PONTIFICIA UNIVERSIDAD CATÓLICA MADRE Y MAESTRA
CAMPUS SANTO TOMÁS DE AQUINO
FACULTAD DE CIENCIAS DE LA INGENIERIA
DEPARTAMENTO DE INGENIERIA EN SISTEMAS Y COMPUTACIÓN

Programación III ISC-314
Grupo 001
Prof. Rodrigo Orizondo.

ISC-314 Proyecto

Presentado por:
Alan Álvarez
2010-5914
Isis Gómez
2011-5696.

Carrera:
Ingeniería en Sistemas y Computación – ISC.

25 de abril del 2014.
Santo Domingo, República Dominicana.

Notaciones

Las expresiones regulares en las que se encuentran definidas las producciones del lenguaje VBA, han sido descritas bajo la forma “Augmented Backus–Naur Form” (ABNF). Lo que significa que antes de iniciar las consultas y análisis relacionadas con estos documentos, es requerido que se realice un la lectura exploratoria para poder comprender el orden y la simbología definida en el ABNF.

Las siguientes producciones en ABNF muestran la correspondencia entre VBA y Powershell.

En VBA:

```
WS = 1*(WSC / line-continuation)
special-token = "," / "." / "!" / "#" / "&" / "(" / ")" / "*" / "+" / "-" / "/" / ":" / ";" / "<" / "=" / ">" / "?" / "\" / "^"
NO-WS = <no whitespace characters allowed here>
NO-LINE-CONTINUATION = <a line-continuation is not allowed here>
EOL = [WS] LINE-END / single-quote comment-body
EOS = *(EOL / ":") ;End Of Statement

single-quote = %x0027 ; '
comment-body = *(line-continuation / non-line-termination-character) LINE-END
INTEGER = integer-literal ["%" / "&" / "^"]
integer-literal = decimal-literal / octal-literal / hex-literal
decimal-literal = 1*decimal-digit
octal-literal = "&" [%x004F / %x006F] 1*octal-digit ; & or &o or &O
hex-literal = "&" (%x0048 / %x0068) 1*hex-digit ; &h or &H
octal-digit = "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7"
decimal-digit = octal-digit / "8" / "9"
hex-digit = decimal-digit / %x0041-0046 / %x0061-0066 ;A-F / a-f

FLOAT = (floating-point-literal [floating-point-type-suffix] ) / (decimal-literal floating-point-type-suffix)
floating-point-literal = (integer-digits exponent) / (integer-digits "." [fractional-digits] [exponent]) / ( "." fractional-digits [exponent])
integer-digits = decimal-literal
fractional-digits = decimal-literal
exponent = exponent-letter [sign] decimal-literal
exponent-letter = %x0044 / %x0045 / %x0064 / %x0065 ; D / E / d / e
sign = "+" / "-"
floating-point-type-suffix = "!" / "#" / "@"

DATE = "#" *WSC [date-or-time *WSC] "#"
date-or-time = (date-value 1*WSC time-value) / date-value / time-value
date-value = left-date-value date-separator middle-date-value [date-separator right-date-value]
left-date-value = decimal-literal / month-name
middle-date-value = decimal-literal / month-name
right-date-value = decimal-literal / month-name
date-separator = 1*WSC / (*WSC ("/" / "-" / ",") *WSC)
month-name = English-month-name / English-month-abbreviation
```

English-month-name = "january" / "february" / "march" / "april" / "may" / "june" / "august" / "september" / "october" / "november" / "december"
English-month-abbreviation = "jan" / "feb" / "mar" / "apr" / "jun" / "jul" / "aug" / "sep" / "oct" / "nov" / "dec"
time-value = (hour-value ampm) / (hour-value time-separator minute-value [time-separator second-value] [ampm])
hour-value = decimal-literal
minute-value = decimal-literal
second-value = decimal-literal
*time-separator = *WSC (":" / ".") *WSC*
*ampm = *WSC ("am" / "pm" / "a" / "p")*

*STRING = double-quote *string-character (double-quote / line-continuation / LINE-END)*
double-quote = %x0022 ; "
string-character = NO-LINE-CONTINUATION ((double-quote double-quote) / non-line-termination-character)

