```
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
```