Самарский государственный аэрокосмический университет
имени академика С.П. Королева

Работа в Delphi

Самара 2003

Министерство образования Российской Федерации
Самарский государственный аэрокосмический университет
имени академика С.П. Королева

Работа в Delphi

Методические указания

Самара 2003

Составитель: Стенгач М.С.

УДК 681.3.06

Работа в Delphi: /Самар. гос. аэрокосм. ун-т; Сост. Стенгач М.С. Самара, 2003, 28 с.

В методических указаниях приводятся начальные сведения о работе в популярной интерактивной среде программирования Delphi.

Предназначены для выполнения курсовых и лабораторных работ по курсу «Информатика» для студентов всех специальностей.

Составлены на кафедре "Компьютерные системы".

Печатаются по решению редакционно-издательского совета Самарского государственного аэрокосмического университета им. академика С.П. Королева.

Рецензент Л.А. Чемпинский

1. Начало работы

Интерактивная среда программирования Delphi позволяет с минимальными затратами времени создавать различные приложения для операционной системы Windows. Поскольку в основе Delphi лежит концепция быстрого создания приложений (RAD — Rapid Application Development), особое внимание в методических указаниях уделено объектно-ориентированной и визуальной технологиям программирования.

После запуска Delphi на экране появляются четыре окна, представленные на рис.1 (версия Delphi 2.0): Главное окно, Окно формы, Окно Инспектора объектов и Окно кода программы.

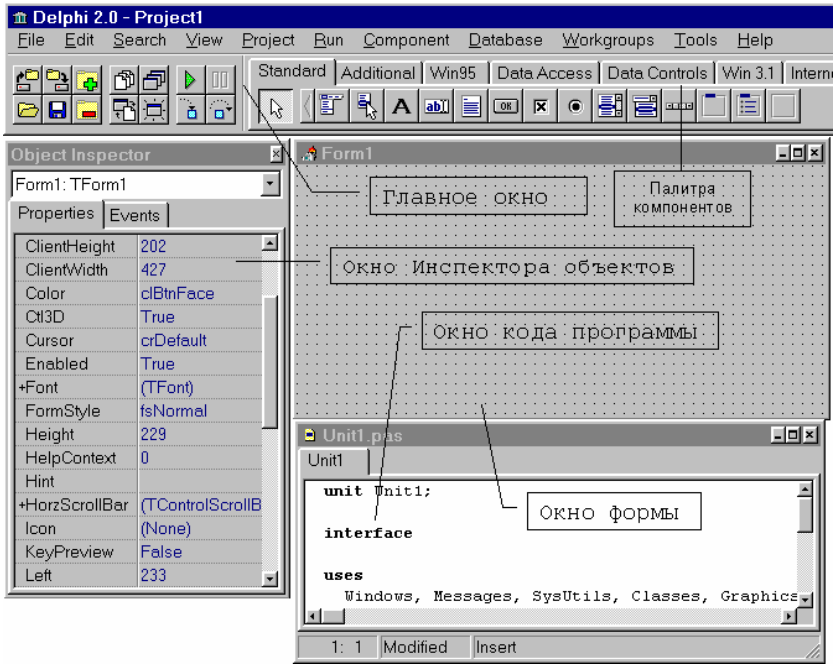


Рис. 1. Окна Delphi

1.1. Главное окно

Главное окно Delphi осуществляет основные функции управления проектом создаваемой программы. Здесь располагается главное меню Delphi, набор пиктографических командных кнопок и палитра компонентов. Все элементы главного окна располагаются на специальных панелях.

1.1.1. Пиктографические кнопки

Пиктографические кнопки главного окна открывают быстрый доступ к наиболее важным опциям главного меню. По функциональному признаку они разделены на 3 группы (приводится набор кнопок для версии Delphi 2.0):

1. Группа Standard

 - открывает созданный ранее проект программы.

 - сохраняет все файлы проекта.

 - добавляет новый файл в проект.

 - открывает существующий файл.

 - сохраняет файл на диске

 - удаляет файл из проекта

2. Группа View

 - выбирает модуль из списка модулей, связанных с текущим проектом.

 - выбирает форму из списка форм, связанных с текущим проектом.

 - переход из окна формы в окно кода программы (F12).

 - создает новую форму и добавляет ее к объекту.

3. Группа Debug

 - компилирует и выполняет программу (F9).

 - пауза в работе отлаживаемой программы.

 - пошаговая трассировка программы с отслеживанием работы вызываемых подпрограмм (F7).

 - пошаговая трассировка программы, но без отслеживания работы вызываемых подпрограмм (F8).

1.1.2. Палитра компонентов

Палитра компонентов Delphi занимает правую часть главного окна (см. рис.1) и имеет набор страниц, содержащих компоненты.

Standard. Большинство компонентов на этой странице являются аналогами экранных элементов Windows - меню, кнопки, полосы прокрутки.

Additional. Эта страница содержит более развитые компоненты. Например, компонент TSpeedButton является специальной кнопкой для инструментальных панелей. Данная страница также содержит компоненты, главное назначение которых – отображение графической информации. Компонент Image загружает и отображает растровые изображения, а компонент Shape – стандартные фигуры: окружность, квадрат и т.д.

Win95 (Win32). Эта страница содержит компоненты, позволяющие созданным с помощью Delphi программам использовать такие нововведения в пользовательском интерфейсе 32-разрядной Windows, как просмотр древовидных структур, просмотр списков, панель состояния, присутствующая в интерфейсе программы Windows Explorer (Проводник), расширенный текстовый редактор и другое.

Data Access и **Data Controls.** Delphi использует механизм баз данных компании Borland (Borland Database Engine, BDE) для организации доступа к файлам баз данных различных форматов. Компоненты этих двух страниц облегчают программам Delphi использование сервиса баз данных, предоставляемого BDE, например многопользовательского считывания, записи, индексации и выдачи запросов для таблиц dBASE и Paradox.

С использованием этих компонентов создание программы просмотра и редактирования базы данных почти не требует навыков в программировании.

Win 3.1. На этой странице находятся компоненты Delphi 1.0, возможности которых перекрываются аналогичными компонентами Windows 95.

Internet. Эта страница предоставляет компоненты для разработки приложений, позволяющих создавать HTML-файлы непосредственно из файлов баз данных и других типов, взаимодействующих с другими приложениями для Internet. Версия Delphi 4 дает возможность создавать приложения для Web-сервера в виде DLL-файлов: (Dynamic Link Library — Динамически компонуемая библиотека). С помощью компонентов страницы Internet довольно просто создавать обработчики событий для обращения к определенному URL (Uniform Resource Locator — Унифицированный локатор ресурса), представлению документов в HTML-формате и пересылки их клиент-программе.

