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 - 5

DELPHI - 5

На этом уроке:

Мы научимся использовать элементы ввода и вывода информации и составим программу расчета корней квадратного уравнения, а также познакомимся с обработкой исключительных ситуаций

Вопросы:

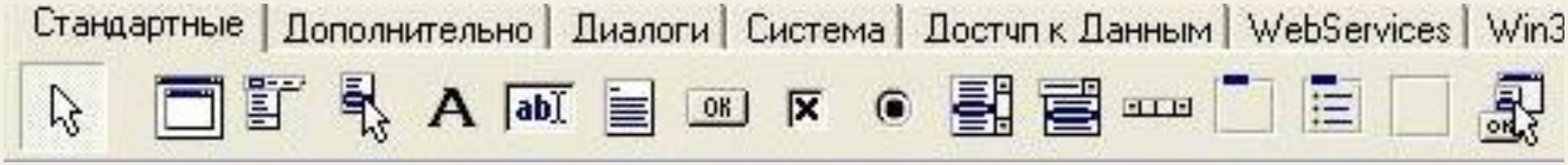
1. Элементы ввода и вывода информации
2. Создание программы расчета корней квадратного уравнения
3. Обработка исключений

Borland®

1. Элементы ввода и вывода информации

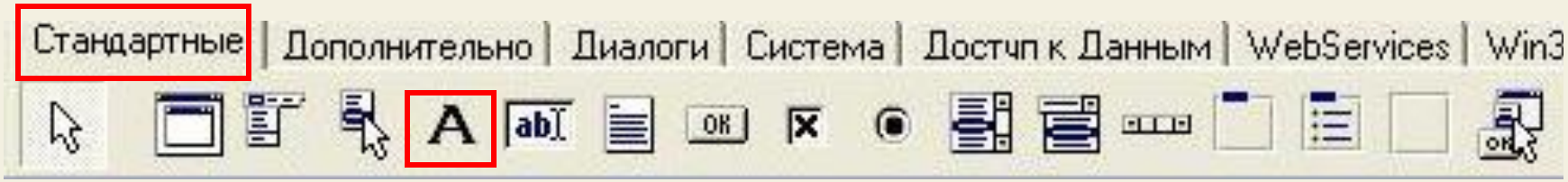
В каждой программе для взаимодействия компьютера и пользователя должны быть средства для ввода информации и получения ее от программы

Delphi содержит большой набор компонент для ввода, вывода и редактирования информации



Рассмотрим подробнее некоторые часто применимые компоненты, их свойства и применение

Компонент Label (метка)



Label (метка)

Применяется для отображения текста, который не изменяется пользователем. (только программно)

Основное свойство: Caption (надпись)

Это свойство можно изменять программно:

```
Label1.Caption:=‘Привет !’;
```

Некоторые дополнительные свойства

Изменение цвета шрифта

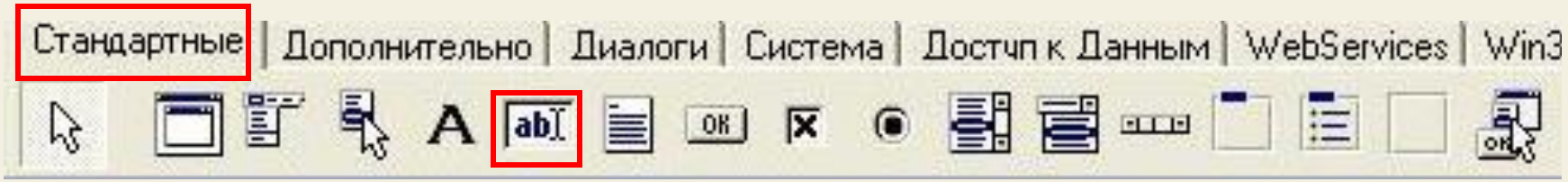
```
Label1.Font.Color:=RGB(200,100,200);
```

Изменение размера шрифта

```
Label1.Font.Size:=24;
```

☝ Посмотрите внимательно набор свойств компонента Label в инспекторе объектов и поэкспериментируйте с ними

Компонент Edit (окно редактирования)



Edit

Отображение, ввод и редактирование однострочных текстов. Имеется возможность оформления объемного бордюра.

Основное свойство: Text

Это свойство можно изменять программно и путем ввода с клавиатуры

Edit1.Text:='Привет!';

Некоторые дополнительные свойства:

Изменение цвета шрифта

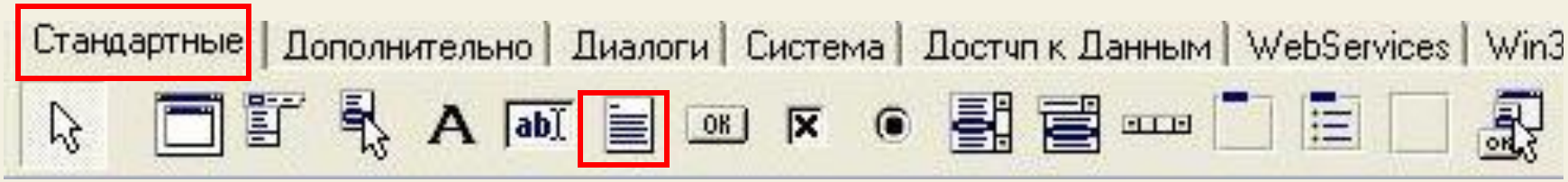
Edit1.Font.Color:=rgb(255,255,255); / цвет шрифта – белый/

