиент) соҳаси сарлавҳа ва чегараларни ҳисобга олмасдан аниқланади. Масштаб топилганидан сўнг, процедура горизонтал ўқнинг ***y*** координатасини (OY) аниқлайди ва граифкнинг координата ўқларини чизади. Сўнгра бевосита функциянинг графигини ясашга ўтилади. (11.10-расм).

GrOfFunc процедурасига мурожаат қилишни onPaint ва onFormResize ходисаларни қайта ишлаш процедуралари бажаради. TForm1.FormPaint процедураси дастур ишга туширилиб, экранда форма пайдо бўлган заҳоти графикни чизилишини таъминлайди. TForm1.FormResize процедураси форма ўлчамлари ўзгарганидан кейин график ясашни ўз зиммасига олган.



11.10-расм. GrOfFunc процедураси билан чизилган график

Келтирилган ушбу дастур универсал ҳисобланади. Дастур матнидаги ***f(x)*** функцияни алмаштириб, бошқа функциянинг графигини ҳам ясаш мумкиш. Функциянинг кўринишидан қатъий назар, унинг графиги формани тўла эгаллайди.

Эслатма: Сизларга таклиф қилинган дастур графиги чизилаётган функция ҳам манфий, ҳам мусбат қийматлар қабул қилганда тўғри ишлайди. Агар сизнинг функциянгиз фақат мусбат, ёки фақат манфий қийматларни қабул қилса, дастур матнига кичик ўзгартиришлар киритишингизга тўғри келади. Бу ўзгаришларни аниқлаш ўзингизга ҳавола.

менюга

11.5. Суратларни экранга чиқариш

Сурат деганда биз олдиндан тайёрланган, (масалан, PAINT ёки бошқа матн муҳаррири ёрдамида) алоҳида файл сифатида ЭҲМ хотира қурилмаларидан бирида сақланаётган тасвирий файлларни назарда тутамиз.

Суратларни ***Image*** компонентаси ёрдамида расм чизиладиган сиртга чиқариш мумкин. Унинг нишони **Additional** қуроллар панелида жойлашган. (11.11-расм). Айниқса кенгайтмаси ***bmp, jpg*** ёки ***ico*** бўлган суратлар экранга осонгина чақирилади.

11.8-жадвалда Image компонентаси асосий ҳусусиятларининг рўйхати келтирилган.

11.11-расм. Image компонентасининг нишони

Image компонентасининг ҳусусиятлари 11.8-жадвал

Ҳусусияти	**Мазмуни**
Picture	Компонента майдонида акс эттириладиган расм
Width, Height	Компонентанинг ўлчамлари. Агар бу ўлчам сурат ўлчамдан кам бўлиб, AutoSize ва Strech лар False бўлса, у

	ҳолда суратнинг маълум бир қисмигина кўрсатилади.
AutoSize	Компонентанинг ўлчамларини суратнинг ҳақиқий ўлчамларига боғлиқ равишда автоматик тарзда ўзгартирилиши
Strech	Суратни компонентанинг ўлчамларига боғлиқ равишда автоматик масштаблаштириш. Бунинг учун AutoSize нинг қиймати False бўлиши керак.
Vizible	Компонента ва сурат экрандаги форма сиртида кўринадиган бўлсинми?

Компонета майдонига чиқариладиган суратларни илова формасини яратиш жараёнида ҳам, дастур ишлаётган вақтда ҳам кўрсатиш мумкин.

Илова формасини тайёрлаётган вақтда суратлар ***picture*** ҳусусиятига стандарт диалог ойнасидан фойдаланган ҳолда файлни танлаш орқали кўрсатилади. ***Picture*** ҳусусияти **Picture Editor** (11.12-расм) ойнасининг **Load** тугмаси чертилганда экранда пайдо бўлади. ***Image Editor*** ни ишга тушириш учун **Object Inspector** ойнасида ***Picture*** ва унинг ёнидаги устундан учта нуқтали тугма чертилади.

Агар суратнинг ўлчамлари компонента ўлчамларидан катта бўлса, у ҳолда strech ҳусусиятига True қийматини бериш ҳамда width ва Height ҳусусиятларига суратнинг ҳақиқий ўлчамларига пропорционал қийматларни бериш лозим.

Суратларни Image майдонига дастурнинг иши жараёни да чиқариш учун Picture ҳусусиятига LoadFromFile методини қўллаш лозим. Бу методнинг параметри сифатида сурат файлининг номи кўрсатилади. Масалан:

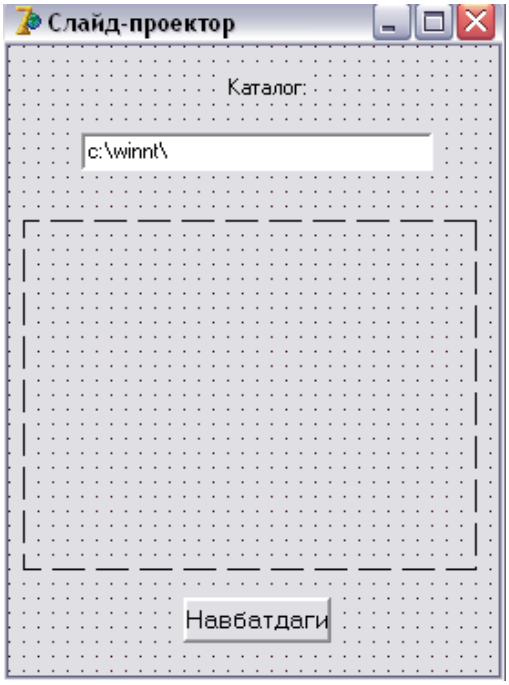
Form1.Image1.Picture.LoadFromFile('e:ŕtempŕbart.bmp')

кўрсатмаси суратни bart.bmp файлидан олади ва уни Image1 майдонига чиқаради.

LoadFromFile методи турли тасвирий форматдаги суратларни экранга чиқариш учун мўлжалланган: BMP, WMF, JPEG (jpg кенгайтмали файллар).

Қуйидаги дастур (унинг матни 11.5-листингда келтирилган) Image компонентасидан фойдаланувчи кўрсатган каталогдаги суратларни кўриш учун хизмат қилади. Бу дастурнинг диалог ойнаси 11.13-расмда кўрсатилган.

11.12-расм. Picture Editor ойнаси



11.13-расм. Слайд-проектор

11.5-листинг. Слайд-проектор

```
unit shpic_;
interface
```

```
uses    Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,    ExtCtrls, {jpeg,} StdCtrls,
Menus;
type
  TForm1 = class(TForm)
    Image1: TImage;
    Button1: TButton;
    Label1: TLabel;
    Edit1: TEdit;
    procedure FormActivate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
  aSearchRec : TSearchRec;
  aPath : String;//сурат жойлашган каталог
  aFile : String;//сурат файли
  n: integer = 0;
  iw,ih: integer;//Image компонетасининг дастлабки ўлчами
implementation
{$R *.DFM}

//сурат чиқариладиган соҳани сурат ўлчамига мослаб ўзгартириш
Procedure ScaleImage;
var
    pw, ph : integer;      //сурат ўлчами
    scaleX, scaleY : real;//X ва Y бўйича масштаб
    scale : real;           // масштаб
begin
     //сурат юкланди. Унинг ўлчамларини  аниқлаймиз
      pw := Form1.Image1.Picture.Width;
      ph := Form1.Image1.Picture.Height;
      if pw > iw//сурат кенглиги  компонента кенглигидан катта
         then scaleX := iw/pw //масштаблаштириш лозим
         else scaleX := 1;
      if ph > ih//сурат баландлиги компонента баландлигидан катта
         then scaleY := ih/ph //масштаблаштириш лозим
         else scaleY := 1;
     //энг кичик  коэффициент танлаймиз
      if  scaleX < scaleY
         then scale := scaleX
         else scale := scaleY;
     //сурат чиқариладиган соҳа ўлчамини ўзгартирамиз
  Form1.Image1.Height:=Round(Form1.Image1.Picture.Height*scale);
  Form1.Image1.Width:=Round(Form1.Image1.Picture.Width*scale);
 //Strech = True ҳамда соҳа ўлчами сурат ўлчамига пропорционал
 //бўлгани учун, сурат бузилишларсиз масштаблаштирилади
end;

//биринчи суратни  чиқариш
procedure FirstPicture;
var
   r : integer;//файлни қидириш натижаси
begin
   aPath := Form1.Edit1.Text;
```

```
   r := FindFirst(aPath + '*.bmp', faAnyFile, aSearchRec);
   if r = 0 then
     begin
       aFile := aPath + aSearchRec.Name;
       Form1.Image1.Picture.LoadFromFile(aFile);//суратни  юклаш
       ScaleImage;
       r := FindNext(aSearchRec);//навбатдаги файлни топиш
       if r = 0 then //яна суратлар борми?
           Form1.Button1.Enabled := True;
     end;
end;

//навбатдаги суратни чиқариш
Procedure NextPicture();
var
  r : integer;
begin
  aFile := aPath + aSearchRec.Name;
  Form1.Image1.Picture.LoadFromFile(aFile);
  ScaleImage;
  //навбатдаги суратни чиқаришга тайёрлаймиз
  r := FindNext(aSearchRec);//навбатдаги файли қидирамиз
  if r <> 0
    then //бошқа суратлар йўқ
      Form1.Button1.Enabled := False;
end;

procedure TForm1.FormActivate(Sender: TObject);
begin
  Image1.AutoSize := False;
  Image1.Stretch := True;    //масштаблашга рухсат берамиз
//суратларнинг дастлаб чиқариш соҳаси ўлчамларини эслаб қоламиз
  iw := Image1.Width;
  ih := image1.Height;
  Button1.Enabled := False;//Навбадаги тугмасига рухсат йўқ
  FirstPicture;//биринчи суратни чиқарилсин
end;
//Навбатдаги тугмаси чертилганда

procedure TForm1.Button1Click(Sender: TObject);
begin
   NextPicture;
end;
end.
```

Дастурни ишга тушириш

Дастур чиқарилаётган суратларни бузилишларсиз масштаб лаштиради. Буни тўғридан-тўғри Strech хусусиятига True қийматини бериб, эришиб бўлмайди. Биринчи ва қолган суратларни юклаш ва чиқаришни мос равишда FirstPicture ва NextPicture процедуралари бажаради. FirstPicture процедураси FindFirst функциясини биринчи учраган BMP-файлнинг номини топиш учун фойдаланади. FindFirst функциясининг параметри сифатида суратлар жойлашган каталог номи, asearchRec структураси (унинг Name майдони қидирилган маълумот топилган бўлса, талабни қаноатлантирувчи файл номини сақлайди) ҳамда сурат файли ниқобини кўрсатилади. Агар FindFirst функцияси чақирилганда кўрсатилган ҳеч бўлмаганда битта BMP-файл мавжуд бўлса, функциянинг қиймати нолга тенг. Бундай ҳолда LoadFromFile методи сурат файлини юклайди, ўнгра ScaleImage функцияси ишга туширилади. У компонента ўлчамини сурат ўлчамига мослаштиради. Юкланган суратнинг ўлчамини Form1.Image1.Picture.Width ва Form1.Image1.Picture.Height ҳусусияти қийматларига мурожаат қилиб, билиш мумкин. Уларнинг қийматлари Image компонентасининг ўлчамларига боғлиқ эмас.

11.6. Битли тасвирлар

Графика билан ишлаганда TBitMap (битли тасвир) типидаги объектлар иблан ишлаш жуда ҳам қулай. Битли тасвир компьютер хотирасида сақланаётган, демак кўринмайдиган сирт ҳисобланади. Унинг устида дастур турли тасвирларни ҳосил қилади. Битли тасвирларни (расмчаларни) осонлик ва тезлик билан форма ёки суратларни чиқариш соҳасига (image) чиқариш мумкин. Шунинг учун дастурда битли тасвирлардан унчалик катта бўлмаган расмлар, масалан, буйруқли тугмалар устидаги расмларни сақлаш учун қулай ҳисобланади.

Керакли тасвирларни битли тасвирга юклаш LoadFromFlie методи билан амалга оширилади, параметр сифатида керакли сурат сақланаётган BMP-файлининг номи кўрсатилади. Масалан, агар дастурда TBitMap типидаги pic ўзгарувчиси эълон қилинган бўлса,

pic.LoadFromFiie('e :/images/aplane.bmp')

буйруғи бажарилганидан сўнг, pic битли тасвири самолёт тасвирига эга бўлади.

Битли ттасвирларни форма ёки суратларни чиқариш соҳаси сиртига чақариш учун (canvas) сиртининг мос ҳусусиятига қўлланадиган Draw методидан фойдаланилади. Масалан.

Image1.Canvas.Draw(x,y, bm)

буйруғи Image1 компонентасининг сиртига bm битли тасвирини чиқаради (бунда x ва y параметрлар расмнинг чап юқори бурчагининг коммпонента сиртидаги ҳолатини белгилайди).

Агар Draw методини қўллашдан аввал TBitMap объектининг Transparent ҳусусиятига True қиймати берилса, у ҳолда расмнинг чап қуйи бурчагидаги ранг билан бир ҳил рангга эга бўлган расм бўлаклари экранда кўрсатилмайди, уларнинг ўрнида фон кўриниб туради. Агар "шаффоф" ранг сифатида расмнинг чап қуйи бурчагидаги рангдан бошқа ранг олинса, у ҳолда Transparentcolor ҳусусиятига керакли рангни кўрсатувчи белгили константани қиймат қилиб бериш лозим.

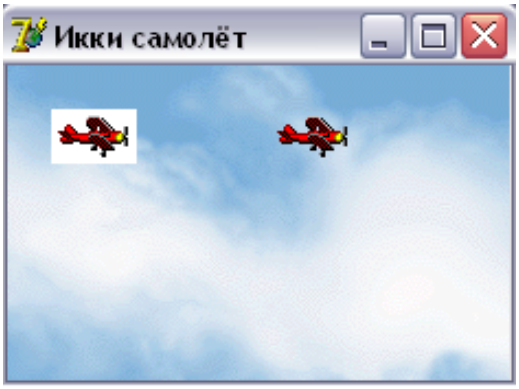
Қуйидаги (11.6-листингда берилган) дастур бир нечта қисмдан иборат битли тасвирларни ҳосил қилиш усулини намойиш қилади.

11.6-листинг. Битли тасвирлардан фойдаланиш

```
unit aplane0;
interface
uses  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,   Dialogs;
type
  TForm1 = class(TForm)
    procedure FormPaint(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
  sky,aplane: TBitMap;//битли тасвирлар: осмон ва самолёт
implementation
{$R *.dfm}
procedure TForm1.FormPaint(Sender: TObject);
begin
//битли тасвирни яратиш
sky := TBitMap.Create;
aplane := TBitMap.Create;
//расмларни юклаш
sky.LoadFromFile('sky.bmp');
aplane.LoadFromFile('aplane.bmp') ;
Form1.Canvas.Draw(0,0,sky);//фон
Form1.Canvas.Draw(20,20,aplane);//чап самолётни чизиш
aplane.Transparent := True;
```

```
//Энди ранги битли тасвирнинг чап қуйи бурчаги ранги билан
//бир хил бўлган расм парчалари кўринмайди
Form1.Canvas.Draw(120,20,aplane);//ўнг самолётни чизиш
//хотирани тозалаш
sky.free; aplane.free;
end;
end.
```

Дастур ишга тушганидан сўнг, илова ойнасида (11.14-расм) осмонда учаётган икки самолёт тасвири пайдо бўлади. Фон ва самолёт расмлари – битли тасвирлар ва улар файллардан юкланади. Чап самолёт атрофидаги оппоқ майдон aplane битли тасвирининг ҳақиқий ўлчамларини кўрсатади. Ўнг самолёт атрофида эса бундай майдон йўқ. Чунки, уни чиқаришдан аввал битли тасвирнинг Transparent ҳусусиятига True қиймати берилган.



11.14-расм. Transparent ҳусусиятини расмга таъсири

11.7. Мультипликация

Мультипликация деганда биз одатда харакатланиб ва ўзгариб турадиган расмларни тушунамиз. Энг содда ҳолатда расм харакат қилиши ёки ўзгариши мумкин.

Юқорида айтиб ўтилдики, расмлар содда график элементлардан иборат бўлади. Расмни сирт бўйлаб сурилишини таъминлаш жуда ҳам осон: дастлаб расмни экранга чиқариш, маълум бир муддат ўтганидан сўнг уни ўчириш, расмни навбатдаги позициядан яна экранга чиқариш ва х.к. Расмни экранга чиқариш ва ўчириш орасидаги вақтни ҳамда расмнинг "эски" ва "янги" позициясини шундай танлаш мумкинки, кўраётган одамда расм экран бўйлаб харакатланаётгандай тасаввур қолдиради. Қуйидаги дастур (11.7-листинг) ва форманинг кўриниши (11.15-расм) илова ойнасининг чап томонидан ўнг томонига харакат қилишини намойиш қилади.

11.15-расм. Харкатланаётган айлана

11.8-листинг. Ҳаракатланаётган айлана

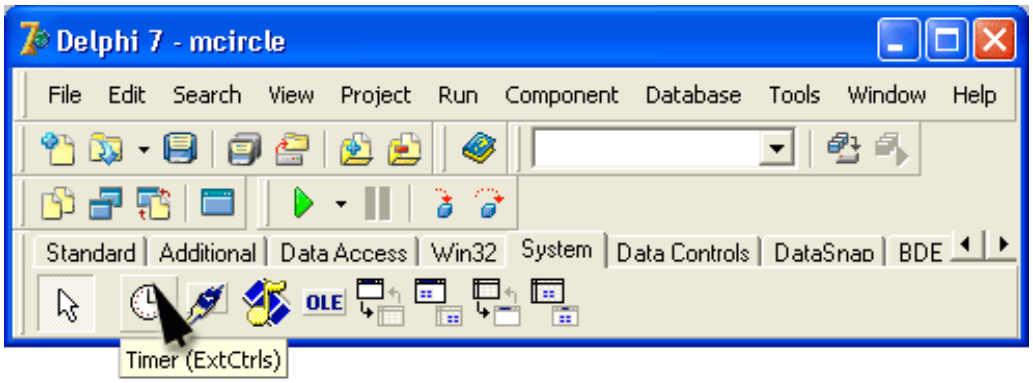
```
unit Unit1;
interface
uses   Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,   Dialogs, ExtCtrls;
type
  TForm1 = class(TForm)
    Timer1: TTimer;
    procedure Timer1Timer(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var
  Form1: TForm1;
implementation
{$R *.dfm}
var
x,y: byte;//айлана марказининг координаталари
dx: byte;//x координатасининг сурилиши
//ўчириш ва янги жойда айлана чизиш
procedure Ris;
begin
//айланани ўчириш
form1.Canvas.Pen.Color:=form1.Color;
form1.Canvas.Ellipse(x, y, x + 30, y + 30);
x := x + dx;
//айланани янги жойдан чизиш
form1.Canvas.Pen.Color := clBlack;
form1.Canvas.Ellipse(x, y, x + 30, y + 30) ;
end;
procedure TForm1.Timer1Timer(Sender: TObject);
begin
   Ris;
end;
procedure TForm1.FormActivate(Sender: TObject);
begin
x := 0; y := 10; dx := 5;
timer1.Interval := 50;
//  OnTimer ходисасининг рўй бериши -0.5 сек
form1.canvas.brush.color := form1.color;
end;
end.
```

Дастурни ишга тушириш

Ушбу дастурда асосий вазифани Ris процедураси бажаради. У айланани ўчиради ва янги позициядан айлана чизади. Айланани ўчириш масаласи шу айланани фон ранги билан бир ҳил рангда устидан чизиш ҳисобига ҳал қилинади.

Илова формасига Ris процедурасини даврий равишда чақириш учун формага кўринмайдиган Timer (таймер) компонентаси киритилган. Унинг нишони **System** куроллар панелида жойлашган. (11.16-расм). Timer компонетасининг ҳусусиятлари 11.9-жадвалда келтирилган.



11. 16-расм. Timer компонентасининг нишони

Timer компонентасини формада одатдаги усуллар билан жойлаштириш мумкин, аммо, бу компонента новизуал, яъни кўринмайдиган бўлгани учун, дастурнинг иши давомида экранга чиқарилмайди. Шунинг учун уни форманинг ихтиёрий жойига ўрнатиш мумкин.

Timer компонентасининг ҳусусиятлари 11.9-жадвал

Ҳусусияти	Мазмуни
Name	Компонентанинг номи. Компонентага мурожаат қилиш учун фойдаланилади.
Interval	OnTimer ходисасининг генерация қилиш даври. Миллисекундларда берилади
Enabled	Ишлашга рухсат. OnTimer ходисасини генерациясига рухсат бор (қиймати True) ёки йўқ (қиймати False)

Timer компонентаси OnTimer ходисасини қайд қилади. Бу ходисанинг рўй бериши миллисекундларда ўлчанади ва Interval ҳусусиятининг қиймати билан аниқланади. Enabled ҳусусиятига эътибор беринг. У дастурга таймерни "ишга тушириш" ёки "тўхтатиб қўйиш" га рухсат беради. Агар Enabled ҳусусияти False га тенг бўлса, OnTimer ходисаси рўй бермайди .

OnTimer ҳодисаси юқоридаги дастурда Timer1Timer процедураси билан қайта ишланади. Бу процедура ўз навбатида Ris процедурасини ишга туширади. Бу дастурда Ris процедурасини даврий равишда, вақти-вақти билан ишга тушириш механизми қўлланган.

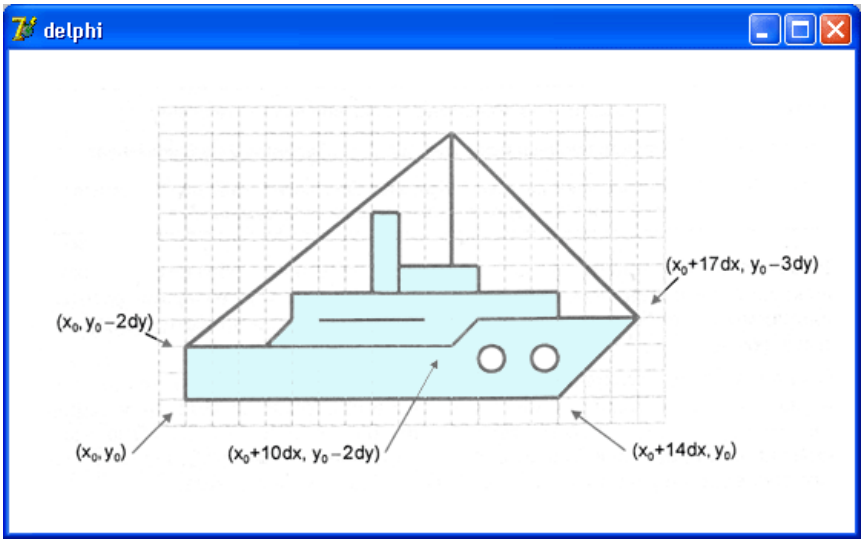
Эслатма: ***x*** ва ***y*** ўзгарувчилар (айлана марказининг координаталари) ва dx (айлананинг харакатланишида x координатани сурилиши) Ris процедурасидан ташқарида эълон қилинган, яни улар глобал ўзгарувчилар ҳисобланади. Шунинг учун уларнинг инициализация қилишни унутмаслик керак. (дастурда глобал ўзгарувчиларнинг инициализациясини FormActivate процедурасида амалга оширилади).

менюга

11.8. Базавий нуқта методи

Мураккаб, яъни бир нечта элементлардан иборат тасвирлар билан ишлаганда базавий нуқта методи деб аталадиган метод фойдаланилади. Бу методнинг ғояси қуйидагича:

1. Тасвирининг бирор нуқтасини базавий нуқта деб танлаб олинади.



11.17-расм. Базавий нуқтага нисбатан тасвир координаталарини аниқлаш

2. Қолган нуқталарнинг координаталари базавий нуқтага нисбатан қайта ҳисобланади.

3. Агар тасвирдаги нуқталарнинг координаталарини бирликларга нисбатан (пикселларда эмас) қайта ҳисобланса, у ҳолда тасвирларни масштаблаштириш (мослаштириш) имконияти юзага келади.

11.7-расмда кемача тасвири берилган. Базавий деб (X_0, Y_0) координатали нуқтани оламиз. Қолган нуқталар координаталарини шу нуқтага нисбатан қайта ҳисоблаймиз 11.8-листингда "сузаётган" кемача тасвири ҳосил қилинган дастур матни берилган.

11.8-листинг. Кемача

```
unit ship_;
interface
uses   Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,   StdCtrls, ExtCtrls;
type
```

```
  TForm1 = class(TForm)
    Timer1: TTimer;
    procedure Timer1Timer(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
  x,y: integer;//кема координаталари (базавий нуқта)
implementation
{$R *.DFM}

procedure Titanik(x,y: integer;  //базавий нуқта координаталари
            color: TColor);   //кема ранги
const
  dx = 5;  dy = 5;
var
  buf: TColor;
begin
  with form1.canvas do
  begin
    buf := pen.Color;  //жорий рангни сақлаймиз
    pen.Color := color;//керакли рангли ўрнатамиз
    //чизамиз ...    //  кема корпусини
    MoveTo(x, y);
    LineTo(x, y-2*dy);
    LineTo(x + 10*dx, y-2*dy);
    LineTo(x + 11*dx, y-3*dy);
    LineTo(x + 17*dx, y-3*dy);
    LineTo(x + 14*dx, y);
    LineTo(x, y);
    //кема устидагилар
    MoveTo(x + 3*dx, y-2*dy);
    LineTo(x + 4*dx, y-3*dy);
    LineTo(x + 4*dx, y-4*dy);
    LineTo(x + 13*dx, y-4*dy);
    LineTo(x + 13*dx, y-3*dy);
    MoveTo(x + 5*dx, y-3*dy);
    LineTo(x + 9*dx, y-3*dy);
    //капитан кўприкчаси
    Rectangle(x + 8*dx, y-4*dy, x + 11*dx, y-5*dy);
    //труба
    Rectangle(x + 7*dx, y-4*dy, x + 8*dx, y-7*dy);
    //иллюминаторлар
    Ellipse(x + 11*dx, y-2*dy, x + 12*dx, y-1*dy);
    Ellipse(x + 13*dx, y-2*dy, x + 14*dx, y-1*dy);
    //мачта
    MoveTo(x + 10*dx, y-5*dy);
    LineTo(x + 10*dx,y-10*dy);
    //оснастка
    MoveTo(x + 17*dx, y-3*dy);
    LineTo(x + 10*dx, y-10*dy);
    LineTo(x, y-2*dy);
    pen.Color := buf;//қаламнинг эски рангини тиклаймиз
  end;
```

```
end;

//таймер сигналини қайта ишлаш
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  Titanik(x, y, form1.color);//расмни ўчириш
  if x < Form1.ClientWidth
    then x := x + 5
    else begin//янги рейс
      x := 0;
      y := Random(50) + 100;
    end;
  Titanik(x,y,clWhite);    //янги нуқтадан чизиш
end;

procedure TForm1.FormActivate(Sender: TObject);
begin
    x := 0;      y := 100;
    Form1.Color := clNavy;
    Timer1.Interval := 50;//таймер сигнали хар 50 мСек да
end;
end.
```

Дастурни ишга тушириш

Кемача тасвирини чизиш ва ўчириш вазифаларин Titanik процедураси бажаради. У параметрлар сифатида базавий нуқта координаталари ва кемача тасвирини чизиш учун рангни олади. Агар процедура чақирилганда ранг форма фони рангидан фарқ қилса, у ҳолда кемача чизилади, акс ҳолда ўчирилади. Titanik процедурасида dx ва dy константалар эълон қилинган бўлиб, улар тасвир нуқталари координаталарини ҳисоблашдаги қадамни (пикселларда) билдиради. Бу константаларнинг қийматларини ўзгартириб, тасвирни масштаблаштиришни амалга ошириш мумкин.

менюга

11.9. Битли тасвирлардан фойдаланиш

Аввалги пунктда бир нечта содда график элементлардан ташкил топган тасвирни кўрган эдик. Энди, битта мураккаб тасвирни бошқа мураккаб тасвир фонида харакатланишини кўрайлик. Масалан: шаҳар устидан ўтаётган самолёт тасвирини ясашга уриниб кўрамиз.

Расмнинг харакатланиш эффектини аввалги позициясига нисбатан расмни даврий равишда расмнинг қайта чизилиши (бир оз сурилиши) тарзида ифодалаймиз. Бунда, расмни ҳам гал қайта чизишдан аввал, унинг олдинги ҳолатини ўчириш лозим бўлади. Расмни ўчириш эса фон-расм ёки унинг харакатланаётган расм билан тўсилган жойини қайта чизиш орқали амалга оширилади.

Кўрилаётган дастурда иккинчи усулни қўллаймиз. Расмлар ёрдамида Imagt компонентасиниг Canvas ҳусусиятига Draw методини қўллаб, форма сиртига чиқарилади. Ўчириш эса нусха олиш усули (copyRect методи) билан буфердан керакли фонни Image компонетаси сиртига чиқариш билан ҳал қилинади. Бу дастурнинг формаси 11.18-расмда, матни эса 11.9-листингда келтирилган.

Image компонентаси фонни ҳосил қилиш учун фойдаланилади, Timer компонентаси эса самолёт тасвирини ўчириш ва янги жойдан экранга чиқариш жараёнини даврий равишда бўлишини таъминлайди

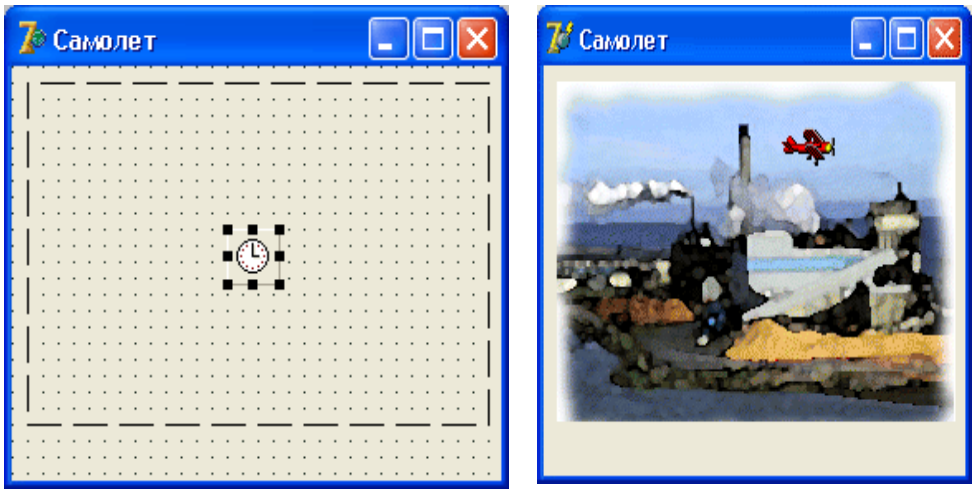
11.9-листинг. Учаётган самолёт

```
unit aplane_;
interface
uses  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,  ExtCtrls, StdCtrls, Buttons;
type
  TForm1 = class(TForm)
```

```
    Timer1: TTimer;
    Image1: TImage;
    procedure FormActivate(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
var
  Back, bitmap, Buf : TBitMap; //фон, картинка, буфер
  BackRct : TRect;//буфердан тикланадиган фон соҳаси
  BufRct: Trect;  //Фонни тиклаш учун керак бўлган буфер соҳаси
   x,y:integer;   //расмнинг жорий ҳолати
  W,H: integer;   //расмнинг ўлчамлари
procedure TForm1.FormActivate(Sender: TObject);
begin
 //учта объект -битли тасвир яратиш -
  Back := TBitmap.Create;   //фон
  bitmap := TBitmap.Create; //расм
  Buf := TBitmap.Create;    //буфер
 //фонни юклаш ва чиқариш
  Back.LoadFromFile('factory.bmp');
  Form1.Image1.canvas.Draw(0,0,Back);
 //харакатланадиган расмни юклаш
  bitmap.LoadFromFile('aplane.bmp');
 //"шаффоф" рангни танлаш
  bitmap.Transparent := True;
  bitmap.TransParentColor := bitmap.canvas.pixels[1,1];
 //расм қўйиладиган фон соҳаси нусхасини сақлаш учун буфер яратиш
  W := bitmap.Width;
  H := bitmap.Height;
  Buf.Width := W;
  Buf.Height := H;
  Buf.Palette := Back.Palette;//палитралар мослигини таъминлаш учун !!
  Buf.Canvas.CopyMode := cmSrcCopy;
 //фонни тиклаш учун зарур бўлган буфер соҳасини аниқлаймиз
  BufRct := Bounds(0, 0, W, H);
 //расмнинг бошланғич ҳолати
  x := -W;
  y := 20;
 //фоннинг сақланадиган қисмини  аниқлаймиз
  BackRct := Bounds(x, y, W, H);
 //ва уни сақлаймиз
  Buf.Canvas.CopyRect(BufRct, Back.Canvas, BackRct);
end;
//таймер сигналини қайта ишлаш
procedure TForm1.Timer1Timer(Sender: TObject);
begin
    //буфердан фонни тиклаб, расмни ўчирамиз
     Form1.image1.canvas.Draw(x,y,Buf);
     x := x + 2;
```

```
    if x>form1.Image1.Width then x := -W;
   //сақланадиган фон соҳасини аниқлаймиз
    BackRct := Bounds(x, y, W, H);
   //унинг нусхасини сақлаймиз
    Buf.Canvas.CopyRect(BufRct, Back.Canvas, BackRct);
   //расмни чиқарамиз
   Form1.image1.canvas.Draw(x,y,bitmap);
end;
//дастур ишини  якунлаймиз
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
//битли тасвир учун ажратиллган хотирани бўшатамиз
 Back.Free;
 bitmap.Free;
 Buf.Free;
end;
end.
```

Дастурни ишга тушириш



11.19-расм. Самолёт дастурининг фони ва иш вақтидаги кўриниши

Фон ва самолётнинг битли тасвирларини, шунингдек самолёт тўсиб қолаётган фон соҳасини сақлаш учун TbitMap типидаги объектлардан фойдаланнлади. Бу объектлар FormActivate процедураси ёрдамида динамик равишда ҳосил қилинади. Шу процедуранинг ўзи фон расми бўлган factory.bmp ҳамда самолётнинг расми aplane.bmp файлларини юклайди, ҳамда самолёт биринчи марта қўйиладиган фон соҳасини сақлаб қолади.

Фоннинг нусхасини сақлаш CopyRect методи ёрдамида амалга оширилади. У битта тўғри тўртбурчак битли тасвир нусхасини бошқасига кўчиришга имкон беради. CopyRect методи қўлланаётган объект битли тасвирни қабул қилувчи бўлади. Методнинг параметрлари нусха кўчириладиган соҳанинг координаталари ва ўлчами, нухса олинадиган сирт, нусҳа олинаётган соҳанинг ҳолати ва ўлчамларидан иборат бўлади. Буферга кўчирилаётган самолёт расми қўйиладиган фон соҳасининг ҳолати ва ўлчами ҳамда буфердан қайта тикланадиган фон соҳаси ҳақидаги маълумотлар Trect типидаги BackRct структурасида сақланади. Бу структурани тўлдириш учун Bounds функциясидан фойдаланилади.

Эътибор беринг, харакатланадиган битли тасвирнинг чап юқори бурчагининг x – координатаси манфий сон ва у битли тасвир кенглигига тенг. Шунинг учун дастур ишининг бошланишида самолёт тасвири кўринмайди. Расмни чизиш кўринадиган соҳадан ташқарида бошланади. Ҳар бир OnTimer ходисаси рўй берганда ***x***-нинг координатаси ўсиб боради ва экранда битли тасвирининг координаталари нолдан катта бўлган қисми пайдо бўлади. Шундай қилиб, фойдаланувчида самолёт ҳудди ойнанинг чап чегарасидан учиб чиқаётгандай тасаввур ҳосил бўлади.

менюга

11.10. Дастур ресурсидан битли тасвирларни юклаш

11.9-листингда келтирилган фон ва самолёт расмлари файллардан юкланади. Бу ҳар доим ҳам қулай бўлавермайди. Delphi тили барча зарурий битли тасвирларни ресурслар кўринишида бажарилувчи дастурнинг файлига қўшиб қўйишга имкон беради. Бу тасвирлардан кейинчалик эхтиёжга қараб ресурсдан, яъни бажарилувчи (EXE-файл) файлдан юклаб мумкин бўлади.

Ресурслар файлини яратиш. Битли тасвирларни ресурсдан юклаш мумкин бўлиши учун аввал, барча битли тасвирларни ўз ичига олган ресурслар файлини яратиш лозим бўлади.

Ресурслар файлини **Image Editor** утилити (тасвирлар муҳаррири) ёрдамида ташкил қилиш мумкин. Бу утилит **Tools** менюсидаги **Image Editor** буйруғини танлаш орқали ишга туширилиши мумкин.

Янги ресурслар файлини яратиш учун **File** менюсидан **New** буйруғини танлаш керак, сўнгра экранда пайдо бўладиган остменюдан - **Resource File** (ресурслар файли) тугмасини чертилади. Натижада янги ресурслар файлининг ойнаси пайдо бўлади, **Image Editor** ойнасининг менюлар сатрида янги пункт - **Resource** пайдо бўлади.

Бу файлга янги ресурсни қўшиш учун **Resourse** менюсидан **New** буйруғини, сўнгра очиладиган рўйхатдан ресурс типи танланади. Бизнинг ҳолда **Bitmap** (битли тасвир) танланади. Бу танловдан кейин **Bitmap Properties** (битли тасвирнинг ҳусусияти) диалог ойнаси экранга узатилади. Ундан фойдаланиб яратиладиган расмларнинг ўлчами ва ранглар сони белгиланади. ОК тугмасини босиш билан **Bitmap Properties** диалог ойнасига тармоқланган рўйхатдан **Bitmap1** элементи чақирилади. Бу элемент файлга қўшилган янги ресурсга мос келади.

Bitmap1 — бу автоматик тарзда яратилган ресурс номи бўлиб, **Resourse** менюсидаги **Rename** буйруғини танлаш ва сўнгра керакли номли ёзиш билан ўзгартирилиши мумкин. **Bitmap1** номини ўзгартирилганидан кейин битли тасвирни яратишга киришиш мумкин. Бунинг учун **Resourse** менюсидан **Edit** буйруғини танлаш лозим. Натижада график муҳаррирнинг таҳрирлаш ойнаси очилади.

Image Editor график редактори дастурчига шундай редакторларга ҳос бўлган қуроллар мажмуасини таклиф қилади. Бу қуроллардан фойдаланиб дастурчи ўзи учун зарур бўлган тасвирларни ясаши мумкин. Агар иш вақтида расм масштабини ўзгартиришга эхтиёж пайдо бўлса, масштабни катталаштириш учун **View** менюсидан **Zoom In** меню буйруғи**, кичиклаштириш учун эса Zoom Out** буйруғи танланади. Тасвирни ҳақиқий масштабда кўриш учун эса **View** менюсидан **Actual Size** буйруғини танлаш лозим.

Агар керакли расм олдиндан алоҳида файл шаклида мавжуд бўлса, уни алмашув буфери (clipboard) ёрдамида ресурслар файлининг битли тасвирига жойлаш мумкин. Бу қуйидагича бажарилади:

1. Дастлаб график редактор, масалан Microsoft Paint ишга туширилади, унга тасвир файлини юкланади ва бу тасвирни тўла ёки маълум бир қисми ажратилади. Ажратиш жараёнида диққатни ажратилган соҳанинг пикселлардаги ўлчами ҳақидаги маълумотга қаратиш лозим. (Paint ажратилаётган соҳанинг ўлчамларини ҳолатлар сатрига чиқаради). Сўнгра **Правка** менюсидан Копировать тугмаси танланади. Натижада тасвирнинг ажратилган қисми буфер хотирага тушади.

2. Сўнгра **Image Editor** га ўтиб, буфердаги тасвир қўшиладиган ресурс танланади ва ресурс характеристикаларини буфердаги тасвир характеристикаси қийматларига мосланади. Ресурс характеристикаси қийматлари **Bitmap Properties** диалог ойнасининг махсус ойна-майдонларига киритилади. Бу диалог ойнасини **BitMap** менюсининг **Image Properties** буйруғи билан экранга чақириш мумкин. Барча характеристикалар ўрнатилганидан сўнг **Edit** менюсидан Paste буйруғи билан буфер хотирадаги тасвирни ресурсга қўшиш мумкин.

3. Ресурслар файлига барча керакли ресурсларни қўшилганидан сўнг, ресурс файлини шу ресурслар учун мўлжалланган бажарилувчи дастур жойлашган каталогда сақлаб қўйиш лозим. Файл одатдаги усул билан сақлаб қўйилади. Image Editor муҳаррири бу файлга **res** кенгайтмасини беради.

Ресурслар файлини ишга тушриш. Ресурслардан дастурда фойдаланиш мумкин бўлиши учун, дастур матнида компиляторга бажарилувчи файлнинг таркибига ресурслар файлини ҳам қўшиб қўйиш ҳақида кўрсатма берилиши лозим. Бу кўрсатма умумий кўринишда қуйидагича ёзилади:

{**$R Ресурслар файли**}

Бу ерда **Ресурслар файли** — ресурслар файлининг номи. Масалан,

{$R images.res}

Ресурслар файлини бажариладиган файл таркибига қўшиш кўрсатмаси одатда модул матнининг

бошланишида ёзилади.

Эслатма: Агар модул файлининг ва ресурслар файлининг номлари бир ҳил бўлса, ресурс файлининг номи ўрнига "*" белгисини қўшиш мумкин. Бу ҳолда юқоридаги кўрсатма қуйидагича ёзилади:

{$R *.res}

Ресурсдан тасвирларни TBitMap типидаги ўзгарувчига юклаш масаласи LoadFromResourceName методи ёрдамида ҳал қилинади. Унинг иккита параметри бор: дастурнинг идентификатори ва ресурс номи. Дастур идентификатори сифатида Hinstance глобал ўзгарувчиси қўлланади. Ресурс номи сатрли константа тарзида ёзилади. Масалан, тасвирни ресурсдан Pic ўзгарувчисига юклаш учун

Pic.LoadFromResourceName(Hinstance,'FACTORY') ;

кўринишидаги буйруқдан фойдаланилади.

Намуна қилиб, матни 11.10-листингда келтирилган дастурни олиш мумкин. Бу дастурда фон ва самолёт татсвирлари ресурсдан юкланади.

11.10-листинг. Тасвирларни ресурсдан юклаш учун намуна.

```
unit aplane1_;
{$R images.res} //ресурслар файлини қўшиш
interface
uses   Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,   ExtCtrls, StdCtrls, Buttons;
type
  TForm1 = class(TForm)
    Timer1: TTimer;
    Image1: TImage;
    procedure FormActivate(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
  Back, bitmap, Buf : TBitMap;//фон, расм, буфер
  BackRct,  BufRct: TRect;    //фон, расм, буфер соҳалари
  x,y:integer;//расмнинг чап юқори  бурчаги координаталари
  W,H: integer;//расмнинг ўлчами
implementation
{$R *.DFM}

procedure TForm1.FormActivate(Sender: TObject);
begin
  Back := TBitmap.Create;  //фон
  bitmap := TBitmap.Create;//расм
  Buf := TBitmap.Create;    //буфер
 //ресурсдан фонни юклаш
  Back.LoadFromResourceName(HInstance,'FACTORY');
  Form1.Image1.canvas.Draw(0,0,Back);
 //ресурсдан харакатланиши талаб қилинган расмни юклаш
  bitmap.LoadFromResourceName(HInstance,'APLANE');
  bitmap.Transparent := True;
  bitmap.TransParentColor := bitmap.canvas.pixels[1,1];
 //устига расм тушадиган фон соҳасини сақлаш учун буфер яратиш
  W := bitmap.Width;
  H := bitmap.Height;
  Buf.Width := W;
```

```
  Buf.Height := H;
 Buf.Palette:=Back.Palette;//палитралар мослигини таъминлаш учун
  Buf.Canvas.CopyMode := cmSrcCopy;
  BufRct := Bounds(0, 0, W, H);
  x := -W;
  y := 20;
 //фоннинг сақланадиган соҳасини аниқлаш
  BackRct := Bounds(x,y,W,H);
 //ва уни сақлааймиз
  Buf.Canvas.CopyRect(BufRct,Back.Canvas,BackRct);
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
   //фонни қайти таклиб, расмни ўчираиз
    Form1.image1.canvas.Draw(x,y,Buf);
    x := x + 2;
    if x>form1.Image1.Width then x := -W;
   //Фоннинг сақланадиган соҳасини аниқлаймиз
    BackRct := Bounds(x, y, W, H);
   //унинг нусхасини сақлаймиз
    Buf.Canvas.CopyRect(BufRct,Back.Canvas,BackRct);
   //расмни чиқарамиз
   Form1.image1.canvas.Draw(x,y,bitmap);
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
 Back.Free;
 bitmap.Free;
 Buf.Free;
end;
end.
```

Тасвирларни ресурсдан юклашнинг афзаллиги катта: дастурларни бошқа компьютерларга ўтказишда ҳамма керакли файлларнинг мавжуд бўлиши ҳақида қайғурилмайди, чунки, дастур учун зарур бўлган барча файллар бажариладиган файл ичида сақланади.

менюга

11.11. "Мультфильм" кўриш

Ушбу пунктда қандай қилиб диалог ойнасида мультфильм намойиш қилиш мумкинлигини кўрамиз.

Телефон ва компьютер ўртасида югураётган қизил квадратча эффектини ҳосил қилиш масаласи диалог ойнасида алмашувчи расмларни навбатма-навбат чиқариш ҳисобига ҳал қилинади.

Мультфильм кадрлари одатда битта файлда ёки битта ресурсда жойлашган бўлади. Дастур ўз ишини бошлаганидан кейин улар буферга юкланади. Шунинг бу расмлар BitMap типидаги тасвирлар бўлса, мақсадга мувофиқ бўлади. Мультфильмни намойиш қилувчи (экранга чиқарувчи) процедуранинг вазифаси навбатдаги кадрни ажратиш ва уни форманинг керакли жойига қўйишдан иборат бўлади.

Кадрларни форма сиртига чиқариш шу форманинг Canvas ҳусусиятига copyRect методини қўллаш ҳисобига амалга оширилади. CopyRect методи берилган график сиртнинг тўғри тўртбурчак шаклидаги соҳасини бошқа сиртга кўчиради.

CopyRect методи умумий кўринишда қуйидагича ёзилади:
Canvas1.CopyRect(Соҳа1, Canvas2, Соҳа2)

Бу ерда ***canvas1*** — нусха кўчириладиган график сирт; ***Canvas2*** — нусхаси кўчириладиган график сирт; ***Соҳа2*** парметри – кўчириладигшан тўғри тўртбурчакл соҳанинг ҳолати ва ўлчамлари; ***Соҳа1*** параметри эса нусханинг Canvas1 сиртидаги ҳолати.

Соҳа1 ва Соҳа2 параметрлари сифатида Trect типидаги структурадан фойдаланилади. Уларнинг майдонлари соҳанинг ҳолати ва ўлчамларини белгилайди. TRect структураси майдонларини Bounds функцияси ёрдамида тўлдирилиши мумкин. Бу функцияга мурожаат қилиш буйруғи қуйидагича:

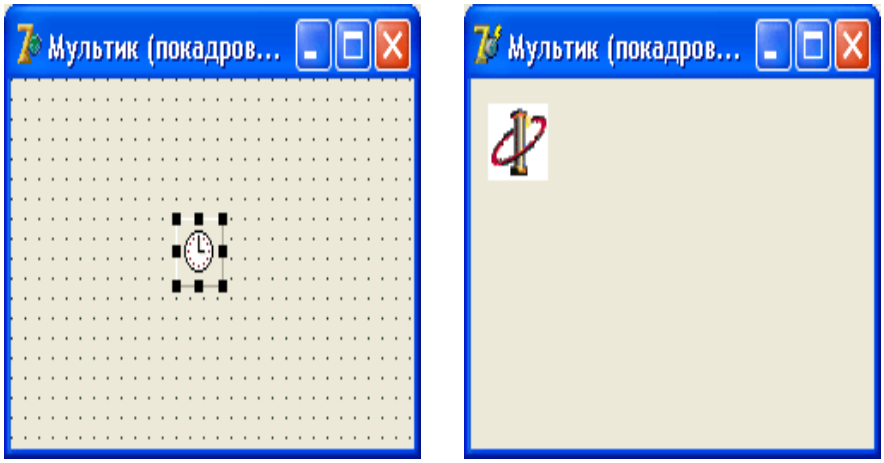
Bounds(x,y,Width,Height)

Бу ерда ***x*** ва ***y*** —соҳанинг чап ва ўнг юқори координаталари; width ва Height — соҳанинг кенглиги ва баландлиги.

Матни 11.11-листингда берилган қуйидаги дастур диалог ойнасига Delphi устунини чиқаради ва унинг атрофида қандайдир объектнинг учиб айланаётганлигини тасвирлаайди. 11.19-расмда шу мультфильмни кадрлари келтирилган. (film.bmp файлидаги тасвир).

Дастурнинг диалог ойнаси 11.20-расмда кўрсатилган. У ягона омпонента-таймерни ўз ичига олган.

11.18-расм. Мультфильм кадрлари



11.20-расм. Дастурнинг формаси

11.11-листинг. Мультфилм (CopRect методидан фойдаланиш)

```
unit multik_;
interface
uses   Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,   ExtCtrls, StdCtrls;
type
  TForm1 = class(TForm)
    Timer1: TTimer;
    procedure FormActivate(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
const
  FILMFILE = 'film2.bmp';//фильм - bmp-файл
  N_KADR = 12;//фильмдаги кадрлар сони (берилган файл учун)
var
  Film: TBitMap;       //фильм - ҳамма кадрлар
  Kadr: TBitMap;       //жорий кадр
  WKadr,HKadr: integer;// кадрнинг кенглиги ва баландлиги
  CKadr: integer;      //жорий кадр номери
  RectKadr: TRect;    //кадрнинг фильмдаги ҳолати ва ўлчамларие
Rect1:Trect;//фильмни кўрсатиш соҳасининг координата ва ўлчами
```

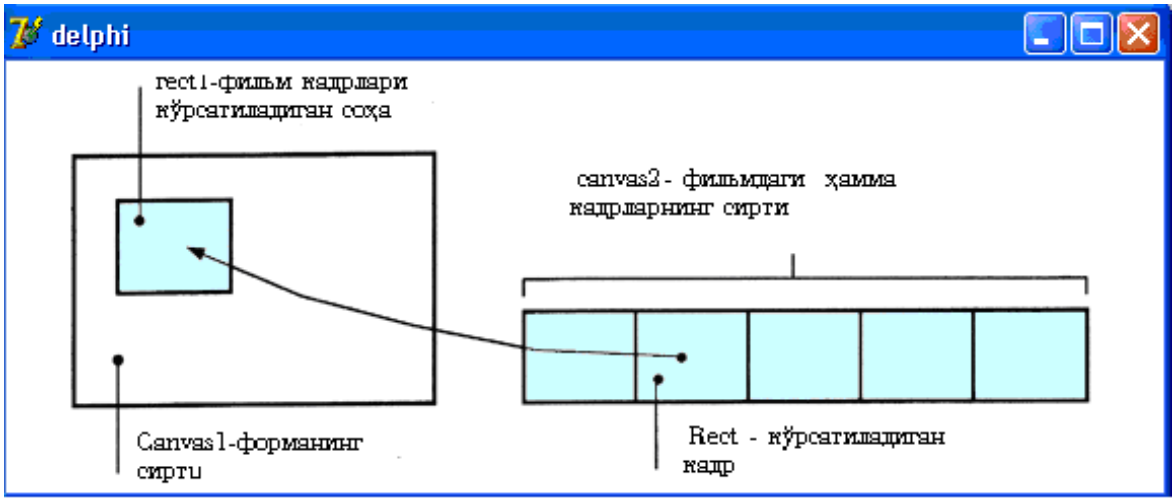
```
procedure TForm1.FormActivate(Sender: TObject);
  begin
      Film := TBitMap.Create;
      Film.LoadFromFile(FILMFILE);
      WKadr :=  Round(Film.Width/N_Kadr);
      HKadr := Film.Height;
      Rect1 := Bounds(10, 10, WKadr, HKadr);
      Ckadr := 0;
      Form1.Timer1.Interval := 150;//кадрларни янгилаш даври
      Form1.Timer1.Enabled := True;//таймерни ишга тушириш
  end;

//кадрни кўрсатиш
  procedure DrawKadr;
  begin
   //жорий кадрнинг филмдаги ўрнини топамиз
    RectKadr := Bounds(WKadr*CKadr, 0, WKadr, HKadr);
   //кадрни фильмдан чиқариш
    Form1.Canvas.CopyRect(Rect1,Film.Canvas,RectKadr);
   //навбатдаги кадрни чиқаришга тайёргарлик
    CKadr := Ckadr + 1;
    if CKadr := N_KADR
       then CKadr := 0;
  end;
          //таймер сигналини қайта ишлаш
  procedure TForm1.Timer1Timer(Sender: TObject);
  begin
      DrawKadr;
  end;
  end.
```

Дастурни ишга тушириш

Дастур учта процедурадан иборат. TForm1.FormActivate процедураси Film объектини ташкил қилади ва унга фильмнинг кадрлари ёзилган BMP-файлни юклайди, сўнгра, юкланган битли тасвир ҳақидаги маълумотлардан фойдаланган ҳолда кадрнинг баландлиги ва кенглигини белгилаб беради.

Шундан кейин жорий кадрни сақлаш учун мўлжалланган TBitMap типидаги Kadr объекти яратилади. Шуни ёдда тутиш керакки, Kadr объекти яратилганидан кейин width ва Height ҳусусиятларининг қийматлари аниқланади. Агар шундай қилинмаса, яратилган объект мавжуд, аммо битли тасвир учун хотирадан жой ажратилмайди. TForml. FormActivate ўз ишининг якунида жорий кадрнинг номерини белгилаб қўяди ва таймерни ишга туширади.

Дастурдаги асосий ишни, яъни фильмнинг навбатдаги кадрини ажратиш ҳамда уни формага чиқаришни DrawKadr процедураси бажаради. DrawKadr процедурасини бошқариш, ишга тушириш вазифасини OnTimer ҳодисасининг қайта ишловчи TForm1.Timer1Timer процедураси бажаради.



11.21-расм. Canvas1. CopyRect (Rect1, Canvas2, Rect2) буйруғи canvas сиртининг Rect1 соҳага Canvas2 сиртнинг Rect2 соҳасии кўчиради.

мундарижага қайтиш

12-БОБ. DELPHI НИНГ МУЛЬТИМЕДИАЛИ ИМКОНИЯТЛАРИ

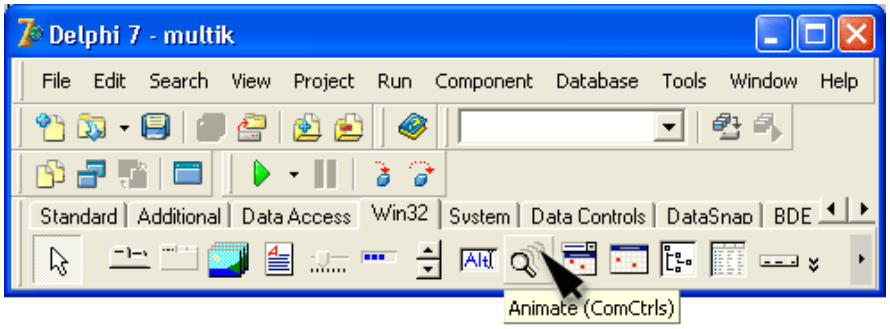
WINDOWS муҳити учун мўлжалланган дастурларнинг каттагина қисми мультимедия дастурларни ташкил қилади. Бундай дастурлар видеороликларни, мультипликация томоша қилиш, мусиқа тинглаш, товушлар ва товушли эффектлар билан ишлашни таъминлайди. Типик мисол сифатида ўйинлар ва ўргатувчи дастурларни кўриш мумкин.

Delphi тили мультимедияли дастур ишлаб чиқиш учун дастурчилар ихтиёрига иккита компонентани таклиф қилади:

- Animate — содда анимацияларни чиқаришга имкон беради (худди файллардан нусха олишда фойдаланувчиларга экранда кўриниб турадиган анимацияларга ўхшаш);
- MediaPlayer — янада мураккаброқ масалаларни ҳал қилиш учун ёрдам беради. Масалан, видеороликлар, овозлар, овозли анимациялар.

12.1. Animate компонентаси

Animate компонентасининг нишони **Win32** қуроллар панелида жойлашган. (12.1-расм.) У кадрлари AVI-файлларда жойлашган содда анимациялар билан ишлашга имкон беради.



12.1-расм. Animate компонентаси

Эслатма: AVI-файлда жойлашган анимациялар овозли эффектлар жўрлигида бўлиши мумкин. Animate компонентаси фақат тасвирларни қайта тиклашни (қўйишни) таъминлай олади. Овозли анимацияларни тўлақонли қўйиш имкониятига эга бўлиш учун **MediaPlayer** компонентасидан фойдаланиш лозим. .

Animate компонентасини формага одатдаги усуллар билан жойлаштириш мумкин. Формага бу компонентани қўшилганидан сўнг унинг ҳусусиятларининг қийматларини белгилаш керак. Бу ҳусусиятлар рўйхати 12.1-жадвалда келтирилган.

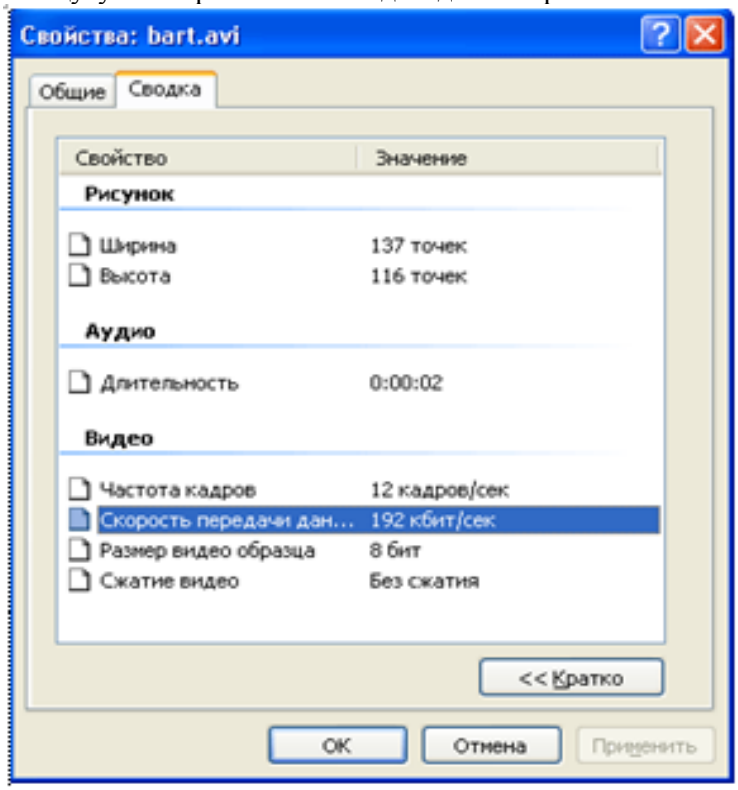
Animate компонентасининг ҳусусиятлари **12.1-жадвал.**

Ҳусусияти	Мазмуни
Name	Компонента номи. Компонента ҳусусиятига мурожаат қилиш, ҳулқини бошқариш учун қўлланади
FileName	Компонента ёрдамида акс эжттириладиган анимация ёзилган AVI-файлнинг номи
StartFrame	Анимацияни қўйиш бошланадиган кадрининг номери
stopFrame	Анимацияни тугатиладиган кадрининг номери
Activate	Анимация кадрларини кўрсатиш жараёнини активлаштириш
Color	Компонентанинг фон ранги. Шу фонда анимациялар кўрсатилади.
Transparent	Анимацияларни кўрсатишда "шаффоф" рангдан фойдаланиш
Repetitions	Анимацияларни такроран кўрсатишлар сони

Шуни ёдда тутиш лозимки, Animate компонентаси фақат анимацияли AVI-файлларни қўйиш учун мослашган. Агар овозли анимацияларни қўйишга уринилса, Delphi кўрсатилган файлни оча олмаслиги ҳақида ахборотни экранга чиқаради: **(Cannot open** AVI).

