**Сортировка листов книги Excel с помощью процедуры VBA**

В настоящей заметке описана разработка утилиты VBA для Excel. Показан процесс анализа задачи и последующего ее решения. Пример рассмотрен в расчете на начинающих.[[1]](#footnote-1) Цель — разработать утилиту, которая изменяет порядок следования листов рабочей книги, сортируя их названия по алфавиту (без кода VBA это сделать невозможно). Если вы часто создаете книги с большим количеством листов, то знаете, что иногда сложно найти интересующий вас лист. Если же их упорядочить по названиям, то любой рабочий лист найти будет значительно проще.

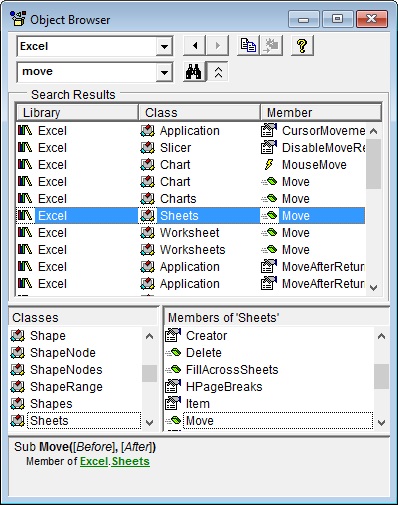


Рис. 1. Метод *Move* объекта *Sheets*

Начнем с перечисления требований к приложению. В процессе разработки вы будете обращаться к этому перечню для проверки правильности выполнения действий.

1. Приложение должно сортировать листы (т.е. рабочие листы и листы диаграмм) активной книги по названиям в алфавитном порядке.
2. Приложение всегда должно быть доступным, т.е., пользователь не должен открывать рабочую книгу для использования этой утилиты.
3. Приложение должно правильно выполняться по отношению к любой открытой рабочей книге.
4. В приложении не должны отображаться сообщения об ошибках VBA.

Часто самой сложной частью проекта является определение того, с чего же начать. В данном случае начнем с перечисления особенностей Excel, которые могут повлиять на соблюдение требований к проекту.

* В Excel отсутствует команда сортировки листов. Следовательно, отпадает вариант записи макроса для упорядочивания листов в алфавитном порядке.
* Лист можно легко переместить, перетащив его за ярлычок (включите функцию записи макросов и перетащите лист в другое место, чтобы узнать, какой код создается при таком действии).
* В Excel можно открыть диалоговое окно *Переместить или скопировать*, щелкнув правой кнопкой мыши на ярлычке листа с последующим выбором команды контекстного меню (убедитесь, что код макроса будет таким же, как и при перемещении листа вручную).
* Следует знать, сколько листов содержится в активной рабочей книге. Эту информацию можно получить с помощью VBA.
* Узнайте названия листов (вновь воспользовавшись VBA).
* В Excel существует команда, сортирующая данные в ячейках рабочего листа. Возможно, стоит перенести названия листов в диапазон ячеек и использовать эту функцию. Или, возможно, в VBA есть метод сортировки, которым можно будет воспользоваться в программе.
* Благодаря диалоговому окну Параметры макроса можно назначить макросу комбинацию клавиш.
* Если макрос сохранен в личной книге макросов, он всегда доступен.
* Вам понадобится тестировать приложение по мере разработки. Естественно, нельзя тестировать приложение в той же рабочей книге, в которой оно разработано. Создайте рабочую книгу, предназначенную специально для тестирования.
* Если разработать программу правильно, то VBA не будет отображать сообщения об ошибках. Не будем принимать желаемое за действительное...

Вот предварительный план, описывающий общие задачи:

1. идентифицировать активную рабочую книгу;
2. получить список названий всех листов в рабочей книге;
3. посчитать листы;
4. отсортировать их (определенным образом);
5. изменить порядок следования листов в соответствии с параметрами сортировки.

Если вам недостаточно информации о конкретных методах и свойствах, обратитесь к электронной справочной системе. Однако для начала лучше всего включить функцию записи макросов и посмотреть, что записывается в результате выполнения действий, связанных с решением поставленной задачи.

