

Tecnológico Nacional de México

Instituto Tecnológico de Tijuana

Subdirección Académica

Departamento de Sistemas y Computación

Ingeniería en Sistemas Computacionales

AGOSTO - DICIEMBRE 2016

Lenguajes y Autómatas 1

6SC6A

4:00 p.m. a 6:00 p.m.

Documentación Léxico

Erasmó Estrada Peña

Pasillas Luis Miguel Angel - #14210423

Tijuana, B.C. del 22 de noviembre del 2016

Índice

II – Expresiones Regulares.....	1
III – Autómatas	3
IV – Analizador Léxico.....	16

II – Expresiones Regulares

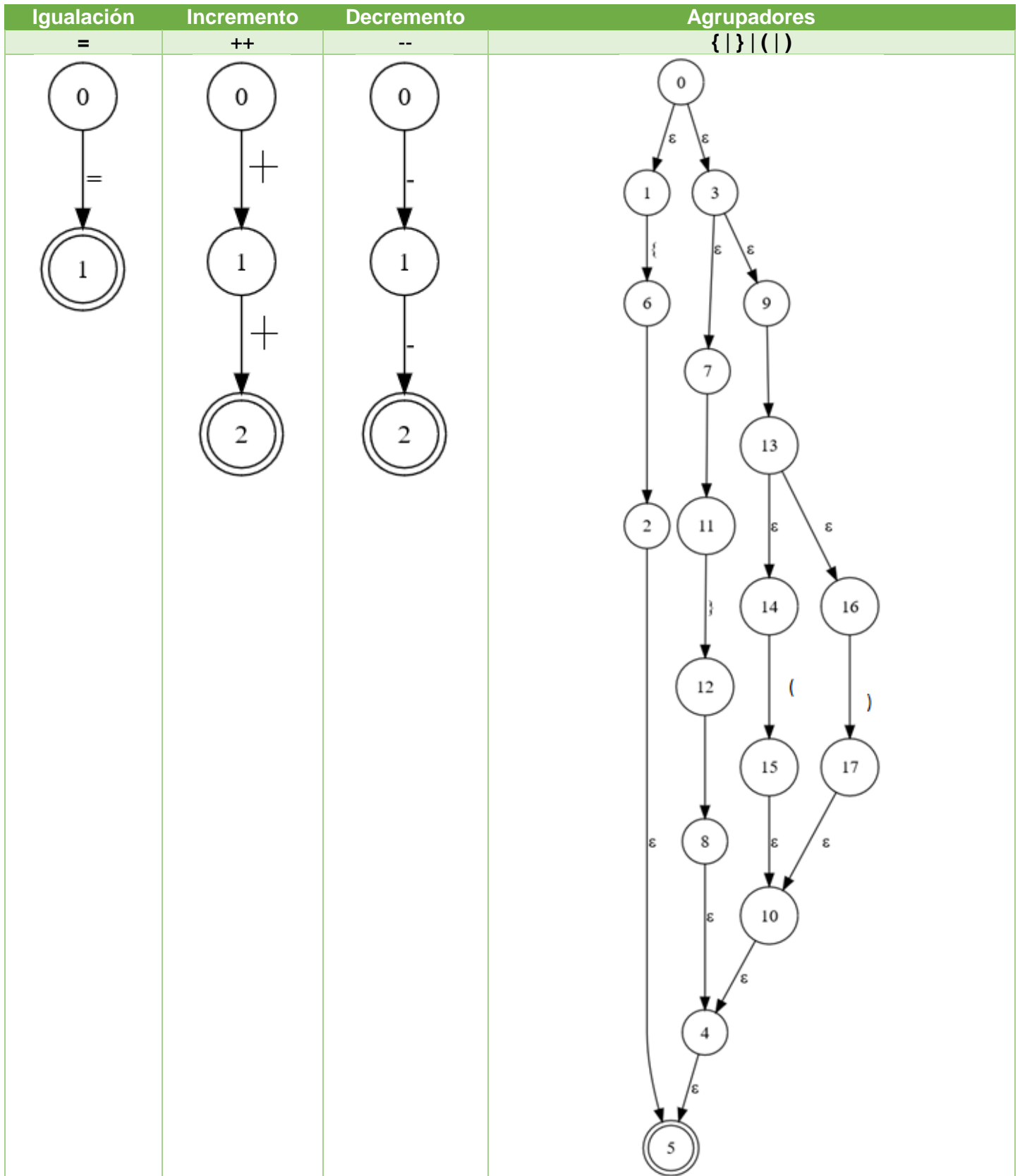
El análisis léxico de un compilador tiene el deber de leer un programa fuente como un archivo de caracteres y dividirlo en *Tokens*. Cada *Token* es una secuencia de caracteres que representan unidad de información en el programa fuente. Como la tarea que realiza el analizador léxico es un caso especial de coincidencias de patrones (*expresiones regulares*), es una regla que genera una secuencia de caracteres que puede representar a un determinado componente léxico. Lexema es una cadena de caracteres que concuerda con un patrón que describe un componen léxico.

