Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ОТЧЕТ**

**ПО ДОМАШНЕЙ КОНТРОЛЬНОЙ РАБОТЕ №9**

**«Работа с типизированными файлами»**

**ПО «МДК 05.02 Разработка кода информационных систем»**

Выполнил: студент учебной группы

ИСПк-203-52-00

Широков Михаил Александрович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2025

**1. Цель работы**

Получить представление об организации хранения и использования информации посредством типизированных файлов, закрепить навыки создания пользовательского интерфейса.

**2. Формулировка задания**

**1.**  Сформировать структуру записи, с которой будет осуществляться работа, и согласовать ее с преподавателем;

**2.** Разработать приложение, позволяющее извлекать набор записей из типизированного файла, визуализировать данный набор, а также позволяющее изменять данные и выполнять сохранение проделанных изменений.

**3 Формы и структуры программы**

**3.1 Запись**

В качестве предметной области был выбран обыкновенный магазин. Запись программы должна содержать основную информацию о товарах данного магазина, отраженную в следующих полях:

– Код товара (Строка) – выполняет функцию первичного ключа;

– Наименование товара (Строка):

– Тип товара (Строка);

– Количество товара (Целое число);

– Цена товара (Вещественное число);

– Является подакцизным (Булево значение).

**3.2 Формы программы**

Данный раздел содержит основные сведения о формах программы и их основных компонентах.

**3.2.1 Главная форма**

Главная форма (MainForm) **–** основная форма приложения, служащая для отображения всех его разделов и выполнения различного рода функций, в число которых входит редактирование таблицы записей, сохранение и создание файлов программы, сборка запросов и настройка программы. Она содержит следующие компоненты:

1. MainMenu – компонент главной формы, необходимый для предоставления функция по созданию, сохранению и открытию файлов, а также просмотра информации о программе:

1.2. FileMenuElement – компонент меню, содержащий компоненты для создания, открытия, сохранения файла и выхода из приложения;

1.3. AboutProgramMenuElement – компонент меню, содержащий компоненты для отображения информации о возможностях программы и ее разработке;

2. OpenDialog – объект-диалог, позволяющий открывать пользовательский диалог, с целью получения ссылки на интересующий пользователя файл базы;

3. SaveDialog – объект-диалог, позволяющий открывать пользовательский диалог, с целью получения ссылки на созданный пользователем файл, в результате работы диалога;

4. FontDialog – объект-диалог, позволяющий открывать пользовательский диалог, с целью предоставления пользователю возможности выбрать шрифт программы;

5. ColorDialog – объект-диалог, позволяющий открывать пользовательский диалог, с целью предоставления пользователю возможности выбрать цвет заднего фона программы;

6. MainControlPage – элемент каталога страниц формы, позволяющий распределить большое количество виджетов на различных страницах, с целью экономии места и логического распределения частей графического интерфейса на форме:

6.1 TablePage – страница MainControlPage, для расположения таблицы и виджетов для работы с ней:

6.1.1 Table – главная таблица программы для хранения записей о товарах;

6.1.2 Sort\_GB – группа виджетов, для сортировки записей в Table:

6.1.2.1 SortButton – кнопка для сортировки таблицы;

6.1.2.2 SortFieldCB – поле со списком для выбора поля записи для сортировки:

6.1.3 Record\_GB – группа виджетов для изменения данных в таблице:

6.1.3.1 AddButton – кнопка для добавления записи в Table;

6.1.3.2 EditButton – кнопка для редактирования выбранной записи;

6.1.3.3 DelButton – кнопка для удаления записи из Table;

6.2 QuaryPage – страница MainControlPage для расположения виджетов, необходимых для составления запроса:

6.2.1 QuaryParametersGB – группа виджетов, для настройки запроса;

6.2.1.1 FieldLabel/ParameterLabel – надписи для указания групп виджетов, отвечающих за отображение полей запроса и их условия отображения;

6.2.1.2 CodeFieldCB, NameFieldCB, TypeFieldCB, CountFieldCB, PriceFieldCB, AckFieldCB – чекбоксы, для включения в запрос соответствующих полей;

6.2.1.3 CodeParameterCB, NameParameterCB, TypeParameterCB, CountParameterCB, PriceParameterCB, AckParameterCB – поля со списком, необходимые для указания условия отбора записи, по соответствующему полю;

6.2.1.4 CodeParameterEdit, NameParameterEdit, TypeParameterEdit, CountParameterEdit, PriceParameterEdit – поля для ввода значений, дополняющих условия, описанные полями со списком из пункта 6.2.1.2;

6.2.2 QuaryButton – кнопка для отображения формы запроса (QuaryForm);

6.3 SettingsPage страница MainControlPage для размещения виджетов настройки программы;

6.3.1 TableSettingGB – группа виджетов, для настройки данных таблицы:

6.3.1.1 CodeNumberCB – чекбокс, отвечающий за возможность использовать буквы в коде товара;

6.3.1.2 PossibleTypeEdit – поле для ввода типа товара, который необходимо добавить или удалить;

6.3.1.3 AddTypeButton – поле для добавления, введенного в PossibleTypeEdit, типа товара;

6.3.1.4 DelTypeButton – поле для удаления, введенного в PossibleTypeEdit, типа товара;

6.3.2 VisualSettingsGB – группа виджетов, для визуальной настройки программы:

6.3.2.1 FomtChangeButton – кнопка для открытия FontDialog и применения пользовательского шрифта ко всем текстам в форме;

6.3.2.2 ColorButton – кнопка для открытия ColorDialog и изменения шрифта заднего фона на всех формах программы.

**3.2.2 Форма для заполнения записи**

Форма (EditForm) для заполнения записи – это форма, предназначенная для добавления и удаления записей из таблицы. Она содержит следующие элементы:

1 ElementAdd\_GB – группа виджетов, с помощью которых пользователь может заполнить и отредактировать каждое поле записи товара:

1.1 CodeEdit, NameEdit, Countedit, PriceEdit – поля, в которые пользователь сможет ввести значения соответствующих параметров записи;

1.2 TypeCB и TypeLabel – поле со списком и надпись для него, в котором пользователь сможет выбрать тип товара;

1.3 AckCB – чекбокс, для указания акциза товара;

2. OkButton – кнопка, служащая для выхода из формы с положительным результатом;

3. CancelButton – кнопка, служащая для выхода из формы c отрицательным результатом.

**3.2.3 Диалоговая форма**

Диалоговая форма (DialogForm) – это форма, необходимая для запроса подтверждения каких-либо действий пользователя и обработке событий на его основе. Она содержит следующие элементы:

1. InfoLabel – надпись для указания подтверждения, требуемого от пользователя;

2. OkButton – кнопка, служащая для выхода из формы с положительным результатом;

3. CancelButton – кнопка, служащая для выхода из формы c отрицательным результатом.

**3.2.4 Форма запроса**

Форма запроса (QuaryForm) – это форма для отображения обработанных результатов запроса. Она содержит всего один элемент – таблицу, в которую будут помещены отобранные записи.

