**Создание макросов и пользовательских функций на VBA**

**Введение**

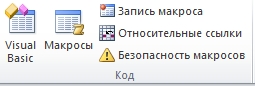
Всем нам приходится - кому реже, кому чаще - повторять одни и те же действия и операции в Excel. Любая офисная работа предполагает некую "рутинную составляющую" - одни и те же еженедельные отчеты, одни и те же действия по обработке поступивших данных, заполнение однообразных таблиц или бланков и т.д. Использование макросов и пользовательских функций позволяет автоматизировать эти операции, перекладывая монотонную однообразную работу на плечи Excel. Другим поводом для использования макросов в вашей работе может стать необходимость добавить в Microsoft Excel недостающие, но нужные вам функции. Например функцию сборки данных с разных листов на один итоговый лист, разнесения данных обратно, вывод суммы прописью и т.д.

*Макрос* - это запрограммированная последовательность действий (программа, процедура), записанная на языке программирования Visual Basic for Applications (VBA). Мы можем запускать макрос сколько угодно раз, заставляя Excel выполнять последовательность любых  нужных нам действий, которые нам не хочется выполнять вручную.

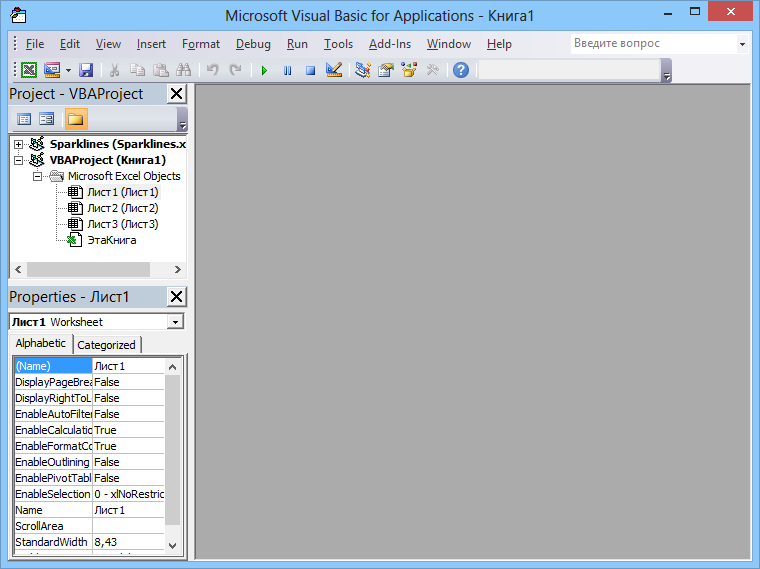
В принципе, существует великое множество языков программирования (Pascal, Fortran, C++, C#, Java, ASP, PHP...), но для всех программ пакета Microsoft Office стандартом является именно встроенный язык VBA. Команды этого языка понимает любое офисное приложение, будь то Excel, Word, Outlook или Access.

**Способ 1. Создание макросов в редакторе Visual Basic**

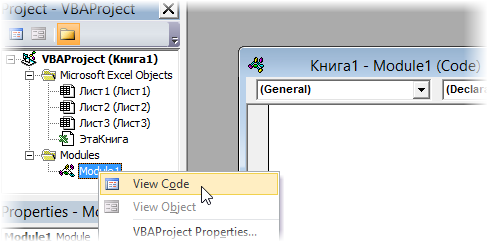
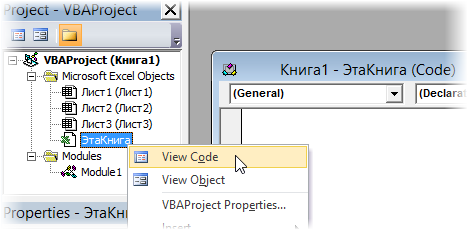
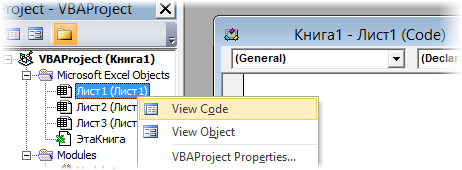
Для ввода команд и формирования программы, т.е. создания макроса необходимо открыть специальное окно - редактор программ на VBA, встроенный в Microsoft Excel.