Откройте новую рабочую книгу, содержащую три рабочих листа. Включите функцию записи макросов и перетащите третий рабочий лист на место первого. Остановите запись макроса. Изучите код:

Sub Макрос1()

Sheets("Лист3").Select

Sheets("Лист3").Move Before:=Sheets(1)

End Sub

Найдите в справочной системе слово *Move* (это метод, перемещающий лист в рабочей книге на новое место). Для этого, например, находясь в окне VBE, нажмите F2 (рис. 1; см. также [справку в Интернете](https://msdn.microsoft.com/en-us/library/office/ff835603(v=office.15).aspx)). Данный метод имеет один аргумент, определяющий будущее положение листа.

Вам также необходимо узнать количество листов в активной рабочей книге. Активизируем окно отладки (*Immediate*) в VBE (нажав Ctrl+G) и введем такой оператор:

? ActiveWorkbook.Sheets.Count

VBA вернула значение 3 (рис. 2).

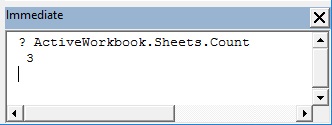


Рис. 2. Использование окна отладки в VBE для тестирования оператора

Теперь введем в окне отладки (*Immediate*) следующий оператор:

? ActiveWorkbook.Sheets(1).Name

В результате будет получено название первого листа — *ЛистЗ*.

Конструкция For Each-Next используется для циклического просмотра всех членов коллекции (см., например, [Основы программирования на VBA](http://baguzin.ru/wp/?p=16416), раздел Управление объектами и коллекциями):

Sub Test()

For Each Sht In ActiveWorkbook.Sheets

MsgBox Sht.Name

Next Sht

End Sub

Макрос отобразил три окна сообщения, в каждом из которых — новое название листа.

Что касается сортировки, справочная система подскажет, что метод *Sort* относится к объекту *Range*. Поэтому одним из решений задачи могло быть перенесение названия листов в диапазон ячеек и сортировка этого диапазона. Однако такая задача слишком сложна. Возможно, целесообразнее сформировать из названий листов массив строк, а затем отсортировать этот массив с использованием кода VBA.

Однако прежде следует задать первоначальные настройки:

1. Создайте пустую рабочую книгу с пятью рабочими листами: названия — Лист1, Лист2, ЛистЗ, Лист4 и Лист5.
2. Разместите листы произвольно, чтобы они следовали не по порядку.
3. Сохраните рабочую книгу как Test.xlsm.
4. Перейдите в VBE (меню Разработчик –> Visual Basic) и выберите проект Personal.xlsb в окне Project (Проект). Если Personal.xlsb не отображается в окне Project, значит вы никогда не использовали личную книгу макросов. Excel создаст для вас эту книгу, когда вы запишете макрос (любой) и определите, что он должен сохраняться в личной книге макросов (подробнее см. [Создание личной книги макросов](http://baguzin.ru/wp/?p=13294)).
5. Добавьте новый модуль VBA (используя команду *Inserts* –> *Module*).
6. Создайте пустую процедуру с названием *SortSheets* (рис. 3).
7. Перейдите в Excel. Выберите команду *Разработчик* –> Код –> Макросы для отображения диалогового окна *Макрос*.
8. В диалоговом окне *Макрос* выберите процедуру *SortSheets* и щелкните на кнопке *Параметры.* В открывшемся окне *Параметры макроса* выберите Ctrl+Shift+S.

