**Бен Форта. Регулярные выражения за 10 минут**

Когда я работал в издательстве, то очень активно пользовался обработкой текста с помощью шаблонов. Тогда я работал в программе PageMaker (ныне InDesign), и использовал язык скриптов. Позже я перенес этот опыт на обработку текста в Word и Excel, но использовал в макросах возможности, предоставляемые самими программами (рис. 1). А совсем недавно Андрей Макаренко опубликовал [заметку](http://baguzin.ru/wp/?p=16527), показывающую мощь регулярных выражений VBA. Поэтому, когда я увидел, что издательство Вильямс готовит повторное издание книги для начинающих в этой области, то решил изучить предмет глубже. В книге Бена Форта представлены все наиболее важные сведения о регулярных выражениях: основные понятия и концепции, наборы символов, метасимволы, повторители, поиск позиции, подвыражения, ссылки назад, контекстный поиск. Книга написана доступным, простым языком, и является отличным введением в предметную область.

Бен Форта. Регулярные выражения за 10 минут. – М.: Вильямс, 2017 г. – 192 с.



Купить книгу в [Ozon](http://www.ozon.ru/context/detail/id/140304603/?partner=baguzin) или [Лабиринте](http://www.labirint.ru/books/586986/?p=13320)

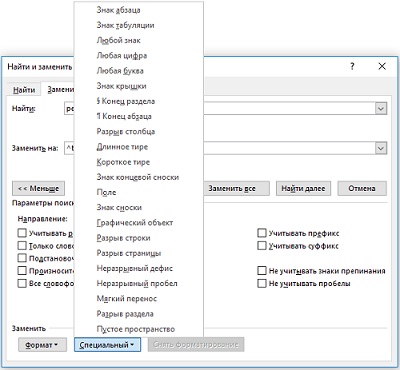


Рис. 1. Использование специальных символов при поиске и замене в Word

### Урок 1. Знакомство с регулярными выражениями

*Регулярные выражения* — это строки, которые используются для поиска и обработки текста. Регулярные выражения записываются на языке регулярных выражений, т.е. на специализированном языке, разработанном и предназначенном для решения задач поиска и замены текста. Для регулярных выражений нет никакой особой программы; это не приложение, которое вы выполняете, и не программное обеспечение, которое вы можете купить или загрузить. Язык регулярных выражений реализован в большом количестве программ. В некоторых языках программирования имеются функции или классы объектов, которые экспортируют функциональные возможности регулярных выражений.

Чтобы вам было легче освоить материал, вы можете загрузить приложение Regular Expression Tester (испытатель регулярных выражений) с [Web-страницы](http://forta.com/books/0672325667/) этой книги.

### Урок 2. Поиск отдельных символов

Символ . (точка) соответствует любому символу. Чтобы найти точку используйте защиту любого специального символа – \ (наклонная черта влево). Наклонная черта влево \ — метасимвол – термин, обозначающий символ со специальным значением, в отличие от обычного символа. В регулярных выражениях \ всегда используется для того, чтобы отметить начало блока из одного или нескольких символов, которые имеют специальное значение. Если нужно найти наклонную черту влево \ (т.е. выполнить поиск \), используйте \\ (две наклонных черты влево).

### Урок З. Соответствие набору символов

Чтобы найти один из двух символов, например, n или s, используйте набор символов, определяемый с помощью метасимволов [ и ]. Символы [ и ] определяют набор символов, состоящий из всех символов между ними. Любой член набора может соответствовать символу, с которым он совпадает (рис. 2).

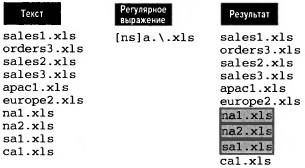


Рис. 2. Соответствие одному из нескольких символов

Используемое здесь регулярное выражение начинается с [ns]; это соответствует либо n, либо s (но не с или любому другому символу). Символы [ и ] не соответствуют никаким символам — они определяют набор. Литерал а соответствует а, точке . соответствует любой символ, наклонной черте с точкой \. соответствует ., а литералу xls соответствует xls. Наборы символов часто используются для выполнения операций поиска (или определения некоторых частей), не зависящих от регистра (рис. 3). Используемый шаблон содержит два набора символов: [Rr], которому соответствуют символы R и r, и [Ее], которому соответствуют символы Е и е. Именно поэтому мы нашли как RegEx, так и regex. Однако цепочка REGEX не была бы найдена.

