MANUAL DE USUARIO

AUTÓMATA FINITO DETERMINÍSTICO: LEXER

Programa: Analizador Léxico para Expresiones Aritméticas.

Lenguaje Utilizado: Python.

Versión: 3.12.0.

Descripción: El Analizador Léxico para Expresiones Aritméticas es una herramienta desarrollada en Python que permite analizar expresiones aritméticas escritas en un archivo de texto, identificar los diferentes componentes de dichas expresiones (tokens) y determinar su tipo correspondiente.

Los tipos de tokens que el programa puede identificar se declaran a continuación

1. **Número Entero:** Este tipo de token representa valores numéricos enteros. Se definen como secuencias de dígitos del conjunto {0-9}, sin incluir decimales ni signos de puntuación. Ejemplos de números enteros son 0, 123, -45, etc.

2. **Número Real:** Los tokens de este tipo representan valores numéricos con parte fraccionaria. Están compuestos por una parte entera seguida de un punto decimal y una parte fraccionaria, ambas partes compuestas por dígitos del conjunto {0-9}. Ejemplos de números reales son 3.14, -0.5, 10.0, etc. También pueden usar notación exponencial con la letra E, mayúscula o minúscula, pero después de la letra E sólo puede ir un entero positivo o negativo (e.g. 2.3E3, 6.345e-5, -0.001E-3, .467E9).

3. **Variable:** Representa identificadores utilizados para denotar valores variables en las expresiones aritméticas. Las variables pueden consistir en una combinación de letras (mayúsculas o minúsculas), dígitos y guiones bajos, pero deben comenzar con una letra. Ejemplos de variables válidas son ‘x’, ‘y’, ‘variable_1’, etc.

4. **Operación:** Este tipo de token denota operadores aritméticos utilizados en las expresiones, tales como suma (+), resta (-), multiplicación (*), división (/) y potenciación (^).

5. **Paréntesis que Abre:** Indica el inicio de un grupo de elementos dentro de una expresión aritmética. Este token es un símbolo de apertura de paréntesis "(" que delimita un conjunto de operandos y operadores.

6. **Paréntesis que Cierra:** Marca el final de un grupo de elementos dentro de una expresión aritmética. Este token es un símbolo de cierre de paréntesis ‘)’ que indica el fin de un conjunto de operandos y operadores iniciado por un paréntesis que abre.

7. **Comentario:** Representa anotaciones o explicaciones dentro del código fuente que no afectan la ejecución del programa. En este contexto, los comentarios comienzan con el símbolo ‘/’ y pueden extenderse hasta el final de la línea.

Instrucciones de Ejecución:

1. Asegúrese de que Python esté correctamente instalado en tu sistema.

2. Descargue el código fuente proporcionado y almacénelo en una ubicación accesible en su sistema con el nombre ‘lexer_aritmetico.py’.

3. Prepare un archivo de texto (.txt) que contenga las expresiones aritméticas que desee analizar. Cada expresión debe estar en una línea separada, siguiendo un formato legible. Por ejemplo:

   3 + (4 * 2)

   5 - 2.5

   x = 10 / (y + 2)

4. Guarde este archivo en la misma ubicación que el código fuente del programa. Es crucial que el archivo se nombre como ‘input.txt’. Si decide usar un nombre diferente, asegúrese de actualizar la línea 465 del programa con el nombre correspondiente. A continuación, se ejemplifica el cambio:

   465 file= "input.txt" → **465** file=” nombre_del_archivo.txt”

5. Abra una terminal o línea de comandos en su sistema operativo.

6. Diríjase al directorio donde ha guardado tanto el archivo ‘lexer_aritmetico.py’ como el archivo de entrada ‘input.txt’.

7. Ejecuta el programa utilizando el siguiente comando en la terminal o línea de comandos: *python lexer_aritmetico.py*

8. El programa analizará las expresiones aritméticas del archivo de entrada y presentará los resultados de manera legible en la terminal.

9. Se generará una tabla que exhibe los tokens identificados junto con sus respectivos tipos. Un ejemplo de la salida se presenta a continuación:

Token	Tipo
3	Número Entero
+	Suma
(	Paréntesis que Abre
4	Número Entero
*	Multiplicación
2	Número Entero
)	Paréntesis que Cierra
+	Suma
5	Número Entero
-	Resta
2.5	Número Real

10. Si se detecta alguna inconsistencia, se notificará al usuario con un mensaje de error específico que indica la presencia de un problema. Los errores que el autómata es capaz de detectar son los siguientes:

    - El autómata puede detectar si hay un desequilibrio en el número de paréntesis que abren y cierran en una expresión aritmética.

