Объектно-ориентированный подход

RAD-технологии

LOGO

Развитие ПО

LOGO

- Применение научных методик программирования привело к бурному развитию ПО.
- Изменился характер решаемых с помощью компьютера задач. Это задачи обработки распределенной информации различного вида («многосредовые», мультимедиа данные).
- Образовался семантический разрыв между структурой современных задач, решаемых ПО, и процедурным подходом. Осознание этого разрыва привело к появлению объектно-ориентированного подхода к программированию

Развитие ПО

LOGO

- Объектный подход привел к появлению языков программирования нового поколения: C++, Object (Delphi) Pascal, Java, Visual Basic.
- Бурное развитие ПО привело к появлению мощных многозадачных операционных систем с графическим интерфейсом пользователя
- Это стимулировало появление визуальных сред программирования, основанных на RAD (Rapid Application Development – быстрая разработка программ) технологии. Пример – визуальная среда программирования Delphi.

Основы построения RAD-систем

LOGO

- Операционные системы
  - Интерфейс
  - Многозадачность и событийная модель
- Компонентный подход
- Визуальное программирование

Интерфейс

LOGO

- **графический оконный интерфейс пользователя;**
- **универсальную, стандартизованную систему управления работой программ.**

Программы, разрабатываемые в визуальной среде RAD, должны подчиняться этому стандарту.

