Gramática libre de contexto

start -> (‘var’ defVar ‘;’ | defStruct | defFunc)* EOF

defVar -> IDENT ‘:’ tipo

defFunc -> IDENT ‘(‘ funcDefParams ‘)’ (‘:’ tipo)? ‘{‘ (‘var’ defVar ‘;’)* sentencia* ‘}’

defStruct -> ‘struct’ IDENT ‘{‘ (campo ‘;’)* ‘}’ ‘;’

campo -> IDENT ‘:’ tipo

tipo -> ‘int’

| ‘float’

| ‘char’

| IDENT

| ‘[‘ LITENT ‘]’ tipo

sentencia -> (‘print’ | ‘printsp’ | ‘println’) expr? ‘;’

| ‘read’ expr ‘;’

| expr ‘=’ expr ‘;’

| ‘if’ ‘(‘ expr ‘)’ ‘{‘ sentencia* ‘}’ (‘else‘ ‘{‘ sentencia* ‘}’)?

| ‘while’ ‘(expr ‘)’ ‘{‘ sentencia* ‘}’

| IDENT ‘(‘ params ‘)’ ‘;’

| ‘return’ expr? ‘;’

expr -> LITENT

| LITREAL

| LITCHAR

| IDENT

| expr ‘.’ expr

| expr ‘[‘ expr ‘]’

| ‘<’ tipo ‘>’ ‘(‘ expr ‘)’

| ‘(‘ expr ‘)’

| ‘!’ expr

| expr (‘*’ | ‘/’ | ‘%’) expr

| expr (‘+’ | ‘-’) expr

| expr ('<' | '>' | '>=' | '<=') expr

| expr (‘==’ | ‘!=’) expr

| expr ‘&&’ expr

| expr ‘||’ expr

| IDENT ‘(‘ params ‘)’

params -> (expr (‘,’ expr)*)?

funcDefParams -> (defVar (‘,’ defVar)*)?