Изменение фона окна

Edit1.Color:=rgb(0,0,0); /фон окна – черный/

☝ Посмотрите внимательно набор свойств компонента **Edit** в инспекторе объектов и поэкспериментируйте с ними

Компонент Memo (многострочное окно редактирования)

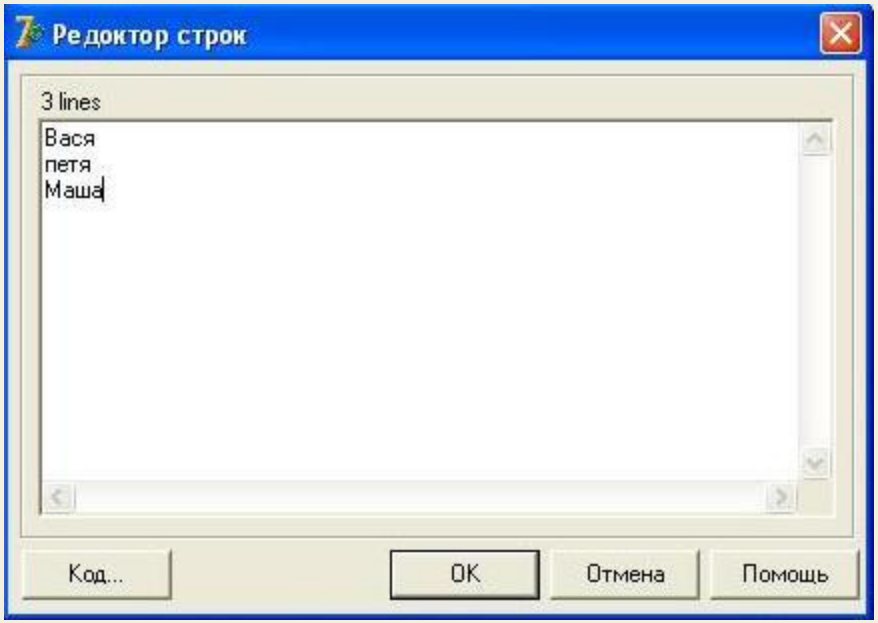


Memo

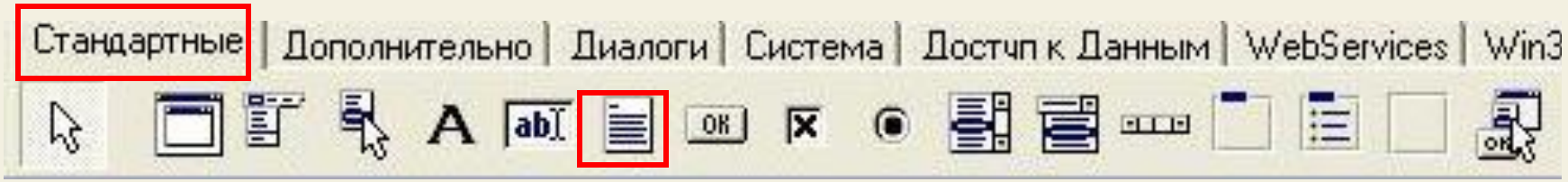
Отображение, ввод и редактирование многострочных текстов. Имеется возможность оформления объемного бордюра.

Основное свойство: Lines

Откройте в инспекторе объектов свойство **Lines** и у Вас откроется редактор строк



Компонент Memo (многострочное окно редактирования)



Memo

Отображение, ввод и редактирование многострочных текстов. Имеется возможность оформления объемного бордюра.

В компоненте **Memo** формат (шрифт, его атрибуты, выравнивание) одинаков для всего текста и определяется свойством **Font**.

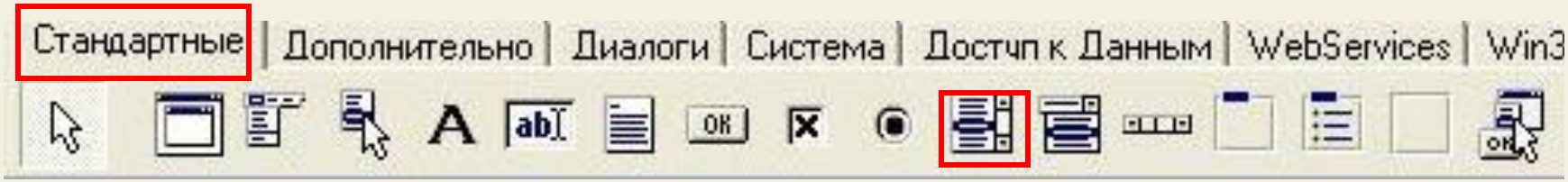
Если в коде мы запишем:

```
memo1.Color:=rgb(0,255,255);
memo1.Font.Size:=14;
memo1.Font.Color:=rgb(255,0,0);
memo1.Text:='Привет !';
```

то увидим следующее:

Посмотрите внимательно набор свойств компонента **Memo** в инспекторе объектов и поэкспериментируйте с ними

Компонент List Box



List Box

Отображение стандартного окна списка Windows, позволяющего пользователю выбирать из него пункты.