Dialogs. Эта страница предоставляет программам Delphi простой доступ к стандартным диалоговым окнам для операций над файлами, выбором шрифтов и цвета.

System. Поскольку не каждая потребность, связанная с обработкой файлов, может быть удовлетворена с помощью стандартных диалоговых окон, страница System предоставляет возможность комбинировать отдельные элементы, такие как списки дисков, каталогов и файлов. Страница System также содержит компоненты, обрабатывающие обмен высокого уровня между программами посредством OLE (Object Linking and Embedding). Компонент Timer может генерировать события через определенные, заранее установленные промежутки времени, а MediaPlayer позволяет программам воспроизводить звук, музыку и видео.

QReport. Эта страница предоставляет компоненты баз данных. Здесь содержатся особые версии надписей, полей, примечаний и других элементов управления.

Samples. Эта страница содержит компоненты, которые не встроены в Delphi. Для этих компонентов нет встроенной интерактивной справки. Все же они не менее полезны, чем компоненты с других страниц.

OCX (ActiveX). Эта страница содержит компоненты ActiveX, разработанные независимыми производителями программного обеспечения: сетка, диаграмма, средство проверки правописания.

Midas и **Decision Cube.** Здесь собраны компоненты для доступа к удаленным серверам и осуществления SQL – запросов.

1.2. Окно формы

Форма представляет собой фундамент программы. Приложение может иметь несколько форм, каждая из которых выполняет свое особое предназначение. Изначально окно формы пустое. Точнее, оно содержит стандартные для Windows интерфейсные элементы – кнопки вызова системного меню, максимизации, минимизации и закрытия окна, полосу заголовка и очерчивающую рамку.

Для размещения компонента на форме требуется щелкнуть на нужной закладке палитры компонентов, а затем на кнопке с пиктограммой соответствующего компонента и после этого щелкнуть в окне формы.

1.3. Окно инспектора объектов

Размещаемые на форме компоненты характеризуются некоторым набором параметров. Для изменения этих параметров предназначено окно Инспектора объектов. Это окно содержит две страницы – `Properties` (Свойства) и `Events` (События). Страница `Properties` служит для установки нужных свойств компонента, страница `Events` позволяет определить реакцию компонента на то или иное событие.

Каждая страница Инспектора состоит из двух колонок. Левая колонка содержит название свойства или события, а правая – конкретное значение свойства или имя подпрограммы, обрабатывающей соответствующее событие. Если слева от названия свойства стоит значок "+", это означает, что данное свойство определяется совокупностью значений. Щелчок по значку "+" приведет к раскрытию списка составляющих данного свойства.

В верхней части окна Инспектора объектов располагается раскрывающийся список всех помещенных на форму компонентов.

1.4. Окно кода программы

Окно кода предназначено для создания и редактирования текста программы. Первоначально окно кода содержит минимальный исходный текст, обеспечивающий нормальное функционирование пустой формы в качестве полноценного Windows-окна.

Сразу после открытия нового проекта в окне кода будут такие строки:

```
unit Unit1;

interface

uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

type
  TForm1 = class(TForm)
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

end.
```

В процессе создания приложения Delphi автоматически вставляет необходимые строки между

```
unit Unit1;
```

и

```
implementation
```

Вручную программистом текст программы вставляется между строками

```
{$R *.DFM}
```

и

```
end.
```

в нижней части окна.

1.5. Размещение компонентов на форме

Размещать компоненты на форме очень просто. Требуется только щелкнуть на нужной вкладке палитры компонентов, затем на кнопке с пиктограммой соответствующего компонента и после этого щелкнуть в окне формы. Можно также щелкнуть на компоненте, а затем нарисовать прямоугольник с помощью мыши на форме – компонент появится внутри этого прямоугольника. Если размеры компонента поддаются изменению, то при появлении на форме он заполнит собой прямоугольник. Если заранее не

известно, на какой странице расположен конкретный компонент, выберите пункт Component List (Компоненты) из меню View (Вид), и на экране появится список компонентов в алфавитном порядке.

Если щелкнуть на каком-либо компоненте в палитре компонентов, то его кнопка окажется нажатой. При выборе другого компонента кнопка предыдущего компонента вернется в исходное состояние, так как одновременно не может быть выбрано несколько компонентов. Для того чтобы все кнопки оказались в исходном состоянии и было восстановлено нормальное использование мыши, следует щелкнуть на кнопке со стрелкой выбора, которая появляется с левой стороны каждой страницы палитры (см. рис. 1).

Пример размещения компонента на форме:

1. Выберите страницу Standard палитры компонентов.
2. Поместите указатель мыши на компонент с изображением буквы **A**. После того как мышь побудет в недвижимом состоянии секунду или две, появится поле помощи, идентифицирующее этот компонент как Label.
3. Щелкните на кнопке, затем щелкните на вашей форме. Появится надпись Label1.
4. Обратите внимание, что поле надписи на форме имеет небольшие квадратные маркеры (угловые и боковые), позволяющие изменять размер. Используйте их, чтобы с помощью мыши изменить размер поля надписи (но не размер самой надписи).
5. Для изменения стандартной надписи Label1 в строке Caption окна Инспектора объектов введите надпись Привет!
6. Для изменения цвета и шрифта надписи щелкните мышью по свойству Font окна Инспектора объектов и с помощью кнопки в правой части строки раскройте диалоговое окно настройки шрифта. В списке Размер этого окна выберите высоту шрифта 24 пункта, а с помощью списка Цвет выберите нужный цвет.
7. Теперь щелкните на надписи Привет! и перетащите ее в новое место.

Видно, что при перемещении и изменении размера компоненты выравниваются по точкам координатной сетки формы. Такая возможность помогает поддерживать порядок в формах. Для отмены координатной сетки или изменения плотности точек, выберите пункт Options из меню Tools. Первая страница параметров предназначена для настройки пользователем параметров среды. На этой странице имеется группа Form designer (Конструктор форм), флажки опций Display grid (Отображение сетки) и Snap to grid (Привязка к сетке) которой определяют, видна ли координатная сетка и активна ли она. Можно также изменить значения параметров Grid size X (Шаг по оси X) и Grid size Y (Шаг по оси Y), что приведет к изменению шага координатной сетки по горизонтали и вертикали, соответственно.

Практика показывает, что для лучшего управления размещением и размерами компонентов значение шага координатной сетки должно быть равным 4 вместо 8, отключено изображение координатной сетки и включен параметр Snap to grid.

