Кайыпбердиев Н.А.

Кожобеков К.Г.

Мадазимов Р.М.

DELPHI

мисалдар менен

Баштоочулар
курсу

Ош - 2006

Лит. библиотека.

Муртазалиева Гульгуль.

ОшГУ. ФМИТ. 4-кр.

в один семестр.

бучение может быть отлично трем трешкам за Осенний Весенний Семестр, что позволит студентам произвольно платку почти ежемесячно (пожалуйста, смотрите академический алендарь).

Студентам, отличающимся свое образование за целый год впереди редоставляется фиксированная скидка.

ИМЭП предоставляет разнообразные стипендии и скидки, основанные на отличной академической успеваемости студента и оказательствах его/ее затруднительного финансового положения.

Помимо указанных скидок, КИМЭП поощряет всех своих студентов к активный поиск других возможных способов получения поощрской поддержки образования в КИМЭП, включая банковские редиты, поиск внешних спонсоров и грантов.

Центр по Приему и Информации, КИМЭП
Казахстан, Алматы, 050010, пр-т Абая 4
Тел: +7 (727) 270 42 13; Факс: +7 (727) 270 42 11
admis@kimep.kz; www.kimep.kz

газа.

ча Центральную Азию, а именно: насколько успеш

и как могут в дальнейшем разви

Кыргыз Республикасынын билим берүү, илим жана жаштар саясаты министрлиги

Ош мамлекеттик университети

Кайыпбердиев Н.А., Кожобеков К.Г., Мадазимов Р.М.

Delphi мисалдар менен Баштоочулар курсу

Окуу колдонмо

Ош- 2006

УДК 004
ББК 32.973-1
К 15

Рецензент: Сопуев А., ф.-м.и.д., профессор,
ОшМУнун компьютердик технологиялар
факультетинин деканы

Кайыпбердиев Н.А., Кожобеков К.Г., Мадазимов Р.М.

К 15 Delphi мисалдар менен. Баштоочулар курсу: Окуу колдонмо /
Кайыпбердиев Н.А., Кожобеков К.Г., Мадазимов Р.М. **- Ош: "ДИП полигафия" ЖЧК, 2006. – 120 б.**

ISBN 9967-427-57-4

Окуу колдонмодо азыркы учурдагы программалоо тилдеринин арасында популярдуу тилдердин бири болгон Delphi программалоо чөйрөсү жөнүндө теориялык жана практикалык материалдар каралат. Колдонмодогу теориялык материалдарда Delphi чөйрөсүнүн компоненттеринин касиеттери, методдору жана окуялары берилип, алардын колдонулушу практикалык мисалдардын жардамында көрсөтүлгөн.

Колдонмодо атайын мисалдар бөлүгү каралган. Мында берилген мисалдар бир нече этаптарга бөлүнүп, чөйрөнүн көптөгөн компоненттерин өз ара байланыштырган, толук бүтүндүккө ээ болгон программалардан турат.

Бул колдонмо кыргыз тилинде окуган жогорку окуу жайлардын студенттери жана Delphi чөйрөсүндө программалоону өз алдынча үйрөнүүнү каалаган программистер үчүн арналат.

ОшМУнун Окумуштуулар Кеңеши тарабынан басмага сунушталды

К 2404090000-06

УДК 004
ББК 32.973-1

ISBN 9967-427-57-4

Мазмуну

Киришүү

Бүгүнкү күндө жаңы информациялык технологияларды колдонуп адам баласынын ишмердүүлүктөрүнүн бардык чөйрөлөрүн автоматташтыруу, эн актуалдуу маселелерден болгондуктан визуалдык программалоо тилдерин окуп үйрөнүүгө жогорку окуу жайларда негизги көңүл бөлүнүүдө. Бул окуу колдонмодо азыркы учурдагы программалоо тилдеринин арасында популярдуу тилдердин бири болгон Delphi программалоо чөйрөсү жөнүндө теориялык жана практикалык материалдар каралат. Колдонмодогу теориялык материалдарда Delphi чөйрөсүнүн компоненттеринин касиеттери, методдору жана окуялары берилип, алардын колдонулушу практикалык мисалдардын жардамында көрсөтүлгөн. Колдонмодо атайын мисалдар бөлүгү каралган. Мында берилген мисалдар бир нече этаптарга бөлүнүп, чөйрөнүн көптөгөн компоненттерин өз ара байланыштырган, толук бүтүндүккө ээ болгон программалардан турат.

Бүгүнкү күндө Delphi чөйрөсүндө программалоону үйрөтүүчү кыргыз тилинде жазылган окуу колдонмолор жок экендигин, ошол эле учурда жогорку окуу жайлардын студенттеринин басымдуу көпчүлүгү кыргыз тилдүү болушкандыктарын эске алып колдонмону кыргыз тилинде жазылды. Сунушталган теориялык, практикалык материалдарды Delphi 5, Delphi 6, Delphi 7, Delphi 8 жана Delphi 2005 чөйрөлөрүндө пайдалануура болот. Бул колдонмону өздөштүрүүчү колдонуучулар Паскаль тилин алдын ала билет деп эсептебиз.

Колдонмо негизинен Ош мамлекеттик университетинин компьютердик технологиялар факультетиндеги «Эсептөө техникаларын жана автоматташтырылган системаларды программалык камсыздоо» (ПОВТАС), «Маалыматтарды иштетүүнү жана башкарууну автоматташтыруу системалары» (АСОИУ), «Информациялык системалар жана технологиялар», «Экономикадагы математикалык методдор», «Полиграфия», Физика, математика жана информатика факультетиндеги «Колдонмо математика жана информатика», «Информатика», «Математика» адистиктеринин студенттерине жана Delphi чөйрөсүндө программалоону өз алдынча үйрөнүүнү каалаган программисттерге сунушталат.

Бул колдонмону жогорку окуу жайларда Delphi чөйрөсү боюнча лекциялык, практикалык жана лабораториялык сабактарды өтүү, студенттердин билимин бышыктоо жана текшерүү үчүн пайдаланса болот.

Колдонмо жөнүндөгү сын-пикирлериңиздерди төмөнкү дарек боюнча жөнөтсөңүздөр болот: 714000, Ош ш., Ленин к., 331, ОшМУ, Компьютердик технологиялар факультети, Программалоо кафедрасы.

1. Кыскача маалыматтар

1.1. Алгачкы түшүнүктөр

Бул темада Delphi чөйрөсүндө программалоо учурунда пайдаланылуучу бир нече терминдер менен таанышабыз. Сөз кылынуучу түшүнүктөр Delphi чөйрөсүнүн көптөгөн компоненттеринде бирдей болгондуктан, аларды жалпылап карап чыгабыз.

Компоненттер - жөнөкөйлөтүп айтканда визуалдаштырылган объекттер деп карооо болот. Алар компоненттердин палитрасында иконка түрүндө көрсөтүлгөн. Иконканы формага жайгаштырганда, ал объекттик класстын тибин түзөт жана объекттин бардык касиеттерине ээ болот. Негизинен, компонент менен объект деген түшүнүктөрдү бир түшүнүк катары пайдалануура болот.

Компоненттер визуалдык же визуалдык эмес болушат. Визуалдык компоненттер программа иштеп жатканда терезенин элементи катары көрсөтүлөт (кнопка, текст, меню ж.б.). Визуалдык эмес компоненттерди киргемени проектирлөө учурунда көрүүгө болот, ал эми программа иштеп жатканда көрүнбөйт.

Компоненттердин палитрасынын сүрөтү.

Касиеттер – объекттин талаалары. Талаалар кайсы бир типтеги чондуктар болушат, алардын жардамында объекттин касиеттерин өзгөртүүгө болот. Мисалы, объекттин түсү, бийиктиги, кендиги, жайгашкан координаталары ж.б.. Көпчүлүк объекттерде тег (tag) талаасы бар. Бул талаа integer тибиндеги чондук болуп, ал объектке эч кандай таасир көрсөтпөйт. Ушул себептүү бул касиетти программист өзүнүн каалосу боюнча иштетиши мүмкүн.

Касиеттердин типтери жөнөкөй, татаал же объекттик типте болушу мүмкүн.

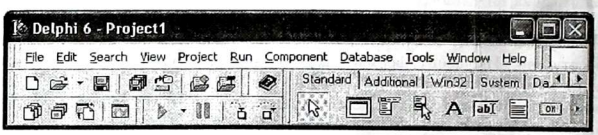
Методдор – объекттин процедуралары жана функциялары. Методдор аркылуу объекттин окуяларына аракеттерди жазууга, объекттин абалын өзгөртүүгө жана башка аракеттерди аткарууга болот.

Окуялар – объекттин үстүнөн аткарылуучу аракеттер. Мисалы, бир щелчок жасалды, ташуу аткарылды, активдешти ж.б. Окуялар объекттерде процедура түрүндө жазылат жана окуялардын аты негизинен «On» сөзүнөн башталат. Программист программанын коддорун, негизинен, объекттердин окуяларына жазат.

1.2. Терезелер

Стандартуу түрдө 5 терезе активдүү болот. Алар: чөйрөнүн негизги терезеси, объекттердин инспекторунун терезеси, форма терезеси, программалык коддун терезеси жана компоненттерди дарак түрүндө көрсөтүүчү терезелер.

Delphi чөйрөсүнүн негизги терезеси.



Негизги терезе панель сыяктуу жасалып, өлчөмүн өзгөртпөйт. Стандарттуу учурда экрандын жогору жагында жайгашкан. Бул терезеде негизги менюнун сабы, пиктограммалык кнопкалар жана компоненттердин палитрасы жайгашкан.

Компоненттердин палитрасында биринчи элемент катары курсор жайгашкан. Курсордун жардамында тандалган компоненттен баш тартууга болот. Компоненттердин группалары өзүнчө беттерге жайгаштырылган. Алар менен кийинки темаларда таанышабыз.

Пиктограммалык кнопкалар негизги менюнун эң көп колдонулуучу опцияларынын аракеттерин тез аткаруу үчүн колдонулат. Аткарган кызматтары боюнча алар бир нече группаларга бөлүнүп, өзүнчө планкаларга жайгаштырылган. Төмөнкү таблицада алардын кызматтары көрсөтүлгөн.

Standard группасы

- менюнун File| New | Other опциясынын аракетине эквиваленттүү болуп, объекттердин репозиторийсин ачат.

- менюнун File | Open File опциясынын аракетине эквиваленттүү болуп, сакталган файлды ачат.

- менюнун File | Save File (Ctrl-S клавишалары) опциясынын аракетине эквиваленттүү болуп, учурдагы программаны файлга сактайт.

- менюнун File | Save All опциясынын аракетине эквиваленттүү болуп, проектин бардык файлдарын сактайт.

- менюнун File | Open Project (Ctrl-F11 клавишалары) опциясынын аракетине эквиваленттүү болуп, мурда түзүлгөн проекттин программалык файлын ачат.

 - Project | Add to project (Shift-F11 клавишалары) аракетине эквивалентүү болуп, проекттин башка файлды кошот.

 - менюнун Project | Remove from Project опциясынын аракетине эквивалентүү болуп, проекттин курамынан файлды өчүрөт.

View группасы

 - менюнун View | units (Shift-F12 клавишалары) опциясынын аракетине эквивалентүү болуп, учурдагы проектке кирген модулдардын тизмесинен зарыл болгон модулду тандоого мүмкүндүк берет.

 - менюнун View | Forms (Ctrl-F12 клавишалары) опциясынын аракетине эквивалентүү болуп, учурдагы проектке кирген формалардын тизмесинен зарыл болгон форманы тандоого мүмкүндүк берет.

 - менюнун View | Toggle Form/Unit (F12 клавишасы) опциясынын аракетине эквивалентүү болуп, форма менен программалык коддун терезесинин өз ара алмаштырат.

 - менюнун File | New Form опциясынын аракетине эквивалентүү болуп, жаңы форманы түзөт жана аны проектке кошот.

Debug группасы

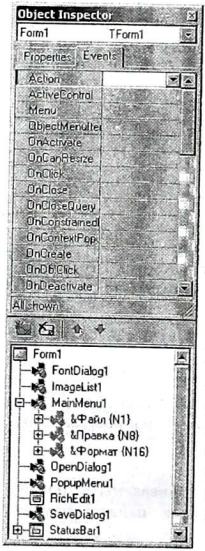
 - менюнун Run | Run (F9 клавишасы) опциясынын аракетине эквивалентүү болуп, программаны компиляциялайт жана аткарат.

 - Run | Program Pause аракетине эквивалентүү болуп, текшерилеп жаткан программанын ишине пауза жасайт.

 - Run | Trace into (F7 клавишасы) аракетине эквивалентүү болуп, пайдаланылуучу подпрограммалардын иштешин текшерүү менен программаны кадамдап аткарууну ишке ашырат.

 - Run | Step Over (F8 клавишасы) аракетине эквивалентүү болуп, программаны кадамдап аткарууну ишке ашырат, бирок пайдаланылуучу подпрограммалардын иштешин текшербейт

Объекттердин инспекторунун терезеси.



Объекттердин инспекторунун терезеси эки бөлүктөн турат. Терезенин Properties бөлүгүндө объекттин касиеттеринин жана алардын маанилеринин тизмеси берилген. Бул тизмеден объекттин касиеттерине зарыл болгон маанилер орнотулат. Ал эми Events бөлүгүндө объекттин окуяларынын тизмеси берилет. Окуяларга аракеттерди жазуу үчүн окуянын атынын он жагындагы бош тилкеге эки щелчок жасоо керек. Бул учурда программалык коддун терезеси пайда болуп, процедуранын аты параметрлери менен, андан кийин begin жана end; бош кашаалары автоматтык түрдө жазылат. Окуянын аракетинин программалык кодун ушул кашаанын ичине жазуу керек. Жөнөкөй учурда форманын терезеси программалык коддун терезесин жаап турат. Терезелердин бөркүнө щелчок жасоо менен биринен экинчисине өтүүгө болот. Өлчөмү чон формалар менен иштегенде форманын терезеси объекттердин инспекторунун терезесин жаап калышы күтүлөт, бул учурда объекттердин инспекторунун терезесине F11 клавишасын басуу аркылуу өтүүгө болот.

Формага тийиштүү компоненттерди жайгаштырып, тиркеменин интерфейсин түзөбүз. Биз түзгөн тиркеменин интерфейси аркылуу колдонуучу программанын иштөөсүн башкарат. Ушул себептүү, формага компоненттерди колдонуучунун иштөөсүнө ыңгайлуу кылып жайгаштыруу зарыл.

Компоненттерди дарак түрүндө көрсөтүүчү терезе.

Бул терезе объекттердин инспекторунун жогору жагында жайгашкан. Көп беттүү интерфейстерди түзгөндө компоненттерди тез тандоого мүмкүндүк берет. Зарыл болгон элементти тизмеден тандаганда, анын касиеттери жана окуялары объекттердин инспекторунун терезесинде пайда болот. Ошондой эле жаңы элементтерди кошууну, ашыкча элементтерди өчүрүүнү жана элементтердин жайгашуу тартибин алмаштырууну анын жогору жагында жайгашкан кнопкаларынын жардамында ишке ашырууга болот.

1.3. Delphi чойросүнүн программалык коду

Delphi чойрөсүндө негизги программанын коду проект (Project) деп аталат. Проект программист түзгөн модулдарды жана формаларды бириктирет жана алардын иштешин жөнгө салып турат. Delphi тарабынан проекттин кодуна, программисттин жасаган аракеттерине жараша, автоматтык түрдө өзгөртүүлөр кийирилип турат. Меню сабынын Project/View source опциясынын жардамында проекттин кодун ачууга болот.

```
program Project1;

uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1};

{$R *.res}

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

Проекттин коду бардык учурларда дээрлик бирдей болот жана ага кошумча коддорду жазуунун азырынча зарылчылыгы жок.

Delphi чөйрөсүндө программалоодо программист, негизинен, жаңы модулдарды жазат. Модулдун структурасы Паскаль тилиндеги модулдун структурасы менен дал келет. Модулдун структурасы менен кыскача таанышып чыгабыз.

```
unit Unit1; // Модулдун бөркү жана аты.

interface  // Ачык бөлүк.

uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs;      // Ачык бөлүктүн модулдарын жарыялоо.

type
  TForm1 = class(TForm)  // Негизги форманын тиби.
    Button1: TButton;       // Формадагы компоненттердин аты
    Edit1: TEdit;           // жана алардын типтери.
    procedure FormClick(Sender: TObject);       // Окуялардын процедура-
    procedure Button1Click(Sender: TObject);   // ларынын тизмеси.
```

```
  private          // Өздүк бөлүк, бөлүктө жарыяланган функция жана
 { Private declarations }   // процедураларды модулдун өзүндө гана
                            //иштетүүгө болот.
public     // Жалпы бөлүк, бөлүктө башка модулдарда пайдалануугa
 { Public declarations }    // мүмкүн болгон функция жана
end;                        // процедуралар жарыяланат.

var                         // Глобалдык чоңдуктарды жарыялоо бөлүгү.
 Form1: TForm1;

Implementation              // Жабык бөлүк.

{$R *.dfm}     // Ресурстук файлдарды жүктөөнүн директивасы
 // Модулдун процедура жана фунцияларынын коддорун жазуу
procedure TForm1.FormClick(Sender: TObject);
begin

end;

procedure TForm1.Button1Click(Sender: TObject);
begin

end;

end.
```

Модулдун структурасы Delphi тарабынан түзүлөт. Мындан кийин курсив шрифт менен Delphi тарабынан түзүлгөн коддор көрсөтүлөт, өзүбүз жазуучу коддор кара курсив менен берилет.

2. Компоненттер менен таанышуу

2.1. Форма

Delphi чөйрөсүндө программалоо башка визуалдык чөйрөлөр сыяктуу эле, көпчүлүк учурларда форманын жардамында жүргүзүлөт. Форма – тиркеменин терезеси, ал TForm объекттик класстын тибинде болгондуктан, объект болуп эсептелинет. Форма менен терезени бир түшүнүк катары кароого болот, айырмасы программалоо учурунда форма, программа иштеп жатканда терезе деп аталат. Бирок, форма терезенин аракеттерин толук аткара албайт. Delphi чөйрөсүндө программалоо, негизинен, формада жүргүзүлгөндүктөн, алгач форманын касиеттери менен таанышабыз (материалдын көлөмүн кыскартуу максатында башка компоненттерде да ушундай эле касиеттер, окуялар жана методдор болсо, аларга токтолбойбуз).

Hint – көрсөтүлүүчү жардамчы текстти (всплывающая подсказка) мазмунун өзгөртүү үчүн колдонулат, тиби string.

ShowHint - жардамчы текти көрсөтүү касиети, тиби boolean. True – болгон учурда жардамчы текст көрсөтүлөт, антпесе көрөтүлбөйт.

Форманын Hint касиетине жардамчы текстти жазуу үчүн, формага щелчок жасайбыз, объекттердин инспекторунун терезесинен Hint талаасын тандайбыз. Оң жагындагы тилкеге текстти жазабыз. Мисалы, «Бул негизги форма» деген сөздү жазалы. Программаны жүктөп, чычкандын көрсөткүчүн терезенин бетине жайгаштырабыз. Эч кандай кабар пайда болгон жок. Терезени жабабыз да, объекттердин инспекторунун терезесине өтүп, ShowHint талаасынын маанисин true деп өзгөртөбүз, кайра программаны аткарабыз. Чычкандын көрсөткүчүн терезенин бетине жайгаштырганыбызда, анын жанында биз жазган текст кыска убакытка көрсөтүлөт.

Constraints касиети өлчөмдү чектөө үчүн колдонулат, башкача айтканда, терезенин максималдык жана минималдык өлчөмүн аныктайт. Бул касиет interger тибиндеги төрт талаадан турат жана аларды көрүү үчүн Constraints сабынын оң жагындагы «+» белгисине щелчок жасоо керек.

MaxHeight - максималдык бийиктиги.

MaxWidth - максималдык кеңдиги.

MinHeight - минималдык бийиктиги.

MinWidth - минималдык кеңдиги.

Мисалы, көрсөтүлгөн төрт талаага 200 маанисин жазабыз. Программаны жүктөп, терезенин өлчөмүн чоңойтууга же кичирейтүүгө аракет жасайлы. Терезенин өлчөмү өзгөрбөйт, анткени терезенин өлчөмүнүн максималдык жана минималдык маанилери бири-бирине барабар.

Height - форманын бийиктигин аныктайт. Integer тибиндеги талаа.

Width - форманын кеңдигин аныктайт. Integer тибиндеги талаа.

Constraints касиетинин маанилери орнотулган учурда, бул талаалардын маанилерин орнотуунун зарылчылыгы жок.

Ctl3D – терезени үч өлчөмдүү сыяктуу көрсөтүүчү касиет. Логикалык типтеги талаа, мааниси true болгон учурда терезенин чек араларында көлөкөлөр пайда болот.

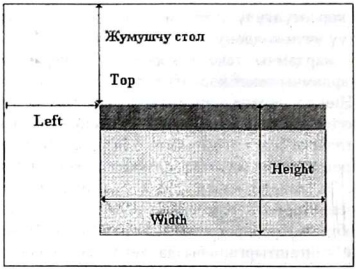
Caption – терезенин бөркүнүн тексти,тиби string.

Форманын Caption талаасына «Негизги форма» сөзүн жазып, программаны жүктөйбүз. Терезенин бөркүндө биз жазган текст пайда болот.

Enabled – логикалык типтеги талаа, мааниси false болгон учурда элемент аракеттерге жооп бербейт жана активдүү эмес абалдын түсү менен көрсөтүлөт.

Top – форманын жогорку сол бурчтун Y координаты.

Left – форманын жогорку сол бурчтун X координаты. Бул координаттар жумушчу столго салыштырмалуу алынат.



Font – терезенин шрифтинин касиеттерин орнотот. Font талаасы TFont - объекттик типтеги талаа. Бул талааны тандаганда оң жагында «+» бегиси, ал эми анын тикесинин сол жагында кнопка пайда болот. Азырынча кнопканы пайдаланабыз. Кнопканы басканда шрифтин диалогдук терезеси пайда болот. Бул диалогдук терезенин жардамында шрифтин касиеттерин орнотууга болот, бирок биз азыр бул касиетти өзгөртпөйбүз.

Icon - терезенин бөркүнүн сол жагында жайгашкан системалык менюнун пиктограммасын орнотот.

Форманын Icon касиетин тандайбыз. Оң жагындагы кнопканы басабыз. Пайда болгон терезенин Load кнопкасын басабыз. Бул учурда файлды ачуунун диалогдук терезеси пайда болот. C:/Program Files/Common Files/Borland Shared/Images/Icons папкасына өтүп, сүрөттү тандайбыз. Open жана OK кнопкаларын кезеги менен басып, терезелерди жабабыз. Эми форманын сол жогорку бурчундагы иконка биз тандаган сүрөт менен алмашып калды.

Cursor - терезенин курсорун орнотот. Талаанын тилкесинин сол жагындагы кнопканы басканда, курсорлордун константалары менен сүрөттөрүнүн тизмеси пайда болот. Тизмеден керектүү курсорду тандап коюу жетиштүү.

Color- форманын түсүн орнотот. Түстү орнотуу үчүн сол жактагы кнопкасын басып, пайда болгон тизмеден түстү тандап коюу зарыл.

Name - Объекттин аты. Delphi объекттердин аттарын компоненттин атына сан кошуу менен түзөт. Мисалы: Form1, Form2, … же Edit1, Edit2 … ж.б.у.с. Компоненттер аз болгон учурда бул эч кандай деле мааниге ээ эмес. Формалардын же андагы компоненттердин саны көп болгон учурда, мындай аттагы объекттер менен иштөө бир кыйла ыңгайсыз болуп калат. Ушул себептүү объекттердин аттарын алардын кызматтары менен логикалык байланыштагы атка өзгөртүү максатка ылайык. Мисалы, киркеме эки форманы пайдаланбасын дейли. Биринчи форма негизги болуп, экинчиси параметрлерди орнотуучу, жардамчы форма болсун. Form1 деген аттын ордуна First же Main, Form2 дегендин ордуна ParamForm же SetForm деген аттарды пайдалануу бир

кыйла ынгайлуу. Ал үчүн Name касиетине объекттин жаңы атын жазабыз жана программада ушул ат менен объектке кайрылабыз.

Жогоруда айтылган касиеттер башка компоненттерде да кезигишет жана ушундай эле кызматтарды аткарышат. Кийинки касиеттер формага гана тийиштүү касиеттер.

BorderStyle – терезенин чек арасынын стили. Ал bsSizeable – жөнөкөй, bsSingle – ичке , bsDialog – диалог терезе сыяктуу же bsNone – жок болушу мүмкүн. Жөнөкөй болгон учурда форманын өлчөмүн чычкандын жардамында өзгөртүүгө болот. Ичке болгон учурда форманын өлчөмүн өзгөртүүгө болбойт. Жок болгон учурда ичке сыяктуу эле болуп, форманын бөркү пайда болбойт.

BorderIcon – терезенин бөркүндөгү кнопкаларды орнотуучу касиет. Төмөнкү логикалык талаалардын маанилери true болгон учурда, biSystemMenu – жабуу жана системалык меню кнопкалары, biMinimize – түрүү кнопкасы, biMaximize – жаюу кнопкасы, biHelp – жардамчы информациянын кнопкасы пайда болот.

FormStyle – терезенин стили. fsNormal – жөнөкөй форма, fsMDIChild – «бала» форма, Word редакторунун документ терезеси сыяктуу, fsMDIForm – «башкы форма», Word редакторунун негизги терезеси сыяктуу, fsStayOnTop – дайыма бардык терезелердин үстүндө туруучу форма, «Пуск» кнопкасы жайгашкан панель сыяктуу.

Position - тиркемени жүктөгөндө терезенин жайгашуу ордун аныктайт. poDefault – аныкталган учур, мында форма left, top, hieght, width талааларынын маанилери боюнча жайгашат. poDefaultPosOnly – позициясы (left, top талааларынын маанилери) гана эске алынат. poDefaultSizeOnly – өлчөмү (hieght, width талааларынын маанилери) гана эске алынат. poDesigned – проектирлөө учурундагы абалы боюнча жайгашат. poDesktopCenter – Windows ОСнун жумушчу столунун ортосуна жайгашат. poMainFormCenter – негизги терезенин ортосуна жайгашат (баш ийүүчү терезелер үчүн). poOwnerFormCenter - башкы терезенин ортосуна жайгашат (fsMDIChild терезелери үчүн). poScreenCenter – экрандын ортосуна жайгашат (жумушчу стол менен экран эки башка түшүнүк).

Position касиетин өз алдыңарча өзгөртүп көргүлө.

WindowState – тиркемени жүктөгөндө анын терезесинин абалын аныктайт. wsNormal – жөнөкөй абал, проектирлөө учурундагыдай терезе пайда болот. wsMaxsimized – терезе жайылган абалда пайда болот. wsMinimized – терезе түрүлгөн абалда пайда болот.

Жогоруда биз форманын бир нече касиеттерин карап чыктык жана форманын мындан башка касиеттери да бар. Delphi чөйрөсүн жаңыдан үйрөнүп жаткандар үчүн ал касиеттер анча чоң мааниге ээ эмес. Эми терезенин окуяларын карап чыгалы.

Тиркемелерде болуп өтүүчү окуяларды бир нече топко бөлүшөт. Аларды чычкандын, клавиатуранын, тармактык түзүлүштөрдүн, принтердин, объекттин аракеттеринин ж.б. окуялары деп айтууга болот. Учурда биз чычкандын, клавиатуранын жана объекттин аракеттеринин окуяларына токтолобуз. Бул окуялардын мисалдарында азырынча карала элек объекттерди, функцияларды

пайдалануура туура келет. Мындай объекттерге жана функцияларга азырынча толук түшүндүрмө берилбейт. Аларды өзүнчө темаларда карайбыз.

Чычкандын окуялары.

OnMouseDown - чычкандын кнопкасы төмөн басылганда пайда болуучу окуя.

Объекттердин инспекторунун Events бөлүгүнө өтүп, OnMouseDown сабынын сол жагындагы тилкеге эки щелчок жасайбыз. Бул учурда программалык коддун терезесине өтөбүз жана төмөнкү саптар пайда болгонун көрөбүз (мындан ары бул аракетти «окуясына жазабыз» дейбиз).

```
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
begin
//*
end;
```

«Begin» жана «end;» сөздөрүнүн арасындагы бош орунга, учурда «//*» белгилери коюлган орунга, төмөнкү бир сапты жазабыз.