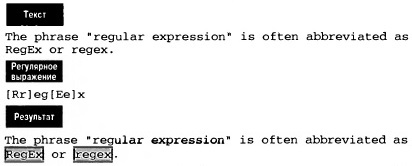


Рис. 3. Использование набора символов для поиска, не зависящего от регистра

При работе с регулярными выражениями часто приходится определять диапазоны символов (от 0 до 9, от А до Z, и т.д.). Чтобы упростить работу с символьными диапазонами, в регулярных выражениях для определения диапазонов используется специальный метасимвол: - (дефис). При использовании диапазонов следите, чтобы конец диапазона не был меньше, чем начало диапазона (диапазон типа [3-1] не допустим). Такая ошибка часто делает неработоспособным весь шаблон. Дефис (-) рассматривается как специальный метасимвол только когда он используется между [ и ]. Вне набора - является литералом и будет соответствовать только -. Несколько диапазонов можно объединить в один набор. Например, [A-Za-z0-9].

Иногда нужно указать соответствие с чем угодно, кроме того, что указано в списке шаблона. Предварите то, что вам не подходит метасимволом ^. Например, шаблон [ns]а[^0-9]\.xls найдет sam.xls, но не sa1.xls.

### Урок 4. Использование метасимволов

Любой метасимвол можно защитить предшествующей ему наклонной чертой влево.

Метасимволы относятся к двум категориям: те, которые используются для поиска текста (они соответствуют искомому тексту; к этому типу относится, например, точка .), и те, которые используются как часть синтаксиса регулярного выражения (к этому типу относятся, например, [ и ]). Начнем с метасимволов, соответствующих пробельным символам (рис. 4).

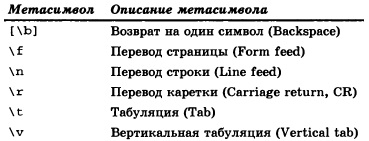


Рис. 4. Метасимволы, обозначающие пробельные символы

Следующий блок текста содержит ряд записей в формате с разделителями-запятыми (часто этот формат называется CSV). Перед обработкой записей из данных нужно удалить все пустые строки (рис. 5).



Рис. 5. Поиск пустых строк

Выражение \r\n соответствует комбинации перевода строки с переводом каретки, используемой (в Windows) как маркер конца строки. Поэтому \r\n\r\n соответствует двум маркерам конца строки, т.е. пустой строке между двумя записями.

Вместо обычно используемых наборов могут использоваться специальные метасимволы. Говорят, что эти метасимволы соответствуют классам символов (рис. 6, 7). Обратите внимание, что синтаксис регулярных выражений чувствителен к регистру. Метасимволы \d и \D имеют различное значение. Также обратите внимание, что классы POSIX сами по себе заключаются в «скобки» [: и :]; поэтому, если классы POSIX использовать внутри наборов, синтаксис будет включать двойные скобки, например [[:rxdigit:][:rxdigit:]].

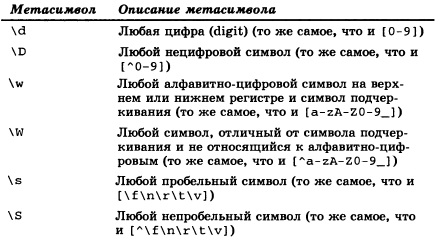


Рис. 6. Метасимволы классов символов

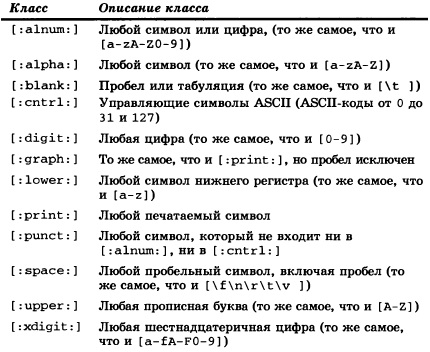


Рис. 7. Символьные классы POSIX

### Урок 5. Повторение совпадений

Чтобы установить соответствие символа (или набора) шаблона с одним или несколькими символами, просто добавьте в конец шаблона символ + (рис. 8).

