Калькулятор на VB - это легко!

(автор: Есин И. В., email: tofriend@list.ru)

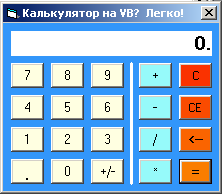
**О чём ...**

Статья имеет две редакции: для начинающих осваивать язык и для тех у кого не только аббревиатура VB не вызывает вопросов, но и более-менее владеющих им.

В данной редакции от читателя требуется знание VB на достаточно хорошем уровне, что означает владение навыками визуального программирования, событийно-управляемого программирования и конечно «основных» операторов языка. Предлагается один из способов поиска решения подобных задач.

**Первый шаг или интерфейс пользователя**

Здесь особо ничего фантазировать не надо, а просто вспомнить, как выглядит обычный калькулятор, какие кнопки у него и каково их назначение. У меня получилось так:



Инструмент **CommandButton** содержит такие интересующие нас свойство и событие, которые дают нам полную ясность его выбора. Аналогичное можно сказать и про инструмент **Label**. Декоративный инструмент **Line** необязателен.

Для понимания моего программного кода вам надо будет согласиться со следующими именами представленных объектов и их свойствами:

1. Форме присвоим имя **Calculator**.
2. Кнопки с цифрами образуют массив кнопок **Knopki** индексы, которых совпадают с их надписями.
3. Кнопка точка - это объект с именем **Tochka**, a кнопка изменения знака числа [+/-] - **Plusmin**.
4. Кнопки арифметических операций имеют общее имя **Operacia**, но проиндексированы. Причём кнопка с надписью + имеет индекс 4, кнопка со знаком минус - индекс 2, кнопка "разделить" - 0, а кнопка соответствующая умножению - 5.
5. Кнопка "сброс" [С] - **Sbros**.
6. Кнопка "очистить" [CE] - **Ochistka**.
7. Кнопка "удалить" [<-] **Udalit.**
8. Кнопка "равно" [=] - **Ravno.**
9. Единственная метка называется **Tablo** (и далее - индикатор отображения вычислений будем называть "табло").

В соответствие с рисунком измените свойство **Caption** всех объектов. Так, например, для объекта **Tablo** его значением будет 0.и свойство **Font**: FontSize = 14. Для всех кнопок свойство **Style** имеет значение 1-Graphical. Свойство **TabStop** всех объектов – False.

Свойство формы **KeyPreview** обязано иметь значение True. Ведь надо же пользователю предоставить возможность вычислять с клавиатуры!

А **BorderStyle -**1-FixedSingle.

Немалая часть проекта уже завершена ... Продолжим работу!

**Второй шаг или "как заставить эту штуковину работать?!!"**

Прежде чем писать обработчики событий для данных элементов управления "возьмём в руки" обычный калькулятор и выясним "механизм" его работы. И сразу же можем сказать, что у него три состояния: включён, выключен и первоначальная готовность. Все эти состояния считайте уже написаны (они обеспечиваются открытием и закрытием окна приложения и значением свойства **Caption** метки **Tablo**)!

Далее, выясним какие действия может осуществлять пользователь над кнопками нашей вещицы и что же при этом происходит на табло?

а) Нажатие на одну из цифр - на табло справа к уже имеющемуся числу прибавляется нажатая цифра. Однако если данное число ноль, его заменяет введённая цифра.

б) Нажатие точки - изменений на табло нет.

в) Нажатие плюс/минус - на табло слева к уже имеющемуся числу прибавляется минус или убирается.

г) Выбор одной из арифметических операций - возможное вычисление значения предыдущей операции с отображением его на табло.

д) Вычислить (кнопка равно) - на табло может отобразиться результат заданной операции.

е) Удаление "последней" цифры числа - на табло удаляется "последняя" цифра.

ж) Очистить (кнопка CE) - на табло "исчезает" набранное число и "превращается" в ноль!

з) Сброс - на табло ноль.

