MANUAL TÉCNICO – IDE LFYDP

Descripción: El proyecto "IDE – LFYDP" es un IDE que se encuentra en fase de desarrollo, sin embargo, el IDE actualmente es capaz de crear, abrir y editar archivos y proyectos, también es capaz de reconocer expresiones determinadas y colorearlas.

Notas:

- **IDE:** Visual Studio 2019, Community.
- **Paradigma Orientado a Objetos.**

INTERFAZ GRÁFICA (Clases):

AperturaArchivo: Clase encargada de mostrar un Windows Form, el cual recibirá una ruta de archivo, y creará un proyecto, en base a su ubicación (la cual se puede ingresar de manera manual, a través de un TextBox o se puede seleccionar a través de un FolderBrowserDialog que se mostrará presionando un botón).

CreacionDeProyecto: Clase encargada de mostrar un Windows Form, el cual recibirá el nombre del proyecto, la ubicación que tendrá el proyecto (la cual se puede ingresar de manera manual, a través de un TextBox o se puede seleccionar a través de un FolderBrowserDialog que se mostrará presionando un botón), y también le permite al usuario decidir si junto a su proyecto creará un archivo inicial (.gt) a través de un CheckBox.

NuevoArchivo: Clase encargada de mostrar un Windows Form, el cual creará un archivo (.gt) dentro de un proyecto ya abierto. El usuario únicamente debe ingresar el nombre del archivo nuevo en un TextBox.

IDE: Clase encargada de mostrar un Windows Form, el cuál muestra el IDE, compuesto por:

- **Área de Texto:** RichTextBox en el que el usuario escribirá.
- **Barra de Herramientas:** Menú desplegable, que brinda opciones al usuario.
- **Barra de Acceso Rápido:** Panel que contiene un grupo de objetos tipo Button, los cuales servirán para la facilidad del usuario, también contiene un Label, el cual indica la posición que el cursor tiene dentro del área de texto.
- **Área de Archivos de Código Fuente:** ListBox que muestra el nombre de todos los archivos que el proyecto contiene, si el usuario presiona el nombre de algún archivo, el programa debe mostrar en el área de texto el código del archivo seleccionado.
- **Área Log:** TextBox multilínea, el cuál se encuentra vació al inicio de la ejecución del programa, este TextBox debe ser llenado cuando el usuario decida compilar su código.

PantallaInicial: Clase encargada de mostrar un Windows Form, el cual le permitirá al usuario decidir si crear un proyecto, abrir un proyecto, crear un archivo o abrir un archivo; esto a través de botones.

ARCHIVOS (Clases):

Automata: Clase encargada de manejar todo lo relacionado con la validación o descarte de tokens recibidos. El autómata ha sido programado por medio de métodos, por lo cual, cada método representa cada estado del autómata y las transiciones se dan en las condiciones que cada método posee. Sus métodos a describir:

- **public bool Comprobar(String cadenaIngresada):** Método encargado de comprobar si el resultado final de las transiciones entre estados es verdadero. Cadena ingresada se refiere al token recibido.
- **private void Inicializacion():** Inicializa las palabras reservadas que el lenguaje tiene.
- **Public String Color:** Getter que devuelve el color del cuál se deberá pintar el token ingresado.

AnalizadorDeToken: Clase encargada de manejar el token, previo a mandarlo al autómata. Sus métodos a describir:

- **public AnalizadorDeToken(RichTextBox txtArea):** Constructor de la clase. El RichtTextBox crea una instancia en la clase del área de texto del IDE.
- **Private void Inicializacion():** Inicializa aquellos tokens que son el final de una cadena de dos caracteres.
- **Public void AnalizarToken():** Este método analiza los diferentes tipos de tokens que el usuario puede ingresar, desde tokens de un carácter, hasta tokens representados por una cadena de caracteres.
- **Private void ActualizarDatos():** Actualiza el número de fila y columna, en el que se encuentra el cursor que está dentro del área de texto.
- **Private void Pintar(int strt, int length):** Pinta la cadena de caracteres según lo establecido por el autómata. Strt marca el inicio de la cadena y length marca la longitud de la cadena.

AnalizadorLog: Clase encargada de Analizar los errores que se mostrarán en el área log. Sus métodos a describir:

- **Public void Analizar(RichTextBox txtArea, TextBox txtLog, int index):** Analiza los errores que se encuentran en el área de texto y los escribe en el área log. txtArea crea en la clase una instancia del área de texto del IDE, txtLog crea en la clase una instancia del área Log del IDE, index es la ubicación en la que se encuentra el cursor dentro del área de texto.

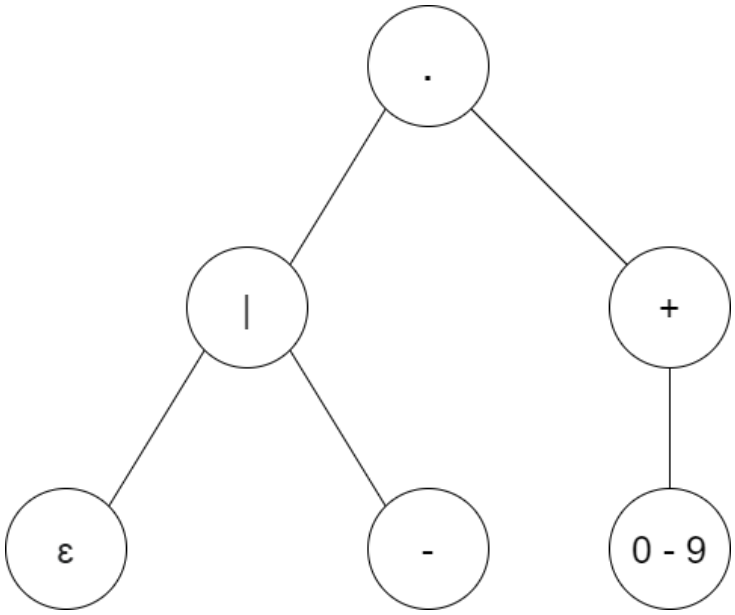
DEFINICIÓN DEL LENGUAJE:

Expresiones Regulares utilizadas para definir el lenguaje:

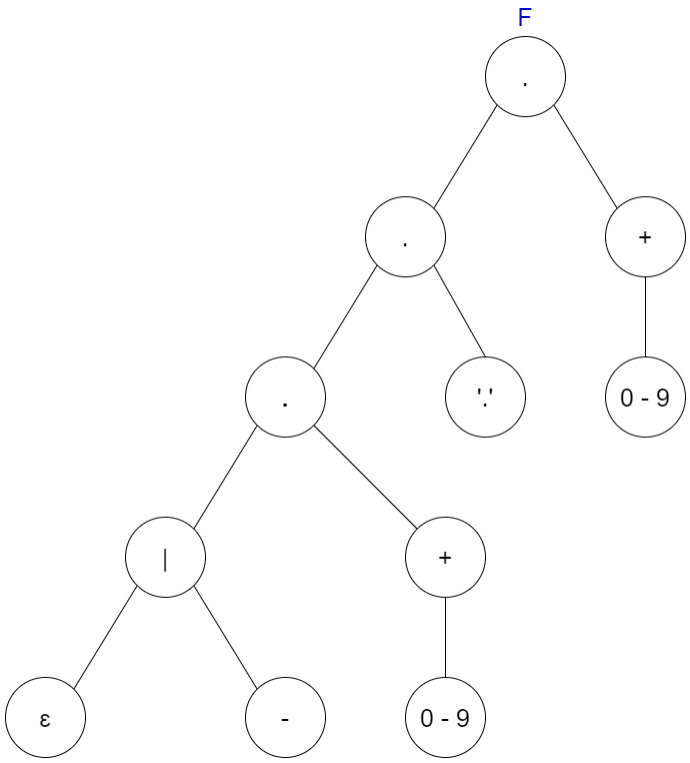
Descripción	Expresión Regular	Comentario
E - ENTERO	(- ? [0 - 9]+)	
D - DECIMAL	(- ? [0 - 9] + '.' [0 - 9]+)	
C - CADENA	(34 ([0 - 33] | [35 - 255])* 34)	Código ASCII para expresar carácter, 34 son comillas, los dos intervalos culaquier otro carácter
OA - OP. ARITMÉTICOS	[((+?+) | (-?-)) | '*' | '/']	
OR - OP.RELACIONALES	[((> | <) =?) | (=|!)=]	
OL - OP. LOGICOS	(('|' '|') | (& &) | !)	
SA - SIG. AGRUPACION	['(' | ')']	
CO - Comment	['/' '*' (char)* '*' '/'] | ['/' '/' (char)* '/n']	char hace referencia a cualquier caracter, con algunas excepciones
PR - PALABRA RESERV.	L | (L | _) +	L hace referencia únicamente a letras
AF - ASIGN. FINAL	(= | ;)	

DEFINICIÓN DEL AUTÓMATA (MÉTODO DEL ÁRBOL)

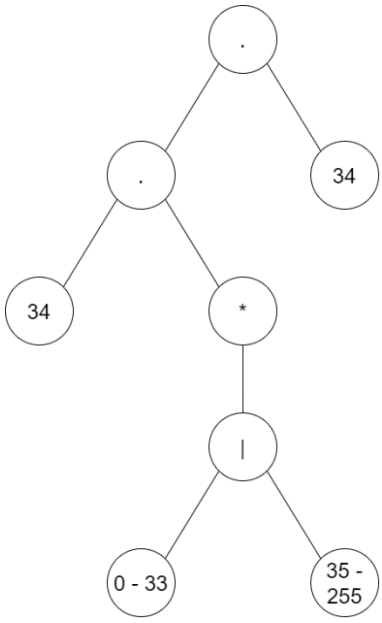
- **Enteros: (- ? [0 - 9]+)**



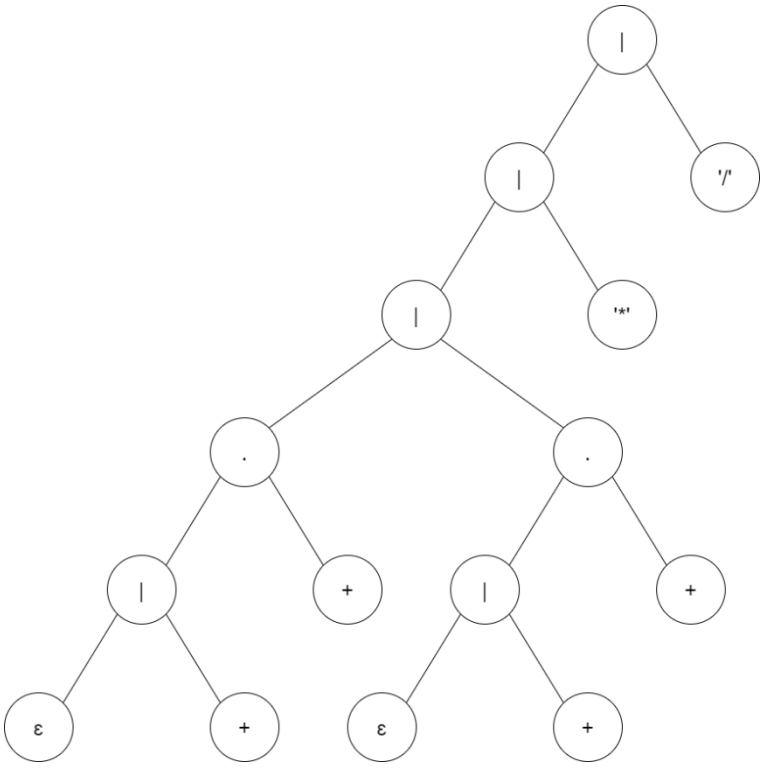
- **Decimal: (- ? [0 - 9] + '.' [0 - 9]+)**



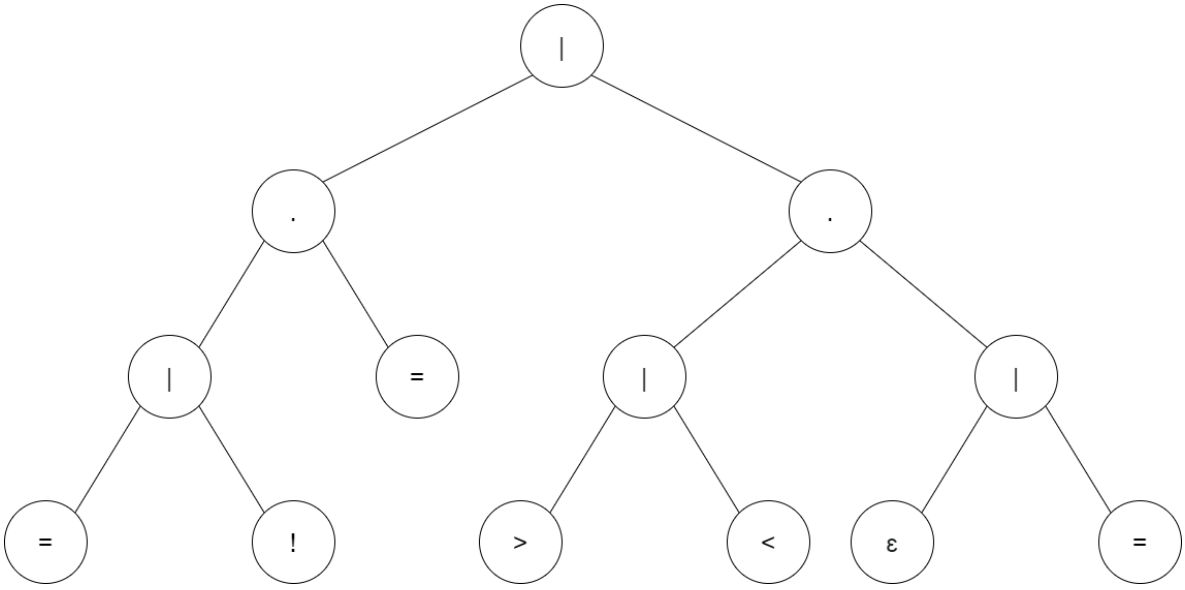
- **Cadena: (34 ([0 - 33] | [35 - 255])* 34)**



