Инструкционная карта № 5.

Тема: «Приложение калькулятор»

Построим простейший калькулятор. Он не будет самым мощным из существующих, но на его примере мы разберем несколько новых приемов практического программирования.

|  |  |
| --- | --- |
|  | Создайте новый проект с формой, сохраните его в новой папке *Калькулятор* под именем Calc.bdsproj. Установите для формы заголовок «*Calc*» и размер шрифта 10 пт. |
|  | Разместите на форме одно поле ввода, дайте ему имя Disp и очистите свойство Text. |

Для калькулятора мы будем использовать еще один вид кнопок — TSpeedButton (группа *Additional*). Их отличие от уже рассмотренных ранее классов TButton и TBitBtn состоит в том, что они не получают фокуса ввода клавиатуры. Это значит, что их нельзя сделать активными, переходя от элемента к элементу с помощью клавиши **Tab**, и вообще их можно привести в действие только мышкой. Таким образом, на форме будет всего один компонент, принимающий сигналы с клавиатуры — это поле ввода.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Для того, чтобы настраивать свойства сразу нескольких компонентов, нужно выделить их все при нажатой клавише **Shift** и после этого изменять свойства в окне *Object Inspector*. | |
|  | | | Добавьте на форму 12 кнопок TSpeedButton для цифр, арифметических действий, знака «равно» и операции «сброс». Установите для всех кнопок размеры 25 на 25 пикселей и разместите их так, как на образце. |

|  |  |
| --- | --- |
|  | Дайте кнопкам-действиям имена btnPlus, btnMinus, btnMul, btnDiv (сложение, вычитание, умножение и деление), а кнопке «равно» — имя btnCalc. |
|  | Для действий и кнопки «*С*» установите жирный шрифт, а для кнопки «*С*» дополнительно — красный цвет шрифта. |

**Ввод данных**

Кнопка «*С*» должна просто стирать содержимое поля ввода Disp. То есть, нужно вызвать метод Disp.Clear.

|  |  |
| --- | --- |
|  | Добавьте обработчик события OnClick для кнопки «*С*». |

Понятно, что когда пользователь щелкнул по кнопке-цифре, нужно добавить эту цифру в конец текста поля ввода. Конечно, можно для каждой кнопки написать обработчик типа

Disp.Text := Disp.Text + '1';

Однако можно сделать и более грамотно, определив для всех кнопок общий обработчик. Заметим, что цифра, которую нужно добавить, это и есть текст кнопки, ее свойство Caption. Таким образом, когда мы назначим всем кнопкам один единственный обработчик, в нем нужно как-то определить, какую именно кнопку мы нажали. А такая возможность есть, ведь обработчик имеет параметр Sender — это адрес компонента, от которого пришло сообщение!

|  |  |
| --- | --- |
|  | Выделите все кнопки-цифры и определите для них общий обработчик события onClick:  Disp.Text := Disp.Text + Sender.Caption;  Попытайтесь запустить программу. |

Скорее всего, вы получили сообщение об ошибке:



Фраза *Indecrared identifier* означает *Необъявленное имя*, то есть, по мнению *Delphi* объект Sender не имеет свойства Caption. Как же так?

Обратим внимание на то, что параметр Sender объявлен в заголовке процедуры как переменная типа TObject, а объект TObject действительно не имеет свойства Caption (это можно проверить по справочной системе).

Но мы знаем, что в нашей программе все компоненты, которые могут вызвать этот обработчик, относятся к типу TSpeedButton, у которого свойство Caption есть. Поэтому можно сказать, что программа должна воспринимать параметр Sender не как просто объект, а как TSpeedButton:

Disp.Text := Disp.Text + TSpeedButton(Sender).Caption;

|  |  |
| --- | --- |
|  | Исправьте код и запустите программу. Проверьте, как она работает при вводе текста и букв с клавиатуры и с помощью мыши. Что вам не понравилось? |

Во-первых, вы увидели, что с клавиатуры можно ввести буквы, которые нам совсем не нужны. Вы уже знаете, как решить эту проблему — нужно поставить обработчик события OnKeyPress для поля ввода, как в предыдущей программе.

|  |  |
| --- | --- |
|  | Заблокируйте ввод неверных символов с клавиатуры. Проверьте работу программы. |

Второй недостаток, который вы могли заметить — при щелчке по кнопке курсор сразу перемещается в начало числа, а хотелось бы, чтобы он оставался справа.

