MACSLang - Gramática do Compilador

Símbolo Inicial:

```
Program ::= { Declaration } EOF
```

Declarações:

```
Declaration ::= VarDeclaration

| FuncDeclaration

| Statement
```

Declaração de Variável:

```
VarDeclaration ::= VAR IDENTIFIER COLON Type [ ASSIGN Expression ] SEMICOLON
```

Tipos:

```
Type ::= INT_KEYWORD

| FLOAT_KEYWORD

| CHAR_KEYWORD

| BOOL_KEYWORD

| STRING_KEYWORD
```

Declaração de Função:

```
FuncDeclaration ::= FUNC IDENTIFIER OPEN_PAREN [ Parameters ] CLOSE_PAREN COLON Type
Block


Parameters ::= Parameter { COMMA Parameter }

Parameter  ::= IDENTIFIER COLON Type
```

Bloco:

```
Block ::= OPEN_BRACE { Declaration | Statement } CLOSE_BRACE
```

Comandos:

```
Statement ::= ExpressionStatement

| IfStatement
```

```
| WhileStatement

| ForStatement

| ReturnStatement

| PrintStatement

| InputStatement

| Block
```

Comando de Retorno:

```
ReturnStatement ::= RETURN [ Expression ] SEMICOLON
```

Print e Input:

```
PrintStatement ::= PRINT OPEN_PAREN Expression CLOSE_PAREN SEMICOLON

InputStatement ::= INPUT OPEN_PAREN IDENTIFIER CLOSE_PAREN SEMICOLON
```

Controle de Fluxo:

```
IfStatement ::= IF OPEN_PAREN Expression CLOSE_PAREN Block [ ELSE Block ]

WhileStatement ::= WHILE OPEN_PAREN Expression CLOSE_PAREN Block

ForStatement ::= FOR OPEN_PAREN VarDeclaration Expression SEMICOLON Expression
CLOSE_PAREN Block
```

Expressão:

```
Expression ::= LogicalOr


LogicalOr ::= LogicalAnd { OR LogicalAnd }

LogicalAnd ::= Equality { AND Equality }

Equality ::= Comparison { (EQUALS | NOT_EQUALS) Comparison }

Comparison ::= Term { (LESS_THAN | GREATER_THAN | LESS_EQUAL | GREATER_EQUAL) Term }

Term ::= Factor { (PLUS | MINUS) Factor }

Factor ::= Unary { (MULTIPLY | DIVIDE | MODULO) Unary }
```

```
Unary ::= (NOT | MINUS)? Primary

Primary ::= IDENTIFIER

| Literal

| FunctionCall

| OPEN_PAREN Expression CLOSE_PAREN
```

Chamada de Função:

```
FunctionCall ::= IDENTIFIER OPEN_PAREN [ Arguments ] CLOSE_PAREN

Arguments ::= Expression { COMMA Expression }
```

Literais:

```
Literal ::= INT_LITERAL

| FLOAT_LITERAL

| CHAR_LITERAL

| STRING_LITERAL

| TRUE

| FALSE
```

Expressão como Comando:

```
ExpressionStatement ::= Expression SEMICOLON
```