Ўзбекистон Республикаси олий ва ўрта махсус таълим вазирлиги

Бухоро озиқ-овқат ва енгил саноат технологияси институти

“КАСБИЙ ТАЪЛИМ” ФАКУЛТЕТИ

“Информатика ва ахборот технологиялари” кафедраси

“Объектга мўлжалланган дастурлаш” фанидан амалий машғулотлар учун

УСЛУБИЙ КЎРСАТМА

Бухоро – 2010 й.

Услубий кўрсатма муаллифи: «Информатика ва ахборот технологиялари» кафедраси к. ўқ. Нарзиев У.З.

Услубий қўлланма _-сон «Информатика ва ахборот технологиялари» кафедраси мажлисида (_.__.10.) тасдикланди.

Услубий қўлланма Бухоро озик овкат ва енгил саноат технологияси институти «Касбий таълим» факултети «МИАТ» мутахассислигида «Объектга йўналтирилган дастурлаш» фанидан Амалий машғулотларни ўтказиш учун мўлжалланган.

Мундарижа

Амалий машғулот №1. Дастурда фойдаланувчи интерфейсини яратиш.

Интерфейслар компьютердаги дастурда ишлаш учун фойдаланувчига кенг ёрдам берувчи восита ҳисобланади. Унинг асосий вазифаси формада тасвирланган объектларнинг тузилиши, йўналиши ва киритиладиган ҳамда олинадиган қийматлар ўрнини ва соҳасини тушунтиришдир.

Интерфейс хусусий ҳолда типларни тасвирлашга ўхшайди. Уни interface сўзи ёрдамида фаоллаштириш мумкин. Масалан:

```
type
  IEdit = interface
    procedure Copy; stdcall;
    procedure Cut; stdcall;
    procedure Paste; stdcall;
    function Undo: Boolean; stdcall;
  end;
```

Бундай эълонлар мавҳум типлаш эълонлари каби ўзида ишлатилган хусусият ва услубларни тасвирлаш шарт эмас.

Агар бирор бир синф интерфейсни қўлласа бу интерфейс номи синфни эълон қилиш сарлавҳасида ёзилади:

```
TEditor = class(TInterfacedObject, IEdit)
  procedure Copy; stdcall;
  procedure Cut; stdcall;
  procedure Paste; stdcall;
  function Undo: Boolean; stdcall;
end;
```

Оддий синфдан форқли равишда интерфейсли синф биттадан кўп юқори интерфейсига эга бўлиши мумкин:

```
type
  IMylnterface = interface procedure Delete; stdcall;
```

```
  end;
  TMyEditor = class(TInterfacedObiect, lEdit, IMylnterface)
    procedure Copy; stdcall;
    procedure Cut; stdcall;
    procedure Paste; stdcall;
    function Undo: , Boolean; stdcall;
    procedure Delete; stdcall;
  end;
```

Ихтиёрий ҳолатда ҳам интерфейс синфини фаоллаштириш учун мос келувчи интерфейс модулларини эълон қилиш керак. Агар, масалан интерфейс

```
IPaint = interface
  procedure CirclePaint(Canva: TCanvas; X, Y, R: Integer);
  procedure RectPaint(Canva: TCanvas; X1, Y1, X2, Y2: Integer);
end;
```

каби эълон қилинса, уни ишлатувчи интерфейс синфи қуйидагича бўлади:

```
TPainter = class(TInterfacedObject, IPaint)
  procedure CirclePaint(Canva: TCanvas; X, Y, R: Integers);
  procedure RectPaint(Canva: TCanvas; X1, Y1, X2, Y2: Integer);
end;
```

implementation бўлимида эса, услубларни кўрсатиш керак:

```
procedure TPainter.CirclePaint(Canva: TCanvas; X, Y, R: Integers;
begin
  with Canva do
    Ellipse(X, Y, X + 2 * R, Y + 2 * R);
end;


procedure TPainter.RectPaint(Canva: TCanvas; X1, Y1, X2, Y2: Integer);
begin
  with Canva do
    Rectangle(XI, Yl, X2, Y2)
end;
```

Энди синфнинг интерфейсли объекти Трainterни эълон қилиш мумкин:

```
procedure TForm1.PaintBoxIPaint(Sender: TObject);
var
  Painter: IPaint;
begin
  Painter := TPainter.Create;
  Painter.CirclePaint(PaintBoxl.Canvas, 10, 0, 10);
  Painter.RectPaint(PaintBoxl.Canvas, 40, 0, 60, 20);
end;
```

Амалий машғулот №2. График тасвирлар билан ишлаш.

График чоп қилиш (TPrinter объекти)

Қандай қилиб, Delphiда яратилган дастурдан график маълумотларни принтерга узатиш мумкин? Бунинг учун махсус (TPrinter синфидаги) Printer объекти мавжуд. У дастурга Printers модулини киритгандан сўнг фаоллашади. Бу объект ёрдамида график маълумотларни принтерга узатиш худди уларни экранга чиқаришдай оддийлашиб қолади. Printer объектининг хусусиятлари ва услубларини кўриб ўтамиз.

Printer хусусиятлари

Aborted – Булев (мантиқий) тип; фойдаланувчи принтер ишини *Abort.*услуби ёрдамида тўхтатганлигини билдиради.

Canvas - Канва, график маълумотларни чиқариш учун жой.

Fonts – Ўрнатиш мумкин бўлган шрифтлар рўйхати.

Handle - Windows API тўғридан тўғри чақирилганда ишлатилади.

Orientation – Саҳифанинг мўлжали, вертикал ёки горизонтал.

PageWidth, *PageHeight*, *PageNumber* – мос ҳолда саҳифанинг кенглиги, баландлиги ва тартиб рақами.

Printers Тизимда ўрнатилган барча принтерларнинг номлари рўйхати

PrinterIndex Жорий принтернинг рақами. ЖҚБ чоп қилиш учун бу ерда -1 туриши керак.

Printing – Чоп қилиш (*BeginDoc* услубида*)* бошланганлигини билдирувчи мантиқий тип.

Title – Print учун сарлавҳа.

Printer услублари:

Abort - *BeginDoc* режимида бошланган чоп қилишни тўхтатиш

BeginDoc - Канвада чизиш бошлангандан олдин чақирилади.

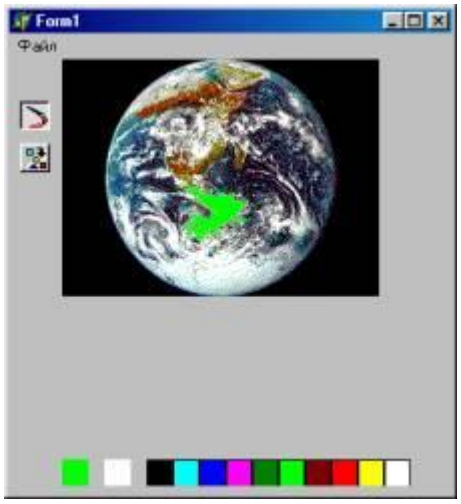
EndDoc – Канвада ҳамма нарса тайёр бўлгач принтер чоп қилишни бошалайди.

NewPage - Янги саҳифага ўтиш.

Қолган услублар билан оддий ҳолларда фойдаланиш керак эмас.

Шундай қилиб, график маълумотларни чоп қилиш тартиби қуйидагича:

- *BeginDoc* услубини бажарамиз
- канвада (Canvas) керакли нарсаларни чизамиз
- маълумотни бир нечта саҳифага жойлаш керак бўлса, *NewPage* услубини қўллаш чизилганларни принтерга йўллаб, *EndDoc* услубини бажаримиз.



