Система визуального объектно-ориентированного программирования Delphi

Главное меню — компонент MainMenu 

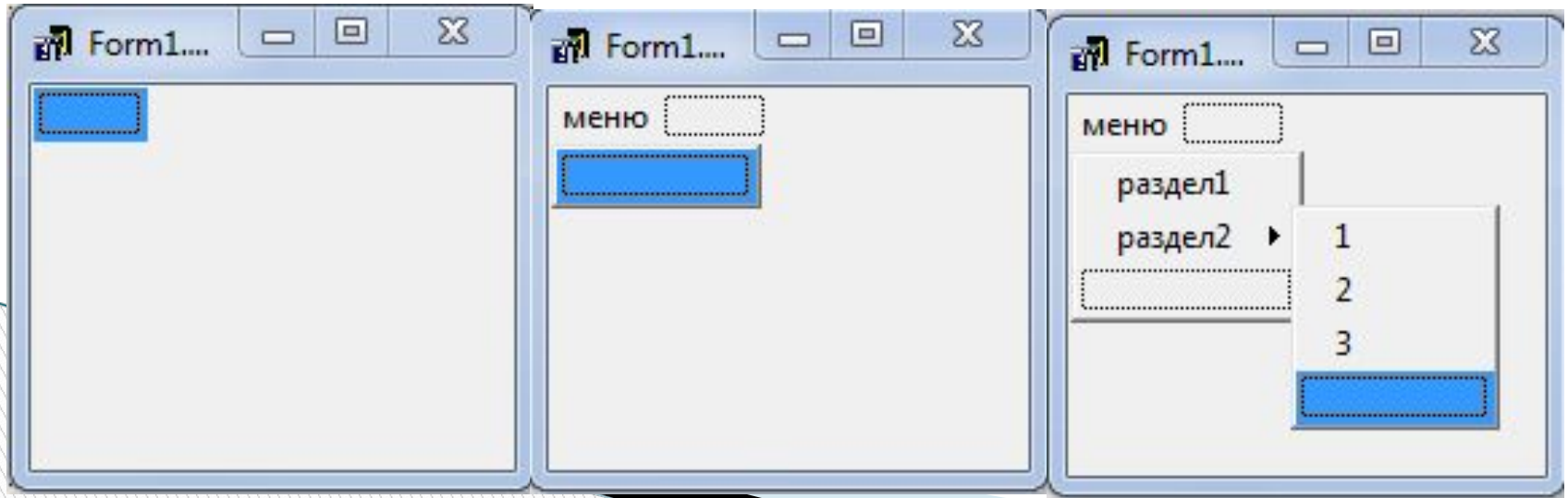
В Delphi имеется два компонента, представляющие меню: **MainMenu** — главное меню, и **PopupMenu** — всплывающее меню. Оба компонента расположены на странице Standard.

Это невизуальный компонент, т.е. место его размещения на форме в процессе проектирования не имеет никакого значения— при выполнении сам компонент не виден, а только меню, сгенерированное им.

Основное свойство компонента — **Items**. Его заполнение производится с помощью Конструктора Меню, вызываемого двойным щелчком на компоненте **MainMenu** или нажатием кнопки с многоточием рядом со свойством **Items** в окне Инспектора Объектов.

При работе в конструкторе меню новые разделы можно вводить, помещая курсор в рамку из точек, обозначающую место расположения нового раздела.

Другой путь ввода нового раздела — использование контекстного меню, всплывающего при щелчке правой кнопкой мыши. Если вы предварительно выделите какой-то раздел меню и выберите из контекстного меню команду Insert, то рамка нового раздела вставится перед ранее выделенным. Из контекстного меню вы можете также выполнить команду Create Submenu, позволяющую ввести подменю в выделенный раздел



Каждый раздел меню, т.е. каждый элемент свойства **Items**, является объектом типа **TMenuItem**, обладающим своими свойствами, методами, событиями.

Свойство **Caption** обозначает надпись раздела.

Если вы в качестве значения **Caption** очередного раздела введете символ минус «-», то вместо раздела в меню появится разделитель.

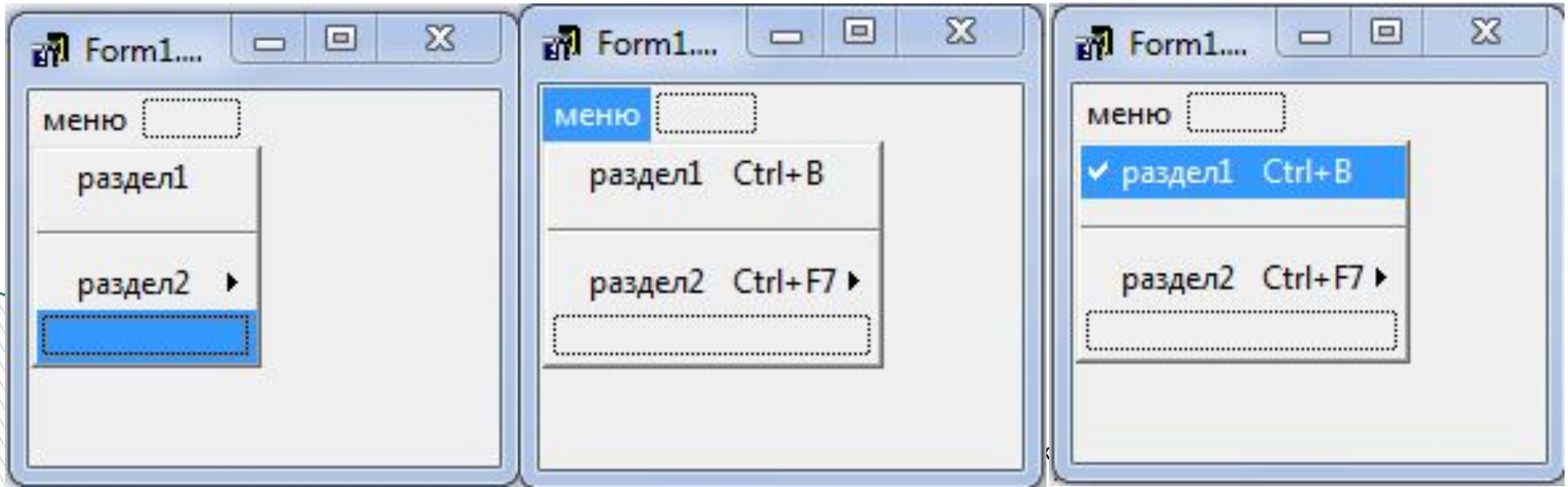
Свойство **Name** задает имя объекта, соответствующего разделу меню.

Свойство **Shortcut** определяет клавиши быстрого доступа к разделу меню, Чтобы их задать, надо открыть выпадающий список свойства **Shortcut** в окне Инспектора Объектов и выбрать из него нужную комбинацию клавиш. Эта комбинация появится в строке раздела меню.

Свойство **Break** используется в длинных меню, чтобы разбить список разделов на несколько столбцов.

Свойство **Checked**, установленное в **true**, указывает, что в разделе меню будет отображаться маркер флажка, показывающий, что данный раздел выбран. Но, сам по себе этот маркер не изменяется и в обработчик события **OnClick** такого раздела надо вставлять оператор типа

Mmenu.Checked := **not** Mmenu.Checked;



Еще одним свойством, позволяющим вводить маркеры в разделы меню, является **RadioItem**. Это свойство, установленное в **true**, определяет, что данный раздел должен работать в режиме радиокнопки совместно с другими разделами, имеющими то же значение свойства **GroupIndex**.

Для каждого раздела могут быть установлены свойства **Enabled** (доступен) и **Visible** (видимый).

Если установить **Enabled = false**, то раздел будет изображаться серой надписью и не будет реагировать на щелчок пользователя.

Если же задать **Visible = false**, то раздел вообще не будет виден, а остальные разделы сомкнутся, заняв место невидимого.

Свойства **Enabled** и **Visible** используются для того, чтобы изменять состав доступных пользователю разделов в зависимости от режима работы приложения.

