Лабораторная работа №2.

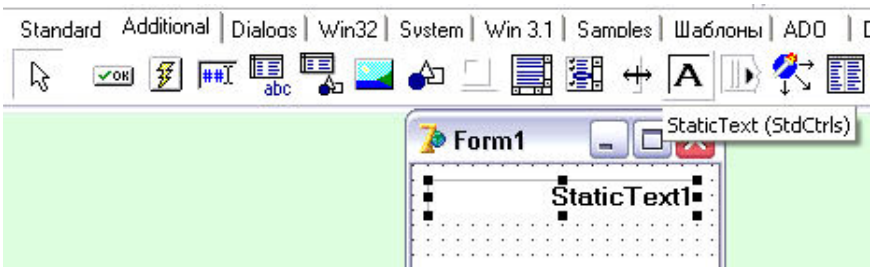
Обработка клавиатурных событий.

Время: 180 мин.
Что нужно освоить:
1) каким образом в Delphi обрабатывать события клавиатуры;
2) чем отличаются процедуры обработки ввода символа и нажатия клавиши клавиатуры;
3) как пользоваться переменной Sender.

ЧАСТЬ I. КЛАВИАТУРНЫЕ СОБЫТИЯ

1. ПРОЦЕДУРА ОБРАБОТКИ ВВОДА СИМВОЛА

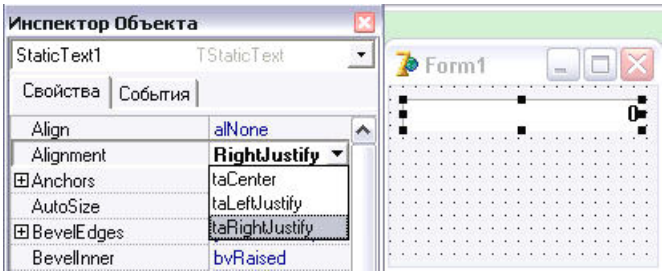
Создайте папку, в которую будете сохранять разрабатываемые приложения. Для каждого приложения следует создавать отдельный каталог. Создайте новое приложение в среде `Delphi`. Разместите на форме компонент `StaticText` с вкладки `Additional`.



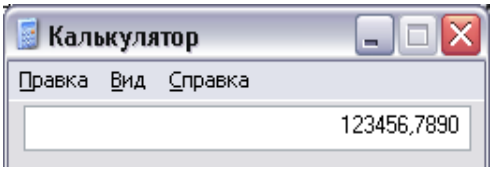
Установите для свойств компонента следующие значения:

```
Alignment     taRightJustify
BorderStyle   sbsSunken
AutoSize      False
Width         150
Caption       0
Font.Size     10
Font.Style    [fsBold]
Color         по усмотрению, один из светлых.
```

Такие настройки позволят сделать `StaticText` похожим на поле `Edit`. Компонент `StaticText` дает нам несколько иные возможности, чем поле Edit. В частности возможность назначить выравнивание содержимого по правому краю.



Обратите внимание на калькулятор `Windows`:

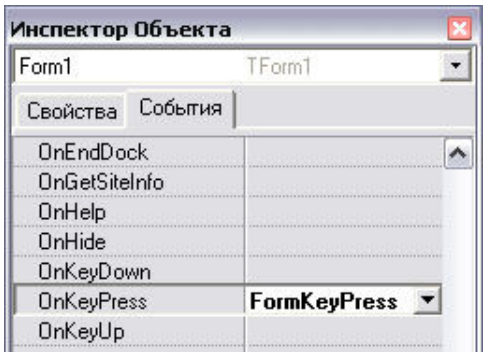


- в поле набора цифр назначено выравнивание по правому краю. Текстовое поле `Edit` не имеет такой возможности.

Однако пользователь не сможет вводить в компонент StaticText символы, как в поле Edit. Организовать ввод числа в StaticText можно как через виртуальные экранные клавиши так и путем нажатия на цифровые клавиши реальной клавиатуры.

Давайте рассмотрим возможности, которые предоставляет нам Инспектор Объектов.

