Автор: Беляков А.Ю.

КОНВЕРТАЦИЯ ДАННЫХ

ВВОДНАЯ ЧАСТЬ

Одна из основных особенностей обучения в высшей школе заключается в том, что постоянный внешний контроль заменяется самоконтролем, активная роль в учении принадлежит уже не столько преподавателю, сколько студенту. Значительное учебное время отводится для самостоятельной работы студента. Основными формами самостоятельной работы студента являются изучение конспекта лекций, анализ рекомендованной литературы, исследование дополнительных вопросов изучаемой темы, выполнение практических проблемных заданий. В процессе самостоятельной работы студент приобретает навыки самоорганизации, самоконтроля, самоуправления, рефлексии и становится активным самостоятельным субъектом учебной деятельности.

ОСНОВНАЯ ЧАСТЬ

Значительный сегмент разрабатываемых программных продуктов связан с автоматизацией обработки данных. Предполагается, что приложение принимает поток входных данных, обрабатывает их согласно предписанному функционалу и сохраняет их в одном из подходящих форматов. В данной последовательности у разработчика могут возникнуть проблемы на этапе обработки входных файлов и формирования выходных.

Какими могут быть входные/выходные файлы? К наиболее распространенным вариантам отнесем: файлы с последовательным доступом (неструктурированы, текстовые), файлы с произвольным доступом (структурированы, основаны на записях, типизированные), файлы специализированных форматов (базы данных, табличные процессоры).

Для успешного освоения текущей темы следует иметь навыки работы с типизированными файлами и с базами данных:

– особенности доступа к текстовым и типизированным файлам, стандартные подпрограммы для работы с ними и обработку ошибок ввода/вывода мы обсуждали в рамках темы «Файлы» – http://delphi-pgsha.narod.ru/labrab/labrab3.pdf.

– особенности подключения к базам данных и выполнения запросов мы обсуждали в теме «Проектирование системы управления базами данных» (понимание этой темы и навыки работы с объектами базы данных через ADO обязательны) – http://delphi-pgsha.narod.ru/labrab/labrab5.pdf.

В данной теме мы затронем вопросы подключения приложения к таблицам из файла Microsoft Excel и формирования в них отчетности (по предметной области).

Необходимо освоить:

– способы доступа к данным (чем открывать файл);

– динамическое определение формата файла (версия офиса);

– варианты организации отображения данных в приложении;

– варианты организации взаимодействия с данными;

– порядок замены данных в файле или сохранения их в новый файл.

Не забывайте каждое новое приложение сохранять в отдельной папке во избежание потерь программного кода и неразберихи в подключаемых модулях.

Способы доступа к данным Microsoft Excel.

Обеспечить доступ к данным в файле Microsoft Excel можно несколькими способами:

1) через OLE–объекты;
2) через ADO–объекты;
3) через библиотеки сторонних производителей.

Прежде чем приступать к исследованию указанных технологий следует немного обсудить сложности и возможности их использования.

1. **OLE** (*Object Linking and Embedding*) – технология связывания и внедрения объектов в другие документы и объекты, разработанная корпорацией Майкрософт. OLE позволяет передавать часть работы от одной программы редактирования к другой и возвращать результаты назад. Например, установленная на персональном компьютере издательская система может послать некий текст на обработку в текстовый редактор, либо некоторое изображение в редактор изображений с помощью OLE-технологии. Однако, если OLE оперирует таблицами Microsoft Excel, то программа Excel должна быть инсталлирована на машине пользователя.

2. **ADO** (*ActiveX Data Object*) – интерфейс программирования приложений для доступа к данным, разработанный корпорацией Майкрософт и основанный на технологии компонентов ActiveX. Прежде всего данная технология ориентирована на обработку данных MS Access и MS SQL Server. Интерфейс ADO позволяет не только обрабатывать данные в объектно-ориентированном виде, но и получать доступ к разнообразным источникам (реляционные базы данных, текстовые файлы, файлы Excel).

3. В качестве альтернативы непосредственного использования указанных выше технологий можно рассмотреть опосредованное их использование в рамках сторонних библиотек (платных или бесплатных), подключаемых к среде программирования:

TMS Flexcel Studio
http://www.tmssoftware.com/site/flexcel.asp – $125

ARExcelReport
http://www.vector-ski.com/reports/arexcelreport_index.htm – $125 , для целей обучения бесплатно

Don Excel Report
http://www.don-soft.com.ar/DonExcelReport/products.php – $75

AfalinaSoft XL Report
http://www.afalinasoft.com/download-xl-report.html – 2003 г.