В принципе, можно совсем отключить курсор, но это не так легко. Профессиональные решения этой задачи можно [найти в Интернете](http://forum.vingrad.ru/faq/s/3b74c9d5c506f6a594c8cc2bd32866cb/topic-157362/anchor-entry1162120/0.html).

Мы сделаем так, чтобы при вводе очередной цифры с помощью кнопки курсор автоматически переводился в конец числа. У компонента TEdit есть свойства SelStart и SelLength, которые определяют начало и длину выделенной части (она обычно обозначается синим фоном). Если SelLength=0 (ничего не выделено), тогда SelStart указывает на позицию курсора в поле ввода. Если установить это свойство равным длине текста, курсор встанет сразу за последним символом.

|  |  |
| --- | --- |
|  | Добавьте в конец обработчика события OnClick код  Disp.SelStart := Length(Disp.Text);  Проверьте работу программы. |

**Вычисления**

Теперь нужно организовать вычисления. Нам нужны две переменных для хранения чисел и одна символьная переменная, в которую будем записывать тип операции. Поскольку при расчетах могут получиться числа с дробной частью, для хранения чисел будем использовать вещественные (а не целые!) переменные.

В простейшем случае для хранения операции можно использовать переменную типа char (один символ), но мы объявим ее как символьную строку, так как при доработке программы могут понадобиться и многосимвольные названия операций, например, *sin*.

|  |  |
| --- | --- |
|  | Объявите в начале программы две вещественных переменные x1 и x2 типа double и одну символьную строку oper. |

Когда мы нажимаем на одну из кнопок-операций, нужно запомнить введенное число в переменной x1 и тип операции в переменной oper. Тип операции (надпись на кнопке) легко узнать, обратившись к свойству Caption. Поэтому можно установить для всех кнопок-операций один обработчик.

Хочется написать так:

x1 := StrToFloat(Disp.Text);

oper := Sender.Caption;

Однако вторая строчка вызовет ошибку, ее причины мы только что выяснили.

|  |  |
| --- | --- |
|  | Выделите все кнопки-операции и создайте для них один обработчик события OnClick, исправив ошибку в приведенном выше коде. |

При нажатии на кнопку «равно» нужно прочитать из поля ввода второе число и выполнить операцию. При этом первое число и тип операции уже должны находиться в переменных x1 и oper. Поскольку в результате деления может получиться число с дробной частью, переменная для хранения результата (назовем ее res) тоже должна быть вещественной.

|  |  |
| --- | --- |
|  | Введите обработчик события OnClick для кнопки «равно»:  var res: double;  begin  x2 := StrToFloat(Disp.Text);  if oper = '+' then res := x1 + x2;  if oper = '-' then res := x1 - x2;  if oper = '\*' then res := x1 \* x2;  if oper = '/' then res := x1 / x2;  Disp.Text := FloatToStr(res);  end;  Запустите программу и проверьте ее работу. Учтите, что для ввода второго числа нужно сначала очистить экран кнопкой «С» (позже мы исправим это неудобство). |

**Совершенствование**

Теперь проверим «защиту от дурака».

|  |  |
| --- | --- |
|  | Закройте окно программы и затем запустите ее заново. Введите какое-нибудь число, а затем сразу щелкните по кнопке «равно». Что получилось? Попытайтесь объяснить этот эффект, учитывая, что при создании глобальной символьной переменной oper в нее записана пустая строка. |

Наверняка вы догадались, что проблема вызвана тем, что мы еще не задали операцию, а уже попытались что-то вычислить. При этом ни один условный оператор не сработал, и значение переменной res не изменялось. А что же в ней было? Откуда взялось это число?

Переменная res объявлена в процедуре, то есть, она *локальная*. Память для локальных переменных выделяется в стеке при каждом новом вызове процедуры, причем эти ячейки не обнуляются. Поэтому в переменной res в нашем последнем эксперименте остался «мусор» — постороннее значение, которое программа и вывела на экран.

