**Телефонный справочник**

Для начала откройте **Lazarus** с новым проектом. Форму, как обычно, называем **fMain**, сохраняем проект в папку **25-01** под именем **telephones**, модулю главной формы присваиваем имя Main. Пойдем дальше. Скорее всего, вы будете делиться этой программой с друзьями, поэтому имеет смысл сразу же отключить от проекта вставку отладочной информации. Если вы помните, нужно выбрать команду "**Проект -> Параметры проекта**", в разделе "**Параметры компилятора**" перейти на подраздел "**Компоновка**" и убрать флажок "**Генерировать отладочную информацию для GDB**". Это позволит нашей программе сразу "похудеть" - от 15 мегабайт готового продукта, до менее чем 2-х мегабайт.

Теперь займемся формой. В свойстве Caption напишите "Телефонный справочник", в BorderStyle выберите bsDialog (нам не нужно, чтобы пользователь менял размеры окна), а в свойстве Position - poDesktopCenter. Чтобы ваше приложение в точности соответствовало моему, выберите ширину формы Width 700 пикселей, а высоту Height 400 пикселей.

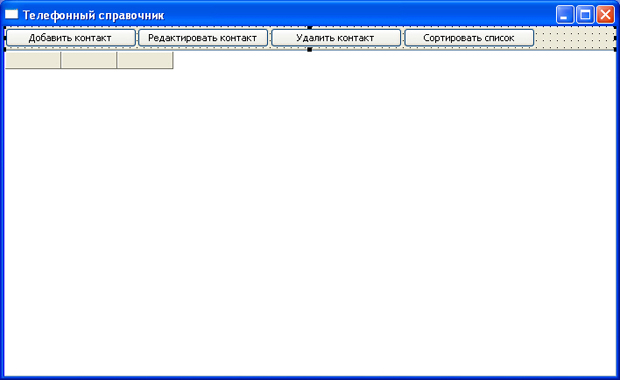
Теперь нам на форме потребуется простая панель инструментов. Установите обычную панель **TPanel**, очистите её свойство Caption, в свойстве Align выберите alTop, чтобы панель заняла весь верх формы, и в свойстве Height установите значение 27 - нам не нужна слишком высокая панель инструментов, узкая будет смотреться аккуратней.

На панель нам нужно установить четыре кнопки **TSpeedButton** с вкладки **Additional Палитры компонентов**. Кнопки **TSpeedButton** нам нужны, чтобы на них не переходил фокус ввода, в этом случае, всегда активной будет только сетка. Присвойте свойству Name кнопок соответственно, имена bAdd, bEdit, bDel и bSort. Первая "b" в именах означает, что это кнопка - button. В свойствах Caption этих кнопок напишите, соответственно, "Добавить контакт", "Редактировать контакт", "Удалить контакт" и "Сортировать список" (без кавычек, разумеется). Картинок на кнопки мы ставить не будем, обойдемся только пояснительным текстом. Свойству Top всех кнопок присвойте значение 2, чтобы они почти прижались к верхней границе панели, свойству Width присвойте 150, чтобы весь текст помещался на кнопках без проблем. Переместите их относительно друг друга так, чтобы от левого края панели и друг от друга их отделяло тоже примерно по 2 пикселя.

Теперь сетка. На свободное место формы, ниже панели, установите сетку TStringGrid. Поскольку нам часто придется обращаться к ней в процессе кодирования, укоротим имя в свойстве Name до SG. В свойстве Align сетки выберите alClient, чтобы сетка заняла всё оставшееся место. Теперь нужно изменить некоторые другие настройки. По умолчанию, в сетке установлено по пять колонок ColCount, и по пять строк RowCount. Колонок нам нужно только три: имя контакта, его телефон и примечание, какой это телефон (мобильный, домашний, служебный, соседский). Поэтому уменьшим количество колонок ColCount до 3. А строка нам нужна и вовсе одна, она будет служить заголовком колонок, и она будет фиксированной. Уменьшайте RowCount до 1.

Теперь о фиксированных строках и колонках. Фиксированная строка FixedRows нам нужна только одна, это значение по умолчанию. А фиксированные столбцы FixedCols нам и вовсе не нужны, тут поставьте ноль.