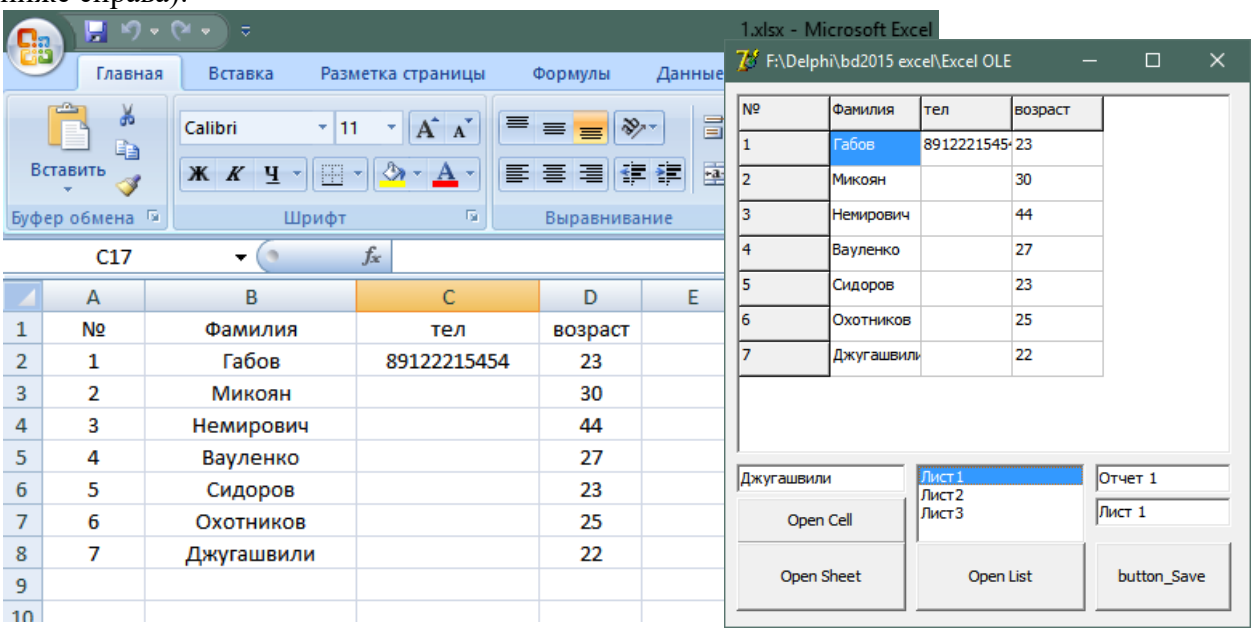
FlexCelReport
http://www.freewebs.com/flexcel/ – бесплатный, для Delphi 5, 6, 7.

Изучение сторонних библиотек выходит за рамки данного пособия.

Технология OLE.

Предварительно опишем функционал приложения – программа будет открывать указанный пользователем файл Microsoft Excel и выводить на форме содержимое первой ячейки с первой страницы из указанной книги.

Для начала создайте директорию для нашего приложения, создайте в ней файл Microsoft Excel, заполните первый лист содержимым по образцу (см. ниже слева), сохраните его и закройте. В программе предстоит много доработок, чтобы не было путаницы присмотритесь к возможному внешнему виду итоговой программы по данному вопросу (см. ниже справа):



Параграф А. Читаем файл.

Создайте приложение в среде Delphi, сохраните проект в той же папке и не забудьте в раздел подключаемых модулей добавить модуль **`ComObj`** !!!

В раздел глобальных переменных добавьте строковую переменную **`dir`** для хранения пути к файлу! На форме разместите диалог открытия файла (переименуйте с opendialog1 в od) со вкладки Dialogs, а также клавишу Button1 и поле Edit1. Можете самостоятельно настроить фильтр диалога для открытия файлов с расширением xls и xlsx.

Организуем работу приложения следующим образом:

– при запуске программы и обработке события создания формы в процедуре FormCreate программа определит текущую директорию и установит её как начальный путь для диалога (od) открытия файла;

– при клике мышкой по клавише Button1 произойдет обработка диалога и при выборе пользователем файла будет запущена функция Open_1 (в которую в качестве атрибута передается имя файла), возвращающая значение первой ячейки с первой страницы из указанной книги;

– функция Open_1 выполняет следующие действия (см. построчно в процедуре ниже) – создает OLE объект для книги Excel, открывает в нём указанную пользователем книгу, загружает данные из первого листа, импортирует значение первой ячейки и закрывает OLE-объект.

