

**ЭЛЕМЕНТЫ АВТОМАТИЗАЦИИ**

**ОБРАБОТКИ ТАБЛИЧНЫХ ДАННЫХ**

**Содержание:**

**ВВЕДЕНИЕ . . . . . . . . . . . 3**

1. **ШАБЛОНЫ . . . . . . . . . . 3**

**Пример 1.1. Рассылка писем . . . . . . . . . 3**

**Пример 1.2. Печать на пластиковых картах . . . . . . . 13**

**Пример 1.3. Коллективный договор ДМС . . . . . . . 15**

1. **ТАБЛИЦЫ . . . . . . . . . . . 17**

**Пример 2.1. Деление ФИО . . . . . . . . . 17**

**Пример 2.2. Удаление лишних пробелов . . . . . . . 18**

**Пример 2.3. Определение пола . . . . . . . . 18**

**Пример 2.4. Разделение слева . . . . . . . . 19**

**Пример 2.5. Разделение справа . . . . . . . . 19**

**Пример 2.6. Поиск дублей . . . . . . . . . 20**

**Пример 2.7. Сравнение таблиц . . . . . . . . 20**

**Пример 2.8. Проверка символов . . . . . . . . 21**

**Пример 2.9. Отправка по EMAIL . . . . . . . . 22**

**ЗАКЛЮЧЕНИЕ . . . . . . . . . . . 22**

**ВВЕДЕНИЕ.**

Значительный опыт обработки табличных данных, позволяет мне поделиться им в сфере автоматизации некоторых, часто встречающихся задач. Не смотря на специфику различных отделов, в нашей повседневной работе, очень часто приходится сталкиваться с одними и теми же ситуациями по обработке поступающих данных. В данной работе, я постараюсь рассказать, как можно успешно автоматизировать некоторые, наиболее актуальные из них, не затрачивая при этом значительного количества сил и времени на комплексное изучение вопросов автоматизации.

Начну с того, что все последующие действия и примеры, исполнены во всем известном нам приложении – Excel, из пакета MS Office, с использованием такого инструмента, как макросы, которые, можно так сказать, являются упорядоченным набором команд для Excel, заставляющим программу выполнять необходимые нам действия. Выбор Excel не случаен, так как его возможности по обработке не только числовых данных, но и текста, довольно широки. Кроме того, стоит отметить тот факт, что такой инструмент как макросы, в той плоскости, которую рассмотрим мы в данной работе, практически не изменялся от версии к версии пакета MS Office, т.е. то, что было написано для версии, скажем 1997 года, без каких-либо изменений, будет работать и в версии пакета 2019 года.

Вообще, макросы – это инструмент, основанный на полноценном объектно-ориентированном языке программирования VBA (Visual Basic for Applications), разработанным компанией Microsoft, но хочу сразу успокоить – изучать программирование не придётся, необходимо будет лишь освежить некоторые знания, полученные в ходе школьного курса информатики. Как показала практика, сотрудники нашего офиса, с которыми я уже делился опытом по описанным ниже элементам автоматизации, и не изучавшие специализировано какие-либо дополнительные курсы по информационным технологиям, весьма быстро схватывали предложенную информацию, а некоторых из них, это даже сподвигло на дальнейшее изучение приоткрытой темы, но уже самостоятельно. Ещё одним фактором, подтолкнувшем их к этому, послужило то, что была оценена колоссальная разница в том, сколько времени уходило на обработку данных ранее и как быстро всё решалось после нескольких манипуляций, что вызывало неподдельный восторг и желание узнать, как же всё это работает. В любом случае, в своём описании, я буду ориентироваться на минимальный уровень подготовки, и максимально подробно расписывать, даже очевидные для кого-то, моменты.

Дальнейшее описание, предложенных мной задач автоматизации, я разделю на две части, разбив на конкретные примеры. В первой из них, пойдет речь о том, как создавать и использовать шаблоны различных печатных форм, а во второй – коснусь темы обработки табличных данных, которые используются в тех же шаблонах.

1. **ШАБЛОНЫ.**

**Пример 1.1. Рассылка писем.**

Начнём с простейшей и довольно распространённой задачи – формирование писем по списку. В данном примере мы создадим печатный шаблон письма и, с помощью макроса, заполним его из имеющегося у нас списка рассылки.

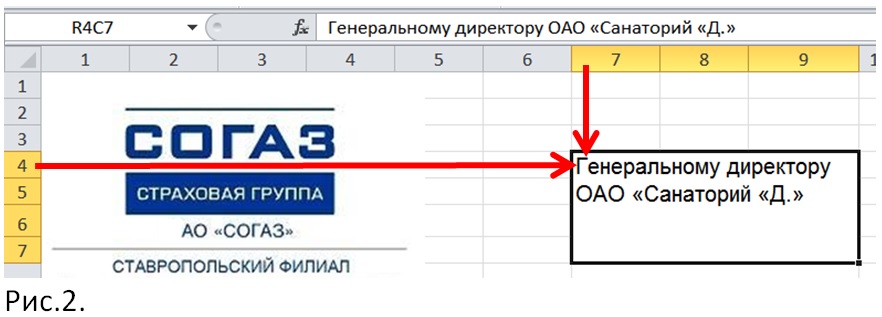
Что же изначально мы имеем? В excel-файле (…\Примеры\Рассылка писем.xlsm) есть два листа, на одном из которых находится таблица с уже подготовленными данными («ЛПУ»), а на втором – форма для вывода на печать («пример письма»), тот самый шаблон, который автоматически заполнится и отправится на печать. На листе «ЛПУ», данные собраны в четыре столбца:



Условно, их можно сгруппировать так:

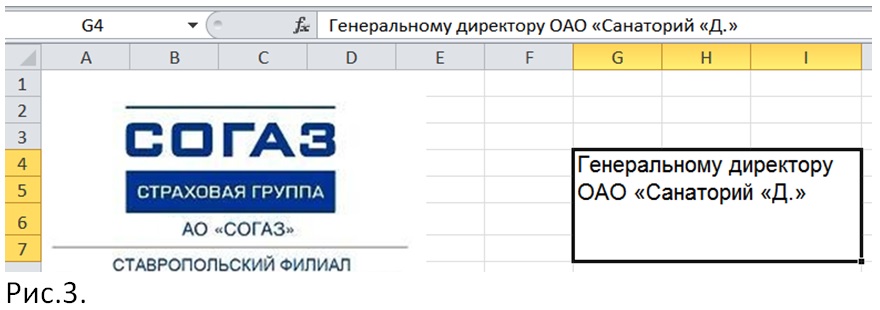
1. Должность и наименование организации;
2. ФИО руководителя;
3. Адрес организации;
4. Имя и отчество руководителя.

При формировании шаблона нам потребуется в одних и тех же местах этого листа, последовательно вставлять различные данные со второго. Для этого нам нужно определить, в каких же местах (ячейках), должны появляться эти данные.

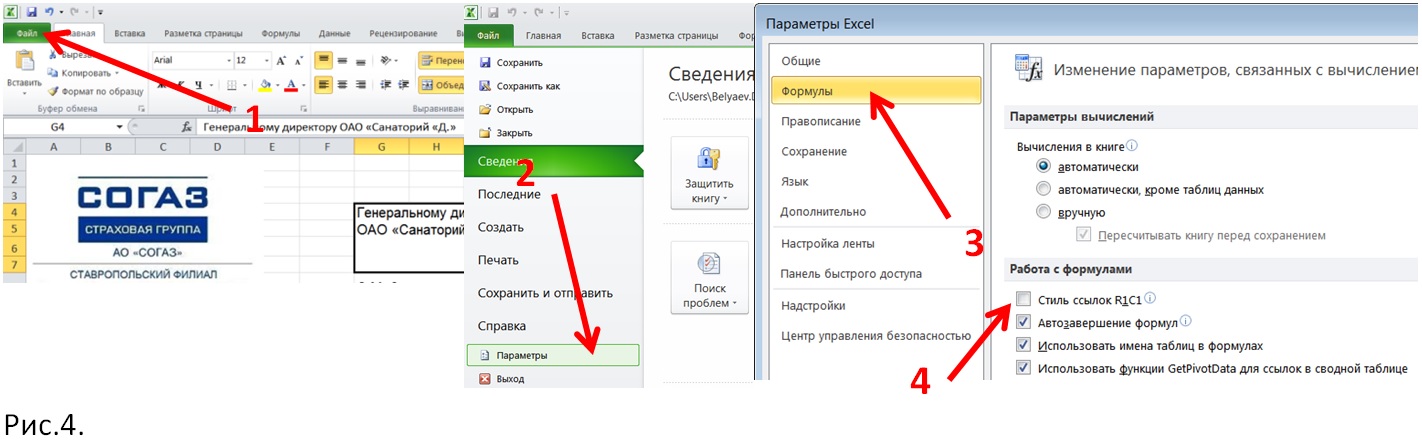


На рисунке 2, видно, что данные о должности руководителя и наименованию организации, стоят в ячейке, которая расположена в строке 4 и столбце 7. Сразу хочу отметить, что порядок обращения к ячейкам, всегда начинается с номера строки, а затем номера столбца, и не наоборот. Это важно и понадобится нам, когда мы начнём давать команды программе. Еще, стоит обратить внимание, что данная ячейка объединена с соседними, в одну большую. Это сделано для того, чтобы даже длинное наименование должности и организации, могли поместиться в ней. Однако, и при этом, номер данной ячейки сохраняется (строка 4, столбец 7), просто увеличиваясь, она как бы накрывает собой другие ячейки.

Сразу хочу предупредить, что в зависимости от настроек Excel, вы можете увидеть несколько другую картину:



На рисунке 3, заголовки полей выглядят иначе и вместо числового представления столбцов, идет буквенное. Это говорит о том, что в настройках программы не установлен необходимый нам формат отображения ссылок. Меняем настройки:



На рисунке 4 показана последовательность действий. Сначала, в меню программы выбираем пункт «Файл», затем «Параметры». В появившемся окне параметров, слева, находим раздел «Формулы» и в нём устанавливаем флажок «Стиль ссылок R1C1». Нажав клавишу «ОК» в окне параметров, подтверждаем применение изменений и получаем нужное представление.

