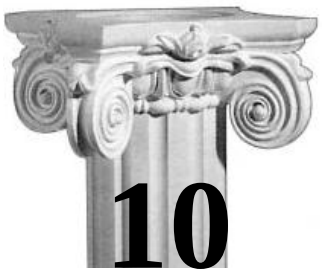


Разработка проектов в Delphi

Если отладка - процесс удаления ошибок, то программирование должно быть процессом их внесения.
Э.Дейкстра

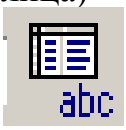
10.1. Стандартные компоненты формы

Начнем создание проекта с рассмотрения чаще всего используемых компонентов. Для удобства опишем их в виде таблицы 10.1.

Таблица. 10.1

Наиболее часто используемые компоненты Delphi.

Компонент	Группа	Описание
Label (метка) 	Standart	Отображение текста, который не изменяется пользователем во время работы.
StaticText (метка с бордюром) 	Additional	В дополнении к метке обеспечивает возможность задания стиля бордюра.
Panel 	Standart	Является контейнером для группирования элементов управления, но может использоваться и для отображения текстовой информации.
Edit 	Standart	Отображение, ввод и редактирование однострочных текстов.
MaskEdit (окно маскированного редактирования) 	Additional	Используется для формирования данных или для ввода символов в соответствии с шаблоном

Memo 	Standart	Отображает ввод и редактирование многострочных текстов.
ListBox (окно списка) 	Standart	Отображение стандартного окна списка windows, позволяющее пользователю выбирать из него пункты.
ComboBox (редактируемый список) 	Standart	Объединяет функции ListBox и Edit. Пользователь может либо ввести текст, либо выбрать его из списка.
StringGrid (таблица) 	Additional	Отображение текстовой информации в таблице из строк и столбцов с возможностью перемещаться по ячейкам и осуществлять выбор.
Checkbox 	Standart	Переключатель для включения/выключения заданного режима.
Radiobutton 	Standart	Переключатель для включения/выключения заданного режима.
Radiogroup 	Standart	Группа переключателей позволяет включать/выключать несколько режимов одновременно.
OpenDialog 	Dialogs	Подключает стандартные диалоги Windows.
Mainmenu (главное меню) 	Standart	Главное меню приложения. Позволяет создать стандартное главное меню для всех Windows приложений.
PageControl 	Win32	Блокнот позволяет на одной форме выводить объекты на разных листах блокнота.
Char	TeeCharStd	Диаграмма позволяет строить диаграммы любой сложности, как по вычисленным

		данным, так и по стандартным функциям.

Для того, чтобы добавить на форму компонент, необходимо в палитре компонент выбрать этот компонент, щелкнув левой клавишей мыши на его пиктограмме, далее установить курсор в точку формы, в которой должен быть левый верхний угол компонента и еще раз щелкнуть левой клавишей мыши. В результате на форме появится компонент стандартного размера. После добавления компонента в инспекторе объектов устанавливаются значения необходимых свойств. Если на форме несколько компонент, то свойства в инспекторе объектов относятся в выделенному компоненту. Если такого компонента нет, то – к форме в целом.

Основные свойства рассмотренных объектов рассмотрим более подробно на примерах в следующих главах.

10.2. Пример построение простого проекта с компонентами: метка, поле редактирования, кнопка.

Для демонстрации возможностей Delphi и технологии визуального проектирования разработаем программу пересчета длины из сантиметров в метры. Работа над новым проектом начинается с создания стартовой формы – окна, которое появляется при запуске приложения.

Форма приложения.

Стартовая форма создается путем изменения свойств формы Form1. Свойства определяют ее внешний вид: размер, положение на экране, текст заголовка, вид рамки… Свойства перечислены на вкладке Properties (свойства) окна инспектора объектов. В левой колонке находятся имена свойств, а в правой – их значения. Размеры устанавливаются в пикселях – точках экрана. Все свойства рассматриваемых компонент описаны в виде

таблиц для наглядности. Общий вид формы с необходимыми объектами представлен на рисунке 10.1.

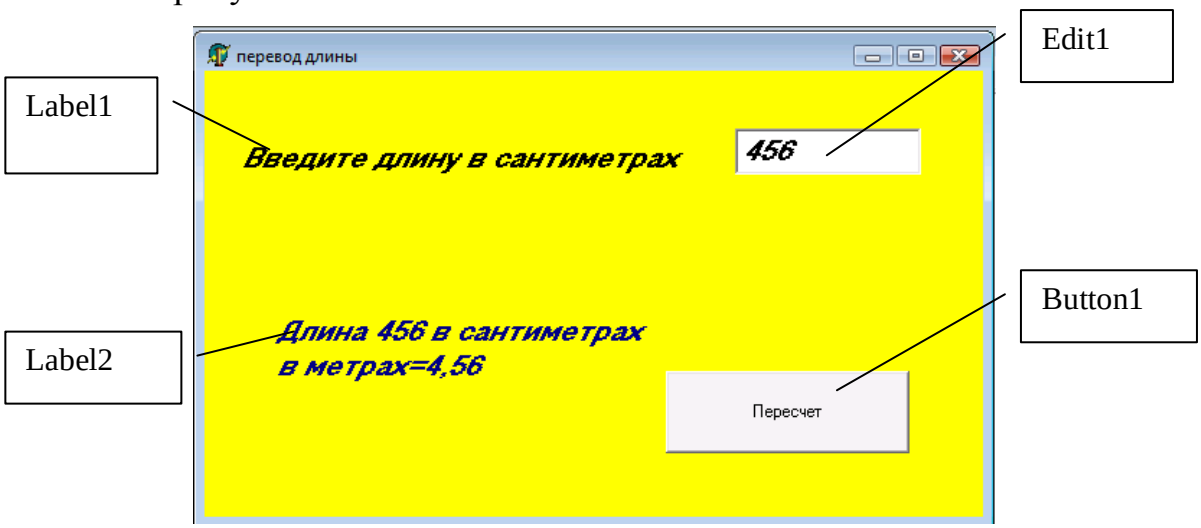


Рис. 10.1. Форма проекта.

Задание свойств начинаем с формы, как основного компонента (табл. 10.2).

Таблица. 10.2.

Основные свойства формы:

Обозначение	Свойство	Значение
Name	Имя формы	Form1
Caption	Заголовок	Перевод длины
Height	Высота	185
Width	Ширина	290
Font.Name	Шрифт	Arial
Font.Size	Размер шрифта	10

Размеры (ширину и высоту) устанавливать в инспекторе объектов не обязательно, можно их изменять с помощью мыши. Все изменения в инспекторе объектов будут отражены автоматически.

Для любого компонента Delphi есть свойство name, которое определяет, как к этому объекту обращаются в тексте программы. По умолчанию это имя соответствует стандартному имени компонента и добавляется номер. Нумерация выполняется автоматически, при каждом добавлении аналогичного компонента - номер увеличивается на единицу.

Программа пересчета длины из сантиметров в метры. Должна получить от пользователя исходные данные – значение длины в сантиметрах. При  работе приложения данные с клавиатуры вводятся в поле редактирования. Добавляем в форму поле редактирования. Из палитры компонент выбирает кнопку «abI» и определим его свойства (табл. 10.3.).

Таблица.10.3.

Основные свойства компонента – поле редактирования.

Обозначение	Свойство	Значение
Name	Имя поля. Используется в программе для доступа к содержимому (тексту) поля	Edit1
Text	Текст, находящийся в поле ввода – редактирования	очищаем
Font	Шрифт, используемый для отображения вводимого текста	Не меняем
ParentFont	Признак наследования свойств шрифта родительской формы. Может принимать два значения True или False. В первом случае шрифт будет такой же как на форме, во втором можно задавать другие параметры шрифта.	True

Помимо полей редактирования окно формы должно содержать поясняющий текст: краткое информационное сообщение. Текст, находящийся в форме называется меткой. Метка вставляется из палитры компонент, кнопка «A».

 В нашу форму добавляем 2 метки: одна для вывода комментария, вторая – для вывода результатов вычислений. Свойства меток представлены в таблице 10.4.

Таблица. 10.4.

Основные свойства компонента - метка.

Обозначение	Свойство	Метка1	Метка2
Name	Имя метки, которое используется в программе для доступа к ней	Label1	Label2
Caption	Текс, который выводится на	Введите	пустое

	форме	длину в сантиметрах	
Autosize	(Автоматический подгон размера) true – автоматически устанавливает размер метки в зависимости от количества символов текста метки, используемого шрифта и его размера. Если надо, чтобы метка представляла собой текст из нескольких строк, то дают значение false и вручную устанавливают значение свойств, определяющих ее размер	False	False

В завершение к форме надо добавить командную кнопку, при щелчке на которую будет выполняться пересчет введенной в поле ввода длины, указанной в сантиметрах, в метры. Из панели компонент выбираем кнопку «ok», определяем ее свойства (табл. 10.5). Щелчок на изображении командной кнопки – это пример того, что в windows называется событием. **Событие** – это то, что происходит во время работы приложения. В Delphi у каждого события есть имя.

Таблица. 10.5.

Основные свойства компонента - кнопка.

Обозначение	Кнопка
Name	Button1
Caption	Пересчет

Реакцией на событие должно быть какое – либо действие. Например, реакцией на событие onckick, произошедшее на кнопке «Пересчет», должен быть пересчет длины из сантиметров в метры.