Создайте для Form1 обработчик события **OnKeyPress** дважды кликнув по полю, находящемуся правее позиции OnKeyPress. В результате будет создана заготовка процедуры FormKeyPress. Одним из аргументов процедуры является переменная Key типа Char, которая возвращает символ, введенный с клавиатуры. Добавьте в заготовку процедуры код, который будет к содержимому StaticText1 добавлять символ ‘1’, если с клавиатуры введен символ ‘1’:




```
procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
begin
  if Key='1' then
    StaticText1.Caption:=StaticText1.Caption+'1';
end;
```

Запустите программу и проверьте работоспособность процедуры.

Очевидная неловкость состоит в том, что когда исходное содержимое StaticText1 это ноль, то добавив к нему ‘1’ мы должны получить не ‘01’, а ‘1’. Естественно, что для исключения этой ситуации можно добавить дополнительную проверку:

```
procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
begin
  if Key='1' then
    if StaticText1.Caption='0'
      then StaticText1.Caption:='1'
      else StaticText1.Caption:=StaticText1.Caption+'1';
end;
```

Теперь добавим возможность ввода любой цифры, а не только ‘1’:

```
procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
begin
  if Key in ['0'..'9'] then
    if StaticText1.Caption='0'
      then StaticText1.Caption:=Key
      else StaticText1.Caption:=StaticText1.Caption+Key;
end;
```

Не хватает запятой, таким же образом можно добавить и её:

```
procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key in ['0'..'9']) or (Key=',') then
    if StaticText1.Caption='0'
      then StaticText1.Caption:=Key
      else StaticText1.Caption:=StaticText1.Caption+Key;
end;
```

Проверьте работоспособность такой процедуры и вы без труда обнаружите некоторые нежелательные моменты в работ этой процедуры, а именно: 1) можно ввести несколько запятых, 2) когда начальное значение ноль, то при нажатии на ‘,’ следует получить результат ‘0,’, а не просто ‘,’. Поэтому внесем в процедуру дополнительный анализ:

```
procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
begin
  if Key in ['0'..'9'] then
    if StaticText1.Caption='0'
      then StaticText1.Caption:=Key
      else StaticText1.Caption:=StaticText1.Caption+Key;
  if ((Key=',') and (Pos(Key,StaticText1.Caption)=0))
    then StaticText1.Caption:=StaticText1.Caption+Key;
end;
```

Проверьте работоспособность доработанной процедуры. Всё бы хорошо, но отсутствует возможность внести исправление в случае ошибочного набора. Нужны два варианта: удалить *последний* набранный символ и удалить *всё* число. Давайте добавим в процедуру возможность исправления последнего набранного символа, для этого будем использовать клавишу `Backspace`. Однако на этом пути возникает одно препятствие – а как ввести в анализ символ `Backspace`? Давайте временно основную часть процедуры поместим в комментарии, но добавим строку вывода символа `Backspace`:

```
procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
Begin
  StaticText1.Caption:=Key;
  {if Key in ['0'..'9'] then
    if StaticText1.Caption='0'
      then StaticText1.Caption:=Key
      else StaticText1.Caption:=StaticText1.Caption+Key;
  if ((Key=',') and (Pos(Key,StaticText1.Caption)=0))
    then StaticText1.Caption:=StaticText1.Caption+Key;}
end;
```

Посмотрите на действие процедуры, нажимая различные клавиши – при вводе букв русского и английского алфавитов, цифр – эти символы выводятся в StaticText1 как есть, однако при нажатии на клавишу Backspace мы получаем неприемлемый результат. Обойти это затруднение можно сменив тип данных результата: давайте будем выводить тип данных не Char, а Byte, то есть к переменной Key типа Char применим функцию Ord:

```
procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
Begin
  StaticText1.Caption:=IntToStr(Ord(Key));
  {if Key in ['0'..'9'] then
    if StaticText1.Caption='0'
      then StaticText1.Caption:=Key
      else StaticText1.Caption:=StaticText1.Caption+Key;
  if ((Key=',') and (Pos(Key,StaticText1.Caption)=0))
    then StaticText1.Caption:=StaticText1.Caption+Key;}
end;
```

Поэкспериментируйте с нажатием клавиш и вы увидите коды вводимых символов, в частности по нажатию ‘0’ получите код 48, а по нажатию ‘ ’ (← это пробел ☺) – код 32. *Узнайте* самостоятельно какой код у Backspace.

