RegExp

RegExp - программа предназначенная для теста регулярных выражений, но не исключает и обработку данных. Имеет несколько библиотек с готовыми регулярными выражениями, для решения некоторых задач парсинга. Программа представляет собой только оболочку, синтаксический анализ (парсинг) выполняет движок PCRE. Программа написана по мотивам ранее написанной мной программы RegExp на AutoIt3

Версия 0.5.2 от 2022г
Автор AZJIO
Сайт http://azjio.ucoz.ru
Обсуждение:

- azjio.ucoz.ru
- purebasic.fr
- purebasic.mybb.ru
- usbtor.ru
- puppyrus.org

Описание

Кнопка Старт

Выполняет поиск/замену и т.д. с помощью регулярного выражения

Библиотека

При запуске программа проверяет файлы в папке "Library" и добавляет их в список библиотек. Библиотека содержит регулярные выражения, флаги и текст обработки. Содержимое библиотеки отображается в правой части окна. Клик на элементе библиотеки вставляет данные в поля. Две кнопки (Добавить и Удалить) предназначены для добавления (перезаписи) и удаления пунктов. По умолчанию используется имя выбранного пункта, что позволяет перезаписать, т.е. обновить данные пункта. Если надо добавить новый, то указать новое имя пункта. Библиотека позволяет не потерять важные регулярные выражения и всегда иметь к ним доступ.

Вывод результата

Результат выводится в нижнее слева окно. Результат зависит от режима поиска. Это может быть текстом найдено-ненайдено или результатом замены или списком элементов, а также предупреждением об ошибке в регулярном выражении на англ. языке.

Режимы

- Поиск - возвращает "Надено" или "Не найдено".
- Замена - возвращает текст, в котором выполнена замена.
  Рядом имеются две галки: поддержка Unescape-символов (\r\n\t) и поддержка групп (\1...\9).
- Массив - возвращает целиком найденные элементы соответствующие регулярному выражению.
- Группы - возвращает элементы регулярного выражения находящиеся в скобках.
- Пошаговый - возвращает позицию, длину и найденный текст регулярного выражения.

Флаг "С разметкой" позволяет сделать вывод результатов более наглядным, с номерами найденных элементов.

Скорость выполнения

Под кнопкой "Старт" есть два поля. Нижнее показывает время выполнения текущего регулярного выражения на текущем тексте. Верхнее - время предыдущего теста. Это необходимо для сравнения скорости. Если текущий тест выполнен быстрее предыдущего, то нижнее поле подсвечивается зелёным цветом, иначе красным.

Стиль оформления

В ini можно задать цвет элементов окна и цвет подсветки метасимволов в поле регулярного выражения. Стиль выбирается так: style=style1, при этом в ini-файле должна быть секция [style1] с настройками цвета. Для примера ini-файл содержит несколько готовых

стилей.

Вставка метасимволов

Некоторые комбинации можно вставить с помощью кнопки "звёздочка" - "*", меню которого можно сделать самостоятельно в Menu.ini.

История

Кнопка истории сохраняет 30 элементов последнего использования регулярных выражений. Регулируется параметром maxhistor=30.

Командная строка

В качестве инструмента в PureBasic можно добавить с ком-строкой:

```
-l:PureBasic -nu -i:4 "%TEMPFILE"
```

Здесь ключи имеют следующее значение:
-l:PureBasic - выбор библиотеки
-nu - не обновлять (или заморозить обрабатываемый текст, чтобы не изменялось при нажатии других пунктов)
-i:4 - выбор 4-го пункта, отсчёт от 0. Параметр обязательно должен быть после ключа -l: иначе отработает когда библиотека не открыта
"%TEMPFILE" путь в кавычках открываемого файла в окно обрабатываемого текста.
Данный функционал позволяет автоматически открывать RegExpPB с нужными настройками и искать или обрабатывать исходник с

помощью регулярных выражений.

Флаги

OnTop - короткое название "Поверх всех окон"
Не обновлять - при вставки из библиотеки не заменяется текст для обработки. Допустим нужно обрабатывать свой текст разными регулярными выражениями.
Также 5 включаемых флагов, которые можно включать в самом регулярном выражении методом (?s) и т.д.

