МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Забайкальский государственный университет»

(ФГБОУ ВО «ЗабГУ»)

Факультет: Энергетический

Кафедра: Информатики, вычислительной техники и прикладной математики

**КУРСОВАЯ РАБОТА**

По: Программированию

На тему: База данных продажи товаров в магазине

Выполнил студент группы: ИВТ-18-2, Цепляев Александр Федорович

Руководитель работы: Доцент, Соловьев Владимир Александрович

Чита

2019 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Забайкальский государственный университет»

(ФГБОУ ВО «ЗабГУ»)

Факультет: Энергетический

Кафедра: Информатики, вычислительной техники и прикладной математики

ЗАДАНИЕ

По дисциплине: программирование

Студенту: Цепляеву Александру Федоровичу

1. Тема курсовой работы: База данных продажи товаров в магазине

2. Срок подачи студентом законченной работы: 31 мая 2019 г.

3. Исходные данные к работе: Создать программу, в которой создается база данных, содержащая сведения о продаже товаров в магазине. Сведения включают: код товара, название товара, единицу измерения товара, стоимость, дата продажи, количество продаж, ФИО продавца. В конце дня подводится итог. Программа должна предоставлять возможность просматривать, добавлять, удалять, копировать, хранить данные. В программе использовать модули. Разработать простейший интерфейс в среде разработки Delphi 7.

4. Перечень подлежащих разработке в курсовой работе вопросов:

1. База данных для обработки списка
2. Графический интерфейс на Delphi
3. Алгоритмы работы со списком

Дата выдачи задания: 30 апреля 2019 г.

Руководитель курсовой работы\_\_\_\_\_\_\_\_\_\_\_\_\_ «Соловьев В.А.»

Задание принял к исполнению: «30» апреля 2019г.

Подпись студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_/Цепляев А.Ф.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Забайкальский государственный университет»

(ФГБОУ ВО «ЗабГУ»)

Факультет: Энергетический

Кафедра: Информатики, вычислительной техники и прикладной математики

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

По: Программированию

На тему: База данных продажи товаров в магазине

Выполнил студент группы: ИВТ-18-2, Цепляев Александр Федорович

Руководитель работы: Доцент, Соловьев Владимир Александрович

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc9941777)

[Глава 1 6](#_Toc9941778)

[1.1 Описание типов данных 6](#_Toc9941779)

[1.2 Действия по работе с базой данных 7](#_Toc9941780)

[Глава 2 8](#_Toc9941781)

[2.1 Описание методов для работы со списком 8](#_Toc9941782)

[Глава 3 13](#_Toc9941783)

[3.1 Создание интерфейса на Delphi 7 13](#_Toc9941784)

[Глава 4 18](#_Toc9941785)

[4.1 Руководство пользователя 18](#_Toc9941786)

[ЗАКЛЮЧЕНИЕ 23](#_Toc9941787)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 24](#_Toc9941788)

[Приложения 25](#_Toc9941789)

# ВВЕДЕНИЕ

В работе представлена база данных, содержащая сведения о продажах товаров в магазине. При ее создании применялись знания, полученные в течение двух семестров. В работе используются простые и составные типы данных: текстовые и типизированные файлы, записи, динамические списки (двусвязные), динамические переменные, процедуры и функции, модули, алгоритмы записи в файл, считывания с файла, перемещения по файлу. При работе с программой используются простейшие элементы интерфейса на базе среды разработки Delphi 7 для графического представления, а именно следующие компоненты: MainMenu, SaveDialog, OpenDialog, Memo.

# Глава 1

## 1.1 Описание типов данных

Работа будет осуществляться с двусвязным списком, состоящим из множества узлов.

Sp = record//тип узла

x: BD\_products;

next, pred: PUzel;

end;

1. 1 поле x – элемент списка, структура которого представляет собой тип записи;
2. 2 поле – указатель на следующий элемент списка (next);
3. 3 поле – указатель на предыдущий элемент списка (pred).

Элемент списка x – это структура записи, которая включает в себя поля, необходимые для заполнения информацией в соответствии с описанием предметной области:

BD\_products = record//запись данных о продажах товаров в магазине

code: integer; //код товара

nameT: string[50]; //наименование товара

price: real; //стоимость покупки

date: string[8];//дата продажи

sum: integer;//количество проданного товара

ed\_iz: string[20];//единица измерения товара

Fam,Name,Otch: string[20];//ФИО продавца

end;

Для работы со списком нужно создать тип указателя, которым будет описываться динамическая переменная. Эта переменная хранит адрес на некоторую ячейку памяти. Тип указателя создается раньше типа узла:

PUzel = ^Sp;//тип указателя

Данные будут храниться в типизированном файле, из которого они будут и восстанавливать в список. Структурой такого файла также является запись:

ft = file of BD\_products;