Итак, теперь добавим в нашу процедуру обработку клавиши Backspace:

```
procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
Begin
  if Key in ['0'..'9'] then
    if StaticText1.Caption='0'
      then StaticText1.Caption:=Key
      else StaticText1.Caption:=StaticText1.Caption+Key;
  if ((Key=',') and (Pos(Key,StaticText1.Caption)=0))
    then StaticText1.Caption:=StaticText1.Caption+Key;
  if Ord(Key)=8 then StaticText1.Caption:=
     copy(StaticText1.Caption,1,length(StaticText1.Caption)-1);
end;
```

Хочу заметить, что для управляющих клавиш предусмотрены константы и условие Ord(Key)=8 допустимо заменить на Ord(Key)=vk_Back. Попробуйте и убедитесь в этом.

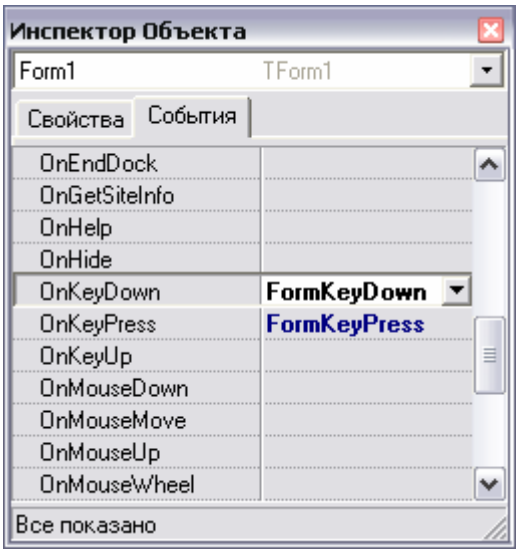
В обработке клавиши Backspace не учтены два момента: 1) корректно перед удалением символа проверить не является ли строка пустой, 2) при удалении последнего символа из строки в StaticText1 следует поместить ‘0’. Учтем их следующим образом:

```
...
if Ord(Key)=vk_Back then
  case length(StaticText1.Caption) of
  0: ;
  1: StaticText1.Caption:='0';
  else
    StaticText1.Caption:=
      copy(StaticText1.Caption,1,length(StaticText1.Caption)-1);
end;
...
```

Теперь аналогичным образом попытайтесь добавить стирание всего содержимого StaticText1 по нажатию клавиши Delete (Помните наши планы: «Нужны два варианта: удалить *последний* набранный символ и удалить *всё* число.»?). Попытка окажется неудачной, так как с помощью процедуры FormKeyPress нельзя проанализировать нажатие таких управляющих клавиш как стрелки, Insert, Delete, функциональных клавиш F1–F12 и т.п. Для преодоления этого препятствия следует использовать иной обработчик – не ввода символа, а нажатия клавиши.

2. ПРОЦЕДУРА ОБРАБОТКИ НАЖАТИЯ КЛАВИШИ

К процедурам обработки нажатия клавиши следует отнести обработчики событий **OnKeyDown** (нажатие клавиши) и **OnKeyUp** (отпускание клавиши).



Создайте для Form1 обработчик события **OnKeyDown** дважды кликнув по полю, находящемуся правее позиции OnKeyDown. В результате будет создана заготовка процедуры FormKeyDown. Одним из аргументов процедуры является переменная Key типа Word (целочисленная, положительная, диапазон: 0..65535), которая возвращает код нажатой клавиши. Вторым аргументом является переменная Shift типа TShiftState. Справка Delphi (загляните и убедитесь сами, это полезно для последующего самостоятельного познания сути иных процедур и их аргументов) гласит, что данный тип данных относится ко множествам:

```
Type TShiftState = set of (ssShift, ssAlt, ssCtrl, ssLeft, ssRight, ssMiddle, ssDouble);
```

