Disciplina: Compiladores

Atividade: BNF da linguagem

Alunos:

- **José Arthur Lopes**
- **Denilson Bulhões**
- **Marcos Ivan**

Linguagem:

- **PHP**

Tipos:

- **Inteiro**
- **Array**

Keywords:

- **if**
- **else**
- **elseif**
- **while**
- **do**

- **for**
- **function**
- **array**
- **switch**
- **case**
- **default**

- **break**
- **continue**
- **static**
- **global**
- **return**
- **echo**

Operadores:

- **+**
- **-**
- *****
- **/**
- **%**
- ******
- **++**
- **--**
- **=**

- **+=**
- **-=**
- ***=**
- **/=**
- **||**
- **&&**
- **<<**
- **>>**
- **===**
- **!==**

- **==**
- **!=**
- **<=**
- **>=**
- **<**
- **>**
- **!**
- **?**
- **~**

PHP_SOURCE_TEXT = { inner_statement | halt_compiler_statement };

halt_compiler_statement = "__halt_compiler" "(" ")" ";" ;

inner_statement = statement

      | function_declaration_statement

      | class_declaration_statement ;

inner_statement_list = { inner_statement } ;

statement = "{" inner_statement_list "}"

      | "if" "(" expr ")" statement {elseif_branch} [else_single]

      | "if" "(" expr ")" ":" inner_statement_list {new_elseif_branch}

        [new_else_single] "endif" ";"

      | "while" "(" expr ")" while_statement

      | "do" statement "while" "(" expr ")" ";"

      | "for" "(" for_expr ";" for_expr ";" for_expr ")" for_statement

      | "switch" "(" expr ")" switch_case_list

      | "break" [expr] ";"

      | "continue" [expr] ";"

      | "return" [expr_without_variable | variable] ";"

      | "global" global_var {"," global_var} ";"

      | "static" static_var { "," static_var } ";"

      | "echo" echo_expr_list ";"

      | T_INLINE_HTML

      | expr ";"

      | "use" use_filename ";" # FIXME: not implemented

      | "unset" "(" variable {"," variable} ")" ";"

      | "foreach" "(" (variable|expr_without_variable)

        "as" foreach_variable ["=>" foreach_variable] ")"

foreach_statement

        | "declare" "(" declare_list ")" declare_statement

        | ";" | "try" "{" inner_statement_list "}" catch_branch {catch_branch}

        | "throw" expr ";" ;

catch_branch = "catch" "(" fully_qualified_class_name T_VARIABLE ")" "{"

        inner_statement_list "}" ;

use_filename = T_CONSTANT_ENCAPSED_STRING

        | "(" T_CONSTANT_ENCAPSED_STRING ")" ;

function_declaration_statement = "function" ["&"] T_STRING

        "(" parameter_list ")" "{" inner_statement_list "}" ;

class_declaration_statement = class_entry_type T_STRING

        [extends_from] [implements_list] "{" {class_statement} "}"

        | "interface" T_STRING [interface_extends_list] "{" {class_statement} "}" ;

class_entry_type = ["abstract" | "final"] "class" ;

for_statement = statement

        | ":" inner_statement_list "endfor" ";" ;

declare_statement = statement

        | ":" inner_statement_list "enddeclare" ";" ;

declare_list = T_STRING "=" static_scalar { "," T_STRING "=" static_scalar } ;

switch_case_list = "{" [";"] {case_list} "}"

        | ":" [";"] {case_list} "endswitch" ";" ;

case_list = "case" expr [":"|";"] inner_statement_list

        | "default" [":"|";"] inner_statement_list ;

while_statement = statement

| ":" inner_statement_list "endwhile" ";" ;

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

new_elseif_branch = "elseif" "(" expr ")" <mark>":" inner_statement_list</mark> ;

else_single = "else" statement ;

new_else_single = "else" <mark>":" inner_statement_list</mark> ;

parameter_list = [parameter {"," parameter}] ;

parameter = [T_STRING | "array"] ["&"] T_VARIABLE ["=" static_scalar] ;

function_call_parameter_list = [function_call_parameter

       { "," function_call_parameter }] ;

function_call_parameter = expr_without_variable

       | variable

       | "&" w_variable ;

global_var = T_VARIABLE

       | "$" r_variable

       | "$" "{" expr "}" ;

static_var = T_VARIABLE ["=" static_scalar] ;

<mark>class_statement = variable_modifiers class_variable_declaration</mark>

       <mark>{"," class_variable_declaration} ";"</mark>

       <mark>| "const" class_constant_declaration {"," class_constant_declaration} ";"</mark>