* В старых версиях (Excel 2003 и старше) для этого идем в меню ***Сервис - Макрос - Редактор Visual Basic (Toos - Macro - Visual Basic Editor)***.
* В новых версиях (Excel 2007 и новее) для этого нужно сначала отобразить вкладку **Разработчик (Developer)**. Выбираем **Файл - Параметры - Настройка ленты (File - Options - Customize Ribbon)** и включаем в правой части окна флажок **Разработчик (Developer)**. Теперь на появившейся вкладке нам будут доступны основные инструменты для работы с макросами, в том числе и нужная нам кнопка **Редактор Visual Basic (Visual Basic Editor)**  
  :

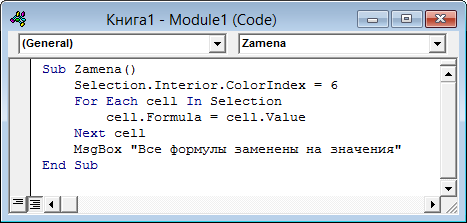
К сожалению, интерфейс редактора VBA и файлы справки не переводятся компанией  Microsoft на русский язык, поэтому с английскими командами в меню и окнах придется смириться:



Макросы (т.е. наборы команд на языке VBA) хранятся в программных модулях. В любой книге Excel мы можем создать любое количество программных модулей и разместить там наши макросы. Один модуль может содержать любое количество макросов. Доступ ко всем модулям осуществляется с помощью окна Project Explorer в левом верхнем углу редактора (если его не видно, нажмите CTRL+R). Программные модули бывают нескольких типов для разных ситуаций:

* **Обычные модули** - используются в большинстве случаев, когда речь идет о макросах. Для создания такого модуля выберите в меню **Insert - Module**. В появившееся окно нового пустого модуля можно вводить команды на VBA, набирая их с клавиатуры или копируя их из другого модуля, с этого сайта или еще откуда нибудь:  
    
  
* **Модуль Эта книга** - также виден в левом верхнем углу редактора Visual Basic в окне, которое называется Project Explorer. В этот модуль обычно записываются макросы, которые должны выполнятся при наступлении каких-либо событий в книге (открытие или сохранение книги, печать файла и т.п.):  
    
  
* **Модуль листа** - доступен через Project Explorer и через контекстное меню листа, т.е. правой кнопкой мыши по ярлычку листа - команда **Исходный текст (View Source)**. Сюда записывают макросы, которые должны выполняться при наступлении определенных событий на листе (изменение данных в ячейках, пересчет листа, копирование или удаление листа и т.д.)  
    
  

 Обычный макрос, введенный в стандартный модуль выглядит примерно так:



Давайте разберем приведенный выше в качестве примера макрос *Zamena*:

* Любой макрос должен начинаться с оператора **Sub**, за которым идет имя макроса и список аргументов (входных значений) в скобках. Если аргументов нет, то скобки надо оставить пустыми.
* Любой макрос должен заканчиваться оператором **End Sub**.
* Все, что находится между **Sub** и **End Sub** - тело макроса, т.е. команды, которые будут выполняться при запуске макроса. В данном случае макрос выделяет ячейку заливает выделенных диапазон (Selection) желтым цветом (код = 6) и затем проходит в цикле по всем ячейкам, заменяя формулы на значения. В конце выводится окно сообщения (MsgBox).

С ходу ясно, что вот так сразу, без предварительной подготовки и опыта в программировании вообще и на VBA в частности, сложновато будет сообразить какие именно команды и как надо вводить, чтобы макрос автоматически выполнял все действия, которые, например, Вы делаете для создания еженедельного отчета для руководства компании. Поэтому мы переходим ко второму способу создания макросов, а именно...

