ЈУЛИЈАНА ПЕТРЕСКА

ПРОГРАМИРАЊЕ

за IV година за профилот електротехничар за компјутерска техника и автоматика

Автор:
Јулијана Петреска,
професор по ОЕМУЦ „Св. Наум Охридски“ - Охрид

Илустратор:
Јулијана Петреска,
професор по ОЕМУЦ „Св. Наум Охридски“ - Охрид

Рецензенти:
д-р Иван Чорбев,
доцент на Факултетот за електротехника и информациски технологии при Универзитетот „Св. Кирил и Методиј“ - Скопје

Билјана Пејовска,
професор во СЕТУ на град Скопје „Михајло Пупин“ - Скопје

Даниела Сандева,
професор во СЕТУ на град Скопје „Михајло Пупин“ - Скопје

Лектор:
Ленче Јовановска

Издавач:
Министрство за образование и наука на Република Македонија

Печати:
Графички центар дооел, Скопје

Тираж:
900

Со решение на Министерот за образование и наука на Република Македонија бр. 22-4303/1 од 28.07.2010 година се одобрува употребата на овој учебник

CIP – Каталогизација во публикација
Национална и универзитетска библиотека “Св.Климент Охридски” , Скопје
004.42(075.3)
ПЕТРЕСКА, Јулијана
Програмирање за IV година за профилот електротехничар за компјутерска техника и автоматика / Јулијана Петреска. - Скопје : Министерство за образование и наука на Република Македонија, 2010. - 186 стр. : илустр. ; 30 см
ISBN 978-608-226-154-6
COBISS.MK-ID 84272138

СОДРЖИНА

1.ОСНОВИ НА ОБЈЕКТНО ОРИЕНТИРАНО ПРОГРАМИРАЊЕ.....7
1.1 Основни карактеристики на програми засновани на прозорци.....7
1.2. Елементи на графичка корисничка површина.....9
1.3. Програми управувани со настани.....13
1.4. Проблеми кои се решаваат со Delphi.....16
ПОГЛАВЈЕ 1 (НАКРАТКО).....17
Прашања и задачи:.....18

2.АНАЛИЗА НА DELPHI ЕКРАНОТ.....19
2.1. Вовед во работната околина на Delphi.....19
2.2 Интерпретер и компајлер 24
2.3 Некои компоненти од палетата на компоненти 25
2.4 Празен проект.....26
2.4.1. Чување и отворање на проектот.....26
2.5. Образец (форма) и нагодување на неговите својства.....28
2.6. Додавање компоненти во формата (Label, Button, Edit) 29
ПОГЛАВЈЕ 2 (НАКРАТКО).....33
Прашања и задачи:.....34

3.ПИШУВАЊЕ НА КОД ВО OBJECT PASCAL.....35
3.1 Мојот прв Delphi проект. Програма Здраво 35
3.2.Елементи на јазикот.....38
3.3.Податоци.....38
3.4.Оператори и изрази.....39
3.5.Наредба за доделување на вредност.....39
3.5.1. Објект Canvas….....41
3.5.2. Timer компонента 42
3.6.Проектирање на апликации од линиска структура.....44
3.6.1.Претворање на податоци.....46
3.7.Разгранети структури.....48
3.8. Компоненти за избор.....50
3.8.1. Check Box контрола.....50
3.8.2. RadioButton контрола.....54
3.8.3 Radio group контрола.....57
3.8.4.Компонента ListBox 63
3.8.5 Компонента ComboBox 67
3.9 Контејнерски компоненти.....69
3.9.1.Компонента GroupBox 69
3.9.2.Panel.....70
3.9.3. Bevel (рамка) 70
3.10. Програмски модули.....71
3.10.1Општ облик на програмскиот модул 71
3.10.2.Опсег на идентификаторот 71
3.11.Циклични структури 75
3.11.1.WHILE-DO наредба.....75
3.11.2.REPEAT UNTIL наредба.....76
3.11.3 FOR naredba 77
3.11.3.1. FOR-TO-DO (За зголемувај до) 77
3.11.3.2. FOR-DOWNTO-DO (За намалувај до) 77
3.12.Компоненти за работа со низи 78
3.12.1. Компонентата (Memo) 78
3.12.2. Компонента StringGrid 85

3.13. Мултимедијални апликации (компоненти Shape и Image) 88
3.13.1Компонентата MediaPlayer 94
ПОГЛАВЈЕ 3 (НАКРАТКО).....98
Прашања и задачи:.....101

4. ПИШУВАЊЕ НА ПОРАКИ (ДИЈАЛОГ РАМКИ)……..103
4.1. ShowMessage…….103
4.2. MessageDlg ……...105
4.3. Messagedlgpos ……...109
4.4 DIALOGS компоненти ……109
4.4.1. OpenDialog ……...110
4.4.2. SaveDialog ……111
4.4.3. ColorDialog ……...111
ПОГЛАВЈЕ 4 (НАКРАТКО) …… 112
Прашања и задачи:……...112

5. КРЕИРАЊЕ НА МЕНИЈА ЗА ФОРМА ……113
5.1. MainMenu ……..113
5.2.PopupMenu ……119
ПОГЛАВЈЕ 5 (НАКРАТКО) ……..122
Вежба ……..122

6. ПРОЦЕДУРИ (PROCEDURE) И ФУНКЦИИ (FUNCTION) ВО DELPHI……...125
6.1. Потпрограми ……125
6.2. Дефинирање на потпрограмата …… 125
6.3. Повикување на потпрограмите ……...127
6.4. ScrollBar – Контрола . Задача со функција ……127
ПОГЛАВЈЕ 6 (НАКРАТКО) ……..132
Прашања и задачи:……...133

7. БАЗИ НА ПОДАТОЦИ ……135
7.1. Поим за база на податоци ……135
7.2. Примарен клуч, надворешен клуч и интегритет на базата на податоци ……137
7.3. Типови на бази на податоци ……140
7.4. Организација на врските со базата на податоци 141
7.4.1. Технологија BDE (Borland Database Engine) 141
7.4.2. Создавање на база на податоци со користење на програмата DatabaseDesktop 143
7.5. Проектирање на апликации за работа со бази на податоци со користење на волшебник ……...147
7.6. Компоненти за врска со базата на податоци ……152
7.6.1 Компоненти за врска со табелата на базата на податоци …….. 152
7.6.2. Компоненти за прикажување и менување на податоци во базата ……..155
7.6.3. Филтрирање на податоци ……161
7.6.4. Обработка на табела од програма ……..163
ПОГЛАВЈЕ 7 (НАКРАТКО) 175
Прашања и задачи:.....176
Одговори 178
Литература 186

ПРЕДГОВОР

Реализацијата на овој учебник е резултат на основната мотивација дека тоа е прв учебник за соодветниот предмет во средните училишта кој покрива материја за која постои поширок интерес, а за која има многу оскудна литература на македонски јазик.

Примарната цел на учебникот е да одговара на потребите за спроведување на наставната програма од предметот програмирање за четврта година. Тоа условува изложената материја да претставува надградба на знаењата стекнати во предметот програмирање во втора и трета година.

Во првото поглавје се разработени основите на објектно ориентирано програмирање. Тука се анлизира Windows и DOS работната околина, паралелното извршување на повеќе програми (multitasking) и можноста за повеќекорисничка работа во мрежа.

Второто поглавје е посветено на анализата на Delphi екранот, односно, неговите четири основни компоненти, работа со менито на програмата, формата како основна компонента на Delphi апликацијата и начините на поставување на компоненти во формата.

Третото поглавје се однесува на пишувањето на програмски код и е најобемно. Се опишуваат настаните и начините на придружување код на настаните. Исто така, ќе се изучуваат елементите на јазикот, програмските структури, програмските модули, опсегот на идентификаторот и сл. Во него се опфатени најголем дел од компонентите, како што се : компоненти за избор, контејнерски компоненти, компоненти за работа со низи, компоненти за работа со мултимедијални апликации и др.

Во четвртото поглавје кое се состои во пишување на пораки (дијалог рамки), даден е поголем број на прозорци и начините на нивното користење. Нив до сега учениците ги сретнале само во работа со Windows и со неговите апликации.

Петтото поглавје ги обработува начините на креирање на менија за формата. Посебно внимание е обрнато на дефинирањето на главното мени, како и на помошното мени.

Процедурите и функциите во Delphi се разработени во шестото поглавје.

Седмото поглавје ги разработува базите на податоци. Тука спаѓаат поделбите на базите и релационите бази на податоци, начинот на поврзување со базите на податоци во Delphi, можностите за користење на волшебници, како и можностите за програмски пристап до податоците во базата на податоци.

Големиот број слики на работните екрани треба да им помогнат на учениците полесно да ги сфатат постапките за извршување на поодделните операции во Delphi. Може да дојде до помали разлики во изгледот на екранот во зависност од верзијата на Delphi која ќе се користи.

Однапред им благодарам на читателите и ги поканувам да ги изнесат своите коментари и забелешки во врска со учебников. Тоа ќе ми помогне да го подигнам квалитетот на изложената материја.

ПОГЛАВЈЕ 1
1.ОСНОВИ НА ОБЈЕКТНО ОРИЕНТИРАНО ПРОГРАМИРАЊЕ

Во ова поглавје ќе научите:
-да правите разлика меѓу DOS и WINDOWS оперативниот систем;
-да разликувате линиска програма од објектно ориентирана програма;
-да толкувате мултитаскинг;
-да објаснувате објектно ориентирано програмирање.

1.1Основни карактеристики на програми засновани на прозорци

Основните принципи на програмирањето се поставени уште во времето на настанувањето на првите компјутери (педесеттите години на минатиот век) . Во согласност со развојот на компјутерската технологија , се менуваа и некои принципи на решавање на задачи со помош на компјутер. Значајни промени во оваа област настанаа со појавата на новиот оперативен систем Windows кој со тек на време во потполност го замени оперативниот систем DOS.

DOS, како и другите оперативни системи, управува со текот на информациите меѓу различните делови на компјутерскиот систем. Со DOS се работи со пишување или со избирање на наредби кои го насочуваат системот да ги изврши поставените задачи.

Оперативниот систем се разликува од другите компјутерски програми по тоа што е неопходен за работа на компјутерот. Компјутер без оперативен систем не може да работи. Оперативниот систем ја усогласува работата на сите делови од кои се состои компјутерскиот систем (екран, глувче, тастатура, хард диск, програмите кои се користат и др.).

Оперативниот систем Windows донесува значајни новини како што се: графичко работно опкружување (кориснички интерфејс), паралелно извршување на повеќе програми (multitasking) и можност за повеќекорисничка работа во мрежа.

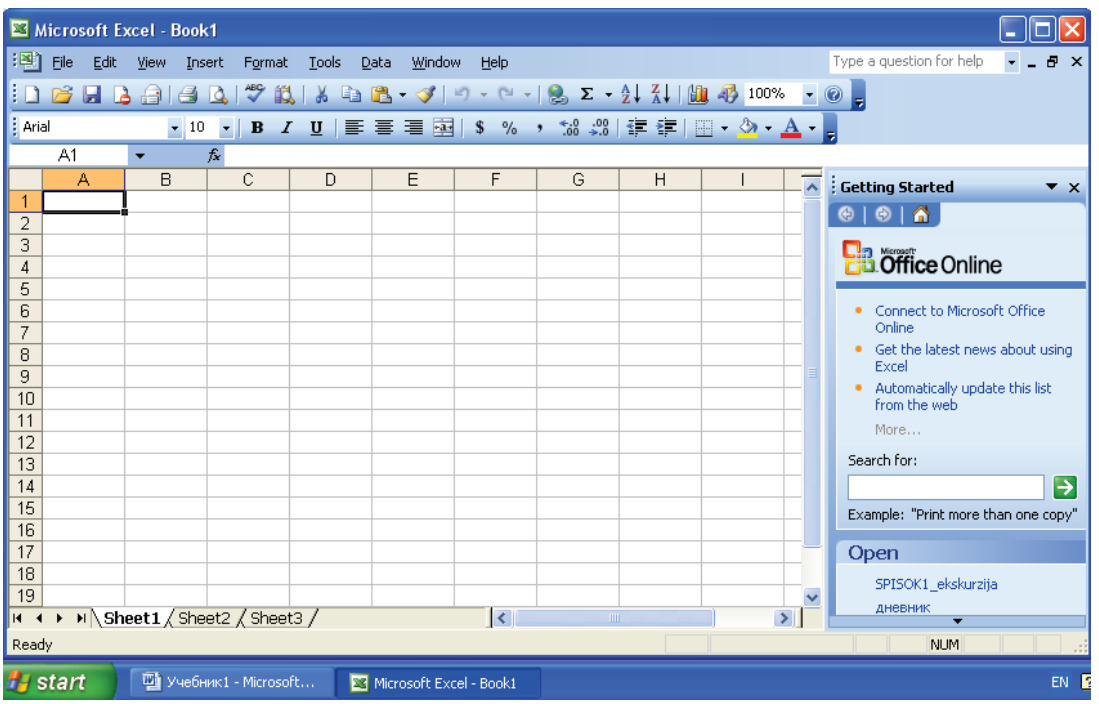
Елементите и процесите на оперативниот систем Windows се посложени и пообемни во однос на DOS и овозможуваат поефикасна контрола на новите можности на оперативниот систем. За да се искористат во поголема мерка новите можности на оперативниот систем, програмерите се принудени да менуваат некои принципи во постапката на решавање на задачата со помош на компјутер.

До појавата на Windows, комуникацијата на корисникот и програмата била многу сложена. Секој производител на програма имал своја идеја за тоа каков интерфејс ќе му понуди на корисникот. Корисникот бил тој кој морал да се прилагодува на визијата на производителот на програмата.

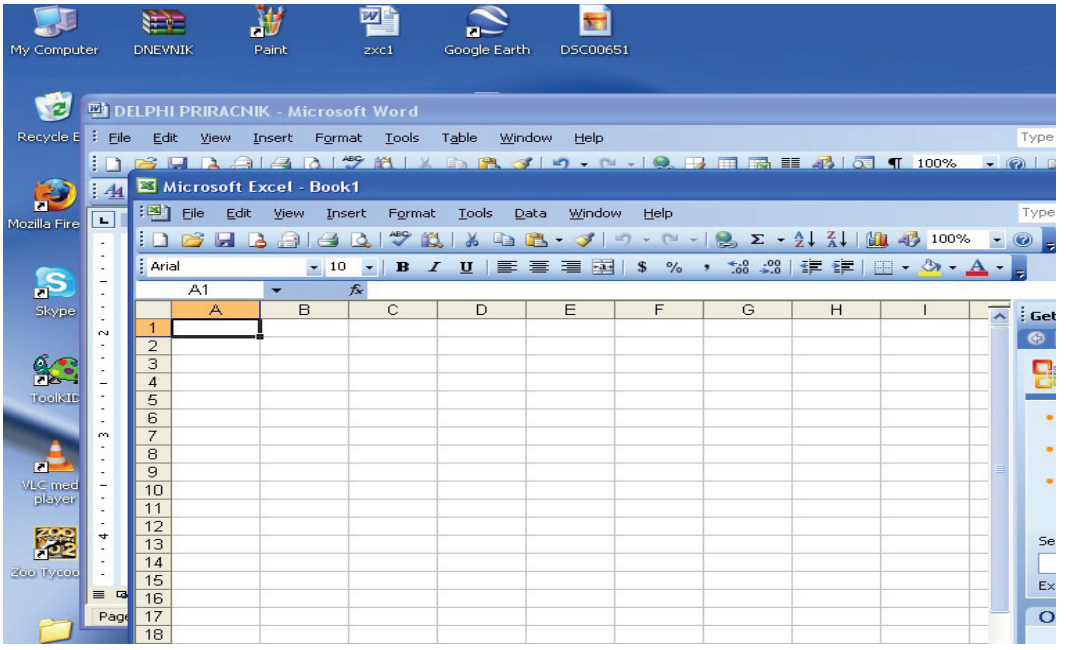
Решение на овој проблем е најдено во модерните оперативни системи, каков што е и Windows. Корисничкиот интерфејс е унифициран на тој начин што во оперативниот систем се вградени елементи кои овозможуваат поедноставна комуникација со корисникот. Сите елементи на интерфејсот се од графичка природа и по изгледот упатуваат на активноста која со нив се спроведува. При пишување на програми, потребно е да се искористат веќе постоечките решенија од оперативниот систем. Што се однесува до корисникот, доволно е да се запознае со начинот на работа и користењето на овие елементи на еден програма. Работата со секоја нова програма ќе има иста филозофија и корисникот побрзо ќе ја совлада неговата употреба.

Со овој пристап на решавање на проблемите креаторот на програмата е ослободен од проблемот на комуникација со корисникот, па поголем дел од работното време може да го посвети на решавање на поставената задача. За да можат да се решаваат задачите на овој начин, неопходно е добро познавање на оперативниот систем и неговите механизми за контрола на поодделни операции кои ќе бидат вклучени во програмата.

Основна карактеристика на програмите напишани за Windows опкружувањето е дека секоја програма се извршува во посебен прозорец. Во еден момент може да бидат вклучени повеќе програми. Секоја од активираните програми има свој прозорец во кој се извршува, а секоја од програмите може да ги користи ресурсите на оперативниот систем. На слика1. е прикажан изглед на графичката корисничка површина кога се активни две програми. На лентата за задачи се забележува дека се испишани две имиња на активираните програми.
Еден од нив е документ во Word, а другиот, кој во моментот е активен, е документ на Excel . Програмата се активира со клик, со глувчето на соодветниот натпис во task лентата – лента за задачи. Распоредот на прозорците на екранот може да биде и поинаков (каскада), како на слика 2.



Слика1. Изглед на кориснички интерфејс кога работат две програми



Слика2. Каскаден распоред на прозорци

1.2.Елементи на графичка корисничка површина

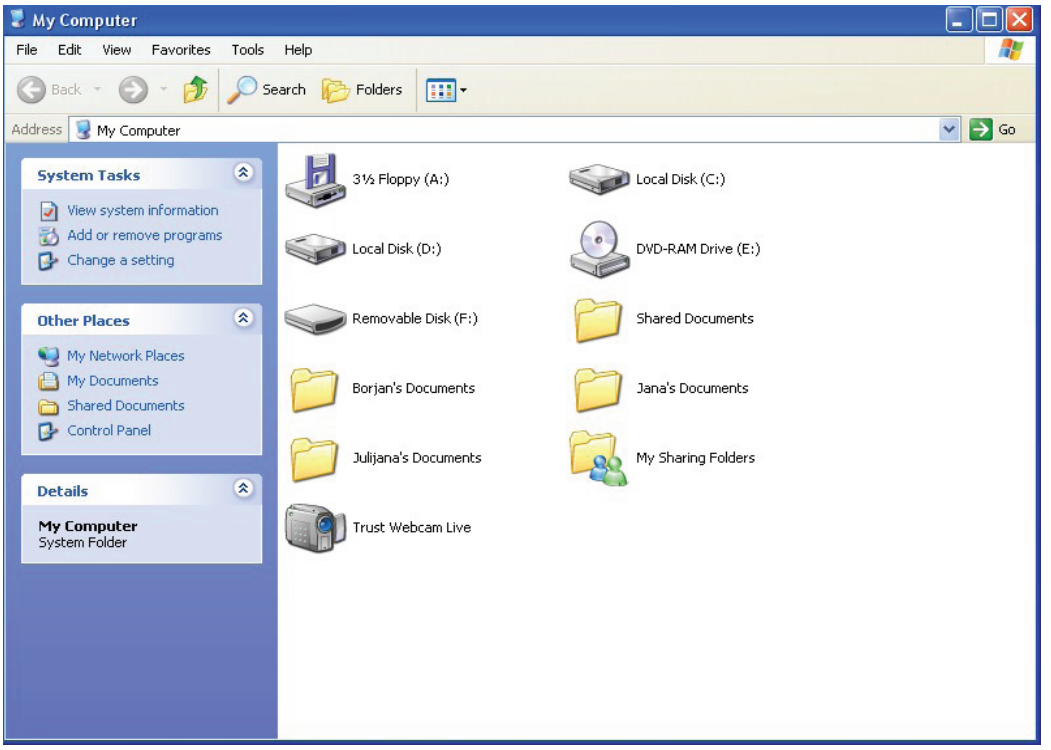
Графичката корисничка површина има сличен изглед за различни кориснички програми напишани за работа со оперативниот систем Windows. Причините за тоа се: полесно користење на различни програми и полесно пишување на програмите благодарение на употребата на готови елементи на оперативниот систем. Веќе спомнавме дека основен елемент на графичката корисничка површина е прозорецот. Во зависност од изгледот и намената, сите прозорци кои се појавуваат при работата со корисничките програми може да се класифицираат во неколку групи:

- воведен прозорец на оперативниот систем (Desktop) или работна површина,
- прозорец на фолдер,
- прозорец - документ,
- дијалог - прозорец.

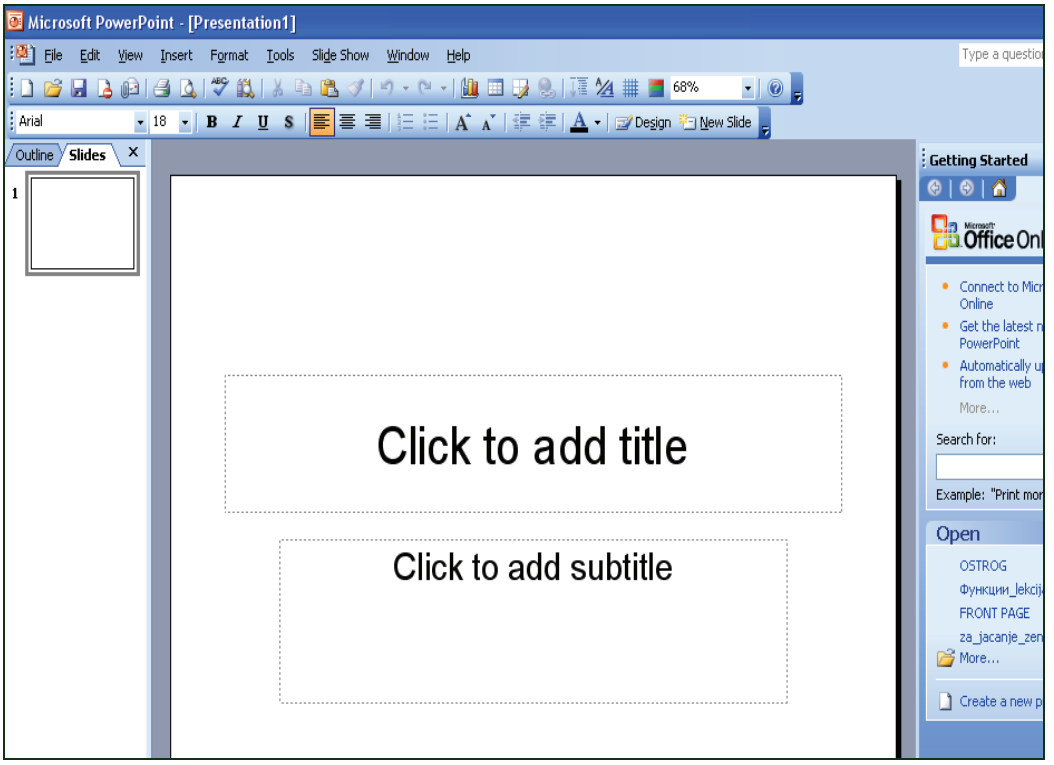
Воведниот прозорец се појавува при секое стартување на оперативниот систем и најчесто се состои од одреден број сликички кои се нарекуваат икони. Изгледот на еден воведен прозорец е даден на слика 3.

Слика 3. Воведен прозорец на оперативниот систем (Desktop)

Прозорецот на фолдери ја претставува хиерархиската организација на податоците во оперативниот систем Windows. Во прозорецот на фолдери може да се најдат други фолдери (потфолдери) доколку постојат и/или датотеки како организирани групи на информации. Изгледот на еден фолдер е даден на слика4.



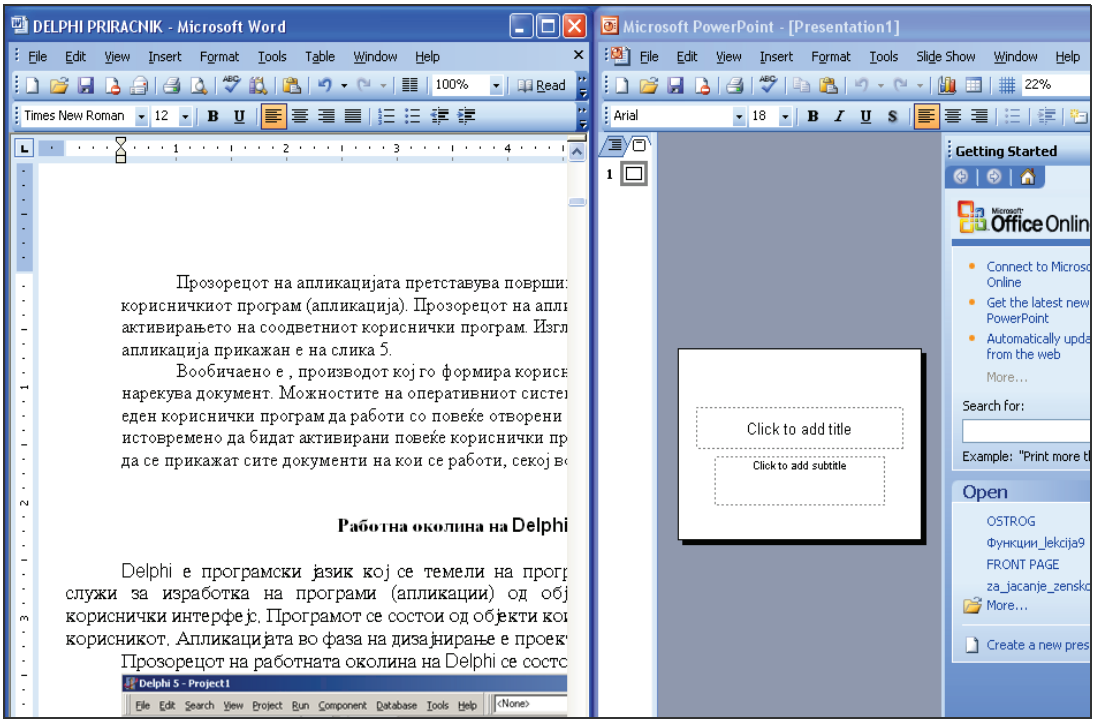
Слика 4. Групен прозорец My computer



Слика 5. Прозорец на апликација PowerPoint

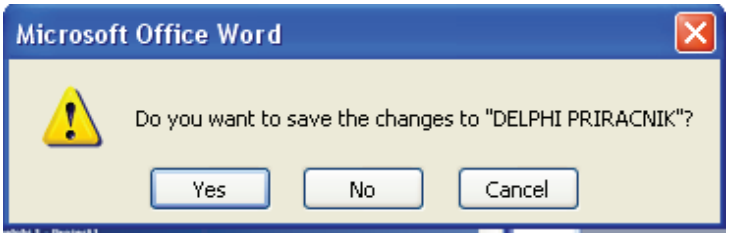
Прозорецот на апликацијата претставува површина на која се извршува корисничката програма (апликација). Прозорецот на апликацијата се појавува после активирањето на соодветната корисничка програма. Изгледот на еден прозорец на апликација е прикажан на слика 5.

Вообичаено е, производот кој го формира корисничката програма да се нарекува документ. Можностите на оперативниот систем Windows дозволуваат една корисничка програма да работи со повеќе отворени документи или истовремено да бидат активирани повеќе кориснички програми. Во тој случај може да се прикажат сите документи на кои се работи, секој во свој прозорец, како на слика 6.



Слика 6.Прозорец на документ

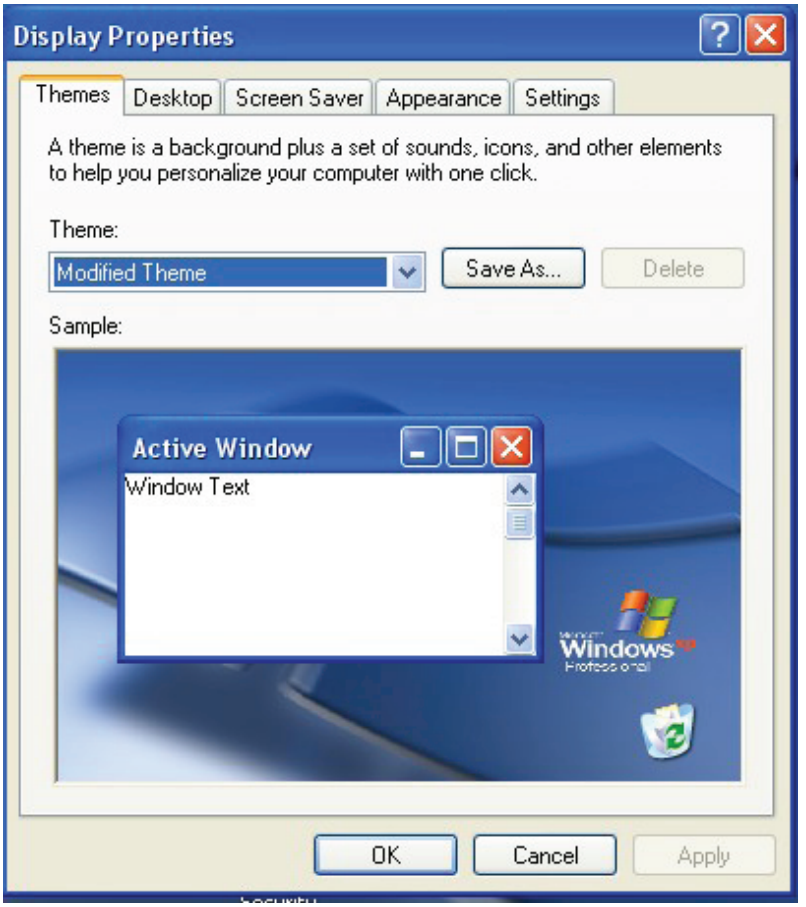
Прозорците за дијалог се различни во зависност од намената. Најчесто се појавуваат во облик на прозорци за предупредување или во облик на прозорци за давање дополнителни информации на оперативниот систем за понатамошната работа. На сликата 7. е прикажан прозорец на предупредување.



Слика 7. Прозорец на предупредување

Овој прозорец се појавува ако корисникот се обиде да ја напушти програмата, а работата на документот не ја снимил. Ова е последно предупредување за чување на претходната работа. На корисникот му се на располагање три можности од кои секоја се бира со притискање на едно од прикажаните копчиња.

Дијалог - прозорецот може да има и посложен изглед, а со тоа и поголема функционалност, како прозорецот на слика 8.



Слика 8.Дијалог - прозорец Properties

Во овој прозорец забележуваме постоење на пет картички (страни), чии јазичиња се дадени на врвот на прозорецот: Themes, Desktop, Screen Saver, Appearance и Settings. На сликата е прикажана функцијата на картичката Themes. Останатите картички се бираат со кликање со глувчето на соодветното јазиче. Во овој дијалог - прозорец се поставуваат многу параметри кои влијаат на работата и однесувањето на оперативниот систем.

Важно е да се напомене дека на некои прозорци не може да им се менува големината која ја дефинирал оперативниот систем, додека на други може. Големината на прозорците е однапред дефинирана.

1.3.Програми управувани со настани

Програмите управувани со настани им овозможуваат на корисниците да ги извршуваат задачите по кој било редослед. Концепцијата на овие програми обезбедува обработка на информации на начин кој е најприфатлив за корисникот. Затоа, програмите управувани од настани суштински се разликуваат од традиционалните процедурални програми. Кај овие програми една акција на корисникот може да придвижи еден настан или низа на настани.

Ние живееме во свет кој е воден од настани. Да земеме, на пример, музички столб. Сите понови уреди од овој тип имаат далечински управувач, со чија помош може да се одбере уредот кој ќе произведува музика (радио, касетофон или CD) , може да се регулира јачината на звукот, квалитетот на звукот, да се исклучи тонот на одредено време, потполно да се исклучи

музичкиот столб итн. Сите овие активности, а и многу други се извршуваат со притисок на соодветното копче на далечинскиот управувач. Овој пример за контрола на музички столб претставува програмирање водено од настани. Корисникот со притисок на копче активира настан, музичкиот столб презема соодветна акција, што значи дека корисникот го контролира одвивањето на настаните.

Во своето опкружување може да пронајдете голем број уреди кои се програмирани да реагираат на предвидени настани.

Во светот на програмирањето и компјутерите настаните можат да бидат многу различни: притисок на тастер на тастатура, движење на глувчето, клик со глувчето, избор од мени, движење на прозорец од страна на корисникот или достигнат временски момент програмиран на интерниот часовник на компјутерот.

Оперативниот систем Windows е чувствителен на настани (ги препознава) и со свои вградени механизми реагира на нив. Кога Windows "почувствува" одреден настан, тој се обидува да "и укаже " на програмата за кој настан станува збор, испраќајќи соодветна порака. Активната програма треба да ја интерпретира таа порака, да го препознае настанот на кој се однесува пораката и потоа да реагира во согласност со тоа.

Програмскиот пакет Delphi овозможува пишување на програми кои се извршуваат само доколку се случи одреден настан. Само ако корисникот притисне одредено командно копче или селектира некоја операција, се извршува одредена програма во Delphi.

Настаните во Delphi програмата можат да бидат активирани со акција на корисникот, интерни функции, како што се тајмери (интерни часовници) или со надворешни фактори (прием на електронска порака). Постојат многу настани кои можат да се случат во корисничката програма, која треба да е способна да ги селектира настаните. Акциите се одвиваат само при одредени настани.

Настанот може да се случи во кој било момент во текот на работата на програмата. Windows и овозможува на корисничката програма да го препознае настанот. Кога Windows ќе го препознае настанот, програмата го извршува програмскиот код кој е напишан за тој настан. Проблемот со настаните е решен со добрата соработка меѓу Windows и програмскоит пакет Delphi за кои може да се каже дека работат заедно.

Во Delphi постојат два основни типа на настани: настан кој го предизвикал корисникот и настан кој го предизвикал оперативниот систем.

Настаните кои ги предизвикал корисникот, му овозможуваат контролирање на правецот во кој се извршува програмата. Тоа значи дека корисникот може да преземе специфични акции кога ќе посака што му дава потполна слобода во контролата на програмата.

Со појавата на објектно ориентирано програмирање (ООП) се олеснува правењето на Windows програмите. Ваквиот начин на програмирање овозможува креирање на компоненти за повеќекратно користење во разни кориснички програми. Основни поими во објектно ориентирано програмирање се: компоненти, објекти, методи, својства и настани.

Компоненти се елементи вградени во Delphi за повеќекратна употреба и се користат при формирање на објектите.

Објект е основен елемент на Delphi програмата кој има соодветни својства кои ги одредуваат неговите карактеристики, методи кои го дефинираат

неговото однесување и можност за препознавање на настаните на кои може да реагира.

Метод е програмски код придружен на објектот кој одредува како објектот ќе ги обработува информациите и како ќе реагира на настаните.

Својства се карактеристики на објектот како што се, на пример: големина, положба, боја или фонт на текстот. Својствата го одредуваат појавувањето, а понекогаш и однесувањето на објектот. Својствата можат да се користат да му набаваат на објектот податоци и да ја вратат информацијата од објектот.

Објектно ориентирано програмирање е засновано на три основни принципи: енкапсулација, наследување и полиморфизам.

Енкапсулација значи дека сите информации за објектот (својства) и процесите (методи) над него се содржат во дефиницијата на објектот. На пример, да набљудуваме компјутерски систем како објект. Него го опишуваме со помош на неговите својства: тип на процесорот, големина на дискот, меморијата, брзината на модемот, големината на мониторот ... Секоја наведена карактеристика претставува својство на компјутерскиот систем. Методите претставуваат начини на кои ќе се однесува компјутерскиот систем во различни ситуации, на пример, активирање на некоја програма. Во тој момент оперативниот систем извршува низа операции, а корисникот притоа не треба на компјутерскиот систем да му соопштува што да направи.

Наследување значи дека еден објект може да се опише врз основа на друг објект. Ако е потребно да се дефинира преносен компјутер, не треба да се зборува за екранот и останатите својства на компјутерскиот систем, туку се претпоставува дека тие се исти како кај компјутерскиот систем. Треба да се додадат само оние својства по кои преносниот компјутер се разликува од класичниот.

Полиморфизам значи дека многу објекти можат да имаат ист метод, а соодветната акција се презема од објектот кој го повикува тој метод. Пример, програмите за обработка на текст можат текстот да го прикажат на екран или печатач. Секој од овие два објекта, со помош на метод за прикажување или печатење , му соопштува на објектот да го прикаже текстот на одредено место. Методот кажува што треба да се направи во зависност од објектот кој го повикува методот.

За оние кои сакаат да знаат повеќе

ОБЈЕКТИ

Објектите може да ги замислиме како предмети во природата. Кај нив забележуваме:

-Својства - како ширина , висина, боја.

-Настани- како на пример, седнување на возачот во автомобил.

-Методи - како придвижување на моторот на автомобилот.

-Подобјекти-како на пример, мотор кој е составен објект на автомобилот.

Одредени својства на објектите ќе се наследуваат од подобјектите, како на пример, возните карактеристики на автомобилот во зависност од типот на моторот.

Значи, објектите имаат свои методи, својства, настани. Имаат и подобјекти од кои ги наследуваат својствата, методите и настаните. Можат да бидат видливи или не.

СВОЈСТВА

Својствата се најчесто варијабли од некој тип, а можат да бидат функции или процедури, односно објекти.

Со промена на својствата објектите поинаку се однесуваат.

Својствата можат да се менуваат во фаза на изработка на интерфејсот (во Object Inspector) или во фаза на извршување на апликацијата (со наредба во Unit прозорецот).

НАСТАНИ

Настани се процедури со точно одредено име , а се активираат сами тогаш кога настапила состојбата која го дефинира настанот. Пример, ако кликнеме на прозорецот, настапува настан Click, а ако изведеме двоен клик, настапува настан DblClick.

Истовремено можат да се јават повеќе настани.

Настаните се програмираат така што при активирањето, апликацијата го прикажува својот интерфејс – прозорците и објектите во нив. Тие сите чекаат корисникот нешто да направи, како би предизвикал некоја нивна акција како одговор на постапката на корисникот. Тоа е вообичаено кај сите Windows апликации, па програмирањето на нив се сведува на програмирање на настани (кои треба да се извршат ако корисникот нешто направи).

МЕТОДИ

Методи или постапки се најчесто процедури кои прават нешто на објектот. Можат да се повикаат со наредби во Unit прозорецот или со промена на својствата. Тие може да иницираат некој настан.

1.4. Проблеми кои се решаваат со Delphi

Delphi е програмски пакет наменет за креирање Windows апликации за персонални компјутери. Краен производ на Delphi е извршна програма која ги решава задачите. Програмскиот јазик кој го користи Delphi е модернизирана верзија на Pascal познат како Object Pascal.

Програмите пишувани во Delphi може да бидат со различна намена. Многу добри карактеристики покажува при решавање на проблеми кои користат бази на податоци.

Бидејќи крајниот резултат на работата во Delphi е извршна (EXE) програма , успешно може да се користи за правење независни комерцијални програми. Пишување на големи програми за обработка на текст и слика се прави во C++ или асемблер, но овие програми може да се прават и во Delphi.

Денес постојат и други програми со сличен начин на работа како Delphi, а тоа се Visual Basic i Visual C++. Секоја од овие програми има своја специфичност.

Првата верзија на Delphi настанала во 1995г, како директна последица на Windows 95. Денес најчесто се користи Delphi 5 i Delphi 7.

ПОГЛАВЈЕ 1
(НАКРАТКО)

DOS, како и другите оперативни системи, управува со текот на информациите меѓу различните делови на компјутерскиот систем. Со DOS се работи со пишување или со избирање на наредби кои го насочуваат системот да ги изврши поставените задачи.

Оперативниот систем **Windows** донесува значајни новини како што се: графичко работно опкружување (кориснички интерфејс), паралелно извршување на повеќе програми (**multitasking**) и можност за **повеќекорисничка** работа во мрежа.

Со појавата на објектно ориентирано програмирање (ООП) се олеснува правењето на Windows програмите.

Основни поими во **објектно ориентирано програмирање** се: компоненти, објекти, методи, својства и настани.

Објект е основен елемент на Delphi програмата кој има соодветни **својства** кои ги одредуваат неговите карактеристики, **методи** кои го дефинираат неговото однесување и можност за препознавање на **настаните** на кои може да реагира.

Метод е програмски код придружен на објектот кој одредува како објектот ќе ги обработува информациите и како ќе реагира на настаните.

Својства се карактеристики на објектот како што се, на пример: големина, положба, боја или фонт на текстот.

Програмскиот пакет Delphi овозможува пишување на програми кои се извршуваат, само доколку се случи одреден **настан**.

Во Delphi постојат **два основни типа на настани**: настан кој го предизвикал корисникот и настан кој го предизвикал оперативниот систем.

Delphi е програмски пакет наменет за креирање Windows апликации за персонални компјутери.

Објектно ориентирано програмирање е засновано на три основни принципи**: енкапсулација, наследување и полиморфизам.**

Прашања и задачи:

1. Командна линија е одлика на _____ оперативен систем, а прозорец и работна маса (desktop) се одлика на _____ оперативниот систем.
2. Windows работната околина се одликува со _____, а DOS со _____.
3. Ако системот го користат повеќе процеси станува збор за _____.
4. Пишувањето Delphi програма вклучува 2 програмски чекори: _____ и _____

Одговор**:** Визуелен и Код програмски чекор.

5. Поимот multiuser кај оперативните системи значи _____

Одговор: дека системот едновремено можат да го користат повеќе корисници.

6. Кои се трите основни принципи на објектно ориентирано програмирање _____
7. Карактеристиките на објектот како што се, на пример: големина, положба, боја или фонт на текстот се нарекуваат _____
8. Програмскиот код придружен на објектот кој одредува како објектот ќе ги обработува информациите и како ќе реагира на настаните се нарекува ____

ПОГЛАВЈЕ 2

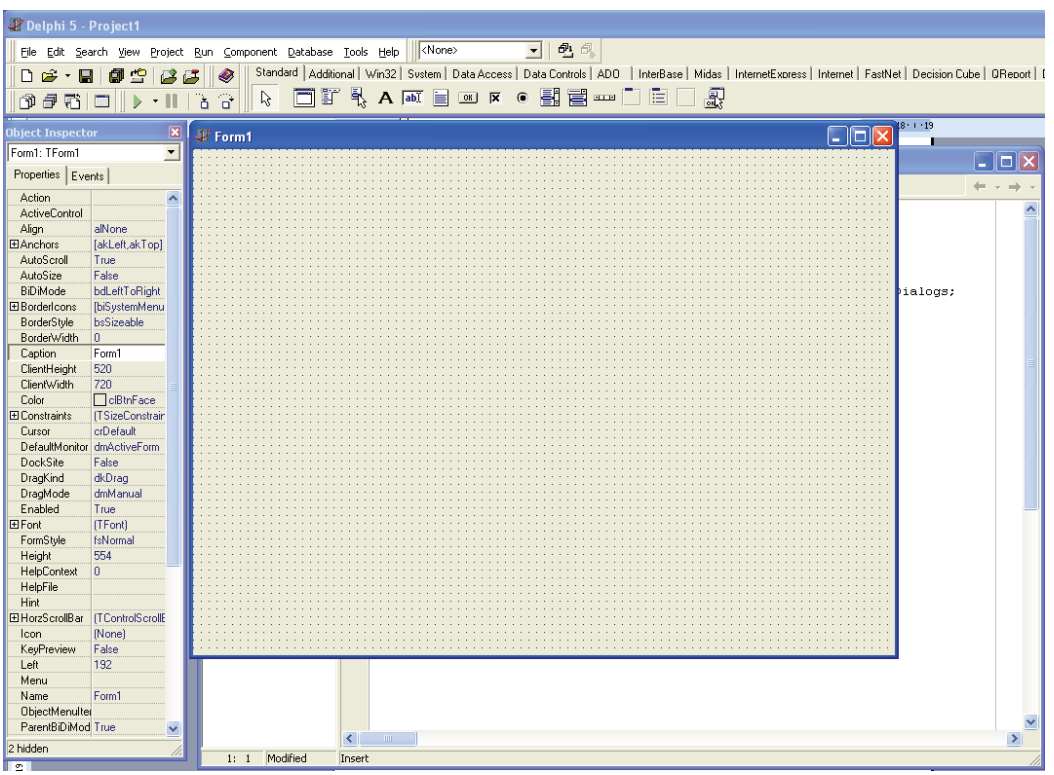
2.АНАЛИЗА НА DELPHI ЕКРАНОТ

Во ова поглавје ќе научите:

-да ги опишувате четирите основни делови на Delphi прозорецот;
-да ги користите поважните опции на главното мени **FILE, EDIT, VIEW, PROJECT, RUN, HELP**.
-да разликувате компајлер и интерпретер;
-да ја истакнувате палетата на компоненти и почесто користените страници;
-да ги познавате компонентите од страницата **STANDARD**;
-да ги познавате компонентите од страницата **ADDITIONAL**;
-да ги познавате компонентите од страницата **SYSTEM;**
-да го објаснувате **DELPHI** проектот и датотеките од кои се состои (.dpr, .dfr, .pas);
-да ја истакнувате ФОРМАТА како прва и најважна компонента;
-да ги толкувате начините за поставување на компонента на формата;
-да променувате својства на формата и компонентите.

2.1.Вовед во работната околина на Delphi

Delphi е програмски пакет кој се темели на програмскиот јазик Pascal, а служи за изработка на програми (апликации) од објектен тип со графички кориснички интерфејс. Програмата се состои од објекти кои реагираат на акциите на корисникот. Апликацијата во фаза на дизајнирање е проект.



Слика 2.1.Почетен изглед на Delphi прозорецот

Прозорецот на работната околина на Delphi се состои од четири дела и тоа:

Главен прозорец (main)

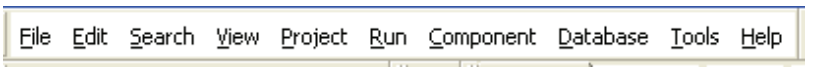
Тој се состои од:

а) насловна - лента
б) мени - лента
в) лента со алатки Toolbars
г) лента со компоненти Component palette

Насловната лента е стандардна Windows насловна линија која содржи

иконче на програмата, име на програмата Delphi, име на проектот – документот и три копчиња за управување со големината на прозорецот на апликацијата.

Мени - линијата е класична Windows мени - линија која содржи 11 опции,



кои се активираат со кликање на глувчето на саканата опција. Секоја опција има одреден број наредби.

Во лентата со алатки се активности како отворање на проектот, снимање на проектот, поглед на кодот, дебагирање на програмата (пронаоѓање и отстранување на грешки), стартување на програмата итн. Овие наредби се задаваат со директно кликање со глувчето на соодветната сликичка. Од лево на десно овие икони го имаат следното значење:

Стандардна лента:
New – нов проект,
Open – отвора постоечки проект, форма,...
Save – го меморира проектот,
Save all – ги снима сите датотеки на проектот,
Open project – отвора постоечки проект,
Add file to project – додава датотека во проектот,
Remove file from project – одзема датотека од проектот,
Help лента – пристап до помош,
Help contents – прозорец за помош,

View лента – лента која овозможува прикажување на програмата или на формата (оваа лента се наоѓа во вториот ред лево),
View Unit –го прикажува програмскиот код на формата која се дизајнира,
View Form – ја прикажува формата која се дизајнира,
Togle Form/Unit – заменува форма и код,
New Form – нова форма во проектот,

Debug лента-лента за поправање на грешките во програмата(debugiranje)
Run – го активира извршувањето на програмата (СО ПРЕТХОДНО ПРЕВЕДУВАЊЕ),
Pause – го прекинува извршувањето на програмата,
Trace into – го започнува извршувањето на програмата наредба по наредба (trace – облик на дебагирање кај кое се следи редоследот на наредбите) или преминува на извршување на следната наредба од

програмата ако trace веќе почнало,
Step over – ја извршува програмата наредба по наредба, земајќи ја потпрограмата како една наредба, за разлика од Trace кој влегува во потпрограмите.

Палетата со компоненти се состои од повеќе страници, при што секоја страница содржи свои компоненти кои можат да се вметнат во формата. По стартувањето на Delphi активна е страната Standard, а нејзините компоненти се прикажани во редот подолу. Активноста на страницата се менува со кликање со глувчето на насловот на страната, со што се менуваат и компонентите кои се наоѓаат во таа страница.

Во левиот дел на прозорецот се наоѓаат:

а) прозорец за хиерархиски приказ на компонентите на апликацијата
б) прозорец за одредување на својствата и настаните на објектите во апликацијата - **Object Inspector**.

Тој има два дела :

- Properties (својства или карактеристики)
- Events (настани).

Во хиерархискиот прозорец се прикажува хиерархиско стебло на објектите поставени во апликацијата.

На врв на прозорецот на Object Inspector -от се наоѓаат две јазичиња (Properties и Events). Тоа значи дека овој прозорец има две страни. Првата страна ги прикажува својствата, а втората настаните за дадениот објект. Прозорецот Properties е поделен на две колони. Во левата се имињата на својствата, а во десната се нивните вредности. Со кликање на јазичето Events се добива список на сите можни настани за разгледуваниот објект. Во полињата за настани се запишуваат имињата на процедурите кои се извршуваат кога ќе настапи одреден настан.

Пример за својства: со промена на својството Color на некоја компонента се менува бојата на таа компонента.

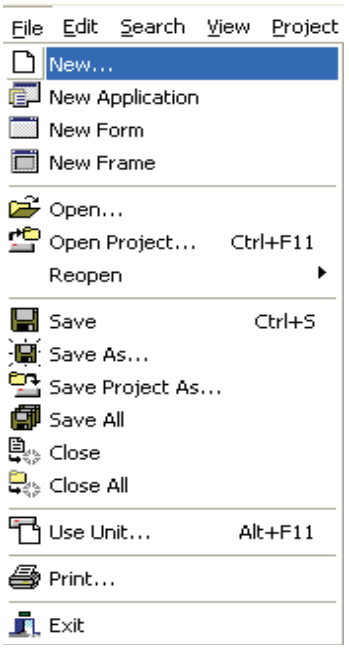
Пример за настан: ако се кликне на одредена компонента се генерира настан кој кажува дека на компонентата е кликнато. Може да се напише програмски код кој одговара на овој настан и кој ќе извршува одредени дејствија кога ќе се појави настанот.

Третиот дел на Delphi околината е прозорец на образец или форма за дизајнирање на апликацијата, каде што ги сместуваме компонентите.

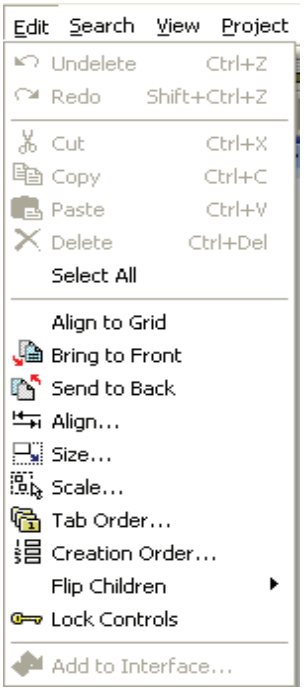
Четвртиот дел на Delphi околината е прозорец за пишување на програмски код кој се наоѓа зад дизајнерот на форми.

Од третиот во четвртиот прозорец преминуваме со F12.
Во текот на креирањето на апликацијата Object inspector, Code editor, Form designer и Component palette работат интерактивно.

Некои опции од мени - лентата



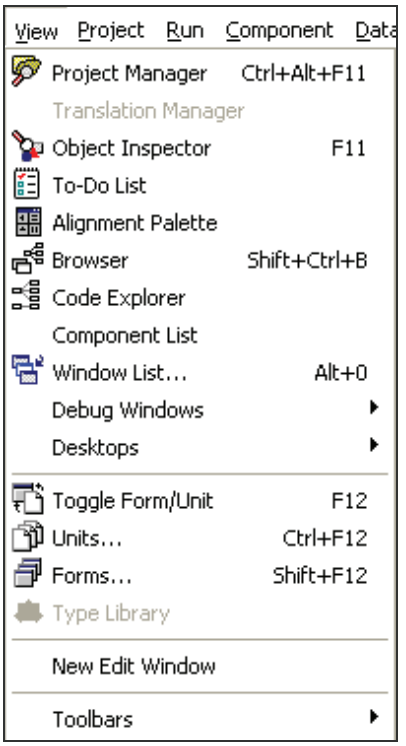
Опцијата **File** од менито ги содржи:
New – нов проект , форма, Unit …;
New Application – нова апликација
New Form – додава нова форма во апликацијата
Open – отвора постоечки проект, форма,...;
Open Project – отвора проект
Reopen – повторно отвора проект, форма ...
Save – го меморира Unit-от;
Save As – го снима unit-от под друго име;
Save Project As – го снима целиот проект;
Save all – ги снима сите датотеки на проектот
Close – ја затвора датотеката,
Close All – ја затвора апликацијата
Print – печатење на формата или на кодот,
Exit – излез од Delphi,



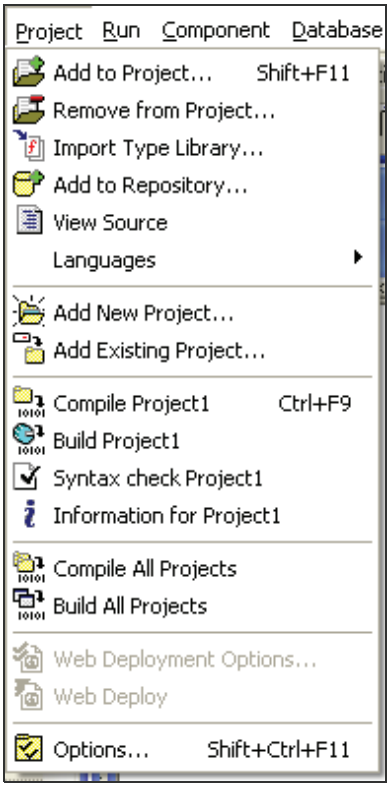
Опцијата **EDIT** ги содржи следните подопции:

Undelete – поништи го бришењето
Undo – поништување на промените направени со последните операции (обично врз текст)
Redo – ги враќа ефектите од Undo,
Cut, Copy, Paste - наредби за копирање и преместување на текст
Select All - означување на целиот текст или сите објекти во формата,
Align to grid – поставување на селектираната компонента на најблиската грид - точка.
Bring to front, Send to back – поставување на компонентата во преден план или во позадина,
Align- поставеност на компонентата
Size - големина, Scale – фактор на зголемување , намалување ,
Tab order – редослед на поставување на фокус на компонентите
Lock Controls – ги заштитува контролите од преместување , зголемување или намалување.

Опцијата **VIEW** ги има следните подопции:



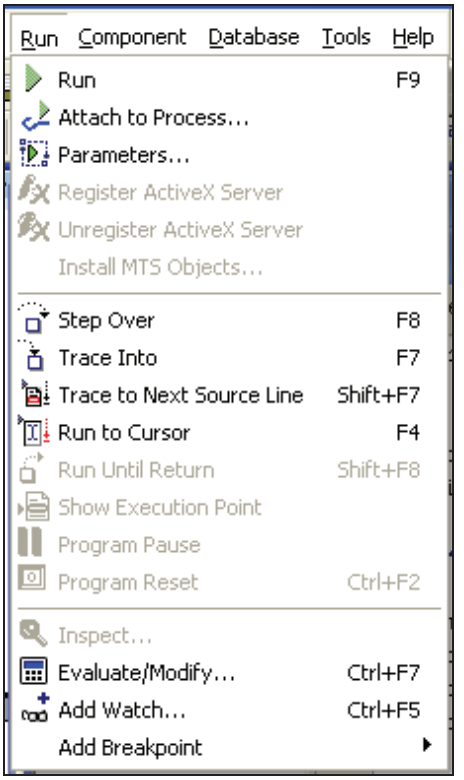
Project Manager – овој прозорец прикажува информации за статусот и фајловите кои ги содржи тековно отворениот проект;
Object Inspector – го прикажува Object inspector ;
Alignment palette – лента со алатки за порамнување;
Units – го прикажува кодот на програмата;
Forms – ги прикажува формите;
Toolbars – вклучува ленти со алатки.



Опцијата **PROJECT** ги има следните подопции:

Add to Project – додава еден постоечки Unit кој го поврзува со тековниот Delphi проект. Кога ќе додадете unit на проектот, Delphi автоматски го додава тој unit во uses клаузулата на project датотеката.
Remove from Project – отстранување од проектот;
Compile Project – ги компајлира сите фајлови од тековниот проект кои биле променети од последната промена, креирајќи нова извршна датотека-executable file (.EXE),. dynamic link library (.DLL), resource file (.RES), итн. Оваа наредба е слична со Build наредбата, со таа разлика што Build ги реизградува сите фајлови без разлика дали биле сменети или не.

Опцијата **RUN** ги има следните подопции:



Run - го активира извршувањето на програмата (СО ПРЕТХОДНО ПРЕВЕДУВАЊЕ)

Program Pause– го прекинува извршувањето на програмата
Program Reset – програмата започнува со извршување од почеток , откако сме ја пронашле грешката или ако варијаблите добиле несакани вредности.

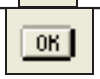
2.2.Интерпретер и компајлер

Интерпретер е преведувач кој секоја линија од изворниот код ја преведува на машински јазик и ја извршува. Ако една програмска линија се повторува повеќе пати, истата ќе биде анализирана онолку пати колку што се појавува и ќе биде преведена во машински јазик исто толку пати со цел да биде извршена. Тоа е причината поради која користењето на интерпретер го успорува извршувањето на програмата. Од друга страна, интерпретерот се одликува со едноставност, затоа што е можно во моментот да се откријат евентуалните грешки (со дебагирање) во текот на извршувањето на програмата.

Компајлерот е програма која секоја наредба од изворниот код ја анализира и преведува на машински код, кој го запишува во излезна извршна датотека. При извршувањето се извршуваат преведените инструкции во машински код, односно извршната датотека. Тоа е причината поради која програмите се извршуваат побрзо во однос на тоа кога извршувањето оди преку интерпретер.

2.3. **Некои компоненти од Палетата на компоненти**

Компонентите кои се наоѓаат на Standard страницата од палетата на компоненти се следните:

	**Frames**	Отвора дијалог - прозорец во кој прикажува листа на рамки вклучени во тековниот проект. Рамката е контејнер за други објекти.
	**MainMenu**	Компонента за главно мени. Секоја форма која треба да има мени, треба да има една ваква компонента.
	**PopupMenu**	Локално мени. Ова мени се креира, така што ќе се појавува кога ќе се кликне на десното копче на глувчето. Тоа се постигнува со поставување AutoPopup на True.
	**Label**	Овозможува појавување на статичен текст на екранот. Содржината на лабелата се впишува во својството Caption на лабелата.
	**Edit**	Компонента за внесување на текст во еден ред. Внесениот текст се придружува на својството Text.Ако во оваа компонента се внесе број, тој мора соодветно да се претвори.
	**Memo**	Компонента слична на Edit, со таа разлика што може да се внесува текст во повеќе редови.
	**Button**	Ова претставува Windows копче. Неговиот наслов се задава со својството Caption. На страната Events се дефинира настан на кој ќе реагира копчето.
	**Check Box**	Компонента од прекинувачки тип. Со притисок на овој објект се вклучува, односно исклучува, логичкото поле Checked.
	**RadioButton**	Ова е компонента слична на CheckBox. Кога повеќе од овие компоненти ќе се најдат во група, само една од нив може да биде вклучена.
	**ListBox**	Компонента која претставува веќе познатата Windows листа. Податоците можат да бидат прикажани во повеќе колони што се дефинира со својството Columns.
	**ComboBox**	Оваа компонента е комбинација од Edit и ListBox компоненти.
	**ScrollBar**	Ова е компонента која овозможува внесување на податоци со помош на лизгач.
	**GroupBox**	Компонента која служи за тоа во неа да се поставуваат други компоненти. Таа овозможува истовремено дејствување врз сите компоненти во рамките на GroupBox.

	**RadioGroup**	Компонента која овозможува радиокопчињата да не се внесуваат како посебни компоненти, туку како елементи на листа.
	**Panel**	Оваа компонента служи за разубавување на формата.
	**ActionList**	Креира множество на акции, со што се централизира **а**пликацијата во согласност со акциите на корисникот.

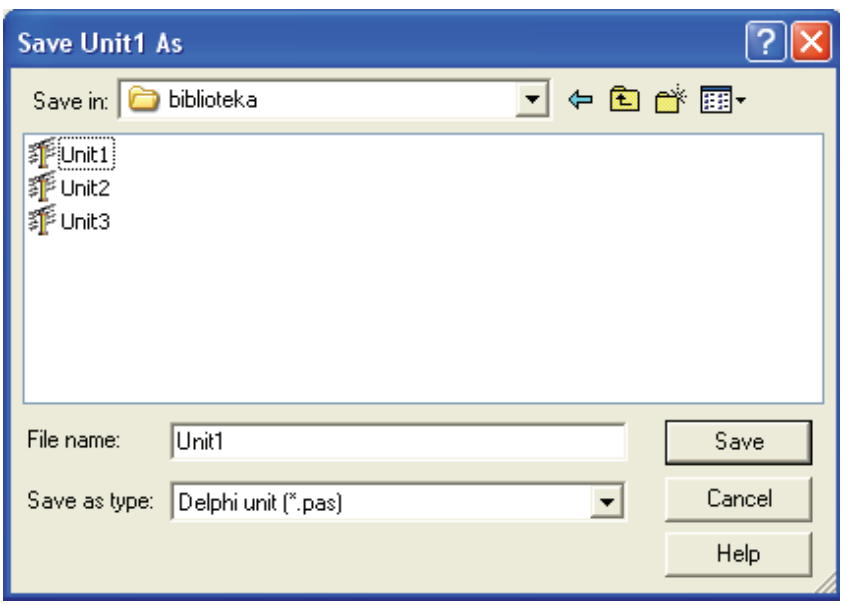
2.4.Празен проект

Празен проект се добива со активирање на Delphi или со користење на опцијата File->New->Application од мени.