**4. Результаты работы программы**

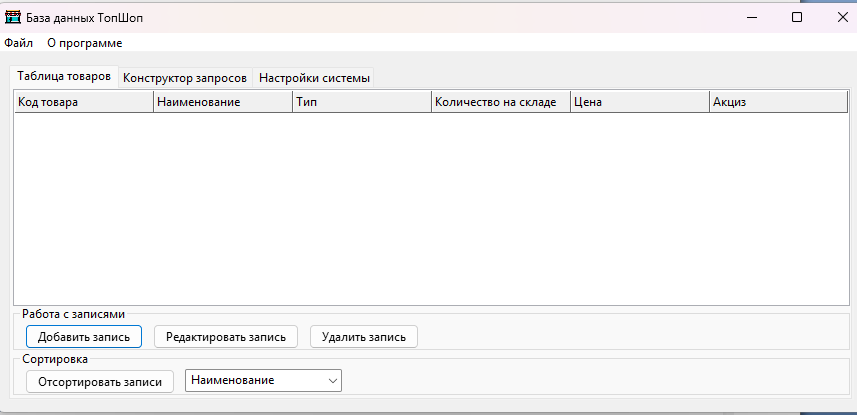


Рисунок 1 – Главная таблица программы

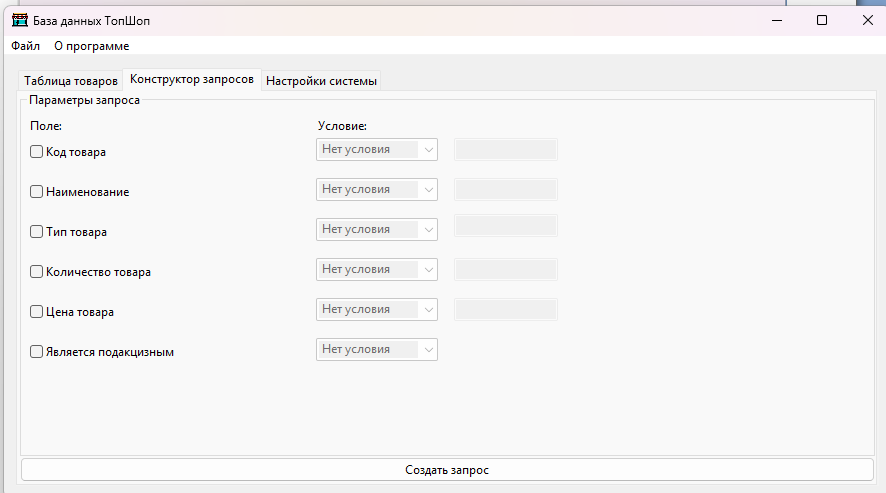


Рисунок 2 – Конструктор запросов программы

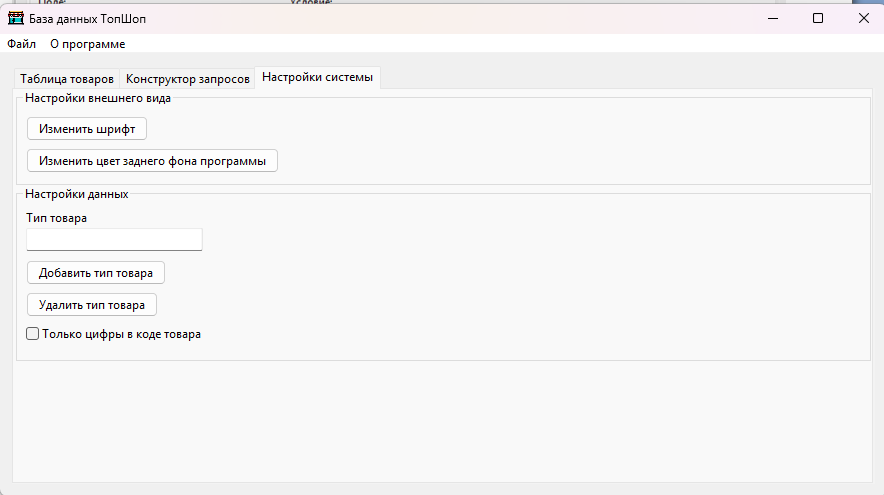


Рисунок 3 – Настройки программы

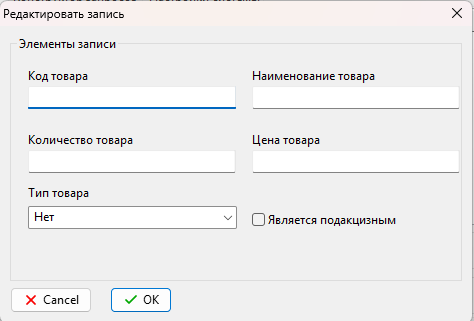


Рисунок 4 – Окно добавления записи

**5. Код программы**

**5.1 Код модуля main\_unit**

unit Main\_unit;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, ComCtrls, Grids,

StdCtrls, Buttons, Menus, ExtCtrls, edit\_unit, ack\_unit, quary\_unit;

type

{ TMainForm }

TMainForm = class(TForm)

AddButton: TBitBtn;

AddTypeButton: TButton;

CodeFieldCB: TCheckBox;

CodeParameterCB: TComboBox;

NameParameterCB: TComboBox;

TypeParameterCB: TComboBox;

CountParameterCB: TComboBox;

PriceParameterCB: TComboBox;

AckParameterCB: TComboBox;

CodeParameterEdit: TEdit;

NameParameterEdit: TEdit;

TypeParameterEdit: TEdit;

CountParameterEdit: TEdit;

PriceParameterEdit: TEdit;

NameFieldCB: TCheckBox;

TypeFieldCB: TCheckBox;

CountFieldCB: TCheckBox;

PriceFieldCB: TCheckBox;

AckFieldCB: TCheckBox;

FieladLebel: TLabel;

ParametersLabel: TLabel;

QuaryButton: TButton;

ColorButton: TButton;

ColorDialog: TColorDialog;

FomtChangeButton: TButton;

DellTypeButton: TButton;

CodeNumberCB: TCheckBox;

EditButton: TBitBtn;

DelButton: TBitBtn;

FontDialog: TFontDialog;

AboutProgramMenuElement: TMenuItem;

QuaryParametersGB: TGroupBox;

Posibilitis: TMenuItem;

Develop: TMenuItem;

VisualSettingsGB: TGroupBox;

PossibleTypeEdit: TLabeledEdit;

TableSettingsGB: TGroupBox;

MainMenu: TMainMenu;

FileMenuElement: TMenuItem;

NewElement: TMenuItem;

SaveElement: TMenuItem;

OpenDialog: TOpenDialog;

OpenMenuElement: TMenuItem;

ExitMenuElement: TMenuItem;

SaveDialog: TSaveDialog;

SortButton: TBitBtn;

SortFieldCB: TComboBox;

Sort\_GB: TGroupBox;

Record\_GB: TGroupBox;

MainControlPage: TPageControl;

Table: TStringGrid;

TablePage: TTabSheet;

QuaryPage: TTabSheet;

SttingsPage: TTabSheet;

type

ElementRecord = record

ECode : string[20];

EName : string[40];

EType : string[40];

ECount : integer;

EPrice : double;

EAck : boolean;

end;

procedure AckFieldCBChange(Sender: TObject);

procedure AddButtonClick(Sender: TObject);

procedure AddTypeButtonClick(Sender: TObject);

procedure CodeFieldCBChange(Sender: TObject);

procedure CodeNumberCBChange(Sender: TObject);

procedure ColorButtonClick(Sender: TObject);

procedure CountFieldCBChange(Sender: TObject);

procedure DelButtonClick(Sender: TObject);

procedure DellTypeButtonClick(Sender: TObject);

procedure DevelopClick(Sender: TObject);

procedure EditButtonClick(Sender: TObject);

procedure ExitMenuElementClick(Sender: TObject);

procedure FomtChangeButtonClick(Sender: TObject);

procedure FormClose(Sender: TObject; var CloseAction: TCloseAction);

procedure NameFieldCBChange(Sender: TObject);

procedure NewElementClick(Sender: TObject);

procedure OpenMenuElementClick(Sender: TObject);

procedure PosibilitisClick(Sender: TObject);

procedure PriceFieldCBChange(Sender: TObject);

procedure QuaryButtonClick(Sender: TObject);

procedure SaveElementClick(Sender: TObject);

procedure SortButtonClick(Sender: TObject);

procedure TypeFieldCBChange(Sender: TObject);

private

public

end;

var

MainForm: TMainForm;

FilePath: string;

PrimaryKeyArray: array of string;

implementation

{$R \*.lfm}

{ TMainForm }

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Методы работы с объектом MainMenu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

procedure TMainForm.NewElementClick(Sender: TObject); // Обработка события создания нового файла пользователем

var

OutPutFile : file of ElementRecord;

OutPutElement: ElementRecord;

i: integer;

NewPath: string;

begin

// Считываем новый файл через диалог

try

if SaveDialog.Execute = True then

NewPath := SaveDialog.FileName;

except

end;

