***Sưu tầm tại GiaiPapExcel.com***

***RegExp (Regular Expression)***  
-Là một biểu thức thường được dùng để mô tả một mẫu chuỗi tuân theo 1 quy luật sắp xếp nào đó (string pattern).  
1.Cách khai báo :

Set objRegEx = CreateObject("VBScript.RegExp")

2. Các phương thức , tham số và thuộc tính của RegExp  
***Property Global As Boolean***=True thì RegExp duyệt toàn bộ String, nếu False thì RegExp duyệt có trùng thì dừng  
***Property IgnoreCase As Boolean***= True so sánh không phân biệt hoa thường , flase thì ngược lại   
default là False  
***Property Pattern As String***Theo cú pháp riêng , đây là phần quan trọng nhất của regExp  
***Property Multiline As Boolean***=True : Duyệt ở chế độ nhiều dòng   
***Execute(sourceString As String) As Object***- Tìm chuỗi theo mẫu và trả về một mảng các giá trị tìm thấy  
***Replace(sourceString As String, replaceString As String) As String***-Thay thế chuỗi bằng chuỗi khác phù hợp với mẫu tìm kiếm.  
***Test(sourceString As String) As Boolean***  
-Trả về true nếu như chuỗi gốc khớp với giá trị trả về của biến\_regex.Ngược lại là false.  
Ngoài ra còn có thêm các phương thức khác như :  
***Property Count As Long  
Property FirstIndex As Long  
Property Value   
Property Item(index As Long)***

***\*Parttern có thể gồm có các ký tự thông thường****(Literal Characters) :*  
Trong RegExp có 11 ký tự mang ý nghĩa đặc biệt: [ \ ^ $ . | ? \* + ( ). Chúng dc gọi là các metacharacter., Ngoài các ký tự đặc biệt trên ra, các ký khác được coi là ký tự thông thường :  
ví dụ : tách chữ excel trong chuỗi “ Dien dan giai phap excel”

Sub Regx()

Dim str As String, kqua

str = " Dien dan giai phap excel "

With CreateObject("VBScript.RegExp")

.Global = True

.Pattern = "excel"

Set kqua = .Execute(str)

Debug.Print kqua(0) 'Tra ve 1 ket qua duy nhat la excel'

End With

End Sub

***\* Các ký tự đặc biệt dùng trong phép toán tìm vị trí :***  
^: Trả về chuỗi kết quả trong trường hợp chuỗi này nằm ở vị trí đầu của chuỗi gốc  
$: Trả về chuỗi kết quả trong trường hợp chuỗi này nằm ở vị trí cuối của chuỗi gốc  
\b: Trả về chuỗi kết quả trong trường hợp chuỗi này nằm ở vị trí đầu của một từ trong chuỗi gốc. Nếu cần so sánh ở vị trí cuối từ, hãy đặt biểu thức \b ở vị trí cuối từ.  
\B: Trả về chuỗi kết quả trong trường hợp chuỗi này không nằm ở vị trí đầu của một từ trong chuỗi gốc. (tuỳ thuộc vào vị trí đặt \B ở đầu hoặc cuối một từ)  
(?=): Trả về chuỗi kết quả nếu theo sau chuỗi đó là một chuỗi nào đó được chỉ định trước" có nhĩa là "?=: Trả về chuỗi kết quả nếu theo sau chuỗi kết quả đó là một chuỗi nào đó được chỉ định trước"

(?!): Trả về chuỗi kết quả nếu theo sau hoặc trước chuỗi đó không phải là một ký tự được chỉ định trước  
ví dụ :

Sub timtheo\_vitri()

Dim str As String, kqua

str = "Lua nep la lua nep lang,lua len lop lop ,long nang lang lang "

With CreateObject("VBScript.RegExp")

.Global = True

.IgnoreCase = True

.Pattern = "Lua"

Set kqua = .Execute(str) ' Kqua se tra ve 1 mang gom 3 chu : Lua lua lua

.Pattern = "^Lua"

