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 - 3

DELPHI - 3

На этом уроке:

Мы научимся программно изменять основные свойства объектов и создавать процедуры обработки событий на основе разбора практических примеров

Вопросы:

1. Изменение свойств объектов и (программа «Ты уважаешь информатику?»)
2. Создание процедуры обработки событий

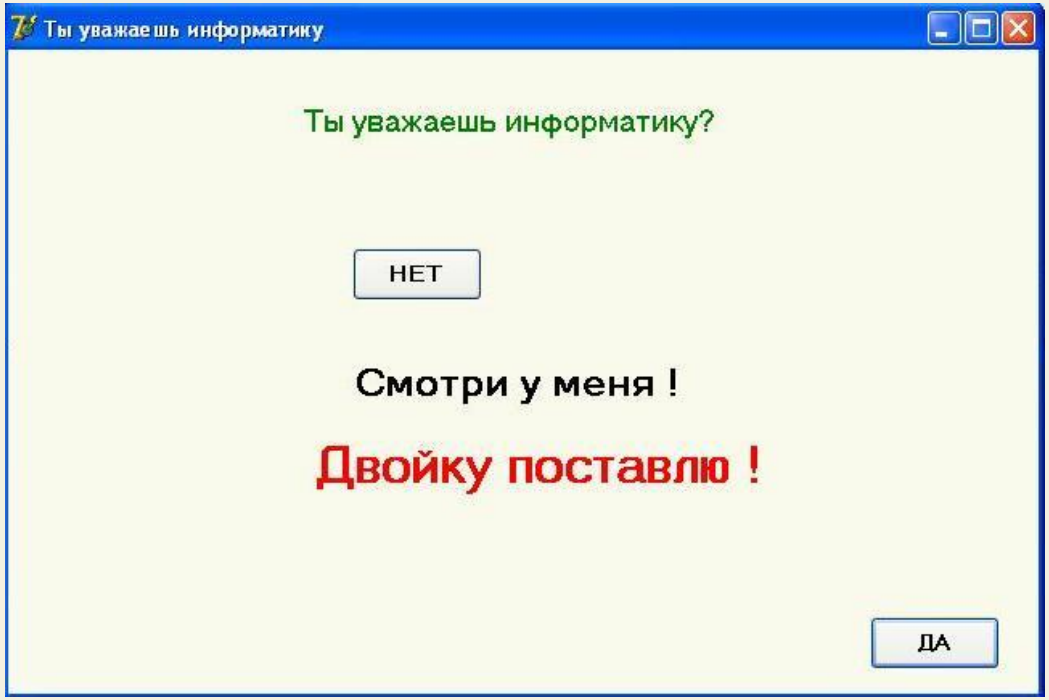
Borland®

1. Изменение свойств объектов

На этом уроке мы научимся , как в процессе работы программы изменять свойства объектов, а также создадим некоторые обработчики событий

Для начала давайте посмотрим программу, которую мы сейчас создадим. Называется она **«Ты уважаешь информатику?»**

Посмотреть ->



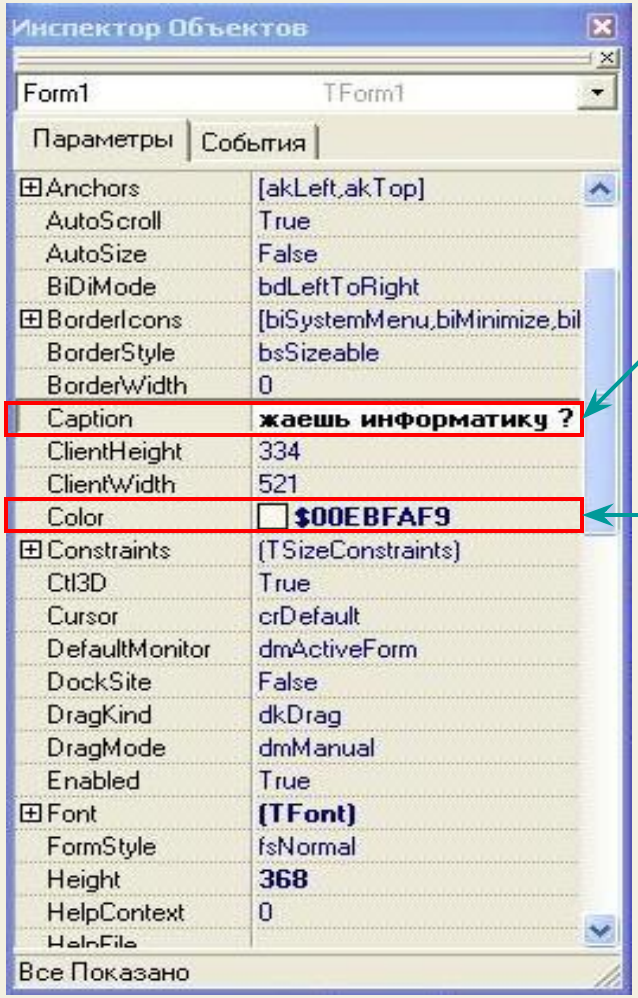
Давайте разберем, как и какие свойства объектов мы изменяли в этой программе

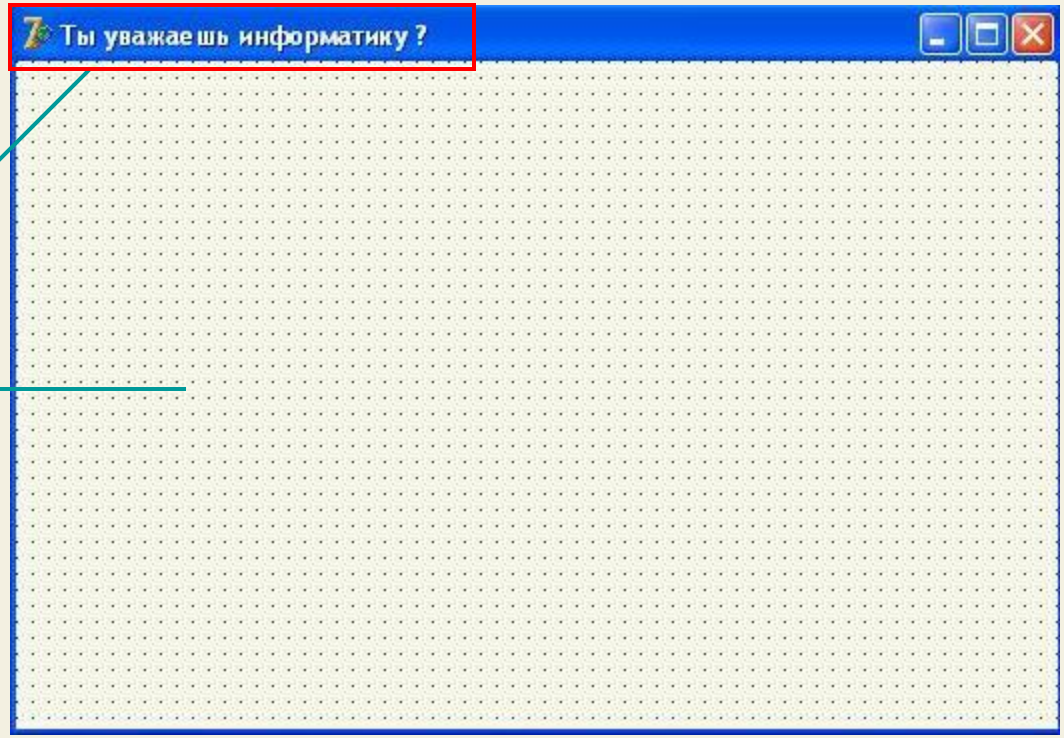
Форма программы содержит 4 компоненты Label (метка) и 3 кнопки «ДА», «НЕТ» и «ВЫХОД», причем в начале программы видна только одна надпись, а при наведении курсора на кнопку нет она должна перепрыгивать в другое место