**Способ 2. Запись макросов макрорекордером**

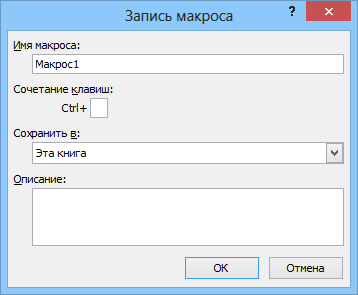
*Макрорекордер* - это небольшая программа, встроенная в Excel, которая переводит любое действие пользователя на язык программирования VBA и записывает получившуюся команду в программный модуль. Если мы включим макрорекордер на запись, а затем начнем создавать свой еженедельный отчет, то макрорекордер начнет записывать команды вслед за каждым нашим действием и, в итоге, мы получим макрос создающий отчет как если бы он был написан программистом. Такой способ создания макросов не требует знаний пользователя о программировании и VBA и позволяет пользоваться макросами как неким аналогом видеозаписи: включил запись, выполнил операци, перемотал пленку и запустил выполнение тех же действий еще раз. Естественно у такого способа есть свои плюсы и минусы:

* Макрорекордер записывает только те действия, которые выполняются в пределах окна Microsoft Excel. Как только вы закрываете Excel или переключаетесь в другую программу - запись останавливается.
* Макрорекордер может записать только те действия, для которых есть команды меню или кнопки в Excel. Программист же может написать макрос, который делает то, что Excel никогда не умел (сортировку по цвету, например или что-то подобное).
* Если во время записи макроса макрорекордером вы ошиблись - ошибка будет записана. Однако смело можете давить на кнопку отмены последнего действия (Undo) - во время записи макроса макрорекордером она не просто возрвращает Вас в предыдущее состояние, но и стирает последнюю записанную команду на VBA.

Чтобы включить запись необходимо:

* в Excel 2003 и старше - выбрать в меню **Сервис - Макрос - Начать запись (Tools - Macro - Record New Macro)**
* в Excel 2007 и новее - нажать кнопку **Запись макроса (Record macro)** на вкладке **Разработчик (Developer)**

Затем необходимо настроить параметры записываемого макроса в окне **Запись макроса**:

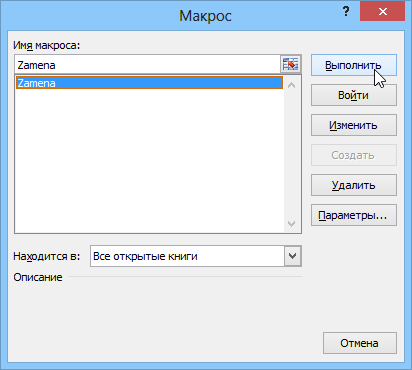


* **Имя макроса** - подойдет любое имя на русском или английском языке. Имя должно начинаться с буквы и не содержать пробелов и знаков препинания.
* **Сочетание клавиш** - будет потом использоваться для быстрого запуска макроса. Если забудете сочетание или вообще его не введете, то макрос можно будет запустить через меню **Сервис - Макрос - Макросы - Выполнить(Tools - Macro - Macros - Run)** или с помощью кнопки **Макросы (Macros)** на вкладке **Разработчик (Developer)** или нажав ALT+F8.
* **Сохранить в...** - здесь задается место, куда будет сохранен текст макроса, т.е. набор команд на VBA из которых и состоит макрос.:
  + *Эта книга* - макрос сохраняется в модуль текущей книги и, как следствие, будет выполнятся только пока эта книга открыта в Excel
  + *Новая книга* - макрос сохраняется в шаблон, на основе которого создается любая новая пустая книга в Excel, т.е. макрос будет содержаться во всех новых книгах, создаваемых на данном компьютере начиная с текущего момента
  + *Личная книга макросов* - это специальная книга Excel  с именем *Personal.xls*, которая используется как хранилище макросов. Все макросы из *Personal.xls* загружаются в память при старте Excel и могут быть запущены в любой момент и в любой книге.

