Цикл презентаций «ООП на Delphi» посвящен объектно – ориентированному программированию с использованием одной из самых распространенных систем быстрой разработки приложений – Delphi

Используя данный учебный курс, можно самостоятельно овладеть основами объектно – ориентированного программирования на Delphi. Для расширения Ваших знаний к курсу приложен ряд учебных пособий и справочников по Delphi

Цикл содержит 13 презентаций:

ООП на Delphi – 1: Знакомство с системой программирования Borland Delphi. Объекты (компоненты) и их свойства и методы

ООП на Delphi – 2: Первая программа на Delphi, сохранение и компиляция

ООП на Delphi – 3: Программное изменение свойств объектов

ООП на Delphi – 4: Условия в Delphi. Создание простого теста

ООП на Delphi – 5: Элементы ввода и вывода информации. Обработка исключений

ООП на Delphi – 6: Заставка программы и элемент таймер

ООП на Delphi – 7: Программируем свою игрушку

ООП на Delphi – 8: Меню программы, панель статуса, диалоги

ООП на Delphi – 9: Создаем свой текстовый редактор

ООП на Delphi – 10: Базы данных на Delphi

ООП на Delphi – 11: Калькулятор на Delphi. Обработка исключительных ситуаций

ООП на Delphi – 12: Создаем тестирующую систему

ООП на Delphi – 13: Графика на Delphi

Delphi использует язык программирования Объект Паскаль, поэтому лучше сначала изучить обычный Паскаль и поработать в ТурбоПаскале, а затем и переходить к Delphi – перейти будет очень просто, т.к синтаксис языка остается неизменным.

Изучение ООП на Delphi желательно проводить в старших профильных классах – количество часов, отводимое на информатику там вполне достаточно для освоения основ ООП на Delphi

Объектно – ориентированное программирование на

Borland® DELPHI - 9

DELPHI - 9

На этом уроке:

Мы научимся использовать стандартные диалоги и создадим свой текстовый редактор

Вопросы:

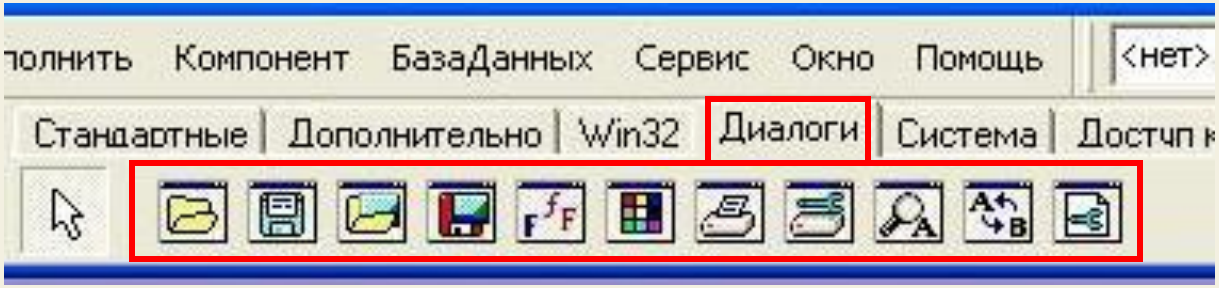
1. Использование стандартных диалогов
2. Создаем текстовый редактор