```
Form1.Caption:='Чычкандын кнопкасы терезенин X=' + inttostr(X) + '
Y=' +inttostr(Y) + ' чекитинде басылды';
```

Программаны жүктөп, текшерип көрөбүз. Терезеде чычкандын сол же оң кнопкасын төмөн басканда, анын бөркүндөгү текст өзгөргөнүн байкайбыз. Программисттердин тили менен айтканда, биздин терезе чычкандын окуясына жооп берип жатат. Окуяларды иштетүүнүн механизмине Windows ОС-сы жооп бергендиктен, ага токтолбостон, окуяларды гана карап чыгабыз.

Процедуранын аты TForm1.FormMouseDown түрүндө жазылып, Form1 объектисинин FormMouseDown процедурасы деп окулат. Процедуранын атындагы TForm1 объекттик тип. Бул жерде ката кетирилген эмес. Муну түшүнүү үчүн программалык коддун терезесин бир аз жогору түрсөнөр, *Form1: TForm1*; жарыялоосун көрөсүнөр. Sender: TObject параметри окуяны кайсы объект пайда кылганын аныктайт. Button: TMouseButton параметри чычкандын кайсы кнопкасы басылганын көрсөтөт. Мында Button mbLeft – чычкандын сол кнопкасы, mbRight – чычкандын оң кнопкасы же mbMiddle – чычкандын ортодогу кнопкасы маанилеринин бирин кайтарышы мүмкүн. Shift: TShiftState параметри чычкандын кнопкасы менен бирге басылган клавишанын же чычкандын кнопкасын аныктайт. Shift маани катары ssShift – Shift клавишасы, ssAlt – Alt клавишасы, ssCtrl – Ctrl клавишасы, ssLeft – чычкандын сол кнопкасы, ssRight – чычкандын оң кнопкасы, ssMiddle – чычкандын ортодогу кнопкасы же ssDouble – эки щелчок жасалды маанилеринин бирин кайтарышы күтүлөт. X жана Y Integer тибиндеги параметрлери чычкандын көрсөткүчүнүн координаталарын кайтарат.

Биз жазган программанын коду терезенин бөркүнө саптык чондуктагы туюнтманын мааниси ('Чычкандын кнопкасы терезенин X=' + inttostr(X) + '

Y=' +inttostr(Y) + ' чекитинде басылды') ыйгарылат дегенди түшүндүрөт. Мында, inttostr(X) integer тибиндеги X чоңдугунун маанисин string тибине өзгөртүүчү функция.

Программалык кодду процедуранын параметрлерин пайдалануу менен кайрадан жазып чыгабыз жана процедурабыз төмөнкү көрүнүшкө келет (биз кара курсив шрифт менен жазылган саптарды гана жазабыз).

procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
s:string [3];
begin
if Button=mbLeft then s:='сол' else s:='он';
Form1.Caption:='Чычкандын '+s+' кнопкасы терезенин X= '+inttostr (X)+' Y=' +inttostr(Y)+' чекитинде басылды';
end;

Процедуранын параметрлери менен таанышып чыккандан кийин программалык кодду түшүнүү анчалык деле татаал болбой калат.

OnMouseUp – чычкандын кнопкасы жогору көтөрүлгөндө (OnMouseDown окуясынан кийин) пайда болуучу окуя. Окуянын процедурасынын бөркү OnMouseDown окуясынын процедурасы сыяктуу эле.

procedure TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

OnMouseMove – чычкандын көрсөткүчү жылган учурда пайда болуучу окуя. Окуянын процедурасынын бөркү OnMouseDown окуясынын процедурасы сыяктуу эле болуп, айырмасы TmouseButton тибиндеги Button параметри жок.

procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);

Бул окуянын процедурасына төмөнкү сапты жазып текшерип көргүлө.

Form1.Caption:='X='+inttostr(X)+', Y='+inttostr(Y);

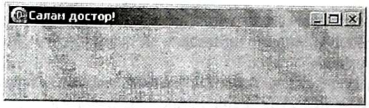
OnClick – сол кнопканын жардамында бир щелчок жасалганда (OnMouseDown окуясынан кийин жана OnMouseUp окуясынан мурда) пайда болуучу окуя. Окуянын процедурасынын бөркү өтө жөнөкөй.

procedure TForm1.FormClick (Sender: TObject);

Төмөнкү мисалды OnClick методуна жазып, текшергиле.

Form1.Caption:='Салам достор!';

Программа туура жазылган болсо, программанын жүктөп, терезе бир щелчок жасаганда, ал төмөнкүдөй көрүнүшкө келет.



Эми объекттин Name касиетине кайрылып, анын өзгөчөлүгүн карап чыгабыз. Программалык кодду текшергенден кийин, форманын Name касиетиндеги Form1 сабын First сабы менен алмаштырабыз да программаны жүктөйбүз. Программа жүктөлбөстөн, Form1.Caption деп башталган сап кызыл түскө боелуп, токтоп калат. Терезенин ылдый жагында жаңы терезе пайда болуп, «[Error] Unit1.pas(28): E2003 Undeclared identifier: 'Form1' » деген ката жөнүндөгү кабар пайда болот. Бул катанын пайда болушунун себеби, биз объекттин атын өзгөртүп койдук, ал эми программалык коддо объекттин эски аты сакталып калган. Бул катаны ондоо үчүн Form1 сөзүн First сөзү менен алмаштырабыз. Программаны кайрадан жүктөсөк, ката пайда болбойт. Ал эми процедуралардын атына көңүл бурсак, мурдагы TForm1 объекттик тиби TFirst тиби менен алмаштырылган (аны Delphi автоматтык түрдө алмаштырыт).

Демек, объекттин Name талаасындагы атты өзгөртүү, программалык коддогу объекттин эски атын да ушул атка өзгөртүүгө алып келет. Ушул себептүү программалык кодду жазуудан мурда, объекттердин Name талаасындагы атты кайра өзгөртүүгө алып келбей тургандай кылып тандоо максатка ылайыктуу.

Мындан кийин кезигүүчү мисалдарды аткарып жатканда, программалык коддо кезигүүчү компоненттердин аттарын, өзүнөр пайдаланып жаткан объекттердин аты менен алмаштырып жазууну унутпагыла.

OnDblClick – сол кнопканын жардамында эки щелчок жасалганда (бир жолу OnClick окуясынан кийин жана экинчи OnMouseDown окуясынан мурда) пайда болуучу окуя.

procedure TForm1.FormDblClick(Sender: TObject);

OnClick жана OnDblClick окуяларына тең аракет жазууга туура келсе, алардын аракеттерин бири-биринен айырмалоо үчүн TTimer компонентин пайдалануу керек. Антпесе, бул окуялар катары менен аткарыла баштап, бири экинчисинин аракеттерине жолтоо болот.

Клавиатуранын окуялары:

OnKeyDown – клавиша төмөн басылган учурда пайда болуучу окуя.

procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);

Процедуранын бөркүндөгү Word тибиндеги Key параметри басылган клавишанын кодун кайтарат. Бул окуянын аракетине төмөнкү сапты жазып текшерип көргүлө.

Form1.Caption:=inttostr(key);

OnKeyUp – клавиша жогору көтөрүлгөн учурда пайда болуучу окуя. Окуянын процедурасынын параметри OnKeyDown окуясынын процедурасындагы параметрлер менен дал келет.

procedure TForm1.FormKeyUp(Sender: TObject; var Key: Word; Shift: TShiftState);

OnKeyPress – клавишаны басылып, кое берилген учурда (OnKeyDown окуясынан кийин жана OnKeyUp окуясынан мурда) пайда болуучу окуя. Окуянын процедурасынын бөкүндөгү Key параметри жогорудагы окуялардан айырмаланып, Char тибинде. Төмөнкү мисалды ушул окуяга жазып, мурдагы мисал менен салыштыргыла.

Form1.Caption:=key;

OnCreate – терезе түзүлүп жаткан учур пайда болуучу окуя. Окуянын процедурасында Sender: TObject параметри гана бар. Терезе активдешкенге чейин аткарылуучу аракеттерди жазууда пайдаланылат. Мисалы, өзгөрүлмөлөрдүн баштапкы маанилерин орнотуу, жардамчы файлдарды ачуу ж.б.

OnDestroy – терезе жок кылынып жаткан учурда пайда болуучу окуя. OnClose окуясынан кийин пайда болот.

OnActivate – терезе активдүү абалга өтүп жаткан учурда пайда болуучу окуя.

OnDeactivate – терезе активдүү эмес абалга өтүп жаткан учурдагы окуя.

OnResize – терезенин өлчөмүн өзгөртүп жатканда пайда болуучу окуя.

OnClose – терезе жабылып жаткан учурдагы окуя.

OnCloseQuery – OnClose окуясынын алдында пайда болуучу окуя. Бул окуянын процедурасынын бөркүндөгү логикалык типтеги CanClose параметри аркылуу OnClose окуясынын аткарылышын башкарууга болот. Параметрдин мааниси false болгондо, OnClose аткарылбайт. Мисалы, Word тексттик процессорунда документте өзгөрүү болгон учурда терезени жабууга аракет кылсак, документти сактоо жөнүндөгү диалог пайда болот. «Отмена» кнопкасын баскан учурда Word редакторунун терезеси жабылбайт. Ушул аракетти ишке ашыруучу мисалды карап көрөлү.

```
procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
begin
CanClose:=true;
if MessageDlg ('Сиз терезени жабасызбы?', mtConfirmation, [mbYes,mbNo], 0) = mrNo then CanClose:=false;
end;
```

Терезни жабууга аркет жасаганыбызда, алгач OnCloseQuery методу аткарыла баштайт. Программанын башталышында CanClose чондугуна true маанисин ыйгарабыз, антпесе бир жолу диалогдун «No» («Нет») кнопкасын баскандан кийин, терезени жабуу мүмкүн болбой калат. MessageDlg функциясынын жардамында диалогду пайда кылабыз. Колдонуучу «No» («Нет») кнопкасын басса, CanClose чондугу false маанисине өзгөрөт да OnClose методу аркарылбай калат. Программадагы MessageDlg фунциясын кийинки темада карайбыз.

Терезенин көптөгөн башка методдору жана фукнциялары бар, аларга башка мисалдарда кайрылабыз.

2.2. Диалогдор

Программаларда диалогдор колдонуучуга кабар берүүнү, тигил же бул аракетти тандоону ишке ашырууда пайдаланылат. Биз стандарттуу диалогдордун үчөө менен таанышабыз.

1. **Showmessage** процедурасы эң жөнөкөй диалог болуп, анда колдонуучуга берилүүчү кабардын тексти жана «ОК» кнопкасы гана болот. Бул процедура колдонуучуга кандайдыр бир окуянын болуп өткөдүгү же болору жөнүндө кабар берет. Ал төмөнкүчө аныкталган.

procedure ShowMessage(const Msg: string);

Msg параметри колдонуучуга көрсөтүлүүчү кабар. Мисал катары терезенин OnClick окуясына ушул процедураны жазабыз.

ShowMessage('Терезеге щелчок жасалды.');

Эми программаны иштетип, терезеге щелчок жасаганда диалог пайда болот.

2. **MessageDlg** функциясы программа иштетилип жатканда, пайда болгон жагдайга жараша колдонуучудан бир нече аракеттердин бирин тандоону же колдонуучу жасаган аракетти дагы бир жолу тастыктоону талап кылуучу диалог. Функция integer тибиндеги маанини кайтарат жана анын жалпы форматы төмөнкүчө:

function MessageDlg(const Msg: string, DlgType: TMsgDlgType, Buttons: TMsgDlgButtons, HelpCtx: Integer): Integer;

Мында, Msg параметрине колдонуучуга көрсөтүлүүчү кабар жазылат. DlgType параметринде диалогдун тибинин константасы көрсөтүлөт. Бул параметрге төмөнкү константалардын бири жазылышы мүмкүн.

 - mtWarning (эскертүү),

 - mtError (катаа),

 - mtInformation (информация),

 - mtConfirmation (суроо) жана

mtCustom (жөнөкөй) сүрөт көрсөтүлбөйт. Диалогдун сол жагында ал константага туура келүүчү сүрөттөр пайда болот. Тандалышына жараша тиешелеш түрдө диалогдун бөркүндө Warning, Error, Information же Confirm сөзү жазылат. Buttons параметри диалогдо пайда болуучу кнопкаларды аныктайт. Ал массив түрүндө жазылып, кнопкалардын константалары бири-биринен үтүр менен ажыратылат. Алар:

Мааниси	Кайтаруучу мааниси
mbOk	mrOk
mbCancel	mrCancel
mbYes	mrYes
mbNo	mrNo
mbAbort	mrAbort
mbRetry	mrRetry
mbIgnore	mrIgnore
mbAll	mrAll
mbNoToAll	mrNoToAll
mbYesToAll	mrYesToAll

HelpCtx параметрине жардам (Help) файлынын ушул диалогго туура келүүчү контекстинин номери көрсөтүлөт. Help файл колдонулбаган убакта параметрдин мааниси нөлгө барабар.

Терезенин OnCloseQuery окуясы менен тааныштып жатканда ушул функцияны колдонгонбуз.

```
CanClose:=true;
if MessageDlg ('Сиз терезени жабасызбы?', mtConfirmation,
[mbYes,mbNo], 0) = mrNo then CanClose:=false;
```

Программаны жүктөгөндөн кийин, терезени жабууга аракет жасасак, сүрөттөгү диалогдук блок пайда болот.

3. **MessageBox** функциясы MessageDlg функциясы сыяктуу integer тибиндеги маанини кайтарат. Айырмасы, диалогдун бөркү программист тарабынан жазылат. Бул функция Application жана Windows модулдарында кезигет жана айтылбаган учурда Windows модулундагы функция жүктөлөт.

Алардын кайсы бири пайдаланылып жаткандыгын компиляторго көрсөтүү үчүн, биринчи учурда Application модулунун атын көрсөтүү менен жазуу зарыл.

Application.MessageBox функциясынын жалпы форматы:

function MessageBox(const Text: string, const Caption: string, Flags: Integer): Integer;.

Мында, Text параметри диалогдо пайда болуучу кабар. Caption диалогдун бөркүндөгү текст. Flags параметринде диалогдо пайда болуучу кнопкалардын константасы жазылат жана ал көрсөтүлгөн маанилердин биринде боло алат.

Мааниси	Түшүндүрмөсү
mb_AbortRetryIgnore	Abort, Retry жана Ignore кнопкалары бар диалог.
mb_Ok	OK кнопкасы бар диалог.
mb_OkCancel	OK жана Cancel кнопкалары бар диалог.
mb_RetryCancel	Retry жана Cancel кнопкалары бар диалог.
mb_YesNo	Yes жана No кнопкалары бар диалог.
mb_YesNoCancel	Yes, No, жана Cancel кнопкалары бар диалог.

Кнопкалардын атына диалогдун сол жагында пайда болуучу сүрөттүн константасын «+» белгисинин жардамында кошуп жазууга болот. Пиктограммалык сүрөттөрдүн көрүнүшү жана константалары:


- mb_IconExclamation,


- mb_IconWarning, mb_IconStop, mb_IconError же mb_IconHand,


- mb_IconInformation же mb_IconAsterisk,


- mb_IconqQuestion.

MessageBox функциясы колдонуучу тандаган кнопканын номерин кайтарат. Кнопкалардын номерлеринин константалары жана сандык маанилери:

Мааниси	Сан мааниси	Түшүндүрмөсү
idOk	1	Колдонуучу OK кнопкасын тандады.
idCancel	2	Колдонуучу Cancel кнопкасын тандады.
idAbort	3	Колдонуучу Abort кнопкасын тандады.
idRetry	4	Колдонуучу Retry кнопкасын тандады.
idIgnore	5	Колдонуучу Ignore кнопкасын тандады.
idYes	6	Колдонуучу Yes кнопкасын тандады.
idNo	7	Колдонуучу No кнопкасын тандады.

Мисал катары, терезенин OnCloseQuery окуясын төмөнкүчө ондоп жазып, диалогдун көрүнүшүн башка диалогдор менен салыштырабыз.

```
CanClose:=true;
if Application.MessageBox('Документти сактайсызбы?', 'Колдонуучуга
кабар', mb_YesNoCancel + mb_IconQuestion) = idCancel then
CanClose:=false;
```

Терезени жабуу учурунда сүрөттөгү диалог пайда болот.

Ал эми модулдун аты көрсөтүлбөй жазылганда Windows модулундагы MessageBox функциясы жүктөлөт жана анын жалпы форматы:

function MessageBox(const hWnd:HWND, const Text: string, const Caption: string, Flags: Integer): Integer;

hWnd параметрине диалогго ээлик кылган терезенин номери жазылат. Анын **Handle** мааниси учурдагы терезени түшүндүрөт, 0 болгон учурда анын ээси жок деп эсептелинет. Калган параметрлери Application.MessageBox функциясындагыдай эле аныкталган. Мисал катары жогорудагы диалогду алмаштырып жазабыз.

```
CanClose:=true;
If MessageBox(Handle, 'Документти сактайсызбы?', 'Колдонуучуга
кабар', mb_YesNoCancel + mb_IconQuestion) = idCancel then
CanClose:=false;
```

Каралып өткөн диалогдорду пайдалануу программист шартка жараша тандап алат.

2.3. Standard барагынын компоненттери

Standard бетинде бардык Windows ОС-сында бирдей болгон стандарттуу элементтердин компоненттери жайгашкан. Алардын бир нечеси менен таанышабыз.

1. TMainMenu – терезеге негизги менюну жайгаштыруучу компонент. Бул көрүнбөөчү компонент, б.а. проектирлөө учурунда аны формага

жайгаштырганда, ал пиктограмма түрүндө жайгашат. Форманын Menu касиетине негизги менюнун атын тизмеден тандап коюу керек. Менюнун опцияларын түзгөндөн кийин, форманын жогору жагында меню сабы пайда болот. Менюнун **Images** касиетине **TImageList** тибиндеги объектти көрсөтүү керек, TImageList компоненти менен кийинки темаларда таанышабыз.

Негизги менюну түзүү үчүн пиктограмманын үстүнө оң щелчок жасайбыз, пайда болгон контекстик менюдан Menu Designer опциясын тандайбыз. Менюнун дизайнеринин терезеси ачылат. Айтылган терезенин жардамында менюнун опцияларын түзөбүз.

Items касиети TMainMenu объектисинин негизги кесиети болуп эсептелинет жана ал менюнун опцияларын кармап турат. Ошол себептүү менюнун дизайнеринин терезесин объекттердин инспекторунун Items пункту аркылуу да ачууга болот. Менюнун опциялары TMenuItems объекттик тибинен турат.

Проектирлөө мезгилинде түзүлүп жаткан менюнун опцияларына оң щелчок жасаганда контекстик меню ачылат, анда Insert, Delete жана Create Submenu опциялары бар. Insert жаңы опцияны кошот, Delete опцияны өчүрөт жана Create Submenu опциясы подменюнун опцияларын түзөт. Подменюнун опциялары да TMenuItems объекттик тибинен турат.

TMenuItems тибинин төмөнкүдөй касиеттери бар.

Caption – менюнун опциясынын бөркү. Текстин арасында & белгиси коюлса, белгиден кийинки тамганын асты чийилип калат жана «Alt+тамга» түрүндөгү комбинация түзүлөт. Мисалы, &Файл дап жазсак, Файл опциясы түзүлөт. Бир жолу «-» белгисин жазсак, бөлүп туруучу сызыкча пайда болот.

Checked – логикалык типтеги талаа, мааниси true болгон учурда, менюнун опциясы тандалган болуп, «✓» түрүндөгү белги коюлат, антпесе тандалган эмес.

AutoCheck – логикалык типтеги талаа, мааниси true болгон учурда тандоонун абалын автоматтык түрдө алмаштырууну ишке ашырат, антпесе бул аракетти программист программалык жол менен камсыздоосу зарыл.

Мисалда «Инструменты» опциясы подменю опцияларынан турат. Подменюнун эки опциясынын тең AutoCheck талааларына жана «Форматирование» опциясынын Checked талаасына true мааниси орнотулган. Программа иштеп жаткан учурда, опцияларды тандаганда алардын абалдары

кезеги менен тандалган же тандалбаган абалга өтүп турат. Опция учурда кайсы абалда тургандыгын билүү үчүн AutoCheck касиетин текшерүү керек.

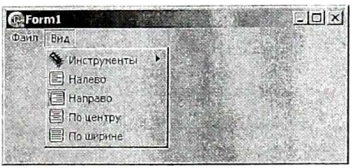
ImageIndex – integer тибиндеги талаа, менюнун опциясынын сол жагында жайгашуучу пиктограммалык сүрөттүн номери көрсөтүлөт. Пиктограммалык сүрөт сүрөттөрдүн коллекциясы – TImageList компонентинен алынат.

Bitmap – ImageIndex сыяктуу эле, айырмасы сүрөт файлдан жүктөлөт. Төмөнкү мисалда менюнун опцияларынын ушул касиетине C:/Program Files/Common Files/Borland Shared/Images/Icons папкасындагы сүрөттөрдөн тандалып коюлган.

GroupIndex – integer тибиндеги талаа, менюнун опцияларын бир группага кирүүнү камсыздайт. Бул касиетине бирдей сан жазылган опциялар бир группага тийиштүү деп эсептелинет.

RadioItem – логикалык типтеги талаа, мааниси true болгон учурда, менюнун бир группага кирген опциялары радио кнопкалар сыяктуу тандалат.

Төмөнкү жаңы опциялардын GroupIndex талааларына 1, RadioItem талааларына true маанилери орнотулган.



Мындай опциялардын окуяларына аракеттерди алардын Checked касиетинин маанисин текшерүү аркылуу жазуу зарыл.

ShortCut – тез клавишаларды аныктайт. Клавишалардын комбинациясы тизмеден тандалат.

Name - объекттин аты, N1, N2, ... түрүндө белгиленет.

2. **TPopupMenu** – контексттик менюнун компоненти жана анын касиеттери TMainMenu компонентинин касиеттерине окшош. Визуалдык компонентке тийиштүү контексттик менюну ал компоненттин PopupMenu касиетине орнотулат. Бир эле тикемеде бир нече контексттик менюнун компоненттерин пайдалануура болот.

3. **TLabel** – тексттик сап. Тексти колдонуучу өзгөртө албайт. Программалык жол менен өзгөртүүгө болот. Программада колдонуучуга түшүндүрмө жазуу үчүн пайдаланылат. Өзгөчөлөнгөн Align, Transparent жана WordWrap касиеттерин карайбыз.

Align – компоненттин формага салыштырмалуу тегизделишин аныктайт. Мааниси тизмеден тандалат жана тандалышына жараша: alNone – тегиздөө жүргүзүлбөйт, alBottom – форманын төмөн жагына, alTop – форманын жогору жагына, alClient – форманын клиенттик областына, alLeft – форманын сол жагына жана alRight – форманын оң жагына салыштырмалуу тегизделет.

Transparent – текстик саптын фонунун көрүнүшүн башкарат. Логикалык типтеги талаа. Анын мааниси true болгон учурда, фондун түсү тунук болот, антпесе Color касиетинде көрсөтүлгөн түс менен боелот.

WordWrap – сөздү жаңы сапка өткөрүүгө уруксат берет. Логикалык типтеги талаа. Мааниси true болгон учурда саптын узундугу элементтин узундугуна батпай калса, сөздү жаңы сапка алып өтөт. AutoSize касиетинин мааниси true болгон учурда, бул аракет ишке ашпайт.

4. **TEdit** - бир саптуу кийирүү компоненти. Компонентти формага жайгаштырганда, кийирүү үчүн тилке пайда болот. Тилкенин касиеттери:

BevelEdges – тилкенин чек арасынын ички тарабынын рельефти. Логикалык типтеги төрт талаадан турат. Ал талаалардын маанилери true болгондо рельефт пайда болот. beLeft, beRight, beTop жана beBottom тиешелеш түрдө сол, оң, жогору жана төмөн жактарын түшүндүрөт.

BevelInner жана **BevelOuter** – тиешелеш түрдө тилкенин чек арасынын ички жана сырткы рельефтинин көрүнүшүн аныктайт. Талаалардын маанилери тизмеден тандалат. Алар bvLowered –төмөн басылган, bvRaised – жогору көтөрүлгөн, bvSpace – тегиз же bvNone – орнотулган эмес маанилеринде боло алышат.

BevelKind – рельефтин түрүн орнотуучу талаа. Маанилери тизмеден тандалып, bkFlat – жалпак, bkSoft – жумшак, bkTile – куйруктуу же bkNone – орнотулган эмес боло алат.

BevelWidth – чек аранын жоондугун орнотот, бүтүн сан.

BorderStyle – чек аранын стилин аныктайт. Стилдин маанилери bsNone – орнотулган эмес же bsSingle – орнотулган боло алат.

CharCase – тексттин тамгаларынын касиети. Маанилерине жараша текст ecNormal – жөнөкөй, ecLowerCase – кичине тамгалар же ecUpperCase – баш тамгалар менен жазылат.

MaxLength – кийирилүүчү тексттин узундугун аныктайт, тиби бүтүн сан. Нөлгө барабар болгон учурда тексттин узундугу чектелбейт.

ReadOnly – окуу үчүн гана касиетин орнотот. Логикалык типтеги талаа, мааниси true болгон учурда колдонуучу тексти ондой албайт.

TabOrder – табуляция клавишасын басканда, фокустун жылышуу тартибин аныктайт. Тиби бүтүн сан.

TabStop – табулация клавишасын басканда элементке фокустун берилишин көрсөтөт. Логикалык типтеги талаа, мааниси true болгон учурда элементке фокус берилет.

AutoSelect - логикалык типтеги талаа, мааниси true болгон кезде табуляция калавишасынын жардамында фокус берилген элементтин тексти автоматтык түрдө тандалат (тексттин фону боелот).

Modified – текстте өзгөрүү болгондугун көрсөтүүчү талаа. Бул касиет объекттердин инспекторунда көрсөтүлбөйт, ал эми программалоодо көрүүгө болот. Логикалык типтеги талаа, текстте өзгөрүү болсо мааниси true болот.

Айтылып өткөн касиеттерди өз алдынарча өзгөртүп, текшерип көргүлө.

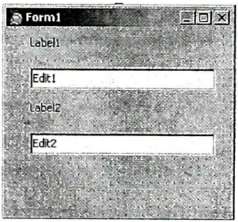
Окуялары:

OnChange – элементтин текстине өзгөртүү кийиргенде пайда болуучу окуя.

OnEnter – элементке фокус берилген учурда пайда болуучу окуя.

OnExit –элемент фокустан чыккан учурда пайда болуучу окуя.

Элементтин окуялары менен таанышуу үчүн мисал карайбыз. Формага Label1, Label2, Edit1 жана Edit2 компоненттерин сүрөттө көрсөтүлгөндөй кылып жайгаштырабыз жана алардын өлчөмдөрүн маркерлердин жардамында өзгөртөбүз.



Label1 компонентинин Caption талаасына «Edit1 компоненти фокуста.» жана Label2 компонентинин ушул эле талаасына «Edit2 компоненти фокуста эмес.» саптарын жазабыз.

Edit1 компонентинин талааларынын маанилерин төмөнкүдөй орнотобуз:

Касиети	Мааниси
TabOrder	0
AutoSelect	False
CharCase	ecUpperCase

Андан кийин, Edit2 компонентинин талааларынын маанилерин орнотобуз:

Касиети	Мааниси
TabOrder	1
AutoSelect	True
CharCase	ecNormal

Edit1 компонентинин OnChange окуясына ***HasChange:=true;*** сабын жазабыз. Ал эми программалык коддун терезесин бир аз жогору түрүп, программанын var бөлүгүнө HasChange логикалык типтеги чоңдукту аныктайбыз.

```
var
 Form1: TForm1;
 HasChange:boolean;
```

Бул чоңдуктун жардамында Edit1 жана Edit2 элементтеринин Text талааларынын маанилеринин өзгөргөндүгүн белгилейбиз. Окуяларга көрсөтүлгөн программалык коддорду жазабыз.

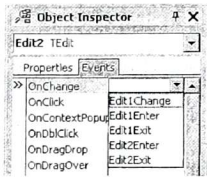
Edit1 компонентинин OnEnter окуясына:
Label1.Caption:=('Edit1 компоненти фокуста.');

Edit1 компонентинин OnExit окуясына:
Label1.Caption:=('Edit1 компоненти фокуста эмес.');

Edit2 компонентинин OnEnter окуясына:
Label2.Caption:=('Edit2 компоненти фокуста.');

Edit2 компонентинин OnExit окуясына:
Label2.Caption:=('Edit2 компоненти фокуста эмес.');

Edit2 компонентинин OnChange окуясынын оң жагындагы кнопканын жардамында тизмени ачып, Edit1.Change сабын тандап коебуз. Анткени, эки учурда тең бир эле аракетти аткарууга туура келет.



Окуялардын мындайча аныкталышын шилтеме жасоо деп аташат. Башкача айтканда Edit2 компонентинин OnChange окуясы Edit1 компонентинин OnChange окуясына шилтеме жасайт дейбиз. Окуяларга шилтеме жасоо программалык коддун кайталануучу фрагметтерин азайткандыктан программалык кодду кыскартат, бирок аны бирдей кодго ээ болгон окуяларга гана пайдалануу максатка ылайыктуу.

Терезенин OnCreate окуясына:
HasChange:=false;

Терезенин OnCloseQuery окуясына:

CanClose:=true;
If HasChange then
If MessageDlg('Тексте озгоруу болду. Сактайсызбы?', mtInformation, [mbYes,mbNoJ,0) = mrYes then CanClose:=False;

Биздин программанын толук коду төмөнкү көрүнүшкө келет:

```
unit Unit1;
interface
uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms, Dialogs, Menus, StdCtrls;
type
 TForm1 = class(TForm)
 Edit2: TEdit;
 Edit1: TEdit;
 Label1: TLabel;
 Label2: TLabel;
 procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
 procedure FormCreate(Sender: TObject);
 procedure Edit2Exit(Sender: TObject);
 procedure Edit2Enter(Sender: TObject);
 procedure Edit1Exit(Sender: TObject);
 procedure Edit1Enter(Sender: TObject);
 procedure Edit1Change(Sender: TObject);
```

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

var
  Form1: TForm1;
  HasChange:boolean;
implementation

{$R *.dfm}

procedure TForm1.Edit1Change(Sender: TObject);
begin
HasChange:=true;
end;

procedure TForm1.Edit1Enter(Sender: TObject);
begin
Label1.Caption:=('Edit1 компоненти фокуста.');
end;

procedure TForm1.Edit1Exit(Sender: TObject);
begin
Label1.Caption:=('Edit1 компоненти фокуста эмес.');
end;

procedure TForm1.Edit2Enter(Sender: TObject);
begin
Label2.Caption:=('Edit2 компоненти фокуста.');
end;

procedure TForm1.Edit2Exit(Sender: TObject);
begin
Label2.Caption:=('Edit2 компоненти фокуста эмес.');
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
HasChange:=false;
end;

procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose:
Boolean);
```

```
  begin
CanClose:=true;
If HasChange then
if MessageDlg('Тексте озгоруу болду. Сактайсызбы?', mtInformation,
[mbYes,mbNo],0) = mrYes then CanClose:=False;
end;
end.
```

Мисалдын программасынын толук көрүнүшү мурда каралган материалдар менен салыштыруу үчүн гана берилди. Көрсөтүлгөн программанын жардамында модулдун стуктурасынын көптөгөн бөлүгү Delphi чөйрөсү тарабынан автоматтык түрдө түзүлөрүн байкоого болот. Башкача айтканда программистен коюлган маселенин чечүү үчүн туура логикалык структураны түзүү жана программалык кодду туура жазуу гана талап кылынат, ал эми модулдун структурасын түзүү Delphi чөйрөсүнө жүктөлөт. Материалды кыскартуу максатында, мындан кийинки мисалдардын программасынын толук коду берилбейт.

Программаны иштөө учурундагы терезенин көрүнүшү.

Программаны жүгөп, Edit1 же Edit2 саптарын өзгөртпөстөн туруп терезени жабууга аракет кылсак, сактоо жөнүндөгү диалог пайда болбойт. Элементтердин жок дегенде бирөөнүн текстин өзгөрткөндөн кийин терезени жабууда диалогдук блок пайда болот. Терезенин мындай аракети колдонуучулар үчүн бир кыйла ыңгайлуу.

Мисалда окуяларга шилтеме жасоону түшүндүрүү үчүн текстти өзгөргөндүгүн көрсөтүүчү HasChange чоңдугун пайдаландык. HasChange чоңдугунун ордуна компоненттин Modified касиетин пайдалануруу болот. Бирок, бул учурда бардык элементтердин Modified касиетинин маанисин текшерүү керек. HasChange чоңдугу пайдаланылган саптарды өчүрүп салып, терезенин OnCloseQuery окуясына төмөнкү өзгөртүүнү киргизгенден кийин текшерип көргүлө:

```
CanClose:=true;
If (Edit1.Modified) or (Edit2.Modified) then
if MessageDlg('Тексте озгоруу болду. Сактайсызбы?', mtInformation,
[mbYes,mbNo],0) = mrYes then CanClose:=False;
```

5. **TMemo** – көп саптуу текстти кийирүү компоненти. Компонентти формага жайгаштырганда текст жазуу үчүн орун пайда болот. Формага TMemo компонентин жайгаштырып, программаны жүктөгөндөн кийин компонентке оң щелчок жасасак, контекстик меню ачылат. Контексттик менюнун опцияларынын жардамында TMemo компоненти текстти редактордун бир нече кесиеттерине ээ экенигин байкоого болот. Компоненттин көптөгөн касиеттери TEdit компонентинин касиеттеринде болуп, алардан айырмаланган төмөнкү касиеттери бар:

ScrollBar – түрүү тилкесин орнотуу. Анын мааниси ssNone -тилке пайда болбойт, ssHorizontal - горизонталдык тилке, ssVertical – вертикалдык тилке же ssBoth – эки тилке тең пайда болот дегенди түшүндүрөт.

WantReturns – жаңы сапка өтүүгө уруксат берүүчү логикалык типтеги талаа. Талаанын мааниси true болсо, Enter клавишасын пайдаланып жаңы сапка өтүүгө болот.

WantTab – текстте табуляцияны пайдалануугa уруксат берүүчү, логикалык типтеги талаа. Талаанын мааниси true болсо Tab клавишасынын жардамында текстке табуляция коюга болот.

Lines – TString объекттик типтеги касиет жана TMemo компонентинин негизги касиети болуп эсептелет. Бул касиеттин методдорун өзүнчө карайбыз.

TMemo компонентинин окуялары да TEdit компонентинин окуяларындай болгондуктан, биз анын окуяларына токтолбойбуз.

TMemo компонентинин методдору менен таанышабыз:

Clear – Lines талаасынын маанисин тазалайт.

ClearSelection – бөлүнүп алынган текстти тазалоо.

CopyToClipboard – бөлүнүп алынган текстти алмашуу буферине көчүрүү.

CutToClipboard - бөлүнүп алынган текстти алмашуу буферине кыркып алуу.

PasteFromClipboard – алмашуу буферинен текстти коюу.

SelectAll – бардык текстти тандоо.

