

ValidateStatementAdder

1.0

作成 : Doxygen 1.7.3

Tue Feb 1 2011 09:38:59

Contents

1	データ構造索引	1
1.1	データ構造	1
2	ファイル索引	3
2.1	ファイル一覧	3
3	データ構造	7
3.1	構造体 abstract_syntax_tree	7
3.1.1	説明	8
3.2	構造体 array_offset	8
3.2.1	説明	8
3.3	構造体 divition_information	9
3.3.1	説明	9
3.4	構造体 for_information	9
3.4.1	説明	10
3.5	構造体 freemem_info	10
3.5.1	説明	10
3.6	構造体 function_information	10
3.6.1	説明	10
3.7	構造体 include_data	11
3.7.1	説明	11
3.8	構造体 memory_allocation_info	11
3.8.1	説明	11
3.9	構造体 param_information	11
3.9.1	説明	12
3.10	構造体 return_info	12
3.10.1	説明	12
3.11	構造体 struct_table	12
3.11.1	説明	13
3.12	構造体 typedef_table	13
3.12.1	説明	13
3.13	構造体 validate_statement	13
3.13.1	説明	14
3.14	構造体 validate_variable	14
3.14.1	説明	15
3.15	構造体 variable_table	15
3.15.1	説明	16
4	ファイル	17

4.1	ANSICInformation/ANSIC_CODE.h	17
4.1.1	説明	17
4.1.2	関数	17
4.1.2.1	ASTLIST_ITERATOR_1	17
4.1.2.2	ASTLIST_ITERATOR_2	18
4.1.2.3	ASTLIST_ITERATOR_3	18
4.1.2.4	ASTLIST_ITERATOR_4	18
4.1.2.5	ASTLIST_ITERATOR_5	18
4.1.2.6	ASTLIST_ITERATOR_6	19
4.1.2.7	ASTLIST_ITERATOR_7	19
4.2	ANSICInformation/AST.h	19
4.2.1	説明	20
4.2.2	型定義	21
4.2.2.1	AST	21
4.2.3	関数	21
4.2.3.1	deleteAST	21
4.2.3.2	findASTAddress	21
4.2.3.3	fprintDataFromAST	21
4.2.3.4	fprintfStatement	22
4.2.3.5	getArgumentAST	22
4.2.3.6	getArgumentString	22
4.2.3.7	getArgumentStringEnableExcept	23
4.2.3.8	getASTWithString	23
4.2.3.9	getStringFromAST	24
4.2.3.10	getStringFromASTEnableExcept	24
4.2.3.11	getStringReplaceASTtoString	24
4.2.3.12	multi_push_back_childrenAST	25
4.2.3.13	new_AST	25
4.2.3.14	printDataFromAST	25
4.2.3.15	printTargetASTNode	25
4.2.3.16	push_back_childrenAST	26
4.2.3.17	same_new_AST	27
4.2.3.18	setASTBlocklevelAndId	27
4.2.3.19	setASTReturnType	27
4.2.3.20	traverseAST	27
4.2.3.21	traverseASTwithXML	28
4.3	ANSICInformation/DivitionDeclarator.h	28
4.3.1	説明	28
4.3.2	関数	29
4.3.2.1	OutputSourceAfterDivitionDeclarator	29
4.4	ANSICInformation/DivitionInformation.h	29
4.4.1	説明	30
4.4.2	型定義	30
4.4.2.1	DIVITION_INFORMATION	30
4.4.3	関数	30
4.4.3.1	getDIVITION_INFORMATION_LIST	30
4.4.3.2	new_DIVITION_INFORMATION	31
4.4.3.3	new_DIVITION_INFORMATION_char	31
4.4.3.4	printDIVITION_INFORMATION_LIST	31
4.5	ANSICInformation/ForInformation.h	32

4.5.1	説明	32
4.5.2	型定義	33
4.5.2.1	FOR_INFORMATION	33
4.5.3	関数	33
4.5.3.1	getFOR_INFORMATION_LIST	33
4.5.3.2	new_FOR_INFORMATION	33
4.5.3.3	print_FOR_INFORMATION_LIST	34
4.5.3.4	searchFOR_INFORMATION_FromAST	34
4.6	ANSICInformation/FreeMemInfo.h	35
4.6.1	説明	35
4.6.2	型定義	36
4.6.2.1	FREEMEMINFO	36
4.6.3	関数	36
4.6.3.1	getFreememInfo	36
4.6.3.2	new_FREEMEMINFO	36
4.6.3.3	printFREEMEMINFO	36
4.7	ANSICInformation/FunctionInformation.h	37
4.7.1	説明	38
4.7.2	型定義	38
4.7.2.1	FUNCTION_INFORMATION	38
4.7.2.2	PARAM_INFORMATION	38
4.7.3	関数	38
4.7.3.1	deleteParameterDefine	38
4.7.3.2	getFunctionInformation	38
4.7.3.3	getFunctionInformationFromFile	39
4.7.3.4	getIN_OUT_FLAG	39
4.7.3.5	getParamInformationFromFunctionDifinition	40
4.7.3.6	getPointerLevelFromFUNCTION_INFORMATION_LIST	40
4.7.3.7	new_FUNCTION_INFORMATION	41
4.7.3.8	new_PARAM_INFORMATION	41
4.7.3.9	printFUNCTION_INFORMATION_LIST	42
4.7.3.10	searchFUNCTION_INFORMATION	42
4.8	ANSICInformation/MallocNumber.h	42
4.8.1	説明	43
4.8.2	関数	43
4.8.2.1	generateMallocNumber	43
4.8.2.2	insertMallocNumberHeader	43
4.9	ANSICInformation/MemallocInfo.h	44
4.9.1	説明	45
4.9.2	型定義	45
4.9.2.1	MEMALLOC_INFO	45
4.9.3	関数	45
4.9.3.1	getCallocInformation	45
4.9.3.2	getMallocInformation	45
4.9.3.3	getMallocMaxsize	46
4.9.3.4	getReallocInformation	46
4.9.3.5	memoryAllocationAnarysis	47
4.9.3.6	new_MEMALLOC_INFO	47
4.9.3.7	new_MEMALLOC_INFO_char	47

4.9.3.8	searchSizeof	48
4.10	ANSICInformation/PointerArrayControl.h	48
4.10.1	説明	50
4.10.2	型定義	50
4.10.2.1	ARRAY_OFFSET	50
4.10.3	関数	51
4.10.3.1	ARRAY_OFFSET_LIST_push_back_ref_not_dup	51
4.10.3.2	checkCallFunction	51
4.10.3.3	checkIdentifierPointerArrayLevel	51
4.10.3.4	checkIgnoreASTList	52
4.10.3.5	copyArrayOffsetList	52
4.10.3.6	createArrayExpression	53
4.10.3.7	createValidateVariableArrayExpression	53
4.10.3.8	deleteOFFSET_LIST	54
4.10.3.9	deletePointer	54
4.10.3.10	deletePointerAndArraySynbol	54
4.10.3.11	get_ARRAY_OFFSET_LISTIgnoreASTNAME	55
4.10.3.12	getArgumentOffsetInfo	55
4.10.3.13	getARRAY_OFFSET_LIST	56
4.10.3.14	getArrayOffsetInAnpasandInfo	56
4.10.3.15	getArrayOffsetInIncDecInfo	57
4.10.3.16	getDeclaratorArrayOffset	57
4.10.3.17	getExpressionOffsetInfo	58
4.10.3.18	getOFFSET_LISTFromVariableTable	58
4.10.3.19	getOffsetLevelFromArrayOffset	59
4.10.3.20	getPointerAccessOrIdentifierList	59
4.10.3.21	getPointerArrayOffset	60
4.10.3.22	getSingleExpressionOffsetInfo	60
4.10.3.23	getUpperExpressionRelationNode	61
4.10.3.24	maxOffsetLevelAddressFromArrayOffsetList	62
4.10.3.25	maxOffsetLevelFromArrayOffsetList	62
4.10.3.26	minusArrayOffsetList	63
4.10.3.27	moveArrayOffsetList	63
4.10.3.28	new_ARRAY_OFFSET	63
4.10.3.29	new_ARRAY_OFFSET_char	64
4.10.3.30	OFFSET_LIST_push_back_alloc	65
4.10.3.31	printASTPOINTER_LIST	66
4.10.3.32	searchARRAY_OFFSET_LIST	66
4.10.3.33	searchExpressionOrPointeArrayOrIden	66
4.10.3.34	searchOffsetLevelAddressFromArrayOffsetList	67
4.10.3.35	searchPointerAccessOrIdentifierOrPrimary	67
4.11	ANSICInformation/PreProcess.h	68
4.11.1	説明	68
4.11.2	型定義	68
4.11.2.1	INCLUDE_DATA	68
4.11.3	関数	69
4.11.3.1	addIncludeDataFromFile	69
4.11.3.2	adjustProgramStart	69
4.11.3.3	includeComment	69
4.11.3.4	new_INCLUDE_DATA	69

4.11.3.5	preProcessor	70
4.11.3.6	readIncludeDataFromFile	70
4.12	ANSICInformation/Return_Info.h	70
4.12.1	説明	71
4.12.2	型定義	71
4.12.2.1	RETURN_INFO	71
4.12.3	関数	71
4.12.3.1	new_RETURN_INFO	71
4.13	ANSICInformation/SubEffectCheck.h	71
4.13.1	説明	72
4.13.2	関数	72
4.13.2.1	checkContainSubEffectStatement	72
4.13.2.2	getAssignment_TYPE	72
4.14	ANSICInformation/Synbol.h	73
4.14.1	説明	74
4.14.2	型定義	74
4.14.2.1	STRUCT_TABLE	74
4.14.2.2	TYPEDEF_TABLE	75
4.14.2.3	VARIABLE_TABLE	75
4.14.3	関数	75
4.14.3.1	deletePointer	75
4.14.3.2	deletePointerAndArraySynbol	75
4.14.3.3	find_STRUCT_TABLE_DATA	75
4.14.3.4	getDeclaratorFromAST	76
4.14.3.5	getMemberList	76
4.14.3.6	getParameterData	77
4.14.3.7	getParameterVARIABLE_TABLE_LIST	77
4.14.3.8	getPointerLevelAndArrayLevel	78
4.14.3.9	getPointerLevelAndArrayLevelFromVARIABLE_TABLE	78
4.14.3.10	getSTRUCT_DATA	78
4.14.3.11	getSTRUCT_TABLE_DATA	79
4.14.3.12	getTYPEDEF_TABLE_DATA	79
4.14.3.13	getTYPEDEFfromAST	79
4.14.3.14	getVARIABLE_TABLE_LIST	80
4.14.3.15	new_STRUCT_TABLE	80
4.14.3.16	new_STRUCT_TABLE_with_char	81
4.14.3.17	new_TYPEDEF_TABLE	81
4.14.3.18	new_TYPEDEF_TABLE_with_char	82
4.14.3.19	new_VARIABLE_TABLE	82
4.14.3.20	new_VARIABLE_TABLE_with_char	83
4.14.3.21	printSTRUCT_TABLE_LIST	83
4.14.3.22	printTYPEDEF_TABLE_LIST	83
4.14.3.23	printVARIABLE_TABLE_LIST	84
4.14.3.24	searchVARIABLE_TABLE_LIST	84
4.15	ANSICInformation/Varidate_statement.h	85
4.15.1	説明	88
4.15.2	型定義	88
4.15.2.1	VALIDATE_STATEMENT	88
4.15.2.2	VALIDATE_VARIABLE	88
4.15.3	関数	88