// Проверяем необходимые условия, что данный файл не текущий, и что пользователь действительно его выбрал, иначе метод завершиться

if not((FilePath = NewPath) or (NewPath = '')) then

begin

// Проверяем, открыт ли какой либо файл, для сохранения в него перед очисткой таблицы

if FilePath <> '' then

begin

AssignFile(OutPutFile, FilePath);

Rewrite(OutPutFile);

for i:= 1 to Table.RowCount - 1 do

begin

OutPutElement.ECode := Table.Cells[0, i];

OutPutElement.EName := Table.Cells[1, i];

OutPutElement.EType := Table.Cells[2, i];

OutPutElement.ECount := strtoint(Table.Cells[3, i]);

OutPutElement.EPrice := strtofloat(Table.Cells[4, i]);

if Table.Cells[5, i] = 'Да' then

OutPutElement.EAck := True

else

OutPutElement.EAck := False;

Write(OutPutFile, OutPutElement);

end;

CloseFile(OutPutFile);

end;

// Изменяем атрибут адреса файла и очищаем таблицу

FilePath := NewPath;

Table.RowCount := 1;

MainForm.Caption:= 'База данных ТопШоп \' + FilePath;

end;

end;

procedure TMainForm.OpenMenuElementClick(Sender: TObject); // Самое сложное событие открытия нового файла

var

InputFile: file of ElementRecord;

OutPutFile: file of ElementRecord;

OutPutElement: ElementRecord;

ReadElement: ElementRecord;

i: integer;

NewPath: string;

begin

// Проверяем наличие адреса сохраненного файла, если его нет - запрашиваем у пользователя

if FilePath = '' then

begin

ack\_unit.DialogForm.InfoLabel.Caption := 'Сохранить текущую таблицу?';

ack\_unit.DialogForm.ShowModal;

if ack\_unit.DialogForm.ModalResult = 1 then

try

if SaveDialog.Execute = True then

FilePath := SaveDialog.FileName;

except

end;

end;

// Через диалог получаем адрес файла

try

if OpenDialog.Execute = True then

NewPath := OpenDialog.FileName;

except

end;

// Проверяем, не равен ли этот адрес предыдущему, и то, что пользователь выбрал адрес, а не просто закрыл окно

if not((FilePath = NewPath) or (NewPath = '')) then

begin

// Сохраняем все данные в текущий открытый файл, если таковой имеется

if FilePath <> '' then

begin

AssignFile(OutPutFile, FilePath);

Rewrite(OutPutFile);

for i:= 1 to Table.RowCount - 1 do

begin

OutPutElement.ECode := Table.Cells[0, i];

OutPutElement.EName := Table.Cells[1, i];

OutPutElement.EType := Table.Cells[2, i];

OutPutElement.ECount := strtoint(Table.Cells[3, i]);

OutPutElement.EPrice := strtofloat(Table.Cells[4, i]);

if Table.Cells[5, i] = 'Да' then

OutPutElement.EAck := True

else

OutPutElement.EAck := False;

Write(OutPutFile, OutPutElement);

SetLength(PrimaryKeyArray, 0);

end;

CloseFile(OutPutFile);

end;

// Очищаем таблицу и изменяем адрес текущего файла на введенный

FilePath := NewPath;

Table.RowCount := 1;

MainForm.Caption := 'База данных ТопШоп \' + FilePath;

// Считываем таблицу из введенного файла

AssignFile(InputFile, FilePath);

Reset(InputFile);

while not EOf(InputFile) do

begin

Read(InputFile, ReadElement);

Table.RowCount := Table.RowCount + 1;

Table.Cells[0, Table.RowCount - 1] := ReadElement.ECode;

Table.Cells[1, Table.RowCount - 1] := ReadElement.EName;

Table.Cells[2, Table.RowCount - 1] := ReadElement.EType;

Table.Cells[3, Table.RowCount - 1] := inttostr(ReadElement.ECount);

Table.Cells[4, Table.RowCount - 1] := floattostr(ReadElement.EPrice);

if ReadElement.EAck = True then

Table.Cells[5, Table.RowCount - 1] := 'Да'

else

Table.Cells[5, Table.RowCount - 1] := 'Нет'

end;

CloseFile(InputFile);

end;

end;

procedure TMainForm.SaveElementClick(Sender: TObject);

var

OutPutFile: file of ElementRecord;

OutPutElement: ElementRecord;

i : integer;

begin

// Проверяяем есть ли открытый файл базы, если его нет, предлагаем пользователю создать его

if FilePath = '' then

try

ack\_unit.DialogForm.InfoLabel.Caption := 'Создать новый файл для сохранения?';

ack\_unit.DialogForm.ShowModal;

if ack\_unit.DialogForm.ModalResult = 1 then

if SaveDialog.Execute = True then

FilePath := SaveDialog.FileName;

except

end;

// Если файл был изначально, или его создал пользователь, необходимо сохранить информацию в него

if FilePath <> '' then

begin

AssignFile(OutPutFile, FilePath);

Rewrite(OutPutFile);

for i:= 1 to Table.RowCount - 1 do

begin

OutPutElement.ECode := Table.Cells[0, i];

OutPutElement.EName := Table.Cells[1, i];

OutPutElement.EType := Table.Cells[2, i];

OutPutElement.ECount := strtoint(Table.Cells[3, i]);

OutPutElement.EPrice := strtofloat(Table.Cells[4, i]);

if Table.Cells[5, i] = 'Да' then

OutPutElement.EAck := True

else

OutPutElement.EAck := False;

Write(OutPutFile, OutPutElement);

end;

