

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Забайкальский государственный университет»  
(ФГБОУ ВО «ЗабГУ»)

Факультет: Энергетический факультет

Кафедра: Информатики и вычислительной техники

## **Курсовая работа**

по Программированию

На тему: База данных продажи товаров в магазине

Выполнил ст. гр. ИВТ-19-2

Раменский А.А.

Проверил Доцент кафедры ИВТ и ПМ

Соловьёв В.А.

Чита

2020

## **Задание для Курсовой работы**

Создать программу, в которой создаётся база данных, содержащая сведения о продаже товаров в магазине, сведения включают: наименование товара, объём продаж, стоимость товара, Ф.И.О. продавца. В конце дня подводится итог.

Программа должна предоставить возможность просматривать, добавлять, удалять, копировать, хранить данные.

В программе использовать модули, функции, процедуры, записи, списки и файлы

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«Забайкальский государственный университет»  
(ФГБОУ ВО «ЗабГУ»)

Факультет: Энергетический факультет

Кафедра: Информатики и вычислительной техники

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе (проекту)

по Программированию

на тему: База данных продажи товаров в магазине

Выполнил студент группы ИВТ-19-2 Раменский А.А.

Руководитель работы: Доцент кафедры ИВТ и ПМ Соловьёв В.А.

# Пояснительная записка

**Целью курсовой работы** является разработка приложения, с минимальным интерфейсом, которое будет работать с базой данных для продажи товаров в магазинах. Также целью будет являться: Применение типов данных, изученных во время обучения, Использование процедур и функций, модулей и файлов, а также создание минимального интерфейса

## Календарный план

№	Наименование раздела курсовой работы	Неделя					
		1	2	3	4	5	6
1	Введение	+					
2	Глава 1		+				
3	Глава 2			+			
4	Глава 3				+		
5	Заключение					+	
6	Литература, приложение						+

# Оглавление

Введение .....	4
Глава 1. Типы данных и операции, реализуемые в курсовой работе. ....	6
1.1. Типы данных, используемые в курсовой работе.....	6
Секция type: .....	6
1.2. Операции, реализуемые в курсовой работе.....	7
Список операций для данной курсовой работы: .....	8
Операции в созданном модуле: .....	9
Примеры использования некоторых операций:.....	10
Глава 2. Интерфейс приложения .....	11
2.1. Компоненты интерфейса, используемых в курсовой работе. Их описание в программе. ....	12
2.2. Реализация обработчиков событий. ....	15
Файл – Сохранить:.....	16
Файл – Открыть: .....	17
Список – Построить: .....	18
Список – Итог дня: .....	18
Список – Удалить: .....	19
Список – Просмотреть: .....	19
Список – Сортировка: .....	21
Глава 3. Тестирование созданного приложения, проверка полученных результатов .....	24
Заключение .....	28

Литература .....	29
Приложение .....	30
Модуль формы: .....	30
Созданный модуль: .....	32

# Введение

**Актуальность данной темы** заключается в том, что в данный момент все торговые предприятия переходят на новый уровень – уровень технологий. То есть, они ставят электронные оборудования на кассах. Также, множество предприятий имеют достаточно много магазинов, супермаркетов, гипермаркетов, в которых работает как минимум по 4 продавца. Довольно тяжело следить за продажами без помощи программ и компьютера.

Поэтому в супермаркетах уже стоят новейшие кассы, которые регистрируют кассиров и при продаже товаров этим кассиром, всё записывается в базу данных. Без таких технологий обрабатывать это всё становится очень затруднительно. Но что делать малым предприятиям, которые не могут себе позволить данные кассовые аппараты. Для них то и следует написать несложную программу, работающую с базой данных, используя минимальный интерфейс.

**База данных** - представленная в объективной форме совокупность самостоятельных материалов, систематизированных таким образом, чтобы материалы могли быть найдены и обработаны с помощью Электронно-вычислительной машины (ЭВМ). [1] В этой работе использовалась простейшая база данных, состоящая из одной таблицы, которая позволяла хранить небольшое кол-во информации.

Для студентов же, то есть нас, это является хорошей практикой для нашей работы в будущем. Уже при работе над данной темой мы можем научиться создавать свою базу данных, а также проработать всё, что касается типов данных, динамических линейных списков, а также работе с Delphi и созданию примитивного интерфейса, понятного каждому человеку, который будет работать с этой программой. В общем, использовать на практике всё то, что было изучено на лекционных занятиях, при самостоятельном изучении во



время дистанционного обучения, а также из дополнительных источников, предоставленных нашим преподавателем.

В данном приложении должны быть введены такие функции как:

1. Хранение информации.
2. Создание базы данных.
3. Внесение изменений в базу данных.
4. Удаление информации из базы данных.
5. Сортировка данных.
6. Подведение итога дня.

**Объектом исследования** данной курсовой работы являются работа различных магазинов и отработка на практике знаний о базе данных и линейных списках, полученных на занятиях. **Предметом исследования** является двунаправленный линейный список, база данных и динамические типы данных в целом

Для выполнения этой курсовой работы были использованы такие источники информации как:

Электронные ресурсы:

[http://www.pascal.helpov.net/index/dynamic\\_lists\\_pascal\\_](http://www.pascal.helpov.net/index/dynamic_lists_pascal_)[https://ru.wikipedia.org/wiki/База\\_данных](https://ru.wikipedia.org/wiki/База_данных), <https://studfile.net/preview/5357642/page:14/>Книги:  
«Программирование в Delphi» Архангельский А.Я., «Delphi 7 справочное пособие» Архангельский А.Я.