Кнопки

- Открыть - Открыть текстовый файл для обработки.
- Очистить - Очищает 4 поля ввода.
- Добавить - Добавить семпл в библиотеку.
- Удалить - Удалить выбранный пункт библиотеки.
- Метасимволы - выбор готовых комбинаций для вставки в шаблон регулярного выражения.
- Переместить вверх - Перемещает текст из результатов в окно для обработки. Полезно если некий текст надо прогнать через десяток регулярных выражений. После каждого примененния нажимать кнопку чтобы перемещать обработанный текст в результатах наверх для обработки следующим регулярным выражением.

- Copy - копирует в буфер обмена готовую конструкцию для вставки в код. Для этого в папке "template" есть одноимённые готовые шаблоны, в которых переменные подменяются соотвествующими полями. Чтобы программа была универсальна есть параметр copysel=0, если поменять его на copysel=1, то вместо использования одноименных шаблонов откроется папка с возможностью выбора файла-шаблона, таким образом можно

даже для одного режима сделать несколько шаблонов, не говоря уже о разных языках программирования.

Описание ini-файла

```
[Set] - секция основных параметров
width = 800 - ширина окна
height = 600 - высота окна
fontsize = 11 - размер шрифта
maxhistor = 30 - максимальное число пунктов истории
topmost = 0 - поверх всех окно
lastlib = PureBasic - последняя библиотека, выбираемая при запуске программы
copysel = 0 - Флаг переключения выбора шаблона методом открытия файла
style = style1 - выбор секции стиля

[regexp] - история регулярных выражений
1 = [А-Яа-яЁё]+ - один из элементов истории
2 = (\r\n|\r|\n){2,}

[style1] - секция стиля
gui = f - цвет окна (только Windows)
gadget = f - цвет гаджетов (поле ввода) (только Windows)
gadgetfont = f - цвет шрифта (тексты окна, чекбоксы, надписи) (только Windows)
type = 1 - тип цвета, фон или шрифт (для регулярных выражений)
background = f - цвет фона
default = 0 - цвет шрифта
select_bg = f99 - цвет фона выделенного текста
select_fnt = 0 - цвет шрифта выделенного текста
caret = 0 - цвет курсора
re_Repeat = BFFFBD - цвет повтора {n,m}
```

```
re_SqBrackets = BDF7FF - цвет квадратных скобок []
re_RndBrackets = FFBDBD - цвет круглых скобок ()
re_AnyText = E6DDFF - цвет любого текста .*?
re_Meta = FFFFA5 - цвет метасимволов \w и т.д.
re_Borders = FFDFA5 - цвет границ \A \b и т.д.
re_ChrH = FFC1F7 - цвет кода символа \x01 \x{01}
```

Цвет может быть задан упрощённо, например "F" означает "FFFFFF", "3F" = "3F3F3F", "F95" = ""FF9955"

Регулярные выражения

Метасимволы вне квадратных скобок

( ) - начало и конец группы, например (text). Означают последовательность. Используются для применения квантификаторов не к одному символу, а к нескольким, а также для дальнейшего использования найденной последовательности.
[ ] - начало и конец описания символьного класса, например [a-z]. Символьный класс возвращает один символ из множества. Изменить это могут повторители.
{ } - начало и конец повторителей, например {3,8}
**** - экранирующий символ, принять метасимвол как обычный символ, например (\\, \., \[, \], \{, \}, *).
^ - начало строки (или начало текста в многострочных текстах с флагом (?m)), например ^text text$
**$** - конец строки (или конец текста в многострочных текстах с флагом (?m)), например ^text text$
. - любой символ, кроме переноса строки @LF (по умолчанию). С флагом (?s) - любой символ
| - символ "или", обычно внутри группы, например (10|20)
? - предыдущий символ либо имеется, либо не имеется, аналогично и для групп. После символа повтора - жадность патерна - (.*?)
***** - повтор предыдущего символа или группы 0 и более раз
+ - повтор предыдущего символа или группы 1 и более раз

Метасимволы внутри квадратных скобок

Часть шаблона, заключенная в квадратные скобки, называется символьным классом. Внутри скобок метасимволы теряют свое специальное значение, кроме метасимволов принадлежащих этому классу. Экранировать требуется только 4 символа \ -] [. Если символ "-

" находится в конце перечисления, то не требует экранирования. В шаблоне могут использоваться метасимволы диапазонов, но не используются метасимволы границ, например \A, \B, \Z, \z, а метасимвол \b означает символ возврата 'backspace'. Учтите, что диапазоны, например [а-я] используют UTF-8 последовательность, а не ASCII.

