

Procesadores de lenguajes

Memoria

G15

Verdaguer Velázquez, Miguel

Rubio Pastor, Cesar Carlos

Antuña Rodríguez , Alejandro

Makitu Koudymba , Fumu Grace

Índice

<u>1 Tiny (0)</u>	<u>2</u>
<u>1.1 Descripción informal</u>	<u>2</u>
Definiciones auxiliares:	2
Clases léxicas:	2
Cadenas ignorables:	3
<u>1.2 Especificación formal del léxico del lenguaje</u>	<u>3</u>
Clases auxiliares:	3
Clases léxicas:	3
Cadenas ignorables:	4
<u>1.3 Diagrama de transiciones</u>	<u>4</u>
<u>1.4 Gramática</u>	<u>4</u>
<u>1.4 Acondicionamiento de la gramática</u>	<u>6</u>
<u>1.5 Directores</u>	<u>8</u>
<u>2. Tiny</u>	<u>9</u>
<u>2.1 Descripción informal</u>	<u>9</u>
Definiciones auxiliares:	9
Clases léxicas:	9
Cadenas ignorables:	11
<u>2.2 Especificación formal del léxico del lenguaje</u>	<u>11</u>
Definiciones auxiliares:	11
Clases léxicas:	11
Cadenas ignorables:	12
<u>2.3 Gramática</u>	<u>12</u>
<u>2.4 Acondicionamiento de la gramática</u>	<u>15</u>

1 Tiny (0)

1.1 Descripción informal

U->Clase Univaluada

M->clase Multivaluada

Definiciones auxiliares:

digPositivo(M) = Un único dígito entre el 1 y el 9

digito(M) = Un único dígito entre el 0 y el 9

parteEntera(M) = Una serie de dígitos sin ceros al principio

parteDecimal(M) = Un cero seguido de una serie de dígitos sin ceros al final

parteExponencial(M) = Una e seguida de una serie de dígitos como un literal entero

segParteReal(M) = La segunda parte de un literal real, bien una parte decimal, bien una parte exponencial o bien una parte decimal seguida de una exponencial

letra(M) = Una letra entre la a y la z, mayúscula o minúscula

Clases léxicas:

int(U) = Palabra reservada para indicar el tipo int

real(U) = Palabra reservada para indicar el tipo real

bool(U) = Palabra reservada para indicar el tipo bool

and(U) = Palabra reservada como operador and

or(U) = Palabra reservada como operador or

not(U) = Palabra reservada como operador not

TRUE(U) = Palabra reservada para el valor de bool verdadero

FALSE(U) = Palabra reservada para el valor de bool falso

suma(U) = Operador suma

resta(U) = Operador resta

mult(U) = Operador multiplicación

div(U) = Operador divis ión

menorque(U) = Operador menor-que

mayorque(U) = Operador mayor-que

menorigual(U) = Operador menor o igual

mayorigual(U) = Operador mayor o igual
 igual_comp(U) = Operador igualdad
 noigual_comp(U) = Operador no igualdad
 par_a(U) = Apertura de paréntesis
 par_c(U) = Cierre de paréntesis
 puntoycoma(U) = Punto y coma, final de instrucción
 igual_op(U) = Operador de asignación de valores
 llave_a(U) = Apertura de llave
 llave_c(U) = Cierre de llave
 dobleat(U) = Fin de declaraciones
 arroba(U) = Comienzo de la instrucción de evaluación
 litEntero(M) = Un literal entero, puede tener un + o - al principio y
 sin ceros no significativos
 litReal(M) = Un literal real, separando con punto la parte decimal y
 entera y sin ceros no significativos
 caracter(M) = Un carácter, letra, dígito o subrayado
 iden(M) = Una secuencia de caracteres que comienza por una letra o
 subrayado

Cadenas ignorables:

espacio(M) = Cualquier tipo de espaciado
 comentario(M) = Un comentario a ignorar. Comienza con ## y puede
 contener cualquier carácter excepto el salto
 de línea

