Цикл презентаций «ООП на Delphi» посвящен объектно – ориентированному программированию с использованием одной из самых распространенных систем быстрой разработки приложений – Delphi

Используя данный учебный курс, можно самостоятельно овладеть основами объектно – ориентированного программирования на Delphi. Для расширения Ваших знаний к курсу приложен ряд учебных пособий и справочников по Delphi

Цикл содержит 13 презентаций:

ООП на Delphi – 1: Знакомство с системой программирования Borland Delphi. Объекты (компоненты) и их свойства и методы

ООП на Delphi – 2: Первая программа на Delphi, сохранение и компиляция

ООП на Delphi – 3: Программное изменение свойств объектов

ООП на Delphi – 4: Условия в Delphi. Создание простого теста

ООП на Delphi – 5: Элементы ввода и вывода информации. Обработка исключений

ООП на Delphi – 6: Заставка программы и элемент таймер

ООП на Delphi – 7: Программируем свою игрушку

ООП на Delphi – 8: Меню программы, диалоги

ООП на Delphi – 9: Создаем свой текстовый редактор

ООП на Delphi – 10: Базы данных на Delphi

ООП на Delphi – 11: Калькулятор на Delphi. Обработка исключительных ситуаций

ООП на Delphi – 12: Создаем тестирующую систему

ООП на Delphi – 13: Графика на Delphi

Delphi использует язык программирования Объект Паскаль, поэтому лучше сначала изучить обычный Паскаль и поработать в ТурбоПаскале, а затем и переходить к Delphi – перейти будет очень просто, т.к синтаксис языка остается неизменным.

Изучение ООП на Delphi желательно проводить в старших профильных классах – количество часов, отводимое на информатику там вполне достаточно для освоения основ ООП на Delphi

Объектно – ориентированное программирование на

Borland® DELPHI - 6

DELPHI - 6

На этом уроке:

Мы научимся создавать программы, имеющие несколько форм, заставку, появляющуюся перед созданием основной формы программы, а также познакомимся с применением компонента таймер

Вопросы:

1. Приложения, содержащие несколько форм
2. Приложения, использующие компонент таймер
3. Создание заставки программы

Borland®

1. Приложения, содержащие несколько форм

В подавляющем большинстве приложения содержат не одну, а несколько форм, которые могут появляться при возникновении определенных событий. Например, приложения с базами данных часто содержат десятки различных форм.

Наша задача – научиться создавать приложения из нескольких взаимосвязанных форм

Для начала давайте создадим приложение из 3 связанных форм. Рассмотрим процесс создания приложения по шагам:

ШАГ 1

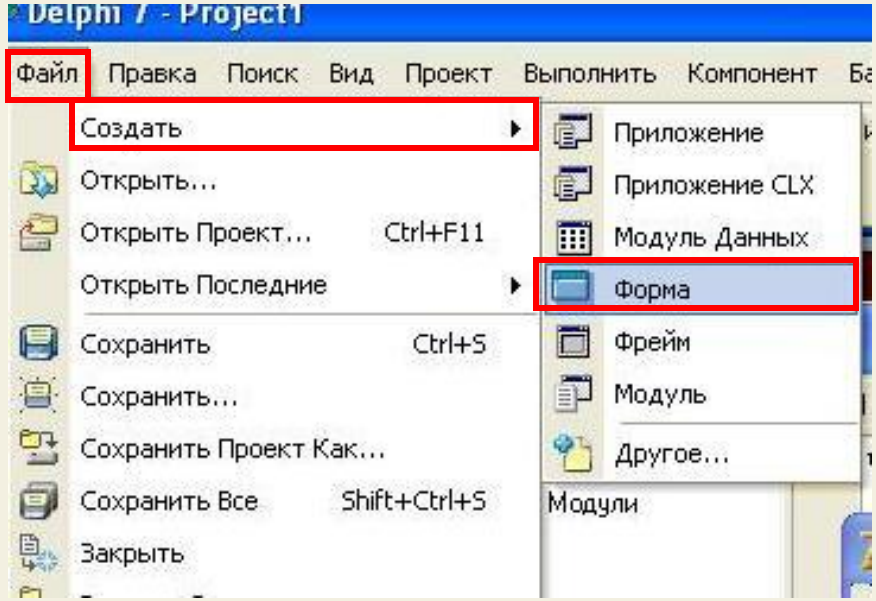
Запускаем Delphi и у нас автоматически создается форма1 (Form1). В свойстве Caption формы назовем ее - «ГЛАВНАЯ». Delphi автоматически делает эту форму главной или стартовой – она первой открывается при запуске приложения (хотя мы можем это изменить)