\ - экранирующий символ
^ - символ исключения, но в случае, когда стоит первым, например [^3] все кроме три
- - символ охвата, например [a-z], то есть все символы от a до z
[ ] - начало и конец описания символьного класса, например [a-z]

Метасимволы подстановки

\1 - \9 - ссылка на найденную группу в самом шаблоне и в шаблоне замены. Отсчёт групп слева по открывающей скобке "("
$1 - $9 - ссылка на найденную группу в шаблоне замены
$0 или \0 - весь шаблон поиска или все группы (9 не ограничение)
\a - Chr(7) - символ с десятичным ASCII-кодом 7 (звонок). При выводе воспроизводит звуковой сигнал. BEL (hex 07)
\cn - управляющий символ, который генерируется при нажатии комбинации клавиш Ctrl+n, где n- символ, например \cD соответствует Ctrl+D. \cA = \001, \cZ = \032, \cM = \r = \015
\e - Chr(27) - символ escape (hex 1B)
\f - Chr(12) перенос страницы (hex 0C)
\h - [\t] - любой горизонтальный пробел, табуляция - Chr(9), Chr(32), Chr(160)
\H - [^\h] - любой символ, который не пробел или табуляция
\K - слева от \K предшествующее совпадение, т.е. **текст1\Kтекст2**, найти **текст2**, перед которым **текст1**.
\n - @LF, Chr(10) - символ переноса на новую строку (hex 0A)
\N - [^\n] Любой символ, который не символ переноса на новую строку (не @LF). Не работает в 3.3.6.1
\Q ... \E - любые метасимволы между \Q и \E воспринимаются как

текст. Не исключайте ошибки: \QD:\Edit\1.txt\E
\r - @CR, Chr(13) - символ возврат каретки (hex 0D)
\R - [\n\f\r\v] Chr(10), Chr(11), Chr(12), Chr(13) любой из символов переноса строки
\t - @TAB, Chr(9) символ табуляции - tab (hex 09)
\v - [\r\n\f] Chr(10), Chr(11), Chr(12), Chr(13) вертикальная табуляция (@CR и @LF и перенос страницы)
\V - [^\v] - любой символ, который не Chr(10), Chr(11), Chr(12), Chr(13) вертикальная табуляция (перенос строки)
\x** - где * - любая шестнадцатеричная цифра, например \x41 соответствует латинской букве 'A', \x50\x65\x72\x6C - слово Perl
\x{..}** - где * - любая шестнадцатеричным цифра, например \x{50}\x{65}\x{72}\x{6C} - слово Perl. Попробуйте от \x{01} до \x{7F}, что в десятеричной системе означает символы от 1 до 127. Или в UTF кодировке \x{044F} равный символу "я"
******* - где * - любая восьмеричная цифра. Например, последовательность \120\145\162\154 представляет слово Perl (\120 - восьмеричный код буквы P, \145 - буквы e, \162 - буквы r, \154 - буквы l). Пробел - \040. Попробуйте от \001 до \177, что в десятеричной системе означает символы от 1 до 127

Метасимволы для задания групп символов

\d - [0-9] - любая десятичная цифра
\D - [^0-9] любая не цифра
\s - [\f\n\r\t\v] - пустой символ: Chr(9), Chr(10), Chr(12), Chr(13), Chr(32) (перенос страницы, табуляция, возврат каретки, перевод строки и пробел).
\S - [^\f\n\r\t\v] - любой непробельный символ
\w - [0-9a-zA-Z_] - любой алфавитно-числовой символ или подчеркивание (только символы латинского алфавита)
\W - [^0-9a-zA-Z_] - любой символ неслова