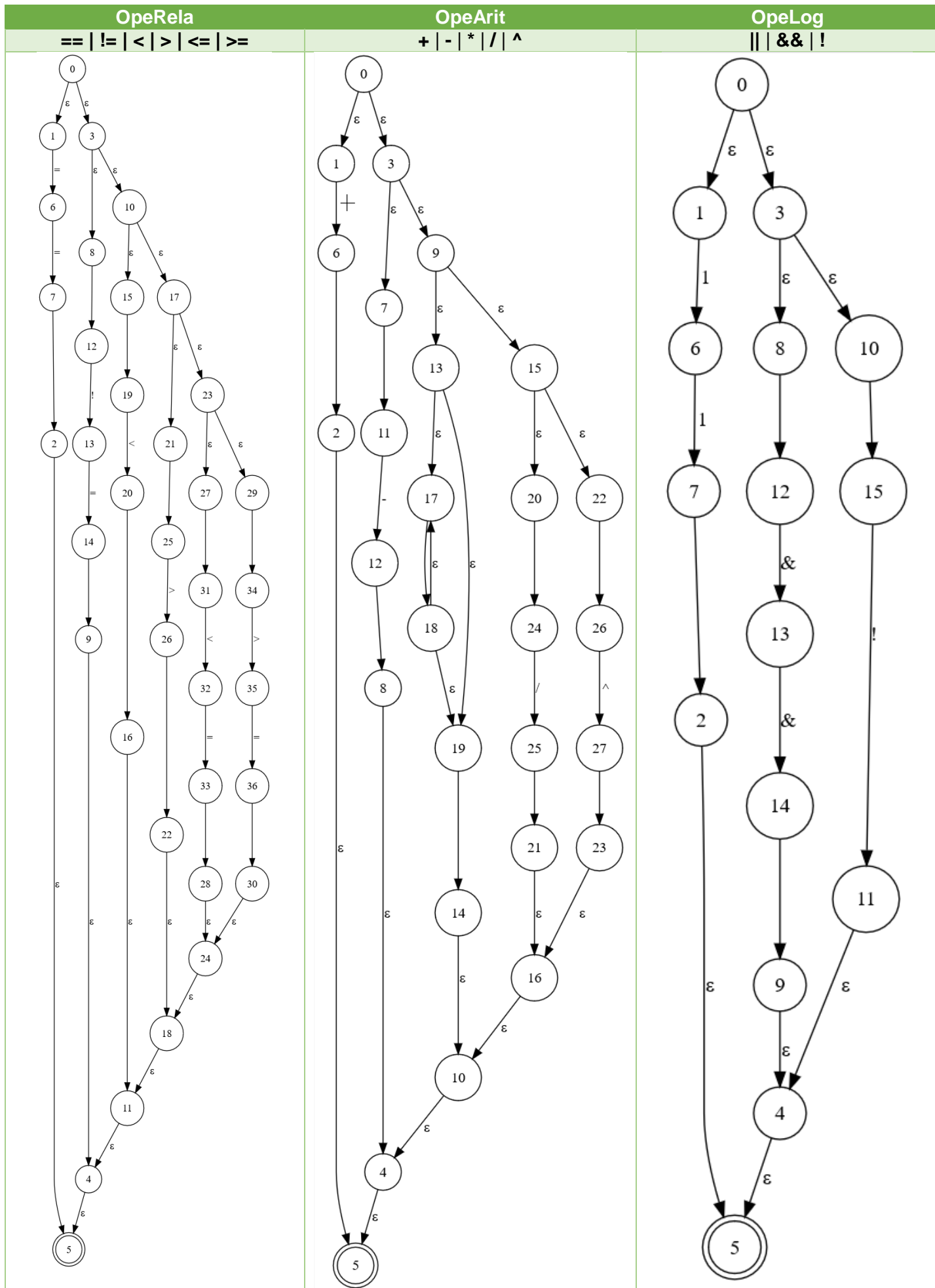
No.	Token	Expresión Regular	Lexema
1	Igualación	<code>^(=)\$</code>	<code>=</code>
2	Incremento	<code>^(++)\$</code>	<code>++</code>
3	Decremento	<code>^(--)\$</code>	<code>--</code>
4	Agrupadores	<code>^(\{ \} ())\$</code>	<code>{ } ()</code>
5	OpeRela	<code>^(== != < > <= >=)\$</code>	<code>== != < > <= >=</code>
6	OpeArit	<code>^(+ - * / ^)\$</code>	<code>+ - * / ^</code>
7	OpeLog	<code>^(&& !)\$</code>	<code> && !</code>
8	TipoBool	<code>^(verdadero falso)\$</code>	<code>verdadero falso</code>
9	TipoDato	<code>^(caracxp cadenaxp entxp bytexp cortoxp boolxp flotanxp flotan64xp uentxp ubytexp ucortoxp ulargoxp decimalxp varxp enumxp structxp)\$</code>	<code>caracxp cadenaxp entxp bytexp cortoxp boolxp flotanxp flotan64xp uentxp ubytexp ucortoxp ulargoxp decimalxp varxp enumxp structxp</code>
10	Instrucción Iteración	<code>^(paraxv paracadaxv mientrasxv hacervx enxv)\$</code>	<code>paraxv paracadaxv mientrasxv hacervx enxv</code>
11	Instrucción Selección	<code>^(sixv sinoxv noxv cambiarxv casoxv)\$</code>	<code>sixv sinoxv noxv cambiarxv casoxv</code>
12	Instrucción Salto	<code>^(descansarxv continuarxv retornarxv defectoxv rendimiendoxv)\$</code>	<code>descansarxv continuarxv retornarxv defectoxv rendimiendoxv</code>
13	Atrapa Errores	<code>^(intentarxv atraparxv finalxv lanzarxv)\$</code>	<code>intentarxv atraparxv finalxv lanzarxv</code>
14	Control Acceso	<code>^(publico privado interno protegido)\$</code>	<code>publico privado interno protegido</code>
15	Referencia	<code>^(clase interface delegado dinamico)\$</code>	<code>clase interface delegado dinamico</code>
16	Modificadores	<code>^(abstractom asincronom constantem eventom externom nuevom anularm parcialm lecturasolom selladom estaticom noguardarm virtualm volatilm)\$</code>	<code>abstractom asincronom constantem eventom externom nuevom anularm parcialm lecturasolom selladom estaticom noguardarm virtualm volatilm</code>
17	Checardor	<code>^(chechar nochechar)\$</code>	<code>chechar nochechar</code>
18	Parámetros	<code>^(paramet referen salida)\$</code>	<code>paramet referen salida</code>
19	Revisión	<code>^(areglado bloquear)\$</code>	<code>areglado bloquear</code>
20	Espacio Nombre	<code>^(espacionombre usando externo)\$</code>	<code>espacionombre usando externo</code>
21	Clave Operadores	<code>^(como esperar es tamañode tipode stackloco explicitoc implicitoc operadorc)\$</code>	<code>como esperar es tamañode tipode stackloco explicitoc implicitoc operadorc</code>
22	Clave Acceso	<code>^(base esta nulo)\$</code>	<code>base esta nulo</code>