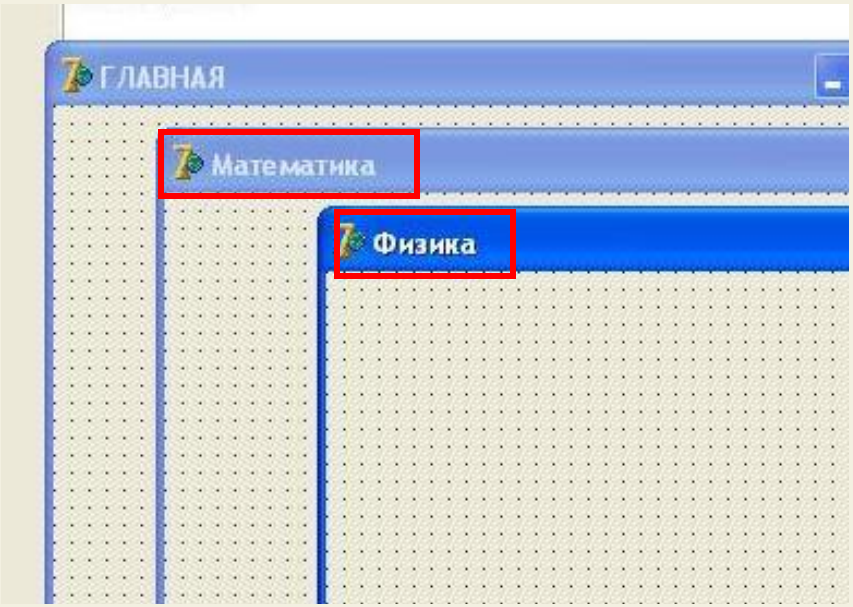
ШАГ 2

Добавим к нашему приложению еще 2 формы, которые назовем «Математика» и «Физика»

Для этого зайдем в меню Файл –> Создать -> Форма

Создадим форму2 (Form2), которую назовем «Математика». Аналогично создадим и форму «Физика»