CloseFile(OutPutFile);

end;

end;

procedure TMainForm.ExitMenuElementClick(Sender: TObject);

begin

MainForm.Close;

end;

procedure TMainForm.FormClose(Sender: TObject; var CloseAction: TCloseAction); // СобытияЮ отслвеживающее закрытие приложения для сохранения файла

var

OutPutFile: file of ElementRecord;

OutPutElement: ElementRecord;

i : integer;

begin

if FilePath = '' then // Если файл не открыт, необходимо предложить пользователю создать его

try

ack\_unit.DialogForm.InfoLabel.Caption := 'Создать новый файл для сохранения?';

ack\_unit.DialogForm.ShowModal;

if ack\_unit.DialogForm.ModalResult = 1 then

if SaveDialog.Execute = True then

FilePath := SaveDialog.FileName;

except

end;

if FilePath <> '' then // Если файл был изначально, или его создал пользователь, необходимо сохранить данные таблицы в него

begin

AssignFile(OutPutFile, FilePath);

Rewrite(OutPutFile);

for i:= 1 to Table.RowCount - 1 do

begin

OutPutElement.ECode := Table.Cells[0, i];

OutPutElement.EName := Table.Cells[1, i];

OutPutElement.EType := Table.Cells[2, i];

OutPutElement.ECount := strtoint(Table.Cells[3, i]);

OutPutElement.EPrice := strtofloat(Table.Cells[4, i]);

if Table.Cells[5, i] = 'Да' then

OutPutElement.EAck := True

else

OutPutElement.EAck := False;

Write(OutPutFile, OutPutElement);

end;

CloseFile(OutPutFile);

end;

end;

procedure TMainForm.PosibilitisClick(Sender: TObject);

begin

showmessage('Программа ТопШоп предоставляет базовые возможности работы с базой данных, включая настройку и редактирование табличной базы и составление алиментарных запросов.');

end;

procedure TMainForm.DevelopClick(Sender: TObject);

begin

showmessage('Программа ТопШоп разработана в рамках курса МДК 05.02 "Разработка кода ИС" и является свободно распространяемым программным обеспечением.');

end;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Методы работы с главным табличным окном\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

procedure TMainForm.AddButtonClick(Sender: TObject);

var

price\_for\_checking\_correct\_input: double;

row:integer;

begin

// Очищаем окно редактирования

edit\_unit.EditForm.CodeEdit.Clear;

edit\_unit.EditForm.NameEdit.Clear;

edit\_unit.EditForm.CountEdit.Clear;

edit\_unit.EditForm.PriceEdit.Clear;

edit\_unit.EditForm.TypeCB.Text := 'Нет';

edit\_unit.EditForm.AckCB.Checked:=False;

// Открываем и обрабатываем события пользователя в окне редактирования

edit\_unit.EditForm.ShowModal;

if edit\_unit.EditForm.ModalResult = 2 then

exit()

else if edit\_unit.EditForm.CodeEdit.Text = '' then

showmessage('Пропущено ключевое поле "Код товара"!')

else

begin

for row:= 1 to Table.RowCount - 1 do

if (edit\_unit.EditForm.CodeEdit.Text = Table.Cells[0, row]) then

begin

showmessage('Данный ключ уже присутствует в таблице!');

exit();

end;

try

price\_for\_checking\_correct\_input := strtofloat(edit\_unit.EditForm.PriceEdit.text);

except

showmessage('Неверно введена цена!');

exit();

end;

Table.RowCount:= Table.RowCount + 1;

Table.Cells[0, Table.RowCount - 1] := edit\_unit.EditForm.CodeEdit.Text;

Table.Cells[1, Table.RowCount - 1] := edit\_unit.EditForm.NameEdit.Text;

Table.Cells[2, Table.RowCount - 1] := edit\_unit.EditForm.TypeCB.Text;

Table.Cells[3, Table.RowCount - 1] := edit\_unit.EditForm.CountEdit.Text;

Table.Cells[4, Table.RowCount - 1] := floattostr(price\_for\_checking\_correct\_input);

if edit\_unit.EditForm.AckCB.Checked = True then

Table.Cells[5, Table.RowCount - 1] := 'Да'

else

Table.Cells[5, Table.RowCount - 1] := 'Нет';

end;

end;

procedure TMainForm.DelButtonClick(Sender: TObject);

begin

if Table.Row < 1 then

exit();

ack\_unit.DialogForm.InfoLabel.Caption := 'Подтвердить удаление записи №' + inttostr(Table.Row) + '?';

ack\_unit.DialogForm.ShowModal;

if ack\_unit.DialogForm.ModalResult = 1 then

Table.DeleteRow(Table.Row);

end;

procedure TMainForm.EditButtonClick(Sender: TObject);

var

price\_for\_checking\_correct\_input: double;

row: integer;

begin

// Проверяем выбранное поле

if Table.Row < 1 then

exit();

// Загружаем данные в окно редактирования

edit\_unit.EditForm.CodeEdit.Text := Table.Cells[0, Table.Row];

edit\_unit.EditForm.NameEdit.Text := Table.Cells[1, Table.Row];

edit\_unit.EditForm.TypeCB.Text := Table.Cells[2, Table.Row];

edit\_unit.EditForm.CountEdit.Text := Table.Cells[3, Table.Row];

edit\_unit.EditForm.PriceEdit.text := Table.Cells[4, Table.Row];

if Table.Cells[5, Table.Row] = 'Да' then

edit\_unit.EditForm.AckCB.Checked := True

else

edit\_unit.EditForm.AckCB.Checked := False;

