

И.Г. Чугреев

**Учебное пособие по ориентированию
электронных карт для создания локальных баз данных.**

Часть 1. Ориентирование. Метод проекций

Учебное пособие для студентов 4 курса, обучающихся

*по направлению 09.03.02 Сбор и представление геоданных в информационных
системах*



М И И Г А и К

2022 г.

Задание 1

Все, что мы делали до сих пор, относилось к линейному программированию, или, как говорил Д. Гриз в своей книге “Наука программирования” (1984 г) - линейное программирование - это метод “всплывающего пузырька”. Т.е. - загрузили данные - получили результат.

Иногда, чтобы получить результат, пользователь программы должен выполнить ряд действий. Например в ГИС системах - загрузить снимок или карту для обработки, найти на документе нужный объект, зарегистрировать опорные точки, выполнить ориентирование документа в заданную систему координат, и т.д.

В данном случае используется программирование с учетом запросов будущего пользователя программы, или, другими словами, программирование по событиям.

Постановка задачи:

Глобальная - разработать информационную систему для работы с электронной топографической картой.

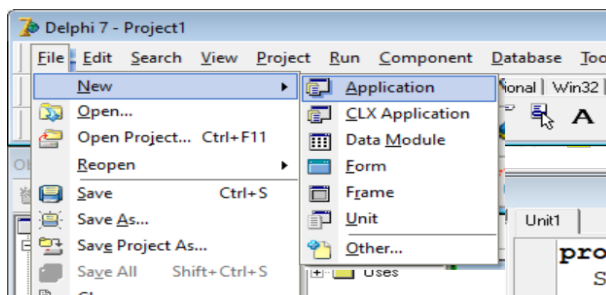
В данном уроке мы познакомимся со средствами Delphi7, позволяющими:

- загружать и отображать файлы формата BMP и JPG;

Приступим.

Готовим интерфейс пользователя.

1.Открыть новый проект. (Основное меню **File>New>Application**)



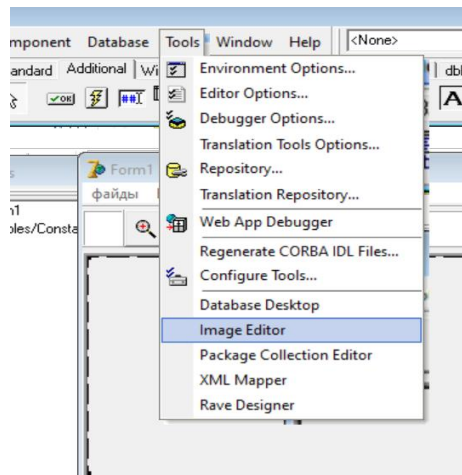
2. Для формы - Form1 -в инспекторе объектов, свойство Align (выравнивание) установить в alClient (Занять при запуске приложения во весь экран)

3. На форму положить:

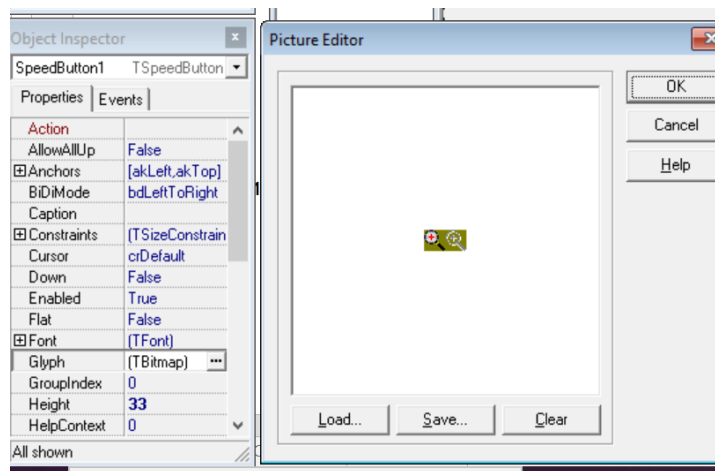
- компонент ToolBar - палитра компонентов WIN32;
- компонент Panel - палитра компонентов Standard; в инспекторе объектов, свойство Align (выравнивание) установить в alClient ;
- На Panel бросить компонент ScrollBox - палитра компонентов Additional.в инспекторе объектов, свойство Align (выравнивание) установить в alClient ;
- На компонент ScrollBox положить компонент PaintBox - палитра компонентов Systems
- На компонент ToolBar положить 3 панели(прямоугольники белого цвета), как на рисунке и 2 компонента SpeedButton - палитра компонентов Additional



е) В основном меню, кнопка Tools, запустили программу Image Editor, в которой создали два файла формата BMP, размером 25х25 пикселей, как на рис. выше и сохранили их в своем проекте.

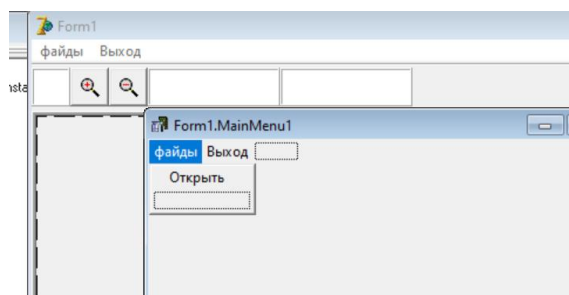


В инспекторе объектов, для каждого SpeedButton_а нашли свойство Glyph и кликнули по нему мышкой.

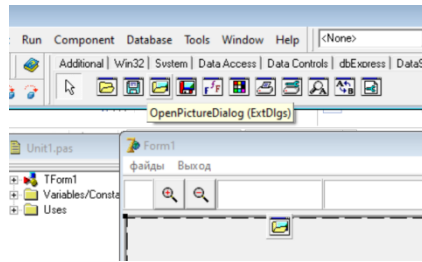


В сплывшем окне - Picture Editor - нажали кнопку Load и нашли соответствующую картинку для SpeedButton. Нажали OK. картинка “приклеилась” к SpeedButton_y.

4. На форму бросили разработчик меню и сделали как на картинке.

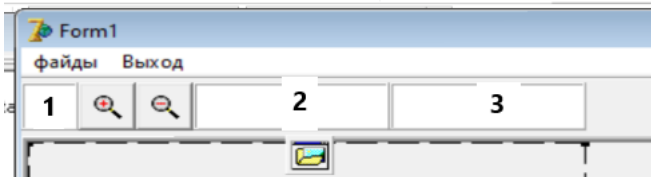


5. Палитра компонентов Dialogs - нашли и бросили на форму компонент OpenPictureDialog



Интерфейс пользователя предварительно готов.

Некоторые пояснения:



1. компонент ToolBar - служит для выравнивания горячих кнопок и окон.
В окно 1 - будет выводиться коэффициент увеличения изображения.
В окна 2 и 3 будут выводиться координаты курсора в системе координат изображения
SpeedButton ы - для увеличения/уменьшения изображения.
2. компонент PaintBox - служит для отображения битовых карт изображений
3. компонент ScrollBar для прокрутки больших изображений

Начинаем писать код программы.

1. В интерфейсной части UNIT1 опишем необходимые переменные:

```
var
  Form1: TForm1;
  Bm : tbitmap; // Битовая карта для загрузки\работы с изображением
  rect : trect; // координаты прямоугольной области окна просмотра изображения
  uv : integer=1; // коэффициент увеличения для окна 1
  uv1:real=1.0; // программный коэффициент увеличения
  Drag : boolean=false; // флаг для перетаскивания картинки изображения
  p,p1 : tpoint; // Экранные координаты курсора
  xp,yp : integer; // координаты курсора в системе картинки изображения с увеличением
  1
  i,j : integer; // счетчики цикла
  us_uv, us_uv1 : byte; // флаги для увеличения/уменьшения изображения
```

implementation

{ \$R *.dfm }

2. Для Form1, в инспекторе объектов, (закладка Events) создаем обработчик события OnCreate.
для этого, справа от OnCreate, в пустом поле, дважды щелкаем мышкой. Если в тексте программы нет синтаксических ошибок, то получим следующую процедуру:

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

end;

в которой запишем следующий код:

```
procedure TForm1.FormCreate(Sender: TObject);  
begin  
    bm:=tbitmap.Create; //создали контейнер битовой карты для загрузки картинки  
    Panel1.Visible:=false; //сделали невидимой панель и все что на ней есть-
```

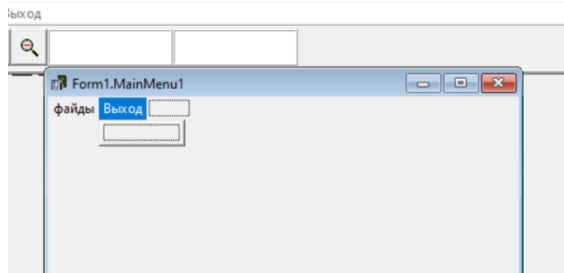
ScrollBar..

```
    Toolbar1.Visible:=false;  
end;
```

Если имеются синтаксические ошибки, их необходимо исправить, откомпилировав текст программы, после чего повторить создание обработчика события OnCreate;

3. Программируем основное меню:

- корректный выход из программы - дважды щелкаем мышкой по кнопке “Выход”.



Получаем обработчик события нажатия на кнопку “Выход”.

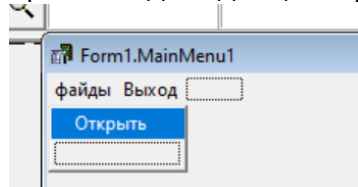
```
procedure TForm1.N3Click(Sender: TObject);  
begin
```

```
end;
```

в котором пишем следующий код

```
procedure TForm1.N3Click(Sender: TObject);  
begin  
    form1.Close; //корректно завершили работу программы  
    bm.Destroy; // удалили из памяти ПВМ битовую карту  
end;
```

- Программируем загрузку картинки - дважды щелкнув по кнопке “Открыть”



Получаем обработчик события нажатия на кнопку “Открыть”

```
procedure TForm1.N2Click(Sender: TObject);  
begin
```

```
end;
```

в который пишем следующий код

```
procedure TForm1.N2Click(Sender: TObject);  
begin
```

```

if openpicturedialog1.Execute then begin // стандартное выражение открытия модального
окна для работы с файлами и папками
  bm.LoadFromFile(openpicturedialog1.FileName); // загружаем картинку в битовую карту
  scrollbox1.Width:=bm.Width; // задаем размер ширины scrollbox1
  scrollbox1.Height:=bm.Height; // задаем размер высоты scrollbox1
  paintbox1.Width:=bm.Width; // задаем размер ширины paintbox1
  paintbox1.Height:=bm.Height;
  rect.Left:=0; // задаем координаты левого верхнего угла области вывода картинки
  rect.Top:=0;
  rect.Right:=paintbox1.Width; // задаем координаты правого нижнего угла области вывода
  rect.Bottom:=paintbox1.Height;
  Panel1.Visible:=true; //делаем видимыми все компоненты расположенные на панели -
PaintBox
  ToolBar1.Visible:=true; //делаем видимыми все компоненты расположенные на ToolBar
  paintbox1.Repaint; //вызываем событие PaintBox_a - OnPaint - отрисовка картинки начиная с
левого верхнего угла с увеличением=1
  panel2.Caption:='1'; //в окне панели на ToolBar выводим текущее увеличение.
end; //if openpicturedialog1.Execute
end;

```

➤ Активируем событие PaintBox_a - OnPaint - для отрисовки картинки для этого, в инспекторе объектов, для PaintBox_a, в закладке Events, находим событие OnPaint и дважды щелкаем мышкой справа от OnPaint, в пустом поле. получаем обработчик события

```

procedure TForm1.PaintBox1Paint(Sender: TObject);
begin

```

```

end;

```

в котором пишем код, для отображения картинки

```

procedure TForm1.PaintBox1Paint(Sender: TObject);
begin
  paintbox1.Canvas.StretchDraw(rect,bm);
end;

```

На сегодня достаточно. Продолжение следует.

Задание 2

В данном уроке мы познакомимся со средствами Delphi7, позволяющими:

- перемещаться по изображению;
- получать координаты точек (пикселей) изображения в системе координат файла изображения.

Полосами прокрутки пользоваться не всегда удобно. Поэтому напомним код программы для перетаскивания картинки с помощью мышки.

Для этого:

1. в инспекторе объектов отобразим свойства PaintBox_a.
2. Перейдем на вкладку Events (события)
3. Найдем событие OnMouseDown (нажата кнопка мыши) и щелкнем справа от этого события по пустому полю. Получим следующий обработчик события

```
procedure TForm1.PaintBox1MouseDown(Sender: TObject; Button: TMouseButton;  
  Shift: TShiftState; X, Y: Integer); //нажата  
begin  
  
end;
```

в который запишем следующий код

```
procedure TForm1.PaintBox1MouseDown(Sender: TObject; Button: TMouseButton; Shift:  
  TShiftState; X, Y: Integer); //нажата  
begin  
  if (ssshift in shift) or (ssctrl in shift) then begin // если нажата клавиша shift или control  
    to  
    drag:=true; // флагу drag присваиваем true – тащить объект  
    getcursorpos(p); //системная процедура – взять координаты положения курсора  
    exit; //выходим из процедуры  
  end; // if  
end;
```

4. Затем в инспекторе объектов находим событие OnMouseMove (Перемещение мыши). Получаем обработчик события

```
procedure TForm1.PaintBox1MouseMove(Sender: TObject; Shift: TShiftState; X,  
  Y: Integer);  
begin  
  
end; // перетаскиваем объект
```

в который запишем следующий код

```

procedure TForm1.PaintBox1MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer); // процедура возвращает текущие координаты пикселей X Y в системе
координат картинки
  var dx,dy : integer; // переменные содержащие количество пикселей перемещения
курсора от нажатия клавиши до ее отжатия
begin
  panel1.Caption:=inttostr(x); // в панели 1 и 3 (на ToolBar) выводим текущие
координаты курсора
  panel3.Caption:=inttostr(y);
  if (drag=true) then begin
    getcursorpos(p1); // системная процедура – взять координаты положения курсора
    dx:=round((p1.X-p.x) * 2.1); // вычисляем перемещение курсора
    dy:=round((p1.y-p.y) * 2.1);
    Scrollbox1.HorzScrollBar.Position:=Scrollbox1.HorzScrollBar.Position-dx; //бегунок
Scrollbox1 смещается на dx и dy – происходит автоматическая перерисовка картинки.
Левый верхний угол выводимой картинки в окне экрана получает значение бегунка
Scrollbox1
    Scrollbox1.VertScrollBar.Position:=Scrollbox1.VertScrollBar.Position-dy;
    getcursorpos(p); // системная процедура – взять координаты положения курсора
после перерисовки
  end; // перетаскиваем объект

```

5. Для окончания перетаскивания картинки необходимо отжать клавишу мышки. Для этого находим событие OnMouseUp (клавиша мыши отпущена) и создать обработчик события (как и ранее):

```

procedure TForm1.PaintBox1MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer); //отпущена
Begin

```

```

end;

```

в который написать следующий код

```

procedure TForm1.PaintBox1MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer); //отпущена
begin
  if (drag=true) then begin
    drag:=false;
    exit;
  end; // if
end;

```

Процедуры для перетаскивания картинки готовы. Откомпилируйте программу и запустите ее. Загрузите карту (старосты должны сделать рассылку карты).