Не каждый компонент виден на форме во время запуска программы. Например, размещение на форме компонента MainMenu приводит к появлению в разрабатываемом приложении меню, но соответствующая пиктограмма во время запуска программы не отображается. Компоненты, представляющие диалоговые окна общего назначения, вообще никак не визуализируются во время работы программы. Размеры невидимого компонента в процессе разработки не изменяются - он всегда отображается в виде пиктограммы.

2. Объектно-ориентированное программирование (язык Object Pascal)

В длинных и сложных программах переменные и функции могут исчисляться сотнями. Наличие переменных с одинаковыми именами в различных подпрограммах может привести к нежелательным конфликтам. Объектно-ориентированное программирование было создано для того, чтобы большие программы можно было разделять на отдельные части.

Двумя фундаментальными концепциями всех объектно-ориентированных языков (Object Pascal, C++) являются *объекты* и *классы.*

Объект – фрагмент программы, предназначенный для решения отдельной задачи. Объекты обладают бóльшими возможностями, чем простые подпрограммы, поскольку в них могут содержаться как процедуры и функции, так и переменные, что облегчает работу с ними.

Объектно-ориентированный язык Object Pascal, который является основным инструментом Delphi, разработан на основе популярного языка программирования Паскаль. По этой причине все основные элементы этих двух языков практически совпадают. Это относится к операторам языка, типам данных, выражениям, операциям, процедурам и функциям. Поскольку языку Паскаль посвящено большое количество изданных ранее на кафедре пособий и методических указаний, мы в этой главе подробно разберем те существенные дополнения, которые были внесены разработчиками в Object Pascal.

2.1. Классы

Классы предназначены для создания объектов.

Структура описания класса:

```
type
```

```
<имя класса> = class
<описание переменных и подпрограмм>
end;
```

Взаимосвязь класса с объектом подобна взаимосвязи типа данных с переменной. Например, в следующем примере создается целая переменная с именем `data`.

```
data: integer;
```

Аналогичная связь существует между классом и объектом. Приближенно класс можно рассматривать как тип объекта. Например, если заранее определить класс `Tmycl`, можно создать объект этого класса с именем `myobj`

```
type
  Tmycl = class
    a: integer;
    …
  end;
var
 myobj: Tmycl;
```

Следует помнить, что все рабочие данные хранятся в объекте. Класс не содержит никаких данных – он лишь описывает общую структуру объекта.

По используемому в Delphi соглашению все имена классов начинаются с буквы T.

2.1.1. Класс TForm и реакция на события

Проведем модернизацию нашей первой программы (см. пункт 1.5): поместим на ее форму еще один компонент – кнопку – и заставим ее откликаться на событие, связанное с нажатием левой клавиши мыши. Компонент "кнопка" изображается пиктограммой на странице `Standard` палитры компонентов как кнопка с надписью `OK`. Щелкните по этой кнопке, а затем щелкните на форме. Появится кнопка с надписью `Button1`.

При щелчке по кнопке мышью в работающей программе возникает событие `OnClick` (по щелчку). Пока это событие никак не обрабатывается программой, и поэтому "нажатие" кнопки не приведет ни к каким последствиям. Чтобы заставить программу реагировать на нажатие кнопки, необходимо написать на языке Object Pascal фрагмент программы, который называется *обработчиком события*. Этот фрагмент программы оформляется в виде процедуры.

Чтобы заставить Delphi самостоятельно сделать заготовку для процедуры обработчика события `OnClick`, дважды щелкните мышью по вновь вставленному компоненту (кнопке). В ответ Delphi активизирует Окно кода, и вы увидите в нем такой текстовый фрагмент:

```
procedure TForm1.Button1Click(Sender: TObject);
begin

end;
```

Попробуем разобраться, что он содержит. Слово **`procedure`** извещает компилятор о начале подпрограммы-процедуры. За ним следует имя процедуры `TForm1.Button1Click`. Это имя – составное: оно состоит из имени класса `TForm1` и собственно имени процедуры `Button1Click`.

В состав Delphi входит несколько сотен классов, созданных программистами корпорации Borland (так называемых стандартных классов). Каждый компонент принадлежит к строго определенному классу, а все конкретные экземпляры компонентов, вставляемые в форму, получают имя класса с добавленным числовым индексом. Таким образом, имя `TForm1` означает имя класса, созданного по образцу стандартного класса `TForm`. Если посмотреть начало текста в окне кода, то мы увидим следующие строки

```
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
```

Строка

```
TForm1 = class(TForm)
```

определяет новый класс TForm1, который *порожден от* (создан по образцу) стандартного класса TForm. Строка

```
Form1: TForm1;
```

создает *объект* этого класса с именем Form1. Стандартный класс TForm описывает пустое Windows-окно, в то время как TForm1 описывает окно с уже вставленными в него компонентами *метка* и *кнопка*. Описание этих компонентов содержат строки

```
Label1: TLabel;
Button1: TButton;
```

Они указывают, что кнопка Button1 представляет собой *экземпляр* компонента TButton, а метка Label1 - экземпляр компонента TLabel.

Кроме того, класс TForm1 содержит процедуру Button1Click на что указывает строка

```
procedure Button1Click(Sender: TObject);
```

За именем процедуры Button1Click в круглых скобках следует описание вызова Sender: TObject (параметр с именем Sender принадлежит классу TObject). Параметр Sender вставлен Delphi "на всякий случай": с его помощью подпрограмма Button1Click может, при желании, определить, какой именно компонент создал событие OnClick.

Само тело процедуры TForm1.Button1Click(Sender: TObject); (т.е. строки между словами **begin…end;)** пока еще не содержит описания каких-либо действий, т.к. Delphi лишь создала заготовку для процедуры. Наполнить тело нужными операторами – задача программиста.

Каждый раз при нажатии кнопки Button1 управление будет передаваться в тело процедуры, а значит, между словами **begin** и **end** мы можем написать фрагмент программы, который будет выполняться в ответ на это событие.

Чтобы убедиться в этом сделаем нашу кнопку "звучащей": напишите в пустой строке между словами **begin** и **end** следующее предложение:

```
messagebeep(mb_ok);
```

и сделайте прогон программы (клавиша F9). Теперь в ответ на нажатие кнопки в динамике компьютера будет раздаваться звуковой сигнал.

2.1.2. Основные понятия класса

В основе классов лежат три фундаментальных принципа, которые называются *инкапсуляция, наследование* и *полиморфизм*.