В данном случае в идентификаторе типа TShiftState слово Shift не относится именно к клавише Shift, а имеет самостоятельный смысл – перевод слова Shift гласит: изменять, переключать, а к клавиатурным переключателям, в свою очередь, относятся клавиши – Shift, Alt и Ctrl. Именно сочетание нажатых клавиш в данной процедуре возвращает эта переменная Shift. Если не нажата ни одна из клавиш переключателей то значение Shift как множества будет []. Если нажата одна из клавиш, то в множество будет включен её идентификатор, например: [ssAlt]. Если нажаты 2 или 3 клавиши, то в множество будут включены и они, например: [ssAlt,ssCtrl].

Итак, давайте используем возможности данного обработчика события OnKeyDown для обработки нажатия клавиши Delete. Начнем с того, что узнаем какой именно код у клавиши Delete:

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  StaticText1.Caption:=IntToStr(Key);
end;
```

Далее изменим код – создадим непосредственно обработчик события, выполняющий необходимое нам действие:

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Key=46 then StaticText1.Caption:='0';
end;
```

Для изучения возможностей использования переменной `Shift` исследуем способ замены стандартного сочетания клавиш для закрытия приложения `Alt+F4` на клавишу `Esc`. Код клавиши `Esc` равен 27 – соответственно добавим выполнение процедуры `Close` при возвращении переменной `Key` кода 27:

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  case Key of
  46: StaticText1.Caption:='0';
  27: Close;
  end;
end;
```

В данном обработчике присутствует некоторая неточность, так как не исключается нажатие клавиш переключателей совместно с клавишей `Esc`, то есть, например, выход из приложения будет происходить не только при нажатии клавиши `Esc`, но и при нажатии сочетания клавиш `Shift+Esc`. Для исключения присутствия иных клавиш кроме `Esc` прямо укажем об этом в процедуре:

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  case Key of
  46: StaticText1.Caption:='0';
  27: if Shift=[] then Close;
  end;
end;
```

Как вы поняли, суть обработчика дословно звучит так: если множество нажатых клавиш переключателей пусто и нажата клавиша `Esc`, тогда закрой приложение.

Теперь исключим возможность выхода из приложения по сочетанию клавиш `Alt+F4`. Суть обработки состоит в том, чтобы сравнить множество нажатых клавиш переключателей с множеством `[ssAlt]` при нажатой функциональной клавише `F4`. В случае такого сочета-

ния переменной `Key` присвоить код 0, то есть сообщить программе, что никакая клавиша и не нажималась (включая и клавишу `F4`). Код клавиши `F4` узнать несложно, через: «`StaticText1.Caption:=IntToStr(Key);`». Итоговая процедура выглядит изящно и компактно (стремитесь к этому и процесс кодирования будет доставлять вам удовольствие ☺):

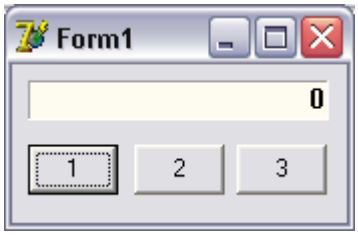
```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  case Key of
  46:  StaticText1.Caption:='0';
  27:  if Shift=[] then Close;
  115: if Shift=[ssAlt] then Key:=0;
  end;
end;
```

Заметьте, что описание всех действий, выполняемых структурным оператором case значительно лаконичнее, чем оно же, но произведенное на русском языке. Попробуйте сравнить описания по количеству использованных символов, включая пробелы и знаки пунктуации. Кстати, никто вам не мешает изменять имена компонентов на более компактные, например, заменить `StaticText1` на `ST`. В данном тексте они оставлены по умолчанию исключительно в учебных целях, для лучшего понимания и наглядности.

Итак, к настоящему моменту вы уже должны понимать разницу между обработкой введенного символа и нажатой клавиши. Однако для полноценной обработки всех возможных событий приложения этого недостаточно.