Для перетаскивания зажмите клавишу Sift или Ctrl, зажмите левую кнопку мыши и с нажатой кнопкой перемещайте курсор по экрану. Если перемещение картинки слишком быстрое, то в выражениях

```
dx:=round((p1.X-p.x) * 2.1);// вычисляем перемещение курсора  
dy:=round((p1.y-p.y) * 2.1);
```

подберите коэффициент (* **2.1**) уменьшите его или увеличьте.

Задание 3

В данном уроке мы познакомимся со средствами Delphi7, позволяющими:

- Масштабировать изображение (увеличивать. Уменьшать);

Для начала, в ранее написанный код внесем некоторые дополнения, касающиеся работы с файлами *.JPG.

В раздел USES интерфейсной части программы добавим динамическую библиотеку JPEG, позволяющую работать с файлами JPG.

unit Unit1;

interface

uses

**Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, ExtDlgs, Menus, ExtCtrls, Buttons, ToolWin, ComCtrls, jpeg;**

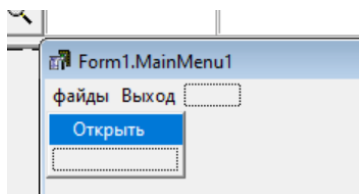
type

.....

В интерфейсной части UNIT1 добавим переменную GP : Tjpegimage в область описания переменных:

```
var
  Form1: TForm1;
  Bm : tbitmap; // Битовая карта для загрузки\работы с изображением
  rect : trect; // координаты прямоугольной области окна просмотра изображения
  uv : integer=1; // коэффициент увеличения для окна 1
  uv1:real=1.0; // программный коэффициент увеличения
  Drag : boolean=false; // флаг для перетаскивания картинки изображения
  p,p1 : tpoint; // Экранные координаты курсора
  xp,yp : integer; // координаты курсора в системе картинки изображения с увеличением
  1
  i,j : integer; // счетчики цикла
  us_uv, us_uv1 : byte; // флаги для увеличения/уменьшения изображения
GP : Tjpegimage;
```

Затем,



в обработчик события нажатия на кнопку “Открыть”

дополняем код

```
procedure TForm1.N1Click(Sender: TObject);
```

```

var rsh : string; //Переменная для расширения файла
begin
  if openPictureDialog1.Execute then begin
    bm:=tbitmap.Create;
    rsh:=ansiuppercase(extractfileext(openPictureDialog1.FileName));// получаем
расширение файла
    if rsh='.BMP' then begin
      bm.LoadFromFile(openPictureDialog1.FileName);
    end;
    if rsh='.JPG' then begin //если расширение rsh='.JPG' то
      gp:=tjpeimage.Create; // создаем контейнер для распаковки файла
      gp.LoadFromFile(openPictureDialog1.FileName);//загружаем и распаковываем файл
      bm.assign(gp); // передаем распакованный файл в стандартную переменную типа
TBitmap
      gp.Destroy;//уничтожаем контейнер GP
    end;
    if (rsh<>'.BMP')and(rsh<>'.JPG') then begin
      showmessage('Формат не поддерживается !');
      Exit;
    end;
    rect.Left:=0;
    rect.Top:=0;
    rect.Right:=bm.Width;
    rect.Bottom:=bm.Height;
    scrollBox1.Width:=bm.Width;
    scrollBox1.Height:=bm.Height;
    PaintBox1.Width:=bm.Width;
    PaintBox1.Height:=bm.Height;
  end;
end;

```

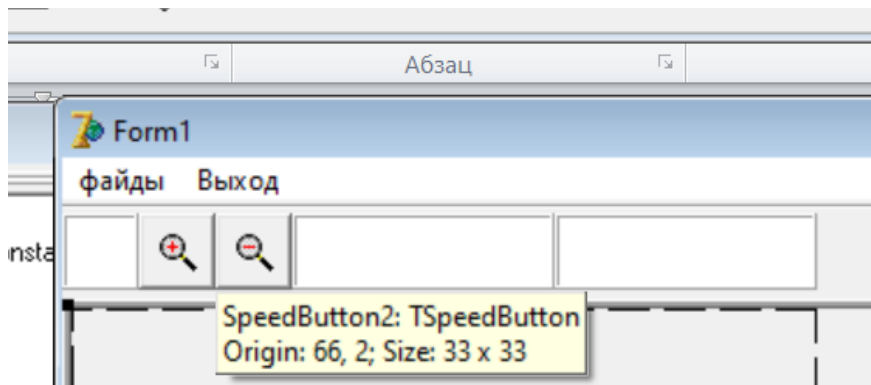
Пояснения к строке

rsh:=ansiuppercase(extractfileext(openPictureDialog1.FileName));// получаем
расширение файла

- Ansiuppercase(ss:string) : функция для работы со строками – преобразует символы строки к верхнему регистру (заглавные буквы)
- Extractfileext(SS : string) функция для работы с файлами – возвращает расширение файла типа String
- SS – выражение типа string – (openPictureDialog1.FileName) – путь к файлу+имя файла с расширением.

Дополнения закончены. Теперь мы можем работать с двумя типами файлов – BMP и JPG.

Приступаем к масштабированию.



Щелкнем дважды мышкой по кнопке. Получим обработчик события:

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin

end;
```

В данный обработчик, флагам увеличения `us_uv`, `us_uv1` присваиваем значения

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
  us_uv:=0; // увеличение плюс
  us_uv1:=1; // увеличение плюс
end;
```



то же самое сделаем для кнопки

```
procedure TForm1.SpeedButton2Click(Sender: TObject);
begin
  us_uv:=1;
  us_uv1:=1;
end;
```

Найдем Для `Paintbox1_a` событие `OnMouseDown` и добавим туда следующий код:

```
procedure TForm1.PaintBox1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer); //нажата
begin
```

```
  if (ssshift in shift) or (ssctrl in shift) then begin // перетаскиваем объект
    drag:=true;
    getcursorpos(p);
    exit;
  end;
```

```
  if (us_uv1=1) and (button=mbLeft) then begin
```

```
    uvelichenie(x,y); // вызываем процедуру увеличения и передаем в нее фактические
    параметры – координаты положения курсора в системе координат изображения
```

```
exit;  
end;  
end;// procedure TForm1.PaintBox1MouseDown
```

- **увеличение(x,y)** – процедура увеличения, которую мы сейчас создадим;
- **x,y** – текущие координаты курсора в системе координат изображения, возвращает процедура
procedure TForm1.PaintBox1MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
- **button=mbLeft** –если нажат левый бUTTON мышки – значение нажатого бUTTONа возвращает та же процедура

Создаем процедуру *увеличение(x,y)*:

1. В раздел описания типов TYPE интерфейсной части программы добавим свой обработчик события. Запишем (выделено красным цветом):

```
unit Unit1;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, ExtDlgs, Menus, ExtCtrls, Buttons, ToolWin, ComCtrls;
```

```
type
```

```
TForm1 = class(TForm)  
  MainMenu1: TMainMenu;  
  N1: TMenuItem;  
  N2: TMenuItem;  
  N3: TMenuItem;  
  OpenPictureDialog1: TOpenPictureDialog;  
  ToolBar1: TToolBar;  
  Panel2: TPanel;  
  SpeedButton1: TSpeedButton;  
  SpeedButton2: TSpeedButton;  
  Panel1: TPanel;  
  Panel3: TPanel;  
  PopupMenu1: TPopupMenu;  
  N4: TMenuItem;  
  N5: TMenuItem;  
  N6: TMenuItem;  
  Panel4: TPanel;  
  ScrollBox1: TScrollBox;  
  PaintBox1: TPaintBox;  
  procedure N2Click(Sender: TObject);
```

```

procedure FormCreate(Sender: TObject);
procedure PaintBox1Paint(Sender: TObject);
procedure N3Click(Sender: TObject);
procedure SpeedButton2Click(Sender: TObject);
procedure SpeedButton1Click(Sender: TObject);
procedure PaintBox1MouseUp(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
procedure PaintBox1MouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
procedure PaintBox1MouseMove(Sender: TObject; Shift: TShiftState; X,
    Y: Integer);
procedure N4Click(Sender: TObject);
procedure N5Click(Sender: TObject);
procedure N6Click(Sender: TObject);
procedure uvelichenie(x,y:integer);

```

private

2. В разделе написания кода – implementation – создадим процедуру обработки события uvelichenie(x,y:integer); :

implementation

{ \$R *.dfm }

```

procedure TForm1.uvelichenie(x,y:integer);
begin
    xp:=trunc(x/uv1); //преобразуем текущие координаты положения курсора
    yp:=trunc(y/uv1); // к масштабу 1:1
    case us_uv of // если флаг увеличения
    0: begin // us_uv=0, то коэфф. увеличения увеличиваем на 1
        uv:=uv+1;
        if uv=-1 then uv:=1;
    end;
    1: begin // us_uv=1 то коэфф. увеличения уменьшаем на 1
        uv:=uv-1;
        if uv=0 then uv:=-2;
    end;
    end; // case
    if (uv<0) then uv1:=1/abs(uv)
    else uv1:=uv;
    if uv>0 then panel2.Caption:=inttostr(uv) //выводим в окна тоолбар текущее
    else panel2.Caption:='-'+inttostr(abs(uv)); // увеличение
    if (uv=1)or(uv=-1)then panel2.Caption:=inttostr(abs(uv));
    paintbox1.Width:=round(bm.Width*abs(uv1)); // изменяем размер paintbox1
    paintbox1.Height:=round(bm.Height*abs(uv1)); // в соответствии с
    //коэффициентом увеличения
    scrollbar1.Width:=round(bm.Width*abs(uv1)); // то же самое для scrollbar1.
    scrollbar1.Height:=round(bm.Height*abs(uv1));
    rect.Right:=paintbox1.Width; // изменяем область вывода картинки
    rect.Bottom:=paintbox1.Height;

```

```

scrollbox1.HorzScrollBar.Position:=trunc(xp*uv1) -
trunc((paintbox1.Parent.ClientWidth/2)); // ползунком scrollbox1
scrollbox1.vertScrollBar.Position:=trunc(yp*uv1-
(paintbox1.Parent.ClientHeight/2)); // задаем координаты левого верхнего угла
end; // участка изображен. для его вывода на экран

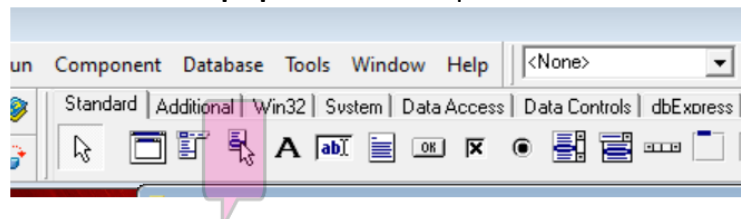
```

Изменения положения ползунков **scrollbox1 вызовут событие OnPaint** - перерисовка произойдет автоматически. Место щелчка мышки отобразится в центре экрана

Свойство **paintbox1.Parent.ClientWidth** – содержит ширину клиентской области экрана монитора (окна PaintBox_a).

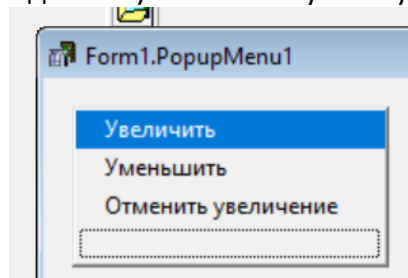
Свойство **paintbox1.Parent.ClientHeight** – содержит высоту клиентской области экрана монитора (окна PaintBox_a).

3. на форму бросим компонент **PopUpMenu** –палитра компонентов **Standard**



PopUpMenu

Щелкнем мышкой дважды по данному компоненту . Получим разработчик меню:



В котором создадим кнопки как на рисунке (надпись создается в инспекторе объектов, свойство Caption).

Щелкнув по кнопке «Увеличить» создадим обработчик событий в который флагам увеличения присвоим значения

```

procedure TForm1.N4Click(Sender: TObject);
begin
  us_uv:=0; //увеличение +
  us_uv1:=1; // выполнить увеличение после щелчка левой кнопкой мыши
end;

```

Щелкнув по кнопке «Уменьшить» создадим обработчик событий в который флагам увеличения присвоим значения

```

procedure TForm1.N5Click(Sender: TObject);
begin
  us_uv:=1; //увеличение минус
  us_uv1:=1; // выполнить увеличение после щелчка левой кнопкой мыши
end;

```

Щелкнув по кнопке «Отменить увеличение» создадим обработчик событий в который флагу увеличения присвоим значения

```

procedure TForm1.N6Click(Sender: TObject);
begin
    us_uv1:=0;// не выполнять увеличение после щелчка левой кнопкой //мыши
end;

```

Добавим в процедуру TForm1.PaintBox1MouseDown код для вызова PopUpMenu

```

procedure TForm1.PaintBox1MouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer); //нажата
begin
    if (ssshift in shift)or(ssctrl in shift) then begin // перетаскиваем объект
        drag:=true;
        getcursorpos(p);
        exit;
    end;
    If Button=mbRight then begin // если нажата правая кнопка мыши
        getcursorpos(p); //взять координаты
        popupmenu1.Popup(p.x,p.y); //установить левый верхний угол PopUpMenu в точку с
                                координатами p.x,p.y

        exit;
    end;
    if (us_uv1=1)and(button=mbLeft) then begin
        uvelichenie(x,y); // вызываем процедуру увеличения и передаем в нее фактические
        параметры –
        //координаты положения курсора в системе координат изображения
        exit;
    end;
end;// procedure TForm1.PaintBox1MouseDown

```

При нажатии правой кнопки мыши появится всплывающее меню.