SpeedButton: SBBrush, SBColor;

GroupIndex := 1;

AllowAllUp := true;

Glyph := ..\Images\Butons\brush.bmp;

Glyph := ..\Images\Butons\one2one.bmp;

Алгоритм

1. Формани тўлдириш;

2. **var**

Bitmap: TBitMap;

3. Form OnCreate;

4. Form OnDestroy;

5. MOpenClick;

6. UndoClick;

7. SBBrushClick и SBColor(тасвирнинг жорий кўринишини сақлаш);

8. Image3MouseDown ва Image4га MouseDownни нусхалаш;

unit UGraphEdit;

```
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, Buttons, ExtCtrls, Menus, ExtDlgs;

type
  TForm1 = class(TForm)
    Image1: TImage;
    Image2: TImage;
    Image3: TImage;
    Image4: TImage;
    SBBrush: TSpeedButton;
    SBColor: TSpeedButton;
    OpenPictureDialog1: TOpenPictureDialog;
    MainMenu1: TMainMenu;
    N1: TMenuItem;
    MOpen: TMenuItem;
    N2: TMenuItem;
    Undo: TMenuItem;
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure MOpenClick(Sender: TObject);
    procedure UndoClick(Sender: TObject);
    procedure SBBrushClick(Sender: TObject);
    procedure Image3MouseDown(Sender: TObject; Button: TMouseButton; Shift:
      TShiftState; X, Y: Integer);
  private { Private declarations }
  public { Public declarations }
  end;
var
  Form1: TForm1;
```

```
implementation
{$R *.DFM}
var
  BitMap: TBitMap;
    //тасвирни сақлаш учун ўзгарувчи

procedure
  TForm1.FormCreate(Sender: TObject);
var
  HW, I: integer;
begin
  BitMap := TBitMap.Create;
  {Асосий ва ёрдамчи ранглар бўёғинитанлаш }
  Image1.Canvas.Brush.Color := clBlack;
  Image2.Canvas.Brush.Color := clWhite;
  { Асосий ва ёрдамчи ранглар ойналарини тўлдириш}
  with
    Image1.Canvas do
    FillRect(Rect(0, 0, Width, Height));
  with Image2.Canvas do
    FillRect(Rect(0, 0, Width, Height));
  {Элемент кенглигини ўрнатиш }
  HW := Image4.Width
    div 10;
  {элементни бўяш }
  with
    Image4.Canvas do
    for I := 1 to 10 do
    begin
      case I of
```

```
      1: Brush.Color := clBlack;
      2: Brush.Color := clAqua;
      3: Brush.Color := clBlue;
      4: Brush.Color := clFuchsia;
      5: Brush.Color := clGreen;
      6: Brush.Color := clLime;
      7: Brush.Color := clMaroon;
      8: Brush.Color := clRed;
      9: Brush.Color := clYellow;
      10: Brush.Color := clWhite;
    end;
    Rectangle((I - 1) * HW, 0, I * HW, Height);
  end;
```


```
with Image3 do
begin
  Canvas.MoveTo(0, 0);
  Canvas.LineTo(Width, Height);
  Canvas.MoveTo(0, Height);
  Canvas.LineTo(Width, 0);
end;
BitMap.Assign(Image3.Picture);
end;

procedure TForm1.FormDestroy(Sender:
```

```
  TObject);
begin
  BitMap.Free;
end;

procedure TForm1.MOpenClick(Sender:
  TObject);
begin
  if OpenPictureDialog1.Execute then
  begin
    Image3.Picture.LoadFromFile(OpenPictureDialog1.FileName);
    BitMap.Assign(Image3.Picture);
  end;
end;

procedure TForm1.UndoClick(Sender:
  TObject);
begin
  Image3.Picture.Assign(BitMap);
end;

procedure
  TForm1.SBBrushClick(Sender: TObject);
begin
  if (Sender as TSpeedButton).Down then
    BitMap.Assign(Image3.Picture);
end;

procedure
  TForm1.Image3MouseDown(Sender: TObject; Button: TMouseButton; Shift:
  TShiftState; X, Y: Integer);
```

```
begin
 if (Sender = Image4) or SBColor.Down then
  { Асосий ва ёрдамчи рангларни ўрнатиш режими}
 begin
  if (Button = mbLeft) then
   with Image1.Canvas do
   begin
    { Асосий рангни ўрнатиш }
    Brush.Color := (Sender as
     TImage).Canvas.Pixels[X, Y];
    FillRect(Rect(0, 0, Width, Height));
   end
  else
   with Image2.Canvas do
   begin
    { Ёрдамчи рангни ўрнатиш }
    Brush.Color := (Sender as
     TImage).Canvas.Pixels[X, Y];
    FillRect(Rect(0, 0, Width, Height));
   end;
 end
 else if SBBrush.Down then
  with Image3.Canvas do
  begin
   if
    Button = mbLeft then
    Brush.Color := Image1.Canvas.Brush.Color
   else
    Brush.Color := Image2.Canvas.Brush.Color;
   FloodFill(X, Y, Pixels[X, Y], fsSurface);
  end;
```

```
end;
end.
```

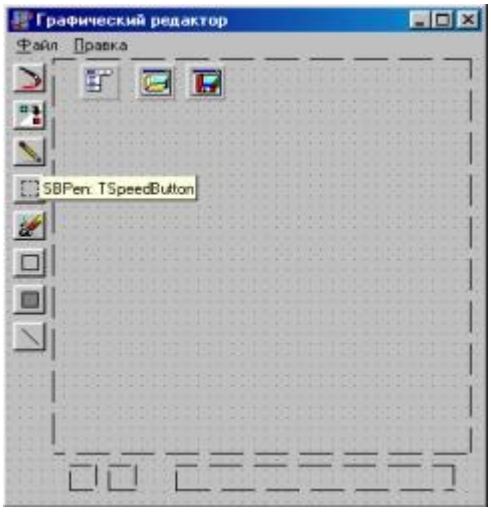
Охирги амалларни кўриб ўтамиз:

```
with Image3.Canvas do
begin
  MoveTo(X0, Y0); //олдинги чизиқларни ўчириш;
  LineTo(X1, Y1);
  Pen.Mode := pmCopy; //янги чизиқлар чизиш;
  MoveTo(X0, Y0);
  LineTo(X, Y);
end;
```

Файлни сақлаш SavePictureDialog компоненти ёрдамида бажарилади:

```
procedure TForm1.MSaveClick(Sender: TObject);
begin
  if SavePictureDialog1.Execute then
  begin
    BitMap.Assign(Image3.Picture); //тасвирни сақлаш;
    BitMap.SaveToFile(SavePictureDialog1.FileName); //файлда сақлаш;
  end;
end;
```

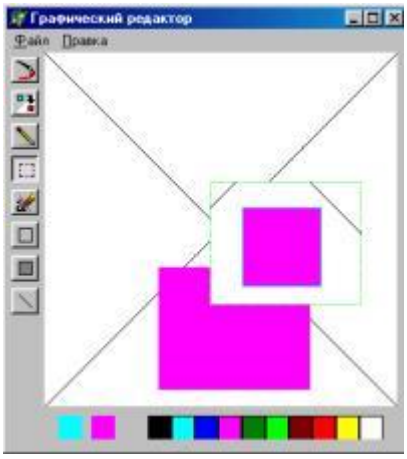
Қуйида муҳаррир дастурининг тўлиқ матни келтирилган:



TForm1 синфига қўшамиз:

```
TForm1 = class(TForm)
```

```
MFile: TMenuItem;
SBRect: TSpeedButton;
SBRectang: TSpeedButton;
SBFillRec: TSpeedButton;
SBErase: TSpeedButton;
SBPen: TSpeedButton;
SBLine: TSpeedButton;
MSave: TMenuItem;
MCut: TMenuItem;
MCopy: TMenuItem;
MPaste: TMenuItem;
SavePictureDialog1: TSavePictureDialog;
procedure Image3MouseDown(Sender: TObject; Button:
  TMouseButton; Shift: TShiftState; X, Y: Integer);.
procedure SBBrushClick(Sender: TObject);
procedure Image3MouseMove(Sender: TObject; Shift:
  TShiftState; X, Y: Integer);
```




```
procedure Image3MouseUp(Sender: TObject; Button:
 TMouseButton; Shift: TShiftState; X, Y: Integer);
 procedure MOpenClick(Sender: TObject);
   procedure MCopyClick(Sender: TObject);
     procedure MPasteClick(Sender: TObject);
       procedure MSaveClick(Sender: TObject);
         ……..
```

```
end;
implementation
{$R *.DFM}
var
  BitMap, BMCopy: TBitMap;
  R, R0: TRect;
  X0, Y0, X1, Y1: longint;
const
  RBegin: boolean = false;
  REnd: boolean = false; //
  RDrag: boolean = false; //байроқ
  перетаскивания

  procedure
  TForm1.Image3MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  if (Sender = Image4) or SBColor.Down then
  begin
    if (Button = mbLeft) then
      with Image1.Canvas do
      begin
        Brush.Color := (Sender as
          TImage).Canvas.Pixels[X, Y];
        FillRect(Rect(0, 0, Width, Height));
      end
    else
      with Image2.Canvas do
      begin
        Brush.Color := (Sender as TI //mage).Canvas.Pixels[X,Y];
          FillRect(Rect(0, 0, Width, Height));
```

```
    end;
end
else
  with Image3.Canvas do
  begin
    X0 := X;
    Y0 := Y;
    if SBPen.Down then
    begin
      MoveTo(X, Y);
      Pen.Color := Image1.Canvas.Brush.Color;
    end
    else if SBLine.Down then
    begin
      X1 := X;
      Y1 := Y;
      Pen.Mode := pmNotXor;
      Pen.Color := Image1.Canvas.Brush.Color;
    end
    else if SBBrush.Down then
    begin
      if
        Button = mbLeft then
        Brush.Color := Image1.Canvas.Brush.Color
      else
        Brush.Color := Image2.Canvas.Brush.Color;
      FloodFill(X, Y, Pixels[X, Y], fsSurface);
    end
    else if SBErase.Down then
    begin
      R := Rect(X - 6, Y - 6, X + 6, Y + 6);
```

```
      DrawFocusRect(R);
      Brush.Color := Image2.Canvas.Brush.Color;
      FillRect(Rect(X - 5, Y - 5, X + 5, Y + 5));
    end
    else if SBRect.Down or SBRectang.Down or
      SBFillRec.Downthen
    begin
    if REnd then
    begin
      DrawFocusRect(R);
      if (X < R.Right) and (X > R.Left) and (Y > R.Top) and
        (Y < R.Bottom)
then
      begin
        RDrag := true;
        REnd := false;
        R0.TopLeft := R.TopLeft;
        R0.BottomRight := R.BottomRight;
        BitMap.Assign(Image3.Picture);
        Brush.Color := Image2.Canvas.Brush.Color;
        MCopy.Enabled := false;
        MCut.Enabled := false;
      end;
    end
    else
    begin
      RBegin := true;
      REnd := false;
      R.TopLeft := Point(X, Y);
      R.BottomRight := Point(X, Y);
      DrawFocusRect(R);
```

```
    end;
  end;
  end;
end;

procedure
 TForm1.SBBrushClick(Sender: TObject);
begin
 if (Sender as TSpeedButton).Down then
  BitMap.Assign(Image3.Picture);
 RBegin := false;
 RDrag := false;
 REnd := false;
end;

procedure
 TForm1.Image3MouseMove(Sender: TObject; Shift: TShiftState; X, Y:
 Integer);
begin
 if not (ssLeft in Shift) then
  exit;
 if SBLine.Down then
  with
   Image3.Canvas do
  begin
   MoveTo(X0, Y0);
   LineTo(X1, Y1);
   MoveTo(X0, Y0);
   LineTo(X, Y);
   X1 := X;
   Y1 := Y;
```

```
  end
else if SBPen.Down then
  Image3.Canvas.LineTo(X, Y)
else if SBErase.Down then
  with Image3.Canvas do
  begin
    {режим ластика}
    DrawFocusRect(R);
    R := Rect(X - 6, Y - 6, X + 6, Y + 6);
    DrawFocusRect(R);
    FillRect(Rect(X - 5, Y - 5, X + 5, Y + 5));
  end
else if (SBRect.Down and (RBegin or RDrag)) or
  SBRectang.Down or SBFillRec.Down then
  with Image3.Canvas do
  begin
    if RBegin then
    begin
      DrawFocusRect(R);
      if X0 < X then
      begin
        R.Left := X0;
        R.Right := X
      end
      else
      begin
        R.Left := X;
        R.Right := X0
      end;
      if Y0 < Y then
      begin
```

```
        R.Top := Y0;
        R.Bottom := Y
      end
      else
      begin
        R.Top := Y;
        R.Bottom := Y0
      end;
      DrawFocusRect(R);
    end
    else if SBRect.Down then
    begin
      CopyRect(R, BitMap.Canvas, R);
      if not (ssCtrl in Shift) then
        FillRect(R0);
      R.Left := R.Left + X - X0;
      R.Right := R.Right + X - X0;
      R.Top := R.Top + Y - Y0;
      R.Bottom := R.Bottom + Y - Y0;
      X0 := X;
      Y0 := Y;
      CopyRect(R, BitMap.Canvas, R0);
      DrawFocusRect(R);
    end;
  end;
end;

procedure
  TForm1.Image3MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
```

```
with Image3.Canvas do
begin
 if SBLine.Down then
 begin
  MoveTo(X0, Y0);
  LineTo(X1, Y1);
  Pen.Mode := pmCopy;
  MoveTo(X0, Y0);
  LineTo(X, Y);
 end
 else if SBRect.Down then
 begin
  if RDrag then
    DrawFocusRect(R);
  if RBegin and not REndthen
  begin
    REnd := true;
    MCopy.Enabled := true;
    MCut.Enabled := true;
  end
 end
 else if SBRectang.Down then
 begin
  Brush.Color := Image1.Canvas.Brush.Color;
  FrameRect(R);
 end
 else if SBFillRec.Down then
 begin
  Brush.Color := Image2.Canvas.Brush.Color;
  Pen.Color := Image1.Canvas.Brush.Color;
  Rectangle(R.Left, R.Top, R.Right, R.Bottom);
```

```
    end
    else if
      SBErase.Downthen
      Image3.Canvas.DrawFocusRect(R);
    RBegin := false;
    RDrag := false;
  end;
end;

procedure
  TForm1.MCopyClick(Sender: TObject);
{var
MyFormat: Word;
AData: THandle;
APalette: HPALETTE;}
begin
  Image3.Canvas.DrawFocusRect(R);
  BMCopy := BitMap.Create;
  BMCopy.Width := R.Right - R.Left;
  BMCopy.Height := R.Bottom - R.Top;
  try
    BMCopy.Canvas.Copyrect(Rect(0, 0, BMCopy.Width, BMCopy.Height),
      Image3.Canvas, R);
    Image3.Canvas.DrawFocusRect(R);
    {BMCopy.SaveToClipBoardFormat(MyFormat,AData,APalette);
    ClipBoard.SetAsHandle(MyFormat,AData);}
    ClipBoard.Assign(BMCopy);
    if (Sender as TMenuItem).Name = 'MCut' then
    begin
      Image3.Canvas.Brush.Color := clWhite;
      Image3.Canvas.FillRect(R);
```

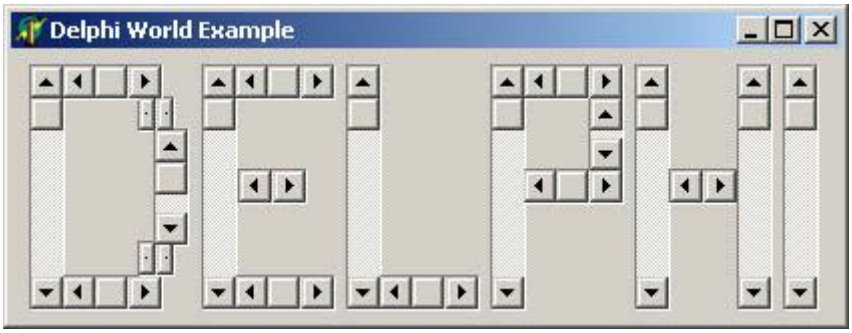
```
    end;
  finally
    BMCopy.Free;
  end;
end;

procedure
  TForm1.MPasteClick(Sender: TObject);
begin
  BMCopy := BitMap.Create;
  try
    try
      BMCopy.LoadFromClipBoardFormat(cf_BitMap,
        ClipBoard.GetAsHandle(cf_Bitmap), 0);
      Image3.Canvas.CopyRect(Rect(0, 0, BMCopy.Width, BMCopy.Height),
        BMCopy.Canvas, Rect(0, 0, BMCopy.Width, BMCopy.Height));
    finally
      BMCopy.Free;
    end;
  except
    on EInvalidGraphic do
      ShowMessage('Нотўғри формат`);
  end;
end;

procedure
  TForm1.MSaveClick(Sender: TObject);
begin
  if SavePictureDialog1.Execute then
  begin
    BitMap.Assign(Image3.Picture);
```

```
   BitMap.SaveToFile(SavePictureDialog1.FileName);
  end;
end;
end.
```

Форма график объект сифатида




```
uses clipbrd;


procedure TShowVRML.Kopieren1Click(Sender: TObject);
var
  bitmap: tbitmap;
begin
  bitmap := tbitmap.create;
  bitmap.width := clientwidth;
  bitmap.height := clientheight;
  try
    with bitmap.Canvas do
      CopyRect(clientrect, canvas, clientrect);
    clipboard.assign(bitmap);
  finally
    bitmap.free;
  end;
```

```
end;
```

Амалий машғулот №3. Мультимедиа дастурлар яратиш

Delphiда мультимедиа

Delphiда TMediaPlayer компоненти мавжуд бўлиб, у сизга мултимедиа дастурларини тузишда ҳамма имкониятларни яратиб беради. Ушбу компонентдан фойдаланиш жуда оддий.