4.15.3.1	ArrayOffsetToValidateStatement	88
4.15.3.2	createCheckUnboundAndUndefineOperationCheck	89
4.15.3.3	createValidateStatementFromIncDecExpr	90
4.15.3.4	createValidateStatement	91
4.15.3.5	createValidateStatementAdderFileEachCheck	93
4.15.3.6	createValidateStatementForFreeAction	93
4.15.3.7	createValidateStatementForMallocAction	94
4.15.3.8	createValidateStatementFromArrayDefine	95
4.15.3.9	createValidateStatementFromPointerDefine	95
4.15.3.10	createViolentFreeOperation	96
4.15.3.11	createZeroDivisionCheck	97
4.15.3.12	fprintProgramDataWithPSIVaridateStatement	97
4.15.3.13	fprintProgramDataWithValidateStatement	98
4.15.3.14	fprintValidateStatement	99
4.15.3.15	fprintValidateStatement_not_assert	100
4.15.3.16	getASTList_FromVALIDATE_STATEMENT_LIST	100
4.15.3.17	getBasisLocationFromAssignmentExpression	101
4.15.3.18	getBasisLocationFromExpression	101
4.15.3.19	getLeftAssignmentInfo	102
4.15.3.20	getNewValidateStatementID	102
4.15.3.21	getRightAssignmentInfo	103
4.15.3.22	getValidate_Variable	104
4.15.3.23	getValidateStatementFromAssignStatement	104
4.15.3.24	getValidateStatementFromCallFunction	105
4.15.3.25	getValidateStatementFromForIteration	105
4.15.3.26	getValidateStatementFromInitializer	106
4.15.3.27	getValidateStatementFromMallocNumber	108
4.15.3.28	getValidateStatementFromPointerOperator	109
4.15.3.29	initVALIDATE_STATEMENT_flag	110
4.15.3.30	new_VALIDATE_STATEMENT	110
4.15.3.31	new_VALIDATE_STATEMENT_char	110
4.15.3.32	new_VALIDATE_VARIABLE	111
4.15.3.33	new_VALIDATE_VARIABLE_with_char	112
4.15.3.34	printProgramDataWithValidateStatement	112
4.15.3.35	printVALIDATE_VARIABLE_LIST	113
4.15.3.36	setValidateVariableFromExprSlicing	113
4.15.3.37	VALIDATE_STATEMENT_LIST_sort_ast	114
4.16	Library/CharStringExtend.h	114
4.16.1	説明	114
4.16.2	関数	115
4.16.2.1	isExpression	115
4.16.2.2	str_extract	115
4.17	Library/CSTLString.h	115
4.17.1	説明	116
4.17.2	関数	116
4.17.2.1	CSTLString_compare_with_char	116
4.17.2.2	CSTLString_delete_tail_str	116
4.17.2.3	CSTLString_ltrim	117
4.17.2.4	CSTLString_printf	117
4.17.2.5	CSTLString_replace_string	117

4.18	Library/FlagDatabase.h	118
4.18.1	説明	118
4.18.2	関数	118
4.18.2.1	getFlagDatabase	118
4.18.2.2	isArrayUnboundCheckMode	119
4.18.2.3	isFreeViolationCheckMode	119
4.18.2.4	isHelpMode	119
4.18.2.5	isProgramSlicingMode	120
4.18.2.6	isUndefineControlCheckMode	120
4.18.2.7	isXmlMode	120
4.18.2.8	isZeroDivitionCheckMode	121
4.19	Library/IdList.h	121
4.19.1	説明	121
4.19.2	関数	121
4.19.2.1	IDLIST_compare_with	121
4.19.2.2	printIDLIST	122
4.19.2.3	SET_STACK_INTToIDLIST	122
4.20	Library/Stack_int.h	122
4.20.1	説明	123
4.20.2	関数	123
4.20.2.1	STACK_INT_at_and_alloc	123
4.20.2.2	STACK_INT_inclement_at	123
4.21	Library/StoreInformation.h	124
4.21.1	説明	124
4.21.2	関数	124
4.21.2.1	getFileName	124
4.21.2.2	setFileName	124
4.22	Main/Help.h	124
4.22.1	説明	125
4.22.2	関数	125
4.22.2.1	viewHelp	125
4.23	ProgramSlicing/DeclarationPSI.h	125
4.23.1	説明	125
4.23.2	関数	126
4.23.2.1	getDeclarationontPSI	126
4.24	ProgramSlicing/ExpressionStatementPSI.h	126
4.24.1	説明	127
4.24.2	関数	127
4.24.2.1	getASI_ARRAY_OFFSET_LIST	127
4.24.2.2	getExpressionStatementPSI	128
4.24.2.3	getInputFunctionPSI	128
4.24.2.4	setARGUMENT_NUMBER	129
4.25	ProgramSlicing/ForStatementPSI.h	130
4.25.1	説明	130
4.25.2	関数	130
4.25.2.1	getForStatementPSI	130
4.26	ProgramSlicing/FunctionPSI.h	131
4.26.1	説明	131
4.26.2	関数	131
4.26.2.1	getFunctionPSI	131

4.26.2.2	getParameterPSI	132
4.27	ProgramSlicing/IfStatementPSI.h	132
4.27.1	説明	133
4.27.2	関数	133
4.27.2.1	getIfStatementPSI	133
4.28	ProgramSlicing/JumpStatementPSI.h	134
4.28.1	説明	134
4.28.2	関数	134
4.28.2.1	getJumpStatementPSI	134
4.29	ProgramSlicing/LabeledStatementPSI.h	135
4.29.1	説明	135
4.29.2	関数	136
4.29.2.1	getLabeledStatementPSI	136
4.30	ProgramSlicing/ProgramSlicing.h	136
4.30.1	説明	137
4.30.2	関数	137
4.30.2.1	createDD_list	137
4.30.2.2	createDD_list_in_argument	137
4.30.2.3	createDD_list_in_Function	138
4.30.2.4	createDD_list_in_global	138
4.30.2.5	createDD_listAll	139
4.30.2.6	createStatementNodeList	139
4.30.2.7	startStaticSlicing	140
4.30.2.8	staticSlicing	140
4.31	ProgramSlicing/ProgramSlicingInformation.h	141
4.31.1	説明	141
4.31.2	関数	142
4.31.2.1	getFunctionGlobalVariable	142
4.31.2.2	getVariableDeclarationFromEXPR_SLICING_LIST	142
4.31.2.3	initExprSlicingListFlag	142
4.31.2.4	new_DD_INFORMATION	143
4.31.2.5	new_EXPR_SLICING	143
4.31.2.6	print_EXPR_SLICING_LIST	144
4.31.2.7	print_tree_EXPR_SLICING_LIST	144
4.31.2.8	registerIncDecVariable	145
4.31.2.9	searchDD	145
4.31.2.10	searchDeclarationDD	146
4.31.2.11	searchFunctionPSI	146
4.31.2.12	setGlobalVariable	147
4.32	ProgramSlicing/ReturnStatementPSI.h	147
4.32.1	説明	148
4.32.2	関数	148
4.32.2.1	getReturnStatementPSI	148
4.33	ProgramSlicing/SwitchStatementPSI.h	149
4.33.1	説明	149
4.33.2	関数	149
4.33.2.1	getSwicthStatementPSI	149
4.34	ProgramSlicing/WhileStatementPSI.h	150
4.34.1	説明	150
4.34.2	関数	151

4.34.2.1	getWhileStatementPSI	151
----------	--------------------------------	-----

Chapter 1

データ構造索引

1.1 データ構造

データ構造の説明です。

abstract_syntax_tree	7
array_offset	8
division_information	9
for_information	9
freemem_info	10
function_information	10
include_data	11
memory_allocation_info	11
param_information	11
return_info	12
struct_table	12
typedef_table	13
validate_statement	13
validate_variable	14
variable_table	15

Chapter 2

ファイル索引

2.1 ファイル一覧

これはファイル一覧です。

ANSICInformation/ANSIC_CODE.h (このファイルは、ANSIC のソースコードを省略させるためのマクロ文が含まれている)	17
ANSICInformation/AST.h (このファイルは、構文解析によって通された C 言語プログラムから、抽象構文木 (AST) を生成させるためのものである。また生成させるほかに、実際に抽象構文木からソースファイルを生成しなおしたり、ある部分のノードから文字列を生成させたりといったことができる。)	19
ANSICInformation/DivitionDeclarator.h (これは簡潔化のために変数定義を分割させるためのものである。 例: int a,b = 1; int a; int b = 1;)	28
ANSICInformation/DivitionInformation.h (このファイルは、除算・剰余算を検出するために使用するのに必要な情報を取得するための命令が含まれている。)	29
ANSICInformation/ForInformation.h (このファイルは、for ループに関する情報を集めるための命令が含まれている)	32
ANSICInformation/FreeMemInfo.h (このファイルは、メモリ解放関係の関数から、メモリ解放関係の情報を取得する命令が含まれている。具体的には、C 言語プログラム上にある free 関数から、どの変数が解放されているかどうかについて取得する。)	35
ANSICInformation/FunctionInformation.h (このファイルは関数に関する情報を取得するための命令が含まれている。)	37
ANSICInformation/MallocNumber.h (これはどの malloc 用識別番号を付加する関数を生成するための命令が含まれている。)	42
ANSICInformation/MemallocInfo.h (このファイルはメモリ確保関係の関数から、メモリ確保関係に関する情報を格納するための命令が含まれている。具体的には、C 言語プログラム上にある、malloc・calloc・realloc などといった関数から、確保している sizeof の型・realloc 使用時に対象としている変数・確保しているサイズを取得する。)	44