В Delphi реакция на событие реализуется как процедура обработки события. Задача программиста состоит в написании необходимых процедур обработки событий. Методику создания подобных процедур рассмотрим на примере процедуры обработки события для кнопки.

Сначала выделяем объект, для которого создается процедура обработки, в нашем случае командная кнопка «Пересчет». Затем выбираем вкладку events (событие) окна инспектора объектов. В результате этих действий в окне инспектора объектов появится вкладка со списком событий, которые способен воспринимать выделенный компонент формы (кнопка). В таблице 10.6. представлены некоторые из возможных событий.

Таблица 10.6.

Таблица основных событий.

Событие	Значение
Onclick	Происходит при щелчке кнопки мыши
Ondblclick	Происходит при двойном щелчке кнопки мыши
Onenter	Происходит перед тем, как элемент управления действительно получает фокус
Onerror	Происходит, когда элемент контроля обнаруживает ошибку и не может вернуть информацию об ошибке вызывающей программе
Onexit	Происходит непосредственно перед тем, как элемент управления теряет фокус
Onkeydown	Происходит при нажатии на клавишу. События OnKeyDown иOnKeyPress – это чередующиеся, повторяющиеся события, которые происходят до тех пор, пока не будет отпущена удерживаемая клавиша (в этот момент происходит событие OnKeyUp)
Onkeypress	Происходит, когда пользователь нажимает клавишу, которая генерирует печатаемый символ. Может происходить также при нажатии клавиши печатаемого символа с <Ctrl>. Не происходит при нажатии клавиш <Tab>, <Enter>, клавиш перемещения курсора
Onkeyup	Происходит при отпускании нажатой клавиши
Onmousedown	Происходит при нажатии кнопки мыши
Onmousemove	Происходит при перемещении мыши
Onmouseup	Происходит при отпускании кнопки мыши

В левой колонке вкладки перечислены имена событий, на которые может реагировать кнопка. Если для события определена процедура

обработки события, то в правой колонке, рядом с именем события выводится ее имя. Чтобы создать процедуру обработки события, надо дважды щелкнуть в поле имени процедуры обработки события. В результате открывается окно редактора кода (окно для написания программы) с макетом процедуры обработки события.

```
procedure TForm1.Button1Click(Sender: TObject);
var
a: real;
begin
  a:=strtoint(edit1.text)/100;
  label2.Caption:='Длина '+edit1.Text+' сантиметров'+#13+
' в метрах='+floattostrf(a,ffgeneral,6,2);
end;
```

Delphi автоматически присваивает процедуре обработки события имя. Это имя состоит из двух частей. Первая часть определяет форму, которой принадлежит объект, для которого создается процедура обработки события (form1). Вторая часть имени определяет сам объект и событие (button1click). Имя процедуры – form1.button1click. В тексте процедуры красным цветом выделены команды, которые вводятся программистом, остальные команды создаются автоматически.

Для сохранения проекта из меню **File** выбираем команду **save project as.** Если сохраняем первый раз, появится окно **save unit1 as** создаем папку для сохранения своего проекта. Затем указываем имя файла проекта и модулей, расширение устанавливается автоматически. В результате будет выдаваться два запроса на сохранение (**Если два запроса не вышло – у Вас ошибка!**). Для каждого проекта создается отдельная папка.

Для запуска программы из меню **Run** выбираем команду **Run**. При этом в программе могут быть обнаружены ошибки.

Если ошибок нет, то создается исполнимый файл программы, который можно запустить из windows. Имя исполняемого файла такое же, как имя

проекта расширение ***.EXE**. Delphi помещает этот файл в тот же каталог, где находится файл проекта.

10.3 Сообщения о наиболее типичных ошибках.

Сообщения о синтаксических ошибках содержат 3 пункта **errors** – количество ошибок, **warnings** – количество предупреждений и **hints** – количество подсказок.

Количество ошибок обычно равно двум. Первая обнаруженная синтаксическая ошибка, вторая фатальная – невозможность генерации исполняемой программы.

Количество предупреждений и подсказок не влияют на исполнение программы. Чаще всего встречается предупреждение: «Имя переменной объявлено, но не используется», подсказка – «Не присвоено начальное значение переменной».

Сам текст ошибок выдается в нижней строке редактора кода, в тексте программы выделяется строка, в которой произошла ошибка. Если в сообщении об ошибке в конце стоит «…», то сообщение полностью не уместилось. Чтобы его просмотреть, курсор мыши устанавливают на слово **error**, стоящее в начале строки.

При стандартной установке пакета окно результатов компиляции на экран не выводится. Чтобы вывести его на экран, из меню **Tools** выбираем команду **environment options**, в которой вкладку **preferences** отключаем флажок **show compiler progress**, который находится в группе **compiling and running**. Наиболее часто встречающиеся ошибки представлены в таблице 10.7.

Таблица 10.7.

Чаще всего встречающиеся сообщения компилятора.

Сообщение компилятора	Вероятная причина
Undeclared identifier (необъявленный идентификатор)	- используется переменная, не объявленная в разделе var программы; - ошибка при написании имени объявленной переменной, например, объявлена переменная summa, а тексте программы написано suma:=… -ошибка при написании имени инструкции, например, вместо const написано conts.
Unterminated string (незавершенная строка)	При записи строковой константы не поставлена завершающая кавычка
Incompatible types … and… (несовместимые типы)	В инструкции присваивания тип выражения не соответствует или не может быть приведен к типу переменной, получающей значение выражения
Missing operator or semicolon (отсутствует оператор или точка с запятой)	Не поставлена точка с запятой после инструкции программы

10.4. Окончательная настройка приложения.

После того как программа отлажена, необходимо выполнить ее окончательную настройку: выбрать запускающую форму, назначить приложению значок, который будет изображать исполняемый файл приложения в папке или на рабочем столе, подключить файл справочной системы и т.д.

Для окончательной настройки приложения из меню **Project** выбирают команду **Options**. В результате открывается диалоговое окно, представленное на рисунке 10.2.

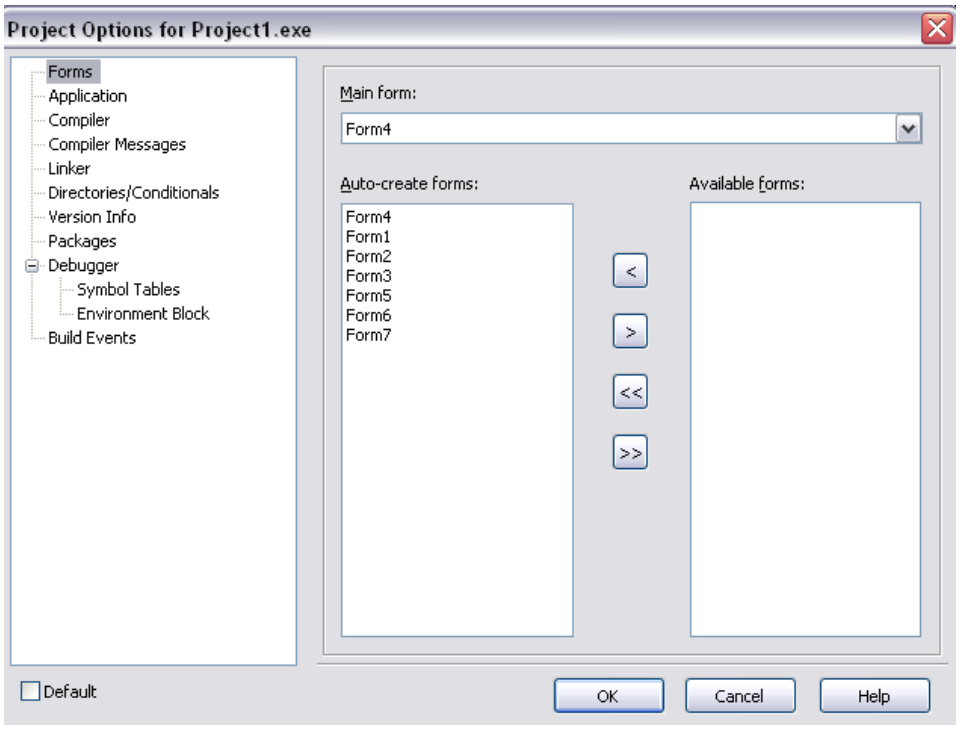


Рис. 10.2. Окно настройки приложения.

В левой части окна выбирается объект настройки, а в правой для выбранного объекта режимы. Для форм (рис.10.2.) можно выбрать запускающую форму (**main form**), автоматически создаваемые формы (**auto-create forms**) и возможные (**available forms**). При отладке проекта удобно пользоваться этим режимом для проверки работоспособности каждой отдельной формы.

На рисунке 10.3. представлено режим настройки приложения.