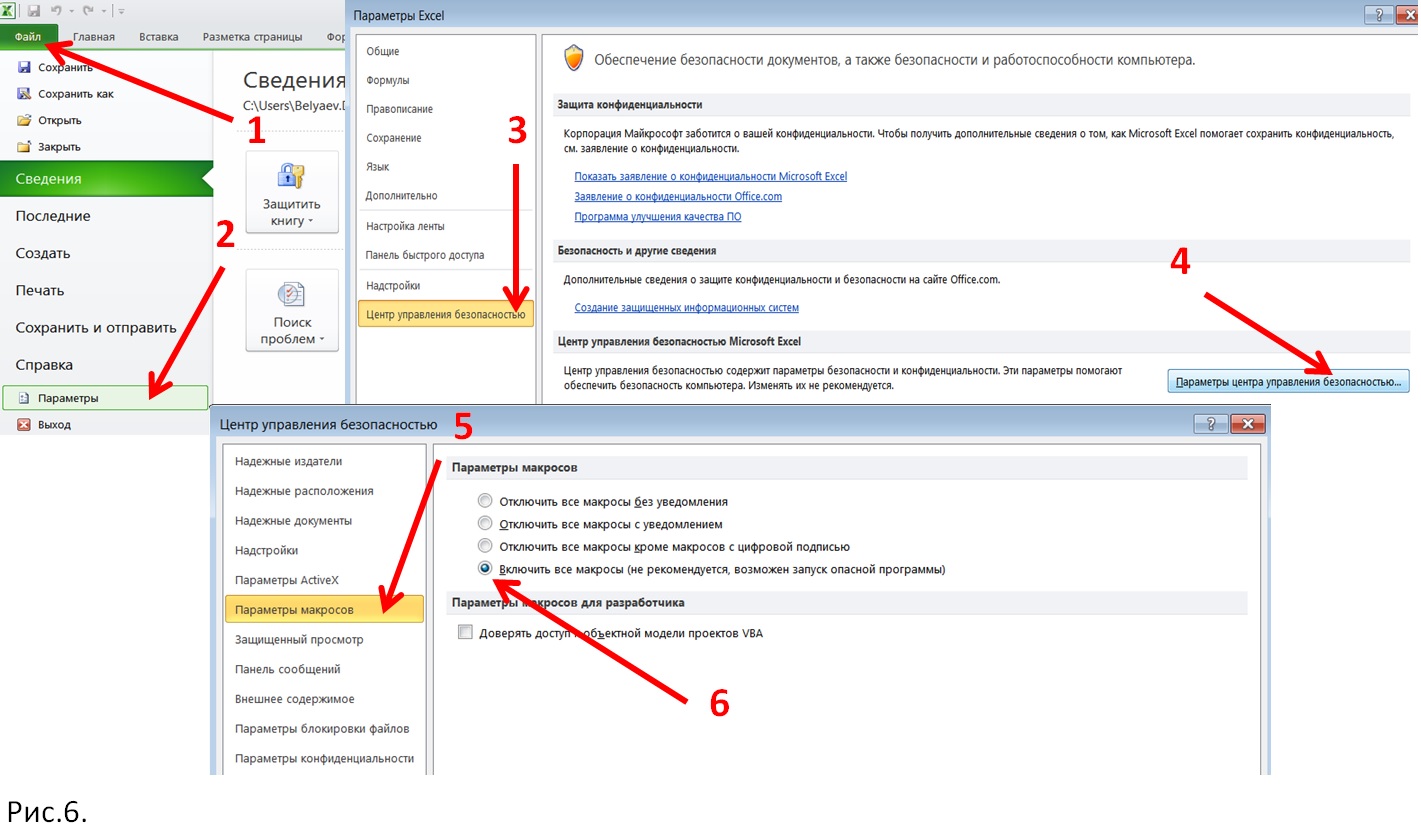
Продолжаем определять необходимые нам ячейки:



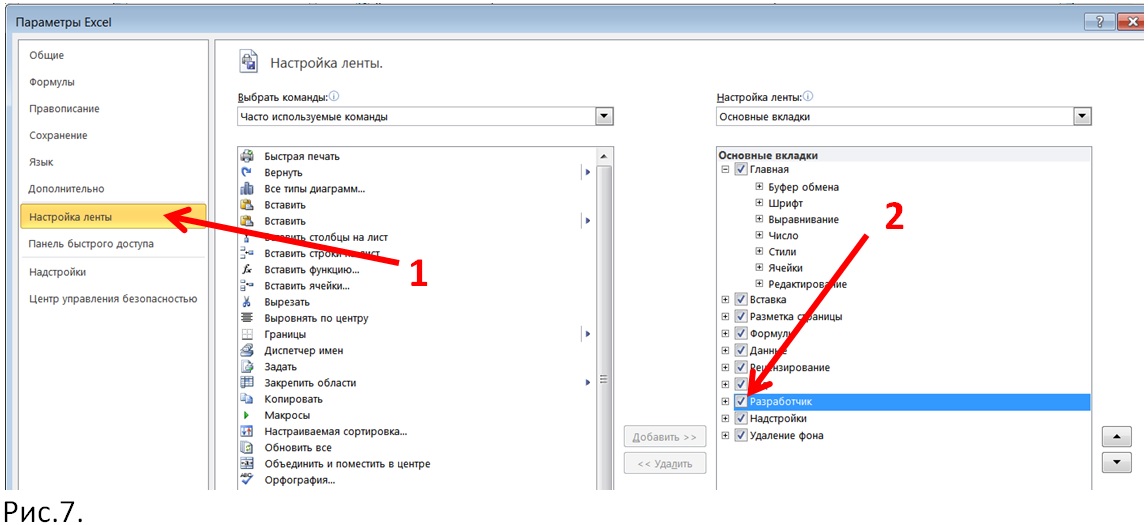
В результате мы получаем соответствие табличных данных, их положению в шаблоне:

1. Должность и наименование организации – строка 4, столбец 7;
2. ФИО руководителя – строка 8, столбец 7;
3. Адрес организации – строка 10, столбец 7;
4. Имя и отчество руководителя – строка 18, столбец 4.

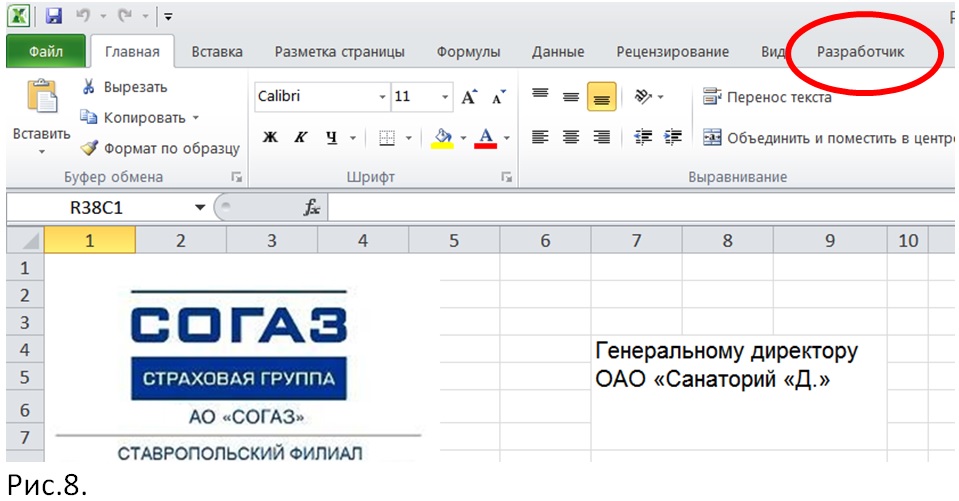
Становится очевидно, что все значения столбцов из каждой строки таблицы, должны попасть в определённые нами ячейки в шаблоне, сформировав полноценный документ, который распечатается, а затем, также обработаются все последующие строки таблицы. Для организации этого автоматического перебора, нам и потребуется такой инструмент, как макросы. По умолчанию, он отключён в Excel и его потребуется включать и выводить на панель.



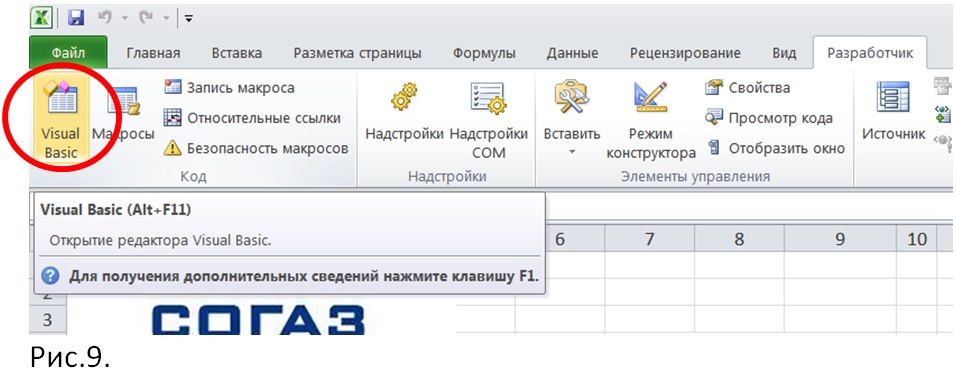
На рисунке 6 показан порядок включения макросов. Сначала, в меню программы выбираем пункт «Файл», затем «Параметры». В появившемся окне параметров, слева, находим раздел «Центр управления безопасностью» и нажимаем справа на кнопку «Параметры центра управления безопасностью». В открывшемся окне, слева находим раздел «Параметры макросов» и там устанавливаем переключатель в положение «Включить все макросы». Нажав клавишу «ОК», подтверждаем применение изменений и возвращаемся в параметры, где чуть выше выбираем раздел «Настройка ленты» и устанавливаем флажок для раздела «Разработчик»:



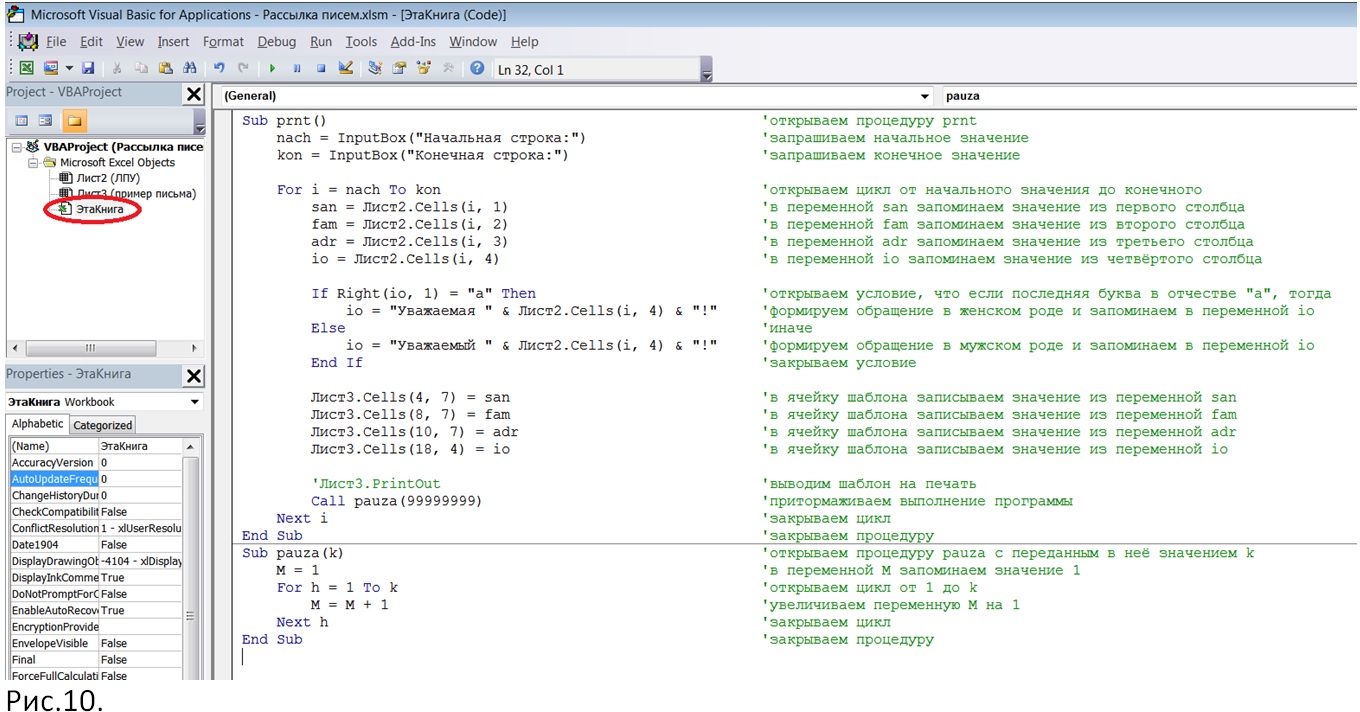
После установки флага и нажатия кнопки «ОК», появляется дополнительный пункт меню «Разработчик», которого ранее на панели не было. Вот через него мы и будем заходить в редактор, позволяющий нам писать команды для Excel:



Итак, наш шаблон готов, данные в таблице заполнены, осталось заставить Excel последовательно, из каждой строки, вставлять данные в выбранные нами места шаблона. Для этого открываем пункт меню «Разработчик» и выбираем раздел «Visual Basic»:



Для тех, кто уж совсем не желает много двигать мышкой, существуют «горячие клавиши» Alt+F11, выполняющие те же действия по открытию редактора. Не стоит пугаться обилию открывшихся элементов, большей частью из них, пользоваться не придётся. В верхнем левом окне нашего проекта, одноимённого с названием файла, мы видим, из каких объектов он состоит. Пока, нас интересует только объект «ЭтаКнига», по нему мы и кликаем дважды, открывая в правой части окна, его содержимое – тот набор команд, который и заставляет всё работать так, как нам надо.



Сразу оговорюсь, что на самом деле для достижения нужного результата, можно было бы ограничиться набором команд в два раза меньшим чем здесь и всё бы работало, но то, что было добавлено, привнесло некоторые удобства, о которых я ниже расскажу. А начну по порядку, подробно описывая каждую команду.

Прежде всего, в программировании любой код для выполнения (процедура), должен быть поименован, для того, чтобы можно было обращаться к нему из любого места программы. В данном случае, я назвал его «prnt». Имена процедур и переменных, можно придумывать какие угодно, но по определённым правилам – они не должны начинаться с цифры и содержать пробелы, не могут состоять только из цифр, а также совпадать с зарезервированными именами команд.

Как вы уже обратили внимание, набор команд внешне выглядит неравномерно, а имеет некоторую вложенность, что позволяет удобнее читать код.

Итак, первая строка программы:

Sub prnt()

Синим цветом, в редакторе помечается зарезервированные команды (операторы), в данном случае Sub – это оператор, объявляющий об открытии нашей процедуры «prnt», т.к. мы не можем просто начать писать набор команд, нам его надо сначала открыть, поименовав, а после всех действий, закрыть, чтобы редактор понял, где ему начинать действовать, а где заканчивать. Забегая вперёд скажу, что значительно ниже идёт команда End Sub, которая и закрывает нашу процедуру. Таким образом, минимальный набор команд для исполнения редактором будет выглядеть как:

Sub prnt()

End Sub

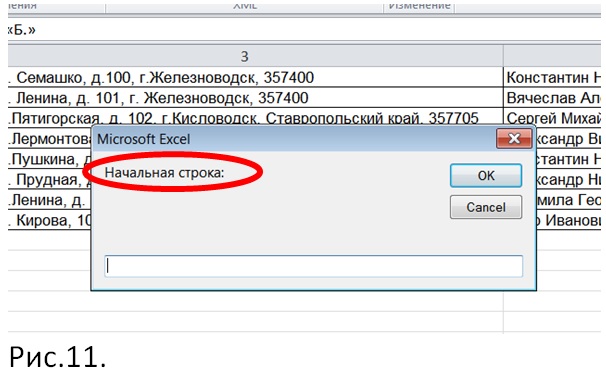
Следует сразу отметить, что каждая команда пишется в отдельной строке, а представленный таким образом набор, при запуске на выполнение, скажет редактору, что необходимо открыть процедуру «prnt», а затем её закрыть, и так как между этими операторами никаких команд не стоит, соответственно при выполнении ничего не произойдёт.

Возвращаясь к первой строке программы, нужно пояснить назначение скобок после названия процедуры. Они предназначены для того, чтобы показать, какие значения передаются нашей процедуре при открытии, а так как в данном случае мы ничего передавать не собираемся, то в скобках ничего и не стоит. Как вы уже заметили, несколько далее, в первой строке, идёт текст зелёного цвета, это – комментарий, программой он не воспринимается и служит только для удобства визуального восприятия каких-либо пометок. А чтобы редактор понял, что это не команды, нужно в начале такого текста поставить апостроф «’» и далее уже можно писать любой текст.

Переходим ко второй строке нашей программы:

nach = InputBox("Начальная строка:")

Так как нам не всегда нужно будет печатать шаблон из всех строк таблицы (были изменены значения в какой-то строке, добавлены новые строки и т.д.), то необходимо сказать программе с какой же строки таблицы надо начинать формировать и печатать шаблон. Для этого мы воспользуемся оператором InputBox(), который при выполнении программы выведет окно, где спросит у нас значение:



Чтобы было понятно, что хочет от нас программа, то в скобках мы указывает тот текст, который нужно вывести в окне, а чтобы редактор понял, где начало этого текста, а где конец – он заключён в кавычки. Полученное из окна значение, нужно где-то запомнить, чтобы потом использовать, и для этого я объявляю переменную «nach», которую приравниваю, полученному из окна запроса, значению.

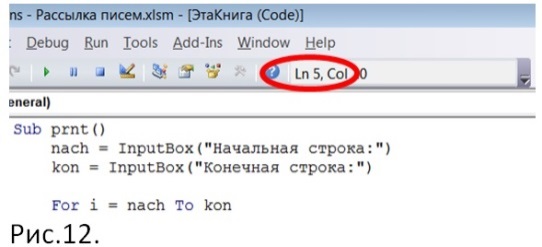
Тоже делаем для конечной строки:

kon = InputBox("Конечная строка:")

Теперь в переменной «nach» у нас хранится начальное значение, а в переменной «kon» - конечное, и мы открываем цикл, в котором будем выполнять одинаковые действия для каждой строки таблицы от начальной (переменная «nach»), до конечной (переменная «kon»):

For i = nach To kon

Это четвёртая строка программы (вообще, по счётчику редактора – пятая, но я специально вставил пустую строку для удобочитаемости кода, и не считаю её):



Для организации цикла от какого-то значения до другого, мы используем специальную конструкцию из операторов:

For To

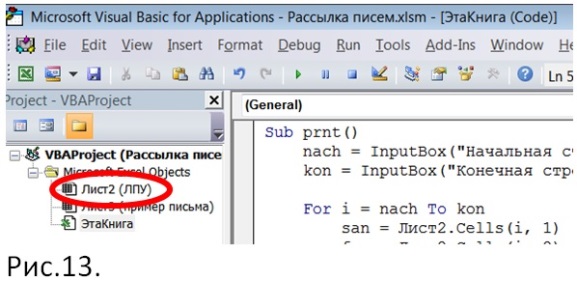
Next

А в качестве счётчика текущей строки, объявляем временную переменную «i», которой присваиваем сначала начальное значение, и, поле оператора To, говорим, до какого значения будет действовать открываемый цикл. Оператор Next, говорит о том, что нужно увеличить счётчик строк на 1 и снова начать обработку содержимого цикла, но уже с увеличенным значением. Те команды (операторы), которые будут находиться между For To и Next, называются телом цикла.

Теперь, начинаем перебирать каждую строку таблицы и рассмотрим первую команду цикла:

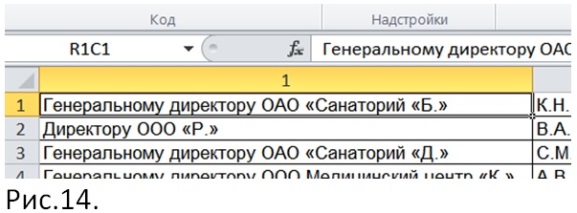
san = Лист2.Cells(i, 1)

Чтобы взять значение из ячейки в нашей таблице, мы должны обратиться к некоему объекту Excel, в котором это значение хранится. Нам известно, что наша таблица находится на листе с именем «ЛПУ», а получить на обработку, мы хотим значение из 1 строки и 1 столбца, этой таблицы. Теперь нужно посмотреть в окне проекта, что за объект называется «ЛПУ»:



Искомый объект, называется Лист2, а в скобках указано его имя, которое мы видим в Excel, оно то и подсказывает нам, что это именно то, что мы ищем. Теперь, зная имя нужного нам объекта, мы обращаемся к его, определённой нами, ячейке. Делается это особым способом (методом), который указывается через точку «.», а затем идёт наименование этого метода, в нашем случае, это – Cells(), где в скобках, через запятую, надо указать сначала номер строки данной ячейки, а затем номер столбца, в котором она находится. Таким образом, получить объект, который хранит значение из первой строки и первого столбца нашей таблицы можно с помощью конструкции:

Лист2.Cells(1,1)



И, в нашем случае, хранить данный объект будет значение «Генеральному директору ОАО «Санаторий «Б.». Но, так как в цикле, номер строки у нас постоянно меняется и хранится в переменной «i», то в теле цикла, описанная конструкция будет выглядеть так:

Лист2.Cells(i, 1)

Вот её то, мы и запоминаем в переменной «san».