ANSICInformation/ PointerArrayControl.h (このファイルは、C 言語プログラム上の複雑な配列参照および直接参照による演算を各次元の配列オフセット情報として格納するための命令が含まれている。各次元の配列オフセット情報とは各次元において、どの部分を指しているかという式に関する情報のことである。)	48
ANSICInformation/ PreProcess.h (このファイルは構文解析を通せるように、Gcc で C 言語にプリプロセス (前処理) をさせるための命令が含まれている。このファイルでできることとしては、 <code>::include</code> をコメントアウトすることで省いて、プリプロセスを掛けることができる。)	68
ANSICInformation/ Return_Info.h (これはリターン命令に関する情報を取得するための命令が含まれている。)	70
ANSICInformation/ SubEffectCheck.h (このファイルには副作用式のチェックする関数や代入式のタイプを求める命令が含まれている。)	71
ANSICInformation/ Synbol.h (このファイルは、構文解析によって生成された抽象構文木 (AST) から、変数・typedef テーブル・構造体テーブルを生成させるための命令が含まれている。とくに、typedef テーブルの生成は、C 言語の構文解析では必須な処理である。)	73
ANSICInformation/ Varidate_statement.h (これは C 言語プログラム上から、不具合を検証するための検証式や検証用を使用する変数などを追加するための命令が含まれている。)	85
ANSICInformation/ y.tab.h	??
Library/ CharStringExtend.h (このファイルは、string.h にないような処理を実現させるための命令が含まれている。具体的には文字列からある場所からある場所までを抽出させるといったことができる。)	114
Library/ CSTLString.h (このファイルは、CSTL ライブラリを用いた文字列型 CSTLString を生成したり、それに関する操作を行うための命令が含まれている。)	115
Library/ FlagDatabase.h (このファイルは、本体に関するフラグ設定を格納するための命令が含まれている。たとえば、xml として出力するのかなどといった設定が含まれている。)	118
Library/ IdList.h (このファイルは、この変数は C 言語プログラム上のどのブロックに宣言しているかどうかを確認するための情報を生成するための命令が含まれている。)	121
Library/ Stack_int.h (このファイルは整数スタックに関する操作をするための命令が含まれている。このファイルで使用される主な用途として、C 言語ソースファイルにおけるブロックレベルを設定するのに使用される。)	122
Library/ StoreInformation.h (これはファイル名などといったプログラム内で共通する部分を確保するための命令が含まれている。主にファイル名などを確保する。)	124
Main/ Help.h (このファイルはヘルプ出力関係の命令が含まれている。)	124
ProgramSlicing/ DeclarationPSI.h (このファイルは Declaration Statement Program Slicing Information の略である。宣言式から、プログラムスライシングに関する情報を抽出するための命令が含まれている。)	125

ProgramSlicing/ ExpressionStatementPSI.h (このファイルは Expression Statement Program Slicing Information の略である。式全般から、プログラムスライシングに関する情報を抽出するための命令が含まれている。)	126
ProgramSlicing/ ForStatementPSI.h (このファイルは For Statement Program Slicing Information の略である。for 文から、プログラムスライシングに関する情報を抽出するための命令が含まれている。)	130
ProgramSlicing/ FunctionPSI.h (このファイルは Function Program Slicing Information の略である。関数定義から、プログラムスライシングに関する情報を抽出するための命令が含まれている。)	131
ProgramSlicing/ IfStatementPSI.h (このファイルは If Statement Program Slicing Information の略である。if 文もしくは、if と else 文から、プログラムスライシングに関する情報を抽出するための命令が含まれている。)	132
ProgramSlicing/ JumpStatementPSI.h (このファイルは Jump Statement Program Slicing Information の略である。GOTO、continue、break 文から、プログラムスライシングに関する情報を抽出するための命令が含まれている。)	134
ProgramSlicing/ LabeledStatementPSI.h (このファイルは Labeled Statement Program Slicing Information の略である。goto ラベル文、case ラベル文、default ラベル文から、プログラムスライシングに関する情報を抽出するための命令が含まれている。)	135
ProgramSlicing/ ProgramSlicing.h (このファイルはプログラムスライシングを行うための命令が含まれている。指定した識別子名および行数を入れることで、それに基づいてプログラムスライシングを行う。)	136
ProgramSlicing/ ProgramSlicingInformation.h (このファイルはプログラムスライシングに関する情報を取り扱う命令が含まれている。)	141
ProgramSlicing/ ReturnStatementPSI.h (このファイルは Return Statement Program Slicing Information の略である。return 文から、プログラムスライシングに関する情報を抽出するための命令が含まれている。)	147
ProgramSlicing/ SwitchStatementPSI.h (このファイルは Switch Statement Program Slicing Information の略である。switch 文から、プログラムスライシングに関する情報を抽出するための命令が含まれている。)	149
ProgramSlicing/ WhileStatementPSI.h (このファイルは While Statement Program Slicing Information の略である。while 文から、プログラムスライシングに関する情報を抽出するための命令が含まれている。)	150

Chapter 3

データ構造

3.1 構造体 `abstract_syntax_tree`

```
#include <AST.h>
```

変数

- `CSTLString * content`
ノード名
- `int line`
ノードの内容
- `CSTLString * return_type`
対象のノードの行数
- `int block_level`
返却値のタイプ (*void*、*int*、*double* などが入る)
- `int block_id`
ブロックの階層レベル (グローバルの場合は0、何らかの関数内であれば1となる)
- `IDLIST * idlist`
各々のブロックを識別するためのもの。0から順に設定される
- `ASTList * children`
ブロックの階層および識別子を同時に識別するためのもの。変数定義に関して確認するのに用いる

3.1.1 説明

C 言語プログラムの抽象構文木に関する情報が含まれる。

この構造体の説明は次のファイルから生成されました:

- ANSICInformation/AST.h

3.2 構造体 array_offset

```
#include <PointerArrayControl.h>
```

変数

- **AST * target_statement**
変数名
- **AST * variable_address**
ターゲットの *statement*(ここでは *expression_statement* などが入る)
- **int pointer_level**
この変数名が指している AST アドレス
- **int array_level**
この変数のポインタレベル
- **int anpasand_flag**
この変数の配列レベル
- **int inc_dec_flag**
この変数はアンパサンドを挟んでいるかどうかのフラグ 1 : 挟んでいる 0 : 挟んでいない
- **OFFSET_LIST * offset_list**
この変数はインクリメントおよびデクリメントが含まれているかどうかのフラグ 0 : 含んでいない 1 : インクリメントが含まれている 2 : デクリメントが含まれている

3.2.1 説明

配列やポインタの各次元のオフセット関係を格納するための構造体である。配列オフセットと呼ばれる。

この構造体の説明は次のファイルから生成されました:

- ANSICInformation/PointerArrayControl.h

3.3 構造体 `divition_information`

```
#include <DivitionInformation.h>
```

変数

- `int type`
対象の式
- `CSTLString * statement`
除算か剰余かどうかのタイプ 0 : 除算式 1 : 剰余式
- `ARRAY_OFFSET_LIST * identifiers`
除算および剰余以下の式

3.3.1 説明

除算および剰余算に関する情報を格納する。

この構造体の説明は次のファイルから生成されました:

- `ANSICInformation/DivitionInformation.h`

3.4 構造体 `for_information`

```
#include <ForInformation.h>
```

変数

- `AST * init_expression`
この `for` 文自体への `AST` ノード
- `AST * continue_condition`
初期式への `AST` ノード
- `AST * inc_dec_expression`
継続式への `AST` ノード
- `AST * statement`
増分式への `AST` ノード

3.4.1 説明

for 文の初期式・継続式・増分式および for 文内全体の式へのノードが格納される。これは、検証式付加ツールで for 文を while 文に使用される。

この構造体の説明は次のファイルから生成されました:

- ANSICInformation/**ForInformation.h**

3.5 構造体 freemem_info

```
#include <FreeMemInfo.h>
```

3.5.1 説明

メモリ確保関係に関係する変数が含まれる。これは、メモリ解放関数の挙動に合わせて検証式を追加するために用いる。

この構造体の説明は次のファイルから生成されました:

- ANSICInformation/**FreeMemInfo.h**

3.6 構造体 function_information

```
#include <FunctionInformation.h>
```

変数

- CSTLString * **return_type**
関数定義の AST ノードへのアドレス
- CSTLString * **function_name**
返却値の型 (*const int* などと表記される)
- PARAM_INFORMATION_LIST * **param_information_list**
関数名 (**func* などと表記される)

3.6.1 説明

関数に関する情報。関数からの検証式の生成などに用いる。

この構造体の説明は次のファイルから生成されました:

- ANSICInformation/**FunctionInformation.h**

3.7 構造体 `include_data`

```
#include <PreProcess.h>
```

変数

- `int line`
include ファイルの名前

3.7.1 説明

インクルードファイルに関する情報が格納される。これは、検証式付加時にインクルードファイルを付加するのに使用する。

この構造体の説明は次のファイルから生成されました:

- ANSICInformation/**PreProcess.h**

3.8 構造体 `memory_allocation_info`

```
#include <MemallocInfo.h>
```

変数

- `ARRAY_OFFSET_LIST * realloc_target`
sizeof の型名
- `CSTLString * size`
realloc 時のターゲット変数の配列オフセットリスト

3.8.1 説明

メモリ割り当てに関する情報を格納するための構造体。

この構造体の説明は次のファイルから生成されました:

- ANSICInformation/**MemallocInfo.h**

3.9 構造体 `param_information`

```
#include <FunctionInformation.h>
```

変数

- `CSTLString * param_name`
パラメータの型
- `int array_level`
パラメータの名前
- `int pointer_level`
配列のレベル
- `int in_out_flag`
ポインタのレベル

3.9.1 説明

引数に関する情報

この構造体の説明は次のファイルから生成されました:

- `ANSICInformation/FunctionInformation.h`

3.10 構造体 `return_info`

```
#include <Return_Info.h>
```

変数

- `ARRAY_OFFSET_LIST * return_array_offset_list`
リターン命令自体へのノードへのアドレス

3.10.1 説明

リターン命令に関する情報が含まれている。

この構造体の説明は次のファイルから生成されました:

- `ANSICInformation/Return_Info.h`

3.11 構造体 `struct_table`

```
#include <Symbol.h>
```


変数

- CStdString * **type**
行数
- CStdString * **struct_name**
構造体のタイプ *union*: 共同体 *struct*: 構造体
- VARIABLE_TABLE_LIST * **member_list**
構造体の名前

3.11.1 説明

構造体に関する情報であり、プログラム中の構造体の識別するのに用いられる。
この構造体の説明は次のファイルから生成されました:

- ANSICInformation/Synbol.h

3.12 構造体 typedef_table

```
#include <Synbol.h>
```

変数

- CStdString * **change_type**
対象の型定義

3.12.1 説明

型定義に関する情報で、BISON による構文解析時の型定義の認識に用いられる。
この構造体の説明は次のファイルから生成されました:

- ANSICInformation/Synbol.h

3.13 構造体 validate_statement

```
#include <Varidate_statement.h>
```

変数

- **int check_or_modify**
この検証式の識別 ID(どの順序でこの検証式を入れていくかを確認するための ID)
- **int used**
検証式をチェックするタイプか、プログラムを元に編集するタイプかを判断するフラグ。0 : チェックするタイプ、1 : 編集するタイプ
- **CSTLString * statement**
この検証式は使用しているかどうかのフラグ 1:使用 0:未使用
- **AST * target_statement**
この検証式の内容

3.13.1 説明

実際に検証式として挿入するための情報である。

この構造体の説明は次のファイルから生成されました:

- ANSICInformation/**Varidate_statement.h**

3.14 構造体 validate_variable

```
#include <Varidate_statement.h>
```

変数

- **int enable_start**
この検証用の変数が使用しているかどうかのフラグ 1:使用 0:未使用
- **int enable_end**
この変数の有効範囲 (ブロックでの) の開始行数を示す
- **int declaration_location**
この変数の有効範囲 (ブロックでの) の終了行数を示す
- **int block_level**
この変数を宣言する行数を示す
- **int block_id**
この変数を宣言するブロックレベルを示す (0 ならグローバル変数・1 なら関数内と示す)

- `CSTLString * type`
この変数を宣言するブロック *ID* を示す (同ブロックレベルで別のブロックを識別するための番号)
- `CSTLString * variable_name`
この変数の型
- `CSTLString * target_variable_name`
この変数名
- `int offset_level`
この変数の検証対象となる変数名

3.14.1 説明

ポインタや配列変数に対する検証用の変数リストを作成するための構造体である。
この構造体の説明は次のファイルから生成されました:

- `ANSICInformation/Varidate_statement.h`

3.15 構造体 `variable_table`

```
#include <Symbol.h>
```

変数

- `int enable_end`
この変数が有効である開始の行数
- `AST * declaration_location_address`
この変数が有効である終了の行数
- `int block_level`
この変数の宣言をしている *AST* ノードへのアドレス
- `int block_id`
この変数が宣言しているブロックレベル (グローバル変数なら 0、関数内のローカル変数なら 1 となる)
- `IDLIST * idlist`
この変数が宣言している各々のブロックの識別子

- **CSTLString * type**

この変数が宣言しているブロックレベルおよび識別子の両方に関する情報で、変数スコープを識別するのに使用される。

- **CSTLString * variable_name**

この変数の型

- **AST * initializer**

この変数の名前

3.15.1 説明

プログラム中の変数に関する情報であり、検証式生成時に変数を識別するのに用いられる。

この構造体の説明は次のファイルから生成されました:

- **ANSICInformation/Synbol.h**

Chapter 4

ファイル

4.1 ANSICInformation/ANSIC_CODE.h

このファイルは、ANSIC のソースコードを省略させるためのマクロ文が含まれている

```
#include "AST.h"
```

関数

- **AST * ASTLIST_ITERATOR_1** (AST *target)
- **AST * ASTLIST_ITERATOR_2** (AST *target)
- **AST * ASTLIST_ITERATOR_3** (AST *target)
- **AST * ASTLIST_ITERATOR_4** (AST *target)
- **AST * ASTLIST_ITERATOR_5** (AST *target)
- **AST * ASTLIST_ITERATOR_6** (AST *target)
- **AST * ASTLIST_ITERATOR_7** (AST *target)

4.1.1 説明

このファイルは、ANSIC のソースコードを省略させるためのマクロ文が含まれている

作者

faithnh

4.1.2 関数

4.1.2.1 **AST* ASTLIST_ITERATOR_1** (AST * *target*) [inline]

対象の AST ノードの 1 番目の子ノードの取得する。

引数

<i>target</i>	対象の AST ノード
---------------	-------------

戻り値

対象の AST ノードの 1 番目の子ノードが返される。

4.1.2.2 **AST* ASTLIST_ITERATOR_2(AST* *target*)** [inline]

対象の AST ノードの 2 番目の子ノードの取得する。

引数

<i>target</i>	対象の AST ノード
---------------	-------------

戻り値

対象の AST ノードの 2 番目の子ノードが返される。

4.1.2.3 **AST* ASTLIST_ITERATOR_3(AST* *target*)** [inline]

対象の AST ノードの 3 番目の子ノードの取得する。

引数

<i>target</i>	対象の AST ノード
---------------	-------------

戻り値

対象の AST ノードの 3 番目の子ノードが返される。

4.1.2.4 **AST* ASTLIST_ITERATOR_4(AST* *target*)** [inline]

対象の AST ノードの 4 番目の子ノードの取得する。

引数

<i>target</i>	対象の AST ノード
---------------	-------------

戻り値

対象の AST ノードの 4 番目の子ノードが返される。

4.1.2.5 **AST* ASTLIST_ITERATOR_5(AST* *target*)** [inline]

対象の AST ノードの 5 番目の子ノードの取得する。

引数

<i>target</i>	対象の AST ノード
---------------	-------------

戻り値

対象の AST ノードの 5 番目の子ノードが返される。

4.1.2.6 AST*ASTLIST_ITERATOR_6 (AST * *target*) [inline]

対象の AST ノードの 6 番目の子ノードの取得する。

引数

<i>target</i>	対象の AST ノード
---------------	-------------

戻り値

対象の AST ノードの 6 番目の子ノードが返される。

4.1.2.7 AST*ASTLIST_ITERATOR_7 (AST * *target*) [inline]

対象の AST ノードの 7 番目の子ノードの取得する。

引数

<i>target</i>	対象の AST ノード
---------------	-------------

戻り値

対象の AST ノードの 7 番目の子ノードが返される。

4.2 ANSICInformation/AST.h

このファイルは、構文解析によって通された C 言語プログラムから、抽象構文木 (AST) を生成させるためのものである。また生成させるほかに、実際に抽象構文木からソースファイルを生成しなおしたり、ある部分のノードから文字列を生成させたりといったことができる。

```
#include <cstl/string.h>
#include <cstl/list.h>
#include <stdio.h>
#include "../Library/CSTLString.h"
#include "../Library/IdList.h"
```

データ構造

- struct **abstract_syntax_tree**

型定義

- typedef struct **abstract_syntax_tree** AST

関数

- **AST * new_AST** (char *new_name, char *new_content, int new_line)
- **AST * same_new_AST** (char *new_name, int new_line)
- void **traverseAST** (AST *root, int tab_level)
- void **traverseASTwithXML** (AST *root, int tab_level)
- void **fprintDataFromAST** (FILE *output, AST *root, int *line)
- void **printDataFromAST** (AST *root, int *line)
- void **getStringFromAST** (CSTLString *output, AST *root)
- void **getStringFromASTEnableExcept** (CSTLString *output, AST *root, int num, char except_list[][256])
- void **deleteAST** (AST *root)
- void **push_back_childrenAST** (AST *parent, AST *child)
- void **multi_push_back_childrenAST** (AST *parent, int num,...)
- void **printTargetASTNode** (AST *root, char *target, int traverse_flag, int xml_flag)
- int **getArgumentString** (CSTLString *output, AST *call_function, int arg_num)
- int **getArgumentStringEnableExcept** (CSTLString *output, AST *call_function, int arg_num, int num, char except_list[][256])
- int **getArgumentAST** (AST **output, AST *call_function, int arg_num)
- **AST * getASTwithString** (AST *root, char *name, int depth)
- void **getStringReplaceASTtoString** (CSTLString *output, AST *root, AST *target, char *replace_string)
- void **setASTReturntype** (AST *root)
- void **setASTBlocklevelAndId** (AST *root)
- int **findASTAddress** (AST *root, AST *target)
- void **fprintfStatement** (FILE *output, AST *root, int *line, int output_subnode_num, AST **out_block_start, AST **out_block_end)

4.2.1 説明

このファイルは、構文解析によって通された C 言語プログラムから、抽象構文木 (AST) を生成させるためのものである。また生成させるほかに、実際に抽象構文木からソースファイルを生成しなおしたり、ある部分のノードから文字列を生成させたりといったことができる。

作者

faithnh

4.2.2 型定義

4.2.2.1 typedef struct abstract_syntax_tree AST

C 言語プログラムの抽象構文木に関する情報が含まれる。

4.2.3 関数

4.2.3.1 void deleteAST (AST * root)

指定した AST ノード以下を全て開放させる。

引数

<i>root</i>	指定した AST ノード
-------------	--------------

戻り値

なし

4.2.3.2 int findASTAddress (AST * root, AST * target)

探す対象の AST ノード root から、指定した AST ノードのアドレス target が存在するかどうか調べる。みつければ、1 を返し、そうでなければ 0 を返す。

引数

<i>root</i>	探す対象の AST ノード
<i>target</i>	指定した AST ノードのアドレス

戻り値

指定した AST ノード target を見つけたら 1 を返し、そうでなければ 0 を返す。

4.2.3.3 void fprintfDataFromAST (FILE * output, AST * root, int * line)

指定された AST ノード以下の内容を、指定されたファイルに対して出力させる

引数

<i>output</i>	出力先のファイル構造体
<i>root</i>	指定されたノード名
<i>line</i>	出力に利用する行数

戻り値

なし

4.2.3.4 void fprintfStatement (FILE * *output*, AST * *root*, int * *line*, int *output_subnode_num*, AST ** *out_block_start*, AST ** *out_block_end*)

指定されたノードに対して、statement の直前まで出力させる。また、出力対象となっているノード番号が来るまでは出力の対象としない。ブロックとして表記されている場合は始まりのブロックや終わりのブロックへの AST ノードのアドレスを *out_block_start* や *out_block_end* に 設定する。そうでなければ、NULL で設定される。