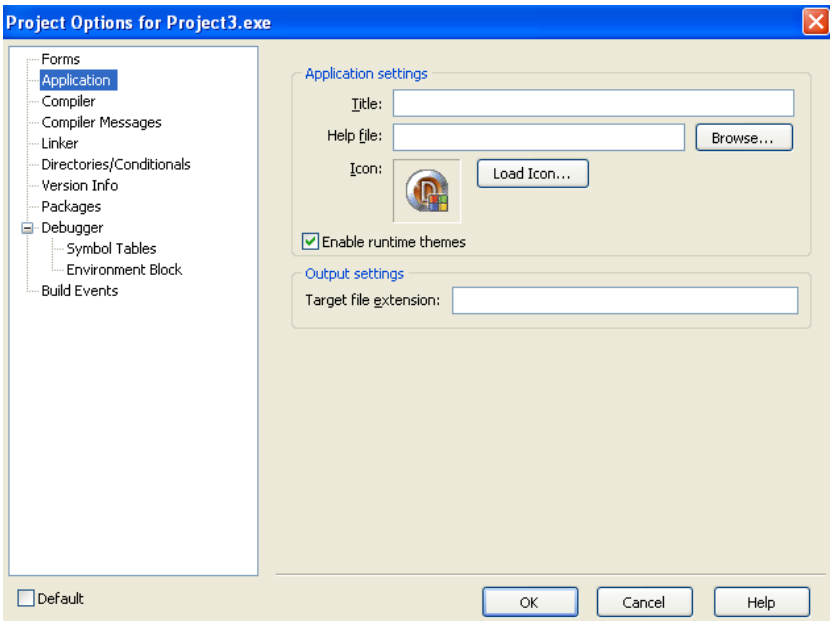


Рис. 10.3. Окно настройки приложения.

Поле **Title** задает название приложения. В поле **Help file** можно указать с каким справочным файлом будет связано приложение. Указанный справочный файл можно будет вызывать, как обычную справку через клавишу F1. Поле **Icon** показывает пиктограмму приложения. Пиктограмма подключается через кнопку **load icon**.

10.5. Использование в проекте стандартных компонент.

10.5.1. Работа со списками

На вкладке **standart** панели компонент есть два компонента, соответствующих спискам. **Listbox** – обычный список, **Combobox** – раскрывающийся список. Независимо от вида списка, принципы работы одинаковы. Рассмотрим основные свойства и режимы работы с ними.

 - ListBox

 - ComboBox

name – имя, используемое в программе listbox1 или combobox1 соответственно.

items (элементы списка) – может использоваться как в программе, так и инспекторе объектов. Определяет значения элементов списка. При создании через инспектор объектов открывается дополнительное окно, в котором вводятся значения элементов, причем каждое значение с новой строки. Все элементы нумеруются, начиная с нуля.

Itemindex - номер выбранного элемента списка. Номер первого элемента списка равен 0, если не выбран ни один элемент номер равен –1. Может использоваться только в программе.

Данное свойство удобнее использовать в сочетании с командой выбора. Например,

Case listbox1.itemindex of

0: команда 1;

1: команда 2;

2: и т.д.;

end;

listbox1.items[n]:=<выражение>; - присваивает значение элементу с номером n. Элемент с таким номером должен существовать.

listbox1.items.count - количество элементов в списке, может использоваться только в программе.

combobox1.items.add('строковое выражение'); - добавляет элемент в список. Элемент будет добавлен в конец списка.

combobox1.items.delete(n); - удаляет n - ую по счету строку.

combobox1.items.clear; - очищает список.

combobox1.items.loadfromfile('имя'); - загружает данные из текстового файла в список. Загружается весь файл целиком. Возможно использовать только текстовые файлы.

combobox1.items.savetofile('имя'); - сохраняет элементы списка в текстовом файле.

combobox1.text - значение элемента, который выбран в списке. Данное свойство есть только у компонента **combobox**.

Например: дан текстовый файл, содержащий информацию о продаже товаров: товар, количество. Вывести в список все товары, количество которых больше введенного значения в поле редактирования.

Форма будет иметь вид, представленный на рисунке 10.4.

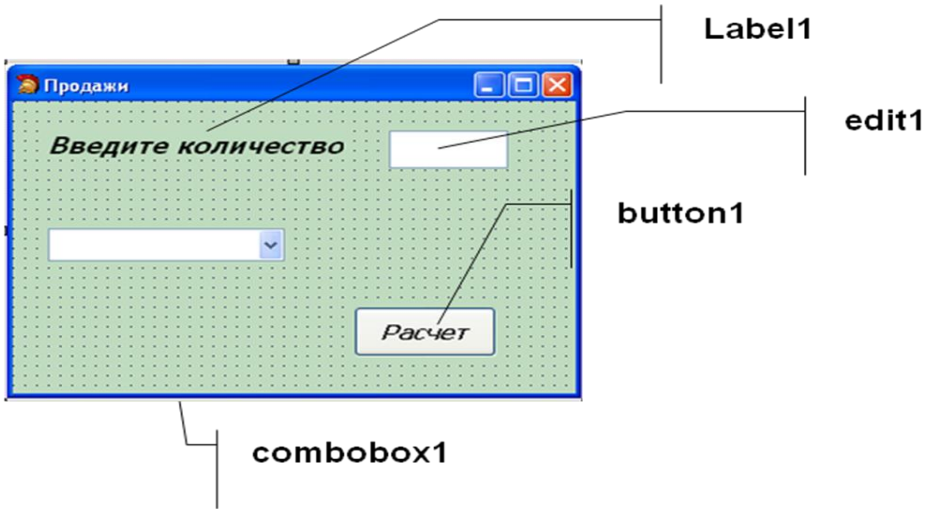


Рис. 10.4. Интерфейс формы.

Структура файла представлена на рисунке 10.5.

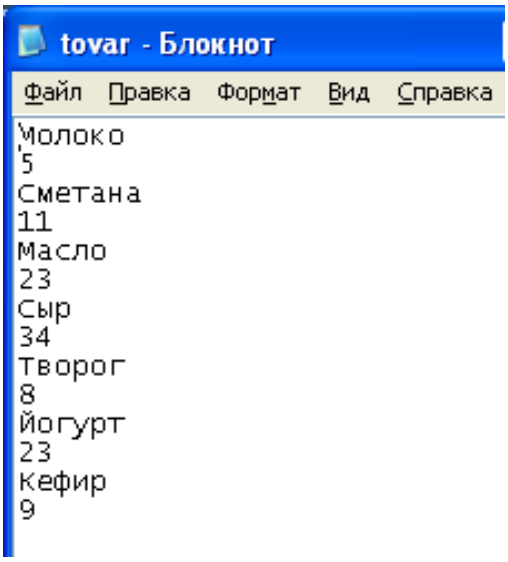


Рис. 10.5. Содержание текстового файла.

Структуру данных и используемые переменные удобнее описывать в интерфейсной части. Аналогичный принцип описания будет использоваться во всех задачах рассмотренных ниже.

```
unit Unit1;
interface
…
type
  TForm1 = class(TForm)
  end;
  tovar=record
```

```
    nt:string[15];
    kol: integer;
  end;
var
  Form1: TForm1;
  tov: tovar;
  f: textfile;
implementation
…
// Заполнение списка
procedure TForm1.Button1Click(Sender: TObject);
begin
  assignfile(f,'G:\tovar.txt');
  reset(f);
  combobox1.items.Clear;
  if edit1.text='' then begin
  showmessage('введите количество'); exit;end;
  while not eof(f) do begin
   readln(f,tov.nt);
   readln(f,tov.kol);
   if tov.kol>strtoint(edit1.Text) then combobox1.items.Add(tov.nt);
  end;
  closefile(f);
end;
end.
```

В начале процедуры список очищается. Это необходимо сделать, т.к. в противном случае при каждом нажатии на кнопку формы в список будут добавляться новые элементы.

Решим еще одну задачу. Дан текстовый файл, содержащий информацию о сотовых телефонах: название модели, цена, изображение модели.

Сформировать список из моделей телефонов. При выборе модели выводить на форму цену и изображение соответствующей модели. Форма содержит компоненты, представленные на рисунке 10.6.

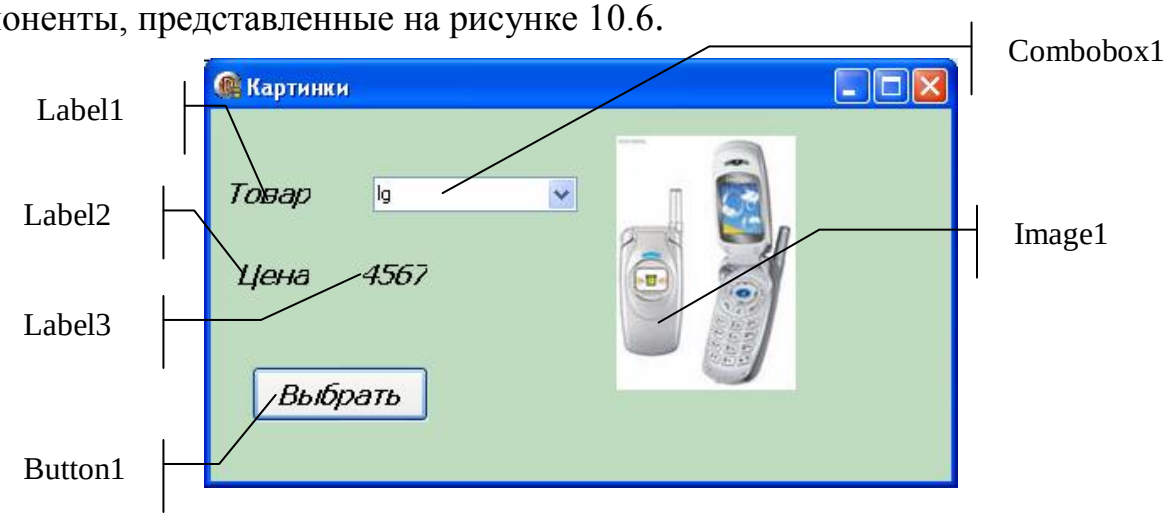


Рис. 10.6. Интерфейс формы.

