<%SUMMARY%>

This Word layout template for Help & Manual defines the entire layout - front page, table of contents, chapter headings, keyword index.

To **customize this template**, use the original file ("Word-Template-A4.docx") from the H&M installation folder, make a copy of it, then tailor it with MS Word to your requirements. You **can use variables** in every place of the document. To insert a variable, insert a COMMENT field in Word, then type the variable name as the field content. Like the  field above this text.

Move on to the next page in this document to customize the table of contents (a placeholder for the real TOC later) and modify text and paragraph format of the chapter headings in the middle of this template. You might also want to change the numbering style.

**Important:** before you use the template for publishing in H&M, make sure that general page format and margins are set correctly. H&M will export images and tables to make them fit in the space available.

and EC Software GmbH

**[Type the company address]**

**[Type the phone number]**

**[Type the fax number]**

**2017**

**TRegExpr - Delphi Expresiones Regulares**

Contents

<%CHAPTER\_HEADING1%> 3

<%CHAPTER\_HEADING2%> 3

<%CHAPTER\_HEADING3%> 3

<%CHAPTER\_HEADING4%> 3

<%CHAPTER\_HEADING5%> 3

<%CHAPTER\_HEADING6%> 3

# Acerca de

TRegExpr es una herramienta poderosa y fбcil de usar para controlar entrada de datos de cadenas de caracteres en base a plantillas (en DBMS y aplicaciones para web), bъsqueda y sustituciуn de texto, utilidades tipo egrep, etc.

Se puede verificar fбcilmente una direcciуn de e-mail, extraer un nъmero de telйfono o cуdigo postal de texto sin formato, cualquier tipo de informaciуn de una pбgina web, y todo lo que pueda imaginar!. Las reglas (plantillas) pueden ser modificadas sin recompilar el programa!

Esta librerнa freeware es una versiуn extendida para Delphi de las rutinas V8 de Henry Spencer,. Trabajan con un subconjunto de las [Expresiones Regulares](#RegExp_Syntax) de Perl.

TRegExpr estб programado en Pascal puro, con el cуdigo fuente completo gratis.

El original en C ha sido mejorado y encapsulado completamente en la clase [TRegExpr](#TRegExpr_Interface) implementada en un sуlo archivo: RegExpr.pas.

Por lo tanto, no se necesita ninguna DLL!

Ver los [ejemplos simples](#Demos) y estudiar la [sintaxis](#RegExp_Syntax) de las expresiones regulares (Se puede usar el [proyecto demo](#tregexpr_testrexp) para probar y depurar las expresiones regulares propias).

Se puede usar Unicode (WideString de Delphi) - ver "[Cómo usar Unicode](#TRegExpr_Interface_Unicode)".

Ver la web-secciуn [Novedades](http://RegExpStudio.com/TRegExpr/Help/Whats_New.html) para los ъltimos cambios.

Cualquier informaciуn de errores, comentarios e ideas son [apreciadas](#Author).

# Cuestiones legales

Copyright (c) 1999-2004 por Andrey V. Sorokin <[anso@mail.ru](`mailto:anso@mail.ru',`',1,`')>

Este software es provisto como estб, sin otorgar ningъn tipo de garantнa. Usarlo a riesgo personal.

Se puede usar este software en cualquier tipo de desarrollo, incluso comercial, y redistribuirlo y modificarlo libremente, bajo los siguientes tйrminos:

1. El origen de este software no debe ser cambiado. Usted no puede asumir la propiedad intelectual del original. Si se usa en cualquier tipo de producto serб apreciado un reconocimiento para el Autor en alguna pantalla informativa o en la documentaciуn de dicho producto.

2. Usted no puede obtener ningъn ingreso de la distribuciуn del cуdigo fuente a otros programadores. Si se usa en un producto comercial, el cуdigo fuente no puede ser vendido en forma separada.

---------------------------------------------------------------

Cuestiones legales del original en C

---------------------------------------------------------------

\* Copyright (c) 1986 por University of Toronto.

\* Escrito por Henry Spencer. No derivado de software con licencia.

\*

\* Se da permiso para usar este software para todo propуsito en cualquier

\* sistema informбtico y para distribuirlo libremente sujeto a los siguientes

\* tйrminos:

\* 1. El autor no es responsable por las consecuencias del uso de

\* este software, sin importar su gravedad, incluso si son producidas

\* por defectos de programaciуn.

\* 2. El origen de este software no debe ser cambiado, ni en forma

\* ni por omisiуn.

\* 3. Versiones modificadas deben ser marcadas como tales, y no

\* ser presentadas como software original.

# Cómo funciona

**Ejemplos simples**

Si no tiene experiencia con las expresiones regulares, por favor vea la secciуn [sintaxis](#RegExp_Syntax).

**Usando las rutinas globales**

Es simple pero poco flexible y efectivo

ExecRegExpr ('\d{3}-(\d{2}-\d{2}|\d{4})', 'Telйfono: 555-1234');

devuelve True

ExecRegExpr ('^\d{3}-(\d{2}-\d{2}|\d{4})', 'Telйfono: 555-1234');

devuelve False, porque hay algunos sнmbolos antes del nъmero de telйfono y estamos usando el metasнmbolo '^' (BeginningOfLine = Inicio de lнnea)

ReplaceRegExpr ('producto', 'Pruebe producto. producto es el mejor !', 'TRegExpr');

devuelve 'Pruebe TRegExpr. TRegExpr es el mejor !'; ;)

**Usando la clase TRegExpr**

Se obtiene todo el poder de la librerнa

*// Esta simple funciуn extrae todas las direcciones de email de la cadena ingresada*

*// y devuelve una lista de estos email en el resultado*

**function** ExtraeEmails (**const** AInputString : **string**) : **string**;

**const**

EmailRE = '[\_a-zA-Z\d\-\.]+@[\_a-zA-Z\d\-]+(\.[\_a-zA-Z\d\-]+)+'

**var**

r : TRegExpr;

**begin**

Result := '';

r := TRegExpr.Create; *// Crea el objeto*

**try** *// asegura la liberaciуn de memoria*

r.Expression := EmailRE;

*// La e.r. se compila automбticamente en estructuras internas*

*// cuando se asigna la propiedad Expression*

**if** r.Exec (AInputString) **then**

**REPEAT**

Result := Result + r.Match [0] + ', ';

**UNTIL** **not** r.ExecNext;

**finally** r.Free;

**end**;

**end**;

**begin**

ExtraeEmails ('Mis e-mails son anso@mail.ru y anso@usa.net');

*// devuelve 'anso@mail.ru, anso@usa.net, '*

**end**.

*// Nota: la compilaciуn de la r.e. realizada al asignar ;a propiedad Expression*

