# SYNTAXE des expressions RÉGULIÈRES PRCE, de NOTEPAD++ v6.0 et PLUS

Guy THEVENOT - 21 Août 2012 - Version 1.0

guy038 - guy.038@wanadoo.fr

ATTENTION :

Pour une bonne LISIBILITÉ et une NAVIGATION agréable, dans le FICHIER 'RegExp_N++.txt', OUVERT dans NOTEPAD++

se conformer aux INSTRUCTIONS suivantes :

- Choisir la police FIXE 'Courrier New', en STYLE 'Standard' et TAILLE '10'

- Ne PAS afficher les N°s de LIGNE ( " Préférences... ", Onglet " Zones d'édition " )

- Régler l'AFFICHAGE en mode NORMAL (100%), par le RACCOURCI clavier CTRL + `/` du PAVÉ NUMÉRIQUE

- Ne PAS demander des RETOURS à la LIGNE ( Choix " Retour automatique à la ligne " NON COCHÉ )

TABLE des MATIÈRES :

# I) REMARQUES PRÉLIMINAIRES

# II) SYNTAXE des expressions RÉGULIÈRES PCRE en partie RECHERCHE :

## 1) CARACTÈRES :

A) Un caractère LITTÉRAL, NON MÉTA-caractère
B) Un MÉTA-caractère LITTÉRAL
C) Un caractère en NOTATION HEXADÉCIMALE ou OCTALE
D) Un caractère d'ÉCHAPPEMENT
E) Un caractère d'ÉCHAPPEMENT ASCII
F) Un caractère JOKER
G) Un caractère d'une CLASSE de CARACTÈRES
H) Un caractère en DEHORS d'une CLASSE de CARACTÈRES
I) Un caractère d'une LISTE de CARACTÈRES
J) Un caractère en DEHORS d'une LISTE de CARACTÈRES
K) Un caractère DOUBLE ( Collating element )
L) Un caractère avec un Nom SYMBOLIQUE POSIX
M) Un caractère de CLASSE ÉQUIVALENTE
N) REMARQUE IMPORTANTE sur les CLASSES ou LISTES de forme NÉGATIVE
O) Exemple RÉCAPITULATIF

## 2) SÉQUENCES d'ÉCHAPPEMENT

## 3) ALTERNATIONS

## 4) ASSERTIONS :

A) TOUT DÉBUT du fichier COURANT
B) TOUTE FIN du fichier COURANT
C) FIN de la DERNIÈRE LIGNE du fichier COURANT
D) TOUT DÉBUT de LIGNE
E) TOUTE FIN de LIGNE
F) FRONTIÈRE de MOT
G) FRONTIÈRE de NON MOT

H) DÉBUT de MOT
I) FIN de MOT
J) FIN de la PRÉCÉDENTE recherche

5) COMMENTAIRES

6) GROUPES et RÉFÉRENCES :

A) Groupes de CAPTURE NON NOMMÉS
B) Groupes de CAPTURE NOMMÉS

C) RÉFÉRENCES à un groupe de CAPTURE, NON NOMMÉ :

a) Référence ARRIÈRE ABSOLUE
b) Référence ARRIÈRE RELATIVE
c) Référence de GROUPE ABSOLUE
d) Référence de GROUPE RELATIVE
e) RÉFÉRENCES (?0) ou (?R)

D) RÉFÉRENCES à un groupe de CAPTURE NOMMÉ :

a) Référence ARRIÈRE
b) Référence de GROUPE

E) Groupes NON CAPTURANTS :

a) Groupe NON CAPTURANT
b) Groupe NON CAPTURANT d'ALTERNATIVES

F) ANNEXES sur les GROUPES

G) RÉFÉRENCES RÉCURSIVES

7) OPTIONS :

A) l'OPTION i et -i
B) l'OPTION m et -m
C) l'OPTION s et -s
D) l'OPTION x et -x

E) PARTICULARITÉS dans NOTEPAD++
F) DÉFINITION de l'OBJET <Bloc>

G) SYNTAXE des OPTIONS :

a) La FORME (?<Bloc>)
b) La FORME ((?<Bloc>)....)
c) Les FORMES (?:(?<Bloc>)....) ou la forme SIMPLIFIÉE (?<Bloc>):....)

8) QUANTIFICATEURS :

A) La FORME {n}
B) La FORME {n,}
C) La FORME {n,m}
D) La FORME ?
E) La FORME *
F) La FORME +

G) TYPES des QUANTIFICATEURS :

a) NON POSSESSIF, de TYPE " Greedy " ( " Fougueux " )
b) NON POSSESSIFS, de TYPE " Lazy " ( " Paresseux " )
c) POSSESSIFS, de type " Greedy " IMPLICITE

d) EXEMPLES

H) Groupes ATOMIQUES
I) Groupes OPTIONNELS
J) DIFFÉRENCES entre QUANTIFICATEURS

9) LOOKAROUNDS :

A) La FORME (?= .....)
B) La FORME (?! .....)
C) La FORME (?<= .....)
D) La FORME (?<! .....)

10) Blocs CONDITIONNELS

## III) SYNTAXE des expressions RÉGULIÈRES PCRE en partie REMPLACEMENT :

7) GROUPEMENT LEXICAL

8) EXPRESSIONS CONDITIONNELLES

9) MODIFICATEURS de CASSE

## IV) SUPPRESSION ou AJOUT de LIGNES VIDES :

1) CAS du remplacement GLOBAL ( Bouton " Remplacer tout " )

2) CAS du remplacement PAS à PAS ( Bouton " Remplacer " )

## V) RECHERCHES et/ou REMPLACEMENTS STANDARD :

1) AJOUT d'une CHAÎNE Ch en DÉBUT de CHAQUE ligne, VIDE ou NON VIDE, d'un FICHIER

2) AJOUT d'une CHAÎNE Ch en DÉBUT de CHAQUE ligne, NON VIDE, d'un FICHIER

3) AJOUT d'une CHAÎNE Ch en FIN de CHAQUE ligne, VIDE ou NON VIDE, d'un FICHIER

4) AJOUT d'une CHAÎNE Ch en FIN de CHAQUE ligne, NON VIDE, d'un FICHIER

5) AJOUT d'une CHAÎNE Ch dans TOUTES les lignes VIDES d'un FICHIER

6) SUPPRESSION de 1 à n CARACTÈRES en DÉBUT de CHAQUE ligne d'un FICHIER

7) SUPPRESSION du PREMIER caractère de CHAQUE ligne d'un FICHIER

8) SUPPRESSION de 1 à n CARACTÈRES en FIN de CHAQUE ligne d'un FICHIER

9) SUPPRESSION du DERNIER caractère de CHAQUE ligne d'un FICHIER

10) SUPPRESSION de TOUTES les lignes VIDES d'un FICHIER, WINDOWS ( LIGNES finissant par CR + LF )

11) SUPPRESSION de TOUTES les lignes VIDES d'un FICHIER, UNIX ( LIGNES finissant par LF )

12) SUPPRESSION de TOUTES les lignes VIDES d'un FICHIER, MAC ( LIGNES finissant par CR )

13) SUPPRESSION de TOUTE ligne VIDE EXCÉDENTAIRE d'un FICHIER, WINDOWS ( LIGNES finissant par CR + LF )

14) SUPPRESSION de TOUTE ligne VIDE EXCÉDENTAIRE d'un FICHIER, UNIX ( LIGNES finissant par LF )

15) SUPPRESSION de TOUTE ligne VIDE EXCÉDENTAIRE d'un FICHIER, MAC ( LIGNES finissant par CR )

16) SUPPRESSION de TOUT caractère 'FIN de LIGNE' d'un FICHIER, WINDOWS ( LIGNES finissant par CR + LF )

17) SUPPRESSION de TOUT caractère 'FIN de LIGNE' d'un FICHIER, UNIX ( LIGNES finissant par LF )

18) SUPPRESSION de TOUT caractère 'FIN de LIGNE' d'un FICHIER, MAC ( LIGNES finissant par CR )

19) SUPPRESSION de TOUTES les lignes COMMENÇANT par l'expression RÉGULIÈRE Re

20) SUPPRESSION de TOUTES les lignes FINISSANT par l'expression RÉGULIÈRE Re

21) SUPPRESSION de TOUTES les lignes COMPORTANT l'expression RÉGULIÈRE Re

22) SUPPRESSION de TOUTES les lignes, NON VIDES, NE comportant PAS l'expression RÉGULIÈRE Re

23) SUPPRESSION de TOUTES les lignes NE comportant PAS l'expression RÉGULIÈRE Re

A) SUPPRESSION des SEULES lignes VIDES

B) SUPPRESSION des lignes NON VIDES, NE comportant PAS l'expression RÉGULIÈRE Re

24) SUPPRESSION de TOUTES les lignes d'EXACTEMENT n CARACTÈRES, NON 'FIN de LIGNE'

25) SUPPRESSION de TOUTES les lignes d'au MOINS n CARACTÈRES, NON 'FIN de LIGNE'

26) SUPPRESSION de TOUTES les lignes d'au PLUS n CARACTÈRES, NON 'FIN de LIGNE'

27) SUPPRESSION de TOUT texte, à PARTIR de la PREMIÈRE occurrence de Re, dans CHAQUE ligne d'un FICHIER

28) SUPPRESSION de TOUT texte, à PARTIR de la DERNIÈRE occurrence de Re, dans CHAQUE ligne d'un FICHIER

29) SUPPRESSION de TOUT texte, JUSQU'À la PREMIÈRE occurrence de Re, dans CHAQUE ligne d'un FICHIER

30) SUPPRESSION de TOUT texte, JUSQU'À la DERNIÈRE occurrence de Re, dans CHAQUE ligne d'un FICHIER

31) SUPPRESSION de TOUT texte, de la 1ÈRE à la DERNIÈRE occurrence de Re, dans CHAQUE ligne d'un FICHIER

32) SUPPRESSION de TOUT texte, ENTRE DEUX occurrences SUCCESSIVES de Re, dans CHAQUE ligne d'un FICHIER

33) SUPPRESSION de la Nème OCCURRENCE de Re dans CHAQUE ligne d'un FICHIER

34) SUPPRESSION de la Nème OCCURRENCE de Re d'un FICHIER

35) SUPPRESSION de TOUTE expression RÉGULIÈRE Re, entre les COLONNES n et m, dans TOUTES les LIGNES

36) SUPPRESSION de TOUTE expression RÉGULIÈRE Re, en COLONNE n, dans TOUTES les LIGNES

37) SUPPRESSION de TOUTE Re, SÉPARÉE de la FIN de LIGNE par n à m CARACTÈRES, dans TOUTES les LIGNES

38) SUPPRESSION de TOUTE Re, SÉPARÉE de la FIN de LIGNE par n CARACTÈRES, dans TOUTES les LIGNES

39) SUPPRESSION de TOUT caractère, en COLONNE n, dans TOUTES les lignes d'un FICHIER

40) SUPPRESSION de TOUT texte, ENTRE les COLONNES n et m, dans TOUTES les lignes d'un FICHIER

41) SUPPRESSION de TOUT texte, à COMPTER de la COLONNE n, dans TOUTES les lignes d'un FICHIER

42) SUPPRESSION de TOUT texte, JUSQU'à la COLONNE n, dans TOUTES les lignes d'un FICHIER

43) SUPPRESSION de TOUT texte, en DEHORS des COLONNES n à m, dans TOUTES les lignes d'un FICHIER

44) AJOUT d'une CHAÎNE Ch, TOUS les n CARACTÈRES, dans TOUTES les lignes d'un FICHIER

45) AJOUT d'une 'FIN de LIGNE', TOUS les n CARACTÈRES, dans TOUTES les lignes d'un FICHIER

46) EXTRACTION du SEUL texte, SATISFAISANT la RegExp Re, de TOUTES les lignes d'un FICHIER

47) RECHERCHE du PREMIER DÉLIMITEUR #, APRÈS n COUPLES #.....# ( avec n >=0 ), dans CHAQUE ligne

48) RECHERCHE du PREMIER DÉLIMITEUR # ou @, APRÈS n COUPLES #.....@ ( avec n >=0 ), dans CHAQUE ligne

49) INTERVERSION de DEUX zones de TEXTE CONTIGUËS dans TOUTES les lignes d'un FICHIER

50) DÉPLACEMENT et/ou SUPPRESSION de ZONES de TEXTE, DÉTERMINÉES en COLONNES ( Cas GÉNÉRAL )

51) DÉPLACEMENT et/ou SUPPRESSION de ZONES de TEXTE, DÉTERMINÉES par un SÉPARATEUR ( Cas GÉNÉRAL )

52) SUPPRESSION des MOTS IDENTIQUES, EXCÉDENTAIRES, de CHAQUE ligne d'un FICHIER

53) SUPPRESSION des LIGNES IDENTIQUES, EXCÉDENTAIRES, d'un FICHIER

54) REMPLACEMENT SIMULTANÉ de PLUSIEURS MOTS ou CARACTÈRES par d'AUTRES MOTS ou CARACTÈRES

## I) REMARQUES PRÉLIMINAIRES :

Ce présent MANUEL vous est proposé en TROIS versions :

- une version PDF, nommée RegExp_N++.pdf, au titre de fichier ORIGINEL, auquel on pourra faire RÉFÉRENCE
- une version HTML, nommée RegExp_N++.html, au titre de fichier ORIGINEL, auquel on pourra faire RÉFÉRENCE
- une version TEXTE, nommée RegExp_N++.txt, à laquelle vous pourrez AJOUTER vos PROPRES notes et RegExp !

L'INTÉGRALITÉ de ce TUTORIAL a été rédigé en FRANÇAIS, ( ma langue MATERNELLE ! ), car il m'a semblé que la PRÉCISION et la CLARTÉ des INFORMATIONS était le point PRIMORDIAL pour une bonne COMPRÉHENSION des DIFFÉRENTS éléments de la SYNTAXE des expressions RÉGULIÈRES PCRE

=> Les NOMS des FENÊTRES, et de leurs RUBRIQUES, sont ceux du FICHIER 'french.xml', par DÉFAUT

REMARQUE :

On notera la PIÈTRE traduction FRANÇAISE de la NOUVELLE case à COCHER " . matches newline ", qui est " . comprend nouvelle ligne " !!

PERSONNELLEMENT, je préfère la FORMULATION " . = TOUT caractère "

TOUTE personne, de langue MATERNELLE ANGLAISE, qui maîtrise CORRECTEMENT le FRANÇAIS, est donc cordialement INVITÉE à EXTRAIRE, de ce MANUEL, une TRADUCTION ANGLAISE décente !! ( I am much obliged to you ! )

Pour les utilisateurs PRESSÉS, ou ne désirant PAS s'INVESTIR, dans un PREMIER temps, dans la COMPRÉHENSION de cette SYNTAXE, lire ce PREMIER chapitre, puis CONSULTER les TROIS DERNIERS chapitres, AVANT la CONCLUSION, intitulés :

- SUPPRESSION et AJOUT de LIGNES VIDES
- RECHERCHES et/ou REMPLACEMENTS STANDARD
- Exemple FINAL avec EXPLICATIONS DÉTAILLÉES

Ce DESCRIPTIF de SYNTAXE n'est VALABLE QUE si les RECHERCHES et/ou REMPLACEMENTS se font DANS l'éditeur NOTEPAD++, lui-même, à COMPTER de la VERSION 6.0 et NON par un AUTRE logiciel ou par une VERSION de

Notepad++ ANTÉRIEURE à la 6.0 !

REMARQUE :

Pour les RÉFRACTAIRES à la version UNICODE de NOTEPAD++, il est POSSIBLE d'utiliser le NOUVEAU moteur de RECHERCHE-REMPLACEMENT et les expressions RÉGULIÈRES PRCE, avec la version 5.9.8 ANSI de NOTEPAD++ !!

=> Il suffit de REMPLACER le fichier 'SciLexer.dll', de la version 5.9.8 ANSI, par le PLUS RÉCENT fichier 'SciLexer.dll' ( ACTUELLEMENT, celui de la version UNICODE 6.1.5 )

TOUTEFOIS, en cas d'ERREUR de SYNTAXE, dans les zones de RECHERCHE et/ou de REMPLACEMENT, le MESSAGE ' Invalid regular expression ' n'apparaît JAMAIS. SEULS, les PREMIERS caractères du fichier COURANT sont SÉLECTIONNÉS, à TORT !

Dans CERTAINS cas, la RECHERCHE peut aussi partir dans une BOUCLE INFINIE, nécessitant l'ARRÊT de NOTEPAD++, par le GESTIONNAIRE de TÂCHES, SANS SAUVEGARDE des fichiers de CONFIGURATION !!

=> Il sera donc PRÉFÉRABLE d'ENREGISTRER TOUS vos fichiers, AVANT de TENTER une opération de RECHERCHE-REMPLACEMENT COMPLEXE, de votre cru !!

Ce présent MANUEL est le MÉLANGE, que j'espère RÉUSSI :

- de mes DIFFÉRENTS essais PERSONNELS avec NOTEPAD++ 6.0 et PLUS

- de mes connaissances ANTÉRIEURES sur les expressions RÉGULIÈRES du monde UNIX

- de diverses DOCUMENTATIONS, trouvées sur INTERNET ( liste NON EXHAUSTIVE ) :

http://www.boost.org/doc/libs/1_48_0/libs/regex/doc/html/boost_regex

http://sourceforge.net/apps/mediawiki/notepad-plus/index.php?title=Regular_Expressions

http://mushclient.com/pcre/pcrepattern.html

http://www.regular-expressions.info/tutorial.html

http://msdn.microsoft.com/en-us/library/az24scfc%28v=vs.71%29.aspx

http://en.wikipedia.org/wiki/Regular_expression

http://en.wikipedia.org/wiki/Perl_Compatible_Regular_Expressions

http://docs.python.org/library/re.html

http://www.regextester.com/pregsyntax.html

http://perldoc.perl.org/perlrequick.html

http://perldoc.perl.org/perlretut.html

http://perldoc.perl.org/perlreftut.html

http://perldoc.perl.org/perlunitut.html

http://www.lumadis.be/regex/tuto_pcre.php

en OMETTANT toutes les ERREURS de TYPOGRAPHIE qu'elles contiennent, ce qui est un COMBLE, quand

on sait que la SYNTAXE des expressions RÉGULIÈRES est TRÈS PRÉCISE et SENSIBLE à la CASSE !!!

Dans la FENÊTRE " Rechercher " ( CTRL [ + SHIFT ] + F ou CTRL + H ) :

- les OPTIONS " Respecter la casse " et " Boucler " sont supposées COCHÉES
- le MODE de RECHERCHE " Expression régulière " est supposé COCHÉ

pour la RÉUSSITE des exemples CI-APRÈS !

Dans la FENÊTRE " Find/Replace " ( CTRL + R ) :

- les OPTIONS " Match case " et " Wrap " sont supposées COCHÉES
- le MODE de RECHERCHE " Regular Expr " est supposé COCHÉ

pour la RÉUSSITE des exemples CI-APRÈS !

La dénomination 'FIN de LIGNE' représente, au CHOIX :

- les DEUX caractères \r\n ( \x0D\x0A ) pour un FICHIER de type WINDOWS
- le caractère UNIQUE \n ( \x0A ) pour un FICHIER de type UNIX

- le caractère UNIQUE \r ( \x0D ) pour un FICHIER de type MAC

- le caractère UNIQUE \f ( \x0C ) QUEL que SOIT le TYPE du fichier

Dans les exemples, CI-APRÈS, on supposera opérer sur des FICHIERS de type WINDOWS, MÊME s'ils contiennent CERTAINES lignes terminées par le SEUL caractère \n ou \r

=> 'FIN de LIGNE' signifie la SEULE suite des DEUX caractères \r\n ou le SEUL caractère \f ( \x0c )

TRÈS IMPORTANT :

Dans les ZONES de SAISIE des FENÊTRES " Rechercher " et " Find/Replace ", BIEN s'assurer de NE PAS placer des ESPACES EXCÉDENTAIRES, en TÊTE et/ou en QUEUE de la SAISIE, SAUF si ces caractères ESPACES sont bien DÉSIRÉS

Dans le cas CONTRAIRE, il est FORT probable que l'expression RÉGULIÈRE cherchée NE sera PAS trouvée !

Dans les ZONES de SAISIE des FENÊTRES " Rechercher " et " Find/Replace ", BIEN s'assurer de la CASSE des ÉLÉMENTS, constitutifs des expressions RÈGULIÈRES saisies :

En effet, les SYMBOLES \r et \R , par exemple, n'ont ABSOLUMENT PAS la même SIGNIFICATION !!

et PAREILLEMENT pour \a et \A , \G1 et \g1 , \L et \l , \Ca et \cA , .....

SINON, il est FORT probable que l'expression RÉGULIÈRE cherchée NE sera PAS trouvée !

NOTES :

La SYNTAXE des expressions RÉGULIÈRES de NOTEPAD++ correspond, GLOBALEMENT, à la BIBLIOTHÈQUE Open Source PCRE ( PERL Compatible Regular Expressions ) de PERL Version 5.10

EXCEPTIONS :

Certaines ENTITÉS spécifiques, ci-DESSOUS, sont INOPÉRANTES avec la version PCRE de NOTEPAD++ :

- Les CLASSES de caractères UNICODE, telles que, par exemple : \p{Lu}, \pM, \P{IsCntrl}
- Les formes d'ÉVALUATION de code PERL, telles que, par exemple : (?{PRINT "Hello";})
- Les VERBES de CONTRÔLE, tels que, par exemple : (*FAIL), (*PRUNE:Name)
- Les DIRECTIVES ( PRAGMA ) telles que, par exemple : 'eval', 'taint', 'debug'

PRCE et PERL ont une OPTIMISATION, permettant l'ÉCHEC RAPIDE de la correspondance, lorsqu'un SEUL caractère LITTÉRAL est placé en FIN du gabarit de RECHERCHE.

EXEMPLE : Si le GABARIT de recherche a*@ est appliqué à la chaîne SUJET 'aaaa....aaaa', l'ÉCHEC

survient IMMÉDIATEMENT car le DERNIER caractère 'a' de la chaîne SUJET est, à

l'évidence, bien DIFFÉRENT du symbole AROBASE '@', placé en FIN du GABARIT

Cette DESCRIPTION se décompose en CINQ parties PRINCIPALES :

- La SYNTAXE des expressions RÉGULIÈRES PCRE, dans la zone de SAISIE de RECHERCHE
- La SYNTAXE des expressions RÉGULIÈRES PCRE, dans la zone de SAISIE de REMPLACEMENT
- La SUPPRESSION ou l'AJOUT de lignes VIDES
- Les RECHERCHES et/ou les REMPLACEMENTS STANDARD
- Un exemple FINAL avec EXPLICATIONS DÉTAILLÉES

REMARQUE :

Si vous n'appréciez PAS le SOULIGNEMENT SYSTÉMATIQUE, utilisé dans ce MANUEL, il est POSSIBLE de le SUPPRIMER, dans sa version TEXTE, par l'opération de RECHERCHE-REMPLACEMENT ci-dessous :

- Ouvrir la BOITE de dialogue " Remplacer " par le raccourci CTRL + H
- Écrire ([ ¯]+)(et|ou)?(?1)(\R) en zone de RECHERCHE ( ESPACE après le CROCHET OUVRANT ! )
- Écrire ?2\1\2\3 en zone de REMPLACEMENT ( Bien COMMENCER par ?2 )
- Ne PAS cocher la CASE " Respecter la Casse "

- COCHER la CASE " Boucler "

- Sélectionner le bouton " Expression Régulière "

- Cliquer sur le BOUTON " TOUT remplacer " ( DURÉE du REMPLACEMENT : environ 5 SECONDES )

Le RÔLE des différents ÉLÉMENTS, des zones de RECHERCHE et de REMPLACEMENT, sera DÉCRIT par la SUITE

PAS de PANIQUE !!!

## II) SYNTAXE des expressions RÉGULIÈRES PCRE en partie RECHERCHE :

### 1) CARACTÈRES :

TOUT caractère d'une expression RÉGULIÈRE, en partie RECHERCHE, est apparié à LUI-MÊME, SAUF :

- le caractère POINT ( . )
- le caractère CROCHET OUVRANT ( [ )
- le caractère ACCOLADE OUVRANTE ( { )
- le caractère ACCOLADE FERMANTE ( } )
- le caractère PARENTHÈSE OUVRANTE ( ( )
- le caractère PARENTHÈSE FERMANTE ( ) )
- le caractère ANTISLASH ( \ )
- le caractère ASTÉRISQUE ( * )
- le caractère ADDITION ( + )
- le caractère POINT d'INTERROGATION ( ? )
- le caractère ACCENT CIRCONFLEXE ( ^ )
- le caractère DOLLAR ( $ )
- le caractère BARRE VERTICALE ( | )

Ces 13 CARACTÈRES, appelés MÉTA-caractères, ont une signification SPÉCIALE, expliquée par la SUITE

Un caractère UNIQUE peut être, au CHOIX :

A) Un caractère LITTÉRAL, NON MÉTA-caractère :

TOUT caractère, DIFFÉRENT des MÉTA-caractères indiqués CI-DESSUS, se représente LUI-MÊME

EXEMPLE : l'expression RÉGULIÈRE 'A3@;zé' recherche la chaîne IDENTIQUE 'A3@;zé' !

NOTE :

Si l'OPTION " Respecter la casse " n'est PAS COCHÉE, l'expression RÉGULIÈRE 'A3@;zé' trouvera

UNE des HUIT chaînes ci-dessous :

A3@;ZÉ , A3@;Zé , A3@;zÉ , A3@;zé , a3@;ZÉ , a3@;Zé , a3@;zÉ , a3@;zé

REMARQUES :

Bien que NON nécessaire, un caractère, NON MÉTA-caractère, peut, également, être PRÉCÉDÉ du

caractère ANTISLASH ( \ )

EXEMPLE : la FORME \@ ( ou @ ) équivaut au caractère LITTÉRAL 'AROBASE' ( @ )

Les formes '\ + LETTRE', ci-dessous, NE participent PAS à la SYNTAXE des expressions RÉGULIÈRES, dans la partie RECHERCHE, et sont IDENTIQUES au caractère LETTRE, indiqué APRÈS le caractère ANTISLASH

\i , \j , \m , \o , \q , \y , \F , \I , \J , \M , \O , \T et \Y

EXEMPLE : La FORME \y ( ou y ) équivaut à la lettre MINUSCULE y

B) Un MÉTA-caractère LITTÉRAL :

Il est constitué du caractère ANTISLASH ( \ ), SUIVI d'un MÉTA-caractère et représente le MÉTA-caractère LITTÉRAL, SANS sa signification SPÉCIALE dans la syntaxe PCRE, en partie RECHERCHE

EXEMPLES :

la FORME * équivaut au caractère LITTÉRAL 'ASTÉRISQUE' ( * )

la FORME \$ équivaut au caractère LITTÉRAL 'DOLLAR' ( $ )

la FORME \\ équivaut au caractère LITTÉRAL 'ANTISLASH' ( \ )

REMARQUE :

Un MÉTA-caractère LITTÉRAL peut aussi s'employer ENTRE CROCHETS ( Voir PLUS LOIN )

Cependant, dans une LISTE entre CROCHETS, la PLUPART des MÉTA-caractères PERDENT leur signification SPÉCIALE

SEULS, TROIS d'entre eux devront, tout de même, être PRÉCÉDÉS d'un ANTISLASH (\)

Ce sont : l'ANTISLASH \\ , le TIRET HAUT \- et le CROCHET FERMANT \]

C) Un caractère en NOTATION HEXADÉCIMALE ou OCTALE :

```
\x0    , \x00     = le caractère de code HEXA     =   00  ( caractère Ascii NUL )
\x{00} , \x{0000} = le caractère de code HEXA     =   00  ( caractère Ascii NUL )


\xN               = le caractère de code HEXA     =   0N   avec N = [0-9A-Fa-f]
\xNN              = le caractère de code HEXA     =   NN   avec N = [0-9A-Fa-f]
\x{MN}            = le caractère de code HEXA     =   MN   avec N = [0-9A-Fa-f] et M = [0-7]

\x{NNNN}          = le caractère de code UNICODE  = NNNN   avec N = [0-9A-Fa-f] si fichier CODÉ en UTF-8

\0   , \00        = le caractère de code OCTAL    =  000  ( caractère Ascii NUL )
\000 , \0000      = le caractère de code OCTAL    =  000  ( caractère Ascii NUL )

\0N               = le caractère de code OCTAL    =  00N   avec N = [0-7]
\0NN              = le caractère de code OCTAL    =  0NN   avec N = [0-7]
\0MNN             = le caractère de code OCTAL    =  MNN   avec N = [0-7] et M = [0-3]
```

EXEMPLES et REMARQUES :

```
\011      représente le caractère de CONTRÔLE 'TAB' ( 11 OCTAL = 9 DÉCIMAL )
\0113     représente la lettre 'K' ( 113 OCTAL = 75 DÉCIMAL )
\011[3]   représente le caractère de CONTRÔLE 'TAB', suivi du CHIFFRE '3'

\01456    représente la CHAÎNE 'e6' ( 145 OCTAL = 101 DÉCIMAL = 'e' )
\014[5]6  représente le caractère de CONTRÔLE 'FF' suivi de la chaîne '56'

\007      représente le caractère de CONTRÔLE 'BEL' ( Alert )
\008      représente le caractère de CONTRÔLE 'NUL', suivi du CHIFFRE 8 ( PAS de car. '8' en OCTAL ! )
\08            "      "      "     "    "       "      "    "     "
\010      représente le caractère de CONTRÔLE 'BS' ( 10 OCTAL = 8 DÉCIMAL )

\x00      représente le caractère de CONTRÔLE 'NUL'
\x000     représente le caractère de CONTRÔLE 'NUL', suivi du CHIFFRE '0'
\x{0}00   représente le caractère de CONTRÔLE 'NUL', suivi de 2 CHIFFRES '00'

\x{263A}  représente le caractère ÉMO-ICÔNE :) ( Visage SOURIANT )  dans un fichier CODÉ en UTF-8
\x{20ac}  représente le caractère MONÉTAIRE EURO ( € )              dans un fichier CODÉ en UTF-8
```

NOTE :

Ces caractères, en NOTATION HEXA ou OCTALE, s'emploient aussi ENTRE CROCHETS ( Voir PLUS LOIN )

D) Un caractère d'ÉCHAPPEMENT :

```
\r    =   \x0D  (CR)   =   Caractère CARRIAGE RETURN  ( Caractère SÉPARATEUR de LIGNE )
\a    =   \x07  (BEL)  =   Caractère BELL             ( Caractère ALERTE )
\t    =   \x09  (TAB)  =   Caractère TABULATION       ( Caractère BLANC HORIZONTAL )
\e    =   \x1B  (ESC)  =   Caractère ESCAPE           ( Caractère ÉCHAPPEMENT )
\n    =   \x0A  (LF)   =   Caractère LINE FEED        ( Caractère SÉPARATEUR de LIGNE )
[\b]  =   \x08  (BS)   =   Caractère BACKSPACE        ( Caractère RETOUR ARRIÈRE )
```

\f = \x0C (FF) = Caractère FORM FEED ( Caractère SÉPARATEUR de LIGNE )

NOTE :

Ces caractères d'ÉCHAPPEMENT peuvent aussi s'employer ENTRE CROCHETS ( Voir PLUS LOIN )

REMARQUE :

[\b] représente le caractère BASKSPACE = \x08 )

\b est une ASSERTION, représentant une POSITION ( Voir PLUS LOIN )

( limite ENTRE un MOT et un NON MOT ou ENTRE un NON MOT et un MOT )

EXEMPLE :

\t\t\t cherche TROIS caractères TABULATION consécutifs

E) Un caractère d'ÉCHAPPEMENT ASCII :

\c@ = \c` = \x00 ( Caractère de CONTRÔLE 'NUL' = Caractère NULL )

\cA = \ca = \x01 ( Caractère de CONTRÔLE 'SOH' = START of HEADER )
\cB = \cb = \x02 ( Caractère de CONTRÔLE 'STX' = START of TEXT )
\cC = \cc = \x03 ( Caractère de CONTRÔLE 'ETX' = END of TEXT )
\cD = \cd = \x04 ( Caractère de CONTRÔLE 'EOT' = END of TRANSMISSION )
\cE = \ce = \x05 ( Caractère de CONTRÔLE 'ENQ' = ENQUIREMENT )
\cF = \cf = \x06 ( Caractère de CONTRÔLE 'ACK' = ACKNOWLEDGEMENT )
\cG = \cg = \x07 ( Caractère de CONTRÔLE 'BEL' = BELL )
\cH = \ch = \x08 ( Caractère de CONTRÔLE 'BS' = BACK SPACE )

```
\cI   =   \ci   =   \x09  (  Caractère de CONTRÔLE  'TAB'   =   HORIZONTAL TABULATION       )
\cJ   =   \cj   =   \x0A  (  Caractère de CONTRÔLE  'LF'    =   LINE FEED                   )
\cK   =   \ck   =   \x0B  (  Caractère de CONTRÔLE  'VT'    =   VERTICAL   TABULATION       )
\cL   =   \cl   =   \x0C  (  Caractère de CONTRÔLE  'FF'    =   FORM FEED                   )
\cM   =   \cm   =   \x0D  (  Caractère de CONTRÔLE  'CR'    =   CARRIAGE RETURN             )
\cN   =   \cn   =   \x0E  (  Caractère de CONTRÔLE  'SO'    =   SHIFT IN                    )
\cO   =   \co   =   \x0F  (  Caractère de CONTRÔLE  'SI'    =   SHIFT OUT                   )
\cP   =   \cp   =   \x10  (  Caractère de CONTRÔLE  'DLE'   =   DELETE                      )
\cQ   =   \cq   =   \x11  (  Caractère de CONTRÔLE  'DC1'   =   DEVICE CONTROL 1           )
\cR   =   \cr   =   \x12  (  Caractère de CONTRÔLE  'DC2'   =   DEVICE CONTROL 2           )
\cS   =   \cs   =   \x13  (  Caractère de CONTRÔLE  'DC3'   =   DEVICE CONTROL 3           )
\cT   =   \ct   =   \x14  (  Caractère de CONTRÔLE  'DC4'   =   DEVICE CONTROL 4           )
\cU   =   \cu   =   \x15  (  Caractère de CONTRÔLE  'NAK'   =   NEGATIVE ACKNOWLEDGEMENT   )
\cV   =   \cv   =   \x16  (  Caractère de CONTRÔLE  'SYN'   =   SYNCHRONISATION            )
\cW   =   \cw   =   \x17  (  Caractère de CONTRÔLE  'ETB'   =   END TRANSMISSION BLOCK     )
\cX   =   \cx   =   \x18  (  Caractère de CONTRÔLE  'CAN'   =   CANCEL                     )
\cY   =   \cy   =   \x19  (  Caractère de CONTRÔLE  'EM'    =   END of MEDIUM               )
\cZ   =   \cz   =   \x1A  (  Caractère de CONTRÔLE  'SUB'   =   SUBSTITUTION               )

\c[   =   \c{   =   \x1B  (  Caractère de CONTRÔLE  'ESC'   =   ESCAPE                     )
\c\   =   \c|   =   \x1C  (  Caractère de CONTRÔLE  'FS'    =   FILE   SEPARATOR           )
\c]   =   \c}   =   \x1D  (  Caractère de CONTRÔLE  'GS'    =   GROUP  SEPARATOR           )
\c^   =   \c~   =   \x1E  (  Caractère de CONTRÔLE  'RS'    =   RECORD SEPARATOR           )
\c_           =   \x1F  (  Caractère de CONTRÔLE  'US'    =   UNIT   SEPARATOR           )
```

NOTE :

Ces caractères d'ÉCHAPPEMENT peuvent aussi s'employer ENTRE CROCHETS ( Voir PLUS LOIN )

F) Un caractère JOKER :

Le . ( POINT ) ou \C représente TOUT caractère, ANSI ou UNICODE, AUTRE que 'LF', 'CR' et 'FF',

c'est à dire DIFFÉRENT du ou des caractères 'FIN de LIGNE'

=> \C = . = [^\r\n\f]

EXEMPLE :

abc..xyz représente une CHAÎNE 'abc', SÉPARÉE d'une CHAÎNE 'xyz', par DEUX caractères, AUTRES que les TROIS caractères \r , \n et \f

BUG :

(?s). avec la CASE " . comprend ligne nouvelle " COCHÉE ou NON ( Voir OPTIONS plus LOIN )

. avec la CASE " . comprend ligne nouvelle " COCHÉE

DEVRAIT trouver N'IMPORTE QUEL caractère, y COMPRIS \r, \n et \f, soit TOUS les caractères du fichier COURANT

MAIS, en fait, si l'on exécute :

- une RECHERCHE, PAS à PAS, avec le BOUTON " Suivant "
- un COMPTAGE, avec le BOUTON " Compter "
- une SUPPRESSION avec la ZONE de REMPLACEMENT laissée VIDE

SEULS, certains caractères sont pris en COMPTE ( et REMPLACÉS ) :

- TOUS les caractères, DIFFÉRENTS de \r et de \n, de TOUTES les lignes NON VIDES

- le CARACTÈRE \r ou \n ou la SUITE \r\n de TOUTES les lignes VIDES, EXCEPTÉ une ligne VIDE ÉVENTUELLE, constituant la PREMIÈRE ligne du fichier

De plus, si la ligne VIDE courante se TERMINE par le SEUL caractère \n ou \r, le MOTEUR de RECHERCHE " SAUTE " la ligne VIDE suivante, ÉVENTUELLE

PAR CONTRE, la RegExp (?s).+ trouve, EFFECTIVEMENT, TOUS les caractères du fichier COURANT et, si la ZONE de REMPLACEMENT est laissée VIDE, la SUPPRESSION, par le bouton " Remplacer tout ", de TOUS les caractères du fichier, est EFFECTIVE

SEULE, 1 ligne VIDE subsiste, si le fichier possèdait 1 ou PLUSIEURS lignes VIDES au DÉBUT

\X représente 1 SEUL caractère formé de :

a) un caractère, NON diacritique, UNIQUE

b) ZÉRO ou PLUS marques DIACRITIQUES d'association

NOTES :

Une marque DIACRITIQUE a une valeur UNICODE, HEXA, comprise, en général, entre 0300 et 036F

\X trouve TOUT caractère UNIQUE, EXCEPTÉ les marques DIACRITIQUES d'association ORPHELINES

EXEMPLE :

L'expression RÉGULIÈRE \X* représente donc, en GÉNÉRAL, TOUT le CONTENU d'un fichier, MÊME si celui-ci est VIDE => le résultat est IDENTIQUE au raccourci clavier CTRL + A !

\R représente un SEUL caractère de type 'FIN de LIGNE', c'est à dire :

- un caractère 'FIN de LIGNE' strict : \r\n ( CR+LF ) ou \n ( LF ) ou \r ( CR )
ou
- un caractère ASCII de type 'FIN de LIGNE' : \x0B ( VT ) ou \f ( FF ) ou \x85 ( … )
ou
- un caractère UNICODE de type 'FIN de LIGNE' : \x{2028} ou \x{2029}

=> \R = \r\n|[\n\v\f\r\x85\x{2028}\x{2029}]

EXEMPLES :

RegExp1\RRegExp2 cherche une expression RÉGULIÈRE RegExp1, SUIVIE d'une 'FIN de LIGNE', SUIVIE d'une expression RÉGULIÈRE RegExp2, en DÉBUT de la ligne SUIVANTE

RegExp\R cherche l'expression RÉGULIÈRE RegExp en FIN de CHAQUE ligne du fichier COURANT,

NOTE :

Si un FICHIER ne contient QUE des PURS caractères 'FIN de LIGNE' ( \n, \r ou \r\n ),
la FORME \R est PRÉFÉRABLE aux formulations \r , \n ou \r\n, car elle reste
VALABLE, QUEL que SOIT le TYPE du fichier ( WINDOWS, UNIX ou MAC ) et QUELLE que
SOIT la FIN de CHAQUE ligne rencontrée

REMARQUE :

Les caractères de type 'JOKER' ne sont SIGNIFICATIFS qu'en DEHORS des LISTES ou PLAGES de
caractères ENTRE CROCHETS ( Voir PLUS LOIN )

EXEMPLE :

[.\X\C\R] recherche le caractère POINT (.) ou l'UNE des LETTRES MAJUSCULES C, R ou X

CONSÉQUENCE :

Pour CHERCHER une expression RÉGULIÈRE RegExp1, SÉPARÉE d'une AUTRE expression RÉGULIÈRE RegExp2
par des caractères DIFFÉRENTS de \r , \n et \r\n, ou MÊME AUCUN ( autrement dit sur une

MÊME ligne ), la FORMULATION RegExp1[^\R]*RegExp2 est INCORRECTE !

=> Utiliser, UNIQUEMENT, la FORME RegExp1[^\r\n]*RegExp2, qui est EXACTE, QUEL que SOIT

le TYPE de 'FIN de LIGNE' et le TYPE du fichier COURANT ( WINDOWS, UNIX ou MAC )

De même, l'expression RÉGULIÈRE abc[^\r\n]* cherche la 1ERE chaîne 'abc' de CHAQUE ligne,

SUIVIE de TOUS les caractères RESTANTS de CHACUNE d'elles

L'expression RÉGULIÈRE similaire abc.* n'a PAS tout à fait le MÊME effet :

Soit la chaîne SUJET : 12345abcdef**FF**ghijk67890

alors, la RECHERCHE abc[^\r\n]* trouve la CHAÎNE 'abcdef**FF**ghijk67890'

MAIS, la RECHERCHE abc.* ne trouve QUE la CHAÎNE 'abcdef', car, par DÉFAUT,

le caractère**FF** n'est PAS reconnu par le MÉTA-caractère .

IMPORTANT :

Suite à un BUG de NOTEPAD++, la recherche de la SEULE SUITE de caractères CONSÉCUTIFS IDENTIQUES,

de type 'FIN de LIGNE', SANS caractère STANDARD placé(s) AVANT et/ou APRÈS cette SUITE, n'est

PAS POSSIBLE, si l'on utilise l'UNE des expressions RÉGULIÈRES, ci-dessous, RÉPÉTÉE n FOIS :

- \r\n\r\n\r\n.... pour un fichier de type WINDOWS
- \n\n\n.... pour un fichier de type UNIX
- \r\r\r.... pour un fichier de type MAC
- \R\R\R.... QUEL que SOIT le TYPE du fichier

En effet, le MOTEUR de RECHERCHE " SAUTE " TOUT caractère 'FIN de LIGNE' strict

( \r\n , \n ou \r )

- qui suit IMMÉDIATEMENT la PRÉCÉDENTE recherche IDENTIQUE \R , \n , \r ou \r\n
- qui DÉBUTE le fichier COURANT

Par CONTRE, l'utilisation du QUANTIFICATEUR + ( Voir PLUS LOIN ) permet, aux expressions RÉGULIÈRES, ci-dessous, la recherche CORRECTE de la SEULE SUITE de caractères 'FIN de LIGNE' CONSÉCUTIFS et IDENTIQUES, SANS caractère STANDARD placé(s) AVANT et/ou APRÈS cette SUITE

- (\r\n)+ pour un fichier de type WINDOWS
- \n+ pour un fichier de type UNIX
- \r+ pour un fichier de type MAC
- \R+ QUEL que SOIT le TYPE du fichier

Il est aussi POSSIBLE d'utiliser le MODE de RECHERCHE " Mode étendu (\n, \r, \t, \0, \x...) "

pour CHERCHER et/ou REMPLACER un PETIT nombre de caractères 'FIN de LIGNE' CONSÉCUTIFS,

en écrivant :

- \r\n\r\n.....\r\n pour un fichier de type WINDOWS
- \n\n.....\n pour un fichier de type UNIX
- \r\r.....\r pour un fichier de type MAC

NOTE :

Du fait des expressions RÉGULIÈRES AMÉLIORÉES, à compter de la VERSION 6.0 de NOTEPAD++,

l'intérêt du MODE de recherche ÉTENDU est désormais LIMITÉ !

En effet, à part l'EXEMPLE ci-dessus, SEULES les DEUX syntaxes \d... et \b........,

SPÉCIFIQUES au mode 'ÉTENDU' et SANS ÉQUIVALENT en mode 'Expression RÉGULIÈRE',

peuvent, encore, s'avérer UTILES

G) Un caractère d'une CLASSE de CARACTÈRES :

IMPORTANT :

Une CLASSE de caractères, POSIX ou NON, correspond TOUJOURS à un SEUL caractère

Elle possède PLUSIEURS syntaxes : - [[:<NOM de CLASSE:]] ou [[:<NOM de CLASSE à 1 Car.>:]]

- \p{<NOM de CLASSE}  ou  \p{<NOM de CLASSE à 1 Car.>}
- \p<NOM de CLASSE à 1 Car.>
- \<NOM de CLASSE à 1 Car. MINUSCULE>

| DANS Classe [..] | HORS Classe de Car. [..] | | | | CONTENU de la CLASSE de caractères | |
|---|---|---|---|---|---|---|
| [:space:] | [:s:] | \p{space} | \p{s} | \ps | \s | [\t\n\x0B\f\r\x20\xA0] | A |
| [:digit:] | [:d:] | \p{digit} | \p{d} | \pd | \d | [0-9¹²³] | B |
| [:lower:] | [:l:] | \p{lower} | \p{l} | \pl | \l | [a-zƒšœžªµºßàáâãäåæçèéêëìíîïðñòóôõöøùúûüýþÿ] | C |
| [:upper:] | [:u:] | \p{upper} | \p{u} | \pu | \u | [A-ZŠŒŽŸÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖØÙÚÛÜÝÞ] | D |
| [:word:] | [:w:] | \p{word} | \p{w} | \pw | \w | [_\d\l\u] | E |
| [:blank:] | [:h:] | \p{blank} | \p{h} | \ph | \h | [\t\x20\xA0] | F |
| [:v:] | | \p{v} | \p{v} | \pv | \v | [\n\x0B\f\r] ( Voir à EXCEPTION, ci-après ) | G |
| [:alnum:] | | \p{alnum} | | | | [\d\l\u] | H |
| [:alpha:] | | \p{alpha} | | | | [\l\u] | I |
| [:cntrl:] | | \p{cntrl} | | | | [\x00-\x1F\x7F\x81\x8D\x8F\x90\x9D] | J |
| [:graph:] | | \p{graph} | | | | [^\x00-\x20\x7F-\x81^\x8D\x8F\x90˜™\x9D\xA0] | K |
| [:print:] | | \p{print} | | | | [\s[:graph:]] | L |
| [:punct:] | | \p{punct} | | | | []\x21-\x2F:;<=>?@\\^_`{\|}~,„…†‡‰‹‘’“”•-—›\xA1-\xBF×÷[] | M |

| | | | | | | |
|---|---|---|---|---|---|---|
| [:xdigit:] | \p{xdigit} | | | | [0-9A-Fa-f] | N |
| [:unicode:] | \p{unicode} } | | | | [\x{0101}-\x{FFFF}] | O |

Ligne A = 1 Caractère BLANC, SÉPARATEUR de ligne ou NON
Ligne B = 1 Caractère CHIFFRE, y compris EXPOSANT
Ligne C = 1 Caractère ALPHABÉTIQUE MINUSCULE, ACCENTUÉ ou NON
Ligne D = 1 Caractère ALPHABÉTIQUE MAJUSCULE, ACCENTUÉ ou NON
Ligne E = 1 Caractère de MOT = Ligne B,C ou D ou TIRET BAS
Ligne F = 1 Caractère BLANC HORIZONTAL, NON SÉPARATEUR de ligne
Ligne G = 1 Caractère BLANC VERTICAL, SÉPARATEUR de ligne ( Voir à EXCEPTION, ci-après )
Ligne H = 1 Caractère ALPHANUMÉRIQUE, ACCENTUÉ ou NON = Ligne B,C ou D
Ligne I = 1 Caractère ALPHABÉTIQUE, ACCENTUÉ ou NON = Ligne C ou D
Ligne J = 1 Caractère de CONTRÔLE ou caractère NON VISUALISABLE
Ligne K = 1 Caractère VISIBLE ( SAUF les 4 caractères € ˆ ˜ ™ )
Ligne L = 1 Caractère AFFICHABLE ( Caractère VISIBLE ou BLANC ) = Ligne A ou K
Ligne M = 1 Caractère de PONCTUATION, 1 SYMBOLE, 1 ACCENT, 1 signe ou valeur MATHÉMATIQUE, 1 symbole MONÉTAIRE
Ligne N = 1 Caractère HEXADÉCIMAL
Ligne O = 1 Caractère UNICODE de code > 256 ( \x{0100} ) Bug : Le caractère \x100 n'est PAS considéré UNICODE !