Многозадачность и событийная модель

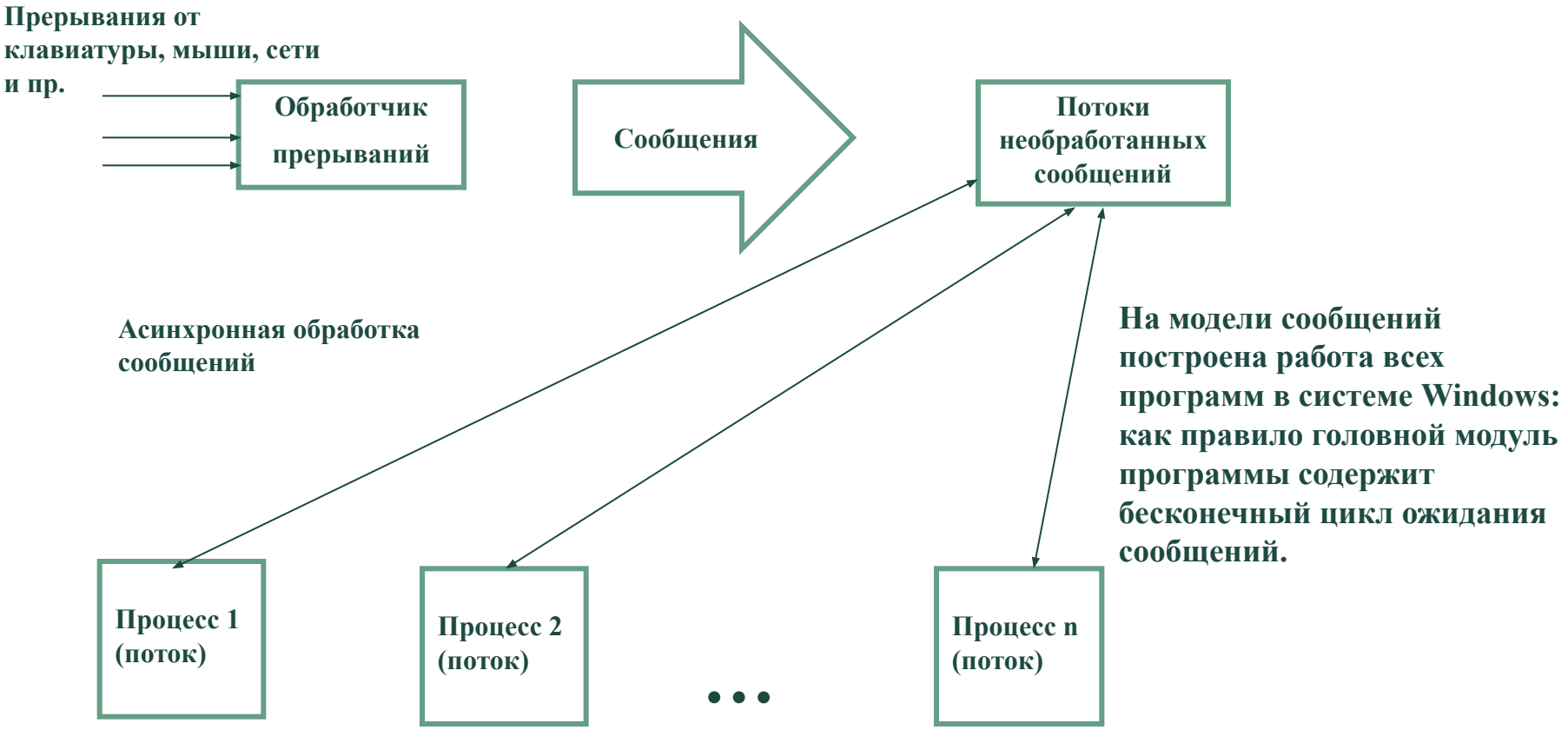
LOGO

Процесс – это загруженное в оперативную память, выполняющееся приложение. Задачи делят между собой пространство памяти компьютера и время микропроцессора.

Поток –это подзадача, выполняемая под управлением родительского процесса. Поток использует память и системные ресурсы, которые предоставил ему родительский процесс. Процесс может состоять из одного, либо нескольких потоков. Каждый процесс имеет свое независимое пространство памяти, доступ к которому из других процессов запрещен.

Модель передачи сообщений в Windows

LOGO



Компонентный подход

LOGO

Компонент – это программный модуль, выполняющий определенную задачу. Откомпилированный код компонента хранится в одной из библиотек RAD-среды разработки и недоступен для изменения.

Компоненты делятся на **визуальные и невизуальные компоненты.**

Визуальные компоненты во время выполнения программы формируют интерфейс пользователя – это кнопки, текстовые блоки, списки, флажки и пр.

Невизуальные компоненты – реализуют обмен данными.

Группы компонентов образуют **палитру компонентов**. Компоненты, как правило, входят в состав групп компонентов, формируемых по признаку близости решаемых ими задач (закладки Standart, Additional, Win32,

Визуальное программирование

LOGO

Во всех визуальных средах основой для создания приложения является **форма.** Приложение может содержать несколько форм, которые могут раскрываться как в отдельных окнах, так и внутри других окон (многооконные приложения).

На форму при программировании помещаются визуальные и невизуальные компоненты. В RAD-средах широко используется известный метод Drag and Drop – «перетащи и брось»: компоненты перетаскиваются мышью на форму. С помощью мыши можно также изменять размер визуальных компонентов и положение компонентов на форме. Для более тонкой настройки свойств компонентов служит специальный инструмент – **Object Inspector (Инспектор объектов) в Delphi и окно Properties (Свойства) в Visual Basic.**

Порядок создания простого приложения

LOGO

1. Программист выбирает нужные ему компоненты на соответствующих закладках палитры компонентов (компонент Label на закладке Standard), размещая их на форме, меняя при необходимости их положение и размер.
2. Далее на закладке Properties окна Object Inspector можно провести тонкую настройку компонента, например, сформировать надпись (свойство Caption компонента Label), изменить названия кнопок, установить цвет, гарнитуру, размер шрифта (свойство Font…) и т.п.
3. Функциональность приложения формируется с помощью **обработчиков событий**. Каждый компонент имеет свой собственный набор событий, реакцию на которые можно задать с помощью процедур и функций. Обработчики событий создаются в **Редакторе кода** среды разработки.

Редактор кода

LOGO

Редактор кода вызывается, когда программист выбирает для выделенного компонента формы (либо самой формы) **обработчик событий на закладке Events Инспектора Объектов.** В верхней части окна Редактора Кода имеется раскрывающийся список доступных для данного компонента обработчиков событий (например OnClick, OnKeyDown и т.п.).

При выборе нужного события в окне кода появляется **шаблон обработчика события** – процедура (функция) с уже сформированным заголовком и телом процедуры (функции). Программист добавляет содержательную часть тела процедуры.