2.4.1. Чување и отворање на проектот

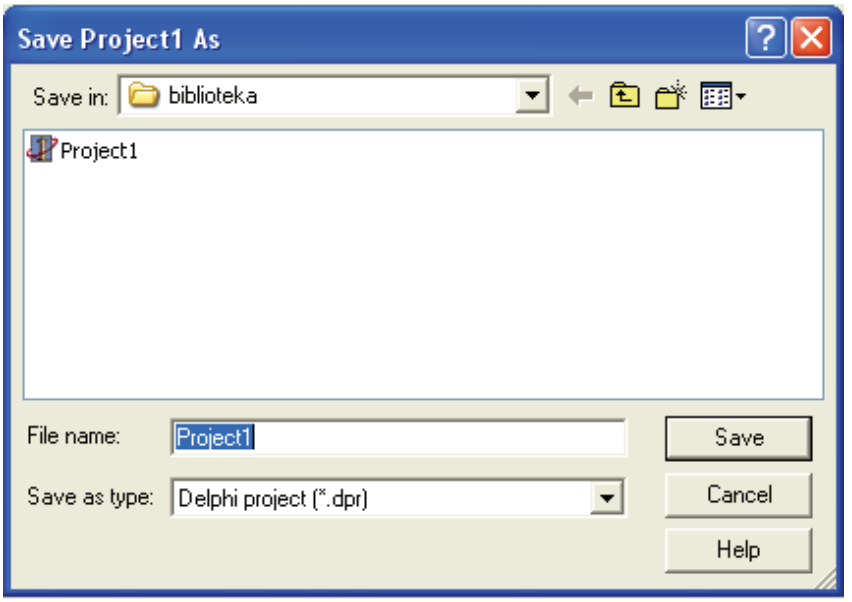
За да го зачуваме проектот, треба да го направиме следното:

- Во мени - линијата одбираме опција File->Save Project As… по што се добива дијалог - прозорец како на сликата 2.2



Слика2.2. Прозорец за чување Unit датотека.

- Потоа во полето за име на датотеката го внесуваме името и кликнуваме Save. Оваа датотека е Pascal датотека.



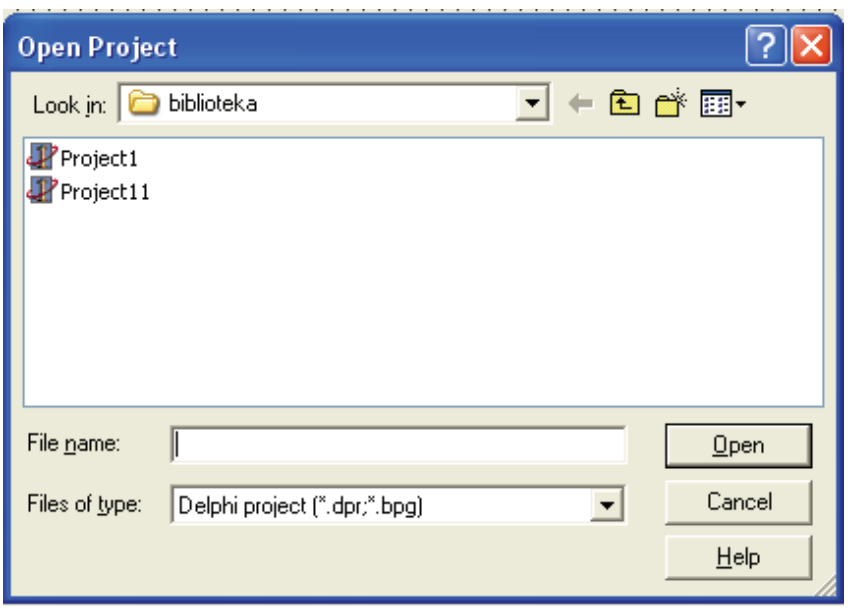
Слика2.3.Прозорец за чување на проект на датотеката

- Во следниот чекор се појавува комуникациски прозорец, како на слика 2.3. во кој треба да се зададе името на проектот.

- После оваа постапка проектот е снимен на диск и компјутерот може безбедно да се исклучи или да се работи со нова апликација.

За да се отвори проектот повторно потребно е да се спроведе следната процедура:

- Од мени се одбира опцијата File->Open Project…Ctrl+F11 и се добива прозорец како на сликата 2.4.



Слика 2.4. Прозорец за отворање на постоечки проект

– Во главниот дел на прозорецот се наоѓа список на снимени проекти. Саканиот проект треба да се селектира и неговото име ќе се појави десно од File name . Потоа треба да се активира копчето Open.

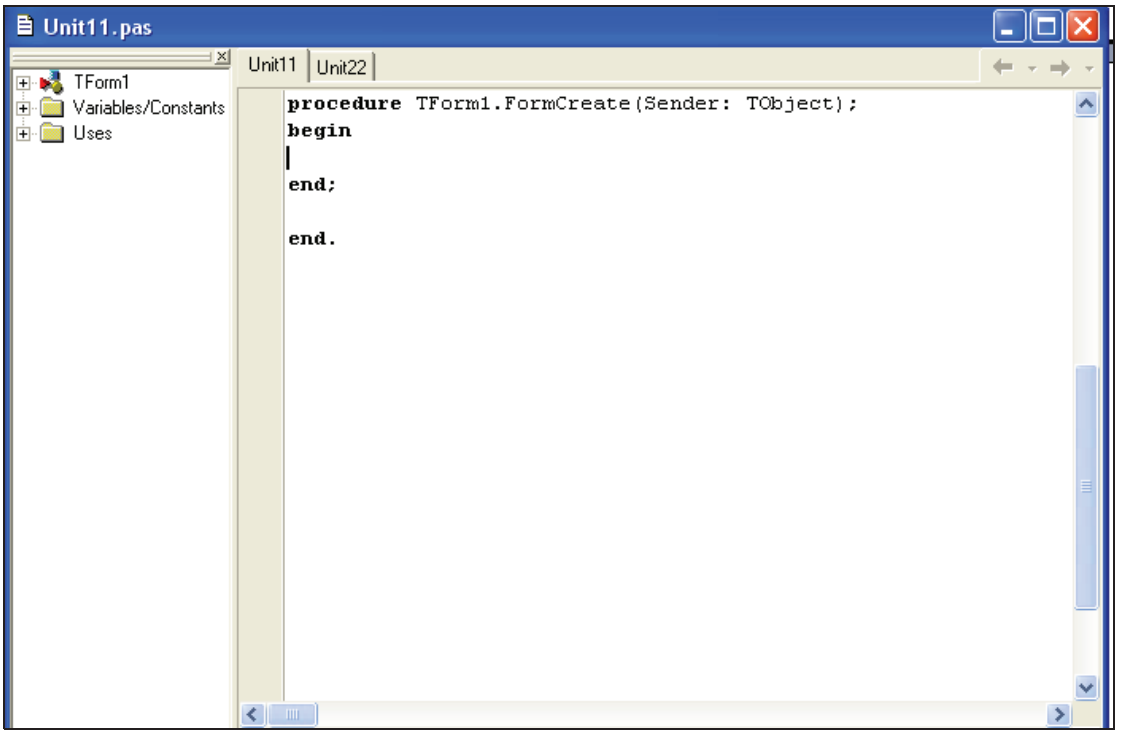
2.6. Образец (форма) и нагодување на неговите својства

На сликата 2.1. гледаме дека насловната линија на образецот го содржи името Form1. Овој наслов име го доделува самиот Delphi. За корисникот, насловот на образецот би требало поблиску да ја одредува наговата суштина. **За да се замени насловот на образецот неопходно е да се користи Object Inspector** .

Својството Caption го чува насловот на образецот. Тоа ќе го промениме од Form1 во Obrazec1. Се менува и изгледот на образецот.

Името на образецот може да се промени со користење на својството Name. Со ова својство на образецот му се задава име кое ќе биде користено во апликацијата. Името, кое исто така е Form1, ќе го промениме во Primer1.

Својствата на компонентите може да се менуваат и со користење на програмски код. До прозорецот за програмски код најлесно доаѓаме ако двапати кликнеме со глувчето на компонентата, во случајот, на формата. Ќе се појави прозорец како на слика 2.5.



Слика 2.5. Прозорец на програмскиот код

Со двојното кликнување на образецот е генериран настан кој произвел формирање на управувач на настани во облик на програмски код даден на следниот листинг.

```
procedure TForm1.FormCreate(Sender: TObject);
begin

end;
end.
```

Овој програмски код Delphi го формира автоматски. Се работи за Pascal код. Во името на процедурата се наоѓа зададеното својство Name на формата. Во првиот случај тоа е Form1 , а во вториот случај Primer1.

```
procedure TPrimer1.FormCreate(Sender: TObject);
begin

end;
end.
```

Пример 1: Меѓу begin и end ќе ја внесеме следната наредба:

Primer1.Caption:=’Novo ime’;

Со оваа наредба му наложивме на Delphi да го смени насловот на образецот во Novo ime. Во програмскиот код на објектот му се пристапува преку неговото име. Апликацијата ја извршуваме со наредбата **Run** од мени.

Проектот треба да се зачува и за таа цел правиме нов фолдер **Vezbanja**. По снимањето, забележуваме дека се произведени осум датотеки. За секоја форма постојат три датотеки:

- Датотека со изворен код на програмата која ја опишува формата и сите потпрограми (тип **.pas** – PAScal source code), која се појавува со внесување на текстот во прозорецот на уредувачот на код;
- Бинарна датотека на преведениот изворен код(тип **.dcu** – Delphi Compiled Unit) која ја формира Delphi пред самото извршување на проектот;
- Бинарна датотека на формата (тип **.dfm** – Delphi graphical form file) која ја опишува формата, а која се формира при снимање на формата;

Ако проектот користи повеќе форми, за секоја од нив се креираат сите три наведени датотеки. Во секој Delphi - проект постојат и следните 5 датотеки.

- Главна датотека на проектот (тип **.dpr** – Delphi Project File) која Delphi автоматски ја прави во текот на проектирањето, а содржи информации за останатите програмски модули кои му припаѓаат на проектот, како и изворен текст на главната програма на целата апликација;
- Текстуална датотека на опциите на проектот (тип **.dot** – Delphi project option file), која содржи општи параметри на проектот;
- Текстуална датотека на конфигурација на проектот (тип **.cfg** Configuration file), која содржи општи параметри за Delphi околината;
- Бинарна датотека на ресурсите на проектот (тип **.res** compiled Resource file), која ја содржи иконата на апликацијата;
- Бинарна датотека со извршен код на апликацијата (тип **.exe** compiled EXEcutable file), која Delphi ја креира пред самото извршување на проектот.

2.7. .Додавање компоненти во формата (Label, Button, Edit)

Конечно сме во можност на формата да поставиме некои компоненти кои ќе направат нешто конкретно.

Пример 2: Да се направи нова апликација, при што името на формата да е Primer2, а нејзиниот натпис Obrazec2. Својствата треба да се променат со помош на Object Inspector.

Секоја форма може да се сфати како кутија во која може да се сместува одреден број компоненти. Компонентите се бираат од палетата на компоненти.

Одбраната компонента во образецот ја поставуваме на еден од следните начини:

-со двоен клик на компонентата, со што таа се донесува на средина на формата,

-со клик на компонентата, а потоа со клик на формата на местото каде сакаме да се донесе компонентата.

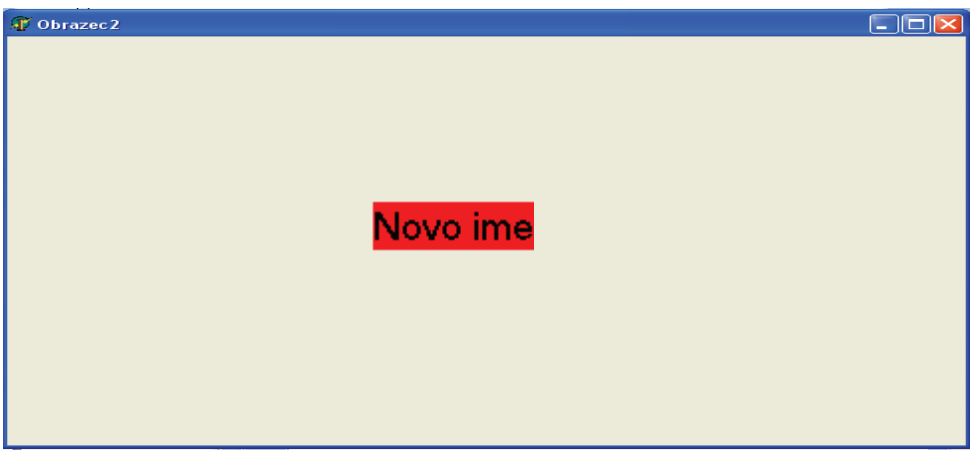
Положбата на секоја компонента во формата може да се менува.

Првата компонента која ќе ја ставиме во формата е Label компонентата. Таа овозможува појавување на статичен текст на екранот. Лабелата има свој натпис кој Delphi автоматски и го доделува. За да може да се променат својствата на лабелата, таа треба да е селектирана (во фокус) и во Object Inspektor, да се промени својството Name во Labela1, а потоа својството Caption во Prv objekt. Својството Font се поставува на Bold со големина 10.

Промена на уште неколку особини на лабелата ќе извршиме со помош на програмски код , кој ќе се активира со двоен клик на лабелата. Изгледот на процедурата е прикажан во следниот листинг:

```
Procedure TPrimer2.Labela1Click(Sender: TObject);
begin
  Labela1.color:=clRed;
  Labela1.Caption:='Novo ime';
end;
```

Со извршување на оваа програма се добива прозорец како на сликата2.6.



Слика 2.6. Промена на бојата и натписот на лабелата.

Промената на бојата и натписот на лабелата се овозможени со двата реда програмски код. Некои особини на објектот може да се менуваат преку Object Inspector и со програмски код; некои само со програмски код, а некои се само за читање и не можат да се менуваат.

Button е следната компонента со која се запознаваме. Тоа е класично Windows копче. Насловот на копчето се задава во својството **caption**. На страната **Events** се дефинира настанот на кој ќе реагира копчето со користење на соодветни методи. За таа цел создаваме нова апликација.

Пример 3: Во образецот ќе ставиме две компоненти: една лабела и едно командно копче. Во табелата ќе бидат прикажани својствата на компонентите кои ќе се променат, заради прегледност.

Положбата на лабелата во формата е прикажана со три црни квадратчиња, но може и воопшто да не се гледа , ако не е во фокус. Причина за ова е вредноста на својството AutoSize (true). Delphi ова својство автоматски го поставува на вредност True. Ако корисникот избере вредност False, просторот во формата за лабелата , ќе биде толкав колкава што е лабелата. Во мала лабела ќе може да се смести подолг текст, ако вредноста на својството е True.

Име на компонентата	Име на својството	Вредност на својството
Form1	Name	Primer3
	Caption	Obrazec3
Label1	Name	Natpis1
	Caption	Prazen
Button1	Name	Startuvaj
	Caption	Startuvaj
	Font	Bold,10

Табела2.1. Вредности на својствата на компонентите

За да се активира апликацијата, потребно е на командното копче Startuvaj да се напише соодветен програмски код. Со двоен клик на копчето се активира управувачот со настани и ни овозможува пишување на код. Со апликацијата сакаме да го постигнеме следното: при активирање на апликацијата, во просторот на лабелата да се појави текстот “Ovoj tekst se pojavuva vo labela”.
Програмата се состои од една програмска линија:

```
Natpis1.Caption:=’Ovoj tekst se pojavuva vo labela’
```

Ако го активираме настанот, со кликање со глувчето на копчето, ќе се изврши програмскиот код за тој настан.

Во програмскиот код ги додаваме следните два реда :

```
Naptis1.Font.Style:=[fsBold];
Natpis1.Font.Size:=12;
```

Едно од правилата кои треба да се почитуваат при креирање на апликацијата е дека покрај можноста за стартување на апликацијата, треба да се обезбеди можност за прекин на работата на апликацијата. На Образец3 додаваме командно копче со исто име и натпис Kraj. На клик на ова копче се отвора програмски код во кој се додава само една наредба Close, која има задача да ја затвори стартуваната апликација.

Edit е компонента за внесување текст само во еден ред. На внесениот текст му се придружува својството **Text.** Со користење на овој објект може да се внесува или чита текст. Ако во овој објект се внесе број , тој мора да се претвори за да се користи како број.

Пример 4: Да се направи нова форма во која ќе сместиме две лабели, две Edit кутии и две копчиња. Својствата на користените компоненти ги прикажуваме во табелата 2.2.

Текстот кој се внесува во Edit кутија се чува во својството Text на компонентата. Идејата на задачата е текстот внесен во првата Edit кутија да се појави во втората.

Име на компонентата	Име на својството	Вредност на својството
Form1	Name	Primer4
	Caption	Obrazec4
Label1	Name	Natpis1
	Caption	Vnesete Ime i Prezime
	Font	Bold,10
Label2	Name	Natpis2
	Caption	Vnesenoto Ime i Prezime e
	Font	Bold,10
Edit1	Name	Vnes
	Text	Prazno
	Font	Italic,10
	Color	Blue
Edit2	Name	Prikaz
	Text	Prazno
	Font	Italic,10
	Color	Red
Button1	Name	Startuvaj
	Caption	Startuvaj
	Font	Bold,10
Button2	Name	Kraj
	Caption	Kraj
	Font	Bold,10

Табела2.2. Вредности на својствата на компонентите

Програмскиот код кој му припаѓа на копчето Startuvaj, на настанот клик, е следниот:

```
Prikaz.text:=Vnes.text;
```

Откако ќе се стартува апликацијата, треба да се внесе име и презиме во Edit1 кутијата, а потоа да се кликне на копчето Startuvaj.

Некои својства на Edit контролата

Text – е варијабла од тип string во која се содржи внесениот текст.
Enabled – логичка вредност која укажува на тоа дали контролата може да прима настани од тастатура или глувче. Ако е со вредност false , контролата ќе биде бледосива и не може да се внесуваат податоци.
Read Only – корисникот има можност само да чита податоци од таа контрола, а не и да пишува во неа.
Set Focus – го поставува фокусот (покажувачот) на контролата која ја повикува.
Password Char – го дефинира знакот со кој ќе се менува внесениот текст.
SelText – својство кое ни го дава сиот означен текст во контролата.

Настани на Edit контролата

On Change – настанува секогаш кога ќе се измени содржината (text) на Edit контролата. Користете го настанот за обработка на внесени податоци од страна на корисникот кога ќе настапи тој настан.
On Enter –настапува кога контролата добива фокус за внес на податоци.
On Exit – настапува кога контролата го губи фокусот за внес на податоци.

ПОГЛАВЈЕ 2
(НАКРАТКО)

Прозорецот на работната околина на Delphi се состои од четири дела и тоа:

1. Главен прозорец (main) кој се состои од:

а) Насловна - лента;

б) мени - лента;

в) лента со алатки Toolbars;

г) лента со компоненти Component palette.

2. Во левиот дел на прозорецот се наоѓаат:

а) прозорец за хиерархиски приказ на компонентите на апликацијата

б) прозорец за одредување на својствата и настаните на објектите во апликацијата - **Object Inspector**.

Тој има два дела :

- Properties (својства или карактеристики)
- Events (настани).

3. Третиот дел на Delphi околината е прозорец на образец или форма за дизајнирање на апликацијата каде што ги сместуваме компонентите.

4. Четвртиот дел на Delphi околината е прозорец за пишување на програмски код кој се наоѓа зад дизајнерот на форми.

Опцијата **FILE** ги содржи следните подопции: NEW, NEW APLICATION, NEW FORM OPEN, OPEN PROJECT, REOPEN SAVE, SAVE AS, SAVE PROJECT AS, SAVE ALL,CLOSE, CLOSE ALL, PRINT, EXIT;

Опцијата **EDIT** ги содржи:UNDELETE, REDO CUT, COPY, PASTE, SE-LECT ALL, ALIGN TO GRID, BRING TO FRONT, SENT TO BACK, ALIGN, SIZE, SCALE ,TAB ORDER, LOCK CONTROLS;

Опцијата **VIEW** ги содржи: PROJECT MANAGER, OBJECT NSPECTOR, ALIGNMENT PALETTE, UNITS, FORMS, TOOLBARS;

Опцијата **PROJECT** ги содржи:ADD TO PROJECT, REMOVE FROM PROJECT, COMPILE PROJECT;

Опцијата **RUN** ги содржи: RUN, PROGRAM PAUSE, PROGRAM RESET

Опција **HELP**.

Интерпретер е преведувач кој секоја линија од изворниот код ја преведува на машински јазик и ја извршува.

Компајлерот е програма која секоја наредба од изворниот код ја анализира и преведува на машински код, односно, формира извршна програма.

Образецот – формата е основна компонента во Delphi апликацијата.

Својствата на компонентите можат да се менуваат преку **Object Inspector** и со користење на **програмски код**.

Одбраната компонента во образецот ја поставуваме на еден од следните начини:

-со двоен клик на компонентата, со што таа се донесува на средина на формата,

-со клик на компонентата, а потоа со клик на формата на местото каде сакаме да се донесе компонентата.

Прашања и задачи:

1. Ако во Object Inspector се промени својството Name, како тоа ќе се одрази на програмскиот код?
2. Од кои елементи се состои работната околина на Delphi?
3. За што се користи Edit контролата?
4. Објасни ги својствата:
   -Enabled
   -Set Focus
   -PasswordChar
   -Read Only – кога се употребува
5. На кои два начина може да се променат својствата на објектите?
6. Од кои елементи се состои Object Inspector ?
7. За што се користи Label контролата?
8. Објасни го настанот:
   -On Click
9. Која процедура ќе се отвори на настан On Click на копчето Button2 од формата Form2.
10. Што се случува кога двапати ќе се кликне со глувчето на компонентата која се наоѓа на образецот , во процесот на креирање на апликацијата?
11. Што е компајлер, а што интерпретер?
12. Како можат да се прикажат бројни податоци во Edit кутија?
13. Објасни ги настаните:
    -On Enter
    -On Change

Задачи

1: Да се отвори нова апликација, да се одбере формата Form1, во Object Inspector да се одбере Properties, да се најде својството Caption и својството Height и да се постават на следните вредности:

```
Form1.Caption:=’Ova e naslov na prozorecot’;
Form1.Height:=100;
```

2. Да се направи програма со една форма и една Edit кутија. Секогаш кога корисникот ќе ја менува содржината на edit контролата, натписот на формата треба да се менува со внесениот текст (настан OnChange кај Edit контролата). На местото на внесениот текст гледаме само ѕвездички (Password Char да е *).

```
Procedure Tform1.EditChange( );
Begin
  Form1.caption:=edit1.text;
End;
```

3. Да се направи програма со една форма, две Edit кутии и две командни копчиња едното за излез, а другото ја копира селектираната содржина од првата edit контрола во втората.

```
Procedure Tform1.Button1Click( );
Begin
  Edit2.text:=edit1.seltext;
End;
```

4. Да се направи програма во која ќе се искористи својството **Password Char,** а која нема да дозволи затворање на програмата ако не се внесе точна лозинка.

5. Да се направи апликација која содржи две edit контроли. Кога корисникот впишува текст во првата компонента, истиот текст треба да се појавува и во втората edit компонента.

ПОГЛАВЈЕ 3

3.ПИШУВАЊЕ НА КОД ВО OBJECT PASCAL

Во рамките на ова поглавје ќе стане збор за:
-картичката EVENTS;
-најчесто користените или подразбирливите настани: EVENTS за соодветните компоненти;
-улогата на настаните;
-користење коментари во кодот;
-пишување команди за доделување;
-користење променливи (INTEGER, SHORTIN, LONGINT, BYTE, WORD, SINGLE, DOUBLE, EXTENDED, REAL-COMP, REAL, BOOLEAN, CHAR, STRING, POINTER);
-користење константи;
-користење команди за гранење(IF и CASE);
-користење команди за повторување (FOR, WHILE i REPEAT);
-трансформирање на цели броеви во стринг и обратно;
-пишување настани за BUTTON;
-пишување настани за LABEL;
-пишување настани за EDIT;
-пишување настани за LIST BOX;
-користење методи и составни елементи на секоја компонента во даден момент.

3.1 Мојот прв Delphi проект
Програма Здраво

Кога се стартува оваа програма, се јавува прозорецот прикажан на слика 3.1. Прозорецот содржи една форма, три командни копчиња и една празна текст кутија Edit. Кога ќе се кликне на командното копче Прикажи, текстот “ЗДРАВО НАРОДЕ!” ќе се појави во текст кутијата. Кога ќе се кликне на копчето Бриши, текстот се брише, а кога ќе се кликне на копчето Излез, програмата завршува.

Промена на Caption својството на формата:

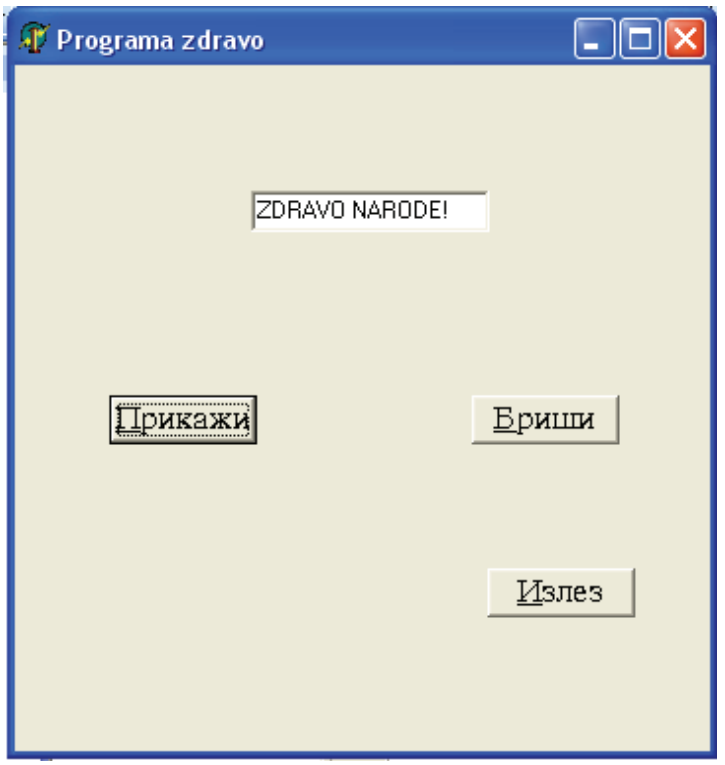
- Се селектира формата,
- Во прозорецот Properties на Object Inspector го бираме својството Caption,
- Во полето десно од Caption, текстот Form1 го заменуваме со “Program zdravo”

Табела на својства на компонентите на Програма Здраво:

Object	Properties	Settings
Form	Name Color Caption	FrmZdravo Blue Program Zdravo
Button1	Name Caption Font Size	CmdIzlez &Izlez Macedonian 14
Button2	Name Caption	CmdBrisi &Brisi

Button3	Name Caption	CmdPrikazi &Prikazi
Edit1	Name Text	TxtPrikazi

(знакот & го подвлекува знакот пред кој е ставен и овозможува копчето да се активира преку тастатура со истовремено притискање на Alt + копчето)



Слика 3.1

Придружување код на објектите

Кодот се извршува како резултат на некој настан. Ако кликнеме на копчето Излез за време на извршување на програмата Програма Здраво, тогаш автоматски се генерира настанот (event) On Click, кој автоматски го извршува програмскиот код што е придружен на овој настан. Тоа значи дека работата се сведува на извршување на соодветен код доделен на настанот на објектот.

Придружување код на командното копче Прикажи:

- I - начин
  - Се селектира копчето
  - Од картичката настани (events ->Object Inspector) се избира настанот On Click,
  - Двојно се кликнува во полето десно од избраниот настан, при што се отвора процедура во која го запишуваме програмскиот код.
- II – начин
  - Кликнуваме двапати врз командното копче Прикажи. Во прозорецот за кодот автоматски се генерира името на процедурата и името на објектот кај кој е активен настанот кој ќе го предизвикува даденото дејство.

```
Procedure TFrmZdravo.CmdPrikaziClick(Sender: TObject);
begin
        txtPrikazi.text:='Zdravo narode!';
end;
```

Придружување код на командното копче Бриши, на настанот OnClick:

```
procedure TFrmZdravo.CmdBrisiClick(Sender: TObject);
begin
        txtPrikazi.text:='  ';
end;
```

Следниот код е за копчето Излез:

```
procedure TFrmZdravo.CmdIzlezClick(Sender: TObject);
begin
        Application.terminate;
end;
```

Задача1: Да се направат промени во претходната програма во Delphi, при што во извршниот мод го добиваме прозорецот од слика 3.2.

.

Слика 3.2.

3.2.Елементи на јазикот

Множество на знаци

Тука спаѓаат сите букви од англиската алфабета, десет децимални цифри и специјални знаци како +, -, *, /, ; , : , [,] , { , } итн. Во службените делови на програмата не се прави разлика меѓу големи и мали букви.

Коментари

Коментарите се објаснувања на читателот на програмата. Нив преведувачот ги запоставува. Се пишуваат меѓу пар големи загради { i }, или меѓу пар загради составени од два знака (* i *).

На пример: *{ Ova e komentar} , (* I ova e komentar*).*

Идентификатори

Идентификаторите служат за означување на сите видови елементи на програмата: променливи, константи, типови на податоци, модули, потпрограми итн. Се состојат од букви, знаци, долна црта _ , при што првиот знак не смее да биде цифра. За полесно разбирање на програмата, идентификаторите треба да бидат имиња кои ќе ги опишуваат објектите кои ги означуваат.

Службени зборови и стандардни директиви

Во Pascal, а со тоа и во Delphi постојат службени зборови кои означуваат наредби, оператори, константи и други службени елементи на јазикот. Тие се резервирани зборови и не смеат да се користат како идентификатори на други објекти во програмите. Службени зборови во Delphi се: var, array, unit, and, case, begin, type итн.

3.3.Податоци

Тоа се објекти кои се обработуваат во програмите. Основна карактеристика на податокот е неговиот тип. Типот ги одредува можните вредности на податоците и можните операции врз нив.

Целобројни типови на податоци се подмножества на целите броеви. Тие се разликуваат по опсегот на вредности: Integer (-2147483648-2147483647), Longinteger (-2147483648-2147483647) , Cardinal (0-4294967295), Shortint (-128-127), Byte(0-255), Word (0-65535) итн.

Реални типови на податоци претставуваат подмножества на рационалните броеви. Тука спаѓаат : Real ($\pm$($5.0*10^{-324}$-$1.7*10^{308}$)), Single($\pm$($1.5*10^{-45}$-$3.4*10^{38}$)) , Double($\pm$($5.0*10^{-324}$-$1.7*10^{308}$)), Extended($\pm$($3.4*10^{-4932}$-$1.1*10^{4932}$)), , Currency …

Логички типови на податоци служат за претставување на резултатот на логичкиот израз, кој може да биде точен или неточен. Во Delphi постојат Boolean, ByteBool, WordBool, LongBool.

Знаковни типови на податоци служат за претставување на букви, цифри и специјални знаци, кои можат да се јават во разни текстови. (Char, AnsiChar и WideChar)

Константи се податоци чии вредности не можат да се променат при извршувањето на програмата.

```
Const
        Konstanta=vrednost;
```

Пример:

```
        Pi=3.14159265359;
        Eps=1E-9;
```

Променливи се податоци чии вредности можат да се променат во текот на извршувањето на програмата. Се претставуваат со помош на идентификатор. Се дефинираат во делот **var** (variable – променливи) за дефинирање променливи со следниот општ облик:

```
Var Promenliva1, Promenliva2,…,PromenlivaN: Tip;
```

Пример:

```
Var
        Pocetok, Kraj:Integer;
        Znak: Char;
```

```
        Dijametar, Krug: Real;
        P,Q,R,S,T:Boolean;
```

Дефинирање на типови. Под ова се подразбира доделување идентификатор на типовите кои ги дефинира корисникот. Типовите се дефинираат во декларативниот дел Type, кој служи за дефинирање на типови, чиј општ облик е:

```
Type
        ImeNaTip=OpisNaTip;
```

Пример:

```
    Type
        TDen=(Pon, Vto, Sre, Cet, Pet, Sab, Ned);
        Tdatum=1..31;
        TNiza=Array[1..10] of Integer;
```

3.4.Оператори и изрази

Оператори претставуваат активности кои се извршуваат над операндите (податоците) и даваат одредени резултати. Изразите се произволно сложени системи од оператори и операнди.

Аритметички оператори

Тие служат за изведување на основните аритметички операции. За собирање+, за множење *, за одземање -. Типот на резултатот на овие оператори е целобројен ако двата операнди се целобројни, инаку е реален.

За делење постои оператор / кој секогаш дава резултат од реален тип и оператор div (целобројно делење) , кај кој двата операнди мора да се целобројни, но и резултатот е целобројен.

Постои и оператор mod чии операнди се целобројни, а исто така и резултатот е целобројен. Овој оператор како резултат дава остаток од делењето.

Релациски оператори

Служат за споредување на нумерички и пребројиви типови на податоци. Резултатите од овие оператори се True или False. Тука спаѓаат: =, >, <, <>, <=, >=.

Логички оператори

Логичките оператори служат за изведување логички оператори над логички податоци и даваат логички резултати. Тука спаѓаат not, and, or, xor.

Приоритет на операторите

Приоритет	Оператори
Највисок	not
	* / div mod and
	+ _ or xor
Најнизок	= <> < <= > >=

Табела 3.1.

3.5.Наредба за доделување на вредност

Ова е наредба со чија помош резултатот од некоја пресметка се сместува во некоја променлива. Општ облик на наредбата е:

```
        Promenliva := izraz;
```

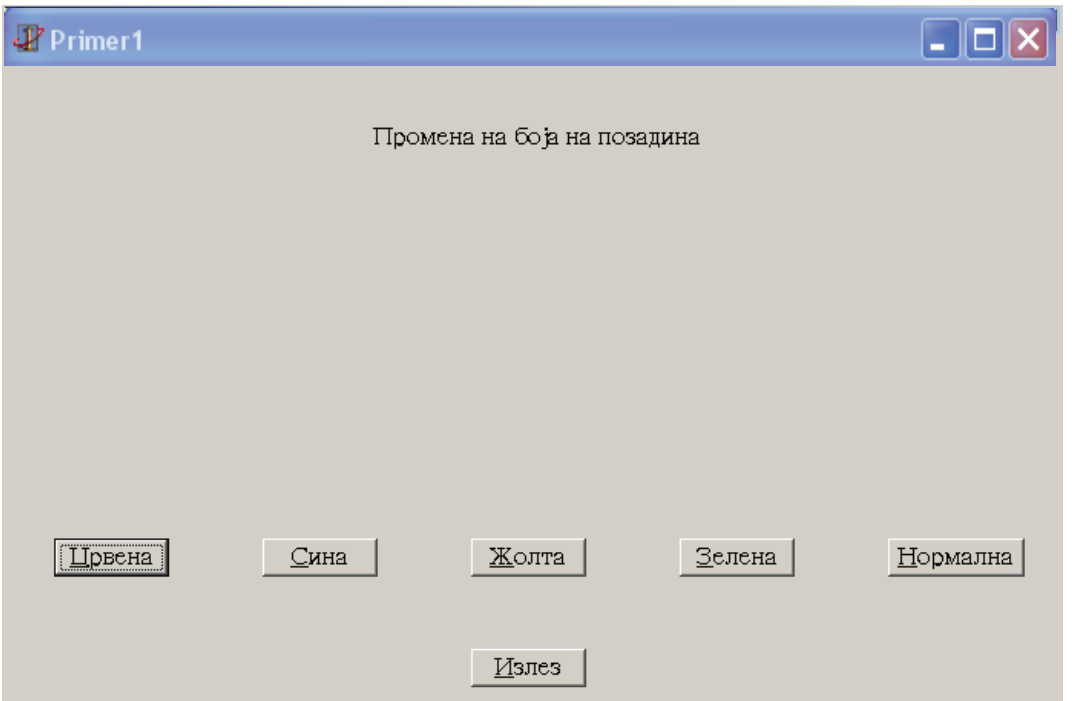
каде што Promenliva е место во меморијата во која се сместува резултатот, а Izraz е изразот чија вредност се пресметува.

Пример:

```
I : =  J + K div L;
A : = S*12.5;
```

Променливата и изразот треба да се сложуваат по тип.

Задача2: Да се направи програма во Delphi со која ќе се менува бојата на позадината на формата. На формата се поставуваат 5 командни копчиња со следните натписи: Црвена, Сина, Жолта, Зелена, Нормална. Со кликање на соодветното копче да се промени бојата на формата. Натписот на формата да биде Primer1.Во извршниот мод го добиваме прозорецот од слика 3.3.



Слика 3.3.

Табела на својства

Object	Properties	Settings
Form1	Caption	Primer 1
Label1	Caption	Промена на боја на позадина
Button1	Caption	Црвена
Button2	Caption	Сина
Button3	Caption	Жолта
Button4	Caption	Зелена
Button5	Caption	Нормална

Табела 3.2.

(Во овој пример не се менуваат својствата Name на компонентите)

Кодот кој се придружува на копчињата е:

```
Procedure TForm1.Button1Click ( );
Begin
    Form1.color:=clred;
End;
```

```
Procedure TForm1.Button2Click ( );
Begin
     Form1.color:=clblue;
End;

Procedure TForm1.Button3Click ( );
Begin
    Form1.color:=clyellow;
End;

Procedure TForm1.Button4Click ( );
Begin
    Form1.color:=clgreen;
End;

Procedure TForm1.Button5Click ( );
Begin
    Form1.color:=clbtnface;
End;
```

Задача 3: Да се отвори нова апликација. Настанот Click на формата да се програмира, така што треба да испише на формата текст ПОЗДРАВ!

Решение: Се активира нова апликација . Се активира настан Клик преку Object Inspector и се пишува следниот код:

```
Canvas.Textout(10,10,' pozdrav!');
```

Задача 4: Текстот "поздрав" да се појавува во точка со случајни координати.

Решение:

```
procedure TForm1.FormClick(Sender: TObject);
var x,y:integer;
begin
    x:=random(690);
    y:=random(480);
    Canvas.Textout(x,y,' pozdrav!');
end;
```

3.5.1.Објект Canvas

Тој е подобјект на формата и служи за исцртување и испишување на текст по неа. Претставува замена за наредбите Read / Write од Pascal, како и некои графички наредби. Методот TextOut испишува некој текст на зададените координати.

Задача 5: Да се направи апликација, која ќе го прикаже вториот прозорец ако се кликне на првиот.

Решение: Да се отвори нова апликација, да се додаде нова форма, да се програмира настанот клик на формата1 со наредбата:

```
Form2.show.
```

Новата форма се додава со иконата New form.

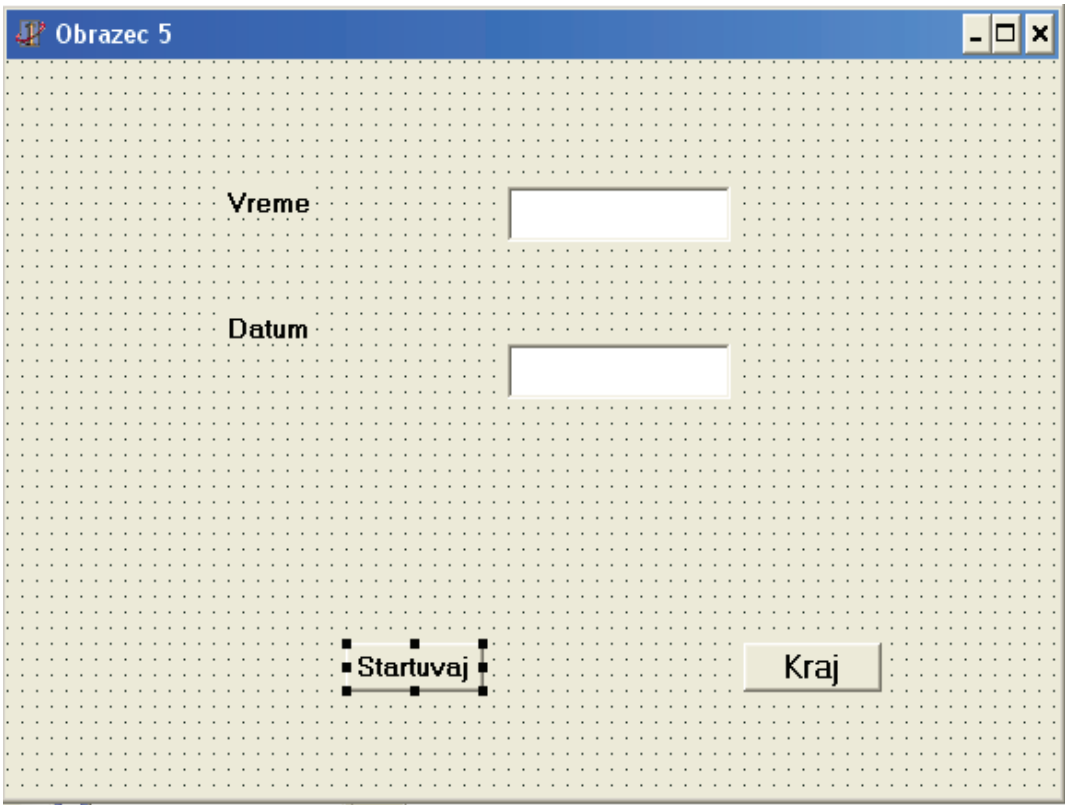
Методот Show ја прикажува зададената форма. Апликациите кои имаат повеќе прозорци, при активирање ги иницијализираат сите прозорци, но видлив е само еден – почетниот прозорец, кој најчесто го содржи интерфејсот, а останатите прозорци можат да се прикажат по потреба како одговор на некоја акција на корисникот.

 **3.5.2.Компонентата Timer (часовник)** се наоѓа во палетата на компоненти на страната System. Компонентите на Delphi се делат на видливи и невидливи, во зависност од тоа дали при извршувањето на апликацијата тие се видливи или не. Поголемиот дел од компонентите се видливи. Tajmer е невидлива компонента, која се користи за да го активира настанот On click во зададен временски интервал . Тајмерот е некој вид штоперица, која периодично извршува дел од програмскиот код , без оглед на состојбата на програмата.

Задача 6: Да се направи нова апликација која ќе ни го прикажува тековното време и датум. Формата содржи две лабели, две edit кутии и две копчиња.

Име на компонентата	Име на својството	Вредност на својството
Form1	Name	Primer5
	Caption	Obrazec5
Label1	Name	Natpis1
	Caption	Vreme
	Font	Bold,10
Label2	Name	Natpis2
	Caption	Datum
	Font	Bold,10
Edit1	Name	Vreme
	Text	Prazno
	Font	Bold,12
Edit2	Name	Datum
	Text	Prazno
	Font	Bold,12
Button1	Name	Startuvaj
	Caption	Startuvaj
	Font	Bold,10
Button2	Name	Kraj
	Caption	Kraj na rabotata
	Font	Bold,10

Табела 3.3. Својства и вредности на компонентите во примерот



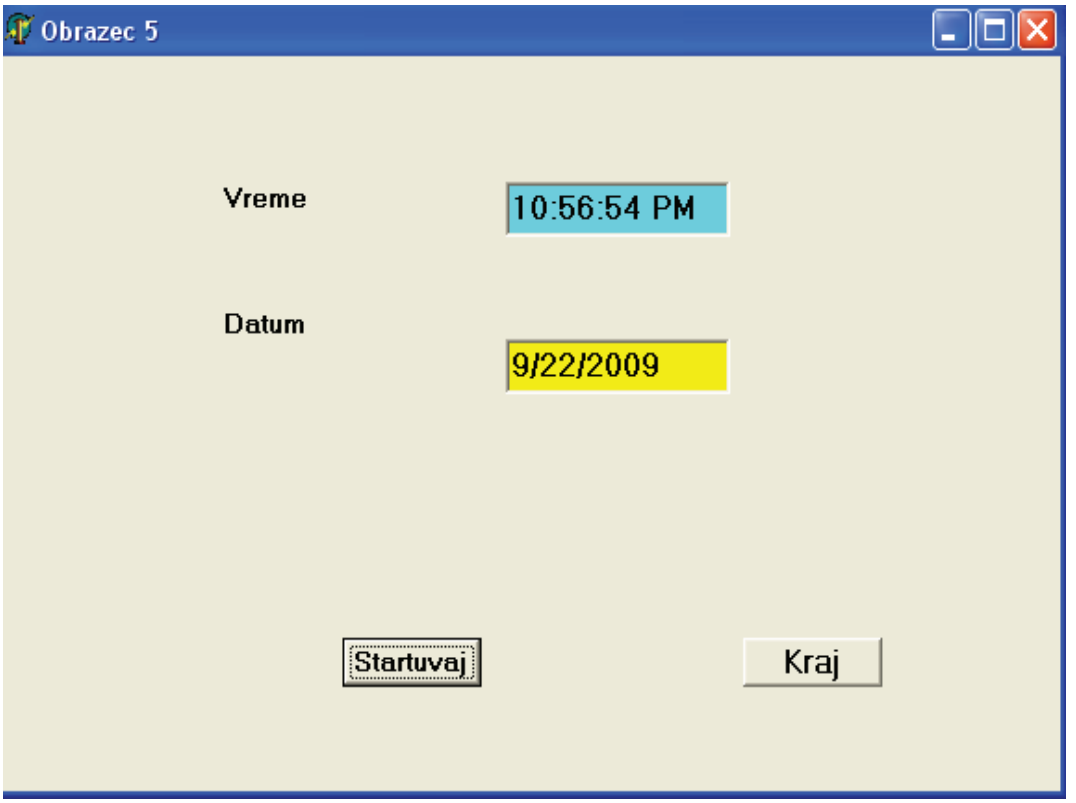
Слика 3.4. Кориснички интерфејс за разгледуваниот пример

Програмскиот код е напишан за настанот клик на копчето Startuvaj:

```
procedure TPrimer5.StartuvajClick(Sender: TObject);
begin
        Vreme.text:=timetostr(time);
        Vreme.color:=claqua;
        datum.text:=datetostr(date);
        datum.color:=clyellow;
end;
```

Анализирајќи го програмскиот код, забележуваме нова функција за конверзија на време и датум во стринг. Ова мора да се направи затоа што Edit кутијата може да прифати само текст. Откако ќе се стартува апликацијата, во Edit кутиите се прикажува време и датум. Очекуваме времето да се менува , но тоа не се случува. Проблемот се решава со додавање на компонентата тајмер и со мала промена на програмскиот код. Timer-от се наоѓа на страната System од палетата на компоненти. Timer от е невидлива компонента , но на формата се гледа. При извршување на апликацијата не се гледа. Ги менуваме својствата на Timer компонентата: Name (Saat) , Enabled (False). Другите две својства не ги менуваме. Важно својство кое влијае на брзината на работа на тајмерот е својството Interval. Вредноста поставена на 1000 произведува настани на една секунда, што одговара на работа на часовник.

Со стартување на апликацијата се добива прозорец, како на слика 3.5, при што во Edit кутијата се прикажува реално време.



Слика 3.5. Кориснички интерфејс за разгледуваниот пример со Timer

Променетиот програмски код е следниот:

```
procedure TPrimer5.KrajClick(Sender: TObject);
begin
        Close;
end;

procedure TPrimer5.StartuvajClick(Sender: TObject);
begin
        saat.enabled:=true;
end;

procedure TPrimer5.satTimer(Sender: TObject);
begin
        Vreme.text:=timetostr(time);
        Vreme.color:=claqua;
        datum.text:=datetostr(date);
        datum.color:=clyellow;
end;
```

3.6. Проектирање на апликации од линиска структура

После запознавањето со компонентите Label, Edit, Button и Timer може да решаваме проблеми од проста линиска алгоритамска структура. Овие

програми имаат својство секоја наредба да се изврши само еднаш во текот на едно извршување на програмата.

Програмите од оваа структура напишани во Pascal содржат наредби за внесување на податоци, за обработка на податоци и за прикажување резултати.

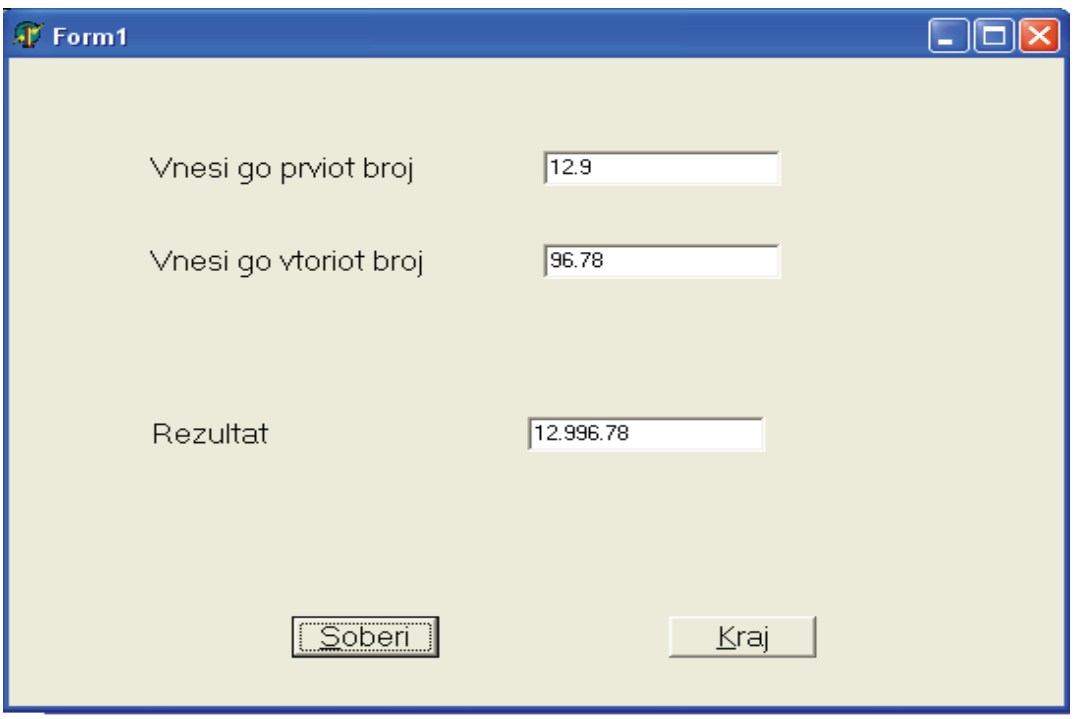
Задача1: Да се напише апликација која ќе собира два броја и ќе го прикажува нивниот резултат.

Решение: Потребни се две Edit кутии за внесување на броевите и една Edit кутија за резултатот. Потребни се три лабели и две копчиња.

Треба да се размисли за следниот програмски код:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    edit3.text:=edit1.text+edit2.text;
end;
```

Со извршување на овој проект, со внесување на потребните вредности во соодветните Edit кутии и со клик на копчето Soberi се добива прозорец како на сликата 3.6.



Слика 3.6.

Очигледно е дека резултатот не е добар. Тој е добиен со надоврзување на овие два броја. Апликацијата ги третира внесените броеви како стрингови, а операцијата како надоврзување на стрингови (конкатенација).

Проблемот мора да се реши во програмскиот код со правилно доставување на податоци на апликацијата.

```
procedure TForm1.Button1Click(Sender: TObject);
Var
        a,b,c:real;
 begin
```

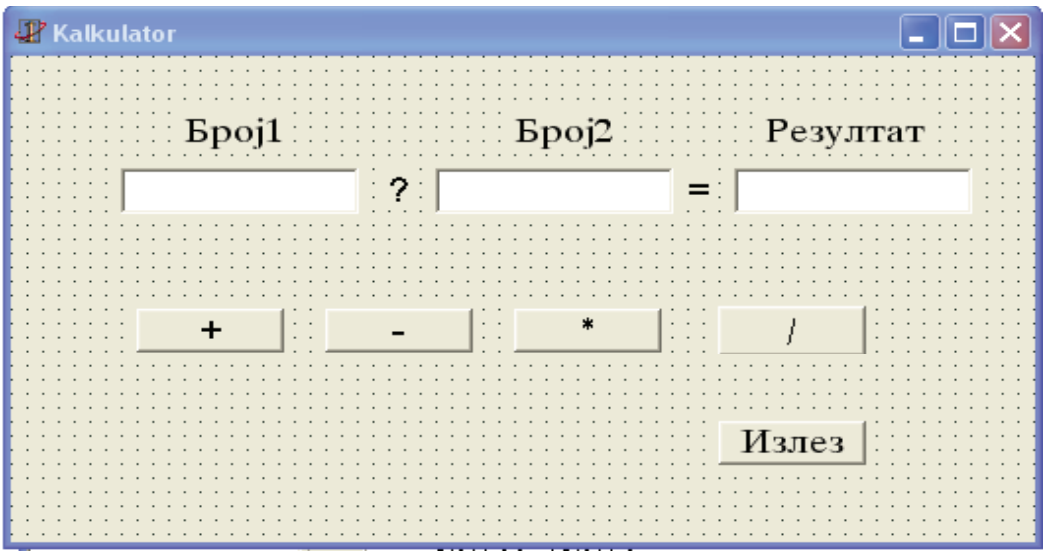
```
        a:=StrToFloat(edit1.text);
        b:=StrToFloat(edit2.text);
        c:=a+b;
        edit3.text:=FloatToStr(c);
end;
```

3.6.1. Претворање на податоци

Во Edit контролата можеме да внесуваме податоци само во облик на String. Ако треба внесените податоци да се нумерички (број), тогаш користиме функции за конверзија.

Function StrToInt (const s:string):Integer;
(претвора string – текст во цел број)
Function StrToFloat(const s:string):Extended;
(претвора string – текст во децимален број)
Function IntToStr(value:Integer):String;
(претворање од цел број во string)
Function FloatToStr(value:Extended):String;
(претворање од децимален број во string)
function TimeToStr(Time: TDateTime): string;
(претворање од време во string)
function DateToStr(Date: TDateTime): string;
(претворање од датум во string)

Задача2: Да се направи програма за пресметување на збир, разлика, производ и количник на 2 броја. Внесувањето се врши преку две Edit кутии, а исто и резултатот се прикажува во Edit кутија.



Слика 3.7.

I - начин

```
Procedure TForm1.Button1Click( );
        Var br1,br2,rez:Integer;
Begin
        Br1:=strtoint(edit1.text);
        Br2:=strtoint(edit2.text);
```

```
            Rez:=br1 + br2 ;
            Edit3.text:=inttostr(rez);
            Label5.caption:='+';
        End;

        Procedure TForm1.Button2Click( );
            Var br1,br2,rez:Integer;
        Begin
            Br1:=strtoint(edit1.text);
            Br2:=strtoint(edit2.text);
            Rez:=br1 - br2 ;
            Edit3.text:=inttostr(rez);
            Label5.caption:='-';
        End;

        Procedure TForm1.Button3Click( );
            Var br1,br2,rez:Integer;
        Begin
            Br1:=strtoint(edit1.text);
            Br2:=strtoint(edit2.text);
            Rez:=br1 * br2 ;
            Edit3.text:=inttostr(rez);
            Label5.caption:='*';
        End;

        Procedure TForm1.Button4Click( );
            Var br1,br2:Integer;
             Rez:Real;
        Begin
            Br1:=strtoint(edit1.text);
            Br2:=strtoint(edit2.text);
            Rez:=br1 / br2 ;
            Edit3.text:=floattostr(rez);
            Label5.caption:='/';
        End;
```

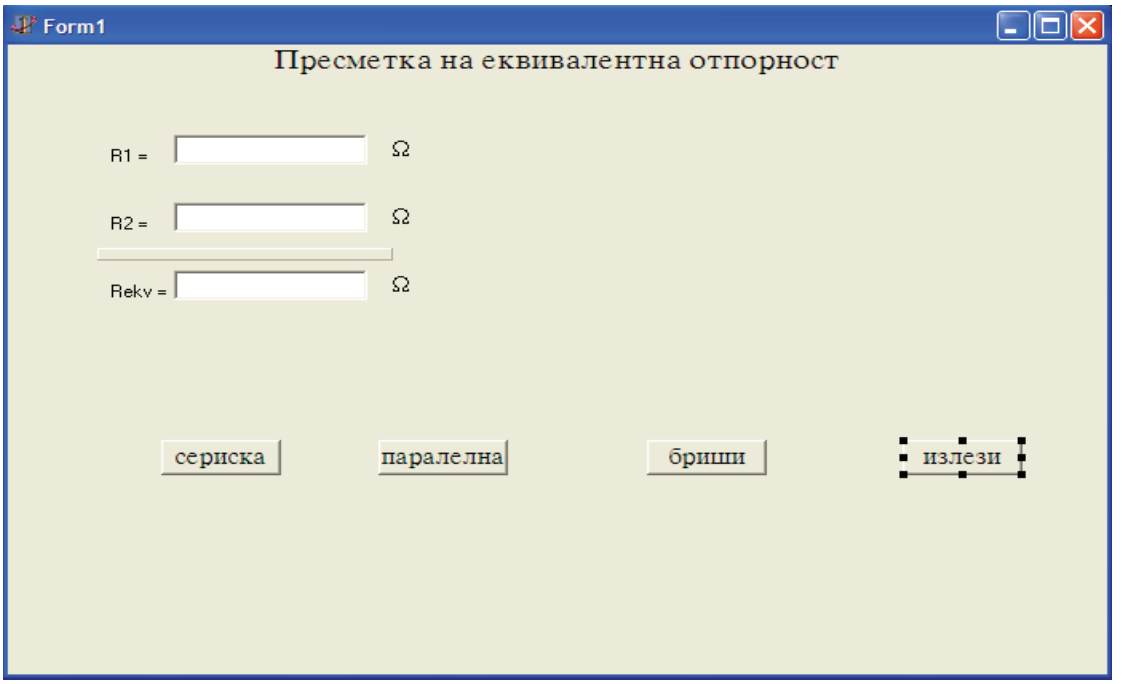
II начин

```
        Procedure TForm1.Button1Click ( );
        Begin
            Edit3.text:=inttostr(strtoint(edit1.text)+strtoint(edit2.text));
            Label5.caption:='+';
        End;
```

За останатите копчиња кодот се пишува на сличен начин. За копчето "делење(/)" треба да се води сметка за типот на резултатот .

Задачи:

3. Да се направи програма во Delphi за пресметка на еквивалентна отпорност на сериски и паралелно поврзани отпорници.



Слика 3.8.

4. Со клик на формата да се промени натписот на формата во „Klikna na mene!!!“

5. Да се напише апликација во која се вчитуваат вредностите на катетите a и b на правоаголен триаголник, се пресметува хипотенузата c, двата остри агли и неговата површина.

Упатство: c=sqrt(sqr(a)+sqr(b)), alfa=arctan(a/b)*180/Pi, beta=arctan(b/a)*180/Pi, P=a*b/2.

3.7. Разгранети структури

Овие структури овозможуваат условно извршување на една или на ниедна наредба од множеството на една или повеќе наредби.

Општ облик на IF наредба или наредба за избор од две можности

If uslov then Naredba1 else Naredba2 или

If uslov then Naredba1,

каде што услов е логички израз , Naredba1 е наредба која се извршува ако условот е исполнет, а Naredba2 е наредба која се извршува ако условот не е исполнет.

Пример: If A > B then C:=A else C:=B;

Општ облик на CASE наредба или наредба за избор од повеќе можности

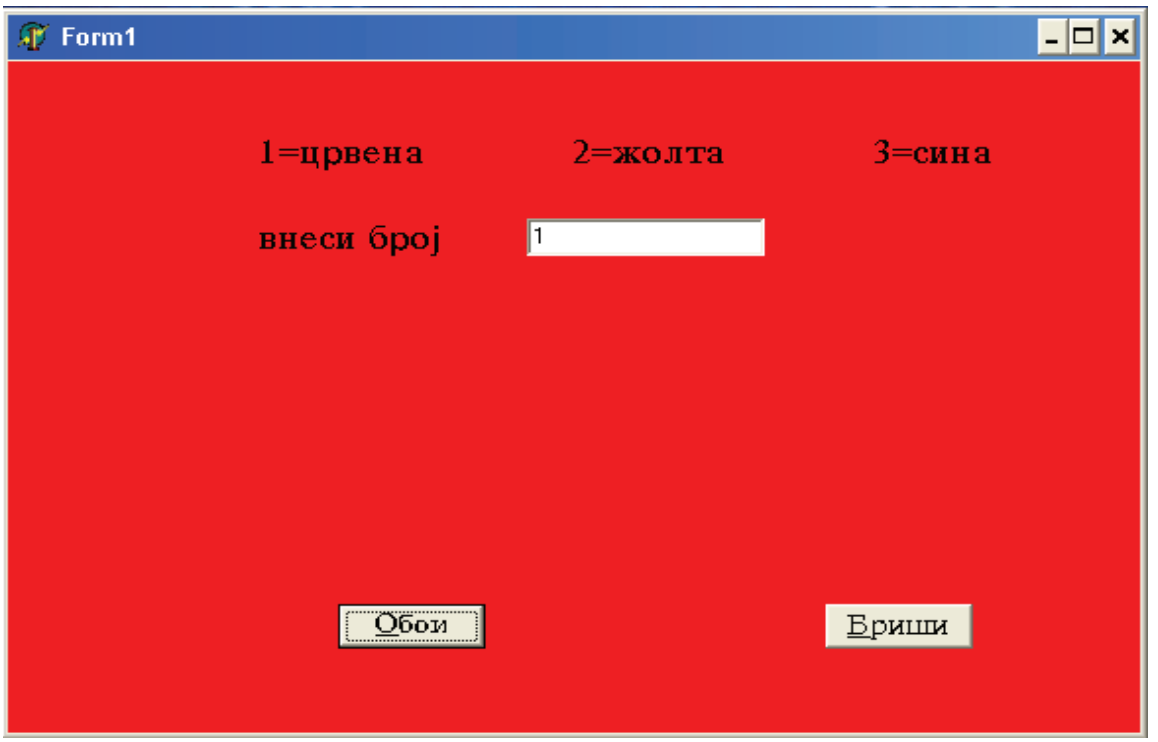
```
Case Izraz of
        NizaNaVrednosti : Naredba;
        NizaNaVrednosti : Naredba;
        …..
        NizaNaVrednosti : Naredba
        Else Naredba;
End;
```

каде што Izraz е израз од кој било пребројив тип, освен Longint чија вредност се користи при одлучувањето, а NizaNaVrednosti се константи од ист тип како Izraz. Прво се пресметува Izraz, а потоа се извршува наредбата пред која наведената константа е еднаква со добиената вредност на Izraz.

Пример:

```
Case Znak of
        '0'..'9' :Poraka.caption:='Cifra';
        'A'..'Z','a'..'z':poraka.caption:='Bukva';
        '+','-','*','/':Poraka.caption:='Operator'
        Else
        Poraka.caption:='Ostanato'
End;
```

Задача 1: Обои ја формата:



Слика 3.9.

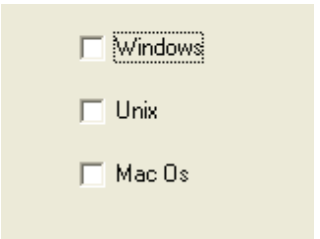
```
procedure TForm1.Button1Click(Sender: TObject);
        var a:integer;
begin
        a:=strtoint(edit1.text);
        if a=1 then form1.Color:=clred;
        if a=2 then form1.Color:=clyellow;
        if a=3 then form1.Color:=clblue;
end;

procedure TForm1.Button2Click(Sender: Tobject);
begin
        form1.Color:=clbtnface;
end;
```

Задача 2: Да се направи програма во Delphi за пресметка на P и L на триаголник откако ќе се провери дали таков триаголник може да постои. Влезни податоци се страните на триаголникот.

3.8. Компоненти за избор
3.8.1. Check Box контрола

Контролата **TCheckBox се** наоѓа на страната Standard. Се користи при решавање задачи кога е потребно да се овозможи вклучување или исклучување на една или повеќе опции. Може да има две состојби: вклучена и исклучена.

Пример: Означете ги оперативните системи кои знаете да ги користите.



Оваа контрола се користи за одбирање на некои вредности кога може да се одберат повеќе различни вредности и ниедна не ја исклучува другата. Секоја вредност посебно може да биде одбрана или не.

Својства:

Checked – својство од булов тип кое одредува дали контролата е селектирана. Можни се две состојби True (селектирана) или False (неселектирана).

Задача1: Да се направи форма во која има лабела, edit кутија, копче за излез и две Checkbox контроли во кои ќе може да се селектира дали внесениот текст е bold или italic.

Слика 3.10.

