ANALIZADOR SINTÁCTICO TINY(1)

Óscar Morujo Fernández

Sofía Capmany Fernández

G16

ESPECIFICACIÓN SINTÁCTICA

No terminales :

PROGRAMA, LDECS, RDECS, DECS, DEC, DEC_VAR, DEC_TIPO, DEC_PROC, PARAMS, LPARAMS, PARAM, TIPO, TIPO BASICO, TIPO_ARRAY, TIPO_REG, TIPO_PUNT, CAMPOS, CAMPO, LINST, INST, INST_ASIG, E0, E1, E2, E3, E4, E5, E6, E7, OP1AI, OP2AI, OP3NA, OP5A, INST_IF_THEN, INST_IF_THEN_ELSE, INST_WHILE, AUX_LINST, INST_READ, INST_WRITE, INST_NL, INST_NEW, INST_DELETE, INST_CALL, REAL_PARAMS, INST_COMPUESTA, BLOQUE.

Terminales :

Terminal	Valor	Terminal	Valor	Terminal	Valor
sep_di	“&&”	**bne**	“<=”	**type**	“type”
semicolon	“;”	**beq**	“==”	**coma**	“,”
id	identificador	**ble**	“<=”	**amp**	“&”
r_real	“real”	**bge**	“>=”	**array**	“array”
r_int	“int”	**bgt**	“>”	**cap**	“[“
r_bool	“bool”	**blt**	“<”	**ccierre**	“]”
r_string	“string”	**var**	“var”	**of**	“of”
igual	“=”	**true**	“true”	**record**	“record”
mas	“+”	**false**	“false”	**llap**	“{“
menos	“-”	**pcierre**	“)”	**llcierre**	“}”
por	“*”	**pap**	“(“	**pointer**	“pointer”
div	“/”	**real**	número real	**while**	“while”
and	“and”	**ent**	número entero	**do**	“do”
or	“or”	**not**	“not”	**endwhile**	“endwhile”
if	“if”	**endif**	“endif”	**read**	“read”
then	“then”	**else**	“else”	**write**	“write”
nl	“nl”	**new**	“new”	**delete**	“delete”
call	“call”	**cadena**	cadena de	**null**	“null”

			caracteres		
mod	“%”	**punto**	“.”	**flecha**	“->”

Gramática:

PROGRAMA := LDECS LINST;
LDECS := DECS **sep_di**;
LDECS := **ƛ** ;
DECS := DECS **scol** DEC;
DECS := DEC;
DEC := DEC_VAR | DEC_TIPO | DEC_PROC;
DEC_VAR := **var** TIPO **id**;
DEC_TIPO := **type** TIPO **id;**
DEC_PROC := **proc id** PARAMS BLOQUE;

PARAMS := **pap** LPARAMS **pcierre**;
LPARAMS := LPARAMS **coma** PARAM;
LPARAMS := PARAM;
LPARAMS := **ƛ** ;
PARAM := TIPO (**amp** | **ƛ**) **id**;

TIPO := TIPO_BASICO | TIPO_ARRAY | TIPO_REG | TIPO_PUNT | **id**;
TIPO_BASICO := **r_real** | **r_int** | **r_bool** | **r_string** ;
TIPO_ARRAY := **array cap ent ccierre of** TIPO;
TIPO_REG := **record llap** CAMPOS **llcierre**;
TIPO_PUNT := **pointer** TIPO**;**

CAMPOS := CAMPOS **scol** CAMPO;
CAMPOS := CAMPO;
CAMPO := TIPO **id**;

LINST := LINST **scol** INST;
LINST := INST;
INST := INST_ASIG | INST_IF_THEN | INST_IF_THEN_ELSE | INST_WHILE | INST_READ | INST_WRITE | INST_NL | INST_NEW | INST_DELETE | INST_CALL | INST_COMPUESTA;
INST_ASIG := E0 **igual** E0;
INST_IF_THEN := **if** E0 **then** AUX_LINST **endif**;
INST_IF_THEN_ELSE := **if** E0 **then** AUX_LINST **else** AUX_LINST **endif**;
INST_WHILE := **while** E0 **do** AUX_LINST **endwhile**;
AUX_LINST := LINST;
AUX_LINST := **ƛ** ;
INST_READ := **read** E0;

INST_WRITE := **write** E0;
INST_NL := **nl**;
INST_NEW := **new** E0;
INST_DELETE := **delete** E0;
INST_CALL := **call id pap** REAL_PARAMS **pcierre**;
REAL_PARAMS := REAL_PARAMS **coma** E0;
REAL_PARAMS := E0;
REAL_PARAMS := **λ** ;
INST_COMPUESTA := BLOQUE;
BLOQUE := **llap** PROGRAMA **llcierre**;
BLOQUE := **llap llcierre**;