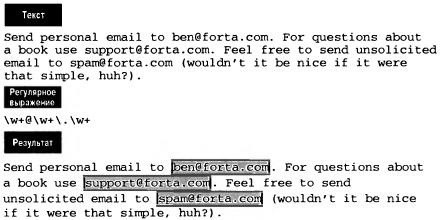


Рис. 8. Поиск соответствия с одним или несколькими символами

Шаблон в точности соответствует всем трем адресам. Регулярное выражение с помощью \w+ сначала находит один или несколько алфавитно-цифровых символов. Затем устанавливается соответствие с @, после чего снова используется \w+, чтобы установить соответствие с одним или несколькими символами, следующими за @. Затем устанавливается соответствие с точкой . (используется защищенная точка \.) и еще один шаблон \w+, чтобы установить соответствие с концом адреса.

Если нужно установить соответствие с необязательными символами, т.е. с такими символами, используйте метасимвол \*. Метасимвол \* используется в точности так, как +; он записывается сразу после символа или набора и будет соответствовать нулю или большему количеству вхождений символа или шаблона.

Знак вопроса ? соответствует только отсутствию вхождений или ровно одному вхождению символа (или набора). Знак ? очень полезен тогда, когда нужно установить соответствие с одним конкретным необязательным символом в блоке текста. Например, если нужно найти адрес с http или https, используйте шаблон https?://[\w./]+.

Чтобы обеспечивать большую свободу управления повторением совпадений, в регулярных выражениях допускается использование интервалов. Интервалы определяются между символами { и }. Например, {3} означает поиск соответствий с тремя экземплярами предыдущего символа или набора. Чтобы определить диапазон количества вхождений (от минимального до максимального значения количества вхождений шаблона), используйте следующий синтаксис {2,4}. Этот диапазон задает 2 в качестве минимального значения для количества вхождений шаблона и 4 — в качестве максимального значения для количества вхождений шаблона.

Например, для проверки правильности формата дат, используйте шаблон \d{1,2}[-\/]\d{1,2}[-\/]\d{2,4}. Шаблону \d{1,2} соответствует одна или две цифры (такая проверка используется для дня и месяца); \d{2,4} соответствует году; а [-\/] соответствует - или / в качестве разделителя даты. В последнем наборе наклонная черта / защищена и записана как \/. Во многих реализациях регулярных выражений защита не нужна, но некоторые синтаксические анализаторы регулярных выражений действительно требуют этого. Поэтому рекомендуется всегда защищать наклонную черту /.

Интервалы могут начинаться с 0. Интервал {0,3} соответствует нулю, одному, двум или трем вхождениям шаблона. Интервалы используются также для того, чтобы определить минимальное количество совпадений с шаблоном, не указывая при этом максимального. Например, {3,}.

Выбранный для следующего примера текст является частью Web-страницы и содержит текст со встроенными HTML-тегами <В>. Регулярное выражение должно найти любой текст внутри тегов <В> (это может потребоваться при замене форматирования; рис. 9).



Рис. 9. Поиск текста внутри тегов

Причина того, что шаблон нашел одно длинное вхождение вместо двух коротких, состоит в том, что метасимволы типа \* и + являются *жадными*; т.е. они ищут самое длинное возможное соответствие, а не наименьшее. Решение состоит в том, чтобы использовать *ленивые* версии этих кванторов (они называются ленивыми, потому что устанавливают соответствие с наименьшим (а не наибольшим) возможным количеством символов). Ленивые кванторы определяются путем добавления в конец ? к используемому квантору, причем для каждого из жадных кванторов имеется ленивый эквивалент (рис. 10).