```
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
        If checkBox1.checked then
        edit1.Font.style:=[fsbold]
        else
        edit1.Font.style:=[]
end;

procedure TForm1.CheckBox2Click(Sender: TObject);
begin
        If checkBox2.checked then
        edit1.Font.style:=[fsitalic]
        else
        edit1.Font.style:=[]
end;
```

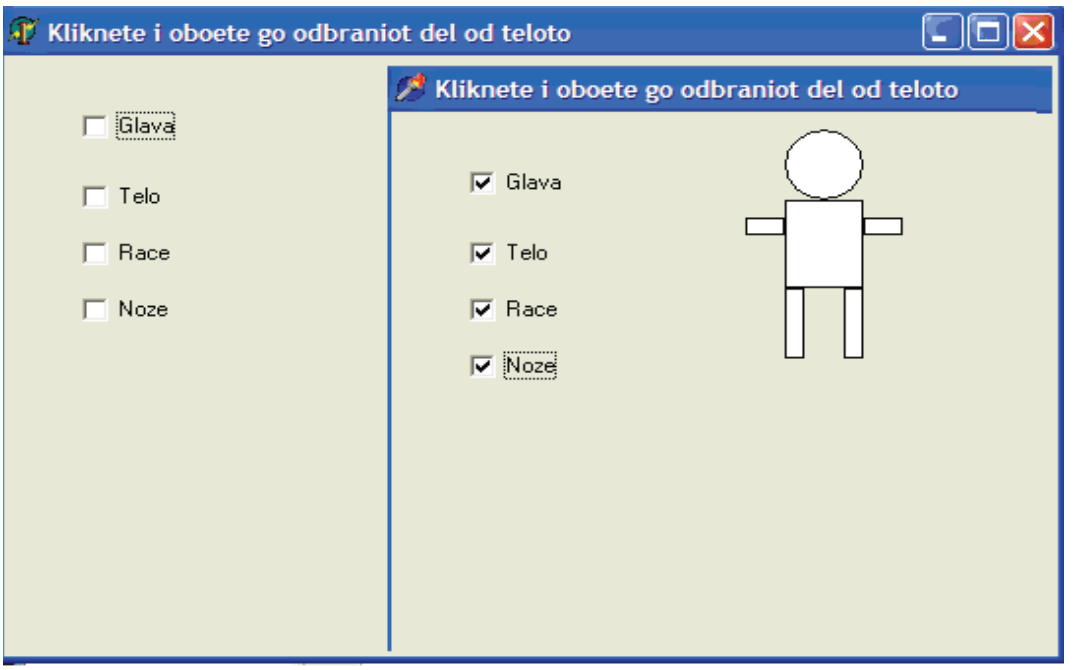
Задача 2: Да се направи код кој ќе ги зема предвид сите комбинации на вклучени и исклучени копчиња.

```
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
          If (checkBox1.checked) and (checkBox2.checked) then
            edit1.Font.style:=[fsbold,fsitalic]
          else
             If checkBox1.checked then
                edit1.Font.style:=[fsbold]
              else
                 If checkBox2.checked then
                     edit1.Font.style:=[fsitalic]
                  else
                     edit1.Font.style:=[]
end; {procedurata se sostoi od 3 vgnezdeni If ciklusi}
```

Истата процедура е и за настанот TForm1.CheckBox2Click.

```
procedure TForm1.CheckBox2Click(Sender: TObject);
begin
          If (checkBox2.checked) and (checkBox1.checked) then
            edit1.Font.style:=[fsbold,fsitalic]
          else
             If checkBox2.checked then
                edit1.Font.style:=[fsitalic]
              else
                 If checkBox1.checked then
                     edit1.Font.style:=[fsbold]
                  else
                     edit1.Font.style:=[]
end; {procedurata se sostoi od 3 vgnezdeni If ciklusi}
```

Задача 3: Да се направи апликација во која во формата има 4 CheckBox контроли кои вклучуваат делови од човечкото тело – глава, тело, раце, нозе.



Слика 3.11.

```
Procedure TForm1.CheckBox1Click(Sender: TObject);
  var t,t1:Tcolor;
  begin
      if checkbox1.Checked then
          form1.canvas.Ellipse(200,10,240,50)
     else
         begin
            t:=form1.Canvas.Brush.Color;
            t1:=form1.Canvas.Pen.Color;
            form1.Canvas.Brush.Color:=form1.color;
            form1.canvas.pen.color:=form1.color
            form1.canvas.ellipse(200,10,240,50);
            form1.Canvas.Brush.Color:=t;
            form1.Canvas.Pen.Color:=t1;
          end;
  end;

Procedure TForm1.CheckBox2Click(Sender: TObject);
  var t,t1:tcolor;
  begin
     if checkbox2.Checked then
          form1.canvas.Rectangle(200,50,240,100)
     else
         begin
            t:=form1.Canvas.Brush.Color;
            t1:=form1.Canvas.Pen.Color;
            form1.Canvas.Brush.Color:=form1.color;
```

```
            form1.canvas.pen.color:=form.color;
            form1.canvas.Rectangle(200,50,240,100);
            form1.Canvas.Brush.Color:=t;
            form1.Canvas.Pen.Color:=t1;
          end;
   end;

Procedure TForm1.CheckBox3Click(Sender: TObject);
  var t,t1:tcolor;
  begin
      if checkbox3.Checked then
        begin
            form1.canvas.Rectangle(180,60,200,70);
            form1.canvas.Rectangle(240,60,260,70);
        end
      else
         begin
            t:=form1.Canvas.Brush.Color;
            t1:=form1.Canvas.Pen.Color;
            form1.Canvas.Brush.Color:=form1.color;
            form1.canvas.pen.color:=form.color
            form1.canvas.Rectangle(180,60,200,70);
            form1.canvas.Rectangle(240,60,260,70);
            form1.Canvas.Brush.Color:=t;
            form1.Canvas.Pen.Color:=t1;
         end;
  end;

procedure TForm1.CheckBox4Click(Sender: TObject);
  var t,t1:tcolor;
  begin
     if checkbox4.Checked then
         begin
           form1.canvas.Rectangle(200,100,210,140);
           form1.canvas.Rectangle(230,100,240,140);
         end
     else
         begin
            t:=form1.Canvas.Brush.Color;
            t1:=form1.Canvas.Pen.Color;
            form1.Canvas.Brush.Color:=form1.color;
            form1.canvas.pen.color:=form.color
            form1.canvas.Rectangle(200,100,210,140);
            form1.canvas.Rectangle(230,100,240,140);
            form1.Canvas.Brush.Color:=t;
            form1.Canvas.Pen.Color:=t1;
          end;
 end;

procedure TForm1.FormPaint(Sender: TObject);
```

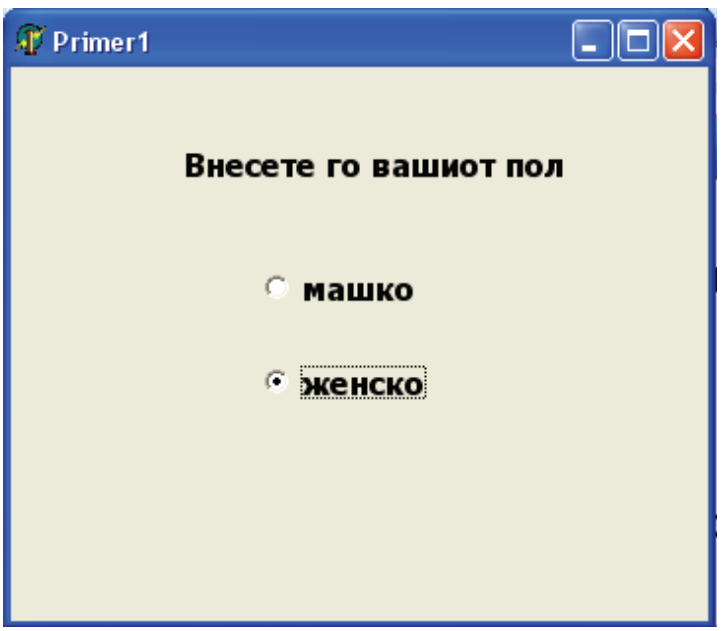
```
begin {Nastan koj nastapuva pred canvas-ot na formata povtorno da se iscrta
odnosno renderira (pomesti ili zgolemi). Znaci koga ke se zgolemi formata (ne
i koga se smaluva), koga se sozdava formata i sl.}
        checkbox1.Checked:=false;
        checkbox2.Checked:=false;
        checkbox3.Checked:=false;
        checkbox4.Checked:=false;
end;
end.
```

3.8.2. RadioButton контрола

RadioButton контролата се користи за избор на една од неколку можни опции. Одбираме една од повеќе понудени можности на **RadioButton** копчиња на подрачјето на само еден објект. Само едно **RadioButton** копче може да биде селектирано. Со вклучување на едно радиокопче се исклучуваат сите останати. Ова копче се користи во групи. На корисникот му овозможува да одбере само една од неколку понудени опции, затоа што изборот на една од нив подразбира автоматско исклучување на сите останати.

За оваа компонента најзначајни се својствата Caption – натпис и Checked – кое ја означува состојбата на радиокопчето: вклучено или исклучено.

Пример: Внесете го вашиот пол.

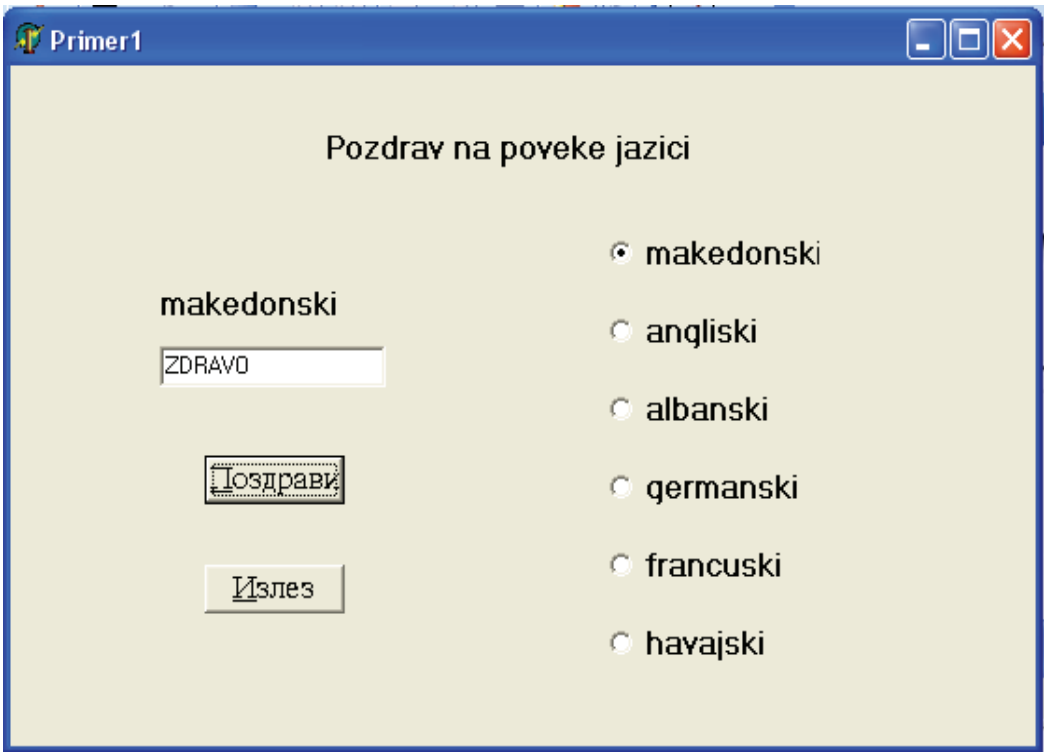


Слика 3.12.

Објектите од тип RadioButton имаат многу сличности со објектите од тип CheckBox.

Задача 1: Да се направи форма во која зборот “Добар ден” или “Здраво” се прикажува на повеќе јазици. Изборот на јазиците се врши со компонентата RadioButton. На формата треба да се постави Edit контрола (со жолта позадина и сини букви) за прикажување на одбраниот поздрав. Над ова поле во компонента Label да се прикаже одбраниот јазик. Изборот на јазиците се врши со

RadioButton , а поздравот да се испишува кога ќе се кликне на командното копче со натпис “Поздрави”.
Изглед на формата:



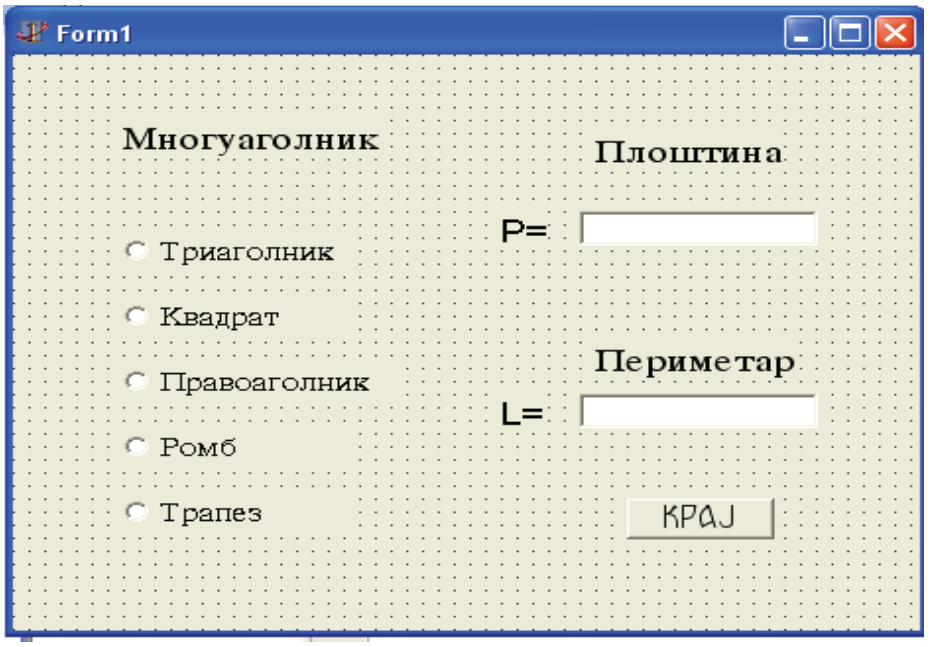
Слика 3.13.

Програмски код:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
if RadioButton1.Checked then
  begin
    Edit1.Text:= 'Zdravo';
    Label2.Caption:= 'Makedonski';
  end;
if RadioButton2.Checked then
  begin
    Edit1.Text:= 'Hello';
    Label2.Caption:= 'Angliski';
  end;
if RadioButton3.Checked then
  begin
    Edit1.Text:= 'Guten Tag';
    Label2.Caption:= 'Germanski';
  end;
if RadioButton4.Checked then
  begin
    Edit1.Text:= 'Bonjour';
    Label2.Caption:= 'Francuski';
  end;
if RadioButton5.Checked then
```

```
  begin
    Edit1.Text:= 'Aloha';
    Label2.Caption:= 'Havajski';
  end;
if RadioButton6.Checked then
  begin
    Edit1.Text:= '???';
    Label2.Caption:= 'Albanski';
  end;

end;

procedure TForm1.Button2Click(Sender: TObject);
begin
application.Terminate;
end;
end.
```

Задача 2:Да се направи апликација која на формата ќе има 5 радиокопчиња, чии натписи се следните геометриски фигури (триаголник, квадрат, правоаголник, ромб и трапез), пет лабели, две Edit кутии и едно копче за крај. Кога ќе се селектира соодветниот RadioButton, во Edit кутиите се прикажуваат формулите за пресметка на P (плоштина) и L(периметар) на таа геометриска фигура.
Упатство: триаголник (P=a*h/2, L=a+b+c), квадрат (P=a*a, L=4*a), правоаголник (P=a*b, L=2*(a+b)), ромб (P=a*h, L=4*a), трапез (P=(a+b)*h/2, L=a+b+c+d),



Слика 3.14

3.8.3 RadioGroup контрола

Радиокопчињата можат да бидат сместени во посебна кутија – RadioGroup. Целта на постоење на RadioGroup е да се поедностави групирањето на радиокопчињата на едно место. Вметнувањето и распоредувањето на радиокопчињата автоматски се регулира со специфичните својства на компонентата RadioGroup.

Својства на RadioGroup се

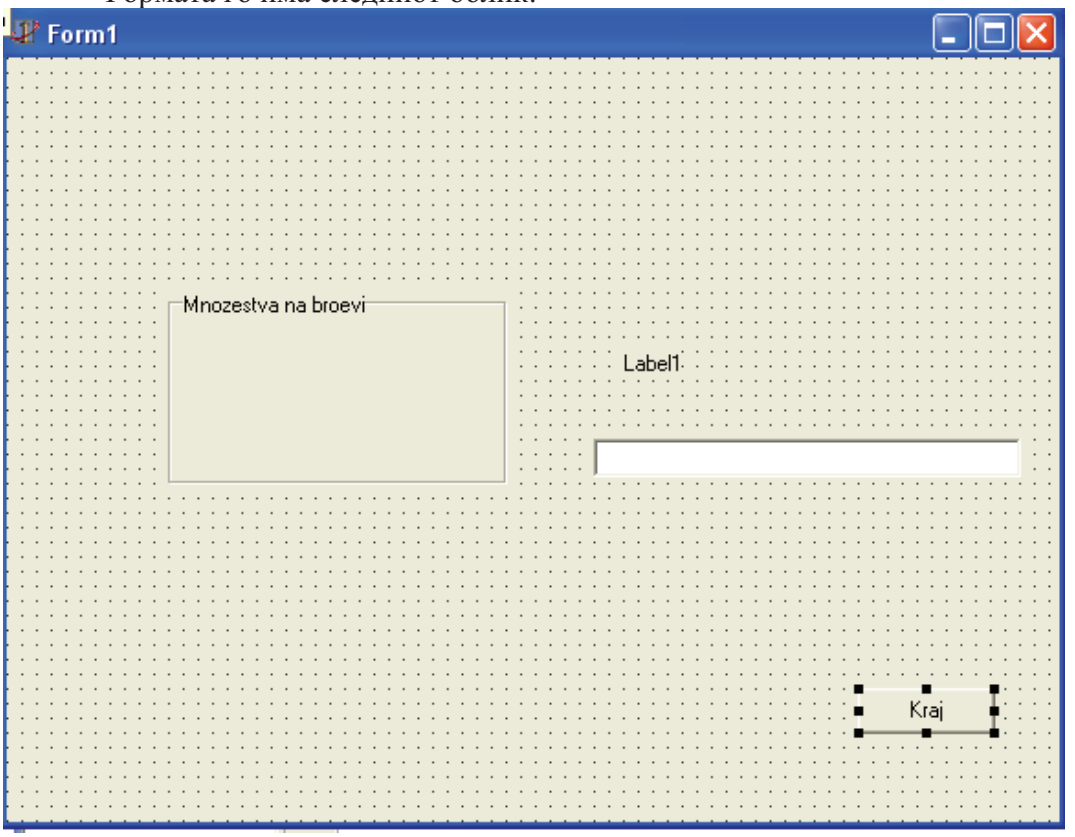
-Columns – ова својство одредува број на колони во кои се распоредени копчињата во рамките на RadioGroup,

-Items – овозможува да се дефинира, чита и менува списокот на натписи и бројот на радиокопчиња кои се наоѓаат во рамките на RadioGroup,

-ItemIndex – го одредува индексот на моментно активниот Radio Button, во рамките на RadioGroup. Првото копче има индекс 0. Ако ни едно копче не е селектирано, индексот е -1.

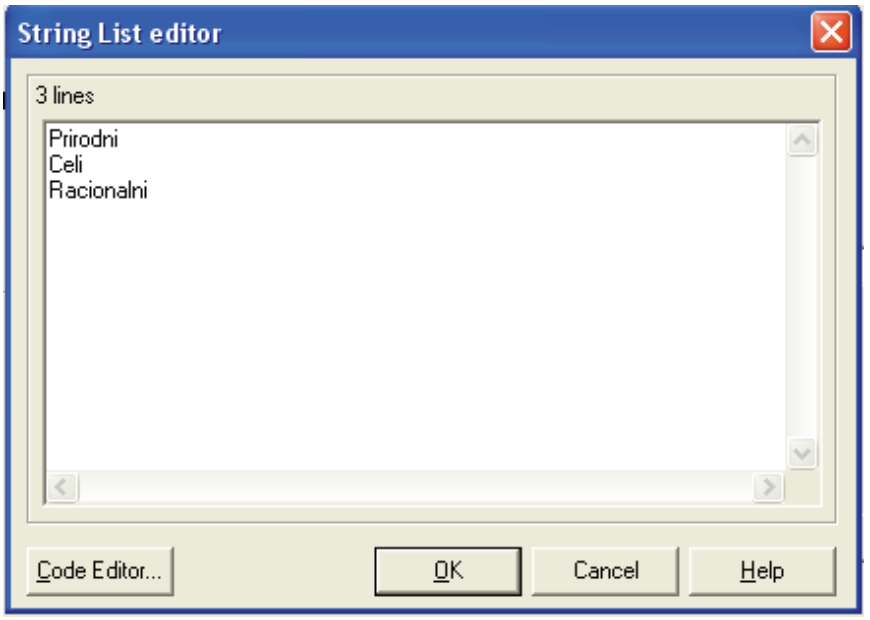
Задача1: Да се креира апликација која во рамките на RadioGroup ќе прикажува множества на броеви и после одбирањето на едно од нив ќе прикажува некои од броевите кои ги карактеризираат.

Формата го има следниот облик:



Слика 3.15.

Поставувањето на радиокопчињата во рамките на заедничката кутија се постигнува со својството Items од Object Inspector . Со кликање на полето со три точки се добива следниот прозорец (слика 3.16.) за внесување на насловите на копчињата:



Слика 3.16.

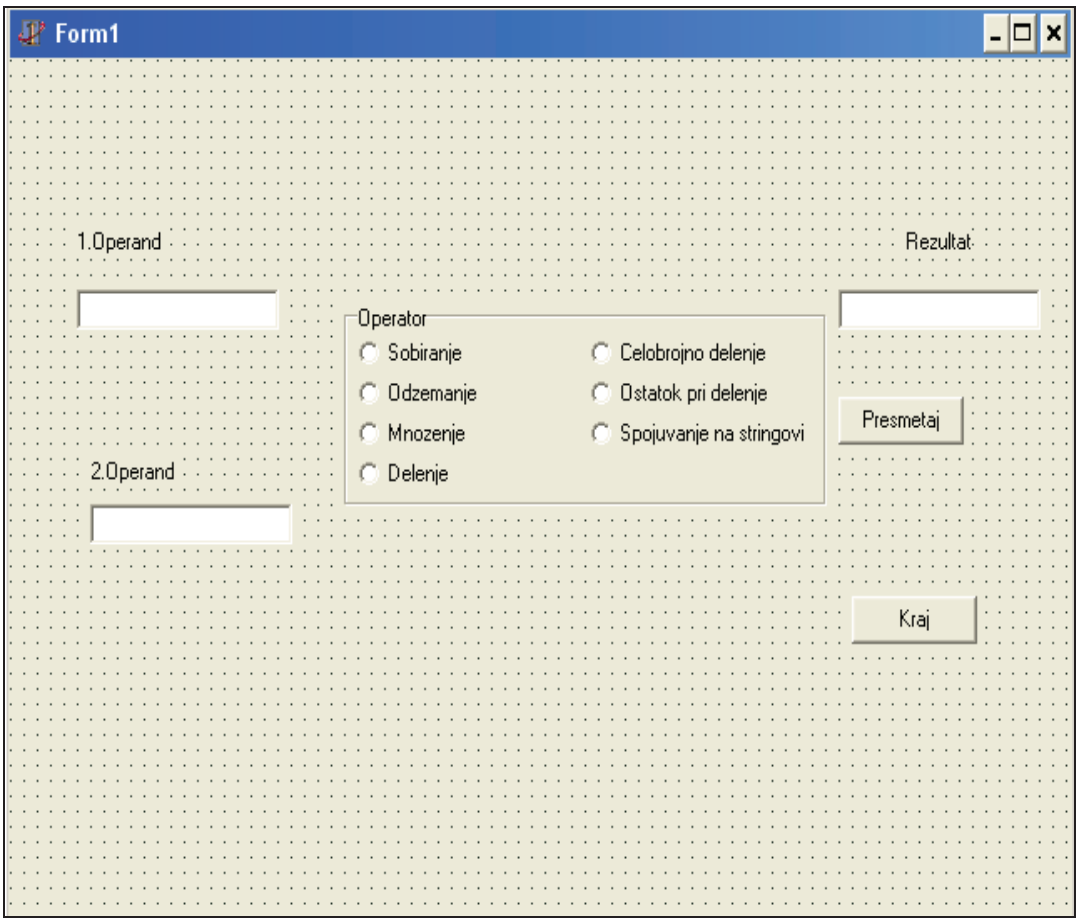
На крај на внесувањето се кликнува на копчето OK. За да бидат копчињата во две колони, својството Columns се поставува на вредност 2.

Програмскиот код е следниот:

```
Procedure TForm1.RadioGroup1Click(Sender: TObject);
begin
   Case RadioGroup1.itemIndex of
        0:Begin
                edit1.text:='1,2,3,4,...';
                label1.caption:='Prirodni';
        end;
        1: Begin
                edit1.text:='...,-3,-2,-1,0,1,2,3,4,...';
                label1.caption:='Celi';
        end;
        2:Begin
                edit1.text:='...,-p/q,...,-1,-1/3,0,1/2,1,...,p/q,...';
                label1.caption:='Racionalni';
        end;
        3:Begin
                edit1.text:='Ne mozat da se napisat vo oblik p/q';
                label1.caption:='Iracionalni';
        end;
        4:Begin
                edit1.text:='Unija na mnozestvoto racionalni i iracionalni';
                label1.caption:='Realni';
        end;
        5:Begin
                edit1.text:='Vo oblik: -2i,i,3i,...';
                label1.caption:='Imaginarni';
        end;
```

```
        6:Begin
                edit1.text:='So oblik a+bi, a-bi';
                label1.caption:='Kompleksni';
        end;
    end; {kraj na Case naredba}
end;
procedure TForm1.Button1Click(Sender: TObject);
        begin
                application.terminate;
        end;

end.
```

Задача2. Да се направи следната апликација .

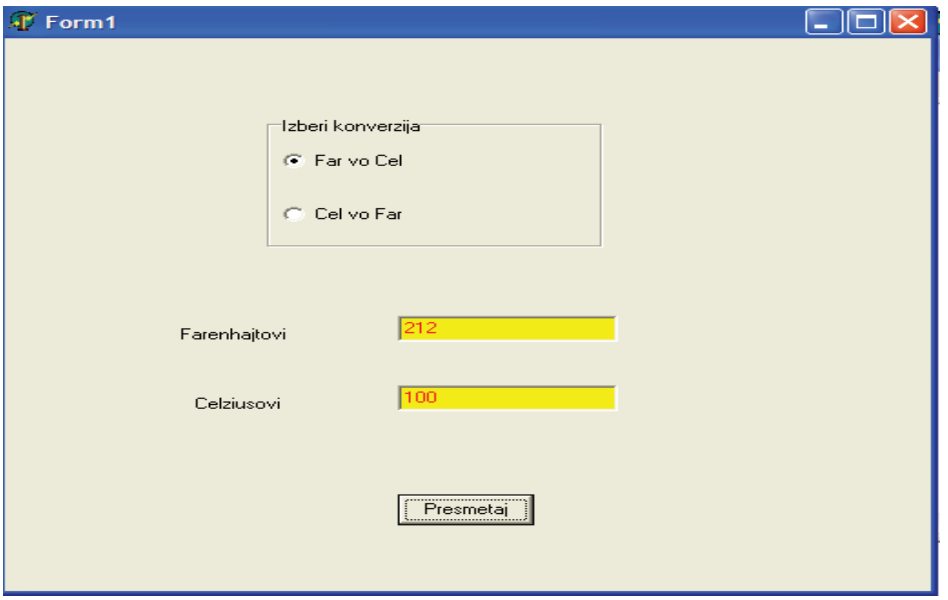


Слика 3.17.

```
procedure TForm1.Button1Click(Sender: TObject);
        var
        x,y,z:real;
        a,b,c,i:integer;
        s:string;
    begin
         Val(edit1.text,x,i);
         Val(edit2.text,y,i);
         Val(edit1.text,a,i);
         Val(edit2.text,b,i);
             Case radiogroup1.ItemIndex of
                  0:Begin
                  z:=x+y; Str(z:8:3,s);edit3.text:=s;
                  end;
                  1:Begin
                  z:=x-y; Str(z:8:3,s);edit3.text:=s;
                  end;
                  2:Begin
                  z:=x*y; Str(z:8:3,s);edit3.text:=s;
                  end;
                  3:Begin
                  z:=x/y; Str(z:8:3,s);edit3.text:=s;
                  end;
                  4:Begin
                  c:=a div b; Str(c,s);edit3.text:=s;
                  end;
                  5: Begin
                  c:=a mod b; Str(c,s);edit3.text:=s;
                  end
                 else
                   edit3.text:=edit1.text+edit2.text;
            end;
       end;
end.
```

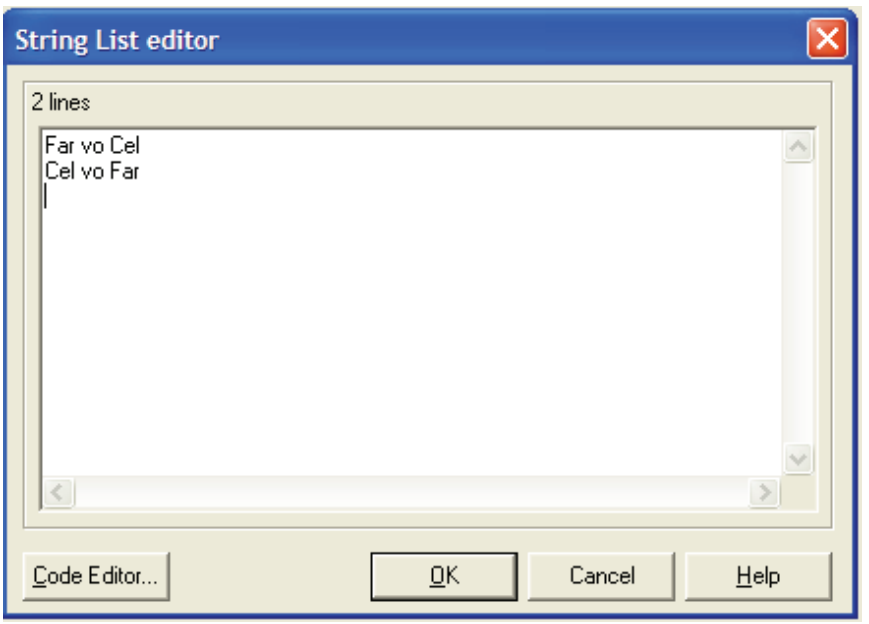
Val (edit1.text,broj,code)
Оваа функција врши претворање на стринг(текст) во број. Индикаторот на претворањето – code укажува на успешноста на претворањето. Тој има вредност 0 ако претворањето е успешно извршено.
(бројот 91вв4 нема да се претвори успешно).
Str (broj, string) врши обратно претворање од број во стринг

Задача 3: Да се направи програма која врши претворање на степени од Фаренхајтови во Целзиусови и обратно. За избор на начинот на претворање се користи компонентата RadioGroup. На формата треба да се постават две Edit компоненти со жолта позадина и црвени бројки за внесување на степените, како и командно копче Пресметај со чија помош ќе се активира пресметката.



Слика 3.18.

Формата содржи компонента RadioGroup која може да содржи повеќе копчиња од кои само едно може да биде одбрано во еден момент. Со кликање на својството Items во Properties се отвора следниот екран:



Слика 3.19.

Во едиторот ги внесуваме насловите на копчињата кои ќе ги користиме .
Главниот дел на кодот на оваа програма е следниот:

```
procedure TForm1.RadioGroup1Click(Sender: TObject);
  begin
      case radiogroup1.itemindex of
```

```
          0:begin
            edit1.setfocus;
            edit1.selectall;
           end;
          1:begin
            edit2.setfocus;
            edit2.selectall;
           end;
       end;
  end;
procedure TForm1.Button1Click(Sender: TObject);
         var s:string;
         n:real;
         gr:integer;
    begin
          case radiogroup1.itemindex of
             0:begin
                   s:=edit1.text;
                   val(s,n,gr);
                   if gr<>0 then
                     begin
                       ShowMessage('Ne se dobro vneseni Farenh');
                       edit1.setfocus;
                        edit1.SelectAll;
                     end
                  else
                       edit2.text:=floattostr((5/9)*(strtofloat(edit1.text)-32));
              end;
            1:begin
                   s:=edit2.text;
                   val(s,n,gr);
                   if gr<>0 then
                     begin
                          ShowMessage('Ne se dobro vneseni Celz');
                          edit2.setfocus;
                          edit2.SelectAll;
                     end
                  else
                       edit1.text:=floattostr(1.8*(strtofloat(edit2.text))+32);
              end;
          end;
    end;
procedure TForm1.FormActivate(Sender: TObject);
begin
         radiogroup1.itemindex:=0 {pocetni uslovi}
end;
end.
```

Задача 4: Да се направи програма за конверзија на валути (евро – денар и обратно).

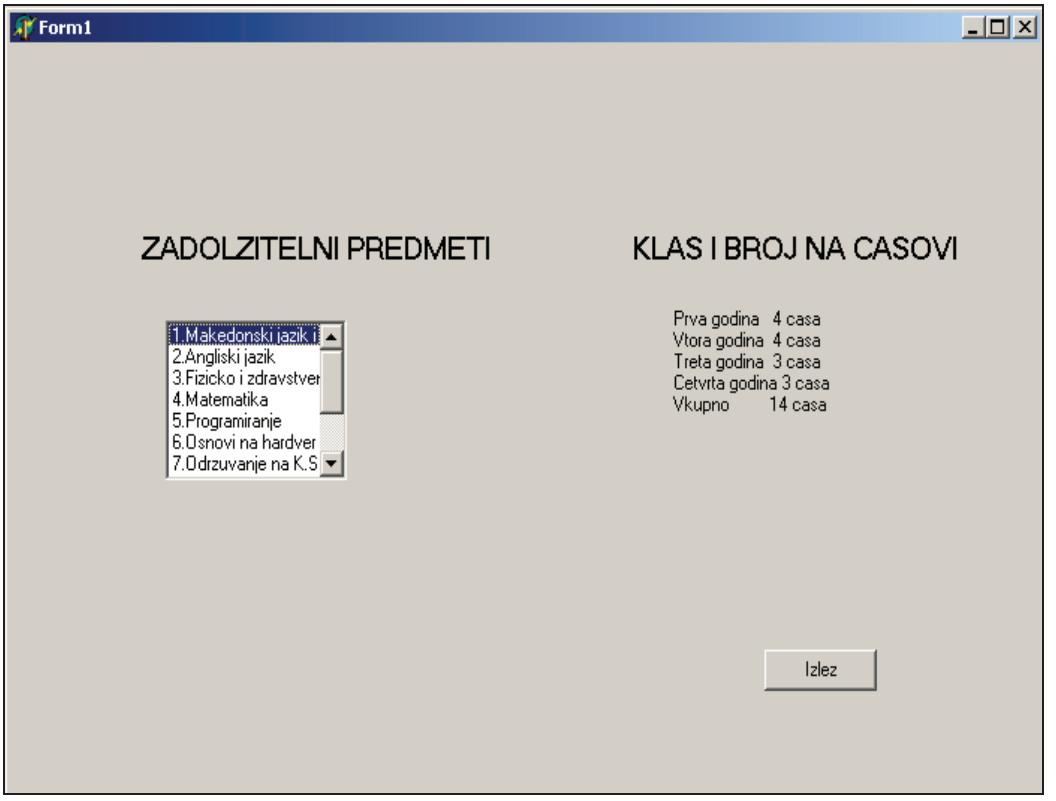
3.8.4.Компонента ListBox

Претставува вообичаен Windows прозорец за листа. Листата содржи список на можности кои корисникот може да ги одбере. Доколку листата содржи повеќе можности отколку што може да се прикажат, во рамката на листата се појавуваат ленти со лизгачи. Оваа компонента овозможува прикажување на содржината во повеќе колони со поставување на својството **Columns** на саканиот број на колони. Кога оваа контрола е во фокусот, корисникот притиска некоја буква од тастатурата и првата опција која започнува со таа буква станува тековна. Ова својство овозможува брзо пронаоѓање на бараните можности.

ListBox најчесто се користи кога податоците кои треба да се прикажат претставуваат список, на пример, на имиња на поголем број луѓе или некакви наслови. Корисникот може да одбере поодделни можности од листата што се регистрира автоматски како соодветно својство. Потоа вредноста на тоа својство може да се користи во програмата за донесување одредени одлуки. Компонентата ListBox се наоѓа на јазичето со стандардни компоненти.

Задача1: На формата да се постават три лабели, една ListBox и едно копче. Третата лабела ги има следните својства: AutoSize=False; WordWrap=True; Caption=’ ‘

Првото својство го укинува автоматското прилагодување на натписот на лабелата спрема големината на текстот. Второто својство овозможува прикажување на текстот во повеќе редови. Заради ова големината на лабелата мора да ја одредиме рачно.



Слика 3.20.

Опциите, односно можностите, ги внесуваме во листата со помош на својството **Items** од Object Inspector .

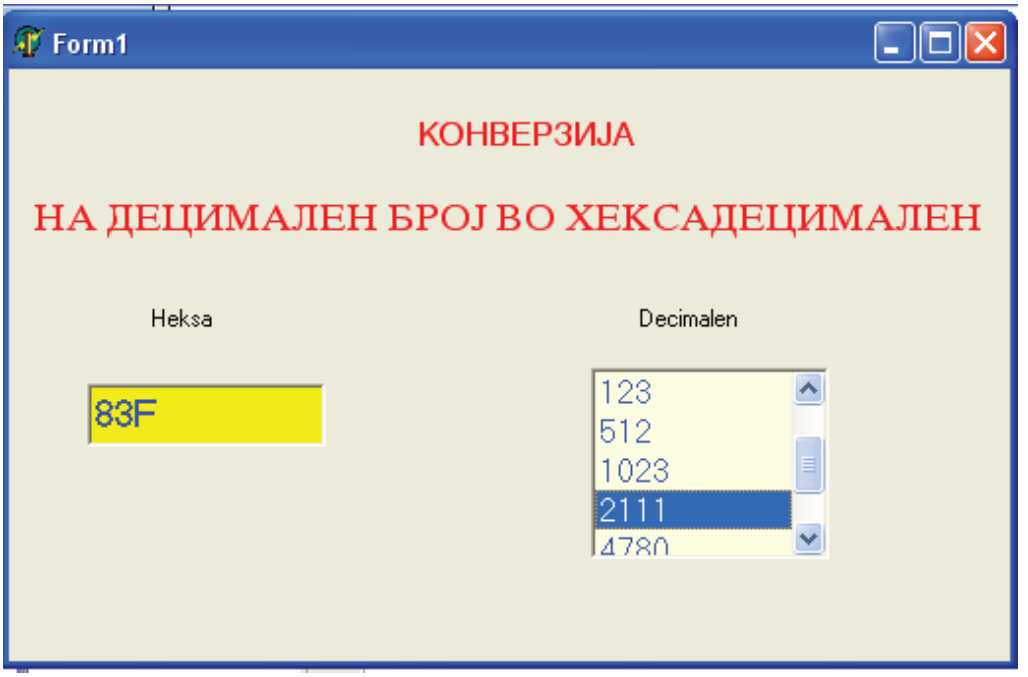
```
procedure TForm1.Button1Click(Sender: TObject);
begin
     application.terminate;
end;

procedure TForm1.ListBox1Click(Sender: TObject);
begin
case listbox1.itemindex of
0:begin
   Label3.caption:='Prva godina   4 casa'+#13+
                'Vtora godina  4 casa'+#13+
                'Treta godina  3 casa'+#13+
                'Cetvrta godina 3 casa'+#13+
                 'Vkupno         14 casa'
   end;
1:  Begin
   Label3.caption:='Prva godina   2 casa'+#13+
                'Vtora godina  2 casa'+#13+
                'Treta godina  2 casa'+#13+
                'Cetvrta godina 2 casa'+#13+
                 'Vkupno         8 casa'
   end;
2:  Begin
   Label3.caption:='Prva godina   2 casa'+#13+
                'Vtora godina  2 casa'+#13+
                'Treta godina  2 casa'+#13+
                'Cetvrta godina 2 casa'+#13+
                 'Vkupno         8 casa'
   end;
3: Begin
   Label3.caption:='Prva godina   4 casa'+#13+
                'Vtora godina  4 casa'+#13+
                'Treta godina  4 casa'+#13+
                'Cetvrta godina 4 casa'+#13+
                 'Vkupno         16 casa'
   end;
 4:Begin
   Label3.caption:='Vtora godina  2 casa'+#13+
                'Treta godina  2 casa'+#13+
                'Cetvrta godina 2 casa'+#13+
                 'Vkupno         6 casa'
   end;
 5:Begin
   Label3.caption:='Treta godina  2 casa'+#13+
                'Cetvrta godina 2 casa'+#13+
                 'Vkupno         4casa'
   end;
6: Begin
   Label3.caption:='Treta godina  2 casa'+#13+
                'Cetvrta godina 2 casa'+#13+
```

```
                'Vkupno        4 casa'
   end;
7:  Begin
   Label3.caption:='Treta godina  2 casa'+#13+
                'Cetvrta godina 2 casa'+#13+
                'Vkupno        4 casa'
   end;
8:  Begin
   Label3.caption:='Treta godina  2 casa'+#13+
                'Cetvrta godina 2 casa'+#13+
                'Vkupno        4 casa'
   end;
9:  Begin
   Label3.caption:='Cetvrta godina 3 casa'+#13+
                'Vkupno        3 casa'
   end;
10:Begin
   Label3.caption:='Vtora godina  4 casa'+#13+
                'Treta godina  5 casa'+#13+
                'Cetvrta godina 5 casa'+#13+
                'Vkupno        14 casa'
   end;
   end;
end;

end.
```

Во програмскиот код се забележуваат некои делови од стрингот кои до сега не се користени: # 13 . Ова е налог прикажувањето на текстот во лабелата после овој знак да продолжи во нов ред. Бројот 13 е **ASCII** КОД на знакот за премин во нов ред. Инаку, преминот во нов ред се реализира со притискање на копчето Enter на тастатурата, со што се генерира точно командниот код број 13. Понатаму се забележува дека долгите стрингови се разбиваат на пократки, а потоа со помош на операторот # (конкатенација) се прикажуваат во рамката на една наредба.

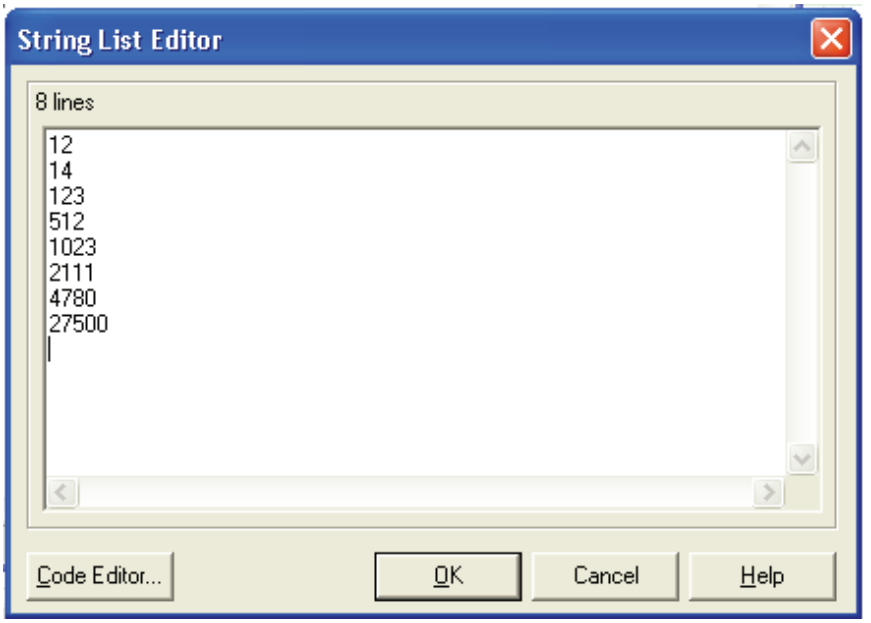
Задача 2: Конверзија на децимален број во хексадецимален со ListBox компонента.

Да се напише програма со која ќе се претвораат следните децимални броеви во heks-a: 12,14,123,512,1023,2111,4780 i 27500. Дадените броеви да се запишат во ListBox компонента (боја на позадината clCream, а бојата на броевите сина). На формата да се дефинира Edit поле (жолта позадина, со сини бројки) за приказ на добиениот хексадецимален број. Над ова поле во компонента Label напиши Heksa, а над ListBox компонентата напиши Decimalen.



Слика 3.21.

Формата ја користи новата компонента ListBox која содржи низа опции и овозможува да се избере една. Со својството Items се отвора прозорец String List Editor со чија помош ги запишуваме членовите на листата – Слика 3.22..



Слика 3.22.

Програмскиот код е следниот:

```
procedure TForm1.ListBox1Click(Sender: TObject);
Var
  i:integer;
begin
     i:=ListBox1.ItemIndex;
```

```
        Edit1.Text :=IntToHex(StrToInt(ListBox1.Items[i]),2);
    end;
end.
```

Во процедурата ListBoxClick (која се повикува со клик на некој број во листата) одредуваме реден број на селектираниот број (i:=ListBox1.ItemIndex;), а потоа ја прикажуваме неговата хексадецимална презентација во полето Edit1. За тоа ја користиме функцијата IntToHex која го претвора целиот број во хексадецимален, како и функцијата StrToInt која одбраниот број од ListBox (кој е од тип String) го претвора во цел број. Бројот 2 во функцијата ни кажува дека хексадецималниот број ќе содржи минимум 2 цифри.

function IntToHex(Value: Integer; Digits: Integer): string;

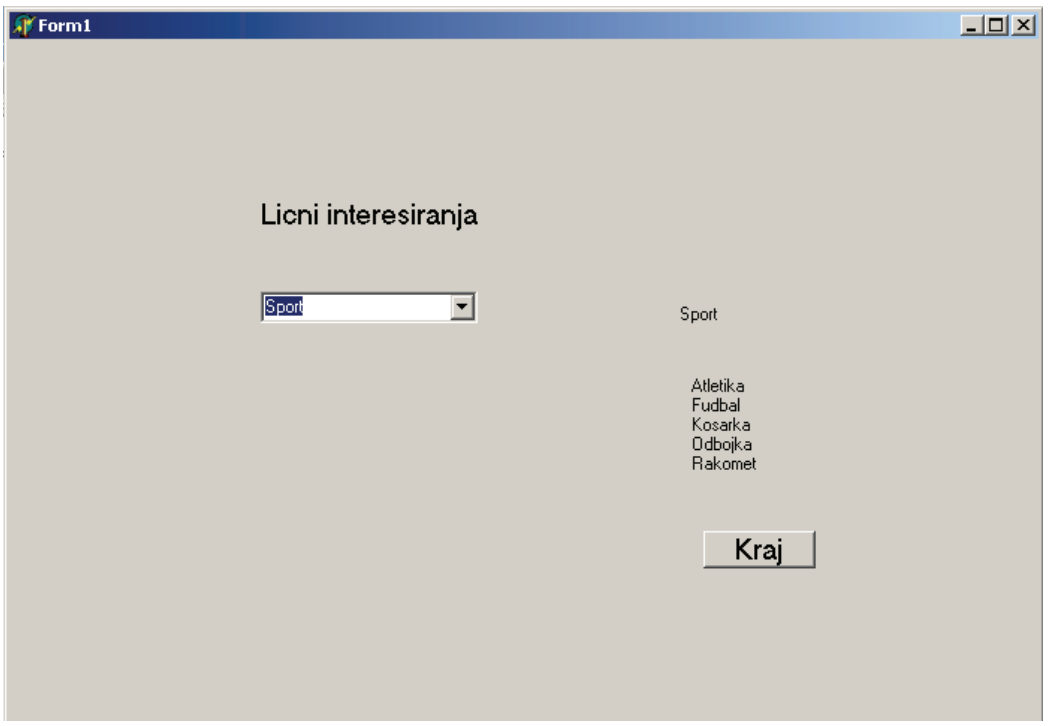
3.8.5 Компонента ComboBox

ComboBox е компонента која ги комбинира можностите за внесување на текст на edit бокс и избор на една од понудените можности на ListBox. Оваа компонента се состои од поле за внесување слично на edit бокс кое на десната страна има копче со стрелка надолу и листа со опции слична на ListBox, која се покажува кога ќе се кликне на малата стрелка надолу. Корисникот може да одбере една од предложените вредности од листата , по што одбраната вредност се пренесува во edit полето, или во edit полето да се внесе сосем нова вредност.

Наједноставниот ComboBox нема можности за додавање на нови опции, туку само може да се одберат понудените вредности.

Пример: Во ComboBox да се напишат областите на интересирање, а во лабелата потоа поблиску се објаснуваат.

Го користиме својството Items, како и кај ListBox за да ги внесеме опциите на ComboBox: Knizevnost, Tehnika, Muzika, Sport, Patuvanja, Rekreacija, Odmor и Kosmos.



Слика 3.23.

```
procedure TForm1.ComboBox1Change(Sender: TObject);
begin
  Case combobox1.ItemIndex of
  0:Begin
    Label2.caption:=combobox1.items[0];
    label3.caption:='Poezija'+#13+
    'Proza'+#13+'Detska'+#13+'Klasicna'+#13+'Sovremena';
    end;
  1:Begin
    Label2.caption:=combobox1.items[1];
    label3.caption:='Radio'+#13+
    'Video'+#13+'Elektro'+#13+'Avio'+#13+'Soobrakajna';
    end;
  2:Begin
    Label2.caption:=combobox1.items[2];
    label3.caption:='Narodna'+#13+
    'Zabavna'+#13+'Detska'+#13+'Dzez'+#13+'Opera';
    end;
  3: Begin
    Label2.caption:=combobox1.items[3];
    label3.caption:='Atletika'+#13+
    'Fudbal'+#13+'Kosarka'+#13+'Odbojka'+#13+'Rakomet';
    end;
  4:Begin
    Label2.caption:=combobox1.items[4];
    label3.caption:='So voz'+#13+
    'So avtobus'+#13+'So avion'+#13+'So brod'+#13+'Peski';
    end;
   5:Begin
    Label2.caption:=combobox1.items[5];
    label3.caption:='Plivanje'+#13+
    'Setanje'+#13+'Planinarenje'+#13+'Lov'+#13+'Ribolov';
    end;
   6:Begin
    Label2.caption:=combobox1.items[6];
    label3.caption:='Radio'+#13+
    'Televizija'+#13+'Citanje'+#13+'Mectaenje'+#13+'Razgovor';
    end;
   7:Begin
    Label2.caption:=combobox1.items[7];
    label3.caption:='Planeti'+#13+
    'Sonce'+#13+'Nabljuduvanje'+#13+'Patuvanje'+#13+'Let';
    end;
    end;
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
   application.terminate;
end;
end.
```

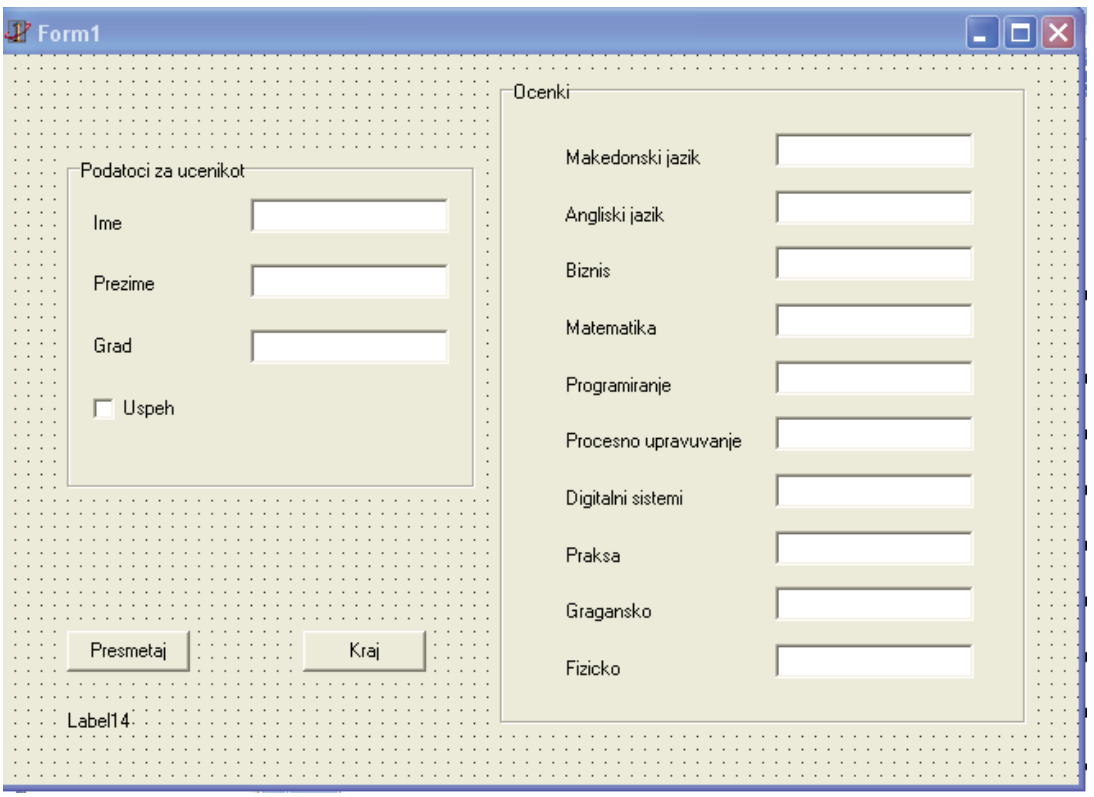
3.9 Контејнерски компоненти

Овие компоненти овозможуваат групирање на поединечните компоненти во одредени целини. Еден тип контејнерска компонента е RadioGroup.

3.9.1.Компонента GroupBox

Оваа компонента нема некое посебно значење независно од останатите компоненти. Нејзиното значење е да биде контејнер (кутија) за група компоненти кои се сместуваат во неа. Ова групирање нема функционално значење , туку треба визуелно да издвои некои компоненти од останатите. Оваа компонента се користи кога треба да се работи со податоци кои логички претставуваат одредена целина. Сите компоненти кои ќе се постават на GroupBox му припаѓаат нему.

Задача 1: Да се направи апликација која ги содржи основните податоци за ученикот и неговите оценки. Да се пресмета средниот успех на ученикот. На сликата има две логички целини на податоци: едната се податоци за ученикот, а другата се оценките.