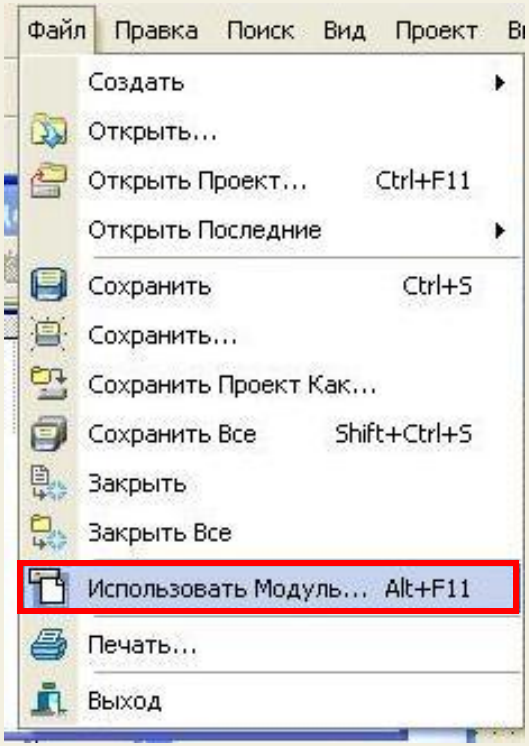


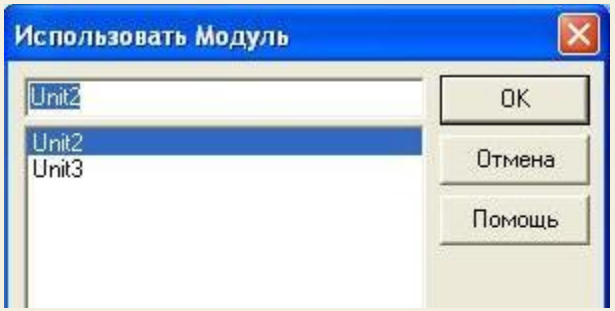


ШАГ 3

Сейчас надо «познакомить» все эти формы. Т.е. главная форма должна «знать» о существовании форм «Математика» и «Физика», а те, в свою очередь, должны знать о существовании ГЛАВНОЙ и друг о друге. Это нужно для того, чтобы мы могли из одной формы вызвать другую

Сделаем активной форму «ГЛАВНАЯ» (просто щелкнем по ней мышкой), зайдем в меню Файл -> Использовать модуль, где укажем на использование формы «Математика» (Unit2). Затем снова зайдем и укажем модуль Unit3 (форма «Физика»)





Сейчас форма «ГЛАВНАЯ» знает о существовании форм «Математика» и «Физика» и может к ним обращаться

Аналогично сделайте активной форму «Математика» и свяжите ее с модулями Unit1 и Unit3. (С формами «ГЛАВНАЯ» и «Физика»)

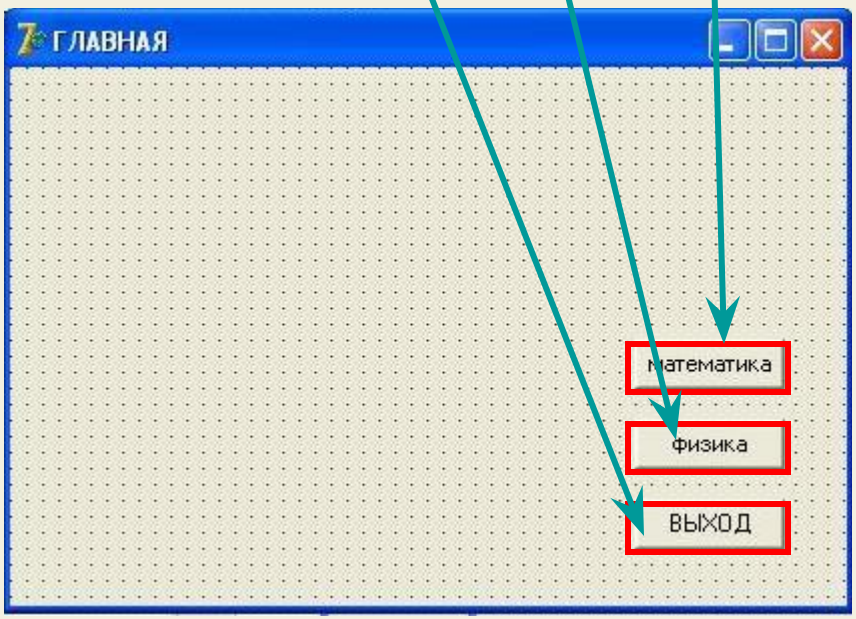
То же самое сделайте для формы «Физика»

ШАГ 4

Итак, мы «познакомили» формы, а сейчас их свяжем, т.е. по событию в одной форме появляется другая

Разместим на форме «ГЛАВНАЯ» 3 кнопки:

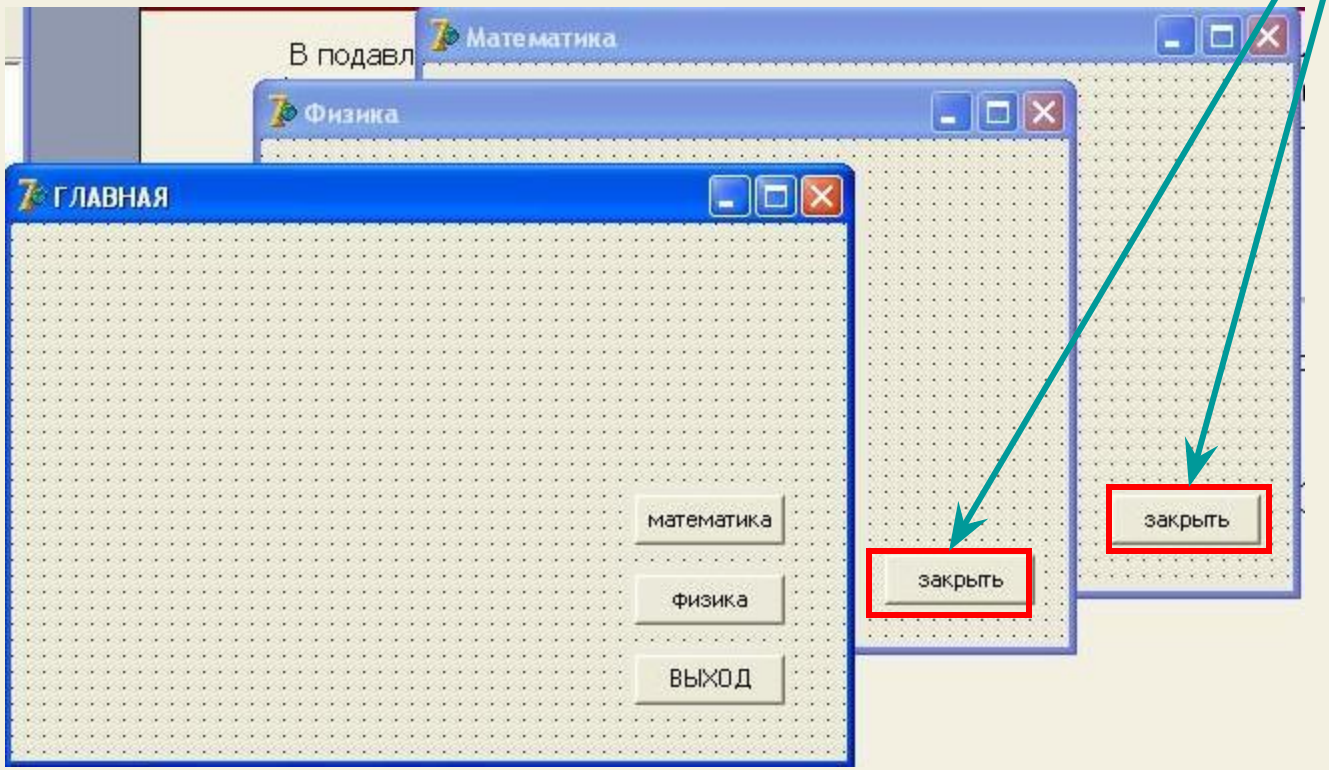
- Математика – для открытия формы «Математика»
- Физика – для открытия формы «Физика»
- ВЫХОД – для выхода из приложения



ШАГ 4

Итак, мы «познакомили» формы, а сейчас их свяжем, т.е. по событию в одной форме появляется другая

Аналогично на формах «Математика» и «Физика» разместим кнопки «Закрыть» для закрытия этих форм



ШАГ 4

Итак, мы «познакомили» формы, а сейчас их свяжем, т.е. по событию в одной форме появляется другая

Сделаем двойной щелчок кнопке «МАТЕМАТИКА» на форме «Главная» для перехода в редактор

Unit1 | Unit2 | Unit3

```
{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
form2.ShowModal
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
form3.Show
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
close
end;
```

В процедуре нажатия на кнопку «Математика» запишем:

Form2.showModal

ШАГ 4

Итак, мы «познакомили» формы, а сейчас их свяжем, т.е. по событию в одной форме появляется другая

Сделаем двойной щелчок кнопке «Физика» на форме «ГЛАВНАЯ» для перехода в редактор

Unit1 | Unit2 | Unit3

```
{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
form2.ShowModal
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
form3.Show
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
close
end;
```