Информация в файле будет храниться в виде, представленном на рисунке 10.7.

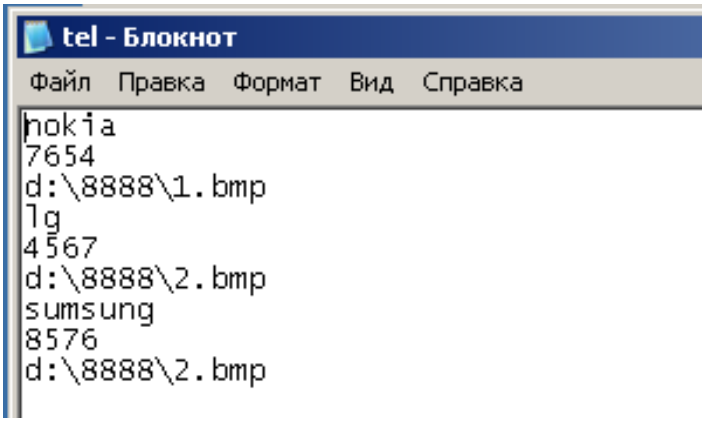


Рис. 10.7. Структура текстового файла.

В интерфейсной части модуля будет описана структура данных и переменные:

```
price=record
   tov: string[10];
   cena: real;
   kar: string[30]; //путь к файлу с изображением телефона
 end;
var
  f: textfile;
```

```
 prl:price;
```

Процедура заполнения списка будет выполняться при активизации формы.

```
procedure TForm1.FormActivate(Sender: TObject);
begin
assignfile(f,'g:\студенты\2 курс\888\tel.txt');
reset(f);
combobox1.Items.Clear;
while not eof(f) do begin
 readln(f,prl.tov);
 readln(f, prl.cena);
 readln(f, prl.kar);
 combobox1.Items.Add(prl.tov);
end;
closefile(f);
end;
```

Процедура выводы цены выбранного товара и его изображения будет выполняться при нажатии на кнопку.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
if combobox1.Text='' then begin showmessage('Надо выбрать товар');
                        exit;end;
assignfile(f,'g:\студенты\2 курс\888\tel.txt');
reset(f);
while not eof(f) do begin
  readln(f,prl.tov);
  readln(f, prl.cena);
  readln(f, prl.kar);
  if trim(combobox1.text)=trim(prl.tov) then begin
   label3.Caption:=floattostr(prl.cena);
```

```
    image1.Picture.LoadFromFile(prl.kar);
  end;
end;
closefile(f);
end;
```

Элементы списков могут быть отсортированы в алфавитном порядке. Наличие или отсутствие сортировки определяет свойство **Sorter** типа **Boolean**. Если это свойство имеет значение **False**, то элементы располагаются в порядке их поступления в список. Если оно имеет значение **True**, то элементы списка автоматически сортируются. Это свойство может быть задано как в инспекторе объектов, так и в программе.

```
listbox1.Sorted:=true;
```

Для обычного списка (без сортировки) может быть использовано свойство одновременного выбора нескольких элементов. По умолчанию выбирается только один элемент. Для выбора двух и более элементов свойству **MultiSelect** типа **Boolean**, управляющему возможностью выбора нескольких строк, устанавливается значение **True**.

Если установлен режим выбора нескольких элементов, то свойство **ExtendedSelect** Типа **Boolean** управляет способом выбора нескольких элементов. Если это свойство установлено в **False**, то новый элемент можно добавить только с помощью мыши. В противном случае и с помощью мыши, и с помощью клавиш **Shift, Ctrl**.

Свойство **SelCount** целочисленного типа возвращает количество выбранных элементов. Свойство **Selected [i: integer]: Boolean** определяет массив выбранных элементов. Если элемент выбран, то соответствующий элемент этого массива примет значение True.

Например, вывести выбранные элементы из списка в метку. Форма будет иметь вид, представленный на рисунке 10.8.

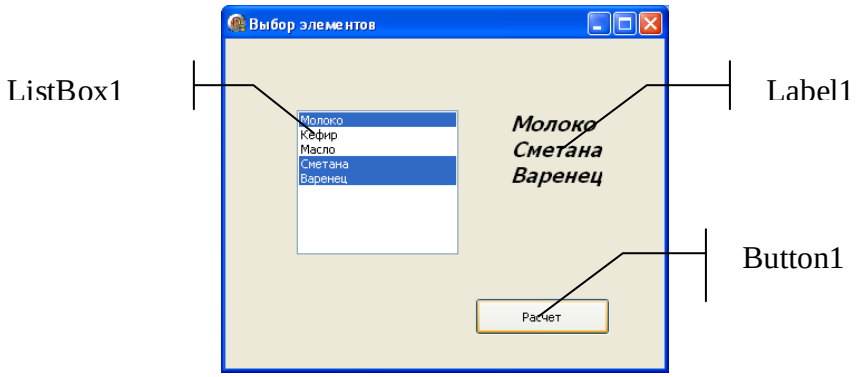


Рис. 10.8. Интерфейс формы.

Процедура будет иметь вид:

```
procedure TForm1.Button1Click(Sender: TObject);
var
i: integer;
begin
label1.caption:='';
  for I := 0 to Listbox1.items.Count - 1 do
    if listbox1.Selected[i] then label1.Caption:=label1.caption+listbox1.Items[i]+#13;
end;
```

10.5.2. Компонент – memo.

Вкладка **standart**. Основные свойства и методы компонента.

Name – имя, используемое в программе (**memo**).

Text – все содержимое компонента независимо от количества заполненных строк. Может использоваться только в программе.

Lines – отдельные строки компонента. Свойство можно задавать через инспектор объектов или через программу. Все строки нумеруются, начиная с нуля.

Memo1.lines[n]:=<выражение>; - присваивает значение n - му элементу. Элемент с таким номером должен существовать.

Memo1.lines.clear; - очищает весь компонент.

Memo1.lines.add('строка'); - добавляет новую строку в компонент. Строка будет добавлена в конец.

Memo1.lines.loadfromfile('имя'); - загружает в компонент данные из текстового файла. Информация из файла будет загружена целиком.

Memo1.lines.savetofile(‘имя’); - сохраняет данные из компонента в текстовом файле.

Например: дан текстовый файл, содержащий информацию о продаже товаров: товар, количество. Вывести в поле memo все товары, количество которых больше введенного значения в поле редактирования.

Структура текстового файла будет аналогична предыдущему примеру (рис. 10.5.), интерфейс формы представлен на рисунке 10.9.

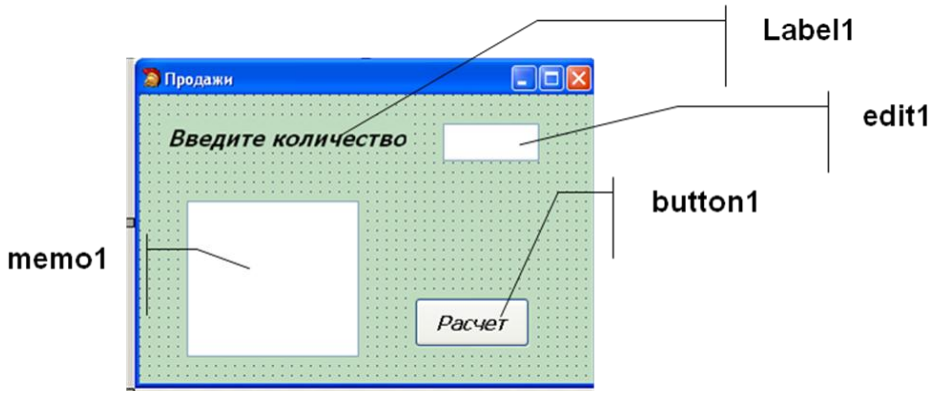


Рис. 10.9. Элементы интерфейса формы.

Описание переменных и данных типа запись аналогично примеру предыдущего параграфа. Процедура будет следующей:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  assignfile(f,'G:\tovar.txt');
  reset(f);
  memo1.Lines.Clear;
  if edit1.text='' then begin
  showmessage('введите количество'); exit;end;
  while not eof(f) do begin
   readln(f,tov.nt);
   readln(f,tov.kol);
   if tov.kol>strtoint(edit1.Text) then memo1.Lines.Add(tov.nt);
  end;
  closefile(f);
end;
```

10.5.3. Работа с переключателями

На вкладке **standart** есть два компонента, соответствующие переключателям. Рассмотрим свойства одного из них (табл. 10.8), т.к. принципы работы с компонентами аналогичны.

 - **checkbox**

 - **radiobutton**

Компоненты позволяют изменять порядок расчетов в программе в зависимости от того, включены они или нет.

Таблица 10.8.

Основные свойства компонента - флажок (checkbox)

Свойство	Обозначение
Имя компонента. Используется в программе для доступа к свойствам компонента.	Name
Текст, поясняющий назначение переключателя.	Caption
Определяет состояние, внешний вид переключателя. Если переключатель выбран (в квадратике, изображающем переключатель, находится «галочка»), то checked=true. Если переключатель не выбран (в квадратике нет «галочки»), то checked=false.	Checked
Определяет состояние переключателя. В отличие от свойства checked позволяет различать выбранное, невыбранное и промежуточное состояния. Состояние переключателя определяют константы: cbchecked (выбран), cbgrayed (серый, неопределенное состояние) и cbunchecked (не выбран).	State
Определяет, может ли переключатель быть в промежуточном, неопределенном состоянии. Если allowgrayed=false, то переключатель может быть только выбранным или невыбранным. Если allowgrayed=true, то промежуточное состояние допустимо.	Allowgrayed

Например, при включенном переключатели, вывести вычисленное максимальное значение элемента массива.