Set kqua = .Execute(str) ' Kqua se tra ve 1 gia tri chu Lua dau tien

End With

' Tuong tu co the ap dung cho cac truong hop khac ^^

End Sub

***\* Ta có thể sử dụng các ký tự thông thường ở trong 1 lớp , gọi là lớp ký tự trong RegExp***  
***Lưu ý :***  
*Trong lớp ký tự, các ký tự mang ý nghĩa đặc biệt ( metacharacter)chỉ bao gồm: ] \ ^ - Các metacharacter nói ở phần trước khi đặt trong lớp ký tự sẽ chỉ dc coi như ký tự thông thường*  
*Đặt dấu ^ sau [ trong lớp ký tự sẽ phủ định lớp ký tự đó. Kết quả là lớp ký tự sẽ so khớp với bất kỳ ký tự nào ko nằm trong lớp ký tự đó. Lớp ký tự phủ định có thể so khớp với cả ký tự line break*  
*Để giải phóng các ký tự đặc biệt và sử dụng chúng như ký tự bình thường ta dùng dấu****\****để giải phóng chúng*  
*ví dụ pattern = “ 1\+1=2 ‘’ sẽ khớp với chuỗi 1+1 =2 , còn nếu viết “ 1+1=2” nó sẽ khớp với cả 111=2*  
***Cụ thể các lớp như sau :***  
  
[xyz]: Tìm một ký tự bất kỳ nằm trong tập ký tự giữa cặp dấu ngoặc vuông  
[x-z]: Tìm một ký tự bất kỳ nằm trong tập ký tự từ x đến z  
[^xyz]: Tìm một ký tự bất kỳ không thuộc tập ký tự giữa cặp dấu ngoặc vuông.  
**.** : Tìm bất kỳ một ký tự nào không phải là ký tự xuống dòng mới (new line) hoặc ký tự kết thúc dòng (line terminator).  
*Các ký tự viết tắt ( có thể được dùng ở trong hoặc ở ngoài lớp )*  
\w: Tìm một ký tự dạng a-Z, 0-9 và dấu gạch dưới.  
\W: Ngược lại với \w  
\d: Tìm một ký tự thuộc tập ký tự từ 0 đến 9  
\D: Ngược lại với \d: Tìm một ký tự không nằm trong tập ký tự từ 0 đến 9  
\s: Tìm ký tự " dấu trắng " ( vd : dấu cách , tab ...)  
\S:Tìm một ký tự không phải là ký tự " dấu trắng "

Function TachSo(Cell As Range) As Double

Set Temp = CreateObject("VBScript.RegExp")

Temp.Global = True

Temp.Pattern = "[^0-9]"

TachSo = Temp.Replace(Cell, "")

End Function

**\* Các phép lặp trong regex**  
  
*RegEx cho phép tìm kiếm lặp bên trong biểu thức*:  
{x}: Lặp một ký tự hoặc một biểu thức con trước đó x lần  
{x,y}: Lặp một ký tự hoặc một biểu thức con trước đó từ x đến y lần  
{x,}: Lặp một ký tự hoặc một biểu thức con trước đó >= x lần  
?: Lặp một ký tự hoặc một biểu thức con trước đó 0 hoặc 1 lần  
\*: Lặp một ký tự hoặc một biểu thức con trước đó >=0 lần  
+: Lặp một ký tự hoặc một biểu thức con trước đó >=1 lần  
Trong RegExp coi ký tự { là ký tự thông thường nên ta không phải giải phóng ký tự đó,  
**\* Gộp nhóm các biểu thức**  
Chúng ta có thể sử dụng các dấu ngoặc tròn () để gộp nhóm như trong các biểu thức toán học thông thường.  
  
(): Tìm kiếm một nhóm các ký tự bên trong cặp dấu ngoặc và lưu vào chuỗi kết quả.  
(?: ): Tìm kiếm chuỗi kết quả không chứa tập ký tự nằm trong cặp dấu ngoặc.

**Sưu tầm tại StackOverFlow.com**

[Regular expressions](http://en.wikipedia.org/wiki/Regular_expressions) are used for Pattern Matching.