Остальные параметры сетки можно оставить без изменений. Если вы всё сделали правильно, у вас в результате должна получиться такая форма:



Главная форма проекта

Пойдем дальше. Прежде, чем приступать к самому кодированию, давайте сделаем еще одну форму - окно редактора контактов. Командой **Файл -> Создать форму** создайте новую форму. Присвойте ей имя **fEdit**, нажмите кнопку "**Сохранить все**" и дайте модулю формы имя Edit.

В свойстве Caption напишите "Редактор контакта". Почему не "контактов"? Каждый раз мы будем загружать в это окно по одному контакту, поэтому и использовано единственное число. В свойстве BorderStyle также выберите bsDialog, в свойстве Position - poMainFormCenter, чтобы окно появлялось по центру главной формы, где бы она ни находилась. Размеры формы: ширина width = 400, высота height = 225 пикселей.

Теперь разместим на форме компоненты. Установите метку TLabel, в Caption которой напишите "Укажите имя контакта:".

Ниже установите строку TEdit. Переименуйте её в eName, ширину сделайте 380. Очистите свойство Text.

Ниже еще одну метку, с текстом в Caption "Укажите телефон:".

Ниже установите еще один TEdit. Переименуйте её в eTelephone, ширину сделайте 210. Очистите свойство Text.

Ниже еще одну метку, с текстом в Caption "Выберите тип телефона:".

Ниже нам потребуется список выбора TComboBox. Мы же не хотим, чтобы пользователю каждый раз самому приходилось писать "мобильный" или "служебный"? Переименуйте TComboBox в CBNote, откройте редактор свойства Items и там наберите следующие строки:

Мобильный

Домашний

Служебный

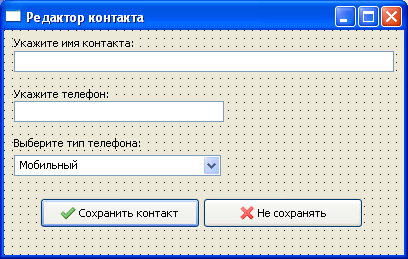
Соседский

Если сможете придумать другие типы, можете дописать их сюда.

В свойстве ItemIndex компонента установите 0, чтобы в списке появилась первая строка "**Мобильный**". Ширину сделайте 210.

Ещё ниже установите на одном уровне кнопки **TBitBtn**. Первую переименуйте в bSave, в свойстве Kind выберите bkOK, чтобы на кнопке появилась картинка - зеленая галочка, убедитесь, что свойство ModalResult кнопки перешло в mrOk. В свойстве Caption напишите "Сохранить контакт". Чтобы текст нормально умещался, увеличьте её ширину до 160 пикселей.

Вторую кнопку переименуйте в bCancel, в свойстве Kind выберите bkCancel, а в Caption напишите "Не сохранять". Свойство ModalResult кнопки должно перейти в mrCancel. В результате, у вас должна получиться вот такая форма:



Форма проекта fEdit

Чтобы более не возвращаться к этой форме, давайте сразу введем весь необходимый код формы, тем более что его совсем мало. Для кнопок нам вообще не нужно писать код. Когда мы изменили их свойства Kind, установив там bkOK и bkCancel, сразу изменилось и их свойство ModalResult, установившись, соответственно, в mrOK и mrCancel. И теперь нажатие на любую из этих кнопок приведет к закрытию формы, и к тому, что в ModalResult формы попадет значение ModalResult нажатой кнопки. То есть, мы сможем узнать, какой кнопкой пользователь закрыл окно.

Но нам нужно все же сделать некоторую подготовительную работу. Прежде, чем показывать форму, нам нужно будет перевести фокус ввода в первый TEdit, ведь от прежней работы с формой выделенным мог остаться и другой компонент - кнопка или список выбора, например. А это будет неудобно для пользователя, ему придется лишний раз кликать мышью, выделяя первую строку.

Выделите форму, щелкнув по свободному участку, в Инспекторе объектов перейдите на вкладку **События** и сгенерируйте событие формы OnShow - это событие возникает всякий раз перед показом формы - идеальное место для подготовительной работы. Код события совсем простой:

procedure TfEdit.FormShow(Sender: TObject);

begin

eName.SetFocus;

end;

И с редактором контактов это все! Вся остальная работа будет происходить в главной форме.