Инкапсуляция. Класс представляет собой единство трех сущностей – полей (переменных), методов (подпрограмм) и свойств. Объединение этих сущностей в единое целое и называется инкапсуляция. Инкапсуляция позволяет сделать класс самодостаточным для решения конкретной задачи. Например, класс TForm содержит в себе (инкапсулирует в себе) все необходимое для создания Windows-окна. Класс TMemo представляет собой полноценный текстовый редактор, класс TTimer обеспечивает работу программы с таймером.

Наследование. Любой класс может быть порожден от другого класса. Для этого при его объявлении указывается имя класса-родителя:

```
TChildClass = class(TParentClass)
```

Порожденный класс автоматически наследует поля, методы и свойства своего родителя и может дополнять их новыми.

Все классы Object Pascal порождены от единственного родителя – класса TObject. Этот класс не имеет полей и свойств, но включает в себя методы самого общего назначения – от создания до уничтожения объектов.

Все новые классы будут дочерними классу TObject. Следующие два объявления идентичны:

```
Tc = class(TObject)
Tc = class
```

Принцип наследования привел к созданию в Delphi дерева классов, где каждый потомок дополняет возможности своего родителя новыми.

Для примера на рис. 2 показан фрагмент дерева классов Delphi. Класс TPersistent обогащает возможности своего родителя – он умеет сохранять данные в файле и получать их из него, в результате это умеют делать все его потомки. Класс TComponent, в свою очередь, умеет взаимодействовать со средой разработчика, TControl дополнительно умеет создавать и обслуживать видимые на экране компоненты, а его потомок TWinControl может создавать Windows-окна и т.д.

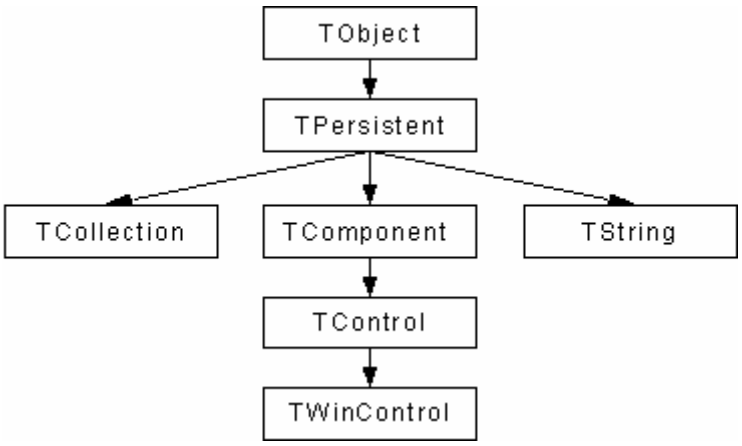


Рис. 2. Фрагмент дерева классов Object Pascal

Полиморфизм – это свойство классов решать сходные по смыслу задачи разными способами. Можно изменить (*перекрыть*) метод родителя в потомке. Для этого необходимо объявить в потомке одноименный метод и реализовать в нем нужные свойства. В результате в родителе и потомке будут действовать два одноименных метода, но с разными свойствами.

2.1.3. Составляющие класса

Поля. Полями называются инкапсулированные в классе данные. Поля могут быть любого типа, в том числе и классами, например:

```
type Tmycl = class
     a:   integer;
     obj: TObject;
     end;
```

Методы. Инкапсулированные в классе процедуры и функции называются методами.

```
type Tmycl = class
     function myfun(a:integer):integer;
     procedure myproc;
  end;
```

Доступ к методам класса, как и к его полям, возможен с помощью составных имен:

```
var myobj: Tmycl;
begin
myobj.myproc;
end;
```

Свойства. Свойства – это специальный механизм классов, регулирующий доступ к полям. Свойства объявляются с помощью зарезервированных слов `property`, `read` и `write`.

Например:

```
type Tmycl = class
     a: integer;
     function getfield:integer;
     procedure setfield(value:integer);
     property i: integer read  getfield
                         write setfield;
end;
```

В программе свойство ведет себя как обычное поле. Например:

```
var myobj: Tmycl;
begin
  myobj.i:=2;
end;
```

Разница между оператором `myobj.a:=2;` и оператором `myobj.i:=2;` заключается в том, что при обращении к свойству автоматически подключается метод `setfield`, в котором могут реализовываться специфические действия.

2.1.4. Объявления класса

Класс может содержать разделы: `public`, `private`, `published`, и `protected`. Внутри разделов сначала определяются поля, а затем – методы и свойства.

Разделы определяют области видимости элементов. Элементы раздела `public` можно вызвать в любом другом модуле программы. Элементы раздела `private` доступны только внутри методов данного

класса и в подпрограммах, находящихся в том же модуле, где описан класс. В разделе published перечисляются свойства, доступные на этапе конструирования программы (т.е. в окне Инспектора объектов). Раздел protected доступен только методам самого класса, а также методам потомков этого класса.

2.2. Структура программы в Delphi

Программа Delphi состоит из файла проекта (файл с расширением dpr) и одного или нескольких модулей (файлы с расширением pas). Файл проекта создается автоматически. Чтобы его увидеть необходимо в главном меню выбрать View/Project Source (или Project/View Source в более поздних версиях). В окне кода программы с закладкой Project1 появится следующий текст:

```
program Project1;

uses Forms,
     Unit1 in 'Unit1.pas';
{$R *.RES}
begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

Модуль Forms является стандартным, а модуль Unit1, находящийся в файле Unit1.pas, - это новый модуль, который создается автоматически (текст такого модуля приведен в разделе 1.4).

В теле проекта происходит обращение к методам объекта Application. В этом объекте собраны данные и подпрограммы, необходимые для функционирования Windows-программы. Метод Create Form создает и показывает на экране окно главной формы, а метод Run реализует бесконечный цикл получения и обработки поступающих от Windows сообщений о действиях пользователя.

2.3. Новые типы данных в Delphi

Comp и **Currency** – вещественные числа с дробной частью фиксированной длины. У Comp дробная часть отсутствует, а у Currency длина дробной части 4 десятичных разряда. Область применения этих типов – бухгалтерские расчеты.

TDateTime – тип дата-время, предназначен для одновременного хранения даты и времени. Представляет собой вещественное число с фиксированной дробной частью. В целой части хранится дата, в дробной – время. Дата определяется как количество суток, прошедших с 30.12.1899 г., а время – как часть суток, прошедших с 0 часов. Так, значение 37327,698 соответствует дате 12.03.2002 г. и времени 16:45:07. Над данными типа TDateTime определены те же операции, что и над вещественными числами.

Для работы с типом дата-время используются следующие функции:

Date: TDateTime; - возвращает текущую дату;

Now: TDateTime; - возвращает текущую дату и время;

DateToStr(D: TDateTime): String; – преобразует дату в строку символов в формате dd.mm.yy;

FormatDateTime(F: String; D: TdateTime): String; – преобразует дату и время в строку символов в формате, указанном в F.

Пример:

```
var D: TDateTime;
    S1,S2,S3: String;
begin
D:=Now;
S1:=DateToStr(D);
S2:=FormatDateTime('dd.mm.yy hh:mm:ss',D);
S3:=FormatDateTime('dd mmmm yyyy',D);
end;
```

В результате работы программы переменные S1,S2,S3 примут следующие значения:

S1= '12.03.02'

S2= '12.03.02 16:45:07'

S3= '12 Март 2002'

3. Компоненты Delphi

В этой главе мы рассмотрим наиболее популярные компоненты Delphi.

3.1. Компоненты страницы Standard

TMainMenu – главное меню. После размещения компонента TMainMenu на форме необходимо задать его пункты. Для этого следует дважды щелкнуть по компоненту левой кнопкой мыши. На экране

покажется окно конструктора меню. Создание пунктов не вызывает проблем. В Окне инспектора объектов в строке `Caption` вводится текст пункта. После нажатия клавиши Enter можно переходить к следующему пункту. Для создания у пункта раскрывающегося меню необходимо щелкнуть мышью по строке ниже пункта и ввести первый пункт подменю. Если в свойстве `Caption` задать символ "-", то в списке подменю появится разделительная черта.

TPopupMenu – локальное меню, появляющееся после нажатия правой кнопки мыши. Может быть создано для любого компонента, для этого в свойстве `PopupMenu` этого компонента необходимо поместить имя компонента-меню.

TLabel – метка для отображения текста. В разделе 1.6. мы уже подробно рассмотрели размещение этого компонента на форме.

TEdit – ввод и отображение строки. Компонент класса TEdit представляет собой однострочный редактор. Главным свойством компонента является свойство `Text`, отображающее строку. Так как строка имеет тип `String`, для ввода чисел необходимо воспользоваться следующими стандартными функциями:

`StrToFloat(s: String): real;` - преобразует строку `s` в вещественное число,

`StrToInt(s: String): integer;` - преобразует строку `s` в целое число.

Например:

```
var x: real;
x:= StrToFloat(Edit1.Text);
```

Чтобы отобразить в компоненте число, необходимо проделать обратную операцию:

```
Edit1.Text:= FloatToStr(x);
```

TMemo – ввод и отображение текста. Компоненты данного класса предназначены для ввода и редактирования длинного текста. Текст хранится в свойстве `Lines` и представляет собой пронумерованный набор строк. При работе с компонентом TMemo наиболее часто употребляются следующие операции:

`Memo1.Lines[i:integer];` – доступ к i-ой строке;

`Memo1.Lines.Count` – общее количество строк;

`Memo1.Lines.add(s: string);` – добавить новую строку `s` в список;

`Memo1.Clear;` - очистить содержимое компонента.

Компонент имеет также свойство `Text`, которое, в отличие от `Lines`, содержит текст в виде длинной строки.

TButton – кнопка. В разделе 2.1.1. уже был рассмотрен пример размещения кнопки на форме, а также создания отклика на событие, связанное с нажатием левой клавиши мыши. Чтобы событие `OnClick` возникало также и при нажатии клавиши Enter, необходимо свойство `Default` задать `True`.

В Windows часто используются *модальные окна*. Модальное окно – это окно, блокирующее работу с другими окнами вплоть до своего закрытия. Обычно в состав модального окна включается несколько кнопок. Если у кнопки свойство `ModalResult` не равно `mrNone`, нажатие на нее приведет к закрытию модального окна.

В отличие от большинства других компонентов кнопка является компонентом самой Windows и поэтому не может изменять свой цвет произвольным образом – она его меняет вместе с изменением палитры самой Windows.

 **TCheckBox** – независимый переключатель или флаг выбора. В составе диалогового окна их может быть несколько и состояние любого из них не зависит от состояния остальных. Логическое свойство `Checked` определяет наличие флага выбора:

`if CheckBox1.Checked then …` – если флаг установлен,

`if not CheckBox1.Checked then …`– если флаг не установлен.

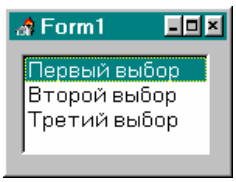
Можно использовать свойство `State`:

```
case CheckBox1.State of
cbChecked:   … - да,
cbUnChecked: … - нет,
cbGrayed:    … - не совсем – в окошке компонента устанавливается флаг выбора, но само окошко
```

окрашивается серым цветом.

 **TRadioButton** – зависимые переключатели. На форме имеет смысл размещать минимум два компонента `TRadioButton`. Если в одном из компонентов свойство `Checked` принимает значение `True`, то у других компонентов оно принимает значение `False`.

TListBox – список выбора.



В списке предусмотрена возможность программной прорисовки элементов, поэтому список может содержать не только строки, но и произвольные изображения. Для ввода пунктов меню необходимо выбрать свойство `Items`. После того, как меню сформировано, двойным щелчком по компоненте

создается реакция на событие:

```
procedure ListBox1Click(Sender: TObject);
```

Количество строк в списке определяется с помощью свойства `Count: integer;`, например:

```
M:=ListBox1.Items.Count;
```

В данном примере `M` будет равно 3.

Номер выбранного пункта определяется свойством `ItemIndex`:

```
I:=ListBox1.ItemIndex;
```

Номер первой строки равен 0, второй 1 и т.д. В случае выбора второго пункта списка возможна такая конструкция:

```
if ListBox1.ItemIndex=1 then … ;
```

TComboBox – раскрывающийся список выбора. Пункты меню также задаются в свойстве `Items`.

В свойстве `Style` можно задать следующие модификации компонента:

`CsDropDown` (по умолчанию) – раскрывается после нажатия кнопки;

`CsSimple` – список всегда раскрыт.

TScrollBar – управление значением величины. 

Свойства компонента:

`Max: integer;` - максимальное значение диапазона изменения числовой величины;

`Min: integer;` - минимальное значение диапазона изменения числовой величины;

`Position: integer;` - текущее значение числовой величины.

TGroupBox и **TPanel** – панели, предназначены для размещения в них групп элементов.

TRadioGroup – контейнер зависимых переключателей. Чтобы создать в контейнере переключатели, следует использовать свойство `Items`. Свойство `ItemIndex` по умолчанию имеет значение `–1` (не выбран ни один переключатель). Если значение `ItemIndex` равно `0`, то выбран первый зависимый переключатель, если равно `1`, то второй и т. д. В случае выбора второго зависимого переключателя возможна такая конструкция:

```
if RadioGroup1.ItemIndex=1 then…
```

3.2. Компоненты страницы Additional

TBitBtn – кнопка с изображением – разновидность кнопки `TButton`. Её отличительная особенность – свойство `Glyph`, с помощью которого определяется изображение, рисуемое на поверхности кнопки. Свойство `Kind` определяет одну из 11 стандартных разновидностей кнопки. Например:

- свойство `Kind` равно `bkOk;`

- свойство `Kind` равно `bkClose.`

Нажатие любой из этих 11 кнопок (кроме `bkCustom` и `bkHelp`) закрывает модальное окно и возвращает в программу результат `mrИмякнопки`, например: `bkOk-mrOk, bkCancel-mrCancel.` У кнопок `bkOk` или `bkYes` свойство `Default` автоматически устанавливается `True`. Кнопка `bkClose` закрывает главное окно программы. Кнопка `bkHelp` автоматически вызывает раздел справочной службы, связанный с `HelpContext` формы, на которую она помещена.

 **TSpeedButton** – кнопка для инструментальных панелей.

Особенности этих кнопок:

1. могут фиксироваться в утопленном состоянии;
2. не могут закрывать модальное окно;
3. не могут быть умалчиваемыми (отсутствует свойство `Default`).

Для фиксации кнопки свойство `GroupIndex` должно быть не равным 0. Если у нескольких кнопок свойство `GroupIndex` имеет одно и то же значение, то эти кнопки принадлежат одной группе.

Логическое свойство `AllowAllUp` определяет поведение кнопки:

– если `AllowAllUp=False` - утопленная кнопка отпускается только при нажатии другой кнопки, входящей в ту же группу,

– если `AllowAllUp=True` - кнопку можно освободить повторным щелчком.

TMaskEdit – специальный редактор ввода текста по шаблону, задаваемому свойством `EditMask: String`.

TStringGrid – таблица строк, предназначена для создания таблиц. Таблица делится на две части – фиксированную (служит для показа заголовков столбцов и рядов) и рабочую.

Главное свойство компонента: `Cells` – двухмерный массив ячеек, где первый номер соответствует столбцу, а второй – ряду. Например:

```
Cells[1,1]:='Левая верхняя ячейка';
```

Свойства компонента:

`ColCount` – общее количество столбцов в таблице;

`RowCount` – общее количество строк в таблице;

`FixedCols` –количество фиксированных столбцов;

`FixedRows` –количество фиксированных строк;

`Options.goEditing` – если задано `True`, то можно редактировать текст в таблице;

TDrawGrid – произвольная таблица, является родителем таблицы строк `TStringGrid`.

TImage – отображение картинок. Служит для размещения на форме изображений. Свойство `Picture` позволяет отыскать файл с нужным изображением и поместить это изображение на форме.

TShape – прорисовка стандартных фигур. В свойстве `Shape` имеется следующий набор стандартных фигур:

`StSquare` – прямоугольник;

`StCircle` – круг;

`StEllipse` – эллипс;

`StRoundRect` – скругленный прямоугольник;

`StRoundSquare` – скругленный квадрат.

Свойства `Brush` и `Pen` позволяют менять цветовое оформление компонента.

TBevel – декоративная рамка. Свойство `Style` определяет стиль рамки – вдавленная или выпуклая, а свойство `Shape` определяет её внешний вид.

TScrollBox – панель с прокруткой. Служит контейнером для размещения других компонентов. Использование компонента не отличается сложностью: положите его на форму и разместите затем на нем другие компоненты. Если очередной компонент выйдет за пределы рабочей области, по сторонам контейнера возникнут полосы прокрутки.

3.3. Компоненты страницы Win95 (Win32)

TTabControl – набор закладок. Представляет собой контейнер с закладками.

Основные свойства компонента:

`Tabs` – определяет название и количество закладок;

`TabIndex` – определяет индекс выбранной закладки. Индексация начинается с 0.

При выборе новой закладки возникает событие `onChange`.

TPageControl – набор страниц с закладками (рис.3). Изначально содержит одну страницу. Чтобы добавить новую страницу необходимо щелкнуть по компоненте правой кнопкой мыши и выбрать `New Page`. Свойство `ActivePage` содержит ссылку на активную панель. Пример:

```
if PageControl1.ActivePage=TabSheet3 then …
```

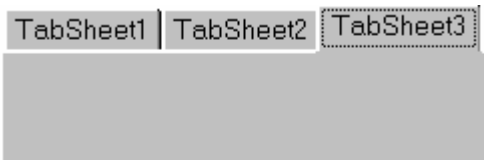


Рис. 3. Демонстрация компонента `TPageControl`

Каждая страница компонента `TPageControl` будет представлять отдельную компоненту со своими свойствами.

TImageList – контейнер для хранения множества рисунков одинакового размера. Двойным щелчком мыши по компоненту раскрывается редактор `ImageList Editor`. С помощью кнопки `Add` можно поместить в контейнер новое изображение. Пример использования контейнера см. ниже у компонента `TListView`.

TTreeView – дерево иерархии. Служит для показа ветвящихся иерархических структур, таких как, например, файловая структура диска. Для наполнения списка необходимо щелкнуть по кнопке в свойстве `Items` – на экране появится окно редактора компонента. Щелкните по кнопке `New Item` и введите связанный с узлом текст в поле `Text`. Для ввода подузла сначала нужно щелкнуть в окошке `Items` по узлу, который должен стать родительским, а затем – на кнопке `New SubItem`.

TListView – список просмотра. Предназначен для показа и выбора нескольких элементов. Допускается не более одного уровня вложенности.

Пример создания списка просмотра:

1. На пустой форме расположите компоненты `TListView` и `TImageList`.
2. Для наполнения контейнера `ImageList1` дважды щелкните по нему мышью.

3. В появившемся окне нажмите кнопку Add и выберите файл Program Files\Borland\Delhi2\Images\Icons\Chemical. Выберите в контейнере Display Options вариант Stretch.
4. Повторите п.3 для выбора нескольких пиктограмм.
5. Поместите в свойствах SmallImages и LargeImages компонента ListView1 значение ImageList1
6. Щелкните по кнопке в свойстве Items – на экране появится окно редактора компонента.
7. Щелкните по кнопке New Item и введите в полях Caption и ImageIndex значение 0 (первая пиктограмма из ImageList1).
8. Повторите п.7 с заданием у последующих элементов списка значений Caption и ImageIndex равными 1, 2, 3 и т.д.
9. Выберите значение свойства ViewStyle равным vsList.

Результат работы программы представлен на рис. 4.

Рис. 4. Демонстрация компонента TListView

THeaderControl – заголовок с регулируемыми размерами ширины колонок. Колонки формируются в свойстве Sections.

– пример заголовка из трех колонок.

При перемещении границ колонок возникает событие onResize, возвращающее следующие значения:

Sections.Items[i].Left – левая граница i-ой колонки,

Sections.Items[i].Width – ширина i-ой колонки,

HeaderControl1.Height – высота заголовка.

TRichEdit – редактор RTF-текста. Текст формата RTF хранит информацию, управляющую свойствами каждого абзаца и сменой шрифта по ходу текста. Для хранения атрибутов шрифта используются объекты класса TTextAttributes. Эти атрибуты распространяются на весь текст через свойство редактора DefAttributes или на выделенную часть текста – через свойство SelAttributes. Атрибуты абзаца хранятся в объектах класса TParaAttributes.

Например, необходимо, чтобы текст в редакторе вводился наклонный. Данную задачу реализует следующая процедура:

```
procedure TForm1.RichEdit1Change(Sender: TObject);
var CurrText: TTextAttributes;
begin
    CurrText:=RichEdit1.SelAttributes;
    CurrText.Style:= CurrText.Style + [fsItalic]
end;
```

TStatusBar – панель состояния. Компонент может иметь несколько секций, а также кнопку изменения размеров окна, в которое он помещен. Пример компоненты TStatusBar из двух панелей и кнопки изменения размеров окна представлен на рис.5.

Рис. 5. Демонстрация компонента TStatusBar

Панели задаются в свойстве Panels. Каждая панель относится к классу TStatusPanel.

TProgressBar – компонент предназначен для отображения хода выполнения длительного по времени процесса (рис.6).

Рис. 6. Демонстрация компонента TProgressBar

Главные свойства компонента:

Position: integer; - текущее значение числовой величины;

Max: integer; - максимальное значение диапазона изменения свойства Position;

Min: integer; - минимальное значение диапазона изменения свойства Position.

Следующий оператор реализует работу компонента TProgressBar:

```
for i:=1 to 100 do ProgressBar1.Position:=i;
```

TUpDown – спаренная кнопка  . Компонент предназначен для регулирования числовой величины. Имеет, как и компонент `TProgressBar`, свойства `Position, Max и Min`, а также свойство `Increment: SmallInt;` - шаг регулируемой величины.

THotKey – компонент служит для ввода и отображения горячих клавиш. При вводе распознает нажатие клавиш Shift, Ctrl и Alt и преобразует их в текст `"Shift+"`, `"Ctrl+"` и `"Alt+"`.

3.4. Компоненты страницы System

TTimer – таймер. Служит для отсчета интервалов реального времени. Его свойство `Interval` определяет интервал времени в миллисекундах, который должен пройти от включения таймера до наступления события `onTimer`. Таймер включается при установке значения `True` в свойстве `Enabled`.

TPaintBox – окно для рисования произвольных рисунков. Графические объекты находятся в свойстве `Canvas`, графические инструменты – в свойствах `Brush, Pen` и `Font.` Например:

```
with PaintBox1, Canvas do
Pen.Width:=10;//толщина линии 10 пиксел
Brush.Color:=clRed; // цвет заливки красный
Font.StyleColor:=[fsItalic]; // шрифт наклонный
Rectangle(0,0,20,20); // рисуется прямоугольник
end;
```

TFileListBox – панель с именами файлов.

Свойство `Directory: String;` определяет каталог размещения файлов, а свойство `Drive: Char;` определяет диск размещения файлов.

TDirectoryListBox – панель с именами каталогов. Совместно с компонентами `TFileListBox, TDriveComboBox` и `TFilterComboBox` может использоваться для создания диалоговых окон доступа к файлам.

TDriveComboBox – список выбора с именами доступных дисков.

TFilterComboBox – список выбора с расширениями файлов.

TMediaPlayer – медиаплейер, представляет собой набор кнопок, предназначенных для управления различными мультимедийными устройствами.

Рис. 7. Кнопки компонента `TMediaPlayer`

Если компьютер оснащен звуковой картой, разместите этот компонент на пустой форме, в свойстве `FileName` поместите название любого файла с расширением WAV или MID, установите в свойстве `AutoOpen` компонента значение `True` и запустите программу – после щелчка мышью по кнопке  прозвучит выбранный музыкальный фрагмент.

TOleContainer – контейнер для размещения OLE-объектов. Такие объекты (таблицы, картинки и пр.) на форме Delphi-программы выглядят как обычно или заменяются пиктограммами. Активизация OLE-объекта с помощью двойного щелчка мышью приводит к активизации связанной с объектом программы, которая называется OLE-сервером и которая после загрузки показывает на экране свое окно и предоставляет пользователю средства редактирования объекта.

Типичными OLE-серверами являются Paint, Word, Excel и др.

Пример создание связанного OLE-объекта:

1. Расположите на пустой форме компонет `TOleContainer`. Установите в его свойстве `Align` значение `AllClient`.

2. Сохраните проект в каком-нибудь каталоге и скопируйте в этот же каталог файл Athena.bmp из каталога `Program Files\Borland\ Delhi2\Images\Splash\16color`.

3. Напишите такой обработчик события `OnCreate` для формы `Form1`:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
OleContainer1.CreateObjectFromFile('athena.bmp',False);
end;
```

Запустите программу и дважды щелкните по изображению мышью. Появится окно графического редактора Paint, возможности которого можно использовать для редактирования изображения. Для завершения редактирования нажмите клавишу Esc.

3.5. Компоненты страницы Samples

TGauge – отображение некоторой изменяющейся числовой величины. Данный компонент отличается от `TProgressBar` разнообразием форм (рис.8).

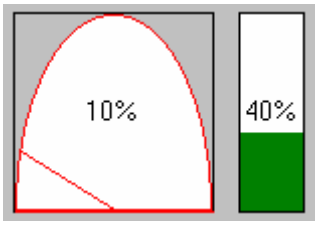


Рис. 8. Примеры вариантов компонента `TGauge`

Главные свойства компонента:

`Kind: TGuageKind;` - определяет форму индикатора;

`BackColor: Tcolor;` - цвет не закрашенной части индикатора;

`ForeColor: Tcolor;` - цвет закрашенной части индикатора;

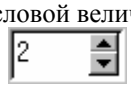
`Progress: integer;` - текущее значение числовой величины.