If checkbox1.checked then label2.caption:='максимум='+inttostr(max);

Можно вместо одного переключателя, использовать группу переключателей – **Radiogroup**. Основные свойства данного компонента представлены в таблице 10.9.

Таблица 10.9.

Основные свойства компонента Radiogroup.

Свойство	Обозначение
Имя компонента. Используется в программе для доступа к свойствам компонента.	Name
Текст, поясняющий назначение переключателя. Указывается над всей группой переключателей.	Caption
Количество колонок, на которые разбит вывод переключателей.	columns
Перечисление значений, которые выбираем. Нумерация пунктов начинается с нуля.	items
Какой пункт выбран	itemindex

10.5.4 Работа со строковыми таблицами.

Элемент **StringGrid** (строковая таблица) находится на дополнительной (**Additional**) странице палитры компонентов. Этот компонент представляет собой двумерный строковый массив. Основные свойства представлены в таблице 10.10.

Таблица 10.10.

Основные свойства компонента StringGrid.

Свойство	Назначение
ColCount	Определяет число колонок таблицы
RowCount	Определяет число строк таблицы
Col	Колонка активной ячейки
Row	Строка активной ячейки
ColWidths	Изменяет ширину текущей колонки
RowHidghts	Изменяет высоту текущей строки
DefaultColWidths	Определяет ширину всех колонок
DefaultRowHidghts	Определяет высоту всех строк

DefaultDrawing	Определяет, будет ли автоматически прорисовываться рамка (TRUE)		
EditorMode	Определяет, возможно ли редактирование содержимого (TRUE)		
Options	Определяет внешний вид таблицы		
	goFixedVerLine	True	Отображаются вертикальные линии между колонками в фиксированных блоках
	goFixedHorLine	True	То же горизонтальные
	goVerLine	True	Отображаются вертикальные линии между колонками
	goHorLine	True	То же горизонтальные
	goRangeSelect	True	Можно выделить блок ячеек (если нет goEditing)
	goDrawFocusSelected	True	Активная ячейка не меняет цвет
	goRowSizing	True	Размеры строк можно менять при работе приложения (за исключением фиксированных)
	goColSizing	True	То же для колонок
	goRowMoving	True	Можно переместить строку на новое место с помощью мыши
	goColMoving	True	То же для колонок
	goEditing	True	Можно редактировать текст (но не выделять блок!)
	goTabs	True	Перемещаться по ячейкам можно через «TAB»
	goRowSelect	True	Можно выделять отдельные строки
	goAlwaysShowEditor	True	Таблица автоматически редактируется (иначе f2 или Enter)

	goThumboTracking	True	Видимая часть таблицы прокручивается синхронно бегунку
FixedColor	Определяет цвет фиксированных элементов таблицы		
FixedCols	Определяет число фиксированных колонок		
FixedRows	То же для строк		
TopRow	Определяет, какая строка первая сверху (если нет фиксированных строк)		
LeftCol	То же для колонок		
GridHeight	Высота таблицы в пикселах		
GridWidth	То же для ширины		
GridLineWidth	Определяет толщину разделительной линии		
Selection	Описывает прямоугольник, содержащий выделенный блок таблицы		
VisibleColCount	Число видимых колонок (за исключением фиксированных)		
VisibleRowCount	То же для строк		
Objects	Массив объектов, соответствующий ячейкам таблицы StringGrid.Objects[10,3]:=333 (запись числа в ячейку)		
Cells	Доступ к тексту из заданной ячейки. Строки и столбцы таблицы нумеруются с нуля. Ячейка cells[j,i] находится на пересечении i-й строки и j-го столбца.		
rows[n]	строка с номером n.		
Cols[n]	столбец с номером n.		
Listbox1.items.assign (stringGrid.cols[3]);	запись данных из столбца с номером 3 в список.		

Рассмотрим несколько примеров работы со строковой таблицей:

1. Добавление новой строки вниз таблицы.

```
Begin
  Sg.rowcount:=sg.rowcount+1;
  Sg.rows[sg.rowcount-1].clear;  //очистка добавленной строки
End;
```

2. Удаление текущей строки (строки, на которой стоит курсор) из таблицы. В результате выполнения процедуры все строки, начиная с текущей

сдвигаются вверх, последняя строка удаляется. Удаление выполняется, если в таблице более двух строк.

```
var n: integer;
begin
 if sg.rowcount=2 then exit;
 for n:=sg.row to sg.rowcount-2 do sg.rows[n]:=sg.rows[n+1];
sg.rowcount:=sg.rowcount-1;
end;
```

3. Фрагмент процедуры, которая суммирует значения третьего столбца таблицы. Sg – значение свойства name таблицы Stringgrid.

```
Var
  i,s:integer;
begin
 s:=0;
for i:=1 to sg.rowcount-1 do
  s:=s+strtoint(sg.cells[3,i]);
end;
```

Рассмотрим примеры программ работы с файлами.

1. Дан текстовый файл, содержащий информацию о продаже сотовых телефонов: модель телефона, дата продажи, количество. Загрузить информацию из файла в строковую таблицу, после внесения изменений сохранить обновленные данные в файл. На рисунке 10.10. представлена форма без заполнения данными и с данными.

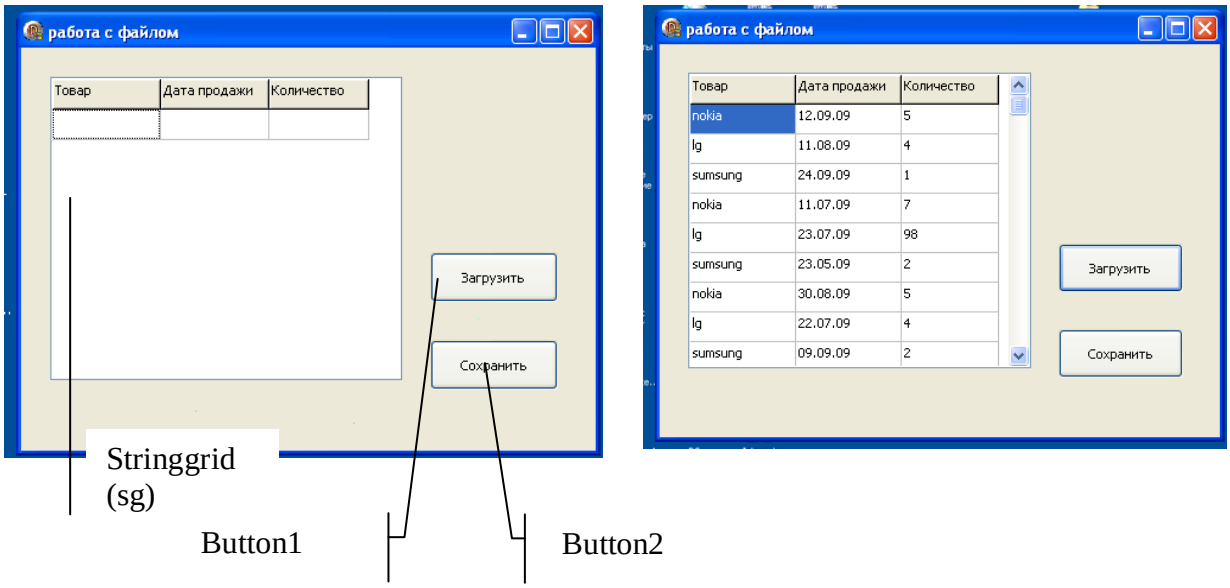


Рис. 10.10. Интерфейс формы.

Структура текстового файла представлена на рисунке 10.11.

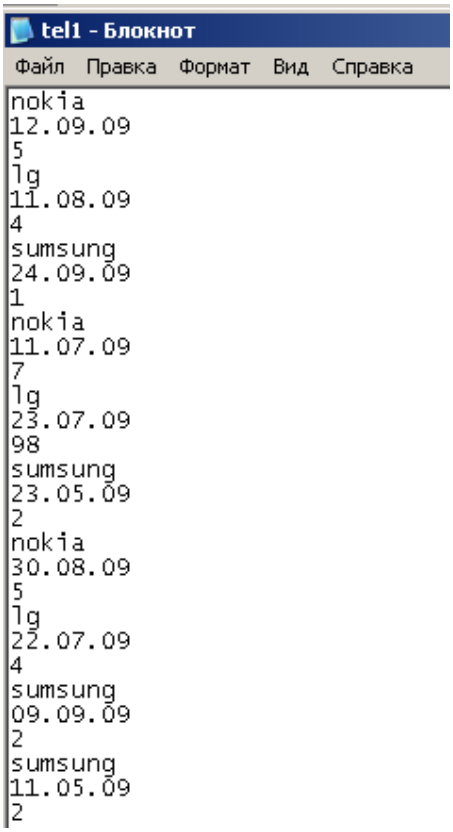


Рис. 10.11. Структура текстового файла.

Зададим свойства для компонента – строковая таблица (табл.10.11)

Таблица 10.11.

Свойства компонента StringGrid.