Переходим к следующим строкам цикла, где запоминаем в своих переменных, соответствующие значения из следующих столбцов строки:

fam = Лист2.Cells(i, 2)

adr = Лист2.Cells(i, 3)

io = Лист2.Cells(i, 4)

Теперь, если при запуске на выполнение, начальной строкой таблицы было указано «1», наши переменные, будут хранить следующие значения:

san = «Генеральному директору ОАО «Санаторий «Б.»

fam = «К.Н. Кхххххх»

adr = «ул. Семашко, д.100, г.Железноводск, 357400»

io = «Константин Николаевич»

Начинаем их обрабатывать. Чтобы вручную, в четвёртом столбце, не указывать полное обращение «Уважаемый Константин Николаевич!», а оно, в зависимости от рода мужского или женского, выглядит по-разному, в наш цикл мы вставим условие, в зависимости от которого, обращение сформируется автоматически. Для этого воспользуемся конструкцией из операторов («если то иначе всё» - вспомнили информатику в школе?):

If Then

Else

End If

В этом условии, пользуясь особенностями русского языка, мы проверим, если последняя буква в отчестве – «а», то логично предположить, что данное имя – в женском роде, в других же случаях – род мужской и обращение будет другим. У нас же, имя и отчество уже запомнено в переменной «io» и хранит значение «Константин Николаевич». Чтобы взять из текстового значения последний символ, а потом проверить, равен ли он «а», воспользуемся оператором Right(), где в скобках указывается сначала текстовое значение, заключённое в кавычки, а затем число символов, которое надо взять справа. Так как наша строка хранится в переменной «io», а символ справа, нам нужен только один, конструкция, предоставляющая нам последний символ в строке с именем и отчеством, будет выглядеть так:

Right(io, 1)

И, в нашем случае, равно оно будет – «ч». Обращаю внимание, что имя переменной в кавычки мы не заключаем, т.к. переменная уже определена, вот если бы мы хотели из текста «Константин Николаевич» выделить последний символ, то написали бы так:

Right("Константин Николаевич", 1)

И получили бы то же «ч», но мы используем цикл, где значения с каждой строкой меняются, а для этого и определяем переменные, которые будут нам сохранять временные значения.

Итак, в новой строке цикла, объявляем открытие условия:

If Right(io, 1) = "а" Then

А в следующей, то, что надо сделать, если это условие выполняется, т.е. если последний символ имени и отчества равен «а»:

io = "Уважаемая " & Лист2.Cells(i, 4) & "!"

Теперь опишу, какое значение получает переменная «io» в этом случае. Оно формируется из трех частей, при помощи оператора слияния «&», что можно описать так: к текстовому значению «Уважаемая » (замечу, что после слова идёт пробел), приклеить значение из четвёртого столбца очередной строки из таблицы, а затем к полученному, приклеить текстовое значение «!».

Далее идёт строка с оператором Else, который говорит, какие команды надо исполнять, если условие не удовлетворено:

io = "Уважаемый " & Лист2.Cells(i, 4) & "!"

Это как раз наш случай, т.к. условие Right(io, 1) = "а" не выполнено, ведь мы помним, что сейчас у нас Right(io, 1) = "ч". Таким образом, после слияния получаем:

"Уважаемый Константин Николаевич!"

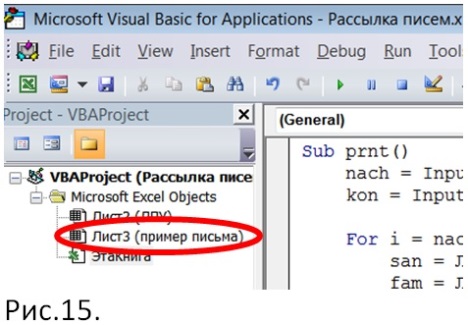
Это то значение мы и запомним в переменной «io».

Следующий оператор закрывает наше условие:

End If

Если до выполнения условия, в нашем случае, переменная «io» хранила значение «Константин Николаевич», то после, там уже будет значение «Уважаемый Константин Николаевич!».

Итак, все переменные определены, хранят нужные нам значения и теперь можно выводить их на шаблон, который, напомню, уже является другим объектом:



А по сему и обращение к его ячейкам будет выглядеть так:

Лист3.Cells(4, 7) = san

Лист3.Cells(8, 7) = fam

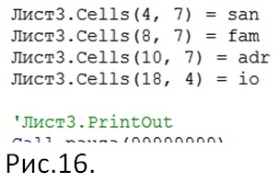
Лист3.Cells(10, 7) = adr

Лист3.Cells(18, 4) = io

Таким образом, все временные значения из текущей строки таблицы, мы размещаем в определённых нами местах шаблона, остаётся только этот шаблон распечатать, для чего используется метод PrintOut, а команда на печать уже готового шаблона будет выглядеть так:

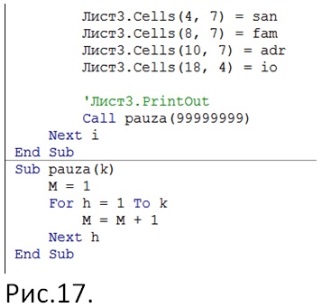
Лист3.PrintOut

Однако у нас, почему-то, перед этой командой стоит апостроф и она зелёного цвета:



Да, так, конечно же, ничего на печать не выведется, а теперь расскажу, почему я закомментировал эту команду. Дело в том, что перед печатью, желательно удостовериться, что всё в шаблон помещается правильно, нигде ничего не вылезло, не пропало, не съехало. Если запускать процедуру на выполнение в таком виде, то мы просто увидим, как в нужных местах на шаблоне появляются разные новые значения, и только если нас всё устраивает, мы удаляем апостроф перед этой командой и запускаем процедуру на выполнение уже с печатью, наверняка зная, что отпечатается всё правильно.

Практика показывает, что если количество сформированных и отправленных на печать шаблонов превышает сотню, то кэш задач принтера набивается и принтер сбивается, начиная выполнять свои задания непоследовательно, а в разнобой, соответственно и порядок выводимых документов не будет соответствовать таблице с исходными данными. Чтобы избежать подобной ситуации, необходимо чтобы принтер успевал выполнять свои задания, а кэш его задач не набивался. Но дело в том, что для современных компьютеров выполнять тот набор команд, который мы ему задали в нашей процедуре, не составляет особого труда и за одну лишь минуту, он может накидать принтеру, таким образом, сотни заданий. Поэтому, наша задача чуть замедлить выполнение данной процедуры, чтобы принтер успевал отрабатывать полученные задания. Для этого была создана дополнительная процедура pauza(k), где в качестве параметра k, передаётся значение задержки. Она расположена ниже нашей основной процедуры «prnt»:



Данная процедура заставляет в цикле перезаписывать переменную M, новыми значениями, увеличивающимися на 1 за каждый цикл, на что уходит определённое время, которое, в конечном итоге, зависит от того, насколько велико значение k, переданное данной процедуре. Опытным путём, это значение я уже рассчитал, оно равно 99999999, однако, при желании, можно поэкспериментировать с ним.

Теперь же, возвращаемся в нашу основную процедуру и в следующей строке цикла, вызываем процедуру, которая делает некоторую паузу в выполнении цикла:

Call pauza(99999999)

Оператор Call, говорит о том, что мы вызываем на выполнение дополнительную процедуру, которая ниже описана, и передаём ей значение 99999999. Следующая строка, завершает наш цикл:

Next i

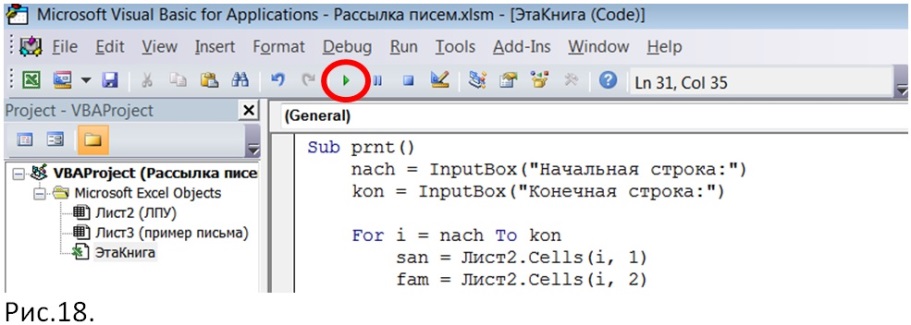
Оператор Next, увеличивает счётчик наших строк, значение «i», на 1 и заставляет выполнять команды цикла уже с новыми значениями.

Последней же строкой нашей процедуры, является объявление о её закрытии:

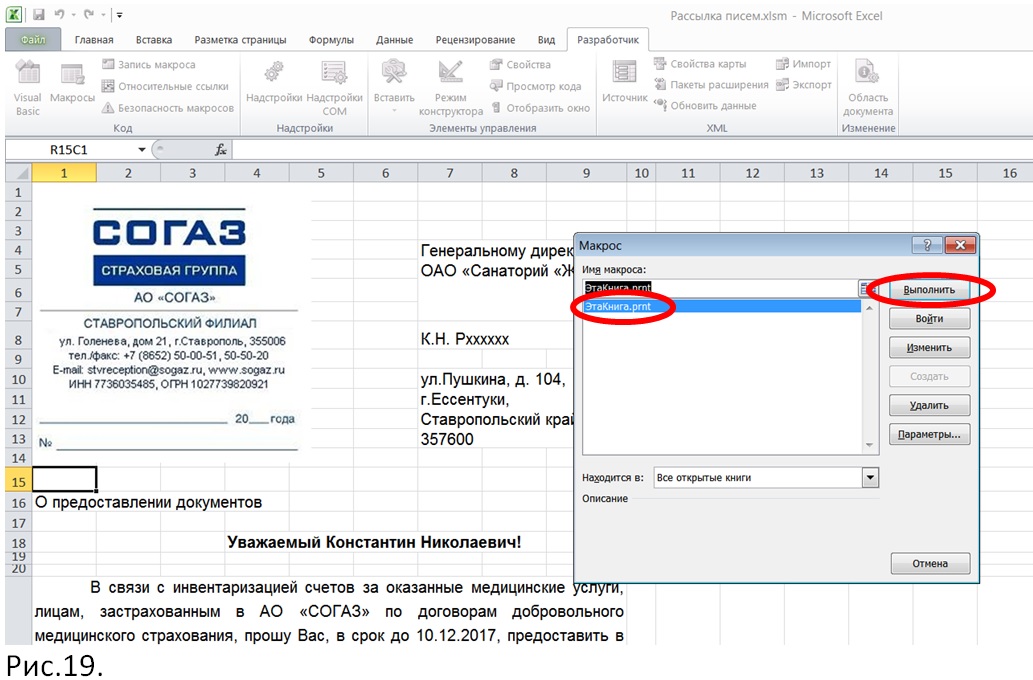
End Sub

После выполнения данной команды, выполнение каких-либо команд прекращается.