Итак, начнем создание и разбор программы по шагам

ШАГ 1

Запускаем Delphi и с помощью инспектора объектов изменяем заголовок формы и выбираем ее цвет

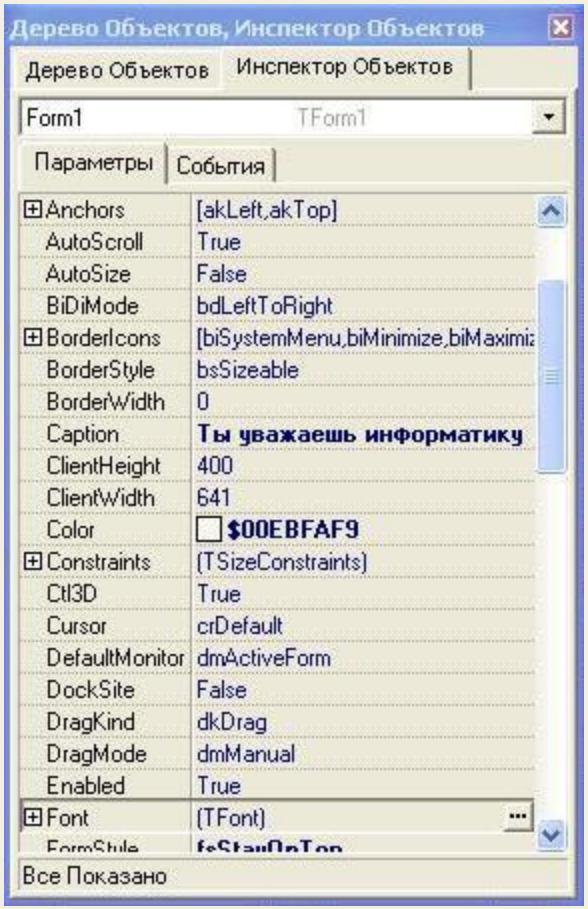


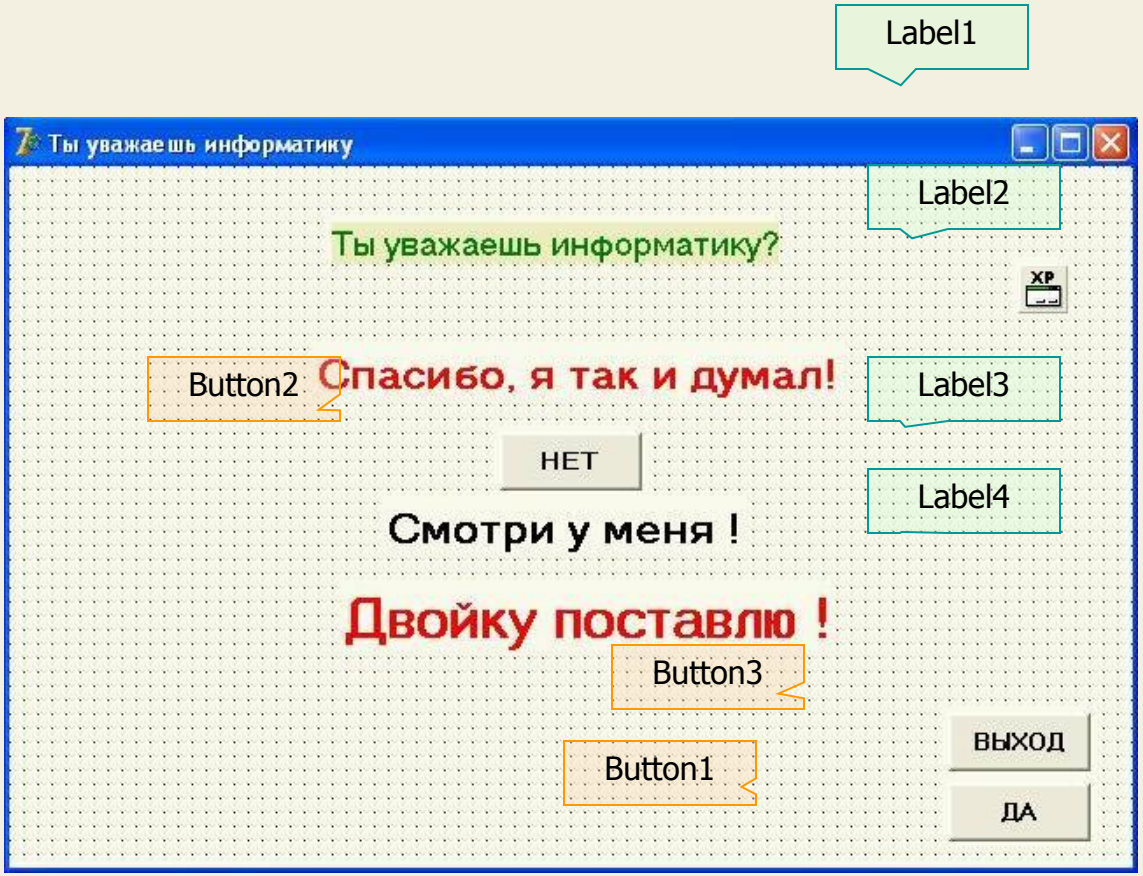


Итак, начнем создание и разбор программы по шагам

ШАГ 2

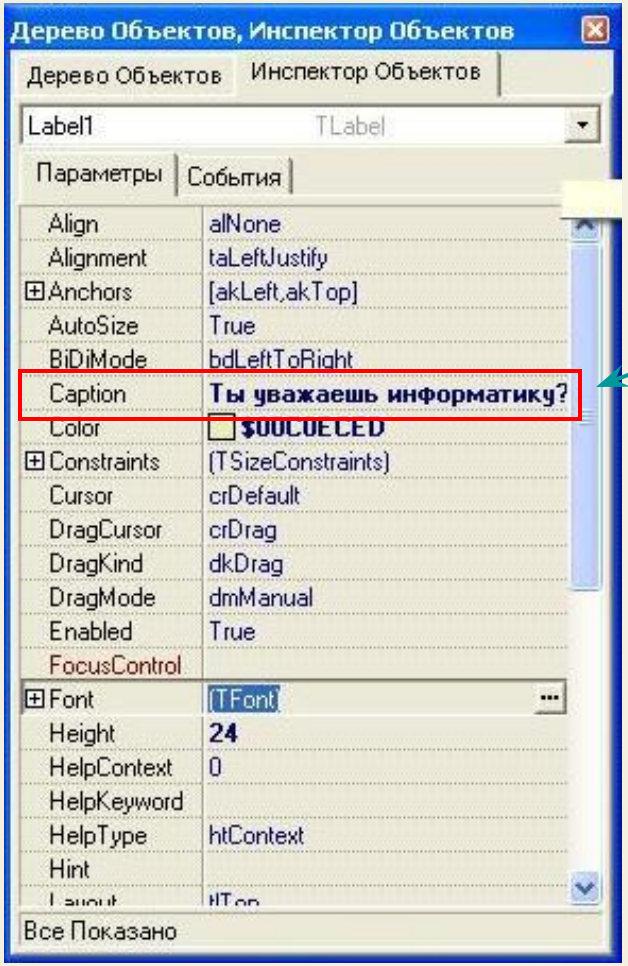
Помещаем на форму с вкладки **Стандартные** 4 компонента **Label** и 3 кнопки (**Button**)

