Universidad de San Carlos de Guatemala
Facultad de ingeniería
Escuela de ciencias y sistemas

Manual de Gramáticas

Fernando Arturo Pensamiento Calderón
3011-83457-0101
201602743

Introducción

Para realizar este proyecto se necesitó realizar una gramática en jison para analizar código de alto nivel.

Objetivos

1. Identificar expresiones regulares utilizadas.
2. Identificar los símbolos terminales.
3. Identificar la precedencia utilizada
4. Identificar símbolos no terminales
    - Explicar en que se utilizan cada uno de estos
5. Explicar las acciones realizadas en los nodos
6. Mostrar la gramatica

Expresiones regulares utilizadas.

```
\s+                                     // se ignoran espacios en blanco
"//".*                                  // comentario simple línea
[/][*][^*]*[*]+([^/*][^*]*[*]+)*[/]     // comentario multiple líneas
["\""](("\\""\"")|[^"\""])*["\""]       // Cadenas
\'[^\'']\'                              // char
[0-9]+("."[0-9]+)\b                     // decimal
[0-9]+\b                                // entero


([a-zA-Z0-9_.-])+(".j")                 //archivo
([a-zA-Z])[a-zA-Z0-9_]*                 //identificador
```

Las expresiones sirven para reconocer cosas que no son palabras reservadas y funcionan igual que las palabras reservadas como terminales

Lista de terminales

Todos los terminales están escritos en mayúsculas.

Los que inician con R seguido de una pablara son producciones de la palabra seguida a la primera letra R

1. RNULL
2. RIMPORT
3. RFALSE
4. RTRUE
5. RSWITCH
6. RCONTINUE
7. RPRIVATE
8. RDEFINE
9. RTRY
10. RINTEGER
11. RVAR
12. RCASE
13. RRETURN
14. RVOID
15. RAS
16. RCATCH
17. RDOUBLE
18. RCONST
19. RIF
20. RDEFAULT
21. RPRINT
22. RFOR
23. RSTRC
24. RTHROW
25. RCHAR
26. RGLOBAL
27. RELSE
28. RBREAK

29. RPUBLIC
30. RWHILE
31. RDO
32. RBOOLEAN
33. COMA ,
34. PCOMA ;
35. ALLAVE {
36. CLLAVE }
37. APAR (
38. CPAR)
39. ACORCH [
40. CCORCH]
41. AND &&
42. OR ||
43. DÓLAR $
44. MASMAS ++
45. MENOSMENOS --
46. MAS +
47. MENOS -
48. POR *
49. DIVIDIDO /
50. POTENCIA ^^
51. MODULO %
52. DOSPUNTOSIGUAL :=
53. MENORIGUAL <=
54. MAYORIGUAL >=
55. TRESIGUAL ===
56. IGUALIGUAL ==
57. DIFIGUAL ¡=
58. MENOR <
59. MAYOR >
60. IGUAL =
61. XOR ^
62. NOT ¡
63. DOSPTS :
64. PUNTO .
65. CADENA la expresión regular de cadena
66. CHAR: la expresión regular de CHAR
67. DECIMAL: la expresión regular de decimal
68. ENTERO : la expresión regular de entero
69. ARCHIVO: la expresión regular de archivo
70. IDENTIFICADOR: la expresión regular de identificador

Se cuentan con 70 no terminales.

Precedencia utilizada

```
/* Asociación de operadores y precedencia */
%izquierda MASMAS, MENOSMENOS
%izquierda XOR
%izquierda OR
%izquierda AND
%izquierda IGUALIGUAL, DIFIGUAL, TRESIGUAL
%izquierda MAYOR, MENOR, MENORIGUAL, MAYORIGUAL
%izquierda MAS, MENOS
%izquierda POR, DIVIDIDO, MODULO
%izquierda POTENCIA
%derecha   UMENOS, NOT
%izquierda APAR,  CPAR
%izquierda ACORCH, CCORCH
```

NO.	Símbolos	Lado de precedencia	Nivel de prioridad
1	++ , --	Izquierda	1
2	^ (xor)	Izquierda	2
3	||	Izquierda	3
4	&&	Izquierda	4
5	== , ¡= , ===	Izquierda	5
6	<, > , <= , >=	Izquierda	6
7	+, -	Izquierda	7
8	*, / , %(modulo)	Izquierda	8
9	^^(potencia)	Izquierda	9
10	-(UMENOS) , ¡ (not)	Derecha	10
11	()	Izquierda	11
12	[]	Izquierda	12