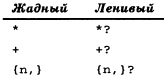


Рис. 10. Жадные и ленивые кванторы

Шаблон <[Вb]>.\*?</[Вb]> найдет в предыдущем примере два совпадения (рис. 11).

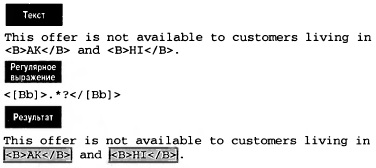


Рис. 11. Поиск текста внутри тегов с помощью ленивого квантора

### Урок 6. Соответствие позиций

Соответствие позиций позволяет указать, где в строке текста должно произойти совпадение. Например, если вы хотите заменить все вхождения слова cat на слово dog, нудно учитывать границы и использовать специальные метасимволы для определения позиции (или границы) перед шаблоном и после него. Символ \b указывает границу слова. Таким образом, используйте регулярное выражение \bcat\b. Соответственно, шаблон \bcat найдет все слова, начинающиеся с cat, а шаблон cat\b – заканчивающиеся на cat.

На самом деле символ \b не соответствует какому-либо символу; он соответствует позиции. Поэтому длина строки, которая находится шаблоном \bcat\b, равна трем (с, а и t), а не пяти символам.

Чтобы указать нечто, не соответствующее границе слова, используйте \В.

Границы строк используются для нахождения соответствий с шаблонами в начале или конце всей строки. Метасимволы для границ строк — крыша ^ (начало строки) и доллар $ (конец строки). Обратите внимание, что метасимвол ^ имеет несколько значений. Этот метасимвол отрицает набор, только если находится в наборе (т.е. заключен в квадратные скобки [ и ]) и является первым символом после открывающей [. Вне набора ^ соответствует началу строки.

В следующем примере необходимо проверить, что открывающий XML-тег является первым текстом в строке (рис. 12).

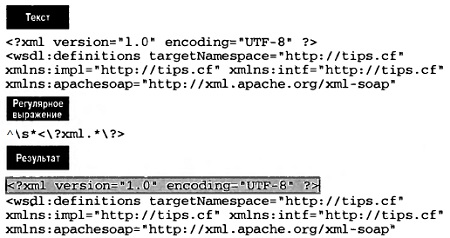


Рис. 12. Поиск строки, начинающейся с XML-тега

Чтобы не попасться в ловушку жадного квантора, лучше использовать ^\s\*<\?xml.\***?**\?>

Знак $ используется во многом аналогичным способом. Этот знак помогает проверить, что после закрывающего тега </html> на Web-странице ничего нет: </[Hh][Tt][Mm][LI]>\s\*$. Наборы используются для каждого из символов Н, Т, М и L (чтобы обработать любую комбинацию символов верхнего и нижнего регистров), a \s\*$ соответствуют любому пробельному символу, за которым следует конец строки.

Во многих реализациях регулярных выражений имеются специальные метасимволы, которые изменяют поведение других метасимволов; один из них — (?m). Этот метасимвол допускает использование многострочного режима. Многострочный режим вынуждает механизм регулярных выражений обрабатывать конец строки как разделитель строк, и в этом режиме ^ соответствует началу строки или началу после конца строки (т.е. началу новой строки), а $ соответствует концу строки или концу после конца строки. Выражение (?m) не поддерживается многими реализациями регулярных выражений.

### Урок 7. Использование подвыражений

Подвыражения — части большего выражения; части группируются так, чтобы они обрабатывались как единый объект. Подвыражения заключаются между символами ( и ). Например, шаблон &nbsp;{2,} найдет &nbsp;;;;; но не &nbsp;&nbsp;. Наоборот, (&nbsp;){2,} найдет &nbsp;&nbsp;.

В следующем примере мы попытаемся найти год в записи о пользователе (рис. 13).



Рис. 13. Поиск на основании группировки выражений

Если оператор | поместить в подвыражение, то он будет знать, что требуется найти один из вариантов в группе. Таким образом, шаблон (19|20)\d{2} найдет 1967 и вообще он соответствует любым четырем цифрам, начинающимся с 19 или 20.

