Fase 0

myR Lang

Diseño de Compiladores

Ernesto Ramón Adame Cendejas A00825923

Profesor: M.C. Elda G. Quiroga, Dr. Héctor Ceballos, PhD

Monterrey Nuevo Leon
Agosto 2023

Objetivo Principal - Categoria

MyR es un

El lenguaje cuenta con la capacidad de realizar:

-

Requerimientos

Tokens

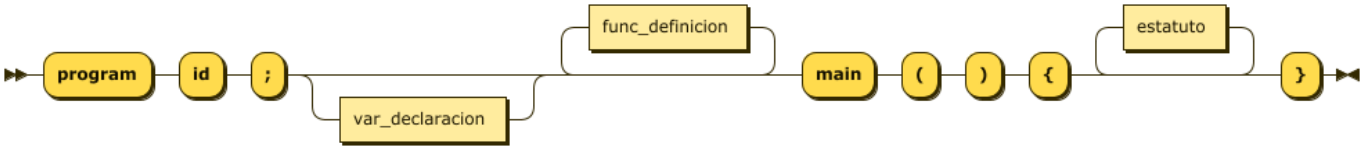
```
#LITERALS
LPAREN : (
RPAREN : )
LBRACK : [
RBRACK : ]
LCURLY : {
RCURLY : }
```

```
GR_THAN  : >
LS_THAN  : <

SQUOTE : '
DQUOTE : "

COLON  : :
SCOLON : ;
DOT    : .
COMMA  : ,

PLUS   : +
MINUS  : -
TIMES  : *
DIVIDE : /
EQUALS : =

AND : &
OR : |
COMMENT : %

#RESERVED
program
main
vars

int
float
char
string

function
return
void

read
write

if
then
else

while
for
do
```

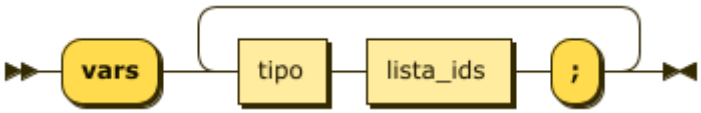
```
#TOKENS
ID
[a-zA-Z][a-zA-Z_0-9]*

VAL_INT
\d+

VAL_FLOAT
\d+\.\d+

VAL_STRING
"[^"]*"

VAL_CHAR
'[a-zA-Z]'

COMP_GR_EQ_THAN # >=
>=

COMP_LS_EQ_THAN # <=
<=

COMP_EQ_TO      # ==
==

COMP_NOT_EQ_TO  # !=
!=
```