Если вы были очень внимательны, можно было заметить, что при трансляции программы в окне *Message* (в нижнем левом углу) было выдано предупреждение:



Это значит «Переменной res, возможно, не будет присвоено никакого значения». Формально это не ошибка, и программа может запуститься, однако к предупреждениям нужно относиться внимательно, потому что они могут указать на скрытые логические ошибки (ошибки в алгоритме), как в нашем случае.

Какой же выход? Проще всего сделать так: если в переменной oper записана пустая строка, мы просто выйдем из процедуры с помощью оператора Exit.

|  |  |
| --- | --- |
|  | Добавьте в начало обработчика события OnClick для кнопки «равно» строчку  if oper = '' then Exit;  Проверьте работу программы. Что вам не нравится? |

Конечно, очень неудобно, что перед вводом второго числа нужно очищать поле ввода, нажимая на кнопку «С». Хотелось бы делать это автоматически.

|  |  |
| --- | --- |
|  | Запустите стандартную программу *Калькулятор* и посмотрите, в какой момент стирается первое число. |

Наверное, вы увидели, что число из поля ввода автоматически стирается, когда после нажатия на кнопки-действия или кнопку «равно» пользователь набирает новое число. Мы введем логическую переменную newNumber, которой будем присваивать значение True в том случае, если нужно начинать вводить новое число.

|  |  |
| --- | --- |
|  | Объявите глобальную логическую переменную newNumber. В конце обработчиков события OnClick для кнопок-действий и кнопки «равно» добавьте строку  newNumber := True; |

Очистку экрана будем делать перед вводом нового числа. Во-первых, дополним обработчики кнопок-цифр.

|  |  |
| --- | --- |
|  | В начало обработчика события OnClick для кнопок-цифр добавьте код  if nextNumber then begin  Disp.Clear;  nextNumber := False;  end; |

Кроме того, нужно учесть, что пользователь может набирать число на клавиатуре. Поэтому нужно дополнить аналогичным кодом обработчик события OnKeyPress для поля ввода. Теперь он должен выглядеть так:

if not (Key in ['0'..'9',#8]) then Key := #0

else

if nextNumber then begin

Disp.Clear;

nextNumber := False;

end;

|  |  |
| --- | --- |
|  | Измените обработчик события OnKeyPress так, как показано выше. Запустите программу и проверьте ее. Можно ли ввести знаки сложения, вычитания, умножения и деления с клавиатуры? Выполняется ли действие при нажатии на клавишу **Enter**? |

Наверное, вы увидели, что пока программа позволяет вводить с клавиатуры только цифры, но не коды действий. Чтобы исправить этот недостаток, мы дополним обработчик OnKeyPress.

Что нужно сделать, когда нажали символ + на клавиатуре? Проще всего *имитировать* щелчок по кнопке, то есть, вызвать процедуру-обработчик события OnClick для кнопок-действий. Предположим, что этот обработчик называется btnDivClick (у вас он может называться по-другому). Но этой функции нужно передать параметр — адрес объекта, пославшего сообщение. Этот объект— кнопка btnPlus, поэтому при нажатии клавиши + нужно выполнить команду

btnDivClick ( btnPlus );

Обработка нажатия других клавишей строится аналогично. При нажатии клавиши **Enter**, имеющей код 13 (он обозначается в программе как #13), нужно вызвать обработчик события OnClick для кнопки «равно» (она должна называться btnCalc).

|  |  |
| --- | --- |
|  | Измените код обработчика OnKeyPress следующим образом:  if not (Key in ['0'..'9',#8]) then begin  case Key of  '+': btnDivClick(btnPlus);  '-': btnDivClick(btnMinus);  '\*': btnDivClick(btnMul);  '/': btnDivClick(btnDiv);  #13: btnCalcClick(btnCalc);  end;  Key := #0;  end  else  if nextNumber then begin  Disp.Clear;  nextNumber := False;  end;  Запустите программу и проверьте ее работу. После этого закройте проект. |

Совершенствовать даже такую простую программу можно бесконечно. Например, вы можете добавить возможность ввода дробных чисел, новые операции и т.д.