Borland®

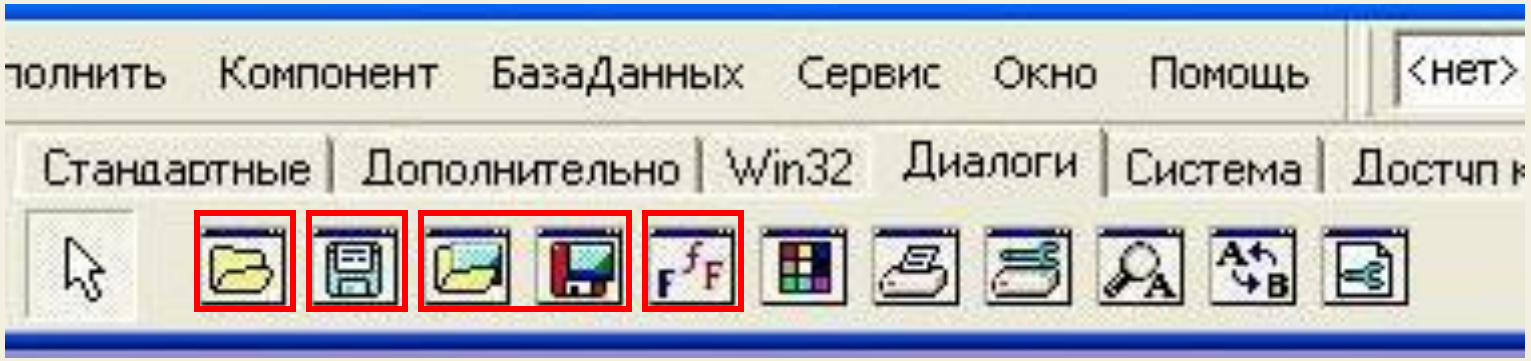
Использование стандартных диалогов

Среда разработки Delphi содержит несколько компонент для организации стандартных диалогов для открытия, поиска, сохранения, выбора шрифтов, настройки печати и т.д.(все эти диалоги мы хорошо знаем, работая с операционной системой Windows)

Рассмотрим некоторые, наиболее часто применимые диалоги, и способы их использования



Компоненты для организации диалогов находятся на вкладке ДИАЛОГИ, и все они являются НЕВИЗУАЛЬНЫМИ, т.е. не видны на форме (вспомните компонент таймер)



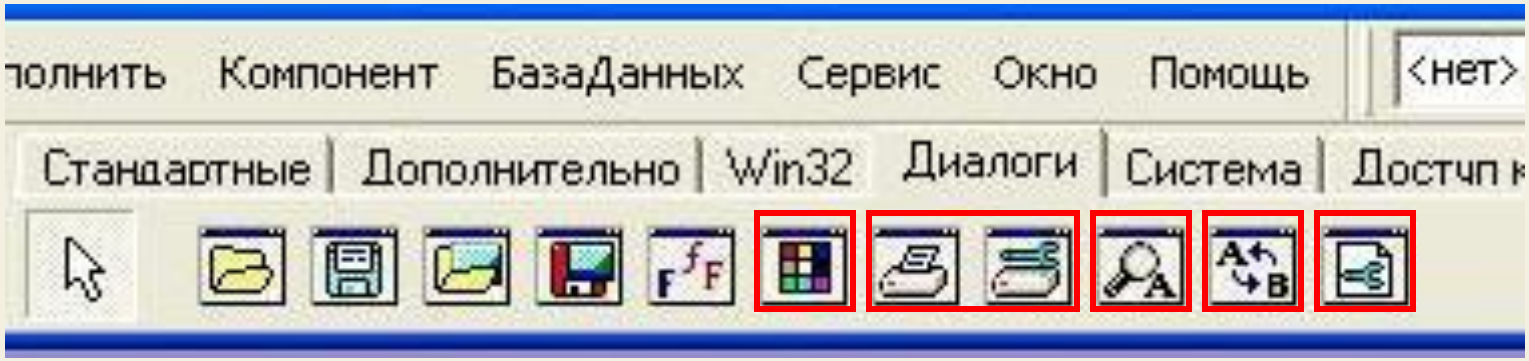
Создание окна для выбора шрифтов

Компонент «Открыть файл»

Предназначен для создания окна открытия файла

Компонент «Сохранить файл как ...» Предназначен для создания окна сохранения файла

Компоненты для открытия и сохранения рисунков



Компонент для выбора цветов

Компоненты настройки печати и установки принтера

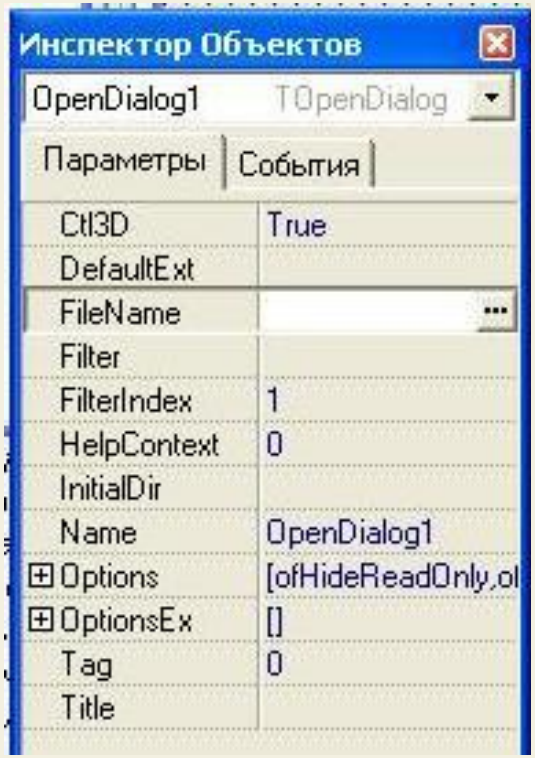
Компонент для поиска

Компонент замены текста

Компонент установки параметров страницы

Как и все объекты Delphi, компоненты организации стандартных диалогов имеют свои свойства и методы