# **Глава 1. Типы данных и операции, реализуемые в курсовой работе**

## **1.1. Типы данных, используемые в курсовой работе[4],[5]**

Данная работа подразумевает использование списка записей, которые можно хранить в файле на компьютере. Поэтому самым первым мы создадим тип записи, содержащий в себе основные элементы базы данных, для выполнения задачи курсовой работы: ФИО продавца, название товара, объём продаж и цена товара – вот какие поля будут в данной записи.

Чтобы реализовать список необходимо снова создать тип записи, но эта запись будет содержать в себе указатель на следующий узел, на предыдущий узел, а также на запись, описанную выше.

Следующим типом данных будет тип указатель на запись, реализующую список.

И последним типом данных, который мы опишем в секции `type`, будет тип файла, способный хранить в себе базу данных.

### **Секция `type`:**

`Type`

`Tovar = record //Тип записи для нашей задачи, который  
будет храниться в базе данных`

`NameTovar:string[30]; //название товара`

```

ObProdaj:integer; //Объём продаж данного товара

CenaTovara:integer; //Цена товара

FIOProdavca:string[30]; //ФИО продавца

end;

PUzel = ^Zl; //Указатель на тип данных, создающий список

Zl = record //Запись, реализующая список

    x:Tovar; //Информация, хранящаяся в узлах списка

    next:puzel; //Указатель на следующий узел

    pred:puzel; //Указатель на предыдущий узел

end;

Fzap = file of Tovar; //Типизированный файл, для хранения базы
данных

```

## 1.2. Операции, реализуемые в курсовой работе

Для этой работы необходимо разработать несколько операций, которые необходимы для работы с базой данных: Сохранение списка в файл, загрузка списка из файла, построение, дополнение, просмотр, сортировка и удаление списка, а также подведение итогов дня для магазина. Они будут описаны в создаваемом модуле и использоваться в модуле формы, при реализации этих операций в приложение.

## Список операций для данной курсовой работы:

- *Сохранение списка в файл* необходимо для того, чтобы сохранить базу данных на хранителе данных, чтобы в дальнейшем его можно было открыть и использовать. Для такого мы будем использовать типизированный файл. Для сохранения результатов будем использовать текстовый файл, так как его можно без проблем прочитать в любом блокноте.
- *Загрузка списка из файла*, нужна как раз таки, чтобы открыть файл, сохранённый ранее, и взять оттуда список для дальнейшего использования.
- *Построение списка* нужно для того, чтобы построить список и начать делать свою базу данных.
- *Дополнение списка* будет использоваться, чтобы ввести новые данные в уже существующий список.
- *Просмотр списка* нужен, чтобы просмотреть содержимое списка.
- *Сортировка* нужна, чтобы сделать список упорядоченным и облегчить поиск по базе данных необходимых элементов. Сортировка будет проводиться по 4 признакам: ФИО продавца, название товара, объём продажи и цены товара – сортировка в алфавитном порядке, либо проводится по возрастанию.
- *Удаление списка* будет удалять базу данных, если она неактуальна.
- *Подведение итогов дня в магазине* – это заключительная операция в данной работе, необходима для того, чтобы подсчитать общее кол-во проданных товаров и общую выручку за день.

### Операции в созданном модуле:

procedure BuildSpisok(var f: PUzel);//Процедура, для построения списка

procedure AddFirst(var f: PUzel; a: PUzel);//Вставить узел а первым в список

procedure AddAfter(var old:PUzel; a: PUzel);//Вставить узел а после old

procedure WriteSpTip(var f: PUzel; var ftip:Fzap);//Записать данные списка в типизированный файл

procedure WriteSpText(var f: PUzel; var ftxt:Text); //Записать в текстовый файл

procedure BuildSpisokFromTip(var f:puzel;var ftip:Fzap);  
//Построить список из данных, взятых из тип. файла

procedure DelFirstElement(var f,a: PUzel);//Выделить первый элемент списка

procedure DelElement(var old,a: PUzel);//Выделить элемент из списка, следующий за old

procedure ItogDay(var f:puzel;var ftxt:text);//Процедура для выполнения задачи

procedure DobVSp(var f:puzel);//Добавляет в список новые элементы, не удаляя старые

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

procedure PoslElem(f:puzel;var a:puzel);//Выбирает последний элемент в списке

```
procedure Sort(f:puzel;var b:puzel);//Сортирует данный список
```

### **Примеры использования некоторых операций:**

#### **Итог дня:**

Допустим, у нас имеется список состоящий из:

*Морковь в количестве: 10 по цене: 15 был продан продавцом: Сидоров С.С.*

*Картошка в количестве: 11 по цене: 40 был продан продавцом: Петров П.И.*

*Лопата в количестве: 5 по цене: 50 был продан продавцом: Сидоров С.С.*

Тогда, результатом данной операции будет:

*Итог дня:*

*Продавец Сидоров С.С. Продал товар: Морковь В объёме: 10 шт. На сумму: 150 руб.*

*Продавец Петров П.И. Продал товар: Картошка В объёме: 11 шт. На сумму: 440 руб.*

*Продавец Сидоров С.С. Продал товар: Лопата В объёме: 5 шт. На сумму: 250 руб.*

*Итого: 840 руб.*

## Сортировка:

Имеется база данных, состоящая из нескольких элементов:

	ФИО продавца	Название товара	Объём продаж	Цена товара	Итого
1	Раменский А.А.	Морковь	15 шт.	40 руб.	600 руб.
2	Говорун Н.А.	Шпатель	5 шт.	60 руб.	300 руб.
3	Раменский Н.А.	Огурец	15 шт.	60 руб.	900 руб.
4	Говорун Н.А.	Лак	5 шт.	40 руб.	200 руб.
				Итого:	2000 руб.