E0 := E1 **mas** E0;
E0 := E1 **menos** E1;
E0 := E1;
E1 := E1 OP1AI E2;
E1 := E2;
E2 := E2 OP2AI E3;
E2 := E3;
E3 := E4 OP3NA E4;
E3 := E4;
E4 := **menos** E5;
E4 := **not** E4;
E4 := E5;
E5 := E5 OP5A;
E5 := E6;
E6 := por E6
E6 := E7;
E7 := **ent** | **real** | **cadena** | **true** | **false** | **id** | **null** | **pap** E0 **pcierre**;

OP5A := **cap** E0 **ccierre** | **punto id** | **flecha id**;
OP3NA := **por** | **div** | **mod**;
OP2AI := **bne** | **beq** | **ble** | **bge** | **blt** | **bgt**;
OP1AI := **and** | **or**;

ACONDICIONAMIENTO DE LA GRAMÁTICA (LL1)

Se añaden los siguientes **no terminales** al conjunto anterior : RLPARAMS, RLINST, RCAMPOS, RES0, RES1, RES2, RES3, RES5, INST_IF, RES_IF, RES_PARAMS , RPARAM y RBLOQUE.
Al sacar factor común en INST_IF_THEN e INST_IF_THEN_ELSE ,ya no hacen falta estos no terminales y se usan INST_IF y RES_IF en su lugar.

Gramática LL1:

PROGRAMA := LDECS LINST;
LDECS := DECS **sep_di**;
LDECS := **λ** ;
DECS := DEC RDECS;
RDECS := **scol** DEC RDECS;
RDECS := **λ** ;
DEC := DEC_VAR | DEC_TIPO | DEC_PROC;
DEC_VAR := **var** TIPO **id**;
DEC_TIPO := **type** TIPO **id**;
DEC_PROC := **proc id** PARAMS BLOQUE;

PARAMS := **pap** LPARAMS **pcierre**;
LPARAMS:= **λ** ;
LPARAMS:=PARAM RLPARAMS;
RLPARAMS:= **coma** PARAM RLPARAMS;
RLPARAMS:= **λ** ;
PARAM := TIPO RPARAM;
RPARAM := **id**;
RPARAM := **amp id**;

TIPO := TIPO_BASICO | TIPO_ARRAY | TIPO_REG | TIPO_PUNT | **id**;
TIPO_BASICO := **r_real** | **r_int** | **r_bool** | **r_string** ;
TIPO_ARRAY := **array cap ent ccierre of** TIPO;
TIPO_REG := **record llap** CAMPOS **llcierre**;
TIPO_PUNT := **pointer** TIPO;
CAMPOS:= CAMPO RCAMPOS;
RCAMPOS:= **scol** CAMPO RCAMPOS;
RCAMPOS:= **λ** ;
CAMPO := TIPO **id**;

LINST := INST RLINST;
RLINST := **scol** INST RLINST;
RLINST := **λ** ;
INST := INST_ASIG | INST_IF_THEN | INST_IF_THEN_ELSE | INST_WHILE |
INST_READ | INST_WRITE | INST_NL | INST_NEW | INST_DELETE |
INST_CALL | INST_COMPUESTA;
INST_ASIG := E0 **igual** E0;
INST_IF := **if** E0 **then** AUX_LINST RES_IF **endif**;
RES_IF := **λ** ;
RES_IF := **else** AUX_LINST;
INST_WHILE := **while** E0 **do** AUX_LINST **endwhile**;
AUX_LINST := LINST;
AUX_LINST := **λ** ;
INST_READ := **read** E0;
INST_WRITE := **write** E0;
INST_NL := **nl**;

INST_NEW := **new** E0;
INST_DELETE := **delete** E0;
INST_CALL := **call id pap** REAL_PARAMS **pcierre**;
REAL_PARAMS := **ƛ** ;
REAL_PARAMS := E0 RES_PARAMS;
RES_PARAMS := **coma** E0 RES_PARAMS;
RES_PARAMS := **ƛ** ;
INST_COMPUESTA := BLOQUE;
BLOQUE := **llap** RBLOQUE;
RBLOQUE := **llcierre**;
RBLOQUE := PROGRAMA **llcierre**;

E0 := E1 RES0;
RES0 := **mas** E0;
RES0:= **menos** E1;
RES0 := **ƛ** ;
E1 := E2 RES1;
RES1 := OP1AI E2 RES1;
RES1 := **ƛ** ;
E2 := E3 RES2;
RES2 := OP2AI E3 RES2;
RES2:= **ƛ** ;
E3 := E4 RES3;
RES3 := OP3NA E4;
RES3:=**ƛ** ;
E4 := **menos** E5;
E4 := **not** E4;
E4 := E5;
E5 := E6 RES5;
RES5 := OP5A RES5;
RES5 := **ƛ** ;
E6 := **por** E6;
E6 := E7;
E7 := **ent** | **real** | **cadena** | **true** | **false** | **id** | **null** | **pap** E0 **pcierre**;

OP5A := **punto id** | **flecha id** | **cap** E0 **ccierre**;
OP3NA := **por** | **div** | **mod**;
OP2AI := **bne** | **beq** | **ble** | **bge** | **blt** | **bgt**;
OP1AI := **and** | **or**;