Для примера используйте следующий листинг процедур:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  getdir(0,dir); od.InitialDir:=dir;
end;
```

```
function Open_1(const FileName: string): string;
var ExlApp, ExlBook, ExlSheet: OLEVariant;
begin
  ExlApp:=CreateOleObject('Excel.Application');
  ExlBook:=ExlApp.Workbooks.Open(FileName);
  ExlSheet:=ExlBook.Worksheets[1];
  result:=ExlSheet.Range['A1'].Text;
  ExlApp.Quit;
end;

procedure TForm1.button1Click(Sender: TObject);
begin
  if od.Execute then edit1.Text:=Open_1(od.FileName);
end;
```

Обратите внимание, что строку функции Open_1, которая возвращает результат можно написать не в виде диапазона, а в более удобном виде с адресацией на номер строки и номер столбца ячейки:

```
result:=ExlSheet.cells.Item[1,1];
```

Апробируйте предложенный вариант и определите: где номер строки, а где – столбца.

Задание # OLE 1.

Теперь самостоятельно выполните такое задание для нашего входного файла: доработайте функцию Open_1 так, чтобы она возвращала в качестве результата фамилию из последней строки таблицы. За основу кода возьмите один из операторов цикла и ищите последнюю строку в таблице путем сравнения содержимого ячейки из первого столбца с пустой строкой.

Очевидно, что затруднительно работать с двумерным массивом данных из таблицы через единственный компонент Edit1, поэтому продолжим разрабатывать приложение и дополним форму компонентом для отображения таблиц StringGrid (переименуйте в sg) с вкладки Additional.

Задание # OLE 2.

Опираясь на код из первого задания создайте процедуру Open_2 заполнения таблицы sg. Организуйте импорт данных использованием двух циклов: по строкам и по столбцам (анализируя заполненность фиксированных строки и столбца). Напомню, что количество строк и столбцов в компоненте StrinGrid можно менять динамически через свойства RowCount и ColCount.

Такая построчная обработка данных важна и в некоторых случаях уместна, особенно при последующем избирательном сохранении данных в файл. Однако для большинства задач оптимальным будет через OLE-объект получать сразу все ненулевые данные с листа. Для этого можно воспользоваться объектом `Cells` (книги Excel) и его свойствами.

Для примера использования объекта `Cells` создадим на форме ещё одну кнопку и привяжем к ней процедуру `Open_3`, которая будет работать аналогично вышеописанной, но будет считывать не одну ячейку, а весь двумерный массив:

```
procedure Open_3(const XLSFile: string; var Grid: TStringGrid);
const xlCellTypeLastCell = 11;
var ExlApp, ExlBook, ExlSheet, ExlCell: OLEVariant;
    i, j: integer;
```

```
begin
  ExlApp := CreateOleObject('Excel.Application');
  ExlBook := ExlApp.Workbooks.Open(XLSFile);
  ExlSheet := ExlBook.Worksheets[1];
  ExlCell := ExlSheet.Cells.SpecialCells(xlCellTypeLastCell);
    Grid.RowCount := ExlCell.Row;
    Grid.ColCount := ExlCell.Column;
    for i := 1 to Grid.RowCount do
      for j := 1 to Grid.ColCount do
        Grid.Cells[j-1,i-1] := ExlSheet.Cells[i,j].Text;
  ExlApp.Quit;
end;

procedure TForm1.button3Click(Sender: TObject);
begin
  if od.Execute then Open_3(od.FileName, sg);
end;
```

Обсудим только отличие процедуры `Open_3`. Обратите внимание, что в данной процедуре появился второй аргумент `sg` – это объект для отображения таблиц `StringGrid`, который мы ранее разместили на форме. Мы передаем его в качестве параметра-переменной именно для того чтобы таблица заполнялась непосредственно в процедуре. Данный подход делает процедуру универсальной – передаём в неё имя файла Excel и имя объекта (того, который именно сейчас нужен) для заполнения и получаем заполненную таблицу. С помощью специального поля (`SpecialCells`) объекта `Cells` и определенной константы (`xlCellTypeLastCell`) мы получаем последнюю в правом-нижнем углу таблицы ячейку и сохраняем её в переменной для хранения ячеек `ExlCell`:

```
ExlCell := ExlSheet.Cells.SpecialCells(xlCellTypeLastCell);
```

Константа `xlCellTypeLastCell=11`, поэтому вполне можно написать так:

```
ExlCell := ExlSheet.Cells.SpecialCells(11);
```

и исключить константу из описания в шапке процедуры.

> Подобного рода константы характерны для продуктов от Microsoft, поэтому более подробное описание вы сможете найти в справке MSDN или просто в сети Интернет.

Далее с помощью этой переменной мы узнаём индексы строки и столбца последней ячейки: `ExlCell.Row` и `ExlCell.Column` – они нам нужны чтобы задать размеры таблицы `sg`. После чего с помощью двойного цикла мы заполняем таблицу `sg`, адресуясь последовательно к ячейкам на листе Excel: `ExlSheet.Cells[i,j].Text`.

Апробируйте процедуру Open_3.

Задание # OLE 3.

Опираясь на код Open_3 создайте новую процедуру Open_4, предназначенную для избирательного заполнения таблицы `sg` – необходимо заполнять таблицу только теми записями, в которых возраст находится в указанном пользователем диапазоне. В качестве простейшего UI используйте кнопку и два однострочных текстовых поля (для нижней и верхней границ диапазона возраста).

В книге Excel как правило несколько листов, рассмотрим вариант организации отображения списка доступных листов и возможности выбора листа для загрузки данных

пользователем. Добавьте на форму компонент ListBox1 со вкладки Standart и клавишу для запуска процедуры. Перенесите в раздел глобальных переменных декларацию всех переменных для хранения OLE-объектов, так как теперь доступ к листам книги, открытой в одной процедуре будет осуществляться уже в другой процедуре. Примерный вид раздела объявления глобальных переменных:

```
var  Form1: TForm1;
     dir: string;
     ExlApp, ExlBook, ExlSheet, ExlCell: OLEVariant;
```

Наименования всех листов книги в цикле перенесем в компонент список ListBox1, предварительно очистив его:

```
procedure TForm1.button_Open_ListClick(Sender: TObject);
var i: integer;
begin
  if od.Execute then
  begin
    ExlApp:=CreateOleObject('Excel.Application');
    ExlBook:=ExlApp.Workbooks.Open(od.FileName);
    ListBox1.Clear;
    for i:=1 to ExlBook.WorkSheets.Count do
      ListBox1.Items.Add(VarToStr(ExlBook.WorkSheets[i].Name));
  end;
end;
```

Подумайте: в каком случае понадобится предварительная очистка списка.

Далее пользователь кликает мышкой по необходимому листу и его содержимое выводится в таблицу sg. Для того чтобы отобразить содержимое выбранного листа не обязательно передавать его имя, достаточно знать его порядковый номер, который будет известен с кликом мышки из индекса компонента ListBox1:

```
ExlSheet:=ExlBook.Worksheets[ListBox1.ItemIndex+1];
```

Задание # OLE 4.

Сгенерируйте процедуру ListBox1Click *и самостоятельно заполните её так, чтобы при клике по имени листа в таблицу выводилось его содержимое. За основу возьмите код процедуры* Open_3.

Параграф Б. Сохраняем файл.

Возникают также задачи сохранения табличных результатов в книгу Excel. Рассмотрим один из вариантов использования технологии OLE для реализации указанной задачи. Подход работает при установленном офисе и наличии соответствующих драйверов в операционной системе. Если заранее нет информации о наличии соответствующего программного обеспечения, то в коде можно предусмотреть альтернативные пути реализации алгоритма.

Добавьте на форму два текстовых поля Edit для имени файла сохранения и для имени первого листа книги. Установите кнопку Button, сгенерируйте код обработки события «клик мышкой» по ней и заполните следующим кодом:

```
procedure TForm1.button_SaveClick(Sender: TObject);
const  xlsOld = $0000002B;  xlsNew = 56;
var  ExlApp, ExlSheet: OLEVariant;  i, j:integer;  s: string;
begin
  ExlApp := CreateOleObject('Excel.Application');
  ExlApp.Workbooks.Add(1);
  ExlSheet := ExlApp.Workbooks[1].WorkSheets[1];
  ExlSheet.name := edit2.Text; // имя листа
  for j:= 1 to sg.RowCount do
    for i:= 1 to sg.ColCount do
      ExlSheet.cells[j,i] := sg.Cells[i-1,j-1];
  ExlApp.DisplayAlerts := False;
  s := dir+'\'+Edit4.Text+'.xls'; // имя файла
  try
    ExlApp.Workbooks[1].saveas(s, xlsNew);
    ShowMessage('сохранено как 2007');
  except
    ExlApp.Workbooks[1].saveas(s, xlsOld);
    ShowMessage('сохранено как 2003');
  end;
  ExlApp.Quit;
end;
```

Апробируйте работу приложения, не забывайте нажимать F5 в проводнике для обновления списка доступных файлов.

Следует учесть, что при таком сохранении все элементы оформления будут утеряны, если в программном коде не прописать строчек, задающих оформление листа.

Если хотим создавать книгу с тремя листами по умолчанию, то следует использовать такой код:

```
ExlApp.Workbooks.Add;
```

Изменение цвета заливки ячейки допустимо следующим методом:

```
ExlSheet.cells[j,i].Interior.Color := RGB(0, 200, 200);
```

Изменение ширины колонки можно организовать так:

```
ExlSheet.Columns.Range['C:C',EmptyParam].ColumnWidth:=30;
```

Множество иных способов работы с оформлением и содержимым листа вы сможете найти в справке MSDN или просто в сети Интернет.

Задание # OLE 5.

Создайте новую процедуру `Save_1` *с параметрами (имя файла, имя листа, таблица с формы приложения) и перенесите туда функционал процедуры* `button_SaveClick` *(по подобию организации процедуры* `Open_3`). А в процедуре `button_SaveClick` оставьте только вызов `Save_1`.

Задание # OLE 6.

Создайте новую процедуру `Save_2` – выполняет ту же задачу что и Save_1, но каждую нечетную строку выделяет цветом заливки (можете дополнительно сделать так, чтобы пользователь определял цвет заливки).

Задание # OLE 7.

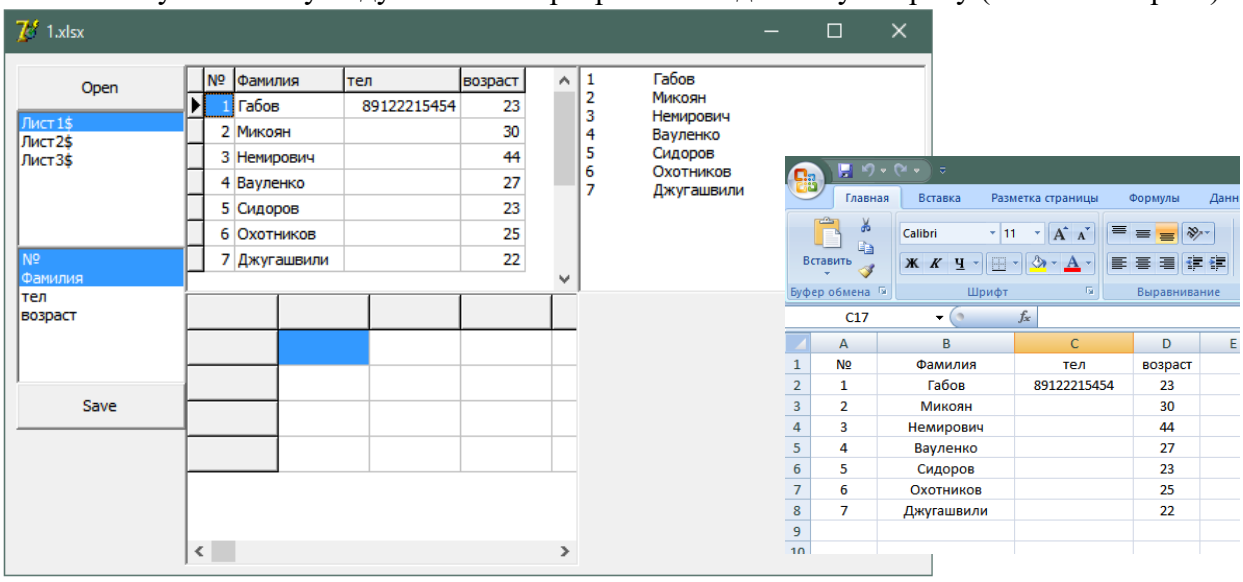
Создайте новую процедуру `Save_3` – выполняет ту же задачу что и `Save_1`, но записывает в файл все колонки кроме колонки с номером телефона.

Технология ADO.

Данная технология разработана корпорацией Microsoft, используется в различных средах программирования и удобна для организации обработки данных из разных источников, включая и Excel. Если вы уже работали в среде программирования с базами данных и подключали их к приложению, то вопрос подключения книги Excel у вас не вызовет затруднений. Через компонент ADOConnection подключается книга, каждый лист книги интерпретируется также как и отдельная таблица базы данных, поэтому можно использовать компоненты ADOQuery или DataSet – чтобы получать данные выборочно или целиком.

Создайте новую директорию, создайте там файл Excel (сделайте две версии в формате xls и xlsx, например, 1.xls и 1.xlsx), заполните содержимым по примеру (см. рис. ниже) и в этой же папке создайте новое приложение Delphi. Предварительно опишем функциональность нового приложения: открывает указанный пользователем файл, автоматически определяет его формат (новый/старый), подгружает список листов книги, при клике по имени листа загружает с него данные в таблицу, организует фильтрацию (выборочный показ данных), предоставляет пользователю выбор «сохранять / не сохранять» изменения.

В программе предстоит много доработок, чтобы не было путаницы присмотритесь к возможному внешнему виду итоговой программы по данному вопросу (см. ниже справа):



Шаг 1. Соединяемся с книгой.

Для размещения компонентов ADO обычно используют DataModule. Создайте его, переименуйте сразу в инспекторе объектов в DM, сохраните модуль под именем UnitDM в той же папке что и приложение и подключите его к модулю Unit1 приложения (меню File/Use Unit или вручную в разделе uses). На форме DM разместите компонент для подключения ADOConnection1 (с вкладки ADO) и переименуйте его в con1. Сразу отключите проверку логина/пароля (в инспекторе объектов). Обычно мы в настройках инспектора объектов ещё на начальном этапе разработки подключаем базу данных, но сейчас книгу Excel мы будем подключать динамически чтобы обеспечить пользователю возможность выбора. Через инспектор объектов для компонента con1 ознакомьтесь с установленными в вашей системе Провайдерами – нас интересуют Microsoft.Jet.OLEDB.4.0 (для старого формата, до 2003 года) и Microsoft.ACE.OLEDB.12.0 (для нового формата, 2007 и старше). Давайте начнем с подключения файла старого формата (*.xls) и сделаем это в программном коде, привязав действие к процедуре клика мышкой по кнопке (добавьте на форму Button1):