Границы символов

\A - начало текста, не зависит от флага "(?m)" и поэтому может встретится только 1 раз.
\G - похожий на **\A**, но несколько раз, если найденные следуют друг за другом от начала.
\z - абсолютный конец текста, не зависит от флага "(?m)" и поэтому может встретится только 1 раз
\Z - конец текста, т. е. граница между любым символом и концом текста или до символа \n, если он в конце строки, не зависит от флага "(?m)" и поэтому может встретится только 1 раз.
\b - начало или конец слова, т. е. граница между символами, один из которых удовлетворяет \W, а другой - удовлетворяет \w (только в англ. текстах)
\B - середина слова, т. е. граница между символами, оба которых удовлетворяют \W или оба которых удовлетворяют \w

Флаги модификаторы

Ставятся в начало регулярного выражения или группы.
Состояние модификаторов выключено по умолчанию, соответственно для использования требуется включить.
Пример использования : (?i)(Text) или ((?-i)Text), допустимо комбинировать (?is)(Text) или ((?imsx)Text)

(?i) - не учитывать регистр символов. Это работает только для символов латинского алфавита.
(?-i) - отменяет ранее включенный (?i)
(?m) - в многострочном тексте символы ^ и $ означают начало и конец строки соответственно, например ^(Text)\r$, иначе начало и конец текста. Символ LF - разделитель строк
(?-m) - отменяет ранее включенный (?m)
(?s) - символ "точка" (.) дополнительно включает в себя перенос строки LF (режим "одна строка")

(?-s) - отменяет ранее включенный (?s)
(?x) - игнорирует пробелы и табуляции в регулярном выражении, кроме тех что в квадратных скобках. Пробелы позволяют сделать регулярное выражение легко читаемым. Позволяет в конце рег. выр. добавить комментарий после символа #
(?-x) - отменяет ранее включенный (?x)
(?J) - allow duplicate names (разрешает дубликаты/двойные названия).
(?U) - инвертировать жадность квантификаторов
(?-U) - отменяет ранее включенный (?U)

Флаги групп

Допустимо комбинировать **(?im-sx:Text)**, флаги sx выключены

(?i:...) - группа, не учитывает регистр символов, например **(?i:Text)**. Это работает только для символов латинского алфавита.
(?-i:...) - группа учитывает регистр символов, например **(?-i:Text)**
(?:...) - исключает группу из найденных, например **(?:Text)**
(?>...) - группа не входящая в поиск, но имеет свойство сверхжадного квантификатора, например **(?>Text)(Text)**

Эти 4 группы имеют фиксированную длину, в них нельзя использовать *****, +, **{n, m}**
(?=...) - группа не входящая в поиск, но проверяющая **совпадение** образца **справа**, например **(Text)(?=Text)**
(?!...) - группа не входящая в поиск, но проверяющая **не совпадение** образца **справа**, например **(Text)(?!Text)**
(?<=...) - группа не входящая в поиск, но проверяющая **совпадение** образца **слева**, например **(?<=Text)(Text)**, см. также **\K**
(?<!...) - группа не входящая в поиск, но проверяющая **не совпадение** образца **слева**, например **(?<!Text)(Text)**

(?<name>...) - именованная ссылка. Вызов именованной ссылки \k<name> это тоже что вызов \1 или $1

(?#...) - группа содержащая комментарий, например (?# это комментарий). Полностью игнорируется интерпретатором

Повтор предыдущего элемента, применяется к символам и группам (квантификаторы)

{n} - повторить предыдущий символ n раз
{n,} - повторить предыдущий символ n и более раз (**{n,}?** - предпочтительно наименьший захват)
{n, m} - повторить предыдущий символ от n до m раз (**{n,m}?** - предпочтительно наименьший захват)
***** - повторить предыдущий символ 0 и более раз. То же что и **{0,}**. Наибольший захват, который позволит совпасть оставшейся части шаблона.
+ - повторить предыдущий символ 1 и более раз. То же что и **{1,}**. Наибольший захват, который позволит совпасть оставшейся части шаблона.
? - предыдущий символ либо имеется, либо не имеется. То же что и **{0,1}**. Второе значение символа **?** после символа повтора **.*?** - жадность, см. ниже
***?** - повторить предыдущий символ 0 и более раз. Ограничится наименьшим захватом, который позволит совпасть оставшейся части шаблона.
+? - повторить предыдущий символ 1 и более раз. Ограничится наименьшим захватом, который позволит совпасть оставшейся части шаблона.
?? - предпочтительно наименьший захват, например **([a-z]??)g** для 'gg' возвращает две пустые строки