Рассмотрим свойства компонента OpenDialog и его использование, остальные компоненты имеют с ним много общего



Основные свойства:

1) FileName (выбранный пользователем файл)

2) Filter (позволяет выбрать нужный тип файлов)

3) DefaultExt (расширение файла по умолчанию)

4) InitialDir (начальный каталог в момент открытия диалога)

5) Options (куча различных опций для диалогового окна)

Рассмотрите внимательно набор свойств компонента OpenDialog в инспекторе объектов, при этом используйте справочник А.Я. Архангельского, приложенный к курсу

Основной метод, которым производится обращение к любому диалогу — **Execute**

Стандартное обращение к диалогу имеет вид:

if <имя компонента-диалога>**.Execute then**

<операторы, использующие выбор пользователя>;

Рассмотрим применение **OpenDialog** и **SaveDialog** на простом примере:

Создадим форму, на которой будет компонент Memo и кнопки ОТКРЫТЬ (для открытия файла в Memo) и СОХРАНИТЬ (для сохранения содержимого Memo в каком-то файле)

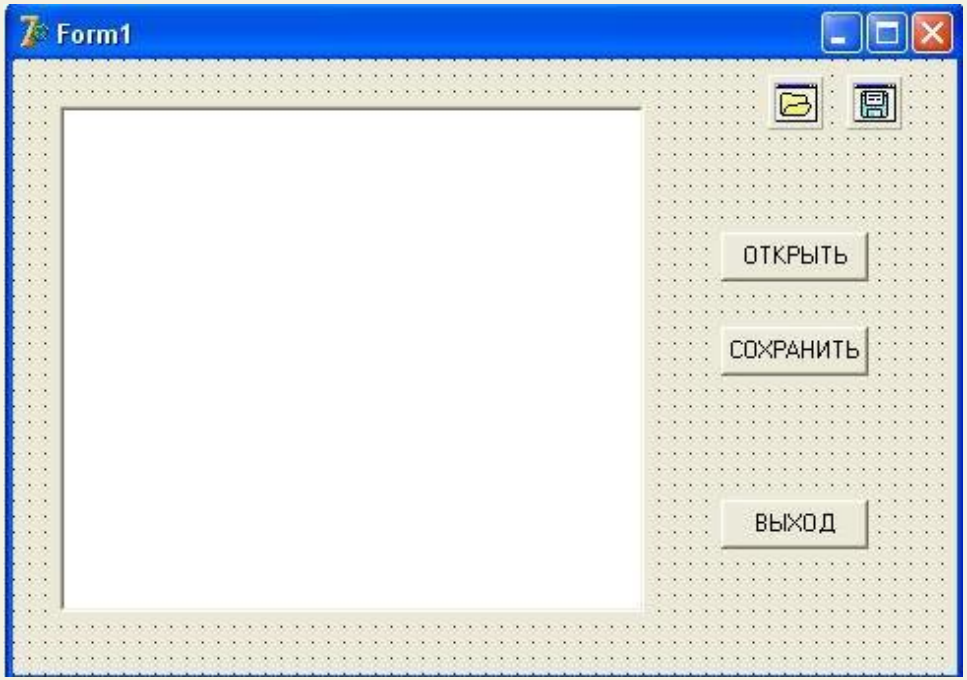
ШАГ 1

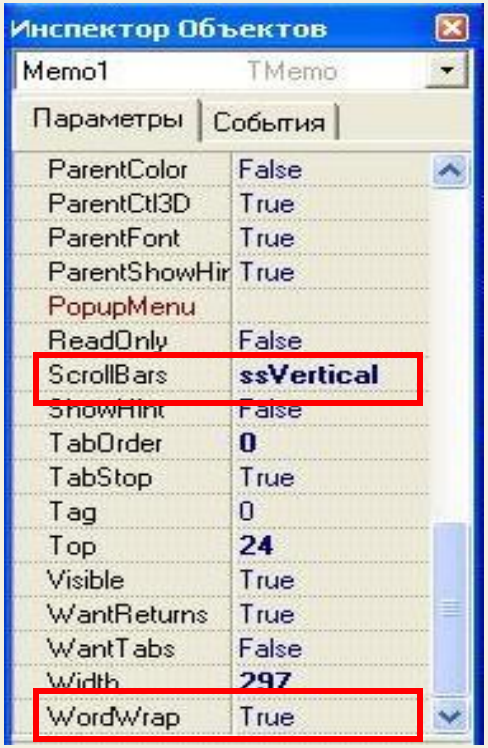
Создадим форму и поместим на нее:

Memo для отображения текста (файла)

Компоненты OpenDialog и SaveDialog

Три кнопки: ОТКРЫТЬ, СОХРАНИТЬ, ВЫХОД





Для компонента **Memo** установим :

- свойство **Scrollbars – ssVertical** (чтобы в Memo была вертикальная полоса прокрутки)
- свойство **WordWrap – True** (разрешим перенос слов в Memo)

ШАГ 2

Запишем код кнопок:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 if OpenDialog1.Execute then
  begin
   FName := OpenDialog1.FileName;
   Memo1.Lines.LoadFromFile(FName);
  end
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
 SaveDialog1.FileName := FName;
if SaveDialog1.Execute then
 begin
  FName := SaveDialog1.FileName;
  Memo1.Lines.SaveToFile(FName);
 end;


end;

procedure TForm1.Button3Click(Sender: TObject);
begin
close
end;
```

Кнопка ОТКРЫТЬ

Этот оператор вызывает диалог, проверяет, выбрал ли пользователь файл (если выбрал, то функция **Execute** возвращает **true**), после чего имя выбранного файла (**OpenDialog1.FileName**) сохраняется в переменной **FName** и файл загружается в текст **Memo1** методом **LoadFromFile**

Но для того, чтобы процедура работала, нужно объявить строковую переменную **FName**

```
var
  Form1: TForm1;
  FName: string;
implementation
```

ШАГ 2

Запишем код кнопок:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 if OpenDialog1.Execute then
  begin
   FName := OpenDialog1.FileName;
   Memo1.Lines.LoadFromFile(FName);
  end
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  SaveDialog1.FileName := FName;
if SaveDialog1.Execute then
  begin
   FName := SaveDialog1.FileName;
   Memo1.Lines.SaveToFile(FName);
  end;


end;

procedure TForm1.Button3Click(Sender: TObject);
begin
close
end;
```

Кнопка СОХРАНИТЬ

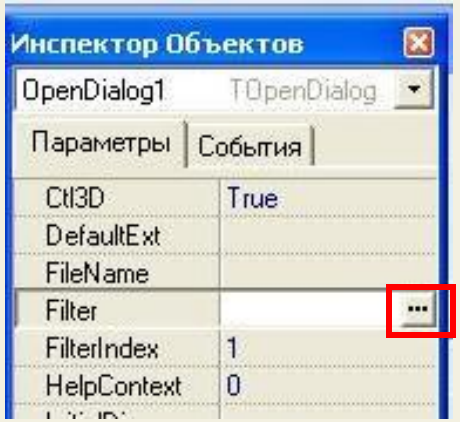
Первый из этих операторов присваивает свойству **FileName** компонента **SaveDialog1** запомненное имя файла. Это имя по умолчанию будет предложено пользователю при открытии диалога **Сохранить как….** Следующий оператор открывает диалог и, если пользователь выбрал в нем файл, запоминает новое имя файла и сохраняет в файле с этим именем текст компонента **Memo1**

Кнопка ВЫХОД

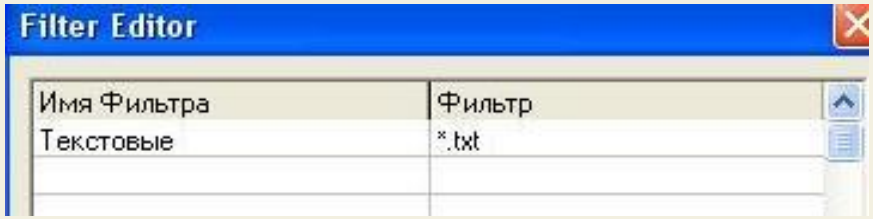
Без комментариев

ШАГ 3

Для компонента **OpenDialog1** установим **фильтр на типы файлов и расширение открываемых файлов по умолчанию**



Раскроем в инспекторе объектов свойство Filter компонента **OpenDialog1**



Укажем **тип файла** (слева) и его **расширение** (справа) – сейчас в диалоговом окне открытия файла будут отображаться только файлы с расширением *.txt

Если записать еще строку, например Вордовские | *.doc , то мы можем выбрать и вордовские файлы (MS Word)