В редакторе форм перейдите на вкладку **Main** - наша главная форма. Прежде всего, в разделе uses главной формы, через запятую, добавьте модуль edit - это наш редактор контактов. Если мы не добавим его в раздел uses, то не сможем с ним работать из главной формы.

Далее, **перед** разделом глобальных переменных, тем, где объявлена переменная формы fMain, создадим еще один раздел type, в котором опишем запись:

type

Contacts = record

Name: string[100];

Telephon: string[20];

Note: string[20];

end; //record

В этой записи три поля. Первое поле Name предназначено для имени контакта. Вы можете заметить, что оно ограничено размером 100 символов. Как уже говорилось в [лекции № 24](http://intuit.valrkl.ru/24/), в записи можно указывать неограниченный тип string, но тогда создать типизированный файл этого типа не получится. Поэтому у нас все строки имеют фиксированный размер. В поле Name пользователь будет вписывать имена контактов, например:

Анатолий Васильевич, директор

Дима, друг

Леночка, секретарь

Может показаться, что 100 символов для имени контакта - это перебор. Однако вспомните, что для текста у нас используется кодировка UTF-8, а в ней каждый символ кириллицы занимает 2 байта. То есть, если писать имена по-русски, реально можно вписать только 50 символов! А фамилии и имена бывают очень длинными, но 50 символов должно хватить. Для телефона мы указали 20 символов. Это обычно, цифры, скобки и знаки тире:

111-11-11

3-(222)-111-11-11

На каждый из этих символов требуется только 1 байт, так что 20 символов хватит за глаза для любого телефона.

Ну и, наконец, поле Note, в котором будет одно из значений:

Мобильный

Домашний

Служебный

Соседский

Эти символы тоже занимают по 2 байта, самое длинное слово имеет 9 символов - 18 байт. А мы указали 20, так что должно хватить.

Далее, в разделе глобальных переменных, после описания самой формы, опишем глобальную переменную adres:

var

fMain: TfMain;

adres: string; //адрес, откуда запущена программа

Эта переменная понадобится нам позже, в нее мы запишем адрес, откуда была запущена программа, чтобы потом по этому адресу отыскать файл с контактами. Ранее мы говорили, что если не указывать адрес, то компьютер будет искать файл в текущей папке, то есть - в папке с программой. Однако так бывает не всегда. Если вы параллельно работаете с несколькими другими программами, и все они то и дело обращаются к файлам, то текущий адрес может измениться. Вот почему всегда лучше подстраховаться, и указать настоящий адрес - это гарантирует вашу программу от любых ошибок.

Далее, "научим" программу принимать новые данные и выводить их в сетку. Сгенерируйте событие OnClick для кнопки "**Добавить контакт**". Код события такой:

procedure TfMain.bAddClick(Sender: TObject);

begin

//очищаем поля, если там что-то есть:

fEdit.eName.Text:= '';

fEdit.eTelephone.Text:= '';

//устанавливаем ModalResult редактора в mrNone:

fEdit.ModalResult:= mrNone;

//теперь выводим форму:

fEdit.ShowModal;

//если пользователь ничего не ввел - выходим:

if (fEdit.eName.Text= '') or (fEdit.eTelephone.Text= '') then exit;

//если пользователь не нажал "Сохранить" - выходим:

if fEdit.ModalResult <> mrOk then exit;

//иначе добавляем в сетку строку, и заполняем её:

SG.RowCount:= SG.RowCount + 1;

SG.Cells[0, SG.RowCount-1]:= fEdit.eName.Text;

SG.Cells[1, SG.RowCount-1]:= fEdit.eTelephone.Text;

SG.Cells[2, SG.RowCount-1]:= fEdit.CBNote.Text;

end;

Давайте разбираться с кодом. Прежде всего, в редакторе контактов fEdit мы очистили текст из строк с именем и номером телефона. Ведь если пользователь уже добавлял или редактировал контакты, в этих строках останутся записи с прошлого сеанса. Затем свойству ModalResult формы мы присваиваем значение mrNone - нет результата. Делаем это затем, чтобы потом точно знать, каким образом пользователь закрыл окно редактора контактов. Если кнопкой "**Сохранить контакт**", то в этом случае ModalResult формы будет mrOk, если кнопкой "**Не сохранять**", ModalResult формы будет mrCancel. Если же он просто закрыл окно крестиком в правом верхнем углу окна или клавишами **<Alt+F4>**, то в этом случае ModalResult формы так и останется mrNone.