// Открываем и обрабатываем ввод в окно редактирования

edit\_unit.EditForm.ShowModal;

if edit\_unit.EditForm.ModalResult = 2 then

exit()

else if edit\_unit.EditForm.CodeEdit.Text = '' then

showmessage('Пропущено ключевое поле "Код товара"!')

else

begin

for row:= 1 to Table.RowCount - 1 do

if (edit\_unit.EditForm.CodeEdit.Text = Table.Cells[0, row]) and (row <> Table.Row) then

begin

showmessage('Данный ключ уже присутствует в таблице!');

exit();

end;

try

price\_for\_checking\_correct\_input := strtofloat(edit\_unit.EditForm.PriceEdit.text);

except

showmessage('Неверно введена цена!');

exit();

end;

Table.Cells[0, Table.Row] := edit\_unit.EditForm.CodeEdit.Text;

Table.Cells[1, Table.Row] := edit\_unit.EditForm.NameEdit.Text;

Table.Cells[2, Table.Row] := edit\_unit.EditForm.TypeCB.Text;

Table.Cells[3, Table.Row] := edit\_unit.EditForm.CountEdit.Text;

Table.Cells[4, Table.Row] := floattostr(price\_for\_checking\_correct\_input);

if edit\_unit.EditForm.AckCB.Checked = True then

Table.Cells[5, Table.Row] := 'Да'

else

Table.Cells[5, Table.Row] := 'Нет';

end;

end;

procedure TMainForm.SortButtonClick(Sender: TObject);

begin

case SortFieldCB.ItemIndex of

0: Table.SortColRow(True, 1);

1: Table.SortColRow(True, 2);

2: Table.SortColRow(True, 5);

end;

end;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Методы для работы окна настроек\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

procedure TMainForm.CodeNumberCBChange(Sender: TObject);

begin

edit\_unit.EditForm.CodeEdit.NumbersOnly:= CodeNumberCb.Checked;

end;

procedure TMainForm.AddTypeButtonClick(Sender: TObject);

var

types: integer;

UserType:string;

begin

UserType := PossibleTypeEdit.Text;

PossibleTypeEdit.Clear;

if UserType <> '' then

begin

for types:= 0 to edit\_unit.EditForm.TypeCB.Items.Count - 1 do

if edit\_unit.EditForm.TypeCB.Items.Strings[types] = UserType then

begin

showmessage('Данный тип уже присутствует!');

exit();

end;

edit\_unit.EditForm.TypeCB.Items.Add(UserType);

showmessage('Тип успешно добавлен!');

end;

end;

procedure TMainForm.DellTypeButtonClick(Sender: TObject);

var

types: integer;

UserType:string;

begin

UserType := PossibleTypeEdit.Text;

PossibleTypeEdit.Clear;

if (UserType <> '') and (UserType <> 'Нет') then

begin

for types:= 0 to edit\_unit.EditForm.TypeCB.Items.Count - 1 do

if edit\_unit.EditForm.TypeCB.Items.Strings[types] = UserType then

begin

edit\_unit.EditForm.TypeCB.Items.Delete(types);

showmessage('Тип успешно удален!');

exit();

end;

showmessage('Данный тип отсутствует!');

end;

end;

procedure TMainForm.FomtChangeButtonClick(Sender: TObject);

var

UserFont: TFont;

begin

try

if FontDialog.Execute = True then

UserFont := FontDialog.Font;

except

end;

if UserFont <> Nil then

begin

MainForm.Font := UserFont;

edit\_unit.EditForm.Font := UserFont;

ack\_unit.DialogForm.Font := UserFont;

quary\_unit.QuaryForm.Font := UserFont;

end;

end;

procedure TMainForm.ColorButtonClick(Sender: TObject);

var

UserColor: TColor;

begin

try

if ColorDialog.Execute = True then

UserColor := ColorDialog.Color;

except

end;

MainForm.Color := UserColor;

edit\_unit.EditForm.Color := UserColor;

ack\_unit.DialogForm.Color := UserColor;

quary\_unit.QuaryForm.Color := UserColor;

end;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Методы для окна запросов\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

procedure TMainForm.CodeFieldCBChange(Sender: TObject);

begin

CodeParameterCB.Enabled := CodeFieldCB.Checked;

CodeParameterEdit.Enabled:= CodeFieldCB.Checked;

if CodeFieldCB.Checked = False then

begin

CodeParameterCB.Text:= 'Нет условия';

CodeParameterEdit.Clear;

end;

end;

procedure TMainForm.NameFieldCBChange(Sender: TObject);

begin

NameParameterCB.Enabled := NameFieldCB.Checked;

NameParameterEdit.Enabled:= NameFieldCB.Checked;

if NameFieldCB.Checked = False then

begin

NameParameterCB.Text:= 'Нет условия';

NameParameterEdit.Clear;

end;

end;

procedure TMainForm.TypeFieldCBChange(Sender: TObject);

begin

TypeParameterCB.Enabled := TypeFieldCB.Checked;

TypeParameterEdit.Enabled:= TypeFieldCB.Checked;

if TypeFieldCB.Checked = False then

begin

TypeParameterCB.Text:= 'Нет условия';

TypeParameterEdit.Clear;

end;

end;

procedure TMainForm.CountFieldCBChange(Sender: TObject);

begin

CountParameterCB.Enabled := CountFieldCB.Checked;