Установим тип файлов, отображаемых при запуске диалога по умолчанию - **txt**

ШАГ 3

Все то же сделаем и для компонента SaveDialog:

установим **фильтр на типы файлов и расширение сохраняемых файлов по умолчанию**

ШАГ 4

Сохраняем и запускаем программу: программа уже умеет открывать файлы и сохранять их (и конечно редактировать содержимое файла)

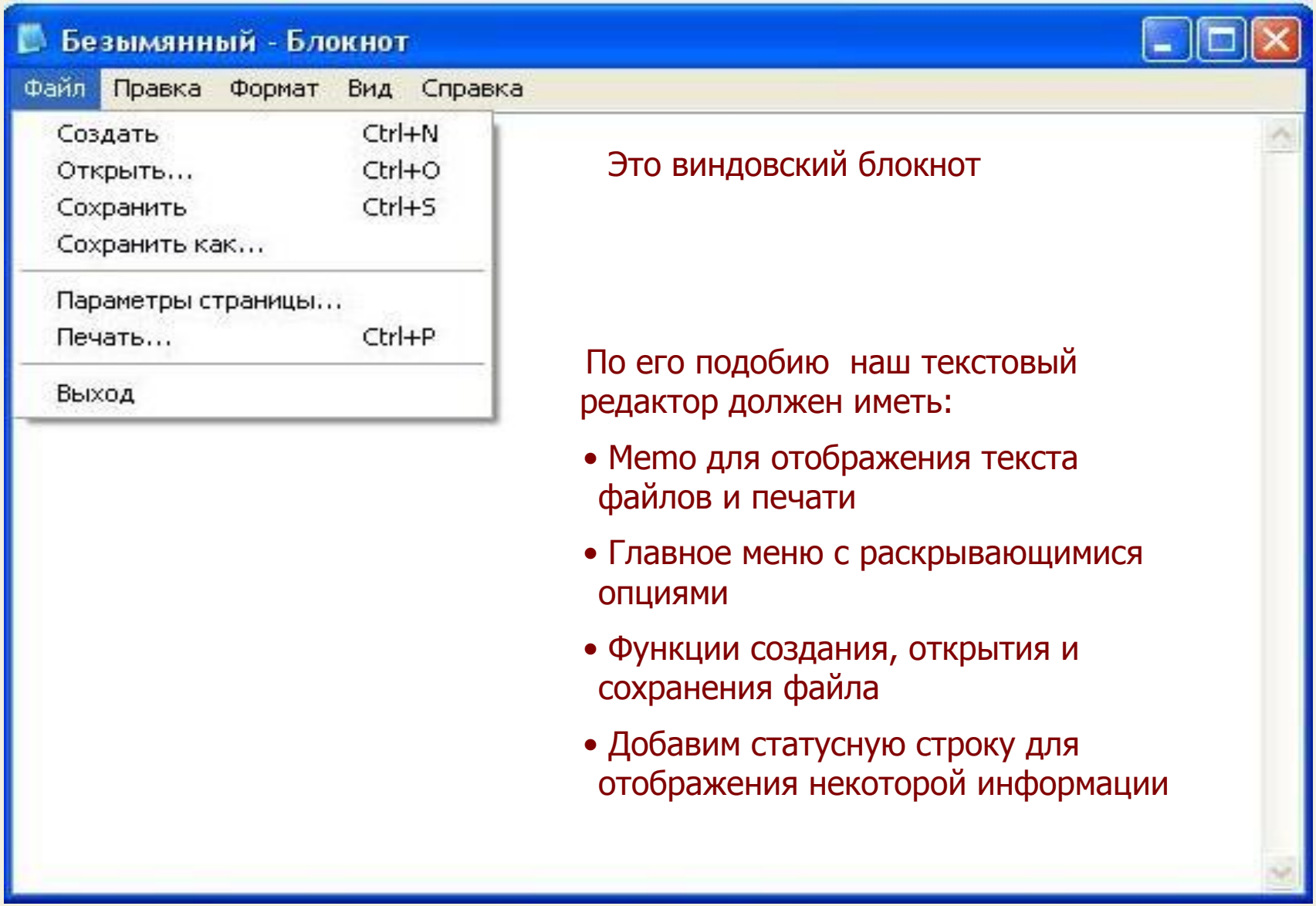
Попробовать -> 

Итак, мы рассмотрели стандартные диалоги, их свойства и методы.
Следующий вопрос – создание своего текстового редактора