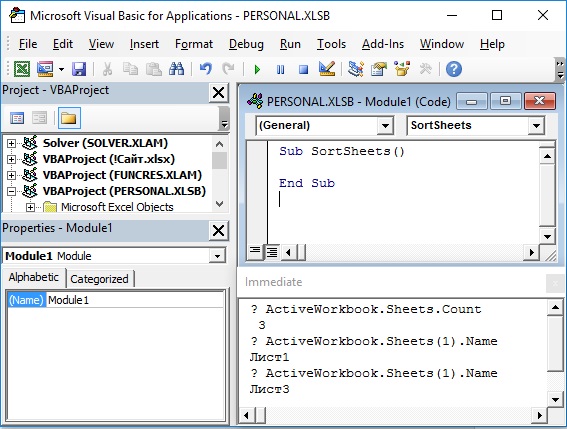


Рис. 3. Пустая процедура в модуле, находящемся в персональной книге макросов

Макрос можно сохранить в любом модуле личной книги макросов. Однако лучше хранить каждый макрос в отдельном модуле. Таким образом, вы сможете легко экспортировать модуль и импортировать его в другой проект.

### Начинаем писать код процедуры

Вначале необходимо поместить названия листов в массив строк. Так как пока неизвестно, сколько листов содержит активная рабочая книга, для объявления массива используем оператор Dim с пустыми скобками. Помните, что затем нужно применить оператор ReDim и изменить размерность массива на требуемое число элементов (подробнее см. [Основы программирования на VBA](http://baguzin.ru/wp/?p=16416), раздел Массивы). В цикл добавим функцию MsgBox, чтобы убедиться, что названия листов на самом деле вводятся в массив.

Sub SortSheets()

' Сортировка листов в активной рабочей книге

Dim SheetNames() as String

Dim i as Long

Dim SheetCount as Long

SheetCount = ActiveWorkbook.Sheets.Count

ReDim SheetNames(1 To SheetCount)

For i = 1 To SheetCount

SheetNames(i) = ActiveWorkbook.Sheets(i).Name

MsgBox SheetNames(i)

Next i

End Sub

Поместим код в модуль VBA, перейдем в Excel, активизируем книгу *Test.xlsm* и нажмем клавиши Ctrl+Shift+S. Появится пять окон сообщений с названиями листов активной рабочей книги. Рекомендуем вам тестировать код по мере его создания. Когда вы убедитесь, что программа работает правильно, удалите операторы *MsgBox*. Вместо того чтобы использовать функцию *MsgBox* в целях тестирования, можно обратиться к методу *Print* объекта *Debug*, который отображает сведения в окне отладки. Для этого замените *MsgBox* следующим оператором:

Debug.Print SheetNames(i)

Этот прием не столь навязчив по сравнению с использованием операторов MsgBox. Не забудьте только удалить оператор по завершении тестирования.

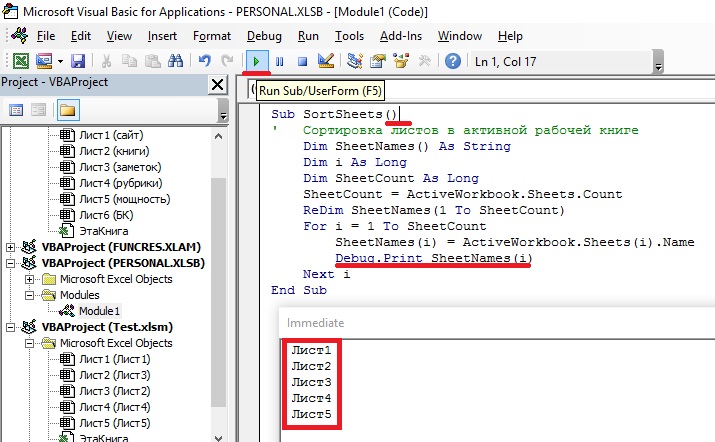


Рис. 4. Использование метода *Print* объекта *Debug* в целях тестирования. Разместите курсор внутри текста процедуры *Sub SortSheets()*, откройте окно *Immediate* (Ctrl+G), нажмите *Run*. В окне *Immediate* отразятся номера листов книги

Пока процедура *SortSheets* всего лишь создает массив названий листов в соответствии с порядком их следования в активной рабочей книге. Теперь нужно отсортировать значения в массиве *SheetNames* и изменить порядок следования листов в книге согласно отсортированному массиву.

### Создание процедуры сортировки

Можно вставить программу сортировки в процедуру *SortSheets*, но лучше написать общую процедуру сортировки, которую можно будет использовать и в других проектах (сортировка массивов — довольно популярная операция).