*// toma cierto tiempo, si se usa esta funciуn muchas veces*

*// se sobrecarga inъtilmente.*

*// Esto se puede optimizar significativamente creando el objeto TRegExpr*

*// y precompilando la expresiуn durante la inicializaciуn del programa.*

*// Este ejemplo extrae nъmeros de telйfono*

*// y los descompone en partes (cуdigos de Ciudad y paнs, nъmero telefуnico* *).*

*// Despuйs substituye estas partes en la mбscara ingresada.*

**function** ParseTel (**const** AInputString, ATemplate : **string**) : **string**;

**const**

IntPhoneRE = '(\+\d \*)?(\(\d+\) \*)?\d+(-\d\*)\*';

**var**

r : TRegExpr;

**begin**

r := TRegExpr.Create; *// Crea el objeto*

**try** *// asegura la liberaciуn de memoria*

r.Expression := IntPhoneRE;

*// La e.r. se compila automбticamente en estructuras internas*

*// cuando se asigna la propiedad Expression*

**if** r.Exec (AInputString)

**then** Result := r.Substitute (ATemplate)

**else** Result := '';

**finally** r.Free;

**end**;

**end**;

**begin**

ParseTel ('El telйfono de AlkorSoft (proyecto PayCash) es +7(812) 329-44-69',

'Cуdigo de Paнs $1, cуdigo de ciudad $2. El nъmero telefуnico completo es $&.');

*// devuelve 'Cуdigo de Paнs +7, cуdigo de ciudad (812) . El nъmero telefуnico completo es +7(812) 329-44-69.'*

**end**.

**Ejemplos mбs complejos**

Se pueden encontrar ejemplos mбs complejos del uso de TRegExpr en el proyecto [TestRExp.dpr](#tregexpr_testrexp) y [HyperLinkDecorator.pas](#HyperLinksDecorator).

Ver tambiйn mis artнculos en [Delphi3000.com](`http://www.delphi3000.com/member.asp?ID=1300',`',1,`') (Inglйs) y [Delphi Kingdom](`http://delphi.vitpc.com/mastering/strings_birds_eye_view.htm',`',1,`') (Ruso).

**Explicaciуn detallada**

Por favor, ver la [descripcion](#TRegExpr_Interface) de la interface de TRegExpr.

# Sintaxis de las Expresiones Regulares

**Introducciуn**

Las Expresiones Regulares son un mйtodo ampliamente empleado para especificar "plantillas" de texto a buscar. Los **metacaracteres** especiales permiten especificar, por ejemplo, que una cadena en particular que se estб buscado aparezca al inicio o al fin de una lнnea, o que contenga **n** repeticiones de cierto caracter.

Las expresiones regulares lucen incomprensibles para los novatos, pero en realidad son muy simples (bueno, usualmente simples ;) ), y son una herramienta prбctica y poderosa.

Les recomiendo especialmente hacer pruebas con e.r. usando [TestRExp.dpr](#tregexpr_testrexp) - ayuda a comprender muchos conceptos. Ademбs hay muchos ejemplos predefinidos con comentarios, incluidos en este proyecto.

Iniciemos el recorrido de aprendizaje!

**Bъsquedas simples**

Cualquier caracter se encuentra a sн mismo, a menos que se trate de un **metacaracter** con significado especial, descriptos abajo.

Una serie de caracteres encuentra esa misma serie en la cadena objetivo, por lo tanto la plantilla "bluh" encontrarб "bluh'' en la cadena objetico. Simple, no?

Se puede conseguir que los caracteres que normalmente funcionan como **metacaracteres** o **secuencias de escape** sean interpretadas literalmente precediйndolas con el sнmbolo "\" (backslash). Por ejemplo, el metacaracter "^" significa inicio de la cadena, pero "\^" encuentra el sнmbolo "^", "\\" encuentra "\", y asн para todos los casos especiales.

**Ejemplos:**

foobar *encuentra la cadena 'foobar'*

\^FooBarPtr *encuentra '^FooBarPtr'*

**Secuencias de escape**

Algunos casos especiales pueden ser especificados usando sintaxis de **secuencias de escape** como las empleadas en C y Perl: "\n'' significa nueva lнnea, "\t'' equivale a tab, etc. Mбs generalmente, \xnn, donde nn es un nъmero hexadecimal, encuentra el caracter cuyo valor ASCII es nn. Para usar cуdigos dobles de Unicode, se puede especificar '\x{nnnn}', donde 'nnnn' - es uno o mбs valores hexadecimales.

\xnn *caracter con cуdigo hexadecimal nn*

\x{nn} *caracter con cуdigo hexadecimal nnnn (un byte para texto comъn y dos para* [*Unicode*](#TRegExpr_Interface)*)*

\t *tab (HT/TAB), lo mismo que \x09*

\n *lнnea nueva (NL), lo mismo que \x0a*

\r *retorno de carro (CR), lo mismo que \x0d*

\f *salto de hoja (FF), lo mismo que \x0c*

\a *alarma (bell) (BEL), lo mismo que \x07*

\e *escape (ESC), lo mismo que \x1b*

**Ejemplos:**

foo\x20bar *encuentra 'foo bar' (notar el espacio en el medio)*

\tfoobar *encuentra 'foobarar' precedido por tab*

**Clases de caracteres**

Se pueden especificar **clases de caracteres** encerrando una lista de caracteres entre corchetes [], la que que encontrarб **uno** cualquiera de los caracteres de la lista.

Si el primer sнmbolo despuйs deI "['' es "^'', la clase encuentra cualquier caracter que **no** estб en la lista.

**Ejemplos:**

foob[aeiou]r *encuentra las cadenas 'foobar', 'foober' etc. pero no 'foobbr', 'foobcr' etc.*

foob[^aeiou]r *encuentra las cadenas 'foobbr', 'foobcr' etc. pero no 'foobar', 'foober' etc.*

Dentro de una lista, el sнbolo "-'' es utilizado para especificar un **rango**, entonces a-z representa todos los caracteres entre "a'' y "z'' inclusive.

Para que "-'' forme parte de la clase hay que ubicarlo al inicio o final de la lista, o usar la secuencia de escape "\-". Para usar ']' en la lista hay que ubicarlo al inicio de la lista o usar la secuencia "\]".