AVI-файлнинг ичида нима борлигини (фақат анимациями ёки овоз ҳам борми?) кўриш учун Windows муҳитида туриб керакли папкани очилади, AVI-файлни ажратилади ва контекст менюсидан **Свойства** буйруғи танланади. Натижада **Свойства** ойнаси очилади. Бу ойнанинг **Сводка** (12.2-расм.) бўлимида танланган файлнинг мазмуни ҳақидги батафсил маълумот экранга чиқарилади.

Матни 12.1-листингда келтирилган дастур Animate компонентасидан фойдаланиб диалог ойнасида анимацияларни қўйиш жараёнини номойиш қилади. Дастур формасининг кўриниши 12.3-расмда, Animate1 компонентасининг ҳусусиятлар и эса 12.2-жадвалда келтирилган.



12.2-расм. Сводка бўлимида AVI-файл ҳақидаги маълумотлар берилади.

12.3-расм. Анимациялар кўриш дастурининг формаси

Animate1 компонентасининг ҳусусиятлари **12.2-жадвал.**

Ҳусусияти	**қиймати**
FileName	bart.avi
Active	False
Transparent	True

Дастур ишга туширилганидан сўнг, формада анимациянинг биринчи кадри намоён бўлади. Дастур анимацияларни икки ҳил режимда, узлуксиз ва кадрлар режимида кўришга имкон беради.

Button1 тугмаси анимацияларни кўриш жараёнини бошлашга ҳамда тўхтатишга хизмат қилади. Анимацияларни узлуксиз кўрсатиш жараёнини **Бошлаш** тугмасининг **Onclick** ходисаларни қайта ишлаш процедураси ҳал қилади. У Active ҳусусиятига True қийматини беради. Шу процедуранинг ўзи Button1 тугмасидаги **Бошлаш** матнини **Тўхтатиш** га алмаштиради. Анимацияларни қўйиш режими RadioButton1 ва RadioButton2 ўчиргичларидан бирини танлаш орқали белгиланади. Бу ўчиргичлардаги Onclick ходисаларни қайта ишлаш процедуралари Enabled ҳусусиятининг қийматига боғлиқ равишда бошқариш тугмаларига рухсат беради ёки таъқиқлайди: анимацияларни қўйиш жараёнини фаоллаштириш (Button1), навбатдаги (Button2) ва аввалги (Button3) кадрларга ўтиш. Анимацияларни узлуксиз равишда кўриш режимида **Тўхтатиш** (Button1) тугмасининг OnClick ходисаларни қайта ишлаш процедураси Active ҳусусиятига False қийматини беради ва шу билан анимацияларни қўйиш жараёнини тўхтатади.

12.2-листинг. Animate компонетасидан фойдаланиш

```
unit ShowAVI_;
interface
uses  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,  StdCtrls, ComCtrls, ExtCtrls;
type
  TForm1 = class(TForm)
    Animate1: TAnimate; //Animate компонентаси
    Button1: TButton;  //Бошлаш-Тўхтатиш тугмалари
    Button2: TButton;  //навбатдаги кадр
    Button3: TButton;  //аввалги кадр
    RadioButton1: TRadioButton;// тўла анимацияларни кўриш
    RadioButton2: TRadioButton;// кадрлар бўйича кўриш
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
```

```
    procedure RadioButton1Click(Sender: TObject);
    procedure RadioButton2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1; //форма
  CFrame: integer;//кўрсатилаётган кадрнинг номери
                  //кўришнинг кадрлар режимида
implementation
{$R *.DFM}
//навбатдаги кадрга
procedure TForm1.Button2Click(Sender: TObject);
begin
    if CFrame = 1 then Button2.Enabled := True;
    if CFrame < Animate1.FrameCount then
    begin
      CFrame := CFrame + 1;
      //кадрни чиқариш
      Animate1.StartFrame := CFrame;
      Animate1.StopFrame := CFrame;
      Animate1.Active := True;
      if CFrame = Animate1.FrameCount//жорий кадр - охиргиси
           then Button2.Enabled := False;
    end;
end;
//аввалги кадрга
procedure TForm1.Button3Click(Sender: TObject);
begin
    if CFrame = Animate1.FrameCount
       then Button2.Enabled := True;
    if CFrame > 1 then
    begin
      CFrame := CFrame - 1;
      //кадрни чиқариш
      Animate1.StartFrame := CFrame;
      Animate1.StopFrame := CFrame;
      Animate1.Active := True;
      if CFrame = 1//жорий кадр - биринчиси
         then Form1.Button3.Enabled := False;
    end;
end;
//анимацияни тўла кўриш режимини фаоллаштириш
procedure TForm1.RadioButton1Click(Sender: TObject);
begin
    Button1.Enabled := True;//Бошлаш тугмаси ишлайди
    //Кадрлар тугмасини ўчириб қўйиш
    Form1.Button3.Enabled := False;
    Form1.Button2.Enabled := False;
end;
//  кадрлар бўйича кўриш режимини фаоллаштириш
procedure TForm1.RadioButton2Click(Sender: TObject);
begin
    Button2.Enabled := True;//Навбатдаги кадр тугмаси ишлайди
    Button3.Enabled := False;//Аввалиги кадр тугмаси ишламайди
```

```
    //Бошлаш тугмасини ўчириб қўйиш
     Button1.Enabled := False;
end;
//анимациялар кўришни бошлаш ва тўхтатиш
procedure TForm1.Button1Click(Sender: TObject);
begin
 if Animate1.Active = False//жорий вақтда анимация кўрсатилмайди
    then begin
    Animate1.StartFrame := 1;//бири нчидан бошлаб чиқариш
    Animate1.StopFrame := Animate1.FrameCount;//охирги кадргача
        Animate1.Active := True;
        Button1.caption := 'Тўхтатиш';
        RadioButton2.Enabled := False;
       end
     else //анимация кўрсатилмоқда
       begin
        Animate1.Active := False;//кўрсатишни тўхтатиш
        Button1.caption := 'Бошлаш';
        RadioButton2.Enabled := True;
       end;
end;
end.
```

Дастурни ишга тушириш

Animate компонетаси дастурчига ўзининг дастурларида Windows даги стандарт анимациялардан фойдаланишга имкон беради. Анимацияларнинг кўриниши CommonAVI ҳусусиятининг қиймати билан белгиланади. Бу қийматлар номланган константалар орқали берилади. 12.3-жадвалда айрим номланган константалар, анимация кўриниши ва анимация маъноси келтирилган.

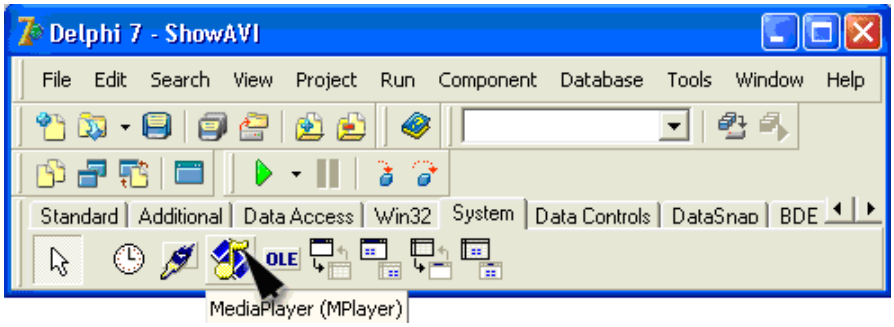
ComonAVI ҳусусиятининг айрим қийматлари **12.3-жадвал.**

Қиймати	Анимацияси	Маъноси
aviCopyFiles		Файллардан нусха олиш
AviDeleteFile		Файлларни ўчириш
AviRecycleFile		Файлларни корзинага жўнатиш

менюга

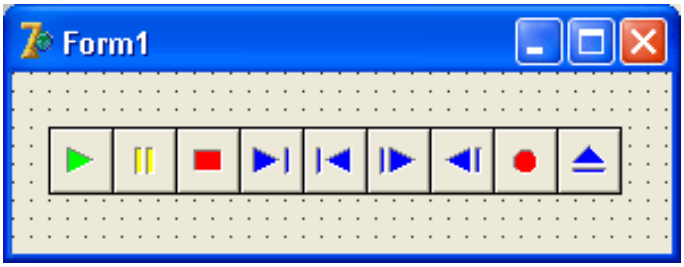
12.2. MediaPlayer компонентаси

MediaPlayer компонетасининг нишони **System** куроллар панелида (12.4-расм.) жойлашган. Бу компонента видеорликлар, овоз ва овозли анимацияларни кўриш учун мўлжалланган.



12.4-расм. MediaPlayer компонентасининг нишони

Формага MediaPlayer компонетаси қўшилганидан сўнг, формада тугмалар группаси пайдо бўлади. (12.5-расм.) Бу тугмаларнинг вазифалари 12.4-жадвалда келтирилган. MediaPlayer компонентасининг ҳусусиятлари эса 12.5-жадвалда берилган.



12.5-расм. MediaPlayer компонентаси

MediaPlayer компонентасининг тугмалари **12.4-жадвал.**

Тугма	**Белгиланиши**	**Вазифаси**
Қўйишни бошлаш	btPlay	Овоз ёки видео кўйишни бошлаш
Пауза	btPause	Кўрсатишда пауза қилиш
Стоп	btStop	Кўрсатишни тўхтатиш
Навбатдагиси	btNext	Навбатдаги кадрга ўтиш
Аввалгиси	btPrev	Аввалги кадрга ўтиш
Қадам	btStep	Навбатдаги овозли парчага ўтиш, масалан, CD даги навбатдаги музикага ўтиш
Орқага	btBack	Аввалги овозли парчага ўтиш, масалан, CD даги аввалги музикага ўтиш
Ёзиш	btRecord	Ёзиш
Очиш/Ёпиш	btEject	CD-диск юритувчини очиш ёки ёпиш

MediaPlayer компонентаснинг ҳусусиятлари **12.5-жадвал.**

Ҳусусияти	Вазифаси
Name	Компонентанинг номи. Компонентаннинг ҳусусиятлари ва плеер билан ишлашда қўлланади.
DeviceType	Қурилманинг номи. MediaPlayer компонентаси аниқ қайси қурилма эканлигини белгилайди. қурилма номи номланган константа билан кўрсатилади: DtAutoselect - қурилмани автоматик тарзда аниқлайди; dtVaweAudio — овоз проигри-вателн; dtAVivideo — видеопроигриватель; dtCDAudio — CD-проигриватель

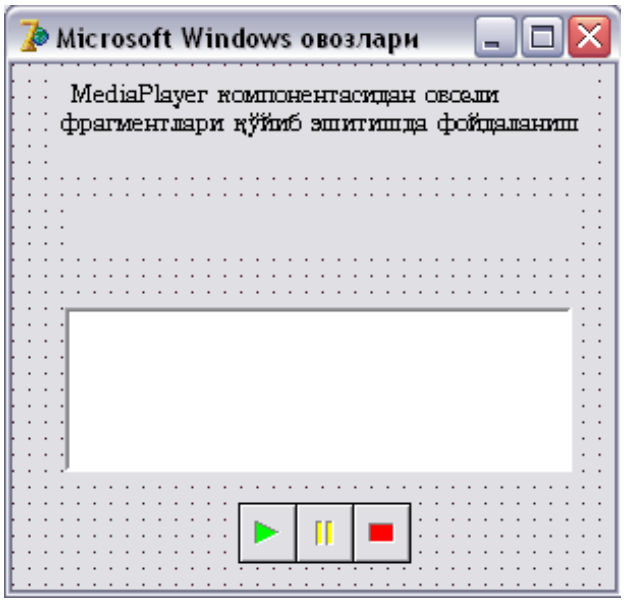
FileName	Файлнинг номи. Қўйиб эшитилмоқчи бўлган овозли фрагмент ёки видеоролик ёзилган файл.
AutoOpen	Овозли фрагмент ёки видеоролик ёзилган файлни дастур ишга тушганда автоматик очилиш белгиси.
Display	Сиртида видеоролик қўйиладиган компонентанинг номи. (Одатда видео кўрсатиш учун экран сифатида Panel дан фойдаланилади.
VisibleButtons	Таркиб ҳусусияти. Компонентанинг кўринадиган тугмаларини белгилайди. Айрим тугмаларни кўринмайдиган қилишга имкон беради.

Овозларни эшитиш. Овозли фрагментлар одатда кенгайтмаси WAV бўлган файлларда сақланади. Масалан, C:/Winnt/Media каталогида Windows нинг стандарт овозли файллари сақланади.

Қуйидаги дастур (матни 12.2-листингда, диалог ойнаси эса 12.6-расмда келтирилган) MediaPlayer компонентасидан овозли фрагментларни (WAV-файлларда ёзилган) қўйиб эшитиш учун фойдаланиш усулини намойиш қилади.

Формада MediaPlayer компонентасидан ташқари ListBox ва иккита Label (биринчиси – ахборотларни чиқариш учун, иккинчиси эса - фойдаланувчи умумий рўйхатдан танлаган фрагмент файлининг номи) компоненталари жойлаштирилган.

Ушбу дастур қуйидагича ишлайди. Экранда диалог ойнаси пайдо бўлганидан кейин "Звук Microsoft" фрагменти қўйилади (эшитилади), сўнгра фойдаланувчи таклиф қилинган рўйхатдан C: / Windows / Media каталогида жойлашган ихтиёрий овозли файлни танлаши мумкин. **Эшитиш** тугмаси чертилганидан сўнг, бу файлда қандай овоз ёзилганлигини эшитиб кўриш мумкин.



12.6-расм. Microsoft Windows овозлари дастурининг диалог ойнаси

MediaPlayer1 компонентаси ҳусусиятларининг ўзгарган қийматлари 12.6-жадвалда келтирилган, қолган ҳусусиятларниг қийматлари ўзгармайди.

MediaPlayer1 компонентасининг ҳусусиятлари **12.6-жадвал.**

Компонента	**қиймати**
DeviceType	DtAutoSelect
FileName	C: / Windows / Media / Звук Microsoft-запуск.wav
AutoOpen	True
VisibleButtons . btNext	False

VisibleButtons .btPrev	False
VisibleButtons . btStep	False
VisibleButtons . btBack	False
VisibleButtons . btRecord	False
VisibleButtons .btEject	False

12.3-листинг. Microsoft Windows овозларини тинглаш

```
unit WinSound_;
interface
uses  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,   StdCtrls, MPlayer;
type
  TForm1 = class(TForm)
    MediaPlayer1: TMediaPlayer;//медиаплеер
    Label1: TLabel;              //ахборот-эслатма
    ListBox1: TListBox;          //WAV-файллар рўйхати
    Label2: TLabel;              //рўйхатдан танланган файл
procedure FormActivate(Sender: TObject);
procedure ListBox1Click(Sender: TObject);
procedure MediaPlayer1Click(Sender: TObject; Button: TMPBtnType;
     var DoDefault: Boolean);
   procedure MediaPlayer1Notify(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
const
SOUNDPATCH = 'c:/windows/media/'; //овозли файллар каталоги
  ModeStr: array[TMPModes] of string = ('Not ready', 'Stopped', 'Playing', 'Recording', 'Seeking', 'Paused', 'Open');
  var
  Form1: TForm1;
implementation
{$R *.DFM}

procedure TForm1.FormActivate(Sender: TObject);
var   SearchRec: TSearchRec;
//кидириш шартини қаноатлантирувчи структура
begin
    Form1.MediaPlayer1.Play;
    //c:/windows/media каталогидаги WAV-файллар рўйхати
    if FindFirst(SOUNDPATCH + '*.wav', faAnyFile, SearchRec) = 0 then
    begin //каталогда WAV кенгайтмали файл мавжуд
        //Бу файлнинг номини рўйхатга қўшамиз
        Form1.ListBox1.Items.Add(SearchRec.Name);
        //хозирча каталогда WAV кенгайтмали бошқа файллар бор
        while (FindNext(SearchRec) = 0) do
        Form1.ListBox1.Items.Add(SearchRec.Name);
    end;
end;

//рўйхатдаги элемент чертилса
procedure TForm1.ListBox1Click(Sender: TObject);
begin
//Label2 майдонида танланган файл номини кўрсатиш
 Label2.Caption := ListBox1.Items[ListBox1.itemIndex];
end;
```

```
// MediaPlayer компонентаси тугмаси чертилса
procedure TForm1.MediaPlayer1Click(Sender: TObject; Button: TMPBtnType;
  var DoDefault: Boolean);
begin
    if (Button = btPlay) and (Label2.Caption <> '') then
    begin       // Play тугмаси босилганда
      with MediaPlayer1 do
        begin
FileName := SOUNDPATCH+Label2.Caption;// танланган файл номи
          Open;
          //MediaPlayer1.Wait := True;
          // Play;
        end;
    end;
end;
procedure TForm1.MediaPlayer1Notify(Sender: TObject);
begin
with Sender as TMediaPlayer do
  begin
    Form1.Caption := ModeStr[Mode];
    { Note we must reset the Notify property to True }
    { so that we are notified the next time the }
    { mode changes }
    Notify := True;
  end;
  end;
end.
```

Дастурни ишга тушириш

Дастур ишга тушган заҳоти onFormActivate ходисаларни қайта ишлаш процедураси MediaPlayer1 компонентасига нисбатан Play методини қўллайди. (Бу методнинг вазифаси **Эшитиш** тугмасининг вазифаси билан бир ҳил.) Шу процедуранинг ўзи *C:\WINDOWS\MEDIA* каталогидаги WAV-файллари рўйхатини аниқлайди. Рўйхатни ҳосил қилишда FindFirst ва FindNext функцияларидан фойдаланилади. Улар мос равишда биринчи ва навбатдаги қидириш шартини қаноатлантирувчи файлларни қидиради. Хар икки функцияга параметр сифатида WAV-файл ниқоби (қидириш шарти) ҳамда Name майдони қидириш шартини қаноатлантирувчи файл номига тенг бўладиган SearchRec структура-ўзгарувчиси берилади.

Рўйхатдаги элементлардан бирини чертиш ходисасига TForm1.ListBox1Click процедураси жавоб беради. У Label2 майдонида фойдаланувчи танлаган файл номини чиқаради (Дастур ишлаётган вақтда ItemIndex ҳусусиятининг қиймати рўйхатдан танланган элементнинг тартиб номерига тенг бўлади.).

MediaPlayer1 тугмаларидан бирини чертиш билан TForm1.MediaPlayer1Click процедураси фаоллашади. У тугмалардан қайси бири чертилганини аниқлайди. Агар **Эшитиш** (btPlay) тугмаси чертилган бўлса, у ҳолда MediaPlayer1 компонентасининг FileName ҳусусиятига фойдаланувчи танлаган файлнинг номи ёзилади, сўнгра Open методи уни юклайди ва эшитиш жараёнини фаоллаштиради.

MediaPiayer компонентасида visible ҳусусиятининг мавжудлиги фойдаланувчидан компонентани кўздан яширишга ҳамда фойдаланувчининг иштирокисиз овозли файлни ишга туширишга имкон беради.

менюга

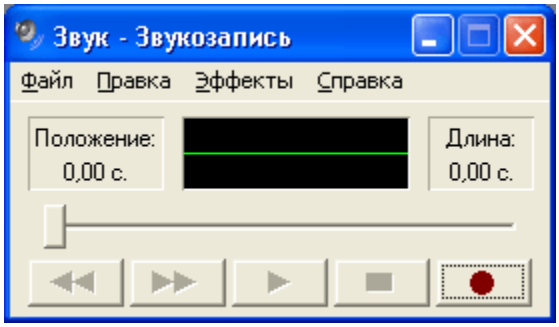
12.3. Овозларни ёзиш

Айрим ҳолларда дастурчига дастур ёзиш жараёнида махсус товушлардан ёки дискларда WAV-файли шаклида сақланмаган музикали парчалардан фойдаланишга тўғри келиб қолади. Бу ҳолда дастурчи олдида янги WAV-файлини яратиш вазифаси пайдо бўлади.

Керакли овозли парчани WAV-файли шаклида ифодалашнинг энг осон усули – бу Windows таркибига кирган **Звукозапись** дастуридир. **Звукозапись** дастури, унинг диалог ойнасининг кўриниши 12.8-расмда келтирилган. Бу дастурни Windows нинг бош менюсидан

Пуск / *Программы* / *Стандартные* / *Развлечения* / *Звукозапись*

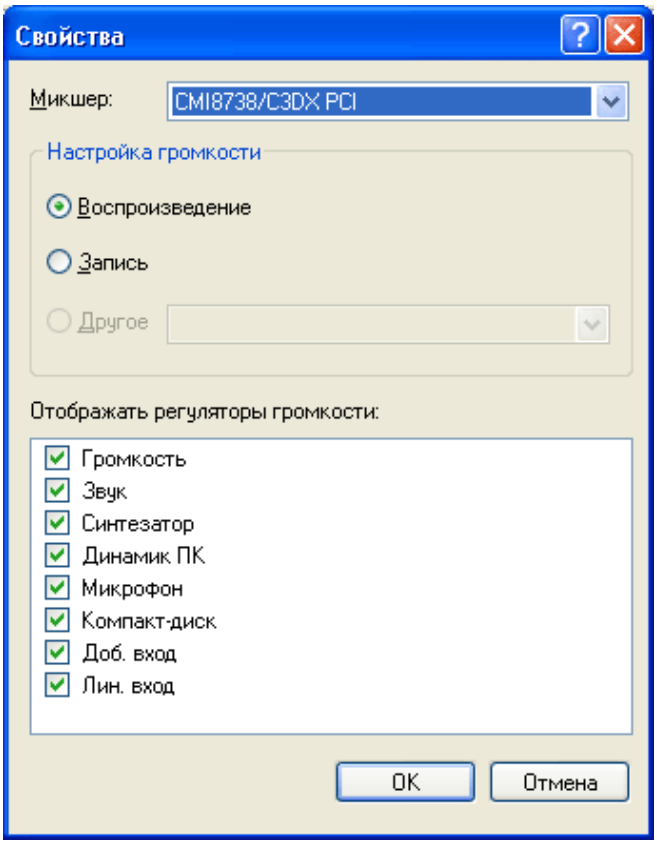
буйруқлари ёрдамида ишга туширилади.



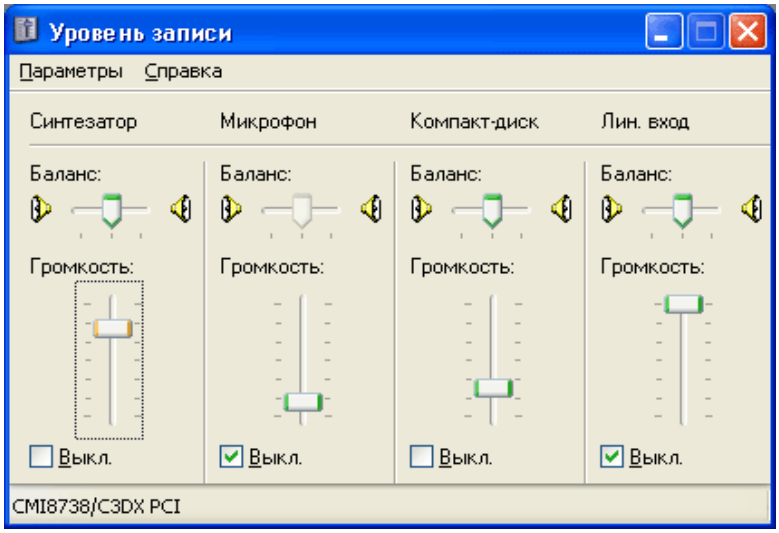
12.8-расм. **Звукозапись** дастурининг диалог ойнаси

Звукозапись дастури учун овоз манбаси сифатида микрофон, аудио-CD ёки компьютер овоз платасининг **Кириш** уясига уланган ихтиёрий овоз қурилмасини, масалан, аудиомагнитофонни олиш мумкин. Бундан ташқари, бир нечта овозларни аралаштириш имконияти ҳам мавжуд.

WAV-файллари қуйидагича усул билан яратилади. Дастлаб овоз манбасини аниқланади. Бунинг учун **Регулятор громкости** ойнаси очилади. (Бу ойна масалалар панелидаги динамик тугмасини чертиш ва эаранда пайдо бўлган ойнадан **Регулятор громкости** буйруғини танлаш орқали очилади.**)** Сўнгра, **Параметрў** менюсидан **Свойства** буйруғи танланади. Экранда очилган **Свойства** ойнасидан (12.9-расм.) **Запись** ўчиргичи танланади ва **Отображаемўе регуляторў громкости** рўйхатидан овоз манбасига мос келадиган қурилмаларга байроқчалар ўрнатилади. ОК тугмаси чертилганидан сўнг, экранда **Уровень записи** ойнаси (12.10-расм.) пайдо бўлади. Ундан фойдаланиб, ҳар бир манбадан келаётган сигналларнинг товуш даражаларини (баланд-пастлигини) ҳамда **Звукозапись** дастурига кираётган умумий сигналларни бошқариш мумкин. Сигналларнинг қиймати регуляторлардаги ана шу сигналларга мос келадиган харакатларга қараб белгиланади. Шуни ёдда тутиш керакки, **Уровень** гуруҳидаги регуляторлардаги харакатлар билан фақат товуш ёзиш жараёнидагина ишлаш мумкин. Шу билан овоз ёзиш учун керак бўладиган тайёргарлик жараёни тугайди. Энди бевосита овоз ёзишга ўтиш мумкин.



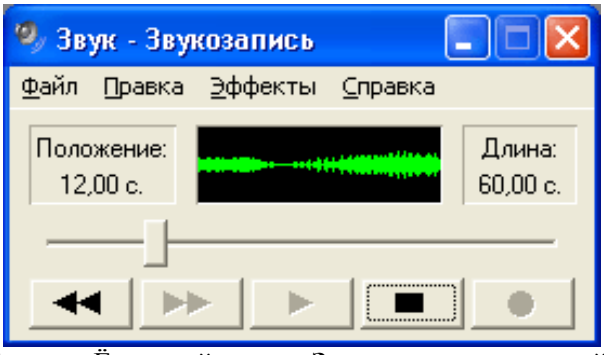
12.9-расм. Свойства диалог ойнаси



12.10-расм. Уровень записи диалог ойнаси.

Музика ёки инсон овози туширилган фрагментни ёзиш учун **Звукозапись** дастурини ишга тушириб, **Уровень** диалог ойнаси очилади. Овоз манбаси бўлган қурилма танланади, овознинг чиқиш ҳолати текширилади, (масалан, овоз CD дан олинаётган бўлса) ва керакли пайтда **Запись** тугмаси чертилади.

Ёзиш вақтида диалог ойналарида сигналларнинг микшердан чиқишдаги ҳамда ёзиш дастурига киришдаги товушнинг ўзгаришини кузатиш мумкин. (**Уровень** диалог ойнасидаги **Громкость** индикатори**)**. 12.11-расмда намуна сифатида овоз ёзиш вақтидаги **Звукозапись** диалог ойнасининг кўриниши тасвирланган.



12.11-расм. Ёзиш пайтидаги **Звукозапись** диалог ойнаси

Овоз ёзиш жараёнини **Стоп** тугмасини чертиш орқали тўхтатиш мумкин.

Ёзиб олинган фрагмент файлда одатдаги усул билан сақлаб қўйилади, яъни **Файл** менюсидан **Сохранить** ёки **Сохранить как** буйруқларидан бирини танланади. **Сохранить как** буйруғи танланганда, ёзиб олинган овозли фрагментни сақлаш учун бошқа форматни ҳам белгилаш мумкин.

Овозли файлларнинг бир нечта форматлари мавжуд. Ҳусусан, овозни стерео ёки моно сифати билан сақласа бўлади. Овозли файлларни сақлаётганда, уларга фомат танлашда шуни ёдда тутиш керакки, ёзув сифати қанча юқори бўлса, уни сақлайдиган файл компьютер хотирасидан шунча кўп жойни банд қилади. Инсон овози ёзилганда, "22050 Гц, 8 бит, моно" формати, музикаларни сақлаш учун эса - "44100 Гц, 16 бит, моно" ёки "44100 Гц, 16 бит, стерео" формати етарли ҳисобланади.

менюга

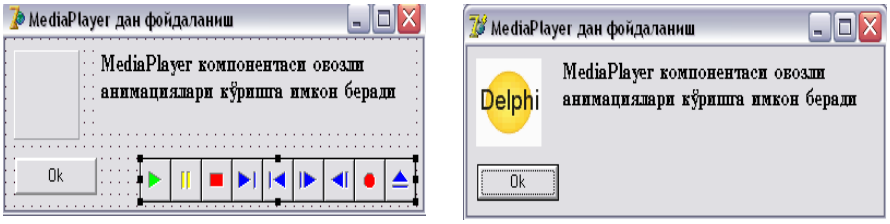
12.4. Видеоролик ва анимацияларни кўриш

MediaPlayer компонентаси овозли файлларни эшитишдан ташқари, AVI-файл шаклида сақланган видеороликлар ва мультипликацион фильмларни томоша қилишга имкон беради. AVI — бу Audio Video Interleave сўзининг қисқарттирмаси бўлиб, "овоз ва видеонинг навбатма-навбат келиши" қабилида таржима қилиниши мумкин. Демак, AVI-файлда ҳам овозли, ҳам тасвирли маълумотлар биргаликда сақланади.

AVI-файллардаги маълумотларни кўриш учун MediaPlayer компонентасидан фойдаланиш мумкин.

Бу жараённи қуйидаги дастур ёрдамида изоҳлаймиз. Ушбу дастур ойнасидаги буйруқли тугма чертилса, форма сиртида оддий ва овозли эффект билан биргаликдаги мультипликация-соат стрелкаси бўйича айланаётган Delphi сўзи пайдо бўлади. (бу мультипликация delphi.avi файлидан сақланади.).

Дастурнинг диалог ойнаси 12.12-расмда, MediaPlayer1 компонентасининг ҳусусиятлари эса 12.8-жадвалда берилган.



12.12-расм. MediaPlayer дан фойдаланиш дастурининг диалог ойнаси

MediaPlayer1 компонентасининг ҳусусиятлари **12.8-жадвал.**

Ҳусусияти	**Қиймати**
Name	MediaPlayer1
FileName	delphi.avi
DeviceType	dtAVIVideo
AutoOpen	True
Display	Panel1
Visible	False

Илова формаси одатдаги усуллар билан ҳосил қилинади. Panel1 компонентаси экран сифатида фойдаланилади ва унинг номини MediaPlayer1 компонентаси Display ҳусусиятига қиймат қилиб берилади. Шунинг учун формага дастлаб Panel компонентасини, сўнгра MediaPlayer компонентасини жойлаштириш лозим. Форма яратишдаги бундай тартиб Display ҳусусиятига қийматни рўйхатдан танлаш усули билан беришга имкон яратади.

Шуни эсда тутиш кераки, анимацияларни панелдаги чиқариш соҳаси панелнинг Width ва Height ҳусусиятлари қийматлари билан белгиланмайди. Бу соҳанинг ўлчами MediaPlayer компонентасининг ҳусусияти билан аниқланади. DisplayRect ҳусусияти билан дастурни ишлаб чиқаётган вақтда ишлаб бўлмайди. (Унинг қиймати **Object Inspector** ойнасига чиқарилмайди.**)** Шунинг учун DisplayRect ҳусусиятининг қиймати дастур ишлаб турган пайтда

MediaPlayer1.DisplayReet := Rect(0,0,60,60)

буйруғи билан ўрнатилади.

Эслатма: AVI-файлидаги кадрларнинг ўлчами ҳақида ахборот олиш учун Windows нинг имкониятларидан фойдаланиш лозим, яъни бу файл сақланаётган папкани очиш керак. Файлнинг номини сичқонча ўнг тугмаси билан чертилади. Экранга чиқарилган контекст менюдан **Свойства** буйруғини танланади ва пайдо бўлган диалог ойнасидан **Сводка** тугмасини чертилади. Натижада белгиланган файл ҳақидаги маълумотлар, шу жумладан кадрларнинг ўлчами ҳақидаги маълумотлар экранга чиқарилади.

Дастурнинг матни 12.4- листингда келтирилган.

12.4-листинг. Овозли анимацияни қўйиб кўриш

```
unit UsMP_;
interface
uses   Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,   MPlayer, StdCtrls, ExtCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;          //информацион
    Panel1: TPanel;          //анимация чиқариладиган панель
    Button1: TButton;        //кнопка Ok
    MediaPlayer1: TMediaPlayer;//универсал проигриватель
```

```
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.Button1Click(Sender: TObject);
begin
     form1.MediaPlayer1.Play;//анимацияни қўйиш
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
//форма сиртида анимация қўйиладиган соҳа ўлчамини аниқлаймиз
     MediaPlayer1.DisplayRect := Rect(0, 0, 60, 60);
end;
end.
```

Анимацияларни қўйиб кўриш жараёни **Play** методи ёрдамида фаоллаштирилади. Бу эса агар **MediaPlayer** компонентасининг тугмаси билан фойдаланувчига ишлаш учун рухсат берилган бўлса, **Play** тугмасини чертиш билан эквивалент.

менюга

12.5. Анимациялар яратиш.

Анимациялар файлини (AVI-файли) яратиш жараёнини мисол орқали изоҳлаймиз. Delphi ибодатхонаси тасвирининг анимациясини яратиш талаб қилинган бўлсин. Бу расмнинг тугалланган варианти 12.13-расмда, анимациянинг бир нечта кадрлари эса 12.14-расмда келтирилган.

12.13-расм. Delphi ибодатхонаси

Бу масалани ечиш учун Macromedia Flash 5 дастуридан фойдаланиш мумкин. Macromedia Flashда анимациялар (уларни роликлар – Movie ҳам дейилади) қатламлардан иборат бўлади. Энг

 1	 2	 3	 4	 5	 6
		 7		 37	

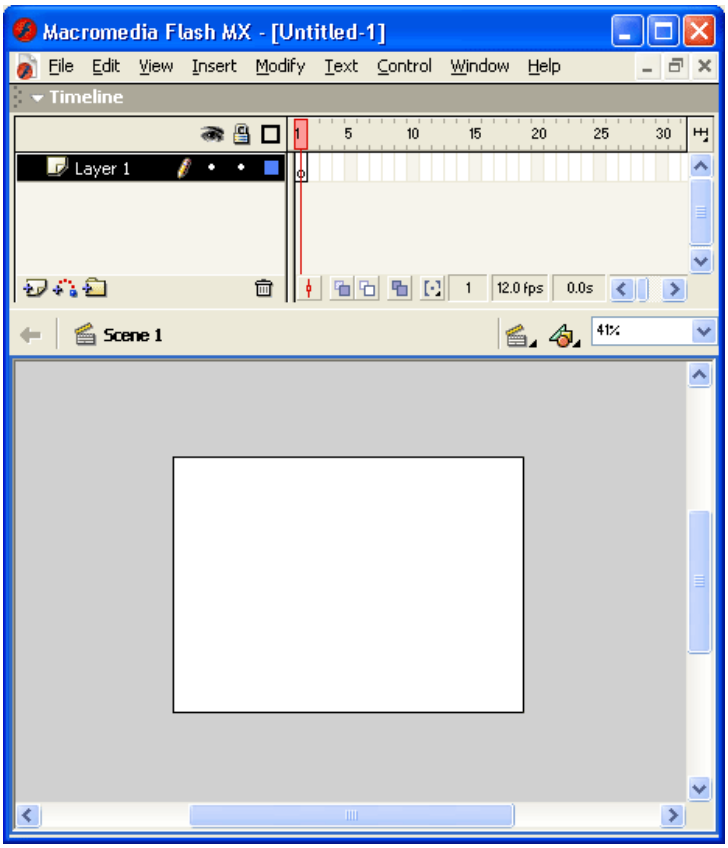
12.14-расм. Delphi ибодатхонаси анимацияларининг кадрлари

содда ҳолда ролик битта қатламдан (Layer) иборат бўлади. Қатлам – бу анимацияларни қўйиш жараёнида навбатма - навбат кўрсатиладиган кадрлар (Frame) кетма-кетлигидир. Агар ролик бир нечта қатламдан иборат бўлса, у ҳолда анимация битта қатламдаги кадрларни бошқа қатламдаги кадрларнинг устига қўйишдан ҳосил бўлади. Масалан, битта қатламда фон бўлса, иккинчи қатламда шу фон устида рўй берадиган ходиса ёки персонажлар тасвири. Қатламларни бир-бирининг устига қўйиб тасвир ҳосил қилиш анимациялар яратиш жараёнини енгиллаштиради. Шундай қилиб, анимация яратиш учун тасвирни

қатламлар бўйича ёйиб чиққан ва ҳар бир қатлам учун кадрлар яратган маъқул.

Macromedia Flash ишга туширилганидан сўнг дастурнинг бош ойнаси фонида **Move1** ойнаси (12.15-расм.) пайдо бўлади. Ундан анимациялар яратишда фойдаланилади. Ойнанинг TimeLine деб аталадиган юқори қисмида анимация структураси, ишчи соҳа деб аталадиган қуйи қисмида эса танланган қатламдаги жорий кадр тасвири кўриниб туради. Macromedia Flash ишга тушганда анимация битта қатламдан (Layer1) иборат бўлиб, унда битта "тоза" кадр бўлади.

Анимация яратишга киришишдан аввал анимация (ролик) нинг умумий характеристикаларини, яъни кадрларнинг ўлчами ва уларни экранга узатиш тезлигини белгилаб қўйиш лозим. Бу характеристикалар **Movie Properties** диалог ойнасининг (12.16-расм.) махсус майдонларига киритилади. Бу диалог ойнани экранга **Modify** менюсидан **Movie** буйруғини танлаш орқали чақириш мумкин. **Frame Rate** ойнасига роликда кадрларни алмашиш тезлигини кўрсатилади. Бу тезлик секундига ўтадиган кадрлар сони билан белгиланади (fps — frame per second, секундига кадрлар).

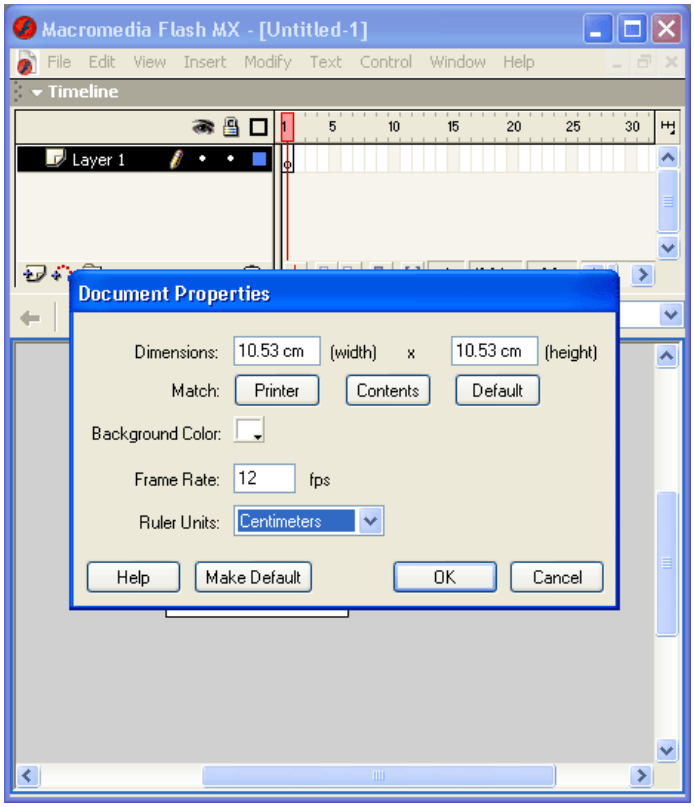


12.15-расм. Янги анимация устида иш бошлашда Movie ойнаси

Width ва **Height** майдонларига эса — кадрларнинг кенглиги ва баландлигини киритилади. Шу ойнанинг ўзида кадрлар учун фон танлаш ҳам мумкин. (**Background Color** рўйхати**).**

Роликнинг характеристикалари белгиланганидан сўнг, анимация кадрларини яратишни бошлаш мумкин.

Биринчи кадрни чизиш лозим. Macromedia Flash да тасвирларни чизиш жараёни оддий. расм чизишнинг стандарт қуроллардан: қалам, чўтка, мўйқалам, пульверизатор, ўчиргич ва х.к. фойдаланиш мумкин. Навбатдаги кадрни яратиш учун **Insert** менюсидан **Keyframe** буйруғи танланади. Натижада жорий қатламга аввалги кадрдаги тасвир кўчирилган янги кадр қўшилади. Бунинг сабаби шуки, одатда, янги кадр аввалгисини бир оз ўзгартириш ҳисобига яратилади. Энди иккинчи кадрни чизишни бошлаш мумкин. Қолган кадрлар ҳам ана шу усулда ҳосил қилинади.



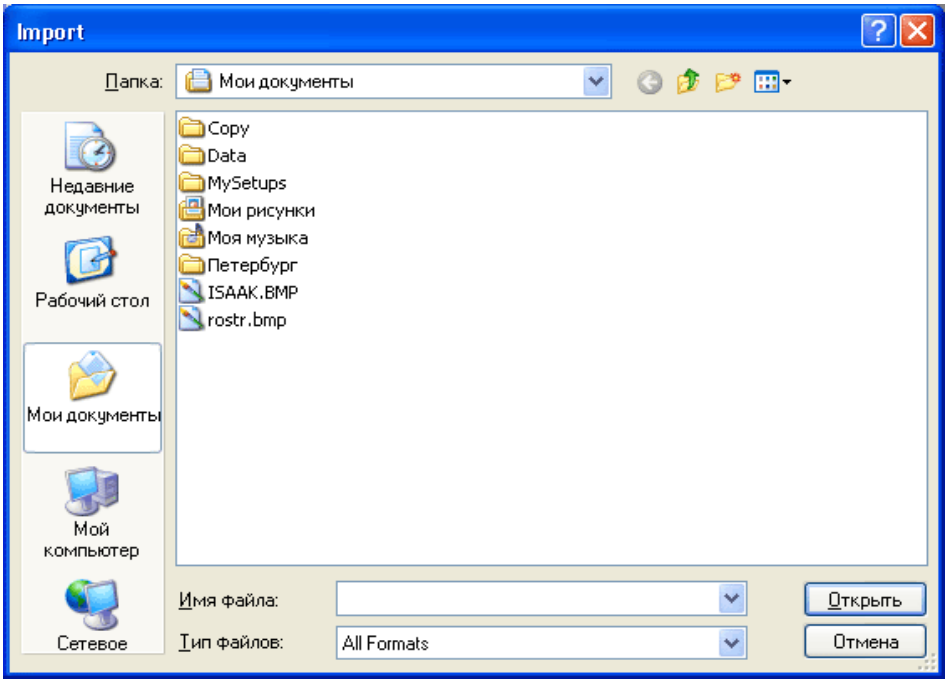
12.16-расм. Movie Properties ойнасидаги роликнинг характеристикалари

Айрим ҳолларда янги кадр аввалги кадрдаги тасвирларни ўз ичига умуман олмаслиги мумкин. Бу ҳолда **Keyframe** буйруғи ўрнига **Blank Keyframe** буйруғидан фойдаланиш мумкин.

Агар айрим тасвирлар маълум бир муддатга, бир нечта кадрлар чиққанда ҳам экранда сақланиб туриши керак бўлса, у ҳолда қатламга бир нечта бир ҳил кадрларни (Keyframe) қўйиш ўрнига, кадрни статик қилиш тавсия қилинади. Агар, тасвири статик бўлиши керак бўлган кадр роликнинг охирги кадри бўлса, TimeLine ойнасида статик бўлиши талаб қилинган кадрдан кейинги кадрни ажратиб, **Insert** менюсидан **Frame** буйруғи танланади. Агар статик бўлиши талаб қилинган кадр роликнинг охирги кадри бўлмаса, бу кадрни ажратиш ва бир неча марта **Insert** менюсидан **Frame** буйруғини танлаш лозим.

Агар тасвирларни асосий ва фон рассмларга ажратиш мумкин бўлса, анимацияларни яратиш жараёни анчагина осонлашган бўлар эди. Бунда уларнинг ҳар бирини алоҳида қатламларга жойлаштирилади. (мультфилмлар ана шу усул билан ишлаб чиқилади.) Дастлаб юқорида айтиб ўтилганидек, фон қатламининг кадри яратилади, сўнгра **Insert** менюсидан **Layer** буйруғини танлаб асосий харакат қатламини қўшилади. Шунга эътибор қаратиш керакки, тасвирларни таҳрирлаш бўйича барча харакатлар танланган қатламдаги жорий кадрга қаратилади. қатламлар рўйхатида танланган қатлам ранги билан, жорий кадрнинг номери эса маркер-қизил квадрат билан ажратиб кўрсатилади.

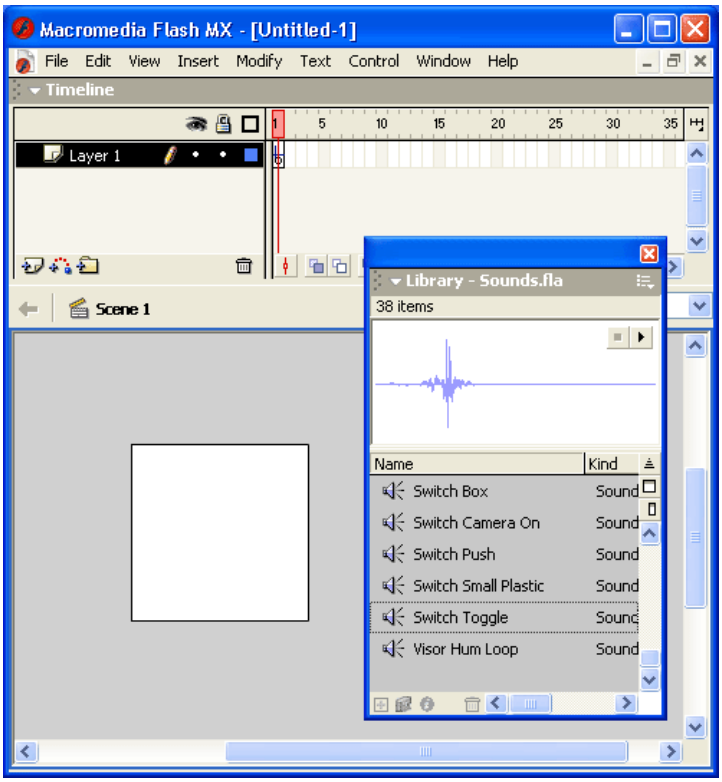
Анимация овозли бўлиши учун дастлаб овозли файлга рухсат берилади. Бунинг учун **File** менюсидан **Import** буйруғини танлаб, лойихага овозли файлни қўшиб қўйиш керак. (12.17-расм.)



12.17-расм. Овозли файлни лойихага қўшиш

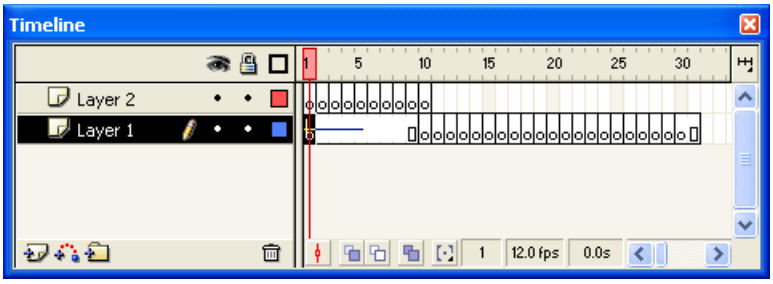
Сўнгра, Timeline ойнасида очилиши билан овозли файл бошланиши керак бўлган кадр **Sound** (12.18-расм.) диалог ойнасидан фойдаланиб танланади. Бунда овозли фрагмент файли ва зарур бўлса, уни қўйиш параметрлари белгиланади. Овозли фрагментнинг такрорланишлари сони **Loops** майдонида кўрсатилади, қўйиш вақтидаги эффект эса **Effect** рўйхатидан танланади.

Намуна сифатида 12.19-расмда анимациялар устида ишлаш жараёнининг якунидаги Timeline ойнасининг кўриниши келтирилган. Анимация иккита қатламдан иборат. Layer2 қатламида фон чизилган. Фоннинг деталлари аста-секин, 9 та кадр давомида пайдо



12.18-расм. Sound диалог ойнаси

бўлади. Шундан кейин фон ўзгармайди, шунинг учун 9 - кадр статик ҳисобланади. Layer1 қатламида асосий харакат тасвирланган. У фон тўла чиқарилганидан кейин бошланади. Анимациянинг чиқиши стандарт овоз-**TADA** (унинг давомийлиги 1 секунд) билан тугайди. Овознинг чиқиши охирги асосий харакатнинг (ролик бошидан санаганда 49-чи кадр) кадри чиққанидан кейин бошланади. Шунинг учун бу кадрни навбатдаги 12 та кадрга статик қилиб белгиланган. (анимацияни чиқариш тезлиги секундига 12 та кадр) Анимацияни овоз билан бир вақтда тугашини ташкил қилинган.



12.19-расм. Анимацияга мисол.

Ролик тайёр бўлганидан кейин, уни сақлаб қўйиш лозим. Бу ишни одатдаги усул билан, яъни **File** менюсидан **Save** буйруғини танлаб амалга оширилади.

Файлни Macromedia Flash форматидан AVI-га ўтказиш учун **File** менюсидан **Export Movie** буйруғи танланади ва файлнинг номи кўрсатилади. Сўнгра экранда пайдо бўлган диалог ойнасидан **Export Windows AVI** тугмаси чертилади, (12.20-расм.) кадрларнинг ўлчами белгиланади, (Width ва Height майдонлари), **Video Format** рўйхатидан роликнинг видеоқисми сақланадиган формат кўрсатилади, **Sound Format** — майдонидан овоз формати танланади.

12-расм. Export Windows AVI диалог ойнаси

Агар **Compress video** ўчиргичи ўрнатилган бўлса, у ҳолда **ОК** тугмаси чертилганда экранда видеотасвирларни қисишнинг стандарт методларидан бирини танлаб олишга имкон берадиган диалог ойнаси пайдо бўлади. Видео ва овоз форматларини танлаганда овоз ва видеотасвирларнинг ёзуви сифатига қанча кўп эътибор берилса, яратилган AVI-файл компьютер хотирасидан шунча кўп жой эгаллайди. Шуни ёдда тутиш керакки, сифатга қўйиладиган катта талаблар ҳир доим ҳам ўзини оқлайвермайди.

менюга

мундарижага қайтиш

13-БОБ. РЕКУРСИЯ

13.1. Рекурсия тушунчаси

Барон Мюнхаузен ҳақидаги эртакларни эсга олайлик. У киши кунлардан бирида ўрмонда асал емоқчи бўлиб, катта бир дарахт ковагидаги асалари уясига киради. Азбаройи асалдан кўп еганидан қорни шишиб кетиб, дарахт каваги тешигидан чиқа олмай, коваккa тиқилиб қолади. Ақлли ва уддабуррон фон Мюнхаузен бу ҳолда ўзини йўқотиб қўймай, югуриб уйига боради ва аррани олиб келиб, дарахтни арралайди ва ўзини дарахт каваги исканжасидан қутқаради. Бошқа бир эпизодда Барон Мюнхаузен ловия экади. Ловия жуда тез ўсиб, ойга етиб боради. Мюнхаузен ловия поясига тирмашиб, ойга чиқади. Ойни маълум бир муддат айланганидан кейин орқага қайтиб, яна ловия поясига осилиб, туша бошлайди.

Тахминан ер билан ойнинг ўртасига келганда, қараса ловиянинг пастки қисми қуриб қолибди. Шунда барон Мюнхаузен хеч иккиланиб ўтирмай, ўзидан юқорида турган ловияни кесиб, ўзидан пастда турган қисмига улайди ва омон-эсон ерга тушиб олади. Бир қараганда кулгили ва ишониб бўлмайдиган бу воқеалар бизга рекурсия тушунчаси маъносини очиб беришга ёрдам қилади.

Рекурсияни қуйидагича изоҳлаш мумкин. Ечилмаган масалани ечиш учун шу масаланинг ўзига мурожаат қилинади. Бошқача айтганда, қўйилган масалани бир ҳил бошланғич маълумотлар учун ечиш мақсадида шу масаланинг ўзига, фақат бошқа бошланғич маълумотлар учун мурожаат қилинади. Агар кейинги ҳолатдаги масаланинг ечими топилса, дастлабки масаланинг ҳам ечимини топиш мумкин бўлади.

Агар дастур ўзини-ўзи процедура ёки функция сифатида фойдаланадиган бўлса, бундай дастурларни рекурсив дастур дейилади. Рекурсив дастурлар икки турга бўлинади:
а) Тўғри рекурсия. Бунда масала ўзига-ўзи мурожаат қилади.
б) Ёндош рекурсия. Бунда 1-масала 2- масалага, 2-масала эса 1-масалага мурожаат қилади.

Рекурсив дастур ёзиш учун қуйидаги икки ҳоолат аниқланган бўлиш керак.
1) реккурент муносабат;
2) шу муносабат учун бошланғич ҳолат аниқланган бўлинши шарт.

Реккурент муносабат деганда бирор жараённинг N ва N-1 қадамларни боғловчи муносабатлар тушунилади. Масалан, N!=N(N-1)! формулани N! учун реккурент муносабат деб қараш мумкин. Бу муносабат учун бошланғич ҳолат бўлиб, 1!=1 хизмат қилади. Бу маълумотларни ҳисобга олсак, факториални ҳисоблаш масаласи учун реккурент муносабатлар қуйидагича бўлади:

$$N!=\begin{cases} N\cdot(N-1)!, & агар\ N>1 \\ 1, & агар\ N=1 \end{cases}$$

Кўриниб турибдики, N! ни ҳисоблаш учун аввал (N-1)! ни ҳисоблаб топиш керак. Лекин, (N-1)! нинг ҳам қиймати биз учун номаълум. Аммо, биз биламизки, (N-l)!=(N-2)! (N-1). Демак, натижа (N-2)! га боғлиқ бўлмоқда. Шунинг учун уни топишга харакат қилинади. (N-2)! эса (N-3)!(N-2) га тенг ва ҳоказо. Шундай қилиб, N!=N·(N-1)·(N-2)...2·1 га тенг экан. Кўриниб турибдики, N! ни ҳисоблаш алгоритми ўзининг ичига ўзи "чўкиб" бормоқда. Бу жараён бошланғич ҳолат содир бўлгунга, яъни 1! гача давом этади. Шундан кейин, "чўкиш" жараёни тўхтайди, 1!=1 эканлиги ҳақида кўрсатма олган ЭҲМ энди юқорига қараб "сузиб" чиқиш босқичини бажаради. Яъни,

$$2!=1\cdot2=2;\ 3!=1\cdot2\cdot3=6$$

ва ҳоказо. Бу ҳолат то N! ҳисобланмагунча давом этаверади. Шу жараён дастурини ёзиш учун қуйидагича мулоҳаза юритилади.

Агар N факториални ҳисоблаш учун функция яратилса, бу функция қийматини топиш учун шу функциянинг ўзига, фақат N-1 бўлган ҳол учун мурожаат қилинади, яъни функция ўзини ўзи ёрдамчи функция сифатида фойдаланади.

13.1-листингда шу масалани ҳал қилиш учун ёзилган рекурсив дастурнинг матни келтирилган.

13.1-листинг. Рекурсив функция

```
function factorial(n: integer): integer;
begin
if n <> 1
then factorial:= n * factorial(n-1)
// функция ўзини ўзи чақиряпти
else factorial := 1; // бошланғич ҳолатга келинди
end;
```

N=4 бўлган ҳол учун рекурсив жараённинг "чўкиш" ва "сузиб чиқиш" жараёни қуйидаги жадвалда берилган:

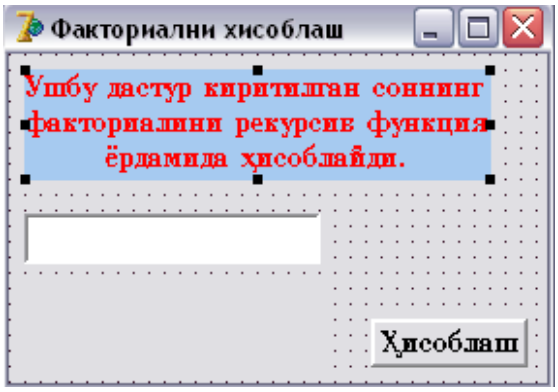
N нинг қийматлари	4 4 3 2 1	якуний рекурсиядан чиқишдаги оралиқ қийматлар
N=1 шарти натижаси	йўқ йўқ йўқ ҳа	
	fak(3)*4 fak(2)*3 fak(1)*2	fac(4) := fac *4=24 fac(3) := fac *3=6 fac(2) := fac *2=2

	1	fac(1) := 1

Демак, ЭҲМ N = 4 бўлган ҳол учун 24 натижани беради.

Агар юқоридаги дастурга диққат билан қараган бўлсангиз, функция ўзини ўзи параметрининг қиймати 1 га тенг бўлмаган ҳолларда чақиряпти. Агар бу параметрнинг қиймати 1 га тенг бўлса, функция ўзининг қийматини ҳсиоблайди, шу билан алгоритмни "чўкиш" жараёни тугаб, "сузиб чиқиш" жараёни бошланади.

Факториални рекурсив функция ёдрамида ҳисоблаш дастурининг диалог ойнаси 13.1-расмда, дастурининг тўла матни эса 13.2-листингда келтирилмоқда.



13.1-расм. Факториални ҳисоблаш дастурининг ойнаси

13.2-листинг. Рекурсив функциядан фойдаланиш.

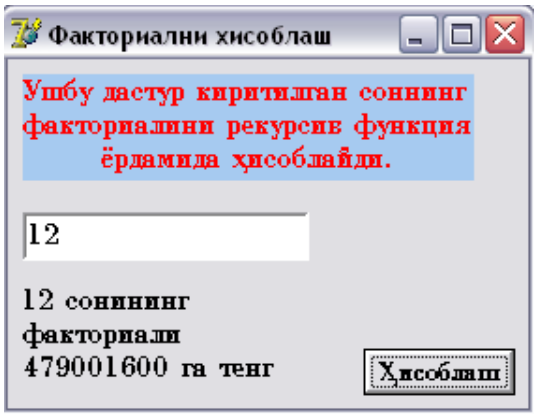
```
unit factor_;
interface
uses Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,   StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Button1: TButton;
    Label2: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
//рекурсив функция

function factorial(n: integer): integer;
begin
  if n > 1
    then factorial := n * factorial(n-1)//функция ўзини-ўзи чақиряпти
    else factorial := 1; //факториал 1 нинг қиймати 1 га тенг
end;

procedure TForm1.Button1Click(Sender: TObject);
var
   k:integer;//факториали ҳисобланаётган сон
   f:integer;//k факториалнинг қиймати
begin
   k := StrToInt(Edit1.Text);
   f := factorial(k);
   label2.caption := Edit1.Text + ' сонининг факториали ' +
```

```
            IntToStr(f)+'  га тенг ';
end;
end.
```

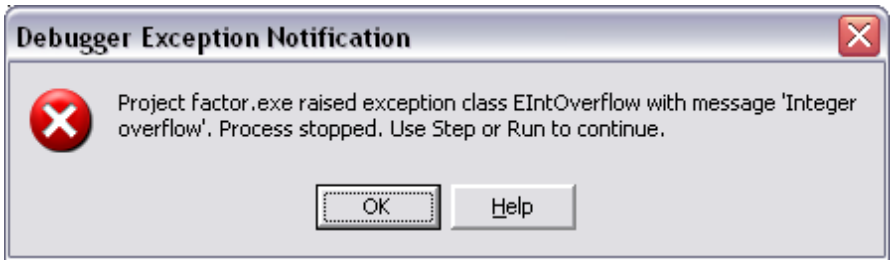
Дастурни ишга тушириш

13.2-расмда юқоридаги дастурнинг турли бошланғич маълумотлоар учун иккита диалог ойнаси келтирилган. 12.2а-расмда берилган факториални ҳисоблаш дастурининг натижаси кутилган сонга мос келади. 13.2б-расмда берилган натижа эса бундай эмас. Қарангки, 44 сонининг факториалини ҳисоблашда бажариш вақтида ҳатолик юзага келиб қолди. Бунинг сабаби шуки, 44! сони жуда катта сон ва Integer типидаги сонлар диапазонидан четга чиқиб кетади.

Delphi тили бажариладиган дастурларда ўзгарувчиларнинг



13.2а-расм. N=12 бўлган ҳол учун дастурнинг диалог ойнаси



13.2а-расм. N=44 бўлган ҳол учун дастурнинг диалог ойнаси

қийматларини белгиланган диапазондан четга чиқиб кетишини назорат қилиши мумкин. Бунинг учун **Project Options** (13.3-расм) диалог ойнасининг **Compiler** тугмаси чертиб, экранда пайдо бўлган ойнадан **Runtime errors** (бажариш вақтидаги ҳатоликлар) гуруҳидаги **Overflow checking** (тўлиб кетишни назорат қилиш) байроқчасини тиклаш лозим.