Тогда при использовании сортировки, к примеру, по ФИО, мы получим вот такой упорядоченный список:

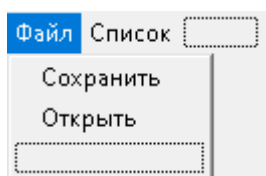
	ФИО продавца	Название товара	Объём продаж	Цена товара	Итого
1	Говорун Н.А.	Морковь	15 шт.	40 руб.	600 руб.
2	Говорун Н.А.	Шпатель	5 шт.	60 руб.	300 руб.
3	Раменский А.А.	Огурец	15 шт.	60 руб.	900 руб.
4	Раменский Н.А.	Лак	5 шт.	40 руб.	200 руб.
				Итого:	2000 руб.

## Глава 2. Интерфейс приложения

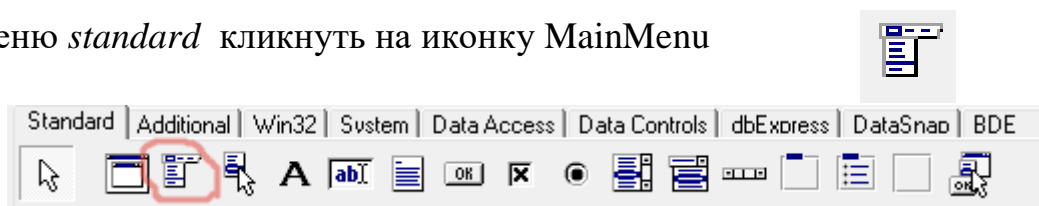
Интерфейс программы для этой курсовой должен быть максимально простым и эффективным, чтобы его понял любой пользователь, но одновременно он должен выполнять множество различных операций и функций для работы с базой данных. В этом нам помогут компоненты интерфейса Delphi: TMainMenu, TStringGreed.

## 2.1. Компоненты интерфейса, используемых в курсовой работе. Их описание в программе.

Для того, чтобы не загромождать всю форму кнопками, надписями, картинками и т.д., мы будем использовать удобный компонент Delphi – MainMenu. Он представляет из себя небольшую полоску наверху формы программы, которая содержит в себе разные меню. Меню в свою очередь состоят из подменю, таким образом, представляя структурированную область для создания множества обработчиков событий, не занимающих много места.

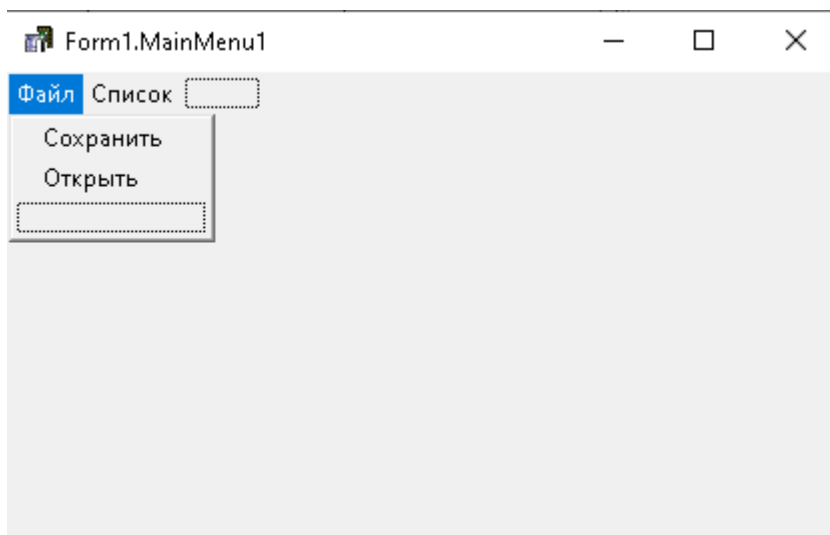


Чтобы его добавить на форму, нужно в меню компонентов в подменю *standard* кликнуть на иконку MainMenu



И разместить на форме в любом месте. После нужно сделать 2-ой клик по значку MainMenu на форме, и тогда откроется редактор меню, в котором мы можем добавлять подменю:

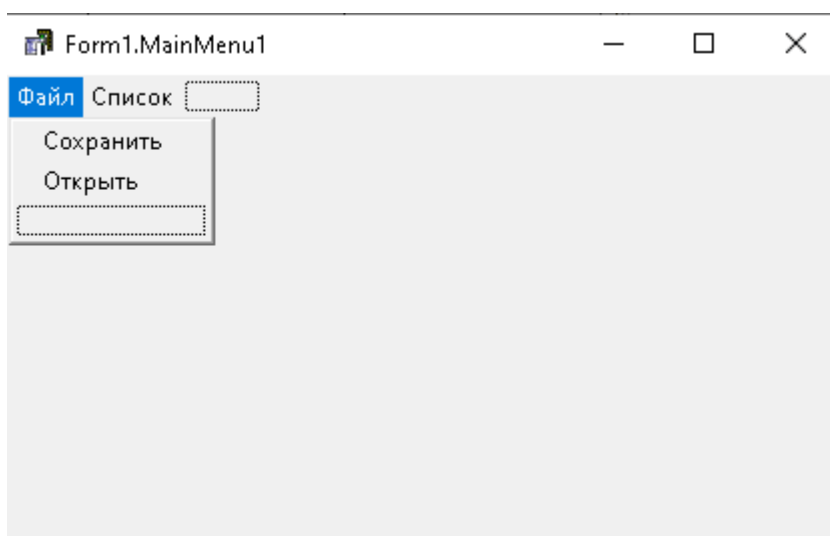




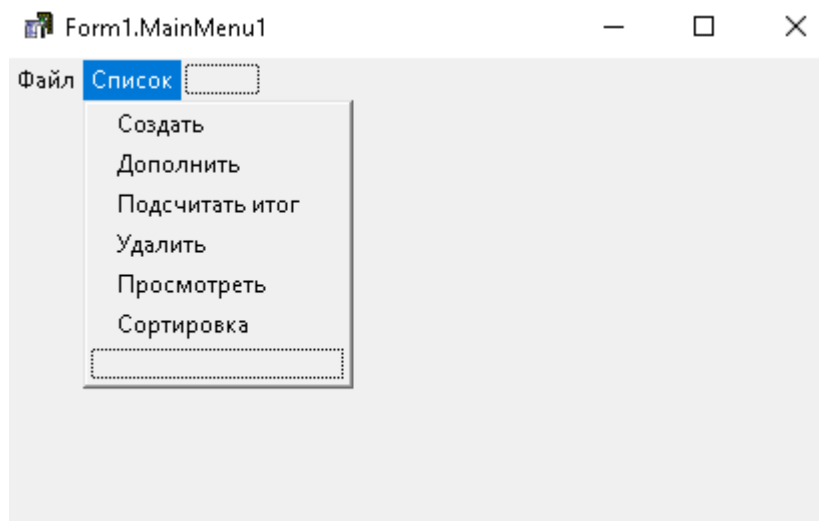
После, добавленные подменю появятся на основной форме и дереве компонентов программы, где мы можем при двойном клике на них, создавать обработчик событий, либо через редактор обработчиков событий.

В нашей программе будет 2 меню: *Файл* и *Список* – они будут содержать в себе основные функции, нужные для работы с базой данных в данной программе.

В меню *Файл* будут находится подменю: *Сохранить* и *Открыть* – они будут содержать код, позволяющий сохранять и открывать список при помощи файлов.



В меню *Список* будут находиться такие подменю как: *Создать*, *Дополнить*, *Подсчитать итог*, *Удалить*, *Просмотреть*, *Сортировка*. Они будут содержать код, позволяющий работать со списком, являющимся нашей базой данных, то есть создать, дополнить, удалить, просмотреть или сортировать список, а также подводить итог дня.



Второй элемент, который будет помогать создавать минимальный, но функциональный интерфейс – таблица *StringGreed*. Это обыкновенная таблица, в которую мы будем выводить базу данных. Заголовками в таблице будут: ФИО продавца, название товара, объём продаж, цена товара и общая сумма продаж этого товара. В самом конце будет подводиться итог.

Таблица, как она выглядит в программе:

	ФИО продавца	Название товара	Объём продаж	Цена товара	Итого
1	Раменский А.А.	Морковь	15 шт.	40 руб.	600 руб.
2	Говорун Н.А.	Шпатель	5 шт.	60 руб.	300 руб.
3	Раменский Н.А.	Огурец	15 шт.	60 руб.	900 руб.
4	Говорун Н.А.	Лак	5 шт.	40 руб.	200 руб.
				Итого:	2000 руб.

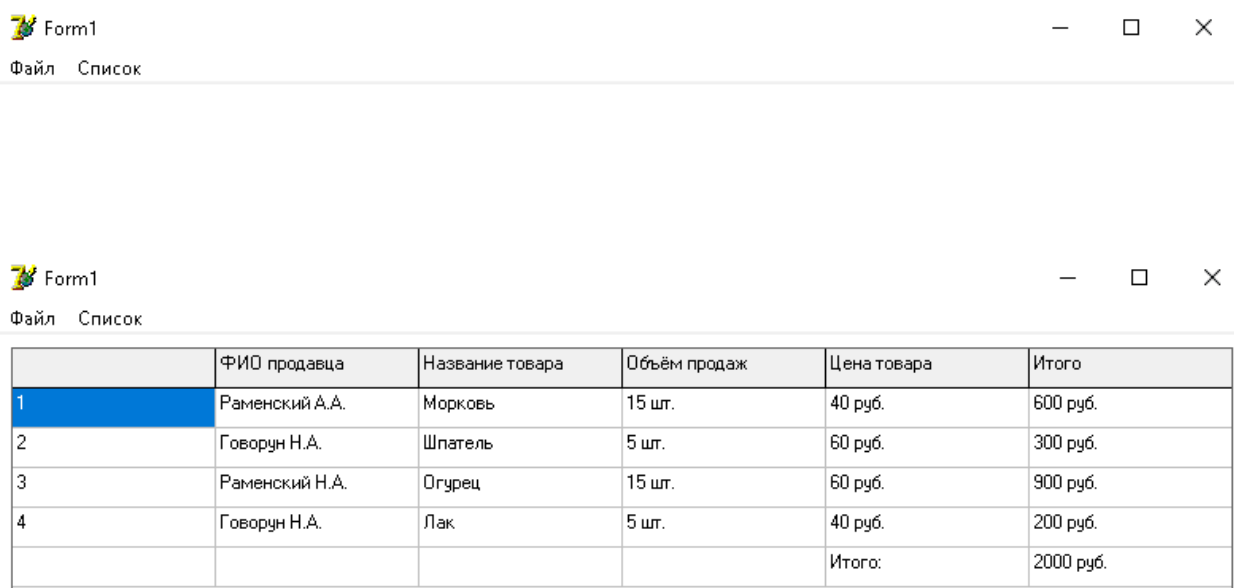
Чтобы добавить таблицу на форму, необходимо в меню компоненты в подменю *Additional* найти значок *StringGreed*





И добавить в любое место на форме. После с таблицей можно начать работать, у неё есть параметр Cells[Номер столбца, Номер строки], который позволяет заполнять таблицу, нумерация столбцов и строк начинается от нуля. Чтобы заполнить таблицу в программе, необходимо заполнить список, либо загрузить его из файла, и нажать на *Список - Просмотреть*. Тогда таблица выведет все данные из данного списка.