**Ejemplos:**

[-az] *encuentra 'a', 'z' y '-'*

[az-] *encuentra 'a', 'z' y '-'*

[a\-z] *encuentra 'a', 'z' y '-'*

[a-z] *encuentra todas las minъsculas de 'a' hasta 'z'*

[\n-\x0D] *encuentra cualquiera de #10,#11,#12,#13.*

[\d-t] *encuentra cualquier dнgito, '-' or 't'.*

[]-a] *encuentra cualquier caracter de ']' hasta 'a'.*

**Metacaracteres**

Los metacaracteres son caracteres especiales que son la esencia de las Expresiones Regulares. Hay diferentes tipos:

**Metacaracteres - separadores de lнneas**

^ *inicio de lнnea*

$ *fin de lнnea*

\A *inicio de texto*

\Z *fin de texto*

. *cualquier caracter en la lнnea*

**Ejemplos:**

^foobar *encuentra 'foobar' sуlo si estб al principio de una lнnea*

foobar$ *encuentra 'foobar' sуlo si estб al final de una lнnea*

^foobar$ *encuentra 'foobar' sуlo si es la ъnica cadena en la lнnea*

foob.r *encuentra cadenas tipo 'foobar', 'foobbr', 'foob1r'*

El metacaracter "^" por defecto sуlo garantiza encontrar coincidencias al principio de la cadena/texto analizados, y "$" sуlo al final. Los separadores de lнnea intermedios no son encontrados por "^'' o "$''.

Sin embargo, se puede tratar una cadena como multilнnea, de esta forma "^'' encontrarб coincidencias despuйs de cualquier separador de lнnea dentro de esta cadena, y "$'' darб resultados positivos antes de cualquier separador. Esto se logra activando el modificador /m.

Las secuencias \A y \Z son como "^'' y "$'', excepto que no dan resultados mъltiples aunque estй activado el modificador /m, mientras que "^'' y "$'' encontrarбn coincidencias en todos los separadores de lнnea internos.

El metacaracter ".'' por defecto encuentra cualquier caracter, pero si se desactiva el modificador /s entonces '.' no encuentra separadores de lнnea internos.

TRegExpr trabaja con los separadores de lнnea segъn las recomendaciones de www.unicode.org ( http://www.unicode.org/unicode/reports/tr18/ ):

"^" al inicio de la cadena ingresada, y si el modificador /m estб activado, tambiйn inmediatamente despuйs de toda ocurrencia de \x0D\x0A, \x0A o \x0D (si se usa la versiуn [Unicode](#TRegExpr_Interface) de TRegExpr, tambiйn \x2028, \x2029, \x0B, \x0C o \x85). Notar que no hay una lнnea vacнa dentro de la secuencia \x0D\x0A.

"$" al final de la cadena ingresada, y si el modificador /m estб activado, tambiйn inmediatamente antes de toda ocurrencia de \x0D\x0A, \x0A, o \x0D (si se usa la versiуn Unicode de TRegExpr, tambiйn \x2028, \x2029, \x0B, \x0C o \x85). Notar que no hay una lнnea vacнa dentro de la secuencia \x0D\x0A.

"." encuentra cualquier caracter, pero si se desactiva el modificador /s entonces "." no encuentra \x0D\x0A, \x0A y \x0D (si se usa la versiуn Unicode de TRegExpr, tambiйn \x2028, \x2029, \x0B, \x0C y \x85).

Notar que "^.\*$" (plantilla de lнnea vacнa) no encuentra la cadena vacнa dentro de la secuencia \x0D\x0A, pero sн dentro de la secuencia \x0A\x0D.

El procesamiento multilнnea puede ser fбcilmenet afinado para sus propуsitos con la ayuda de las propiedades LineSeparators y LinePairedSeparator de TRegExpr. Se pueden usar sуlo separadores tipo Unix ( \n ) o sуlo tipo DOS/Windows ( \r\n) o todos juntos (como se describe arriba y usado por defecto), o incluso definir sus propios deparadores de lнnea !

**Metacaracteres - clases predefinidas**

\w *un caracter alfanumйrico (incluye "\_")*

\W *un caracter no alfanumйrico*

\d *un caracter numйrico*

\D *un caracter no numйrico*

\s *cualquier espacio (lo mismo que [ \t\n\r\f])*

\S *un no espacio*

Se pueden usar las clases \w, \d y \s dentro de las **clases de caracteres** personales.

**Ejemplos:**

foob\dr *encuentra cadenas como 'foob1r', ''foob6r' pero no 'foobar', 'foobbr'*

foob[\w\s]r *encuentra cadenas como 'foobar', 'foob r', 'foobbr' pero no 'foob1r', 'foob=r'*

TRegExpr usa las propiedades SpaceChars y WordChars para definir las clases de caracteres \w, \W, \s, \S, por lo tanto se pueden redefinir fбcilmente.

**NOTA PARA USUARIOS DE IDIOMA ESPAСOL:**

La propiedad WordChars por defecto estб definida con el siguiente conjunto de caracteres:

WordChars = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ\_'

Para nuestro idioma lo correcto serнa:

WordChars = '0123456789aбbcdeйfghiнjklmnсoуpqrstuъvwxyzAБBCDEЙFGHIJKLMNСOУPQRSTUЪVWXYZ\_'

**Metacaracteres - lнmites de palabras**

\b *encuentra lнmite de palabra*

\B *encuentra distinto a lнmite de palabra*

Un lнmite de palabra (\b) es un punto entre dos caracteres que estб limitado por un \w de un lado y un \W en el otro (en cualquier orden), contando los caracteres imaginarios del inicio y final de la cadena como coincidencias con \W.

**Metacaracteres - iteradores**

Cualquier item de una expresiуn regular puede ser seguido por otro tipo de metacaracteres, los **iteradores**. Usando estos metacaracteres se puede especificar el nъmero de ocurrencias del caracter previo, de un **metacaracter** o de una **subexpresion**.

\* *cero o mбs ("voraz"), similar a {0,}*

+ *una o mбs ("voraz"), similar a {1,}*

? *cero o una ("voraz"), similar a {0,1}*

{n} *exactamente n veces ("voraz")*

{n,} *por lo menos n veces ("voraz")*

{n,m} *por lo menos n pero no mбs de m veces ("voraz")*

\*? *cero o mбs ("no voraz"), similar a {0,}?*

+? *una o mбs ("no voraz"), similar a {1,}?*

?? *cero o una ("no voraz"), similar a {0,1}?*

{n}? *exactamente n veces ("no voraz")*

{n,}? *por lo menos n veces ("no voraz")*

{n,m}? *por lo menos n pero no mбs de m veces ("no voraz")*

Entonces, los dнgitos entre llaves de la forma {n,m}, especifican el mнnimo nъmero de ocurrencias en n y el mбximo en m. La forma {n} es equivalente a {n,n} y coincide exactamente n veces. La forma {n,} encuentra ocurrencias de n o mбs veces. No hay lнmites para los nъmero n o m, pero si son muy grandes se consume mбs memoria y la ejecuciуn de la e.r. se hace mбs lenta.

Si una llave aparece en otro contexto se la trata como un caracter regular.