После включения записи и выполнения действий, которые необходимо записать, запись можно остановить командой **Остановить запись (Stop Recording)***.*

**Запуск и редактирование макросов**

Управление всеми доступными макросами производится в окне, которое можно открыть с помощью кнопки **Макросы (Macros)** на вкладке **Разработчик (Developer)** или - в старых версиях Excel - через меню **Сервис - Макрос - Макросы (Tools - Macro - Macros)**:



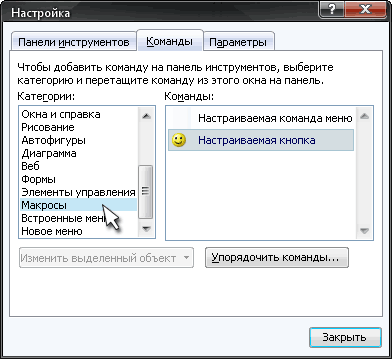
* Любой выделенный в списке макрос можно запустить кнопкой **Выполнить (Run)**.
* Кнопка **Параметры (Options)** позволяет посмотреть и отредактировать сочетание клавиш для быстрого запуска макроса.
* Кнопка **Изменить (Edit)** открывает редактор Visual Basic (см. выше) и позволяет просмотреть и отредактировать текст макроса на VBA.

**Создание кнопки для запуска макросов**

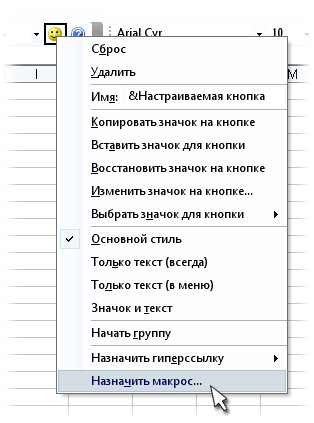
Чтобы не запоминать сочетание клавиш для запуска макроса, лучше создать кнопку и назначить ей нужный макрос. Кнопка может быть нескольких типов:

**Кнопка на панели инструментов в Excel 2003 и старше**

Откройте меню **Сервис - Настройка (Tools - Customize)** и перейдите на вкладку **Команды (Commands)**. В категории **Макросы** легко найти веселый желтый "колобок" - **Настраиваемую кнопку (Custom button):**

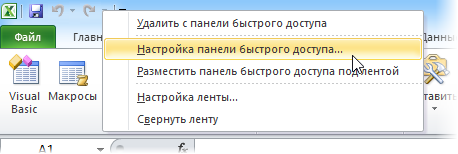


Перетащите ее к себе на панель инструментов и затем щелкните по ней правой кнопкой мыши. В контекстом меню можно назначить кнопке макрос, выбрать другой значок и имя:

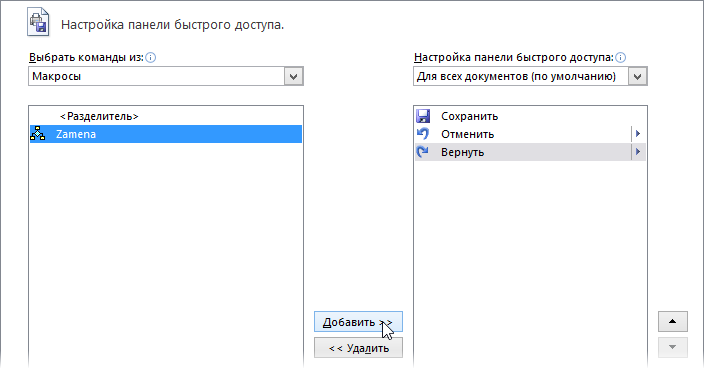


**Кнопка на панели быстрого доступа в Excel 2007 и новее**

Щелкните правой кнопкой мыши по панели быстрого доступа в левом верхнем углу окна Excel и выберите команду **Настройка панели быстрого доступа (Customise Quick Access Toolbar)**:



Затем в открывшемся окне выберите категорию **Макросы** и при помощи кнопки **Добавить (Add)** перенесите выбранный макрос в правую половину окна, т.е. на панель быстрого доступа:

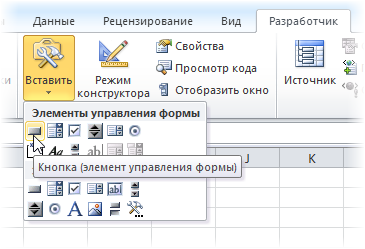


**Кнопка на листе**

Этот способ подходит для любой версии Excel. Мы добавим кнопку запуска макроса прямо на рабочий лист, как графический объект. Для этого:

* В Excel 2003 и старше - откройте панель инструментов **Формы**через меню**Вид - Панели инструментов - Формы (View - Toolbars - Forms)**
* В Excel 2007 и новее - откройте выпадающий список **Вставить (Insert)** на вкладке **Разработчик (Developer)**

Выберите объект **Кнопка (Button):**

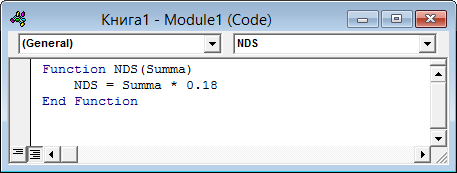


Затем нарисуйте кнопку на листе, удерживая левую кнопку мыши. Автоматически появится окно, где нужно выбрать макрос, который должен запускаться при щелчке по нарисованной кнопке.

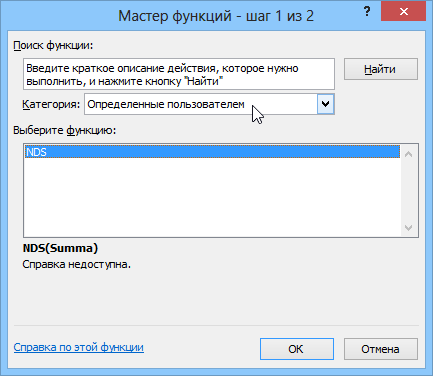
**Создание пользовательских функций на VBA**

Создание пользовательских функций или, как их иногда еще называют, UDF-функций (User Defined Functions) принципиально не отличается от создания макроса в обычном программном модуле. Разница только в том, что макрос выполняет последовательность действий с объектами книги (ячейками, формулами и значениями, листами, диаграммами и т.д.), а пользовательская функция - только с теми значениями, которые мы передадим ей как аргументы (исходные данные для расчета).

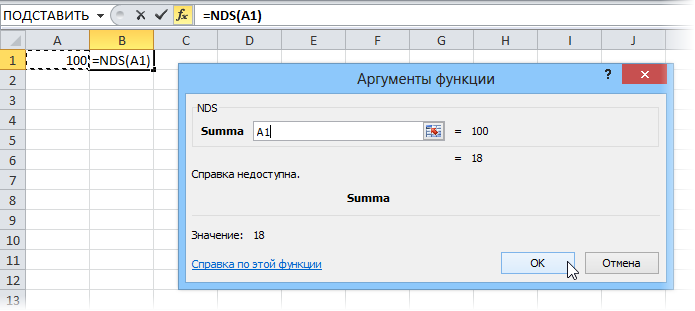
Чтобы создать пользовательскую функцию для расчета, например, налога на добавленную стоимость (НДС) откроем редактор VBA, добавим новый модуль через меню***Insert - Module*** и введем туда текст нашей функции:



Обратите внимание, что в отличие от макросов функции имеют заголовок **Function** вместо **Sub** и непустой список аргументов (в нашем случае это *Summa*). После ввода кода наша функция становится доступна в обычном окне Мастера функций (***Вставка - Функция***) в категории **Определенные пользователем (User Defined)**:



После выбора функции выделяем ячейки с аргументами (с суммой, для которой надо посчитать НДС) как в случае с обычной функцией:



**едизайнер таблиц**

3837 10.10.2012 [Скачать пример](https://www.planetaexcel.ru/upload/iblock/bce/redesigner.xls)

Не секрет, что большинство пользователей Excel, создавая таблицы на листах, думают в первую очередь о собственном комфорте и удобстве. Так рождаются на свет красивые,  со сложными "шапками", пестрые и громоздкие таблицы, которые при этом совершенно нельзя ни отфильтровать, ни отсортировать, а про автоматический [отчет сводной таблицей](https://www.planetaexcel.ru/techniques/8/130/)лучше и не думать вообще.