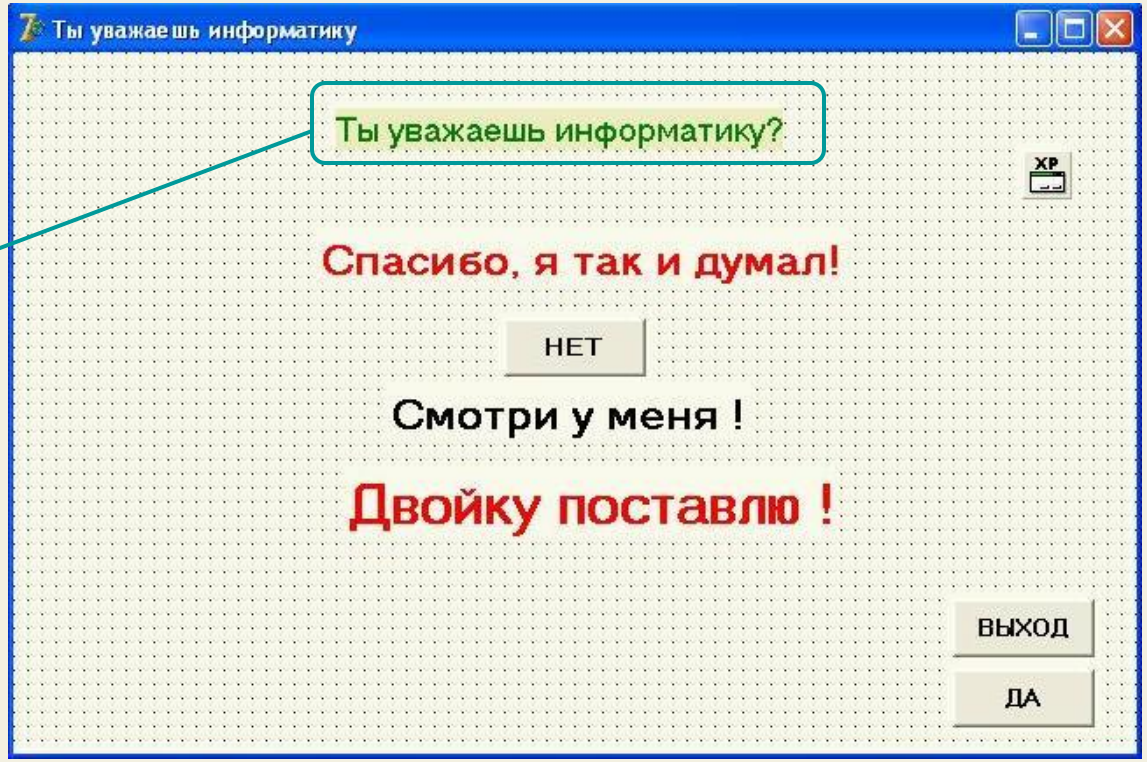




ШАГ 3

У компонента Label1 через инспектор объектов меняем свойство Caption (надпись) – «Ты уважаешь информатику ?»