IMPORTANT :

Les CLASSES de type POSIX [:....:] sont utilisées UNIQUEMENT dans une expression entre CROCHETS

EXEMPLE :

[AB[:digit:]x-z] représente le caractère UNIQUE A, B, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, x, y ou z

Les 6 CLASSES de caractères \s, \d, \l, \u, \w, et \h peuvent s'employer, INDIFFÉREMMENT,

DANS une expression entre CROCHETS ou en DEHORS de celle-ci

EXCEPTION :

\v représente un BLANC VERTICAL = [\n\x0B\f\r]

MAIS [\v] représente le SEUL caractère \x0B ( VERTICAL_TABULATION )

Les CLASSES débutant par \p sont utilisées UNIQUEMENT en DEHORS d'une expression entre CROCHETS

NOTES :

Les NOMS de CLASSE, dans les formes [:....:], \p{....} et \p., peuvent, INDIFFÉREMMENT, s'écrire en lettres MAJUSCULES ou en MINUSCULES

EXEMPLE : [[:diGIt:]] ou \p{DigiT} ou \p{D} ou \pD

H) Un caractère en DEHORS d'une CLASSE de CARACTÈRES :

IMPORTANT :

Une CLASSE de caractères, POSIX ou NON, correspond TOUJOURS à un SEUL caractère

Elle possède PLUSIEURS syntaxes : - [^[:<NOM de CLASSE:]] ou [^[:<NOM de CLASSE à 1 Car.>:]]

- \P{<NOM de CLASSE}  ou  \P{<NOM de CLASSE à 1 Car.>}
- \P<NOM de CLASSE à 1 Car.>
- \<NOM de CLASSE à 1 Car. MAJUSCULE>

| DANS Classe [^..] | HORS Classe de Car. [^..] | | | | CONTENU de la CLASSE de caractères | |
|---|---|---|---|---|---|---|
| [:space:] | [:s:] | \P{space} | \P{s} | \Ps | \S | [^\t\n\x0B\f\r\x20\xa0] | P |
| [:digit:] | [:d:] | \P{digit} | \P{d} | \Pd | \D | [^0-9¹²³] | Q |
| [:lower:] | [:l:] | \P{lower} | \P{l} | \Pl | \L | [^a-zƒšœžªµºßàáâãäåæçèéêëìíîïðñòóôõöøùúûüýþÿ] | R |
| [:upper:] | [:u:] | \P{upper} | \P{u} | \Pu | \U | [^A-ZŠŒŽŸÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖØÙÚÛÜÝÞ] | S |
| [:word:] | [:w:] | \P{word} | \P{w} | \Pw | \W | [^_\d\l\u] | T |
| [:blank:] | [:h:] | \P{blank} | \P{h} | \Ph | \H | [^\t\x20\xA0] | U |
| [:v:] | | \P{v} | \P{v} | \Pv | \V | [^\n\x0B\f\r] | V |
| [:alnum:] | | \P{alnum} | | | | [^\d\l\u] = [\D\L\U] | W |
| [:alpha:] | | \P{alpha} | | | | [^\l\u] = [\L\U] | X |
| [:cntrl:] | | \P{cntrl} | | | | [^\x00-\x1F\x7F\x81\x8D\x8F\x90\x9D] | Y |
| [:graph:] | | \P{graph} | | | | [\x00-\x20\x7F-\x81^\x8D\x8F\x90˜™\x9D\xA0] | Z |
| [:print:] | | \P{print} | | | | [\x00-\x08\x0E-\x1f\x7F-\x81^\x8D\x8F\x90˜™\x9D] | € |
| [:punct:] | | \P{punct} | | | | [^]\x21-\x2F:;<=>?@\\^_`{\|}~,„…†‡‰‹‘’“”•–—›\xA1-\xBF×÷[] | $ |
| [:xdigit:] | | \P{xdigit} | | | | [^0-9A-Fa-f] | £ |

| [:unicode:] | \P{unicode} } | | | [^\x{0101}-\x{FFFF}] | Ÿ |
|---|---|---|---|---|---|

Ligne P = 1 Caractère NON BLANC = TOUT, SAUF \t, \n, \x0B, \f, \r, \x20 et \xA0
Ligne Q = 1 Caractère NON CHIFFRE, dont EXPOSANT = TOUT, SAUF un CHIFFRE de 0 à 9 et un EXPOSANT ( ¹²³ )
Ligne R = 1 Caractère NON ALPHABÉTIQUE MINUSCULE, ACCENTUÉ ou NON
Ligne S = 1 Caractère NON ALPHABÉTIQUE MAJUSCULE, ACCENTUÉ ou NON
Ligne T = 1 Caractère NON caractère de MOT = TOUT, SAUF CHIFFRE, EXPOSANT, car. ALPHABÉTIQUE et TIRET BAS
Ligne U = 1 Caractère NON BLANC HORIZONTAL = TOUT, SAUF \t, \x20 et \xA0
Ligne V = 1 Caractère NON BLANC VERTICAL, = TOUT, SAUF \n, \x0B, \f et \r
Ligne W = 1 Caractère NON ALPHANUMÉRIQUE, = TOUT, SAUF CHIFFRE, EXPOSANT et caractère ALPHABÉTIQUE
Ligne X = 1 Caractère NON ALPHABÉTIQUE, = TOUT, SAUF caractère ALPHABÉTIQUE, ACCENTUÉ ou NON
Ligne Y = 1 Caractère NON de CONTRÔLE, VISUALISABLE = TOUT, SAUF caractère de CONTRÔLE ou car. NON VISUALISABLE
Ligne Z = 1 Caractère NON VISIBLE = caractère de CONTRÔLE ou NON AFFICHABLE, BLANC, et € ˆ ˜ ™
Ligne € = 1 Caractère NON AFFICHABLE = caractère de CONTRÔLE ou NON AFFICHABLE + les 4 car. € ˆ ˜ ™
Ligne $ = 1 Caractère NON de PONCTUATION, ni SYMBOLE, ni ACCENT, ni signe MATHÉMATIQUE, ni symbole MONÉTAIRE
Ligne £ = 1 Caractère NON HEXADÉCIMAL = TOUT, SAUF CHIFFRES, LETTRES de A à F et LETTRES de a à f
Ligne ¥ = 1 Caractère, SANS équivalent UNICODE, de code < 256 ou le caractère \x100 ( Bug )

IMPORTANT :

La forme NÉGATIVE d'une classe de type POSIX [:....:], employée SEULE, peut être, INDIFFÉREMMENT,

écrite sous la FORME [^[:.....:]] ou sous la FORME [[:^.....:]]

EXEMPLE : [^[:word:]] = [[:^word:] = [^[:w:]] = [[:^w:]] = \W = [^_\d\l\u]

Les classes de type POSIX [:....:] sont utilisées UNIQUEMENT dans une expression entre CROCHETS

EXEMPLE : [^AB[:digit:]x-z] représente un caractère AUTRE que :

A, B, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, x, y et z

Les 7 CLASSES de caractères \S, \D, \L, \U, \W, \H et \V peuvent s'employer, INDIFFÉREMMENT, DANS une expression entre CROCHETS ou en DEHORS de celle-ci

Les CLASSES débutant par \P sont utilisées UNIQUEMENT en DEHORS d'une expression entre CROCHETS

NOTES :

Les NOMS de CLASSE, dans les formes [:....:], \P{....} et \P., peuvent, INDIFFÉREMMENT, s'écrire en lettres MAJUSCULES ou en MINUSCULES

EXEMPLE : [[:diGIt:]] ou \P{DigiT} ou \P{D} ou \PD

I) Un caractère d'une LISTE de CARACTÈRES :

[<Liste>] = UN des caractères de <Liste>, si elle NE contient PAS un des 4 caractères ^ \ - ou ]

[<Car1>-<Car2>] = UN des caractères de l'ÉTENDUE entre les caractères <Car1> et <Car2> INCLUS

IMPORTANT : Le code ASCII de Car2 doit être SUPÉRIEUR ou ÉGAL au code ASCII de Car1

NOTES :

Entre DEUX CROCHETS, on peut combiner les DEUX formes ci-dessus :

[123A-C_|x-z] représente UN des 11 CARACTÈRES 1, 2, 3, A, B, C, _, |, x, y ou z

La forme [X-b] ou [XYZ\[\\]\^_`ab] cherche UN des 11 CARACTÈRES X, Y, Z, [, \, ], ^, _, `, a, b

Mais, si l'OPTION (?i) est placée AVANT [W-c] ou si la CASE " Respecter la casse " est DÉCOCHÉE,
l'expression devient INCORRECTE car elle autorise le CAS [w-C] qui génère une ERREUR

REMARQUES :

Les expressions [\d-z] et [0-9-z] sont INCORRECTES, MAIS [a-\d] équivaut bien à [a-d]

[abc] recherche UNE des 3 lettres MINUSCULES a, b ou c MAIS \[abc] recherche la CHAÎNE '[abc]'

Le SYMBOLE ']' de FIN de classe de caractère peut s'écrire LITTÉRALEMENT si :

- au TOUT DÉBUT de <Expression>
- indiqué sous la forme \]

EXEMPLES :

[X-]b] représente 3 CARACTÈRES : la lettre 'X' ou le symbole MOINS '-', SUIVI de la CHAÎNE 'B]'

[X-\]b] représente UN des 7 CARACTÈRES X, Y, Z, [, \, ] ou b

[]X-b] représente UN des 11 CARACTÈRES X, Y, Z, [, \, ], ^, _, `, a ou b

Le SYMBOLE '-' d'ÉTENDUE de caractères peut s'écrire LITTÉRALEMENT si :

- au TOUT DÉBUT ou en TOUTE FIN de <Expression>
- indiqué sous la forme \-

EXEMPLES :

[X-b] représente UN des 11 CARACTÈRES X, Y, Z, [, \, ], ^, _, `, a, ou b

[X\-b] ou [-Xb] ou [Xb-] représente UN des 3 CARACTÈRES -, X, ou b

Le SYMBOLE '\' d'ÉCHAPPEMENT de caractère peut s'écrire LITTÉRALEMENT, si indiqué sous la forme \\

EXEMPLES :

[X-c] représente UN des 12 CARACTÈRES X, Y, Z, [, \, ], ^, _, `, a, b ou c

[X-\c] est une expression INCORRECTE

[X-\\c] représente UN des 6 CARACTÈRES X, Y, Z, [, \ ou c

J) Un caractère en DEHORS d'une LISTE de CARACTÈRES :

[^<Liste>] = Un SEUL caractère, DIFFÉRENT de TOUS les caractères de <Liste>

[^<Car1>-<Car2>] = UN SEUL caractère, en DEHORS de l'ÉTENDUE des caractères <Car1> à <Car2> INCLUS

EXEMPLE :

[^A-D12] trouve TOUT caractère DIFFÉRENT de A,B,C,D,1 et 2, dont 'FIN de LIGNE' ( \r, \n ou \f) !

Le SYMBOLE '^' de NÉGATION de CLASSE de caractères peut s'écrire LITTÉRALEMENT si :

- PAS en PREMIER caractère de la LISTE entre CROCHETS
- indiqué sous la forme \^

EXEMPLES :

[^Z-a] représente TOUS les CARACTÈRES, AUTRES que Z, [, \, ], ^, _, ` et la lettre MINUSCULE a

[Z^-a] représente UN des 5 CARACTÈRES Z, ^, _, ` ou la lettre MINUSCULE a

[Z-^a] représente UN des 6 CARACTÈRES Z, [, \, ], ^ ou la lettre MINUSCULE a

[Z-a^] représente UN des 8 CARACTÈRES Z, [, \, ], ^, _, ` ou la lettre MINUSCULE a

IMPORTANT : Pour RESTREINDRE l'ÉTENDUE de caractères de la FORME \w, on peut aussi employer la

la SYNTAXE : [^\W<Caractères à EXCLURE>]

EXEMPLES :

[^\W_a-z] équivaut à [0-9A-Z]

[^\W_] équivaut à [0-9A-Za-z]

K) Un caractère DOUBLE ( Collating element ) :

Il forme une SEULE UNITÉ formée de DEUX caractères

EXEMPLES :

La FORME [[.ae.]] représente la chaîne des DEUX caractères 'ae'

La FORME [[.ch.]-g] représente UN des 5 CARACTÈRES 'ch', d, e, f ou g

IMPORTANT : Le moteur PRCE, de NOTEPAD++, reconnaît 7 caractères DOUBLES, dans 3 DIFFÉRENTES casses :

| | | | |
|---|---|---|---|
| [[.AE.]] | [[.Ae.]] | [[.ae.]] | |
| [[.CH.]] | [[.Ch.]] | [[.ch.]] | ( Caractère Espagnol ) |
| [[.DZ.]] | [[.Dz.]] | [[.dz.]] | |
| [[.LJ.]] | [[.Lj.]] | [[.lj.]] | |
| [[.LL.]] | [[.Ll.]] | [[.ll.]] | ( Caractère Espagnol ) |
| [[.NJ.]] | [[.Nj.]] | [[.nj.]] | |

[[.SS.]] , [[.Ss.]] , [[.ss.]] ( Caractère Allemand )

REMARQUE : La POSITION des caractères DOUBLES, dans l'ordre ALPHABÉTIQUE, se situe, SELON les exemples,

AVANT ou APRÈS leur lettre RACINE, MAJUSCULE ou MINUSCULE ?

EXEMPLES :

[[.dz.]-[.ll.]] représente UN des CARACTÈRES dz, e, f, g, h, i, j, k, l ou ll

[[.dz.]-[.lj.]] représente UN des CARACTÈRES dz, e, f, g, h, i, j, k, l ou lj

[[.dz.]-e] représente UN des CARACTÈRES dz ou e ( IDENTIQUE à [[.dz.]e] )

[[.dz.]-d] est une expression INCORRECTE

[[.dz.]d] représente UN des CARACTÈRES d ou dz

[d[.dz.]] représente le CARACTÈRE d UNIQUEMENT ( !! )

L) Un caractère avec un Nom SYMBOLIQUE POSIX :

DEUX syntaxes sont possibles : [[.<Nom SYMBOLIQUE>.]] ou \N{<Nom SYMBOLIQUE>}

EXEMPLES :

La FORME [[.IS2.]] représente le caractère RECORD SEPARATOR ( RS ), de code HEXA \x1E

La FORME \N{plus-sign} représente le signe ADDITION ( + ), de code HEXA \x2B

TRÈS IMPORTANT : La CASSE des noms SYMBOLIQUES doit OBLIGATOIREMENT être respectée !

REMARQUE : la forme \N{....} peut également s'employer ENTRE CROCHETS ( Voir CI-DESSUS )

LISTE des Noms SYMBOLIQUES possibles :

| Nom SYMBOLIQUE POSIX | CARACTÈRE | VALEUR Caractère |
|---|---|---|
| NUL | NUL | 000 | \x00 |
| SOH | SOH | 001 | \x01 |
| STX | STX | 002 | \x02 |
| ETX | ETX | 003 | \x03 |
| EOT | EOT | 004 | \x04 |
| ENQ | ENQ | 005 | \x05 |
| ACK | ACK | 006 | \x06 |
| alert | BEL | 007 | \x07 |
| backspace | BS | 008 | \x08 |
| tab | TAB | 009 | \x09 |
| newline | LF | 010 | \x0A |
| vertical-tab | VT | 011 | \x0B |
| form-feed | FF | 012 | \x0C |
| carriage-return | CR | 013 | \x0D |
| SO | SO | 014 | \x0E |
| SI | SI | 015 | \x0F |
| DLE | DLE | 016 | \x10 |
| DC1 | DC1 | 017 | \x11 |
| DC2 | DC2 | 018 | \x12 |
| DC3 | DC3 | 019 | \x13 |
| DC4 | DC4 | 020 | \x14 |
| NAK | NAK | 021 | \x15 |

| | | | |
|---|---|---|---|
| SYN | SYN | 022 | \x16 |
| ETB | ETB | 023 | \x17 |
| CAN | CAN | 024 | \x18 |
| EM | EM | 025 | \x19 |
| SUB | SUB | 026 | \x1A |
| ESC | ESC | 027 | \x1B |
| IS4 | FS | 028 | \x1C |
| IS3 | GS | 029 | \x1D |
| IS2 | RS | 030 | \x1E |
| IS1 | US | 031 | \x1F |
| space | SP | 032 | \x20 |
| exclamation-mark | ! | 033 | \x21 |
| quotation-mark | " | 034 | \x22 |
| number-sign | # | 035 | \x23 |
| dollar-sign | $ | 036 | \x24 |
| percent-sign | % | 037 | \x25 |
| ampersand | & | 038 | \x26 |
| apostrophe | ' | 039 | \x27 |
| left-parenthesis | ( | 040 | \x28 |
| right-parenthesis | ) | 041 | \x29 |
| asterisk | * | 042 | \x2A |
| plus-sign | + | 043 | \x2B |
| comma | , | 044 | \x2C |
| hyphen | - | 045 | \x2D |
| period | . | 046 | \x2E |
| slash | / | 047 | \x2F |
| zero | 0 | 048 | \x30 |
| one | 1 | 049 | \x31 |
| two | 2 | 050 | \x32 |
| three | 3 | 051 | \x33 |
| four | 4 | 052 | \x34 |
| five | 5 | 053 | \x35 |
| six | 6 | 054 | \x36 |
| seven | 7 | 055 | \x37 |
| eight | 8 | 056 | \x38 |
| nine | 9 | 057 | \x39 |
| colon | : | 058 | \x3A |
| semicolon | ; | 059 | \x3B |
| less-than-sign | < | 060 | \x3C |

| | | | |
|---|---|---|---|
| equals-sign | = | 061 | \x3D |
| greater-than-sign | > | 062 | \x3E |
| question-mark | ? | 063 | \x3F |
| commercial-at | @ | 064 | \x40 |
| left-square-bracket | [ | 091 | \x5B |
| backslash | \ | 092 | \x5C |
| right-square-bracket | ] | 093 | \x5D |
| circumflex | ^ | 094 | \x5E |
| underscore | _ | 095 | \x5F |
| grave-accent | ` | 096 | \x60 |
| left-curly-bracket | { | 123 | \x7B |
| vertical-line | \| | 124 | \x7C |
| right-curly-bracket | } | 125 | \x7D |
| tilde | ~ | 126 | \x7E |
| DEL | DEL | 127 | \x7F |

NOTE : <NOM SYMBOLIQUE> peut, également, représenter le caractère LUI-MÊME !

EXEMPLES :

[[.].]] représente le caractère CROCHET FERMANT ( ]) ( Préférer les FORMES ] ou \] )

\N{{} représente le caractère ACCOLADE OUVRANTE ({) ( Préférer les FORMES { ou \{ )

M) Un caractère de CLASSE ÉQUIVALENTE :

La FORME [=<Car>=] donne TOUS les ÉQUIVALENTS du CARACTÈRE <Car>, QUELLES QUE SOIENT la CASSE,

l'ACCENTUATION, la TAILLE et autres SPÉCIFICITÉS des caractères ÉQUIVALENTS

NOTE : <Car> peut donc, aussi, être un caractère DOUBLE ou un NOM SYMBOLIQUE du tableau PRÉCÉDENT

EXEMPLES : ( Pour un fichier ANSI, avec CARACTÈRES de code ASCII < 256 )

[[=a=]] représente UN des 15 CARACTÈRES A, a, ª, À, Á, Â, Ã, Ä, Å, à, á, â, ã, ä ou å

[[=two=]] représente le CARACTÈRE 2 ou le CARACTÈRE EXPOSANT ²

[[=Ae=]] représente la LIGATURE Æ ou la LIGATURE æ

[[=SS=]] représente le CARACTÈRE ALLEMAND ß

[[===]] représente le CARACTÈRE = ( Cas d'un caractère NON LITTÉRAL, ni SYMBOLIQUE ni DOUBLE )

Pour les Noms SYMBOLIQUES des caractères de CONTRÔLE, on a :

```
[[=tab=]]              =  Caractère TABULATION,      de code ASCII  \x09
[[=newline=]]          =  Caractère LINE FEED,       de code ASCII  \x0A
[[=vertical-tab=]]     =  Caractère VERTICAL TAB,    de code ASCII  \x0B
[[=form-feed=]]        =  Caractère FORM FEED,       de code ASCII  \x0C
[[=carriage-return=]]  =  Caractère CARRIAGE RETURN, de code ASCII  \x0D
```

TOUS les AUTRES Noms SYMBOLIQUES des codes de CONTRÔLE ( [[=NUL=]], [[=SI=]], [[=ESC=]], ... ) représentent UN des CARACTÈRES de la LISTE [\x00-\x08\x0E-\x1F'\-\x7F\x81\x8D\x8F\x90\x96\x97\x9D\xAD]

N) REMARQUE IMPORTANTE sur les CLASSES ou LISTES de forme NÉGATIVE :

Quand la forme NÉGATIVE d'une CLASSE ou d'une LISTE de CARACTÈRES ne fait PAS, directement ou

implicitement, RÉFÉRENCE à un CARACTÈRE 'FIN de LIGNE', celui-ci est alors AUTOMATIQUEMENT INCLUS dans la CLASSE ou LISTE cherchée

EXEMPLES :

[^\d]+ signifie une SUITE, NON VIDE, de caractères NON CHIFFRES ni EXPOSANT, c'est à dire DIFFÉRENT de la LISTE [0123456789¹²³] . Aussi, l'expression RÉGULIÈRE [^\d]+ trouvera, entre AUTRES :

- un caractère ALPHABÉTIQUE
- un caractère de PONCTUATION
- un caractère de code ASCII > 127
- un caractère de CONTRÔLE de code ASCII < 32, dont LF ( \x0A) et/ou CR (\x0D)
- .....

DONC, si le fichier examiné NE COMPORTE AUCUN chiffre, l'expression RÉGULIÈRE [^\d]+ sélectionnera ABSOLUMENT TOUS les CARACTÈRES du fichier ( équivalant à CTRL + A ), puisqu'elle INCLUT les 'FINS de LIGNE' !!!

Il en serait de même pour les SEPT expressions RÉGULIÈRES ci-dessous :

[^[:digit:]]+ , [^[:d:]]+ , [^0-9¹²³]+ , \P{digit}+ , \P{d}+ , \Pd+ et \D+

( Voir parties G et H, ci-dessus )

[^[:cntrl:]]+ ou \P{cntrl}+, par CONTRE, correspondent BIEN à l'INTÉGRALITÉ de CHAQUE ligne d'un FICHIER, si celui-ci ne contient AUCUN caractère de CONTRÔLE, AUTRES que les DEUX caractères LF (\x0A) et/ou CR (\x0D), selon le TYPE du fichier ( Windows, Unix ou Mac )

([^@\r\n]*@[^@\r\n]*@)+[^@\r\n]* trouve la PLUS GRANDE SUITE de caractères CONTENANT un nombre PAIR, NON NUL, de caractères '@' ( AROBASE )

CONCLUSION :

BIEN vérifier TOUS les caractères POSSIBLES, satisfaisant la forme NÉGATIVE d'une CLASSE ou d'une LISTE de CARACTÈRES, AVANT de l'ENGLOBER dans une expression RÉGULIÈRE plus IMPORTANTE !

0) Exemple RÉCAPITULATIF :

L'Expression RÉGULIÈRE Mr[\x21\033\a\c~[:Upper:]\h[.Dz.]\N{slash}[=d=]p-t\\-]\RGuy[^.[:u:]^] THEVENOT\+

trouve la CHAÎNE 'Mr', suivie d'UNE Lettre MAJUSCULE, MÊME ACCENTUÉE, OU de l'UN des 20 CARACTÈRES :

!, ESC, BEL, RS, SP, TAB, \xA0, Dz, /, D, d, Ð, ð, p, q, r, s, t, \, -

suivi d'UN des 6 CARACTÈRES de TYPE 'Fin de LIGNE' \r\n|[\n\x0B\f\r\x85], suivi de la CHAÎNE 'Guy'

suivie d'UN caractère, AUTRE que . et ^ et 1 Lettre MAJUSCULE, MÊME ACCENTUÉE

suivi de la CHAÎNE ' THEVENOT+'

Cette expression RÉGULIÈRE trouve, par exemple, le TEXTE : MrðFFGuy@ THEVENOT+

2) SÉQUENCES d'ÉCHAPPEMENT :

Les SÉQUENCES d'ÉCHAPPEMENT sont des SUITES, NON NULLES, de caractères UNIQUEMENT LITTÉRAUX

Elles DÉBUTENT par le caractère d'ÉCHAPPEMENT \Q et se TERMINENT par le caractère d'ÉCHAPPEMENT \E

EXEMPLES :

\Q+*?$\E cherche la CHAÎNE de 4 caractères '+*?$'

\Q*+\Ea+ cherche la CHAÎNE '*+', suivie de 1 ou PLUSIEURS lettres a MINUSCULE

( Voir PLUS LOIN, la description des QUANTIFICATEURS )

NOTE :

Si le caractère d'ÉCHAPPEMENT \E est ABSENT, TOUS les caractères placés, APRÈS le caractère d'ÉCHAPPEMENT \Q JUSQU'à la FIN de l'expression RÉGULIÈRE, sont considérés LITTÉRAUX

EXEMPLES :

\Q[]{}()<> trouve la CHAÎNE de 8 caractères '[]{}()<>'

\Q*+a+ trouve la CHAÎNE de 5 caractères '*+a+'

IMPORTANT :

- Les SÉQUENCES d'ÉCHAPPEMENT sont INOPÉRANTES dans les LISTES de caractères entre CROCHETS

Par exemple :

l'expression RÉGULIÈRE [0-9\Q[-]\Ea-z]

NE signifie PAS :

1 CHIFFRE ou un CROCHET ([]) ou un TIRET (-) ou une lettre MINUSCULE

mais SIGNIFIE :

1 CHIFFRE ou la LETTRE MAJUSCULE Q ou un CROCHET OUVRANT ([) ou un TIRET (-),
suivi de la CHAÎNE 'Ea-z]' !

l'expression RÉGULIÈRE [0-9\Qa-z\E@#] cherche un CHIFFRE ou une lettre MINUSCULE ou
UN des QUATRE caractères 'Q' , 'E' , '@' ou '#'

- L'expression RÉGULIÈRE \Q+*?$\E{2,4}a+ équivaut, en fait, à l'expression \Q+*?\E${2,4}a+, soit :

les 3 CARACTÈRES '+*?', suivie de 2 à 4 CARACTÈRES '$', suivie de 1 ou PLUSIEURS 'a' MINUSCULE

NOTE : Écrite sous cette FORME, \Q+*?\E${2,4}a+ => MESSAGE 'Invalid regular expression'

MAIS \Q+*?${2,4}a+ ( Caractère d'ÉCHAPPEMENT \E ABSENT ) représente la chaîne LITTÉRALE

des 11 caractères '+*?${2,4}a+'

- L'expression RÉGULIÈRE (\Q+*?$\E){2,4} représente de 2 à 4 FOIS la CHAÎNE '+*?$'

- La RegExp abc\Q\Edef signifie, en fait, la CHAÎNE 'abcdef' car \Q\E représente une chaîne VIDE

3) ALTERNATIONS :

L'opérateur d'ALTERNATION est le symbole LIGNE VERTICALE (|) qui joue le rôle de SÉPARATEUR entre 2 GABARITS SIMULTANÉMENT recherchés

NOTE :

Cet OPÉRATEUR a la PLUS FAIBLE priorité d'EXÉCUTION ( Voir PLUS LOIN )

Aussi, le REGROUPEMENT, par un COUPLE de PARENTHÈSES (), est parfois NÉCESSAIRE afin de

BIEN IDENTIFIER les différentes ALTERNATIVES

EXEMPLES :

| | |
|---|---|
| abc\|def | cherche les CHAÎNES 'abc' ou 'def' |
| ab(c\|d)ef | cherche les CHAÎNES 'abcef' ou 'abdef' |
| ^abc\|def$ | cherche la CHAÎNE 'abc' en DÉBUT de ligne ou la CHAÎNE 'def' en FIN de ligne |
| ^(abc\|def)$ | cherche les LIGNES, constituées des SEULES chaînes 'abc' ou 'def' |
| abc(def\|xyz) | cherche les CHAÎNES 'abcdef' ou 'abcxyz' |
| abc(def\|xyz\|) | cherche les CHAÎNES 'abcdef' ou 'abcxyz' ou 'abc' |
| chat(ière\|on\|) : | cherche les CHAÎNES 'chatière :' ou 'chaton :' ou 'chat :' |
| chatière\|on\| : | cherche les CHAÎNES 'chatière' ou 'on' ou ' :' |

IMPORTANT :

- Quand un jeu d'ALTERNATIVES est ENTOURÉ de PARENTHÈSES, l'ensemble forme un GROUPE de CAPTURE :

  - qui peut être SUIVI de QUANTIFICATEURS
  - qui permet l'utilisation d'une RÉFÉRENCE ARRIÈRE ou d'une RÉFÉRENCE de GROUPE, relative à ce GROUPE de CAPTURE ( Voir ces notions, PLUS LOIN )

(a|b)c\1  cherche UNE des DEUX chaînes 'aca' ou 'bcb'

(a|b){2,3} cherche UNE des 12 chaînes :

'aa' , 'ab' , 'ba' , 'bb'

'aaa' , 'aab' , 'aba' , 'abb' , 'baa' , 'bab' , 'bba' , 'bbb'

- La PREMIÈRE alternative d'une LISTE, qui est TROUVÉE dans la chaîne SUJET, est UTILISÉE, et ce MÊME SI 1 ou PLUSIEURS AUTRES alternatives, placées APRÈS, pouvaient AUSSI convenir

=> l'ORDRE des différentes ALTERNATIVES peut donc MODIFIER la RECHERCHE

Soit la chaîne SUJET 'Elle a fait trois longueurs de bassin. c'est trop long !'

Si l'expression RÉGULIÈRE vaut :

- long|longueur => SEULS les 4 CARACTÈRES 'long' sont trouvés, et donc la CHAÎNE 'longueur' NE pourra PAS être trouvée avec cette RegExp

- longueur|long => les CHAÎNES 'longueur' ou 'long' seront trouvées

La RegExp (a|ab)b trouve la CHAÎNE 'ab' dans la chaîne SUJET 'abbcd'

et (ab|a)b trouve la CHAÎNE 'abb' dans la chaîne SUJET 'abbcd'

- Le symbole d'ALTERNATION est considéré comme un caractère LITTÉRAL dans une liste ENTRE CROCHETS :

=> L'expression RÉGULIÈRE [les|la|le] peut, PLUS simplement, s'écrire [aels|]

REMARQUE :

La recherche des VOYELLES MINUSCULES , par exemple, est PLUS efficace et le GABARIT PLUS lisible sous la forme d'une CLASSE de caractères [aeiouy] qu'avec le JEU d'ALTERNATIVES (a|e|i|o|u|y)

4) ASSERTIONS :

Les ASSERTIONS sont des expressions RÉGULIÈRES, de longueur NULLE, ne consommant AUCUN caractère dans la chaîne SUJET à analyser, qui représentent, NON PAS un CARACTÈRE, mais une POSITION, dans la chaîne SUJET, SATISFAISANT une CONDITION particulière, AVANT ou APRÈS la position COURANTE du POINTEUR de RECHERCHE

DEUX types d'ASSERTIONS sont possibles :

- Les assertions SIMPLES, aussi appelées ANCRES, représentent des positions STANDARD
- les assertions COMPLEXES, aussi appelées LOOKAROUND, représentent des positions UTILISATEUR

( qui seront étudiées PLUS LOIN )

BUG : Dans NOTEPAD++, les ASSERTIONS ARRIÈRE, ( Voir PLUS LOIN ) c'est à dire :

- les ANCRES placées AVANT l'expression RÉGULIÈRE cherchée ( SAUF ^ )
- les LOOKAROUNDS, placés AVANT l'expression RÉGULIÈRE cherchée ( LOOKBEHINDS )

consomment, à TORT, UN ou PLUSIEURS caractères, déplaçant, aussi, à TORT, la position COURANTE du POINTEUR de recherche

CONSÉQUENCE : Si la RegExp de recherche, qui suit l'assertion ARRIÈRE, est un SEUL caractère, QUELCONQUE, du genre . ou \w , le RÉSULTAT de la recherche est, en général, FAUX

EXEMPLES :

\b. , \B. , \A\w ne donnent PAS le résultat ESCOMPTÉ !

MAIS .\b , .\B , .\z , .\Z fonctionnent CORRECTEMENT

.(?=.) trouve bien TOUS les caractères, DIFFÉRENTS de \n, \f et \r, SUIVIS d'un caractère DIFFÉRENT de \n, \f ou \r, c'est à dire TOUS les caractères du fichier COURANT, SAUF le DERNIER de CHAQUE ligne

(?<=.). DEVRAIT trouver TOUS les caractères DIFFÉRENTS de \n, \f et \r, PRÉCÉDÉS d'un caractère DIFFÉRENT de \n, \f ou \r, c'est à dire

TOUS les caractères du fichier COURANT, SAUF le PREMIER de CHAQUE ligne

MAIS, en fait, n'en trouve effectivement qu'UN sur DEUX, environ !

RAPPEL :

Un caractère de MOT ( \w ) correspond à un (au) CARACTÈRE :

- TIRET BAS : [_]
- CHIFFRE ou ASSIMILÉ : [0-9¹²³]
- ALPHABÉTIQUE MINUSCULE, ACCENTUÉ ou NON ou ASSIMILÉ : [a-zƒšœžªµºßàáâãäåæçèéêëìíîïðñòóôõöøùúûüýþÿ]
- ALPHABÉTIQUE MAJUSCULE, ACCENTUÉ ou NON ou ASSIMILÉ : [A-ZŠŒŽŸÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖØÙÚÛÜÝÞ]

Une ASSERTION SIMPLE, appelée ANCRE, peut être, au CHOIX, la POSITION :

A) TOUT DÉBUT du fichier COURANT : \A ou \` ou (?-m)^

EXEMPLE : \AN\+\+ cherche la CHAÎNE 'N++' au TOUT DÉBUT du fichier COURANT

ATTENTION :

Si la PREMIÈRE ligne du fichier n'est PAS VIDE ( Voir Problème ci-DESSUS ) :

\A\w trouve, à TORT, le 1ER MOT du fichier , au lieu de la 1ÈRE LETTRE, UNIQUEMENT

\A. trouve, à TORT, la 1ÈRE LIGNE du fichier, au lieu du 1ER CARACTÈRE, UNIQUEMENT

Si la PREMIÈRE ligne du fichier commence par des CHIFFRES ( Voir Problème ci-DESSUS ) :

(?-m)^\d trouve, à TORT, ce NOMBRE, au lieu du 1ER CHIFFRE, au TOUT DÉBUT du fichier

NOTE : La FORME (?-m)^, IDENTIQUE à \A, sera expliquée par la SUITE

L'OPTION (?-m) signifie que l'ASSERTION ^ ne concerne que le TOUT DÉBUT de fichier

B) TOUTE FIN du fichier COURANT : \z ou \' ou (?-m)$

EXEMPLES :

OK\.\z cherche la CHAÎNE 'OK.' en TOUTE FIN de fichier

.\z cherche le DERNIER caractère du fichier

=> la DERNIÈRE ligne NE DOIT PAS se terminer par un caractère de 'FIN de LIGNE'

OK\.\r\n\z' cherche la CHAÎNE 'OK.\r\n' en TOUTE FIN de fichier

=> la DERNIÈRE ligne DOIT se terminer par un caractère de 'FIN de LIGNE' WINDOWS ( \r\n )

NOTE : La FORME (?-m)$, IDENTIQUE à \z, sera expliquée par la SUITE

L'option (?-m) signifie que l'ASSERTION $ ne concerne que la TOUTE FIN de fichier

C) FIN de la DERNIÈRE LIGNE du fichier COURANT : \Z

EXEMPLES :

OK\.\Z cherche la CHAÎNE 'OK.' en FIN de la DERNIÈRE ligne du fichier

.\Z cherche le DERNIER caractère de la DERNIÈRE ligne du fichier

REMARQUE : En fait, la chaîne SUJET, en FIN de la DERNIÈRE ligne, peut être SUIVIE, ou NON, par une

SUITE QUELCONQUE de caractères \r , \n ou \f

=> \Z équivaut à l'ASSERTION AVANT, appelée LOOKAHEAD (?=[\n\f\r]*\z)

( Voir PLUS LOIN )

D) TOUT DÉBUT de LIGNE : ^

EXEMPLES :

^N\+\+ cherche la CHAÎNE 'N++' en TOUT DÉBUT des LIGNES du fichier COURANT

^. cherche le PREMIER caractère de CHAQUE ligne du fichier COURANT

REMARQUES :

En fait l'ASSERTION ^ représente la POSITION :

- ENTRE le TOUT DÉBUT du fichier COURANT et le PREMIER caractère du fichier
- ENTRE un caractère 'FIN de LIGNE' et le PREMIER caractère de la ligne SUIVANTE

=> ^ équivaut à l'ALTERNATIVE ENTRE les 2 ASSERTIONS ARRIÈRE (\A|(?<=[\n\f\r]))

qui NE fonctionne PAS car \A NON PLUS ! ( Voir PROBLÈME ci-dessus )

Si l'OPTION (?-m), expliquée par la SUITE, est placée DEVANT l'expression RÉGULIÈRE, l'ASSERTION ^ est ÉQUIVALENTE à l'ASSERTION \A et représente la POSITION ENTRE le TOUT DÉBUT du fichier et le PREMIER caractère du fichier

E) TOUTE FIN de LIGNE : $

EXEMPLES

OK\.$ cherche la CHAÎNE 'OK.' en TOUTE FIN des LIGNES du fichier COURANT

.$ cherche le DERNIER caractère de CHAQUE ligne du fichier COURANT

REMARQUES :

En fait l'ASSERTION $ représente la POSITION :

- ENTRE le DERNIER caractère d'une LIGNE et un caractère 'FIN de LIGNE'
- ENTRE le DERNIER caractère du fichier et la TOUTE FIN du fichier COURANT

=> $ équivaut à l'ALTERNATIVE ENTRE les 2 ASSERTIONS AVANT ((?=[\n\f\r])|\z)

Si l'OPTION (?-m), expliquée par la SUITE, est placée DEVANT l'expression RÉGULIÈRE, l'ASSERTION $ est ÉQUIVALENTE à l'ASSERTION \z et représente la POSITION ENTRE le DERNIER caractère du fichier et la TOUTE FIN du fichier

F) FRONTIÈRE de MOT : \b

Elle représente la POSITION :

- ENTRE le TOUT DÉBUT du fichier OU 1 caractère NON de MOT (\A|\W) et 1 caractère de MOT (\w)

OU

- ENTRE 1 caractère de MOT (\w) et 1 caractère NON de MOT OU la TOUTE FIN de fichier (\W|\z)

RAPPEL : \n, \f ou \r représente ÉGALEMENT des caractères NON de MOT

Cette ASSERTION équivaut donc à l'ALTERNATIVE :

- (\A|(?<=\W)) si le caractère SUIVANT est un caractère de MOT (\w)

ou

- ((?=\W)|\z) si le caractère PRÉCÉDENT est un caractère de MOT (\w)

ou

- (?<=\w) si le caractère SUIVANT est un caractère NON de MOT (\W)

ou

- (?=\w) si le caractère PRÉCÉDENT est un caractère NON de MOT (\W)

=> \b\w = (\A|(?<=\W))\w et \w\b = \w((?=\W)|\z)

et \b\W = (?<=\w)\W et \W\b = \W(?=\w)

BUG : les FORMES \b\w et (\A|(?<=\W))\w NE donnent PAS le RÉSULTAT escompté ( Voir PROBLÈME PLUS HAUT et suite DYSFONCTIONNEMENT de \A )

=> Employer UNIQUEMENT la FORME (^|(?<=\W))\w

EXEMPLES :

(^|(?<=\W))\w|\w((?=\W)|\z) ou (^|(?<=\W))\w|\w\b trouve le 1ER et DERNIER car. de TOUS les MOTS

(?<=\w)\W|\W(?=\w) ou \b\W|\W\b trouve le 1ER et DERNIER caractère de TOUS les NON MOTS

\be ou (^|(?<=\W))e cherche TOUS les caractères 'e' MINUSCULES, en DÉBUT de MOT

e\b ou e((?=\W)|\z) cherche TOUS les caractères 'e' MINUSCULES, en FIN de MOT

\be|e\b ou (^|(?<=\W))e|e((?=\W)|\z) cherche TOUS les 'e' MINUSCULES en DÉBUT ou FIN de MOT

(^|(?<=\W))\w cherche TOUS les caractères de MOT, en DÉBUT de MOT

\w((?=\W)|\z) ou \w\b cherche TOUS les caractères de MOT, en FIN de MOT

La FORME .\b cherche :

- un caractère de MOT, SUIVI d'un caractère NON de MOT, y COMPRIS \n, \f ou \r

ou

- un caractère NON de MOT, DIFFÉRENT de [\n\f\r], SUIVI d'un caractère de MOT

BUG : la FORME \b. DEVRAIT trouver :

- un caractère de MOT, PRÉCÉDÉ d'un caractère NON de MOT, y COMPRIS \n, \f ou \r

ou

- un caractère NON de MOT, DIFFÉRENT de [\n\f\r], PRÉCÉDÉ d'un caractère de MOT

et trouve, à TORT, TOUS les caractères DIFFÉRENTS de [\n\f\n] ! ( Voir ci-DESSUS )

G) FRONTIÈRE de NON MOT : \B

Elle représente la POSITION :

- ENTRE 1 caractère NON de MOT (\W) et 1 AUTRE caractère NON de MOT (\W), y COMPRIS \n, \f ou \r

OU

- ENTRE 1 caractère de MOT (\w) et 1 AUTRE caractère de MOT (\w)

Cette ASSERTION équivaut donc à l'ALTERNATIVE :

- (?<=\w) si le caractère SUIVANT est un caractère de MOT (\w)

ou

- (?=\w) si le caractère PRÉCÉDENT est un caractère de MOT (\w)

ou

- (?<=\W) si le caractère SUIVANT est un caractère NON de MOT (\W), y compris \n, \f ou \r

ou

- (?=\W) si le caractère PRÉCÉDENT est un caractère NON de MOT (\W), y compris \n, \f ou \r

=> \B\w = (?<=\w)\w et \w\B = \w(?=\w)

et \B\W = (?<=\W)\W et \W\B = \W(?=\W)

BUGS : les FORMES \B\w ou (?<=\w)\w, d'une part et \B\W ou (?<=\W)\W, d'autre part, NE donnent PAS

le RÉSULTAT escompté ( Voir PROBLÈME PLUS HAUT, et suite RÉPÉTITION \w\w ou \W\W )

EXEMPLES :

\Be ou (?<=\w)e cherche TOUS les caractères 'e' MINUSCULES, PRÉCÉDÉS d'un caractère de MOT

e\B ou e(?=\w) cherche TOUS les caractères 'e' MINUSCULES, SUIVIS d'un caractère de MOT

\Be|e\B ou (?<=\w)e|e(?=\w) cherche TOUS les 'e' MINUSCULES, PRÉCÉDÉS ou SUIVIS d'un car. de MOT

(?<=\w)\w ou \B\w DEVRAIT trouver TOUS les car. de MOT, PRÉCÉDÉS d'un caractère de MOT ( BUG ! )

\w(?=\w) ou \w\B cherche TOUS les caractères de MOT, SUIVIS d'un caractère de MOT

La FORME .\B cherche :

- un caractère de MOT, SUIVI d'un AUTRE caractère de MOT

ou

- un caractère NON de MOT, DIFFÉRENT de [\n\f\r], SUIVI d'un AUTRE caractère NON de MOT,

y COMPRIS \n, \f ou \r

BUG : la FORME \B. DEVRAIT trouver :

- un caractère de MOT, PRÉCÉDÉ d'un AUTRE caractère de MOT

ou

- un caractère NON de MOT, DIFFÉRENT de [\n\f\r], PRÉCÉDÉ d'un AUTRE caractère NON de MOT,

y COMPRIS \n, \f ou \r

et trouve, à TORT, TOUS ces caractères, UNE fois sur DEUX ! ( Voir problème PLUS HAUT, et suite RÉPÉTITION \w\w ou \W\W )

H) DÉBUT de MOT : \<

Elle représente la POSITION :

ENTRE le TOUT DÉBUT du fichier OU 1 caractère NON de MOT (\A|\W) et 1 caractère de MOT (\w)

EXEMPLE : \<e cherche TOUS les CARACTÈRES 'e' MINUSCULES, figurant en DÉBUT de MOT

BUG : Suite au PROBLÈME, concernant les ASSERTIONS ARRIÈRE dans NOTEPAD++, évoqué PLUS HAUT, la RegExp '\<e', appliquée à la chaîne SUJET 'eeeeeeeeabc' sélectionne, à TORT, 'eeeeeeee'

De même, la RegExp \<. sélectionne, à TORT, TOUS les caractères de MOT du fichier COURANT !