Слика 3.24.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
        application.terminate;
end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
    Var a,b,i,j:integer;
    c:real;
    s:string;
begin
     i:=0; b:=0;
       if length(ime.text)>1 then inc(i);
       if length(prezime.text)>1 then inc(i);
       if length(grad.text)>1 then inc(i);
           val(prv.Text,a,j);
           if (j=0) and ((a>1) and (a<6)) then begin b:=b+a;inc(i);end;
           val(vtor.Text,a,j);
           if (j=0) and ((a>1) and (a<6)) then begin b:=b+a;inc(i);end;
            val(tret.Text,a,j);
           if (j=0) and ((a>1) and (a<6)) then begin b:=b+a;inc(i);end;
            val(cetvrt.Text,a,j);
           if (j=0) and ((a>1) and (a<6)) then begin b:=b+a;inc(i);end;
             val(petti.Text,a,j);
           if (j=0) and ((a>1) and (a<6)) then begin b:=b+a;inc(i);end;
            val(sesti.Text,a,j);
           if (j=0) and ((a>1) and (a<6)) then begin b:=b+a;inc(i);end;
            val(sedmi.Text,a,j);
           if (j=0) and ((a>1) and (a<6)) then begin b:=b+a;inc(i);end;
             val(osmi.Text,a,j);
           if (j=0) and ((a>1) and (a<6)) then begin b:=b+a;inc(i);end;
             val(devetti.Text,a,j);
           if (j=0) and ((a>1) and (a<6)) then begin b:=b+a;inc(i);end;
              val(desetti.Text,a,j);
           if (j=0) and ((a>1) and (a<6)) then begin b:=b+a;inc(i);end;
            c:=b/(i-3);
            str(c:4:2,s);
            if (i=13) and (uspeh.checked=true) then
            label14.caption:='Sreden uspeh na ucenikot e:'+s
            else label14.caption:='Ne se dobri podatocite - proveri!'
  end;
end.
```

Функцијата Length ја одредува должината на аргументот кој е знаковен податок. Функцијата Inc ја зголемува тековната вредност на аргументот за 1. Со програмата е предвидено да се внесат сите податоци, во спротивно се издава соодветна порака дека тоа не е направено. Ако некоја оценка е неважечка, односно не е меѓу 2 и 5 , нема да се пресмета среден успех.

3.9.2.Panel

Панелот е контејнерска компонента, чија основна намена е да ги групира другите компоненти на својата површина. Својствата кои ги поседува плочата, овозможуваат таа да се користи за добивање естетски објекти на формата.

3.9.3. Bevel (рамка)

Оваа компонента овозможува исцртување на линии, рамки или плочи со ефект на тридимензионалност.

3.10. Програмски модули

Модулите овозможуваат сместување на изворниот текст на целата програма во повеќе датотеки. Секој модул може да содржи произволен број потпрограми и се сместува во посебна датотека.

3.10.1Општ облик на програмскиот модул

Заглавие на модулот – наслов (Unit)
Interface
    Декларативни делови
    Заглавие на потпрограмата
Implementation
    Декларативни делови
    Дефиниција на потпрограмите
Initialization (почетни вредности)
    наредби
finalization
    наредби
    end.

Заглавието го има следниот облик:

Unit ImeNaModulot;

ImeNaModulot – име на датотеката во која се наоѓа тој модул.

Останатиот дел од програмскиот модул се дели на четири делови:

а) Дел за спрега (interface) – содржи информации за врските со другите модули

Uses Modul1,Modul2…

се наведуваат имиња на потпрограмите, односно процедурите кои ќе се користат во дадениот модул;

б) Дел за реализација (implementation) – содржи дефиниции кои не треба да се достапни за другите модули. Во овој дел:

-се дефинираат идентификатори

-се дефинираат потпрограми кои припаѓаат на дадениот модул

(се дефинираат сите програми чии заглавија се дадени во делот за спрега);

в) Дел за иницијализација (Initialization) –ги содржи извршните наредби со кои се поставуваат почетни вредности на некои променливи од модулот пред да почне да се извршува главната програма.

Овој дел не е задолжителен и може да не постои.

г) Дел за финализација (finalization) – содржи извршни наредби за завршните дејствија и не е задолжителен.

Завршниот “end.” мора да постои.

3.10.2.Опсег на идентификаторот

Опсег на идентификаторот е дел од изворната програма во која тој идентификатор може да се користи. Опсегот е од местото на воведување на идентификаторот до крајот на програмата, потпрограмата или модулот.

Идентификаторите воведени во главната програма се **глобални** и можат да се користат во сите потпрограми дефинирани во таа програма.

Во потпрограмите се дефинираат идентификатори кои се наречени **локални** и важат само во таа потпрограма.

Игра со копче (I – верзија)

Задача: Да се направи игра “Фати го копчето”. Потребно е копчето да се движи во прозорецот на формата во сите четири насоки. Кога корисникот ќе го фати

копчето, апликацијата завршува. Да се воведат бодови од 55000, па надолу, во зависност од тоа колку трае играта.

Се отвора нова апликација кај која во формата се поставува објект Button Програмирање на настани:

За да се реши хоризонталното движење, се менува својството Button1.Left, а за вертикално движење се менува својството Button1.Top. Движење десно и долу се постигнува со додавање на некоја вредност на Left и Top, додека за лево и горе се одзема некоја вредност од Left и Top. Ова практично поедноставно се реализира со дефинирање одредена константа на која и се менува предзнакот.

Колкава е вредноста за која ќе се помести копчето, се одредува на ниво на цела програма. Таа вредност треба да биде достапна за сите потпрограми, па затоа е потребно да се пријават **глобалните варијабли Sx и Sy.**

Постапката за изготвување на програмата се состои од четири чекори:

- Се пријавуваат глобалните варијабли Sx, Sy;
- Им се задаваат почетни вредности;
- Се програмира поместувањето и одбивањето на настанот Form1.Click;
- Завршувањето се програмира на настанот Button1.

1. Глобални варијабли се пријавуваат во прозорецот Unit во делот Var

```
Var
Form1: TForm;
Sx,Sy:Integer;
```

2. Почетни вредности се задаваат на настанот FormActivate, тој настан се јавува во моментот кога формата е прикажана, а процедурата е следната:

```
procedure TForm1.FormActivate( );
begin
Sx:=3;
Sy:=3;
end;
```

3. Програмирање на поместувањето и одбивањето се сместува во настанот Form1.Form Click. Поместување се случува ако кликаме во формата.

```
procedure TForm1.FormClick( );
Begin
  Button1.Left:=Button1.Left+Sx;
  Button1.Top:=Button1.Top+Sy;
 if(Button1.Left<0)or((Button1.Left+Button1.Width+10)>form1.Width)
       then
       Sx:=-Sx;
 if(Button1.Top<0)or((Button1.Top+Button1.Height+30)>form1.Height)
       then
       Sy:=-Sy;
end;
```

4. Крајот на работата се програмира на копчето Button1:

```
procedure TForm1.Button1Click(      );
begin
 application.terminate;
end;
```

Игра со копче (II – верзија)

Во оваа верзија копчето само ќе се движи без потреба од кликање во прозорецот. Исто така ќе се додадат и бодовите. На претходната верзија се додава Timer компонента и две лабели.

Својства:

Timer1.interval 10

Label1.Caption Score:(Бодови)

Програмирање на настаните

За да се овозможи движење на копчето без кликање на прозорецот, ќе го користиме објектот Timer. Со тоа добиваме временски циклус кој ќе го движи копчето. Наредбите од настанот **Form1.Click** од претходната програма се преместуваат во настанот **Timer1.Timer**.

```
Procedure TForm1.Timer1Timer( )
Begin
    Button1.Left:=Button1.Left + sx;
    Button1.Top:= Button1.Top +sy;
    if(Button1.Left<0)or((Button1.Left+Button1.Width+10)>form1.Width)
    then
        Sx:=-Sx;
    if(Button1.Top<0)or((Button1.Top+Button1.Height+30)>form1.Height)
    then
        Sy:=-Sy;
    Score:=Score – 1;
    Label2.Caption:=IntToStr(Score);
End;
```

Бодовите ќе се намалуваат на секои 10[ms] што зависи од Interval – от на Timer - от, а во **label2** ќе се прикажуваат бодовите.

Променливата Score треба да е глобална и да се допише во делот за пријавување на глобални варијабли.

Иницијализација на глобалната варијабла score (почетна вредност) се доделува во процедурата FormActivate.

Ако играчот го промаши копчето може да се одземат 3 бода. Ова се програмира во настанот FormClick.

```
Procedure Form1.Form Click( )
Begin
    Score:=Score – 3;
End;
```

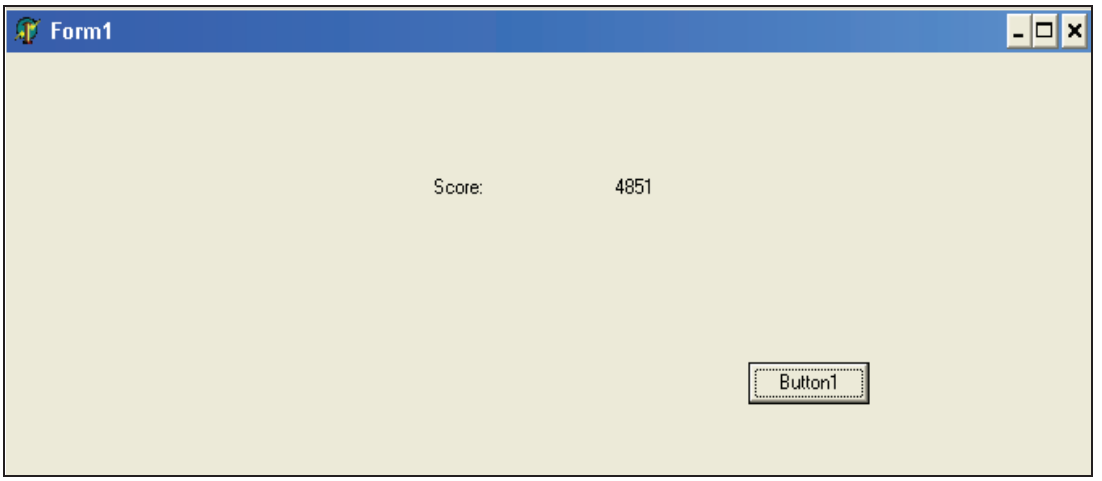
Крајот на апликацијата останува ист.

Игра со копче (III – верзија)

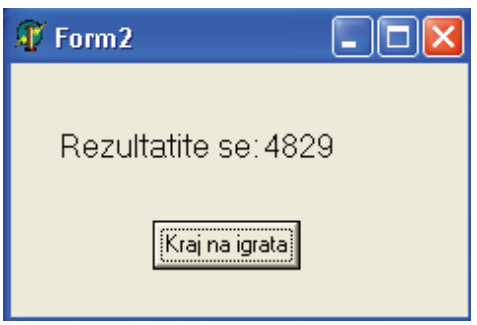
Играта може да се подобри, така што кога ќе се кликне на Button1 ќе запре броењето на бодовите. Тогаш тие може да се гледаат. Ова се постигнува со додавање на нов прозорец Form2, кој ќе се активира со кликање на Form1.Button1. Втората форма мора да чита бодови од првата форма.

Изработка на формата

Додаваме нова форма. На неа поставуваме Label1 со натпис “Резултатите од играта се:” , Label2 која ќе прикажува крајни бодови и Button1 со натпис “Крај на играта”.



Слика 3.25.



Слика 3.26.

Програмирање на настаните

Во Form1 го менуваме настанот Button1Click.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
        timer1.interval:=0;   {zapiranje na tajmerot a so toa i odbivanjeto na
        bodovi}
        form1.visible:=false;  {se skriva prvata forma}
        form2.show;  {se prikazuva vtorata forma}
end;
var
         Form1: TForm1;
         sx,sy:integer;
         score:integer;

implementation
        uses unit2;       {se koristi unit2, istoto se pravi vo Unit2 za Unit1}

{$R *.DFM}
```

Прикажувањето на бодови е најсоодветно во моментот на прикажување на прозорецот на Form2:

```
procedure TForm2.FormActivate(Sender: TObject);
begin
```

```
        label2.caption:=form1.label2.caption;
end;

procedure TForm1.FormActivate(Sender: TObject);
begin
        sx:=3;
        sy:=3;
        score:=5000;
end;

procedure TForm1.FormClick(Sender: TObject);
begin
        score:=score-1;
end;


procedure TForm1.Timer1Timer(Sender: TObject);
begin
        button1.left:=button1.left+sx;
        button1.top:=button1.top+sy;
        if (button1.left<0) or ((button1.left+button1.width+10)>form1.Width)
        then
        sx:=-sx;
        if (button1.top<0) or ((button1.top+button1.height+10)>form1.height)
        then
        sy:=-sy;
        score:=score-1;
        label2.caption:=inttostr(score);
end;
```

Вежба : Да се направи програма калкулатор со употреба на глобални варијабли.

3.11.Циклични структури

Циклуси се управувачки структури кои овозможуваат извршувањето на некоја наредба да се повторува.

3.11.1.WHILE-DO naredba

Оваа наредба се употребува за реализација на структурата *повторување со излез на почеток.*

Синтаксата на наредбата е :

While uslov do
        naredba

Значењето на наредбата е:

Додека **(while)** е исполнет условот, извршувај **(do)** наредба.

Ако има повеќе наредби (блок), тие се стават меѓу BEGIN и END.

Пр:
```
While uslov do
    Begin
     naredba1;
```

```
    naredba2;
      ….
    naredba N;
   End;
```

Зборовите WHILE и Do се резервирани зборови. Услов е некој израз кој се пресметува пред секое извршување на наредбата или блокот на наредби. Вредноста на услов може да биде true или false. Ако е true се извршуваат наредбите по Do, а ако е false не се извршуваат наредбите , туку се прескокнуваат.

Пример:

```
While N<=10 do
 Begin
    Write(N:3);
    N:=N+1;
 End;
```

Може де се случи наредбите на циклусот ниеднаш да не се извршат.

3.11.2.REPEAT UNTIL наредба

Оваа наредба се употребува за реализација на структурата *повторување со излез на крај.*

Синтаксата на наредбата е :

```
Repeat
   naredbi
Until uslov;
```

Значењето на наредбата е:

Повторувај (*Repeat*) го извршувањето на наредбите до (*Until*) исполнувањето на условот.

Условот може да биде израз или релација и може да има вредност на една од логичките константи True или False. Со извршување на наредбите по Repeat, се менуваат променливите од условот во циклусот, со што може да се промени и вредноста на условот на крајот на циклусот.

По секое извршување на наредбите, се испитува дали е исполнет условот. Ако условот не е исполнет , се повторува извршувањето на наредбите, а ако е исполнет, се продолжува со наредбата по Until. Значи, повторувањето завршува кога *uslov* ќе добие вредност True.

Наредбите што се извршуваат со Repeat не се стават во блок меѓу Begin и End.

Пример:

```
K:=0;
   Repeat
         K:=K+1;
         S:=S+k;
   Until K>10;
```

3.11.3 FOR naredba

3.11.3.1. FOR-TO-DO (За зголемувај до)

Оваа наредба се употребува за реализација на структурата *повторување со броење на циклусите.* Повторувањата се бројат со посебен бројач, за кој се задаваат почетната и крајната вредност на бројачот.

Синтаксата на наредбата е :

For *Promenliva:=pocetna_vrednost* **to** *krajna vrednost* **do**
naredba;

Значењето на наредбата е:

За вредности на променливата од некоја почетна вредност, зголемувајќи ја по 1 до некоја крајна вредност, да се изврши следната наредба.

Пример1:

```
For I:=1 to 5 do
        Writeln(I);
```

Пример2:

```
For Bukva:='A' to 'Z' do
        Write(Bukva);
```

- Променливата (во примерите I и Bukva) претставува бројач на циклусите и таа мора да биде од линеарно подреден тип, а не смее да биде реална променлива.
- Почетната и крајната вредност на променливата можат да бидат константи или изрази, меѓутоа мора да бидат од ист тип со неа.
- Овие вредности не смеат да се менуваат во наредбата по резервираниот збор DO.
- На променливата автоматски и се доделува следната вредност за следниот циклус, во кој се извршува наредбата.
- Ако има повеќе наредби по DO, тогаш тие се стават во блок меѓу Begin и End.

Пример:

```
For  Promenliva:=pocetna_vrednost to krajna vrednost do
        Begin
                naredba;
                  …
                naredba;
        End;
```

3.11.3.2. FOR-DOWNTO-DO (За намалувај до)

Синтаксата на наредбата е :

For *Promenliva:=pocetna_vrednost* **downto** *krajna vrednost* **do**
naredba;

Значењето на наредбата е:

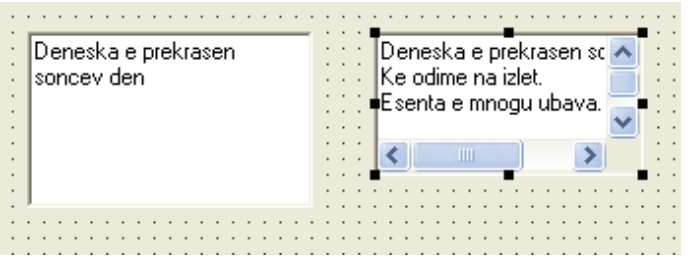
За вредности на променливата од некоја почетна вредност, намалувајќи ја по 1 до некоја крајна вредност, да се изврши следната наредба.

Пример:

```
For I:=15 downto 5 do
        Writeln(I);
```

3.12.Компоненти за работа со низи

3.12.1. Компонентата (Memo) се наоѓа на страната Standard во палетата на компоненти. Таа е слична на Edit box. Основната разлика е во тоа што во Memo компонентата може да се прикажува текст во повеќе редови.



Слика 3.27.

Оваа рамка може да биде без или со лизгач. Текстот може да се порамнува кон левиот или десниот раб и да се центрира.

Својства:

Lines (редови) од тип Tstrings е главно својство кое го содржи текстот на компонентата Memo,
Allignement – порамнување по лев, десен раб или центрирање,
Counts – го содржи вкупниот број на впишани линии текст,
Scrollbars – одредува кои лизгачи постојат (хоризонтален, вертикален или двата)–
Names – својство кое ни овозможува доделување и читање имиња на стринговите. На секој стринг му се пристапува со помош на неговиот индекс – реден број. Првиот е со индекс 0, вториот 1 итн.
Names[Index:Integer]:String;
Text – сиот текст од линиите заедно,
Values – добива или поставува вредност на стрингот во низата,
Function Add(const:string):Integer; - додава стринг на крајот на листата,

Пример 1: Што работи следната процедура?

```
Procedure Tform1.button1click();
Var I: integer;
Begin
        Memo1.lines.clear;
        For i:=0 to 5 do
                Memo1.lines.add( 'Ova e pominuvanje '+ inttostr(i));
        Memo1.lines.add(' ');
End;
```

Пример2: Да се направи програма за одредување на најголем член на низата.

Слика 3.28

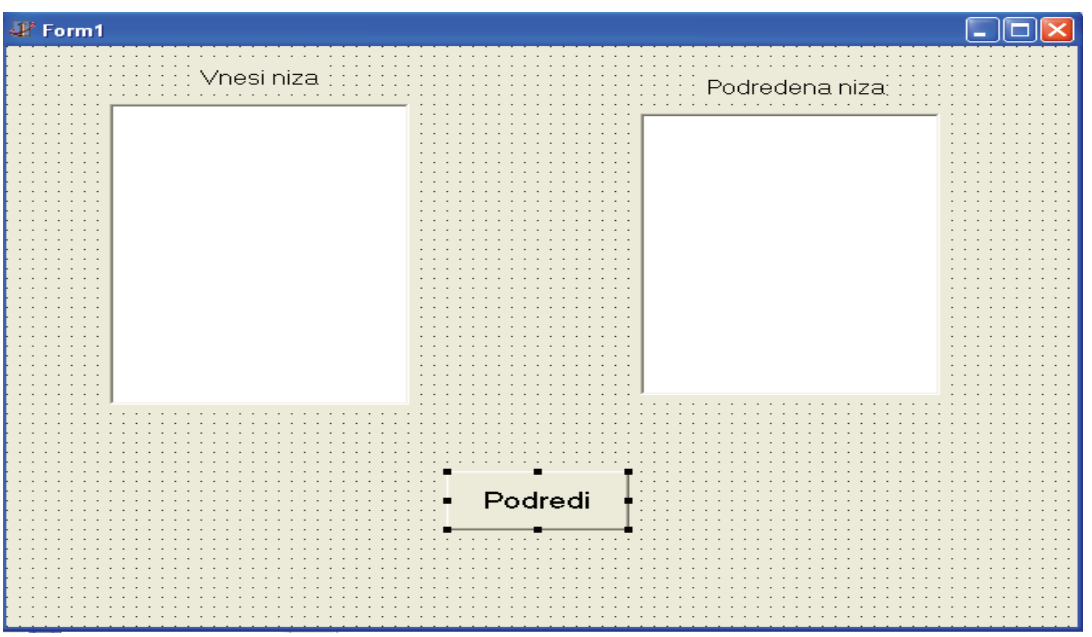
```
var
  Form1: TForm1;
  a:array[1..30] of integer;
implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
Var  max,i,j,n,gr: integer;
     s: string;
begin
           For i:=0 To 9 Do
             Begin
               s:=Memo1.Lines[i];
                 Val(s,n,gr); j:=i+1;
                    If gr = 0 Then a[j]:=StrToInt(Memo1.Lines[i]);
              End;
           max:=a[1];
             For i:=2 To 10 Do
             If a[i] > max Then max:=a[i];
             Edit1.Text:=IntToStr(max);
 end;

procedure TForm1.Button2Click(Sender: TObject);
Var i: integer;
```

```
begin
        memo1.lines.clear;
        For i:=1 To 100 Do
         a[i]:=0;
         edit1.text:=' ';
         memo1.setfocus;
         end;
end.
```

II начин

```
procedure TForm1.Button1Click(Sender: TObject);
var
        max,i,j,n,gr:integer;
        s:string;
        a:array[1..30] of integer;
begin
        max:=0;
        for i:=1 to 10 do
                a[ i ]:=0;
         for i:=0 to 9 do
         begin
                s:=memo1.lines[ i ];
                val(s,n,gr);
                j:=i+1;
                if gr=0 then a[ j ]:=strtoint(memo1.lines[ i ]);
          end;
          max:=a [ 1 ];
          for i:=1 to 10 do
                if a[ i ] > max then max:=a[ i ];
                    edit1.text := inttostr(max);
end;
```

Пример3: Да се направи програма за подредување на податоците во низата:



Слика 3.29.

```
…
var
  Form1: TForm1;
   a:array[1..30] of integer;   {globalna varijabla}
implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
var
     i,j,n,gr,pom:integer;
     s:string;
 begin
     for i:=0 to 9 do
          begin
                    s:=memo1.lines[i];
                    val(s,n,gr);
                     j:=i+1;
                     if gr=0 then a[j]:=strtoint(memo1.lines[i]);
          end;
     for i:=1 to 9 do
           for j:=i+1 to 10 do
                     if a[i]>a[j] then
                               begin
                                        pom:=a[i];
                                        a[i]:=a[j];
                                        a[j]:= pom;
                               end;
     for i:=0 to 9 do
          memo2.lines.add(inttostr(a[i+1]));
  end;

procedure TForm1.FormCreate(Sender: TObject);
          var i:integer;
begin
           for i:=1 to 10 do
            a[i]:=0;
end;
```

Треба да се внесат 10 податоци и да се внесе податок и во првиот ред.

Пример 4: Што работи следната програма?

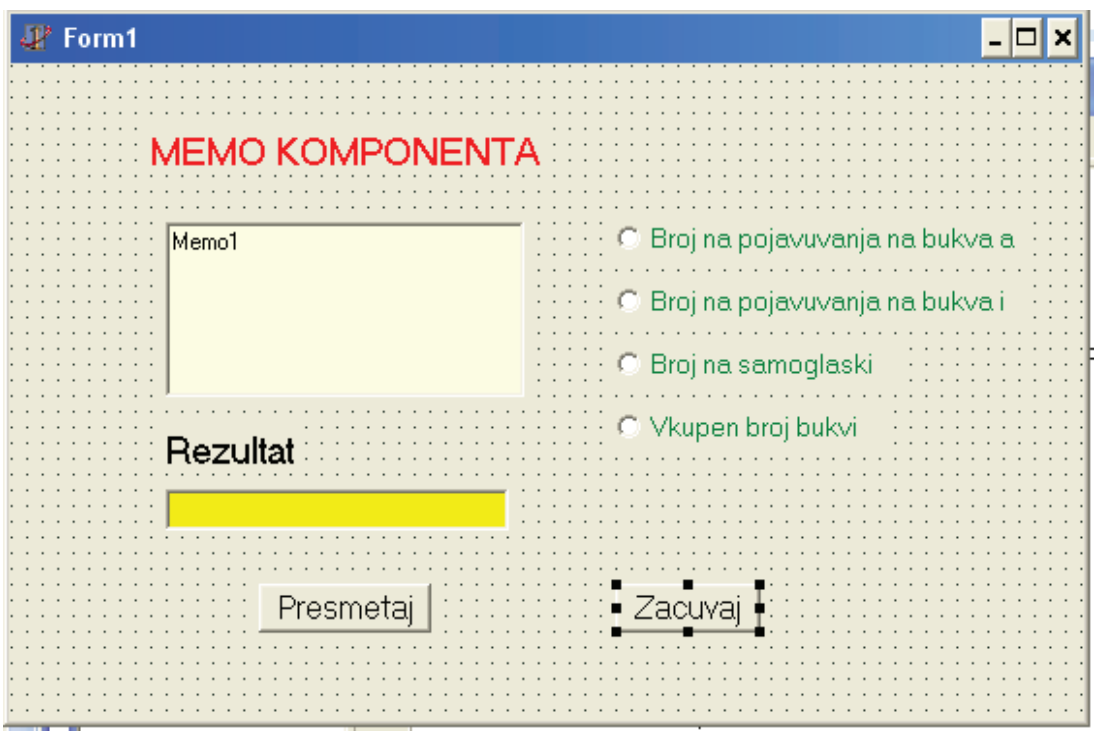
```
procedure TForm1.Button1Click(Sender: TObject);
begin
if memo1.seltext<>' ' then
  begin
              memo2.lines.text:=memo1.lines.text;
              memo2.Lines.savetofile('tekst.txt');
  end
  else
               memo1.Lines.savetofile('tekst.txt'); end;
```

Задача – проект : Отвори нова форма и на неа додај **Memo** компонента чија боја е **clInfoBk** со овозможен вертикален лизгач. Десно од неа додај 4 **RadioButton** компоненти со следните наслови: Број на појавување на буквата а, Број на појавување на буквата і, Број на самогласки и Вкупен број на букви (Font **Ms Sans Serif**, **Bold**, големина **10 и боја Green**) при што првата е чекирана. На формата постави копче со наслов Пресметај. Постави **Edit1 поле** (боја clYellow, Font **Ms Sans Serif**, **Bold**, големина **12 и** боја **Navy) со наслов над него Rezultat.** На формата постави копче со натпис **Зачувај**. Со притисок на ова копче треба да се зачува содржината на **Memo** рамката во датотеката **Vezba.txt**.

Откако ќе се повика програмата, треба да се внесат 4-5 редови латиничен текст во **Memo** рамката **,** а потоа да се повикаат зададените пресметки и да се пополни **Табелата I**.

Селектирај го и избриши го внесениот текст во **Memo** рамката, а внеси го следното: во првата реченица – ваше име и презиме, во втората датум и место на раѓање , во третата улица и место на живеење, во четвртата вашиот број на домашен телефон и мобилен и во петтата хоби, а потоа притиснете копче Зачувај. Отворете ја датотеката **Vezba.txt** и проверете дали дадената содржина е зачувана во неа.

Опис на решението:



Слика 3.30.

1.Го снимаме нашиот проект со наредба **Save Project As во** фолдер **Delphi_vezbi** на **C дискот** давајќи и име на програмата **Prog_recenica**.

2. Со копчето **Presmetaj** се повикува процедурата во која се вршат пресметките, зависно од тоа кој **RadioButton** е чекиран. Пресметката на бројот на појавувања на буквата а е следната:

```
If RadioButton1.Checked Then
Begin
        For i:=0 To Memo1.Lines.Count-1 Do
```

```
        Begin
                recen:=Memo1.Lines[i];
                For j:=1 To Length(recen) Do
                If (recen[j]='a') or (recen[j]='A') Then br:=br+1;
        End;
End;
```

Се гледа дека земаме една по една реченица од **Memo** компонентата со помош на првиот **For** циклус. Потоа одбраната реченица ја доделуваме на стрингот **recen**(**recen:=Memo1.Lines[i]**).

Во следниот **For** циклус земаме еден по еден карактер од дадената реченица и испитуваме дали тој е еднаков со мала или голема А буква. Ако е еднаков, тогаш бројачот **br** го зголемуваме за 1.

При пресметувањето на вкупниот број на букви во дадениот текст, се користи следната постапка:

Големите букви имаат ASCII вредности од 65 до 90, а малите од 97 до 122, па ако е ASCII вредноста на буквата во дадениот опсег тогаш ја зголемуваме вредноста на бројачот.

3. Со притискање на копчето Зачувај се повикува процедурата Спаси во која е наредбата:

Memo1.Lines.SaveToFile('vezba1.txt');

со чија помош го зачувуваме внесениот текст во **Memo** компонентата во датотека **vezba1.txt**.

Табела I
Broj na pojavuvanja na bukvata **a**
Broj na pojavuvanja na bukvata **i**
Broj na samoglaski
Vkupen broj na bukvi

PROGRAM:

```
program Prog1;
uses
. var
  Form1: TForm1;
   br:integer;
implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
     var i,j:integer;
     recen:string;
begin
     If RadioButton1.Checked Then
          Begin
          For i:=0 To Memo1.Lines.Count-1 Do
          Begin
                  recen:=Memo1.Lines[i];
                  For j:=1 To Length(recen) Do
                  If (recen[j]='a') or (recen[j]='A') Then br:=br+1;
```

```
        End;
    edit1.text:=inttostr(br);
    br:=0;
End;
    If RadioButton2.Checked Then
        Begin
          For i:=0 To Memo1.Lines.Count-1 Do
                Begin
                      recen:=Memo1.Lines[i];
                      For j:=1 To Length(recen) Do
                      If (recen[j]='i') or (recen[j]='I') Then br:=br+1;
                End;
           edit1.text:=inttostr(br);
           br:=0;
        End;
    If RadioButton3.Checked Then
        Begin
          For i:=0 To Memo1.Lines.Count-1 Do
                Begin
                    recen:=Memo1.Lines[i];
                    For j:=1 To Length(recen) Do
                    If (recen[j]='a') or (recen[j]='A')or (recen[j]='i') or (recen[j]='I') or
                    (recen[j]='e') or (recen[j]='E')or (recen[j]='o') or (recen[j]='O')or
                    (recen[j]='u') or (recen[j]='U')
                     Then br:=br+1;
                End;
         edit1.text:=inttostr(br);
         br:=0;
        End;
 If RadioButton4.Checked Then
  Begin
        For i:=0 To Memo1.Lines.Count-1 Do
        Begin
        recen:=Memo1.Lines[i];
        For j:=1 To Length(recen) Do
        If ((65<=ord(recen[j]))and (ord(recen[j])<90)) or ((97<=ord(recen[j]))and
        (ord(recen[j])<122)) Then
         br:=br+1;
        End;
   edit1.text:=inttostr(br);
    br:=0;
  End;
end;


procedure TForm1.FormActivate(Sender: TObject);
begin
   br:=0;
end;
```

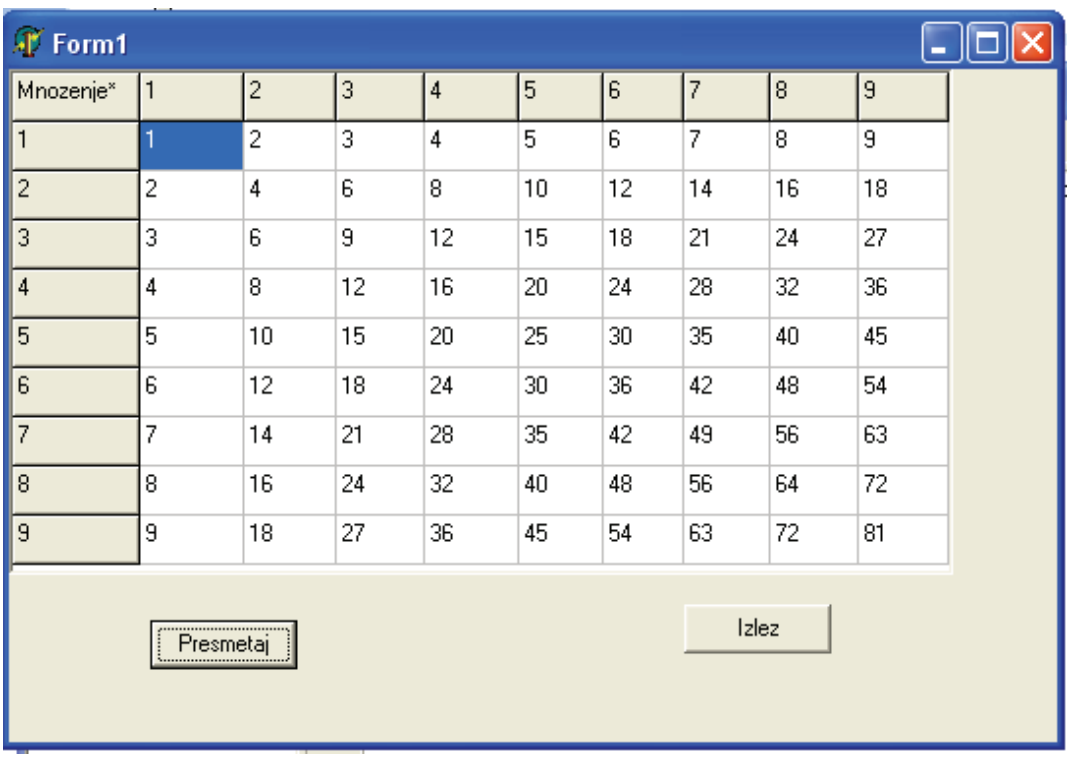
```
procedure TForm1.Button2Click(Sender: TObject);
begin
   Memo1.Lines.SaveToFile('vezba1.txt');
end;
end.
```

3.12.2. Компонента StringGrid

Оваа компонента овозможува податоците да се прикажуваат во облик на табела. Таа се наоѓа на втората картичка (Additional). Табелата е организирана по колони и редови. Во пресекот на колона и ред се наоѓа ќелија. Содржината на ќелијата е дефинирана како знаковен податок, но на секоја ќелија, освен текст, може да се придружи кој било објект кој постои во Delphi.

Компонентата StringGrid е слична со табела во Excel. Бројот на колони и редови се задава со својството ColCount и RowCount. Ќелиите во првиот ред и првата колона се со друга боја – сиви, што значи дека тие се фиксирани. Фиксирањето се постигнува со задавање на вредности на FixedRows и FixedCols (двете особини имаат вредност 1). До ќелиите се пристапува програмски врз основа на индексите кои ја одредуваат колоната и редицата, слично како кај дводимензионални полиња. Првата ќелија е [0,0].

Задача1:Да се направи апликација за таблица на множење на едноцифрени броеви.

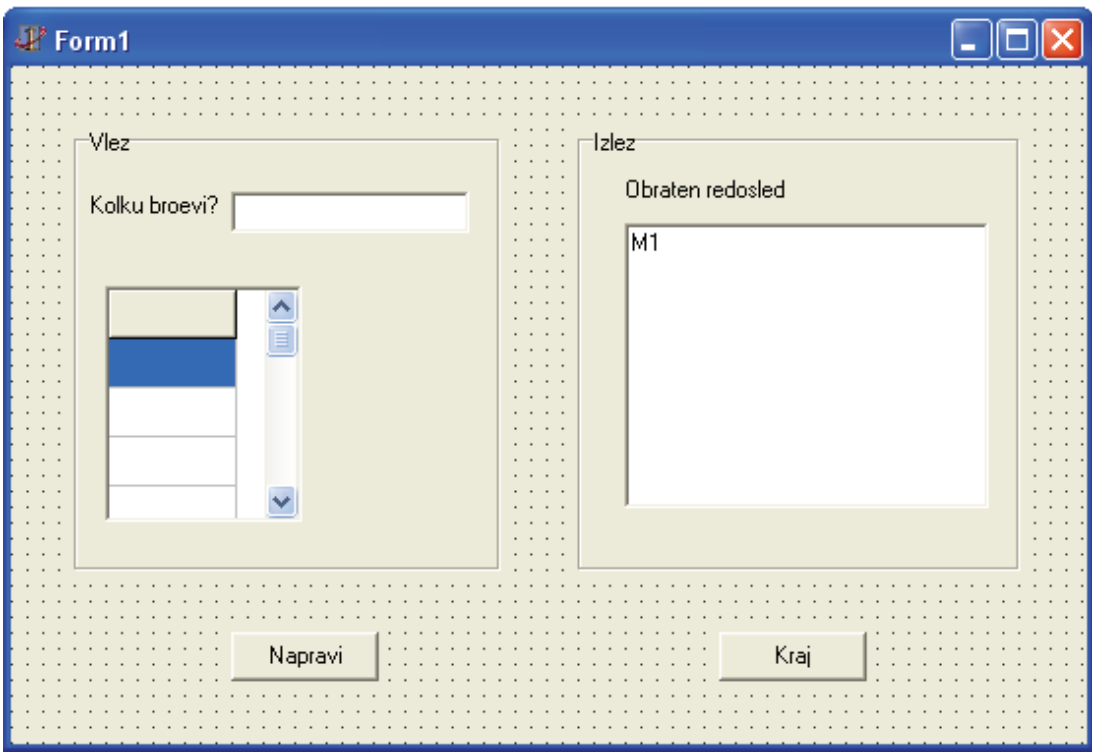


Слика 3.31.

```
Procedure Tform1.Button1Click();
Var
   I,j,n:Integer;
   S,t:String;
Begin
    Tablica.Cells[0,0]:=’Mnozenje*’;
      For i:=1 to 9 do
        Begin
                Str(i,s);
                Tablica.Cells[0,i]:=s;
                Tablica.Cells[i,0]:=s;
        End;
      For i:=1 to 9 do
        For j:=1 to 9 do
        Begin
                N:=i*j;
                Str(n,t);
                Tablica.cells[i,j]:=t;
        End;
End;
End.
```

Задача2:Да се направи апликација која овозможува внесување на n броеви и нивно прикажување во обратен редослед.

Решение: Најпрво ќе обезбедиме информација за бројот на броеви кои се внесуваат. Потоа со помош на StringGrid компонента ќе ги внесеме броевите, а потоа во Memo контрола ќе се прикажат. Формата е прикажана на слика 3.32.



Слика 3.32

На формата се поставени 9 компоненти. Тие се групирани според намената на податоците. За StringGrid компонентата треба да се постават следните својства: ColCount=1, FixedCols=0, FixedRows=1, RowCount=5, With=100 и Name=Niza

Програмскиот код со кој е решена задачата е следниот:

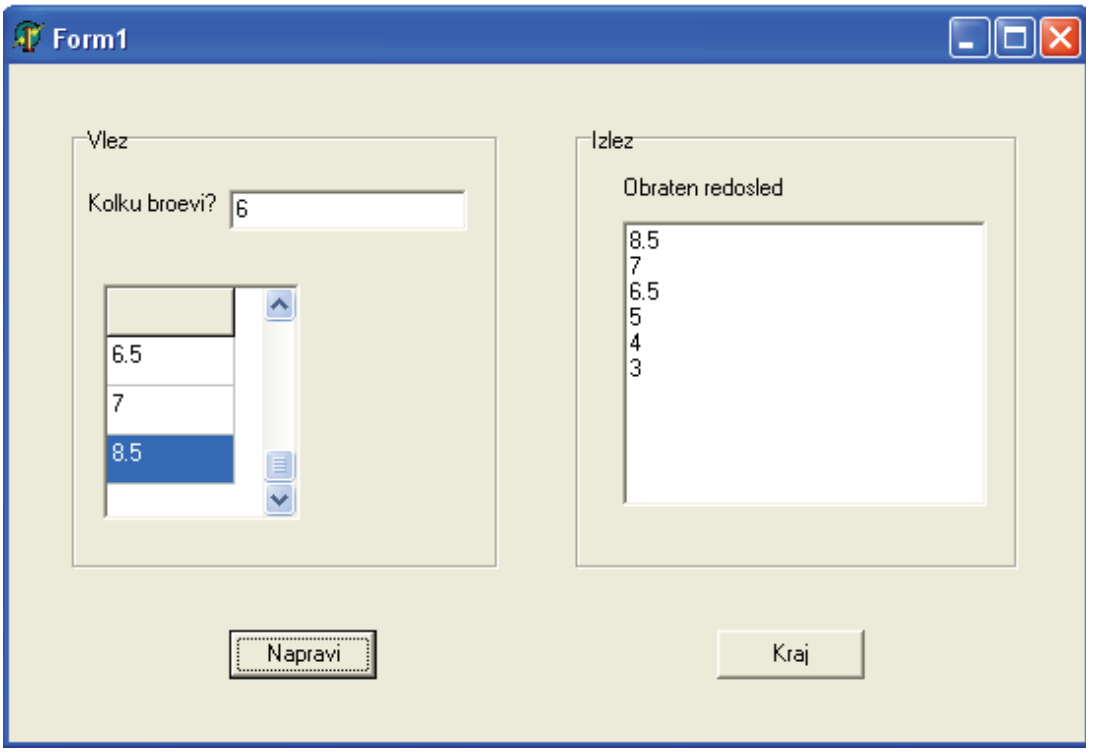
```
procedure TForm1.Button2Click(Sender: TObject);
begin
        application.terminate;
end;

procedure TForm1.NizaSelectCell(Sender: TObject; ACol, ARow: Integer;
  var CanSelect: Boolean);
  var i,n:integer;
begin
          val(broj.text,n,i);
          Niza.RowCount:=n+1;
 end;

procedure TForm1.Button1Click(Sender: TObject);
var i,j,n,m:integer;
begin
          val(broj.text,n,j);
          m1.lines.clear;
          For i:=n downto 0 do
          m1.lines.add(niza.cells[0,i]);
end;
end.
```

За да може да се внесуваат податоци во полињата на компонентата StringGrid, потребно е во својството Options, опцијата goEditing да се постави на вредност True.

На компонентата StringGrid и доделуваме настан OnSelectCell кој се активира кога корисникот се обидува да ја селектира ќелијата. Овој настан ни овозможува во ќелиите на табелата со една колона да ги внесеме вредностите на саканите броеви.

Со копчето Направи го активираме настанот OnClick, кој овозможува во Memo полето да се прикажат внесените броеви во обратен редослед. Линијата „m1.lines.clear;“ овозможува бришење на содржината на Memo компонентата, додека линијата „ m1.lines.add(niza.cells[0,i]); “, овозможува прикажување на броевите внесени во табелата во обратен редослед.

Со извршување на апликацијата се добива прозорецот од слика 3.33:



Слика 3.33

Задача 3: Во претходниот пример да се направат корекции, така што ќе се пресмета и ќе се прикаже збирот на внесените броеви.

3.13. Мултимедијални апликации

Со развојот на компјутерската технологија, создадени се можности за користење графика и звук во рамките на апликацијата. Delphi овозможува програмирање на мултимедијални апликации. За прикажување графика во Delphi, се користат следните компоненти:

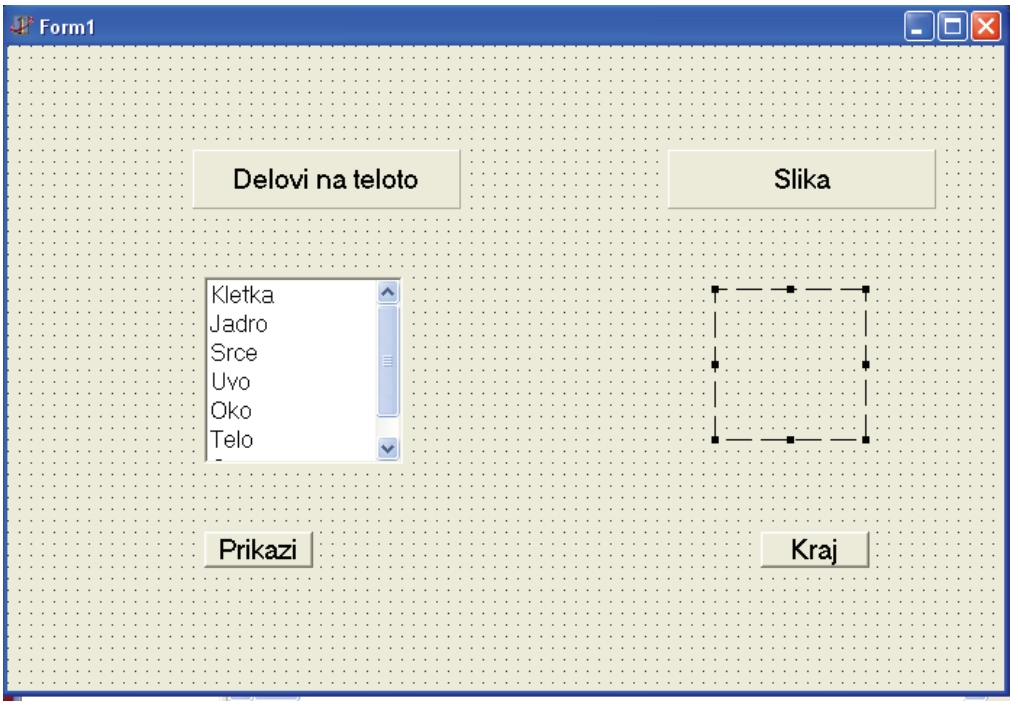
Компонента Shape. Се наоѓа на картичката Additional и се користи за додавање едноставни облици на формата..

Важни својства се Brush и Pen. Brush ја одредува бојата на четката или дезенот на внатрешноста . Pen го дефинира стилот и бојата на линијата со која се црта.

Може да се цртаат правоаголници, елипси, квадрати...

Image компонента се користи за прикажување на слика на формата. Таа се наоѓа на картичката Additional. Во Delphi се поддржани следните формати на слики: jpg, jpeg, bmp, ico,emf, wmf. Ако имаме слика од друг формат, таа треба да се преведе во некој од форматите кои се препознатливи за Delphi. Ако својството Autosize се постави на true, тогаш компонентата Image ќе се прилагоди на големината на сликата. Додека, ако својството Strech се постави на true, тогаш големинатата на сликата ќе се прилагоди на големината на компонентата. Ако двете својства се постават на False, сликата ќе зазема само дел од компонентата или нема да се гледа цела во компонентата, во зависност од нејзината големина. Која слика ќе се донесе, зависи од својството Picture со кое се појавува Picture editor, преку кој се внесува сликата со копчето Load.

Задача 1: Да се направи апликација која врз основа на постоечки графички материјал ќе ги прикажува деловите на човечкото тело – мала енциклопедија. Формата е прикажана на слика 3.34.



Слика 3.34.

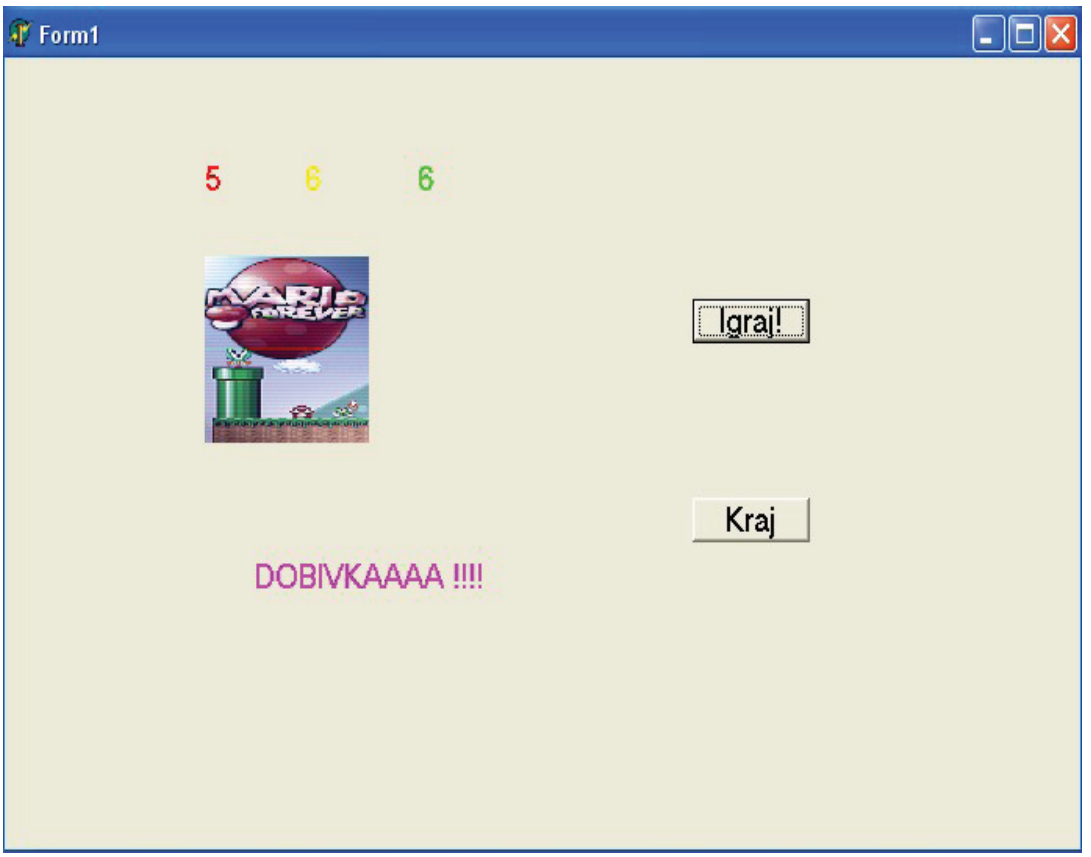
На формата има 2 панели, Listbox, слика и 2 командни копчиња. Корисникот избира една опција од листата , кликнува на копчето Prikazi и соодветната слика се прикажува во предвидениот дел на формата.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
     application.terminate;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
       image1.visible:=false;
       case listbox1.itemindex of
       0:begin
       image1.visible:=true;
       image1.picture.loadfromfile('c:\program files\borland\'+
      '\delphi5\projects\sliki\sl1.bmp'); {pat do folderot so sliki}
       end;
       end;
end;
end.
```

Во програмскиот код се користи својството Visible на компонентата Image за овозможување или оневозможување на приказот на сликата.

Можно е и второ решение на задачата , при што прикажувањето на сликата ќе оди само со кликање низ опциите на листата.

Задача 2: Да се направи апликација со која ќе се симулира работата на апарат за среќа. Работата на тој апарат е следната: по секое кликање на копчето Igraj! се појавуваат три случајни броеви од интервалот од 0 до 9. Ако еден од тие броеви е 5, играчот добива пари. Симулација на добивка се реализира со појава на пари на местото на компонентата за слика. Формата е прикажана на слика 3.35.



Слика 3.35.

```
procedure TForm1.Button1Click(Sender: TObject);
var r,p,q:Integer;
s:String;
begin
     Image1.visible:=false;
     r:=random(10);
     str(r,s);
     Label1.caption:=s;
     p:=random(10);
     str(p,s);
     Label2.caption:=s;
     q:=random(10);
     str(q,s);
     Label3.caption:=s;
      If (p=5) or (r=5) or (q=5) then
        begin
            image1.visible:=true;
            Beep;
```

```
            label4.caption:='DOBIVKAAAA !!!!';
        end;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
   application.terminate;
end;
```

Функцијата Random(10) генерира случаен број во интервалот од 0 до 9. Во кодот, сликата на почеток е невидлива. Таа ќе стане видлива, ако барем еден од генерираните броеви е 5.

При секое ново стартување на апликацијата се добиваат исти серии на броеви. За да се надмине тоа, се вметнува наредбата Randomize која го користи системскиот часовник на компјутерот за одредување навистина случаен број. Оваа функција се вметнува на настанот FormActivate и ќе се изврши при секое стартување на апликацијата.

```
procedure TForm1.FormActivate(Sender: TObject);
begin
   randomize;
   image1.visible:=false;
end;
```

Освен овој начин на прикажување на слики, Delphi овозможува прикажување на слики, цртање на слики и внесување на текст на произволно место на прозорецот на извршување на апликацијата. За оваа цел се користи посебен објект, платно за цртање Canvas. Овој објект се користи за цртање на работната површина на формата .

Canvas објектот се користи, така што најпрво со помош на својствата се одредуваат параметрите за цртање, а потоа се црта. Значајни својства за цртање се видот и дебелината на линијата, обликот на фигурата и полнењето со боја, изгледот на фонтот и сл. Неопходни параметри се координатите на цртежот, неговата големина, боја и сл.

Објекти кои најчесто се користат при цртањето и пишувањето на Canvas се: перо - пенкало, четка и фонтови.

Пенкалото (pen) го дефинира начинот на прикажување на линијата при цртањето. На пенкалото може да му се дефинираат следните својства: боја, стил и ширина (дебелина).

Четката (brush) овозможува пополнување на нацртаната област. Тоа може да биде пополнување со боја , но и шрафура.

Постојат голем број методи за цртање по платното (Canvas). Методи кои најчесто се користат се следните:

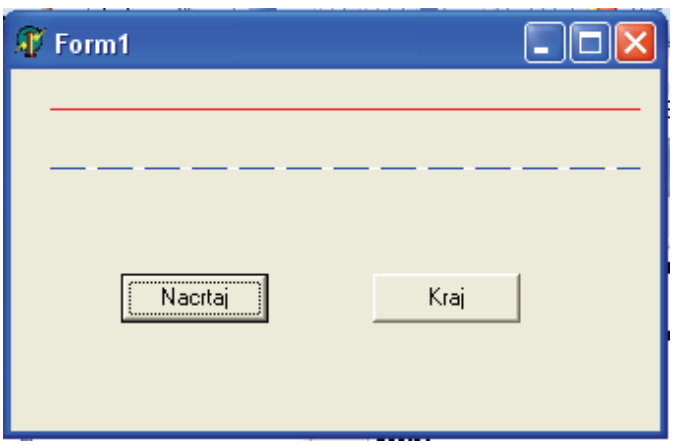
1. Ellipse, која црта елипса со моментно одбраното перо и пополнува со моментно одбраната четка.
2. LineTo(x,y), која црта линија од тековната положба до положбата дефинирана со координатите x и y.
3. MoveTo, која поставува тековна положба за натамошно цртање.
4. Polygon, која црта полигон на дадени кординати.
5. PolyLine - црта линија низ дадена низа на точки.
6. Rectangle - црта правоаголник.
7. RoundRect - црта правоаголник со заоблени темиња.

Задача3: Да се нацртаат две линии на работната површина на апликацијата, една црвена полна и една сина испрекината. На формата треба да поставиме копче и да го програмираме неговиот настан OnClick. На наведениот настан му го придружуваме следниот програмски код :

```
procedure TForm1.Button1Click(Sender: TObject);
begin
        Canvas.pen.color:=clred;
        canvas.MoveTo(20,20);
        canvas.lineto(320,20);
        canvas.pen.color:=clblue;
        canvas.pen.style:=psdash;
        canvas.moveto(20,50);
        canvas.LineTo(320,50)
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
        application.terminate;
end;
end.
```

Од кодот може да заклучиме дека платното за цртање е координатна рамнина, кај која до секоја точка може да се пристапи со наведување на нејзините координати. Координатниот почеток е поставен во горниот лев агол на прозорецот.

Со извршување на апликацијата се добива прозорец како на слика 3.36.



Слика 3.36.

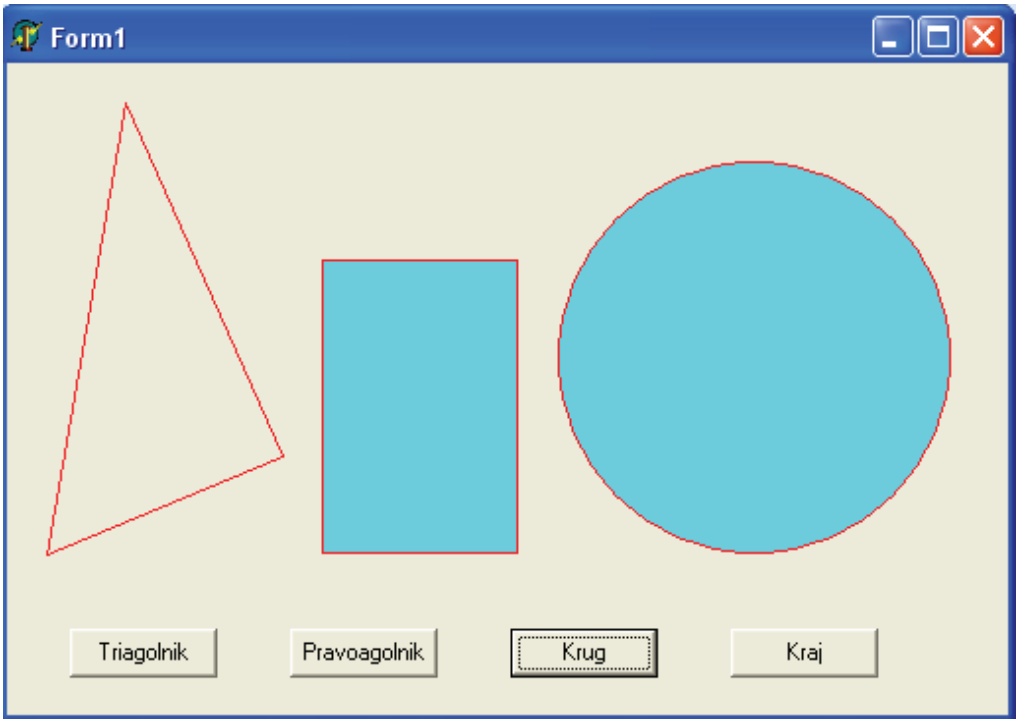
Задача4: Да се формира апликација која на canvas – от ќе нацрта по еден триаголник, правоаголник и круг. На формата поставуваме четири командни копчиња со натпис Triagolnik, Pravoagolnik, Krug и Kraj.

Програмскиот код е следниот:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
        canvas.MoveTo(20,250);
        canvas.lineto(140,200);
        canvas.lineto(60,20);
        canvas.LineTo(20,250)
end;
```

```
procedure TForm1.Button2Click(Sender: TObject);
begin
        application.terminate;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
        canvas.rectangle(160,250,260,100);
end;
procedure TForm1.Button4Click(Sender: TObject);
begin
        canvas.ellipse(480,50,280,250);
end;
procedure TForm1.FormActivate(Sender: TObject);
begin
        canvas.Pen.color :=clred;
        canvas.brush.color:=claqua;
end;
end.
```

Формата ќе го има изгледот прикажан на слика 3.37.



Слика 3.37.

Во методот „canvas.rectangle(160,250,260,100);“ се наведени аргументите кои го имаат следното значење : (160,250) претставува координата на долното лево теме, а (260,100) е координата на горното десно теме на правоаголникот. Координатите зададени во методот за цртање елипса, претставуваат темиња на правоаголникот во кој е впишана елипсата.

Освен цртање на фигури во canvas , може да се прикаже (црта) текст со помош на методите TextOut и TextRect.

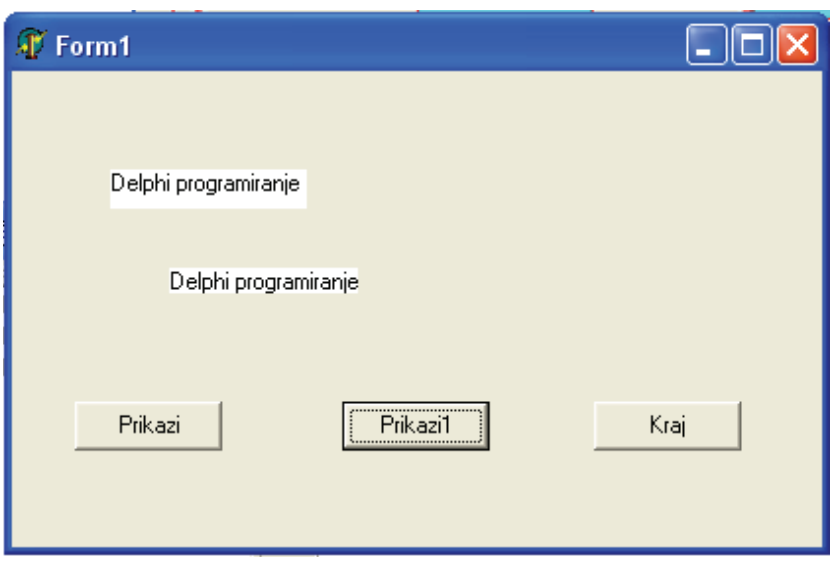
Првиот метод овозможува прикажување на текст на позиција одредена со координатата на горниот лев агол од кој се прикажува текстот.

Доколку е потребно, текстот кој се прикажува да биде со одредена должина и во рамките на некоја правоаголна област, се користи методот TextRect.

Програмскиот код за практична примена на овие две наредби е следниот:

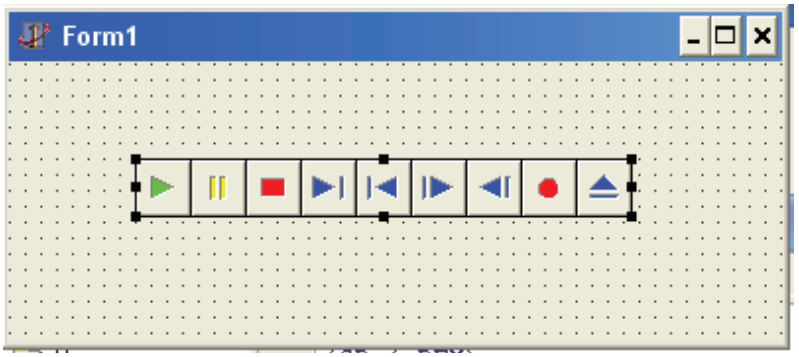
```
procedure TForm1.Button1Click(Sender: TObject);
begin
        canvas.textOut(80,100,'Delphi programiranje');
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
        application.terminate;
end;
procedure TForm1.Button3Click(Sender: TObject);
begin
        canvas.textrect(rect(50,50,150,70),50,50,'Delphi programiranje');
end;
```

Формата, откако ќе се активира апликацијата, го има изгледот како на слика 3.38.



Слика3.38.

3.13.1.Компонентата MediaPlayer за едноставни мултимедијални апликации се наоѓа на страната System . Оваа компонента може да извршува звучни датотеки и видеодатотеки, доколку поседуваме датотеки чиј формат Delphi го препознава. Ако при стартување на некоја датотека, Delphi испрати порака дека не го препознава тој формат, неопходно е да се набави соодветен драјвер. Со поставување на оваа компонента на формата се добива форма како на сликата3.39.



Слика 3.39.

На сликата гледаме дека оваа компонента содржи копчиња за стартување, пауза, запирање, брз премин напред, прескокање напред и назад, снимање и исфрлање на медиумот. Значи, на контролниот панел се наоѓа се она што се наоѓа на повеќето касетофони.

Користењето на компонентата MediaPlayer е многу едноставно. Откако ќе се постави компонентата на формата, треба само да се постави нејзиното својство FileName на вредност која е име на соодветната мултимедијална датотека. Потоа се стартува апликацијата и се кликнува на копчето Play. Ако спомнатото копче не е достапно, значи дека својството кое одредува дали уредот е отворен после креирањето е поставено на вредност False. Во тој случај, својството AutoOpen треба да се постави на вредност true. После кликот на копчето Play, треба да се појави звук зачуван во датотеката чие име е доделено на својството FieldName.

Компонентата MediaPlayer има вграден механизам со чија помош автоматски се препознава дали станува збор за аудио или видеодатотека.