Подвыражения могут быть вложены.

### Урок 8. Использование ссылок назад

Предположим, что в тексте нужно найти все повторяющиеся (подряд) слова (т.е. опечатки, где-то же самое слово было по ошибке напечатано дважды). Очевидно, при поиске второго вхождения слова должно быть известно предыдущее слово. Ссылки назад позволяют в шаблоне регулярного выражения обратиться к предыдущим совпадениям (в данном случае к ранее найденному слову).

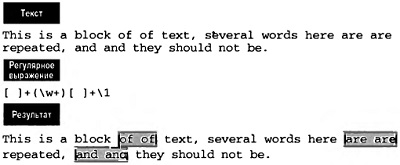


Рис. 14. Поиск двух одинаковых слов подряд с помощью ссылок назад

Выражение [ ]+ соответствует одному или нескольким пробелам, \w+ соответствует одному или нескольким алфавитно-цифровым символам, а [ ]+ соответствует любым пробелам в конце. Но заметьте, что шаблон \w+ заключен в круглые скобки, потому он является подвыражением. Это подвыражение используется не для того, чтобы найти повторные совпадения; никаких символов повторения здесь нет. Подвыражение используется для того, чтобы сгруппировать символы в одно выражение, пометить его и идентифицировать для использования в дальнейшем. Заключительная часть этого шаблона \1; это и есть ссылка назад, на подвыражение.

Термин «ссылка назад» обозначает объект, который ссылается назад на предыдущее выражение. Что же в точности означает \1? Это выражение соответствует первому подвыражению, используемому в шаблоне. Выражение \2 соответствует второму подвыражению, \3 – третьему, и т.д. Ссылки назад подобны переменным.

Рассмотрим пример с заголовками HTML-документа. Используя ссылки назад, можно создать шаблон, который найдет любой открывающий тег заголовка и соответствующий закрывающий тег (и игнорирует любой непарный тег; рис. 15).

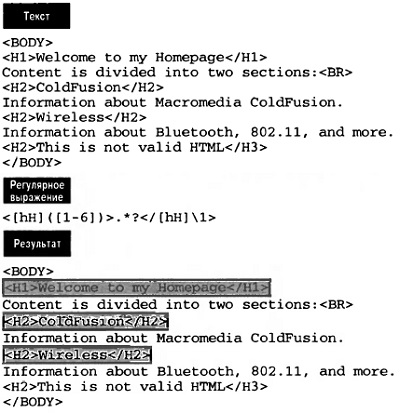


Рис. 15. Поиск парных открывающих и закрывающих тегов с помощью ссылок назад

Были найдены три совпадения: одна пара <Н1> и две пары <Н2>. Шаблон <[hH]([1-6] )> соответствует любому открывающему тегу заголовка. Диапазон [1-6] заключен в скобки ( и ), поэтому он рассматривается как подвыражение. Следовательно, шаблон для закрывающего тега заголовка может именовать это подвыражение как \1 в </[hH]\1>. Подвыражение ([1-6]) соответствует цифрам от 1 до б, и поэтому \1 соответствует только та же самая цифра, которую нашло это подвыражение. По этой причине <H2>This is not valid HTML</H3> не соответствует шаблону.

Ссылки назад будут работать только в том случае, если выражение, на которое они ссылаются, является подвыражением (и потому заключено в скобки). К сожалению, синтаксис ссылок назад очень зависит от реализации регулярных выражений в приложениях. В некоторых более новых реализациях регулярных выражений поддерживается захват по именам (named capture) —возможность, благодаря которой каждому подвыражению можно дать уникальное название (имя); впоследствии имя может использоваться для того, чтобы обратиться к подвыражению (по этому имени, а не по его относительной позиции).

*Выполнение операций замены.* Допустим, нужно сделать так, чтобы по любому адресу электронной почты, встречающемуся в тексте, можно было отправить письмо. В HTML-документе вы бы использовали тег <А HREF= "mailto: user@address.com">user©address.соm/А>, чтобы создать адрес электронной почты, активизирующийся по щелчку на нем. Может ли регулярное выражение конвертировать (автоматически преобразовать) адрес в тексте в формат адреса, который активизируется по щелчку на нем? (рис. 16).