=> Employer UNIQUEMENT l'ASSERTION composée (^|(?<=\W)), à la PLACE de l'ASSERTION \<

EXEMPLE : Appliquée à la chaîne SUJET 'eeeeeeeeabc', l'expression RÉGULIÈRE (^|(?<=\W))e trouve, CORRECTEMENT, QUE le PREMIER caractère 'e' MINUSCULE de la chaîne SUJET 'eeeeeeeeabc'

I) FIN de MOT : \>

Elle représente la POSITION :

ENTRE 1 caractère de MOT (\w) et 1 caractère NON de MOT OU la TOUTE FIN de fichier (\W|\z)

EXEMPLE : e\> cherche TOUS les CARACTÈRES 'e' MINUSCULES, figurant en FIN de MOT

On peut, cependant, employer les ASSERTIONS ÉQUIVALENTES, ci-DESSOUS :

((?=\W)|\z) ou ((?=\W)|\') ou ((?=\W)|\Z) ou ((?=\W)|$)

EXEMPLES :

Les expressions RÉGULIÈRES, ci-dessous, trouvent, TOUTES, le DERNIER caractère de CHAQUE MOT du fichier COURANT :

\w((?=\W)|\z) , \w((?=\W)|\') , \w((?=\W)|\Z) , \w((?=\W)|$) , \w\> ou .\>

J) FIN de la PRÉCÉDENTE recherche : \G

Elle représente UNE des POSITIONS suivantes :

- le TOUT DÉBUT du fichier COURANT
- la position JUSTE APRÈS la FIN de la PRÉCÉDENTE recherche
- la position du CURSEUR, déplacé VOLONTAIREMENT par l'UTILISATEUR

EXEMPLE :

Soit une partie de code ADN 'TGAATTCTTGAATTCAAATGAAGGTTCTGACGTCATGATGAC', en 1ÈRE ligne d'un fichier

Dans cette SÉQUENCE, la suite des 3 BASES 'TGA' :

- est un CODON, aux positions SOULIGNÉES, ci-dessus ( Positions 1, 19 et 28 )
- n'est PAS un CODON, aux positions en POINTILLÉS, ci-dessus ( Positions 9, 36 et 39 )

Alors, l'expression RÉGULIÈRE trouve, à COMPTER de la position COURANTE de recherche :

| | | |
|---|---|---|
| la PLUS LONGUE suite de BASES, finissant par 'TGA' | si | .*TGA |
| la PLUS LONGUE suite de BASES, MULTIPLE de 3 car., finissant par 'TGA' | si | (\w\w\w)*TGA |
| la PLUS LONGUE suite de CODONS, finissant par le CODON 'TGA' | si | \G(\w\w\w)*TGA |
| la PLUS PETITE suite de BASES, finissant par 'TGA' | si | .*?TGA |

la PLUS PETITE suite de BASES, MULTIPLE de 3 car., finissant par 'TGA' si (\w\w\w)*?TGA

la PLUS PETITE suite de CODONS, finissant par le CODON 'TGA' si \G(\w\w\w)*?TGA

NOTE : Le QUANTIFICATEUR *?, de type 'LAZY', sera expliqué par la SUITE

5) COMMENTAIRES :

La FORME (?#.....), utilisée en DEHORS d'une CLASSE de CARACTÈRES, représente un COMMENTAIRE de l'UTILISATEUR

TOUS les caractères placés ENTRE la chaîne '(?#' et la 1ERE PARENTHÈSE FERMANTE qui suit, sont IGNORÉS

par le MOTEUR de recherche

EXEMPLES :

T+(?# T MAJUSCULE de 1 à N fois)CA cherche, par exemple, la CHAÎNE 'TCA' ou 'TTTTTCA'

T+CA(?# T MAJUSCULE de 1 à N fois, suivi de CA ) cherche, par exemple, la CHAÎNE 'TCA' ou 'TTTTTCA'

IMPORTANT :

Si l'OPTION (?x) ( Voir PLUS LOIN ) est placée en TÊTE de l'expression RÉGULIÈRE :

- les caractères ESPACE ( \x20 ) sont IGNORÉS du MOTEUR de recherche

- le PREMIER caractère DIÈSE ( # ) trouvé, en DEHORS d'une CLASSE de CARACTÈRES, et NON IMMÉDIATEMENT PRÉCÉDÉ d'un caractère BACKSLASH (\), DÉBUTE, également, une ZONE de COMMENTAIRE, et ce jusqu'à la FIN du GABARIT de recherche

EXEMPLES : L'expression RÉGULIÈRE 'T+CA' peut s'écrire :

```
(?x) T+      (?# T MAJUSCULE de 1 à N fois) CA
(?x) T+ CA   (?# T MAJUSCULE de 1 à N fois suivi de CA)
(?x) T+ C A   #   T MAJUSCULE de 1 à N fois, suivi de CA
```

REMARQUE : L'expression RÉGULIÈRE ac(?#Petit Essai){2,3}b signifie, en fait, ac{2,3}(?#Petit Essai)b, soit UNE des DEUX chaînes 'accb' ou 'acccb'

6) GROUPES et RÉFÉRENCES :

Les GROUPES représentent des expressions RÉGULIÈRES, MÉMORISÉES par le MOTEUR de RECHERCHE, pouvant être RÉPÉTÉES et, ÉVENTUELLEMENT, RÉUTILISÉES dans le GABARIT de RECHERCHE

A) Groupes de CAPTURE NON NOMMÉS :

La FORME (.....) représente un GROUPE de CAPTURE, NON NOMMÉ :

- qui peut être RÉPÉTÉ, s'il est SUIVI de QUANTIFICATEURS ( Voir PLUS LOIN )
- qui constitue un GROUPE de CARACTÈRES, dont la VALEUR ou le GABARIT sont RÉUTILISABLES dans l'expression RÉGULIÈRE de RECHERCHE

Un groupe de CAPTURE NON NOMMÉ est IDENTIFIÉ par un NUMÉRO N, avec 0 < N <= 65535, correspondant au NUMÉRO de la Nème PARENTHÈSE OUVRANTE, NON SUIVIE de la CHAÎNE '?:' ou de la CHAÎNE '?|'

EXEMPLES :

Soit la RegExp La (Dame) (Blanche) et la chaîne SUJET 'La Dame Blanche est perdue'

=> les groupes de CAPTURE 'Dame' et 'Blanche' sont RESPECTIVEMENT numérotés 1 et 2

Soit la RegExp ((chien|chat) (blanc|noir)) et la chaîne SUJET 'Le petit chat blanc'

=> les groupes 'chat blanc', 'chat' et 'blanc' sont RESPECTIVEMENT numérotés 1, 2 et 3

Soit la RegExp ((?:chien|chat) (blanc|noir)) et la chaîne SUJET 'Le petit chat noir'

=> les groupes 'chat noir' et 'noir' sont RESPECTIVEMENT numérotés 1 et 2

Soit la RegExp (\d\d)-(?:\u\u)-(\l\l) et la chaîne SUJET '12-AB-cd'

=> les groupes de CAPTURE '12' et 'cd' sont RESPECTIVEMENT numérotés 1 et 2

B) Groupes de CAPTURE NOMMÉS :

La FORME (?<Nom>.....) ou (?'Nom'.....) représente un groupe de CAPTURE NOMMÉ :

- qui peut être RÉPÉTÉ, s'il est SUIVI de QUANTIFICATEURS ( Voir PLUS LOIN )
- qui constitue un GROUPE de CARACTÈRES, dont la VALEUR ou le GABARIT sont RÉUTILISABLES dans l'expression RÉGULIÈRE de RECHERCHE

EXEMPLE :

L'expression RÉGULIÈRE La (Dame) (Blanche), de l'exemple PRÉCÉDENT, peut aussi s'écrire

La (?<Pièce>Dame) (?'Couleur'Blanche), définissant DEUX groupes de CAPTURE NOMMÉS :

Groupe 'Pièce' et Groupe 'Couleur'

REMARQUES :

Les NOMS de groupes de CAPTURE sont UNIQUEMENT composés de CARACTÈRES de MOT (\w)

TOUS les exemples de NOMS de groupe, ci-dessous, sont, par exemple, VALIDES :

Été_76 , Manœuvre , µBiologie , Son_ø , Ægyrine , Lettre_ƒ_Hameçon

Les NOMS des groupes de CAPTURE NOMMÉS doivent comporter 32 CARACTÈRES, au MAXIMUM

Le NOMBRE MAXIMUM de groupes de CAPTURE NOMMÉS est de 10000

Les NOMS de groupes de CAPTURE sont SENSIBLES à la CASSE :

EXEMPLE : les groupes (?<Essai>....) et (?'essaI') représentent 2 groupes DIFFÉRENTS

Si une expression RÉGULIÈRE comporte DEUX ou PLUSIEURS NOMS de groupe de CAPTURE IDENTIQUES,

SEUL le PREMIER groupe est pris en COMPTE, les SUIVANTS étant IGNORÉS

EXEMPLE : ABC(?'Gr'12)DEF(?'Gr'34)GHI(?'Gr'56)JKL peut se SIMPLIFIER en l'expression

RÉGULIÈRE suivante : ABC(?'Gr'12)DEF34GHI56JKL

C) RÉFÉRENCES à un groupe de CAPTURE, NON NOMMÉ :

a) Référence ARRIÈRE ABSOLUE :

- La FORME \N avec 0 < N <= 9
- la FORME \gN , \g{N} , \g<N> ou \g'N', avec 0 < N <= 65535

- la FORME \kN , \k{N} , \k<N> ou \k'N', avec 0 < N <= 65535

représente une RÉFÉRENCE ARRIÈRE, égale au CONTENU ACTUEL du groupe NON NOMMÉ, de numéro ABSOLU N, placée AVANT, dans l'expression RÉGULIÈRE de RECHERCHE

EXEMPLE :

Les REgExp, ci-DESSOUS, recherchent, TOUTES, la chaîne 'ABC', ENTOURÉE d'une MÊME suite, NON NULLE, de CHIFFRES :

(\d+)ABC\1 , (\d+)ABC\g1 , (\d+)ABC\k{1} , (\d+)ABC\g<1> , (\d+)ABC\k'1'

b) Référence ARRIÈRE RELATIVE :

La FORME \g-X , \g{-X} , \g<-X> ou \g'-X'

La FORME \k-X , \k{-X} , \k<-X> ou \k'-X'

représente une RÉFÉRENCE ARRIÈRE, égale au CONTENU ACTUEL du groupe NON NOMMÉ, de numéro RELATIF X, placée AVANT, dans l'expression RÉGULIÈRE de RECHERCHE

NOTE :

La NUMÉROTATION est RELATIVE, avec 1 <= X <= N° de groupe MAXIMUM de l'expression RÉGULIÈRE

EXEMPLES :

Les RegExp, ci-DESSOUS, recherchent, TOUTES, un MOT, ou PORTION de MOT, formant un PALINDROME de 4 LETTRES :

(\w)(\w)\g-1\g-2 , (\w)(\w)\g{-1}\k{-2} , (\w)(\w)\g-1\k-2

(\w)(\w)\k<-1>\g<-2> , (\w)(\w)\k'-1'\k'-2' , (\w)(\w)\g-1\k-2

IMPORTANT : Bien noter, qu'en employant la NUMÉROTATION ABSOLUE, il faudrait, pour obtenir le MÊME résultat, écrire, par exemple :

(\w)(\w)\2\1 , (\w)(\w)\g2\g1 ou (\w)(\w)\g{2}\g{1}

c) Référence de GROUPE ABSOLUE :

La FORME (?N), avec 0 = N <= 65535, représente une RÉFÉRENCE de GROUPE, égale au GABARIT du groupe NON NOMMÉ, de numéro ABSOLU N, placée AVANT ou APRÈS le GROUPE, dans l'expression RÉGULIÈRE de RECHERCHE

EXEMPLE :

La REgExp (\d+)ABC(?1) cherche la chaîne 'ABC', à l'INTÉRIEUR d'une SUITE de CHIFFRES,

NON NULLE

TRÈS IMPORTANT :

Soit, par exemple, le groupe NON NOMMÉ (\d+) et la chaîne SUJET ACTUELLE '123'. Alors :

- la RÉFÉRENCE ARRIÈRE \1 (ou \g{1} ...) vaut EXACTEMENT la chaîne SUJET ACTUELLE '123'
- la RÉFÉRENCE de GROUPE (?1) vaut TOUTE chaîne satisfaisant le GABARIT (\d+)

Soit les 9 chaînes SUJET ci-DESSOUS :

```
1ABC1
12345ABC12345
456ABC456
789ABC789

456ABC789
789ABC456
0ABC123456789
0123456789ABC1
111ABC999
```

- l'expression RÉGULIÈRE (\d+)ABC\1 trouve les 4 PREMIÈRES chaînes SUJET, UNIQUEMENT
- l'expression RÉGULIÈRE (\d+)ABC(?1) trouve les 9 chaînes SUJET, indiquées ci-DESSUS

=> L'expression (\d+)ABC(?1) équivaut, en fait, à l'expression (\d+)ABC(\d+)

Soit les 4 chaînes SUJET ci-DESSOUS :

Chat + Chat
Chien + Chien

Chat + Chien
Chien + Chat

- l'Exp. RÉGULIÈRE (Chat|Chien) \+ \g1 trouve les 2 PREMIÈRES chaînes SUJET, UNIQUEMENT

- l'Exp. RÉGULIÈRE (Chat|Chien) \+ \(?1) trouve les 4 chaînes SUJET ci-DESSUS

REMARQUES :

Une RÉFÉRENCE de GROUPE ABSOLUE peut être placée DEVANT le groupe auquel elle fait RÉFÉRENCE

Aussi, l'expression RÉGULIÈRE (\d+)ABC(?1) peut donc s'écrire (?1)ABC(\d+)

Une RÉFÉRENCE de GROUPE ABSOLUE, placée à l'INTÉRIEUR du groupe, auquel elle fait RÉFÉRENCE,
est une RÉFÉRENCE RÉCURSIVE ( Voir PLUS LOIN )

Les RÉFÉRENCES de GROUPES sont traitées comme des groupes ATOMIQUES, MÊME si le groupe
auquel elles font RÉFÉRENCE, n'est PAS un groupe ATOMIQUE ( Voir PLUS LOIN )

d) Référence de GROUPE RELATIVE :

La FORME (?-X) ou (?+X) représente une RÉFÉRENCE de GROUPE, égale au GABARIT du groupe

NON NOMMÉ, de numéro RELATIF X, placée AVANT ou APRÈS le GROUPE, dans l'expression RÉGULIÈRE de RECHERCHE

NOTE : La NUMÉROTATION est RELATIVE, avec 1 <= X <= N° de groupe MAXIMUM de la RegExp

EXEMPLE :

La REgExp (\d+)ABC(?-1) cherche une chaîne 'ABC', à l'INTÉRIEUR d'une SUITE de CHIFFRES, NON NULLE et trouve également les 9 chaînes SUJET ci-dessus

REMARQUES :

Une RÉFÉRENCE de GROUPE peut être placée AVANT le groupe auquel elle fait RÉFÉRENCE

L'exemple PRÉCÉDENT, peut aussi s'écrire (?+1)ABC(\d+)

Les RÉFÉRENCES de GROUPES sont traitées comme des groupes ATOMIQUES, MÊME si le groupe auquel elles font RÉFÉRENCE, n'est PAS un groupe ATOMIQUE ( Voir PLUS LOIN )

e) RÉFÉRENCES (?0) ou (?R) :

Les RÉFÉRENCES de GROUPE (?0) et (?R) ne sont PAS liées à un groupe PARTICULIER mais, en fait, représentent l'ENSEMBLE du GABARIT de RECHERCHE

Ce sont donc des RÉFÉRENCES RECURSIVES, par DÉFAUT, au PRÉCÉDENT GABARIT de RECHERCHE

EXEMPLES :

Rappel : ( Les QUANTIFICATEURS , + ... seront revus au CHAPITRE 8 ! )

\d+ABC(?0)? cherche TOUTE chaîne 'ABC', PRÉCÉDÉE de CHIFFRES, et SUIVIE d'un 2ÈME groupe : CHIFFRES + 'ABC', comme la chaîne SUJET '123ABC56789ABC'

NOTES :

(\d+ABC)(?1) est strictement IDENTIQUE à l'expression RÉGULIÈRE \d+ABC(?0)?

(?0)?\d+ABC entraîne le MESSAGE d'ERREUR 'Invalid regular expression'

Bien sûr, la FORME \d+ABC(?0) ne pourra JAMAIS RIEN trouver !

\d+ABC|\((?R)\) cherche TOUTE chaîne 'ABC', PRÉCÉDÉE de CHIFFRES ou ce MÊME groupe, CHIFFRES + 'ABC', mis ENTRE PARENTHÈSES

NOTES :

(\d+ABC)|\((?1)\) est IDENTIQUE à l'expression RÉGULIÈRE \d+ABC|\((?R)\)

\((?R)\)|\d+ABC trouve UNIQUEMENT la 2ÈME ALTERNATIVE, car en DÉBUT du GABARIT, la RÉFÉRENCE (?R) ( TOUT le GABARIT ) n'est PAS ENCORE définie

D) RÉFÉRENCES à un groupe de CAPTURE NOMMÉ :

a) Référence ARRIÈRE :

La FORME \g{Nom} , \g<Nom> , \g'Nom'

La FORME \k{Nom} , \k<Nom> ou \k'Nom'

représente une RÉFÉRENCE ARRIÈRE, égale au CONTENU ACTUEL du groupe NOMMÉ, de NOM = Nom, placée AVANT, dans l'expression RÉGULIÈRE de RECHERCHE

EXEMPLES :

Les 12 expressions RÉGULIÈRES, ci-DESSOUS, comportant un GROUPE nommé 'Ess', cherchent, TOUTES, la chaîne 'ABC', ENTOURÉE d'une MÊME suite, NON NULLE, de CHIFFRES :

(?<Ess>\d+)ABC\g{Ess} , (?<Ess>\d+)ABC\g<Ess> , (?<Ess>\d+)ABC\g'Ess'

(?<Ess>\d+)ABC\k{Ess} , (?<Ess>\d+)ABC\k<Ess> , (?<Ess>\d+)ABC\k'Ess'

(?'Ess'\d+)ABC\g{Ess} , (?'Ess'\d+)ABC\g<Ess> , (?'Ess'\d+)ABC\g'Ess'

(?'Ess'\d+)ABC\k{Ess} , (?'Ess'\d+)ABC\k<Ess> , (?'Ess'\d+)ABC\k'Ess'

NOTE : L'utilisation de références ARRIÈRE, ABSOLUES ou RELATIVES, comme si le GROUPE n'était PAS NOMMÉ, reste POSSIBLE :

| | | | | | |
|---|---|---|---|---|---|
| \N | | | | avec | 0 < N <= 9 |
| \gN | \g{N} | \g<N> | \g'N' | avec | 0 < N <= 65535 |
| \kN | \k{N} | \k<N> | \k'N' | avec | 0 < N <= 65535 |
| \g-X | \g{-X} | \g<-X> | \g'-X' | avec | 1 <= X <= n° MAXIMUM |
| \k-X | \k{-X} | \k<-X> | \k'-X' | avec | 1 <= X <= n° MAXIMUM |

b) Référence de GROUPE :

La FORME (?&Nom) ou (?P>Nom) représente une RÉFÉRENCE de GROUPE, égale au GABARIT du groupe NOMMÉ de NOM = Nom, placée AVANT ou APRÈS ce GROUPE, dans l'expression RÉGULIÈRE de RECHERCHE

EXEMPLES :

Les 4 expressions RÉGULIÈRES, ci-DESSOUS, comportant un GROUPE nommé 'Ess', cherchent, TOUTES, une chaîne 'ABC', à l'INTÉRIEUR d'une SUITE de CHIFFRES, NON NULLE

(?<Ess>\d+)ABC(?&Ess) , (?<Ess>\d+)ABC(?P>Ess)

(?'Ess'\d+)ABC(?&Ess) , (?'Ess'\d+)ABC(?P>Ess)

NOTE : L'utilisation de références de GROUPE, ABSOLUES ou RELATIVES, comme si le GROUPE n'était PAS NOMMÉ, reste POSSIBLE :

(?N) avec 0 < N <= 65535

(?-X) ou (?+X) avec 1 <= X <= n° MAXIMUM

REMARQUES :

Une RÉFÉRENCE de GROUPE peut être placée AVANT le groupe auquel elle fait RÉFÉRENCE

Les 4 expressions RÉGULIÈRES, ci-DESSUS, peuvent donc aussi s'écrire :

(?&Ess)ABC(?<Ess>\d+) , (?P>Ess)ABC(?<Ess>\d+)

(?&Ess)ABC(?'Ess'\d+) , (?P>Ess)ABC(?'Ess'\d+)

Une RÉFÉRENCE de GROUPE, d'un groupe NOMMÉ, placée à l'INTÉRIEUR du groupe, auquel elle fait RÉFÉRENCE, est une RÉFÉRENCE RÉCURSIVE ( Voir PLUS LOIN )

Les RÉFÉRENCES de GROUPES sont traitées comme des groupes ATOMIQUES, MÊME si le groupe auquel elles font RÉFÉRENCE, n'est PAS un groupe ATOMIQUE ( Voir PLUS LOIN )

E) Groupes NON CAPTURANTS :

DEUX formes sont possibles :

a) La FORME (?:.....) représente un groupe NON CAPTURANT :

- qui peut être RÉPÉTÉ, s'il est SUIVI de QUANTIFICATEURS ( Voir PLUS LOIN )
- qui NE constitue PAS un GROUPE, réutilisable, dans le GABARIT de RECHERCHE

EXEMPLES :

\d{3,7} recherche un NOMBRE ENTIER de 3 à 7 CHIFFRES quelconques

(\d){3,7} recherche un NOMBRE ENTIER de 3 à 7 CHIFFRES quelconques, dont le DERNIER chiffre, CAPTURÉ, est RÉUTILISABLE

(\d){3,7},\1 recherche un NOMBRE DÉCIMAL avec :

- une partie ENTIÈRE de 3 à 7 CHIFFRES quelconques
- une partie DÉCIMALE de 1 CHIFFRE, égale au CHIFFRE des UNITÉS

(?:\d){3,7}\1  Le groupe NON CAPTURANT (?:\d) => ERREUR car AUCUN groupe de CAPTURE n° 1

NOTES : Si 1 ou PLUSIEURS OPTION(S) sont indiquées au DÉBUT d'un groupe NON CAPTURANT

DEUX syntaxes sont possibles :

(?:(?i).....)  ou  (?i:......)

( Les OPTIONS et QUANTIFICATEURS seront vus PLUS LOIN )

b) La FORME (?|.....) représente un groupe NON CAPTURANT d'ALTERNATIVES :

- qui peut être RÉPÉTÉ, s'il est SUIVI de QUANTIFICATEURS  ( Voir PLUS LOIN )
- qui NE constitue PAS un GROUPE, réutilisable, dans le GABARIT de RECHERCHE
- qui RÉINITIALISE le COMPTAGE des GROUPES de CAPTURE INTERNES, pour CHAQUE ALTERNATIVE trouvée

EXEMPLES :

- Soit l'expression RÉGULIÈRE  (a)(?|x(y)z|(p(q)r)|(t)u(v))(w) qui cherche UNE des TROIS chaînes 'axyzw', 'apqrw' ou 'atuvw'

La FORME (?|.....) implique la NUMÉROTATION des groupes de CAPTURE suivante :

n° 1 : Groupe (a)

n° 2 : Groupe (y) n° 2 : Groupe (p(q)r) n ° 2 : Groupe (t)

n° 3 : NON défini n° 3 : Groupe (q) n ° 3 : Groupe (v)

n° 4 : Groupe (w)

- Soit l'expression RÉGULIÈRE, (a)(x(y)z|(p(q)r)|(t)u(v))(w), recherchant UNE de ces TROIS MÊMES chaînes

Avec le groupe de CAPTURE (.....), la NUMÉROTATION, de ces MÊMES groupes, devient :

n° 1 : Groupe (a)

n° 2 : Groupe x(y)z n° 2 : Groupe (p(q)r) n ° 2 : Groupe (t)u(v)

n° 3 : Groupe (y) n° 4 : Groupe (p(q)r) n ° 6 : Groupe (t)

n° 5 : Groupe (q) n° 7 : Groupe (v)

n° 8 : Groupe (w)

F) ANNEXES sur les GROUPES :

Du fait des MULTIPLES syntaxes ÉQUIVALENTES concernant les GROUPES et les RÉFÉRENCES à ces GROUPES, il est utile de DÉFINIR une syntaxe UNIQUE et MINIMALE, couvrant la MAJORITÉ des CAS :

| GROUPE | RÉFÉRENCE | de N° ABSOLU | de N° RELATIF |
|---|---|---|---|
| (?:.....) NON CAPTURANT | AUCUNE | xxxxxxx | xxxxxxx |
| (?\|.....) NON CAPTURANT | AUCUNE | xxxxxxx | xxxxxxx |
| (.....) NON NOMMÉ | ARRIÈRE de MÊME Valeur<br>de GROUPE de MÊME Gabarit | \N ou \gN<br>(?N) | \g-X<br>(?±X) |
| (?<Nom>.....) NOMMÉ | ARRIÈRE de MÊME Valeur<br>de GROUPE de MÊME Gabarit | \g<Nom><br>(?&Nom) | xxxxxxx<br>xxxxxxx |

Les groupes NOMMÉS ou NON NOMMÉS ne sont PAS RECONNUS à l'INTÉRIEUR d'une LISTE entre CROCHETS !

EXEMPLE :

[abc(?<Guy>0+)] ne signifie PAS une LETTRE MINUSCULE a, b ou c, SUIVIE d'une SUITE, NON NULLE de

de CHIFFRES 0, correspondant au GROUPE 'Guy'

MAIS représente un caractère UNIQUE de la LISTE [()+0<>?Gabcuy], mise dans l'ordre ASCII

Les RÉFÉRENCES, ARRIÈRE ou de GROUPE, ne sont PAS RECONNUES à l'INTÉRIEUR d'une LISTE entre CROCHETS !

EXEMPLE :

(?<Nb>\d+)[\g<Nb>abc(?+1)] (\u*) signifie, en fait, un NOMBRE, SUIVI d'un caractère UNIQUE de la LISTE [()+1<>?Nabcg], dans l'ordre ASCII, SUIVI d'une ESPACE et TERMINÉ par une SUITE, MÊME NULLE, de LETTRES MAJUSCULES

IMPORTANT :

Les GROUPES de CAPTURE, NOMMÉS ou NON NOMMÉS, sont NUMÉROTÉS, par le MOTEUR de RECHERCHE, dans leur ORDRE d'APPARITION dans le GABARIT de la RECHERCHE :

Dans l'expression RÉGULIÈRE (\d+)(?<Guy>Le)(\s+)(?'Thevenot'MEILLEUR)(, bien sûr !+), satisfaisant la chaîne SUJET '12345Le MEILLEUR, bien sûr !!!', on a, par EXEMPLE :

```
Groupe NON NOMMÉ              n° 1   =   '12345'
Groupe NOMMÉ      'Guy'       n° 2   =   'Le'
Groupe NON NOMMÉ              n° 3   =   '          '  ( 10 ESPACES )
Groupe NOMMÉ      'Thevenot'  n° 4   =   'MEILLEUR'
Groupe NON NOMMÉ              n° 5   =   ', bien sûr !!!'
```

TOUTE RÉFÉRENCE ARRIÈRE à un groupe de CAPTURE :

- NOMMÉ, avec un nom NON DÉFINI
- NON NOMMÉ, avec un n° SUPÉRIEUR au MAXIMUM ou NON VALABLE

entraîne, AUTOMATIQUEMENT le message d'ERREUR ' Invalid regular expression '

EXEMPLES :

(?<Toto>\d+)ABC\g<Titi>DEF(?&Tata)GHI => KO car les groupes 'Titi' et 'Tata' n'existent PAS

ABC(\d+)DEF\2GHI\g-3JKL => KO car les groupes 2 et -3 n'existent PAS

TOUTE RÉFÉRENCE ARRIÈRE située à l'INTÉRIEUR du groupe, NOMMÉ ou NON, auquel elle fait RÉFÉRENCE,

entraîne, AUTOMATIQUEMENT le message d'ERREUR ' Invalid regular expression '

EXEMPLES :

ABC(?'Toto'DEF\g'Toto'GHI)JKL => KO car \g'Toto' à l'INTÉRIEUR du GROUPE 'Toto'

ABC(\d+\1)DEF => KO car \1 à l'INTÉRIEUR du 1er GROUPE de PARENTHÈSES

Dans QUELQUES cas, les SYNTAXES, indiquées dans le TABLEAU ci-dessus, ne pourront PAS être utilisées

EXEMPLE :

Soit un GABARIT (\d+)ABC\g1 à trouver, qui DOIT être SUIVI de la CHAÎNE '23'

La syntaxe SIMPLE (\d+)ABC\g123 => Message d'ERREUR ' Invalid regular expression '

DEUX solutions sont possibles :

- (\d+)ABC\g{1}23 ( On ISOLE le NUMÉRO de la RÉFÉRENCE ARRIÈRE )

- (\d+)ABC\g1[2]3 ( On ISOLE le 1ER CHIFFRE, situé APRÈS la RÉFÉRENCE ARRIÈRE )

Le NOMBRE de caractères CONTENUS dans un GROUPE, NOMMÉ ou NON NOMMÉ, est d'environ 30000

NOTE :

En fait, après de NOMBREUX essais :

- j'ai réussi à faire "tenir" 4258000 caractères dans une RÉFÉRENCE ARRIÈRE \1 :

par la RECHERCHE de (.*), avec l'OPTION " . comprend ligne nouvelle "

par le REMPLACEMENT \1Guy !

- j'ai réussi à sélectionner PLUS de 1 MILLION de lettres MINUSCULES 'a' avec a{1048570} !

- j'ai réussi à créer un groupe NOMMÉ, dont le NOM faisait 1015 caractères !

=> On en déduit que le MOTEUR de recherche OPTIMISE l'utilisation de la MÉMOIRE,

en MODIFIANT, de manière DYNAMIQUE, les LIMITES des divers PARAMÈTRES :

- Nombre MAXIMUM de RÉPÉTITIONS d'un CARACTÈRE ou d'un GROUPE
- Nombre MAXIMUM de CARACTÈRES dans un GROUPE
- Nombre MAXIMUM de GROUPES, NOMMÉS et NON NOMMÉS
- Nombre MAXIMUM de CARACTÈRES dans le NOM d'un GROUPE
- Nombre MAXIMUM de RÉFÉRENCES ARRIÈRE ou de GROUPE

Dans la PLUPART des cas, ces limites THÉORIQUES ne seront JAMAIS atteintes !

REMARQUES :

La valeur CAPTURÉE par une RÉFÉRENCE ARRIÈRE, ABSOLUE ou RELATIVE, à un GROUPE de CAPTURE RÉPÉTÉ,

correspond TOUJOURS à l'occurrence FINALE, satisfaisant le GABARIT du groupe qui est RÉPÉTÉ

EXEMPLE :

Soit l'expression RÉGULIÈRE (abc\d{3} *)+ => \\1 = \1

Cette RegExp satisfait la chaîne SUJET 'abc576 abc000 abc999 abc852 => \1 = 'abc852'

car la RÉFÉRENCE ARRIÈRE \1 représente, UNIQUEMENT, la CHAÎNE 'abc852,

et NON, l'UNE des chaînes PRÉCÉDENTES 'abc576 ' , 'abc000 ' ou 'abc999 '

La valeur CAPTURÉE d'un GROUPE peut être ÉGALE à :

- la chaîne VIDE, si ce GROUPE n'a PAS ENCORE été DÉFINI

- à la VALEUR d'une PRÉCÉDENTE occurrence, si la DERNIÈRE occurrence ne fait PAS APPEL à ce GROUPE

EXEMPLE :

Si l'expression RÉGULIÈRE recherchée est (a|(b))+ , alors :

\1 = 'a' et \2 = '' si la chaîne SUJET est 'aaaa'

\1 = 'a' et \2 = 'b' si la chaîne SUJET est 'abaa'

\1 = 'b' et \2 = 'b' si la chaîne SUJET est 'aaab'

L'expression RÉGULIÈRE (a|(bc))\2 trouve la CHAÎNE 'bcbc', mais PAS la CHAÎNE 'abc'

En effet, en prenant la PREMIÈRE alternative, la RÉFÉRENCE ARRIÈRE \2 n'est PAS ENCORE définie

( BIEN que, dans le 2ÈME cas, la RÉFÉRENCE ARRIÈRE \1 soit DÉFINIE et ÉGALE à la CHAÎNE 'a' )

DIFFÉRENCES entre une RÉFÉRENCE ARRIÈRE ou de GROUPE :

- à un groupe OPTIONNEL entier, tels que a(\d*)\1b ou C((a|b)*)(?1)D
- au SOUS-ensemble NON OPTIONNEL d'un groupe OPTIONNEL, tels que a(\d)*\1b ou C(a|b)*(?1)D

QUAND il n'y a PAS de CHIFFRES entre les lettres MINUSCULES 'a' et 'b' ou PAS de LETTRES 'a' et 'b', entre les Lettres MAJUSCULES 'C' et 'D' :

- dans le 1ER cas, le GROUPE (\d*) ou ((a|b)*), étant OPTIONNEL, correspond à la chaîne VIDE, ainsi que les RÉFÉRENCES \1 ou (?1)

  => les CHAÎNES 'ab' et 'CD' sont bien TROUVÉES

- dans le 2ÈME cas, le GROUPE \1 = (\d) ou (?1) ~ (a|b) n'est PAS DÉFINI, car AUCUN CHIFFRE n'existe dans la CHAÎNE 'ab ' et AUCUNE lettre 'a' ou 'b' dans 'CD'

  => les CHAÎNES 'ab' et 'CD' NE sont donc PAS trouvées

(q?)b\1 cherche UNE des DEUX chaînes SUJET 'qbq' ou 'b', car si AUCUNE lettre q, les expressions q? et \1 représentent une chaîne VIDE

(q)?b\1 ne trouve QUE la chaîne 'qbq' et NON les chaînes SUJET 'bq' ou 'b' car, dans ces DEUX dernières, la RÉFÉRENCE ARRIÈRE \1 concerne un GROUPE (q) NON ENCORE trouvé

=> Échec GLOBAL de l'expression RÉGULIÈRE

(a|(bc))\1 cherche UNE des DEUX chaînes SUJET 'aa' ou 'bcbc' car \1 représente l'UN ou l'AUTRE des DEUX cas de l'ALTERNATIVE

(a|(bc))\2 ne trouve QUE la chaîne 'bcbc' et NON la lettre UNIQUE 'a', car la RÉFÉRENCE ARRIÈRE \2 n'est PAS DÉFINIE lorsque la partie GAUCHE ( lettre MINUSCULE a )de l'ALTERNATIVE est concernée !

(\d)(\g{1})(?1) trouve DEUX chiffres IDENTIQUES, suivi d'un TROISIÈME chiffre QUELCONQUE, car la RÉFÉRENCE de groupe (?1) représente le PREMIER groupe \d

(\d)(\g{1})(?2) trouve TROIS chiffres IDENTIQUES, car la RÉFÉRENCE de groupe (?2) représente la RÉFÉRENCE ARRIÈRE \g{1} au PREMIER groupe \d

(\d)((?1))(?1) trouve TROIS chiffres QUELCONQUES, car la RÉFÉRENCE de groupe (?1) représente le

PREMIER groupe \d

(\d)((?1))(?2) trouve TROIS chiffres QUELCONQUES, car la RÉFÉRENCE de groupe (?2) représente la RÉFÉRENCE de groupe (?1), qui représente, ELLE-MÊME, un chiffre QUELCONQUE \d !

Exemple RÉCAPITULATIF :

La RegExp AB(?&Ess)CD(?1)EF(?+2)GH(\d+)IJ\1KL(?:MNO)PQ(?<Ess>[a-z]+)RS\g-2TU(?-1)VW\g'Ess'XYZ

trouve, par exemple, les TROIS lignes ci-dessous :

ABceciCD12345EFestGH789IJ789KLMNOPQunRS789TUtoutVWunXYZ

ABpetitCD000999EFessaiGH123IJ123KLMNOPQdeRS123TUtexteVWdeXYZ

ABpourCD11111EFvoirGH110011IJ110011KLMNOPQcelaRS110011TUabcdefghVWcelaXYZ

Pour une MEILLEURE compréhension, les différents GROUPES, RÉFÉRENCES et BLOCS de TEXTE, de l'expression RÉGULIÈRE et des 3 chaînes SUJET, ont été disposés en COLONNES, dans le TABLEAU ci-DESSOUS, et dans lequel :

- les GROUPES NON CAPTURANTS sont SOULIGNÉS, avec le signe ÉGAL

- les GROUPES de CAPTURE sont SOULIGNÉS, avec le signe TRAIT HAUT

- les RÉFÉRENCES ARRIÈRE sont SOULIGNÉES, avec le signe POINTILLÉ

- les RÉFÉRENCES de GROUPES sont SOULIGNÉES, avec le signe CHAPEAU

et

- Gn signifie le Nème GROUPE, NOMMÉ ou NON, de L'expression RÉGULIÈRE

- An signifie la RÉFÉRENCE ARRIÈRE égale à la VALEUR ACTUELLE du GROUPE n

- Rn signifie la RÉFÉRENCE de GROUPE égale au GABARIT du GROUPE n

| AB | (?&Ess) | CD | (?1) | EF | (?+2) | GH | (\d+) | IJ | \1 | KL | (?:MNO) | PQ | (?<Ess>[a-z]+) | RS | \g-2 | TU | (?-1) | VW | \g'Ess' | XYZ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | R2 | | R1 | | R2 | | G1 | | A1 | | | | G2 | | A1 | | R2 | | A2 | |

| AB | ceci | CD | 12345 | EF | est | GH | 789 | IJ | 789 | KL | MNO | PQ | un | RS | 789 | TU | tout | VW | un | XYZ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ^^^^ | | ^^^^^ | | ^^^ | | | | ¨¨¨ | | === | | | | ¨¨¨ | | ^^^^ | | ¨¨ | |
| | R2 | | R1 | | R2 | | G1 | | A1 | | | | G2 | | A1 | | R2 | | A2 | |
| AB | petit | CD | 000999 | EF | essai | GH | 123 | IJ | 123 | KL | MNO | PQ | de | RS | 123 | TU | texte | VW | de | XYZ |
| | ^^^^ | | ^^^^^^ | | ^^^^^ | | | | ¨¨¨ | | === | | | | ¨¨¨ | | ^^^^^ | | ¨¨ | |
| | R2 | | R1 | | R2 | | G1 | | A1 | | | | G2 | | A1 | | R2 | | A2 | |
| AB | pour | CD | 11111 | EF | voir | GH | 110011 | IJ | 110011 | KL | MNO | PQ | cela | RS | 110011 | TU | abcdefgh | VW | cela | XYZ |
| | ^^^^ | | ^^^^^ | | ^^^^ | | | | ¨¨¨¨¨¨ | | === | | | | ¨¨¨¨¨¨ | | ^^^^^^^^ | | ¨¨¨¨ | |
| | R2 | | R1 | | R2 | | G1 | | A1 | | | | G2 | | A1 | | R2 | | A2 | |

G) RÉFÉRENCES RÉCURSIVES :

TOUTE RÉFÉRENCE de GROUPE (?N), avec N > 0, située à l'INTÉRIEUR du groupe, NOMMÉ ou NON, auquel elle fait RÉFÉRENCE, est considérée comme une RÉFÉRENCE RÉCURSIVE au GABARIT de ce GROUPE

EXEMPLES : ( Voir QUANTIFICATEURS, ci-APRÈS )

(\d(?1)*) équivaut à l'expression RÉGULIÈRE (\d)(?1)* , soit à la RegExp \d{1,} ou \d+

(\d(?1)+) équivaut à l'expression RÉGULIÈRE (\d)(?1)+ , soit à la RegExp \d{2,}

(\d(?1)?) équivaut à l'expression RÉGULIÈRE (\d)(?1)? , soit à la RegExp \d{1,2}

NOTE :

BIZARREMENT, les 3 FORMES (\d*(?1)) , (\d+(?1)) et (\d?(?1)) équivalent à la SÉLECTION de TOUT le FICHIER, MÊME s'il contient des caractères NON CHIFFRES ?!

La RÉFÉRENCE particulière, à TOUT le GABARIT de RECHERCHE, (?R) ou (?0), est TOUJOURS, par DÉFINITION même, une RÉFÉRENCE RÉCURSIVE au gabarit ENTIER

Les RÉFÉRENCES RÉCURSIVES au groupe N (?N) ou à TOUT le GABARIT (?R) et (?0) sont considérées comme des groupes ATOMIQUES par DÉFAUT, c'est à dire qu'AUCUN BACKTRACKING n'aura lieu pour obtenir une

CONCORDANCE de l'expression RÉGULIÈRE globale, même si d'autres ALTERNATIVES n'ont PAS été testées !

7) OPTIONS :

Le MOTEUR de RECHERCHE d'expression RÉGULIÈRES PCRE de Notepad++ possède 4 OPTIONS internes :

A) l'OPTION i et l'OPTION -i, son CONTRAIRE :

Quand l'OPTION i est utilisée, la RECHERCHE des LETTRES se fait SANS respect de la CASSE

Quand l'OPTION -i est utilisée, la RECHERCHE des LETTRES se fait AVEC respect de la CASSE

Quand AUCUNE des 2 OPTIONS i et -i n'est PRÉSENTE, la RECHERCHE des LETTRES se fait :

- SANS RESPECT de la CASSE si la CASE " Respecter la casse " est DÉCOCHÉE
- AVEC RESPECT de la CASSE si la CASE " Respecter la casse " est COCHÉE

B) l'OPTION m et l'OPTION -m, son CONTRAIRE :

Quand l'OPTION m est utilisée OU qu'AUCUNE des 2 OPTIONS m et -m n'est PRÉSENTE :

- L'ANCRE ^ représente la POSITION :

- ENTRE le TOUT DÉBUT du fichier COURANT et le PREMIER caractère du fichier

- ENTRE un caractère 'FIN de LIGNE' et le PREMIER caractère de la ligne SUIVANTE

- L'ANCRE $ représente la POSITION :

- ENTRE le DERNIER caractère d'une LIGNE et un caractère 'FIN de LIGNE'

- ENTRE le DERNIER caractère du fichier et la TOUTE FIN du fichier COURANT

Quand l'OPTION (?-m) est utilisée :

- l'ANCRE ^ représente la POSITION entre le TOUT DÉBUT du fichier et le PREMIER caractère, UNIQUEMENT

- l'ANCRE $ représente la POSITION entre le DERNIER caractère et la TOUTE FIN du fichier, UNIQUEMENT

C) l'OPTION s et l'OPTION -s, son CONTRAIRE :

Quand l'OPTION -s est utilisée :

- Le JOKER . représente TOUT caractère UNIQUE, AUTRE que les TROIS caractères :

  \n ( \x0A ) , \f ( \x0C ) et \r ( \x0D )

Quand l'OPTION s est utilisée :

- Le JOKER . représente TOUT caractère UNIQUE, y COMPRIS l'UN des TROIS caractères :

  \n ( \x0A ) , \f ( \x0C ) ou \r ( \x0D )

Quand AUCUNE des 2 OPTIONS s et -s n'est PRÉSENTE, le JOKER . représente :

- TOUT caractère UNIQUE, AUTRE que \n, \f et \n , si la CASE " . comprend ligne nouvelle " est DÉCOCHÉE
- TOUT caractère UNIQUE, y COMPRIS \n, \f et \n , si la CASE " . comprend ligne nouvelle " est COCHÉE

D) l'OPTION x et l'OPTION -x, son CONTRAIRE :

Quand l'OPTION -x est utilisée OU qu'AUCUNE des 2 OPTIONS x et -x n'est PRÉSENTE :

- Le caractère ESPACE ( \x20 ) est un caractère SIGNIFICATIF de l'expression RÉGULIÈRE

Quand l'OPTION x est utilisée :

- Le caractère ESPACE ( \x20 ) est un caractère NON SIGNIFICATIF et IGNORÉ de l'expression RÉGULIÈRE

NOTE : Si le caractère ESPACE est PRÉCÉDÉ d'un ANTISLASH \ ou placé entre CROCHETS [], il est rendu, de nouveau, SIGNIFICATIF

E) PARTICULARITÉS dans NOTEPAD++ :

- les OPTIONS i et -i sont PRIORITAIRES sur le MARQUAGE, ou NON, de la CASE " Respecter la casse "

- les OPTIONS s et -s sont PRIORITAIRES sur le MARQUAGE, ou NON, de la CASE " . comprend ligne nouvelle "

- l'OPTION m est ACTIVÉE, par DÉFAUT => Utiliser l 'OPTION -m, si nécessaire pour la RegExp

- l'OPTION x n'est PAS ACTIVÉE par DÉFAUT => Utiliser l'OPTION x, si nécessaire pour la RegExp

F) DÉFINITION de l'OBJET <Bloc> :

<Bloc> = <Options>|-<Options>|<Options>-<Options> avec <Options> = [imsx]+

EXEMPLES de <Bloc> :

| | |
|---|---|
| i | Option i ACTIVÉE |
| -m | Option m DÉSACTIVÉE |
| ix | Options i et x ACTIVÉES |
| i-m | Option i ACTIVÉE et option m DÉSACTIVÉE |
| sx-mi | Options s et x ACTIVÉES et options m et i DÉSACTIVÉES |

G) SYNTAXE des OPTIONS :

TROIS formes sont possibles :

a) La FORME (?<Bloc>) :

Les OPTIONS, dans <Bloc>, sont ACTIVES ou INACTIVES :

- jusqu'au PROCHAIN bloc d'OPTIONS rencontré
- jusqu'à la FIN de l'expression RÉGULIÈRE, si PAS d'AUTRE bloc d'OPTIONS

EXEMPLES : ( avec CASES " Respecter la casse " COCHÉE et " . comprend ligne nouvelle " DÉCOCHÉE )

| | |
|---|---|
| a(?i)bc | cherche UNE des 4 CHAÎNES 'abc' , 'aBc' , 'abC' ou 'aBC' |
| a(?i)b\|c | cherche UNE des 4 CHAÎNES 'ab' , 'AB' , 'c' ou 'C' |
| ABC(?i)DEF(?-i)GHI | cherche 'ABC' suivi de 'DEF', QUELLE que SOIT la CASSE, suivi de 'GHI' |
| ABC(?i)DEFGHI | cherche la CHAÎNE 'ABC' suivi de 'DEFGHI', QUELLE que SOIT la CASSE |
| ABC(?x-i) d E f G | cherche la chaîne EXACTE 'ABCdEfG' |
| (?s-i) AbC.*dEf | cherche la PLUS LONGUE chaîne, débutant par 'AbC', EXACTEMENT et finissant par 'dEf', EXACTEMENT, MÊME répartie sur PLUSIEURS lignes |

NOTE : Les FORMES (?i), (?s), (?m) et (?x) NE constituent PAS des groupes de CAPTURE, en soi !

b) La FORME ((?<Bloc>)....) :

Les OPTIONS, dans <Bloc>, sont ACTIVES ou INACTIVES dans ce SEUL groupe de CAPTURE, UNIQUEMENT

EXEMPLES : ( Cases " Respecter la casse " COCHÉE et " . comprend ligne nouvelle " DÉCOCHÉE )

(a(?i)b)c cherche UNE des 2 CHAÎNES 'abc' ou 'aBc'

((?-i)aBc) \1 cherche la chaîne EXACTE 'aBc', SUIVIE d'un 2ÈME groupe de ces 3 MÊMES lettres, SÉPARÉ par une ESPACE, soit la CHAÎNE 'aBc aBc'

((?-i)aBc) (?i)\1 cherche la chaîne EXACTE 'aBc', SUIVIE d'un 2ÈME groupe de ces 3 MÊMES lettres, QUELLE que SOIT leur CASSE, SÉPARÉ par une ESPACE :

'aBc abc' , 'aBc aBc' , 'aBc abC' , 'aBc aBC'

'aBc Abc' , 'aBc ABc' , 'aBc AbC' , 'aBc ABC'

((?i)abc) \1 cherche 1 CHAÎNE 'abc', de casse QUELCONQUE, RÉPÉTÉE, avec 1 ESPACE entre

'abc abc' , 'aBc aBc' , 'abC abC' , 'aBC aBC'

'Abc Abc' , 'ABc ABc' , 'AbC AbC' , 'ABC ABC'

((?i)abc) (?1) cherche DEUX chaînes 'abc', de casse QUELCONQUE, séparées d'une ESPACE

telles que 'abc abc' , 'ABC ABC' , 'ABC abc' ou 'AbC aBc'

((?-i)abc) (?i)(?1) cherche la chaîne EXACTE 'abc', SÉPARÉE de cette MÊME chaîne, par une

ESPACE , soit la SEULE chaîne 'abc abc'

((?i)[[:upper:]])\1 cherche une LETTRE quelconque, MINUSCULE ou MAJUSCULE, DOUBLÉE

c) Les FORMES (?:(?<Bloc>)....) ou la forme SIMPLIFIÉE (?<Bloc>):....) :

Les OPTIONS dans <Bloc> sont ACTIVES ou INACTIVES dans ce SEUL groupe NON CAPTURANT, UNIQUEMENT

EXEMPLES : ( Cases " Respecter la casse " COCHÉE et " . comprend ligne nouvelle " DÉCOCHÉE )

(?-m:^ABC.....XYZ$) cherche un FICHIER, d'une SEULE ligne, avec les 26 LETTRES de l'ALPHABET

T(?i:es)t cherche UNE des QUATRE chaînes 'Test' , 'TeSt' , 'TEst' ou 'TESt'

12(?x: \l \u \ u)3 4 cherche la CHAÎNE '12' + 1 MINUSCULE + 1 MAJUSCULE + la CHAÎNE ' u3 4'

(?i:)ABC cherche la chaîne EXACTE 'ABC', car le groupe (?i:) représente, en fait, un groupe NON CAPTURANT VIDE, et l'OPTION (?i) s'applique à ce SEUL groupe et NON à la FIN de la RegExp, soit 'ABC'

REMARQUES :

Les OPTIONS sont sensibles à la CASSE => Les FORMES (?I), (?M), (?M) ou (?X) provoquent une ERREUR !