**Ejemplos:**

foob.\*r *encuentra cadenas como 'foobar', 'foobalkjdflkj9r' y 'foobr'*

foob.+r *encuentra cadenas como 'foobar', 'foobalkjdflkj9r' pero no 'foobr'*

foob.?r *encuentra cadenas como 'foobar', 'foobbr' y 'foobr' pero no 'foobalkj9r'*

fooba{2}r *encuentra la cadena 'foobaar'*

fooba{2,}r encuentra cadenas como *'foobaar', 'foobaaar', 'foobaaaar' etc.*

fooba{2,3}r *encuentra cadenas como 'foobaar', o 'foobaaar' pero no 'foobaaaar'*

Una mнnima explicacion acerca de la "voracidad". "Voraz" toma la mayor cantidad posible, "no voraz" toma la mнnima cantidad posible. Por ejemplo, 'b+' y 'b\*' aplicados a la cadena 'abbbbc' devuelven 'bbbb', 'b+?' devuelve 'b', 'b\*?' devuelve unacadena vacнa, 'b{2,3}?' devuelve 'bb', 'b{2,3}' devuelve 'bbb'.

Se pueden activar todos los iteradores para que funcione en modo "no voraz" (ver el modificador /g).

**Metacaracteres - alternativas**

Se puede especificar una serie de **alternativas** para una plantilla usando "|'' para separarlas, entonces fee|fie|foe encontrarб cualquier "fee'', "fie'', o "foe'' en la cadena objetivo (lo mismo serнa f(e|i|o)e). La primera alternativa incluye todo desde el ultimo delimitador ( "('', "['', o el inicio de la plantilla) hasta el primer "|'', y la ъltima alternativa contiene todo desde el ъltimo "|''hasta el siguiente delimitador de plantilla. Por esta razуn es una prбctica comъn incluir las alternativas entre parйntesis, para minimizar la confusiуn de dуnde se inician y dуnde terminan.

Las alternativas son evaluadas de izquierda a derecha, por lo tanto la primera alternativa que coincide plenamente con la expresiуn analizada es la que se selecciona. Esto significa que las alternativas no son necesariamente voraces. Por ejemplo: si se buscam foo|foot en "barefoot'', sуlo la parte "foo'' da resultado positivo, porque es la primera alternativa probada, y porque tiene йxito en la bъsqueda de la cadena analizada. (Esto puede no parecer importante, pero lo es cuando se estб capturando el texto buscado usando parйntesis.)

Tambiйn recordar que "|'' es interpretado literalmente dentro de los corchetes, entonces si se escribe [fee|fie|foe] lo ъnico que se encuentra es [feio|].

**Ejemplo:**

foo(bar|foo) *encuentra las cadenas 'foobar' o 'foofoo'.*

**Metacaracteres - subexpresiones**

La construcciуn ( ... ) tambiйn puede ser empleada para definir subexpresiones de e.r. (despuйs del anбlisis se obtienen las posiciones de las subexpresiones, su longitud y el valor actual en las propiedades MatchPos, MatchLen y Match de TRegExpr; y se pueden substituir en cadenas de plantillas con TRegExpr.Substitute).

Las subexpresiones son numeradas de izquierda a derecha en base al orden de sus parйntesis de apertura. La primera subexpresiуn es la '1' (la e.r. completa tiene el nъmero '0' - Se puede substituir en TRegExpr.Substitute como '$0' o '$&').

**Ejemplos:**

(foobar){10} *encuentra cadenas que contienen 8, 9 o 10 instancias de 'foobar'*

foob([0-9]|a+)r *encuentra 'foob0r', 'foob1r' , 'foobar', 'foobaar', 'foobaar' etc.*

**Metacaracteres - memorias (backreferences)**

Los **metacaracteres** \1 a \9 son interpretados como memorias. \<n> encuentra la subexpresiуn previamente encontrada #<n>.

**Ejemplos:**

(.)\1+ *encuentra 'aaaa' y 'cc'.*

(.+)\1+ *tambiйn encuentra 'abab' y '123123'*

(['"]?)(\d+)\1 *encuentra '"13" (entre comillas dobles), o '4' (entre comillas simples) o 77 (sin comillas) etc*

**Modificadores**

Los modificadores son para cambiar el comportamiento de TRegExpr.

Hay varias formas de configurar los modificadores.

Cualquiera de estos modificadores pueden incluirse dentro de las expresiones regulares usando la estructura (?...).

Tambiйn se pueden asignar las propiedades adecuadas de TRegExpr (ModifierX por ejemplo, para cambiar /x, o ModifierStr para cambiar todos los modificadores simultбneamente). Los valores por defecto para nuevas instancias de TRegExpr estбn definidos en variables globales, por ejemplo la variable global RegExprModifierX define el valor de la propiedad ModifierX en las nuevas instacias del objeto TRegExpr.

**i**

Bъsquedas insensibles a mayъsculas, ver tambiйn InvertCase.

**m**

Tratamiento de cadenas como mъltiples lнneas. Esto es, cambia a "^'' y "$'' de encontrar solo en el inicio y fin de la cadena al inicio y fin de cada lнnea dentro de la cadena, ver tambiйn Separadores de lнneas.

**s**

Tratamiento de cadenas cуmo lнnea simple. Esto es, cambia ".'' para encontrar cualquier caracter en cualquier lado, incluso separadores de lнnea (ver Separadores de lнnea), que normalmente no son encontradod.

**g**

Modificador no standard. Al desactivarlo se cambian todos los operadores siguientes en modo no voraz (por defecto este modificador esta activado). Entonces, si el modificador /g estб Off entonces '+' funciona como '+?', '\*' como '\*?', etc.

**x**

Aumenta la legibilidad de las plantillas al permitir espacios en blanco y comentarios (ver la explicaciуn mбs abajo).

**r**

Modificador no standard para incluir letras rusas en el rango de caracteres.

Perdуn a los usuarios extranjeros, pero estб activado por defecto. Para desactivarlo por defecto cambiar a False la variable global RegExprModifierR.

El modificador /x necesita un poco mбs de explicaciуn. Le dice a TRegExpr que ignore los espacios blancos que no estбn precedidos por "\" o no incluнdos en una clase de caracteres. Se puede usar para separar las expresiones regulares en partes mбs legibles. El caracter # es tratado como metacaracter para comentarios, por ejemplo:

(

(abc) #comentario 1

| #Se pueden usar espacios para formatear e.r.

(efg) #comentario 2

)

Esto significa que si se desea usar realmente un espacio o # en una plantilla (fuera de clases de caracteres, donde no son afectadas por /x), hay que usar "\" o codificarlos en su valor ASCII en octal o hexadecimal.

