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, панель статуса, 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 - 11

DELPHI - 11

На этом уроке:

Мы создадим свой калькулятор, подобный встроенному в ОС Windows

Вопросы:

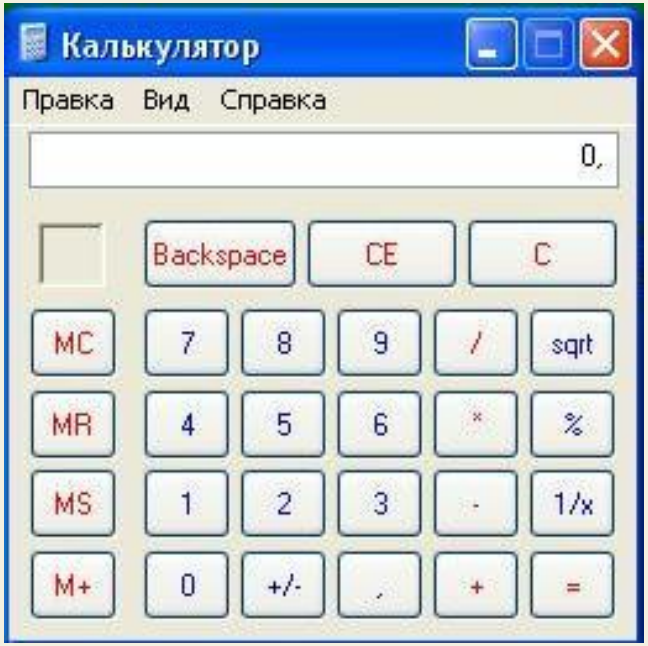
Весь урок мы будем создавать калькулятор и обрабатывать некоторые исключения

Borland®

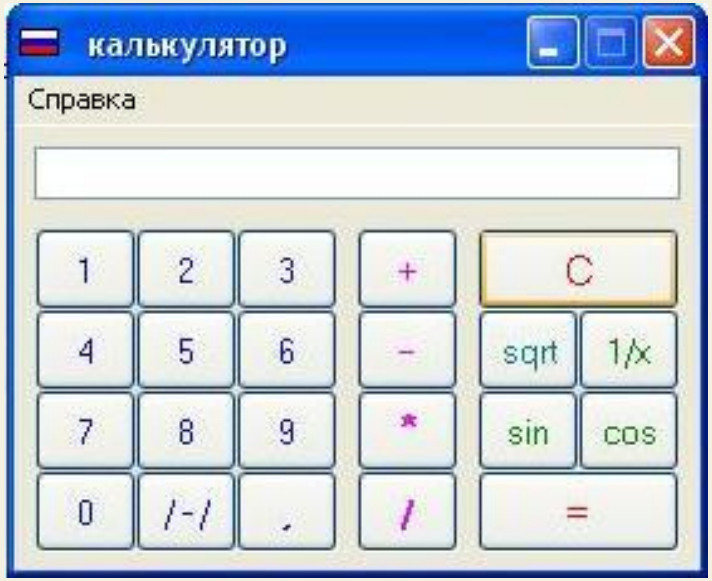
Создание калькулятора

На этом уроке мы попробуем создать **программу – калькулятор**, похожий на калькулятор Windows.

Windows - калькулятор



А такой мы создадим



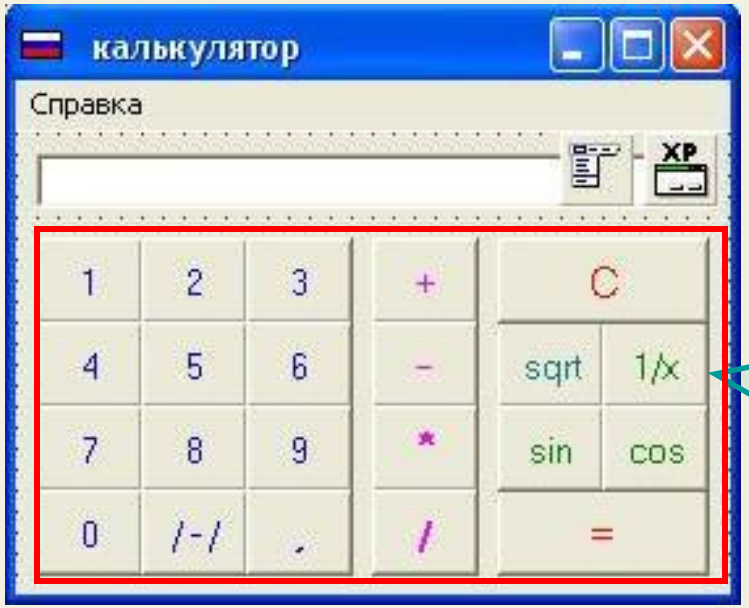
Сначала давайте попробуем в использовании калькулятор, а затем займемся его конструированием

