Statement\_list -> Statement statement\_list

|épsilon

Statement -> **assign**

|**printf**(expression);

|**scanf**([“i”| “d”|”c” | “s” | “f”],ID [,[ “i”| “d”|”c” | “s” | “f”], ID]\*);

|**if** (expression) Compound\_statement **else** Compound\_statement

|**while** (expression) Compound\_statement

|**do** Compund\_statement **while**(expresion);

|**for (**expression opt; expression opt2; expression op3**)** Compound\_statement  
 | **return**;  
 | **break**;

|**switch** (expression){[case CHAR|INT : optional\_statements;]\* default: Statement\_list;

Compound\_statement -> **{** Statement\_list |epsilon**}**

Compund\_Struct -> {[assign;]\*}

Assign -> tipo lvalue[;| = expression];

lvalue -> ID arreglos

arreglos -> [expresion]  
 | epsilon

Statement\_p -> **(**expression\_list**)**

|**[**expression**]**

|epsilon

<method\_call2> -> ‘(‘(<expr>(‘,’<expr>)\*)? ‘)

<exprP> -> ’[‘ <expr> ‘]’  
 | | ‘(‘ <methodcall2>

expression\_list -> expression **,** expression\_list

|epsilon

Expression -> Relacional

<Relacional> -> | AritmeticoSumaResta ( != AritmeticoSumaResta) \*

| AritmeticoSumaResta ( <= AritmeticoSumaResta) \*

| AritmeticoSumaResta ( >= AritmeticoSumaResta) \*

| AritmeticoSumaResta ( == AritmeticoSumaResta) \*

| AritmeticoSumaResta ( < AritmeticoSumaResta) \*

| AritmeticoSumaResta ( > AritmeticoSumaResta) \*

< AritmeticoSumaResta > -> Produccion (‘+’ Produccion) \*

| Produccion (‘-‘ Produccion) \*

| Produccion (‘or’ Produccion) \*

<Produccion> -> Shift (‘/’ Shift) \*

|Shift( ‘\*’ Shift)\*

| Shift (‘%’ Shift)\*

| Shift(‘ and’ Shift)\*

<Shift> -> Term (‘>>’ Term) \*

|Term (‘<<’ Term)\*

<Term> -> <Tipo>

|’(‘ <expr> ‘)’

| ID exprP

parexp -> (expression)  
 | tipo

Tipo -> char String  
 | int  
 |double  
 |float  
 |boolconst

String -> \*   
 | epsilon

Boolconst -> true|false