После нажатия на соответствующую кнопку PopUpMenu левой кнопкой – меню исчезнет.
Будет выполняться одна из выбранных команд.

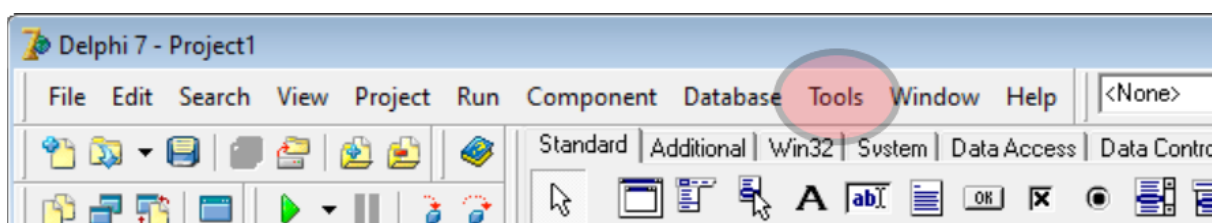
Откомпилируйте программу и запустите ее.
Выполните тестирование программы.

Задание 4

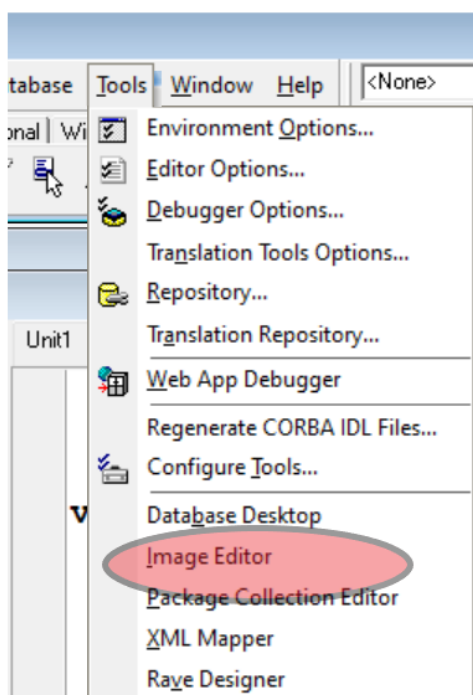
При разработке различных задач и удобства их использования, иногда приходится разрабатывать собственные ресурсы, такие как иконки для кнопок, курсоры и т.д. Например, в нашей задаче, при выполнении масштабирования, пользоваться стандартными курсорами дельфи не совсем удобно. Было бы желательно при включении режима увеличения иметь курсор, который бы подсказывал нам, что такой режим включен.

В данном уроке мы познакомимся со средствами Delphi7, позволяющими создавать и подгружать собственные курсоры, создавать ресурсные файлы для их хранения, в нужный момент подгружать необходимый курсор. Кроме этого научимся создавать примитивные командные файлы.

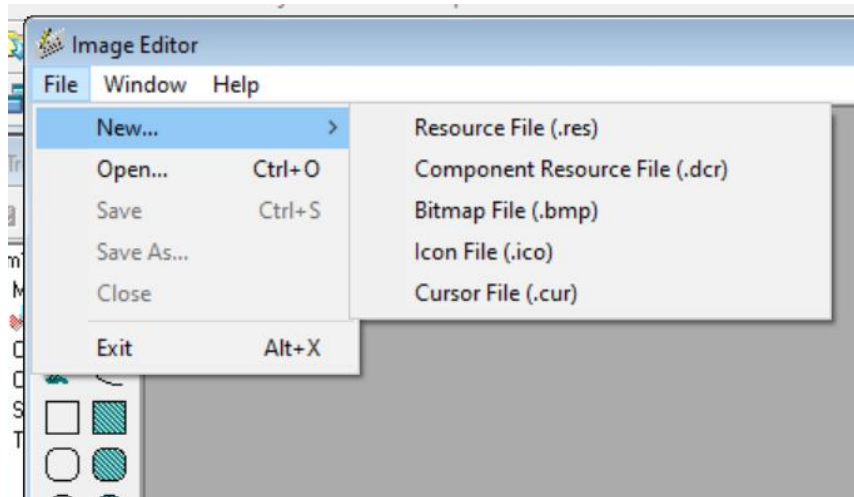
Для разработки собственных ресурсов, в дельфи, в основном меню Tools, имеется



среда разработки собственных ресурсов – Image Editor, с которым мы уже поверхностно познакомились в прошлом семестре, создавая иконки для кнопок.



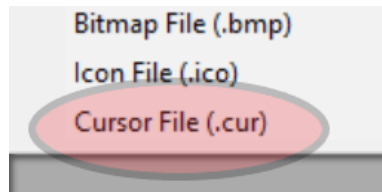
Графический редактор Image Editor позволяет создавать ресурсы картинок в формате BMP, иконки в формате ICO, курсоры в формате CUR. Как правило, иконки и картинки являются статическими элементами на кнопках, буттонах и т.д. Их подклеивают один раз, в процессе разработки программы и, в общем то нет необходимости хранить их в ресурсных файлах. Чего не скажешь о курсорах. При разработке интерактивных программ, за один сеанс работы курсор может поменять свой вид несколько раз. Например, при загрузке больших файлов часто, вместо стандартного курсора – стрелка – часто используется курсор –



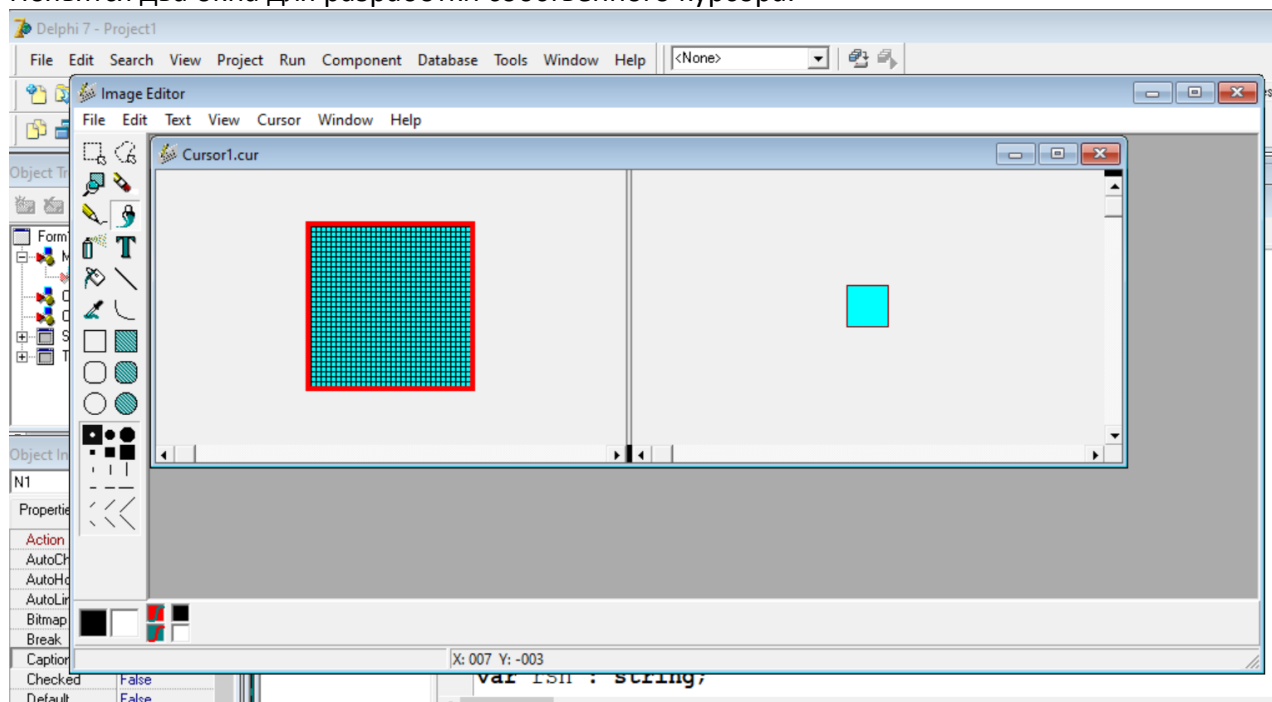
песочные часы – подсказывающий исполнителю, что идет процесс с задержкой работы программы. По окончании задержки, курсор изменяет свой вид.

Приступим к созданию собственных курсоров.

В меню Image Editor выберем пункт **Cursor File (.cur)**



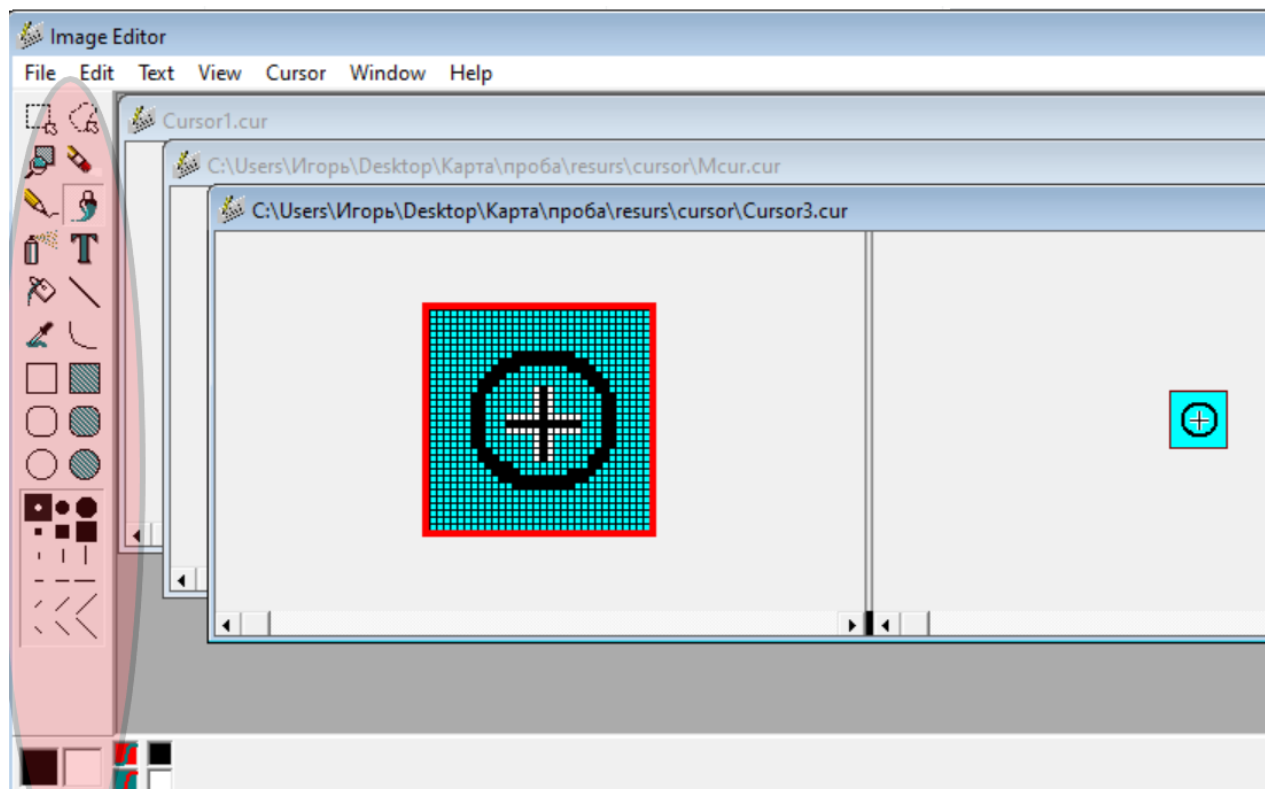
Появятся два окна для разработки собственного курсора.



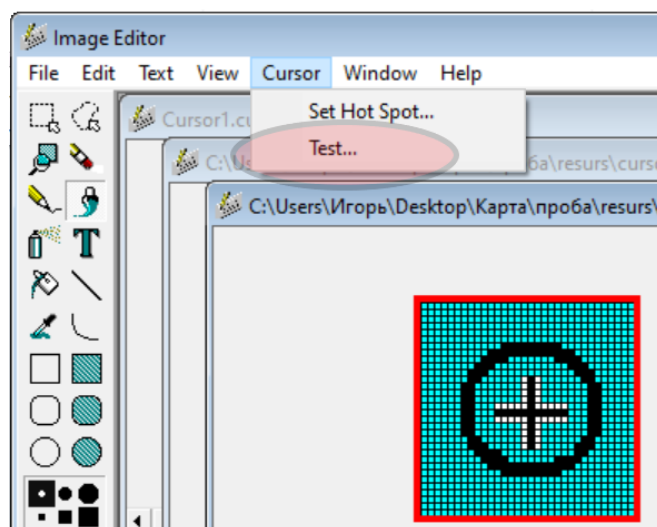
Правое окно – показывает, как будет выглядеть ваш курсор в вашей программе .

левое окно – для разработки собственного курсора.

