# Списки хранящие записи

Рассмотрим более сложные списки, те списки одним из полей которых является запись.

Задача №506. Багаж пассажира характеризуется:

1. Фамилией, именем и отчеством пассажира.
2. Серия и номер паспорта.
3. Количеством вещей.
4. Весом багажа.

Из списка багажей найти багаж, средний вес одной вещи в котором отличается не более чем на 0,5 кг от общего среднего веса вещи в багаже всех пассажиров из списка.

Определимся с подзадачами, которые нужно реализовать, чтобы выполнить поставленную задачу:

1. Создать тип записи, в которой будут храниться данные о багаже.
2. Создать тип указателя на узел, одним из полей которого является запись о багаже.
3. Создать список из узлов.
4. Открыть текстовый файл.
5. Вывести содержимое списка в текстовый файл.
6. Закрыть текстовый файл.
7. Удалить список.
8. Закрыть приложение и посмотреть содержимое текстового файла.

Это первая часть задачи, решение которой позволяет перейти к основной части, а именно – анализу созданного списка. Сюда можно включить ещё одну подзадачу – сохранить данные из списка в типизированном файле, что позволит при новом запуске приложения использовать для создания списка данные из типизированного файла.

Выполнение первой части :

1. Создаём папку для сохранения проекта, например **Z506.**
2. В этой папку создадим текстовый файл для хранения результатов запуска приложения, например, **textZ506.txt**.
3. Запустим приложение, изменим имя формы на **F506** и сохраним проект под именем **P506**, а модуль формы пол именем **U506.**
4. Создаём модуль, в котором будут описаны используемые типы и подпрограммы **Unit506**.

**unit** Unit506;

**interface**

uses SysUtils, Dialogs; //Модули для ввода/вывода данных

**type**

Bag = **record** // Тип багажа

Fam, Nam, Otch : string[20]; // Фио клиента

ColV: byte; // Количество вещей в багаже

VesV: real; // Вес вещей в багаже

Npasp: string[20]; // Серия и номер паспорта клиента

end;

PUzel = ^Zp2; // Тип указателя на узел

Zp2 = **record** // Тип узла списка

x: Bag; // Сведения о багаже

next: PUzel; // Указатель на следующий узел

pred: PUzel; // Указатель на предыдущий узел

end;

FZap = file of Bag; //Файловый тип для хранения данных о багаже

{Операции для работы с двусвязанным списком }

**procedure** AddFirst(var f: PUzel; a: PUzel);{Вставить узел a первым в список}

**procedure** AddAfter(var old:PUzel; a: PUzel);{Вставить узел a после old}

{Построить список; f -указатель на голову списка}

**procedure** BuildSpisok(var f: PUzel);

{Вывод списка в текстовый файл}

**procedure** WriteSpText(var f: PUzel;var ftxt:Text);

{Выделить из списка первый узел и вернуть его пользователю}

**procedure** DelFirstElement(var f,a: PUzel);

{Выделить из списка узел,следующий за узлом old и вернуть его пользователю}

**procedure** DelElement(var old,a: PUzel);

**procedure** DelSpisok(var f: PUzel); //Удалить список

**implementation**

**procedure** AddFirst(var f: PUzel; a: PUzel);

begin

a^.next := f;

if f<> nil then f^.pred := a;

f:=a;

end;

**procedure** AddAfter(var old:PUzel; a: PUzel);

begin

a^.next := old^.next;

old^.next := a;

if a^.next <> nil then a^.next^.pred := a;

a^.pred := old;

end;

**procedure** BuildSpisok(var f: PUzel);

var

a,d :PUzel;

ch: char;

begin

f:= nil;

repeat

new(a);

with a^.x do

begin

Fam := InputBox('Введите фамилию',' ',' ');

Nam := InputBox('Введите имя',' ',' ');

Otch := InputBox('Введите отчество',' ',' ');

Npasp:= InputBox('Введите серию и номер паспорта',' ',' ');

ColV:= StrToInt(InputBox('Введите количество вещей',' ',' '));