Сохраняться данные и результаты работы с ними будут в текстовом файле:

t: TextFile;

## 1.2 Действия по работе с базой данных

При работе с базой данных выполняются следующие действия:

1. Создание списка
2. Удаление списка
3. Создание очередного элемента списка
4. Вставка элемента:
   1. В начало списка
   2. В конец списка
   3. В любую позицию
5. Удаление элемента
6. Поиск элемента по ключу
7. Определение числа элементов в списке
8. Сохранение данных в типизированный файл
9. Сохранение данных в текстовый файл
10. Восстановление данных из типизированного файла
11. Расчёт объема продаж за указанный пользователем день и вывод отчета в текстовый файл

# Глава 2

## 2.1 Описание методов для работы со списком

При создании приложения использовались процедуры и функции – отдельные, независимые части от всей программы.

1) procedure BuildSpisok(var f: PUzel);

Процедура реализует создание двусвязного динамического списка вручную с нуля. Формальный параметр f – указатель на «голову» списка. При построении вызываются процедуры:

procedure AddFirst(var f, a: PUzel);//вставка узла в голову списка

procedure AddAfter(var old: PUzel; a: PUzel);//вставка узла после old

2) procedure DelSpisok(var f: PUzel);

Процедура реализует удаление списка, начиная с его «головы». В ней вызывается следующая процедура:

procedure DelFirstElement(var f, a: PUzel);//выделение элемента и передача указателя пользователю

3) procedure InsertInHead(var f: PUzel);//вставка элемента в начало списка

Выделяется память под переменную a, вводятся данные для нового элемента, и вызывается AddFirst. В качестве параметров передаются: f – указатель на начало, a – указатель на создаваемый узел.

4) procedure InsertAnyPos(var f: PUzel); //вставка элемента в любую позицию

Процедура вставляет создаваемый в ней элемент в любую позицию списка, в том числе и в конец. Для реализации этого метода в первую очередь необходимо определить количество элементов в списке, чтобы знать, в какую позицию вставлять. В методе вызывается следующая процедура:

procedure FindPosEl(var f, PosEl: PUzel);

Перемещаясь по списку, подсчитывается количество элементов. Затем пользователь вводит i-й элемент, после которого будет произведена вставка. Пользователю возвращается указатель на этот элемент.

Вернувшись в процедуру вставки, выделяется память под переменную a, вводятся данные для нового элемента, и вызывается метод AddAfter. В качестве параметров передается: PosEl – указатель на выделенный элемент, a – указатель на создаваемый узел.

5) procedure Delete1(var f: PUzel);//удаление любого элемента списка

При помощи процедуры можно удалять элемент списка, вводя код существующего товара. Если элемент существует – пользователь получит указатель на этот элемент.

Если выбранный элемент стоит первым в списке, тогда вызывается процедура AddFirstElement. Иначе вызывается следующая процедура:

procedure DelElement(var a: PUzel);

Если элемент стоит последним, тогда нужно разорвать связь только с предыдущим элементом. В противном случае разорвать связи и с предыдущим и со следующим элементами.

6) procedure write\_in\_tip\_file(var f: PUzel; var tipf: ft);//запись в типизированный файл

7) procedure write\_in\_text(var t: text; var f: PUzel);//запись в текстовый файл

8) procedure Search\_by\_name(var f:PUzel);//поиск товара по названию

9) procedure Search\_by\_code(var f: PUzel);//поиск товара по коду

8 и 9 процедуры реализуют поиск по ключу. Они полностью идентичны, за исключением типа ключа. В первом случае это строковый тип, во 2 – целый.

Их работа заключается в следующем: пользователь вводит ключ, следуя подсказке, затем запускается цикл, в котором сравниваются поле в узле списка и введенный ключ. Если есть совпадение, то выводится на экран и в текстовый файл информация о товаре. Если совпадений нет, то выводится сообщение, что товар по указанному ключу не найден.

Код для данного метода:

{поиск товара по коду}

**procedure** Search\_by\_code(**var** f: PUzel);

**var** key: integer; p: PUzel; s: string; d: boolean;

t: text;

**begin**

AssignFile(t, 'Данные о товарах2.txt');

Rewrite(t);

**if** (f = nil) **then**

ShowMessage('Список пуст!')