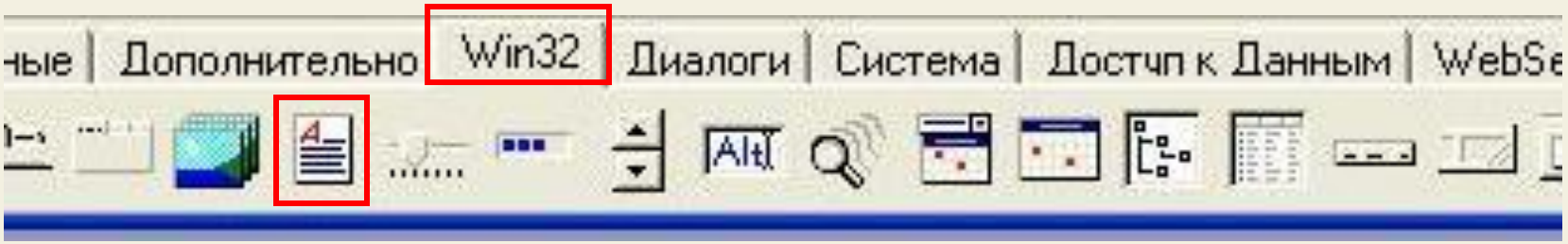
Основное свойство: Items

listbox1.Items.Add('Оля'); - добавление записи Оля

listbox1.Color:=rgb(0,255,255);; - установка цвета фона бокса

Посмотрите внимательно набор свойств компонента **List Box** в инспекторе объектов и поэкспериментируйте с ними

Компонент Rich Edit



Rich Edit

Компонент представляет собой окно редактирования в стиле Windows 95 в обогащенном формате RTF, позволяющее производить выбор атрибутов шрифта, поиск текста и многое другое .

Элемент **Rich Edit** похож по свойствам на Memo, но обладает большими возможностями

Вы можете менять атрибуты текста, выполняя отдельные фрагменты различными шрифтами, размерами, цветами, стилями. Устанавливаемые атрибуты влияют на выделенный текст или, если ничего не выделено, то на атрибуты нового текста, вводимого начиная с текущей позиции курсора

☝ Посмотрите внимательно набор свойств компонента **Rich Edit** в инспекторе объектов и поэкспериментируйте с ними

На этом знакомство с компонентами ввода, вывода и редактирования информации закончим

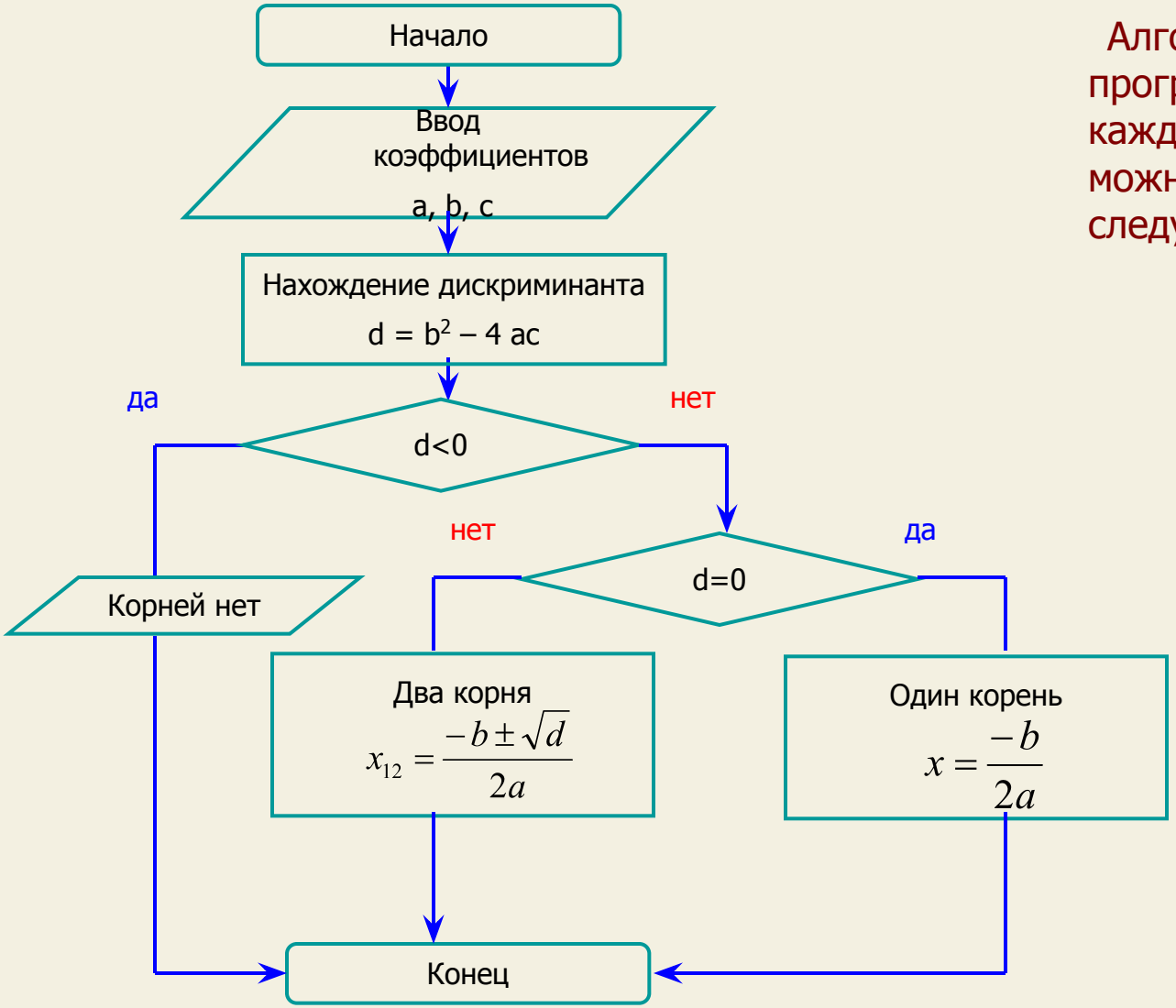
Отметим, что мы рассмотрели лишь часто применимые компоненты, с другими вы можете познакомиться в справочнике А.Я. Архангельского «100 компонентов общего назначения Delphi», который приложен к данному курсу