менюга

13.2. Файлларни қидириш

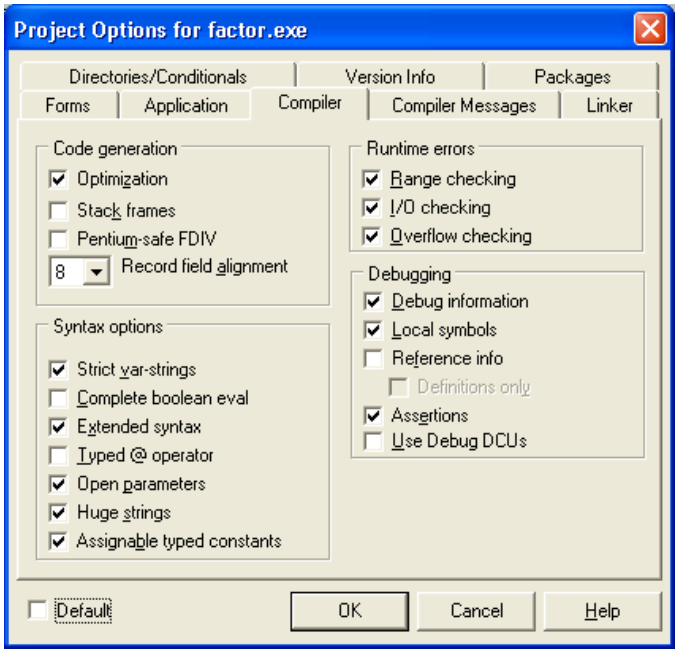
Рекурсиядан фойдаланишга навбатдаги мисол қилиб, берилган файлни компьютерда бор ёки йўқлигини текшириш масаласини олишимиз мумкин. Масалан, фойдаланувчи кўрсатган каталог ва унинг барча ички каталогларида жойлашган барча .bmp кенгайтмали файлларнинг рўйхатини олиш талаб қилинган бўлсин.

Бу масаланинг алгоритмини қуйидагича ифодалаш мумкин:

1. Қидириш шартини қаноатлантирувчи барча файллар рўйхатини чиқарилсин.

2. Агар каталогнинг ички осткаталоглари мавжуд бўлса, кўрсатилган файл улардан ҳам қидирилсин.

Бу алгоритм (блок-схемаси 13.4-расмда келтирилган) рекурсив ҳисобланади. Белгиланган файлни осткаталоглардан ҳам

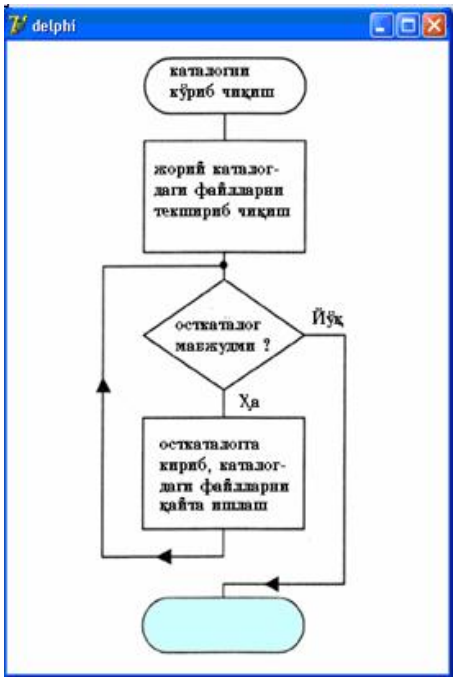


13.3-расм. Project Options диалог ойнаси **Compiler** буйруғининг ойнаси

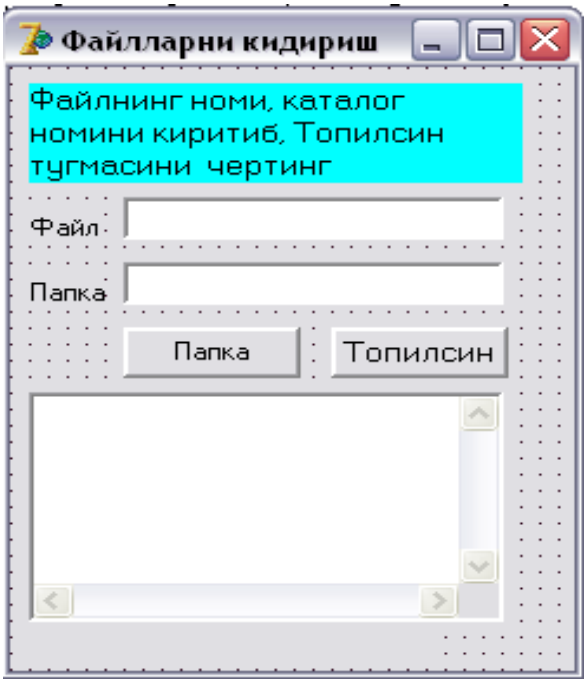
қидириш лозим. Бунинг учун қидириш процедураси ўзига мурожаат қилади.

Дастурнинг диалог ойнасининг кўриниши 13.5-расмда, матни эса 13.3-листингда берилган.

Файл (Edit1) майдони қидилаётган файлнинг номи ёки ниқобини (бир ҳил типдаги файлларни қидириш учун) киритиш учун фойдаланилади. Файл қидириладиган каталог номини бевосита **Папка** майдонига киритиш, ёки стандарт **Обзор папок** диалог ойнасидан танланиши мумкин. **(13.**6-расм.) Бу ойнани экранга Selectdirectory функцияси ёрдамида чақирилади. Эътибор бериш керакки, **Обзор папок** диалог ойнасида фойдаланилаётган файл номи Selectdirectory функциясига WhideChar сатри кўринишидаги қиймат сифатида берилиши лозим. Оддий сатрни WhideChar сатрига айлантириш учун StringToWhideChar функцияси ёрдам беради.

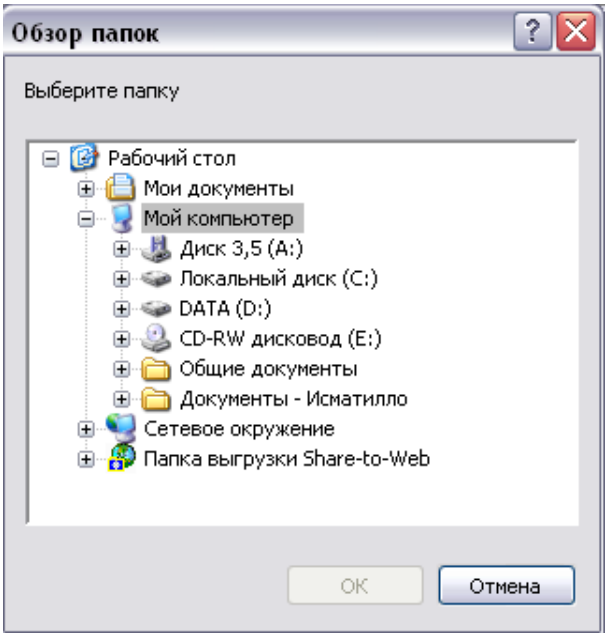


13.4-расм. Файлларни қидириш нинг **р**екурсив алгоритми



13.5-расм. Дастурнинг диалог ойнаси

Обзор папок диалог ойнаси **Папка** тугмаси чертилганда экранга чиқарилади.



13.6-расм. **Обзор папок** диалог ойнаси

Ушбу дастурда асосий масалани Find рекурсив функцияси ҳал қилади. Find функцияси ягона параметр-searchRec структурасига эга. Бу структурадан FindFirst ва FindNext функциялари мос равишда қидириш шартини қаноатлантирувчи биринчи ва навбатдаги файлларни қидириб топиш учун фойдаланади. Қидирув олиб борилаётган каталогга, унинг ост каталогларини дастур қандай кўриб чиқаётганлигига алоҳида эътибор беринг. Агар жорий каталог ўзак каталог бўлмаса, бошқа каталоглардан ташқари, яна иккита каталогга эга: .. ҳамда . каталоглари. Улар олдинги даражали каталоглар ҳисобланади ва дастур ёрдамида кўриб чиқилмайди. Чунки, бу каталоглар ота каталогга чиқишни англатади. Бу ҳолни ҳисобга олинмаса, дастур циклга тушиб қолади.

13.3-листинг. Файлларни қидириш дастури.

```
//кўрсатилган каталог ва унингосткаталогидан қидириш
// Find рекурсив процедурасидан фойдаланилади
 unit FindFile_;
 interface
 uses   Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,   Dialogs, StdCtrls,  FileCtrl;
 type
   TForm1 = class(TForm)
     Edit1: TEdit;          //нима қидирилади
     Edit2: TEdit;          //қаердан
     Memo1: TMemo;     //қидириш натижаси
     Button1: TButton;  //ТОПИЛСИН тугмаси
     Button2: TButton;  //ПАПКА тугмаси
     Label1: TLabel;
     Label2: TLabel;
     Label3: TLabel;
     Label4: TLabel;
     procedure Button1Click(Sender: TObject);
     procedure Button2Click(Sender: TObject);
   private
     { Private declarations }
   public
     { Public declarations }
   end;
 var
   Form1: TForm1;
 implementation
 {$R *.dfm}
 var
    FileName: string; //қидириладиган файлнинг номи ёки ниқоби
```

```
   cDir: string;
   n: integer;          // қидириш шартини қаноатлантирадиган файллар сони
//жорий каталогдан қидириш
 procedure Find;
 var
   SearchRec: TSearchRec;// файл ёки каталог ҳақида маълумот
 begin
   GetDir(0,cDir);// жорий каталог номини олиш
   if cDir[length(cDir)] <> '' then cDir := cDir+'';
   if FindFirst(FileName, faArchive,SearchRec) = 0 then
      repeat
        if (SearchRec.Attr and faAnyFile) = SearchRec.Attr then
          begin
            Form1.Memo1.Lines.Add(cDir +  SearchRec.Name);
            n := n + 1;
          end;
       until FindNext(SearchRec) <> 0;
      // жорий каталогнинг ость каталогларини қайта ишлаш
      if FindFirst('*', faDirectory, SearchRec) = 0 then
        repeat
         if (SearchRec.Attr and faDirectory) = SearchRec.Attr then
           begin
           // булар ҳам каталоглар, аммо уларга кириш керак эмас
            if SearchRec.Name[1] <> '.' then
               begin
                 ChDir(SearchRec.Name); // каталогга кириш
                  Find;               //  осткаталогдан қидириш
                  ChDir('..');  // каталогдан чиқиш
                end;
             end;
        until FindNext(SearchRec) <> 0;
 end;
//фойдаланувчи танлаган каталог
 function GetPath(mes: string):string;
 var
  Root: string;     // ўзак каталог
  pwRoot : PWideChar;
  Dir: string;
 begin
  Root := '';// ўзак каталог-РАБОЧИЙ СТОЛ папкаси
  GetMem(pwRoot, (Length(Root) + 1) * 2);
  pwRoot := StringToWideChar(Root,pwRoot,MAX_PATH*2);
  if SelectDirectory(mes, pwRoot, Dir)
     then
        if length(Dir) + 2 // фойдаланувчи ўзак каталогни танлаган
           then GetPath := Dir + ''
           else GetPath := Dir
     else
        GetPath := '';
 end;
// ТОПИЛСИН тугмасни чертиш
 procedure TForm1.Button1Click(Sender: TObject);
 begin
       Memo1.Clear;          // Memo1 майдонини тозалаш
       Label4.Caption := '';
       FileName := Edit1.Text; // нимани қидирилади
       cDir := Edit2.Text;     // қаердан қидирилади
```

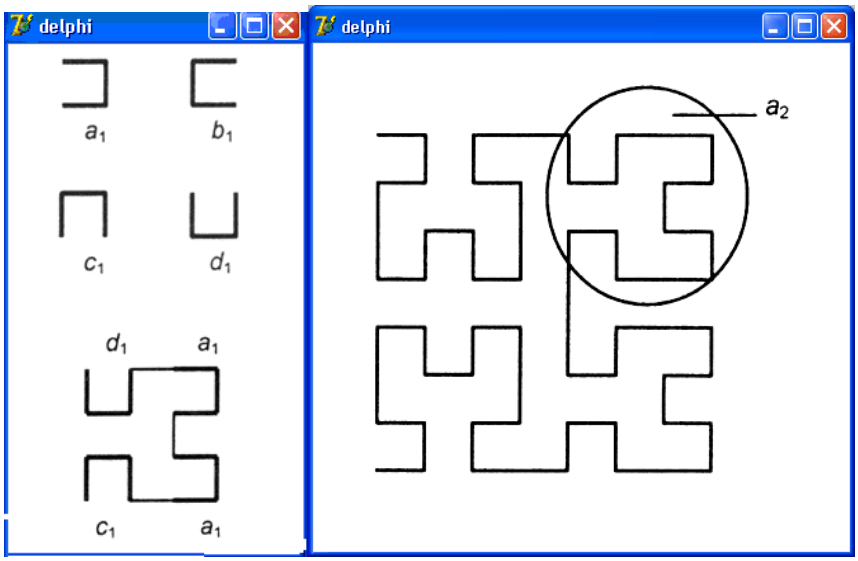
```
        n := 0;                    //Топилган файллар сони
        ChDir(cDir);       //қидириш бошланган каталогга кириш
        Find;              //қидиришни бошлаш
   if n = 0 then
   ShowMessage('қидириш шартини қаноатлантирадиган файллар йўқ. ')
   else Label4.Caption := 'Тоиплган папкалар сони :' + IntToStr(n);
end;
//ПАПКА тугмасини чертиш
procedure TForm1.Button2Click(Sender: TObject);
var
  Path: string;
begin
  Path := GetPath('папкани танланг');
  if Path <> ''
    then Edit2.Text := Path;
end;
end.
```

Дастурни ишга тушириш

менюга

13.3. Гильберт эгри

Қуйидаги дастур Гильберт эгри чизиғини чизиш учун мўлжалланган. 13.7-расмда 1, 2 ва 3-тартибли Гильберт эгри чизиқлари берилган.



13.7-расм. 1, 2 ва 3-тартибли Гильберт эгри чизиқлари

13.8-расм. Бешинчи тартибли Гильбер эгри чизиғи

Агар диққат билан қаралса, 2-тартибли Гильберт эгри чизиғи тўртта 1-тартибли Гильберт эгри чизиқларини тўғри чизиқ билан бирлаштиришдан, 3-тартибли Гильберт эгри чизиғи эса тўртта 2-тартибли Гильберт эгри чизиқларини бирлаштиришдан ҳосил бўлади. Шундай қилиб, Гильберт эгри чизиқларини чизиш алгоритми рекурсив бўлади.

5-тартибли Гильберт эгри чизиқлари жойлашган диалог ойнасининг кўриниши 13.8-расмда, дастурнинг матни эса 13.4-листингда келтирилган.

13.4-листинг. Гильберт эгри чизиқлари

```
unit gilbert_;
interface
uses   Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,   Dialogs, StdCtrls, ComCtrls;
type
  TForm1 = class(TForm)
    procedure FormPaint(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
var
  p: integer = 5; //эгри чизиқнинг тартиби
  u: integer = 7; //штрих узунлиги
{ Гильберт эгри чизиғи бирлаштирилган тўртта a, b, c ва d элементлардан иборат.  Ҳар бир элементни мос процедура чизади. }
procedure a(i:integer; canvas: TCanvas); forward;
procedure b(i:integer; canvas: TCanvas); forward;
procedure c(i:integer; canvas: TCanvas); forward;
procedure d(i:integer; canvas: TCanvas); forward;
//эгри чизиқ элементлари
procedure a(i: integer; canvas: TCanvas);
  begin
    if i > 0 then begin
  d(i-1, canvas); canvas.LineTo(canvas.PenPos.X+u,canvas.PenPos.Y);
  a(i-1, canvas); canvas.LineTo(canvas.PenPos.X,canvas.PenPos.Y+u);
  a(i-1, canvas); canvas.LineTo(canvas.PenPos.X-u,canvas.PenPos.Y);
  c(i-1, canvas);
```

```
    end;
  end;
procedure b(i: integer; canvas: TCanvas);
  begin
   if i > 0 then
   begin
  c(i-1, canvas); canvas.LineTo(canvas.PenPos.X-u,canvas.PenPos.Y);
  b(i-1, canvas); canvas.LineTo(canvas.PenPos.X,canvas.PenPos.Y-u);
  b(i-1, canvas); canvas.LineTo(canvas.PenPos.X+u,canvas.PenPos.Y);
  d(i-1, canvas);
   end;
  end;
procedure c(i: integer; canvas: TCanvas);
  begin
   if i > 0 then
   begin
  b(i-1, canvas); canvas.LineTo(canvas.PenPos.X,canvas.PenPos.Y-u);
  c(i-1, canvas); canvas.LineTo(canvas.PenPos.X-u,canvas.PenPos.Y);
  c(i-1, canvas); canvas.LineTo(canvas.PenPos.X,canvas.PenPos.Y+u);
  a(i-1, canvas);
   end;
  end;
procedure d(i: integer; canvas: TCanvas);
  begin
   if i > 0 then
   begin
  a(i-1, canvas); canvas.LineTo(canvas.PenPos.X,canvas.PenPos.Y+u);
  d(i-1, canvas); canvas.LineTo(canvas.PenPos.X+u,canvas.PenPos.Y);
  d(i-1, canvas); canvas.LineTo(canvas.PenPos.X,canvas.PenPos.Y-u);
  b(i-1, canvas);
  end;
  end;
procedure TForm1.FormPaint(Sender: TObject);
  begin
    Form1.Canvas.MoveTo(u,u);
    a(5,Form1.Canvas);//Гильберт эгри чизиғини чизиш
  end;
end.
```

Дастурни ишга тушириш

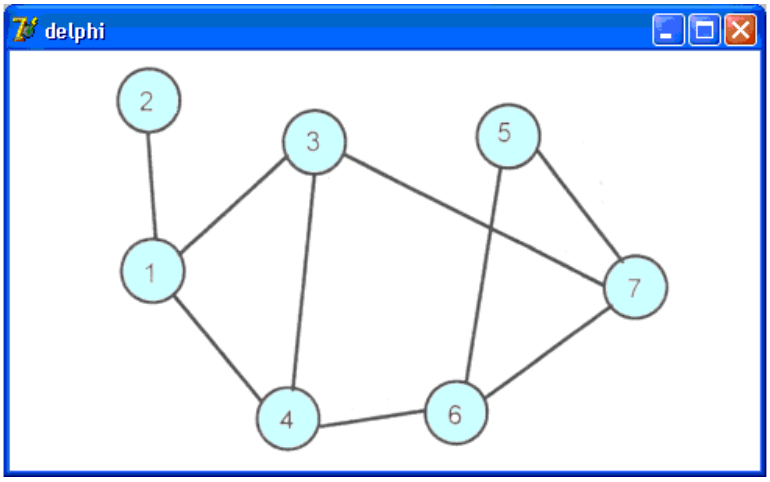
Дастурнинг ўзига ҳос томонларига эътибор беринг. a элементни чизаётган процедура ўзидан ташқари, d ва b процедураларини (уларнинг матни a процедурадан кейин келтирилган) чақиради. Компилятор ҳатолик ҳақида ахборот бермаслиги учун дастур матнига процедура forward хизматчи сўзи билан бирга эълон қилинган. Бу ҳол "эълонлар кейинроқ берилади" демакдир. Шундай қилиб, a процедурасини компиляция қилиш жараёнида компилятор b ва d номлари процедураларни англатишини "тушунади".

менюга

13.4. Йўл қидириш масаласи

Бир-бири билан бир нечта йўллар орқали боғланган бир нечта шаҳарлар берилган бўлсин. Айрим йўлларнинг боши берк бўлиши ҳам мумкин. Бир шаҳардан иккинчисига олиб борувчи барча маршрутларни топиш масаласи қўйилган бўлсин.

Мавжуд бўлган барча йўлларнинг ҳаритаси граф (шаҳарларни англатувчи учлар ва йўлларни билдирувчи қирраларнинг тўплами) ёрдамида берилган бўлсин. (13.9-расм)



13.9-расм. Граф кўринишидаги йўллар харитаси

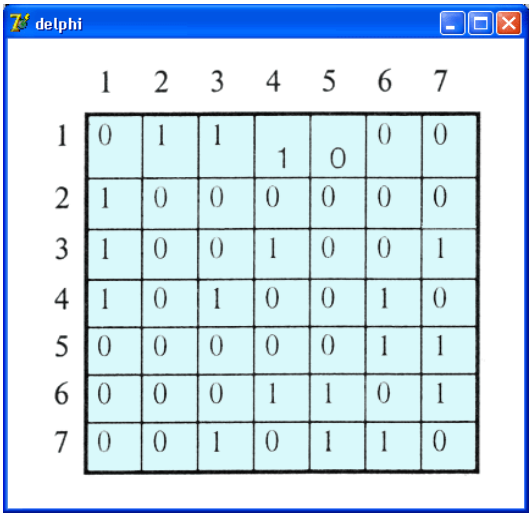
Қидириш жараёнини қадамлар кетма-кетлиги сифатида ифодалаш мумкин. Ҳар бир қадамда жорий нуқтадан туриб, бирор шарт остида ўтиш мумкин бўлган иккинчи нуқта танланади. Агар навбатдаги нуқта берилган нуқта билан устма-уст тушса, у ҳолда масала ечилди. Акс ҳолда яна бир қадам қўйиш керак. Жорий нуқтани бошқа бир нечта нуқталар билан бирлаштириш имкоииияти бўлгани учун танлашнинг бирор шартини танлаш лозим. Энг содда ҳолда энг кичик номерли нуқтани олиш мумкин.

Фараз қилайлик, 1- нуқтадан 5-чига ўтиш талаб қилинган бўлсин. Шартга кўра дастлаб 2-нуқтани танлаймиз. Кейинги қадамда 2-нуқтанинг боши берк эканлигини аниқлаймиз. Шунинг учун 1-нуқтага қайтиб, 3-нуқта томонга юрамиз. Ундан 4-нуқтага ўтамиз. 4-дан 6-га, ундан эса 5-нуқтага борамиз. Мумкин бўлган йўллардан бирини топдик. Шундан кейин 6-га қайтамиз. Ундан туриб, 5-дан фарқли яна бошқа йўлнинг мавжудлигини текширамиз. Бунинг иложи бўлганлиги учун, 7-нуқтага юрамиз. 7-дан эса 5-га ўтиш мумкин. Яна битта йўл топилди. Шундай қилиб, йўлларни қидириш жараёни олдинга ва орқага юришлардан иборат бўлади. Қидириш масаласи харакат бошланган жойдан бошқа борадиган бирорта ҳам йўл қолмаганидан сўнг тугатилади.

Қидириш алгоритми рекурсив характерга эга: янги қадамни қўйиш учун нуқта танланади ва қадам қўйилади. Бу жараён то мақсадимизга эришмагунимизча, давом этади.

Барча йўлларни қидириш масаласи янги нуқтани (шаҳарни) танлаш ва йўлнинг қолган қисмини қидириш масаласига айланди. Бу ерда рекурсия кўриниб турибди.

Графни икки ўлчовли массив деб қараймиз. Уни map деб атайлик. map[i,j] - бу i ва j шаҳарлар ўртасидаги масофа. Агар улар орасида йўл бўлмаса, бу масофа 0, акс ҳолда 1 га тенг. Юқоридаги граф учун Map массиви қуйидагича ёзилади (13.10-расм)



	1	2	3	4	5	6	7
1	0	1	1	1	0	0	0
2	1	0	0	0	0	0	0
3	1	0	0	1	0	0	1
4	1	0	1	0	0	1	0
5	0	0	0	0	0	1	1
6	0	0	0	1	1	0	1
7	0	0	1	0	1	1	0

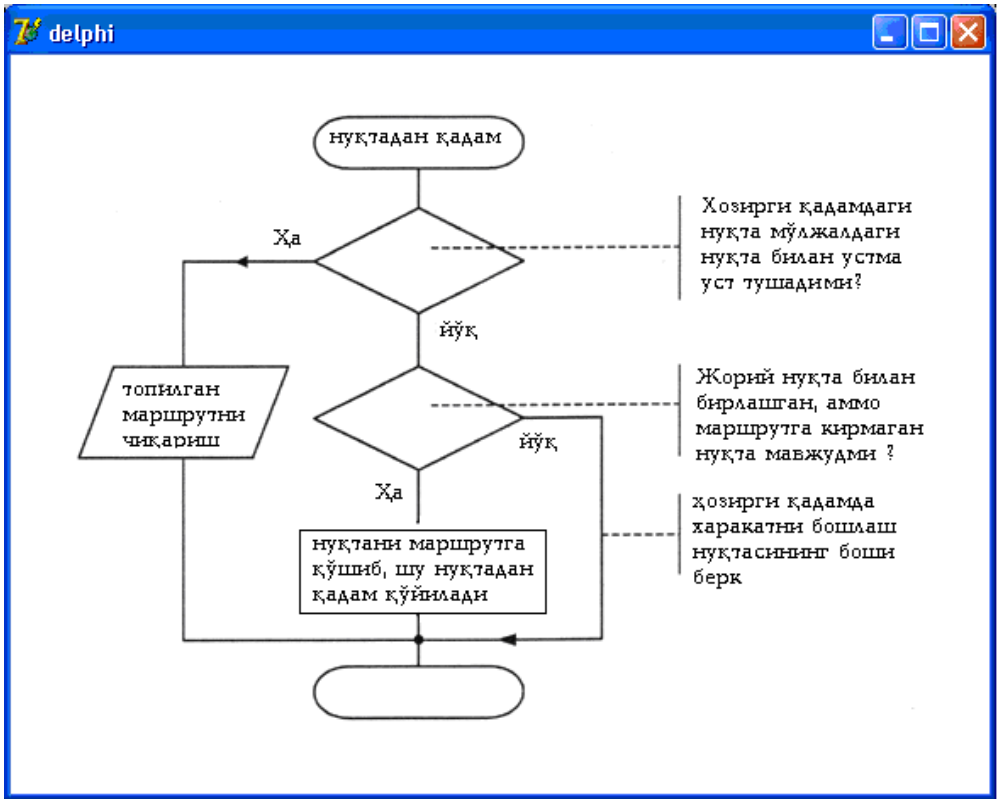
Рис. 12.10. Map массиви

Map масссивидан ташқари бизга яна road (йўл) ва incl(include – киритилсин сўзидан олинган) массивлари ҳам керак бўлади. Road массивига биз ўтилган шаҳарларни ёзиб борасиз. Охирги нуқтага борилганда бу массивга барча ўтилган нуқта-шаҳарлар, яъни маршрутлар ёзиб қўйилади. Incl[i] массивига

эса i-номерли нуқта маршрутга кирган бўлса – true ёзамиз. Бу бизга бир марта борилган нуқтага яна қайтиб бормаслик учун керак.

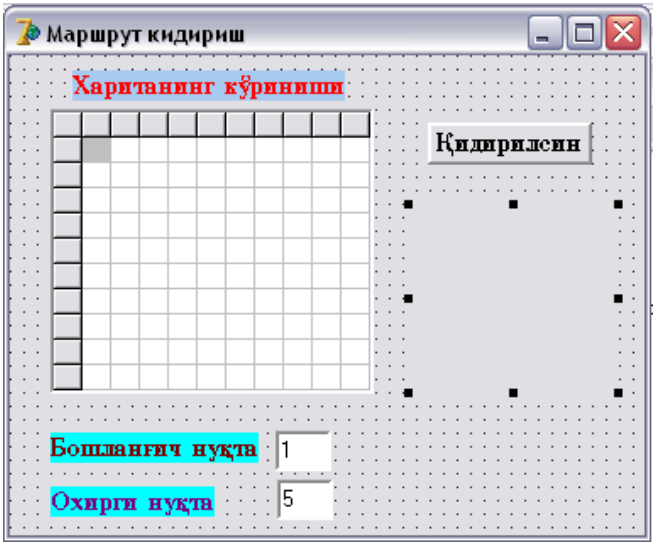
Биз рекурсив процедурадан фойдаланганимиз учун, рекурсив жараённинг тугатишига алоҳида эътибор қаратиш лозим. Процедура жорий нуқта мўлжалдаги нуқта билан устма-уст тушгунча ўзини ўзи чақираверади.

13.11-расмда навбатдаги нуқтани танлаш процедураси алгритмининг блок-схемаси, 13.12-расмда эса дастурнинг диалог ойнаси келтирилган.



13.11-расм. Маршрут қадамини танлаш процедурасининг блок-схемаси

Графни ўз ичига олган массивни киритиш учун stringGridl компонентасидан фойдаланилади. (Унинг қийматлари 13.1-жадвалда берилган), натижани (топилган маршрутни) чиқариш учун - Label1 майдони киритилади. Бошланғич ва охирги нуқталар Edit1 ва Edit2 таҳрирлаш майдонларига киритилади. Қидириш жараёни **Қидирилсин** (Buttonl) тугмаси чертилганда бошланади. Label2, Label3 ва Label4 майдонлар изоҳлар учун мўлжалланган.



13.12-расм. Маршрут қидириш дастурининг диалог ойнаси

StringGrid1 компонентасининг ҳусусият қийматлари **13.1-жадвал.**

Ҳусусияти	**Қиймати**

Name	StringGrid1
ColCount	11
RowCount	11
FixedCols	1
FixedRows	1
Options . goEditing	TRUE
DefaultColWidth	16
DefaultRowHeight	14

Бу масала дастурининг матни 13.5- листингда келтирилган.

13.5-листинг. Маршрутни қидириш.

```
unit road_;
interface
uses   Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,   StdCtrls, Grids;
type
  TForm1 = class(TForm)
    StringGrid1: TStringGrid;
      Edit1: TEdit;
      Edit2: TEdit;
      Label1: TLabel;
      Label2: TLabel;
      Label3: TLabel;
      Button1: TButton;
      Label4: TLabel;
      procedure FormActivate(Sender: TObject);
      procedure Button1Click(Sender: TObject);
    private
      { Private declarations }
    public
      { Public declarations }
    end;
  var
    Form1: TForm1;
  implementation
  {$R *.DFM}
  procedure TForm1.FormActivate(Sender: TObject);
  var
     i:integer;
  begin
    // сатрларни номерлаш
     for I := 1 to 10 do
        StringGrid1.Cells[0,i] := IntToStr(i);
    // устунларни номерлаш
     for I := 1 to 10 do
        StringGrid1.Cells[i,0] := IntToStr(i);
    // харити
     StringGrid1.Cells[1,2] := '1';
     StringGrid1.Cells[2,1] := '1';
     StringGrid1.Cells[1,3] := '1';
     StringGrid1.Cells[3,1] := '1';
     StringGrid1.Cells[1,4] := '1';
     StringGrid1.Cells[4,1] := '1';
```

```
  StringGrid1.Cells[3,7] := '1';
  StringGrid1.Cells[7,3] := '1';
  StringGrid1.Cells[4,6] := '1';
  StringGrid1.Cells[6,4] := '1';
  StringGrid1.Cells[5,6] := '1';
  StringGrid1.Cells[6,5] := '1';
  StringGrid1.Cells[5,7] := '1';
  StringGrid1.Cells[7,5] := '1';
  StringGrid1.Cells[6,7] := '1';
  StringGrid1.Cells[7,6] := '1';
end;
procedure TForm1.Button1Click(Sender: TObject);
const
  N=10;{ граф учларининг сони}
var
  map:array[1..N,1..N]of integer;
  road:array[1..N]of integer;
  incl:array[1..N]of boolean;
  start,finish:integer;          { бошланғич ва охирги нуқталар }
  found:boolean;
  i,j:integer;
procedure step(s, f, p:integer);
var
  c : integer;{ Навбатдаги қадам қўйиладиган нуқта номери}
  I : integer;
begin
  if s = f then
  begin
     { s ва f нуқталар устма-усттушмайди!}
     found := TRUE;
     Label1.caption := Label1.caption + #13 + 'Йўл:';
     for i := 1 to p-1 do
       Label1.caption := Label1.caption + ' ' + ntToStr(road[i]);
  end
  else begin
        { навбатдаги нуқтани танлаймиз }
  for c := 1 to N do
    begin {ҳамма учларни текширамиз }
      if(map[s,c]<> 0)and(NOT incl[c])
 { нуқта жорий нуқта билан бирлашган, аммо маршрутга кирмаган}
       then begin
          road[p] := c;{ учни йўлга қўшамиз }
          incl[c] := TRUE;{ учни маршрутга кирган деб белгилаймиз }
                        step(c, f, p + 1);
                        incl[c] := FALSE;
                road[p] := 0;
            end;
        end;
  end;
end;{ step процедураси тамом}
begin
  Label1.caption := '';
  { массив элементларини аниқлаш }
  for i := 1 to N do road[i] := 0;
  for i := 1 to N do incl[i] := FALSE;
  { массивга элементларни  SrtingGrid.Cells дан киритамиз}
  for i := 1 to N do
     for j := 1 to N do
```

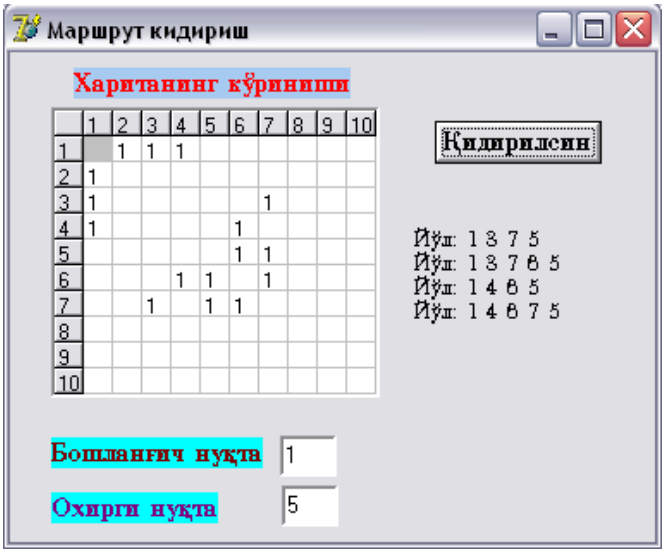
```
        if StringGrid1.Cells[i,j] <> ''
           then map[i,j] := StrToInt(StringGrid1.Cells[i,j])
           else map[i,j] := 0;
  start := StrToInt(Edit1.text);
  finish := StrToInt(Edit2.text);
  road[1] := start;{ нуқтани маршрут киритамиз}
  incl[start] := TRUE;{ уни маршрутга кирган деб белгилаймиз }
 step(start,finish,2);{ маршрутнинг иккинчи нуқтасини аниқлаймиз}
  // ҳеч бўлмаса битта йўл топилганлигини текширамиз
  if not found
     then Label1.caption := 'Берилган қуқталар орасида йўл йўқ !';
end;
end.
```

Дастурни ишга тушириш

Дастур ишга тушганидан сўнг, форманинг фаоллашиши даврида OnActivate ходисасини қайта ишлаш процедураси StringGrid1.cells массивини харитадаги қийматлар билан тўлдиради. Шу процедуранинг ўзи StringGrid1 жадвалининг устун ва сатрларини фиксирланган биринчи устун ва сатр ячейкаларини тўлдирган ҳолда номерлайди.

Маршрутни Tform1.Button1click процедураси қидиради. Бу процедура бошланғич нуқта билан туташтирилган нуқтани топиш учун step процедурасини чақиради. У бошланғич нуқта билан бирлашган биринчи нуқтани танлаб, уни маршрутга қўшганидан кейин ўзини ўзи чақиради. Бунда бошланғич нуқта сифатида берилган нуқта эмас, балки маршрутга ҳозиргина киритилган жорий нуқта олинади.

Келтирилган граф учун дастурнинг берган натижаси 13.13-расмда кўрсатилган.



13.13-расм. Дастурнинг диалог ойнасидаги натижалар

менюга

мундарижага қайтиш

14-БОБ. ДАСТУРДАГИ ҲАТОЛИКЛАР БИЛАН ИШЛАШ

Компиляция жараёнининг муваффақиятли тугаши дастурда ҳали ҳатоликлар йўқ дегани эмас. Дастурнинг тўғри ишлаётганлигига фақат унинг иши натижаларини таҳлил қилгандан кейингина, бошқача айтганда, берилган таст талабларига тўла жавоб бериб, кутилган натижаларни олгандагина ишонч ҳосил қилиш мумкин.

Одатда камдан-кам дастурлар бирданига кутилган натижаларни бера олади. Уларнинг кўпчилиги фақат маълум бир бошланғич қийматлар учун тўғри натижа беради, бошқа бошланғич маълумотлар учун эса нотўғри натижа беради ёки умуман ишламайди. Бу ҳол дастурда алгоритмик ҳатолар мавжудлигидан далолат беради. Биз ушбу бобда ана шу ҳатоликларни қидириш топиш ва бартараф қилиш усуллари билан танишамиз.

14.1. Ҳатоликлар классификацияси

Дастурда мавжуд бўлиши мумкин бўлган ҳатоликларни учта гуруҳга бўлиш мумкин:

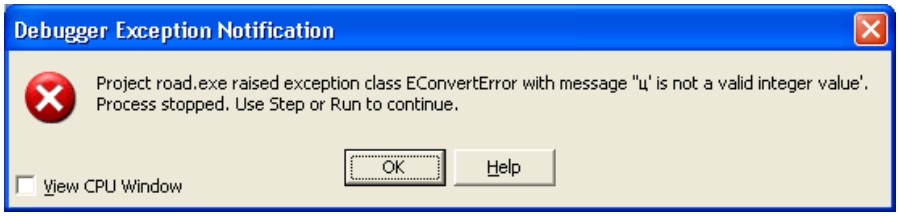
- синтаксик;
- бажариш вақтидаги ҳатоликлар;
- алгоритмик.

Синтактик ҳатоликлар, уларни компиляция вақтидаги ҳатоликлар **(Compile-time error)** деб ҳам аталади, энг осон бартараф қилинадиган ҳатоликлар ҳисобланади. Уларни компилятор қидириб топади. Компиляция қилишда дастур матнини Delphi тилида қабул қилинган қонун-қоидаларга мувофиқ ёзилганлини текширилади. Текшириш давомида компилятор ҳатоликлар мавжудлигини "сезиб" қолса, фойдаланувчига топилган бу ҳатоликлар ҳақида ахборот беради. Дастурчи эса дастур матнига зарур ўзгаришларни киритади ва дастурни қайта компиляция қилади.

Бажариш вақтидаги ҳатоликларни (уларни Delphi да йўқотиш **(exception)** деб ҳам юритилади) ҳам осонгина аниқлаш ва бартараф қилиш мумкин. Бу гуруҳдаги ҳатоликлар одатда, дастурни дастлабки бир неча марта ишга туширилиши ҳамда дастурни тестдан ўтказиш жараёнида аниқланади.

Delphi да туриб, ишга туширилган дастурларда ҳатолик юзага келиб қолса, дастур ўз ишини тўхтатади, Буни Delphi бош саҳифаси биринчи сатридаги қавслар ичида бериладиган **stopped** ахборотидан билиш мумкин. Экранда махсус диалог ойнаси пайдо бўлиб, унга йўл қўйилган ҳатолик ҳамда бу ҳатоликининг характери, типи ҳақидаги ахборот чиқарилади. 14.1-расмда мавжуд бўлмаган файлни очишга нотўғри уриниш ҳақидаги ахборот берилган.

Ҳатолик юзага келганидан кейин, дастурчи дастур ишини тўхтатиши (бунинг учун **Run** менюсидан **Program Reset** буйруғини танлайди) ёки ҳар бир буйруқ натижасини кузатган ҳолда дастурнинг бажарилишини қадам-бақадам давом эттириши (бунинг учун **Run** менюсида **Step** буйруғини танланади**)** мумкин.



14.1-расм. Дастурни Delphi дан ишга туширилгандаги ҳатоликка мисол

Агар дастур Windows дан ишга туширилган бўлса, ҳатоликлар юзага келганда экранда ҳатолик ҳақида ахборот пайдо бўлади, аммо унда ҳатоликнинг типи кўрсатилмайди. (14.2-расм). **ОК** тугмаси босилгандан кейин, агар иложи бўлса ҳатолик мавжуд бўлган дастур ўз ишини давом эттиради.

14.2. Дастур Windows дан ишга туширилгандаги ҳатоликка мисол

Алгоритмик ҳатоликлар билан ишлаш бир оз мураккаброқ. Одатда алгоритмик ҳатоси бўлган дастур матнини компиляция қилинганда, ҳеч қандай муаммо юзага келмайди. Дастур синов тариқасида ишга туширилганда ҳам "дастур ўзини яхши тутади", аммо бу дастур берган натижаларни таҳлил қилинганда, унинг нотўғри ишлаётганлиги билиниб қолади. Алгоритмик ҳатоси бўлган дастур ишини яхшилаш учун алгоритм чуқур таҳлил қилиш, зарур бўлса, уни "қўлда" бажарилишини назорат қилиш лозим.

менюга

14.2. Ҳатоликларни бартараф қилиш ва қайта ишлаш

Одатда дастур ишга туширилганидан кейин, фойдаланувчи айби билан йўл қўйилган ҳатоликлар юзага чиқиб қолиши мумкин. Масалан, фойдаланувчи бошланғич маълумотларни нотўғри киритиши ёки дастурнинг ишлаши учун зарур файлларни ўчириб юборган бўлиши мумкин.

Дастур ишининг бузилиши йўқотиш деб аталади. Бундай ҳатоликларни қайта ишлаш, шунингдек уларга мос ахборотларни экранга чиқариш вазифасини бажарилаётган дастур матнига автоматик тарзда қўшиб қўйиладиган махсус код (дастур) ўз зиммасига олган. Агар зарурат бўлса, ҳатоликларни қайта ишлаш жараёнини Delphi тили бажарилаётган дастурнинг зиммасига юклаш имкониятига ҳам эга.

Йўқотишларни қайта ишлаш буйруғи умумий кўринишда қуйидагича ёзилиши мумкин:

```
try
//ҳатолик юзага келиши мумкин бўлган буйруқ
except//ҳатоликларни бартарафи қилиш бўлимининг бшланиши
on ҲатоликТипи1 do ҚайтаИшлаш1;
on ҲатоликТипи2 do ҚайтаИшлаш2;
...  ...  ...
on ҲатоликТипиN do ҚайтаИшлашN;
else
//қолган ҳолларни қайта ишлаш
end;
```

бу ерда

- **try** — калит сўз бўлиб, ундан кейин бажарилганда ҳатолик юзага келиши мумкин бўлган буйруқ келишини ҳамда бу ҳатоликни қайта ишлаш дастур зиммасига юклатилганлигини англатади;
- **except** — калит сўз бўлиб, ҳатоликларни бартараф қилиш бўлимини бошланганлигини кўрсатади. Бу бўлимдаги кўрсатмалар фақат кўрсатилган буйруқни бажаришда ҳатолик юзага келгандагина бажарилади;
- **on** — калит сўз, ундан кейин ҳатолик типи ва **do** ҳизматчи сўзидан кейин бу ҳатолик типига дастурнинг жавоби белгилаб қўйилади;
- **else** — except бўлимида кўрсатилмаган типдаги ҳатоликлар юзага келганда дастурнинг жавобини кўрсатади.

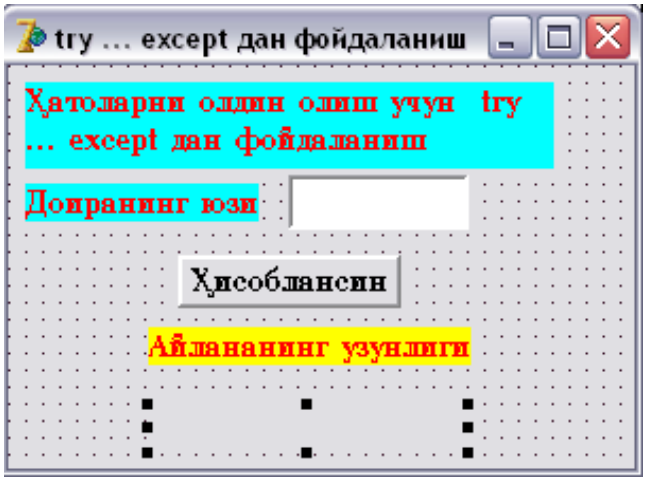
Юқорида таъкидлаб ўтдикки, йўқотишнинг асосий характеристикаси унинг типидан иборат. 14.1-жадвалда энг кўп учрайдиган йўқотишлар ва уларнинг юзага келишига мумкин бўлган сабаблар келтирилган.

Типик йўқотишлар жадвали **14.1-жадвал**

Йўқотиш типи	**Юзага келиши**
EZeroDivide	Бўлиш амалида учрайди, бўлувчи нолга тенг бўлган ҳолда

EConvertError	Алмаштириш вақтида, агар алмашти-рилаётган миқдорни талаб қилинган типга ўтказиб бўлмаса. Кўпинча белгилар сатрини сонга алмаштиришда учрайди.
EFilerError	Файлга мурожаат қилганда. Кўпинча талаб қилинган файл мавжуд бўлма-ганда, дисклардан фойдаланилганда, диск юритувчига диск қўйилмаган ҳолларда учрайди.
EmathError	Математик функцияларнинг аргументлари мумкин бўлган диапазондан четга чиққанда. Масалан, квадрат илдизда манфий сон келганда учрайди.

Қуйидаги дастур (диалог ойнаси 14.3-расмда, матни эса 14.1-листингда келтирилган) try буйруғи ёрдамида ҳатоликларни қайта ишлашга мисол сифатида берилмоқда.



14.3-расм. Айлана узунлигини ҳисоблаш дастурининг диалог ойнаси

Бу дастур доиранинг юзаси S берилган бўлса, шу доирани ўраб турган айлана узунлигини ҳисоблаш учун мўлжалланган.

Маълумки, бу масалани ечиш учун биз дастлаб доира радиусини $r = \sqrt{s / 2\pi}$ формула топамиз. Сўнгра $l = 2\pi r$ формуласи билан айлана узунлигини топамиз. Кўриниб турибдики, бу масала учун дастур ёзганда иккита ҳатолик бўлиши мумкин: S ўрнига манфий сон келиши ёки ҳақиқий сонни нотўғри (унинг бутун ва каср қисми вергул ўрнига нуқта билан ажратилган) киритилган бўлиши мумкин. Ана шу ҳатоликларни дастурда ҳисобга оламиз.

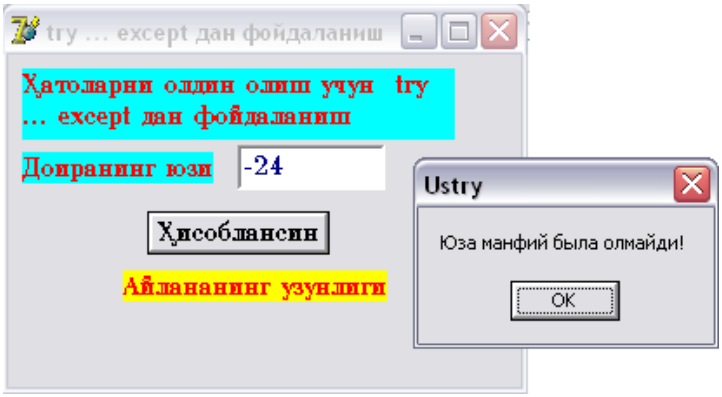
14.1-листинг. Йўқотишларни қайта ишлаш

```
unit UsTry_;
interface
uses Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,   StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label4: TLabel;
    Edit1: TEdit;       //Юза
    Label5: TLabel;     //айлана узунллиги
    Button1: TButton; //Ҳисоблаш тугмаси
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
```

```
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.Button1Click(Sender: TObject);
var
   s: real;//доиранинг юзи
   r: real;//радиус
   l: real;//айланананинг узунлиги
begin
   Label5.Caption := '';
     //хатолик юзага келиши мумкин бўлган вазиятлар
     try
     s := StrToFloat(Edit1.Text);
     r := sqrt(s/(2*pi));
     except
       on EMathError do //квадрат илдиз остида манфий сон
          begin
            ShowMessage(' Юза манфий бўла олмайди!');
            exit;
          end;
       on EConvertError do //сартларни сонга айлантириб бўлмади
          begin
            ShowMessage('Юзани сон билан ёзинг.' +#13 +
                  'Ҳақиқий сонни ёзишда вергулдан фойдаланинг.');
            exit;
          end;
     end;
     L := 2*pi*r;
     Label5.caption :='L= ' + FloatToStr(l);
end;
end.
```

Дастурни ишга тушириш

Келтирилган дастур Windows муҳитида туриб ишга туширилса, ҳамда фойдаланувчи дастурда ҳисобга олган манфий сонни киритса, дастур 13.4-расмдаги жавобни беради.



13.4-расм. Манфий сон учун дастурнинг жавоби

Ҳар икки ҳолдаги ҳатолик юзага келса, бу дастур йўл қўйилган ҳатолик ҳақида ахборотни беради ва ўз ишини тўхтатади.

менюга

14.3. Отладчик

Delphi муҳити дастурчиларга дастурдаги ҳатоларни аниқлаш ва бартараф қилиш учун жуда ҳам кучли восита – отладчикни таклиф қилади. Отладчик дастурчига дастурнинг трассировкаси, (қадам-бақадам бажарилиши) ўзгарувчиларнинг қийматларини кузатиш, назорат қилиш ҳамда чиқарилаётган маълумотларни назорат қилиш каби имкониятларни беради.

Дастурнинг трассировкаси. Дастур ишлаётган вақтда унинг буйруқлари процессорнинг ишлаш тезлигида бажарилади. Шунинг учун дастурчи жорий вақтда қайси буйруқ ва унинг қандай бажарилаётганини назорат қила олмайди. Демак, дастурнинг буйруқлари дастурчи тузган алгоритм бўйича ишлаётганлигини ҳам билиб бўлмайди.

Дастур нотўғри ишлаётган бўлса, унинг ҳақиқий иш тартибини кўриш лозим бўлади. Бунинг учун дастурни трассировка қилишга тўғри келади. Трассировка — бу дастурнинг қадам-бақадам (step-by-step) бажарилиши жараёнидир. Трассировка вақтида дастурчи дастурнинг навбатдаги буйруғини бажариш учун компьютерга кўрсатма беради.

Delphi муҳитида икки ҳил режимдаги трассировкага рухсат берилган: процедурага кирмасдан (Step over) ҳамда процедурага кириб (Trace into) трассировка қилиш. Процедурага кирмасдан трассировка қилиш режимида фақат асосий процедурагина трассиврока қилинади, қисм дастурлар трассировка қилинмайди. Бутун қисм дастур битта буйруқ тарзида трассировка қилинади. Процедурага кириб трассировка қилиш режимида эса ҳам асосий процедуранинг буйруқлари, ҳам қисм дастурларнинг буйруқлари трассировка қилинади.

Трассировкани бошлаш учун **Run** менюсидан **Step over** ёки **Trace into** буйруқларидан бири танланади. Натижада кодлар муҳаррири ойнасида дастурнинг биринчи буйруғи ажратилади. Ажратилган буйруқни бажариш учун **Run** менюсидан **Step over** (<F8> клавишасини босиш) ёки **Trace into** (<F7> тугмасини босиш) буйруғи танланади. Бу буйруқ бажарилганидан сўнг, навбатдаги буйруқ ажратилади ва х.к.

Ихтиёрий вақтда трассировка жараёнини тугатиш ва дастурнинг қолган буйруқларини ҳақиқий тезликда бажариб давом эттириш мумкин. Бунинг учун **Run** менюсидан **Run** буйруғини танлаш лозим.

Зарурат бўлса, трассировкани дастурнинг бирор буйруғидан бошлаш мумкин. Бунинг учун курсорни дастурнинг керакли буйруғига ўрнатиш ва **Run** менюсидан **Run to cursor** буйруғини танлаш ёки <F4> клавишасини босиш мумкин. Сўнгра, <F7> ёки <F8> тугмаларидан бирини босиб, дастурнинг керакли парчасини трассировка қилиш мумкин.

Трассировка вақтида нафақат буйруқларнинг бажарилиши тартибини, балки ўзгарувчиларнинг қийматларини ҳам кузатиб бориш мумкин.

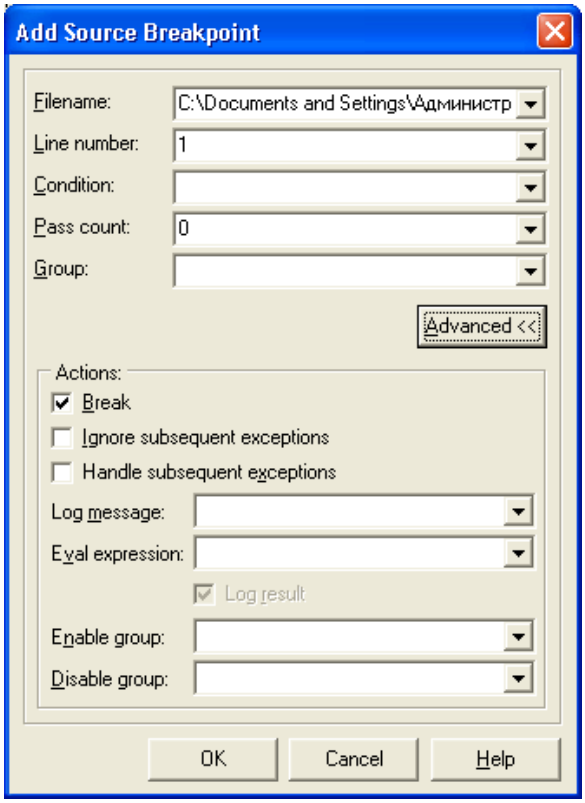
Дастурнинг тўхтатиш нуталари. Ҳатолар билан ишлаш жараёнида дастурларнинг тўхтатиш нуқтаси деган усулдан ҳам кенг фойдаланилади. Бу усулнинг ғояси қуйидагича: Дастурчи дастурнинг айрим буйруқларини белгилаб қўяди (тўхтатиш нуқталарини кўрсатади). Шу буйруққа келганда дастур ўз ишини тўхтатади ва дастурчи трассировка жараёнини бошлаши ёки ўзгарувчиларнинг қийматларини кузатишни бошлаши мумкин.

Тўхтатиш нуқталарини қўшиш. Дастурга тўхтатиш нуқтасини (breakpoint) қўшиш учун **Run** менюсидан **Add Breakpoint** (тўхтатиш нуқтасини қўшиш) буйруғини, сўнгра кейинги даражали ойнадан - **Source Breakpoint** буйруғини танлаши лозим**.**

Натижада **Add Source Breakpoint диалог ойнаси** (14.5-расм) пайдо бўлади. Унда қўшилаётган тўхтатиш нуқтасини ҳақидаги ахборотни кўриш мумкин. **Filename** майдони тўхтатиш нуқтаси қўшиладиган дастур файлининг номини, **Line number** майдони— дастурда тўхтатиш нуқтаси қўйилган сатр номерини англатади. **ОК тугмаси чертилганидан кейин,** тўхтатиш нуқтаси қўйилган сатр қизил нуқта билан белгиланади, ва бошқа рангда ажратиб қўйилади. (14.6-расм).

Тўхтатиш нуқтасини дастурнинг тўхтатиш нуқтаси қўйиладиган сатрини кўрсатувчи кўк рангли нуқтани сичқонча билан чертиб ҳам қўйиш мумкин. (Эътибор берган бўлсангиз, дастур матнида ҳатолар бўлмаса, компилятор дастур буйруқларини кўк ранг билан белгилаб қўяди).

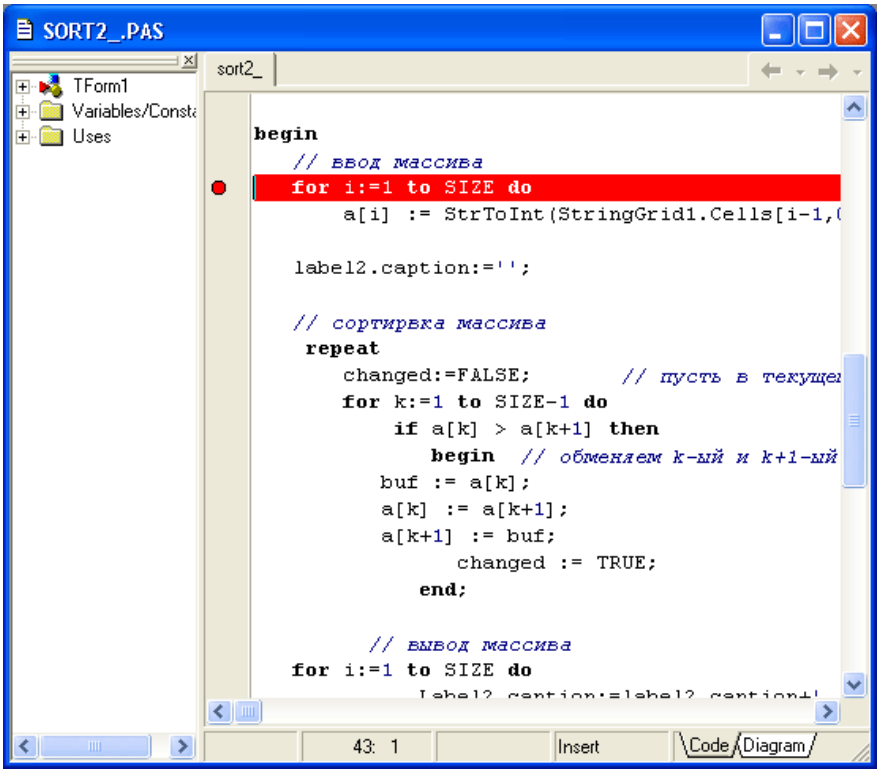
Тўхтатиш нуқтасини бирор шартга боғлаб қўйиш ҳам мумкин. Шу шарт ўринли бўлганда дастур ўз ишини шу нуқтанинг



14.5-расм. Add Source Breakpoint буйруғининг диалог ойнаси

ўзида тўхтатади. (Масалан, ўзгарувчининг қиймати бирор сонга тенг бўлса.) шарт (мантиқий ифода) **Add Source Breakpoint** диалог ойнасининг **Condition** майдонига ёзилади. Агар тўхтатиш нуқтаси бирор шартга боғланган бўлса, дастур ўз ишини ана шу шарт True бўлгандагина тўхтатади.

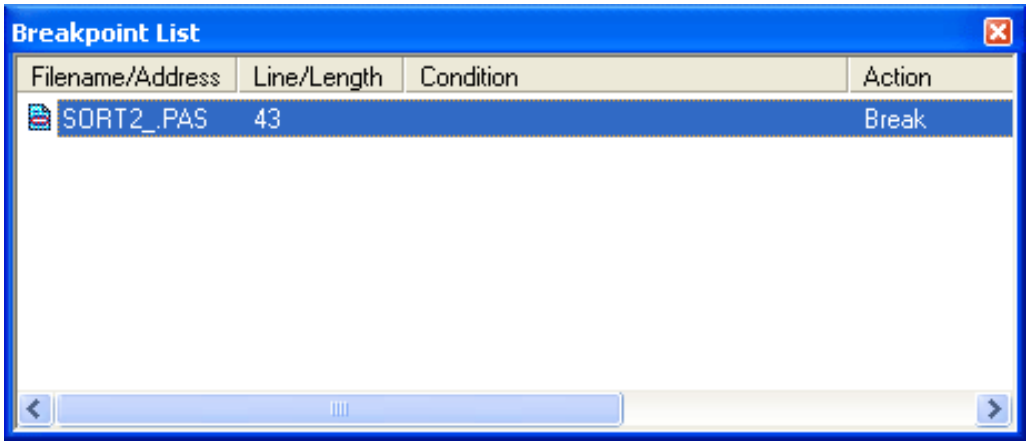
Тўхтатиш нуқтасини бирор буйруқнинг бажарилиш сонига ҳам боғлаш мумкин. Тўхтатиш нуқтасини дастурга қўшиш вақтида агар **Add Source Breakpoint** диалог ойнасининг **Pass count** (ўтишлар сони) майдонига нолдан фарқли сон киритилган бўлса, у ҳолда дастур шу нуқтадаги ўз ишини кўрсатилган буйруқ белгиланган марта бажарилганидан сўнг тўхтатади.