Borland®

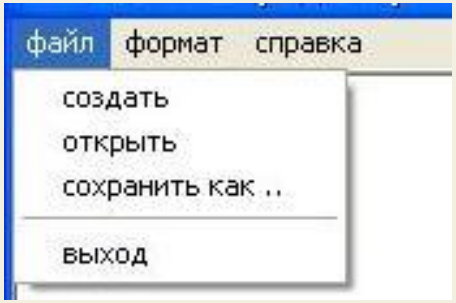
Создаем текстовый редактор

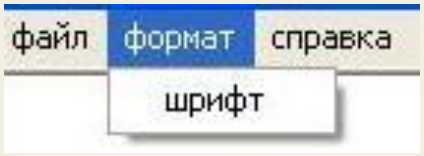
Итак, давайте попробуем **создать свой текстовый редактор**, подобный WINDOWS-кому Блокноту (немного попроще)

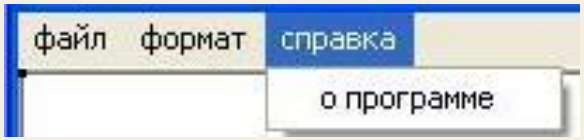


ШАГ 1 Запускаем Delphi и размещаем на форме следующие компоненты:

1. MainMenu. Создадим с помощью дизайнера систему раскрывающихся меню



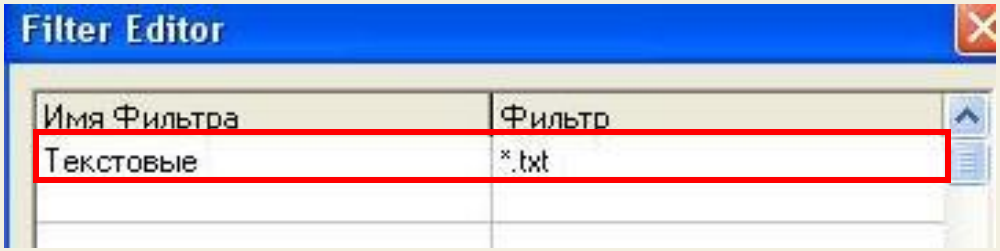


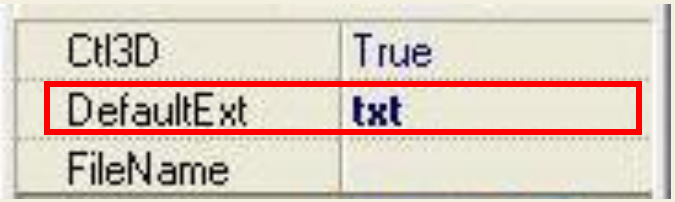


2. Memo для отображения текста. Установим свойство **Align** в **AlClient** (при этом Memo будет занимать всю клиентскую область формы, что нам и надо)

ШАГ 1 Запускаем Delphi и размещаем на форме следующие компоненты:

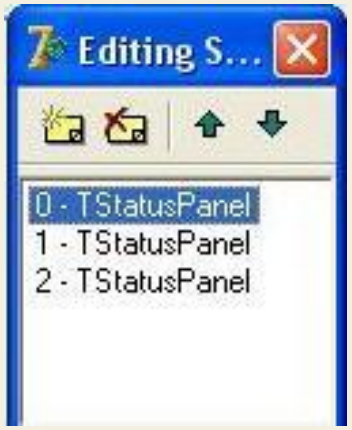
3. OpenDialog. Установим фильтр для текстовых файлов и расширение по умолчанию **txt**





