**Регулярные выражения VBA**

*Заметка написана Андреем Макаренко*

Регулярные выражения (regular expressions) — очень мощный механизм для обработки строк. С его помощью можно найти нужные части текста, проверить, удовлетворяет ли строка определённой маске, заменить найденный текст. Такие выражения вcтроены во многие языки программирования, такие, как Perl, Php, JavaScript, и, конечно VBA.

Рассмотрим, как это работает на примере обработки счетов на доставку товаров. Исходные данные содержатся в отчете Excel в форме полного адреса (рис. 1). Наша задача из строки вида «677000, Россия, Саха /Якутия/ Респ., г. Якутск, ул. Ойунского» выделить название города.

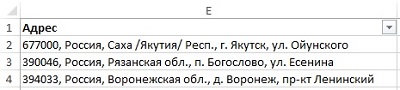


Рис. 1. Исходные данные

Для использования регулярных выражений, необходимо подключить библиотеку *MS Windows Script*. Для подключения, запускаем VBA (меню *Разработчик* –> *Visual Basic*). В открывшемся окне VBA проходим по меню *Tools* –> *References* и в окне *References - VBAProject* ставим флажок в строке *Microsoft VBScript Regular Expressions 5.5* (рис. 2).

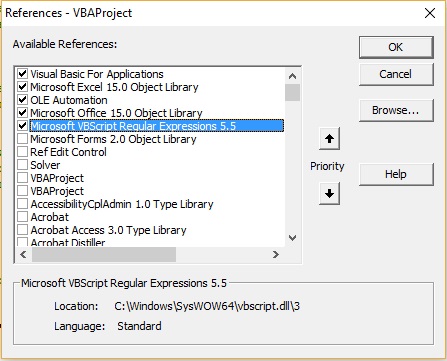


Рис. 2. Подключение библиотеки *MS Windows Script* для работы с регулярными выражениями

Теперь в Visual Basic добавился объект RegExp, который содержит в себе всё, что нужно для работы с регулярными выражениями. Код, выполняющий поставленную задачу, выглядит так:

Sub RegExp()

Dim myRegExp As New RegExp ' создаем экземпляр RegExp

Dim aMatch As Match ' один из совпавших образцов

Dim colMatches As MatchCollection ' коллекция этих образцов

Dim strTest As String ' тестируемая строка

'устанавливаем свойства объекта RegExp

myRegExp.Global = False ' если Global = True, то поиск ведётся во всей строке, \_

если False, то только до первого совпадения

myRegExp.IgnoreCase = True ' игнорировать регистр символов при поиске

myRegExp.Pattern = ", (г|c|п)\. .\*?," 'шаблон для поиска

strTest = Sheets("Накладные").Range("e2").Text 'присваиваем переменной текст из текущей ячейки

Set colMatches = myRegExp.Execute(strTest) 'получаем коллекцию совпадений с образцом

'перебираем коллекцию и просматриваем результаты

For Each aMatch In colMatches ' проходим по всей коллекции

a = aMatch.FirstIndex 'порядковый номер первого символа найденного образца

b = aMatch.Length 'кол-во символов в найденном образце

c = aMatch.Value 'полный образец

Next aMatch

c = Mid(c, 6, Len(c) - 6)

MsgBox a & " | " & b & " | " & c ' смотрим, что получилось

'производим замену найденного выражения

d = myRegExp.Replace(strTest, " (здесь раньше был город)")

MsgBox d 'смотрим, что получилось

End Sub

В примере мы рассмотрели работу с одной ячейки. Для обработки массива данных, нужно организовать цикл, двигаясь по строчкам, например, до строчки с пустым значением.

Отдельного рассмотрения заслуживает синтаксис создания шаблона для поиска. Возможности его поистине неисчерпаемы. Несколько лет назад я работал в издательстве телегида. Еженедельно требовалось в ограниченные сроки подготовить тексты телепрограмм для публикации. В приведении их к единому стандарту были заняты три редактора и два верстальщика в течение нескольких часов. Программная обработка с использованием регулярных выражений занимала несколько минут после чего один корректор и один верстальщик «подчищали» результат в течение получаса. Реальная проблема, с которой мне пришлось столкнуться, это потеря предсказуемости результата при построении слишком сложных конструкций. Поэтому рекомендую вместо одной сложной использовать несколько последовательных трансформаций.

Описание синтаксиса для создания шаблона можно найти в Интернете (см., например, [msdn](https://msdn.microsoft.com/library/ae5bf541(v=vs.100).aspx)). Здесь я дам лишь краткое представление о возможностях (рис. 3) и прокомментирую шаблон, использованный в примере.

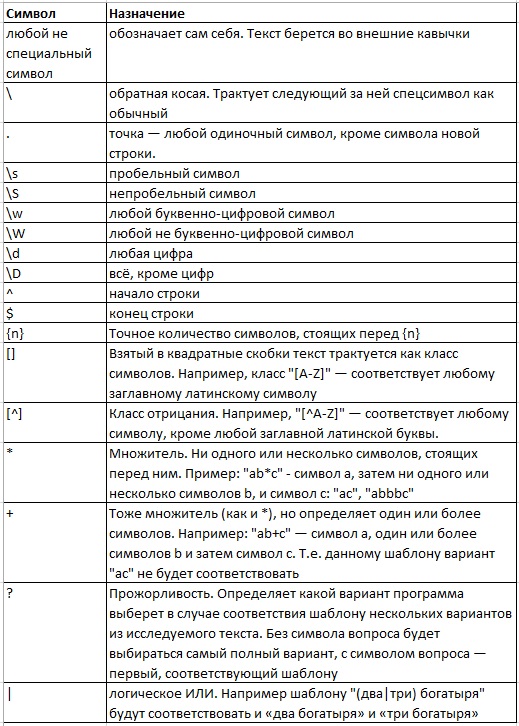


Рис. 3. Специальные символы

Разберем использованный нами в примере шаблон:

", (г|c|п)\. .\*?,"

Сам шаблон берется в кавычки. Начинается шаблон с запятой и пробела, далее следует конструкция (г|c|п), которая позволят выбрать один из трех вариантов обозначения населенного пункта: г – город, с – село, п – поселок.

\. — это просто точка. Обратная косая поставлена для того, чтобы отличить ее от спецсимвола

пробел — это и есть пробел

. — точка — любой символ

\* — множитель, берет ни одного или множество символов, стоящих слева от него (а это точка – то есть, любой символ) между пробелом и запятой

? — ограничивает поиск первой встреченной запятой. Если его не использовать, то выражение вернет значение «, г. Якутск, ул. Ойунского,». т.е., до последней запятой в тексте.

Итак, шаблон позволяет найти текст, который начинается с запятой, пробела, буквы («г», «с» или «п») с точкой и пробела, а заканчивается первой встреченной после этого запятой. В нашем случае шаблон извлекает следующий текст:

, г. Якутск,

Далее оператор c = Mid(c, 6, Len(c) - 6) оставляет от него лишь

Якутск