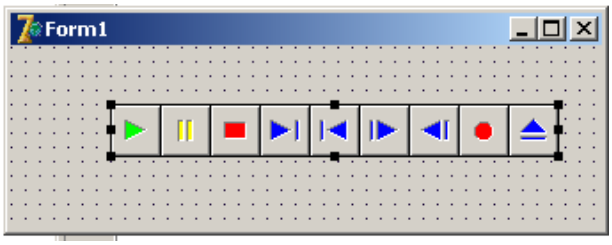
Бу оддийликни икки маънода тушуниш мумкин:

- Бир томондан – бу ихтиёрий дастурчига мультимедиа дастурини тузишга имкон яратилган.
- Иккинчи томондан, бу компонентда мультимедиа файллари билан бажариладиган ҳамма асосий амаллар киритилган, лекин сизга қуйи функциялар керак бўладиган бўлса, Delphi дастурлаш тили имкониятларидан фойдалинишингиз мумкин.

Ушбу маърузада компонент ишлаганда унинг ички функциялари қандай ишлатилишига тўхталмаймиз.

TMediaPlayer компоненти

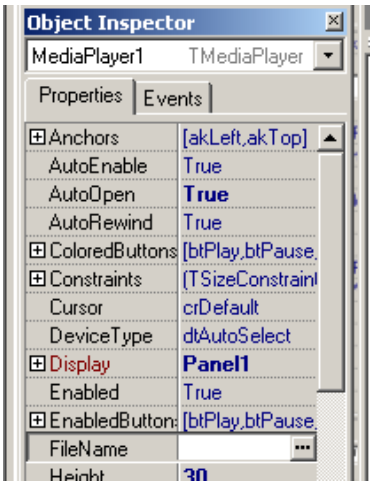
Дастлаб янги лоиҳа яратамиз ва формага TMediaPlayer (System саҳифаси) компонентини ташлаймиз.



15-расм: Формадаги TMediaPlayer компоненти.

TMediaPlayer компоненти қурилмани бошқариш панели каби тузилган. Магнитофонлардаги каби бу ерда ҳам “ишга тушириш”, “ўтказиш”, “ёзиш” ва бошқа тугмалари бор.

Компонентни формага ташлагач, сиз Объектлар Инспекторида "FileName" хусусиятини кўрасиз.



16- Расм. TMediaPlayerнинг Объектлар инспекторидаги хусусиятлари

Шу ерда сичқонча тугмасини босинг ва рўйхатдан AVI, WAV ёки MID кенгайтмали файлни танланг. Сўнгра AutoOpen хусусиятини True қийматга ўрнатиш керак.

Бу қадамлар бажарилгач, дастур юкланишга тайёр бўлади. Дастурни юклаб ишга тушириш (Play) тугмасини босинг ва сиз танлаган видеоклипингизни кўришингиз ёки мусиқани тинглашингиз мумкин. Агар овоз ёки видеоклип ишга тушмасдан хатолик ҳақида хабар чиқарилса, икки ҳолат бўлиши мумкин:

1. Сиз файл номини ёки унгача бўлган йўлни нотўғри киритгансиз.
2. Компьютерингиздаги мультимедиа қурилмалари Windowsга тўғри созланмаган. Бу эса, сизда керакли қурилманинг йўқлигидан ёки керакли драйверлар ўрнатилмаганлигидан далолат беради.

TMediaPlayer компонентининг яна бир муҳим хусусияти - Display. Дастлабки мисолимизда бу хусусият бўш қолдирилган ва видеоклип алохида ойнада кўринган эди. Лекин, видеоклип намойишида экран сифатида масалан, **Панел**дан фойдаланиш мумкин. Формага Tpanel компонентини ташланг ва Caption хусусиятидаги матнни ўчиринг. Сўнгра TMediaPlayer учун Display хусусиятидаги рўйхатдан Panel1ни танланг. Шундан сўнг дастурни юклаб, ишга тушириш тугмасини босиш керак.

```
function waveInGetNumDevs: UINT; stdcall;
```

waveInGetDevCaps функцияси қурилмалар характеристикаларини олишда ёрдам беради.

```
function waveInGetDevCaps(
  hwi: HWAVEIN;
  lpCaps: PWaveInCaps;
  uSize: UINT
  ): MMRESULT; stdcall;
```

Бу ерда:

hwi - waveInOpen функцияси томонидан очилган қурилма рақами;

lpCaps – TWAVEINCAPS структура манзили;

uSize - TWAVEINCAPS структурасининг байтдаги ҳажми.

```
type
  TWaveInCaps = record
    wMid: Word;
    wPid: Word;
    vDriverVersion: MMVERSION;
    szPname: array[0..MAXPNAMELEN - 1] of AnsiChar;
    dwFormats: DWORD;
    wChannels: Word;
    wReserved1: Word;
  end;
```

```
WAVE_FORMAT_1M08 11.025 kHz, mono, 8-bit
WAVE_FORMAT_1M16 11.025 kHz, mono, 16-bit
WAVE_FORMAT_1S08 11.025 kHz, stereo, 8-bit
WAVE_FORMAT_1S16 11.025 kHz, stereo, 16-bit
WAVE_FORMAT_2M08 22.05 kHz, mono, 8-bit
WAVE_FORMAT_2M16 22.05 kHz, mono, 16-bit
WAVE_FORMAT_2S08 22.05 kHz, stereo, 8-bit
WAVE_FORMAT_2S16 22.05 kHz, stereo, 16-bit
```

```
WAVE_FORMAT_4M08 44.1 kHz, mono, 8-bit
WAVE_FORMAT_4M16 44.1 kHz, mono, 16-bit
WAVE_FORMAT_4S08 44.1 kHz, stereo, 8-bit
WAVE_FORMAT_4S16 44.1 kHz, stereo, 16-bit
```

Қарийиб ҳамма овоз хариталари ёзиш ўқиш режимлари оралиғини қўллашига этиборни қаратинг.

wChannels: Word – кириш каналлари сони (1-моно, 2-стерео)
wReserved1: Word - заҳирада

Функция waveInGetErrorText бажариш вақтида юзага келувчи хатоликларни матн кўринишида қайтаради.

```
function waveInGetErrorText(
  mmrError: MMRESULT;
  lpText: PChar;
  uSize: UINT
  ): MMRESULT; stdcall;
```

mmrError – хатолик коди;

Қуйида аудиокириш қурилмалари ҳақида маълумот чиқарувчи процедурага мисол келтирилган.