1.2 Especificación formal del léxico del lenguaje

I -> Case insensitive

Clases auxiliares:

digPositivo = [1-9]
 digito = {digPositivo}|0
 parteEntera = ({digPositivo}{digito}*|0)
 parteDecimal = \.({digito}*{digitoPositivo}|0)
 parteExponencial = (e|E){litEntero}
 segParteReal = {parteDecimal}|{parteExponencial}|{parteDecimal}
 {parteExponencial}
 letra = [a-z,A-Z]

Clases léxicas:

int = [l,l][n,N][t,T]
 real = [r,R][e,E][a,A][l,L]
 bool = [b,B][o,O][o,O][l,L]
 and = [a,A][n,N][d,D]
 or = [o,O][r,R]
 not = [n,N][o,O][t,T]
 null = [n,N][u,U][l,L][l,L]
 TRUE = [t,T][r,R][u,U][e,E]
 FALSE = [f,F][a,A][l,L][s,S][e,E]
 suma = \+
 resta = \-
 mult = *
 div = /
 menorque = <

Declaraciones_opt -> ϵ
 Instrucciones_opt -> Instrucciones
 Instrucciones_opt -> ϵ
 Declaraciones -> Lista_declaraciones &&
 Lista_declaraciones -> Declaracion
 Lista_declaraciones -> Lista_declaraciones ; Declaracion
 Declaracion -> Tipo Identificador
 Tipo -> **int**
 Tipo -> **real**
 Tipo -> **bool**

Instrucciones -> Instruccion
 Instrucciones -> Instrucciones ; Instruccion
 Instruccion -> @ Expresion
 Expresion_basica -> Lit_entero
 Expresion_basica -> Lit_real
 Expresion_basica -> Lit_booleano
 Expresion_basica -> Identificador
 Expresion -> E0
 E0 -> E1
 E0 -> E1 Op0 E0
 E1 -> E2
 E1 -> E1 Op1 E2
 E2 -> E3
 E2 -> E2 Op2_izq E3
 E2 -> E3 Op2_noasoc E3
 E3 -> E4
 E3 -> E4 Op3_der E3
 E3 -> E4 Op3_noasoc E4
 E4 -> E5
 E4 -> E4 Op4 E5

E5 -> E6
 E5 -> Op5 E5
 E6 -> (E0)
 E6 -> Expresion_basica
 Op0 -> =
 Op1 -> ==
 Op1 -> !=
 Op1 -> <
 Op1 -> >
 Op1 -> <=
 Op1 -> >=
 Op2_izq -> +
 Op2_noasoc -> -
 Op3_der -> and
 Op3_noasoc -> or
 Op4 -> *
 Op4 -> /
 Op5 -> -
 Op5 -> not

1.4 Acondicionamiento de la gramática

Gramática original	Gramática acondicionada
-Lista_declaraciones -> Declaración -Lista_declaraciones -> Lista_declaraciones ; Declaración	-Lista_declaraciones -> Declaración R_lista_declaraciones -R_lista_declaraciones -> ; Declaración R_lista_declaraciones -R_lista_declaraciones -> ε
-Instrucciones -> Instrucción -Instrucciones -> Instrucciones ; Instrucción	-Instrucciones -> Instrucción R_instrucciones -R_instrucciones -> ; Instrucción R_instrucciones -R_instrucciones -> ε
-E0 -> E1 -E0 -> E1 Op0 E0	E0 -> E1 R0 R0 -> Op0 E0

	R0 -> ϵ
E1 -> E2 E1 -> E1 Op1 E2	E1 -> E2 R1 R1 -> Op1 E2 R1 R1 -> ϵ
E2 -> E3 E2 -> E2 Op2_izq E3 E2 -> E3 Op2_noasoc E3	E2 -> E3 R2 R21 R2 -> Op2_noasoc E3 ϵ R21 -> Op2_izq E3 R21 ϵ
E3 -> E4 E3 -> E4 Op3_der E3 E3 -> E4 Op3_noasoc E4	E3 -> E4 R3 R3 -> Op3_der E3 Op3_noasoc E4 R3 -> ϵ
E4 -> E5 E4 -> E4 Op4 E5	E4 -> E5 R4 R4 -> Op4 E5 R4 R4 -> ϵ