    - El autómata es capaz de detectar si se hace referencia a una variable que no ha sido declarada previamente.

    - El autómata puede identificar si una variable que se intenta declarar no cumple con las reglas establecidas para la declaración de variables.

    - Puede identificar si hay operadores en la expresión que no tienen operandos asociados. Por ejemplo, si hay un operador + o - al principio o al final de la expresión sin operandos correspondientes.

    - El autómata puede señalar si hay operandos en la expresión que no tienen operadores asociados. Por ejemplo, si hay números o variables consecutivos sin ningún operador entre ellos.

    - El autómata puede detectar si se hace un uso incorrecto de signos, como un signo negativo seguido de un signo positivo, lo cual no tiene sentido en una expresión aritmética válida.

ANEXOS

1. Características del Autómata Finito Determinista

Alfabeto

$\sum =$

{0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z,+,-,*,/,.,^,=,),(,_}

Estados

$\boldsymbol{Q} =$ {qInicial, qVariable, qVarAsignacion, qConstructorComentario, qComentario, qNumeroEntero, qParentesisAbre, qParentesisCierra, qNumeroReal, qNumeroExponencial, qEnteroExp, qDivision, qMultiplicacion, qSuma, qResta, qPotencia, qSignoNegtivo, qTerminal}

Estado Inicial

$\boldsymbol{q_0} =$ qInicial

Estados Finales

$\boldsymbol{A} =$ {qTerminal}

Función de Transiciones

δ

(qInicial, [a-z] [A-Z]) → qVariable

(qInicial, [0-9]) → qNumeroEntero

(qInicial, "- ") → qSignoNegativo

(qInicial, "(") → qParéntesisAbre

(qInicial, "/") → qConstructorComentario

(qInicial, ["\n"] [""]) → qTerminal

(qVariable, ["\n"] [""]) → qTerminal

(qVariable, [a-z] [A-Z]) → qVariable

(qVariable, [0-9]) → qVariable

(qVariable, "_") → qVariable

(qVariable, "=") → qVarAsignación

δ

(qVariable, “+”) → qSuma

(qVariable, “-”) → qResta

(qVariable, “*”) → qMultiplicación

(qVariable, “/”) → qDivision

(qVariable, “^”) → qPotencia

(qVariable, “)”) → qParentesisCierra

(qNumeroEntero, [“\n”] [“”]) → qTerminal

(qNumeroEntero, [0-9]) → qNumeroEntero)

(qNumeroEntero, “.”) → qNumeroReal)