За да може програмски да се управува со компонентата MediaPlayer, потребно е да се запознаеме со нејзините основни својства:

1. AutoOpen одредува дали уредот ќе биде отворен по креирањето на компонентата MediaPlayer.Стартната вредност е False.
2. AutoRewind овозможува поставување на покажувачот на позицијата на датотеката на почеток кога датотеката ќе се изврши до крај.
3. DeviceType овозможува автоматско прилагодување на уредот на типот на датотека. За да се овозможи тоа, вредноста на ова својство треба да е dtAutoSelect.
4. Display е важно својство за видеоуредите. Тоа овозможува прикажување на видеодатотеката на компонента која може да прими приказ.
5. DisplayRect е својство кое е во непосредна врска со претходното и овозможува големината на приказот на видеодатотеката да се прилагоди со големината на компонентата на која се прикажува. Вредноста на својството се поставува програмски.
6. VisibleButtons одредува кои копчиња на MediaPlayer ќе бидат активни. Се подразбира дека сите копчиња ќе бидат активни. Вредноста на ова својство може да се постави програмски или преку Object Inspector.
7. Wait дефинира дали контролата се враќа на апликацијата во моментот или по завршувањето на музичката датотека.

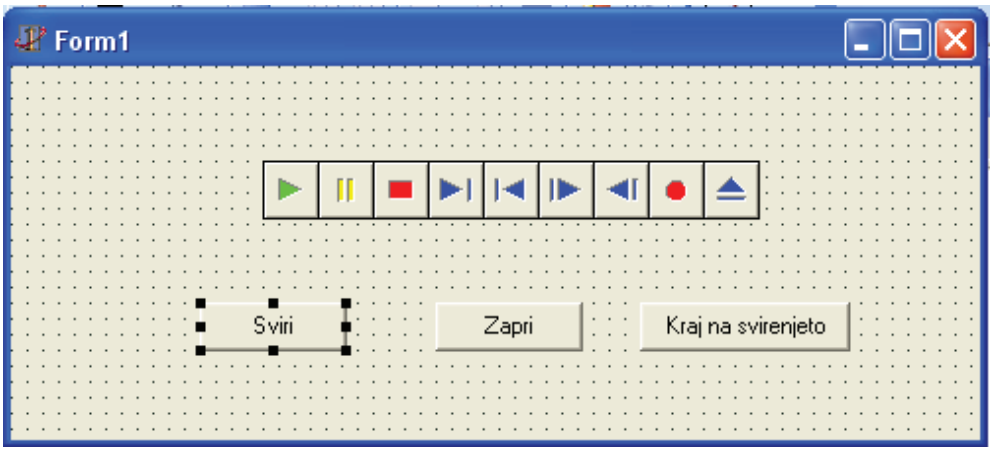
Најзначајни методи кои ги користи компонентата MediaPlayer се:

1. Close е метод кој го затвора уредот.

2. Eject е метод, кој го исфрла медиумот ако е можно.
3. Next метод преминува на почеток на следната лента, доколку уредот поддржува ленти.
4. Open метод го отвора уредот. Овој метод се користи ако својството AutoOpen е поставено на False.
5. Pause метод привремено го запира извршувањето на датотеката.
6. Play метод го активира извршувањето на датотеката.
7. Stop метод ја запира тековната активност.

Во некои случаи пожелно е компонентата MediaPlayer да биде невидлива во периодот на извршување на звучната датотека. Во тој случај, со компонентата управуваме со помош на програмски код.

Пример1: Да се формира нова апликација, чија форма изгледа како на сликата 3.40.

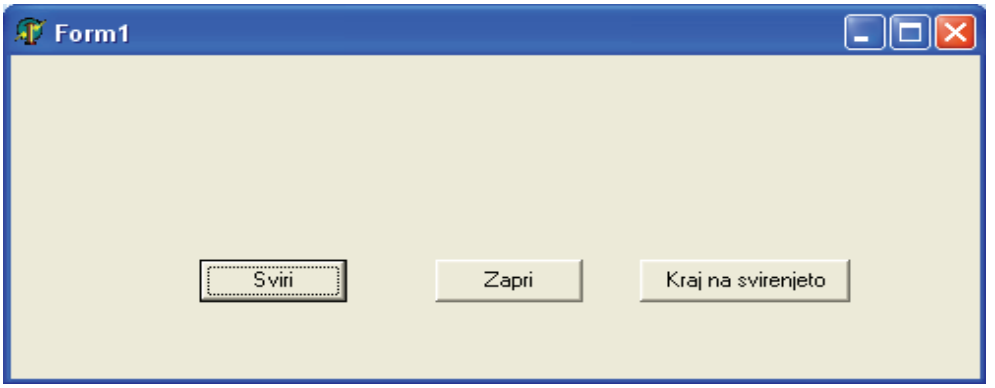


Слика 3.40.

Програмскиот код кој овозможува скривање на компонентата MediaPlayer и нејзина програмска контрола е следниот:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
        muzika.visible:=false;
        muzika.filename:='C:\Program
        Files\Borland\Delphi5\Projects\muzika\01_track_1.mp3';
        muzika.open;
        muzika.play;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
        muzika.stop;
        muzika.visible:=true;
end;
procedure TForm1.Button3Click(Sender: TObject);
begin
        application.terminate;
end; end.
```

На сликата 3.41.е прикажан работниот прозорец на апликацијата за време на работа на музичката датотека, чие извршување може привремено да се запре или потполно да се прекине.



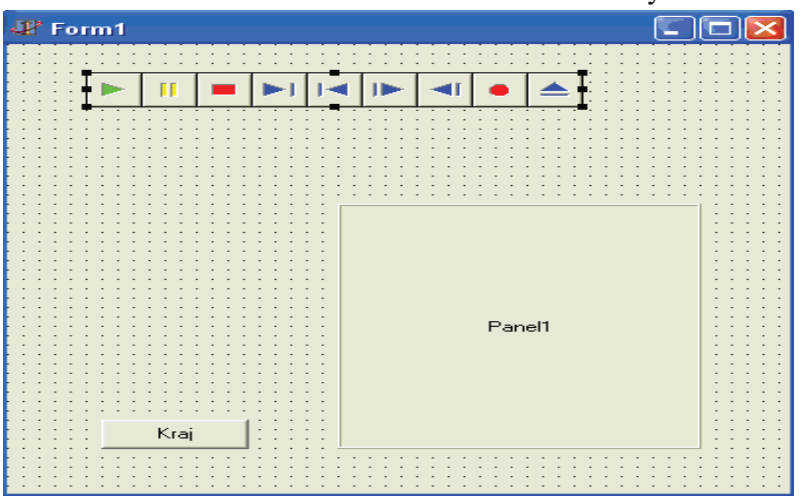
Слика 3.41.

Користењето на аудиодатотека од CD дискови е многу едноставно, кога за тоа се користи MediaPlayer. Потребно е да се промени својството DeviceType на CDAudio, а потоа да се кликне на копчето Play.

Користењето на MediaPlayer компонентата за прикажување видеодатотеки, се сведува на поставување на својството FileName од Object Inspector – от на вредност, која претставува пат и име на видеодатотеката и клик на копчето Play.

Можно е приказот на видеодатотеката да се насочи и на некоја друга компонента, на пример, панел. Ова се постигнува со поставување на својството Display од MediaPlayer компонентата на вредност на името на компонентата (пример Panel1). Ако целата слика на видеодатотеката не може да се прикаже, во одбраната компонента ќе се прикаже само еден нејзин дел. Со поставување на својството DisplayRect, сликата може да се собере или рашири, така што целата ќе биде во одбраната компонента. Тоа се постигнува програмски.

Пример 2: Да се направи апликација која ќе овозможи прикажување на видеодатотека во компонента Panel. Со стартување на апликацијата, се добива работен прозорец во кој во компонентата Panel се извршува видеодатотеката. Својството Name на компонентата MediaPlayer го менуваме во Slika и го програмираме настанот OnClick на компонентата MediaPlayer.



Слика 3.42.

```
procedure TForm1.SlikaClick(Sender: TObject; Button: TMPBtnType;
  var DoDefault: Boolean);
begin
  slika.displayrect:=panel1.clientrect;
end;
```

ПОГЛАВЈЕ 3
(НАКРАТКО)

Кодот се извршува како резултат на некој настан. Ако кликнеме на копчето, автоматски се генерира настанот (event) On Click кој автоматски го извршува програмскиот код што е придружен на овој настан.

Придружување код на командното копче Прикажи:

- I - начин
  - Се селектира копчето
  - Од картичката настани (events ->Object Inspector) се избира настанот On Click
  - Двојно се кликнува во полето десно од избраниот настан, при што се отвора процедура во која го запишуваме програмскиот код.
- II – начин
  - Кликнуваме двапати врз командното копче Прикажи. Во прозорецот за кодот автоматски се генерира името на процедурата и името на објектот, кај кој е активен настанот што ќе го предизвикува даденото дејство.

Елементи на јазикот се:

Множество на знаци, коментари, идентификатори, службени зборови и стандардни директиви

Податоци

Целобројни, реални, логички, знаковни , константи, променливи

Оператори и изрази

Аритметички оператори
Релациски оператори
Логички оператори

Наредба за доделување на вредност „:=“

Проектирање на апликации од линиска структура:

Програмите од оваа структура напишани во Pascal содржат наредби за внесување на податоци, за обработка на податоци и за прикажување резултати.

Претворање на податоци:

```
Function StrToInt (const  s:string):Integer;
Function StrToFloat(const s:string):Extended;
Function IntToStr(value:Integer):String;
Function FloatToStr(value:Extended):String;
function TimeToStr(Time: TDateTime): string;
function DateToStr(Date: TDateTime): string;
```

Разгранети структури:

Општ облик на IF наредба или наредба за избор од две можности
Општ облик на CASE наредба или наредба за избор од повеќе можности

Компоненти за избор:

CheckBox контрола. Контролата **TCheckBox се** наоѓа на страната Standard. Се користи при решавање задачи, кога е потребно да се овозможи вклучување или исклучување на една или повеќе опции.

RadioButton контрола. **RadioButton** контролата се користи за избор на една од неколку можни опции.

Radiogroup контрола. Радиокопчињата можат да бидат сместени во посебна кутија – RadioGroup. Целта на постоењето на RadioGroup е да се поедностави групирањето на радиокопчињата на едно место.

Компонента ListBox. Претставува вообичаен Windows прозорец за листа. Листата содржи список на можности кои корисникот може да ги одбере.

Компонента ComboBox. ComboBox е компонента која ги комбинира можностите за внесување на текст на Edit бокс и избор на една од понудените можности на ListBox.

Контејнерски компоненти

Компонента GroupBox . Оваа компонента нема некое посебно значење независно од останатите компоненти. Нејзиното значење е да биде контејнер (кутија) за група компоненти кои се сместуваат во неа.

Panel

Bevel (рамка)

Програмски модули

Општ облик на програмскиот модул

Заглавие на модулот – наслов (Unit)
Interface
    Декларативни делови
    Заглавие на потпрограмата
Implementation
    Декларативни делови
    Дефиниција на потпрограмите
Initialization (почетни вредности)
    наредби
finalization
    наредби
    end.

Опсег на идентификаторот. Опсег на идентификаторот е дел од изворната програма во која тој идентификатор може да се користи.

Циклични структури

WHILE-DO naredba
REPEAT UNTIL наредба
FOR naredba
FOR-TO-DO (За зголемувај до)
FOR-DOWNTO-DO (За намалувај до)

Компоненти за работа со низи

Компонентата (Memo) . Компонентата (Memo) се наоѓа на страната Standard во палетата на компоненти. Таа е слична на Edit box. Основната разлика е во тоа што во Memo компонентата може да се прикажува текст во повеќе редови.

Компонента StringGrid . Оваа компонента овозможува податоците да се прикажуваат во облик на табела.

Со развојот на компјутерската технологија, создадени се можности за користење графика и звук во рамките на апликацијата. Delphi овозможува програмирање на **мултимедијални апликации**.

Компонента Shape. Се наоѓа на картичката Additional и се користи за додавање едноставни облици на формата..

Важни својства се **Brush** и **Pen**. Brush ја одредува бојата на четката или дезенот на внатрешноста . Pen го дефинира стилот и бојата на линијата со која се црта.

Image komponenta се користи за прикажување на слика на формата. Таа се наоѓа на картичката Additional.

Canvas објектот се користи, така што најпрво со помош на својствата се одредуваат параметрите за цртање, а потоа се црта. Значајни својства за цртање се видот и дебелината на линијата, обликот на фигурата и полнењето со боја, изгледот на фонтот и сл. Неопходни параметри се координатите на цртежот, неговата големина, боја и сл.

Освен цртање фигури, во canvas – от може да се прикаже (црта) текст со помош на методите **TextOut и TextRect**.

Компонентата **MediaPlayer** за едноставни мултимедијални апликации се наоѓа на страната System . Оваа компонента може да извршува звучни датотеки и видеодатотеки, доколку поседуваме датотеки чиј формат Delphi го препознава.

Прашања и задачи:

1. Кој настан се јавува ако се кликне на формата на апликацијата, а кој ако се кликне на копчето?
2. На кои два начина може да се отвори управувачот на настани за пишување на програмски код на командното копче Button1?
3. Објасни ги функциите:
   -IntToStr
   -StrToFloat.
   -StrToInt
   -FloatTo Str
4. Која процедура ќе се отвори, ако кликнете на копчето Button1 од формата Form1?
5. Што се постигнува со наредбата за доделување на вредност?
6. За да се овозможи вклучување или исклучување на една или повеќе опции, се користи контролата_____.
7. Што е CheckBox контрола и кога се користи?

Напиши наредба со која ќе се провери дали е кликнато на оваа контрола и ако е, да отпечати во Labela1 дека е кликнато, а ако не е кликнато, да отпечати во Labela1 дека не е кликнато.

8. Објасни ги програмските редови:
   -StringGrid1.Row:=ComboBox1.itemindex+1;
   -Edit1.SetFocus;
   -StringGrid1.Cells[0,1]:='1.';
9. Што е RadioButton контрола и кога се користи?

Напиши наредба со која ќе се провери дали е кликнато на оваа контрола и ако е, да отпечати во Label2 дека е кликнато, а ако не е кликнато, да отпечати во Label2 дека не е кликнато.

10. Објасни ги својствата:
    -Items
    -ItemIndex
11. Која од наведените опции е дел на Delphi работна околина?
    А) Object Projector,
    Б) Object Inspector,
    В) Object Detector.
12. На кои начини може да се внесува текст во Memo компонента?
13. Дали секогаш може да се менува текстот внесен во Memo компонента?
14. Како се однесува Memo компонентата при внесување на текст ако својството WordWrap е поставено на False?
15. Која Delphi компонента се користи за прикажување на податоци во повеќе редови и во повеќе колони?
16. Кои објекти се прикажуваат со StringGrid компонентата?
17. Кој број го означува индексот на првиот елемент во низата 0 или 1?
18. Во третата колона, која е празна, напиши ја буквата (а,б,в) која се наоѓа пред наредбата за циклична структура која е напишана на левата страна, а која е објаснета со некој од исказите на десната страна.

а)	Repeat – until		структура повторување со услов на крај на циклусот,
б)	While do		структура повторување со услов на почеток на циклусот,
в)	For do		структура повторување со скок на одреден програмски ред
			Структура повторување со броење на циклусите

19. Во третата колона, која е празна, напиши ја буквата (а,б,в,г) која се наоѓа пред исказот кој е напишан на левата страна, а кој е објаснет со некој од исказите на десната страна.

а)	Properties		настани
б)	Events		својства
в)	Tools		палета на компоненти
г)	Component palette		ленти со алатки

Задачи

1. Направи програма со која ќе се внесуваат 5 броеви и ќе се пресметува аритметичка средина од тие броеви. (Пример со Edit компоненти).
2. Да се направи апликација со која ќе се обои edit контролата во зависност од внесениот број во edit. (1-сина, 2-жолта, 3-розова).
3. Направи програма со која ќе се споредуваат 2 броја и ќе се печати поголемиот од нив. (Пример со Edit компоненти).
4. Да се направи нова апликација која ќе ни го прикажува тековното време и датум.
5. Да се направи апликација која ќе одредува и ќе прикажува колку пати секоја самогласка се појавува во реченицата.
6. Да се напише апликација за одредување на староста на личноста врз основа на датумот на раѓање и денешниот датум.
7. Да се направи програма која ќе работи како квиз со 6 прашања. За секое прашање треба да има по 4 понудени одговори (опции), од кои само еден е точен. На крај, во втора форма да се прикажат освоените бодови и добиената оценка.
8. Да се направи игра „Едноракиот Џек“ . На формата во три полиња автоматски се менуваат броевите. Кога ќе се кликне на копчето „ запри ме “, доколку сите три броја не се еднакви и понатаму се менуваат, а доколку се еднакви, се јавува порака – честитка и прашање за повторно играње.
9. Да се напише апликација која ќе црта линии со случајна должина на екранот.

ПОГЛАВЈЕ 4
4.ПИШУВАЊЕ НА ПОРАКИ (ДИЈАЛОГ РАМКИ)

Во ова поглавје ќе бидат разработени следните можности:
-опишување на системот за пораки;
-применување на командата showmessage;
-применување на командата messagedlg;
-применување на командата messagedlgpos.

Формите како основни објекти во Delphi може да се третираат како дијалог - прозорци. Но кога ќе речеме дијалог - прозорец, подразбираме специјален вид прозорец кој ги има следните особини:

1.Големината на рамката не може да се менува,

2.Овој прозорец најчесто користи копче OK, но некои прозорци имаат копче Close, Cancel, Help,

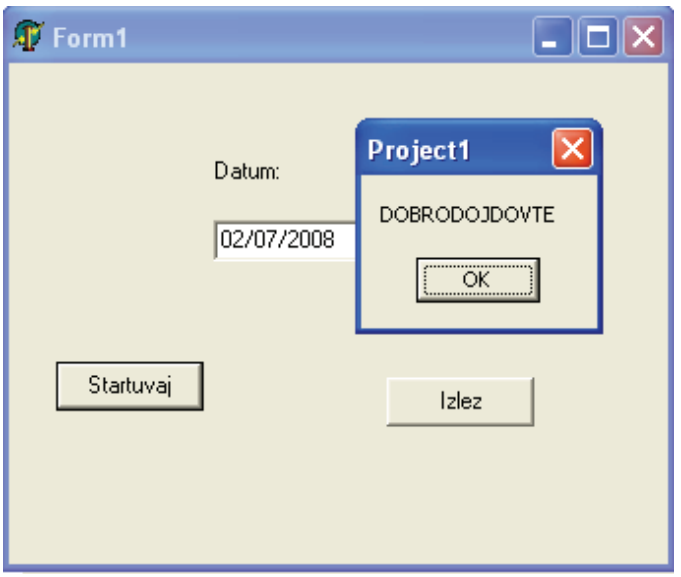
3. Во насловната лента нема копче за минимизација и максимизација.

Во рамките на Delphi постојат рамки кои сами може да ги дефинираме и прозорци кои се веќе дефинирани.

4.1. ShowMessage

Тој се користи за прикажување соодветни пораки при извршување на апликацијата. Појавувањето на дијалог - прозорецот го запира извршувањето на апликацијата, до неговото затворање, со кликање на копчето OK. Насловната лента го содржи името на проектот. Овој дијалог се користи кога апликацијата треба да му прикаже на корисникот некоја порака.

Задача1: Да се направи апликација која на корисникот му кажува дали му е дозволена работа со апликацијата или не. Работата е дозволена само во работни денови. Неработни денови се сабота и недела.



Слика 4.1.

```
procedure TForm1.Button1Click(Sender: TObject);
var
    datum:TdateTime;
    i:integer;
begin
    Datum:=StrToDate(edit1.text);
     i:=DayOfWeek(datum);
      if (i=1) or (i=7) then
        begin
        ShowMessage('DENES E NERABOTEN DEN NE MOZETE DA
        PRISTAPITE NA APLIKACIJATA');
        Close;
        end
        else
        ShowMessage('DOBRODOJDOVTE');
        end;
end.
```

За реализација на поставената задача, користена е функцијата **DayOfWeek(datum)** која врз основа на датумот го дава редниот број на денот во неделата. Според неа, прв ден е недела: 1, а последен сабота: 7.

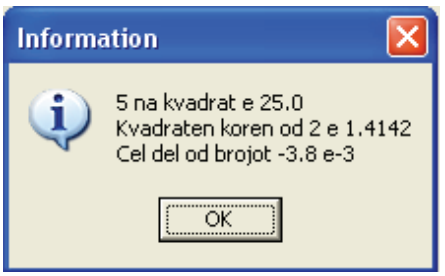
Програмскиот код овозможува внесување на погрешен датум, пример, за неработен ден да се внесе датум од работен ден.

Да се направи корекција на програмскиот код, на тој начин што ќе се оневозможи пристап до апликацијата ако системскиот датум не се слага со внесениот. Во тој случај да се испише порака :
ВНЕСЕНИОТ ДАТУМ НЕ Е ДЕНЕШЕН.

```
procedure TForm1.Button1Click(Sender: TObject);
var
   datum:TdateTime;
    i:integer;
begin
        Datum:=StrToDate(edit1.text);
        i:=DayOfWeek(datum);
        Label2.Caption := 'Today is  ' + DateToStr(Date);
          if datum<>date then
                BEGIN
                ShowMessage('NE VNESOVTE TOCEN DATUM');
                CLOSE;
                END
                ELSE
                  if (i=1) or (i=7) then
                    begin
                        ShowMessage('DENES E NERABOTEN DEN NE
                    MOZETE DA PRISTAPITE   NA APLIKACIJATA');
                       Close;
                    end
                   else
                       ShowMessage('DOBRODOJDOVTE');
end;  end.
```

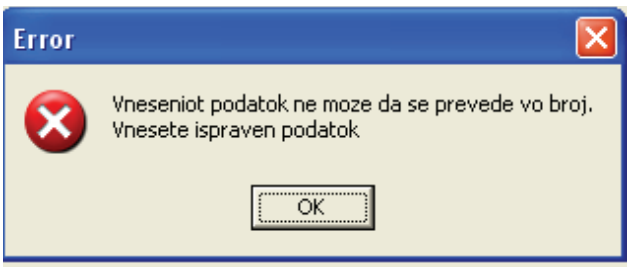
4.2. MessageDlg

Овој дијалог - прозорец личи на претходниот , со тоа што натписот во насловот припаѓа на одредено множество Windows натписи.



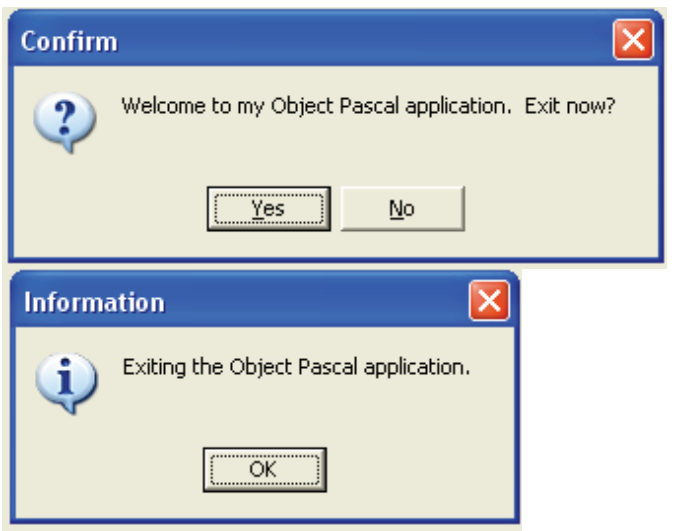
Слика 4.2

Во линијата за наслов се јавува зборот Information . Освен него, во линијата за натпис може да се јават следните зборови: Warning (жолт знак за предупредување), Error (црвена икона на знак стоп), Confirmation (прашалник), Custom (не содржи икона).



Слика 4.3

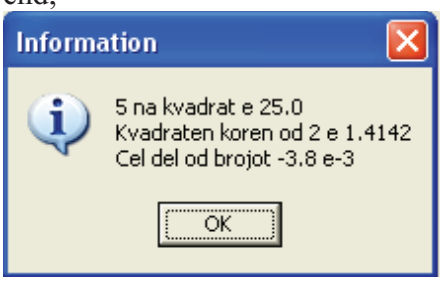
Бројот на копчиња за напуштање на дијалог - прозорецот може да биде различен.



Слика 4.4

Задача2: Да се направи апликација која во текот на извршувањето ќе даде вредности за некои елементарни функции, како што се квадрат на бројот 5 , корен од 2 и цел дел од бројот -3.8.
На формата поставуваме копче за кое ќе го напишеме следниот програмски код:

```
Procedure TForm1.Button1Click(Sender: TObject);
var
s,t:string;
begin
      str(sqr(5.0):3:1,t);
      s:='5 na kvadrat e ' + t + #13;
      str(sqrt(2.0):5:4,t);
      s:=s+'Kvadraten koren od 2 e ' + t + #13;
      str(trunc(-3.8),t);
      s:=s+'Cel del od brojot -3.8 e'+t;
      MessageDlg(s,mtInformation,[mbOk],0);
end;
```



Слика 4.5.

Аргументите на функцијата MessageDlg го имаат следното значење: s – го претставува текстот кој ќе се појави во дијалог - прозорецот, mtInformation го претставува текстот во насловната лента и знакот – иконата во прозорецот, [mbOk] е името на копчето кое ќе се појави во дијалог - прозорецот, а нулата му служи на Delphi како индикатор.

Со следната наредба во Delphi се добива дијалог - прозорец од слика 4.3:

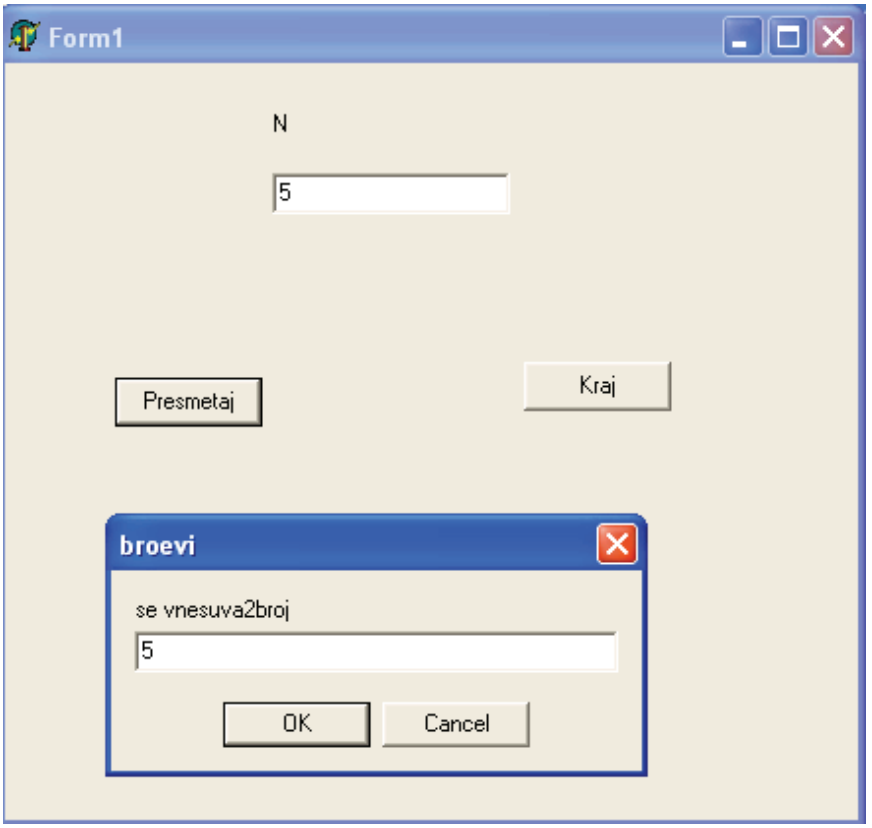
MessageDlg('Vneseniot podatok ne moze da se prevede vo broj. '+#13+ 'Vnesete ispraven podatok', mtError,[mbOk],0);

Со следната процедура во Delphi се добива дијалог - прозорец од слика4.4:

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  if MessageDlg('Welcome to my Object Pascal application.  Exit now?',
   mtConfirmation, [mbYes, mbNo], 0) = mrYes then
 begin
   MessageDlg('Exiting the Object Pascal application.', mtInformation,
    [mbOk], 0);
   Close;
 end;
end;
```

Задача 3: Да се направи апликација со која за произволен број на влезни броеви ќе се пресмета сумата.

Упатство: Се внесува N – број на броеви кои ги собираме, а потоа броевите се внесуваат во дијалог - прозорец во рамките на некој циклус.



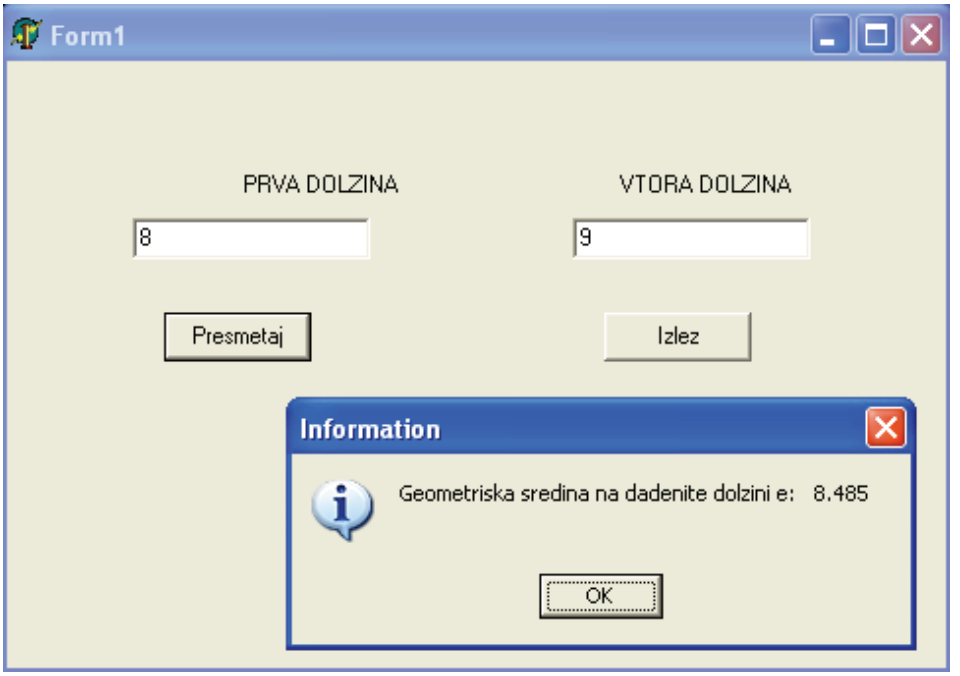
Слика 4.6.

```
Procedure TForm1.Button1Click(Sender: TObject);
        var
        s,x:real;
        t,p:string;
        n,i,j:integer;
begin
        val(edit1.text,n,j);
        if(j<>0)or(n<=0)then
            begin
                messageDlg('Podatokot ne e dobar',mtError,[mbOk],0);
                close;
            end;
            s:=0;
            i:=1;
            while i<=n do
                begin
                        str(i,p);
                        t:=inputBox('broevi','se vnesuva'+p+'-broj',' ');
                        val(t,x,j);
                        if j<>0then
                                begin
                                messageDlg('Podatokot ne e dobar',mtError,[mbOk],0);
                                close;
                                 end;
```

```
            s:=s+x;inc(i);
            end;
        str(s:8:3,p);
        showmessage('zbirot e s='+p);
        end;

procedure TForm1.Button2Click(Sender: TObject);
begin
        application.terminate;
end;
```

Наредбата t:=inputBox('broevi','se vnesuva'+p+'-broj',' '); ја користи функцијата InputBox, која содржи 3 параметри. Првиот параметар "broevi" одредува што се појавува во насловната лента, вториот параметар 'se vnesuva'+p+'broj' ја дава пораката што ќе се појави во дијалог - прозорецот, а третиот параметар ја претставува вредноста на податокот кој ќе биде внесен. Во нашиот случај тоа е празен податок – нема почетна вредност.

Задача: Да се направи апликација која определува геометриска средина на две зададени должини. Мерните броеви на дадените должини мора да се позитивни броеви.



Слика 4.7.

```
procedure TForm1.PresmetajClick(Sender: TObject);
Var
a,b,c:Real;
s:String;
i,j:Integer;
begin
  Val(edit1.text,a,i);
```

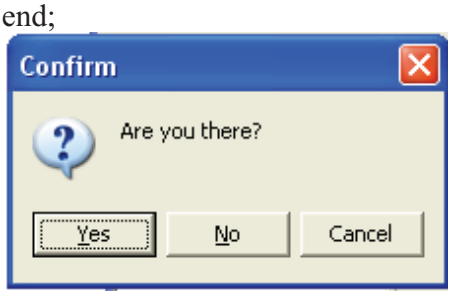
```
 Val(edit2.text,b,j);
 If (i=1) or (j=1) then
   MessageDlg('Vneseniot podatok ne e mozno da se prevede vo broj'+#13+
   'Vnesete tocen podatok', mtError,[mbOk],0);
 if (a<=0) or (b<=0) then
    MessageDlg('Merniot broj na dolzinata ne e dobar!'+#13+
    'Vnesete pozitivna vrednost', mtError,[mbOk],0)
    else
    begin
     c:=sqrt(a*b);
     str(c:8:3,s);
     MessageDlg('Geometriska sredina na dadenite dolzini e:'+s,mtInformation,
[mbOk],0);
     end;
end;
end.
```

4.3. Messagedlgpos

Го прикажува дијалог - прозорецот со порака на специфицирани координати на екранот.

Овој пример користи копче на формата. Кога корисникот кликнува на копчето, се појавува прозорец со порака со три копчиња: Yes, No, и Cancel копче на него:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  MessageDlgPos('Are you there?',mtConfirmation, mbYesNoCancel, 0, 200, 200);
end;
```



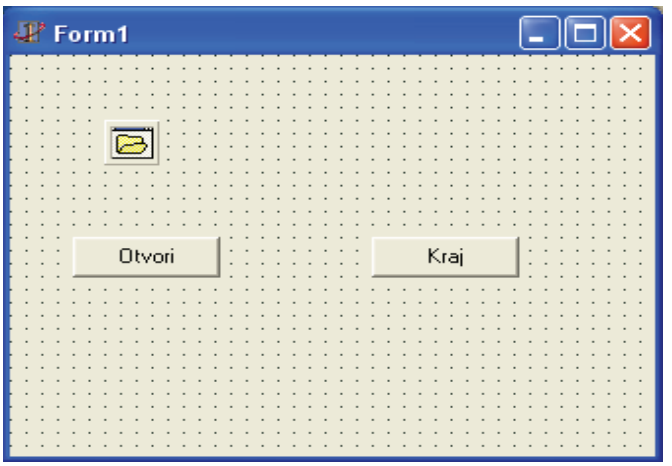
Слика 4.8.

4.4 DIALOGS компоненти

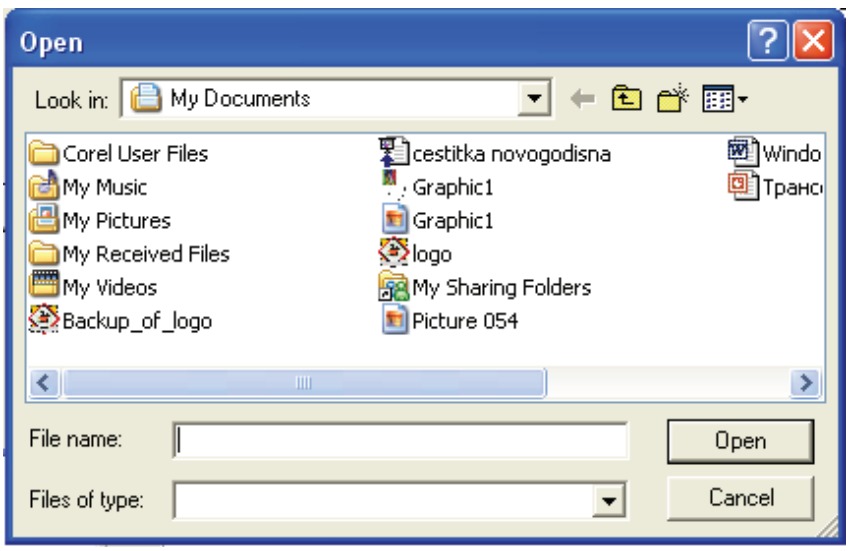
Во картичката Dialogs се наоѓаат различни компоненти за дијалог како OpenDialog, SaveDialog, ColorDialog и др. Тие се употребуваат заради унифициран пристап кон почестите операции кои се извршуваат во апликациите. Во програмите, дијалогот се активира со методот Execute. Ниеден дијалог не врши конкретна акција, туку само му овозможува на корисникот да изврши избор, а конкретните акции се реализираат со програмски код.

4.4.1. OpenDialog му овозможува на корисникот да ја одбере датотеката од дискот која сака да ја чита.

Задача1: Да се направи форма со две копчиња и OpenDialog.



Слика 4.9.

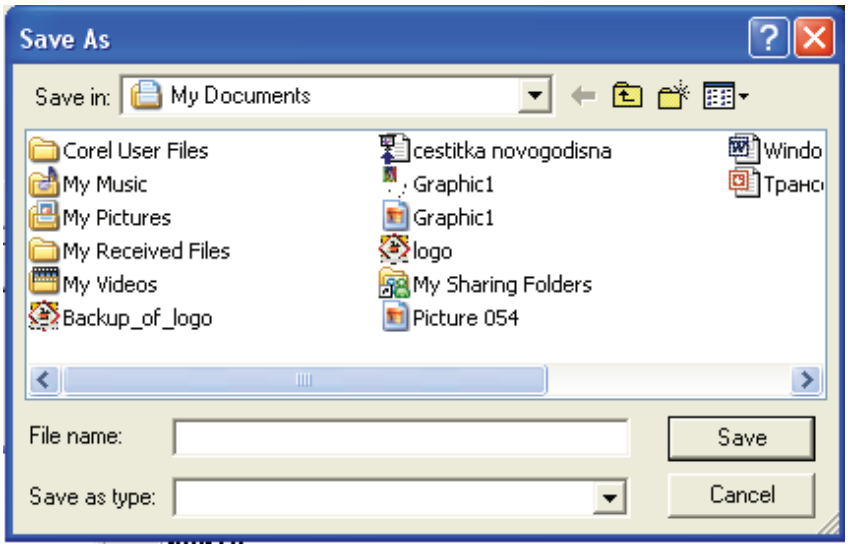


Слика 4.10.

Програмскиот код кој овозможува прикажување на OpenDialog е следниот:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    OpenDialog1.Execute;
end;
```

4.4.2. SaveDialog претставува стандарден Windows дијалог - прозорец. Наредбата со која се активира овој дијалог - прозорец е следната:
SaveDialog1.Execute;



Слика 4.11.

4.4.3. ColorDialog овозможува на стандарден начин да се одбере боја. Наредбата со која се активира овој дијалог - прозорец е следната:
ColorDialog1.Execute;

Слика 4.12.

ПОГЛАВЈЕ 4
(НАКРАТКО)

Дијалог - прозорец се користи за прикажување соодветни пораки при извршување на апликацијата.

Се користат следните видови дијалог - прозорци:

- **ShowMessage**

Тој се користи за прикажување соодветни пораки при извршување на апликацијата. Појавувањето на дијалог - прозорецот го запира извршувањето на апликацијата до неговото затворање, со кликање на копчето OK.

- **MessageDlg**

Овој дијалог - прозорец личи на претходниот, со тоа што натписот во насловот припаѓа на одредено множество Windows натписи.

Во линијата за наслов се јавува зборот Information . Освен него, во линијата за натпис може да се јават следните зборови: Warning (жолт знак за предупредување), Error (црвена икона на знак стоп), Confirmation (прашалник), Custom (не содржи икона).

- **Messagedlgpos**

Го прикажува дијалог - прозорецот со порака на специфицирани координати на екранот.

DIALOGS компоненти

OpenDialog му овозможува на корисникот да ја одбере датотеката од дискот која сака да ја чита.

SaveDialog претставува стандарден Windows дијалог - прозорец.
Наредбата со која се активира овој дијалог - прозорец е следната:
SaveDialog1.Execute;

ColorDialog овозможува на стандарден начин да се одбере боја.
Наредбата со која се активира овој дијалог - прозорец е следната:
ColorDialog1.Execute;

Прашања и задачи:

1. Наведете ги причините за вклучување на дијалог - прозорци во апликацијата.
2. Дали проблемите може да се решат без употреба на дијалог - прозорци?
3. Од што зависи насловот на дијалог - прозорецот ShowMessage?
4. Во кои случаи е пожелно да се користи дијалог - прозорецот InputBox?
5. Кои се основни делови на прозорецот InputBox?
6. Која е функцијата на Messagedlgpos?

ПОГЛАВЈЕ 5
5. КРЕИРАЊЕ НА МЕНИЈА ЗА ФОРМА

Во ова поглавје ќе стане збор за:

-опишување на системот за менија;
-користење на компонентата MAIN MENU;
-користење на компонентата POPUP MENU, SPEED MENU;
-поврзување на опциите од менито со други форми и вметнување во тековниот проект.

5.1. MainMenu

Менито е значаен дел од Windows апликациите. Се наоѓа во првиот ред под насловната лента. Се состои од повеќе зборови наречени опции и има подмени.

Во Standard компонентите се наоѓа компонентата MainMenu и неа ја поставуваме во формата. Со двојно кликнување на MainMenu, се отвора дизајнер на мени . Тој изгледа како празна форма.
За првата опција од менито треба да се променат својствата:

Name – FileMenu
Caption – &File (Alt + F)

Се појавува празно место под File и десно од него.

File	
New	
Open	
Save	
SaveAs	

Name – FileNew
Caption - & New
Претходните два чекора се повторуваат за да се креираат деловите од менито: Open, Save, SaveAs.

Се препорачува стандардизирање на менијата, односно тие да изгледаат како во другите Windows апликации.

Сепаратор е хоризонтална линија за разделување на опциите од менито. Се креира на следниот начин:

Во полето под Save As на својството Caption се става (-)

Додавање на подмени се прави на следниот начин:

Insert – Sub menu
Create – Sub menu
На десен клик Insert – Sub menu

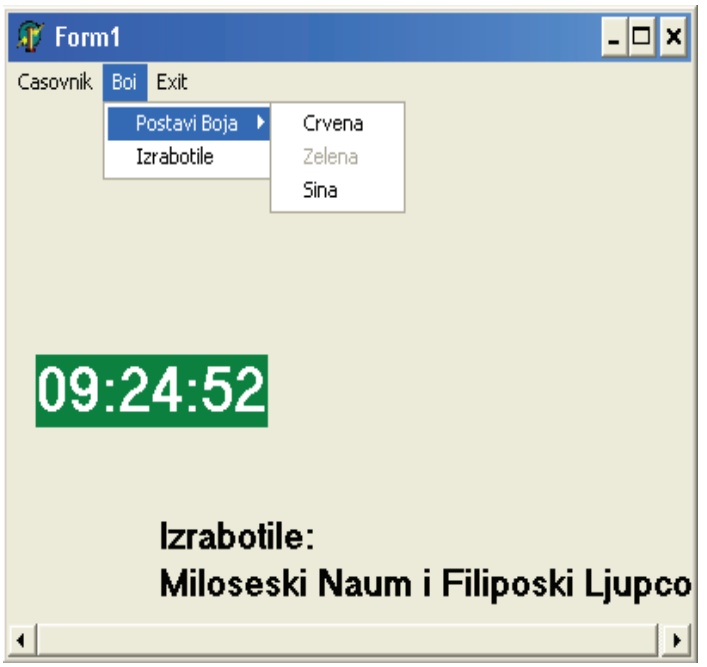
Кој било избор на опциите го активира настанот OnClick и се извршуваат наредбите во таа процедура.

Задача1: Да се направи апликација часовник во боја. Во формата ги ставаме следните компоненти: MainMenu, Timer и Label.

Кодот на програмата е следниот:

```
Procedure TForm1.MenuVremeClick(Sender:Tobject);
Begin
        Label1.caption:=TimeToStr(time);
End;
```

```
Procedure TForm1.MenuDatumClick(Sender:Tobject);
Begin
        Label1.caption:=DateToStr(Date);
End;
Procedure TForm1.Crvena1Click();
Begin
    Label1.color:=clred;
    Crvena1.enabled:=False;
    Zelena1.enabled:=True;
    Sina1.enabled:=True;
End;

Procedure TForm1.Zelena1Click();
Begin
    Label1.color:=clgreen;
    Crvena1.enabled:=True;
    Zelena1.enabled:=False;
    Sina1.enabled:=True;
End;

Procedure
TForm1.Sina1Click();
Begin
    Label1.color:=clblue;
    Crvena1.enabled:=True;
    Zelena1.enabled:=True;
    Sina1.enabled:=False;
End;
```



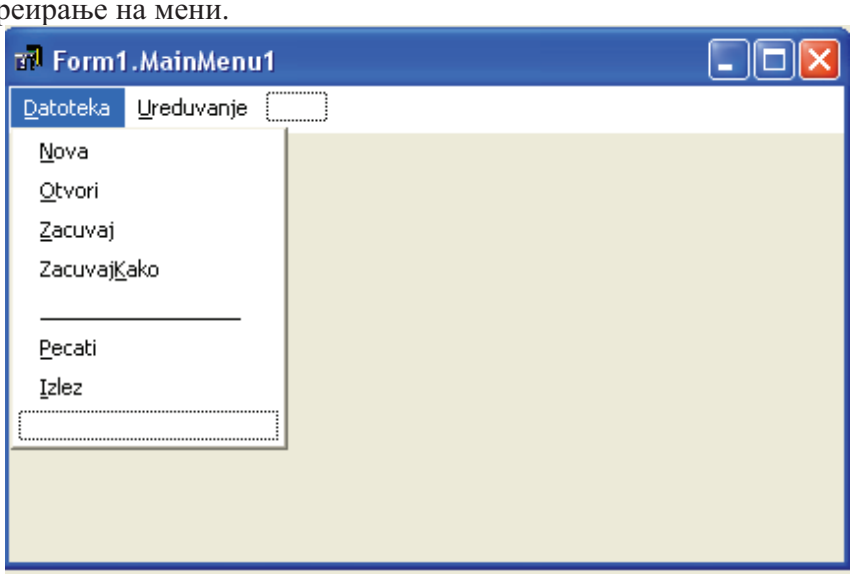
Слика 5.0.

Timer компонентата може д а се искористи за прикажување тековно време.

Задача2: Да се направи апликација која ќе работи како едноставна програма за обработка на текст. Со програмата ќе се обработуваат датотеки и текст. Кај датотеките активностите се: формирање на датотека, вчитување на постоечка, снимање, печатење. Обработката на текстот е: селектирање, сечење, копирање на дел од текстот, лепење на дел од текстот и сл.

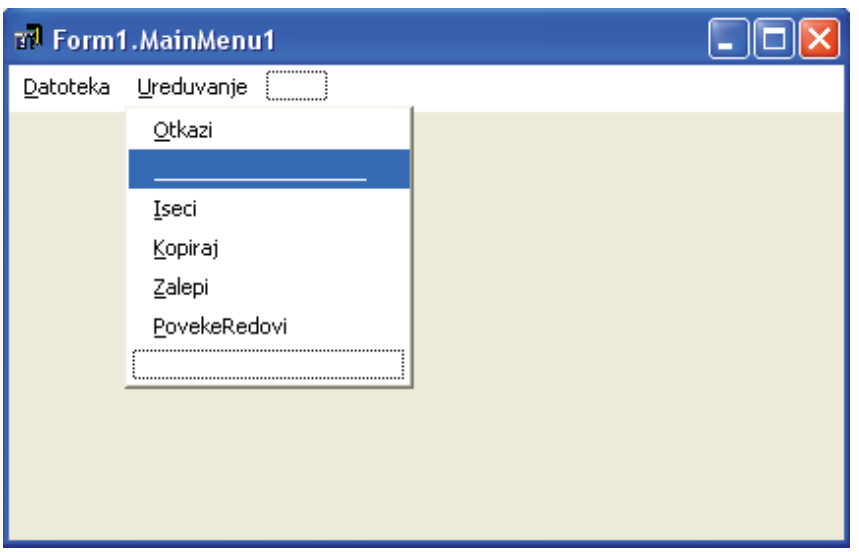
- Се отвора нова апликација.
- Се додава Memo компонента на неа.
  Се брише текстот Memo преку едиторот на текст (својство Lines)
  својство ScrollBars -> ssVertical
  поставуваме системски фонт Fixedsys
  Align->alClient со што Memo компонентата ја зафаќа целата форма.
- Во формата се додава компонента MainMenu.

- Креирање на мени.



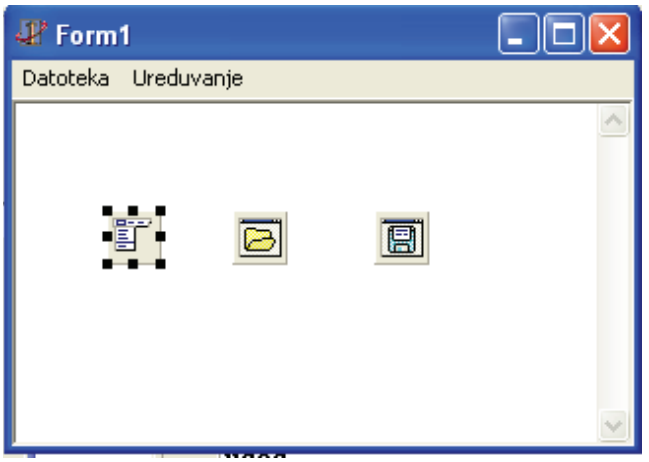
Слика 5.1.

- Додавање кратенки на опциите од мени:
  -да се обележи Нова во опцијата Датотека;
  -во својството ShortCut да се одбере Ctrl+N.
- Десно од датотека треба да се креира опцијата Уредување од менито.



Слика 5.2.

- Додавање на опции во подменито се врши со кликање на копчето Insert, а бришење со Delete.
- Недостапност на опцијата се врши со својството Enabled – False.
- Во менито додаваме компоненти од дијалог - картичката и тоа OpenDialog, SaveDialog.



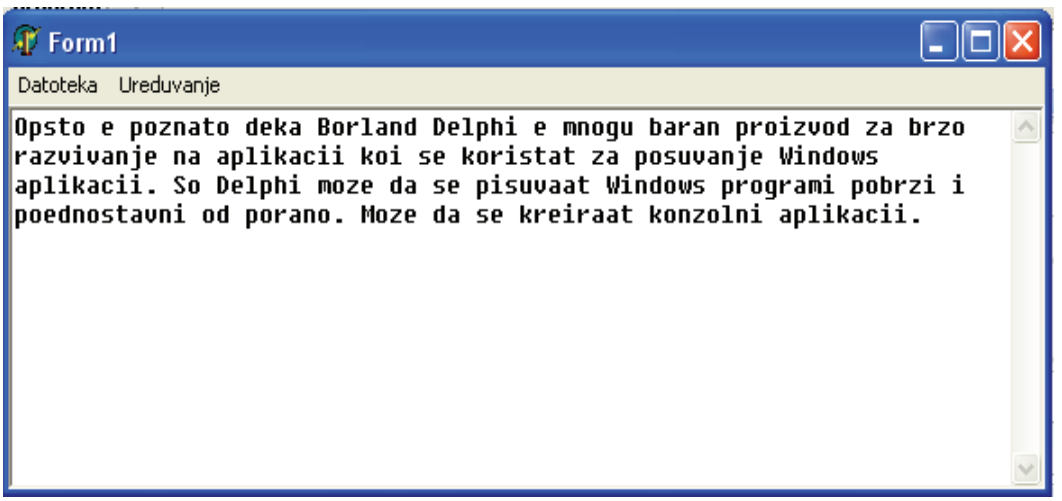
Слика 5.3.

```
procedure TForm1.Izlez1Click(Sender: TObject);
begin
close;
end;

procedure TForm1.Nova1Click(Sender: TObject);
Var res:integer;

begin
  if memo1.modified then
    begin
     Res:=Application.MessageBox('Tekovnata datoteka e promeneta. Zacuvaj
promeni?',
     'Tekst1Message',MB_YESNOCANCEL);
     If res=IDYES then Zacuvaj1Click(Sender);
     If res=IDCANCEL then exit;
     end;
     if memo1.Lines.count>0 then memo1.clear;
     savedialog1.filename:=' ';
end;

procedure TForm1.Zacuvaj1Click(Sender: TObject);
begin
       if savedialog1.filename<>' ' then
       begin
       memo1.lines.savetofile(savedialog1.filename);
       memo1.modified:=false;
       end
       else
         zacuvajkako1click(sender);
end;

procedure TForm1.Otvori1Click(Sender: TObject);
var res:Integer;
begin
      if memo1.modified then
        begin
```

```
        Res:=Application.MessageBox('Tekovnata datoteka e promeneta.
Zacuvaj promeni?',
    'Tekst1Message',MB_YESNOCANCEL);
    If res=IDYES then Zacuvaj1Click(Sender);
    If res=IDCANCEL then exit;
    end;
    opendialog1.filename:='FileName ';
    if opendialog1.execute then
     begin
         if memo1.lines.count>0 then memo1.clear;
         memo1.lines.loadfromfile(opendialog1.filename); {otvoranje}
         savedialog1.filename:=opendialog1.filename;
     end;

end;

procedure TForm1.ZacuvajKako1Click(Sender: TObject);
begin
savedialog1.title:='Zacuvaj kako';
if savedialog1.execute then
  begin
    memo1.lines.savetofile(savedialog1.filename); {snimanje}
    memo1.modified:=false;
   end;
end;

procedure TForm1.Ponisti1Click(Sender: TObject);
begin
   SendMessage(memo1.handle,WM_UNDO,0,0);
end;

procedure TForm1.Preseci1Click(Sender: TObject);
begin
   memo1.CutToClipboard;
end;


procedure TForm1.Kopiraj1Click(Sender: TObject);
begin
   memo1.CopyToClipboard;
end;

procedure TForm1.Zalepi1Click(Sender: TObject);
begin
   memo1.PasteFromClipboard;
end;

procedure TForm1.PovekeRedovi1Click(Sender: TObject);
begin
      memo1.wordwrap:=not memo1.WordWrap;
      povekeredovi1.checked:=memo1.WordWrap;
```

```
        if memo1.WordWrap then memo1.scrollbars:=ssVertical
        else
         memo1.scrollbars:=ssboth;
end;

procedure TForm1.ObeleziSe1Click(Sender: TObject);
begin
          memo1.selectAll;
end;
end.
```

Текстот напишан со оваа апликација е прикажан на слика 5.4.



Слика 5.4.

Процедурата за излез овозможува завршување на работата и излез од апликацијата, но не овозможува корисникот при излегување од апликацијата да биде опоменат дека во датотеката се извршени некои промени после последното снимање.

Процедурата Нова отвора нова датотека, но претходно се проверува дали во тековната датотека е извршена промена по последното снимање и на корисникот му се нуди можност промените да ги зачува.

Процедурата Отвори ја отвора одбраната датотека од одбраниот фолдер. Претходно се врши проверка на тековната датотека дали се снимени последните промени. Програмскиот код содржи OpenDialog и LoadFromFile.

Процедурата Зачувај извршува снимање -чување на датотеката. Оваа опција треба да се користи ако на датотеката веќе и е доделено име. Во спротивно се бира опцијата Зачувај Како.

Уредувањето на текстот се врши со следните неколку процедури

Процедурата Поништи ги поништува претходните активности со кои е извршена некоја промена во датотеката. Бидејќи Delphi нема таква можност, се користи Windows наредбата WM_UNDO,0,0.

Процедурата Обележи овозможува обележување на сé што е дотогаш внесено во датотеката.

Процедурата Исечи го пренесува обележениот дел од текстот во привремена меморија Clipboard и воедно го отстранува од датотеката.

Процедурата Копирај работи исто како Исечи, само што обележениот дел не го отстранува од датотеката.

Процедурата Залепи ја враќа содржината од привремената меморија во датотеката на местото на покажувачот.

Процедурата Повеќе редови овозможува запишување на текстот во датотеката во повеќе редови. Оваа опција треба да биде чекирана, а потоа потребен е само вертикален лизгач, инаку, потребни се двата лизгачи.

5.2.PopupMenu

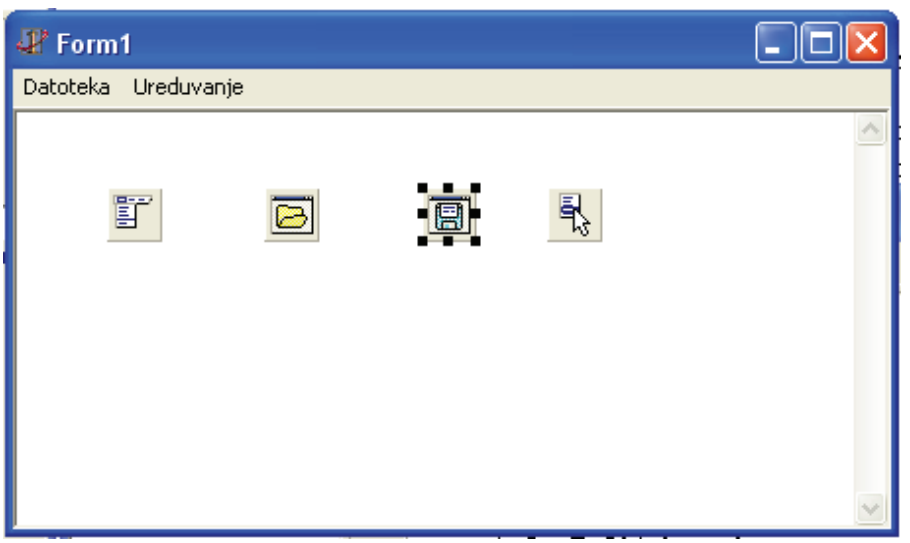
Во Delphi може да се користи помошно мени (PopupMenu) . Креирањето на ова мени е слично со начинот на креирање на главното мени. Ефектот на доделување на ова мени е она множество на опции што се појавуваат при десен клик на копчето од глувчето на некоја компонента или текст. Пишувањето на кодот за ова мени е слично со пишувањето на кодот од главното мени.

Ќе ја искористиме претходната апликација за додавање можност за користење на помошно мени. Во програмите за обработка на текст, вообичаено е кога некој дел од текстот ќе се означи и на него ќе се кликне со десен клик , помошното мени овозможува сечење, копирање и лепење на текстот.

За да може во главното мени да го вклучиме и помошното мени, кое ќе се однесува на опцијата Уредување, се оди по следната постапка:

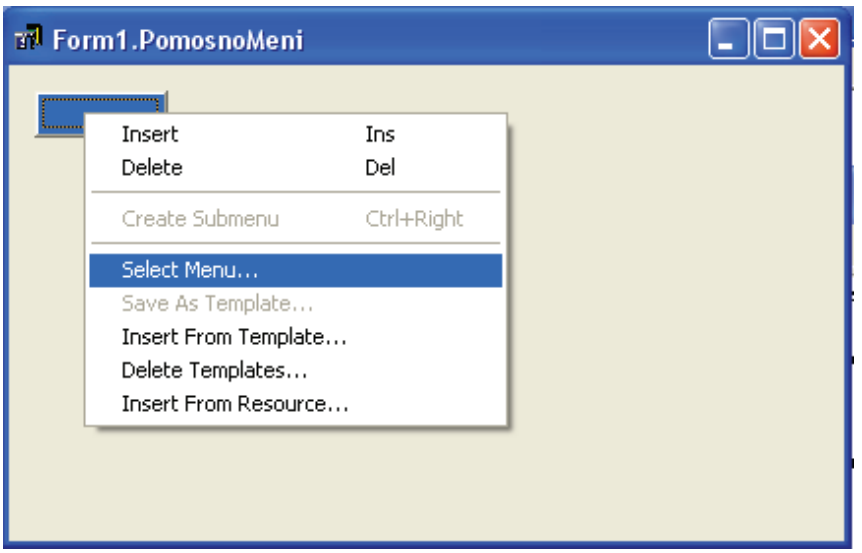
1. Ја вчитуваме апликацијата која се однесува на главното мени, бидејќи на неа се врши надградба.
2. Компонентата PopupMenu ја поставуваме во формата.
3. Името и го менуваме во PomosnoMeni.

Формата го има изгледот како на слика 5.5.



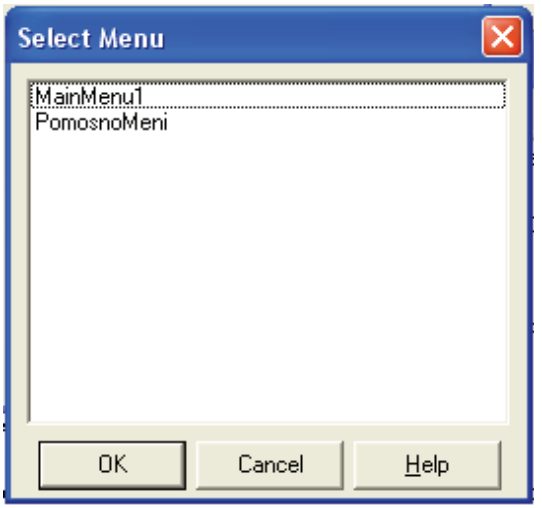
Слика 5.5.

4. Двапати кликаме на оваа компонента за да го стартуваме дизајнерот на мени. Потоа се појавува прозорец со син правоаголник, во неговиот горен лев агол. Ова мени може да се креира на истиот начин, како и главното мени, а може и на друг начин, при што ќе се искористат опциите кои се креирани во главното мени.
5. Со десен клик на синиот правоаголник го повикуваме помошниот мени дизајнер. Од подменито одбираме Select Menu (слика 5.6.), по што се појавува прозорец кој ги прикажува менијата, достапни во апликацијата слика5.6.