Свойство	Значение
name	sg
rowcount	2
colcount	3
fixedcols	0
fixedrows	1
goediting	true

В интерфейсной части модуля будет описана структура данных и переменные:

```
prodajy=record
   model:string[15];
   data:string[8];
   kol:integer;
  end;
var
   f:textfile;
  prod:prodajy;
  i: integer;
```

При активизации формы заполняем название столбцов таблицы:

```
procedure TForm1.FormActivate(Sender: TObject);
begin
  sg.cells[0,0]:='Товар';
  sg.cells[1,0]:='Дата продажи';
  sg.cells[2,0]:='Количество';
end;
```

Процедура заполнения таблицы:

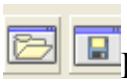
```
procedure TForm1.Button1Click(Sender: TObject);
begin
 sg.RowCount:=2;
 assignfile(f,'d:\8888\tel1.txt');
 reset(f);
 i:=1;
 while not eof(f) do begin
  readln(f,prod.model);
 readln(f,prod.data);
 readln(f,prod.kol);
  sg.cells[0,i]:=prod.model;
  sg.cells[1,i]:=prod.data;
  sg.cells[2,i]:=inttostr(prod.kol);
  i:=i+1;
  sg.RowCount:=sg.RowCount+1;
 end;
 sg.RowCount:=sg.RowCount-1;
 closefile(f);
end;
```

Процедура сохранения данных в файл.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
 assignfile(f,'d:\8888\tel1.txt');
 rewrite(f);
 for I := 1 to sg.rowcount - 1 do begin
  prod.model:=sg.cells[0,i];
  prod.data:=sg.cells[1,i];
  prod.kol:=strtoint(sg.cells[2,i]);
  writeln(f,prod.model);
  writeln(f,prod.data);
```

```
   writeln(f,prod.kol);
end;
 closefile(f);
end;
```

10.5.5. Подключение стандартных диалогов

 На странице **Dialogs** имеется палитра, содержащая 10 стандартных диалогов (невизуальные объекты). Они реализуют диалоги общего назначения. Эти диалоги используются многими приложениями Windows для выполнения таких операций, как открытие, сохранение и печать файлов, поэтому их часто называют стандартными. Все эти диалоги работают, например, в Word. Чаще всего используют следующие диалоги:

Opendialog – выбор открываемого файла.

Savedialog – выбор сохраняемого файла.

Openpicturedialog – выбор открываемого графического файла.

Savepicturedialog – выбор сохраняемого графического файла.

Fontdialog – настройка параметров шрифта.

Colordialog – выбор цвета.

Printdialog – вывод на принтер.

Чтобы стандартный диалог мог использоваться, соответствующий ему компонент должен быть помещен на форму и его свойствам установлены необходимые значения. После этого следует связать вызов диалога с каким-либо событием. Чаще всего таким событием является выбор пункта меню или нажатие кнопки.

Для вызова любого стандартного диалога используется метод **execute** – функция, возвращающая логическое значение. При закрытии диалога кнопкой OK (открыть или сохранить) функция возвращает значение **true**, а при отмене диалога – значение **false**.

После закрытия стандартного диалога он возвращает через свои свойства значения, выбранные или установленные в процессе диалога. Например, при открытии файла возвращаемым значением является имя возвращаемого файла (**opendialog1.filename**), а при выборе цвета – новый цвет (**colordialog1.color**).

В качестве примера рассмотрим диалоги открытия и сохранения файлов. Основными свойствами компонентов **opendialog** и **savedialog** являются:

Filename:string – указывает имя и полный путь файла, выбранного в диалоге. Имя файла отображается в строке редактирования с названием **Имя файла** и является результатом диалога.

Title:string – задает заголовок окна. Если это свойство не установлено, то по умолчанию используется заголовок **Открытие файла** для **opendialog** и заголовок **Закрытие файла** - для **savedialog.** Устанавливается в инспекторе объектов.

Initialdir:string – определяет каталог, содержимое которого отображается при вызове окна диалога. Если каталог не задан, то отображается содержимое текущего каталога.

Defaultext:string – задает расширение, автоматически подставляемое к имени файла, если пользователь не указал расширение имени.

Filter:string – задает маски имен файлов, отображаемых в раскрывающемся списке под названием Типы файлов. В окне диалога видны имена файлов, которые совпадают с указанной маской. По умолчанию пустая строка, т.е. все файлы. ***.txt;*.doc** (если два или больше). Устанавливается в инспекторе объектов (рис. 10.12).

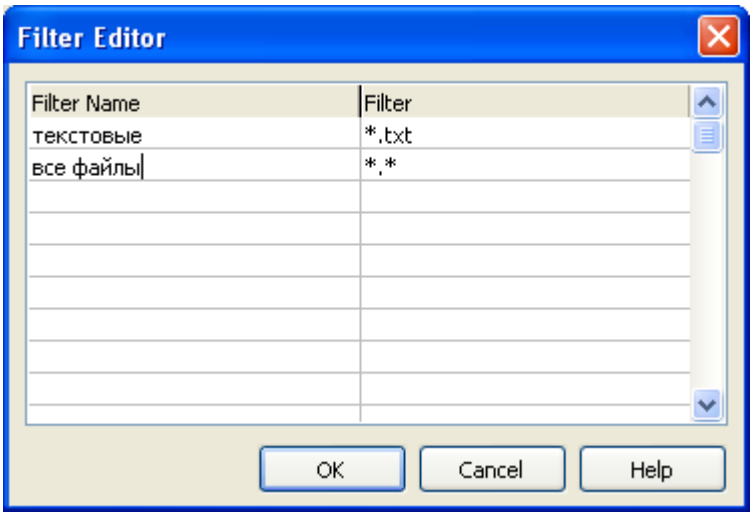


Рис.10.12. Вид окна для определения фильтра.

filterindex:integer – указывает какая из масок фильтра отображается в списке. По умолчанию значение равно единицы, и используется первая маска. Устанавливается в инспекторе объектов.

Options:topenoptions – используется для настройки параметров, управляющих внешним видом и функциональными возможностями диалога. Каждый параметр (флажок) может быть установлен или сброшен. Основные свойства:

Ofallowmultiselect – из списка можно одновременно выбрать более одного файла.

Ofcreateprompt – при вводе несуществующего файла выдается запрос на создание файла.

Ofnolongnames – имена файлов отображаются как короткие (имя содержит до восьми символов и 3 символа расширение).

Например, открытие файла через диалог записывается так:

if opendialog1.execute then assignfile(f,opendialog1.filename);

Например, информация о реализации сотовых телефонов хранится в двух текстовых файлах. Файл продажи имеет структуру: модель телефона, дата продажи и количество. Файл прайс – лист имеет структуру: модель телефона, цена и изображение товара. Определить стоимость каждой сделки. Все обозначения взяты из предыдущих примеров. Формы приложения представлена на рисунке 10.13.

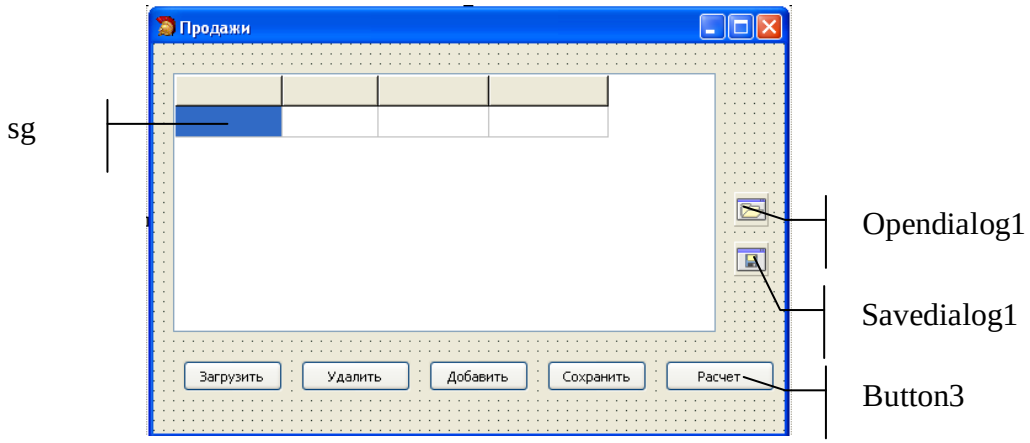


Рис. 10.13. Интерфейс формы.

В интерфейсной части модуля будет описана структура данных и переменные:

```
prodajy=record
   nt: string[15];
   fir:string[15];
   dat:string[8];
   kol: integer;
  end;
  tovar=record
   ntov: string[15];
   cena: integer;
   kar: string[30];
  end;
var
  f,f1: textfile;
  prod: prodajy;
  tov: tovar;
  i: integer;
```

Процедура расчета:

```
procedure TForm1.Button3Click(Sender: TObject);
begin
if sg.Cells[0,1]='' then begin
```