Существует несколько способов сортировки массивов. Мы выбрали пузырьковый метод (хотя это не очень быстрый прием, но его легко запрограммировать). В данном конкретном приложении высокая скорость выполнения операций не так уж важна. В пузырьковом методе используется вложенный цикл For-Next, в котором оценивается каждый элемент массива. Если элемент массива больше, чем следующий, то эти два элемента меняются местами. Такое сравнение повторяется для каждой пары элементов (т.е. *n – 1* раз).

Sub BubbleSort(List() As String)

' Сортировка массива List по возрастанию

Dim First As Long, Last As Long

Dim i As Long, j As Long

Dim Temp As String

First = LBound(List)

Last = UBound(List)

For i = First To Last – 1

For j = i + 1 To Last

If List(i) > List(j) Then

Temp = List(j)

List (j) = List(i)

List(i) = Temp

End If

Next j

Next i

End Sub

Эта процедура имеет один аргумент: одномерный массив с названием List. Массив, который передается в процедуру, может быть любой длины. Для присвоения нижней и верхней границ массива переменным First и Last использовались функции Lbound и UBound соответственно.

Ниже приведен код для тестирования процедуры BubbleSort:

Sub SortTester()

Dim x(1 To 5) As String

Dim i As Long

x(1) = "собака"

x(2) = "кот"

x(3) = "слон"

x(4) = "трубкозуб"

x(5) = "птица"

Call BubbleSort(x)

For i = 1 To 5

Debug.Print i, x(i)

Next i

End Sub

Процедура SortTester создает массив из пяти строк, передает его процедуре BubbleSort и отображает отсортированный массив в окне отладки *Immediate*. После того как код выполнил свое предназначение, он был удален.

Убедившись в том, что код работает надежно, я изменил процедуру SortSheets путем добавления вызова в процедуру BubbleSort, передачи массива SheetNames в качестве аргумента. Начиная с этого момента, модуль приобретает следующий вид.

Sub SortSheets()

' Сортировка листов в активной рабочей книге

Dim SheetNames() as String

Dim i as Long

Dim SheetCount as Long

SheetCount = ActiveWorkbook.Sheets.Count

ReDim SheetNames(1 To SheetCount)

For i = 1 To SheetCount

SheetNames(i) = ActiveWorkbook.Sheets(i).Name

Next i

Call BubbleSort(SheetNames)

End Sub

По окончании работы процедуры SortSheets образуется массив, состоящий из отсортированных названий листов активной рабочей книги. Чтобы проверить это, можно отобразить содержимое массива в окне отладки, добавив в конец процедуры перед оператором End Sub такой код:

For i = 1 То SheetCount

Debug.Print SheetNames(i)

Next i

Осталось написать программу для изменения порядка следования листов в книге в соответствии с отсортированными элементами массива SheetNames. Нам пригодится фрагмент макроса, записанного в начале заметки: Sheets("ЛистЗ").Move Before:=Sheets(1)

Напишем цикл For-Next, который просматривает каждый лист и перемещает его в соответствующее место, указанное в массиве SheetNames.

For i = 1 То SheetCount

Sheets(SheetNames(i)).Move Before:=Sheets(i)

Next i

Например, в первой итерации цикла счетчик *i = 1*. Первый элемент массива *SheetNames* – *Лист1*. Следовательно, выражение для метода *Move* в цикле будет таким:

Sheets("Лист1").Move Before:= Sheets(1)

Вторая итерация цикла:

Sheets("Лист2").Move Before:= Sheets(2)

В конец процедуры SortSheets добавим новый код:

Sub SortSheets ()

Dim SheetNames() As String

Dim SheetCount as Long

Dim i as Long

SheetCount = ActiveWorkbook.Sheets.Count

ReDim SheetNames(1 To SheetCount)

For i = 1 To SheetCount

SheetNames(i) = ActiveWorkbook.Sheets(i).Name

Next i

Call BubbleSort(SheetNames)

For i = 1 To SheetCount

ActiveWorkbook.Sheets(SheetNames(i)).Move \_

Before:=ActiveWorkbook.Sheets(i)

Next i

End Sub

Теперь необходимо собрать весь код. Объявим все переменные, используемые в процедурах, и добавим несколько комментариев, а также пустых строк, чтобы программу можно было легче прочесть. В результате процедура SortSheets будет приведена к следующему виду:

Sub SortSheets ()

' Эта процедура сортирует листы

' активной рабочей книги по возрастанию.