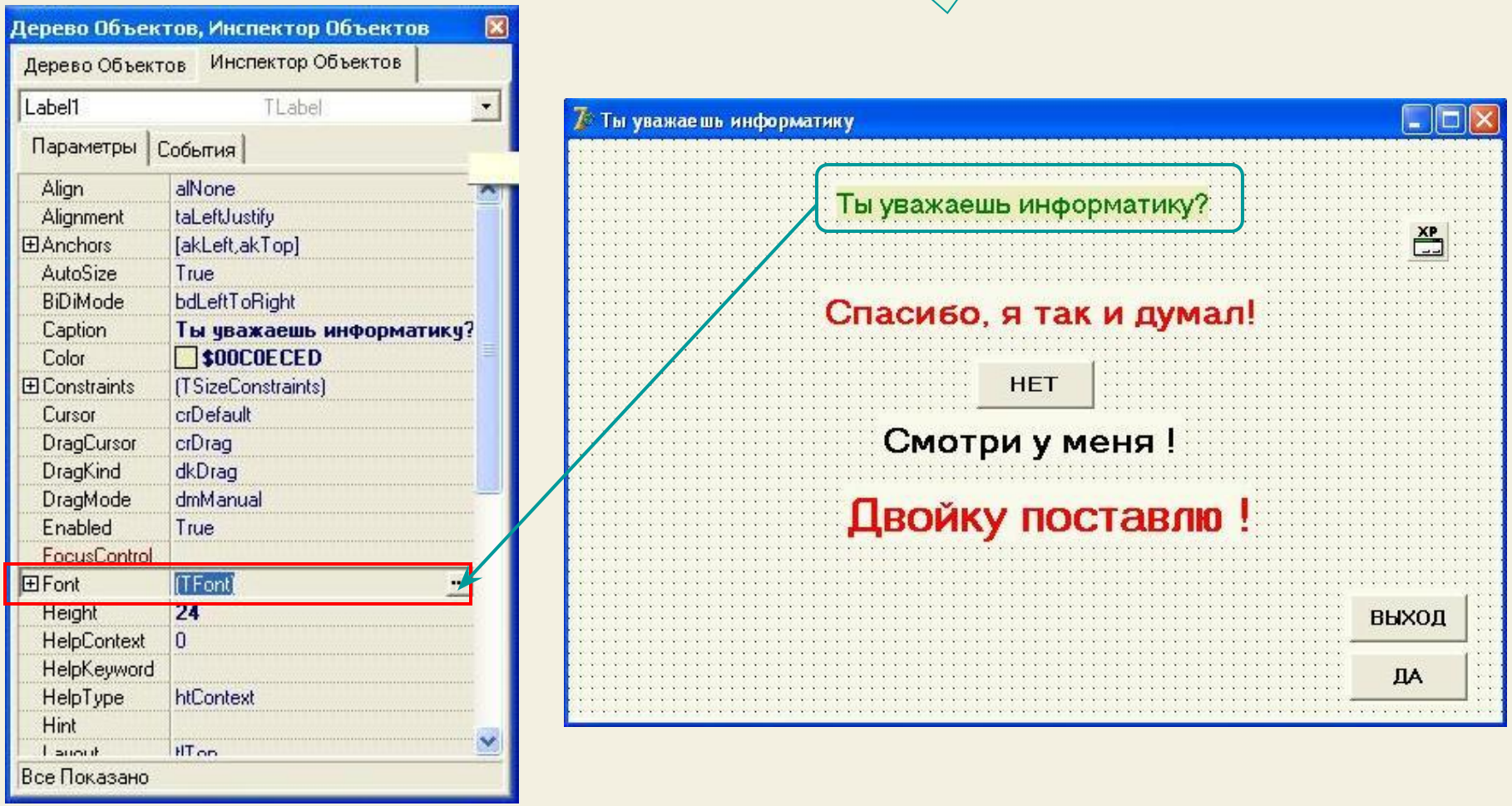




ШАГ 3

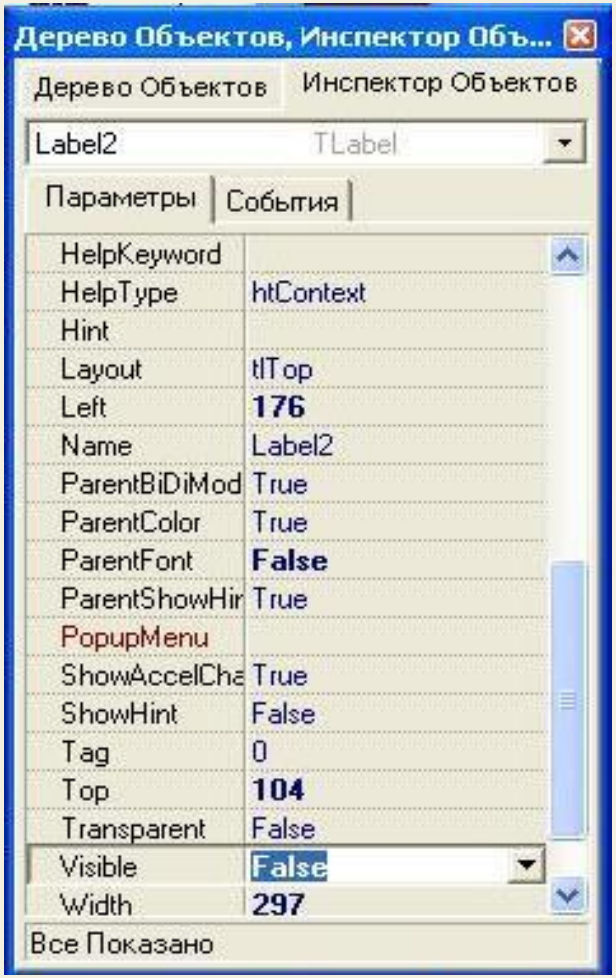
У компонента Label1 через инспектор объектов меняем свойство Caption (надпись) – «Ты уважаешь информатику ?»

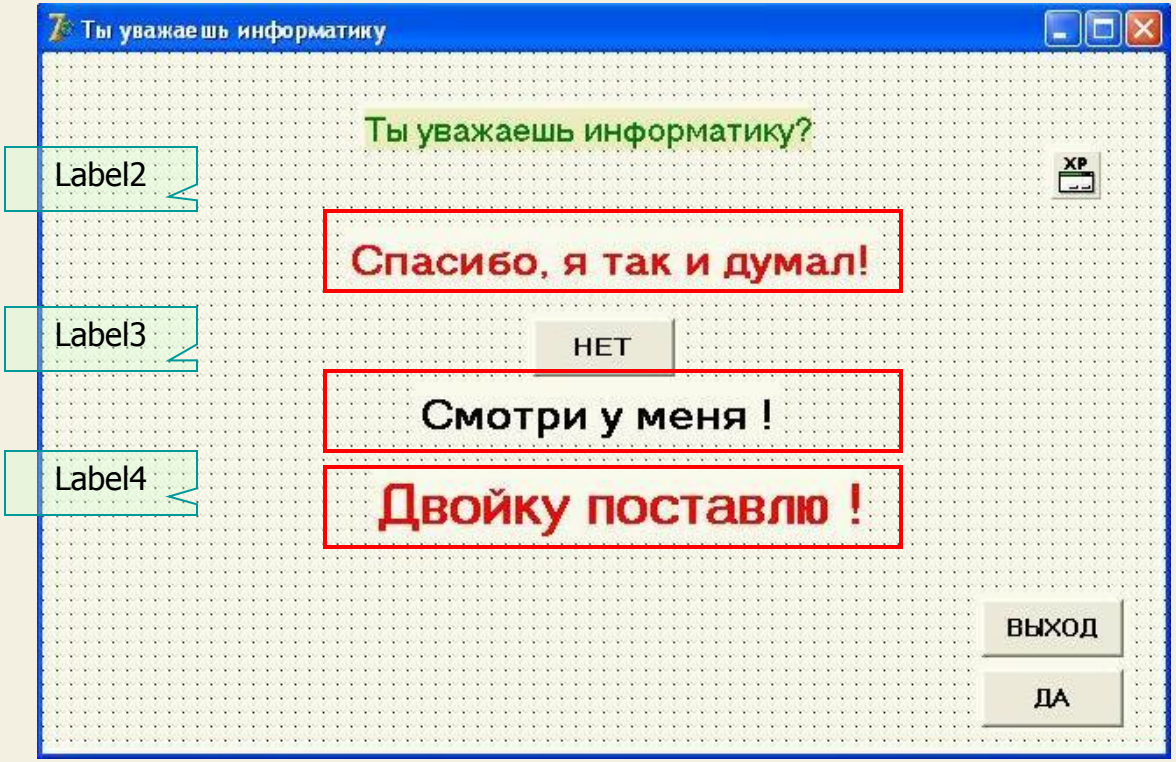
Раскрываем свойство Font объекта Label1 и подбираем нужный размер и цвет шрифта



ШАГ 4

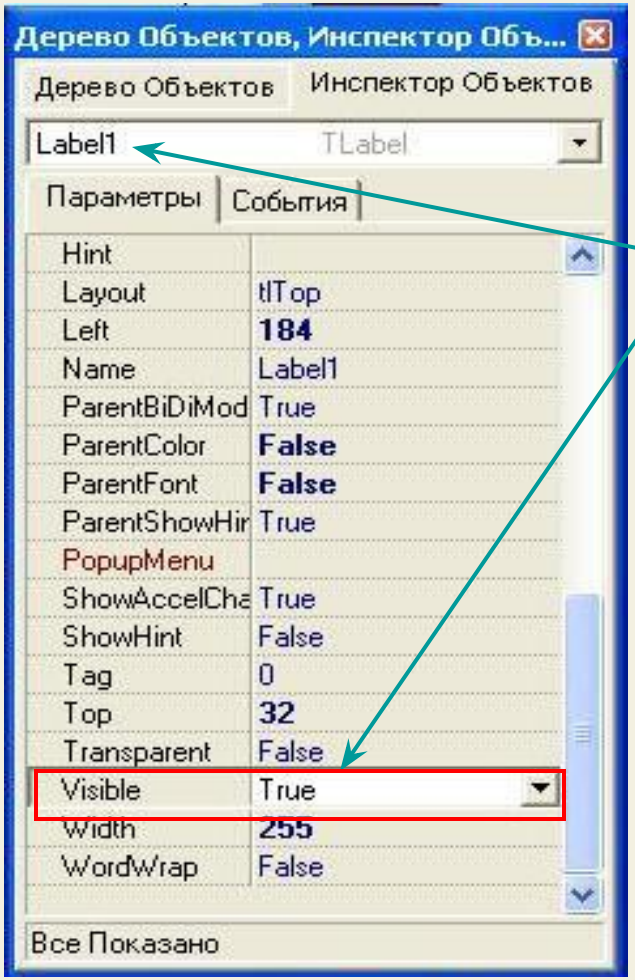
Аналогичные операции проделываем с надписями Label2, Label3, Label4. Для того, чтобы изменять свойства например Label2, щелкните по нему кнопкой и в инспекторе объектов открываются его свойства