Statement-keyword = "Call" / "Case" /"Close" / "Const"/ "Declare" / "DefBool" / "DefByte" / "DefCur" / "DefDate" / "DefDbl" / "DefInt" / "DefLng" / "DefLngLng" / "DefLngPtr" / "DefObj" / "DefSng" / "DefStr" / "DefVar" / "Dim" / "Do" / "Else" / "ElseIf" / "End" / "EndIf" / "Enum" / "Erase" / "Event" / "Exit" / "For" / "Friend" / "Function" / "Get" / "Global" / "GoSub" / "GoTo" / "If" / "Implements"/ "Input" / "Let" / "Lock" / "Loop" / "LSet" / "Next" / "On" / "Open" / "Option" / "Print" / "Private" / "Public" / "Put" / "RaiseEvent" / "ReDim" / "Resume" / "Return" / "RSet" / "Seek" / "Select" / "Set" / "Static" / "Stop" / "Sub" / "Type" / "Unlock" / "Wend" / "While" / "With" / "Write"
rem-keyword = "Rem"
marker-keyword = "Any" / "As"/ "ByRef" / "ByVal "/"Case" / "Each" / "Else" /"In"/ "New" / "Shared" / "Until" / "WithEvents" / "Write" / "Optional" / "ParamArray" / "Preserve" / "Spc" / "Tab" / "Then" / "To"
operator-identifier = "AddressOf" / "And" / "Eqv" / "Imp" / "Is" / "Like" / "New" / "Mod" / "Not" / "Or" / "TypeOf" / "Xor"

control-statement = if-statement / control-statement-except-multiline-if
control-statement-except-multiline-if = call-statement / while-statement / for-statement / exit-for-statement / do-statement / exit-do-statement / single-line-if-statement / select-case-statement /stop-statement / goto-statement / on-goto-statement / gosub-statement / return-statement / on-gosub-statement /for-each-statement / exit-sub-statement / exit-function-statement / exit-property-statement / raiseevent-statement / with-statement

call-statement = "Call" (simple-name-expression / member-access-expression / index-expression / with-expression)
call-statement =/ (simple-name-expression / member-access-expression / with-expression) argument-list

while-statement = "While" boolean-expression EOS statement-block "Wend"

for-statement = simple-for-statement / explicit-for-statement
simple-for-statement = for-clause EOS statement-block "Next"
explicit-for-statement = for-clause EOS statement-block ("Next" / (nested-for-statement ",")) bound-variable-expression
nested-for-statement = explicit-for-statement / explicit-for-each-statement
for-clause = "For" bound-variable-expression "=" start-value "To" end-value [step-clause]
start-value = expression

end-value = expression
step-clause = "Step" step-increment
step-increment = expression

for-each-statement = simple-for-each-statement / explicit-for-each-statement
simple-for-each-statement = for-each-clause EOS statement-block "Next"
explicit-for-each-statement = for-each-clause EOS statement-block ("Next" / (nested-for-statement ",")) bound-variable-expression
for-each-clause = "For" "Each" bound-variable-expression "In" collection
collection = expression

exit-for-statement = "Exit" "For"

do-statement = "Do" [condition-clause] EOS statement-block "Loop" [condition-clause]
condition-clause = while-clause / until-clause
while-clause = "While" boolean-expression
until-clause = "Until" boolean-expression

exit-do-statement = "Exit" "Do"

*if-statement = LINE-START "If" boolean-expression "Then" EOL statement-block *[else-if-block] [else-block] LINE-START (("End" "If") / "EndIf")*
else-if-block = LINE-START "ElseIf" boolean-expression "Then" EOL LINE-START statement-block
else-if-block =/ "ElseIf" boolean-expression "Then" statement-block
else-block = LINE-START "Else" statement-block

single-line-if-statement = if-with-non-empty-then / if-with-empty-then
if-with-non-empty-then = "If" boolean-expression "Then" list-or-label [single-line-else-clause]
if-with-empty-then = "If" boolean-expression "Then" single-line-else-clause
single-line-else-clause = "Else" [list-or-label]
*list-or-label = (statement-label *[":" [same-line-statement]]) / ([":"] same-line-statement *[":" [same-line-statement]])*
same-line-statement = file-statement / error-handling-statement / data-manipulation-statement / control-statement-except-multiline-if

