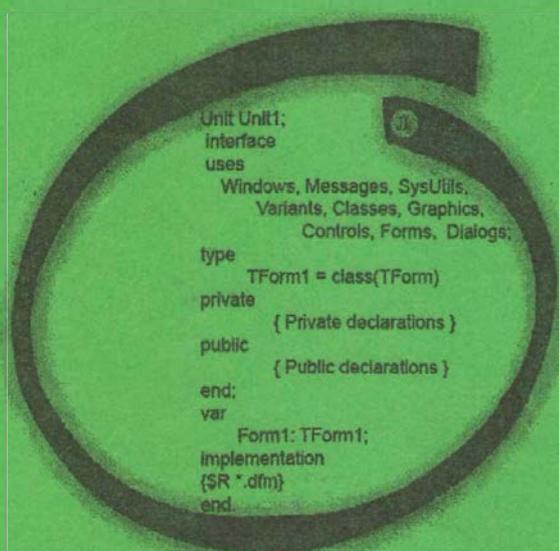


А.О. КЫБЫРАЕВ, А.Ж. КУДУЕВ

ЛАБОРАТОРНЫЙ ПРАКТИКУМ НА DELPHI

Учебно-методическое пособие



Borland Delphi



УДК 004
ББК 32.973 - 01
К- 88

Рецензенты: Сагындыков М.К., к.ф.-м.н., доцент, зав. каф. «Информатики» ОшГУ

Сатыбаев А.Дж., д.ф.-м.н., профессор, декан факультета КУУ

Кыбыраев А.О., Кудуев А.Ж.
К- 88 Лабораторный практикум на Delphi: Учебно-методическое пособие. – Ош: 2008.- 65с.

ISBN 978 - 9967- 03 - 376 - 4

В учебном пособии приведены пошаговые описания алгоритмов выполнения лабораторных работ для решения разнообразных примеров, которые демонстрируют возможности среды Delphi. Каждая лабораторная работа подкреплена программным приложением, что важно для самостоятельного изучения проектирования средствами Delphi. Для наглядности усвоения материалов, пособие снабжено рисунками объектов.

Данное пособие может использоваться преподавателями и студентами вузов при выполнении практических и лабораторных работ по программированию на языке Delphi. Оно также может быть полезным для тех, кто самостоятельно изучает этот объектно-ориентированный язык, который в настоящее время является одним из популярных языков программирования.

Рекомендовано к печати Ученым Советом ОшГУ.

К-2404090000- 07

УДК 004
ББК 32.973-01

ISBN 978-9967-03-376-4

© ОшГУ, 2008

А.О. КЫБЫРАЕВ, А.Ж. КУДУЕВ

ЛАБОРАТОРНЫЙ ПРАКТИКУМ НА DELPHI

Учебно-методическое пособие

Ош-2008

СОДЕРЖАНИЕ

Введение	5
Лабораторная работа №1. Среда программирования Delphi.	6
Лабораторная работа №2. Создание простого приложения	8
Лабораторная работа №3. Работа с кнопками.	9
Лабораторная работа №4. Управления текстом надписи.	11
Лабораторная работа №5. Программа «Шутка».	13
Лабораторная работа №6. Расчет суммы чисел.	14
Лабораторная работа №7. Работа с компонентом CheckBox.	15
Лабораторная работа №8. Разработка приложению «Светофор».	17
Лабораторная работа №9. Работа с компонентом Timer.	18
Лабораторная работа №10. Отображение геометрических фигур.	20
Лабораторная работа №11. Вставка графики в ListBox.	22
Лабораторная работа №12. Работа со списками.	25
Лабораторная работа №13. Разработка проекта «СПРАВОЧНИК ЦВЕТОВ»	27
Лабораторная работа №14. Разработка проекта цвета в формате RGB.	31
Лабораторная работа №15. Использование поля примечаний.	33
Лабораторная работа №16. Многострочное поле ввода (Мемо).	34
Лабораторная работа №17. Игра «15».	36
Лабораторная работа №18. Индикация состояния процесса.	38
Лабораторная работа №19. Компоненты управления файлами.	40
Лабораторная работа №20. Работа с меню (главное меню – MainMenu).	41
Лабораторная работа №21. Использование средств мультимедиа.	45
Лабораторная работа №22. Воспроизведение немых видео клипов - компонент Animate.	48
Лабораторная работа №23. SDI-приложения.	50
Лабораторная работа №24. MDI интерфейс.	54
Лабораторная работа №25. Базы данных в Delphi.	59
Список использованной литературы	67

ВВЕДЕНИЕ

Среди разработчиков программных продуктов под Windows в мире особой популярностью пользуется среда быстрой разработки приложений Delphi. Это популярность завоевана, прежде всего, ее простотой, легкостью в изучении и использовании.

Среда Delphi обладает практически всеми возможностями современных систем управления базами данных (СУБД). С помощью Delphi можно разрабатывать как локальные, так и удаленные базы данных.

Данное учебно-методическое пособие предназначено тем, кто только начинает изучать язык Delphi. Здесь собраны разнообразные примеры, которые не только демонстрируют возможности среды Delphi, но и знакомят с принципами работы с графикой, звуком и базами данных. Следует обратить внимание на то, что в нем описаны лабораторные работы на каждый пример и потому его можно использовать как пособие по выполнению лабораторных работ.

Авторы считают, что данное учебно-методическое пособие будет весьма полезным для широкого круга пользователей и будут благодарны всем тем лицам за ценные предложения и замечания, которые несомненно послужат усовершенствованию содержания и качества данного пособия.

ЛАБОРАТОРНАЯ РАБОТА №1

Тема: Среда программирования Delphi

Цель работы: изучить основные элементы визуальной и объектно-ориентированной среды.

Запуск программы осуществляется одним из обычных способов, характерных для Windows.

- ✚ Для этого выберите из меню **Пуск** → **Программы** → **Borland Delphi 5** → **Delphi 5**.
- ✚ Щелчком мыши по ярлычку **Delphi32**.

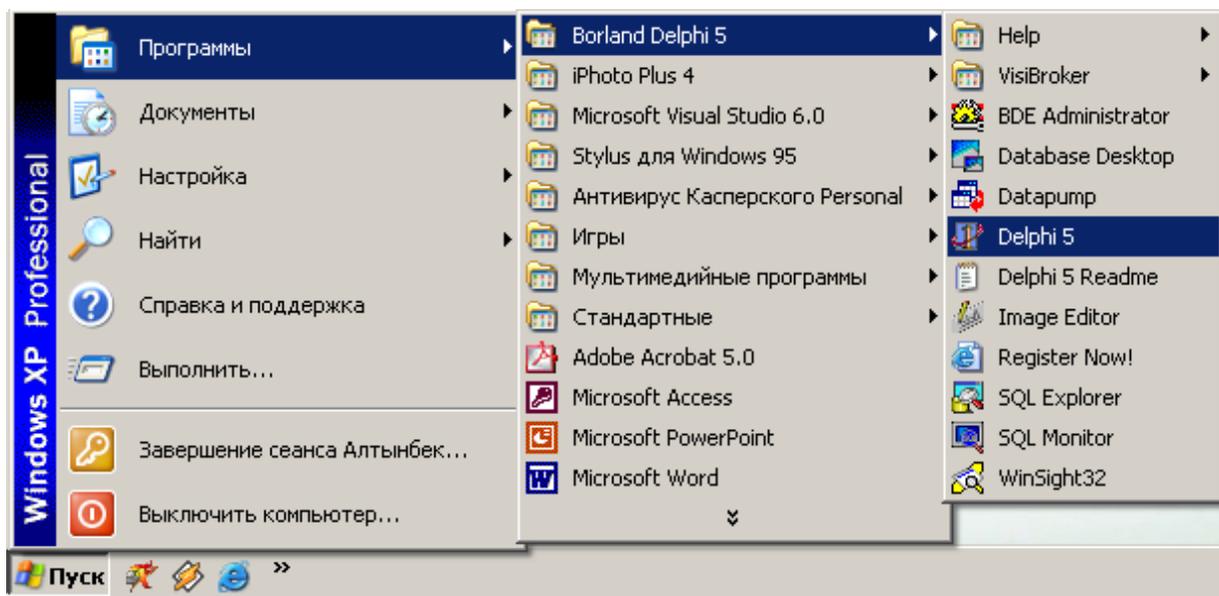


Рис. 1. Запуск Borland Delphi 5

После запуска перед вами откроется окно, показанное на рис. 2.

1. *Главное окно* всегда присутствует на экране и предназначено для управления процессом создания приложения;
2. *Главное меню* содержит все необходимые средства для управления проектом;
3. *Пиктограммы главного меню* облегчают доступ к наиболее часто применяемым командам;
4. *Окно Инспектора Объектов* предназначено для изменения свойств выбранных компонентов и состоит из двух страниц. Страница *Properties* (Свойства) предназначена для изменения необходимых свойств компонента. Страница *Events*

(События) – для определения реакции компонента на то или иное событие (например, щелчок кнопки “мыши”);

5. *Окно Редактора Кода* предназначено для просмотра, создания и редактирования текстов модулей проекта. При первоначальной загрузке в окне Редактора Кода находится текст модуля, содержащий минимальный набор операторов для нормального функционирования пустой Формы в качестве Windows-приложения. При размещении некоторого компонента в окне Формы, текст модуля автоматически дополняется необходимыми операторами.

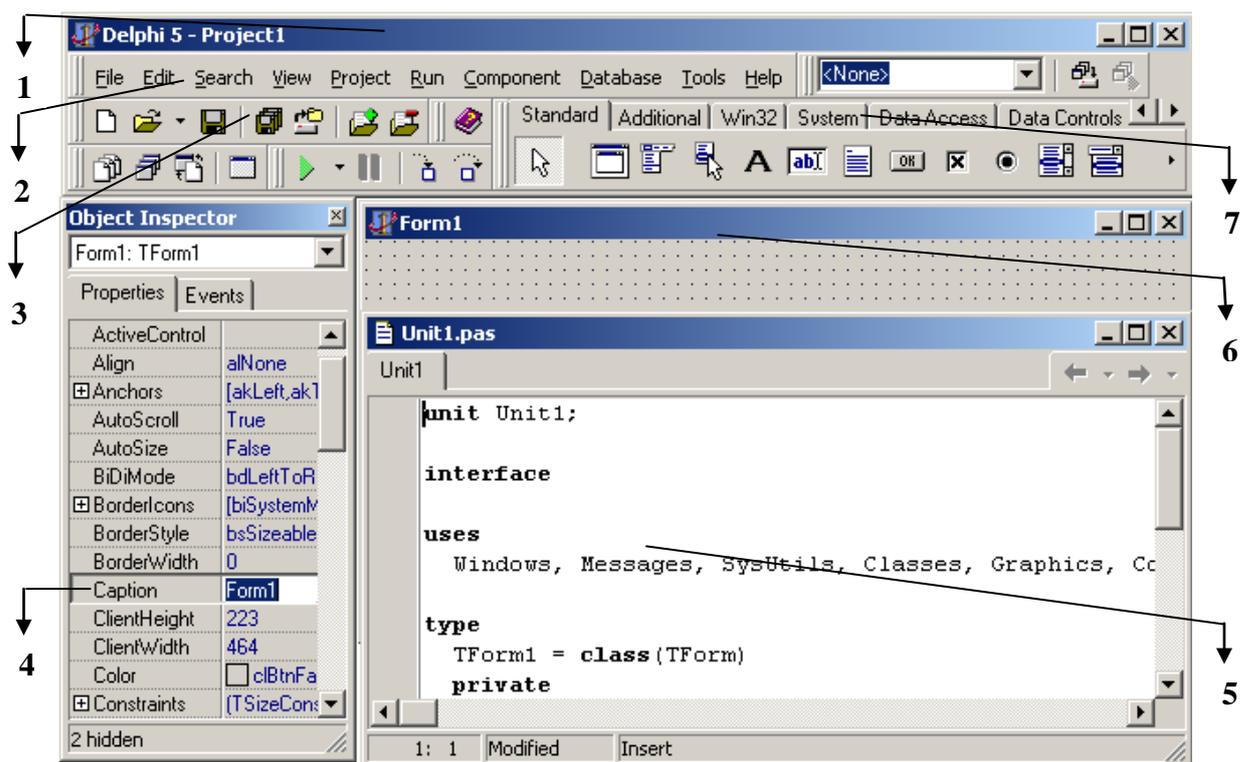


Рис. 2.

1 - Главное окно, 2 – Главное меню, 3 – Пиктограммы главного меню, 4 - Окно инспектора объектов;
5 – Окно Редактора Кода, 6 - окно пустой Формы; 7 – Палитра Компонентов.

6. *Окно Формы* представляет собой интерфейс проектируемого Windows-приложения. В это окно на этапе проектирования приложения помещаются необходимые компоненты, которые разработчик берет из Палитры Компонентов. Каждой Форме проекта соответствует модуль (**Unit**), текст которого на языке Object Pascal размещается в окне Редактора Кода;

7. *Палитра Компонентов* обеспечивает доступ к набору библиотечных программ среды DELPHI, которые описывают некоторый элемент (компонент), помещенный

программистом в окно Формы. Каждый компонент имеет определенный набор свойств, которые программист может выбирать и изменять по своему усмотрению. Например, заголовок окна, надпись на кнопке, размер, цвет и тип шрифта и др. Свойства компонентов приведены в HELP;

ЛАБОРАТОРНАЯ РАБОТА №2

Тема: Создание простого приложения

Цель работы: Уметь создавать простейшее приложение с надписями и кнопки управления формой.

Простой алгоритм создания приложения будет следующее:

1. Открыть новый проект
2. Разместить в форме следующие компоненты: метку *Label* и две кнопки *Button*.



Рис. 3.

3. Установите следующие свойства метки.

№	Свойство	Значение
1	AutoSize	False
2	Caption	Label1
3	Color	clHighlightText
4	Height	16
5	Left	9
6	Name	Label1
7	Top	16
8	Width	257

4. Выделить кнопку *Button2*, перейти в *Object Inspector* на страницу *Properties* (свойства), найти *Caption* (заголовок) и изменить заголовок *Button2* на заголовок *Выход*.

5. Перейти на страницу *Events* (события) *Object Inspector*, найти событие *OnClick*, справа от него дважды щелкнуть мышкой. Оказавшись в коде программы, точнее, в заготовке процедуры кнопки *Button2*, надо написать лишь одну команду:

```
Procedure TForm1.Button2Click(Sender: TObject);
```

```
begin
```

```
    Close;
```

```
end;
```

обязательно поставить точку с запятой после *Close*.

6. Сохранить проект под именем, например, *Unit1.pas* и *Lab_1.dpr*.
7. Запустить программу, затем закрыть окно проекта кнопкой *Выход*.
8. Выделить форму, в *Object Inspector* в свойстве *Caption* заменить слово *Form1* на **Моя первое приложения**. Это и будет заголовком основного окна программы.
9. Выделить кнопку *Button1*, найти в *Object Inspector* свойство *Caption* и заменить слово *Button1* на название кнопки *Приветствие*. При необходимости увеличить длину кнопки.
10. Перейти на страницу **Events** (события) *Object Inspector* и найти *OnClick*, справа от него дважды щелкнуть мышкой. Попав в код программы, но теперь в процедуру кнопки *Button1*, надо написать следующий код:

```
Procedure TForm1.Button1Click(Sender: TObject);
```

```
begin
```

```
    Label1.Caption:='Очен простое приложение!';
```

```
    Label1.Alignment:=taCenter;
```

```
end;
```

11. Сохранить проект окончательно, запустить и протестировать его.

ЛАБОРАТОРНАЯ РАБОТА № 3

Тема: Работа с кнопками

Цель работы: Создание Windows-приложения с заданным заголовком окна, цветом формы содержащее кнопки *Свернуть*, *Развернуть*, *Восстановить* и *Закреть*.

Алгоритм выполнения задания:

1. Создать папку для сохранения разработанных приложений.
2. Запустить *Delphi*.

3. Изменить заголовок окна формы с *Form1* на **Привет**: в окне инспектора объектов (*Object Inspector*) установить для свойства *Caption* значение **Привет**.
4. Изменить цвет формы со стандартного на другой: в окне инспектора объектов установить для свойства *Color* значение *clBackground*.
5. Добавить в форму кнопки *Button*.
6. Установить следующие свойства объектов:

Объект	Свойства	Значение
Form1	Caption	Привет
	Height	185
	Width	369
Button1	Caption	Свернуть
Button2	Caption	Развернуть
Button3	Caption	Восстановить
Button4	Caption	Выход

7. Записать код для процедуры:

```
Procedure TForm1.Button1Click(Sender: TObject);
begin
    windowstate:=wsMinimized;
end;
```

8. Самостоятельно записать код для процедур:

Procedure TForm1.Button2Click(Sender: TObject), **Procedure** TForm1.Button3Click(Sender: TObject) и **Procedure** TForm1.Button4Click(Sender: TObject).