Símbolos no terminales

1	ini	Donde inicia todo
2	instrucciones_cuerpo	Es una lista de instrucciones_c
3	instruccion_c	Contiene lo que puede ir en el ambito global, funciones, declaración y estructuras
4	declararfuncion	sirve para declarar una función
5	params	sirve para poner o no parámetros
6	params2	Contiene una lista de parámetros.
7	instrucciones_fun	Lista de instrucción_f
8	instruccion_f	Contiene lo que puede ir en el cuerpo de una función que es casi todo menos declaración de funciones.
9	try_catch	el try y catch
10	defest	definición de estructuras
11	listaest	listado de los atributos de las estructuras
12	aest	el atributo de la estructura
13	fprint	función para imprimir
14	llamadaMetodo	sirve para llamar a las funciones o métodos

15	lista_params	puede tener o no tener una lista de parámetros
16	lista_params2	es la lista de parámetros
17	param	el parámetro como tal
18	sparam	para ver si se manda el parámetro por referencia o valor
19	asignacion	sirve para la asignación
20	otro	sirve para lista de accesos. Por si en asignación no es id = exp;
21	otro2	tiene identificador o []
22	otrofin	tienen identificador.identificador o identificador[]
23	ciclos	llama al for, while y do_while
24	ifor	es el inicio del for el cual o es asignación o nada
25	mfor	tiene el medio del for, una expresión o nada
26	ffor	tiene el final del for que es una asignación o incemento_decremento o nada
27	cases	los casos en el switch
28	transferencia	break, return y continue
29	switch	la gramática del swtich
30	si	la gramática del sí (else y else if)
31	declaracion	Sirve para declarar una variable.
32	pvalor	para declaración ya que se puede igualar a una expresión o a un casteo o a una declaración de arreglo o estructura
33	valarray	definición de los arreglos o llaman a arr value
34	arrvalue	declaración de arreglos del modo {exp , exp}
35	listavarr	es la lista que contiene los elementos del arreglo
36	elemarr	es el elemento de un arreglo
37	tipo	tiene tp o tp con corchetes para hacer un arreglo
38	tp	tiene los diferentes tipos integer, char …
39	listaID	sirve para la lista de identificadores de la declaración.
40	import	guarda un import
41	listaimport	contiene una lista de imports
42	exp	contine las operaciones que devuelven un valor, llamadas a funciones, expresiones aritméticas y otras cosas.
43	inc_dec	para identificador ++ o --
44	visibilidad	public | private aunque se quitó del enunciado no lo quite de la gramática
45	fin	para que venga o no el ; y este sea opcional

En cada no terminal lo que se realiza es la construcción de un AST esto se hace gracias a una clase abstracta llamada nodo la cual tiene un arreglo de el mismo y lo único que se hace en cada no terminal es ir construyendo este árbol.