**else**

**begin**

p:=f;

d:=false;

key := StrToIntDef(InputBox('Введите код товара:','',''),0);

**while** (p <> nil) **do**

**begin**

**if** (p^.x.code = key) **then**

**begin**

d := true;

**with** p^.x **do**

**begin**

s:='Данные о товаре:

…

//выводятся все поля записи

…

ShowMessage(s);

writeln(t,s);

writeln(t);

**end**;

**end**;

p:=p^.next;

**end**;

**if** (**not** d) **and** (key <> 0) **then**

ShowMessage('Товар с кодом '+IntToStr(key)+' не найден.');

**end**;

CloseFile(t);

**end**;

10) procedure VolOfSale(var f: PUzel);//расчёт объема продаж за день

Как указано в исходных данных к работе, в программе должна присутствовать возможность в конце дня подводить итог, то есть рассчитывать объем продаж. Объем продаж – это сумма денежных средств, поступивших на счет компании за реализованные товары и услуги за определенный период времени.

В методе пользователь вводит дату, за которую хочет получить отчет. При перемещении по списку ищутся совпадающие данные: день, месяц, год. Если товары с такой датой продажи имеются, то пользователю возвращается true. Затем происходит подсчёт объема продаж отдельного товара: для этого количество проданного товара умножается на его цену, и в сумматор добавляется этот объем продаж. После чего выводится в текстовый файл следующая информация: название товара, выручка за него.

Когда цикл заканчивается, в текстовый файл выводится общая выручка за все товары, полученная путем суммирования выручек от отдельных товаров.

Если же введенная пользователем дата оказывается не совпадающей с датами из списка, то в отчете появляется сообщение о том, что за указанный период не было продаж.

Код для данного метода:

{расчет объема продаж за день}

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

**var**

p: PUzel; d:string[8]; t: text;

ras, s: real; st: string; key: boolean;

**begin**

AssignFile(t,'Отчет по объему продаж.txt');

Rewrite(t);

key := false;

p := f;

s := 0;

d := InputBox('Введите дату:','ДД.ММ.ГГ','');

st := 'Отчет за '+d;

writeln(t, st);

**while** (p <> nil) **do**

**begin**

**with** p^.x **do**

**begin**

**if** (d = date) **then**

**begin**

key := true;

ras := price \* sum;//расчет выручки за отдельный товар

s := s + ras;//общая сумма всех продаж

st := 'Товар: '+nameT+', выручка: '+FloatToStr(ras)+' р.';

writeln(t, st);

**end**;

**end**;

p := p^.next;

**end**;

**if** key **then**

writeln(t, 'Общая выручка: ',s:0:2,' р.') **else**

writeln(t, 'За указанный день не было продаж');

CloseFile(t);

**end**;

11) procedure RTF(var f:PUzel;var tipf:ft);//восстановление данных из типизированного файла в список

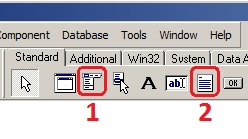
12) procedure Change\_product\_name(var f: PUzel);//изменение полей записи

Процедура предоставляет возможность менять значение полей узла списка. Используется оператор case of, с помощью которого пользователь выбирает поле для ввода нового значения.

# Глава 3

## 3.1 Создание интерфейса на Delphi 7

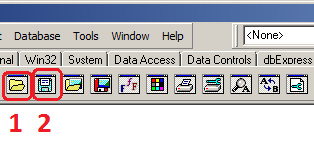
Для создания простейшего интерфейса использовались следующие компоненты: MainMenu, Memo, расположенные на вкладке Standard, и SaveDialog, OpenDialog – на вкладке Dialogs.



1 – MainMenu

2 - Memo

Рисунок 3.1 – Компоненты MainMenu, Memo



1 – OpenDialog

2 – SaveDialog

Рисунок 3.2 – Компоненты диалоговых окон

Для помещения соответствующего компонента на форму, необходимо дважды кликнуть по его иконке.

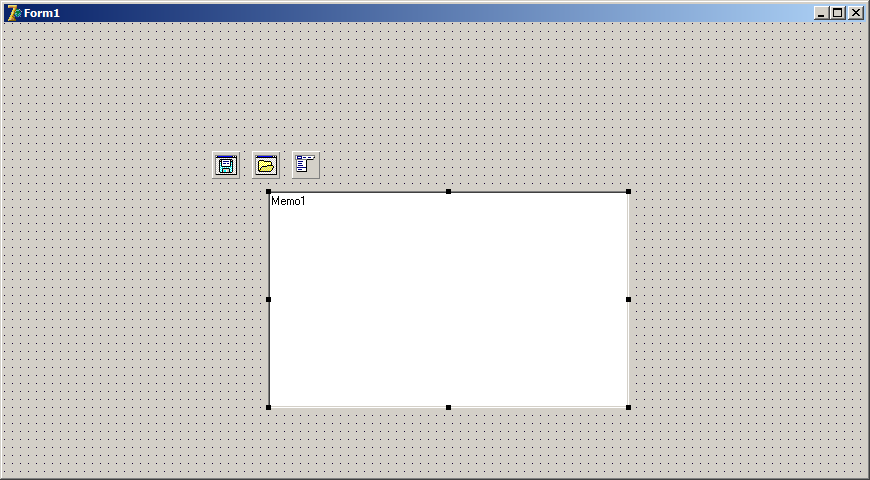


Рисунок 3.3 – Компоненты на форме

Теперь можно приступать к созданию меню. Два раза кликнув по иконке MainMenu, откроется окно редактирования.