8. Запустить приложение - меню *Run*, *Run* или *F9* или кнопка на панели инструментов.
9. Закончить работу приложения, закрыв его окно.
10. Следует выбрать команду Save Project:

 в окне диалога *Save Unit As* сохранить файл модуля <имя1>.pas

 в окне диалога *Save Project As* сохранить файл проекта <имя2>.dpr

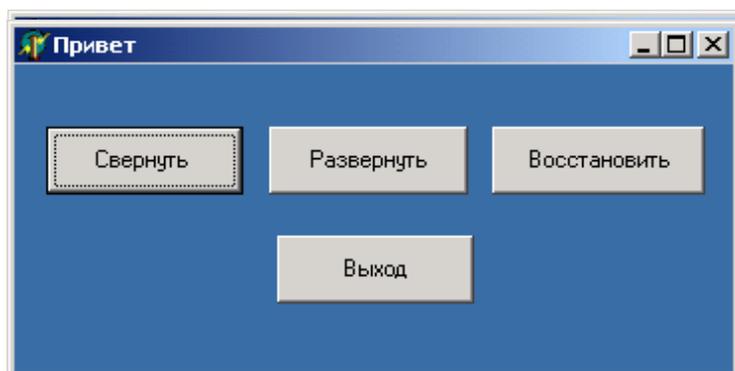


Рис. 4.

ЛАБОРАТОРНАЯ РАБОТА № 4

Тема: Управления текстом надписи.

Цель работы: Создание Windows-приложения, которое содержит текст "Моя третья приложения!" и кнопки, позволяющие изменять размер шрифта и двигать текст.

Алгоритм выполнения задания:

1. Поместить объект *Label* в окно формы *Form1*:
2. Переместить объект *Label1* на желаемое место в форме.
3. Изменить свойства объекта *Label1*:

В окне инспектора объектов (*Object Inspector*) установить следующие значения для свойств объекта:

Объект	Свойство	Значение
Label1	Caption	Моя вторая приложения
	Font	12 p., красный, жирный
	Alignment	taCenter
	Color	Желтый (Yellow)
	AutoSize	False

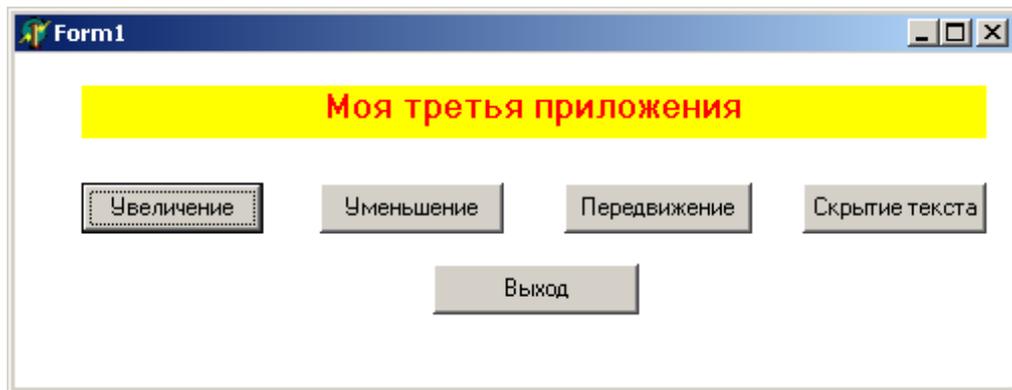


Рис. 5.

4. Выполнить приложение: меню *Run*, *Run* или *F9*.
5. Сохранить форму и проект на диске: Меню *File*, *Save All*, установить свою папку, ввести имя Лабораторная работа №4.
6. Поместить объект *Button* (командная кнопка) в окно *Form1*. Он по умолчанию получит имя *Button1*. Изменить его размеры.
7. Установить свойство *Caption* объекта *Button1* в значение "Увеличение".
8. Написать код для события *Click* на объекте *Button1*: Два раза щелкнуть по объекту *Button1* в форме Между словами *Begin* и *End* написать следующий код: `Label1.Font.Size := Label1.Font.Size + 2;`
9. Выполнить программу. Обратит внимание на то, что происходит при нажатии кнопки с надписью "Увеличение".
10. Сохранить форму и проект на диске: Меню *File*, *Save*.
11. Создать объект "командная кнопка" для уменьшения размера шрифта в тексте.
12. Создать объект "командная кнопка" для того, чтобы двигать текст.
Код: `Label1.Left := Label1.Left + 10;`
`Label1.Top := Label1.Top + 10;`
13. Создать объект "командная кнопка" для того, чтобы сделать текст невидимым.
Код: `Label1.visible := false;`
14. Создать объект "командная кнопка" для выхода из работы программы.
Код: `Close;`
15. Сохранить форму и проект.

ЛАБОРАТОРНАЯ РАБОТА № 5

Тема: Программа «Шутка»

Цели работы: Составить программу, которая при выборе кнопок, внутри окна она пошутила над вами. Точнее говоря, вы немножко развеселились.

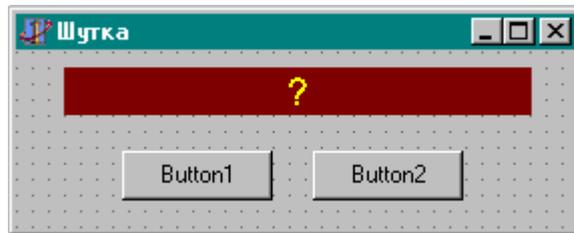


Рис. 6.

Для этого первоначально сделаете следующие шаги:

1. Поместить компоненты *Label* и *Button* в форму в соответствии с рисунком
2. Установить следующие свойства объектов

Объект	Свойство	Значение
Form1	Caption	Шутка
Label1	Caption	?
	Color	clMaroon
	Font.Size	18, Жирный
	Font.Color	Синий
	Alignment	taCenter
	AutoSize	False

3. Установить свойство объекта *Button2*: **DragMode = dmAutomatic**
4. Записать код для обработки события *MouseMove* на объекте *Button2*:

```
Procedure TForm1.Button2MouseMove(Sender: TObject; Shift: TShiftState;  
X,Y: Integer);
```

```
begin
```

```
    Button2.Left := Button2.Left+10;
```

```
    Button2.Top := Button2.Top+10;
```

```
end;
```

5. Записать код для обработки события *Click* на объекте *Button1*:

```
Procedure TForm1.Button1Click(Sender: TObject);
```

```
begin
```

```
    Label1.Caption := 'Мы были в этом уверены!'
```

end;

5. Выполнить программу.

6. Развитие задачи:

Задание. Изменить программу, т.е. чтобы при подводе курсора мыши к кнопке *Button2* кнопка исчезала, а при отводе курсора - появлялась.

ЛАБОРАТОРНАЯ РАБОТА № 6

Тема: **Расчет суммы чисел**

Цель работы: Составление проекта для суммирования двух чисел.

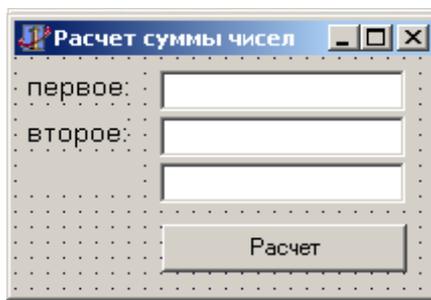


Рис. 7.

Алгоритм выполнения задания:

1. Как показано рис. 7. сначала на форме нужно разместить два надписи (объект *Label*) с задаваемыми свойствами *Caption*, и третья надпись (объект **Label**) с пустой *Caption* – для отображения суммы.

2. Определить две строки ввода для суммируемых чисел (против меток "первое" и "второе") и третья строка служит для показания суммы двух чисел. Все текстовые поля свойствами *Text* должны быть пустыми, т. е. удаленными.

3. И одну кнопку "**Расчет**" для запуска процедуры суммирования после ввода чисел.

4. Изменить заголовок окна формы с *Form1* на «**Расчет суммы чисел**»: в окне инспектора объектов (*Object Inspector*) установить для свойства *Caption* значение «**Расчет суммы чисел**».

5. После двойного щелчка на кнопке «**Расчет**» можно заполнить шаблон процедуры реакции на нажатие этой кнопки.

```
Procedure TForm1.Button1Click(Sender: TObject);
```

```
Var a,b,c: real;
```

```
s: string; code: integer;
```

begin

```
{ ввод данных из полей редактирования }  
val(edit1.text,a,code);  
val(edit2.text,b,code); c:=a+b;  
str(c:-10:4,s); { перевод числа в строку }  
edit3.text:=s; label3.caption:='сумма';  
end;
```

end.

6. Сохранить проект и дайте команду на выполнение. Ваше приложение выглядит следующим образом:

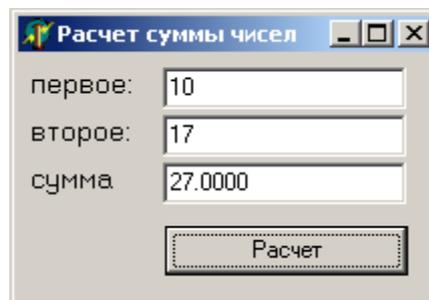


Рис. 8.

ЛАБОРАТОРНАЯ РАБОТА № 7

Тема: Работа с компонентом CheckBox

Цель работы: Теперь мы переходим к рассмотрению компонента *CheckBox*. Как всегда, давай создадим маленькую программу, которая будет использовать этот компонент.

Алгоритм выполнения задания:

1. Создай новое приложение, брось на него одну кнопку и два компонента *TCheckBox* .
2. Первому компоненту дай заголовок (Caption) равным «Разрешить закрытие программы» и имя (Name) равным «CloseCheckBox». Второму компоненту дай

заголовок (Caption) равным «Отключить кнопку» и имя (Name) равным «EnableButtonCheckBox». Кнопке мы дали имя «MyButton».

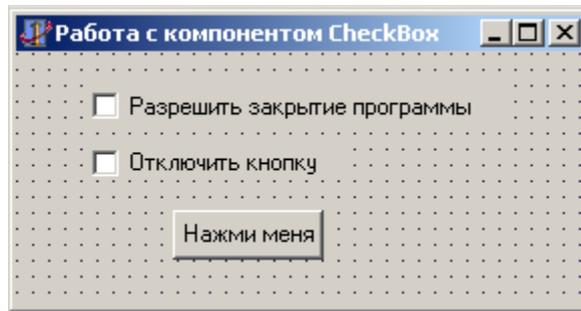


Рис. 9.

3. Создай обработчик события *OnClick* для компонента *EnableButtonCheckBox* (это второй *CheckBox*). В нём напиши следующее:

```
Procedure TForm1.EnableButtonCheckBoxClick(Sender: TObject);
```

```
begin
```

```
    MyButton.Enabled:=not EnableButtonCheckBox.Checked;
```

```
end;
```

Здесь мы присваивали свойству *Enabled* нашей кнопки значение *not EnableButtonCheckBox.Checked*. Что это значит? Свойство *Checked* компонента *EnableButtonCheckBox* показывает: стоит ли галочка на это *CheckBox*-е. Если да, то свойство *Checked* равно *True*, иначе *False*. Оператор *not* меняет это состояние на противоположное. Это значит, что если свойство *Checked* было равно *True*, то в *MyButton.Enabled* будет присвоено противоположное *False*.

Можешь попробовать запустить пример, и посмотреть что происходит. Когда тыставишь галочку напротив «Отключить кнопку», свойство *Checked* этого компонента меняется на *True*. Срабатывает событие *OnClick* и в свойство *Enabled* кнопки присваивается значение свойства *Checked* компонента *CheckBox*, изменённое на противоположное, т.е. *False*. А когда свойство *Enabled* кнопки равно *False* она становится недоступной.

Чтобы окончательно разобраться с работой примера понажимай на *EnableButtonCheckBox* при запущенной программе. Потом попробуй убрать из исходного кода оператор *not* и снова запусти программу.

4. Теперь давай создадим обработчик *OnClick* для кнопки. В нём напиши следующее:

```
Procedure TForm1.MyButtonClick(Sender: TObject);
```

```
begin
```

if AllowCloseCheckBox.Checked **then** Close;

end;

Здесь мы проверяем, если свойство *Checked* компонента *CloseCheckBox* (первый *CheckBox* на форме) равно *True*, то закрыть программу (выполнить метод *Close*). Иначе ничего не произойдет.

В принципе, это всё, что нужно знать и как можно использовать компонент *TComboBox*. Он достаточно простой, но в большинстве программ незаменим.

5. Теперь запускайте приложения и смотрите.

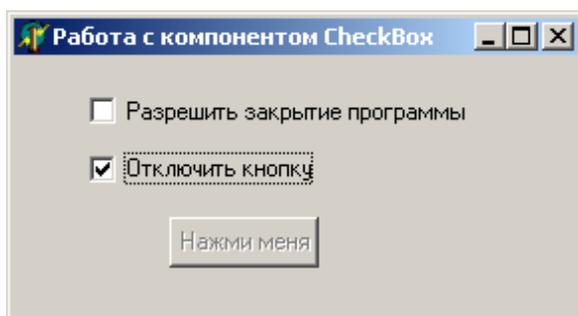


Рис. 10.

Лабораторная работа № 8

Тема: Разработка приложения «Светофор».

Цели работы: Создание Windows - приложения, в котором при щелчке на радиокнопке с названием цвета на светофоре загорается соответствующий цвет согласно рис.11.

Алгоритм выполнения задания:

1. Поместить компоненты *Label*, *Panel*, *GroupBox*, *RadioButton*, *Button* (страница **Standard**) в форму.
2. Установить следующие свойства объектов, используя, Инспектор объектов:

Label1	Caption	Светофор
	AutoSize	False
Panel1,2,3	Caption	
GroupBox1	Caption	Цвет
RadioButton1	Caption	Красный
RadioButton2	Caption	Желтый
RadioButton3	Caption	Зеленый
Button1	Caption	Выход

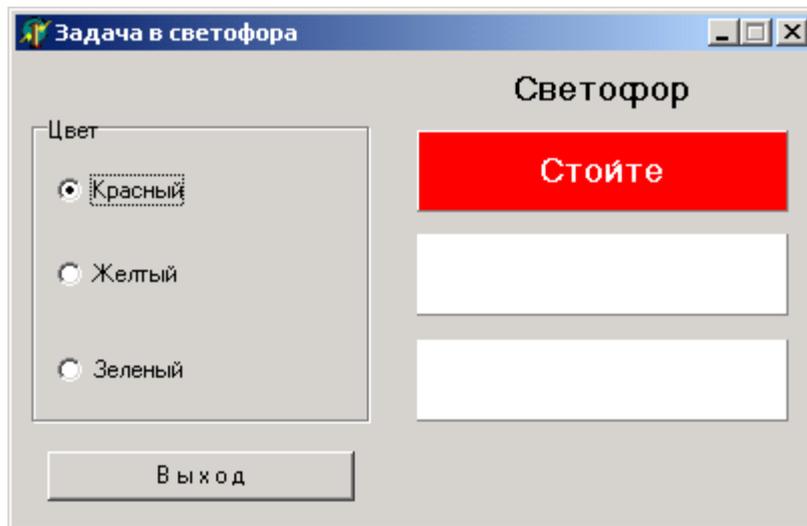


Рис. 11

3. Записать код для процедуры обработки события *Click* (щелчок мыши) на объекте *RadioButton1*:

Procedure TForm1.RadioButton1Click(Sender: TObject);

begin

Panel1.Color := clRed;

Panel2.Color := clWhite;

Panel3.Color := clWhite;

Panel1.Font.Style :=[fsbold];

end;

4. Самостоятельно записать код для процедур: *TForm1.RadioButton2Click*, *TForm1.RadioButton3Click* и *TForm1.Button1Click*.

5. Добавить печать информации "Стойте", "Внимание", "Идите" на панели с соответствующим сигналом белым цветом шрифта жирным начертанием 12п.

ЛАБОРАТОРНАЯ РАБОТА № 9

Тема: Работа с компонентом **Timer**

Цели работы: Создание Windows - приложения, в котором работают цифровые часы с разной скоростью.

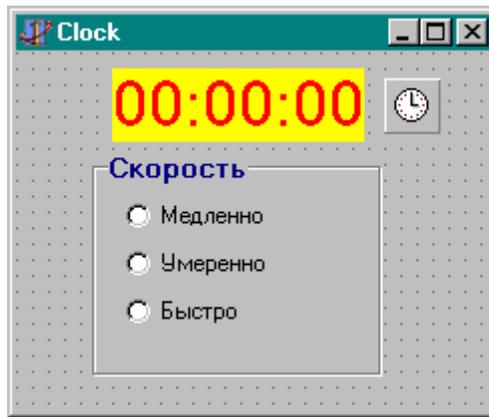


Рис. 12.

Алгоритм выполнения задания:

1. Поместить компоненты *Label* (вкладка **Standard**) и *Timer* (вкладка **System**) в форму *Form1*.