14.6-расм. Тўхтаиш нуқтаси қўшилганидан кейинги ойна

Тўхтатиш нуқтаси характеристикаларини ўзгартириш. Дастурчи тўхтатиш нуқтаси

характеристикаларини ўзгартиришми мумкин. Бунинг учун **View** менюсидан **Debug Windows** буйруғини танлаши, сўнгра кейинги даражали менюдан - **Breakpoints** буйруғини танлаши лозим. Очилган **Breakpoint List** диалог ойнасидан (14.7-расм.) сичқончанинг ўнг тугмасини керакли тўхтатиш нуқтаси турган сатрда чертиш керак. Шунда очилган контекст менюсидан **Properties** буйруғи танланади. Натижада **Source Breakpoint Properties** диалог ойнаси очилади. Унда тўхтатиш нуқтаси характеристикаларини ўзгартириш мумкин. Масалан, дастурни шу нуқтадаги тўхтатиш шартини (**Condition** майдонидаги**)** ўзгартириш мумкин. Шу контекст менюсидан фойдаланиб, тўхтатиш нуқтаси турган сатрга тезгина ўтиш мумкин. Бунинг учун **Edit Source** буйруғини танлаш лозим.

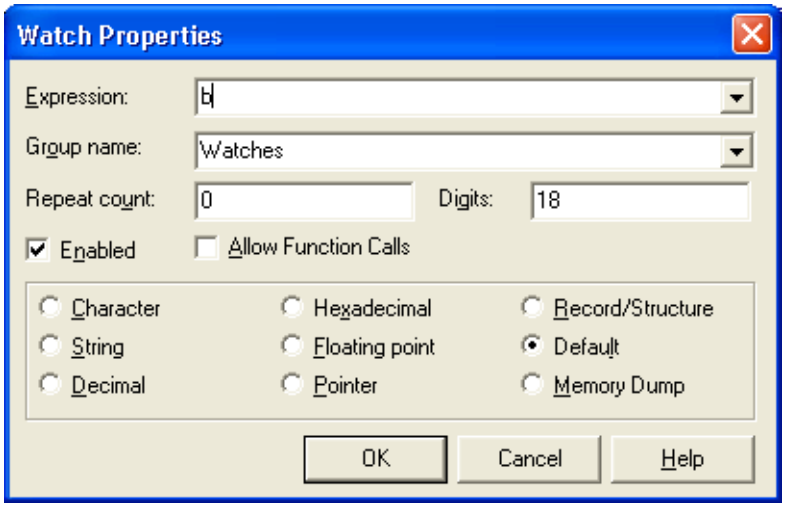


14.7. Breakpoint List ойнаси

Тўхтатиш нуқтасини ўчириш. Тўхтатиш нуқтасини ўчириш учун **Breakpoint List** диалог ойнасидаги ўчирилиши керак бўлган тўхтатиш нуқтаси ҳақидаги маълумот турган сатрда сичқончанинг ўнг тугмаси чертилади. Экранда пайдо бўладиган контекст менюсидан **Delete** буйруғини танлаш лозим.

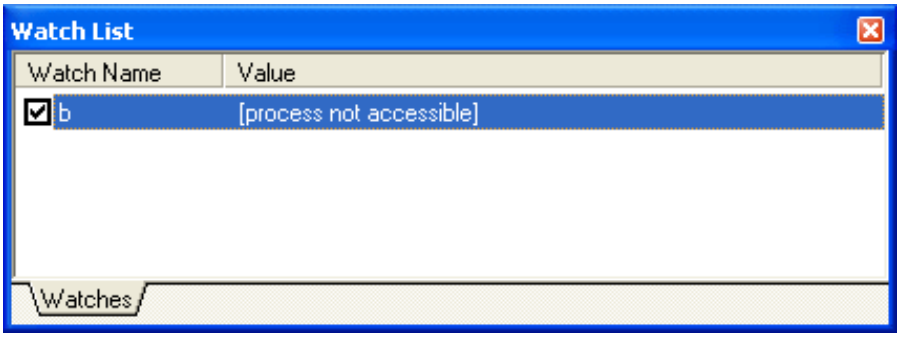
Шунингдек, кодларнинг муҳаррири ойнасида сичқончани ўчирилиши лозим бўлган тўхтатиш нуқтасининг қизил нуқтасида чертиш ҳам мумкин. .

Ўзгарувчиларнинг қийматларини кузатиб бориш. Дастурни отлада қилинаётганда, дастурни қадам-бақадам бажарилиши режимида у ёки бу ўзгарувчининг қиймати нимага тенг бўлаётганини билиш фойдадан ҳоли бўлмайди. Отладчик ўзгарувчиларнинг қийматларини кузатиб боришга имкон беради.



14.8-расм. Watch List рўйхатига ўзгарувчи номини қўшиш

Дастурнинг буйруқларини қадам-бақадам бажарилиши давомида ўзгарувчиларнинг қийматларини кузатиб боришга имкониятига эга бўлиш учун, бу ўзгарувчиларнинг номларини кузатиладиган элементлар рўйхатига (Watch List) қўшиб қўйиш керак. Бунинг учун **Run** менюсидан **Add Watch** (кузатиладиган элемент қўшиш) буйруғини танлаш ва очилган **Watch Properties** (14.8-расм) диалог ойнасининг **Expression** майдонига ўзгарувчининг номини киритиш лозим.



14.9. Watch List рўйхатига ўзгарувчи номини қўшиш натижаси.

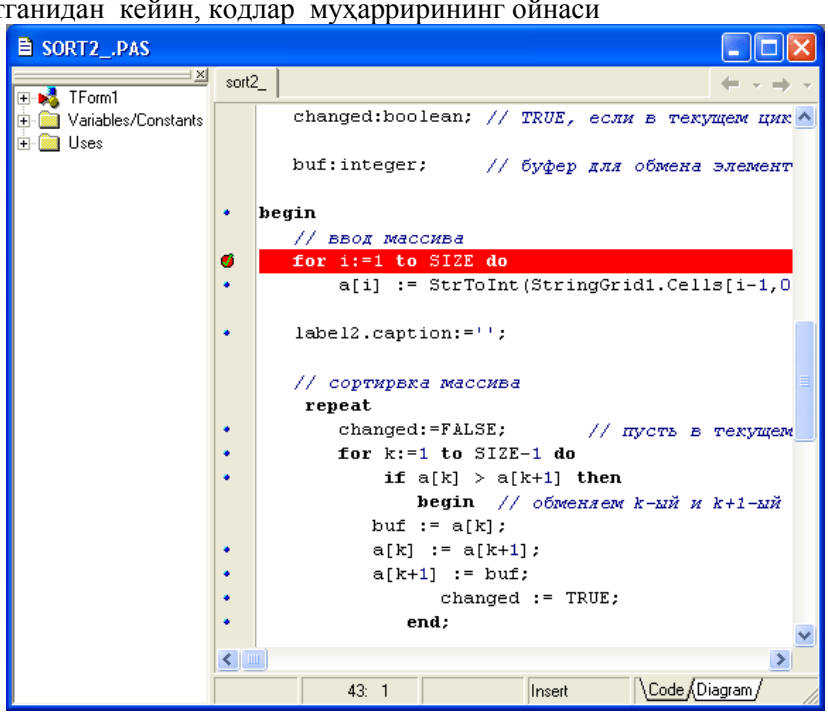
Натижада рўйхати **Watch List** (14.9-расм) диалог ойнасида бериладиган ўзгарувчилар орасида янги элемент пайдо бўлади. Дастурнинг ўзгарувчилари билан фақат дастур ишлаётган вақтдагина ишлаш мумкин бўлгани учун, ўзгарувчининг номидан кейин қуйидаги ахборот чиқарилади:

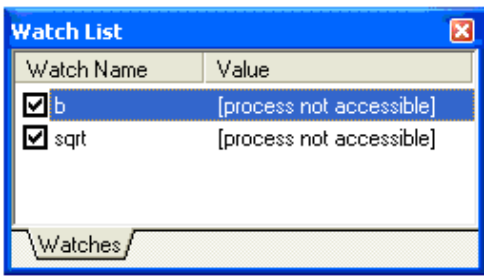
process not accessible (жараён мумкин эмас).

Мисол тариқасида 14.10-расмда массив элементларини тартиблаш дастурининг буйруқлари қадам-бақадам бажарилаётганда кодлар муҳаррирининг ойнаси ҳамда **Watch List** ойнасининг кўриниши келтирилган.

Кодлар муҳаррири ойнасида стрелка билан навбатда бажарилиши керак бўлган буйруқ (<F8> клавишаси босилганда ёки **Run** менюсидан **Step Over** буйруғи танланганда**)** кўрсатилган. **Watch List** диалог ойнасида эса ўзгарувчиларнинг қйматлари чиқарилган.

Ўзгарувчиларнинг қийматларини кузатиб боришнинг яна бир усули мавжуд. Бу усулда ўзгарувчининг номи Watch List рўйхатига қўшилмайди. Буни қуйидагича амалга ошириш мумкин. Дастур тўхташ нуқтасига етганидан кейин, кодлар муҳаррирининг ойнаси

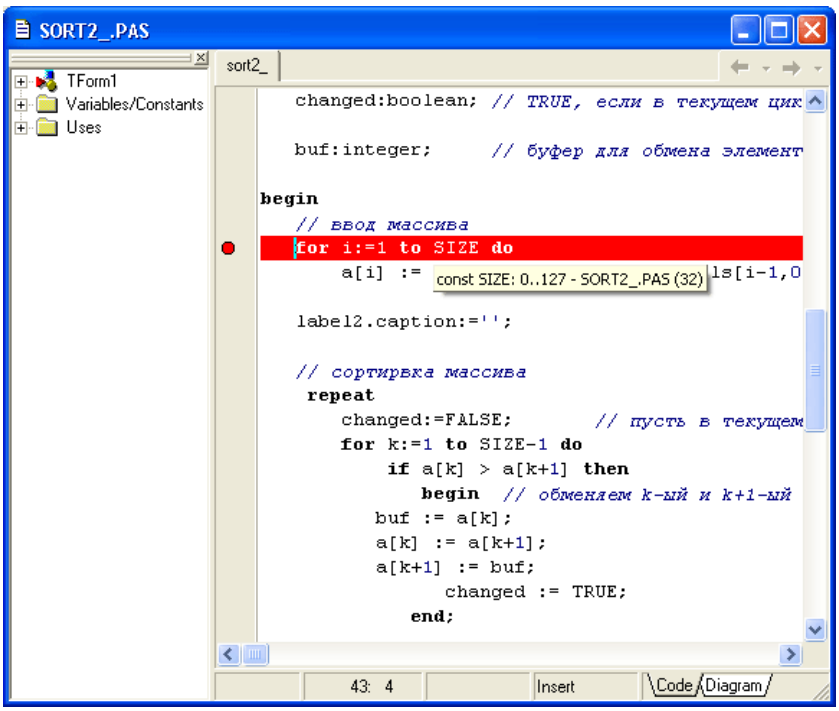




14.10қрасм. Дастурнинг қадам-бақадам бажарилишида ўзгарувчиларнинг қийматларини кузатиб бориш

очилади. Сичқонча курсорини қийматини кўриш керак бўлган ўзгарувчининг устига келтирилади. Кодлар муҳаррири ойнасида эслатмалар ойнаси очилиб, унда ўзгарувчининг қиймати кўрсатилади. (14.11-расм)

Дастурнинг қадам-бақам бажарилиши жараёнини тўхтатиш учун **Run** менюсидан **Program Reset** буйруғини танланади.



14.11-расм. Ўзгарувчиларнинг қийматларини уларнинг номини Watch List рўйхатига қўшмаган ҳолда кўриш

менюга

мундарижага қайтиш

15-боб. ЁРДАМЧИ МАЪЛУМОТНОМАЛАР СИСТЕМАСИ

15.1. Кириш

Ҳар бир дастур фойдаланувчига ёрдамчи маълумотномалар системасидан (ЁМС) фойдаланишга,

дастур ҳақидаги тўла маълумотлар, дастур билан ишлаш йўл-йўриқлари ҳақида ахборот олиш имкониятини яратиб бериши керак.

Windows муҳитида ишладиган дастурларнинг ЁМС, шу жумладан Delphi нинг ЁМС маълум бир структурадаги файллар тўпламидан иборат. Улардан фойдаланган Winhelp дастури талаб қилинган маълумотларни экранга чиқаради.

ЁМС нинг асосий элементи бўлиб ўз ичига турли ёрдамчи маълумотларни олган HLP-файллари хизмат қилади. Энг содда ҳолда, дастурнинг ЁМС битта HLP-файлидан иборат бўлиши мумкин.

ЁМС ни (HLP-файлларни) Delphi таркибига кирган Microsoft Help Workshop дастури ёрдамида ҳам яратиш мумкин. HLP-файлини яратиш учун "Бошланғич материал" бўлиб RTF-файл кўринишида сақланган маълумотнома матнлари хизмат қилади.

МС ни (HLP-файлини) яратиш жараёнини иккита босқичдан иборат деб қараш мумкин:

1. Маълумотнома матнларининг ҳужжат файлларини тайёрлаш ;
2. Бу файлларни МС файлига айлантириш.

менюга

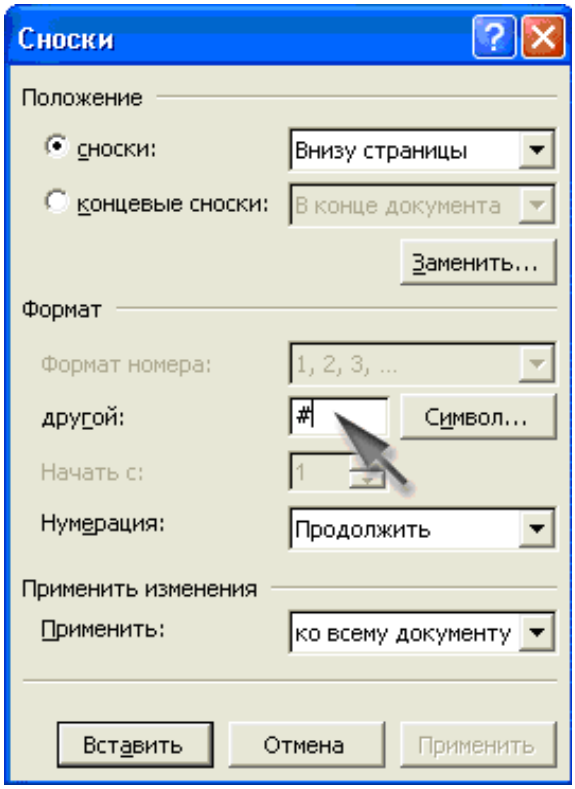
15.2 Маълумотнома ҳужжатининг файли

Ёрдамчи маълумотнома ҳужжатининг файли деганда маълум бир структурага эга бўлган ва RTF-файл кўринишида сақланган файлни тушуниш лозим. RTF-файлларини матн муҳаррирлари, масалан, Microsoft Word ёрдамида тайёрлаш мумкин. Дастлаб маълумотнома бўлимларининг матнини териш лозим. Бўлимлар сарлавҳаларини **Заголовок** стилларидан бирида, масалан, **Заголовок1** стилида расмийлаштириш керак. Ҳар бир бўлимнинг матни алоҳида саҳифада терилиши ҳамда матнлар албатта "*разрыв страницы*" (саҳифанинг охири) белгиси билан тугаши шарт.

Бўлимларнинг матнлари териб бўлинганидан кейин, 15.1-жадвалдаги махсус белгилардан (сноскадан) фойдаланиб, маълумотнома бўлимларининг сарлавҳаларини белгилаб чиқилади. (сноскалар компилятор томонидан ёрдамчи маълумотномалар системасини RTF-файлларни HLP-файлга ўтказиш жараёнида фойдаланилади.)

RTF-файлни белгилаш учун сноскалар 15.1-жадвал

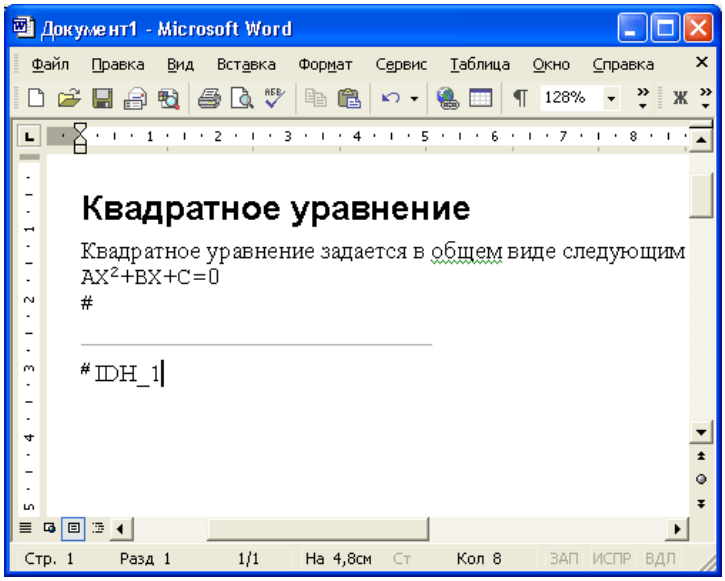
Сноска	Вазифаси
#	Маълумотнома бўлимининг идентификаторини белгилайди. Бошқа бўлимлардан шу сноска билан белгиланган бўлимга ўтиш учун фойдаланилади.
$	Бўлимнинг номини англатади ва ЁМС дан фойдаланишда қидириш рўйхатидаги бўлим маълумотнома бўлимини идентификация қилиш учун керак бўлади.
K	Калит сўзлар рўйхатини билдиради. Улардан бири танланганда шу сноска билан белгиланган маълумотнома бўлимига ўтишга ёрдам беради.



15.1-расм. Сноски диалог ойнаси

Бўлим сарлавҳасини сноска билан белгилаш учун курсорни бўлим сарлавҳасининг биринчи ҳарфи олдига келтириб, **Вставка** менюсидан **Сноска** буйруғи танланади. Очилган **Сноски** (15.1-расм) диалог ойнасидан **Вставить сноску** гуруҳидан ўчиргични **обўчную** ҳолатига ўтказилади, **Нумерация** гуруҳида эса **другая** ҳолатини белгиланади. Сноска номерини киритиш майдонига "#" белгисини киритилади ва **ОК** тугмаси босилади. Натижада хужжатга # сноскаси қўйилади, хужжат ойнасининг қуйи қисмида эса сноскаларни таҳрирлаш ойнаси пайдо бўлади. Унда сноска белгиси билан ёнма-ён маълумотнома бўлимининг идентификаторини киритиш лозим. (15.2-расм).

Идентификатор сифатида бўлим сарлавҳасининг қисқартма вариантидан фойдаланиш мумкин. Аммо, маълумотномалар бўлимининг идентификаторини IDH_ ҳарфлари билан бошлаган маъқул. Бу ҳолда RTF-файлини компиляция қилинаётганда мурожаатларнинг тўғрилиги текширилади: компилятор лойиха файлининг [MAP] бўлимида кўрсатилган, аммо RTF-файлида бўлмаган идентификаторлар рўйхатини экранга чиқаради.

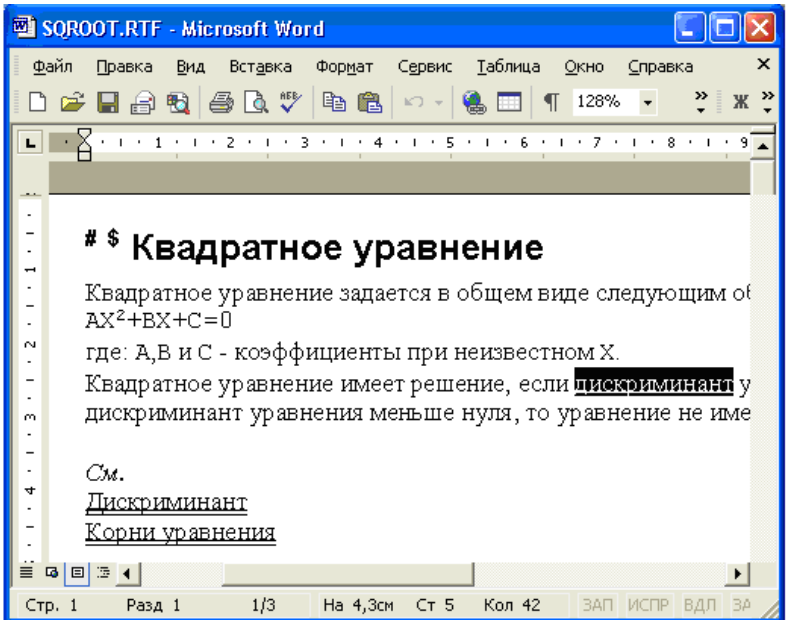


15.2-расм. Ҳужжатга бўлим номини белгиловчи сноска қўйиш.

Одатда маълумотнома бўлимлари бошқа бўлимларга ҳам мурожаат қилишларни ўз ичига олади. ЁМС ойнасида танланганидан бошқа бўлимга мурожаат қиладиган тушунчалар (сўзлар) матндан ранги билан ажратилади ва тагига чизиб қўйилади.

Маълумотномалар матнини тайёрлаш жараёнида системаси танланиши бошқа бўлимга мурожаат қиладиган мурожаат-сўзни тагига қўш чизиқ тортиш шамда сўз тугаши билан бўш жой қолдирмай, ўтилиши керак бўлган маълумотнома бўлими идентификаторини ёзиш керак. Бу идентификаторни яширин матн шаклида расмийлаштириш лозим.

15.3-расмдаги матн муҳаррирининг ойнасида квадрат тенгламани ечиш дастури учун тайёрланаётган маълумотнома файли келтирилган. "Дискриминант" сўзи маълумотноманинг бошқа бўлимига мурожаат сифатида ёзилган. Бу мурожаат қилинаётган бўлим ҳам маълумотноманинг дискриминантлар ҳақида ахборот берувчи битта бўлими бўлиб, # сноскасига эга ҳамда IDH_2 идентификатори билан белгиланган бўлиши лозим.



15.3-расм. Маълумотноманинг бошқа бўлимига мурожаат қилиш

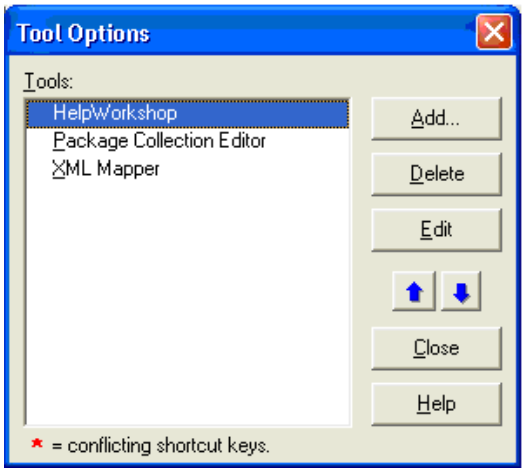
Маълумотноманинг бошқа бўлимларига мурожаат қилишни таъминловчи мурожаат-ссилкалардан ташқари, ҳужжатларга изоҳларга (сузиб чиқувчи ойналарга) мурожаатларни ҳам киритиш мумкин. ЁМС ишлаётган вақтда изоҳларга мурожаатлар бошқа ранг билан ажратилади ва тагига пунктир чизиқ билан чизиб қўйилади. ЁМС ҳужжатларини тайёрлаш жараёнида изоҳлар ҳам маълумотнома бўлимлари каби алоҳида саҳифаларга жойланади, аммо изоҳ матнлари сарлавҳасиз ёзилади. # сноскаси изоҳ матни олдига қўйилган бўлиш керак. Изоҳларга мурожаат қилиш қуйидагича расмийлаштирилади: дастлаб танланганда изоҳга мурожаат қиладиган сўзнинг тагига якка чизиқ билан чизилади, сўнгра шу сўздан кейин яширин матн тарзидаги изоҳнинг идентификатори қўйилади.

менюга

15.3. Ёрдамчи маълумотномалар системасини яратиш

ЁМС лойиҳасини яратиш. Маълумотнома файли (RTF-файл) тайёр бўлганидан кейин, ЁМС ни яратишга киришиш мумкин. Бунинг учун Delphi дастурий таъминоти таркибига кирган ва Hcw.exe файлида сақланадиган Microsoft Help Workshop дастуридан фойдаланиш анча қулай.

Microsoft Help Workshop дастурини Windows дан ёки Delphi дан **Tools** менюсидан **Help Workshop** буйруғини танлаб ишга тушириш мумкин.

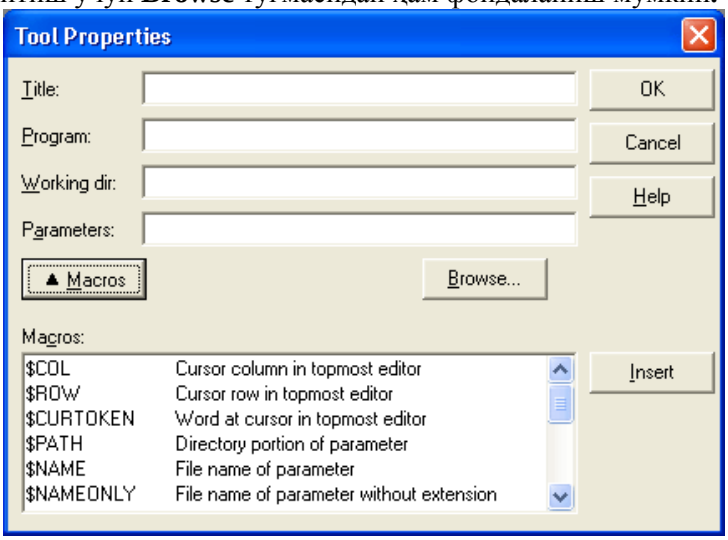


15.4-расм. Tool Options диалог ойнаси

Агар **Tools** менюсида **Help Workshop** буйруғи бўлмаса, шу менюнинг ўзидан **Configure Tools** буйруғи танланади ва очилган **Tool Options** (15.4-расм) диалог ойнасидан **Add** тугмаси чертилади. Натижада **Tool Properties** (15.5-расм) диалог ойнаси очилади. Унинг **Title** майдонига дастурнинг номи "Help workshop", **Program** майдонига эса тўлиқ, яъни йўлларини ҳам кўрсатган ҳолда — **Microsoft Help** дастурининг бажариладиган файли номини киритилади. Масалан:

C:/Program Files/Borland/Delphi7/Help/Tools/HCW.exe.

Файл номини киритиш учун **Browse** тугмасидан ҳам фойдаланиш мумкин**.**



15.5-расм. Tool Properties диалог ойнаси

Microsoft Help Workshop дастури ишга тушганидан кейин экранда дастурнинг бош ойнаси пайдо бўлади.

ЁМС яратишни бошлаш учун **File** менюсидан **New** буйруғи танланади, сўнгра очилган диалог ойнасидан яратилаётган файлнинг типи - **Help Project** белгиланади. Натижада **Project File Name** диалог ойнаси очилади. Бу ойнада дастлаб, ЁМС тайёрланаётган файл жойлашган папка ҳамда маълумотномаларнинг ҳужжат файллари (RTF-файллар) жойлашган папка кўрсатилади. Сўнгра **Имя файла** майдонига ЁМС лойиҳа файлининг номи киритилади. **Сохранить** тугмаси чертилганидан кейин экранда ЁМС лойиҳасининг ойнаси пайдо бўлади.

ЁМС ойнасидан фойдаланиб, лойиҳага зарур барча компоненталарни қўшиш, ЁМС ойнасининг характеристикаларини кўрсатиш, лойиҳани компиляция қилиш ҳамда яратилган ЁМС файлини синов тариқасида ишга тушириш мумкин.

Лойиҳага ЁМС файлини (RTF-файлини) киритиш учун **Files** тугмаси чертилади ва очилган **Topic Files** диалог ойнасидан **Add** тугмаси чертилади. Натижада **Открўтие файла** стандарт ойнаси очилади. Ундан керакли RTF-файлни танланади. Натижада лойиҳа ойнасида [FILES] бўлими пайдо бўлади ва унда маълумотнома файлининг номи кўрсатилади. Агар бу файллар бир нечта бўлса, файлларни қўшиш амалини такрорлаш лозим.

менюга

15.4. Ёрдамчи маълумотлар системаси ойнасининг ҳарактеристикалари

ЁМС бош ойнасининг характеристикаларини белгилаш учун лойиҳа ойнасида **Windows** тугмаси чертилади ва очилган **Create a window** ойнасининг **Create a window named** майдонига **main** сўзи ёзилади. **ОК** тугмаси босилганидан кейин **Window Properties** ойнаси пайдо бўлади, **General** пунктининг **Title bar text** майдонига яратилаётган ЁМС нинг бош ойнаси номини киритилади.

Window Properties диалог ойнасининг **Position** пунктидан (15.13-расм) фойдаланиб МС нинг ҳолати ва ойнасининг ўлчамларини киритиш мумкин. **Position** пунктида **Auto-Sizer** тугмаси мавжуд ва у **Help Window Auto-Sizer** (15.14-расм) ойнасини очади. Унинг ҳолати ва ўлчамлари **Position** пункти майдонларидаги сонлар билан белгиланади. Сичқонча ёрдамида бу ойнанинг ўлчамлари ва ҳолатини ўзгартириш мумкин. **ОК** тугмаси чертилганидан сўнг, **Help Window Auto-Sizer** ойнаси координаталари ва ўлчамлари **Position** пунктидаги майдонларга автоматик тарзда ёзиб қўйилади.

Color пунктидан фойдаланиб ёрдамчи маълумотлар бўлими сарлавҳаси **(Nonscrolling area color)** ҳамда маълумотлар матни турган жойларга **(Topic area color)** фон бериш мумкин. Бунинг учун **Change** нинг мос тугмасини чертиб, **Цвет** диалог ойнасида керакли ранг танланади.

Маълумотнома бўлимлари идентификаторларига сонли қиймат тайинлаш. ЁМС дан фойдаланаётган дастур аниқ бир бўлимга мурожаат қилиши учун бўлимларнинг идентификаторларига сонли қийматлар бериш керак. Бунинг учун ЁМС лойиҳасининг ойнасида **Map** тугмаси чертилади. Оқибатда **Map** диалог ойнаси очилади. Бу ойнадан **Add** тугмасини босиб, очилган **Add Map Entry** диалог ойнасининг **Topic ID** ойнасига маълумотнома бўлимининг идентификаторини киритилади, **Mapped numeric value** майдонига эса шу идентификаторга мос келадиган сон ёзилади. **Comment** майдонига изоҳларни (идентификаторга мос келадиган маълумотнома бўлимининг номи) киритиш мумкин.

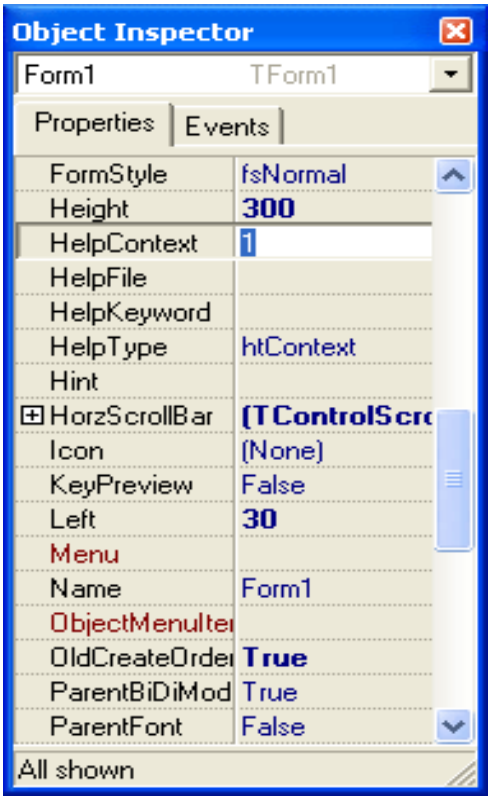
Лойиҳани компиляция қилиш. Лойиҳа файли тайёр бўлганидан кейин, лойиҳа ойнасидаги **Save and Compile** тугмасини чертиб, компиляция қилиш мумкин. Аммо, биринчи мартасида, компиляцияни **File** менюсидан **Compile** буйруғи ёрдамида бажарган маъқул. Бунинг натижасида **Compile a Help File** (15.19-расм) диалог ойнаси очилади.

Бу ойнадаги **Automatically display Help file in WinHelp when done** (компиляция тугаганидан сўнг, яратилган ЁМС ни автоматик тарзда кўрсатиш) байроқчасини ўрнатилганидан сўнг, **Compile** тугмасини чертиш керак. Компиляция тугаганидан кейин, экранда компиляция натижалари ҳақидаги ахборот пайдо бўлади. Агар компиляция муваффақиятли якунланган бўлса, яратилган ЁМС нинг ойнаси чиқарилади. Компилятор яратган ЁМС файли (HLP-файли) лойиҳа файли турган папкага ёзилади.

менюга

15.5. Ёрдамчи маълумотномалар системасидан фойдаланиш

Дастур ишлаётган вақтда фойдаланувчи <F1> тугмасини босиб, ўзи учун керакли маълумотни олиши учун илованинг бош ойнасининг HelpFile ҳусусиятига ЁМС файли номини, HelpContext ҳусусиятига эса керакли бўлимнинг сонли идентификаторини қиймат қилиб бериб қўйилиши шарт.



15.6. HelpFile ҳусусиятига МС файлини қиймат қилиб бериш.

Илованинг ЁМС файлини бажариладиган файл ёзилган папкада сақлаган маъқул.

Форманинг ҳар бир компонентаси учун, масалан, киритиш майдони учун ўзининг ёрдамчи маълумот бўлимни ташкил қилиш мумкин. Агар фокус компонентани кўрсатиб турганда ва фойдаланувчи <F1> тугмасини чертганда экранда пайдо бўладиган ёрдамчи маълумот бўлими шу компонентанинг Helpcontext ҳусусиятининг қиймати билан аниқланади. Агар бошқарув элементи HelpContext ҳусусиятининг қиймати 0 га тенг бўлса, <F1> тугмаси босилганда илова формаси учун белгиланган маълумотнома бўлими чақирилади.

Агар диалог ойнасида **Справка** тугмаси бўлса, маълумотнома ахборотлари бошқача чиқарилади. Бу тугма учун OnClick ходисаларни қайта ишлаш процедураси яратилади. У Winhelp функциясига мурожаат қилиш билан Windows Help дастурини (Winhlp32.exe файлини) ишга туширади. Winhelp функциясини чақирганда параметр сифатида маълумотнома ахборотини сўраётган ойнанинг идентификатори, ЁМС файлининг номини, Windows Help дастури бажариши керак бўлган амални белгиловчи константалар кўрсатилади.

Эслатма: Ойнанинг идентификатори — бу илова формасининг Handle ҳусусиятидир. Handle ҳусусияти билан фақат дастур ёрдамидагина ишлаш мумкин. Шунинг учун у Object Inspector ойнасидаги ҳусусиятлар рўйхатида йўқ.

Агар маълумотноманинг конкрет бўлимини экранга чиқариш керак бўлса, параметр сифатида HELP_CONTEXT константасидан фойдаланилади. Аниқловчи параметр бу ҳолда экранга чиқарилиши керак бўлган маълумотнома бўлимини белгилаб беради.

+уйида, намуна тариқасида квадрат тенгламани ечиш дастури диалог ойнасининг **Справка** (Button4) тугмаси учун Onclick ходисаларни қайта ишлаш процедураси келтирилган.

```
// Справка тугмаси чертилганда
procedure TForm1.Button4Click(Sender: TObject);
begin
winhelp(Form1.Handle, 'sqroot.hlp', HELP_CONTEXT, 1);
end;
```

менюга

15.6. HTML Help Workshop

Замонавий дастурлар ёрдамчи маълумот ахборотларини Internet-стилда экранга чиқаради, яъни

ахборот чиқарилиши учун мўлжалланган ойна Internet Explorer ойнасини эслатади. Бу ажабланарли эмас, чунки маълумотнома ахборотларини чиқариш учун Microsoft Internet Explorer асосини ташкил қилувчи компоненталардан фойдаланилади. Бу ахборотларни экранда кўрсатиш системаси Операцион системанинг бир қисми ҳисобланади, шунинг учун маълумотнома ахборотларини экранга чиқаришда ҳеч қандай қўшимча воситаларнинг кераги йўқ.

Ёрдамчи маълумот ахборотлари chm-кенгайтмали файлларда сақланади. CHM-файли — бу компиляция қилинган HTML-ҳужжат ҳисобланади. CHM-файллари бир нечта HTML –файллардан иборат бўлган HTML-ҳужжатларни ташкил қилувчи файлларни компиляция қилиш (бирлаштириш) йўли билан ҳосил қилинади.

HTML-ҳужжатни ЁМС га айлантириш жараёнини компиляция деб аталади. ЁМС компилятори учун бошланғич маълумот бўлиб, HTML-файллари, тасвирли файллар ҳамда лойиҳа файли хизмат қилади. Компиляция натижасида маълумотнома ахборотларини тўлалигича ўз ичига олган CHM-файл ҳосил бўлади.

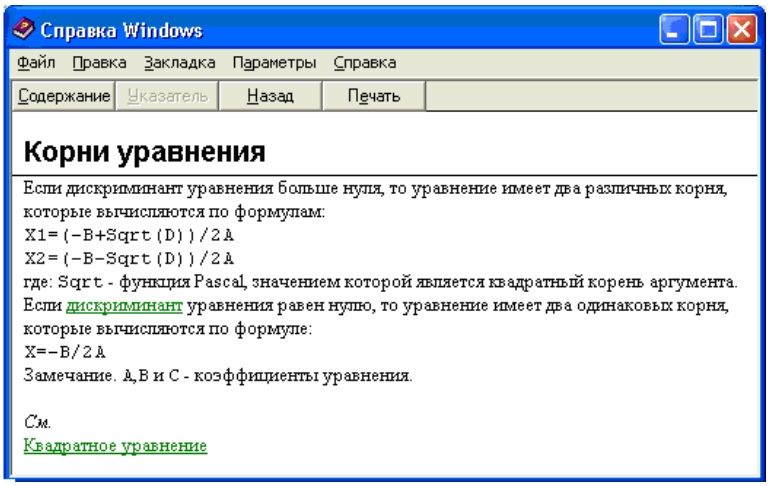
МС яратиш учун

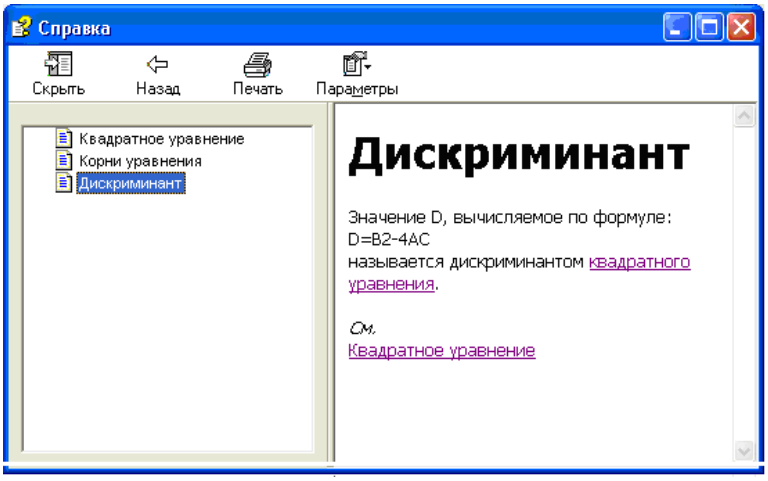
- Маълумотнома ахборотлари файлларини тайёрлаш;
- Лойиҳа файлини яратиш ;
- Контекст файлини яратиш (мундарижа);
- Компиляция қилиш

каби амалларни бажариш лозим. Бу амалларнинг охирги учтаси HTML Help Workshop дастури ёрдамида амалга оширилади.

Маълумотнома ахборотларини тайёрлаш. HTML-файлини ихтиёрий матн муҳаррири ёрдамида тайёрлаш мумкин. Агар муҳаррир терилган матнни HTML-форматида сақлашга имкон берса, бу иш янада осонлашади. Агар матнни Windows таркибига кирган WordPad (блокнот) муҳарририда териладиган бўлса, HTML тили асосларини ўрганишга тўғри келади.

Энг содда ҳолда, ЁМС ни тўлалигича битта HTML-файлга ёзиш мумкин. Аммо, ЁМС бўйлаб йўл топишда маълумотнома бўлимлари рўйхатини ўз ичига олган **Содержание** пунктидан фойдаланиш кўзда тутилган бўлса, ҳар бир бўлимнинг ахборотларини алоҳида HTML-файлида сақлаш тавсия қилинади. Намуна тариқасида 15.7-расмда **Квадрат тенглама** дастурининг ЁМС ойнаси тасвирланган. **Содержание** пункти ўз ичига учта тугмани олган. Бу эса бошланғич маълумотнома ахборотлари учта HTML-файллари билан берилганлигини англатади





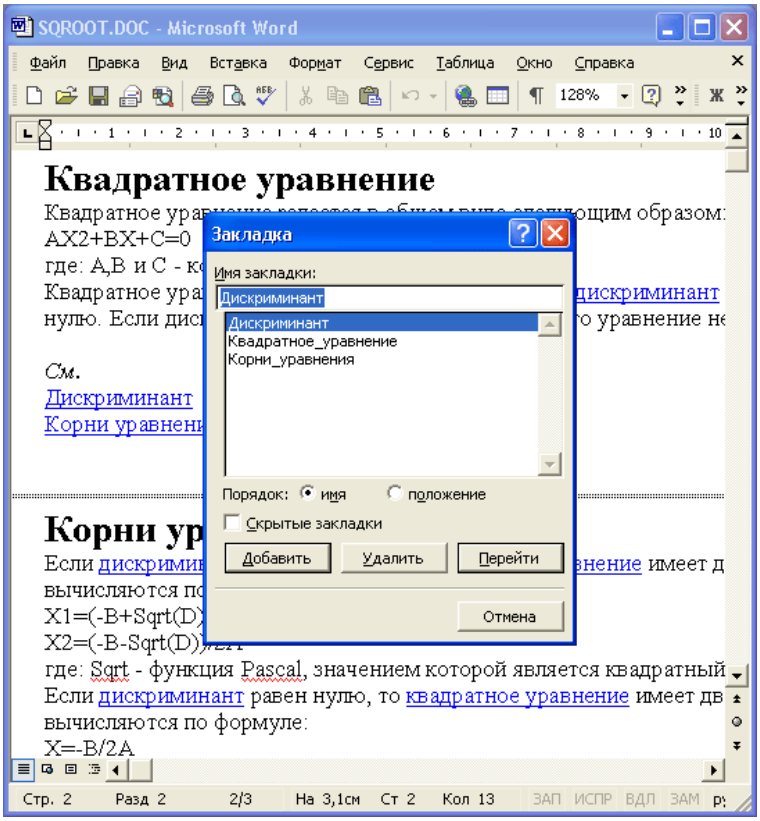
15.7-расм. МС бўйича йўл топишда Содержание пунктидан фойдаланиш мумкин

менюга

15.7. Microsoft Word матн муҳарриридан фойдаланиш

Дастлаб ёрдамчи маълумот бўлимларининг матнларини териб чиқиш лозим. Бунда ҳар бир бўлим алоҳида файлда бўлгани маъқул. Бўлимлар ва уларнинг қисмларини **Заголовок** стилларидан бирида ёзилади. Одатда, бўлимларни Заголовок1 стилида, уларнинг қисмларини эса Заголовок2 стилида расмийлаштирилади.

Кейинги қилинадиган вазифа - ҳужжатнинг бошқа нуқтасидан ўтиладиган нуқталарга хатчўп қўйишдир. Бунинг учун матннинг хатчўп қўйиладиган ерига курсорни келтирилади ва **Вставка** менюсидан **Закладка** буйруғини танланади. Шундан кейин очиладиган **Закладка** диалог ойнасининг (15.8-расм) **Имя закладки** майдонига хатчўп номини ёзилади.

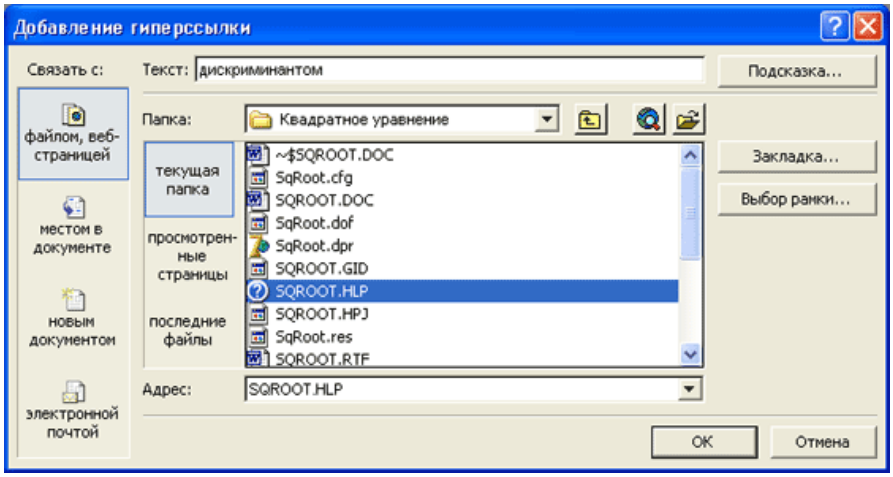


15.8.-расм. Хатчўп қўйиш

Хатчўп номи ўзи ёрдамида ўтиладиган матн парчасининг маъносини ифодалаши керак. Закладка номини ёзишда "бўш жой" белгисидан фойдаланиб бўлмайди. Унинг ўрнига тагига чизиш - "_" белгисини

қўллаш мумкин. **Заголовок** стилида расмийлаштирилган сарлавҳаларни хатчўп билан белгилаш шарт эмас. Агар яратилаётган ЁМС да фақат бўлимларнинг сарлаҳаларига ўтиш мўлжалланган бўлса, хатчўплардан фойдаланмаса ҳам бўлади. Шундан кейингина гиперссилкаларни ўрнатишга ўтиш мумкин.

Ҳужжатларда шу ҳужжатнинг ўзидаги хатчўпларга мурожаат қилишни ташкил қилиш учун гиперссилка бўлиши керак бўлган матн парчаси (сўз ёки жумла) ажратиб олинади, **Вставка** менюсидан **Гиперссўлка** буйруғи танланади ва очилган **Добавление Гиперссўлки** (15.9-расм) диалог ойнасида дастлаб **Связать с местом в этом документе** (шу ҳужжатдаги жой билан боғлаш) тугмаси, сўнгра ўтиш талаб қиладиган хатчўп ёки сарлавҳани кўрсатилади.



15.9-расм. Ҳужжатдаги ўтиш нуқтасини белгилаш

Агар бир файлда туриб, бошқа файлдаги маълумотнома бўлимига мурожаат қилиш керак бўлса, у ҳолда **Добавление гиперссўлки** диалог ойнасидан **Файл** тугмаси чертилади ва очилган ойнадан керакли HTML-файлининг номи кўрсатилади.

Ҳужжатга барча зарур гиперссилкаларни қўйилганидан сўнг, ҳужжатни HTML-форматда сақлаб қўйилади.

HTML Help Workshop дан фойдаланиш. HTML-файлларни HTML Help Workshop таркибига кирган HTML-редактор ёрдамида ҳам тайёрлаш мумкин. Аммо, бунинг учун HTML тили –гиперматнли ишора қўйиш тили асослари билан ишлашни билиш талаб қилинади. Биз қуйида қисқа, аммо ЁМС яратиш учун етарли бўлган HTML тили имкониятларини келтирамиз.

HTML-файлини яратиш учун HTML Help Workshop дастурини ишга туширилади ва **File** менюсидана **New/HTML File** буйруғи танланади ва очилган **HTML Title** ойнасида яратилаётган файлда матни жойлашадиган маълумот бўлими номини кўрсатилади. ОК тугмаси чертилганидан сўнг, HTML-редактор ойнаси очилади. Унда HTML-ҳужжатнинг шаблони (тайёр ишланмаси) тасвирланган. Шу ойнада <BODY> сатридан кейин, матнни териш мумкин.

менюга

15.8. HTML асослари

HTML-ҳужжат ўз ичига оддий матндан ташқари, теглар деб аталадиган махсус белгилар кетма-кетлигини ҳам олади. Теглар < белгиси билан бошланади ва > белгиси билан тугайди. Теглар HTML-ҳужжатларни кўриш ойнасида форматлаш (турли ўлчамларда ифодалаш) учун фойдаланилади. (Бунда тегларнинг ўзи кўринмайди).

Тегларнинг кўпчилиги жуфт-жуфт қўлланади. Масалан, $<H2>$ ва $</H2>$ теглари дастурга шу теглар орасида кўрсатилган матн парчаси иккинчи даражали сарлавҳа эканлигини ва уларни махсус стилда кўрсатиш лозимлигини билдиради.

15.2-жадвалда тегларнинг маълум бир қисми келтирилган. Бу жадвалга кирган элементлардан фойдаланиб, кейинчалик ЁМС нинг CHM-файлига айлантириш мумкин бўлган HTML-файлларини тайёрлаш мумкин.

HTML-теглари 15.2-жадвал

Тег	Вазифаси
<TITLE> ном </TITLE>	HTML-ҳужжати номини билдиради. HTML-ҳужжатларни кўрсатиш дастурлари одатда ҳужжат кўрсатилаётган ойна сарлаҳасида ҳужжат номини чиқаради. Агар сарлавҳа кўрсатилмаган бўлса, ойна сарлавҳасида файлнинг номи чиқарилади.
<BODY BACKGROUND = "Файл" BGCOLOR=" Ранг" TEXT="Ранг">	BACKGROUND параметри фон тасвирини белгилайди, BGCOLOR — фон ранги, TEXT — HTML-ҳужжат белгиларининг ранги.
<BASEFONT FACE="Шрифт" SIZE=n>	Матн учун асосий шрифтни тайинлайди: FACE - шрифт номи, SIZE – ўлчами. Кўрсатилмаган бўлса, SIZE=3. сарлавҳа шрифтининг ўлчами (<H> тегига қаранг) SIZE параметри бўлча олинади.
<H1> </H1>	Бу теглар орасидаги матн 1-даражали сарлавҳа каби ёзилади. <H2></H2> теглар жуфтлиги 2-даражали, <H3>, </H3> теглар эса 3-даражали сарлавҳани англатади
 	Сатрнинг охири. Бу тегдан кейинги сатр янги сатрнинг бошидан бошлаб чиқарилади.
<в> </в>	Бу теглар жуфтлиги ўртасидаги матн қорайтириб ёзилади.
	Бу теглар жуфтлиги ўртасидаги матн қийшайтириб ёзилади.
<AHREF="Файл.htm # Хатчўп"> </A>	Ҳужжат парчасини гиперссилка каби ажратади. У танланганда номи HREF да кўрсатилган хатчўпга ўтилади.
<IMG SRC=" Иллюстрация" >	Файлининг номи SCR параметрида кўрсатилган тасвирларни чиқаради.
<!- - -- >	Изоҳлар. Дефислар орасидаги матн экранга чиқарилмайди

HTML-матн оддий матн каби ёзилади. Тегларни катта ва кичик ҳарфлар билан ёзиш мумкин. Аммо, ҳужжатнинг структураси яхши кўринишда бўлиши учун тегларни катта ҳарфлар билан ёзишни тавсия қилинади. HTML-ҳужжатларни кўрсатувчи дастурлар ортиқча "бўш жой" белгиларини ҳамда бошқа кўринмайдиган белгиларни (табуляция, янги сатр кабиларни) кўрсатмайди. Демак, янги сатрни бошлаш учун аввалги сатр охирида
 теги, агар сатрлар орасида бўш сатр қўйилиши керак бўлса, HTML-матнга икки марта
 тегини кетма-кет қўйиш керак.

HTML Help Workshop дастурининг HTML-редактори билан ишлаганда, HTML-матнни териш жараёнида ёзилаётган матннинг ҳолатни кўриб бориш мумкин. Бунинг учун **View** менюсидан **In Browser** буйруғини танлаш ёки Internet Explorerнинг стандарт нишони турган тугмани чертиш лозим. .

```
<HTML>
<TITLE>Квадрат тенглама</TITLE>
<BODY BGCOLOR=<FFFFFF>
<BASEFONT FACE="Tahoma"SIZE=2>
<A NAME="Квадрат_тенглама">
<H2>Квадрат тенглама</H2>
</A>Квадрат тенглама умумий кўринишда қуйидагича ёзилади: <BR>
AX2+BX+Cқ0<BR>
Бу ерда : A, B ва C — номаълум X олдидаги коэффициентлар.<BR>
Квадрат тенглама
<A HREFқ"sqroot_02.htm#илдизларга ">илдиз
</A>( ечимга ), эга бўлади, агар <A HREF/"sqroot_O3.htm #Дискриминант" > дискриминант </A>нолдан катта ёки нолга тенг бўлса. Агар дискриминант нолдан кичик бўлса, тенглама ечимга эга эмас. <BR>
<BR>
<I>CM.</I><BR>
<A HREF="sqroot_O3.htm#Дискриминант">Дискриминант</A><ВК>
<A HREF="sqroot_O3.htm#илдизларга">тенгламанинг илдизлари </A> <ВК>
<BR>
```

</BODY>
</HTML>

менюга

15.9. Маълумотнома файлини яратиш

Маълумотномаларнинг HTML-файли яратилганидан кейин, энди ЁМС ни яратишга киришиш мумкин.

HTML Help Workshop ишга туширилганидан сўнг, **File** менюсидан **New/Project** буйруғи танланади ва **New Project — Destination** ойнасида МС лойиҳа файлининг номи киритилади. Бу ва навбатдаги **HTML Help Workshop** ойналаридаги **Далее** тугмаси чертилганидан биринчи қилинадиган иш - бу бўлимлар бўйича HTML-файлларни ўз ичига олган **[FILES]** бўлимни яратилишдир. Бу бўлимга файл номини қўшиш учун **Add/Remove topic files** тугмаси чертилади ва очилган **Topic Files** диалог ойнасидан — **Add** тугмаси босилади. Сўнгра очилган стандарт **Открўть** диалог ойнасидан HTML-файл танлаш лозим. Агар маълумотномалар бир нечта файллар бўйлаб тақсимланган бўлса, файлларни қўшиш амалини бир неча марта такрорлашга тўғри келади. **Topic Files** диалог ойнасида ЁМС яратиш учун керак бўлган барча файллар тўпланганидан кейин, ОК тугмаси чертилади. Натижада лойиҳа файлида ЁМС учун зарур бўлган барча файлларни ўз ичига олган [FILES] бўлими пайдо бўлади.

Навбатдаги қилинадиган иш – бу бош бўлим ва МС чиқариладиган ойнанинг сарлавҳасини кўрсатишдир. Сарлавҳа матни ва бош бўлим файлининг номи мос равишда **Options** диалог ойнасининг **General** пунктидаги **Title** ва **Default file** майдонларига, киритилади. У **Change project options** тугмасининг чертилиш натижасида экранга чиқарилади**.**

ЁМС бўйлаб йўл топиш учун **Мундарижа** пунктидан фойдаланиш кўзда тутилган бўлса, контекст файлини яратиш лозим. Бунинг учун **Contents** тугмаси чертилади ва янги контекст файлини яратиш амали тасдиқланади ҳамда контекст файлининг номи кўрсатилади. Бу ном сифатида лойиҳа номидан фойдаланиш мумкин. Натижада **Contents** пунктига мундарижа – ЁМС бўлимларининг номи кўрсатилган рўйхатни киритишга рухсат берилади.

ЁМС мундарижасини дарахтсимон рўйхат шаклида ифодалаш қабул қилинган. Юқори даражали элементлар бўлимларга мос келади, уларга бўйсинувчи даражаларга эса бўлим ичидаги мавзулар киради.

Contents пунктига ЁМС бўлимига мос келувчи янги элемент қўшиш учун **Insert a heading** тугмаси чертилади ва очилган **Table of Contents Entry** диалог ойнасининг **Entry title** майдонига бўлим номи киритилади ва **Add** тугмаси чертилади. Экранда **Path or URL** ойнаси пайдо бўлади. Бу ойнанинг **HTML titles** майдонида лойиҳа файларига кирган бўлимларнинг номлари (HTML-файлларнинг сарлавҳалари) кўрсатилади. (айнан шу файлларнинг номлари **Project** пунктининг [FILES] бўлимида берилган**).** Агар бўлим номи ўрнига файл номи кўрсатилган бўлса, бу ҳол шу файлда <TITLE> теги йўқлигидан далолат беради. Сарлавҳа ёки номи бўйича керакли файлни танлаб, ОК тугмаси босилади. Бу амалларнинг натижасида **Contents** пунктида маълумотнома бўлими номи кўрсатилган стар пайдо бўлади.

Агар қўшилган бўлимнинг нишонини алмаштириш лозим бўлса, у ҳолда **Edit selection** тугмаси чертилади ва **Table of Contents** ойнасининг **Advanced** пунктидаги **Image index** рўйхатидан фойдаланиб**,** керакли нишон танланади. (Одатда бўлим ёки унинг қисмини номи ёнида китоб нишони туради.)

Бўлимнинг қисми ҳам худди бўлим каби қўшилади ва фақат бўоим қисми қўшилганидан сўнг, **Move selection right** тугмасини чертиш лозим. Натижада сарлавҳа даражаси пасаяди, яъни бўлим бўлимнинг қисмига айланади.

Мундарижанинг маълумотнома мавзуларига мос келадиган элементлари ҳам худди шу усул билан қўшилади, аммо қўшиш жараёни **Insert a page** тугмасини чертиш билан бошланади.

Айрим ҳолларда мундарижа элементлари рўйхатининг тартибини ўзгартиришга зарурат пайдо бўлади. Бу вазифани стрелкаларнинг тасвири туширилган тугмалар ёрдамида ҳал қилиш мумкин. **Move selection up** ва **Move selection down** тугмалари рўйхатнинг ажратилган элементини рўйхат бўйлаб юқорига ва пастга қараб суриш мумкин. **Move selection right** тугмаси ажратилган элементни ўнгга суради, яъни уни рўйхатнинг аввалги элементига тобе қилиб қўяди. **Move selection left** тугмаси эса танланган элементни тобеликдан қутқаради.

менюга

15.10. Компиляция

Компиляция — бу бошланғич маълумотномаларни ЁМС файлига айнтириш жараёнидир.

HTML Help компилятор учун бошланғич маълумотнома бўлиб, лойиҳа файллари (HHP-файллар), контекст файллари (HHC), Маълумотнома файллари (HTM-файллар), иллюстрация файллари (GIF- ва JPG-файллар) хизмат қилади. .

Компиляция натижаси MC файлидан (CHM-файл) иборат бўлади.

Компиляция қилиш учун **File** менюсидан **Compile** буйруғи танланади ва очилган **Create a compiled file** диалог ойнасидаги **Automatically display compiled help file when done** (компиляциядан кейин яратилган маълумотнома файлини кўрсатиш) ўчиргичини ўрнатиб **Compile** тугмаси чертилади. Натижада маълумотнома файли яратилади ва экарнда бош бўлимдаги маълумотларни ўз ичига олган ЁМС ойнаси пайдо бўлади.

менюга

15.11. Маълумотномаларни чиқариш

CHM-файлда сақланаётган маълумотномаларни экранга чиқариш учун Windows таркибидаги махсус динамик кутубхонага (Hhopen.ocx файли) кирган Hhopen – бошқариш элементининг ActiveX-компонентасидан фойдаланилади.

Биринчидан, Hhopen компонентасини компоненталар палитрасидаги бирор пунктга ўрнатилади. Бунинг учун **Component** менюсида **Import ActiveX Control** буйруғи танланади. Экранда **Import ActiveX** ойнаси пайдо бўлади ва унда Windows реестрига кирган барча компоненталар рўйхати кўрсатилади. **Import ActiveX** ойнасидаги қайд қилинган компоненталар орасидан **hhopen OLE Control module** сатрини танлаб, **Install** тугмаси чертилади. Бунинг натижжасида экранда **Install** диалог ойнаси пайдо бўлади ва дастурчи ўрнатилаётган компонентани қўшилиш керак бўлган пакетни (packege — пакет, компоненталар кутубхонаси) танлайди. Дастурчи қўшаётган компоненталар "номи кўрсатилмаса", Dclusr пакетига қўшилади. **ОК** тугмаси чертилганидан сўнг, экранда **Package** ойнас пайдо бўлади ва пакетни қайта компиляция қилишга рухсат сўровномаси чиқарилади. Компиляция жараёни тугаганидан кейин, экранга компонента пакетга қўшилганлиги ва қайд қилинганлиги ҳақидаги ахборот чиқарилади. Hhopen компонента нишони **ActiveX** пунктига қўшилади. Компиляция жараёнида компонентани тавсифлаш файли - HHOPENLib_TLIB.pas модули яратилади ва у компонентанинг ходисалари, методлари ва ҳусусиятларини ўз ичига олади.