3. ПРИОРИТЕТ ОБРАБОТКИ НАЖАТИЯ КЛАВИШИ

Начнем с демонстрации неудачи. Добавьте на форму три кнопки, примерно так:



Теперь запустите приложение и вы убедитесь, что все, что было сделано ранее уже не работает. В чем же дело? А в том, что эти экранные клавиши берут на себя фокус активности. Кстати, между клавишами можно перемещать фокус нажатием клавиши `Tab` или стрелочками. Теперь создайте процедуры обработки нажатия этих экранных клавиш:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if StaticText1.Caption='0'
    then StaticText1.Caption:='1'
    else StaticText1.Caption:=StaticText1.Caption+'1';
end;
```

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  if StaticText1.Caption='0'
    then StaticText1.Caption:='2'
    else StaticText1.Caption:=StaticText1.Caption+'2';
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
  if StaticText1.Caption='0'
    then StaticText1.Caption:='3'
    else StaticText1.Caption:=StaticText1.Caption+'3';
end;
```

Запустите программу и убедитесь, что эти обработчики можно инициализировать кликом мыши по соответствующей экранной клавише или нажатием на клавишу ПРОБЕЛ при заблаговременно установленном фокусе на экранной клавише.

У каждого из компонентов есть свои обработчики событий и у экранных клавиш `Button` тоже. Посмотрите в Инспекторе Объектов на возможные события для `Button1`. Пока только есть обработчик события при клике мышкой по клавише – `Button1Click`, а обработчик события `OnKeyDown` пока не назначен. Поэтому ничего и не происходит при нажатии таких клавиш как `Esc` или `Delete`, даже при наличии фокуса на `Button1`. Давайте назначим обработчик для `Button1`:

```
procedure TForm1.Button1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  case Key of
  46:  StaticText1.Caption:='0';
  27:  if Shift=[] then Close;
  115: if Shift=[ssAlt] then Key:=0;
  end;
end;
```

При наличии фокуса на `Button1` обработчик нажатия клавиш будет работать как и ранее, когда экранных клавиш на форме ещё не было. То есть один из возможных способов это сделать копии всех процедур обработки нажатия клавиш для всех компонентов (для `Button1`, `Button2`, `Button` 3 и т.д.). Но такой подход крайне неудобен, особенно, при значительном количестве компонентов.

Кроме того, в калькуляторе `Windows` ни одна из клавиш не выделена (не имеет фокуса) и нам этого не нужно. Отключить это можно в Инспекторе Объектов. Обратите внимание на свойства кнопок `TabStop` типа `Boolean` и `TabOrder` целочисленного типа. Свойство `TabStop`, равное `true`, означает, что на данном компоненте можно установить фокус. Свойство `TabOrder` позволяет назначить очередность (между компонентами) установки фокуса в работающем приложении последовательным переключением клавишей `Tab`. Нам фокус на клавишах ни к чему, поэтому для всех клавиш установите свойство `TabStop` равное `false`.

После выключения свойства `TabStop` отключается и возможность инициализировать созданные ранее обработчики нажатием клавиши ПРОБЕЛ, так как отсутствует возможность установить фокус на экранной клавише. Но нам этого и не нужно.

Правильный подход состоит в использовании встроенных возможностей Delphi, в частности, события OnShortCut.

Внимание: в данной работе не обсуждается использование компонента HotKey с вкладки Win32 и назначение с помощью него «горячих клавиш» – комбинаций клавиш, на которые может реагировать приложение, даже если оно не имеет фокуса или запущено в трее. В данной работе обсуждается только непосредственная привязка обработчиков к событию: нажатие клавиши.

Дословно ShortCut можно перевести как кратчайший путь. Событие OnShortCut можно найти в Инспекторе Объектов только для формы, но не для компонента Button. Убедитесь в этом сами. Событие OnShortCut характерно только для формы приложения и возникает при нажатии на клавишу, но до инициализации события KeyPress или KeyDown формы или компонентов. Создайте заготовку процедуры обработки события OnShortCut дважды кликнув на соответствующей позиции в Инспекторе Объектов:

```
procedure TForm1.FormShortCut(var Msg: TWMKey; var Handled:
Boolean);
begin

end;
```

В процедуре используются два аргумента: Msg (сообщение) и Handled (глагол: обработать).