2. Установить следующие свойства объектов

Объект	Свойство	Значение
Form1	Name	Clock
Label1	Caption	00:00:00
	Color	clYellow
	Font.Size	24, Жирный
	Font.Color	Красный

3. Записать код обновления времени для процедуры *TClock.Timer1Timer: Label1.Caption:=TimeToStr(Time);*
4. Добавление кнопок регулирования скорости обновления времени.
 - 4.1 . Добавить в форму компоненты *GroupBox* и *RadioButton*.
 - 4.2 . Установить следующие свойства объектов:

Объект	Свойство	Значение
GroupBox1	Caption	Скорость
	Font.Size	10, Жирный
	Font.Color	Темно-Синий
RadioButton1	Caption	Быстро
RadioButton2	Caption	Медленно
RadioButton3	Caption	Умеренно

4.3 . Записать код для процедуры TForm1.RadioButton1Click: Timer1.Interval := 1000;

Самостоятельно записать код для процедур: *TForm1.RadioButton2Click (2000)* и *TForm1.RadioButton3Click (3000)*

ЛАБОРАТОРНАЯ РАБОТА № 10

Тема: Отображение геометрических фигур

Цель работы: Создание простейшего Windows – приложения с выпадающим списком и полосы прокрутки, которые влияют на отображаемый рисунок.

1. Для начала создаете стандартный проект.
2. Поместите четыре компонента *Shape*, *BitBtn*(Вкладка *Additional*), *ScrollBar* и *ComboBox*(Вкладка *Standard*) в форму. Установите следующие их свойства:

Объект	Свойства	Значение
Shape1	Shape	stRectangle
	Height	153
	Left	144
	Top	97
	Width	241
ScrollBar1	Height	17
	Kind	sbHorizontal
	Left	9
	Top	353
	Width	472
ScrollBar2	Height	360
	Kind	sbVertical
	Left	480
	Top	7
	Width	16
ComboBox1	Height	20
	Hint	Выберите фигуры
	Left	6
	ShowHint	True

	Top	12
	Width	147
	Items	Прямоугольник Квадрат Круглый Прямоугольник Круглый Квадрат Эллипс Круг
BitBtn1	Caption	Цвет фигуры
BitBtn2	Caption	Цвет формы
BitBtn3	Caption	Выход
	Kind	bkClose

3. Создай обработчик события *On Change*:

```
Procedure TForm1.ComboBox1Change(Sender: TObject);
```

```
begin
```

```
    Shape1.Shape:=TShapeType(ComboBox1.ItemIndex);
```

```
end;
```

4. Еще добавьте в форму компонент *ColorDialog* (*Вкладка Dialogs*) и присвойте ему имя *Open_Color*.

5. Записать код для процедуры обработки события *Click* (щелчок мыши) на объекте *BitBtn1*:

```
Procedure TForm1.BitBtn1Click(Sender: TObject);
```

```
begin
```

```
    if Open_Color.Execute then
```

```
        Shape1.Brush.Color:=Open_Color.Color;
```

```
end;
```

6. Записать код для процедуры:

```
Procedure TForm1.ScrollBar2Change(Sender: TObject);
```

```
begin
```

```
    Shape1.Height:=Scrollbar2.Position*2;
```

```
end;
```

7. Самостоятельно записать код для процедур:

Procedure TForm1.BitBtn2Click(Sender: TObject);

8. Сохранить форму и проект на диске. Внешний вид проекта выглядит следующим образом:

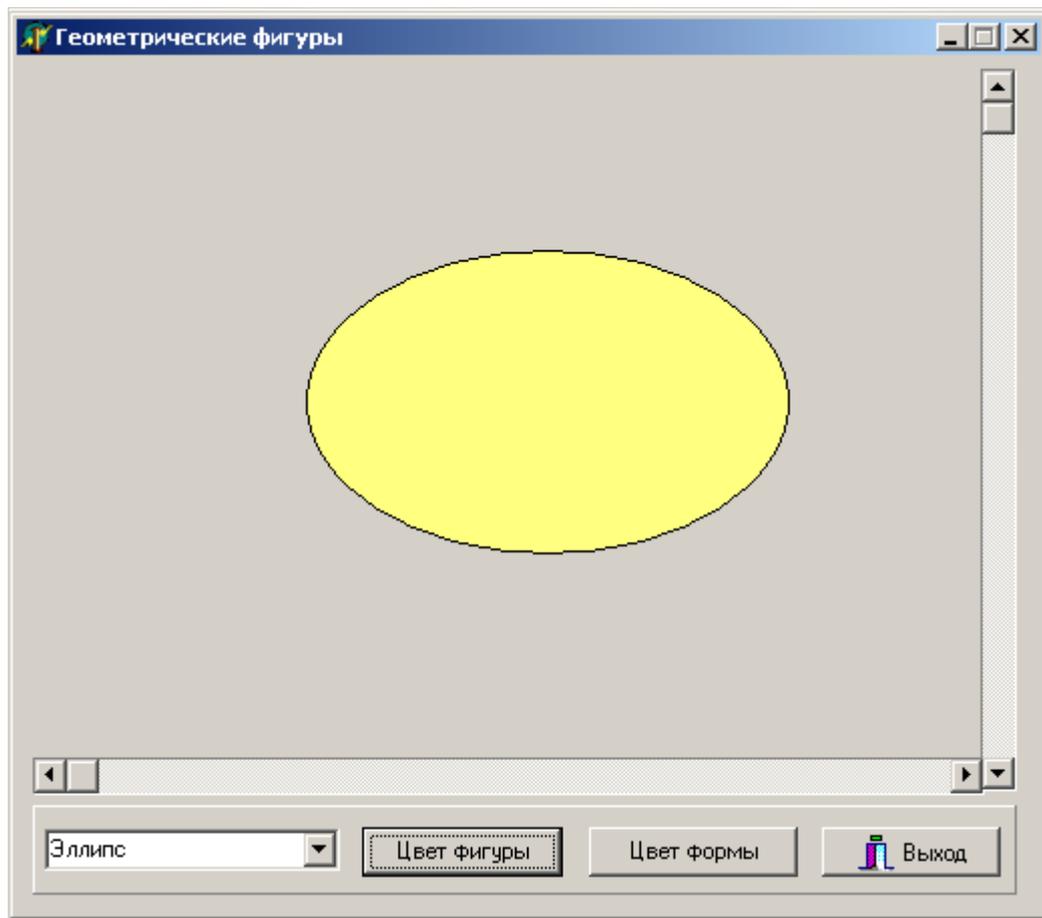


Рис. 13.

ЛАБОРАТОРНАЯ РАБОТА № 11

Тема: Вставка графики в ListBox

Цель работы: Отображение названия файлов формата *.bmp из какой-либо директории вместе с их картинками на *ListBox*.

У класса *TListBox* (и *TComboBox* тоже) есть свойство *Style*, определяющее порядок рисования объекта. По-умолчанию оно установлено в *lbStandard* и за внешний вид объекта отвечает **Windows**. Если установить это значение в *lbOwnerDrawFixed* или *lbOwnerDrawVariable*, то можно несколько разнообразить внешний вид объекта.

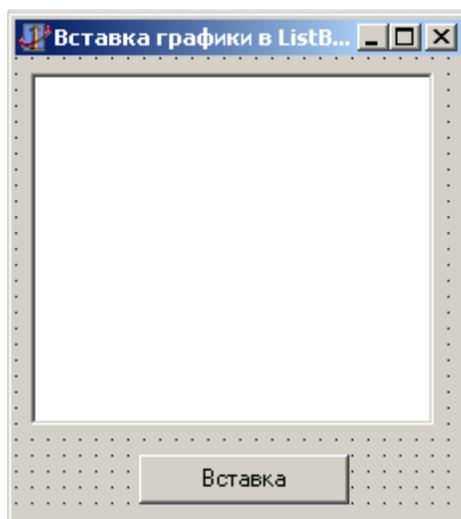


Рис. 14

Для этого сделайте следующие шаги:

1. Поместить компоненты *ListBox* и *Button* в форму.
2. Установить следующие свойства объектов:

Объект	Свойство	Значение
Form1	Caption	Вставка графики в ListBox
	Height	305
	Name	LBox
	Width	276
	Position	poScreenCenter
ListBox	Height	217
	Left	8
	Style	lbStandard
	Top	8
	Width	250
Button	Caption	Вставка
	Height	25
	Left	72
	Top	240
	Width	105

3. Записать код для обработки события *Click* на объекте *Button1*:

Procedure TLBox.Button1Click(Sender: TObject);

var

```

    s : String;
begin
    s:='c:\windows\*.bmp'#0;
    ListBox1.Perform(LB_DIR, DDL_READWRITE, Longint(@s[1]));
end;

```

Здесь мы указали **ListBox**, какие файлы требуется отображать. Далее, как уже было сказано, свойство *Style* нужно установить в *lbOwnerDrawFixed* и создать обработчик события *OnDrawItem* (объект **ListBox1**):

```

Procedure TLBox.ListBox1DrawItem(Control: TWinControl; Index: Integer;
                                   Rect: TRect; State: TOwnerDrawState);

var
    Bitmap: TBitmap;
    Offset: Integer;
    BMPRect : TRect;

begin
    with (Control as TListBox).Canvas do
begin
        FillRect(Rect);
        Bitmap:=TBitMap.Create;
        Bitmap.LoadFromFile('c:\windows\'+ListBox1.Items[Index]);
        if Bitmap <> nil then begin { вычисляем квадрат для показа картинки }
            BMPRect:=Bounds(Rect.Left + 2, Rect.Top + 2, Rect.Bottom-Rect.Top-2,
                Rect.Bottom-Rect.Top-2);
            { рисуем картинку }
                StretchDraw(BMPRect, BitMap);
                Offset := Rect.Bottom-Rect.Top + 6;
            end;
            { ВЫВОДИМ ТЕКСТ }
                TextOut(Rect.Left+Offset,Rect.Top,ListBox1.Items[Index]);
                Bitmap.Free;
            end;
        end;
    end;

```

4. Сохраните проект и дайте команду на выполнение и смотрите Рис.5.

Примечание: Чтобы картинки получились по больше, значение свойства *ItemHeight* можно увеличить.

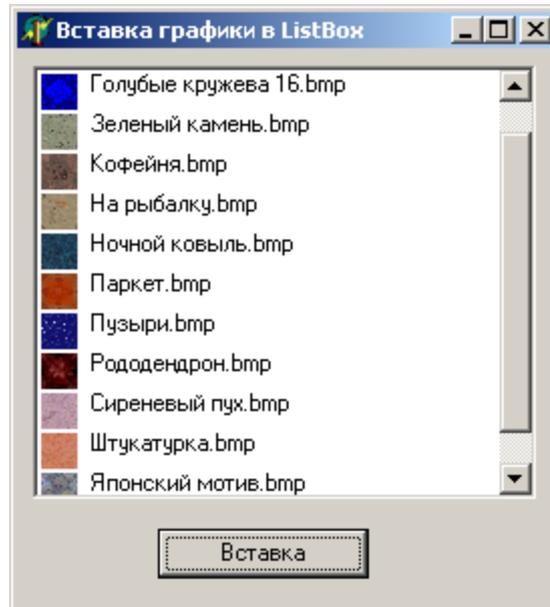


Рис. 15.

ЛАБОРАТОРНАЯ РАБОТА № 12

Тема: Работа со списками.

Цель работы: Создадим Windows-приложение, позволяющее вводить имена и строить из них список.

Предоставим пользователю возможность добавлять имена в список, сортировать список и очищать окно списка. Для создания этой приложении, мы предлагаем несколько способов:

1 - способ: Необходимо разместить в форме

- ✚ компонент управления редактированием (*Edit1*) - для ввода нового имени
- ✚ окно списка (*ListBox1*) - для вывода списка
- ✚ три кнопки (*Button1, Button2, Button3*):
 - первая - для добавления нового имени к списку;
 - вторая - для сортировки списка по алфавиту
 - третья - для очистки окна списка

2 - способ: Необходимо изменить размеры элементов формы, а также произвести выравнивание

Для выравнивания кнопок объедините их в группу, а затем выберите команду Выравнивание (*Align*) из меню *Edit* или из контекстного меню. Следует разместить кнопки с выравниванием по левому краю (*Left sides*), а также с равными интервалами (*Vertical - Space equally*).

3 - способ: Необходимо изменить свойства объектов

№	Объект	Свойство	Значение
1	Form1	Caption	Демонстрационное окно списка
2	Edit1	Text	
3	Button1	Caption	Добавить
4	Button2	Caption	Сортировать
5	Button3	Caption	Очистить

4 - способ: Необходимо создать обработчики некоторых событий

При нажатии на кнопку Добавить приложение должно брать текст из элемента правления редактированием и добавлять его в окно списка, при этом очищая элемент управления редактированием.

Обработчик события *OnClick* кнопки **Добавить**:

```
ListBox1.Items.Add (Edit1.Text);  
Edit1.Text := '';
```

При нажатии на кнопку Сортировать приложение должно отсортировать список и перерисовать содержимое окна списка

Обработчик события *OnClick* кнопки **Сортировать**:

```
ListBox1.Sorted :=True;
```

При нажатии на кнопку Очистить приложение должно очистить окно списка.

Обработчик события *OnClick* кнопки **Очистить**:

```
ListBox1.Clear;  
ListBox1.Sorted :=False;
```

5 - способ: Необходимо сохранить проект.

Следует выбрать команду *Save Project*:

- в окне диалога *Save Unit As* сохранить файл модуля <имя1>.pas
- в окне диалога *Save Project As* сохранить файл проекта <имя2>.dpr

б - способ: Запустить приложение - меню *Run*, *Run* или *F9* или кнопка на панели инструментов.

После выполнение приложений ваш проект, возможно, выглядит таким образом:

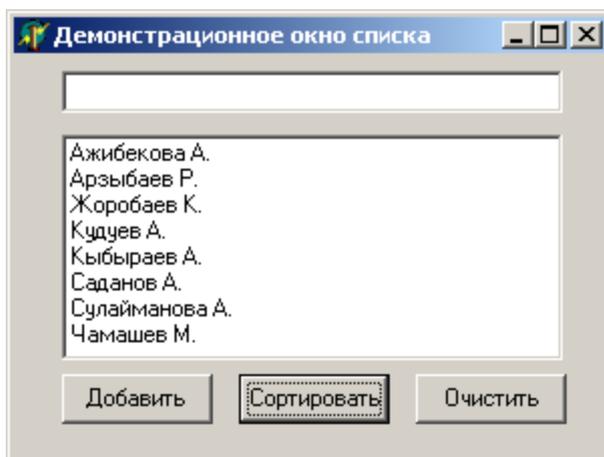


рис. 16.

ЛАБОРАТОРНАЯ РАБОТА № 13

Тема: Разработка проекта «СПРАВОЧНИК ЦВЕТОВ»

Цель работы: После запуска программы пользователь выбирает с помощью мышки или стрелок название цвета и нажимает клавишу *Enter*. На экране появляется название цвета и код в формате **RGB**. Программа заканчивает свою работу по нажатию клавиши *Выход* (см.Рис.17.).

Алгоритм выполнения задания:

1. Выберите команду *File* → *New Application*, и появится пустое приложение.
2. Установите следующие свойства форм.

№	Свойство	Значение	
1	BorderIcons	biMaximize	False
2	BorderStyle		bsSingle
3	Caption		Список цветов
4	Height		310
5	Name		LoadclRGB
6	Position		poScreenCenter
7	Width		415

3. Поместите компоненты *ListBox*, *Panel*, *Button*, *Label* (вкладка **Standard**) и *BitBtn*(вкладка **Additional**) в форму *Form1*. Установите следующие их свойства:

№	Объект	Свойство	Значение
1	ListBox1	Height	217
		Hint	Список цветов
		Left	8
		Name	ListBox1
		ShowHint	True
		Top	8
		Width	185
2	Label1	Alignment	taCenter
		Caption	Справочник записи цвета в формате RGB
		Height	32
		Left	208
		Name	Label1
		Top	8
		Width	185
3	Panel1	Caption	-
		Height	130
		Left	208
		Name	Panel1
		Top	56
		Width	185
4	Label2	Caption	-
		Height	32
		Left	208
		Name	Label2
		Top	200
		Width	185

5	Button1	Caption	Получить список цветов
		Height	30
		Left	8
		Name	LoadListColor
		Top	240
		Width	185
6	BitBtn1	Height	30
		Kind	bkClose
		Left	208
		Caption	&Выход
		Name	BitBtn1
		Top	240
		Width	185

4. Теперь самое время сохранить проект, выбрав в меню команду *File* → *Save Project As*. Сохраните *Unit1* как *clRGB*, а проект - как *RGB*.