Сейчас, используя полученные знания, создадим программу расчета корней квадратного уравнения, применив элементы ввода, редактирования и вывода информации

Borland®

2. Создаем программу расчета корней квадратного уравнения

Прежде, чем приступить к разработке программы, мы должны разработать ее алгоритм (что является, пожалуй, самым сложным), а затем реализовать его в конкретной системе программирования



Алгоритм данной программы известен каждому школьнику, его можно изобразить в виде следующей блок - схемы

После разработки алгоритма приступим к его реализации в Delphi

На форме должны быть:

- 3 Edit-а для ввода коэффициентов уравнения
- 1 Edit для отображения вычисления дискриминанта
- 2 Edit-а для отображения вычисления корней уравнения
- 2 кнопки:

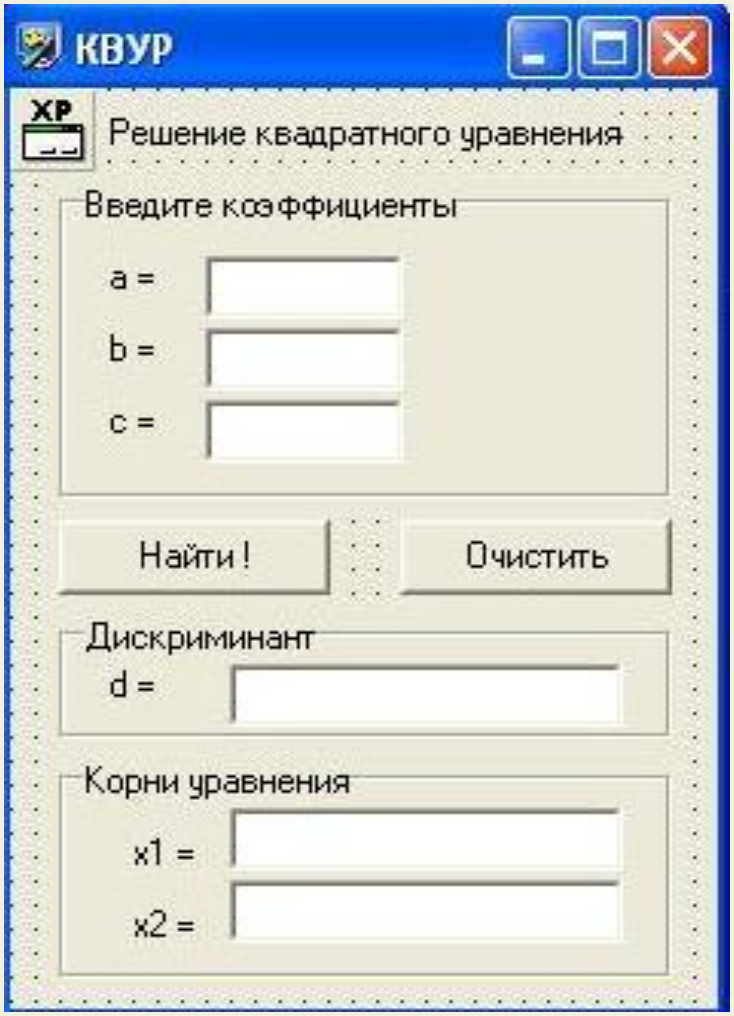
  «НАЙТИ» - для вычисления корней

  «ОЧИСТИТЬ» - для очистки всех Edit – ов
- несколько Label –ов для вывода текста

Причем для украшения программы применим компонент Manifest XP, а также Group Box

Рассмотрим создание программы по шагам, причем объяснений будет уже меньше – мы уже кое – что умеем (смотри предыдущие уроки)

ШАГ 1 Запускаем Delphi, размещаем на форме необходимые компоненты:



Group Box2

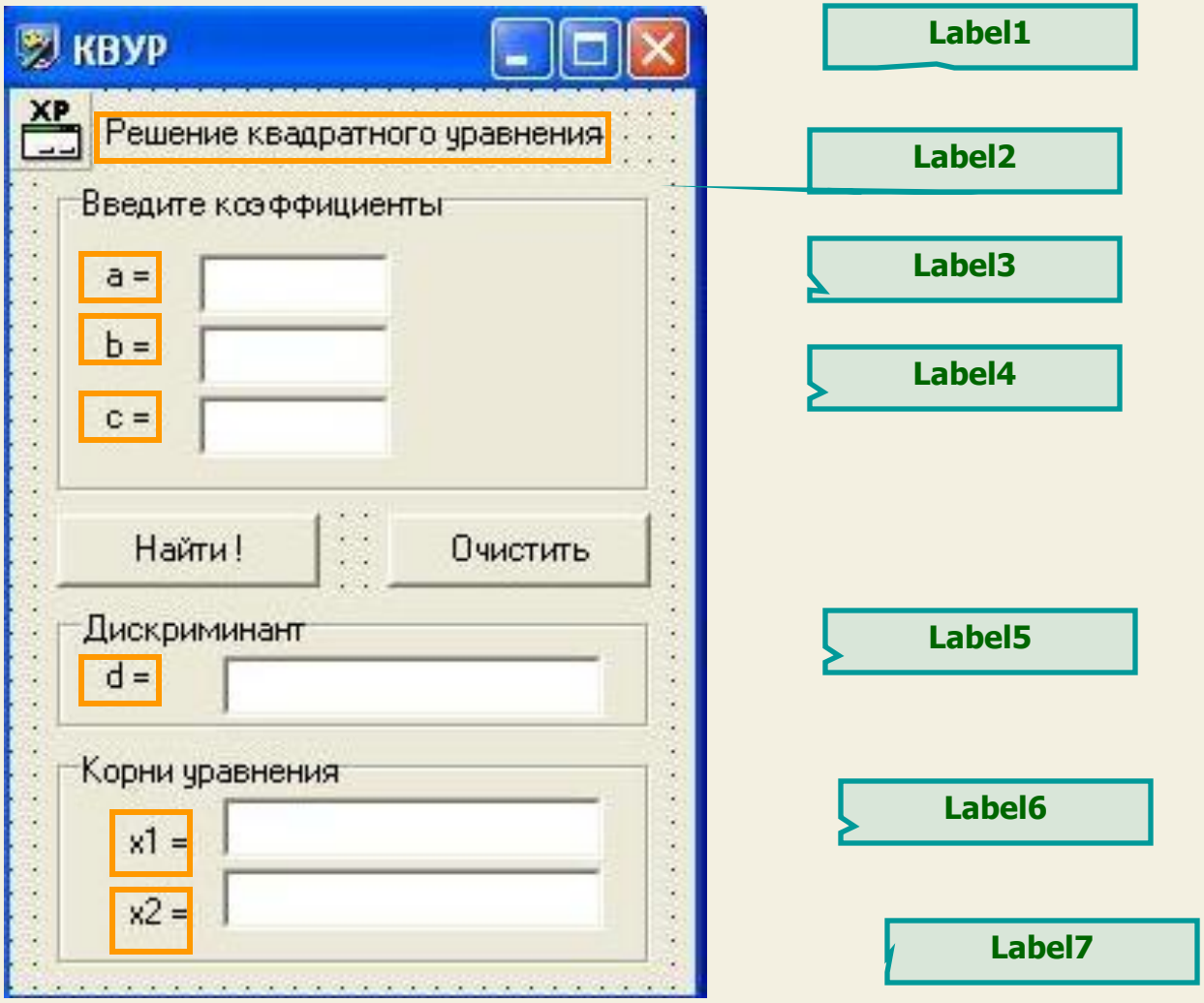
Group Box3

Group Box1

Компонент Group Box находится на стандартной панели компонент

В свойстве Group Box – Caption делаем соответствующие надписи

ШАГ 1 Размещаем все Label –ы для отображения текста



ШАГ 1 Размещаем Edit-ы для отображения и ввода данных

Edit1

Edit2

Edit3

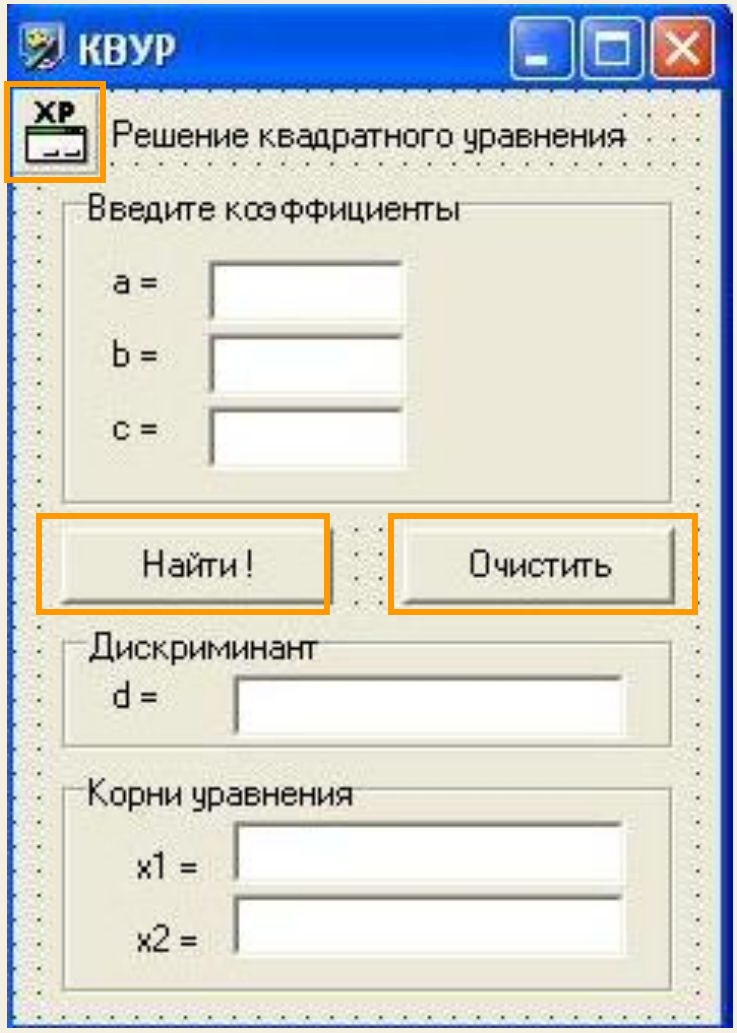
Edit6

Edit4

Edit5

ШАГ 1

И наконец размещаем кнопки и манифест XP



Manifest XP

Button1

Button2

Делаем соответствующие надписи на кнопках, подгоняем размер формы – и сейчас можно приступать к написанию кода

ШАГ 2

Сейчас в интерфейсной части модуля необходимо объявить используемые переменные (нажмите F12 – и вы в редакторе)