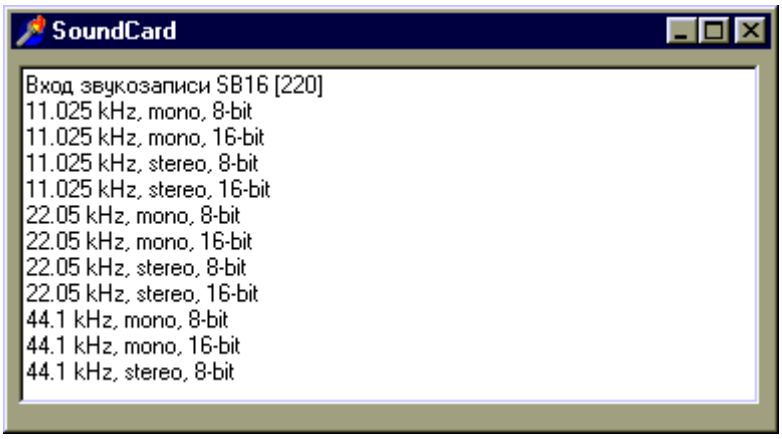
```
uses Windows, MMSystem;
type
  TModeDescr = record
    mode: DWORD; // иш режими коди
    descr: string[32];
  end;

const
  modes: array[1..12] of TModeDescr = ((mode: WAVE_FORMAT_1M08; descr:
```

```
   '11.025 kHz, mono, 8-bit'),
   (mode: WAVE_FORMAT_1M16; descr: '11.025 kHz, mono, 16-bit'),
   (mode: WAVE_FORMAT_1S08; descr: '11.025 kHz, stereo, 8-bit'),
   (mode: WAVE_FORMAT_1S16; descr: '11.025 kHz, stereo, 16-bit'),
   (mode: WAVE_FORMAT_2M08; descr: '22.05 kHz, mono, 8-bit'),
   (mode: WAVE_FORMAT_2M16; descr: '22.05 kHz, mono, 16-bit'),
   (mode: WAVE_FORMAT_2S08; descr: '22.05 kHz, stereo, 8-bit'),
   (mode: WAVE_FORMAT_2S16; descr: '22.05 kHz, stereo, 16-bit'),
   (mode: WAVE_FORMAT_4M08; descr: '44.1 kHz, mono, 8-bit'),
   (mode: WAVE_FORMAT_4M16; descr: '44.1 kHz, mono, 16-bit'),
   (mode: WAVE_FORMAT_4S08; descr: '44.1 kHz, stereo, 8-bit'),
   (mode: WAVE_FORMAT_4S16; descr: '44.1 kHz, stereo, 16-bit'));

procedure ShowInfo;
var
  WaveNums, i, j: integer;
  WaveInCaps: TWaveInCaps;
    // қурилма ҳақидаги маълумот
begin
  WaveNums := waveInGetNumDevs;
  if WaveNums > 0 then // агар тизимда аудио қурилмалар бўлса
  begin
    for i := 0 to WaveNums - 1 do
      // ҳамма мавжуд қурилмалар хусусиятларини оламиз
    begin
      waveInGetDevCaps(i, @WaveInCaps, sizeof(TWaveInCaps));
      // қурилма номини киритамиз
      MainForm.Memo.Lines.Add(PChar(@WaveInCaps.szPname));
      for j := 1 to High(modes) do
      begin
```

```
      // мос келувчи иш режимлари ва қурилмалар рўйхатини чиқарамиз
      if (modes[j].mode and WaveInCaps.dwFormats) = modes[j].mode then
        Memo.Lines.Add(modes[j].descr);
    end;
  end;
 end;
end;
```



Расм 1. ShowInfo процедураси ёрдамида чиқарилувчи маълумотлар.

waveInOpen функцияси мавжуд Waveform audio киритиш қурилмасини сигналнисонли қилиш учун очади.

```
function waveInOpen(
 lphWaveIn: PHWAVEIN;
 uDeviceID: UINT;
 lpFormatEx: PWaveFormatEx;
 dwCallback,
 dwInstance,
 dwFlags: DWORD
 ): MMRESULT; stdcall;
```

Бу ерда

lphWaveIn - Waveform audio қурилмасининг аниқловчисига кўрсаткич.

uDeviceID – очилувчи қурилма рақами.

lpFormatEx - TWaveFormatEx типидаги структурага кўрсаткич

```
type
  TWaveFormatEx = packed record
    wFormatTag: Word; { format type }
    nChannels: Word; { number of channels (i.e. mono, stereo, etc.) }
    nSamplesPerSec: DWORD; { sample rate }
    nAvgBytesPerSec: DWORD; { for buffer estimation }
    nBlockAlign: Word; { block size of data }
    wBitsPerSample: Word; { number of bits per sample of mono data }
    cbSize: Word; { the count in bytes of the size of }
  end;
```

В случае использование Callback процедуры она имеет следующий вид:

```
procedure waveInProc(hwi: HWAVEIN; uMsg, dwInstance,
  dwParam1, dwParam2: DWORD); stdcall;
begin
  //амал
end;
```

WIM_CLOSE қурилма waveInClose функцияси ёрдамида ёпилганда бажарилади; WIM_OPEN қурилма waveInOpen функцияси ёрдамида ёпилганда бажарилади; waveInPrepareHeader функцияси буферни маълумотлар киритилиши учун тайёрлайди:

```
function waveInPrepareHeader(
  hWaveIn: HWAVEIN;
```

```
 lpWaveInHdr: PWaveHdr;
 uSize: UINT
 ): MMRESULT; stdcall;
```

Бу ерда :

hWaveIn – очиқ қурилма аниқловчиси;

lpWaveInHdr – WaveHdr структураси манзили:

```
type
 TWaveHdr = record
  lpData: PChar; { pointer to locked data buffer }
  dwBufferLength: DWORD; { length of data buffer }
  dwBytesRecorded: DWORD; { used for input only }
  dwUser: DWORD; { for client's use }
  dwFlags: DWORD; { assorted flags}
  dwLoops: DWORD; { loop control counter }
  lpNext: PWaveHdr; { reserved for driver }
  reserved: DWORD; { reserved for driver }
 end;
```

waveInAddBuffer функцияси буферга маълумотлар киритилиши навбатини тайёрлайди. Буфер тўлганда тизимга бу ҳақда хабар чиқарилади.

```
function waveInAddBuffer(
 hWaveIn: HWAVEIN;
 lpWaveInHdr: PWaveHdr;
 uSize: UINT
 ): MMRESULT; stdcall;
```

Бу ерда:

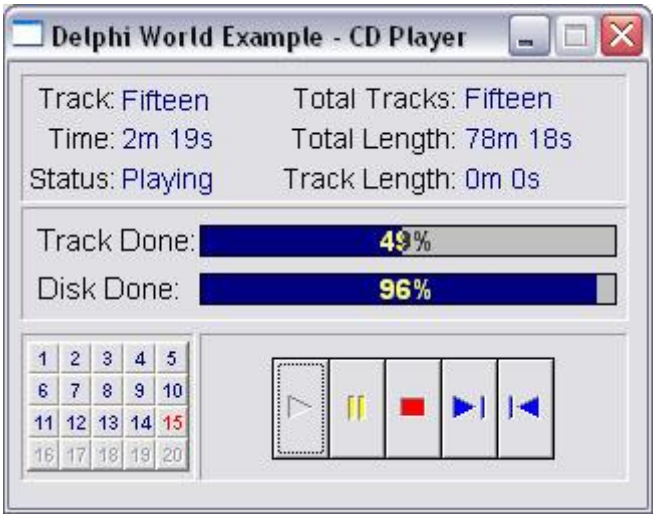
hWaveIn – очиқ Waveform audio киритиш қурилмаси аниқловчиси;

lpWaveInHdr – TwaveHdr структураси манзили;

uSize - WaveHdrнинг байтдаги ўлчами;

```
function waveInReset(
  hWaveIn: HWAVEIN
  ): MMRESULT; stdcall;
```