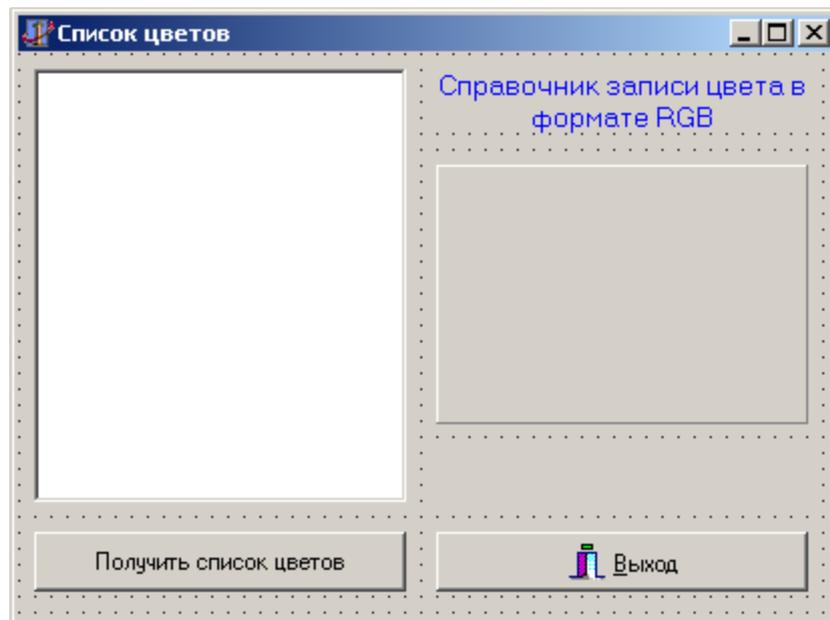


Рис. 17.

5. Теперь, после создания приложения (интерфейса), перейдем к написанию исходного текста вашего приложения.
6. В редакторе кода после строки *Private* (Частные декларации) объявляем следующую процедуру:

Private

```
{ Private declarations }
```

Procedure GetOneColor (const S:String);

7. Дважды щелкните на компоненте *ListBox1*, и *Delphi* выведет окно редактора и автоматически создаст обработчик события *OnClick*. Введите код:

Procedure TLoadclRGB.ListBox1Click(Sender: TObject);

begin

with ListBox1 **do begin**

Panel1.Caption:=Items[Index];

Panel1.Color:=StringToColor(Items[Index]);

Label2.Caption:=IntToStr(ColorToRGB(StringToColor(Items[Index])));

end;

end;

8. Записать код для процедуры обработки события *OnClick* (щелчок мыши) на объекте *Button1(LoadListColor)*:

Procedure TLoadclRGB.LoadListColorClick(Sender: TObject);

begin

GetColorValues (GetOneColor);

end;

9. Компилируйте, запускайте и смотрите.

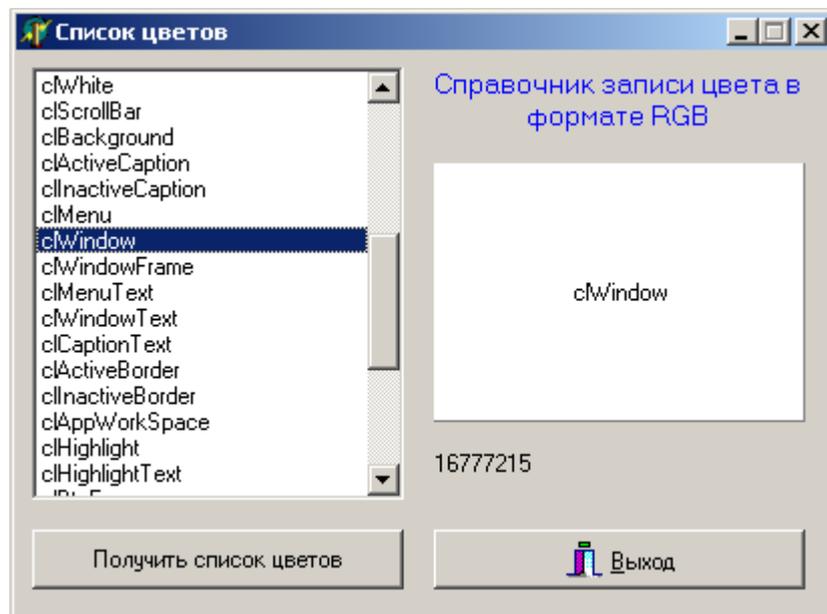


Рис. 18.

ЛАБОРАТОРНАЯ РАБОТА № 14

Тема: Разработка проекта цвета в формате RGB

Цель работы: Создать программу, с помощью которой пользователь мог бы увидеть в зависимости от значений насыщенности красного, зеленого и синего результирующий цвет.

При перемещении бегунка *ScrollBar* генерируется событие *OnChange* (при попытке перемещения – *OnScroll*). При этом можно получать и обрабатывать данные как свойство *Position*. Во время работы программы значения свойств *Position*, *Min*, *Max* можно задавать с помощью метода *SetParams*.

Алгоритм выполнения задания:

1. Открыть новый проект.
2. Поместить компоненты *Label*, *Panel*, *ScrollBar* (вкладка **Standard**) в соответствии с рисунком.

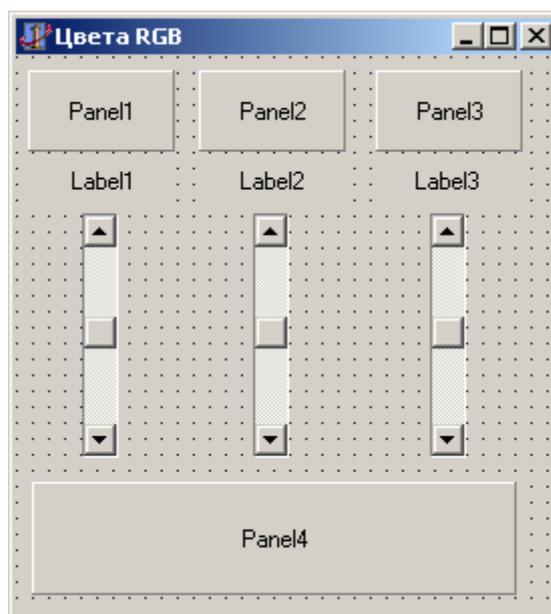


Рис.19.

3. Сохранить проект под именами, например, *Unit1.pas* и *Lab_RGB.dpr*.
4. Установить следующие свойства объектов:

Объект	Свойство	Значение
Form1	Name	ColorRGB
	Caption	Цвета RGB
ScrollBar1	Name	RedBar
ScrollBar2	Name	GreenBar

ScrollBar3	Name	BlueBar
Panel1	Caption	Красный
Panel2	Caption	Зеленый
Panel3	Caption	Синий

5. Полоса прокрутки *ScrollBar* может быть горизонтальной (по умолчанию) или вертикальной. Это определяется свойством **Kind**. В нашем случае используется вертикальная полоса прокрутки.

6. Записать код для обработки события *OnChange* на объекте *ScrollBar1*:

```
Procedure TForm1.RedBarChange(Sender: TObject);
```

```
begin
```

```
    Panel1.Color:= TColorRef(RGB(RedBar.Position,0,0));
```

```
    Label1.Caption:=IntToStr(RedBar.Position);
```

```
    Panel4.Color:= TcolorRef(RGB (RedBar.Position, GreenBar.Position,  
    BlueBar.Position));
```

```
end;
```

7. Аналогично задайте значения для *ScrollBar2* и *ScrollBar3*, проследите за правильность записи параметров в функции *RGB* и *IntToStr*.

8. Сохранить проект окончательно, запустить и протестировать его.

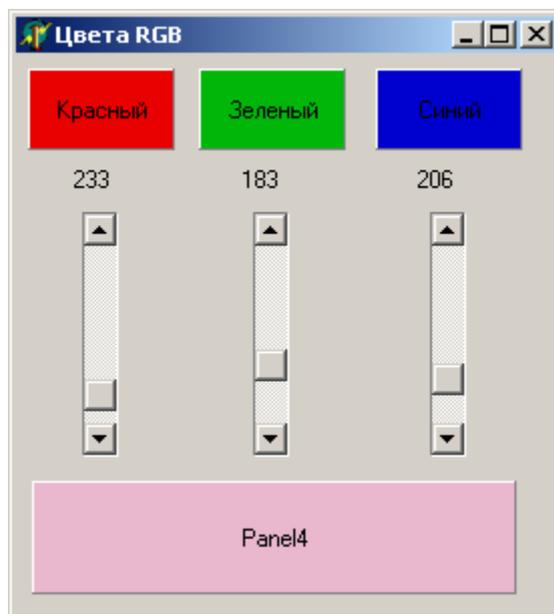


Рис. 20.

ЛАБОРАТОРНАЯ РАБОТА № 15

Тема: Использование поля примечаний.

Цели работы: Познакомиться с многострочными компонентами ввода информации.

Примечание (визуальный компонент **Мемо**) может включать множество строк в отличие от строки ввода *Edit*, то есть подразумевает работу с большими текстами. В *Memo* можно переносить слова, сохранять в *Clipboard* фрагменты текста и восстанавливать их, выполнять другие базовые функции текстового редактора с ограничением на объем текста 32Кб. Свойства *BorderStyle*, *ReadOnly*, *HideSelection*, *MaxLenght* – те же, что и у строки ввода.

Специфические свойства:

- ✚ *Lines* – содержимое *Memo* как набор строк;
- ✚ *Align* – заполнение пространства формы по краям;
- ✚ *AlingMent* – выравнивание текста внутри формы;
- ✚ *ScrollBars* – наличие полос прокрутки содержимого поля;
- ✚ *WordWrap* – автоперенос текста от правого края.

Для удобства работы с полем примечаний как с текстовым редактором имеются свойства *WantTabs* и *WantReturns*, которые "восстанавливают в правах" функции клавиш *Enter* и *Tab* применительно к работе с текстом.

Свойство **Lines** можно не только определять, но и загружать из файла и выводить в файл. Пример:

Разместим на форме *Memo* с заголовком "Редактор файла *sum.pas*". Установим свойства *ScrollBars=ssBoth*, *Align=alClient* (полное заполнение формы). На событие *OnCreate* формы введем:

```
Memol.Lines.LoadFromFile('d:\FMIT\sum.pas');  
Memol.Lines.SaveToFile('sum.bak');
```

На событие *OnCloseQuery* формы введем:

```
Memol.Lines.SaveToFile('sum.pas');
```

При этом получим простой текстовый редактор файла *file1.pas* с записью резервной копии в файл *sum.bak* (см.Рис.21).

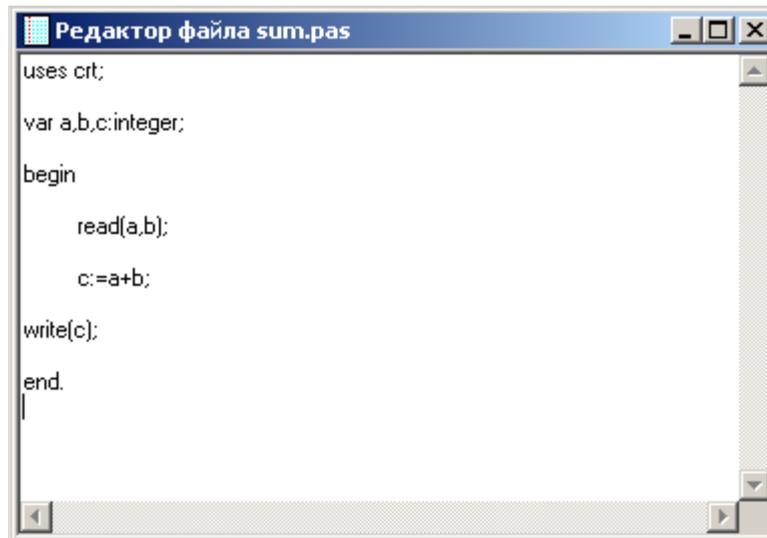


рис. 21.

ЛАБОРАТОРНАЯ РАБОТА № 16

Тема: Многострочное поле ввода (Мемо)

Цель работы: Создание текстового редактора, в который можно загрузить файл, отредактировать его и сохранить.

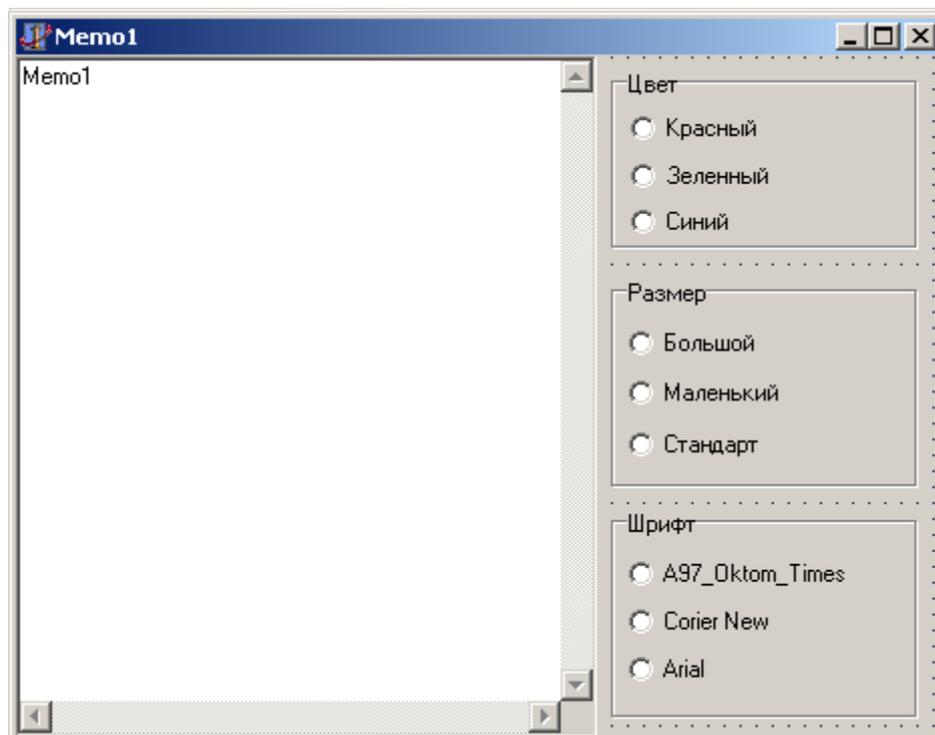


Рис. 22.

Алгоритм выполнения задания:

1. Вызвать текстовый редактор **Блокнот** и создать в нем текстовый файл *Автобиография.txt* с содержанием:

Button	Кнопка
RadioButton	Переключатель
Label	Надпись
Edit	строка текста
Memo	редактор текста

2. Сохранить файл в папку **МЕМО**.

3. Начать новый проект и сразу сохранить его в папке **Мемо**.

4. Поместить компонент *Memo* в форму и установить для свойства *ScrollBars* (линейки прокрутки) значение *ssBorth*, а для свойства *Align* (размещение) значение *alLeft* (левая часть формы).

5. Записать код для загрузки файла при создании формы:

```
Procedure TForm1.FormCreate(Sender: TObject);
```

```
begin
```

```
    Memo1.Lines.LoadFromFile('My_text.txt');
```

```
end;
```

6. Записать код, позволяющий сохранить файл при закрытии формы:

```
Procedure TForm1.FormClose (Sender: TObject; var Action: TCloseAction);
```

```
begin
```

```
    Memo1.Lines.SaveToFile('My_text.txt');
```

```
end;
```

7. Запустить программу. Добавить что-нибудь в появившийся текст. Закрыть программу.

8. Запустить ее еще раз и убедиться, что загружается откорректированный текст.

9. Добавить группу переключателей (*RadioButton*) для выбора цвета шрифта и записать код для обработки события *OnClick* на объекте *RadioButton1*:

```
Procedure TForm1.RadioButton1Click(Sender: TObject);
```

```
begin
```

```
    Memo1.Font.Color:=clRed;
```

```
end;
```

Самостоятельно записать код для процедур: **Procedure TForm1.RadioButton2Click**,
Procedure TForm1.RadioButton3Click.

10. Добавить группу переключателей для выбора размера шрифта и записать следующую код для процедуры **Procedure TForm1.RadioButton4Click**:

```
Procedure TForm1.RadioButton4Click(Sender: TObject);
```

```
begin
```

```
    Memo1.Font.Size:=200;
```

```
end;
```

Запишите самостоятельно коды процедур **RadioButton5Click** и **RadioButton6Click**.

11. Добавить группу переключателей для выбора вида шрифта. Введите код для обработки события **OnClick** на объекте **RadioButton7**.

```
Procedure TForm1.RadioButton7Click(Sender: TObject);
```

```
begin
```

```
    Memo1.Font.Name:='A97_Okton_Times';
```

```
    Memo1.Font.Name:='Corier New';
```

```
    Memo1.Font.Name:='Arial';
```

```
end;
```

12. Запустите и при успешной работе сохраните программу.

ЛАБОРАТОРНАЯ РАБОТА № 17

Тема: **Игра «15»**

Цели работы: На экране находятся 16 клеток по 4 в ряд. В начале игры в 15 клетках высвечено случайное число в диапазоне 1...15. Последняя клетка не заполнена. Игрок должен за минимальное количество ходов расположить числа в порядке возрастания. Каждый ход заключается в щелчке по одной панели с цифрами, после чего она перемещается на пустую панель.