*select-case-statement = "Select" "Case" WS select-expression EOS *[case-clause] [case-else-clause] "End" "Select"*
case-clause = "Case" range-clause ["," range-clause] EOS statement-block
case-else-clause = "Case" "Else" EOS statement-block
range-clause = expression
range-clause =/ start-value "To" end-value
range-clause =/ ["Is"] comparison-operator expression
start-value = expression
end-value = expression
select-expression = expression
comparison-operator = "=" / ("<" ">") / (">" "<") / "<" / ">" / (">" "=") / ("=" ">") / ("<" "=") / ("=" "<")

En Powershell:

<valueRule> =
'(' <assignmentStatementRule> ')' |
'$(' <statementListRule> ')' |
'@(' <statementListRule> ')' |
<cmdletBodyRule> |
'@{' <hashLiteralRule> '}' |
<unaryOperatorToken> <propertyOrArrayReferenceRule> |
<AttributeSpecificationToken> <propertyOrArrayReferenceRule> |
<AttributeSpecificationToken> |
<PrePostfixOperatorToken> <lvalue> |
<NumberToken> |
<LiteralStringToken> |
<ExpandableStringToken> |
<variableToken>

<ExpandableStringToken> = "."*
<StringToken> = '.'*
<VariableToken> = \$[:alnum:]+ | \${.+}
<VariableToken> = \$[:alnum:]+ | \${.+}
<ParameterToken> = -[:letter:]+[:]{0 |1}
<ReferenceOperatorToken> = "." | "::" | "["

<statementBlockRule> =
'{' <statementListRule> '}'
<statementListRule> =
*<statementRule> [<statementSeparatorToken> <statementRule>]**

<lvalueExpression> =
*<lvalue> [? |? <lvalue>]**
<lvalue> =
*<simpleLvalue> <propertyOrArrayReferenceOperator>**
<simpleLvalue> =
<AttributeSpecificationToken> <variableToken>*

<statementRule> =
<ifStatementRule> |
<switchStatementRule> |
<foreachStatementRule> |
<forWhileStatementRule> |
<doWhileStatementRule> |
<functionDeclarationRule> |
<parameterDeclarationRule> |
<flowControlStatementRule> |
<trapStatementRule> |
<finallyStatementRule> |
<pipelineRule>

<ifStatementRule> =
'if' '(' <pipelineRule> ')' <statementBlockRule> [
*'elseif' '(' <pipelineRule> ')' <statementBlockRule>]**
['else' <statementBlockRule>]{0|1}

<switchStatementRule> =
'switch' ['-regex' | '-wildcard' | '-exact']{0 |1}
['-casesensitive']{0|1}
['-file' <propertyOrArrayReferenceRule> |
'(' <pipelineRule> ')']
'{' [
['default' | <ParameterArgumentToken> |
<propertyOrArrayReferenceRule> | <statementBlockRule>]
<statementBlockRule>]+ '}'

<doWhileStatementRule> =
<LoopLabelToken>{0 |1} 'do' <statementBlockRule> ['while' | 'until']
'('<pipelineRule> ')'

<forWhileStatementRule> =
<LoopLabelToken>{0 |1} 'while' '(' <pipelineRule> ')'
<statementBlockRule> |
<LoopLabelToken>{0 |1} 'for' '(' <pipelineRule>{0 |1} ';'
<pipelineRule>{0 |1} ';' <pipelineRule>{0 |1} ')'
<statementBlockRule>

<functionDeclarationRule> =
<FunctionDeclarationToken> <ParameterArgumentToken>
['(' <parameterDeclarationExpressionRule> ')']
<cmdletBodyRule>

<flowControlStatementRule> =
['break' | 'continue']
[<propertyNameToken> | <propertyOrArrayReferenceRule>]{0 |1} |
'return' <pipelineRule>

<parameterDeclarationRule> =
<ParameterDeclarationToken> '('
<parameterDeclarationExpressionRule> ')'

<parameterDeclarationExpressionRule> =
<parameterWithIntializer>
*[<CommaToken> <parameterWithIntializer>]**

<parameterWithIntializer> =
<simpleLvalue> ['=' <expressionRule>]

