Delphi

- Delphi — это среда быстрой разработки, в которой в качестве языка программирования используется язык Delphi.
- Язык Delphi — строго типизированный объектно-ориентированный язык, в основе которого лежит Object Pascal.

Embarcadero Delphi

- **Embarcadero Delphi**, ранее **Borland Delphi** и **CodeGear Delphi**, — интегрированная среда разработки ПО для Microsoft Windows на языке Delphi (ранее носившем название Object Pascal), созданная первоначально фирмой Borland и на данный момент принадлежащая и разрабатываемая Embarcadero Technologies. Embarcadero Delphi является частью пакета Embarcadero RAD Studio.

RAD

- **RAD** (от англ. *rapid application development* — быстрая разработка приложений) — концепция создания средств разработки программных продуктов, уделяющая особое внимание быстроте и удобству программирования, созданию технологического процесса, позволяющего программисту максимально быстро создавать компьютерные программы.

Этапы разработки приложения

1. Постановка задачи, изучение предметной области, построение модели (математической, информационной);
2. Проектирование ООП-приложения:
   - разработать эскизы того, что должно появляться на экране компьютера,
   - написать сценарий работы будущей программы,
   - разработать алгоритмы процедур, реализующих предусмотренные в сценарии действия;
3. Разработка интерфейса пользователя;
4. Программирование приложения;
5. Тестирование и отладка;
6. Разработка документации.

Этапы разработки программы

При создании приложения необходимо:

- уяснить задачу, которую надо решить;
- разработать эскизы того, что должно появляться на экране компьютера;
- написать сценарий работы будущей программы;
- разработать (при необходимости) алгоритмы процедур, реализующих предусмотренные в сценарии действия;
- реализовать проект;
- выполнить тестирование и отладку;
- подготовить проект к распространению

Этапы реализации проекта в среде RAD

1. создание интерфейса приложения

Интерфейс определяет способ взаимодействия пользователя и приложения: какие применяются окна, каким образом пользователь управляет приложением.

Интерфейс создаётся путём размещения на форме компонентов. При проектировании интерфейса приложения действует принцип **WYSIWYG (What You See Is What You Get)** – что видите, то и получите.

- *создание формы*; при этом минимальный код строится автоматически и сразу получается работоспособная программа;
- *расстановка на форме элементов интерфейса* (поля ввода, кнопки, списки) с помощью мыши;

2. определение функциональности

Функциональность определяется процедурами, которые выполняются при возникновении определённых событий, происходящих при действиях пользователя.

- *создание обработчиков событий* двойным щелчком мыши, минимальный код также строится автоматически;
- *написание кода обработчиков*, который реализует нужные алгоритмы обработки данных.

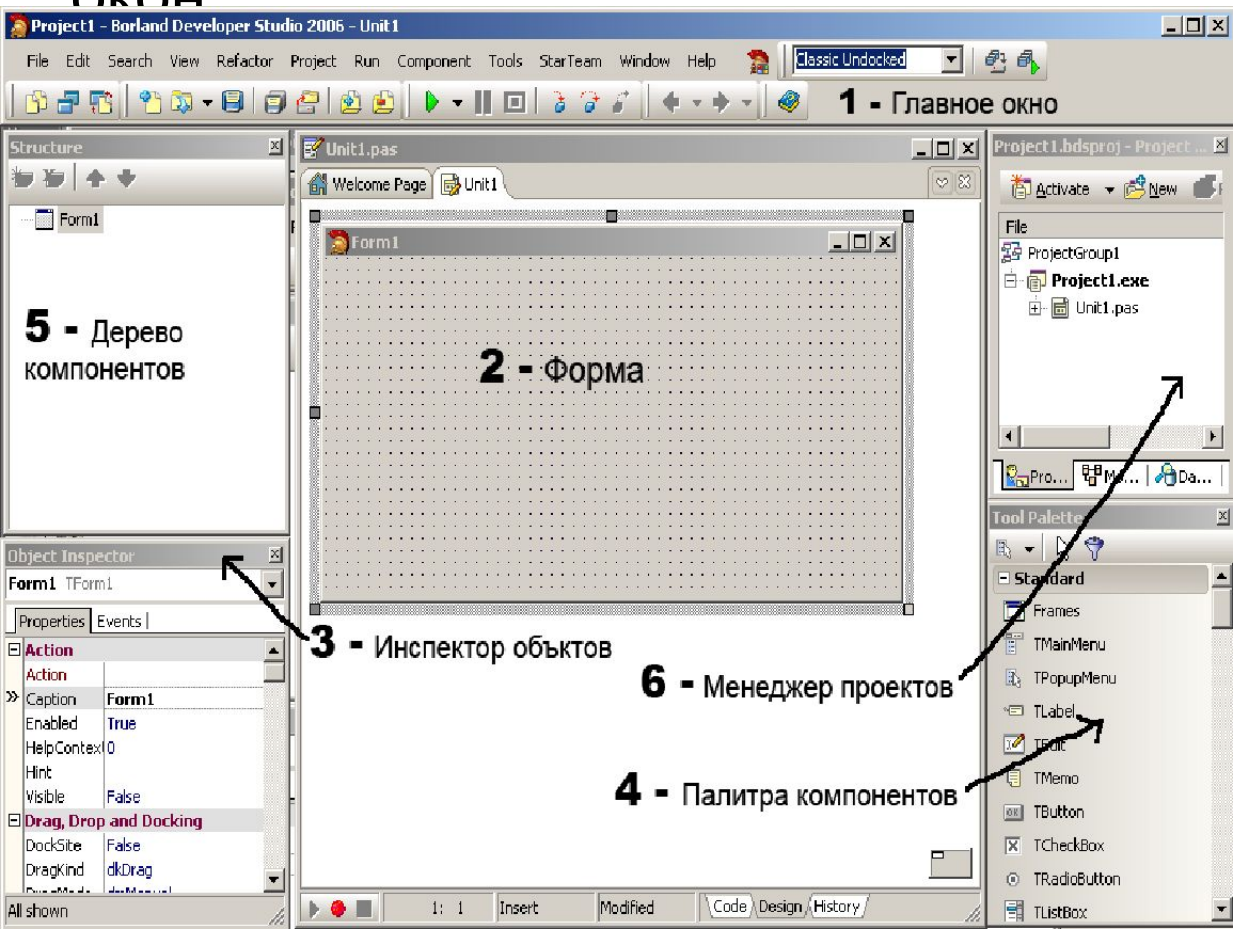
Среда разработки Delphi предоставляет пользователю формы, где размещаются с помощью мыши необходимые компоненты, имеющиеся в библиотеке Delphi. С помощью манипуляций мышью можно изменять размеры и расположение этих компонент.