Рис. 23.

Алгоритм выполнения задания:

1. Установите следующие свойства форм.

№	Свойство		Значение
1	BorderIcons	biMaximize	False
2	BorderStyle		bsSingle
3	Caption		Игра "15"
4	Height		195
5	Name		Form1
6	Position		poScreenCenter
7	Top		107
8	Width		207

2. Поместить на форму объекты *Panel1*,..., *Panel16* и выделить их. Размеры 30x36. В окне *Object Inspector* выбрать страницу *Events*. Дважды щелкнуть на событии *OnClick*. Будет создан обработчик событий, общий для всех выделенных объектов. Записать код:

```
Procedure TForm1.Panel1Click(Sender: TObject);
```

```
var T,L : integer;
```

```
begin
```

```
    With ( Sender as TPanel ) do
```

```
    begin
```

```
        L := Panel16.Left;
```

```
        T := Panel16.Top;
```

```
        Panel16.Left := Left;
```

```
        Panel16.Top := Top;
```

```
        Left := L; Top := T;
```

```
    end;
```

```
end;
```

4. Добавить подсчет количества сделанных игроком ходов.

ЛАБОРАТОРНАЯ РАБОТА № 18

Тема: Индикация состояния процесса

Цель работы: Изучение метода, с помощью которого создается индикация состояния процесса

Мы остановились на понятии «Индикация состояния процесса», потому что именно для этих целей, чаще всего применяется **TProgressBar**. Вспомни любое окно копирования файлов или любых других данных.

Практически в любом таком окне есть бегунок, который показывает, сколько процентов сейчас выполнено.

TProgressBar:

У этого компонента есть три необходимых свойств:

-  **Max** – максимальное значение (по умолчанию = 100)
-  **Min** – минимальное значение (по умолчанию = 0)
-  **Position** – позиция

Рассмотрим пример для изучения. Допустим, что тебе нужно вычислить в цикле 100 чисел. В этом случае очень удобно поставить на форму компонент **TProgressBar** и отображать в нём текущее вычисляемое значение. Для этого рассмотрим пример такого случая на реальном примере кода.

Алгоритм выполнения задания:

1. Открыть новый проект.
2. Разместить в форме следующие компоненты: компонент **TProgressBar**(Вкладка **Win32**), кнопку **Button** и метку **Label**. Установите следующие свойства:

Объект	Свойства	Значение
Form1	Caption	Пример работы ProgressBar
	Height	255
	Width	445
TProgressBar	Height	25
	Left	8
	Max	80
	Top	95
	Width	417
Button1	Caption	Старт

	Height	25
	Left	8
	Top	65
	Width	75
Label1	Caption	Здесь можно делать какой-то расчёт. После расчёта отображаем текущее состояние
	Height	25
	Left	88
	Top	64
	Width	337

3. Теперь по нажатию кнопки напиши следующее:

Procedure TForm1.Button1Click(Sender: TObject);

var

i:Integer;

begin

for i:=0 to 80 **do**

begin

ProgressBar1.Position:=i;

Sleep(100);

end;

ProgressBar1.Position:=0;

end;

В данном случае мы запускаем цикл от 0 до 80. На каждом этапе цикла позиция *ProgressBar* увеличивается на 5 и на двадцатом шаге выполнения цикла будет равна своему максимальному значению – 100.

4. Теперь запускайте приложение и смотрите (Рис.24), как оно выглядит.

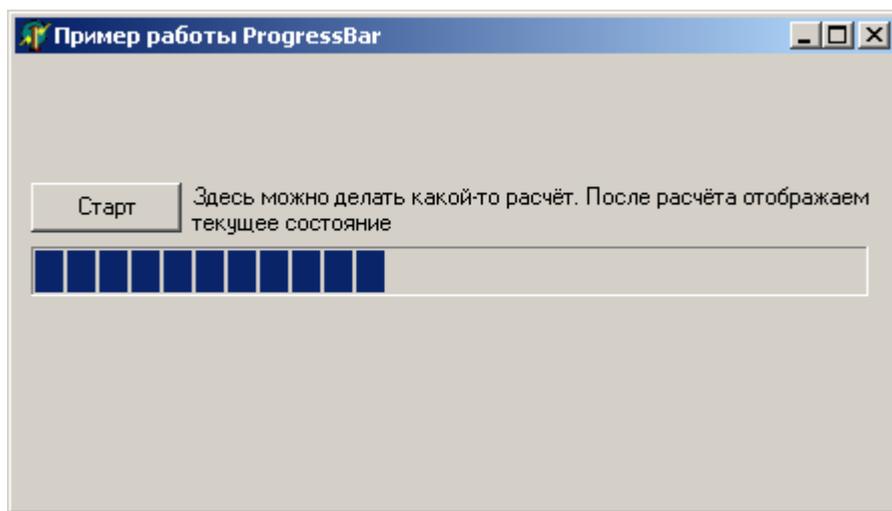


Рис. 24.

ЛАБОРАТОРНАЯ РАБОТА № 19

Тема: Компоненты управления файлами

Цель работы: Работа с графическими файлами.

Для работы с файлами используем простые файловые компоненты с закладки

Win3.1. Таких компонент здесь четыре:

	<i>DriveComboBox</i> – список логических устройств
	<i>DirectoryListBox</i> – список каталогов указанного диска
	<i>FileListBox</i> – список файлов указанного каталога
	<i>FilterComboBox</i> – задание шаблона для <i>FileListBox</i>

Приведем пример проекта работы с файлами с использованием простых файловых компонент.

Алгоритм выполнения задания:

1. Поместим компоненты *DriveComboBox*, *DirectoryListBox*, *FileListBox*, *FilterComboBox*(страница **Win3.1**), и *Button* (страница *Standard*) в форму.

2. Добавим перечисленные выше компоненты на форму вместе с компонентом *Image* (страница **Additional**) для просмотра выбранных из списка файлов, содержащих графические изображения.

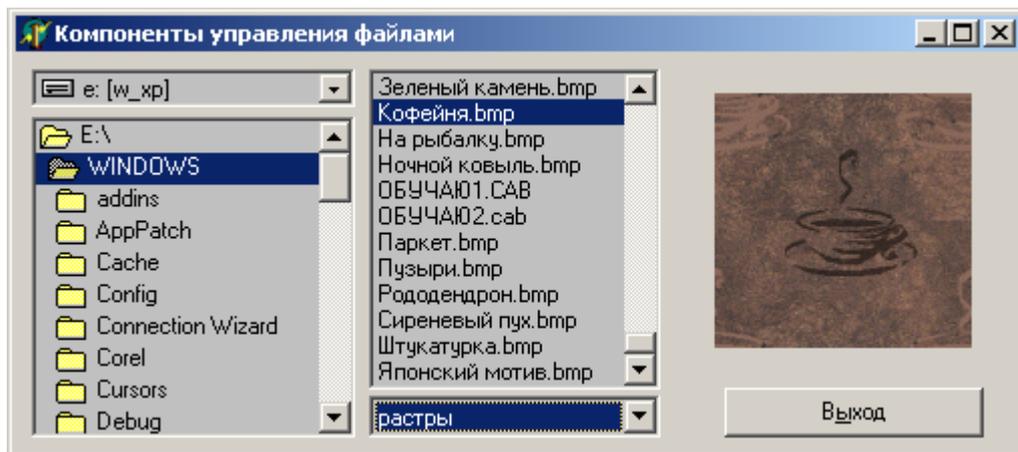


Рис. 25.

3. Свойство **Filter** шаблона файлов *FilterComboBox* определим при создании формы:

```
Procedure TForm1.FormCreate(Sender: TObject);
```

```
begin
```

```
    FilterComboBox1.Filter:= 'метафайлы|*.wmf|'+ 'иконки|*.ico|'+ 'растры|*.bmp'
```

```
end;
```

4. Свойство *DbClick* для *DirectoryListBox* определяем как

```
FileListBox1.Directory:=DirectoryListBox1.Directory
```

5. Свойство *OnChange* для *DriveComboBox* (смена диска) определяем как

```
DirectoryListBox1.Drive:=DriveComboBox1.Drive;
```

```
FileListBox1.Directory:=DirectoryListBox1.Directory
```

6. Выбранный файл должен отображаться в области компонента *Image1*, поэтому в *FileListBox* на событие *OnClick* вводим

```
Image1.Picture.Loadfromfile(Filelistbox1.Filename)
```

9. Сохранить форму и проект.

ЛАБОРАТОРНАЯ РАБОТА № 20

Тема: Работа с меню (главное меню – MainMenu)

Цель работы: Приобретение навыков работы с компонентом MainMenu.

Для более широкого понятия, создадим маленькую программу, которая будет содержать какое-то меню. Какое именно не особо важно, лишь бы научится с ним работать.

Алгоритм выполнения задания:

1. Открыть новый проект. Брось на форму один компонент **MainMenu** . Теперь посмотрим, какие есть свойства у этого компонента:
 - ✚ **AutoHotkeys** – будут ли создаваться автоматически клавиши быстрого вызова. Если ты выберешь *maAutomatic*, то **Delphi** будет автоматически создавать клавиши. При *maManual* придётся это делать вручную.
 - ✚ **AutoMerge** – автоматическое слияние с дочерними окнами.
 - ✚ **Images** – сюда можно подключать списки картинок, которые смогут отображаться на пунктах меню.
 - ✚ **Items** – здесь описываются пункты меню.
2. Сразу подключим список картинок. Брось на форму компонент **ImageList** с закладки **Win32**.
3. Теперь дважды щёлкни по нему и перед тобой откроется окно:

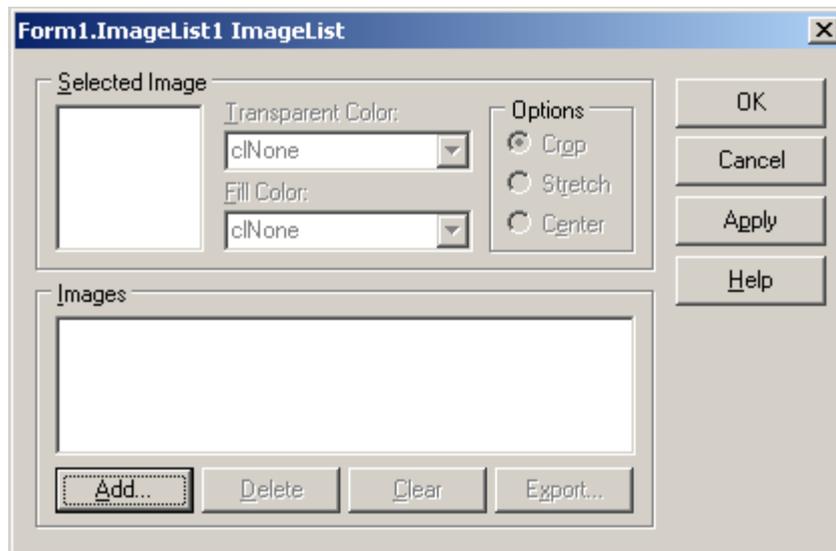


Рис. 26. Добавление картинок

Здесь нажми кнопку *Add* чтобы добавить картинку. Откроется стандартное окно открытия файла. Открой какую-нибудь картинку, и она добавится в список. Желательно, чтобы она была размером 16x16.

Можешь таким образом добавить несколько картинок. То, что мы добавили, смотри на рисунке 27.:

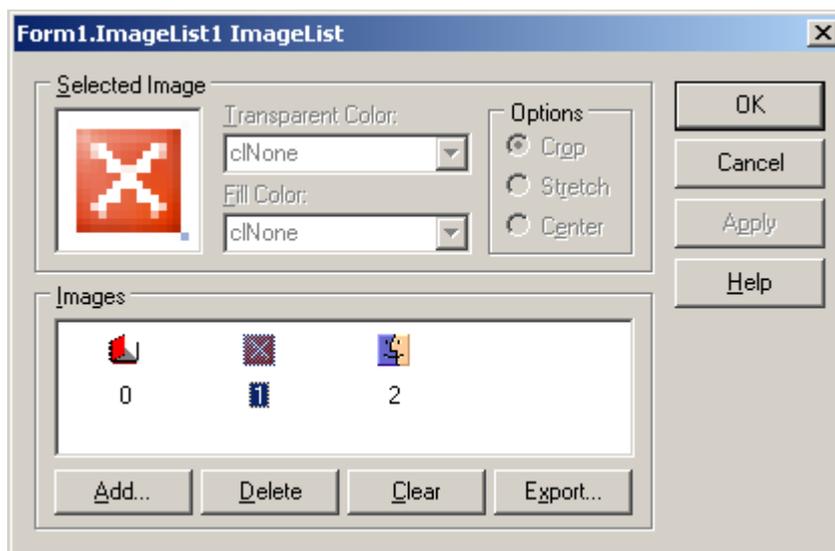


Рис. 27. Заполненный список картинок.

4. Теперь подключим наш список картинок к меню. Выдели компонент *MainMenu1* и у свойства *Images*, в выпадающем списке выбери пункт *ImageList1*.
5. Теперь создадим само меню. Для этого дважды щёлкни по свойству *Items* и перед тобой откроется редактор меню:



Рис. 28. Редактор меню

Этот же редактор можно вызвать, если дважды щёлкнуть по компоненту *MainMenu1*. Желтым кругом на рисунке 28. мы выделили уже созданный пункт. Перейди в объектный инспектор и намери в свойстве *Caption* слово «**Файл**». Как только ты нажмёшь кнопку *Enter*, будет создано меню «**Файл**»:

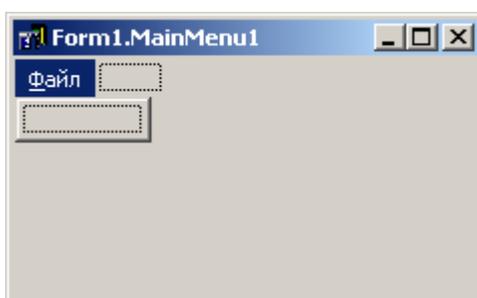


Рис. 29. Меню «Файл»

6. Для продолжения меню создадим меню «**Помощь**». Щёлкни справа от созданного

меню (в рамочке обведённой пунктиром) и снова перейди в объектный инспектор. Там введи в свойстве *Caption* слово «Помощь». Щёлкни в рамке чуть ниже меню «Помощь», как показано на рисунке 30.:

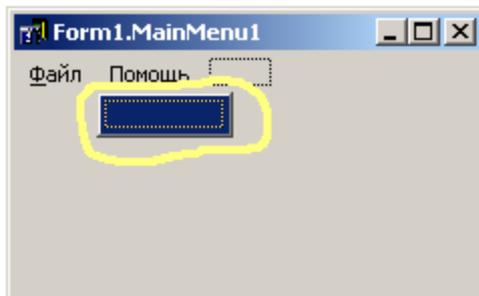


Рис. 30. Меню «Помощь»

Здесь, в свойстве *Caption* мы введём слово «О программе». У тебя должно получиться что-то похожее на этот рисунок:

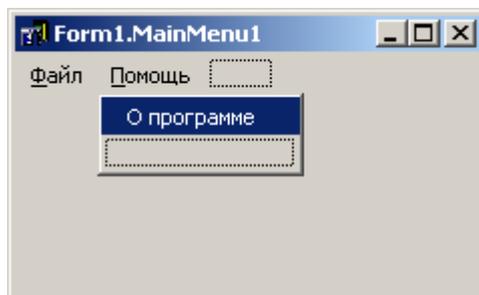


Рис. 31. Подменю «О программе»

7. Теперь, таким же образом заполним меню «Файл». Установить следующие свойства команд меню «Файл»:

Меню	Подменю	Свойства	Значение
Файл	Открыть	Caption	Отк&рыть
		ImageIndex	0
		ShortCut	Ctrl+O
	Линия (разделитель)	Caption	«-»
	Выход	Caption	В&ыход
		ImageIndex	1
ShortCut		F10	

8. Теперь создадим обработчик события нажатия по пункту меню. Для этого выбери в меню пункт «Выход» и щёлкни по нему дважды или перейди на закладку *Events* и дважды щёлкни по событию *OnClick*. Эти действия заставят **Delphi** создать обработчик события по нажатию меню. В этом обработчике напишем следующее:

Procedure TForm1.N4Click(Sender: TObject);

begin

Form1.Close;

end;

9. Сохранить проект под именем, например, *NewMMenu.pas* и *Menu.dpr*.

10. Теперь запускайте приложения и смотрите.

