Лабораторная работа №4

Задача состоит в том, чтобы из Excel-файла взять данные и поместить в файл другого формата: dbf, csv, xml – причём xml в двух вариантах – через элементы и через атрибуты.

Исходный Excel-файл приготовьте самостоятельно со своими какими-нибудь данными. Предлагается следующий образец:

	A	B	C	D	E	F
1			**Список пролеченных в стационаре**			
2			**март 2022 год**			
3						
4	№ п/п	мес	Ф.И.О.	Дата рождения	к/дни	Сумма
5						
6						

Обычно подобные файлы служат для обмена информацией между какими-то сторонами, которые и договариваются о формате обмена – то есть, если это XML, то – как должна выглядеть его структура, названия полей, типы, форматы типов (даты, вещественных чисел и т.д.).

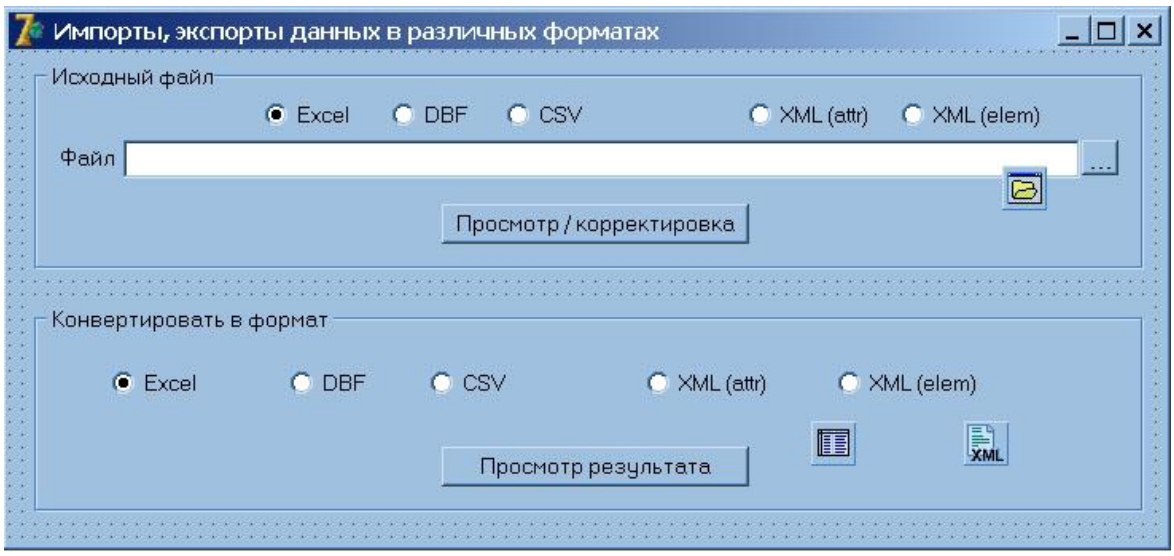
Аналогично – по DBF-формату.

По CSV-формату дополнительно ещё требуется договориться о знаках разделения полей – чтобы как можно надёжнее (однозначно) были приняты данные другой стороной. В нашем случае считаем, что поля разделяются знаками табуляции (код 09). Обычно ещё для надёжности просят в первой строке перечислить названия полей. Сам же CSV-формат представляет собой текстовый файл: одна строка – одна запись. Таким образом, с CSV-файлом вполне можно работать, как с текстовым файлом – с соответствующими стандартными функциями и процедурами.