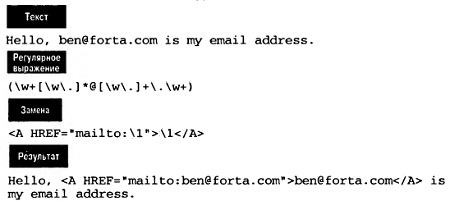


Рис. 16. Поиск и замена в электронном адресе

Ссылки назад могут содержать шаблоны, так что подвыражение, найденное в первом шаблоне, может использоваться во втором шаблоне. (\w+[\w\.]\*@[\w\ .]+\.\w+) —шаблон, который определен как подвыражение. Благодаря этому найденный текст может использоваться в шаблоне замены. В <А HREF="mailto:\1">\1</А> используется найденное подвыражение дважды: один раз в атрибуте HREF (чтобы определить mailto:) и другой раз в качестве текста, активизируемого по щелчку.

Рассмотрим еще один пример. Информация о пользователях хранится в базе данных, причем номера телефона хранятся в формате 313-555-1234. Необходимо переформатировать номера телефонов, чтобы представить их в виде (313) 555-1234 (рис. 17).

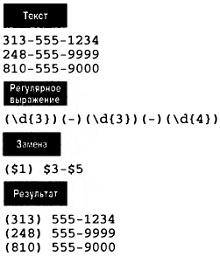


Рис. 17. Поиск и замена формата номера телефона

Выражение (\d{3})(-)(\d{3})(-)(\d{4}) соответствует номеру телефона, но разбивает его на пять подвыражений (чтобы найти его части). Эти пять частей могут использоваться независимо, причем так, как необходимо, так что ($1) $3-$5 просто переформатирует номер, используя только три подвыражения и игнорируя другие два. В результате 313-555-1234 превращается в (313) 555-1234.

*Замена регистра.* В некоторых реализациях регулярных выражений можно выполнять преобразования с помощью дополнительных метасимволов (рис. 18).

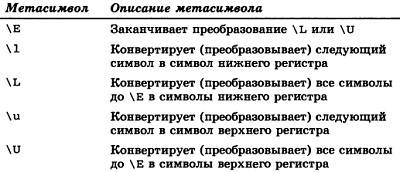


Рис. 18. Метасимволы замены регистра

Чтобы изменить регистр следующего символа, \l и \u помещаются перед символом (или выражением). Метасимволы \L и \U конвертируют (преобразовывают) регистр всех символов, пока не встретится метасимвол завершения этой операции \Е. Например, текст внутри пары заголовочных тегов уровня <Н1> преобразуем к верхнему регистру.

Текст: <H1>Welcome to my Homepage</H1>

Регулярное выражение: (<[Hh]1>)(.\*?)(</[Hh]1>)

Замена: $1\U$2\E$3

Результат: <H1>WELCOME TO MY HOMEPAGE</H1>

### Урок 9. Просмотр вперед и назад

Иногда нужно создавать шаблон так, чтобы он содержал совпадения, которые не возвращаются, иными словами, совпадения, которые используются только для того, чтобы правильно найти местоположение основного совпадения, но сами вспомогательные совпадения не рассматриваются как части найденного совпадения. Синтаксически шаблон для просмотра вперед выглядит как подвыражение, которому предшествует ?=, а искомый текст следует после знака =.

Иногда в документации по регулярным выражениям используется термин потреблять (consume) для обозначения того, что будет найдено и возвращено; о том же, что было найдено в результате просмотра вперед, говорят, что оно не потребляется, не используется (not consume).

Следующий текст содержит список URL, и в каждом URL нужно извлечь часть, соответствующую протоколу (возможно, чтобы знать, как обработать эти URL; рис. 19).