Рано или поздно пользователь такой таблицы приходит к мысли, что "пусть будет не так красиво, зато можно работать" и начинает упрощать дизайн своей таблицы, приводя его в соответствие с классическими рекомендациями:

* простая однострочная шапка, где у каждого столбца будет свое уникальное название (имя поля)
* одна строка - одна законченная операция (сделка, продажа, проводка, проект и т.д.)
* без объединенных ячеек
* без разрывов в виде пустых строк и столбцов

Но если сделать однострочную шапку из многоэтажной или разбить один столбец на несколько достаточно просто, то реконструирование таблицы может занять много времени (особенно при больших размерах ). Имеется ввиду следующая ситуация:

|  |  |  |  |
| --- | --- | --- | --- |
| Из |  | сделать |  |

В терминах баз данных правую таблицу обычно называют плоской (flat) - именно по таким таблицам лучше всего строить отчеты [сводных таблиц (pivot tables)](https://www.planetaexcel.ru/techniques/8/130/) и проводить аналитику.

Преобразовать двумерную таблицу в плоскую можно при помощи простого макроса. Откройте редактор Visual Basic через вкладку **Разработчик - Visual Basic (Developer - Visual Basic Editor)** или сочетанием клавиш Alt+F11. Вставьте новый модуль (***Insert - Module***) и скопируйте туда текст этого макроса:

[?](https://www.planetaexcel.ru/techniques/24/55/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | **Sub** Redesigner()  **Dim** i **As** **Long**  **Dim** hc **As** **Integer**, hr **As** **Integer**  **Dim** ns **As** Worksheet        hr = InputBox("Сколько строк с подписями сверху?")      hc = InputBox("Сколько столбцов с подписями слева?")        Application.ScreenUpdating = **False**        i = 1  **Set** inpdata = Selection  **Set** ns = Worksheets.Add    **For** r = (hr + 1) **To** inpdata.Rows.Count  **For** c = (hc + 1) **To** inpdata.Columns.Count  **For** j = 1 **To** hc                  ns.Cells(i, j) = inpdata.Cells(r, j)  **Next** j    **For** k = 1 **To** hr                  ns.Cells(i, j + k - 1) = inpdata.Cells(k, c)  **Next** k                ns.Cells(i, j + k - 1) = inpdata.Cells(r, c)              i = i + 1  **Next** c  **Next** r  **End** **Sub** |

После этого можно закрыть редактор VBA и вернуться в Excel. Теперь можно выделить исходную таблицу (полностью, с шапкой и первым столбцом с месяцами) и запустить наш макрос через **Разработчик - Макросы (Developer - Macros)** или нажав сочетание Alt+F8.

Макрос вставит в книгу новый лист и создаст на нем новый, реконструированный вариант выделенной таблицы. С такой таблицей можно работать "по полной программе", применяя весь арсенал средств Excel для обработки и анализа больших списков.

Sub Redesigner()

    Dim inpdata As Range, realdata As Range, ns As Worksheet

    Dim i&, j&, k&, c&, r&, hc&, hr&

    Dim out(), dataArr, hcArr, hrArr

    hr = Val(InputBox("Сколько строк с подписями данных сверху?"))

    hc = Val(InputBox("Сколько столбцов с подписями данных слева?"))

    Set inpdata = Selection

    If inpdata.Rows.Count <= hr Or inpdata.Columns.Count <= hc Then Exit Sub

    Set realdata = inpdata.Offset(hr, hc).Resize(inpdata.Rows.Count - hr, inpdata.Columns.Count - hc)

    dataArr = realdata.Value

    If hr Then hrArr = inpdata.Offset(0, hc).Resize(hr, inpdata.Columns.Count - hc).Value

    If hc Then hcArr = inpdata.Offset(hr, 0).Resize(inpdata.Rows.Count - hr, hc).Value

    ReDim out(1 To Application.CountA(realdata), 1 To hr + hc + 1)

    Set ns = Worksheets.Add

    For i = 1 To UBound(dataArr, 1)

        For j = 1 To UBound(dataArr, 2)

            If Not IsEmpty(dataArr(i, j)) Then

                k = k + 1

                For c = 1 To hc: out(k, c) = hcArr(i, c): Next c

                For r = 1 To hr: out(k, c + r - 1) = hrArr(r, j): Next r

                out(k, c + r - 1) = dataArr(i, j)

            End If

    Next j, i

    ns.Cells(2, 1).Resize(UBound(out, 1), UBound(out, 2)) = out

End Sub

**Сборка листов из разных книг в одну**

1852 07.10.2012 [Скачать пример](https://www.planetaexcel.ru/upload/iblock/fab/fabb239fa1cb7b78861a3e4bbccb7e81.xls)

Предположим, имеется куча книг Excel, все листы из которых надо объединить в один файл. Копировать руками долго и мучительно, поэтому имеет смысл использовать несложный макрос.