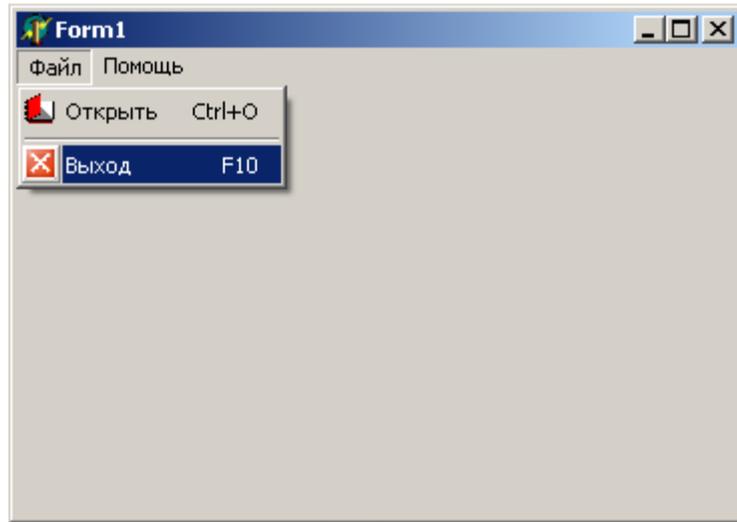


Рис. 32. Подменю «О программе»

ЛАБОРАТОРНАЯ РАБОТА № 21

Тема: Использование средств мультимедиа.

Цели работы: Использование возможностей элемента управления MCI.

Для управления устройствами мультимедиа в *Delphi* применяется специальный интерфейс MCI (*Multimedia Control Interface*). Основным компонентом этого интерфейса является элемент управления MediaPlayer с большим набором свойств и команд, позволяющих управлять мультимедиа и полностью их контролировать.

Для изучения работы элемента управления MediaPlayer выполните следующие действия:

1. Поместить компоненты *Label* (вкладка **Standard**), *MediaPlayer*, *Timer* (вкладка **System**) две кнопки по имени *BitBtn* (вкладка *Additional*), *OpenDialog* (вкладка **Dialogs**) и *Gauge* (вкладка **Samples**) в форму *Form1*.

2. Прямо на форме разместить компонент *Panel* (вкладка **Standard**) с размерами 349x281.

4. Установить следующие свойства объектов:

Объект	Свойство	Значение
Form1	Caption	Возпроизведение
Label1	Caption	
	Alignment	TaLeftJustify
	Height	25
	Width	348
	AutoSize	False
BitBtn1	Caption	
	Glyph	E:\Program Files\Microsoft Visual Studio\Common\Graphics\Bitmaps\TIBr_W95\Open.bmp
BitBtn2	Caption	
	Glyph	E:\Program Files\Microsoft Visual Studio\Common\Graphics\Bitmaps\TIBr_W95\Undo.bmp
OpenDialog	Filter	Аудио *.wav, *.mid *.wav; *.mid
		Видео *.avi *.avi
		Все файлы *.*
Gauge	Height	25
	Width	348
Panel1	Caption	
	BevelInner	bvLowered

4. Записать код для обработки события *Click* на объекте *BitBtn1*:

```
Procedure TForm1.BitBtn1Click(Sender: TObject);
```

```
begin
```

```
    OpenFileDialog1.Execute;
```

```
    MediaPlayer1.FileName:=OpenDialog1.FileName;
```

```
    MediaPlayer1.Open;
```

```
    Label1.Caption:='Загружен файл <<' + MediaPlayer1.FileName+ '>>';
```

```
end;
```

5. После двойного щелчка на кнопке «MediaPlayer1» можно заполнить шаблон процедуры реакции на нажатие этой кнопки.

Procedure TForm1.MediaPlayer1Click(Sender: TObject; Button: TMPBtnType; var DoDefault: Boolean);

begin

MediaPlayer1.Display:=Panel1;

MediaPlayer1.Play;

end;

6. Записать код для процедуры обработки события *Click* (щелчок мыши) на кнопке

Timer1:

Procedure TForm1.Timer1Timer(Sender: TObject);

begin

with MediaPlayer1 do

if FileName<>" then

Gauge1.Progress:=Round(100*Position/Length);

end;

7. Самостоятельно записать код для процедур:

procedure TForm1.Button2Click(Sender: TObject)

8. Запустить и при успешной работе сохранить программу.

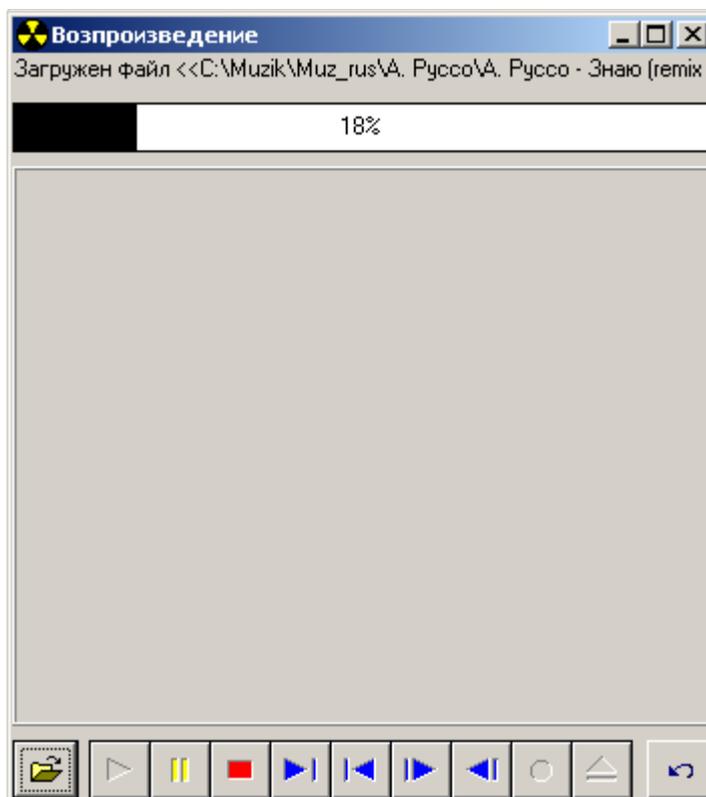


Рис. 33.

ЛАБОРАТОРНАЯ РАБОТА № 22

Тема: Воспроизведение немых видео клипов - компонент Animate

Цель работы: Компонент **Animate** позволяет воспроизводить на форме стандартные видео клипы Windows (типа копирования, поиска файлов и т. п.) и немые видео файлы .avi. Эти файлы представляют собой последовательность кадров битовых матриц.

Алгоритм выполнения задания:

1. Создайте новое приложение, перенесите на форму компонент **Animate** (вкладка **Win32**). Воспроизводимое изображение задается одним из двух свойств: *Filename* или *CommonAVI*. Первое свойство позволяет программно задать имя воспроизводимого файла. А второе - воспроизводить стандартные мультипликации **Windows**.

2. Установить следующие свойства объектов:

Объект	Свойство	Значение
Form1	Caption	Стандартные видео клипы Windows
	Height	182
	Position	poScreenCenter
	Width	342
Animate1	Active	False
	CommonAVI	aviCopyFiles
	Height	60
	Left	24
	Top	8
	Width	272
	Visible	False
Animate2	Active	False
	FileName	'E:\Program Files\Borland\Delphi5\Demos\Coolstuff\cool.avi'
	Height	40
	Left	312
	Top	20
	Width	60

	Visible	False
--	---------	-------

3. Поместите компоненты *Button* (вкладка **Standard**) *Timer*(вкладка **System**) и *ProgressBar*(вкладка **Win32**) в форму *Form1*. Необходимо изменить свойства объектов.

Объект	Свойство	Значение
ProgressBar1	Height	25
	Left	8
	Top	80
	Width	377
Button1	Caption	Стандартные видео клипы Windows
	Name	cbStart
Timer	Enabled	False
	Interval	50

- ✚ Кнопку **Старт** - которая будет начинать процесс воспроизведения;
- ✚ Компонент **Timer**, который служит для отсчета интервалов времени.
- ✚ Компонент **ProgressBar1** - предназначенный для отображения хода выполнения длительного по времени процесса, который начинается с момента срабатывания таймера. Свойства *max* и *min* - значения диапазона изменения. Свойство *Position* - содержит текущее значение отображаемой величины.

4. Напишем два обработчика событий *Procedure TForm1.cbStartClick* – запускает процесс, и *Procedure TForm1.Timer1Timer* – которая завершает процесс и подготавливает компоненты к следующему запуску.

5. *Первое процедура (Procedure TForm1.cbStartClick):*

Procedure TForm1.cbStartClick(Sender: TObject);

begin

 Animate1.CommonAVI :=aviCopyFile;

 Animate1.Active:=True;

 Animate1.Visible:=true;

 Animate2.FileName :='E:\Program Files\Borland\Delphi5\Demos\Coolstuf\cool.avi';

 Animate2.Active:=True;

 Animate2.Visible:=true;

 ProgressBar1.Show;

 Timer1.Enabled:=True;

end;

6. **Второе процедура (Procedure TForm1.Timer1Timer):**

```
Procedure TForm1.Timer1Timer(Sender: TObject);
```

```
Begin
```

```
    ProgressBar1.Position:= ProgressBar1.Position + 1;
```

```
    if ProgressBar1.Position >= ProgressBar1.Max then begin Timer1.Enabled:=False;
```

```
    ProgressBar1.Position:=0;
```

```
    ProgressBar1.Hide;
```

```
    end;
```

```
end;
```

7. После выполнения нашего приложения эффект возможно выглядит таким образом:

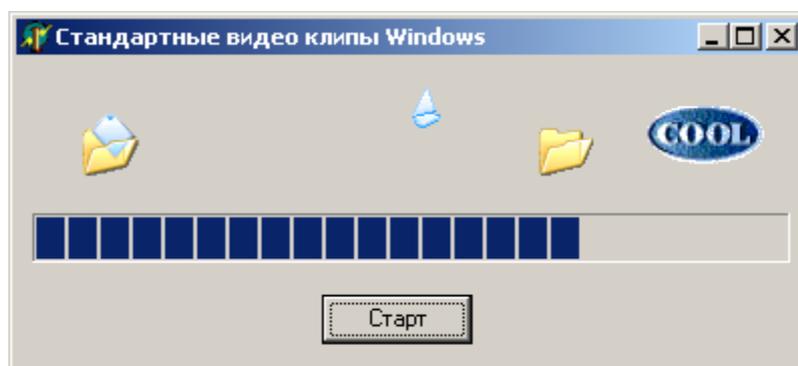


Рис. 34.

ЛАБОРАТОРНАЯ РАБОТА № 23

Тема: SDI-приложения

Цель работы: Создадим простую программу просмотра изображения для демонстрации SDI.

Построение интерфейса

Обычно первым шагом построения программы является создание интерфейса. Не будем отступать от традиций, и выполним следующие действия:

1. Выберите команду *File* → *New Application*, и появится пустое приложение.
2. Установите следующие свойства форм.

№	Свойство	Значение
1	Caption	Одно - документный интерфейс
2	FormStyle	fsNormal

3	Name	frmMain
4	ShowHint	True

5. Поместите компонент *TPanel* в форму. Установите следующие его свойства.

№	Свойство	Значение
1	Align	alTop
2	Caption	-

6. Поместите три компонента *TSpeedButton* в *TPanel* и назовите их *spbtnLoad*, *spbtnStretch* и *spbtnCenter*. Установите следующие их свойства.

№	Объект	Свойство	Значение
1	SpeedButton1	Name	spbtnLoad
		Hint	Открыть файл
		Left	8
		Top	100
2	SpeedButton2	Name	spbtnStretch
		AllowAllUp	True
		GroupIndex	1
		Hint	Протяжение
		Left	120
		Top	100
3	SpeedButton3	Name	spbtnCenter
		AllowAllUp	True
		GroupIndex	2
		Hint	Центр
		Left	230
		Top	100

7. Поместите компонент *TImage* во вновь созданную *TPanel* и установите следующие его свойства.

№	Свойство	Значение
1	Align	alClient
2	Name	imgMain

8. Добавьте в форму *TOpenDialog* со следующими свойствами.

№	Свойство	Значение
1	Filter	Bitmaps (*.bmp) *.bmp
2	Name	opndlgLoad
3	Options	[ofPathMustExist,ofFile MustExist]

9. Теперь самое время сохранить проект, выбрав в меню команду *File* → *Save Project As*. Сохраните *Unit1* как *Main*, а проект - как *EgSDIApp*.

10. Теперь, после создания интерфейса, перейдем к написанию исходного текста вашего приложения. Сначала загрузите изображение следующим образом.

8.1. Дважды щелкните на компоненте *spbtnLoad*, и *Delphi* выведет окно редактора и автоматически создаст обработчик события *OnClick*.

Введите код:

```
Procedure TfrmMain.spbbtnLoadClick(Sender: TObject);  
begin  
if opndlgLoad.Execute then  
    imgMain.Picture.LoadFromFile(opndlgLoad.FileName);  
end;
```

Метод *opndlgLoad.Execute* вызывает стандартное диалоговое окно открытия файла. Если вы выбрали файл и щелкнули на *OK*, метод возвращает *True* и загружает в свойство *FileName* полный путь к имени файла. При щелчке на *Cancel* или нажатии клавиши *<Esc>* метод вернет *False*.

Компонент *TImage* предоставляет свойство *Picture*, которое является экземпляром класса *TPicture*. Этот класс обеспечивает работу с растровыми изображениями, пиктограммами и метафайлами. Один из его методов, *LoadFromFile*, служит для загрузки изображения по имени файла.

Выберите команду *Run* → *Run* для компиляции и запуска приложения и попытайтесь открыть картинку.

8.2. Теперь добавьте возможность растягивания изображения. Дважды щелкните на компоненте *spbtnStretch*, и *Delphi* выведет окно редактора и автоматически создаст обработчик события *OnClick*. Введите код.

```
Procedure TfrmMain.spbbtnStretchClick(Sender: TObject);
```

```
begin  
    imgMain.Stretch:= spbtnStretch.Down;  
end;
```

Компонент TSpeedButton имеет свойство Down, которое равно True при нажатой кнопке. Свойство Stretch класса TImage позволяет растянуть картинку.

8.3. Для выравнивания картинки по центру воспользуйтесь приведенной выше инструкцией (только используйте компонент spbtnCenter) и введите следующий код:

```
Procedure TfrmMain.spbtnCenterClick(Sender: TObject);  
begin  
    imgMain.Center:= spbtnCenter.Down;  
end;
```

9. Компилируйте, запускайте и смотрите Рис.35. SDI-приложение создано!

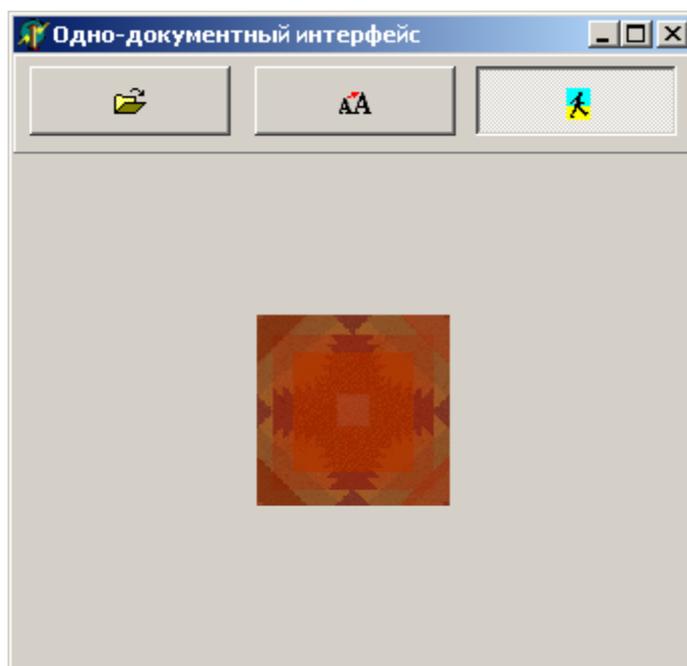


Рис. 35.

ЛАБОРАТОРНАЯ РАБОТА № 24

Тема: MDI интерфейс

Цель работы: Интерфейс MDI - приложения очень похож на интерфейс разработанного ранее SDI - приложения, но каждое изображение выводится в отдельной, а не в главной форме.

Выполните следующие действия для создания родительской формы:

1. Выберите команду *File* → *New Application*, и появится пустое приложение.
2. Установите следующие свойства:

№	Свойство	Значение
1	Caption	Много - документный интерфейс
2	FormStyle	FsMDIForm
3	Name	frmMDIParent
4	ShowHint	True

3. Поместите компонент *TPanel* в форму. Установите следующие его свойства:

№	Свойство	Значение
1	Align	AlTop
2	Caption	-

4. Поместите три компонента *TSpeedButton* в *TPanel* и назовите их *spbtnLoad*, *spbtnStretch* и *spbtnCenter*. Установите следующие их свойства:

№	Объект	Свойство	Значение
1	SpeedButton1	Name	spbtnLoad
		Hint	Открыть файл
		Left	8
		Top	155
2	SpeedButton2	Name	spbtnStretch
		AllowAllUp	True
		GroupIndex	1
		Hint	Протяжение
		Left	167

		Top	155
3	SpeedButton3	Name	SpbtnCenter
		AllowAllUp	True
		GroupIndex	2
		Hint	Центр
		Left	192
		Top	155

5. Добавьте в форму компонент *TOpenDialog* и установите следующие его свойства:

№	Свойство	Значение
1	Filter	Bitmaps (*.bmp) *.bmp
2	Name	opndlgLoad
3	Options	[ofPathMustExist,ofFile MustExist]

Теперь создадим дочернюю форму.