**Extensiones de Perl**

(?imsxr-imsxr)

Se pueden usar dentro de las e.r. para cambiar modificaciones instantбneamente. Si esta construcciуn estб incluнda dentro de una subexpresiуn, entonces sуlo afecta a la subexpresiуn.

**Ejemplos:**

(?i)Saint-Petersburg *encuentra 'Saint-petersburg' y 'Saint-Petersburg'*

(?i)Saint-(?-i)Petersburg *encuentra 'Saint-Petersburg' pero no 'Saint-petersburg'*

(?i)(Saint-)?Petersburg *encuentra 'Saint-petersburg' y 'saint-petersburg'*

((?i)Saint-)?Petersburg *encuentra 'saint-Petersburg', pero no 'saint-petersburg'*

(?#text)

Comentario, el texto es ignorado. Notar que TRegExpr cierra el comentario apenas encuentra un ")", por lo tanto no hay forma de poner un ")" literal en el comentario.

**Explicaciуn del mecanismo interno**

Le parece que necesita algunos de los secretos internos de TRegExpr?

Bien, estб en construcciуn, por favor espere un tiempo...

Por ahora, no olvide de leer las [FAQ](#FAQ) (especialmente la [pregunta](#FAQ) de optimizaciуn de 'no voracidad').

# Interface de TRegExpr

**Mйtodos pъblicos y propiedades de TRegExpr**:

**funciуn VersionMajor** : integer;

**funciуn VersionMinor** : integer;

Devuelve versiones mayor y menor, por ejemplo, para v. 0.944 VersionMajor = 0 y VersionMinor = 944

**propiedad Expression** : string

Expresiуn regular.

Para optimizaciуn, TRegExpr la compilarб automбticamente en 'P-code' (se puede ver con el mйtodo Dump) y la almacena en estructuras internas. La [re]compilaciуn real ocurre sуlo cuando es necesario - al llamar a Exec[Next], Substitute, Dump, etc y sуlo si Expression u otra propiedad P-code fue modificada despuйs de la ъltima [re]compilaciуn.

Si se produce cualquier error durante la [re]compilaciуn de llama al mйtodo Error (por defecto Error genera una excepciуn - ver abajo)

**propiedad ModifierStr** : string

valores por defecto para los modificadores de e.r. El formato de la cadena es similar a (?ismx-ismx). Por ejemplo ModifierStr := 'i-x' activa el modificador /i, desactiva /x y deja sin cambios el resto.

Si se intenta cambiar un modificador inexistente, se llamarб al procedimiento Error (por defecto Error genera una excepciуn ERegExpr).

**propiedad ModifierI** : boolean

Modificador /i - ("insensible a mayъsculas"), inicializado por el valor de RegExprModifierI.

**propiedad ModifierR** : boolean

Modificador /r - ("Extensiones para Ruso"), inicializado con el valor de RegExprModifierR.

**propiedad ModifierS** : boolean

Modificador /s - '.' funciona como cualquier char (no encuentra Separadores de lнnea y LinePairedSeparator), inicializado con el valor de RegExprModifierS.

**propiedad ModifierG** : boolean;

Modificador /g Desactiva el modificador /g cambiando todos los operadores a estilo no voraz, entonces si ModifierG = False todos los '\*' trabajan como '\*?', todos los '+' como '+?', etc, inicializado con el valor de RegExprModifierG.

**propiedad ModifierM** : boolean;

Modificador /m Tratamiento de cadenas como lнneas mъltiples. Esto es, cambia `^' y `$' de encontrar sуlo al inicio o final de la cadena al inicio o final de cualquier salto de lнnea DENTRO de la cadena, inicializado con el valor de RegExprModifierM.

**propiedad ModifierX** : boolean;

Modificador /x - ("sintaxis eXtendida"), inicializado con RegExprModifierX.

**funciуn Exec** (const AInputString : string) : boolean;

ejecuta el programa sobre la cadena AInputString. Exec guarda AInputString en la propiedad InputString.

**funciуn ExecNext** : boolean;

busca la siguiente coincidencia:

Exec (AString); ExecNext;

funciona igual que

Exec (AString);

if MatchLen [0] = 0 then ExecPos (MatchPos [0] + 1)

else ExecPos (MatchPos [0] + MatchLen [0]);

pero es mбs simple !

**funciуn ExecPos** (AOffset: integer = 1) : boolean;

busca coincidencias en InputString comenzando el la posiciуn AOffset

(AOffset=1 - primer caracter de InputString)

**propiedad InputString** : string;

devuelve la cadena corriente (desde la ъltima llamada a Exec o la ъltima asignaciуn a esta propiedad).

Cualquier asignaciуn de esta propiedad limpia las propiedades Match\* !

**funciуn Substitute** (const ATemplate : string) : string;

Devuelve ATemplate con '$&' o '$0' reemplazados por la ocurrencia completa de la e.r. y '$n' reemplazado por la ocurrencia de la subexpresiуn #n.

Desde la v.0.929 '$' se usa en vez de '\' (para ampliaciones futuras y por mayor compatibilidad con Perl) y acepta mбs de un dнgito.

Si es necesario incluir en la plantilla los sнmbolos '$' o '\', usar el prefijo '\'

Ejemplo: '1\$ es $2\\rub\\' -> '1$ es <Match[2]>\rub\'

Si hay que incluir un dнgito despuйs de '$n' se debe delimitar n con llaves '{}'.

Ejemplo: 'a$12bc' -> 'a<Match[12]>bc', 'a${1}2bc' -> 'a<Match[1]>2bc'.

**procedimiento Split** (AInputStr : string; APieces : TStrings);

Parte AInputStr en APieces por las ocurrencias de la e.r.

Internamente llama a Exec[Next]

**function Replace** (AInputStr : RegExprString;

const AReplaceStr : RegExprString;

AUseSubstitution : boolean = False) : RegExprString;

Devuelve AInputStr con las ocurrencias de la e.r. reemplazadas por AReplaceStr

Si AUseSubstitution es verdadero se usa AReplaceStr como plantilla para mйtodos de sustituciуn.

Por ejemplo:

Expression := '({-i}block|var)\s\*\(\s\*([^ ]\*)\s\*\)\s\*';

Replace ('BLOCK( test1)', 'def "$1" value "$2"', True);

devolverб: def 'BLOCK' value 'test1'

Replace ('BLOCK( test1)', 'def "$1" value "$2"', False)

devolverб: def "$1" value "$2"

Internamente llama a Exec[Next]

**propiedad SubExprMatchCount** : integer; // ReadOnly

Nъmero de subexpresiones que han sido encontradas en la ъltima llamada a Exec\*.