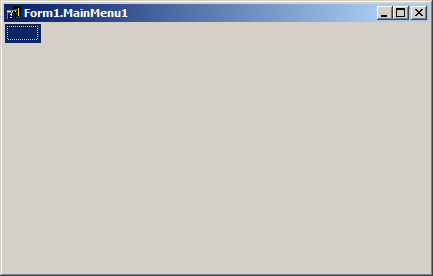


Рисунок 3.4 – Окно редактирования кнопок меню

Теперь нужно перенести взгляд на Инспектор объектов слева, в котором нужно будет менять во вкладке Properties заголовки меню и подменю (свойство Caption).

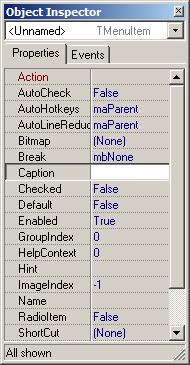


Рисунок 3.5 – Инспектор объектов, вкладка Properties (свойства)

После изменения первого раздела меню, можно добавить к нему соответствующие подменю. Новые поля заголовков генерируются автоматически. Например, для меню File подменю будет Save to Text File.

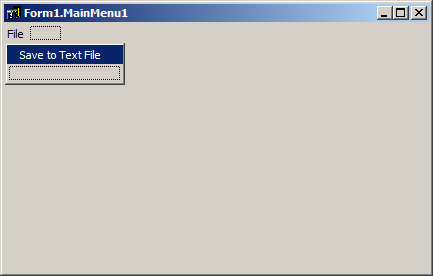


Рисунок 3.6 – Добавление меню и подменю

Таким образом, конструируем интерфейс, добавляя кнопки, которые будут выполнять основные действия по работе с базой данных.

Также к подменю можно добавлять дополнительные кнопки справа - Sub Menu.

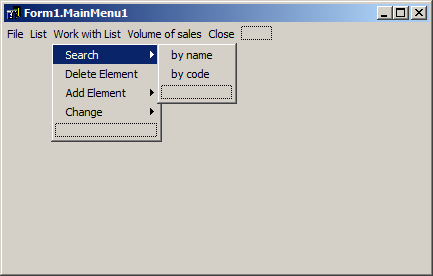


Рисунок 3.7 – Sub menu

Компонент Memo – это простой текстовый редактор, который можно сравнить с блокнотом Windows, так как в него можно добавлять любой текст и редактировать аналогичным образом. Нужен он для вывода базы данных на экран. У данного компонента меняется размер: длина и ширина.

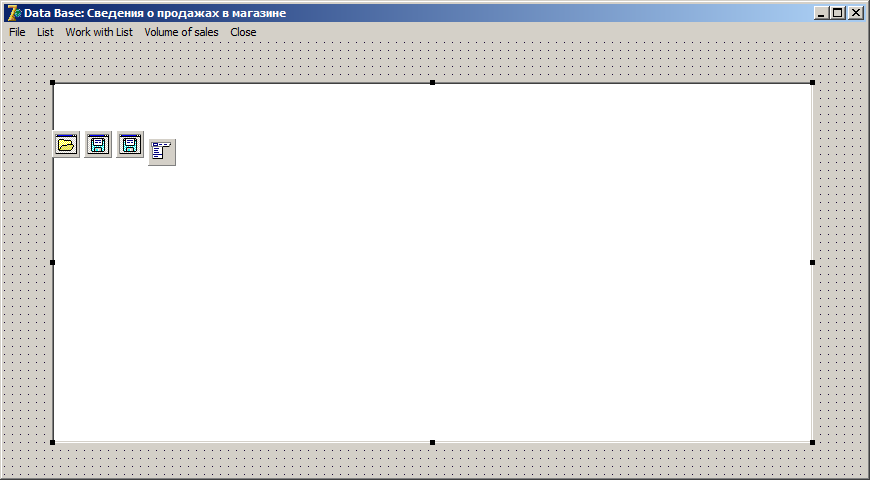


Рисунок 3.8 – Memo поле

Компоненты SaveDialog, OpenDialog – простые диалоговые окна Windows, которые будут осуществлять сохранение файла и открытие.