Переменная Msg типа record имеет несколько полей, но нас будет интересовать только поле CharCode, которое и возвращает код нажатой клавиши. Именно *нажатой клавиши*, а **не** *введенного символа*. Это, например, имеет значение при вводе символа ‘1’ с основной клавиатуры или дополнительного блока цифровой клавиатуры (физически клавиши разные, а вводимый символ один).

Переменная Handled типа Boolean позволяет управлять последующей (после процедуры FormShortCut) обработкой нажатия клавиши. Если эта переменная переведена в состояние True, то даже при наличии обработчиков событий OnKeyPress, OnKeyDown и/или OnKeyUp, иных действий кроме описанных в процедуре FormShortCut производиться не будет.

Проведем эксперимент.

Для начала уточним текущее состояние программы.

Выход из программы по нажатию клавиши Esc описан в процедуре FormKeyDown и в процедуре Button1KeyDown. Однако процедура FormKeyDown не действует, так как события «нажатие клавиши» перехватывается тем компонентом из списка Button1, Button2 и Button3, у которого находится фокус. Если у Button1, то выход по Esc возможен, так как Button1KeyDown содержит такое действие, если же у Button2 или у Button3, то выход по Esc не возможен, так как у этих компонентов обработчик события ButtonKeyDown не описан.

Что мы хотим получить?

При наличии фокуса на любом из компонентов формы обработчик нажатия клавиш должен обеспечить:

1) выход из программы по нажатию клавиши Esc;

2) ввод цифр и запятой в StaticText1;

3) обнуление StaticText1 клавишей Delete и стирание последнего введенного символа клавишей Backspace.

Будем решать задачу последовательно с проверкой результатов работы. Добавьте в процедуру FormShortCut один структурный оператор case:

```
procedure TForm1.FormShortCut(var Msg: TWMKey; var Handled:
Boolean);
begin
  case Msg.CharCode of
  27: close;
  end;
end;
```

Проверьте его работоспособность и, если работает, то «убейте» процедуру `FormKeyDown` – она нам больше не нужна.

Правильно избавиться от процедуры можно следующим образом: 1) выделите все её содержимое, находящееся внутри операторных скобок `begin end` и удалите, 2) нажмите `Ctrl+F9` – произойдет проверка и компиляция кода без запуска приложения, при этом удалится сам шаблон процедуры, её описание в разделе `type TForm1 = class(TForm)`, а также ссылка на неё в Инспекторе Объектов. Чтобы не загромождать программу процедурами удалите и все остальные процедуры описанным выше способом за исключением, конечно, процедуры `FormShortCut`. Все необходимые действия поместим именно в неё.

Итак в модуле у вас только одна процедура – `FormShortCut`, добавим в неё ввод запятой в компонент `StaticText1`. Для этого допишем в структурный оператор `case` соответствующий код:

```
procedure TForm1.FormShortCut(var Msg: TWMKey; var Handled:
Boolean);
begin
  case Msg.CharCode of
  27: close;
  110,188,191:
      if Pos(',',StaticText1.Caption)=0
        then StaticText1.Caption:=StaticText1.Caption+',';
  end;
end;
```

Здесь коды 110, 188 и 191 соответствуют различным вариантам ввода символа «запятая». Убедитесь в этом сами проанализировав значение `Msg.CharCode` при нажатии различных клавиш.

Теперь добавим к структурному оператору `case` еще и возможность ввода цифр в `StaticText1`:

```
...
48..57,96..105:
    begin
      if Msg.CharCode>95 then d:=48 else d:=0;
      if StaticText1.Caption='0'
        then StaticText1.Caption:=chr(Msg.CharCode-d)
        else StaticText1.Caption:=StaticText1.Caption+chr(Msg.CharCode-d);
    end;
...
```

Диапазон 48..57 означает обработку нажатия цифровых клавиш на основной клавиатуре, а – 96..105 – на дополнительной (цифровой). Переменная `d` типа `byte` (которую, кстати, необходимо объявить в разделе `var` процедуры `FormShortCut`), позволяет обойтись одним обработчиком нажатия клавиши с цифрой, смещая код на 48 в случае, если нажата цифра на дополнительной клавиатуре.