Si no hay subexpresiones pero se encontrу la expresiуn entera (Exec\* devolviу True), entonces SubExprMatchCount=0, si no hay subexpresiones ni expresiуn completa de la e.r. encontradas (Exec\* devolviу False) entonces SubExprMatchCount=-1.

Por ejemplo: Expression := '(1)?2(3)?';

Exec ('123'): SubExprMatchCount=2, Match[0]='123', [1]='1', [2]='3'

Exec ('12'): SubExprMatchCount=1, Match[0]='12', [1]='1'

Exec ('23'): SubExprMatchCount=2, Match[0]='23', [1]='', [2]='3'

Exec ('2'): SubExprMatchCount=0, Match[0]='2'

Exec ('7') - devuelve False: SubExprMatchCount=-1

**propiedad MatchPos** [Idx : integer] : integer; // ReadOnly

Ubicaciуn de inicio de la subexpresiуn nъmero #Idx en la ejecuciуn de la ъltima llamada a Exec\*. La primera subexpresiуn tiene Idx=1, la ъltima es igual a MatchCount, la e.r. completa tiene Idx=0.

Devuelve -1 si en la e.r. no hay subexpresiones o no se encontraron en la cadena ingresada.

**propiedad MatchLen** [Idx : integer] : integer; // ReadOnly

Longitud de la cadena de la subexpresion nъmero Idx en la ejecuciуn de la ъltima llamada a Exec\*. La primera subexpresiуn tiene Idx=1, la ъltima es igual a MatchCount, la e.r. completa tiene Idx=0.

Devuelve -1 si en la e.r. no hay subexpresiones o no se encontraron en la cadena ingresada.

**propiedad Match** [Idx : integer] : string; // ReadOnly

== Copy(InputString, MatchPos [Idx], MatchLen [Idx])

Devuelve -1 si en la e.r. no hay subexpresiones o no se encontraron en la cadena ingresada.

**funciуn LastError** : integer;

Devuelve el cуdigo de identificaciуn del ъltimo error, 0 si no hay errores (No se puede usar si el mйtodo Error genera una excepciуn) y limpia el status interno a 0 (sin errores).

**funciуn ErrorMsg** (AErrorID : integer) : string; virtual;

Devuelve el mensaje de error de cуdigo AErrorID.

**propiedad CompilerErrorPos** : integer; // ReadOnly

Devuelve la posiciуn en la e.r. donde se detuvo el compilador.

Util para diagnosticar errores.

**propiedad SpaceChars** : RegExprString

Contiene los caracteres que son tratados como \s (inicializada con la constante global RegExprSpaceChars)

**propiedad WordChars** : RegExprString;

Contiene los caracteres que son tratados como \w (inicializada con la constante global RegExprWordChars)

**propiedad LineSeparators** : RegExprString

Caracteres que son separadores de lнnea (como \n en Unix), inicializada con la constante global RegExprLineSeparators)

Ver separadores de línea

**propiedad LinePairedSeparator** : RegExprString

Pares de separadores de lнnea (como \r\n en DOS y Windows).

Debe contener exactamente dos caracteres o ninguno, inicializada con la constante global RegExprLinePairedSeparator)

Ver separadores de línea

Por ejemplo, si se necesita comportamiento tipo Unix asignar a LineSeparators := #$a (caracter de lнnea nueva) y a LinePairedSeparator := '' (cadena vacнa), si se quiere aceptar como separadores de lнnea \x0D\x0A pero no \x0D o \x0A solos, entonces asignar LineSeparators := '' (cadena vacнa) y a LinePairedSeparator := #$d#$a.

Por defecto se usa el modo 'mixto' (definido en las constantes globales RegExprLine[Paired]Separator[s]): LineSeparators := #$d#$a; LinePairedSeparator := #$d#$a. El comportamiento de este modo es descripto con mбs detalle en la secciуn sintaxis.

**funciуn de clase InvertCaseFunction** (const Ch : REChar) : REChar;

Convierte Ch en mayъsculas si estб en minъsculas o en minъsculas si estб en mayъsculas (usa la configuraciуn local corriente)

**propiedad InvertCase** : TRegExprInvertCaseFunction;

Activar esta propiedad si se desea anular la funcionalidad de insensibilidad a mayъsculas.

Create la inicializa a RegExprInvertCaseFunction (InvertCaseFunction por defecto)

**procedimiento Compile**;

[Re]compila la e.r. Util para editores GUI de e.r. (para controlar la validez de todas las propiedades).

**funciуn Dump** : string;

Descarga una e.r. compilada en una forma vagamente comprensible.

**Constantes globales**

Valores por defecto de los modificadores:

**RegExprModifierI** : boolean = False; // TRegExpr.ModifierI

**RegExprModifierR** : boolean = True; // TRegExpr.ModifierR

**RegExprModifierS** : boolean = True; // TRegExpr.ModifierS

**RegExprModifierG** : boolean = True; // TRegExpr.ModifierG

**RegExprModifierM** : boolean = False; // TRegExpr.ModifierM

**RegExprModifierX** : boolean = False; // TRegExpr.ModifierX

**RegExprSpaceChars** : RegExprString = ' '#$9#$A#$D#$C;

// valor por defecto de la propiedad SpaceChars

**RegExprWordChars** : RegExprString =

'0123456789'

+ 'abcdefghijklmnopqrstuvwxyz'

+ 'ABCDEFGHIJKLMNOPQRSTUVWXYZ\_';

// valor por defecto de la propiedad WordChars

//NOTA AYUDA EN ESPAСOL

// agregar бйнуъсС

**RegExprLineSeparators** : RegExprString =

#$d#$a{$IFDEF UniCode}#$b#$c#$2028#$2029#$85{DIF};

// valor por defecto de la propiedad LineSeparators

**RegExprLinePairedSeparator** : RegExprString =

#$d#$a;

// valor por defecto de la propiedad LinePairedSeparator

**RegExprInvertCaseFunction**: TRegExprInvertCaseFunction = TRegExpr.InvertCaseFunction;

// valor por defecto de la propiedad

**Funciones globales prбcticas**

**funciуn ExecRegExpr** (const ARegExpr, AInputStr : string) : boolean;

True si la cadena AInputString es encontrada en la e.r. ARegExpr

Se genera una excepciуn si hay errores de sintaxis en ARegExpr

**procedure SplitRegExpr** (const ARegExpr, AInputSttStr : string; APieces : TStrings);

Parte la cadena AInputStr en subcadenas APieces por las ocurrencia de la e.r. ARegExpr

**funciуn ReplaceRegExpr** (const ARegExpr, AInputStr, AReplaceStr : string;

AUseSubstitution : boolean = False) : string;

Devuelve AInputStr con las ocurrencias de la e.r. reemplazadas por AReplaceStr

Si AUseSubstitution es verdadero AReplaceStr serб usado como plantilla para los mйtodos de sustituciуn.