Тавсифлаш модулини кодлар муҳаррири ойнасига /Delphi7/Lib каталогидан юклаб кўриш мумкин. Бу ойнада HHOPENLib_TLIB.pas модулини варақлаб, THhopen классининг кўринишини топиш мумкин. (15.1-листинг).

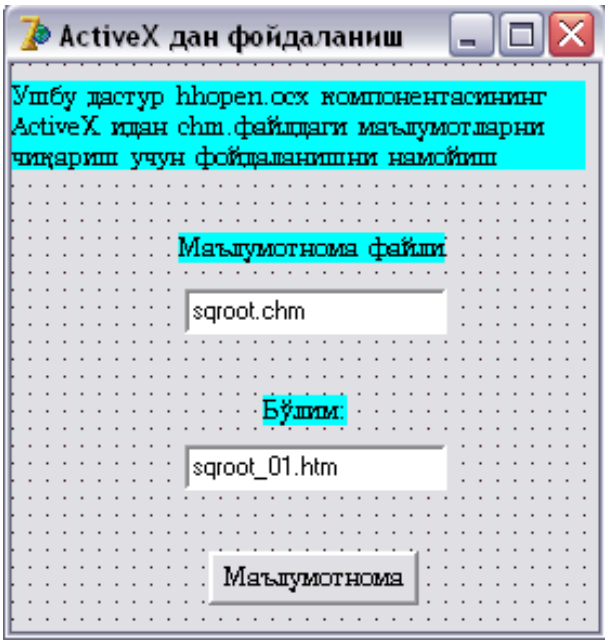
15.1-листинг. THhopen класси

```
THhopen = class(Telecontrol)
private
FIntf: _DHhopen;
function GetControlInterface: _DHhopen;
protected
procedure CreateControl;
procedure InitControlData;
override;
public
function OpenHelp(const HelpFile: WideString;
const HelpSection: WideString): Integer;
procedure CloseHelp;
property ControlInterface:_DHhopen
read GetControlInterface;
property DefaultInterface:_DHhopen
read GetControlInterface;
published
```

property isHelpOpened: WordBool index 1
read GetWordBoolProp
write SetWordBoolProp
stored False;
end;

THhopen класси иккита методни тавсия қилади: OpenHelp ва CloseHelp. OpenHelp методи маълумотнома ахборотларини экранга чиқаришни таъминлайди, CloseHelp методи эса ЁМС ойнасини ёпади. OpenHelp методининг параметрлар иккита – маълумотнома ахбороти файлининг номи ва мазмуни экранга чиқариладиган бўлимнинг номи. Бўлим номи сифатида CHM-файлини яратиш жараёнида HTML Help Workshop дастури томонидан фойдаланилган HTML-файлининг номини ҳам олиш мумкин. Ҳар икки параметрнинг widechar сатрлари эканлигини назардан қочирманг.

+уйидаги дастур (унинг диалог ойнаси 14.10-расмда, дастур матни эса 15.2-листингда келтирилган) Hhopen нинг ActiveX-компонентасидан маълумотнома ахборотини чиқариш учун қўлланишини намойиш қилади. Hhopen компонентаси формага одатдаги усул билан қўшилади. Дастур ишлаётган вақтида бу компонента экранга чиқарилмагани учун, форманинг ихтиёрий қисмига жойлаштириш мумкин.



15.10-расм. ActiveX дан фойдаланиш дастурининг диалог ойнаси

15.2-листинг. Hhopen дан фойдаланиш

```
unit ushh_;
interface
uses  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,  OleCtrls, HHOPENLib_TLB, StdCtrls;
type   TForm1 = class(TForm)
   Label1: TLabel;
   Edit1: TEdit;    //Маълумотнома файли
   Edit2: TEdit; //Маълумотнома бўлими (html файлининг номи)
   Button1: TButton; //  Hhopen компонентаси ActiveX и
   Label2, Label3: TLabel;
   procedure Button1Click(Sender: TObject);
   private     { Private declarations }
   public      { Public declarations }
  end;
var    Form1: TForm1;
implementation  {$R *.DFM}
//Ёрдам тугмасини  чертиш
procedure TForm1.Button1Click(Sender: TObject);
var   HelpFile : string;       //Маълумотнома файли
      HelpTopic : string;      //Маълумотнома бўлими
 pwHelpFile:PWideChar;//файли номи WideChar сатрига кўрсаткич
 pwHelpTopic:PWideChar;//бўлим номи WideChar сатрига кўрсаткич
```

```
begin
  HelpFile := Edit1.Text;    HelpTopic := Edit2.Text;
 //WideChar сатр учун хотирадан жой ажратиш
  GetMem(pwHelpFile, Length(HelpFile) * 2);
  GetMem(pwHelpTopic, Length(HelpTopic)*2);
 //Ansi сатрни WideString сатрга айлантириш
pwHelpFile := StringToWideChar(HelpFile, pwHelpFile,
MAX_PATH*2);
  pwHelpTopic := StringToWideChar(HelpTopic,pwHelpTopic,32);
 //маълумотнома ахборотини чиқариш
  Form1.Hhopen1.OpenHelp(pwHelpFile,pwHelpTopic);
end;
end.
```

Дастурни ишга тушириш

Маълумотнома ахборотини экранга **Маълумотнома** тугмасига боғланган OnClick ходисаларни қайта ишлаш процедураси ёрдамида чиқарилади. OpenHelp методининг параметрлари wideChar бўлгани учун, дастлаб ANSI-сатрни WideChar сатрига айлантирилади.

менюга

мундарижага қайтиш

16-боб. ДАСТУРЧИНИНГ КОМПОНЕНТАЛАРИ

16.1. Янги компонента яратиш

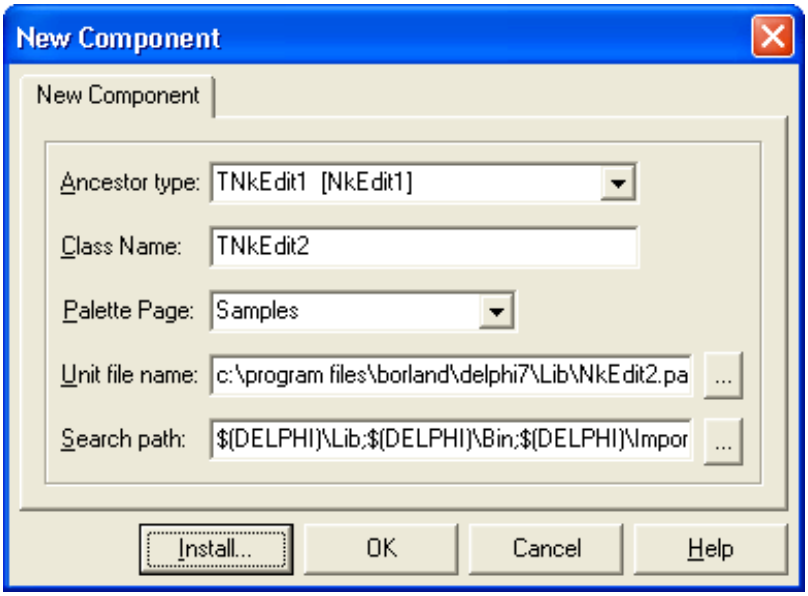
Delphi тили дастурчиларга зарурат бўлганда ўзларининг шахсий компоненталарини яратишга ҳамда уни компонентаналар палитрасига жойлаб, иловаларни тайёрлашда бошқа стандарт компонентаналар каби фойдаланишга имкон беради.

Янги компонента яратиш жараёни қуйидаги босқичлар ёрдамида амалга оширилади:
1. базавий классни танлаш;
2. компонента модулини яратиш;
3. компонентани тестдан ўтказиш;
4. компонентани компонентаналар пакетига қўшиш.

Дастурчининг компонентасини яратиш жараёнини каср сонларни киритиш учун мўлжалланган

NkEdit компонентасини яратиш мисолида кўрамиз.

Базавий классни танлаш. Янги компонента яратишга киришилар экан, унинг вазифасини аниқ белгилаб олиш лозим. Сўнгра Delphi компонеталари ичидан ўзининг функционал имкониятлари бўйича яратиладиган компонетага яқинроқ турган компонентани аниқланади. Айнан шу компонентани базавий қилиб танланади.



16.1-расм. New Component диалог ойнаси

Компонента модулини яратиш. Иш бошлашдан аввал компонентанинг модули ва бошқа ёрдамчи файллари учун алоҳида папка очиш лозим. Шундан кейин компонента модулини яратиш жараёнини бошлаш мумкин.

Компонента модулини яратиш учун **Component** менюсидан **New Component** буйруғи танланади ва очилган **New Component** (16.1-расм) диалог ойнасига яратилаётган компонента ҳақидаги маълумотлар киритилади.

Ancestor type майдонига яратиладиган компонента учун базавий тип кўрсатилади. Базавий тип номини майдонга тўғридан-тўғри киритиш ёки очилган рўйхатдан танлаб киритиш ҳам мумкин. Биз яратмоқчи бўлган компонента учун базавий қилиб Edit (киритиш-таҳрирлаш майдони) компонентасини танладик. Шунинг учун янги компонентанинг базавий типи деб TEdit ни оламиз.

Class Name майдонида яратилаётган компонентанинг класси номини кўрсатилади, масалан, TNkEdit. Delphi да типларнинг номи T ҳарфи билан бошланишини эсга олинг.

Palette Page майдонига янги компонента яратилганидан кейин, унинг нишони қайси куроллар панелига қўшиб қўйилиши ёзилади. Куроллар панели номини очиладиган рўйхатдан ҳам танлаш мумкин. Агар **Palette Page** майдонига ҳали мавжуд бўлмаган куроллар панели номи ёзилса, аввал шундай номдаги куроллар панели яратилади ва унга янги компонента қўшилади.

Unit, file name майдонидан яратилаётган компонента модули файлининг номи жой олади. Delphi компонента модулига компонента номи билан бир хил (аммо "T" ҳарфисиз) ном беради. Учта нуқтали тугмани чертиб, компонента модулини сақлаш учун каталог танлаш мумкин.

OK тугмаси чертилганидан кейин, жорий лойиҳага модул компонентаси учун махсус яратилган шаблони (тайёр ишланмаси) қўшилади. Бу модулнинг матни 16.1-листингда келтирилган.

16.1-листинг. Компонента модулининг шаблони

```
unit NkEdit;
interface
uses
Windows, Messages, SysUtils, Classes, Controls, StdCtrls;
type
TEdit1 = class(TEdit)
private
{ Private declarations }
protected
```

```
{ Protected declarations }
public
{ Public declarations }
published
{ Published declarations }
end;
procedure Register;
implementation
procedure Register;
begin
RegisterComponents('Samples', [TNkEdit]);
end;
end.
```

Янги классни эълон қилишда фақат ота класси кўрсатилган ҳалос. Реализация бўлимида **Register** процедураси жойлашган ва у янги классни қайд қилишда Delphi нинг кўрсатилган қуроллар панелига дастурчи яратган компонентани ўрнатиш мақсадида фойдаланилади.

Янги компонента классининг эълонига қуйидаги қўшимчаларни киритиш лозим: ҳусусиятларни кўрсатиш, бу ҳусусиятларнинг майдонлари, майдондаги маълумотлар билан ишлаш учун функциялар, майдонга маълумотлар киритиш процедуралари, конструктор ва деструктор. Агар айрим ходисаларни янги компонента базавий компонента каби қайта ишлаши талаб қилнмаса, у ҳолда класснинг эълонига уларга мос ходисаларни қайта ишлаш процедураларини эълон қилинади.

16.2-листингда NkEdit компонентаси модулининг ўзгартиришлар киритилганидан кейинги ҳолати келтирилган.

16.2-листинг. NkEdit компонентаси модули.

```
unit NkEdit;
interface
uses
Windows, Messages, SysUtils,
Classes, Graphics, Controls,
Forms, Dialogs, StdCtrls;
type
TNkEdit = class(TEdit)
private
FNumb: single;//таҳрирлаш майдонидаги сони
//Fnumb майдонига маълумот киритиш ва қайта ишлаш функцияси
function GetNumb: single;
procedure SetNumb(value:single);
protected
procedure KeyPress(var Key: Char);
override;
public
published
constructor Create(AOwner:TComponent);
override;
property Numb : single//компонентанинг ҳусусияти
read GetNumb;
write SetNumb;
end;
procedure Register;
implementation
//компонентани қайд қилиш процедураси
procedure Register;
begin
RegisterComponents('Samples',[TNkEdit]);
end;
//компонента конструктори
constructor TNkEdit.Create(AOwner:TComponent);
```

```
begin
// don't forget to call the ancestors' constructor
inherited Create(AOwner);
end;
// FNumb майдони билан ишлаш функцияси
function TNkEdit.GetNumb:single;
begin
try // Text майдони бўш бўлиши мумкин
Result := StrToFloat(text);
except
on EConvertError do begin
Result := 0; text :=' ' ;
end;
end;
end;
// Fnumb майдонига ёзиш процедураси
procedure TNkEdit.SetNumb(Value:single);
begin
Fnumb := Value;
Text := FloatToStr(value);
end;
// KeyPress ходисаларни қайта ишлаш процедураси
procedure TNkEdit.KeyPress(var key:char) ;
begin
case key of
'0'.. '9', #8, #13: ;
'-': if Length(text)<>0 then key := #0;
else
if not ((key = DecimalSeparator) and
(Pos(DecimalSeparator,text) = 0))
then key := #0;
end;
inherited KeyPress(key);
// Ота классидаги OnKeyPress процедурасини чақириш
end;
end.
```

TNkEdit классини ифодалашда Numb ҳусусияти эълон қилинган ва у таҳрирлаш майдонига киритилган сондан иборат бўлади. Numb ҳусусияти қийматини сақлаш учун Fnumb майдонидан фойдаланилади. GetNumb функцияси Fnumb майдонига кириш, setNumb процедураси эса ҳусусияти қийматини белгилаш мақсадида қўлланмоқда.

TNkEdit классининг конструктори дастлаб ота классининг конструктори -TEdit ни чақиради, унинг Text ҳусусиятига қиймат беради, сўнгра Numb ҳусусиятининг қийматини аниқлайди.

NkEdit компонентасининг бирор клавишани босилишига реакцияси TNkEdit.KeyPress ходисаларни қайта ишлаш процедураси билан аниқланади. TNkEdit.KeyPress процедураси параметр сифатида босилган клавиша кодини олади. Ота классидаги OnKeyPress ходисаларни қайта ишлаш процедурасини чақиришдан аввал босилган клавиша коди ҳақиқий сонларни ёзишда ишлатиладими ёки йўқлиги текширилади. Агар NkEdit компонентаси учун мумкин бўлмаган клавиша босилган бўлса, у ҳолда киритилган код нол билан алмаштирилади. Маълумки, NkEdit компонентаси учун рақамлар, бутун ва каср қисмни ажратувчи белги (Windows нинг созланишига қараб нуқта ёки вергул), "минус", <Backspase> (нотўғри киритилган белгини ўчиради) ва <Enter> тугмаларидан фойдаланиш мумкин ҳалос.

Бу ерда шуни эсга олиш керакки, дастур матнида соннинг бутун ва каср қисми бир-биридан нуқта билан ажратилади. Дастур иш бошлаганда эса, малумотларни киритишда Windows созланишига қараб, нуқта ёки вергулдан фойдаланилади. Келтирилган OnKeyPress ходисаларни қайта ишлаш процедураси Windows созланиши ўзгариши мумкинлиги эътиборга олади ва фойдаланувчи киритган белгини константа билан эмас, балки қиймати ажратувчи белгига тенг бўлган DecimalSeparator глобал ўзгарувчининг қиймати билан таққослайди.

Компонента модули матни киритилганидан сўнг, модулни компиляция қилиб, сақлаб қўйилади.

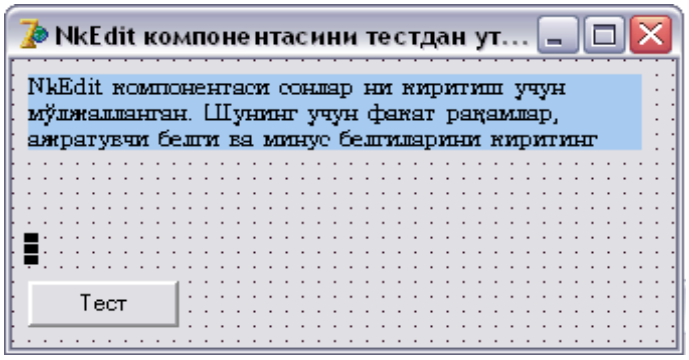
менюга

16.2. Компонента модулини тестдан ўтказиш

Янги компонентани компоненталар палитрасига қўшишдан аввал, уни ҳар томонлама текширувдан ўтказиш лозим. Бунинг учун шу компонентадан фойдаланувчи илова яратиш ва компонента кўнгилдагидек ишлаётганлигини таҳлил қилинади.

Форма яратиш жараёнида ҳали нишони компоненталар палитрасида мавжуд бўлмаган компонентани формага қўшиб бўлмайди. Бундай компонентани динамик, яъни дастур ишлаб турганда формага жойлаш мумкин.

Тест ўтказувчи илова одатдаги иловалар каби ташкил қилинади: дастлаб илова формаси, сўнгра илова модули яратилади.



16.2-расм NkEdit компонентасини текшириш

NkEdit компонентасининг текширувчи илова формасининг қўриниши 16.2-расмда берилган.

Формада иккита Label ва битта Button тугмалари мавжуд. Label1 эслатма-ахборотни экарнга чиқаради, Label2 эса (расмда у ажратиб кўрсатилган) киритиш майдонидаги сонни чиқариш учун мўлжаланган. Ҳозирча NkEdit тугмаси формада йўқ. Бу компонент дастур ишга тушганидан кейин ддинамик тарзда яратилади ва формага қўйилади.

Форма яратилганидан кейин, Delphi автоматик тарзда ҳосил қилган илова модулига қуйидаги ўзгаришларни киритиш лозим:

1. Фойдаланиладиган модуллар рўйхатига (uses бўлими) тестдан ўтказилаётган компонента (NkEdit) модулининг номини қўшамиз.

2. Ўзгарувчиларни эълон қилиш бўлимига (var) компонентани эълон қилиш буйруқлари ёзилади. Шуни ёдда тутиш керакки, янги компонента объект ҳисобланади, шунинг учун компонентани ўзгарувчиларни эълон қилиш бўлимида эълон қилишнинг ўзи янги компонента яратилишини таъминламайди, фақат шу компонентага кўрсаткични қайд қилади ҳалос. Демак, ҳақиқатдан ҳам янги компонента ярата оладиган объект конструкторини чақиришга тўғри келади.

3. Илова формаси учун OnCreate ходисаларни қайта ишлаш процедурасини қўшиш керак. У тестдан ўтказилаётган компонента конструкторини чақириб, компонента яратади ва унинг қийматларини ўрнатади.

16.3 – листингда NkEdit компонентасининг илова модули матни келтирилган.

16.3 – листинг. NkEdit компонентасининг илова модули

```
unit tstNkEdit_;

interface
uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, NkEdit; // компонента модули
type
TForm1 = class(TForm)
Label1: TLabel;
Label2: TLabel;
Buttonl: TButton;
```

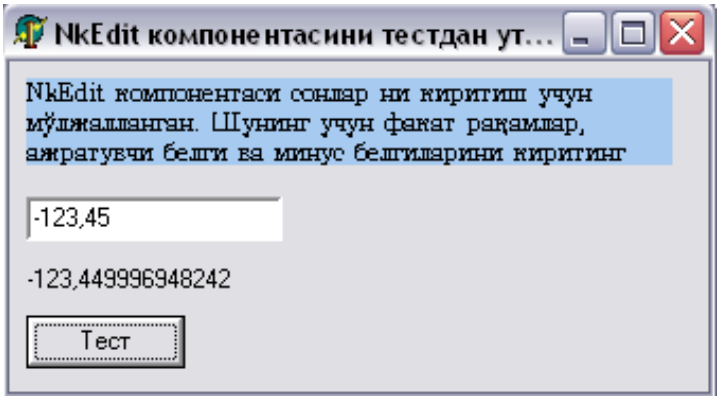
```
procedure FormCreate(Sender: TObject);
procedure ButtonlClick(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;
var
Form1: TForm1;
myEdit: TnkEdit;// NkEdit компонентаси
implementation
{$R *.dfm)
procedure TForm1.FormCreate(Sender: TObject);
begin
//компонентани яратиб, формага жойлаймиз
myEdit := TNkEdit.Create(self);
myEdit.Parent := self;
myEdit.Left := 8;
myEdit.Top := 64;
end;
procedure TForm1.ButtonlClick(Sender: TObject);
begin
label2. Caption := FloatToStr (myEdit .Numb) ;
end;
end.
```

Дастурни ишга тушириш

Тестдан ўтказилаётган компонента Formcreate (форма яратиш) ходисаларни қайта ишлаш процедураси билан компонента конструкторини чақириш воситасида яратилади. Унга параметр сифатида Self қиймати берилади, бу эса илова формаси яратилган компонентанинг эгаси эканлигини англатади.

Компонента яратилганидан кейин, муҳим бир амални бажариш лозим: Parent ҳусусиятига қиймат бериш керак. Биз кўраётган ҳолда тестдан ўтказилаётган компонента илова формасида жойлашган, шунинг учун Parent ҳусусиятига Self қиймати берилади.

16.3-расмда **NkEdit компонентасини тестдан ўтказиш** дастурининг ишлаш вақтидаги илова формасини, яъни таҳрирлаш майдонига сон киритилган ва **Тест** тугмаси чертилгандан кейинги вазият тасвирланган**.**



16.3-расм. NkEdit компонентаси – киритиш майдонини тестдан ўтказиш

Компонентани ўрнатиш. Янги компонентанинг нишони компонента лар палитрасида пайдо бўлиши учун бу компонентани бирор қуроллар пакети (Packages) таркибига қўшиб қўйиш лозим. Қуроллар (компонента лар) пакети – бу .dpk (Delphi Package File) кенгайтмали файллардир. Масалан, дастурчи яратган компонента лар Dclusr70.dpk пакетида жойлашади.

Компонентани Delphi пакетига қўшиш учун компонента модули ва компонентанинг битли тасвир - нишони ёзилган ресурслар файлидан фойдаланилади. Ресурслар файли .DCR (Dynamic Component Resource) кенгайтмали файллардир. Ресурс файллари ичидаги битли тасвирлар компонента номи билан бир ҳил номга эга бўлиши лозим.

Компонентанинг ресурслари. Компонентанинг ресурслари файлини **Tools** менюсидан **Image Editor** буйруғи билан ишга тушириладиган Image Editor утилити ёрдамида яратиш мумкин.

Янги компонента ресурсларии файлини яратиш учун **File** менюсидан **New** буйруғи танланадаи ва очилган рўйхатдан яратиладиган файлнинг типи-**Component Resource File** танланади. Натижада ресурслар файли Untitledl.dcr нинг ойнаси очилади, **Image Editor** диалог ойнасининг менюсида эса янги пункт - **Resource** пайдо бўлади. Энди **Resource** менюсидан **New / Bitmap** буйруғи танланади ва очилган **Bitmap Properties** ойнасида компонента нишонининг битли тасвири характеристикалари ўрнатилади: **Size** — 24x24 пиксел, **Colors — 16.** Бу бажарилган амаллар натижасида яратилаётган ресурслар файлига номи **Bitmap1** бўлган янги ресурс-битли тасвир қўшилади. Ресурс номи **Bitmapl** да сичқончани икки марта чертилса, битли тасвир муҳаррири ойнаси очилади ва энди унда керакли тасвирни чизиш мумкин бўлади.

Зарурат бўлса, график муҳаррир ойнасидаги тасвирни катталаштириш мумкин. Бунинг учун **View** менюсидан **Zoom In** буйруғини танлаш лозим.

Шуни эсда сақлаш керакки, тасвир ўнг қуйи бурчагининг ранги "шаффоф ранг" ни белгилайди. Компонента нишонининг шу ранг билан чизилган элементлари Delphi компонеталари палитрасида кўринмайди.

Компонента ресурслари файлини сақлашдан аввал битли тасвирга ном бериш керак. Бу ном компонента классининг номи билан устма-уст тушиши лозим. Битли тасвирга ном бериш учун битли тасвир номи **Bitmap1** устида сичқончанинг ўнг тугмаси чертилади ва очилган кентекст менюсидан **Rename** буйруғини танланади ва файлнинг янги номи киритилади.

Яратилган компонента ресурслари файлини компонента модули сақланаётган каталогда сақлаш лозим. Бунинг учун **File** менюсидан **Save** буйруғи танланади.

Диққат! Компонента ресурслари файлининг номи (Edit.dcr) компонента модулининг номи (Edit.pas), битли тасвирнинг номи эса (Edit) — компонента классининг номи (Edit) билан бир ҳил бўлиши керак.

менюга

16.3. Компонентани ўрнатиш.

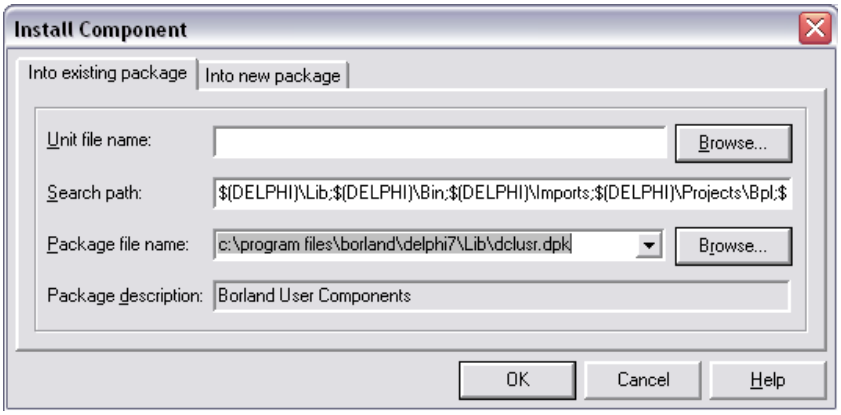
Компонента ресурслари файли яратилганидан сўнг, компонентани ўрнатишга киришиш мумкин. Бунинг учун **Component** менюсидан **Install Component** буйруғини танлаш ва очилган ойнанинг **Install Component** майдонини тўлдириш керак. (16.4-расм).

Unit file name ойнасига модул файлининг номи киритилади. Бу вазифани **Browse** тугмасидан фойдаланиб, осонгина ҳал қилиш мумкин.

Эслатма: Компонента ресурслар файли **Search path** майдонида кўрсатилган каталоглардан бирида сақланган бўлиш керак. Акс ҳолда, унинг ота классининг нишони шу компонентага тайинланади.

Package file name майдонига компонента қўшиладиган пакетнинг номи киритилади. Бу ном кўрсатилмаса, дастурчи яратган янги компонента Dclusr70.dpk пакетига қўшилади.

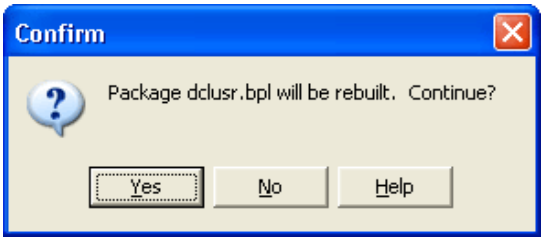
Package description майдонига пакетнинг номи ёзилади. Dclusr70.dpk пакети учун бу ном



16.4-расм. Install Component диалог ойнаси

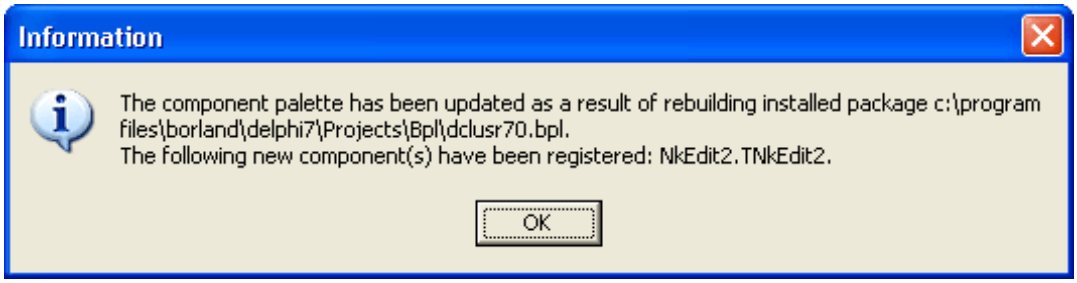
Borland User's Components.

матнидан иборат. Юқорида кўрсатилган майдонлар тўлдирилганидан сўнг, **ОК** тугмаси чертилади ва компонентани ўрнатиш жараёни бошланади. Дастлаб, экранда **Confirm** (16.5-расм) ойнаси пайдо бўлади ва унда Delphi пакетни янгилашга рухсат сўрайди.



16.5. Пакетни янгилаш учун рухсат сўраш.

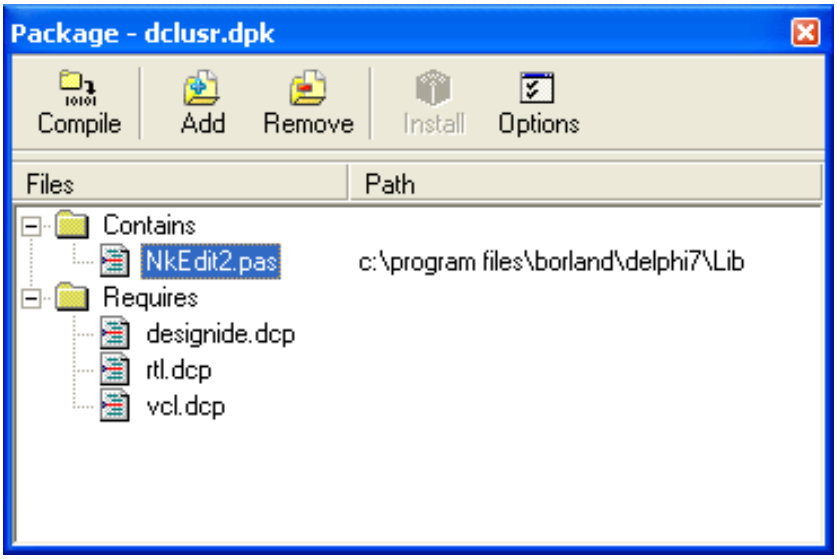
Yes тугмаси босилганидан сўнг, компонентани ўрнатиш жараёни давом этади. Агар у кўнгилдагидек тугаса, экранга пакетнинг янгиланганлиги ҳақидаги ахборот (16.6-расм) чиқарилади. Демак, компоненталар палитрасига янги компонента қўшилди.



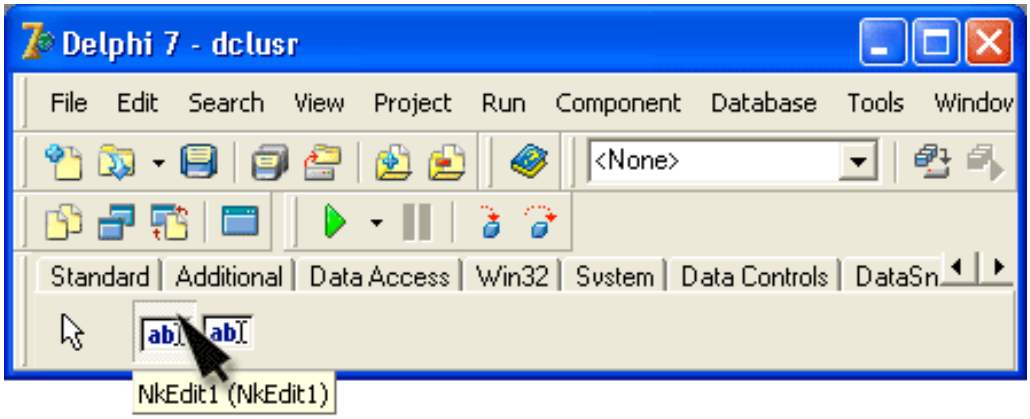
16.6-расм. Компонентанинг ўрнатилганлиги ҳақидаги ахборот

Компонента ўрнатилганидан сўнг, **Package** (компоненталар пакетининг муҳаррири) (16.7-расм.) диалог ойнаси очилади ва унда пакетдаги компоненталар рўйхати кўрсатилади.

Шу билан компонента ўрнатиш жараёни тугайди. Натижада янги компонента кирган гуруҳнинг куроллар панелида унинг нишони пайдо бўлади. (16.8-расм).



16.7-расм. Компоненталар пакетининг редактори



16.8-расм. NkEdit компонентаси ўрнатилганидан кейинги кўриниш.

менюга

16.4. Компонентани ўрнатишдаги хатолар.

Янги компонента билан ишлашда энг кўп учрайдиган хатолик - бу бирор пакетда жойлашган компонентани янгидан ўрнатишга уринишдир. (айниқса, бундай истак компонента модулига ўзгаришлар киритилганидан сўнг юзага келиши мумкин.) бу ҳолда Delphi экранга қуйидаги ахборотни чиқаради:

The package already contains unit named...

(Пакетда ... деб аталадиган модул мавжуд)

Компонентани ўрнатиш жараёни шу билан тўхтайди. Бу хатонинг олдини олиш учун ҳамда компонентани керакли пакетга қўшиши учун, ёки унинг янгиланган версиясини пакетга қўшиш учун, дастлаб компонентани пакетдан ўчирилади ва қайтадан ўрнатилади.

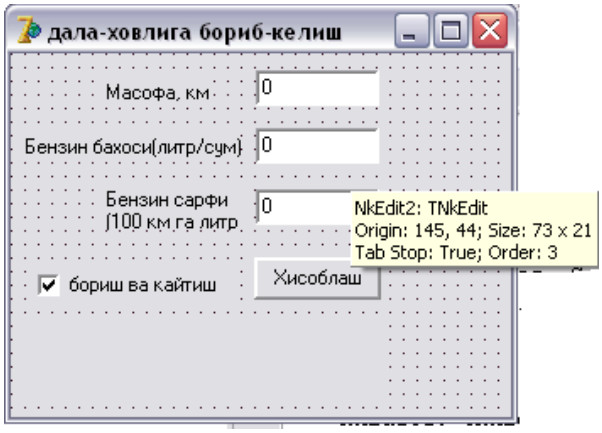
менюга

16.5. Компонентани тестдан ўтказиш

Компонента пакетга қўшилганидан кейин, компонентанинг иловалар ишлаб чиқиш пайтидаги ҳулқини таҳлил қилиш лозим. (Компонентанинг ишга лаёқати уни илова формасига динамик қўшилаётган вақтда текшириб кўрилган эди.)

Компонентанинг тўғри ишлаётганлигига илова формасини ишлаб чиқиш жараёнида уни формага қўшиб, **Object Inspector** ойнаси ёрдамида унинг янги ва отасидан олган ҳусусиятларининг қийматларини ўрнатиш мумкин бўлгандагина ишонч ҳосил қилиш мумкин.

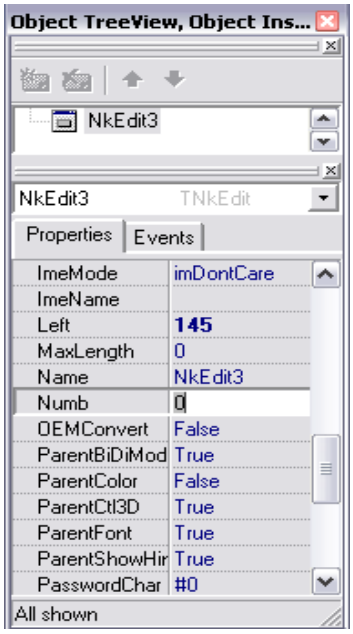
NkEdit компонентасининг имкониятларини **дала-ҳовлига бориш** дастури ёрдамида текширамиз. Бу дастурнинг формаси 16.9-расмда берилган.



16.9-расм. Дала-ҳовлига бориб-келиш дастурининг диалог ойнаси
(NkEdit компонентасининг киритиш-таҳрирлаш майдон)

Бир қараганда, бу форма аввалги формалардан фарқ қилмайдиганга ўхшайди. Аммо, киритиш-таҳрирлаш майдонларига ўтилса, бу объектни **Object Inspector** ойнасида TnkEdit класси типидаги компонента эканлиги кўрсатилади. Ҳусусиятлари рўйхатида эса (стандарт Edit компонентасига нисбатан) янги ҳусусият - Numb (16.10-расм) ни кўриш мумкин.

16.4-листингда **Дала-ҳовлига бориб-келиш** иловасининг модули келтирилган. Бу датур матнининг 6-бобдаги матнга қараганда анча ихчамлиги кўзга ташланиб турибди.



16.10-расм. NkEdit компонентаси ҳусусиятлари Object Inspector ойнасида

16.4-листинг 16.4. Дала-ҳовлига бориб-келиш иловаси ёрдамида компонентани тестдан ўтказиш

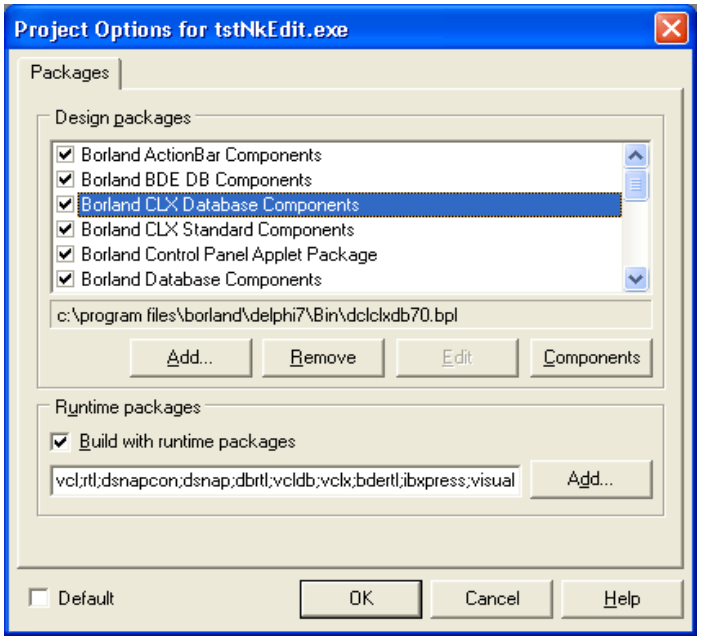
```
unit fazenda_;
interface
uses   Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,   Dialogs, StdCtrls, NkEdit;
type
  TForm1 = class(TForm)
    NkEdit1: TNkEdit;    //масофа
    NkEdit2: TNkEdit;    //бензин баҳоси
    NkEdit3: TNkEdit;    //100 км га бензин сарфи
    CheckBox1: TCheckBox;//True - бориб келиш
    Button1: TButton;    //ҳисоблаш тугмаси
    Label4: TLabel;      //Ҳисоблаш натижасини чиқариш учун
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
```

```
    procedure Button1Click(Sender: TObject);
    procedure NkEdit1KeyPress(Sender: TObject; var Key: Char);
    procedure NkEdit2KeyPress(Sender: TObject; var Key: Char);
    procedure NkEdit3KeyPress(Sender: TObject; var Key: Char);
  private     { Private declarations }
  Public     { Public declarations }   end;
var
Form1: TForm1;
implementation
{$R *.dfm}
//масофа майдонида клавиша босиш
procedure TForm1.NkEdit1KeyPress(Sender: TObject; var Key: Char);
begin
    if Key = Char(VK_RETURN)
      then NkEdit2.SetFocus; //нарх майдонига ўтказиш
end;
//нарх майдонида клавиша босиш
procedure TForm1.NkEdit2KeyPress(Sender: TObject; var Key: Char);
begin
    if Key = Char(VK_RETURN)
     then NkEdit3.SetFocus;//курсорни истеъмол майдонига ўтказиш
end;
//истеъмол масофа майдонида клавиша босиш
procedure TForm1.NkEdit3KeyPress(Sender: TObject; var Key: Char);
begin
    if Key = Char(VK_RETURN)
      then Button1.SetFocus;//Ҳисоблаш тугмасини фаоллаштириш
end;
//Ҳисоблаш тугмасини босиш
procedure TForm1.Button1Click(Sender: TObject);
var
  rast : real;//масофа
  cena : real;//нарх
  potr : real;//100 км га бензин истеъмоли
  summ : real;//суммаси
  mes: string;
begin
  rast := StrToFloat(NkEdit1.Text);
  cena := StrToFloat(NkEdit2.Text);
  potr := StrToFloat(NkEdit3.Text);
  summ := rast/100 * potr * cena;
  if  CheckBox1.Checked  then
     summ := summ * 2;
  mes := 'дала-Ҳовлига бориш';
  if CheckBox1.Checked then     mes := mes + ' ва қайтиб келиш';
  mes := mes + FloatToStrF(summ,ffGeneral,4,2) + ' сумга тушади.';
  Label4.Caption := mes;
end;
end.
```

менюга

16.6. Компонентани ўчириш

Айрим ҳолларда компонентани пакетдан ўчиришга эхтиёж пайдо бўлади. Бу ишни компоненталар пакети муҳаррири ёрдамида бажарилиши мумкин.

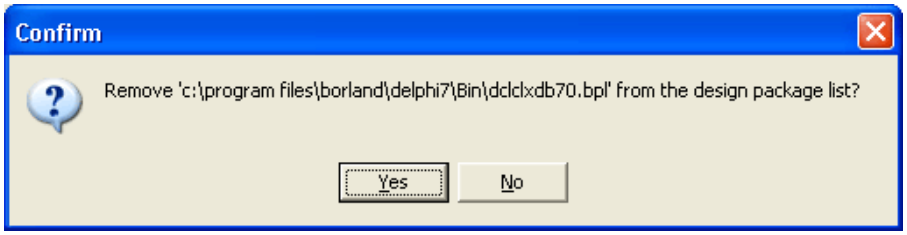


16.1-расм. Таҳрирлаш учун пакетлардан бирини танлаш

Компоненталар пакети муҳарририни ишга тушириш учун **Component** менюсидан **Install Packages** буйруғи танланади. Сўнгра очилган **Project Options** диалог ойнасидаги (16.11-расм) **Design packages** рўйхатидан ўчирилиши талаб қилинган пакетни танланади ва **Edit** тугмаси босилади. Натижада **Confirm** ойнаси (16.12-расм) очилади ва

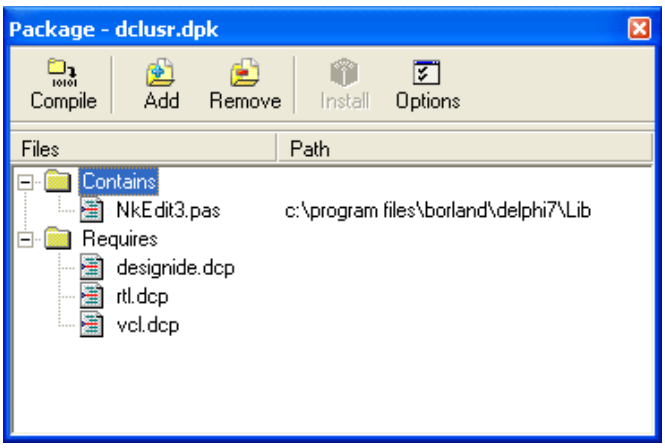
cancel this dialog box and open...
(бу диалог ойнасини ёпиш ва ... пакетини очиш)

сўровномасига **Yes** тугмасини босиб жавоб берилади.



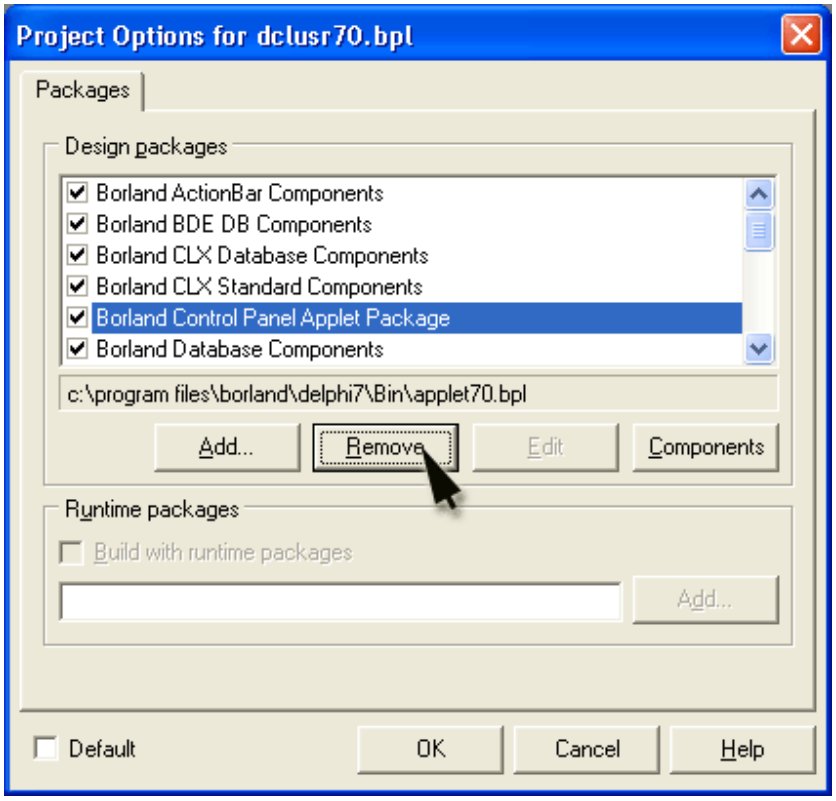
16.12-расм. Confirm диалог ойнаси

Шундан кейин, **Package** пакет муҳаррири ойнаси (16.13-расм) ойнаси экранда пайдо бўлади ва унинг **Contains** (Мундарижа) ойнасида шу пакетдаги компоненталар рўйхати берилади.



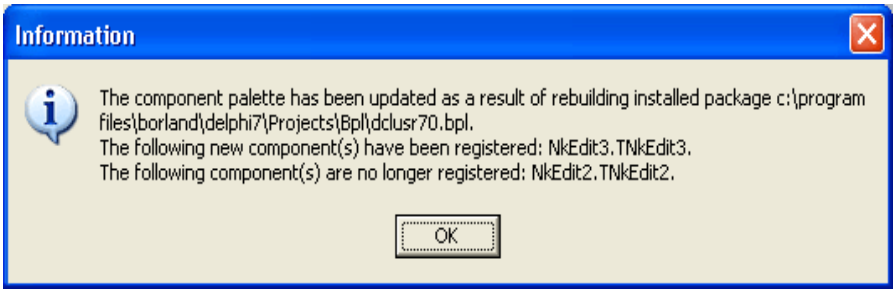
16.13-расм. Пакет муҳаррирининг ойнаси

Пакетдан бирор компонентани ўчириш учун **Remove** тугмаси босилади. Очилган **Remove From Project** диалог ойнасидан (16.14-расм) ўчирилиши талаб қилинган компонента танланади ва **ОК** тугмаси чертилади.



16.14-расм. Пакетдан ўчириладиган компонентани танлаш.

Компонента пакетдан ўчирилганидан сўнг, пакетни албатта қайта компиляция қилиш лозим. Бунинг учун пакет муҳаррири ойнасидан **Compile** тугмасини чертилади. Қайта компиляция ўтказилганидан сўнг, Delphi ўчирилган компонента бошқа қайд қилинмагалиги ҳақидаги ахборотни экранга чиқаради. (16.15-расм).



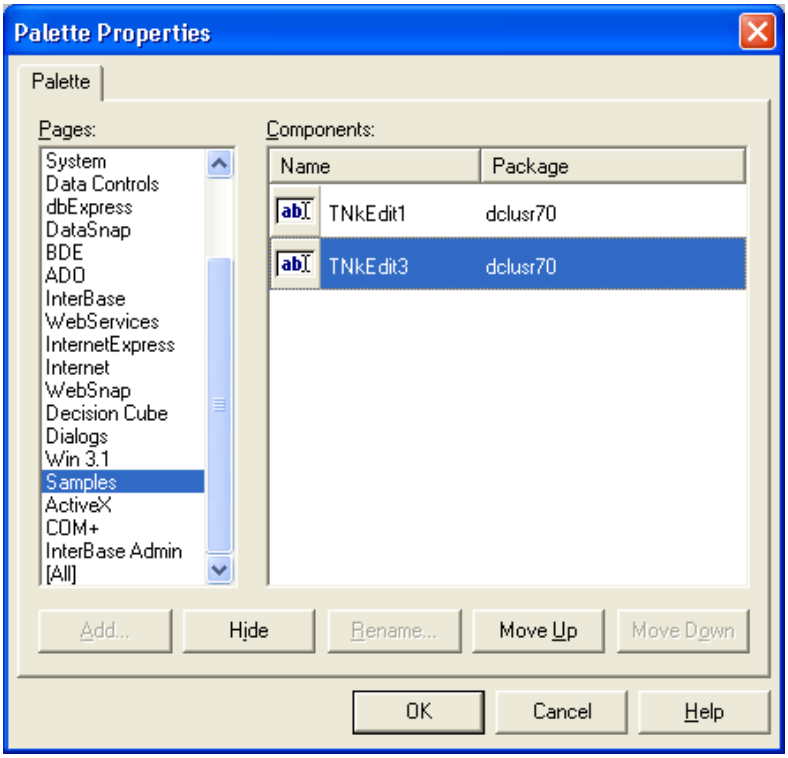
16.15-расм. Компонента ишламаётганлиги ҳақидаги ахборот

Қайта компиляция ўтказилганидан кейин, пакет муҳаррири ойнасини ёпиш ва очилган диалог ойнасида компонента ўчирилган пакетдаги ўзгаришларни сақлаб қўйиш лозим.

менюга

16.7. Компонента лар палитрасини созлаш

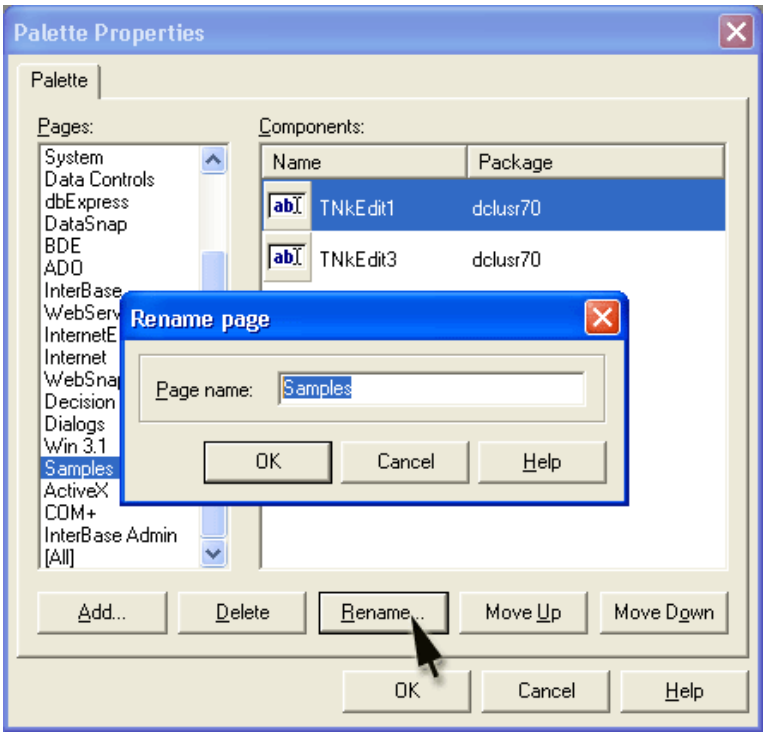
Delphi тили қуроллар панели тартибини ўзгартириш, уларнинг номларини алмаштириш, тугмаларнинг экранга чиқариш тартибини ўзгартиришга имкон беради. Бу созлаш ишларини **Palette Properties** диалог ойнасида бажарилади. У **Component** менюсидан **Configure Palette** буйруғи танланганда очилади. (16.16-расм).



16.16-расм. Palette Properties диалог ойнаси

Дастлаб, **Pages** рўйхатидан керакли компоненталар палитраси танланади. Сўнгра, қуроллар панели тартибини ўзгартириш учун **Move Up** ва **Move Down** тугмаларидан фойдаланиб, танланган қуроллар панели номини **Pages** рўйхатида лозим топилган ерга ўрнатилади. Компонента нишонлари тартибини ўзгартириш учун эса **Components** рўйхатидан керакли нишонни таналб олиб, **Move Up** ва **Move Down** тугмаларидан фойдаланган ҳолда бу нишонни янги жойга қўяди.

Қуроллар панели номини ўзгартиришга зарурат пайдо бўлиб қолса, **Pages** рўйхатидан керакли қуроллар панели номини танланади ва **Rename** тугмаси чертилади ҳамда **Rename page** диалог ойнасининг **Page name** майдонига (16.17-расм) янги ном киритилади.



16.17-расм. Rename page диалог ойнаси

менюга

мундарижага қайтиш

17-боб. ПРИНТЕР БИЛАН ИШЛАШ

Кўпинча дастурларни ишлаб чиқишда дастурчилар иловалар орқали олинган натижалар, матнлар ва экрандаги тасвирларни чоп қилиш муаммосига дуч келишади. DELPHI муҳити бу масалани ечиш учун етарлича содда имкониятларни ўз ичига олган.

Чоп қилиш механизми махсус printers. pas модули ёрдамида ташкил қилинади. Бунинг учун принтер билан ишлашга тўғри келадиган кодларнинг uses секциясига унинг номини қўшиб қўйиш етарли. Шундан кейин код принтер билан ишлашга тайёр бўлади.

Жорий компьютерга операцион система орқали ўрнатилган принтерлар билан ишлашни ташкил қилишда Tprinter классининг ҳусусият ва методларидан фойдаланилади. Бу класс ёрдамида кодларларда нафақат матнларни, балки растрли тасвирлар ҳамда график элементларни ҳам чоп қилиш мумкин.

17.1. Tprinter класси

Принтердан фойдаланишни кўзда тутадиган ҳар бир илованинг uses секциясига printers модулини киритиб қўйиш лозим. Шундан кейин иловани бажариш жараёнида тўғридан-тўғри дастурчиларга маълумотларни компьютерга уланган принтер орқали чоп қилиш учун зарур бўладиган барча воситаларни таклиф қилувчи Tprinter классининг экземпляри ҳосил қилинади. Шу класснинг энг муҳим ҳусусият ва методларини кўрайлик.

Tprinter классининг энг муҳим ҳусусият ва методлари -жадвал

ҳусусият ва методлар	Вазифаси
property Aborted: Boolean;	Чоп қилиш учун берилган топшириқ-нинг фойдаланувчи томонидан бекор қилинганлигини аниқлайди. Агар топшириқ бекор қилинган бўлса, унинг қиймати - True.
property Canvas: TCanvas ;	Принтернинг канваси. У фойдаланувчиларга тасвирларни чоп қилиш учун Brush, Font ва Pen ҳусусуиятларининг қийматларини белгилаш имконини беради.

TPrinterCapability=(pcCopies, pcOrientation, pcCollation); TPrinterCapabilities = set of TPrinterCapability;	Принтернинг жорий параметрларини ўрнатади: PcCopies - нусхалар сони, PcOrientation — қоғознинг ориентацияси.
property Copies: Integer;	Ҳужжатнинг керакли нусхалар сонини кўрсататди
property Fonts: TStrings ;	Жорий принтер учун ўрнатилган барча шрифтлар рўйхатидан иборат
property Handle: HDC;	қурилма контексти. Windows API функциялар программасидан мурожаат қилинади.
TPrinterOrientation = (poPortrait, poLandscape) ;	Қоғознинг ҳолати (ориентацияси) ни белгилайди: PoPortrait – вертикаль.
property Orientation: TPrinterOrientation;	PoLandscape — горизонталь
property PageHeight: Integer;	Саҳифанинг пикселлардаги баландлигини ифодалайди.
property PageNumber: Integer;	Жорий саҳифа номери
property PageWidth: Integer;	Саҳифанинг пикселлардаги кенглигини ифодалайди.
property Printerlndex: Integer;	Printers рўйхатида жорий принтер номининг индекси
property Printers: TStrings;	Система томонидан ўрнатилган принтерлар рўйхати
property Printing: Boolean;	Чоп қилиш жараёни тугамагунча True қийматини олади
property Title: String;	жорий принтернинг холатлар сатрида чоп қилинаётган хужжатни билдирувчи сатр
property Capabilities: TPrinterCapabilities;	PcCollation – нусхалар бўйича ажратиш параметрининг ишга тушганлигини аниқлайди
procedure Abort;	Бу метод чоп қилиш жараёнини тўхтади ва принтер навбатдаги топшириқни чоп қилишни бошлайди. Бунда Aborted ҳусусиятининг қиймати True бўлади.
procedure BeginDoc;	Навбатдаги чоп қилинадиган ҳужжатни бошлаш.
procedure EndDoc;	Ҳужжатни чоп қилишни тугатади ва принтер буферини тозалайди ва зарур бўлса, оҳирги саҳифани охиригача айлантиради. Бу методни чоп қилиш жараёни Abort методи билан тўхтатилмаган бўлса қўллаш мумкин.
procedure NewPage;	Чоп қилиш янги саҳифадан бошланишини таъминлайди. Бунда PageNumber ҳусусиятининг қиймати бирга орттирилади.

Бу ҳусусият ва методлар билан ишлаганда шуни ёдда тутиш керакки, Fonts, Handle, PageWidth, PageHeight, Aborted, Printing, Capabilities каби бир қатор ҳусусиятлар фақат ўқиш режимидагина ишлайди.

TPrinter класссини яратиш ва унинг экземплярига мурожжат қилишда Printers модулида эълон қилинган

function Printer: TPrinter;

функциясидан фойдаланилади. У класснинг яратилган экземпляри кўрсаткичини англатади.

Юқоридаги жадвалдан кўриниб турибдики, TPrinter класси нафақат жорий принтернинг турли параметрлари билан ишлашни таъминлайди, балки система томонидан ўрнатилган барча принтерларнинг

индексланган рўйхатини ҳам ташкил қилади. Бу рўйхатдаги ихтиёрий принтерни printerindex ҳусусияти орқали фаоллаштириш мумкин.

Чоп қилиш жараёнини бошқариш Abort, BeginDoc, EndDoc ва NewPage методлари ёрдамида амалга оширилади. Принтер канвасини ифодаловчи canvas ҳусусияти ҳам жуда муҳим. Унинг ёрдамида тасвирларни ҳосил қилиш ва принтерга узатиш ҳамда форма, ккомпонента каби маълумотларни чоп қилишни ташкил қилиш мумкин.

менюга

17.2. Матнларни чоп қилиш

Дастурчилик нуқтаи-назаридан DELPHI иловалари ёрдамида матнларни чоп қилиш механизми жуда ҳам содда. Бу иш Turbo Pascal даги каби амалга оширилади. Маълумотларни чоп қилиш амали чиқариш қурилмаси принтер бўлган write ҳамда writeln процедуралари ёрдамида амалга оширилади.