При запуске приложение выглядит очень просто, белый фон и меню, таблица в начале скрыта, ведь она пустая и смысла в ней на экране нет, она появляется когда пользователь решит просмотреть список.



Интерфейс программы не содержит лишних кнопок, в которых можно запутаться, не содержит лишних надписей, а также выполняет все необходимые функции

## 2.2. Реализация обработчиков событий. [1],[3],[4],[5]

В данной главе будут представлены коды для обработчиков событий, описанных выше.

### **Файл – Сохранить:**

```
procedure TForm1.SaveToTxt1Click(Sender: TObject);
//Сохраняет список в текстовый и типизированный файл

var

    s,s1:string;//Имена файлов, в которые будут произведены
сохранения

begin

    if (savedialogTxt.execute) then begin

        s:=SaveDialogTxt.FileName; //Присваивание переменной s
название текстового файла

        AssignFile(ftxt,s);

        Append(ftxt);

        WriteSpText(Sp,ftxt); //Сохранение списка в текстовый
файл

        closefile(ftxt);

    end

    else exit;

    if savedialogTip.execute then begin

        s1:=SaveDialogTip.FileName; //Присваивание переменной s1
название типизированного файла
```

```

AssignFile(ftip,s1);

rewrite(ftip);

WriteSpTip(Sp,ftip); //Сохранение списка в типизированный
файл

closefile(ftip);

end

else exit;

end;

```

#### **Файл – Открыть:**

procedure TForm1.Open1Click(Sender: TObject); //Открывает  
типизированный файл и загружает оттуда список, сохранённый  
ранее

```

var

s:string; //Имя файла, который будет открыт

begin

if not OpenFileDialog1.Execute then exit;

s:=OpenDialog1.FileName; // Присваивание переменной s
название текстового файла

Assignfile(ftip,s);

reset(ftip);

BuildSpisokFromTip(Sp,ftip); //Построение списка из файла

```

```
closefile(ftip);
```

```
end;
```

### **Список – Построить:**

```
procedure TForm1.Build1Click(Sender: TObject); //Создаёт  
новый список
```

```
begin
```

```
    BuildSpisok(Sp); //Процедура построения списка
```

```
end;
```

### *Список – Дополнить:*

```
procedure TForm1.DobavitVSpisok1Click(Sender: TObject);  
//Добавляет в список новые элементы
```

```
begin
```

```
    DobVSp(Sp); //Процедура добавления новых элементов в  
список
```

```
end;
```

### **Список – Итог дня:**

```
procedure TForm1.Itog1Click(Sender: TObject); //Рассчитывает  
итог дня и записывает его в текстовый файл
```

```

var

    s:string; // Переменная для названия файла

begin

    if not OpenFileDialog2.Execute then exit;

    s:=OpenFileDialog2.FileName; // Присваивание переменной s
название файла

    assignfile(ftxt,s);

    append(ftxt);

    ItogDay(Sp,ftxt); //Процедура подведения итога дня

    closeFile(ftxt);

end;

```

#### **Список – Удалить:**

```

procedure TForm1.Delete1Click(Sender: TObject); // Удаляет
список

begin

    DelSpisok(Sp); //Процедура удаления списка

end;

```

#### **Список – Просмотреть:**

```

procedure TForm1.Look1Click(Sender: TObject);
//Выводит список в таблицу, давая возможность просмотреть
список

var

    a,b:puzel; // Узлы необходимые для данной процедуры

    sum,sum1,k:integer; // sum – Итого для товара, sum1 -
итог дня, k – счётчик для кол-ва элементов в списке

begin

    a:=sp;

    Pos1Elem(a,b); //Находит последний элемент

    sum1:=b^.x.ObProdaj*b^.x.CenaTovara; // Находим
Итого для последнего элемента, так как программа в цикле
почему-то не берёт в расчёт последний элемент

    a:=sp;

    k:=1;

    tabl.Visible:=true; // Делаем таблицу видимой

    tabl.cells[1,0]:='ФИО продавца'; //Заполнение первой
строки таблицы (+ то что ниже)

    tabl.cells[2,0]:='Название товара';

    tabl.cells[3,0]:='Объём продаж';

    tabl.cells[4,0]:='Цена товара';

    tabl.cells[5,0]:='Итого';

```



While not(a=nil) do begin //Цикл для заполнения  
остальной таблицы

```
    sum:=a^.x.ObProdaj*a^.x.CenaTovara;  
  
    Tabl.cells[0,k]:=IntToStr(k);  
  
    Tabl.cells[1,k]:=a^.x.FIOProdavca;  
  
    Tabl.cells[2,k]:=a^.x.NameTovar;  
  
    Tabl.cells[3,k]:=IntToStr(a^.x.ObProdaj)+' шт.';  
  
    Tabl.cells[4,k]:=IntToStr(a^.x.CenaTovara)+' руб.';  
  
    Tabl.cells[5,k]:=IntToStr(sum)+' руб.';  
  
    sum1:=sum1+sum;  
  
    inc(k);  
  
    Tabl.RowCount:=k+1;  
  
    Tabl.Height:=Tabl.RowCount*26;  
  
    Form1.Height:=Tabl.height+(3*26);  
  
    Tabl.cells[4,k]:='Итого:';  
  
    Tabl.cells[5,k+1]:=IntToStr(sum1)+ ' руб.';  
  
    a:=a^.next;  
  
end;  
  
end;
```

