programa -> **programa** **ID**  
{ ADD\_INST(‘prog ‘ || ID.value); }  
declaraciones decl\_de\_subprogs  
**comienzo** lista\_de\_sentencias’ **fin ;**{ ADD\_INST('halt'); }

declaraciones -> **variables** lista\_de\_ident **:** tipo **;**  
{ if ( IS\_INTEGER(tipo.tipo) ||  
 IS\_REAL(tipo.tipo) ||  
 IS\_BOOLEAN(tipo.tipo) )  
 {  
 FOREACH ( lista\_de\_ident.ids AS ident )  
 ADD\_INST(TYPE\_OF(tipo.tipo) || ' ' || ident);  
 }  
 else if ( IS\_ARRAY(tipo.tipo) )  
 {  
 FOREACH ( lista\_de\_ident.ids AS ident )  
 ADD\_INST('array\_' || TYPE\_OF(ARRAY\_CONTENT(tipo.tipo))  
 || ' ' || ident || ',' ||  
 ARRAY\_SIZE(tipo.tipo));  
 }  
}  
declaraciones  
| ξ

lista\_de\_ident -> **ID** resto\_lista\_ident  
{ lista\_de\_ident.ids =  
 JOIN(INIT\_LIST(ID.value), resto\_lista\_ident.ids); }

resto\_lista\_ident -> **,** **ID** resto\_lista\_ident  
{ resto\_lista\_ident.ids =  
 JOIN(INIT\_LIST(ID.value), resto\_lista\_ident.ids); }  
| ξ  
{ resto\_lista\_ident.ids = EMPTY\_LIST(); }

tipo -> **entero** { tipo.tipo = NEW\_BASIC\_TYPE(INTEGER); }| **real** { tipo.tipo = NEW\_BASIC\_TYPE(REAL); }| **booleano** { tipo.tipo = NEW\_BASIC\_TYPE(REAL); }| **array** **[** lista\_de\_enteros **] de** tipo  
{ tipo.tipo =  
 NEW\_ARRAY\_TYPE(lista\_de\_enteros.ints, tipo.tipo); }

lista\_de\_enteros -> **INTEGER** resto\_lista\_enteros  
{ lista\_de\_enteros.ints =  
 JOIN(INIT\_LIST(INTEGER.value), resto\_lista\_enteros.ints); }

resto\_lista\_enteros -> **, INTEGER** resto\_lista\_enteros  
{ resto\_lista\_enteros.ints =  
 JOIN(INIT\_LIST(INTEGER.value), resto\_lista\_enteros.ints); }  
| ξ { resto\_lista\_enteros.ints = EMPTY\_LIST(); }

decl\_de\_subprogs -> decl\_de\_procedimiento decl\_de\_subprogs  
| decl\_de\_funcion decl\_de\_subprogs  
| ξ

decl\_de\_procedimiento -> cabecera\_procedimiento declaraciones **comienzo** lista\_de\_sentencias’ **fin ;**

decl\_de\_funcion -> cabecera\_funcion declaraciones **comienzo** lista\_de\_sentencias’ **retornar** expresion **fin ;**

cabecera\_procedimiento -> **procedimiento ID** argumentos

cabecera\_funcion -> **funcion ID** argumentos **retorna** tipo

argumentos -> **(** lista\_de\_param **)**  
| ξ

lista\_de\_param -> lista\_de\_ident **:** clase\_param tipo resto\_lis\_de\_param

resto\_lis\_de\_param -> **;** lista\_de\_ident **:** clase\_param tipo resto\_lis\_de\_param  
| ξ

clase\_param -> **entrada** clase\_param’  
| **salida**

clase\_param’ -> **salida**  
| ξ

lista\_de\_sentencias’ -> lista\_de\_sentencias

lista\_de\_sentencias -> sentencia lista\_de\_sentencias  
| ξ

sentencia -> **ID** expresiones **;**  
| **si** expresion **entonces** M lista\_de\_sentencias **fin si** M **;**  
| **hacer** M lista\_de\_sentencias **mientras** expresion **fin hacer** M **;**  
| **salir si** expresion M **;**| **get ( ID** id\_o\_array **) ;**| **put\_line (** expresion **) ;**

id\_o\_array -> acceso\_a\_array  
| ξ

expresiones -> **=** expresion  
| acceso\_a\_array **=** expresion  
| parametros\_llamadas

acceso\_a\_array -> **[** lista\_de\_expr **]**

parametros\_llamadas -> **(** lista\_de\_expr **)**

expresion -> disyuncion

disyuncion -> conjuncion disyuncion’

disyuncion’ -> **or** conjuncion disyuncion’  
| ξ

conjuncion -> relacional conjuncion’

conjuncion’ -> **and** relacional conjuncion’  
| ξ

relacional -> aritmetica relacional’

relacional’ -> **oprel** aritmetica relacional’  
| ξ

aritmetica -> termino aritmetica’

aritmetica’ -> **opl2** termino aritmetica’  
| ξ

termino -> negacion termino’

termino’ -> **opl1** negacion termino’  
| ξ

negacion -> **not** factor  
| factor

factor -> **-** factor’  
| factor’

factor’ -> **ID** array\_o\_llamada  
| **INTEGER**| **REAL**  
| booleano  
| **(** expresion **)**

array\_o\_llamada -> parametros\_llamadas  
| acceso\_a\_array  
| ξ

opl1 -> **\*** | **/**

opl2 -> **+** | **-**

oprel -> **>**| **<**| **>=**| **<=**| **==**| **/=**

booleano -> **true** | **false**

M -> ξ

lista\_de\_expr -> expresion resto\_lista\_expr

resto\_lista\_expr -> **,** expresion resto\_lista\_expr  
| ξ