Энг содда масалалардан бири - принтер орқали “DELPHI дастурлаш тили” жумласини чоп қилиш масаласини кўрайлик. Бунинг учун янги консол илова яратамиз. Бунинг учун менюнинг **FileNewConsole application** тугмаларини чертамиз ва қуйидаги кодни ёзамиз: (uses секциясига Printers модулини қўшишни ёдда тутинг) :

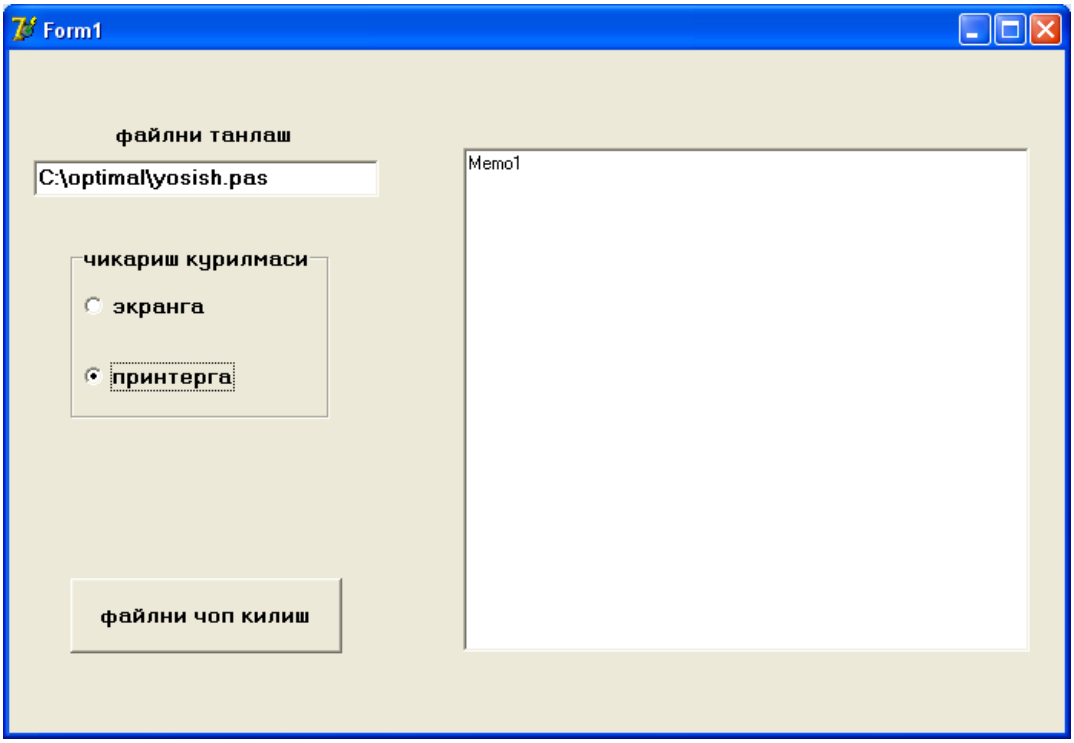
```
program Project1;
{$APPTYPE CONSOLE}
uses SysUtils, Printers;
var FPrint: TextFile;
begin
AssignPrn(FPrint);
Rewrite(FPrint);
Writeln(FPrint,' DELPHI дастурлаш тили ');
CloseFile(FPrint);
end.
```

Бу ерда Printers модулининг Assignprn процедураси матнли файлли ўзгарувчини системанинг жорий принтери билан боғлайди ва хотирада чоп қилинадиган маълумотлар учун буфер ҳосил қилади. Rewrite оператори чоп қилиш қурилмасини ишга туширади. Writeln процедураси матнни чоп қилади ва курсорни янги сатрнинг бошига ўтказади. CloseFile процедураси эса чоп қилиш жараёнини тугатиб, файлли ўзгарувчи ва принтер ўртасидаги алоқани узиб қўяди.

Битта write ёки writeln процедураси ёрдамида турли типдаги бир нечта маълумотларни чоп қилиш мумкин. Бунинг учун бу маълумотлар бир-биридан вергул белгиси билан ажратиб ёзилади. Чоп қилинадиган маълумотлар орасига сатрли TSrings ва TstringList каби объектларни ҳам жойлаштириш мумкин.

Намуна тариқасида кичик бир DemoPrint лойихаси кўрайлик. Унда диалог ойнаси ёрдамида чоп қилинадиган файл танланади ва бу файл RadioGroup ўчиргичларининг ҳолатига қараб экранга ёки принтерга узатилади.

Ушбу лойиҳа учун ташкил қилинган код листинги қуйидагича ёзилади:



1-расм. DemoPrint лойиҳасининг бош ойнаси

Листинг 17.1. DemoPrint лойиҳасининг листинги.

```
unit DemoPrint;
    unit Printp;
    interface
    uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  Dialogs, StdCtrls, ExtCtrls,
    Printers;
    type
      TForm1 = class(TForm)
        Edit1: TEdit;
        Label1: TLabel;
        RadioGroup1: TRadioGroup;
        Button1: TButton;
        Memo1: TMemo;
        procedure Button1Click(Sender: TObject);
      private
        { Private declarations }
      public
        { Public declarations }
      end;
    var
      Form1: TForm1;
    implementation
    {$R *.dfm}
    procedure TForm1.Button1Click(Sender: TObject);
      var f,printtxt: textfile;   S : string;
    begin
    if RadioGroup1.ItemIndex = 0
    then Memo1.Lines.LoadFromFile(edit1.text)
    else begin
    AssignFile(f, edit1.text);
    AssignPrn(PrintTxt);
    Reset(F);
    Rewrite(PrintTxt);
    while Not EOF(f) do
    begin Readln(f, s);
```

```
    Writeln(printtxt, s)
    end;
    CloseFile(f);
    CloseFile(PrintTxt);
    RadioGroup1.ItemIndex:= 0;
    end;
    end;
    end.
```

Ўйлаймизки, ушбу листинг ҳеч бир изоҳга муҳтож эмас. Фақат шуни айтиш мумкинки, листингдаги RadioGroup1.Itemlndex = 0;

кўрсатмаси маълумотлар принтер орқали чоп қилингандан сўнг, принтерни узиб қўяди.

Маълумотларни чоп қилиш жараёнида қуйидаги бошқарувчи белгилардан фойдаланиш мумкин:

- #9 — табуляция;
- # 10 —янги сатр;
- #13 — ENTER тугмаси;
- ^L — янги саҳифа.

Масалан,

```
Write(FPrint, 'Шу сатрдан кейин янги сатрга ўтилади', #10, #13);
Writeln (FPrint, 'Бу саҳифа оҳирига айлантирилади', ^L);
```

Матнли файлларни чоп қилишда тўғридан-тўғри 10 ўлчамли System шрифти ўрнатилади. Эҳтиёж пайдо бўлганда бу шрифт ва ўлчамни ўзгартириш мумкин. Бунинг учун Printer объектининг Canvas ҳусусиятидан фойдаланилади. Қуйидаги намунага эътибор беринг:

```
with Printer.Canvas.Font do
begin
Name := 'Bodo_uzb';
Size := 12;
end;
```

менюга

17.3. Тасвирларни чоп қилиш

Тасвирларни чоп қилишда Printers модулининг Tprinter классидаги TCanvas типидаги объектнинг Canvas ҳусусиятидан фойдаланилади. Бу класснинг канваси турли тасиврий элементларни ҳамда растрли график элементларни тасвирлаш учун етарлича бой имкониятларга эга. Биз бу имкониятлар билан 11-параграфда танишган эдик. Биз ушбу бобьда фақат ана шу тасвирларни чоп қилиш учун зарур бўлган маълумотлар билан танишамиз.

Тасвирларни канвадан принтерга узатиш жараёнининг ўзига ҳос томонлари фойдаланувчилар кўзидан яшириб қўйилган.

Тасвирларни чоп қилиш учун қуйидаги амаллар кетма-кетлигини бажариш лозим:

1. Чоп қилиш жараёни BeginDoc методи ёрдамида бошланади.
2. TCanvas классининг ҳусусият ва методлари ёрдамида canvas ҳусусиятида кўрсатилган жорий принтер учун яратилаётган тасвир ҳосил қилинади. Махсус методларга мурожаат қилинганида бу тасвир принтерга жўнатилади.
3. Чоп қилиш жараёнини EndDoc методи ёрдамида амалга оширилади.

Принтернинг канвасига мурожаат қилиш чоп қилиш жараёни бошлангандан сўнг бажарилишига эътибор беринг. Акс ҳолда бажариш вақтидаги ҳатолик юз беради.

Чоп қилинган маълумот ва унинг экрандаги кўриниши принтернинг параметрларига боғлиқ бўлади. Шунинг учун иложи борича чоп қилиш жараёнини бошлашдан аввал тасвирларнинг ўлчамларини аниқлаш ва зарур бўлса масштаблаштириш ишларини бажариш лозим. Масштаблаштириш амали фақат катта тасвирлар учунгина эмас, балки кичик расмларни қоғоз ўлчамларини ҳисобга олган ҳолда катталаштиришда ҳам қўлланади. Қоғознинг ўлчамларини Tprinter классининг PageHeight ва PageWidth ҳусусиятлари ёрдамида аниқлаш мумкин. Бундан ташқари, тасвирнинг экран ва қоғоздаги кўриниши принтер ва экраннинг руҳсат этилган имкониятларига қараб (разрешение) сифат жиҳатидан катта фарқ қилиши мумкин. Принтернинг жорий имкониятларини Windows API GetDeviceCaps функцияси ёрдамида аниқлаш мумкин. У масштабнинг бир дюйм масофага тўғри келадиган горизонтал ва вертикал йўналишлардаги нуқталар сонидан иборат қийматларни қабул қилади.

менюга

17.4. Растрли тасвирларни чоп қилиш

Растрли тасвирларни чоп қилиш учун уни TBitMap классидаги экземплярга юклаш лозим. Иловаларда Timage компонентасидан ёки мустақил яратилган TBitMap типидаги объектлардан фойдаланиш ҳам қулай ҳисобланади. Сўнгра тасвирни махсус методлар ёрдамида принтер канвасига жўнатиш мумкин ва Tprinter класси уни чоп қилинишини таъминлайди. Масалан, саҳифанинг марказида масштабни ўзгартирмай туриб чоп қилиш қуйидагича амалга оширилиши мумкин.

```
procedure TForml.ButtonlClick(Sender: TObject);
begin
with Printer, Imagel do
begin
if (Picture.Width > PageWidth) or(Picture.Height > PageHeight) then begin
ShowMessage('Тасвирнинг ўлчамлари саҳифадан катта ');
Exit;
end;
BeginDoc;
Canvas.Draw((PageWidth - Picture.Width) div 2,(PageHeight - Picture.Height) div 2, Picture.Bitmap);
EndDoc;
end;
end;
```

Агар тасвир қоғоздан кичик бўлса, Canvas.Draw методи уни саҳифанинг ўртасига чиқарилишини таъминлайди.

Нопропорционал бўлмаган масштаблаштириш амалини саҳифани ўлчамларини ҳисобга олган ҳолда қуйидагича амалга оширилиши мумкин:

```
procedure TFormI.ButtonlClick(Sender: TObject);
var ImageRect: TRect;
begin
with Printer, Imagel do
begin
ImageRect.Top := 0;
ImageRect.Left := 0;
ImageRect.Right :- PageWidth;
ImageRect.Bottom := PageHeight;
BeginDoc;
Canvas.StretchDraw(ImageRect, Picture.Bitmap);
EndDoc;
end;
end;
```

Бу ҳолда масштаблаштириш бўйича барча ишларни канванинг StretchDraw методи бажаради. У тасвирнинг ўлчамларини ImageRect тўғри тўртбурчаги томонларининг ўлчамларига мос равишда ўзгартиради.

Пропорционал масштаблаштириш учун тўғри тўртбурчак томонларининг ўлчамларини ҳисобга олган ҳолда элементар ҳисоб ишларини бажариш лозим.

менюга

17.5. Содда тасвирларни чоп қилиш

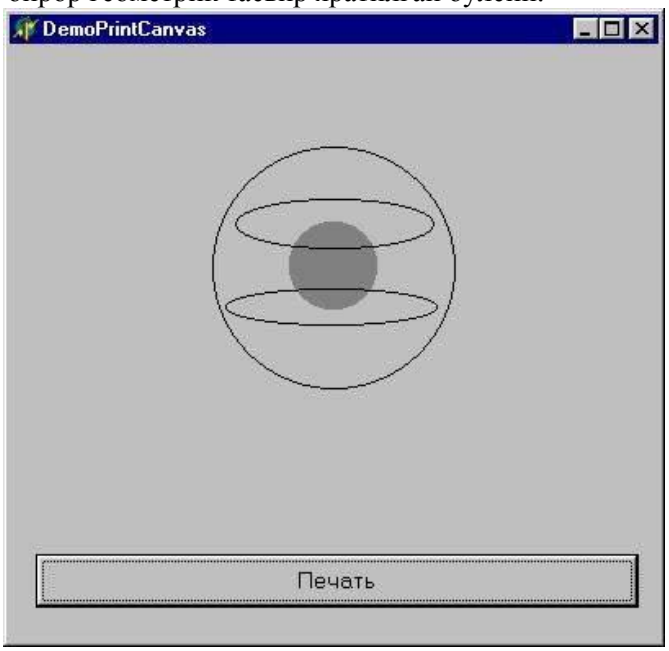
Ихтиёрий график фигура ва матнларни тасвирий элемент сифатида растрли тасвирларни яратмай туриб ҳам Printer объектининг Canvas ҳусусиятларидан фойдаланиб чоп қилиш мумкин.

```
procedure Tform1.Button1Click(Sender: TObject);
var R: TRect;
P: TPoint;
begin
Printer.BeginDoc;
with Printer.Canvas do begin Pen.Color := clBlack;
Brush.Color := clBlue;
Brush.Style := bsCross;
```

```
Font.Size := 14;
R := Rect( 10, 10, 160, 160);
Ellipse(R.Left, R.Top, R.Right, R.Bottom);
P := Point(60, 180);
TextOut (P.X, P.Y, 'Ellipse');
end;
Printer.EndDoc;
end;
```

Ушбу код бажарилганда, принтер ёрдамида остига “Ellips” сўзи ёзилган айлана тасвирир ҳосил қилинади. Бу ҳолда матнни график воситалардан фойдаланиб чоп қилинаётганлигига эътибор беринг.

TPrinter классининг Canvas ҳусусияти яна бир қизиқ имкониятни таклиф қилади: форманинг канвасидаги тасвирли ва матнли маълумотларни ортиқча қийинчиликларсиз чоп қилиш мумкин. Фараз қилайлик, форма канвасида бирор геометрик тасвир яратилган бўлсин.



2-расм. Форма канвасидаги геометрик тасвир

Уни ясашда TCanvas классининг ҳусусият ва методларидан фойдаланилади. DemoPrintCanvas иловасида (2-расм) тасвирни ҳосил қилиш учун OnPaint методидан фойланилган. Бу тасвирни чоп қилиш учун форма канвасидан керакли ўлчамдаги тўғри тўртбурчак ажратиб олинади ва уни принтер канвасига узатилади. 2-листингда бу вазифани printBtnclick методида CopyRect методи ёрдамида амалга оширилган.

Листинг .2 *DemoPrintCanvas лоёиҳасининг бош формасининг implementation секцияси*

```
implementation
{$R *.DFM)
uses Printers;
procedure TMainForm.FormPaint(Sender: TObject);
begin with Canvas do begin Pen.Color := clBlack;
Pen.Width :=2;
Brush.Color := clWhite;
Brush.Style := bsClear;
Font.Size := 14;
Ellipse( 10, 10, 160, 160 );
Brush.Color := clRed;
Ellipse( 60, 60, 110, 110);
Brush.Color := clBlue;
Brush.Style := bsFDiagonal;
Ellipse( 17, 35, 153, 75);
Brush.Color := clBlue;
Brush.Style := bsBDiagonal;
Ellipse(16, 85, 154, 135);
TextOut(50, 180, 'Эллипс');
```

```
end;
end;
procedure TMainForm.PrintBtnClick(Sender: TObject);
var ImageRect: TRect;
begin ImageRect := Rect(10, 10, 160 , 200);
with Printer do begin BeginDoc;
Canvas.CopyRect(ImageRect, MainForm.Canvas, ImageRect);
EndDoc;
end;
end;
end.
```

менюга

17.6. Форма ва бошқарув элементларини чоп қилиш

Ушбу пунктда биз жуда ҳам содда, аммо фойдали бўлган амал билан танишамиз. Бу амал форманинг экрандаги кўринишини қандай бўлса шундайича чоп қилишдан иборат.

Бу амални бажариш учун форманинг print методига мурожаат қилиш лозим. Бунда uses секциясига printers модулини қўшиш шарт эмас.

TForm классининг Print методининг ишлаш механизми қуйидача. У жорий ойнанинг клиент соҳасини экрандан ташқаридаги растрли тасвирга алмаштиради, сўнгра бу тасвирни Windows API нинг stretchDiBits функцияси ёрдамида принтерга узатади. Агар масштаблаштиришга эҳтиёж пайдо бўлса, уни

```
type TPrintScale қ (poNone, poProportional, poPrintToFit) ;
property PrintScale: TPrintScale;
```

ҳусусиятларидан фойдаланиб амалга ошириш мумкин. Бу ҳусусиятнинг қийматлари қуйидаги маъноларни англатади:

- PoNone — масштаблаштириш йўқ. Чоп қилинадиган тасвирнинг ўлчамлари принтернинг имкониятларига боғлиқ бўлади. Одатда қоғоздаги нусха оригиналнинг ўлчамларидан кичикроқ бўлади.
- poProportional — форма экрандаги ўлчамларига мос равишда чоп қилинади.
- poPrintToFit — масштаблаштириш қоғознинг ўлчамларин ҳисобга олиб, уни тўла эгаллайдиган тарзда амалга оширилади.

Форма ва TwinControl нинг ворислари бўлган бошқарув элементларини чоп қилишнинг бошқа усули ҳам мавжуд. Барча ойнали элементларнинг paintTo методи бўлиб, у «ўз тасвирини» ихтиёрий канвада ҳосил қилиши мумкин. Бизнинг ҳолимизда бу канва принтердан иборат бўлади.

менюга

17.7. Чоп қилишдаги ҳатоликларни назорат қилиш

Иловаларнинг принтер билан ишладаги ишончлилигини ошириш учун бевосита ҳатоликларни қайта ишлашнинг Exception базавий классидан келиб чиқадиган EPrinter классидан фойдаланиш мумкин. У стандарт усулда try ... except блогининг ичида кўрсатилади:

```
with Printer do try
BeginDoc;
Canvas.TextOut(100, 100, 'DELPHI tili' );
EndDoc;
except
on E: EPrinter do ShowMessage('чоп қилишдаги ҳатолик');
end;
```

менюга

мундарижага қайтиш

18-боб. МАЪЛУМОТЛАР БАЗАСИ

18.1. Бошланғич маълумотлар

Фойдаланувчи нуқтаи назаридан маълумотлар базаси (МБ) – бу компьютер хотира қурилмаларидан бирида сақланаётган маълумотлар билан ишлаш учун мўлжалланган дастурдир. Бундай дастурни ишга туширилганда, одатда жадвал ҳосил бўлади ва ундан фойдаланувчи ўзи узун зарур бўлган маълумотларни топишга харакат қилади. Агар система рухсат берса, у МБ га ўзгартиришлар киритиши, яъни янги маълумотларни киритиши, илгари киритилган эски маълумотлардаги ҳатоликларни бартараф қилиши ёки нокерак бўлган маълумолтларни ўчириши мумкин.

Дастурчи нуқтаи назаридан МБ – бу маълум бир мазмундаги маълумотларни ўз ичига олган файллар тўплами. Фойдаланувчи учун дастурчи МБ яратар экан, шундай дастур ёзадики, у маълумотларнинг файллари билан ишлашни таъминлай олади.

Ҳозирги кунда, локал Мб яратиш ва улардан фойдаланиш учун жуда катта сондаги дастурий системалар мавжуд: dBASE, FoxPro, Access, Paradox, Interbase, Oracle, Sysbase, Infomix, Microsoft SQL Server ва х.к.

Delphi таркибига турли системалар ёрдамида яратилган маълумотларнинг файллари (МФ) билан ишлаш учун етарли компоненталар киритилган. Шунингдек, Delphi дастурчига Borland Database Desktop утилитдан фойдаланиб, турли форматдаги МБ файлларини яратишга имкон беради.

менюга

18.2. Маълумотлар базасининг классификацияси

МБ маълумотлардан фойдаланувчи дастурларнинг ҳолати, маълумотларнинг ўзи ҳамда бир нечта фойдаланувчилар ўртасида маълумотларни тақсимланишига қараб локал ва узоқлаштирилган МБ ларга бўлинади.

Локал маълумотлар базаси. Локал МБ нинг маълумотлари (МФ) битта (локал) қурилмада жойлашади Бу қурилма сифатида компьютер диски ёки тармоқ диски (тармоқда ишлаётган бошқа компьютер диски) ни олиш мумкин.

Маълумотларни фойдаланувчилар сифатидаги бир ёки бир нечта компьютерларда ишлаётган дастурлар ўртасида тақсимлаш (маълумотлар билан ишга рухсат бериш) учун локал МБ ларда файлларни тўсиш деб аталадиган методи қўлланадаи. Бу методнинг моҳияти шундаки, битта фойдаланувчи МБ дан фойдаланаётган вақтда қолган фойдаланувчилар бу маълумотлардан фойдалана олмайди, яъни маълумотлар улардан тўсиб қўйилади.

Локал Мб ларга мисол қилиб Paradox, dBase, FoxPro ва Access дастурий воситаларини олиш мумкин.

Узоқлаштирилган маълумотлар базаси. Узоқлаштирилган маълумотлар базасидаги маълумотлар (файллар) узоқлаштирилган компьютерларда жойлашади. (Уларнинг каталогларига тармоқ дисклари сифатида қараб бўлмайди.)

Узоқлаштирилган маълумотлар базаси билан ишлаш дастурлари икки қисмдан иборат бўлади: мижоз ва сервер қисмлари. Фойдаланувчининг компьютерида ишлаётган дастурнинг мижоз қисми сервер дастур билан ўзаро алоқани таъминлайди: узоқлаштирилган компьютерга узатиладиган сўровномалар воситасида маълумотлар билан ишлашга рухсат берилади.

Узоқлаштирилган компьютерда ишлаётган дастурнинг сервер қисми сўровномаларни қабул қилади, уларни бажаради ва маълумотларни мижоз дастурига жўнатади. Сўровномалар SQL (Structured Query Language) — структуралаштирилган сўровномалар тилида ёзилган буйруқлардан иборат бўлади.

Узоқлаштирилган компьютерда ишлаётган дастур шундай тузиладики, бир вақтнинг ўзида бир нечта фойдаланувчига маълумотлар билан ишлашга рухсат берилади. Маълумотлар билан ишлаш учун блокировка (тўсиқ қўйиш) механизми ўрнига транзакциялар механизми қўлланади.

Транзакция — бу маълумотларни узатилишидан аввал, шу узатиладиган маълумотлар устида албатта бажарилиши шарт бўлган айрим амаллар кетма-кетлигидир. Бирор амални бажариш жараёнида ҳатолик учраб қолса, транзакцияни ташкил қилувчи барча амаллар кетма-кетлиги қайтадан бажарилади. Шундай қилиб, транзакциялар механизми аппарат бузилишларидан ҳимояни таъминлайди. Шунингдек у маълумотларга кўп фойдаланувчилар томонидан бир вақтда мурожаат қилинишини ҳам таъминлайди.

Узоқлаштирилган МБ дастурларини ишлаб чиқиш жуда мураккаб, оғир ва кўп меҳнат талаб қиладиган масала ҳисобланади. Уни ҳал қилиш учун дастурчи катта малакага эга бўлиши шарт. Шунинг учун, узоқлаштирилган маълумотлар базаларини ишлаб чиқиш масаласи ушбу китоб доирасида кўрилмайди.

менюга

18.3. Маълумотлар базасининг структураси

МБ — бу одатда бир жинсли, бирон бир критерия бўйича тартибланган маълумотлар тўпламидир. МБ ни қоғоздаги ёки компьютердаги вариант кўринишида ифодалаш мумкин.

Қоғоздаги вариантига мисол қилиб кутубхона каталоги – ўз ичига китоблар ҳақидаги маълумотларни жамлаган қоғоз карточкалар тўпламини олиш мумкин. Бу базадаги маълумотлар бир жинсли, яъни фақат китоблар ҳақидаги маълумотларни сақлайди. Бу катрочкалар бирор бир тартиб билан, масалан, китоб номларини муаллифларнинг фамилияларини алфавит бўйича тартиблаган ҳолда сақлайди. МБ га бошқа мисол қилиб, телефон абонентлари бўйича маълумотнома, поездлар харакатининг жадвали кабиларни ҳам олиш мумкин.

МБ нинг компьютер вариантида маълумотлар файлларда (ёки файллар тўпламида) сақланади. Бу МБ лар ёзувлардан ташкил топади. Ҳар бир ёзув битта экземпляр ҳақидаги маълумотларни сақлайди. Масалан, "Ўқув Юрти Талабалари" МБ даги ҳар бир ёзув битта экземпляр, яъни битта талаба ҳақидаги маълумотни сақлайди. Ёзувлар майдонлардан иборат бўлади. Ҳар бир майдон экземплярнинг фақат битта характеристикасини сақлаш учун хизмат қилади. Масалан, "Ўқув Юрти Талабалари" МБ қуйидаги майдонлардан иборат бўлиши мумкин: Фамилияси, исми, факультети, гуруҳи, туғилган санаси. Бу майдонлардаги маълумотлар биргаликда битта талаба ҳақидаги маълумотларни сақлайди.

Шунга эътибор бериш керакки, хар бир ёзув бир ҳилдаги майдонлардан иборат бўлади. Айрим бир майдонларга маълумотлар ёзилмаган бўлиши ҳам мумкин. Аммо, уларнинг ҳаммасида маълумотлар мавжуд деб қаралади.

Қоғоздаги МБ ни жадвал кўринишида ифодалаш қулай. (18.1-расм). Жадвалнинг ҳар бир сатрига битта ёзув, ячейкага эса майдон мос келади. Бунда жадвал устунининг номи майдон номига, сатр номери эса ёзув номерига мос келади.

Одатда компьютер МБ сидаги маълумотларни экранга жадвал кўринишида чиқарилади. Шунинг учун адабиётларда "Маълумотлар файли" жумласи ўрнига "маълумотлар жадвали" ёки оддий қилиб, "жадвал" сўзлари қўлланиши мумкин.

Фамилияси	Исми	факультети	Гуруҳи	Туғилган санаси
Абдуллаев	Илхом	математика	103	05.06.85

Ботирова	Клара	физика	201	13.11.86
Даминов	Содиқжон	физика	402	24.09.84
Комилова	Гулчеҳра	Химия	302	16.03.85
Салимова	Гавҳарой	География	105	09.08.86
Турсунов	Абдулбоқи	математика	401	07.07.84

18.1-расм. МБ ни жадвал кўринишида ифодалаш

менюга

18.4. Маълумотлар базасининг Delphi даги модели

Ҳар бир жадвал алоҳида файлда сақланади. Аммо, МБ билан жадвални тенглаштириш ўринли эмас. Чунки, МБ да битта ёзувнинг майдонлари бир нечта жадваллар бўйлаб тарқалиб кетган бўлиши мумкин. Демак, битта МБ бир нечта файлларда сақланиши мумкин.

Энг содда ҳолда, МБ билан ишлаётган дастур учун маълумот манбаси жадвал бўлиши мумкин. Аммо, одатда фойдаланувчини МБ даги барча маълумотлар эмас, балки унинг маълум бир қисми қизиқтиради. У МБ дан ўзининг сўровига кўра айрим маълумотларни танлаб олади ва кўради. Шунинг учун МБ билан ишлашга мўлжалланган дастурларга Мб доирасидаги турли сўровномаларни ҳосил қилиш ва бу сўровномаларга жавоблар ахтариш имкониятлари ҳам киритилади.

Маълумотлар базасининг тахаллуси. Дастурчи МБ учун дастур ёзар экан, МБ файллари қайси дискда ёки қайси каталогда жойлашишини билмаслиги мумкин. Масалан, дастурчи МБ файлларини C:, D: ёки тармоқ дискларидан бирида ташкил қилиши мумкин. Шунинг учун дастурга МБ файлларининг жойлашуви ҳақидаги маълумотларни узатиш муаммоси пайдо бўлади.

Delphi да дастурга МБ файлларининг жойлашуви ҳақидаги маълумотни бериш масаласи МБ лар тахаллусидан фойдаланиб ҳал қилинади. Тахаллус (Alias) — бу МБ нинг тўла манзили ва номига мос келувчи қисқартирилган сўздир. Масалан, C:/OUY/Oquv_Yurti файлига Talaba деган ном бериш мумкин. МБ лар билан ишлаганда, файлнинг манзили ва тўла номи ўрнига, унинг тахаллусидан фойдаланилади.

Маълумотлар билан ишлаш учун, МБ билан ишлашни таъминловчи дастур Borland Database Engine (BDE) кутубхонасини ишга солади. Бу кутубхона ўз навбатида системада қайд қилинган барча тахаллуслар ҳақидаги маълумотларни ўз ичига олган файлдан фойдаланади.

МБ нинг тахаллусини BDE Administrator утилити ёрдамида яратилиши (қайд қилиниши) мумкин. Шу утилитнинг ўзи тахаллус билан боғланган каталогни ўзгартиришга имкон беради.

менюга

18.5. Маълумотлар базасини яратиш

МБ – бу маълумотлар сақланаётган файллар (жадваллар) тўпламидир. Одатда, МБ битта каталогда жойлашган бир нечта жадваллардан ташкил топади. Янги МБ учун каталогни одатдаги усуллар билан, масалан, Проводник ёрдамида яратилиши мумкин. Жадвалларни эса Delphi таркибидаги Borland Database Desktop утилити ёрдамида ёки МБ серверига SQL-сўровномалар ташкил қилиб яратиш мумкин.

МБ файллари (жадваллари) даги маълумотлар билан ишлаш учун BDE кутубхонаси файллар жойлашган каталог номидан эмас, балки унинг тахаллусидан фойдаланади. Шунинг учун, янги МБ жадвалини ҳосил қилишдан аввал, шу МБ учун тахаллус ўйлаб топилади. Шундай қилиб, янги МБ яратиш жараёнини қуйидаги учта босқичдан иборат деб қараш мумкин:

1. каталог яратиш;
2. тахаллус ўйлаб топиш;.
3. жадвал яратиш.

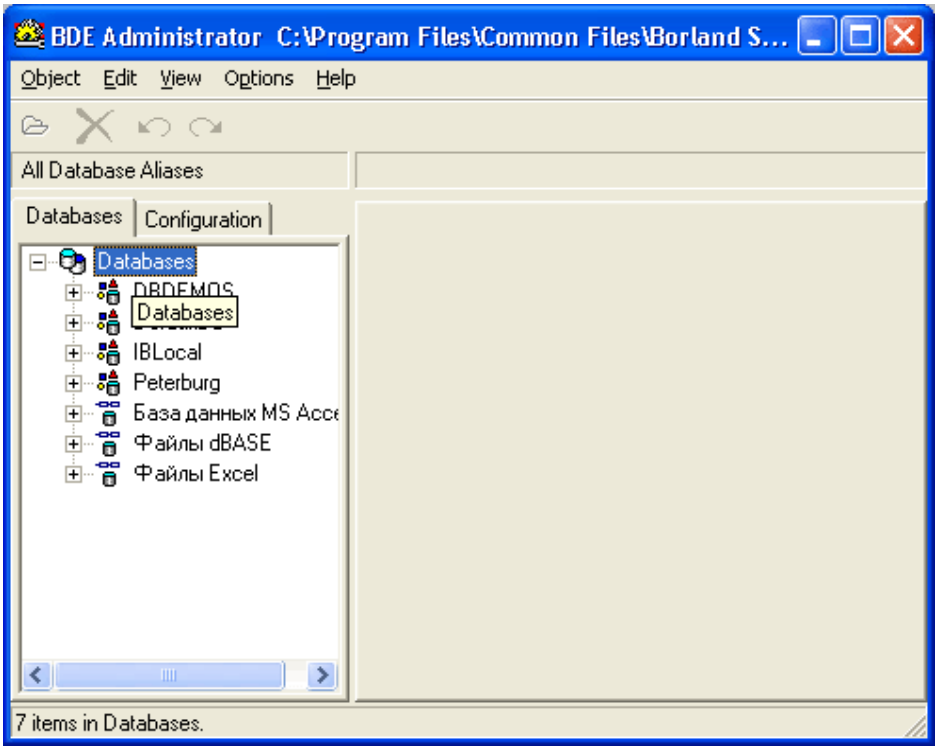
Каталог яратиш. Янги МБ файллари (жадваллари) учун каталогни (папкани) одатдаги усуллар билан, масалан, Проводник ёрдамида яратилиши мумкин. Одатда, локал МБ файллари МБ билан ишлаш учун ёзилган дастур жойлашган каталогнинг қисм каталогида жойлаштирилади.

Эслатма: Биз бундан кейин Uquv_Yurti МБ файлини C: дискнинг OUY каталогида сақланаётган деб қабул қиламиз.

Тахаллус яратиш. Мб нинг тахаллусини Delphi таркибига кирувчи BDE Administrator утилити ёрдамида яратилади. Бу утилит Windows муҳитидан **Программў / Borland Delphi** 7 менюсидан **BDE**

Administrator буйруғи билан ишга туширилади.

BDE Administrator нинг диалог ойнаси 18.2-расмда келтирилган.



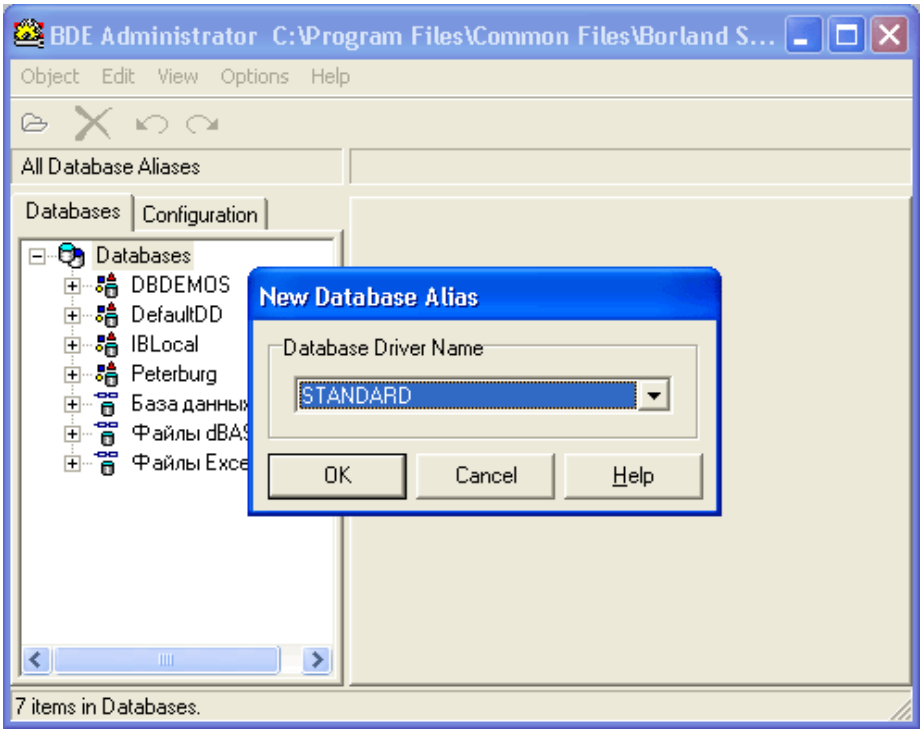
18.2-расм. BDE Administrator диалог ойнаси

Ойнанинг чап томонида, **Databases** пунктида шу компьютерда қайд қилинган тахаллуслар рўйхати берилган. Янги таҳаллусни яратиш учун **Object** менюсидан **New** буйруғи таналанади. Сўнгра очилган **New Database Alias** (МБ нинг янги тахаллуси) менюсидаги МБ билан ишлаш учун системада қайд қилинган драйверларнинг кўрсатилган **Database Driver Name** рўйхатидан яратилаётган МБ учун драйвер танланади (18.3-расм), яъни яратилаётган МБ нинг типи белгиланади.

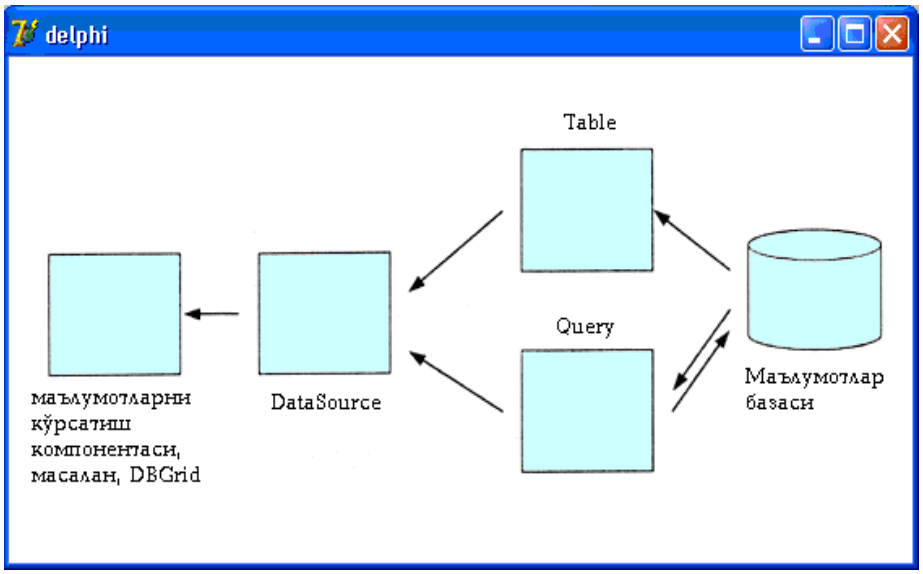
Тахаллус яратишда, агар кўрсатилмаган бўлса, Paradox форматидаги жадваллар бир ишлашни таъминлайдиган **STANDARD** (default driver) драйверидан фойдаланилади.

Драйвер танланиб, ОК тугмаси чертилганидан сўнг, тахаллуслар рўйхатига янги элемент қўшилади. (18.4-расм).

Шундан кейин, администратор томонидан яратилган янги таҳаллусни таҳрирланади, яъни унинг номи ва у англатиб турган МБ файлининг йўли ўзгартирилади.



18.3-расм. New Database Alias диалог ойнаси



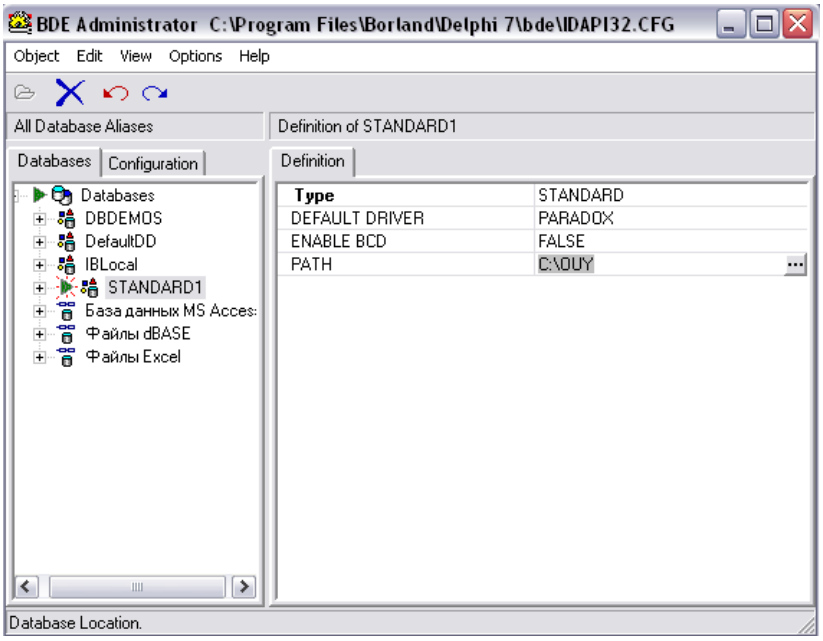
18.4-расм. Янги таҳаллусни қайд қилиш

Таҳаллусларни Windows учун оддий усул билан ўзгартириш мумкин: сичқончанинг ўнг тугмасини таҳаллус номида чертилади. Очилган контекст менюсидан **Rename** (қайта номлаш) буйруғи танланади ва очиладиган диалог ойнасида таҳуллуснинг янги номи киритилади.

МБ файлларига йўлни **Definition** пунктидаги **Path** майдонида клавиатура ёрдамида ёки стандарт **Path** майдонининг охирида турган учта нуқтали тугмани босиб очиладиган **Select Directory** (каталогни танлаш) диалог ойнасидан кўрсатилиши мумкин.

Намуна сифатида 18.5-расмда Uquv_yurti МБ файли учун Talaba таҳаллуси яратилганидан кейинги **BDE Administrator** ойнаси келтирилган.

Яратилган таҳаллус конфигурация файлида (Idapi.cfg) қайд қилиниши учун **Object** менюсидан **Apply** (қўллансин) буйруғи танланади. Очилган **Confirm** диалог ойнасида киритилган ўзгартиришларни конфигурация файлида сақлаб қўйиш яна бир марта тасдиқланади.



18.5-расм. Таҳаллусни сақлаш натижаси.

менюга

18.6. Жадвал яратиш

МБ яратишда энг муҳим вазифалардан бири маълумотларни ёзувнинг майдонлари орасида тақсимлаш масаласидир. Майдонлар ўртасида маълумотларни турли усуллар билан тақсимлаш мумкинлиги табиий. Масалан, талабалар ҳақидаги маълумотларни ўз ичига олган маълумотлар базаси "фамилия", "исм", "факультет", "гуруҳ" ва "туғилган санаси" майдонларидан иборат бўлган ёзувлар шаклида сақлаш ташкил қилинган бўлсин.

Агар МБ дан фойдаланишда бирор критерия бўйича танлаш масаласи ҳам назарда тутилган бўлса, шундай танловни таъминловчи маълумотларни алоҳида майдонларда сақлаш тавсия қилинади.

Ёзувнинг майдонлари рўйхати аниқланганидан сўнг, майдонларни жадвалларга тақсимлаш лозим. Битта содда МБ да ҳамма маълумотларни битта жадвалга киритиш мумкин. Мураккаб МБ ларда эса маълумотлар бир нечта жадваллар бўйлаб тақсимланади ҳамда бу жадваллар ўртасидаги алоқани белгиловчи қўшимча маълумот ҳам тайинланади.

Эслатма: Бир неча жадвалларда жойлашиб, бир-бири билан боғланган МБ ларни релацион МБ дейилади. Реляцион МБ ларда жадваллардаги айрим маълумотларни такрор ва такрор учрамаслиги учун ёзувларни бир қийматли қилиб тенглаштириш мақсадида қўшимча хизматчи маълумотлар қўшилади.

МБ ёзувларининг структураси аниқланганидан сўнг, энди жадвал яратишга ўтиш мумкин. Жадвалларни Delphi таркибига кирган Database Desktop утилити ёрдамида ташкил қилиш мумкин.

Database Desktop утилити МБ билан ишлашда зарур бўладиган барча амалларни бажаришга имкон беради. У МБ ни яратиш, кўриш, таҳрирлаш, турли форматдаги (Paradox, dBASE, Microsoft Access) МБ лар билан ишлаш каби вазифаларни бажара олади. Бундан ташқари, бу утилит сўровномалар ёрдамида МБ дан маълумотларни танлаш имкониятига ҳам эга.

Янги жадвал яратиш учун **Tools** менюсидан **Database Desktop** буйруғи танланади. Database Desktop утилити ишга тушади. Сўнгра, унинг диалог ойнасидаги **File** менюсидан **New** буйруғи танланади ва очилган рўйхатдан яратилаётган файлнинг типи — **Table** белгиланади. Навбатдаги очилган **Create Table** диалог ойнасидан яратилаётган жадвалнинг типи кўрсатилади. Агар бу тип кўрсатилмаса, у Paradox 7 типи деб қабул қилинади. Натижада **Create Paradox 7 Table** диалог ойнаси очилади ва энди унга жадвал структурасини киритиш мумкин.

Жадвалнинг ҳар бир майдони учун ном, тип ва зарур бўлса ўлчам бериш керак. Майдон номи шу майдон элементлари билан ишлаганда фойдаланилади. Майдон номларини **Field Name** устунига ёзилади. Номларни ёзишда лотин ҳарфлари ва рақамлардан иборат ҳамда умумий узунлиги 25 дан кўп бўлмаган белгилар кетма-кетлигидан фойдаланиш мумкин.

Майдоннинг типи шу майдонга ёзиладиган маълумотларнинг типини аниқлайди ва Type устунига

махсус белгили константа киритиш билан эълон қилинади. 18.1-жадвалда майдонларнинг типлари ва уларга мос белгили константалар рўйхати келтирилган.

Майдон типлари ва уларга мос константалар. 18.1-жадвал

Тип	**Конс-танта**	**Майдондаги маълумотлар**
Alpha	A	Белгилар сатри. Сатрнинг максимал узунлиги Size билан белгиланади ва 1 дан 255 гача диапазонда бўлиши мумкин
Number	N	10^{-307}... 10^{308} диапазондаги ва 15-та ишончли рақами бўлган сон
Money	$	Пул форматидаги сон. Соннинг рақамлари разрядларни ажратувчи белги билан гуруҳларга бўлинади. Шунингдек пул белгиси ҳам чиқарилади.
Short	S	-32767...32767 диапазондаги бутун сон
Long Integer	I	-2 147 483 648...2 147 483 647 диапазондаги бутун сон
Date	D	Сана (Дата)
Time	T	Ярим тундан бошлаб, ўтган миллисекунддаги вақт
Time stamp	@	Вақт ва сана
Memo	M	Ихтиёрий узунликдаги сатр. Бу тип Alpha типида сақлаш мумкин бўлмаган матнли маълумотларни сақлаш учун хизмат қилади. Майдоннинг (1—240) ўлчами жадвалда матннинг қанча белгиси сақланишини кўрсатади. Қолган матн эса жадвал номи билан бир хил номдаги, .mb кенгайтмали файлда сақланади.
Formatted Memo	F	Ихтиёрий узунликдаги матн.(Memo каби). Шрифтнинг типи, ўлчами, стили ва белгиларининг рангини кўрсатиш мумкин.
Graphic	G	Графика
Logical	L	Мантиқий қиймат: "Рост" (True) ёки "ёлғон" (False)
Auto-increment	+	Бутун сон. Жадвалга янги ёзув қўшилганда бу майдонга аввалги ёзувнинг шу майдонида турган сонга бир сонини қўшиб, ёзилади.
Bytes	Y	Иккилик саноқ системасидаги маълумотлар. Бу типдаги маълумотлар Database Desktop қайта ишлаш олмайдиган сонларни ёзиш учун ишлатилади.
Binary	B	Иккилик саноқ системасидаги маълумотлар. Бу типдаги маълумотлар Database Desktop қайта ишлаш олмайдиган сонларни ёзиш учун ишлатилади. Memo типи каби бу тип маълумотлари ҳам жадвалнинг файлида сақланмайди. Одатда, Binary типидаги майдонлар audio маълумотларни сақлайди.

Типни кўрсатувчи белгили константа клавиатурадан киритилиши ёки **Type** устуни чертилганда очиладиган рўйхатдан майдон типини белгилаш орқали кўрсатилиши мумкин.

Бир ёки бир нечта майдонларни асосий (ҳал қилувчи ёки калит) майдон тарзида белгилаш мумкин. Бу майдон маълумотларнинг мантиқий тартибини белгилайди. Масалан, Fam майдони (Alpha типи) асосий майдон сифатида белгиланган бўлса, жадвал элементларини бу майдон бўйича алфавит тартибида

тартибланган ҳолда чиқарилади. Агар асосий майдонлар бўлмаса, маълумотлар жадвалга киритилган тартибида чиқарилади. Шуни ёддас ақлаш керакки, асосий майдонларда бир ҳил маълумотли иккита ёзувнинг бўлиши мумкин эмас. Шунинг учун биз кўраётган мисолда асосий қилиб Fam (фамилияси) ва Ism (исми) майдонларини белгилаш мумкин. Аммо, бу ҳолда исми ва фамилияси бир ҳил бўлган маълумотларни жадвалга киритиб бўлмайди. Ана шундай ҳолатларнинг олдини олиш учун асосий майдон қилиб шу майдондаги бошқа маълумотлоар билан устма-уст тушмайдиган маълумотлар сақланадиган майдонларни асосий қилиб белгиланади. Масалан, одамлар ҳақидаги маълумотлар учун мўлжалланган жадвалга Pasp (Паспорт) майдонини киритиб, уни асосий қилиб белгилаган маъқул.

Майдонни асосий деб белгилаш учун **Key** устунини икки марта чертилади. Иложи борича, асосий майдонларни жадвалнинг юқори қисмига жойлаштириш керак.

Ёзувларнинг айрим майдонлари бўш бўлиши мумкин. Бўш бўлмаслиги шарт бўлган майдонлар учун албатта **Required Field** байроқчасини ўрнатиб қўйиш лозим. Масалан, Fam (фамилияси) майдони бўш бўла олмайди. Бу вақтда Tel (Телефон) майдони бўш бўлиши мумкин.

Агар майдонга киритилган маълумотлар маълум бир диапазонда бўлиши талаб қилинса, **Minimum value** (Минимал қиймат) ва **Maximum value** (Максимал қиймат) майдонларига зарур қийматларни киритиб, диапазон чегарасини белгилаб қўйиш мумкин.

Default value майдони, агар шу майдонга қиймат киритилмаса, унинг қиймати тўғридан-тўғри қандай бўлиши кераклигини кўрсатади. Бу қиймат жадвалга янги ёзув қўшилганда, майдонга автоматик тарзда ёзиб қўйилади.

Picture майдони майдонга киритилаётган маълумотларни тўғрилигини махсус шаблон билан назорат қилиб туриш учун мўлжалланган. Бу шаблон оддий ва махсус белгилар кетма-кетлигидан иборат бўлади. Махсус белгилар 18.2-жадвалда кўрсатиб ўтилган. Махсус белги турган майдонга фақат шу белгига мос келадиган белгиларнигина киритиш мумкин. Масалан, шаблон позициясида # белгиси турган бўлса, бу белгига мос позицияга фақат рақамни киритиш мумкин, бошқа белгиларни компьютер инкор қилади. Агар шаблон майдонида оддий белги турган бўлса, у ҳолда шу майдонга маълумотлар киритилганда шу позицияда автоматик тарзда кўрсатилган белги қўйилади. Масалан, Tel майдони А типида (белгилар сатри) телефон номерларини сақлаш учун мўлжалланган бўлиб, шу МБ билан ишлайдиган дастурда ҳам оддий кўринишда ифодаланади, яъни гуруҳлари бир-биридан тире билан ажратилади деб қабул қилсин. Бу ҳолда **Picture** майдонига ###-##-## шаблонини қўйиш керак. Tel майдонига маълумотларни киритишда фақат рақамлар қалул қилинади ва чиқарилади (қолган барча тугмалдар инкор қилинади), шунингдек, учинчи ва бешинчи рақамлардан кейин тире белгиси автоматик тарзда майдонга қўйилади.

Шаблонлар учун махсус белгилар 18.2-жадвал

Белги шаблони	Киритишда мумкин бўлган белгилар
*	Ихтиёрий рақам ва ҳарфлар. Ихтиёрий ҳарф (автоматик тарзда катта ҳарфга алмаштирилади)
&	Ихтиёрий белги. Агар ҳарф киритилган бўлса, у автоматик тарзда катта ҳарф билан алмашади)
@	"нуқтали вергул" дан кейин келаётган белгини шаблон белгиси эмас, балки оддий белги деб қабул қилинади.
*.	"*" дан кейин келган шаблон белгиси билан аниқланадиган иҳтиёрий марта такрорланадиган белгилар.

Майдондаги маълумотларнинг айрим элементларини кўрсатиш шарт эмас, масалан, телефон МБ си учун шаҳарларнинг коди мажбур эмас. Майдонларга киритилиши шарт бўлмаган маълумотларнинг шаблонлари квадрат қавслар ичида кўрсатилади. Шунинг учун [(###)]###-##-## шаблони телефон номерларини шаҳарларнинг коди билан ҳам, кодисиз ҳам киритишга имкон беради.

Шаблонлар нафақат киритилаётган маълумотларни назорат қилади, балки маълумотларни киритилишини ҳам автоматлаштира олади. Бу вазифа шаблондаги квадрат ёки фигурали қавслар ичида майдоннинг мумкин бўлган қийматлари рўйхатини кўрсатиш орқали ҳал қилинади. Масалан, fak (факультет) майдони учун [физика, математика, география, химия]*@ ёки {биология, педагогика, ўзбек тили}*@ тарзида шаблонлар эълон қилинган бўлса, Бу майдонга маълумот киритишда мос факультет номининг биринчи ҳарфини (ф, м, г, х ҳарфларидан бирини) киритиш етарли, факультетнинг номи бирданига майдонда пайдо бўлади. Бу шаблонларнинг фарқи шундаки, фигурали қавсли шаблонда

майдонга киритиладиган маълумот рўйхатдаги факультетларнинг бирининг номидан иборат бўлиши керак, квадрат қавсли шаблонда эса факулътет бошқача аталиши мумкин, аммо унинг номини тўла киритиш шарт.

Жадвалнинг структураси аниқланганидан сўнг, жадвални албатта сақлаб қўйиш шарт. Бунинг учун **Save As** тугмаси босилади. Натижада экранда **Save Table As** диалог ойнаси очилади. Шу ойнадаги **Alias** рўйхатидан МБ таҳаллусини танлаш лозим. **Имя файла** майдонида эса яратилган жадвални сақлаш керак бўлган файлнинг номи кўрсатилади.

Агар **Сохранить** тугмасини босишдан аввал **Display table** байроқчаси ўрнатилган бўлса, **Сохранить** тугмаси босилганидан кейин **Table** диалог ойнаси очилади ва унга ҳозиргина яратилган жадвалга маълумотларни киритиш мумкин бўлади.

Агар МБ жадвали ёпиқ бўлса, бу жадвалга маълумотларни киритиб бўлмайди. Маълумотларни киритиш учун уни очиш керак. Бунинг учун **File** менюсидан **Open / Table** буйруғини танланади, сўнгра очилган **Open table** диалог ойнасининг **Alias** рўйхатидан керакли МБ ва жадвалнинг таҳуллусини кўрсатилади. Бунда МБ фақат кўриш режимида ишлаётган бўлади, яъни МБ га ўзгартириш киритиб бўлмайди. МБ жадвалига маълумотларни киритиш ва таҳрирлаш имкониятига эга бўлиш учун жадвални таҳрирлаш режимини фаоллаштириш лозим. Бунинг учун **Table** менюсидан **Edit Data** буйруғини танланади.

Маълумотлар ёзув майдонларига оддий усул билан, клавиатурадан фойдаланиб киритилади. Навбатдаги майдонга ўтиш учун <Enter> клавишаси босилади. Агар майдон ёзувнинг охирги майдони бўлса, <Enter> клавишаси босилганидан кейин жадвалга янги ёзув қўшилади.

Агар жадвалнинг тўлдириш вақтида маълумот киритилган бирор майдондаги маълумотни таҳрирлашга эҳтиёж пайдо бўлса, ўзгартириладиган майдон танланади ва <F2> клавишаси босилади. Шундан кейин майдондаги маълумотни ўзгартириш мумкин.

Агар маълумотларни жадвалга киритишда рус алифбеси ҳарфлари экранда нотўғри кўрсатилса, маълумотларни чиқариш учун мўлжалланган шрифтни алмаштирилади. Бунинг учун **Edit** менюсидан **Preferences** буйруғини танлаб, очилган диалог ойнасининг **General** пунктидаги **Change** тугмаси чертилади. Натижада **Change Font** диалог ойнаси экранда пайдо бўлади ва ундан руслаштирилган шрифтни танлаш мумкин. Шуни ёдда сақлаш керакки, Windows 2000 (Windows XP) системаси Open Type типидаги шрифтлардан, Database Desktop утилити эса TrueType шрифтлари билан ишлайди. Шунинг учун шрифтлар рўйхатидан айнан руслаштирилган TrueType шрифтини танлаш лозим. Шундан кейин, Database Desktop билан ишни тугатиш мумкин, чунки конфигурацияга киритилган барча ўзгартиишлар утилитни қаайта юкланганидан кейингина ишга тушади.

менюга

18.7. Маълумотлар базасини бошқариш дастурлари

Маълумотлар базасини бошқариш дастурларини яратиш жараёнини "Talaba" маълумотлар базасини яратиш мисолида кўриб чиқамиз.

МБ сини бошқариш дастурларини ишлаб чиқишдан аввал, Database Desktop утилитидан фойдаланиб МБ жадвалини яратиш ва унга бир нечта ёзувларни киритиб қўйиш лозим. 18.3-жадвалда "Talaba" жадвалининг майдонлари, яъни структураси келтирилган.

"Talaba" жадвалининг структураси 18.3-жадвал

Майдон	**Тип**	**ўлчами**	**Мазмуни**
Fam	A	15	Талабанинг фамилияси
Ism	A	40	Исми
Fak	A	255	Факультети
Gurux	A	12	Гуруҳи
Tug_kun	D	8	Туғилган куни
Zachetka	A	6	Синов дафтарчасининг номери

Ушбу МБ жадвали структурасини яратганимиздан сўнг, энди клавиатурадан фойдаланиб, бир нечта ёзувлар, яъни талабалар ҳақидаги маълумотларни жадвалга киритамиз.

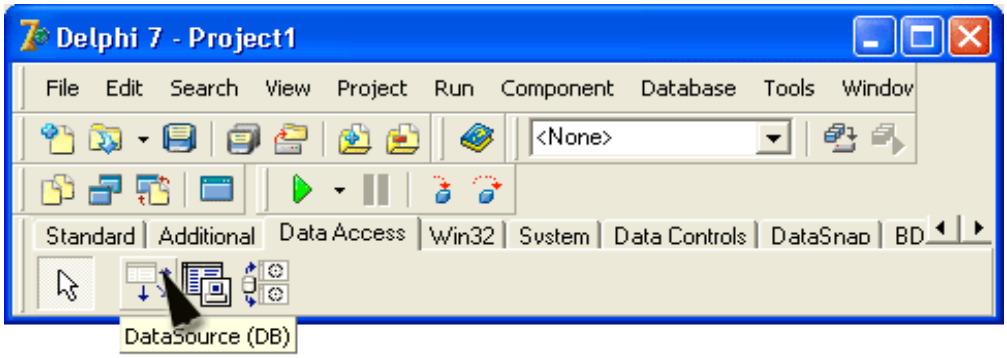
Талабалар жадвали 18.4-жадвал

Fam	Ism	Fak	Gurux	Tug_kun
Абдуллаев	Илхом	Математика	103	05.06.85
Ботирова	Клара	Физика	201	13.11.86
Даминов	Содиқжон	Физика	402	24.09.84
Комилова	Гулчеҳра	Химия	302	16.03.85
Салимова	Гавҳарой	География	105	09.08.86
Турсунов	Абдулбоқи	Математика	401	07.07.84

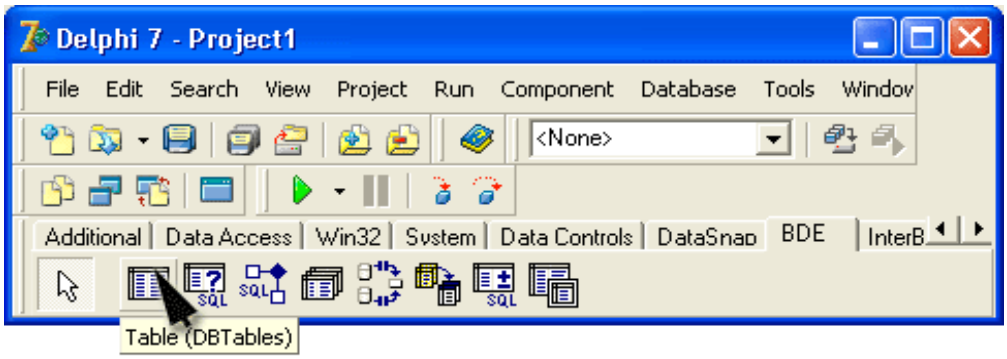
Энди илова ишлаб чиқишга киришиш мумкин. МБ балан ишлаш учун мўлжалланган дастурларни тайёрлаш усули бошқа оддий дастурлар ёзиш усулларидан ортиқча фарқ қилмайди: дастурчи формага зарур бўлган компоненталарни ўрнатади, уларнинг ҳусусият қийматларини киритади ҳамда зарур бўлган ходисаларни қайта ишлаш процедураларини яратади.

МБ билан ишлашга мўлжалланган иловалар МБ жадвалларидагр майдонларга маълумотларни киритиш, қайта ишлаш, таҳрирлаш ва кўриш каби масалаларни ҳал қилиш учун зарур бўладиган компоненталарни ўз ичига олган бўлиши лозим. Бу компоненталар **Data Access** қуроллар панелида, маълумотларни экранга чиқариш компоненталри эса **Data Controls** пунктида жойлашган.

МБ (жадваллар) билан ишлаш. МБ даги маълумотлар билан ишлаш учун нишонлари **Data Access** ва **BDE** қуроллар панелида жойлашган **Database, Table, Query** ҳамда **DataSource** компоненталри хизмат қилади.