CD Player




```
unit Splash;

interface

uses Windows, Classes, Graphics, Forms, Controls, StdCtrls,
  ExtCtrls;

type
  TSplashScreen = class(TForm)
    StatusPanel: TPanel;
  end;

var
```

```
  SplashScreen: TSplashScreen;

implementation

{$R *.DFM}

begin
  SplashScreen := TSplashScreen.Create(Application);
  SplashScreen.Show;
  SplashScreen.Update;
end.
unit CDMain;

interface

uses
  SysUtils, Windows, Classes, Graphics, Forms, Controls, MPlayer, StdCtrls,
  Menus, MMSystem, Messages, Buttons, Dialogs, ExtCtrls, Splash, Gauges;

type
  TMainForm = class(TForm)
    tmUpdateTimer: TTimer;
    MainScreenPanel: TPanel;
    LblStatus: TLabel;
    Label2: TLabel;
    LblCurTrk: TLabel;
    Label4: TLabel;
    LblTrackTime: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    LblTotTrk: TLabel;
```

```
LblTotalLen: TLabel;
Label12: TLabel;
LblTrackLen: TLabel;
Label15: TLabel;
CDInfo: TPanel;
SBPanel: TPanel;
Panel1: TPanel;
mpCDPlayer: TMediaPlayer;
sbTrack1: TSpeedButton;
sbTrack2: TSpeedButton;
sbTrack3: TSpeedButton;
sbTrack4: TSpeedButton;
sbTrack5: TSpeedButton;
sbTrack6: TSpeedButton;
sbTrack7: TSpeedButton;
sbTrack8: TSpeedButton;
sbTrack9: TSpeedButton;
sbTrack10: TSpeedButton;
sbTrack11: TSpeedButton;
sbTrack12: TSpeedButton;
sbTrack13: TSpeedButton;
sbTrack14: TSpeedButton;
sbTrack15: TSpeedButton;
sbTrack16: TSpeedButton;
sbTrack17: TSpeedButton;
sbTrack18: TSpeedButton;
sbTrack19: TSpeedButton;
sbTrack20: TSpeedButton;
ggTrackDone: TGauge;
ggDiskDone: TGauge;
Label1: TLabel;
```

```
    Label3: TLabel;
    procedure tmUpdateTimerTimer(Sender: TObject);
    procedure mpCDPlayerPostClick(Sender: TObject; Button: TMPBtnType);
    procedure FormCreate(Sender: TObject);
    procedure sbTrack1Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
    OldTrack, CurrentTrack: Byte;
    m, s: Byte;
    TotalTracks: Byte;
    TotalLengthM: Byte;
    TotalLengthS: Byte;
    procedure GetCDTotals;
    procedure ShowTrackNumber;
    procedure ShowTrackTime;
    procedure ShowCurrentTime;
    procedure ShowPlayerStatus;
    procedure AdjustSpeedButtons;
    procedure HighlightTrackButton;
    function TrackNumToString(InNum: Byte): string;
  end;

var
  MainForm: TMainForm;

implementation

{$R *.DFM}

const
```

```
  { Array of strings representing numbers from one to twenty: }
  NumStrings: array[1..20] of string[10] =
  ('One', 'Two', 'Three', 'Four', 'Five', 'Six', 'Seven', 'Eight', 'Nine',
    'Ten', 'Eleven', 'Twelve', 'Thirteen', 'Fourteen', 'Fifteen', 'Sixteen',
    'Seventeen', 'Eighteen', 'Nineteen', 'Twenty');
  MSFormatStr = '%dm %ds';
  PlayButtons: TButtonSet = [btPause, btStop, btNext, btPrev];
  StopButtons: TButtonSet = [btPlay, btNext, btPrev];

function TMainForm.TrackNumToString(InNum: Byte): string;
{ This function returns a string corresponding to a integer between 1 and 20.
  If the number is greater than 20, then the integer is returned as a string. }
begin
  if (InNum > High(NumStrings)) or (InNum < Low(NumStrings)) then
    Result := IntToStr(InNum) { if not in array, then just return number }
  else
    Result := NumStrings[InNum]; { return the string from NumStrings array }
end;

procedure TMainForm.AdjustSpeedButtons;
{ This method enables the proper number of speed buttons }
var
  i: integer;
begin
  { iterate through form's Components array... }
  for i := 0 to SBPanel.ControlCount - 1 do
    if SBPanel.Controls[i] is TSpeedButton then // is it a speed button?
      { disable buttons higher than number of tracks on CD }
      with TSpeedButton(SBPanel.Controls[i]) do
        Enabled := Tag <= TotalTracks;
end;
```

```
procedure TMainForm.GetCDTotals;
{ This method gets the total time and tracks of the CD and displays them. }
var
  TimeValue: longint;
begin
  mpCDPlayer.TimeFormat := tfTMSF; // set time format
  TimeValue := mpCDPlayer.Length; // get CD length
  TotalTracks := mci_Tmsf_Track(mpCDPlayer.Tracks); // get total tracks
  TotalLengthM := mci_msf_Minute(TimeValue); // get total length in mins
  TotalLengthS := mci_msf_Second(TimeValue); // get total length in secs
  { set caption of Total Tracks label }
  LblTotTrk.Caption := TrackNumToString(TotalTracks);
  { set caption of Total Time label }
  LblTotalLen.Caption := Format(MSFormatStr, [TotalLengthM, TotalLengthS]);
  { intitialize gauge }
  ggDiskDone.MaxValue := (TotalLengthM * 60) + TotalLengthS;
  { enable the correct number of speed buttons }
  AdjustSpeedButtons;
end;

procedure TMainForm.ShowPlayerStatus;
{ This method displays the status of the CD Player the the CD is currently
  being played. }
begin
  if mpCDPlayer.EnabledButtons = PlayButtons then
    with LblStatus do
    begin
      case mpCDPlayer.Mode of
        mpNotReady: Caption := 'Not Ready';
        mpStopped: Caption := 'Stopped';
```

```
      mpSeeking: Caption := 'Seeking';
      mpPaused: Caption := 'Paused';
      mpPlaying: Caption := 'Playing';
    end;
  end
    { If these buttons are displayed the CD Player must be stopped... }
 else if mpCDPlayer.EnabledButtons = StopButtons then
  LblStatus.Caption := 'Stopped';
end;

procedure TMainForm.ShowCurrentTime;
{ This method displays the current time of the current track }
begin
 { Minutes for this track }
 m := mci_Tmsf_Minute(mpCDPlayer.Position);
 { Seconds for this track }
 s := mci_Tmsf_Second(mpCDPlayer.Position);
 { update track time label }
 LblTrackTime.Caption := Format(MSFormatStr, [m, s]);
 { update track gauge }
 ggTrackDone.Progress := (60 * m) + s;
end;

procedure TMainForm.ShowTrackTime;
{ This method changes the track time to display the total length of the
 currently selected track. }
var
 Min, Sec: Byte;
 Len: Longint;
begin
 { Don't update the information if player is still on the same track }
```

```
  if CurrentTrack <> OldTrack then
  begin
    Len := mpCDPlayer.TrackLength[mci_Tmsf_Track(mpCDPlayer.Position)];
    Min := mci_msf_Minute(Len);
    Sec := mci_msf_Second(Len);
    ggTrackDone.MaxValue := (60 * Min) + Sec;
    LblTrackLen.Caption := Format(MSFormatStr, [m, s]);
  end;
  OldTrack := CurrentTrack;
end;

procedure TMainForm.HighlightTrackButton;
{ This procedure changes the color of the speedbutton font for the current
  track to red, while changing other speedbuttons to navy blue. }
var
  i: longint;
begin
  { iterate through form's components }
  for i := 0 to ComponentCount - 1 do
    { is it a speedbutton? }
    if Components[i] is TSpeedButton then
      if TSpeedButton(Components[i]).Tag = CurrentTrack then
        { turn red if current track }
        TSpeedButton(Components[i]).Font.Color := clRed
      else
        { turn blue if not current track }
        TSpeedButton(Components[i]).Font.Color := clNavy;
end;

procedure TMainForm.ShowTrackNumber;
{ This method displays the currenty playing track number. }
```

```
var
  t: byte;
begin
  t := mci_Tmsf_Track(mpCDPlayer.Position); // get current track
  CurrentTrack := t; // set instance variable
  LblCurTrk.Caption := TrackNumToString(t); // set Curr Track label caption
  HighlightTrackButton; // Highlight current speedbutton
end;

procedure TMainForm.tmUpdateTimerTimer(Sender: TObject);
{ This method is the heart of the CD Player.  It updates all information at
  every timer interval. }
begin
  if mpCDPlayer.EnabledButtons = PlayButtons then
  begin
    mpCDPlayer.TimeFormat := tfMSF;
    ggDiskDone.Progress := (mci_msf_minute(mpCDPlayer.Position) * 60 +
      mci_msf_second(mpCDPlayer.Position));
    mpCDPlayer.TimeFormat := tfTMSF;
    ShowTrackNumber; // Show track number the CD player is currently on
    ShowTrackTime; // Show total time for the current track
    ShowCurrentTime; // Show elapsed time for the current track
  end;
end;

procedure TMainForm.mpCDPlayerPostClick(Sender: TObject;
  Button: TMPBtnType);
{ This method displays the correct CD Player buttons when one of the buttons
  are clicked. }
begin
  case Button of
```

```
    btPlay:
      begin
        mpCDPlayer.EnabledButtons := PlayButtons;
        LblStatus.Caption := 'Playing';
      end;
    btPause:
      begin
        mpCDPlayer.EnabledButtons := StopButtons;
        LblStatus.Caption := 'Paused';
      end;
    btStop:
      begin
        mpCDPlayer.Rewind;
        mpCDPlayer.EnabledButtons := StopButtons;
        LblCurTrk.Caption := 'One';
        LblTrackTime.Caption := '0m 0s';
        ggTrackDone.Progress := 0;
        ggDiskDone.Progress := 0;
        LblStatus.Caption := 'Stopped';
      end;
    btPrev, btNext:
      begin
        mpCDPlayer.Play;
        mpCDPlayer.EnabledButtons := PlayButtons;
        LblStatus.Caption := 'Playing';
      end;
  end;
end;

procedure TMainForm.FormCreate(Sender: TObject);
{ This method is called when the form is created. It opens and initializes the
```

```
  player }
begin
  try
    mpCDPlayer.Open; // Open the CD Player device.
    { If a CD is already playing at startup, show playing status. }
    if mpCDPlayer.Mode = mpPlaying then
      LblStatus.Caption := 'Playing';
    GetCDTotals; // Show total time and tracks on current CD
    ShowTrackNumber; // Show current track
    ShowTrackTime; // Show the minutes and seconds for the current track
    ShowCurrentTime; // Show the current position of the CD
    ShowPlayerStatus; // Update the CD Player's status
  except
    { If a error occurred, the system may be incapable of playing CDs. }
    on EMCIDeviceError do
    begin
      MessageDlg('Error Initializing CD Player.  Program will now exit.',
        mtError, [mbOk], 0);
      Application.Terminate;
    end;
  end;
  { Check the current mode of the CD-ROM and enable the appropriate buttons. }
  case mpCDPlayer.Mode of
    mpPlaying: mpCDPlayer.EnabledButtons := PlayButtons;
    mpStopped, mpPaused: mpCDPlayer.EnabledButtons := StopButtons;
  end;
  SplashScreen.Release; // Close and free the splash screen
end;

procedure TMainForm.sbTrack1Click(Sender: TObject);
{ This method sets the current track when the user presses one of the track
```

```
  speed buttons.  This method works with all 20 speed buttons, so by looking at
  the 'Sender' it can tell which button was pressed by the button's tag. }
begin
  mpCDPlayer.Stop;
  { Set the start position on the CD to the start of the newly selected track }
  mpCDPlayer.StartPos := mpCDPlayer.TrackPosition[(Sender as TSpeedButton).Tag];
  { Start playing CD at new position }
  mpCDPlayer.Play;
  mpCDPlayer.EnabledButtons := PlayButtons;
  LblStatus.Caption := 'Playing';
end;

procedure TMainForm.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  mpCDPlayer.Close;
end;

end.
```