Список основных файлов

LOGO

Среда разработки порождает,как правило,множество файлов, образующих **проект**. Основными из них являются:

- ***.dpr** – головной модуль проекта;
- ***.dfm** – файлы описания форм проекта;
- ***.pas** – модули исходного кода форм, оформленные как модули (Unit) Паскаля;
- ***.dof, *.cfg, *.res** – ряд файлов, описывающих состояние среды разработки для данного проекта;
- ***.exe** – загрузочный файл проекта.

Головной модуль проекта

LOGO

Головной модуль проекта формируется автоматически и для всех проектов имеет одинаковую структуру:

- **Инициализация приложения;**
- **Создание формы;**
- **Запуск цикла обработки сообщений.**

Пример кода головной программы:

```
Program Project1;
Uses
Forms, Unit1  in Únit1.pas’{Form1};
{$R*.RES}
Begin
Application.Initialize;
Application.CreateForm(TForm1,Form1);
Application.Run;
End.
```

Основные принципы ООП

LOGO

Инкапсуляция

Наследование

Полиморфизм

LOGO

- Процедурный подход – описание реальных систем в виде последовательности действий
- Объектно-ориентированный подход – описание системы в виде взаимодействия объектов

Объект – базовое понятие в ООП

LOGO

Объект — имеет → Свойства или характеристики (атрибуты)

Состояние объекта задается значениями его признаков

Объект → методы

Объект → События, на которые он реагирует

Объект «знает», как решать определенные задачи, т.е. располагает методами решения. Это определяет поведение объекта.

ИНКАПСУЛЯЦИЯ

LOGO

Инкапсуляция – это сокрытие информации о внутреннем устройстве объекта.

В основе построения объектно-ориентированных систем положен принцип: **объекты должны знать об устройстве друг друга только то, что необходимо для их взаимодействия и не более того.**

ИНКАПСУЛЯЦИЯ

LOGO

Инкапсуляция – объединение данных и обрабатывающих их методов (подпрограмм) внутри класса. ***Класс – специальный тип данных для описания объектов.***

В классе инкапсулируются (объединяются и помещаются внутри класса) поля, свойства и методы. Класс приобретает функциональность.

Состояние объекта

LOGO

У объекта есть

- *состояние*
- *поведение*
- *возможность отличить его от других объектов.*

Состояние объекта характеризуется текущим значением его *атрибутов.*

Идентификация объекта

LOGO

У каждого объекта должна быть уникальная характеристика

Ответственность за правильность идентификации объекта лежит на программисте, разрабатывающем систему.

Многие объектные модели предлагают встроенные методы идентификации объектов.

Интерфейс объекта

LOGO

Интерфейс – это описание того, как объект взаимодействует с окружающим миром.

Объекты взаимодействуют между собой с помощью сообщений. Получая сообщение объект выполняет действие. Эти действия называются методами.

Интерфейс объекта

LOGO

Объект известен другим объектам только по своему интерфейсу. Внутренняя структура его скрыта.

Суть принципа инкапсуляции – в разделении внутренней структуры и интерфейса объекта.

Классы

LOGO

- однотипные объекты объединяются в классы
- все объекты одного и того же класса обладают одинаковым интерфейсом и реализуют этот интерфейс одним и тем же способом

Классы

LOGO

Объект, имеющий тип какого-либо класса, является **экземпляром** этого класса или переменной этого типа;

Класс – особый тип записи из элементов:

– поля;

– свойства;

– методы.

Классы

LOGO

- **поля** – аналогично полям записи; служат для хранения информации об объекте;
- **методы** – процедуры и функции для обработки полей;
- **свойства** – промежуточное понятие между полями и методами:
  - свойства можно использовать как поля, присваивая им значения с помощью оператора «:=»;
  - с другой стороны, внутри класса доступ к значениям свойств выполняется методом класса.

Поля

LOGO

Поле класса описывается как обычная переменная и может быть любого типа. Согласно принятому соглашению имена полей должны начинаться с префикса F (field – поле).

Пример:

```
Type TNewClass=Class (TObject)
    Private
        FCode:integer;
        FSign:char;
        FNote:string;
End;
```

Чем дальше от базового класса, тем больше полей у объекта-потомка.