Les OPTIONS, prises en compte par les RÉFÉRENCES ARRIÈRE, sont celles EXISTANTES au MOMENT de l'APPEL effectif à ces GROUPES de CAPTURE, par une RÉFÉRENCE ARRIÈRE

Les OPTIONS, prises en compte par les RÉFÉRENCES de GROUPE, sont celles EXISTANTES au MOMENT de la DÉFINITION de ces GROUPES de CAPTURE ( et NON au MOMENT de leur UTILISATION, avec la FORME (?.) )

EXEMPLE : ( Case " Respecter la casse " COCHÉE )

Le GABARIT ([ab][ab])(?i)\1 trouve les 16 chaînes :

| | | | |
|---|---|---|---|
| 'aaaa' | 'abab' | 'baba' | 'bbbb' |
| 'aaaA' | 'abaB' | 'babA' | 'bbbB' |
| 'aaAa' | 'abAb' | 'baBa' | 'bbBb' |
| 'aaAA' | 'abAB' | 'baBA' | 'bbBB' |

=> \1 = VALEUR du Groupe 1 = les 2 PREMIERS car. EXACTEMENT, QUELLE que SOIT leur CASSE

=> l'OPTION (?i) est EFFECTIVE, car relative à la VALEUR de la RÉFÉRENCE ARRIÈRE au Groupe 1

Le GABARIT ([ab][ab])(?i)(?1) trouve les 16 chaînes :

| | | | |
|---|---|---|---|
| 'aaaa' | 'abaa' | 'baaa' | 'bbaa' |
| 'aaab' | 'abab' | 'baab' | 'bbab' |
| 'aaba' | 'abba' | 'baba' | 'bbba' |
| 'aabb' | 'abbb' | 'babb' | 'bbbb' |

=> (?1) = GABARIT du Groupe 1 = [ab][ab] avec la SEULE casse MINUSCULE, MALGRÉ l'OPTION (?i)

=> l'OPTION (?i) est SANS effet, car NON EXISTANTE, au moment de la DÉFINITION du Groupe 1

QUAND la CASE " Respecter la casse " n'est PAS COCHÉE, la RegExp ((?-i)aBc) \1 , ci-AVANT, cherche :

- la chaîne EXACTE 'aBc' car l'OPTION (?-i) est PRIORITAIRE sur l'ÉTAT de la CASE " Respecter la casse "
- le caractère ESPACE
- la MÊME chaîne 'aBc', QUELLE que SOIT sa CASSE, car la CASE " Respecter la casse " est DÉCOCHÉE

=> les 8 CHAÎNES ci-DESSOUS, sont trouvées

'aBc abc' , 'aBc abC' , 'aBc aBc' ou 'aBc aBC'

'aBc Abc' , 'aBc AbC' , 'aBc ABc' ou 'aBc ABC'

SI la CASE " Respecter la casse " est COCHÉE, alors :

- la FORME Samedi|Dimanche cherche l'UNE des DEUX chaînes 'Samedi' ou 'Dimanche'

- les 4 FORMES (?i)Samedi|Dimanche , (?i)(Samedi|Dimanche) , (?i:Samedi|Dimanche) ou
  (?:(?i)Samedi|Dimanche) cherchent l'UN des MOTS 'Samedi' ou 'Dimanche', QUELLE que SOIT leur CASSE

- les 2 FORMES Samedi|(?i)Dimanche ou Samedi|(?i:Dimanche) cherchent la chaîne EXACTE 'Samedi' ou
  le MOT 'Dimanche', QUELLE que SOIT sa CASSE

- les 2 FORMES ((?i)Samedi)|Dimanche ou (?i:Samedi)|Dimanche cherchent la chaîne EXACTE 'Dimanche'
  ou le MOT 'Samedi', QUELLE que SOIT sa CASSE

8) QUANTIFICATEURS :

Les QUANTIFICATEURS de RÉPÉTITION peuvent être placés APRÈS CHACUNE des FORMES suivantes :

- un caractère UNIQUE ( selon UNE des 13 SYNTAXES possibles, indiquées au CHAPITRE 2, PARTIE 1 )
- une RÉFÉRENCE ARRIÈRE ou de GROUPE, relatif à un GROUPE de CAPTURE, NOMMÉ ou NON

- une liste d'ALTERNATIVES (..|..|..), SAUF si les différentes ALTERNATIVES sont des ASSERTIONS

- un GROUPE de CAPTURE (........), SAUF s'il s'agit d'une ASSERTION

- un groupe NON CAPTURANT (?:......) ou NON CAPTURANT d'ALTERNATIVES (?|..|...|...)

Il existe 3 TYPES de QUANTIFICATEURS :

- les QUANTIFICATEURS NON POSSESSIFS, de type " Greedy " ( ~ " Fougueux " )

- les QUANTIFICATEURS NON POSSESSIFS, de type " Lazy " ( ~ " Paresseux " )

- les QUANTIFICATEURS POSSESSIFS, de type " Greedy " ( ~ " Entier " )

Il existe 6 FORMES de QUANTIFICATEURS :

A) La FORME : {n} :

Elle signifie que l'entité PRÉCÉDENTE, à laquelle elle est RATTACHÉE, doit être PRÉSENTE,

EXACTEMENT n FOIS

EXEMPLES :

ab{5}c cherche la SEULE chaîne 'abbbbbc'

a.{5}c cherche les DEUX lettres MINUSCULES 'a' et 'c', SÉPARÉES par 5 caractères QUELCONQUES

NOTE : On a la RELATION : 0 <= n <= 65535

B) La FORME : {n,} :

Elle signifie que l'entité PRÉCÉDENTE, à laquelle elle est RATTACHÉE, doit être PRÉSENTE, AU MOINS n FOIS

EXEMPLES :

ab{5,}c cherche les DEUX lettres MINUSCULES 'a' et 'c', SÉPARÉES par AU MOINS 5 MINUSCULES 'b'

a.{5,}c cherche les DEUX lettres MINUSCULES 'a' et 'c', SÉPARÉES par AU MOINS 5 car QUELCONQUES

NOTE : On a la RELATION : 0 <= n <= 65535

C) La FORME : {n,m} :

Elle signifie que l'entité PRÉCÉDENTE, à laquelle elle est RATTACHÉE, doit être PRÉSENTE,

AU MOINS n FOIS et AU PLUS m FOIS

EXEMPLES :

ab{5,7}c  cherche les DEUX lettres MINUSCULES 'a' et 'c', SÉPARÉES par 5, 6 ou 7 MINUSCULES 'b'

a.{5,7}c  cherche les DEUX lettres MINUSCULES 'a' et 'c', SÉPARÉES par 5 à 7 caractères QUELCONQUES

NOTE :  On a les 3 RELATIONS :  0 <= n <= 65535  ,  0 <= m <= 65535  et  n <= m

D) La FORME : ? :

Elle signifie que l'entité PRÉCÉDENTE, à laquelle elle est RATTACHÉE, peut être ABSENTE ou être PRÉSENTE, 1 FOIS, SEULEMENT

EXEMPLES :

ab?c  cherche UNE des DEUX chaînes 'ab' ou 'abc', UNIQUEMENT

a.?c  cherche les DEUX lettres MINUSCULES 'a' et 'c', JOINTES ou SÉPARÉES par 1 caractère QUELCONQUE

NOTE : En fait, la FORME ? est ÉQUIVALENTE à la FORME {0,1}

E) La FORME : * :

Elle signifie que l'entité PRÉCÉDENTE, à laquelle elle est RATTACHÉE, peut être ABSENTE ou être

PRÉSENTE, AU MOINS 1 FOIS

EXEMPLES :

ab*c  cherche DEUX MINUSCULES a et c, SÉPARÉES par un Nbre QUELCONQUE de MINUSCULES 'b', MÊME AUCUNE

a.*c  cherche DEUX MINUSCULES a et c, SÉPARÉES par un Nbre QUELCONQUE de car. QUELCONQUES, MÊME AUCUN

NOTE : En fait, la FORME * est ÉQUIVALENTE à la FORME {0,}

F) La FORME : + :

Elle signifie que l'entité PRÉCÉDENTE, à laquelle elle est RATTACHÉE, doit être PRÉSENTE, AU MOINS 1 FOIS

EXEMPLES :

ab+c  cherche DEUX MINUSCULES 'a' et 'c', SÉPARÉES par un nombre, NON NUL, de lettres MINUSCULES 'b'

a.+c  cherche DEUX MINUSCULES 'a' et 'c', SÉPARÉES par un nombre, NON NUL, de caractères QUELCONQUES

NOTE : En fait, la FORME + est ÉQUIVALENTE à la FORME {1,}

G) TYPES des QUANTIFICATEURS :

a) NON POSSESSIF, de TYPE " Greedy " ( " Fougueux " ) :

Les 6 QUANTIFICATEURS de type " Greedy " USUELS sont {n} , {n,} , {n,m} , ? , * et +

Avec un QUANTIFICATEUR de TYPE " Greedy ", le MOTEUR de RECHERCHE sélectionne, en PREMIER, la forme RÉPÉTÉE, avec la valeur MAXIMALE indiquée pour ce QUANTIFICATEUR

Si, le RESTE de l'expression RÉGULIÈRE NE correspond PAS à la chaîne SUJET, le MOTEUR de RECHERCHE sélectionne la forme RÉPÉTÉE, avec la valeur MAXIMALE - 1

Si, le RESTE de l'expression RÉGULIÈRE NE correspond PAS à la chaîne SUJET, le MOTEUR de RECHERCHE sélectionne la forme RÉPÉTÉE, avec la valeur MAXIMALE - 2

Et, AINSI DE SUITE, jusqu'à sélection de la forme RÉPÉTÉE, avec la valeur MINIMALE du QUANTIFICATEUR

Si le RESTE de l'expression RÉGULIÈRE NE correspond TOUJOURS PAS => AUCUNE solution trouvée

b) NON POSSESSIFS, de TYPE " Lazy " ( " Paresseux " ) :

Un QUANTIFICATEUR de type " Lazy " s'obtient en AJOUTANT le CARACTÈRE '?' APRÈS le QUANTIFICATEUR de type " Greedy ", de MÊME nature

Les 5 QUANTIFICATEURS de type " Lazy " USUELS sont {n,}? , {n,m}? , ?? , *? et +?

Avec un QUANTIFICATEUR de TYPE " Lazy ", le MOTEUR de RECHERCHE sélectionne, en PREMIER, la forme RÉPÉTÉE, avec la valeur MINIMALE indiquée pour ce QUANTIFICATEUR

Si, le RESTE de l'expression RÉGULIÈRE NE correspond PAS à la chaîne SUJET, le MOTEUR de RECHERCHE sélectionne la forme RÉPÉTÉE, avec la valeur MINIMALE + 1

Si, le RESTE de l'expression RÉGULIÈRE NE correspond PAS à la chaîne SUJET, le MOTEUR de RECHERCHE sélectionne la forme RÉPÉTÉE, avec la valeur MINIMALE + 2

Et, AINSI DE SUITE, jusqu'à sélection de la forme RÉPÉTÉE, avec la valeur MAXIMALE du QUANTIFICATEUR

Si le RESTE de l'expression RÉGULIÈRE NE correspond TOUJOURS PAS => AUCUNE solution trouvée

IMPORTANT :

Si l'expression RÉGULIÈRE se TERMINE par la forme RÉPÉTÉE, SEULE la valeur MINIMALE du QUANTIFICATEUR sera donc utilisée

c) POSSESSIFS, de type " Greedy " IMPLICITE :

Un QUANTIFICATEUR POSSESSIF s'obtient en AJOUTANT le CARACTÈRE '+' APRÈS le QUANTIFICATEUR de type " Greedy ", de MÊME nature

Les 5 QUANTIFICATEURS POSSESSIFS USUELS sont {n,}+ , {n,m}+ , ?+ , *+ et ++

Avec un QUANTIFICATEUR POSSESSIF, le MOTEUR de RECHERCHE sélectionne la forme RÉPÉTÉE, avec la valeur MAXIMALE indiquée pour ce QUANTIFICATEUR, comme pour le TYPE " Greedy "

Par contre, si le RESTE de l'expression RÉGULIÈRE NE correspond PAS à la chaîne SUJET, le MOTEUR de RECHERCHE NE revient PAS en ARRIÈRE ( BACKTRACKING ) pour essayer d'AUTRES valeurs de ce QUANTIFICATEUR

=> FIN de la RECHERCHE et AUCUNE solution trouvée

d) EXEMPLES :

- Les SOUS-GABARITS '\d+' ou '\d+?' , AJUSTENT le NOMBRE de CHIFFRES, de manière à ce que la

SUITE ÉVENTUELLE, du gabarit CORRESPONDE

- Le SOUS-GABARIT '\d++', ATOMIQUE, utilise le nbre MAXIMUM de CHIFFRES, QUELLE que SOIT la SUITE, ÉVENTUELLE, du gabarit à APPARIER, pouvant, parfois, entraîner l'ÉCHEC de la RECHERCHE

Les expressions RÉGULIÈRES .*abcDE ou .*?abcDE trouvent la chaîne SUJET '012345abcDE'

L'expression RÉGULIÈRE .*+abcDE, elle, ne trouve RIEN, car la 1ÈRE PARTIE du GABARIT '.*+' consomme, DÉJÀ, TOUS les caractères jusqu'au(x) caractère(s) 'FIN de LIGNE', EXCLUS

La RegExp ab+?c ( ou ab+c ) trouve TOUTE chaîne COMMENÇANT par 'a' et FINISSANT par 'c' avec un nombre NON NUL de CARACTÈRES 'b', ENTRE les DEUX

La RegExp ab+? ne trouve que la SEULE chaîne 'ab', puisque le QUANTIFICATEUR " Lazy " n'est suivi d'AUCUN caractère !

Si la chaîne SUJET 'abbbbbbbbbbbc' :

- la RegExp ab+\l\l trouve celle-ci : Il y a RETOUR en ARRIÈRE d'1 caractère et '\l\l' est pris par la chaîne 'bc'

- la RegExp ab++\l\l ne trouve RIEN car 'b++' consomme TOUS les CARACTÈRES 'b' et '\l\l' NE peut PAS correspondre au SEUL caractère 'c' FINAL

Si la chaîne SUJET est 'abbbbbbbbbbbbcd', les DEUX gabarits, ci-dessus, fonctionnent car '\l\l' est représenté, dans les DEUX cas, par la chaîne 'cd'

La RegExp \b(cat|dog|mouse)(?:\W+\w+){0,6}?\W+(cat|dog|mouse)\b trouve la CHAÎNE 'cat', 'dog' ou 'mouse', SÉPARÉE par 6 MOTS au PLUS, MÊME sur PLUSIEURS lignes, d'une AUTRE chaîne 'cat', 'dog' ou 'mouse'

La RegExp \b(cat|dog|mouse)(?:[^\w\r\n]+\w+){0,6}?[^\w\r\n]+(cat|dog|mouse)\b trouve la CHAÎNE 'cat', 'dog' ou 'mouse', SÉPARÉE par 6 MOTS au PLUS d'une MÊME ligne, d'une AUTRE chaîne 'cat', 'dog' ou 'mouse'

NOTE : Le CAS 'dog........dog', par exemple, est donc POSSIBLE

d) REMARQUE :

Soit la chaîne SUJET 'abcabcabcabcabcabcdefdefdefdefdefdefghighighighighighi' ( 6 FOIS la

CHAÎNE 'abc', SUIVIE de 6 FOIS la CHAÎNE 'def', SUIVIE de 6 FOIS la CHAÎNE 'ghi' )

- La RegExp (abc|def|ghi){3,4} représente 3 ou 4 groupes de TROIS lettres, pouvant être,

  CHOIX 'abc' , 'def' ou 'ghi' , soit, SUCCESSIVEMENT, les QUATRE chaînes CONTIGUËS :

  'abcabcabcabc' , 'abcabcdefdef' , 'defdefdefdef' et, enfin, 'ghighighighi'

- La RegExp (abc){3,4}|(def){3,4}|(ghi){3,4} représente 3 ou 4 groupes de TROIS lettres,

  CHACUN d'eux DEVANT être ÉGAL à 'abc' , 'def' ou 'ghi, soit, SUCCESSIVEMENT, les TROIS chaînes NON contiguës :

  'abcabcabcabc' , 'defdefdefdef' et, enfin, 'ghighighighi'

H) Groupes ATOMIQUES :

Un groupe ATOMIQUE est la syntaxe ORIGINELLE d'une forme RÉPÉTÉE, avec un QUANTIFICATEUR POSSESSIF

On a les ÉQUIVALENCES suivantes :

| | | |
|---|---|---|
| Caractère<Quantificateur POSSESSIF> | == | (?>Caractère<Quantificateur GREEDY>) |
| (.......)<Quant. Possessif> | == | (?>(.......)<Quantificateur GREEDY>) |
| (?:.....)<Quant. Possessif> | == | (?>(?:.....)<Quantificateur GREEDY>) |

REMARQUE :

Un groupe ATOMIQUE ne constitue PAS, en soi, un groupe de CAPTURE. Si celui-ci doit être RÉUTILISÉ, on utilisera un COUPLE supplémentaire de PARENTHÈSES INTERNES

EXEMPLES :

(?>(\d+))abc\1  cherche TOUTE chaîne 'abc', ENTRE DEUX séries IDENTIQUES de CHIFFRES

Quand on recherche l'INTÉGRALITÉ des LIGNES d'un fichier FINISSANT par une CHAÎNE ¤¤¤ FIXE, la recherche sera PLUS efficace et PLUS rapide, en combinant groupe ATOMIQUE et LOOKBEHIND POSITIF ( Voir PARTIE 9, ci-APRÈS )

La RegExp (?>(.*))(?<=¤¤¤)\R? ou la forme POSSESSIVE équivalente (.*+)(?<=¤¤¤)\R? sera ainsi préférée à (.*¤¤¤)$\R?

En effet :

- le groupe ATOMIQUE (?>(.*)) prend, OBLIGATOIREMENT, TOUS les caractères de CHAQUE ligne, depuis le PREMIER caractère jusqu'au DERNIER caractère ( => ANCRE $ NON nécessaire )

- le groupe de CAPTURE (.*) les MÉMORISE pour une réutilisation ÉVENTUELLE

- l'ASSERTION (?<=¤¤¤) vérifie si la chaîne FIXE ¤¤¤ est bien PRÉSENTE à la FIN de CHAQUE ligne

MAIS, en cas de résultat NÉGATIF, AUCUN BACKTRACKING n'a lieu

=> La recherche échoue PLUS VITE et on passe IMMÉDIATEMENT à la ligne SUIVANTE !

## I) Groupes OPTIONNELS :

Un groupe OPTIONNEL représente, par CONVENTION, un GROUPE de CAPTURE qui peut être PRÉSENT ou NON, c'est à dire, TOUT groupe, SUIVI d'UN des 4 QUANTIFICATEURS suivants :

{0,n} avec n > 0 , (0,) , ? ou *

EXEMPLES :

(\d)* représente un CHIFFRE, un NOMBRE ou la chaîne VIDE => GROUPE 1 = CHIFFRE de 0 à 9

(abc)? représente la CHAÎNE 'abc' ou la chaîne VIDE => GROUPE 1 = CHAÎNE 'abc'

(\xFF){0,5} représente de 1 à 5 caractères \x0C ou la chaîne VIDE => GROUPE 1 = CARACTÈRE \x0C

IMPORTANT :

Un groupe OPTIONNEL peut TOUJOURS s'écrire sous la FORME d'un groupe NON OPTIONNEL, SUIVI du QUANTIFICATEUR ? :

- la FORME ((RegExp){1,n})? est IDENTIQUE à la FORME (RegExp){0,n}
- la FORME ((RegExp){1,})? est IDENTIQUE à la FORME (RegExp){0,}
- la FORME ((RegExp)+)? est IDENTIQUE à LA FORME (RegExp)*

Cette SYNTAXE ' groupe NON OPTIONNEL, SUIVI du QUANTIFICATEUR ? ' sera NÉCESSAIRE pour l'exécution CORRECTE des remplacements CONDITIONNELS ( Voir CHAPITRE 3, PARTIE 8 )

REMARQUES :

Soit la chaîne SUJET '678abc123DEF123'

(\w+)?123, IDENTIQUE à l'expression RÉGULIÈRE \w*123, trouve TOUTE la chaîne SUJET

\w+?123 cherche la PLUS PETITE suite de caractères de MOT, NON NULLE, SUIVIE de la CHAÎNE '123' soit la SOUS-chaîne '678abc123' de la chaîne SUJET

La POSITION d'une ANCRE, SELON sa place à l'INTÉRIEUR ou à l'EXTÉRIEUR d'un groupe OPTIONNEL,

peut MODIFIER le comportement de l'expression RÉGULIÈRE

Soit les SIX lignes, ci-DESSOUS, constituées d'une TABULATION, SUIVIE d'un NOMBRE

de 1 à 5 CHIFFRES, éventuellement ABSENT :

```
	1
	12
	123
	1234
	12345
```

La RECHERCHE de \t(\d\d\d)?$ ne trouve QUE les CHAÎNES '	' et '	123', car elle

signifie 1 TABULATION, suivie de RIEN ou de 3 CHIFFRES, qui TERMINENT la ligne

La RECHERCHE de \t(\d\d\d$)? trouve une TABULATION '	' ou la CHAÎNE '	123', car

elle signifie 1 TABULATION, ÉVENTUELLEMENT suivie de 3 CHIFFRES, TERMINANT la ligne

J) DIFFÉRENCES entre QUANTIFICATEURS :

Soit la chaîne SUJET Liste : "chat","chien","tortue","hamster"... au DÉBUT de la 1ÈRE ligne :

la RECHERCHE ^.* trouve la CHAÎNE 'Liste : "chat","chien","tortue","hamster"...'

la RECHERCHE ^.*" trouve la CHAÎNE 'Liste : "chat","chien","tortue","hamster"'

la RECHERCHE ^.*", trouve la CHAÎNE 'Liste : "chat","chien","tortue",'

la RECHERCHE ^.*"\. trouve la CHAÎNE 'Liste : "chat","chien","tortue","hamster".'

Dans ces QUATRE recherches, on cherche, à PARTIR du DÉBUT de ligne, le MAXIMUM de caractères
en SOI ou jusqu'à la CHAÎNE '"' ou jusqu'à la CHAÎNE '",' ou jusqu'à la CHAÎNE '".'

la RECHERCHE ^.*? Ne trouve RIEN, car le MINIMUM de caractères en SOI est 0, soit la chaîne VIDE

la RECHERCHE ^.*?" trouve la CHAÎNE 'Liste : "'

la RECHERCHE ^.*?", trouve la CHAÎNE 'Liste : "chat",'

la RECHERCHE ^.*?"\. trouve la CHAÎNE 'Liste : "chat","chien","tortue","hamster".'

Dans ces QUATRE recherches, on cherche, à PARTIR du DÉBUT de ligne, le MINIMUM de caractères
en SOI ou jusqu'à la CHAÎNE '"' ou jusqu'à la CHAÎNE '",' ou jusqu'à la CHAÎNE '".'

la RECHERCHE ^.*+ trouve la CHAÎNE 'Liste : "chat","chien","tortue","hamster"...'

la RECHERCHE ^.*+" Ne trouve RIEN

la RECHERCHE ^.*+", Ne trouve RIEN

la RECHERCHE ^.*+"\. Ne trouve RIEN

Les TROIS DERNIÈRES recherches ne trouve JAMAIS RIEN, car ^.*+ consomme TOUJOURS le MAXIMUM
de CARACTÈRES de la LIGNE, y COMPRIS les signes GUILLEMET ("), VIRGULE (,) ou POINT (.)

Il ne reste donc RIEN pour correspondre au CARACTÈRE(S), placé(s) à la FIN de l'expression

RÉGULIÈRE. Comme, de PLUS, le QUANTIFICATEUR est POSSESSIF, AUCUN backtracking n'a lieu

=> AUCUNE solution !

Soit la chaîne SUJET 123456789abc1234abcd au DÉBUT de la 1ÈRE ligne d'un FICHIER :

la RECHERCHE \d+\w{5} trouve les DEUX chaînes '123456789abc12' puis '34abcd'

Pour la 2ÈME chaîne '34abcd' trouvée, il y a eu BACKTRACKING, car \d+ = '3' et \w{5} = '4abcd'

la RECHERCHE \d+?\w{5} trouve les TROIS chaînes '123456' puis '789abc' et enfin '1234ab'

AUCUN backtracking n'est NÉCESSAIRE pour obtenir ces TROIS concordances

Dans ces TROIS résultats, \d+? représente un SEUL chiffre et \w{5} les CINQ DERNIERS caractères

la RECHERCHE \d++\w{5} trouve UNIQUEMENT la chaîne '123456789abc12'

avec \d++ = '123456789' et \w{5} = 'abc12'

En effet, comme AUCUN backtracking n'est AUTORISÉ, du fait de la forme POSSESSIVE \d++ , la

SECONDE chaîne '34abcd' NE satisfait PAS l'expression RÉGULIÈRE

Les groupes ATOMIQUES - POSSESSIFS seront, de PRÉFÉRENCE, utilisés dans des expressions RÉGULIÈRES COMPLEXES :

- pour DIMINUER le nombre TOTAL de CAS à ANALYSER par le MOTEUR de RECHERCHE

- pour ÉVITER la formation de BOUCLES INFINIES

9) LOOKAROUNDS :

Les LOOKAROUNDS, ou ASSERTIONS utilisateur, sont des expressions RÉGULIÈRES, de longueur NULLE, ne consommant AUCUN caractère dans la chaîne SUJET à analyser, et représentant une CONDITION particulière, DEVANT être réalisée ou NON, AVANT ou à PARTIR de la position COURANTE du POINTEUR de RECHERCHE

TRÈS IMPORTANT :

Les expressions RÉGULIÈRES, placées dans les LOOKAROUNDS, ne font ABSOLUMENT PAS partie du GABARIT de la RECHERCHE, mais doivent être SATISFAITES, à la position COURANTE du POINTEUR, lors de leur ANALYSE

PENDANT la VÉRIFICATION des CONDITIONS, indiquées dans les LOOKAROUNDS, le POINTEUR de RECHERCHE n'est JAMAIS déplacé !

Il existe 4 FORMES de LOOKAROUND ( Test ) :

A) La FORME (?= .....) :

Cette ASSERTION, appelée POSITIVE LOOKAHEAD ( Test APRÈS POSITIF ) cherche, dans la chaîne SUJET, si l'expression RÉGULIÈRE, contenue dans le LOOKAHEAD, est SATISFAITE, à COMPTER de la position COURANTE du POINTEUR de RECHERCHE, SANS modifier cette POSITION

EXEMPLE :

q(?=u) cherche la lettre MINUSCULE 'q', lorsque celle-ci est SUIVIE de la lettre MINUSCULE 'u'

.(?=.) cherche TOUT caractère DIFFÉRENT de 'FIN de LIGNE', lorsqu'il est SUIVI d'un AUTRE caractère DIFFÉRENT de 'FIN de LIGNE' soit, globalement, TOUT caractère SAUF le DERNIER de CHAQUE ligne

IMPORTANT :

Les FORMES (?=u) et [u] n'ont PAS la MÊME signification !

La RegExp q[u] ou qu représente DEUX lettres MINUSCULES, lettre 'q' SUIVIE de la lettre 'u'

La RegExp q(?=u) ne représente QUE la lettre MINUSCULE 'q' lorsque CELLE-CI est

SUIVIE de la lettre MINUSCULE 'u'

Soit la chaîne SUJET 'Orques'

=> la RECHERCHE q[u] ou qu trouve les DEUX caractères 'qu'

=> la RECHERCHE q(?=u) trouve la lettre MINUSCULE 'q', UNIQUEMENT

REMARQUE :

La RegExp q(?=u)i NE peut PAS trouver le DÉBUT de la chaîne SUJET 'quittance', par exemple. En effet, au DÉBUT, une lettre MINUSCULE 'q' est PRÉSENTE et BIEN SUIVIE de la lettre MINUSCULE 'u'

Mais, au MOMENT de vérifier la PRÉSENCE de la MINUSCULE 'i', dans la chaîne SUJET, le POINTEUR ne se trouve que DEVANT la lettre 'u', car l'ASSERTION (?=u) n'a PAS consommé de CARACTÈRE

Comme cette lettre 'u' NE correspond PAS à la lettre MINUSCULE 'i' du GABARIT => AUCUNE solution

Par CONTRE, la RegExp q(?=u).i trouve, EFFECTIVEMENT, la CHAÎNE 'qui', car le caractère . du GABARIT, auquel correspond la lettre 'u' de la chaîne SUJET, fait AVANCER le POINTEUR de recherche SUR la lettre MINUSCULE 'i' !

De MÊME, la RegExp q.(?=i)i trouve, EFFECTIVEMENT, la CHAÎNE 'qui' du MOT 'quittance', car, APRÈS avoir trouvé la CHAÎNE 'qu' avec le DÉBUT de la RegExp q. , le POINTEUR de RECHERCHE

est SUR la lettre MINUSCULE 'i' et il est ÉVIDENT que la FIN de la RegExp (?=i)i est

SATISFAITE en présence de la lettre MINUSCULE 'i' de la chaîne SUJET 'quittance'

B) La FORME (?! .....) :

Cette ASSERTION, appelée NEGATIVE LOOKAHEAD ( Test APRÈS NÉGATIF ) cherche, dans la chaîne SUJET,

si l'expression RÉGULIÈRE, contenue dans le LOOKAHEAD, n'est PAS SATISFAITE, à COMPTER de la

position COURANTE du POINTEUR de RECHERCHE, SANS modifier cette POSITION

EXEMPLE :

q(?!u) cherche la lettre MINUSCULE 'q', lorsque celle-ci n'est PAS SUIVIE de la lettre MINUSCULE 'u'

.(?!.) cherche TOUT caractère DIFFÉRENT de 'FIN de LIGNE', lorsqu'il n'est PAS SUIVI d'un AUTRE

caractère DIFFÉRENT de 'FIN de LIGNE' soit, globalement, le DERNIER ou SEUL car. de CHAQUE ligne

IMPORTANT :

Les FORMES (?!u) et [^u] n'ont PAS la MÊME signification !

La RegExp q[^u] représente DEUX lettres, la MINUSCULE 'q' SUIVI d'UN caractère DIFFÉRENT de 'u'

La RegExp q(?!u) représente QUE la lettre MINUSCULE 'q', lorsque CELLE-CI n'est PAS SUIVIE

de la lettre MINUSCULE 'u'

Soit la chaîne SUJET 'Iraq' suivie de \r\n ( MOT Iraq en FIN de ligne )

=> la RECHERCHE q[^u] trouve la lettre MINUSCULE 'q' SUIVI de '\r'

=> la RECHERCHE q(?!u) trouve la lettre MINUSCULE 'q', UNIQUEMENT

C) La FORME (?<= .....) :

Cette ASSERTION, appelée POSITIVE LOOKBEHIND ( Test AVANT POSITIF ) cherche, dans la chaîne SUJET, si l'expression RÉGULIÈRE, contenue dans le LOOKBEHIND, est SATISFAITE, AVANT la position COURANTE du POINTEUR de RECHERCHE, SANS modifier cette POSITION

EXEMPLE :

(?<=q)u cherche la lettre MINUSCULE 'u', lorsque celle-ci est PRÉCÉDÉE de la lettre MINUSCULE 'q'

IMPORTANT :

Les FORMES (?<=q) et [q] n'ont PAS la MÊME signification !

La RegExp [q]u ou qu représente DEUX lettres MINUSCULES, 'u' PRÉCÉDÉE d'un 'q'

La RegExp (?<=q)u ne représente QUE la lettre MINUSCULE 'u' lorsque CELLE-CI est PRÉCÉDÉE de la lettre MINUSCULE 'q'

Soit la chaîne SUJET 'Orques'

=> la RECHERCHE [q]u ou qu trouve les DEUX caractères 'qu'

=> la RECHERCHE (?=q)u trouve la lettre MINUSCULE 'u', UNIQUEMENT

REMARQUE :

Dans CERTAINS cas, l'utilisation d'un LOOKBEHIND POSITIF ne donne PAS le résultat ESCOMPTÉ :

(?<=.). devrait TROUVER TOUT caractère DIFFÉRENT de 'FIN de LIGNE', quand il est PRÉCÉDÉ d'un AUTRE caractère DIFFÉRENT de 'FIN de LIGNE' soit, globalement, TOUT les caractères, SAUF le PREMIER de CHAQUE ligne

MAIS, suite au BUG de NOTEPAD++, ( voir PLUS HAUT ), on trouve, UN caractère sur DEUX !!

(?<=\n).* devrait TROUVER, tout comme la RegExp .* , une SUITE de caractères STANDARD, PRÉCÉDÉE de \n , soit globalement, CHAQUE ligne COMPLÈTE du fichier COURANT

MAIS, suite au BUG de NOTEPAD++, on trouve, en fait, UNE ligne sur DEUX !!

D) La FORME (?<! .....) :

Cette ASSERTION, appelée NEGATIVE LOOKBEHIND ( Test AVANT NÉGATIF ) cherche, dans la chaîne SUJET,
si l'expression RÉGULIÈRE, contenue dans le LOOKBEHIND, n'est PAS SATISFAITE, AVANT la
position COURANTE du POINTEUR de RECHERCHE, SANS modifier cette POSITION

EXEMPLE :

(?<!q)u cherche la lettre MINUSCULE 'u', lorsque celle-ci n'est PAS PRÉCÉDÉE de la MINUSCULE 'q'

IMPORTANT :

Les FORMES (?<!q) et [^q] n'ont PAS la MÊME signification !

La RegExp [^q]u représente 2 lettres, la MINUSCULE 'u', PRÉCÉDÉE d'UN caractère DIFFÉRENT de 'q'
La RegExp (?<!q)u représente QUE la lettre MINUSCULE 'u', lorsque CELLE-CI n'est PAS PRÉCÉDÉE
de la lettre MINUSCULE 'q'

Soit la chaîne SUJET 'futile'

=> la RECHERCHE [^q]u trouve la lettre MINUSCULE 'u' PRÉCÉDÉE de la MINUSCULE 'f'
=> la RECHERCHE (?<!q)u trouve la lettre MINUSCULE 'u', UNIQUEMENT

REMARQUE :

Dans CERTAINS cas, l'utilisation d'un LOOKBEHIND NÉGATIF ne donne PAS le résultat ESCOMPTÉ :

(?<!.). devrait TROUVER TOUT caractère DIFFÉRENT de 'FIN de LIGNE', quand il n'est PAS PRÉCÉDÉ d'un caractère DIFFÉRENT de 'FIN de LIGNE' soit, UNIQUEMENT, le PREMIER caractère de CHAQUE ligne

MAIS, suite au BUG de NOTEPAD++, ( voir PLUS HAUT ), on trouve, TOUS les caractères !!

(?<!\n).* devrait TROUVER une SUITE de caractères STANDARD, NON PRÉCÉDÉE de \n , soit globalement, la SUITE des caractères, à partir de la POSITION 2 , de CHAQUE ligne du fichier COURANT

MAIS, suite au BUG de NOTEPAD++, on trouve, en fait, TOUS les caractères !!

TRÈS IMPORTANT :

L'expression RÉGULIÈRE abc(?!def) cherche la CHAÎNE 'abc', NON SUIVIE de la CHAÎNE 'def'

MAIS le gabarit INVERSE (?!abc)def NE signifie PAS une CHAÎNE 'def', PRÉCÉDÉE de TROIS caractères DIFFÉRENTS de 'abc' !

En effet, lorsque le POINTEUR de RECHERCHE est situé JUSTE AVANT 'def', le LOOKAHEAD POSITIF (?!abc) est TOUJOURS VRAI car la CHAÎNE 'def' est TOUJOURS DIFFÉRENTE de 'abc'

=> TOUTES les occurrences de 'def' sont considérées JUSTES

Par CONTRE, le LOOKBEHIND (?<!abc)def permet EFFECTIVEMENT de trouver une CHAÎNE 'def', NON PRÉCÉDÉE de la CHAÎNE 'abc' !

NOTES DIVERSES :

Une expression RÉGULIÈRE peut être, SIMULTANÉMENT, PRÉCÉDÉE d'un LOOKBEHIND et SUIVIE d'un LOOKAHEAD

(?<!Mr )Guy(?! THE) trouve les CHAÎNES 'Guy', NON PRÉCÉDÉES de 'Mr ' et NON SUIVIES de ' THE'

Un lOOKAHEAD ( Test APRÈS ) peut aussi être placé AVANT l'expression RÉGULIÈRE à chercher !

(?=....THE)Guy trouve TOUTE chaîne 'Guy', si elle est SUIVIE de la CHAÎNE ' THE'

( Exemples PLUS utiles exposés PLUS LOIN ! )

Un LOOKBEHIND ( Test AVANT ) peut aussi être placé APRÈS l'expression RÉGULIÈRE à chercher !

Guy(?<=Mr....) trouve TOUTE chaîne 'Guy', si elle est PRÉCÉDÉE de la CHAÎNE 'Mr '

( Exemples PLUS utiles exposés PLUS LOIN ! )

Les LOOKAROUNDS sont des groupes ATOMIQUES, par DÉFAUT et ne forment PAS des GROUPES de CAPTURE

Aussi, si le CONTENU d'un LookAROUND doit être réutilisé, on placera un COUPLE de PARENTHÈSES supplémentaires, à l'INTÉRIEUR du LOOKAROUND. Cependant AUCUN BACKTRACKING n'aura lieu DANS le LOOKAROUND pour essayer d'AUTRES permutations, afin de satisfaire l'expression RÉGULIÈRE !

Soit la REGEXP : (?=(\d+))\w+\1 ( Donc \1 = \d+ )

Si la chaîne SUJET est '123x12', alors \1 = '123', mais le POINTEUR de recherche est TOUJOURS DEVANT la chaîne SUJET car \w+ = '123x12' puis revient en ARRIÈRE jusqu'au PREMIER '1', tentant, SANS succès de trouver '123', APRÈS \w+ et comme il n'y a AUCUN succès, non plus, en AVANÇANT le POINTEUR de RECHERCHE aux positions SUIVANTES => AUCUNE solution trouvée

Si le BACKTRACKING était possible dans le LOOKAROUND, on aurait eu, à la FIN du PROCESSUS :

\d+ = '12' , \w+ = '3x' et \1 = '12', comme avec la RegExp similaire (\d+)\w+\1

Cependant, si la chaîne SUJET est '456x56', le BACKTRACKING dans \d+ n'est PAS nécessaire.

En effet, \d+ = \1 = '456' et \w+ = '456x56' puis revient en ARRIÈRE jusqu'au PREMIER '4' tentant, SANS succès de trouver '456', APRÈS \w+, PUIS, le pointeur se DÉPLACE DEVANT le '5' => \1 = '56' et \w+ = '56x56' et, APRÈS retour en ARRIÈRE, on a bien :

\d+ = '56' , \w+ = 'x' et \1 = '56' => la CHAÎNE '56x56' est la SOLUTION !

Les LOOKAROUNDS peuvent être IMBRIQUÉS :

(?<=(?<!999)ABC)123 trouve TOUTE chaîne '123', PRÉCÉDÉE de la chaîne 'ABC', elle-même NON PRÉCÉDÉE de la chaîne '999'

123(?=ABC(?!999)) trouve TOUTE chaîne '123', SUIVIE de la chaîne 'ABC', elle-même NON SUIVIE de la chaîne '999'

Les LOOKAROUNDS peuvent contenir des ALTERNATIVES

(?<=000|99.)ABC trouve TOUTE chaîne 'ABC', si PRÉCÉDÉE de la chaîne '000' ou de la chaîne '99', SUIVIE d'UN caractère QUELCONQUE, DIFFÉRENT de \n, \f et \r

ABC(?=000|99|123.) trouve TOUTE chaîne 'ABC', SUIVIE d'UNE des 3 chaînes '000' , '99' ou de la chaîne '123', SUIVIE d'UN caractère QUELCONQUE, DIFFÉRENT de \n, \f et \r

REMARQUE :

CONTRAIREMENT au LOOKAHEADS, les ALTERNATIVES des LOOKBEHINDS doivent, OBLIGATOIREMENT, être de MÊME longueur

(?<=000|99)ABC => Message d'ERREUR 'Invalid regular expression'

L'expression RÉGULIÈRE, figurant dans un LOOKBEHIND, ne doit être COMPOSÉE QUE de sous-expressions de LONGUEUR FINIE

=> Les OPÉRATEURS * , + , ? , {n,} et {m,n} sont INTERDITS

IMPORTANT :

Un LOOKBEHIND POSITIF, de FORME (?<= RegExp), doit être, de préférence, REMPLACÉ par la FORME RegExp\K, qui permet des expressions de longueur NON FINIE, dans RegExp

En effet, UNE FOIS la RegExp ( avec POSSIBILITÉ de zones de longueur NON FINIE ) ÉVALUÉE, elle est IGNORÉE par le MOTEUR de RECHERCHE et SEULE la partie, placée APRÈS \K, constituera le GABARIT de la RECHERCHE ( Voir EXEMPLES ci-dessous )

EXEMPLES :

(?<=\d{3}..xyz)ABC cherche la CHAÎNE 'ABC', si celle-ci est PRÉCÉDÉE par une CHAÎNE composée de 3 CHIFFRES, suivis de DEUX caractères QUELCONQUES, DIFFÉRENTS de \n, \f et \r, suivis de la CHAÎNE 'xyz'

(?<=a{3})ABC cherche la CHAÎNE 'ABC', si celle-ci est PRÉCÉDÉE par la CHAÎNE 'aaa'

(?<=a){3}ABC cherche également la CHAÎNE 'aaaABC', bien que le QUANTIFICATEUR soit en DEHORS

du LOOKBEHIND

(?<=\d{3,}..xyz)ABC => Message d'ERREUR 'Invalid regular expression'

(?<=(?=.{0,3}abc).{6})def cherche la chaîne 'def', PRÉCÉDÉE d'une chaîne 'abc', placée DANS les SIX caractères PRÉCÉDANT la chaîne 'def'

La FORME : (?=.{0,3}abc).{6}\Kdef est cependant à PRÉFÉRER

A une POSITION p donnée, on VÉRIFIE si 'abc' ou .abc ou ..abc ou ...abc

Si OK, à partir de cette MÊME position p, on CHERCHE 6 caractères QUELCONQUES suivis de la CHAÎNE 'def', MAIS on ne CONSERVE QUE la partie APRÈS \K, c'est à dire 'def'

DEF\d+\Kabc cherche la CHAÎNE 'abc' séparée d'une chaîne 'DEF' ANTÉRIEURE par une SUITE QUELCONQUE de CHIFFRES, NON NULLE

SEULE la chaîne 'abc', placée APRÈS la forme \K, est CONSERVÉE en tant que GABARIT

( La forme : (?<=DEF\d+)abc est, en effet, NON AUTORISÉE ! )

Pour SUPPRIMER le mot 'foo', lorsque celui-ci est PRÉCÉDÉ d'au MOINS 3 CHIFFRES, TROIS formes

sont DONC possibles :

Partie RECHERCHE : (\d{3})foo Partie REMPLACEMENT : \1 ( vaut \d{3} )

Partie RECHERCHE : (?<=\d{3})foo Partie REMPLACEMENT : RIEN

Partie RECHERCHE : \d{3}\Kfoo Partie REMPLACEMENT : RIEN

La DERNIÈRE syntaxe, avec \K, est toutefois la plus SIMPLE

ATTENTION :

Utiliser UNIQUEMENT le bouton " Remplacer tout " pour les DEUX DERNIERS cas

Pour RECHERCHER la VALEUR de TOUS les CHAMPS 'name' de NativeLang.xml, SANS les GUILLEMETS :

name .+?"\K.+(?=") , avec 2 ESPACES, 1 AVANT et 1 APRÈS le mot 'name'

L'expression RÉGULIÈRE, figurant dans un LOOKBEHIND, ne doit PAS comporter de RÉFÉRENCES ARRIÈRE
ou de RÉFÉRENCES de GROUPE

EXEMPLE :

Si on cherche la PLUS LONGUE chaîne de 3 à 6 lettres MAJUSCULES, et, PAS PLUS de 2 LETTRES

CONSÉCUTIVES IDENTIQUES, avec le LOOKBEHIND ci-dessous

[A-Z][A-Z](?:([A-Z])(?<!\1\1\1)){1,4} => ERREUR = 'Invalid regular expression'

En fait, il faut utiliser un LOOKBEHIND, pour revenir du NOMBRE de caractères DÉSIRÉS,
puis placer un LOOKAHEAD, à l'INTÉRIEUR du LOOKBEHIND, qui vérifie la NON PRÉSENCE
de 3 lettres MAJUSCULES CONSÉCUTIVES IDENTIQUES, aux POSITIONS 1, ..., n-2, avec n
représentant la LONGUEUR de la CHAÎNE de lettres MAJUSCULES

[A-Z][A-Z](?:([A-Z])(?<=(?!\1\1\1)...)){1,4} ( CAS 1 )

ou

[A-Z][A-Z](?:(?<=(?!([A-Z])\1\1)..)[A-Z]){1,4} ( CAS 2 )

On peut aussi ÉVITER le LOOKBEHIND, avec la FORME :

(?:([A-Z])(?!\1\1)){1,4}[A-Z][A-Z] ( CAS 3 )

qui peut se traduire par :

De UNE à QUATRE lettres MAJUSCULES, CHACUNE d'elles n'étant PAS RÉPÉTÉE DEUX fois,
SUIVIE(S) par DEUX lettres MAJUSCULES

EXPLICATIONS du CAS 1 :

RAPPEL :

Soit la STRUCTURE générale ((Car)(Test sur Car)){1,n}

On considère la chaîne MAXIMUM de n CARACTÈRES ( Quantificateur GREEDY )

Le TEST se trouvant à l'INTÉRIEUR du groupe QUANTIFIÉ, il est VÉRIFIÉ à CHAQUE POSITION de la CHAÎNE ( 1, 2, 3... JUSQU'À n )

DÈS que le TEST n'est PLUS vérifié, à une POSITION x donnée :

- Si x > 1 la CHAÎNE des x - 1 PREMIERS caractères VÉRIFIE la STRUCTURE ci-dessus

  => CHAQUE POSITION, de 1 à x-1, de la RegExp DOIT satisfaire le TEST

- Si x = 1 le POINTEUR du MOTEUR de RECHERCHE est AVANCÉ d'une POSITION et on RELANCE la recherche de la STRUCTURE

  => La LONGUEUR de la chaîne MAXIMUM vaut, à présent, n - 1 CARACTÈRES

Soit la chaîne SUJET 'AAABBB', constituant l'UNIQUE ligne d'un FICHIER

- [A-Z][A-Z] correspond aux caractères en POSITION 1 et 2 soit 'AA'

- (?:([A-Z])Test){1,4} correspond aux caractères en POSITION 3, 4, 5, et 6, soit 'ABBB'

- Quand le POINTEUR est APRÈS la POSITION 3, le LOOKBEHIND (?<=...) déplace FICTIVEMENT
  le POINTEUR de TROIS caractères en ARRIÈRE, soit JUSTE AVANT le PREMIER caractère

- Le LOOKAHEAD, interne, (?!\1\1\1) VÉRIFIE que le 3ème CARACTÈRE (A) n'est PAS présent
  aux POSITIONS 1 et 2. Comme c'est le CAS, le TEST n'est donc PAS VÉRIFIÉ et la
  CHAÎNE 'ABBB', après les DEUX caractères 'AA', NE convient PAS

- Le POINTEUR de recherche AVANCE d'une POSITION => la chaîne SUJET devient 'AABBB'

  - [A-Z][A-Z] correspond aux caractères en POSITION 1 et 2 soit 'AA'
  - (?:([A-Z])Test){1,4} correspond aux caractères en POSITION 3, 4, et 5 soit 'BBB'
  - TEST si les 3 caractères, aux POSITIONS 1, 2 et 3, ne sont PAS TOUS ÉGAUX => OK
  - TEST si les 3 caractères, aux POSITIONS 2, 3 et 4, ne sont PAS TOUS ÉGAUX => OK
  - TEST si les 3 caractères, aux POSITIONS 3, 4 et 5, ne sont PAS TOUS ÉGAUX => KO

    => SEULS, les 2 PREMIERS caractères B conviennent pour (?:([A-Z])Test){1,4}

DONC, pour la chaîne SUJET 'AAABBB', la RegExp [A-Z][A-Z](?:([A-Z])(?<=(?!\1\1\1)...)){1,4}
trouve la chaîne totale 'AABB'

REMARQUES :

- C'est TOUJOURS l'ensemble ( LOOKAROUND(S) + CARACTÈRE ) qui est SUIVI du QUANTIFICATEUR,

et NON le caractère SEUL !