Lines касиети TMemo компонентинин негизги касиети болуп эсептелинет жана анын методдору менен таанышабыз:

Append (S: string) – текстгин аягына жаңы жолчо кошуп, ал жолчого S параметринде көрсөтүлгөн сапты жайгаштырат.

Көпчүлүк методдор параметрлер аркылуу жазылышат. Параметрлер кашаанын ичине жазылып, көрсөтүгөн типтер менен дал келүүсү зарыл.

Insert (Index: Integer, S: string) – катары Index параметринде көрсөтүлгөн санга барабар болгон жолчого S параметриндеги сапты жайгаштырат, ал эми саптардын катары нөлдөн башталат.

SaveToFile(FileName:string) – текстти FileName параметринде көрсөтүлгөн ат менен файлга жазуу.

LoadFromFile (FileName:string) – аты FileName параметринде көрсөтүлгөн файлдан текстти жүктөө.

Delete(Index:Integer) – катар номери Index параметринде көрсөтүлгөн сапты өчүрөт.

TMemo компенентин пайдаланып чычкандын окуяларынын пайда болуу тартибин карап чыгабыз. Ал үчүн форманы толук ээлебей тургандай кылып, TMemo компонентин формага жайгаштырабыз. Memo1 компонентинин Lines талаасынын он жактагы кнопкасын басып, пайда болгон терезедеги текстти өчүрүп салагандан кийин терезени жабабыз.

Форманын көрсөтүлгөн окуяларына программалык коддорду жазабыз.

OnMouseDown окуясына:

```
If Button = mbLeft then
Memo1.Lines.Append('Сол кнопка томон басыды.')
else Memo1.Lines.Append('Он кнопка томон басыды.');
```

OnMouseUp окуясына:

```
If Button = mbLeft then
Memo1.Lines.Append('Сол кнопка жогору которулду.')
else Memo1.Lines.Append('Он кнопка жогору которулду.');
```

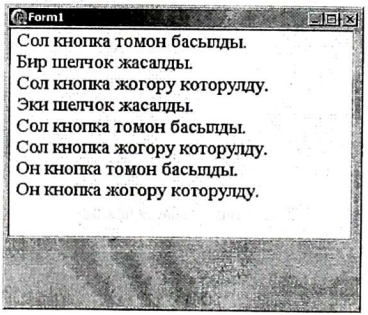
OnClick окуясына:

```
Memo1.Lines.Append('Бир щелчок жасалды.');
```

OnDblClick окуясына:

```
Memo1.Lines.Append('Эки щелчок жасалды.');
```

Программаны жүктөп, терезеге чычкандын сол кнопкасы менен эки щелчок жана оң кнопкасы менен бир щелчок жасайбыз. Программалык код туура жазылып жана эки щелчок туура аткарылса, TMemo элементтинде сүрөттөгүдөй кабарлар пайда болот. Кабарлардан байкалгандай чычкандын оң кнопкасында бир жана эки щелчок окуялары пайда болбойт.

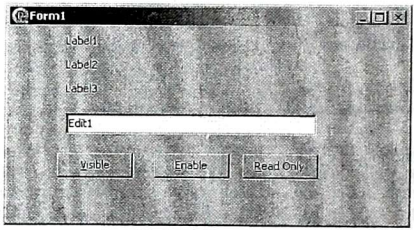


6. **TButton** – командалык кнопканын компоненти.

Caption талаасына & белгисин кошуп жазууга болот.

ModalResult талаасына кнопка басылганда, ал кайтаруучу маани көрсөтүлөт. Бул маанилерди программист өзүнүн диалогдук терезесин түзүүдө пайдалана алат. Алардын константалары жана түшүндүрмөлөрү MessageDlg функциясында берилген константалар жана түшүндүрмөлөр менен дал келет.

Командалык кнопкалардын жардамында TEdit компонентинин Visible, Enable жана ReadOnly касиеттери менен таанышабыз. Формага үч TLabel, бир TEdit, жана үч TButton компоненттерин жайгаштырабыз. үч командалык кнопканы формага тез жайгаштыруу үчүн Shift клавишасын басып, кое бербестен, командалык кнопканын пиктограммасына щелчок жасайбыз. Shift клавишасын кое берип форманын командалык кнопкалар жайгаша турган орундарына щелчок жасайбаз. Ушундай эле аракеттердин жардамында TLabel компоненттерин да жайгаштырабыз. Кнопкаларды жайгаштырып болгон соң «курсор» кнопкасын тандап коебуз. Формага компоненттерди сүрөттө көрсөтүлгөндөй кылып жайгаштырып, кнопкалардын Caption касиеттерин оңдойбуз.



Кнопкалардын окуяларына төмөнкү программалык коддорду жазып чыгабыз (программага берилген түшүндүрмөлөрдү жазуунун зарылчылыгы жок).

«Visible» кнопкасынын OnClick окуясына:

```
if Edit1.Visible then      //Visible касиетинин мааниси true болсо
begin
Edit1.Visible:=false;      //Анын маанисин false'го өзгөртөбүз
Label1.Caption:='Visible:=false';   // Кабарды чыгарабыз
end
else      //антпесе, Visible касиетинин мааниси false болсо
begin
Edit1.Visible:=true;   //Анын маанисин true'га өзгөртөбүз
Label1.Caption:='Visible:=true';   //Кабарды чыгарабыз
end;
```

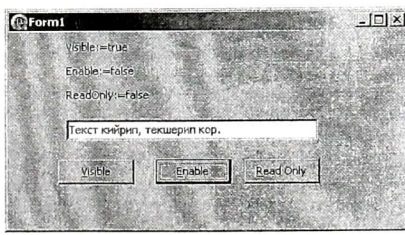
«Enable» кнопкасынын OnClick окуясына (түшүндүрмөсү жогоркудай эле):

```
if Edit1.Enabled then
begin
Edit1.Enabled:=false;
Label2.Caption:='Enable:=false';
end
else
begin
Edit1.Enabled:=true;
Label2.Caption:='Enable:=true';
end;
```

«Read Only» кнопкасынын OnClick окуясына:

```
if Edit1.ReadOnly then
begin
Edit1.ReadOnly:=false;
Label3.Caption:='ReadOnly:=false';
end
else
begin
Edit1.ReadOnly:=true;
Label3.Caption:='ReadOnly:=true';
end;
```

Программаны жүктөп, кнопкаларды кезеги менен басып, Edit1 компонентинин касиеттеринин өзгөрүүсүнө көңүл бурабыз. «Enable» жана «Read Only» кнопкаларын басып, Edit1 компонентиндеги текстти оңдого аракет жасайбыз. Ал эми касиеттер учурда кандай мааниге ээ экендигин TLabel компоненттери көрсөтүп турат. Edit1 компонентинин касиетин кайра өзгөртүү үчүн тиешелеш кнопкаларды кайра басуу керек, касиеттер маанилерин кезеги менен алмаштырышат.



7. **TCheckBox** – опцияны тандоонун кутуча компоненти. Компоненттин жардамында түзүлгөн опцияларды бири - биринен көз карандысыз тандоого болот. Кутучага щелчок жасаганда анын тандалуу абалы өзгөрөт. Кутучанын Caption касиетине жазылган текст анын оң жагына жайгашат жана бул касиетинде «&» белгисин пайдалануура болот.

Checked – кутучанын тандалган же тандалбаган абалын көрсөтөт. Логикалык типтеги талаа. Мааниси true болсо, опция тандалган болуп, кутучага желекче коюлат. Кутучанын абалына жараша маанисин өзгөртөт.

State – кутучанын абалын аныктоочу касиет. Checked касиетине окшош. Айырмасы, бул касиет кутучанын cbChecked - тандалган, cbUnchecked – тандалбаган жана cbGrayed-толук эмес тандалган абалдарын аныктай алат.

AllowGrayed – кутучанын cbGrayed абалынын пайда болуусун камсыздайт. Логикалык типтеги талаа. Мааниси true болгон учурда кутучанын тандалган абалын Checked касиети аркылуу текшерүү максатка ылайыксыз, анткени Checked касиети аркылуу кутучанын cbGrayed абалын аныктоого мүмкүн эмес. Бул учурда кутучанын абалы State касиети менен аныкталат.

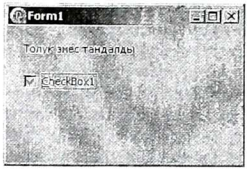
Жогорудагы касиеттердин өзгөчөлүгүн мисалдын жардамында салыштырабыз. Формага бирден TLabel жана TCheckedBox компоненттерин жайгаштырабыз. TCheckedBox компонентинин OnClick окуясына төмөнкү сапты жазабыз.

```
If CheckBox1.Checked then Label1.Caption:='Тандалды' else
Label1.Caption:= 'Тандалбады';
```

Программаны иштетип, кутучанын тандалган абалын өзгөртүп көрөбүз. Label1 компонентинде пайда болуп жаткан кабар кутучанын абалын туура көрсөтүп жатканын байкайбыз. Терезени жабабыз. CheckedBox1 компонентинин AllowGrayed талаасына true маанисин орнотобуз жана программанын иштөөсүн текшеребиз. Кутуча cbGrayed абалына өткөндө, Label1 компонентинде «Тандалбады» кабары өзгөргөн жок, башкача айтканда кутучанын абалы туура эмес аныкталды. Кутучанын абалын туура аныктоо үчүн жогорудагы программанын коддун кайрадан ондойбуз. Мурдагы сапты өчүрүп салып, анын ордуна төмөнкү саптарды жазабыз.

```
If CheckBox1.State=cbChecked then
Label1.Caption:='Тандалды' else
If CheckBox1.State=cbUnchecked then
Label1.Caption:='Тандалбады' else
Label1.Caption:='Толук эмес тандалды';
```

Программаны жүктөп, кутучанын абалын өзгөрткөнүбүздө, эми программа кутучанын үч абалын тең туура аныктап жаткандыгын байкайбыз. Программанын иштөө учурундагы көрүнүшүнүн сүрөтү берилди.

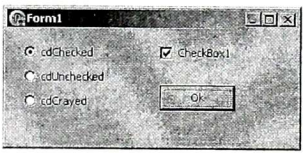


8. **TRadioButton** – опцияларды тандоонун радио кнопкасы. Радио кнопкалар бир нече опциядан бирөөнү гана тандоого уруксат берет. Радио кнопкалар группа түрүндө иштегендиктен алардын саны бирден көп болуусу зарыл. Радио кнопканын тандалганын билүү үчүн Checked касиетин пайдаланабыз. Компонент бизге мурдагы компоненттерден белгилүү болгон касиеттерге жана окуяларга ээ.

Формага үч TRadioButton жана бирден TCheckBox, TButton компоненттерин жайгаштырабыз. Радио кнопкалардын Caption талааларына тиешелеш түрдө cbChecked, cbUnchecked жана cbGrayed маанилерин жазабыз. Экинчи радио кнопканын Checked талаасына true маанисин орнотобуз. CheckBox1 компонентинин AllowGrayed касиетине true маанисин орнотобуз. Командалык кнопканын OnClick окуясына көрсөтүлгөн программалык кодду жазып, программанын иштөөсүн текшеребиз.

```
if RadioButton1.Checked then CheckBox1.State:=cbChecked;
if RadioButton2.Checked then CheckBox1.State:=cbUnchecked;
if RadioButton3.Checked then CheckBox1.State:=cbGrayed;
```

Мисалдын терезесинин көрүнүшү.



9. **TListBox** – тизме компоненти. Тандалуучу маанилер өтө көп болгон учурда аларды тизмеге киргизүү үчүн арналган. Мында ар бир маани тизмеге өзүнчө сап катары жайгашат. Колдонуучу маанини тизмеден чычкандын же клавиатуранын жардамында тандай алат, бирок жаңы маани кийире албайт.

Тизменин касиеттери:

AutoComplete – тизмеде тандала турган маанини автоматтык түрдө алмаштыруу режими. Логикалык типтеги талаа. Мааниси true болгон учурда

режим орнотулат. Колдонуучу клавиатурадан киргизип жаткан сапка дал келүүчү маанини тизмеден издейт. Дал келүүчү маани табылса, тандоо тилкесин ошол сапка жылдырат.

Columns – мамычалар, тиби бүтүн он сан. Мааниси нөл болгондо тизме бир мамычалуу, калган учурларда көп мамычалуу болот.

MultiSelect – тандоонун режимин орнотот. Логикалык типтеги талаа. Мааниси true болгон учурда колдонуучу Shift жана Ctrl клавишаларынын жардамында тизмеден бир нече сапты бир учурда тандай алат.

Sorted – тизменин маанилерин сорттоону ишке ашырат. Логикалык типтеги талаа. Мааниси true болгон учурда тизменин саптары маанисине жараша сорттолот.

Items – TString объекттик тибиндеги тизменин саптарынын объектиси.

Тизменин окуялары биз мурда каралган окуялардан турат. Биз, тизменин бир нече методдору жана касиеттери менен таанышабыз.

Count – тизмедеги саптардын санын көрсөтүүчү касиет. Тиби бүтүн сан.

DeleteSelected – тандалган сапты өчүрүүчү метод. Өчүрүлгөн саптар учурдагы сеанска гана тийиштүү. Программанын кийинки жүктөлүшүндө тизменин саптары толук боюнча көрсөтүлөт.

ItemIndex – тизмеден тандалган саптын катар номерин көрсөтүүчү касиет. Тиби бүтүн сан. Тизменин саптарынын номери нөлдөн башталат.

SelCount – тандалган саптардын санын көрсөтүүчү касиет. MultiSelect режимине true мааниси орнотулганда, колдонуучу тандаган саптардын санын билүү үчүн пайдаланылат.

Selected [X: integer]:boolean – тизмеден катар номери X болгон саптын тандалгандыгын текшерүүчү касиет. Логикалык типти кайтарат. Көрсөтүлгөн номердеги сап тандалган болсо, true маанисин кайтарат.

Items тибинин касиеттери:

Text – тизмедеги бардык саптардын маанилерин текст түрүндө кайтаруучу касиет, тиби string.

String [X:integer] – тизменин катар номери X болгон саптын маанисин кайтаруучу касиет, тиби string.

SaveToFile (FileName:string) – тизменин саптарынын маанилерин файлга жазат.

LoadFromFile (FileName:string) – тизменин саптарынын маанилерин файлдан жүктөйт.

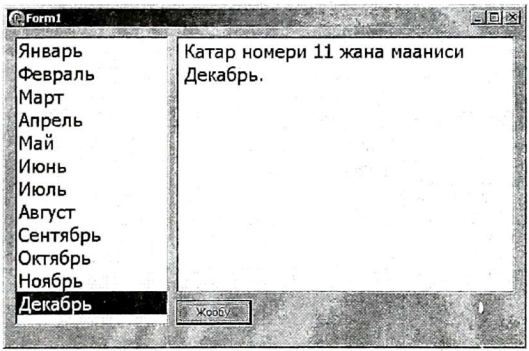
Items тибинин көпчүлүк методдору TMemo компонентиндеги Lines тибинин методдору менен дал келет.

Тизме менен иштөөчү мисалды карайбыз. Формага тизменин компонентин жайгаштырабыз. Анын Items касиетинин оң жагындагы кнопканы басып, пайда болгон терезеге айлардын атын кийиребиз. MultiSelect касиетине false маанисин орнотобуз. Формага TMemo жана TButton компоненттерин жайгаштырабыз. Button1 компонентинин Caption касиетине «Жообу» сабын жазабыз. Ал эми анын OnClick окуясына программалык кодду кийиребиз:

```
Memo1.Clear;
Memo1.Lines.Append('Катар номери '+inttostr(Listbox1.ItemIndex)+'
жана мааниси '+ListBox1.Items.Strings[Listbox1.ItemIndex]+'.');
```

Программанын биринчи сабы Memo1 компонентинин мазмунун тазалайт. Кийинки Listbox1.ItemIndex сабы тизмеден тандалган саптын катар номерин жана ListBox1.Items.Strings [Listbox1.ItemIndex] ал саптын маанисин кабардын текстине бириктирип, Memo1 компонентине жайгаштырат.

Программанын терезесинин сүрөтү төмөндө берилди.

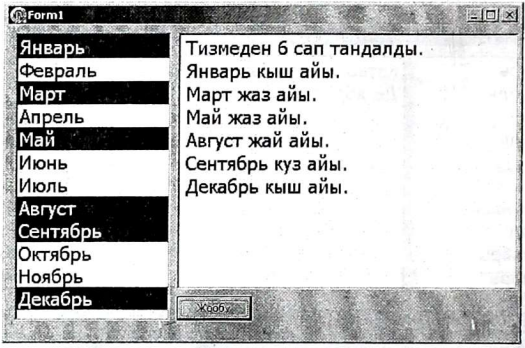


Жогорудагы программа тизмеден бир гана сапты тандоого мүмкүндүк берет. Ушул эле форманы пайдаланып MultiSelect режиминде тандоо жүргүзүүгө мисал карайбыз. Тизменин MultiSelect талаасын true маанисине өзгөртөбүз жана командалык кнопканын программалык кодун жаңыдан жазып чыгабыз.

```
procedure TForm1.Button1Click(Sender: TObject);
var
i:integer;
begin
Memo1.Clear;
Memo1.Lines.Append('Тизмеден '+inttostr(Listbox1.SelCount)+' сап
тандалды.');
For i:=0 to ListBox1.Count-1 do
begin
if ListBox1.Selected[i] then
case i of
0,1,11: Memo1.Lines.Append(Listbox1.Items.Strings[i]+' кыш айы.');
```

```
    2..4: Memo1.Lines.Append(Listbox1.Items.Strings[i]+' жаз айы.');
5..7: Memo1.Lines.Append(Listbox1.Items.Strings[i]+' жай айы.');
8..10: Memo1.Lines.Append(Listbox1.Items.Strings[i]+' куз айы.');
end;
end;
end;
```

Программанын терезесинин иштөө учурундагы көрүнүшү сүрөттө берилди.



Тизменин саптарынын саны бирден баштап эсептелинип, ал эми номери нөлдөн башталгандыктан, тизменин эң чоң номери тизменин санынан бирге аз болот. Ошол себептүү цикл нөлдөн башталып ListBox1.Count-1 де аяктайт деп жаздык. Саптын тандалган же тандалбаганын ListBox1.Selected[i] боюнча текшеребиз. Сап тандалган болсо, анын маанисин Listbox1.Items.Strings[i] аркылуу тизмеден алабыз. Мында і тизмедеги сабынын номери.

10. **TComboBox** – комбинацияланган тизменин компоненти. Тизме компонентинен кийирүү тилкеси менен айырмаланат. Колдонуучуга зарыл болгон маани тизмеде жок болсо, ал маанини колдонуучу кийирүү тилкесине жаза алат. Бул компоненттин бир нече түрү бар, бирок төмөн түшүүчү тизмеси бар түрү кеңире колдонулат. Тизме компонентинен айырмаланган касиеттерин карайбыз.

AutoCloseUp – тизмени автоматтык түрдө түрүү. Логикалык типтеги талаа, мааниси true болгон учурда колдонуучу клавиатуранын жардамында маанини тандагандан кийин тизме автоматтык түрдө жабылат.

AutoDropDown – тизмени автоматтык түрдө жаюу. Логикалык типтеги талаа, мааниси true болгон учурда колдонуучу клавиатуранын жардамында маанини кийирүү тилкесине кийире баштаганда тизме төмөн түшүп ачылат.

DropDownCount – жайылган учурдагы тизменин бийиктиги. Бүтүн оң сан, саптардын санына барабар.

ItemIndex – айтылбаган учурда, номери көрсөтүлгөн тизмедеги саптын мааниси кийирүү тилкесине орнотулат. Тиби бүтүн сан.

Text – кийирүү тилкесиндеги текст. String тибиндеги чоңдук.

Бизге белгилүү болгон окуялардан башка компоненттин жаңы эки окуясы менен таанышабыз:

OnDropDown – тизме ачылып жатканда пайда болуучу окуя.

OnSelect – тизмеден маани тандалган учурда пайда болуучу окуя.

Компоненттин Items объекттик касиетинин өзгөчө касиеттери жана методдору менен таанышабыз.

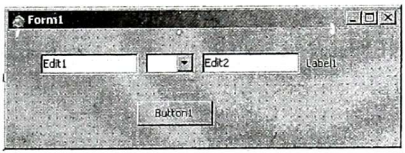
Delimiter – ажыратуучу белгини көрсөтүүчү касиет. Char тибиндеги чоңдук. DelimitedText касиети үчүн пайдаланылат.

DelimitedText – тизмеге кирген саптардын маанилери Delimiter касиетинде көрсөтүлгөн белгинин жардамында ажыратып жазылган сап. String тибиндеги касиет.

Exchange(Index1: Integer, Index2: Integer) – тизменин Index1 номерлүү сабын номери Index2 болгон сап менен орун алмаштырып жайгаштыруучу метод.

Башка методдору жана касиеттери тизме компонентинин методдору жана касиеттери менен дал келет.

Мисал катары формага эки TEdit жана бирден TCombobox, Tlabel, TButton компоненттерин жайгаштырабыз. Аларды формага сүрөттө көрсөтүлгөндөй кылып жайгаштырабыз.

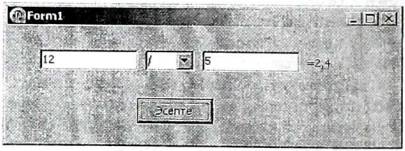


Edit1 жана Edit2 элементтеринин Text касиеттерине нөл санын жазабыз. Label1 элементинин Caption касиеттерине барабар белгисин, командалык кнопканын ушул эле касиетине «Эсепте» сөзүн жазабыз. Combobox1 элементинин Items касиетинин терезесин ачып «+, -, *,/» белгилеринин ар бирин жаңы сапка кийиребиз. Командалык кнопканын OnClick окуясына төмөнкү саптарды жазабыз.

```
if ComboBox1.Text='+' then Label1.Caption:= '='+floattostr(strtofloat
(Edit1.Text)+strtofloat(Edit2.Text));
if ComboBox1.Text='-' then Label1.Caption:= '='+floattostr(strtofloat
(Edit1.Text)-strtofloat(Edit2.Text));
```

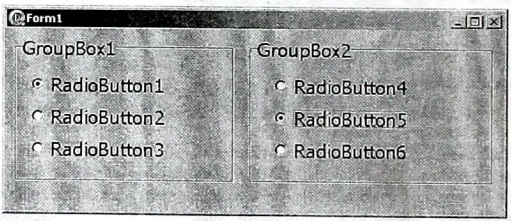
```
if ComboBox1.Text='*' then Label1.Caption:= '='+floattostr(strtofloat
(Edit1.Text)*strtofloat(Edit2.Text));
if (ComboBox1.Text='/') and (Edit2.Text<>'0') then Label1.Caption:= '='+
floattostr(strtofloat(Edit1.Text)/strtofloat(Edit2.Text));
```

Программаны жүктөп, текшерип көрөбүз. Иштөө учурундагы терезенин сүрөтү төмөндө берилди.



11. **TGroupBox** – группа компоненти. Терезенин элементтерин группаларга ажыратуу үчүн колдонулуучу компонент. Бул компонентке атайын программалык код дээрлик жазылбагандыктан, бул компоненттин касиеттерине жана окуяларына токтолбойбуз. Аны колдонууну мисалда карайбыз.

Формага эки TGroupBox компонентин жайгаштырабыз жана ар бирине үчтөн радио кнопкаларды жайгаштырабыз. Программаны жүктөп, эки башка группада жайгашкан радио кнопкалардын тандалышына көңүл бурабыз. Бир группада жайгашкан радио кнопкалар экинчи группадагы радио кнопкалардан көз карандысыз тандалат.



12. **TPanel** – панель компоненти. Терезенин бөлүгүнө окшогон элемент. Панелдин областына жайгаштырылган элементтерге ээлик кылат. Көпчүлүк касиеттери жана окуялары мурда карап чыккан касиеттер жана окуялар менен дал келет.

Формага бир панель жана эки командалык кнопканы жайгаштырабыз. Панелге бир нече башка элементтерди жайгаштырабыз. Формадагы биринчи кнопканын OnClick окуясына ***Panel1.Visible:=false;*** сабын жазабыз. Экинчи

кнопканын ушул эле окуясына, ***Panel1.Visible:=true;*** сабын жазабыз. Программаны жүктөгөндөн кийин биринчи кнопканы басканда, панель менен бирге ага жайгашкан элементтер да көрүнбөй калат.

2.4. Additional барагынын компоненттери

Additional барагында кооз интерфейсти түзүүгө арналган компоненттер жайгашкан.

1. **TBitBtn** компоненти TButton компоненти сыяктуу кнопка. Айырмасы, анын **Glyph** касиетинин жардамында кнопкага сүрөт жайгаштырууга болот. Кнопкага жайгаштырылуучу сүрөттөрдү C:/Program Files /Common Files /Borland Shared /Images /Button папкасынан алсаңар болот. Мындагы сүрөттөр эки бөлүктөн турат. Биринчи бөлүгү түстүү болуп кнопканын активдүү абалына, экинчиси түссүз болуп кнопканын активдүү эмес абалына туура келет. Сүрөт кнопканын абалына жараша өз алдынча алмашып турат.

2. **TSpeedButton** – тез кнопка компоненти. Инструменттердин панелине жайгаштырууга арналган кнопка. **Glyph** касиетиндеги сүрөтү гана көрсөтүлөт. Caption касиети бар, бирок **Glyph** касиетине сүрөт орнотулган учурда ага жазылган текст көрсөтүлбөйт. Мындан сырткары башка кнопкалардан айырмаланган төмөнкү касиеттери бар.

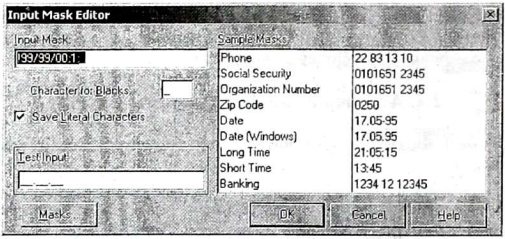
Flat – кнопканын жалпак көрүнүшүн орнотот. Логикалык типтеги талаа, true болгон учурда кнопка жалпак көрүнүштө болуп, чычкандын көрсөткүчү кнопканын үстүнө келген кезде жогору көтөрүлөт.

GroupIndex – группанын номери. Бүтүн оң сан. Мааниси нөл болсо, кнопка группага кирбейт, антпесе бирдей номерлерге ээ болгон кнопкалар бир группага тийиштүү болушуп, алар бири-биринен көз каранды түрдө иштешет.

Down – учурда тандалып турган кнопка. Логикалык типтеги талаа, true болгон учурда кнопка төмөн басылган абалда болот. Группага кирген кнопкалар үчүн гана орнотууга болот. Программалоо учурунда группага кирген кнопкалардын кайсы бири тандалып турганын билүү үчүн алардын ушул касиетин текшерүү керек.

Окуялары башка кнопкаларга салыштырмалуу бир кыйла аз:

3. **MaskEdit** – формattалган маанини кийирүү компоненти. Негизги касиеттери TEdit компонентинин касиеттери менен дал келип, **EditMask** касиети менен гана айырмаланат. Бул касиетте кийирилүүчү маанинин форматы түзүлөт. Оң жактагы кнопкасын басканда маскалардын терезеси ачылат. Атайын тизмеде зарыл болгон формattы тандоого болот. Тизмедеги маскалардан айырмаланган маскаңы кийирүү тилкесинде атайын символдордун жардамында түзүүгө болот.



Маска үтүрлүү чекит менен ажыратылган үч бөлүктөн турат. Биринчи бөлүк масканын өзү, анда иштетилүүчү символдор төмөнкүлөр:

Символ	Түшүндүрмө
!	Символ маскада колдонулса, сөзсүз эмес символдор масканын алдына коюлат, антпесе коюлбайт
>	Бул символдон кийинки символдор автоматтык түрдө жогорку регистрге которулат
<	Бул символдон кийинки символдор автоматтык түрдө төмөнкү регистрге которулат
<>	Регистлер боюнча которууну токтотот
\	Мындан кийинки символдор маскага коюлат. Шаблондун элементи катары эсептелген символдорду кошуу үчүн колдонулат
L	Бул позицияга тамганы гана кийирүүгө уруксат берилет
l	Бул позицияга тамганы гана кийирүүгө уруксат берилет же бош калтырууга болот
A	Бул позицияга тамга же цифра гана кийирүүгө уруксат берет
a	Бул позицияга тамга же цифра гана кийирүүгө уруксат берилет же бош калтырууга болот
C	Бул позицияга каалагандай символ кийирүүгө уруксат берилет
c	Бул позицияга каалагандай символ кийирүүгө уруксат берилет же бош калтырууга болот
0	Бул позицияга цифра гана кийирүүгө уруксат берилет
9	Бул позицияга цифра гана кийирүүгө уруксат берилет же бош калтырууга болот
#	Бул позицияга цифра, «+» же «-»символдорун гана кийирүүгө уруксат берилет же бош калтырууга болот
:	Бул позицияга убакытты жазуу үчүн Windows системасында орнотулган саат, минута жана секунданы ажыратуучу белги коюлат.
_	Бул позицияга пробел коюлат.

Масканын экинчи бөлүгү текстгин жыйынтыгына атайын символдордун кошулуп жазылышын аныктайт. «0» болгон учурда атыйын символдор кошулуп жазылбайт, каалагандай башка символ болсо атайын символдор кошулуп жазылат. Үчүнчү бөлүгүндөгү символ колдонуучу тарабынан бош калтырылган шаблондун орундарын толтурат (айтылбаган учурда пробел коюлат).

4. **TstringGrid** – саптардын таблицасынын компоненти. Текстгик информацияларды кийирүүнү же чыгарууну ишке ашыра турган эки өлчөмдүү таблица. Талицанын мамычаларынын жана саптарынын номерлери нөлдөн башталат.

Касиеттери:

ColCount – мамычалардын саны, бүтүн оң сан.

ColRow – саптардын саны, бүтүн оң сан.

FixedColot – бөрк түрүндө көрсөтүлүүчү ячейкалардын түсү.

FixedCol – бөрк түрүндө көрсөтүлүүчү мамычалардын саны. Бүтүн оң сан. Таблица бөрк түрүндөгү мамычага ээ болбосо, бул талаанын маанисине нөл жазуу керек. Бөрк түрүндө көрсөтүлгөн мамычанын бийиктигин колдонуучу өзгөртө албайт.

FixedRow – бөрк түрүндө көрсөтүлүүчү саптардын саны. Бүтүн оң сан. Таблица бөрк түрүндөгү сапка ээ болбосо бул талаанын маанисине нөл жазуу керек. Бөрк түрүндө көрсөтүлгөн саптын кеңдигин колдонуучу өзгөртө албайт.

GridLineWidth – таблицанын ячейкаларын бөлүп туруучу сызыктардын жоондугу, бүтүн оң сан. «0» болгон учурда сызыктар көрсөтүлбөйт.

Options – логикалык типтеги бир нече талаалардан турган касиет. Төмөндө алардын тизмеси жана мааниси true болгон учурдагы түшүндүрмөсү берилди.

Аты	Түшүндүрмөсү
goFixedVertLine	Бөркүндө вертикалдык сызыктар көрсөтүлөт
goFixedHorzLine	Бөркүндө горизонтал сызыктар көрсөтүлөт
goVertLin	Вертикалдык сызыктар көрсөтүлөт
goHorzLine	Горизонтал сызыктар көрсөтүлөт
goRowSizing	Колдонуучу саптардын бийиктигин өзгөртө алат
goColSizing	Колдонуучу мамычалардын кеңдигин өзгөртө алат
goRowMoving	Колдонуучу саптардын ордун алмаштыра алат
goColMoving	Колдонуучу мамычалардын ордун алмаштыра алат
goEditing	Колдонуучу ячейкалардын маанилерин оңдой алат
goTabs	Колдонуучу ячейкаларга Tab жана Shift+Tab клавишаларынын жардамында өтө алат

goRowSelect	Колдонуучу сапты толук тандай алат, goEditing, goTabs опциялары аракеттерин жоготот.
goAlwaysShowEditor	Колдонуучу F2 же Enter клавиштеринин жардымында редактирлөө режимине өтө алат. goEditing касиети true болуусу зарыл

Cells[ARow:integer; ACol:integer]:string - эки өлчөмдүү массив түрүндөгү касиети таблицадагы ARow номерлүү саптын жана ACol номерлүү мамычанын кесилишиндеги ячейканын маанисин кайтарат.