Init1

```
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    XPManifest1: TXPManifest;
    GroupBox3: TGroupBox;
    Label5: TLabel;
    Edit6: TEdit;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
   a,b,c,d,x1,x2:real;
implementation

{$R *.dfm}
```

Объявляем переменные и указываем их тип – real (все величины могут принимать действительные значения

ШАГ 3

Приступаем к написанию кода, сначала для кнопки «НАЙТИ» (Button1). Делаем по ней двойной щелчок и мы в редакторе кода, где записываем следующий код (это знакомый нам Паскаль, но есть небольшие отличия)

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 a:=strtofloat(edit1.Text);
 b:=strtofloat(edit2.Text);
 c:=strtofloat(edit3.Text);
 d:=b*b-4*a*c;
 form1.Edit6.Text:=floattostr(d);
 if d<0 then
 begin
 form1.Edit5.Text:='Корней нет';
 form1.Edit4.Text:='Корней нет';
  end
  else
  begin
 form1.Edit5.Text:= floattostr((-b+sqrt(d))/(2*a));
 form1.Edit4.Text:= floattostr((-b-sqrt(d))/(2*a));
 end;
end;
```

Это процедура нажатия на кнопку «НАЙТИ», созданная Delphi автоматически

А эти три строчки нам пока незнакомы

Что это?

ШАГ 3

Приступаем к написанию кода, сначала для кнопки «НАЙТИ» (Button1). Делаем по ней двойной щелчок и мы в редакторе кода, где записываем следующий код (это знакомый нам Паскаль, но есть небольшие отличия)

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 a:=strtofloat(edit1.Text);
 b:=strtofloat(edit2.Text);
 c:=strtofloat(edit3.Text);
 d:=b*b-4*a*c;
 form1.Edit6.Text:=floattostr(d);
 if d<0 then
 begin
 form1.Edit5.Text:='Корней нет';
 form1.Edit4.Text:='Корней нет';
  end
  else
  begin
 form1.Edit5.Text:= floattostr((-b+sqrt(d))/(2*a));
 form1.Edit4.Text:= floattostr((-b-sqrt(d))/(2*a));
 end;
end;
```

Дело в том, что переменные a,b и c – вещественного типа, а значение окна редактирования (Edit.Text) –имеет строковый тип

Поэтому присваивание напрямую:

a:= Edit1.Text вызовет ошибку несоответствия типов

Выражение **strtofloat** – указание компилятору перевести строковый тип, присущий тексту Edit-a, в вещественный тип, соответствующий переменной a (b и c)

ШАГ 3

Приступаем к написанию кода, сначала для кнопки «НАЙТИ» (Button1). Делаем по ней двойной щелчок и мы в редакторе кода, где записываем следующий код (это знакомый нам Паскаль, но есть небольшие отличия)

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 a:=strtofloat(edit1.Text);
 b:=strtofloat(edit2.Text);
 c:=strtofloat(edit3.Text);
 d:=b*b-4*a*c;
 form1.Edit6.Text:=floattostr(d);
 if d<0 then
 begin
 form1.Edit5.Text:='Корней нет';
 form1.Edit4.Text:='Корней нет';
  end
  else
  begin
 form1.Edit5.Text:= floattostr((-b+sqrt(d))/(2*a));
 form1.Edit4.Text:= floattostr((-b-sqrt(d))/(2*a));
 end;
end;
```

Это понятно: вычисление дискриминанта

А здесь опять преобразование типов, только наоборот:

Значению Edit6.Text присваивается значение дискриминанта, но при этом вещественный тип переменной d преобразуется в строковый тип значения Edit-a

(floattostr)

ШАГ 3

Приступаем к написанию кода, сначала для кнопки «НАЙТИ» (Button1). Делаем по ней двойной щелчок и мы в редакторе кода, где записываем следующий код (это знакомый нам Паскаль, но есть небольшие отличия)

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 a:=strtofloat(edit1.Text);
 b:=strtofloat(edit2.Text);
 c:=strtofloat(edit3.Text);
 d:=b*b-4*a*c;
 form1.Edit6.Text:=floattostr(d);
 if d<0 then
 begin
 form1.Edit5.Text:='Корней нет';
 form1.Edit4.Text:='Корней нет';
  end
  else
  begin
 form1.Edit5.Text:= floattostr((-b+sqrt(d))/(2*a));
 form1.Edit4.Text:= floattostr((-b-sqrt(d))/(2*a));
 end;
end;
```

Проверяем условие (если дискриминант меньше нуля), то выводим в Edit-ах для корней значения «Корней нет»

Иначе рассчитываем корни и выводим их в соответствующих Edit-ах

(Здесь мы немного упростили, объединив две ветви алгоритма в одну)

ШАГ 4

Сейчас запишем код для кнопки «ОЧИСТИТЬ» (Button2)