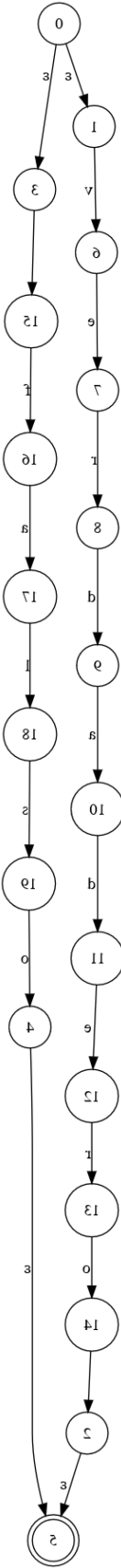
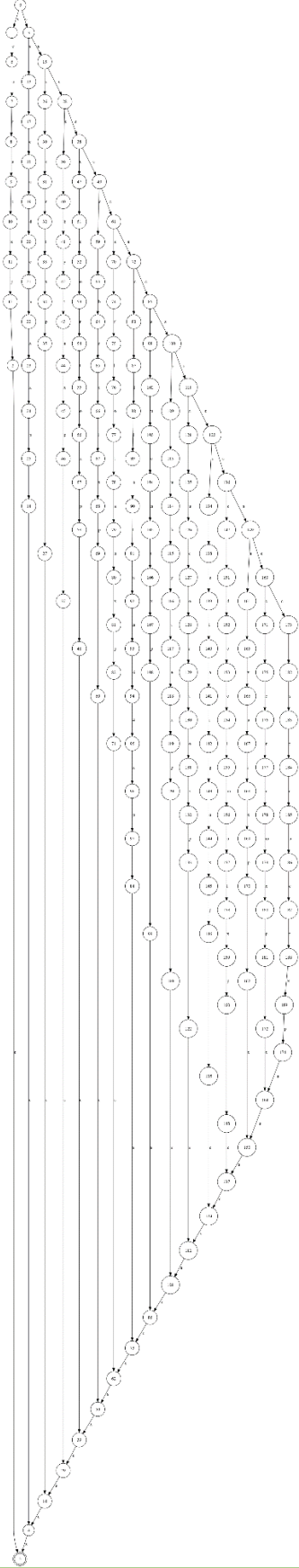
23	Clave Contextual	^(obtener global parcial remover poner evaluar donde anadir)\$	obtener global parcial remover poner evaluar donde anadir
24	Clave Consultas	^(desde donde seleccionar grupo dentro ordenarpor unirse dejar en sobre igual por acendente decendente)\$	desde donde seleccionar grupo dentro ordenarpor unirse dejar en sobre igual por acendente decendente
25	Indicadores de Formato	^(omitir izquierda derecha interno deca octa hexa mostrarbase mostrarpunto letramayus mostrarposi cientifico arreglado enterobuf justarcampo basecampo flotantecampo)\$	omitir izquierda derecha interno deca octa hexa mostrarbase mostrarpunto letramayus mostrarposi cientifico arreglado enterobuf justarcampo basecampo flotantecampo
26	Impresión	^(Efsanf Efsanln Impri Imprif Impriln Escan Escanf Escanln ESprint ESprintf ESprintln Eescan EEscanf)\$	Efsanf Efsanln Impri Imprif Impriln Escan Escanf Escanln ESprint ESprintf ESprintln Eescan EEscanf
27	Funciones Cadenas	^(Contar Igualdoble Campos Funcampo tenerprefijo Tenersufijo Repetir Reemplazar Cortar)\$	Contar Igualdoble Campos Funcampo tenerprefijo Tenersufijo Repetir Reemplazar Cortar
28	Función Principal	^(vacio inicio funcion)\$	vacio inicio funcion
29	Función Matemáticas	^(abstr facos fasin fatan fcos fcosh fexp flog fmax fmin fpow fsign ftan ftanh ftruncate)\$	abstr facos fasin fatan fcos fcosh fexp flog fmax fmin fpow fsign ftan ftanh ftruncate
30	Palabras Reservadas	^(factivate fdouble floctors fadd fdrop flock faster fdssize flockmax falias fdynamic fall feach flong)\$	factivate fdouble floctors fadd fdrop flock faster fdssize flockmax falias fdynamic fall feach flong

III – Autómatas

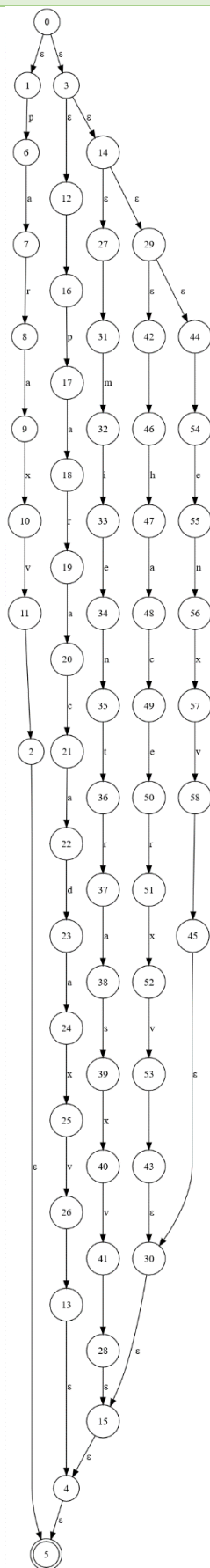
Un Autómata es una construcción lógica que recibe como entrada una cadena de símbolos y produce una salida indicando si la salida es una cadena que pertenece a un determinado lenguaje. Autómata Finito No Determinista: Si se permite que desde un estado se realicen cero, una o más transiciones mediante el mismo símbolo de entrada, se dice que el autómata finito es no determinista.