В процессе проектирования можно постоянно видеть результат – изображение формы и расположенных на ней компонентов.

Во время проектирования формы редактор кода Delphi автоматически генерирует код программы, включая в нее соответствующие фрагменты, описывающие данный компонент. В соответствующих диалоговых окнах можно изменить заданные по умолчанию свойства компонентов и, при необходимости, написать обработчики событий.

ИНТЕРФЕЙС СРЕДЫ DELPHI

После запуска Delphi на экране отображаются несколько окон:



1. Главное окно программы,
2. Окно Конструктора формы,
3. Окно Инспектора объектов,
4. Окно Палитры компонентов,
5. Окно Дерева компонентов,
6. Окно Менеджера проектов.

Окна Delphi можно перемещать, убирать с экрана, а также изменять их размеры.

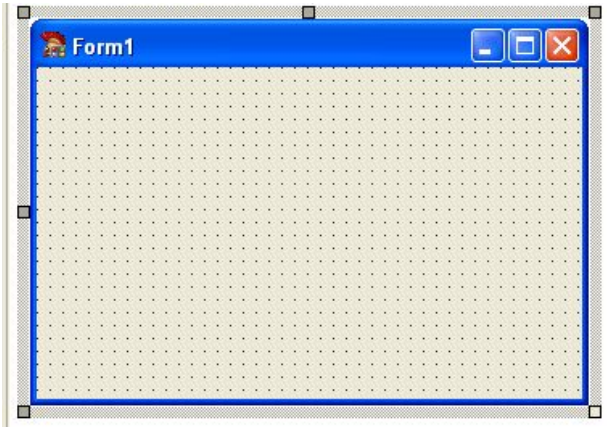
Delphi — однодокументная среда, т.е. среда, позволяющая одновременно работать только с одним приложением.

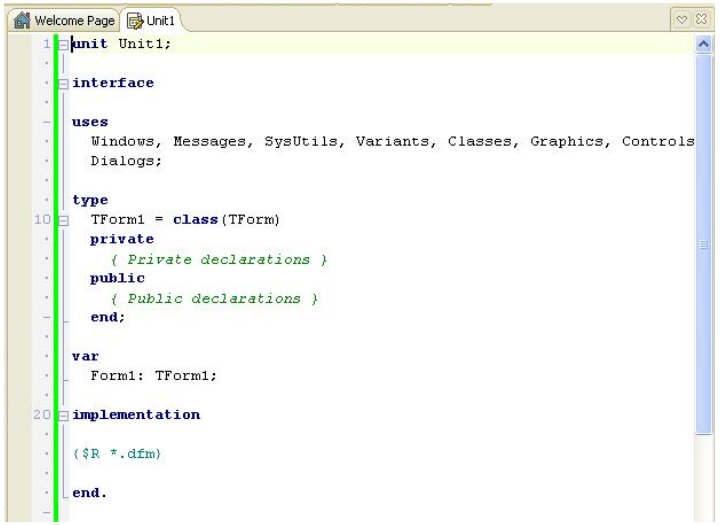
Главное окно

- **Главное окно** осуществляет основные функции управления проектом создаваемой программы. В нем находится основное меню и панели инструментов.
- **Примечание** Главное окно остается открытым все время работы **IDE**. Закрывая его, вы тем самым закрываете Delphi и все открытые в нем окна.

Окно конструктора формы

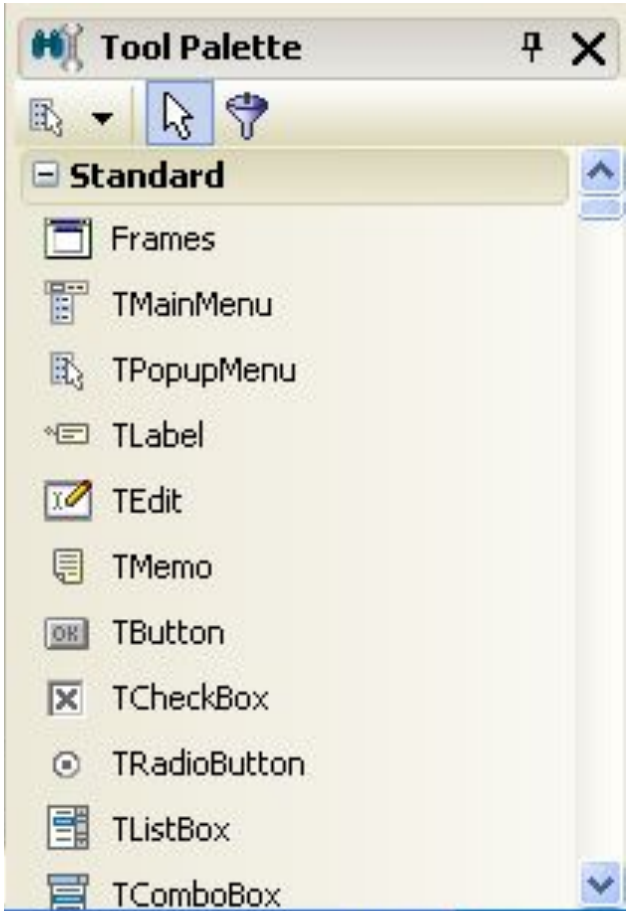
- **Окно конструктора (проектировщика) формы** – главное место, где происходит сборка программы из компонентов, содержащихся в палитре компонентов. Сама форма – это уже готовая к исполнению программа. По ходу работы система формирует в окне **Code** (Редактора кодов) текст программы на языке Object Pascal, связанной с формой.
- **Code Editor** (Редактор кода) представляет собой текстовый редактор с подсветкой синтаксиса языка программирования.





Окно Палитры компонентов

- **Tool Palette** (Палитра компонентов) представляет собой множественные тематические страницы, на которых располагаются компоненты. Все компоненты разделены на категории, которые можно для удобства сворачивать и разворачивать.
- Работа с палитрой компонентов очень проста. В режиме редактирования формы можно дважды нажать левой кнопкой мыши на интересующем компоненте, и он будет добавлен на форму, или перетащить объект на форму, зажав левую кнопку мыши.