' Нажмите клавиши <Ctrl+Shift+S> для выполнения

Dim SheetNames() As String

Dim SheetCount As Long

Dim i As Long

' Определение количества листов и массива ReDim

SheetCount = ActiveWorkbook.Sheets.Count

ReDim SheetNames(1 To SheetCount)

SheetNames(i) = ActiveWorkbook.Sheets(i).Name

' Заполнение массива названиями листов

For i = 1 To SheetCount

SheetNames(i) = ActiveWorkbook.Sheets(i).Name

Next i

' Сортировка массива по возрастанию

Call BubbleSort(SheetNames)

' Перемещение листов

For i = 1 To SheetCount

ActiveWorkbook.Sheets(SheetNames(i)).Move \_

Before:= ActiveWorkbook.Sheets(i)

Next i

End Sub

### Дополнительное тестирование

Наверное, вы считаете, что работа окончена. Однако тот факт, что процедура работает с рабочей книгой Test.xlsm, не означает, что она будет работать со всеми рабочими книгами. Чтобы проверить программу, загрузим несколько других рабочих книг и вновь запустим программу. Скоро вы убедитесь в том, что приложение неидеально (если быть точным, оно далеко от идеала). Были обнаружены следующие проблемы:

* 1. Рабочие книги с большим количеством листов сортируются очень долго, так как при операциях перемещения окно постоянно обновляется.
  2. Сортировка не всегда выполняется. Например, лист с названием *SUMMARY* (все буквы в верхнем регистре) был размещен перед листом *Sheet1*. Эта проблема вызвана процедурой *BubbleSort* (так как U в верхнем регистре считается больше, чем h в нижнем).
  3. Если на экране не отображаются окна рабочих книг, при нажатии Ctrl+Shift+S код выдает ошибку.
  4. Если структура рабочей книги защищена, метод *Move* не работает.
  5. После сортировки последний лист рабочей книги становится активным. Изменение активного листа — не очень удачное решение проблемы; лучше, если бы активным оставался лист, который был таковым до начала выполнения программы.
  6. При прерывании макроса с помощью комбинации клавиш Ctrl+Break VBA отображает сообщение об ошибке.
  7. Макрос не может быть «обращен вспять» (вы не можете воспользоваться командой *Отменить*). Если пользователь случайно нажмет клавиши Ctrl+Shift+S, листы рабочей книги отсортируются, и в исходное состояние придется возвращать их вручную.

### Устранение проблем

*Проблема 1.* Чтобы решить проблему обновления изображения на экране вставьте в начале процедуры SortSheets инструкцию: Application.ScreenUpdating = False. Этот оператор «замораживает» окна Excel во время выполнения макроса. Еще один положительный момент заключается в том, что увеличивается скорость выполнения макроса. Когда выполнение макроса завершится, обновление экрана включится автоматически.

*Проблема 2.* Можно использовать функцию UCase для сравнения названия листов в верхнем регистре. В процедуре BubbleSort место строки If List(i) > List(j) Then вставьте строку If UCase(List(i)) > UCase(List(j)) Then. Проблему регистра можно решить иначе: добавьте в начало модуля оператор: Option Compare Text. В этом случае VBA выполняет сравнение строк на основе нечувствительных к регистру правил сортировки. Другими словами, *А* считается тем же, что и *а*.