Для правильной работы процедуры следует также добавить в неё код: «`Handled:=true;`». Он означает, что обработка нажатия клавиши уже произведена и никакими иными процедурами нажатие более обрабатывать не следует.

Добавим в тот же структурный оператор обработку клавиш `Backspace` и `Delete` и, в итоге, вместо множества отдельных процедур имеем одну, отвечающую за все предусмотренные нами функции :

```
procedure TForm1.FormShortCut(var Msg: TWMKey; var Handled: Boolean);
var d: byte;
begin
  case Msg.CharCode of
  8: case length(StaticText1.Caption) of
     0: ;
     1: StaticText1.Caption:='0';
     else
       StaticText1.Caption:=copy(StaticText1.Caption,1,length(
         StaticText1.Caption)-1);
  end;
  46: StaticText1.Caption:='0';
  27: close;
  110,188,191:
      if Pos(',',StaticText1.Caption)=0
        then StaticText1.Caption:=StaticText1.Caption+',';
  48..57,96..105:
      begin
        if Msg.CharCode>95 then d:=48 else d:=0;
        if StaticText1.Caption='0'
          then StaticText1.Caption:=chr(Msg.CharCode-d)
          else StaticText1.Caption:=StaticText1.Caption+chr(Msg.CharCode-d);
      end;
  end;
  Handled:=true;
end;
```

ЧАСТЬ II. ПЕРЕМЕННАЯ SENDER.

А как же экранные клавиши? Они есть, но ничего не обрабатывают. Напомню, что раньше мы привязывали к ним ввод соответствующей цифры. Теперь у нас работает ввод цифр с клавиатуры, но процедуры ввода цифр по нажатию мышкой на экранные клавиши мы «убили». И сделали это не зря. Дело в том, что ранее мы шли экстенсивным путем – на каждую экранную клавишу писали отдельную процедуру. Но программный код в них по существу повторялся. Сейчас мы исследуем возможности переменной `Sender` для оптимизации программного кода на примере создания заготовки для программы «Калькулятор».

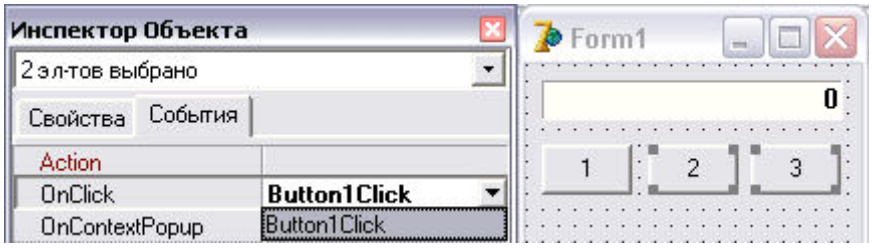
Итак необходимо разместить на форме нужные клавиши, назначение которых вводить цифры в компонент `StaticText1`. У нас на форме уже есть три кнопки, рассмотрим общий принцип использования переменной Sender на примере этих трех кнопок.

Для начала создайте процедуру `Button1Click` (кликнув дважды по самой клавише или через инспектор объектов) и заполните ее следующим образом:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if StaticText1.Caption='0'
     then StaticText1.Caption:='1'
     else StaticText1.Caption:=StaticText1.Caption+'1';
end;
```

Проверьте работоспособность этой процедуры – очевидно, что она обеспечивает ввод цифры 1 при нажатии на экранную клавишу Button1.

Пока у экранных клавиш Button2 и Button3 нет аналогичных обработчиков. Давайте не будем для них создавать новых обработчиков, а назначим им уже существующий. Для этого выделите обе клавиши, кликнув один раз сначала по одной из них (для того чтобы выделить её) и затем при нажатой клавише Shift кликните один раз по другой. Теперь можно менять свойства или назначать обработчики событий сразу для обеих клавиш. Перейдите в Инспектор Объектов на вкладку События (Events) и кликните один раз на поле правее OnClick – активируется выпадающий список:



Там для выбора будет только один созданный нами обработчик Button1Click, его и следует выбрать. То есть для всех трех клавиш мы назначили один обработчик. Испытаем программу – очевидно, что при клике на любую из трех клавиш в StaticText1 будет добавляться одна и та же цифра – 1. При этом клавиатурный обработчик всех ранее созданных нами функций остается в работоспособном состоянии (Backspace, Delete и др.). То есть нам осталось только добавить анализатор нажатой экранной клавиши и каким-то образом передать эту информацию в процедуру Button1Click. Вот тут-то нам и пригодиться переменная Sender, кстати, в переводе на русский язык это слово означает отправитель. Отправителем в случае работы приложения является тот компонент, который и вызвал срабатывание данной процедуры. У нас это могут быть клавиши: Button1, Button2 и Button3. С учетом этих знаний доработаем процедуру:

```
procedure TForm1.Button1Click(Sender: TObject);
var c: char;
begin
  if Sender=Button1 then c:='1';
  if Sender=Button2 then c:='2';
  if Sender=Button3 then c:='3';
  if StaticText1.Caption='0'
     then StaticText1.Caption:=c
     else StaticText1.Caption:=StaticText1.Caption+c;
end;
```

Проверьте работоспособность процедуры, теперь ввод цифр происходит в соответствии с нажатой экранной клавишей. Однако сам код не выглядит оптимальным и универсальным, ведь компонентов на форме может быть существенно больше чем три.

Есть другой способ использования переменной Sender. Можно непосредственно через Sender обратиться к свойствам компонента (Name, Caption, Tag и др.) через конструкцию вида: Sender as TButton. Изменим процедуру:

```
procedure TForm1.Button1Click(Sender: TObject);
var c: char;
begin
  if (Sender as TButton).Name='Button1' then c:='1';
  if (Sender as TButton).Name='Button2' then c:='2';
  if (Sender as TButton).Name='Button3' then c:='3';
  if StaticText1.Caption='0'
     then StaticText1.Caption:=c
     else StaticText1.Caption:=StaticText1.Caption+c;
end;
```

Пока еще выглядит не оптимально, но есть уже намеки на лучший исход: конструкция `(Sender as TButton).Name` одинакова во всех условных операторах, но возвращает разные значения соответствующие свойству `Name` нажатой экранной клавиши. Воспользуемся этим так:

```
procedure TForm1.Button1Click(Sender: TObject);
var c: char;
begin
  c:=(Sender as TButton).Name[7];
  if StaticText1.Caption='0'
     then StaticText1.Caption:=c
     else StaticText1.Caption:=StaticText1.Caption+c;
end;
```

Обратите внимание, что свойство `Name` имеет строковый тип, поэтому с ним можно обращаться как со строкой. Строка это массив символов и в процедуре мы анализируем седьмой символ, который и несет номер клавиши. Можно, конечно, так не усложнять и взять номер прямо из свойства `Caption`, ведь у нас прямо на кнопках написаны их номера:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if StaticText1.Caption='0'
     then StaticText1.Caption:=(Sender as TButton).Caption
     else StaticText1.Caption:=StaticText1.Caption+(Sender as TButton).Caption;
end;
```

Или пойти несколько иным путем:

```
procedure TForm1.Button1Click(Sender: TObject);
var s: TButton;
begin
  s:=(Sender as TButton);
  if StaticText1.Caption='0'
     then StaticText1.Caption:=s.Caption
     else StaticText1.Caption:=StaticText1.Caption+s.Caption;
end;
```

Таким образом вы создали две процедуры, обеспечивающие работу калькулятора:

1) процедура обработки клавиатурных событий;

2) процедура обработки нажатия экранных клавиш.

На форме калькулятора из управляющих элементов присутствуют только кнопки, поэтому данного подхода к обработке переменной `Sender` будет достаточно. Однако этим не исчерпываются все возможности по оптимизации программного кода с использованием этой переменной. А вам предстоит доработать созданный ранее калькулятор с учетом полученных знаний об обработке клавиатурных событий и переменной `Sender`…