Попробовать ->

ШАГ 1

Итак, начнем:

Разместим на форме нужные нам компоненты:



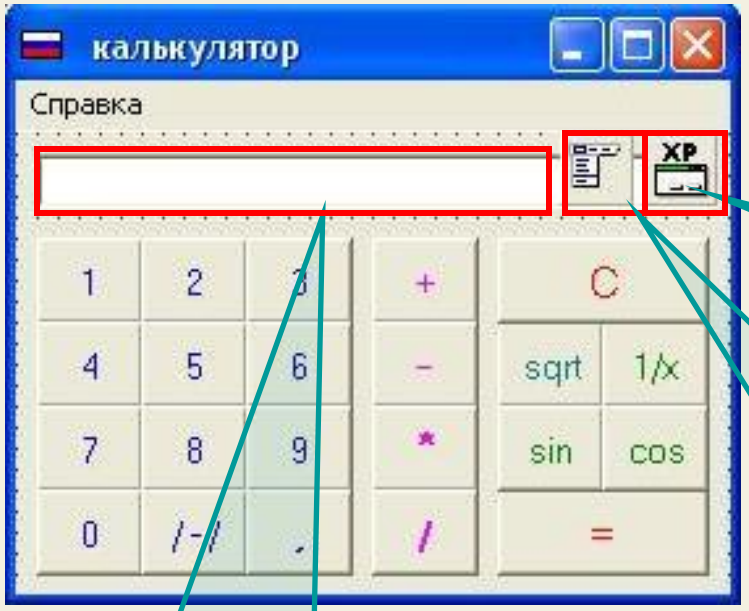
Кнопки цифр и действий (Вместо обычных кнопок Button возьмем кнопки **BitButton** – т.к на них можно менять цвет шрифта, а на обычных кнопках Windows не дает это сделать)

На кнопках сделаем соответствующие надписи, которые раскрасим через свойство Font кнопки

ШАГ 1

Итак, начнем:

Разместим на форме нужные нам компоненты:

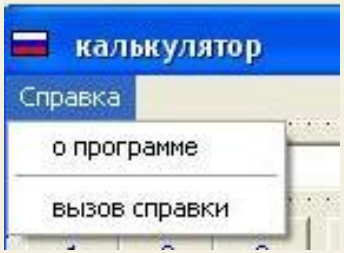


Манифест XP для украшения программы в стиле Windows XP

Edit для ввода и отображения цифр

MainMenu , в котором с помощью дизайнера создадим опции:

- О программе
- Вызов справки



ШАГ 2

Приступим к написанию кода

```
var
  Form1: TForm1;
   i,p,r:real;
  mode:integer;
```

Сначала объявим переменные:

i – в ней будет храниться первое введенное число (арифметические операции являются двухместными, например при сложении есть два числа (переменные), а результат – третье число (переменная)

p – в ней будет храниться второе число для осуществления операции

r – здесь будет результат нашего действия

mode – переменная, в которой будем хранить код действия (например сложению поставим в соответствие код 1, делению – 2 и т.д. Это нужно для осуществления множественного выбора при выполнении действий)

ШАГ 2 Приступим к написанию кода

1. Кнопки цифр

```
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
  edit1.Text:=edit1.Text+'1';
end;

procedure TForm1.BitBtn5Click(Sender: TObject);
begin
  edit1.Text:=edit1.Text+'2';
end;
```