Рис. 19. Просмотр вперед

В перечисленных URL протокол отделен от имени хоста двоеточием :. Шаблон .+ соответствует любому тексту (http в первом соответствии), а подвыражение (?=:) соответствует :. Но заметьте, что это двоеточие : не было найдено; ?= указывает, что механизм регулярных выражений должен установить соответствие с двоеточием : и в то же время не потреблять (не использовать) двоеточие.

Выражение .+(:) находит текст вместе с двоеточием :. Выражение .+(?=:) находит текст до двоеточия :.

Просмотр вперед ищет фрагмент, находящийся после возвращаемого текста. Просмотр назад ищет фрагмент, предшествующий возвращаемому тексту. Для обозначения просмотра назад используется оператор ?<=.

Операции просмотра вперед и просмотра назад могут использоваться совместно (рис. 20).

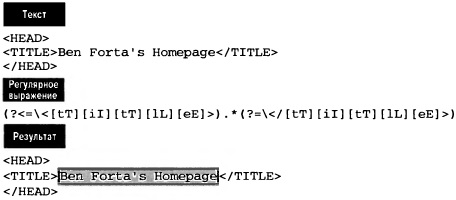


Рис. 20. Совместное использование просмотра вперед и назад

Шаблон (?<=\<[tT][iI][tТ][lL][eE]>) определяет операцию просмотра назад, которая устанавливает соответствие с открывающим тегом заголовка <TITLE> (но не потребляет его); подобным образом (?=\</[tT][iI][tТ][lL][eE]>) устанавливает соответствие с закрывающим тегом заголовка </TITLE>. Возвращается только текст заголовка (поскольку это все, что было использовано). Обратите внимание, что знак < (первый символ, с которым устанавливается соответствие) защищен для того, чтобы предотвратить двусмысленность. Поэтому написано (?<=\<, а не (?<=<.

### Урок 10. Встроенные условия

Обработка условных выражений поддерживается не всеми реализациями регулярных выражений. Условия в регулярных выражениях определяются с помощью знака ?. Фактически, вы уже встречали несколько очень специфических условий:

* ? соответствует предыдущему символу (или выражению), если он существует.
* ?= и ?<= соответствуют тексту впереди или позади, если он существует.

Условие в ссылке назад позволяет использовать выражение только в том случае, если предыдущий поиск подвыражения завершился успешно. Синтаксис для этого типа условия – *(?{ссылка назад) истина)*.

Условия могут также иметь *else*-выражения, т.е. выражения, которые выполняются только в том случае, если ссылка назад не существует (условие не выполняется). Синтаксис для этой формы условия — *(?{ссылка назад) истина] ложь)*. В этом синтаксисе предусмотрено условие, а также выражения истина и ложь, одно их них будет выполнено, причем какое именно — зависит от того, выполнено ли условие.

В Северной Америке приемлемыми считаются два формата для представления номеров телефонов – (123) 456-7890 и 123-456-7890. А вот 1234567890, (123)-456-7890 и (123-456-7890 хотя и содержат правильное количество цифр, но отформатированы крайне плохо (рис. 21).

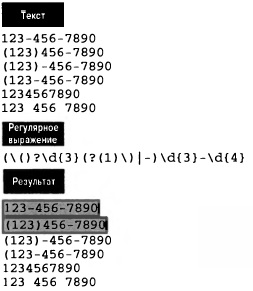


Рис. 21. Встроенные условия

Выражение (\()? проверяет наличие открывающей скобок, но сама проверка заключена в круглые скобки, чтобы создать подвыражение. Выражение \d{3} соответствует 3-цифровому коду города. Выражение (?(1)\)|-) соответствует либо ) либо - в зависимости от того, удовлетворено ли условие. Если (1) существует (это значит, что была найдена открывающая скобка), то нужно найти и \); в противном случае нужно найти -. Благодаря этому круглые скобки всегда должны быть парными, а дефис, отделяющий код города от номера, присутствует только в том случае, если круглые скобки не используются.

См. также [Знакомство с регулярными выражениями](https://msdn.microsoft.com/ru-ru/library/28hw3sce(v=vs.100).aspx)