To use in Excel follow these steps :

Step 1: Add VBA reference to "Microsoft VBScript Regular Expressions 5.5"

* Select "Developer" tab ([I don't have this tab what do I do?](http://msdn.microsoft.com/en-us/library/bb608625.aspx))
* Select "Visual Basic" icon from 'Code' ribbon section
* In "Microsoft Visual Basic for Applications" window select "Tools" from the top menu.
* Select "References"
* Check the box next to "Microsoft VBScript Regular Expressions 5.5" to include in your workbook.
* Click "OK"

Step 2: Define your pattern

Basic definitions:

- Range.

* E.g. a-z matches an lower case letters from a to z
* E.g. 0-5 matches any number from 0 to 5

[] Match exactly one of the objects inside these brackets.

* E.g. [a] matches the letter a
* E.g. [abc] matches a single letter which can be a, b or c
* E.g. [a-z] matches any single lower case letter of the alphabet.

() Groups different matches for return purposes. See examples below.

{} Multiplier for repeated copies of pattern defined before it.

* E.g. [a]{2} matches two consecutive lower case letter a: aa
* E.g. [a]{1,3} matches at least one and up to three lower case letter a, aa, aaa

+ Match at least one, or more, of the pattern defined before it.

* E.g. a+ will match consecutive a's a, aa, aaa, and so on

? Match zero or one of the pattern defined before it.

* E.g. Pattern may or may not be present but can only be matched one time.
* E.g. [a-z]? matches empty string or any single lower case letter.

\* Match zero or more of the pattern defined before it. - E.g. Wildcard for pattern that may or may not be present. - E.g. [a-z]\* matches empty string or string of lower case letters.

. Matches any character except newline \n

* E.g. a. Matches a two character string starting with a and ending with anything except \n

| OR operator

* E.g. a|b means either a or b can be matched.
* E.g. red|white|orange matches exactly one of the colors.

^ NOT operator

* E.g. [^0-9] character can not contain a number
* E.g. [^aA] character can not be lower case a or upper case A

\ Escapes special character that follows (overrides above behavior)

* E.g. \., \\, \(, \?, \$, \^

Anchoring Patterns:

^ Match must occur at start of string

* E.g. ^a First character must be lower case letter a
* E.g. ^[0-9] First character must be a number.

$ Match must occur at end of string

* E.g. a$ Last character must be lower case letter a

**Một vài ví dụ hay để hiểu cách thức hoạt động cơ bản của Regular Expression**

**Example 1**: *Run as macro*

The following example macro looks at the value in cell A1 to see if the first 1 or 2 characters are digits. If so, they are removed and the rest of the string is displayed. If not, then a box appears telling you that no match is found. Cell A1 values of 12abc will return abc, value of 1abc will return abc, value of abc123 will return "Not Matched" because the digits were not at the start of the string.

Private Sub simpleRegex()

Dim strPattern As String: strPattern = "^[0-9]{1,2}"

Dim strReplace As String: strReplace = ""

Dim regEx As New RegExp

Dim strInput As String

Dim Myrange As Range

Set Myrange = ActiveSheet.Range("A1")

If strPattern <> "" Then

strInput = Myrange.Value

With regEx

.Global = True

.MultiLine = True

.IgnoreCase = False

.Pattern = strPattern

End With

If regEx.Test(strInput) Then

MsgBox (regEx.Replace(strInput, strReplace))

Else

MsgBox ("Not matched")

End If

End If

End Sub

**Example 2**: Run as an in-cell function

This example is the same as example 1 but is setup to run as an in-cell function. To use, change the code to this:

Function simpleCellRegex(Myrange As Range) As String

Dim regEx As New RegExp

Dim strPattern As String

Dim strInput As String

Dim strReplace As String

Dim strOutput As String

strPattern = "^[0-9]{1,3}"

If strPattern <> "" Then

strInput = Myrange.Value

strReplace = ""

With regEx

.Global = True

.MultiLine = True

.IgnoreCase = False

.Pattern = strPattern

End With

If regEx.test(strInput) Then

simpleCellRegex = regEx.Replace(strInput, strReplace)

Else

simpleCellRegex = "Not matched"

End If

End If

End Function

**Example 3**: Loop Through Range

This example is the same as example 1 but loops through a range of cells.