- On peut, aussi, dans ces 3 CAS, transformer le groupe NON CAPTURANT en un GROUPE de CAPTURE

il est alors NÉCESSAIRE d'utiliser la RÉFÉRENCE ARRIÈRE \2 ( au LIEU de \1 )

[A-Z][A-Z](([A-Z])(?<=(?!\2\2)...)){1,4}

[A-Z][A-Z]((?<=(?!([A-Z])\2\2)..)[A-Z]){1,4}

(([A-Z])(?!\2\2)){1,4}[A-Z][A-Z]

EXEMPLES GÉNÉRAUX :

^[^#\r\n]*\K#(?!.*#) trouve le SEUL caractère #, de TOUTE ligne d'un FICHIER, comportant 1 SEUL DIÈSE

ATTENTION :

La forme ^[^#\r\n]*\K#[^#\r\n]*$ est aussi CORRECTE, mais n'aurait PAS le MÊME effet, car TOUT

caractère, situé APRÈS le DIÈSE, serait, également SÉLECTIONNÉ

La forme ^.*?\K#(?!.*#), qui semble IDENTIQUE, est, par contre, INCORRECTE

En effet, l'idée INITIALE est : Trouve un 1ER DIÈSE d'une LIGNE, NON SUIVI d'un AUTRE DIÈSE,

soit les RegExp ^.*?\K# d'une part et (?!.*#) d'autre part

MAIS, en fait, cette RegExp signifie : Trouve un DIÈSE, NON SUIVI d'un AUTRE DIÈSE et SÉPARÉ du DÉBUT de ligne, par des caractères QUELCONQUES, AUTRES que 'FIN de LIGNE', DONT le DIÈSE !

=> La LIGNE peut donc contenir, CONTRAIREMENT à l'hypothèse, PLUSIEURS caractères '#' !

(?s)/*.*?*/ trouve TOUT texte de COMMENTAIRE d'un fichier, en langage 'C', y COMPRIS ses DEUX bornes '/*' et '*/', et ce :

- MÊME si un COMMENTAIRE ne contient AUCUN texte
- MÊME si PLUSIEURS commentaires SUCCESSIFS sur une MÊME ligne
- MÊME si un COMMENTAIRE s'étend sur PLUSIEURS lignes

NOTE :

Les commentaires NE doivent PAS être IMBRIQUÉS ( /*abc.../*123...789*/...xyz*/ )

abcd(?!.*abcd) trouve une CHAÎNE 'abcd' qui n'est PAS suivie d'une AUTRE CHAÎNE 'abcd'

=> Cette expression RÉGULIÈRE trouve donc la DERNIÈRE occurrence 'abcd' de CHAQUE ligne

NOTE : La RegExp .*abcd semble remplir la MÊME fonction, mais, en fait, elle sélectionne TOUT le texte, ENTRE le DÉBUT de la LIGNE et la DERNIÈRE chaîne 'abcd' de la LIGNE

(?<=tic)(?=tac) dans la partie RECHERCHE et la CHAÎNE '-' dans la partie REMPLACEMENT, AJOUTE un TIRET (-) entre les DEUX mots 'tic' et 'tac' de la CHAÎNE 'tictac'

NOTES :

Bien que le GABARIT, résultant des DEUX ASSERTIONS, est, en fait, une chaîne VIDE '', le MOTEUR de RECHERCHE parcoure, CARACTÈRE par CARACTÈRE, l'ensemble des POSITIONS de la chaîne SUJET !

Le REMPLACEMENT n'est EFFECTIF QUE si l'on clique sur le bouton " Remplacer tout " et NON sur bouton " Remplacer "

La RECHERCHE de TOUS les mots de 6 à 12 LETTRES, contenant l'UNE des CHAÎNES 'cat', 'dog' ou 'mouse' nécessite DEUX expressions RÉGULIÈRES :

\b\w{6,12}\b ( MOT de 6 à 12 LETTRES )

\b\w*(cat|dog|mouse)\w*\b ( MOT contenant l'UNE des CHAÎNES 'cat' , 'dog' ou 'mouse' )

L'utilisation d'un LOOKAHEAD POSITIF permet de COMBINER les DEUX RegExp

(?=\b\w{6,12}\b)\b\w*(cat|dog|mouse)\w*\b

Cette RegExp peut, néanmoins, être OPTIMISÉE :

- Les 3ème et 4ème ASSERTIONS \b sont INUTILES car les 2 PREMIÈRES déterminent DÉJÀ un MOT

  => (?=\b\w{6,12}\b)\w*(cat|dog|mouse)\w*

- La RECHERCHE de 'cat', 'dog' ... est INUTILE, APRÈS le 9ème caractère du mot ENGLOBANT

  => (?=\b\w{6,12}\b)\w{0,9}(cat|dog|mouse)\w*

- La 1ère ASSERTION \b, de longueur NULLE, peut être placée en DEHORS du LOOKAHEAD

  => Finalement, l'expression OPTIMISÉE est \b(?=\w{6,12}\b)\w{0,9}(cat|dog|mouse)\w*

L'expression RÉGULIÈRE ^$ cherche TOUTES les LIGNES VIDES d'un fichier

Si la partie REMPLACEMENT contient une CHAÎNE, celle-ci sera INSÉRÉE dans TOUTES les lignes VIDES

NOTE :

Bizarrement, si la partie REMPLACEMENT est VIDE, les lignes VIDES ne sont PAS SUPPRIMÉES !

SOLUTION : MARQUER les lignes VIDES lors de la RECHERCHE et utiliser la commande N++

" Supprimer les lignes marquées "

Soit ¤¤¤, représentant le GABARIT d'une expression RÉGULIÈRE, pouvant être une SIMPLE chaîne, à CHERCHER dans les LIGNES d'un fichier. Alors, si AUCUNE chaîne de REMPLACEMENT n'est indiquée :

(?=.*?¤¤¤).+\R? ou .*?¤¤¤.*\R? supprime TOUTES les LIGNES, satisfaisant le gabarit ¤¤¤

^(?!.*?¤¤¤).+\R? supprime TOUTES les lignes NON VIDES, NE satisfaisant PAS le gabarit ¤¤¤

NOTES :

Exemples de GABARIT ¤¤¤ possibles : abc , \d+ , [123XYZ] , a.*?z .....

\R? représente le, ou les, caractère(s) de 'FIN de LIGNE', éventuellement ABSENT

( CAS de la DERNIÈRE ligne d'un fichier )

Quand on TESTE l'ABSENCE du gabarit ¤¤¤, il faut AJOUTER l'ANCRE ^ pour BIEN chercher le gabarit DEPUIS le PREMIER caractère des LIGNES. En effet, si ¤¤¤ ne se trouve, par exemple, qu'UNE FOIS, au MILIEU d'une ligne, TOUTES les positions SUIVANTES satisfont le GABARIT (?!.*?¤¤¤), faisant croire, à TORT, que la ligne ne contient PAS le GABARIT ¤¤¤ !

Le GABARIT ^(?!.*?¤¤¤).*\R? censé représenter TOUTES les lignes VIDES ou NON, NE satisfaisant PAS le gabarit ¤¤¤, n'est PAS FIABLE, suite au BUG de \R ( Voir à caractères JOKER ) => Ce gabarit est à ÉVITER !

(?=.*?\bcat\b.*?\bdog\b.*?\bmouse\b).+ cherche TOUS les caractères d'une LIGNE, contenant les TROIS mots 'cat', 'dog' et 'mouse', dans cet ordre EXACT

(?=.*?\bcat\b)(?=.*?\bdog\b)(?=.*?\bmouse\b).+ cherche TOUS les caractères d'une LIGNE, contenant les TROIS mots 'cat', 'dog' et 'mouse', dans n'IMPORTE QUEL ordre, MÊME RÉPÉTÉS

^(?=.*?\bcat\b)(?!.*?\b(dog|mouse)\b).* cherche TOUTES les lignes avec le mot 'cat', au MOINS 1 FOIS, mais PAS le mot 'dog', NI le mot 'mouse'

DEF(?=.*999) trouve TOUTE chaîne 'DEF' si le RESTE de la ligne COURANTE contient la CHAÎNE '999'

DEF(?!.*999) trouve TOUTE chaîne 'DEF' si le RESTE de la ligne COURANTE NE contient PAS la CHAÎNE '999'

(?<=\d{3})(?<!999)abc cherche TOUTE chaîne 'abc', PRÉCÉDÉE de 3 CHIFFRES et, de plus, DIFFÉRENTS de '999'

(?<=\d{3}...)(?<!999)abc cherche TOUTE chaîne 'abc', PRÉCÉDÉE de 6 caractères, dont les TROIS PREMIERS

sont des CHIFFRES et les TROIS DERNIERS forme une chaîne DIFFÉRENTE de '999'

Si CHAQUE ligne d'un FICHIER comporte un BLOC de texte, ENTRE les SEULES BALISES 'DEB' et 'FIN',

QUELLE QUE SOIT la POSITION des BALISES, dans la LIGNE, comme ci-DESSOUS, alors :

```
DEB.............................FIN
......... DEB..........ABC............FIN............
DEB......................FIN..............
................................DEB...................FIN
DEB.......ABC.....................FIN
................................DEB........ABC.........FIN
DEB........ABC.............FIN..............
......... DEB.........................FIN............
```

.*?DEB(.*?ABC.*?)FIN.*  en partie RECHERCHE et \1 en partie REMPLACEMENT ( CAS 1 )

.*?DEB(((?!ABC).)*?)FIN.*  en partie RECHERCHE et \1 en partie REMPLACEMENT ( CAS 2 )

permet de REMPLACER CHAQUE ligne, par le SEUL texte, situé ENTRE les 2 BALISES, NON INCLUSES

- si ce BLOC de texte CONTIENT la CHAÎNE 'ABC' ( CAS 1 )
- si ce BLOC de texte NE contient PAS la chaîne 'ABC' ( CAS 2 )

NOTE :

Si, dans le Cas 2, la partie ((?!ABC).)*? est RÉÉCRITE sous la FORME (.(?!ABC))*?

LE RÉSULTAT est légèrement DIFFÉRENT ( CAS 3 )

En effet, si la CHAÎNE 'ABC' est placée JUSTE APRÈS la BALISE 'DEB', le MOTEUR de RECHERCHE considère, à TORT, la CHAÎNE '...DEBABC....FIN...' CORRECTE !

CAS 3 : APRÈS la POSITION du 'A' de 'ABC', il y a une CHAÎNE 'ABC' ? NON => OK

CAS 2 : APRÈS la POSITION du 'B' de 'DEB', il y a une CHAÎNE 'ABC' ? OUI => KO

(?<=AB|z{2}|[ \t].|\d\d|[[:cntrl:]]\u)Notepad++ cherche TOUTES les CHAÎNES 'Notepad++'

- qui sont PRÉCÉDÉES de la chaîne 'AB'
- ou qui sont PRÉCÉDÉES de 2 caractères 'z'
- ou qui sont PRÉCÉDÉES de 1 ESPACE ou TABULATION, suivie d'un caractère QUELCONQUE
- ou qui sont PRÉCÉDÉES de 2 CHIFFRES
- ou qui sont PRÉCÉDÉES de 1 caractère de CONTRÔLE, suivi d'1 lettre MAJUSCULE

RAPPEL : TOUTES les ALTERNATIVES d'un LOOKBEHIND sont de MÊME longueur ( Ici, 2 CARACTÈRES )

Soit les DIFFÉRENTS changements à effectuer quand un CHIFFRE C ou une LETTRE L sont SÉPARÉS d'un AUTRE chiffre C ou LETTRE L, par une ESPACE, une VIRGULE ou une COMBINAISON des DEUX,

( On suppose utiliser le format ANGLAIS des NOMBRES : 123,456,789.12 )

| | | | | | |
|---|---|---|---|---|---|
| | 2 | C,L | => | C, L | Insertion ESPACE, APRÈS la VIRGULE |
| | 2 | L,C | => | L, C | Insertion ESPACE, APRÈS la VIRGULE |
| | 2 | L,L | => | L, L | Insertion ESPACE, APRÈS la VIRGULE |
| | | C,C | => | | Ne RIEN faire |
| | | C L | => | C L | Ne RIEN faire |
| | | L C | => | L C | Ne RIEN faire |
| | | L L | => | L L | Ne RIEN faire |
| 1 | | C C | => | CC | Suppression ESPACE entre DEUX chiffres |
| | | C, L | => | C, L | Ne RIEN faire |
| | | L, C | => | L, C | Ne RIEN faire |
| | | L, L | => | L, L | Ne RIEN faire |
| 1 | | C, C | => | C,C | Suppression ESPACE, APRÈS la VIRGULE, dans un CHIFFRE |
| 1 | 2 | C ,L | => | C, L | Suppression ESPACE, AVANT VIRGULE puis Insertion ESPACE, APRÈS |
| 1 | 2 | L ,C | => | L, C | Suppression ESPACE, AVANT VIRGULE puis Insertion ESPACE, APRÈS |
| 1 | 2 | L ,L | => | L, L | Suppression ESPACE, AVANT VIRGULE puis Insertion ESPACE, APRÈS |
| 1 | | C ,C | => | C,C | Suppression ESPACE, AVANT VIRGULE, dans un CHIFFRE |

Ces 16 CAS possibles peuvent se RÉSUMER en DEUX propositions UNIQUEMENT :

- SUPPRIMER le caractère ESPACE si :

(((CHIFFRE ,) ou CHIFFRE ) AVANT ET CHIFFRE APRÈS ) OU si une VIRGULE APRÈS

=> RECHERCHE de l'expression RÉGULIÈRE \d,?\K (?=\d)| (?=,)

=> REMPLACEMENT par la chaîne VIDE

- INSERTION du caractère ESPACE, APRÈS la VIRGULE, si :

( PAS SUIVIE d'un ESPACE ) ET ( NON PRÉCÉDÉE OU NON SUIVIE) d'un CHIFFRE

=> RECHERCHE de l'expression RÉGULIÈRE ,(?! )((?<!\d,)|(?!\d))

=> REMPLACEMENT par LA CHAÎNE ', '

NOTES :

Dans la 1ÈRE opération de RECHERCHE-REMPLACEMENT, il y a DEUX recherches INDÉPENDANTES du caractère ESPACE :

- la PREMIÈRE avec la FORME \K et un LOOKAROUND POSITIF
- la SECONDE avec un LOOKAHEAD POSITIF

Dans la 2ÈME opération de RECHERCHE-REMPLACEMENT, il y a recherche UNIQUE d'une VIRGULE :

- qui SATISFAIT un LOOKAHEAD NÉGATIF

et

- qui SATISFAIT un des DEUX LOOKAROUNDS NÉGATIFS de l'ALTERNATIVE

REMARQUES :

La PREMIÈRE opération de RECHERCHE-REMPLACEMENT pourrait, en THÉORIE, s'écrire :

(?<=\d,?) (?=\d)| (?=,)

Cependant, comme la CHAÎNE '\d,?' n'a PAS UNE longueur FIXE, l'utilisation d'un

LOOKBEHIND n'est PAS autorisée => ERREUR : Invalid regular expression

MAIS, comme il s'agit d'un LOOKBEHIND POSITIF, on peut utiliser la forme

ÉQUIVALENTE \K , qui permet l'utilisation de CHAÎNES de longueur NON FIXE

La 2ÈME opération de RECHERCHE-REMPLACEMENT pourrait, également, s'écrire, en PERL :

```
(?<=,            #   APRÈS une VIRGULE,
   (?!           #     Si PAS :
      (?<=\d,)   #           un CHIFFRE, AVANT la VIRGULE
                 #       ( ET implicite )
      (?=\d)     #           un CHIFFRE  APRÈS la VIRGULE
   |             #       OU
      (?= )      #           un ESPACE   APRÈS la VIRGULE
   )
)
                 #     alors :
                 #
", "             #       Mettre un ESPACE, APRÈS CETTE VIRGULE
```

10) Blocs CONDITIONNELS :

Les Blocs CONDITIONNELS permettent la RECHERCHE de gabarit(s) DIFFÉRENTS, selon qu'une CONDITION est

RÉALISÉE, ou NON, à la position COURANTE du POINTEUR de recherche

Les DEUX principales SYNTAXES d'un Bloc CONDITIONNEL sont :

(?CONDITIONGabarit_si_VRAI)

(?CONDITIONGabarit_si_VRAI|Gabarit_si_FAUX)

dans lesquelles :

Gabarit_si_VRAI est le GABARIT à RECHERCHER, si la CONDITION à été PRÉCÉDEMMENT réalisée

Gabarit_si_FAUX est le GABARIT à RECHERCHER, si la CONDITION n'à PAS été PRÉCÉDEMMENT réalisée

CONDITION représente, au CHOIX :

- (N) = Nème GROUPE de CAPTURE, NON NOMMÉ, créé PRÉCÉDEMMENT
- (<Nom>) ou ('Nom') = GROUPE de CAPTURE, NOMMÉ 'Nom', créé PRÉCÉDEMMENT
- (?= .....) ou (?! .....) = ASSERTION utilisateur, de type LOOKAHEAD, POSITIF ou NÉGATIF
- (?<= .....) ou (?<!.....) = ASSERTION utilisateur, de type LOOKBEHIND, POSITIF ou NÉGATIF
- (R) = Référence RÉCURSIVE à TOUT le GABARIT ou à TOUT GROUPE de CAPTURE
- (Rn) = Référence RÉCURSIVE au SEUL groupe de CAPTURE, NON NOMMÉ, n
- (R&Nom) = Référence RÉCURSIVE au SEUL groupe de CAPTURE, NOMMÉ 'Nom'

Pour CHACUNE de ces SYNTAXES, la CONDITION est VRAIE si :

- le GABARIT, du GROUPE de CAPTURE indiqué, a été PRÉCÉDEMMENT réalisé
- le GABARIT, de l'ASSERTION utilisateur, a été PRÉCÉDEMMENT réalisé
- si un APPEL RÉCURSIF, au GABARIT de la référence RÉCURSIVE indiquée, a été PRÉCÉDEMMENT réalisé

La 3ème et dernière SYNTAXE de bloc CONDITIONNEL est :

(?(DEFINE)Gabarit_si_VRAI)

dans laquelle :

Gabarit_si_VRAI représente une SUITE de Groupes NOMMÉS, NE faisant PAS partie du GABARIT à CHERCHER

(DEFINE) représente une CONDITION qui est TOUJOURS FAUSSE ( => JAMAIS réalisée )

Cette SYNTAXE permet de DÉFINIR UN, ou PLUSIEURS, groupe(s) NOMMÉ(S) dans la partie Gabarit_si_VRAI de la structure CONDITIONNELLE, qui ne fait JAMAIS partie du GABARIT de RECHERCHE, suite à la valeur 'FAUX' de la CONDITION (DEFINE), mais qui permet la RÉUTILISATION du (de ces) groupe(s) NOMMÉ(S), par RÉFÉRENCE(S) de GROUPE, dans le GABARIT lui-même

REMARQUE :

Avec cette FORME, on peut donc DÉFINIR une BIBLIOTHÈQUE de groupes NOMMÉS, pouvant être repris dans de NOMBREUX gabarits DIFFÉRENTS

Dans NOTEPAD++, cependant, la CRÉATION d'une expression RÉGULIÈRE, sur PLUSIEURS lignes, étant IMPOSSIBLE, cette 3ème FORME est de PEU d'intérêt ! ( Voir EXEMPLE ci-après )

Le bloc CONDITIONNEL (DEFINE) est, en général, placé AVANT le GABARIT de RECHERCHE, mais ce n'est PAS OBLIGATOIRE

NOTES :

Si la CONDITION, de la structure CONDITIONNELLE, est un LOOKAHEAD, le MOTEUR de RECHERCHE testera la partie Gabarit_si_VRAI ou la partie Gabarit_si_FAUX, à la MÊME position que le TEST du LOOKAHEAD, constituant la CONDITION

Les structures CONDITIONNELLES ne constituent PAS, en soi, de GROUPES de CAPTURE. Donc, si les expressions RÉGULIÈRES Gabarit_si_VRAI et/ou Gabarit_si_FAUX sont désirées, il faudra placer une PAIRE de PARENTHÈSES autour de la PARTIE concernée ou de la structure CONDITIONNELLE toute ENTIÈRE

Les DEUX syntaxes deviennent alors :

(?CONDITION(Gabarit_si_VRAI)) ou ((?CONDITIONGabarit_si_VRAI))

(?CONDITION(Gabarit_si_VRAI)|Gabarit_si_FAUX)

ou

(?CONDITIONGabarit_si_VRAI|(Gabarit_si_FAUX))

ou

((?CONDITIONGabarit_si_VRAI|Gabarit_si_FAUX))

Si la partie Gabarit_si_VRAI et/ou la partie Gabarit_si_FAUX contiennent des ALTERNATIVES, placer LA ( ou les DEUX ) partie(s) concernées, ENTRE PARENTHÈSES

Les DEUX syntaxes deviennent alors :

(?CONDITION(Gabarit_1|Gabarit_2||Gabarit_3|.....))

(?CONDITION(Gabarit_1|Gabarit_2||Gabarit_3|.....)|(Gabarit_4|Gabarit_5|Gabarit_6|.....))

IMPORTANT :

Pour un fonctionnement CORRECT des structures CONDITIONNELLES, qui font RÉFÉRENCE à un GROUPE de CAPTURE, NOMMÉ ou NON, PRÉCÉDEMMENT défini, les 3 SYNTAXES (n), (<Nom>) et ('Nom') DEVRONT concerner un groupe de CAPTURE NON OPTIONNEL

En effet, un groupe OPTIONNEL est TOUJOURS considéré 'VRAI' par une structure CONDITIONNELLE,

puisqu'il signifie : Groupe EXISTANT et VÉRIFIÉ ou groupe NON EXISTANT !!

SOLUTION :

Transformer le groupe OPTIONNEL en un groupe NON OPTIONNEL, mis entre PARENTHÈSES,

et SUIVI du QUANTIFICATEUR ?

Dans ce cas, si le groupe NON OPTIONNEL :

- est trouvé => le groupe est DÉFINI et la CONDITION est VRAIE
- n'est PAS trouvé => le groupe n'est PAS DÉFINI et la CONDITION est FAUSSE

EXEMPLES :

La forme NON OPTIONNELLE (*{0,5}) sera RÉÉCRITE sous la FORME (*{1,5})?

Dans ce CAS, la CONDITION (1) est bien :

- VRAIE, si une CHAÎNE de 1 à 5 ASTÉRISQUES, au PLUS, est trouvée
- FAUSSE, si AUCUN caractère ASTÉRISQUE n'est trouvé

NOTE :

La FORME, presque SIMILAIRE, (*){0,5}, est OPTIONNELLE, mais le GROUPE 1 vaut *, qui

est bien une forme NON OPTIONNELLE

Dans ce CAS, la CONDITION (1) signifie :

- VRAIE, si 1 ASTÉRISQUE, au MOINS, a pu être trouvée
- FAUSSE, si AUCUN caractère ASTÉRISQUE n'a pu être trouvé

La forme NON OPTIONNELLE (\w*) sera RÉÉCRITE sous la FORME (\w+)?

Dans ce CAS, la CONDITION (1) est bien :

- VRAIE, si UN, ou PLUSIEURS, caractères de MOT ont pu être trouvés
- FAUSSE, si AUCUN caractère de MOT n'a pu être trouvé

EXEMPLES de structures CONDITIONNELLE :

(?(?=.*[a-z])\d{2}-[a-z]{3}-\d{2}|\d{2}-\d{2}-\d{2}) cherche :

- une SUITE de 2 CHIFFRES, puis 3 lettres MINUSCULES, et, enfin, 2 CHIFFRES, séparés par des TIRETS,
  SI la ligne COURANTE comporte AU MOINS 1 lettre MINUSCULE

- une SUITE de 2 CHIFFRES, répétée TROIS fois, et séparée par des TIRETS

SI la ligne COURANTE NE contient PAS de lettre MINUSCULE

Dans cette structure CONDITIONNELLE :

- la CONDITION est un LOOKAHEAD POSITIF (?=.*[a-z])
- la partie Gabarit_si_VRAI est \d{2}-[a-z]{3}-\d{2} ( Forme JJ-mmm-AA )
- la partie Gabarit_si_FAUX est \d{2}-\d{2}-\d{2} ( Forme JJ-MM-AA )

^(?<Guy>\()?[^()\r\n]+(?(<Guy>)\)|[^()])$ ( avec groupe NOMMÉ < Guy>, OPTIONNEL )

^(\()?[^()\r\n]+(?(1)\)|[^()])$ ( avec groupe NON NOMMÉ, OPTIONNEL )

trouve l'INTÉGRALITÉ des LIGNES d'un fichier :

- sans PARENTHÈSE
- avec DEUX PARENTHÈSES, UNIQUEMENT, en DÉBUT ( APRÈS \n ) et FIN de ligne ( AVANT \r )

Dans cette structure CONDITIONNELLE :

- la CONDITION est une RÉFÉRENCE au GROUPE (<Guy>) ou (1) , soit la RegExp (\()
- la partie Gabarit_si_VRAI est \) PARENTHÈSE, à la FIN
- la partie Gabarit_si_FAUX est [^()] Caractère DIFFÉRENT de PARENTHÈSE, à la FIN

NOTE :

SANS structure CONDITIONNELLE, le GABARIT peut s'écrire avec l'ALTERNATIVE :

^([^()\r\n]+|\([^()\r\n]+\))$, qui recherche, ENTRE \n et \r :

- soit une SUITE de caractères NON PARENTHÈSE, et AUTRES que les DEUX caractères \n et \r
- soit une SUITE de caractères NON PARENTHÈSE, et AUTRES que les DEUX caractères \r et \n ,
  INCLUSE dans un COUPLE de PARENTHÈSES

^(\w+)(\w+)?(?(2)\g2\g1|\g1)$ trouve TOUS les mots d'une LISTE, à raison de UN mot par LIGNE, dont la STRUCTURE correspond à l'UNE des DEUX formes GLOBALES 'XYYX' ou 'XX', comme ci-DESSOUS :

'toto' , 'aBBa' , '55' , 'THEVENOT_Guy_GuyTHEVENOT' , 'couscous' , 'BeririBe' , etc.

Dans cette structure CONDITIONNELLE :

- la CONDITION est une RÉFÉRENCE au GROUPE 2 (2) , soit la 2ème RegExp \w+
- la partie Gabarit_si_VRAI est \g2\g1 ( Forme YX )
- la partie Gabarit_si_FAUX est \g1 ( Forme X )

NOTE :

SANS structure CONDITIONNELLE, la RegExp peut s'écrire avec un groupe NON CAPTURANT de 2 ALTERNATIVES

^(?|(\w+)(\w+)\2\1|(\w+)\1)$ , qui recherche, ENTRE \n et \r,

- (\w+)(\w+)\2\1 c'est à dire une FORME de type 'XYYX'
- (\w+)\1 c'est à dire une FORME de type 'XX'

^((From|To)|Subject): ((?(2)\w+@\w+\.[a-z]+|.+)) cherche :

- l'UNE des CHAÎNES 'From', 'To' ou 'Subject' en DÉBUT de ligne, SUIVIE de ': '
- une adresse E-MAIL VALIDE de l'EXPÉDITEUR ou du DESTINATAIRE ( Forme '~~~~~~~@~~~~~~~~.~~~' )
- enfin, le SUJET du MAIL ( Suite NON VIDE de caractères, DIFFÉRENTS de \n, \f et \r )

Dans cette structure CONDITIONNELLE :

- la CONDITION est une RÉFÉRENCE au GROUPE 2 (2) soit la RegExp From|To
- la partie Gabarit_si_VRAI est \w+@\w+\.[a-z]+ ( Adresse E-MAIL )
- la partie Gabarit_si_FAUX est .+ ( Texte NON VIDE )

NOTES :

L'ENTÊTE est mémorisée dans le GROUPE 1 ( UNE des TROIS chaînes 'From' , 'To' ou 'Subject' )

La structure CONDITIONNELLE est ENTOURÉE de PARENTHÈSES pour mémoriser, dans le GROUPE 3 :

- soit l'adresse E-MAIL de l'EXPÉDITEUR ou du DESTINATAIRE \w+@\w+\.[a-z]+

- soit le SUJET du MAIL .+

^(a)?(?(1)a|b)+$ cherche les chaînes SUJET suivantes, à raison d'UNE par LIGNE :

- une SUITE de lettres 'a' MINUSCULES, si une lettre MINUSCULE 'a' EXISTE en DÉBUT la LIGNE
- une SUITE de lettres 'b' MINUSCULES, si PAS de lettre 'a' MINUSCULE en DÉBUT de LIGNE

Dans cette structure CONDITIONNELLE :

- la CONDITION est une RÉFÉRENCE au GROUPE 1 (1) soit 'a'
- la partie Gabarit_si_VRAI est 'a'
- la partie Gabarit_si_FAUX est 'b'

(?(DEFINE)(?<Octet>25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d))\b(?&Octet)(\.(?&Octet)){3}\b trouve TOUTE adresse IPv4

Cette expression RÉGULIÈRE est composée de DEUX parties :

- un bloc CONDITIONNEL (?(DEFINE)(?<Octet>25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d))

  qui ne fait PAS PARTIE du GABARIT, car la CONDITION (DEFINE) est TOUJOURS FAUSSE

  mais sert UNIQUEMENT à DÉFINIR le groupe de CAPTURE 'Octet' pour utilisation ULTÉRIEURE

- Le GABARIT de RECHERCHE \b(?&Octet)(\.(?&Octet)){3}\b

NOTES :

- L'emploi de groupes NOMMÉS, avec la structure CONDITIONNELLE (DEFINE), n'est PAS OBLIGATOIRE :

  => La FORME (?(DEFINE)(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d))\b(?1)(\.(?1)){3}\b est POSSIBLE

- Le GABARIT de RECHERCHE peut être placé DEVANT la structure CONDITIONNELLE (?(DEFINE).....)

  => La FORME \b(?2)(\.(?2)){3}\b(?(DEFINE)(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)) est POSSIBLE

  REMARQUE :

  Le PREMIER groupe, commençant par \. , la RÉFÉRENCE de GROUPE doit être (?2),

  correspondant au DEUXIÈME groupe 25[0-5].....[1-9]?\d

- L'emploi d'une RÉFÉRENCE ARRIÈRE, à la place d'une RÉFÉRENCE de GROUPE, est IMPOSSIBLE avec la structure CONDITIONNELLE (?(DEFINE).....)

\b(?&N1)\.(?&N2)\.(?&N3)\.(?&N4)\b(?(DEFINE)(?<N1>25[0-5])(?<N2>2[0-4]\d)(?<N3>1\d\d)(?<N4>[1-9]?\d))

ou

\b(?1)\.(?2)\.(?3)\.(?4)\b(?(DEFINE)(25[0-5])(2[0-4]\d)(1\d\d)([1-9]?\d)) ( SANS groupe NOMMÉ )

trouve TOUTE adresse IPv4, tels que :

```
250  <=  1er  Octet  <=  255
200  <=  2ème octet  <=  249
100  <=  3ème Octet  <=  199
0    <=  4ème Octet  <=  99   ( SANS 0 NON significatif )
```

11) PRIORITÉS des OPÉRATEURS :

Les différents OPÉRATEURS, AUTRES que les caractères LITTÉRAUX et le JOKER . , sont listés ci-DESSOUS, par ordre de PRIORITÉ DÉCROISSANTE, de la PLUS GRANDE à la PLUS PETITE :

| | |
|---|---|
| 1 - les CLASSES de type 'POSIX' | [: :] , [. .] , [= =] |
| 2 - le symbole d'ÉCHAPPEMENT | \ |
| 3 - les LISTES entre CROCHETS | [ ] , [^ ] |
| 4 - les RÉFÉRENCES ARRIÈRE | \g ou \k SUIVIE de N , { } , < > ou ' ' |
| 4 - les RÉFÉRENCES de GROUPE | (?N) , (?& ) , (?P> ) |
| 5 - le GROUPEMENT entre PARENTHÈSES | ( ) , (? ) |
| 6 - les symboles de DUPLICATION | * , + , ? , {n} , {n,} , {n,m} |
| 7 - la CONCATÉNATION IMPLICITE | |
| 8 - les symboles d'ANCRAGE | ^ , $ |

9 - le symbole d'ALTERNATION |

Exemples de PRÉCÉDENCE entre 2 OPÉRATEURS ADJACENTS :

Entre 1 et 2 [[=\=]] représente le caractère ANTISLASH ( et NON la RegExp [[==]] qui est INVALIDE )

( la recherche d'un CROCHET OUVRANT ou FERMANT ou du signe ÉGAL s'obtient par la RegExp []=[] )

Entre 2 et 3 [\]0-9] représente un CROCHET FERMANT ou un CHIFFRE ( et NON un ANTISLASH suivi de la CHAÎNE '0-9]'

qui peut être trouvé avec l'expression RÉGULIÈRE \\0-9] )

Entre 3 et 4 (a)[\g1] représente UNE des 2 CHAÎNES 'ag' ou 'a1', ( et NON la CHAÎNE 'aa' : GROUPE CAPTURÉ +

RÉFÉRENCE ARRIÈRE, qui peut être trouvé avec l'expression RÉGULIÈRE (a)\g1 )

Entre 4 et 5 (A)\g({1}) n'est PAS VALIDE, mais l'expression RÉGULIÈRE (A)(\g{1}) est CORRECTE ( la chaîne

LITTÉRALE 'A\g{1}' peut être trouvée par l'expression RÉGULIÈRE A\\g\{1\} )

Entre 5 et 6 (abc)+ représente la CHAÎNE 'abcabc....abc' ( et NON la CHAÎNE 'ab', SUIVIE d'une SUITE NON NULLE de

caractères 'c', qui peut être trouvé avec l'expression RÉGULIÈRE abc+ )

Entre 6 et 7 abc+ représente la CHAÎNE 'ab' SUIVIE d'une SUITE de lettres 'c' ( et NON la CHAÎNE 'abcabc...abc' qui peut être trouvée avec l'expression RÉGULIÈRE (abc)+ )

Entre 7 et 8 ^a?b représente l'UNE des CHAÎNES 'ab' ou 'b' en DÉBUT de ligne ( et NON la CHAÎNE 'ab' en DÉBUT de ligne ou la CHAÎNE 'b' à un endroit QUELCONQUE, qui peut être trouvé par l'UNE des DEUX expression RÉGULIÈRES (^a)?b ou ^ab|b )

Entre 8 et 9 ^ab|cd$ représente 'ab' en DÉBUT ou 'cd' en FIN de ligne ( et NON les LIGNES formées des DEUX caractères 'ab' ou 'cd', qui peuvent être trouvées avec l'expression RÉGULIÈRE ^(ab|cd)$ )

12) MEILLEURE CONCORDANCE d'une expression RÉGULIÈRE :

En POSITION p de la chaîne SUJET, La MEILLEURE CONCORDANCE d'une expression RÉGULIÈRE est la PREMIÈRE CONCORDANCE trouvée, la PLUS à GAUCHE ( la PLUS PROCHE de la POSITION p COURANTE, dans la chaîne SUJET ), constituée de la MEILLEURE CONCORDANCE d'UN ou de PLUSIEURS des SOUS-ensembles du TABLEAU ci-après :

NOTES :

Les symboles RE, RE1 et RE2 représentent des expressions RÉGULIÈRES QUELCONQUES

L'ENTITÉ 'Cond' représente la CONDITION d'une STRUCTURE CONDITIONNELLE et peut-être, au CHOIX :

- un LOOKAHEAD ou un LOOKBEHIND, POSITIF ou NÉGATIF

- un GROUPE de CAPTURE, NOMMÉ ou NON

- une RÉFÉRENCE RÉCURSIVE, NOMMÉE ou NON

| FORME | La MEILLEURE CONCORDANCE de la FORME vaut : |
|---|---|
| RE1RE2 | La MEILLEURE concordance pour RE1 qui peut être SUIVIE d'UNE concordance de RE2 |
| RE1\|RE2 | La MEILLEURE concordance pour RE1, MÊME si UNE concordance de RE2 EXISTE à la POSITION p |
| RE1\|RE2 | La MEILLEURE concordance pour RE2, SI AUCUNE concordance de RE1 n'EXISTE à la POSITION p |
| RE{n} | EXACTEMENT n FOIS, la MEILLEURE concordance de RE |
| RE{n,} | AU MOINS n FOIS, la MEILLEURE concordance de RE et le MAXIMUM de FOIS possible |
| RE{n,m} | ENTRE n et m FOIS, la MEILLEURE concordance de RE et le MAXIMUM de FOIS possible |
| RE? | ENTRE 0 et 1 FOIS, la MEILLEURE concordance de RE et le MAXIMUM de FOIS possible |
| RE* | AU MOINS 0 FOIS, la MEILLEURE concordance de RE et le MAXIMUM de FOIS possible |
| RE+ | AU MOINS 1 FOIS, la MEILLEURE concordance de RE et le MAXIMUM de FOIS possible |

| | |
|---|---|
| RE{n,}? | AU MOINS n FOIS, la MEILLEURE concordance de RE et le MINIMUM de FOIS nécessaire |
| RE{n,m}? | ENTRE n et m FOIS, la MEILLEURE concordance de RE et le MINIMUM de FOIS nécessaire |
| RE?? | ENTRE 0 et 1 FOIS, la MEILLEURE concordance de RE et le MINIMUM de FOIS nécessaire |
| RE*? | AU MOINS 0 FOIS, la MEILLEURE concordance de RE et le MINIMUM de FOIS nécessaire |
| RE+? (1) | AU MOINS 1 FOIS, la MEILLEURE concordance de RE et le MINIMUM de FOIS nécessaire |

| | |
|---|---|
| (?>RE) (2) | TOUJOURS la MEILLEURE concordance de RE, SANS essai SUPPLÉMENTAIRE de CONCORDANCE pour RE<br>( AUCUN BACKTRACKING n'a lieu dans RE, pour obtenir une CONCORDANCE de la RegExp COMPOSÉE ) |

| | |
|---|---|
| (?=RE) , (?=(RE)) | TOUJOURS la MEILLEURE concordance de RE, à COMPTER de la POSITION p, dans la chaîne SUJET |
| (?<=RE) , (?<=(RE)) | TOUJOURS la MEILLEURE concordance de RE, AVANT la POSITION p, dans la chaîne SUJET |

| | |
|---|---|
| (?!RE) | L'ABSENCE de concordance de RE, à COMPTER de la POSITION p, dans la chaîne SUJET |
| (?<!RE) | L'ABSENCE de concordance de RE, AVANT la POSITION p, dans la chaîne SUJET |

| | |
|---|---|
| (?(Cond)RE) | La MEILLEURE concordance de RE, si la CONDITION Cond est VRAIE, à la POSITION p du SUJET |
| (?(Cond)RE1\|RE2) | La MEILLEURE concordance de RE1, si la CONDITION Cond est VRAIE, à la POSITION p du SUJET<br>La MEILLEURE concordance de RE2, si la CONDITION Cond est FAUSSE, à la POSITION p du SUJET |

EXEMPLES :

(1) BIEN voir la DIFFÉRENCE entre, par EXEMPLE, les DEUX expressions RÉGULIÈRES 'y+?' et 'y+?z'

avec la chaîne SUJET 'yyyyyzzz'

Dans le PREMIER cas : SEULE, UNE lettre MINUSCULE y est sélectionnée ( Minimum SUFFISANT )

Dans le SECOND cas : la CHAÎNE 'yyyyyz' est sélectionnée ( Minimum NÉCESSAIRE )

(2) Bien voir la DIFFÉRENCE entre, par EXEMPLE, les 2 expressions RÉGULIÈRES 'a+ab' et '(?>a+)ab',

avec la chaîne SUJET 'aaabbb'

Dans le PREMIER cas : la CHAÎNE 'aaab' est trouvée car a+ représente 'aa' ( PAS le MAXIMUM )

Dans le SECOND cas : AUCUNE concordance est trouvée, car AUCUN BACKTRACKING sur la PARTIE a+

REMARQUE :

Le MOTEUR de RECHERCHE utilise, par DÉFAUT, les Expressions RÉGULIÈRES Compatibles PERL ( PCRE )

Cependant, si le MOTEUR de RECHERCHE utilisait les Expressions RÉGULIÈRES Étendues POSIX

( POSIX ERE ), la MEILLEURE CONCORDANCE, pour une expression RÉGULIÈRE, serait alors

DÉTERMINÉE selon l'ALGORITHME, communément appelé ' The Leftmost-Longest Rule ',

PLUS COMPLEXE, qui NE sera PAS explicité dans le cadre de ce MANUEL

13) Exemples SUPPLÉMENTAIRES :

(^|(?<= ))(http://[\w.]+?)((?= )|$) dans la partie RECHERCHE

<a href="\2">\2</a> dans la partie REMPLACEMENT

cherche de la CHAÎNE 'http://', SUIVIE d'une SUITE, NON NULLE de caractères de MOT ou du caractère POINT,

- qui DÉBUTE une ligne ou est PRÉCÉDÉE d'une ESPACE

et

- qui TERMINE une ligne ou est SUIVIE d'une ESPACE

et la REMPLACE par le code HTML qui la change en un HYPERLIEN cliquable, comme indiqué ci-DESSOUS :

http://site.com devient <a href="http://site.com">http://site.com</a>

REMARQUE :

Ne PAS utiliser le BOUTON " Remplacer " ( remplacement PAS à PAS ) mais, UNIQUEMENT, le BOUTON " Remplacer tout " ( remplacement GLOBAL )

Ce BUG de Notepad++, survient lorsque la chaîne de RECHERCHE contient des LOOKAHEADS ou des LOOKBEHINDS POSITIFS, c'est à dire les FORMES (?=.....) et (?<=.....)

( Voir le CHAPITRE III, relative à la SYNTAXE des RegExp en zone de REMPLACEMENT )

IMPORTANT :

Ne PAS utiliser les DEUX formes [ \n] et [ \r], à la place des 2 ASSERTIONS (^|(?<= )) et ((?= )|$)

Suite à la MAUVAISE gestion des caractères 'FIN de LIGNE' de Notepad++, l'expression RÉGULIÈRE [\n ]Ch[ \r] NE trouve PAS, en effet, TOUTES les occurrences de la CHAÎNE Ch, CONTRAIREMENT à la FORME (^|(?<= ))(Ch)((?= )|$) , qui recense bien TOUS les CAS !

L'expression RÉGULIÈRE ^ ?(Ch) ?$ recense, également, TOUS les CAS, mais les ESPACES éventuels, situés AVANT et APRÈS la CHAÎNE Ch, font partie INTÊGRANTE des RÉSULTATS

NOTE :

On peut, aussi, utiliser un groupe NON CAPTURANT d'ALTERNATIVES qui montre les 4 CAS désirés :

(?| (http://[\w.]+) |^(http://[\w.]+) | (http://[\w.]+)$|^(http://[\w.]+)$) en partie RECHERCHE

<a href="\1">\1</a> en partie REMPLACEMENT ( Grâce au BLOC (?|..., CHACUN des cas = Groupe 1 )

On peut, également, utiliser un structure CONDITIONNELLE de type (DEFINE), qui MÉMORISE le GABARIT http://[\w.]+ en tant que GROUPE 1. Celui-ci est repris, par RÉFÉRENCE de GROUPE dans le bloc NON CAPTURANT d'ALTERNATIVES qui le suit :

(?(DEFINE)(http://[\w.]+))(?| ((?1)) |^((?1)) | ((?1))$|^((?1))$)  en partie RECHERCHE

<a href="\2">\2</a>  en partie REMPLACEMENT ( Groupe 2 est ÉGAL au BLOC () entourant (?1) )

(?i)<(\h*[a-z0-9_-]*\h*>)(\R|\h)*<\h*/(?1) en partie RECHERCHE, et chaîne VIDE en partie REMPLACEMENT, SUPPRIME

les blocs de TAGS HTML OUVRANTS-FERMANTS VIDES, c'est à dire JUXTAPOSÉS ou séparés UNIQUEMENT par :

- des caractères BLANCS HORIZONTAUX : ESPACE, TABULATION ou \xA0

ou

- des caractères 'FIN de LIGNE' ou assimilés  : \r\n , \r , \n , \x0B , \f  et  \x85

Chaque NOM, ÉVENTUEL, de bloc, entre les BALISES < et >, est UNIQUEMENT composé de LETTRES, de CHIFFRES

et des DEUX caractères TIRET (-) et (_), éventuellement PRÉCÉDÉS et/ou SUIVIS de BLANCS HORIZONTAUX

EXEMPLE :

La chaîne SUJET  '123< td >    < / TD >456'  est TRANSFORMÉE en  '123456'

NOTE :

Pour SUPPRIMER des BLOCS IMBRIQUÉS, on exécutera PLUSIEURS fois le cycle de RECHERCHE-REMPLACEMENT

(?x) (  (  (  [^(,)\r\n]  )*  \(  (  [^()\r\n]  |  (?2)  )*  \)  (?3)*  )  |  (?3)+  )  ,?  en partie RECHERCHE

et \r\n$1 en partie REMPLACEMENT,

CHANGE, dans des lignes, comportant des PARENTHÈSES IMBRIQUÉES, TOUTE VIRGULE du PLUS HAUT niveau ( soit HORS PARENTHÈSES ) par un PASSAGE à la ligne SUIVANTE

EXEMPLE :

a,b,(c,d)f2,op3(e,(op5(h,op6(k,l)f6)f5,g)f4)f3,op7(op 8(t,u),s)f8,de,op9(y,z),op10(g,op11(t,e)h)

est MODIFIÉ en :

```
a
b
(c,d)f2
op3(e,(op5(h,op6(k,l)f6)f5,g)f4)f3
op7(op 8(t,u),s)f8
de
op9(y,z)
op10(g,op11(t,e)h)
```

NOTES :

l'OPTION (?x) permet, pour une MEILLEURE visibilité, de SÉPARER les DIFFÉRENTS blocs d'une expression RÉGULIÈRE, par des ESPACES, qui NE font PAS partie du GABARIT de RECHERCHE

On peut DÉCOMPOSER le GABARIT de RECHERCHE en : (?x) ( Gabarit_1 | Gabarit_2 ) ,? =

Gabarit_1 , représentant le Groupe 2, :

TOUTE suite, même NULLE, de car. <> de VIRGULE, PARENTHÈSE, \r et \n = ([^(,)\r\n])*

\+ 1 PARENTHÈSE OUVRANTE = \(

\+ 1 Car. <> de PARENTHÈSE, \r, \n OU RÉCURSION à Groupe 2, POSSIBLE = ( [^()\r\n] | (?2) )*

\+ 1 PARENTHÈSE FERMANTE = \)

\+ TOUTE suite, même NULLE, de car. <> de VIRGULE, PARENTHÈSE, \r et \n = (?3)*

OU

Gabarit_2 :

Toute SUITE, NON NULLE, de car. <> de VIRGULE, PARENTHÈSE, \r et \n = (?3)+

SUIVI d'une VIRGULE ÉVENTUELLE = ,?