Слика 5.6.

6. Го одбираме MainMenu1 и копчето OK.
7. Во добиениот прозорец ја одбираме опцијата Ureduvanje, во која ќе ги одбереме подопциите Iseci, Kopiraj и Zalepi.
8. Ги пренесуваме одбраните опции во помошната меморија со постапка на копирање од Delphi програмата. (Ctrl+C).



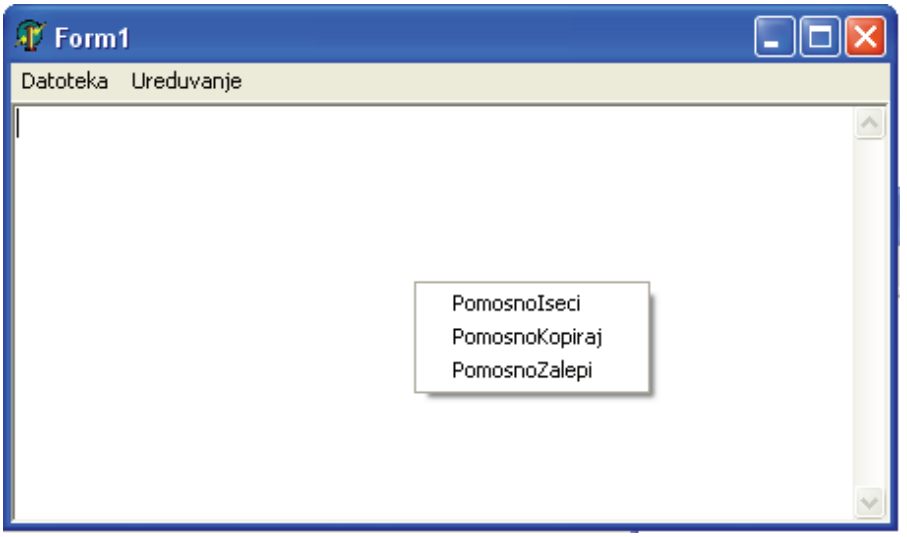
Слика 5.7.

9. Потоа одбираме SelectMenu, а во добиениот прозорец одбираме PomosnoMeni и копче OK. Се прикажува празно помошно мени.
10. Ја пренесуваме содржината на привремената меморија, со што сме ги ископирале опциите за помошното мени.
11. Ги менуваме имињата на внесените опции на помошното мени во : PomosnoIseci, PomosnoKopiraj и PomosnoZalepi.

Треба да се напише програмски код за управувачите со настани за опциите на помошното мени. Работата може да ни се олесни, затоа што веќе имаме напишано програмски кодови за истите настани во главното мени. Имаме можност за истите настани да користиме заеднички управувачи на настани. Постапката за поврзување на опциите на помошното мени со постоечките управувачи на настани ја вршиме на следниот начин:

12. Во дизајнерот на мени одбираме опција Isechi.
13. Во ObjectInspector бираме картичка Events.
14. Кликнуваме на копчето со стрелка надолу , кое се наоѓа десно од настанот OnClick. Се прикажува листа на процедури кои се досега креирани во апликацијата.
15. Од понудената листа треба да се одбере UreduvanjeIsechi . На овој начин сме го поврзале настанот со веќе постоечкиот програмски код.
16. Чекорите од 12 до 15 треба да се повторат за преостанатите две опции на помошното мени.
17. За да се заврши креирањето на помошното мени, на формата ја селектираме компонентата Memo и својството PopupMenu го менуваме во PomosnoMeni.

Со ова е завршена постапката на креирање на помошно мени. Ако ја стартуваме апликацијата и во прозорецот за текст кликнеме со десен клик на глувчето, ќе се добие прозорецот од Слика 5.8.



Слика 5.8. Изглед на апликација со помошно мени.

ПОГЛАВЈЕ 5
(НАКРАТКО)

Менито е значаен дел од Windows апликациите. Се наоѓа во првиот ред под насловната лента. Се состои од повеќе зборови, наречени опции и има подмени.

Во Standard компонентите се наоѓа компонентата **MainMenu**, неа ја поставуваме во формата. Со двојно кликнување на MainMenu се отвора дизајнер на мени .

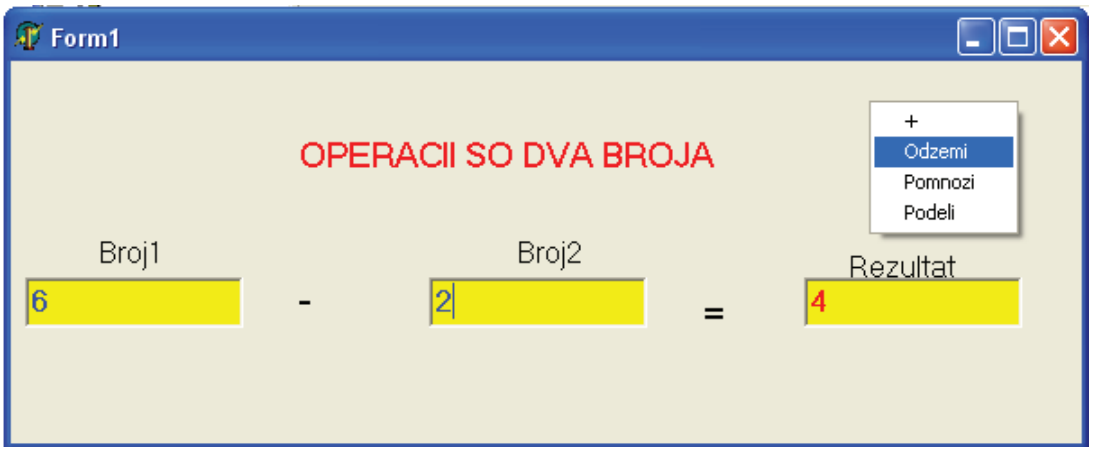
PopupMenu - помошно мени. Ефектот на доделување на ова мени е она множество на опции што се појавуваат при десен клик на копчето од глувчето на некоја компонента или текст.

Вежба

Операции со броеви (PopupMenu компонента)

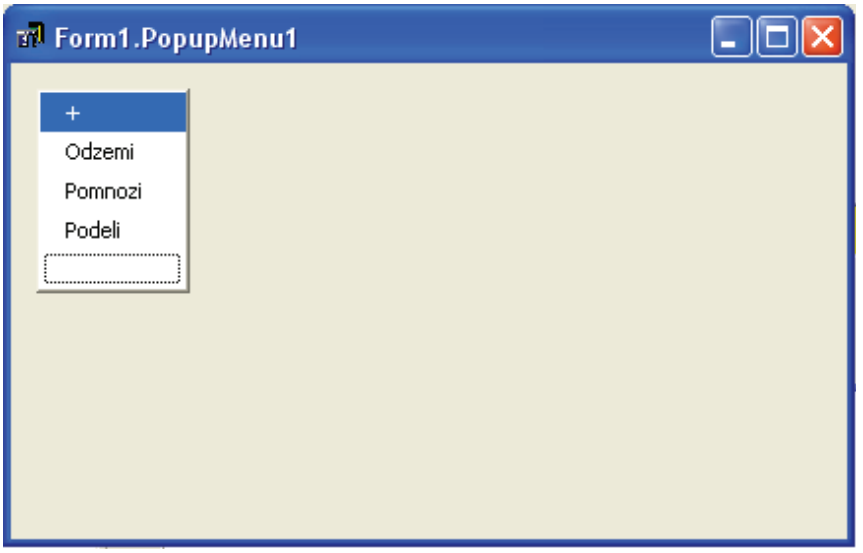
Да се напише програма со која се вршат операциите собирање, одземање, множење и делење на два броја. На формата да се дефинираат соодветни Edit полиња (жолта позадина со сини бројки) за внесување на broj1 и broj2, додека за резултатот исто така edit поле (жолта позадина со црвени бројки) . На формата да се постави PopuupMenu во кое се дефинирани операциите Собери, Одземи, Помножи и подели.

Изгледот на формата на решението на оваа задача е прикажан на слика5.9..



Слика 5.9.

Формата ги користи стандардните компоненти **Label**, **Edit** и компонентата **PopupMenu,** која се активира на ниво на формата со клик со десното копче од глувчето. За да се додадат опциите на менито, треба двојно да се кликне на **Items** во својствата на **PopupMeni** со чија помош се додаваат опциите на менито како што е прикажано на слика 5.10.



Слика 5.10.

Главниот дел на програмскиот код е следниот:

```
procedure TForm1.Kontrola_na_prv_broj(Sender: TObject);
Var s:string;
  n:real;
  gr: integer;
Begin
  s:=Edit1.Text; Val(s,n,gr);
  If gr <> 0 Then Begin
        ShowMessage('Ne e dobro vnesen brojot 1');
        Edit1.SetFocus; Edit1.SelectAll;
  End
end;
procedure TForm1.Kontrola_na_vtor_broj(Sender: TObject);
Var s:string;
  n:real;
  gr: integer;
Begin
  s:=Edit2.Text; Val(s,n,gr);
  If gr <> 0 Then Begin
          ShowMessage('Ne e dobro vnesen brojot 2');
          Edit2.SetFocus; Edit2.SelectAll;
       End
end;
procedure TForm1.Soberi(Sender: TObject);
begin
     Edit3.Text:=FloatToStr(StrToFloat(Edit1.Text)+StrToFloat(Edit2.Text));
     Label2.Caption:='+';
end;
```

```
procedure TForm1.Odzemi(Sender: TObject);
begin
   Edit3.Text:=FloatToStr(StrToFloat(Edit1.Text)-StrToFloat(Edit2.Text));
   Label2.Caption:='-';
end;

procedure TForm1.Pomnozi(Sender: TObject);
begin
   Edit3.Text:=FloatToStr(StrToFloat(Edit1.Text)*StrToFloat(Edit2.Text));
   Label2.Caption:='X';
end;

procedure TForm1.Podeli(Sender: TObject);
begin
   Label2.Caption:='/';
   If(StrToFloat(Edit2.Text))=0 Then Begin
        ShowMessage('Delenje so nula');
        Edit2.SetFocus;
        Edit2.SelectAll
  End
  Else
      Edit3.Text:=FloatToStr(StrToFloat(Edit1.Text)/StrToFloat(Edit2.Text));
  end;
end.
```

Да ги објасниме главните карактеристики на овој код:

Содржината на **Edit** полето е string променлива, па неопходно е да се проконтролира дали е внесен нумерички податок во полето за број. Тоа се контролира со процедурите **Kontrola_na_prv_broj** и **Kontrola_na_vtor_broj**. Во настаните (**Events**) **onExit** ги внесуваме имињата на процедурите (настанот се случува по напуштањето на полето). Во процедурата на променливата **s** и доделуваме вредност на содржината на полето (**s:=Edit1.Text**), а потоа ја повикуваме процедурата **Val** која string променливата ја претвора во нумеричка. Во случај да string-от не е нумерички, третиот параметар на таа процедура **gr** ќе има вредност различна од **0**. Во тој случај се испраќа порака за грешка **ShowMessage**, се враќаме повторно на тоа поле со методот **SetFocus,** а неговата содржина се селектира со методот **SelectAll**.

Во процедурите **Soberi**, **Odzemi**, **Pomnozi** и **Podeli** ја вршиме соодветната аритметичка операција и го прикажуваме добиениот резултат. За да ја извршиме соодветната операција, неопходно е прво содржината на полето која е од тип string да се претвори во нумеричка променлива со помош на функцијата **StrToFloat** и на крај, добиениот резултат да се претвори во string со функцијата **FloatToStr**. Исто така, во полето **Label2** го прикажуваме знакот од математичката операција нпр. за собирање: **Label2.Caption:='+';**

Кога е во прашање операцијата делење, неопходно е да се испита дали делителот е еднаков на 0, па ако е, прикажуваме соодветна порака за грешка.

ПОГЛАВЈЕ 6
6. ПРОЦЕДУРИ (PROCEDURE) И ФУНКЦИИ (FUNCTION) ВО DELPHI

Во ова поглавје ученикот ќе научи:
-да ги објаснува процедурите-**procedure**;
-да ги објаснува функциите-**function;**
-да ги интерпретира разликите помеѓу процедурите и функциите;
-да пишува заглавие за процедури и функции;
-да доделува и пренесува параметри кон функции;
-да ја познава постапката за употреба на процедури;
-да пишува процедури за одделни проблеми;
-да ја познава постапката за употреба на функции;
-да пишува функции за проблеми што се повторуваат;
-да решава примери и задачи од сложени типови на податоци;
-да применува функции за трансформација на реален број во стринг и обратно.)

6.1. Потпрограми

Потпрограмите се механизми за разбивање на сложените програмски системи на помали логички целини. Сложените обработки се остваруваат со комбинирање на такви потпрограми.

Потпрограмите претставуваат независни програмски целини кои имаат свои влезни податоци и даваат излезни резултати.

Структурата на потпрограмата е слична на структурата на главната програма. Општ облик на потпрограма е:

```
ZaglavieNaPotprograma
Blok
;
```

ZaglavieNaPotprograma го одредува типот на потпрограмата (функција или процедура) , името на потпрограмата и аргументите за прифаќање на податоци од повикувачката програма (главна програма или друга потпрограма) и за враќање на резултатите од обработката.

Blok има иста структура, како и главната програма.

6.2. Дефинирање на потпрограмата

Постојат два типа потпрограми: функции и процедури.

Функциите се потпрограми кои врз основа на своите аргументи даваат еден резултат. Тој резултат се нарекува *вредност на функцијата*.

Процедурите се потпрограми, кои врз основа на своите аргументи можат да дадат повеќе резултати. Тие резултати се исто така аргументи на процедурата.

Општ облик на заглавието на функцијата и процедурата е :

```
Function ImeNaFunkcija(Arg; … ;Arg):TipNaRezultat;
Procedure ImeNaProcedura (Arg; … ;Arg);

Arg =       ImeArg, ImeArg, … , ImeArg, ImeArg: TipArg;     или
      var   ImeArg, ImeArg, … , ImeArg, ImeArg: TipArg;     или
      out   ImeArg, ImeArg, … , ImeArg, ImeArg: TipArg;     или
      const ImeArg, ImeArg, … , ImeArg, ImeArg: TipArg;
```

каде што ImeNaFunkcija , ImeNaProcedura се идентификатори за именување на потпрограмата. TipNaRezultat е идентификатор на типот на резултатот на

функцијата, ImeArg – идентификатор за именување на аргументите, TipArg – идентификатор на типот на аргументот.

Ако потпрограмата нема аргументи, треба да се изостават и заградите.

Аргументите во заглавието на потпрограмата се нарекуваат **формални аргументи,** кои при повикување на потпрограмата ќе бидат заменети со соодветни **вистински аргументи.**

Пример 1: Збир на првите N природни броеви.

I начин:

```
Function Suma (N:Integer):Integer;
Var
    I: Integer;
Begin
    Result:=0;
    For I:=1 to N do Result:=Result+I;
End;
```

Секоја функција на јазикот Object Pascal има локална променлива со име Result. Оваа променлива скриено ја декларира преведувачот , а се користи да ја зачува вредноста која ја враќа фукцијата. На променливата Result и се доделува вредност во рамките на функцијата.

II начин:

```
Function Suma (N:Integer):Integer;
        Var
            I: Integer;
        Begin
            Suma:=0;
            For I:=1 to N do
                Suma:=Suma+I;
        End;
```

Вредноста која ја враќа функцијата и се доделува на променливата Suma. Во телото на функцијата мора да постои наредба за доделување вредност на името на функцијата.

Пример 2: Да се направи процедура со која ќе се пресметува аритметичка средина на првите N природни броеви.

```
Procedure Sredna Vrednost;
   Var I,zbir:Integer;
        Begin
    zbir:=0;
      For I:=1 to N do
          zbir:=zbir+I;
     Sred:=zbir/N;
   End;
```

Пример 3: Да се направи процедура за збир и производ на првите N природни броеви:

```
Procedure ZbirProizvod (N:Integer; var S, P: Integer);
   Var I:Integer;
    Begin
```

```
        S:=0;
    P:=0;
        For I:=1 to N do
        begin
          S:=S+I;
          P:=P*I;
        End;
  End;
```

6.3. Повикување на потпрограмите

ZaglavieNaPotprograma треба да се дефинира во делот Interface.

Целосна дефиниција на потпрограмата треба да имаме во делот Implementation.

Потпрограмите се повикуваат со нивното име во главната програма. Зад името на потпрограмата се наведуваат вистинските аргументи. Вистинските аргументи по број и по тип мора да се исти со формалните аргументи.

Постојат потпрограми кои се повикуваат самите себеси и се нарекуваат **рекурзивни** потпрограми. Тие овозможуваат лесно и елегантно решение на проблемите кои по природа се рекурзивни.

Пример 4: За рекурзивната дефиниција на факториел: n! =n*(n-1)! , при што 0! = 1, може да се напише следната рекурзивна функција:

```
function fact (n:integer):longint;
  begin
    if n>0 then fact:=n*fact(n-1) {rekurzivno povikuvanje na funkcijata fact}
           else fact:=1; {tocka na terminiranje}
end;
```

6.4. ScrollBar – Контрола . Задача со функција

Оваа контрола се користи за внесување на некои бројни вредности меѓу зададените гранични вредности. Може да биде хоризонтално или вертикално поставена.

Својства:

Kind – прикажува од кој тип е лизгачот (хоризонтален или вертикален);

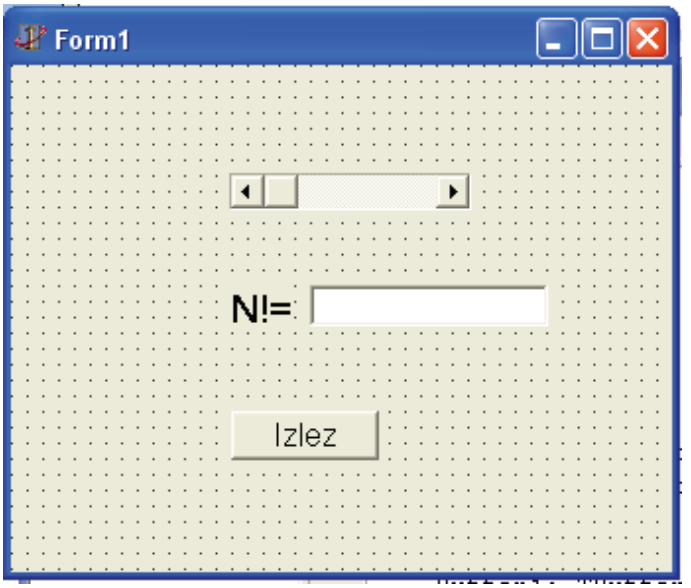
Min,Max (Integer) – ги одредува граничните вредности кои можат да се претставуваат со лизгачот;

Position (integer) – ја одредува моменталната положба на лизгачот;

Настани:

On Scroll – настапува кога се движи лизгачот на ScrollBar .

Пример 5: Да се направи програма во Delphi која ќе пресметува N! = N*(N-1)*(N-2)*…*1 , при што N се внесува со лизгач.



Слика 6.1.

```
implementation

{$R *.dfm}

function fact (n:integer):longint;
begin
        if n=0 then fact:=1
        else fact:=n*fact(n-1);
end;
Procedure TForm1.ScrollBar1Change(Sender:TObject );
Begin
        Edit1.text:=IntToStr(fact(ScrollBar1.position));
        Label1.Caption:=IntToStr(ScrollBar1.position)+'!=';
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
        application.terminate;
end;
end.
```

Пример 6: Да се направи програма за конверзија ЕВРО – ДЕНАР со помош на ScrolBar компонентата.

Пример 7:Да се направи програма со која ќе се внесуваат два операнди во две Edit кутии и врз нив ќе се извршуваат основните аритметички операции: собирање, одземање, множење и делење. Операторот се бира со RadioButton. Проблемот да се реши со потпрограма.

```
Procedure Operacii(op1,op2:Real;operator:char;Var Rez:Real);
    Begin
        Case operator of
          '+':Rez:=op1+op2;
          '-':Rez:=op1-op2;
          '*':Rez:=op1*op2;
          '/':Rez:=op1/op2;
        End;
```

```
      End;
Procedure TForm1.Button1Click(Sender: TObject);
begin
        a:=strtofloat(edit1.text);
        b:=strtofloat(edit2.text);
        If radiobutton1.checked then o:='+';
        If radiobutton2.checked then o:='-';
        If radiobutton3.checked then o:='*';
        If radiobutton4.checked then o:='/';
        Operacii(A,B,o,C); {povikuvanje na procedurata Operacii}
   edit4.text:=floattostr(c);
end;
end.
```

Пример 8: Да се направи програма за наоѓање најмал елемент во низата (A(i)n,(n=10)) и неговиот реден број.

```
        type
         Niza=Array[1..20] of integer;          {prijavuvanje sopstven tip}
          TForm1 = class(TForm)
            Memo1: TMemo;
            Button1: TButton;
            Edit1: TEdit;
            procedure Button1Click(Sender: TObject);
            procedure FormActivate(Sender: TObject);
          private
            { Private declarations }
          public
            { Public declarations }
          end;
        Var                         {del za prijavuvanje globalni varijabli}
          Form1: TForm1;
          a:array[1..30] of integer;
        implementation

        {$R *.DFM}
        Procedure najmal(a:niza; n:byte; var min:integer; var rbr:byte);
        Var i:byte;
        Begin
          Min:=a[1];
          Rbr:=1;
          For i:=2 to n do
            If a[i]<min then
             Begin
                Min:=a[i];
                Rbr:=I;
             End;
        End;

        Procedure TForm1.Button1Click(Sender: TObject);
         var
                i,j,n,gr,min:integer;
```

```
            s:string;
            Pole:niza;
            br:byte;
      begin
            for i:=0 to 9 do
            begin
                  s:=memo1.lines[i];
                  val(s,n,gr);
                   j:=i+1;
                   if gr=0 then a[j]:=strtoint(memo1.lines[i]);
            end;
            Najmal(pole,10,min,br);        {povikuvanje procedura}
          edit1.text:=Inttostr(min);
 end;



procedure TForm1.FormActivate(Sender: TObject);
Var i:integer;
begin
  for i:=1 to 10 do
  a[i]:=0;
end;
end.
```

Пример 9: Да се направи потпрограма, која даденото време изразено во секунди ќе го претвори во часови, минути и секунди.

```
Procedure Vreme(sek:integer; Var C,M,S:Integer);
 Begin
        S:=sek mod 60;
        Sek:=sek div 60;
        M:=sek mod 60;
        Sek:=sek div 60;
        C:=sek ;
End;
```

Пример 10: Да се направи процедура во која ќе се формираат два стринга, едниот само од самогласките, а другиот од согласките на даден стринг.

```
Procedure Bukvi(s:string; var samo,sogl:string);
var
        dol,i:byte;
        azbuka, samog, soglaski:set Of char;
begin
        dol:=length(s);
        azbuka:=['a'..'z','A'..'Z'];
        samog:=['a','e','i','o','u','A','E','I','O','U'];
        soglASKI:=azbuka-samog;
        samo:=' ';
        sogl:=' ';
```

```
for i:=1 to dol do
  begin
   if s[i] in samog
     then
     samo:=samo+s[i] ;
   if  s[i] in soglaski
     then
     sogl:=sogl+s[i]  ;
  end;
 end;
```

Пример 11 : Да се направи програма во која ќе се пресметува степен (експонент) X^n.

```
Function   XnaN(N:integer ; X:Real):Real;
          Var   i:Integer ; P:Real;
  Begin
    P:=1;
      For  i:=1 to N do
        P:=P*X;
      XnaN:=P;
    End;
procedure TForm1.Button1Click(Sender: TObject);
         Var  N:Integer;
           A,Rezultat :Real;
Begin
      a:=strtofloat(edit1.text);
      n:=strtoint(edit2.text);
      Rezultat :=XnaN(N,a);
      edit3.text:=floattostr(rezultat);
end;
end.
```

Слика 6.2

ПОГЛАВЈЕ 6
(НАКРАТКО)

Потпрограмите се механизми за разбивање на сложените програмски системи на помали логички целини. Сложените обработки се остваруваат со комбинирање на такви потпрограми.

Функциите се потпрограми кои врз основа на своите аргументи даваат еден резултат. Тој резултат се нарекува *вредност на функцијата*.

Процедурите се потпрограми кои врз основа на своите аргументи можат да дадат повеќе резултати. Тие резултати се исто така аргументи на процедурата.

Аргументите во заглавието на потпрограмата се нарекуваат **формални аргументи** кои при повикување на потпрограмата ќе бидат заменети со соодветни **вистински аргументи.**

Потпрограмите се повикуваат со нивното име во главната програма. Зад името на потпрограмата се наведуваат вистинските аргументи. Вистинските аргументи по број и по тип мора да се исти со формалните аргументи.

Постојат потпрограми кои се повикуваат самите себеси и се нарекуваат **рекурзивни** потпрограми.

Прашања и задачи:

1. Напиши Delphi процедура наречена *Multiply*, која прифаќа два integer броја, *number1* и *number2*, а го печати резултатот од множењето на двата броја.

а)
```
procedure Multiply( number1, number2 : integer );
  var  Result : Integer;
  begin
        Result := number1 * number2;
        Edit1.text:= inttostr( Result )
  end;
```

б)
```
procedure Multiply( number1, number2 : integer );
  begin
        edit1.text:=inttostr( number1 + number2 )
  end;
```

в)
```
function Multiply( number1, number2 : integer ) : Result;
  var  Result : Integer;
  begin
        Result := number1 * number2;
        Edit1.text:=inttostr( Result )
  end;
```

---

2. Кој е излезот на следната Pascal програма:

```
program Sample( output );
var  x, y : integer;

procedure godoit( x, y : integer );
begin
        x := y;  y := 0;
        write( x, y );
end;
begin
        x := 1; y := 2;
        godoit( x, y );
        writeln( x, y )
end.
```

а) 2 0 0 1
б) 1 0 2 0
в) 2 0 1 2
г) 1 2 3 4

3. Напиши Pascal функција наречена *Multiply2* која враќа integer резултат . Функцијата прифаќа два integer параметри, *number1* и *number2,* а враќа вредност од множењето на двата параметри

а)
```
procedure Multiply2( number1, number2 : integer ) : integer;
    var Result : integer;
    begin
          Result := number1 * number2;
          Multiply2 := Result
    end;
```

б)
```
function Multiply2( number1, number2 : integer ) : real;
    var Result : integer;
    begin
            Result := number1 * number2;
            Multiply2 := Result
    end;
```

в)
```
function Multiply2( number1, number2 : integer ) : integer;
    var Result : integer;
    begin
            Result := number1 * number2;
            Multiply2 := Result
    end;
```

4. Кои променливи се глобални?
5. Објасни вистински параметри.
6. Што е потпрограм?
7. Кои параметри се формални?
8. Објасни локални променливи.
9. Да се направи функција за множење на два броја. Таа треба да се повикува во главната програма на настан On Click на копчето Button1. Броевите да се внесуваат преку 2 edit контроли во формата.
10. Да се направи функција за пресметување faktoriel. Таа да се повикува во главната програма на настан On Change на ScrollBar контролата. Формата треба да содржи ScrollBar, Edit –за прикажување резултати , Label,Button - Izlez .
11. Да се направи функција за пресметување Suma_n. Таа да се повикува во главната програма на настан On Change на ScrollBar контролата. Формата треба да содржи ScrollBar, Edit –за прикажување резултати , Label,Button - Излез .
12. Напиши програма за претворање на целзиусови во фаренхајтови степени со Scrollbar контрола (1f=1.8*c+32)
13. Напиши програма за претворање на евра во денари со Scrollbar контрола (1E=61.5 ден)
14. Во кој дел од програмата се декларира процедурата? Напиши општ облик на наредби со кои се врши декларирање на процедурата во програмата .
15. Нека е формирана следната процедура:

Procedure Vreme(sek:integer; Var C,M,S:Integer);

а) повикајте ја оваа процедура во главната програма.

16. Нека е формирана следната процедура:

Procedure najmal(a:niza; n:byte; var min:integer; var rbr:byte);

а) повикајте ја оваа процедура во главната програма.

ПОГЛАВЈЕ 7
БАЗИ НА ПОДАТОЦИ

Во ова поглавје учениците ќе се запознаат и ќе стекнат основни познавања од следните области:
-дефиниција на поле, запис, табела, база на податоци;
-разлика помеѓу бази на податоци со повеќе табели во различни датотеки и бази на податоци со повеќе табели во една датотека;
-серверски бази на податоци;
-архитектурата на базите на податоци во DELPHI;
-компонентите во картичката DATA ACCESS;
-компонентите во картичката DATA CONTROLS;
-врската помеѓу компонентите од двете картички DATA ACCESS и DATA CONTROLS;
-примена на методите за приказ на состојбата на DataSet;
-примена на методите за движење низ DataSet;
-користење на компонентата TABLE;
-пребарување на табелите;
-изработка на кратка база на податоци во ACCESS;
-изработка на интерфејс за одредена база на податоци.

Поим за база на податоци

Во современиот свет компјутерот стана неопходно средство за обработка на податоци и најефикасен уред за добивање информации.

Базите на податоци постоеле и пред појавата на компјутерите, а со појавата на компјутерите тие добиле електронски облик. Кога и да правите листа од податоци, како на пример, листа со имиња, адреси или телефонски броеви, вие, всушност, креирате база на податоци.

База на податоци е збир од записи зачувани во компјутерот на систематски начин, така што до нив лесно може да се пристапува, управува и надградува.

Најпознати модели на бази се:

- хиерархиски,
- мрежни и
- релациони.

Најчесто користен модел е моделот на **релациона база на податоци**. Кога е потребно да се обработат податоци од повеќе датотеки, тогаш датотеките се поврзуваат во логички систем, наречен база на податоци. Датотеките графички се претставуваат со табели, а за база ќе го користиме и терминот поврзување на табели. Табелите се составени од редици и колони. Редиците претставуваат **записи - слогови**, а колоните **полиња** во записите.

		K_ID	K_Ime	K_Prezime	K_Adresa	K_Telefon	K_E-mail	K_Licna_k	K_EMB	K_Pol	K_Chlen	K_Zachlenuv
▶	+	1	Rade	Najdovski	"Debarska" 15/3/17	3237019	rade@yahoo.com	2115342	1705987400123	mas	3783	12/23/2003
	+	2	Jovana	Gjorgjieva	"Ohridska" 23	3074012	jove@gmail.com	1552432	2111957500176	zen	2345	7/15/2005
	+	3	Maja	Ancevska	"Ilindenska" 74/2	2738202	maja@yahoo.com	1935478	2707936370001	zen	1872	2/28/2004
	+	4	Mice	Krstevski	"Prolet" 34/2/7	5729372	mice@hotmail.com	2007945	3104968527233	mas	754	11/6/2004
	+	5	Igor	Velovski	"Partizanska" 54/1/5	3124502	igor_v@goowy.com	2123877	0707977537862	mas	23	10/25/2005
*		nber)						0			0	

Табела 7.1

Секој запис има сопствени податоци во полињата, со што истиот се карактеризира како посебен **објект,** наречен **ентитет**.

film : Table

	F_ID	F_naslov	F_zhanr	F_glavna_uloga	F_format	F_godina	F_jazik	F_cena
+	1	Balkan Kan	Drama	Vlado Jovanovski	DVD	2005	Makedon	100.00 ден.
+	2	Pred dozdot	Drama	Rade Serbedzija	DVD	1998	Makedon	100.00 ден.
+	3	Sam doma 3	Komedija	Mekoli Kalkan	VHS	1990	Angliski	100.00 ден.
+	4	Smrtonosno oruzje	Akcija	Mel Gibson	VHS	1993	Angliski	100.00 ден.
+	5	So glava vo zid	Drama	Maks Klajn	DVD	2005	Angliski	50.00 ден.
+	6	Barajki go Nemo	Animiran	Nemo	DVD	2004	Angliski	100.00 ден.
+	7	Javac na grobnici	Akcija	Angelina Jolie	DVD	2004	Angliski	50.00 ден.
+	8	Odneseno so viorot	Drama	Vivien Li	VHS	1937	Angliski	50.00 ден.
+	9	Kazablanka	Avantura	Hemfri Bogart	VHS	1944	Angliski	100.00 ден.
+	10	Mis Ston	Istoriski	Petre Prlicko	VHS	1957	Makedon	50.00 ден.
*	umber)					0	Angliski	0.00 ден.

Табела 7.2.

На пример, записите во табелата7.1 се однесуваат на лица, т.е. ентитет е лице. Во табелата 7.2, ентитет е филм. Полето претставува една карактеристика на ентитетот и се нарекува **атрибут**.

Пред да се направи базата, се прави анализа на услугите кои треба да ги дава системот кој ќе управува со базата. Треба да се дојде до податоците кои ќе бидат внесени во базата и системот за управување со базата да ни овозможи нејзино проширување и измени.

Базата се состои од повеќе табели, меѓу кои се поставени одредени релации. Таквата база се нарекува **релациона база на податоци.**

Во базата ќе фигурираат 2 ентитета и тоа: филмови и клиенти. Затоа прво треба да се дефинираат нивните атрибути.

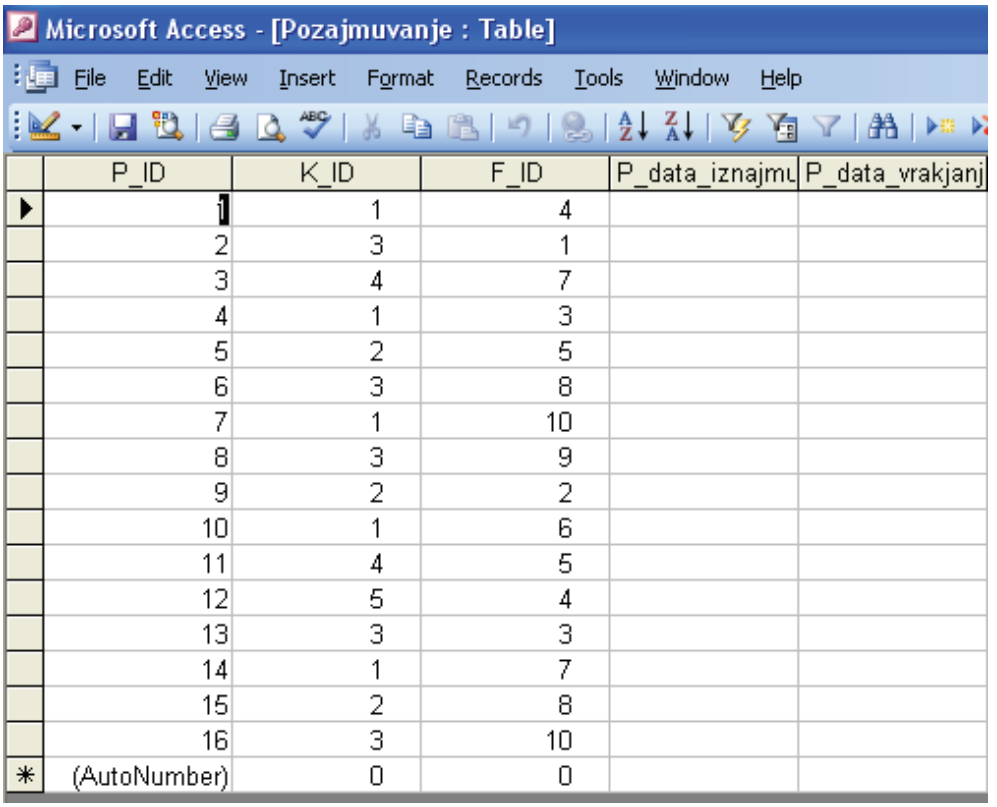
Кај клиенти, атрибути се: име, презиме, тел, e-mail, лична карта итн.

Кај филмови, атрибути се: наслов, жанр, главна улога, формат, година, јазик и цена.

Веднаш може да се дефинира типот на полињата и тоа, текстуални се: име, презиме, адреса, тел, e-mail, наслов, жанр, главна улога, формат, јазик, лична карта, додека нумерички се: година и цена.

Може да се случи некои податоци во полињата да бидат исти. За да не дојде до несакани грешки, во табелите се додава уште една колона со идентификациски број. Идентификациските броеви се генерираат автоматски и не може да дојде до доделување ист број на различни записи.

Следно што треба да направиме е да ги поврземе овие две табели за да може да работи видеотеката/деведетеката. Нејзината работа се состои во изнајмување на некој филм на некое лице – клиент. При секое зајмување треба да се запише клиентот, филмот и датумот на зајмување. Исто така, при враќање на филмот, треба да се запише датумот на враќање и цената која ја платил. Затоа е потребно да креираме табела со полиња кои ќе ги содржат овие податоци.



P_ID	K_ID	F_ID	P_data_iznajmu	P_data_vrakjanj
1	1	4		
2	3	1		
3	4	7		
4	1	3		
5	2	5		
6	3	8		
7	1	10		
8	3	9		
9	2	2		
10	1	6		
11	4	5		
12	5	4		
13	3	3		
14	1	7		
15	2	8		
16	3	10		
(AutoNumber)	0	0		

Табела 7.3. Зајмувања

7.2. Примарен клуч, надворешен клуч и интегритет на базата на податоци

Во секоја табела мора да има барем едно поле кое ќе биде **единствено** за секој запис , односно содржината на тоа поле **нема да се повторува** во ниеден запис. Тоа се обично идентификациони броеви (нумерички податоци) или некое друго поле чија вредност не се повторува (матичен број). Ваквото поле се нарекува **примарен клуч** на табелата.

Примарниот клуч овозможува **сите слогови од една табела да бидат поврзани** со записи од друга табела.

За поврзување на табелите постојат следните видови релации:

еден кон повеќе (1:n) – ова е најчесто користена релација . Во неа на еден ред од првата табела може да се придружат повеќе редови од втората табела, но на еден ред од втората табела може да се придружи само еден ред од првата табела. Еден клиент може да изврши повеќе зајмувања. Истото е и со филмовите – еден филм може да биде зајмен повеќе пати,

еден кон еден (1:1) – е најмалку користена релација. Кај неа еден ред од првата табела има само еден соодветен ред во втората табела. Таа се користи кога треба голема табела да се подели на два дела,

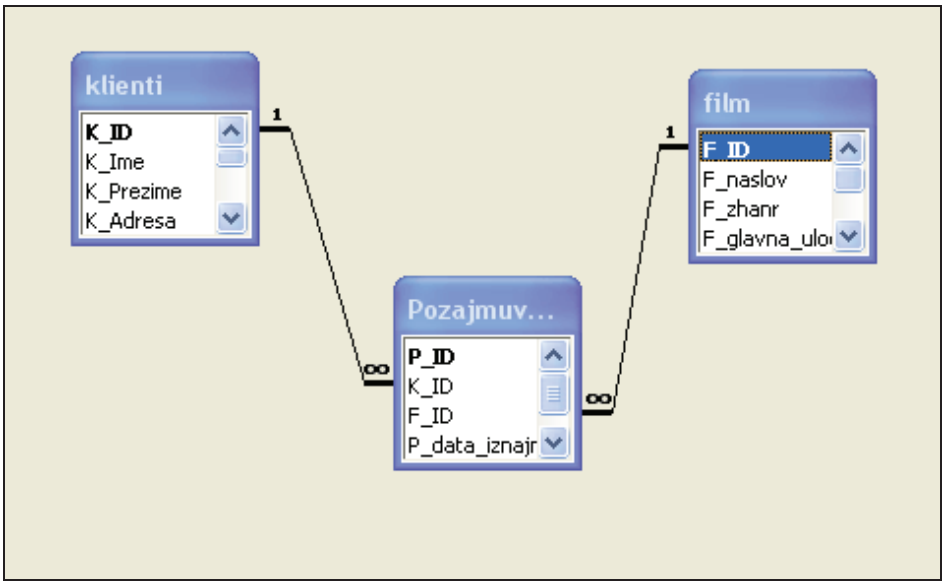
повеќе кон повеќе (n:n) – во оваа релација на еден ред од првата табела може да се придружат повеќе соодветни редови од втората табела и обратно. На пример, нека имаме табела со класови и табела со професори. Бидејќи во многу класови предаваат исти професори, многу записи во табелата со класови ќе содржат ист професор. Исто така , бидејќи еден професор предава во многу класови, многу записи во табелата со професори ќе содржат ист клас.

Во примерот со деведетеката, табелите се поврзуваат преку еквивалентни полиња, т.е. полиња кои содржат ист вид податоци. Полињата не мора да имаат исти имиња. Вообичаено, врската се поставува помеѓу полето, кое е примарен клуч во едната табела (примарната), со поле од друга табела (секундарна), кое одговара на она од првата табела .

Надворешен клуч (Foreign Key)

Тоа е атрибут, кој е еднаков на вредноста на примарниот клуч од друга табела. Со негова помош се спојуваат табелите една со друга.

Релациите претставуваат врска меѓу табелите. Во поголем број случаи се поврзуваат, примарен клуч од една табела со надворешен клуч од друга табела.



Слика 7.1 Релации меѓу табелите

Интегритетот на базата на податоци го сочинуваат множество на правила кои обезбедуваат исправност на зависноста и непроменливост на состојбата на базата при внесување, освежување и бришење на податоци. Основно правило за интегритет е дека за секој слог од секундарната табела, мора да постои само еден слог од примарната. Еве некои правила на интегритет:

1. Не е можно да се внесе слог во секундарната табела, доколку не постои слог во примарната,
2. Не може да се избрише слог од примарната табела, ако постои слог кој е поврзан во секундарната табела,
3. Не е дозволена измена на слог во секундарна табела ако за надворешниот клуч нема соодветни вредности во примарната табела,
4. Не може да се промени вредност на примарен клуч во примарна табела, сé додека постои поврзан слог во секундарна табела.

За оние кои сакаат да знаат повеќе

Планирање на табелите

Технички гледано, потребна е само една табела за функционирање на базата на податоци. ***Голема грашка е сместувањето на многу информации***

во една табела. *Во релационите бази на податоци може да се работи со повеќе табели и да се постават релации меѓу нив.*

Пример1: Нека се претпостави дека треба да се чува информација за контакт со секој потрошувач, врз основа на секоја направена трансакција. Ако сето ова се чува во една табела, ќе мора да има повторување на името на потрошувачот, адресата и телефонскиот број при секое внесување на нова трансакција.

Потрошувач	Тел	Адреса	Датум	Вкупно
Картинг	211-111	Маркова река 2	01.03.05	1,400,000.00
Оли-ви	231-123	Рајкова куќа 234	05.03.05	650,000.00
Картинг	211-111	Маркова река 2	07.03.05	780,000.00
Водно	222-333	Јане Сандански 26	08.03.05	450,000.00

Табела 7.а.

Ако потрошувачот ја промени адресата, тогаш таа ќе треба да се промени во сите трансакции.

Подобар начин е на секој потрошувач да му се додели идентификациски број (ID) – шифра. Тој би бил содржан во табела потрошувачи, во која би биле име, адреса и телефонски број на потрошувачот. Во табелата за трансакции , исто така би го вклучиле ID – бројот на потрошувачот како врска со првата табела, а би биле содржани информации за трансакциите. .

Шифра на потр	Фирма	Тел	Адреса
1	Картинг	211-111	Маркова река 2
2	Оли-ви	231-123	Рајкова куќа 234
3	Водно	222-333	Јане Сандански 26

Табела 7.б : Потрошувачи

Шифра на потр	Датум	Вкупно
1	01.03.05	140,000.00
2	05.03.05	65,000.00
1	07.03.05	78,000.00
3	08.03.05	45,000.00

Табела 7.в: Трансакции

Втора голема грешка е настојувањето да се креира посебна табела за секој извештај. *Во овој случај, податоците се повторуваат во повеќе табели. Ова е непотребно, бидејќи и во извештајот можат да се вклучат информации од повеќе табели. Тоа значи дека е доволно информацијата да постои само во една табела.*

Пример 2: Во дадена организација треба да се чуваат информации за тоа кои вработени завршиле одредени курсеви.

Решението со една табела е следното:

Име и презиме	Раб_место	Тел	Курс	Часови
Борка Кирчова	Економист	455-554	Word	25
Трајче Петковски	Инкасант	421-123	Excel	20
Борка Кирчова	Економист	455-554	Excel	20
Борка Кирчова	Економист	455-554	Windows	15
Слободан Постолов	Референт	124-015	Outlook	20

Табела 7.г.

Ако еден вработен заврши повеќе курсеви, мора да се додадат дупли записи во табелата. Ако еден вработен ја напушти фирмата, неговите записи

треба да се бришат, со што може да се изгубат податоци за траењето на курсот. (со Борка Кирчова се врзани курсевите за Windows и Word).

Име и презиме	Раб_место	Тел	Курс	Часови
Трајче Петковски	Инкасант	421-123	Excel	20
Слободан Постолов	Референт	124-015	Outlook	20

Табела 7.д Табела со избришани податоци

Подобар начин е креирање одвоени табели за вработените, за курсевите и за завршените курсеви, а врската меѓу нив да се обезбедува со идентификациони броеви – шифри.

	Шифра на врабо	Име и презиме	Раб_место	Тел
▶	1	Борка Кирчова	Економист	455-554
	2	Трајче Петковски	Инкасант	421-123
	3	Слободан Постол	Референт	124-015

Табела 7.ѓ. Вработени

Шифра на курс	Курс	Часови
K1	Word	25
K2	Excel	20
K3	Windows	15
K4	Outlook	20

Табела 7.е. Курсеви

Шифра на врабо	Шифра на курс	Датум	Оценка
1	K1	11.02.05	4
2	K2	22.04.05	5
1	K2	22.04.05	4
1	K3	01.02.05	4
3	K4	18.02.05	5

Табела 7.ж. Завршени курсеви

7.3. Типови бази на податоци

Во однос на сложеноста , постојат три различни типови на бази на податоци:

- локални бази на податоци,
- клиент/сервер бази на податоци и
- дистрибуирани бази на податоци.

Локалните бази на податоци претставуваат наједноставен тип на база. Тоа е база која се наоѓа на еден компјутер. Сите промени на податоците директно се запишуваат во базата. Претставници на локалните бази се Access, dBase и Paradox. Во конкретната работа со базите на податоци ќе ги користиме локалните бази на податоци.

Вториот тип на база на податоци е клиент/сервер. Кај овој тип на бази на податоци базата се наоѓа на посебен компјутер, кој се нарекува сервер на базата на податоци (file server) и тој управува со базата. Еден или повеќе корисници

(client) можат да пристапат до базата. Корисниците најчесто се во мрежа, независни едни од други и во ист момент повеќе од нив можат да сакаат да пристапат до базата. Оваа состојба ја решава серверот, кој има вградени механизми за решавање на проблемите на истовремен пристап до базата на податоци.

Заради заштита на податоците, корисниците не пристапуваат директно во базата на податоци, туку со помош на програми на локалните компјутери. Овие програми, кои се нарекуваат клиент програми, се грижат за тоа корисниците да почитуваат одредени правила и да не извршуваат дејствија што можат да и наштетат на базата на податоци.

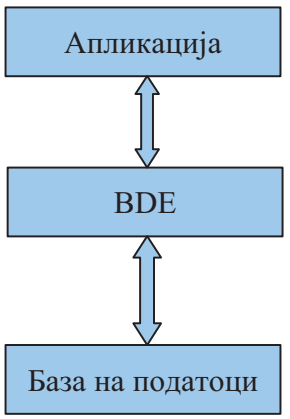
Посебен вид клиент/сервер бази на податоци се дистрибуираните бази на податоци.

7.4. Организација на врските со базата на податоци

7.4.1. Технологија BDE (Borland Database Engine)

За да им овозможи пристап на базите на податоци, Delphi ја користи технологијата БДЕ. Таа е збир на услужни програми кои овозможуваат пристап до разни бази на податоци. Начинот на комуникација на апликацијата со базата на податоци е прикажан на слика 7.1.

Форматите на базите кај различни производители се разликуваат, па затоа BDE се испорачува со множество драјвери кои им овозможуваат на програмите да комуницираат со различни типови податоци. Овие драјвери ги преведуваат наредбите до базите на податоци во команди специфични за даденот тип база. Сите верзии на Delphi имаат драјвер кој поддржува пристап на Paradox i dBASE бази на податоци. Овој драјвер се нарекува STANDARD и обезбедува се што е потребно за работа со овие типови бази.

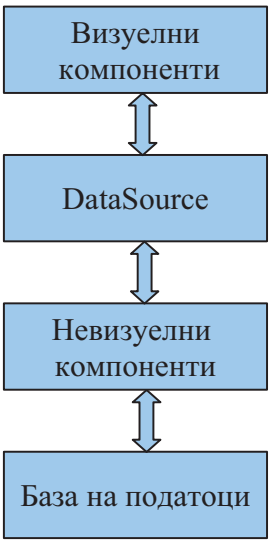


Слика 7.1.

За коректен пристап до базата БДЕ користи алиас (alias). Тоа е збир на параметри кои го опишуваат начинот на поврзување со базата на податоци. Алиасот му кажува на БДЕ, кој драјвер да го употреби и каде на дискот се наоѓаат датотеките со базата на податоци.

Најголем број компоненти за работа со базите на податоци се наоѓа на страните BDE, Data Access и DataControl. Компонентите за работа со базите може да се поделат на невизуелни и визуелни. Невизуелните компоненти

овозможуваат пристап до базата на податоци, односно до податоците во базата. Визуелните компоненти овозможуваат работа со податоците (преглед, промена...). Најчесто користена невизуелна компонента е Table. Визуелни компоненти кои често се користат се DBEdit, DBListBox, DBGrid, DBNavigator и др. За комуникација на невизуелните и визуелните компоненти се користи невизуелна компонента DataSource.

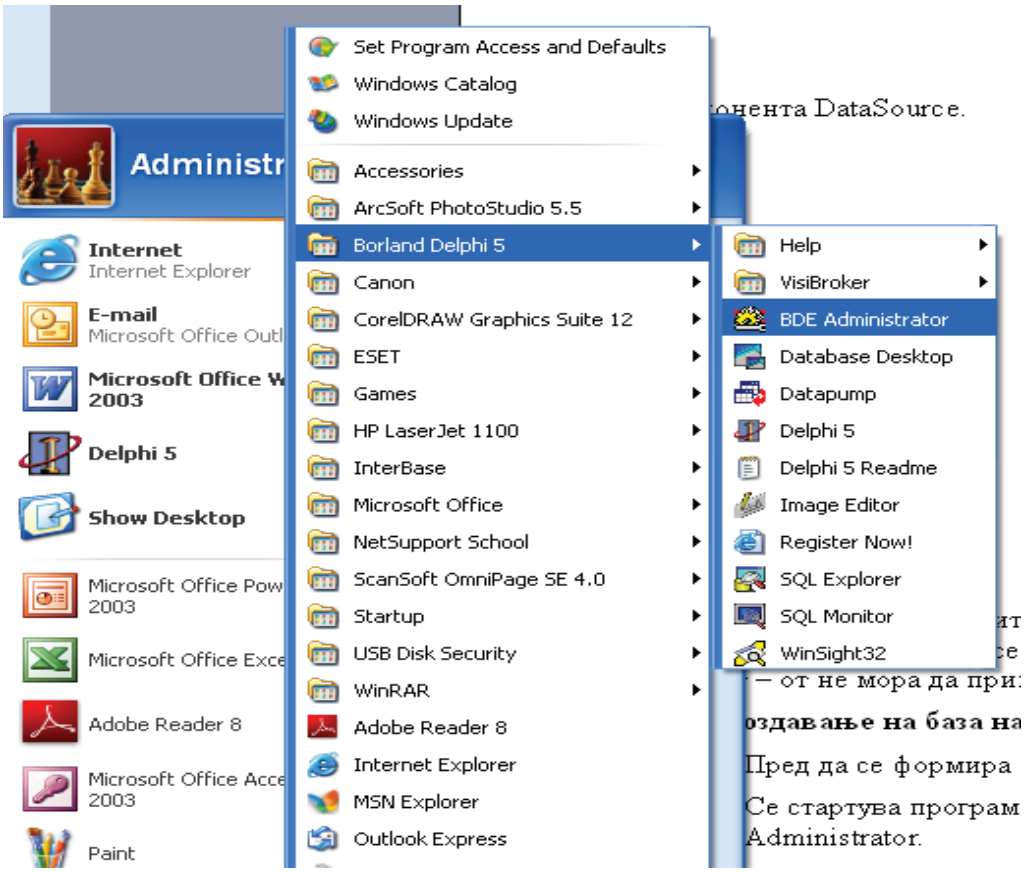


Слика 7.2.

Во работата со базите на податоци се користи поимот DataSet. Тоа е множество на податоци до кои може да се дојде на основа на податоците содржани во базата. Податоците од DataSet не мора да припаѓаат на една табела.

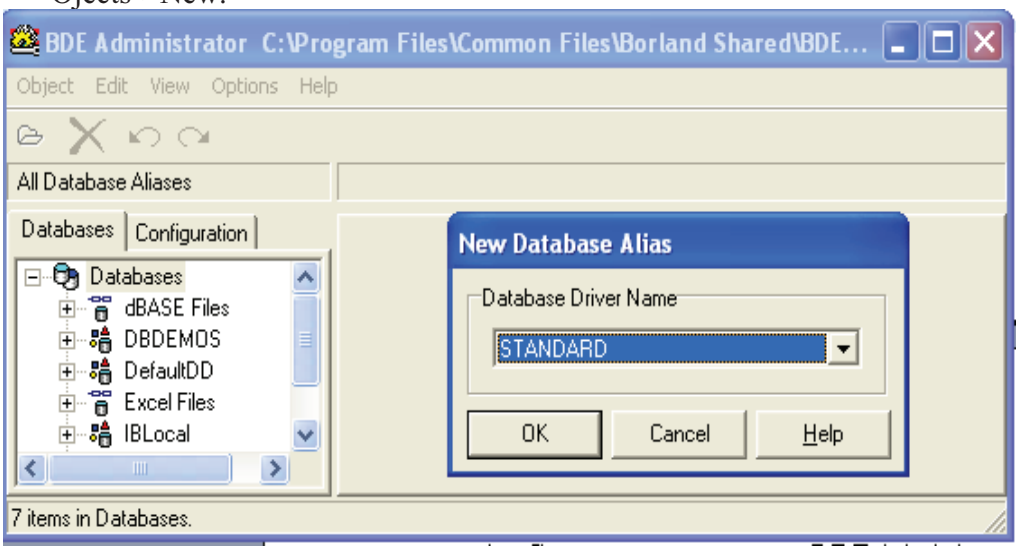
7.4.2. Создавање на база на податоци со користење на програмата DatabaseDesktop

Пред да се формира базата треба да се направи алиас за одредена база.



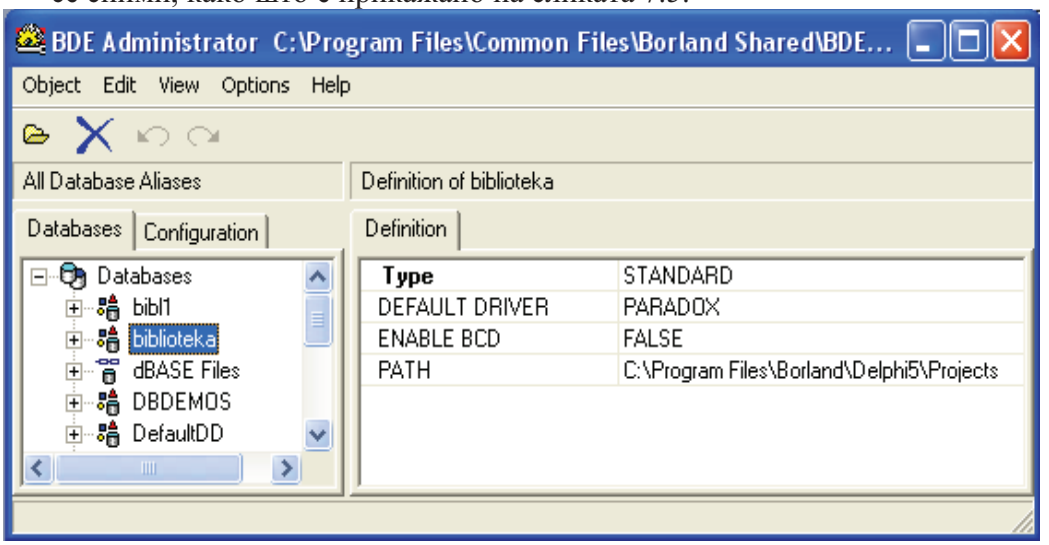
Слика 7.3.

- Се стартува програмата BDE Administrator, преку Start->Borland Delphi7 -> BDE Administrator.
- Во прозорецот се одбира картичката Databases, а во мени се одбира Ojects->New.



Слика 7.4

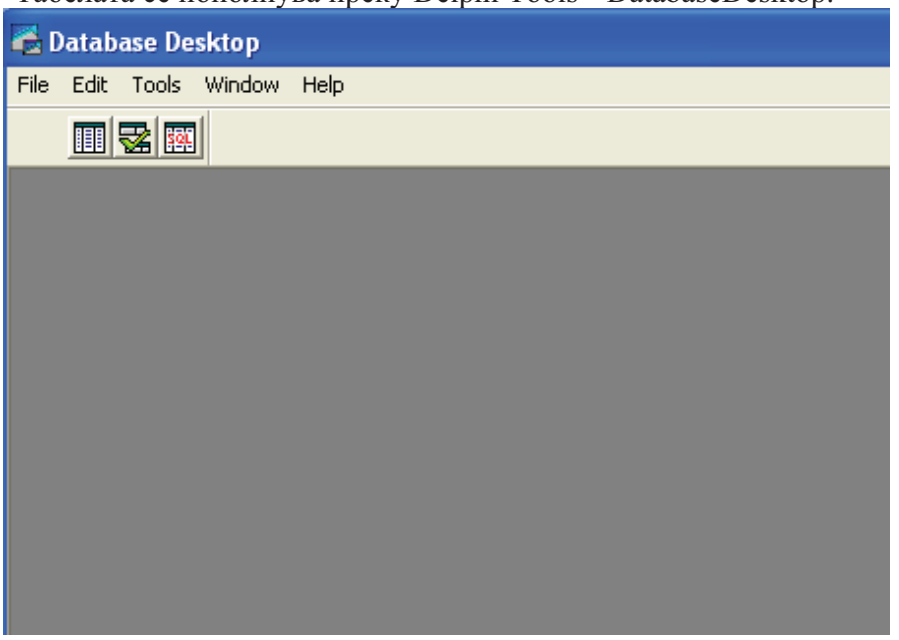
- Програмата го нуди за користење драјверот STANDARD за формирање алиаси,
- Го внесуваме името на новиот алиас –Biblioteka и патот (PATH) каде ќе се сними, како што е прикажано на сликата 7.5.



Слика 7.5.

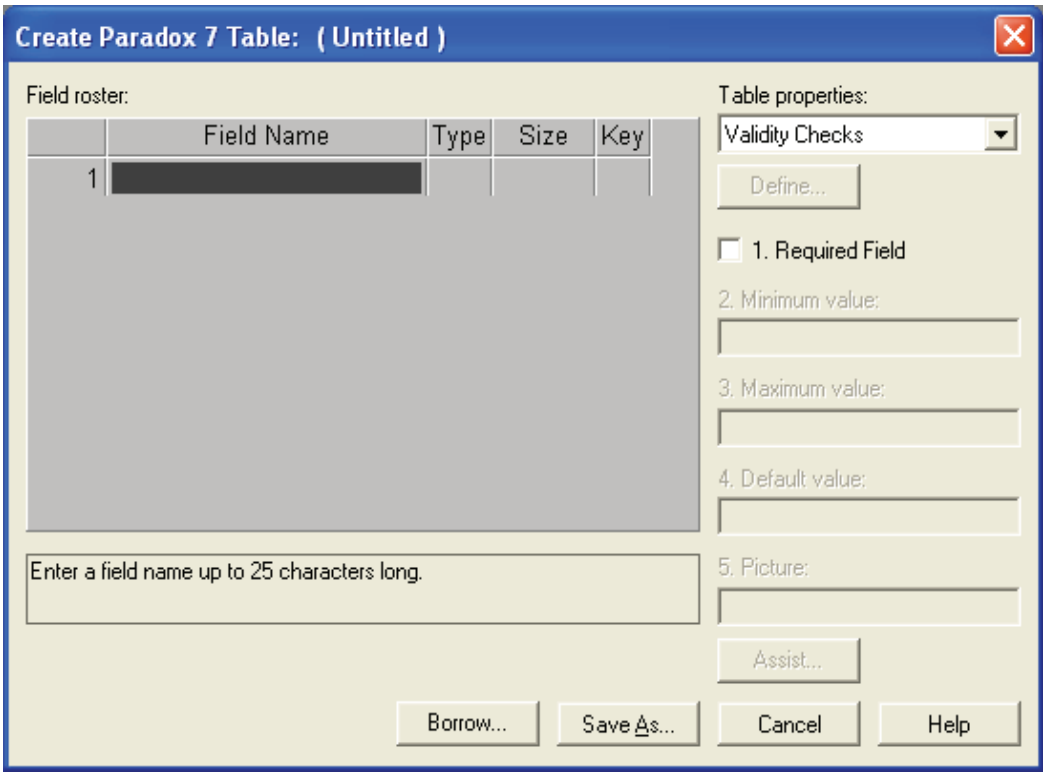
- Се враќаме во Delphi.
  Delphi овозможува правење табели преку мени :

-Tools->Database desktop
-File-New-Table (tip PARADOX),
-Потоа се дизајнира табелата . За секое поле се дефинира име,тип, големина и клуч.
-Структурата на табелата се снима со Save As.
-Табелата се пополнува преку Delphi Tools->DatabaseDesktop.