Компоненттин окуялары:

OnColumnMoved – мамычанын ордун алмаштыргандан кийин пайда болуучу окуя. Төмөнкү түрдө аныкталган:

procedure TForm1.StringGrid1ColumnMoved (Sender: TObject; FromIndex, ToIndex: Integer);

FromIndex – параметри мамычанын ордун алмаштырганга чейинки орундун номери.

ToIndex – параметри мамычанын ордун алмаштыргандан кийинки орундун номери.

OnRowMoved – саптын ордун алмаштыргандан кийин пайда болуучу окуя.

procedure TForm1.StringGrid1RowMoved(Sender: TObject; FromIndex, ToIndex: Integer);

Параметрлери OnColumnMoved окуясындай эле.

OnGetEditMask –масканы орнотуу, ячейканын маанисин редактирлөөгө өткөндө пайда болуучу окуя.

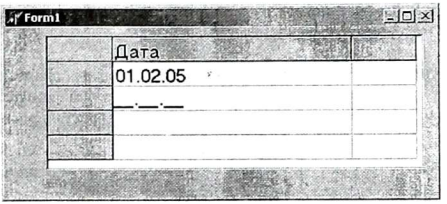
procedure TForm1.StringGrid1GetEditMask(Sender: TObject; ACol, ARow: Integer; var Value: String);

Мында, ACol, ARow мамычанын жана саптын номери, Value масканын шаблону.

Мисалы:

```
procedure TForm1.StringGrid1GetEditMask(Sender: TObject; ACol, ARow: Longint; var Value: string);
begin
  if ACol = 1 then
  Value := '!00/00/00;1;_';
end;
```

Колдонуучу биринчи мамычанын ячейкаларын редактилөөгө өткөндө, кийирүүнүн шаблону пайда болот.



OnGetEditText – ячейканы редактирлөөгө өткөндө пайда болот.

procedure TForm1.StringGrid1GetEditText(Sender: TObject; ACol, ARow: Integer; var Value: String);

Параметрлери OnGetEditMask окуясындай эле, айырмасы Value параметри ячейкадагы саптын маанисин көрсөтөт.

OnSelectCell – ячейканы тандаган учурда пайда болуучу окуя.

procedure TForm1.StringGrid1SelectCell(Sender: TObject; ACol, ARow: Integer; var CanSelect: Boolean);

CanSelect параметри ячейканы тандоого уруксат берүүнү же бербөөнү ишке ашырат.

Төмөнкү мисалда маани киргизилген ячейканы кайра тандоого уруксат берилбейт.

```
procedure TForm1.StringGrid1SelectCell(Sender: TObject; ACol, ARow: Integer; var CanSelect: Boolean);
begin
CanSelect := (StringGrid1.Cells[ACol,ARow]='')
end;
```

5. **TImage** – формада графикалык сүрөттү көрсөтөт. bmp, emf, wmf жана ico форматтарын кабал алат. Компонентти формага жайгаштырганда тик бурчтук пайда болот. Эгерде сүрөт проектирлөө учурунда компонентке жайгаштырылган болсо, ал компиляция учурунда exe файлга жазылат да, exe файлдын өлчөмү бир кыйла чоңоюп кетет.

Касиеттери:

Center – логикалык типтеги талаа, мааниси true болгон учурда, сүрөттүн өлчөмү компоненттин өлчөмүнөн кичине болсо, сүрөт компоненттин ортосуна жайгашат. AutoSize касиети true учурунда эффект байкалбайт.

Strecth – логикалык типтеги талаа, мааниси true болгон учурда, сүрөттүн өлчөмү компоненттин өлчөмүнө ылайык кысылат же чоюлат. AutoSize касиети true учурунда эффект байкалбайт.

Picture – сүрөт касиети. TBitmap тибиндеги объекттик типтеги талаа. TBitmap бир кыйла татаал объекттик типтен түзүлгөндүктөн анын касиеттерине

токтолбойбуз. Айта кетчүү нерсе, бизге тааныш болгон LoadFromFile жана SaveToFile мотеддору ушул касиетте аныкталган.

6. **TShape** – компоненти формада түрдүү геометриялык фигураларды чийүү үчүн колдонулат. Анын **Shape** касиетинен конкреттүү фигураны тандоого болот. Анын мааниси төмөнкү константалардын биринде боло алат:

Мааниси	Фигуранын формасы
stCircle	Айлана
stEllipse	Эллипс
stRectangle	Тик бурчтук
stRoundRect	Бурчтары томпок болгон тик бурчтук
stRoundSquare	Бурчтары томпок болгон квадрат
stSquare	Квадрат

Мындан сырткары фигуранын сызыгын **Pen** – калем жана түсүн **Brush** – кист касиеттеринин жардамында өзгөртүүгө болот.

7. **TBevel** – рамка компоненти, кооздук үчүн рамка жана сызыктарды түзүү үчүн пайдаланылат. Окуяга ээ эмес. Анын көрүнүшүн башкаруучу эки касиетине токтолобуз.

Shape – рамканын образы. Анын мааниси төмөдөгү константалардын биринде боло алат:

Мааниси	Рамканын формасы
bsBox	Тик бурчтук. Ички областтынын көрүнүшү Style касиетинин мааниси менен аныкталар
bsFrame	Тик бурчтук. Ички областты өзгөрбөйт
bsTopLine	Тандалган областтын жогорку чек арасы гана көрсөтүлөт
bsBottomLine	Тандалган областтын төмөнкү чек арасы гана көрсөтүлөт
bsLeftLine	Тандалган областтын сол чек арасы гана көрсөтүлөт
bsRightLine	Тандалган областтын оң чек арасы гана көрсөтүлөт
bsSpacer	Рамка көрсөтүлбөйт. Проектирлөө учурунда областтарга бөлүүнү белгилөө үчүн гана колдонулат

Style – рамканын сызыктарынын көрүнүшүнүн стили. bsLowered – төмөн басылган же bsRaised – томпок сызыктарды аныктайт.

8. **TScrollBox** – түрүү кутучасынын компоненти. Формага өлчөмү экрандан бир кыйла чоң болгон түрүлүүчү областты түзөт. Бул областка башка компоненттерди жайгаштырууга болот. Негизги касиеттери жана окуялары мурда карап өткөн касиеттер жана окуялар менен дал келет.

9. **TCheckListBox** – TListBox жана TCheckBox компоненттерин бириктирүүдөн пайда болгон компонент. Негизги касиеттери жана окуялары TListBox жана TCheckBox компоненттеринин касиеттери жана окуялары менен дал келет.

10. **TSplitter** – бөлгүч,терезени өлчөмү өзгөрүүчү бир нече областка бөлүүнү ишке ашыруучу компонент. Колдонууга өтө жөнөкөй, аны туура жайгаштыруу гана талап кылынат.

Касиеттери:

Beveled – компоненттин көрүнүшү, логикалык типтеги чоңдук, мааниси true болгон учурда компонент үч өлчөмдүү көрүнүшкө келет.

AutoSnap – өлчөмү кыскарып жаткан областтын өлчөмүн нөлгө чейин кыскартууга уруксат берүү. Логикалык типтеги чоңдук, true болгон учурда нөлгө чейин кыскартууга болот, антпесе өлчөм MinSize касиетинин маанисине чейин гана кыскарат.

MinSize – областтын өлчөмүнүн минималдык мааниси, бүтүн оң сан (багыты align касиетинен аныкталат).

ResizeStyle – жылдыруу процессинде бөлгүчтүн экрандагы көрүнүшүн аныктай. Төмөнкү маанилердин биринде боло алат:

Мааниси	Көрсөтүүнүн ыкмасы
rsNone	Бөлүүнүн болжолдуу орду көрсөтүлбөйт. Чычкандын кнопкасын кое бергенден кийин областтын өлчөмү өзгөрөт
rsLine	Бөлүүнүн болжолдуу орду туташ кара сызык менен көрсөтүлөт
rsPattern	Бөлүүнүн болжолдуу орду пунктир сызык менен көрсөтүлөт
rsUpdate	Областтын өлчөмү бирге өзгөрөт

Компоненттин окуялары:

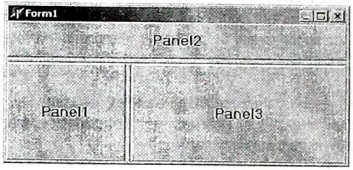
OnCanResize – компонент областтын өлчөмүн өзгөртө алышын көрсөтүүчү окуя, областын өлчөмүн өзгөртүп жатканда пайда болот.

procedure TForm1.Splitter1CanResize(Sender: TObject; var NewSize: Integer; var Accept: Boolean);

Мында, NewSize областтын жаңы өлчөмү, Accept жылдырууга мумкүн экендигин көрсөтүүчү чоңдук.

OnMoved – бөлгүчтү жылдыргандан кийин пайда болуучу окуя.

Компонентти пайдалануу көрсөтүүчү мисалды карайбыз. Формага панель жайгаштырып, анын align касиетине alLeft маанисин орнотобуз. Бөлгүчтү жайгаштырабыз, align касиетине alLeft маанисин орнотобуз. Формага экинчи панелди жайгаштырып, align касиетине alTop маанисин орнотобуз. Экинчи бөлгүчтү жайгаштырып, анын да align касиетине alTop маанисин орнотобуз. Форманын бош калган областына үчүнчү панелди жайгаштырып, анын align касиетине alClient маанисин орнотобуз. Программаны жүктөгөндө сүрөттөгү терезе пайда болот.



11. **TStaticText** – мурда каралган TLabel компонентинен айырмасы тексттти рамканын ичинде чыгарууга мүмкүн. Рамканын формасы BorderStyle касиетинин жардамында орнотулат.

12. **TControlBar** – башкаруу тилкеси, компоненттин областтына жайгаштырылган визуалдык компоненттер үчүн атайын сызгыч түзүлөт. Бир нече кнопканы бир сызгычтын үстүнө жайгаштыруу үчүн алгач панель компоненттин жайгаштыруу керек.

13. **TLabelEdit** – TLabel жана TEdit компоненттеринин биригүүсүнөн пайда болгон компонент. Ар бир жолу TEdit компонентин пайдаланганда ага кийирилүүчү маани жөнүндөгү кабарды жазуу үчүн TLabel компонентин колдонобуз. Бул компоненттерди бири-бирине дал келтирип жайгаштыруу жана жылдыруу убакытты талап кылат. Ал эми TLabelEdit компоненти бул аракеттерди жасоого бир кыйла ыңгайлуу.

Касиеттери:

LabelPosition – кабар жазылуучу саптын кийирүү тилкесине салыштырмалуу жайгашышы. Анын мааниси lpAbove – жогору, lpBelow – төмөн, lpLeft – сол жана lpRight – оң жагында боло алат.

LabelSpacing – кабар жазылуучу сап менен кийирүү тилкесинин ортосундагы бош аралык, бүтүн оң сан.

EditLabel – кабар жазылуучу саптын көпчүлүк касиеттерин орнотуучу талаалар жана окуялардын тизмеси.

14. **TColorBox** – түстөрдүн комбинацияланган тизмесинин компоненти. Касиеттери жана окуялары комбинацияланган тизменин касиеттеринен жана окуяларынан турат. Айырмасы, тандалган түстү **Selected** касиети аркылуу алууга болот.

Компоненттин Style касиетинде тизмеде пайда болуучу түстөрдүн жыйындысы тандалат, алар логикалык типтеги талаалар. Таблицада бул талаалардын түшүндүрмөсү берилген.

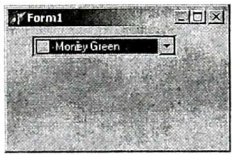
Аталышы	Түшүндүрмөсү
cbStandardColors	Тизмеде стандарттуу 16 түс көрсөтүлөт
cbExtendedColors	Кеңейтилген түстөр пайда болот. Алар

	clMoneyGreen, clSkyBlue, clCream, жана clMedGray
cbSystemColors	Windows ОС-сында пайдаланылган түстөрдүн тизмеси пайда болот
cbIncludeNone	Тизмеге clNone мааниси кошулат. cbSystemColors стили орнотулганда гана пайда болот
cbIncludeDefault	Тизмеге clDefault мааниси кошулат. cbSystemColors стили орнотулганда гана пайда болот
cbCustomColor	Тизмеге clCustom мааниси кошулат. Колдонуучу бул маанини тизмеден тандаганда түстөрдүн диалогдук терезеси пайда болуп, андан колдонуучу тандаган түс ушул мааниге ыйгарылат
cbPrettyNames	Түстөрдүн аттарынын алдындагы префикстик бөлүк жазылбай. Мисалы тизмедеги «clBlack» мааниси «Black» атына өзгөрөт

Мисалы, формага TColorBox компонентин жайгаштырабыз жана анын OnClick окуясына төмөнкү сапты жазабыз.

Form1.Color:=Colorbox1.Selected;

Программаны жүктөп, комбинацияланган тизмеден түстү тандаганда, форманын түсү өзгөрөт. Сүрөттө программанын иштеп жаткандагы көрүнүшү берилди.



2.5. Win32 барагынын компоненттери

Windows ОС-нын Windows 95, 98, NT жана 2000 32 биттүү версиялары үчүн арналган компоненттер жайгашкан барак. Алардын бир нечеси менен таанышабыз.

1. **TTabControl** – көп беттүү компоненттерди иштетүүдө пайдаланылуучу закладкалардын компоненти. Закладкалар бир гана областка ээ болушуп, алар алмашканда область алмашпайт. Закладкалардын бөркүнө текст жана сүрөт жайгаштырууга болот.

Касиеттери:

HotTrack – логикалык типтеги талаа, мааниси true болсо, чычкандын көрсөткүчү закладкага келгенде, анын тексти түсүн өзгөртөт.

MultiLine – логикалык типтеги талаа, мааниси true болсо, закладкалар бир нече сапка бөлүнүп жайгаштырылат. Антпесе бир сапка жайгашып оң жана сол жакка түрүүчү кнопкалар пайда болот.

RiggedRight – логикалык типтеги талаа, мааниси true болсо, закладкалардын саптары мамычалар боюнча тегизделбейт. MultiLine касиети true болгон учурда колдонулат.

ScrollOppasite – логикалык типтеги талаа, мааниси true болсо, закладкаларды тандаганда бирин-бири түрүп жаткандай эффект менен тандалат. MultiLine касиетин да тиешелеш түрдө өзгөртөт.

Style – закладкалардын стили. tsTab - жөнөкөй, tsButton – кнопка же tsFlatButton – жалпак кнопка түрүндө боло алат.

TabPosition – заладкалардын областка салыштырмалуу жайгашуусун аныктайт. Анын маанилери tpTop – жогору, tpBottom – төмөн, tpLeft- сол же tpRight – оң жагында жайгаштырат. tpLeft, tpRight мааналери Style касиетинин мааниси tsTab болгондо гана орнотулат.

TabIndex – тандалган закладканын номери. Бүтүн сан, терс болсо бир да закладка тандалган эмес.

Tabs – закладкалар, TString объекттик типтеги талаа. TListBox компонентинин Items касиетине окшош.

Компонент өзгөчө окуяларга ээ эмес.

2. **TPageControl** – закладкалары бар панелдердин жыйындысы. Ар бир панель өзүнүн интерфейстик элементтерине ээ боло алат жана закладкага щелчок жасоо менен тиешелүү панель тандалат. Башка касиеттери TTabControl компонентине окшош.

3. **TImageList** – сүрөттөрдүн коллекциясы. Бирдей өлчөмдөгү бир нече сүрөттөрдү сактоо үчүн колдонулат. Көпчүлүк компоненттердин Images касиетине ушул компонентти көрсөтүү керек.

Касиеттери:

AllocBy – жаңы кошулуучу элементтердин саны. Мисалы, AllocBy касиети төрткө барабар болсо, бешинчи сүрөттү кошкондо тизменин саны сегизге өзгөрөт.

BkColor – сүрөттүн фонунун түсү. Сүрөттүн фону түссүз болгондо же Transparent Color касиетине түс орнотулган учурда колдонулат.

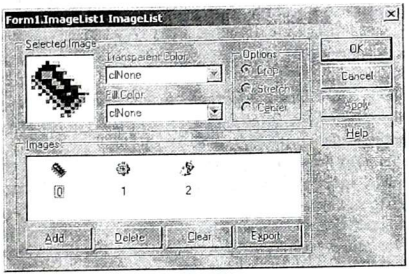
BlendColor – алдынкы фондун түсү.

DrawingStyle – сүрөттү экранга чыгаруунун ыкмасы. dsTransparent маанисинде сүрөт түссүз, калган учурларда BkColor жана BlendColor түстөрүнүн катышынан турган фон менен тартылат.

Hight жана **Width** сүрөттүн бийиктиги жана узундугу, пиксел менен эсептелинет.

Сүрөттөрдү коллекцияга кийирүү үчүн формага жайгаштырылган компоненттин контекстик менюсун пайдаланабыз. Андан ImageList Editor опциясын тандап, атайын терезени ачабыз. Терезенин көрүнүшү сүрөттө көрсөтүлгөн. Мында Selected Image – учурда тандалган сүрөт, Images –

коллекцияга киргизилген сүрөттөрдүн тизмеси, Transparent Color – сүрөттө түссүз кылып көрсөтүлүүчү түс, Options – сүрөттү позициялоонун ыкмасы.



Add кнопкасынын жардамында жаңы сүрөт кошулат, Delete кнопкасы учурдагы сүрөттү өчүрөт, Clear – сүрөттөрдүн тизмесин тазалайт, Expotr –сүрөттү башка файлга экспорттойт.

4. **TRichEdit** – формattалган тексттин көп саптуу редактору. Standard барагындагы TMemo компонентинен айырмаланып TRichEdit компонентиндеги текст кеңейтилген тексттик форматтын (RTF - Rich Text Format) эрежелерине баш ийет, ошондой эле тексттин түсү, шрифти сыяктуу жана башка касиеттерин өзгөртүүгө болот.

Paragraph касиети **TRichEdit** компонентинин негизги касиеттеринин бири катары эсептелинет жана ал абзацтарды форматтоонун объекттик типтеги касиети. Анын **FirstIndent, LeftIndent, RightIndent** бүтүн типтеги касиеттерине тиешелеш түрдө абзацтын биринчи сабынын жана анын сол, он жагынан калтырылуучу бош орундун (пикселде) маанилери көрсөтүлөт. **Numbering** – мааниси nsBullet болсо, абзацтын сол жагына маркер коюу менен аны тизмеге айландырат, nsNone болсо жөнөкөй абзац.

SelAttributes – белүнүп алынган тексттин мүнөзүн өзгөртүүнүн объекттик типтеги касиети. Негизинен шрифтин касеттеринен турат. Алар:

Name – шрифтин аты, string тибинде.

Color – шрифтин түсү.

Size – шрифтин өлчөмү, бүтүн сан.

Style – шрифтин стили, fsBold – кара, fsItalic – *курсив*, fsUnderline – <u>астын сызылган</u> жана fsStrikeOut – ~~чийилген~~ маанилери массив түрүндө ыйгарылат. Мисалы, ***Richedit1.SelAttributes.Style:=[fsBold,fsItalic];***

Орнотулган стилди алып салуу үчүн [ ] бош массиви жазылат.

5. **TTrackBar** - регулятор. Программадагы кээ бир чоңдуктардын маанисин башкаруу үчүн колдонулат. Мисалы, мультимедиялык

программада үндүн бийиктигин өзгөртүү.

Касиеттери:

Frequency – бөлүү сызыкчаларынын жыштыгы. Бүтүн сан.

Min жана **Max** – мүмкүн болгон чектин минималдык жана максималдык мааниси.

Orientatoin – ориентациясы, trHorizontal – горизонталдык же trVertical вертикалдык абалда.

SliderVizible – логикалык типтеги талаа, true болгон учурда жылып жүрүүчү жебе көрсөтүлөт.

ThumbLength –тилкесинин кендиги, бүтүн сан.

TickMarks – бөлүү сызыкчаларынын жайгашуу абалы. tmBottomRight – төмөн, tmTopLeft – жогору же tmBoth – эки тарабында тең маанилеринин биринде боло алат.

Position – жылып жүрүүчү жебенин учурдагы ордун аныктайт, тиби бүтүн сан. Программада анын маанисин окууга жана өзгөртүүгө болот, маанисин өзгөркөн учурда OnChange окуясын пайда кылат.

6. **TProgressBar** – процесстин индикатору. Бул компоненттин жардамында узак убакыт ичинде аткарылуучу процесстин жүрүшүн көрсөтүүгө болот, мисалы файлдарды диске көчүрүү процессин.

Көпчүлүк касиеттери TtrackBar компонентинин касиеттериндей.

Smooth – логикалык типтеги талаа, мааниси true болгон учурда тилке үзгүлтүксүз сызык, антпесе сегменттер менен толтурулат.

7. **TUpDown** – сандык регулятор. Сандык мааниге ээ болгон компонент менен бирге иштөөчү жогору жана төмөн жебелери бар кнопкалардан турган компонент. Маанилеринин диапозону –-32768 ..32767 чейинки бүтүн сандар. Анын касиеттери:

Associate – сандык регулятордун маанисин көрсөтүү максатында ага бириктирилген элемент, мисалы, бул касиетке TEdit компонентин көрсөтүүгө болот.

ArrowKeys – логикалык типтеги талаа, мааниси true болгон учурда, сандык регулятордун маанисин багыттык клавишалардын жардамында өзгөртүүгө уруксат берилет.

Increment – маанилердин өзгөрүү кадамы. Бүтүн сан.

Thousands – логикалык типтеги талаа, мааниси true болгон учурда, ар бир үч цифрадан кийин үтүр белгиси коюлат.

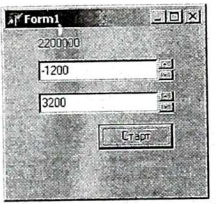
Wrap – логикалык типтеги талаа, мааниси true болгон учурда, регулятордун маанеси максимумга жеткенде кайра минимум маанисине өтөт жана тескерисинче.

Мисал катары сандардын берилген диапозонунан так сандардын суммасын табуунун программасын карайбыз. Формага экиден TEdit, TUpDown жана бирден TLabel, TButton компоненттерин жайгаштырабыз. UpDown1 жана UpDown2 компоненттеринин Max касиетине 3200 жана Min касиетине -3200 маанилерин кийирип, Associate касиеттерине тиешелеш түрдө Edit1 жана Edit2

компоненттерин тандайбыз. Button1 компонентинин Caption касиетине «Старт» сөзүн жазып, анын OnClick касиетине төмөндөгү программалык кодду жазабыз:

```
procedure TForm1.Button1Click(Sender: TObject);
var
i,s,n,m:integer;
begin
If UpDown1.Position<UpDown2.Position then
Begin
s:=0;
n:=UpDown1.Position;
m:=UpDown2.Position;
end
else
begin
n:=UpDown2.Position;
m:=UpDown1.Position;
end;
for i:=n to m do  if odd(i) then s:=s+i;
Label1.Caption:=inttostr(s);
end;
```

Сүрөттө программанын иштеп жаткан учурундагы сүрөтү көрсөтүлдү.



8. **TAnimate** - мультипликатор. Чоң эмес AVI форматындагы видеоклипти ойноо үчүн арналган, видеоклиптер үнсүз ойнолот. Анын касиеттери:

Active – логикалык типтеги талаа, мааниси true болгон учурда мультипликатор активдешет.

CommonAVI – Windows ОС-нын стандартуу клиптеринен тандоо. Бул талаанын маанилери төмөнкүлөрдүн биринде боло алат:

Клиптин аты	Мазмуну
aviNone	FileName касиетинде көрсөтүлгөн клип
aviFindFolder	Папкаларды издөө
aviFindFile	Файлды издөө
aviFindComputer	Компьютерди издөө
aviCopyFiles	Файлдарды көчүрүү
aviCopyFile	Файлды көчүрүү
aviRecycleFile	Файлды корзинага жылдыруу
aviEmptyRecycle	Корзинаны тазалоо
aviDeleteFile	Файлды өчүрүү

Windows ОС-нын клиптеринин өлчөмдөрү ар түрдүү, ушул себептүү AutoSize касиетине true маанисин орнотуу зарыл.

StartFrame жана **StopFrame** – ойнолуучу клиптин кадрларынын диапозонунун баштапкы жана акыркы маанилери, бүтүн сандар. Бул маанилер орнотулса, клиптин көрсөтүлгөн кадрлардан турган фрагменти гана ойнолот.

FrameCount – клиптеги кадрлардын жалпы саны, бүтүн сан.

Repetitions – клипти кайталап ойноонун саны, бүтүн сан. Нөл болгон учурда, клип чексиз кайталанып ойнолот жана active касиетинин мааниси false болгондо ойноо токтойт.

Формага TAnimate жана эки TButton компоненттерин жайгаштырабыз. Button1 компонентинин Caption касиетине «Ойноо» жана Button2 компонентинин ушул эле каситине «Токтотуу» сөздөрүн жазабыз. Animate1 компонентинин CommonAVI касиетине маанилердин тизмесинен aviEmptyRecycle сабын тандайбыз.

Button1 кнопкасынын OnClick окуясына **Animate1.Active:=true;** сабын, ал эми Button2 кнопкасынын OnClick окуясына **Animate1.Active:=false;** саптарын жазабыз. Программаны жүктөп, кнопкалардын жардамында мультипликатордун иштөөсүн башкарабыз.

Программанын иштөө учурундагы терезеси сүрөттө берилди.

9. **TDateTimePicker** – убакыт же датаны тандоо. Датаны же убакытты чычкандын жардамында тандоого мүмкүндүк берүүчү компонент. Пайдалануура өтө ыңгайлуу жана жөнөкөй элемент. Анын касиеттери:

DateFormat – датанын форматынын dfShort – кыска же dfLong – узун көрүнүшүн аныктайт.

DateMode – маанилерине жараша компонент dmComboBox – комбинацияланган тизме же dmUpDowm – сандык регулятор түрүндө иштешин аныктайт.

Kind – мааниси dtkDate болгондо компонент датаны, ал эми dtkTime болгондо убакытты кийирүү үчүн арналат.

ShowCheckbox – логикалык типтеги талаа, true болгон учурда кийирүү тилкесинде тандоо кутучасы пайда болот. Кутучанын абалын Checked касиетинин мааниси аркылуу билүүгө болот.

MaxDate жана **MinDate** – датаны көрсөтүүнүн максималдык жана минималдык маанилери орнотулат.

11. **TMonthCalendar** – календарь компоненти. Чычкандын жардамында датаны тандоо үчүн колдонулат. Компоненттин бир нече кеңейтилген касиеттери бар.

Date – календардан тандалган күндү көрсөтүүчү талаа.

FirstDayOfWeek – календарда биринчи туруучу жуманын күнү. dowLocaleDefault мааниси коюлса компьютерге орнотулган Windows ОС-ынын стандартына ылайык коюлат.

MultiSelect – логикалык типтеги талаа, true болгон учурда колдонуучу Shift клавишасынын жардамында бир нече күндү календардан тандай алат.

EndDate – MultiSelect касиети true болгон учурда, колдонуучу тандаган эң акыркы датаны кайтарат. Тандоонун баштапкы датасы Date касиетинен алынат.

CalColors – календардын элементтеринин түсүн орнотуучу касиет. Бул касиет TColor тибиндеги бир нече талаалардан турат. Ал талаалардын аталышы жана түшүндүрмөсү таблицада берилди:

Аталышы	Түшүндүрмөсү
BackColor	Фондун түсү
MonthBackColor	Айдын күндөрү жазылган зонанын фонунун түсү
TextColor	Айдын күндөрүнүн түсү
TitleBackColor,	Бөрктүн фонунун түсү
TitleTextColor,	Бөрктүн текстинин түсү
TrailingTextColor	Учурдагы айга тийиштүү эмес күндөрдүн түсү

ShowToday – логикалык типтеги талаа, true болгон учурда календардын төмөн жагында учурдагы дата көрсөтүлөт.

ShowTodayCircle - логикалык типтеги талаа, true болгон учурда календарда учурдагы дата кызыл айлана менен белгиленет.

WeekNumbers – логикалык типтеги талаа, true болгон учурда календарда жумалардын номерлери көрсөтүлөт.

10. **TStatusBar** – статус панели. Түрдүү кызматчы кабарларды көрсөтүү үчүн колдонулат. Форманын төмөн жагына жайгашып, өлчөмүн

формага жараша өзү өзгөртөт. Кабарларды жайгаштыруу үчүн анын областты бир нече панелдерге бөлүнөт. Панелдерге бөлүү үчүн компоненттин контекстик менюсунун Panels Editor опциясынан пайдаланабыз.

11. **TToolBar** – инструменттердин панели. Бул компонент TBitBtn кнопкаларынын контейнери болуп саналат жана кнопкаларды кошкондо же өчүргөндө өлчөмүн автоматтык түрдө өзгөртүрөт. Компоненттин контексттик менюсунун New Button опциясынын жардамында жаңы кнопка, New Seperator опциясынын жардамында кнопкаларды группаларга ажыратып туруучу бөлүгүч кошулат. Мындан сыртқары бул компонентке TContolBar компоненттерин да жайгаштырууга болот.

Касиеттери:

ButtonHeight, ButtonWidth – кнопкалардын бийиктиги жана узундугу. Бардык кнопкалардын өлчөмдөрү бирдей болот.

DisabledImages – кнопка «өчүп» турган абалга дал келүүчү сүрөттөрдүн коллекциясы (TImageList) орнотулат.

HotImages – кнопкалардын үстүнө чычкандын көрсөткүчү келген кезде пайда болуучу сүрөттөрүнүн коллекциясы (TImageList) орнотулат.

Images – жөнөкөй учурдагы кнопкалардын сүрөттөрүнүн коллекциясы (TImageList) орнотулат.

List – логикалык типтеги талаа, true болгон учурда, кнопканын бөркү анын сүрөтүнүн сол жагына, антпесе төмөн жагына чыгат.

ShowCaption – логикалык типтеги талаа, true болгон учурда, кнопканын бөркү панелде көрсөтүлөт.

Инструметтер панелинде пайдалануучу кнопкалар башка кнопкалардан айырмаланган касиеттерге ээ, ал касиеттер:

Style – кнопканын стили, касиеттин мааниси таблицада көрсөтүлгөн маанилердин биринде боло алат:

Мааниси	Стили
tbsButton	Командалык кнопка
tbsCheck	Тандалуучу кнопка
tbsDropDown	Меню кнопка
tbsSeperator	Бөлгүч кнопка
tbsDivider	Таякча түрүндөгү бөлгүч

AllowAllUp – логикалык типтеги талаа, бир группага кирген кнопкалардын бардыгы төмөн басылбаган абалда болсо, талаа true маанисин кайтарат.

DropdownMenu – төмөн түшүүчү меню, кнопкага менюну бириктирет жана кнопка басылганда меню ачылат. Бул талаага TPopupMenu компоненти көрсөтүлөт.

Index – кнопканын катар номери, бүтүн сан.

Marked – логикалык типтеги талаа, true болгон учурда кнопка белгиленген, үстү тор менен капталган көрүнүштө болот.

MenuItem – талаа кнопканы кайсы бир менюнун опциясы менен байланыштырат. Эгерде мындай байланыш орнотулса, кнопканы басканда ал менюнун опциясынын аракетин аткарат.