**Список – Сортировка:**

```

procedure TForm1.Poisk1Click(Sender: TObject); //Сортирует
список

var

  a:puzel;

  sum,k:integer; // sum – Итого для товара, k – счётчик для
кол-ва элементов в списке

begin

  SpP:=nil; //SpP – сортированный список

  a:=Sp;

  sort(a,SpP);

  k:=1;

  tabl.visible:=true; // Если вдруг пользователь сначала нажмёт
на сортировку, вместо показать, то сделает таблицу видимой

  While not(a=nil) do begin //Цикл для внесения в таблицу
сортированного списка

    sum:=a^.x.ObProdaj*a^.x.CenaTovara;

    Tabl.cells[0,k]:=IntToStr(k);

    Tabl.cells[1,k]:=a^.x.FIOProdavca;

    Tabl.cells[2,k]:=a^.x.NameTovar;

    Tabl.cells[3,k]:=IntToStr(a^.x.ObProdaj)+' шт.';

    Tabl.cells[4,k]:=IntToStr(a^.x.CenaTovara)+' руб.';

    Tabl.cells[5,k]:=IntToStr(sum)+' руб.';

```

```
inc(k);

Tabl.RowCount:=k+1;

Tabl.Height:=Tabl.RowCount*26;

Form1.Height:=Tabl.height+(3*26);

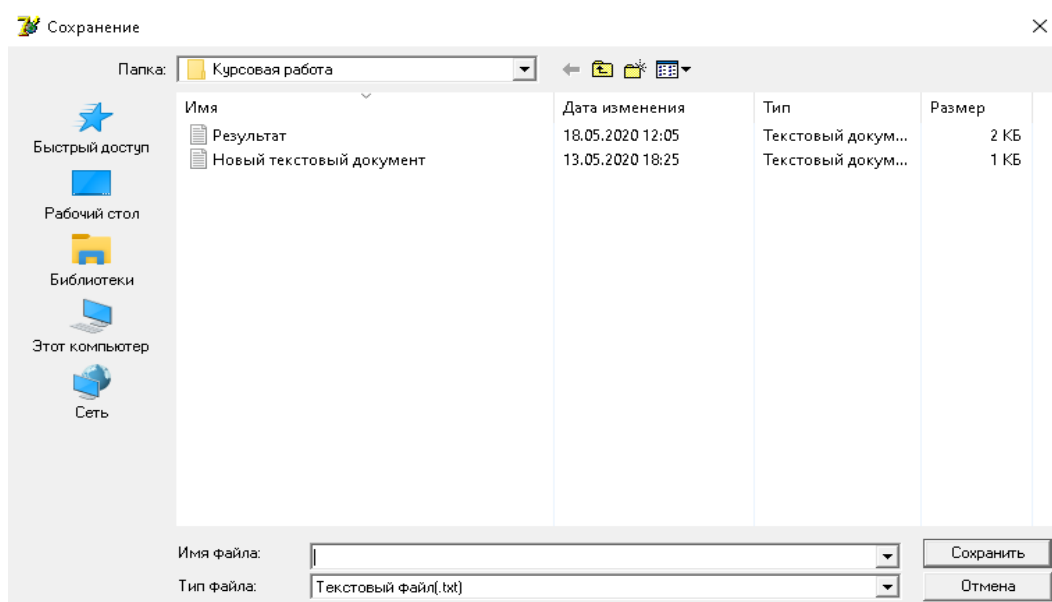
a:=a^.next;

end;

end;
```

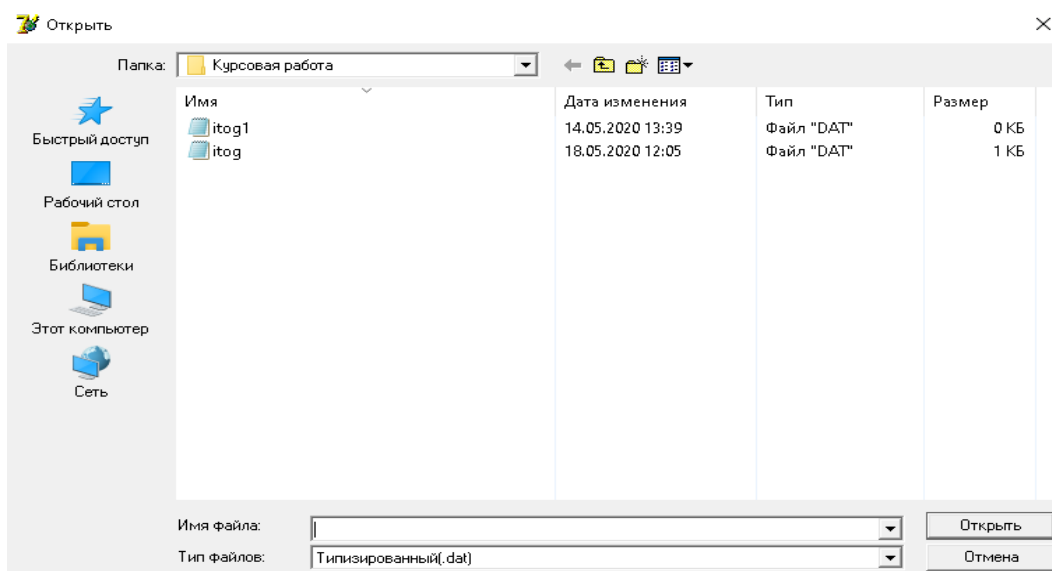
## Глава 3. Тестирование созданного приложения, проверка полученных результатов