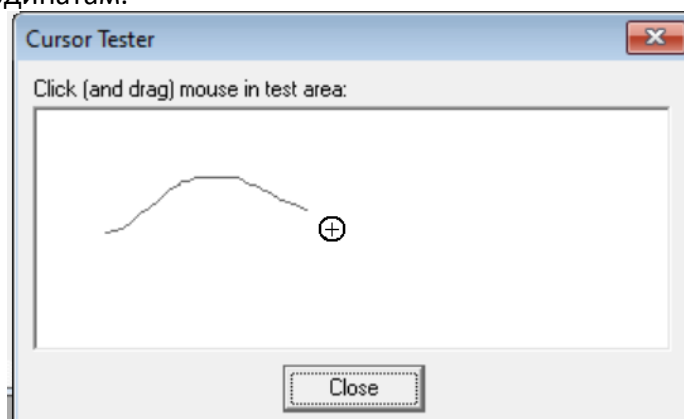
С помощью набора инструментов Image Editor (слева, выделено розовым), создадим курсор для плюсового (+) увеличения изображения примерно следующего вида



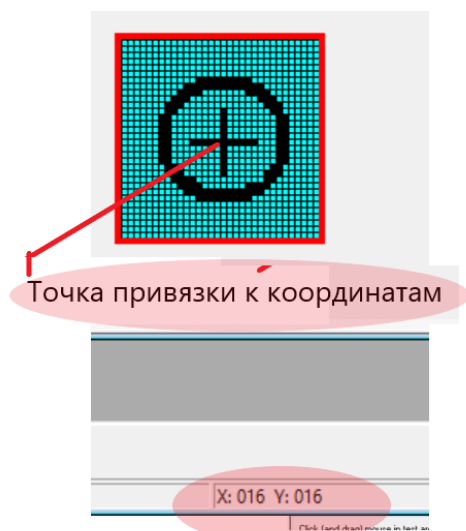
и протестируем его. Для тестирования курсора в меню Image Editor выберем кнопку Cursor->Test



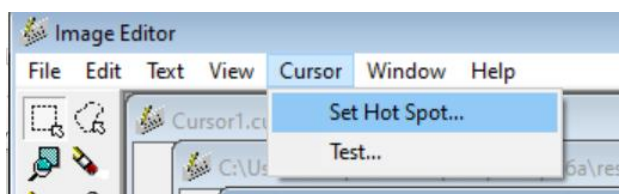
Появится окно с разработанным курсором, в котором его можно перемещать и отслеживать точку привязки к координатам.



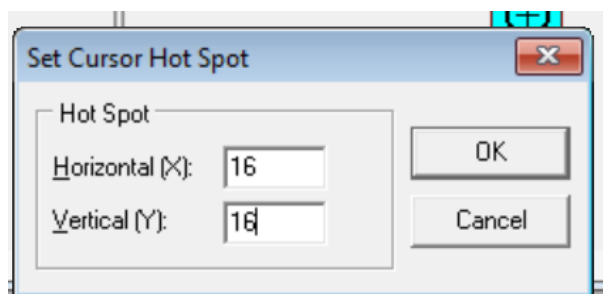
Для этого, необходимо с нажатой левой кнопкой мыши подвигать курсор в окне тестирования. Линия карандаша покажет, от какой точки берутся координаты. В нашем случае координаты определяются относительно левого верхнего угла окна разработки курсора. Для того, чтобы задать «горячую» точку на курсоре, относительно которой будут определяться текущие координаты, необходимо в окне разработки курсора определить точку привязки к координатам - X: 016 Y: 016



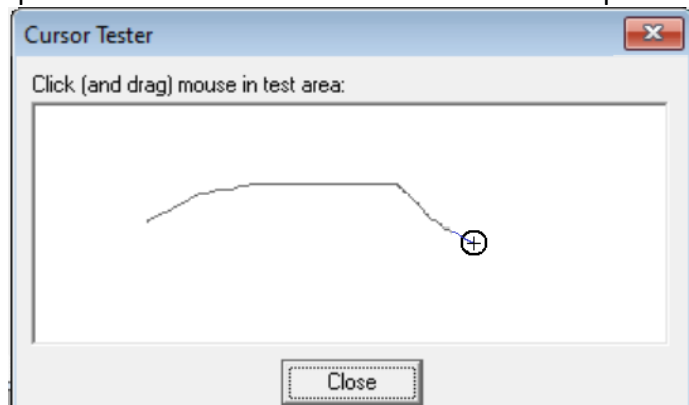
Затем в меню Image Editor выбрать кнопку Cursor->Set Hot Spot



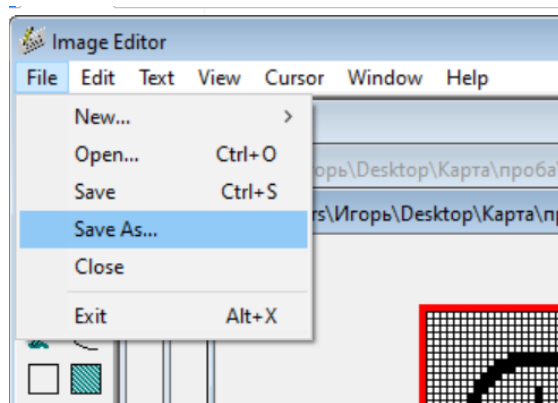
И в появившемся окне



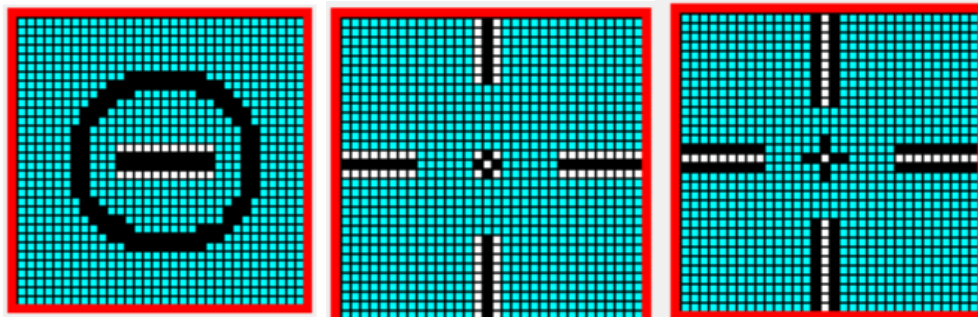
Задать координаты «горячей» точки. После чего выполнить повторное тестирование.



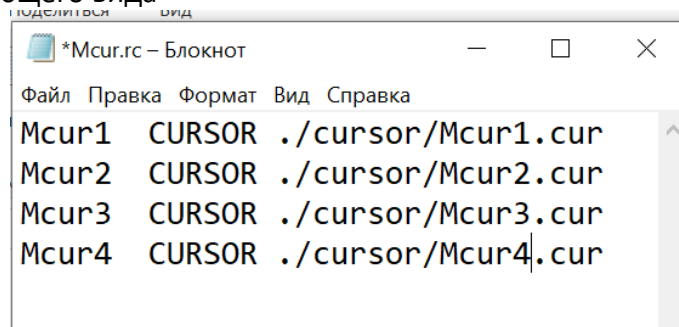
После того, как курсор оттестирован, сохраняем его под своим именем в нужную папку.



Далее, таким же образом, создадим курсоры следующих видов



Для хранения и использования курсоров создадим ресурсный файл. Для этого, в БЛОКНОТЕ создадим файл следующего вида



Где в строке:

Mcur1 CURSOR ./cursor/Mcur1.cur

Mcur1 – имя курсора в программе для его загрузки

CURSOR - идентификатор ресурса курсора

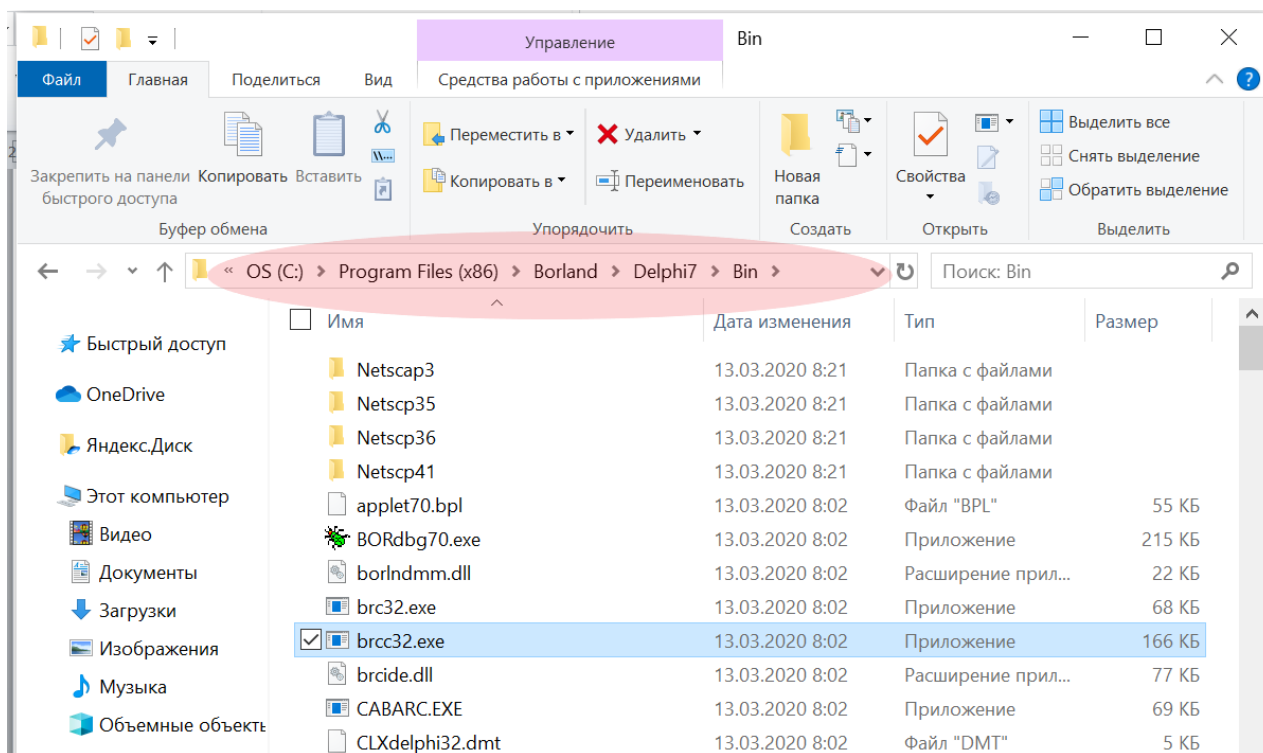
./cursor/ - путь к файлам курсора (Файлы курсоров находятся в папке «курсор»)

Mcur1.cur – имя файла (битовой карты) курсора.

Созданный файл сохраним с расширением .rc – например Mcur.rc в папку, куда вложена папка с курсорами.

Созданный файл необходимо откомпилировать с помощью программы brcc32.exe.

Данный файл расположен на системном диске, в моем случае это диск C:/ На рисунке ниже, розовым цветом выделен путь по которому вы найдете необходимый файл.

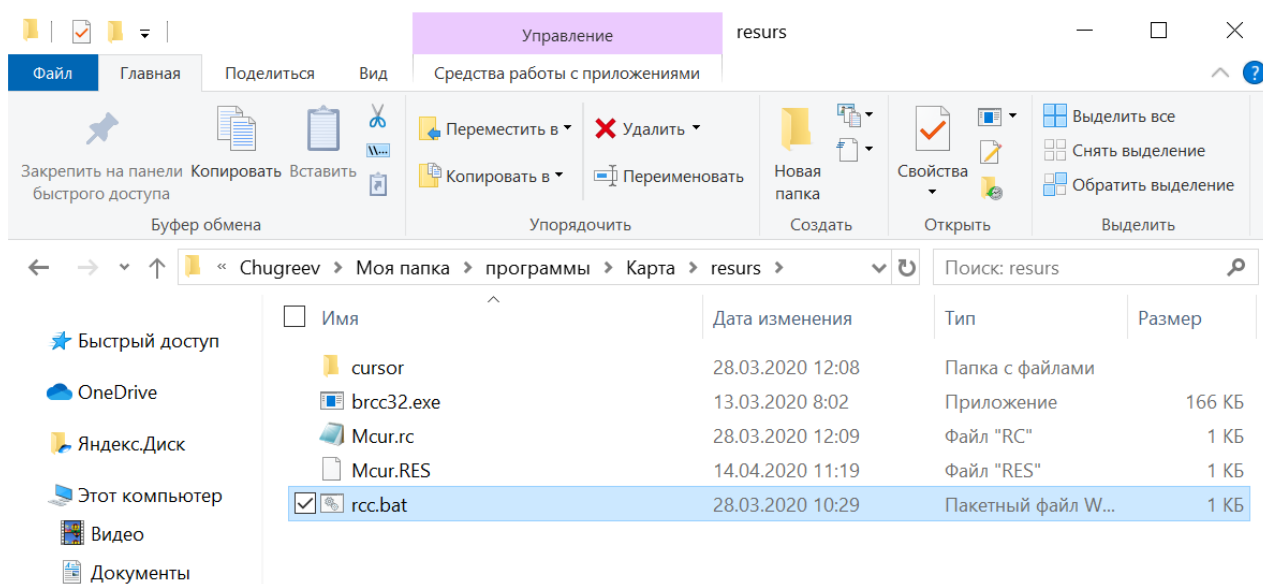


Данный файл необходимо скопировать в папку, где хранится файл **Mcur.rc**.

Затем в БЛОКНОТЕ создадим командный файл для компиляции файла **Mcur.rc**:

brcc32.exe Mcur.rc

и сохраним его с расширением **.bat** в ту же папку, где хранится **Mcur.rc**. Например **rcc.bat**



На данном рисунке представлена папка «ресурс» в которую вложена папка с файлами курсоров «курсор».

Дважды щелчком мышкой по **rcc.bat** – если в файле **Mcur.rc** все в порядке, создастся ресурсный файл **Mcur.res**.

Данный файл необходимо скопировать в папку, где находятся UNIT_ы разрабатываемой программы.

На сегодня достаточно. Продолжение следует.

Задание 5

В данном уроке научимся работать с ресурсными файлами.

И так, мы создали ресурсный файл курсоров - **Mcur.res** и сохранили его в папке разрабатываемого приложения. Теперь его необходимо включить в разрабатываемое приложение. Для этого, в области **IMPLEMENTATION** головного модуля UNIT1, прописываем директиву **{\$R Mcur.res}** – подключить ресурсный файл.

.....
implementation

{\$R *.dfm}
{\$R Mcur.res}

.....

Здесь :

\$ – данный символ в фигурных скобках { } означает, что после него следует ключ системной команды паскаля;

\$R – означает – подключить ресурсный файл;

{\$R *.dfm} – подключить все ресурсные файлы с расширением **dfm** – (Delpfi Form Menadger) --отвечающий за расположение всех компонентов на форме. Данный ключ прописывается автоматически во всех UNIT_ах, поскольку файлы DFM создаются автоматически.