Private Sub simpleRegex()

Dim strPattern As String: strPattern = "^[0-9]{1,2}"

Dim strReplace As String: strReplace = ""

Dim regEx As New RegExp

Dim strInput As String

Dim Myrange As Range

Set Myrange = ActiveSheet.Range("A1:A5")

For Each cell In Myrange

If strPattern <> "" Then

strInput = cell.Value

With regEx

.Global = True

.MultiLine = True

.IgnoreCase = False

.Pattern = strPattern

End With

If regEx.Test(strInput) Then

MsgBox (regEx.Replace(strInput, strReplace))

Else

MsgBox ("Not matched")

End If

End If

Next

End Sub

**Example 4**: Splitting apart different patterns

This example loops through a range (A1, A2 & A3) and looks for a string starting with three digits followed by a single alpha character and then 4 numeric digits. The output splits apart the pattern matches into adjacent cells by using the (). $1 represents the first pattern matched within the first set of ().

Private Sub splitUpRegexPattern()

Dim regEx As New RegExp

Dim strPattern As String

Dim strInput As String

Dim Myrange As Range

Set Myrange = ActiveSheet.Range("A1:A3")

For Each C In Myrange

strPattern = "(^[0-9]{3})([a-zA-Z])([0-9]{4})"

If strPattern <> "" Then

strInput = C.Value

With regEx

.Global = True

.MultiLine = True

.IgnoreCase = False

.Pattern = strPattern

End With

If regEx.test(strInput) Then

C.Offset(0, 1) = regEx.Replace(strInput, "$1")

C.Offset(0, 2) = regEx.Replace(strInput, "$2")

C.Offset(0, 3) = regEx.Replace(strInput, "$3")

Else

C.Offset(0, 1) = "(Not matched)"

End If

End If

Next

End Sub

**Example 5**

Function RegEx1(strIN As String, strPattern As String, Optional ByVal OutPattern As String = "$0") As Variant

Dim InRegEx As Object, OutRegEx As Object, OutReplaceRegEx As Object

Dim InMatches As Object, replaceMatches As Object, replaceMatch As Object

Dim replaceNumber As Integer

Set InRegEx = CreateObject("VBScript.RegExp")

Set OutRegEx = CreateObject("VBScript.RegExp")

Set OutReplaceRegEx = CreateObject("VBScript.RegExp")

With InRegEx

.Global = True

.MultiLine = True

.Pattern = strPattern

End With

With OutRegEx

.Global = True

.MultiLine = True

.Pattern = "\$(\d+)"

End With

With OutReplaceRegEx

.Global = True

.MultiLine = True

End With

Set InMatches = InRegEx.Execute(strIN)

If InMatches.Count = 0 Then

RegEx = False

Else

Set replaceMatches = OutRegEx.Execute(OutPattern)

For Each replaceMatch In replaceMatches

replaceNumber = replaceMatch.SubMatches(0)

OutReplaceRegEx.Pattern = "\$" & replaceNumber

If replaceNumber = 0 Then

OutPattern = OutReplaceRegEx.Replace(OutPattern, InMatches(0).Value)

Else

If replaceNumber > InMatches(0).SubMatches.Count Then

'regex = "A to high $ tag found. Largest allowed is $" & InMatches(0).SubMatches.Count & "."

RegEx = CVErr(xlErrValue)

Exit Function

Else

OutPattern = OutReplaceRegEx.Replace(OutPattern, InMatches(0).SubMatches(replaceNumber - 1))

End If

End If

Next

RegEx = OutPattern

End If

End Function

**Example 6** : Sử dụng function kết hợp với putInCliboard

'[a1:b30000] = [{"ABC123-009",""}]: Dim t As Double: t = Timer ' used for testing