Методы

LOGO

По умолчанию все методы, объявленные в классе, являются статическими – они вызываются как обычные подпрограммы.

Методы, которые предназначены для создания объектов, называются **конструкторами**, а для удаления объектов – **деструкторами**.

Наследование

LOGO

Заключается в порождении новых **объектов-потомков** от существующих **объектов-родителей**, при этом потомок берет от родителя все его поля, свойства и методы. В дальнейшем наследуемые поля можно использовать в неизменном виде или переопределять (модифицировать).

Наследование

LOGO

Из нового объекта можно породить следующий объект. Образуется дерево объектов или иерархия классов:

- в начале дерева – базовый класс TObject, который реализует элементы, наиболее общие для всех объектов (создание, удаление объекта);
- чем дальше объект от базового класса, тем более он специфичен.

Полиморфизм

LOGO

Полиморфизм - заключается в том, что методы различных объектов могут иметь одинаковые имена, но различное содержание. Это достигается переопределением родительского метода в классе-потомке. В результате родитель и потомок ведут себя по-разному. При этом обращение к одноименным методам различных объектов выполняется аналогично.

Delphi

Управляющие структуры языка

LOGO

Условие.Операторы сравнения

LOGO

Оператор	Описание	Результат сравнения
>	Больше	True, если первый операнд больше второго, иначе False
<	Меньше	True, если первый операнд меньше второго, иначе False
=	Равно	True, если первый операнд равен второму, иначе False
<>	Не равно	True, если первый операнд не равен второму, иначе False
>=	Больше или равно	True, если первый операнд больше или равен второму, иначе False
<=	Меньше или равно	True, если первый операнд меньше или равен второму, иначе False

Например:

LOGO

(ch >= '0') **and** (ch <= '9')

(day = 7) or (day = 6)

(Forml.Editl.Text <> ' ') or (Forml.Edit2.Text <> '')

Выбор

LOGO

Выбор в точке разветвления алгоритма очередного шага программы может быть реализован при помощи инструкций if и case.

Инструкция *if*

Инструкция if позволяет выбрать один из двух возможных вариантов развития программы. Выбор осуществляется в зависимости от *выполнения* условия.

В общем виде инструкция if записывается так:

```
if условие then begin
        // здесь инструкции, которые надо выполнить,
        // если условие истинно.
    end
  else begin
        // здесь инструкции, которые надо выполнить,        // если условие ложно.
end;
```

Выбор

LOGO

Обратите внимание, что перед else (после end) точка с запятой не ставится.

Выполняется инструкция if следующим образом:

1. Вычисляется значение условия (условие — выражение логического типа, значение которого может быть равно True или False).

2. Если условие истинно (значение выражения *условие* равно True), то выполняются инструкции, следующие за словом then (между begin и end). На этом выполнение операции if заканчивается, то есть инструкции, следующие за else, не будут выполнены.

Если условие ложно (значение выражения *условие* равно False), то выполняются инструкции, следующие за словом else (между begin и end).

Выбор

LOGO

В языке Delphi есть инструкция case, которая позволяет эффективно реализовать множественный выбор. В общем виде она записывается следующим образом:

```
case Селектор of
   список1: begin  { инструкции 1 } end;
список2: begin { инструкции 2 } end;


список N : begin { инструкции N } end;
else

begin { инструкции ) end;
end;
```

Циклы

LOGO

Алгоритмы решения многих задач являются циклическими, т. е. для достижения результата определенная последовательность действий должна быть выполнена несколько раз.

Пример: для того чтобы найти фамилию человека в списке, надо проверить первую фамилию списка, затем вторую, третью и т. д. до тех пор, пока не будет найдена нужная фамилия или не будет достигнут конец списка.

В программе цикл может быть реализован при помощи инструкций **for**, **while** и **repeat**.

Инструкция for

LOGO

```
for счетчик := нач_знач to кон_знач do
    begin  // здесь инструкции, которые надо выполнить
    несколько раз
    end
```

где:

счетчик — переменная-счетчик числа повторений инструкций цикла;

нач_знач-- выражение, определяющее начальное значение счетчика циклов;

кон_знач — выражение, определяющее конечное значение счетчика циклов.