Открываем книгу, куда хотим собрать листы из других файлов, входим в редактор Visual Basic сочетанием клавиш Alt+F11 одноименной кнопкой на вкладке **Разработчик (Developer - Visual Basic)**, добавляем новый пустой модуль (в меню ***Insert - Module***) и копируем туда текст вот такого макроса:

[?](https://www.planetaexcel.ru/techniques/12/49/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | **Sub** CombineWorkbooks()  **Dim** FilesToOpen  **Dim** x **As** **Integer**        Application.ScreenUpdating = **False**  'отключаем обновление экрана для скорости        'вызываем диалог выбора файлов для импорта      FilesToOpen = Application.GetOpenFilename \_        (FileFilter:="All files (\*.\*), \*.\*", \_        MultiSelect:=**True**, Title:="Files to Merge")    **If** TypeName(FilesToOpen) = "Boolean" **Then**          MsgBox "Не выбрано ни одного файла!"  **Exit** **Sub**  **End** **If**        'проходим по всем выбранным файлам      x = 1  **While** x <= UBound(FilesToOpen)  **Set** importWB = Workbooks.Open(Filename:=FilesToOpen(x))          Sheets().Copy After:=ThisWorkbook.Sheets(ThisWorkbook.Sheets.Count)          importWB.Close savechanges:=**False**          x = x + 1      Wend        Application.ScreenUpdating = **True**  **End** **Sub** |

После этого можно вернуться в Excel и запустить созданный макрос на вкладке **Разработчик**кнопкой **Макросы (Developer - Macros)** или нажав Alt+F8. Отобразится диалоговое окно открытия файла, где необходимо указать один или несколько (удерживая Ctrl или Shift) файлов, листы из которых надо добавить к текущей книге - и задача решена!

**Sub** Get\_All\_File\_from\_Folder()

**Dim** sFolder **As** **String**, sFiles **As** **String**

**Dim** wb **As** Workbook

'диалог запроса выбора папки с файлами

**With** Application.FileDialog(msoFileDialogFolderPicker)

**If** .Show = **False** **Then** **Exit** **Sub**

sFolder = .SelectedItems(1)

**End** **With**

sFolder = sFolder & IIf(Right(sFolder, 1) = Application.PathSeparator, "", Application.PathSeparator)

'отключаем обновление экрана, чтобы наши действия не мелькали

Application.ScreenUpdating = **False**

sFiles = Dir(sFolder & "\*.xls\*")

**Do** **While** sFiles <> ""

'открываем книгу

**Set** wb = Application.Workbooks.**Open**(sFolder & sFiles)

'действия с файлом

'Запишем на первый лист книги в ячейку А1 - www.excel-vba.ru

wb.Sheets(1).Range("A1").Value = "www.excel-vba.ru"

'Закрываем книгу с сохранением изменений

wb.**Close** **True** 'если поставить False - книга будет закрыта без сохранения

sFiles = Dir

**Loop**

'возвращаем ранее отключенное обновление экрана

Application.ScreenUpdating = **True**

**End** **Sub**

Private Sub Worksheet\_Change(ByVal Target As Range)

If Target.Count > 1 Then Exit Sub

If Target.Column = 1 Then

Target.Value =Round(Target.Value,1)

End If

End Sub

Можно RoundUp

Private Sub Worksheet\_Change(ByVal Target As Range)

Dim r, b As Integer

Dim str As String

If Target.Count > 1 Then Exit Sub

If Target.Column = 1 Then

str = Target.Value

r = InStr(str, ",")

b = Len(str)

If b > r + 1 Then

Target.Value = CDbl(Left(str, r + 1))

End If

End If

End Sub

Sub CheckProfit()

If Range("C5") >= 10000 And Range("C6") < 5000 Then

MsgBox "$5,000 profit achieved!"

Else

Msgbox "Profit not achieved!"

End If

End Sub