Programa -> Bloque

Bloque -> { Declaraciones_opt Instrucciones_opt }

Declaraciones_opt -> Declaraciones

Declaraciones_opt -> ϵ

Instrucciones_opt -> Instrucciones

Instrucciones_opt -> ϵ

Declaraciones -> Lista_declaraciones &&

Lista_declaraciones -> Declaracion R_lista_declaraciones

R_lista_declaraciones -> ; Declaracion R_lista_declaraciones

R_lista_declaraciones -> ϵ

Declaracion -> Tipo Identificador

Tipo -> int

Tipo -> real

Tipo -> bool

Instrucciones -> Instruccion R_instrucciones

$R_instrucciones \rightarrow ; Instruccion R_instrucciones$

$R_instrucciones \rightarrow \epsilon$

$Instruccion \rightarrow @ Expresion$

$Expresion \rightarrow E0$

$E0 \rightarrow E1 R0$

$R0 \rightarrow Op0 E0$

$R0 \rightarrow \epsilon$

$E1 \rightarrow E2 R1$

$R1 \rightarrow Op1 E2 R1$

$R1 \rightarrow \epsilon$

$E2 \rightarrow E3 R2$

$E2 \rightarrow E3 R2 R21$

$R2 \rightarrow Op2_noasoc E3$

$R2 \rightarrow \epsilon$

$R21 \rightarrow Op2_izq E3 R21$

$R21 \rightarrow \epsilon$

$E3 \rightarrow E4 R3$

$R3 \rightarrow Op3_der E3$

$R3 \rightarrow Op3_noasoc E4$

$R3 \rightarrow \epsilon$

$E4 \rightarrow E5 R4$

$R4 \rightarrow Op4 E5 R4$

$R4 \rightarrow \epsilon$

$E5 \rightarrow E6$

$E5 \rightarrow Op5 E5$

$E6 \rightarrow (E0)$

$E6 \rightarrow Expresion_basica$

$Expresion_basica \rightarrow Lit_entero$

$Expresion_basica \rightarrow Lit_real$

$Expresion_basica \rightarrow Lit_booleano$

$Expresion_basica \rightarrow Identificador$

Op0 -> =
 Op1 -> ==
 Op1 -> !=
 Op1 -> <
 Op1 -> >
 Op1 -> <=
 Op1 -> >=
 Op2_izq -> +
 Op2_noasoc -> -
 Op3_der -> **and**
 Op3_noasoc -> **or**
 Op4 -> *
 Op4 -> /
 Op5 -> -
 Op5 -> **not**

1.5 Directores

Dir(Programa) = { /{ }

Dir(Bloque) = { /{ }

Dir(Declaraciones_opt) = { ∅ }

Dir(Instrucciones_opt) = { ∅ }

Dir(Declaraciones) = {int, real, bool}

Dir(Lista_declaraciones) = {int, real, bool}

Dir(R_lista_declaraciones) = { ∅ }

Dir(Declaracion) = {int, real, bool}

Dir(Tipo) = {int, real, bool}

Dir(Instrucciones) = {@}

Dir(R_instrucciones) = { ∅ }

Dir(Instruccion) = {@}

$\text{Dir}(\text{Expresion}) = \{ \backslash(, \text{Lit_entero}, \text{Lit_real}, \text{Lit_booleano}, \text{Identificador}, \backslash-, \text{not} \}$

$\text{Dir}(\text{Expresion_basica}) = \{ \text{Lit_entero}, \text{Lit_real}, \text{Lit_booleano}, \text{Identificador} \}$

$\text{Dir}(\text{Expresion_compuesta}) = \{ \backslash(, \text{Lit_entero}, \text{Lit_real}, \text{Lit_booleano}, \text{Identificador}, \backslash-, \text{not} \}$