12. **TCoolBar** – инструменттердин (Cool - мыкты) панели. TToolBar компонентинен айырмасы, кнопкалардан башка элементтерди да жайгаштырууга боло турган контейнерлерден турат. Контексттик менюсунун Bands Editor опциясы же Bands касиетинин жардамында TCoolBands объектик тибиндеги жылып жүрүүчү контейнерлер кошулат. Бул контейнерлерге каалагандай башкаруу элементтерин жайгаштырууга болот. Компоненттин касиеттери:

Bitmap – фондун сүрөтү.

FixedOrder – логикалык типтеги талаа, мааниси true болсо жана колдонуучу TCoolBands тибиндеги контейнерди жылдырган учурда, анын катарын алмаштырууга тыюу салат.

FixedSize – логикалык типтеги талаа, мааниси true болсо TCoolBar компонентинин өлчөмү фиксирленген болот.

Vertical – логикалык типтеги талаа, мааниси true болсо контейнерлер жогорудан төмөн карай, антпесе солдон оңго карай жайгашышат.

BandMaximize – контейнерди жаюунун аракети. Маанисинин константалары bmNone – контейнерди жаюуга аракет көрсөтүлбөдү, bmClick – чычкандын сол кнопкасы менен бир щелчок жасоо жана bmDblClick – эки щелчок жасоо менен жаюуга болот дегенди түшүндүрөт. Колдонуучу контейнерлерди жылдырып бир катарга жайгаштырганда, алар бирин-бири жаап калат. Мындай учурда жабылып турган контейнерди тез ачууну ушул касиет ишке ашырат. Ал үчүн колдонуучу контейнердин сол жагындагы вертикалдык сызыкчага, касиетке орнотулган мааниге жараша, бир же эки щелчок жасоосу керек.

TCoolBands контейнерлеринин касиеттери:

Break – логикалык типтеги талаа, мааниси true болгон учурда, контейнерлер бири – бири менен туташпайт (проектирлөө учурунда true маанисин орнотуу максатка ылайыктуу). Бул касиет FixedSize касиетинен көз каранды.

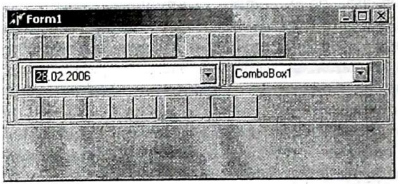
FixedSize – логикалык типтеги талаа, мааниси true болсо учурда контейнердин өлчөмү фиксирленген болот да контейнерди жылдырууга уруксат берилбейт (контейнердин сол жагындагы вертикалдык сызыкча пайда болбойт).

Control – контейнерге жайгаштырылуучу башкаруу элементи көрсөтүлөт.

Мисал үчүн формага TCoolBar компонентин жайгаштырып, анын Bands касиетинин жардамында үч контейнер кошобуз. Эми формага (CoolBar1 компонентине эмес) эки TToolBar жана бир TControlBar компоненттерин жайгаштырабыз. TToolBar жана TControlBar компоненттерине мурда айтылган жолдор менен бир нече кнопкаларды жана башкаруу элементтерин жайгаштырабыз. Объекттерди дарак түрүндө көрсөтүүчү терезеден TCoolBand элементтерин кезеги менен тандап алардын Contol касиеттерине тиешелеш

түрдө ToolBar1, ToolBar2 жана ControlBar1 компоненттерин тандап чыгабыз. CoolBar1 компонентинин AutoSize касиетине true маанисин орнотуп, программаны жүктөйбүз.

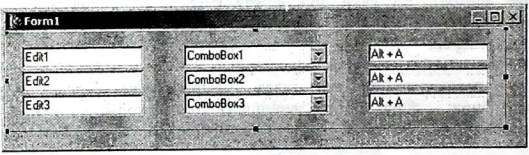
Эми контейнерлерди жылдырып, ордуларын алмаштырып жана өлчөмдөрүн өзгөртүп, текшерип көргүлө. Программанын иштеп жаткан учурундагы көрүнүшү сүрөттө берилди.



13. **TPageScroller** – түрүлүүчү бет. Control касиетине башкаруу элементи

көрсөтүлөт, элементтин өлчөмү TPageScroller компонентинин өлчөмүнөн чоң болсо, жебенин сүрөтү бар түрүү кнопкаларын пайда кылат. TScrollBox компоненти сыяктуу компонент болуп, айырмасы түрүү тилкеси оң жана сол же жогору жана төмөн жагында гана боло алат.

Формага панель компонентин жайгаштырып, анын бийиктигине (Height касиети) 100 жана узундугуна (Width касиети) 560 маанилерин орнотобуз. Панелди болжолдуу түрдө үч бөлүккө бөлүп тургандай кылып бир нече башкаруу элементтерин мамыча түрүндө жайгаштырабыз (сүрөттү кара).



Формага TPageScroller компонентин жайгаштырып, анын бийиктигин жана узундугун тиешелеш түрдө 100 жана 200 деп белгилейбиз. Control касиетинин тизмесинен Panel1 компонентин тандайбыз. Бул учурда панель PageScroller1 компонентинин ичине жайгашат.

Программаны жүктөп текшерип көргүлө. Эгерде панелде жайгашкан элементтерди өзгөртүүгө туура келсе, PageScroller1 компонентинин Control касиетин тазалап салып Enter клавишасын басуу керек.

14. **TComboBoxEx** – компоненти TComboBox сыяктуу эле компонент,

айырмасы тизменин саптарында пиктограммалык сүрөттөрдү көрсөтүү мүмкүнчүлүгү бар.

2.6. System барагынын компоненттери

System барагында компьютердин жана ага орнотулган системанын өзгөчөлүктөрүнө жараша иштөөчү компоненттер жайгашкан. Алардын ичинен үч компонентти гана карайбыз.

1. **TTimer** – таймер, убакыт генератору менен иштөөчү компонент. Убакыттын белгилүү аралыктарында өз алдынча ишке кирүүчү аракеттерди жазуу үчүн арналган.

Interval касиетинде аракети ишке кирүү убакыт аралыгы көрсөтүлөт. Бүтүн сан, бир секунда минге барабар. Active касиетинин мааниси true болгон учурда көрсөтүлгөн убакыт аралыгы өткөн сайын, анын жалгыз **OnTimer** окуясына жазылган программалык код аткарылып турат.

2. **TPaintBox** – сүрөт тартуу үчүн область. Сүрөт тартуу үчүн анын **TCanvas** объекттик типтеги **Canvas** касиети пайдаланылат. TCanvas объекттик тибинин көптөгөн башка касиеттери жана методдору тиркемеде берилди.

3. **TMediaPlayer** – мультимедиянын компоненти. MCI (Media Control Interface) драйверлери иштете алуучу мультимедиялык форматтагы файлдарды ойноону ишке ашырат жана ал Window ОС-нын Универсалдык мультимедиялык плееринин мүмкүнчүлүктөрүнө ээ.

Касиеттери:

AutoEnable – логикалык типтеги талаа, true болгон учурда башкаруу кнопкаларынын абалдары автоматтык түрдө өзгөрөт.

AutoOpen – логикалык типтеги талаа, true болгон учурда мультимедиялык файлды ойноого арналган MCI түзүлүшү автоматтык түрдө ачылат.

AutoRewind – логикалык типтеги талаа, true болгон учурда мультимедиялык файл ойнолуп бүткөндөн соң, автоматтык түрдө башталышына түрүү жүргүзүлөт.

ColoredButtons – кнопкалардын түстүү болушун башкарат. Кнопка түстүү болсо, кнопкалардын константалары жазылган тийиштүү саптарга true мааниси орнотулат.

Кнопканын аты	Кызматы
btPlay	Ойноо
btPause	Пауза (убактылуу токтотуу)
btStop	Токтотуу
btStep	Бир нече кадр алдыга өтүү
btBack	Бир нече кадр артка өтүү
btNext	Кийинки жолчого өтүү

btPrev	Мурдагы жолчого өтүү
btRecord	Жазуу
btEject	Диск жайгаштырылуучу түзүлүштү ачуу же жабуу

DeviceType – FileName касиетинде көрсөтүлгөн ойнолуучу мультимедиялык файлдын тибине туура келүүчү түзүлүштү тандоо. dtAutoSelect мааниси автоматтык түрдө тандоону ишке ашырат. Башка типтердин болушу системага орнотулган түзүлүштөрдүн драйверлеринен көз каранды. Алар таблицада берилди.

Талаанын мааниси	Файлдын тиби	Тректерди иштетиши	Экран терезени иштетиши
dtAVIVideo	AVI видео	-	+
dtCDAudio	Аудио CD диск	+	-
dtDAT	Цифралык аудио кассета	+	-
dtDigitalVideo	AVI, MPG, MOV	-	+
dtMMMovie	MM фильмдер	-	+
dtOverlay	Аналогдук видео	-	+
dtScanner	Сканерден сүрөттү алуу	-	-
dtSequencer	MIDI файлдар	+	-
dtVCR	Видео кассетага жазуу	-	+
dtWaveAudio	WAV файлдар	-	-

Display – видеоинформацияны көрсөтүү үчүн орун (экран). Мааниси бош болгон учурда жаңы терезеде көрсөтүлөт. Экран катары TPanel компонентин пайдаланууга болот.

EnabledButtons – кнопкаларды пайдаланууга тыюу салууну ишке ашырат. Кнопкалардын константалары жана кызматтары ColoredButtons касиетинде көрсөтүлгөн эле.

VisibleButtons – кнопкаларды көрсөтүү же көрсөтпөнү ишке ашырат. Кнопкалардын константалары жана кызматтары ColoredButtons касиетинде көрсөтүлгөн эле.

StartPos, EndPos – ойноо башталуучу жана аяктоочу чекиттер. Түзүлүштүн тибине жараша кадрлардын же убакыттын интервалы көрсөтүлөт.

Frames – кадрлар боюнча алдыга же артка жылдырууда жылышуунун кадрлар боюнча саны.

Position – файлдын учурда ойнолуп жаткан позициясы, кадр же убакыт бирдигинде аныкталат.

2.7. Win 3.1 барагынын компоненттери

Window 3.1 версиясында колдонулган өзгөчө интерфейстик элементтердин компоненттери жайгашкан барак. Компоненттердин көпчүлүгү биз карап чыккан компоненттерди кайталагандыктан, алардын ичинен бешөөнү гана карайбыз.

1. **TNoteBook** – блокнот компоненти, көп беттүү интерфейсти түзүүдө колдонулат. Программалык жол менен блокноттун зарыл болгон беттине ал беттин номери аркылуу өтүүгө болот. Эгер блокноттун бети колдонуучу тарабынан тандалса, ушул эле баракчадагы **TTabSet** же Win32 барагындагы TTabControl элементтери менен биргеликте пайдалануу ыңгайлуу.

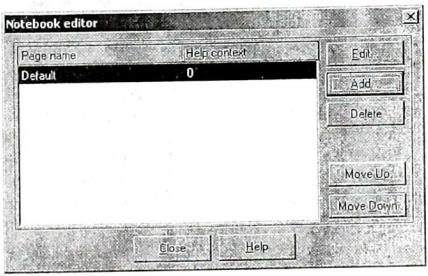
Pages – блокноттун беттерин башкаруучу, объекттик типтеги касиет. Оң жактагы кнопканы басып, пайда болгон терезенин жардамында, блокнотко жаңы бет кошуу, бетти өчүрүү жана беттин атын кайра ондоону ишке ашырууга болот.

PageIndex – блокноттун бетинине номери аркылуу өтүү, тиби бүтүн сан. Проектирлөө учурунда блокноттун контексттик менюсунун Next Page жана Previous опцияларынын жардамында да кийинки жана мурдагы беттерге өтүүгө болот.

ActivePage – аткарган кызматы боюнча PageIndex касиетине окшош, айырмасы беттин аты боюнча өтүү аткарылат.

OnPageChanged – окуясы блокноттун бети алмашкандан кийин пайда болот.

Мисал катары формага, анын өлчөмүн толук ээлебей тургандай кылып, TNoteBook компонентин жайгаштырабыз. Pages касиетинин оң жагындагы кнопканын жардамында беттерди башкаруу терезесин ачабыз. Терезенин көрүнүшү сүрөттө берилди.



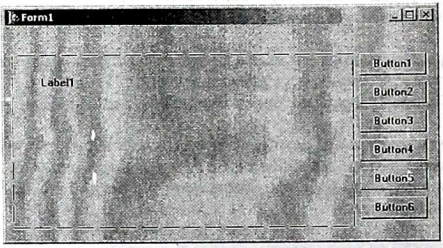
Мында Page name – беттин аты жана Help context – жардам файлындагы бетке туура келүүчү контексттин номери орнотулуучу параметрлери, ошондой

эле Edit – беттин жогоруда айтылган параметрлерин ондоо, Add – жаңы бетти кошуу, Delete – учурдагы бетти өчүрүү, Move Up – бетти жогору жылдыруу жана Move Down – бетти төмөн жылдыруу менен тартибин алмаштыруучу кнопкалар бар.

Add кнопкасынын жардамында Page1, Page2 жана Page3 аттары менен жаңы беттерди кошобуз. Аларды Help context параметрин нөл боюнча калтырабыз.

PageIndex касиетине нөл санын кийирип NoteBook1 компонентинин биринчи бетине өтөбүз жана бул бетке Standard барагындагы TLabel компонентин жайгаштырабыз. NoteBook1 компонентине оң щелчок жасап, контексттик менюнун Next Page опциясынын жардамында кийинки бетке өтөбүз. Бул бетке TEdit компонентин жайгаштырабыз. Ушундай эле жол менен блокноттун үчүнчү бетине TCombobox жана төртүнчү бетине TMemo компоненттерин жайгаштырабыз. Мында биз беттерди бири – биринен айырмалоо үчүн аларга түрдүү компоненттерди жайгаштырдык.

Формага TButton компоненттеринен алтоону сүрөттө көрсөтүлгөндөй кылып жайгаштырабыз.



TButton компоненттеринин Caption касиеттерин таблицада берилгендей кылып ондойбуз.

Компоненттин аты	Caption касиети
Button1	>>
Button2	<<
Button3	Default
Button4	Page1
Button5	Page2
Button6	Page3

Эми программалык коддорду жазабыз.

Алгач var бөлүгүнө ***i:integer;*** сабын жазабыз.

Форманын OnCreate окуясына:

```
NoteBook1.PageIndex:=0;
```

Button1 компонентинин OnClick окуясына:

```
if i<3 then
begin
inc(i);
Notebook1.PageIndex:=i;
Button2.Enabled:=true;
end
else
begin
Button1.Enabled:=false;
```

Button2 компонентинин OnClick окуясына:

```
if i>0 then
begin
dec(i);
Notebook1.PageIndex:=i;
Button1.Enabled:=true;
end
else Button2.Enabled:=false;
```

Button3 компонентинин OnClick окуясына:

```
Notebook1.ActivePage:='default';
```

Button4 компонентинин OnClick окуясына:

```
Notebook1.ActivePage:='Page1';
```

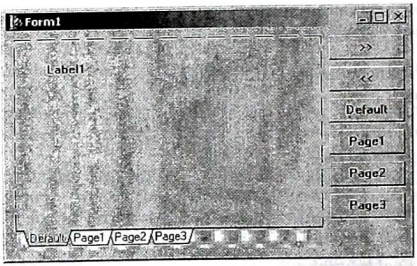
Button5 компонентинин OnClick окуясына:

```
Notebook1.ActivePage:='Page2';
```

Button6 компонентинин OnClick окуясына:

```
Notebook1.ActivePage:='Page3';
```

Окуяларга аракеттерди жазып болгон соң программаны иштешин текшерип көрөбүз. Эми TNoteBook компонентин TTabSet компоненти менен биргеликте пайдалануу карайбыз. TTabSet компонентинин касиеттери жана методдору Win32 барагында жайгашкан TTabControl компонентинин касиеттери жана методдоруна окшош. Ушул себептүү TTabSet компонентине өзүнчө токтолбойбуз.



Жогорудагы формага TTabSet компонентин TNoteBook компонентинин төмөн жагына жайгаштырып, анын Tabs касиетине Default, Page1, Page2 жана Page3 саптарын кийиребиз. Анын OnClick окуясына ***Notebook1.PageIndex:=TabSet1.TabIndex;*** сабын жазабыз. Программаны жүктөп, текшерип көргөнүбүздө TabSet1 компонентинин закладкалары менен кнопкалардын аракеттери бири – бири менен бир ыргакта иштебей жатканын байкайбыз. Бул кемчиликти ондоо үчүн бардык кнопкалардын программалык кодуна ***TabSet1.TabIndex:=Notebook1.PageIndex;*** сабын кошуп жазабыз.

Мисалдагы TTabSet жана TNoteBook компоненттеринин ордуна ушул эле баракта жайгашкан TTabbedNoteBook компонентин пайдалануура болот. Анткени TTabbedNoteBook компоненти TTabSet жана TNoteBook компоненттеринин жардамында түзүлгөн. Анын касиеттери жана методдору TTabSet жана TNoteBook компоненттеринин касиеттери жана методдору менен дал келгендиктен, бул компонентке да өзүнчө токтолбойбуз.

2. **TDirectoryListBox** – папкалардын тизмеси. Эки касиети менен гана айырмаланат.

DirLabel – тандалган папканын толук жолун көрсөтүүнү ишке ашырат. TLabel компоненти менен байланышат.

FileList – папкага тийиштүү файлдардын тизмеси, TFileListBox компонентти менен байланышат.

3. **TFileListBox** – файлдардын тизмеси. Папкадагы файлдардын тизмеси көрсөтүлөт. Негизги касиеттери TListBox компонентинин касиеттерине окшош. Өзгөчө касиеттери:

FileEdit – файлдын атын кийирүү тилкеси, TEdit компоненти менен байланышат. Тизмеден тандалган файлдын аты тилкеде пайда болот.

FileType – тизмеде көрсөтүлүүчү файлдардын тиби тандалат. Логикалык типтеги талаалар, көрсөтүлүүчү типтердин туурасына true маанисин коюу керек.

Аталышы	Түшүндүрмө
ftReadOnly	Окуу учун аттрибуту бар файлдар
ftHidden	Бекитилген аттрибуту бар файлдар
ftSystem	Системалык аттрибуту бар файлдар
ftArchive	Архивдик аттрибуту бар файлдар
ftVolumeID	Дисктин аты
ftDirectory	Папкалар
ftNormal	Аттрибуту жок файлдар

Mask – тизмеде көрсөтүлүүчү файлдардын маскасы. Маскалар TFilterComboBox компонентинде көрсөтүлөт.

ShowGlyphs – логикалык типтеги талаа, true болгон учурда, файлдардын жанында алардын тибине жараша пиктограммалык сүрөттөр көрсөтүлөт.

IntegralHeight – логикалык типтеги талаа, true болгон учурда, тизменин өлчөмү бүтүн элементтер гана көрсөтүлө тургандай болуп өзгөрөт.

4. **TDriveComboBox** – дисктердин комбинацияланган тизмеси. Негизги касиеттери TComboBox компонентинин касиеттериндей. Өзгөчөлөнгөн **DirList** касиети дисктеги папкаларды көрсөтүүчү TDirectoryListBox компонентти менен байланышат.

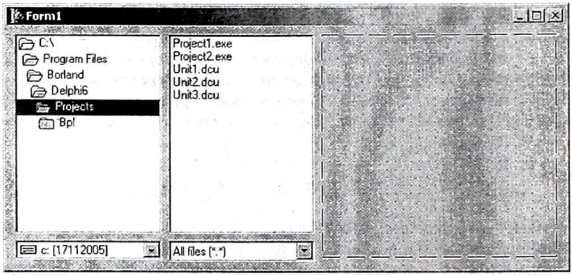
5. **TFilterComboBox** – файлдарды көрсөтүүдө пайдаланылуучу фильтрлердин комбинацияланган тизмеси.

Filter – пайдаланылуучу фильтрлердин тизмеси, талаанын оң жактагы кнопкасынын басып, атайын терезеде фильтрлерди жазууга болот.

FileList – фильтрди пайдаланылуучу файлдардын тизмеси, TFileListBox компонентти менен байланышат.

Мисал катары графикалык файлдардагы сүрөттөрдү көрүүгө арналган программаны түзүүнү карайбыз. Алгач формага TDirectoryListBox, TDriveComboBox, TFilterComboBox, TFileListBox жана TImage компоненттерин сүрөттө көрсөтүлгөндөй кылып жайгаштырабыз.

DriveComboBox1 компонентинин DirList касиетинеин тизмесинен DirectoryListBox1 компонентини тандайбыз. FilterComboBox1 жана DirectoryListBox1 компонентинин FileList касиетине FileListBox1 компонентин көрсөтөбүз.



FilterComboBox1 компонентинин Filter касиетинин оң жагындагы кнопкасынын жардамында филтрлердин терезесин ачабыз. Ал терезеге төмөнкү саптарды жазабыз.

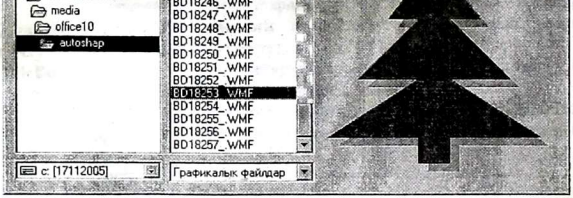
Графикалык файлдар	*.bmp;*.wmf;*.ico;
Bmp файлдар	*.bmp
WMF файлдар	*.wmf
Ico файлдар	*.ico

Image1 компонентинин Stretch касиетине true маанисин орнотобуз.

FileListBox1 компонентинин OnClick окуясына төмөнкү бир сапты жазабыз.

Image1.Picture.LoadFromFile(FileListBox1.FileName);

Программаны жүктөп, графикалык файлдар сакталган папкага өткөнүбүздө FileListBox1 компонентинде файлдардын тизмеси пайда болот. Бул тизмеден файл тандалганда Image компонентинде сүрөт көрсөтүлөт.



2.8. Dialogs барагының компоненттери

Dialogs барагында Windows ОС-нын бир нече стандарттуу диалогдору жайгашкан. Программадан бул диалогдорго кайрылганда ар бир диалог өзүнө мүнөздүү болгон терезелерди пайда кылат. Диалогдорго кайрылуу Execute методу менен ишке ашырылат жана бул метод логикалык типтеги маанини кайтарат.

1. **TOpenDialog** – файлдарды ачуунун диалогу.

Касиеттери:

DefaultExt – айтылбаган учурдагы пайдаланылуучу файлдардын кеңейтилиши, тиби string.

FilterIndex – учурда орнотулган филтрдин номери, бүтүн сан. Номерлөө бирден башталат.

InitialDir – диалогго биринчи жолу кайрылганда мазмуну көрсөтүлүүчү папка, тиби string.

Options – опциялардын тизмеси, логикалык типтеги талаалар. Файлдарды тандоо терезесинин иштөө режимин орнотуу үчүн пайдаланылат.

Title – диалогдук терезенин бөркүндөгү текст.

Окуялары:

OnCanClose – колдонуучу терезени жабууга аракет жасады. Бул окуяда колдонуучу файлды туура тандагандыгына же файлдын атын туура киргизгендигине жараша терезени жабууга уруксат берүү же бербей коюуну, **CanClose** логикалык типтеги параметри аркылуу ишке ашырууга болот.

OnFolderChange – колдонуучу башка папкага өткөндө пайда болот.

OnSelectionChange – колдонуучу башка файлды тандаганда пайда болот.

OnTypeChange – колдонуучу файлдардын жаңы маскасын тандаганда пайда болот.

2. **TSaveDialog** – файлды сактоонун диалогу. Касиеттери TOpenDialog компонентинин касиеттеринен дээрлик айырмаланбайт.

3. **TOpenPictureDialog** жана **TSavePictureDialog** компоненттери TOpenDialog жана TSaveDialog компоненттеринен түзүлүп, алардын касиеттери жана методдору окшош. Бул компоненттер сүрөттөрдү ачуу жана сактоо үчүн колдонулат. Диалогдук терезелер файлдардагы сүрөттөрдү көрүүгө арналган кошумча областка ээ.

4. **TFontDialog** – шрифттин диалогу.

Касиеттери:

Device – түзүлүшкө тийиштүү шрифтердин тизмеси тандалат. Мааниси fdScreen – экранга, fdPrinter – принтерге же fdBoth – экөөнө тең тийиштүү боло атат.

Font – колдонуучу тандаган шрифт, тиби TFont.

MaxFontSize жана **MinFontSize** – шрифттердин өлчөмдөрүнүн тизмесинен тандоо үчүн чектөөнүн масималдык жана минималдык манаанилери.

Options – терезенин көрүнүшүнүн кошумча мүнөздөмөлөрү.

5. **TColorDialog** – түстөрдү тандоонун стандарттуу диалогу.

Касиеттери:

Color – колдонуучу тарабынан тандалган түс.

CustomColor – кошумча түстөрдүн тизмеси. Тизмеге түстөр RGB стандарты (түстүн түзүүчүлөрү болгон R – кызыл, G – жашыл жана B – көк түстөрдүн маанилери) боюнча киргизилет жана ар бир байт он алтылык эсептөө системасында берилген эки символдон турат. Мисалы ABFF08 түрүндөгү жазуу R байты AB, G байты FF жана B байты 08 маанисине барабар дегенди түшүндүрөт.

3. Мисалдар

Компоненттер менен таанышканыбызда алардын көпчүлүк касиеттери жана методдоруна токтолгон жокпуз. Ушул себептүү, мисалдарда колдонулуп жаткан методдорго кеңири токтолобуз.

Көрсөтүлгөн мисалдарды аткаруунун алдында, дискке папка түзгүлө, ар бир мисалдын файлдарын ушул папканын ичине жаңы папканы түзүп, андан кийин сактагыла. Учурда C: дискине Misal папкасы түзүлгөн деп эсептейбиз. Кыргызча шрифттер Delphi чөйрөсүндө туура эмес көрсөтүлгөндүктөн, мисалдарда элементтердин бөрктөрүн негизинен орусча сөздөр менен жазабыз.

Мисалдардын аткарылышы түшүнүктүү болсун үчүн аларды бир нече этаптарга бөлүп жасайбыз. Мисалдарды аткарып жатканда, программалык коддо кезигүүчү компоненттердин аттарын, өзүнөр пайдаланып жаткан компоненттердин аттары менен алмаштырып жазууну эсиңерге салабыз.

3.1. Геометрия

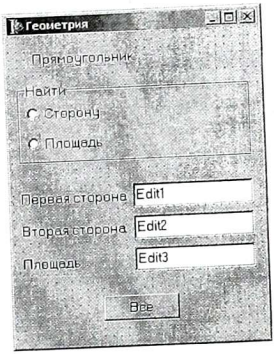
Программа тик бурчтуктун жагын же аянтын эсептейт.

I этап. Формага компоненттерди жайгаштыруу

Формага төрт TLabel, үч TEdit жана бирден TRadioGroup, TButton (Standard) компоненттерин жайгаштырабыз жана алардын касиеттерин таблицаларда көрсөтүлгөндөй кылып өзгөртөбүз (кашаалардын ичинде компоненттер жайгашкан баракчалар көрсөтүлдү).

Компоненттин аты	Caption касиети
Label1	Прямоугольник
Label2	Первая сторона
Label3	Вторая сторона
Label4	Площадь
RadioGroup1	Найти
Button1	Все

Edit2 компонентинин ReadOnly касиетине true маанисин орнотобуз. RadioGroup1 компонентинин Item касиетинин оң жагындагы кнопкасын басып, пайда болгон терезеге Сторону жана Площадь сөздөрүн кийирип, терезени жабабыз. ItemIndex касиетине 0 маанисин жазабыз. Формага компоненттерди сүрөттө көрсөтүлгөндөй кылып жайгаштырабыз.



II этап. Программалык коддорду жазуу

Формага щелчок жасайбыз жана объекттердин инспектору терезесинин Event бөлүгүн ачабыз. OnCreate окуясынын оң жагындагы бош сапка эки щелчок жасайбыз. Пайда болгон терезедеги Begin жана End сөздөрүнүн арасына төмөнкү саптарды жазабыз.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
Edit1.Text:='0';    //Баштапкы маанилерди орнотобуз
Edit2.Text:='0';
Edit3.Text:='0';
end;
```

Жөнөкөй курсив шрифттер менен Delphi чөйрөсү тарабынан жазылган программалык коддор, кара курсив шрифттер менен биз жазган программалык коддор көрсөтүлөрүн дагы бир жолу эске салабыз. // белгилеринен кийин түшүндүрмө жазылат, аларды кунт коюп окугула.

RadioGroup1 компонентинин OnClick окуясынын программалык кодун төмөнкүдөй жазабыз.

```
procedure TForm1.RadioGroup1Click(Sender: TObject);
begin
case RadioGroup1.ItemIndex of  //кайсы радио кнопка тандалганын аныктайбыз
0: begin                  //биринчи радио кнопка тандалса
  Edit2.Text:='0';
  Edit2.ReadOnly:=true; //Edit2-нин маанисин өзгөртүүгө тыюу салабыз
  Edit3.ReadOnly:=false; //Edit3-түн маанисин өзгөртүүгө уруксат беребиз
  end;
1: begin                  // экинчи радио кнопка тандалса
  Edit3.Text:='0';          // жогоркунун тескериси жасалат
  Edit3.ReadOnly:=true;
  Edit2.ReadOnly:=false;
  end;
end;
```

Button1 компонентинин OnClick окуясында колдонуучу тарабынан радио кнопкалардын тандалышына жараша эсептөө жүргүзөбүз.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
case RadioGroup1.ItemIndex of
0: if (Edit1.Text<>'0') or (Edit1.Text<>'') then
Edit2.Text:=FloatToStr(StrToFloat(Edit3.Text) / StrToFloat(Edit1.Text));
1: Edit3.Text:=FloatToStr(StrToFloat(Edit1.Text) * StrToFloat(Edit2.Text));
end;
end;
```

Биз бул коддо Edit1 компонентинин Text касиетинин мааниси нөлгө барабар же бош болуп калган абалын гана текшердик. Ошондуктан TEdit компоненттеринин Text касиеттерине сөзсүз сан жазуу зарыл, ал эми бөлчөк санды жазуу үчүн үтүр белгини пайдалануу керек. Башка учурларда программанын иштешинде ката пайда болот.

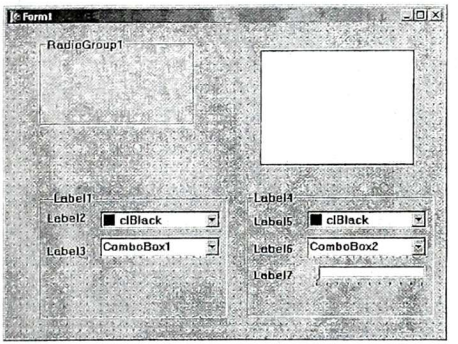
Мына ушундай эле жол менен башка фигуралардын аянтын жана көлөмүн эсептөөчү программаларды жазууга болот жана аларды өз алдыңарча түзүп чыгуу сунушталат.

3.2. Фигура

Бул программада TShape компонентинин Brush жана Pen касиеттерин өзгөртүүнү карайбыз. Бул касиеттер объекттик касиеттер болуп, TBrush жана TPen объекттери менен байланышкан.

I этап. Формага компоненттерди жайгаштыруу жана алардын касиеттерин орнотуу

Формага жети TLabel, эки TComboBox, бир TRadioGroup (Standard), эки TBevel, TColorBox, бир TShape (Additional) жана бир TTrackBar (Win32) компоненттерин сүрөттө көрсөтүлгөндей кылып жайгаштырабыз жана касиеттерин орнотобуз.



Компоненттин аты	Caption касиети
Label1	Область фигуры
Label2	Цвет
Label3	Стиль
Label4	Линия фигуры
Label5	Цвет
Label6	Стиль
Label7	Толщина
RadioGroup1	Фигура
Form1	Фигуры

RadioGroup1 компонентинин Items касиетине stRectangle, stRoundRect, stRoundSquare, stSquare, stCircle жана stEllipse саптарын кийирип, ItemIndex касиетине 0 санын жазабыз.