Ревнивая или сверхжадная квантификация

Захват без возврата к предыдущим шагам поиска. Захватывает всё, что удовлетворяет предыдущему символу, не заботясь о совпадении оставшейся части шаблона. Символ или диапазон символов следующий за сверхжадным метасимволом не должен поглащатся его диапазоном, иначе такой шаблон никогда не будет найден и является бессмысленным. Единственная цель сверхжадного метасимвола - ускорить захват.

***+** - повторить предыдущий символ 0 и более раз.
++ - повторить предыдущий символ 1 и более раз.
{n,}+ - повторить предыдущий символ n и более раз.

Символьные классы POSIX

Пример [[:upper:]]{2} - поиск повтора заглавных букв. Инвертировать диапазон так: [[:^digit:]]

[:alnum:] - буквы и цифры [0-9A-Za-z] (как \w, но без "_")
[:alpha:] - буквы [A-Za-z] (без "_")
[:ascii:] - символы от Chr(0) до Chr(127)
[:blank:] - пробел и символ табуляции Chr(9) и Chr(32), тоже что [\t]
[:cntrl:] - управляющие символы от Chr(0) до Chr(31) и Chr(127)
[:digit:] - десятичные цифры, тоже что \d, [0-9]
[:graph:] - тоже что символы, отображаемые при печати [:print:], но кроме пробела (от Chr(33) до Chr(126))
[:lower:] - прописные буквы [a-z]
[:print:] - символы, отображаемые при печати, включая пробел (от Chr(32) до Chr(126))
[:punct:] - символы, отображаемые при печати, кроме букв и цифр Chr=(33-47, 58-64, 91-96, 123-126), те, что не входят ни в [:alnum:], ни в [:cntrl:]
[:space:] - пробельные символы (как \s, но включая символ VT: Chr(11)) от Chr(9) до Chr(13) и Chr(32). Тоже что [\f\n\r\t\v]
[:upper:] - заглавные буквы [A-Z]

[:word:] - символы слов, тоже что \w
[:xdigit:] - шестнадцатеричные цифры [0-9A-Fa-f]

Условные подмаски

(?(если)то) - например (?(?=[a-z])\d), (?(условие)шаблон_при_успехе)
(?(если)то|иначе) - например (?(?<=\d)a|b) или (?:(?>(?=[^a-z]*[a-z])())?(?:(?=\1)aa|(?!\1)1)), (?(условие)шаблон_при_успехе|шаблон_при_не_успехе)
(?=[\w]+)| (?R) - рекурсивный вызов

Эти флаги не действуют в AutoIt3
\p любой символ пунктуации
\l - означает, что следующий символ регулярного выражения преобразуется в нижний регистр.
\u - означает, что следующий символ регулярного выражения преобразуется в верхний регистр.
\L...\E - означает, что все символы в регулярном выражении между \L и \E преобразуются в нижний регистр.
\U...\E - означает, что все символы в регулярном выражении между \U и \E преобразуются в верхний регистр.
\x - любой шестнадцатеричный символ
\< - начало слова, т. е. граница между символом, удовлетворяющим \W и символом, удовлетворяющим \w
\> - конец слова, т. е. граница между символом, удовлетворяющим \w и символом, удовлетворяющим \W
{,n} - повторить предыдущий символ от 0 до n раз

Примеры конструкций

.* - повтор любого символа, а значит весь текст
[...] - одиночный символ множества, например [aeiou] - любой из строчных гласных
[^ ...] - ни один из символов множества, например [^aeiou] - ни один