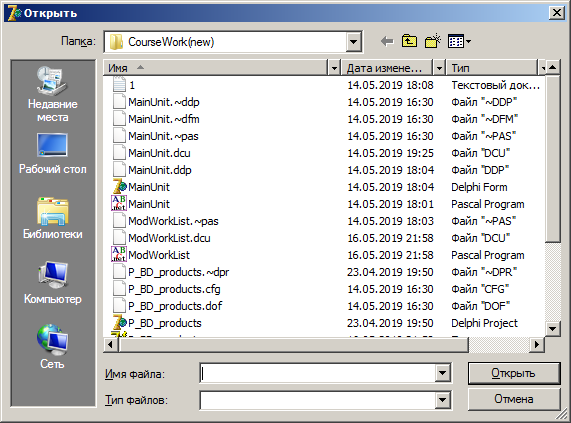


Рисунок 3.9 – Компонент OpenDialog

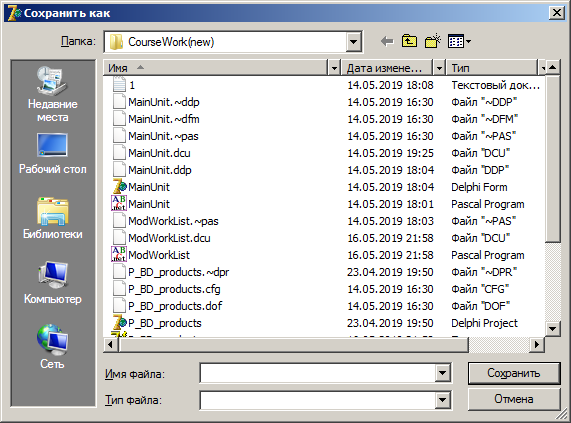


Рисунок 3.10 – Компонент SaveDialog

# Глава 4

## 4.1 Руководство пользователя

Чтобы открыть приложение, нужно найти файл с именем P\_BD\_products.exe в каталоге, где хранятся модули и программные файлы Delphi.

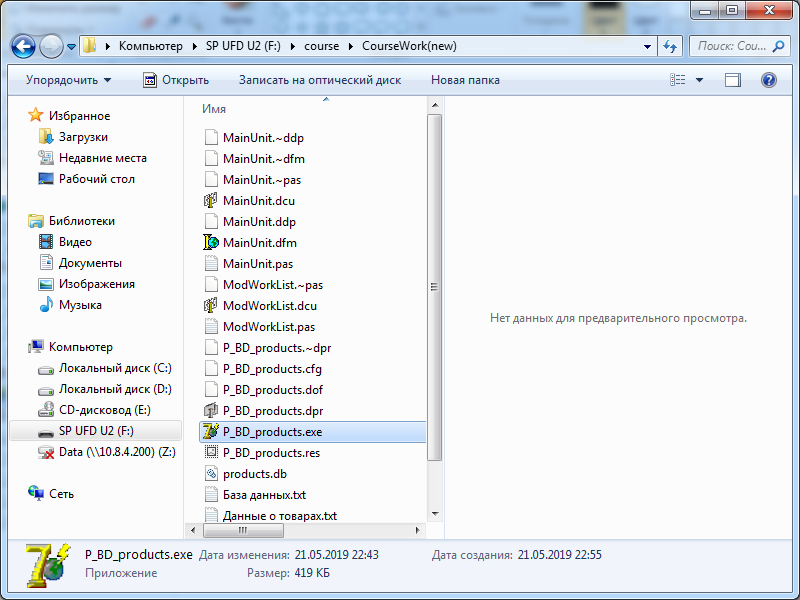


Рисунок 4.1 – Приложение P\_BD\_products.exe

Главное окно представлено на рисунке 4.2.

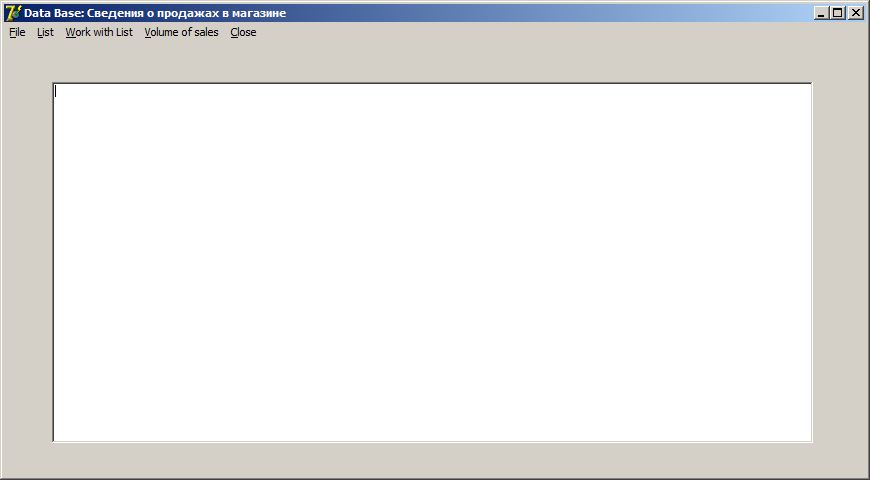


Рисунок 4.2 – Главное окно

За выполнение работы отвечают обработчики события. Они обрабатывают клик по пунктам подменю, после чего вызываются соответствующие процедуры.