Запускать же на выполнение нашу процедуру можно и горячими клавишами, и из редактора, нажав кнопку выполнения (при этом курсор должен быть внутри той процедуры, которую намереваетесь запускать):

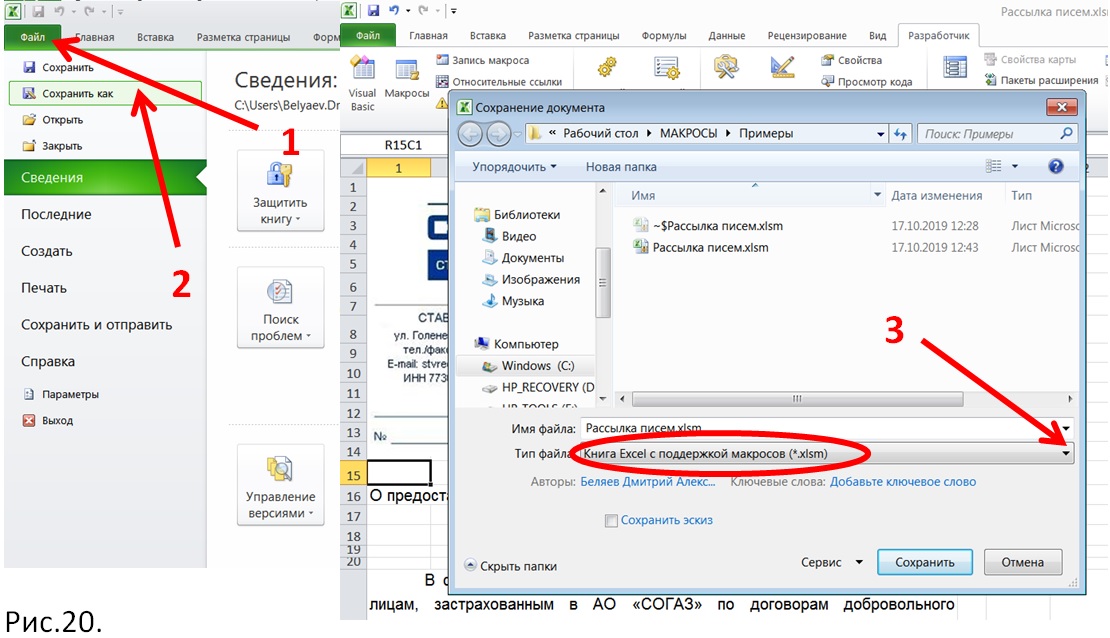


Но запуская, таким образом, окно редактора будет закрывать окно Excel, и мы не увидим, что происходит в самом шаблоне, поэтому лучше сначала активировать тот лист Excel, где у нас шаблон, а затем воспользоваться горячими клавишами – левый Alt+F8:



В появившемся окне, нужно выбрать, какую процедуру мы желаем запустить и, либо дважды на неё кликнуть мышью, либо нажать кнопку «Выполнить». После этого от нас сначала потребуется ввести начальное значение строки в исходной таблице, потом конечное, а затем мы будем наблюдать, как в указанных нами ячейках будут появляться и меняться значения, взятые из таблицы.

Завершая описание первого примера, хочу сразу предупредить, что сохранять файлы, имеющие макросы, необходимо в формате с расширением .xlsm, иначе, написанные макросы, в файле не сохранятся.

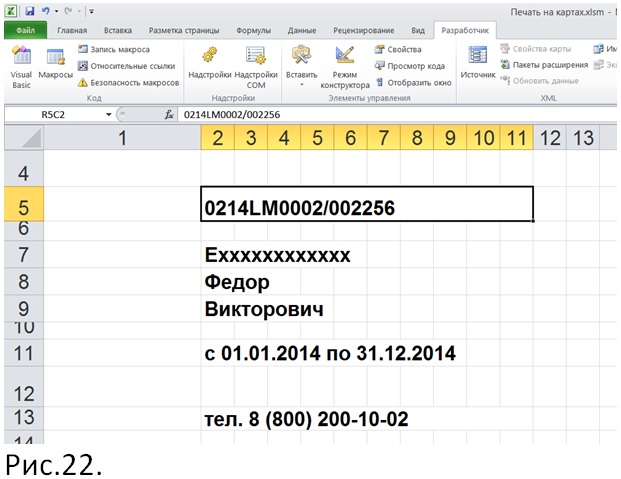


**Пример 1.2. Печать на пластиковых картах.**

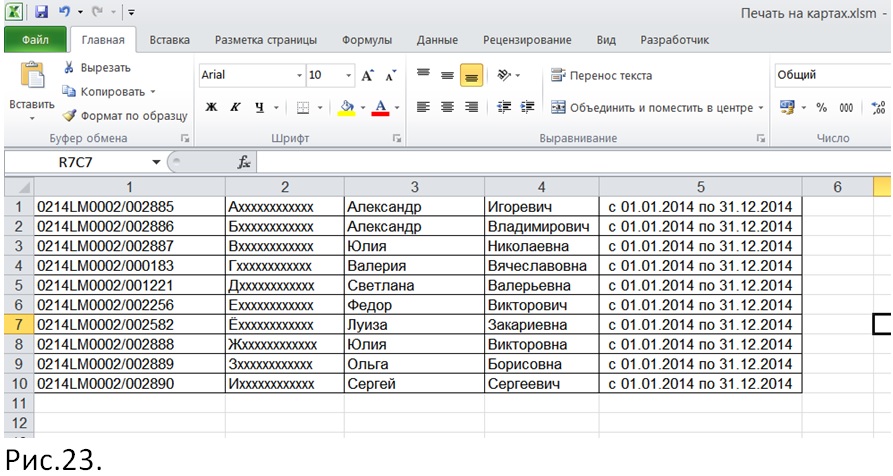
Ещё одна, довольно актуальная, задача – подготовка пластиковых карт по договорам ДМС:



Для печати на пластиковых картах, нашим филиалом был приобретён специальный, сублимационный принтер Evolis Pebble 4, который, после подключения к компьютеру, определяется операционной системой, как обычный принтер. Разница лишь в том, что область печати у него маленькая, которая опытным путём была определена, и, подгоняя границы строк и столбцов, я создал шаблон, для вывода данных на пластиковую карту, разместив его, на листе с именем «Шаблон»:



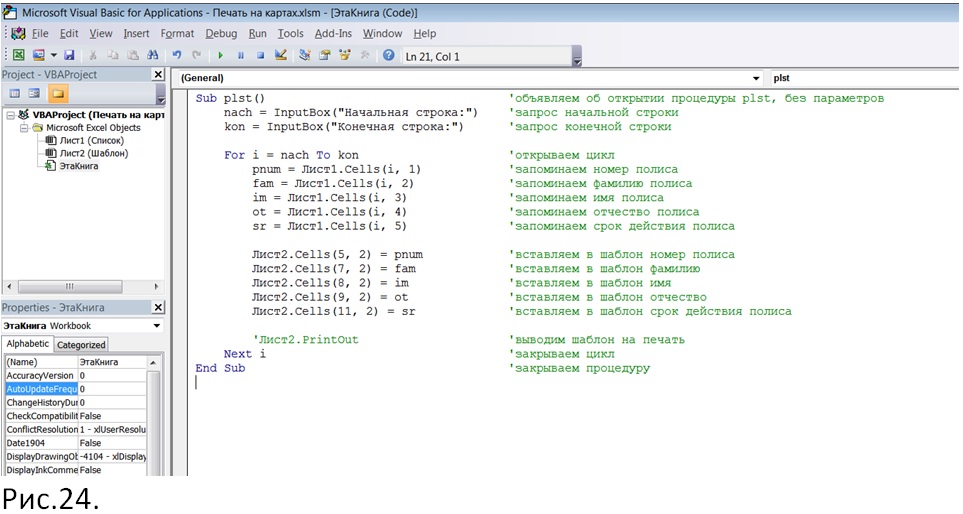
Как и в предыдущем примере, у меня уже была таблица с данными для вывода на печать, на листе «Список», в excel-файле (…\Примеры\Печать на картах.xlsm):



Сначала определяемся, что, куда должно попасть:

1. Номер полиса – строка 5, столбец 2 шаблона;
2. Фамилия – строка 7, столбец 2 шаблона;
3. Имя – строка 8, столбец 2 шаблона;
4. Отчество – строка 9, столбец 2 шаблона;
5. Срок действия – строка 11, столбец 2 шаблона.

Как и мы уже делали ранее, определяемся с объектами и создаём в редакторе процедуру plst():



Команду вывода на печать объекта Лист2, где у нас находится шаблон, мы опять закомментируем и будем убирать комментирование только в случае уже проверенной и подготовленной печати. Еще, очень важно помнить, что команда печати использует принтер, установленный в системе по умолчанию, и, в большинстве случаев, это обычный принтер, поэтому перед запуском на печать, текущим принтером, необходимо установить сублимационный.

Подробно описывать команды я не стану, они уже очевидны из прошлого примера. В общем, можно сказать лишь о том, что данная процедура спрашивает начальную и конечную строку таблицы, где находятся исходные данные, затем организует цикл, в котором сначала запоминает в соответствующих переменных значения из таблицы, а потом выводит их в нужных местах в шаблоне.

Логично предположить, что можно обойтись и без промежуточных переменных, и приравнивать ячейки из таблицы, сразу ячейкам из шаблона, тогда тело цикла будет выглядеть так:

Лист2.Cells(5, 2) = Лист1.Cells(i, 1)

Лист2.Cells(7, 2) = Лист1.Cells(i, 2)

Лист2.Cells(8, 2) = Лист1.Cells(i, 3)

Лист2.Cells(9, 2) = Лист1.Cells(i, 4)

Лист2.Cells(11, 2) = Лист1.Cells(i, 5)

Лист2.PrintOut

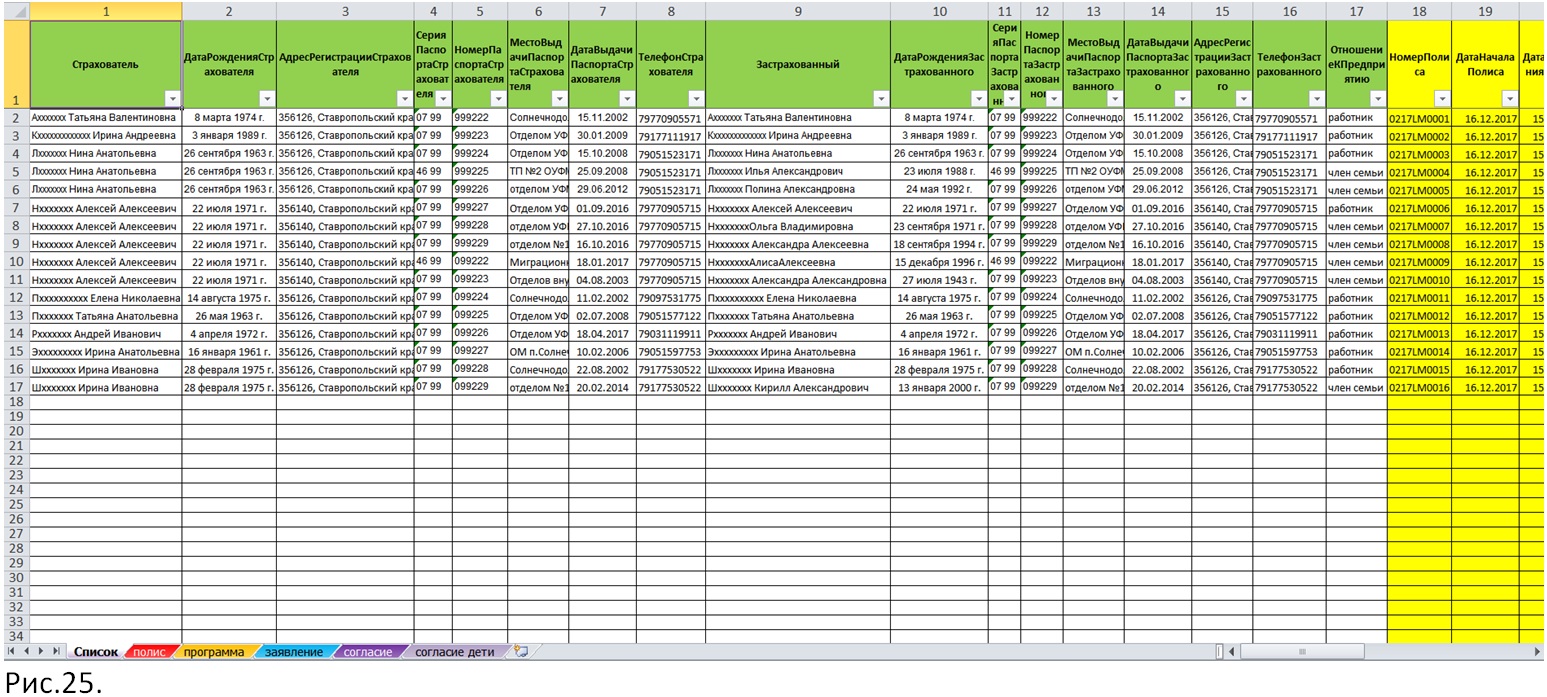
Соглашусь, но лишь для данного, простого примера, ведь когда команд становится много, то ориентироваться в именах переменных гораздо легче, чем в индексах объектов.

