Universidad
Nacional Autónoma de México

Facultad de Ciencias

Gramática Proyecto Final

Integrantes:
Yonathan Berith Jaramillo Ramírez. 419004640
Garcia Hernández Jose Antonio. 318337078
Sarah Sophia Olivares García. 318360638

Profesor: Adrián Ulises Mercado Martínez
Ayudantes: Yessica Janeth Pablo Martínez
Carlos Gerardo Acosta Hernández

13 Dec, 2024

Gramática Original

programa → decl_proto decl_var decl_func

decl_proto → proto tipo id (argumentos) ; decl_proto | ϵ

decl_var → tipo lista_var ; decl_var | ϵ

tipo → basico | compuesto | struct { decl_var } | puntero

puntero → ptr basico

basico → int | float | double | complex | rune | void | string

compuesto → [literal_entera] compuesto | ϵ

lista_var → lista_var , id | id

decl_func → func tipo id (argumentos) bloque decl_func | ϵ

argumentos → lista_args | ϵ

lista_args → lista_args , tipo id | tipo id

bloque → { declaraciones instrucciones }

instrucciones → instrucciones sentencia | sentencia

sentencia → parte_izquierda = exp ;
| if (exp) sentencia
| if (exp) sentencia else sentencia
| while (exp) sentencia
| do sentencia while (exp) ;
| break ;
| bloque
| return exp ;
| return ;
| switch (exp) { casos }
| print exp ;
| scan parte_izquierda

casos → caso casos | ϵ | predeterminado

caso → case opcion : instrucciones

opcion → literal_entera | literal_runa

predeterminado → default : instrucciones

parte_izquierda → id localizacion | id

exp → exp operador_bin exp | parte_izquierda | literal

localizacion → arreglo | estructurado

arreglo → arreglo [exp] | [exp]

Tras eliminar la ambigüedad y los factores izquierdos obtuvimos...

Gramática Final

Gramática Ajustada para LR(1)

- **programa** → decl_proto decl_var decl_func
- **decl_proto** → proto tipo id (argumentos) ; decl_proto | ε
- **decl_var** → tipo lista_var ; decl_var | ε
- **decl_func** → func tipo id (argumentos) bloque decl_func | ε
- **tipo** → basico compuesto | struct { decl_var } | puntero
- **puntero** → ptr basico
- **basico** → int | float | double | complex | rune | void | string
- **compuesto** → [literal_entera] compuesto | ε
- **lista_var** → lista_var , id | id
- **argumentos** → lista_args | ε
- **lista_args** → lista_args , tipo id | tipo id
- **bloque** → { declaraciones instrucciones }
- **instrucciones** → instrucciones sentencia | sentencia
- **sentencia** →
  - if (exp) sentencia
  - if (exp) sentencia else sentencia
  - while (exp) sentencia
  - do sentencia while (exp)
  - break ;
  - bloque
  - return exp ;
  - return ;
  - switch (exp) { casos }
  - print exp ;

  - scan parte_izquierda
  - parte_izquierda = exp ;

- **casos** → caso casos | ε | predeterminado
- **caso** → case opcion : instrucciones
- **opcion** → literal_entera | literal_runa
- **predeterminado** → default : instrucciones
- **parte_izquierda** → . id localizacion
- **localizacion** → arreglo | estructurado
- **arreglo** → [exp] arreglo | [exp]
- **estructurado** → . id estructurado | . id
- **exp** → exp || exp_and | exp_and
- **exp_and** → exp_and && exp_or_bit | exp_or_bit
- **exp_or_bit** → exp_or_bit | exp_xor_bit | exp_xor_bit
- **exp_xor_bit** → exp_xor_bit ^ exp_and_bit | exp_and_bit
- **exp_and_bit** → exp_and_bit & exp_igualdad | exp_igualdad
- **exp_igualdad** → exp_igualdad == exp_relacional | exp_igualdad != exp_relacional | exp_relacional
- **exp_relacional** → exp_relacional < exp_shift | exp_relacional > exp_shift | exp_relacional <= exp_shift | exp_relacional >= exp_shift | exp_shift
- **exp_shift** → exp_shift << exp_aditiva | exp_shift >> exp_aditiva | exp_aditiva
- **exp_aditiva** → exp_aditiva + exp_multiplicativa | exp_aditiva - exp_multiplicativa | exp_multiplicativa
- **exp_multiplicativa** → exp_multiplicativa * exp_unaria | exp_multiplicativa / exp_unaria | exp_multiplicativa % exp_unaria | exp_unaria
- **exp_unaria** → - exp_unaria | ! exp_unaria | ~ exp_unaria | exp_primaria
- **exp_primaria** → (exp) | id localizacion | id (parametros) | id | false | true | literal_entera | literal_runa | literal_flotante | literal_doble | literal_compleja | literal_cadena

- **parametros** → lista_param | ε
- **lista_param** → lista_param , exp | exp

Gramática en EBNF

```
programa ::= decl_proto decl_var decl_func

decl_proto ::= ( proto tipo id '(' argumentos ')' ';'
decl_proto ) | ''

decl_var ::= ( tipo lista_var ';' decl_var ) | ''

decl_func ::= ( func tipo id '(' argumentos ')'
bloque decl_func ) | ''

tipo ::= basico | compuesto | 'struct' '{' decl_var '}'
| puntero

puntero ::= 'ptr' basico

basico ::= 'int' | 'float' | 'double' | 'complex' | 'rune'
| 'void' | 'string'

compuesto ::= ( '[' literal_entera ']' compuesto ) | ''

lista_var ::= ( lista_var ',' id ) | id

argumentos ::= lista_args | ''

lista_args ::= ( lista_args ',' tipo id ) | tipo id

bloque ::= '{' declaraciones instrucciones '}'

instrucciones ::= ( instrucciones sentencia ) | sentencia

sentencia ::= 'if' '(' exp ')' sentencia
          | 'if' '(' exp ')' sentencia 'else' sentencia
```

```
            | 'while' '(' exp ')' sentencia
            | 'do' sentencia 'while' '(' exp ')'
            | 'break' ';'
            | bloque
            | 'return' exp ';'
            | 'return' ';'
            | 'switch' '(' exp ')' '{' casos '}'
            | 'print' exp ';'
            | 'scan' parte_izquierda
            | parte_izquierda '=' exp ';'

casos ::= ( caso casos ) | '' | predeterminado

caso ::= 'case' opcion ':' instrucciones

opcion ::= literal_entera | literal_runa

predeterminado ::= 'default' ':' instrucciones

parte_izquierda ::= id localizacion | id

localizacion ::= arreglo | estructurado

arreglo ::= ( arreglo '[' exp ']' ) | ( '[' exp ']' )

estructurado ::= ( estructurado '.' id ) | ( '.' id )

exp ::= exp '||' exp
    | exp '&&' exp
    | exp '|' exp
    | exp '^' exp
    | exp '&' exp
    | exp '==' exp
    | exp '!=' exp
    | exp '<' exp
    | exp '<=' exp
    | exp '>=' exp
```

```
    | exp '>' exp
    | exp '<<' exp
    | exp '>>' exp
    | exp '+' exp
    | exp '-' exp
    | exp '*' exp
    | exp '/' exp
    | exp '%' exp
    | '-' exp
    | '!' exp
    | '~' exp
    | '(' exp ')'
    | id localizacion
    | id '(' parametros ')'
    | id
    | 'false'
    | 'true'
    | literal_entera
    | literal_runa
    | literal_flotante
    | literal_doble
    | literal_compleja
    | literal_cadena

parametros ::= lista_param | ''

lista_param ::= ( lista_param ',' exp ) | exp
```