$\text{Dir}(E0) = \{ \backslash(, \text{Lit_entero}, \text{Lit_real}, \text{Lit_booleano}, \text{Identificador}, \backslash-, \text{not} \}$

$\text{Dir}(R0) = \{ \emptyset \}$

$\text{Dir}(E1) = \{ \backslash(, \text{Lit_entero}, \text{Lit_real}, \text{Lit_booleano}, \text{Identificador}, \backslash-, \text{not} \}$

$\text{Dir}(R1) = \{ \emptyset \}$

$\text{Dir}(E2) = \{ \backslash(, \text{Lit_entero}, \text{Lit_real}, \text{Lit_booleano}, \text{Identificador}, \backslash-, \text{not} \}$

$\text{Dir}(R2) = \{ \backslash- \}$

$\text{Dir}(E3) = \{ \backslash(, \text{Lit_entero}, \text{Lit_real}, \text{Lit_booleano}, \text{Identificador}, \backslash-, \text{not} \}$

$\text{Dir}(R3) = \{ \text{and}, \text{or} \}$

$\text{Dir}(E4) = \{ \backslash(, \text{Lit_entero}, \text{Lit_real}, \text{Lit_booleano}, \text{Identificador}, \backslash-, \text{not} \}$

$\text{Dir}(R4) = \{ \emptyset \}$

$\text{Dir}(E5) = \{ \backslash(, \text{Lit_entero}, \text{Lit_real}, \text{Lit_booleano}, \text{Identificador}, \backslash-, \text{not} \}$

$\text{Dir}(E6) = \{ \backslash(, \text{Lit_entero}, \text{Lit_real}, \text{Lit_booleano}, \text{Identificador} \}$

$\text{Dir}(Op0) = \{ \backslash= \}$

$\text{Dir}(Op1) = \{ ==, !=, <, >, <=, >= \}$

$\text{Dir}(Op2_izq) = \{ \backslash+ \}$

$\text{Dir}(Op2_noasoc) = \{ \backslash- \}$

$\text{Dir}(Op3_der) = \{ \text{and} \}$

$\text{Dir}(Op3_noasoc) = \{ \text{or} \}$

$\text{Dir}(Op4) = \{ \backslash*, \backslash/ \}$

$\text{Dir}(Op5) = \{ \backslash-, \text{not} \}$

2. Tiny

2.1 Descripción informal

Definiciones auxiliares:

digPositivo(M) = Un único dígito entre el 1 y el 9
digito(M) = Un único dígito entre el 0 y el 9
parteEntera(M) = Una serie de dígitos sin ceros al principio
parteDecimal(M) = Un cero seguido de una serie de dígitos sin ceros al final
parteExponencial(M) = Una e seguida de una serie de dígitos como un literal entero
segParteReal(M) = La segunda parte de un literal real, bien una parte
letra(M) = Una letra entre la a y la z, mayúscula o minúscula

Clases léxicas:

int(U) = Palabra reservada para indicar el tipo int
real(U) = Palabra reservada para indicar el tipo real
bool(U) = Palabra reservada para indicar el tipo bool
string(U) = Palabra reservada para indicar el tipo string
and(U) = Palabra reservada como operador and
or(U) = Palabra reservada como operador or
not(U) = Palabra reservada como operador not
null(U) = Palabra reservada para la expresión básica vacía
TRUE(U) = Palabra reservada para el valor de bool verdadero
FALSE(U) = Palabra reservada para el valor de bool falso
proc(U) = Palabra reservada para declarar un procedimiento
if(U) = Palabra reservada para indicar el comienzo de una instrucción if
else(U) = Palabra reservada para indicar la segunda parte de una instrucción if-else
while(U) = Palabra reservada para indicar el comienzo de una instrucción while
struct(U) = Palabra reservada para declarar tipos como estructuras
new(U) = Palabra reservada para la instrucción de reserva de memoria
delete(U) = Palabra reservada para la instrucción de liberación de memoria
read(U) = Palabra reservada para la instrucción de lectura de memoria
write(U) = Palabra reservada para la instrucción de escritura en memoria
nl(U) = Palabra reservada para la instrucción de nueva línea
type(U) = Palabra reservada para las declaraciones de tipo
call(U) = Palabra reservada para las llamadas a procedimientos
suma(U) = Operador suma
resta(U) = Operador resta
mult(U) = Operador multiplicación
div(U) = Operador división
porcentaje(U) = Operador módulo entero
menorque(U) = Operador menor-que
mayorque(U) = Operador mayor-que
menorigual(U) = Operador menor o igual
mayorigual(U) = Operador mayor o igual
igual_comp(U) = Operador igualdad