Итак, для восстановления базы данных из заранее созданного типизированного файла, нужно зайти в пункт меню List и кликнуть по RestoreFromipFile. Откроется диалоговое окно, где будут отображаться файлы формата db.

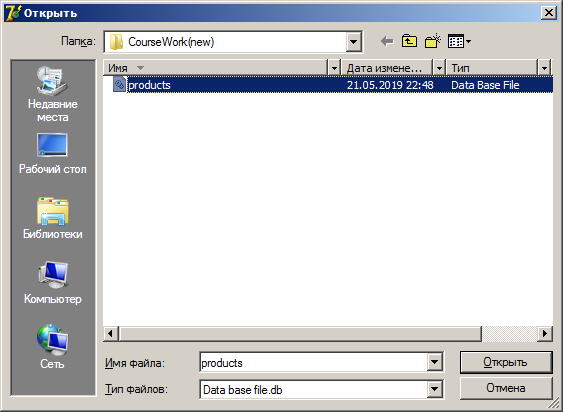


Рисунок 4.3 – Восстановление базы данных

После открытия файла products.db восстановится список, который можно вывести в memo поле (List --> Output on screen).

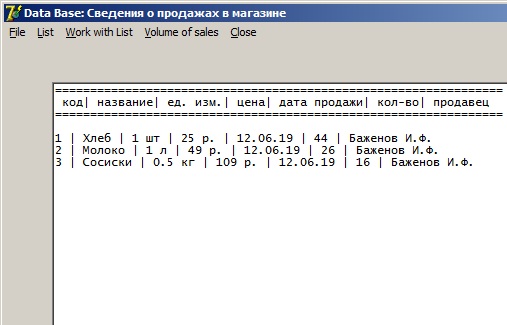


Рисунок 4.4 – Вывод в Memo

Для примера работы можно рассчитать объем продаж (Volume of sales --> To calculate), где у пользователя потребуют ввести дату.

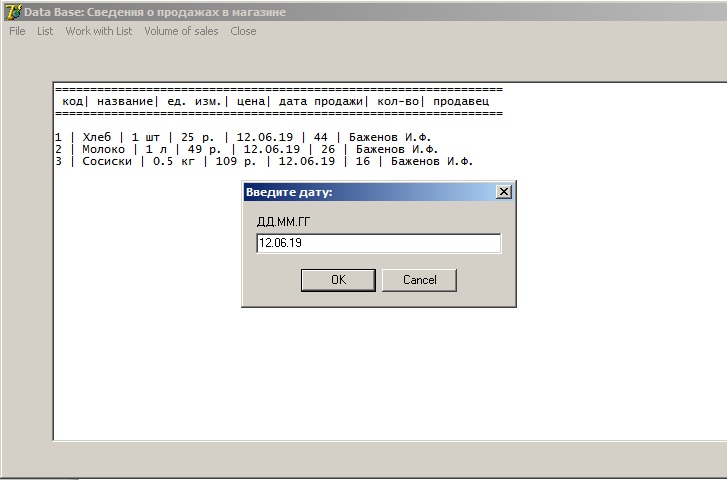


Рисунок 4.5 – Выполнение расчета объема продаж

После нажатия на ОК, будет выведен отчет в текстовый файл, который сохранится в каталоге, где находится приложение (см. рисунки 4.5 и 4.6).