Инструкция while

LOGO

while *условие* **do**

begin // *здесь инструкции, которые надо выполнить несколько раз*

end

где условие — выражение логического типа, определяющее условие выполнения инструкций цикла.

Инструкция repeat

LOGO

repeat

// инструкции

until *условие*

где *условие* — выражение логического типа,

определяющее условие завершения цикла.

Форма

LOGO

Работа над *новым проектом,* так в Delphi называется разрабатываемое приложение, начинается с создания стартовой формы. Так на этапе разработки программы называют диалоговые окна.

Свойства формы определяют ее внешний вид: размер, положение на экране, текст заголовка, вид рамки.

Для просмотра и изменения значений свойств формы и ее компонентов используется окно **Object Inspector.** В верхней части окна **Object Inspector** указано имя объекта, значения свойств которого отображается в данный момент. В левой колонке вкладки **Properties** (Свойства) перечислены свойства объекта, а в правой — указаны их значения.

Свойства формы

LOGO

Свойство	Описание
Name	Имя формы. В программе имя формы используется для управления формой и доступа к компонентам формы
Caption	Текст заголовка
Width	Ширина формы
Height	Высота формы
Top	Расстояние от верхней границы формы до верхней границы экрана
Left	Расстояние от левой границы формы до левой границы экрана

Свойства формы

LOGO

Свойство	Описание
BorderStyle	Вид границы. Граница может быть обычной (bsSizeable), тонкой (bsSingle) или отсутствовать (bsNone). Если у окна обычная граница, то во время работы программы пользователь может при помощи мыши изменить размер окна. Изменить размер окна с тонкой границей нельзя. Если граница отсутствует, то на экран во время работы программы будет выведено окно без заголовка. Положение и размер такого окна во время работы программы изменить нельзя

Свойства формы

LOGO

Свойство	Описание
BorderIcons	Кнопки управления окном. Значение свойства определяет, какие кнопки управления окном будут доступны пользователю во время работы программы. Значение свойства задается путем присвоения значений уточняющим свойствам biSystemMenu, biMinimaze, biMaximaze и biHelp. Свойство biSystemMenu определяет доступность кнопки Свернуть и кнопки системного меню, biMinimaze— кнопки Свернуть, biMaximaze— кнопки Развернуть, biHelp — кнопки вывода справочной информации

Свойства формы

LOGO

Icon	Значок в заголовке диалогового окна, обозначающий кнопку вывода системного меню
Color	Цвет фона. Цвет можно задать, указав название цвета или привязку к текущей цветовой схеме операционной системы. Во втором случае цвет определяется текущей цветовой схемой, выбранным компонентом привязки и меняется при изменении цветовой схемы операционной системы

Свойства формы

LOGO

Font	Шрифт. Шрифт, используемый "по умолчанию" компонентами, находящимися на поверхности формы. Изменение свойства Font формы приводит к автоматическому изменению свойства Font компонента, располагающегося на поверхности формы. То есть компоненты наследуют свойство Font от формы (имеется возможность запретить наследование)

Свойства компонента Edit

LOGO

Свойство	Описание
Name	Имя компонента. Используется в программе для доступа к компоненту и его свойствам, в частности — для доступа к тексту, введенному в поле редактирования
Text	Текст, находящийся в поле ввода и редактирования
Left	Расстояние от левой границы компонента до левой границы формы
Top	Расстояние от верхней границы компонента до верхней границы формы

Свойства компонента Edit

LOGO

Height	Высота поля
Width	Ширина поля
Font	Шрифт, используемый для отображения вводимого текста
ParentFont	Признак наследования компонентом характеристик шрифта формы, на которой находится компонент. Если значение свойства равно True, то при изменении свойства Font формы автоматически меняется значение свойства Font компонента

Свойства компонента Label

LOGO

Свойство	Описание
Name	Имя компонента. Используется в программе для доступа к компоненту и его свойствам
Caption	Отображаемый текст
Font	Шрифт, используемый для отображения текста
ParentFont	Признак наследования компонентом характеристик шрифта формы, на которой находится компонент. Если значение свойства равно True, текст выводится шрифтом, установленным для формы