noigual_comp(U) = Operador no igualdad
 par_a(U) = Apertura de paréntesis
 par_c(U) = Cierre de paréntesis
 puntoycoma(U) = Punto y coma, final de instrucción
 igual_op(U) = Operador de asignación de valores
 corch_a(U) = Apertura de corchetes
 corch_c(U) = Cierre de corchetes
 punto(U) = Punto, operador de acceso a registro
 capirote(U) = Operador de indirección
 coma(U) = Coma, separador de listas
 llave_a(U) = Apertura de llave
 llave_c(U) = Cierre de llave
 at(U) = Operador de referencia
 dobleat(U) = Fin de declaraciones
 arroba(U) = Comienzo de la instrucción de evaluación
 litEntero(M) = Un literal entero, puede tener un + o - al principio y sin ceros no significativos
 litReal(M) = Un literal real, separando con punto la parte decimal y entera y sin ceros no significativos
 caracter(M) = Un carácter, letra, dígito o subrayado

 iden(M) = Una secuencia de caracteres que comienza por una letra o subrayado
 litCadena(M) = Un literal de tipo cadena, comienza por una doble comilla y continúa con una secuencia de 0 o más caracteres, terminado en otra doble comilla

Cadenas ignorables:

espacio(M) = Cualquier tipo de espaciado
 comentario(M) = Un comentario a ignorar. Comienza con ## y puede contener cualquier carácter excepto el salto de línea

2.2 Especificación formal del léxico del lenguaje

I -> Case insensitive

Definiciones auxiliares:

digPositivo = [1-9]
 digito = {digPositivo}|0
 parteEntera = ({digPositivo}{digito}*|0) letra = [a-z,A-Z]
 parteDecimal = \.({digito}*{digitoPositivo}|0)
 segParteReal = {parteDecimal}|{parteExponencial}|{parteDecimal} { parteExponencial}
 parteExponencial = (e|E){litEntero}
 letra = [a-z,A-Z]

Clases léxicas:

int (I) = [i,I][n,N][t,T]
 real (I) = [r,R][e,E][a,A][l,L]
 bool (I) = [b,B][o,O][o,O][l,L]

```

string (l) = [s,S][t,T][r,R][i,I][n,N][g,G]
and (l) = [a,A][n,N][d,D]
or (l) = [o,O][r,R]
not (l) = [n,N][o,O][t,T]
null (l) = [n,N][u,U][l,L][l,L]
TRUE (l) = [t,T][r,R][u,U][e,E]
FALSE (l) = [f,F][a,A][l,L][s,S][e,E]
proc (l) = [p,P][r,R][o,O][c,C]
if (l) = [i,I][f,F]
else (l) = [e,E][l,L][s,S][e,E]
while (l) = [w,W][h,H][i,I][l,L][e,E]
struct (l) = [s,S][t,T][r,R][u,U][c,C][t,T]
new (l) = [n,N][e,E][w,W]
delete (l) = [d,D][e,E][l,L][e,E][t,T][e,E]
read (l) = [r,R][e,E][a,A][d,D]
write (l) = [w,W][r,R][i,I][t,T][e,E]
nl (l) = [n,N][l,L]
type (l) = [t,T][y,Y][p,P][e,E]
call (l) = [c,C][a,A][l,L][l,L]
suma = \+
resta = \-
mult = \*
div = /
porcentaje = %
menorque = <
mayorque = >
menorigual = <=
mayorigual = >=
igual_comp = ==
noigual_comp = !=
par_a = \ (
par_c = \ )
puntoycoma = ;
igual_op = =
corch_a = \ [
corch_c = \ ]
punto = \ .
capirote = \ ^
coma = \ ,
llave_a = \ {
llave_c = \ }
at = &
dobleat = &&
arroba = @

litEntero = ([\+, \-])? {parteEntera}
litReal = {litEntero} {segParteReal}
caracter = {letra} | {digito} | _
iden = ({letra} | _){caracter}*
litCadena = "([^\EOF])*"

