Universidad Nacional Autónoma de México
Semestre 2020-2
Compiladores
Gramática
Profesor: Adrián Ulises Mercado Martínez

# Gramática Final

1. programa → declaraciones funciones
2. declaraciones → tipo lista_var**;** declaraciones
   | tipo_registro lista_var**;** declaraciones
   | $\varepsilon$
3. tipo_registro → **estructura inicio** declaraciones **fin**
4. tipo → base tipo_arreglo
5. base → **ent** | **real** | **dreal** | **car** | **sin**
6. tipo_arreglo → **[num]** tipo_arreglo | $\varepsilon$
7. lista_var → lista_var**,** **id** | **id**
8. funciones → **def** tipo **id**(argumentos) **inicio** declaraciones sentencias **fin** funciones
   | $\varepsilon$
9. argumentos → listar_arg | **sin**
10. lista_arg → lista_arg**,** arg | arg
11. arg → tipo_arg **id**
12. tipo_arg → base param_arr
13. param_arr → [ ] param_arr | $\varepsilon$
14. sentencias → sentencias sentencia | sentencia
15. sentencia → **si** e_bool **entonces** sentencia **fin**
    | **si** e_bool **entonces** sentencia **sino** sentencia **fin**
    | **mientras** e_bool **hacer** sentencia **fin**
    | **hacer** sentencia **mientras** e_bool**;**
    | **segun** (variable) **hacer** casos predeterminado **fin**
    | variable **:=** expresion **;**
    | **escribir** expresion **;**
    | **leer** variable **;** | **devolver;**
    | **devolver** expresion**;**
    | **terminar;**
    | **inicio** sentencias **fin**
16. casos → **caso num:** sentencia casos | **caso num:** sentencia
17. predeterminado → **pred:** sentencia | $\varepsilon$
18. e_bool → e_bool **o** e_bool | e_bool **y** e_bool | **no** e_bool | **(** e_bool **)**
    | relacional | **verdadero** | **falso**
19. relacional → relacional oprel relacional | expresion
20. oprel → > | < | >= | <= | <> | =
21. expresion → expresion oparit expresion
    | expresion **%** expresion | **(**expresion **)** | **id**
    | variable | **num** | **cadena** | **caracter** | **id**(parametros)

22. oparit → + | − | ∗ | /
23. variable → dato_est_sim | arreglo
24. dato_est_sim → dato_est_sim **.id** | **id**
25. arreglo → **id** [ expresion ] | arreglo [ expresion ]
26. parametros → lista_param | $\varepsilon$
27. lista_param → lista_param**,** expresion | expresion

## Ejercicios

1. Separar en terminales y no terminales
2. Con base en el conjunto de terminales generar la expresiones regulares para reconocerlos.
3. Con las expresiones regulares hacer un programa en lex que permita retornar solo las clases léxicas como un número entero para cada diferente tipo de token.
4. Además el analizador léxico debe reconocer dos tipos de comentarios
   - De una sola línea que comienza con $--$
   - Multilínea que comienza con $< *$ y termina con $* >$ sin que exista entre esos símbolos un $* >$