引数

<i>output</i>	出力先のファイル構造体
<i>root</i>	指定された AST ノード
<i>line</i>	出力に利用する行数
<i>output_subnode_num</i>	出力対象となるノード番号 1 ~ n (1 に指定すると、すべての子ノードが出力の対象となる)
<i>out_block_start</i>	ブロックとして表記されている場合は、始まりのブロックへの AST ノードのアドレスを返却する
<i>out_block_end</i>	ブロックとして表記されている場合は、終わりのブロックへの AST ノードのアドレスを返却する。

戻り値

なし

4.2.3.5 int getArgumentsAST (AST ** *output*, AST * *call_function*, int *arg_num*)

指定された *call_function* に相当する関数から、任意の引数目へのノードを取得する。このとき、指定されたノード名は *call_function* でなければならない。

引数

<i>output</i>	取得する対象のノード
<i>call_function</i>	名前が <i>call_function</i> である AST ノード
<i>arg_num</i>	何番目の引数

戻り値

成功したかどうかを示す。成功したならば 1、失敗した場合は 0 を返す。

4.2.3.6 int getArgumentsString (CSTLString * *output*, AST * *call_function*, int *arg_num*)

指定された *call_function* に相当する関数から、任意の引数目の情報を取得する。このとき、指定されたノード名は *call_function* でなければならない。

引数

<i>output</i>	出力される引数の内容
<i>call_function</i>	名前が <i>call_function</i> である AST ノード
<i>arg_num</i>	何番目の引数

戻り値

成功したかどうかを示す。成功したならば 1、失敗した場合は 0 を返す。

4.2.3.7 int getArgumentsEnableExcept (CSTLString * output, AST * call_function, int arg_num, int num, char except_list[256])

getArgumentsEnableExcept の拡張版。出力対象にしない文字列を除いた引数の情報をすべて出力させる。

引数

<i>output</i>	出力される引数の内容
<i>call_function</i>	名前が <i>call_function</i> である AST ノード
<i>arg_num</i>	何番目の引数
<i>num</i>	除外したい AST ノードリストの数
<i>except_list</i>	除外したい AST ノード

戻り値

成功したかどうかを示す。成功したならば 1、失敗した場合は 0 を返す。

4.2.3.8 AST* getASTwithString (AST * root, char * name, int depth)

指定した AST ノード *root* から、指定した名前 *name* の AST ノードを探す。見つけた AST ノードへのアドレスを返す。

引数

<i>root</i>	指定した AST ノード
<i>name</i>	指定した名前
<i>depth</i>	探索する子ノードの深さ (無限にする場合は -1 を指定、子ノードは探索しない場合は 0 にする)

戻り値

見つけた AST ノードへのアドレスを返す。

4.2.3.9 void getStringFromAST (CSTLString * output, AST * root)

指定した AST ノード以下の内容を出力対象の文字列データに出力させる。

引数

<i>output</i>	出力対象の文字列データ
<i>root</i>	新しいノード名

戻り値

なし

4.2.3.10 void getStringFromASTEnableExcept (CSTLString * output, AST * root, int num, char except_list[][256])

getStringFromAST の拡張版。出力対象から除外した AST ノード名を指定できる。

引数

<i>output</i>	出力対象の文字列データ
<i>root</i>	新しいノード名
<i>num</i>	除外したい AST ノードリストの数
<i>except_list</i>	除外したい AST ノード

戻り値

なし

4.2.3.11 void getStringReplaceASTtoString (CSTLString * output, AST * root, AST * target, char * replace_string)

指定した AST ノード root に対しての情報を文字列 output を出力させる。そのとき、AST ノードの target のアドレス値と一致するノードを見つけた（アドレスとして全く同じ AST ノードを見つけた）場合、そのノードだけ replace_string に変換して出力させる。

引数

<i>root</i>	指定した AST ノード
<i>output</i>	出力する情報
<i>target</i>	変換対象の AST ノードのアドレス
<i>replace_string</i>	変換先の文字列

戻り値

なし

4.2.3.12 void multi_push_back_childrenAST (AST * parent, int num, ...)

指定された親 AST ノードに任意の子 AST ノードを追加する。

引数

<i>parent</i>	指定された親 AST ノード
<i>num</i>	追加する子 AST ノードの数
<i>...</i>	追加する任意の子 AST ノード

戻り値

なし

4.2.3.13 AST* new_AST (char * new_name, char * new_content, int new_line)

新しい AST ノードを生成する。

引数

<i>new_name</i>	新しいノード名
<i>new_content</i>	新しい内容
<i>new_line</i>	新しい行数

戻り値

新しく生成された AST ノードへのアドレスが返される。

4.2.3.14 void printDataFromAST (AST * root, int * line)

指定した AST ノード以下の内容出力させる。AST ノードの行数にしたがって出力される。

引数

<i>root</i>	新しいノード名
<i>line</i>	指定する行数 (基本的に 1 を入力する)

戻り値

なし

4.2.3.15 void printTargetASTNode (AST * root, char * target, int traverse_flag, int xml_flag)

指定された親 AST ノードから、指定したノード名より下位についての情報をすべて表示させる。。また、traverse_flag を 1 にすることで、指定したノード名についてのすべての情報をツリー構造で表示させることもできる。さらに、xml_flag

を 1 にすることで、ツリー構造で表示させる内容が XML として出力できるようになる (traverse_flag が 1 になっていることが前提である)。

引数

<i>root</i>	指定された親 AST ノード
<i>target</i>	指定した文字列
<i>traverse_flag</i>	指定したノード名以下をツリー構造で表示させるかどうかのフラグ。1 なら表示させ、0 なら表示させない
<i>xml_flag</i>	XML として出力するかどうかのフラグ。1 なら XML として出力させる

戻り値

なし

指定された親 AST ノードから、指定したノード名より下位についての情報をすべて表示させる。また、traverse_flag を 1 にすることで、指定したノード名についてのすべての情報をツリー構造で表示させることもできる。さらに、xml_flag を 1 にすることで、ツリー構造で表示させる内容が XML として出力できるようになる (traverse_flag が 1 になっていることが前提である)。

引数

<i>root</i>	指定された親 AST ノード
<i>target</i>	指定した文字列
<i>traverse_flag</i>	指定したノード名以下をツリー構造で表示させるかどうかのフラグ。1 なら表示させ、0 なら表示させない
<i>xml_flag</i>	XML として出力するかどうかのフラグ。1 なら XML として出力させる

戻り値

なし

4.2.3.16 void push_back_childrenAST (AST * parent, AST * child)

指定された親 AST ノードに指定された子 AST ノードを追加する。

引数

<i>parent</i>	指定された親 AST ノード
<i>child</i>	指定された子 AST ノード

戻り値

なし

4.2.3.17 AST* same_new_AST (char * new_name, int new_line)

名前と内容が同じである AST ノードを生成する。

引数

<i>new_name</i>	新しいノード名
<i>new_line</i>	新しい行数

戻り値

新しく生成された AST ノードへのアドレスが返される。

4.2.3.18 void setASTBlocklevelAndId (AST * root)

対象の AST ノードに、ブロック ID およびブロックレベルを付加する。

引数

<i>root</i>	対象の AST ノード
-------------	-------------

戻り値

なし

4.2.3.19 void setASTReturnType (AST * root)

対象の AST ノードの root 以下にあるに、関数の返却値のタイプを指定する。

引数

<i>root</i>	対象の AST ノード
-------------	-------------

戻り値

なし

4.2.3.20 void traverseAST (AST * root, int tab_level)

AST のノードをたどり、root ノード以下のノードの名前・内容・行数を表示させる。また、下位レベルのノードはタブを挿入し表示する。例えば、1 レベル下位のノードはタブが 1 つ挿入した上で表示される。

引数

<i>root</i>	新しいノード名
<i>tab_level</i>	タブレベル(この数分タブが挿入される)

戻り値
なし

4.2.3.21 void traverseASTwithXML (AST * root, int tab_level)

AST のノードをたどり、root ノード以下のノードの名前・内容・行数を XML 形式で出力する。(デバッグ用) また、下位レベルのノードはタブを挿入し出力する。例えば、1 レベル下位のノードはタブが 1 つ挿入した上でタグが出力される。XML については次の形式で出力する (a はルートノード、a_sub はリーフノードであるとする) <leaf name="a_sub" content="+" line="1">

引数

root	新しいノード名
tab_level	タブレベル (この数分タブが挿入される)

戻り値
なし

4.3 ANSICInformation/DivitionDeclarator.h

これは簡潔化のために変数定義を分割させるためのものである。 例 : int a,b = 1; int a; int b = 1;
#include "AST.h"
#include "Synbol.h"

関数

- void **OutputSourceAfterDivitionDeclarator** (FILE *output, AST *programAST, VARIABLE_TABLE_LIST *variable_table_list)

4.3.1 説明

これは簡潔化のために変数定義を分割させるためのものである。 例 : int a,b = 1; int a; int b = 1;

作者
faithnh

4.3.2 関数

4.3.2.1 void OutputSourceAfterDivitionDeclarator (FILE * *output*, AST * *programAST*,
VARIABLE_TABLE_LIST * *variable_table_list*)

変数定義を分割させた状態で出力させる。 例：int a,b = 1; int a; int b = 1;

引数

<i>output</i>	出力先のファイル構造体
<i>programAST</i>	プログラムに対する AST
<i>variable_-table_list</i>	変数テーブル

戻り値

なし

4.4 ANSICInformation/DivitionInformation.h

このファイルは、除算・剰余算を検出するために使用するのに必要な情報を取得するための命令が含まれている。

```
#include <cstl/list.h>
#include "AST.h"
#include "Synbol.h"
#include "../Library/CSTLString.h"
#include "PointerArrayControl.h"
#include "FunctionInformation.h"
```

データ構造

- struct **divition_information**

型定義

- typedef struct **divition_information** **DIVITION_INFORMATION**

関数

- **DIVITION_INFORMATION** * **new_DIVITION_INFORMATION** (AST **target_-expression*, int *type*, CSTLString **statement*, ARRAY_OFFSET_LIST **identifiers*)
- **DIVITION_INFORMATION** * **new_DIVITION_INFORMATION_char** (AST **target_expression*, int *type*, char **statement*, ARRAY_OFFSET_LIST **identifiers*)

- void **getDIVISION_INFORMATION_LIST** (AST *expression_ast, FUNCTION_INFORMATION_LIST *function_information_list, VARIABLE_TABLE_LIST *vtlist, DIVISION_INFORMATION_LIST *division_information_list, AST *target_expression, ASTPOINTER_LIST *ignore_ast_list)
- void **printDIVISION_INFORMATION_LIST** (DIVISION_INFORMATION_LIST *division_information_list)