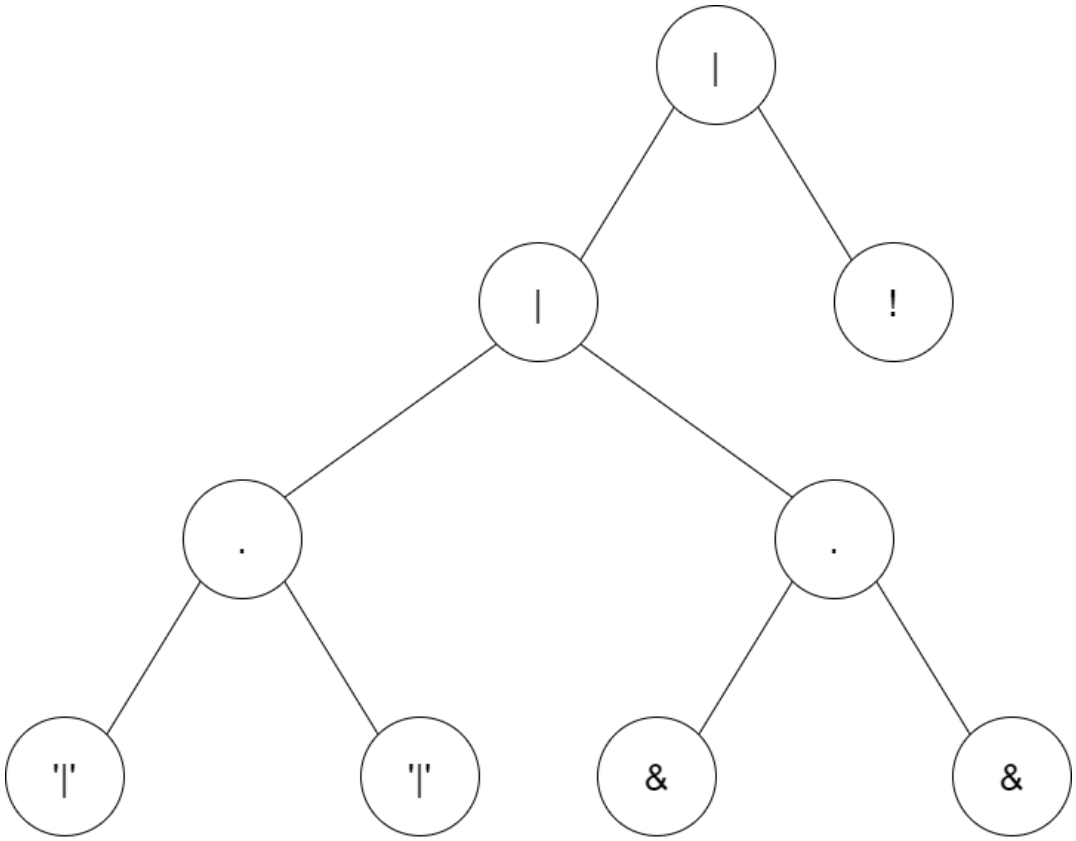
- **Operadores Aritméticos: [((+?+) | (-?-)) | '*' | '/']**



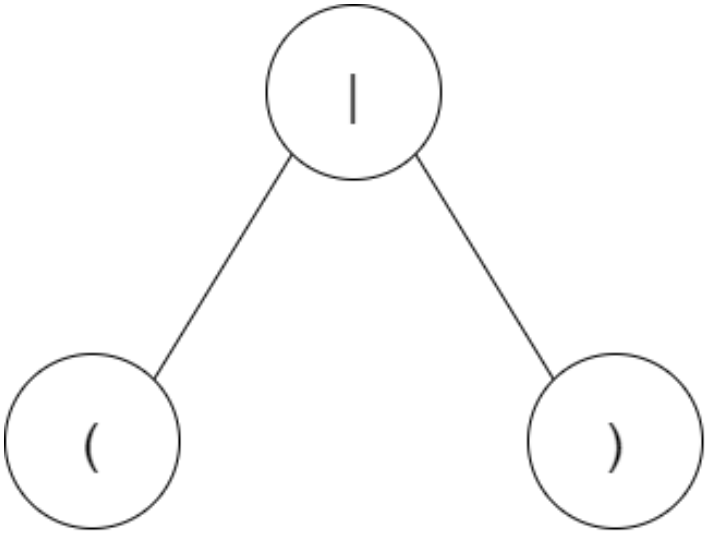
- **Operadores Relacionales: [((> | <) =?) | (=|!)=]**



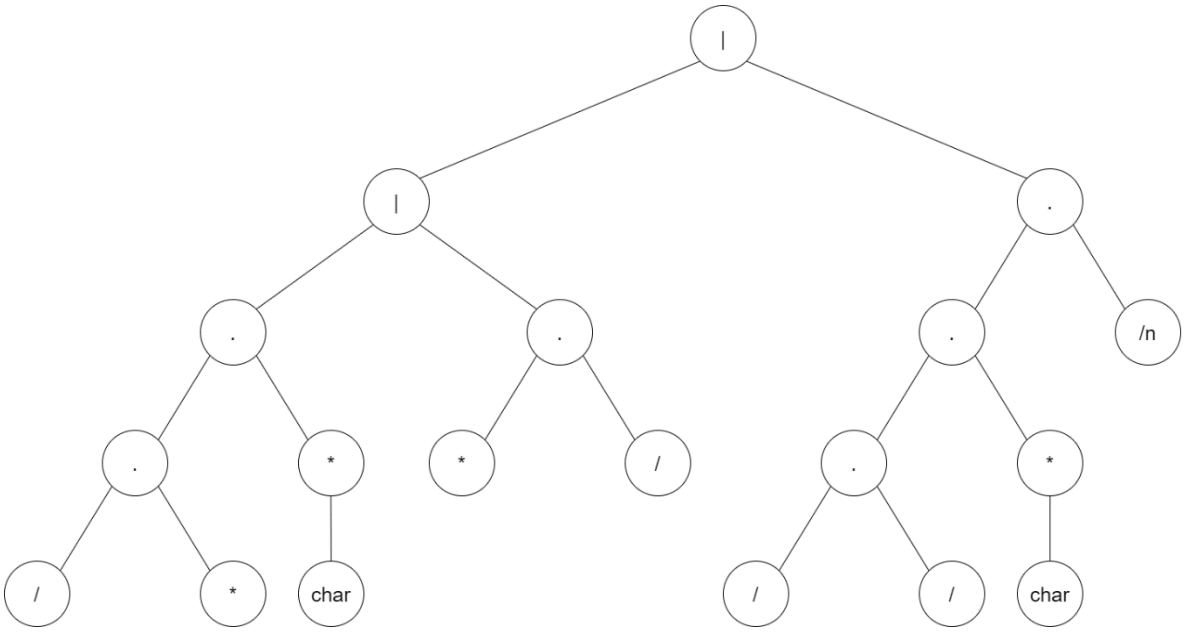
- **Operadores Lógicos: (('|' '|') | (& &) | !)**