А связаны ли как-то между собой эти действия или с какими-нибудь состояниями? И ... когда возможно данное действие, а когда нет? Ответим в той же последовательности.

а) Если была нажата точка, то набирается дробное число. Очевидно, мы не можем ввести число, которое не умещается в табло. И нам надо учитывать, что число не обязано быть целым (то есть не всегда число оканчивается точкой).

б) Нет согласованных действий. Нельзя поставить точку, если число уже дробное.

в) Нет согласованных действий. Действие можно выполнить всегда!

г) Вычисление значения заданной операции. Действие можно выполнить всегда.

д) Нет согласованных. Действие можно выполнить всегда, если значение принадлежит области допустимых значений для данной арифметической операции

е) Нет согласованных. Нельзя выполнить, если на табло ноль.

ж) Нет согласованных. Действие выполнимо всегда!

з) Состояние первоначальной готовности. Действие выполнимо всегда!

Заметим также, что включение совпадает с первоначальным состоянием нашей счётной машинки.

Итак ... определили не только какие действия может выполнить пользователь калькулятора и что при этом отобразится на табло, но и выявили некоторую взаимосвязь между действиями и даже одним состоянием. Остаётся "рассказать" про это всё компьютеру. Но каким образом? В нашем распоряжении VB!

**Третий шаг или приближение к программному коду!**

"Увидим" же программный код! При определении чего нам надо, будем исходить только из того какие элементы управления у пользователя "в руках", зачем ему нужен калькулятор и тех выкладок, которые мы с лёгкостью проделали в предыдущем шаге! А потребуется нам из того богатого ассортимента возможностей, который предоставляет VB всего-то несколько операторов, функций, свойств и событий известных объектов да понять какие управляющие переменные создать, каков их тип.

Как калькулятор поймет, что его только-что включили или был произведён сброс в первоначальное состояние? (з) И как он узнает какое действие ему задал пользователь? (г) Понятно, что без переменной здесь не обойтись! Поэтому в перменной **mode** типа *String*, будем хранить данные режима вычислений.

А каким образом калькулятор "запомнит" слагаемое, множитель, делимое, вычитаемое (то есть "первое" число операции)? Для этого можно объявить глобальную перменную **pervoe**.

На вопрос нужны ли ещё переменные отвечать пока воздержимся.

*Вначале была точка.*

Поскольку представление целого числа на табло оканчивается точкой, то будем "её ставить" в процедуре обработке события (далее - процедуре) **Change** метки **Tablo**:

Private Sub Tablo\_Change()

Dim z As Byte, t As Byte

z = InStr(1, Tablo.Caption, ",") ' если результат - дробное число ищем позицию запятой

t = InStr(1, Tablo.Caption, ".") ' определяем позицию точки

If t = 0 Then

' если точка не найдётся, то ставим её в конце:

Tablo.Caption = Tablo.Caption & "."

ElseIf z <> 0 Then

' если событие наступает, в результате "выброса" на табло значения операции, которое является дробным числом, то запятую меняем на точку:

Tablo.Caption = Mid(Tablo.Caption, 1, z - 1) & "." & Mid(Tablo.Caption, z + 1, Len(Tablo.Caption) - z - 1)

End If

Здесь, как видите, учитывается тот случай, когда результатом операции является дробное число или уже имеется дробное число.

*Учимся писать ... числа.*

Теперь наступило время "набрать" нашему пользователю число! Позволим ему сделать это ... но пока с помощью мыши. При этом предусмотрим следующее: 1) он вводит цифру, когда на табло ноль.

2) он не знает (или может забыть) сколько цифр может поместиться в табло.

3) он вводит "второе" число операции, либо число после нажатия кнопки равно. 4) он вводит цифру, после того как щёлкнул по "точке".

Рассмотрение первой и второй особенности не представляет особого интереса, поскольку это нетрудно проверяется, сравнивая значение табло с "нулём" и количеством допустимых символов соответственно.

Второе число вводится после указания желаемой арифметической операции, где в соответствующей процедуре укажем значение "внутренней" переменной **Tag** кнопки **Ravno** значение "второе". И как только введена первая цифра "второго" числа, данная переменная становится пустым символом.