<ComparisonOperatorToken> =
"-eq" | "-ne" | "-ge" | "-gt" | "-lt" | "-le" |
"-ieq" | "-ine" | "-ige" | "-igt" | "-ilt" | "-ile" |
"-ceq" | "-cne" | "-cge" | "-cgt" | "-clt" | "-cle" |
"-like" | "-notlike" | "-match" | "-notmatch" |
"-ilike" | "-inotlike" | "-imatch" | "-inotmatch" |
"-clike" | "-cnotlike" | "-cmatch" | "-cnotmatch" |
"-contains" | "-notcontains" |
"-icontains" | "-inotcontains" |
"-ccontains" | "-cnotcontains" |
"-isnot" | "-is" | "-as" |
"-replace" | "-ireplace" | "-creplace"

<AssignmentOperatorToken> = "=" | "+=" | "-=" | "=" | "/=" | "%="*
<LogicalOperatorToken> = "-and" | "-or"
<BitwiseOperatorToken> = "-band" | "-bor"
<RedirectionOperatorToken> =
"2>&1" | ">>" | ">" | "<<" | "<" | ">|" | "2>" | "2>>" | "1>>"

<FunctionDeclarationToken> = "function" | "filter"
<ParameterArgumentToken> = [^-($0-9].[^ \t]*
<UnaryOperatorToken> = "!" | "-not" | "+" | "-" | "-bnot" |
<attributeSpecificationToken>
<FormatOperatorToken> = '-f'
<LoopLabelToken> = [:letter:][:alnum:]:*
<ParameterToken> = "param"
<PrePostfixOperatorToken> = '++' | <MinusMinusToken>
<MultiplyOperatorToken> = '' | '/' | '%'*
<AdditionOperatorToken> = '+' | '-' | emDash | enDash | horizontalBar

<cmdletBodyRule> =
'{' ['(' <parameterDeclarationExpressionRule> ')'] (
['begin' <statementBlock> |
'process' <statementBlock> |
'end' <statementBlock>] |*
<statementList> '}'

<expressionRule> = <logicalExpressionRule>

<logicalExpressionRule> =
<bitwiseExpressionRule>
*[<LogicalOperatorToken> <bitwiseExpressionRule>]**

<bitwiseExpressionRule> =
<comparisonExpressionRule> [<BitwiseOperatorToken>
*comparisonExpressionRule>]**

<addExpressionRule> =
*<multiplyExpressionRule> [<AdditionOperatorToken> <multiplyExpressionRule>]**

<comparisonExpressionRule> =
<addExpressionRule>
*[<ComparisonOperatorToken> <addExpressionRule>]**

<multiplyExpressionRule> =
<formatExpressionRule>
[<MultiplyOperatorToken> <formatExpressionRule>]

<formatExpressionRule> =
<rangeExpressionRule>
*[<FormatOperatorToken> <rangeExpressionRule>]**

<rangeExpressionRule> =
*<arrayLiteralRule> [<RangeOperatorToken> <arrayLiteralRule>]**

<arrayLiteralRule> =
*<postfixOperatorRule> [<CommaToken> <postfixOperatorRule>]**

<postfixOperatorRule> =
<lvalueExpression> <PrePostfixOperatorToken> |
<propertyOrArrayReferenceRule>

<propertyOrArrayReferenceRule> =
*<valueRule> <propertyOrArrayReferenceOperator>**

<propertyOrArrayReferenceOperator> =
'[' <expressionRule> ']'] |
'.' [<PropertyNameToken> <parseCallRule>{0|1} | valueRule>]

<parseCallRule> = '(' <arrayLiteralRule> ')'

<pipelineRule> =
*<assignmentStatement> | <firstPipelineElement> ['|' <cmdletCall>]**

<finallyStatementRule> =
'finally' <statementBlockRule>

Palabras Reservadas

Powershell	
Palabras Reservadas	**Comparadores**
Break	-eq -ne -gt -ge -lt -le -like -notlike -match -notmatch -contains -notcontains -replace
Continue	
Do	
Else	
Elseif	
Filter	
For	
Foreach	
Function	
If	
In	
Local	
Private	
Return	
Switch	
Until	
Where	
While	

VBA			
Byte	As	Call	Case
Catch	Continue	Date	Char
Const	Default	Delegate	Decimal
Else	Do	Double	Dim
Enum	ElseIf	End	Each
Exit	False	In	EndIf
Integer	If	Is	For
Not	Nothing	New	Long
Public	Of	On	Module
REM	Optional	Or	Dim
Step	Private	String	Try
Sub	True	Then	When
To	While	Return	

Comparadores	
= <> <	>= <= >

Operadores
+ - * /

Relación VBA – Powershell.