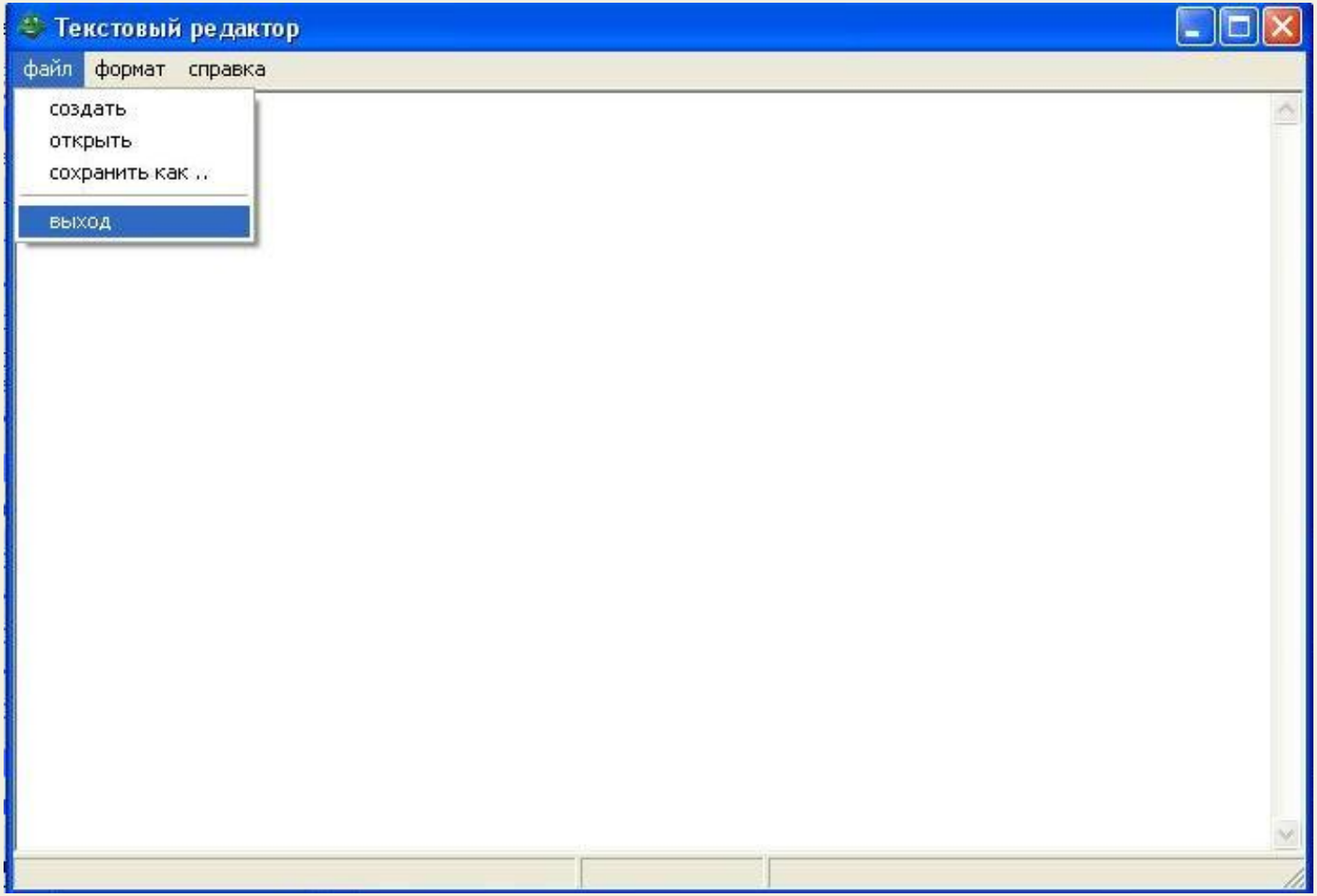
4. SaveDialog. Также установим фильтр для текстовых файлов и расширение по умолчанию **txt**

5. StatusBar. Сформируем в полосе состояния 3 панели с помощью редактора панелей



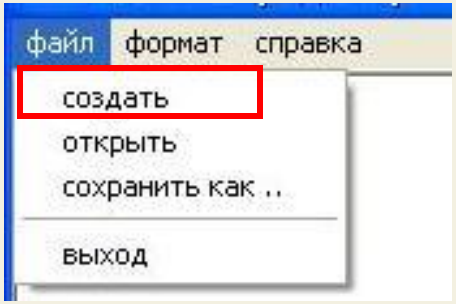
ШАГ 1 Запускаем Delphi и размещаем на форме следующие компоненты:

Если скомпилировать проект, то к нас получается примерно такая программа



ШАГ 2

Сейчас опишем события выбора разделов меню (открыть, создать, сохранить как ...)



1. **СОЗДАТЬ**. Для этого щелкнем по разделу создать и запишем код:

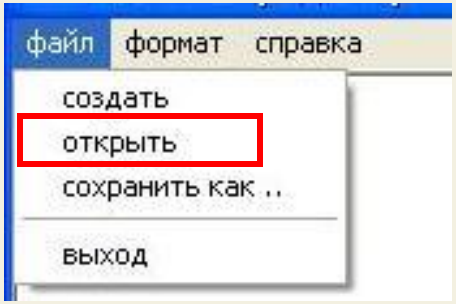
```
procedure TForm1.N2Click(Sender: TObject);
begin
 Memo1.Lines.Clear;
 Form1.Caption:=('Текстовый редактор');
end;
```

Очищаем наш **Memo**, если в нем что-то есть

В заголовке формы пишем «Текстовый редактор»

ШАГ 2

Сейчас опишем события выбора разделов меню (открыть, создать, сохранить как ...)