Собственные, разработанные, ресурсные файлы подключаются вручную:
{\$R Mcur.res}, с указанием имени ресурсного файла - **Mcur.res**.

Теперь разработанные нами курсоры включены в разрабатываемую программу. Осталось только воспользоваться ими по мере необходимости.



Щелкнем дважды мышкой по кнопке. Получим обработчик события:

В данный обработчик добавим команду – загрузить из ресурсного файла соответствующий курсор и указываем область – компонент, в котором он будет работать.

```
procedure TForm1.SpeedButton1Click(Sender: TObject);  
begin  
    us_uv:=0; // увеличение плюс  
    us_uv1:=1; // выполнить увеличение плюс  
    screen.cursors[5]:=LoadCursor(HInstance,'Mcur3');//загружаем курсор  
    PaintBox1.Cursor:=5; // устанавливаем область действия курсора  
end;
```

Загрузка курсора и дальнейшее с ним общение производится по номеру - **screen.cursors[5]** – номер можно задать произвольно.



то же самое сделаем для кнопки

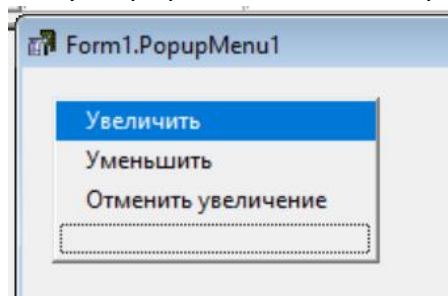
```

procedure TForm1.SpeedButton2Click(Sender: TObject);
begin
    us_uv:=1;
    us_uv1:=1;
    screen.cursors[5]:=LoadCursor(HInstance,'Mcur4');
    PaintBox1.Cursor:=5;
end;

```

Теперь, при выполнении команды - увеличить/уменьшить изображение - будет отображаться соответствующий курсор, своей формой показывая, какой режим включен в данный момент.

Добавим соответствующие коды в «всплывающее меню» - PopUpMenu1. Для этого, дважды щелкнем по компоненту PopUpMenu1. Появится разработчик меню.



Дважды щелкнем по кнопке «Увеличить», появится обработчик события, в который добавим код - загрузить из ресурсного файла соответствующий курсор и указать область его действия.

```

procedure TForm1.N4Click(Sender: TObject);
begin
    us_uv:=0;
    us_uv1:=1;
    screen.cursors[5]:=LoadCursor(HInstance,'Mcur3');
    PaintBox1.Cursor:=5;
end;

```

То же самое сделаем с кнопкой «Уменьшить».

```

procedure TForm1.N5Click(Sender: TObject);
begin
    us_uv:=1;
    us_uv1:=1;
    screen.cursors[5]:=LoadCursor(HInstance,'Mcur4');
    PaintBox1.Cursor:=5;
end;

```

Для кнопки «Отменить увеличение» добавим код

```

procedure TForm1.N6Click(Sender: TObject);
begin
    us_uv1:=0; // отменить режим увеличения
    PaintBox1.Cursor:=crDefault; // курсор по умолчанию - стрелка
end;

```

Запустим программу и протестируем ее.

Задание 6

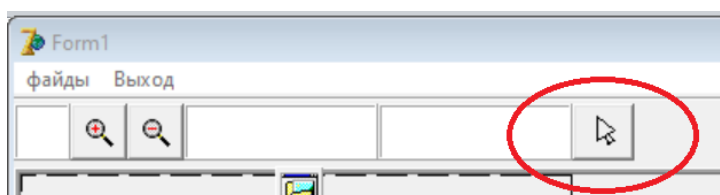
В данном уроке продолжаем учиться работать с ресурсными файлами.

В зависимости от задач, при работе с картами или снимками, требуется устанавливать тот или иной курсор для удобства получения информации.

Такой информацией может быть:

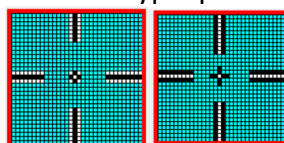
- ✓ Получение координат точек объекта
- ✓ Семантическая информация (характеристика объекта)
- ✓ Необходимость редактирования объекта и ряд других задач.

Для удобства переустановки курсора добавим на тоолбар еще один speedbutton.

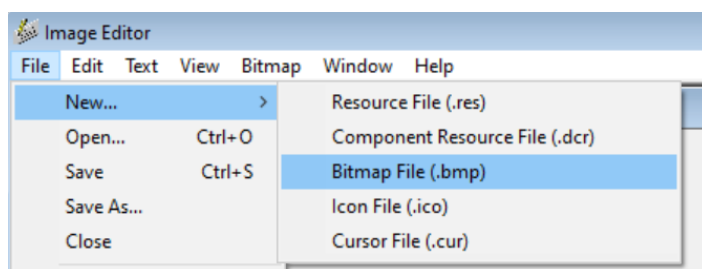


1. В Image Editor создадим файл типа BMP с изображением стандартного курсора и сохраним его в папке разработанных курсоров. В моем случае – bitmap2.bmp. С помощью свойства Glyph speedbutton_a (в инспекторе объектов) «подклеим» полученную картинку стандартного курсора.

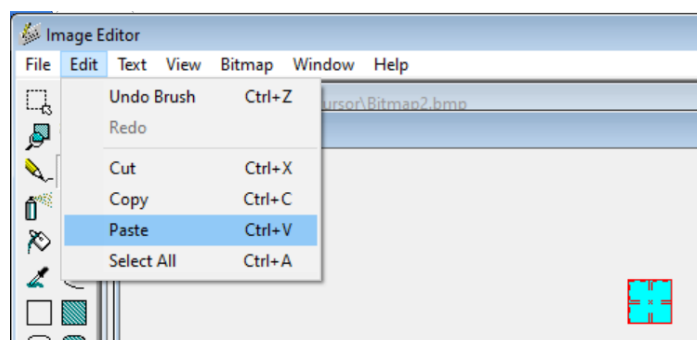
В Image Editor откроем ранее разработанные курсоры:



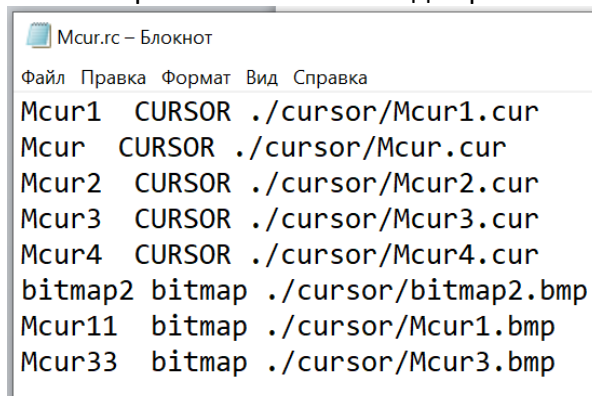
Затем, последовательно, каждый курсор скопируем, и откроем в Image Editor



раздел создания новой картинки типа BMP. Зададим размер 30x30 и вставим скопированный курсор



Изображение курсоров сохраним в папке разработанных курсоров, как файлы типа BMP . Эти файлы мы будем «подклеивать» на speedbutton, чтобы визуально видеть, какой рабочий курсор мы подгрузили.
Дополним наш ресурсный файл вновь созданными картинками. Откроем в БЛОКНОТЕ файл Mcur.rc и дополним его нашими картинками типа BMP для рабочих курсоров.



Дополненный файл сохраним. Файл с расширением RES удалим. С помощью ранее созданного командного файла **rcc.bat** создадим новый ресурсный файл **Mcur.res**. Данный файл сохраним в папке разрабатываемого проекта.

2. Дополним обработчик события создания формы – на выделенную панель выведем коэффициент увеличения загружаемой картинки



```
procedure TForm1.FormCreate(Sender: TObject);
begin
    bm:=tbitmap.Create;
    Panel4.Visible:=false;
    Toolbar1.Visible:=false;
    panel2.Caption:='1'; // вывели коэффициент увеличения
end;
```

3. В интерфейсной части, в разделе описания переменных добавим новую переменную (выделена красным) – счетчик рабочих курсоров:

```
.....
var
    Form1: TForm1;
    Bm : tbitmap;
    rect:trect;
    uv : integer=1;
    uv1:real=1.0;
    Drag : boolean=false;
    p,p1 : tpoint;
    xp,yp,i,j :integer;
    us_uv,us_uv1 : byte;
    gp : tjpegimage;
```

Cur : byte=1;

implementation

{ \$R *.dfm }

{ \$R Mcur.res }

.....

Замечание: При загрузке проекта курсор по умолчанию – crDefault, что соответствует значению счетчика = 1.

4. Мышкой, дважды щелчком по новому speedbutton_y, получим обработчик события

procedure TForm1.SpeedButton3Click(Sender: TObject);

begin

end;

в который запишем следующие директивы:

procedure TForm1.SpeedButton3Click(Sender: TObject);

begin

cur:=cur+1;

if cur>3 then cur:=1;

M_cursor;

end;

Здесь:

cur:=cur+1; – изменили номер счетчика;

if cur>3 then cur:=1; - выполнили проверку на количество курсоров – в нашем случае их три.

M_cursor; - обратились к процедуре переустановки курсора.

5. Создадим процедуру переустановки курсора. Для этого, в интерфейсной части, в разделе описания типов **TYPE** добавим объявление (выделено красным):

.....

procedure SpeedButton3Click(Sender: TObject);

procedure M_Cursor;

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

Bm : tbitmap;

.....

После чего, в области разработки кода – implementation запишем (ручками) код процедуры переустановки курсора:

implementation

```
{ $R *.dfm }  
{ $R Mcur.res }
```

```
procedure TForm1.M_Cursor;  
begin  
  case cur of // в зависимости от счетчика курсора  
    1: begin  
      paintbox1.Cursor:=crDefault;  
      speedbutton3.Glyph.LoadFromResourceName(Hinstance,'Bitmap2');  
    end;  
    2: begin  
      screen.Cursors[5]:=loadcursor(Hinstance,'Mcur1');  
      paintbox1.Cursor:=5;  
      speedbutton3.Glyph.LoadFromResourceName(Hinstance,'mcur33')  
    end;  
    3: begin  
      screen.Cursors[5]:=loadcursor(Hinstance,'Mcur2');  
      paintbox1.Cursor:=5;  
      speedbutton3.Glyph.LoadFromResourceName(Hinstance,'mcur11')  
    end;  
  end;  
end;  
end;
```

Здесь:

paintbox1.Cursor:=crDefault; - если счетчик равен 1 то курсор дельфи по умолчанию - стрелка

speedbutton3.Glyph.LoadFromResourceName(Hinstance,'Bitmap2'); - загружаем и подклеиваем картинку из ресурсного файла на speedbutton, с изображением текущего курсора

screen.Cursors[5]:=loadcursor(Hinstance,'Mcur1'); - загружаем курсор из ресурсного файла

paintbox1.Cursor:=5; - устанавливаем область действия курсора - paintbox1

6. В всплывающем меню PopUpMenu1, в обработчик события кнопки «Отменить увеличение», добавляем директиву:

```
procedure TForm1.N6Click(Sender: TObject);  
begin  
  us_uv1:=0;  
  M_Cursor;  
end;
```

а директиву **PaintBox1.Cursor:=crDefault** – убираем.

Запустим программу и протестируем смену курсоров.

Задание 7

В данном задании выполним подготовку приложения для ориентирования листа карты.

Выполнить ориентирование – это получение математических коэффициентов, которые позволят нам перейти от экранных координат монитора к реальным координатам карты.

Дополним наше приложение.

1. Объявим переменные, необходимые для ориентирования. В интерфейсной части юнита, дополним раздел описания переменных:

.....

public

{ Public declarations }

end;

var

Form1: TForm1;

Bm : tbitmap;

rect:trect;

uv : integer=1;

uv1:real=1.0;

Drag : boolean=false;

p,p1 : tpoint;

xp,yp,i,j :integer;

us_uv,us_uv1 : byte;

gp : tjpegimage;

Cur : byte=1;

path : string; // путь для сохранения данных (диск, папка)

ss,ss1,ss2,ss3: string; // переменные для преобразования типов данных

x,y : array of real; // координаты опорных точек

xi,yi : array of real; // координаты измеренных точек

dx,dy : array of real; // приращения координат опорных точек

dxi,dyi,s,si : array of real; // приращения координат измеренных точек, S- расстояние вычисленное в реальных координатах, Si - расстояние вычисленное в Экранных координатах – пикселях изображения

kx,ky,al,ali,dal : array of real;// kx,ky – коэффиц. Преобразования экранных координат к реальным, al – дирекционный угол реальных координат, ali – измеренных координат, dal – угол доворота измеренного дир. Угла к реальному

t : textfile; // файловая переменная

nt : byte; // номер измеренной точки

us_msh : byte=0; // флаг масштаба

x_izm,y_izm,alfa,als : real; // x_izm,y_izm ,alfa – промежуточные переменные для вычислений, als – средний угол доворота

kxs : real=1.0; // kxs, kys – средние коэфф. преобразования измеренных координат

kys : real=1.0;

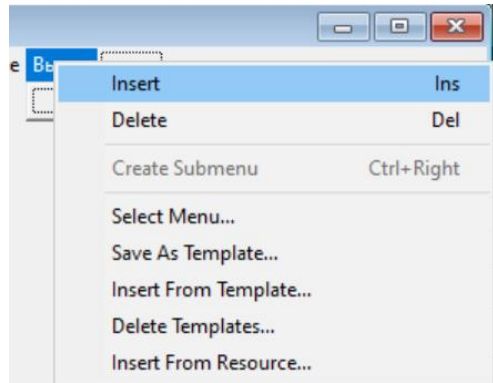
implementation

-
2. На форму бросим еще одну панель. Свойству Align (выравнивание) панели в инспекторе объектов зададим AlClitnt – по всей форме. Свойству Visible панели присвоим False. Кстати, форме тоже присвойте Align_ AlClitnt_
 3. На панель бросьте компонент StringGrid. В инспекторе объектов зададим свойства: Align – AlClitnt; ColCount – 15; DefoltColWifth -90; RowCount – 10.
 4. Дополним проседуру создания формы:

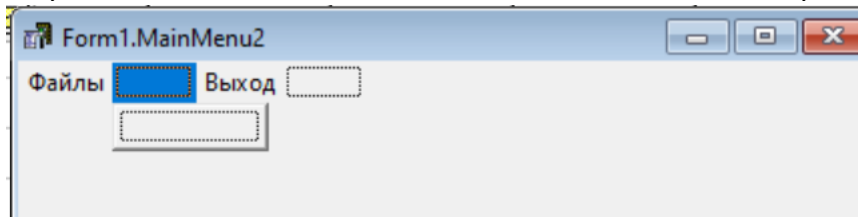
```
procedure TForm1.FormCreate(Sender: TObject);
begin
  bm:=tbitmap.Create;
  Panel4.Visible:=false;
  Toolbar1.Visible:=false;
  Panel5.Align:=AlNone; //выравнивание для новой панели
  panel2.Caption:='1';
  stringgrid1.Cells[0,0]:=' N/N';
  stringgrid1.Cells[1,0]:=' X';
  stringgrid1.Cells[2,0]:=' Y';
  stringgrid1.Cells[3,0]:=' dX';
  stringgrid1.Cells[4,0]:=' dY ';
  stringgrid1.Cells[5,0]:=' S';
  stringgrid1.Cells[6,0]:=' X izm';
  stringgrid1.Cells[7,0]:=' Y izm ';
  stringgrid1.Cells[8,0]:=' dX izm ';
  stringgrid1.Cells[9,0]:=' DY izm ';
  stringgrid1.Cells[10,0]:=' S izm ';
  stringgrid1.Cells[11,0]:=' KS';
  stringgrid1.Cells[12,0]:=' ALFA';
  stringgrid1.Cells[13,0]:=' ALFA izm ';
  stringgrid1.Cells[14,0]:=' d_ALFA ';
  setlength(x,1);
  setlength(y,1);
  setlength(xi,1);
  setlength(yi,1);
  setlength(dx,1);
  setlength(dy,1);
  setlength(dxi,1);
  setlength(dyi,1);
  setlength(kx,1);
  setlength(s,1);
  setlength(si,1);
  setlength(al,1);
  setlength(ali,1);
  setlength(dal,1);
end;
```

5. В основное метю – MainMenu1 – добавим кнопки

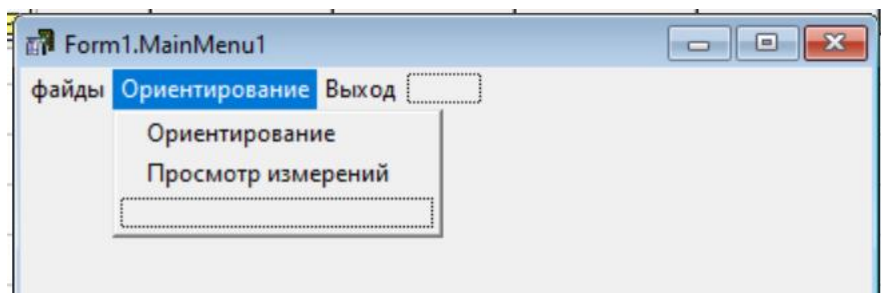
Для этого дважды щелкнем по компоненту MainMenu1. Затем выделим кнопку «Выход» и щелкнем правой кнопкой мышки. Появится окно команд:



Выберем кнопку **Insert** и щелкнем левой кнопкой мышки. Появится пустая кнопка.



Оформим как на рисунке



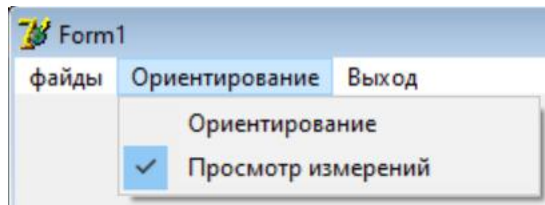
6. Дважды щелкнем по кнопке «Просмотр измерений». Получим обработчик событий **procedure TForm1.N8Click(Sender: TObject);**
begin

end;

в который запишем код для просмотра измерений в StringGrid_e :

```
procedure TForm1.N8Click(Sender: TObject);
begin
  If n8.Checked=false then begin
    Panel5.Visible:=true;
    n8.Checked:=true;
  end
  else begin
    Panel5.Visible:=False;
    n8.Checked:=false;
  end;
end;
```

Свойство Checked=true - это галочка впереди кнопки:



Если Checked=False – то галочка отсутствует.

7. Бросим на форму компонент OpenDialog.
8. В папку, где лежат карты, вставим текстовый файл с опорными точками. Опорными точками будут служить пересечения линий километровой сетки:

Опора.txt – Блокнот

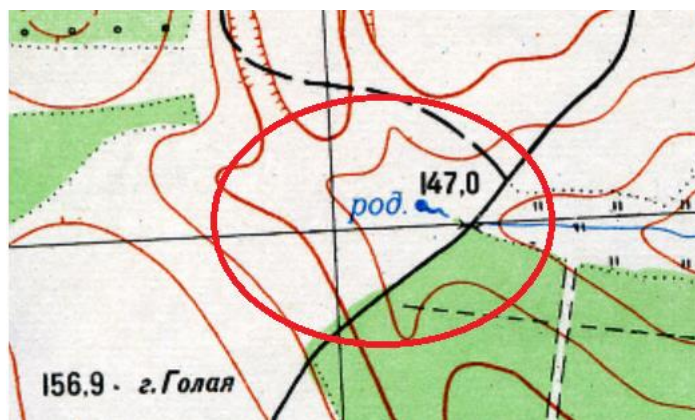
Файл	Правка	Формат	Вид	Справка
1	6066000.0		4312000.0	
2	6066000.0		4311000.0	
3	6065000.0		4311000.0	
4	6065000.0		4314000.0	
5	6066000.0		4314000.0	
6	6068000.0		4314000.0	
7	6068000.0		4311000.0	

Например точка №1 имеет координаты

X = 6066000.0 м

Y = 4312000.0 м

Это пересечение линий километровой сетки:



9. Вставим в проект ранее разработанные процедуры – OGZ и RG. Для этого в раздел описания типов интерфейсной части юнита, добавим объявления

```

.....
type
  TForm1 = class(TForm)  procedure N8Click(Sender: TObject);
.....
    procedure N9Click(Sender: TObject);
      procedure ogz(x,y :real);
      procedure rg(a :real);

  private
    { Private declarations }
  Public
.....

```

а сами процедуры вставим в исполняемую часть юнита:

implementation

```
{ $R *.dfm }  
{ $R Mcur.res }
```

```
procedure TForm1.rg(a :real);  
var g,g1,m,m1,s : real;  
begin  
  g:=a/pi*180;  
  g1:=trunc(g);  
  m:=(g-g1)*60;  
  m1:=trunc(m);  
  s:=(m-m1)*60;  
  str(g1:5:0,ss);  
  str((abs(m1)):5:0,ss1);  
  str((abs(s)):6:1,ss2);  
end;  
  
procedure TForm1.ogz(x,y :real);  
var r: real;  
begin  
  if x=0 then begin  
    if y>0 then alfa:=pi/2  
    else alfa:=3*pi/2;  
    exit;  
  end;  
  if y=0 then begin  
    if x>0 then alfa:=0  
    else alfa:=pi;  
    exit;  
  end;  
  r:=arctan(y/x);  
  if (x>0)and(y>0) then alfa:=r;  
  if (x<0)and(y>0) then alfa:=r+pi;  
  if (x<0)and(y<0) then alfa:=r+pi;  
  if (x>0)and(y<0) then alfa:=r+2*pi;  
end;
```

10. Щелкнем по кнопке «Ориентирование». Получим обработчик события:

```
procedure TForm1.N9Click(Sender: TObject);  
begin  
  
end;
```

в который запишем следующий код:

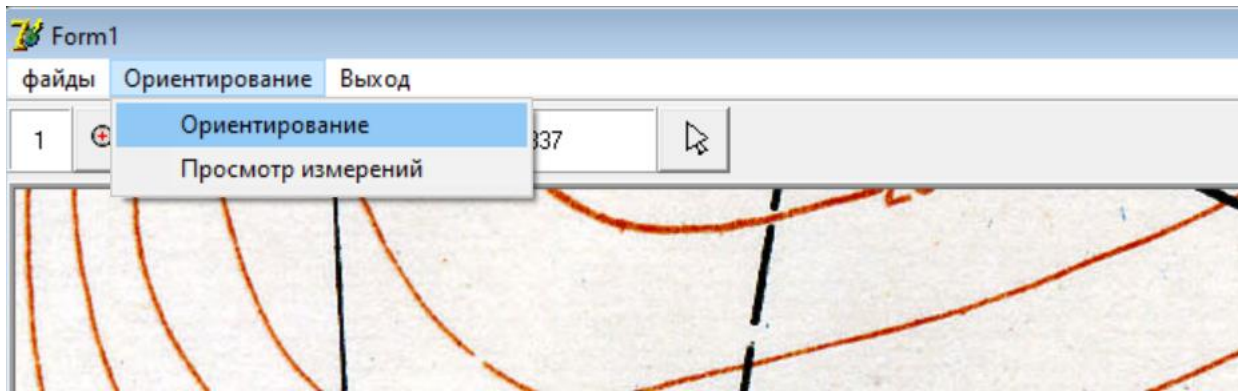
```
procedure TForm1.N9Click(Sender: TObject);  
begin  
  if opendialog1.Execute then begin  
    assignfile(t,opendialog1.FileName); //открываем и грузим файл опорных
```

```

reset(t); // точек
path:=extractfilepath(opendialog1.FileName); // выделяем путь к каталогу
i:=0; // начальное значение счетчика опоры
while not eof(t) do begin
  if length(x)<=i then begin
    setlength(x,length(x)+1);
    setlength(y,length(y)+1);
    setlength(s,length(s)+1);
    setlength(si,length(si)+1);
    setlength(xi,length(xi)+1);
    setlength(yi,length(yi)+1);
    setlength(dx,length(dx)+1);
    setlength(dy,length(dy)+1);
    setlength(dxi,length(dxi)+1);
    setlength(dyi,length(dyi)+1);
    setlength(kx,length(kx)+1);
    setlength(al,length(al)+1);
    setlength(ali,length(ali)+1);
    setlength(dal,length(dal)+1);
  end;
  readln(t,j,x[i],y[i]); // читаем координаты в массив данных
  ss:=floattostrf(x[i],ffFixed,12,2); // преобразуем координаты в текст
  ss1:=floattostrf(y[i],ffFixed,12,2);
  stringgrid1.Cells[0,i+1]:=inttostr(j); // заполняем таблицу – номер точки
  stringgrid1.Cells[1,i+1]:=ss; // координата X
  stringgrid1.Cells[2,i+1]:=ss1; // координата Y
  if i<>0 then begin
    dx[i]:=(x[i]-x[0]); // находим приращения координат относительно
    dy[i]:=(y[i]-y[0]); // первой точки
    ss:=floattostrf(dx[i],ffFixed,12,2); // преобразуем приращения в текст
    ss1:=floattostrf(dy[i],ffFixed,12,2);
    stringgrid1.Cells[3,i+1]:=ss; // заполняем таблицу
    stringgrid1.Cells[4,i+1]:=ss1;
    ogz(dx[i],dy[i]); // определяем дирекционный угол текущей точки
    al[i]:=alfa; // относительно первой точки
    rg(alfa); // преобразуем угол в текст
    stringgrid1.Cells[12,i+1]:=ss+ss1+ss2; // заполняем таблицу
    s[i]:=sqrt(dx[i]*dx[i]+dy[i]*dy[i]); // вычисляем расстояние между точками
    ss:=floattostrf(S[i],ffFixed,12,2); // преобразуем в текст
    stringgrid1.Cells[5,i+1]:=ss; // заполняем таблицу
  end;
  inc(i); // увеличиваем счетчик цикла на 1
end; // while not eof(t) // продолжаем чтение
closefile(t);
  us_msh:=1; // задали флаг – выполняются измерения опорных точек.
end; // if opendialog1.Execute
end;

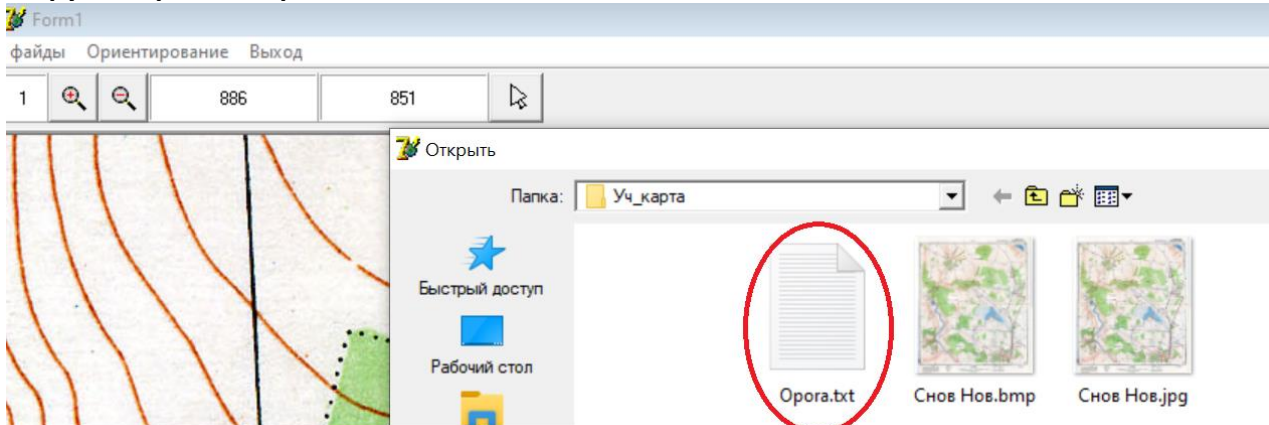
```

Откомпилируйте проект и исправьте ошибки. Запустите проект. Загрузите карту. Затем войдите в меню

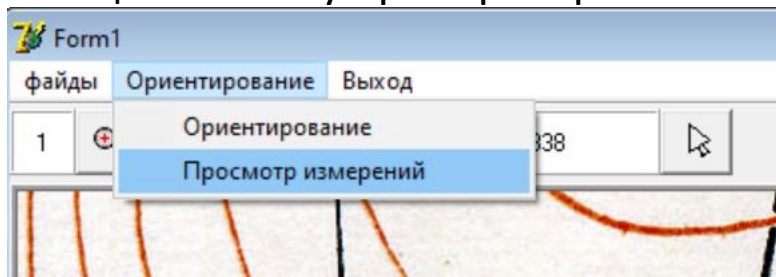


И щелкните по кнопке «Ориентирование»

Загрузите файл опорных точек



Опять войдите в меню и щелкните кнопку «просмотр измерений»



Откроется таблица загруженных опорных точек, и вычисленных значений

N/N	X	Y	dX	dY	S	X изм	Y изм	dX изм	dY изм	S изм	KS	ALFA	ALFA изм
1	6066000.00	4312000.00											
2	6066000.00	4311000.00	0.00	-1000.00	1000.00							270 0 0.0	
3	6065000.00	4311000.00	-1000.00	-1000.00	1414.21							225 0 0.0	
4	6065000.00	4314000.00	-1000.00	2000.00	2236.07							116 33 54.2	
5	6066000.00	4314000.00	0.00	2000.00	2000.00							90 0 0.0	
6	6068000.00	4314000.00	2000.00	2000.00	2828.43							45 0 0.0	
7	6068000.00	4311000.00	2000.00	-1000.00	2236.07							333 26 5.8	

При повторном нажатии на кнопку «просмотр измерений» - таблица закроется. Вернемся на карту.