Окно Инспектора объектов

Object Inspector (Инспектор объектов) - отображает свойства активного компонента или самой формы. Имя активного компонента находится под заголовком панели.

Эта панель имеет две вкладки – **Properties** (Свойства) и **Events** (События).

На первой вкладке (**Properties**) постоянно отображаются все доступные свойства выбранного компонента. В левой колонке содержится список, а в правой – текущие значения по умолчанию.

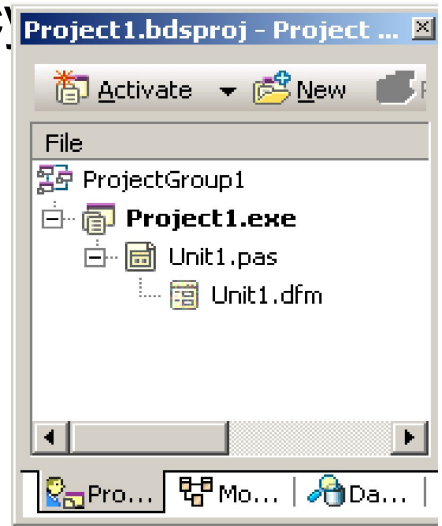
На второй закладке (**Events**) отображаются возможные обработчики событий для выбранного компонента. В левой колонке – названия, а в правой – соответствующие свойства или процедуры.

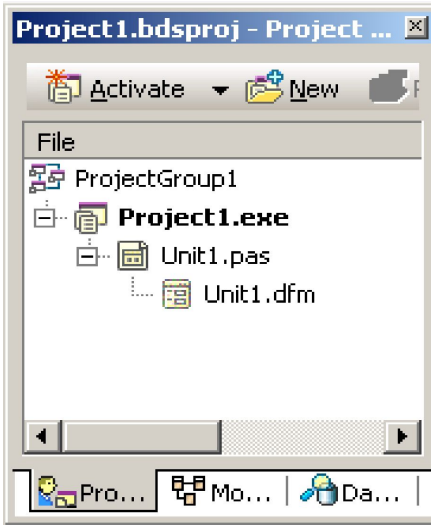
Окно Дерево компонентов

- На панели **Object TreeView** (Дерево компонентов) представлен полный перечень всех компонентов вашего проекта. Здесь легко найти компонент даже, если он перекрывается другим компонентом.

Окно Менеджера проектов

- Файлы, образующие приложение, формы и модули – собраны в проект. **Project Manager** (Менеджер проектов) показывает списки файлов и модулей приложения и позволяет осуществлять навигацию между ними.
- Структура проекта выполнена в виде иерархичного дерева. Корнем дерева проектов является **Project Group** (Группа проектов). Она объединяет в себе один или более проектов. Каждый проект, в свою очередь, содержит файлы кода, заголовочные файлы, файлы рес прочее.

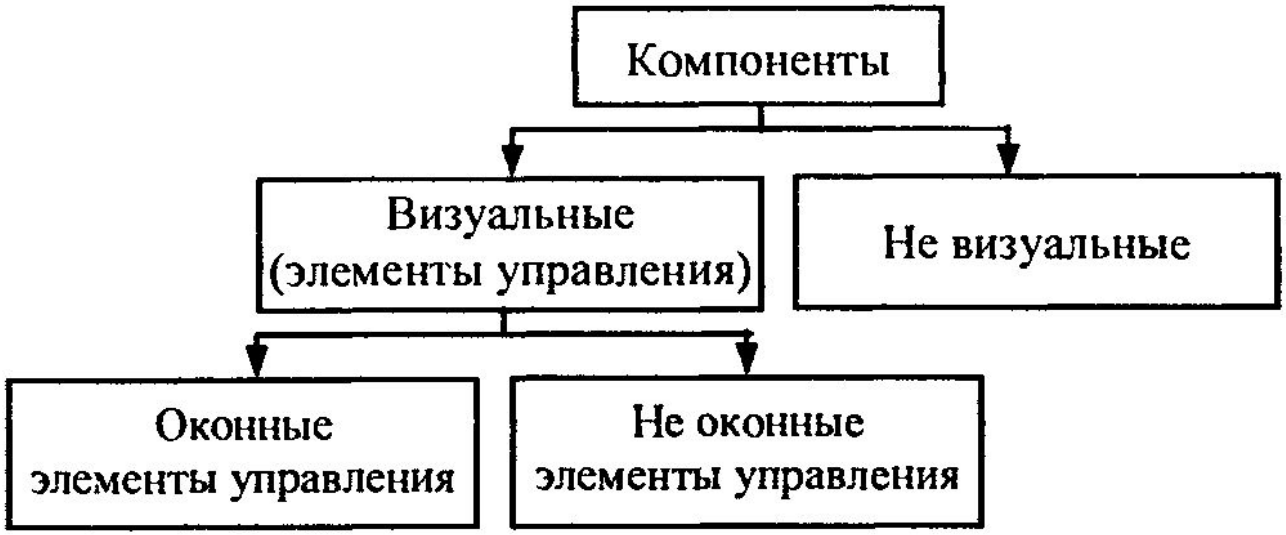




- ProjectGroup1 (Заголовок дерева) – имя группы проектов.
- Project1.exe – имя проекта (приложения). Когда создается новое приложение, Delphi дает ему имя Project, прибавляя порядковый номер.
- Unit1.pas – модуль. Проект состоит из отдельных модулей. Каждое окно программы хранится в отдельном модуле. Файлы с расширением pas содержат исходный код модуля.
- Unit1.dfm – визуальная форма. Она сохраняется в файле с таким же именем, как и у модуля, но с расширением dfm.

Если в проекте несколько приложений, то только одно из них является активным, и только с ним вы можете работать. Имя активного приложения выделено жирным шрифтом. Чтобы изменить активное приложение, достаточно дважды щелкнуть по его имени левой кнопкой мыши.