*Проблема 3.* Чтобы избежать сообщения об ошибке, которое появляется, когда все рабочие книги свернуты, добавим процедуру проверки ошибок. Применим оператор On Error Resume Next (подробнее см. [Работа с процедурами VBA](http://baguzin.ru/wp/?p=16451), раздел Обработка ошибок), чтобы проигнорировать ошибку, и проверим значение Err. Если Err не равно нулю, это означает, что произошла ошибка. Следовательно, процедура заканчивается. Ниже приведен код проверки ошибок.

On Error Resume Next

SheetCount = ActiveWorkbook.Sheets.Count

If Err <> 0 Then Exit Sub ' нет активной рабочей книги

Можно и не использовать оператор On Error Resume Next. Альтернатива – поместить следующий оператор в верхнюю часть процедуры *SortSheets*:

If ActiveWorkbook Is Nothing Then Exit Sub

*Проблема 4.* Обычно для защиты структуры рабочей книги имеется серьезная причина. Мы не будем снимать защиту; программа должна отображать предупреждение, чтобы пользователь снял защиту и снова выполнил макрос. Проверку защищенной структуры книги выполнить легко — свойство ProtectStructure объекта WorkBook возвращает True, если книга защищена. Поэтому добавим в проект следующий код:

' Проверка защиты структуры рабочей книги

If ActiveWorkbook.ProtectStructure Then

MsgBox ActiveWorkbook.Name & " защищена.", \_

vbCritical, "Невозможно отсортировать листы."

Exit Sub

End If

Обратите внимание, что, поскольку ActiveWorkbook.ProtectStructure возвращает значение True, не требуется вводить If ActiveWorkbook.ProtectStructure = True. Если структура рабочей книги защищена, пользователь увидит окно сообщения (рис. 5). После нажатия Ok, процедура завершит свою работу.

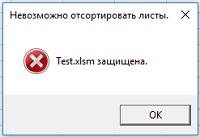


Рис. 5. Окно сообщения, возвращаемое процедурой SortSheets(), если книга защищена

*Проблема 5.* Для повторной активизации листа после завершения сортировки я написал код, который сопоставляет исходный лист с объектной переменной OldActiveSheet, а также активизирует этот лист после завершения процедуры. Ниже показан оператор, который инициализирует переменную.

Set OldActive = ActiveSheet

А следующий оператор активизирует рабочий лист, который был изначально активным:

OldActive.Activate

*Проблема 6.* После нажатия комбинации клавиш Ctrl+Break выполнение макроса обычно приостанавливается, и VBA выдает сообщение об ошибке. Но так как одна из целей проекта— избежать сообщений об ошибке, необходимо вставить команду предотвращения подобной ситуации. В справочной системе указано, что объект Application обладает свойством EnableCancelKey, которое может отключить комбинацию клавиш Ctrl+Break. Поэтому добавим следующий оператор в начало программы:

Application.EnableCancelKey = xlDisabled

Будьте внимательны, когда отключаете прерывание макроса, выполняемое с помощью клавиш Ctrl+Break. Если программа попадет в бесконечный цикл, выйти из него вы не сможете. Лучше использовать этот оператор, когда все работает идеально.

*Проблема 7.* Для предотвращения проблемы, возникающей из-за случайной сортировки листов, перед отключением клавиш Ctrl+Break в процедуру был добавлен следующий оператор:

If MsgBox("Сортировать листы в активной рабочей книге?", \_

vbQuestion + vbYesNo) <> vbYes Then Exit Sub

После вызова на выполнение процедуры SortSheets на экране появится сообщение (рис. 6). Макрос будет запущен только после нажатия *Да*. Случайный запуск макроса предотвращен.

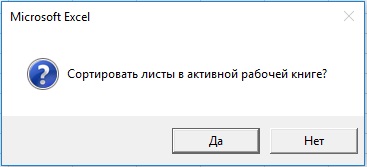


Рис. 6. Окно подтверждения необходимости сортировки листов

Финальный код процедуры можно найти в модуле VBA приложенного Excel-файла.