Задание 8

Продолжаем предыдущее задание.

Поскольку в математике вы еще не сильны (не знаете, что такое дробно-линейное трансформирование, метод наименьших квадратов.....), будем использовать тот математический аппарат, с которым вы познакомились на геодезии – **обратную геодезическую задачу и прямую геодезическую задачу.**

Суть ориентирования карты сведется к следующему:

1. Первая опорная точка должна находиться как можно ближе к центру карты. Она будет являться **реперной** точкой, или точкой, от которой будут вычисляться приращения координат до всех измеренных точек на карте.
2. Для всех опорных точек найдем приращения координат относительно первой опорной точки:

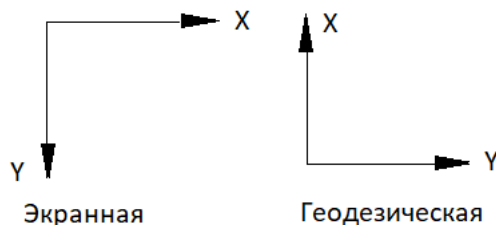
$$dx_i := x_i - x_0.$$

$$dy_i := y_i - y_0.$$

Используя обратную геодезическую задачу, по приращениям координат опорных точек (dx и dy) найдем направления (дирекционные углы - α) с точки 1 на все остальные точки, а так же расстояния - S_i

3. Выполним измерения опорных точек в системе координат изображения. Получим X_i _изм, Y_i _изм. Найдем приращения измеренных координат относительно первой опорной точки в геодезической (пока условной) системе координат.

При этом необходимо учесть систему координат экрана компьютера и геодезическую систему координат:



Для перехода от экранных координат к геодезическим (условным) необходимо выполнить простые преобразования:

$$dX_{i_изм} := Y0_изм - Y_{i_изм};$$

$$dY_{i_изм} := X_{i_изм} - X0_изм;$$

Найдем по измеренным приращениям направления (дирекционные углы $-\alpha_{изм}$) с точки 1 на все остальные точки, а так же расстояния - $S_{i_изм}$.

4. Найдем коэффициент преобразования длин линий $S_{i_изм}$ и линий S_i – вычисленных по координатам опорных точек (масштабный коэффициент)

$$Kfs_i := S_i / S_{i_изм}.$$

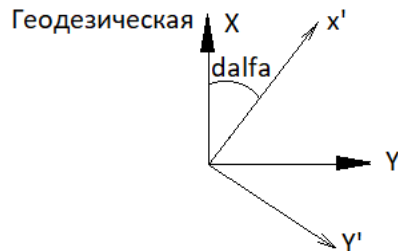
В идеальном случае данные коэффициенты, полученный по всем измеренным линиям должны быть одинаковыми, но поскольку присутствует ряд погрешностей, таких как: деформация бумаги листа карты, погрешности сканирования листа карты, погрешности тиражного оттиска (нанесение ситуации) на лист подложки (бумагу) карты, погрешности измерений – данный коэффициент будет различаться на некоторую малую величину. Поэтому усредним данный коэффициент:

$$Kfs := \text{сумма}(Kfs_i) / N$$

Где N – число вычисленных линий.

Данный коэффициент потребуется для преобразования расстояния полученного из измерений (в пикселях) к истинному расстоянию в метрах, полученному с карты с учетом ее масштаба.

5. Найдем угол разворота отсканированного листа карты относительно геодезической системы координат - dalfa_i . Для этого найдем разность дирекционных углов соответствующих направлений (угол доворота систем координат):



Здесь $X' Y'$ – система координат отсканированного листа карты.

$$\text{dalfa_i} := \text{alfa}_{i_{\text{изм}}} - \text{alfa_i}$$

Здесь

dalfa_i – угол доворота отсканированного листа карты к геодезической системе координат;

$\text{alfa}_{i_{\text{изм}}}$ – дирекционный угол из измерений;

alfa_i – дирекционный угол вычисленный по координатам опорных точек.

Как и в случае с коэффициентом Kfs , присутствует ряд погрешностей. Поэтому находим среднее значение из dalfa_i :

$$\text{dalfa} := \text{сумма}(\text{dalfa_i}) / N$$

6. Полученные значения – Kfs и dalfa – позволят нам преобразовать измеренные координаты в геодезические. Для этого необходимо определить на электронной карте условные (экранные) координаты, вычислить по условным координатам приращения координат измеренной точки, дирекционный угол $\text{alfa}_{i_{\text{изм}}}$ и расстояние S относительно первой точки

$$dX_{i_{\text{изм}}} := Y0_{\text{изм}} - Y_{i_{\text{изм}}};$$

$$dY_{i_{\text{изм}}} := X_{i_{\text{изм}}} - X0_{\text{изм}};$$

и используя алгоритм прямой геодезической задачи и полученные значения – Kfs и dalfa , вычислить геодезические координаты измеренной точки:

$$dx_o := (x_{\text{изм}} - x_{\text{изм}} [0]);$$

$$dy_o := (y_{\text{изм}} [0] - y_{\text{изм}});$$

$$s_o := \sqrt{dx_o^2 + dy_o^2} / kfs; \quad // \text{вычисляем истинное расстояние}$$

$$\text{ogz}(dy_o, dx_o); \quad // \text{определяем дир. угол на измеренную точку - } \text{alfa}$$

$$x_o := (x_{\text{изм}} + (s_o * \cos(\text{alfa} - \text{dalfa}))); \quad // \text{вычисляем геодезические координаты}$$

$$y_o := Y_{\text{изм}} + ((s_o * \sin(\text{alfa} - \text{dalfa}))); \quad // \text{прямая геодезическая задача}$$

x_o, y_o – прямоугольные геодезические координаты.

Необходимо отметить, что СКП тиражного оттиска карты составляет ~ 0.8 мм в масштабе карты. С учетом перечисленных ранее погрешностей, ожидаемая предельная погрешность определения координат данным способом может составить ~ 1.6 мм в масштабе карты.

Другие способы ориентирования, например дробно-линейное трансформирование - дадут более точный результат $\sim$ в 3-4 раза.

Приступим непосредственно к программированию нашего проекта.

В предыдущем задании мы загрузили в проект файл опорных точек, по координатам опорных точек вычислили приращения координат относительно первой точки, длины линий, дирекционные углы.

Продолжаем:

1. В предыдущем задании, мы установили флаг выполнения измерений опорных точек.

us_msh:=1; // задали флаг – выполняются измерения опорных точек.

Дополним обработчик события TForm1.PaintBox1MouseDown

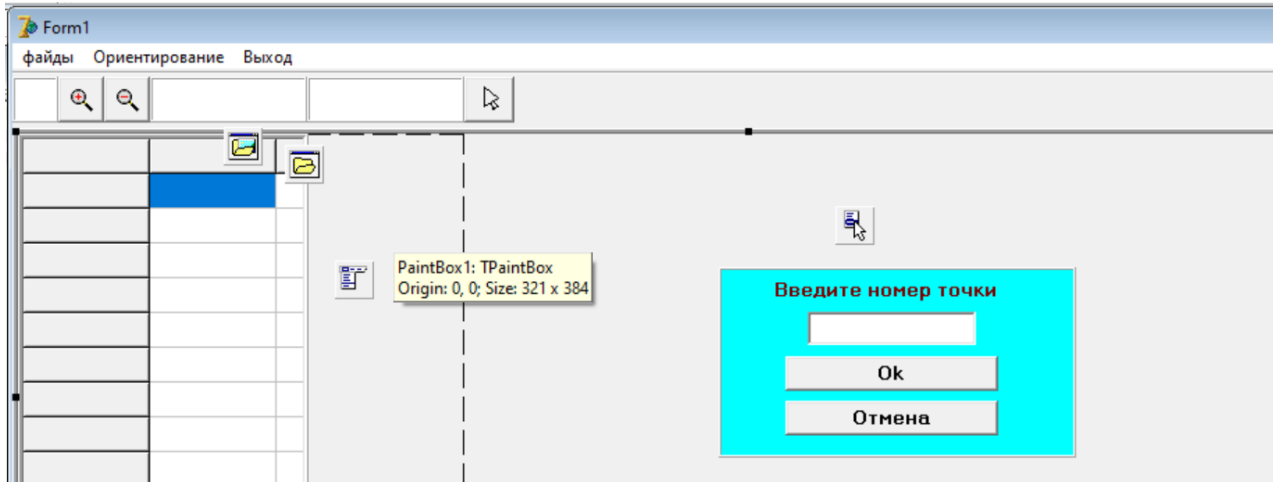
```
procedure TForm1.PaintBox1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer); //нажата
begin
  if (ssshift in shift) or (ssctrl in shift) then begin // перетаскиваем объект
    drag:=true;
    getcursorpos(p);
    exit;
  end;
  if (us_uv1=1) and (button=mbLeft) then begin
    uvelichenie(x,y);
  end;
end;
```

следующим кодом (выделено красным)

```
procedure TForm1.PaintBox1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer); //нажата
begin
  if (ssshift in shift) or (ssctrl in shift) then begin // перетаскиваем объект
    drag:=true;
    getcursorpos(p);
    exit;
  end;
  if (us_uv1=1) and (button=mbLeft) then begin //увеличиваем объект
    uvelichenie(x,y);
  end;
  if (us_msh>0) and (button=mbLeft) then begin //выполняем регистрацию точек
  if (us_uv1>0) and (drag=true) then begin
    exit;
  end;
  Case us_msh of // в дальнейшем us_msh будет отличаться от 1
  1: begin
    x_izm:=x/uv1; // в промежуточным переменным присваиваем текущие
    y_izm:=y/uv1; // X, Y - приведенные к увеличению карты = 1
    getcursorpos(p); // получаем экранные координаты курсора
    panel6.Left:=p.X; // левому верхнему углу панели присваиваем экранные
    panel6.Top:=p.Y-panel6.Height; // координаты
    panel6.Visible:=true; // делаем видимой Panel6 на экране
    edit1.Text:=""; // очищаем поле редактора
    edit1.SetFocus; // устанавливаем курсор в поле редактора
  end;
  end; //case
end;
```

end;

2. На панель, на которой находится компонент PaintBox, бросим еще одну панель и оформим ее как на рисунке ниже. Свойству панели Visible:=false;



Данная панель (**У меня она под № 6**) необходима для синхронизации номеров точек - измеренной и опорной. **Измеренная точка - ее номер - должна соответствовать опорной точке.**

Щелчком дважды по кнопке "Отмена" . Получим обработчик событий:

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

end;
```

в который запишем код

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  Panel6.Visible:=false;
end;
```

Щелчком дважды по кнопке "Ok" . Получим обработчик событий:

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

end;
```

в который запишем код:

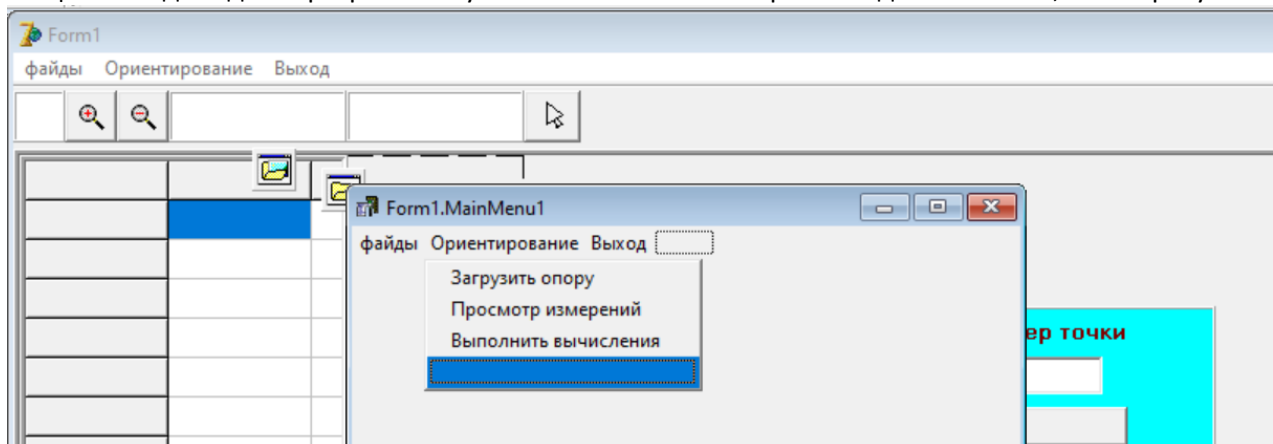
```
procedure TForm1.Button1Click(Sender: TObject);
var
  k : integer; //переменная кода преобразования
begin
  val(edit1.Text,nt,k);
  if k<>0 then begin
    showmessage('Ошибка ввода данных');
    exit;
  end;
  if nt>i then begin // i - количество опорных точек - соответствуют номеру
    showmessage(' Проверьте правильность введенного номера');
    exit;
```