Иерархия групп компонентов

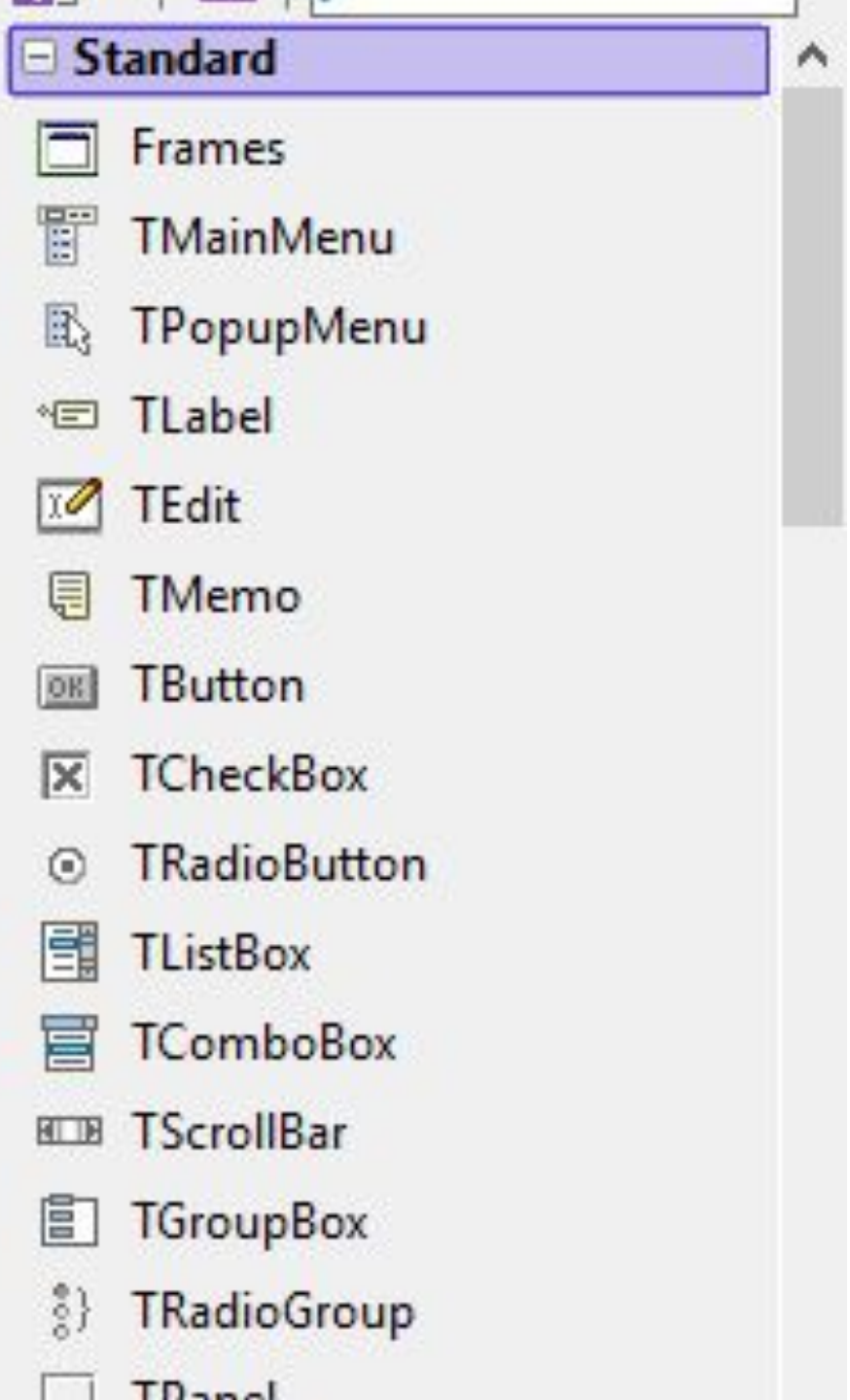


Визуальные компоненты (*элементы управления*) характеризуются наличием свойств размеров и положения в области окна и на стадии разработки приложения обычно находятся на форме в том же месте, что и во время выполнения приложения (например, кнопки, списки, переключатели, надписи). Визуальные компоненты имеют две разновидности — «оконные» и «неоконные» (графические).

- *«Оконные»* визуальные компоненты (самая многочисленная группа компонентов) — это компоненты, которые могут получать *фокус ввода* (т.е. становиться активными для взаимодействия с пользователем) и содержать другие визуальные компоненты.

- *«Неоконные»* (графические) визуальные компоненты не могут получать фокус и содержать другие визуальные компоненты (например, надписи и графические кнопки).

Невизуальные компоненты на стадии разработки не имеют своего фиксированного местоположения и размеров. Во время выполнения приложения некоторые из них иногда становятся видимыми (например, стандартные диалоговые окна открытия и сохранения файлов), а другие остаются невидимыми всегда (например, таблицы базы данных).


Standard
Frames
TMainMenu
TPopupMenu
TLabel
TEdit
TMemo
TButton
TCheckBox
TRadioButton
TListBox
TComboBox
TScrollBar
TGroupBox
TRadioGroup

Основные компоненты

Frames — позволяет формировать на форме фреймы.

TMainMenu — позволяет поместить главное меню в разрабатываемую программу. Создание меню включает в себя три шага:

- помещение TMainMenu на форму;
- вызов Дизайнера меню через свойство Items в Инспекторе объектов;
- определение пунктов меню в Дизайнере меню.

TPopupMenu — позволяет создавать контекстные меню. У всех видимых объектов имеется свойство PopupMenu, где и указывается требуемое меню. Создается аналогично главному меню.

TLabel — служит для отображения текста на экране.

TEdit — стандартный управляющий элемент Windows для ввода. Может использоваться для отображения короткого фрагмента текста и позволяет пользователю вводить текст во время выполнения программы.

TMemo — другая форма TEdit. Подразумевает работу с большими текстами. Может переносить слова, сохранять в ClipBoard фрагменты текста и восстанавливать их, выполнять другие основные функции редактора. Имеет ограничения на объем текста — 32 Кбайт.

TButton — позволяет выполнить какие-либо действия при нажатии кнопки во время выполнения программы.

Вкладка Standart

Вкладка Standart

 TCheckBox — отображает строку текста и небольшое рядом стоящее окошко, в котором можно поставить отметку, означающую, что что-то выбрано (объект выбора описывается указанной строкой текста).

 TRadioButton — позволяет выбрать только одну опцию из нескольких.

 TListBox — требуется для показа прокручиваемого списка.

 TComboBox — аналогичен TListBox и кроме того позволяет водить информацию в поле ввода поверх TListBox. Имеется несколько типов TComboBox, но наиболее популярен спадающий вниз.

 TScrollbar — полоса прокрутки, появляющаяся автоматически в объектах редактирования при необходимости прокрутки текста для просмотра.

Вкладка Additional

 TBitBtn — кнопка, аналогичная TButton, однако на ней можно разместить картинку (glyph). Имеет несколько предопределенных типов (bkClose, bkOK и др.), при выборе которых принимает соответствующий вид.