Предусмотрена возможность ввода в разделы меню изображений. За это ответственны свойства разделов **Bitmap** и **ImageIndex**. Первое из них позволяет непосредственно ввести изображение в раздел, выбрав его из указанного вами файла. Второе позволяет указать индекс изображения, хранящегося во внешнем компоненте **ImageList**.

Основное событие раздела — **OnClick**, возникающее при щелчке пользователя на разделе или при нажатии «горячих» клавиш быстрого доступа.

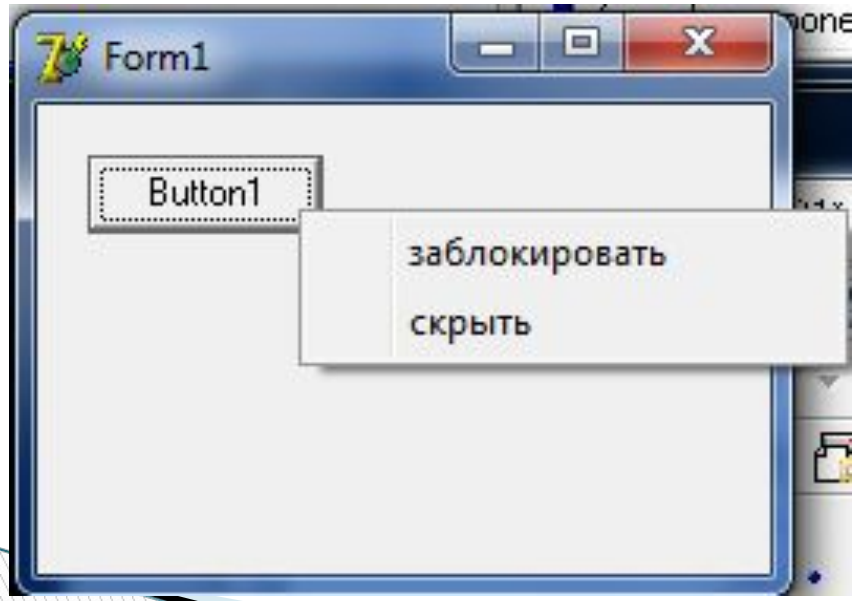
Контекстное всплывающее меню — компонент PopupMenu

Контекстное меню привязано к конкретным компонентам. Оно всплывает, если во время, когда данный компонент в фокусе, пользователь щелкнет правой кнопкой мыши. Обычно в контекстное меню включают те команды главного меню, которые в первую очередь могут потребоваться при работе с данным компонентом.

Оконные компоненты: панели, окна редактирования, а также метки и др. имеют свойство **PopupMenu**, которое по умолчанию пусто, но куда можно поместить имя того компонента **PopupMenu**, с которым будет связан данный компонент.

Формирование контекстного всплывающего меню производится с помощью Конструктора Меню, вызываемого двойным щелчком на **PopupMenu**, точно так же, как это делалось для главного меню.

В остальном работа с **PopupMenu** не отличается от работы с **MainMenu**.

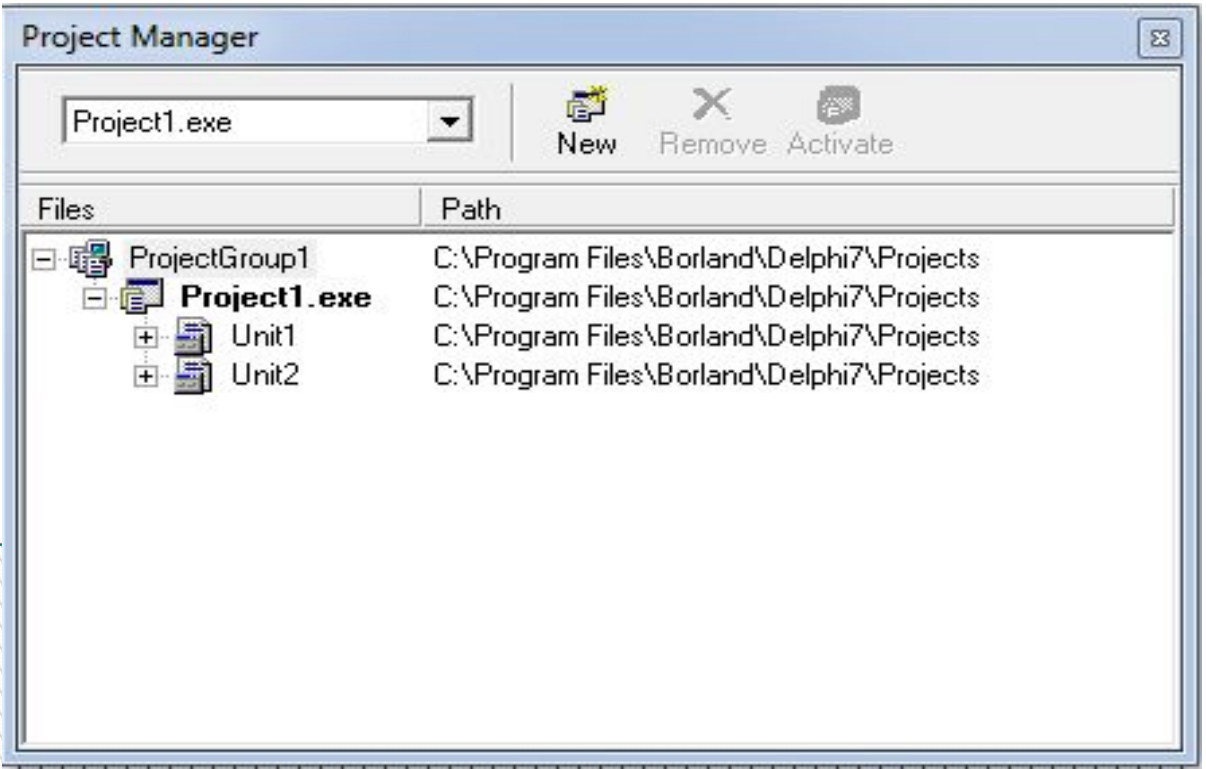


Создание дочерних окон

Для того чтобы создать новую форму, необходимо из меню **File** выбрать пункт **New**, а затем **Form**.

Delphi создаст новую чистую форму.

Для того, чтобы переключаться между формами можно войти в менеджер проектов (меню View-Project Manager) и дважды щелкнуть по нужной форме (*Unit1* или *Unit2*).

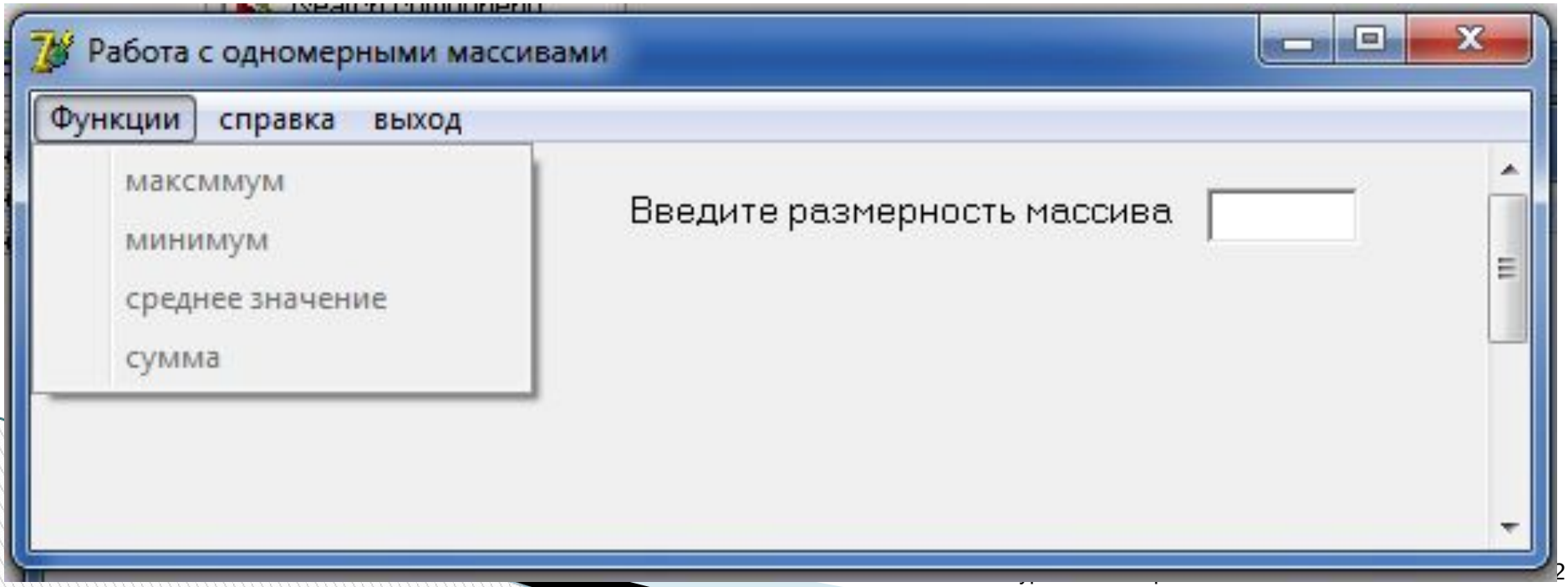


Дочерние окна могут быть модальные и не модальные.