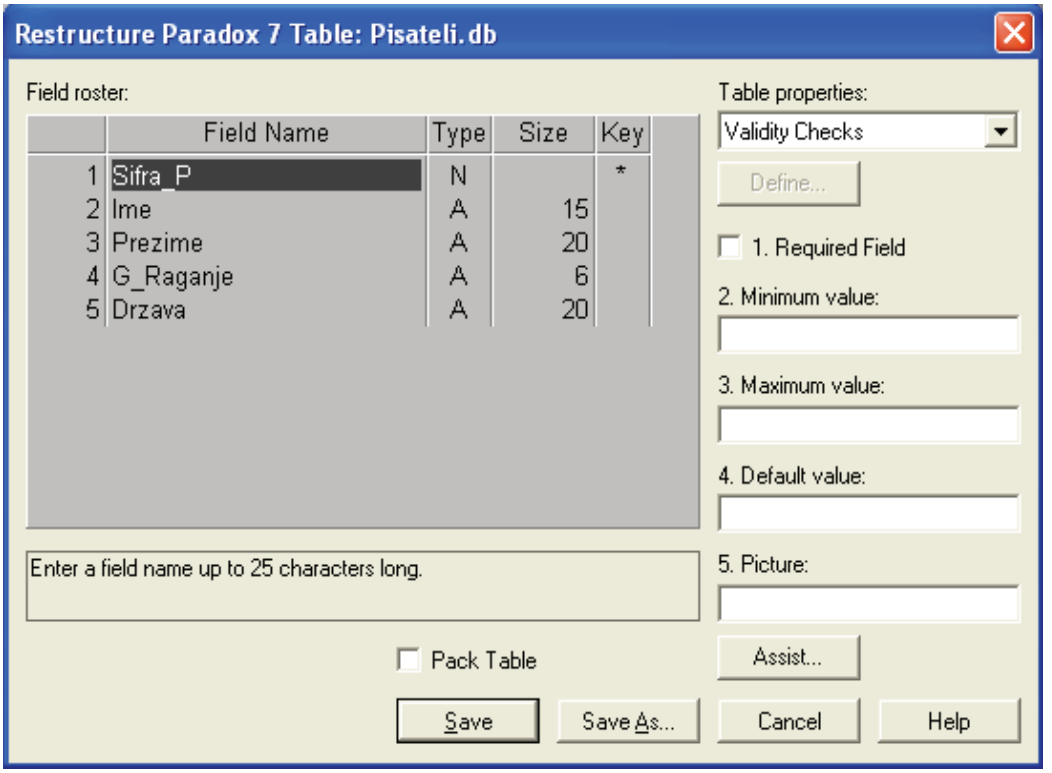
Слика 7.6.

Во опцијата File->New ->Table ја креираме табелата. Ги внесуваме имињата на полињата, нивниот тип, големина и клуч.



Слика 7.7.

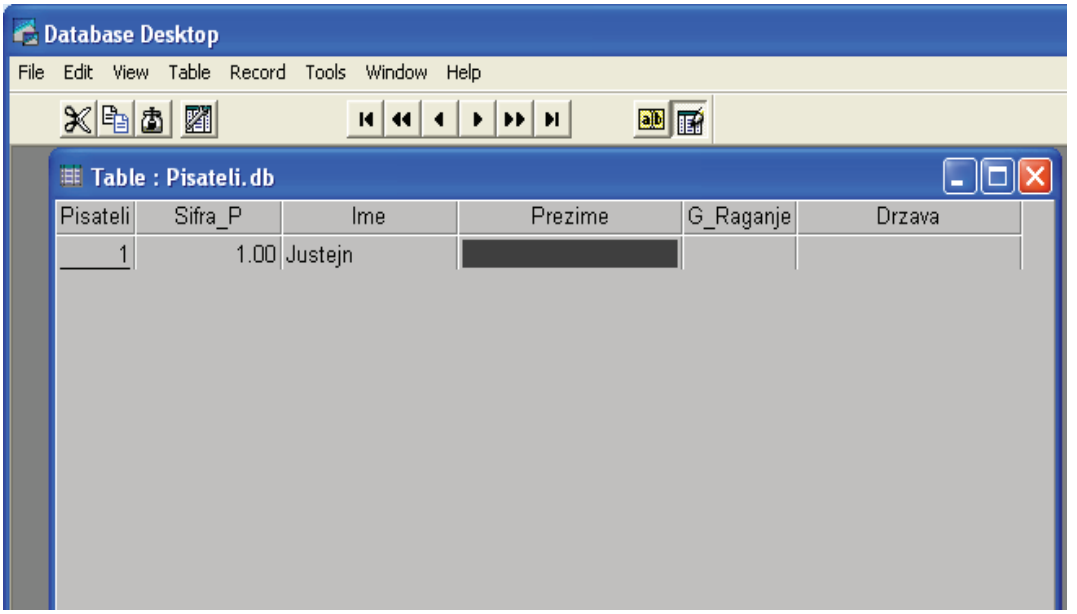
Структурата на табелата Pisateli која се формира го има следниот облик:



Слика 7.8.

Таа се зачувува на Save As како Pisateli .

Се пополнува откако ќе се отвори во Database desktop, а потоа од менито Table->Edit .



Слика 7.9.

Дизајнирана и пополнета, табелата Pisateli ќе го има изгледот како на слика 7.10

Pisateli	Sifra_P	Ime	Prezime	G_Raganje	Drzava
1	1.00	Justejn	Gordber	1952	Norveska
2	2.00	Zak	Prever	1910	Francija
3	3.00	Jovan	Ducic	1871	Hercegovina
4	4.00	Fjodor	Dostoevski	1821	Rusija
5	5.00	Zan	Molier	1622	Francija
6	6.00	Ivo	Andric	1892	Bosna
7	7.00	Borisav	Stankovic	1876	Srbija
8	8.00	Sima	Pandurovic	1883	Srbija
9	9.00	Ivan	Gundulic	1589	Dubrovnik
10	10.00	Franc	Presern	1800	Slovenija
11	11.00	Ivan	Cankar	1876	Slovenija
12	12.00	Laza	Lazarevic	1851	Srbija
13	13.00	Henrik	Sjenkjevic	1846	Polska
14	14.00	Viliem	Sekspir	1564	Anglija
15	15.00	Milos	Crnjanski	1893	Ungarija
16	16.00	Onore	De Balzak	1799	Francija

Слика 7.10.

На сличен начин се формира табелата Knigi чиј изглед е прикажан на слика 7.11.

Table : Knigi.DB

Knigi	Sifra_K	Sifra_P	Naslov	Izdavac	Pismo
1	1.00	1.00	Devojka so portokali	Gocmar	Latinica
2	2.00	2.00	Pesni	Tri	Cirilica
3	3.00	3.00	Bogatstvoto na carot Radovan	Misla	Cirilica
4	4.00	4.00	Zlostorstvo i kazna	Zavod	Cirilica
5	5.00	5.00	Skrzavec	Tri	Cirilica
6	6.00	6.00	Mostot na Drina	Tri	Cirilica
7	7.00	7.00	Necista krv	Bigz	Latinica
8	8.00	8.00	Pesni	Kairos	Cirilica
9	9.00	9.00	Osman	Prosveta	Cirilica
10	10.00	1.00	Dijagnoza i drugi noveli	Gocmar	Latinica
11	11.00	1.00	Bozicna misterija	Gocmar	Latinica
12	12.00	10.00	Soneten venec	Prosveta	Cirilica
13	13.00	7.00	Vo nokta - Kostana	Prosveta	Cirilica
14	14.00	6.00	Prokleta avlija	Kniga	Cirilica
15	15.00	6.00	Travnicka hronika	Tri	Cirilica
16	16.00	11.00	Kralot na Betajnova	Zavod	Cirilica

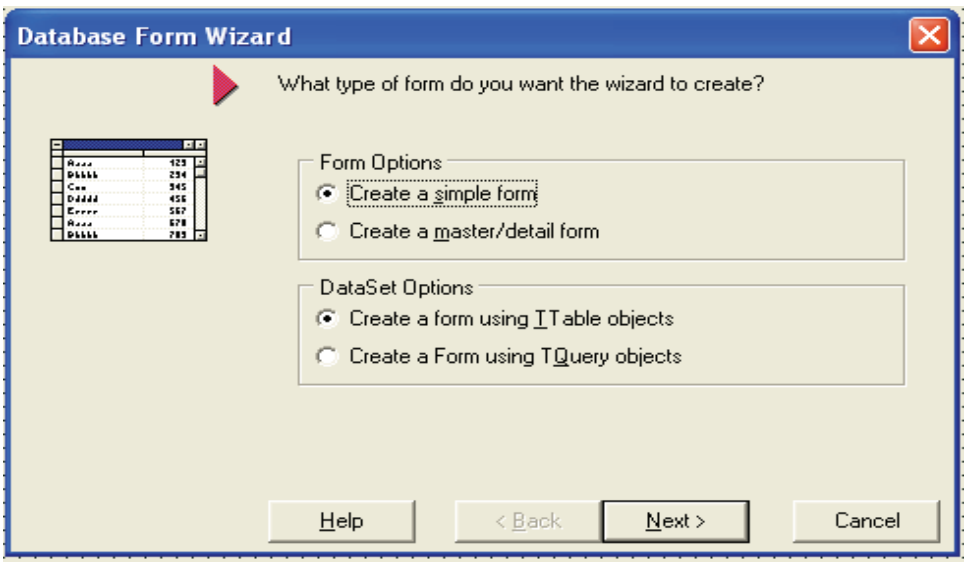
Слика 7.11.

Со ова во Delphi создадовме "жива" база на податоци која може да се користи за различни намени.

7.5. Проектирање на апликации за работа со бази на податоци со користење на волшебник

Податоците од постоечките бази на податоци може да се користат без употреба на компоненти за работа со бази на податоци. Delphi во опцијата Database има наредба Form Wizard, со чија помош може да се направи форма за внесување на податоци во базата на податоци.

- Со активирање на наредбата Form Wizard се добива дијалог - прозорец како слика 7.12.

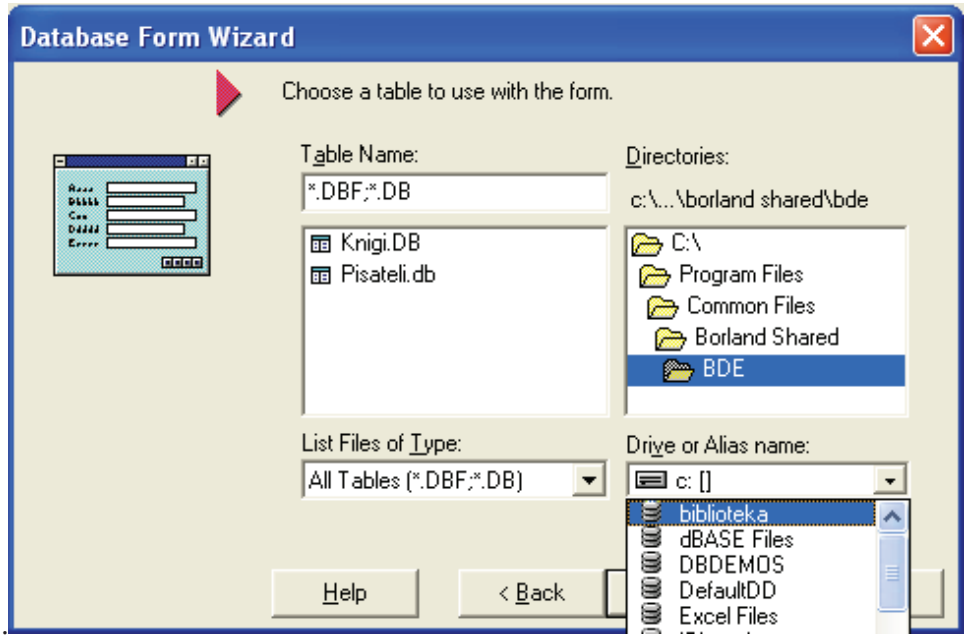


Слика 7.12.

Во овој дијалог - прозорец се одредува типот на базата и типот на DataSet кој ќе биде користен. За нашата работа одбираме

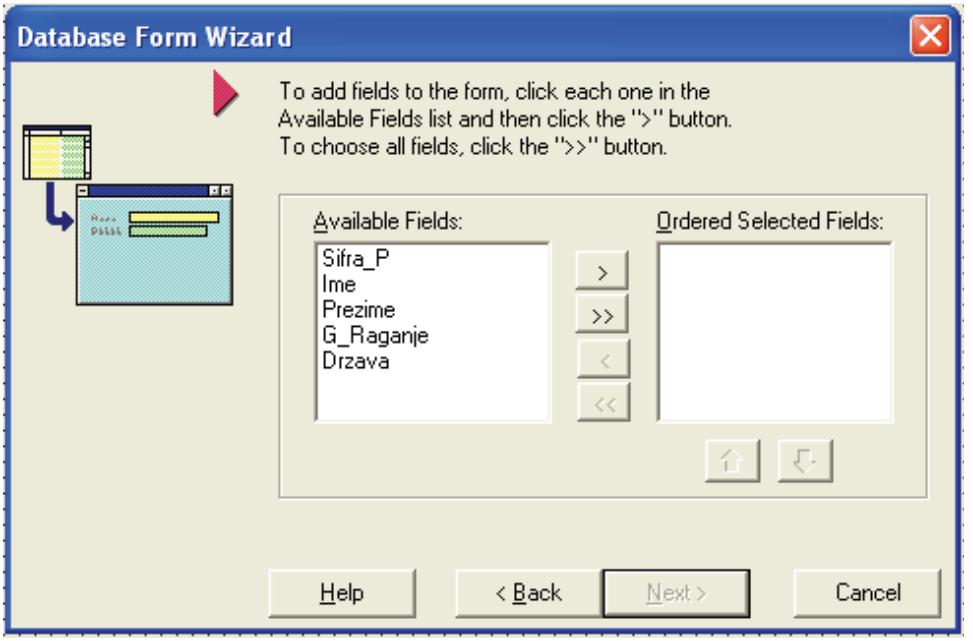
- Create a simple form
- Create a form using simple TTable objects

- Во следниот дијалог – прозорец, кој е прикажан на слика 7.13. се овозможува избор на табелата која ќе биде користена врз основа на соодветниот драјв и алиас. Треба да се пронајде во ComboBox, алиасот – Biblioteka , а потоа треба да се одбере табелата Knigi.db.



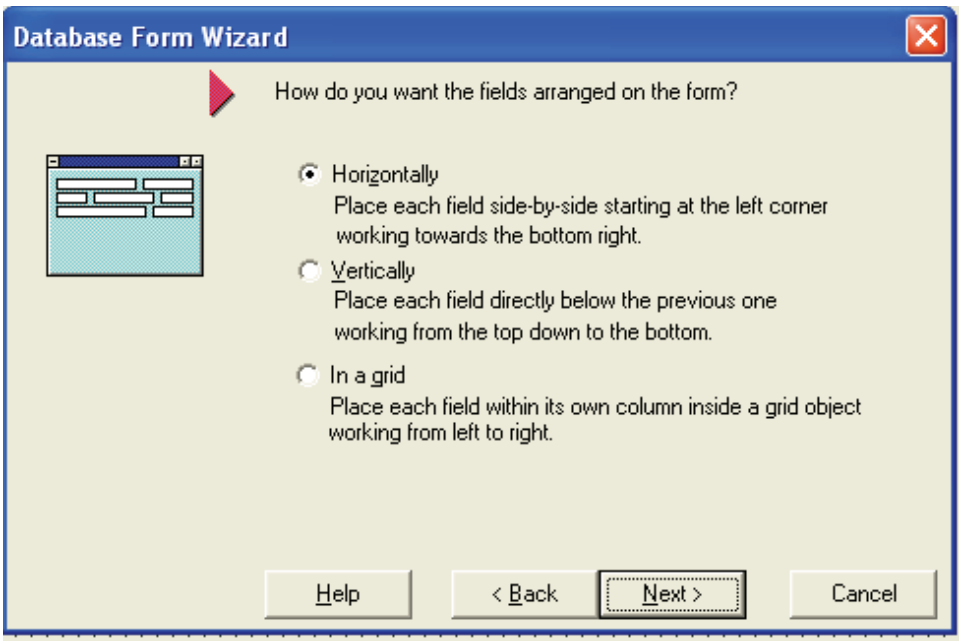
Слика 7.13.

- Во третиот дијалог - прозорец (слика 7.14.) треба да се одберат полињата од одбраната табела кои ќе се појават во формата. Полињата можат да се пренесуваат поединечно (>) или сите одеднаш (>>). За понатамошна работа ќе ги пренесеме сите полиња од левата во десната листа.



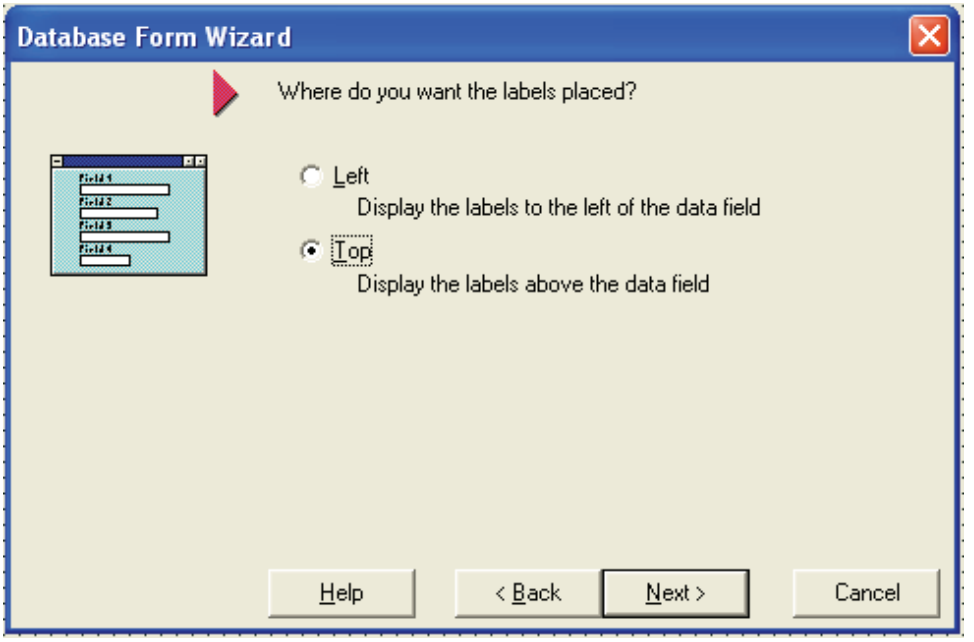
Слика 7.14.

- Потоа се одбира распоредот на компоненти:хоризонтален, вертикален или во облик на мрежа (слика 7.15.). Ќе одбереме вертикален распоред



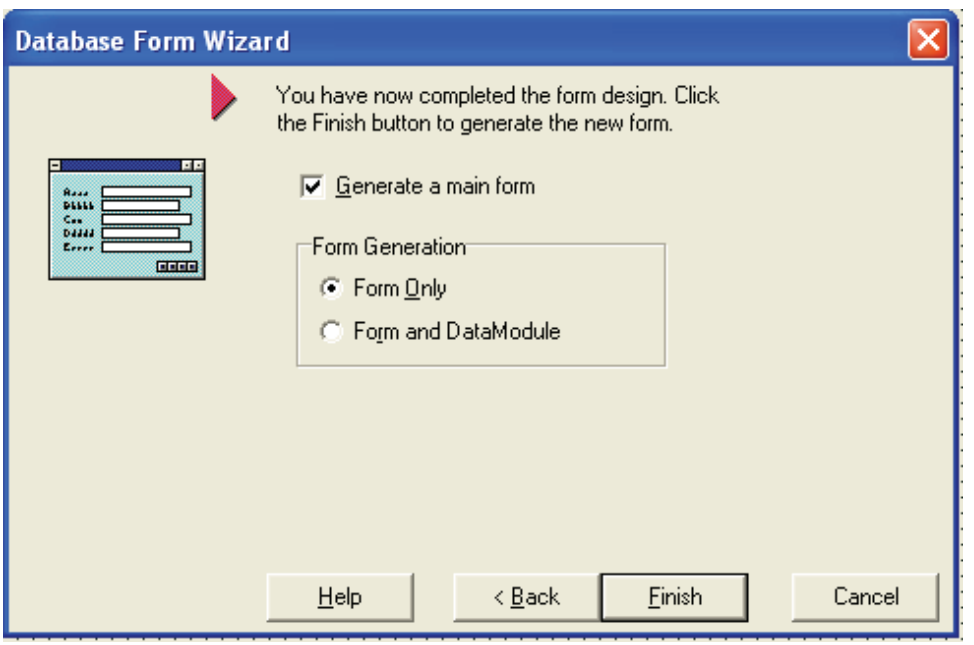
Слика 7.15.

- Се одбира положбата на натписите (лабелите) за полињата во однос на секоја компонента. Тие можат да бидат поставени лево или над компонентите.



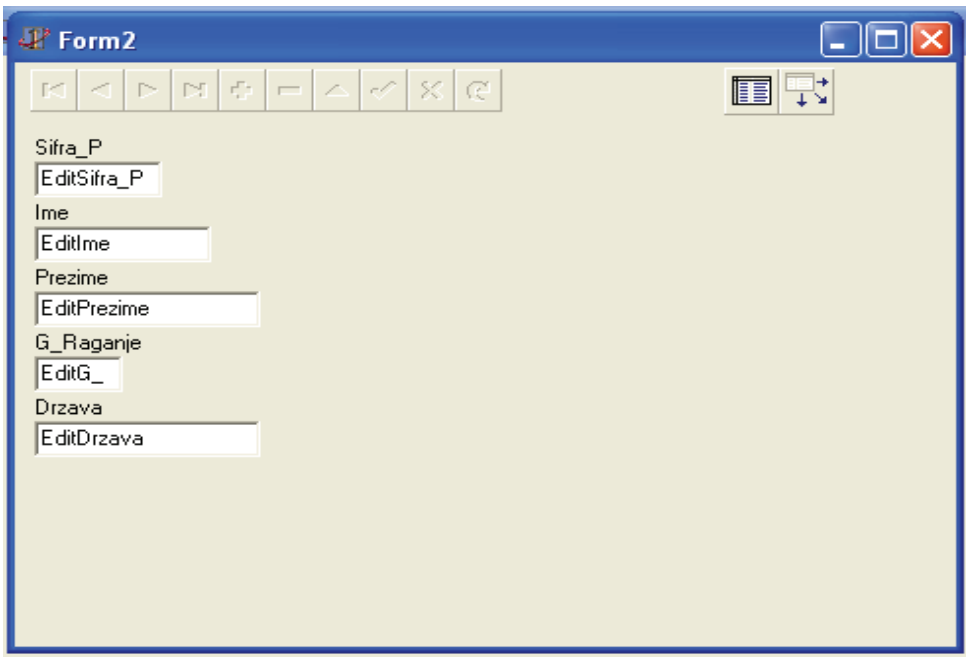
Слика 7.16.

- Во следниот прозорец може да се одбере креирање само на форма или креирање и на форма и на модул за податоци.
    - Се одбира опцијата Form only.



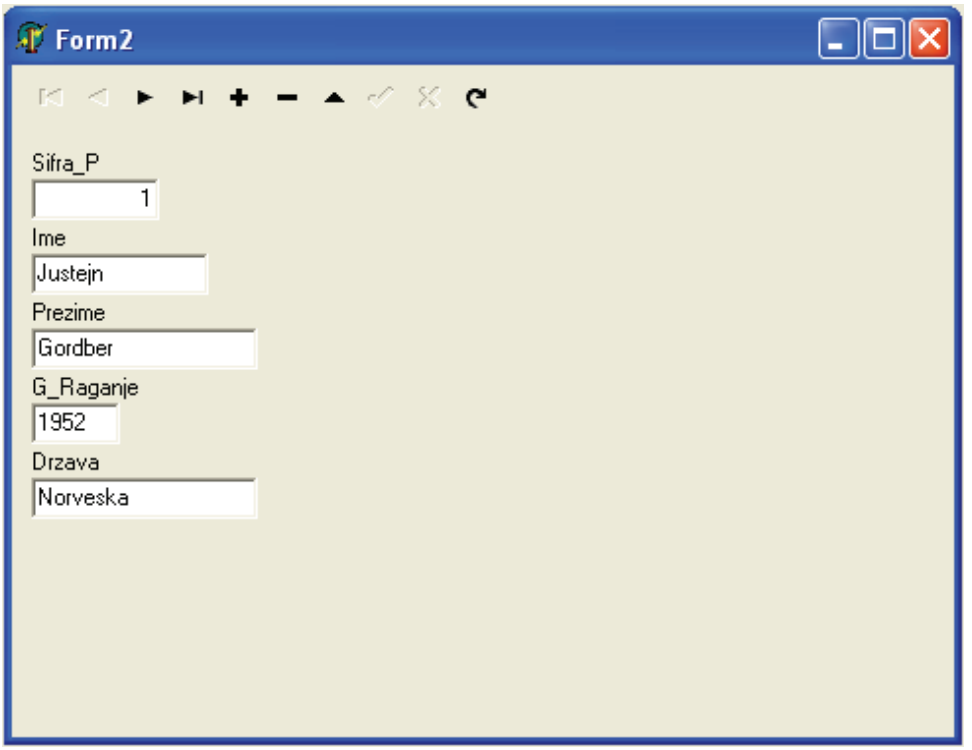
Слика 7.17.

По кликање на копчето Finish се добива следната форма:



Слика 7.18.

Со клик на копчето Run се добива форма, во чиј горен дел се наоѓа компонента за движење низ табелата од базата на податоци – DBNavigator. Со помош на оваа компонента може да се прегледува табелата писатели ,а исто така може да се менуваат податоците. Промените се прифаќаат ако се активира копчето за чекирање или ако се премине на следниот слог, а не се прифаќаат ако се селектира копчето за поништување на промените (X).



Слика 7.19.

7.6. Компоненти за врска со базата на податоци

7.6.1 Компоненти за врска со табелата на базата на податоци

Компонентите за работа со бази на податоци , се наоѓаат на страните BDE, Data Access и Data Controls на линијата на компоненти. Тие функционираат како визуелни и невизуелни. За да може Delphi да комуницира со базата на податоци во елементарен облик, неопходно е да се користат следните три компоненти:

1. Table
2. DataSource
3. DBGrid

 Компонентата Table се наоѓа на страната Data Access. Таа овозможува најбрз и наједноставен пристап до табелите во базата на податоци. Таа е доволна за работа со локални бази на податоци.

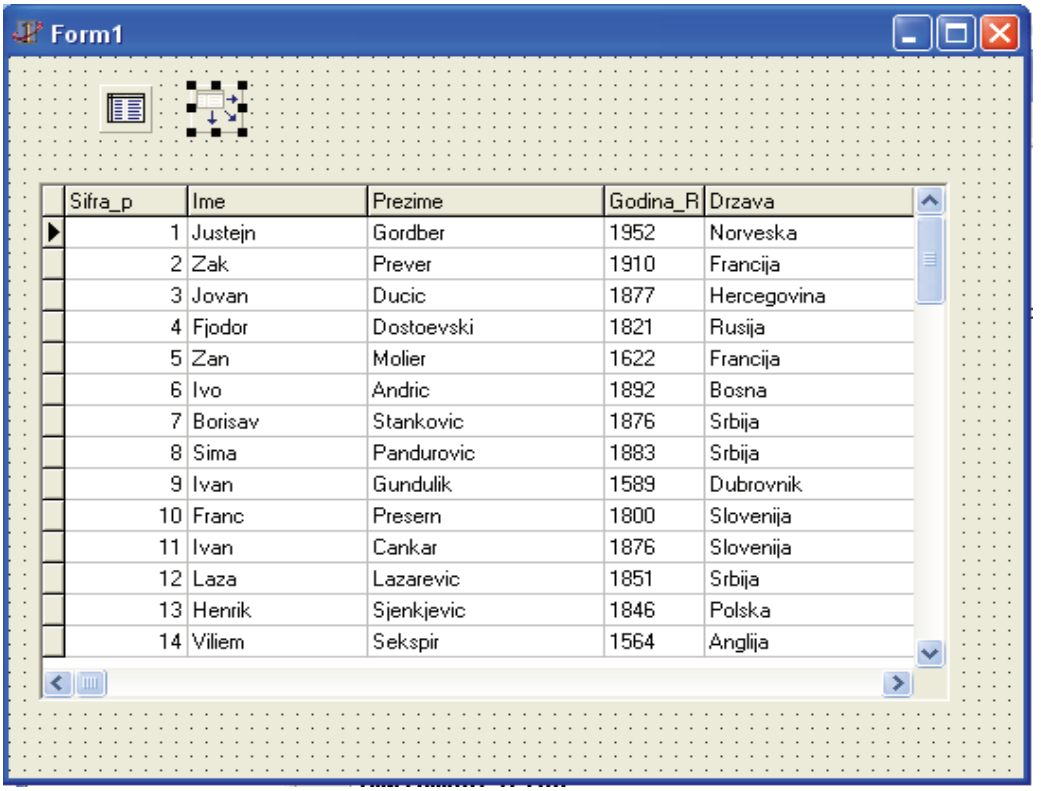
 Компонентата DataSource, исто така се наоѓа на страната Data Access. Оваа компонента обезбедува механизам на поврзување на невизуелните компоненти (Table,..) со визуелните компоненти кои прикажуваат податоци (DBGrid, …). Сите компоненти за работа со податоци кои се сместени на формата, поврзани се преку DataSource за множеството на податоци - DataSet.

 Компонентата DBGrid овозможува прикажување на DataSet во табеларен облик.

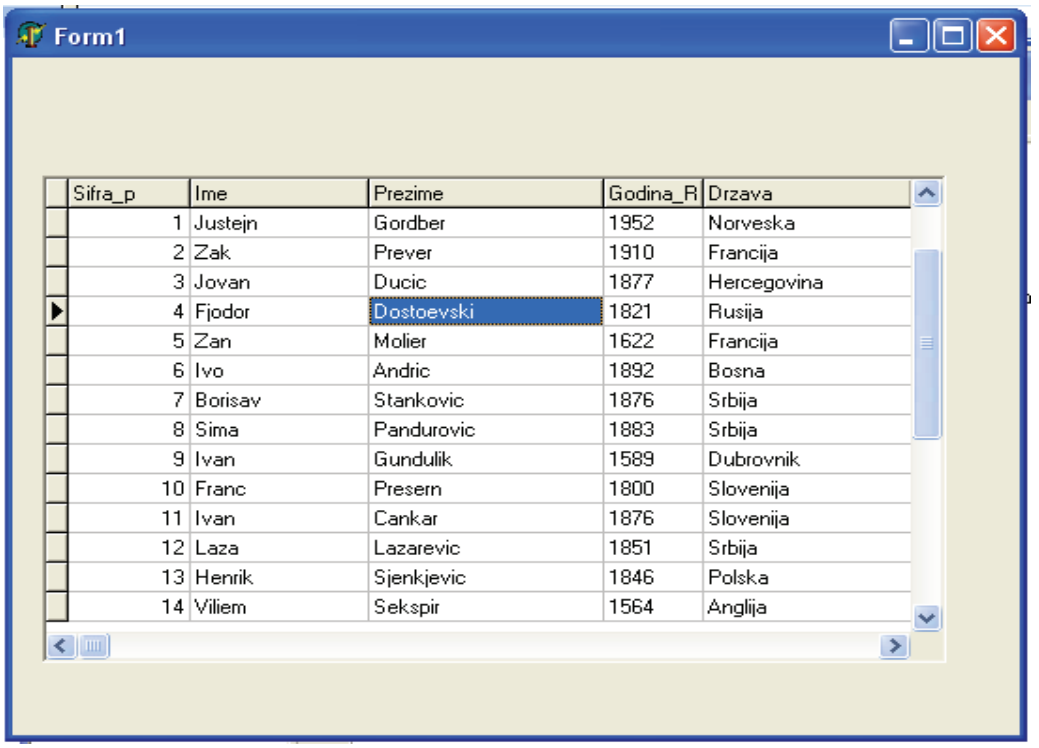
Со помош на наведените компоненти, можеме на формата да прикажеме податоци од некоја табела од базата на податоци. Постапката за тоа е следната:

1. Треба да се стартува нова апликација и на формата треба да се постави компонента Table.
2. Во ObjectInspector, својството DatabaseName треба да се постави на вредност biblioteka .
3. Во својството TableName одбираме табела Pisateli, што значи дека ќе работиме со табелата писатели.
4. На формата се поставува компонентата DataSource и за нејзиното својство DataSet одбираме Table1. Со оваа постапка изворот на податоци е поврзан со DataSet .
5. Следен чекор е поставувањето на компонентата DBGrid на формата. Својството DataSource на оваа компонента треба да се постави на вредност DataSource1. Со оваа постапка мрежата се поврзува со изворот на податоци, а индиректно и со DataSet .
6. Конечно на компонентата Table која се наоѓа на формата , треба да и се промени својството Active на вредност True. Со ова се овозможува прикажување на податоци од табелата Pisateli во DBGrid контролата.

Изгледот на формата која ќе се добие е прикажан на слика 7.20.
Со стартување на оваа апликација се добива работен прозорец како на слика 7.21 Во оваа табела може да се пристапи до секој податок. Вредностите во полето може да се менуваат , но после прекинот на работата на апликацијата промените нема да бидат зачувани во табелата. За да се овозможи обработка на податоците во базата на податоци, неопходно е да се постави уште една компонента во формата која ги овозможува потребните манипулации со податоците.

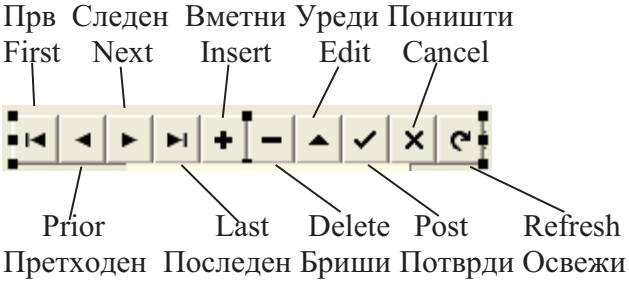


Слика 7.20.



Слика 7.21.

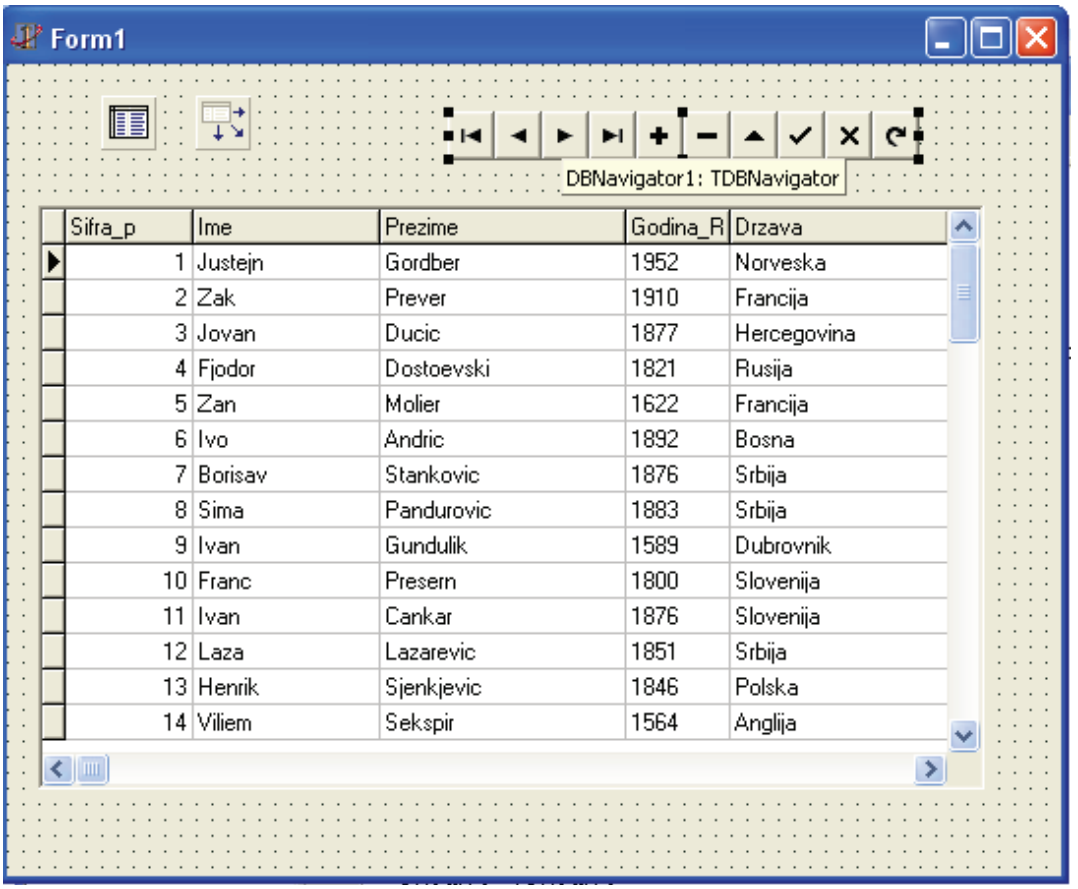
 Компонентата DBNavigator му овозможува на корисникот контрола над операциите запишување, бришење , промени и движење низ слоговите на базата на податоци. DBNavigator најчесто се користи со другите компоненти за компоненти за комуникација со базата на податоци (Table, DataSource, DBGrid, …). Компонентата DBNavigator го има следниот изглед.



DBNavigator содржи десет копчиња со следните функции:

- копчето First овозможува премин на првиот слог во табелата,
- копчето Prior овозможува поместување на покажувачот за еден слог наназад,
- копчето Next овозможува поместување на покажувачот за еден слог нанапред,
- копчето Last овозможува поместување на покажувачот на последниот слог,
- копчето Insert овозможува вметнување слог во табелата,
- копчето Delete брише слог од табелата на адресата каде што се наоѓа покажувачот,
- копчето Edit овозможува промена на содржината на полето,
- копчето Post врши запишување на содржината на слогот во базата,
- копчето Cancel ги поништува направените измени во слогот,
- копчето Refresh ги освежува податоците на компонентите за работа со базата.

По поставување на компонентата DBNavigator на формата, се добива нов изглед на формата, кој е прикажан на слика 7.22. Со стартување на апликацијата, се добива нов работен прозорец, во кој можат да се користат наведените копчиња од DBNavigator.



Sifra_p	Ime	Prezime	Godina_R	Drzava
1	Justejn	Gordber	1952	Norveska
2	Zak	Prever	1910	Francija
3	Jovan	Ducic	1877	Hercegovina
4	Fjodor	Dostoevski	1821	Rusija
5	Zan	Molier	1622	Francija
6	Ivo	Andric	1892	Bosna
7	Borisav	Stankovic	1876	Srbija
8	Sima	Pandurovic	1883	Srbija
9	Ivan	Gundulik	1589	Dubrovnik
10	Franc	Presern	1800	Slovenija
11	Ivan	Cankar	1876	Slovenija
12	Laza	Lazarevic	1851	Srbija
13	Henrik	Sjenkjevic	1846	Polska
14	Viliem	Sekspir	1564	Anglija

Слика 7.22.

7.6.2. Компоненти за прикажување и менување на податоци во базата

Покрај претходно наброените компоненти, постојат и голем број други компоненти кои можат да се користат во работата со базата на податоци. Голем дел од нив водат потекло од веќе обработените компоненти од страната Standard. Се разликуваат во тоа што имаат префикс DB, што укажува на тоа дека овие компоненти поседуваат механизми за автоматска комуникација со компонентите за пристап на базите. Оваа комуникација се остварува преку компонентата DataSource. Кога компонентата е поврзана со изворот на податоци, односно кога се поставени нејзините својства DataSource и DataField , податоците автоматски се прикажуваат во неа.

Постојат својства кои се заеднички за сите компоненти за работа со бази на податоци: DataSource, DataField и ReadOnly. Компонентите DBGrid и DBNavigator немаат својство DataField, затоа што овие компоненти вршат операции над целиот слог.

Во DataSource својството се задава името на DataSource компонентата со која се поврзува оваа компонента.

Својството DataField го обезбедува полето чија содржина ќе биде прикажана во компонентата.

Со поставување на својството ReadOnly на вредност True, му се оневозможува на корисникот да ја промени содржината на компонентата, а со тоа и содржината на полето.

 Компонентата DBText се користи за прикажување на содржината на полето од табелата. Не постои можност за промена на нејзината содржина.

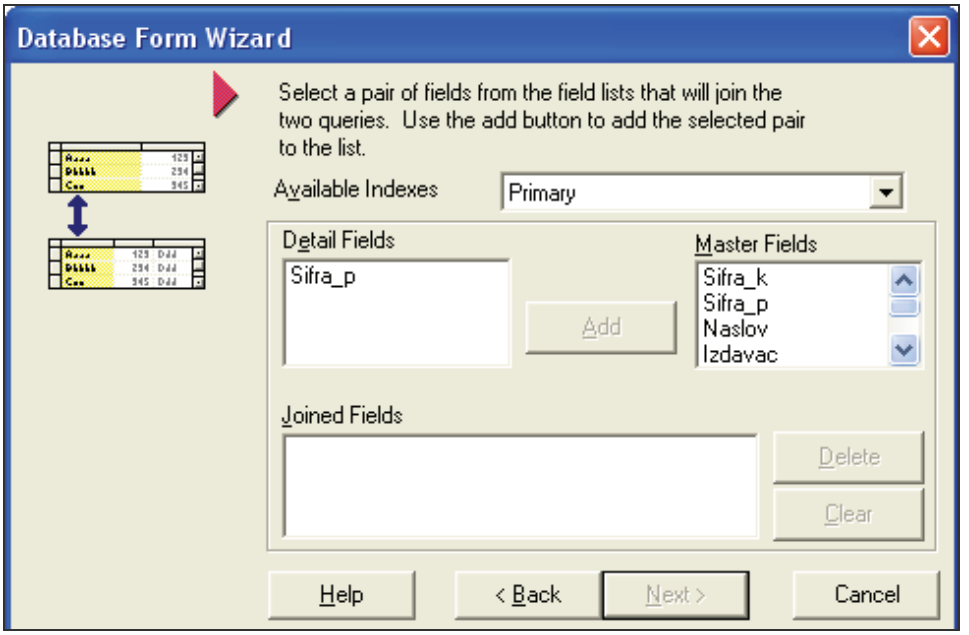
 Како и Edit компонентата , DBEdit компонентата може да го прикаже податокот со можност за негова промена од страна на корисникот. Кога ќе се поврзат со DataSource, се прикажуваат податоци од одредено поле на тековниот слог на табелата. Ако табелата дозволува корекции (вредноста на својството CanModify на DataSource компонентата е True), тогаш корисникот може да го промени податокот. Промената се запишува во табела кога ќе се повика методот Post на Table компонентата.

 DBMemo е компонента која се користи за прикажување на содржината на полето кое може да се менува. Таа има својства како стандардна Memo компонента, која може да прикажува и менува текст во повеќе редови. Својството AutoDisplay контролира дали податоците од DataSet се прикажуваат автоматски (True) или не (False).

 DBImage – обезбедува прикажување на слика од базата на податоци. Компонентата овозможува прикажување на сликата кога ќе побара корисникот, со поставување на соодветните својства или програмски.

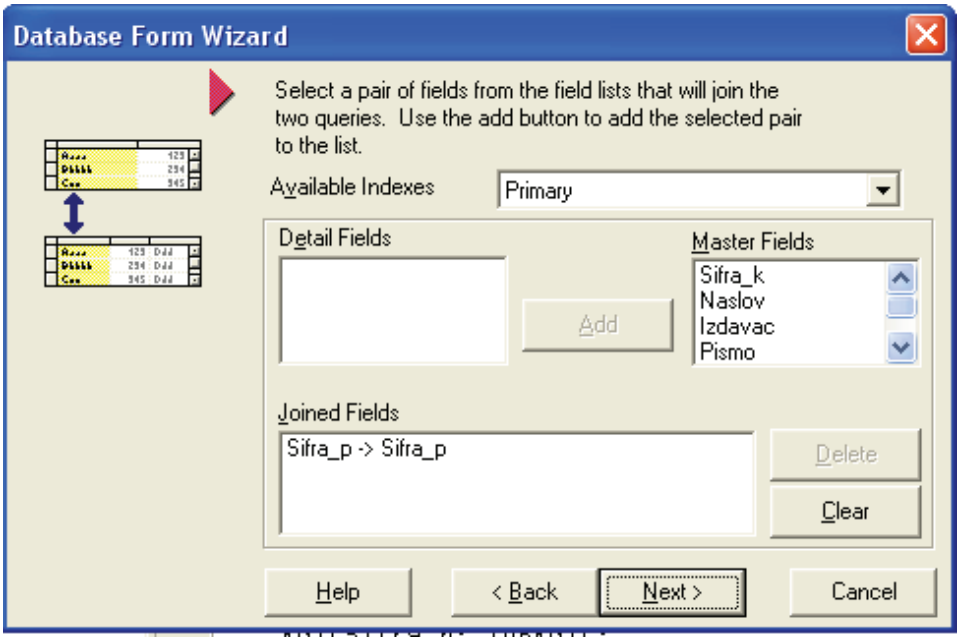
Во претходниот пример со бази формиравме алиас biblioteka и две табели – pisateli и knigi. Овие две табели може да се поврзат во однос: еден кон повеќе. Практично поврзување може да извршиме со помош на волшебник на следниот начин:

1. Од мени стартуваме: Database->FormWizard. Во првиот дијалог - прозорец, во делот за креирање форма ја бираме втората опција Create a master/detail form.
2. Следниот чекор е избор на примарна (master) табела. За таа цел прво бираме alias-> biblioteka, од прозорецот за табели табелата knigi, а потоа Next.
3. Ги одбираме полињата Sifra_K, Sifra_P и Naslov и ги додаваме во прозорецот Ordered Selected Fields, а потоа кликнуваме Next.
4. Одбираме хоризонтален распоред на компонентите
5. Бираме секундарна (detail) табела , а тоа е табелата Pisateli.
6. Ги пренесуваме полињата Sifra_P, Ime и Prezime во прозорецот Ordered Selected Fields, а потоа кликнуваме Next.
7. Ја избираме опцијата на поставување на компонентите во мрежа (In a grid).



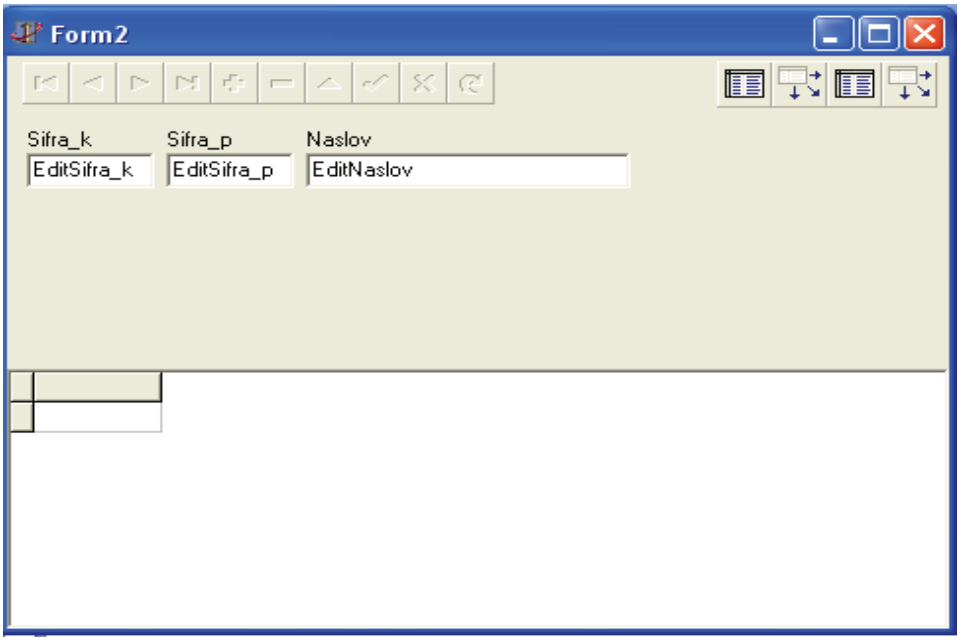
Слика 7.23.

8. На сликата гледаме дека во прозорците Detail Fields и Master Fields постојат само две исти полиња. Ги означуваме двете полиња. Меѓу нив се појавува копче Add , а по кликањето на него во долниот прозорец Joined Fields се појавува Sifra_P->Sifra_P, што значи дека табелите се поврзани според полето Sifra_P.
9. Треба да се селектира Generate a main form -> Finish.



Слика 7.24.

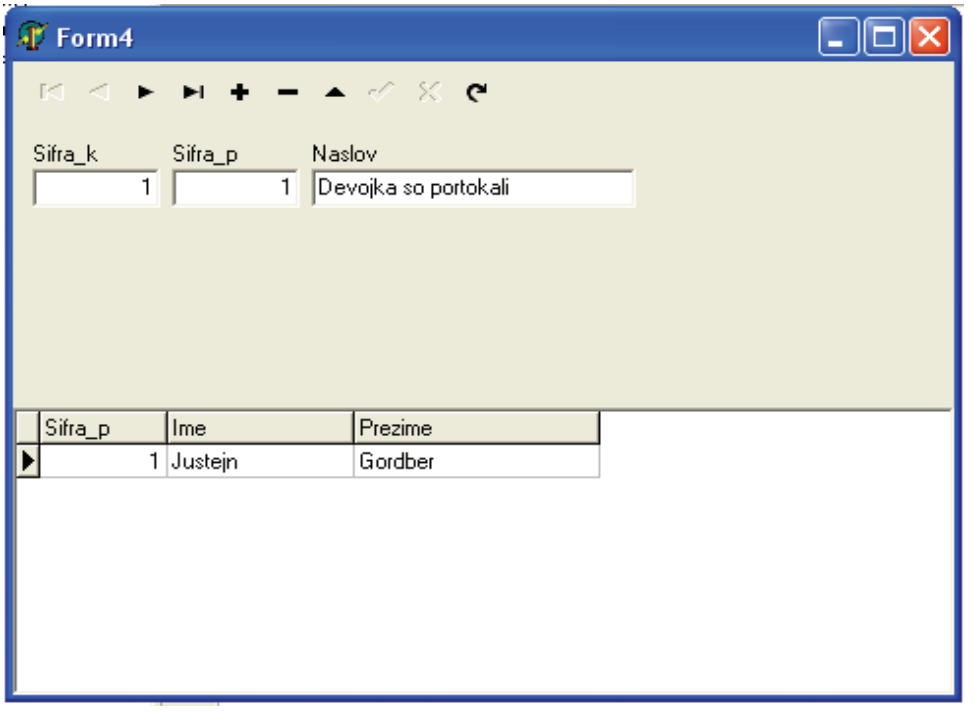
После деветтиот чекор се појавува прозорец како на слика 7.25.



Слика 7.25.

На сликата гледаме дека волшебникот ги поставил сите потребни компоненти за апликацијата (Table, DataSource, Label, DBEdit, DBNavigator, DBGrid).

По стартувањето на апликацијата се добива прозорец како на слика 7.26.



Слика 7.26.

Со користење на копчето DBNavigator може да се движиме низ примарната табела при што се прикажуваат и одбраните полиња од секундарната табела.

Во Delphi постојат бази кои тој ги инсталирал. Следниот пример е за таква база. Користиме alias DBDEMOS и табелата ANIMALS.DB.

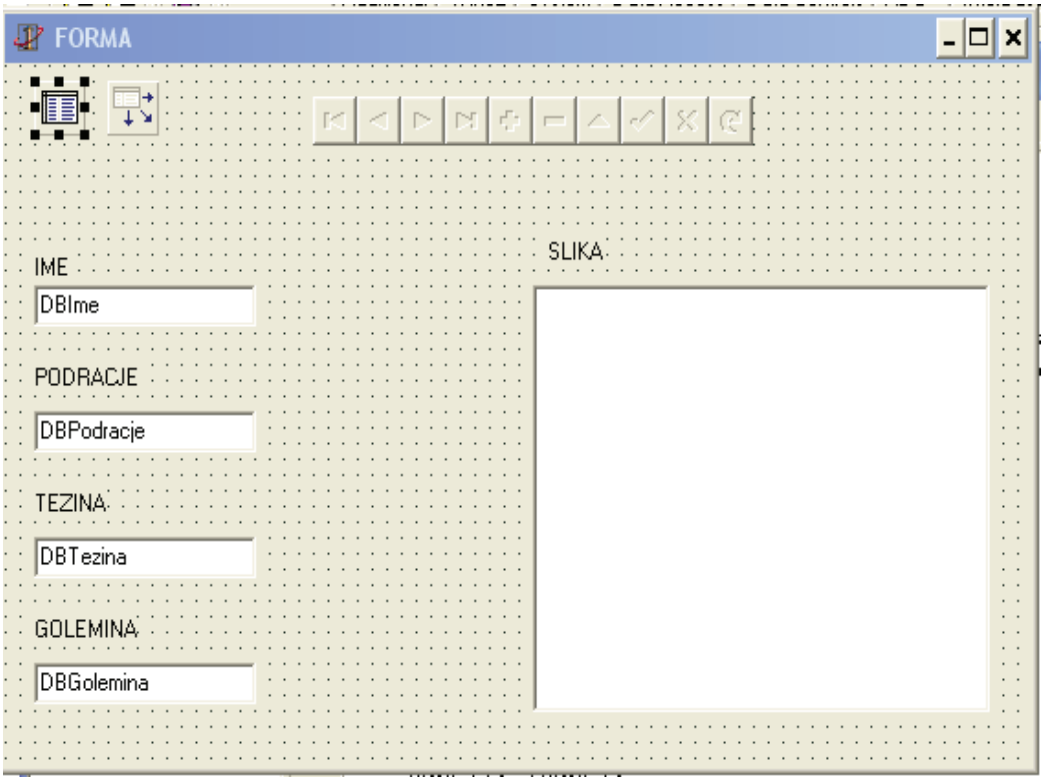
Апликацијата ќе ја прикажува содржината на табелата слог по слог. Неа ќе ја формираме во следните чекори:

1. Стартуваме нова апликација и на формата се поставува компонента Table. Во табелата на следната слика се дадени својствата кои треба да се променат за компонентите од оваа апликација.
2. На формата поставуваме компонента DataSource и во ObjectInspector се поставуваат вредностите од табелата.

Име на компонентата	Својство	Вредност
Form1	Name	Forma1
	Caption	FORMA
Table	DataBaseName	DBDEMOS
	TableName	Animals.dbf
DataSource	DataSet	Table1
DBNavigator	DataSource	DataSource1
DBEdit1	Name	DBIme
	DataSource	DtaSource1
	DataField	IME
Label1	Name	Natpis1
	Caption	IME
DBEdit2	Name	DBPodracje
	DataSource	DataSource1
	DataField	PODRACJE
Label2	Name	Natpis2
	Caption	PODRACJE
DBEdit3	Name	DBTezina
	DataSource	DtaSource1
	DataField	TEZINA
Label3	Name	Natpis3
	Caption	TEZINA
DBEdit4	Name	DbGolemina
	DataSource	DtaSource1
	DataField	GOLEMINA
Label4	Name	Natpis4
	Caption	GOLEMINA
DBImage	Name	BMPImage
	DataSource	DtaSource1
	DataField	BMP
Label5	Name	Natpis5
	Caption	SLIKA

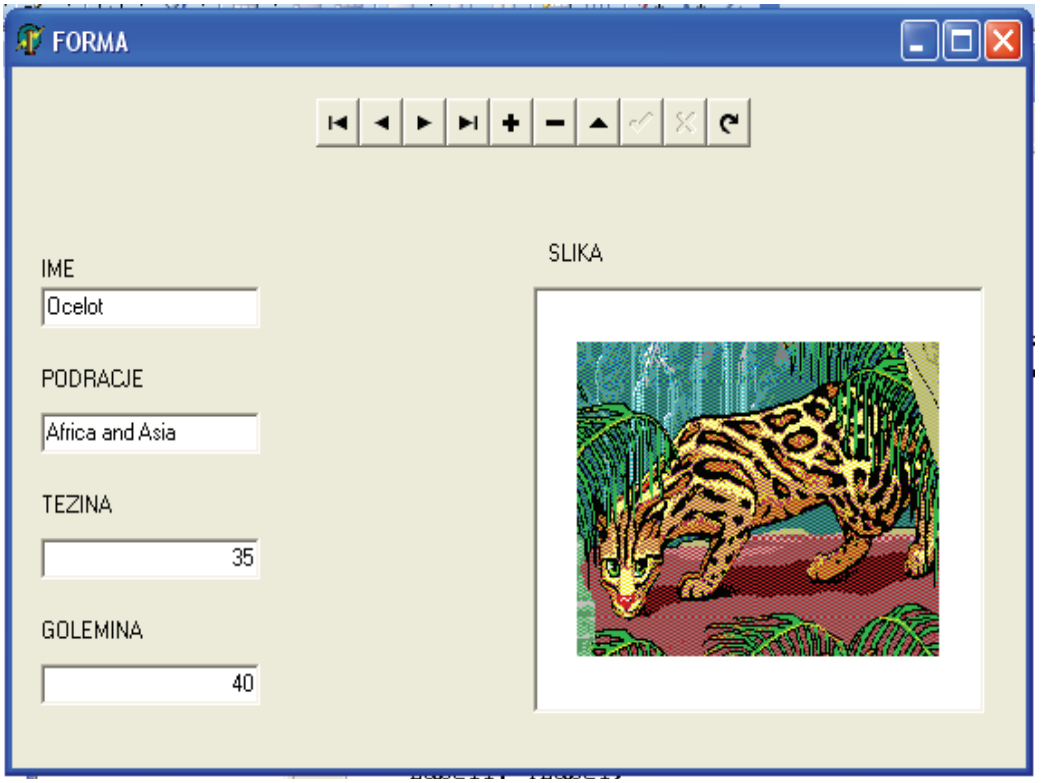
3. Се поставува компонента DBNavigator (горе лево)
4. Во левиот дел се поставуваат 4 DBEdit компоненти од страната DataControls.
5. Над секоја Edit компонента се поставуваат Label компоненти.
6. Бидејќи табелата чии слогови ги прикажуваме содржи и слики, на нашата форма треба да додадеме компонента со која ќе ја прикажеме сликата. Тоа е компонентата DBImage , а над неа се поставува лабела.

Формата на апликацијата ќе го има следниот облик (слика 7.27).



Слика 7.27.

Пред да се стартува апликацијата потребно е за компонентата Table да се промени својството Active на вредност True. Со копчињата од DBNavigator се движиме низ слоговите на базата DBDEMOS.



Слика 7.28.

7.6.3. Филтрирање на податоци

Операција која многу често се користи при работа со базите на податоци е прикажување на податоци кои задоволуваат одредени услови. Оваа операција се нарекува филтрирање.

Примерот за филтрирање може да го објасниме кај табели во база на податоци за ученици. Нека во една табела се наоѓаат имињата на учениците на еден клас, основните податоци и нивните оценки. За извештај на ниво на училиште се бараат податоци за бројот на ученици со позитивен успех, негативен успех, број на оправдани и неоправдани изостаноци.За да се овозможи добивање на DataSet само со бараните податоци се користи филтер.

За компонентата Table во ObjectInspector постојат две својства: Filtered и Filter кои овозможуваат прикажување на саканите податоци од табелата. Својството Filtered одредува дали е табелата филтрирана или не. Ако својството има вредност True , табелата ќе биде филтрирана. Критериумот по кој се врши филтрирањето се наведува во својството Filter. Вредноста на ова својство се состои од име на полето, логичкиот оператор и вредноста. При филтрирањето важат следните правила:

1. Имињата и полињата на логичките оператори може да се опишат со мали и големи букви.
2. Кога се филтрира текст, својството FilterOptions дефинира дали ќе се обрнува внимание на мали и големи букви.

Филтерот може да го има следниот изглед:

Ime=’Mihajlo’

Prezime=’Filiposki’ and GodinaR>’1/1/68’

Prezime=’FILIPOSKI’ AND GODINAR>’1/1/68’

Првиот филтер овозможува прикажување на сите слогови кои во полето Ime имаат вредност Mihajlo. Другите два примери за резултат имаат иста вредност, доколку во својството FilterOptions не се бара малите и големите букви да се разликуваат.