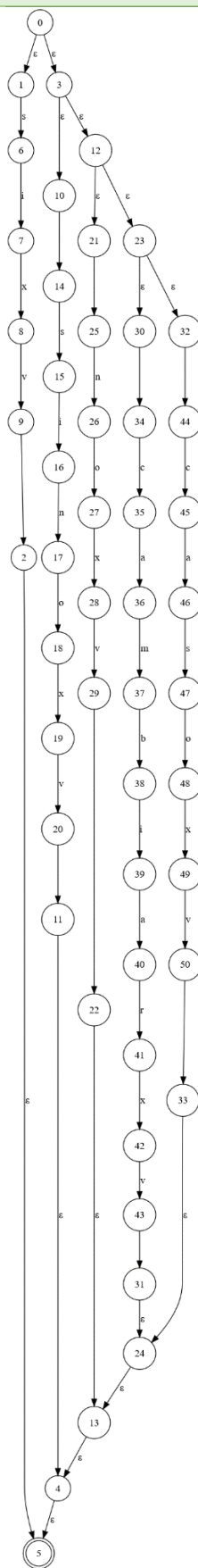


TipoBool verdadero falso	Tipo Dato caracxp cadenaxp entxp bytexp cortoxp boolxp flotanxp flotan64xp uentxp ubytexp ucortoxp ulargoxp decimalxp varxp enumxp structxp
 <pre> graph TD 0((0)) -- 3 --> 1((1)) 0 -- 3 --> 2(((2))) 1 -- v --> 3((3)) 1 -- d --> 4((4)) 3 -- e --> 5((5)) 4 -- e --> 6((6)) 5 -- f --> 7((7)) 6 -- t --> 8((8)) 7 -- b --> 9((9)) 8 -- s --> 10((10)) 9 -- b --> 11((11)) 10 -- 2 --> 12((12)) 11 -- o --> 13((13)) 12 -- e --> 14((14)) 13 -- e --> 15((15)) 14 -- 3 --> 16((16)) 15 -- 3 --> 16 16 -- 3 --> 16 </pre>	

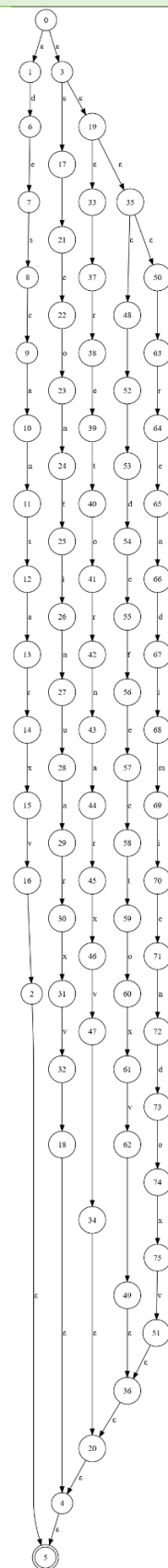
**paraxv | paracadaxv | mientrasxv |
hacerxv | enxv**