Dim r As Range, s As String

Set r = ThisWorkbook.Worksheets("Data").UsedRange.Resize(, 1) ' Data!A1:A30000

With New MSForms.DataObject ' needs reference to "Microsoft Forms 2.0 Object Library" or use a bit slower late binding - With CreateObject("New:{1C3B4210-F441-11CE-B9EA-00AA006B1A69}")

r.Copy

.GetFromClipboard

Application.CutCopyMode = False

s = .GetText

.Clear ' optional - clear the clipboard if using Range.PasteSpecial instead of Worksheet.PasteSpecial "Text"

With New RegExp ' needs reference to "Microsoft VBScript Regular Expressions 5.5" or use a bit slower late binding - With CreateObject("VBScript.RegExp")

.Global = True

'.IgnoreCase = False ' .IgnoreCase is False by default

.Pattern = "[^0-9A-Za-z\r\n]+" ' because "[^\w\r\n]+" also matches \_ and Unicode letters

s = .Replace(s, vbNullString)

End With

.SetText s

.PutInClipboard

End With

' about 70% of the time is spent here in pasting the data

r(, 2).PasteSpecial 'xlPasteValues ' paste the text from clipboard in B1

'Debug.Print Timer - t

**Example 7**: Kết hợp RegExp với mảng

Sub Regex()

Dim inputSh As Worksheet

Dim inputRng As Range

Set inputSh = Sheets("Data")

Set inputRng = inputSh.Range("A1:A30000")

Dim outputSh As Worksheet

Dim outputRng As Range

Set outputSh = Sheets("Regex")

Set outputRng = outputSh.Range("A1:A30000")

Dim time1 As Double, time2 As Double

time1 = MicroTimer

Dim arr As Variant

arr = inputRng

Dim objRegex As Object

Set objRegex = CreateObject("vbscript.regexp")

With objRegex

.Global = True

.ignorecase = True

.Pattern = "[^\w]"

End With

Dim i As Integer

For i = LBound(arr) To UBound(arr)

arr(i, 1) = objRegex.Replace(arr(i, 1), vbNullString)

Next i

outputRng = arr

time2 = MicroTimer

Debug.Print (time2 - time1) \* 1000

End Sub

**Example 8 – 10**

'---------------------------------------------------------------------------------------vv

' Procedure : RegEx

' Author : Mike

' Date : 9/1/2010

' Purpose : Perform a regular expression search on a string and return the first match

' or the null string if no matches are found.

' Usage : If Len(RegEx("\d{1,2}[/-]\d{1,2}[/-]\d{2,4}", txt)) = 0 Then MsgBox "No date in " & txt

' : TheDate = RegEx("\d{1,2}[/-]\d{1,2}[/-]\d{2,4}", txt)

' : CUSIP = Regex("[A-Za-z0-9]{8}[0-9]",txt)

'---------------------------------------------------------------------------------------

'^^

Function RegEx(Pattern As String, TextToSearch As String) As String 'vv

Dim RE As Object, REMatches As Object

Set RE = CreateObject("vbscript.regexp")

With RE

.MultiLine = False

.Global = False

.IgnoreCase = False

.Pattern = Pattern

End With

Set REMatches = RE.Execute(TextToSearch)

If REMatches.Count > 0 Then

RegEx = REMatches(0)

Else

RegEx = vbNullString

End If

End Function '^^

'---------------------------------------------------------------------------------------

' Procedure : RegExReplace

' Author : Mike

' Date : 11/4/2010

' Purpose : Attempts to replace text in the TextToSearch with text and back references

' from the ReplacePattern for any matches found using SearchPattern.

' Notes - If no matches are found, TextToSearch is returned unaltered. To get

' specific info from a string, use RegExExtract instead.

' Usage : ?RegExReplace("(.\*)(\d{3})[\)\s.-](\d{3})[\s.-](\d{4})(.\*)", "My phone # is 570.555.1234.", "$1($2)$3-$4$5")

' My phone # is (570)555-1234.