18.6а-расм. Data Access қуроллар панелининг компоненталари

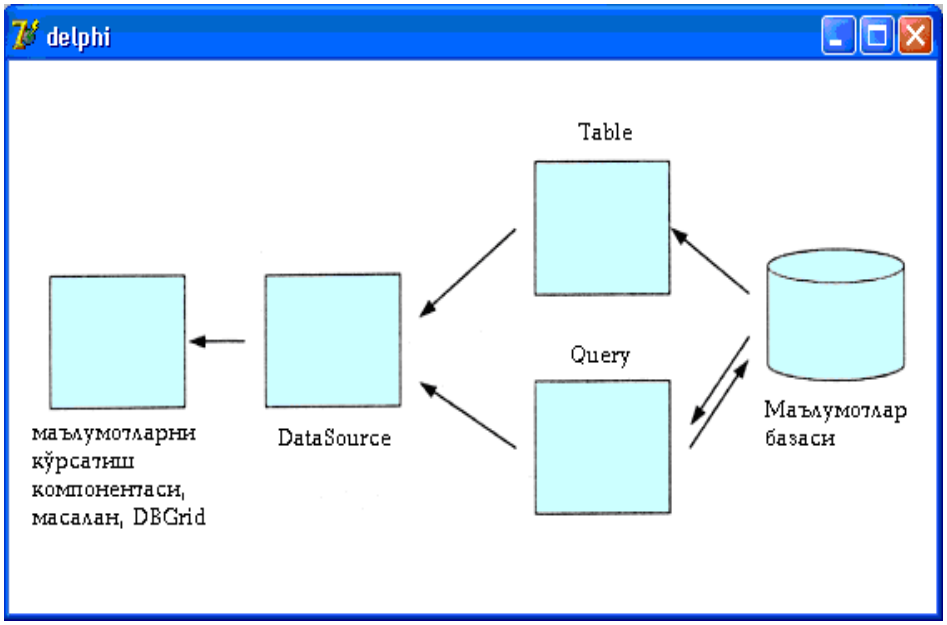


18.6б-расм. BDE қуроллар панелининг компоненталари

Database компонентаси МБ ни битта, умумий, яъни жадваллар тўплами сифатида ифодалайди, Table компонентаси эса МБ жадвалларидан бирини кўрсатади. DataSource (маълумотлар манбаси) кўрсатиш-таҳрирлаш компонентасини (Масалан, DBGrid компонентаси) ва маълумот манбаси (жадвал ёки SQL-сўровнома натижаси-SQl-компонента) ўртасидаги муносабатни белгилайди. DataSource компонентаси оператив тарзда маълумотлар манбасини танлаш, битта компонента, масалан, DBGrid компонентасидан

жадвалдаги маълумотларни ёки SQL-сўровномани шу жадвалга қўллаш натижасини кўрсатиш учун фойдаланиш имконини беради.

Кўрсатиш-таҳрирлаш компонеталарининг DataSource компонентаси орқали ўзаро боғланишини 18.7-расмда тасвирланган.



18.7-расм. Кўрсатиш ва маълумотлар билан ишлаш компонеталарининг ўзаро боғланиши

Энг содда ҳолда, яъни МБ битта жадвалдан иборат бўлганда МБ билан ишлаш иловаси битта Table ва битта DataSource компонентасига эга бўлади.

18.5-жадвалда Table компонентасининг ҳусусиятлари, 18.6-жадвалда эса DataSource компонентаси ҳусусиятлари келтирилган. Бу ҳусусиятларнинг ҳаммаси формага компонента ўрнатилганидан кейин аниқланади.

Table компонентасининг ҳусусиятлари 18.5-жадвал

Ҳусусияти	Мазмуни
Name Database	Компонента номи. Компонента ҳусусиятларига мурожаат қилишда фойдаланилади
NameTable	Компонента қўлланаётган жадвал (маълумотлар файли) нинг асоси бўлган МБ номи. Бу ҳусусиятнинг қиймати сифатида шу МБ нинг таҳаллуси олинади.
Name Table	Компонента қўлланаётган жадвал (ёки маълумот файли) нинг номи
Type	Жадвалнинг типи. Жадвал Paradox («Paradox), dBase (ttDBase), FoxPro («FoxPro) форматидаги маълумотлардан иборат бўлиши ёки форматланган файл (TTASCII) бўлиши мумкин.
Active	Маълумотлар файли (жадвалининг) фаоллашган лигининг белгиси. Бу ҳусусиятга True қиймати берилса, жадвал файли очилади.

DataSource компонентасининг ҳусусиятлари 18.6-жадвал

Ҳусусияти	**Мазмуни**
Name	Компонентанинг номи. Компонента ҳусусиятларига мурожаат қилишда ишлатилади.
DataSet	Кирувчи маълумотлардан иборат бўлган компонента номи

Илова формасини яратаётган пайтда, DatabaseName ҳамда TableName ҳусусиятларига қийматлар мавжуд рўйхатдан танлаш орқали берилади. DatabaseName рўйхатида барча қайд қилинган таҳаллуслар,

TableName – рўйхатида эса таҳаллусга мос келадиган жадвал файларининг номлари келтирилади.

DataSet ҳусусияти ёзувлар билан ишлашга мўлжалланган компоненталар, жадвал ёки сўровнома кўринишидаги компоненталарни бир-бири билан боғлаш учун хизмат қилади. Бу ҳусусиятнинг мавжудлиги маълумотлар манбасини танлашга имкон беради. Масалан, МБ шундай ташкил қилинган бўлиши мумкинки, катта сондаги ёзувларни ўз ичига олган жадвал бир нечта бир ҳил структурали қисм жадвалларга бўлинган бўлиши мумкин. Бу ҳолда иловада ҳар бир қисм жадвалга ўзининг Table компонентаси мос келади, аниқ бир қисм жадални танлаш DataSet ҳусусиятининг қийматини ўрнатиш орқали амалга оширилади.

Ташкил қилинаётган илова учун Table ва DataSource ҳусусиятларининг қийматлари қуйидаги жадвалларда берилган.

DataSource компонентаси ҳусусиятининг қийматлари 18.7-жадвал

Ҳусусияти	**Қиймати**
Name	DataSourcel
DataSet	Table1

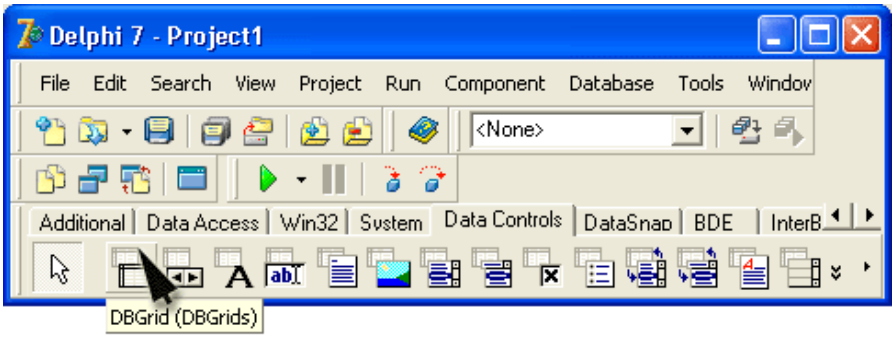
Table компонентаси ҳусусиятининг қийматлари 18.8-жадвал

Ҳусусияти	**Қиймати**
Name	Table1
DatabaseName	STANDART1
TableName	Talaba.db
Active	True

менюга

18.8. Маълумотлар базасини кўриш

Фойдаланувчи Мб ни форма режимида ёки жадвал режимида кўриши мумкин. Форма режимида фақат битта ёзувни, жадвал режимида эса бир нечта ёзувларни бир вақтда кўриш мумкин. Кўпинча, бу икки режимни бирлаштириш мумкин бўлади. Қисқа маълумотлар (айрим асосий майдонлардаги маълумотлар) жадвал кўринишида, зарур бўлганда эса ёзувни кўриш учун форма режимга ўтилади. МБ майдонларидаги маълумотларни кўриш ва таҳрирлаш учун мўлжалланган компоненталар **Data Controls** қуроллар панелида жойлашган. (18.8-расм). Маълумотларни форма режимида кўришни таъминлаш учун формага кўришга имкон берувчи компоненталар қўшилади. Агар зарур бўлса, майдонлардаги маълумотларни таҳрирлаш учун таҳрирлаш компонентасини (ҳар бир майдонга биттадан компонента) ҳам формага ўрнатилади.



18.8-расм. МБ майдонларини кўриш ва таҳрирлаш компоненталари

DBText компонентаси майдонлардаги маълумотларни кўришга, DBEdit ва DBMemo

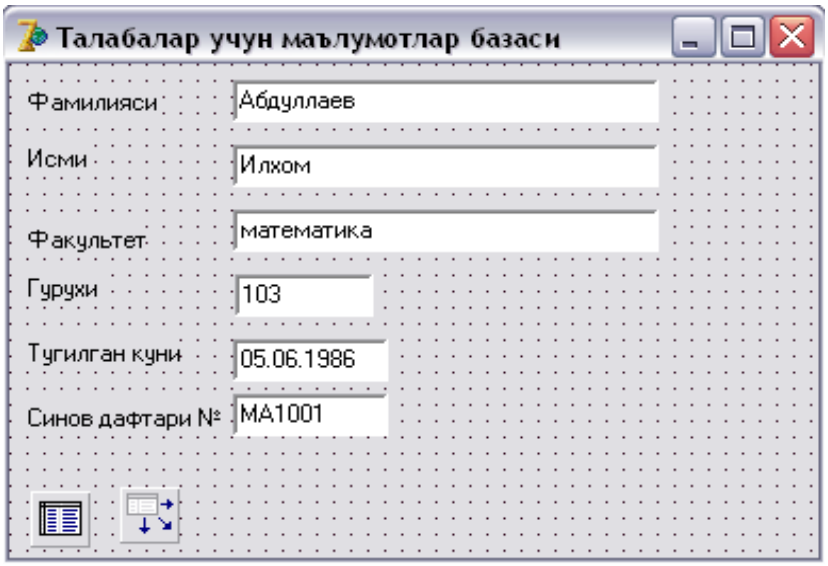
компоненталари эса маълумотларни ҳам кўриш, ҳам таҳрирлашга имкон беради. 18.9-жадвалда бу компоненталарнинг айрим ҳусусиятлари саанаб ўтилган. Илова формасига бу компонента қўшилганидан кейин, ҳусусиятлар кетма-кетлиги жадвалда кўрсатилган тартибда белгиланади.

DBText, DBEdit ва DBMemo компонента ҳусусиятлари 18.9-жадвал

Ҳусусияти	**Мазмуни**
Name	Компонента номи. Компонента ҳусусиятларига мурожаат қилиш учун ишлатилади
DataSource	Маълумот манбасининг компонентаси
DataField	Компонента қўлланаётган МБ майдонининг номи

DBEdit ва DBMemo компоненталаридан фойдаланишга мисол сифатида "Talaba" МБ си билан ишлаш учун мўлжалланган дастурни кўрамиз. Илова формасининг кўриниши 18.9-расмда берилган.

Форма қуйидагича усул билан яратилади. Дастлаб бўш формага Table ва Datasource компоненталарини жойлаштириб, уларнинг ҳусусият қийматларини 18.10-жадвалга мос равишда ўзгартирилади. Қийматларнинг ўрнатиш тартиби жадвалда кўрсатилган тартибга тўла мос бўлиши лозим.



18.9-расм. Талабалар учун МБ иловасининг формаси

Table1 ва DataSourcel компоненталарининг қийматлари. 18.10-жадвал

Ҳусусияти	**Қиймати**	**Изоҳ**
Tablel. DatabaseName	STANDART1	Мб нинг таҳаллуси (BDE Administrator утилити билан ҳосил қилинади.)
Tablel. TableName	Talaba. db	МБ жадвали (Database Desktop утилити билан яратилади)
Tablel. Active	True	
DataSource1. Dataset	Tablel	

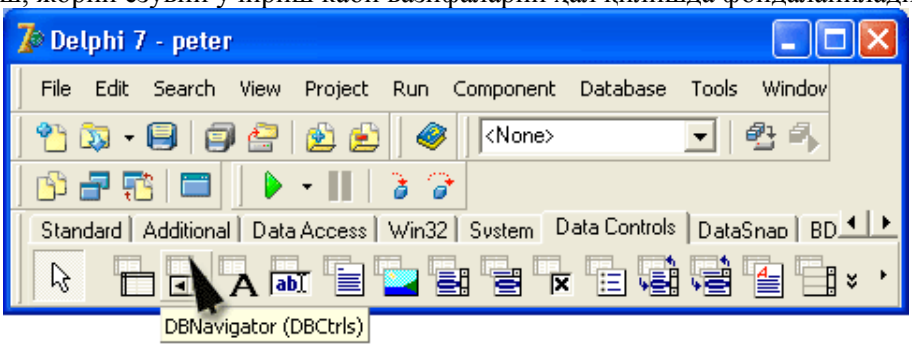
Table ва Datasource компоненталари созланганидан сўнг, формага олтита майдоннинг ҳар бирига биттадан DBEdit компоненталари ўрнатилади. Майдонларни кўриш-таҳрирлаш компоненталари бўлган барча DBEdit1 – DBEdit6 компоненталари ҳусусиятларининг қийматлари қуйидагича белгиланади: ҳаммаси учун DataSource ҳусусиятининг қиймати DataSource1 га , DataField ҳусусиятининг қиймати эса мос равишда Fam, Ism, Fak, Gurux, Tug_kun, Zachetka га тенг.

Table1 компонентасининг Active ҳусусиятига True қиймати берилгани учун, DataField ҳусусиятига қиймат берилган заҳоти DBEdit компонентаси ойнасида МБ жадвалининг биринчи ёзувидаги белгиланган

майдондаги маълумот чиқарилади. Агар жадвалга маълумотлар киритилмаган бўлса, бу майдон бўш қолади. Агар Table1 компонентасининг Active ҳусусиятига False қиймати берилган бўлса, у ҳолда DBEdit компонента майдонида унинг номи, Name ҳусусиятининг қиймати чиқарилади.

Шундан кейин дастурни компиляция қилиб, ишга туширилса, экранда МБ жадвалидаги биринчи ёзувдаги маълумотларни ўз ичига олган форма пайдо бўлади. (18.9-расм)

МБ жадвалидаги бошқа ёзувлар билан ишлаш имкониятига эга бўлиш учун, илова формасига нишони **Data Controls** қуроллар панелида жойлашган DBNavigator компонентасини (18.10-расм) ҳам ўрнатиш лозим. DBNavigator компонентаси (18.11-расм) тугмалар тўпламидан иборат бўлиб, дастур ишлаётган вақтда ёзувларнинг кўрсаткичини олдинга, орқага, биринчи ёзувга, охирги ёзувга суриш ҳамда янги ёзувни қўшиш, жорий ёзувни ўчириш каби вазифаларни ҳал қилишда фойдаланилади.



18.10-расм. DBNavigator компонентаси Data Controls да жойлашган

18.11-расм. DBNavigator компонентаси

18.11-жадвалда DBNavigator компонентасининг тугмалари ва улар бажарадиган амаллар рўйхати берилган. DBNavigator компонентаси ҳусусиятларининг қийматлари эса 18.13-жадвалдан жой олган.

DBNavigator компонентасининг тугмалари **18.12-жадвал.**

	тугма	**Белги-ланиши**	**вазифаси**
	Биринчига	nbFirst	Ёзувлар кўрсаткичини биринчи ёзувга ўтказади
	Аввалгисига	nbPrior	Ёзувлар кўрсаткичини битта аввалги ёзувга ўтказади
	Навбатдагиси	nbNext	Ёзувлар кўрсаткичини битта кейинги ёзувга ўтказади
	Охиргисига	nbLast	Ёзувлар кўрсаткичини охирги ёзувга ўтказади
	Қўшилсин	Nblnsert	Маълумотлар файлига янги ёзув қўшади
	Ўчириш	nbDelete	Маълумотлар файлидан жорий ёзув ўчирилади.
	Таҳрирлаш	nbEdit	Жорий ёзувни таҳрирлаш режимини ўрнатади
	Сақлаш	nbPost	Жорий ёзувга киритилган ўзгаришлар сақланади.
	Бекор қилиш	Cancel	Жорий ёзувга киритилган ўзгаришларни бекор қилади.
	Янгилаш	nbRefresh	Қилинган барча ўзгаришлар файлда сақлаб қўйилади.

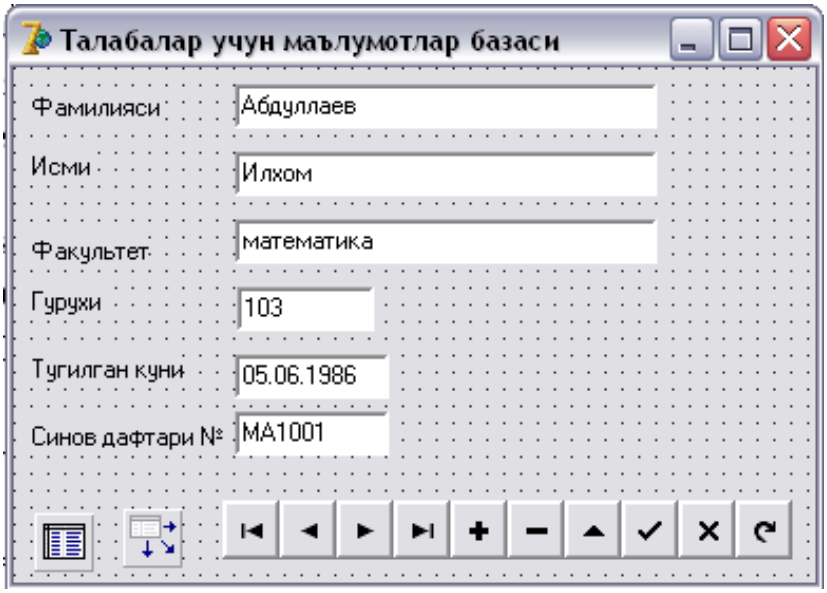
DBNavigator компонентасининг ҳусусиятлари. 18.13-жадвал

Ҳусусияти	**Вазифаси**
VisibleButtons	Кўринадиган буйруқли тугмалар
Name	Компонента номи. Компонента ҳусусиятларига мурожаат қилиш учун ишлатилади.

DataSource	Маълумотлар манбаси бўлган компонентанинг номи. Маълумот манбаси сифатида Маълумотлар базаси (Database компонентаси), жадвал (Table компонентаси) ёки сўровнома натижаси (Query компонентаси келиши мумкин)

VisibleButtons хусусиятига алоҳида эътибор беринг. У DBNavigator компонентасининг айрим тугмаларини яширишга имкон беради. Бу билан маълумотлар файли устида бажариладиган амалларни таъқиқлаб қўйиш мумкин. Масалан, МБ жадвалидан маълумотларни ўчиришни таъқиқлаш учун VisibleButtons.nbDelete хусусиятига False қиймати берилади.

18.12-расмда Талабалар учун МБ иловасининг DBNavigator компонентаси ўрнатилганидан кейинги форманинг ҳолати тасвирланган. DBNavigator нинг DataSource хусусиятига Table1 қийматини бериб қўйилган.



18.12-расм. Талабалар учун МБ иловасининг якуний кўриниши.

Формага DBNavigator компонентасини қўшгандан кейин МБ ни бошқариш учун соддагина дастурни тайёр деб айтиш мумкин. Бу дастур Мб жадвалидаги маълумотларни кўриш, киритиш ва таҳрирлашни таъминлай олади. Янги ёзувларни қўшади, нокеракларини ўчиради.

18.1-листингда Талабалар учун маълумотлар базаси дастурининг матни келтирилган.

18.1-листингда Талабалар учун маълумотлар базаси

```
unit Talaba_MB;
interface
uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, Mask, DBCtrls, DB, DBTables, ExtCtrls;
type
  TForm1 = class(TForm)
    Table1: TTable;
    DataSource1: TDataSource;
    DBEdit1: TDBEdit;
    DBEdit2: TDBEdit;
    DBEdit3: TDBEdit;
    DBEdit4: TDBEdit;
    DBEdit5: TDBEdit;
    DBEdit6: TDBEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
```

```
  DBNavigator1: TDBNavigator;
 private
  { Private declarations }
 public
  { Public declarations }
 end;
var
 Form1: TForm1;
implementation
{$R *.dfm}
end.
```

Ушбу дастур ишга туширилганда "Talaba" МБ жадвалидаги биринчи ёзув экранга чиқади. Фойдаланувчи DBNavigator1 компонентасининг тугмаларидан керагини босиб, бу МБ жадвали устида бошқа ёзувларни кўриш, таҳрирлаш, янги ёзувларни жадвалга қўшиш, нокерак бўлган ёзувларни ўчириш имкониятига эга бўлади.

менюга

18.9. Маълумотларни жадвал режимида кўриш.

"Талабалар учун МБ" дастури маълумотларни форма режимида экранга чиқаради. Жорий вақт мобайнида фойдаланувчи фақат битта ёзувни кўра олади. МБ жадваллари билан ишлаганда бундай усул ҳар доим ҳам қулай ҳисобланавермайди. Агар бир вақтнинг ўзида бир нечта ёзувни кўришга эхтиёж пайдо бўлса, маълумотларни жадвал режимида кўришни таъминлаш лозим бўлади.

Маълумотларни жадвал режимида кўриш учун дастур ёзишни "Мактаб" МБ си мисолида кўрамиз.

"Мактаб" МБ си (тахаллуси-maktab) uquvchi.db файлидаги жадвалдан иборат бўлсин. Унинг структураси қуйидагича бўлсин: Fam (Фамилия), Ism (исми), sinf (синфи), Adr (манзили) ва N (шахсий номери). Fam, Ism, sinf ва Adr майдонлари ҳарфий, (A типи), N —майдони эса сонли ва автоматик тарзда ўзгариб борувчи.

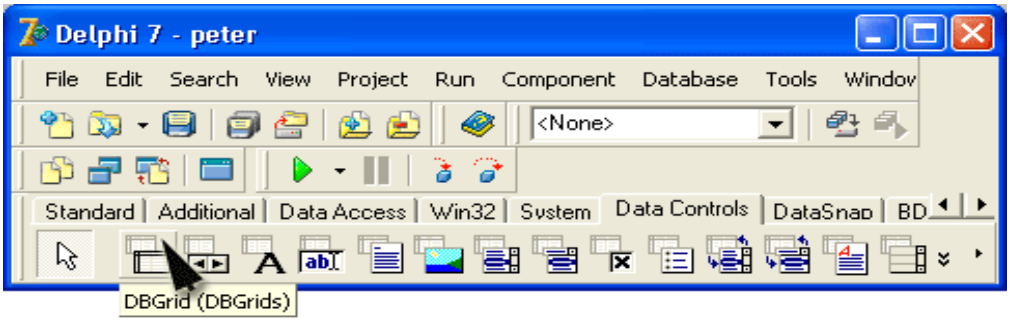
Эслатма: Maktab таҳаллусини BDE Administrator, uquvchi.db жадвалини Database Desktop утилитлари ёрдамида ташкил қилинади.

Дастлаб яратилаётган илова формасига Table ва DataSource компоненталарини ўрнатамиз. Улар маълумотлар файли билан ишлашга имкон беради. Бу компоненталар хусусиятларининг қийматларини қуйидагича ўрнатамиз.

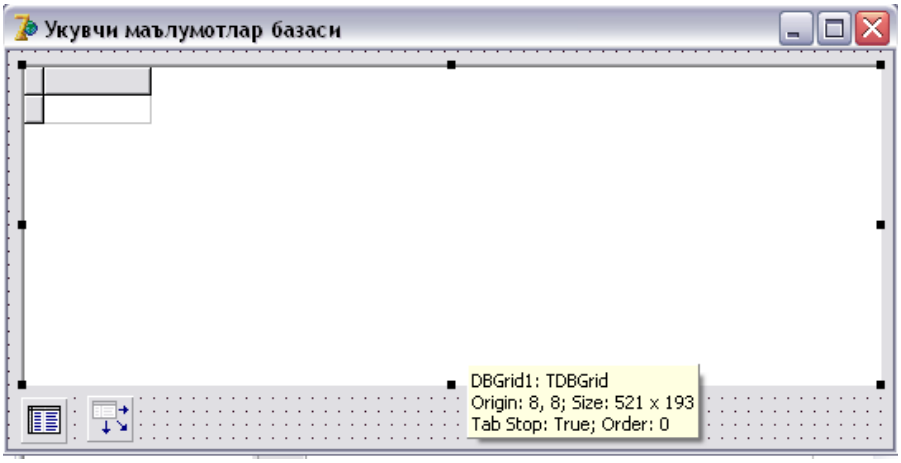
Table ва DataSource хусусиятларининг қийматлари. 18.14-жадвал

Хусусияти	**қиймати**
Tablel . DatabaseName	Maktab
Tablel . TableName	uquvchi.db
Tablel. Active	True
DataSourcel . Dataset	Tablel

Маълумотларни жадвал режимида кўриш ва таҳрирлаш учун формага нишони **Data Controls** куроллар панелида жойлашган DBGrid компонентаси (18.13-расм) жойлаштирилади. Яратилаётган илова формасининг кўриниши 18.14-расмда келтирилган.



18.13-расм. DBGrid компонентасининг нишони



18.14-расм. DBGrid компонентаси қўшилган форманинг кўриниши

DBGrid компонентаси МБ ни жадвал кўринишида ифодалашга имкон беради. DBGrid1 компонентасининг ҳусусиятлари жадвалнинг ташқи қиёфаси ва дастур ишлаётган вақтда маълумотлар устида бажарилиши мумкин бўлган амалларни белгилайди. 18.15-жадвалда DBGrid компонентасининг айрим ҳусусиятлари келтирилган.

DBGrid компонентасининг ҳусусиятлари. **18.15-жадвал**

Ҳусусияти	**Вазифаси**
Name	Компонента номи
DataSource	Жадвалдаги маълумотларнинг манбаси
Columns	Жадвалда кўринадиган маълумотлар
Options . dgTitles	Устунлар сарлавҳасини чиқариш
Options . dgIndicator	Индикаторлар устуни. МБ билан ишлаган-да жорий ёзув - учбурчак, янги ёзув -"*", таҳрирланаётган ёзув эса махсус белги билан кўрсатилади.
Options . dgColumnResize	Дастур ишлаётганда устун кенглигини ўзгартиришга рухсат беради.
Options . dgColLines	Жадвал устунларини ажратувчи чизиқни чиқаришга рухсат беради.
Options . dgRowLines	Жадвал сатрларини ажратувчи чизиқни чиқаришга рухсат беради.

Дастур ишлаётган вақтда қандай маълумотлар экранда кўрсатилиши кераклигини белгилаш учун дастлаб жадвалнинг маълумотлар манбасини аниқлаш (DataSource ҳусусиятини ўрнатиш) лозим. Сўнгра Columns ҳусусиятининг қийматларини белгиловчи параметрларни ўрнатилади. DataSource ҳусусиятининг қиймати одатдаги усул билан, яъни **Object Inspector** ойнасидан фойдаланиб белгиланади. Columns ҳусусиятига қиймат бериш учун **Object Inspector** ойнасидан бу ҳусусиятни танлаб, учта нуқтали тугма чертилади. Натижада устунлар диалог ойнаси очилади. (18.15-расм.)

DBGrid компонентасига МБ файлининг ёзув майдонларида сақланаётган маълумотларни кўришни таъминлайдиган устунларни қўшиш учун экраннинг юқори сатрида жойлашган қуроллар панелидаги **Add New** тугмаси босилади. Шундан кейин, қўшилган элементни ажратилади ва **Object Inspector** ойнасидан фойдаланиб, бу устуннинг ҳусусият қийматларини ўрнатилади. (18.16-жадвал).

18.15-расм. Устунлар муҳаррири

DBGrid компонентасининг columns ҳусусияти элементлари TColumn типида бўлган массивни ифодалайди. Ҳар бир устунга массивнинг элементи мос келади. Column компонентаринининг ҳусусият қийматларини ўрнатар экан, дастурчи DBGrid компонентаси устунларининг қиёфасини белгилаши шарт, шу билан бирга жадвални тўлалигича кўринишини аниқлайди.

Column компонентасининг ҳусусиятлари 18.16-жадвал

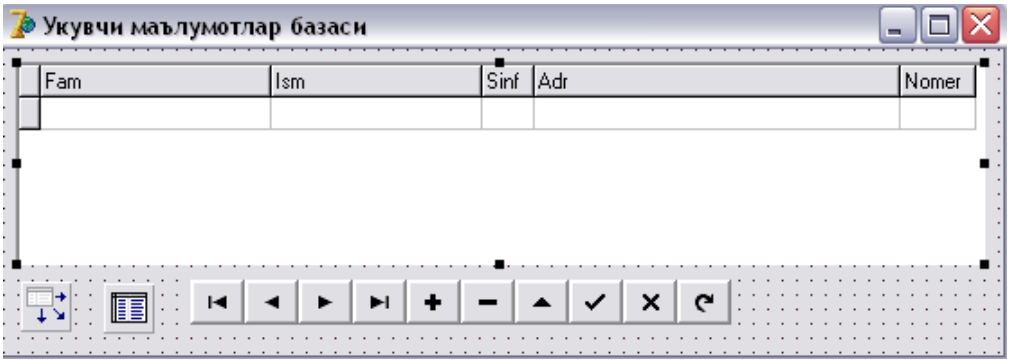
Ҳусусияти	**Вазифаси**
FieldName	Устунга чиқариладиган майдоннинг номи
Width	Устуннинг пикселдаги кенглиги
Font	Устун ячейкасидаги матн учун шрифт
Color	Ранги
Alignment	Ячейкаларда матнни текислаш усули. Матнни чап томондан (taLeftJustify), марказий (taCenter) ёки ўнг томондан (taRightJustify) текислаш мумкин.
Title. Caption	Устун сарлавҳаси. Сарлавҳа кўрсатилмаса, қиймати майдон номига тенг бўлади.
Title .Alignment	Сарлавҳаларни текислаш усули. Сарлавҳа чап томондан (taLeftJustify), марказий (taCenter) ёки ўнг томондан (taRightJustify) текислаш мумкин.
Title. Color	Устун сарлавҳасининг фон ранги
Title. Font	Устун сарлавҳасининг шрифти

Энг содда ҳолда, ҳар бир устун учун FieldName ҳусусияти қийматини ўрнатиш етарли. У устунга чиқариладиган майдон номини белгилайди. Шунингдек, устун сарлавҳасини кўрсатувчи Title.Caption ҳусусиятининг қийматини ҳам бериш шарт. 18.18-жадвалда DbGrid1 компонентасининг columns ҳусусияти қийматлари келтирилган.

DBGrid1 компонентаси ҳусусиятининг қийматлари. 18.18-жадвал

Компонента	**FieldName**	**Title . Caption**
DBGrid1. Columns [0]	Fam	Фамилияси
DBGrid1. Columns [1]	Ism	Исми
DBGrid1. Columns [2]	Sinfis	Синфи
DBGrid1. Columns [3]	Adr	Манзили, телефони
DBGrid1.Columns [4]	Nomer	Тартиб номери

Ишнинг охирида формага DBNavigator компонентасини жойлаштириб, унинг ишини маълумотлар манбаси бўлган жадвалга созлаймиз. (DataSource ҳусусиятига Table1 қийматини берамиз). Шундан кейин, илова формаси ўзининг якуний кўринишига эга бўлади. (18.16-жадвал)



18.16-расм. DBGrid1 компонентаси созланганидан кейинги форма

Шундан кейин, дастурни компиляция қилиб, ишга тушириш мумкин. Дастур ишга тушганидан сўнг, экранда маълумотлар пайдо бўлиши учун, ёки агар база бўш бўлса, янги маълумотларни киритиш учун маълумот манбаси бўлган жадвалнинг Active ҳусусиятига True қиймати берилган бўлиши керак.

Жадвал кўринишидаги МБ билан ишлаш Microsoft Excel жадвали билан ишлашга ўхшаб кетади. Курсорни суриш стрелкалари ёрдамида МБ даги ёзувларни кўриш мумкин. <Ins> тугмасини босиб, янги ёзувни қўшиш, <Del> тугмаси ёрдамида ёзувларни ўчириш мумкин. Ёзув майдонидаги маълумотларни таҳрирлаш учун курсорни керакли майдонга ўрнатиб, <F2> тугмасини босиш лозим.

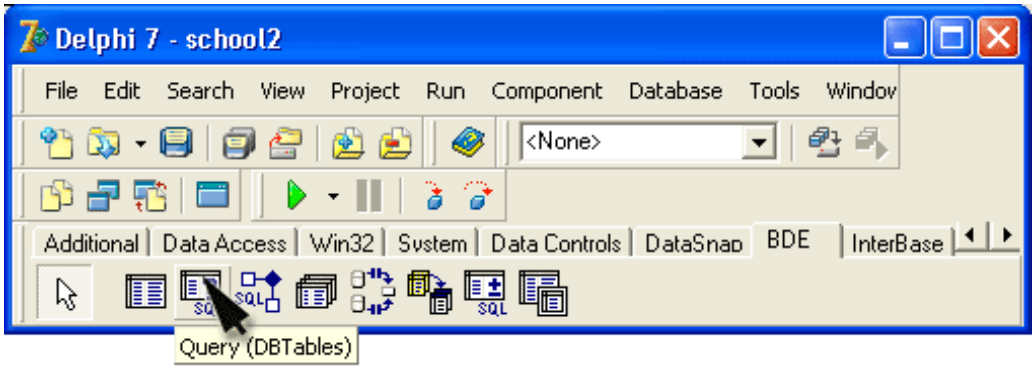
менюга

18.10. Маълумотлар базасидан ахборот танлаш

Фойдаланувчи МБ билан ишлар экан, одатда уни жадвалдаги тўлалигича камдан-кам ҳоллардагина қизиқтиради. Кўпинча, у маълум бир критерияга жавоб бера оладиган битта ёки бир нечта маълумотни қидириш билан шуғулланади. Зарур бўлса, ёзувларни бирма-бир варақлаб, фойдаланувчи ўзи учун керакли маълумотларни топиши мумкин. Аммо, бундай усулдан фойдаланишнинг самараси юқори бўлмайди.

МБ бошқариш тизимларининг кўпчилигида талаб қилинган маълумотларни сўровномалар ёрдамида ажратиб олинади. Фойдаланувчи маълум бир қонун-қоидалар асосида сўровномалар ташкил қилади. Бунда у маълумот қандай шартларни қаноатлантиришини кўрсатади. Система эса ана шу шартларни қаноатлантирувчи маълумотларни МБ дан қидириб топади ва экранга чиқаради.

МБ даги маълумотлар ичидан қандайдир критерияларга жавоб берадиган маълумотларни қидириб топиш учун Query (18.18-расм) компонентасидан фойдаланилади.



18.18-расм. Query компонентасининг нишони

Query компонентаси Table га ўхшаб кетади, уларнинг фарқи шундаки, Table тўла жадвалдан иборат бўлса, Query компонентаси жадвалнинг сўровнома шартини қаноатлантирувчи қисмидан иборат бўлади. Унинг айрим ҳусусиятлари 18.18-жадвалда келтирилган.

Query компонентасининг ҳусусиятлари 18.18-жадвал

Ҳусусияти	**Вазифаси**
Name	Компонента номи. Datasource компонентаси ундан сўровнома натижаси ҳамда ёзувларни кўриш компонентасини (масалан, DBGrid) боғлаш мақсадида фойдаланади.

SQL	SQL тилида МБ га (жадвалга) ёзилган сўровнома
Active	Агар қиймати True бўлса, сўровномани бажариш режими фаоллашади.

Дастурни ишлаб чиқариш вақтида сўровнома натижасида қандай маълумотлар ажратилишини кўрсатиш учун, SQL ҳусусияти маълумотларни танлаш учун SQL тилида ёзилган сўровномага тенг бўлиши керак.

Жадвалдан маълумотларни ажратиб олиш учун сўровномалар умумий ҳолда қуйидагича ёзилади:

SELECT майдонлар рўйхати **FROM** жадвал **WHERE** (шартлар) **ORDER BY** майдонлар рўйхати ;

Бу ерда **SELECT** — ёзувларни жадвалдан танлаб олиб, номи рўйхатда кўрсатилган майдонлардаги маълумотларни экранга узатиш буйруғи; **FROM** — буйруғнинг параметри бўлиб, қидириш қайси жадвал бўйича олиб борилишини кўрсатади; **WHERE** — қидириш шартини белгиловчи параметр; **ORDER BY** – сўровнома шартларини қаноатлантирувчи ёзувларни тартиблаш критерияси. Масалан,

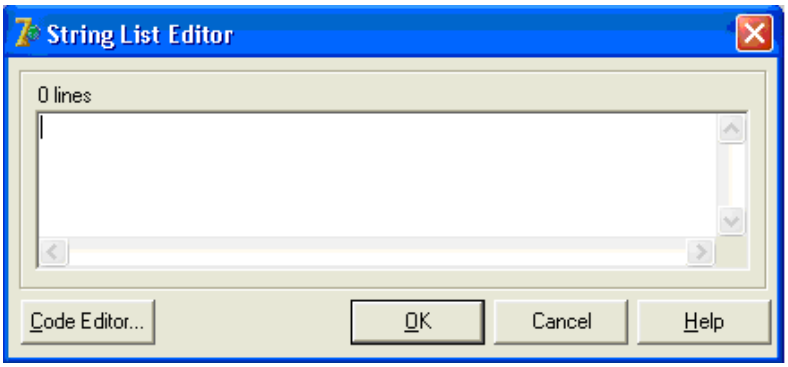
SELECT Fam, Ism **FROM** ':maktab:uquvchi.db' **WHERE**
(Sinf = '10a') ORDER BY Fam, Ism ;

сўрови maktab МБ сидаги (uquvchi.db файлидан) Sinf майдонида "10a" маълумоти турган ёзувларни ажратиб олади ва бу ажратиб олинган маълумотларни Fam ва Ism майдонлари бўйича тартиблаб, экранга чиқаради.

SELECT Fam, Ism FROM ":maktab:uquvchi.db " **WHERE**
(Fam > 'К') **and** (Fam < 'Л') ORDER BY Fam, Ism

сўрови эса фамилияси "К" ҳарфи билан бошланадиган ўқувчилар ҳақидаги ахборотларни ажратиб олиб, уларни алфавит тартибида тартиблаб, экранга чиқаради.

Сўровномаларнинг SQL ҳусусияти қийматини форма ташкил қилинаётганда ёки дастур ишлаётган пайтда кўрсатиш мумкин.



18.18-расм. Сатрлар рўйхатининг муҳаррири ойнаси

Форма ташкил қилинаётган вақтда SQL ҳусусиятига сўровнома ёзиш учун сатрлар рўйхатининг муҳарриридан (18.1-расм) фойдаланиш мумкин. Унинг ойнаси **Object Inspector** ойнасидаги SQL ҳусусияти турган сатрнинг ёнида турган уч нуқтали тугмани чертиш орқали очилади.

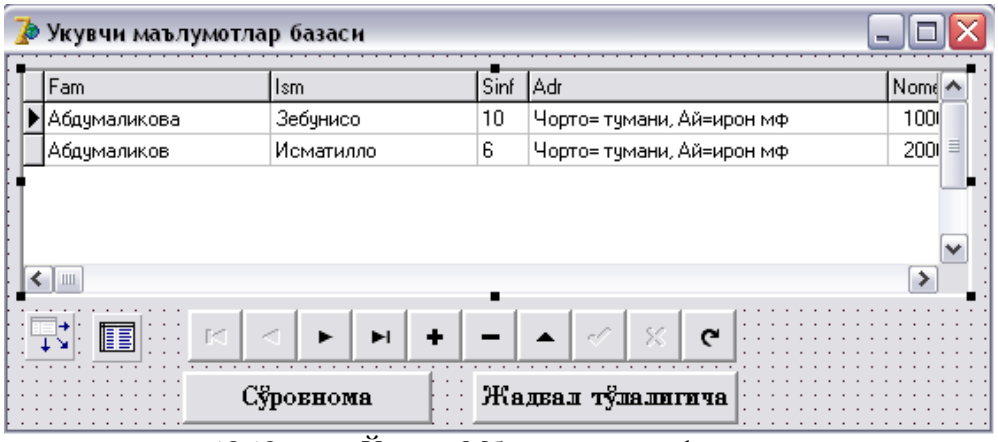
SQL сўровномаси сатрлар рўйхатидан иборат бўлади. Шунинг учун дастур ишлаётган вақтда SQL-сўровнома ташкил қилиш учун Add методидан фойдаланиб, SQL-рўйхатга янги SQL-сатр ёки кўрсатмани қўшиб қўйиш мумкин.

Қуйидаги код фрагментида аниқ бир инсон ҳақидаги маълумотни топиш учун сўровнома ёзилмоқда. (Бу ерда танлаш критерияси Fam майдонидаги маълумот билан fam1 ўзгарувчисининг қиймати устма-уст тушиши керак.)

```
with form1.Queryl do begin
Close;// Файлни ёпиш – аввалги сўровнинг натижаси
SQL.Clear;// аввалги сўровнома натижасини ўчириш
// SQL ҳусусиятига янги қийматларни ёзиш
SQL.Add('SELECT Fam, Ism, Sinf[1]');
SQL.Add('FROM ":Maktab:uquvchi.db"');
SQL.Add('WHERE');
SQL.Add('(Fam = '" + fam1 + "')');
```

```
SQL.Add('ORDER BY Fam, Ism');
Open;// сўровноманинг бажарилишин фаоллаштирамиз
end;
```

Матни 18.2-листингда берилган, диалог ойнаси эса 18.19-расмда кўрсатилган дастур сўровномани, аниқроғи танлов критерия-



18.19-расм. Ўқувчи Мб иловасининг формаси

сини ўзгартириш дастур ишлаб турган вақтда қандай амалга оширилишини намойиш қилади. Дастур ўқувчилар рўйхатини тўла ёки маълум бир қисмини экранга чиқарилишини таъминлайди. Сўровнома бажарилганидан сўнг, аниқ бир ўқувчи ҳақидаги маълумотлар чиқарилади.

МБ ни кўриш ҳамда сўровнома натижасини кўриш учун DBGrid1 компонентасидан фойдаланилади. У DataSourcel компонентаси орқали Table1 (МБ ни тўла кўриш вақтида) компонентаси ёки Query (сўровнома натижасини кўриш вақтида) компонентаси билан боғланган бўлади.

18.2-листинг. "Ўқувчи" маълумотлар базаси

```
unit uquvchi;
interface
uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, Grids, DBGrids, DB,
DBTables, ExtCtrls, DBCtrls, StdCtrls;
type
  TForm1 = class(TForm)
    DataSource1: TDataSource;
    Table1: TTable;
    DBGrid1: TDBGrid;
    DBNavigator1: TDBNavigator;
    Button1: TButton;
    Button2: TButton;
    Query1: TQuery;
    procedure FormActivate(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private      { Private declarations }
  public       { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.FormActivate(Sender: TObject);
begin
    DataSource1.DataSet := Table1;
Table1.Active := True;
end;
// жадвал тўлалигича тугмаси чертилганда
procedure TForm1.Button2Click(Sender: TObject);
```

```
begin
DataSource1.DataSet := Table1;// Маълумотлар манбаси - жадвал
end;
// сўровнома тугмаси чертилганда
procedure TForm1.Button1Click(Sender: TObject);
var
fam: string[30];
begin
fam := InputBox('Маълумотларни МБ дан танлаш',
'Фамилияни кўрсатиб, ОК тугмасини чертинг.' , '');
if fam <> '' // Фойдаланувчи фамилияни киритди
then
begin
with form1.Query1 do begin
Close;// Файлни ёпиш-аввалги сўровномани тугатиш
SQL.Clear;// Аввалги сўровномани ўчириш
// SQL хусусиятига янги сўровнома жўнатамиз
SQL.Add('SELECT Fam, Ism, Sinf');
SQL.Add('FROM ":Maktab:uquvchi.db"');
SQL.Add('WHERE');
SQL.Add('(Fam = "'+ fam + '")');
SQL.Add('ORDER BY Fam, Ism');
Open;// сўровноманинг бажарилишини фаоллаштирамиз
end;
{ *** сўровномани алмаштиришнинг бошқа формаси
begin
Query1.Close;
Query1.SQL[3] := '(Fam="'+ fam + '")';
Query1.Open;
DataSource1.DataSet:=Query1;
end; }
if Query1.RecordCount <> 0 then
DataSource1.DataSet := Query1// сўровнома натижасини кўрсатиш
else begin
ShowMessage('МБ да сиз сўраган маълумотлар йўқ.');
DataSource1.DataSet := Table1;
end;
end;
end;
end.
```

TForm1.Button1Click процедураси **Сўровнома** тугмаси чертилганда ишга тушади. У фойдаланувчидан фамилияни қабул қилиб олади ва SQL-хусусиятига янги сатр-сўровнома ҳосил қилиб қўшади. Сўнгра бу процедура Open методи ёрдамида сўровномани фаоллаштиради.

Эътибор беринг, SQL-сўровнома хусусиятини ўзгартиришдан аввал, уни Close методи билан ёпиляпти. (Чунки, сўровнома бажарилганда, маълумотлар файли (жадвали) яратилади.)

TForm1.Button2Click процедураси **Жадвал тўлалигича** тугмаси чертилганда ишга тушади ҳамда DataSourcel компонентаси учун маълумотлар манбаи сифатида Table1 компонентасини ўрнатади, шу билан жадвални тўла кўриш режимига ўтишни таъминлайди.

Агар SQL хусусиятига қидириш критериясини илова формасини яратилаётган вақтда ёзиб қўйилган бўлса, бу критерияни бошқаси билан алмаштириш учун дастур ишлаётган пайтда ўзгартирилиши талаб қилинган критерияга сўровнома матнидаги мос келадиган сатрни ўзгартириш етарли. Масалан, қуйидаги

SELECT DISTINCT Fam, Ism, sinf FROM ":maktab:uquvchi.db" **WHERE** (Sinf= '10a') ORDER BY Fam, ism

сўровномаси учун сўровнома шартини алмаштириш буйруғини мана бундай ёзиш мумкин:

```
forml.Query1.SQL[3] := '(Fam="' + fam+ '")'
```

Эслатма: SQL хусусияти Tstrings типидаги структура бўлгани учун, сатрлар нолдан бошлаб

номерланади.

менюга

18.11. Динамик яратиладиган таҳаллуслар

МБ билан ишлашга рухсат берадиган таҳуллуслардан фойдаланиш дастурни маълумотларнинг система томонидан жойлаштирилишига боғлиқ бўлмаслигини таъминлайди. Тахаллуслар дастурни ҳам, МБ ни ҳам турли компьютерларда, ҳаттоки тармоқ компьютерларида ҳам сақлашга имкон беради. Содда МБ лар учун, МБ файлларини дастурнинг ўзи сақланаётган каталогнинг осткаталогида сақлаш тавсия қилинади. Шундай қилиб, МБ билан ишлайдиган дастур доимо маълумотлар каерда эканлигини "билади". Масалага бундай ёндошилганда, МБ учун таҳаллусни BDE Administrator ёрдамида ташкил қилмасдан, таҳаллус яратиш вазифасини МБ билан ишлайдиган дастурнинг ўзига топшириш мумкин. Бу усулда, таҳаллус дастур ишлаётган вақтда яратилади, дастур ўз ишини тугатган зоҳоти йўқотилади. Бу эса МБ сини бошқаришни енгиллаштиради. Шу усул билан яратиладиган таҳаллусларни динамик таҳаллус деб атаймиз.

Юқори даги фикрларга намуна қилиб, 18.3-листингда "Ўқувчи" МБ билан ишлашда динамик таҳаллус яратиш ва ундан фойдаланиш процедурасининг матни келтирилган.

18.3-листинг. "Ўқувчи" МБ билан ишлашда МБ таҳаллусини динамик ташкил қилиш процедурасининг матни. (Ушбу матн 18.2-листингдаги procedure FormActivate процедураси ўрнига ёзилади.)

```
//формани фаоллаштириш
procedure TForm1.FormActivate(Sender: TObject);
begin
with Session do
begin
ConfigMode := cmSession;
try
{ Агар маълумотлар файли бажарилувчи дастур ёзилган каталогда жойлашган бўлса, дастурда маълумотлар
файлига йўлни буйруқлар сатридан ExtractFilePath(ParamStr(0)) функцияси орқали топамиз.
Келтирилган мисолда маълумотлар файли дастур жойлашган каталогнинг DATA осткаталогида
жойлашган.}
//МБ учун вақтинчалик таҳаллус яратамиз.
AddStandardAlias( 'maktab',
ExtractFilePath(ParamStr(0))+'DATA/', 'PARADOX');
Table1.Active := True;//МБ ни очамиз
finally
ConfigMode := cmAll;
end;
```

Дастурнинг қаралаётган вариантида МБ дастурнинг бажарилувчи файли сақланган каталогнинг DATA осткаталогида жойлашган деб фараз қиламиз. Таҳаллусни TForm1.FormActivate процедураси яратади. Таҳаллус бу процедура таркибига кирган AddstandardAlias процедураси томонидан ҳосил қилинади ва унга параметр сифатида таҳаллуснинг номи ҳамда унга мос келувчи каталогнинг номи берилган. МБ билан ишлаш дастурининг тайёрлаш жараёнида бу дастурни, шунингдек, DATA осткаталогини ҳам қайси каталогга ва қандай жойлаштирилишини олдиндан билиб бўлмайди. Бажариладиган файлнинг қаердалигини дастур ишлаётган вақтда ParamStr(0) ва ExtractFilePatch функциялари ёрдамида аниқлаш мумкин. Бу функциялардан биринчисининг қиймати – бажариладиган файлнинг тўлиқ номи, иккинчисиники эса шу файлга "бориш" йўлига тенг. Шундай қилиб, AddstandardAlias процедурасига МБ каталогининг тўлиқ номи берилади.

менюга

18.12. МБ бошқариш дастурини бошқа компьютерга ўтказиш

Кўпинча МБ бошқариш дастурларини бошқа компьютерга ўтказишга эҳтиёж пайдо бўлади. Бу ҳолда оддий усулдаги, яъни файлларни тўғридан-тўғри кўчириш усули етарли ҳисобланмайди. МБ

бошқариш дастурларини кўчиришда BDE ни ҳам кўчириш лозим.

Бу ерда, BDE амалий дстурларни МБ билан ишлашини таъминлайдиган дастурлар, кутубхоналар ва драйверлар тўпламидан иборат эканлигини ёдга олиш керак. BDE ни бошқа компьютерга "қўлда" кўчиришнинг иложи йўқ.

Шунинг учун, Borland барча зарурий файлларни, шу жумладан BDE нинг керакли компоненталарини ҳам кўчирадиган ўрнатувчи дастурларни яратишни тавсия қилади. Borland ўрнатувчи дискларни яратишда Delphi нинг барча версиялари таркибига кирган InstallShield Express утилитидан фойдаланишни маслаҳат беради. Бу утилит BDE ни созлаш ҳамда дастурларни бошқа компьютерга кўчириш масаласига мослаштирилган.

BDE оддий усул билан ўрнатиш мумкин. Қуйида Paradox даги маълумотлар базаси билан ишлаш учун зарур бўлган файллар рўйхати (бу рўйхат синовлар ёрдамида аниқланган) келтилган:

BLW32.DLL, IDAPI32.DLL, IDBAT32.DLL, IDPDX32.DLL, IDR20009.DLL, USA.BLL, CHARSET.BLL.

Бу файлларни фойдаланувчининг компьютерига ўрнатиш лозим. Сўнгра, Windows реестрида қуйида санаб ўтилган бўлимлар ва параметрларнинг мавжудлигини текшириш лозим:

- HKEY_LOCAL_MACHINE / Software / Borland / Database engine — бўлими. DLLPATH параметри DLL-BDE файлларига йўлни кўрсатиши керак;
- HKEY_LOCAL_MACHINE / Software / Borland / BLW32 — бўлими. BLL-BDE файлларига йўлни кўрсатиши керак.

менюга

мундарижага қайтиш

19-БОБ. OLE АСОСЛАРИ

19.1. Асосий тушунчалар

OLE асосларини кўришдан аввал, биз баёнимизда учрайдиган асосий терминологиялар билан танишамиз.

OLE Аббревиатура Objects Linked and Embedded (Бирлаштирилган ва Ичига Киритилган Объектлар - БИКО) маъносини англатади.

Иловалар ўртасида тақсимланадиган маълумотларни **OLE объекти** деб аталади.

OLE объектларини ўз ичига олган иловаларни **OLE контейнер** (OLE Container) дейилади.

Маълумотларини OLE контейнерга OLE объект сифатида олиш мумкин бўлган иловаларни **OLE сервер** деб аталади.

Масалан, MicroSoft Word иловаси ҳужжат таркибига график объектларни, аудио ва видео клипларни ҳамда кўплаб турдаги бошқа объектларни олиши мумкин. (Бундай ҳужжатларни таркиб ҳужжати - compound document деб ҳам юритилади.)

Номидан кўриниб турибдики, OLE объектларни ёки OLE контейнерга бирлаштириш мумкин, ёки унинг ичига киритиб қўйиш мумкин. Биринчи ҳолда, маълумотлар дискдаги файлларда сақланади ва ихтиёрий иловаларга бу файлдаги маълумотлар билан ишлаш ва ўзгартириш ҳуқуқи берилади. Иккинчи ҳолда, маълумотлар OLE контейнер таркибига қўшилади ва фақат OLE контейнерга бу маълумотларни кўриш ва ўзгартириш учун рухсат берилади.

OLE иловалар ўртасида маълумотлар тақсимлаш ғоясининг ривожлантирилиши натижасида юзага келган. Агар DDE ёрдамида фақат ҳужжатлар билан ишлаш мумкин бўлса, OLE ихтиёрий типдаги маълумотларни иловалар таркибига осонгина қўшиб қўйиш имконини беради. Мижоз-илова OLE контейнерини тўғри ишлаши учун OLE сервернинг мавжуд бўлиши шарт. Мижоз дастурида фойдаланувчи OLE объектга ҳар гал маълумотларни кўриш ёки таҳрирлаш учун мурожаат қилганда, (одатда объект номини кўрсатиб, сичқонча икки марта чертилади) илова-сервер ишга тушади ва маълумотлар устида ишлаш рўй беради.

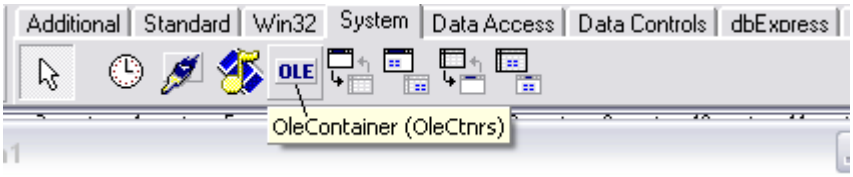
OLE –серверларни фаоллаштириш усулига қараб, OLE нинг бир нечта турлари мавжуд. 1-версиядаги OLE серверни алоҳида ойнада ишга туширади. 2-версиядаги OLE эса 2 ***in-place activation and editing*** ни амалга оширади, яъни сервер мижоз-илованинг "ичида" ишга туширилади ҳамда система

менюси, куроллар панели ва бошқаларни ўзига мослаб ўзгартирилади. OLE ғоясини ривожлантириш ***OLE automation*** га олиб келди. Бунда мижоз-илова сервернинг маълум бир қисм вазифасини бажаради. Мижоз-дастурга жойлаштирилган OLE объектнинг типи сервернинг қайси OLE версиясида ишлаши билан аниқланади.

менюга

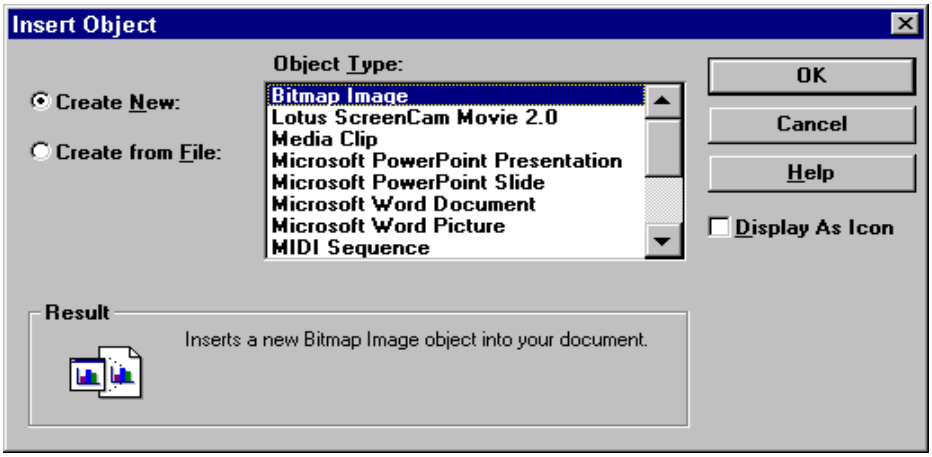
19.2. TOLEContainer объекти

TOLEContainer компонентасининг нишони **System** куроллар панелида (19.1-расм) жойлашган. Бу компонента OLE-контейнерлар иловаларини яратиш учун хизмат қилади. TOLEContainer компонентаси OLE нинг барча ички ташкилий қийинчиликларини яширади ва фойдаланувчи учун жуда ҳам қулай бўлган интерфейсни таклиф қилади. OLE объектидан фойдаланишга мўлжалланган соддагина илова яратишга уриниб кўрамиз.



19.1-расм. OLEContainer компонентасининг нишони

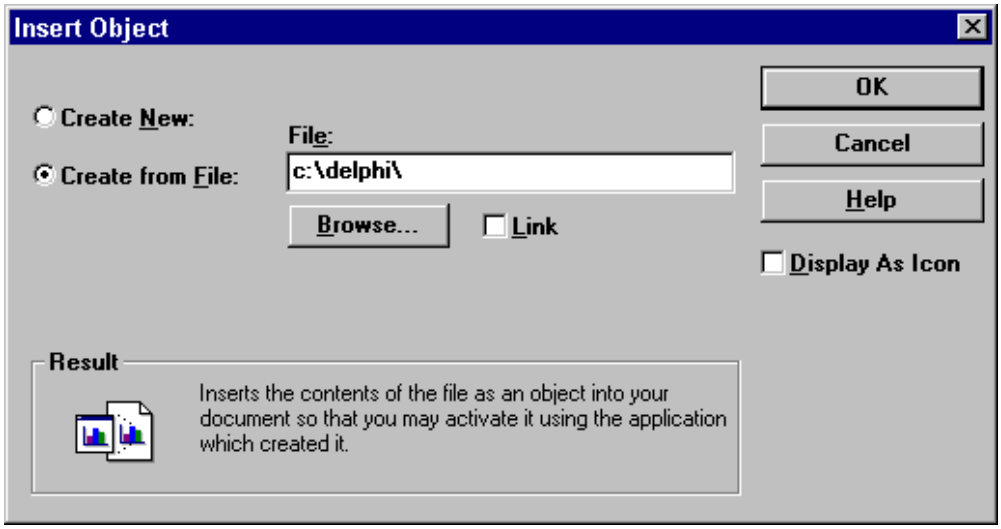
Янги лойиҳа яратиб, илованинг формасига TOLEContainer компонентасини ўрнатамиз. Сўнгра Объектлар инспектори ойнасида сичқончани ObjClass ёки ObjDoc ҳусусиятида икки марта чертамиз. Натижада, Windowsнинг стандарт “Insert Object” диалог ойнаси экранда пайдо бўлади. (19.2-расм.)



19.2-расм. “Insert Object" диалог ойнаси

Бу диалог ойнасида системада қайд қилинган барча OLE-серверлар рўйхати келтирилади. Бу қайд қилиш дастурларни шу компьютерга ўрнатилаётганда) инсталляция қилинганда қайд қилинади. OLE объектнинг типи биз кўрсатган сервер билдан аниқланади. Агар биз янги объект (Create New) яратаётган бўлсак, ОК тугмаси чертилганда OLE-сервер дастури ишга тушиб, янги объектни ҳосил қилади. Илова-сервер дастуридан чиқилганидан кейин, янги OLE объекти дастур такрибига киритилади (embedded object).

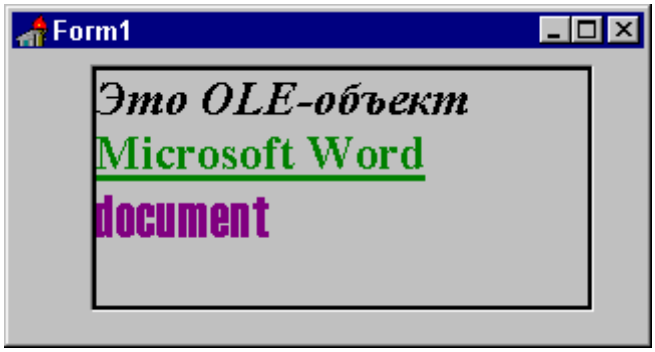
OLE объектини мавжуд файллардан фойдаланиб, OLE-сервер форматларидан бирида ҳам яратиш мумкин. Бунинг учун **Create from File** тугмасини чертиш лозим. (19.3-расм)



19.3-расм. Create from File тугмасини чертиш

Танланган объектни иловага киритиш ҳам, **Link** тугмасини босиб, бирлаштириш ҳам мумкин.