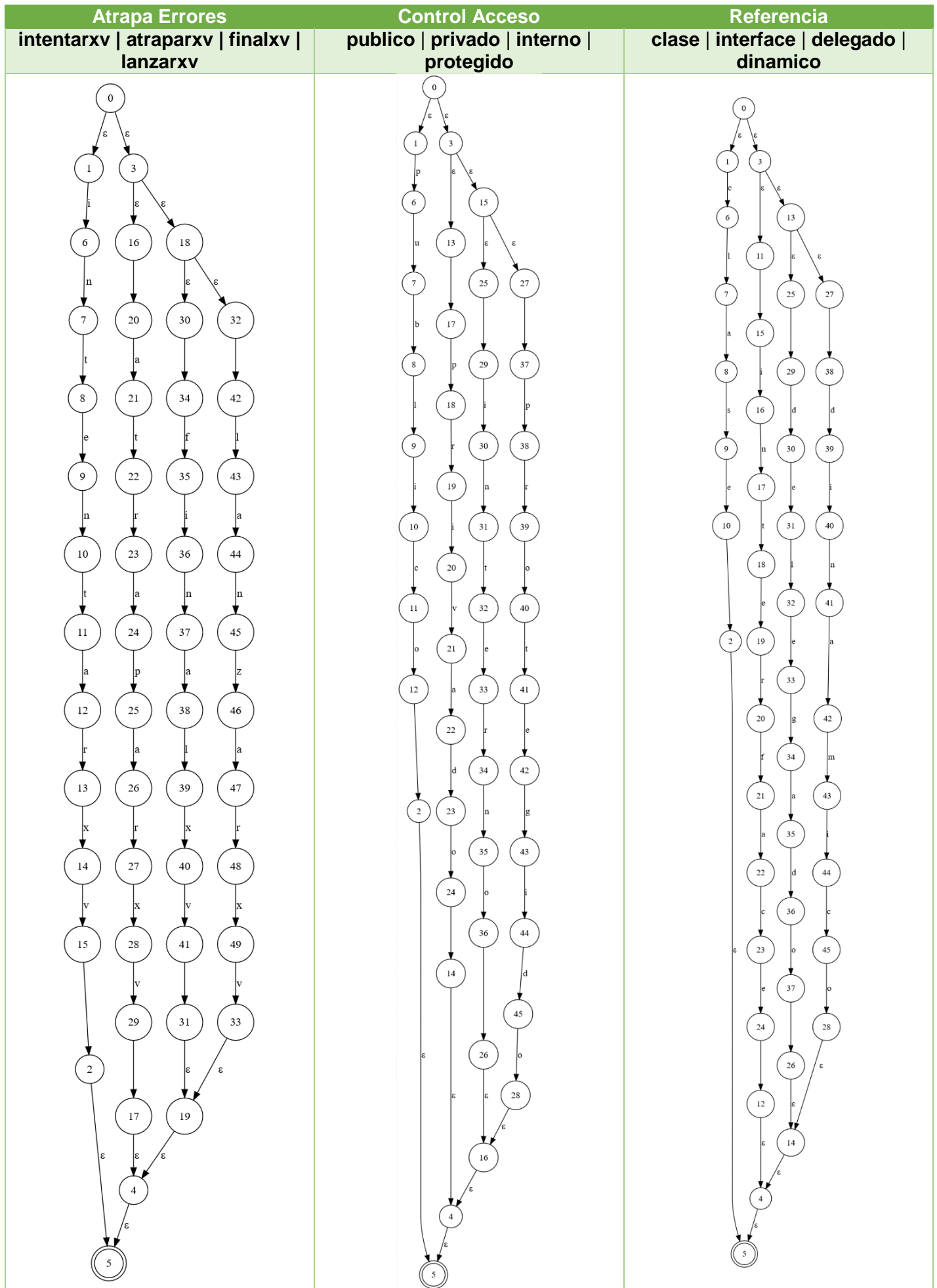


**sixv | sinoxv | noxv |
cambiarxv | casoxv**

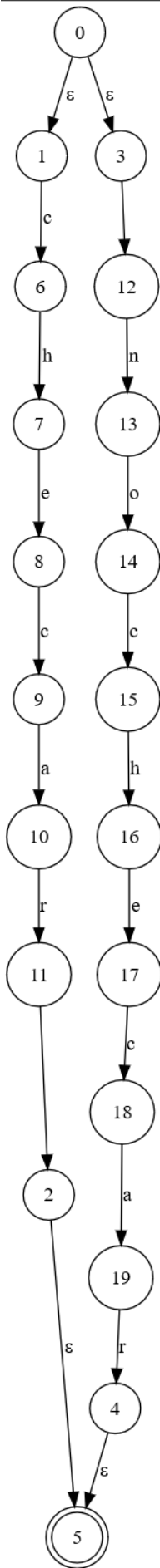
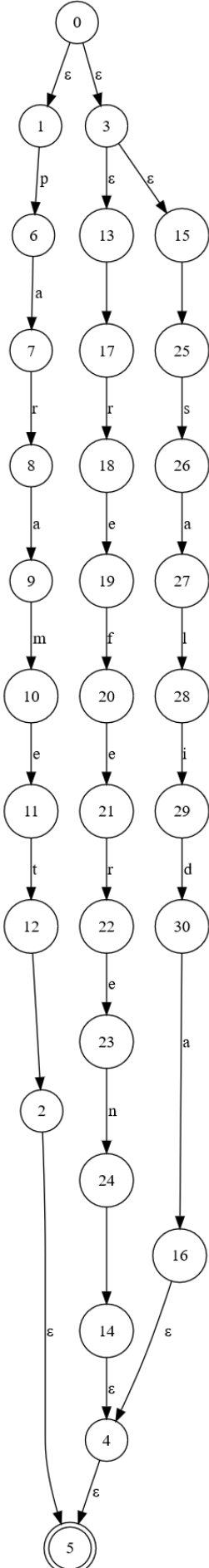
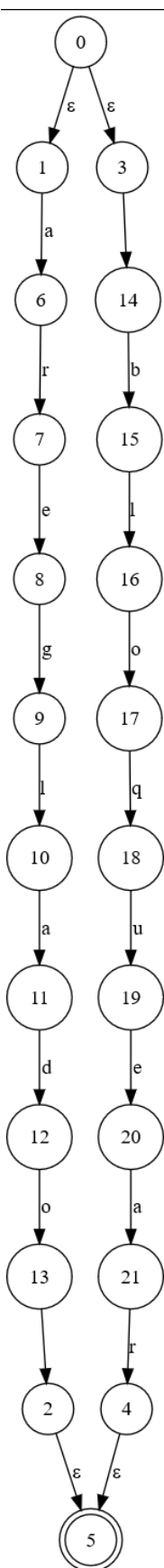


**descansarxv |
continuarxv |
retornarxv | defectoxv |
rendimiendoxv**





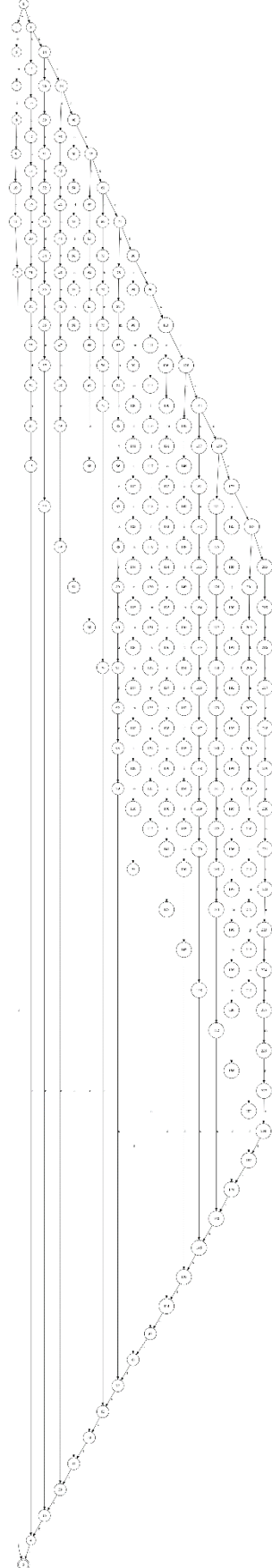
Modificadores	Clave Operadores	Clave Contextual
abstractom asincronom constantem eventom externom nuevom anularm parcialm lecturasolom selladom estaticom noguardarm virtualm volatilm	como esperar es tamañode tipode stackloco explicitoc implicitoc operadorc	obtener global parcial remove poner evaluar donde anadir

Checadores		Parametros			Revisión	
chechar nochechar		paramet	referen	salida	areglado	bloquear
						

<div> <div>Espacio Nombre</div> <div>espacionombre usando externo</div> </div>	<div> <div>Clave Acceso</div> <div>base esta nulo</div> </div>	<div> <div>Funcion Principal</div> <div>vacio inicio funcion</div> </div>
<p>NFA diagram for 'Espacio Nombre'. States: 0 to 36, 4, 5. Transitions: 0→1 (ε), 0→3 (ε), 1→6 (e), 3→19 (ε), 3→21 (ε), 6→7 (s), 19→23 (u), 21→30 (e), 7→8 (p), 23→24 (s), 30→31 (x), 8→9 (a), 24→25 (s), 31→32 (t), 9→10 (c), 25→26 (a), 32→33 (t), 10→11 (i), 26→27 (n), 33→34 (e), 11→12 (o), 27→28 (d), 34→35 (r), 12→13 (n), 28→29 (o), 35→36 (n), 13→14 (o), 29→20 (ε), 36→22 (o), 14→15 (m), 20→4 (ε), 22→4 (ε), 15→16 (b), 4→5 (ε), 16→17 (r), 17→18 (e), 18→2 (ε), 2→5 (ε).</p>	<p>NFA diagram for 'Clave Acceso'. States: 0 to 22, 13, 4, 5. Transitions: 0→1 (ε), 0→3 (ε), 1→6 (b), 3→10 (ε), 3→12 (ε), 6→7 (a), 10→14 (e), 12→19 (n), 7→8 (s), 14→15 (e), 19→20 (u), 8→9 (e), 15→16 (s), 20→21 (u), 9→2 (ε), 16→17 (t), 21→22 (l), 2→5 (ε), 17→18 (a), 22→13 (o), 13→11 (ε), 11→4 (ε), 4→5 (ε).</p>	<p>NFA diagram for 'Funcion Principal'. States: 0 to 28, 2, 4, 5. Transitions: 0→1 (ε), 0→3 (ε), 1→6 (v), 3→11 (ε), 3→13 (ε), 6→7 (a), 11→15 (i), 13→22 (f), 7→8 (c), 15→16 (i), 22→23 (f), 8→9 (i), 16→17 (n), 23→24 (u), 9→10 (o), 17→18 (i), 24→25 (n), 10→2 (ε), 18→19 (c), 25→26 (c), 2→5 (ε), 19→20 (i), 26→27 (i), 20→21 (o), 27→28 (o), 21→12 (ε), 28→14 (n), 12→4 (ε), 14→4 (ε), 4→5 (ε).</p>