Модальное - это значит, что управление полностью передаётся ему. Как только программа натыкается на код *Form2.ShowModal*, работа главной формы останавливается, и управление полностью передаётся дочерней форме. Пока модальное окно не закроется, главная форма работать не будет.

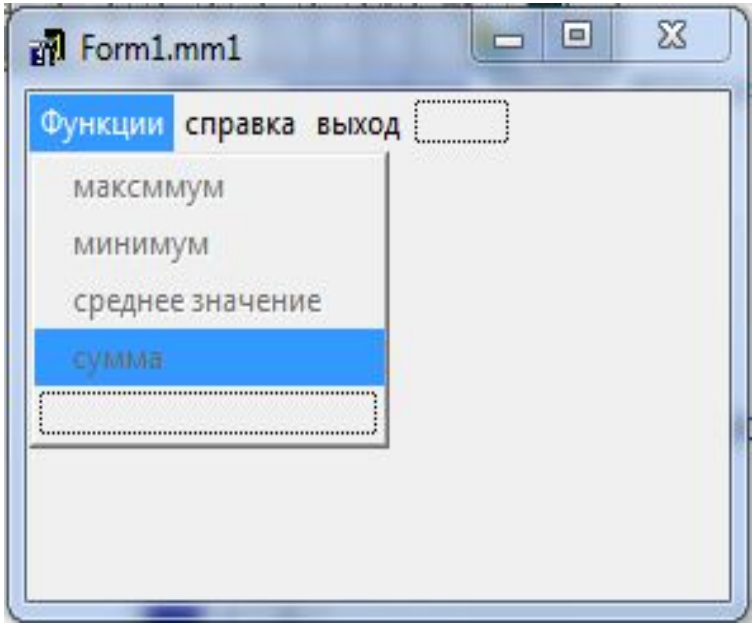
Для того чтобы создать не модальное окно, нужно вызвать метод ***Show***. В этом случае главная форма создаст дочернее, показав его на экране, и продолжит выполняться дальше. Это позволяет работать с обеими формами одновременно, переключаться между ними и код обеих форм будет выполнятся как бы параллельно.

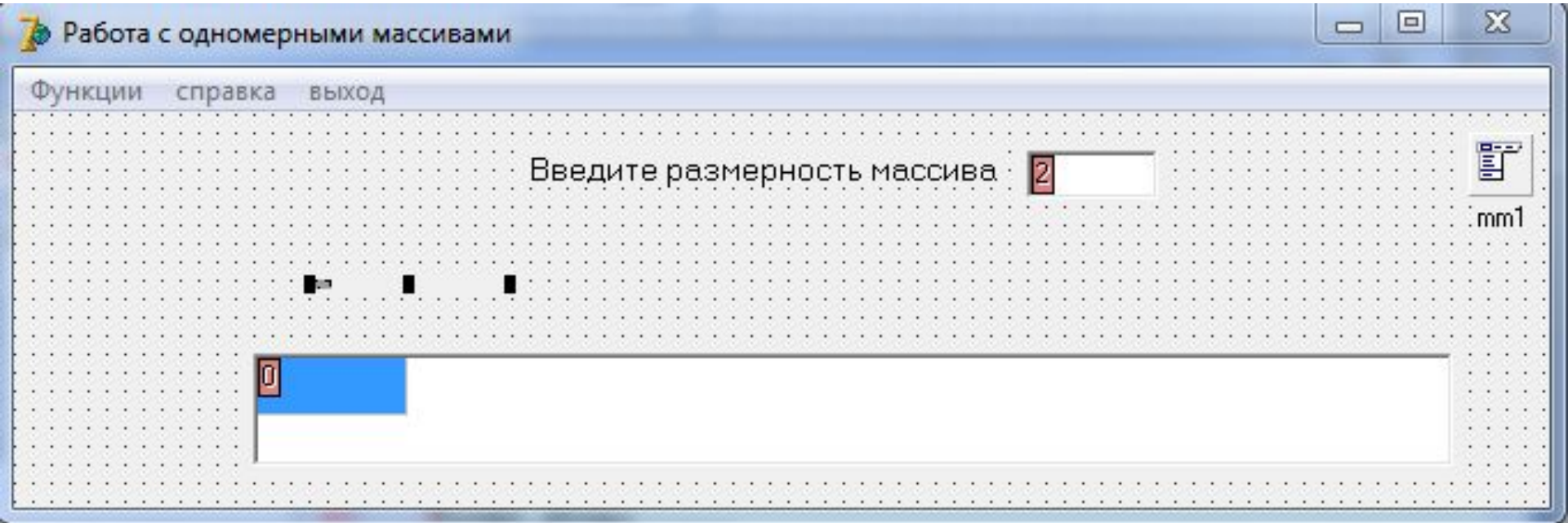
Программа вычисляет характеристики одномерного массива: максимум, минимум, сумму и среднее значение. Размер массива и элементы вводит пользователь. Выбор функции через меню. Есть справка о программе и отдельный пункт меню для выхода из программы. При вызове которого запрашивается подтверждение выхода.



Создаем новый проект и бросаем на форму компонент MainMenu. Дважды щелкнув по войдем в редактор меню и создадим само меню.

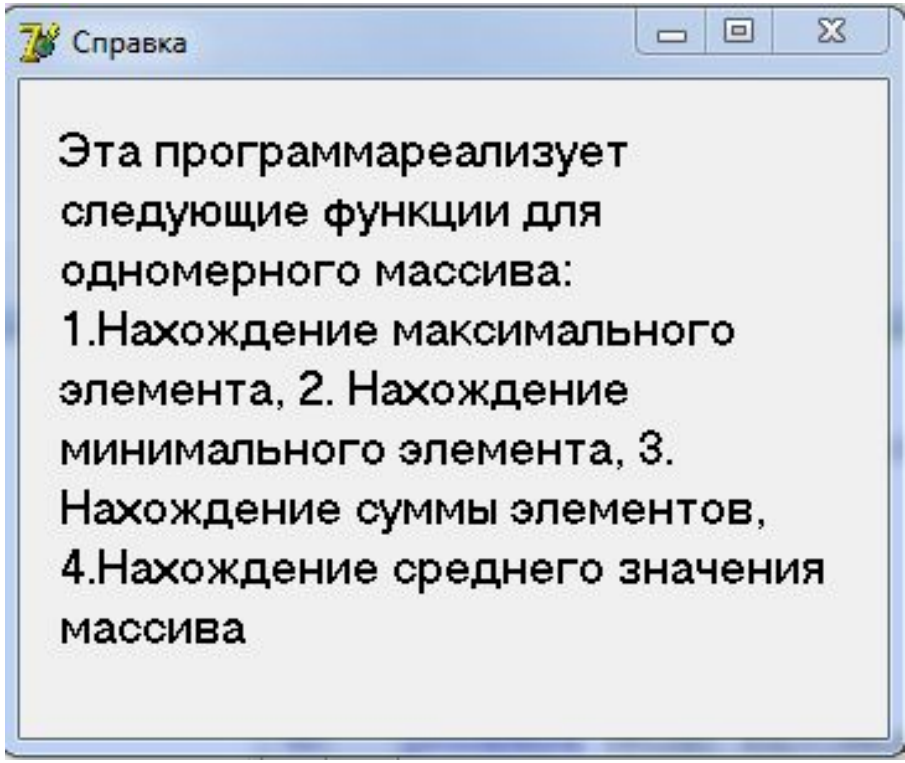
Затем бросаем на форму Edit для ввода размерности массива и метку Label с соответствующим комментарием, StringGrid и StaticText для вывода результата.