VesV:= StrToFloat(InputBox('Введите вес вещей',' ',' '));

end;

a^.next := nil; //Это перестраховка. Почему?

a^.pred := nil; //Это перестраховка. Почему?

if (f = nil) then

begin

AddFirst(f,a);

d:= f;

end

else

begin

AddAfter(d,a);

d := a;

end;

ch:= InputBox('Для завершения ввода нажмите Y',' ',' ')[1];

until (ch = 'Y') or (ch ='y');

end;

**procedure** WriteSpText(var f: PUzel; var ftxt:Text);

var

p: PUzel;

s: string;

y: Bag;

begin

p:= f;

while not(p = nil) do

begin

y:= p^.x;

s:=y.Fam +' ' + y.Nam +' '+ y.Otch + ' '+ y.Npasp +' '+

IntToStr(y.ColV) + ' '+ FloatToStr(y.VesV);

writeln(ftxt, s);

p:= p^.next;

end;

end;

**procedure** DelFirstElement(var f,a: PUzel);

begin

a := f;

f := f^.next;

a^.next := nil;

if f<>nil then f^.pred := nil;

end;

**procedure** DelElement(var old,a: PUzel);

begin

if (old^.next = nil) then a:= nil //old последний узел в списке

else

if (old^.next^.next = nil) then //old предпоследний узел в списке

begin

a := old^.next;

a^.pred:= nil;

old^.next:= nil;

end

else

begin //за old не менее двух узлов в списке

a := old^.next;

old^.next := a^.next;

old^.next^.pred:= old;

a^.next := nil;

a^.pred:= nil;

end;

end;

**procedure** DelSpisok(var f: PUzel); //Удалить список

var

a: PUzel;

begin

while (f <> nil) do

begin

DelFirstElement(f,a);

Dispose(a);

end;

end;

end.

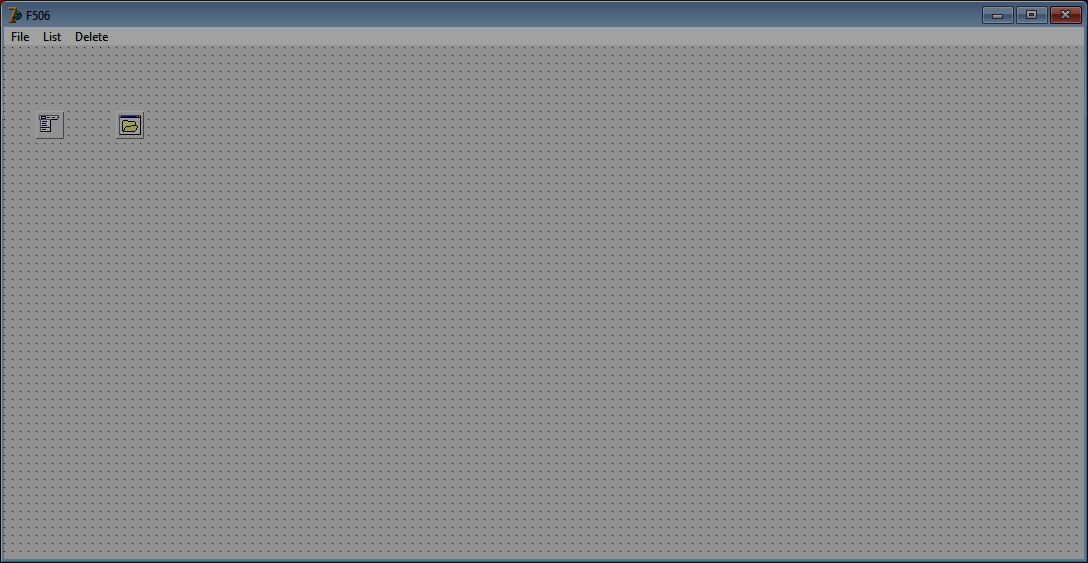
Данный модуль отличается от модуля предыдущей задачи:

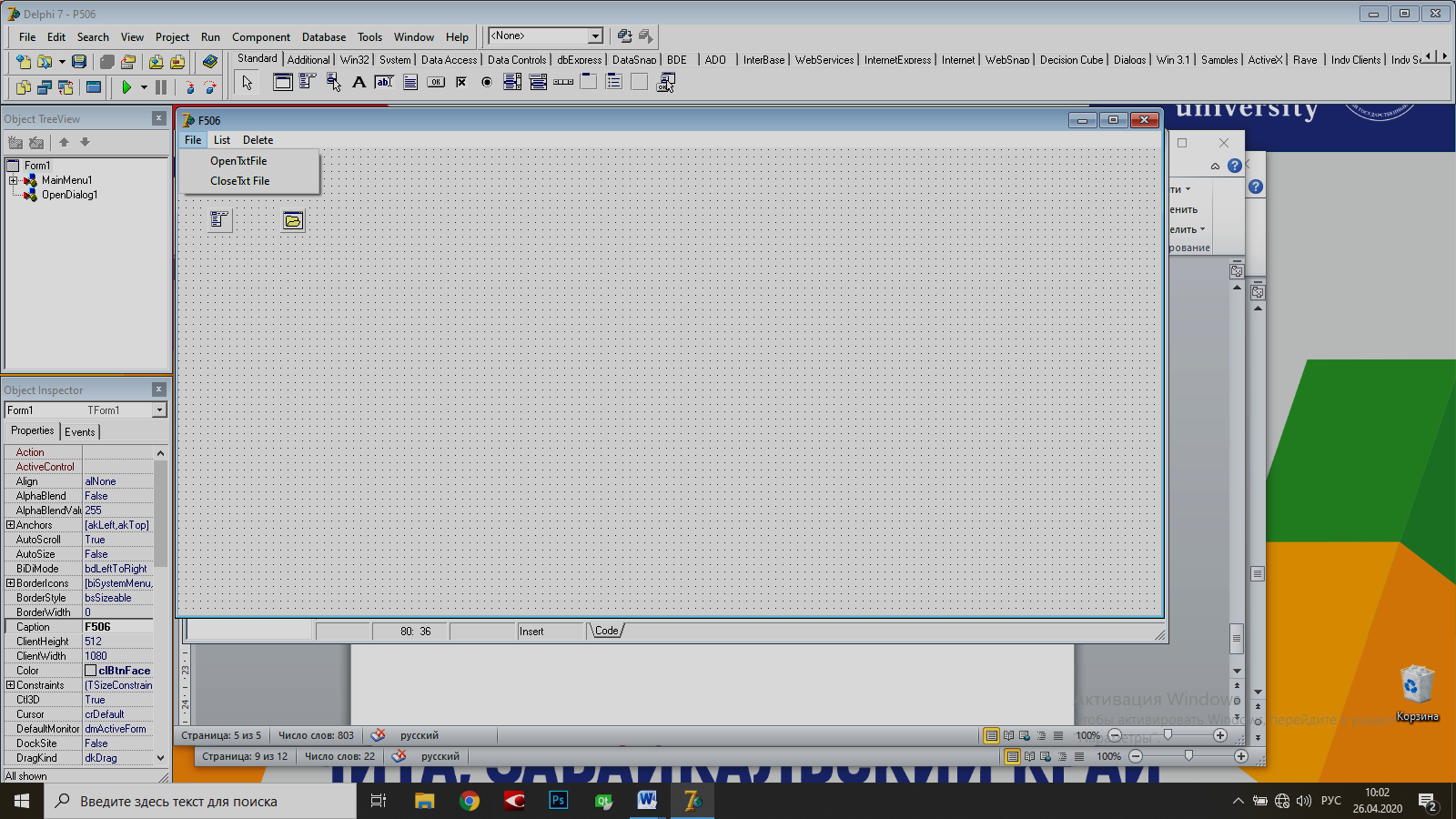
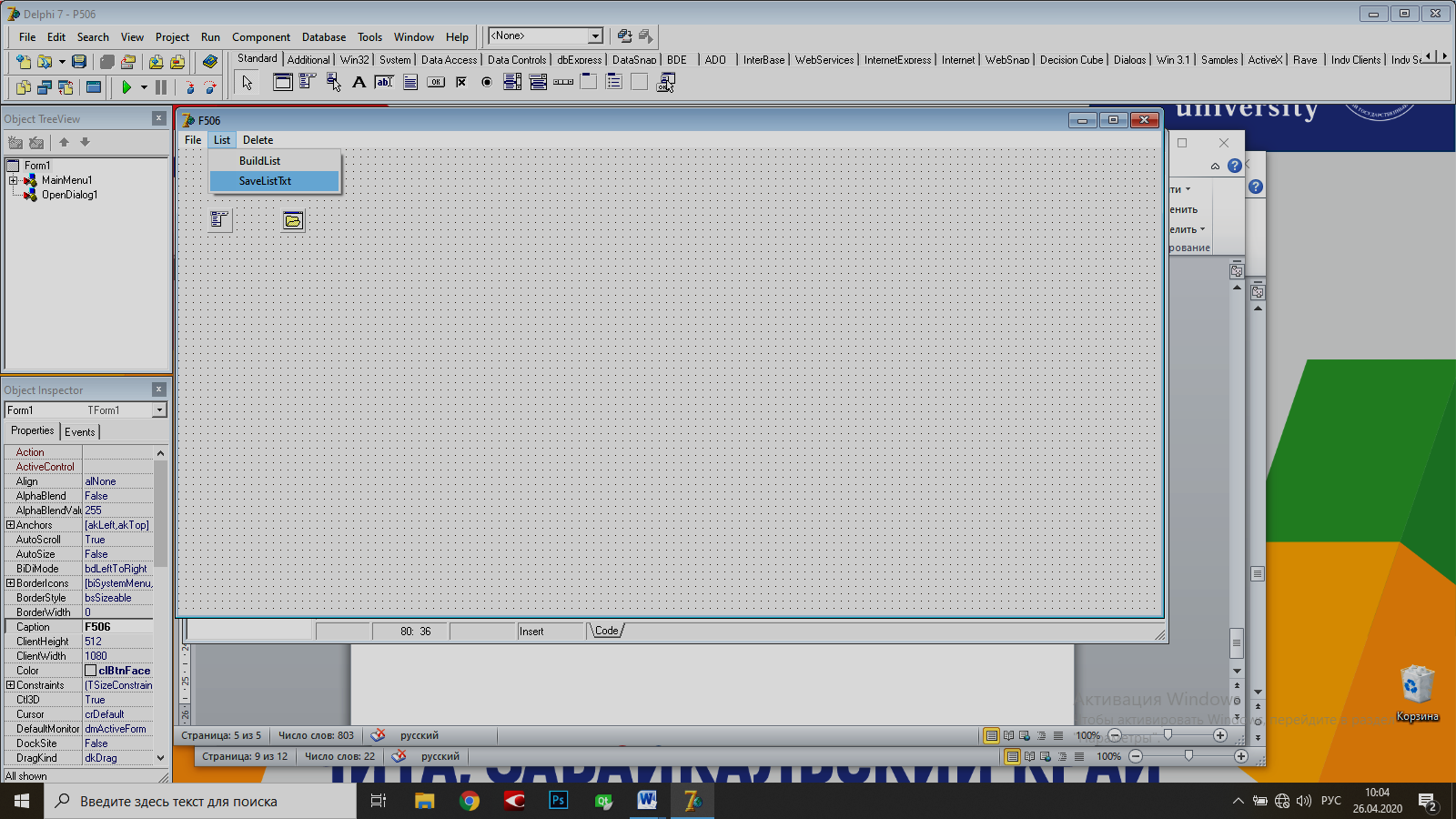
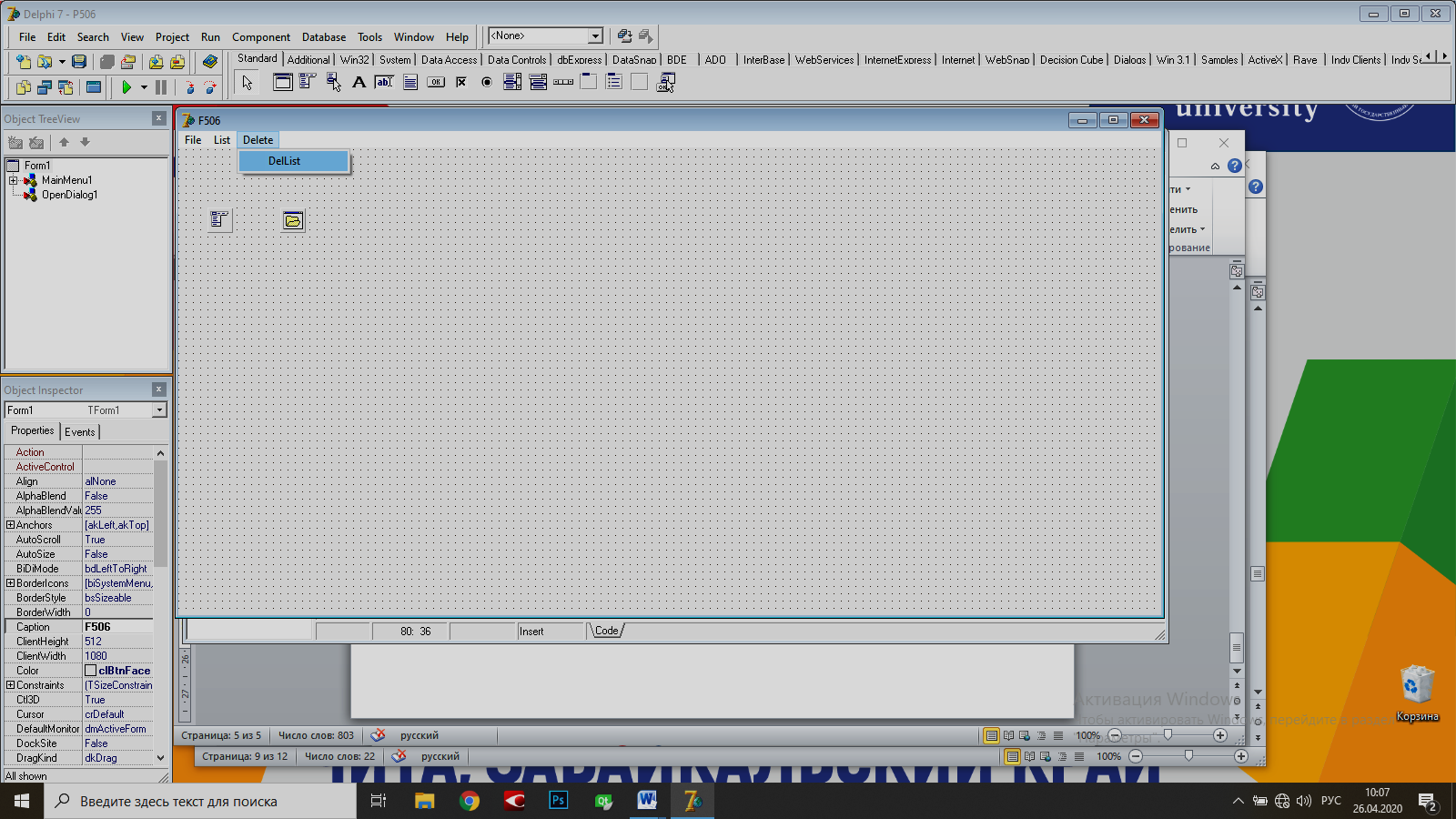
* Структурой узла – данные узла теперь хранятся в записи.
* Изменилась процедура инициализации данных узла, приходится вводить каждое поле. (Процедура **BuildSpisok**)
* Изменилась процедура вывода данных из узлов списка в текстовый файл, нужно выводить в определённом порядке каждое поле записи, хранящейся в узле. (Процедура **WriteSpText**)