Свойства компонента Label

LOGO

AutoSize	Признак того, что размер поля определяется его содержимым
Left	Расстояние от левой границы поля вывода до левой границы формы
Top	Расстояние от верхней границы поля вывода до верхней границы формы
Height	Высота поля вывода
Width	Ширина поля вывода
Wordwrap	Признак того, что слова, которые не помещаются в текущей строке, автоматически переносятся на следующую строку

Свойства компонента Button

LOGO

Свойство	Описание
Name	Имя компонента. Используется в программе для доступа к компоненту и его свойствам
Caption	Текст на кнопке
Enabled	Признак доступности кнопки. Кнопка доступна, если значение свойства равно True, и недоступна, если значение свойства равно False
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

Свойства компонента ListBox

LOGO

Свойство	Определяет
Name	Имя компонента. В программе используется для доступа к свойствам компонента
Items	Элементы списка
Itemindex	Номер выбранного элемента списка. Номер первого элемента списка равен нулю
Left	Расстояние от левой границы списка до левой границы формы

Свойства компонента ListBox

LOGO

Top	Расстояние от верхней границы списка до верхней границы формы
Height	Высоту поля списка
Width	Ширину поля списка
Font	Шрифт, используемый для отображения элементов списка
Parent-Font	Признак наследования свойств шрифта родительской формы

StringGrid

LOGO

Для ввода массива удобно использовать компонент StringGrid. Значок компонента StringGrid находится на вкладке **Additional .**

Компонент StringGrid представляет собой таблицу, ячейки которой содержат строки символов.

Свойства компонента StringGrid

LOGO

Свойство	Определяет
ColCount	Количество колонок таблицы
RowCount	Количество строк таблицы
Cells	Соответствующий таблице двумерный массив. Ячейка таблицы, находящаяся на пересечении столбца номер col и строки номер row определяется элементом cells [col, row]
FixedCols	Количество зафиксированных слева колонок таблицы. Зафиксированные колонки выделяются цветом и при горизонтальной прокрутке таблицы остаются на месте

Свойства компонента StringGrid

LOGO

FixedRows	Количество зафиксированных сверху строк таблицы. Зафиксированные строки выделяются цветом и при вертикальной прокрутке таблицы остаются на месте
Options . goEditing	Признак допустимости редактирования содержимого ячеек таблицы. True — редактирование разрешено, False — запрещено
Options . goTab	Разрешает (True) или запрещает (False) использование клавиши <TaЬ> для перемещения курсора в следующую ячейку таблицы

Свойства компонента StringGrid

LOGO

Options . GoAlways-ShowEditor	Признак нахождения компонента в режиме редактирования. Если значение свойства False, то для того, чтобы в ячейке появился курсор, надо начать набирать текст, нажать клавишу <F2> или сделать щелчок мышью
DefaultColWidth	Ширину колонок таблицы
DefaultRowHeight	Высоту строк таблицы
GridLineWidth	Ширину линий, ограничивающих ячейки таблицы

Свойства компонента StringGrid

LOGO

Left	Расстояние от левой границы поля таблицы до левой границы формы
Top	Расстояние от верхней границы поля таблицы до верхней границы формы
Height	Высоту поля таблицы
Width	Ширину поля таблицы
Font	Шрифт, используемый для отображения содержимого ячеек таблицы
ParentFont	Признак наследования характеристик шрифта формы

Memo

LOGO

В некоторых случаях для ввода массива можно использовать компонент Memo. Компонент Memo позволяет вводить текст, состоящий из достаточно большого количества строк, поэтому его удобно использовать для ввода символьного массива. Значок компонента находится на вкладке **Standard.**

Свойства компонента Memo

LOGO

Свойство	Определяет
Name	Имя компонента. Используется в программе для доступа к свойствам компонента
Text	Текст, находящийся в поле Memo. Рассматривается как единое целое
Lines	Текст, находящийся в поле Memo. Рассматривается как совокупность строк. Доступ к строке осуществляется по номеру
Lines .Count	Количество строк текста в поле Memo
Left	Расстояние от левой границы поля до левой границы формы