### Доступность

Макрос SortSheets сохранен в личной книге макросов, поэтому он всегда доступен при запуске Excel. На этом этапе макрос может выполняться при выборе названия макроса в диалоговом окне *Макрос*. Это окно можно отобразить, пройдя по меню *Вид* –> *Макросы* –> *Макросы*, или нажав Alt+F8. Можно сразу запустить макрос нажав Ctrl+Shift+S. Команду вызова макроса можно также добавить на ленту. Для этого:

1. Щелкните правой кнопкой мыши на ленте и в контекстном меню выберите команду *Настройка ленты*.
2. На вкладке *Настройка ленты* диалогового окна *Параметры Excel* в списке *Выбрать команды* выберите категорию *Макросы*.
3. Щелкните на значке *PERSONAL.XLSB!SortSheets*.
4. Используйте элементы управления в правом окне для создания новой вкладки и группы ленты (вы не сможете добавить команду в существующую группу.)

Я создал группу *Мои макросы* во вкладке *Вид* и переименовал новый, добавленный в эту группу элемент, на *Сортировка листов* (рис. 7).

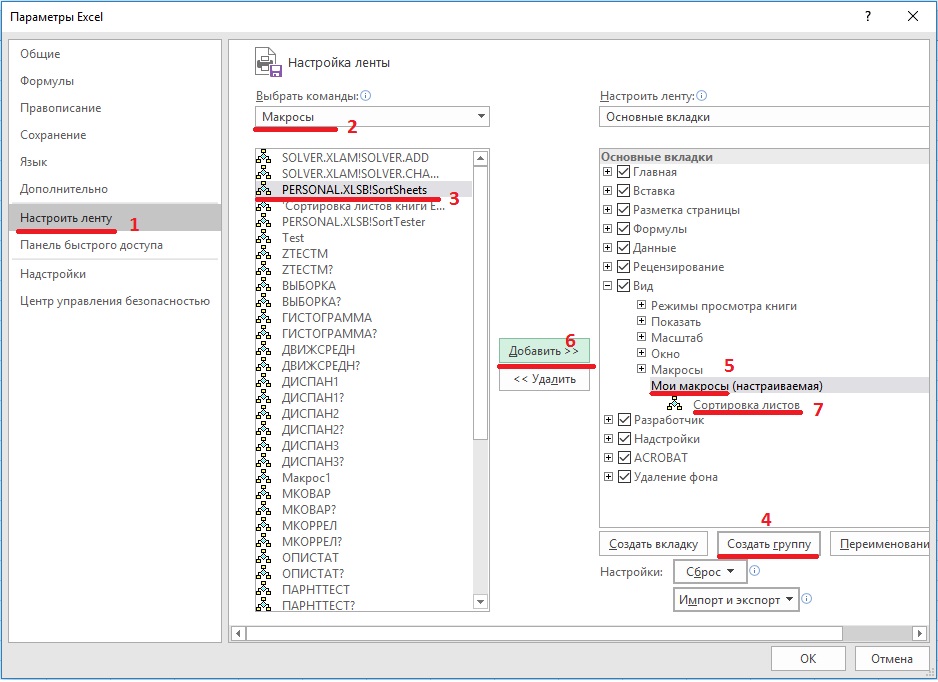


Рис. 7. Добавление новой команды на ленту

Итак, утилита соответствует всем изначальным требованиям: она сортирует все листы в активной рабочей книге, ее можно легко выполнить, она всегда доступна, она выполняется (что легко проверить) для всех рабочих книг и пока еще не отображала сообщений об ошибке VBA.

Правда, в процедуре все еще присутствует одна небольшая проблема: сортировка достаточно строгая и не всегда кажется «логичной». Например, после сортировки лист *Лист10* размещается перед *Лист2*. Большинство пользователей предпочитают видеть *Лист2* перед *Лист10*. Решить эту проблему довольно сложно. Если вы знаете, что листов будет более девяти, можно рекомендовать называть их *Лист01, Лист02, … Лист09, Лист10, ...*

1. По материалам книги [Джон Уокенбах. Excel 2010. Профессиональное программирование на VBA](http://baguzin.ru/wp/?p=16255). – М: Диалектика, 2013. – С. 273–286. [↑](#footnote-ref-1)