Далее, мы выводим окно редактора контактов, как модальное.

К следующей строке кода программа доберется, когда пользователь введет (или не введет) новый контакт и закроет окно редактора. Прежде, чем что-то добавлять в сетку, мы делаем проверку - а ввел ли пользователь имя нового контакта, и телефон? Ведь если он ничего не ввел, то и сохранять нечего. И если он ввел один только телефон, без имени, или имя без телефона - какой смысл сохранять такой контакт? Поэтому, если нет имени или телефона, мы просто выходим из процедуры:

if (fEdit.eName.Text= '') or (fEdit.eTelephone.Text= '') then exit;

Однако может случиться, что пользователь ввел и имя, и телефон, но потом передумал сохранять контакт и нажал кнопку "**Не сохранять**", или просто закрыл окно редактора. Поэтому нам нужно сделать еще одну проверку:

if fEdit.ModalResult <> mrOk then exit;

Здесь мы смотрим значение свойства ModalResult формы. Оно сможет стать mrOK только в одном случае - если пользователь нажал кнопку "**Сохранить контакт**". И если это не так, то выходим из процедуры, ничего не сохраняя.

Если мы не вышли из процедуры в предыдущих двух проверках, то это означает, что

1. нам есть, что сохранять
2. пользователь нажал "**Сохранить контакт**"

И если это так, значит, добавляем новую строку в сетку SG и заполняем три её колонки значениями из редактора контактов. Дополнительных проверок на существование текста примечания мы не делали, так как там в любом случае будет указан один из типов телефонов:

//иначе добавляем в сетку строку, и заполняем её:

SG.RowCount:= SG.RowCount + 1;

SG.Cells[0, SG.RowCount-1]:= fEdit.eName.Text;

SG.Cells[1, SG.RowCount-1]:= fEdit.eTelephone.Text;

SG.Cells[2, SG.RowCount-1]:= fEdit.CBNote.Text;

Сохраните проект и запустите его на выполнение. Если вы всё сделали верно, программа позволит вам добавлять новые контакты. Не обращайте внимания на недоработанный вид сетки - узкие столбцы без заголовков - это мы исправим чуть позже.

Но этого мало, хотелось бы ещё и сохранять эти контакты в файл. Для этого нам лучше всего подойдет событие формы OnClose, которое будет выполняться только один раз за сеанс работы программы - перед её закрытием. Форма вся закрыта компонентами, но по прошлым лекциям вы должны помнить, что форму можно выделить в верхнем окошке **Инспектора объектов**, где компоненты указаны в виде дерева, в самой вершине которого форма **fMain**. Еще можно выделить компонент верхнего над формой уровня - панель или сетку, и нажать **<Esc>**. При этом выделение перейдет на более низкий уровень - на форму.

Итак, выделите форму, в **Инспекторе объектов** перейдите на вкладку **События** и сгенерируйте событие OnClose. Код события такой:

procedure TfMain.FormClose(Sender: TObject; var CloseAction: TCloseAction);

var

MyCont: Contacts; //для очередной записи

f: file of Contacts; //файл данных

i: integer; //счетчик цикла

begin

//если строки данных пусты, просто выходим:

if SG.RowCount = 1 then exit;

//иначе открываем файл для записи:

try

AssignFile(f, adres + 'telephones.dat');

Rewrite(f);

//теперь цикл - от первой до последней записи сетки:

for i:= 1 to SG.RowCount-1 do begin

//получаем данные текущей записи:

MyCont.Name:= SG.Cells[0, i];

MyCont.Telephon:= SG.Cells[1, i];

MyCont.Note:= SG.Cells[2, i];

//записываем их:

Write(f, MyCont);

end;

finally

CloseFile(f);

end;

end;

Как видите, мы объявили переменную типа нашей записи Contacts, и типизированный файл с таким же типом. Еще мы объявили целочисленную переменную, которая понадобится нам для организации цикла.