Dans ce GABARIT, on a :

Groupe 3 = [^(,)\r\n] = (?3) = TOUT caractère DIFFÉRENT de VIRGULE, PARENTHÈSE, \r et \n

Groupe 2 est utilisé comme RÉFÉRENCE RÉCURSIVE (?2), située DANS le GROUPE 2, LUI-MÊME

Groupe 1 = $1 = Gabarit CHERCHÉ, SANS VIRGULE possible FINALE, à réécrire en REMPLACEMENT

REMARQUE :

Avec un LANGAGE de PROGRAMMATION, cette DÉCOMPOSITION est FACILE à obtenir :

RÉSULTAT en MS-DOS Qbasic, version 1.0, par exemple :

```
l$ = "a,b,(c,d)f2,op3(e,(op5(h,op6(k,l)f6)f5,g)f4)f3"
l$ = l$ + ",op7(op 8(t,u),s)f8,de,op9(y,z),op10(g,op11(t,e)h)"
CLS : p = 0
DO
  l$ = MID$(l$, p + 1)
  n = 0
  FOR p = 1 TO LEN(l$)
    c$ = MID$(l$, p, 1)
    IF c$ = "(" THEN n = n + 1
    IF c$ = ")" THEN n = n - 1
    IF c$ = "," AND n = 0 THEN
      PRINT MID$(l$, 1, p - 1)
      EXIT FOR
    END IF
  NEXT
LOOP WHILE p <= LEN(l$)
PRINT l$
```

Si (?sx) (. (?<!DEB) (?<!FIN) (?<! ¤) ) * DEB ( ( (?! DEB).) *? | (?R) ) FIN ( (?! DEB) (?! FIN ) (?! ¤) . ) *

en partie RECHERCHE, et ' ¤¤¤¤¤<$2>\r\n', SANS les GUILLEMETS SIMPLES, en partie REMPLACEMENT :

- EXTRACTION de TOUT le TEXTE, COMPRIS entre 2 mots BALISES, NON INCLUS, PRÉCÉDÉ de la CHAÎNE ' ¤¤¤¤¤',
  ENTOURÉ par les DEUX caractères '<' et '>' et enfin SUIVI de \r\n

- SUPPRESSION de TOUT texte, SANS balise NON APPARIÉE, situé en DEHORS des 2 BALISES, de NIVEAU 0 ACTUEL

EXEMPLE :

123 DEB Je vais DEB faire un FIN essai
de texte
pour voir FIN 456 Je pense que DEB DEB ca marche FIN bien ! FIN 99
DernièreDEB ligne Youpi, c'estFIN OK !!!

est MODIFIÉ, après le PREMIER remplacement GLOBAL en :

¤¤¤¤¤¤< Je vais DEB faire un FIN essai
de texte
pour voir >
¤¤¤¤¤¤< DEB ca marche FIN bien ! >
¤¤¤¤¤¤< ligne Youpi, c'est>

est MODIFIÉ, après le DEUXIÈME remplacement GLOBAL en :

¤¤¤¤¤ ¤¤¤¤¤¤< faire un >
¤¤¤¤¤ ¤¤¤¤¤¤< ca marche >
¤¤¤¤¤¤< ligne Youpi, c'est>

NOTES :

Le texte EXTRAIT peut, ÉVENTUELLEMENT contenir d'AUTRES blocs, avec les DEUX mots BALISES, mais, UNIQUEMENT, si ceux-ci sont APPARIÉS, à CHAQUE niveau d'IMBRICATION ! ( Utilisation de la RÉCURSIVITÉ )

Le texte EXTRAIT peut être RÉPARTI sur PLUSIEURS lignes

Si on utilise le BOUTON " Remplacer tout " ( Remplacement GLOBAL ), RÉ-EXÉCUTER N fois le cycle de RECHERCHE-REMPLACEMENT, N étant la PROFONDEUR d'IMBRICATION des BLOCS de texte 'DEB ..... FIN'

Le NOMBRE de CHAÎNES '¤¤¤¤¤' - 1 donne le niveau d'IMBRICATION du TEXTE qui suit, à partir du NIVEAU 0

L'OPTION (?s) signifie que le MÉTA-caractère . = TOUT caractère, y COMPRIS \r , \n ou \f

Voir l'exemple PRÉCÉDENT, concernant l'OPTION (?x)

La RÉFÉRENCE RÉCURSIVE (?R) se rapporte au GABARIT ENTIER de la RECHERCHE

^((?:.*?(RegExp1)){N-1}.*?)\2 en partie RECHERCHE et \1RegExp2 en partie REMPLACEMENT, permet de REMPLACER TOUTES les Nèmes OCCURRENCES de l'expression RÉGULIÈRE 'RegExp1', trouvées dans CHAQUE ligne du FICHIER, par l'expression RÉGULIÈRE 'RegExp2'

NOTE :

Pour SUPPRIMER TOUTES les Nèmes OCCURRENCES de l'expression RÉGULIÈRE 'RegExp1', mettre UNIQUEMENT \1 en partie REMPLACEMENT

EXEMPLE :

Pour SUPPRIMER le 24ème POINT-VIRGULE (;) de CHAQUE ligne d'un FICHIER, mettre :

^((?:.*?;){23}.*?); en partie RECHERCHE et \1 en partie REMPLACEMENT

JUSTESSE d'une expression, MÊME VIDE et/ou RÉPARTIE sur PLUSIEURS lignes, comprise ENTRE les DEUX SÉPARATEURS

'<' et '>', avec un nombre ARBITRAIRE, MÊME NUL, de COUPLES des DEUX signes '<' et '>' APPARIÉS :

Soit l'expression RÉGULIÈRE suivante : <(?:(?(R)(?:[\d\r\n])++|[^<>]*+)|(?R))*> , comportant TROIS parties :

- DEUX signes, INFÉRIEUR '<' et SUPÉRIEUR '>', BORNANT l'expression RÉGULIÈRE, et comportant :

- Soit une structure CONDITIONNELLE (?(R) .... ) avec :

  - un bloc Gabarit_si_VRAI ATOMIQUE (?:[\d\r\n])++ s'il y a DÉJÀ eu une RÉCURSION

  - un bloc Gabarit_si_FAUX ATOMIQUE [^<>]*+ s'il n'y a PAS ENCORE eu de RÉCURSION ( niveau SUPÉRIEUR )

- Soit la référence RÉCURSIVE (?R) à la TOTALITÉ de l'expression RÉGULIÈRE, sur les caractères trouvés,

  eux-mêmes ENTRE DEUX signes INFÉRIEUR '<' et SUPÉRIEUR '>'

Elle signifie, globalement :

TANT qu'il n'y a PAS RÉCURSION, on recherche, ENTRE les signes '<' et '>', BORNANT la RegExp, un ensemble

ATOMIQUE, MÊME NUL, :

- de caractères, DIFFÉRENTS des DEUX signes INFÉRIEUR et SUPÉRIEUR '<' et '>'

ou

- de caractères, eux-mêmes ENTRE DEUX signes '<' et '>', satisfaisant l'expression RÉGULIÈRE ENTIÈRE, suite à la référence RÉCURSIVE (?R) à la TOTALITÉ de la RegExp sur cette SOUS-CHAÎNE

DÈS qu'il y a RÉCURSION, on recherche :

- un ensemble ATOMIQUE, NON NUL, de caractères CHIFFRES ( [0-9¹²³] ) , de \r ou de \n

ou

- un ensemble ATOMIQUE, MÊME NUL, eux-mêmes ENTRE DEUX signes '<' et '>', satisfaisant l'expression RÉGULIÈRE ENTIÈRE, suite à la référence RÉCURSIVE (?R) à la TOTALITÉ de la RegExp sur cette SOUS-CHAÎNE

Ainsi, cette expression RÉGULIÈRE recherche et sélectionne le PLUS GRAND ensemble de caractères, ENTRE DEUX signes INFÉRIEUR '<' et SUPÉRIEUR '>' comprenant, ÉVENTUELLEMENT, des caractères INFÉRIEUR et SUPÉRIEUR, correctement APPARIÉS

EXEMPLE :

Soit la chaîne SUJET : Mr<<123<456>78>904>0Guy0<<54<6>4>>THE<12345>VENOT<>, alors les DIFFÉRENTES SÉLECTIONS sont :

Si curseur AVANT le M de 'Mr' => <<123<456>78>904> , <<54<6>4>> , <12345> et <>

Si curseur ENTRE les DEUX PREMIERS signes '<' => <123<456>78> , <<54<6>4>> , <12345> et <>

Si curseur APRÈS le 1ER CHIFFRE 1 => <456> , <<54<6>4>> , <12345> et <>

EXEMPLE :

Soit la chaîne MODIFIÉE Mr<<1A3<456>78>904>0Guy0<<54<6>4>>THE<12345> ( LETTRE A, au lieu du CHIFFRE 2 )

Si curseur AVANT le M de 'Mr', la 1ÈRE recherche sélectionne <1A3<456>78> ( et NON <<1A3<456>78>904> ! )

En effet, dans la chaîne '<1A3<456>78>' la LETTRE A, située au niveau SUPÉRIEUR, HORS RÉCURSION, est

est CORRECTE ( caractère BIEN DIFFÉRENT de '<' et de '>' ) => Cette chaîne est SÉLECTIONNÉE

Par CONTRE, dans la chaîne ENGLOBANTE '<<1A3<456>78>904>', la LETTRE A, située au 1ER niveau d'IMBRI-

CATION ( <<.A. ), PENDANT la RÉCURSION, n'est PAS CORRECTE, car DIFFÉRENTE d'un CHIFFRE, de \r et \n

=> Cette chaîne n'est donc PAS SÉLECTIONNÉE

EXEMPLE :

Soit la chaîne MODIFIÉE Mr<<1A3<4B6>78>904>0Guy0<<54<6>4>>THE<12345> ( LETTRES A et B, au lieu de 2 et 5 )

Si curseur AVANT le M de 'Mr', la 1ÈRE recherche sélectionne <4B6> ( et NON les CHAÎNES <1A3<4B6>78> et

<<1A3<4B6>78>904> )

Pour la MÊME raison, le chaîne <4B6>, contenant QUE LETTRE B, au niveau SUPÉRIEUR, HORS RÉCURSION, est donc SÉLECTIONNÉE,

tandis que les DEUX chaînes suivantes ne sont PAS SÉLECTIONNÉES :

- '<1A3<4B6>78>', car contenant la LETTRE B au 1ER niveau d'IMBRICATION, PENDANT la RÉCURSION
- <<1A3<4B6>78>904>, car contenant la LETTRE A au 1ER niveau d'IMBRICATION, PENDANT la RÉCURSION
  et contenant la LETTRE B au 2ÈME niveau d'IMBRICATION, PENDANT la RÉCURSION

L'expression RÉGULIÈRE \(((?>[^()]+)|(?R))*\) recherche et sélectionne le PLUS GRAND ensemble de caractères, MÊME VIDE, entre DEUX PARENTHÈSES, OUVRANTE et FERMANTE, contenant un nombre ARBITRAIRE, MÊME NUL, de COUPLES de PARENTHÈSES APPARIÉS

L'expression RÉGULIÈRE ([^()]*\((?R)\))*[^()]* recherche et sélectionne, à PARTIR de la POSITION du CURSEUR, le PREMIER PLUS GRAND ensemble de caractères, contenant un nombre ARBITRAIRE, MÊME NUL, de COUPLES de PARENTHÈSES APPARIÉS

14) IMPORTANCE de la FORME du texte SUJET analysé et des DONNÉES désirées par l'UTILISATEUR :

Ne JAMAIS oublier que la FORME de l'expression RÉGULIÈRE de RECHERCHE dépend, de manière IMPORTANTE, des DONNÉES du ( ou des ) FICHIER(S) parcourus et de la FORME des RÉSULTATS, attendus par l'UTILISATEUR

EXEMPLES :

Soit la RECHERCHE d'un NOMBRE, SÉPARÉ d'un MOT ALPHABÉTIQUE, AVEC ou SANS caractères ACCENTUÉS, par UN ou PLUSIEURS caractères BLANC, SÉPARATEUR de ligne ou NON ( \t , \n , \x0B , \f , \r , \x20 ou \xA0 )

La RegExp \d+\s+[\l\u]+ permet une TELLE recherche GLOBALE

mais \d+\s+\l+ est PRÉFÉRABLE si le fichier NE contient PAS de lettres MAJUSCULES

mais \d+\s+\[a-z]+ est PRÉFÉRABLE si le MOT ne DOIT contenir QUE des MINUSCULES, NON ACCENTUÉES

mais \d+\h+\l+ est PRÉFÉRABLE si MOT et NOMBRE, sont SÉPARÉS QUE par des ESPACES ou TABULATIONS

mais \d+ +\l+ est PRÉFÉRABLE si MOT et NOMBRE, sont SÉPARÉS, QUE par UNE ou PLUSIEURS ESPACES

mais \d+ \l+ est PRÉFÉRABLE si MOT et NOMBRE, sont SÉPARÉS, QUE par UNE ESPACE

Soit la RECHERCHE d'un caractère QUELCONQUE, qui est REDOUBLÉ

La RegExp (.)\1 permet une TELLE recherche GLOBALE

mais ([^ \t])\1 est PRÉFÉRABLE pour ÉVITER les CHAÎNES de 2 ESPACES ou de 2 TABULATIONS

mais (\w)\1 est PRÉFÉRABLE pour AFFICHER 2 CHIFFRES ou LETTRES IDENTIQUES ou 2 TIRETS BAS (_)

mais ([\u\l])\1 est PRÉFÉRABLE pour AFFICHER 2 LETTRES IDENTIQUES, ACCENTUÉES ou NON

mais (\u)\1 est PRÉFÉRABLE pour AFFICHER 2 lettres MAJUSCULES IDENTIQUES, ACCENTUÉES ou NON

mais ([A-Z])\1 est PRÉFÉRABLE pour AFFICHER 2 LETTRES MAJUSCULES IDENTIQUES, NON ACCENTUÉES

15) NOTE concernant la RÉCURSIVITÉ :

Le PRINCIPE de RÉCURSIVITÉ, ( présence d'une RÉFÉRENCE de GROUPE, à l'INTÉRIEUR du GROUPE qu'elle REPRÉSENTE ),
a été évoqué, dans ce MANUEL, à travers QUELQUES exemples, mais mériterait des APPROFONDISSEMENTS, qui ont,
ACTUELLEMENT, été négligés, vu la COMPLEXITÉ à définir des exemples CONCRETS et des explications CLAIRES

Après INVESTIGATIONS, cette TECHNIQUE fera l'objet d'un AJOUT ultérieur à ce présent DESCRIPTIF !

III) SYNTAXE des expressions RÉGULIÈRES PCRE en partie REMPLACEMENT :

1) RAPPELS IMPORTANTS :

Pour exécuter des REMPLACEMENTS, utiliser, UNE des DEUX méthodes en CLIQUANT :

- DEUX fois, ou PLUS, sur le BOUTON " Remplacer ", de l'ONGLET " Remplacer " ( Méthode PAS à PAS )

- UNE SEULE fois sur le BOUTON " Remplacer tout ", de l'ONGLET " Remplacer " ( Méthode GLOBALE )

Si la CASE " Boucler " de la fenêtre " Rechercher ", accessible par ' CTRL + F ', n'est PAS COCHÉE, les RECHERCHES, REMPLACEMENTS et COMPTAGES porteront sur l'INTÉGRALITÉ du fichier COURANT, en cliquant sur :

- le BOUTON " Rechercher dans le document actuel ", de l'ONGLET " Rechercher "
- le BOUTON " Compter ", de l'ONGLET " Rechercher "
- le BOUTON " Trouver tout " ou " Remplacer tout ", de l'ONGLET " Rechercher dans les fichiers d'un dossier "

EXCEPTIONS :

Les REMPLACEMENTS ou les MARQUAGES se feront DEPUIS la position ACTUELLE du CURSEUR, jusqu'à la TOUTE FIN du fichier COURANT, en cliquant sur :

- le BOUTON " Remplacer tout ", de l'ONGLET " Remplacer "
- le BOUTON " Rechercher tout ", de l'ONGLET " Marquer "

Si, le TYPE de RECHERCHE choisi n'est PAS le mode " Expression régulière ", et que la DIRECTION indiquée est " Haut ", les REMPLACEMENTS ou les MARQUAGES se feront DEPUIS la position ACTUELLE du CURSEUR, jusqu'au TOUT DÉBUT du fichier COURANT, en cliquant sur :

- le BOUTON " Remplacer tout ", de l'ONGLET " Remplacer "

- le BOUTON " Rechercher tout ", de l'ONGLET " Marquer "

Si la zone de RECHERCHE comporte des LOOKAROUNDS POSITIFS, NE PAS utiliser la méthode PAS à PAS, lors du REMPLACEMENT ( Bouton " Remplacer " )

=> Cliquer EXCLUSIVEMENT sur le bouton " Remplacer tout " ( Méthode GLOBALE )

Si on utilise la boîte de RECHERCHE-REMPLACEMENT ( CTRL + R ) du PLUGIN " TextFX ", avec la CASE " Regular Expr " COCHÉE, NE PAS utiliser, de manière GÉNÉRALE, la méthode PAS à PAS ( BOUTON " Repl&FAgain " ), mais :

- Cliquer une PREMIÈRE fois sur le bouton " Find " ( Recherche de la 1ÈRE OCCURRENCE )
- Cliquer, ENSUITE, sur le bouton " Replace Rest " ( Méthode GLOBALE )

2) CARACTÈRES :

TOUT caractère d'une expression RÉGULIÈRE, en partie REMPLACEMENT, est apparié à LUI-MÊME, SAUF :

- le caractère PARENTHÈSE OUVRANTE ( ( )
- le caractère PARENTHÈSE FERMANTE ( ) )
- le caractère ANTISLASH ( \ )
- le caractère DOLLAR ( $ )
- le caractère POINT d'INTERROGATION ( ? )
- le caractère DEUX-POINTS ( : )

Ces 6 CARACTÈRES, appelés MÉTA-caractères, ont une signification SPÉCIALE, expliquée par la SUITE

REMARQUE :

Par rapport aux MÉTA-caractères de RECHERCHE, SEUL, le MÉTA-caractère DEUX-POINTS (:) est NOUVEAU

Un caractère UNIQUE, de REMPLACEMENT, peut être, au CHOIX :

A) Un caractère LITTÉRAL, NON MÉTA-caractère :

TOUT caractère, DIFFÉRENT des MÉTA-caractères indiqués CI-DESSUS, se représente LUI-MÊME

EXEMPLE :

L'expression RÉGULIÈRE 'A3@;zé', en partie REMPLACEMENT, remplace l'expression RÉGULIÈRE cherchée par la chaîne LITTÉRALE 'A3@;zé'

REMARQUES :

Bien que NON nécessaire, un caractère, NON MÉTA-caractère, peut, également, être PRÉCÉDÉ du caractère ANTISLASH ( \ )

EXEMPLE :

la FORME \@ ( ou @ ) équivaut au caractère LITTÉRAL 'AROBASE' ( @ ), qui est SUBSTITUÉ

à l'expression RÉGULIÈRE cherchée

Les formes '\ + LETTRE', ci-dessous, NE participent PAS à la SYNTAXE des expressions RÉGULIÈRES, en partie REMPLACEMENT, et sont IDENTIQUES au caractère LETTRE, indiqué APRÈS le caractère ANTISLASH

\b , \c , \d , \g , \h , \i , \j , \k , \m , \o , \p , \q , \s ,
\w , \x , \y , \z , \A , \B , \C , \D , \F , \G , \H , \I , \J ,
\K , \M , \N , \O , \P , \Q , \R , \S , \T , \V , \W , \X , \Y et \Z

EXEMPLE :

La FORME \Q ( ou Q ), en zone REMPLACEMENT, équivaut à la lettre MAJUSCULE Q, qui sera SUBSTITUÉE à l'expression RÉGULIÈRE recherchée

REMARQUE :

Une FORME '\+ LETTRE', qui n'a AUCUNE signification SPÉCIALE, en partie REMPLACEMENT peut être SIGNIFICATIVE en partie RECHERCHE

La forme \Q, ci -dessus, signifie DÉBUT d'une séquence d'ÉCHAPPEMENT, en partie RECHERCHE

B) Un MÉTA-caractère LITTÉRAL :

Il est constitué du caractère ANTISLASH ( \ ), SUIVI d'un MÉTA-caractère et représente le MÉTA-caractère LITTÉRAL, SANS sa signification SPÉCIALE dans la syntaxe PCRE, en partie REMPLACEMENT

EXEMPLES :

la FORME \$ équivaut au caractère LITTÉRAL 'DOLLAR' ( $ )

la FORME \\ équivaut au caractère LITTÉRAL 'ANTISLASH' ( \ )

C) Un caractère en NOTATION HEXADÉCIMALE ou OCTALE :

```
\x00   , \x0       = le caractère de code HEXA     =    00  ( caractère Ascii NUL )
\x{00} , \x{0000}  = le caractère de code HEXA     =    00  ( caractère Ascii NUL )

\xN                = le caractère de code HEXA     =    0N   avec N = [0-9A-Fa-f]
\xNN               = le caractère de code HEXA     =    NN   avec N = [0-9A-Fa-f]
\x{NN}             = le caractère de code HEXA     =    NN   avec N = [0-9A-Fa-f]
\x{NNNN}           = le caractère de code UNICODE  =  NNNN   avec N = [0-9A-Fa-f] , si fichier codé UTF-8

\0   , \00         = le caractère de code OCTAL    =   000  ( caractère Ascii NUL )
\000 , \0000       = le caractère de code OCTAL    =   000  ( caractère Ascii NUL )

\0N                = le caractère de code OCTAL    =   00N   avec N = [0-7]
\0NN               = le caractère de code OCTAL    =   0NN   avec N = [0-7]
\0MNN              = le caractère de code OCTAL    =   MNN   avec N = [0-7] et M = [0-3]
```

EXEMPLES et REMARQUES :

```
\011          représente le caractère de CONTRÔLE 'TAB' ( 11 OCTAL = 9 DÉCIMAL )
\0113         représente la lettre 'K' ( 113 OCTAL = 75 DÉCIMAL )
\011\063      représente le caractère de CONTRÔLE 'TAB', suivi du CHIFFRE '3'

\01456        représente la CHAÎNE 'e6' ( 145 OCTAL = 101 DÉCIMAL = 'e' )
\014\065\x36  représente le caractère de CONTRÔLE 'FF' suivi de la CHAÎNE '56'

\x010         représente le caractère de CONTRÔLE 'SOH', suivi du CHIFFRE '0'
\x1AA         représente le caractère de CONTRÔLE 'SUB', suivi de la lettre MAJUSCULE 'A'
\x{263A}      représente le caractère ÉMO-ICÔNE :) ( Visage SOURIANT )
\x{20ac}      représente le caractère MONÉTAIRE EURO ( € )
```

BUG :

TOUT caractère, dans la zone de REMPLACEMENT, situé APRÈS un ou PLUSIEURS caractère 'NUL', de code ASCII 0, QUELLE QUE SOIT sa REPRÉSENTATION ( \0 , \c@ , \x{00} , .... ), n'est PAS écrit, comme ATTENDU, lors du remplacement EFFECTIF de l'expression RÉGULIÈRE cherchée, MAIS produit des caractères, SANS rapport APPARENT avec les caractères SAISIS

SOLUTION :

- REMPLACER l'expression CHERCHÉE par une chaîne de REMPLACEMENT, SANS caractère NULL
- REMPLACER le (ou les) caractères CONCERNÉS par un (ou des) caractère(s) 'NUL' (\x00), éventuellement PRÉCÉDÉ(S), mais PAS SUIVI(S), d'AUTRES caractères

D) Un caractère d'ÉCHAPPEMENT :

```
\r    =   \x0D  (CR)   =   Caractère CARRIAGE RETURN     ( Caractère SÉPARATEUR de LIGNE )
\a    =   \x07  (BEL)  =   Caractère BELL                ( Caractère ALERTE )
\t    =   \x09  (TAB)  =   Caractère TABULATION          ( Caractère BLANC HORIZONTAL )
\e    =   \x1B  (ESC)  =   Caractère ESCAPE              ( Caractère ÉCHAPPEMENT )
\n    =   \x0A  (LF)   =   Caractère LINE FEED           ( Caractère SÉPARATEUR de LIGNE )
\v    =   \x0B  (VT)   =   Caractère VERTICAL TABULATION ( Caractère TABULATION VERTICALE )
\f    =   \x0C  (FF)   =   Caractère FORM FEED           ( Caractère SÉPARATEUR de LIGNE )
```

EXEMPLE :

\r\n , en partie REMPLACEMENT, SUPPRIME l'expression RÉGULIÈRE cherchée et la REMPLACE par une 'FIN de LIGNE' Windows

E) Un caractère d'ÉCHAPPEMENT ASCII :

```
\c@   =   \c`   =   \x00  (  Caractère de CONTRÔLE  'NUL'   =   Caractère NULL              )

\cA   =   \ca   =   \x01  (  Caractère de CONTRÔLE  'SOH'   =   START of HEADER           )
\cB   =   \cb   =   \x02  (  Caractère de CONTRÔLE  'STX'   =   START of TEXT             )
\cC   =   \cc   =   \x03  (  Caractère de CONTRÔLE  'ETX'   =   END   of TEXT             )
\cD   =   \cd   =   \x04  (  Caractère de CONTRÔLE  'EOT'   =   END   of TRANSMISSION     )
\cE   =   \ce   =   \x05  (  Caractère de CONTRÔLE  'ENQ'   =   ENQUIREMENT              )
\cF   =   \cf   =   \x06  (  Caractère de CONTRÔLE  'ACK'   =   ACKNOWLEDGEMENT          )
\cG   =   \cg   =   \x07  (  Caractère de CONTRÔLE  'BEL'   =   BELL                     )
\cH   =   \ch   =   \x08  (  Caractère de CONTRÔLE  'BS'    =   BACK SPACE              )
\cI   =   \ci   =   \x09  (  Caractère de CONTRÔLE  'TAB'   =   HORIZONTAL TABULATION    )
\cJ   =   \cj   =   \x0A  (  Caractère de CONTRÔLE  'LF'    =   LINE FEED               )
\cK   =   \ck   =   \x0B  (  Caractère de CONTRÔLE  'VT'    =   VERTICAL   TABULATION   )
```

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| \cL | = | \cl | = | \x0C | ( | Caractère de CONTRÔLE | 'FF' | = | FORM FEED | ) |
| \cM | = | \cm | = | \x0D | ( | Caractère de CONTRÔLE | 'CR' | = | CARRIAGE RETURN | ) |
| \cN | = | \cn | = | \x0E | ( | Caractère de CONTRÔLE | 'SO' | = | SHIFT IN | ) |
| \cO | = | \co | = | \x0F | ( | Caractère de CONTRÔLE | 'SI' | = | SHIFT OUT | ) |
| \cP | = | \cp | = | \x10 | ( | Caractère de CONTRÔLE | 'DLE' | = | DELETE | ) |
| \cQ | = | \cq | = | \x11 | ( | Caractère de CONTRÔLE | 'DC1' | = | DEVICE CONTROL 1 | ) |
| \cR | = | \cr | = | \x12 | ( | Caractère de CONTRÔLE | 'DC2' | = | DEVICE CONTROL 2 | ) |
| \cS | = | \cs | = | \x13 | ( | Caractère de CONTRÔLE | 'DC3' | = | DEVICE CONTROL 3 | ) |
| \cT | = | \ct | = | \x14 | ( | Caractère de CONTRÔLE | 'DC4' | = | DEVICE CONTROL 4 | ) |
| \cU | = | \cu | = | \x15 | ( | Caractère de CONTRÔLE | 'NAK' | = | NEGATIVE ACKNOWLEDGEMENT | ) |
| \cV | = | \cv | = | \x16 | ( | Caractère de CONTRÔLE | 'SYN' | = | SYNCHRONISATION | ) |
| \cW | = | \cw | = | \x17 | ( | Caractère de CONTRÔLE | 'ETB' | = | END TRANSMISSION BLOCK | ) |
| \cX | = | \cx | = | \x18 | ( | Caractère de CONTRÔLE | 'CAN' | = | CANCEL | ) |
| \cY | = | \cy | = | \x19 | ( | Caractère de CONTRÔLE | 'EM' | = | END of MEDIUM | ) |
| \cZ | = | \cz | = | \x1A | ( | Caractère de CONTRÔLE | 'SUB' | = | SUBSTITUTION | ) |
| | | | | | | | | | | |
| \c[ | = | \c{ | = | \x1B | ( | Caractère de CONTRÔLE | 'ESC' | = | ESCAPE | ) |
| \c\ | = | \c\| | = | \x1C | ( | Caractère de CONTRÔLE | 'FS' | = | FILE SEPARATOR | ) |
| \c] | = | \c} | = | \x1D | ( | Caractère de CONTRÔLE | 'GS' | = | GROUP SEPARATOR | ) |
| \c^ | = | \c~ | = | \x1E | ( | Caractère de CONTRÔLE | 'RS' | = | RECORD SEPARATOR | ) |
| \c_ | | | = | \x1F | ( | Caractère de CONTRÔLE | 'US' | = | UNIT SEPARATOR | ) |

EXEMPLE :

\c@, en zone REMPLACEMENT, SUBSTITUE le caractère NULL ( \x00 ) à l'expression RÉGULIÈRE recherchée

3) RÉFÉRENCE aux GROUPES NOMMÉS, PRÉDÉFINIS de la RECHERCHE :

En partie REMPLACEMENT, on peut utiliser UN ou PLUSIEURS des 5 groupes NOMMÉS, PRÉDÉFINIS, ci-dessous, qui représentent les ZONES de TEXTE suivantes :

A) TOUTE la CHAÎNE de RECHERCHE : $& ou $0 ou $MATCH ou ${^MATCH}

EXEMPLES :

#$0# APRÈS remplacement, la RegExp recherchée est DÉLIMITÉE par DEUX caractères DIÈSE '#'

-> $MATCH APRÈS remplacement, la RegExp recherchée est PRÉCÉDÉE de la CHAÎNE de 3 caractères '-> '

B) La zone AVANT la CHAÎNE de RECHERCHE : $` ou $PREMATCH ou ${^PREMATCH}

Cette ZONE représente l'ENSEMBLE des caractères compris ENTRE :

- le DÉBUT de la chaîne SUJET et la PREMIÈRE occurrence, du GABARIT de RECHERCHE, trouvée
- la PRÉCÉDENTE occurrence trouvée et l'occurrence COURANTE, du GABARIT de RECHERCHE, trouvée

EXEMPLE :

Soit une chaîne SUJET constituée de MOTS de longueur QUELCONQUE, CHACUN d'eux étant composé de LETTRES ou du TIRET BAS et PRÉCÉDÉ d'un NOMBRE de longueur QUELCONQUE, comme ci-dessous :

```
123Ceci456est_un789Essai
012de6Texte12345Z
```

Alors, si la zone de RECHERCHE contient l'expression RÉGULIÈRE [\l\u_]+ et la zone de REMPLACEMENT contient l'expression RÉGULIÈRE $`${^MATCH}, chaque NOMBRE, situé AVANT

un MOT est REDOUBLÉ, APRÈS appui sur le bouton " Remplacer tout "

=> 123123Ceci456456est_un789789Essai
012012de66Texte1234512345Z

IMPORTANT :

Les remplacements SUCCESSIFS, par APPUI sur le bouton " Remplacer ", sont INOPÉRANTS et, SEUL l'appui sur le bouton " Remplacer tout " donnera le résultat CORRECT !

De PLUS, s'il n'existe qu'UNE SEULE occurrence de l'expression RÉGULIÈRE de RECHERCHE, dans la chaîne SUJET, la CASE " Boucler " devra être COCHÉE

Si la zone, située AVANT la CHAÎNE cherchée, DÉBUTE par N caractères 'FIN de LIGNE', le groupe, PRÉDÉFINI, $` ( ou $PREMATCH ou ${^PREMATCH} ) COMMENCE par N-1 caractères 'FIN de LIGNE' UNIQUEMENT

C) La zone APRÈS la CHAÎNE de RECHERCHE : $' ou $POSTMATCH ou ${^POSTMATCH}

Elle représente l'ENSEMBLE des caractères, situés APRÈS l'occurrence ACTUELLE du GABARIT de RECHERCHE

EXEMPLE :

Soit une chaîne SUJET, composée de lettres MAJUSCULES ou MINUSCULES, CHACUNE d'elles étant PRÉCÉDÉE par une CHAÎNE de 5 TIRETS HAUT '-----', et se TERMINANT par une 'FIN de LIGNE' :

-----a-----B-----c-----Z

Alors, si la zone de RECHERCHE contient l'expression RÉGULIÈRE [\l\u]+ et la zone de REMPLACEMENT contient l'expression RÉGULIÈRE $0${^POSTMATCH}, on obtient, APRÈS appui sur le bouton " Remplacer tout " , le TEXTE ci-dessous :

-----a-----B-----c-----Z
-----B-----c-----Z
-----c-----Z
-----Z

Si, par CONTRE, la chaîne SUJET '-----a-----B-----c-----Z' NE se TERMINE PAS par une 'FIN de LIGNE', on obtient, APRÈS appui sur le bouton " Remplacer tout ", la CHAÎNE :

-----a-----B-----c-----Z-----B-----c-----Z-----c-----Z-----Z

IMPORTANT :

Les remplacements SUCCESSIFS, par APPUI sur le bouton " Remplacer ", sont INOPÉRANTS et, SEUL l'appui sur le bouton " Remplacer tout " donnera le résultat CORRECT !

D) Le DERNIER groupe de CAPTURE, ACTUELLEMENT APPARIÉ : $^N ou $LAST_SUBMATCH_RESUL ou ${^LAST_SUBMATCH_RESUL}

Ce groupe, PRÉDÉFINI, représente le CONTENU du groupe de CAPTURE, possédant la DERNIÈRE parenthèse FERMANTE, de la PARTIE, de l'expression RÉGULIÈRE de RECHERCHE, ACTUELLEMENT APPARIÉE

EXEMPLE :

Soit une chaîne SUJET '-----abcdef-----'

Alors, si l'expression RÉGULIÈRE de RECHERCHE est (a)|b|(c)(d)e|(f) et l'expression RÉGULIÈRE de REMPLACEMENT est <$^N> , on obtient, APRÈS appui sur le bouton " Remplacer tout ", la CHAÎNE :

-----<a><><d><f>-----

NOTES :

Quand le gabarit TROUVÉ est (a) ou (f) , l'ENTITÉ $^N vaut la valeur ELLE-MÊME

Quand le gabarit TROUVÉ est b ( PAS de GROUPE ), l'ENTITÉ $^N vaut la chaîne VIDE

Quand le gabarit TROUVÉ est (c)(d)e , l'ENTITÉ $^N vaut la MINUSCULE 'd'

REMARQUE :

En FONCTION de la PLACE et du NOMBRE de PARENTHÈSES dans le gabarit INITIAL abc , et pour une

chaîne SUJET '---abc---', sont indiqués, dans les TABLEAUX ci-dessous, le gabarit EFFECTIF

et la VALEUR du groupe PRÉDÉFINI $^N qui lui correspond :

| GABARIT | $^N |
|---|---|
| abc | RIEN |
| (ab)c | ab |
| (a(b))c | ab |
| ((a)b)c | ab |
| ((a)(b))c | ab |
| (abc) | abc |
| (a(b)c) | abc |
| (ab(c)) | abc |
| (a(b)(c)) | abc |
| (a)bc | a |
| a(b)c | b |
| (a)(b)c | b |

| GABARIT | $^N |
|---|---|
| (a)(b)(c) | c |
| a(bc) | bc |
| a((b)c) | bc |
| a((b)c) | bc |
| a((b)(c)) | bc |
| ((a)bc) | abc |
| ((a)(b)c) | abc |
| ((a)b(c)) | abc |
| ((a)(b)(c)) | abc |
| ab(c) | c |
| (a)b(c) | c |
| a(b)(c) | c |

E) Le GROUPE de CAPTURE de PLUS GRAND NUMÉRO : $+ ou $LAST_PAREN_MATCH ou ${^LAST_PAREN_MATCH}

Ce groupe, PRÉDÉFINI, représente le CONTENU du groupe de CAPTURE, de PLUS GRAND numéro, de l'expression RÉGULIÈRE cherchée, à CONDITION que ce groupe CAPTURANT soit ACTUELLEMENT celui APPARIÉ

EXEMPLE :

Soit une chaîne SUJET '-----abcdef-----'

Alors, si l'expression RÉGULIÈRE de RECHERCHE est (a)|b|(c)(d)e|(f) et l'expression RÉGULIÈRE de REMPLACEMENT est <$+> , on obtient, APRÈS appui sur le bouton " Remplacer tout ", la CHAÎNE :

-----<><><><f>----

NOTES :

Quand le gabarit TROUVÉ est (a) ou (b) ou (c)(d)e , l'ENTITÉ $+ vaut la chaîne VIDE

Quand le gabarit TROUVÉ est (f) , l'ENTITÉ $+ vaut la MINUSCULE 'f'

REMARQUE :

En FONCTION de la PLACE et du NOMBRE de PARENTHÈSES dans le gabarit INITIAL abc , et pour une

chaîne SUJET '---abc---', sont indiqués, dans les TABLEAUX ci-dessous, le gabarit EFFECTIF

et la VALEUR du groupe PRÉDÉFINI $+ qui lui correspond :

| GABARIT | $+ |
|---|---|
| abc | RIEN |
| (ab)c | ab |
| (a(b))c | b |
| ((a)b)c | a |
| ((a)(b))c | b |
| (abc) | abc |
| (a(b)c) | b |
| (ab(c)) | c |
| (a(b)(c)) | c |
| (a)bc | a |
| a(b)c | b |
| (a)(b)c | b |

| GABARIT | $+ |
|---|---|
| (a)(b)(c) | c |
| a(bc) | bc |
| a((b)c) | b |
| a((b)c) | b |
| a((b)(c)) | c |
| ((a)bc) | a |
| ((a)(b)c) | b |
| ((a)b(c)) | c |
| ((a)(b)(c)) | c |
| ab(c) | c |
| (a)b(c) | c |
| a(b)(c) | c |

RAPPEL :

Les NOMS des 5 groupes PRÉDÉFINIS doivent être, IMPÉRATIVEMENT, écrits avec leur casse EXACTE

EXEMPLE :

Soit la chaîne SUJET : 123abc

Si abc en partie RECHERCHE et $PREMATCH_$0 en partie REMPLACEMENT => 123123_abc

Si abc en partie RECHERCHE et $PREMATcH_$0 en partie REMPLACEMENT => 123$PREMATcH_abc

NOTE :

Si on DÉSIRE remplacer, par exemple, l'expression RÉGULIÈRE de RECHERCHE par la chaîne LITTÉRALE '${LAST_PAREN_MATCH}, on écrira $${LAST_PAREN_MATCH} ou \${LAST_PAREN_MATCH} dans la zone de REMPLACEMENT

4) RÉFÉRENCE aux GROUPES de CAPTURE NON NOMMÉS de la RECHERCHE :

En partie REMPLACEMENT, le CONTENU du groupe CAPTURANT, NON NOMMÉ, de numéro ABSOLU N, de l'expression RÉGULIÈRE de RECHERCHE, s'obtient par :

- la FORME $N , avec la condition N > 0

- la FORME ${N} , avec la condition N > 0

- la FORME \N , avec la DOUBLE condition 0 < N < 9

NOTES :

Si le NOMBRE N indiqué est SUPÉRIEUR au PLUS GRAND numéro de GROUPE de l'expression RÉGULIÈRE cherchée,

les FORMES $N ou ${N} ou \N renvoient alors la chaîne VIDE

Si un CHIFFRE suit IMMÉDIATEMENT la forme $N , on écrira ce CHIFFRE sous sa forme OCTALE ou HEXADÉCIMALE

EXEMPLES :

Si $0359 , en REMPLACEMENT, la SUBSTITUTION vaut le CONTENU du GROUPE n° 359, s'il EXISTE

Si $03\0659 , en REMPLACEMENT, la SUBSTITUTION vaut le CONTENU du GROUPE n° 3, SUIVI de '59'

L'expression ${03}59, est, TOUTEFOIS, PLUS compréhensible !

REMARQUE :

La POSITION, dans la RegExp de RECHERCHE, du groupe de CAPTURE, remis en partie REMPLACEMENT, influe sur

le NUMÉRO ABSOLU du groupe CONCERNÉ, ( Cas de la structure CONDITIONNELLE (?(DEFINE).....) )

EXEMPLE :

Soit la RECHERCHE d'une adresse INTERNET

- DÉBUTANT une ligne ou PRÉCÉDÉE d'une ESPACE
- FINISSANT une ligne ou SUIVIE d'une ESPACE

et le REMPLACEMENT par BORNAGE de cette MÊME adresse, ESPACES incluses, avec les DEUX caractères '<' et '>'

=> RECHERCHE = (?(DEFINE)(http://[\w.]+))(?|( (?1) )|^((?1) )|( (?1))$|^((?1))$)

(groupe) 1 2 2 2 2

=> REMPLACEMENT = <\2>

Si la structure CONDITIONNELLE (DEFINE) est placée APRÈS le GABARIT de RECHERCHE, on a :

=> RECHERCHE = (?|( (?2) )|^((?2) )|( (?2))$|^((?2))$)(?(DEFINE)(http://[\w.]+))

(groupe) 1 1 1 1 2

=> REMPLACEMENT = <\1>

Dans le PREMIER cas :

le GABARIT de RECHERCHE 'http://[\w.]+', de la partie (DEFINE), est le GROUPE 1

le GROUPE à réécrire, ALTERNATIVE du bloc NON CAPTURANT (?|....), est le GROUPE 2

Dans le SECOND cas :

le GABARIT de RECHERCHE 'http://[\w.]+', de la partie (DEFINE), est le GROUPE 2

le GROUPE à réécrire, ALTERNATIVE du bloc NON CAPTURANT (?|....), est le GROUPE 1

5) RÉFÉRENCE aux GROUPES de CAPTURE NOMMÉS de la RECHERCHE :

En partie REMPLACEMENT, le CONTENU d'un groupe CAPTURANT NOMMÉ, de NOM 'Nom', de l'expression RÉGULIÈRE de RECHERCHE, s'obtient par la FORME : $+{Nom}

NOTES :

Si le NOM 'Nom' indiqué n'EXISTE PAS dans l'expression RÉGULIÈRE de RECHERCHE, la FORME ${+Nom} renvoie alors la chaîne VIDE

6) ÉCRITURE du caractère LITTÉRAL $ :

Pour ANNULER le SENS du MÉTA-caractère DOLLAR $ ( Désignation d'une RÉFÉRENCE ), on utilisera l'UNE des DEUX formes \$ ou $$

EXEMPLES :

Si \$03 , en REMPLACEMENT, la RegExp de RECHERCHE est SUBSTITUÉE par la CHAÎNE '$03'

Si $${+Guy} , en REMPLACEMENT, la RegExp de RECHERCHE est SUBSTITUÉE par la CHAÎNE '${+Guy}'

TOUTE forme, DÉBUTANT par le caractère DOLLAR, DIFFÉRENTE des FORMES '$$' , '\$' et de CELLES indiquées aux PARTIES 2), 3) et 4), ci-DESSUS, s'écrit sous forme LITTÉRALE

EXEMPLES :

Si $Toto , en REMPLACEMENT, la RegExp de RECHERCHE est SUBSTITUÉE par la CHAÎNE '$Toto'

Si ${5Guy} , en REMPLACEMENT, la RegExp de RECHERCHE est SUBSTITUÉE par la CHAÎNE '${5Guy}'

Si ${----} , en REMPLACEMENT, la RegExp de RECHERCHE est SUBSTITUÉE par la CHAÎNE '${-----}'

Si ab $@@ , en REMPLACEMENT, la RegExp de RECHERCHE est SUBSTITUÉE par la CHAÎNE 'ab $@@'

7) GROUPEMENT LEXICAL :

Les DEUX parenthèses, OUVRANTE et FERMANTE, permettent de DÉFINIR la CHAÎNE, située ENTRE celles-ci, comme un groupe LEXICAL, qui sera GÉNÉRALEMENT utilisé dans les STRUCTURES CONDITIONNELLES ( Voir paragraphe SUIVANT )

NOTE :

Pour SUBSTITUER une PARENTHÈSE LITTÉRALE, on utilisera les FORMES \( et/ou \)

EXEMPLE :

\(\)\(\) , en REMPLACEMENT, SUBSTITUE la RegExp de RECHERCHE par la CHAÎNE de 4 cars. '()()'

8) EXPRESSIONS CONDITIONNELLES :

Les EXPRESSIONS CONDITIONNELLES, en partie REMPLACEMENT, permettent de SUBSTITUER :

- une VALEUR à CHAQUE ALTERNATIVE TROUVÉE, de numéro = N ou de NOM = 'Nom' , de la RegExp de RECHERCHE
- une valeur DIFFÉRENTE, selon que le GROUPE, de numéro = N ou de NOM = 'Nom', est TROUVÉ ou NON trouvé

Les DEUX formes GÉNÉRALES des EXPRESSIONS CONDITIONNELLES sont :

?ID_GroupeValeur_si_VRAI

?ID_GroupeValeur_si_VRAI:Valeur_si_FAUX

avec ID_Groupe = le NUMÉRO ou le NOM du GROUPE évalué

et Valeur_si_VRAI = la VALEUR de SUBSTITUTION, si le GROUPE ID_Groupe est APPARIÉ

et Valeur_si_FAUX = la VALEUR de SUBSTITUTION, si le GROUPE ID_Groupe n'est PAS APPARIÉ

Les 6 SYNTAXES des EXPRESSIONS CONDITIONNELLES sont les suivantes :

La FORME ?NombreValeur_si_VRAI , si Nombre < 10

La FORME ?NombreValeur_si_VRAI:Valeur_si_FAUX , si Nombre < 10

La FORME ?{Nombre}Valeur_si_VRAI

La FORME ?{Nombre}Valeur_si_VRAI:Valeur_si_FAUX

La FORME ?{Nom}Valeur_si_VRAI

La FORME ?{Nom}Valeur_si_VRAI:Valeur_si_FAUX

REMARQUES :

Afin de PRÉVENIR toute AMBIGUÏTÉ, il est conseillé d'ENTOURER les EXPRESSIONS CONDITIONNELLES par des PARENTHÈSES

Cependant, les PARENTHÈSES FERMANTES, qui TERMINENT la zone de REMPLACEMENT, peuvent être OMISES

EXEMPLES :

Soit (Alain)|(perroquet)|(vert) , dans la zone de RECHERCHE

et (?1Élisabeth)(?2chat)?3gris et blanc , dans la zone de REMPLACEMENT

=> TOUTE occurrence de la CHAÎNE 'Alain' est REMPLACÉE par la CHAÎNE 'Élisabeth'

TOUTE occurrence de la CHAÎNE 'perroquet' est REMPLACÉE par la CHAÎNE 'chat'

TOUTE occurrence de la CHAÎNE 'vert' est REMPLACÉE par la CHAÎNE 'gris et blanc'

Dans le cas GÉNÉRAL, de SUBSTITUTION d'un CARACTÈRE par un AUTRE caractère, on prendra :

(Car.1)|(Car.2)|.......|(Car.n) , en partie RECHERCHE

(?1Car.A)(?2Car.B).......(?nCar.X) , en partie REMPLACEMENT

avec la CASE " Respecter la casse " COCHÉE

APRÈS remplacement GLOBAL => Car. 1 devient Car. A , Car. 2 devient Car. B, etc.