```

end
else begin
  Xi[nt-1]:=x_izm; // точке с номером nt-1 присваиваем зарегистрированную
  yi[nt-1]:=y_izm; // координату
  ss:=floattostrf(xi[nt-1],ffFixed,12,2); // преобразуем в текст
  ss1:=floattostrf(yi[nt-1],ffFixed,12,2);
  stringgrid1.Cells[6,nt]:=ss; // выводим текст в свою позицию Stringgrid_a
  stringgrid1.Cells[7,nt]:=ss1;
  if (xi[0]>0)and(nt>1) then begin
    dxi[nt-1]:=(xi[nt-1]-xi[0]); // находим приращение координат измеренных
    dyi[nt-1]:=(yi[nt-1]-yi[0]); // точек
    str(dxi[nt-1]:12:5,ss);
    stringgrid1.Cells[8,nt]:=ss;
    str(dyi[nt-1]:12:5,ss);
    stringgrid1.Cells[9,nt]:=ss;
    si[nt-1]:=sqrt(dxi[nt-1]*dxi[nt-1]+dyi[nt-1]*dyi[nt-1]); //находим расстояния
    str(si[nt-1]:12:2,ss);
    stringgrid1.Cells[10,nt]:=ss;
    kx[nt-1]:=abs(si[nt-1]/s[nt-1]); // находим масштабный коэффициент
    ss:=floattostrf(kx[nt-1],ffFixed,12,2);
    stringgrid1.Cells[11,nt]:=ss;
    ogz(dyi[nt-1],dxi[nt-1]); // определяем дир. угол на измеренную
    ali[nt-1]:=alfa; // точку
    rg(alfa);
    stringgrid1.Cells[13,nt]:=ss+ss1+ss2;
    dal[nt-1]:=ali[nt-1]-al[nt-1]; // находим угол доворота систем координат
    rg(dal[nt-1]);
    stringgrid1.Cells[14,nt]:=ss+ss1+ss2;
  end; // if (xi[0]>0)and(nt>1) then begin
end; // else begin
panel6.Visible:=false; // удаляем панель с экрана
end;

```

3. Щелкнем дважды по разработчику меню MainMenu1. Исправим и дополним его, как на рисунке



щелкнем дважды по кнопке “Выполнить вычисления”. в обработчик события

```

procedure TForm1.N10Click(Sender: TObject);
begin

```

```

end;

```

запишем код

```

procedure TForm1.N10Click(Sender: TObject);
var j : word;

```

```

begin
als:=0; // обнуляем ячейки для суммирования dal[j] - угол доворота
kxs:=0; // обнуляем ячейки для суммирования kx[j] - масштабный коэфф.

for j:=1 to i-1 do begin
  kxs:=kxs+kx[j]; // суммируем
  als:=als+dal[j];
end;
kxs:=kxs/(i-1); // находим среднее
als:=als/(i-1);
x_izm:=x[0]; // x_izm - координата первой опорной точки
y_izm:=y[0]; // Y_izm - координата первой опорной точки
assignfile(t,'kxs.dat'); // открываем файл для сохранения данных
rewrite(t); // создаем пустой файл для сохранения
writeln(t,(i-1):3); // выводим количество точек
for i:=0 to i-1 do // выводим данные опоры измерений и вычислений
writeln(t,i+1:3,x[i]:12:2,y[i]:12:2,dx[i]:12:2,dy[i]:12:2,s[i]:12:2,xi[i]:12:2,yi[i]:12:2,
dxi[i]:12:2,dyi[i]:12:2,si[i]:12:2,kx[i]:12:10,al[i]:14:9,ali[i]:14:9,dal[i]:14:9);

str(kxs:12:10,ss);
str(als:12:8,ss1);
str(x_izm:12:2,ss2);
str(y_izm:12:2,ss3);
writeln(t,ss+' '+ss1+' '+ss2+' '+ss3);
str(xi[0]:12:8,ss);
str(yi[0]:12:8,ss1);
writeln(t,ss+' '+ss1);
closefile(t);
rg(als); // преобразуем средний угол доворота в текст
stringgrid1.Cells[13,9]:=ss+ss1+ss2;
str(kxs:10:3,ss);
stringgrid1.Cells[11,9]:=ss;
us_msh:=0; // флаг окончания измерений для ориентирования
us_model:=1; // флаг - модель построена
end;

```

4. Дополним проседуру **procedure TForm1.PaintBox1MouseMove**:

```

procedure TForm1.PaintBox1MouseMove(Sender: TObject; Shift: TShiftState; X,
Y: Integer);
var dx,dy : integer;
begin
  panel1.Caption:=inttostr(x);
  panel3.Caption:=inttostr(y);
  if (drag=true) then begin // перетаскиваем объект
    getcursorpos(p1);
    dx:=round((p1.X-p.x) * 2.1);
    dy:=round((p1.y-p.y) * 2.1);
    Scrollbox1.HorzScrollBar.Position:=Scrollbox1.HorzScrollBar.Position-dx;
    Scrollbox1.VertScrollBar.Position:=Scrollbox1.VertScrollBar.Position-dy;
    getcursorpos(p);
  end; // перетаскиваем объект
end;

```

в которую добавим код

```

procedure TForm1.PaintBox1MouseMove(Sender: TObject; Shift: TShiftState; X,
Y: Integer);
var
    dx_o,dy_o,s_o,a,xo,yo: real;
    dx,dy : integer;
begin
    if us_model=1 then begin // если модель построена
        dx_o:=(x/uv1-xi[0]); //находим приращения координат по измерениям
        dy_o:=(yi[0]-y/uv1);
        s_o:=sqrt(dx_o*dx_o+dy_o*dy_o)/kxs; //находим расстояния по измерениям с учетом
//масштабного коэффициента
        ogz(dy_o,dx_o); // находим дир. угол по измерениям
        xo:=(x_izm+(s_o*cos(alfa-als)));// Прямая геодезическая задача с учетом угла доворота
        yo:=Y_izm+((s_o*sin(alfa-als)));// определяем реальные координаты
        ss:=floattostrf(xo,ffFixed,10,2); //преобразуем вычисленные координаты модели
        ss1:=floattostrf(yo,ffFixed,10,2);
    end// если модель построена
    else begin //если модель не построена
        ss:=floattostrf(x,ffFixed,10,2); // преобразуем текущие координаты картинки
        ss1:=floattostrf(y,ffFixed,10,2);
    end;
    panel1.Caption:=ss; //выводим координаты
    panel3.Caption:=ss1;
    if (drag=true) then begin //перетаскиваем объект
        getcursorpos(p1);
        dx:=round((p1.X-p.x) * 2.1);
        dy:=round((p1.y-p.y) * 2.1);
        Scrollbox1.HorzScrollBar.Position:=Scrollbox1.HorzScrollBar.Position-dx;
        Scrollbox1.VertScrollBar.Position:=Scrollbox1.VertScrollBar.Position-dy;
        getcursorpos(p);
    end; // перетаскиваем объект
end;

```

5. Опишем переменную **Us_model** в интерфейсной части юнита

```

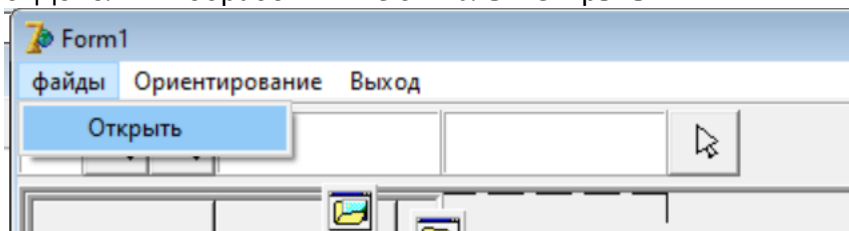
.....
    kxs : real=1.0;
    kys : real=1.0;
    us_model : byte=0;

```

implementation

.....

6. Дополним обработчик меню “Фалы >Открыть”



```

procedure TForm1.N2Click(Sender: TObject);
var rsh : string;
    i1,j1 : integer;
begin
    if openPictureDialog1.Execute then begin
        rsh:=ansiuppercase(extractfileext(openPictureDialog1.FileName));
    end;
end;

```

```

if rsh='.BMP' then begin
  bm.LoadFromFile(openPictureDialog1.FileName);
end;
if rsh='.JPG' then begin
  gp:=tjpegimage.Create;
  gp.LoadFromFile(openPictureDialog1.FileName);
  bm.assign(gp);
  gp.Destroy;
end;
if (rsh<>'.BMP')and(rsh<>'.JPG') then begin
  showmessage('Ôîðìàò íâ ïäååðæèääàòñý !');
  Exit;
end;
scrollbox1.Width:=bm.Width;;
scrollbox1.Height:=bm.Height;;
paintbox1.Width:=bm.Width;
paintbox1.Height:=bm.Height;
rect.Left:=0;
rect.Top:=0;
rect.Right:=paintbox1.Width;
rect.Bottom:=paintbox1.Height;
scrollbox1.Visible:=true;
paintbox1.Repaint;
panel2.Caption:='1';
toolbar1.Visible:=true;
scrollbox1.visible:=true;

```

{Si-} // ключ паскаля отключающий исключительную ситуацию при отсутствии файла **'kxs.dat'**

- передает код ошибки открытия файла в системную переменную **ioresult**

assignfile(t,'kxs.dat'); // файл с данными модели, если она построена

reset(t);

{Si+} // ключ паскаля включающий исключительную ситуацию на ошибки исполнения

программы

if ioresult<>0 then begin //проверка наличия файла. Если **ioresult<>0** - файл отсутствует

exit;

end

else begin // файл имеется

readln(t,i1); // читаем количество измерений

i:=i1+1;

stringgrid1.RowCount:=i+3; // устанавливаем число строк

for j:=0 to i1 do begin // читаем данные модели из файла

if length(x)<=j then begin

setlength(x,length(x)+1);

setlength(y,length(y)+1);

setlength(s,length(s)+1);

setlength(si,length(si)+1);

setlength(xi,length(xi)+1);

setlength(yi,length(yi)+1);

setlength(dx,length(dx)+1);

setlength(dy,length(dy)+1);

setlength(dxi,length(dxi)+1);

setlength(dyi,length(dyi)+1);

setlength(kx,length(kx)+1);

setlength(al,length(al)+1);

setlength(ali,length(ali)+1);

setlength(dal,length(dal)+1);

end;

```

readln(t,j1,x[j],y[j],dx[j],dy[j],s[j],xi[j],yi[j],dxi[j],dyi[j],si[j],kx[j],al[j],ali[j],dal[j]);
stringgrid1.Cells[0,j+1]:=inttostr(j1);
stringgrid1.Cells[1,j+1]:=floattostrf(x[j],ffixed,14,2);
stringgrid1.Cells[2,j+1]:=floattostrf(y[j],ffixed,14,2);
if dx[j]<>0 then stringgrid1.Cells[3,j+1]:=floattostrf(dx[j],ffixed,14,2);
if dy[j]<>0 then stringgrid1.Cells[4,j+1]:=floattostrf(dy[j],ffixed,14,2);
if s[j]<>0 then stringgrid1.Cells[5,j+1]:=floattostrf(s[j],ffixed,14,2);
stringgrid1.Cells[6,j+1]:=floattostrf(xi[j],ffixed,14,2);
stringgrid1.Cells[7,j+1]:=floattostrf(yi[j],ffixed,14,2);
if dxi[j]<>0 then stringgrid1.Cells[8,j+1]:=floattostrf(dxi[j],ffixed,14,2);
if dyi[j]<>0 then stringgrid1.Cells[9,j+1]:=floattostrf(dyi[j],ffixed,14,2);
if si[j]<>0 then stringgrid1.Cells[10,j+1]:=floattostrf(si[j],ffixed,14,2);
if kx[j]<>0 then stringgrid1.Cells[11,j+1]:=floattostrf(kx[j],ffixed,14,10);
if al[j]<>0 then begin
    rg(al[j]);
    stringgrid1.Cells[12,j+1]:=ss+ss1+ss2;
    rg(ali[j]);
    stringgrid1.Cells[13,j+1]:=ss+ss1+ss2;
    rg(dal[j]);
    stringgrid1.Cells[14,j+1]:=ss+ss1+ss2;
end;
end;
readln(t,kxs,als,x_izm,y_izm); // читаем коэф. преобразования и коорд. опорной точки 1
stringgrid1.Cells[11,i1+2]:=floattostrf(kxs,ffixed,14,10);
rg(als);
stringgrid1.Cells[14,i1+2]:=ss+ss1+ss2;
readln(t,xi[0],yi[0]); // читаем измеренные коорд. опорной точки 1

us_model:=1; // установили флаг наличия модели
end;
closefile(t);
end;
end;

```

PS Panel5 на которой расположен stringgrid должна лежать на форме! В вашем Object TreeView должно быть такое же расположение панели.

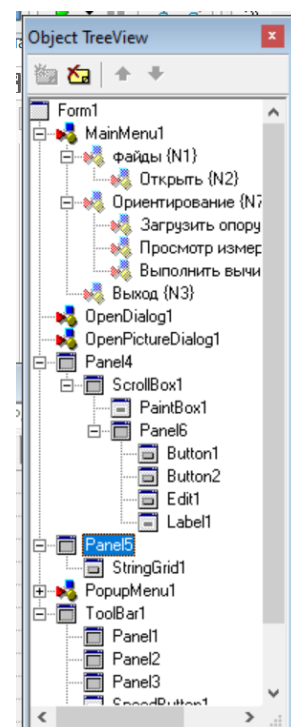
Все готово к ориентированию карты.

Откомпилируем программу, исправим ошибки.

Для ориентирования карты необходимо выполнить следующие действия:

1. Загрузить карту.
2. Загрузить опору.
3. Последовательно отнаблюдать - навести курсор на соответствующее координатам перекрестие километровой сетки - и щелкнуть левой кнопкой мышки.
4. В появившемся окне ввести номер точки.
5. После того, как отнаблюдаете все точки, войти в меню "Ориентирование" и выбрать пункт "Выполнить вычисления".

Выполнив данные пункты, при перемещении курсора по карте, на



соответствующих панелях ToolBar_а будут отображаться реальные прямоугольные координаты с некоторой погрешностью, связанной с алгоритмом ориентирования.

Во 2 части пособия будет рассмотрен алгоритм дробнолинейного трансформирования, решаемого по методу наименьших квадратов. Данный алгоритм пригоден не только для ориентирования карт, но и для одиночных снимков.