**Пример 1.3. Коллективный договор ДМС.**

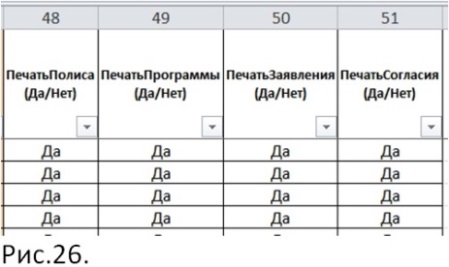
Завершу описание работы с шаблонами, примером комплексного решения по коллективному договору ДМС, с печатью сопутствующих документов, включая вывод шаблона полиса на фирменном бланке. Особенностью данного договора является то, что застрахованы не только работники предприятия, но и члены их семей. При этом, каждому страхователю выдаётся целый комплект документов:

1. Полис на бланке;
2. Программа страхования;
3. Заявление в бухгалтерию предприятия;
4. Согласие на обработку персональных данных (с учётом детей).

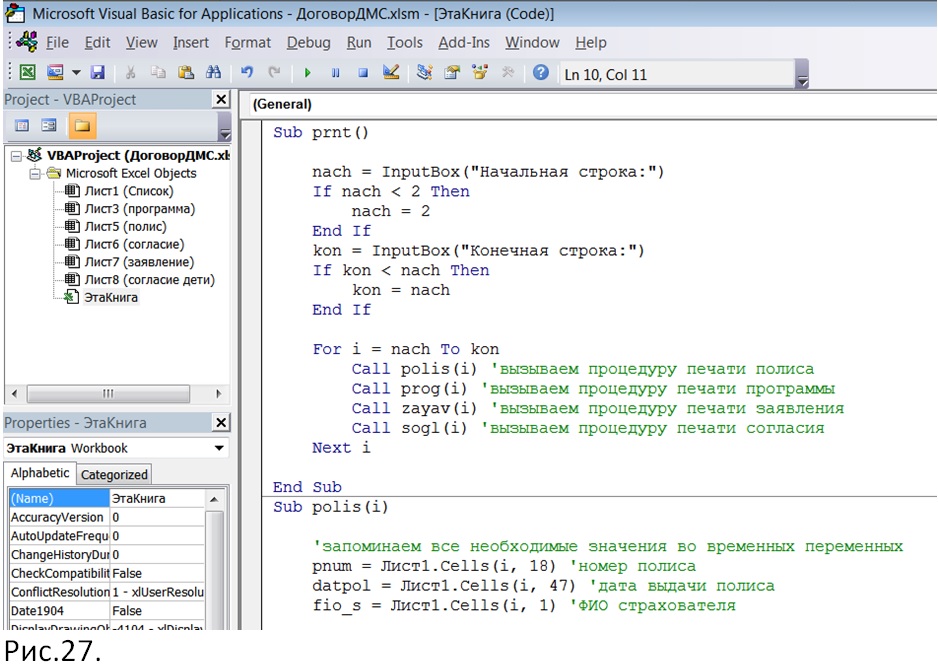
Как и ранее, перед созданием шаблонов, у нас должна быть исходная таблица, в которой уже есть все необходимые данные (…\Примеры\ДоговорДМС.xlsm):



Ещё одной интересной особенностью данного договора, является рассроченная оплата, которая уже рассчитана и указана в соответствующих полях таблицы. Перед запуском процедуры prnt() на выполнение, необходимо не только проверить полноту исходных данных, но и обратить внимание на последние, управляющие столбцы исходной таблицы:



Управляющие они потому, что включают и отключают печать тех или иных документов из всего комплекта. Например, если необходимо распечатать по какому-то страхователю один комплект, то во всех колонках должно стоять «Да», тогда, положив бланк полиса в принтер, можно запустить процедуру по одной, необходимой нам строке, при этом сначала распечатается полис на бланке, а затем на обычной бумаге, оставшиеся документы. Если же необходимо распечатать всех страхователей сразу, то «Да» нужно поставить только в столбце «ПечатьПолиса», а в принтер заложить стопку бланков, на которых сначала распечатаются все полисы, а затем убрать «Да» из этого столбца, а в графах «ПечатьПрограммы», «ПечатьЗаявления», «ПечатьСогласия» - поставить, чтобы печать остальных документов была уже на обычной бумаге.



Как можно увидеть на рисунке 27, в нашей процедуре prnt(), мы сначала запрашиваем начальное и конечное значение, а затем, в цикле, по очереди передаём значение текущей строки соответствующим процедурам печати, которые находятся ниже. Детально описывать их, думаю уже не нужно, учитывая тот факт, что практически все строки там закомментированы, что поможет понять весь механизм работы, без особых затруднений.

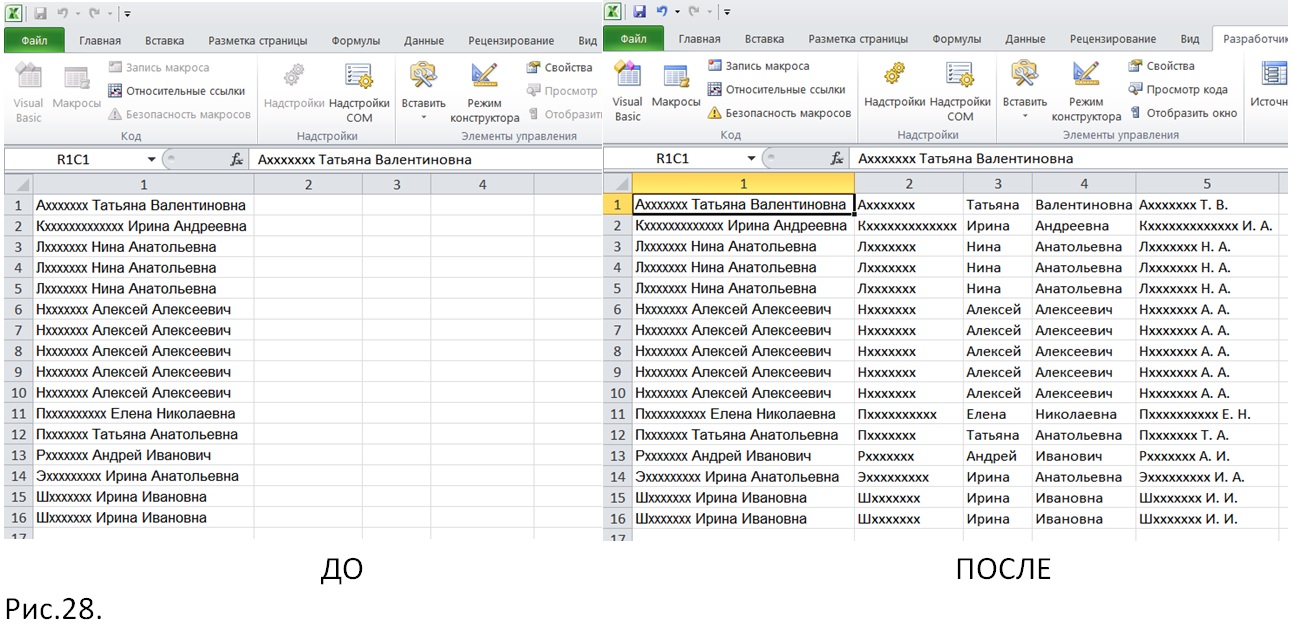
В заключении данного примера, необходимо напомнить, что границы печати на разных принтерах, разные, и, возможно, кое-где шаблоны не будут помещаться в область печати. В таком случае, двигая границы ячеек, нужно будет подгонять шаблон. Ещё, можно отметить, что, так как программа по договору довольно длинная и занимает три страницы, то целесообразно запускать печать на принтерах, поддерживающих двухстороннюю печать, что позволит существенно сэкономить бумагу.

**2.ТАБЛИЦЫ.**

Используя навыки, полученные из предыдущей части, переходим к рассмотрению вопросов, связанных с подготовкой исходных таблиц, что, зачастую, требует больше трудозатрат, чем создание шаблона. В следующих примерах, мы рассмотрим наиболее часто встречающиеся задачи по обработке табличных данных. Для удобства, они собраны в одном файле (…\Примеры\ ОбработкаТаблиц.xlsm).

**Пример 2.1. Деление ФИО.**

Если, с помощью Excel-функции «СЦЕПИТЬ», можно без труда объединить две ячейки в одну, то обратная операция, может вызвать некоторые затруднения. А подобная ситуация при обработке списков застрахованных, возникает довольно часто. На листе «ДелениеФИО», нашего файла с макросами, в первом столбце мы размещаем данные по ФИО и запускаем процедуру razdel():



В результате выполнения, каждая строка с ФИО, будет разделена на фамилию, имя, отчество и фамилию с инициалами, что тоже иногда надо, особенно при работе с шаблонами.