```
procedure TForm1.button1Click(Sender: TObject);
begin
  with dm do
  begin
```

```
    con1.Connected:=false;
    con1.ConnectionString:=
      'Provider=Microsoft.Jet.OLEDB.4.0;'+
      'Data Source=1.xls;'+
      'Extended Properties="Excel 8.0";';
    con1.Connected:=True;
    Form1.Caption:='связь с файлом '+od.FileName+' установлена';
  end;
end;
```

Строку подключения я разбил на три только для удобства чтения и редактирования. Предварительное отключение компонента con1 поставлено тут для последующей доработки – это пригодиться на тот случай, когда пользователь будет менять подключаемый файл Excel. После запуска программы нажмите на клавишу Button1, если не выскочит никаких окон с обозначением ошибки, значит подключение прошло успешно!

Шаг 2. Пользователь выбирает книгу.

Сделайте копию файла 1.xls – 2.xls, сохраните в той же папке. На форму добавьте диалог открытия файла, переименуйте в od и настройте фильтры на открытие файлов *.xls и *.xlsx (это для следующего шага).

Во время старта программы будем определять текущую директорию, которую установим как начальный путь для диалога od. Обработчик события клик по клавише доработаем предварительным анализам – выбрал ли пользователь файл:

```
var Form1: TForm1;
    dir: string;
implementation
uses UnitDM;
{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
begin
  GetDir(0,dir); od.InitialDir:=dir;
end;

procedure TForm1.button1Click(Sender: TObject);
begin
  if od.Execute then
  with dm do
  begin
    con1.Connected:=false;
    con1.ConnectionString:=
      'Provider=Microsoft.Jet.OLEDB.4.0;'+
      'Data Source='+ od.FileName +';'+
      'Extended Properties="Excel 8.0";';
    con1.Connected:=True;
    Form1.Caption:='связь с файлом '+od.FileName+' установлена';
  end;
end;
```

Шаг 3. Автоматический выбор драйвера.

Если установленные в операционную систему драйверы нам позволят, то можно будет организовать открытие файлов разных форматов:

```
procedure TForm1.button1Click(Sender: TObject);
var s: string;
begin
```

```
  if od.Execute then
  with dm do
  begin
    con1.Connected:=false;
    s:=od.FileName;
    case s[Length(s)] of
    'x': con1.ConnectionString:=
           'Provider=Microsoft.ACE.OLEDB.12.0;'+
           'Data Source='+ s +';'+
           'Extended Properties="Excel 12.0 Xml";';
    's': con1.ConnectionString:=
           'Provider=Microsoft.Jet.OLEDB.4.0;'+
           'Data Source='+ s +';'+
           'Extended Properties="Excel 8.0";';
    end;
    con1.Connected:=True;
    Form1.Caption:=ExtractFileName(s);
  end;
end;
```

Обратите внимание, что в измененной процедуре появилась строковая переменная s для хранения имени файла – это просто для удобства последующей обработки. В операторе многоальтернативного выбора мы проверяем последний символ переменной s, чтобы определиться с форматом открываемого файла.

Шаг 4. Загружаем список листов книги.

Итак у пользователя уже есть возможность выбрать файл. Теперь добавим возможность выбора листа из книги Excel. Разместите на форме список ListBox1 (с вкладки Standart), переименуйте его в Box1 и после строчки con1.Connected:=True; в процедуре button1Click добавьте код:

```
con1.GetTableNames(Box1.Items);
```

Данный метод GetTableNames позволяет получить список таблиц/листов и загрузить его в переменную типа TStringList, в качестве которой мы подставили список Box1.Items – после загрузки мы сразу увидим результат в компоненте Box1. Апробируйте работу приложения.

Шаг 5. Загружаем данные с листа.

Добавим функционала – при клике мышкой по имени листа пользователь увидит данные с него. Прежде всего, заполните разные листы книги Excel немного разными данными. Для реализации задуманного следует добавить в модуль данных компонент ADODataSet (переименуйте в set1). ADODataSet – компонент для получения набора данных из одной или нескольких таблиц из файла или передачи набора данных из визуального компонента в файл. Следует отметить, что ADODataSet совмещает возможности ADOTable и ADOQuery, то есть может формировать или обрабатывать набор данных из одной таблицы или избирательно из разных таблиц. Через инспектор объектов подсоедините set1 к con1 (поле Connection). Создайте процедуру для обработки клика по списку:

```
procedure TForm1.Box1Click(Sender: TObject);
begin
  with dm do
  begin
    set1.Close;
    set1.CommandText:=
      'SELECT * FROM ['+Box1.Items[Box1.ItemIndex]+']';
    set1.Open;
  end;
end;
```

Обратите внимание, что отбор данных осуществляется с помощью инструкций SQL, таким образом повышается гибкость обработки данных. Постарайтесь ответить на вопрос: зачем нужны прямоугольные скобки в инструкции SQL? Апробируйте процедуру – но где же отображаются данные с листа? Мы ещё не организовали вывод – данные находятся в наборе set1.

Используйте для пробы такой подход:

```
Form1.Caption:=set1.Fields.Fields[0].AsString;
```

– эту строчку следует добавить сразу за командой открытия набора данных `set1.Open`. Апробируйте приложение, посмотрите что происходит при переключениях между листами, попробуйте пояснить происходящее. Что произойдет, если индекс 0 сменить на 1?

Отмечу, что набор данных ADODataSet считает первую строчку в листе Excel заголовками (полями), поэтому к ним можно обратиться через коллекцию FieldList. Разместите на форме ещё один список и переименуйте его в Box2, добавьте в процедуру Box1Click также после set1.Open ещё одну строчку кода :

```
Box2.Items:=set1.FieldList;
```

– это позволит нам увидеть список столбцов выбранной таблицы. Данный код не простая демонстрация возможностей, а очень полезен для организации гибкого интерфейса пользователя по фильтрации полезной информации (рассмотрим далее).

Итак, раз мы имеем доступ к набору данных, то мы сможем и вывести его на визуальный компонент. Возможны несколько вариантов, но мы апробируем вывод в компонент в DBGrid, Memo и в StringGrid.

Вывод в DBGrid.

Добавьте в модуль DM компонент DataSource (переименуйте в ds1) с вкладки DataAccess и подсоедините к set1. Затем добавьте на главную форму приложения компонент DBGrid (переименуйте в dg) и подсоедините к ds1. Приготовления уже закончены и больше ничего делать не нужно – апробируйте работу приложения. При клике мышкой по имени листа в компоненте dg будет отображаться таблица.

Если вас не устраивает ширина столбцов, то её можно динамически менять:

```
for i:=0 to dg.FieldCount-1 do
  dg.Columns[i].Width:=60;
```

Вывод в Memo.

Добавьте на форму компонент Memo1. Организуем такой подход: очистим Memo, установим указатель на первую запись в наборе данных, пока указатель не достиг позиции после последней записи реализуем добавление в Memo1 текущей записи с помощью цикла и переходим к следующей записи. Итоговая процедура может иметь такой код:

```
procedure TForm1.Box1Click(Sender: TObject);
var i: Integer; s: string;
begin
  with dm do
  begin
    set1.Close;
    set1.CommandText:=
      'SELECT * FROM ['+Box1.Items[Box1.ItemIndex]+']';
    set1.Open;

    box2.Items:=set1.FieldList;
    for i:=0 to dg.FieldCount-1 do
      dg.Columns[i].Width:=59;

    memo1.Clear;
    set1.First;
    while not set1.Eof do
    begin
      s:='';
```

```
      for i:=0 to set1.FieldCount-1 do
        s:=s+set1.Fields.Fields[i].AsString+chr(9);
      memo1.Lines.Add(s);
      set1.Next;
    end;