 TTabSet — горизонтальные закладки, обычно используемые вместе с TNoteBook для создания многостраничных окон.

 TNoteBook — используется совместно с TTabSet для создания многостраничного диалога, причем на каждой странице располагается свой набор объектов.

 TOutline — используется для представления иерархических отношений связанных данных (например, дерева директорий).

Свойства формы

Свойство	Описание
Name	Имя формы. Используется в программе для управления формой и для доступа к ее компонентам
Caption	Текст заголовка
Top	Расстояние от верхней границы формы до верхней границы экрана
Left	Расстояние от лев… экрана
Width	Ширина формы
Height	Высота формы
ClientWidth, ClientHeight	Соответственно ш… области формы, т… границ
BorderStyle	Вид границы, кот… тонкой (bsSingle) … обычная граница, …ватель может с по… Изменить размер … граница отсутству… заголовка, положе… программы измен…

Свойство	Описание
BorderIcons	Кнопки управления окном. Значение свойства определяет, какие кнопки управления окном будут доступны пользователю во время работы программы. Значение свойства задается посредством присвоения значений уточняющим свойствам: biSystemMenu, biMinimize, biMaximize и biHelp. Свойство biSysteinMenu определяет доступность кнопки [Свернуть] и кнопки системного меню, biMinimize — доступность кнопки [Свернуть], biMaximize — доступность кнопки [Развернуть], biHelp — доступность кнопки вывода справочной информации
Icon	Значок в заголовке диалогового окна, обозначающий кнопку вывода системного меню
Color	Цвет фона, который можно задать, указав его название или привязку к текущей цветовой схеме операционной системы. В последнем случае цвет определяется текущей цветовой схемой, выбранным компонентом привязки и изменяется при изменении цветовой схемы операционной системы
Font	Шрифт, используемый по умолчанию компонентами, находящимися на поверхности формы
Canvas	Поверхность, на которую можно вывести графику

Свойства текста

Свойство	Описание
Name	Имя компонента. Используется в программе для доступа к компоненту и его свойствам
Caption	Отображаемый текст
Left	Расстояние от левой границы поля вывода до левой границы формы
Top	Расстояние от верхней границы формы
Height	Высота поля вывода
Width	Ширина поля вывода
AutoSize	Признак того, что разм
Wordwrap	Признак того, что сло щей строке, автоматич строку (значение свой

Свойство	Описание
Alignment	Задает способ выравнивания текста внутри поля: по левому краю (taLeftJustify), по центру (taCenter) или по правому краю (taRightJustify)
Font	Шрифт, используемый для отображения текста. Уточняющие свойства определяют способ начертания символов (Font.Name), их размер (Font.Size) и цвет (Font.Color)
ParentFont	Признак наследования компонентом характеристик шрифта формы, на которой он находится. Если значение свойства равно True, то текст выводится шрифтом, установленным для формы
Color	Цвет фона области вывода текста
Transparent	Управляет отображением фона области вывода текста. Значение True делает область вывода текста прозрачной (область вывода не закрашивается цветом, заданным свойством Color)
Visible	Позволяет скрыть текст (False) или сделать его видимым (True)

Свойства командной кнопки

Свойство	Описание
Name	Имя компонента. Используется в программе для доступа к компоненту и его свойствам
Caption	Текст на кнопке
Left	Расстояние от левой границы кнопки до левой границы формы
Top	Расстояние от верхней границы кнопки до верхней границы формы
Height	Высота кнопки
Width	Ширина кнопки
Enabled	Признак доступности кнопки. Если значени True, то кнопка доступна. Если значение св False, то кнопка не доступна, например в р мышью на кнопке событие Click не возника
Visible	Позволяет скрыть кнопку (False) или сделать
Hint	Всплывающая подсказка-текст, который по указателем мыши при позиционировании у мандной кнопке (чтобы текст появился, зна ShowHint должно быть True)
ShowHint	Разрешает (True) или запрещает (False) отс сказки при позиционировании указателя на

Свойства Memo

Свойство	Описание
Name	Имя компонента. Используется для доступа к свойствам компонента
Text	Текст, находящийся в поле Memo. Рассматривается как единое целое
Lines	Массив строк, соответствующий содержимому поля. Доступ к строке осуществляется по номеру. Строки нумеруются с нуля
Lines.Count	Количество строк текста в поле Memo
Left	Расстояние от левой границы поля до левой границы формы
Top	Расстояние от верхней границы поля до верхней границы формы
Height	Высота поля
Width	Ширина поля
Font	Шрифт, используемый для отображения вводимого текста
ParentFont	Признак наследования свойств шрифта «родительской» формы

Свойства радиокнопки

Свойство	Описание
Name	Имя компонента. Используется для доступа к свойствам компонента
Caption	Текст, который находится справа от кнопки
Checked	Состояние, вид кнопки: если кнопка выб True; если кнопка не выбрана, то Checke
Left	Расстояние от левой границы флажка до формы
Top	Расстояние от верхней границы флажка формы
Height	Высота поля вывода поясняющего текста
Width	Ширина поля вывода поясняющего текс
Font	Шрифт, используемый для отображения
ParentFont	Признак наследования характеристик шр ской» формы

Свойства переключателя

0Свойство	Описание
Name	Имя компонента. Используется для доступа к свойствам компонента
Caption	Текст, который находится справа от флажка
Checked	Состояние, вид флажка: если флажок установлен, то Checked = True; если флажок сброшен, то Checked = False
State	Состояние флажка. В отличие от свойства Checked, позволяет различать установленное, сброшенное и промежуточное состояния. Состояние флажка определяет одна из констант: cbChecked (установлен); cbGrayed (серый, неопределенное состояние); cbUnChecked (сброшен)
AllowGrayed	Определяет, может ли флажок быть в промежуточном состоянии: если AllowGrayed = False, то флажок может быть только установленным или сброшенным; если AllowGrayed = True, то допустимо промежуточное состояние
Left	Расстояние от левой границы флажка до левой границы формы
Top	Расстояние от верхней границы флажка до верхней границы формы
Height	Высота поля вывода поясняющего текста
Width	Ширина поля вывода поясняющего текста
Font	Шрифт, используемый для отображения поясняющего текста
ParentFont	Признак наследования характеристик шрифта «родительской» формы