Soit ^(\d+)?.*$ , dans la zone de RECHERCHE

Soit (?1Age:Nom) : $& , dans la zone de REMPLACEMENT

et la chaîne SUJET, en partie GAUCHE, qui devient, APRÈS remplacement GLOBAL, le TEXTE en partie DROITE :

```
Pierre                          ==>             Nom : Pierre
26                              ==>             Age : 26
Jacques                         ==>             Nom : Jacques
28                              ==>             Age : 28
Christian                       ==>             Nom : Christian
42                              ==>             Age : 42
```

```
Guy                              ==>          Nom : Guy
60 ( Eh oui, déjà !)             ==>          Age : 60 ( Eh oui, déjà )
```

NOTES :

Si un NOMBRE est lu, en DÉBUT de LIGNE, le GROUPE 1 est APPARIÉ

=> La CHAÎNE 'Age : ' est INSÉRÉE, AVANT la ligne COMPLÈTE ($&)

Si un PRÉNOM est lu, en DÉBUT de LIGNE, le GROUPE 1 n'est PAS APPARIÉ

=> La CHAÎNE 'Nom : ' est INSÉRÉE, AVANT la ligne COMPLÈTE ($&)

Pour l'utilisation CORRECTE des EXPRESSIONS CONDITIONNELLES, il est nécessaire de TRANSFORMER le groupe OPTIONNEL, sur lequel porte le TEST d'appariement, en un groupe NON OPTIONNEL, SUIVI du QUANTIFICATEUR ?

En effet, (\d+)? est VÉRIFIÉE si EXISTE des CHIFFRES et NON VÉRIFIÉE si EXISTE des LETTRES

Mais (\d*) est TOUJOURS vérifiée ( car cela signifie 1 ou PLUSIEURS ou AUCUN chiffre )

=> la partie Valeur_si_FAUX, dans ce CAS, ne se présente donc JAMAIS !

REMARQUES :

Pour écrire un signe POINT d'INTERROGATION (?) LITTÉRAL ou un signe DEUX-POINTS (:) LITTÉRAL, employer les FORMES \? et/ou \:

Les expressions CONDITIONNELLES de REMPLACEMENT peuvent être IMBRIQUÉES :

EXEMPLE :

Soit un FICHIER, constitué des SIX lignes, selon le FORMAT : NOM-PRÉNOM + TABULATION(S) + Âge

comme ci-DESSOUS ( l'âge peut être ABSENT ou INDÉFINI ) :

```
Jean-Pierre DUPONT  3
Jean DURAND         ???
Guy THEVENOT        60
Robert BOUCHARD
Don HO              1025
René MONJOT         103
```

Alors la RegExp ^(.+)\t+(\d?(\d)?\d$)? en partie RECHERCHE

et la RegExp \1\t\t âge (?2correct et (?3>=:<) cent ans:erroné \: ) en partie REMPLACEMENT

permettent d'obtenir le FORMATAGE suivant :

```
Jean-Pierre DUPONT      âge correct et < cent ans
Jean DURAND             âge erroné : ???
Guy THEVENOT            âge correct et < cent ans
Robert BOUCHARD         âge erroné :
Don HO                  âge erroné : 1025
René MONJOT             âge correct et >= cent ans
```

EXPLICATIONS :

^(.+)\t+ sélectionne les NOMS-PRÉNOMS, en DÉBUT de LIGNE, suivis de 1 ou PLUSIEURS TABULATIONS

Suite à un BACKTRACKING, le 1ER groupe (.+) est constitué de CHAQUE nom-prénom, suivi

du NOMBRE de TABULATIONS - 1, car \t+ correspond TOUJOURS à la DERNIÈRE tabulation \t

(\d?(\d)?\d$)?, cherche une SUITE de CHIFFRES, OPTIONNELLE, constituant le GROUPE 2, qui est

composée de 3 CHIFFRES, dont les 2 PREMIERS sont OPTIONNELS, et dont le DERNIER chiffre

TERMINE la LIGNE

Le 3ÈME groupe (\d) correspond au 2ÈME CHIFFRE du NOMBRE à 3 chiffres :

- si nombre à 1 CHIFFRE, les 2 PREMIERS termes \d, OPTIONNELS, valent la chaîne VIDE
- si nombre à 2 CHIFFRES, le 1ER \d correspond aux DIZAINES et le 2ÈME \d à chaîne VIDE
- si nombre à 3 CHIFFRES, le 1ER \d vaut les CENTAINES et le 2ÈME \d vaut les DIZAINES
- si nombre à PLUS de 3 CHIFFRES, la SUITE \d?(\d)?\d$ n'est JAMAIS possible => les GROUPES 2 et 3 ne sont, alors, PAS DÉFINIS

'\1\t\t âge ', en REMPLACEMENT, réécrit le 1ER groupe, SUIVI de 2 TABULATIONS, puis de ' âge '

Si le GROUPE 2 EXISTE, on écrit la CHAÎNE 'correct et ' ( Partie THEN, AVANT signe : )

Si le GROUPE 3 EXISTE, on écrit '>=' ( Partie THEN, AVANT signe : )

Si le GROUPE 3 N'existe PAS, on écrit '<' ( Partie ELSE, APRÈS signe : )

On TERMINE en écrivant la CHAÎNE ' cent ans'  ( Partie THEN, AVANT signe : )

Si le GROUPE 2 N'existe PAS, on écrit la CHAÎNE 'erroné : '  ( Partie ELSE, APRÈS signe : )

9) MODIFICATEURS de CASSE :

Les MODIFICATEURS de CASSE sont des séquences d'ÉCHAPPEMENT, qui MODIFIE la CASSE du ( ou des ) CARACTÈRES,
placés APRÈS, à SUBSTITUER à l'expression RÉGULIÈRE de RECHERCHE

Les MODIFICATEURS de CASSE peuvent être INSÉRÉS et AGIR :

- DANS les chaînes LITTÉRALES
- DEVANT les RÉFÉRENCES de GROUPE PRÉDÉFINIS, EXCEPTÉ les DEUX références $PREMATH et $POSTMATCH
- DEVANT les RÉFÉRENCES de GROUPE, NOMMÉS ou NON NOMMÉS
- DEVANT les EXPRESSIONS CONDITIONNELLES

Les 5 FORMES de MODIFICATEURS de CASSE sont :

- la FORME \l qui AFFICHE le caractère SUIVANT en MINUSCULE
- la FORME \u qui AFFICHE le caractère SUIVANT en MAJUSCULE

- la FORME \L qui AFFICHE TOUS les caractères SUIVANTS en MINUSCULES, EXCEPTÉ un caractère PRÉCÉDÉ de la la FORME \u , jusqu'à l'APPARITION d'UNE des DEUX formes \U ou \E

- la FORME \U qui AFFICHE TOUS les caractères SUIVANTS en MAJUSCULES, EXCEPTÉ un caractère PRÉCÉDÉ de la la FORME \l , jusqu'à l'APPARITION d'UNE des DEUX formes \L ou \E

EXEMPLES :

\w+ en partie RECHERCHE et \u$0 en partie REMPLACEMENT, SUBSTITUE TOUS les MOTS, commençant par des LETTRES par ces MÊMES mots, avec la PREMIÈRE lettre en MAJUSCULE

\UaBc\lAbcDef en partie REMPLACEMENT, SUBSTITUE l'expression RÉGULIÈRE recherchée par la CHAÎNE 'ABCaBCDEF'

\LaBc\uabcDef en partie REMPLACEMENT, SUBSTITUE l'expression RÉGULIÈRE recherchée par la CHAÎNE 'abcAbcdef'

(?1Age:Nom) : $& en partie REMPLACEMENT, peut AUSSI s'écrire \u(?1age:nom) : $&

(a)|b|(cde)|(f) en partie RECHERCHE et <\u$^N> en partie REMPLACEMENT, SUBSTITUE la chaîne SUJET 'fabcdef' par la CHAÎNE de REMPLACEMENT '<F><A><><Cde><F>'

IV) SUPPRESSION ou AJOUT de LIGNES VIDES :

Les 4 TABLEAUX, ci-dessous, récapitulent DIFFÉRENTES manières de SUPPRIMER ou d'AJOUTER des lignes VIDES, dans un FICHIER, avec les expressions RÉGULIÈRES, à indiquer dans les DEUX zones de RECHERCHE et de REMPLACEMENT, pour CHAQUE cas

IMPORTANT :

- AVANT toute SUPPRESSION de lignes VIDES par RECHERCHE-REMPLACEMENT, il est recommandé de SUPPRIMER TOUTES les lignes VIDES, placées en TOUT DÉBUT du fichier COURANT

- Dans la partie RECHERCHE, les 'FIN de LIGNE' sont trouvées par l'UTILISATION du caractère JOKER \R

  Cependant, le SYMBOLE \R trouve, en PLUS des caractères 'FIN de LIGNE' STANDARD ( CR+LF, LF, CR ), 5 AUTRES caractères, ASSIMILÉS à une 'FIN de LIGNE' : \x0B ( VT ) , \f ( FF ) , \x85 ( … ) , \x{2028} et \x{2029}

  Si le fichier COURANT comporte UN ou PLUSIEURS de ces CARACTÈRES, il sera NÉCESSAIRE de REMPLACER, en partie RECHERCHE, CHAQUE symbole \R par :

  \r\n si le TYPE du fichier est 'WINDOWS'

  \n si le TYPE du fichier est 'UNIX'

  \r si le TYPE du fichier est 'MAC'

- Dans les TABLEAUX, des caractères 'FIN de LIGNE' WINDOWS, soit la SUITE '\r\n' sont, parfois, INDIQUÉS
  en partie REMPLACEMENT :

  Si le TYPE de fichier est 'UNIX' on écrira UNIQUEMENT le caractère \n

  Si le TYPE de fichier est 'MAC', on écrira UNIQUEMENT le caractère \r

Dans les TABLEAUX ci-dessous, les CONVENTIONS suivantes sont utilisées :

FL désigne une 'FIN de LIGNE', soit CR + LF si fichier WINDOWS, LF si fichier UNIX ou CR si fichier MAC

=> - entre DEUX lignes JUXTAPOSÉES, , il y a ZÉRO 'FIN de LIGNE'

- entre DEUX lignes CONSÉCUTIVES, , il y a UNE 'FIN de LIGNE'

- entre DEUX lignes, séparées par une INTERLIGNE, il y a DEUX 'FIN de LIGNE' et ainsi de suite...

Le nombre POSITIF ou NUL n représente le NOMBRE de symbole \R, en partie RECHERCHE

Le nombre POSITIF ou NUL m représente le NOMBRE de suites '\r\n' ( ou \n ou \r ), en partie REMPLACEMENT

Le nombre POSITIF ou NUL p représente le NOMBRE de suites '\1', en partie REMPLACEMENT

Le nombre POSITIF x représente le NOMBRE de 'FIN de LIGNE', ACTUELLEMENT trouvé par la recherche COURANTE

ATTENTION :

Certaines RECHERCHES utilisent (\R)* ou (\R)+ et d'AUTRES recherches utilisent (\R*) ou (\R+) !!

| Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton " Remplacer " | Bouton " Remplacer tout " |
|---|---|---|---|
| \R* | \r\n......\r\n | AJOUT de m lignes VIDES | BUG - BOUCLAGE |
| (\R)* | \r\n......\r\n\1....\1 | AJOUT de m lignes VIDES | BUG - BOUCLAGE |

| Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton Remplacer | Bouton " Remplacer tout " |
|---|---|---|---|
| (\R)* | \1.....\1<br>---------<br>p fois | BUG | Si x <= 1 FL => IDEM<br>Si x > 1 FL => p+1 FL |
| \R\R....\R(\R)*<br>----------<br>n fois ( >0 ) | \r\n......\r\n\1....\1<br>----------<br>m fois p fois | Si x < n+1 FL => IDEM<br>Si x = n+1 FL => m+1 FL<br>Si x > n+1 FL => m+p+1 FL | Si x < n FL => IDEM<br>Si x = n FL => m FL<br>Si x > n FL => m+p FL |

| Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton Remplacer | Bouton " Remplacer tout " |
|---|---|---|---|

| | | | |
|---|---|---|---|
| (\R*) | \1.....\1<br>---------<br>p fois | BUG | Si x FL => p(x-1)+1 FL |
| \R\R....\R(\R*)<br>----------<br>n fois ( >0 ) | \r\n......\r\n\1<br>--------------<br>m fois | Si x < n+1 FL => IDEM<br><br>Si x >= n+1 FL => x+m-n FL | Si x < n FL => IDEM<br><br>Si x >= n FL => x+m-n FL |

| Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton Remplacer | Bouton " Remplacer tout " |
|---|---|---|---|
| \R\R....\R(\R)+<br>----------<br>n fois | \r\n......\r\n\1....\1<br>----------<br>m fois p fois | Si x <= n+1 FL => IDEM<br><br>Si x > n+1 FL => m+p+1 FL | Si x <= n FL => IDEM<br><br>Si x > n FL => m+p FL |

| Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton Remplacer | Bouton " Remplacer tout " |
|---|---|---|---|
| \R\R....\R(\R+)<br>----------<br>n fois | \r\n......\r\n\1<br>--------------<br>m fois | Si x <= n+1 FL => IDEM<br><br>Si x > n+1 FL => x+m-n FL | Si x <= n FL => IDEM<br><br>Si x > n FL => x+m-n FL |

A PARTIR des formules GÉNÉRALES, ci-dessus, on peut en déduire QUELQUES cas PARTICULIERS intéressants :

1) CAS du remplacement GLOBAL ( Bouton " Remplacer tout " ) :

| | Zone de RECHERCHE | Zone de REMPLACEMENT | NBRE de FL ÉCRITES, selon le NBRE de FL TROUVÉES | EFFET du REMPLACEMENT Bouton " Remplacer tout " |
|---|---|---|---|---|
| | \R* | RIEN | Si x <= 1 FL => IDEM<br>Si x > 1 FL => 1 FL | SUPPRESSION de TOUTES les lignes VIDES, soit TOUS LES INTERLIGNES |
| | (\R)* | \1 | Si x <= 1 FL => IDEM<br>Si x > 1 FL => 2 FL | SUPPRESSION de TOUTES les lignes EXCÉDENTAIRES 2ème, 3ème... ligne VIDE |
| | \R(\R)* | \1....\1<br>--------<br>p fois | Si x < 1 FL => IDEM<br>Si x = 1 FL => 0 FL<br>Si x > 1 FL => p FL | JUXTAPOSITION des BLOCS de lignes NON VIDES, séparés de p 'FIN de LIGNE' |
| | \R(\R*) | \1 | Si x < 1 FL => IDEM<br>Si x >= 1 FL => x-1 FL | DIMINUTION de TOUTE SUITE de 'FIN de LIGNE' de 1 'FIN de LIGNE' |
| (*) | \R\R(\R*) | \r\n\1 | Si x < 2 FL => IDEM<br>Si x >= 2 FL => x-1 FL | DIMINUTION de TOUTE SUITE de 'FIN de LIGNE' > 1 de 1 'FIN de LIGNE' |
| | \R+ | RIEN | Si x <= 0 FL => IDEM<br>Si x > 0 FL => 0 FL | 1 SEULE ligne RÉSULTANTE ESPACES et TABULATIONS (\t) sont TOUS CONSERVÉS |
| (*) | \R+ | \r\n | Si x <= 0 FL => IDEM<br>Si x > 0 FL => 1 FL | SUPPRESSION de TOUTES les lignes VIDES, soit TOUS LES INTERLIGNES |
| | \R....\R(\R+)<br>-------- | \1 | Si x <= n FL => IDEM | DIMINUTION de TOUTE SUITE de 'FIN de LIGNE' > n |

| n fois | | Si x > n FL => x-n FL | de n 'FIN de LIGNE' |
|---|---|---|---|
| \R(\R+) | \1 | Si x <= 1 FL => IDEM<br>Si x > 1 FL => x-1 FL | DIMINUTION de TOUTE SUITE<br>de 'FIN de LIGNE' > 1<br>de 1 'FIN de LIGNE' |

IMPORTANT :

TOUTES ces RECHERCHES-REMPLACEMENTS, ci-dessus, fonctionnent QUEL que SOIT le TYPE du fichier COURANT

( WINDOWS, UNIX ou MAC ), EXCEPTÉ les DEUX formes, repérées par une ASTÉRISQUE, opérationnelles,

par DÉFAUT, pour des fichiers de type WINDOWS, SEULEMENT

2) CAS du remplacement PAS à PAS ( Bouton " Remplacer " ) :

| Zone de RECHERCHE | Zone de REMPLACEMENT | NBRE de FL ÉCRITES, selon le NBRE de FL TROUVÉES | EFFET du REMPLACEMENT Bouton " Remplacer " |
|---|---|---|---|
| \R* | RIEN | BUG | BUG |
| (\R)* | \1 | BUG | BUG |
| \R(\R)* | \1....\1<br>-------- | Si x < 2 FL => IDEM<br>Si x = 2 FL => 1 FL | INTERLIGNE seul SUPPRIMÉ<br>TOUTE SUITE de FL > 2 |

| | | | | |
|---|---|---|---|---|
| | p fois | Si x > 2 FL => p+1 FL | est AJUSTÉE à p+1 FL | |
| | \R(\R*) | \1 | Si x < 2 FL => IDEM | DIMINUTION de TOUTE SUITE de 'FIN de LIGNE' > 1 |
| | | | Si x >= 2 FL => x-1 FL | de 1 'FIN de LIGNE' |
| (*) | \R\R(\R*) | \r\n\1 | Si x < 3 FL => IDEM | DIMINUTION de TOUTE SUITE de 'FIN de LIGNE' > 2 |
| | | | Si x >= 3 FL => x-1 FL | de 1 FIN de LIGNE |
| | \R+ | RIEN | Si x <= 1 FL => IDEM | SUPPRESSION de TOUTES les lignes VIDES, soit |
| | | | Si x > 1 FL => 1 FL | TOUS LES INTERLIGNES |
| (*) | \R+ | \r\n | Si x <= 1 FL => IDEM | SUPPRESSION de TOUTES les lignes EXCÉDENTAIRES |
| | | | Si x > 1 FL => 2 FL | 2ème, 3ème... ligne VIDE |
| | \R....\R(\R+) -------- n fois | \1 | Si x <= n+1 FL => IDEM | DIMINUTION de TOUTE SUITE de 'FIN de LIGNE' > n+1 |
| | | | Si x > n+1 FL => x-n FL | de n 'FIN de LIGNE' |

IMPORTANT :

Le REMPLACEMENT PAS à PAS n'est PAS conseillé ( BUGS ), EXCEPTÉ pour un nombre RÉDUIT de REMPLACEMENTS

TOUTES ces RECHERCHES-REMPLACEMENTS, ci-dessus, fonctionnent QUEL que SOIT le TYPE du fichier COURANT

( WINDOWS, UNIX ou MAC ), EXCEPTÉ les DEUX formes, repérées par une ASTÉRISQUE, opérationnelles,

par DÉFAUT, pour des fichiers de type WINDOWS, SEULEMENT

V) RECHERCHES et/ou REMPLACEMENTS STANDARD :

IMPORTANT :

- Le REMPLACEMENT par l'appui sur le BOUTON " Remplacer tout " ( Méthode GLOBALE ) est TOUJOURS correct

- Le REMPLACEMENT par l'appui sur le BOUTON " Remplacer " ( Méthode PAS à PAS ) fonctionne GÉNÉRALEMENT
  Quand ce n'est PAS le cas ( par BUG ou résultat ERR0NÉ ), une NOUVELLE colonne, 'Bouton Remplacer'
  est CRÉÉE, avec l'indication 'NON'

- Quand la RECHERCHE SEULE, SANS remplacement est POSSIBLE, la SYNTAXE de la ZONE de RECHERCHE est indiquée
  dans un SECOND tableau, à DROITE du PREMIER tableau

- AVANT une action de RECHERCHE-REMPLACEMENT, il est recommandé de SUPPRIMER TOUTES les lignes VIDES,
  ÉVENTUELLES, placées en TOUT DÉBUT du fichier COURANT, afin d'ÉVITER quelques BUGS MINEURS de N++ !

- AVANT une action de RECHERCHE-REMPLACEMENT, il est recommandé de SUPPRIMER TOUS les caractères 'FF' (\x0C)
  du fichier COURANT, car :

    - Vis à vis du caractère JOKER '.', celui-ci est vu, à TORT, comme un CARACTÈRE de 'FIN de LIGNE'

- Il peut, parfois, BLOQUER le REMPLACEMENT, au COUP par COUP, avec le BOUTON " Remplacer "

Les DEUX conditions PRÉCÉDENTES seront supposées RÉALISÉES, dans les EXEMPLES ci-après !

Les RECHERCHES-REMPLACEMENTS, ci-dessous, dans les lignes NON VIDES d'un fichier, fonctionnement, SAUF EXCEPTION indiquée, quand la DERNIÈRE ligne NON VIDE du fichier NE se termine PAS par une 'FIN de LIGNE'

1) AJOUT d'une CHAÎNE Ch en DÉBUT de CHAQUE ligne, VIDE ou NON VIDE, d'un FICHIER :

| | Zone de RECHERCHE | Zone de REMPLACEMENT |
|---|---|---|
| (1) | ^ | Ch |
| (2) | .* | Ch$0 |

NOTE :

La CHAÎNE Ch peut, aussi, être un SIMPLE caractère UNIQUE

Cet AJOUT est EFFECTIF sur TOUTES les LIGNES du fichier, VIDES ou NON VIDES

2) AJOUT d'une CHAÎNE Ch en DÉBUT de CHAQUE ligne, NON VIDE, d'un FICHIER :

| | Zone de RECHERCHE | Zone de REMPLACEMENT |
|---|---|---|
| (1) | ^(.) | Ch\1 |
| (2) | .+ | Ch$0 |

NOTE :

La CHAÎNE Ch peut, aussi, être un SIMPLE caractère UNIQUE

3) AJOUT d'une CHAÎNE Ch en FIN de CHAQUE ligne, VIDE ou NON VIDE, d'un FICHIER :

| | Zone de RECHERCHE | Zone de REMPLACEMENT |
|---|---|---|
| (1) | \r | Ch\r |
| (2) | $ | Ch |
| (3) | .* | $0Ch |

NOTES :

La CHAÎNE Ch peut, aussi, être un SIMPLE caractère UNIQUE

Cet AJOUT est EFFECTIF sur TOUTES les LIGNES du fichier, VIDES ou NON VIDES

Dans le CAS 1, le FICHIER doit être PUR WINDOWS ( FIN de LIGNE = \r\n et AUCUN \r ou \n ORPHELIN )

Dans les CAS 2 et 3, le FICHIER ne doit contenir AUCUN caractère 'FF' ( \x0c ), sinon CRASH de N++

Dans le CAS 1, la CHAÎNE Ch n'est PAS AJOUTÉE à la DERNIÈRE ligne NON VIDE, SANS 'FIN de LIGNE'

4) AJOUT d'une CHAÎNE Ch en FIN de CHAQUE ligne, NON VIDE, d'un FICHIER :

| | Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton 'Remplacer' |
|---|---|---|---|
| (1) | (.)$ | \1Ch | Oui |
| (2) | .+ | $0Ch | Oui |
| (3) | .\K$ | Ch | NON |

NOTES :

La CHAÎNE Ch peut, aussi, être un SIMPLE caractère UNIQUE

NE JAMAIS utiliser le bouton " Remplacer " ( Méthode au COUP par COUP ) dans le CAS 3

5) AJOUT d'une CHAÎNE Ch dans TOUTES les lignes VIDES d'un FICHIER :

| Zone de RECHERCHE | Zone de REMPLACEMENT |
|---|---|
| ^$ | Ch |

NOTES :

La CHAÎNE Ch peut, aussi, être un SIMPLE caractère UNIQUE

La CHAÎNE Ch n'est JAMAIS insérée dans la PREMIÈRE ligne VIDE, ÉVENTUELLE, d'un FICHIER

6) SUPPRESSION de 1 à n CARACTÈRES en DÉBUT de CHAQUE ligne d'un FICHIER :

| Zone de RECHERCHE | Zone de REMPLACEMENT | Recherche SEULE SANS Remplacement |
|---|---|---|

| ^.{1,n}(.*) | \1 | | ^.{1,n} |
|---|---|---|---|

NOTES :

TOUS les CARACTÈRES, des LIGNES de n caractères au PLUS, sont donc SUPPRIMÉS

La FORME ^.{1,n} en partie RECHERCHE, et RIEN en partie REMPLACEMENT, ne fonctionne PAS ( BUG Notepad++ )

7) SUPPRESSION du PREMIER caractère de CHAQUE ligne d'un FICHIER :

| Zone de RECHERCHE | Zone de REMPLACEMENT | | Recherche SEULE SANS Remplacement |
|---|---|---|---|
| ^.(.*) | \1 | | ^. |

NOTE :

La FORME ^. en partie RECHERCHE, et RIEN en partie REMPLACEMENT, ne fonctionne PAS ( BUG Notepad++ ! )

8) SUPPRESSION de 1 à n CARACTÈRES en FIN de CHAQUE ligne d'un FICHIER :

| Zone de RECHERCHE | Zone de REMPLACEMENT | Recherche SEULE SANS Remplacement |
|---|---|---|
| .{1,n}$ | RIEN | .{1,n}$ |

NOTES :

TOUS les CARACTÈRES, des LIGNES de n caractères au PLUS, sont donc SUPPRIMÉS

9) SUPPRESSION du DERNIER caractère de CHAQUE ligne d'un FICHIER :

| Zone de RECHERCHE | Zone de REMPLACEMENT | Recherche SEULE SANS Remplacement |
|---|---|---|
| .$ | RIEN | .$ |

10) SUPPRESSION de TOUTES les lignes VIDES d'un FICHIER, de type WINDOWS ( LIGNES finissant par CR + LF ) :

| Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton 'Remplacer' |
|---|---|---|
| (\r\n)+ | \r\n | NON |

| | | | | |
|---|---|---|---|---|
| | (\r\n)+ | \1 | NON | |
| (3) | \R+ | \r\n | NON | Recherche SEULE SANS Remplacement |
| (4) | \R* | RIEN | NON | |
| (5) | ^\R+ | RIEN | NON | ^\R+ |
| (6) | \R\K\R+ | RIEN | NON | \R\K\R+ |

NOTES :

NE JAMAIS utiliser le bouton " Remplacer " ( Méthode au COUP par COUP ) pour cette SUPPRESSION

Les PREMIÈRE et DEUXIÈME lignes VIDES , ÉVENTUELLES, du FICHIER ne sont JAMAIS SUPPRIMÉES

Dans les CAS 3 à 6, les lignes VIDES, égales à CR , LF ou FF, UNIQUEMENT, sont aussi SUPPRIMÉES

11) SUPPRESSION de TOUTES les lignes VIDES d'un FICHIER, de type UNIX ( LIGNES finissant par LF ) :

| | Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton 'Remplacer' | |
|---|---|---|---|---|
| | \n+ | \n | NON | |
| | (\n)+ | \1 | NON | |
| (3) | \R+ | \n | NON | |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | Recherche SEULE |
| (4) | \R* | RIEN | NON | | SANS Remplacement |
| (5) | ^\R+ | RIEN | NON | | ^\R+ |
| (6) | \R\K\R+ | RIEN | NON | | \R\K\R+ |

NOTES :

NE JAMAIS utiliser le bouton " Remplacer " ( Méthode au COUP par COUP ) pour cette SUPPRESSION

Les PREMIÈRE et DEUXIÈME lignes VIDES , ÉVENTUELLES, du FICHIER ne sont JAMAIS SUPPRIMÉES

Dans les CAS 3 à 6, les lignes VIDES, égales à CR , FF ou CR + LF, sont aussi SUPPRIMÉES

12) SUPPRESSION de TOUTES les lignes VIDES d'un FICHIER, de type MAC ( LIGNES finissant par CR ) :

| | Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton 'Remplacer' | |
|---|---|---|---|---|
| | \r+ | \r | NON | |
| | (\r)+ | \1 | NON | |
| (3) | \R+ | \r | NON | |
| | | | | Recherche SEULE |
| (4) | \R* | RIEN | NON | SANS Remplacement |

| | | | | | |
|---|---|---|---|---|---|
| (5) | ^\R+ | RIEN | NON | | ^\R+ |
| (6) | \R\K\R+ | RIEN | NON | | \R\K\R+ |

NOTES :

NE JAMAIS utiliser le bouton " Remplacer " ( Méthode au COUP par COUP ) pour cette SUPPRESSION

Les PREMIÈRE et DEUXIÈME lignes VIDES , ÉVENTUELLES, du FICHIER ne sont JAMAIS SUPPRIMÉES

Dans les CAS 3 à 6, les lignes VIDES, égales à LF , FF ou CR + LF, sont aussi SUPPRIMÉES

13) SUPPRESSION de TOUTES les lignes VIDES EXCÉDENTAIRES d'un FICHIER, de type WINDOWS ( LIGNES finissant par CR + LF ) :

| | Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton 'Remplacer' | |
|---|---|---|---|---|
| | \r\n(\r\n)+ | \r\n\r\n | NON | |
| | \r\n(\r\n)+ | \1\1 | NON | |
| (3) | \R\R+ | \r\n\r\n | NON | |
| (4) | (\R)* | \1 | NON | Recherche SEULE SANS Remplacement |
| (5) | \R\R\K\R+ | RIEN | NON | \R\R\K\R+ |

NOTES :

NE JAMAIS utiliser le bouton " Remplacer "  ( Méthode au COUP par COUP ) pour cette SUPPRESSION

Les PREMIÈRE, DEUXIÈME et TROISIÈME lignes VIDES , ÉVENTUELLES, du FICHIER ne sont JAMAIS SUPPRIMÉES

Dans les CAS 3 à 5, les lignes VIDES, EXCÉDENTAIRES, égales à CR , LF, ou FF, SEULEMENT, sont SUPPRIMÉES

14) SUPPRESSION de TOUTES les lignes VIDES EXCÉDENTAIRES d'un FICHIER, de type UNIX ( LIGNES finissant par LF ) :

| | Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton 'Remplacer' |
|---|---|---|---|
| | \n\n+ | \n\n | NON |
| | \n(\n)+ | \1\1 | NON |
| (3) | \R\R+ | \n\n | NON |
| (4) | (\R)* | \1 | NON |
| (5) | \R\R\K\R+ | RIEN | NON |

| |
|---|
| Recherche SEULE SANS Remplacement |
| \R\R\K\R+ |

NOTES :

NE JAMAIS utiliser le bouton " Remplacer "  ( Méthode au COUP par COUP ) pour cette SUPPRESSION

Les PREMIÈRE, DEUXIÈME et TROISIÈME lignes VIDES , ÉVENTUELLES, du FICHIER ne sont JAMAIS SUPPRIMÉES

Dans les CAS 3 à 5, les lignes VIDES, EXCÉDENTAIRES, égales à CR , FF ou CR + LF, sont aussi SUPPRIMÉES

15) SUPPRESSION de TOUTES les lignes VIDES EXCÉDENTAIRES d'un FICHIER, de type MAC ( LIGNES finissant par CR ) :

| | Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton 'Remplacer' | |
|---|---|---|---|---|
| | \r\r+ | \r\r | NON | |
| | \r(\r)+ | \1\1 | NON | |
| (3) | \R\R+ | \r\r | NON | Recherche SEULE SANS Remplacement |
| (4) | (\R)* | \1 | NON | |
| (5) | \R\R\K\R+ | RIEN | NON | \R\R\K\R+ |

NOTES :

NE JAMAIS utiliser le bouton " Remplacer " ( Méthode au COUP par COUP ) pour cette SUPPRESSION

Les PREMIÈRE, DEUXIÈME et TROISIÈME lignes VIDES , ÉVENTUELLES, du FICHIER ne sont JAMAIS SUPPRIMÉES

Dans les CAS 3 à 5, les lignes VIDES, EXCÉDENTAIRES, égales à LF , FF ou CR + LF, sont aussi SUPPRIMÉES

16) SUPPRESSION de TOUS caractères 'FIN de LIGNE ' d'un FICHIER, de type WINDOWS ( LIGNES finissant par CR + LF ) :

| | Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton 'Remplacer' | |
|---|---|---|---|---|
| (1) | (\r\n)+ | RIEN | NON | Recherche SEULE SANS Remplacement |
| (2) | \R+ | RIEN | NON | \R+ |

NOTES :

La SUPPRESSION est EFFECTIVE sur TOUTES les LIGNES, VIDES ou NON VIDES

NE JAMAIS utiliser le bouton " Remplacer "  ( Méthode au COUP par COUP ) pour cette SUPPRESSION

APRÈS remplacement, le FICHIER possède une SEULE ligne, ÉVENTUELLEMENT précédée d'une 'FIN de LIGNE'

La PREMIÈRE 'FIN de LIGNE', ÉVENTUELLE, du FICHIER n'est JAMAIS SUPPRIMÉE

Utiliser le PREMIER cas QUE si le FICHIER comporte AUCUN caractère CR ou LF qui soient ORPHELINS

Dans le 2ÈME cas, les 'FINS de LIGNE', égales à CR , LF ou FF, UNIQUEMENT, sont aussi SUPPRIMÉES

17) SUPPRESSION de TOUS caractères 'FIN de LIGNE ' d'un FICHIER, de type UNIX ( LIGNES finissant par LF ) :

| | Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton 'Remplacer' | |
|---|---|---|---|---|
| (1) | \n+ | RIEN | NON | Recherche SEULE SANS Remplacement |
| (2) | \R+ | RIEN | NON | \R+ |

NOTES :

La SUPPRESSION est EFFECTIVE sur TOUTES les LIGNES, VIDES ou NON VIDES

NE JAMAIS utiliser le bouton " Remplacer " ( Méthode au COUP par COUP ) pour cette SUPPRESSION

APRÈS remplacement, le FICHIER possède une SEULE ligne, ÉVENTUELLEMENT précédée d'une 'FIN de LIGNE'

La PREMIÈRE 'FIN de LIGNE', ÉVENTUELLE, du FICHIER n'est JAMAIS SUPPRIMÉE

Utiliser le 1ER cas, QUE si le FICHIER ne comporte PAS de caractère CR ORPHELIN ou de caractères CR + LF

Dans le 2ÈME cas, les 'FINS de LIGNE', égales à CR , FF ou CR + LF, sont aussi SUPPRIMÉES

18) SUPPRESSION de TOUS caractères 'FIN de LIGNE ' d'un FICHIER, de type MAC ( LIGNES finissant par CR ) :

| | Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton 'Remplacer' | |
|---|---|---|---|---|
| (1) | \r+ | RIEN | NON | Recherche SEULE SANS Remplacement |
| (2) | \R+ | RIEN | NON | \R+ |

NOTES :

La SUPPRESSION est EFFECTIVE sur TOUTES les LIGNES, VIDES ou NON VIDES

NE JAMAIS utiliser le bouton " Remplacer "  ( Méthode au COUP par COUP ) pour cette SUPPRESSION

APRÈS remplacement, le FICHIER possède une SEULE ligne, ÉVENTUELLEMENT précédée d'une 'FIN de LIGNE'

La PREMIÈRE 'FIN de LIGNE', ÉVENTUELLE, du FICHIER n'est JAMAIS SUPPRIMÉE

Utiliser le 1ER cas, QUE si le FICHIER ne comporte PAS de caractère LF ORPHELIN ou de caractères CR + LF

Dans le 2ÈME cas, les 'FINS de LIGNE', égales à LF , FF ou CR + LF, sont aussi SUPPRIMÉES

19) SUPPRESSION de TOUTES les lignes COMMENÇANT par l'expression RÉGULIÈRE Re :

| Zone de RECHERCHE | Zone de REMPLACEMENT |
|---|---|
| ^Re.*\R? | RIEN |

NOTES :

Les lignes CONCERNÉES sont SUPPRIMÉES, QUEL que SOIT leur TYPE de 'FIN de LIGNE' et le TYPE du FICHIER

L'expression RÉGULIÈRE Re peut, aussi, être une SIMPLE chaîne LITTÉRALE

20) SUPPRESSION de TOUTES les lignes FINISSANT par l'expression RÉGULIÈRE Re :

| Zone de RECHERCHE | Zone de REMPLACEMENT |
|---|---|
| .*Re\R? | RIEN |

NOTES :

Les lignes CONCERNÉES sont SUPPRIMÉES, QUEL que SOIT leur TYPE de 'FIN de LIGNE' et le TYPE du FICHIER

L'expression RÉGULIÈRE Re peut, aussi, être une SIMPLE chaîne LITTÉRALE

21) SUPPRESSION de TOUTES les lignes COMPORTANT l'expression RÉGULIÈRE Re :

| Zone de RECHERCHE | Zone de REMPLACEMENT |
|---|---|
| .*Re.*\R? | RIEN |

NOTES :

Les lignes CONCERNÉES sont SUPPRIMÉES, QUEL que SOIT leur TYPE de 'FIN de LIGNE' et le TYPE du FICHIER

L'expression RÉGULIÈRE Re peut, aussi, être une SIMPLE chaîne LITTÉRALE

22) SUPPRESSION de TOUTES les lignes, NON VIDES, NE comportant PAS l'expression RÉGULIÈRE Re :

| Zone de RECHERCHE | Zone de REMPLACEMENT |
|---|---|
| ^(?!.*Re).+\R? | RIEN |

NOTE :

Les lignes CONCERNÉES sont SUPPRIMÉES, QUEL que SOIT leur TYPE de 'FIN de LIGNE' et le TYPE du FICHIER

L'expression RÉGULIÈRE Re peut, aussi, être une SIMPLE chaîne LITTÉRALE

23) SUPPRESSION de TOUTES les lignes NE comportant PAS l'expression RÉGULIÈRE Re :

A) SUPPRESSION des SEULES lignes VIDES :

| Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton 'Remplacer' |
|---|---|---|
| \R+ | \r\n | NON |
| \R* | RIEN | NON |

NOTES :

NE JAMAIS utiliser le bouton " Remplacer " ( Méthode au COUP par COUP ) pour cette SUPPRESSION

Les PREMIÈRE et DEUXIÈME lignes VIDES , ÉVENTUELLES, du FICHIER ne sont JAMAIS SUPPRIMÉES

Les lignes VIDES, égales à CR , LF ou FF, UNIQUEMENT, sont aussi SUPPRIMÉES

B) SUPPRESSION des lignes NON VIDES, NE comportant PAS l'expression RÉGULIÈRE Re :

| Zone de RECHERCHE | Zone de REMPLACEMENT |
|---|---|
| ^(?!.*Re).+\R? | RIEN |

NOTE :

Les lignes CONCERNÉES sont SUPPRIMÉES, QUEL que SOIT leur TYPE de 'FIN de LIGNE' et de FICHIER

L'expression RÉGULIÈRE Re peut, aussi, être une SIMPLE chaîne LITTÉRALE

24) SUPPRESSION de TOUTES les lignes d'EXACTEMENT n CARACTÈRES, NON 'FIN de LIGNE' :

| Zone de RECHERCHE | Zone de REMPLACEMENT |
|---|---|
| ^(?=.{n}$).+\R? | RIEN |

NOTES :

Les lignes CONCERNÉES sont SUPPRIMÉES, QUEL que SOIT leur TYPE de 'FIN de LIGNE' et de FICHIER

25) SUPPRESSION de TOUTES les lignes d'au MOINS n CARACTÈRES, NON 'FIN de LIGNE' :

| Zone de RECHERCHE | Zone de REMPLACEMENT |
|---|---|
| ^(?=.{n}).+\R? | RIEN |

NOTES :

Les lignes CONCERNÉES sont SUPPRIMÉES, QUEL que SOIT leur TYPE de 'FIN de LIGNE' et de FICHIER

26) SUPPRESSION de TOUTES les lignes d'au PLUS n CARACTÈRES, NON 'FIN de LIGNE' :

| Zone de RECHERCHE | Zone de REMPLACEMENT |
|---|---|
| ^(?=.{1,n}$).+\R? | RIEN |

NOTES :

Les lignes CONCERNÉES sont SUPPRIMÉES, QUEL que SOIT leur TYPE de 'FIN de LIGNE' et de FICHIER

27) SUPPRESSION de TOUT texte, à PARTIR de la PREMIÈRE occurrence de Re, dans CHAQUE ligne d'un FICHIER :

| | Zone de RECHERCHE | Zone de REMPLACEMENT | Recherche SEULE SANS Remplacement |
|---|---|---|---|
| (1) | Re.* | RIEN | Re.* |
| (2) | .*?\KRe.* | RIEN | |

NOTES :

L'expression RÉGULIÈRE Re peut, aussi, être une SIMPLE chaîne LITTÉRALE

Si l'expression RÉGULIÈRE Re NE doit PAS être SUPPRIMÉE :

- CAS 1 : Mettre (Re).* en partie RECHERCHE et \1 en partie REMPLACEMENT
- CAS 2 : Mettre .*?Re\K.* en partie RECHERCHE, et NE PAS utiliser le BOUTON " Remplacer "

28) SUPPRESSION de TOUT texte, à PARTIR de la DERNIÈRE occurrence de Re, dans CHAQUE ligne d'un FICHIER :

| | Zone de RECHERCHE | Zone de REMPLACEMENT |
|---|---|---|
| (1) | (.*)Re.* | \1 |
| (2) | (Re)(?!.*\1).* | RIEN |
| (3) | Re(?!.*Re).* | RIEN |
| (4) | .*\KRe.* | RIEN |

| Recherche SEULE SANS Remplacement |
|---|
| (Re)(?!.*\1).* |
| Re(?!.*Re).* |

NOTES :

L'expression RÉGULIÈRE Re peut, aussi, être une SIMPLE chaîne LITTÉRALE

Si l'expression RÉGULIÈRE Re NE doit PAS être SUPPRIMÉE :

- CAS 1 : Mettre (.*Re).* en partie RECHERCHE
- CAS 2 : Mettre \1 en partie REMPLACEMENT
- CAS 3 : Mettre (Re)(?!.*Re).* en partie RECHERCHE et \1 en partie REMPLACEMENT
- CAS 4 : Mettre .*Re\K.* en partie RECHERCHE, et NE PAS utiliser le BOUTON " Remplacer "

29) SUPPRESSION de TOUT texte, JUSQU'à la PREMIÈRE occurrence de Re, dans CHAQUE ligne d'un FICHIER :

| Zone de RECHERCHE | Zone de REMPLACEMENT | Recherche SEULE SANS Remplacement |
|---|---|---|
| .*?Re(.*) | \1 | ^.*?Re |

NOTES :

L'expression RÉGULIÈRE Re peut, aussi, être une SIMPLE chaîne LITTÉRALE

Si l'expression RÉGULIÈRE Re NE doit PAS être SUPPRIMÉE, mettre .*?(Re.*) en partie RECHERCHE

30) SUPPRESSION de TOUT texte, JUSQU'à la DERNIÈRE occurrence de Re, dans CHAQUE ligne d'un FICHIER :

| Zone de RECHERCHE | Zone de REMPLACEMENT | Recherche SEULE SANS Remplacement |
|---|---|---|
| .*Re | RIEN | .*Re |

NOTES :

L'expression RÉGULIÈRE Re peut, aussi, être une SIMPLE chaîne LITTÉRALE

Si l'expression RÉGULIÈRE Re NE doit PAS être SUPPRIMÉE, mettre .*(Re) en partie RECHERCHE et

\1 en partie REMPLACEMENT

31) SUPPRESSION de TOUT texte, de la PREMIÈRE à la DERNIÈRE occurrence de Re, dans CHAQUE ligne d'un FICHIER :

| | Zone de RECHERCHE | Zone de REMPLACEMENT | Recherche SEULE SANS Remplacement |
|---|---|---|---|
| (1) | (Re).*\1 | RIEN | (Re).*\1 |
| (2) | Re.*Re | RIEN | Re.*Re |

NOTES :

L'expression RÉGULIÈRE Re peut, aussi, être une SIMPLE chaîne LITTÉRALE

Si l'expression RÉGULIÈRE Re est SIMPLE et LITTÉRALE, employer, de PRÉFÉRENCE, le CAS 2

Si l'expression RÉGULIÈRE Re NE doit PAS être SUPPRIMÉE, mettre \1\1 en partie REMPLACEMENT

32) SUPPRESSION de TOUT texte, ENTRE DEUX occurrences SUCCESSIVES de Re dans CHAQUE ligne d'un FICHIER :

| | Zone de RECHERCHE | Zone de REMPLACEMENT | Recherche SEULE SANS Remplacement |
|---|---|---|---|
| (1) | (?<=(Re)).*?(?=\1) | RIEN | (?<=(Re)).*?(?=\1) |
| (2) | (?<=Re).*?(?=Re) | RIEN | (?<=Re).*?(?=Re) |

NOTES :

L'expression RÉGULIÈRE Re peut, aussi, être une SIMPLE chaîne LITTÉRALE

Si l'expression RÉGULIÈRE Re est SIMPLE et LITTÉRALE, employer, de PRÉFÉRENCE, le CAS 2

33) SUPPRESSION de la Nème OCCURRENCE de Re dans CHAQUE ligne d'un FICHIER :

| | Zone de RECHERCHE | Zone de REMPLACEMENT |
|---|---|---|
| (1) | ^((?:.*?(Re)){N-1}.*?)(?2) | \1 |
| (2) | ^((?:.*?(Re)){N-1}.*?)\2 | \1 |
| (3) | ^((?:.*?Re){N-1}.*?)Re | \1 |

| | Zone de | Zone de | Recherche SEULE |
|---|---|---|---|

| | RECHERCHE | REMPLACEMENT | SANS Remplacement |
|---|---|---|---|
| (4) | ^(?:.*?(Re)){N-1}.*?\K(?1) | RIEN | ^(?:.*?(Re)){N-1}.*?\K(?1) |
| (5) | ^(?:.*?(Re)){N-1}.*?\K\1 | RIEN | ^(?:.*?(Re)){N-1}.*?\K\1 |
| (6) | ^(?:.*?Re){N-1}.*?\KRe | RIEN | ^(?:.*?Re){N-1}.*?\KRe |

NOTES :

L'expression RÉGULIÈRE Re peut, aussi, être une SIMPLE chaîne LITTÉRALE

Si l'expression RÉGULIÈRE Re est LITTÉRALE, employer, de PRÉFÉRENCE, le CAS 2 ou 5

Si l'expression RÉGULIÈRE Re est SIMPLE et LITTÉRALE, employer, de PRÉFÉRENCE, le CAS 3 ou 6

Pour REMPLACER la Nème OCCURRENCE de l'expression RÉGULIÈRE 'Re', de CHAQUE ligne, par la CHAÎNE Ch, mettre :

- \1Ch , en partie REMPLACEMENT, pour les CAS 1, 2 ou 3
- Ch , en partie REMPLACEMENT, pour les CAS 4, 5 ou 6

34) SUPPRESSION de la Nème OCCURRENCE de Re d'un FICHIER :

| | Zone de RECHERCHE | Zone de REMPLACEMENT |
|---|---|---|
| (1) | (?s)((?:.*?(Re)){N-1}.*?)(?2) | \1 |
| (2) | (?s)((?:.*?(Re)){N-1}.*?)\2 | \1 |
| (3) | (?s)((?:.*?Re){N-1}.*?)Re | \1 |

| | Zone de RECHERCHE | Zone de REMPLACEMENT | Recherche SEULE SANS Remplacement |
|---|---|---|---|
| (4) | (?s)(?:.*?(Re)){N-1}.*?\K(?1) | RIEN | (?s)(?:.*?(Re)){N-1}.*?\K(?1) |
| (5) | (?s)(?:.*?(Re)){N-1}.*?\K\1 | RIEN | (?s)(?:.*?(Re)){N-1}.*?\K\1 |
| (6) | (?s)(?:.*?Re){N-1}.*?\KRe | RIEN | (?s)(?:.*?Re){N-1}.*?\KRe |