Амалий машғулот №4. Маълумотлар омбори ва у билан ишлаш

Маълумотлар омборига қўйиладиган талаблар.

Демак, яхши лоиҳаланган маълумотлар омбори:

- фойдаланувчиларнинг маълумотлар омборига бўлган барча талабларини қондиради. Шунинг учун маълумотлар омборини лоиҳалашдан олдин фойдаланувчиларнинг маълумотлар омборига бўлган талабларини кенг миқёсда ўрганиб чиқиш зарур.
- Маълумотларнинг мослиги ва ишончлилигини кафолатлайди. Жадвалларни тузишда фойдаланувчи томонидан мумкин бўлмаган маълумотлар киритилашининг олдини олиш учун маълум шартларни киритиш керак.

Delphiда маълумотлар омбори билан ишловчи компонентлар шарҳи

Delphiда маълумотлар омбори билан ишловчи жуда кўп компонентлар бўлиб, биз улардан асосан TTable, TDbgrids, TDataSource, TDbNavigator компонентлари билан танишамиз.

 - TTable объекти маълумотлар омборидаги мавжуд жадвал билан мулокот ўрнатиш учун хизмат қилади. TTable ихтиёрий типдаги (FoxPro, ODBC, SQL ...) маълумотлар омборининг ҳар бир ёзувига ва майдонига тўғридан тўғри мурожаат қила олади. Бу компонент шунингдек, алоҳида ҳисоботлар билан ҳам мулоқот ўрната олади.

TTABLE объектидан фойдаланишдан олдин унга маълумотлар омбори алясини улаш керак, яъни шу компонентнинг DatabaseName хусусиятида чиқадиган рўйхатдан керакли алясни танлаш ва TableName хусусиятидаги рўйхатдан керакли жадвал номини танлаш керак. TTABLE объектини фаоллаштириш учун Active хусусияти қийматини ***true*** га ўтказиш керак.

 TADOTABLE объекти ҳам худди TTABLE объекти каби маълумотлар омборидаги бирор жадвалга боғланиш ва унга мурожаат қилишучун хизмат қилади. Бу объектдан асосан MSAccess маълумотлар омборини бошқариш тизимида яратилган омборлар билан ишлашга мўлжалланган. Бу объект асосан TADOconnection объекти билан бирга қўлланилиб, TADOconnection маълумотлар

омборига уланади. Шундан сўнг бир ёки бир нечта TAdotable объектлари Connection хусусияти ёрдамида TADOconnectionга уланади ва TableName хусусияти ёрдамида керакли жадвалга уланади. Объектни фаоллаштириш учун Active хусусияти қийматини ***true*** га ўтказиш керак. Бу объект ёрдамида маълумотлар омборидан маълумотларни бирор филтр ёрдамида ажратиб олиш мумкин.

 TDATASOURCE объекти бевосита TTable ёки TAdoTablega боғланиб, маълумотлар омборидаги ёзувларни таҳрирлаш, уларга мурожаат қилиш имконини беради. Бунинг учун компонентнинг DataSet хусусиятидаги рўйхатдан керакли Table элементи танланади ва шу орқали икки объект бирбирига боғланади. Ҳар бир алоҳида . TDATASOURCE объекти битта маълумотлар омборидаги битта жадвалга улана олади.

Юқоридаги учала объект ҳам дастур бажарилиш вақтида кўринмайдиган объект бўлиб, Формалар Дизайнери кўринишида уларни формага ташлаганда ўлчамларни ўзгартириб бўлмайди. Уларни маълумотлар омборига Формалар Дизайнери режимида ҳам, дастурий йўл билан дастур бажарилиш вақтида ҳам боғлаш мумкин.

Бунинг учун қуйидагича кодлар ёзилади:

```
begin
        Table1.DatabaseName:='DBDEMOS';
        Table1.TableName:='animals.dbf';
        Table1.Active:=True;
        DataSource1.DataSet:=Table1;
        DBGrid1.DataSource:=DataSource1;
end;
```

 TDBGRID объекти маълумотлар омборидаги ҳисоботлар, жадволлар ва сўровлардаги маълумотларни жадвал кўринишида намойиш этиш учун қўлланади. Бу объект ёрдамида маълумотлар омборидаги ёзувларни намойиш қилиш, таҳрирлаш ва ўзгартириш мумкин. Киритилган ўзгартиришлар жорий ёзув устида боради ва бу ўзгаришлар фақат сиз бошқа ёзувга ўтганингизда, ёки дастурни ёпганингизда сақлаб қолинади. TDBGRID объекти бевосита Datasource хусусияти

ёрдамида TDatasource объектга боғланади ва шу орқали маълумотларни намойиш этади.

 TDBNavigator (QDBCtrls) объекти дастурда TDBGRID ёки TDBEDIT компонентлаи орқали маълумотлар омбори ёзувларига мурожаат қилинаётга вақтда қўлланилади. TDBNavigator фойдаланувчига маълумотлар омборидаги ёзувларни таҳрирлаш ёки кўриб чиқишда қўл келади. Фойдаланувчи TDBNavigator тугмалардан бирини босганда ша тугма билан боғланган амал дастурда бажарилади. Масалан, фойдаланувчи Insert тугмасини босганда маълумотлар жадвалида битта бўш қатор ҳосил бўлади.

