.+	string
Programa	init_pgrm
;	e_stmt
'.'	char
(	s_par
)	e_par
{	s_bck
}	e_bck
%% \w	—
función	func
principal	main_f
var	var_dec
int | float | char	var_type
[	s_corch
]	e_corch
,	separ
(+|-)?[0-9]+	integer
:	definer
=	eq
int | float | char | void	func_type
[a-zA-Z_$][a-zAZ0-9_$]*	id
regresa	ret
lee	read
escribe	write
"	comillasXD
si	if
entonces	then
'	simple_com

sino	else
mientras	while
haz	do
desde	from
hasta	to
hacer	dof
+|-	op_t2
*|/	op_t1
&||	op_t4
!=|>=|<=|<|>|==	op_t3
(+|-)?[0-9]+\.[0-9]+(e[0-9]+)?	float

```
S → [init_pgrm] [id] [e_stmt] SS
SS → VG FD* M
M → [main_f] [s_par] [e_par] B
B → [s_bck] ST* [e_bck]
VG → [var_dec] TD | ε
TD → [var_type] [definer] TDL1 [e_stmt] TDR
TDL1 → DIMID TDL2
TDL2 → [separ] [id] TDL2 | ε
TDR → FD | ε
FD → [func] FTYPE [id]
      [s_par] PDL1 [e_par] [e_stmt]
      VG B FD | ε
PDL1 → [var_type] [id] PDL2 | ε
PDL2 → [separ] [var_type] [id] PDL2
      | ε
ST → STDEF e_stmt ST | ε
STDEF → ASI | CALL | RET |
      REE | WRT | DEC | REP
CALL → [id] [s_par] CALA [e_par]
CALA → XPO CALA2 | ε
CALA2 → [separ] XPO CALA2 | ε
ASI → DIMID [eq] XPO
RET → [ret] [s_par] XPO [e_par]
REE → [read] [s_par] DIMID REE_ [e_par]
REE_ → [separ] DIMID REE_ | ε
```

XP0 → XP1 [XP0_

XP0_ → [op_t4] XP0 | ε

A||B&&C

XP1 → XP2 XP1_

XP1_ → [op_t3] XP1 | ε

XP2 → XP3 XP2_

XP2_ → [op_t2] XP2 | ε

XP3 → XP4 XP3_

XP3_ → [op_t1] XP3 | ε

XP4 → XPP | DIMID | CALL | [char] | [integer] | [float]

XPP → [s_par] XP0 [e_par]

DIMID → [id] DIMID_

DIMID_ → [s_corch] [XP0] [s_corch] | ε

WRT → [write] [s_par] WL [e_par]

W_C → STR | XP0

STR → [comillasXD] [string] [comillasXD]

WL → W_C WL1

WL1 → [separ] W_C WL1 | ε

DEC → if [s_par] XP0 [e_par] [then] B DEC_

DEC_ → [else] B | ε

REP → COND | NCOND

COND → [while] [s_par] XP0 [e_par] [do] B

NCOND → [from] ASI [to] XP0 [dof] B

FTYPE → [var_type] | [void]