В процедуре нажатия на кнопку «Физика» запишем:

Form3.show

ШАГ 4

Итак, мы «познакомили» формы, а сейчас их свяжем, т.е. по событию в одной форме появляется другая

Сделаем двойной щелчок кнопке «ВЫХОД» на форме «ГЛАВНАЯ» для перехода в редактор

Unit1 | Unit2 | Unit3

```
{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
form2.ShowModal
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
form3.Show
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
close
end;
```

В процедуре нажатия на кнопку «ВЫХОД» запишем:

close

ШАГ 4

Итак, мы «познакомили» формы, а сейчас их свяжем, т.е. по событию в одной форме появляется другая

Unit1 | Unit2 | Unit3

```
{$R *.dfm}

procedure TForm1.
begin
form2.ShowModal
end;

procedure TForm1.
begin
form3.Show
end;

procedure TForm1.
begin
close
end;
```

Разберемся с кодом:

Для вызова формы на экран (показа ее) в Delphi существуют метод **Show** (**Show Modal**)

Метод **Show Modal** вызывает модальное окно (окно, которое полностью берет на себя управление программой и пока мы его не закроем, мы не сможем выполнять какие – либо действия в другом окне)

(Окно формы «Математика» у нас является модальным)

Примером модальных окон являются системные сообщения Windows

Метод **Show** выводит обычное окно, при этом мы можем что-то делать в другом окне, не закрывая первого

(Окно формы «Физика» у нас не является модальным – оно обычное)

ШАГ 4

Думаю, не требуется объяснений для написания кода кнопок «Закрыть» на формах «Математика» и «Физика»

```
procedure TForm2.Button1Click
begin
close
end;

end.
```

ШАГ 5

И последний шаг: сохраняем проект и компилируем его (посмотрите внимательно, что у Вас сохранилось: кроме файла проекта сохранились файлы каждого модуля (каждой формы)

Все.

Можно запустить приложение и попробовать его работу (и сравните поведение окон форм «Математика» и «Физика» - одно из них модальное, а другое – нет)

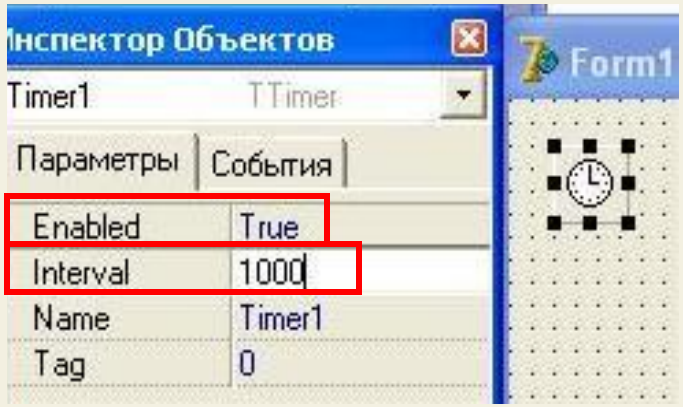
Запустить -> 

Итак, мы научились создавать приложения из многих взаимосвязанных форм (конечно же в самом простом варианте их использования)

Borland®

2. Приложения, содержащие компонент таймер

Сначала познакомимся с компонентом Таймер (Timer) и его свойствами.



Компонент Таймер является невидимым, т.е при работе приложения его не видно на форме, но мы также можем управлять им – запускать или останавливать эти «часики». И по этим часам через определенные, заданные нами промежутки времени, могут происходить различные события

Основные свойства таймера:

1. **Enabled (доступность).**

   Если Enabled имеет значение True, то таймер запущен (часы идут)

   Если Enabled имеет значение False, таймер остановлен

2. **Interval** (промежуток времени «тиканий» часов – через каждый такой промежуток может происходить какое – то заданное нами событие).