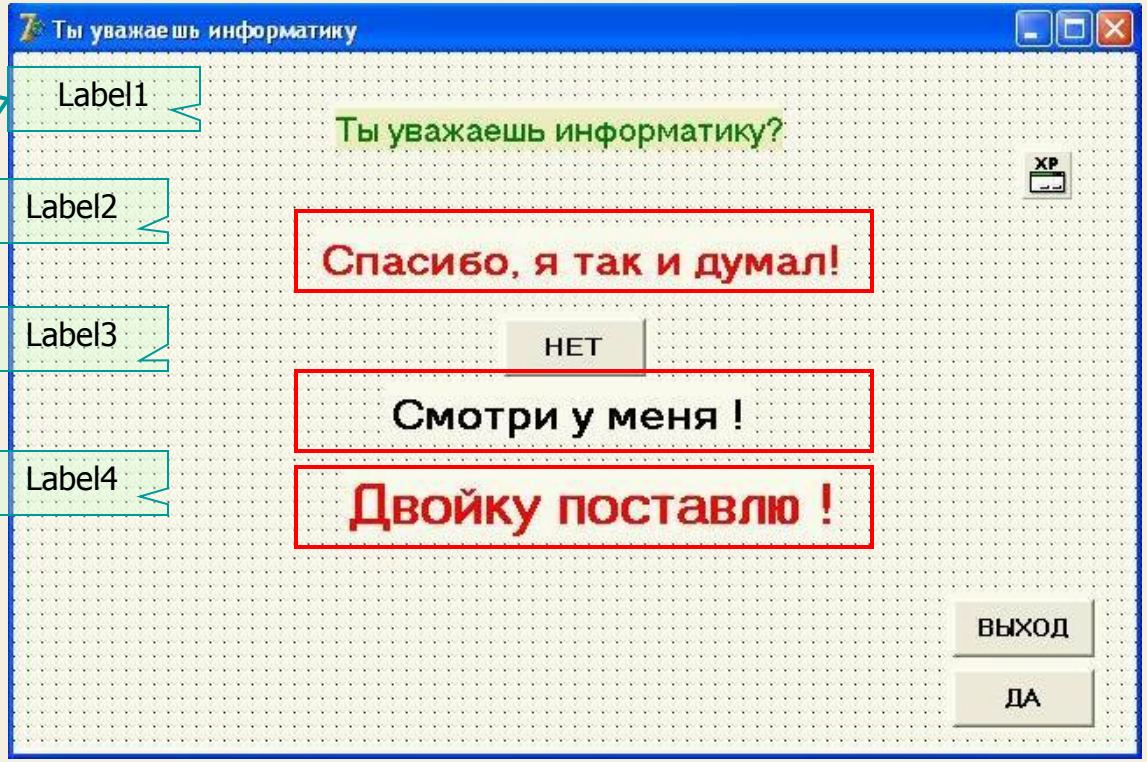




ШАГ 5

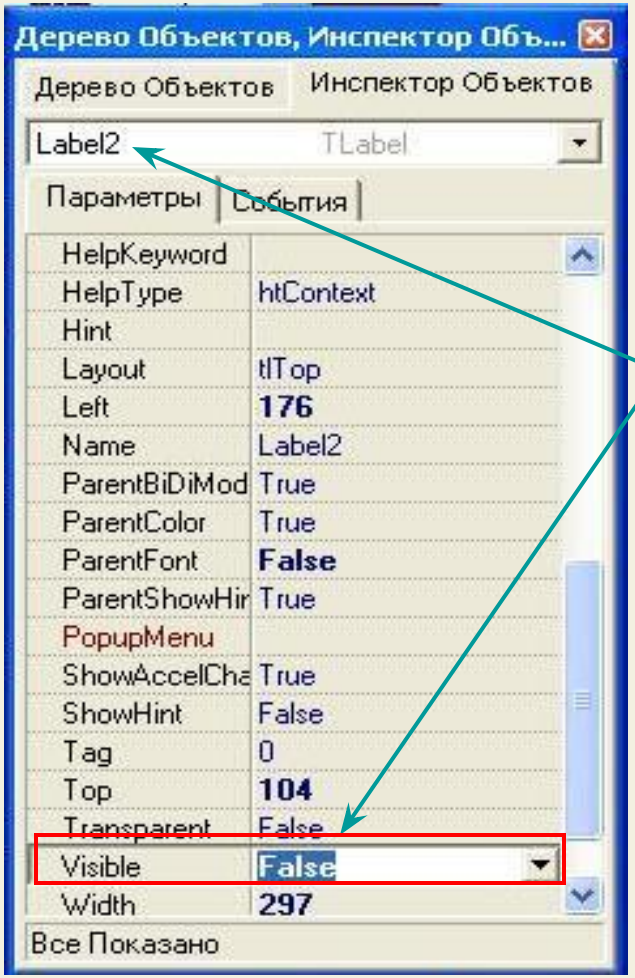
Сейчас установим нужные значения свойства Visible (видимость) у надписей. В момент запуска программы должна быть видна только надпись Label1(ее свойство Visible должно иметь значение True (истинно)), а остальные надписи не видны (у них свойство Visible должно иметь значение False (ложно))

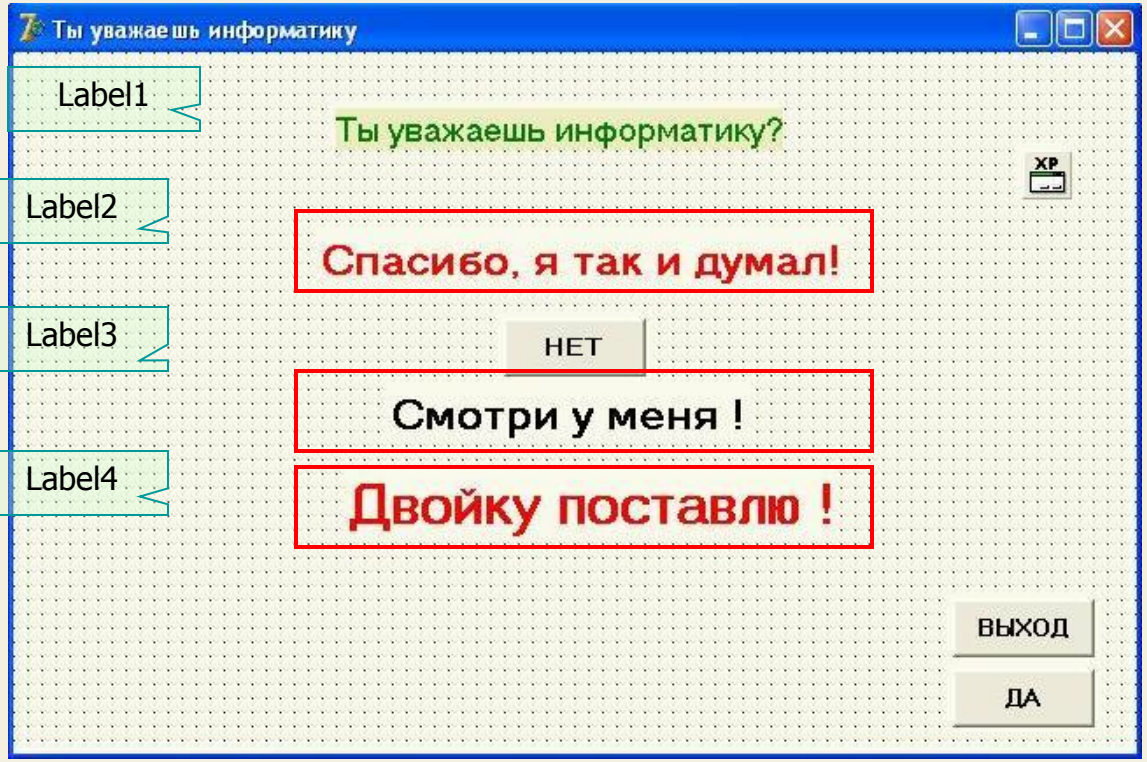




ШАГ 5

Сейчас установим нужные значения свойства Visible (видимость) у надписей. В момент запуска программы должна быть видна только надпись Label1(ее свойство Visible должно иметь значение True (истинно)), а остальные надписи не видны (у них свойство Visible должно иметь значение False (ложно))