'---------------------------------------------------------------------------------------

'

Function RegExReplace(SearchPattern As String, TextToSearch As String, ReplacePattern As String, \_

Optional GlobalReplace As Boolean = True, \_

Optional IgnoreCase As Boolean = False, \_

Optional MultiLine As Boolean = False) As String

Dim RE As Object

Set RE = CreateObject("vbscript.regexp")

With RE

.MultiLine = MultiLine

.Global = GlobalReplace

.IgnoreCase = IgnoreCase

.Pattern = SearchPattern

End With

RegExReplace = RE.Replace(TextToSearch, ReplacePattern)

End Function

'---------------------------------------------------------------------------------------

' Procedure : RegExExtract

' Author : Mike

' Date : 11/4/2010

' Purpose : Extracts specific information from a string. Returns empty string if not found.

' Usage : ?RegExExtract("(.\*)(\d{3})[\)\s.-](\d{3})[\s.-](\d{4})(.\*)", "My phone # is 570.555.1234.", "$2$3$4")

' 5705551234

' ?RegExExtract("(.\*)(\d{3})[\)\s.-](\d{3})[\s.-](\d{4})(.\*)", "My name is Mike.", "$2$3$4")

'

' ?RegExReplace("(.\*)(\d{3})[\)\s.-](\d{3})[\s.-](\d{4})(.\*)", "My name is Mike.", "$2$3$4")

' My name is Mike.

'---------------------------------------------------------------------------------------

'

Function RegExExtract(SearchPattern As String, TextToSearch As String, PatternToExtract As String, \_

Optional GlobalReplace As Boolean = True, \_

Optional IgnoreCase As Boolean = False, \_

Optional MultiLine As Boolean = False) As String

Dim MatchFound As Boolean

MatchFound = Len(RegEx(SearchPattern, TextToSearch)) > 0

If MatchFound Then

RegExExtract = RegExReplace(SearchPattern, TextToSearch, PatternToExtract, \_

GlobalReplace, IgnoreCase, MultiLine)

Else

RegExExtract = vbNullString

End If

End Function

**Example 10** : Chú ý với trường hơp có nhiều submatches tìm thấy cần nối

Function ExtractSDI(ByVal text As String) As String

Dim result As String

Dim allMatches As Object

Dim RE As Object

Set RE = CreateObject("vbscript.regexp")

RE.pattern = "(sdi \d+)"

RE.Global = True

RE.IgnoreCase = True

Set allMatches = RE.Execute(text)

If allMatches.count <> 0 Then

result = allMatches.Item(0).submatches.Item(0)

End If

ExtractSDI = result

End Function

‘=======================================================================

Function RegexExtract(ByVal text As String, \_

ByVal extract\_what As String, \_

Optional seperator As String = "") As String

Dim i As Long, j As Long

Dim result As String

Dim allMatches As Object

Dim RE As Object

Set RE = CreateObject("vbscript.regexp")

RE.pattern = extract\_what

RE.Global = True

Set allMatches = RE.Execute(text)

For i = 0 To allMatches.count - 1

For j = 0 To allMatches.Item(i).submatches.count - 1

result = result & seperator & allMatches.Item(i).submatches.Item(j)

Next

Next

If Len(result) <> 0 Then

result = Right(result, Len(result) - Len(seperator))

End If

RegexExtract = result

End Function

**Một vài vd về pattern cần đưa ra cho String bằng RegExp**

Ex1: Lấy mail của string = “ Peter Gordon: some@email.com, 47”

Pattern = "\w+@\w+\.\w+"

Ex2 : Thêm dấu cách vào tên vd: NguyenVan A

Pattern = "([a-z])([A-Z])"

.Replace(strIN, "$1 $2")

Ex3: Cho 1 chuỗi, “123A45678”, cần tách ra 3 chuỗi : 123 ; A; 45678

Pattern = (^[0-9]{3})([a-zA-Z])([0-9]{4})"

Ex4: Thử tách dữ liệu ngày tháng dạng: dd/mm/yyyy hoặc dd-mm-yyyy  
 pattern: = "\d{1,2}[/-]\d{1,2}[/-]\d{2,4}"

Hoặc (chưa test): (?:((?:19|20)[0-9]{2})[, /.]([a-zA-Z]+)[, /.](0[1-9]|[12][0-9]|3[01])[, /.]) ^^ ^^

Then replace with $1-$2-$3 as you are currently doing.