  end;
end;
```

Попробуйте дать пояснения этим участкам кода:

- `chr(9);`
- `set1.FieldCount-1.`

Вы, конечно, уже обратили внимание, что табличное отображение данных в текстовом виде не совсем удобно. Однако часто встречаются задачи именно в таком виде хранить и обрабатывать данные. Для DBGrid и StringGrid этот недостаток отсутствует.

Задание # ADO 1.

Доработайте процедуру так чтобы в Memo выводились данные без столбца телефон – пока используя номер столбца как известную константу.

Вполне вероятно возникновение такого требования к пользовательскому интерфейсу: необходимо чтобы пользователь сам мог выбрать столбцы для отображения. Как же это реализовать? Очень удобный и интуитивно понятный вариант, когда в списке появляется возможность мультивыбора опций. Установите для компонента Box2, через инспектор объектов, значение поля MultiSelect в положение true. Теперь во время работы программы появляется возможность, при зажатой клавише CTRL, выделять мышкой несколько опций. Только следует разнести функицонал: в первом списке (процедура `Box1Click`) оставьте часть кода, которая получает набор данных (таблицу) и выводит во второй список названия полей (заголовки таблицы). Создайте аналогичную процедуру для второго списка – `Box2Click`. Туда перенесите заполнение Memo1 циклами while и for. Если перенесете код без изменений, то при любом клике по второму списку поле Memo1 будет заполняться всеми столбцами из набора данных set1. Для реализации задуманного функционала можно в цикле проверять свойство Box2.Selected[i], которое возвращает true, если на текущий момент i-тая опция выделена.

Задание # ADO 2.

Доработайте процедуру `Box2Click` *так чтобы в Memo выводились только те столбцы, которые выделил пользователь.*

Вывод в StringGrid.

Добавьте на форму компонент для отображения табличных данных – StringGrid (переименуйте в sg). Здесь используется тот же подход, что и при выводе данных в Memo, то есть двойной цикл – перебор по строкам и по столбцам. Следует учесть необходимость динамического изменения размеров таблицы – через свойства ColCount и RowCount.

Задание # ADO 3.

Доработайте процедуру `Box2Click` *так чтобы в StringGrid выводились данные из с выбранного пользователем листа, но только те столбцы, которые выделил пользователь (по аналогии с предыдущим заданием).*

Шаг 6. Сохраняем данные.

Вывод информации из Memo или StringGrid в текстовый или типизированный файл не представляют особой сложности и здесь не будут обсуждаться. Дальше будем обсуждать работу с компонентом DBGrid, варианты внесения изменений в данные и подход к сохранению данных. Если мы собираемся работать с тем же файлом, книгой и листом, то открытое ранее соединение с набором данных `set1.Open` не должно быть закрыто к моменту сохранения. Изменения в ячейки можно двумя способами:

– внести вручную (как в обычном Excel – попробуйте);

– тривиальным присвоением.

Если вы вручную внесли / изменили / удалили данные в DBGrid, то чтобы изменения были сохранены в листе Excel нужно перевести набор данных в режим редактирования с последующей записью данных в файл:

```
set1.Edit;
set1.Post;
```

Задание # ADO 4.

Поместите на форму кнопку сохранения «ручных» изменений и внесите туда указанный код, апробируйте её работу.

Если вы собираетесь автоматизировать процесс заполнения таблицы, то нужно сначала установить курсор на необходимую запись (строку таблицы) и затем уже присвоить значение в нужное поле (столбец), например, так:

```
  DM.Set1.RecNo:=1;
  DM.Set1.Edit;
  DM.set1.FieldByName('возраст').AsString := '30';
  DM.Set1.Post;
```

или так:

```
  with DM.set1 do
  begin
    RecNo:=2;
    Edit;
    Fields.FieldByNumber(4).AsInteger := 606;
    Post;
  end;
```

Если предстоит обработать всю таблицу, то самый удобный вариант – поместить приведенный код в тело цикла. Обратите внимание, что можно использовать для реализации указанной возможности:

– цикл с предусловием через предварительную установку курсора в первую строку (`Set1.First`), проверкой условия не достижения конца таблицы (`not set1.Eof`) и смещением на одну строку курсора на каждом шаге цикла (`set1.Next`) – для примера можно взять процедуру вывода в Memo (см. выше).

– параметрический цикл `for`, с установкой на каждом шаге курсора на новую строку с помощью метода `RecNo` (см. выше).

Корректное написание приложения предполагает, что выходя из программы, мы освободим все открытые ресурсы, поэтому добавьте такую процедуру к своему приложению:

```
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  DM.Set1.Close;
end;
```

Задание # ADO 5.

Разработайте процедуру стирания столбца телефонов.

Задание # ADO 6.

Разработайте процедуру изменения формата записи телефонов – нужно убрать первую цифру 8, оставив только 10 цифр номера.

ПОДВЕДЕНИЕ ИТОГОВ

Описанные подходы не исчерпывают многообразия методов работы с данными.

Все разработанные приложения сохраните в отдельных папках, ко всем апробированным «исходникам» компонентов добавьте своё описание, результаты проделанной работы на электронном носителе предоставьте на проверку преподавателю.

Оцениваться будет объем и полнота проделанной работы, сложность разработанных приложений, корректность настроек компонентов.