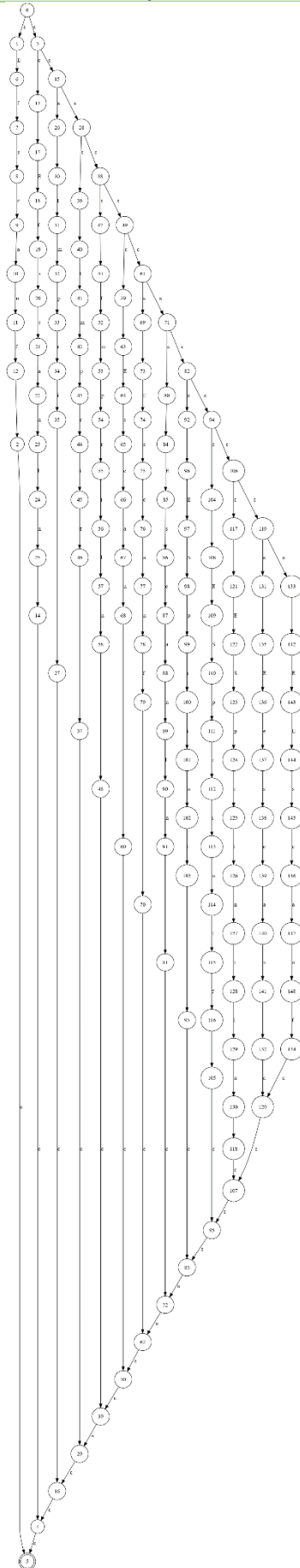
Indicadores de Formato

omitir | izquierda | derecha | interno | deca | octa | hexa | mostrarbase | mostrarpunto | letramayus |
mostrarposi | científico | arreglado | enterobuf | justarcampo | basecampo | flotantecampo



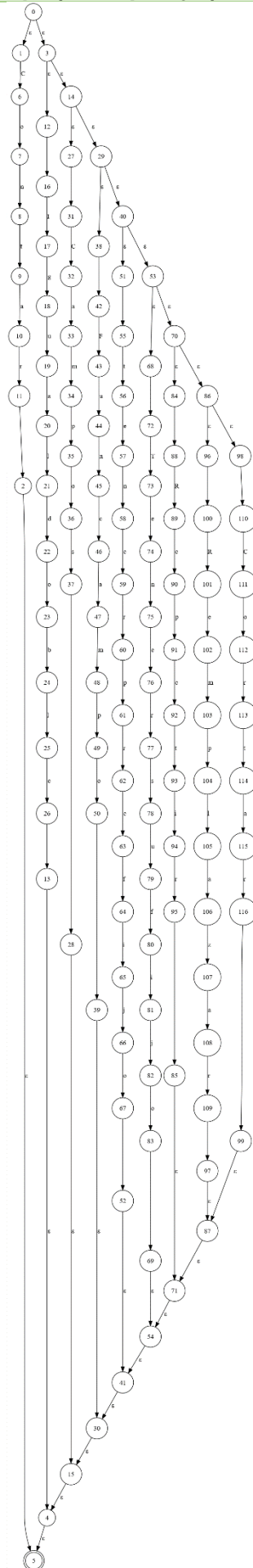
Impresión

Efscanf | Efscanln | Impri | Imprif | Impriln | Escan | Escanf | Escanln | ESprint | ESprintf | ESprintln |
Eescan | EEscaf



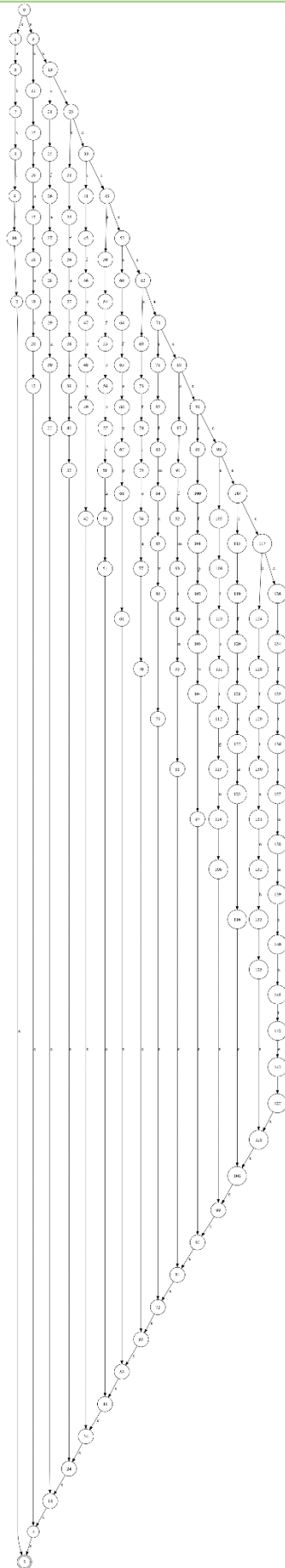
Funciones Cadenas

Contar | Igualdoble | Campos | Funcampo | tenerprefijo | Tenersufijo | Repetir | Remplazar | Cortar