Свойства списка

Свойство	Описание
Name	Имя компонента. В программе исполь к компоненту и его свойствам
Items	Элементы списка — массив строк
Count	Количество элементов списка
Sorted	Признак необходимости автоматическ списка после добавления очередного з
ItemIndex	Номер выбранного элемента (элемент с нуля). Если в списке ни один из эле значение свойства равно –1
Left	Расстояние от левой границы списка д формы
Top	Расстояние от верхней границы списк формы
Height	Высота поля списка
width	Ширина поля списка
Font	Шрифт, используемый для отображен
ParentFont	Признак наследования свойств шрифта формы

Свойства BitBtn

Свойство	Описание
Name	Имя компонента. Используется в программе для доступа к компоненту и его свойствам
Caption	Текст, отображаемый на кнопке
Font	Шрифт, который используется для отображения текста
Left	Расстояние от левой границы формы до левой границы компонента
Top	Расстояние от верхней границы формы до верхней границы компонента
Width	Ширина поля компонента
Height	Высота поля компонента
Enabled	Признак доступности кнопки: кнопка доступна, если значение свойства равно True, и недоступна, если оно равно False
Visible	Признак видимости кнопки на поверхности формы: если значение свойства равно True — кнопка отображается, в противном случае она невидима
Glyph	Картинка, отображаемая на кнопке (файл изображения)
NumGlyphs	Количество картинок в файле изображения, указанного в свойстве Glyph
Layout	Определяет взаимоположение картинки и текста на кнопке. Может принимать следующие значения: blGlyphLeft — картинка располагается слева от надписи, blGlyphRight — справа, blGlyphTop — сверху, blGlyphBottom — снизу
Spacing	Расстояние от картинки до надписи. Расстояние задается в пикселах

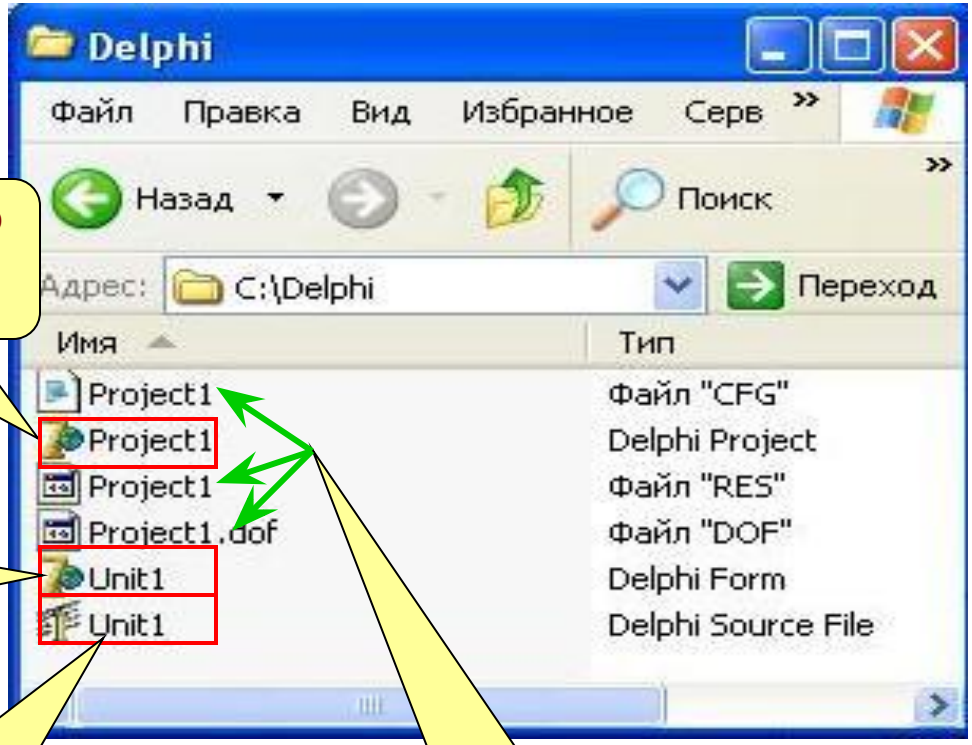
Некоторые события Delphi

- **onClick** - возникает при щелчке мышкой на компоненте
- **onDblClick** - возникает при двойном щелчке мышкой по компоненту;
- **onKeyDown** - когда при нажатии на кнопку на клавиатуре она оказалась в нижнем положении;
- **onKeyUp** - когда при отпускании клавиатурной кнопки она оказалась в верхнем положении;
- **onKeyPress** - возникает при нажатии на клавиатурную кнопку. От событий onKeyDown иonKeyUp оно отличается типом используемого параметра Key ;
- **onMouseDown** - когда при нажатии кнопки мышки она оказалась в нижнем положении;
- **onMouseUp** - когда при отпускании кнопки мышки она оказалась в верхнем положении;
- **onMouseMove** - возникает при перемещении указателя мышки над компонентом.
- **onChange** – возникает при изменении текста компонента Edit
- **onEnter** – возникает, когда объект получает фокус ввода
- **onExit** - возникает, когда компонент теряет фокус ввода;

Создание и сохранение проекта

- Команда **File/New/VCL Forms Aplication** создает новый проект
- Сохранить проект можно так же с помощью команды **File/Save Project As..**
- Для каждого нового проекта лучше отводить отдельную папку, т.к. проекты включают множество файлов. Это позволит вам упростить процесс копирования проекта на другой компьютер.
- Названия файлов должны состоять из букв английского алфавита, символа подчеркивания и арабских цифр, первым символом не может быть цифра. Пробелы – недопустимы.

Какие файлы у нас сохранились?



Главный файл нашего проекта

Файл нашей формы с описанием ее свойств

Файл модуля с расширением *.pas – здесь исходный код нашей программы

Дополнительные файлы ресурсов, которые Delphi создает автоматически

ОБЩАЯ ОРГАНИЗАЦИЯ ПРОГРАММЫ В DELPHI

Общая организация программы в Delphi

- Программа создаваемая в среде Delphi в процессе проектирования приложения, основана на *модульном принципе*.
- В среде Delphi разрабатывается проект – набор файлов, из которых состоит приложение.

Состав проекта

В любой проект входит файлы:

- **project1.dpr** – главный файл проекта, формируется системой при создании нового приложения;
- **unit1.pas** – первый модуль (unit) программы, который автоматически появляется в начале работы;
- **unit1.dfm** – файл описания формы, используется для сохранения информации о внешнем виде главной формы;
- **project1.res** – файл ресурсов, в нём хранятся иконки, растровые изображения, курсоры. Как минимум, содержит иконку приложения;
- **project1.dof** – файл опций, является текстовым файлом для сохранения установок, связанных с данным проектом (например директив компилятора);
- **project1.cfg** – файл конфигурации, содержит информацию о состоянии среды.