Создаем новую форму (она будет вызываться при нажатии на пункт меню «Справка» , переименуем ее, и бросим на нее StaticText с пояснениями.

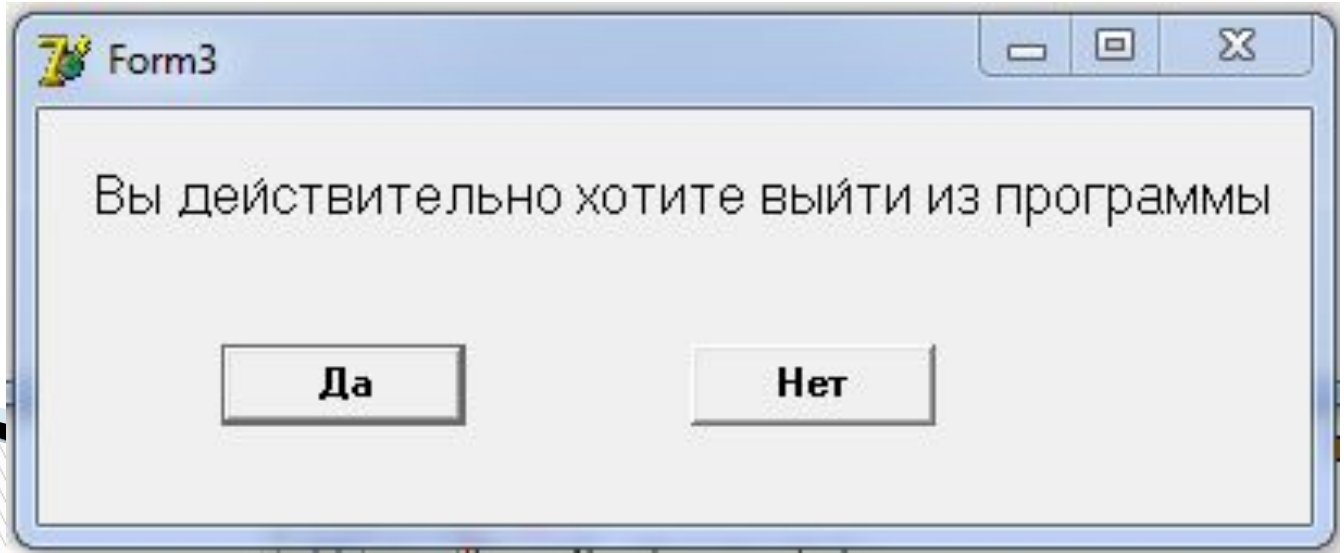
Подключим ее к нашему пункту меню справка. Для этого щелкнем по этому пункту и в появившейся процедуре обработки события напишем




```
procedure TForm1.N6Click(Sender: TObject);
  begin
    form2.show;
  end;
```

Если запустить проект на выполнение, то появится сообщение об ошибке, смысл которого в том, что из главного модуля идет ссылка на форму *Form2*, которая не объявлена в модуле *Unit1*. Будет предложено подключить этот модуль. Если нажать «Yes», то Delphi сам сделает все действия для подключения, в противном случае нужно прописывать это вручную.

Создадим еще одну форму, которая будет вызываться при выборе пункта меню «Выход». Бросим на нее две кнопки («Да» и «Нет») и попросим подтверждение выхода.



Напишем обработчик пункта меню «Выход»

```
procedure TForm1.N7Click(Sender: TObject);
begin
form3.showmodal;
end;
```

И обработчики нажатия кнопок на самой третьей форме

```
procedure TForm3.Button1Click(Sender: TObject);
begin
 form1.close;
end;
procedure TForm3.Button2Click(Sender: TObject);
begin
close;
end;
```

Обработаем корректный ввод информации в Edit и задание размерности для StringGrid.

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var
Key: Char);
begin
case key of
'1'..'9',#8:;
'0':if Length(edit1.Text)=0 then key:=#0;
#13: begin
strngrd1.Visible:=True;
strngrd1.ColCount:=StrToInt(Edit1.Text)
end
else key:=#0;
end;
end;
```

Обработчики всех пунктов меню «Функции»

```
procedure TForm1.N2Click(Sender: TObject);
var max,i:Integer;
begin
  max:=StrToInt(strngrd1.Cells[0,0]);
  for i:=1 to strngrd1.ColCount-1 do
    if max<StrToInt(strngrd1.Cells[i,0]) then
        max:=StrToInt(strngrd1.Cells[i,0]);
 txt1.Caption:='max='+inttostr(max);
end;
```

```
procedure TForm1.N3Click(Sender: TObject);
var min,i:Integer;
begin
  min:=StrToInt(strngrd1.Cells[0,0]);
  for i:=1 to strngrd1.ColCount-1 do
    if min>StrToInt(strngrd1.Cells[i,0]) then
        min:=StrToInt(strngrd1.Cells[i,0]);
 txt1.Caption:='min='+inttostr(min);
end;
procedure TForm1.N5Click(Sender: TObject);
var sum,i:Integer;
begin
  sum:=0;
  for i:=0 to strngrd1.ColCount-1 do
    inc(sum,StrToInt(strngrd1.Cells[i,0]));
 txt1.Caption:='сумма'+inttostr(sum);
end;
```

```
procedure TForm1.N4Click(Sender: TObject);
var sum,i:Integer; sr:Real;
begin
  sum:=0;
  for i:=0 to strngrd1.ColCount-1 do
    inc(sum,StrToInt(strngrd1.Cells[i,0]));
    sr:=sum/strngrd1.ColCount;
 txt1.Caption:='среднее'+floattostr(sr);
end;
```

Изначально установим все функции недоступными, и откроем доступ к ним, только если пользователь ввел количество элементов в массиве.