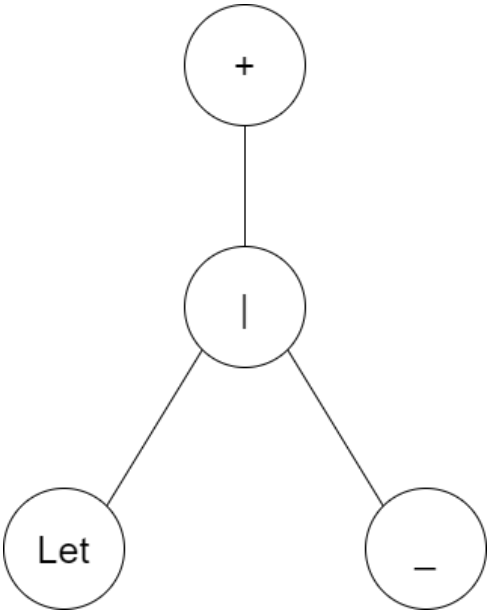
- **Signos de Agrupación: ['(' | ')']**



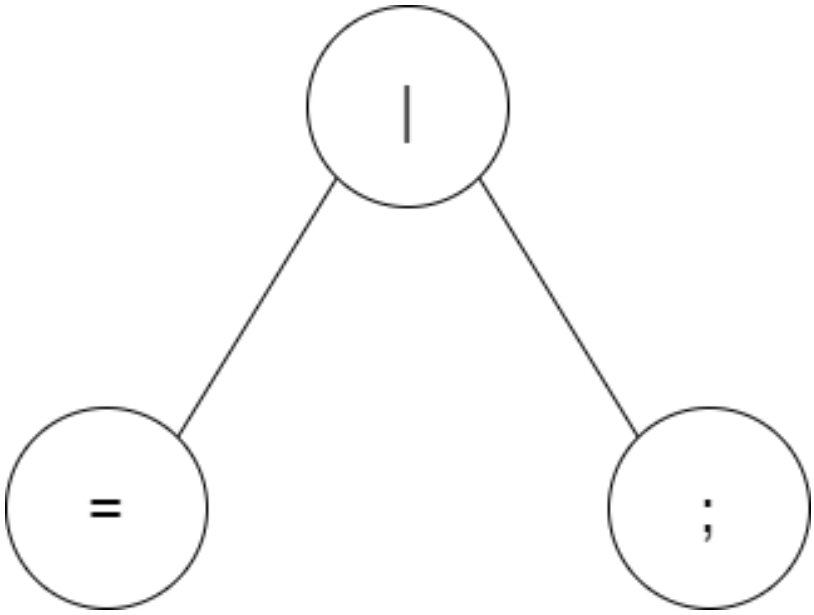
- **Comentario: ['/' '*' (char)* '*' '/'] | ['/' '/' (char)* '/n']**



- **Palabras: L | (L | _) +**

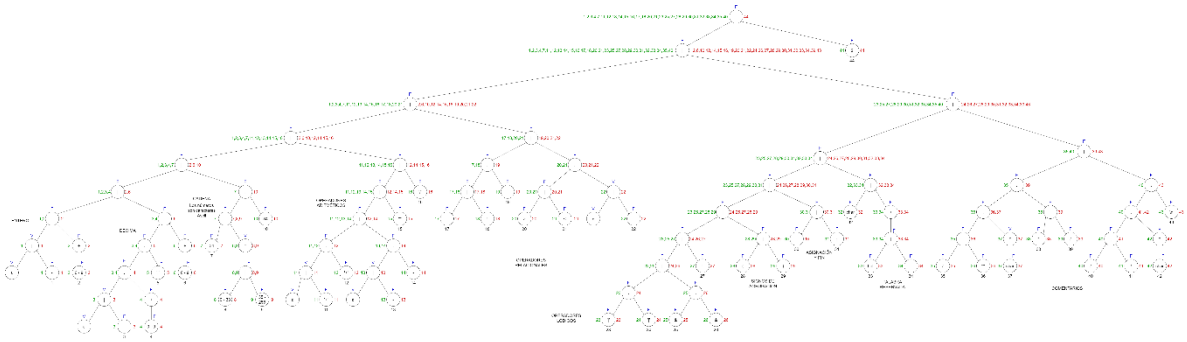


- **Asignación y Fin de Sentencia: (= | ;)**



RESOLUCIÓN:

- Se crea el árbol, se determinó cada nodo era anulable o no era anulable y se hizo el cálculo de siguientes:



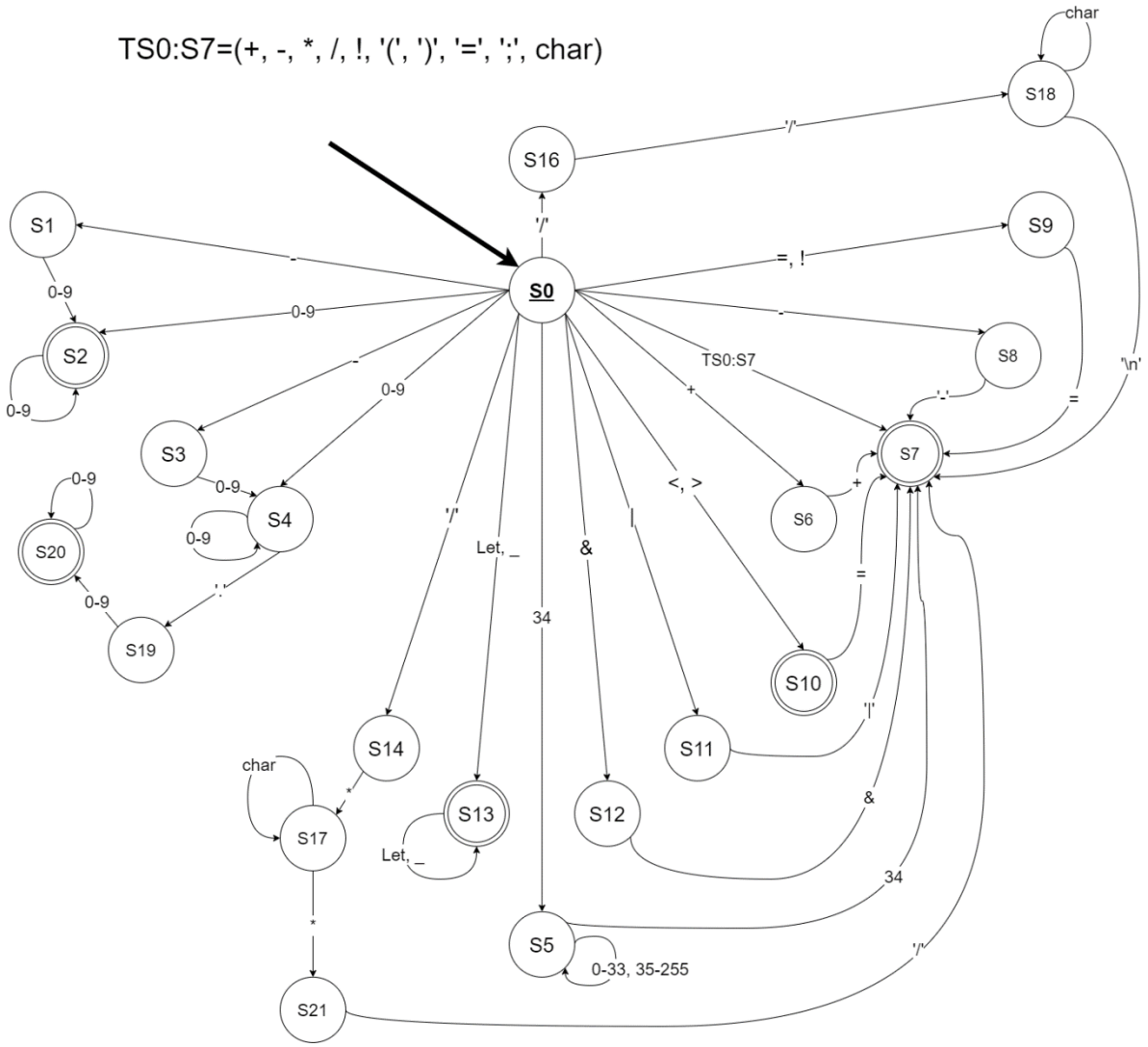
(Debido a la baja calidad de la imagen en este documento, la imagen se adjunta a la carpeta del proyecto en mejor calidad)

Se hizo la tabla de siguientes y transiciones:

No	Σ	Siguiente
1	-	2
2	0 - 9	2,44
3	-	4
4	0 - 9	4,5
5	.	6
6	0 - 9	6,44
7	34	8,9,10
8	0 - 33	8,9,10
9	35 - 255	8,9,10
10	34	44
11	+	12
12	+	44
13	-	14
14	-	44
15	*	44
16	/	44
17	=	19
18	!	19
19	=	44
20	>	22,44
21	<	22,44
22	=	44
23	|	24
24	|	44
25	&	26
26	&	44
27	!	44
28	(	44
29	)	44
30	=	44
31	;	44
32	char	44
33	Let	33,34,44
34	_	33,34,44
35	/	36
36	*	37,38
37	char	37,38
38	*	39
39	/	44
40	/	41
41	/	42,43
42	char	42,43
43	\n	44
44	$	

Q/Σ	Siguientes	Transición
S0 = 1,2,3,4,7,11,12,13,14,15,16,17,18,20,21,23,25,27,28,29,30,31,32,33,34,35,40	Sig(1)={2}=S1	δ(S0,-)=S1
	Sig(2)={2,44}=S2	δ(S0,0-9)=S2
	Sig(3)={4}=S3	δ(S0,-)=S3
	Sig(4)={4,5}=S4	δ(S0,0-9)=S4
	Sig(7)={8,9,10}=S5	δ(S0,34)=S5
	Sig(11)={12}=S6	δ(S0,+)=S6
	Sig(12)={44}=S7	δ(S0,+)=S7
	Sig(13)={14}=S8	δ(S0,-)=S8
	Sig(14)={44}=S7	δ(S0,-)=S7
	Sig(15)={44}=S7	δ(S0,*)=S7
	Sig(16)={44}=S7	δ(S0,/)=S7
	Sig(17)={19}=S9	δ(S0,=)=S9
	Sig(18)={19}=S9	δ(S0,!)=S9
	Sig(20)={22,44}=S10	δ(S0,>)=S10
	Sig(21)={22,44}=S10	δ(S0,<)=S10
	Sig(23)={24}=S11	δ(S0,|)=S11
	Sig(25)={26}=S12	δ(S0,&)=S12
	Sig(27)={44}=S7	δ(S0,'!)=S7
	Sig(28)={44}=S7	δ(S0,'(')=S7
	Sig(29)={44}=S7	δ(S0,')')=S7
	Sig(30)={44}=S7	δ(S0,=)=S7
	Sig(31)={44}=S7	δ(S0,;)=S7
	Sig(32)={44}=S7	δ(S0,char)=S7
	Sig(33)={33,34,44}=S13	δ(S0,Let)=S13
	Sig(34)={33,34,44}=S13	δ(S0,_)=S13
	Sig(35)={36}=S14	δ(S0,/)=S14
	Sig(40)={41}=S16	δ(S0,/)=S16
S1 = {2}	Sig(2)={2,44}=S2	δ(S1,0-9)=S2
S2 = {2,44}	Sig(2)={2,44}=S2	δ(S2,0-9)=S2
S3 = {4}	Sig(4)={4,5}=S4	δ(S3,0-9)=S4
S4 = {4,5}	Sig(4)={4,5}=S4	δ(S4,0-9)=S4
	Sig(5)={6}=S19	δ(S4,.)=S19
S5 = {8,9,10}	Sig(8)={8,9,10}=S5	δ(S5,0-33)=S5
	Sig(9)={8,9,10}=S5	δ(S5,35-255)=S5
	Sig(10)={44}=S7	δ(S5,34)=S7
S6 = {12}	Sig(12)={44}=S7	δ(S6,+)=S7
S7 = {44}		
S8 = {14}	Sig(14)={44}=S7	δ(S8,-)=S7
S9 = {19}	Sig(19)={44}=S7	δ(S9,=)=S7
S10 = {22,44}	Sig(22)={44}=S7	δ(S10,=)=S7
S11 = {24}	Sig(24)={44}=S7	δ(S11,|)=S7
S12 = {26}	Sig(26)={44}=S7	δ(S12,&)=S7
S13 = {33,34,44}	Sig(33)={33,34,44}=S13	δ(S13,Let)=S13
	Sig(34)={33,34,44}=S13	δ(S13,_)=S13
S14 = {36}	Sig(36)={37,38}=S17	δ(S14,*)=S17
S16 = {41]	Sig(41)={42,43}=S18	δ(S16,/)=S18
S17 = {37,38}	Sig(37)={37,38}=S17	δ(S17,char)=S17
	Sig(38)={39}=S21	δ(S17,*)=S21
S18 = {42,43}	Sig(42)={42,43}=S18	δ(S18,char)=S18
	Sig(43)={44}=S7	δ(S18,\n)=S7
S19 = {6}	Sig(6)={6,44}=S20	δ(S19,0-9)=S20
S20 = {6,44}	Sig(6)={6,44}=S20	δ(S20,0-9)=S20
S21 = {39}	Sig(39)={4}=S7	δ(S21,/)=S7