(qNumeroEntero, “+”) → qSuma

(qNumeroEntero, “-”) → qResta

(qNumeroEntero, “*”) → qMultiplicacion

(qNumeroEntero, “/”) → qDivision

(qNumeroEntero, “^”) → qPotencia

(qNumeroEntero, “)”)→qParentesisCierra

(qSignoNegativo, [0-9]) → qNumeroEntero

(qSignoNegativo, [a-z] [A-Z]) → qVariable

(qSignoNegativo, “(“) → qParentesisAbre

(qConstructorComentario, “/“) → qComentario

(qParentesisAbre, [a-z] [A-Z]) → qVariable

(qParentesisAbre, [0-9]) → qNumeroEntero

(qParentesisAbre, “- “) → qSignoNegativo

(qParentesisAbre, “(“) → qParentesisAbre

(qNumeroReal, [“\n”] [“”]) → qTerminal

(qNumeroReal, [0-9]) → qNumeroReal

(qNumeroReal, [“E”] [“e”]) → qNumeroExponencial

(qNumeroReal, “+”) → qSuma

(qNumeroReal, “-”) → qResta

(qNumeroReal, “*”) → qMultiplicacion

δ

(qNumeroReal, “/”) → qDivision

(qNumeroReal, “^”) → qPotencia

(qNumeroReal, “)”) → qParentesisCierra

(qNumeroExponencial, [0-9]) → qEnteroExp

(qNumeroExponencial, “- “) → qEnteroExp

(qEnteroExp, [“\n”] [“”]) → qTerminal

(qEnteroExp, [0-9]) → qEnteroExp

(qEnteroExp, “+”) → qSuma

(qEnteroExp, “-”) → qResta

(qEnteroExp, “*”) → qMultiplicacion

(qEnteroExp, “/”) → qDivision