Следующий оператор реализует работу компонента `TGauge`:

```
for i:=1 to 100 do Gauge1.Progress:=i;
```

TColorGrid – выбор и отображение цвета из 16-цветной палитры. Выбирать можно два цвета: основной и фоновый.

TSpinButton – спаренная кнопка, не связанная с числовой величиной.

TSpinEdit – редактор для ввода целого числа 

Главное свойство компонента `Value` – текущее значение числовой величины. С помощью свойства `Value` также устанавливается и начальное значение числовой величины.

TDirectoryOutLine – отображение древовидной структуры каталогов. В отличие от `TDirectoryListBox` компонент отображает полную структуру каталогов, а не маршрут доступа к одному из них.

TCalendar – отображает календарь на выбранный месяц и год. Свойства `Day`, `Month` и `Year` могут содержать любую дату от 1 до 9999 года.

4. Форма

Форма является основным строительным блоком Delphi. Любая программа имеет как минимум одну связанную с ней форму, которая называется главной, - эта форма появляется в момент старта программы.

4.1. Разновидности форм

Разновидности формы определяются значениями свойств `TFormStyle`:

```
property FormStyle: TFormStyle;
TFormStyle=(fsNormal,fsMDIChild,fsMDIForm,fsStayOnTop);
```

`fsNormal` – обычная форма, в том числе форма общего управления программой (главная форма).

`FsMDIChild` и `fsMDIForm` – формы приложений в стиле MDI (Multi Document Interface). Этот стиль предполагает создание главного (рамочного) окна, внутри которого появляются дочерние окна. Дочерние окна не могут выходить за границы рамочного окна и не имеют собственного главного меню. Для создания рамочного окна используется стиль `fsMDIForm`, а для создания дочернего MDI-окна – стиль `fsMDIChild`.

`FsStayOnTop` – окно-поплавок (popup), которое располагается над всеми другими окнами программы.

Современные многооконные приложения чаще всего строятся в стиле SDI (Single Document Interface), который не накладывает ограничения на размер и положение вспомогательных форм. Для создания формы в этом случае используется стиль `fsNormal`.

Delphi имеет множество стандартных форм-заготовок, предназначенных для решения конкретных задач (опция `File/New/Other`). Помимо универсальной пустой формы Form (страница `New`) имеются, например, следующие специализированные формы:

Название	Страница	Назначение
About Box	Forms	Окно "О программе"
Password Dialog	Dialogs	Диалоговое окно с редактором `TEdit`, кнопками `OK` и `Cancel` для ввода пароля
Quick Report Wizard	Business	Мастер создания отчетов для баз

		данных

4.2. Создание приложений с несколькими формами

Выбранная форма автоматически подключается к проекту. Самая первая подключенная к проекту форма (`Form1`) становится главным окном. Можно назначить главным окном любую форму, раскрыв в опции

`Project/Options…` список `Main form` и выбрав нужную форму.

Каждое следующее окно становится видно только после обращения к его методу `Show` или `ShowModal`. Для этого нужно сослаться на модуль этого окна в разделе `implementation` модуля главного окна.

Например, главное окно имеет имя `Form1` и имя связанного модуля `Unit1`. Следующее созданное окно имеет имя `Form2` и имя связанного модуля `Unit2`. В разделе `implementation` модуля `Unit1` помещаем:

```
uses Unit2;
```

после чего вызываем окно на экран:

```
Form2.Show; или Form2.ShowModal;
```

Delphi автоматизирует вставку ссылки на модуль. Для этого на этапе конструирования нужно активизировать главное окно, щелкнув по нему мышью, после чего обратиться к опции `File/Uses Unit`. В появившемся диалоговом окне нужно выбрать модуль и нажать `OK`.

При вызове метода `Show` второе окно появляется на экране и работает одновременно с первым, поэтому управление сразу передается оператору, стоящему за обращением к этому методу.

Обращение к методу `ShowModal` создает модальное окно, которое полностью берет на себя дальнейшее управление программой, поэтому оператор за обращением к `ShowModal` в вызывающей части программы получит управление только после закрытия модального окна. В момент закрытия диалога модальное окно должно поместить число, соответствующее решению пользователя, в свое свойство `ModalResult`. Некоторые специальные кнопки (`OK`, `Yes`, `No`, `Cancel` и т.п.) автоматически выполняют это действие: помещают нужное число в `ModalResult` и закрывают окно.

В файле проекта рядом с описанием включенного в проект модуля содержится строка комментария, в которой Delphi указывает имя файла формы. Этот комментарий появляется в диалоговом окне после выбора опции `File/Forms` главного меню. Если поместить в этом комментарии произвольный текст, он также будет виден в окне и поможет вспомнить назначение конкретной формы.

Содержание

1. НАЧАЛО РАБОТЫ 3

1.1. ГЛАВНОЕ ОКНО 4
1.1.1. Пиктографические кнопки 4
1.1.2. Палитра компонентов 5
1.2. ОКНО ФОРМЫ 5
1.3. ОКНО ИНСПЕКТОРА ОБЪЕКТОВ 6
1.4. ОКНО КОДА ПРОГРАММЫ 6
1.5. РАЗМЕЩЕНИЕ КОМПОНЕНТОВ НА ФОРМЕ 6

2. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ (ЯЗЫК OBJECT PASCAL) 7

2.1. Классы 7
2.1.1. Класс TForm и реакция на события 8
2.1.2. Основные понятия класса 9
2.1.3. Составляющие класса 10
2.1.4. Объявления класса 10
2.2. Структура программы в Delphi 11
2.3. Новые типы данных в Delphi 11
3. КОМПОНЕНТЫ DELPHI 11
3.1. Компоненты страницы Standard 11
3.2. Компоненты страницы Additional 13
3.3. Компоненты страницы Win95 (Win32) 14
3.4. Компоненты страницы System 16
3.5. Компоненты страницы Samples 17
4. ФОРМА 17
4.1. Разновидности форм 17
4.2. Создание приложений с несколькими формами 17

Учебное издание

РАБОТА В DELPHI

Составитель: *Стенгач Михаил Сергеевич*

Редактор Н.С. Куприянова

Лицензия ЛР 020301 от 30.12.96 г.

Подписано в печать ___.___.200__ г. Формат 60x84 1/16

Бумага офсетная. Печать офсетная.

Усл. печ.л. 1,62 Усл.кр.-отт. 1,74 Уч.-изд.л. 1,75

Тираж 200 экз. Арт. С-__(Д__)/2003. Зак. _____

Самарский государственный аэрокосмический университет имени академика С.П. Королева.

443086 Самара, Московское шоссе, 34.

ИПО СГАУ 443001 Самара, ул. Молодогвардейская, 151.