- Se obtuvo el autómata (En este caso un Autómata Finito No Determinista):



(La imagen original del autómata se adjunta en la carpeta del proyecto)

- Se hizo una tabla para convertir el AFnD en AFD

Q/Σ	-	0-9	.	34	0-33	35-255	+	*	/	=	!	>	<	|	&	(	)	;	char	Let	_	\n
Q0	S1,S3,S7,S8=Q1	S2,S4=Q2		S5=Q3			S6,S7=Q4	S7=Q5	S7,S14,S16=Q6	S7,S9=Q7	S7,S9=Q7	S1=Q8	S10=Q8	S11=Q9	S12=Q10	S7 = Q5	S7 = Q5	S7=Q5	S7=Q5	S13=Q11	S13=Q11	
Q1={S1,S3,S7,S8}	S7=Q5	S2,S4=Q2																				
Q2={S2,S4}		S2,S4=Q2	S19=Q12																			
Q3={S5}				S7=Q5	S5=Q3	S5=Q3																
Q4={S6,S7}							S7=Q5															
Q5={S7}																						
Q6={S7,S14,S16}								S17=Q13	S18=Q17													
Q7={S7,S9}										S7=Q5												
Q8={S10}										S7=Q5												
Q9={S11}														S7=Q5								
Q10={S12}															S7=Q5							
Q11={S13}																				S13=Q11	S13=Q11	
Q12={S19}		S20=Q15																				
Q13={S17}								S21=Q16												S17=Q13		
Q15={S20}		S20=Q15																				
Q16={S21}									S7=Q5													
Q17={S18}																				S18=Q17		S7=Q5

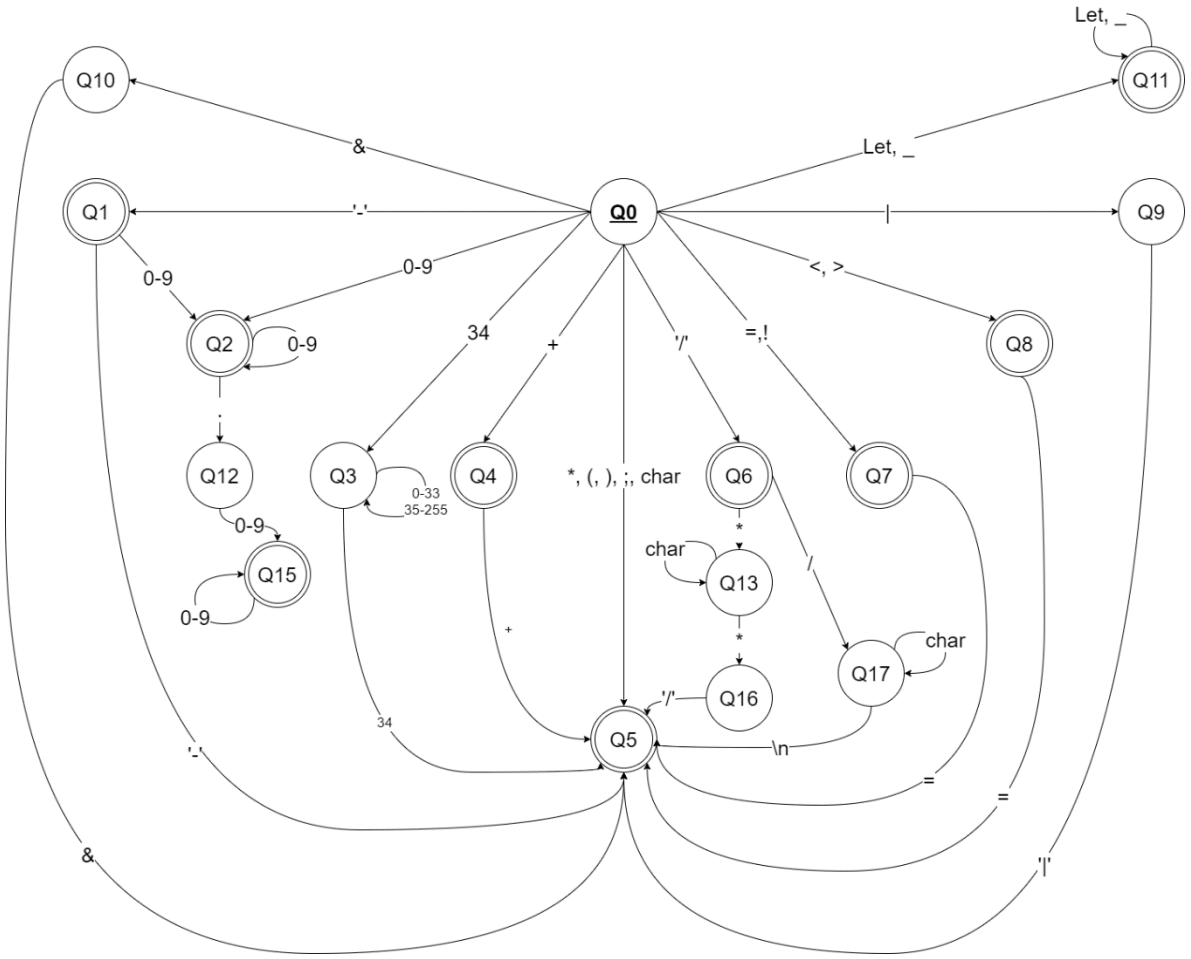
(Debido al tamaño de la tabla, la misma se adjunta a la carpeta del proyecto)

- Se hizo una tabla de transiciones:

	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17
Q0		-	0-9	34	+	*, (,), ;, char	/	'=' , !	<, >	|	&	Let, _						
Q1			0-9			-												
Q2			0-9										.					
Q3				0-33, 35-255		34												
Q4						+												
Q5																		
Q6														*				/
Q7						=												
Q8						=												
Q9						|											
Q10						&												
Q11												Let, _						
Q12																0-9		
Q13														char			*	
Q15																0-9		
Q16						/												
Q17						\n												char

(Debido al tamaño de la tabla, la misma se adjunta a la carpeta del proyecto)

- Se obtuvo el Autómata Finito Determinista



(La imagen original del autómata se adjunta en la carpeta del proyecto)

DEFINICIÓN FORMAL

A = {Q, Σ, δ, q0, F}

Q = {Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, Q10, Q11, Q12, Q13, Q15, Q16}

Σ = {-, +, '.', 0-9, &, |, <, >, =, ¡, /, *, (,), ;, CHAR, 34, 0-33, 35-255, Let, _}

Q0 = q0

F = {Q1, Q2, Q4, Q5, Q6, Q7, Q8, Q11, Q15}

Función de Transición Extendida:

δ(Q0, .) = Q1
δ(Q0, 0-9) = Q2
δ(Q0, 34) = Q3
δ(Q0, +) = Q4
δ(Q0, *) = Q5
δ(Q0, '(') = Q5
δ(Q0, ')') = Q5
δ(Q0, ;) = Q5
δ(Q0, char) = Q5
δ(Q0, /) = Q6
δ(Q0, =) = Q7
δ(Q0, '!') = Q7
δ(Q0, <) = Q8
δ(Q0, >) = Q8
δ(Q0, |) = Q9
δ(Q0, &) = Q10
δ(Q0, Let) = Q11
δ(Q0, '_') = Q11
δ(Q1, 0-9) = Q2
δ(Q1, -) = Q5
δ(Q2, 0-9) = Q2
δ(Q2, .) = Q12
δ(Q3, 0-33) = Q3
δ(Q3, 35-255) = Q3
δ(Q4, +) = Q5
δ(Q6, *) = Q13
δ(Q6, /) = Q17
δ(Q7, =) = Q5
δ(Q8, =) = Q5
δ(Q9, |) = Q5
δ(Q10, &) = Q5
δ(Q11, Let) = Q11
δ(Q11, _) = Q11
δ(Q12, 0-9) = Q15
δ(Q13, char) = Q13
δ(Q13, *) = Q16
δ(Q15, 0-9) = Q15
δ(Q16, /) = Q5
δ(Q17, \n) = Q5
δ(Q17, char) = Q17

GRAMÁTICA LL(1)

G(N, T, S, P):

- **N:**
  - Codigo
  - SimboloInicial
  - Texto
  - DeclaVariable
  - Incrementable
  - VariablesL
  - id
  - idVar
  - valor
  - TipoValor
  - Suma
  - Suma’
  - Multi
  - Multi’
  - Pot
  - Pot’
  - Unar
  - Elem
  - Impresión
  - ValImprimir
  - Imprimir
  - Append
  - Lectura
  - ValLeer
  - Condicion
  - DescCondicion
  - CuerpoDescCond
  - Cond2
  - ValCondicion
  - DescValCondicion
  - Condicionante
  - SigDif
  - CondicionantePlus
  - CicloM
  - CicloHM
  - DescCicloHM
  - CicloF
  - EstructCicloF
  - EstructDesde
  - EstructHasta
  - EstructIncremento
  - desde
  - hasta
- **T:**
  - PRprincipal
  - (
  -)
  - {
  - }
  - PRDatoPrimi
  - TokId
  - ++
  - --
  - ,
  - ;
  - cadena
  - entero
  - decimal
  - booleano
  - char
  - Num
  - PRImprimir
  - +
  - –
  - *
  - /
  - ^
  - PRLeer
  - PRSI
  - =
  - PRSINO_SI
  - PRSINO
  - Numero
  - OpRel
  - PRBooleana
  - !
  - OpLogico
  - PRHACER
  - PRMIENTRAS
  - PRDESDE
  - PRHASTA
  - PRINCREMENTO
  - NumE
- **S:** Codigo

- **P: Producción**

Codigo	→	PRprincipal SimboloInicial Texto
SimboloInicial	→	( ) {
Texto	→	VariablesL Texto | DeclaVariable Texto | Impresión Texto | Lectura Texto | Condicion Texto | CicloM Texto | CicloHM Texto | CicloF Texto | }
DeclaVariable	→	TokId Incrementable ;
Incrementable	→	++ | -- | = TipoValor
VariablesL	→	PRDatoPrimi idVar id
id	→	, idVar id | ;
idVar	→	TokId valor
valor	→	= TipoValor | e
TipoValor	→	cadena | Suma | booleano | char
Suma	→	Multi Suma’

Suma’	→	+ Multi Suma’ | - Multi Suma’ | e
Multi	→	Pot Multi’
Multi’	→	* Pot Multi’ | / Pot Multi’ | e
Pot	→	Unar Pot’
Pot’	→	^ Pot | e
Unar	→	- Elem | Elem
Elem	→	TokId | Num | (Suma)
Impresión	→	PRImprimir ValImprimir
ValImprimir	→	(Imprimir Append
Imprimir	→	TokId | cadena | Num | char
Append	→	+ Imprimir Append |) ;
Lectura	→	PRLeer ValLeer ;
ValLeer	→	(TokId)
Condicion	→	PRSI DescCondicion Cond2

DescCondicion	→	ValCondicion CuerpoDescCond
CuerpoDescCond	→	{ Texto
Cond2	→	PRSINO_SI DescCondicion Cond2 | PRSINO CuerpoDescCond | e
ValCondicion	→	(DescValCondicion)
DescValCondicion	→	SigDif Condicionante CondicionantePlus
Condicionante	→	Suma OpRel Suma | PRBooleana
SigDif	→	! SigDif | e
CondicionantePlus	→	OpLogico DescValCondicion | e
CicloM	→	PRMIENTRAS ValCondicion CuerpoDescCond
CicloHM	→	DescCicloHM ValCondicion
DescCicloHM	→	PRHACER CuerpoDescCond PRMIENTRAS
CicloF	→	EstructCicloF CuerpoDescCond
EstructCicloF	→	EstructDesde EstructHasta EstructIncremento

EstructDesde	→	PRDESDE desde
EstructHasta	→	PRHASTA hasta
EstructIncremento	→	PRINCREMENTO NumE
desde	→	TokId = NumE
hasta	→	TokId OpRel NumE