(qEnteroExp, “^”) → qPotencia

(qEnteroExp, “)”) → qParentesisCierra

(qParentesisCierra, [“\n”] [“”]) → qTerminal

(qParentesisCierra, “+”) → qSuma

(qParentesisCierra, “-”) → qResta

(qParentesisCierra, “*”) → qMultiplicacion

(qParentesisCierra, “/”) → qDivision

(qParentesisCierra, “^”) → qPotencia

(qParentesisCierra, “)”) → qParentesisCierra

(qSuma, [0-9]) → qNumeroEntero

(qSuma, [a-z] [A-Z]) → qVariable

(qSuma, “(“) → qParentesisAbre

(qResta, [0-9]) → qNumeroEntero

(qResta, [a-z] [A-Z]) → qVariable

(qResta, “(“) → qParentesisAbre

(qMultiplicacion, [0-9]) → qNumeroEntero

(qMultiplicacion, [a-z] [A-Z]) → qVariable

(qMultiplicacion, “(“) → qParentesisAbre

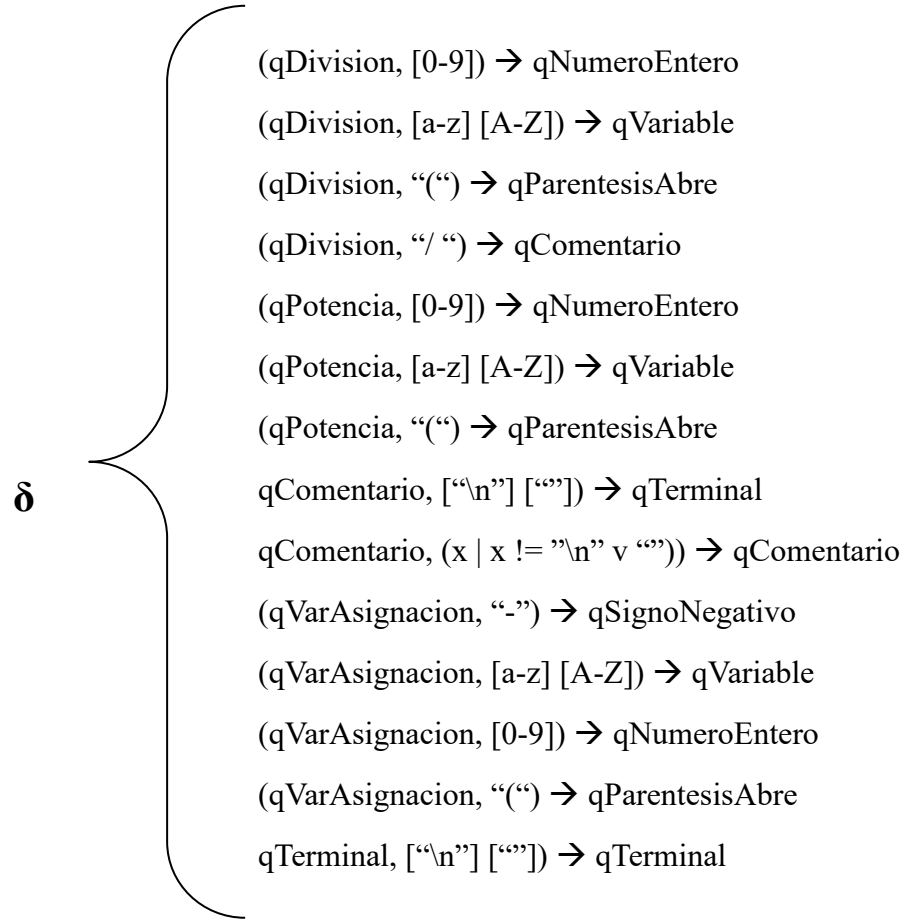
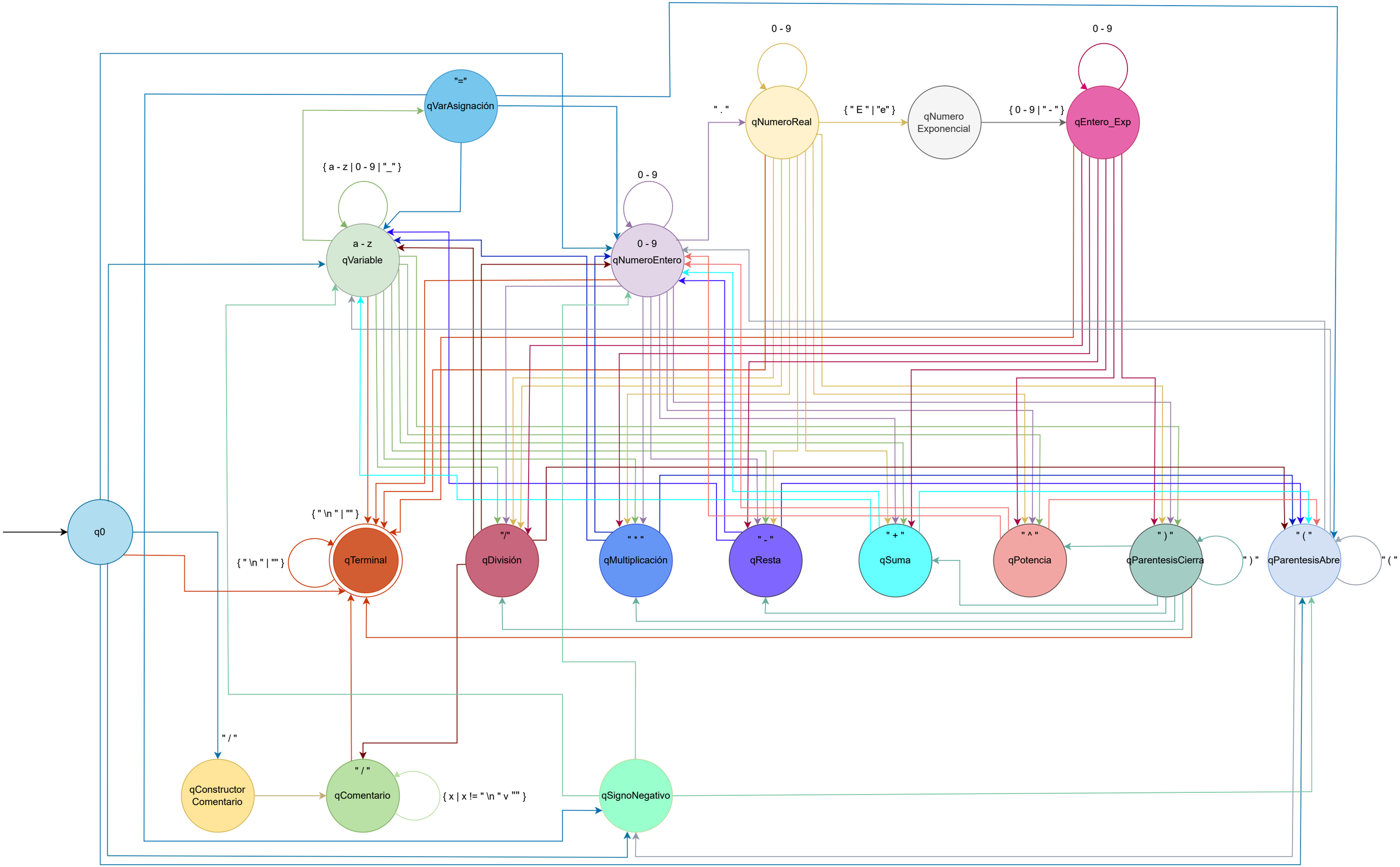


Tabla de Transiciones

Estado	Símbolo de Entrada	Estado de Transición
qInicial	(a-z) (A-Z)	qVariable
qInicial	(0-9)	qNumeroEntero
qInicial	“-“	qSignoNegativo
qInicial	“(“	qParentesisAbre
qInicial	“/”	qConstructorComentario
qInicial	(“\n”) (“”)	qTerminal
qVariable	(“\n”) (“”)	qTerminal
qVariable	(a-z) (A-Z)	qVariable
qVariable	(0-9)	qVariable
qVariable	“_”	qVariable
qVariable	“=”	qVarAsignación
qVariable	“+”	qSuma