Por ejemplo:

ReplaceRegExpr ('({-i}block|var)\s\*\(\s\*([^ ]\*)\s\*\)\s\*',

'BLOCK( test1)', 'def "$1" value "$2"', True)

devolverб: def 'BLOCK' value 'test1'

ReplaceRegExpr ('({-i}block|var)\s\*\(\s\*([^ ]\*)\s\*\)\s\*',

'BLOCK( test1)', 'def "$1" value "$2"')

devolverб: def "$1" value "$2"

**funciуn QuoteRegExprMetaChars** (const AStr : string) : string;

Reemplaza todos los metacaracteres por su representaciуn segura , por ejemplo 'abc$cd.(' es convertida en 'abc\$cd\.\('

Esta funciуn es prбctica para autogeneraciуn de e.r. a partir de datos del usuario.

**funciуn RegExprSubExpressions** (const ARegExpr : string;

ASubExprs : TStrings; AExtendedSyntax : boolean = False) : integer;

Genera una lista de subexpresiones encontradas en la e.r. ARegExpr

En ASubExps cada item representa una subexpresiуn, en el formato:

String - texto de la subexpresiуn (sin '()')

low word of Object - posiciуn inicial en ARegExpr, incluyendo '(' si existe! (la primera posiciуn es 1)

high word of Object - longitud, incluyendo el '(' inicial y el ')' final si existen!

AExtendedSyntax - debe ser True si el modificador /x estarб activado mientras se usa la e.r.

Prбctico para editores GUI de e.r., etc (se puede ver un ejemplo de uso en el proyecto [TestRExp.dpr](#tregexpr_testrexp))

**Resultado** **Significado**

**0** Exito. No se encontraron paréntesis desbalanceados;

**-1** No hay suficientes paréntesis de cierre ')';

**-(n+1)** En la posición **n** se encontró un '[' abriendo sin el correspondiente ']' de cierre;

**n** En la posición **n** se encontró un ')' cerrando sin el correspondiente '(' de apertura.

Si el resultado es <> 0, entonces ASubExprs puede contener items vacнos o ilegales

**Exception type**

El administrador de errores por defecto de TRegExpr genera una excepciуn:

**ERegExpr** = class (Exception)

public

ErrorCode : integer; // cуdigo de error. Los errores de compilaciуn son menores a 1000.

CompilerErrorPos : integer; // Posiciуn en la e.r. donde se ocurriу el error de compilaciуn

end;

**Cуmo usar Unicode**

TRegExpr ahora soporta UniCode, pero funciona **muy lentamente** :(

Quiйn quiere optimizarlo ? ;)

Usarlo exclusivamente si realmente se necesita soporte de Unicode !

Sacar el '.' en {.$DEFINE UniCode} en regexpr.pas. Despuйs de йsto todas las cadenas serбn tratadas como WideString.

# Proyecto demo (TestRExp)

Programa simple para explorar y probar e.r., distribuнda como cуdigo fuente (proyecto TestRExp.dpr) y TestRExp.exe.

Nota: usa algunas propiedades de VCL que sуlo existen en Delphi 4 o superior. Mientras se compila en Delphi 3 o Delphi 2 se recibirбn mensajes de error acerca de **propiedades desconocidas**. Se pueden ignorar, estas propiedades son sуlo para ajustar tamaсo y justificaciуn de componentes cuando el formulario cambia su tamaсo.

Con la ayuda de este programa se puede determinar fбcilmente el nъmero de subexpresiones, saltar a cualquiera de ellas (en el cуdigo de la e.r. o en los resultados de la cadena exprorada), probar las funciones Substitute, Replace y Split.

Ademбs se incluyen muchos ejemplos que se pueden usar mientras se aprende la sintaxis de e.r. o en la exploraciуn rбpida de las capacidades de TRegExpr.

# Ejemplo: Hyper Links Decorator

Funciones para decorar hipervнnculos mientras se convierte texto puro en HTML.

Por ejemplo, reemplaza '*http://anso.da.ru*' con '*<a href="http://anso.da.ru">anso.da.ru</a>*' o '*anso@mail.ru*' con '*<a href="mailto:anso@mail.ru">anso@mail.ru</a>*'.

**Funciуn DecorateURLs**

Busca y reemplaza hipervнnculos como 'http://...' or 'ftp://..' asн como vнnculos sin protocolo pero que comienzan con 'www.' Si quiere modificar direcciones de correo electrуnico tiene que usar la funciуn DecorateEmails (ver mбs abajo).

**function DecorateURLs** (const AText : string; AFlags : TDecorateURLsFlagSet = [durlAddr, durlPath]) : string;

**Descripciуn**

Devuelve el texto AText con los hipervнnculos decorados.

AFlags indica quй parte del hipervнnculo debe ser incluнda en la parte VISIBLE del link:

Por ejemplo, si el flag es [durlAddr] entonces el link 'http://anso.da.ru/index.htm' serб decorado como '<a href="http://anso.da.ru/index.htm">anso.da.ru</a>'

**type**

**TDecorateURLsFlags** = (durlProto, durlAddr, durlPort, durlPath, durlBMark, durlParam);

**TDecorateURLsFlagSet** = set of TDecorateURLsFlags;

**Descripciуn**

Estos son los valores posibles:

**Valor** **Significado**

durlProto Protocolo (como 'ftp://' or 'http://')

durlAddr Direcciуn TCP o nombre de dominio (como 'anso.da.ru')

durlPort Nъmero de puerto, si estб especificado (como ':8080')

durlPath Ruta al documento (como 'index.htm')

durlBMark Book mark (como '#mark')

durlParam Parбmetros de la URL (como '?ID=2&User=13')

**Funciуn DecorateEMails**

Reemplaza todos los e-mails de sintaxis correcta con '<a href="mailto:ADDR">ADDR</a>'. Por ejemplo, reemplaza 'anso@mail.ru' con '<a href="mailto:anso@mail.ru">anso@mail.ru</a>'.

**function DecorateEMails** (const AText : string) : string;

**Descripciуn**

Devuelve el texto AText con los e-mails decorados.

# FAQ

P.

**Cуmo puedo usar TRegExpr con Borland C++ Builder?**

Tengo un problema porque no hay un archivo de cabecera (.h or .hpp) disponible.

R.

· Agregar RegExpr.pas al proyecto bcb.

· Compilar el proyecto. Esto genera el archivo RegExpr.hpp.

· Ahora se puede escribir cуdigo que use la unidad RegExpr.

· No olvidar agregar #include "RegExpr.hpp" donde haga falta.

P.