4.4.1 説明

このファイルは、除算・剰余算を検出するために使用するのに必要な情報を取得するための命令が含まれている。

作者

faithnh

4.4.2 型定義

4.4.2.1 typedef struct divition_information DIVISION_INFORMATION

除算および剰余算に関する情報を格納する。

4.4.3 関数

4.4.3.1 void getDIVISION_INFORMATION_LIST (AST * expression_ast, FUNCTION_INFORMATION_LIST * function_information_list, VARIABLE_TABLE_LIST * vtlist, DIVISION_INFORMATION_LIST * division_information_list, AST * target_expression, ASTPOINTER_LIST * ignore_ast_list)

指定された式から、div_expr および mod_expr を見つけ、それ以下の式を除算および剰余残に関する情報に格納する

引数

<i>expression_ast</i>	指定された式への AST ノードのアドレス
<i>function_information_list</i>	関数に関する情報のリスト
<i>vtlist</i>	変数テーブルリスト
<i>division_information_list</i>	除算および剰余算に関する情報を格納するためのリスト
<i>target_expression</i>	対象の式への AST ノードのアドレス
<i>ignore_ast_list</i>	重複防止のために使用する AST ノードへのリスト

戻り値

なし

**4.4.3.2 DIVITION_INFORMATION* new_DIVITION_INFORMATION (AST *
target_expression, int type, CSTLString * statement, ARRAY_OFFSET_LIST * identifiers
)**

DIVITION_INFORMATION のリスト 除算および剰余算に関する情報を生成する。

引数

<i>target_ - expression</i>	対象の式への AST ノードのアドレス
<i>type</i>	除算か剰余かどうかのタイプ 0 : 除算式 1 : 剰余式
<i>statement</i>	除算および剰余以下の式
<i>identifiers</i>	式内の識別子一覧

戻り値

生成された情報へのアドレスが返される

**4.4.3.3 DIVITION_INFORMATION* new_DIVITION_INFORMATION_char (AST *
target_expression, int type, char * statement, ARRAY_OFFSET_LIST * identifiers)**

除算および剰余算に関する情報を生成する。

引数

<i>target_ - expression</i>	対象の式への AST ノードのアドレス
<i>type</i>	除算か剰余かどうかのタイプ 0 : 除算式 1 : 剰余式
<i>statement</i>	除算および剰余以下の式
<i>identifiers</i>	式内の識別子一覧

戻り値

生成された情報へのアドレスが返される

**4.4.3.4 void printDIVITION_INFORMATION_LIST (DIVITION_INFORMATION_LIST *
division_information_list)**

除算および剰余算に関する情報のリストの内容を出力させる

引数

<i>division_</i> <i>information_</i> <i>list</i>	除算および剰余算に関する情報のリスト
--	--------------------

戻り値

なし

4.5 ANSICInformation/ForInformation.h

このファイルは、for ループに関する情報を集めるための命令が含まれている

```
#include "AST.h"
#include <cstl/list.h>
```

データ構造

- struct **for_information**

型定義

- typedef struct **for_information** **FOR_INFORMATION**

関数

- **FOR_INFORMATION * new_FOR_INFORMATION** (AST *target_expression, AST *init_expression, AST *continue_condition, AST *inc_dec_expression, AST *statement)
- void **print_FOR_INFORMATION_LIST** (FOR_INFORMATION_LIST *for_information_list)
- **FOR_INFORMATION * searchFOR_INFORMATION_FromAST** (FOR_INFORMATION_LIST *for_information_list, AST *target_ast)
- void **getFOR_INFORMATION_LIST** (FOR_INFORMATION_LIST *for_information_list, AST *for_iteration_ast)

4.5.1 説明

このファイルは、for ループに関する情報を集めるための命令が含まれている

作者

faithnh

4.5.2 型定義

4.5.2.1 typedef struct for_information FOR_INFORMATION

for 文の初期式・継続式・増分式および for 文内全体の式へのノードが格納される。
これは、検証式付加ツールで for 文を while 文に使用される。

4.5.3 関数

4.5.3.1 void getFOR_INFORMATION_LIST (FOR_INFORMATION_LIST * for_information_list, AST * for_iteration_ast)

for_iteration の AST ノードから、for 文に関する情報のリストを取得する。

引数

<i>for_information_list</i>	for 文に関する情報のリスト
<i>for_iteration_ast</i>	for 文に関する情報のリスト

戻り値

なし

4.5.3.2 FOR_INFORMATION* new_FOR_INFORMATION (AST * target_expression, AST * init_expression, AST * continue_condition, AST * inc_dec_expression, AST * statement)

これは、FOR_INFORMATION の実装を確かめる for 文に関する情報 FOR_INFORMATION を生成する。

引数

<i>target_expression</i>	この for 文自体の情報
<i>init_expression</i>	初期式
<i>continue_condition</i>	継続式
<i>inc_dec_expression</i>	増分式 AST *statement for 文内の全体の式を指す

戻り値

生成された FOR_INFORMATION へのアドレスを返す。

for 文に関する情報 FOR_INFORMATION を生成する。

引数

<i>target_- expression</i>	この for 文自体の情報
<i>init_- expression</i>	初期式
<i>continue_- condition</i>	継続式
<i>inc_dec_- expression</i>	増分式
<i>statement</i>	for 文内の全体の式を指す

戻り値

生成された FOR_INFORMATION へのアドレスを返す。

4.5.3.3 void print_FOR_INFORMATION_LIST (FOR_INFORMATION_LIST * *for_information_list*)

for 文に関する情報のリスト *for_information_list* を出力する。

引数

<i>for_- information_- list</i>	for 文に関する情報のリスト
---	-----------------

戻り値

なし

for 文に関する情報のリスト *for_information_list* を出力する

引数

<i>for_- information_- list</i>	for 文に関する情報のリスト
---	-----------------

戻り値

なし

4.5.3.4 FOR_INFORMATION* searchFOR_INFORMATION_FromAST (FOR_INFORMATION_LIST * *for_information_list*, AST * *target_ast*)

for 文に関する情報のリスト *for_information_list* から、指定した AST ノードに対応したものを出力させる。

引数

<i>for_information_list</i>	検索対象の for 文に関する情報のリスト
<i>target_ast</i>	検索する AST

戻り値

マッチしたものが見つければ、それに対応したもののアドレスを出力させる。見つからない場合は NULL を返す。

4.6 ANSICInformation/FreeMemInfo.h

このファイルは、メモリ解放関係の関数から、メモリ解放関係の情報を取得する命令が含まれている。具体的には、C 言語プログラム上にある free 関数から、どの変数が解放されているかどうかについて取得する。

```
#include "PointerArrayControl.h"
#include "FunctionInformation.h"
```

データ構造

- struct **freemem_info**

型定義

- typedef struct **freemem_info** **FREEMEMINFO**

関数

- **FREEMEMINFO * new_FREEMEMINFO** (ARRAY_OFFSET_LIST *free_variable)
- void **getFreememInfo** (**FREEMEMINFO** **freememinfo, **AST** *call_function, **FUNCTION_INFORMATION_LIST** *function_information_list, **VARIABLE_TABLE_LIST** *vtlist, **AST** *target_statement)
- void **printFREEMEMINFO** (**FREEMEMINFO** *freememinfo)

4.6.1 説明

このファイルは、メモリ解放関係の関数から、メモリ解放関係の情報を取得する命令が含まれている。具体的には、C 言語プログラム上にある free 関数から、どの変数が解放されているかどうかについて取得する。

作者

faithnh

4.6.2 型定義

4.6.2.1 typedef struct freemem_info FREEMEMINFO

メモリ確保関係に関係する変数が含まれる。これは、メモリ解放関数の挙動に合わせて検証式を追加するために用いる。

4.6.3 関数

4.6.3.1 void getFreememInfo (FREEMEMINFO ** *freememinfo*, AST * *call_function*, FUNCTION_INFORMATION_LIST * *function_information_list*, VARIABLE_TABLE_LIST * *vtlist*, AST * *target_statement*)

対象の関数への AST ノードから free 関数を見つけ、それに関する情報を取得する。

引数

<i>freememinfo</i>	取得先のメモリ解放関係の情報
<i>call_function</i>	対象の関数への AST ノード
<i>function_information_list</i>	関数に関する情報のリスト
<i>vtlist</i>	変数テーブルの情報
<i>target_statement</i>	対象の関数が位置している式への AST ノード

戻り値

なし

4.6.3.2 FREEMEMINFO* new_FREEMEMINFO (ARRAY_OFFSET_LIST * *free_variable*)

新しいメモリ解放関係の情報を生成する。

引数

<i>free_variable</i>	新しい解放先の変数に関するオフセットリスト
----------------------	-----------------------

戻り値

新しく生成されたメモリ解放関係の情報へのアドレスを返す。

4.6.3.3 void printFREEMEMINFO (FREEMEMINFO * *freememinfo*)

指定したメモリ解放関係の情報を出力する。

引数

<i>freememinfo</i>	出力するメモリ解放関係の情報
--------------------	----------------

戻り値

なし

4.7 ANSICInformation/FunctionInformation.h

このファイルは関数に関する情報を取得するための命令が含まれている。

```
#include <cstl/list.h>
#include "../Library/CSTLString.h"
#include "AST.h"
```

データ構造

- struct **param_information**
- struct **function_information**

型定義

- typedef struct **param_information** **PARAM_INFORMATION**
- typedef struct **function_information** **FUNCTION_INFORMATION**

関数

- **FUNCTION_INFORMATION * new_FUNCTION_INFORMATION** (AST *function_node, CSTLString *return_type, CSTLString *function_name, PARAM_INFORMATION_LIST *param_information_list)
- **PARAM_INFORMATION * new_PARAM_INFORMATION** (CSTLString *param_type, CSTLString *param_name, int array_level, int pointer_level, int in_out_flag)
- void **getFunctionInformation** (FUNCTION_INFORMATION_LIST *function_information_list, AST *root)
- void **getFunctionInformationFromFile** (FUNCTION_INFORMATION_LIST *function_information_list, char *file_name)
- void **deleteParameterDefine** (CSTLString *target_string)
- void **printFUNCTION_INFORMATION_LIST** (FUNCTION_INFORMATION_LIST *function_information_list)
- **FUNCTION_INFORMATION * searchFUNCTION_INFORMATION** (CSTLString *target_function_name, FUNCTION_INFORMATION_LIST *function_information_list)