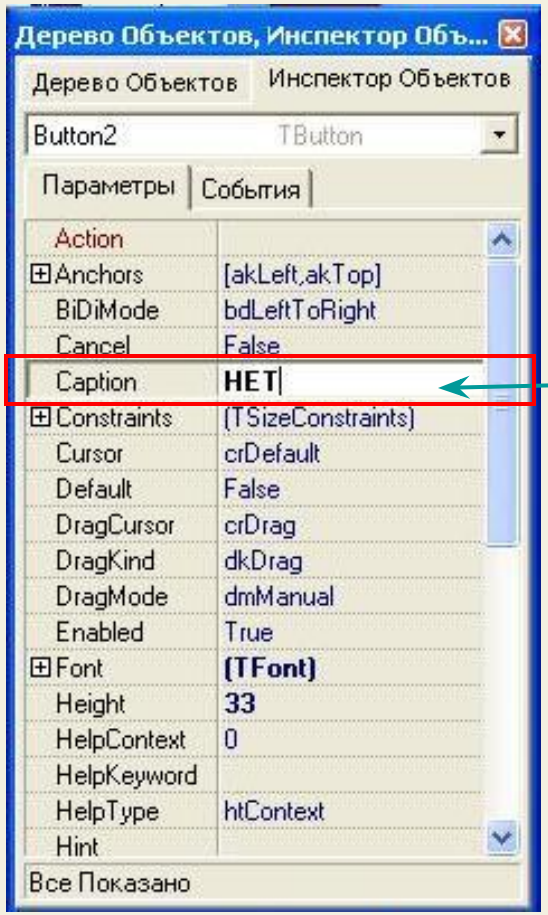


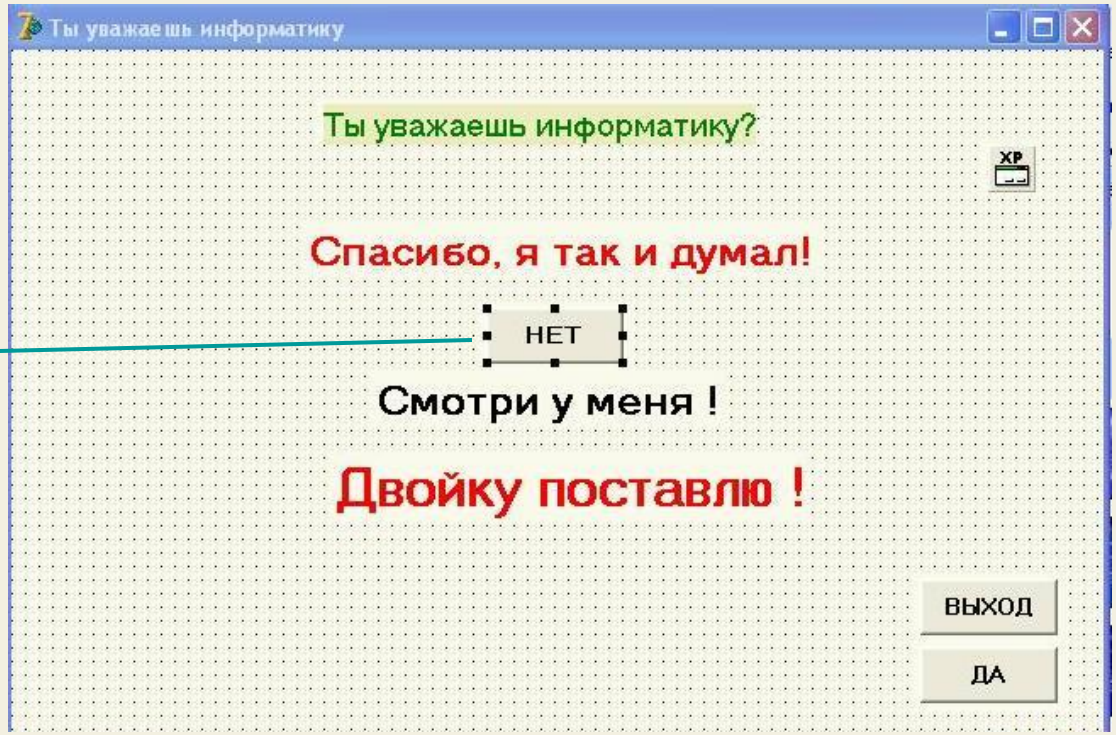
ШАГ 6

А сейчас давайте установим свойства кнопок. Изменим надписи на кнопках следующим образом:

На кнопке 1 (Button1) – надпись «ДА», на кнопке 2 – надпись «НЕТ» и на кнопке 3 – надпись «ВЫХОД», затем разместим кнопки как на рисунке