**Porquй TRegExpr devuelve mбs de una lнnea?**

Por ejemplo, e.r. <font .\*> devuelve la primera lнnea <font, y entonces **el resto del archivo** incluso el ъltimo </html>...

R.

Por compatibilidad con versiones anteriores el modificador /s estб activado por defecto.

Desactivarlo y '.' encotrarб todo menos separadores de lнnea.

A propуsito, le sugiero '<font ([^\n>]\*)>', serб la URL en Match[1].

P.

**Porquй TRegExpr devuelve mбs de lo esperado?**

Por ejemplo, la e.r. '<p>(.+)</p>' aplicada a la cadena '<p>a</p><p>b</p>' devuelve 'a</p><p>b' pero no 'a' como esperaba.

R.

Por defecto, todos los operadores funcionan en modo 'voraz', entonces devuelven lo mбximo posible.

Para operaciуn 'no voraz' se pueden usar operadores no voraces como '+?' (nuevo en v. 0.940) o cambiar todos los operadores a modo 'no voraz' con la ayuda del modificador 'g' (usando las propiedades de TRegExpr o construcciones como '?(-g)' en la e.r.).

P.

**Cуmo se pueden descomponer textos como HTML con la ayuda de TRegExpr**

R.

Lo siento amigos, pero es prбcticamente imposible!

Por supuesto, se puede usar fбcilmente TRegExpr para extraer alguna informaciуn del HTML, como se muestra en mis ejemplos, pero para desomponer en forma precisa hay que usar un cуdigo real de descomposiciуn, no e.r.!

Pueden obtener la explicaciуn completa en el libro 'Perl Cookbook' de Tom Christiansen y Nathan Torkington. Brevemente, hay muchas construcciones que sуn fбcilmente descompuestas por el programa apropiado, pero en absoluto por una e.r., y un descomponedor real es MUCHO mбs rбpido porque la e.r. no hace simplemente una bъsqueda, incluye una optimizaciуn que puede llevar una gran cantidad de tiempo.

P.

**Hay forma de obtener mъltiples coincidencias de una plantilla en TRegExpr?**

R.

Se puede hacer un bucle e iterar una por una con el mйtodo ExecNext.

No se puede hacer mбs fбcil porque Delphi no es un intйrprete como Perl (y es un beneficio, los intйrpretes son muy lentos).

Para ver algъn ejemplo ver la implementaciуn del mйtodo TRegExpr.Replace, o los ejemplos en [HyperLinksDecorator.pas](#HyperLinksDecorator)

P.

**Estoy controlando entradas de usuarios. Porquй TRegExpr devuelve 'True' para cadenas incorrectas?**

R.

En muchos casos los usuarios de TRegExpr olvidan que las expresiones regulares son para BUSCAR en una cadena. Entonces, si se pretende que un usuario ingrese sуlo 4 dнgitos y se usa para ello la expresiуn '\d{4,4}', se puede errar la detecciуn de parбmetros incorrectos como '12345' o 'cualquier letra 1234 y cualquier otra cosa'. Hay que agregar control para inicio y fin de lнnea para asegurarse de que no hay nada alrededor: '^\d{4,4}$'.

P.

**Porquй los iteradores no voraces a veces funcionan como en modo voraz?**

Por ejemplo, la e.r. 'a+?,b+?' aplicada a la cadena 'aaa,bbb' encuentra 'aaa,b', pero deberнa No encontrar 'a,b' a causa de la no voracidad del primer iterador?

R.

Esta es la limitaciуn de las matemбticas usadas por TRegExpr (y de las e.r. de Perl y muchos Unix) - e.r. realiza sуlo una optimizaciуn de bъsqueda 'simple', y no trata de hacer **la mejor optimizaciуn**. En algunos casos esto es malo, pero en los comunes es mayor la ventaja que la limitaciуn - por motivos de rapidez y predecibilidad.

La regla principal - la e.r. antes que nada intenta encontrar coincidencia desde la posiciуn actual y sуlo si es completamente imposible avanzar un caracter e intentar nuevamente desde ese lugar. Entonces, si se usa 'a,b+?' se encuentra 'a,b', pero en el caso de 'a+?,b+?' es '**no recomendado**' (a causa de la no voracidad) **pero posible** encontrar mбs de una 'a', entonces TRegExpr lo hace y finalmente obtiene una correcta (pero no уptima) coincidencia. TRegExpr como las e.r. de Perl o Unix no intenta avanzar y volver a chequear - lo que serнa una 'mejor' coincidencia. Mбs aъn, esto no puede ser comparado en absoluto en tйrminos de 'mejor o pero coincidencia'.

Por favor, leer la secciуn '[Sintaxis](#RegExp_Syntax_Engine_Internals)' para una mayor explicaciуn explanation.

# Autor

Andrey V. Sorokin

Saint Petersburg, Russia

[anso@mail.ru](mailto:anso@mail.ru)

[http://RegExpStudio.com](http://RegExpStudio.com/)

**Por favor, si cree que encontrу un error o tiene preguntas acerca de TRegExpr, obtenga la ъltima versiуn TRegExpr de mi home page y lea las** [**FAQ**](#FAQ) **antes de enviarme la consulta a mн!**

Esta librerнa es derivada del cуdigo de Henry Spencer.

Mi trabajo fue la traducciуn de los fuentes de C a Object Pascal, implementar el entorno de objeto y agregar algunas nuevas funciones.

Muchas caracterнsticas son sugerencias e implementaciones parciales de usuarios de TRegExpr (ver los Agradecimientos).

---------------------------------------------------------------

Agradecimientos

---------------------------------------------------------------

· Guido Muehlwitz - encontrу y arreglу un feo error en el procesamiento de cadenas grandes

· Stephan Klimek - pruebas en CPPB y sugerencias/implementaciуn de muchas funciones

· Steve Mudford - implementaciуn del parбmetro Offset

· Martin Baur ([www.mindpower.com](`http://www.mindpower.com',`',1,`')) - Ayuda en Alemбn y ъtiles sugerencias

· Yury Finkel - implementaciуn de soporte para UniCode, encontrу y arreglу algunos errores

· Ralf Junker - Implementу algunas funciones, varias sugerencias de optimizacion

· Simeon Lilov - Ayuda en Bъlgaro

· Filip Jirsбk y Matthew Winter (wintermi@yahoo.com) - colaboraron en la implementaciуn del modo no voraz

· Kit Eason varios ejemplos para la secciуn de introducciуn

· Juergen Schroth - cazador de errores y sugerencias ъtiles

· Martin Ledoux - Ayuda en Francйs

· Diego Calp (info@agroideas.com) Ayuda en Espaсol

y muchos otros - por un gran trabajo detectando errores !

Todavнa estoy buscando ayuda para traducir este documento a otros idiomas

Keyword index

**No index entries found.**