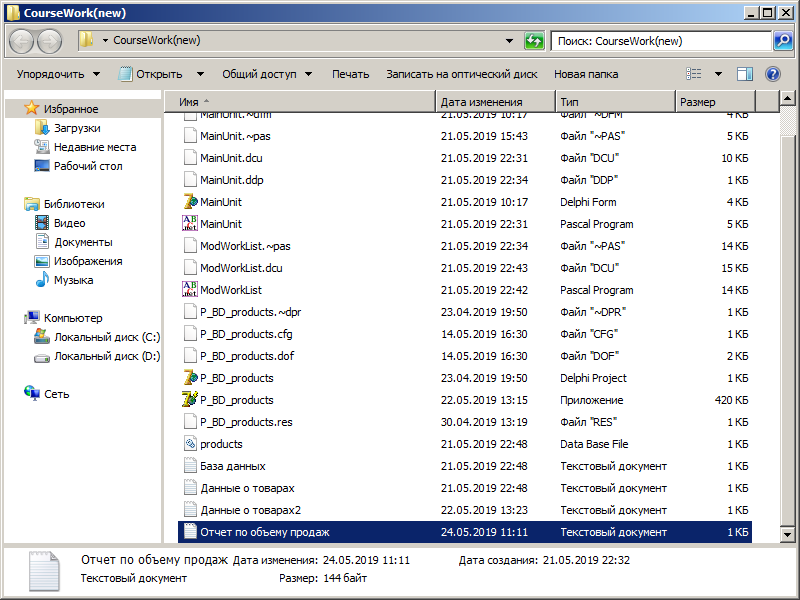


Рисунок 4.6 – Файл с отчетом в каталоге

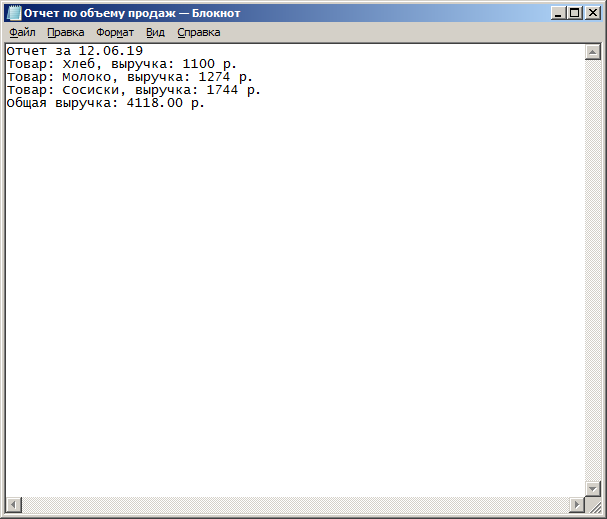


Рисунок 4.7 – Отчет по объему продаж

Убедившись, что программа работает, можно удалить список (List --> Delete List) и закрыть ее (Close). При повторном вызове процедуры удаления выведется на экран следующее сообщение (см. рисунок 4.7), что обережет пользователя от ошибки.

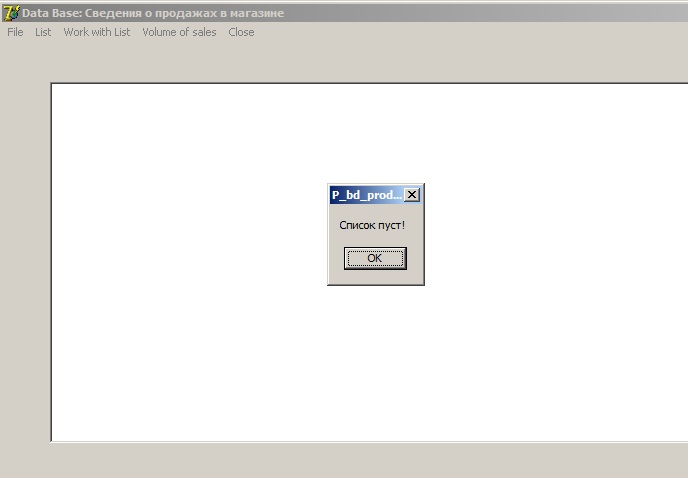


Рисунок 4.8 – Сообщение, предостерегающее от ошибок

# ЗАКЛЮЧЕНИЕ

Была создана база данных для магазина. В ней реализованы основные процедуры для работы со списком, такие как: создание списка, удаление списка, добавление нового элемента, удаление элемента и т.д. В том числе были созданы дополнительные процедуры: операция расчета объема продаж и вывода отчета в файл, операция изменения полей записи, восстановления списка из типизированного файла, поиск по ключу. Операция сортировки не была реализована по причине нехватки времени. Программа тестировалась в течение двух недель, сбоев и ошибок в работе программы не было обнаружено. Также был создан простейший интерфейс, предоставляющий работу пользователю с базой данных. Он состоит из следующих компонентов: MainMenu, SaveDialog, OpenDialog, Memo. MainMenu используется для создания меню приложения. SaveDialog, OpenDialog были изучены самостоятельно, и используются для работы с файловой системой. Компонент Memo был также изучен самостоятельно, и используется в программе для просмотра базы данных на экране приложения. Для того чтобы уберечь пользователя от возможных ошибок, было предусмотрено появление предупреждающих окон, о чем говорится в пункте 4.1.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Фаронов В.В. TurboPascal 7.0, начальный курс: учебное пособие. – М: КноРус, 2005. - 576 с.
2. Уроки Delphi начинающим с нуля [Электронный ресурс] – Режим доступа: <http://www.delphi-manual.ru/>
3. Малыхина, М. П. Программирование на языке высокого уровня Turbo Pascal / М.П. Малыхина. - М.: БХВ-Петербург, 2012. - 544 c.
4. Delphi для начинающих [Электронный ресурс] – Режим доступа: <http://www.cyberforum.ru/delphi-beginners/>
5. Delphi – Форум программистов [Электронный ресурс] – Режим доступа: <http://www.programmersforum.ru/forumdisplay.php?f=1>