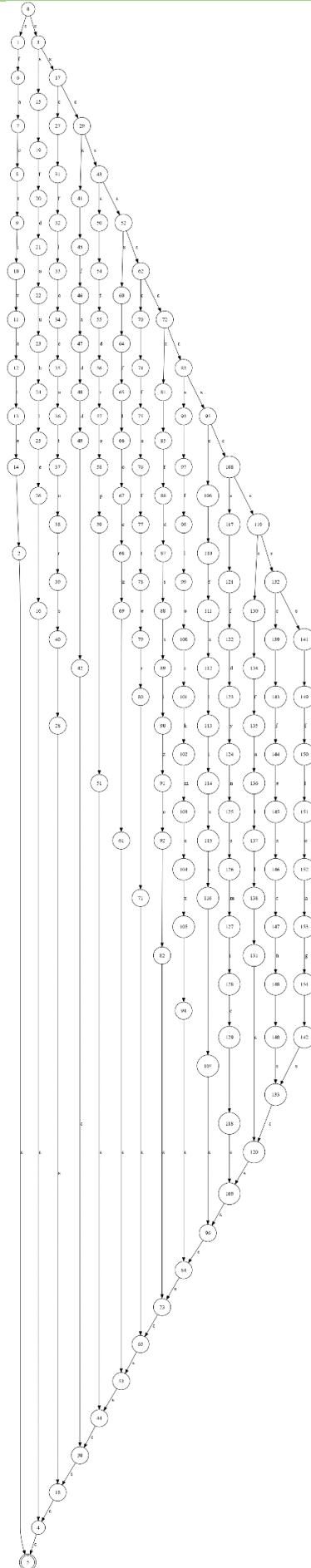
Función Matemáticas

abstr | facos | fasin | fatan | fcos | fcosh | fexp | flog | fmax | fmin | fpow | fsign | ftan | ftanh | ftruncate



Palabras Reservadas

**factivate | fdouble | floctors | fadd | fdrop | flock | faster | fdssize | flockmax | falias | fdynamic | fall
| feach | flong**



IV – Analizador Léxico

El análisis léxico es la primera fase de un compilador. Toma el código fuente del lenguaje que se escriben en forma de frases, El analizador léxico rompe esta sintaxis en una serie de “tokens”, eliminando cualquier espacio en blanco o comentarios del código fuente. Si el analizador léxico encuentra un token inválido, se genera un error. El analizador léxico trabaja en estrecha colaboración con el analizador de sintaxis. Lee flujos de caracteres del código fuente, identifica tokens válidos y pasa los datos al analizador de sintaxis.

Captura de Código del Programa

Muestra el código que se usó para la generar el Compilador Léxico, específicamente se observan las expresiones regulares y las comparaciones.