Надеюсь понятно, как была заполнена заготовка создаваемого модуля. Копировался код соответствующего модуля предыдущей задачи, а затем в трёх местах он был изменён.

Для проверки работы созданного модуля переходим в окно формы **F506** и

оформим центр управления проектом. Переносим туда компоненты **MainMenu** и **OpenDialog,** которые находятся на страницах Standard и Dialog. Для проверки первой части задачи создадим меню:



В модуле формы появится код:

type

TForm1 = class(TForm)

MainMenu1: TMainMenu;

OpenDialog1: TOpenDialog;

File1: TMenuItem;

OpenTxtFile1: TMenuItem;

CloseTxtFile1: TMenuItem;

List1: TMenuItem;

BuildList1: TMenuItem;

Delete1: TMenuItem;

DelList1: TMenuItem;

SaveListTxt1: TMenuItem; //элемент меню раздела List1

Остаётся написать обработчики событий для созданных кнопок. Вначале в окне редактора модуля U506 в uses предложение добавим имя созданного

модуля Unit506:

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, Unit506, Menus;

Затем в секцию var добавим глобальные параметры:

var

Form1: TForm1;

PList: PUzel; //указатель на список

ftxt: text; //файловая переменная

Обработчики событий создаём также как в предыдущем проекте, т.е. в окне Form506 последовательно щёлкаем ЛКМ на каждом из элементов меню и в интерфейсной части модуля U506 получим заголовки обработчиков событий:

procedure OpenTxtFile1Click(Sender: TObject);

procedure CloseTxtFile1Click(Sender: TObject);

procedure BuildList1Click(Sender: TObject);

procedure DelList1Click(Sender: TObject);

procedure SaveListTxt1Click(Sender: TObject);

а в секции реализации заготовки процедур для написания кода обработчиков, которые после написания кода имеют вид:

procedure TForm1.OpenTxtFile1Click(Sender: TObject);

var

s: string;

begin

if not OpenDialog1.Execute then exit;

s := OpenDialog1.FileName;

AssignFile(ftxt,s);

Append(ftxt);

end;

procedure TForm1.CloseTxtFile1Click(Sender: TObject);

begin

CloseFile(ftxt);

end;

procedure TForm1.BuildList1Click(Sender: TObject);

begin

BuildSpisok(PList);

end;

procedure TForm1.DelList1Click(Sender: TObject);

begin

DelSpisok(PList);

end;

procedure TForm1.SaveListTxt1Click(Sender: TObject);

begin

writeln(ftxt,'Содержимое двусвязанного списка ');

WriteSpText(PList,ftxt);

writeln(ftxt,' ')

end;

end.

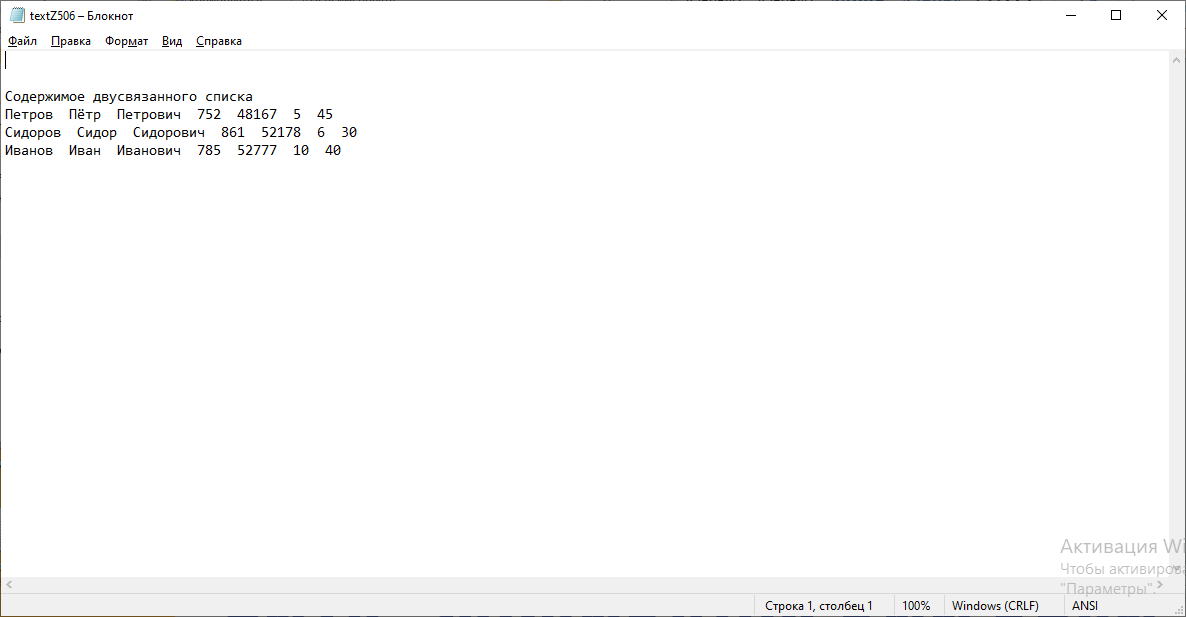
Запускаем приложение: Run|Run и выполняем следующую последовательность:

* Открываем текстовый файл. Он был создан в папке проекта.
* Создаём список, содержащий следующие сведения:

1. Петров Петр Петрович 752 48167 5 45
2. Сидоров Сидор Сидорович 861 52178 6 30
3. Иванов Иван Иванович 785 52777 10 40

Где последние цифры- вес багажа в кг., предпоследняя группа – число вещей в багаже, остальные цифры – серия и номер паспорта пассажира.

* Сохраняем список в текстовом файле.
* Закрываем текстовый файл, для данного примера это textZ506.
* Удаляем список.
* Закрываем приложение.
* Заходим в папку Z506 и открываем текстовый файл textZ506:



Т.е. список создаётся, сохраняется в текстовом файле, файл текстовый открывается и закрывается, список уничтожается. Первая часть задачи решена.

Для удобной работы с данным приложением нужно добавить ещё две процедуры:

* процедуру сохранения списка в типизированном файле,
* процедуру создания списка, используя записи из типизированного файла.

Это позволит многократно тестировать уже существующий типизированный файл. Эти процедуры должны быть созданы в модуле **Unit506**, а для проверки их работы в **MainMenu** нужно добавить элементы, обработчики которых будут:

* открывать типизированный файл,
* закрывать типизированный файл,
* сохранять список в типизированном файле.

В качестве подсказки – алгоритм процедуру сохранения списка в типизированном файле:

1. Процедура получает **по ссылке** указатель на голову списка и файловую переменную, связанную с типизированным файлом. **Файл должен быть открыт перед использованием процедуры**. Расширение файла **.dat** . открывать его командой **Reset.**
2. В процедуре используются один локальных параметра – текущий указатель, который пробегает значения от головы списка до хвоста(также как и в **WriteSpText**).
3. Присвоить текущему указателю значение указателя на голову списка.
4. Создать цикл пока текущий указатель не станет равным **nil,** а в теле цикла:

* записывать в типизированный файл значение поля **х** текущего указателя,
* переместить текущий указатель на следующий узел списка.

Процедура создания списка, используя данные из типизированного, подобна процедуре **BuildSpisok** . Отличие в том, что после выделения памяти под узел(**new(a);**) нужно считать данные из типизированного файла в поле **х** указателя **а** : (**read(ftip, a^.x)**;). А далее вставить узел **а** всоздаваемый список, всё как в процедуре **BuildSpisok** . Учтите, что **ftip** – это файловая переменная, связанная с типизированным файлом, который должен быть открыт перед использованием процедуры создания списка, точнее восстановления списка. **ftip –** глобальный параметр, который нужно описать в секции var модуля формы:

var

Form1: TForm1;

PList: PUzel;

ftxt : text;

ftip : FZap; // FZap – тип типизированного файла, определён выше