```
procedure TForm1.mfunctionClick(Sender: TObject);
begin
if Edit1.Text='' then
begin
N2.Enabled:=false ;
N3.Enabled:=false;
N4.Enabled:=false;
N5.Enabled:=false
end
else
begin
N2.Enabled:=true ;
N3.Enabled:=true;
N4.Enabled:=True;
N5.Enabled:=true
  end;
end;
```

Компоненты страницы Win32, их использование и свойства

Многостраничные панели – PageControl

Многостраничные панели позволяют экономить пространство окна приложения, размещая на одном и том же месте страницы разного содержания.

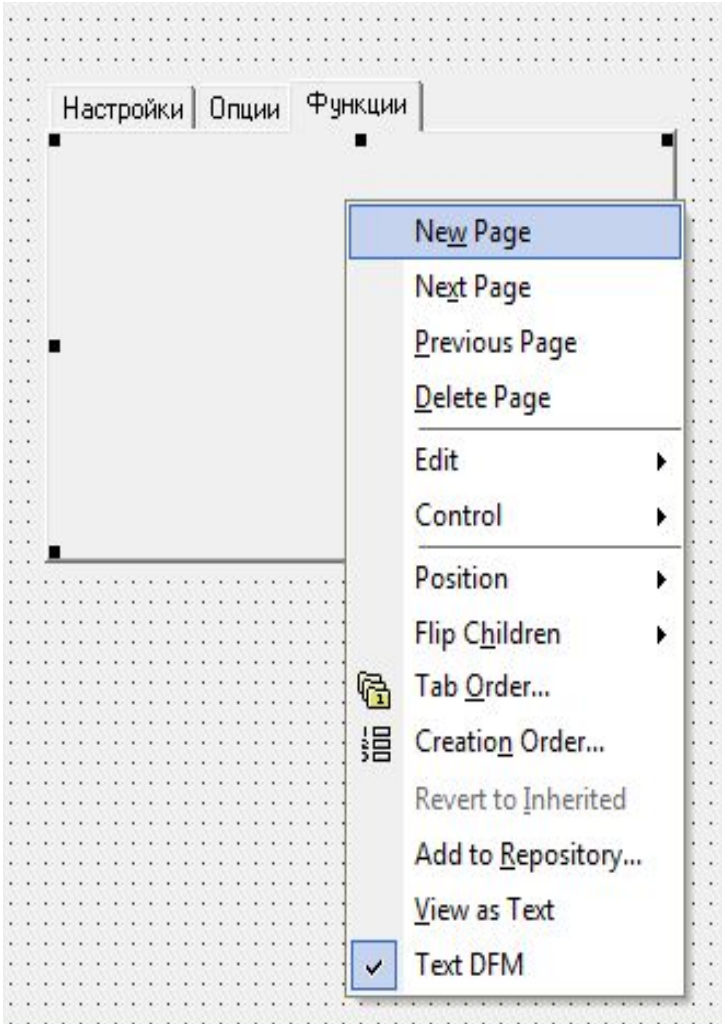
Чтобы задавать и редактировать страницы **PageControl**, надо щелкнуть на нем правой кнопкой мыши.

Во всплывшем меню команды:

New Page

Next Page

Previous Page

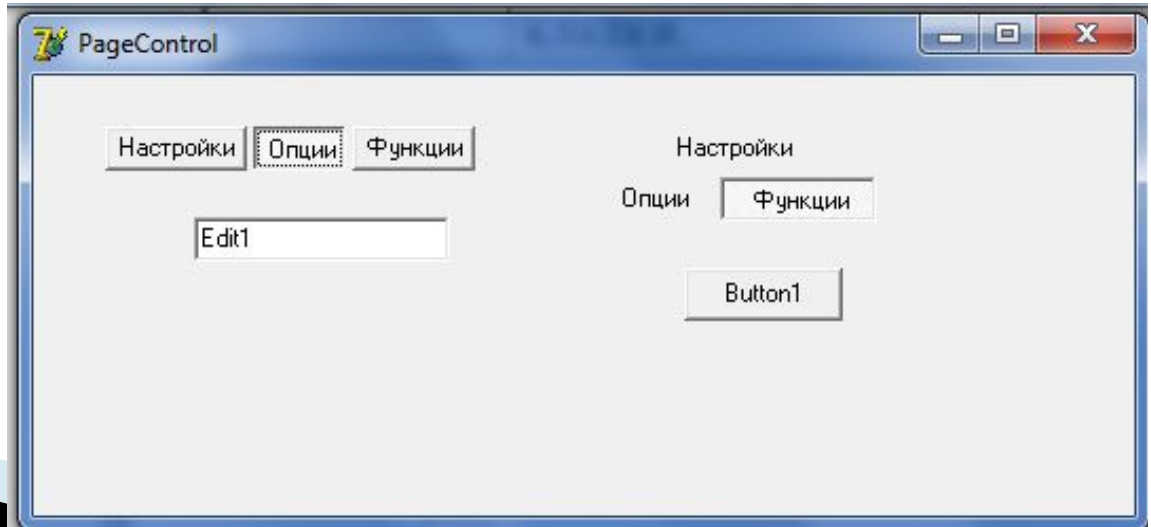


Основные свойства

Style - стиль отображения компонента

tsTabs — закладки, **tsButtons** — кнопки, **tsFlatButtons** — плоские кнопки

- ***ActivePage*** - имя активной страницы
- ***Pages[Index: Integer]*** – доступ к странице по индексу (начиная с 0). Свойство только для чтения.
- ***PageCount*** - количество страниц. Свойство только для чтения.
- ***MultiLine*** – размещение закладок в один или несколько рядов (false или true)



Основные события

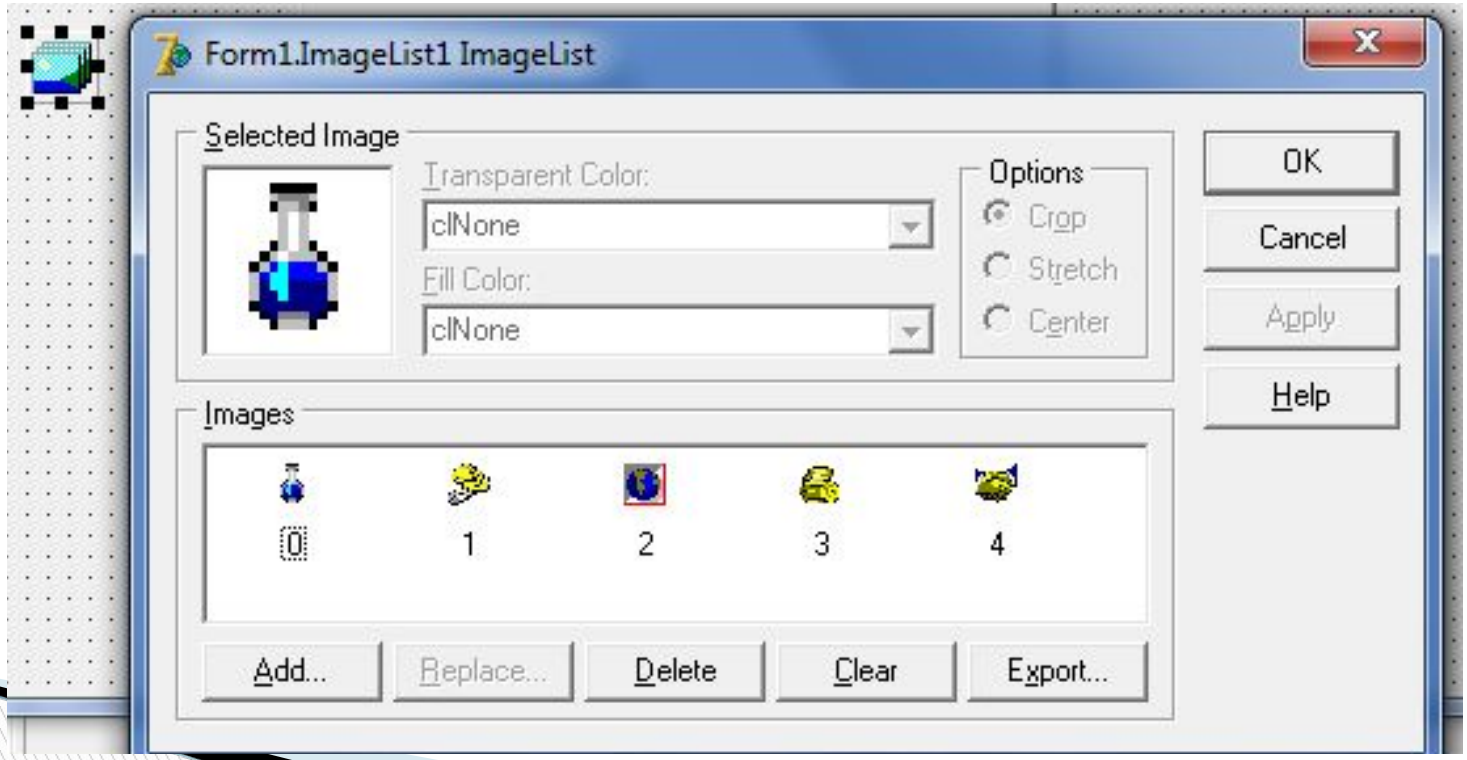
OnChange - происходит сразу после переключения.

OnChanging - происходит непосредственно перед переключением на другую страницу.
При этом в обработчик события передается по ссылке параметр **AllowChange** — разрешение переключения. Если в обработчике задать **AllowChange = false**, то переключение не произойдет. Событие **OnChange** присходит сразу после переключения.

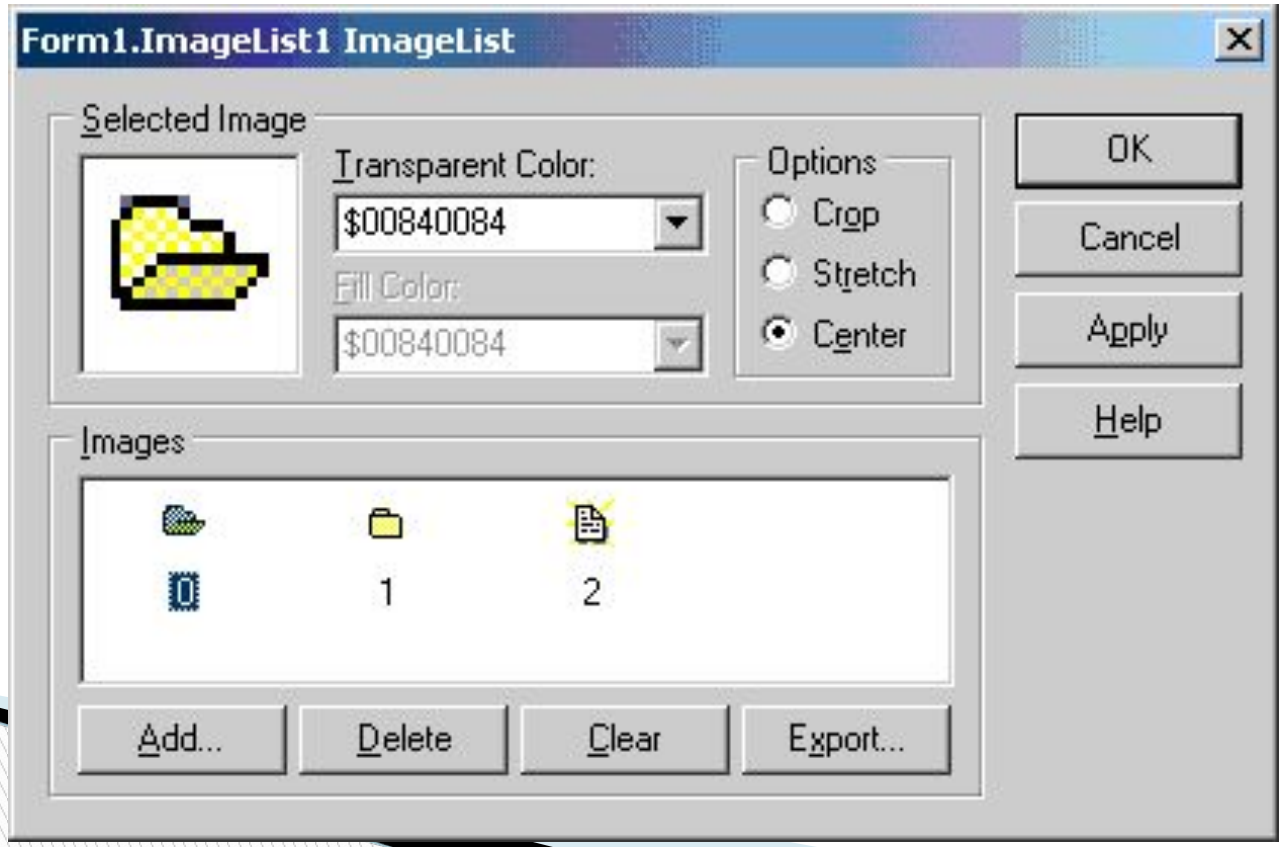
```
procedure TForm1.PageControl1Changing(Sender: TObject;
  var AllowChange: Boolean);
begin
   AllowChange:=false;
end;
```

Список изображений — ImageList

Компонент ***ImageList*** представляет собой набор изображений одинаковых размеров, на которые можно ссылаться по индексам.
Этот компонент позволяет организовать эффективное и экономное управления множеством пиктограмм



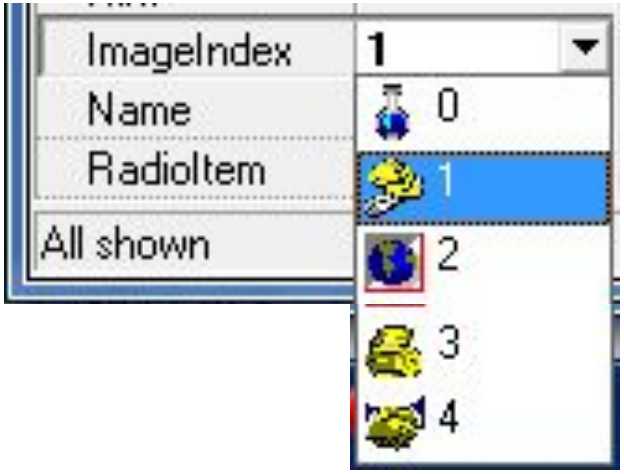
Изображения в компонент **TImageList** могут быть загружены в процессе проектирования с помощью редактора списков изображений. Окно редактора вызывается двойным щелчком на компоненте **TImageList** или щелчком правой кнопки мыши и выбором команды контекстного меню ImageList Editor.

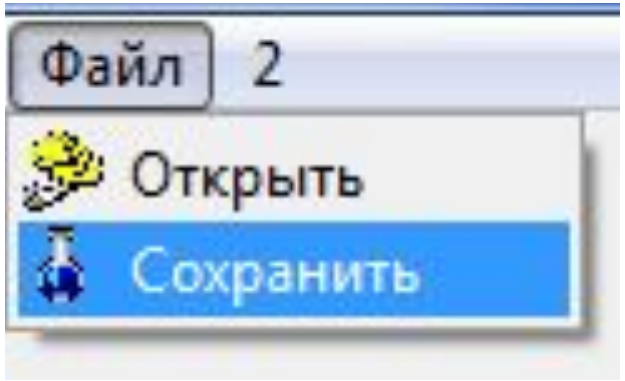


ImageList и другие компоненты

В компоненте Page Control рядом с названиями страниц могут располагаться изображения

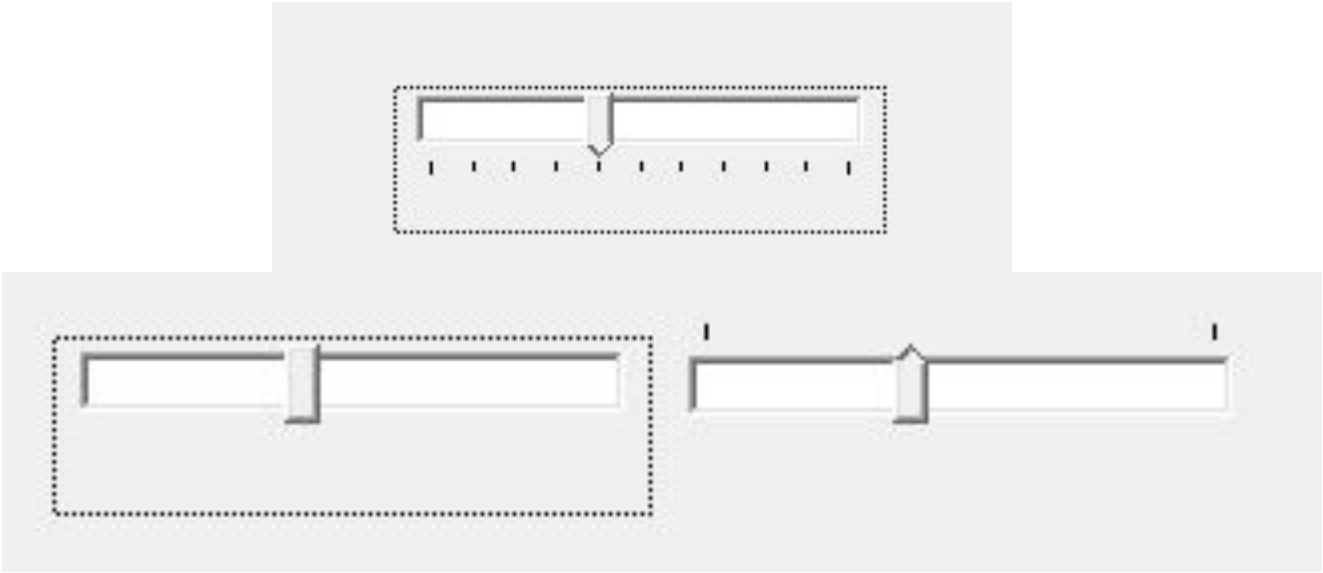
В компоненте MainMenu рядом с названиями пунктов могут располагаться изображени