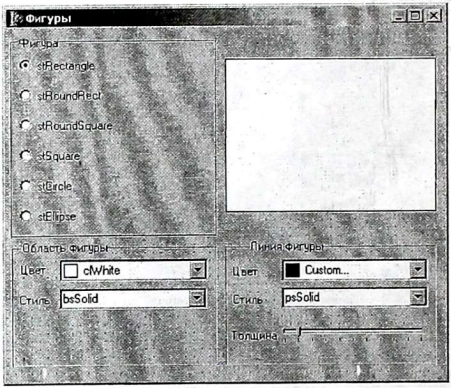
ColorBox1 компонентинин Selected касиетинен clWhile маанисин тандайбыз. ColorBox1 жана ColorBox2 компоненттеринин Style касиетинин cbCustomColor талаасына true маанисин орнотобуз.

ComboBox1 компонентинин Items касиетине bsSolid, bsClear, bsHorizontal, bsVertical, bsFDiagonal, bsBDiagonal, bsCross жана bsDiagCross саптарын кийирип, ItemIndex касиетине 0 санын жазабыз. Ошондой эле, ComboBox1

компонентинин Items касиетине psSolid, psDash, psDot, psDashDot, psDashDotDot жана psClear саптарын кийирип, ItemIndex касиетине 0 санын жазабыз.

TrackBar1 компонентинин Position касиетин 1–ге жана ThumbLength касиетин 10 – го ондойбуз.

Бардык касиеттерди орнотуп болгон соң форма төмөнкү көрүнүшкө келет.



II этап. Программалык коддорду жазуу

RadioGruop1 элементинен колдонуучу фигуранын атын тандаганда Shape1 компонентасынын Shape касиети да тандалган фигурага жараша өзгөрүүгө тийиш. RadioGruop1 компонентинин OnClick окуясынын программалык коду:

```
procedure TForm1.RadioGroup1Click(Sender: TObject);
begin
case RadioGroup1.ItemIndex of
0: Shape1.Shape:=stRectangle;
1:Shape1.Shape:=stRoundRect;
2:Shape1.Shape:=stRoundSquare;
3:Shape1.Shape:=stSquare;
4:Shape1.Shape:=stCircle;
5:Shape1.Shape:=stEllipse;
end;
end;
```

ColorBox1 компонентинен колдонуучу түстү тандаганда, фигуранын ички областынын түсүн өзгөртүүбүз керек. Ал үчүн Shape1 компонентине тийиштүү Brush объекттик касиетинин Color талаасына ColorBox1 компонентинин Selected талаасындагы маанини ыйгарабыз. ColorBox1 компонентинин OnChange окуясынын программалык коду:

```
procedure TForm1.ColorBox1Change(Sender: TObject);
begin
Shape1.Brush.Color:=ColorBox1.Selected;
end;
```

ComboBox1 компонентинен тандалган сапка жараша, Shape1 компонентине тийиштүү Brush объекттик касиетинин Style талаасына тиешелүү маанини ыйгарабыз. ComboBox1 компонентинин OnChange окуясынын программалык коду:

```
procedure TForm1.ComboBox1OnChange (Sender: TObject);
begin
Case ComboBox1.ItemIndex of
0:Shape1.Brush.Style:=bsSolid;
1:Shape1.Brush.Style:=bsClear;
2:Shape1.Brush.Style:=bsHorizontal;
3:Shape1.Brush.Style:=bsVertical;
4:Shape1.Brush.Style:=bsFDiagonal;
5:Shape1.Brush.Style:=bsBDiagonal;
6:Shape1.Brush.Style:=bsCross;
7:Shape1.Brush.Style:=bsDiagCross;
end;
end;
```

ColorBox2 компонентинин OnChange окуясы ColorBox1 компонентинин окуясына окшош, айырмасы Pen объекттик касиетинин Color талаасын өзгөртөт. ColorBox2 компонентинин OnChange окуясынын программалык коду:

```
procedure TForm1.ColorBox2Change(Sender: TObject);
begin
Shape1.Pen.Color:=ColorBox2.Selected;
end;
```

ComboBox2 компонентинин OnChange окуясынын программалык коду тандалган сапка жараша Shape1 компонентине тийиштүү Pen объекттик касиетинин Style талаасынын маанисин өзгөртөт.

```
procedure TForm1.ComboBox2Change (Sender: TObject);
begin
```

```
    case ComboBox2.ItemIndex of
0:Shape1.Pen.Style:=psSolid;
1:Shape1.Pen.Style:=psDash;
2:Shape1.Pen.Style:=psDot;
3:Shape1.Pen.Style:=psDashDot;
4:Shape1.Pen.Style:=psDashDotDot;
5:Shape1.Pen.Style:=psClear;
end;
end;
```

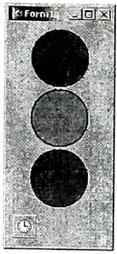
TrackBar1 компонентинин жардамында фигуранын контурунун жоондугун башкарабыз. TrackBar1 компонентинин OnChange окуясынын программалык коду:

```
procedure TForm1.TrackBar1Change(Sender: TObject);
begin
Shape1.Pen.Width:=TrackBar1.Position;
end;
```

3.3. Светофор

Бул мисалда TTimer компонентин пайдалануу үйрөнөбүз.

I этап. Формага компоненттерди жайгаштыруу жана алардын касиеттерин орнотуу



Формага үч TShape (Additional) жана бир TTimer (System) компоненттерин жайгаштырабыз. TShape компоненттеринин Shape касиеттерине stCircle маанисин жана Brush касиеттеринин Color талааларына көрсөтүлгөн маанилерди орнотобуз.

Компоненттин аты	Color талаасынын мааниси
Shape1	clRed
Shape2	clOlive
Shape3	clGreen

Timer1 элементинин Interval касиетине 2000 маанисин кийиребиз жана элементтерди сүрөттө көрсөтүлгөндөй кылып жайгаштырабыз.

II этап. Программалык кодду жазуу.

Убакыттын белгилүү аралыктарында светофордун чырактарынын түсү автоматтык түрдө алмашып туруусун камсыздоо үчүн Timer1 компонентин пайдаланабыз. Чындыгында светофордун сары түсү анын кызыл жана жашыл түстөрүнөн бир аз убакытка кыска жанат. Ошондуктан программада түстөр

өзгөргөндө убакыттын интервалын да өзгөртөбүз. Программалык кодду Timer1 компонентинин OnTimer окуясына жазабыз.

Глобалдык чоңдуктарды жарыялоо бөлүгүнө integer тибиндеги lamp чоңдугун баштапкы мааниси 1ге барабар деп жарыялайбыз. Бул чоңдук светофордун учурдагы түсүн аныктоодо колдонулат.

```
var
  Form1: TForm1;
  Lamp:integer=1;
```

Эми процедуранын кодун кийрийбиз.

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
if Lamp=1 then // түсү кызыл болсо
begin
Shape1.Brush.Color:=clMaroon; // Кочкул кызыл түскө алмаштырабыз.
Shape2.Brush.Color:=clYellow; // Сары түскө алмаштырабыз.
Lamp:=2;                      // Учурда кызылдан кийинки сары түс деп белгилейбиз.
Timer1.Interval:=1000; // Убакыттын интервалын кыскартабыз.
end
else if Lamp=2 then // Кызыл түстөн кийинки сары түс болсо,
begin
Shape2.Brush.Color:=clOlive; // ачык күрөң  түскө алмаштырабыз.
Shape3.Brush.Color:=clLime; // Ачык жашыл түскө алмаштырабыз.
Lamp:=3;                          // Учурда жашыл түс деп белгилейбиз.
Timer1.Interval:=2000;    // Убакыттын интервалын чоңойтобуз.
end
else if Lamp=3 then // Түсү жашыл болсо,
begin
Shape2.Brush.Color:=clYellow; // сары түскө алмаштырабыз.
Shape3.Brush.Color:=clGreen; // Жашыл түскө алмаштырабыз.
Lamp:=4;       // Учурда жашылдан кийинки сары түс деп белгилейбиз.
Timer1.Interval:=1000; // Убакыттын интервалын кыскартабыз.
end
else
begin      // Жашылдан кийинки сары түс болсо,
Shape1.Brush.Color:=clRed;          // кызыл түскө алмаштырабыз.
Shape2.Brush.Color:=clOlive;        // Ачык күрөң түскө алмаштырабыз.
 Lamp:=1;        // Учурда кызыл түс деп белгилейбиз.
Timer1.Interval:=2000;  // Убакыттын интервалын чоңойтобуз.
end;
end;
```

3.4. Калькулятор

Бул мисалда стандарттуу көрүнүштөгү калькуляторду түзөбүз.

I этап. Негизги терезени түзүү

Алгач формага бир TMemo (Standard), TStaticText (Additional) жана 24 TSpeedButton (Win32) компоненттерин сүрөттө көрсөтүлгөндөй кылып жайгаштырабыз.

Таблицада кнопка объектилеринин аттары жана алардын Capion касиеттери көрсөтүлгөн. Мисалды аткарып жатканда, программалык коддо кезигүүчү объекттердин аттарын тиешелүү түрдө алмаштырып жазгыла. Мисалы, көрсөтүлүп жаткан мисалда 0 саны жазылган кнопканын объекттисинин аты SpeedButton23, ага кайрылуу үчүн SpeedButton23 деп жазылган. Ал эми силер түзүп жаткан формада бул кнопканын объектисинин аты SpeedButton11 болсо, анда көрсөтүлүп жаткан мисалдагы SpeedButton23 сөзүнүн ордуна SpeedButton11 деп жазгыла.

Name	Caption
SpeedButton1	←
SpeedButton2	CE
SpeedButton3	C
SpeedButton4	MC
SpeedButton5	7
SpeedButton6	8
SpeedButton7	9
SpeedButton8	/
SpeedButton9	sqrt
SpeedButton10	MR
SpeedButton11	4
SpeedButton12	5
SpeedButton13	6
SpeedButton14	*

Name	Caption
SpeedButton14	%
SpeedButton16	MS
SpeedButton17	1
SpeedButton18	2
SpeedButton19	3
SpeedButton20	-
SpeedButton21	1/x
SpeedButton22	M+
SpeedButton23	0
SpeedButton24	+/-
SpeedButton25	,
SpeedButton26	+
SpeedButton27	=

Формадагы компоненттердин бирдей касиеттерин бир учурда өзгөртүү үчүн, алардын Shift клавишасын басып, кое бербестен мыштын жардамында белгилейбиз.

Форманын Font касиетинин Size талаасына 10, Color талаасына clBlue жана анын Style касиетиндеги fsBold талаасына true маанилерин орнотобуз. Тиешелүү кнопкаларды Shift клавишасынын жардамында белгилеп, алардын Font касиетинин Color талаасынын маанисин clRed, ал Memo1 жана StaticText1 компоненттеринин ушул эле касиеттерин clBlack маанисине өзгөртөбүз. Ар бир кнопканын Caption касиетине тийиштүү маанилерди жазып чыгабыз жана Memo1 компонентинин Lines касиетинин оң жагындагы кнопкасын басып, пайда болгон терезедеги саптын ордуна нөл санын кийиребиз. StaticText1 компонентинин Caption касиетиндеги текстти өчүрүп салабыз. Кнопкалардын жана StaticText1 компонентинин бийиктиги (Height) жана узундугу (Width) 30, чоң кнопкалардын узундугу 50гө барабар.

Форманын, Memo1 компонентинин башка касиеттери:

Форма	
Касиеттин аты	Мааниси
Constraints	
MaxHeigh	250
MaxWidth	300
MinHeight	250
MinWidth	300
Caption	Калькулятор

Memo1	
Касиеттин аты	Мааниси
Alignment	taRightJustify
MaxLength	32
Heigh	25
Width	230
WantReturns	False
ReadOnly	True

StaticText1 компонентинин башка касиеттери:

Касиеттин аты	Мааниси
BevelKind	bkFlat
BorderStyle	sbsSunken
AutoSize	false

Бул мисалда TMemo компонентинин ордуна TEdit компонентин пайдалансак деле болмок, бирок TEdit компонентинин сапты тегиздөө (Alignment) касиети жок болгондуктан, андагы текст дайыма сол жагына тегизделет.

II этап. Калькулятордун программалык кодун жазуу

Кнопкалардын аракеттерин жазуу үчүн бир нече жаңы өзгөрүлмөлөрдү жарыялап алуу зарыл. Ал үчүн глобалдык чоңдуктарды жарыялоо бөлүгүнө төмөнкү чоңдуктарды жарыялайбыз.

```
var
  Form1: TForm1;
  mark:integer;
```

```
    us:boolean;
mem, firts:Extended;
```

Мында, mem эс менен иштөөчү MS, M+, MR жана MC кнопкаларынын аракетинде пайдаланылат. First чондугу аралык маанилерди сактоо үчүн, us кийирүү сабына жаңы маани кийириле баштаганын көрсөтүү үчүн, mark чондугу /, *, -, + жана = белгилери коюлган кнопкалардын кайсы бири басылгандыгын эстеп калуу үчүн колдонобуз.

1..9 кнопкаларынын аракеттери бирдей, кнопка басылганда биринчи сан нөл болбосо, тиешелүү цифраны сандын аягына кошуп жазууну ишке ашырышат, антпесе нөл сан менен алмаштырылат (түшүндүрүү жеңил болсун үчүн кнопкаларды белгилениши боюнча айтабыз). 1 саны жазылган кнопканын OnClick окуясына ушул аракетти жазабыз.

```
procedure TForm1.SpeedButton17Click(Sender: TObject);
begin
if us then Memo1.Lines[0]:='0';
if Length(Memo1.Lines[0])<32 then
begin
if Memo1.Lines[0]='0' then Memo1.Text:=SpeedButton17.Caption
else Memo1.Text:=Memo1.Text+SpeedButton17.Caption;
end;
end;
```

Мында Length(memo1.Lines[0])<32 киргизилген саптын узундугун текшерет, б.а. калькуляторго колдонуучу 32 разряддуу санды кийире алат.

2 саны жазылган кнопканын кодун жазабыз.

```
procedure TForm1.SpeedButton18Click(Sender: TObject);
begin
if us then Memo1.Lines[0]:='0';
if Length(Memo1.Lines[0])<32 then
begin
if Memo1.Lines[0]='0' then Memo1.Text:=SpeedButton18.Caption
else Memo1.Text:=Memo1.Text+SpeedButton18.Caption;
end;
end;
```

Бул кодду 1 саны жазылган кнопканын коду менен салыштырсак, SpeedButton17 компоненти SpeedButton18 компоненти менен алмаштырылып жазылганын байкайбыз. Тагыраак айтканда, окуяны кайсы объект пайда кылып жатса ошол объекттин Caption касиетин пайдаланып жатабыз. Ушундай эле ой жүгүртүү менен калган цифралык кнопкалардын программалык коддору да дээрлик бирдей экендигин түшүнүүгө болот. Бул көрүнүш бизди экинчи жагынан кызыктырат. Учурда пайда болгон абалда аракеттерге жазылуучу

программалык коддордун кайталануусун кантип кыскартууга болот деген суроо туулат. Процедуранын параметриндеги Sender кайсы компонент тарабынан окуя пайда болгонун көрсөтүүчү чондук экендигин айтканбыз. Биз ушул чондукту пайдаланып 1 саны жазылган кнопканын программалык кодун өзгөртүп жазабыз.

```
procedure TForm1.SpeedButton17Click(Sender: TObject);
begin
if us then Memo1.Lines[0]:='0';
if Length(Memo1.Lines[0])<32 then
begin
if Memo1.Lines[0]='0' then
Memo1.Text:= TspeedButton(Sender).Caption
else Memo1.Text:=Memo1.Text+TSpeedButton(Sender).Caption;
end;
end;
```

SpeedButton17 сабыны TSpeedButton(Sender) сабы менен алмаштырдык. Окуяны пайда кылган TSpeedButton тибиндеги объектти көрсөтүп жатабыз дегенди түшүндүрөт.

Калган цифра жазылган кнопкалардын OnClick окуяларынан ушул кнопканын окуясына шилтеме жасайбыз.

sqrt кнопкасынын программалык кодунда Паскаль тилиндегидей эле sqrt – сандын квадраттык тамырын кайтаруучу функцияны пайдаланабыз. Бул функциянын аргументи болгон сандын нөлдөн чоң болушу зарыл.

```
procedure TForm1.SpeedButton9Click(Sender: TObject);
begin
if StrToFloat(Memo1.Lines[0])>=0 then Memo1.Text:=FloatToStr
(sqrt(StrToFloat(Memo1.Lines[0]))) else MessageDlg('Отрицательное
число под корнем!',mtError, [mbYes], 0);
end;
```

Мында StrToFloat жана FloatToStr функцияларын колдондук. Биринчиси string тибиндеги чондукту Real тибине которот, экинчиси анын тескерисин жасайт.

1/x кнопкасынын кодуна

```
if StrToFloat(Memo1.Lines[0])<>0 then Memo1.Text:=FloatToStr
(1/(StrToFloat(Memo1.Lines[0]))) else MessageDlg('Деление на
ноль!',mtError, [mbYes], 0);
```

+/- кнопкасынын кодуна

```
if StrToFloat(Memo1.Lines[0])<>0 then Memo1.Text:=FloatToStr
(-1*(StrToFloat(Memo1.Lines[0])));
```

Чыныгы сандарда үтүр белгиси бир гана жолу коюлат, ошондуктан санга үтүр белгиси колганын текшерүү керек. Үтүр белгиси жазылган кнопканын аракетинде санга үтүр белгиси коюлгандыгын белгилеп туруу үчүн ушул эле кнопканын tag касиетин пайдаланабыз. Санга үтүр белгиси коюлса бул касиеттин маанисин 1 деп белгилейбиз, антпесе 0 маанисин коебуз. Анын программалык кодун төмөнкүчө жазабыз.

```
procedure TForm1.SpeedButton25Click(Sender: TObject);
begin
If SpeedButton25.Tag=0 then
begin
Memo1.Text:=Memo1.lines[0]+',';
SpeedButton25.Tag:=1;
end;
end;
```

CE сабы жазылган кнопка колдонуучу учурда кийирип жаткан маанини тазалайт. Анын аркетине ***Memo1.Text:='0';*** сабын жазабыз.

/,*,-,+,= жана % белгилери жазылган кнопкалардын Tag касиетине тиешелеш түрдө 1, 2, 3, 4, 5 жана 6 маанилерин кийиребиз. + белгиси коюлган кнопканын OnClick окуясына көрсөтүлгөн программалык кодду жазабыз.

```
procedure TForm1.SpeedButton26Click(Sender: TObject);
begin
if mark = 0 then
begin
first:=StrToFloat(Memo1.Lines[0]);
mark:=TSpeedButton(Sender).Tag;
end
else
begin
case mark of
1: if StrToFloat(Memo1.Lines[0])<>0 then
first:=first/StrToFloat(Memo1.Lines[0]) else MessageDlg('Деление на
ноль!',mtError, [mbYes], 0);
2: first:=first*StrToFloat(Memo1.Lines[0]);
3: first:=first-StrToFloat(Memo1.Lines[0]);
4: first:=first+StrToFloat(Memo1.Lines[0]);
5: first:=StrToFloat(Memo1.Lines[0]);
6: first:=first/100*StrToFloat(Memo1.Lines[0]);
end;
mark:=TSpeedButton(Sender).Tag;
Memo1.Text:=FloatToStr(first);
end;
us:=true;
end;
```

Башка белгилер жазылган кнопкалардын OnClick окуяларына ушул кнопканын окуясын шилтеме түрүндө көрсөтөбүз.

С тамгасы жазылган кнопка чоңдуктарды баштапкы калыбына келтирет. Анын программалык коду:

```
procedure TForm1.SpeedButton3Click(Sender: TObject);
begin
Memo1.Text:='0';
us:=false;
first:=0;
mark:=0;
SpeedButton25.Tag:=0;
end;
```

← белгиси коюлган кнопка акыркы цифраны өчүрөт. Эң акыркы цифра өчүрүлгөндө Memo1 компонентинде нөл саны пайда болуусу зарыл. Бул кнопканын аракетинин программалык коду:

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
var
s:string;
i:integer;
begin
if Memo1.Lines[0]<>'0' then
if Length(Memo1.Lines[0])=1 then Memo1.Lines[0]:='0'
else
begin
s:=Memo1.Text;
i:=Length(Memo1.Lines[0]);
Delete(s,i,1);
Memo1.Text:=s;
end;
end;
```

MC сабы жазылган кнопка маанини убактылуу сактоочу эсти тазалайт. Анын коду:

```
procedure TForm1.SpeedButton4Click(Sender: TObject);
begin
mem:=0;
Statictext1.CleanupInstance;
us:=true;
end;
```

MR сабы жазылган кнопка эстеги маанини Memo1 компонентине чыгарат.

```
procedure TForm1.SpeedButton10Click(Sender: TObject);
begin
Memo1.Text:=FloatToStr(mem);
us:=true;
end;
```

MS сабы жазылган кнопка эске учурдагы маанини жазат.

```
procedure TForm1.SpeedButton16Click(Sender: TObject);
begin
mem:=strtofloat(Memo1.Lines[0]);
Statictext1.Caption:='M';
us:=true;
end;
```

M+ сабы жазылган кнопка учурдагы маанини эстеги мааниге кошот.

```
procedure TForm1.SpeedButton22Click(Sender: TObject);
begin
mem:=mem+strtofloat(Memo1.Lines[0]);
Statictext1.Caption:='M';
us:=true;
end;
```

Бул мисалдагы эң акыркы программалык код калькуляторду клавиатуранын жардамында башкарууну ишке ашырат. Ал үчүн басылган клавиша генерациялаган символду же анын номеринин аныктап, тиешелүү объекттин OnClick окуясын пайда кылуучу Click методуна кайрылабыз. Форманын OnKeyPress окуясына төмөнкү саптарды жазабыз.

```
procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
begin
case Key of
'0': speedbutton23.Click;
'1': speedbutton17.Click;
'2': speedbutton18.Click;
'3': speedbutton19.Click;
'4': speedbutton11.Click;
'5': speedbutton12.Click;
'6': speedbutton13.Click;
'7': speedbutton5.Click;
'8': speedbutton6.Click;
```

```
'9': speedbutton7.Click;
'/': speedbutton8.Click;
'*': speedbutton14.Click;
'-': speedbutton20.Click;
'+': speedbutton26.Click;
'=',#13: speedbutton27.Click; // = же Enter клавишасы басылды.
'%':speedbutton15.Click;
'x','X':speedbutton21.Click;
'q','Q':speedbutton9.Click;
#8:speedbutton1.Click; // Backspace клавишасы басылды.
#27:speedbutton2.Click; // Esc клавишасы басылды.
'm','M':speedbutton22.Click;
's','S':speedbutton16.Click;
'r','R':speedbutton10.Click;
'c','C':speedbutton4.Click;
'e','E':speedbutton3.Click;
',':speedbutton25.Click;
'`': speedbutton24.Click; // тескери апостроф (1 клавишасынын алдындагы) клавишасы басылды.
end;
end;
```

Бул коддо программалык жол менен башка объекттин окуясын пайда кылуучу методду колдонууну карадык. Объекттердин башка окуяларын да программада ушундай жол менен пайда кылабыз.

Memo1 компонентинин OnKeyPress окуясынан форманын OnKeyPress окуясына шилтеме жасайбыз.

3.5. Тексттик редактор

Формattалган текст менен иштөөчү редакторду түзөбүз.

I этап. Редактордун негизин түзүү

Формага TMainMenu, TPopupMenu (Standard), TToolBar, TImageList, TStatusBar, TRichEdit (Win32), TFontDialog, TOpenDialog жана TSaveDialog (Dialogs) компоненттерин жайгаштырабыз.

Алгач ImageList1 элементине сүрөттөрдүн коллекциясын түзөбүз, ал үчүн C:\Program Files\Common Files\Borland Shared\Images\Buttons папкасындагы сүрөттөрдөн тандап алууга болот. Сүрөттөрдүн коллекциясы түзүлгөндөн кийин, MainMenu1 компонентин тандап, анын Images тилкесинин оң жагындагы кнопкасыны басып, тизмеден ImageList1 сабын тандайбыз. Бул аракетти, касиетке компонентти байланыштыруу дейбиз.

MainMenu1 компонентине менюнун опцияларын кийиребиз жана ар бир опциянын ImageIndex касиетине ага туура келүүчү сүрөттү орнотобуз.

Таблицада опциялар менен бирге ShortCut касиетинин маанилери берилген.

Caption	Short Cut	Объект-тин аты
&Файл		N1
&Новый	Ctrl+N	N2
&Открыть	Ctrl+O	N3
&Сохранить	Ctrl+S	N4
Со&хранить как		N5
-		N6
В&ыход	Ctrl+Q	N7

Caption	Short Cut	Объект-тин аты
&Правка		N8
&Отменить	Ctrl+Z	N9
-		N14
&Вырезать	Ctrl+X	N10
&Копировать	Ctrl+C	N11
Вст&авить	Ctrl+V	N12
&Удалить	Ctrl+D	N13
В&ыделить все	Ctrl+A	N15

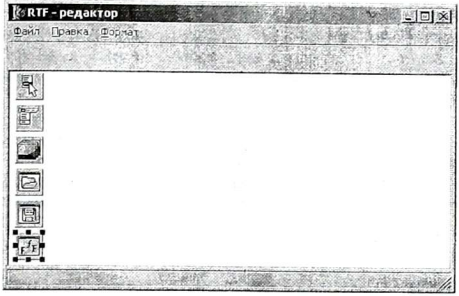
Формат менюсунун опциялары GroupIndex, RadioItem жана AutoCheck касиеттеринин маанилери менен берилди.

Caption	GroupIndex	RadioItem	AutoCheck	Объекттин аты
&Формат				N16
&Шрифт				N17
&Абзац				N18
&Список			true	N23
-				N19
По &левому краю	1	true	true	N20
По &правму краю	1	true	true	N21
По &центру	1	true	true	N22

Мындан сырткары «По левому краю» опциясынын Checked касиетине true маанисин орнотобуз. Опциялардын Name касиетине көңүл бургула, алар N1, N2, N3 … түрүндө жазылып жатат.

RichEdit1 компонентинин Align касиетине alClient маанисин орнотуп, Lines касиетинин он жагындагы кнопканын жардамында терезени ачып, андагы текстти өчүрүп салабыз.

Форманын проектирлөө учурундагы сүрөтү.



II этап. Терезенин жана менюнун окуяларына аракеттерди жазуу

Аракеттерди жазуудан мурда, OpenDialog1 жана SaveDialog1 компоненттерин касиеттерин орнотобуз.

Касиеттери	Маанилери	
DefaultExt	rtf	
FileName	Noname	
InitialDir	C:\	
Filter	Атайын терезеге кийрийбиз	
	Rich Text Format	*.rtf
	Текстовые файлы	*.txt
	Все	*.*

Форманын OnCreate окуясына:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
Form1.Caption:='RTF редактор-'+SaveDialog1.FileName;
End;
```

сабын жазабыз. Ошондой эле форманын OnCloseQuery окуясына төмөнкү аракеттерди жазабыз.

```
procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose:
Boolean);
begin
CanClose:=true;                  // Терезени жабууга уруксат берилет.
if RichEdit1.Modified then       // Эгер документте өзгөрүү болсо, экранга диалог
чыгарабыз.
```

```
case MessageDlg('Документ изменен, сохранить?' ,mtConfirmation,
[mbYes,mbNo,mbCancel],0) of   // Колдонуучунун жообун текшеребиз.
mrYes: begin                  // Колдонуучу «Yes» деп жооп берсе,
if SaveDialog1.FileName='Noname.rtf' then  // касиеттин мааниси өзгөрбөгөн
болсо, файлга сактоонун диалогдук терезесин ачабыз.
begin
if Savedialog1. Execute then RichEdit1.Lines.SaveToFile
(Savedialog1.FileName)        // Колдонуучу диалогдук терезени «Сохранить»
кнопкасынын жардамында жапса, документти сактайбыз.
else CanClose:=false; // «Cancel» кнопкасынын жардамында жапса, терезени
жабууга уруксат берилбейт.
end
else RichEdit1.Lines.SaveToFile(Savedialog1.FileName);
 // Файл мурда сакталган болсо, эски аты менен сактайбыз, диалогдук терезе пайда
болбойт
end;
mrCancel: CanClose:=false; // Колдонуучу аракетинен баш тартса, терезени
жабууга уруксат берилбейт
end;
end;
```

Меню сабынан опцияны тандаганда, опцияга тийиштүү OnClick окуясынын процедурасы түзүлөт. Кийинки программалык коддорду жазуу үчүн көрсөтүлгөн опцияны тандап, пайда болгон программалык кодго аны кошуп жазгыла.

Колдонуучу «Новый» опциясын тандаганда RichEdit1 компонентинин мазмуну тазаланышы керек. Тазалоонун алдында, колдонуучу текстке өзгөртүү киргизгендигин текшерүү керек. Эгерде өзгөртүү киргизилген болсо текстти сактоо жөнүндөгү диалог чыгарылат. Колдонуучунун суроого жооп беришине жараша аракеттерди аткарып болгон соң, RichEdit1 компонентинин мазмуну тазаланат. Бул опциянын окуясына төмөнкү кодду жазабыз.

```
procedure TForm1.N2Click(Sender: TObject);
begin
if RichEdit1.Modified then  // Документте өзгөрүү болсо:
begin
case MessageDlg('Документ изменен, сохранить?', mtConfirmation,
[mbYes,mbNo,mbCancel],0) of  // Колдонуучунун жообун текшеребиз.
mrYes: begin  // колдонуучу «Yes» деп жооп берсе, файлды сактоо диалогун ачабыз.
if Savedialog1.FileName='Noname' then
begin
if Savedialog1.Execute then
begin
RichEdit1.Lines.SaveToFile(Savedialog1.FileName);
RichEdit1.Clear;
```

```
    end;
  end
  else
  begin
  RichEdit1.Lines.SaveToFile(Savedialog1.FileName);
  RichEdit1.Clear;
  end;
  end;
  mrNo:RichEdit1.Clear;  // Колдонуучу «No» деп жооп берсе.
  end;
  end
  else // Документте өзгөрүү болбосо,
  begin
  RichEdit1.Clear;
  Savedialog1.FileName:='Noname'; // касиетти калыбына келтиребиз.
  Form1.Caption:='RTF редактор-'+Savedialog1.FileName;
  end;
  end;
```

Программадагы аракеттер терезени жабуу алдындагы аракеттерге окшош. Айта кетчү нерсе, RichEdit1 объектисинин Clear методу анын Modified касиетин false маанисине өзгөртөт.

«Сохранить» опциясына жазылуучу программанын коду:

```
procedure TForm1.N4Click(Sender: TObject);
begin
if Savedialog1.FileName='Noname' then //Документ жаңы түзүлгөн болсо,
begin
if Savedialog1.Execute then  // колдонуучу диалогдук терезени "Сохранить"
кнопкасынын жардамында жапса, документти сактайбыз.
begin
RichEdit1.Lines.SaveToFile(Savedialog1.FileName);
Richedit1.Modified:=false;
end;
end
else      // Документ мурда сакталган болсо, эски аты менен сактайбыз.
begin
RichEdit1.Lines.SaveToFile(Savedialog1.FileName);
Richedit1.Modified:=false;
end;
Form1.Caption:='RTF редактор-'+Savedialog1.FileName;
end;
```

Бул опциянын программалык коду да мурдагы опциялардын коддоруна окшош эле. Мында SaveToFile методу Modified касиетинин маанисин өзгөртпөгөндүктөн, аны программалык жол менен өзгөртөбүз.

«Открыть» опциясынын программалык коду бир аз татаал көрүнүшкө ээ. Бул коддо файлды ачуу алдындагы текшерүүлөр жана чондуктардын маанилерин шартка жараша алмаштыруу бир нече жолу кайталанат.