Unit1

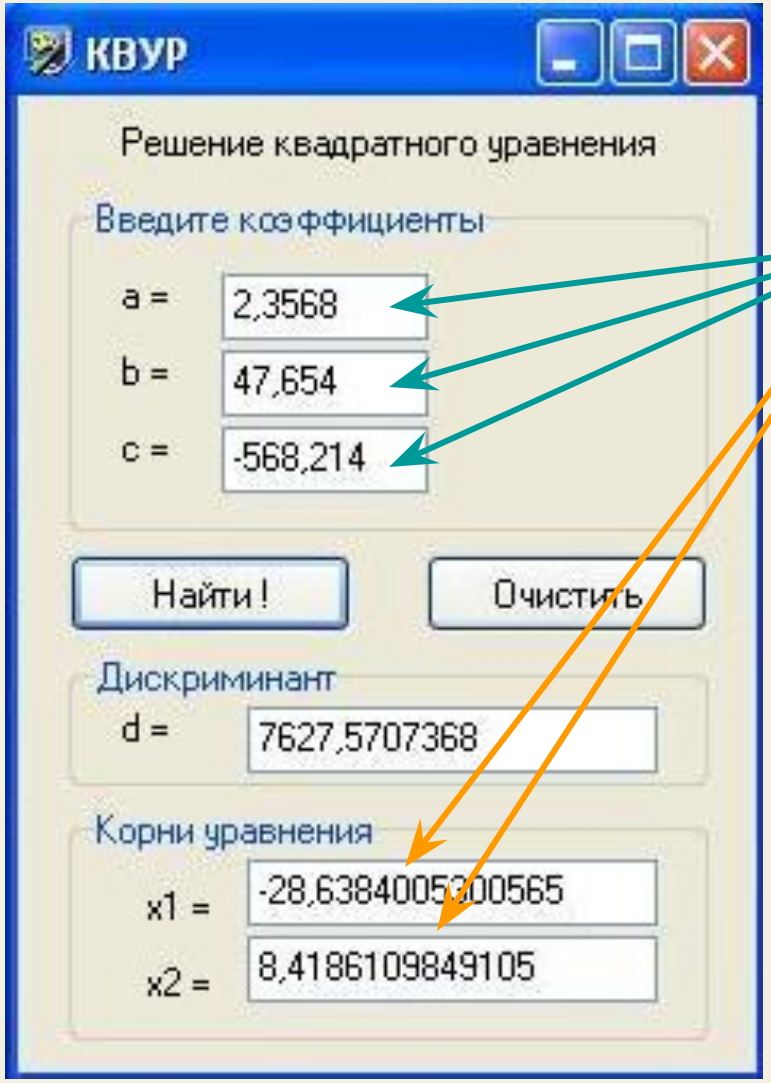
```
procedure TForm1.Button2Click(Sender: TObject);
begin
form1.Edit1.Text:='';
form1.Edit2.Text:='';
form1.Edit3.Text:='';
form1.Edit4.Text:='';
form1.Edit5.Text:='';
form1.Edit6.Text:='';
end;

end.
```

Свойству Text каждого Edit- а присваиваем пустое значение

ШАГ 6

Сохраняем проект, компилируем и запускаем готовую программу



Попробуем ввести значение коэффициентов и посмотреть результат

Попробовать ->

А сейчас проверьте работу программы по контрольным примерам:

a	b	c	x1	x2
1	2	1	-1	-1
2.18	-23.54	0.35	0.01488	10.7832
12.5	2.354	235.12	корней нет	корней нет

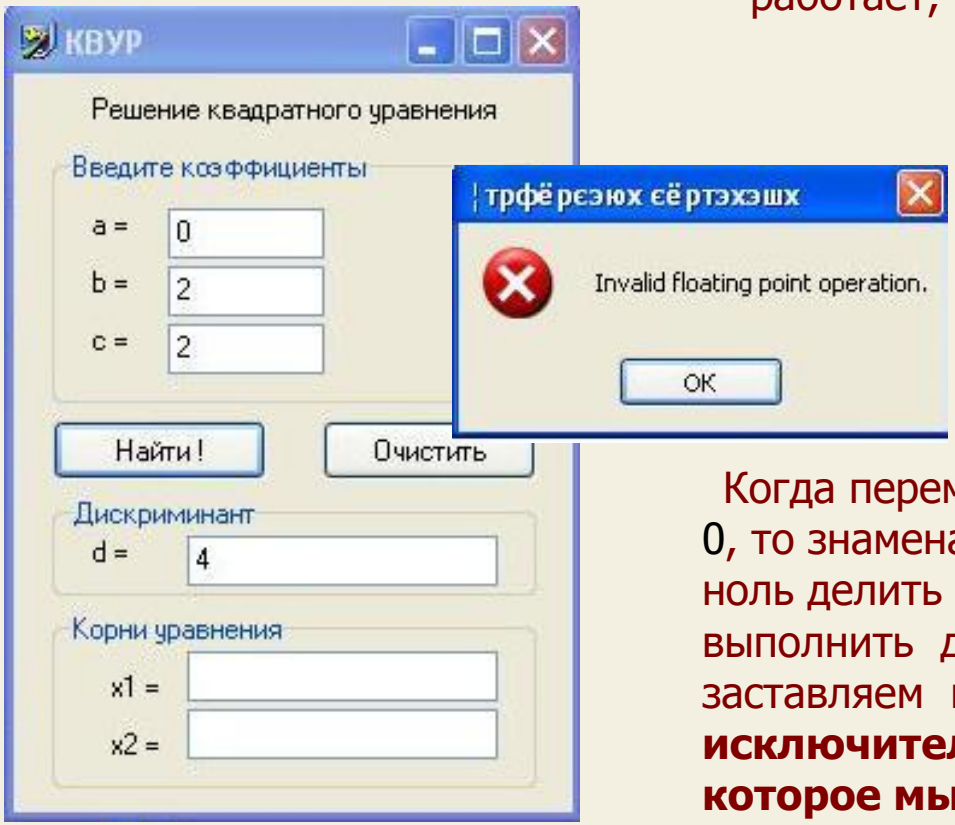
Borland®

2. Обработка исключений

Давайте разберемся, что такое исключение

Запустим калькулятор и попробуем ввести следующие значения коэффициентов:

(a=0, b и c – произвольные)



При нажатии кнопки «НАЙТИ» программа не работает, выходит окно сообщения

В чем дело ?