Делаем это опять через инспектор объектов



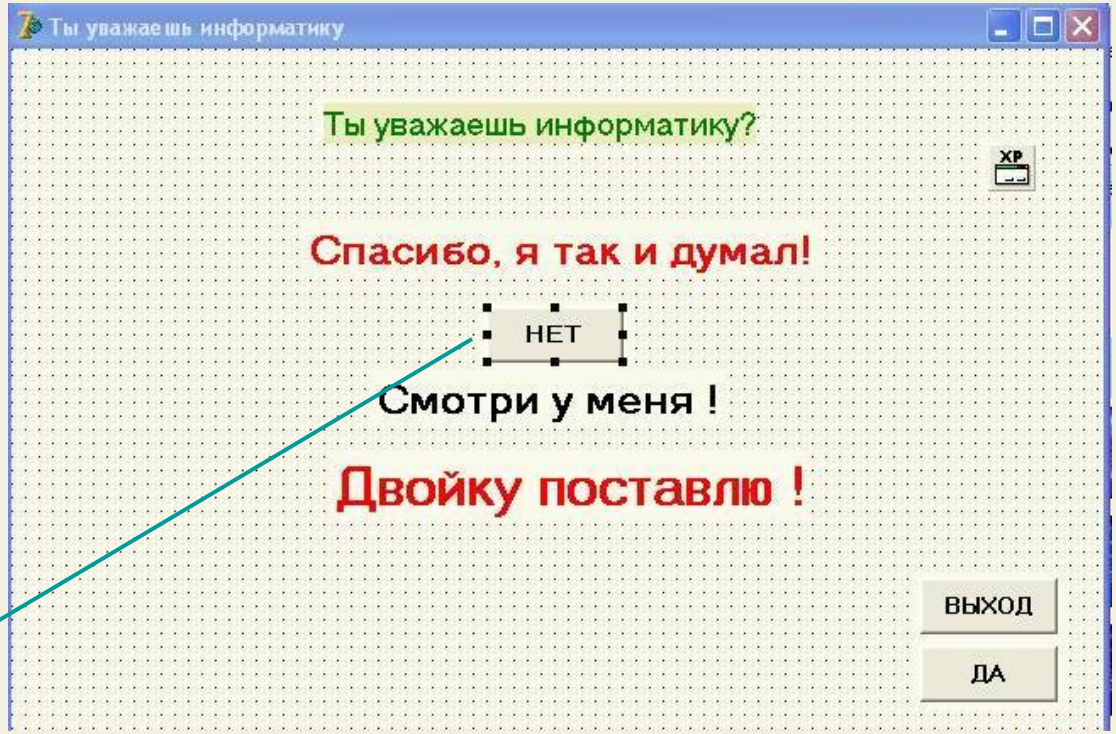


Аналогично сделаем надписи на кнопках 2 и 3

ШАГ 7

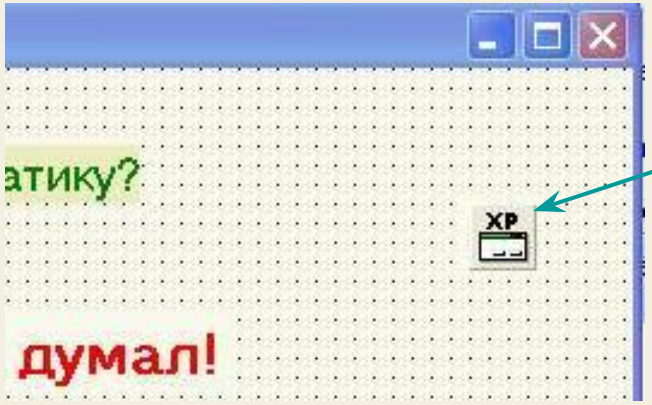
В момент запуска программы должны быть видны кнопки «ДА» и «НЕТ», а кнопка «ВЫХОД» не видна, поэтому установим свойство Visible для кнопок таким:

Для кнопок 1 и 2 свойство Visible устанавливаем в True, а для кнопки 3 («ВЫХОД») - в False



ШАГ 8
(необязательный)

Вы наверное заметили в правом верхнем углу формы значок



Это так называемый манифест XP - элемент, украшающий наше приложение (автоматически кнопочки и другие элементы становятся в стиле Windows XP - более привлекательные

Находится он на вкладке **Win32** панели компонентов

Щелкните по нему левой, а затем щелкните в любом месте формы.

Этот элемент является невизуальным, т.е. он существует и что-то делает, но при работе программы на форме он невиден, поэтому помещать его можно на форме где угодно.

В Delphi много невидимых компонент (например Таймер), они не видны, но делают свое дело – дальше мы их рассмотрим

Borland®

1. Создание процедуры обработки событий

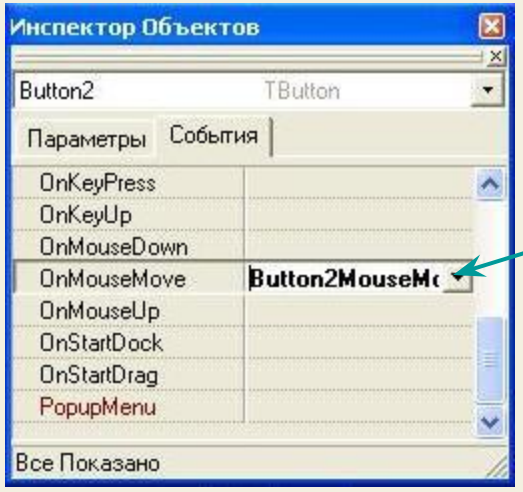
ШАГ 9

Итак, мы создали форму, разместили на ней компоненты и установили им нужные свойства, настало время писать код обработки событий

Сначала для кнопки «НЕТ» (Button1). Эта кнопка должна «убегать» при попытке наведения на нее курсора с целью нажатия, причем убегать случайным образом

Вот здесь мы и используем событие, которое возникает при наведении курсора на компонент – **OnMouseMove** – при этом координаты (положение кнопки «НЕТ») должно изменяться случайно (но за пределы формы она не должна выскакивать)

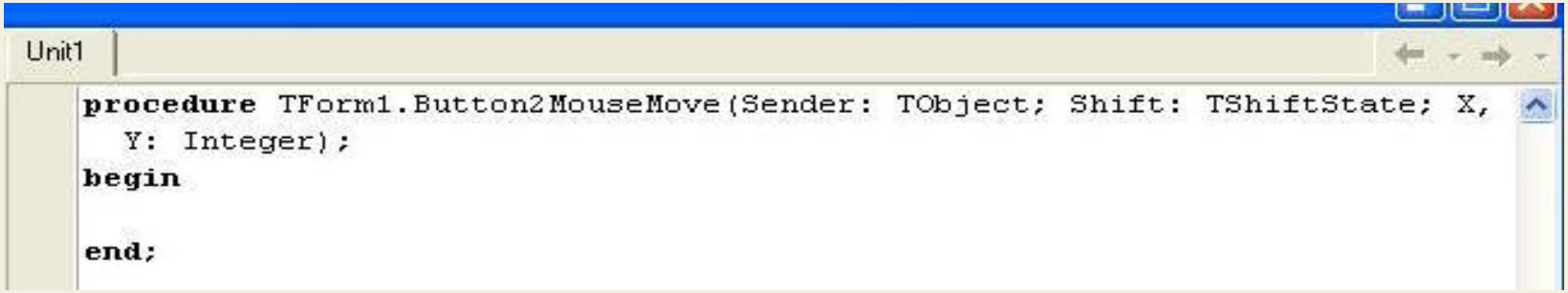
И еще – сделаем ученику предупреждение, первый раз после 3-х попыток наведения на кнопку «НЕТ», и последнее предупреждение после 6 попыток



Щелкнем один раз по кнопке «НЕТ», а затем в инспекторе объектов выберем вкладку СОБЫТИЯ. Далее находим событие **OnMouseMove** и в правой части делаем двойной щелчок – и мы оказываемся в редакторе исходного кода программы

ШАГ 10

Delphi автоматически создала процедуру обработки события – наведения курсора на кнопку «НЕТ»



В этой процедуре между **begin** и **end** мы вставим код

```
Button2.Left:=200+random(250);
Button2.Top:=50+random(250);
```

Положение кнопки2 от левого края формы будет изменяться случайно от 200 до 450 (вспомните Паскаль)

Положение кнопки2 от верхнего края формы будет изменяться случайно от 50 до 300

Таким образом при наведении курсора координата кнопки по горизонтали и вертикали случайно изменится – кнопка перепрыгнет на другое место

ШАГ 10

Сейчас запрограммируем предупреждения. Для этого нам понадобится **счетчик** – переменная, значение которой при каждой попытке наведения курсора на кнопку увеличивалось бы на единицу, и когда ее значение превысит 3 – выходит первое предупреждение, а когда 6 - второе

Назовем переменную - **k.** И ее необходимо объявить – описать в интерфейсной части кода модуля:

```
  private
    { Private declarations }
  public
    { Public declarations }
  end;

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

И еще – каждый раз при открытии формы (запуске программы) значение счетчика (k) должно устанавливаться на ноль. Сделаем это так: делаем двойной щелчок по нашей форме, при этом открывается редактор кода и в нем автоматически созданная процедура создания формы **OnCreate** – впишем туда присваивание k:=0;

```
procedure TForm1.FormCreate(Sender: TObject);
begin
k:=0;
end;
```

ШАГ 10

А дальше в процедуру Mouse Move для кнопки «НЕТ» дописываем код проверки условия (как в Паскале)

Unit1