       <mark>| {modifier} "function" ["&"] T_STRING "(" parameter_list ")"</mark>

       <mark>method_body ;</mark>

<mark>method_body = ";"</mark>

       <mark>| "{" inner_statement_list "}" ;</mark>

<mark>variable_modifiers = "var" | modifier {modifier} ;</mark>

<mark>modifier = "public" | "protected" | "private" | "static" | "abstract"</mark>

class_variable_declaration = ("var" | modifier {modifier}) T_VARIABLE ["=" static_scalar] ;

class_constant_declaration = T_STRING "=" static_scalar ;

echo_expr_list = expr {"," expr} ;

for_expr = [expr {"," expr}] ;

expr_without_variable = "list" "(" assignment_list ")" "=" expr

    | variable "=" expr

    | variable "=" "&" variable

    | variable "=" "&" "new" class_name_reference [ctor_arguments]

    | "new" class_name_reference [ctor_arguments]

    | "clone" expr

    | variable ("+=" | "-=" | "*=" | "/=" | ".=" | "%=" | "&=" | "|=" |

    "^=" | "<<=" | ">>=") expr

    | rw_variable "++"

    | "++" rw_variable

    | rw_variable "--"

    | "--" rw_variable

    | expr ("||" | "&&" | "or" | "and" | "xor" | "|" | "&" | "^" | "." |

    "+" | "-" | "*" | "/" | "%" | "<<" | ">>" | "===" | "!==" |

    "<" | "<=" | ">" | ">=") expr

    | ("+" | "-" | "!" | "~") expr

    | expr "instanceof" class_name_reference

    | "(" expr ")"

    | expr "?" expr ":" expr

    | internal_functions

    | "(int)" expr

| "(double)" expr

        | "(float)" expr

        | "(real)" expr

        | "(string)" expr

        | "(array)" expr

        | "(object)" expr

        | "(bool)" expr

        | "(boolean)" expr

        | "(unset)" expr # FIXME: not implemented

        | "exit" [exit_expr]

        | "die" [exit_expr]

        | "@" expr

        | scalar

        | "array" "(" [array_pair_list] ")"

        | "`" encaps_list "`"

        | "print" expr ;

function_call = T_STRING "(" function_call_parameter_list ")"

        | fully_qualified_class_name "::" T_STRING

          "(" function_call_parameter_list ")"

        | fully_qualified_class_name "::" variable_without_objects

          "(" function_call_parameter_list ")"

        | variable_without_objects "(" function_call_parameter_list ")" ;

fully_qualified_class_name = T_STRING ;

class_name_reference = T_STRING

        | dynamic_class_name_reference ;

method_parameters = "(" function_call_parameter_list ")" ;

variable_without_objects = reference_variable

      | simple_indirect_reference reference_variable ;

static_member = fully_qualified_class_name "::" variable_without_objects ;

base_variable_with_function_calls = base_variable | function_call ;

base_variable = reference_variable

      | simple_indirect_reference reference_variable

      | static_member ;

reference_variable = compound_variable { selector } ;

selector = "[" [expr] "]" | "{" expr "}" ;

object_property = variable_name { selector }

      | variable_without_objects ;

variable_name = T_STRING | "{" expr "}" ;

simple_indirect_reference = "$" {"$"} ;

assignment_list = [assignment_list_element] {"," [assignment_list_element]} ;

assignment_list_element = variable

      | "list" "(" assignment_list ")" ;

array_pair_list = array_pair {"," array_pair} [","] ;

array_pair = "&" w_variable

      | expr "=>" "&" w_variable

      | expr "=>" expr ;

encaps_list =

      {

      encaps_var

      | T_STRING

      | T_NUM_STRING

| T_ENCAPSED_AND_WHITESPACE

          | T_CHARACTER

          | T_BAD_CHARACTER

          | "["

          | "]"

          | "{"

          | "}"

          | "->"

          } ;

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_NUM_STRING | T_VARIABLE ;

internal_functions = "isset" "(" variable {"," variable} ")"

          | "empty" "(" variable ")"

          | "include" expr

          | "include_once" expr

          | "eval" "(" expr ")"

          | "require" expr

          | "require_once" expr ;

class_constant = fully_qualified_class_name "::" T_STRING ;

LABEL = (letter | "_") {letter | digit | "_"} ;

T_STRING = LABEL;

T_BAD_CHARACTER = "\x00".."\x08" | "\x0b" | "\x0c" | "\x0e".."\x1f" ;

T_VARIABLE = "$" T_STRING ;

T_LNUMBER = octal | decimal | hexadecinal ;

octal = "0" {"0".."7"} ;

decimal = "1".."9" {digit} ;

hexadecinal = "0x" hexdigit {hexdigit} ;

digit = "0".."9" ;

hexdigit = digit | "a".."f" | "A".."F" ;

letter = "a".."z" | "A".."Z" | "\x7f".."\xff" ;