При клике на *Файл – Сохранить* будет появляться окно,

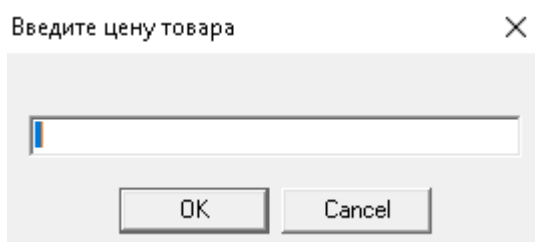
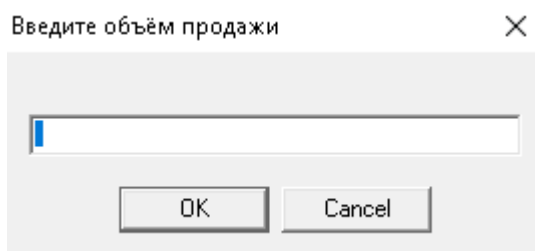
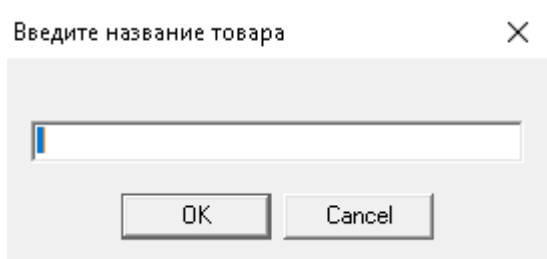


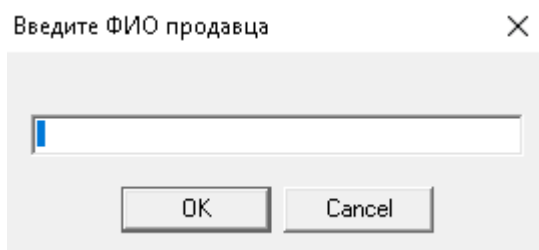
В котором мы можем выбрать или создать файл, в который мы хотим сохранить наш список

При клике на *Файл – Открыть* появится похожее окно, но с одним лишь отличием, тут можно будет только выбрать файл, который хотим открыть.

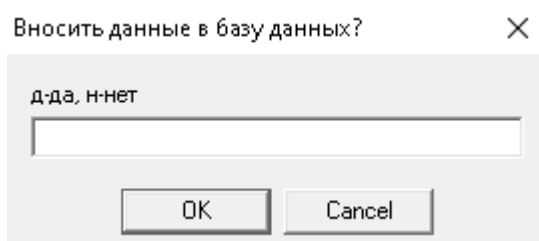


При нажатии на *Список – Построить*, *Список - Дополнить* программа будет поочерёдно спрашивать вас о том, что вы хотите ввести, но при нажатии на *Дополнить* список не будет заполняться сначала, он продолжит тот список, который был уже создан:

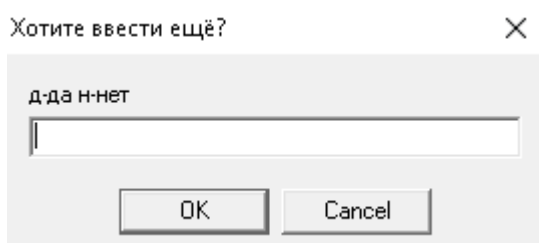




После заполнения данных, программа спросит о том, вносить ли данные в базу данных, где, если вы неправильно заполнили данные, можно отменить их внос в базу данных и заполнить заново.



После вноса данных в базу данных, программа спросит вас о дальнейшем вводе. Если вы согласитесь, программа вернёт вас в начало ввода, и вы начнёте заполнять новые данные.



При нажатии на *Список - Итог дня* программа подсчитывает общую сумму, и записывает результат в текстовый файл в виде

Итог дня:  
Продавец Сидоров С.С. Продал товар: Морковь В объёме: 10 шт. На сумму: 150 руб.  
Продавец Петров П.И. Продал товар: Картошка В объёме: 11 шт. На сумму: 440 руб.  
Продавец Сидоров С.С. Продал товар: Лопата В объёме: 5 шт. На сумму: 250 руб.  
Итого: 840 руб.

При нажатии на *Список – Удалить* программа удаляет данный список.

При нажатии на *Список – Просмотреть* программа выводит список в таблицу.

При нажатии на *Список – Сортировка* программа сортирует данные по возрастанию и алфавиту:

Имеется база данных, состоящая из нескольких элементов:

	ФИО продавца	Название товара	Объём продаж	Цена товара	Итого
1	Раменский А.А.	Морковь	15 шт.	40 руб.	600 руб.
2	Говорун Н.А.	Шпатель	5 шт.	60 руб.	300 руб.
3	Раменский Н.А.	Огурец	15 шт.	60 руб.	900 руб.
4	Говорун Н.А.	Лак	5 шт.	40 руб.	200 руб.
				Итого:	2000 руб.

Тогда при использовании сортировки, к примеру, по ФИО, мы получим вот такой упорядоченный список:

	ФИО продавца	Название товара	Объём продаж	Цена товара	Итого
1	Говорун Н.А.	Морковь	15 шт.	40 руб.	600 руб.
2	Говорун Н.А.	Шпатель	5 шт.	60 руб.	300 руб.
3	Раменский А.А.	Огурец	15 шт.	60 руб.	900 руб.
4	Раменский Н.А.	Лак	5 шт.	40 руб.	200 руб.
				Итого:	2000 руб.