Процедура нажатия на кнопку с цифрой 1

При нажатии на кнопку к тексту **Edit-a** прибавляется (приписывается) символ 1

При нажатии на кнопку с цифрой 2 к тексту **Edit-a** прибавляется символ 2

Аналогично описываем и все последующие кнопки с цифрами, а для десятичной запятой в коде будет ...+ ' , '

ШАГ 2 Приступим к написанию кода

2. Кнопки действий

Кнопка изменения знака числа

```
procedure TForm1.BitBtn8Click(Sender: TObject);
begin
if  edit1.Text <> '' then
begin
  i:=StrToFloat(edit1.Text);
   i:=-1*i;
     edit1.Text:=FloatToStr(i)
end;
end;
```

Если содержимое **Edit**-а не пустое, то меняем знак числа на противоположный, и выводим результат в том же **Edit**-е

Заметьте, что здесь мы еще и обработали один «глюк», который может появляться, если попробовать изменить знак числа, а числа в Edit-е еще нет

```
begin
if  edit1.Text <> '' then
```

ШАГ 2

Приступим к написанию кода

2. Кнопки действий

```
procedure TForm1.BitBtn13Click(Sender: TObject);
begin
  i:=StrToFloat(edit1.Text);
mode:=1;
edit1.Text:='';
end;

procedure TForm1.BitBtn14Click(Sender: TObject);
begin
  i:=StrToFloat(edit1.Text);
mode:=2;
edit1.Text:='';
end;

procedure TForm1.BitBtn15Click(Sender: TObject);
begin
  i:=StrToFloat(edit1.Text);
mode:=3;
edit1.Text:='';
end;

procedure TForm1.BitBtn16Click(Sender: TObject);
begin
i:=StrToFloat(edit1.Text);
mode:=4;
```

Кнопка сложения

Переменной **i** присваиваем значение, введенное в Edit

Переменной **mode** присваиваем 1 – код, соответствующий операции сложения

Очищаем содержимое **Edit**-а

ШАГ 2 Приступим к написанию кода

2. Кнопки действий

```
procedure TForm1.BitBtn13Click(Sender: TObject);
begin
  i:=StrToFloat(edit1.Text);
mode:=1;
edit1.Text:='';
end;
```

Кнопка вычитания

```
procedure TForm1.BitBtn14Click(Sender: TObject);
begin
  i:=StrToFloat(edit1.Text);
mode:=2;
edit1.Text:='';
end;
```

Кнопка умножения

```
procedure TForm1.BitBtn15Click(Sender: TObject);
begin
  i:=StrToFloat(edit1.Text);
mode:=3;
edit1.Text:='';
end;
```

Кнопка деления

```
procedure TForm1.BitBtn16Click(Sender: TObject);
begin
i:=StrToFloat(edit1.Text);
mode:=4;
```

Коды кнопок аналогичны кнопке сложения

ШАГ 2 Приступим к написанию кода

2. Кнопки действий

```
procedure TForm1.BitBtn17Click(Sender: TObject);
begin
if StrToFloat(edit1.Text)<0 then
 edit1.Text:= 'Недопустимый аргумент функции
 else
 begin
  i:=StrToFloat(edit1.Text);
 i:=sqrt(i);
  edit1.Text:=FloatToStr(i)
  end;
end;
```

Кнопка извлечения корня

И опять обрабатываем **исключение** (если попытаться извлечь корень из отрицательного числа, то будет ошибка)

Если содержимое **Edit**-а меньше нуля, то в **Edit**-е выводим «Недопустимый аргумент функции», иначе вычисляем корень

```
procedure TForm1.BitBtn18Click(Sender: TObject);
begin
 if StrToFloat(edit1.Text)=0 then
 edit1.Text:= 'Деление на ноль запрещено'
 else
 begin
     i:=StrToFloat(edit1.Text);
   i:=1/i;
     edit1.Text:=FloatToStr(i);
end;
end;
```

Кнопка 1/x

Аналогично проверяем, нет ли в **Edit**-е нуля: если есть, то напоминаем, что на ноль делить нельзя, иначе вычисляем

```
procedure TForm1.BitBtn21Click(Sender: TObject);
begin
  i:=StrToFloat(edit1.Text);
   i:=i*pi/180;
    edit1.Text:=FloatToStr(sin(i));
```