- **int getPointerLevelFromFUNCTION_INFORMATION_LIST** (CSTLString *target_function_name, FUNCTION_INFORMATION_LIST *function_information_list)
- **void getParamInformationFromFunctionDefinition** (AST *param_info_node, PARAM_INFORMATION_LIST *param_information_list)
- **int getIN_OUT_FLAG** (CSTLString *param_type, int pointer_level, int array_level)

4.7.1 説明

このファイルは関数に関する情報を取得するための命令が含まれている。

作者

faithnh

4.7.2 型定義

4.7.2.1 typedef struct function_information FUNCTION_INFORMATION

関数に関する情報。関数からの検証式の生成などに用いる。

4.7.2.2 typedef struct param_information PARAM_INFORMATION

引数に関する情報

4.7.3 関数

4.7.3.1 void deleteParameterDefine (CSTLString * target_string)

指定した関数の定義自体から、パラメータ定義などを削除する。

引数

<i>target_string</i>	対象の関数の定義自体の文字列
----------------------	----------------

戻り値

なし

4.7.3.2 void getFunctionInformation (FUNCTION_INFORMATION_LIST * function_information_list, AST * root)

指定したプログラムの AST ノードから関数定義を探し、それに基づいて関数に関する情報を生成し、それらのリスト function_information_list に登録する。

引数

<i>function_information_list</i>	登録先の関数に関する情報のリスト
<i>root</i>	指定したプログラムの AST ノード

戻り値

なし

4.7.3.3 void getFunctionInformationFromFile (FUNCTION_INFORMATION_LIST * *function_information_list*, char * *file_name*)

指定したファイル名 *file_name* で定義された関数に関する情報を関数に関する情報のリスト *function_information_list* に設定する。

引数

<i>function_information_list</i>	登録先の関数に関する情報のリスト
<i>file_name</i>	指定したファイル名

戻り値

なし

4.7.3.4 int getIN_OUT_FLAG (CSTLString * *param_type*, int *pointer_level*, int *array_level*)

パラメータのポインタレベル・配列レベルおよび型名から、このパラメータは入力型か出力型か判定する。

引数

<i>param_type</i>	パラメータの型名
<i>pointer_level</i>	ポインタのレベル
<i>array_level</i>	配列のレベル

戻り値

入力型ならば 1、出力型ならば 0 を返す。

4.7.3.5 void getParamInformationFromFunctionDifinition (AST * *param_info_node*, PARAM_INFORMATION_LIST * *param_information_list*)

パラメータリストを示すノード *param_info_node* から、パラメータの情報を取得し、*param_information_list* に登録する。

引数

<i>param_info_node</i>	パラメータリストを示すノード
<i>param_information_list</i>	登録先のパラメータ情報リスト

戻り値

なし

4.7.3.6 int getPointerLevelFromFUNCTION_INFORMATION_LIST (CSTLString * *target_function_name*, FUNCTION_INFORMATION_LIST * *function_information_list*)

指定した関数名が関数に関する情報リストから探し、ポインタレベルを返す。

引数

<i>target_function_name</i>	指定した関数名
<i>function_information_list</i>	検索対象の関数に関する情報リスト

戻り値

見つかった場合はその関数のポインタレベルを返す、そうでない場合は-1を返す。

指定した関数名が関数に関する情報リストから探し、ポインタレベルを返す。

引数

<i>target_function_name</i>	指定した関数名
<i>function_information_list</i>	検索対象の関数に関する情報リスト

戻り値

見つかった場合はその関数のポインタレベルを返す。そうでない場合は-1を返す。

4.7.3.7 FUNCTION_INFORMATION* new_FUNCTION_INFORMATION (AST
* *function_node*, CSTLString * *return_type*, CSTLString * *function_name*,
PARAM_INFORMATION_LIST * *param_information_list*)

関数に関する情報を生成する。

引数

<i>function_node</i>	対象の関数へのノード
<i>return_type</i>	返却値のタイプ
<i>function_name</i>	関数名
<i>param_information_list</i>	パラメータに関する情報

戻り値

生成された関数に関する情報へのアドレスを返す。

4.7.3.8 PARAM_INFORMATION* new_PARAM_INFORMATION (CSTLString *
param_type, CSTLString * *param_name*, int *array_level*, int *pointer_level*, int *in_out_flag*
)

パラメータに関する情報を生成する。

引数

<i>param_type</i>	パラメータの型
<i>param_name</i>	パラメータの名前
<i>array_level</i>	配列のレベル
<i>pointer_level</i>	ポインタのレベル
<i>in_out_flag</i>	入力型か出力型かの判定 1 : 入力 0 : 出力 2 : 入出力

戻り値

生成されたパラメータに関する情報へのアドレスを返す。

4.7.3.9 void printFUNCTION_INFORMATION_LIST (FUNCTION_INFORMATION_LIST * *function_information_list*)

関数に関する情報リストの内容を出力させる。

引数

<i>function_information_list</i>	出力対象の関数に関する情報リスト
----------------------------------	------------------

戻り値

なし

4.7.3.10 FUNCTION_INFORMATION* searchFUNCTION_INFORMATION (CString * *target_function_name*, FUNCTION_INFORMATION_LIST * *function_information_list*)

関数に関する情報リストから、指定した関数名を探し、それに関する構造体へのアドレスを返す。

引数

<i>target_function_name</i>	指定した関数名
<i>function_information_list</i>	検索対象の関数に関する情報リスト

戻り値

見つかった場合はその関数に関する構造体へのアドレスを返す。みつからなければ、NULL を返す。

4.8 ANSICInformation/MallocNumber.h

これはどの malloc 用識別番号を付加する関数を生成するための命令が含まれている。

```
#include "../Library/CSTLString.h"
#include "PreProcess.h"
```

関数

- void **generateMallocNumber** (CString **target_directory*)
- void **insertMallocNumberHeader** (INCLUDE_LIST **include_list*)

4.8.1 説明

これはどの malloc 用識別番号を付加する関数を生成するための命令が含まれている。

作者

faithnh

4.8.2 関数

4.8.2.1 void generateMallocNumber (CString * *target_directory*)

malloc 用識別番号付加関数をカレントディレクトリ上に生成させる。

引数

<i>target_directory</i>	作成するディレクトリ先
-------------------------	-------------

戻り値

なし

malloc 用識別番号関数をカレントディレクトリ上に生成させる。

引数

<i>target_directory</i>	作成するディレクトリ先
-------------------------	-------------

戻り値

なし

4.8.2.2 void insertMallocNumberHeader (INCLUDE_LIST * *include_list*)

インクルード情報に malloc 用識別番号付加関数が含まれているヘッダーファイルを追加させる。。

引数

<i>include_list</i>	インクルード情報へのリスト
---------------------	---------------

戻り値

なし

インクルード情報に malloc 用識別番号付加関数が含まれているヘッダーファイルを追加させる。

引数

<code>include_list</code>	インクルード情報へのリスト
---------------------------	---------------

戻り値

なし

4.9 ANSICInformation/MemallocInfo.h

このファイルはメモリ確保関係の関数から、メモリ確保関係に関する情報を格納するための命令が含まれている。具体的には、C 言語プログラム上にある、`malloc`・`calloc`・`realloc` などといった関数から、確保している `sizeof` の型・`realloc` 使用時で対象としている変数・確保しているサイズを取得する。

```
#include "PointerArrayControl.h"
#include "AST.h"
#include "Synbol.h"
```

データ構造

- struct **memory_allocation_info**

型定義

- typedef struct **memory_allocation_info** **MEMALLOC_INFO**

関数

- **MEMALLOC_INFO * new_MEMALLOC_INFO_char** (char *sizeof_type, ARRAY_OFFSET_LIST *realloc_target, char *size)
- **MEMALLOC_INFO * new_MEMALLOC_INFO** (CSTLString *sizeof_type, ARRAY_OFFSET_LIST *realloc_target, CSTLString *size)
- **MEMALLOC_INFO * memoryAllocationAnarysis** (AST *root, VARIABLE_TABLE_LIST *vtlist)
- void **getMallocInformation** (AST *root, CSTLString *sizeof_type, CSTLString *constant)
- void **getCallocInformation** (AST *root, CSTLString *sizeof_type, CSTLString *constant)
- void **getReallocInformation** (AST *root, VARIABLE_TABLE_LIST *vtlist, CSTLString *sizeof_type, CSTLString *constant, ARRAY_OFFSET_LIST *realloc_target)
- int **searchSizeof** (CSTLString *sizeof_type, AST *root)
- void **getMallocMaxsize** (CSTLString *constant, AST *root)

4.9.1 説明

このファイルはメモリ確保関係の関数から、メモリ確保関係に関する情報を格納するための命令が含まれている。具体的には、C 言語プログラム上にある、`malloc`・`calloc`・`realloc` などといった関数から、確保している `sizeof` の型・`realloc` 使用時で対象としている変数・確保しているサイズを取得する。

作者

faithnh

4.9.2 型定義

4.9.2.1 typedef struct memory_allocation_info MEMALLOC_INFO

メモリ割り当てに関する情報を格納するための構造体。

4.9.3 関数

4.9.3.1 void getCallocInformation (AST * root, CSTLString * sizeof_type, CSTLString * constant)

`calloc` 関数に関する情報を取得する。

引数

<i>root</i>	
<i>sizeof_type</i>	<code>sizeof</code> での型
<i>constant</i>	<code>sizeof</code> 以外での式 (すなわち確保している型に対するサイズに相当する)

戻り値

なし

4.9.3.2 void getMallocInformation (AST * root, CSTLString * sizeof_type, CSTLString * constant)

`malloc` 関数に関する情報を取得する

引数

<i>root</i>	
<i>sizeof_type</i>	<code>sizeof</code> での型
<i>constant</i>	<code>sizeof</code> 以外での式 (すなわち確保している型に対するサイズに相当する)

戻り値

なし

malloc 関数に関する情報を取得する。

引数

<i>root</i>	
<i>sizeof_type</i>	sizeof での型
<i>constant</i>	sizeof 以外での式 (すなわち確保している型に対するサイズに相当する)

戻り値

なし

4.9.3.3 void getMallocMaxsize (CSTLString * *constant*, AST * *root*)

malloc などの関数内の式から、確保したサイズを示す式を取得する。これは、sizeof(型) を 1 に書き直しながら、式を出力させることで行っている。

引数

<i>constant</i>	取得する sizeof 以外での式 (すなわち確保している型に対するサイズに相当する)
<i>root</i>	malloc などの関数内の式への AST ノード