En la siguiente tabla mostramos ejemplos de cómo son las estructuras para el desarrollo de programas tanto para VBA como para Powershell .

	VBA	Powershell
Declaración de Variables	`Dim posX, posY As Double`	`$posX = 0.0` `$posY = 0.45`

Definición e Implementación de Funciones	`Sub Hello()`
	`  MsgBox ("Hello, world!")`
	`End Sub`
		`function Hello ($dir, $minSize)`
		`{`
		`  [System.Windows.Forms.MessageBox]::Show("Hello, Word!")`
		`}`

Llamada de Funciones	`Sub Main()`
	`  Hello`
	`  Call Hello ()`
	`End Sub`
		`Hello()`

Sentencias	`MsgBox "You have a pretty name."`	`"File length: " + $file.Length`

Ciclos	`Sub TwosTotal()`
	`  For i = 1 To 5 Step 1`
	`  MsgBox("i")`
	`  Next j`
	`  MsgBox "The total is " & total`
	`End Sub`
		`for ($i=1; $i -lt 5; $i++)`
		`{`
		`  [System.Windows.Forms.MessageBox]::Show(i)`
		`}`

Condiciones	`Sub AlertUser(value as Long)`
	` If value = 0 Then`
	` AlertLabel.ForeColor = vbRed`
	` AlertLabel.Font.Bold = True`
	` AlertLabel.Font.Italic = True`
	` Else`
	` AlertLabel.Forecolor = vbBlack`
	` AlertLabel.Font.Bold = False`
	` AlertLabel.Font.Italic = False`
	` End If`
	`End Sub`
		`if($temperature -le 0)`
		`{`
		`  "High Freezing"`
		`}`
		`elseif($temperature -le 32)`
		`{`
		`  "Freezing"`
		`}`
		`elseif($temperature -le 50)`
		`{`
		`  "Cold"`
		`}`
		`elseif($temperature -le 70)`
		`{`
		`  "Warm"`
		`}`
		`else`
		`{`
		`  "Hot"`
		`}`

Fuentes Útiles.

Para la correcta traducción de VBA a Powershell, recomendamos los siguientes libros, sitios web y canales de videos. Los mismos serán útiles para la producción de módulos en Powershell. De la misma manera sirvieron como fuentes para el desarrollo del proyecto.

Libros:

- ✓ Windows PowerShell cookbook: the complete guide to scripting Microsoft's command shell, By: Holmes, Lee. O'Reilly, 2013.
  En este libro contiene las especificaciones y recomendaciones para quienes sean iniciar el desarrollo de scripting con powershell.

Sitios Webs:

- ✓ The Scripting Guys - http://blogs.technet.com/b/heyscriptingguy/
  Blog Oficial de Microsoft donde se comparten ejemplos y módulos utiles para el desarrollo procesos automatizados.
- ✓ Visual Basic for Application - http://msdn.microsoft.com/en-us/library/office/gg264383(v=office.15).aspx
  Página Oficial de Microsoft donde se contemplan las especificaciones del lenguaje VBA.
- ✓ Visual Basic for Application in Excel 2010 - http://msdn.microsoft.com/en-us/library/ee814737(v=office.14).aspx
  Página Oficial de Microsoft contiene ejemplos del empleo de VBA en la Microsoft Office Excel 2010.
- ✓ ABNF
  http://en.wikipedia.org/wiki/Augmented_Backus%E2%80%93Naur_Form
- ✓ BNF http://es.wikipedia.org/wiki/Notaci%C3%B3n_de_Backus-Naur

Canales Youtube:

- ✓ MrPowerScripts - https://www.youtube.com/user/MrPowerScripts
  Canal en youtube dedicado a compartir conocimientos del desarrollo de scripting en Powershell. Videos breves y objetivos respecto a temas o tareas en especifico.