```
showmessage('Заполните таблицу'); exit; end;
sg.ColCount:=sg.colcount+1;
sg.cells[4,0]:='Стоимость';
if opendialog1.Execute then begin
  assignfile(f1,opendialog1.FileName); reset(f);
  for i:=1 to sg.rowcount - 1 do begin
    reset(f1);
    while not eof(f1) do begin
     readln(f1,tov.ntov);
     readln(f1,tov.cena);
     readln(f1,tov.kar);
     if trim(tov.ntov)=trim(sg.Cells[0,i]) then
      sg.cells[4,i]:=inttostr(strtoint(sg.cells[3,i])*tov.cena);
     end;
  end;
 end;
 closefile(f1);
end;
```

В начале, выполняется проверка, если таблица не заполнена исходными данными из файла продажи, то появится соответствующее сообщение и прервется выполнение процедуры.

10.5.6. Построение графиков

Компонент – диаграмма **TChart** вкладка **TeeChart Std** позволяет строить диаграммы любой сложности.

Свойства устанавливаются через **Editing Chart**, которое вызывается из инспектора объектов командой **Legend**.

Для каждой диаграммы можно установить следующие основные параметры:

- тип;
- название;
- оси;
- легенду;
- источник данных.

У данного объекта много свойств, которые указываются в окне редактора (рис. 10.14.).

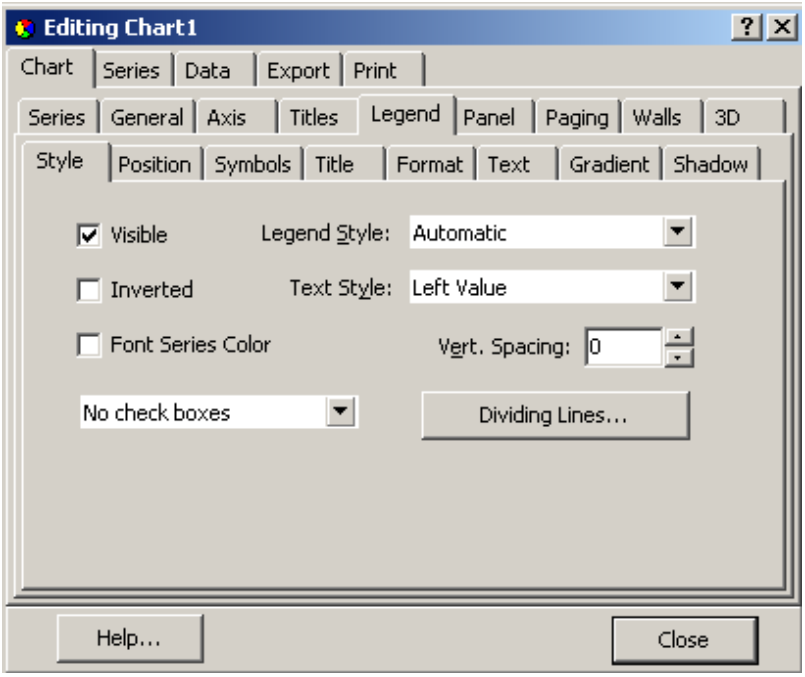


Рис. 10.14. Окно редактора диаграмм.

Все свойства сгруппированы по вкладкам в зависимости от назначения. Основных вкладок пять:

Char – устанавливает общие свойства графика;

Series – форматируются отдельные линии данных, отображаемые на графике;

Data – добавляются и изменяются данные, которые будут отображены на графике;

Export – сохраняются данные в различных форматах;

Print – управление печатью графика.

Построение графика начинают с выбора типа диаграммы и количества рядов данных. Для этого с вкладки **Char** выбирают режим **Series** и через кнопку Добавить определяют нужную диаграмму (рис. 10.15.). Возможное количество рядов данных зависит от выбранного типа диаграммы. Например, для круговой диаграммы допустим только один ряд данных.

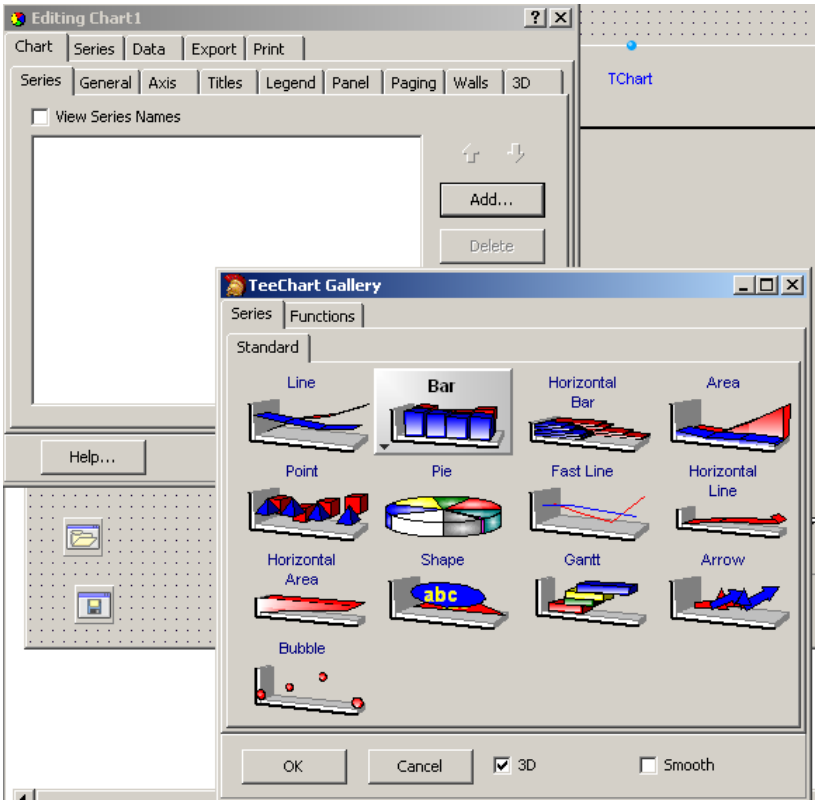


Рис. 10.15. Выбор диаграммы и количество рядов данных.

После этого задаем параметры названия диаграммы. Для этого с вкладки **Char** выбирают режим **Title** (рис. 10.16.). В этом режиме можно определить стиль, выравнивание, местоположения названия и пр.

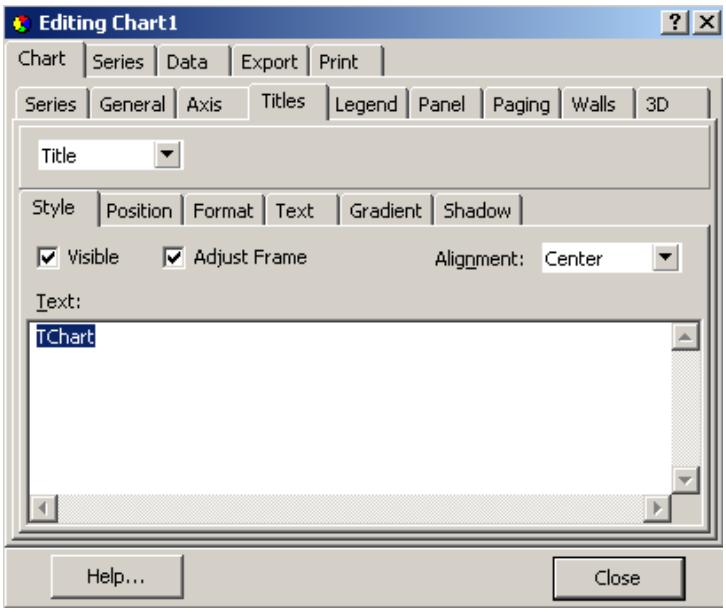


Рис. 10.16. Окно настройки названия диаграммы.

Источник данных выбирается на странице **Series** вкладка **datasource** (рис. 10.17.). В качестве источника данных используются: значения, определяемые функцией; случайные значения или значения, вводимые программно. Если данные для построения диаграммы будут вычисляться в процессе работы проекта, выбирают **Random** с параметром 0.

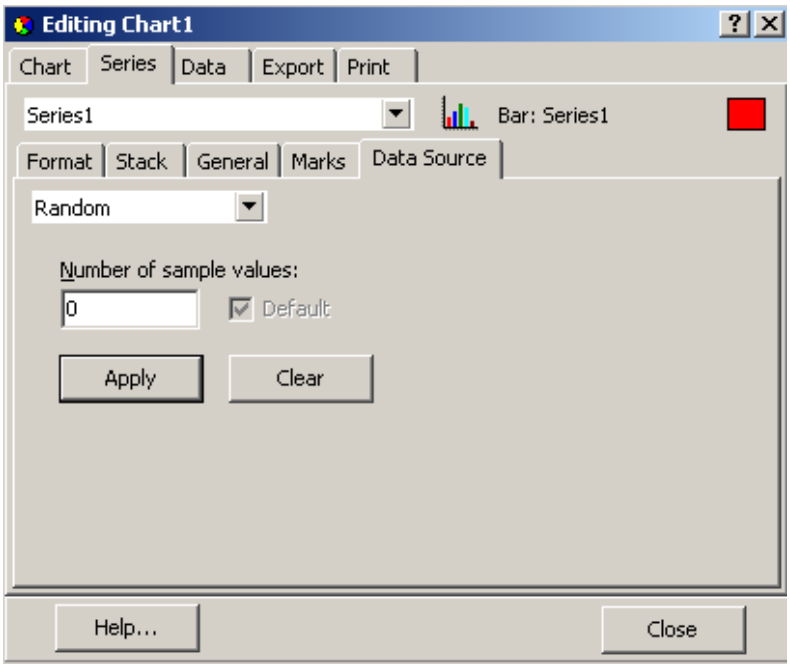


Рис. 10.17. Окно настройки источника данных.

Основные команды построения диаграммы:

chart1.Title.text.add(‘текст'); - задает название диаграммы;

chart1.SeriesList[n].title:=<выражение>; - задает легенду диаграммы;

chart1.title.text.clear; - очищает название диаграммы;

chart1.series[n].clear; - очищает указанный ряд диаграммы;

chart1.SeriesList[n].add(значение,подпись,цвет); - строит указанный ряд диаграммы.

При использовании последней команды необходимо указывать цвет, который определяется встроенными константами, указанными в таблице 10.12.

Таблица 10. 12.

Константы, задающие цвет

константа	цвет
Claqua	Ярко-голубой
Clblack	Черный
Clblue	Голубой
Clfuchsia	Сиреневый
Clgray	Серый
Clgreen	Зеленый
Clime	Ярко-зеленый
Clmaroon	Темно-красный
Clnavy	Темно-синий
Clolive	Оливковый
Clpurple	Фиолетовый
Clred	Красный
Clsilver	Серебряный
Clteal	Сине-зеленый
Clwhite	Белый
clyellow	Желтый

Например, даны сведения о реализации сотовых телефонов в двух текстовых файлах. Первый файл – прайс - лист: название модели телефона, цена, изображение товара. Второй файл – продажи: модель телефона, дата продажи, количество. Построить диаграмму общей стоимости реализации по

моделям телефонов. На рисунке 10.18. представлена форма в режиме работы, на рисунке 10.19. форма в режиме редактирования.

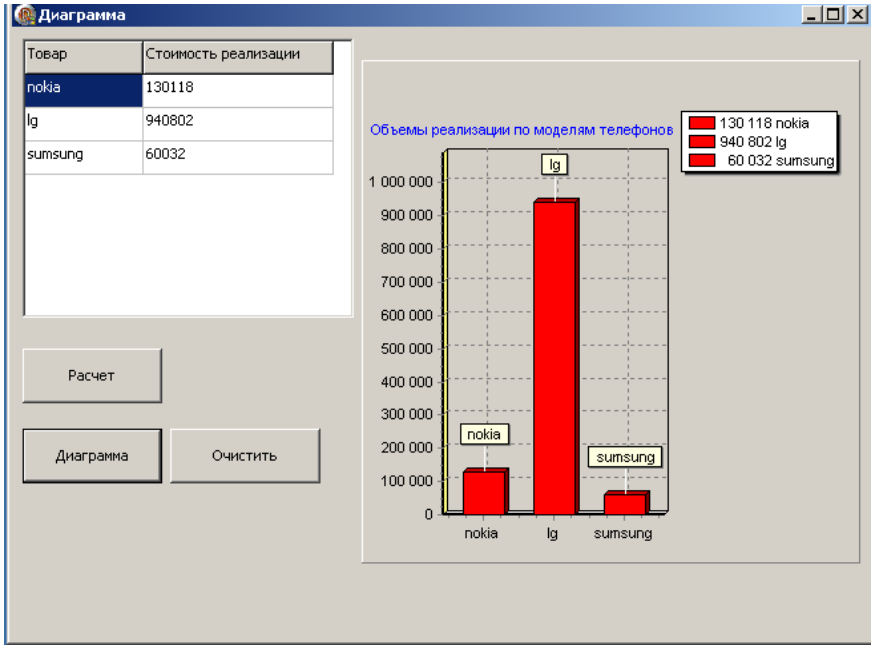


Рис. 10.18. Форма в режиме работы.

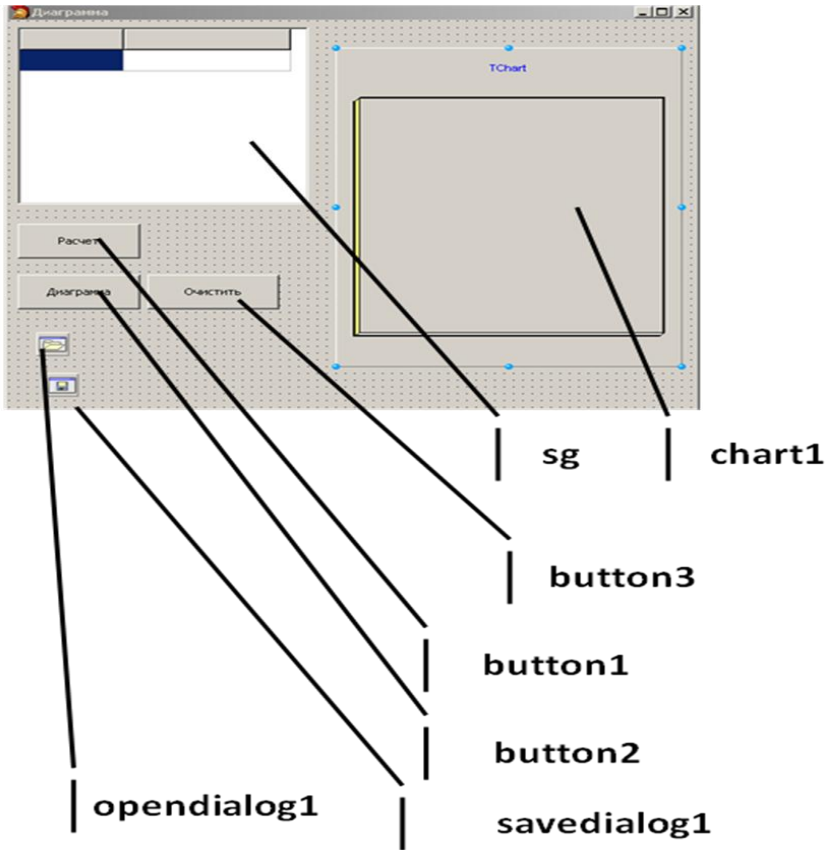


Рис. 10.19. Форма в режиме редактирования.

Структуру данных и переменные описываем в интерфейсной части.