# Приложения

**Приложение А**

Интерфейсная часть модуля ModWorkList

**unit** ModWorkList;

**Interface**

**uses** SysUtils, Dialogs;

**type**

BD\_products = **record**//запись данных о продажах товаров в магазине

code: integer; //код товара

nameT: string[50]; //наименование товара

price: real; //стоимость покупки

date: string[8];//дата продажи

sum: integer;//количество проданного товара

ed\_iz: string[20];//единица измерения товара

Fam,Name,Otch: string[20];//ФИО продавца

**end**;

PUzel = ^Sp;//тип указателя

Sp = **record**//тип узла

x: BD\_products;

next, pred: PUzel;

**end**;

ft = **file of** BD\_products;

**procedure** AddFirst(**var** f, a: PUzel);//вставка узла в голову списка

**procedure** AddAfter(**var** old: PUzel; a: PUzel);//вставка узла после old

**procedure** BuildSpisok(**var** f: PUzel);//создание списка с нуля

**procedure** DelFirstElement(**var** f, a: PUzel);//выделение элемента, с начала

**procedure** DelSpisok(**var** f: PUzel);//удаление списка, начиная с головы

**procedure** write\_in\_text(**var** t: text; **var** f: PUzel);//запись в текстовый файл

**procedure** write\_in\_tip\_file(**var** f: PUzel; **var** tipf: ft);//запись в

//типизированный файл

**procedure** RTF(**var** f:PUzel;**var** tipf:ft);//восстановление списка из

//типизированного файла

**procedure** FindPosEl(**var** f, PosEl: PUzel);//определение позиции элемента

**procedure** DelElement(**var** a:PUzel);//выделение элемента

**procedure** Search\_by\_name(**var** f:PUzel);//поиск товара по названию

**procedure** Search\_by\_code(**var** f: PUzel);//поиск товара по коду

**procedure** InsertInHead(**var** f: PUzel);//вставка элемента в начало списка

**procedure** InsertAnyPos(**var** f: PUzel);//вставка элемента в любую позицию

**procedure** VolOfSale(**var** f: PUzel);//рассчет объема продаж за день

**procedure** Delete1(**var** f: PUzel);//удаление любого элемента списка

**procedure** Change\_Element(**var** f: PUzel);//изменение полей записи

**Приложение Б**

Интерфейсная часть модуля формы

**unit** MainUnit;

**Interface**

**uses**

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

Dialogs, Menus, ModWorkList, StdCtrls;

**type**

TForm1 = **class**(TForm)

MainMenu1: TMainMenu;

File1: TMenuItem;

List1: TMenuItem;

BuildList1: TMenuItem;

DeleteList1: TMenuItem;

WorkList1: TMenuItem;

Search1: TMenuItem;

byname1: TMenuItem;

bycode1: TMenuItem;

Close1: TMenuItem;

DisposeEl1: TMenuItem;

SaveDialog1: TSaveDialog;

SaveDialog2: TSaveDialog;

OpenDialog1: TOpenDialog;

Volumeofsales1: TMenuItem;

tocalculate1: TMenuItem;

InsertElement1: TMenuItem;

SavetoTextFile1: TMenuItem;

SavetoTipFile1: TMenuItem;

RestoreFromTipFile1: TMenuItem;

Insertinhead1: TMenuItem;

Insertinanyposition1: TMenuItem;

Memo1: TMemo;

Outputonscreen1: TMenuItem;

ChangeElement1: TMenuItem;

**procedure** BuildList1Click(Sender: TObject);

**procedure** DeleteList1Click(Sender: TObject);

**procedure** Close1Click(Sender: TObject);

**procedure** DisposeEl1Click(Sender: TObject);

**procedure** byname1Click(Sender: TObject);

**procedure** SavetoTextFile1Click(Sender: TObject);

**procedure** SavetoTipFile1Click(Sender: TObject);

**procedure** RestoreFromTipFile1Click(Sender: TObject);

**procedure** tocalculate1Click(Sender: TObject);

**procedure** bycode1Click(Sender: TObject);

**procedure** Insertinanyposition1Click(Sender: TObject);

**procedure** Insertinhead1Click(Sender: TObject);

**procedure** Outputonscreen1Click(Sender: TObject);

**procedure** ChangeElement1Click(Sender: TObject);

**private**

{ Private declarations }

**public**

{ Public declarations }

**end**;

**var**

Form1: TForm1; head: PUzel;

t: TextFile; tipf: ft;