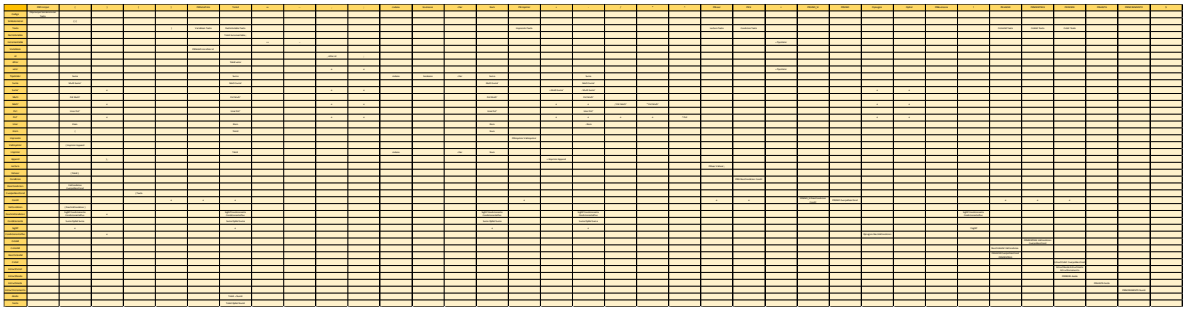
Primeros:

No Terminal	Primeros
Codigo	PRprincipal
SimboloInicial	(
Texto	PRDatoPrimi, TokId, PRImprimir, PRLeer, PRSI, PRMIENTRAS, PRHACER, PRDESDE. }
DeclaVariable	TokId
Incrementable	++, --, =
VariablesL	PRDatoPrimi
id	",", ;
idVar	TokId
valor	=, e
TipoValor	cadena, -, TokId, Num, (, booleano, char
Suma	-, TokId, Num, (
Suma’	+, -, e
Multi	-, TokId, Num, (
Multi’	*, /, e
Pot	-, TokId, Num, (
Pot’	^, e
Unar	-, TokId, Num, (
Elem	TokId, Num, (
Impresión	PRImprimir
ValImprimir	(
Imprimir	TokId, cadena, Num, char
Append	+,)
Lectura	PRLeer
ValLeer	(
Condicion	PRSI
DescCondicion	(
CuerpoDescCond	{
Cond2	PRSINO_SI, PRSINO, e
ValCondicion	(
DescValCondicion	!, -, TokId, Num, (, e
Condicionante	-, TokId, Num, (
SigDif	!, e
CondicionantePlus	OpLogico, e
CicloM	PRMIENTRAS
CicloHM	PRHACER
DescCicloHM	PRHACER
CicloF	PRDESDE
EstructCicloF	PRDESDE
EstructDesde	PRDESDE
EstructHasta	PRHASTA
EstructIncremento	PRINCREMENTO
desde	TokId
hasta	TokId

Siguientes:

No Terminal	Siguientes
Codigo	$
SimboloInicial	PRDatoPrimi, TokId, PRImprimir, PRLeer, PRSI, PRMIENTRAS, PRHACER, PRDESDE. }
Texto	$, PRSINO_SI, PRSINO, PRDatoPrimi, TokId, PRImprimir, PRLeer, PRSI, PRMIENTRAS, PRHACER, PRDESDE, }, PRMIENTRAS, e
DeclaVariable	PRDatoPrimi, TokId, PRImprimir, PRLeer, PRSI, PRMIENTRAS, PRHACER, PRDESDE. }
Incrementable	;
VariablesL	PRDatoPrimi, TokId, PRImprimir, PRLeer, PRSI, PRMIENTRAS, PRHACER, PRDESDE. }
id	PRDatoPrimi, TokId, PRImprimir, PRLeer, PRSI, PRMIENTRAS, PRHACER, PRDESDE. }
idVar	",", ;
valor	",", ;
TipoValor	;, ",", , ;
Suma	;, ",", ;,), OpRel, OpLogico, e
Suma’	;, ",", ;,), OpRel, OpLogico, e
Multi	+, -, e, ';, ",", ;,), OpRel, OpLogico, e
Multi’	+, -, e, ';, ",", ;,), OpRel, OpLogico, e
Pot	*, /, e, +, -, e, ';, ",", ;,), OpRel, OpLogico, e
Pot’	*, /, e, +, -, e, ';, ",", ;,), OpRel, OpLogico, e
Unar	*, /, e, +, -, e, ';, ",", ;,), OpRel, OpLogico, e
Elem	*, /, e, +, -, e, ';, ",", ;,), OpRel, OpLogico, e
Impresión	PRDatoPrimi, TokId, PRImprimir, PRLeer, PRSI, PRMIENTRAS, PRHACER, PRDESDE. }
ValImprimir	PRDatoPrimi, TokId, PRImprimir, PRLeer, PRSI, PRMIENTRAS, PRHACER, PRDESDE. }
Imprimir	+,)
Append	PRDatoPrimi, TokId, PRImprimir, PRLeer, PRSI, PRMIENTRAS, PRHACER, PRDESDE. }
Lectura	PRDatoPrimi, TokId, PRImprimir, PRLeer, PRSI, PRMIENTRAS, PRHACER, PRDESDE. }
ValLeer	;
Condicion	PRDatoPrimi, TokId, PRImprimir, PRLeer, PRSI, PRMIENTRAS, PRHACER, PRDESDE. }
DescCondicion	PRSINO_SI, PRSINO, PRDatoPrimi, TokId, PRImprimir, PRLeer, PRSI, PRMIENTRAS, PRHACER, PRDESDE. }, e
CuerpoDescCond	PRSINO_SI, PRSINO, PRDatoPrimi, TokId, PRImprimir, PRLeer, PRSI, PRMIENTRAS, PRHACER, PRDESDE, }, PRMIENTRAS, e
Cond2	PRDatoPrimi, TokId, PRImprimir, PRLeer, PRSI, PRMIENTRAS, PRHACER, PRDESDE. }
ValCondicion	{, pSig(CicloHM)
DescValCondicion	)
Condicionante	OpLogico,), e
SigDif	-, TokId, Num, (
CondicionantePlus	)
CicloM	PRDatoPrimi, TokId, PRImprimir, PRLeer, PRSI, PRMIENTRAS, PRHACER, PRDESDE. }
CicloHM	PRDatoPrimi, TokId, PRImprimir, PRLeer, PRSI, PRMIENTRAS, PRHACER, PRDESDE. }
DescCicloHM	(
CicloF	PRDatoPrimi, TokId, PRImprimir, PRLeer, PRSI, PRMIENTRAS, PRHACER, PRDESDE. }
EstructCicloF	{
EstructDesde	PRHASTA
EstructHasta	PRINCREMENTO
EstructIncremento	{
desde	PRHASTA
hasta	PRINCREMENTO

Tabla de Análisis Sintáctico:



(Debido al Tamaño de la tabla, la misma es ilegible, por lo tanto se añade a la carpeta “Documentación Fase 2” un archivo de excel el cuál contiene la tabla).

Diagrama de Clases