```
unit Unit1;
Interface
type
   …
 telef=record
   modt:string[15];
   cena:integer;
   kar:string[50];
 end;
  prodajy=record
   model:string[15];
   data:string[8];
   kol:integer;
 end;

var
 Form1: TForm1;
  f:textfile;    //файл продаж
 prod:prodajy;
  f1: textfile;      //файл прайс -лист
 tel:telef;
 i,j: integer;
 s: real;
```

Процедура, связанная с активизацией формы, задает названия столбцов таблицы.

```
procedure TForm1.FormActivate(Sender: TObject);
begin
 sg.cells[0,0]:='Товар';
 sg.cells[1,0]:='Стоимость реализации';
end;
```

Процедура выполняющая расчет стоимости по моделям телефонов.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 sg.RowCount:=2;
 if  opendialog1.Execute then begin
    assignfile(f,opendialog1.filename); reset(f);
        if  opendialog1.Execute then begin
            assignfile(f1,opendialog1.filename);
           reset(f1);
           i:=1;
 while not eof(f1) do begin
      readln(f1,tel.modt);
      readln(f1,tel.cena);
      readln(f1,tel.kar);
      sg.cells[0,i]:=tel.modt;
      reset(f);
      s:=0;
      while not eof(f) do begin
           readln(f,prod.model);
          readln(f,prod.data);
          readln(f,prod.kol);
           if trim(prod.model)=trim(tel.modt) then s:=s+prod.kol*tel.cena;
      end;
      sg.cells[1,i]:=floattostr(s);
     i:=i+1;
     sg.RowCount:=sg.RowCount+1;
 end;
 sg.RowCount:=sg.RowCount-1;
 closefile(f);
 closefile(f1);
```

```
 end;
end;
end;
```

Процедура построения диаграммы, на основании вычисленных данных. В качестве диаграммы выбрана гистограмма.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  if sg.cells[0,1]='' then begin showmessage('Заполните таблицу');
                          exit; end;
  chart1.Title.Text.Add('Объемы реализации по моделям телефонов');
  for i := 1 to sg.rowcount - 1 do
        chart1.SeriesList[0].Add(strtofloat(sg.Cells[1,i]),sg.cells[0,i],clred);
  chart1.SeriesList[0].Title:=sg.cells[1,0];
end;
```

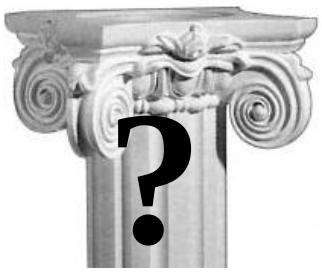
Процедура очистки диаграммы.

```
procedure TForm1.Button3Click(Sender: TObject);
begin
 chart1.Title.Text.Clear;
 chart1.SeriesList[0].Clear;
 chart1.SeriesList[0].Title:='';
end;
```

При построении круговой диаграммы можно задавать цвета каждого сектора. В этом случае процедура примет следующий вид:

```
procedure TForm1.Button2Click(Sender: TObject);
Conts
 sv: array [1..10] of Tcolor=(Claqua, Clblack, Clblue, Clfuchsia,
                  Clgray, Clgreen, Clime, Clmaroon, Clnavy, Clolive);
begin
  if sg.cells[0,1]='' then begin showmessage('Заполните таблицу');
                          exit;  end;
```

```
chart1.Title.Text.Add('Объемы реализации по моделям телефонов');
for i := 1 to sg.rowcount - 1 do
    chart1.SeriesList[0].Add(strtofloat(sg.Cells[1,i]),sg.cells[0,i],sv[i]);
chart1.SeriesList[0].Title:=sg.cells[1,0];
End;
```



Контрольные вопросы

1. Визуальные элементы: метка, поле редактирования, кнопка. Назначение и основные свойства.
2. Визуальные элементы radiogroup, memo, многостраничный блокнот. Назначение и основные свойства.
3. Визуальный элемент stringgrid. Назначение и основные свойства.
4. Невизуальный объект «главное меню». Назначение и основные свойства.
5. Подключение стандартных диалогов.
6. Построение диаграмм средствами Delphi.