Analisador Sintático - EBNF

```
main: (<?PHP inner_statement* ?>)* [EOF];

inner_statement: statement
    | function_declaration_statement

statement: "{" inner_statement* "}"
    | "if" "(" expr ")" statement elseif_branch? else_single? new_else_single? "endif" ";"
    | "while" "(" expr ")" while_statement
    | "do" statement "while" "(" expr ")" ";"
    | "for" "(" for_expr ";" for_expr ";" for_expr ")" for_statement
    | "break" expr? ";"
    | "continue" expr? ";"
    | "return" expr_without_variable ";"
    | "return" variable? ";"
    | "global" global_var ("," global_var)* ";"
    | expr ";"
    | "foreach" "(" variable "as" foreach_variable ("=>" foreach_variable)? ")" foreach_statement
    | "foreach" "(" expr_without_variable "as" foreach_variable ("=>" foreach_variable)? ")"
foreach_statement
    | ";"

function_declaration_statement: "function" "&"? T_STRING
    "(" parameter_list ")" "{" inner_statement* "}" ;

foreach_variable: "&"? variable ;

for_statement: statement
    | ":" inner_statement* "endfor" ";" ;

foreach_statement: statement
    | ":" inner_statement* "endforeach" ";" ;

declare_statement: statement
    | ":" inner_statement* "enddeclare" ";" ;

declare_list: T_STRING "=" static_scalar ("," T_STRING "=" static_scalar)*;

while_statement: statement
    | ":" inner_statement* "endwhile" ";" ;

elseif_branch: "elseif" "(" expr ")" statement ;

new_elseif_branch: "elseif" "(" expr ")" ":" inner_statement* ;

else_single: "else" statement ;

new_else_single: "else" ":" inner_statement* ;

parameter_list: parameter ("," parameter)* ;

parameter: T_STRING "&"? T_VARIABLE ("=" static_scalar)? | "array" "&"? T_VARIABLE ("="
static_scalar)? ;

function_call_parameter_list: function_call_parameter
```

```
    ("," function_call_parameter)* ;

function_call_parameter: expr_without_variable
    | variable
    | "&" T_VARIABLE ;

global_var: T_VARIABLE
    | "$" T_VARIABLE
    | "$" "{" expr "}" ;

variable_modifiers: "var" | modifier (modifier)* ;

for_expr: expr ("," expr)* ;

expr_without_variable: "list" "(" assignment_list_element ("," assignment_list_element)* ")" "="
expr
    | variable "=" expr
    | variable "=" "&" variable
    | "clone" expr
    | T_VARIABLE "++"
    | "++" T_VARIABLE
    | T_VARIABLE "--"
    | "--" T_VARIABLE
    | "(" expr ")"
    | expr "?" expr ":" expr
    | "(int)" expr
    | "(double)" expr
    | "(float)" expr
    | "(real)" expr
    | "(string)" expr
    | "(array)" expr
    | "(object)" expr
    | "(bool)" expr
    | "(boolean)" expr
    | "(unset)" expr
    | "exit" exit_expr?
    | "die" exit_expr?
    | "@" expr
    | scalar
    | "array" "(" array_pair_list? ")"
    | "`" encaps* "`" ;

function_call: T_STRING "(" function_call_parameter_list ")"
    | variable_without_objects "(" function_call_parameter_list ")" ;

exit_expr: "(" expr? ")" ;

common_scalar: T_LNUMBER | T_DNUMBER | T_CONSTANT_ENCAPSED_STRING
    | "__LINE__" | "__METHOD__" | "__FUNCTION__" ;

static_scalar: common_scalar
    | T_STRING
    | "+" static_scalar
    | "-" static_scalar
    | "array" "(" static_array_pair_list? ")" ;


scalar: T_STRING
```

```
    | T_STRING_VARNAME
    | common_scalar
    | "\"" encaps* "\""
    | "'" encaps*    "'"

static_array_pair_list: static_array_pair ("," static_array_pair)* ","? ;

static_array_pair: static_scalar ("=>" static_scalar)? ;

expr: T_VARIABLE | expr_without_variable ;

variable: base_variable_with_function_calls ( "->" object_property
      method_parameters ("->" object_property method_parameters)* )? ;

method_parameters: "(" function_call_parameter_list ")" ;

variable_without_objects: reference_variable
    | simple_indirect_reference reference_variable ;

base_variable_with_function_calls: base_variable | function_call ;

base_variable: reference_variable
    | simple_indirect_reference reference_variable

reference_variable: compound variable (selector)* ;

compound_variable: T_VARIABLE | "$" "{" expr "}" ;

selector: "[" expr? "]" ;

variable_name: T_STRING ;

simple_indirect_reference: "$" ("$")* ;

assignment_list_element: variable
    | "list" "(" assignment_list_element ("," assignment_list_element)* ")" ;

array_pair_list: array_pair ("," array_pair)* ","? ;

array_pair: "&" T_VARIABLE
    | expr "=>" "&" T_VARIABLE
    | expr "=>" expr ;

encaps:
        encaps_var
        | T_STRING
        | "["
        | "]"
        | "{"
        | "}"
        | "->"
     ;

encaps_var: T_VARIABLE ( "[" encaps_var_offset "]" )?
    | T_VARIABLE "->" T_STRING
    | "${" expr "}"
    | "${" T_STRING_VARNAME "[" expr "]" "}"
    | T_CURLY_OPEN variable "}" ;
```

```
encaps_var_offset: T_STRING | T_VARIABLE ;

# Some tokens from the scanner (see file Zend/zend_language_scanner.l):

LABEL: "[a-zA-Z_][a-zA-Z0-9_]*";
T_STRING: "[a-zA-Z_][a-zA-Z0-9_]*";
T_VARIABLE: "\$[_a-zA-Z_][a-zA-Z0-9_]*" ;

T_LNUMBER: "-?0|[1-9][0-9]*";
T_DNUMBER: "-?(0?|[1-9][0-9]*)\.[0-9]+([eE][-+]?[0-9]+)?";

T_CURLY_OPEN: "${";

T_CONSTANT_ENCAPSED_STRING: "'[^']*'|\"[^\"]*\"";

T_STRING_VARNAME: "[a-zA-Z_][a-zA-Z0-9_]*";

NEWLINE: "\r|\n|\r\n";
```