Вначале мы проверяем - есть ли в сетке данные? Если SG.RowCount = 1, то это означает, что в сетке есть только одна строка - фиксированная, та, где у нас находятся заголовки колонок (а меньше 1 строки и быть не может, пользователь не сможет удалить фиксированную строку). Другими словами, данных нет. В этом случае мы просто выходим из процедуры.

Если программа пошла дальше, значит, данные в сетке есть, правда? Тогда мы начинаем работу по сохранению данных в файл. Эту работу мы помещаем в блок try-finally-end, работу с файлами (в том числе типизированными) мы с вами неоднократно рассматривали, так что подробно останавливаться тут не буду. Но пару замечаний сделать все же нужно. Во-первых, при ассоциации с файлом мы указали не только имя, но и адрес файла, который должен находиться в переменной adres:

AssignFile(f, adres + 'telephones.dat');

Однако эту переменную мы объявили, но ничего туда пока не внесли. Не обращайте на это внимания - мы сделаем это в следующем коде. Пока считайте, что адрес там есть, однако не спешите запускать программу на выполнение, точнее, не пытайтесь закрыть эту программу, если там есть данные.

Во-вторых, запись данных в файл мы организовали в виде цикла:

for i:= 1 to SG.RowCount-1 do begin

То есть, мы делаем эту работу от строки с индексом 1 (нулевой индекс у фиксированной строки, она нам для обработки данных не нужна), до строки с индексом SG.RowCount-1. Так, если в сетке у нас 10 строк, то индекс последней строки = 9, а RowCount вернет 10, потому мы отнимаем единицу.

Внутри цикла мы считываем в переменную типа запись данные из сетки, из текущей строки данных:

MyCont.Name:= SG.Cells[0, i];

Когда считали все значения, мы записываем эту переменную в типизированный файл. Поскольку типы переменной и файла совпадают, то запись будет успешной, а указатель сместится в конец файла. Затем цикл продолжится. При каждом новом шаге цикла значение i будет увеличиваться на единицу, и код будет ссылаться на новую следующую строку. Записи будут записываться в файл одна за другой, до самого конца. В результате у нас получится нечто вроде базы данных.

Но чтобы от нашей базы был толк, программа при загрузке должна эти данные считывать обратно в сетку. Удобнее всего, как уже говорилось ранее, подготовительную работу помещать в событие формы OnCreate. Сгенерируйте это событие, код будет таким:

procedure TfMain.FormCreate(Sender: TObject);

var

MyCont: Contacts; //для очередной записи

f: file of Contacts; //файл данных

i: integer; //счетчик цикла

begin

//сначала получим адрес программы:

adres:= ExtractFilePath(ParamStr(0));

//настроим сетку:

SG.Cells[0, 0]:= 'Имя';

SG.Cells[1, 0]:= 'Телефон';

SG.Cells[2, 0]:= 'Примечание';

SG.ColWidths[0]:= 365;

SG.ColWidths[1]:= 150;

SG.ColWidths[2]:= 150;

//если файла данных нет, просто выходим:

if not FileExists(adres + 'telephones.dat') then exit;

//иначе файл есть, открываем его для чтения и

//считываем данные в сетку:

try

AssignFile(f, adres + 'telephones.dat');

Reset(f);

//теперь цикл - от первой до последней записи сетки:

while not Eof(f) do begin

//считываем новую запись:

Read(f, MyCont);

//добавляем в сетку новую строку, и заполняем её:

SG.RowCount:= SG.RowCount + 1;

SG.Cells[0, SG.RowCount-1]:= MyCont.Name;

SG.Cells[1, SG.RowCount-1]:= MyCont.Telephon;

SG.Cells[2, SG.RowCount-1]:= MyCont.Note;

end;

finally

CloseFile(f);

end;

end;

Здесь мы, прежде всего, получаем адрес запускаемой программы в переменную adres:

adres:= ExtractFilePath(ParamStr(0));