qVariable	“-“	qResta
qVariable	“*”	qMultiplicación
qVariable	“/”	qDivision
qVariable	“^”	qPotencia
qVariable	“)”	qParentesisCierra
qNumeroEntero	(“\n”) (“”)	qTerminal
qNumeroEntero	(0-9)	qNumeroEntero
qNumeroEntero	(“.”)	qNumeroReal
qNumeroEntero	“+”	qSuma
qNumeroEntero	“-“	qResta
qNumeroEntero	“*”	qMultiplicacion
qNumeroEntero	“/”	qDivision
qNumeroEntero	“^”	qPotencia
qNumeroEntero	“)”	qParentesisCierra
qSignoNegativo	(0-9)	qNumeroEntero
qSignoNegativo	(a-z) (A-Z)	qVariable
qSignoNegativo	“(“	qParentesisAbre
qConstructorComentario	“/”	qComentario
qParentesisAbre	(a-z) (A-Z)	qVariable
qParentesisAbre	(0-9)	qNumeroEntero
qParentesisAbre	“-“	qSignoNegativo
qParentesisAbre	“(“	qParentesisAbre
qNumeroReal	(“\n”) (“”)	qTerminal
qNumeroReal	(0-9)	qNumeroReal
qNumeroReal	(“E”) (“e”)	qNumeroExponencial
qNumeroReal	“+”	qSuma
qNumeroReal	“-“	qResta
qNumeroReal	“*”	qMultiplicacion
qNumeroReal	“/”	qDivision
qNumeroReal	“^”	qPotencia
qNumeroReal	“)”	qParentesisCierra
qNumeroExponencial	(0-9)	qEnteroExp
qNumeroExponencial	“-“	qEnteroExp
qEnteroExp	(“\n”) (“”)	qTerminal
qEnteroExp	(0-9)	qEnteroExp
qEnteroExp	“+”	qSuma
qEnteroExp	“-“	qResta
qEnteroExp	“*”	qMultiplicacion
qEnteroExp	“/”	qDivision
qEnteroExp	“^”	qPotencia

qEnteroExp	“)”	qParentesisCierra
qParentesisCierra	(“\n”) (“”)	qTerminal
qParentesisCierra	“+”	qSuma
qParentesisCierra	“-”	qResta
qParentesisCierra	“*”	qMultiplicacion
qParentesisCierra	“/”	qDivision
qParentesisCierra	“^”	qPotencia
qParentesisCierra	“)”	qParentesisCierra
qSuma	(0-9)	qNumeroEntero
qSuma	(a-z) (A-Z)	qVariable
qSuma	“(“	qParentesisAbre
qResta	(0-9)	qNumeroEntero
qResta	(a-z) (A-Z)	qVariable
qResta	“(“	qParentesisAbre
qMultiplicación	(0-9)	qNumeroEntero
qMultiplicación	(a-z) (A-Z)	qVariable
qMultiplicación	“(“	qParentesisAbre
qDivision	(0-9)	qNumeroEntero
qDivision	(a-z) (A-Z)	qVariable
qDivision	“(“	qParentesisAbre
qDivision	“/”	qComentario
qPotencia	(0-9)	qNumeroEntero
qPotencia	(a-z) (A-Z)	qVariable
qPotencia	“(“	qParentesisAbre
qComentario	(“\n”) (“”)	qTerminal
qComentario	{x | x != “\n “ v “”}	qComentario
qVarAsignacion	“-“	qSignoNegativo
qVarAsignacion	(a-z) (A-Z)	qVariable
qVarAsignacion	(0-9)	qNumeroEntero
qVarAsignacion	“(“	qParentesisAbre
qTerminal	(“\n”) (“”)	qTerminal


"="
qVarAsignación
{ a - z | 0 - 9 | "_" }
a - z
qVariable
0 - 9
0 - 9
qNumeroEntero
" . "
0 - 9
qNumeroReal
{ " E " | "e" }
qNumero
Exponencial
{ 0 - 9 | " - " }
0 - 9
qEntero_Exp
q0
{ " \n " | "" }
{ " \n " | "" }
qTerminal
"/"
qDivisión
" * "
qMultiplicación
" - "
qResta
" + "
qSuma
" ^ "
qPotencia
") "
qParentesisCierra
") "
" ("
qParentesisAbre
" ("
" / "
qConstructor
Comentario
" / "
qComentario
{ x | x != " \n " v "" }
qSignoNegativo