Diagramas de sintaxis

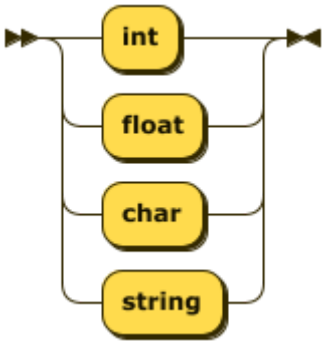
Programa



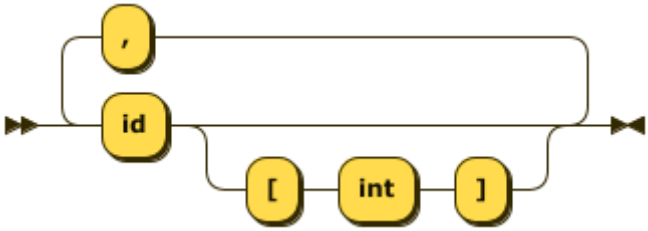
var_declaracion



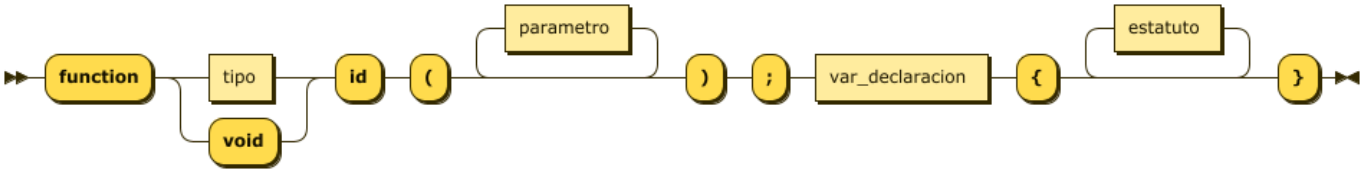
tipo



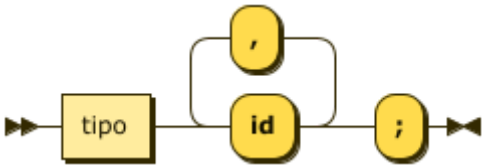
lista_ids



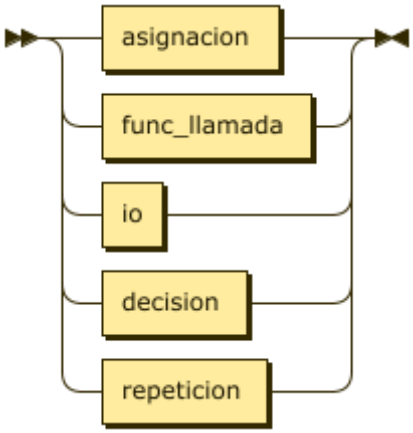
func_definicion



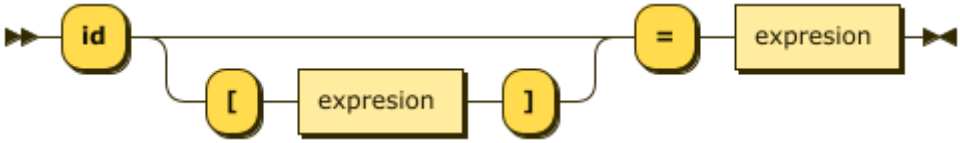
parametro



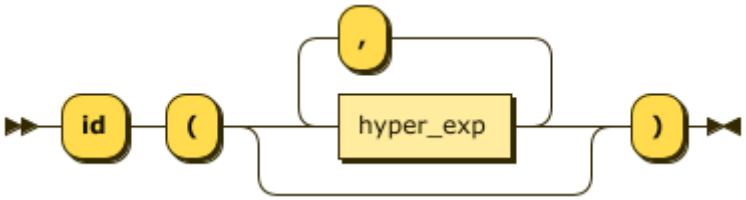
estatuto



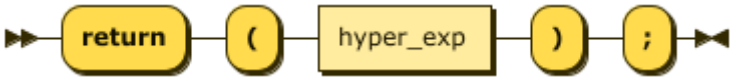
asignacion



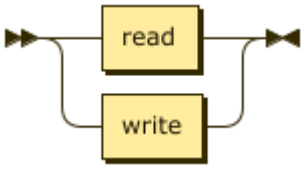
func_llamada



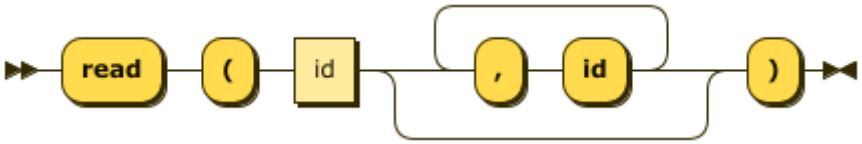
func_retorno



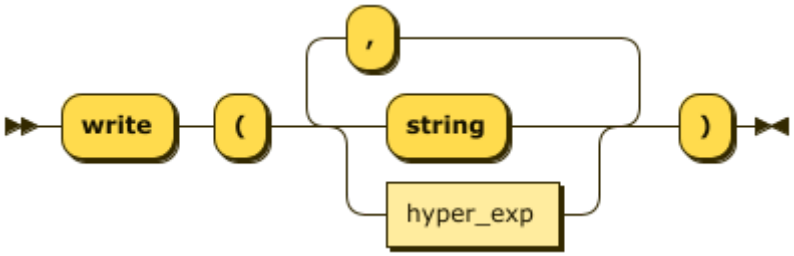
io



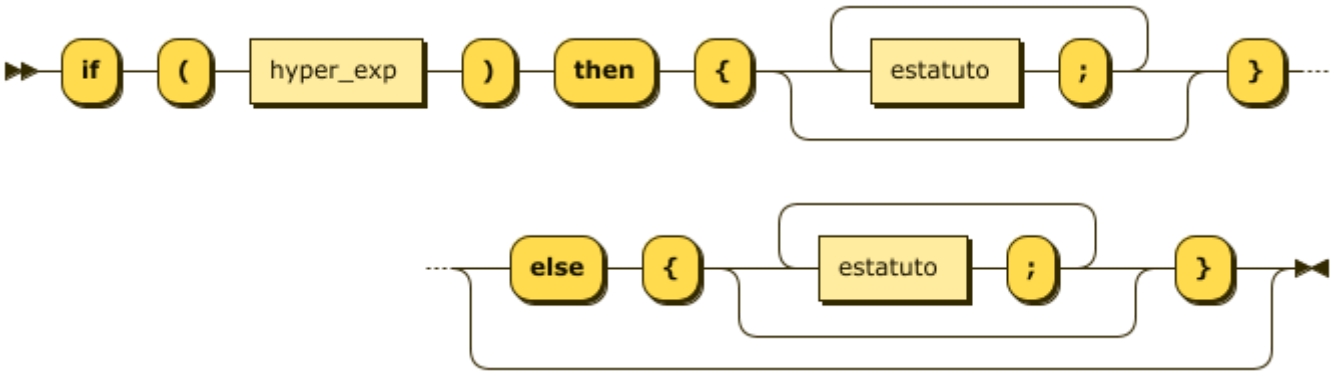
read



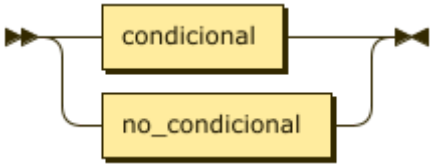
write



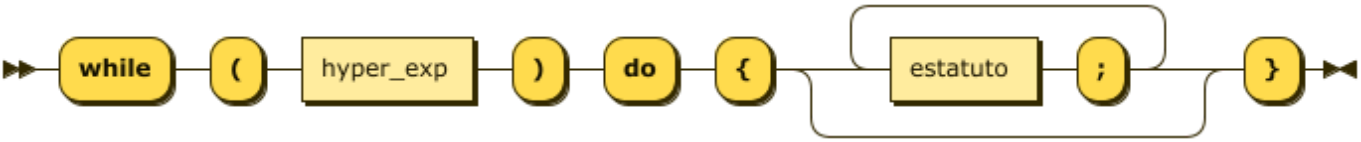
decision



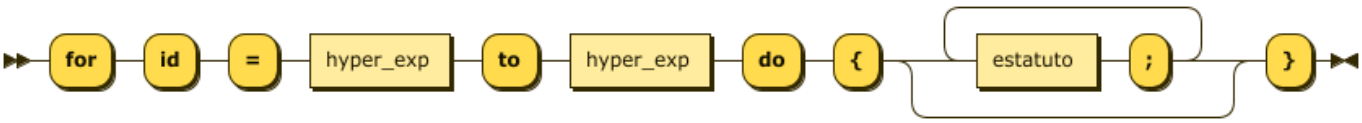
repeticion



condicional



no_condicional



Gramática

```
#todo
programa -> ID ; var_declaracion func definicion MAIN ( ) { estatuto }
                  |

var_declaracion -> VARS tipo lista_ids ; loop_var_decl

loop_vd -> tipo lista_ids ; loop_vd | epsilon

tipo -> INT | FLOAT | CHAR | STRING

lista_ids -> id loop_lista_ids
                   | id [ INT ] loop_lista_ids

loop_lista_ids -> , id loop_lista_ids
                                | , id [ INT ] loop_lista_ids
                                | epsilon

#todo
func_definicion ->

parametro -> tipo ID loop_parametro ;

loop_parametro -> , ID loop_parametro
                                | epsilon

estatuto -> asignacion | func_llamada | io | decision | repeticion

asignacion -> ID = expresion