Для реализации режима ввода дробного числа воспользуемся также свойством **Tag**, но уже объекта **Tablo**, так что если его значение равно "не целое", то слева от введённой цифры будет ноль.

Данные выкладки можно реализовать в виде такого программного кода:

Private Sub knopka\_Click(Index As Integer)

' если были выбраны равно или операция - ввод нового числа:

If Ravno.Tag = "второе" Then Tablo.Caption = "0": Ravno.Tag = ""

If Len(Tablo.Caption) < 16 Then

' если длина не превышает допустимой длины числа:

If Tablo.Caption = "0." And Tablo.Tag = "" Then

' если на табло ноль и нет режима начала ввода дробного числа - заменяем цифрой c точкой:

Tablo.Caption = Index & "."

Else

If Mid(Tablo.Caption, Len(Tablo.Caption), 1) = "." And Tablo.Tag <> "не целое" Then

' на табло целое число и нет режима ввода дробного числа - добавляем цифру и ставим точку в конце:

Tablo.Caption = Mid(Tablo.Caption, 1, Len(Tablo.Caption) - 1) & Index

Else

' на табло дробное число или режим ввода дробного числа - добавляем цифру надписи табло

Tablo.Caption = Tablo.Caption & Index

End If

End If

End If

Tablo.Tag = "" ' возможен (...) ввод дробного числа

End Sub

Однако этого кода достаточно чтобы обеспечить ввод целых положительных чисел. Добавим ещё две относительно простые по пониманию процедуры. Первая - изменение знака числа:

Private Sub Plusmin\_Click()

Select Case Val(Tablo.Caption)

' меньше нуля - "удаляем" минус:

Case Is < 0

Tablo.Caption = Mid(Tablo.Caption, 2, Len(Tablo.Caption))

' больше нуля - "прибавляем" минус:

Case Is > 0

Tablo.Caption = "-" & Tablo.Caption

End Select

End Sub

Другая - возможность ввода дробного числа:

Private Sub Tochka\_Click()

' если число целое, то разрешаем ввести дробное число:

If Fix(Val(Tablo.Caption)) = Val(Tablo.Caption) Or Ravno.Tag = "второе" Then Tablo.Tag = "не целое"

' к тому же ... если вводить дробное число после указания операции, то целая его часть - ноль:

If Ravno.Tag = "второе" Then Tablo.Caption = "0"

End Sub

*Исправление ошибок ...*

Позволим также стирать набранное число и удалять последнюю цифру числа на табло, что обеспечит следующий фрагмент программного кода:

Private Sub ochistka\_Click()

' щелчёк по кнопке CE "превращает" число на табло в ноль:

Tablo.Caption = "0"

Tablo.Tag = "" ' вводим целое число

End Sub

Private Sub udalit\_Click()

umnaya.SetFocus

' является ли число целым?

If Fix(Val(Tablo.Caption)) = Val(Tablo.Caption) Then

' если целое - удалим последнюю цифру

Tablo.Caption = Mid(Tablo.Caption, 1, Len(Tablo.Caption) - 2)

' а если число было нулём - то так и оставим - "ноль":

If Tablo.Caption = "." Then Tablo.Caption = "0"

Else

' если на табло число не целое, то просто удалим последнюю цифру надписи

Tablo.Caption = Mid(Tablo.Caption, 1, Len(Tablo.Caption) - 1)

End If

End Sub

*Считать, считать и ещё раз считать!*

Далее пользователь, скорее всего, захочет указать одно из арифметических действий: сложить, вычесть, умножить, разделить заданное число с каким-то другим. Предоставим ему такую возможность!

Здесь необходимо предусмотреть следующее: 1) пользователь может изменить выбранную операцию (не изменив до этого числа на табло), 2) если пользователь уже выбирал до этого какую-то операцию, не вычислял её значение с помощью клавиши равно или не задавал состояние первоначальной готовности, то вычисляется результат этой операции.