из строчных гласных
[0-9A-Fa-f]{6} - Шестнадцатеричное число, например FF0000.
[А-яЁё] - Диапазон для русских букв. Или так [А-Яа-яЁё]
(\r\n|\r|\n){2,}, заменить на **\1** - удаление пустых строк
(?<![А-яЁё])([А-яЁё]+) \1, заменить на **\1** - удаление повторов слов
[A-ZA-ЯЁ]{2,}?[a-za-яё]+ - выявит файлы, в которых есть ошибки вида "НАйти"- не преднамеренный повтор заглавной буквы
(.{35,}?[ ])(.*?), заменить на **'$0' & @CRLF** - где @CRLF символ переноса строки. Поставить флаг "Вычислить". Выполнить перенос строки на границе первого попавшегося пробела после каждых 35 символов.
(?si)(?:.*?)?(https?:\/\/[\w.:]+\/?(?:[\w\/?&=.~;\-+!*_#%])*) - найти ссылки
[A-Za-z0-9._-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}) - найти почтовые ящики

Ссылки

Учебники
Регулярные выражения. Дж. Фридл. 3-е издание, а таксже html
edlinsoft.blogspot.com - для .NET Framework, обьяснение на примерах.
msdn.ru
php.ru, php.net
Управляющие символы (wiki)
ajaxforum.ru
Регулярные выражения на wiki
regexpstudio.com - Регулярные выражения для Delphi
docs.notepad-plus-plus.org - Официальный источник Notepad++ (англ. яз.)
pcre.org - Официальный справочник движка PCRE (англ. яз.)

Онлайн тестеры
regexr.com - здесь в Community множество готовых регулярных выражений

для javascript, для php (pagecolumn.com)
cuneytyilmaz.com - для javascript
php-include.ru -на флеш-плеере
regex101.com
easyregexp.ru
debuggex.com - показывает структурно

Программы для теста
RegexBuddy - крутая и платная
RegExp - бесплатно, AZJIO, PCRE, AutoIt3
Expresso
The Regex Coach

Лицензия

Лицензия на программу

Программа бесплатна

Для использования "как есть", ответственность от неправильного использования ложится на пользователей.

Программа может входить в коммерческий проект, но без увеличения стоимости за счёт программы, так как она бесплатна. Это допускает использование в любых сборках как дополнительный инструмент.

Лицензия на исходник

Разрешается модернизировать исходник только для собственного пользования.

Разрешается изучать и использовать части кода.

Запрещается распространять модернизированную версию.

Запрещается менять диалог "О программе".

По любым не прописанным здесь вопросам обращаться к автору для возможности обсуждения условий.

Обновления

31.10.2022

Добавлена галка "Поверх всех окон"
Добавлена кнопка "Переместить вверх" для обработки текста многократно разными регулярными выражениями.

11.07.2022

Исправлена замена со ссылками на группы без наличия групп
Добавлена командная строка вида -l:PureBasic -nu -i:4 "%TEMPFILE"

Кнопка "Добавить" использует имя выделенного пункта, и если не будет изменено, то предлагает перезаписать текущий.
Исправлен регвыр чтения библиотеки, не мог прочитать с пустыми данными.
Добавлен "Не обновлять"
Добавлен параметр ini-файла copysel = 0. Включает выбор шаблона в Проводнике.
Добавлена библиотека регулярных выражений PureBasic

02.05.2022

Теперь битовые флаги при чтении/записи в библиотеку (библиотеки конвертированы с битовым флагом)
Проверка если в поле замены ссылки на группы, то ставится галочка поддержки групп.
Добавлена история регулярных выражений с сохранением.
Добавлен хоткей Ctrl+Enter
Добавлен цвет таймера.
Добавлены параметры ini-файла: "Поверх всех окон", число пунктов истории.
Добавлено сохранение выбора библиотеки
Добавлено 2 кнопки "Добавить" и "Удалить", чтобы наполнять библиотеки регулярных выражений.
Добавлена кнопка "открыть файл", а также "перетащить и бросить" в окно "текст для обработки" (UTF-8).
Настройки цвета выведены в ini-файл.

26.04.2022

Добавлены библиотеки регулярных выражений

План

Так как в Linux было двойное чтение данных при клике на пункте, то сделан игнор повторного чтения, но проблема что для обновления надо кликать иной пункт.
При отсутствии библиотеки при нажатии кнопки "Добавить" создать библиотеку.
Для Linux исправить функционал вычисления времени выполнения регулярного выражения.
Добавить флаг скрывать 5 флагов в GUI для тех кто не пользуется PureBasic и вводит флаги в самом регулярном выражении. При этом

нужен их сброс.