```
procedure TForm1.Button2MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
 Button2.Left:=200+random(250);
 Button2.Top:=50+random(250);
 k:=k+1;
 if k>3 then
 label3.Visible:=true;
 if k>6 then
 label4.Visible:=true;
end;
```

При каждом наведении курсора на кнопку прибавляем счетчику единицу

ШАГ 10

А дальше в процедуру Mouse Move для кнопки «НЕТ» дописываем код проверки условия (как в Паскале)

Unit1

```
procedure TForm1.Button2MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
 Button2.Left:=200+random(250);
 Button2.Top:=50+random(250);
 k:=k+1;
 if k>3 then
 label3.Visible:=true;
 if k>6 then
 label4.Visible:=true;
end;
```

Проверяем условие, и если значение счетчика оказывается больше трех, то делаем надпись «СМОТРИ У МЕНЯ» видимой (значение Visible делаем равным True) – появляется первое предупреждение ученику, который не любит информатику

ШАГ 10

А дальше в процедуру Mouse Move для кнопки «НЕТ» дописываем код проверки условия (как в Паскале)

Unit1

```
procedure TForm1.Button2MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
 Button2.Left:=200+random(250);
 Button2.Top:=50+random(250);
 k:=k+1;
 if k>3 then
 label3.Visible:=true;
 if k>6 then
 label4.Visible:=true;
end;
```

Проверяем условие, и если значение счетчика оказывается больше шести, то делаем надпись «ДВОЙКУ ПОСТАВЛЮ» тоже видимой (значение Visible делаем равным True) – появляется и второе предупреждение ученику, который не любит информатику

Как бы ни пытался ученик очень быстро нажать на кнопку «НЕТ», она все равно ускачет от него, даже если бы он и успел нажать вперед компьютера, то все равно ничего бы не произошло, т.к. процедуру обработки нажатия на кнопку **OnClick** мы не создали – нажимай сколько угодно ☺

ШАГ 11

Сейчас создадим процедуру обработки кнопки «ДА» (Button1). Для этого делаем двойной щелчок по кнопке и мы опять в редакторе кода:

По нажатию на эту кнопку должна появиться надпись «Спасибо, я так и думал», остальные надписи стать невидимы, кнопки «ДА» и «НЕТ» должны исчезнуть, а кнопка «ВЫХОД» - появиться

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 label2.Visible:=true;
 Button1.Visible:=false;
 Button2Visible:=false;
 Button3.Visible:=true;
 label3.Visible:=false;
 label4.Visible:=false;
end;
```

Делаем видимой надпись «Спасибо, я так и думал» (Label2)

ШАГ 11

Сейчас создадим процедуру обработки кнопки «ДА» (Button1). Для этого делаем двойной щелчок по кнопке и мы опять в редакторе кода:

По нажатию на эту кнопку должна появиться надпись «Спасибо, я так и думал», остальные надписи стать невидимы, кнопки «ДА» и «НЕТ» должны исчезнуть, а кнопка «ВЫХОД» - появиться

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 label2.Visible:=true;
 Button1.Visible:=false;
 Button2Visible:=false;
 Button3.Visible:=true;
 label3.Visible:=false;
 label4.Visible:=false;
end;
```

Делаем невидимыми кнопки «ДА» и «НЕТ»

ШАГ 11

Сейчас создадим процедуру обработки кнопки «ДА» (Button1). Для этого делаем двойной щелчок по кнопке и мы опять в редакторе кода:

По нажатию на эту кнопку должна появиться надпись «Спасибо, я так и думал», остальные надписи стать невидимы, кнопки «ДА» и «НЕТ» должны исчезнуть, а кнопка «ВЫХОД» - появиться

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 label2.Visible:=true;
 Button1.Visible:=false;
 Button2Visible:=false;
 Button3.Visible:=true;
 label3.Visible:=false;
 label4.Visible:=false;
end;
```

Делаем видимой кнопку «ВЫХОД»

ШАГ 11

Сейчас создадим процедуру обработки кнопки «ДА» (Button1). Для этого делаем двойной щелчок по кнопке и мы опять в редакторе кода:

По нажатию на эту кнопку должна появиться надпись «Спасибо, я так и думал», остальные надписи стать невидимы, кнопки «ДА» и «НЕТ» должны исчезнуть, а кнопка «ВЫХОД» - появиться

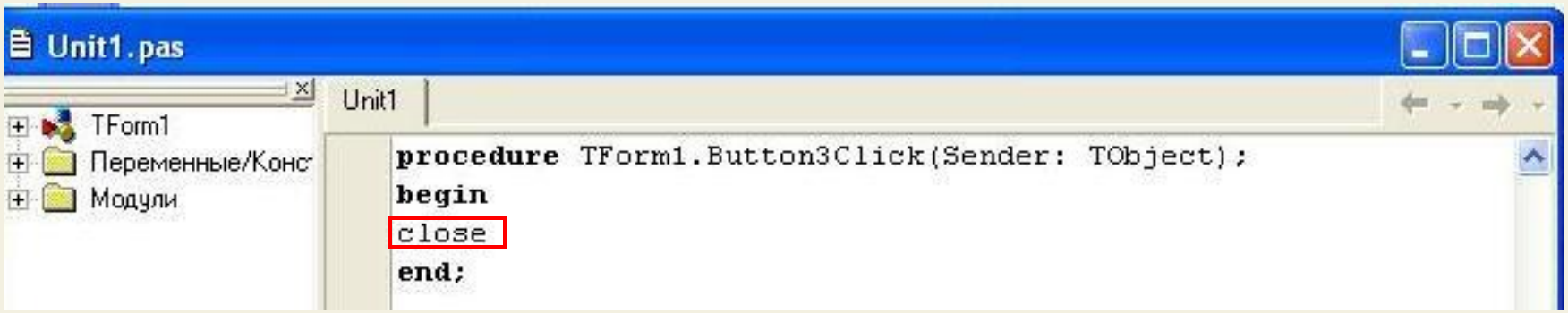
```
procedure TForm1.Button1Click(Sender: TObject);
begin
 label2.Visible:=true;
 Button1.Visible:=false;
 Button2Visible:=false;
 Button3.Visible:=true;
 label3.Visible:=false;
 label4.Visible:=false;
end;
```

Делаем невидимыми предупреждения ученику

Осталось написать код для кнопки **«ВЫХОД»**

ШАГ 12

Для этого делаем двойной щелчок по кнопке «ВЫХОД» и в редакторе кода записываем оператор **close** (закрыть форму)



ШАГ 13

Ну вот и все, можно проверить работу программы, но прежде давайте все сохраним (это мы уже умеем: Файл – сохранить все)

Сейчас запускаем программу (F9) и проверяем работу (при этом произойдет и компиляция программы – появится EXE – файл в каталоге сохранения)

Если возникают ошибки компиляции (компилятор выдает предупреждение о характере ошибки) – посмотрите внимательно код программы. Помните, что если мы забыли, например, точку с запятой (;) – то работать не будет

Для проверки к уроку приложены исходники программы (файл проекта и .pas – файл)

ИТОГИ УРОКА:

На этом уроке мы научились программно изменять некоторые свойства объектов и создавать процедуры обработки событий на примере создания игровой программы

НА СЛЕДУЮЩЕМ УРОКЕ:

ООП на Delphi – 4:

Мы научимся создавать и проверять условия, пользоваться компонентами Radio Button и Checkbox и составим простейшую тестирующую программу

Домнин Константин Михайлович
E – mail: kdomnin@list.ru
2006 год.