6. Выберите из меню *File* → *New Form*, и появится пустая форма..
7. Установите следующие ее свойства:

№	Свойство	Значение
1	FormStyle	fsMDIChild
2	Name	frmMDIChild
3	Position	poDefaultPosOnly

8. Поместите компонент *TImage* во вновь созданную форму и установите его следующие свойства:

№	Свойство	Значение
1	Align	alClient
2	Name	imgMain

9. Удалите дочернюю форму из списка автоматически создаваемых форм следующим образом:

- 9.1. Выберите команду *Project* → *Options*, и появится диалоговое окно *Project Options*, показанное на рис. 36.
 - 9.2. Выберите *frmMDIChild* в списке *Auto-create forms*.
 - 9.3. Щелкните на кнопке. Форма *frmMDIChild* при этом будет перенесена в список *Available forms*.
 - 9.4. Щелкните на кнопке *OK*.
10. Теперь самое время сохранить проект, выбрав команду *File* → *Save Project As*. Сохраните *Unit1* как *MDIParent* и *Unit2* как *MDIChild*, а проект - как *MDI*.
 11. Создав интерфейс, перейдем к написанию исходного текста приложения, который будет очень похож на код для *SDI*-приложения.

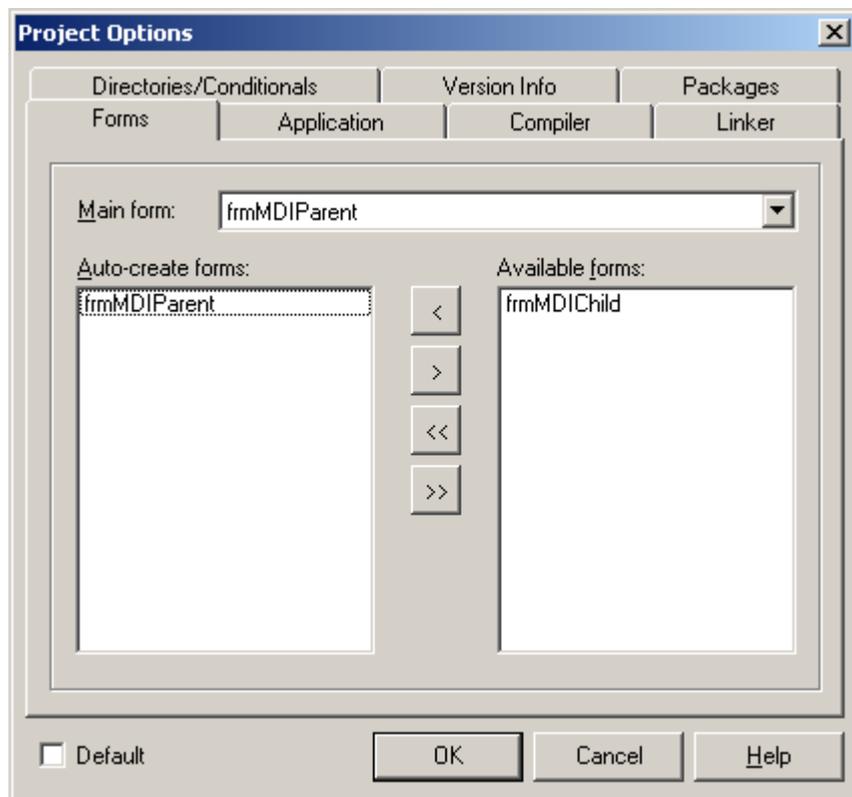


Рис. 36.

Сначала загрузим изображение. Введите следующий код в обработчик события *OnClick* компонента *spbtnLoad*:

```
Procedure TfrmMDIParent.spbtnLoadClick(Sender: TObject);
begin
    if opndlgLoad.Execute then
        with TfrmMDIChild.Create(Application) do
            begin
```

```

Caption:= opndlgLoad.FileName;
imgMain.Picture.LoadFromFile(opndlgLoad.FileName);
ClientWidth:= imgMain.Picture.Width;
ClientHeight:= imgMain.Picture.Height;
end;
end;

```

После запуска диалогового окна создается новый экземпляр дочерней формы и загружается файл изображения. После загрузки размеры дочерней формы изменяются так, чтобы можно было видеть все изображение.

Еще пара стрихов - и приложение заработает, как и предусматривалось. Поскольку модуль ссылается на тип *TfrmMDIChild*, находящийся в модуле *MDIChild*, после строки *Implementation* следует добавить еще одну строку:

```
uses MDIChild;
```

12. Теперь можно приступить к компиляции и запуску приложения. Однако заметьте, что, когда вы щелкаете на кнопке *Close*, дочерняя форма не закрывается, а сворачивается в пиктограмму. Чтобы заставить ее закрыться, следует добавить в код обработчика *OnClose* класса *TfrmMDIChild* маленькую деталь - изменить свойство *Action*:

```
Action:= caFree;
```

13. Компоненты *TSpeedButton Stretch* и *Center* выполняют те же функции, что и в *SDI* - приложении, однако их обработчики события *OnClick* следует изменить следующим образом

```

Procedure TfrmMDIParent.spbtnStretchClick(Sender: TObject);
begin
if not (ActiveMDIChild = Nil) then
    if ActiveMDIChild is TfrmMDIChild then
TfrmMDIChild(ActiveMDIChild).imgMain.Stretch:= spbtnStretch.Down;
end;

```

и

```

Procedure TfrmMDIParent.SpeedButton1Click(Sender: TObject);
begin
if not (ActiveMDIChild = Nil) then
    if ActiveMDIChild is TfrmMDIChild then
TfrmMDIChild(ActiveMDIChild).imgMain.Center:= spbtnCenter.Down;
end;

```

Остается последняя проблема - состояния кнопок *Stretch* и *Center* одинаковы для всех дочерних форм. Для решения этой задачи добавьте в обработчик события *OnActivate* класса *TfrmMDIChild* строки.

```
frmMDIParent.spbtnStretch.Down:= imgMain.Stretch;  
frmMDIParent.spbtnCenter.Down:= imgMain.Center;
```

И, наконец, самый последний из последних штрихов - в модуле *MDIChild* добавьте после строки *Implementation* строку:

```
uses MDIParent;
```

14. Компилируйте, запускайте и смотрите Рис.37. *MDI* - приложение создано!

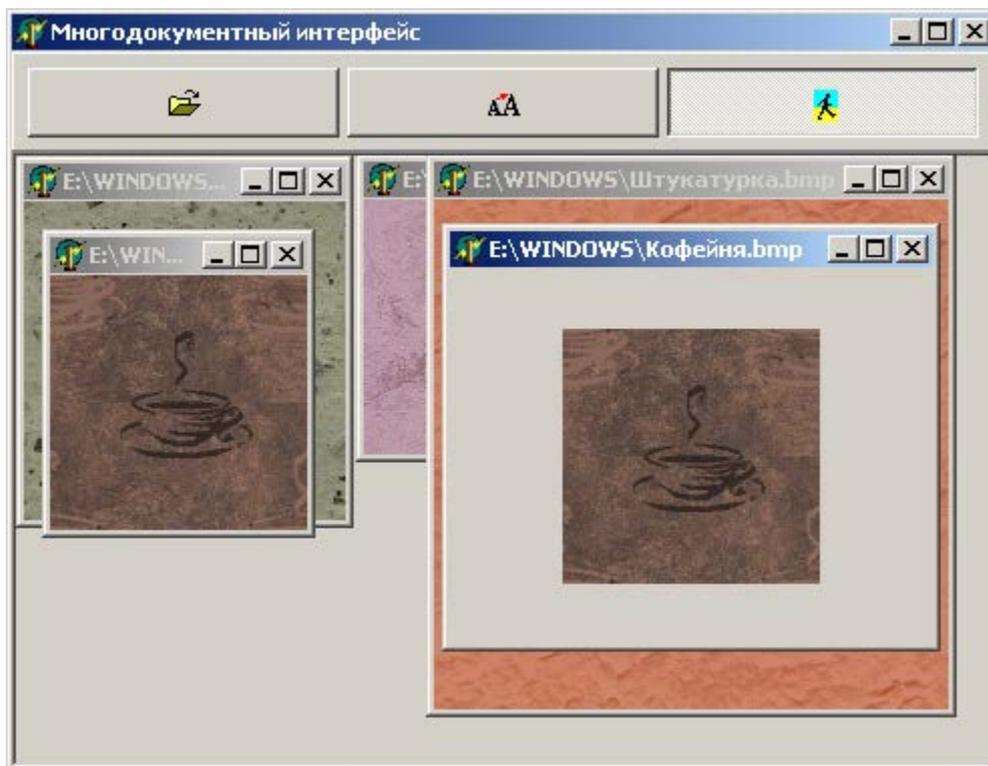


Рис. 37.

ЛАБОРАТОРНАЯ РАБОТА № 25

Тема: Базы данных в Delphi.

Цель работы: Изучение методов по созданию баз данных Access и связывание таблиц с приложениями Delphi.

Создание приложения, связанного с базами данных (БД) в Delphi, включает несколько этапов:

I. Создание первой базы данных Access

II. Связывание таблицы с приложением Delphi

I. Создание первой базы данных Access

Рассмотрим создания и использования базы данных Access. Для последующей работы необходимо, чтобы на вашем компьютере был установлен *MS Office* и его компонент *MS Access*. Именно в нём и будут создаваться базы, а вот работать с ними мы будем уже из *Delphi*.

Алгоритм выполнения задания:

1. Запусти *Access* и выбери в меню *Файл* → *Создать*. В мастере создания базы выберите пункт "*Новая база данных*" и нажмите "*ОК*" (Рис. 38.). Вам предложат выбрать имя базы и место расположения, укажите что угодно, а точнее свой файл **bd_prepod.mdb**.

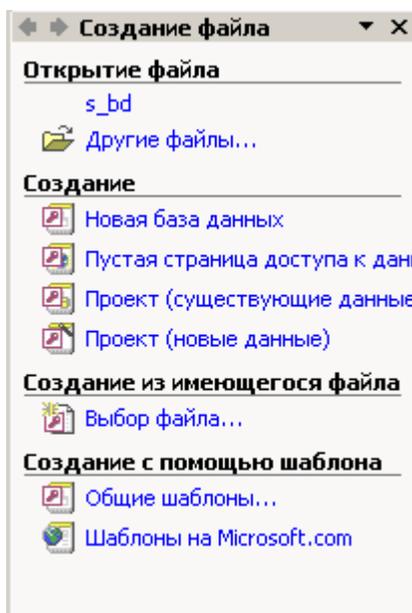


Рис. 38. Окно создания новой базы данных

2. После этого Access создает базу и сохранит её по указанному пути. А вы увидите окно как на рисунке 39., в котором и происходит работа с базой.

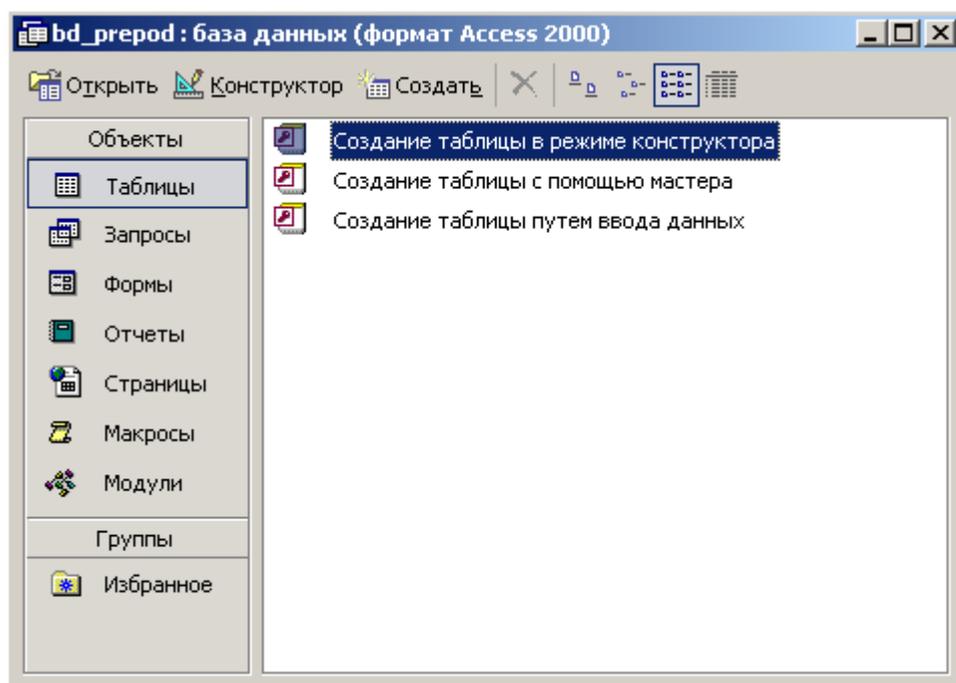


Рис 39. Окно создания новой базы данных

С левой стороны окна находится колонка выбора объектов, с которыми вы хотите работать. Первым находится пункт "Таблицы" (он выделен по умолчанию) который и будет нас интересовать. Если этот объект у вас не выделен, то выделяйте его. В окне справа находится три пункта:

-  *Создание таблицы в режиме конструктора*
-  *Создание таблицы с помощью мастера*
-  *Создание таблицы путём ввода данных*

С помощью этих команд можно создать таблицы внутри созданной базы данных, Access, которая может хранить в одном файле несколько таблиц.

3. Давайте попробуем создать базу данных справочника, чтобы увидеть всё на практике. Щёлкните по "Создание таблицы в режиме конструктора" чтобы создать новую таблицу в базе данных. Перед вами откроется окно, как на рисунке 40.

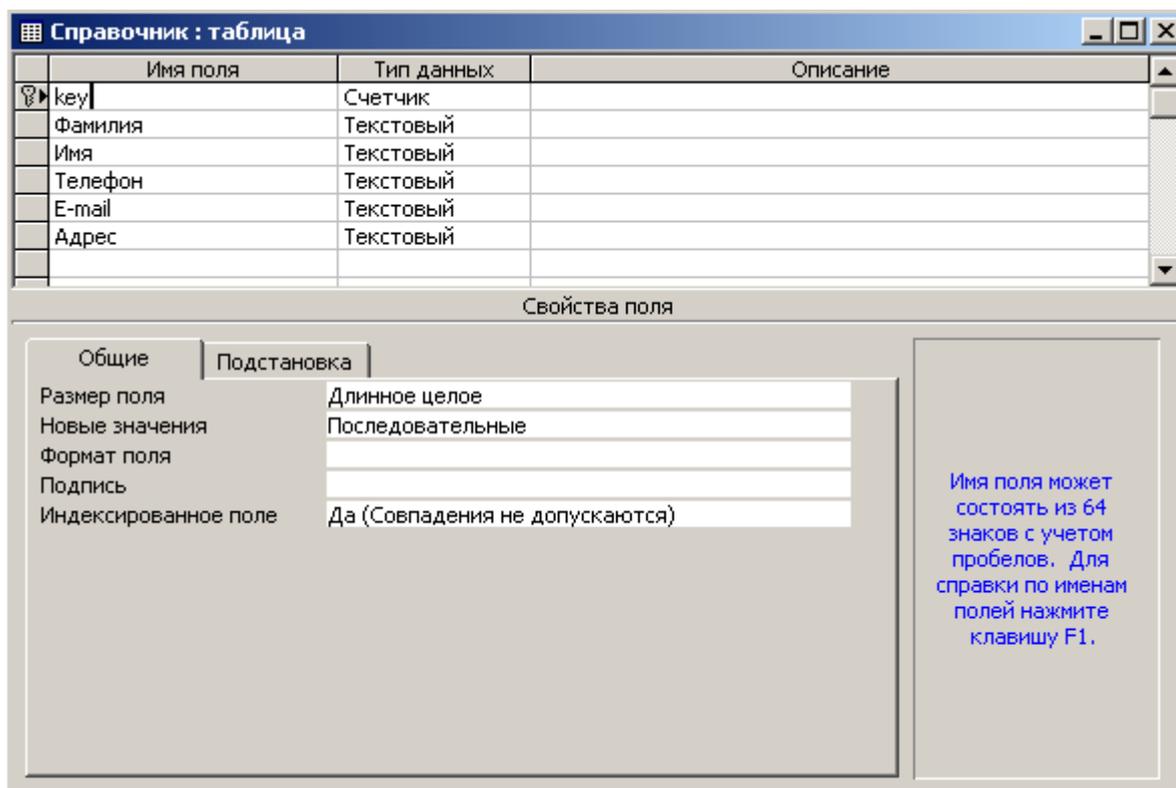


Рис 40. Окно создания таблицы

Создайте шесть полей:

1. Имя поля - *Key1*. Тип - счётчик. Это у нас будет ключик. Размер поля - "Длинное целое". Индексированное поле - "Да (Совпадения не допускаются)".
2. Имя поля – *Фамилия*. Тип - текстовый. Размер поля - 50. Индексированное поле - "Да (Допускаются совпадения)".
3. Имя поля – *Имя*. Тип - текстовый. Размер поля - 50. Индексированное поле - "Да (Допускаются совпадения)".
4. Имя поля – *Телефон*. Тип - текстовый. Размер поля - 10. Индексированное поле - "Да (Допускаются совпадения)".
5. Имя поля - *e-mail*. Тип - текстовый. Размер поля - 20. Индексированное поле - "Да (Допускаются совпадения)".
6. Имя поля – *Адрес*. Тип - текстовый. Размер поля - 50. Индексированное поле - "Да (Допускаются совпадения)".