Gramatica.

```
--Tomaremos Ɛ como vacío--

ini::= import instrucciones_cuerpo EOF
    |  instrucciones_cuerpo EOF

instrucciones_cuerpo::= instrucciones_cuerpo instruccion_c
    | Ɛ

instruccion_c ::=  declaracion fin
        | visibilidad declararfuncion
        | declararfuncion
        | defest
        | error PCOMA

declararfuncion ::= tipo IDENTIFICADOR APAR params CPAR ALLAVE instrucciones_fun CLLAVE
                |   RVOID IDENTIFICADOR APAR params CPAR  ALLAVE instrucciones_fun CLLAVE

params ::= params2
         | Ɛ

params2 ::=   params2 COMA tipo IDENTIFICADOR
        | tipo IDENTIFICADOR

instrucciones_fun ::= instrucciones_fun instruccion_f
                    | Ɛ

instruccion_f ::= declaracion fin
                | defest
                | si
                | switch
                | transferencia
                | cases
                | asignacion fin
                | ciclos
                | llamadaMetodo fin
                | fprint fin
                | inc_dec fin
                | try_catch
                | error PCOMA

try_catch ::= RTRY  ALLAVE instrucciones_fun CLLAVE RCATCH APAR IDENTIFICADOR IDENTIFICADOR C
PAR ALLAVE instrucciones_fun CLLAVE

defest ::= RDEFINE IDENTIFICADOR RAS ACORCH  listaest   CCORCH fin

listaest ::= listaest COMA aest
        |  aest

aest ::=      tipo IDENTIFICADOR
```

```
        | tipo IDENTIFICADOR IGUAL exp


fprint ::= RPRINT APAR exp CPAR

llamadaMetodo ::= IDENTIFICADOR  lista_params CPAR

lista_params ::=  APAR
        | APAR lista_params2

lista_params2 ::= param
            | lista_params2 COMA param

param ::=  sparam
    | IDENTIFICADOR IGUAL sparam

sparam ::= exp
    | DOLAR exp

asignacion ::= IDENTIFICADOR  IGUAL pvalor
            | otro IGUAL pvalor

otro ::=  otro PUNTO otro2
        | otrofin

otro2 ::= IDENTIFICADOR ACORCH exp CCORCH
    |   IDENTIFICADOR
    |   IDENTIFICADOR  lista_params CPAR

otrofin ::= IDENTIFICADOR ACORCH exp CCORCH
    |     IDENTIFICADOR PUNTO IDENTIFICADOR
    |     IDENTIFICADOR PUNTO IDENTIFICADOR  lista_params CPAR

ciclos::=  RWHILE APAR exp CPAR ALLAVE instrucciones_fun CLLAVE | RDO ALLAVE instrucciones_fu
n CLLAVE RWHILE APAR exp CPAR fin
        | RFOR APAR ifor PCOMA mfor PCOMA ffor CPAR ALLAVE instrucciones_fun CLLAVE


ifor ::= declaracion | Ɛ
mfor ::= exp         | Ɛ
ffor ::= asignacion  | Ɛ

cases ::=  RCASE exp DOSPTS
    |    RDEFAULT DOSPTS

transferencia ::= RBREAK fin
                | RCONTINUE fin
                | RRETURN PCOMA
                | RRETURN exp PCOMA
```

```
switch ::= RSWITCH APAR exp CPAR ALLAVE instrucciones_fun CLLAVE

si ::= RIF APAR exp CPAR ALLAVE instrucciones_fun CLLAVE
    | RIF APAR exp CPAR ALLAVE instrucciones_fun CLLAVE RELSE ALLAVE instrucciones_fun CLLAVE
    | RIF APAR exp CPAR ALLAVE instrucciones_fun CLLAVE RELSE si

declaracion ::=   tipo listaID IGUAL pvalor                    //declaracion tipo 1
            |   RVAR IDENTIFICADOR DOSPUNTOSIGUAL pvalor      //declaracion tipo 2
            |   RCONST IDENTIFICADOR DOSPUNTOSIGUAL pvalor    //declaracion tipo 3
            |   RGLOBAL IDENTIFICADOR DOSPUNTOSIGUAL pvalor   //declaracion tipo 4
            |   tipo listaID                                   //Declaracion tipo 5

pvalor ::= exp
        | valarray

valarray ::= RSTRC tp ACORCH exp CCORCH
        | RSTRC IDENTIFICADOR ACORCH exp CCORCH
        |  arrvalue


arrvalue ::= ALLAVE listavarr CLLAVE

 listavarr ::= listavarr COMA elemarr
        | elemarr

elemarr ::= exp

tipo ::=tp
    | tp ACORCH CCORCH
    | IDENTIFICADOR
    | IDENTIFICADOR ACORCH CCORCH

tp ::= RINTEGER
    | RDOUBLE
    | RCHAR
    | RBOOLEAN


listaID ::=  listaID COMA IDENTIFICADOR
        | IDENTIFICADOR

import ::= RIMPORT  listaimport fin

listaimport ::= listaimport COMA ARCHIVO
    |  ARCHIVO

exp
    ::= MENOS exp %prec UMENOS
    | NOT exp
    | exp MAS exp
```

```
    | exp MENOS exp
    | exp POR exp
    | exp DIVIDIDO exp
    | exp MODULO exp
    | exp POTENCIA exp
    | exp XOR exp
    | exp OR  exp
    | exp AND  exp
    | exp IGUALIGUAL exp
    | exp DIFIGUAL exp
    | exp TRESIGUAL exp
    | exp MAYOR exp
    | exp MENOR exp
    | exp MAYORIGUAL exp
    | exp MENORIGUAL exp
    | APAR exp CPAR
    | RFALSE
    | RTRUE
    | ENTERO
    | DECIMAL
    | CADENA
    | CHAR
    | IDENTIFICADOR
    | otro
    | inc_dec
    | llamadaMetodo
    | APAR RINTEGER CPAR exp
    | APAR RCHAR CPAR exp

inc_dec::=IDENTIFICADOR MASMAS
    |   IDENTIFICADOR MENOSMENOS

visibilidad ::= RPUBLIC | RPRIVATE  ;
fin ::= PCOMA |  Ɛ;
```