TDBNavigator тугмалари ва улар бажарадиган амаллар

Қуйида TDBNavigator тугмалари ва улар бажарадиган амалларни кўриб ўтамиз:

Тугма	**Амал**
 First	Маълумотлар омборидаги дастлабки ёзувни фаоллаштириш. У фақат жорий ёзув дастлабки ёзув бўлмагандагина фаол бўлади.
 Prior	Маълумотлар омборида жорий ёзувдан олдинги ёзувни фаоллаштириш. У фақат жорий ёзув дастлабки ёзув бўлмагандагина фаол бўлади.
 Next	Маълумотлар омборида жорий ёзувдан кейинги ёзувни фаоллаштириш. У фақат жорий ёзув охирги ёзув бўлмагандагина фаол бўлади.
 Last	Маълумотлар омборидаги охирги ёзувни фаоллаштириш. У фақат жорий ёзув охирги ёзув

	бўлмагандагина фаол бўлади.
 Insert	Жадвалга маълумотларни киритиш учун янги сатр қўшиш. Бунда сатрнинг ихтиёрий майдонига маълумот киритилганда ўзгаришлар сақланади
 Delete	Жорий ёзувни ўчириш. Бунда ёзувни ўчириш ҳақида сўров чиқарилади ва ўчирилган ёзув қайта тикланмайди.
 Edit	Жорий ёзувни ўзгартириш, таҳрирлаш мумкин бўлган ҳолатга ўтказиш.
 Post	Киритилган ўзгаишларни хотирада сақлаш. Бунда жорий майдонинг олдинги маълумотлари ўрнига киритилган ўзгаришлар сақланади.
 Cancel	Жорий ёзувга киритилган ўзгаришларни бекор қилиш. Бу амалдан жорий ёзувни алмаштиргунча фойдаланиш мумкин.

Қуйида маълумотлар омборига дастурий уланиш коди кўрсатилган:

```
ADOQuery1.Active := false;
ADOQuery1.SQL.Clear;
ADOQuery1.SQL.Add('SELECT max(id)'); // -- Сбой здесь !!!
ADOQuery1.SQL.Add('AS idmax');
ADOQuery1.SQL.Add('FROM main');
```

```
ADOQuery1.Active := true;
```

Типлар муаммоси

1. Сўровни бир сатрда бериш. Мисол:

```
ADOQuery1.Active := false;
ADOQuery1.SQL.Text := 'SELECT max(id) AS idmax FROM main;';
ADOQuery1.Active := true;
```

2. Tools->Debugger Options->Language Exceptions->Stop on Delphi Exceptions байроғини олиб ташлаш

3. Бу камчиликни бекор қилиш

Изоҳ:

Delphi 5 даги TADOQuery компоненти ўхшаш кодга эга:

```
ADOQuery1.SQL.BeginUpdate;
try
  ADOQuery1.SQL.Clear;
  ADOQuery1.SQL.Add('SELECT max(id)');
  ADOQuery1.SQL.Add('AS idmax');
  ADOQuery1.SQL.Add('FROM main');
finally
  ADOQuery1.SQL.EndUpdate;
end;
```

Delphiда ADO билан тўғридан тўғри ишлаш.

```
var
  OptionalParam: OleVariant;
  VarData: PVarData;
begin
  OptionalParam := DISP_E_PARAMNOTFOUND;
```

```
  VarData := @OptionalParam;
  VarData^.VType := varError;
```

Яна бир мисол:

```
var
  MyConn: _Connection;
  MyComm: _Command;
  MyRecSet: _Recordset;
  prm1: _Parameter;
begin
  MyConn := CoConnection.Create;
  MyConn.ConnectionString := 'DSN=pubs;uid=sa;pwd=;'; MyConn.Open( '', '', '', -1 );
  MyCommand := CoCommand.Create;
  MyCommand.ActiveConnection := MyConn;
  MyCommand.CommandText := 'SELECT * FROM blahblah WHERE BlahID=?'
  Prm1 := MyCommand.CreateParameter( 'Id', adInteger.adParamInput, -1, <value> );
  MyCommand.AppendParameter( Prm1 );
  MyRecSet := CoRecordSet.Create;
  MyRecSet.Open( MyCommand, OptionalParam, adOpenDynamic, adLockReadOnly,
adCmdText );
```

Delphi да бўш қиймат ўрнида ишлатилувчи EmptyParam "константа"си мавжуд.

Амалий машғулот №5. Сўровлар ва ҳисоботлар билан ишлаш

SQL – ҳисобланувчи майдонларни тартиблаш

Баъзида сўров натижаси сифатида чиқувчи жадвалнинг майдонлари ҳисобланувчи бўлади. Бу Delphi дастурларида жуда оддий, лекин қўлланилаётган типга қараб, турли натижаларни бериши мумкин..

Локал SQLучун ҳисобланувчи майдонларга AS калит сўзи орқали мурожаат қилинади. Бунда бирор майдонга тартиблаш мақсадида ORDER BY калит сўзидан фойдаланиш мумкин. Масалан, ITEMS.DB жадвалидан фойдаланамиз:

```
SELECT I."PARTNO", I."QTY", (I."QTY" * 100) AS TOTAL
FROM "ITEMS.DB" I
ORDER BY TOTAL
```

Бу мисолда ҳисобланувчи майдонга TOTAL номини бердик (вақтинча, фақат мурожаат учун), шундан сўнг унга ORDER BY ифодасини қўллаш мумкин.

```
SELECT EMP_NO, SALARY, (SALARY / 12) AS MONTHLY
FROM EMPLOYEE
ORDER BY 3 DESCENDING
SELECT I."PARTNO", I."QTY", (I."QTY" * 100) AS TOTAL
FROM "ITEMS.DB" I
ORDER BY 3
```

Номида бўш жой ёки махсус рамзли майдонлардан сўров олиш

Delphi даги TQuery компоненти ёрдамида SQL-сўровлар яратишда бўш жой ёки махсус рамзли майдонлардан билан ишлаш учун махсус синтаксис талаб қилинади. Бунда SQL Select ифодасидан қуйидагича фойдаланиш мумкин:

```
SELECT
 Species No,
 Category,
 Common_Name,
 Species Name,
```

```
 Length (cm),
 Length_In,
 Notes,
 Graphic
FROM
 BIOLIFE
```

Қуйида тўғри тасвирланган код келтирилган:

```
SELECT
 BIOLIFE."Species No",
 BIOLIFE."Category",
 BIOLIFE."Common_Name",
 BIOLIFE."Species Name",
 BIOLIFE."Length (cm)",
 BIOLIFE."Length_In",
 BIOLIFE."Notes",
 BIOLIFE."Graphic"
FROM
 "BIOLIFE.DB" BIOLIFE
```

Изоҳ: бу мисолдан фақатгина DatabaseName хусусияти аниқланганда фойдаланиш мумкин.

```
SELECT
 BIOLIFE."Species No",
 BIOLIFE.Category,
 BIOLIFE.Common_Name,
 BIOLIFE."Species Name",
 BIOLIFE."Length (cm)",
 BIOLIFE.Length_In,
 BIOLIFE.Notes,
 BIOLIFE.Graphic
FROM
 BIOLIFE
```

Акс ҳолда мисол қуйидаги кўринишни олади:

```
SELECT
 "C:\DELPHI\DEMOS\DATA\BIOLIFE.DB"."Species No",
 "C:\DELPHI\DEMOS\DATA\BIOLIFE.DB"."Category",
 "C:\DELPHI\DEMOS\DATA\BIOLIFE.DB"."Common_Name",
 "C:\DELPHI\DEMOS\DATA\BIOLIFE.DB"."Species Name",
 "C:\DELPHI\DEMOS\DATA\BIOLIFE.DB"."Length (cm)",
 "C:\DELPHI\DEMOS\DATA\BIOLIFE.DB"."Length_In",
 "C:\DELPHI\DEMOS\DATA\BIOLIFE.DB"."Notes",
 "C:\DELPHI\DEMOS\DATA\BIOLIFE.DB"."Graphic"
FROM
 "C:\DELPHI\DEMOS\DATA\BIOLIFE.DB"
```

Адабиётлар

1. Компьютердаги ёрдам файллари.
2. Бобровский «Delphi 5», «Питер» Москва 1997г.
3. Шумаков «Delphi 4 разработка баз данных», «Питер» Москва 1996г.
4. Пачеко, Тейксера «Delphi 5 пособие программиста», «Питер» Москва 1999 г.
5. Фаронов «Delphi 4 учебное пособие», «Питер» Москва 1995 г.