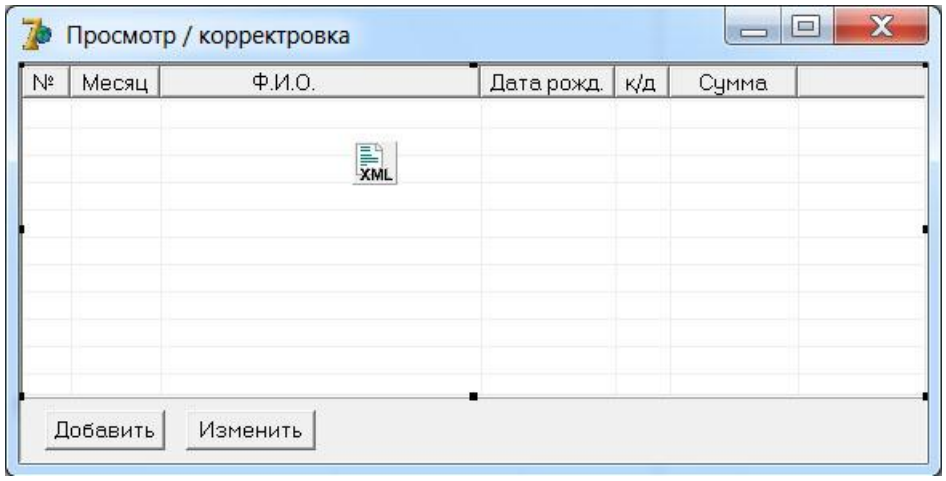
XML-файл – чтобы его прочитать, приходится пользоваться компонентом XMLDocument (есть такой в Delphi, вкладка Internet). При формировании его можно обойтись методами текстового файла, понимая, как должен выглядеть этот текст, хотя, изучив XMLDocument, можно и с его помощью.

С DBF-форматом понятно, что удобнее всего использовать компонент Table, который находится на вкладке BDE.

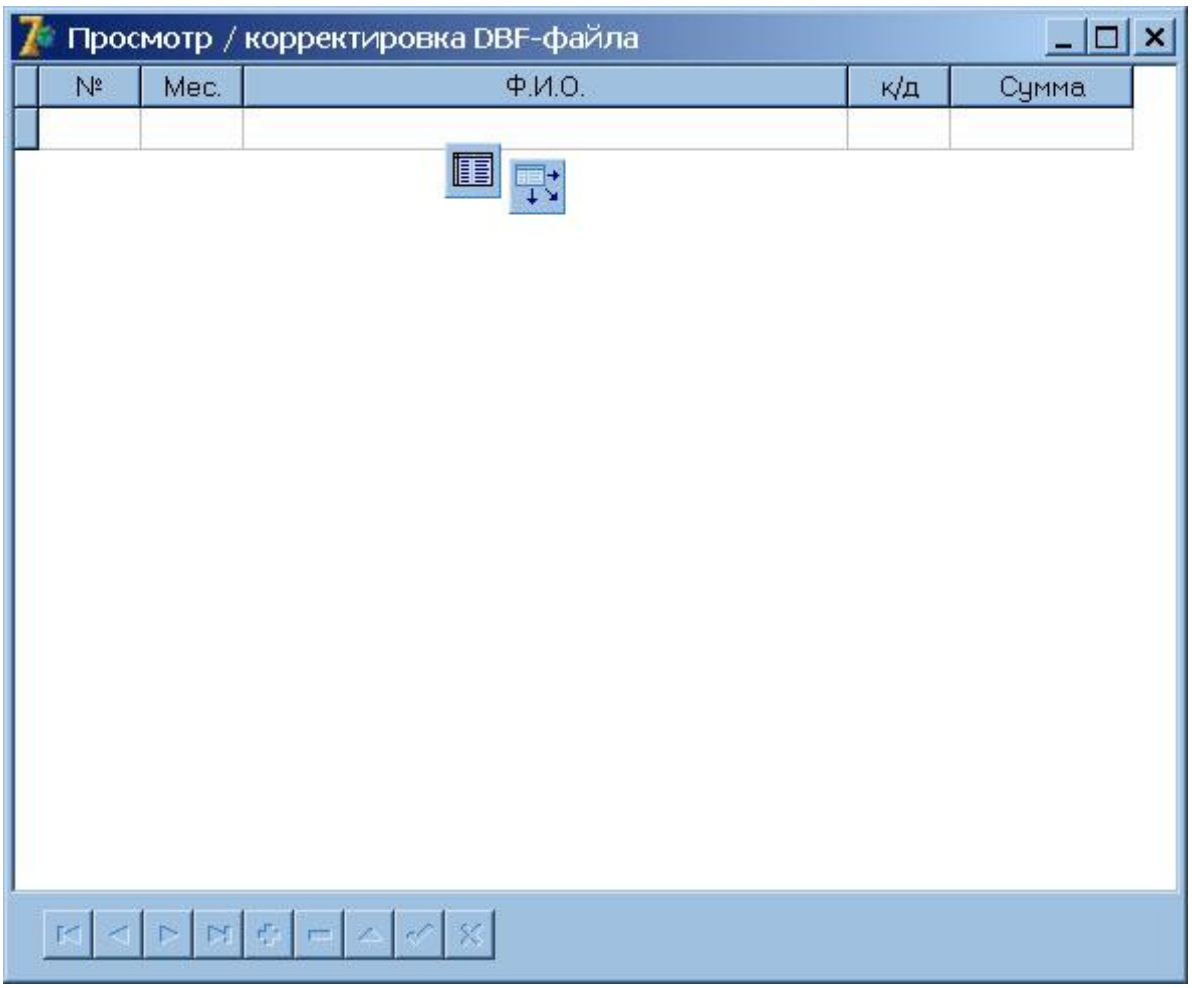
Интерфейс программы может выглядеть следующим образом:



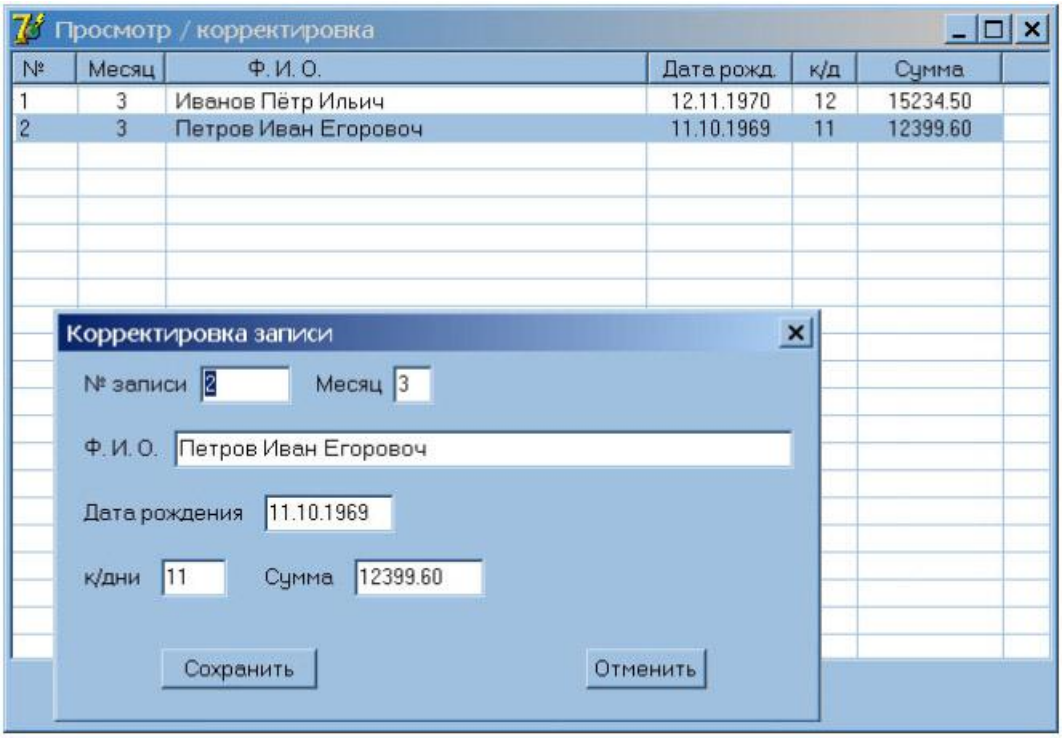
Далее подсоединяются ещё формы:



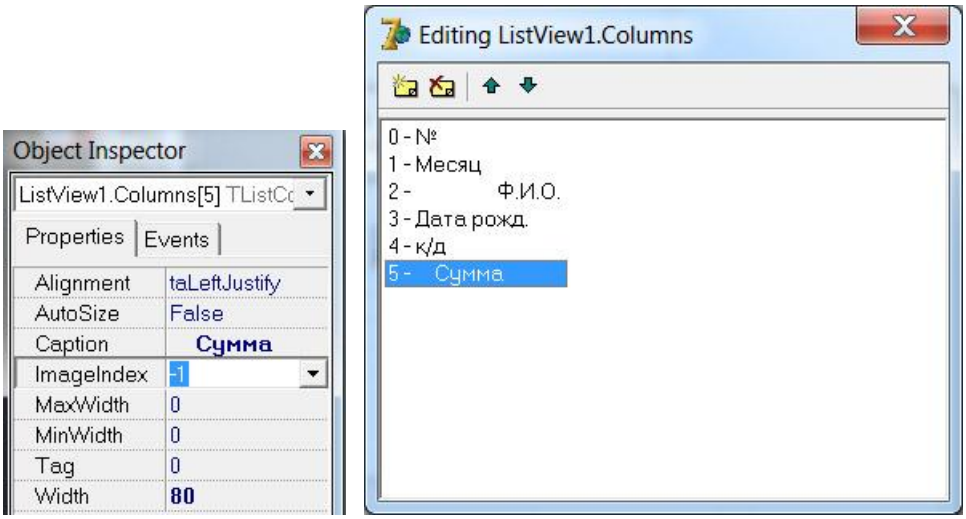
Здесь используется компонент ListView – для просмотра загруженных файлов типа XML, CSV, XLS. А ниже – форма для просмотра файла типа DBF.



В компоненте ListView невозможно делать корректировки данных в полях записи, поэтому этот процесс вынесен в отдельную форму:



Объект ListView особенный. Во-первых, в Object Inspector надо указать свойство ViewStyle=vsReport, RowSelect=True, GridLines=True. Во-вторых, надо настроить столбцы. Щёлкнув правой кнопкой мышки по ListView, вызываем редактор столбцов:



Здесь, чтобы показать программную работу с компонентом ListView, для примера показан процесс добавления строк. В реальной программе этот фрагмент, конечно, надо будет переделать.

```
procedure TForm1.BtnAddClick(Sender: TObject);
var
  LI: TListItem;
begin
  ListView1.Clear;

  LI:= ListView1.Items.Add;
  LI.Caption:= '1';
  LI.SubItems.Add('3');
  LI.SubItems.Add('Иванов Пётр Ильич');
  LI.SubItems.Add('12.11.1970');
  LI.SubItems.Add('12');
  LI.SubItems.Add('15234.50');

  LI:= ListView1.Items.Add;
  LI.Caption:= '2';
  LI.SubItems.Add('3');
  LI.SubItems.Add('Петров Иван Егорович');
  LI.SubItems.Add('11.10.1969');
  LI.SubItems.Add('11');
  LI.SubItems.Add('12399.60');
end;
```

Пример обращения к выделенной строке в ListView:

```
var
  LI: TListItem;
begin
  LI:= ListView1.Selected;
  ShowMessage(LI.Caption + ' ' + LI.SubItems[0]+ ' ' + LI.SubItems[1]);
```

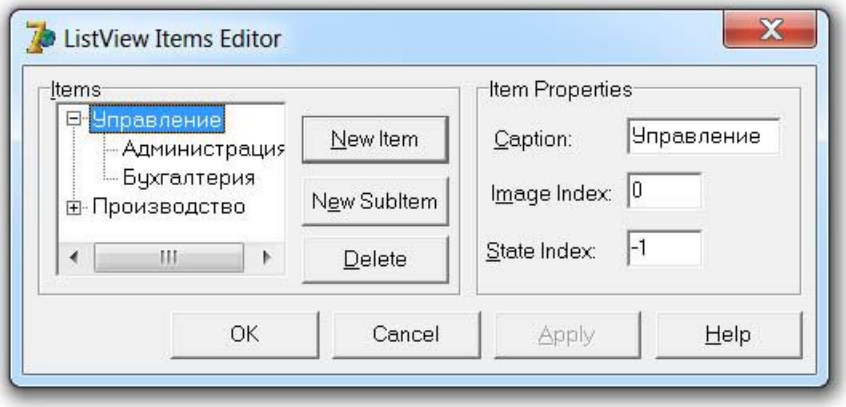
Пример перебора всех строк в ListView:

```
var
  LI: TListItem;
  i:integer;
begin
  for i:=0 to ListView1.Items.Count - 1 do begin
    LI:= ListView1.Items[i];
    ShowMessage(LI.Caption + ' ' + LI.SubItems[0]+ ' ' + LI.SubItems[1]);
  end;
```

Описанное здесь начало проекта студент может взять за основу и далее уже исправлять и дополнять его, чтобы получить рабочий вариант программы.