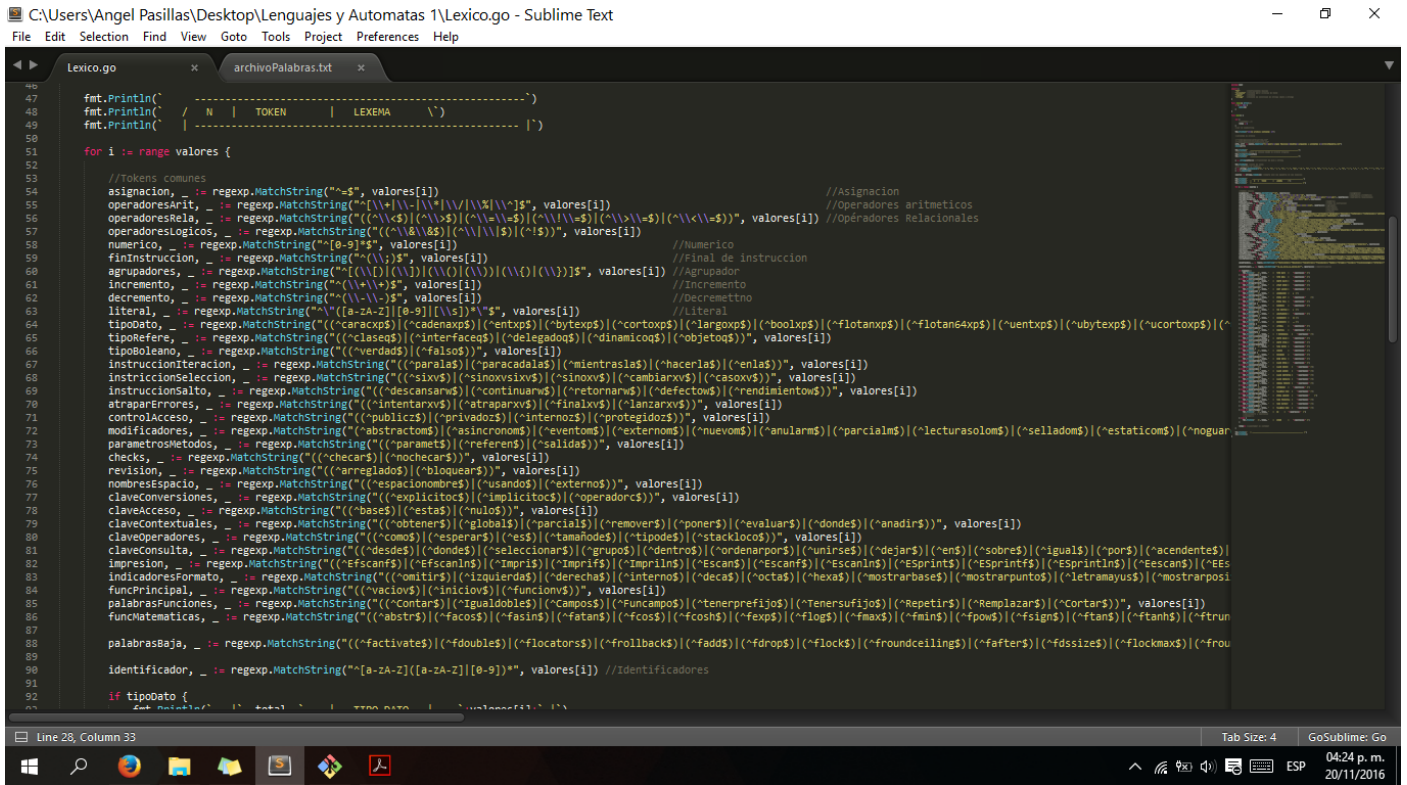


Imagen 1 – Expresiones Regulares

```

91
92     if tipoDato {
93         fmt.Println(` | TIPO DATO | `+valores[i]+` |`)
94     } else if tipoBoleano {
95         fmt.Println(` | TIPO BOOL | `+valores[i]+` |`)
96     } else if instruccionSalto {
97         fmt.Println(` | INSTR SALTO | `+valores[i]+` |`)
98     } else if atraparErrores {
99         fmt.Println(` | ATRAP ERROR | `+valores[i]+` |`)
100    } else if controlAcceso {
101        fmt.Println(` | CONT ACCESO | `+valores[i]+` |`)
102    } else if asignacion {
103        fmt.Println(` | ASIGNACION | = |`)
104    } else if operadoresArit {
105        fmt.Println(` | OPERA ARIT | `+valores[i]+` |`)
106    } else if operadoresRela {
107        fmt.Println(` | OPERA RELA | `+valores[i]+` |`)
108    } else if numerico {
109        fmt.Println(` | NUMERICO | `+valores[i]+` |`)
110    } else if finInstruccion {
111        fmt.Println(` | FIN INSTRUC | ; |`)
112    } else if agrupadores {
113        fmt.Println(` | AGRUPADORES | `+valores[i]+` |`)
114    } else if incremento {
115        fmt.Println(` | INCREMENTO | ++ |`)
116    } else if decremento {
117        fmt.Println(` | DECREMENTO | -- |`)
118    } else if literal {
119        fmt.Println(` | LITERAL | `+valores[i]+` |`)
120    } else if modificadores {
121        fmt.Println(` | MODIFICA | `+valores[i]+` |`)
122    } else if tipoRefere {
123        fmt.Println(` | T REFERENCIA | `+valores[i]+` |`)
124    } else if instruccionSeleccion {

```

Imagen 2 – Comparaciones para averiguar su Token.

Archivo de las Palabras Reservadas

Documento que lee el programa.

C:\Users\Angel Pasillas\Desktop\Lenguajes y Automatas 1\archivoPalabras.txt - Sublime Text

File Edit Selection Find View Goto Tools Project Preferences Help

```

1  = == != < > <= >= ++ -- + - * / ^ && || ! verdad falso caracxp cadenxp entxp bytexp
cortoxp largoxp boolxp flotanxp flotan64xp uentxp ubytxp ucortoxp decimalxp varxp enumxp
structxp parala paracadala mientrasla hacerla enla sixv sinoxvsixv sinoxv cambiarxv casoxv
descansarw continuarw retornarw defectow rendimientow intentarxv atraparxv finalxv lanzarxv
publicz privadoz internoz protegidoz publicz privadoz internoz protegidoz claseq interfaceq
delegadoq dinamicoq objetoq abstractom asincronom constantem eventom externom nuevomanularm
parcialm lecturasolom selladom estaticom noguardarm virtualm volatilm checar nocheckar
paramet referen salida arreglado bloquear espacionombre usando externo como esperar es
tamañode tipode stackloco explicitoc implicitoc operadorc base esta nulo obtener global
parcial remover poner evaluar donde anadir desde donde seleccionar grupo dentro ordenarpor
unirse dejar en sobre igual por acendente decendente omitir izquierda derecha interno deca
octa hexa mostrarbase mostrarpunto letramayus mostrarposi cientifico arreglado enterobuf
justarcampo basecampo flotantecampo Efscanf Efscanln Impri Imprif Impriln Escan Escanf
Escanln ESprintf ESprintf ESprintf Eescan EEsanf Contar Igualdoble Campos Funcampo
tenerprefijo Tenersufijo Repetir Remplazar Cortar vaciov iniciov funcionv abstr facos fasin
fatan fcos fcosh fexp flog fmax fmin fpow fsign ftan ftanh ftruncate factivate fdouble
flocators frollback fadd fdrop flock froundceiling faster fdssize flockmax frounddown
falias fdynamic froundfloor fall feach flong froundhalfdown feditproc floop fallow
fmaintained fasensitive frows frownnumber froutine froundup fmaterialized fencryption
fassociate fmaxvalued fdefinition flanguage frelease fvolumes frename freset fleve fctype {
} ( )

```

Line 1, Column 1780

Tab Size: 4 Plain Text

04:44 p. m. 20/11/2016

Programa Ejecutado

MINGW64:/c/Users/Angel Pasillas/Desktop/Lenguajes y Automatas 1

/	N	TOKEN	LEXEMA	\
	0	ASIGNACION	=	
	1	OPERA RELA	==	
	2	OPERA RELA	!=	
	3	OPERA RELA	<	
	4	OPERA RELA	>	
	5	OPERA RELA	<=	
	6	OPERA RELA	>=	
	7	INCREMENTO	++	
	8	DECREMENTO	--	
	9	OPERA ARIT	+	
	10	OPERA ARIT	-	
	11	OPERA ARIT	*	
	12	OPERA ARIT	/	
	13	OPERA ARIT	^	
	14	OPERA LOGICOS	&&	
	15	OPERA LOGICOS		
	16	OPERA LOGICOS	!	
	17	TIPO BOOL	verdad	
	18	TIPO BOOL	falso	
	19	TIPO DATO	caracxp	
	20	TIPO DATO	cadenaxp	
	21	TIPO DATO	entxp	
	22	TIPO DATO	bytexp	
	23	TIPO DATO	cortoxp	
	24	TIPO DATO	largoxp	
	25	TIPO DATO	boolxp	
	26	TIPO DATO	flotanxp	
	27	TIPO DATO	flotan64xp	
	28	TIPO DATO	uentxp	
	29	TIPO DATO	ubytexp	
	30	TIPO DATO	ucortoxp	
	31	TIPO DATO	decimalxp	
	32	TIPO DATO	varxp	
	33	TIPO DATO	enumxp	
	34	TIPO DATO	structxp	
	35	INSTR ITERA	para la	
	36	INSTR ITERA	paracada la	