Структура головной программы приложения Delphi

- Главный файл проекта представляет собой текстовый файл, содержащий программный код, записанный на языке Object Pascal. Этот файл подключает все используемые программные модули и содержит операторы для запуска приложения.

Структура головной программы приложения Delphi

```
program Project1;
uses
  Vcl.Forms,
  Unit1 in 'Unit1.pas' {Form
{$R *.res}
begin
  Application.Initialize;
  Application.CreateForm(TForm
  Application.Run;
end.
```

Структуры модуля приложения Delphi

- Текст программ хранится в модулях, название которого должно совпадать с именем файла. Модуль состоит из трех разделов: *интерфейса, реализации* и *инициализации*. Структура модуля приложения Delphi:

`Unit Unit1;`	Название модуля (Это название используется в предложении `Uses` при подключении модуля к программе).
`Interface` …	Раздел интерфейса
`Implementation` …	Раздел реализации (исполняемая часть)

`begin`
…
`end.`
	Раздел инициализации

Раздел интерфейса

- Начинается ключевым словом **Interface**.
- Сообщает компилятору, какая часть модуля является доступной для других модулей программы.
- Здесь могут размещаться списки подключаемых модулей, объявления типов, констант, переменных, заголовки функций и процедур, к которым будет доступ из других модулей. Иными словами, в этом разделе перечисляется, все то, что должно быть видимым из программы, которая его использует.

unit1.pas

Unit1

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Control
  Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
Form1.Color:=RGB(100+random(155), 100+random(155), 100+random(155))
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
 close
end;
```

заголовок модуля проекта – **Unit 1**

Интерфейсная часть (объявление всех объектов модуля – типов, переменных …)

unit1.pas

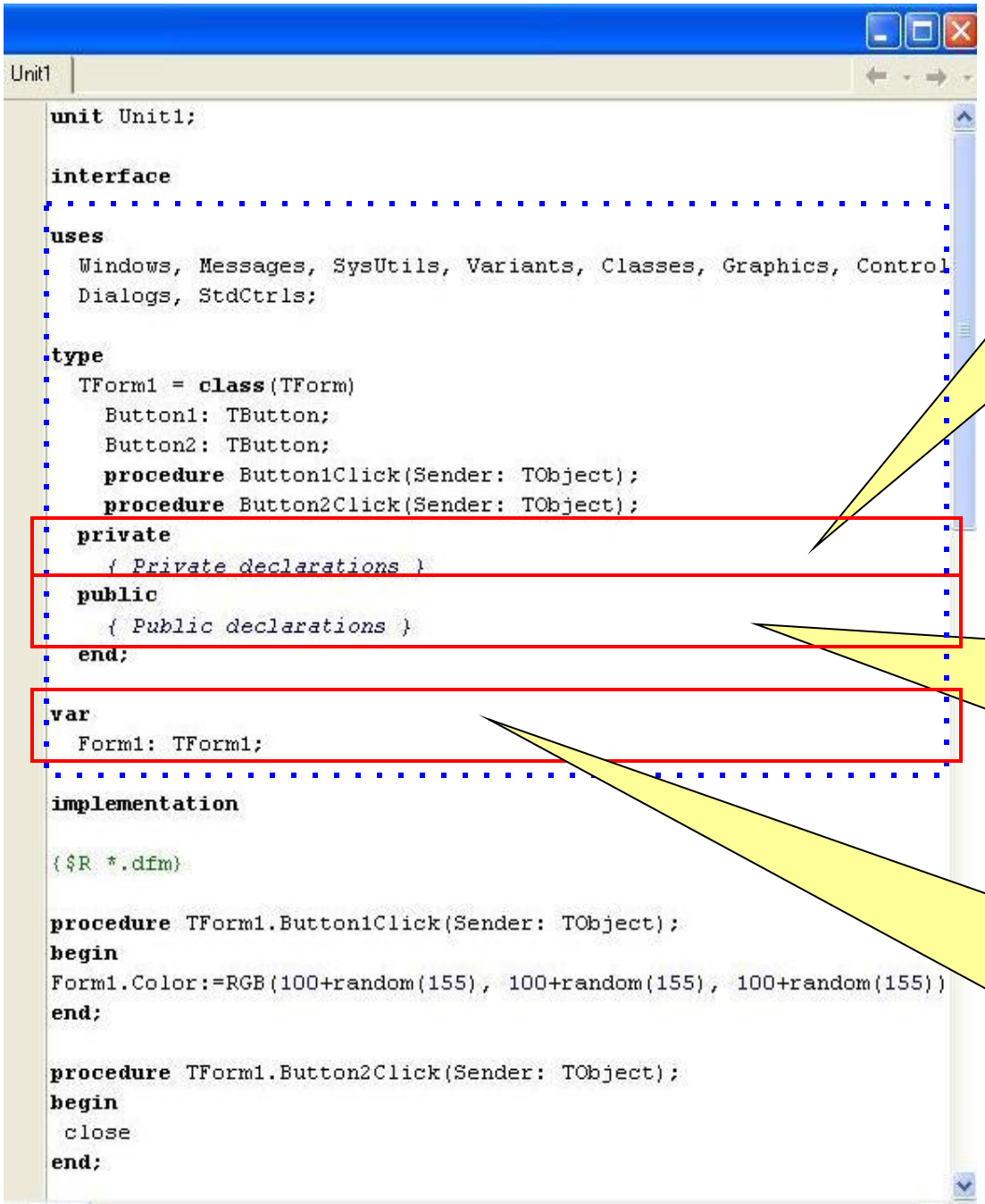
Unit1

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Control
  Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
Form1.Color:=RGB(100+random(155), 100+random(155), 100+random(155))
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
 close
end;
```

Это автоматически подключаемые Delphi модули

объявление объектов, которые используются в нашем проекте: форма (TForm1), Button1 и Button2, а также процедуры обработки событий нажатия на эти кнопки: Button1.Click и Button2.Click

unit1.pas



закрытый раздел класса
Сюда могут помещаться объявления переменных, функций и процедур, включаемых в класс формы, но не доступных для других модулей

Открытый раздел класса
Сюда могут помещаться объявления переменных, функций и процедур, включаемых в класс формы и