Помимо этого, у всех полей значение "*Обязательно поле*" стоит в "*Нет*", и "*Пустые строки*" выставлено в "*Да*". Если вы сделаете поле обязательным, то во всех строках обязательно должно быть заполнено соответствующее поле. Если вы запретите пустые строки (поставишь «*Нет*»), то в указанном поле должно быть обязательно что-то введено, иначе произойдёт ошибки. В реальных условиях, если какое-то поле обязательно

должно иметь значение, то лучше сделать его обязательным. Не надо надеяться на добропорядочность пользователя, потому что они слишком часто подводят. Пусть лучше база данных следит за правильностью данных.

4. Теперь выделите первое поле (*Key1*), щёлкните правой кнопкой мыши и выберите пункт "Ключевое поле" (рисунок 41.). Задание ключевого поля является обязательным действием, если вы этого не сделаете, то таблица не сможет редактироваться, а это значит, что в неё нельзя будет добавить строки.

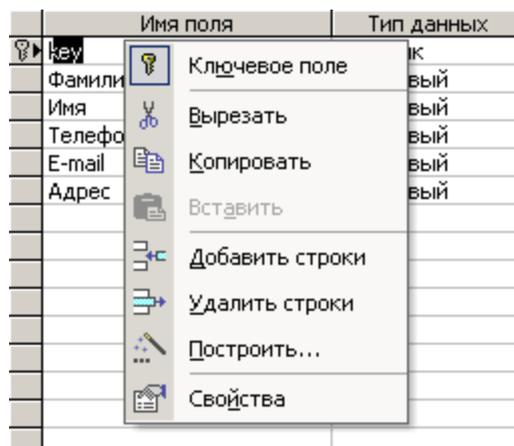


Рис 41. Задание ключевого поля

5. Теперь таблицу можно сохранять и закрывать. На вопрос: «Сохранить таблицу» отвечайте положительно, и сохраняйте под именем «Справочник». Наша первая база данных готова.

II. Связывание таблицы с приложением Delphi

Мы пишем программу, которая будет работать с базой данных *MS Access*. Для разработок от *MS Access* лучше всего использовать *ADO*. Напишем наше первое приложение для работы с базой данных.

1. Создайте новый проект. Теперь сбрасывайте на форму компонент *ADOConnection* с закладки *ADO* палитры компонентов. Теперь настроим соединение с сервером, которое должно быть прописано в свойстве *ConnectionString*. Для этого надо дважды щёлкнуть по строке *ConnectionString* и перед нами открывается окно, как на рисунке 42.

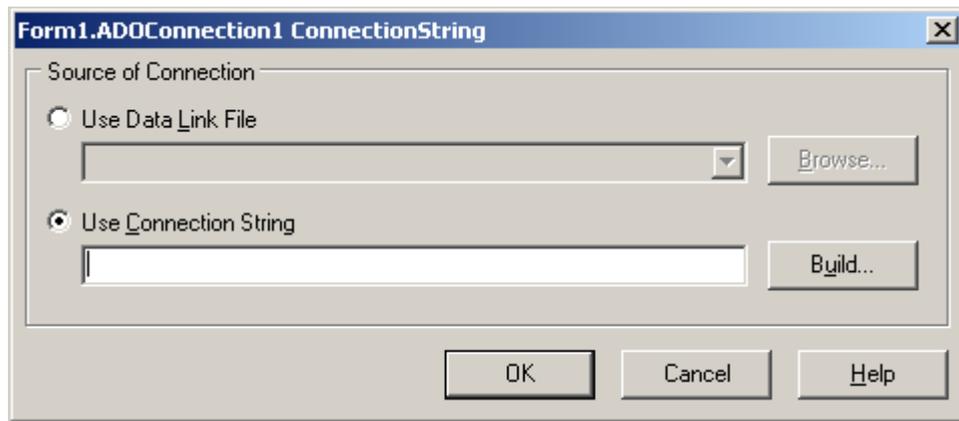


Рис 42. Окно создания подключения к базе

Здесь перед нами стоит выбор:

- ✚ Использовать специальный файл (*Use Data Link File*);
- ✚ Использовать строку подключения (*Use Connection String*).

2. Второе является более предпочтительнее, поэтому создадим строку подключения. Для этого щёлкаем кнопку Build и перед нами открывается ещё одно окно, показанное на рисунке 43.

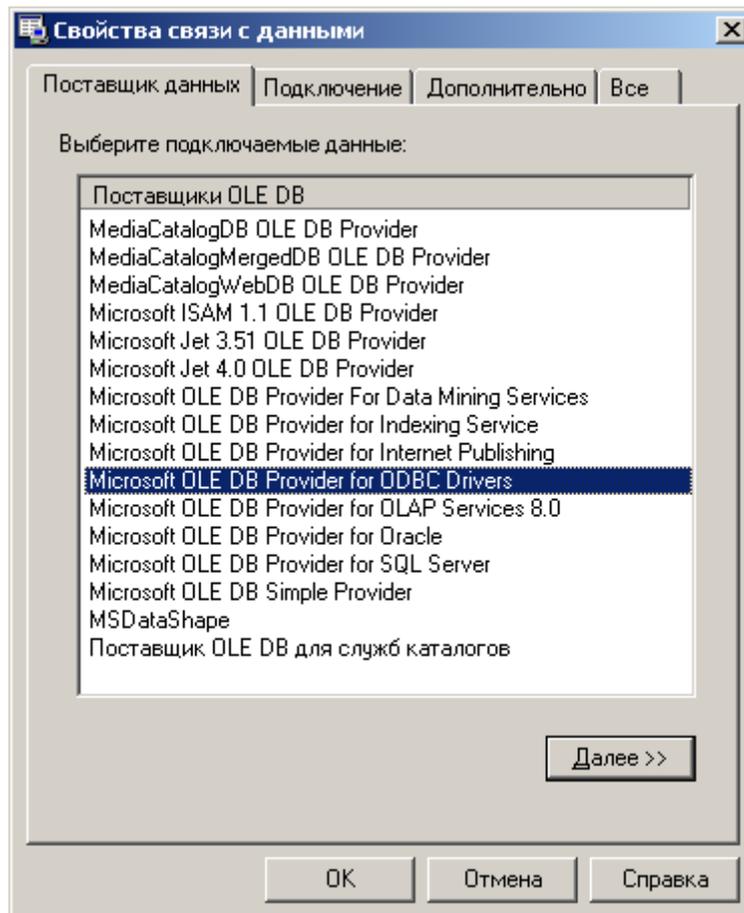


Рис 43. Окно создания строки подключения

На закладке *Provider* перечислены все доступные ADO драйверы доступа к базам данных. Если какого-то драйвера нет, то можно попробовать выделенный по умолчанию «*Microsoft OLE DB Provider for ODBC Drivers*». Этот драйвер позволяет получить доступ к базе данных через ODBC драйвер, которые есть к большинству существующих баз данных (единственное, он может быть не установленным на вашем компьютере).

В нашем случае, для доступа к базам данных MS Access используется драйвер «*Microsoft Jet OLE DB Provider*». Такой драйвер обязательно устанавливается на машину вместе с MS Office, а в последних версиях Windows он устанавливается по умолчанию.

3. На нашей машине установлено сразу две версии этого драйвера, поэтому мы выберем более новую - «*Microsoft Jet 4.0 OLE DB Provider*». После этого нажимаем кнопку *Next*, или переходим на закладку «*Connection*».

Вид закладки *Connection* зависит от выбранного драйвера. В нашем случае она должна выглядеть, как показано на рисунке 44.

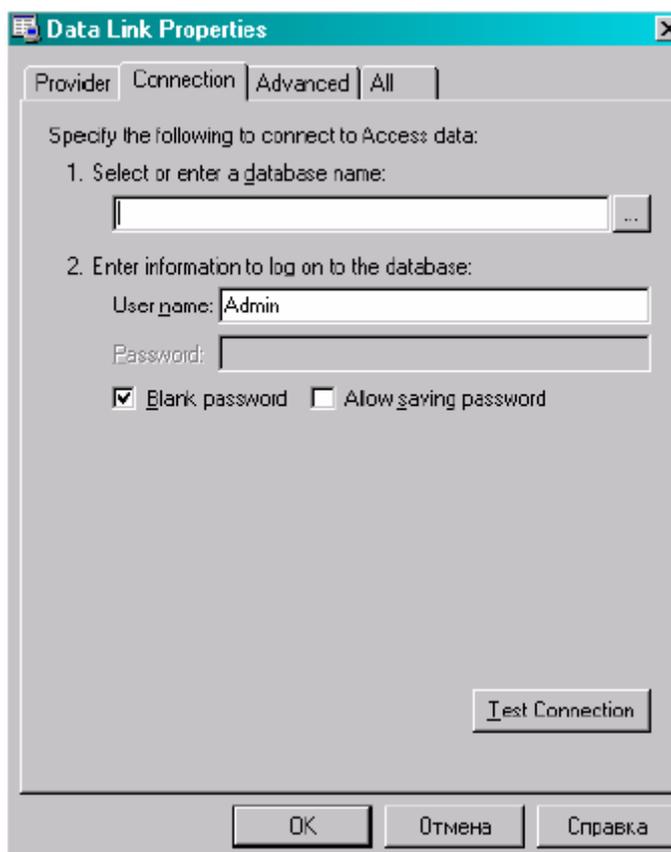


Рис 44. Закладка *Connection*

Первым делом, в этом окне надо ввести имя (если надо то и путь) базы данных в строку «*Select or enter a database name*». Если база данных будет располагаться в той же директории, что и запускной файл, то путь указывать не надо. Советуем хранить базы в одной директории с запускными файлами. Если вы будете держать файлы отдельно от

запускового, то вам придётся указывать полный путь, а это может вызвать проблемы при переносе программы на другой компьютер. Ведь там программа будет искать базу по указанному пути, который может измениться. Если хотите держать файлы в другой директории, то указывай относительный путь относительно текущей директории.

Чтобы легче было выбрать файл базы данных необходимо щёлкнуть по кнопке с точками справа от строки ввода.

Помимо этого нам надо заполнить следующие поля:

- ✚ Имя пользователя (*User name*), можно оставить по умолчанию, если не заданно иное при создании базы в MS Access;
- ✚ Пароль (*Password*) – если база имеет пароль, то его необходимо указать;
- ✚ Пустой пароль (*Blank password*) – если пароль не нужен, то здесь желательно поставить галочку;
- ✚ Позволять сохранять пароль (*Allow saving password*). Если здесь поставить галочку, то пароль может быть сохранён.

4. Как только выберешь базу данных, нажмите кнопку *Test Connection*, чтобы протестировать соединение. Если всё указано правильно, то вы должны увидеть сообщение «*Test connection succeeded*». Всё, можно нажать ОК, чтобы закрыть окно создания строки подключения и ещё раз ОК, чтобы закрыть окно редактора строки подключения.

5. Теперь установите следующие свойства компонента ADOConnection.

№	Компонент	Свойство	Значение
1	ADOConnection	LoginPrompt	False
		Connected	True

6. На этом соединении можно считать оконченным. Теперь нам надо получить доступ к созданной нами таблице «Справочник». Для этого сбрасывайте на форму компонент *ADOTable* с закладки *ADO* палитры компонентов, и установите следующие свойства компонента ADOTable.

№	Компонент	Свойство	Значение
1	ADOTable	Name	HelpBook
		Connection	ADOConnection1
		TableName	Справочник
		Active	True

7. Для отображения данных из таблицы надо ещё установить на форму компонент

DataSource с закладки *Data Access* палитры компонентов. Теперь этому компоненту надо указать, какую именно таблицу он должен отображать. Для этого в свойстве *DataSet* нужно из выпадающего списка выбрать нашу таблицу *HelpBook*.

8. Все приготовления готовы, можно приступать к реальному отображению данных. Самый простой способ отобразить таблицу - установить компонент *DBGrid* с закладки *Data Controls* палитры компонентов. Это компонент-сетка, которая может отображать данные в виде таблицы. В этом же компоненте можно добавлять, удалять и редактировать строки нашей таблицы.

9. И последний этап создания нашего приложения - связывание компонента сетки с компонентом отображения таблицы. Для этого в свойстве *DataSource* компонента *DBGrid* нужно указать созданный нами компонент *DataSource1* (см.Рис.45).

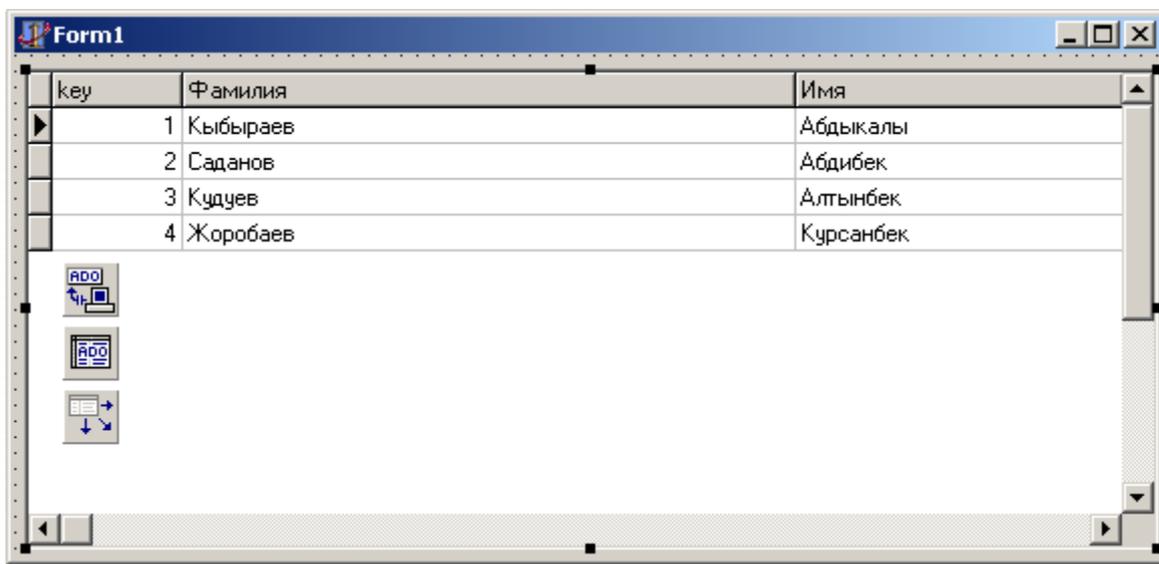


Рис 45. Форма нашего приложения

10. Попробуйте запустить этот пример и создайте несколько строк, отредактируйте уже существующие и удалите что-нибудь. Для вставки строки используйте клавишу **Ins**, а для удаления **Ctrl+Del**.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Архангельский А. Я. Интегрированная среда разработки Delphi. От версии 1 до версии 5. Москва ЗАО «Издательство БИНОМ» 1999 г. 256 с.
2. Владимир Гофман, Анатолий Хомоненко. Delphi 6. Наиболее полное руководство. Санкт-Петербург «БХВ-Петербург» 2002 г.
3. В. Б. Попов. TURBO PASCAL для школьников Москва «Финансы и статистика» 2001 г.
4. Культин Н. Б. Delphi в задачах и примерах. Санкт-Петербург «БХВ-Петербург» 2003 г. 288 с.
5. И. Ю. Баженова. «Delphi 5 Самоучитель программиста»
6. П. Турот, Г. Брент, Р. Багдазиан, С. Тендон. «DELPHI 3», DiaSoft, Киев, 1997 г.
7. Л. М. Климова. Практическое программирование. Решение типовых задач. PASCAL 7.0 Москва 2000 г.
8. Вячеслав Понамарев. «САМОУЧИТЕЛЬ» Базы данных в Delphi 7 2003 г. 223 с.
9. Гофман В., Хомоненко А. Работа с базами данных в Delphi.

Подписано в печать 20.04.2008 г.

Формат 60x84 1/16

Объем: 4 п.л.

Заказ

Тираж: 200 экз

Типография «Кагаз ресурстары»

Г. Ош, ул. Курманжан Датка - 287, тел.: 2-52-50