Свойства компонента Memo

LOGO

Top	Расстояние от верхней границы поля до верхней границы формы
Height	Высоту поля
Width	Ширину поля
Font	Шрифт, используемый для отображения вводимого текста
ParentFont	Признак наследования свойств шрифта родительской формы

Свойства компонента CheckBox

LOGO

Свойство	Определяет
Name	Имя компонента. Используется в программе для доступа к свойствам компонента
Caption	Текст, поясняющий назначение флажка
Checked	Состояние, внешний вид флажка: если флажок установлен (в квадратике есть "галочка"), то checked = TRUE; если флажок сброшен (нет "галочки"), то Checked=FALSE
State	Состояние флажка. В отличие от свойства Checked, позволяет различать установленное, сброшенное и промежуточное состояния. Состояние флажка определяют константы: cbChecked (установлен); cbGrayed (серый, неопределенное состояние); cbUnChecked (сброшен)

Свойства компонента CheckBox

LOGO

AllowGrayed	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	Высоту поля вывода поясняющего текста

Свойства компонента CheckBox

LOGO

Width	Ширину поля вывода поясняющего текста
Font	Шрифт, используемый для отображения поясняющего текста
ParentFont	Признак наследования характеристик шрифта родительской формы

ComboBox

LOGO

Компонент ComboBox, значок которого находится на вкладке **Standard**, дает возможность ввести данные либо непосредственно в поле ввода-редактирования, либо путем выбора из списка, который появляется в результате щелчка на кнопке раскрывающегося списка.

Свойства компонента ComboBox

LOGO

Свойство	Определяет
Name	Имя компонента. Используется для доступа к свойствам компонента
Text	Текст, находящийся в поле ввода-редактирования
Items	Элементы раскрывающегося списка
DropDownCount	Количество отображаемых элементов в раскрытом списке
Left	Расстояние от левой границы компонента до левой границы формы
Top	Расстояние от верхней границы компонента до верхней границы формы

Свойства компонента ComboBox

LOGO

Height	Высоту компонента (поля ввода-редактирования)
Width	Ширину компонента
Font	Шрифт, используемый для отображения элементов списка
ParentFont	Признак наследования свойств шрифта родительской формы

image

LOGO

Наиболее просто вывести иллюстрацию, которая находится в файле с расширением bmp, jpg или ico, можно при помощи компонента image, значок которого находится на вкладке **Additional** палитры .

Свойства компонента image

LOGO

Свойство	Определяет
Picture	Иллюстрацию, которая отображается в поле компонента
Width, Height	Размер компонента. Если размер компонента меньше размера иллюстрации, и значение свойств AutoSize и strech равно False, то отображается часть иллюстрации
AutoSize	Признак автоматического изменения размера компонента в соответствии с реальным размером иллюстрации

Свойства компонента image

LOGO

Strech	Признак автоматического масштабирования иллюстрации в соответствии с реальным размером компонента. Чтобы было выполнено масштабирование, значение свойства AutoSize должно быть False
Visible	Отображается ли компонент, и, соответственно, иллюстрация, на поверхности формы

Timer

LOGO

Компонент Timer генерирует событие OnTimer. Период возникновения события OnTimer измеряется в миллисекундах и определяется значением свойства Interval. Следует обратить внимание на свойство Enabled. Оно дает возможность программе "запустить" или "остановить" таймер. Если значение свойства Enabled равно False, то событие OnTimer не возникает.

Свойства компонента Timer

LOGO

Свойство	Определяет
Name Interval	Имя компонента. Используется для доступа к компоненту Период генерации события OnTimer. Задается в миллисекундах
Enabled	Разрешение работы. Разрешает (значение True) или запрещает (значение False) генерацию события OnTimer

Animate

LOGO

Компонент Animate, значок которого находится на вкладке **Win32**, позволяет воспроизводить простую анимацию, кадры которой находятся в AVI-файле.