Рассмотрим немного подробнее компонент **ListView**. Он позволяет отображать данные в виде списков, таблиц, крупных и мелких пиктограмм. С подобным отображением все вы сталкиваетесь, раскрывая папки Windows.

Стиль отображения информации определяется свойством **ViewStyle**, которое может устанавливаться в процессе проектирования или программно во время выполнения. Свойство может принимать значения: **vsIcon** – крупные значки, **vsSmallIcon** – мелкие значки, **vsList** – список, **vsReport** – таблица. Что означает каждое из этих значений вы можете посмотреть в любой папке Windows на рабочем столе.



Окно редактора списка объектов компонента **ListView**

В этой задаче используем режим vsReport. Основное свойство компонента, описывающее состав отображаемой информации – **Items**. Во время проектирования оно может быть установлено специальным редактором, вызываемом щелчком на кнопке с многоточием рядом с этим свойством в окне Инспектора Объектов. Оно похоже на окно, описанное для компонента **TreeView**. Точно так же в нем задаются новые узлы кнопкой New Item и дочерние узлы – кнопкой New SubItem. Только смысл дочерних узлов другой: это информация, которая появляется только в режиме **vsReport** – в виде таблицы.

Для каждого узла задается свойство **Caption** – надпись, появляющаяся около пиктограммы. Для дочерних узлов это свойство (**SubItems**) соответствует надписи, появляющейся, в ячейках таблицы в режиме **vsReport**. Таким образом, в режиме **vsReport** для первого столбца здесь используется Caption, а для остальных – SubItems[0], SubItems[1], … и т.д.

Пример:

```
procedure TForm1.FormCreate(Sender: TObject);
var LI: TListItem;
begin
  LI:= ListView1.Items.Add;  // добавка пустой строки
  LI.Caption:= 'Первый столбец'; // заполнение строки
  LI.SubItems.Add('Второй столбец');
  LI.SubItems.Add('Третий столбец');
end;
```

Свойство Image Index определяет индекс пиктограммы. Индекс соответствует спискам изображений, хранящимся в отдельных компонентах **ImageList**. Указания на эти компоненты вы можете задать в свойствах LargeImages для режима **vsIcon** и SmallImages для режимов **vsSmallIcon**, **vsList** и **vsReport**. Индексы начинаются с 0. Если вы укажете индекс –1 (значение по умолчанию), пиктограммы изображаться не будут. Свойство State Index в панели Item Properties позволяет добавить вторую пиктограмму в данный объект. Подобная пиктограмма может просто служить дополнительной характеристикой объекта. Индекс, указываемый как State Index, соответствует списку изображений, хранящихся в отдельном компоненте **ImageList**, указанном в свойстве StateImages компонента **ListView**.

Свойство **Columns** определяет список заголовков таблицы в режиме **vsReport** при свойстве **ShowColumnHeaders** (показать заголовки), установленном в **true**. Свойство **Columns** можно задать в процессе проектирования специальным редактором заголовков, вызываемом двойным щелчком на компоненте **ListView** или щелчком на кнопке с многоточием рядом со свойством **Columns** в окне Инспектора Объектов.

Свойство **Checkboxes** определяет отображение индикатора с флажком около каждого элемента списка. Индикаторы можно устанавливать программно или их может изменять пользователь во время выполнения. Тогда узнать программно, установлен ли индикатор в некотором элементе **Items[i]**, можно проверкой его свойства **Checked**. Например:

```
for i:=0 to ListView1.Items.Count-1 do
  if ListView1.Items[i].Checked then
    ...;
```

Свойство списка **Selected** определяет выделенный пользователем элемент списка (типа `TListItem`).