Как вы помните про консольные приложения, в любом проекте доступен параметр ParamStr(0). В этом параметре хранится адрес и имя загружаемой программы. Функция ExtractFilePath отсекает имя файла, возвращая только его адрес с завершающим "\", например:

**C:\Program Files\MyProg\**

Адрес, откуда запущена программа, мы и сохраняем в переменную adres. Поскольку она глобальная, то становится доступной во всем модуле главной формы. Поэтому мы воспользовались этой переменной для получения адреса в событии OnClose, в котором прописали код сохранения данных в файл. Так как событие OnCreate выполняется первым, то адрес в переменной adres действительно, уже будет. Так что теперь данные будут сохраняться.

Далее мы настраиваем сетку - указываем ширину столбцов, вписываем заголовки колонок. Как это делается, мы разбирали в прошлой лекции.

Затем мы смотрим - есть ли файл? Программа может запускаться впервые, или пользователь мог удалить старый файл, в любом из этих случаев мы выходим, не производя дальнейших действий. Сетка все равно уже настроена, так что программа будет выглядеть, как надо. Да и адрес программы мы уже поместили в переменную adres.

Но если файл есть, то нужно считать из него данные, и поместить их в сетку. Работа с файлом вам знакома, так что описывать процесс не буду. Замечу только, что в данном случае мы воспользовались циклом

while not Eof(f) do begin

То есть, делать, пока не конец файла. При открытии файла его указатель устанавливается в начало. Когда мы считываем первую запись в переменную MyCont, то указатель перемещается к следующей записи. Так, раз за разом, считывая запись в переменную, а затем, копируя из нее данные в сетку, мы двигаемся до конца файла, обрабатывая все записи, какие там есть. Причем при каждом шаге цикла мы сначала добавляем в сетку новую строку:

SG.RowCount:= SG.RowCount + 1;

а затем в эту, последнюю строку, мы записываем считанные из файла данные.

Итак, на данном этапе, мы научили программу добавлять новые записи (событие OnClick кнопки bAdd), научили её сохранять эти записи в файл (событие формы OnClose) и считывать их из файла (событие формы OnCreate). Осталось научить программу редактировать существующие записи - вдруг пользователь что то решит исправить, например, сменился телефон контакта, удалять контакты, ставшие ненужными, а также сортировать список по алфавиту. Ведь пользователь будет вносить контакты, как придется, не по порядку, а нужную запись гораздо проще найти в сортированном списке.

Итак, сгенерируйте событие OnClick для кнопки bEdit ("**Редактировать контакт**"). Код:

procedure TfMain.bEditClick(Sender: TObject);

begin

//если данных в сетке нет - просто выходим:

if SG.RowCount = 1 then exit;

//иначе записываем данные в форму редактора:

fEdit.eName.Text:= SG.Cells[0, SG.Row];

fEdit.eTelephone.Text:= SG.Cells[1, SG.Row];

fEdit.CBNote.Text:= SG.Cells[2, SG.Row];

//устанавливаем ModalResult редактора в mrNone:

fEdit.ModalResult:= mrNone;

//теперь выводим форму:

fEdit.ShowModal;

//сохраняем в сетку возможные изменения,

//если пользователь нажал "Сохранить":

if fEdit.ModalResult = mrOk then begin

SG.Cells[0, SG.Row]:= fEdit.eName.Text;

SG.Cells[1, SG.Row]:= fEdit.eTelephone.Text;

SG.Cells[2, SG.Row]:= fEdit.CBNote.Text;

end;

end;

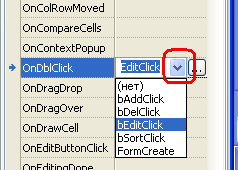
Давайте разбираться с кодом. Как обычно, мы выходим из процедуры, если данных в сетке нет - редактировать ведь нечего. Иначе перед показом окна редактора контактов мы проделываем подготовительную работу - заносим в строки TEdit этой формы данные контакта - имя и телефон. Также свойству Text списка выбора TComboBox мы присваиваем текст примечания. Все это позволит представить пользователю данные текущей записи. Вы помните, как узнать индекс текущей строки сетки? Этот индекс храниться в свойстве Row:

fEdit.eName.Text:= SG.Cells[0, SG.Row];

Далее, ModalResult формы редактора мы устанавливаем в mrNone, чтобы потом можно было определить, каким образом пользователь закрыл этот редактор.