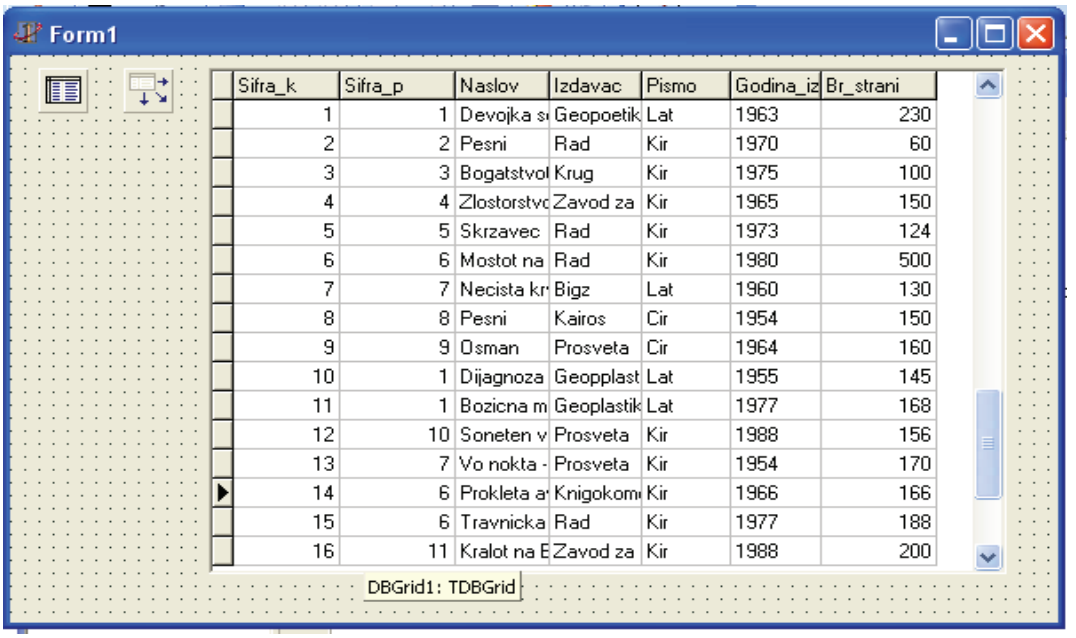
Во следната табела се прикажани логички изрази кои можат да се користат при филтрирање.

Оператор	Значење
>	Поголемо од
<	Помало од
=	Еднакво
<>	Различно
<=	Помало или еднакво
>=	Поголемо или еднакво
()	Се користи за промена на редоследот на извршување
{}	Се користи за означување на имињата на полињата кои содржат празнини
AND,OR,NOT	Логички оператори

Пример за филтрирање

1. Да се стартува нова апликација и на формата да се стави компонента Table. Својството Active се става на вредност True. DatabaseName го поставуваме на алиас biblioteka. Од базата на податоци го одбираме името TableName Knigi.
2. Ја поставуваме компонентата DataSource на формата . Неговото својство DataSet го поставуваме на вредност DataSource1.

3. На формата поставуваме DBGrid компонента, а на нејзиното својство DataSource му доделуваме вредност DataSource1.



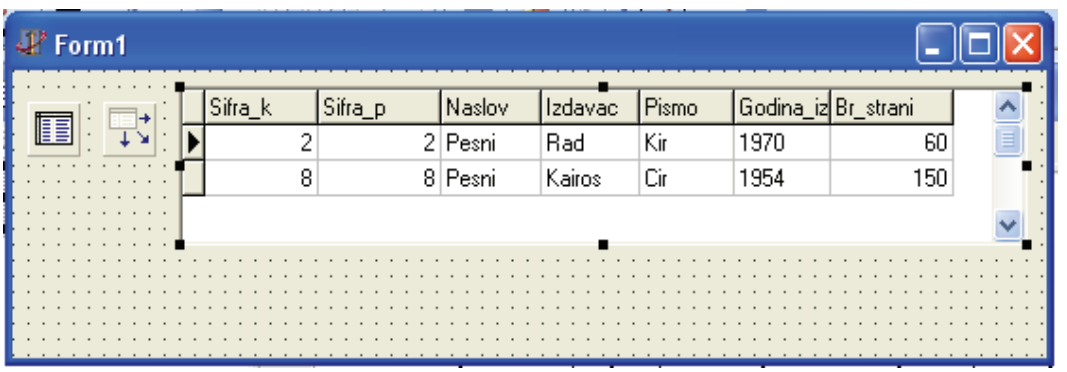
Слика 7.29.

Во овој момент формата ќе го има обликот како на слика 7.29.

Следните чекори овозможуваат филтрирање на табелата:

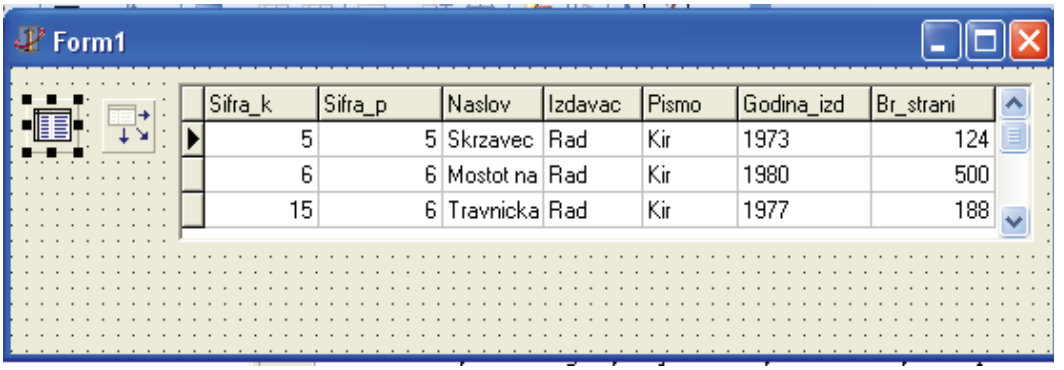
4. Во ObjectInspector на компонентата Table, во својството Filter треба да се внесе Naslov='Pesni'
5. На истата компонента , вредноста на својството Filtered да се постави на True.

Добиениот DataSet ќе изгледа како на слика 7.30.



Слика 7.30.

Ако во полето Filter запишеме Izdavac='Rad' and Godina_iz>1970, ќе добиеме DataSet како на слика 7.31.

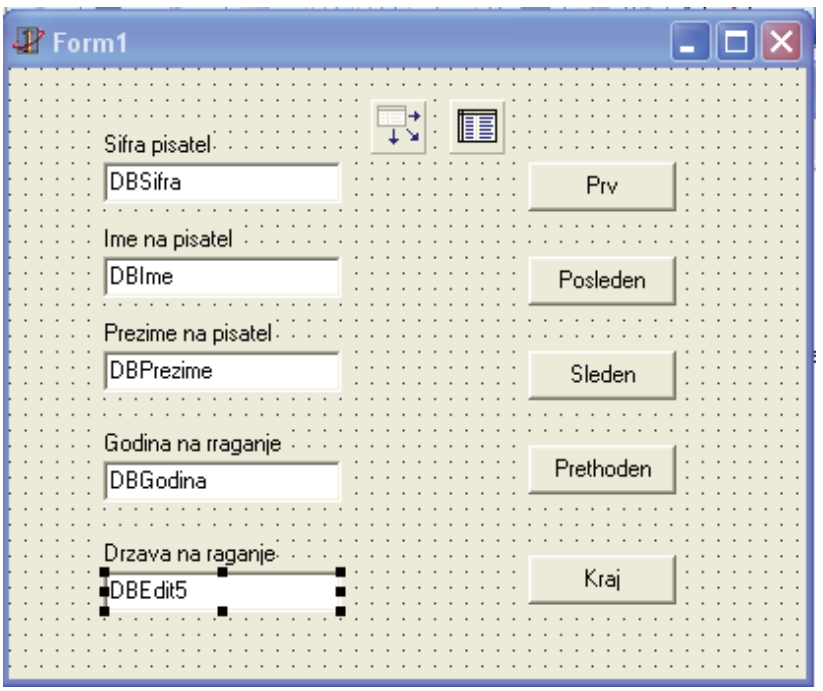


Слика 7.31.

Обработка на табела од програма

До податоците во табелата може да се пристапи и од програма. Кога компонентата Table е креирана и табелата е отворена, активен е само еден слог на табелата. Пристап е можен само до полињата од активниот слог. По отворањето на табелата активен е првиот слог. Преминот на следен, претходен, прв или последен се врши со помош на методите Next, Prior, First или Last.

За да го прикажеме движењето низ табелата користејќи програмски код, ја формираме апликацијата која од табелата на податоци Pisateli ќе ги прикажува содржините на полињата на слоговите до кои се пристапува со наведените методи. Формата го има обликот прикажан на слика 7.32.

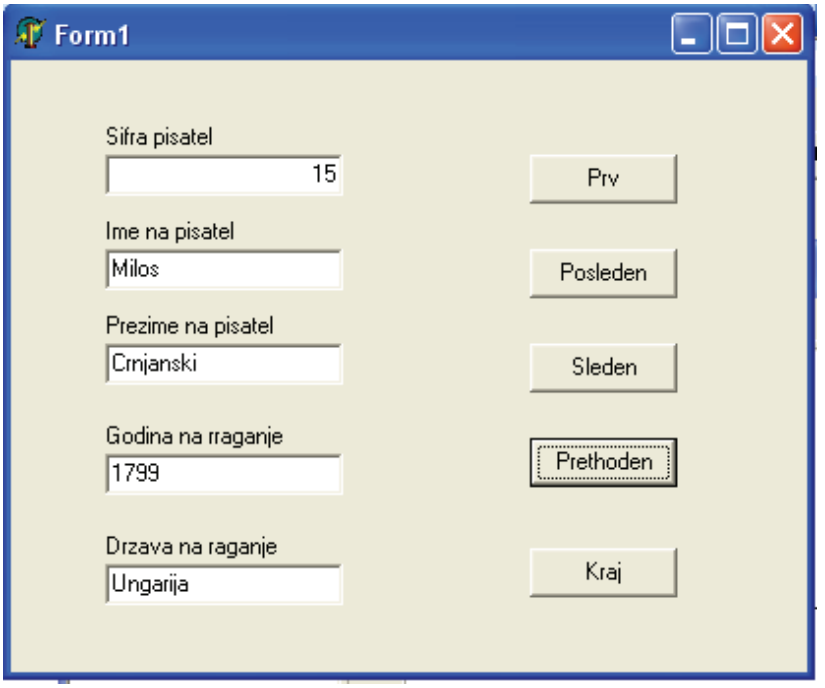


Слика 7.32.

За секое копче дефиниран е настан OnClick чии програмски кодови се прикажани во следниот Листинг.

```
procedure TForm1.Button5Click(Sender: TObject);
begin
application.terminate;
```

```
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
        Table1.active:=true;
        Table1.first;
        DBsifra.Text:=Table1.Fields[0].Value;
        DBIme.Text:=Table1.Fields[1].Value;
        DBPrezime.Text:=Table1.Fields[2].Value;
        DBGodina.Text:=Table1.Fields[3].Value;
        DBDrzava.Text:=Table1.Fields[4].Value;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
        Table1.active:=true;
        Table1.last;
        DBsifra.Text:=Table1.Fields[0].Value;
        DBIme.Text:=Table1.Fields[1].Value;
        DBPrezime.Text:=Table1.Fields[2].Value;
        DBGodina.Text:=Table1.Fields[3].Value;
        DBDrzava.Text:=Table1.Fields[4].Value;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
        Table1.active:=true;
        Table1.next;
        DBsifra.Text:=Table1.Fields[0].Value;
        DBIme.Text:=Table1.Fields[1].Value;
        DBPrezime.Text:=Table1.Fields[2].Value;
        DBGodina.Text:=Table1.Fields[3].Value;
        DBDrzava.Text:=Table1.Fields[4].Value;
end;
procedure TForm1.Button4Click(Sender: TObject);
begin
        Table1.active:=true;
        Table1.prior;
        DBsifra.Text:=Table1.Fields[0].Value;
        DBIme.Text:=Table1.Fields[1].Value;
        DBPrezime.Text:=Table1.Fields[2].Value;
        DBGodina.Text:=Table1.Fields[3].Value;
        DBDrzava.Text:=Table1.Fields[4].Value;
end;
end.
```



Слика 7.33.

При извршување на програмата ќе се појави прозорецот од слика 7.33.

Во програмскиот код имаме четири слични делови. Секој дел го дефинира настанот за соодветниот метод на движење низ табелата. На формата е поставена соодветна табела и DataSource. Табелата во програмата можеме да ја отвориме со поставување на методот Active на вредност True или со повикување на методот Open . Записите во програмскиот код

```
Table1.Active:=True;
Table1.Open
```

имаат исто значење. Доколку во апликацијата не се менуваат својствата DatabaseName и TableName, не мора да се користи методот на затворање на отворената табела, затоа што тоа се случува автоматски на крајот на работата на апликацијата. Во спротивно, активната табела мора да се затвори со користење на една од следните методи:

```
Table1.Active:=False;
Table1.Close.
```

Записот Table1.Prior овозможува пристап на претходниот слог во однос на тековниот. Слично, записот Table1.Next овозможува пристап на следниот запис, Table1.Last пристап на последниот запис и Table1.First пристап на првиот запис во табелата на податоци.

Програмските редови:

```
DBsifra.Text:=Table1.Fields[0].Value;
DBIme.Text:=Table1.Fields[1].Value;
DBPrezime.Text:=Table1.Fields[2].Value;
DBGodina.Text:=Table1.Fields[3].Value;
DBDrzava.Text:=Table1.Fields[4].Value;
```

обезбедуваат пристап до вредностите на полињата на активниот слог од табелата на базата на податоци и прикажување на нивните содржини во соодветни DBEdit компоненти. Се забележува дека полето до кое се пристапува е означено со

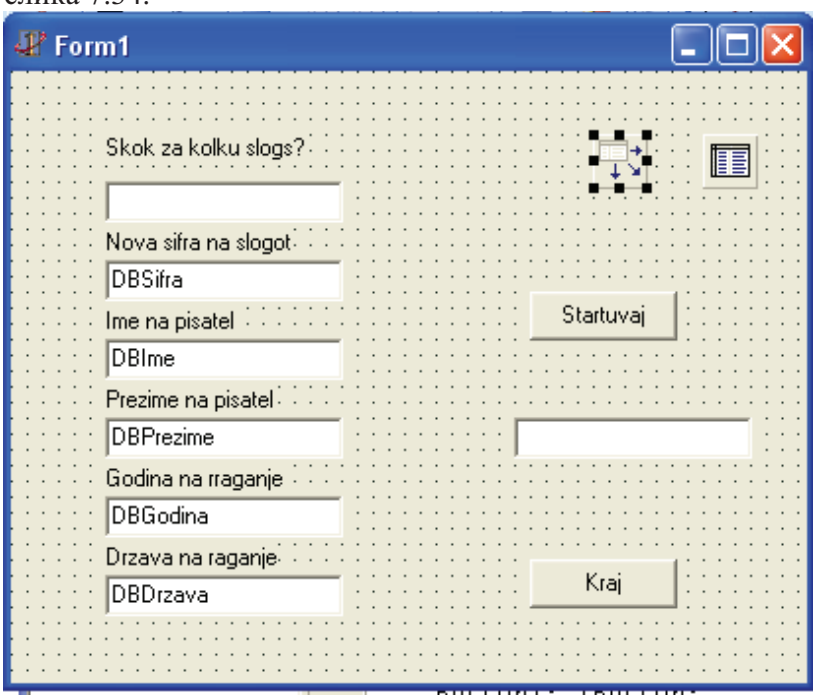
положбата, односно редниот број во слогот. Првото поле има реден број нула (0).

Откако ќе се стартува апликацијата, се добива работен прозорец како на слика 7.33.

Покрај наведените можности на движење низ табелата, Delphi овозможува поместување за одреден број слогови во однос на тековниот слог.

Пример 1: Да се направи апликација која во базата на податоци biblioteka, користејќи ја табелата Pisateli.DB, ќе овозможи премин за зададен број слогови во однос на тековниот број на слогот и ќе го извести корисникот кога ќе се дојде на првиот или на последниот слог во табелата.

Формата за горенаведената апликација ќе го има обликот прикажан на слика 7.34.



Слика 7.34.

Програмскиот код со кој е реализирана апликацијата е следниот:

```
Procedure TForm1.FormCreate(Sender: TObject);
begin
        Table1.open;
end;
```

Во оваа апликација за разлика од претходната, табелата се отвора во моментот на креирање на формата.

```
Procedure TForm1.Button1Click(Sender: TObject);
Var
  n,i:Integer;
begin
        Val(Broj.text,n,i);
        posleden.text:=' ';
        Table1.MoveBy(n);
        DBsifra.Text:=Table1.Fields[0].Value;
```

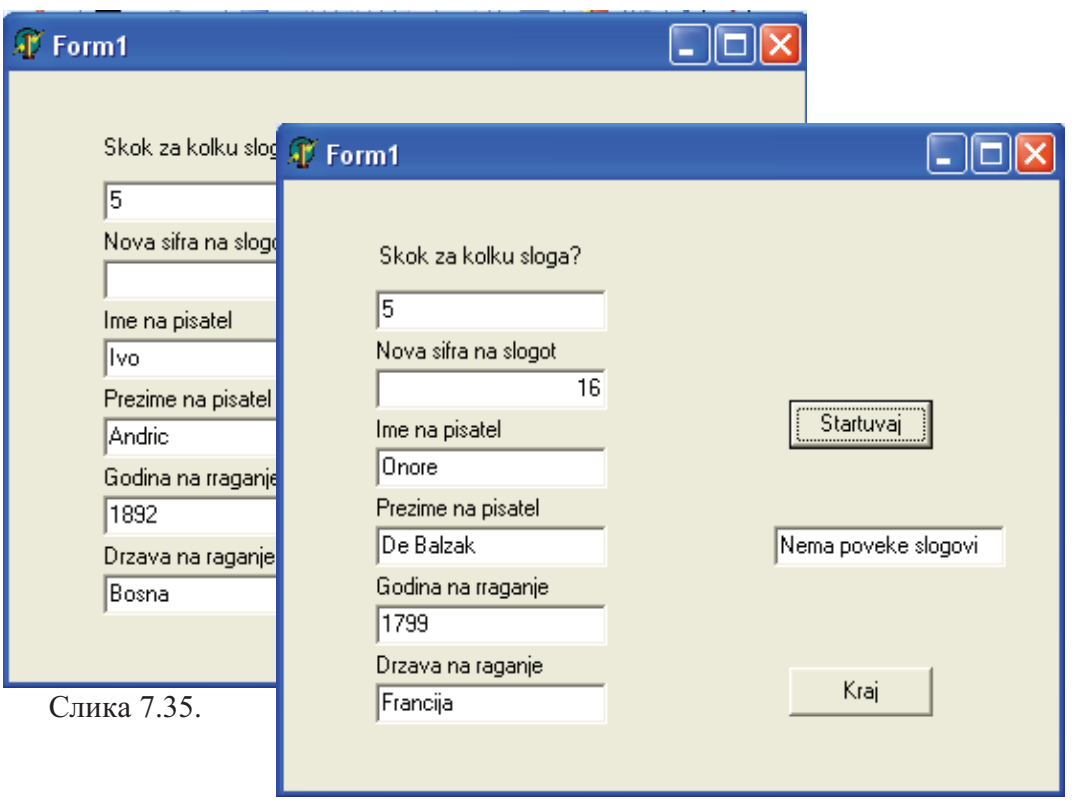
```
            DBIme.Text:=Table1.Fields[1].Value;
            DBPrezime.Text:=Table1.Fields[2].Value;
            DBGodina.Text:=Table1.Fields[3].Value;
            DBDrzava.Text:=Table1.Fields[4].Value;
            if (table1.eof) or (table1.Bof) then
                  Posleden.text:='Nema poveke slogovi';
end;
Procedure TForm1.Button5Click(Sender: TObject);
begin
            application.terminate;
end;
end.
```

Програмскиот ред **Table1.MoveBy(n);** овозможува скок на слог оддалечен N слогови од тековниот. Ако N е со негативен предзнак , движењето се врши кон првиот слог.

Другиот програмски ред ја содржи следната наредба:

```
            if (table1.eof) or (table1.Bof) then
                  Posleden.text:='Nema poveke slogovi';
```

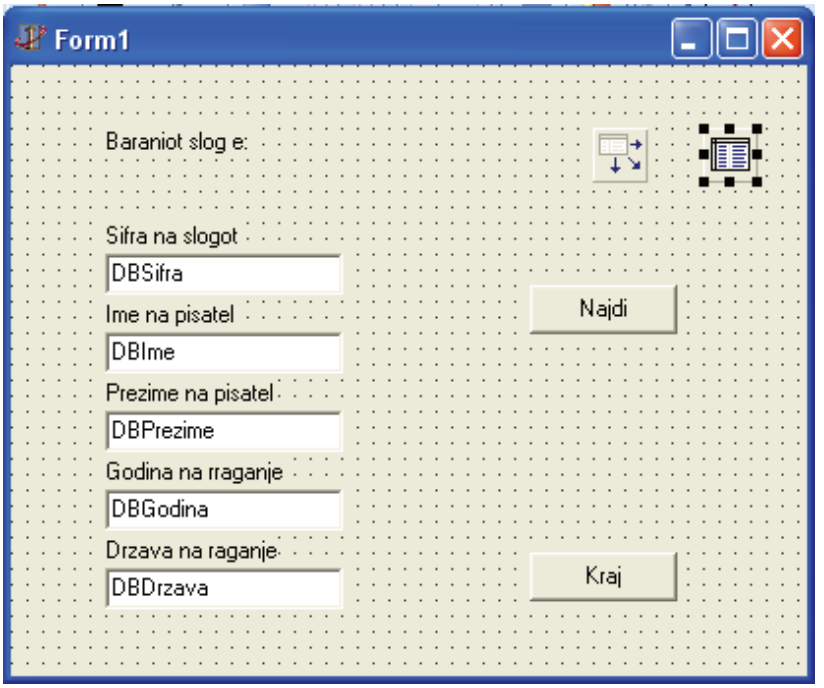
каде се користат својствата Bof и Eof кои овозможуваат контрола на активирање на првиот или последниот слог на табелата. Својствата Bof и Eof имаат вредност True ако тековниот слог е прв, односно последен.



Слика 7.35.

Во Delphi постои можност директно да се пристапи до слогот преку полињата и вредностите на полињата (едно или повеќе полиња). За таа цел се користи методот Locate. Овој метод го пронаоѓа првиот слог кој ги задоволува наведените услови. Ако во нашата табела Pisateli, сакаме да го пронајдеме слогот

кој во полето Ime има содржина Ivan , треба да се направи следната програма, а изгледот на формата е прикажан на слика 7.36.



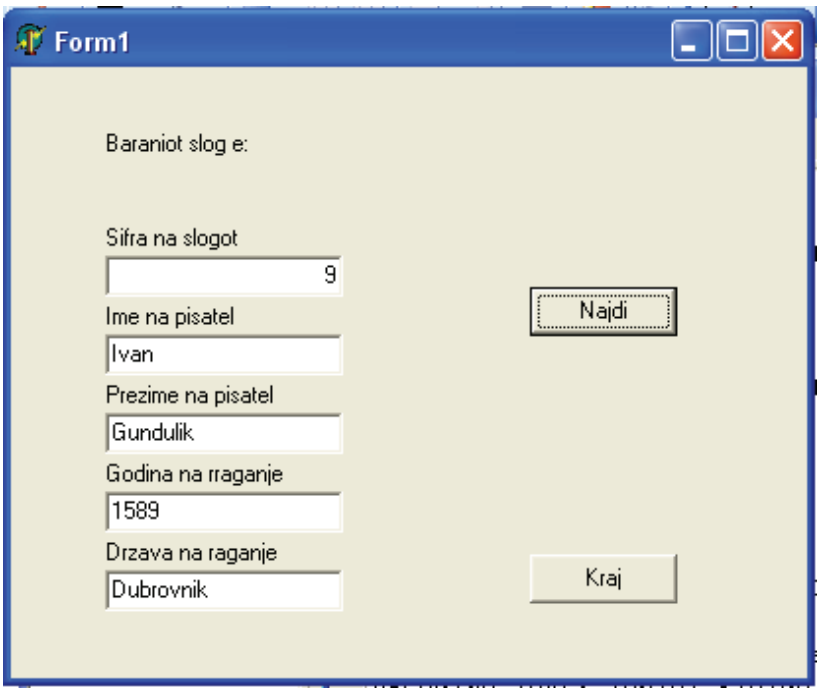
Слика 7.36.

Програмскиот код е следниот:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
        Table1.Active:=true;
end;
procedure TForm1.Button1Click(Sender: TObject);
Var
  O:TLocateOptions;
begin
If Table1.Locate('Ime','Ivan',O) then
  begin
    DBsifra.Text:=Table1.Fields[0].Value;
    DBIme.Text:=Table1.Fields[1].Value;
    DBPrezime.Text:=Table1.Fields[2].Value;
    DBGodina.Text:=Table1.Fields[3].Value;
    DBDrzava.Text:=Table1.Fields[4].Value;
end
  else MessageBox(Handle,'Slogot ne e pronajden','Message',MB_OK);
end;
procedure TForm1.Button5Click(Sender: TObject);
begin
        application.terminate;
end;  end.
```

Во кодот забележуваме декларација на променливата O, како посебен вид променлива од типот TlocateOptions, која е потребна за извршување на методот Locate. Доколку записот е пронајден во табелата, методот Locate враќа вредност True која се користи во условната наредба If.

По стартувањето на апликацијата се добива прозорец како на слика 7.37.



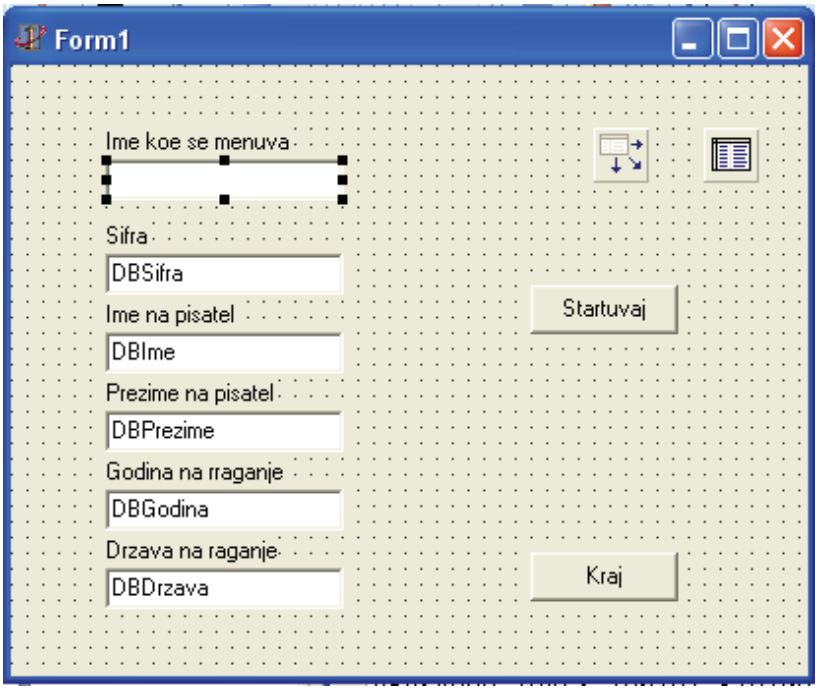
Слика 7.37.

Ако во програмскиот ред If Table1.Locate('Ime','Ivan',O) then, знаковниот податок 'Ivan' го замениме со 'Iva', ќе го добиеме следниот одговор на апликацијата:

Слика 7.38.

За да го промениме полето во слогот од табелата, потребно е најпрво да се постави покажувачот на слогот чии полиња сакаме да се променат, а потоа да се повика методот за менување Edit . Во следната апликација во табелата pisateli.DB ќе го промениме името Ivo во Iva.

Изгледот на формата прикажан е на слика 7.39.



Слика 7.39.

Програмскиот код за оваа апликација е следниот:

```
procedure TForm1.Button1Click(Sender: TObject);
Var
  s:String;
  O:TLocateOptions;
begin
        s:=Ime1.text;
        Table1.Locate('Ime',s,O);
        DBsifra.Text:=Table1.Fields[0].Value;
        DBIme.Text:=Table1.Fields[1].Value;
        DBPrezime.Text:=Table1.Fields[2].Value;
        DBGodina.Text:=Table1.Fields[3].Value;
        DBDrzava.Text:=Table1.Fields[4].Value;
        table1.edit;
        table1.Fields[1].value:='Iva';
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
        Table1.active:=true;
end;
```

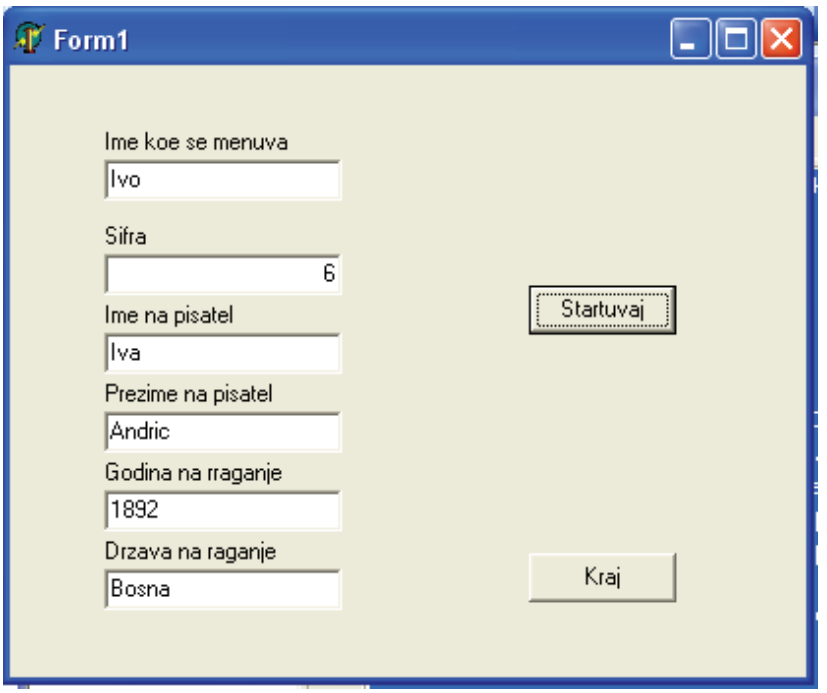
Наредбите

```
        s:=Ime1.text;
        Table1.Locate('Ime',s,O);
```

ни овозможуваат да го пронајдеме слогот чие поле се менува, врз основа на зададената вредност за Ime во Edit контролата. Потоа се применува методот Edit и се задава новата содржина на полето која треба да се промени.

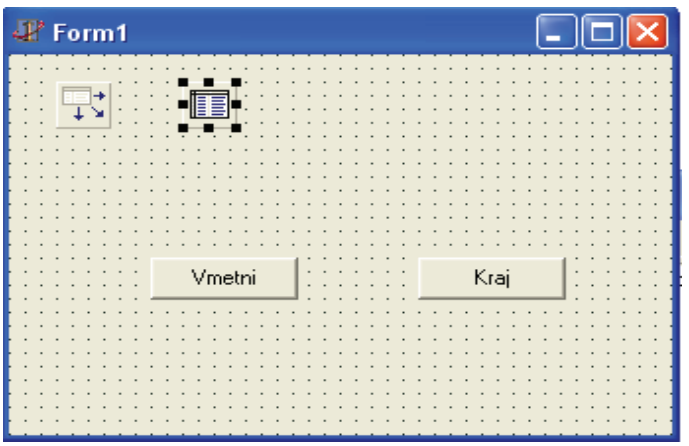
```
        table1.edit;
        table1.Fields[1].value:='Iva';
```

Откако ќе се изврши апликацијата, се добива прозорец како на слика 7.40.



Слика 7.40.

Пример: Ако сакаме да вметнеме нов слог во табелата , тогаш треба да го користиме методот Insert. Апликацијата која вметнува слогови во табела може да го има изгледот на формата како на слика 7.41.



Слика 7.41.

Програмскиот код со кој се извршува операцијата на вметнување на слогови во табелата pisateli е следниот:

```
procedure TForm1.Button1Click(Sender: TObject);
  Var  t:String;
begin
   With Table1 do
begin
    Insert;
    FieldByName('Sifra_P').AsInteger:=22;
    FieldByName('Ime').AsString:='Iva';
```

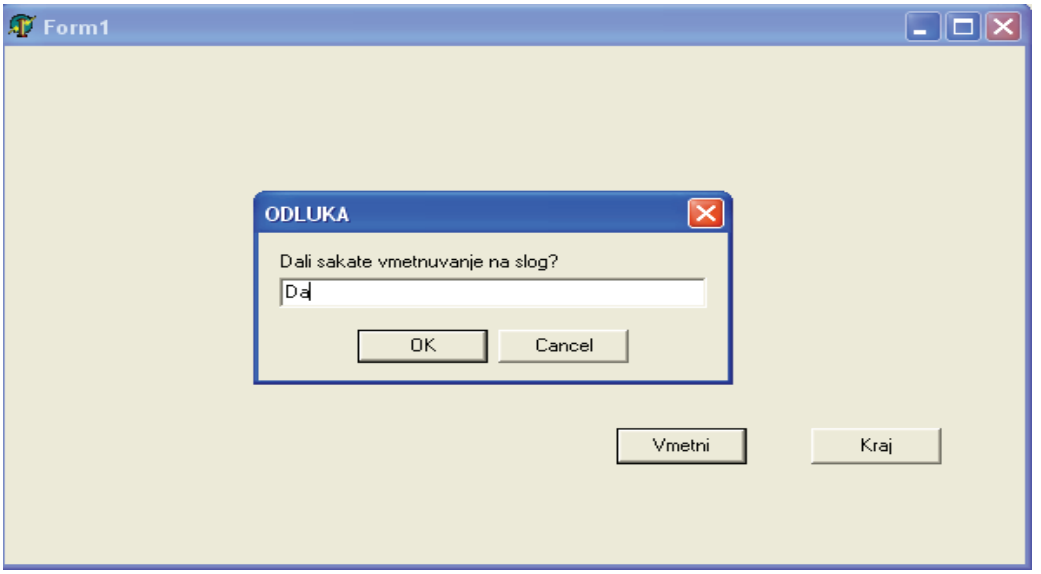
```
    FieldByName('Prezime').AsString:='Kovacevic';
    FieldByName('Godina_R').AsString:='1976';
    FieldByName('Drzava').AsString:='Srbija';
end;
 t:=inputBox('ODLUKA','Dali sakate vmetnuvanje na slog?','Ne');
 if t='Da'
   then Table1.Post   else Table1.Cancel;
 end;
procedure TForm1.FormCreate(Sender: TObject);
begin
  table1.open;
end;
```

Во овој програмски код забележуваме нов начин на пристап до табелата,како и до полињата на табелата. Тој е овозможен со употреба на наредбата :

```
With Table1 do.
```

Новиот начин на пристап до полињата во табелата се извршува врз основа на името на полето на слогот наместо со користење на редниот број на полето во слогот.

Прозорецот кој се добива кога апликацијата се извршува е прикажан на слика 7.42.



Слика 7.42.

Во работата со базите на податоци често е потребно да се избришат подделни слогови од активната табела. За таа цел се користи методот Delete. Најпрво слогот кој се брише се прогласува активен, а потоа се активира методот Delete. Апликацијата која го брише слогот кој го вметнавме во претходниот пример, може да ја има формата прикажана на слика 7.43.

Слика 7.43.

Како што се гледа, апликацијата е збогатена со можност за прикажување на слогот кој се брише. Програмскиот код на оваа апликација е следниот:

```
procedure TForm1.Button1Click(Sender: TObject);
Var
        t:String;
begin
      With Table1 do
         begin
           SetKey;
           FieldByName('Sifra_P').AsInteger:=22;
           GoToNearest;
            DBEdit1.text:= FieldByName('Sifra_P').AsString;
            DBEdit2.text:= FieldByName('Ime').AsString;
            DBEdit3.text:= FieldByName('Prezime').AsString;
            DBEdit4.text:= FieldByName('Godina_R').AsString;
            DBEdit5.text:= FieldByName('Drzava').AsString;
        end;
   t:=inputBox('ODLUKA','Dali sakate brisenje na slog?','Ne');
     if t='Da'   then Table1.Delete
       else Table1.Cancel;
 end;
procedure TForm1.Button2Click(Sender: TObject);
begin
         application.terminate;
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
         table1.open;
end;  end.
```

Во овој програмски код се користат три нови методи:

Методот SetKey овозможува поставување клуч на активната табела.

Вториот метод GoToNearest, го лоцира слогот во табелата чија вредност е најблиска до поставената вредност во наредбата над GoToNearest методот.

Третиот метод е Delete и тој извршува бришење на слогот од табелата. Овој метод треба да се користи внимателно, затоа што со негова примена податоците се губат неповратно.

ПОГЛАВЈЕ 7
(НАКРАТКО)

Со групирање на сродни податоци во облици погодни за користење, настануваат **базите на податоци**. Основна карактеристика на релационите бази на податоци е тоа што информациите се поделени во логички множества на податоци кои се сместени **во табели**. Табелите се основни објекти на релационите бази на податоци.

Најмала количина на информации во табелата е **поле - колона.**

Збирот на сите полиња во еден ред на табелата се нарекува **слог.**

За да се поврзат табелите во базата на податоци, референтната табела мора да има **примарен клуч.** Тоа е поле (или збир на полиња) кое **еднозначно** го одредува секој слог во табелата. Два слога во табелата не можат да имаат иста вредност во полето на примарниот клуч. Примарниот клуч овозможува **сите слогови од една табела да бидат поврзани со записи од друга табела.** Во секундарната табела полето со кое се врши поврзување, се нарекува **надворешен клуч** и не мора да содржи единствена вредност.

Индекс е поле кое се користи за логичко подредување на слоговите во табелата.

За да им овозможи пристап на базите на податоци, Delphi ја користи технологијата BDE**(Borland Database Engine)**. Таа е **збир на услужни програми** кои овозможуваат пристап до разни бази на податоци.

Сите верзии на **Delphi** имаат драјвер кој поддржува пристап на **Paradox** i **dBASE бази** на податоци.

За коректен пристап до базата BDE користи **алиас (alias).** Тоа е **збир на параметри** кои го опишуваат начинот на поврзување со базата на податоци. Алиасот му кажува на BDE кој драјвер да го употреби и каде на дискот се наоѓаат датотеките со базата на податоци.

Компонентите за работа со базите можат да се поделат на **невизуелни и визуелни.**

Создавање на база на податоци со користење на програмата DatabaseDesktop:

-се формира алиас,

-се креира табела,

-се пополнува табелата.

Проектирање на апликации за работа со бази на податоци со користење на волшебник.

Компоненти за врска со табелата на базата на податоци

Table

DataSource

DBGrid

Компонентата DBNavigator му овозможува на корисникот контрола над операциите запишување, бришење , промени и движење низ слоговите на базата на податоци.

Компоненти за прикажување и менување на податоци во базата : DBText, DBEdit, DBMemo, DBImage

Филтрирање на податоци

Операција која многу често се користи при работа со базите на податоци е прикажување на податоци кои задоволуваат одредени услови.

Обработка на табела од програма

До податоците во табелата може да се пристапи и од програма. Кога компонентата Table е креирана и табелата е отворена, активен е само еден слог на табелата. Пристап е можен само до полињата од активниот слог. По отворањето на табелата активен е првиот слог. Преминот на следен, претходен, прв или последен се врши со помош на методите **Next, Prior, First или Last.**

Во Delphi постои можност директно да се пристапи до слогот преку полињата и вредностите на полињата (едно или повеќе полиња). За таа цел се користи методот **Locate**.

За да го промениме полето во слогот од табелата, потребно е најпрво да се постави покажувачот на слогот чии полиња сакаме да се променат, а потоа да се повика методот за менување **Edit.**

Ако сакаме да вметнеме нов слог во табелата , тогаш треба да го користиме методот **Insert.**

Во работата со базите на податоци често е потребно да се избришат подделни слогови од активната табела. За таа цел се користи методот **Delete.**

Прашања и задачи:

1. Што е локална база на податоци?
2. Која е улогата на BDE во Delphi?
3. Објасни ги разликите меѓу табела и DataSet.
4. Што е алиас?
5. Опишете на кој начин заеднички функционираат вашата програма, BDE и базата на податоци.
6. Креирајте нов BDE алиас , а потоа креирајте припаѓачка база на податоци која ќе има потребен број на табели со податоци за деведетека.
7. Во Object Inspector, DBEdit компонентата нема својство Text. На кој начин може да се пристапи до содржината на оваа компонента?
8. Кои се начините на кои може да се движите низ слоговите на табелата на базата на податоци?
9. На кој начин може да се контролира распоредот и бројот на колони кој ќе биде прикажан во DBGrid компонентата?
10. Кои компоненти можат да се користат за DataSet да се прикаже во табеларен облик?
11. На кој начин поодделни копчиња на компонентата DBNavigator можат да се направат неактивни?
12. Која компонента можеме да ја користиме за прикажување и измена на текст - полето во табелата на базата на податоци?
13. Да се напише апликација за прикажување на табелата filmovi од базата на податоци која претходно сте ја направиле (слика 7.44).
14. Направете апликација која ќе овозможи исправка, вметнување и бришење на слогови во табелата klienti во базата на податоци (слика 7.45.).

film : Table

	F_ID	F_naslov	F_zhanr	F_glavna_uloga	F_format	F_godina	F_jazik	F_cena
+	1	Balkan Kan	Drama	Vlado Jovanovski	DVD	2005	Makedon	100.00 ден.
+	2	Pred dozdot	Drama	Rade Serbedzija	DVD	1998	Makedon	100.00 ден.
+	3	Sam doma 3	Komedija	Mekoli Kalkan	VHS	1990	Angliski	100.00 ден.
+	4	Smrtonosno oruzje	Akcija	Mel Gibson	VHS	1993	Angliski	100.00 ден.
+	5	So glava vo zid	Drama	Maks Klajn	DVD	2005	Angliski	50.00 ден.
+	6	Barajki go Nemo	Animiran	Nemo	DVD	2004	Angliski	100.00 ден.
+	7	Javac na grobnici	Akcija	Angelina Jolie	DVD	2004	Angliski	50.00 ден.
+	8	Odneseno so viorot	Drama	Vivien Li	VHS	1937	Angliski	50.00 ден.
+	9	Kazablanka	Avantura	Hemfri Bogart	VHS	1944	Angliski	100.00 ден.
+	10	Mis Ston	Istoriski	Petre Prlicko	VHS	1957	Makedon	50.00 ден.
*	umber)					0	Angliski	0.00 ден.

Слика 7.44 Табела Филмови

	K_ID	K_Ime	K_Prezime	K_Adresa	K_Telefon	K_E-mail	K_Licna_k	K_EMB	K_Pol	K_Chlen	K_Zachlenuv
+	1	Rade	Najdovski	"Debarska" 15/3/17	3237019	rade@yahoo.com	2115342	1705987400123	mas	3783	12/23/2003
+	2	Jovana	Gjorgjieva	"Ohridska" 23	3074012	jove@gmail.com	1552432	2111957500176	zen	2345	7/15/2005
+	3	Maja	Ancevska	"Ilindenska" 74/2	2738202	maja@yahoo.com	1935478	2707936370001	zen	1872	2/28/2004
+	4	Mice	Krstevski	"Prolet" 34/2/7	5729372	mice@hotmail.com	2007945	3104968527233	mas	754	11/6/2004
+	5	Igor	Velovski	"Partizanska" 54/1/5	3124502	igor_v@goowy.com	2123877	0707977537862	mas	23	10/25/2005
*	nber)						0			0	

Слика 7.45 Табела Клиенти

	P_ID	K_ID	F_ID	P_data_iznajmu	P_data_vrakjanj
	1	1	4		
	2	3	1		
	3	4	7		
	4	1	3		
	5	2	5		
	6	3	8		
	7	1	10		
	8	3	9		
	9	2	2		
	10	1	6		
	11	4	5		
	12	5	4		
	13	3	3		
	14	1	7		
	15	2	8		
	16	3	10		
*	(AutoNumber)	0	0		

Слика 7.46. Табела Поврзување

Одговори

Поглавје 1

1. Командна линија е одлика на DOS оперативен систем, а прозорец и работна маса (desktop) се одлика на Windows оперативниот систем.
2. Windows работната околина се одликува со икони, а DOS со наредби
3. Повеќепроцесен систем (multitasking).
4. Визуелен и Код програмски чекор.
5. Дека системот едновремено можат да го користат повеќе корисници.
6. Енкапсулација, наследување и полиморфизам.
7. Својства.
8. Метод.

Поглавје 2

1. Ако во Object Inspector се промени својството Name, во програмскиот код се менуваат имињата на процедурите.
2. Главен прозорец (main),Object Inspector,прозорец на образец или форма, прозорец за пишување на програмски код
3. **Edit e** компонента за внесување текст само во еден ред. На внесениот текст му се придружува својството **Text.**
4. **Enabled** – логичка вредност која укажува на тоа дали контролата може да прима настани од тастатура или глувче. Ако е со вредност false , контролата ќе биде бледосива и не може да се внесуваат податоци.

Read Only – корисникот има можност само да чита податоци од таа контрола, а не и да пишува во неа.

Set Focus – го поставува фокусот (покажувачот) на контролата која ја повикува.

Password Char – го дефинира знакот со кој ќе се менува внесениот текст, се употребува кога се креираат лозинки.

5. Програмски и преку Object Inspector.
6. Properties, Events.
7. Label контролата овозможува појавување на статичен текст на екранот.
8. Настанот On Click се јавува кога ќе се кликне на компонентата.
9. procedure TForm2.Button2Click(Sender: TObject);
10. Се отвора уредувачот на код со името на соодветната процедура која го содржи името на формата, името на компонентата и името на настанот.
11. **Интерпретер** е преведувач, кој секоја линија од изворниот код ја преведува на машински јазик и ја извршува.**Компајлерот** е програма која секоја наредба од изворниот код ја анализира и преведува на машински код, односно формира извршна програма.
12. Со соодветна конверзија.
13. **On Enter** –настапува кога контролата добива фокус за внес на податоци.

On Change – настанува секогаш кога ќе се измени содржината (text) на Edit контролата. Користете го настанот за обработка на внесени податоци од страна на корисникот кога ќе настапи тој настан.

Задачи

4. **Password Char (edit2) <=*;**

```
procedure TForm1.Button2Click(Sender: TObject);
begin
edit2.visible:=true;
edit2.SetFocus;
if edit2.text='emuc' then
```

application.terminate;
end;
5. procedure TForm1.Edit1Change(Sender: TObject);
begin
edit2.text:=edit1.text;
end;

Поглавје 3

1. procedure TForm1.FormCreate, procedure TForm1.Button1Click.
2. Преку картичката настани (events ->Object Inspector), се избира настанот On Click или со двојно кликнување врз командното копче .

-IntToStr (претворање од цел број во string), StrToFloat (претвора string – текст во децимален број), StrToInt (претвора string – текст во цел број), FloatTo Str (претворање од децимален број во string)

3. procedure Tform1.Button1Click(Sender: TObject);
4. Promenliva := izraz;

каде што Promenliva е место во меморијата во која се сместува резултатот, а Izraz е изразот чија вредност се пресметува.

5. CheckBox.
6. Контролата **TCheckBox се** наоѓа на страната Standard. Се користи при решавање задачи кога е потребно да се овозможи вклучување или исклучување на една или повеќе опции .
7. If checkBox1.checked then
   Label1.caption:=’Kliknato’
   else
   Label1.caption:=’Ne e kliknato’
8. Објасни ги програмските редови:

-StringGrid1.Row:=ComboBox1.itemindex+1; - бројот на редови на компонентата StringGrid ќе биде еднаков со бројот на податоци во ComboBox компонентата.
-Edit1.SetFocus; - фокусот го поставува на Edit1 компонентата.
-StringGrid1.Cells[0,1]:='1.'; - во ќелијата од редот 0 и колоната 1 впишува „1“.

9. **RadioButton** контролата се користи за избор на една од неколку можни опции.
   If radiobutton1.checked then
   Label1.caption:=’Kliknato’
   else
   Label1.caption:=’Ne e kliknato’
10. -Items – овозможува да се дефинира, чита и менува списокот на натписи и бројот на радиокопчиња кои се наоѓаат во рамките на RadioGroup,
    -ItemIndex – го одредува индексот на моментно активниот Radio Button, во рамките на RadioGroup. Првото копче има индекс 0. Ако ни едно копче не е селектирано, индексот е -1.
11. Б)
12. Преку својство Lines , во извршен мод директно во компонентата, снимање од датотека.
13. Не.
14. Со својство се оневозможува прикажување на текстот во повеќе редови.
15. StringGrid компонента .
16. Содржината на ќелијата е дефинирана како знаковен податок, но на секоја ќелија, освен текст, може да се придружи кој било објект кој постои во Delphi.

17. „ 0“
18.

а)	For do	В	структура повторување со услов на крај на циклусот,
б)	While do	Б	структура повторување со услов на почеток на циклусот,
в)	Repeat – until	-	структура повторување со скок на одреден програмски ред
		А	Структура повторување со броење на циклусите

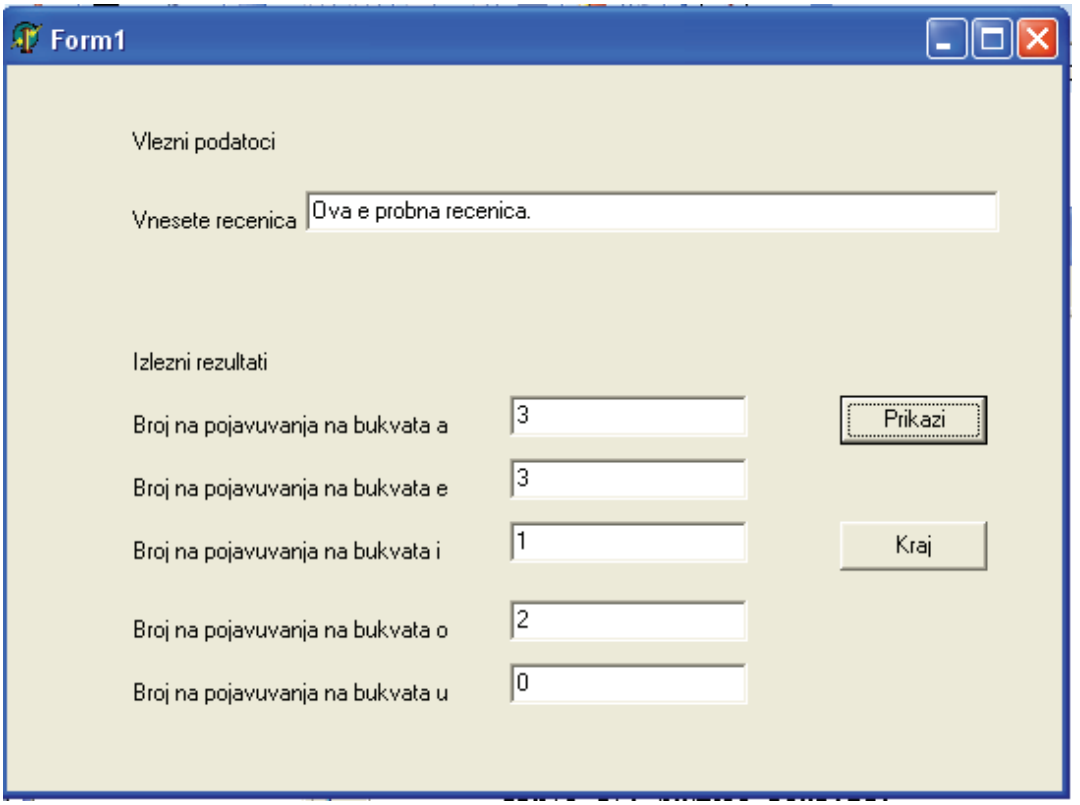
19.

а)	Properties	Б	настани
б)	Events	А	својства
в)	Tools	Г	палета на компоненти
г)	Component palette	В	ленти со алатки

Задачи

1.
```
Procedure TForm1.Button4Click( );
            Var br1,br2,br3,br4,br5:Integer;
             Rez:Real;
   Begin
            Br1:=strtoint(edit1.text);Br2:=strtoint(edit2.text);
            Br3:=strtoint(edit3.text);Br4:=strtoint(edit4.text);
            Br5:=strtoint(edit5.text);
            Rez:=(br1 + br2+br3+br4+br5)/5 ;
            Edit6.text:=floattostr(rez);
   End;
```
2.
```
 procedure TForm1.Button1Click(Sender: TObject);
   var a:integer;
begin
      a:=strtoint(edit1.text);
      if a=1 then form1.Color:=clblue;
      if a=2 then form1.Color:=clyellow;
      if a=3 then form1.Color:=clpink;
end;
```
3.
```
 Procedure TForm1.Button4Click( );
            Var br1,br2,rez:Integer;
   Begin
            Br1:=strtoint(edit1.text);
            Br2:=strtoint(edit2.text);
            If br1>br2  then rez:=br1 else rez:=br2;
            Edit3.text:=inttostr(rez);
```
4.
```
End;
```
5.
```
 procedure TPrimer5.satTimer(Sender: TObject);
   begin
            Vreme.text:=timetostr(time);
            Vreme.color:=claqua;
            datum.text:=datetostr(date);
            datum.color:=clyellow;
   end;
```

6.




```
 procedure TForm1.Button1Click(Sender: TObject);
Var a,e,i,o,u,j:integer;
s,t:string;
begin
   a:=0; e:=0; i:=0; o:=0; u:=0;
   s:=recenica.text;
   for j:=1 to length(s) do
     begin
       t:=copy(s,j,1);
       if (t='a') or (t='A') then a:=a+1;
       if (t='e') or (t='E') then e:=e+1;
       if (t='i') or (t='I') then i:=i+1;
       if (t='o') or (t='O') then o:=o+1;
       if (t='u') or (t='U') then u:=+1;
     end;
     str(a,s); bukvaa.text:=s;
      str(e,s); bukvae.text:=s;
       str(i,s); bukvai.text:=s;
        str(o,s); bukvao.text:=s;
         str(u,s); bukvau.text:=s;
end;
```

7.

```
procedure TForm1.Button1Click(Sender: TObject);
Var dd,md,gd,dr,mr,gr,d,m,g:integer;
Begin
   dd:=strtoint(edit1.text);md:=strtoint(edit2.text); d:=strtoint(edit3.text);
   dr:=strtoint(edit4.text);mr:=strtoint(edit5.text); gr:=strtoint(edit6.text);
```

```
            if dd<dr then
                begin
                    dd:=dd+30;
                    md:=md-1;
                end;
            d:=dd-dr;
            if md<mr then
                begin
                    md:=md+12;
                    gd:=gd-1;
                end;
            m:=md-mr;
            g:=gd-gr;
        edit7.text:=inttostr(d); edit7.text:=inttostr(m); edit7.text:=inttostr(g);
        end;
```

8. Да се направи програма која ќе работи како квиз со 6 прашања. За секое прашање треба да има по 4 понудени одговори (опции), од кои само еден е точен. На крај во втора форма да се прикажат освоените бодови и добиената оценка.
9. Во формата се поставуваат 3 labeli, button, 3 timeri

Својства: caption својствата на label-ите да се стават на три различни броја. Тајмерите да се постават на различен интервал.
Настани: button.click, timer1,2,3.timer

```
procedure TForm1.Button1Click(Sender: TObject);
begin
timer1.enabled:=false;
timer2.enabled:=false;
timer3.enabled:=false; {zapri gi site tri tajmeri}
if (label1.caption= label2.caption)and (label1.caption= label3.caption) then { ako e
caption na site  tri labeli ednakov togas poraka }
if messagebox(0,'You are our new lucky winner! Do you want to play
again?','congratulations',mb_yesno)=mryes then {ako e kliknato na  yes togas
povtorno se startuvaat tajmerite za da prodolzi igrata }
begin
form1.show;
timer1.enabled:=true;
timer2.enabled:=true;
timer3.enabled:=true;
end
else { ako e kliknato na NO se zatvora aplikacijata}
application.Terminate
else
form1.show;
timer1.enabled:=true;
timer2.enabled:=true;
timer3.enabled:=true;
{ ako ne se  site tri broja ednakvi igrata prodolzuva,te tajmerite se aktiviraat }
end;
```

```
procedure TForm1.Timer3Timer(Sender: TObject);
begin
label3.caption:=inttostr((strtoint(label3.caption)+1)mod 10); {generira slucaen broj
0-9 }
end;

procedure TForm1.Timer2Timer(Sender: TObject);
begin
label2.caption:=inttostr((strtoint(label2.caption)+1)mod 10);
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
label1.caption:=inttostr((strtoint(label1.caption)+1)mod 10);
end;
```

Предизвик : Наместо броеви вметнете слики – за тоа може да се користат bitbuttoni.

10.

```
procedure TForm1.Button1Click(Sender: TObject);
Var  x,y,x1,y1:integer;
begin
          x:=random(690);
          y:=random(480);
         x1:=random(690);
          y1:=random(480);
         Canvas.pen.color:=clred;
         canvas.MoveTo(x1,y1);
         canvas.lineto(x,y);
end;
```

Поглавје 4

1. Дијалог - прозорците во апликацијата овозможуваат различни начини на внесување на влезни податоци потребни за работа на апликацијата , а исто така и приказот на излезните податоци може да биде претставен во различни облици.
2. Да
3. Од името на проектот.
4. При внесување на податоци во апликацијата.
5. Функцијата InputBox која содржи 3 параметри. Првиот параметар одредува што се појавува во насловната лента, вториот параметар ја дава пораката што ќе се појави во дијалог - прозорецот, а третиот параметар ја претставува вредноста на податокот кој ќе биде внесен.
6. Го прикажува дијалог - прозорецот со порака на специфицирани координати на екранот.

Поглавје 6

1. а)
2. в)
3. в)
4. Глобални се променливите кои се дефинираат во главната програма.

5. При повикување на потпрограмата аргументите се нарекуваат вистински аргументи.
6. Потпрограмите претставуваат независни програмски целини кои имаат свои влезни податоци и даваат излезни резултати.
7. Аргументите во заглавието на потпрограмата се нарекуваат формални аргументи.
8. Локални променливи се декларираат во потпрограмата.
9. function Mnoz(br1, br2 : integer) : integer;

```
var Result : integer;
begin
        Result := br1 * br2;
        Mnoz := Result
end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
    Var  a,b,rezultat:Integer;
Begin
    a:=strtoint(edit1.text);
    b:=strtoint(edit2.text);
    Rezultat :=mnoz(a,b);
    edit3.text:=inttostr(rezultat);
end;
end.
```

10. function fact (n:integer):longint;

```
begin
        if n=0 then fact:=1
        else fact:=n*fact(n-1);
end;
Procedure TForm1.ScrollBar1Change(Sender:TObject );
Begin
        Edit1.text:=IntToStr(fact(ScrollBar1.position));
        Label1.Caption:=IntToStr(ScrollBar1.position)+'!=';
end;
```

11. function suman (n:integer):longint;

```
begin
        if n=1 then suman:=1
        else suman:=n+suman(n-1);
end;
Procedure TForm1.ScrollBar1Change(Sender:TObject );
Begin
        Edit1.text:=IntToStr(suman(ScrollBar1.position));
        Label1.Caption:=IntToStr(ScrollBar1.position)+'=';
end;
```

12.

```
function far (cel:integer):real;
begin
        far:=1.8*cel+32
end;

procedure TForm1.ScrollBar1Change(Sender: TObject);
begin
```

```
 Edit1.text:=FloatToStr(far(ScrollBar1.position));
Label1.Caption:=IntToStr(ScrollBar1.position)+'=';
end;
```

13.
```
function den (eur:integer):real;
 begin
        den:=eur/61.5;
end;

procedure TForm1.ScrollBar1Change(Sender: TObject);
begin
 Edit1.text:=FloatToStr(den(ScrollBar1.position));
Label1.Caption:=IntToStr(ScrollBar1.position)+'=';
end;
```

14. *ZaglavieNaPotprograma* треба да се дефинира во делот Interface.

Целосна дефиниција на потпрограмата треба да имаме во делот Implementation.

Општ облик на заглавието на функцијата и процедурата е :

Function ImeNaFunkcija(Arg; … ;Arg):TipNaRezultat;
Procedure ImeNaProcedura (Arg; … ;Arg);

15. Vreme (s,a,b,c);
16. Najmal(pole,10,min,br);

Поглавје 7

1. Локалните бази на податоци претставуваат наједноставен тип на база. Тоа е база која се наоѓа на еден компјутер. Сите промени на податоците директно се запишуваат во базата.
2. За да им овозможи пристап на базите на податоци, Delphi ја користи технологијата БДЕ. Таа е збир на услужни програми кои овозможуваат пристап до разни бази на податоци.
3. Во работата со базите на податоци се користи поимот DataSet. Тоа е множество на податоци до кои може да се дојде на основа на податоците содржани во базата. Податоците од DataSet не мора да припаѓаат на една табела.
4. За коректен пристап до базата БДЕ користи алиас (alias). Тоа е збир на параметри кои го опишуваат начинот на поврзување со базата на податоци. Алиас-от му кажува на БДЕ кој драјвер да го употреби и каде на дискот се наоѓаат датотеките со базата на податоци.
5. Апликација – БДЕ – база на податоци . БДЕ претставува врска меѓу базата на податоци и апликацијата.
6. DBEdit.Text.
7. DBNavigator .
8. Columns editor
9. Table, DataSource, DBGrid
10. DBEdit компонентата.

Литература

Миодраг Стојановиќ: *Рачунарство и информатика*
Ласло Краус: *Програмско окружење Delphi 7*
Ѓорѓи Јованчевски: *Информатичка технологија (за прва година средно образование)*
Драган Андонов , Цане Котевски: *Основи на софтвер*
Marco Cantu: *Mastering Delphi 4*
Niklaus Wirth: *Pascal Prirucnik*
Адреси од интернет:
http://vojo.milanovic.org/
http://www.hkbu.edu.hk/~bba_ism/ISM2110/index.htm (*Pascal Programming*)
http://www.tutorijali.org/delphi.php
Marko Švaljek : *Borland Delphi (учебник на интернет)*