```
procedure TForm1.N3Click(Sender: TObject);
begin
if RichEdit1.Modified then  // Документте өзгөрүү болсо:
begin
case MessageDlg('Документ изменен, сохранить?', mtConfirmation,
[mbYes,mbNo,mbCancel],0) of  // Колдонуучунун жообун текшеребиз.
mrYes: begin   // Колдонуучу «Yes» деп жооп берсе
if Savedialog1.FileName='Noname' then // жана документ жаңы түзүлгөн болсо:
begin
if Savedialog1.Execute then // Колдонуучу диалогдук терезени "Сохранить"
кнопкасынын жардамында жапса, документти сактайбыз.
begin
RichEdit1.Lines.SaveToFile(Savedialog1.FileName);
// Файлды ачуунун диалогдук терезесин жүктөйбүз.
If opendialog1.Execute then // Колдонуучу диалогдук терезени "Открыть"
кнопкасынын жардамында жапса, документти ачабыз.
begin
RichEdit1.Lines.LoadFromFile(opendialog1.FileName);
Savedialog1.FileName:= OpenDialog1.FileName; // Диалогдун касиетин
алмаштырабыз.
Richedit1.Modified:=false;
end;
end;
end
else     // Документ мурда сакталган болсо, эски аты менен сактайбыз.
begin
RichEdit1.Lines.SaveToFile(Savedialog1.FileName);
If opendialog1.Execute then
begin
RichEdit1.Lines.LoadFromFile(OpenDialog1.FileName);
Savedialog1.FileName:= OpenDialog1.FileName;
Richedit1.Modified:=false;
end;
end;
Form1.Caption:='RTF редактор-'+Savedialog1.FileName;
end;
mrNo: begin // Колдонуучу «No» деп жооп берсе:
If opendialog1.Execute then
```

```
    begin
RichEdit1.Lines.LoadFromFile(opendialog1.FileName);
Savedialog1.FileName:=opendialog1.FileName;
Richedit1.Modified:=false;
end;
end;
end;
end
else // Докуметте өзгөрүү болбосо:
If opendialog1.Execute then
begin
RichEdit1.Lines.LoadFromFile(opendialog1.FileName);
Savedialog1.FileName:=opendialog1.FileName;
Richedit1.Modified:=false;
end;
Form1.Caption:='RTF редактор-'+Savedialog1.FileName;
end;
```

Колдонуучу менюнун «Сохранить как» опциясын тандаса, файлга жазуунун диалогдук терезесин жүктөйбүз, бул учурда документтеги өзгөрүүлөр текшерилбейт.

```
procedure TForm1.N5Click(Sender: TObject);
begin
if Savedialog1.Execute then
begin
RichEdit1.Lines.SaveToFile(Savedialog1.FileName);
Richedit1.Modified:=false;
Form1.Caption:='RTF редактор-'+Savedialog1.FileName;
end;
end;
```

«Выход» опциясында форманы жабабыз, ал бир эле саптан турат.

```
procedure TForm1.N7Click(Sender: TObject);
begin
Form1.Close;
end;
```

Эми «Правка» менюсунун опцияларынын аракеттерин жазып чыгабыз. Бул аракеттер бир саптан турушат, алар:

«Отменить» опциясына ***RichEdit1.Undo;***
«Вырезать» опциясына ***RichEdit1.CutToClipboard;***
«Копировать» опциясына ***Richedit1.CopyToClipboard;***
«Вставить» опциясына ***RichEdit1.PasteFromClipboard;***

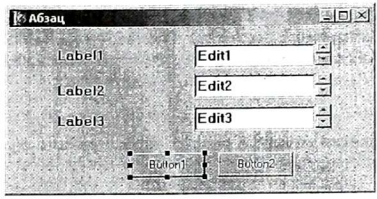
«Удалить» опциясына ***RichEdit1.ClearSelection;***
«Выделить все» опциясына ***RichEdit1.SelectAll;*** саптарын жазабыз.

«Формат» менюсунун опцияларынын аракеттерин карайбыз.
«Шрифт» опциясы тексттин шрифтин өзгөртөт, бул опция тандалганда FontDialog1 компонентинин Execute методун пайдаланабыз. Методду чакыруудан мурда диалогдук терезедеги шрифттин касиеттерин тандалган тексттин шрифтинин касиеттери менен дал келтиребиз. Колдонуучу диалогдук терезеден шрифттин касиетин өзгөрткөндөн кийин, жогорудагы аракетти тескерисинен жасап чыгабыз.

```
procedure TForm1.N17Click(Sender: TObject);
begin
FontDialog1.Font.Color:=RichEdit1.SelAttributes.Color;
FontDialog1.Font.Charset:=RichEdit1.SelAttributes.Charset;
FontDialog1.Font.Name:=RichEdit1.SelAttributes.Name;
FontDialog1.Font.Size:=RichEdit1.SelAttributes.Size;
FontDialog1.Font.Style:=RichEdit1.SelAttributes.Style;
if FontDialog1.Execute then
begin
RichEdit1.SelAttributes.Color:=FontDialog1.Font.Color;
RichEdit1.SelAttributes.Charset:=FontDialog1.Font.Charset;
RichEdit1.SelAttributes.Name:=FontDialog1.Font.Name;
RichEdit1.SelAttributes.Size:=FontDialog1.Font.Size;
RichEdit1.SelAttributes.Style:=FontDialog1.Font.Style;
end;
end;
```

«Список» опциясына (N23 опциянын объектинин аты),
If N23.Checked then RichEdit1.Paragraph.Numbering:=nsBullet else RichEdit1.Paragraph.Numbering:=nsNone;
«По левому краю» опциясына,
RichEdit1.Paragraph.Alignment:=taLeftJustify;
«По правому краю» опциясына,
RichEdit1.Paragraph.Alignment:=taRightJustify;
«По центру» опциясына,
RichEdit1.Paragraph.Alignment:=taCenter; сабын жазабыз.

«Абзац» опциясы документтин абзацына тийиштүү бир нече касиетти орнотот. Бирок, анын стандарттуу диалогдук терезеси жок болгондуктан, аны өзүбүз жасайбыз. Ал үчүн File/New/Form опциясын тандайбыз. Жаңы форма пайда болот. Анын BorderStyle касиетине bsDialog жана Position касиетине poDesktopCenter маанисин орнотобуз. Формага элементтерди сүрөттө көрсөтүлгөндөй кылып жайгаштырабыз.



Label1, Labe2 жана Label3 компоненттеринин Caption касиеттерине тиешелеш түрдө «Отступ первой строки», «Отступ слева» жана «Отступ справа» саптарын жазабыз.

UpDown1, UpDown2 жана UpDown3 компоненттеринин касиеттерин таблицада көрсөтүлгөндөй кылып кийрибиз.

Касиети	UpDown1 үчүн	UpDown2 үчүн	UpDown3 үчүн
Associate	Edit1	Edit2	Edit3
Max	100	200	200
Min	0	-100	0

Button1 жана Button2 кнопкаларынын касиеттерин орнотобуз.

Касиети	Button1 үчүн	Button2 үчүн
Caption	Применить	Отмена
ModalResult	mrOk	mrCancel

Биринчи форма экинчи формага кайрылуу жасагандыктан Unit1 модулунун Uses бөлүгүнө Unit2 модулун кошобуз.

```
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, Menus, ToolWin, ComCtrls, StdCtrls, ImgList, Unit2;
```

Эми «Абзац» опциясынын окуясына программалык кодду жазабыз.

```
procedure TForm1.N18Click(Sender: TObject);
begin
Form2.Edit1.Text:=Inttostr(RichEdit1.Paragraph.FirstIndent);
Form2.Edit2.Text:=Inttostr(RichEdit1.Paragraph.LeftIndent);
Form2.Edit3.Text:=Inttostr(RichEdit1.Paragraph.RightIndent);
Form2.ShowModal;
If Form2.ModalResult=mrOk then
begin
```

```
    RichEdit1.Paragraph.FirstIndent:=strtoint(Form2.Edit1.Text);
RichEdit1.Paragraph.LeftIndent:=strtoint(Form2.Edit2.Text);
RichEdit1.Paragraph.RightIndent:=strtoint(Form2.Edit3.Text);
end;
end;
```

Мына ушинтип менюнун бардык опцияларынын окуяларына аракеттерди жазып чыктык.

III этап. Инструменттердин панелин жана контексттик менюну түзүү

Toolbar1 компонентинин контексттик менюсунун NewButton жана New Seperator опцияларынын жардамында кнопкаларды сүрөттө көрсөтүлгөндөй кылып жайгаштырабыз жана анын Images касиетине ImageList1 компонентин көрсөтөбүз.

Талицада ToolBar1 компонентинин кнопка объекттеринин аттары жана аракеттери көрсөтүлдү.

Объекттин аты	Аракети
ToolButton1	Новый
ToolButton2	Открыть
ToolButton3	Сохранить
ToolButton4	|
ToolButton5	Отмена
ToolButton6	Вырезать
ToolButton7	Копировать
ToolButton8	Вставить
ToolButton9	|

Объекттин аты	Аракети
ToolButton10	По левому краю
ToolButton11	По правому краю
ToolButton12	По центру
ToolButton13	|
ToolButton14	Жирный
ToolButton15	Курсив
ToolButton16	Подчеркнутый
ToolButton17	|
ToolButton18	Список

Бөлгүчтөрдүн Style касиеттерине tbsDivider маанисин орнотобуз.

Алгачкы 10 жана 14 – кнопканы кезеги менен тандап (сүрөт боюнча санаганда), алардын MenuItem касиетине меню сабынын тиешелүү опцияларын көрсөтөбүз. Эсиңерде болсо менюнун опциялары N1, N2 ж.б. түрүндө белгиленет. Кнопканын бул касиети орнотулганда кнопканын пиктограммалык сүрөтү менюнун опциясынын сүрөтү менен алмаштырылат. 14 – кнопканын Style касиетине tbsCheck маанисин орнотобуз. Зарыл болгон менюнун опцияларын тез табуу үчүн компоненттерди дарак түрүндө көрсөтүүчү терезени пайдаланабыз.

8, 9 жана 10 – кнопкалардын Grouped касиеттерине true маанисин, Style касиетине tbsCheck маанисин орнотобуз. 11, 12 жана 13 – кнопкалардын AllowAllUp касиетине true жана Style касиетине tbsCheck маанисин орнотобуз.

11- кнопканын OnClick окуясына (мисалда ToolButton14)

```
procedure TForm1.ToolButton14Click(Sender: TObject);
begin
if (not ToolButton14.Down) and (not ToolButton15.Down) and (not
ToolButton16.Down) then RichEdit1.SelAttributes.Style:=[];
if (ToolButton14.Down) and (not ToolButton15.Down) and (not
ToolButton16.Down) then RichEdit1.SelAttributes.Style:=[fsBold];
if (not ToolButton14.Down) and (ToolButton15.Down) and (not
ToolButton16.Down)then RichEdit1.SelAttributes.Style:=[fsItalic];
if (not ToolButton14.Down) and (not ToolButton15.Down) and
(ToolButton16.Down) then RichEdit1.SelAttributes.Style:=[fsUnderline];
if (ToolButton14.Down) and (ToolButton15.Down) and (not
ToolButton16.Down) then RichEdit1.SelAttributes.Style:=[fsBold,fsItalic];
if (ToolButton14.Down) and (not ToolButton15.Down) and
(ToolButton16.Down) then RichEdit1.SelAttributes.Style:=[fsBold,
fsUnderline];
if (not ToolButton14.Down) and (ToolButton15.Down) and
(ToolButton16.Down) then RichEdit1.SelAttributes.Style:=[fsItalic,
fsUnderline];
if (ToolButton14.Down) and (ToolButton15.Down) and (ToolButton16.Down)
then RichEdit1.SelAttributes.Style:=[fsBold,fsItalic, fsUnderline];
end;
```

Бул программалык коддо үч кнопканын басылып же басылбай турган абалдарынын комбинациясына жараша тексттин шрифтинин стили орнотулат. 12 жана 13 – кнопкалардын OnClick окуяларыны 11 – кнопканын окуясына шилтемелейбиз.

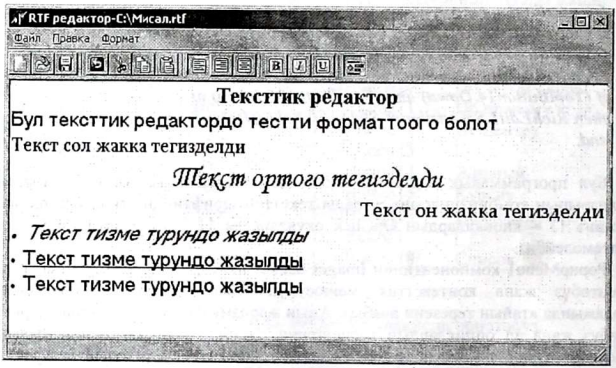
PopupMenu1 компонентинин Images касиетине сүрөттөрдүн коллекциясын көрсөтөбүз жана контексттик менюсунун Menu Designer опциясынын жардамында атайын терезени ачабыз. Анын жардамында менюнун опцияларын түзөбүз жана ал опциялардын касиеттерин төмөндө көрсөтүлгөндөй кылып орнотобуз. Ошондой эле ImageIndex касиеттерине опцияга тийешелүү сүрөттү тандап коюуну унутпагыла.

Caption касиети	Башка касиеттери	Окуясы
&Вырезать	MainMenu1	MainMenu1
&Копировать	компонентинин	компонентинин
Вст&авить	тиешелеш	тиешелеш
-	опциясынын	опциясынын
В&ыделить все	касиетиндей	окуясына
&Отменить	орнотулат	шилтеме жасашат

-
&Шрифт
&Абзац
&Список
-
По &левому краю
По &правму краю
По &центру

Опцияларды түзүп, касиеттерин жана окуяларын орнотуп болгон соң RichEdit1 компонентинин PopupMenu касиетине тизмеден PopupMenu1 сабын тандап коебуз.

Ушуну менен текстгик редактордун программансын түзүүнү аяктадык. Бул редакторду толук түзүлгөн редактор катары эсептөөгө болбойт. TRichEdit компонентинин башка касиеттерин пайдалануу менен редактордун мүмкүнчүлүгүн кеңейтип, документти форматтоону дагы жакшыртуу мүмкүнчүлүгү бар. Бул жумушту колдонуучу өз алдынча аткаруусун сунуштайбыз.

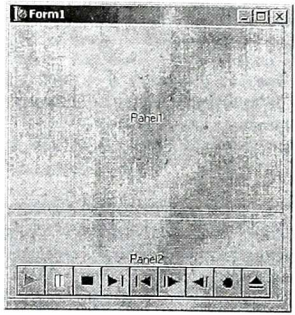


3.6. Медияплеер

Алгач жөнөкөй медияпрлеерди түзөбүп алабыз, андай кийин анын мүмкүнчүлүгүн кеңейтебиз.

I этап. Жөнөкөй медияплеердин негизги терезесин түзүү.

Формага эки TPanel (Standard) жана бир TMediaPlayer (System) компоненттерин сүрөттө көрсөтүлгөндөй кылып жайгаштырып алабыз (Mediaplayer1 компонентин Panel2 компонентине жайгаштыруу керек).



Panel1 жана Panel2 элементтеринин Caption касиеттерин тазалап салабыз. MediaPlayer1 элементинин Display касиетине тизмеден Panel1 сабын тандайбыз, AutoOpen касиетине true маанисин жана FileName касиетине C:\Program Files\Borland\Delphi6\Demos\CoolStuf \speedis.avi файлын орнотобуз. Программаны жүктөп анын иштөөсүн текшеребиз.

II этап. Медияплеерде башка файлдарды ойноо

Жогорку мисалда бир гана файлды ойноону карадык. Эми колдонуучу тандаган файлды ойноону ишке ашырабыз. Ал үчүн формага TOpenDialog (Dialogs) жана TSpeedButton (Additional) компоненттерин жайгаштырабыз (SpeedButton1 элементин Panel2 элементине жайгаштырабыз). MediaPlayer1 элементинин FileName касиетин тазалап салабыз жана AutoOpen касиетине false маанисин орнотобуз. OpenDialog1 элементинин Filter касиетине төмөнкү тизмедеги маанилерди кийиришиз.

Филтрдин аты	Филтрлер
Avi	*.avi
Mpeg	*.mpe
Midi	*.mid
Mp3	*.mp3
Wave	*.wav
Все	*.*

SpeedButton1 кнопкасынын Glyph касиетинде тийешелүү сүрөттү тандап, анын OnClick окуясына төмөнкү саптарды жазабыз.

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
if Opendialog1.Execute then
begin
MediaPlayer1.Close;
MediaPlayer1.FileName:=Opendialog1.FileName;
MediaPlayer1.Open;
MediaPlayer1.Play;
end;
end;
```

Медияплеер ачык учурда ага башка файлды жүктөөгө болбойт, ошондуктан жаңы файлды медияплеерге жүктөөрдүн алдында аны жабуу керек. Файлдын атын алмаштыргандан кийин кайра медияплеерди ачабыз.

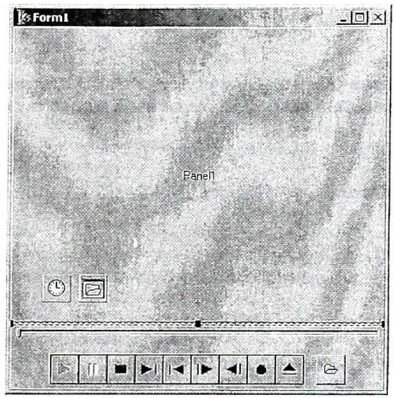
III этап. Экрандын өлчөмүн өзгөртүү

2 – этаптагы мисалды иштетип көргөнүбүздө, көрсөтүлүп жаткан кадрлар панелди толук ээлебегенин байкайбыз. Экрандын өлчөмүн толук ээлөө үчүн медияплеердин жана панелдин жаңы касиеттери менен таанышабыз. Алар TRect тибиндеги медияплеердин DisplayRect жана панелдин BoundsRect касиеттери. Бул касиеттер элементтер ээлеген тик бурчтуу областтын чондуктарынын тизмесинен турат. TRect тиби тизме болуп integer тибиндеги Left, Top, Right, Bottom талааларына ээ. Бул талаалар элементтин областынын жактарынын координаттарын көрсөтөт. Биздин учурда, экрандын өлчөмүн панелдин өлчөмүнө ушул касиеттери боюнча дал келтиребиз жана жогорудагы программа төмөнкү көрүнүшкө келет.

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
if Opendialog1.Execute then
begin
MediaPlayer1.Close;
MediaPlayer1.FileName:=Opendialog1.FileName;
MediaPlayer1.Open;
MediaPlayer1.DisplayRect:= Panel1.BoundsRect;
MediaPlayer1.Play;
end;
end;
```

IV этап. Интерфейсти жакшыртуу

Форманын өлчөмүн бир аз чоңойтобуз. Panel2 компонентини Aling касиетине alBottom жана Height касиетине 65 маанилерин орнотобуз. TSplitter (Additional) элементин форманын бош жерине жайгаштырып, анын да Aling касиетине alBottom жана Height касиетине 5 маанилерин орнотобуз. Эми Panel1 компонентини Aling касиетине alClient маанисин орнотобуз. Panel2 компонентине TTackBar (Win32) компонентин жайгаштырып, анын Aling касиетине alTop жана ThumbLength касиетине 10 жана TickStyle касиетине tsNone маанилерин орнотобуз. TTimer (System) компонентини жайгаштырып, Enabled касиетине false маанисин орнотобуз. Жалпысынан форма эми төмөнкү сүрөттөгү көрүнүшкө келет.



Эми программалык коддорду кайрадан түзүп чыгабыз.

SpeedButton1 кнопкасынын OnClick окуясына, TrackBar1 компонентинин максималдык маанисине медия файлдын ойноо узундугун ыйгарууну ишке ашырган жана таймерди активдештирген саптарды кошобуз. Эми ал төмөнкүчө болот.

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
if Opendialog1.Execute then
begin
Timer1.Enabled:=false;
MediaPlayer1.Close;
MediaPlayer1.FileName:=Opendialog1.FileName;
```

```
    MediaPlayer1.Open;
TrackBar1.Max:=MediaPlayer1.Length;
MediaPlayer1.DisplayRect:=Panel1.BoundsRect;
MediaPlayer1.Play;
Timer1.Enabled:=true;
end;
end;
```

Panel1 элементинин өлчөмү өзгөргөндө экрандын өлчөмүн да өзгөртүү зарыл. Бул аракетти OnResize окуясына жазабыз.

```
procedure TForm1.Panel1Resize(Sender: TObject);
begin
MediaPlayer1.DisplayRect:=Panel1.BoundsRect;
end;
```

Timer1 элементинин OnTimer окуясына ойнолуп жаткан медиа файлдын учурдагы позициясын TrackBar1 компонентинде көрсөтүүнү ишке ашырабыз.

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
TrackBar1.Position:=Mediaplayer1.Position;
end;
```

TrackBar1 элементинин OnChange окуясында ушул элементтин жардамында мультимедиялык файлды түрүүнүн программасын жазабыз.

```
procedure TForm1.TrackBar1Change(Sender: TObject);
begin
Mediaplayer1.Position:=TrackBar1.Position;
Mediaplayer1.Resume;
end;
```

4 - этапта биз түзгөн медияплеердин учурда бир жетишпеген жагы, кээ бир кадрларда токтоп калуу эффекти байкалат. Анын себеби, OnTimer окуясында убакыттын өтүшү менен TrackBar1 позициясын өзгөртүп турсун үчүн TrackBar1.Position:= Mediaplayer1.Position; сабын пайдаландык. TrackBar1 позициясын өзгөрткөндө анын OnChange окуясы пайда болот. Бул окуяда биз жогорку сапты чондуктардын ордун алмаштыруу менен жаздык, башкача айтканда учурдагы кадрды кайра эле медияплеерге орноттук. Мына ушинтип ар бир секунданын аягындагы бир кадр 2 жолу кайталанып жаткандыктан, токтоп калуу эффектисин пайда кылууда.

3.7. Саат

Бул мисалда биз белгиленген убакытта экранга кабар чыгарып туруучу сааттын программасын түзөбүз.

I этап. Сааттын негизин түзүү

Формага TStaticText (Addition) компоненттин жайгаштырып анын касиеттерин объекттердин инспекторунун терезесинен төмөнкүдөй кылып өндөйбүз.

Касиети	Мааниси
Color	clBlack
BorderStyle	sbsSunken
BevelKind	bkSoft
Alignment	alCenter
Width	130
Hieght	30
Font:	
Size	18
Color	clGreen
Style:	
fsBold	true

Формага TTimer (System) компоненттин жайгаштырабыз. Таймердин касиеттерин өзгөртпейбүз. Объекттердин инспекторунун Events бөлүгүнө өтүп, OnTimer окуясына эки төмөнкү сапты жазабыз.

```
begin
StaticText1.Caption:=TimeToStr(now);
end;
```

Программанын жүктөгөндө, сааттагы убакыт бир аз кечигүү менен пайда болгону байкалат. Кечигүүнү жок кылуу үчүн форманын OnCreate окуясынын оң жагындагы кнопканы басып, пайда болгон тизмеден Timer1Timer сабын тандайбыз. Башкача айтканда форманын OnCreate окуясы таймердин OnTimer окуясына шилтеме жасайт.

Эми эки TLabel (Standard) жана бир TSeedButton элементтерин жайгаштырабыз. Label1 жана Label2 элементтеринин AutoSize касиетине false, Alingment касиетине alCenter маанилерин орнотобуз. SpeedButton1 кнопкасынын Gliph касиетине тиешелүү сүрөттү жайгаштырабыз жана элементтерди сүрөттө көрсөтүлгөндөй кылып жайгаштырабыз.

Программалык коддо төмөнкү үч чондукту жарылайбыз.

```
var
Form1: TForm1;
mess,dat,tim:string;
```

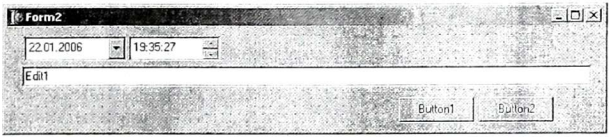
Таймердин OnTimer окуясын кайрадан оңдоп жазабыз.

```
procedure TForm1.Timer1Timer(Sender: TObject);
var
s:string [11];
begin
StaticText1.Caption:=TimeToStr(now);
Label1.Caption:=DateToStr(now);
case DayOfWeek(now)of
1 : s:='Воскресенье';
2 : s:='Понедельник';
3 : s:='Вторник';
4 : s:='Среда';
5 : s:='Четверг';
6 : s:='Пятница';
7 : s:='Суббота';
end;
Label2.Caption:= trim(s);
end;
```

Программабыз эми датаны жана жуманын күнүн көрсөтө алат.

II этап. Убакытты жана кабарды көрсөтүү

Саатка убакытты бегилөө жана кабарды жазуу үчүн жаңы форманы ушул проектке кошобуз. Ал үчүн менюнун File/New/Form опциясын тандайбыз. Ал формага эки TDateTimePicker (Win32), бир TEdit жана эки TButton (Standard) элементтерин сүрөттөгүдөй кылып жайгаштырабыз.

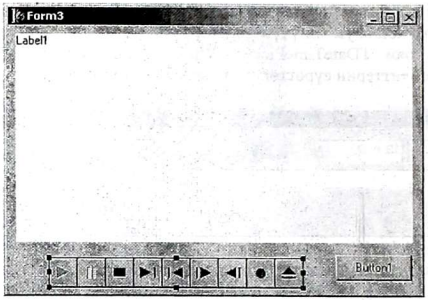


DateTimePicker1 элементинин Kind касиетине dtkDate жана DateTimePicker2 элементинин ушул эле касиетине dtkTime маанилерин орнотобуз. Button1 жана Button2 элеметтеринин Caption касиеттерине «Принять» жана «Отмена», ModalResult касиеттерине mrOK жана mrCancel маанилерин тиешелеш түрдө орнотобуз.

Form2 формасынын OnCreate касиетине төмөнкү кодду жазабыз.

```
procedure TForm2.FormCreate(Sender: TObject);
begin
DateTimePicker1.Date:=now;
DateTimePicker1.MinDate:=now;
DateTimePicker2.Time:=now;
end;
```

Кабарды экранга чыгарууну ишке ашыруу үчүн жогоруда көрсөтүлгөн жол менен, дагы бир форманы түзөбүз. Ал формага TLabel, TButton (Standard) жана TMediaPlayer (System) элементтерин жайгаштырабыз. Label1 элементинин AutoSize касиетине false жана WordWrap касиетине true маанилерин орнотуп, анын өлчөмүн тоңойтобуз. Button1 элементинин Caption касиетине «Закрыть» жана ModalResult касиетине mrOK маанилерин орнотобуз. MediaPlayer1 элементинин FileName касиетине C:\WINNT\Media\ папкасынан бир файлды тандайбыз жана AutoOpen касиетине true, Visible касиетине false маанилерин орнотобуз. Аларды формага төмөнкүдөй кылып жайгаштырабыз.



Button1 кнопкасынын OnClick окуясыны көрсөтүлгөн программаны жазабыз.

```
procedure TForm3.Button1Click(Sender: TObject);
begin
Mediaplayer1.Stop;
end;
```

Ал эми Unit1 модулунун Uses бөлүгүнө Unit2 жана Unit3 модулдарын кошуп жазабыз.

```
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, Buttons, ExtCtrls, StdCtrls, unit2, unit3;
```

Биринчи формадагы SpeedButton кнопкасынын OnClick окуясына төмөндөгү саптарды жазабыз.

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
Form2.ShowModal;
if Form2.ModalResult=mrOk then
begin
Mess:=Form2.Edit1.Text;
Dat:=DateToStr(Form2.DateTimePicker1.Date);
Tim:=TimeToStr(Form2.DateTimePicker2.Time);
end;
end;
```

Ал эми Timer1 элементинин OnTimer окуясына төмөндө көрсөтүлгөдөй кылып түзөтүү киргизебиз.

```
procedure TForm1.Timer1Timer(Sender: TObject);
var
s:string [11];
begin
StaticText1.Caption:=timetostr(now);
Label1.Caption:=datetostr(now);
case DayOfWeek(now)of
1 : s:='Воскресенье';
2 : s:='Понедельник';
3 : s:='Вторник';
4 : s:='Среда';
5 : s:='Четверг';
6 : s:='Пятница';
7 : s:='Суббота';
end;
Label2.Caption:= trim(s);
if (Dat=DateToStr(now)) and (Tim=TimeToStr(now)) then
begin
Form3.Label1.Caption:=mess;
Form3.MediaPlayer1.Play;
Form3.ShowModal;
end;
```

Delphi чөйрөсүнү терендетип үйрөнгөндөн кийин, бул программага башка көптөгөн функцияларды кошуу менен өркүндөтүүгө болот.

3.8. Функциянын графиги

Бул мисалда TCanvas объекттик тибиндеги касиеттин жардамында функциянын графигин тургузууну карайбыз. TCanvas классынын методдору менен тиркемеден таанышууга болот.

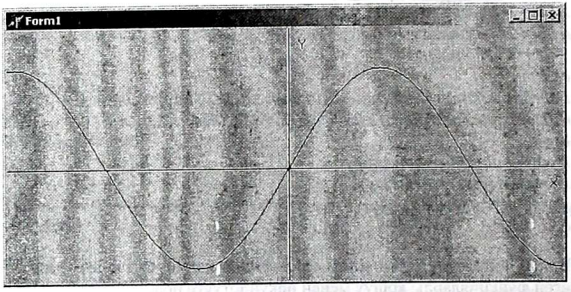
Мисал катары форманын бетине синусоиданын графигин тургузабыз. Ал үчүн форманын OnPaint окуясына төмөнкү саптарды жазабыз.

```
procedure TForm1.FormPaint(Sender: TObject);
var
i:integer;
begin
Form1.Canvas.MoveTo(0,Form1.Height div 2);
Form1.Canvas.LineTo(Form1.Width,Form1.Height div 2);
Form1.Canvas.TextOut(Form1.Width-20,Form1.Height div 2+10,'X');
Form1.Canvas.MoveTo(Form1.Width div 2,0);
Form1.Canvas.LineTo(Form1.Width div 2,Form1.Height);
```

```
    Form1.Canvas.TextOut(Form1.Width div 2+10,10,'Y');
Form1.Canvas.MoveTo(0,Form1.Height);
for i:=-540 to 540 do
Form1.Canvas.LineTo(Form1.Width div 2+i,Form1.Height div 2-
trunc(100*sin(i*pi/180)));
end;
```

Бул программалык коддо форманын Canvas касиетин колдондук. Бул касиетти сүрөт тартуу үчүн альбомдун бети сыяктуу элестетүүгө болот. Ал эми ага чийүүнү ишке ашыруучу графикалык приметивдердин көпчүлүгү Паскаль тилиндегидей эле аныкталган.

Программаны жүктөгөндө сүрөттө көрсөтүлгөн графиктин пайда болгонун көрөбүз.



Бирок, форманын өлчөмүн өзгөрткөндө график бузулат. Ал эми аны түрүп жана калыбына келтиргенде кайрадан туура чийме пайда болот. Демек, форманын өлчөмү өзгөргөндө чиймени калыбына келтирүүбүз керек. Ал үчүн форманы тазалап, кайрадан графикти чийүүнү ишке ашырабыз.

Форманы OnResize окуясына төмөнкү эки сапты жазабыз.

```
procedure TForm1.FormResize(Sender: TObject);
begin
Form1.Invalidate;
Form1.Paint;
end;
```

Формага чийилген чиймени тазалоо үчүн форманын Invalidate методду колдонулду. Экинчи сапта форманын Paint методуна кайрылдык, анткени бул метод форманын OnPaint окуясын пайда кылгандыктан чийме кайрадан тартылат. Бул сапты төмөнкүчө да жазууга болот.

```
    procedure TForm1.FormResize(Sender: TObject);
begin
Form1.Invalidate;
Form1.OnPaint(Sender);
end;
```

Бул учурда окуяны метод аркылуу пайда кылбастан, аны түздөн түз пайдаландык. Sender параметри бар гана окуяны түздөн түз пайдалануурга болот. Ал эми бул прараметри жок окуяларга, мурдагы мисалда карагандай, анын методу аркылуу кайрылабыз.