   Интервал измеряется в миллисекундах (Например, если значение Interval=1000, значит период срабатываний таймера 1000 миллисекунд,т.е. 1 секунда)

А сейчас давайте рассмотрим его использование на примере создания программы – секундомера (по шагам)

Определим требования к программе:

Одноформенное приложение с прямым отсчетом времени (без обратного отсчета), кнопками ПУСК и СТОП, индикацией прошедшего промежутка времени с точностью до десятых долей секунды

☝ По мере продвижения в изучении Delphi и создания прикладных программ наши объяснения работы с компонентами и кодом будут все меньше, чтобы не повторяться и не загромождать курс.

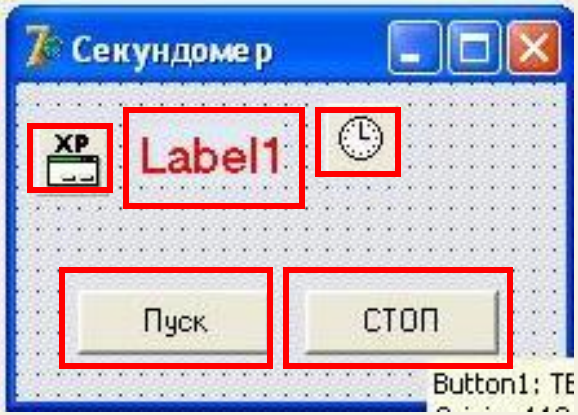
Если встречается что – то непонятное – смотрите внимательно предыдущие уроки, а также справочник А.Я.Архангельского «100 компонентов общего назначения Delphi»

ШАГ 1

Запускаем Delphi и на форме размещаем необходимые компоненты. Делаем соответствующие надписи

Label, в котором будет динамически отображаться ход времени

Манифест XP для украшения приложения в стиле Windows XP



Timer, который будет управлять ходом времени (Свойству Interval в инспекторе объектов установим значение 100 (0,1 сек))

Кнопка ПУСК, которая будет запускать таймер и обнулять показания Label -а

Кнопка СТОП, которая будет останавливать таймер

Создаем обработчики событий

ШАГ 2 Событие создания формы

Первое событие – это создание формы (On Create), которое происходит каждый раз при запуске приложения

При запуске приложения в Label –е должен быть ноль, переменная k, значение которой будет отображаться в ходе подсчета в Label-е – тоже ноль, а таймер должен «стоять»

```
procedure TForm1.FormCreate(Sender: TObj
begin
label1.caption:='0';
k:=0;
timer1.Enabled:=false;
end;
```

Пишем ноль в Label-е

Присваиваем ноль k

Таймер останавливаем

Создаем обработчики событий

ШАГ 3 Событие нажатия на кнопку ПУСК

При нажатии на кнопку ПУСК таймер должен запускаться, а показания Label-а и k обнуляться, чтобы счет шел сначала, а не нарастающим итогом

```
procedure TForm1.Button2Click(S
begin
 k:=0;
label1.Caption:='0';
 timer1.Enabled:=true;
```

Обнуляем значение k

В Label-е выводим ноль

Запускаем таймер

Создаем обработчики событий

ШАГ 4 Событие нажатия на кнопку СТОП

При нажатии на кнопку СТОП таймер должен остановиться

```
procedure TForm1.Button1Click(Sender: TObject);
begin
timer1.Enabled:=false;
end;
```

Останавливаем таймер

ШАГ 5 Заставим таймер считать (сделаем двойной щелчок по таймеру и запишем код)

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
 k:=k+0.1;
label1.Caption:=floattostr(k);
```

При каждом срабатывании таймера к значению k должно прибавляться 0,1 (100 мс)

При каждом срабатывании таймера в Label-е будет отображаться значение k

Создаем обработчики событий

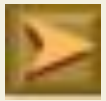
ШАГ 6

Не забудьте объявить переменную k – ее тип будет конечно real