IMPORTANT :

Normalement, l'ANCRE \A, désignant le TOUT DÉBUT du fichier, DEVRAIT être employée, à la PLACE de ^,

pour cette RECHERCHE-REMPLACEMENT

Dû au BUG de NOTEPAD++, concernant ASSERTION \A , AUCUNE assertion de POSITION n'a été indiquée

Pour un comportant CORRECT, il suffit de REPLACER le CURSEUR au TOUT DÉBUT du fichier COURANT,

avec le RACCOURCI CTRL + Origine, AVANT l'exécution de l'expression RÉGULIÈRE

En fait, cette RegExp cherche TOUJOURS la Nème OCCURRENCE de Re, à PARTIR de la POSITION du CURSEUR

NOTES :

L'expression RÉGULIÈRE Re peut, aussi, être une SIMPLE chaîne LITTÉRALE

Si l'expression RÉGULIÈRE Re est LITTÉRALE, employer, de PRÉFÉRENCE, le CAS 2 ou 5

Si l'expression RÉGULIÈRE Re est SIMPLE et LITTÉRALE, employer, de PRÉFÉRENCE, le CAS 3 ou 6

Pour REMPLACER la Nème OCCURRENCE de l'expression RÉGULIÈRE 'Re', d'un FICHIER , par

la CHAÎNE Ch, mettre :

- \1Ch , en partie REMPLACEMENT, pour les CAS 1, 2 ou 3
- Ch , en partie REMPLACEMENT, pour les CAS 4, 5 ou 6

35) SUPPRESSION de TOUTE expression RÉGULIÈRE Re, entre les COLONNES n et m, dans TOUTES les lignes d'un FICHIER :

| | Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton 'Remplacer' |
|---|---|---|---|
| (1) | ^(.{n-1,m-1})Re | \1 | Oui |
| (2) | ^.{n-1,m-1}\KRe | RIEN | NON |

| Recherche SEULE SANS Remplacement |
|---|
| ^.{n-1,m-1}\KRe |

IMPORTANT :

Le TEXTE, RELATIF à l'expression RÉGULIÈRE Re, NE doit apparaître qu'UNE FOIS, entre les COLONNES n et m

Dans le cas CONTRAIRE, c'est l'OCCURRENCE la PLUS PROCHE de la COLONNE m qui sera prise en COMPTE

NOTES :

NE JAMAIS utiliser le bouton " Remplacer " ( Méthode au COUP par COUP ) dans le CAS 2

L'expression RÉGULIÈRE Re peut, aussi, être une SIMPLE chaîne LITTÉRALE

Pour REMPLACER le TEXTE, RELATIF à l'expression RÉGULIÈRE Re, trouvée ENTRE les COLONNES n et m,

UNIQUEMENT, indiquer le NOUVEAU texte dans la zone de REMPLACEMENT

36) SUPPRESSION de TOUTE expression RÉGULIÈRE Re, en COLONNE n, dans TOUTES les lignes d'un FICHIER :

| | Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton 'Remplacer' |
|---|---|---|---|
| (1) | ^(.{n-1})Re | \1 | Oui |
| (2) | ^.{n-1}\KRe | RIEN | NON |

| Recherche SEULE SANS Remplacement |
|---|
| ^.{n-1}\KRe |

NOTES :

NE JAMAIS utiliser le bouton " Remplacer " ( Méthode au COUP par COUP ) dans le CAS 2

L'expression RÉGULIÈRE Re peut, aussi, être une SIMPLE chaîne LITTÉRALE

Pour REMPLACER le TEXTE, RELATIF à l'expression RÉGULIÈRE Re, trouvée en COLONNE n, UNIQUEMENT,

indiquer le NOUVEAU texte dans la zone de REMPLACEMENT

37) SUPPRESSION de TOUTE RegExp Re, SÉPARÉE de la FIN de LIGNE par n à m CARACTÈRES, dans TOUTES les lignes d'un FICHIER :

| Zone de RECHERCHE | Zone de REMPLACEMENT | Recherche SEULE SANS Remplacement |
|---|---|---|
| Re(.{n,m})$ | \1 | Re(?=.{n,m}$) |

IMPORTANT :

Le TEXTE, RELATIF à l'expression RÉGULIÈRE Re, NE doit apparaître qu'UNE FOIS, SÉPARÉE, de la FIN de

LIGNE, par n à m CARACTÈRES. Dans le cas CONTRAIRE, c'est l'occurrence la PLUS ÉLOIGNÉE de la FIN

de LIGNE qui sera prise en COMPTE

NOTES :

L'expression RÉGULIÈRE Re peut, aussi, être une SIMPLE chaîne LITTÉRALE

Pour REMPLACER le TEXTE, RELATIF à l'expression RÉGULIÈRE Re, SÉPARÉE de la FIN de LIGNE, de n à m CARACTÈRES UNIQUEMENT, indiquer le NOUVEAU texte dans la zone de REMPLACEMENT

38) SUPPRESSION de TOUTE RegExp Re, SÉPARÉE de la FIN de LIGNE par n CARACTÈRES, dans TOUTES les lignes d'un FICHIER :

| Zone de RECHERCHE | Zone de REMPLACEMENT | Recherche SEULE SANS Remplacement |
|---|---|---|
| Re(.{n})$ | \1 | Re(?=.{n}$) |

NOTES :

L'expression RÉGULIÈRE Re peut, aussi, être une SIMPLE chaîne LITTÉRALE

Pour REMPLACER le TEXTE, RELATIF à l'expression RÉGULIÈRE Re, SÉPARÉE de la FIN de LIGNE, par n CARACTÈRES UNIQUEMENT, indiquer le NOUVEAU texte dans la zone de REMPLACEMENT

39) SUPPRESSION de TOUT caractère, en COLONNE n, dans TOUTES les lignes d'un FICHIER :

| | Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton 'Remplacer' |
|---|---|---|---|
| (1) | ^(.{n-1}). | \1 | Oui |
| (2) | (?<=^.{n-1}). | RIEN | NON |
| (3) | ^.{n-1}\K. | RIEN | NON |

| Recherche SEULE SANS Remplacement |
|---|
| ^.{n-1}\K. |

NOTES :

NE JAMAIS utiliser le bouton " Remplacer " ( Méthode au COUP par COUP ) dans les CAS 2 et 3

Pour REMPLACER le caractère, en COLONNE n, UNIQUEMENT, indiquer le NOUVEAU caractère ou TEXTE dans la zone de REMPLACEMENT

Le CAS 2, correspond à un LookBEHIND POSITIF de longueur n -1 FIXE, qui implique une zone OBLIGATOIRE de n-1 caractères, en DÉBUT de ligne, AVANT la COLONNE n à SUPPRIMER

LE CAS 3, à la MÊME signification, mais utilise la CONSTRUCTION \K, qui permet d'OUBLIER la PARTIE du GABARIT de RECHERCHE, située AVANT \K

40) SUPPRESSION de TOUT texte, ENTRE les COLONNES n et m, dans TOUTES les lignes d'un FICHIER :

| | Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton 'Remplacer' |
|---|---|---|---|
| (1) | ^(.{n-1}).{1,m-n+1} | \1 | Oui |
| (2) | (?<=^.{n-1}).{1,m-n+1} | RIEN | NON |
| (3) | ^.{n-1}\K.{1,m-n+1} | RIEN | NON |

| Recherche SEULE SANS Remplacement |
|---|
| ^.{n-1}\K.{1,m-n+1} |

NOTES :

NE JAMAIS utiliser le bouton " Remplacer " ( Méthode au COUP par COUP ) dans les CAS 2 et 3

Pour REMPLACER le TEXTE, compris ENTRE les COLONNES n et m, UNIQUEMENT, indiquer le NOUVEAU texte dans la zone de REMPLACEMENT

SEULES, les lignes de n CARACTÈRES ou PLUS sont donc MODIFIÉES

Le CAS 2, correspond à un LookBEHIND POSITIF de longueur n -1 FIXE, qui implique une zone OBLIGATOIRE de n-1 caractères, en DÉBUT de ligne, AVANT le bloc de COLONNES à SUPPRIMER

Le CAS 3, à la MÊME signification, mais utilise la CONSTRUCTION \K, qui permet d'OUBLIER la PARTIE

du GABARIT de RECHERCHE, située AVANT \K

41) SUPPRESSION de TOUT texte, à COMPTER de la COLONNE n, dans TOUTES les lignes d'un FICHIER :

| | Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton 'Remplacer' | |
|---|---|---|---|---|
| (1) | ^(.{n-1}).+ | \1 | Oui | Recherche SEULE SANS Remplacement |
| (2) | ^.{n-1}\K.+ | RIEN | NON | ^.{n-1}\K.+ |

NOTES :

NE JAMAIS utiliser le bouton " Remplacer " ( Méthode au COUP par COUP ) dans le CAS 2

Pour REMPLACER le TEXTE, à COMPTER de la COLONNE n, de CHAQUE ligne, indiquer le NOUVEAU TEXTE dans la zone de REMPLACEMENT, APRÈS \1

SEULES, les lignes de n CARACTÈRES ou PLUS sont donc MODIFIÉES

42) SUPPRESSION de TOUT texte, JUSQU'à la COLONNE n, dans TOUTES les lignes d'un FICHIER :

| Zone de RECHERCHE | Zone de REMPLACEMENT | Recherche SEULE SANS Remplacement |
|---|---|---|
| ^.{1,n}(.*) | \1 | ^.{1,n} |

NOTES :

Si la LIGNE fait MOINS de n CARACTÈRES, elle est REMPLACÉE par une ligne VIDE

Pour REMPLACER le TEXTE, JUSQU'à la COLONNE n, de CHAQUE ligne, indiquer le NOUVEAU TEXTE dans la zone de REMPLACEMENT, AVANT \1

43) SUPPRESSION de TOUT texte, en DEHORS de la ZONE de COLONNES n à m, dans TOUTES les lignes d'un FICHIER :

| | Zone de RECHERCHE | Zone de REMPLACEMENT | Recherche SEULE SANS Remplacement |
|---|---|---|---|
| (1) | ^.{n-1}+(.{1,m-n+1}).*\|.{1,n-1}$ | (?1\1) | ^.{1,n-1}\|.{m-n+1}\K.+ |
| (2) | ^.{n-1}+(.{1,m-n+1}).*\|.{1,n-1}$\R | (?1\1) | |

NOTES :

Dans le CAS 1, si la LIGNE fait MOINS de n CARACTÈRES, elle est REMPLACÉE par une ligne VIDE

Dans le CAS 2, si la LIGNE fait MOINS de n CARACTÈRES, elle est SUPPRIMÉE

Pour MODIFIER le TEXTE, ENTRE les COLONNES n et m, de CHAQUE ligne, indiquer le(s) NOUVEAU(X)
TEXTE(S) dans la zone de REMPLACEMENT, AVANT et/ou APRÈS la RÉFÉRENCE ARRIÈRE \1

EXPLICATIONS du CAS n° 2 :

- ^.{n-1}+ => Recherche OBLIGATOIRE de n-1 CARACTÈRES en DÉBUT de LIGNE
- (.{1,m-n+1}) => Recherche de la CHAÎNE, NON VIDE, de TOUT CARACTÈRE, ENTRE les COLONNES n et m
  Celle-ci est mise ENTRE PARENTHÈSES => Elle est REPRÉSENTÉE par le Groupe 1
- .* => Recherche de TOUS les caractères RESTANTS, ÉVENTUELS, de la LIGNE
- | => Symbole de l'ALTERNATION
- .{1,n-1}$\R => Recherche de n-1 caractères MAXIMUM, SUIVIS d'une 'FIN de LIGNE'
- (?1\1) => Si le GROUPE 1 EXISTE ( CHAÎNE ENTRE m et n ), on le RÉÉCRIT
  Si le GROUPE 1 N'existe PAS ( LIGNE de MOINS de n CARACTÈRES ), on REMPLACE par
  la chaîne VIDE ( PAS de partie ELSE, avec le SIGNE : )

44) AJOUT d'une CHAÎNE Ch, TOUS les n CARACTÈRES, dans TOUTES les lignes d'un FICHIER :

| | Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton 'Remplacer' |
|---|---|---|---|
| (1) | (.{n}) | \1Ch | Oui |
| (2) | .{n} | $0Ch | Oui |
| (3) | .{n}\K | Ch | NON |

NOTES :

NE JAMAIS utiliser le bouton " Remplacer " ( Méthode au COUP par COUP ) dans le CAS 3

La CHAÎNE Ch peut, aussi, être un caractère UNIQUE de SÉPARATION, du genre '|'

45) AJOUT d'une 'FIN de LIGNE', TOUS les n CARACTÈRES, dans TOUTES les lignes d'un FICHIER :

| | Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton 'Remplacer' |
|---|---|---|---|
| (1) | (.{n}) | \1\r\n | Oui |
| (2) | .{n} | $0\r\n | Oui |

| | Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton 'Remplacer' |
|---|---|---|---|
| (3) | .{n}\K | \r\n | NON |

| | Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton 'Remplacer' |
|---|---|---|---|
| (4) | (.{n})(?!\R) | \1\r\n | Oui |
| (5) | .{n}(?!\R) | $0\r\n | Oui |
| (6) | .{n}(?!\R)\K | \r\n | NON |

NOTES :

NE JAMAIS utiliser le bouton " Remplacer " ( Méthode au COUP par COUP ) dans les CAS 3 et 6

Les CAS 4 à 6, sont IDENTIQUES aux CAS 1 à 3, mais ÉVITENT l'AJOUT de 'FIN de LIGNE', NON désirées :

il n'y a AJOUT de 'FIN de LIGNE', QUE lorsque la ligne COURANTE contient PLUS de n CARACTÈRES !

Pour un FICHIER de TYPE 'UNIX', on remplacera les DEUX caractères \r\n par le SEUL caractère \n

Pour un FICHIER de TYPE 'MAC', on remplacera les DEUX caractères \r\n par le SEUL caractère \r

46) EXTRACTION du SEUL texte, SATISFAISANT la RegExp Re, de TOUTES les lignes d'un FICHIER :

| | Zone de RECHERCHE | Zone de REMPLACEMENT |
|---|---|---|
| (1) | (?s).*?(Re)\|.*\z | (?1\1\r\n) |
| (2) | (?s).*?(Re)(\R)?\|.*\z | (?1\1(?2\2:\r\n)) |

NOTES :

CHAQUE texte, SATISFAISANT la RegExp Re est RÉÉCRIT sur une NOUVELLE ligne

Le CAS 2 est PRÉFÉRABLE, car il ÉVITE l'AJOUT d'une 'FIN de LIGNE' si la RegExp Re TERMINE la ligne

Pour un FICHIER de TYPE 'UNIX', on remplacera les DEUX caractères \r\n par le SEUL caractère \n

Pour un FICHIER de TYPE 'MAC', on remplacera les DEUX caractères \r\n par le SEUL caractère \r

Pour EXTRAIRE TOUT texte, ENTRE 2 DÉLIMITEURS IDENTIQUES, comme "....." ou '......' ,, il sera PRÉFÉRABLE de vérifier, au PRÉALABLE, s'il n'existe pas de DÉLIMITEUR ORPHELIN, par la RECHERCHE n° 47, qui suit

Pour EXTRAIRE TOUT texte, ENTRE 2 DÉLIMITEURS DIFFÉRENTS, comme [.....] ou <.......> , il sera PRÉFÉRABLE de vérifier, au PRÉALABLE, s'il n'existe pas de DÉLIMITEUR ORPHELIN, par la RECHERCHE n° 48, ci-APRÈS

On peut EXTRAIRE un SOUS-ensemble, SEULEMENT, de CHAQUE texte, SATISFAISANT l'expression RÉGULIÈRE Re

EXEMPLE :

Pour EXTRAIRE la VALEUR de TOUS les CHAMPS 'name' de 'NativeLang.xml', SANS les GUILLEMETS :

- GÉNÉRER l'expression RÉGULIÈRE Re et GROUPER la ZONE que l'on désire EXTRAIRE :

  name .+?"(.+?)" , avec 2 ESPACES, placés DEVANT et APRÈS 'name' ( SEULE la partie ENTRE les DEUX GUILLEMETS, est à EXTRAIRE, dans cet EXEMPLE )

- GÉNÉRER la zone de RECHERCHE associée à la RegExp Re : (?s).*? name .+?"(.+?)"(\R)?|.*\z

- RÉÉCRIRE la zone de REMPLACEMENT, qui NE change PAS : (?1\1(?2\2:\r\n))

- DÉCOCHER, si nécessaire, la CASE " Respecter la casse " et COCHER la CASE " Boucler "

- Se REMETTRE en DÉBUT de FICHIER et CLIQUER sur le BOUTON " Remplacer tout " ( ~ 2 sec. )

EXPLICATIONS sur le CAS GÉNÉRAL n° 2 :

- (?s) => Recherche MULTI-LIGNES ( Le JOKER (.) trouve AUSSI les 'FINS de LIGNE'

- .*?(Re) => Recherche de la 1ÈRE RegExp Re, avec TOUT ce qui la PRÉCÈDE

  Re est mise ENTRE PARENTHÈSES => Elle est REPRÉSENTÉE par le Groupe 1

- (\R)? => Recherche d'1 CARACTÈRE de type 'FIN de LIGNE', ÉVENTUEL, APRÈS la RegExp Re

  \R est mis ENTRE PARENTHÈSES => Il est REPRÉSENTÉ par le Groupe 2

- | => Symbole de l'ALTERNATION

- .*\z => Si PAS de RegExp Re ( FIN fichier ), on prend, alors, TOUS les caractères RESTANTS

- (?1\1...) => Si le GROUPE 1 EXISTE ( la RegExp Re ), on le RÉÉCRIT et on EXÉCUTE la partie ....

  Si le GROUPE 1 N'existe PAS ( FIN de fichier ) on REMPLACE par la chaîne VIDE

  ( PAS de partie ELSE, avec le SIGNE : )

- (?2\2:\r\n) => Si le Groupe 2 EXISTE ( Une 'FIN de LIGNE' APRÈS la RegExp Re ) on le RÉÉCRIT (\2)

  Si le Groupe 2 N'existe PAS ( PAS de 'FIN de LIGNE' APRÈS la RegExp Re ), on GÉNÈRE

  un PASSAGE automatique à la ligne SUIVANTE par la RÉÉCRITURE de '\r\n'

47) RECHERCHE du PREMIER DÉLIMITEUR #, APRÈS n COUPLES #.....# ( avec n >=0 ), dans CHAQUE ligne du fichier COURANT :

| | Recherche SEULE<br>SANS Remplacement |
|---|---|
| (1) | ^(([^#\r\n]*?#)(?2))*+.*\K# |
| (2) | ^(?:([^#\r\n]*?#)(?1))*+.*\K# |

NOTES :

Cette RECHERCHE permet d'IDENTIFIER :

- des ZONES de TEXTE, BORNÉES par un SEUL caractère DÉLIMITEUR, telles que "....." ou '.....' ,
  qui sont MAL appariées ( Délimiteur MANQUANT ou EXCÉDENTAIRE )

- des LIGNES avec une SEULE occurrence de ce DÉLIMITEUR

Dans le TABLEAU ci-dessus, le DÉLIMITEUR '#' peut être REMPLACÉ par n'IMPORTE QUEL AUTRE caractère
Cependant, s'il est, ÉGALEMENT, un MÉTA-caractère de RECHERCHE, placer un CARACTÈRE '\' DEVANT

Le DÉLIMITEUR, à chercher, est indiqué TROIS fois dans CETTE expression RÉGULIÈRE

Employer, de PRÉFÉRENCE, le CAS 2, avec groupe NON CAPTURANT, qui permet une recherche PLUS RAPIDE,
en particulier, sur un FICHIER de GRANDE taille

48) RECHERCHE du PREMIER DÉLIMITEUR # ou @, APRÈS n COUPLES #.....@ ( avec n >=0 ), dans CHAQUE ligne du fichier COURANT :

| | Recherche SEULE<br>SANS Remplacement |
|---|---|
| (1) | ^(([^#@\r\n]*?)#(?2)@)*+.*\K[#@] |
| (2) | ^(?:([^#@\r\n]*?)#(?1)@)*+.*\K[#@] |

NOTES :

Cette RECHERCHE permet d'IDENTIFIER :

- des ZONES de TEXTE, BORNÉES par DEUX caractères DÉLIMITEURS, telles que [......] ou <......> ,
  qui sont MAL appariées ( Délimiteur MANQUANT ou EXCÉDENTAIRE ou délimiteurs INVERSÉS )

- des LIGNES avec une SEULE occurrence d'UN des DEUX DÉLIMITEURS

Dans le TABLEAU ci-dessus, les DÉLIMITEURS '#' et/ou '@' peuvent être REMPLACÉS par d'AUTRES caractères
Cependant, s'ils sont, ÉGALEMENT, des MÉTA-caractères de RECHERCHE, placer un CARACTÈRE '\' DEVANT

Les DEUX DÉLIMITEURS, # et @, sont indiqués, CHACUN, TROIS fois dans CETTE expression RÉGULIÈRE

Employer, de PRÉFÉRENCE, le CAS 2, avec groupe NON CAPTURANT, qui permet une recherche PLUS RAPIDE,

en particulier, sur un FICHIER de GRANDE taille

TRÈS IMPORTANT :

Cette RECHERCHE est INOPÉRANTE, en cas de BLOCS #.....@ IMBRIQUÉS, tels que #....#-----@.........@,

Elle fonctionne, UNIQUEMENT, avec une SUITE QUELCONQUE de BLOCS #.....@, INCLUS dans CHAQUE ligne

=> Dans le cas de BLOCS IMBRIQUÉS, il serait NÉCESSAIRE de faire APPEL aux expressions RÉGULIÈRES RÉCURSIVES

49) INTERVERSION de DEUX zones de TEXTE CONTIGUËS dans TOUTES les lignes d'un FICHIER :

PRÉAMBULE :

CHAQUE ligne du fichier COURANT est DÉCOUPÉE en QUATRE zones 1, 2 ,3 et 4 :

- la PREMIÈRE zone COMMENCE à la COLONNE 1 de la ligne COURANTE
- la DEUXIÈME zone COMMENCE à la COLONNE x de la ligne COURANTE
- la TROISIÈME zone COMMENCE à la COLONNE y de la ligne COURANTE
- la QUATRIÈME zone COMMENCE à la COLONNE z de la ligne COURANTE

La zone 1 DÉBUTE la ligne COURANTE et la zone 4 TERMINE la ligne COURANTE

APRÈS remplacement, les ZONES 2 et 3 sont ÉCHANGÉES et CHAQUE ligne est COMPOSÉE des zones 1, 3, 2 et 4

| | Zone de RECHERCHE | Zone de REMPLACEMENT |
|---|---|---|
| (1) | ^(.{x-1})(.{y-x)(.{z-y}) | \1\3\2 |
| (2) | ^(.{x-1})(.{y-x)(.{1,z-y}) | \1\3\2 |

NOTES :

La QUATRIÈME zone, en FIN de LIGNE, peut être ABSENTE et, dans ce cas, la VALEUR z correspond à la DERNIÈRE colonne de la ZONE 3, AUGMENTÉE de 1

=> la LARGEUR de CHAQUE ligne DOIT être SUPÉRIEURE ou ÉGALE à la VALEUR z-1 dans le CAS 1

Si, de PLUS, la largeur, NON NULLE, de la TROISIÈME zone est VARIABLE, on emploiera plutôt le CAS 2

Dans ce CAS, la LARGEUR de CHAQUE ligne DOIT, alors, être SUPÉRIEURE ou ÉGALE à la VALEUR y

50) DÉPLACEMENT et/ou SUPPRESSION de ZONES de TEXTE, DÉTERMINÉES en COLONNES ( Cas GÉNÉRAL ) :

HYPOTHÈSES :

CHAQUE ligne du fichier COURANT est DÉCOUPÉE en ZONES 1, 2 ,3 ....., n

- la PREMIÈRE zone COMMENCE à la COLONNE a de la ligne COURANTE ( => a = 1 )
- la DEUXIÈME zone COMMENCE à la COLONNE b de la ligne COURANTE
- la TROISIÈME zone COMMENCE à la COLONNE c de la ligne COURANTE

  ..... .....

- l'AVANT-DERNIÈRE zone COMMENCE à la COLONNE x de la ligne COURANTE
- la DERNIÈRE zone n COMMENCE à la COLONNE y de la ligne COURANTE

- la zone SUIVANTE, FICTIVE, COMMENCE à la COLONNE z de la ligne COURANTE

TOUTES les ZONES de texte, EXCEPTÉE la DERNIÈRE, possèdent une LARGEUR FIXE

La DERNIÈRE zone n peut avoir, au CHOIX, une longueur FIXE ou VARIABLE :

Si la ZONE n a une largeur FIXE, la LONGUEUR de CHAQUE ligne est ÉGALE à la VALEUR z-1

Si la DERNIÈRE zone n a une largeur VARIABLE, NON NULLE, la LONGUEUR des LIGNES DOIT être >= VALEUR y

| Zone de RECHERCHE | Zone de REMPLACEMENT |
|---|---|
| ^(.{b-1})(.{c-b})(.{d-c}).....(.{y-x})(.{1,z-y}) | Voir ci-DESSOUS |

Zone 1 Zone 2 Zone 3 ..... Zone n-1 Zone n

NOTES :

En partie REMPLACEMENT, CHACUNE des n ZONES, \1, \2, ..., \n , peut être ABSENTE ou PRÉSENTE, à une position QUELCONQUE, par RAPPORT aux zones ADJACENTES

EXEMPLES :

\4\3 , en partie REMPLACEMENT, ne CONSERVE, dans un FICHIER découpé en 5 ZONES, QUE les ZONES 3 et 4, INTERVERTIES, en OMETTANT les 2 PREMIÈRES et la DERNIÈRE zone de CHAQUE ligne

N° \1|\2|\3 \: \n , en partie REMPLACEMENT, ne CONSERVE, dans un FICHIER découpé en n ZONES, QUE les 3 PREMIÈRES et la DERNIÈRE zones avec :

- la CHAÎNE 'N° ' AVANT la ZONE 1
- le SÉPARATEUR '|' ENTRE les ZONES 1 et 2 et ENTRE les ZONES 2 et 3
- la CHAÎNE ' : ' ENTRE les ZONES 3 et n

Le SIGNE : est un MÉTA-CARACTÈRE, en partie REMPLACEMENT

=> il est donc ÉCRIT sous la FORME \:

51) DÉPLACEMENT et/ou SUPPRESSION de ZONES de TEXTE, DÉTERMINÉES par un SÉPARATEUR ( Cas GÉNÉRAL ) :

HYPOTHÈSES :

On UTILISE, à cet effet, un caractère SÉPARATEUR quelconque, qui n'EXISTE PAS encore dans le fichier COURANT

CHAQUE ligne du fichier COURANT est alors DÉCOUPÉE en ZONES 1, 2 ,3 ....., n, et CHACUNE d'elles, EXCEPTÉ

la DERNIÈRE zone n, est SÉPARÉE de la SUIVANTE par ce caractère SÉPARATEUR

=> Il y a donc, OBLIGATOIREMENT, n-1 SÉPARATEURS dans CHAQUE ligne du fichier COURANT

| | Zone de RECHERCHE | Zone de REMPLACEMENT |
|---|---|---|
| (1) | (.*?#)(.*?#)(.*?#).....(.*?#)(.*) | Voir ci-DESSOUS |
| (2) | (.*?)#(.*?)#(.*?)#.....(.*?)#(.*) | Voir ci-DESSOUS |

Zn 1 Zn 2 Zn 3 ..... Zn n-1 Zn n

NOTES :

Dans le TABLEAU ci-dessus, le SÉPARATEUR '#' peut être REMPLACÉ par n'IMPORTE QUEL AUTRE caractère

Cependant, s'il est, ÉGALEMENT, un MÉTA-caractère de RECHERCHE, placer le CARACTÈRE '\' DEVANT

Si le caractère SÉPARATEUR, NE doit PAS être CONSERVÉ, APRÈS remplacement, on utilisera plutôt le CAS 2

En partie REMPLACEMENT, CHACUNE des n ZONES, \1, \2, ..., \n , peut être ABSENTE ou PRÉSENTE, à une position QUELCONQUE, par RAPPORT aux zones ADJACENTES

EXEMPLES :

\4#\3 , en partie REMPLACEMENT, avec le CAS 1, ne CONSERVE, dans un FICHIER découpé en 4 ZONES, QUE les ZONES 3 et 4, INTERVERTIES, en OMETTANT les 2 PREMIÈRES zones de CHAQUE ligne

REMARQUES :

- la DERNIÈRE zone 4, NE finissant PAS par le SÉPARATEUR '#', il est AJOUTÉ, APRÈS \4
- Si nécessaire, SUPPRIMER le SÉPARATEUR '#', en FIN de ligne, suite RECOPIE de \3

--> \1 \2 \3\(\n\) , en partie REMPLACEMENT, avec le CAS 2, ne CONSERVE, dans un FICHIER découpé en n ZONES, QUE les 3 PREMIÈRES et la DERNIÈRE zones avec :

- la CHAÎNE '--> ' AVANT la ZONE 1

- un caractère ESPACE ' ' ENTRE les ZONES 1 et 2 et ENTRE les ZONES 2 et 3

- la ZONE 3, ENTOURÉE par l'ensemble PARENTHÈSE OUVRANTE - PARENTHÈSE FERMANTE

Les SIGNES '(' et ')' sont des MÉTA-CARACTÈRES, en partie REMPLACEMENT

=> ils sont donc ÉCRITS sous la FORME \( et \)

52) SUPPRESSION des MOTS IDENTIQUES, EXCÉDENTAIRES, de CHAQUE ligne d'un FICHIER :

| Zone de RECHERCHE | Zone de REMPLACEMENT |
|---|---|
| (\b(\w+)\b.*?),\2(,\|\R) | \1\3 |

NOTES :

TOUS les mots EXCÉDENTAIRES, situés à DROITE du mot MULTIPLE COURANT, sont SUPPRIMÉS, dans CHAQUE ligne

La VIRGULE, figurant DEUX fois, dans le TABLEAU ci-dessus, est le SÉPARATEUR des MOTS, dans CHAQUE ligne

Celui-ci peut être REMPLACÉ par n'IMPORTE QUEL AUTRE caractère

Cependant, s'il est, ÉGALEMENT, un MÉTA-caractère de RECHERCHE, placer le caractère '\' DEVANT

Cette RECHERCHE-REMPLACEMENT peut se faire, en une SEULE PASSE, si on utilise le remplacement RÉCURSIF de la FENÊTRE " Find/Replace " ( CTRL + R ) du plugin " TextFX "

- Se POSITIONNER à COMPTER de la ZONE du fichier COURANT à explorer ou au DÉBUT du fichier
- SÉLECTIONNER la FENÊTRE " Find/Replace " ( CTRL + R )
- COPIER les zones de RECHERCHE et de REMPLACEMENT, du TABLEAU ci-dessus
- COCHER les DEUX cases " Regular Expr " et " Recurse Repl "
- COCHER, si nécessaire , la CASE " Wrap " pour une RECHERCHE sur TOUT le fichier
- CLIQUER sur le BOUTON " Find " pour SÉLECTIONNER la 1ÈRE occurrence TROUVÉE de la RegExp
- CLIQUER sur le BOUTON " Replace Rest " pour EXÉCUTER un remplacement GLOBAL, sur la zone CHOISIE ou sur TOUT le fichier COURANT

REMARQUE :

L'utilisation du BOUTON " Repl&FAgain " ( REMPLACEMENT au COUP par COUP ), n'est PAS POSSIBLE, suite BUG !

Si on utilise la boite de dialogue HABITUELLE ( CTRL + F ) et l'onglet " Remplacer ", la RECHERCHE-REMPLACEMENT se fera en PLUSIEURS PASSES et portera NÉCESSAIREMENT sur TOUT le fichier COURANT

- COCHER les DEUX cases " Expression régulière " et " Boucler "

- CLIQUER sur l'UN des BOUTONS " Remplacer " ou " Remplacer tout ", AUTANT de FOIS que nécessaire,

pour effectuer TOUS les REMPLACEMENTS

EXEMPLE :

Soit le FICHIER suivant, contenant une LISTE d'animaux de COMPAGNIE :

|hamster|lapin|chat|chien|tortue|perroquet|

chat|chien|perroquet|chien|tortue|lapin|perroquet|hamster|hamster|hamster

perroquet|lapin|hamster|chat|chien|perroquet|chien|tortue|lapin|chat

chat|chien|perroquet|chien|tortue|chat|lapin|chat|hamster|perroquet|hamster

chien|tortue|chat|chat|chien|perroquet|chat|lapin|tortue|chat|perroquet|hamster

tortue|chat|chien|perroquet|chat|chien|tortue|chat|lapin|chien|chat|hamster|perroquet|chat|tortue

APRÈS une RECHERCHE-REMPLACEMENT, avec (\b(\w+)\b.*?)\|\2(\||\R) en partie RECHERCHE et \1\3

en partie REMPLACEMENT, le FICHIER est TRANSFORMÉ en :

|hamster|lapin|chat|chien|tortue|perroquet|

chat|chien|perroquet|tortue|lapin|hamster

perroquet|lapin|hamster|chat|chien|tortue

chat|chien|perroquet|tortue|lapin|hamster

chien|tortue|chat|perroquet|lapin|hamster

tortue|chat|chien|perroquet|lapin|hamster

53) SUPPRESSION des LIGNES IDENTIQUES, EXCÉDENTAIRES, d'un FICHIER :

| | Zone de RECHERCHE | Zone de REMPLACEMENT | Bouton 'Remplacer' | Recherche SEULE SANS Remplacement |
|---|---|---|---|---|
| (1) | (.+\R)\1+ | \1 | Oui | (.+\R)\1+ |
| (2) | ((^.+\R)(?:.*\R)*?)\2 | \1 | NON | |

NOTES :

Employer le CAS 1, quand les LIGNES, du fichier COURANT, ont été PRÉALABLEMENT TRIÉES. C'est le CAS le PLUS RAPIDE ( Environ 60s pour un FICHIER de 634 lignes, DUPLIQUÉ 39 fois de SUITE, soit 24726 lignes en TOUT )

REMARQUES :

Le TRI sera fait par la FONCTION de TRI du PLUGIN " TextFX " ou par un autre UTILITAIRE

Le CAS 1 NE nécessite PAS de remplacement RÉCURSIF et la RECHERCHE-REMPLACEMENT peut donc être

réalisée, INDIFFÉREMMENT, par l'UNE des DEUX boîtes de dialogue ( CTRL + H ) ou ( CTRL + R )

Le remplacement au COUP par COUP n'est POSSIBLE qu'avec la boîte de dialogue ( CTRL + H )

Employer le CAS 2, lorsque les LIGNES, du fichier COURANT, ne sont PAS TRIÉES. Ce CAS est TOUTEFOIS NETTEMENT MOINS RAPIDE que le CAS 1 ( Environ 25 fois PLUS LENT ! )

REMARQUES :

Le CAS 2 NÉCESSITE le remplacement RÉCURSIF et donc la RECHERCHE-REMPLACEMENT NE peut être réalisée QUE par la boîte de dialogue ( CTRL + R ) du PLUGIN " TextFX "

Le CAS 2 n'est PAS COMPATIBLE avec le REMPLACEMENT au COUP par COUP et nécessite OBLIGATOIREMENT le remplacement GLOBAL

Si la TAILLE du FICHIER et/ou le NOMBRE de DOUBLONS sont IMPORTANTS, le RÉSULTAT du processus de RECHERCHE-REMPLACEMENT est généralement trop LENT et finalement INCOHÉRENT !

54) REMPLACEMENT SIMULTANÉ de PLUSIEURS MOTS ou CARACTÈRES par d'AUTRES MOTS ou CARACTÈRES, dans CHAQUE ligne :

| Zone de RECHERCHE | Zone de REMPLACEMENT | Recherche SEULE SANS Remplacement |
|---|---|---|

| (Mot1)\|(Mot2)\|(Mot3)\|...\|(Motn) | (?1MotA)(?2MotB)(?3MotC)...(?nMotX) | Mot1\|Mot2\|Mot3\|...\|Motn |
|---|---|---|

NOTES :

APRÈS REMPLACEMENT :

- MotA est SUBSTITUÉ à Mot1
- MotB est SUBSTITUÉ à Mot2
- MotC est SUBSTITUÉ à Mot3

  .....

- MotX est SUBSTITUÉ à Motn

Les MOTS Mot1, Mot2, .... et/ou MotA, MotB, .... peuvent aussi REPRÉSENTER un caractère UNIQUE

Cependant, placer le CARACTÈRE '\' DEVANT celui-ci, s'il est ÉGALEMENT :

- un MÉTA-caractère de RECHERCHE, dans la ZONE de RECHERCHE
- un MÉTA-caractère de REMPLACEMENT, dans la ZONE de REMPLACEMENT

VI) Exemple FINAL avec EXPLICATIONS DÉTAILLÉES :

Cette RECHERCHE-REMPLACEMENT, ci-après, a été ÉCRITE sur un des FORUMS de NOTEPAD++, le 28 Octobre 2011

Consulter http://sourceforge.net/projects/notepad-plus/forums/forum/331754/topic/4781855?message=10760819

En voici la TRADUCTION française :

casolol

2011-10-28 02:54:59 PDT

Salut tout le monde,

J'utilise la nouvelle version 5.9.5 de Notepad++.

TROIS problèmes se posent à moi lors d'un FILTRAGE d'un fichier TEXTE.

Heureusement, quelqu'un aura la solution ;)

Je désire effectuer les TROIS opérations ci-dessous, dans l'ORDRE indiqué :

- SUPPRIMER PLUSIEURS lignes COMMENÇANT et FINISSANT par un MOT-CLÉ ( 'test1' - 'test2', dans l'exemple )
  qui sont RÉPÉTÉES, PLUSIEURS fois dans le fichier.

- SUPPRIMER les DEUX PREMIERS et les QUATRE DERNIERS caractères de CHAQUE ligne RESTANTE.

- LISTER les caractères de CHAQUE bloc RESTANT en une SEULE ligne.

Voici mon EXEMPLE :

```
test1
drtztztr
ssd  345
33
test2
A 1         hh54
B 2           hhm3
C 23      ahdf
D 456    ..g4
E 656    #h65

test1
dfd
df32
test2
A 4          ooh4
B 7        hhi-
C 86   ahfg
D 466  fg.g
E 446  jgä6
```

ÉTAPE 1 :

SUPPRIMER TOUT caractère ENTRE les MOTS 'test1' et 'test2' ( et les DEUX mots 'test1' et 'test2' ) :

```
A 1         hh54
B 2      hhm3
C 23     ahdf
D 456    ..g4
E 656   #h65


A 4          ooh4
B 7      hhi-
```

```
C 86     ahfg
D 466    fg.g
E 446      jgä6
```

ÉTAPE 2 :

SUPPRIMER TOUS les caractères ADDITIONNELS de CHAQUE ligne ( hh54, hhm3,... ) :

```
A 1
B 2
C 23
D 456
E 656

A 4
B 7
C 86
D 466
E 446
```

ÉTAPE 3 :

SUPPRIMER le 1ER caractère et l'ESPACE qui la suit, dans CHAQUE ligne :

```
1
2
23
456
656

4
7
```

86
466
446

ÉTAPE 4 :

Placer TOUS les caractères de CHAQUE bloc en une SEULE ligne :

1 2 23 456 656
4 7 86 466 446

Quelqu'un a-t-il une idée ? Quelqu'un peut-il y regarder et m'aider ?

Merci d'avance,

Amicalement,

Marc

Il existe une SOLUTION, qui, en une SEULE opération de RECHERCHE-REMPLACEMENT, donne le RÉSULTAT souhaité !!!

LANCER notre éditeur chéri NOTEPAD++

OUVRIR une NOUVEAU fichier avec le RACCOURCI ' CTRL + N '

OUVRIR la fenêtre de RECHERCHE-REMPLACEMENT avec le RACCOURCI ' CTRL + H '

Dans la zone de RECHERCHE, ÉCRIRE ou RECOPIER (?s)(\R*test1.*?test2\R+)|(?-s)..(\d+).*\R(\R)*

Dans la zone de REMPLACEMENT, ÉCRIRE ou RECOPIER (?1:\2\3(?3: ))

COCHER la CASE " Expression régulière "

DÉCOCHER les CASES " Respecter la casse " et " Boucler "

Dans Adobe Reader, se RENDRE à la FIN de ce DESCRIPTIF, par le RACCOURCI ' CTRL + Fin '

RECOPIER TOUT le TEXTE, située APRÈS la ligne de TIRETS, jusqu'en FIN de FICHIER

Dans NOTEPAD++, COLLER ce TEXTE dans le fichier VIERGE, par le raccourci ' CTRL + V '

CLIQUER, enfin, sur le BOUTON " Remplacer tout "

..... WAOOOOOOU ! Ça décoiffe !

**********
* Vous SAISISSEZ, à présent, TOUTE la PUISSANCE des expressions RÉGULIÈRES PRCE !!! *
**********

EXPLICATIONS de la SYNTAXE utilisée dans les DEUX zones de RECHERCHE et de REMPLACEMENT :

La zone de RECHERCHE procède à DEUX recherches INDÉPENDANTES, SÉPARÉES par le signe d'ALTERNATION '|'

- La PREMIÈRE (?s)(\R*test1.*?test2\R+) recherche le BLOC, MULTI-lignes MINIMUM 'Test1... Test2',
ÉVENTUELLEMENT PRÉCÉDÉ de lignes VIDES est SUIVI de lignes VIDES, représentant le GROUPE 1

- La DEUXIÈME (?-s)..(\d+).*\R(\R)* recherche un NOMBRE( GROUPE 2 ), PRÉCÉDÉ de DEUX caractères,
SUIVI de caractères QUELCONQUES, eux-mêmes SUIVIS d'au MOINS UNE 'FIN de LIGNE'

- Si une SUITE EXCÉDENTAIRE de 'FIN de LIGNE' EXISTE => \R représente le GROUPE 3

- Si AUCUNE suite EXCÉDENTAIRE de 'FIN de LIGNE', APRÈS \R => Le GROUPE 3 est alors VIDE

Dans la 1ÉRE recherche, l'OPTION (?s) indique que le MÉTA-caractère POINT (.) représente TOUT caractère, y COMPRIS les caractères 'FIN de LIGNE' => la RECHERCHE s'étend ainsi sur PLUSIEURS lignes

Dans la 2ÈME recherche, l'OPTION (?-s) indique le MÉTA-caractère POINT (.) représente, à NOUVEAU, un caractère STANDARD, AUTRE que les TROIS caractères \n (LF) , \r (CR) et \f (FF)

La zone de REMPLACEMENT correspond, aussi, à DEUX remplacements INDÉPENDANTS, SÉPARÉS par le SIGNE (:)

- Si le GROUPE 1 EXISTE (?1....:....), suite 1ÈRE recherche TROUVÉE, on REMPLACE ce GROUPE 1 par la partie située APRÈS le n° de GROUPE 1 et AVANT le SIGNE (:), c'est à dire la chaîne VIDE

  => le BLOC 'Test1'.....'Test2', PRÉCÉDÉ et/ou SUIVI de 'FIN de LIGNE' est alors SUPPRIMÉ

- Si le GROUPE 1 N'existe PAS, suite 2ÈME recherche TROUVÉE, on REMPLACE la ligne COURANTE par la partie située APRÈS le SIGNE (:) jusqu'à la PARENTHÈSE FERMANTE du groupe CONDITIONNEL, soit '\2\3(?3: )', c'est à dire par :

  - le GROUPE 2, représentant le NOMBRE, puis

- le GROUPE 3, qui est, en général, VIDE, EXCEPTÉ quand il existe PLUSIEURS 'FIN de LIGNE'
  ( FIN de BLOC ), auquel cas \3 représente \R, soit le PASSAGE à la ligne SUIVANTE
- Une ESPACE, quand il n'y a PAS de 'FIN de LIGNE' EXCÉDENTAIRE ( PAS de GROUPE 3 )

NOTE :

La partie REMPLACEMENT pourrait aussi s'écrire, SANS les PARENTHÈSES : '?1:\2\3?3: '

## VII) CONCLUSION :

Voilà pratiquement QUATRE mois et DEMI que j'ai commencé cette DESCRIPTION des EXPRESSIONS RÉGULIÈRES PCRE ! Enfin TERMINÉ ! J'espère qu'elle s'avérera UTILE pour RÉSOUDRE vos PROPRES besoins, en matière de RECHERCHES-REMPLACEMENTS.

Il me serait bien DIFFICILE d'expliquer POURQUOI j'adore les EXPRESSIONS RÉGULIÈRES ! Peut-être, est-ce parce que j'ai commencé l'informatique sous UNIX, avec comme SEUL éditeur VI, il y a environ 35 ANS ou aussi, parce que j'ai TOUJOURS aimé la PROGRAMMATION, en général ( quoique uniquement en QBASIC ! ) Eh oui, l'ANNIVERSAIRE de mes 60 ANS, c'était en Mai 2012 ! ( Mais..., rassurez-vous, comme dans BEAUCOUP de PAYS, je dois encore CONTINUER pour une RETRAITE à peu près DÉCENTE :)! )

L'AJOUT des EXPRESSIONS RÉGULIÈRES PCRE à NOTEPAD++ est une vraie AVANCÉE et DÉCUPLE les performances de

l'éditeur, malgré quelques BUGS, évoqués dans ce MANUEL, qui seront TRÈS VITE réparés, je n'en doute pas !

Voici bientôt QUATRE ans que je " suis " NOTEPAD++ ( depuis la version 4.9.2 EXACTEMENT ! ). Mais, ce n'est QUE MAINTENANT, que j'interviens, ENFIN, dans les FORUMS ! Accordez-moi une période de MISE en ROUTE, et je serai alors tout disposé, à vous AIDER à la CRÉATION de vos RECHERCHES-REMPLACEMENTS spécifiques !

Bien entendu, toute SUGGESTION, CRITIQUE, AMÉLIORATION, ainsi que toute ERREUR détectée, concernant ce DESCRIPTIF, seront les BIENVENUES. Toutefois, mon activité de TECHNICIEN informatique ITINÉRANT est assez PRENANTE. Aussi, ne soyez pas ÉTONNÉS si je ne vous semble pas très RÉACTIF !

ANNEXE :

Exemple FINAL de casolol, avec AJOUT de lignes VIDES + un COUPLE test1 - test2 mis en MAJUSCULES :

- SÉLECTIONNER, dans Adobe Reader, le TEXTE, situé JUSTE APRÈS la ligne de TIRETS, ci-dessous
- RECOPIER celui-ci dans le PRESSE-PAPIERS par le RACCOURCI habituel CTRL + C
- Ouvrir NOTEPAD++ et COLLER ce TEXTE, dans un NOUVEAU fichier, par CTRL + V
- EXÉCUTER la RECHERCHE-REMPLACEMENT, décrite au CHAPITRE VI, ci-dessus
- ADMIREZ ......!!!!!

```
-------------------------------------

test1
drtztztr
ssd  345
33
test2



A 1         hh54
B 2        hhm3
C 23      ahdf
D 456    ..g4
E 656    #h65




TEST1
dfd
df32
TEST2



A 4          ooh4
B 7     hhi-
C 86    ahfg
D 466   fg.g
E 446   jgä6
```