Легко понять, что эти особенности взаимосвязаны между собой: вторая возможна только тогда, кода пользователь не изменяет выбор операции.

Первая особенность выбора арифметического действия частично рассмотрена. Вспомним, что при указании пользователем операции программа запишет в переменную **Ravno.Tag** значение "второе". Таким образом, при вхождении в процедуру обработки события щелчка мыши по одной из кнопок **Operacia** условный оператор программного кода с помощью данной переменной проверит: "А не изменяет ли пользователь выбранную операцию?!". При отрицательном ответе с помощью переменной **deystvie** будет решать надо ли передавать управление процедуре вычисления значения резултата **Rezultat()**. В любых случаях, переменная **deystvie** примет значение соответствующее выбранному арифметическому действию. При этом уже знакомая нам переменная **pervoe** примет значение числа табло, а **Ravno.Tag** будет сообщать процедуре ввода числа, что вводится новое число и предоставлять возможность изменить операцию. Всё это находим в следующей процедуре:

Private Sub Operacia\_Click(Index As Integer)

If Ravno.Tag = "" Then ' нет изменения выбора операции?

' если операция была задана - вычисляем её значение:

If deystvie <> "нет" Then vtoroe = Val(Tablo.Caption): rezultat

End If

Select Case Index ' запишем выбранное действие

Case 0

deystvie = "разделить"

Case 2

deystvie = "вычесть"

Case 4

deystvie = "сложить"

Case 5

deystvie = "умножить"

End Select

pervoe = Val(Tablo.Caption) ' первое число операции - то что на табло при выборе операции

Ravno.Tag = "второе" ' режим ввода второго числа

End Sub

Несложно догадаться, что процедура вычисления результата будет вызываться также из процедуры обработки нажатия кнопки "равно". И прежде чем рассматривать первую, напишем программный код для второй.

Сделаем так, что если: 1) второе число не введено и кнопка "равно" нажимается один раз, то значение не вычисляется, 2) если кнопка нажимается два раза подряд, то в качестве второго числа операции выбирается текущее число табло и результат вычисляется.

Во всех случаях нам будет способствовать переменная **Ravno.Tag**. Если она равна "второе", то это удовлетворяет первому условию. Однако чтобы реализовать второе назначим данной переменной пустой символ. И уже здесь вследствие условия положим значение переменной **vtoroe** число, "которое на табло" и запустим процедуру **Rezultat()**, затем зададим калькулятору подобие начального состояния (без отображения на табло нуля). Можно написать так:

Private Sub Ravno\_Click()

If Ravno.Tag <> "второе" Then

vtoroe = Val(Tablo.Caption)

Call rezultat

Ravno.Tag = "второе" ' режим ввода "второго" числа

Tablo.Tag = "" ' две переменные ука-

deystvie = "нет" ' зывают начальное состояние

Else

Ravno.Tag = "" ' можем в "следующий раз" считать первым числом текущее число табло

End If

End Sub

*«Ноль», «нельзя» и «делить»*

А вот и процедура обработки результата (понимание которой у вас, я думаю, не составит труда):

Private Sub rezultat()

Select Case deystvie

Case "сложить"

Tablo.Caption = pervoe + vtoroe

Case "вычесть"

Tablo.Caption = pervoe - vtoroe

Case "разделить"

If vtoroe <> 0 Then

Tablo.Caption = pervoe / vtoroe

Else ' на ноль действительно нельзя делить!

Tablo.Caption = "на ноль не делят!"

End If

Case "умножить"

Tablo.Caption = pervoe \* vtoroe

End Select

End Sub

В ней, как видите, результат сразу "выбрасывается" на табло.