Типтер

Delphi чөрөсүнүн жөнөкөй типтери Паскаль тилиндеги типтерден бир аз айырмаланышат жана аларга тийиштүү функция жана процедуралардын да катары кеңейтилген. Тикемеде типтерди жана аларга тийиштүү функция жана процедуралар менен кыскача таанышабыз.

1. Бүтүн сандардын тиби

Аталышы	Узундугу,байт	Маанилеринин диапозону
Cardinal	4	0 ... 4294967295
Byte	1	0...255
Shortint	1	-128...+127
Smallint	2	-32 768...+32 767
Word	2	0...65 535
Integer	4	-2 147 483 648...+2 147 483 647
Longint	4	-2 147 483 648...+2 147 483 647
Int64	8	-2^63 . ..+2^63-1
LongWord	4	0. . .4 294 967 295

Бүтүн сандарды башка типтерге конверттөөнүн функциялары:

function IntToBin(Value: cardinal): string - бүтүн санды ондук эсептөө системасынан экилик эсептөө системасына которуп, жыйынтыкты сап түрүндө кайтарат.

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

function IntToHex(Value: Int64; Digits: Integer): string – бүтүн санды ондук эсептөө системасынан он алтылык эсептөө системасына которуп, жыйынтыкты сап түрүндө кайтарат.

function IntToStr(Value: Integer): string жана

function IntToStr(Value: Int64): string – бүтүн санды саптык типке өзгөртөт.

2. Чыныгы сандардын тиби

Аталышы	Узундугу, байт	Мантиссасы	Маанилеринин диапозону
Real	8	15…16	5.0*10^-324…1.7*10^308
Real48	6	11…12	2.9 * 10^–39 .. 1.7 * 10^38
Single	4	7…8	1.5*10^-45…3.4*10^38
Double	8	15…16	5.0*10^-324…1.7*10^308

Extended	10	19...20	3.6*10^-4951...1.1*10^4932
Comp	8	19...20	-2^63...+2^63-1
Currency	8	19...20	+/-922 337 203 685477,5807

Чыныгы сандарды башка типтерге конверттөөнүн функциялары:

function CompToCurrency(Value: Comp): Currency – Comp тибиндеги санды Currency тибине өзгөртөт.

function CompToDouble(Value: Comp): Double – Comp тибиндеги санды Double тибине өзгөртөт.

function CurrToStr(Value: Currency): string – Currency тибиндеги санды саптык типке өзгөртөт.

function DoubleToComp(Value: Double; var Result: Comp) – Double тибиндеги чоңдукту Comp тибине өзгөртөт.

function CurrToStrF(Value: Currency; Format: TFloatFormat; Digits: Integer): string - Currency тибиндеги санды көрсөтүлгөн форматта саптык типке өзгөртөт.

function FloatToCurr(const Value: Extended): Currency – чыныгы санды Currency тибине өзгөртөт. Сандын минималдык жана максималдык маанилери Currency тибинин диапозонунда жатышы зарыл.

function FloatToDateTime(const Value: Extended): TDateTime – чыныгы санды TDateTime тибине өзгөртөт. Сандын минималдык жана максималдык маанилери төмөнкүчө чектелген.

MinDateTime: TDateTime = -657434.0; {01/01/0100 12:00:00.000 AM},
MaxDateTime: TDateTime = 2958465.99999; {12/31/9999 11:59:59.999 PM}

function FloatToStr(Value: Extended): string – чыныгы санды саптык типке өзгөртөт.

function FloatToStrF(Value: Extended; Format: TFloatFormat; Precision, Digits: Integer): string - чыныгы санды көрсөтүлгөн форматта саптык типке өзгөртөт. Мында Format параметри төмөнкү маанилердин биринде боло алат. TFloatFormat = (ffGeneral, ffExponent, ffFixed, ffNumber, ffCurrency).

3. Саптык типтер

Delphi чөйрөсүндөгү саптык чоңдуктар:

- кыска саптар – **ShortString** же **String** [n], мында n <= 255;
- узун саптар – **String**;
- кең саптар – **WideString**;
- нуль-терминалдык сап – **Pchar**.

ShortString же String [n] типтери Паскалдагы String тиби менен дал келет жана символдору бир байттан турат. String тиби жогорку типтен узундугу боюнча айырмаланат жана анын узундугу 2 Гб – ка чейин боло алат. WideString тиби узундугу String тибиндегидей, бирок бул типтеги символдор эки байт

менен аныкталган. Pchar тиби аягы #0 менен бүтүүчү символдордун тизмеги болуп, өлчөмү чектелген эмес.

Саптык чондуктарды башка типтерге конверттөөнүн функциялары:

function StrToBool(const S: string): Boolean – саптык чондукту логикалык тибине өзгөртөт. Саптык чондуктун мааниси '1' (true) же '0' (false) болуусу зарыл.

function StrToCard(AVal: String): Cardinal – саптык чондукту Cardinal тибине өзгөртөт.

function StrToCurr(const S: string): Currency – саптык чондукту Currency тибине өзгөртөт.

function StrToFloat(const S: string): Extended – саптык чондукту чыныгы сандын тибине өзгөртөт.

function StrToInt(const S: string): Integer – саптык чондукту бүтүн сандын тибине өзгөртөт.

function StrToInt64(const S: string): Int64 – саптык чондукту Int64 тибиндеги бүтүн санга өзгөртөт.

4. Дата – убакыт тиби

TDateTime – тиби бир учурда датанын жана убакыттын маанилеринен турат. Ички түзүлүшү боюнча 8 байт болгон фиксирленген бөлчөк бөлүгү бар чыныгы сан. Бул сандын бүтүн бөлүгүндө дата, ал эми бөлчөк бөлүгүндө убакыт белгиленет.

TDateTime тиби менен иштөө үчүн кээ бир функциялар жана процедуралар:

function Date: TDateTime – учурдагы системалык датаны кайтарат.

function DateToStr(D: TDateTime): String – датаны сапка өзгөртөт. **function DateTimeToStr(DateTime: TDateTime): string** – датаны жана убакытты сапка өзгөртөт.

function DayOfWeek(Date: TDateTime): Integer – жуманын күнүнүн номерин кайтарат (1- жекшемби, …, 7 - ишемби).

procedure DecodeDate(Date: TDateTime; var Year, Month, Day: Word) – датадан жылды, айды жана күндү бөлүп алат.

procedure DecodeTime(Time: TDateTime; var Hour, Min, Sec, MSec: Word) – убакыттан саатты, минутаны, секунданы жана миллисекунданы бөлүп алат.

function EncodeDate(Year, Month, Day: Word): TDateTime – аныкберилген жыл, ай жана күндөн TDateTime форматындагы датаны түзөт.

function TryEncodeDate(Year, Month, Day: Word; outDate: TDateTime): Boolean – аныкберилген жыл, ай жана күндөн датаны түзүү мүмкүн экендигин текшерүү менен датаны түзөт.

function EncodeTime(Hour, Min, Sec, MSec: Word): TDateTime –аныкберилген саат, минута, секунда жана миллисекундадан TDateTime форматындагы убакытты түзөт.

function TryEncodeTime(Hour, Min, Sec, MSec: Word; out Time: TDateTime): Boolean - аныk берилген саат, минута, секунда жана миллисекундадан убакытты түзүү мүмкүн экендигин текшерүү аркылуу убакытты түзөт.

function FormatDateTime (Format: String ; Value: TDateTime): String – Value параметриндеги маанини Format спецификациясы боюнча символдордун сабына өзгөртөт.

function IsLeapYear(Year: Word): Boolean – узун жыл (февраль айында 29 күн) болсо true маанисин кайтарат.

function Now: TDateTime – учурдагы дата жана убакыты кайтарат.

function StrToDate(const S: string): TDateTime – дата форматында жазылган сапты TDateTime тибине өзгөртөт.

function StrToDateTime(const S: string): TDateTime – Windows-тун локалдык талабына ылайык дата- убакыт форматта жазылган сапты TDateTime тибине өзгөртөт.

function StrToTime(const S: string): TDateTime – убакыт форматында жазылган сапты TDateTime тибине өзгөртөт.

function Time: TDateTime – учурдагы убакытты кайтарат.

function TimeToStr(T: TDateTime): String - датаны сапка өзгөртөт.

Көрсөтүлгөн стандарттуу функциялар жана процедуралар SysUtils модулуна тийиштүү. TDateTime тибинин көптөгөн башка функциялары жана процедуралары DateUtils модулунда аныкталган.

5. SYSTEM модулунун өзгөрүлмөлөрү, процедуралары жана функциялары

SYSTEM модулу бардык программаларга автоматтык түрдө кошулат, ушул себептүү аны uses бөлүгүндө көрсөтүүнүн зарылчылыгы жок. Айта кетүүчү нерсе, Delphi чөйрөсүндө бул модулдун көпчүлүк процедурлары жана функциялары Паскаль тилиндегидей сакталып калган, кээ бирлери бир аз өзгөрүүгө учураган.

function Abs(X) – X санын абсолюттук маанисин кайтарат (бүтүн же чыныгы тип).

procedure Append (var F: Text) – жаңы саптарды кошуу үчүн F тексттик файлын ачат.

function ArcTan(X: Extended): Extended – X санын Арктангенсин кайтарат (радиада).

procedure AssignFile (var F; File Name: String) – FileName парамеринде көрсөтүлгөн файды F файлдык өзгөрүлмөсү менен байланыштырат.

procedure ChDir(S: Strings) – учурдагы каталогду S параметринде көрсөтүлгөн каталог менен алмаштырат.

function Chr(X: Byte): Char – X символун байтка өзгөртүрөт.

procedure Close(var F) – F файлын жабат.

procedure CloseFile(var F) – F файлын жабат.

function CompToCurrency(acomp:Comp): Currency – Comp тибин Currency тибине өзгөртөт.

function CompToDouble(acomp:Comp): Double – Comp тибини Double тибине өзгөртөт.

function Concat(sl [, s2,..., sn]: String): String – sn саптарын бир сапка бириктирет.

procedure Continue – for, while же repeat циклдарынын кезектеги итерациясын аткарбайт.

function Copy(S:String; Index, Count:Integer): String – саптын бөлүгүн кайтарат.

function Cos(X: Extended): Extended – радиан ченинде берилген X аргументинин косинусун кайтарат.

procedure CurrencyToComp (acurrency: Currency; var result:Comp) – Currency тибин Comp тибине өзгөртөт.

procedure Dec(var X [; N: LongInt]) – X санының маанисин N- ге кемитет, N көрсөтүлбөсө бирге кемийт.

procedure Delete(var S: String; Index, Count: Integer) – S сабынан Index символунан баштап Count символду өчүрөт.

function DoubleToComp(adouble: Double; var result: Comp) – Double тибини Comp тибине өзгөртөт.

function Eof(var F): Boolean – F файлынын аягына жеткенде true маанисин кайтарат.

function Eoln [(var F: Text)]: Boolean – F тексттик файлынын же саптын аягына жеткенде true маанисин кайтарат.

procedure Erase(var F) – F өзгөрүлмөсү менен байланышкан файлды өчүрөт.

procedure Exit – программа иштешин аяктайт.

function Exp(X: Real): Real – е санынын X даражасын эсептейт.

var FileMode: Byte – Reset процедурасы менен ачылган файлдын режимин көрсөтүүчү чондук: 0 – окуу үчүн гана; 1 – жазуу үчүн гана; 2 – окуу жана жазуу.

function FilePos(var F): Longint – F файлындагы учурдагы позицияны көрсөтөт.

function FileSize(var F): Integer – файлдын өлчөмүн кайтарат.

function Frac(X: Extended): Extended – X санын бөлчөк бөлүгүн кайтарат.

procedure GetDir(D: Byte; var S: String) – дисктин номери боюнча анын атын кайтарат.

procedure Inc(var X [; N: LongInt]) – X санынын маанисин N-ге чоңойтот, N көрсөтүлбөсө бирге чоңоет.

procedure Insert(Source: String; var S: String; Index: Integers) – Source сабын S сабынын Index позициясына жайгаштырат.

function Int(X: Extended): Extended – чыныгы сандын бүтүн бөлүгүн кайтарат.

function Length (S): Integer –S сабынын узундугун кайтарат.

function Ln(X: Real): Real – X санының натуралдык логарифмин кайтарат.

const Maxint = High(Integer) – Integer тибинин максималдык маанисине барабар чондук.

const MaxLongint = High(Longint) - Longint тибинин максималдык маанисине барабар чондук.

procedure MkDir(S: String) – жаңы каталог түзөт.

function Odd(X: Longint): Boolean – X аргументи так сан болсо true маанисин кайтарат.

procedure OleStrToStrVar (Source: PWideChar; var Dest: String) – «кең» (эки байттуу) сапты жөнөкөй сапка көчүрөт.

function Pi: Extended – Pi санынын маанисин кайтарат.

var RandSeed: Longint – псевдо кокустук удаалаштыктын генераторунун башталгыч маанисин аныктоочу чондук.

function Random [(Range: Integer)] – кезектеги псевдо кокустук санын кайтарат.

procedure Randomize - псевдо кокустук удаалаштыктын генераторун ишке киргизет.

procedure ReadLn(var F: Text; VI [, V2, . . ., Vn]) – F тексттик файлынан берилген сандагы сапты окуйт жана Vi өзгөрүлмөлөрүнө жайгаштырат.

procedure Rename(var F; Newname:String) жана

procedure Rename(var F; Newname:Pchar) – F файлдык өзгөрүлмөсү менен байланышкан файлдын атын өзгөртөт.

procedure Reset(var F [: File; RecSize: Word]) – бар файлды окуу жана/же жазуу үчүн ачат.

procedure Rewrite(var F: File [;Recsize: Word]) – жаңы файлды түзөт жана жазуу үчүн ачат.

procedure RmDir(S: Strings) – бош каталогду өчүрөт.

function Round(X: Extended):Int64 - чыныгы санды жакынкы бүтүн санга чейин тегеректейт.

procedure Seek(var F; N: LongInt) – файлдын башталышынан N байтты өткөрүп салат.

function SeekEof [(var F: Text)]: Boolean – файлдын аягына чейин бардык байттарды өткөрүп салат.

function Sin(X: Extended): Extended – аргументтин синусун кайтарат (радианда).

function SizeOf(X): Integer – X өзгөрүлмөсүнүн узундугун байтта кайтарат.

function Sqrt(X: Extended): Extended – аргументтин квадраттык тамырын кайтарат.

function StringOfChar(Ch: Char; Count: Integer): String - Count жолу кайталанган Ch символунан турган сапты түзөт.

function StringToOleStr(const Source: String): PWideChar – жөнөкөй сапты эки байттуу сапка көчүрөт.

function StringToWideChar (const Source: String; Dest: PWideChar; DestSize: Integer): PWideChar - жөнөкөй сапты UNICODE символдуу сапка өзгөртөт.

function Trunc(X: Extended): Int64 – чыныгы сандын бөлчөк бөлүгүн алып салуу менен бүтүн санга өзгөтөт.

function UpCase(Ch: Char): Char – Ch кичине тамгасын баш тамгага өзгөртөт.

procedure Val(S: String; var V;var Code: Integer) – саптык маанини бүтүн же чыныгы санга өзгөртөт.

procedure WideCharLenToStrVar (Source: PWideChar; SourceLen: Integer; var Dest: String) – UNICODE сабынын SourceLen санынан көп болбогон символун жөнөкөй сапка өзгөртөт.

procedure WideCharToStrVar (Source: PWideChar; var Dest: String) - UNICODE сабын жөнөкөй сапка өзгөртөт.

6. MATCH модулунун процедуралары жана функциялары

MATCH модулунун процедураларын жана функцияларын пайдалануу үчүн Uses бөлүгүнө молуду жазуу зарыл. Бул модулда көптөгөн математикалык, статистикалык жана финансылык процедуралар жана функциялар бар. Алардын эсетөө тездиги бир кыйла чоң.

Тригонометриялык процедуралар жана функциялар

function ArcCos(X: Extended): Extended – арккосинус.

function ArcSin(X: Extended): Extended – арксинус.

function ArcTan2(Y, X: Extended): Extended – Y/X арктангенсин эсептейт жана бурчту туура квадрантын кайтарат.

function Cotan(X: Extended): Extended – котангенс.

function Hypot (X, Y: Extended): Extended – (X** 2 + Y** 2) –нин квадраттык тамыры, эки катети боюнча тик бурчтуу үч бурчтуктун гипотенузасы.

procedure SinCos (Theta: Extended; var:Sin, Cos: Extended) – бир учурда Theta бурчунун синусун жана косинусун эсептейт (өз өзүнчө эсептөөгө караганда 2 эсе тез).

function Tan(X: Extended): Extended – тангенс.

Бучтарды өзгөтүүнүн функциялары

function CycleToRad(Cycles: Extended) : Extended – айлананы радианга өзгөртүү Radians := Cycles * 2PI.

function DegToRad(Degrees: Extended) Extended – градусту радианга өзгөртүү Radians := Degrees * PI / 180.

function GradToRad(Grads: Extended): Extended – градды радианга өзгөртүү Radians := Grads * PI / 200.

function RadToDeg(Radians: Extended) Extended – радианды градуска өзгөртүү Degrees := Radians * 180 / PI.

function RadToGrad(Radians: Extended) : Extended – радианды градга өзгөртүү Grads := Radians * PI / 200.

function RadToCycle(Radians: Extended) : Extended – радианды айланага өзгөртүү Cycles := Radians / 2PI.

Гиперболикалык функциялар

function ArcCosh(X: Extended): Extended – гиперболалык арккосинус

function ArcSinh(X: Extended): Extended – гиперболалык арксинус.

function ArcTanh(X: Extended): Extended – гиперболалык арктангенс.

function Cosh(X: Extended): Extended – гиперболалык косинус.
function Sinh(X: Extended): Extended – гиперболалык синус.
function Tanh(X: Extended): Extended – гиперболалык тангенс.

Логарифмалык функциялар
function LnXP1 (X: Extended) : Exteaded – (X+1) натуралдык логарифми (X нөлгө жакын болгондо пайдаланылат).
function Log10(const X: Extended): Extended – ондук логарифм.
function Log2(const X: Extended): Extended – экилик логарифм.
function LogN(Base, X: Extended): Extended – Base негизи боюнча X –тин логарифми.

Экспоненциалдык функциялар
function IntPower(Base: Extended; Expopient: Integer) : Extended – Base негизин Exponent бүтүн типтеги даражага көтөрүү.
function Power(Base, Exponent: Extended) : Extended – Base негизин Exponent чыныгы типтеги даражага көтөрүү.

Башка процедуралар жана функциялар
function Ceil(X: Extended): Integer – жакынкы кичине бүтүн сан.
function Floor (X: Extended): Integer – жакынкы чоң бүтүн сан.
procedure Frexp(X: Extended; var Mantissa: Extended; var Exponent: Integer) – чыныгы сандын мантиссасын жана даражасын кайтарат.
function Ldexp(X: Extended; P: Integer) : Extended – X * (2**P) туюнтмасынын маанисин кайтарат.

Статистикалык процедуралар жана функциялар
function Max(A,B: Int64): Int64,
function Max (A, B: Integer): Integer,
function Max(A,B: Single): Single,
function Max(A,B: Double): Double жана
function Max(A,B: Extended): Extended – эки сандын чоңун кайтарат (функциялар типтери боюнча айырмаланышат).
function Poly(X: Extended; const Coefficients: array of Double): Extended – Coefficients[0] + Coefficients[1]*X + ... + Coefficients[N]*(X**N) полиномунун мааниси, коэффициенттер даражасынын өсүү тартибинде берилет.
function MaxIntValue(const Data: array of Integer): Integer – бүтүн сандардын жыйындысынын максимумун кайтарат.
function MaxValue(const Data: array of Double): Double – чыныгы сандардын жыйындысынын максимумун кайтарат.
function Mean(const Data: array of Double): Extended – чыныгы сандардын жыйындысы үчүн арифметикалык орточосун эсептейт.

procedure MeanAndStdDev(const Data: array of Double; var Mean, StdDev: Extended) – чыныгы сандардын жыйындысы үчүн арифметикалык орточосун жана стандартуу четтөөнү эсептейт.

function Min(A,B: Integer): Integer,

function Min(A,B: Int64): Int64,

function Min(A,B: Single): Single,

function Min(A,B: Double): Double жана

function Min(A,B: Extended): Extended – эки сандын минимумун кайтарат (функциялар типтери боюнча айырмаланышат)..

function MinIntValue(const Data: array of Integer): Integer – бүтүн сандардын жыйындысынын минимумун кайтарат.

function MinValue(const Data: array of Double): Double – чыныгы сандардын жыйындысынын минимумун кайтарат

function Norm(const Data: array of Double): Extended – чыныгы сандардын нормасын (квадраттардын суммасынын квадраттык тамыры) кайтарат.

function StdDev(const Data: array of Double): Extended – чыныгы сандардын жыйындысы үчүн орточо квадраттык четтөөнү эсептейт.

function Sum(const Data: array of Double): Extended– чыныгы сандардын суммасын кайтарат.

procedure SumsAndSquares(const Data: array of Double; var Sum, SumOfSquares: Extended) – чыныгы сандардын жыйындысы үчүн сандардын суммасы жана квадраттарынын суммасын бир учурда эсептейт.

function Sumint(const Data: array of Integer): Integer– бүтүн сандардын жыйындысынын суммасы.

function SumOfSquares(const Data: array of Double): Extended - чыныгы сандардын жыйындысынын квадраттарынын суммасын эсептейт.

Финансылык функциялар

function DoubleDecliningBalance (Cost, Salvage: Extended; Life, Period: Integer): Extended – экилик баланс методу аркылуу амортизацияны эсептейт.

function FutureValue(Rate: Extended; NPeriods: Integer; Payment, PresentValue: Extended; PaymentTime: TPaymentTime): Extended – чегерүүнүн келечектеги мааниси.

function InterestPayment(Rate: Extended; Period, NPeriods: Integer; PresentValue, FutureValue: Extended; PaymentTime: TPaymentTime): Extended – ссуданын проценттин эсептөө.

function InterestRate(NPeriods: Integer; Payment, PresentValue, Future-Value: Extended; PaymentTime: TPaymentTime) : Extended – көрсөтүлгөн сумманы алуу үчүн зарыл болгон кирешенин нормасын эсептейт.

function InternalRateOfReturn (Guess: Extended; const CashFlows: array of Double): Extended – удаалаш төлөөлөрдүн катары үчүн чегерүүнүн айлануусунун ички тездигин эсептейт.

function NetPresentValue(Rate: Extended; const CashFlows: array of Double; PaymentTime: TPaymentTime): Extended – проценттик ставканын эсеби менен удаалаш төлөөлөрдүн катары үчүн чегерүүнүн таза баасын эсептейт.

function NumberOfPeriods(Rate, Payment, PresentValue, FutureValue: Extended; PaymentTime: TPaymentTime): Extended – чегерүү көрсөтүлгөн чондука жетүүчү мезгилдин санын эсептейт.

function Payment(Rate: Extended; NPeriods: Integer; PresentValue, FutureValue: Extended; PaymentTime: TPaymentTime): Extended – ссуданы көрсөтүлгөн мезгилде, проценттик ставкада, учурдагы жана келечектеги ссуданын маанисинде төлөп бүтүү үчүн мезгилдик төлөмдүн өлчөмү.

function PeriodPayment(Rate: Extended; Period, NPeriods: Integer; PresentValue, FutureValue: Extended; PaymentTime: TPaymentTime): Extended – көрсөтүлгөн мезгил үчүн проценттер боюнча төлөм.

function PresentValue(Rate: Extended; NPeriods: Integer; Payment, FutureValue: Extended; PaymentTime: TPaymentTime): Extended – чегерүүнүн учурдагы мааниси.

function SLNDepreciation (Cost, Salvage: Extended; Life: Integer): Extended – туруктуу норма методу боюнча амортизацияны эсептөө.

function SYDDepreciation (Cost, Salvage: Extended; Life, Period: Integer): Extended – амортизацияны салмактык коэффициенттер методу боюнча эсептөө.

7. TCanvas – классынын айрым методдору

procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer) – эллипистин бөлүгүн чийүү.

procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer) - хорданы чийүү. Эллипс менен түздүн кесилишинен алынган туюк фигура.

procedure Draw(X, Y: Integer; Graphic: TGraphic) – графикалык сүрөттү көрсөтүлгөн координаттар боюнча жайгаштыруу.

procedure Ellipse(X1, Y1, X2, Y2: Integer) – эллипсти чийүү.

procedure Ellipse(const Rect: TRect) – жогорудагы сыяктуу эле, айырмасы координаталар TRect тибиндеги массивде берилет.

procedure FillRect(const Rect: TRect) – Brush касиетине орнотулган түс менен боёлгон тик бурчтук чийүү.

procedure FloodFill(X, Y: Integer; Color: TColor; FillStyle: TFillStyle) – областы боё. Берилген чекиттин түсү менен дал келген түскө ээ болгон жанаша жаткан чекиттер аркылуу боё жүргүзүлөт.

procedure LineTo(X, Y: Integer) – PenPos касиетиндеги чекиттен методдун параметрде көрсөтүлгөн чекитке чейин түз чийүү.

procedure MoveTo(X, Y: Integer) – графикалык курсорду методдун параметрде көрсөтүлгөн чекитке жылдырат.

procedure Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer) – көрсөтүлгөн тик бурчтука жайгаштырылган эллипстин секторун чийүү.

procedure Polygon(Points: array of TPoint) – кесиндилердин жардамында түзүлгө татаал фигураны чийүү. Биринчи чекит акыркы чекит менен туташып туюк фигура түзүлөт.

procedure Rectangle(X1, Y1, X2, Y2: Integer) – тик бурчтук чийүү.

procedure RoundRect(X1, Y1, X2, Y2, X3, Y3: Integer) – бурчтары томпок болгон тик бурчтук чийүү.

procedure StretchDraw(const Rect: TRect; Graphic: TGraphic) – графикалык сүрөттү тик бурчтуу областтын өлчөмүнө ылайыктап жайгаштыруу.

function TextExtent(const Text: string): TSize – учурдагы шрифт менен жазылган тексттин бийиктигин жана узундугунун маанисин пикселде кайтарат. Мында TSize = record

cx: Longint;

cy: Longint;

function TextHeight(const Text: string): Integer – учурдагы шрифт менен жазылган тексттин бийиктигин пикселдерде кайтарат.

procedure TextOut(X, Y: Integer; const Text: string) – текстти берилген координаталардан баштап чыгаруу.

procedure TextRect(Rect: TRect; X, Y: Integer; const Text: string) – текстти берилген областка чыгаруу. Областтан чыгып кеткен тексттин бөлүгү көрсөтүлбөйт.

function TextWidth(const Text: string): Integer жана

function TextWidth(const Text: WideString): Integer - учурдагы шрифт менен жазылган тексттин узундугун пикселдерде кайтарат.

TRect тиби төмөнкүчө аныкталган:

```
type
TRect = packedrecord
    case Integer of
     0: (Left, Top, Right, Bottom: Integer);
     1: (TopLeft, BottomRight: TPoint);
     end;
```

8. Виртуалдык клавишалардын коддору

Код	Мааниси	Клавиша	Код	Мааниси	Клавиша
vk_Back	8	Backspace	vk_Tab	9	Tab
vk_Clear	12	[5]	vk_Return	13	Enter
vk_Shift	16	Shift	vk_Control	17	Cttl
vk_Menu	18	Alt	vk_Pause	19	Pause
vk_Capital	20	Caps Lock	vk_Escape	27	Esc
vk Space	32	Пробел	vk_Prior	33	Page Up

vk_Next	34	Page Down	vk_End	35	End
vk_Home	36	Home	vk_Left	37	Курсор солго
vk_Up	38	Курсор жогору	vk_Right	39	Курсор оңго
vk_Down	40	Курсор төмөн	vk_Insert	45	Insert
vk_Delete	46	Delete	vk_0..vk_9	48..57	0..9
vk_A..vk Z	65..90	A. .Z	vk_LWin	91	Сол Windows
vk_RWin	92	Оң Windows	vk_Numpad0.. vk_Numpad9	96..105	[0]..[9]
vk_Multiply	106	[*]	vk_Add	107	[+]
vk_Subtract	109	[-]	vk_Decimal	110	[Del]
vk_Divide	111	[/]	vk_Fl..vk_F12	112..123	Fl..F12
vk_Numlock	144	Num Lock	vk_Scroll	145	Scroll Lock

9. Windows операциялык системасындагы символдордун коддору (ANSI стандартынын Windows-1251 версиясы боюнча)

Кызматчы символдар

Код	Символ
9	Табуляция
11	Жаңы сап
13	Абзацтын аягы
32	Пробел

33-дөн 126-га чейинки символдор.

Код	Символ	Код	Символ	Код	Символ	Код	Символ
33	!	34	“	35	#	36	$
37	%	38	&	39	'	40	(
41	)	42	*	43	+	44	,
45	-	46	.	47	/	48	0
49	1	50	2	51	3	52	4
53	5	54	6	55	7	56	8
57	9	58	:	59	;	60	<
61	=	62	>	63	?	64	@
65	A	66	B	67	C	68	D
69	E	70	F	71	G	72	H
73	I	74	J	75	K	76	L

77	M	78	N	79	O	80	P
81	Q	82	R	83	S	84	T
85	U	86	V	87	W	88	X
89	Y	90	Z	91	[	92	\
93	]	94	^	95	_	96	'
97	a	98	b	99	c	100	d
101	e	102	f	103	g	104	h
105	i	106	j	107	k	108	i
109	m	110	n	111	o	112	p
113	q	114	r	115	s	116	t
117	u	118	v	119	w	120	x
121	y	122	z	123	{	124	|
125	}	126	~				

192-дөн 255-ге чейинки символдор.

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

Адабияттар

1. Бобровский С.И. Delphi 7. Учебный курс. – СПб.: Питер, 2003. -736 с.
2. Гофман В.Э., Хомоненко А.Д. Delphi. Быстрый старт. - СПб.: БХВ-Петербург, 2003.-288 с.
3. Гофман В.Э., Хомоненко А.Д. Delphi: Основы работы в среде Delphi; Приемы создания приложений; Разработка и использование баз данных; Примеры программ . - СПб.: БХВ-Петербург, 2003.-288 с.
4. Дарахвелидзе П. Г., Марков Е.П. Программирование в Delphi 7. .-СПб.: БХВ-Петербург, 2004.-784 с.
5. Ревич Ю.В. Нестандартные приемы программирования на Delphi - СПб.: БХВ-Петербург, 2003.- 560 с.
6. Фаронов В.В. Delphi. Программирование на языке высокого уровня: Учебник для вузов.– СПб.: Питер, 2004. - 675 с.
7. Фаронов В.В. Delphi 2005. Язык, среда, разработка приложений.- СПб.: Питер, 2005. -560 с.
8. Фленов М.Е. Библия Delphi.- - СПб.: БХВ-Петербург, 2004.-880 с.
9. Фленов М.Е. Delphi в шутку и всерьез: Что умеют хакеры.– СПб.: Питер, 2005. - 272 с.
10. Фленов М.Е. Delphi 2005. Секреты программирования – СПб.: Питер, 2005. - 272 с.

Murzacmatova Nurgul.

Сдано в набор 20.03.2006 г.
Подписано к печати 03.04.2006 г.
Заказ: № 426. Тираж 100 экз.
Отпечатано в ОсОО "ДИП Полиграфия"
г. Ош, ул. Ломоносова, 6. Тел.: 7-47-07.