Accassias

Developer Manual

Eric Therond

Sommaire

Chapitre 1

Introduction

1.1 Présentation d'Accassias

1.1.1 Description générale

Accassias est un outil pour l'aide à l'analyse statique de codes sources. Il est écrit dans un but d'apprentissage, à la fois pour l'étude de la programmation en *C++* et la recherche automatisée de défauts logiciel.

Chapitre 2

Implémentation

2.1 Analyse lexicale

Symbole lexical	Chaine
T_READPOL	t_readpol
T_PRINT	t_print
T_CLASS	class
T_NEW	new
T_FUNCTION	function
T_IDENTIFIER	([a-z]_)*
T_STRING	".*"
T_DECLR	declr
T_VARIABLE	$
T_NUMBER	[0-9]*
T_COMMA	,
T_DOTCOMMA	;
T_RIGHTBRACKET	]
T_LEFTBRACKET	[
T_RIGHTBRACE	}
T_LEFTBRACE	{
T_DOUBLEQUOTE	"
T_RIGHTPARENTHESIS	)
T_LEFTPARENTHESIS	(
T_EQUAL	=
T_ADD	+
T_SUB	-
T_DIV	
T_MUL	*
T_UNDEF	
T_END	
T_ERROR	
T_RETURN	return
T_UP_ARROW	
T_DOWN_ARROW	
T_TAB	
T_IF	if
T_ELSEIF	elseif
T_ELSE	else
T_FOR	for
T_DO	do
T_WHILE	while
T_THIS	this
T_SUP	>
T_INF	<
T_SUPEQUAL	>=
T_INFEQUAL	<=
T_SYSTEM	t_system
T_FPUTS	t_fputs
T_CFG_DOT	t_cfg_dot
T_CFG_COMPUTE	t_cfg_compute

2.2 Analyse syntaxique

```
aca ::= instruction*;

instruction ::= for_instruction | if_instruction | system_instruction
| readpol_instruction | include_instruction | print_instruction
| fputs_instruction | cfg_dot_instruction | cfg_compute_instruction
| ast_dot_instruction | return_instruction | call_instruction
| variable | variable_declaration | function_declaration | class_declaration

for_instruction ::= T_FOR T_LEFTPARENTHESIS variable T_DOTCOMMA expression T_DOTCOMMA
variable T_RIGHTPARENTHESIS bloc_instructions

if_instruction ::= T_IF T_LEFTPARENTHESIS expression T_RIGHTPARENTHESIS bloc_instructions
(T_ELSEIF T_LEFTPARENTHESIS expression T_RIGHTPARENTHESIS bloc_instructions)*
(T_ELSE bloc_instructions)?

system_instruction ::= T_SYSTEM

readpol_instruction ::= T_READPOL

include_instruction ::= T_INCLUDE

print_instruction ::= T_PRINT

fputs_instruction ::= T_FPUTS

cfg_dot_instruction ::= T_CFG_DOT

cfg_compute_instruction ::= T_CFG_COMPUTE

ast_dot_instruction ::= T_AST_DOT

return_instruction ::= T_RETURN expression T_DOTCOMMA

call_instruction ::= T_IDENTIFIER T_LEFTPARENTHESIS
((expression T_COMMA)* (expression T_COMMA))? T_RIGHTPARENTHESIS T_DOTCOMMA

variable ::= T_VARIABLE (T_IDENTIFIER | T_THIS) (T_SUB T_SUP)?
((T_LEFTBRACKET expression T_RIGHTBRACKET)? variable_affectation)
| call_instruction

variable_declaration ::= T_DECLR T_VARIABLE T_IDENTIFIER
(T_LEFTBRACKET expression T_RIGHTBRACKET)? variable_end

function_declaration ::= T_FUNCTION T_IDENTIFIER T_LEFTPARENTHESIS
(T_DECLR variable_declaration)* (T_RIGHTPARENTHESIS | T_END) bloc_instructions

class_declaration ::= T_CLASS T_IDENTIFIER bloc_classe

expression ::= term ((T_ADD | T_SUB | T_INF | T_INFEQUAL | T_SUP | T_SUPEQUAL) term)?

term ::= factor ((T_MUL | T_DIV) factor)?

factor ::= (T_NUMBER | T_STRING | T_VARIABLE | T_IDENTIFIER |
(T_LEFTPARENTHESIS expression T_RIGHTPARENTHESIS)

bloc_instructions ::=  T_LEFTBRACE (for_instruction | if_instruction | system_instruction
| readpol_instruction | include_instruction | print_instruction | fputs_instruction
| cfg_dot_instruction | cfg_compute_instruction | ast_dot_instruction | return_instruction
| call_instruction | variable | variable_declaration)* (T_RIGHTBRACE | T_END)
```

```
variable_affectation ::= T_EQUAL (T_NEW classe_instance | expression)? variable_end

variable_end ::=  T_RIGHTPARENTHESIS | T_DOTCOMMA

bloc_classe ::= T_LEFTBRACE (function_declaration | variable_declaration | variable)*
(T_END | T_RIGHTBRACE)

classe_instance ::= T_IDENTIFIER
```