ШАГ 2 Приступим к написанию кода

2. Кнопки действий

```
procedure TForm1.BitBtn21Click(Sender: TObject);
begin
  i:=StrToFloat(edit1.Text);
   i:=i*pi/180;
     edit1.Text:=FloatToStr(sin(i));
end;
```

Кнопка извлечения синуса

- Присваиваем **i** значение **Edit**-а
- Переводим **i** из градусов в радианы (вычисление тригонометрических функций в Паскале происходит в радианах, поэтому перед тем, как считать, надо перевести)
- Отображаем в **Edit**-е значение синуса введенного угла (дополнительно можете обработать ситуацию, когда при вычислении синуса ничего в **Edit** не введено)

```
procedure TForm1.BitBtn22Click(Sender: TObject);
begin
  i:=StrToFloat(edit1.Text);
   i:=i*pi/180;
```

Кнопка извлечения косинуса (думаю здесь все понятно)

ШАГ 2 Приступим к написанию кода

2. Кнопки действий

```
procedure TForm1.BitBtn20Click(Sender: TObject);
begin
  case mode of
1:begin
   p:=StrToFloat(edit1.Text);
   r:=i+p;
   edit1.Text:=FloatToStr(r);
  end;
2:begin
   p:=StrToFloat(edit1.Text);
   r:=i-p;
   edit1.Text:=FloatToStr(r);
  end;
4:begin
   p:=StrToFloat(edit1.Text);
   if p<>0 then
   begin
   r:=i/p;
   edit1.Text:=FloatToStr(r);
  end
   else
    edit1.Text:='Деление на ноль запрещено'
    end;
3:begin
   p:=StrToFloat(edit1.Text);
   r:=i*p;
   edit1.Text:=FloatToStr(r);
  end;
end;
```

Кнопка = (при ее нажатии определяется код действия, затем с помощью **case** – выбора оно выполняется)

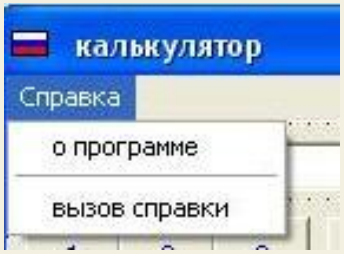
сложение

вычитание

Деление (и опять заметьте обработку ситуации, когда делитель оказывается равен нулю)

умножение

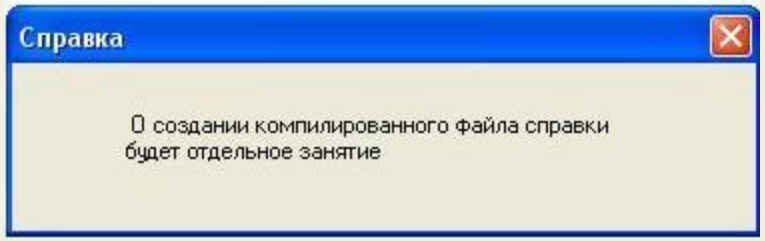
ШАГ 3



Сейчас необходимо создать форму (Form2) для вывода **информации о программе**. Создание компилированного файла справки не входит в этот урок, поэтому создадим Form3, на которой выведем об этом информацию

Форма 2 (О программе)

Форма 3 (Вызов справки)



Поработайте над дизайном форм и не забудьте их «познакомить» , а для их открытия используйте метод **ShowModal**

ШАГ 4

Сохраняем все, компилируем и смотрим результат

Смотрим ->

Итак, мы создали свой простой, но работоспособный калькулятор. Конечно же, у него еще есть много необработанных исключений (глюков) – попробуйте их выявить и исправить.

А пока наш проект закончен, присвоим ему версию 1.000. Понятно, что функциональность программы можно значительно расширить до уровня инженерного калькулятора и это Вам уже под силу – пробуйте !

На этом наш урок закончен

ИТОГИ УРОКА:

На этом уроке мы создали Windows - приложение - свой калькулятор и научились обрабатывать исключительные ситуации

НА СЛЕДУЮЩЕМ УРОКЕ:

ООП на Delphi – 12:

Мы рассмотрим создание тестирующей системы

Домнин Константин Михайлович

E – mail: kdomnin@list.ru

2006 год.