Биз илова лойиҳасини тайёрлашда Microsoft Word Document ни танлаб, янги объект ҳосил қиламиз. OK тугмасини босилса, MicroSoft Winword иловаси ишга тушади. Унда ихтиёрий матнни териш мумкин. Масалан, "*Это OLE-объект* Microsoft Word document") матни ёзилган бўлсин. Ишни тугатиш учун **File** менюсидаги **Close and Return to Form1** (Win'95 + MS Word 7.0) буйруғидан фойдаланилади. Шундан кейин иловани ишга туширсак, у тахминан 19.4-расмдагига ўхшаш тасвирни экранга чиқаради.



19.4. Янги яратилган OLE объектининг кўриниши

Сичқончани OLE-контейнерда икки марта чертилса, MS Word иловаси OLE-объектидагши ҳужжат билан ишга тушади. Бу ҳужжатни таҳрирлаш мумкин. Киритилган барча ўзгаришлар OLE-объектда сақлаб қўйилади..

Эслатма: Агар илова дастурини яратиш вақтида объектни OLE-контейнерга киритиш учун танланса, у тўлалигича форма файлига (FORM1.DFM) ёзиб қўйилади ва кейинчалик, EXE файл тарзида компиляция қилинади. Агар объект жуда ҳам катта бўлса, бу компьютер ишининг секинлашувига, узоқ таннафусларга, ҳаттоки, "Out of resource" тарзидаги ҳатоликка ҳам олиб келиши мумкин. Шунинг учун катта ҳажмли объектларни бирлаштириш (linked) тавсия қилинади.

TOLEContainer компонентаси дастурда объектларни "табиий кўринишда" экранга чиқаришга имкон беради. Агар зарурат бўлса, Zoom ҳусусияти ёрдамида уни катталаштириш ёки кичиклаштириш мумкин. Шунингдек, **Display as Icon** (19.3-расм) байроқчасини ўрнатиш билан пиктограмма шаклига келтириш мумкин.

OLE-объектини фақатгина илова дастурини яратиш жараёнида эмас, балки дастур ишлаётган вақтда ҳам танлаш мумкин. Бу объект билан ишлаш натижаларини файлда сақлаб қўйиш ва кейинчалик эхтиёжга қараб, файлдан қайта тиклаш мумкин. Бунинг учун TOLEContainer компонентаси иккита методга эга: SaveToFile ҳамда LoadFromFile методлари.

менюга

19.3. OLE иловага намуна

Delphi таркибига кирган намойиш дастурларининг ичида иккитаси OLE объект билан ишлашга алоқадор:

X :/DELPHI/DEMOS/OLE2

ҳамда

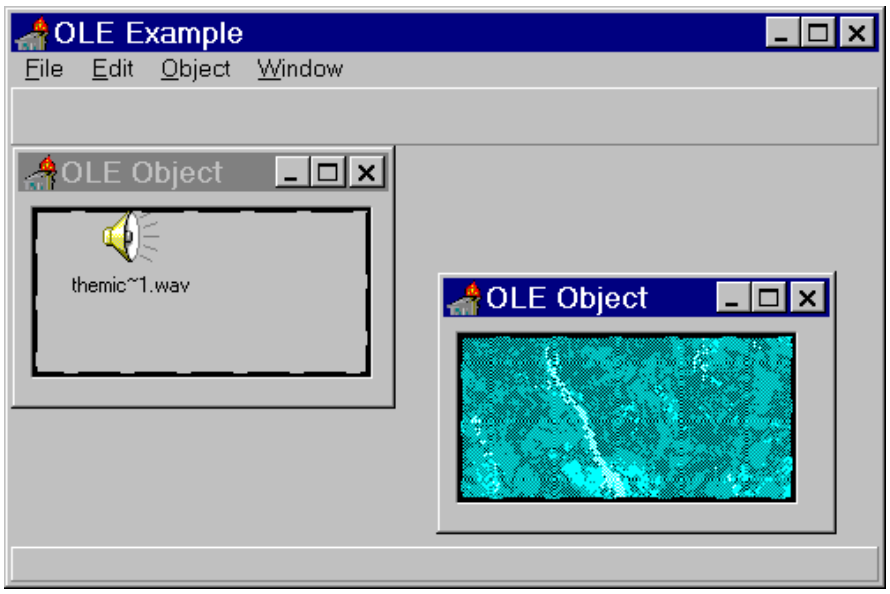
X :/DELPHI/DEMOS/DOC/OLE2.

Уларнинг ичида иккинчиси тўлароқ бўлиб, қўшимча равишда MDI илова қуришга ҳам мисол бўла олади. Бу дастур TOLEContainer компонентасининг ҳамма имкониятларини намойиш қилади ҳамда:

- дастурнинг бажарилиши жараёнида янги OLE контейнер яратишга имкон беради;

- OLE объектни ёки Windowsнинг стандар “Insert Object” диалог ойнасида, ёки Clipboard ёрдамида, ёки "олиб ўтиш ва ташлаш" техникаси (drag-and-drop) ёрдамида инициализация қилади;

- OLE объектларни файлда сақлаш ва файлдан қайта тиклаш.



19.5-расм. MDI илова

19.5-расмда MDI иловага мисол келтирилган. Унда формага иккита OLE объектли ойналар ўрнатилган.

Янги OLE объектини яратиш учун **File** менюсидан **New** буйруғи танланади. Сўнгра **Edit** менюсидан Insert Object буйруғи танланади. Экранда Windowsнинг OLE объектларини инициализация қилиш учун стандарт диалог ойнаси пайдо бўлади. (19.1-расм). Агар OLE-сервер иловаси OLE объект ҳақидаги маълумотларни Clipboard да сақлаб қўйиш имкониятига эга бўлса, OLE объектларни **Edit** менюсининг **Paste Special** буйруғи ёрдамида ҳам инициализация қилиш мумкин.

Drag-and-drop техникасини OLE объектларига нисбатан қўллаш ҳам анча қизиқ. Бунинг учун MS Word ни ишга туширамиз. (Унинг ойнасини шундай жойлаштириш керакки, OLE илова ҳам кўриниб турсин.). Бирор матнни терамиз. Шундан кейин бу матнни ажратиб олиб, сичқонча ёрдамида уни MDI илованинг бош ойнасига ташлаймиз. Экранда шу матнни ўз ичига олган OLE контейнерли қўшимча ойна пайдо бўлади. Бу имкониятни дастурлаш жараёни етарлича мураккаб.

менюга

19.4. OLE объектларни Маълумотлар базасида сақлаш

Айрим ҳолларда, OLE объектларни файлларда эмас, балки маълумотлар базасида (жадвалдаги BLOB майдони) сақлашга зарурат пайдо бўлади. Одатда, МБ даги маълумотларини бир компьютердан иккинчисигша ўтказиш талаб қилингани учун, OLE объектни ичига киритилган (embedded) кўринишида яратиш тавсия қилинади. Бахтга қарши, Delphi да бундай мақсадлар учун махсус TDBOLEContainer типидаги компоненталар киритилмаган, аммо, OLE объектларни SaveToStream методи билан сақлаб қўйиб, кейинчалик LoadFromStream методи билан қайта тиклаш мумкин.

```
procedure TOLEForm.SaveOLE(Sender: TObject);
var
BlSt : TBlobStream;
begin
With Table1 do
BlSt :=TBlobStream.Create(BlobField(FieldByName('OLE')),
bmReadWrite);
OLEContainer.SaveToStream(BlSt as TStream);
BlSt.Free;
end;
```

менюга

мундарижага қайтиш

20-БОБ. ЎРНАТУВЧИ ДИСКЛАР ЯРАТИШ

20.1. Бошланғич маълумотлар

Барча замонавий дастурлар компакт-дисклардa тарқатилмоқда. Дастурларни бошқа компьютерларга ўрнатиш жараёни одатда фақатгина махсус каталогларни яратиб, бу каталогларга бажарилувчи файллар ва уларнинг ёрдамчи файлларини кўчириб қўйишнигина эмас, балки системани ҳам созлашдан иборат бўлади. Системани созлаш масаласини ҳар қандай фойдаланувчи ҳам ҳал қила олмайди. Шунинг учун одатда, амалий дастурлар дастасини бошқа компьютерга ўрнатишни махсус дастурлар зиммасига юкланади. Бу файл ҳам амалий дастурлар дастаси ёзилган дискда сақланади. Демак, дастурчидан асосий масалани ҳал қилишдан ташқари, ўрнатувчи (инсталляцион) дастурни яратиш ҳам талаб қилинади.

Инсталляцион дастурлар ҳам бошқа дастурлар каби яратилади. Инсталляция жараёнида ҳал қилиниши керак бўлган масалалар ҳам одатдаги масалалар ҳисобланади. Бу масалаларни ҳал қилиш учун керакли барча воситалар мавжуд ва бу воситалардан фойдаланиб, ҳаттоки бир сатр буйруқ ёзмай, осонгина ўрнатувчи дастурларни яратиш мумкин.

менюга

20.2.InstallShield Express дастури

Ўрнатувчи дастурлар яратиш учун энг кўп қўлланадиган дастурлардан бири – бу InstallShield Express пакетидир. Borland фирмаси айнан шу дастурдан фойдаланишни тавсия қилади ва бу дастур Borland Delphi 7 Studio дастурининг барча ўрнатувчи дисклари таркибига киритилган.

InstallShield Express дастурини одатдаги усуллар билан ўрнатиш мумкин. Уни фаоллаштириш учун Delphi ни ўрнатиш дастурини ишга туширилади. (ўрнатувчи CD-ROM дискини диск юритувчига қўйилади.) сўнгра очилган **Delphi Setup Launcher** диалог ойнасидан **InstallShield Express — Borland Limited Edition** буйруғини танланади. Натижада дастурларни ўрнатиш устаси ишга тушади.

Ўрнатиш жараёни тугаганидан кейин, Windows бош менюсидаги **Пуск / Программы / InstallShield** тугмалари ёрдамида экранда **Express** буйруғи пайдо бўлади, Уни чертиб, **InstallShield Express** дастурини ишга туширилади.

Ўрнатувчи дисклар яратишни конкрет мисол орқали изоҳлаймиз.

Бизнинг олдимизга квадрат тенглама дастури учун инталляцион диск яратиш масаласи қўйилган бўлсин. Бевосита бу масала учун ўрнатувчи дастур яратишдан аввал, биз бошқа компьютерларга ўтказилиши талаб қилинган файллар рўйхатини аниқлаб олишимиз даркор. Бундан ташқари, матн муҳарриридан фойдаланиб, лицензион келишувнинг RTF-файли (EULA — End User Licensia Agreement) ҳамда қисқа маълумотнома файли (Readme-файл) ҳам бизга керак бўлади. Бошқа компьютерга ўтказилиши керак бўлган файллар рўйхати 20.1-жадвалда берилган.

Квадрат тенглама дастури учун файллар рўйхати 20.1-жадвал.

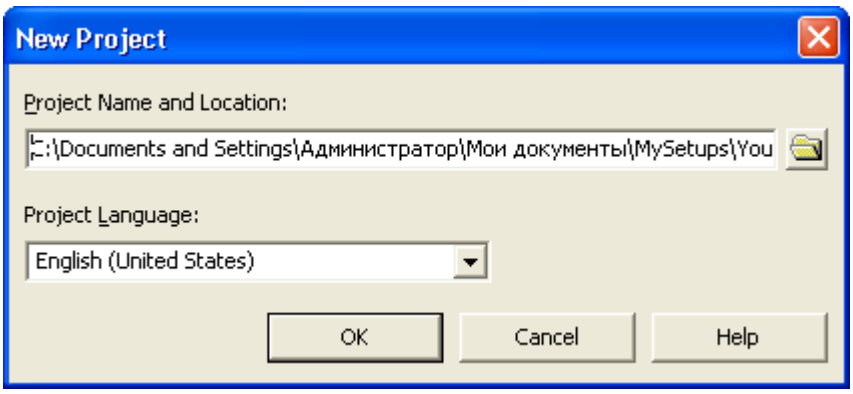
Файл	Вазифаси	Қаерга ўрнатилади

Kw_teng.exe	дастур	Program Files/kw_t
SqRoot.Hlp	маълумотнома файли	Program Files/kw_t
Readme.rtf	дастур ҳақида қисқа маълумотнома	Program Files/kw_t
Eula.rtf	Лицензион келишув файли	Program Files/kw_t

менюга

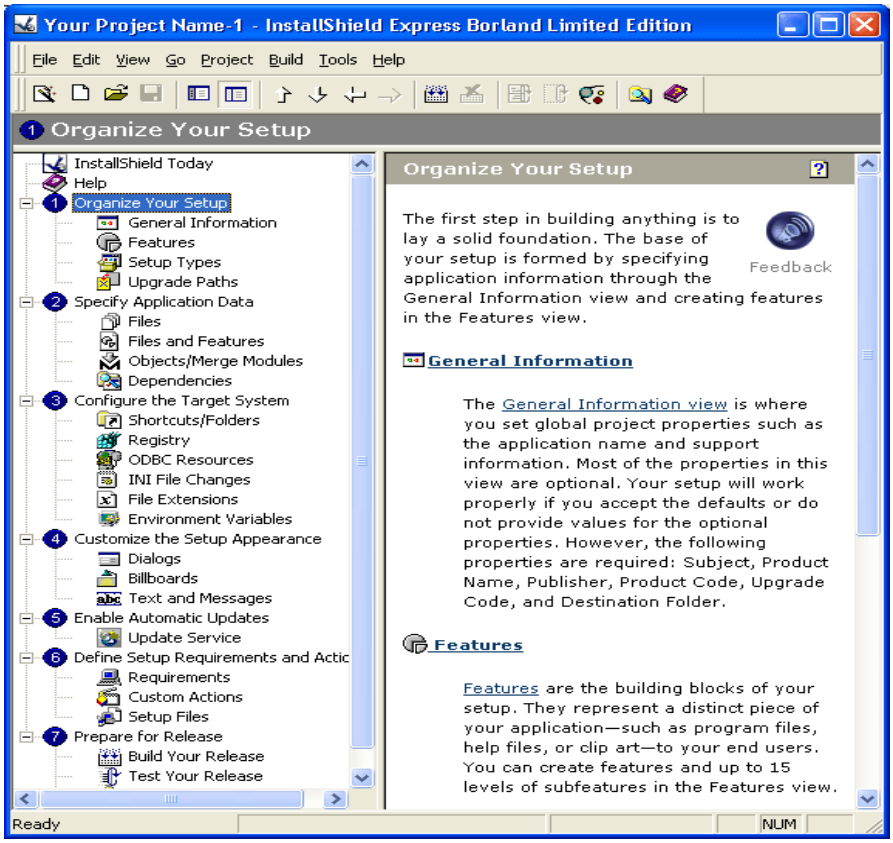
20.3. Янги лойиҳа

Файллар рўйхати аниқ бўлганидан кейин, InstallShield Express ни ишга туширамиз ва **File** менюсидан **New** буйруғини танлаймиз ҳамда **Project Name and Location** майдонига лойиҳа фйли номини киритамиз. (20.1-расм).



20.1. Янги лойиҳа устида иш бошлаш

ОК тугмаси чертилганидан сўнг, инсталяцион дастур лойиҳасини ёзиш ойнаси очилади (20.2-расм). Ойнанинг чап томонида инсталляцион дастур яратиш босқичлари ҳамда параметрларини кўрсатиш учун буйруқлар рўйхати келтирилган.



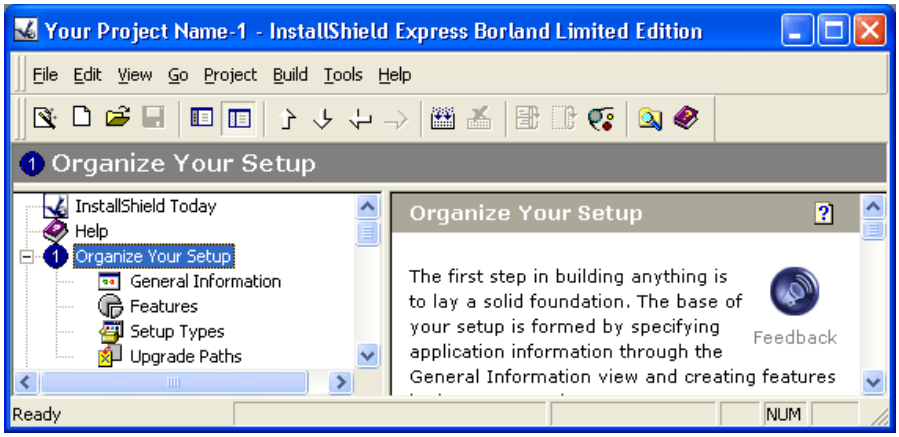
20.2. Янги лойиҳа ойнасининг умумий кўриниши

менюга

20.4. Инсталляцион дастур структураси

Organize Your Setup гуруҳининг (20.3-расм) буйруқлари ўрнатиш дастури структурасини белгилашга имкон беради.

Ўрнатиладиган дастур ва унинг ишлаб чиқарувчиларни кўрсатувчи параметрлардан бошқа кўплаб параметрларни ўзгаришсиз қолдириш мумкин. Ўзгартирилиши талаб қилинган параметрлар рўйхати 20.2-жадвалда берилган.



20.3-расм. Organize Your Setup гуруҳининг буйруқлари

General Information буйруғининг параметрлари 20.2-жадвал

Параметр	**мазмуни**	**қиймати**
Product Name	Ўрнатилаётган дастурнинг номи	Kw_teng

Product Version	Ўрнатилаётган дастурнинг версияси	1.01.0001
INSTALLDIR	Дастур ўрнатилиши талаб қилинадиган фойдаланувчи компьютеридаги каталог номи	[ProgramFilesFolder] KW_T

INSTALLDIR параметрига эътибор беринг. Агар дастур ўрнатиладиган каталог номи кўрсатилмаса, бу дастур дастурлар учун мўлжалланган каталогга ўрнатилади. Фойдаланувчининг компьютерида бу каталогнинг номи ва қайси дискда жойлашганини билиб бўлмагани учун, каталог ўрнига унинг [ProgramFilesFolder] таҳаллусидан фойдаланамиз. Фойдаланувчининг компьютерига дастурни ўрнатиш жараёнида инсталляцион дастур Windows реестридан дастурлар каталогининг номини олади ва таҳаллусни шу ном билан алмаштиради.

InstallShield Express дастури фойдаланадиган бошқа таҳаллуслар 20.3-жадвалда берилган.

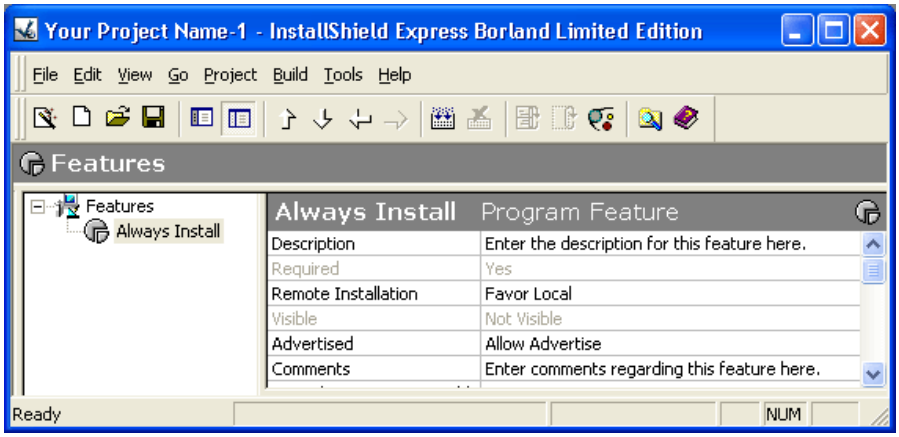
Windows нинг айрим каталогларининг таҳаллуслари. 20.3-жадвал

Таҳаллус	**Каталог**
[WindowsVolume]	Windows жойлашган она (ўзак) каталог
[Windows Folder]	Windows каталоги, масалан, C:/Winnt
[SystemFolder]	Windows нинг системали каталоги, масалан, C:/Winnt/System32
[ProgramFilesFolder]	Программалар каталоги, Масалан, C:/Program Files
[PersonalFolder]	Ишчи столидаги **Мои документў** папкаси. (Бу папканинг жойлашиши ОС версиясига ҳамда системага кириш усулларига боғлиқ.)

Ўрнатилган дастурнинг имкониятлари ўрнатилган компонентлар таркиби билан аниқланиши табиий. Масалан, агар ёрдамчи маълумотномалар системаси файли ўрнатилган бўлса, у ҳолда фойдаланувчи ушбу дастурдан фойдаланиш жараёнида бу системадаги ахборотларни олиши мумкин. **Features** (имкониятлар) буйруғи дастурнинг имкониятларини аниқловчи ҳамда алоҳида ўрнатилиши мумкин бўлган компонентлар гуруҳини белгилайди. Компонентларни гуруҳларга бўлиш кўп вариантли, шу жумладан, фойдаланувчи белгилайдиган компонентлар гуруҳини ўрнатувчи дастурларни ёзишга имкон беради.

Энг содда ҳолда **Features** гуруҳи битта **Always Install** элементидан иборат бўлади. **Features** гуруҳига янги элемент қўшиш учун сичқончанинг стрелкасини **Features** гуруҳларига келтирилади ва ўнг тугмаси чертилади. Экранда пайдо бўлган контекст менюсидан **New Feature Ins** буйруғи танланади ва янги гуруҳ номи киритилади. Масалан, Help Files and Samples. Шундан кейин **Description** майдонига элементнинг қисқа харатеристикасини, **Comments** — майдонига эса изоҳларни ёзилади.(20.4-расм).

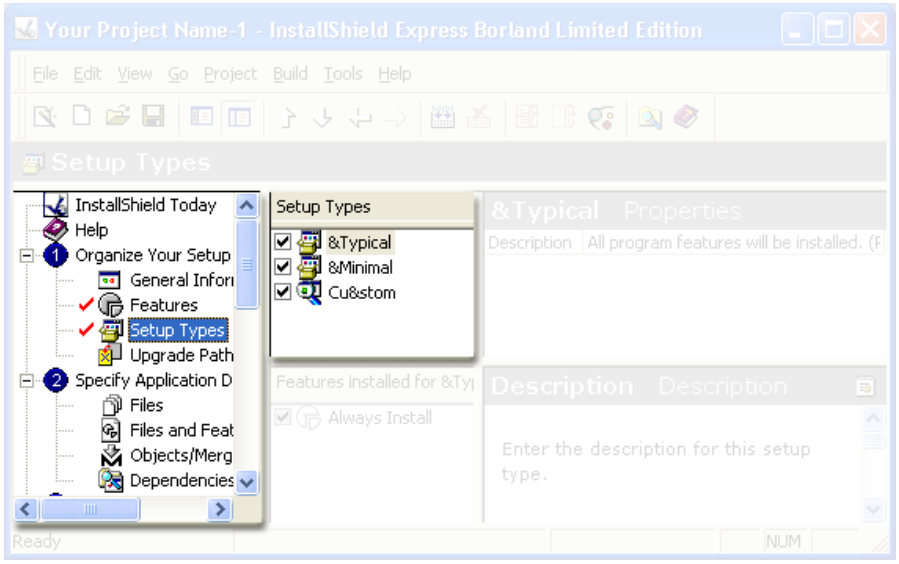
Setup Types буйруғи фойдаланувчига дастурни ўрнатиш жараёнида (**Setup Type** диалог ойнасида) ўрнатиш вариантини танлаш имкони берилишини белгилайди. Маълумки, дастурларни ўрнатиш жараёни оддий **(Typical),** минимал **(Minimal)** ёки танланадиган (**Custom)** бўлиши мумкин**.** Агар ўрнатиладиган дастур



19.4. Features гуруҳидаги бир нечта элемент кўп вариантли ўрнатишни таъминлайди.

мураккаб, яъни бир нечта компоненталардан иборат бўлса, одатда бундай имконият фойдаланувчиларга берилади.

Kw_teng дастури учун ўрнатиш варианти битта - **Typical.** Шунинг учун **Minimal** ва **Custom** байроқчаларини ўрнатмаса ҳам бўлади. (20.5-расм).

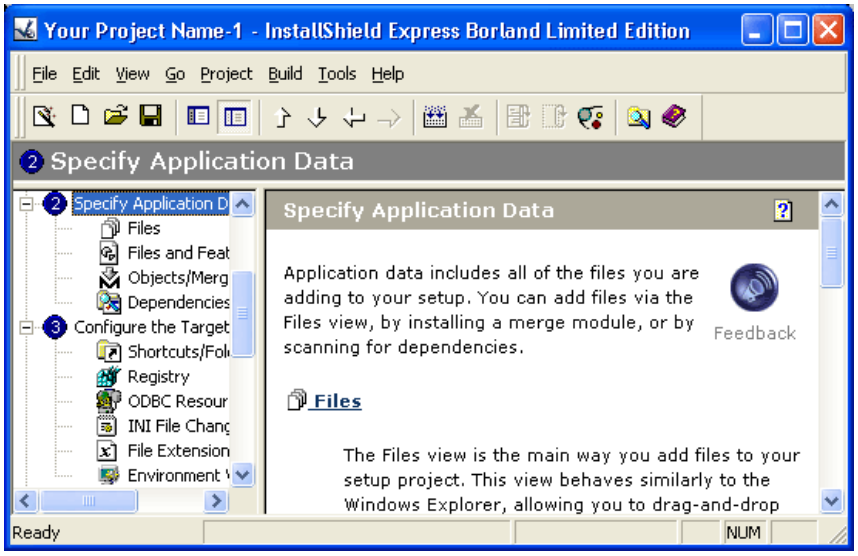


20.5-расм. Setup Types буйруғи кўп вариантли ўрнатиш режимини аниқлайди.

менюга

20.5. Ўрнатиладиган компоненталарни танлаш

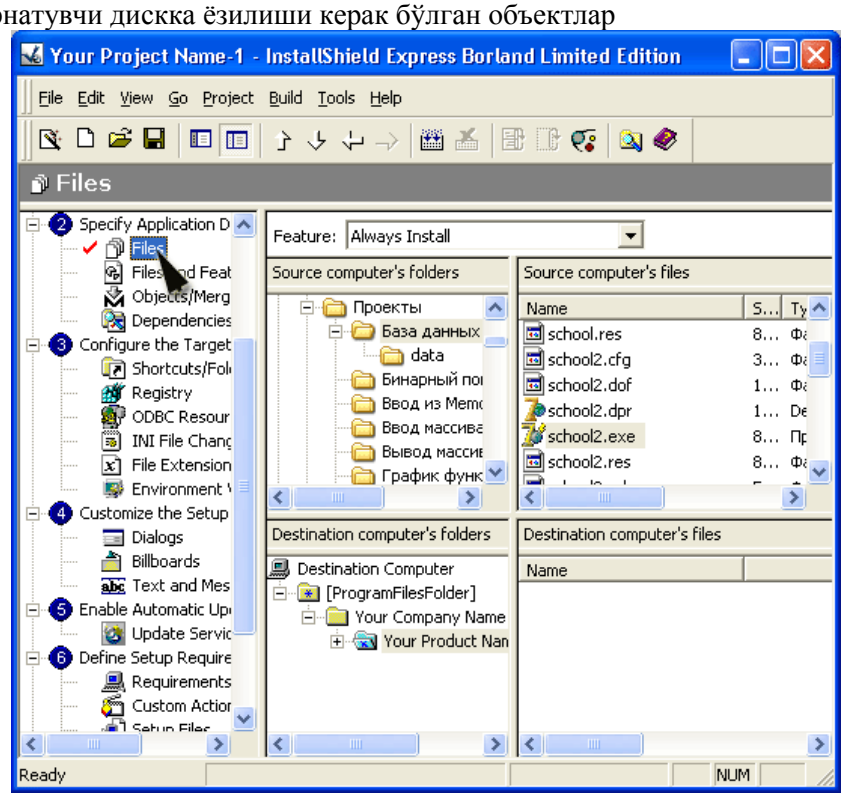
Specify Application Data (20.6-расм) гуруҳининг буйруқлари фойдаланувчининг компьютерига ўрнатилиши талаб қилинган дастур компоненталарини белгилашга имкон беради. Агар лойиҳада компоненталарнинг бир неечта гуруҳи белгиланган бўлса, (Features буйруғига қаранг), у ҳолда ҳар бир гуруҳ учун компоненталар таркибини аниқлаш лозим.



20.6-расм. Specify Application Data гуруҳининг буйруқлари

Files буйруғини танлаш натижасида экран бир нечта соҳаларга (20.7-расм) бўлиниб кетади. **Source computer's files** соҳасида фойдаланувчининг компьютерига ўтказилиши керак бўлган файлларни танлаш мумкин. **Destination computer's folders** соҳасида эса бу файллар ўрнатиладиган папкани танланади. +айси файлларни фойдаланувчининг компьютерига ўтказилишини кўрсатиш учун бу файлларни сичқонча билан тўғридан-тўғри **Source computer's files** соҳасидан **Destination computer's files** соҳасига олиб ўтиш керак. Агар **Features** гуруҳида бир нечта элементлар бўлса, ҳар бир элементлар учун файллар кўрсатиш лозим.

Object/Merge Modules буйруғи қайси объектлар, масалан, динамик кутубхоналар ёки компоненталар пакети бошқа компьютерга, шунингдек ўрнатувчи дискка ўтказилиши кераклигини белгилаб беради. Ўрнатувчи дискка ёзилиши керак бўлган объектлар



20.7-расм. Бошқа компьютерга ўтказиладиган файлларни танлаш

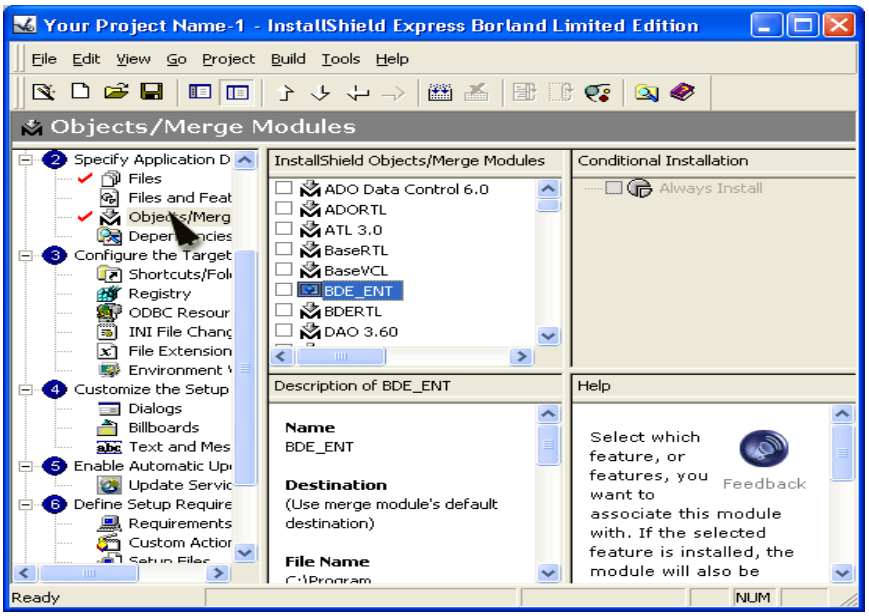
InstallShield Objects/Merge Modules (20.8-расм) рўйхатидан олинади.

менюга

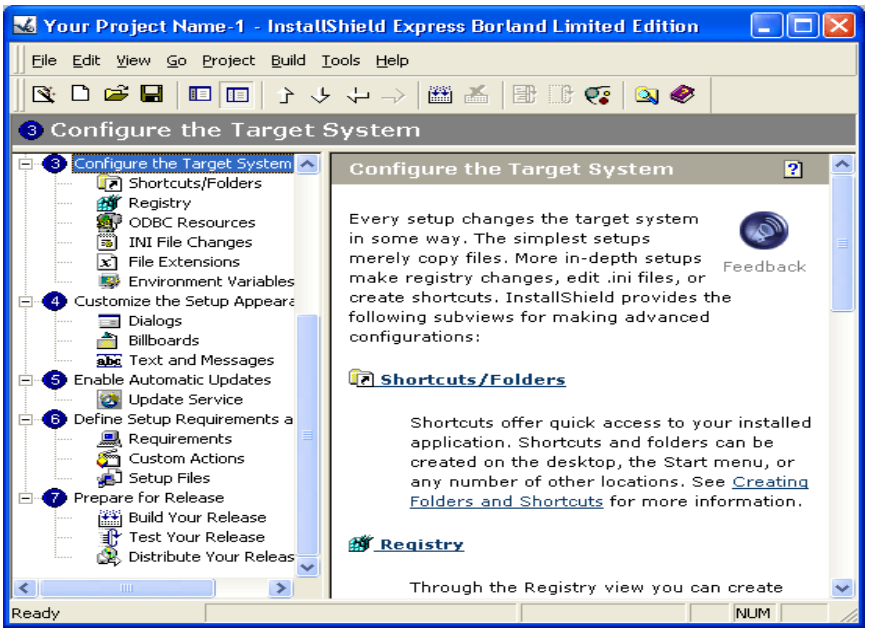
20.6. Фойдаланувчи компьютери системасини созлаш

Configure the Target System (20.9-расм) гуруҳининг буйруқлари фойдаланувчи компьютери ўрнатилган дастур билан ишлай олиши учун системага қандай ўзгаришлар киритилиши кераклигини белгилаб беради.

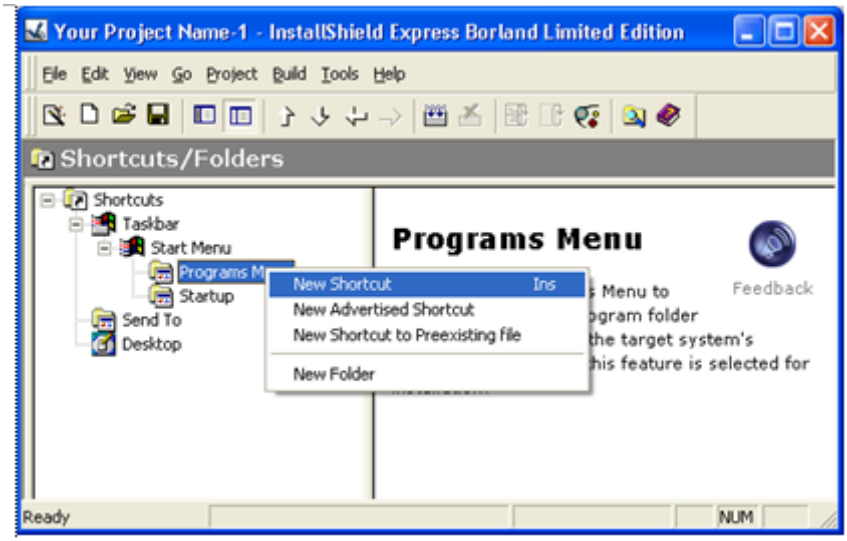
Shortcuts/Folders буйруғи ўрнатиладиган дастурнинг ишга тушириш ёрлиғини қаерга ўрнатиш лозимлигини кўрсатади. Бу буйруқ танланганида, ойнанинг ўнг томонида меню ва папкаларнинг дарахтсимон рўйхати пайдо бўлади. Бу рўйхатдан ёрлиқни жойлаштириш учун керакли менюни танлаш зарур. Сичқончанинг ўнг тугмаисни босиб, очилган рўйхатдан **New Shortcut** (20.10-расм) буйруғини танланади.



20.8. Бошқа компьютерга ўрнатиладиган объектларни танлаш

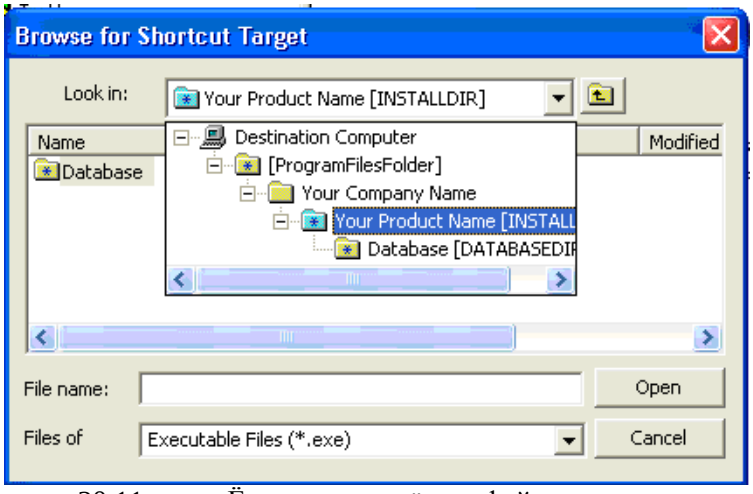


20.9-расм. Configure the Target System гуруҳининг буйруқлари

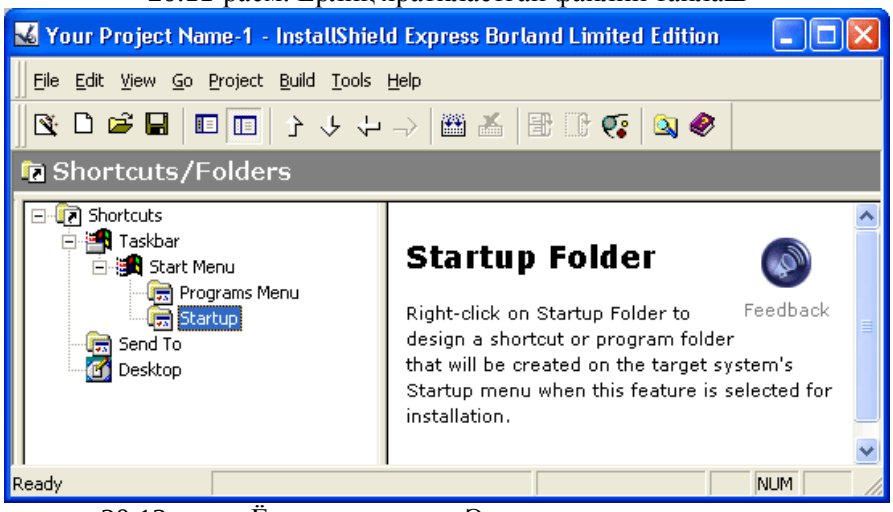


20.10-расм. Shortcuts рўйхатидан ёрлиғи учун меню танланади.

Сўнгра, **Browse for Shortcut Target** диалог ойнасида дастур файлини танлаб, (20.11-расм), **Open** тугмаси чертилади ва ёрлиқ номи киритилади. Шундан кейин ёрлиқни якуний созлаш мумкин. Масалан, **Arguments** майдонида буйруқлар сатрининг параметрлари, **Working Directory** — майдонида эса ишчи каталоги (20.12-расм) кўрсатилади.



20.11-расм. Ёрлиқ яратилаётган файлни танлаш



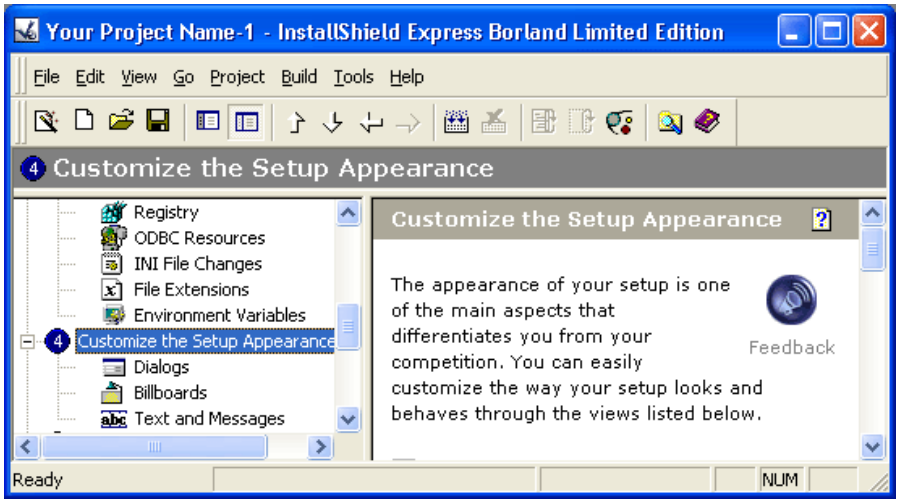
20.12-расм. Ёрлиқ яратилди. Энди уни созлаш мумкин.

менюга

20.7. Диалог ойнасини созлаш

Фойдаланувчи билан биргаликда ишлаш учун, ўрнатиш дастури стандарт диалог ойналаридан фойдаланади. Ўрнатувчи дастурни яратар экан, дастурчи фойдаланувчилар учун дастурларни ўрнатиш

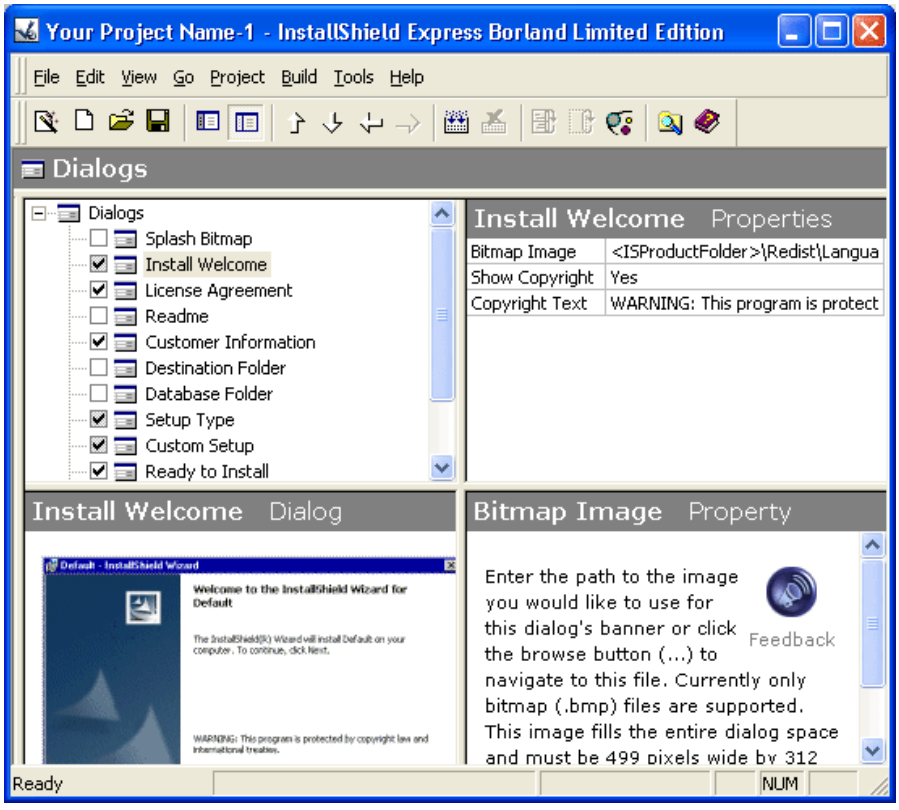
жараёнида қайси диалог ойналарини кўриши мумкинлигини белгилаб қўйиши мумкин.



20.13-расм. Customize the Setup Appearance гуруҳининг буйруқлари

Ўрнатувчи дастур ишлаётган вақтда экранга чиқиши керак бўлган диалог ойналарини белгилаб қўйиш учун, **Customize the Setup Appearance** (20.13-расм) буйруқлар гуруҳидан **Dialogs** буйруғи танланади ва очилган **Dialogs** (20.14-расм) рўйхатидан ўрнатиш дастурига қўшиладиган диалог ойналари танланади.

Properties жадвалида (диалог ойналари рўйхатининг ўнг томонида) танланган диалог ойнасининг ҳусусияти санаб ўтилган. Дастурчи бу ҳусусиятларнинг қийматларини ўзгартириши, яъни диалог ойнасини созлаши мумкин. Масалан, **Readme** диалог ойнаси учун файл номини кўрсатиш лозим (**Readme File** ҳусусияти**).** Одатда бу файлда ўрнатилаётган дастур ҳақидаги қисқа ахборот сақланади.



20.14-расм **Dialogs** рўйхатидан ўрнатиш дастурига қўшиладиган диалог ойналарини танлаш.

Кўпчилик диалог ойналари учун баннер (**Banner Bitmap ҳусусияти)** — расмларни танлаш мумкин. Бу расмлар диалог ойнасининг юқори қисмига чиқарилади. Баннер файлининг формати — BMP, ўлчамлари эса — 499x58 пиксел.

20.4-жадвалда ўрнатувчи дастур ишлаётган вақтда экранга чиқарилиши мумкин бўлган айрим диалог ойналарининг рўйхатини келтирамиз.

Ўрнатиш жараёнининг диалог ойналари. 20.4-жадвал

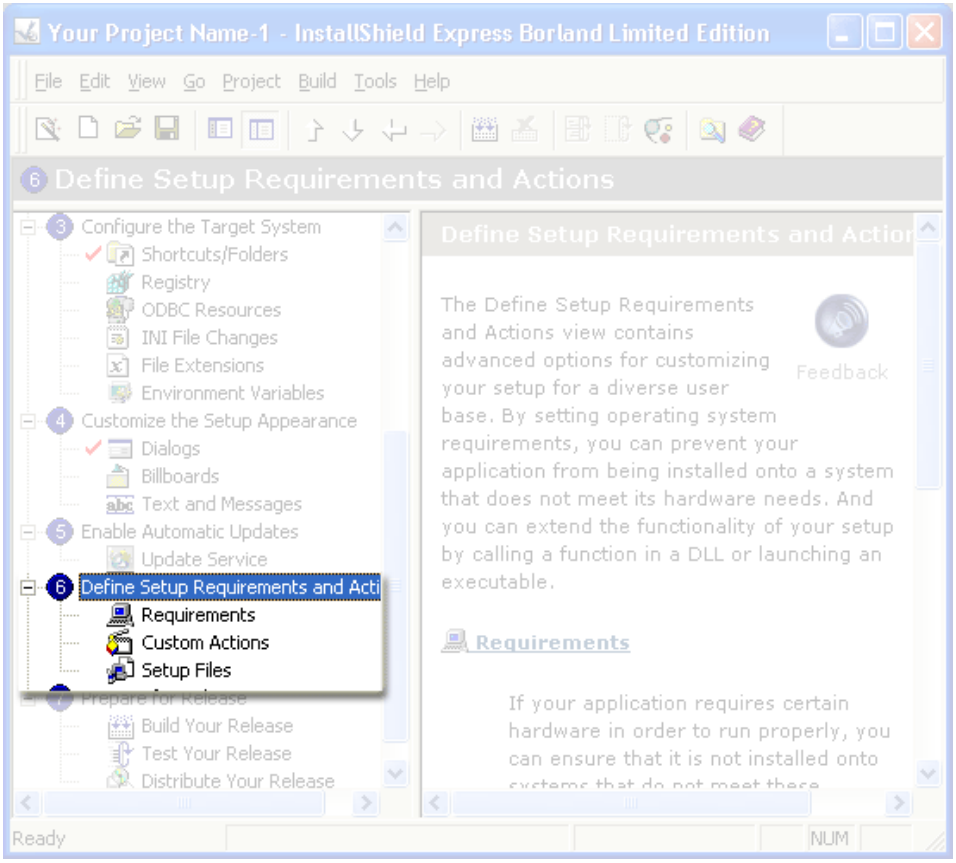
Диалог ойнаси	Вазифаси
Splash Bitmap	Ўрнатилаётган дастур ҳақидаги расмни чиқариш. Расм ўлчами - 465x281 пиксел, формати –BMP.
Install Welcome	Расм фонида ахборотни чиқариш. (ўлчами 499x312 пиксел.)
License Agreement	RTF-файлидаги лицензион келишув ҳақидаги ахборот. Агар келишув бўлмаса, ўрнатиш жараёни тўхтатилади.
Readme	Ўрнатиладиган дастур ҳақидаги ахборот чиқариш
Customer Information	Фойдаланувчи (исми, ташкилот номи) ҳақидаги маълумотларни, ўрнатилаётган нусханинг серия номерини сўрайди.
Destination Folder	Фойдаланувчига ўрнатилаётган дастур учун каталог танлашга имкон беради.
Database Folder	Фойдаланувчига маълумотлар базаси учун каталог танлашга имкон беради.
Setup Type	Фойдаланувчига дастур ўрнатиш типини белгилаш имкони беради. (Typical — оддий, Minimal — минимал, Custom — танлаш)
Custom Setup	Фойдаланувчига дастурларни танлаб ўрнатишда (custom) компоненталар рўйхатини белгилаш имконини беради.
Setup Complete Success	Ўрнатиш жараёни тугаганлиги ҳақида ахборот беради. Шундан сўнг, ишга тушиши керак бўлган дастур номини, шунингдек Readme файлдаги маълумотларни кўрсатишга имкон беради.
Setup Progress	Ўрнатиш жараёни вақтида бажарилган иш фоизини кўрсатади.
Ready to Install	Фойдаланувчи олдинги қадамларда киритган маълумотларни ўрнатиш жараёнини бошлашдан аввал текшириш мақсадида чиқариш.

Диалог ойналари инсталляцион дастур ишлаётган вақтда экранга чиқиши учун диалог ойнасининг ёнида турган байроқчани тиклаш лозим. **License Agreement и Readme** ойналари учун мос ахборотлар сақланаётган RTF-файлларнинг номи кўрсатилиши лозим. Энг содда ҳолда, ўрнатиш дастури қуйидаги диалог ойналарини чиқариш билан чегараланиши мумкин: Readme; Destination Folder; Ready to Install; Setup Progress; Setup Complete Success.

менюга

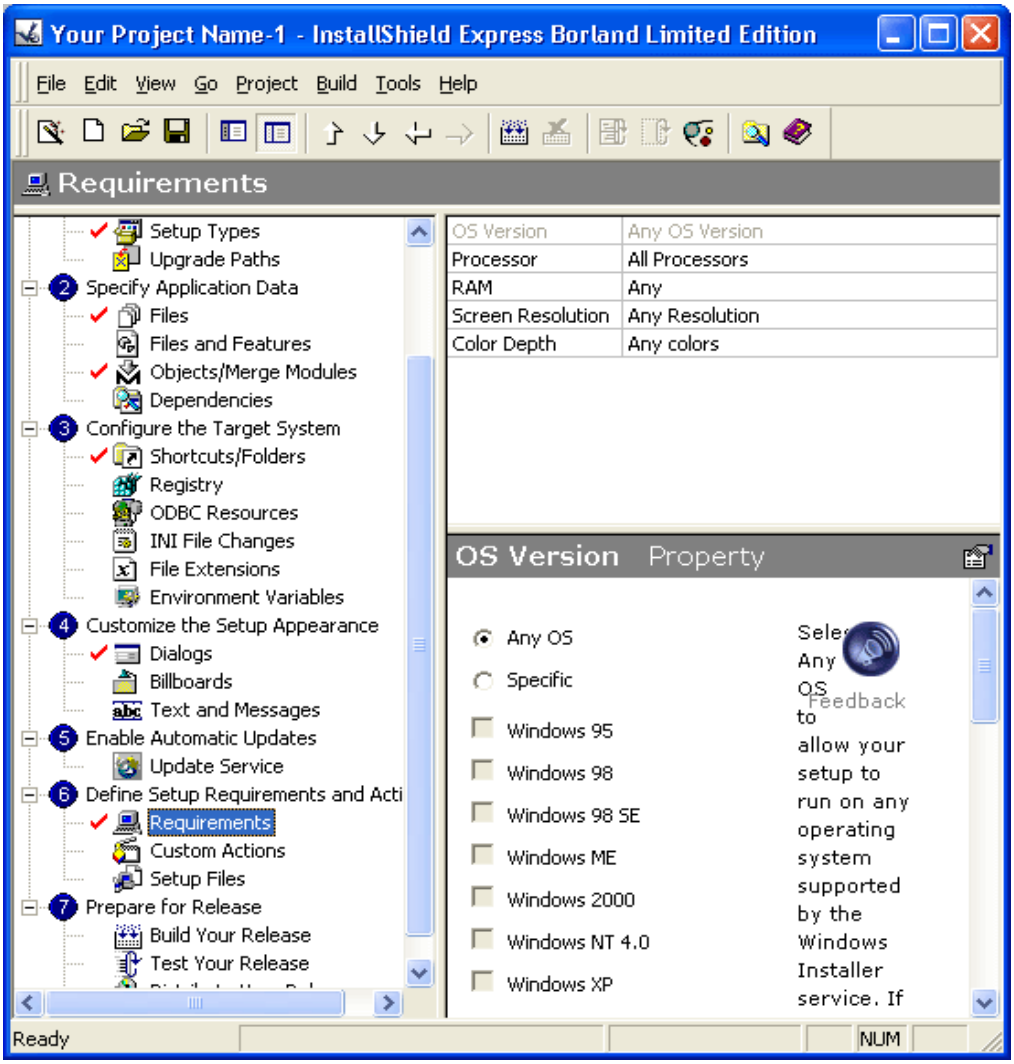
20.8. Системага бўлган талаблар

Агар ўрнатилаётган дастур система ресурсларига қандайдир талаблар қўядиган бўлса, бу талабларни **Define Setup Requirements and Actions** (20.15-расм) буйруқлар гуруҳи ёрдамида белгиланиши мумкин.



20.15-расм. Define Setup Requirements and Actions гуруҳининг буйруқлари

Requirements буйруғи танланганда экранда 20.16-расмдаги жадвал пайдо бўлади. Бу жадвалга системанинг асосий характеристикаларини билдирувчи параметрларнинг қийматларини киритиш лозим: операцион система версияси (OS Version), процессорнинг типи (Processor), оператив хотира хажми (RAM), экраннинг нуқталар сиғими (Screen Resolution) ҳамда ранглар палитраси (Color Depth). Характеристикаларнинг қийматларини параметрнинг қиймати майдонида турган нишонни чертилганда очиладиган рўйхатдан танланади.



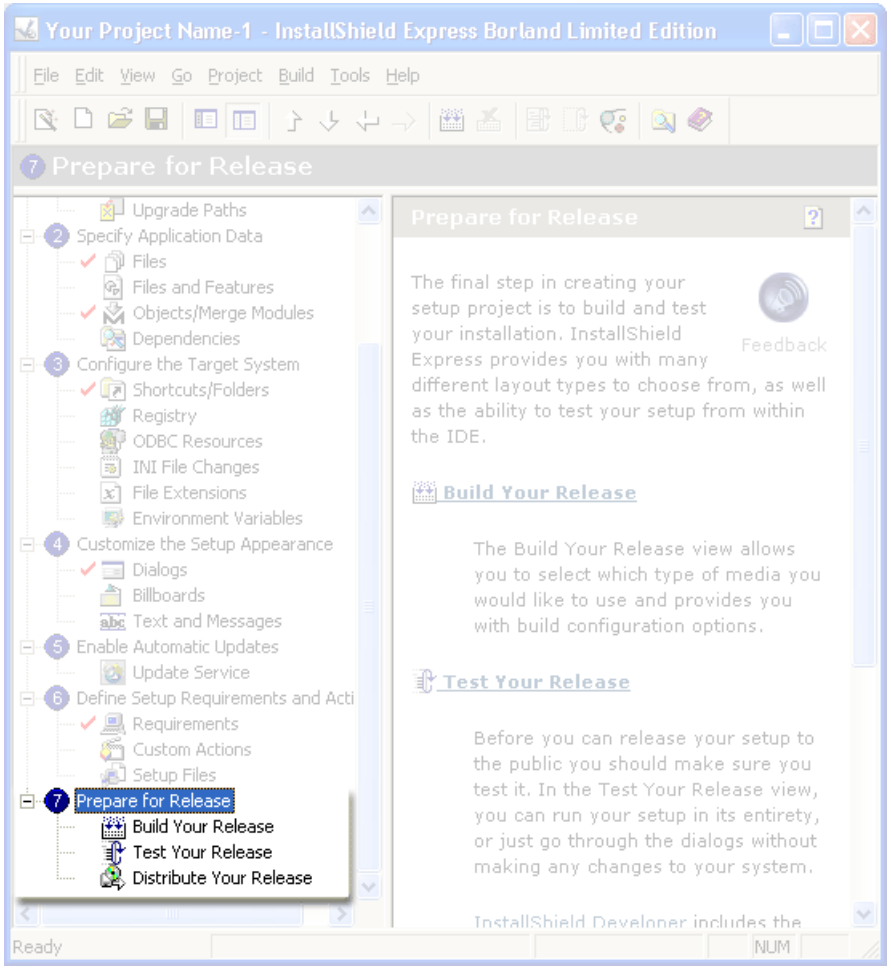
20.16-расм. Системани характерловчи параметрлар.

Агар дастур система конфигурациясига ортиқча махсус талаблар қўймаса, **Define Setup Requirements and Actions** буйруқлар гуруҳига тегмаса ҳам бўлаверади.

менюга

20.9. Ўрнатувчи диск образини яратиш

Prepare for Release (20.17-расм) гуруҳининг буйруқлари ўрнатувчи диск (CD-ROM) образини яратишга ҳамда қандай ишлашини текшириш имконини беради.

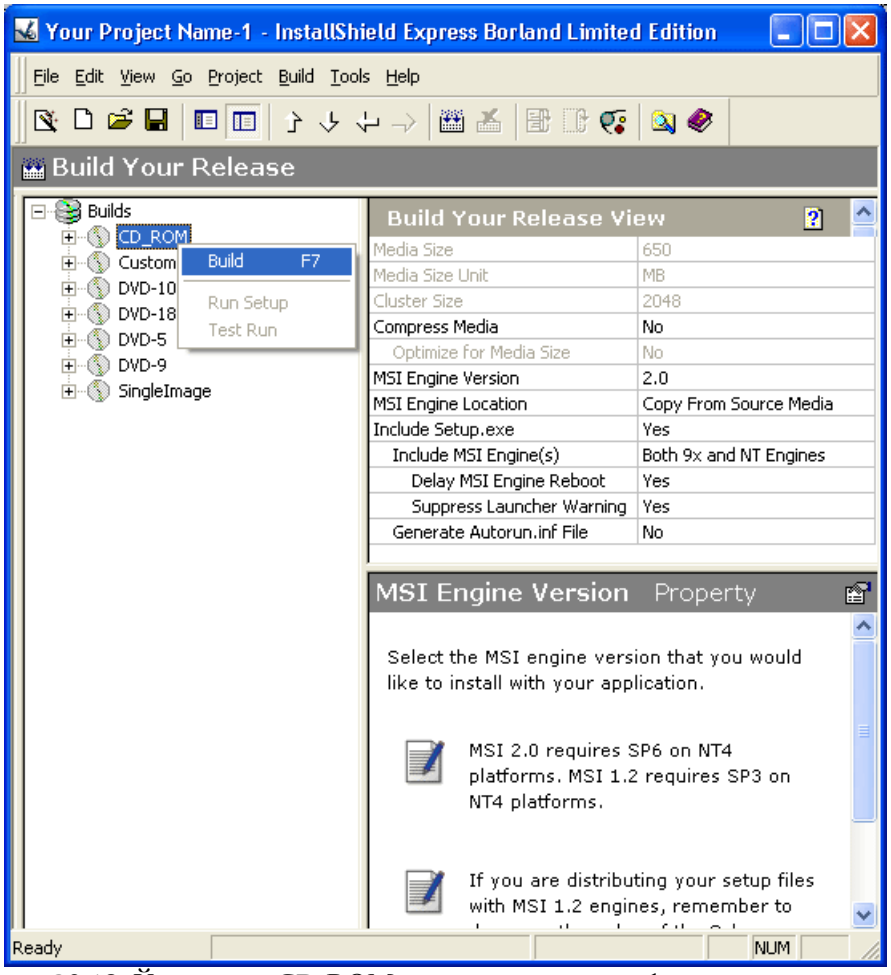


20.17. Prepare for Release гуруҳининг буйруқлари

Ўрнатувчи диск (CD-ROM) образини яратиш жараёнини ишга тушириш учун **Build Your Release** буйруғини танланади ва сичқончанинг ўнг тугмасини ўрнатувчи дастурни ёзиш мўлжалланган дискнинг нишонида чертилади. Очилган контекст менюсидан эса **Build** (20.18-расм) буйруғини танланади. Натижада компьютер дискидаги лойиҳа папкасида ўрнатувчи дискнинг образи яратилади. Агар диск сифатида CD-ROM танланган бўлса, образ

/Express/Cd_rom/DiskImages/Disk1

каталогига жойланади.



20.18. Ўрнатувчи CD-ROM яратиш жараёнини фаоллаштириш

менюга

Mundarija