Ползунки - TrackBar

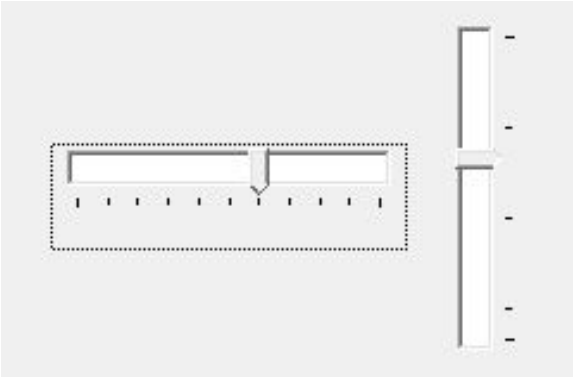
Элемент управления в виде ползунка, который пользователь может перемещать с помощью клавиатуры и мыши. Можно управлять процессами (громкостью звука, размером изображений и т.д.)



Основные свойства

Position – текущая позиция ползунка.
При перемещении пользователем ползунка можно прочитать значение **Position**, характеризующее позицию, в которую пользователь переместил ползунок. Для возможности такого чтения служит событие **OnChange**.

Frequency – частота отображения рисок.



Orientation – вид ползунка.
trHorizontal — горизонтальная,
trVertical — вертикальная
Max и ***Min*** – максимальное
и минимальное значение ползунка.
TickMarks – расположение шкалы относительно рисок.
tmBottomRight — снизу или справа, **tmTopLeft** — сверху или слева, **tmBoth** — с обеих сторон.

Индикация состояния процесса ProgressBar

Используется для индикации процессов, занимающих заметное время, например, копирования больших файлов, настройку приложения, установку приложения на компьютере и т.п.

Свойства схожи со свойствами TrackBar.

Smooth – непрерывное или дискретное отображения процесса.

Step - шаг приращения позиции.

Кнопка-счетчик UpDown

Компонент **UpDown** превращает окно редактирования **Edit** в компонент, в котором пользователь может выбирать целое число, изменяя его кнопками со стрелками. Если к тому же установить в **true** свойство окна **ReadOnly**, то пользователь просто не сможет ввести в окно какой-либо свой текст и вынужден будет ограничиться выбором числа.

Основные свойства:

Основное свойство компонента **UpDown** — **Associate**, связывающее кнопки со стрелками с одним из оконных компонентов, обычно с **Edit**.

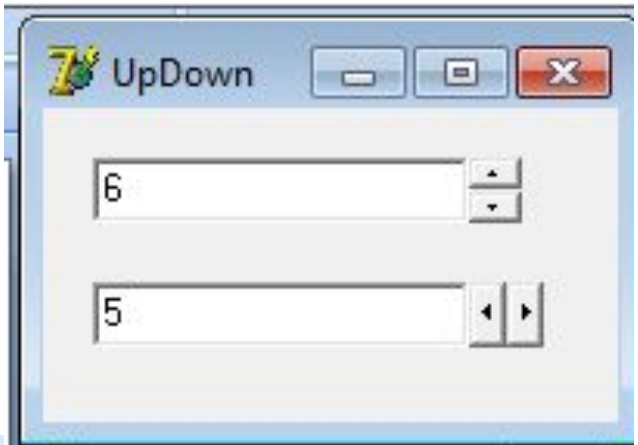
Перенесите на форму окно редактирования **Edit1**, расположив его там, где это требуется, а **UpDown** — в любом месте формы. Далее в выпадающем списке свойства **Associate** компонента **UpDown** выберите **Edit1**. Компонент **UpDown** немедленно переместится к **Edit** и сольется с ним.

Свойство **AlignButton** компонента **UpDown**, которое может принимать значения **udLeft** или **udRight**, определяет, слева или справа от окна будут размещаться кнопки.

Свойство **Orientation**, которое может принимать значения **udHorizontal** или **udVertical**, определяет, расположатся ли кнопки по вертикали или по горизонтали.

Свойство **ArrowKeys** определяет, будут ли управлять компонентом клавиши клавиатуры со стрелками.

Свойства **Min** и **Max** компонента **UpDown** задают соответственно минимальное и максимальное значения чисел.