## Заключение

При написании данной курсовой работы было использовано шесть источников:

Чтобы выполнить практическое задание курсовой работы была изучена теория для работы с такими компонентами Delphi как: MainMenu, StringGreed – они также и были включены в работу самой программы. Для работы приложения были созданы такие операции как:

- Сохранение списка
- Загрузка списка из типизированного файла
- Создание списка
- Дополнение списка
- Удаление списка
- Просмотр списка
- Сортировка списка
- Подведение итога дня

Данные операции работают исправно, из возможных ошибок: Ошибка при вводе пустого значения во время построения или дополнения списка. Решением является закрытие окна ошибки, и ввод данных сначала. Программа тестировалась множество раз – около 50 пробных запусков.

Руководство пользователя подробно описано с использованием скриншотов, наглядно показано как работает каждый обработчик событий.



## Литература

1. Pascal-helpov Программирование. Динамические списки Паскаль [Электронный ресурс] / Режим доступа:  
[http://www.pascal.helpov.net/index/dynamic\\_lists\\_pascal\\_programming](http://www.pascal.helpov.net/index/dynamic_lists_pascal_programming)
2. Wikipedia База данных [Электронный ресурс] / Режим доступа:  
[https://ru.wikipedia.org/wiki/База\\_данных](https://ru.wikipedia.org/wiki/База_данных)
3. StudFiles Динамические списки [Электронный ресурс] / Режим доступа:  
<https://studfile.net/preview/5357642/page:14/>
4. Архангельский А.Я. Программирование в Delphi: учебник - М.: – ООО «Бином – Пресс», 2003. – 1152 с.
5. Архангельский А.Я. Delphi 7. Справочное пособие: учебник – М.: Бином, 2004. – 1024 с.
6. Оформление курсовой работы [Электронный ресурс] / Общие требования к построению и оформлению текстовой документации ЗабГУ. – Режим доступа:  
[http://zabgu.ru/files/html\\_document/pdf\\_files/fixed/Normativny'e\\_dokumenty'/MI\\_\\_01-02-2018\\_Obshhie\\_trebovaniya\\_k\\_postroeniyu\\_i\\_oformleniyu\\_uchebnoj\\_tekstovoj\\_dokumentacii.pdf](http://zabgu.ru/files/html_document/pdf_files/fixed/Normativny'e_dokumenty'/MI__01-02-2018_Obshhie_trebovaniya_k_postroeniyu_i_oformleniyu_uchebnoj_tekstovoj_dokumentacii.pdf)

# Приложение

## Модуль формы:

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics,  
Controls, Forms, Dialogs, Menus, UnitmodulK, Grids;

type

TForm1 = class(TForm)

MainMenu1: TMainMenu;

File1: TMenuItem;

SaveToTxt1: TMenuItem;

Open1: TMenuItem;

Spisok1: TMenuItem;

Build1: TMenuItem;

Itog1: TMenuItem;

Delete1: TMenuItem;

Look1: TMenuItem;

SaveDialogTxt: TSaveDialog;

SaveDialogTip: TSaveDialog;

OpenDialog1: TOpenDialog;

```

DobavitVSpisok1: TMenuItem;

OpenDialog2: TOpenDialog;

Tab1: TStringGrid;

Poisk1: TMenuItem;

procedure SaveToTxt1Click(Sender: TObject); // Сохранение в
текстовый файл

procedure Open1Click(Sender: TObject); // Открытие из
типизированного файла

procedure Build1Click(Sender: TObject); // Построение списка

procedure Itog1Click(Sender: TObject); // Подведение итога

procedure Delete1Click(Sender: TObject); // Удаление списка

procedure DobavitVSpisok1Click(Sender:
TObject); // Добавление в список новых элементов

procedure Look1Click(Sender: TObject); // Просмотр списка

procedure Poisk1Click(Sender: TObject); // Сортировка списка

private
    { Private declarations }
public
    { Public declarations }
end;

var

    Form1: TForm1;
    ftxt: text;
    ftip: fzap;

```

```
Sp,SpP:puzel;
```

## **Созданный модуль:**

```
uses
```

```
    SysUtils, Dialogs;
```

```
Type
```

```
    Tovar = record //Тип записи для нашей задачи
```

```
        NameTovar:string[30]; //Название товара
```

```
        ObProdaj:integer; //Объём продаж данного товара
```

```
        CenaTovara:integer; //Цена товара
```

```
        FIOProdavca:string[30]; //ФИО продавца
```

```
    end;
```

```
    PUzel = ^Zl;
```

```
    Zl = record
```

```
        x:Tovar;
```

```
        next:puzel;
```

```
        pred:puzel;
```

```
    end;
```

```
    Fzap = file of Tovar; //Файловый тип для хранение базы данных
```

procedure BuildSpisok(var f: PUzel);//Процедура, для построения списка

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

procedure WriteSpTip(var f: PUzel; var ftip:Fzap);//Записать данные списка в типизированный файл

procedure WriteSpText(var f: PUzel; var ftxt:Text); //Записать в текстовый файл

procedure BuildSpisokFromTip(var f:puzel;var ftip:Fzap);  
//Построить список из данных, взятых из тип. файла

procedure DelFirstElement(var f,a: PUzel);//Выделить первый элемент списка

procedure DelElement(var old,a: PUzel);//Выделить элемент из списка, следующий за old

procedure ItogDay(var f:puzel;var ftxt:text);//Процедура для выполнения задачи

procedure DobVSp(var f:puzel);//Добавляет в список новые элементы, не удаляя старые

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

procedure PoslElem(f:puzel;var a:puzel);//Выбирает последний элемент в списке

procedure Sort(f:puzel;var b:puzel);//Сортирует данный список