Свойства компонента Animate

LOGO

Свойство	Определяет
Name	Имя компонента. Используется для доступа к свойствам компонента и управлением его поведением
FileName	Имя AVI-файла в котором находится анимация, отображаемая при помощи компонента
StartFrame	Номер кадра, с которого начинается отображение анимации
stopFrame	Номер кадра, на котором заканчивается отображение анимации

Свойства компонента Animate

LOGO

Activate	Признак активизации процесса отображения кадров анимации
Color	Цвет фона компонента (цвет "экрана"), на котором воспроизводится анимация
Transparent	Режим использования "прозрачного" цвета при отображении анимации
Repetitions	Количество повторов отображения анимации

Компонент MediaPlayer

LOGO

Компонент MediaPlayer, значок которого находится на вкладке **System**, позволяет воспроизводить видеоролики, звук и сопровождаемую звуком анимацию.

В результате добавления к форме компонента MediaPlayer на форме появляется группа кнопок, подобных тем, которые можно видеть на обычном аудио- или видеоплеере.

Кнопки компонента MediaPlayer

LOGO

Кнопка	Обозначение	Действие
Воспроизведение	btPlay	Воспроизведение звука или видео
Пауза	btPause	Приостановка воспроизведения
Стоп	btStop	Остановка воспроизведения
Следующий	btNext	Переход к следующему кадру
Предыдущий	btPrev	Переход к предыдущему кадру
Шаг	btStep	Переход к следующему звуковому фрагменту, например, к следующей песне на CD
Назад	btBack	Переход к предыдущему звуковому фрагменту, например, к предыдущей песне на CD
Запись	btRecord	Запись
Открыть/Закрыть	btEject	Открытие или закрытие CD-дисковода компьютера

Свойства компонента MediaPiayer

LOGO

Свойство	Описание
Name Device	Имя компонента. Используется для доступа к свойствам компонента и управлением работой плеера
Type	Тип устройства. Определяет конкретное устройство, которое представляет собой компонент MediaPiayer. Тип устройства задается именованной константой: dtAutoSelect — тип устройства определяется автоматически; dtVaweAudio — проигрыватель звука; dtAVivideo — видеопроигрыватель; dtCDAudio — CD-проигрыватель

Свойства компонента MediaPiayer

LOGO

FileName	Имя файла, в котором находится воспроизводимый звуковой фрагмент или видеоролик
AutoOpen	Признак автоматического открытия сразу после запуска программы, файла видеоролика или звукового фрагмента
Displa	Определяет компонент, на поверхности которого воспроизводится видеоролик (обычно в качестве экрана для отображения видео используют компонент Panel)
VisibleButtons	Составное свойство. Определяет видимые кнопки компонента. Позволяет сделать невидимыми некоторые кнопки

MainMenu

LOGO

Главное меню программы. Значок компонента MainMenu находится на вкладке **Standard**.

Свойства объекта TMenuItem

LOGO

Свойство	Определяет
Name	Имя элемента меню. Используется для доступа к свойствам
Caption	Название меню или команды
Bitmap	Значок, который отображается слева от названия элемента меню
Enabled	Признак доступности элемента меню. Если значение свойства равно False, то элемент меню недоступен, при этом название меню отображается серым цветом

События

LOGO

Событие	Происходит
OnClick	При щелчке кнопкой мыши
OnDblClick	При двойном щелчке кнопкой мыши
OnMouseDown	При нажатии кнопки мыши
OnMouseUp	При отпускании кнопки мыши
OnMouseMove	При перемещении мыши
OnKeyPress	При нажатии клавиши клавиатуры
OnKeyDown	При нажатии клавиши клавиатуры. События OnKeyDown и OnKeyPress — это чередующиеся, повторяющиеся события, которые происходят до тех пор, пока не будет отпущена удерживаемая клавиша (в этот момент происходит событие OnKeyUp)

События

LOGO

OnKeyUp	При отпускании нажатой клавиши клавиатуры
OnCreate	При создании объекта (формы, элемента управления). Процедура обработки этого события обычно используется для инициализации переменных, выполнения подготовительных действий
OnPaint	При появлении окна на экране в начале работы программы, после появления части окна, которая, например, была закрыта другим окном, и в других случаях
OnEnter	При получении элементом управления фокуса
OnExit	При потере элементом управления фокуса