Сюда могут помещаться объявления типов, констант, переменных, функций и процедур, к которым будет доступ из других модулей, но которые не включаются в класс формы

Раздел реализации

- Начинается ключевым словом **Implementation** и содержит объявления локальных переменных, процедур и функций, поддерживающих работу формы.
- Директива {$R *.dfm} указывает компилятору, что в раздел реализации нужно вставить инструкции установки значений свойств формы, которые находятся в файле с расширением .dfm, имя которого совпадает с именем модуля.
- Далее в разделе реализации могут помещаться предложения uses, объявления типов, констант, переменных, к которым не будет доступа из других модулей. Здесь же располагаются все тексты процедур и функций, объявленных в разделе interface. Заголовки процедур и функций могут полностью совпадать с заголовками из интерфейсной части или могут отличаться от них полным отсутствием параметров. Если в этой части набран текст функции или процедуры, не представленной в Interface, то данная функция или процедура будет локальной.

unit1.pas

Unit1

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Control
  Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
Form1.Color:=RGB(100+random(155), 100+random(155), 100+random(155));
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
 close
end;
```

Исполняемая часть модуля, содержащая основной код и отражающая логику и алгоритм работы программы

Процедура обработки первой кнопки

Процедура обработки второй кнопки

Структура событийной процедуры

`Procedure <название процедуры>`	Заголовок процедуры. Название процедуры состоит из двух частей: *названия объекта* + *название события*

`Const`
`<имя константы>=<значение константы>;`
`…`
`<имя константы>=<значение константы>;`
	Раздел описания констант

`Type`
`<имя типа> = <тип>;`
`…`
`<имя типа> = <тип>;`
	Раздел типов

`Var`
`<имя переменной>:<тип>;`
`…`
`<имя переменной>:<тип>;`
	Раздел описания переменных

`<Тексты локальных процедур и функций>`	Раздел процедур и функций
`Begin` `<инструкции>`	Раздел, в котором пишутся инструкции процедуры

Раздел инициализации

- **Раздел инициализации** позволяет выполнить инициализацию переменных модуля. Располагается после раздела реализации между Begin и End.
- В этой части помещают операторы, которые должны выполняться один раз при первом обращении к модулю (после ключевого слова Initialization). Или операторы, выполняемые при любом завершении модуля (после ключевого слова Finalization). Эта часть является необязательной. На практике данный подход используется редко, поэтому обычно после Implementation сразу ставится End с точкой (без Begin).

Файл формы

В коде мы видим описание формы и объектов, находящихся на ней

Unit1

```
object Form1: TForm1
  Left = 384
  Top = 90
  Width = 228
  Height = 316
  Caption = #1062#1042#1045#1058
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  OldCreateOrder = False
  PixelsPerInch = 96
  TextHeight = 13
  object Button1: TButton
    Left = 16
    Top = 224
    Width = 75
    Height = 25
    Caption = #1062#1042#1045#1058
    TabOrder = 0
    OnClick = Button1Click
  end
  object Button2: TButton
    Left = 112
    Top = 224
    Width = 75
    Height = 25
    Caption = #1042#1067#1061#1054#1044
    TabOrder = 1
    OnClick = Button2Click
  end
end
```

Описание формы и ее свойств

Описание свойств первой кнопки

Описание свойств второй кнопки

Самостоятельно разберитесь, что означают строки кода для формы и кнопок

Хорошее приложение

Можно считать хорошим приложение, которое:

- имеет простой, удобный, интуитивно понятный интерфейс, со всеми присущими Windows атрибутами: окнами, кнопками, меню и т.д.;
- управляется как мышью, так и клавиатурой;
- отказоустойчиво, корректно обрабатывает любые ошибки пользователя;
- работает быстро;
- имеет хорошую справочную систему;
- обеспечивает обмен данными с другими приложениями;
- может (при необходимости) использовать средства мультимедиа;
- может работать с базами данных;
- разрабатывается быстро.

Разработка интерфейса пользователя.

При разработке интерфейса пользователя необходимо соблюдать ряд правил.

1. При проектировании **окон** (**форм**) ввода данных:

- для команд всегда следует создавать клавишные эквиваленты (при этом необходимо там, где это уместно, сохранять привычные эквиваленты), не заставляя пользователя применять исключительно мышь;
- расположение элементов на главной форме должно быть согласовано с задачами пользователя;
- при вводе данных должна присутствовать заметная, но ненавязчивая связь с пользователем (например, синтаксический контроль и, если возможно, исправление ошибок и опечаток);
- для нескольких разных форм ввода не следует использовать существенно отличающиеся интерфейсы.

Разработка интерфейса пользователя.

2. При создании **меню**:

- следовать стандартным соглашениям о расположении пунктов меню, принятым в Windows;
- группировать пункты меню в логическом порядке и по содержанию;
- для группировки пунктов в раскрывающихся меню использовать разделительные линии;
- избегать избыточных меню;
- использовать клавиатурные эквиваленты команд и «горячие» клавиши;
- помещать на панель инструментов часто используемые команды меню.

Разработка интерфейса пользователя.

3. При работе приложения в процессе **ожидания** следует информировать пользователя о ходе работы (например, с помощью индикатора состояния выполнения задания).
4. Общие **дизайнерские** требования: сочетаемость цвета фона приложения и цвета выводимого на него текста, небольшое количество используемых цветов и др.

Компиляция и выполнение проекта

- Компиляция проекта выполняется командой Project|Compile ProjectName или использованием комбинации клавиш Ctrl+F9. При этом компилируются все исходные модули, содержимое которых изменялось после последней компиляции: для каждого программного модуля создаётся файл с расширением dcu. Затем компилируется файл проекта и компонуется (собирается) из dcu-модулей исполняемый файл, имя которого совпадает с именем файла проекта.
- Готовый к использованию файл может быть приложением (*.exe) или динамически загружаемой библиотекой (*.dll).

Компиляция и выполнение проекта

Готовый проект можно запустить на выполнение. Выполнение приложения из среды Delphi задаётся командой Run|Run (клавишей F9 или кнопкой с зелёным треугольником) и имеет следующие особенности:

- нельзя запустить несколько копий приложения;
- при возникновении исключительных ситуаций сначала выводятся сообщения Delphi, а затем – приложения;
- для аварийного завершения приложения (например при зацикливании) необходимо выполнить команду Run|Program Reset;
- для продолжения разработки проекта приложение надо закрыть.