CountParameterEdit.Enabled:= CountFieldCB.Checked;

if CountFieldCB.Checked = False then

begin

CountParameterCB.Text:= 'Нет условия';

CountParameterEdit.Clear;

end;

end;

procedure TMainForm.PriceFieldCBChange(Sender: TObject);

begin

PriceParameterCB.Enabled := PriceFieldCB.Checked;

PriceParameterEdit.Enabled:= PriceFieldCB.Checked;

if PriceFieldCB.Checked = False then

begin

PriceParameterCB.Text:= 'Нет условия';

PriceParameterEdit.Clear;

end;

end;

procedure TMainForm.AckFieldCBChange(Sender: TObject);

begin

AckParameterCB.Enabled := AckFieldCB.Checked;

if AckFieldCB.Checked = False then

AckParameterCB.Text:= 'Нет условия';

end;

procedure TMainForm.QuaryButtonClick(Sender: TObject);

var

FirstParameter, SecondParameter, ThirdParameter, FourthParameter, FifthParameter, SixthParameter: boolean;

MainTableRow, ColomnQuaryNumber: integer;

begin

quary\_unit.QuaryForm.QuaryTable.RowCount:= 1;

quary\_unit.QuaryForm.QuaryTable.ColCount:= 0;

if CodeFieldCB.Checked = True then

begin

quary\_unit.QuaryForm.QuaryTable.ColCount := quary\_unit.QuaryForm.QuaryTable.ColCount + 1;

quary\_unit.QuaryForm.QuaryTable.Cells[quary\_unit.QuaryForm.QuaryTable.ColCount - 1, 0] := 'Код товара';

end;

if NameFieldCB.Checked = True then

begin

quary\_unit.QuaryForm.QuaryTable.ColCount := quary\_unit.QuaryForm.QuaryTable.ColCount + 1;

quary\_unit.QuaryForm.QuaryTable.Cells[quary\_unit.QuaryForm.QuaryTable.ColCount - 1, 0] := 'Наименовение';

end;

if TypeFieldCB.Checked = True then

begin

quary\_unit.QuaryForm.QuaryTable.ColCount := quary\_unit.QuaryForm.QuaryTable.ColCount + 1;

quary\_unit.QuaryForm.QuaryTable.Cells[quary\_unit.QuaryForm.QuaryTable.ColCount - 1, 0] := 'Тип товара';

end;

if CountFieldCB.Checked = True then

begin

quary\_unit.QuaryForm.QuaryTable.ColCount := quary\_unit.QuaryForm.QuaryTable.ColCount + 1;

quary\_unit.QuaryForm.QuaryTable.Cells[quary\_unit.QuaryForm.QuaryTable.ColCount - 1, 0] := 'Количество товара на складе';

end;

if PriceFieldCB.Checked = True then

begin

quary\_unit.QuaryForm.QuaryTable.ColCount := quary\_unit.QuaryForm.QuaryTable.ColCount + 1;

quary\_unit.QuaryForm.QuaryTable.Cells[quary\_unit.QuaryForm.QuaryTable.ColCount - 1, 0] := 'Цена товара';

end;

if AckFieldCB.Checked = True then

begin

quary\_unit.QuaryForm.QuaryTable.ColCount := quary\_unit.QuaryForm.QuaryTable.ColCount + 1;

quary\_unit.QuaryForm.QuaryTable.Cells[quary\_unit.QuaryForm.QuaryTable.ColCount - 1, 0] := 'Является подакцизным';

end;

for MainTableRow:= 1 to Table.RowCount - 1 do

begin

if (CodeFieldCB.Checked = False) or (CodeParameterCB.Text = 'Нет условия') or ((CodeParameterCB.Text = 'Равно') and (Table.Cells[0, MainTableRow] = CodeParameterEdit.Text)) or ((CodeParameterCB.Text = 'Содержит') and (Pos(CodeParameterEdit.Text, Table.Cells[0, MainTableRow]) <> 0)) then

FirstParameter := True

else

FirstParameter := False;

if (NameFieldCB.Checked = False) or (NameParameterCB.Text = 'Нет условия') or ((NameParameterCB.Text = 'Равно') and (Table.Cells[1, MainTableRow] = NameParameterEdit.Text)) or ((NameParameterCB.Text = 'Содержит') and (Pos(NameParameterEdit.Text, Table.Cells[1, MainTableRow]) <> 0)) then

SecondParameter := True

else

SecondParameter := False;

if (TypeFieldCB.Checked = False) or (TypeParameterCB.Text = 'Нет условия') or ((TypeParameterCB.Text = 'Равно') and (Table.Cells[2, MainTableRow] = TypeParameterEdit.Text)) or ((TypeParameterCB.Text = 'Содержит') and (Pos(TypeParameterEdit.Text, Table.Cells[2, MainTableRow]) <> 0)) then

ThirdParameter := True

else

ThirdParameter := False;

if (CountFieldCB.Checked = False) or (CountParameterCB.Text = 'Нет условия') or ((CountParameterCB.Text = 'Равно') and (strtoint(Table.Cells[3, MainTableRow]) = strtoint(CountParameterEdit.Text))) or ((CountParameterCB.Text = 'Не равно') and (strtoint(Table.Cells[3, MainTableRow]) <> strtoint(CountParameterEdit.Text))) or ((CountParameterCB.Text = 'Больше') and (strtoint(Table.Cells[3, MainTableRow]) > strtoint(CountParameterEdit.Text))) or ((CountParameterCB.Text = 'Меньше') and (strtoint(Table.Cells[3, MainTableRow]) < strtoint(CountParameterEdit.Text))) then