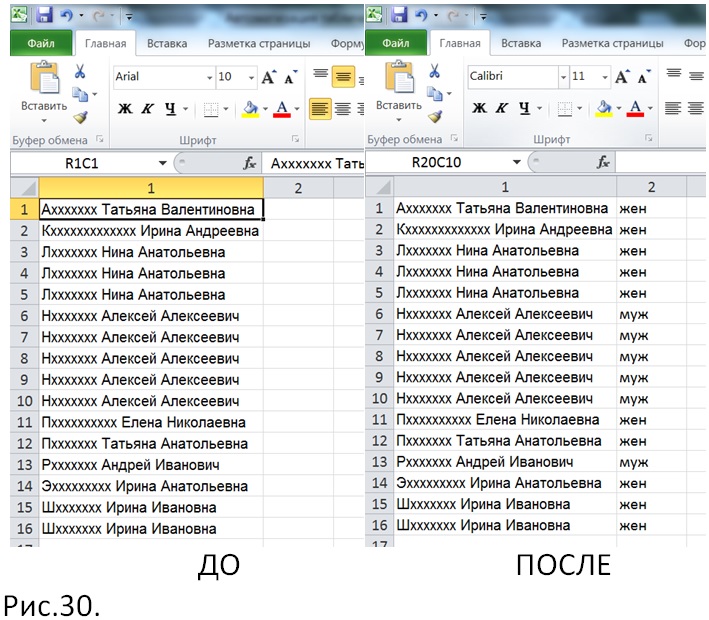
**Пример 2.2. Удаление лишних пробелов.**

Часто случается, что в полученных исходных данных, находятся лишние пробелы. Конечно же, удалить их можно и с помощью встроенного инструмента Excel, указав, допустим, что два пробела нужно заменить одним, но если их много, то такую замену нужно будет запускать несколько раз. Процедура probel(), сразу поудаляет не только лишние пробелы внутри строк, но и те, что иногда встречаются в начале строки или в её конце. А я напомню, что, допустим, для определения пола, мы берём последний символ в строке, и, если он не «а», то пол будет определён как мужской. Так вот, если в конце отчества застрахованной женского пола, будет стоять пробел, а визуально мы это не определим, то процедура запишет данную застрахованную в мужской род, так как пробел никак не равен «а». Чтобы подобного не случалось, данные надо обрабатывать на наличие лишних пробелов. Для этого, их нужно будет поместить в первом столбце, на листе «ЛишниеПробелы», а затем запустить процедуру probel() на выполнение:



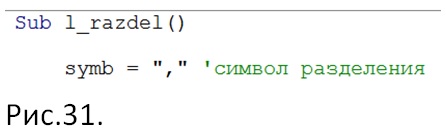
**Пример 2.3. Определение пола.**

Теперь, когда данные очищены от лишних пробелов, можно определять пол. Заходим на лист «ОпределениеПола», вставляем данные в первый столбец и запускаем процедуру pol():

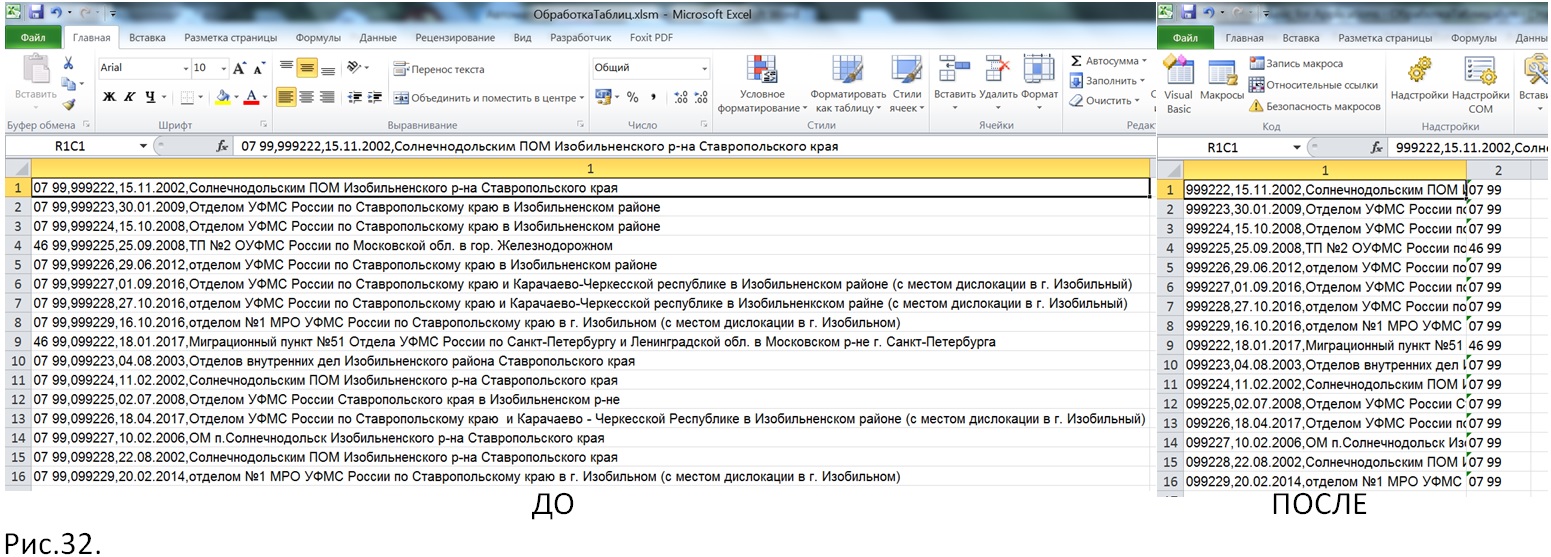
****

**Пример 2.4. Разделение слева.**

Бывает, особенно при работе с паспортными данными, что они приходят к нам одной строкой, разделённой запятыми, и их, иногда, приходится разделять. Для этого создана процедура l\_razdel(), которая берёт данные из первого столбца на листе «РазделениеСлева». Перед запуском процедуры, необходимо проверить, что же она будет считать символом разделения, ведь иногда, это может быть не только запятая, но и пробел, или символ «\*», да и, в принципе, любой.

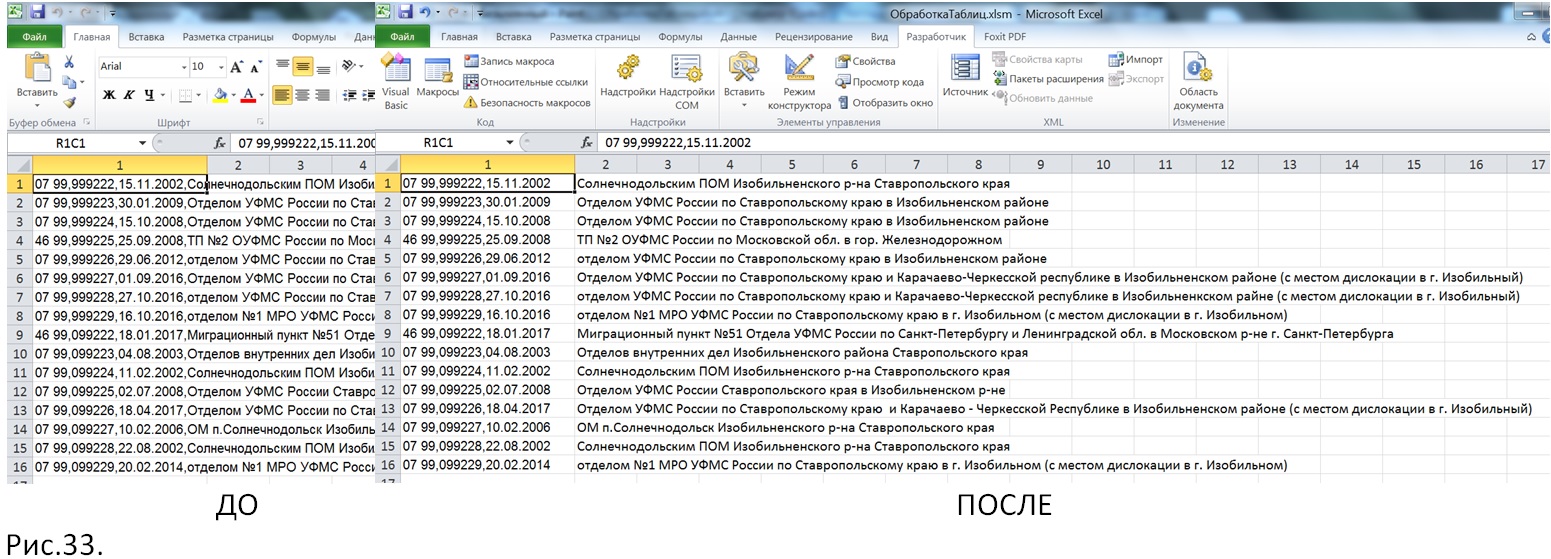


Запускаем процедуру и она, по предложенному символу, отрезает из текущей строки слева, ту часть текста, которая стоит до разделителя и вставляет в отдельную ячейку. Скопировав появившийся столбец, можно запустить процедуру ещё раз и отрезать очередную часть текста. Таким образом, мы сможем разделить всю оставшуюся строку.

****

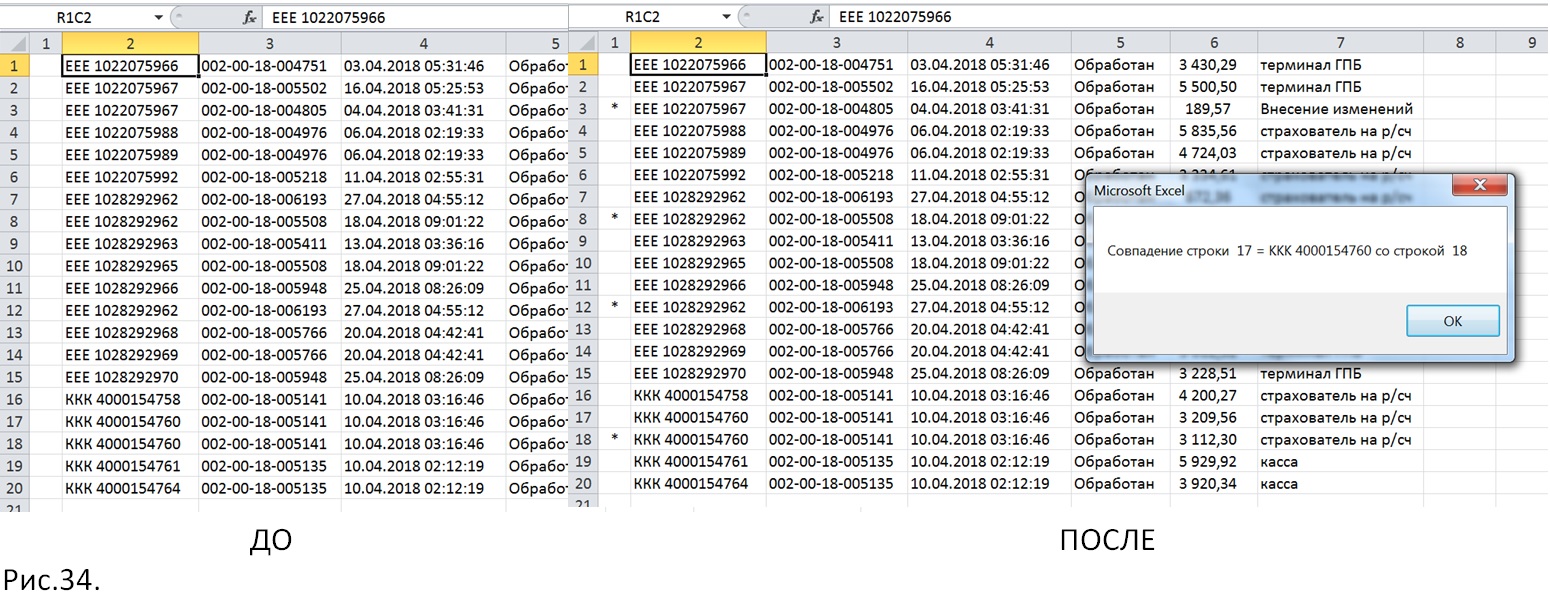
**Пример 2.5. Разделение справа.**

Но бывают случаи, когда отделить по разделителю нужно с конца строки, для этого есть процедура p\_razdel(), а данные для неё мы вставляем в первый столбец листа «РазделениеСправа»:

****

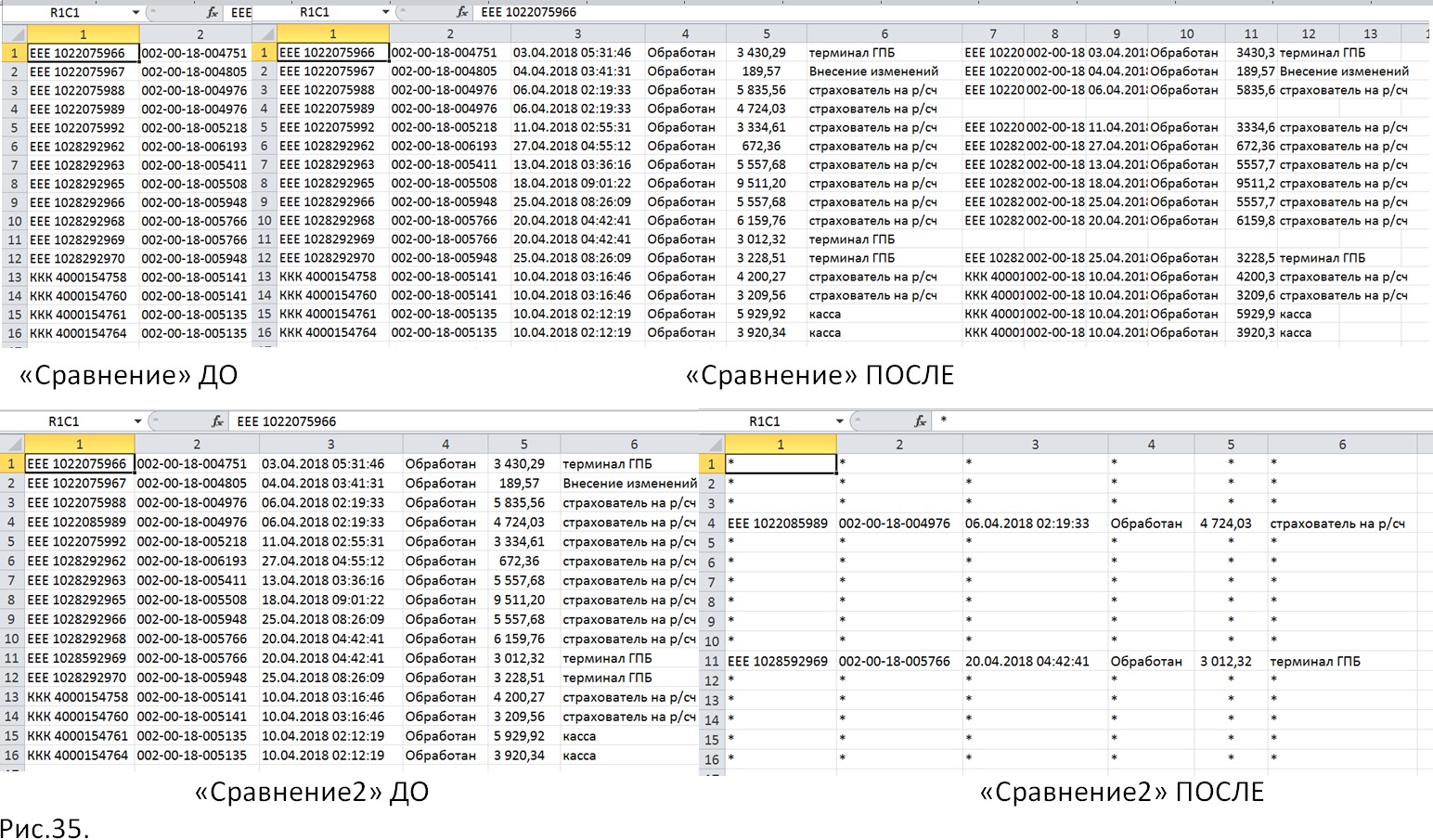
**Пример 2.6. Поиск дублей.**

Ещё одной очень распространённой задачей, является поиск дублированных строк в таблице, и если таблица у нас маленькая, то найти их можно и визуально. Однако, когда строки таблицы исчисляются тысячами, потребуются дополнительные усилия. Облегчит эту задачу процедура dbl(), которая обрабатывает данные на листе «ПоискДублей». Особенность её в том, что данные нам необходимо вставлять не в первый столбец листа, а во второй, так как первый, остаётся для того, чтобы установить там признак дублирования:



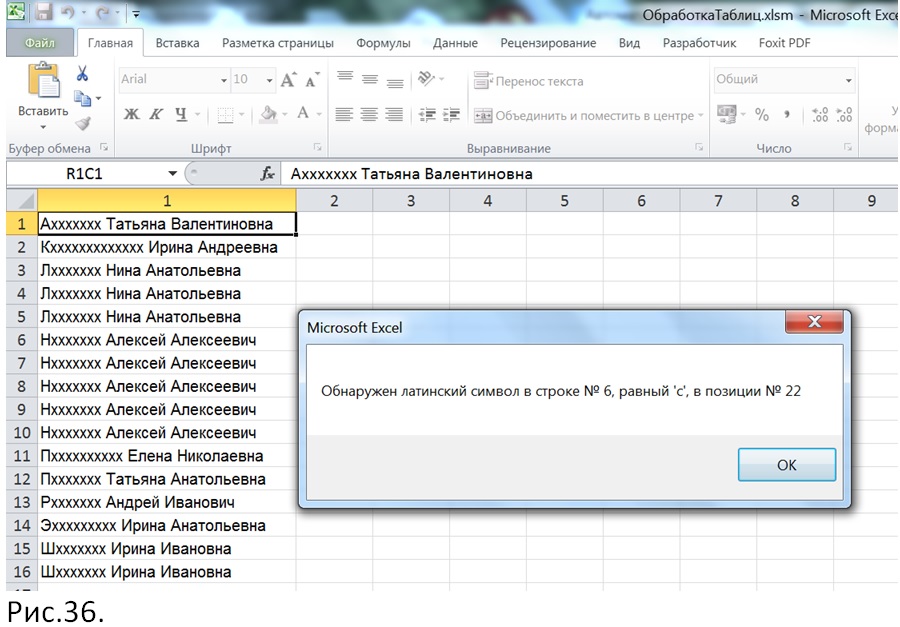
**Пример 2.7. Сравнение таблиц.**

Существует множество способов сравнения данных в таблицах, и так как задача это довольно актуальная, то и на этот случай есть своя процедура – sravn(). Работает она уже с двумя листами. На первом – «Сравнение», мы располагаем таблицу, с которой будем сравнивать другую, расположенную на листе «Сравнение2», причём на обоих листах, столбец, по которому будет происходить сравнение, должен быть первым на листе. Расположив, таким образом, обе таблицы, запускаем процедуру сравнения, и она будет сравнивать каждую ячейку первого столбца второго листа, с каждой ячейкой первого столбца первого листа сравнения, и при наличии совпадения, перенесёт всю строку со второго листа на первый, напротив строки совпадения. На втором же листе, в этом месте, появятся звёздочки, сигнализируя о переносе данной строки.

****

**Пример 2.8. Проверка символов.**

В своей практике, мне не однократно приходилось сталкиваться со случаями, когда при сравнении визуально, строки вроде бы совпадают, однако программа ни в какую не желает признавать их равными. Причиной тому, являлось нахождение в русском тексте, латинских символов, которые на экране выглядят также, однако программой воспринимаются по-разному. Особенно часто, везёт с буквой «с», которая, как известно, на клавиатуре расположена одинаково и для русской раскладки и для латинской. Опять же, при определении пола, латинская буква «а» в конце отчества застрахованной, даст нам мужской род при обработке. Процедура symbol(), берёт по очереди каждую ячейку первого столбца на листе «ПроверкаСимволов» и побуквенно начинает проверять наличие в тексте символов латинского алфавита, сигнализируя о нахождении и месте его расположения в тексте.



**Пример 2.9. Отправка по EMAIL.**

Завершить, предложенные примеры, хочу ещё одной процедурой, которая может понадобиться в нашей деятельности – email(), где по данным, расположенным на листе «РассылкаEMAIL», сформирует и отправит, с помощью текущего профиля приложения Outlook, электронные письма.



К сожалению, а может быть и к счастью, но, по умолчанию, у Outlook есть защита от подобной рассылки, и каждое сформированное письмо будет запрашивать разрешение на свою отправку. При необходимости, данная защитная функция отключается в параметрах безопасности приложения, и тогда за минуту, можно отправить сотни писем, только для этого потребуются права администратора.

**ЗАКЛЮЧЕНИЕ.**

Завершая данную работу, хотел бы обратить внимание на тот факт, что рассмотренные здесь варианты, не являются единственно верными решениями поставленных задач. Многое можно сделать и используя стандартные формулы Excel, что-то – более изящными макросами. Однако, своей целью, я ставил не только предложение готовых решений для распространённых ситуаций, а ещё и демонстрацию простоты и доступности для понимания механизма работы с макросами, которые теперь, при необходимости, можно без труда изменить под свои нужды, а получив даже такие минимальные навыки, пробовать сделать уже что-то своё. Понятно, что подобное упрощение скрывает поистине огромные возможности, заложенные в средства VBA, но при желании, расширить свои познания не составит особого труда, так как сейчас на просторах интернета можно найти массу информации по данной тематике. Кроме того, если у кого-то возникнут какие-либо вопросы по затронутой в этой работе теме, что-то вдруг будет не получаться, или же понадобится помощь в реализации каких-то своих идей для автоматизации, то, конечно же, я готов помочь, посоветовать, поделиться своим опытом.

30 ноября 2019г.

С уважением,

Беляев Дмитрий

Системный администратор

Ставропольского филиала АО "СОГАЗ"

Тел.: (8652) 50-50-20, доб. 220

IP тел.: 702600220

[belyaev.dmitriy@sogaz.ru](mailto:belyaev.dmitriy@sogaz.ru)

<http://webtutor.sogaz.ru/view_doc.html?mode=doc_type&custom_web_template_id=6743541225986353286&doc_id=6743542595786337893>