```

Cadenas ignorables:

```

espacio = [\b, \r, \t, \n]

```

comentario = `##([^\n,EOF])*`

2.3 Gramática

Programa -> Bloque

Bloque -> { Declaraciones_opt Instrucciones_opt }

Declaraciones_opt -> Declaraciones | ϵ

Instrucciones_opt -> Instrucciones | ϵ

Declaraciones -> Lista_declaraciones &&

Lista_declaraciones -> Declaracion

Lista_declaraciones -> Lista_declaraciones ; Declaracion

Declaracion -> Dec_variable | Dec_tipo | Dec_procedimiento

Dec_variable -> Tipo Identificador

Dec_tipo -> **type** Tipo Identificador

Dec_procedimiento -> **proc** Identificador Parametros_form Bloque

Parametros_form -> (Parametros_form_lista_opt)

Parametros_form_lista_opt -> Parametros_form_lista | ϵ

Parametros_form_lista -> Parametro_form

Parametros_form_lista -> Parametros_form_lista , Parametro_form

Parametro_form -> Tipo Ampersand_opt Identificador

Ampersand_opt -> & | ϵ

Tipo -> Tipo0

Tipo0 -> Tipo0 [Lit_entero] | Tipo1

Tipo1 -> Tipo2 | ^ Tipo1

Tipo2 -> **int** | **real** | **bool** | **string** | **struct** Campos | Identificador

Campos -> { Campos_lista }

Campos_lista -> Campo

Campos_lista -> Campos_lista , Campo

Campo -> Tipo Identificador

Instrucciones -> Instruccion

Instrucciones -> Instrucciones ; Instruccion

Instruccion -> Instr_eval | Instr_if | Instr_if_else | Instr_while |
Instr_read | Instr_write | Instr_nl | Instr_reserva | Instr_liberacion |
Instr_invocacion | Instr_compuesta

Instr_eval -> @ Expresion

Instr_if -> **if** Expresion Bloque

Instr_if_else -> **if** Expresion Bloque **else** Bloque

Instr_while -> **while** Expresion Bloque

Instr_read -> **read** Expresion

Instr_write -> **write** Expresion

Instr_nl -> **nl**

Instr_reserva -> **new** Expresion

Instr_liberacion -> **delete** Expresion

Instr_invocacion -> **call** Identificador Parametros_reales

Parametros_reales -> (Parametros_reales_lista_opt)

Parametros_reales_lista_opt -> Parametros_reales_lista | ϵ

Parametros_reales_lista -> Expresion

Parametros_reales_lista -> Parametros_reales_lista , Expresion

Instr_compuesta -> Bloque

Expresion_basica -> Lit_entero | Lit_real | Lit_booleano | Lit_cadena |
Identificador | **null**

Expresion -> E0

E0 -> E1

E0 -> E1 Op0 E0

E1 -> E2

E1 -> E1 Op1 E2

E2 -> E3

E2 -> E2 Op2_izq E3

E2 -> E3 Op2_noasoc E3

E3 -> E4
 E3 -> E4 Op3_der E3
 E3 -> E4 Op3_noasoc E4
 E4 -> E5
 E4 -> E4 Op4 E5
 E5 -> E6
 E5 -> Op5 E5
 E6 -> E7
 E6 -> E6 Op6
 E7 -> (E0)
 E7 -> Expresion_basica
 Op0 -> =
 Op1 -> == | != | < | > | <= | >=
 Op2_izq -> +
 Op2_noasoc -> -
 Op3_der -> and
 Op3_noasoc -> or
 Op4 -> * | / | %
 Op5 -> - | not
 Op6 -> Op_indexacion | Op_acceso_registro | Op_indireccion
 Op_indexacion -> [Expresion]
 Op_acceso_registro -> . Identificador
 Op_indireccion -> ^