                    | ID [ expresion ] = expresion
```

```
func_llamada -> ID ( ) ;
              | ID ( hyper_exp loop_func_llamada ) ;

loop_func_llamada -> , hyper_exp loop_func_llamada
                                  |  epsilon

func_retorno -> RETURN ( hyper_exp ) ;

io -> read | write

read -> READ ( ID loop_read ) ;

loop_read -> , ID loop_read
                   | epsilon

write -> ( write_option loop_write ) ;

write_option -> string | hyper_exp

loop_write -> , write_option loop_write
                        | epsilon

decision -> if ( hyper_exp ) then { epsilon }
          | if ( hyper_exp ) then { epsilon }

#todo
loop_decision -> epsilon
                           | estatu
```

Consideraciones Semánticas

Jerarquía de operadores

Nombre	Símbolo	Prioridad
Paréntesis	( )	0
Multiplicación	*	1
División	/	1
Suma	+	2

Nombre	Símbolo	Prioridad
Resta	-	2
Relacionales	< > == !=	3
Lógicos	and or	4
Asignación	=	5

Tipos de Datos

A	B	*	/	+	-	Relational Op	Logical
int	int	int	float	int	int	bool	bool
int	float	float	float	float	float	bool	bool
int	bool	err	err	err	err	err	bool
int	string	err	err	err	err	err	err
float	int	float	float	float	float	bool	bool
float	float	float	float	float	float	bool	bool
float	bool	err	err	err	err	err	bool
float	string	err	err	err	err	err	err
bool	int	err	err	err	err	err	bool
bool	float	err	err	err	err	err	bool
bool	bool	err	err	err	err	err	bool
bool	string	err	err	err	err	err	err
string	int	err	err	err	err	err	err
string	float	err	err	err	err	err	err
string	bool	err	err	err	err	err	err
string	string	err	err	err	err	err	err

Ejemplo

Programa ejemplo

```
```