2. **ОТКРЫТЬ**. Для этого щелкнем по разделу создать и запишем код:

```
procedure TForm1.N3Click(Sender: TObject);
begin
 If OpenDialog1.Execute Then
 If OpenDialog1.Filename<>'' Then
  Begin
    Memo1.Lines.LoadFromFile(OpenDialog1.Filename);
    Form1.Caption:=(OpenDialog1.Filename+' - Текстовый редактор');
```

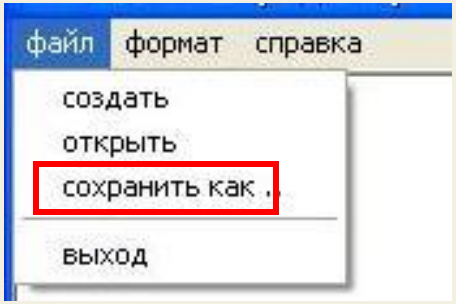
Запускаем **диалог открытия файла** и, если в нем файл выбран (<> ' '), то ...

Загружаем в наш **Memo** текст из выбранного файла

В заголовке формы записываем **имя выбранного в диалоге файла** + «Текстовый редактор»

ШАГ 2

Сейчас опишем события выбора разделов меню (открыть, создать, сохранить как ...)



3. **СОХРАНИТЬ КАК ....**

```
procedure TForm1.N4Click(Sender: TObject);
begin
  if SaveDialog1.Execute then
  if SaveDialog1.Filename<>'' then
  Begin
    Memo1.Lines.SaveToFile(SaveDialog1.Filename);
    Form1.Caption:=(SaveDialog1.Filename+' – Текстовый редактор');
```

Запускаем **диалог сохранения файла** и если файл выбран (дано имя), то ...

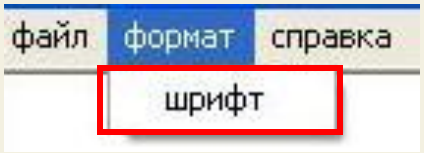
Сохраняем в выбранном файле содержимое **Memo**

В заголовке формы записываем имя выбранного (заданного нами) в диалоге файла + «Текстовый редактор»

ШАГ 2

Сейчас опишем события выбора разделов меню (открыть, создать, сохранить как ...)

3. **ФОРМАТ -> ШРИФТ ....**



Запускаем диалог выбора формата **шрифта**

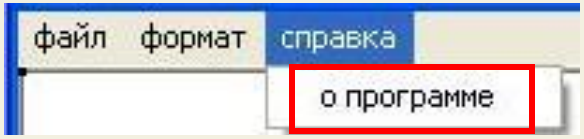
```
procedure TForm1.N8Click(Sender: TObject);
begin
form1.FontDialog1.Execute ;
form1.Memo1.Font:= form1.FontDialog1.Font
end;
```

Присваиваем шрифту **Memo** тот шрифт, который выбран нами в FontDialog

ШАГ 2

Сейчас опишем события выбора разделов меню (открыть, создать, сохранить как ...)

3. **СПРАВКА - > О ПРОГРАММЕ ….**




```
procedure TForm1.N10Click(Sender: TObject);
begin
form2.ShowModal
end;
```

Открываем в модальном режиме форму с информацией о программе

Не забудьте создать **новую форму** (Form2) и разместить на ней информацию о программе

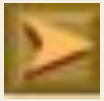
Познакомьте формы друг с другом. (Смотри предыдущие уроки о многоформенных приложениях)

ШАГ 3

В панелях **статусной строки** можно вывести нужную информацию (например дату, время, тип файла, режим работы ...) – см. предыдущие уроки

ШАГ 4

И последнее: сохраняем все, компилируем и запускаем появившийся EXE - файл

Запускаем -> 

Итак, мы создали свой текстовый редактор, который многое умеет и вполне может заменить входящий в Windows блокнот. А сейчас можете открыть MS Word и посмотреть на него глазами программиста: Да! Чрезвычайно мощный текстовый редактор (не зря в Microsoft хлеб едят ...)

На этом наш урок закончен

ИТОГИ УРОКА:

На этом уроке мы познакомились с организацией стандартных диалогов и создали свой текстовый редактор

НА СЛЕДУЮЩЕМ УРОКЕ:

ООП на Delphi – 10:

Мы рассмотрим использование в Delphi баз данных и создадим содержащее их приложение

Домнин Константин Михайлович
E – mail: kdomnin@list.ru
2006 год.