T_DNUMBER = DNUM | EXPONENT_DNUM;

DNUM = digit ["."] digit {digit} | digit {digit} ["."] {digit};

EXPONENT_DNUM = (LNUM | DNUM) ("e"|"E") ["+"|"-"] LNUM;

LNUM = digit {digit};

T_CURLY_OPEN = "${";

T_CONSTANT_ENCAPSED_STRING = single_quoted_constant_string | double_quoted_constant_string;

FIXME

single_quoted_constant_string =

        "'" { "any char except ' and \\" | "\\" "any char" } "'";

FIXME

double_quoted_constant_string =

        "\"" { "any char except $ \" and \\" | "\\" "any char" } "\"";

T_STRING_VARNAME = LABEL;

T_NUM_STRING = LNUM | hexadecinal;

T_START_HEREDOC = "<?php;

NEWLINE = "\r"|"\n"|"\r\n";

T_END_HEREDOC = "?>"

        LABEL [";"] NEWLINE;

PHP_SOURCE_TEXT = { inner_statement }

inner_statement = statement

    | function_declaration_statement ;

inner_statement_list = { inner_statement } ;

statement = "{" inner_statement_list "}"

    | "if" "(" expr ")" statement {elseif_branch} [else_single]

    | "while" "(" expr ")" while_statement

    | "do" statement "while" "(" expr ")" ";"

    | "for" "(" for_expr ";" for_expr ";" for_expr ")" for_statement

    | "switch" "(" expr ")" switch_case_list

    | "break" [expr] ";"

    | "continue" [expr] ";"

    | "return" [expr_without_variable | variable] ";"

    | "global" global_var {"," global_var} ";"

    | "static" static_var { "," static_var } ";"

    | "echo" echo_expr_list ";"

    | expr ";"

function_declaration_statement = "function" ["&"] T_STRING

    "(" parameter_list ")" "{" inner_statement_list "}" ;

for_statement = statement

    | ":" inner_statement_list "endfor" ";" ;

switch_case_list = "{" [";"] {case_list} "}"

    | ":" [";"] {case_list} "endswitch" ";" ;

case_list = "case" expr [":"|";"] inner_statement_list

    | "default" [":"|";"] inner_statement_list ;

while_statement = statement

        | ":" inner_statement_list "endwhile" ";" ;

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

new_elseif_branch = "elseif" "(" expr ")" ":" inner_statement_list ;

else_single = "else" statement ;

new_else_single = "else" ":" inner_statement_list ;

parameter_list = [parameter {"," parameter}] ;

parameter = [T_STRING | "array"] ["&"] T_VARIABLE ["=" static_scalar] ;

function_call_parameter_list = [function_call_parameter

        { "," function_call_parameter }] ;

function_call_parameter = expr_without_variable

        | variable

        | "&" w_variable ;

global_var = T_VARIABLE

        | "$" r_variable

        | "$" "{" expr "}" ;

static_var = T_VARIABLE ["=" static_scalar] ;

echo_expr_list = expr {"," expr} ;

for_expr = [expr {"," expr}] ;

expr_without_variable = "list" "(" assignment_list ")" "=" expr

        | variable "=" expr

        | variable "=" "&" variable

        | variable ("+=" | "-=" | "*=" | "/=" | "%=") expr

        | rw_variable "++"

        | "++" rw_variable

| rw_variable "--"

| "--" rw_variable

| expr ("||" | "&&" | "+" | "-" | "*" | "/" | "%" | "<<" | ">>" | "===" | "!==" |

 "<" | "<=" | ">" | ">=") expr

| ("+" | "-" | "!" | "~") expr

| "(" expr ")"

| expr "?" expr ":" expr

| internal_functions

| "(int)" expr

| "(array)" expr

| "array" "(" [array_pair_list] ")"

| "`" encaps_list "`"

| "print" expr ;

function_call = T_STRING "(" function_call_parameter_list ")"

        "(" function_call_parameter_list ")"

        "(" function_call_parameter_list ")"

        | variable_without_objects "(" function_call_parameter_list ")" ;

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 } ;

selector = "[" [expr] "]" | "{" expr "}" ;

variable_name = T_STRING | "{" expr "}" ;

simple_indirect_reference = "$" {"$"} ;

assignment_list = [assignment_list_element] {"," [assignment_list_element]} ;

assignment_list_element = variable

      | "list" "(" assignment_list ")" ;

array_pair_list = array_pair {"," array_pair} [","] ;

array_pair = "&" w_variable

      | expr "=>" "&" w_variable

      | expr "=>" expr ;

T_START_HEREDOC = "<?php;

NEWLINE = "\r"|"\n"|"\r\n";

T_END_HEREDOC = "?>"