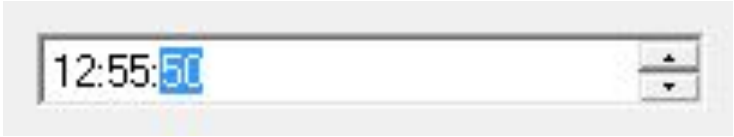
Свойство **Increment** задает приращение числа при каждом нажатии на кнопку.

Свойство **Position** определяет текущее значение числа. Это свойство можно читать, чтобы узнать, какое число задал пользователь. Его можно задать во время проектирования в диапазоне **Min** — **Max**. Тогда это будет значение числа по умолчанию, отображаемое в окне в начале выполнения приложения.

Свойство **Wrap** определяет, как ведет себя компонент при достижении максимального или минимального значений.

Ввод и отображение дат – DateTimePicker и MonthCalendar

Свойство **Kind** определяет режим работы компонента: **dtkDate** — ввод даты, **dtkTime** — ввод времени.



При вводе дат можно задать свойство **DateMode** равным **dmComboBox** — наличие выпадающего календаря, или равным **dmUpDown** — наличие кнопок увеличения и уменьшения

Формат представления дат определяется свойством **DateFormat**, которое может принимать значения **dfShort** — краткий формат (например, 01.12.99), или **dfLong** — полный формат (например, 1 декабря 1999 г.).

Значение даты по умолчанию можно задать в Инспекторе Объектов через свойство **Date**. Это же свойство читается для определения заданной пользователем даты.

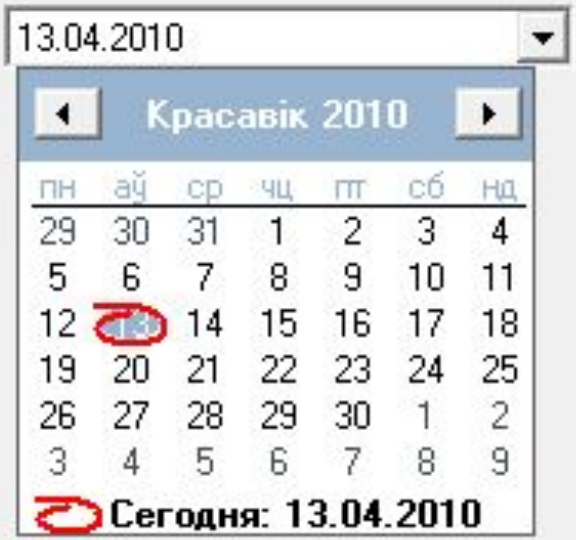
Для преобразования значения свойства **Date** в строку можно воспользоваться функцией **DateToStr**.

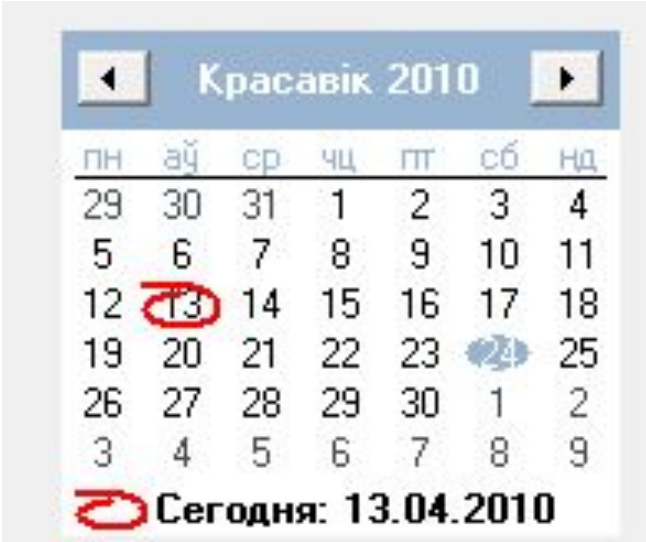
При вводе дат можно задать значения свойств **MaxDate** и **MinDate**, определяющих соответственно максимальную и минимальную дату, которую может задать пользователь.

В режиме ввода времени **dtkTime** введенное пользователем значение можно найти в свойстве **Time**. Преобразовать время в строку можно функцией **TimeToStr**.

Компонент MonthCalendar

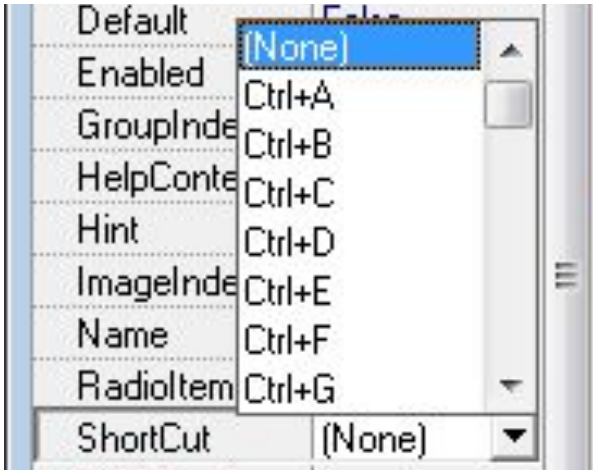
Компонент **MonthCalendar** похож на компонент **DateTimePicker**, работающий в режиме ввода дат, но в компоненте **MonthCalendar** предусмотрены некоторые дополнительные возможности: можно допустить множественный выбор дат в некотором диапазоне (свойство **MultiSelect**), можно указывать в календаре номера недель с начала года (свойство **WeekNumbers**) и т.п.

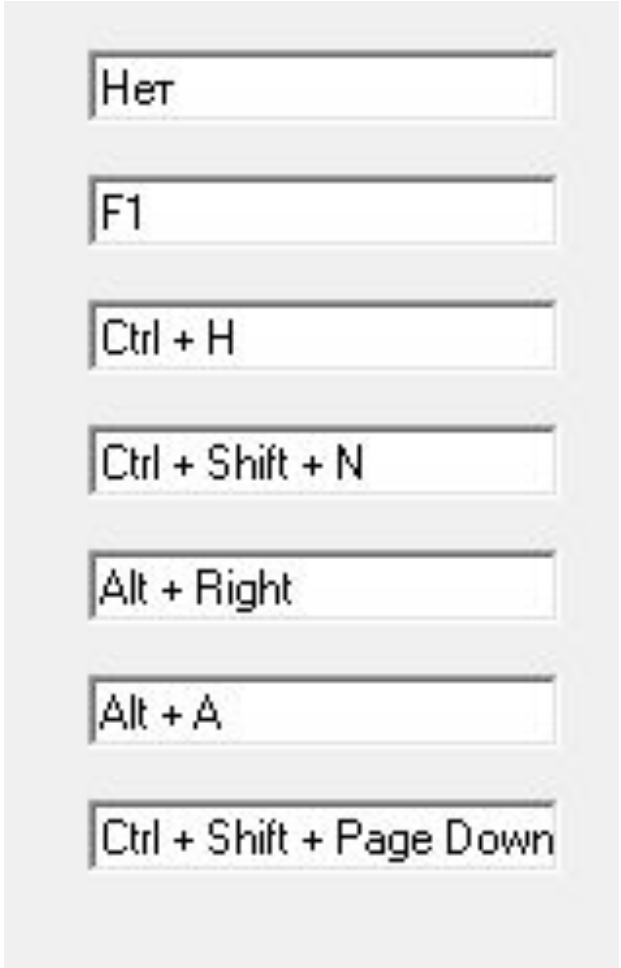




Горячие клавиши - HotKey

Обеспечивает возможность задания самим пользователем горячих клавиш, определяющих быстрый доступ к разделам меню. Позволяет задать такие сочетания клавиш, которые не предусмотрены в выпадающем списке ***ShortCut*** для пунктов меню.