2.4 Acondicionamiento de la gramática

Gramática original	Gramática acondicionada
-Lista_declaraciones -> Declaración -Lista_declaraciones -> Lista_declaraciones ; Declaración	-Lista_declaraciones -> Declaración R_lista_declaraciones -R_lista_declaraciones -> ; Declaración R_lista_declaraciones -R_lista_declaraciones -> ε

-Parametros_form_lista -> Parametro_form -Parametros_form_lista -> Parametros_form_lista , Parametro_form	-Parametros_form_lista -> Parametro_form R_parametros_form_lista -R_parametros_form_lista -> , Parametro_form R_parametros_form_lista -R_parametros_form_lista -> ϵ
Tipo -> Tipo0 Tipo0 -> Tipo0 [Lit_entero] Tipo1	Tipo0 -> Tipo1 RT0 RT0 -> [Lit_entero] RT0 ϵ
-Campos_lista -> Campo -Campos_lista -> Campos_lista , Campo	-Campos_lista -> Campo R_campos_lista -R_campos_lista -> , Campo R_campos_lista -R_campos_lista -> ϵ
-Instrucciones -> Instruccion -Instrucciones -> Instrucciones ; Instruccion	-Instrucciones -> Instruccion R_instrucciones -R_instrucciones -> ; Instruccion R_instrucciones -R_instrucciones -> ϵ
-Instr_if -> if Expresion Bloque -Instr_if_else -> if Expresion Bloque else Bloque	-Instr_if_else -> if Expresion Bloque Else_opt -Else_opt -> else Bloque -Else_opt -> ϵ
-Parametros_reales_lista -> Expresion -Parametros_reales_lista -> Parametros_reales_lista , Expresion	-Parametros_reales_lista -> Expresion R_parametros_reales_lista -R_parametros_reales_lista -> , Expresion R_parametros_reales_lista ϵ
-E0 -> E1 -E0 -> E1 Op0 E0	E0 -> E1 R0 R0 -> Op0 E0 R0 -> ϵ
E1 -> E2 E1 -> E1 Op1 E2	E1 -> E2 R1 R1 -> Op1 E2 R1 R1 -> ϵ
E2 -> E3 E2 -> E2 Op2_izq E3 E2 -> E3 Op2_noasoc E3	E2 -> E3 R2 R21 R2 -> Op2_noasoc E3 R2 -> ϵ R21 -> Op2_izq E3 R21 R21 -> ϵ
E3 -> E4	E3 -> E4 R3

E3 -> E4 Op3_der E3 E3 -> E4 Op3_noasoc E4	R3 -> Op3_der E3 Op3_noasoc E4 ϵ
E4 -> E5 E4 -> E4 Op4 E5	E4 -> E5 R4 R4 -> Op4 E5 R4 R4 -> ϵ
E6 -> E7 E6 -> E6 Op6	E6 -> E7 R6 R6 -> Op6 R6 ϵ

Programa -> Bloque

Bloque -> { Declaraciones_opt Instrucciones_opt }

Declaraciones_opt -> Declaraciones

Declaraciones_opt -> ϵ

Instrucciones_opt -> Instrucciones

Instrucciones_opt -> ϵ

Declaraciones -> Lista_declaraciones &&

Lista_declaraciones -> Declaracion R_lista_declaraciones

R_lista_declaraciones -> ; Declaracion R_lista_declaraciones

R_lista_declaraciones -> ϵ

Declaracion -> Dec_variable

Declaracion -> Dec_tipo

Declaracion -> Dec_procedimiento

Dec_variable -> Tipo Identificador

Dec_tipo -> **type** Tipo Identificador

Dec_procedimiento -> **proc** Identificador Parametros_form Bloque

Parametros_form -> (Parametros_form_lista_opt)