Потом мы выводим форму, как модальное окно. В заключение мы проверяем - нажал ли пользователь кнопку "**Сохранить контакт**"? Если да, то мы перезаписываем данные в сетке на новый вариант, из формы редактора.

Нажав на кнопку "**Редактировать контакт**", пользователь сможет воспользоваться этим кодом. Однако в хороших программах пользователю часто предоставляют различные инструменты для выполнения одной задачи. Например, хорошим тоном считается редактировать запись сетки при двойном клике по ней. Давайте, реализуем эту возможность, тем более что это совсем просто. Выделите сетку. В **Инспекторе объектов** перейдите на вкладку **События**. Только не торопитесь генерировать новое событие, ведь писать новый код нам не придется. В событии OnDblClick нажмите кнопку выбора, и выберите там событие bEditClick - то есть то, которое мы только что описали:



Выбор уже существующего события

Таким образом, и при нажатии кнопки "**Редактировать контакт**", и при двойном клике по записи сетки будет выполнена одна и та же процедура.

Теперь научим программу удалять контакты. Сгенерируйте OnClick для кнопки "**Удалить контакт**":

procedure TfMain.bDelClick(Sender: TObject);

begin

//если данных нет - выходим:

if SG.RowCount = 1 then exit;

//иначе выводим запрос на подтверждение:

if MessageDlg('Требуется подтверждение',

'Вы действительно хотите удалить контакт "' +

SG.Cells[0, SG.Row] + '"?',

mtConfirmation, [mbYes, mbNo, mbIgnore], 0) = mrYes then

SG.DeleteRow(SG.Row);

end;

Этот код проще, да? Опять мы выходим из процедуры, если данных нет, ведь тогда и удалять нечего. Удалить запись просто, но нужно ли это делать? Пользователь мог нажать на кнопку не подумав, или случайно. Поэтому перед удалением мы запросим подтверждения:

if MessageDlg('Требуется подтверждение',

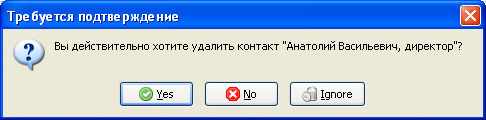
'Вы действительно хотите удалить контакт "' +

SG.Cells[0, SG.Row] + '"?',

mtConfirmation, [mbYes, mbNo, mbIgnore], 0) = mrYes then

SG.DeleteRow(SG.Row);

Программа выведет подобное сообщение:



Запрос на подтверждение

И только если пользователь подтвердит удаление контакта, он будет удален:

SG.DeleteRow(SG.Row);

Нам осталось сгенерировать событие OnClick для кнопки "**Сортировать список**". Его код:

procedure TfMain.bSortClick(Sender: TObject);

begin

//если данных в сетке нет - просто выходим:

if SG.RowCount = 1 then exit;

//иначе сортируем список:

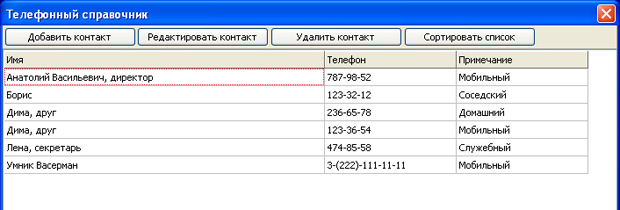
SG.SortColRow(true, 0);

end;

Код достаточно простой, мы выполняем сортировку, если есть данные, по первому столбцу (с именами контактов):

SG.SortColRow(true, 0);

Сохраните проект, запустите его и попробуйте в работе. Приложение должно работать, и выглядеть примерно так:



Окно работающего приложения

Если есть желание, можете дополнительно расширить проект - самостоятельно сделать окно "**О программе**" к этому приложению, реализовать механизм сортировок иначе, где пользователь мог бы выбрать способ сортировки: по именам контактов, по телефонам, или по типам телефонов.