*О пользе клавиатуры.*

И прежде чем подготовиться к последнему желанию пользователя - иметь возможность производить вычисления с клавиатуры, добавим к уже имеющейся части программного кода процедуру обработки нажатия кнопки [С] (сброс в первоначальную) готовность:

Private Sub Sbros\_Click()

deystvie = "нет" ' стираем, возможно, указанное действие

Tablo.Caption = "0" ' на табло - ноль

Tablo.Tag = "" ' по умолчанию - вводится целое число

End Sub

Пользователь будет вводить число с клавиатуры используя клавиши с цифрами, клавиши с точкой, а для перемены знака числа клавишу +/=. Редактировать число с помощью клавиш Backspace и Пробел. Данные действия оперделим в коде процедуры **Form\_KeyPress()**, используя замечательный аргумент этой процедуры - **KeyAscii**.

Поэтому если знать (или узнать) Ascii-коды клавиш, то понимание этой процедуры не займёт много времени:

Private Sub Form\_KeyPress(KeyAscii As Integer)

Select Case KeyAscii

Case 8

udalit\_Click

Case 42, 43, 45, 47 ' теперь стало ясным почему индексы

Operacia\_Click (47 - KeyAscii) ' следуют не по порядку?

Case 61 ' меняем знак

Plusmin\_Click

Case Is > 47 ' вводим цифры

If KeyAscii < 58 Then knopka\_Click (KeyAscii - 48)

Case 32 ' а здесь пробелом "чистим" табло

ochistka\_Click

End Select

End Sub

Данная процедура не позволяет отлавливать нажатие «точки», так как Ascii-коды «точки» на малой цифровой клавиатуре отличны при разных раскладках. Напишем соответствующий код в следующей процедуре.

*Исход будет!*

Однако мы до сих пор не указали заменители кнопок "сброса в первоначальное состояние" и "равно". Для первого случая сложность заключается в придумывании клавиши, а для другого - способа "отлавливания" нажатия клавиши (я думаю ни у кого не возникнет сомнения в том, что нажатие клавиши Enter будет соответствовать щелчку по кнопке равно).

Сбрасываем в начальное состояние клавишей Delete:

Private Sub Form\_KeyDown(KeyCode As Integer, Shift As Integer)

If KeyCode = 46 Then sbros\_Click

' поскольку значения "точки" малой цифровой клавиатуры у KeyAscii не одинаковые

' при различных раскладках, то "ловим" её с помощью KeyCode

If KeyCode = 110 Then tochka\_Click

End Sub

Поймать нажатие клавиши Enter на кнопке можно следующим образом. Для этого вначале бросим на форму текстбокс. И спрячем его под границу будущего окна приложения (вначале увеличим высоту, затем поместим "туда" текстбокс и зададим высоту формы до её изменения). Изменим лишь свойства **TabIndex** на 1 и **Name** на **Umno**. Как вы уже скорее всего догадались нажатие по этому объекту будет выявлять нажатие клавиши Enter. Поэтому имеем такую процедуру:

Private Sub Umno\_KeyPress(KeyAscii As Integer)

If KeyAscii = 13 Then Ravno\_Click

End Sub

Однако, это сработает тогда, когда объект **Umno** имеет фокус. А что, если щёлкнуть по какой-то из кнопок? Тогда фокус переходит этой кнопке. И значит, нам надо вначале всех процедур обработки щелчков кнопок написать:

Umno.SetFocus

и тогда фокус будет всегда возвращаться на нужное нам место!

И, наконец, завершим наш проект, возвратясь почти к самому началу ...

"Покрасим" элементы управления!

Для этого процедуру **Form\_Load** можно написать так:

Private Sub Form\_Load()

Me.BackColor = RGB(50, 150, 250)

Sbros.BackColor = RGB(250, 50, 0)

Ochistka.BackColor = RGB(250, 75, 0)

Udalit.BackColor = RGB(250, 100, 0)

Ravno.BackColor = RGB(250, 125, 0)

Operacia(4).BackColor = RGB(150, 250, 250)

Operacia(2).BackColor = RGB(150, 250, 250)

Operacia(0).BackColor = RGB(150, 250, 250)

Operacia(5).BackColor = RGB(150, 250, 250)

End Sub

**Четвёртый шаг или "делу время, а потехе – час!»**

F5 - для проверки!