FourthParameter := True

else

FourthParameter := False;

if (PriceFieldCB.Checked = False) or (PriceParameterCB.Text = 'Нет условия') or ((PriceParameterCB.Text = 'Равно') and (strtoint(Table.Cells[4, MainTableRow]) = strtoint(PriceParameterEdit.Text))) or ((PriceParameterCB.Text = 'Не равно') and (strtoint(Table.Cells[4, MainTableRow]) <> strtoint(PriceParameterEdit.Text))) or ((PriceParameterCB.Text = 'Больше') and (strtoint(Table.Cells[4, MainTableRow]) > strtoint(PriceParameterEdit.Text))) or ((PriceParameterCB.Text = 'Меньше') and (strtoint(Table.Cells[4, MainTableRow]) < strtoint(PriceParameterEdit.Text))) then

FifthParameter := True

else

FifthParameter := False;

if (AckFieldCB.Checked = False) or (AckParameterCB.Text = 'Нет условия') or ((AckParameterCB.Text = 'Подакцизный') and (Table.Cells[5, MainTableRow] = 'Да')) or ((AckParameterCB.Text = 'Не подакцизный') and (Table.Cells[5, MainTableRow] = 'Нет')) then

SixthParameter := True

else

SixthParameter := False;

if FirstParameter and SecondParameter and ThirdParameter and FourthParameter and FifthParameter and SixthParameter then

begin

quary\_unit.QuaryForm.QuaryTable.RowCount := quary\_unit.QuaryForm.QuaryTable.RowCount + 1;

ColomnQuaryNumber := 0;

if CodeFieldCB.Checked = True then

begin

quary\_unit.QuaryForm.QuaryTable.Cells[ColomnQuaryNumber, quary\_unit.QuaryForm.QuaryTable.RowCount - 1] := Table.Cells[0, MainTableRow];

ColomnQuaryNumber += 1;

end;

if NameFieldCB.Checked = True then

begin

quary\_unit.QuaryForm.QuaryTable.Cells[ColomnQuaryNumber, quary\_unit.QuaryForm.QuaryTable.RowCount - 1] := Table.Cells[1, MainTableRow];

ColomnQuaryNumber += 1;

end;

if TypeFieldCB.Checked = True then

begin

quary\_unit.QuaryForm.QuaryTable.Cells[ColomnQuaryNumber, quary\_unit.QuaryForm.QuaryTable.RowCount - 1] := Table.Cells[2, MainTableRow];

ColomnQuaryNumber += 1;

end;

if CountFieldCB.Checked = True then

begin

quary\_unit.QuaryForm.QuaryTable.Cells[ColomnQuaryNumber, quary\_unit.QuaryForm.QuaryTable.RowCount - 1] := Table.Cells[3, MainTableRow];

ColomnQuaryNumber += 1;

end;

if PriceFieldCB.Checked = True then

begin

quary\_unit.QuaryForm.QuaryTable.Cells[ColomnQuaryNumber, quary\_unit.QuaryForm.QuaryTable.RowCount - 1] := Table.Cells[4, MainTableRow];

ColomnQuaryNumber += 1;

end;

if AckFieldCB.Checked = True then

begin

quary\_unit.QuaryForm.QuaryTable.Cells[ColomnQuaryNumber, quary\_unit.QuaryForm.QuaryTable.RowCount - 1] := Table.Cells[5, MainTableRow];

ColomnQuaryNumber += 1;

end;

end;

end;

quary\_unit.QuaryForm.Show;

end;

end.

**5.2 Код модуля edit\_unit**

unit edit\_unit;

{$mode ObjFPC}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, Buttons,

ExtCtrls;

type

{ TEditForm }

TEditForm = class(TForm)

AckCB: TCheckBox;

TypeCB: TComboBox;

TypeLabel: TLabel;

CodeEdit: TLabeledEdit;

NameEdit: TLabeledEdit;

CountEdit: TLabeledEdit;

PriceEdit: TLabeledEdit;

OKButton: TBitBtn;

CancelButton: TBitBtn;

ElementAdd\_GB: TGroupBox;

private

public

end;

var

EditForm: TEditForm;

implementation

{$R \*.lfm}

{ TEditForm }

end.

**5.3 Код модуля ack\_unit**

unit ack\_unit;

{$mode ObjFPC}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, Buttons;

type

{ TDialogForm }

TDialogForm = class(TForm)

OkButton: TBitBtn;

CancelButton: TBitBtn;

InfoLabel: TLabel;

private

public

end;

var

DialogForm: TDialogForm;

implementation

{$R \*.lfm}

end.

**5.4 Код модуля Quary\_unit**

unit quary\_unit;

{$mode ObjFPC}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, Grids;

type

{ TQuaryForm }

TQuaryForm = class(TForm)

QuaryTable: TStringGrid;

private

public

end;

var

QuaryForm: TQuaryForm;

implementation

{$R \*.lfm}

end.

**6. Вывод**

Работа завершена. В ходе работы были изучены и освоены основные принципы работы с типизированными файлами, записями и диалогами в IDE Lazarus, а также закреплены навыки работы с основными компонентами графического интерфейса. В результате работы была разработана программа – база данных ТопШоп для хранения и изменения данных о товарах магазина с указанием различных характеристик, а также возможностью составления алиментарных запросов на выборку.