Давайте посмотрим, как мы находим корни уравнения

$$x_{12} = \frac{-b \pm \sqrt{d}}{2a}$$

Когда переменной a присваивается значение 0, то знаменатель этой дроби тоже ноль, а на ноль делить нельзя –программа не может выполнить действие, которое мы ее заставляем и прерывается – возникает **исключительная ситуация (исключение), которое мы должны обработать – написать код**

Обработка этого исключения

Ясно, что мы перед расчетом дискриминанта и корней должны проверить – а не введен ли в Label1 ноль. Если введен – то это уже не квадратное уравнение и считать надо по другой формуле (x= - c / b), если не введен – можно считать как обычно

```
procedure TForm1.Button1Click(Sender: TObject);
begin
a:=strtofloat(edit1.Text);
 b:=strtofloat(edit2.Text);
 c:=strtofloat(edit3.Text);
 if a=0 then
  begin
   form1.Edit4.Text:= ('Один корень');
    form1.Edit5.Text:= floattostr(-c/b);
    end
    else
    begin
 d:=b*b-4*a*c;
 form1.Edit6.Text:=floattostr(d);
 if d<0 then
 begin
 form1.Edit5.Text:='Корней нет';
  form1.Edit4.Text:='Корней нет';
  end
  else
  begin
  form1.Edit5.Text:= floattostr((-b+sqrt(d))/(2*a));
 form1.Edit4.Text:= floattostr((-b-sqrt(d))/(2*a));
 end;
end;
end;
```

Откроем файл нашего проекта (ведь он у нас сохранен) и перейдем в редактор кода

После присвоения переменным a,b и c значений из соответствующих Label-ов вставляем проверку этого условия

Если условие выполняется, то в одном Label-е выводим сообщение, а в другом – результат вычисления по другой формуле

Обработка этого исключения

Ясно, что мы перед расчетом дискриминанта и корней должны проверить – а не введен ли в Label1 ноль. Если введен – то это уже не квадратное уравнение и считать надо по другой формуле (x= - c / b), если не введен – можно считать как обычно

```
procedure TForm1.Button1Click(Sender: TObject);
begin
a:=strtofloat(edit1.Text);
 b:=strtofloat(edit2.Text);
 c:=strtofloat(edit3.Text);
 if a=0 then
  begin
   form1.Edit4.Text:= ('Один корень');
    form1.Edit5.Text:= floattostr(-c/b);
    end
    else
    begin
 d:=b*b-4*a*c;
 form1.Edit6.Text:=floattostr(d);
 if d<0 then
 begin
 form1.Edit5.Text:='Корней нет';
  form1.Edit4.Text:='Корней нет';
  end
  else
  begin
  form1.Edit5.Text:= floattostr((-b+sqrt(d))/(2*a));
 form1.Edit4.Text:= floattostr((-b-sqrt(d))/(2*a));
 end;
end;
end;
```

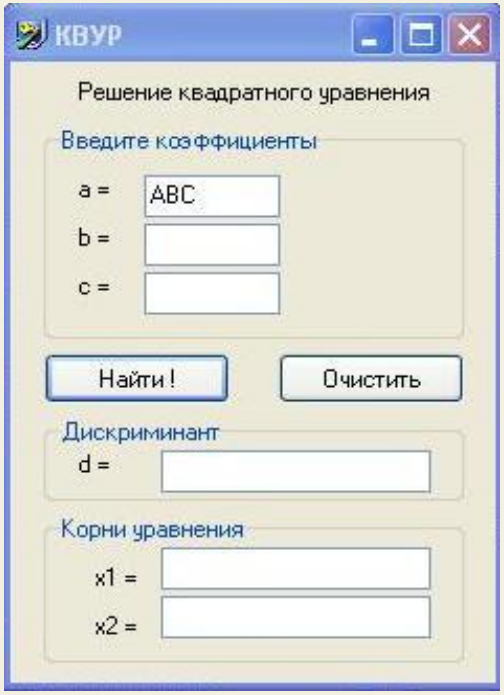
Иначе действуем по старому, не забудьте добавить begin …end

Сохраните, скомпилируйте, запустите программу и попробуйте ввести для a ноль, для b,c – произвольно: программа считает

попробовать

Таким образом мы обработали одно исключение, но может быть, есть еще что-то?

А что будет, если мы случайно в окно ввода коэффициента уравнения введем вместо числа – текст или какие – то знаки препинания (или вообще ничего не введем)?



Опять возникает ситуация, когда наша программа не может выполнить действие и прерывается, а нам выводится сообщение об этом . В этом случае мы заставляем программу переводить символы ABC в вещественное число, но эти символы числом не являются – возникает несоответствие типов

При создании даже простой программы мы всегда должны обдумывать и обрабатывать исключения, хотя часто они «не видны» и проявляются только в некоторых ситуациях при работе уже готовой программы

Обработка исключений – дело довольно кропотливое и в наш курс не входит, хотя на следующих уроках мы будем на это обращать внимание

ИТОГИ УРОКА:

На этом уроке мы научились использовать элементы ввода и вывода информации и составили программу расчета корней квадратного уравнения, использующие эти компоненты, познакомились с исключениями и как их обрабатывать

НА СЛЕДУЮЩЕМ УРОКЕ:

ООП на Delphi – 6:

Мы научимся создавать

- **приложения, содержащие несколько форм,**
- **заставку, появляющуюся при запуске программы**
- **приложения, использующие элемент таймер**

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