Parametros_form_lista_opt -> Parametros_form_lista

Parametros_form_lista_opt -> ϵ

Parametros_form_lista -> Parametro_form R_parametros_form_lista

R_parametros_form_lista -> , Parametro_form R_parametros_form_lista

R_parametros_form_lista -> ϵ

Parametro_form -> Tipo Ampersand_opt Identificador

Ampersand_opt -> &

Ampersand_opt -> ϵ

Tipo -> Tipo0

Tipo0 -> Tipo1 RT0

RT0 -> [Lit_entero] RT0

RT0 -> ϵ

Tipo1 -> Tipo2

Tipo1 -> ^ Tipo1

Tipo2 -> **int**

Tipo2 -> **real**

Tipo2 -> **bool**

Tipo2 -> **string**

Tipo2 -> **struct** Campos

Tipo2 -> Identificador

Campos -> { Campos_list }

Campos_lista -> Campo R_campos_lista

R_campos_lista -> , Campo R_campos_lista

R_campos_lista -> ϵ

Campo -> Tipo Identificador

Instrucciones -> Instruccion R_instrucciones

R_instrucciones -> ; Instruccion R_instrucciones

R_instrucciones -> ϵ

Instruccion -> Instr_eval

Instruccion -> Instr_if_else

Instruccion -> Instr_while

Instruccion -> Instr_read

Instruccion -> Instr_write

Instruccion -> Instr_nl

Instruccion -> Instr_reserva

Instruccion -> Instr_liberacion

Instruccion -> Instr_invocacion

Instruccion -> Instr_compuesta

Instr_eval -> @ Expresion

Instr_if_else -> if Expresion Bloque Else_opt

Else_opt -> else Bloque

Else_opt -> ϵ

Instr_while -> while Expresion Bloque

Instr_read -> read Expresion

Instr_write -> write Expresion

Instr_nl -> nl

Instr_reserva -> new Expresion

Instr_liberacion -> delete Expresion

Instr_invocacion -> call Identificador Parametros_reales

Parametros_reales -> (Parametros_reales_lista_opt)

Parametros_reales_lista_opt -> Parametros_reales_lista

Parametros_reales_lista_opt -> ϵ

Parametros_reales_lista -> Expresion R_parametros_reales_lista

R_parametros_reales_lista -> , Expresion R_parametros_reales_lista

R_parametros_reales_lista -> ϵ

Instr_compuesta -> Bloque

Expresion -> E0

E0 -> E1 R0

R0 -> Op0 E0

R0 -> ϵ

E1 -> E2 R1

R1 -> Op1 E2 R1

R1 -> ϵ

E2 -> E3 R2 R21

R2 -> Op2_noasoc E3

R2 -> ϵ

R21 -> Op2_izq E3 R21

R21 -> ϵ

E3 -> E4 R3

R3 -> Op3_der E3

R3 -> Op3_noasoc E4

R3 -> ϵ

E4 -> E5 R4

R4 -> Op4 E5 R4

R4 -> ϵ

E5 -> E6

E5 -> Op5 E5

E6 -> E7 R6

R6 -> Op6 R6

R6 -> ϵ

E7 -> (E0)

E7 -> Lit_entero

E7 -> Lit_real

E7 -> Lit_booleano

E7 -> Lit_cadena

E7 -> Identificador

E7 -> null

Op0 -> =

Op1 -> ==

Op1 -> !=

Op1 -> <

Op1 -> >

Op1 -> <=

Op1 -> >=

Op2_izq -> +

Op2_noasoc -> -

Op3_der -> **and**

Op3_noasoc -> **or**

Op4 -> *

Op4 -> /

Op4 -> %

Op5 -> -

Op5 -> **not**

Op6 -> Op_indexacion

Op6 -> Op_acceso_registro

Op6 -> Op_indireccion

Op_indexacion -> [Expresion]

Op_acceso_registro -> . Identificador

Op_indireccion -> ^