```
var
  Form1: TForm1;
  k:real;
implementation
```

ШАГ 7

Сохраняем, компилируем и запускаем программу

Запустить -> 

Мы познакомились с таймером и научились его использовать. В следующих примерах мы также будем его применять, в частности при создании заставки программы

Borland®

3. Создание заставки программы

Во многих приложениях перед открытием главного (стартового окна программы) возникает **заставка** – окно с информацией о программе, логотипом и пр., которое обычно само исчезает через несколько секунд. Причем заставка может сопровождаться и музыкальным фрагментом

Посмотреть пример ->

Вы уже поняли, что в качестве примера мы создадим заставку для нашего секундомера, который мы только что создали

ШАГ 1

Запускаем Delphi и открываем проект с нашим секундомером, затем создаем новую форму (Файл -> Создать -> Форма) – эта форма и будет нашей заставкой

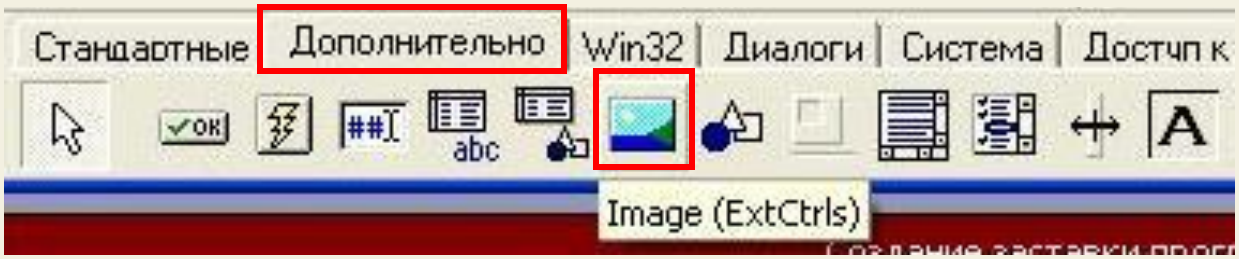
На этой форме размещаем информацию, картинки и т.д. – поработаем над дизайном

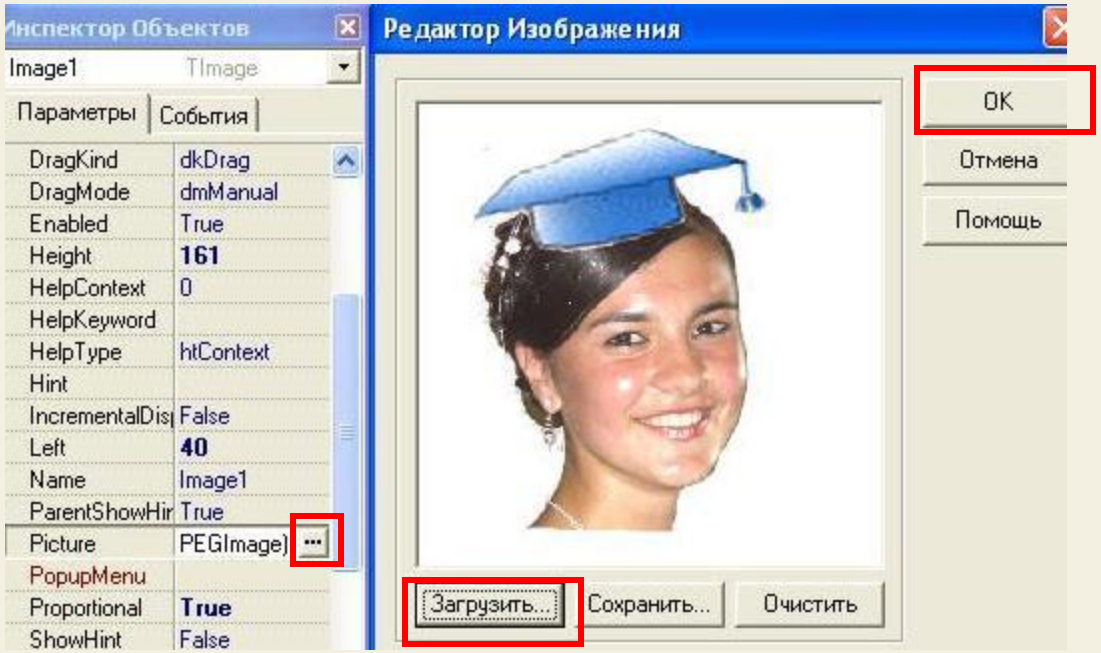
Свойству BorderStyle этой формы даем значение BsNone, чтобы у формы, как обычно бывает у заставки, не было границ

И помещаем на форму компонент таймер – он будет «показывать» нам заставку определенное нами время (Поставим интервал таймера – 3000, а Enabled =True)

Как поместить на форму картинку?

Для этого служит компонент Image, который находится на вкладке Дополнительно





Помещаем компонент на форму и раскрываем в инспекторе объектов его свойство Picture

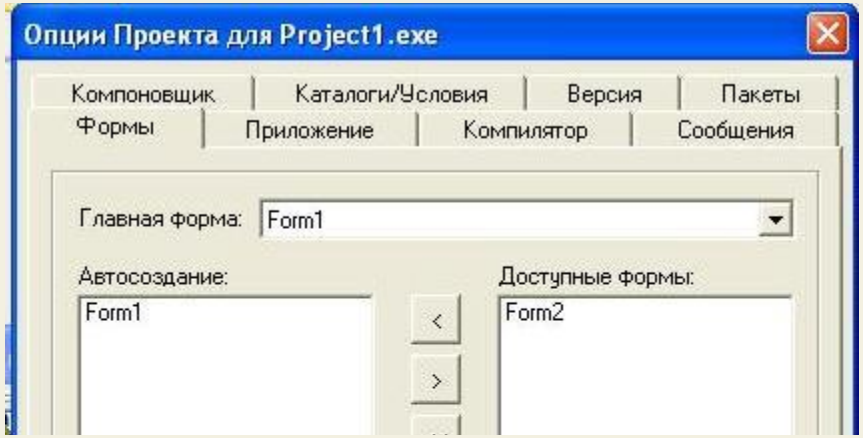
В редакторе изображения щелкаем кнопку «Загрузить» и появившемся окне загрузки изображения находим нужную картинку на диске компьютера

Осталось нажать ОК и картинка вставлена

Посмотрите в инспекторе объектов свойства компонента Image, попробуйте изменять их значения и посмотрите, к чему это приведет

ШАГ 2

Сейчас заходим в меню Delphi: Проект-> Опции и переносим форму2 (заставку) из раздела Автосоздание в раздел Доступные формы



Делаем двойной щелчок на Таймере и в обработчике события пишем:

```
procedure TForm2.Timer1Timer(Senc
begin
 Timer1.Enabled := false;
end;

end.
```

Т.е. через 3 секунды таймер сработает и сам себя выключит, а заставка исчезнет с экрана (при открытии формы Enabled мы ставили True и отсчет времени сразу пошел)

ШАГ 3

А сейчас откроем файл проекта, нажав Ctrl+F12 (и выберем Проект1), в котором вставим немного кода (выделено красным)

Что было

```
program Project1;

uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

Что станет

```
program Project1;

uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1},
  Unit2 in 'Unit2.pas' {Form2};

{$R *.res}

begin
  Application.Initialize;
  Form2 := TForm2.Create(Application);
  Form2.Show;
  Form2.Update;
  while Form2.Timer1.Enabled do
    Application.ProcessMessages;
  Application.CreateForm(TForm1, Form1);
  Form2.Hide;
  Form2.Free;
  Application.Run;
end.
```

Разбор кода проекта оставим на будущее

ШАГ 4

Последний шаг: сохраняем, компилируем и запускаем

Запустить ->

На этом урок закончен

ИТОГИ УРОКА:

На этом уроке мы научились создавать приложения, содержащие несколько форм, познакомились со свойствами и применением таймера, а также создали заставку для программы

НА СЛЕДУЮЩЕМ УРОКЕ:

ООП на Delphi – 7:

Мы попробуем создать свою игру, используя компоненты, применять которые мы уже умеем

Домнин Константин Михайлович

E – mail: kdomnin@list.ru

2006 год.