Компонент **HotKey** внешне выглядит как обычное окно редактирования **Edit**. Но если в него входит пользователь, то оно переводит нажимаемые им клавиши в тип **TShortCut**, хранящий комбинацию горячих клавиш. Например, если пользователь нажимает клавиши Ctrl-ф, то в окне **HotKey** появится текст «Ctrl + ф».

Основное свойство компонента — **HotKey**, равное по умолчанию комбинации клавиш Alt-A. Это свойство можно прочесть и присвоить свойству **ShortCut** какого-то раздела меню.

Окно редактирования в формате RTF 

Компонент **RichEdit** работает с текстом в обогащенном формате RTF. При желании изменить атрибуты вновь вводимого фрагмента текста вы можете задать свойство **SelAttributes**.

Это свойство типа **TTextAttributes**, которое в свою очередь имеет подсвойства:

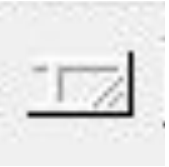
Color (цвет),

Name (имя шрифта),

Size (размер),

Style (стиль) и ряд других.

Полоса состояния StatusBar



Ряд панелей, отображающих полосу состояния в стиле Windows XP, обычно располагается внизу экрана.

Свойство **SimplePanel** определяет, включает ли полоса состояния одну или множество панелей.
Если **SimplePanel = true**, то вся полоса состояния представляет собой единственную панель, текст которой задается свойством **SimpleText**. Если же **SimplePanel = false**, то полоса состояния является набором панелей, задаваемых свойством **Panels**. В этом случае свойство **SizeGrip** определяет, может ли пользователь изменять размеры панелей в процессе выполнения приложения.

Каждая панель полосы состояния является объектом типа **TStatusPanels**. Свойства панелей вы можете задавать специальным редактором наборов.

Вызвать редактор можно тремя способами: из Инспектора Объектов кнопкой с многоточием около свойства **Panels**, двойным щелчком на компоненте **StatusBar** или из контекстного меню, выбрав команду **Panels Editor**.

В окне редактора вы можете перемещаться по панелям, добавлять новые или уничтожать существующие. При перемещении по панелям в окне Инспектора Объектов вы будете видеть их свойства.

Основное свойство каждой панели — **Text**, в который заносится отображаемый в панели текст.

Другое существенное свойство панели — **Width** (ширина).

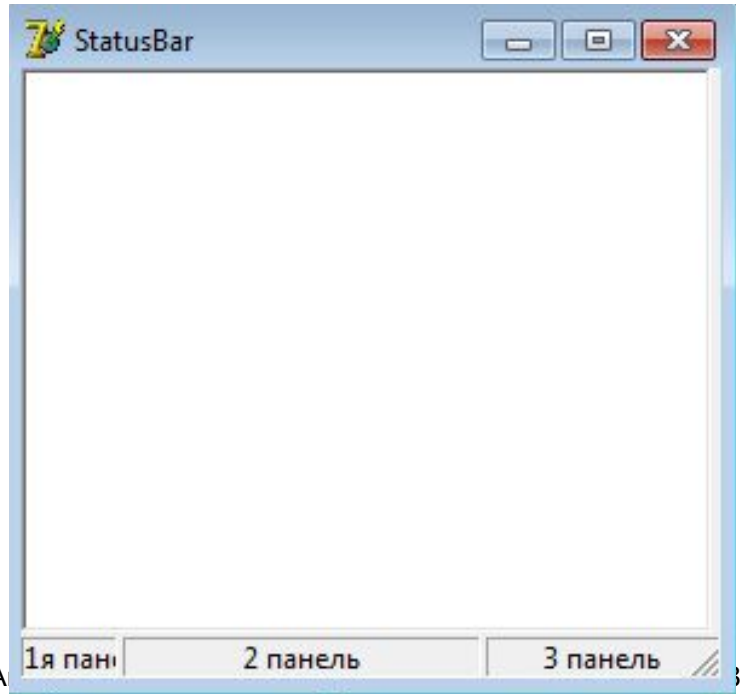
Программный доступ к текстам отдельных панелей можно осуществлять двумя способами: через индексированное свойство **Panels** или через его индексированное подсвойство **Items**.

Например, два следующих оператора дадут идентичный результат:

```
StatusBar1.Panels[0].Text := 'текст 1';
StatusBar1.Panels.Items[0].Text := 'текст 1';
```

Оба они напечатают текст «текст 1» в первой панели.

Количество панелей полосы состояния можно определить из подсвойства **Count** свойства **Panels**.



Всё!