戻り値

なし

4.9.3.4 void getReallocInformation (AST * *root*, VARIABLE_TABLE_LIST * *vtlist*, CSTLString * *sizeof_type*, CSTLString * *constant*, ARRAY_OFFSET_LIST * *realloc_target*)

realloc 関数に関する情報を取得する。

引数

<i>root</i>	対象の AST ノード
<i>vtlist</i>	対象のプログラムの変数リスト
<i>sizeof_type</i>	sizeof での型
<i>constant</i>	sizeof 以外での式 (すなわち確保している型に対するサイズに相当する)
<i>realloc_target</i>	realloc が対象とするオフセット情報

戻り値

なし

4.9.3.5 MEMALLOC_INFO* memoryAllocationAnarysis (AST * *root*, VARIABLE_TABLE_LIST * *vtlist*)

指定された AST ノードからメモリ確保関係の関数に関する情報を取得する。

引数

<i>root</i>	指定された AST ノード
<i>vtlist</i>	realloc で realloc のターゲット情報のオフセット情報を取得するのに必要なプログラムの変数リスト

戻り値

メモリ確保関係の構造体へのアドレスを返す。

4.9.3.6 MEMALLOC_INFO* new_MEMALLOC_INFO (CSTLString * *sizeof_type*, ARRAY_OFFSET_LIST * *realloc_target*, CSTLString * *size*)

メモリ割り当てに関する情報を格納するための構造体のデータを生成させる。

引数

<i>sizeof_type</i>	sizeof の型名
<i>realloc_target</i>	realloc 時のターゲットタイプ
<i>size</i>	malloc 時のサイズ

戻り値

メモリ割り当てに関する情報を格納するための構造体へのアドレスを返す。

4.9.3.7 MEMALLOC_INFO* new_MEMALLOC_INFO_char (char * *sizeof_type*, ARRAY_OFFSET_LIST * *realloc_target*, char * *size*)

メモリ割り当てに関する情報を格納するための構造体のデータを生成させる。

引数

<i>sizeof_type</i>	sizeof の型名
<i>realloc_target</i>	realloc 時のターゲットタイプ
<i>size</i>	malloc 時のサイズ

戻り値

メモリ割り当てに関する情報を格納するための構造体へのアドレスを返す。

4.9.3.8 int searchSizeof (CSTLString * sizeof_type, AST * root)

指定した式の AST に対して、sizeof を探索し、その型名を sizeof_type で取得する。もし、式に異なる 2 種類の sizeof 定義があれば、探索フラグは失敗する。また、見つからなければ、sizeof_type は何も指定していないままの状態である。

引数

<i>sizeof_type</i>	出力する sizeof の型名
<i>root</i>	探索対象の式への AST ノード

戻り値

なし

4.10 ANSICInformation/PointerArrayControl.h

このファイルは、C 言語プログラム上の複雑な配列参照および直接参照による演算を各次元の配列オフセット情報として格納するための命令が含まれている。各次元の配列オフセット情報とは各次元において、どの部分を指しているかという式に関する情報のことである。

```
#include <cstl/list.h>
#include "../Library/CSTLString.h"
#include "AST.h"
#include "Synbol.h"
#include "FunctionInformation.h"
```

データ構造

- struct **array_offset**

型定義

- typedef struct **array_offset** **ARRAY_OFFSET**

関数

- **ARRAY_OFFSET * new_ARRAY_OFFSET_char** (char *variable_name, AST *target_statement, AST *variable_address, OFFSET_LIST *offset_list, int pointer_level, int array_level, int anpasand_flag, int inc_dec_flag)

- **ARRAY_OFFSET * new_ARRAY_OFFSET** (CSTLString *variable_name, AST *target_statement, AST *variable_address, OFFSET_LIST *offset_list, int pointer_level, int array_level, int anpasand_flag, int inc_dec_flag)
- void **OFFSET_LIST_push_back_alloc** (OFFSET_LIST *offset_list, char *string)
- int **getPointerArrayOffset** (AST *root, FUNCTION_INFORMATION_LIST *function_information_list, VARIABLE_TABLE_LIST *vtlist, int offset_level, ASTPOINTER_LIST *ignore_ast_list, **ARRAY_OFFSET **array_offset**, AST *target_statement, int anpasand_flag, int inc_dec_flag)
- void **getARRAY_OFFSET_LIST** (AST *root, **ARRAY_OFFSET_LIST *array_offset_list**, FUNCTION_INFORMATION_LIST *function_information_list, VARIABLE_TABLE_LIST *vtlist, ASTPOINTER_LIST *ignore_ast_list, AST *target_statement)
- void **get_ARRAY_OFFSET_LISTIgnoreASTNAME** (AST *root, **ARRAY_OFFSET_LIST *array_offset_list**, FUNCTION_INFORMATION_LIST *function_information_list, VARIABLE_TABLE_LIST *vtlist, ASTPOINTER_LIST *ignore_ast_list, AST *target_statement, CSTLString *ignore_ast_name)
- void **getArrayOffsetInAnpasandInfo** (AST *root, FUNCTION_INFORMATION_LIST *function_information_list, VARIABLE_TABLE_LIST *vtlist, **ARRAY_OFFSET_LIST *array_offset_list**, ASTPOINTER_LIST *ignore_ast_list, AST *target_statement)
- void **getArrayOffsetInIncDecInfo** (AST *root, FUNCTION_INFORMATION_LIST *function_information_list, VARIABLE_TABLE_LIST *vtlist, **ARRAY_OFFSET_LIST *array_offset_list**, ASTPOINTER_LIST *ignore_ast_list, AST *target_statement, int inc_dec_flag)
- int **getUpperExpressionRelationNode** (AST *target, AST *root, **AST **output**, **AST **output2**)
- void **deletePointerAndArraySymbol** (CSTLString *target)
- void **deletePointer** (CSTLString *target)
- int **searchExpressionOrPointerArrayOrIden** (AST *root, **AST **output**)
- int **searchPointerAccessOrIdentifierOrPrimary** (AST *root, **AST **output**)
- void **getPointerAccessOrIdentifierList** (AST *root, **AST ***output**, int *getSize)
- int **checkIdentifierPointerArrayLevel** (AST *identifier, int offset_level, VARIABLE_TABLE_LIST *variable_table_list, ASTPOINTER_LIST *ignore_ast_list, int *pointer_level, int *array_level)
- void **checkCallFunction** (AST *call_function, int offset_level, FUNCTION_INFORMATION_LIST *function_information_list)
- int **checkIgnoreASTList** (AST *ast_data, ASTPOINTER_LIST *ignore_ast_list)
- void **getOFFSET_LISTFromVariableTable** (OFFSET_LIST *offset_list, **VARIABLE_TABLE *variable_table**)
- void **deleteOFFSET_LIST** (OFFSET_LIST *offset_list)
- int **getOffsetLevelFromArrayOffset** (**ARRAY_OFFSET *array_offset**)
- **ARRAY_OFFSET * maxOffsetLevelAddressFromArrayOffsetList** (**ARRAY_OFFSET_LIST *array_offset_list**)
- **ARRAY_OFFSET * searchOffsetLevelAddressFromArrayOffsetList** (AST *root, **ARRAY_OFFSET_LIST *array_offset_list**, int pointer_level, int array_level)
- int **maxOffsetLevelFromArrayOffsetList** (**ARRAY_OFFSET_LIST *array_offset_list**)

- void **getExpressionOffsetInfo** (AST *expression, FUNCTION_INFORMATION_LIST *function_information_list, VARIABLE_TABLE_LIST *vtlist, ASTPOINTER_LIST *ignore_ast_list, ARRAY_OFFSET_LIST *array_offset_list, AST *target_expression, int *switch_mode, int allow_subeffect)
- void **getArgumentOffsetInfo** (AST *argument, FUNCTION_INFORMATION_LIST *function_information_list, VARIABLE_TABLE_LIST *vtlist, ASTPOINTER_LIST *ignore_ast_list, ARRAY_OFFSET_LIST *array_offset_list, AST *target_expression, int *switch_mode)
- void **getSingleExpressionOffsetInfo** (AST *expression, FUNCTION_INFORMATION_LIST *function_information_list, VARIABLE_TABLE_LIST *vtlist, ASTPOINTER_LIST *ignore_ast_list, ARRAY_OFFSET_LIST *array_offset_list, AST *target_expression, int *switch_mode)
- void **createArrayExpression** (CSTLString *output, ARRAY_OFFSET *array_offset, int output_level)
- int **createValidateVariableArrayExpression** (CSTLString *output, ARRAY_OFFSET *array_offset)
- void **moveArrayOffsetList** (ARRAY_OFFSET_LIST *fromlist, ARRAY_OFFSET_LIST *tolist, int move_start)
- void **copyArrayOffsetList** (ARRAY_OFFSET_LIST *fromlist, ARRAY_OFFSET_LIST *tolist, int move_start)
- void **getDeclaratorArrayOffset** (ARRAY_OFFSET_LIST *declarator_array_offset_list, AST *declarator_expression, AST *target_expression, VARIABLE_TABLE_LIST *vtlist)
- ARRAY_OFFSET * **searchARRAY_OFFSET_LIST** (ARRAY_OFFSET_LIST *array_offset_list, CSTLString *name)
- void **minusArrayOffsetList** (ARRAY_OFFSET_LIST *target_array_offset_list, ARRAY_OFFSET_LIST *delete_array_offset_list)
- void **ARRAY_OFFSET_LIST_push_back_ref_not_dup** (ARRAY_OFFSET_LIST *target_array_offset_list, ARRAY_OFFSET *array_offset)
- void **printASTPOINTER_LIST** (ASTPOINTER_LIST *astpointer_list)

4.10.1 説明

このファイルは、C 言語プログラム上の複雑な配列参照および直接参照による演算を各次元の配列オフセット情報として格納するための命令が含まれている。各次元の配列オフセット情報とは各次元において、どの部分を指しているかという式に関する情報のことである。

作者

faithnh

4.10.2 型定義

4.10.2.1 typedef struct array_offset ARRAY_OFFSET

配列やポインタの各次元のオフセット関係を格納するための構造体である。配列オフセットと呼ばれる。

4.10.3 関数

4.10.3.1 void ARRAY_OFFSET_LIST_push_back_ref_not_dup (ARRAY_OFFSET_LIST *
target_array_offset_list, ARRAY_OFFSET * array_offset)

配列オフセットリストに配列オフセット情報を追加する。ただし、変数名が重複するのであれば、追加しない。

引数

<i>target_</i> - <i>array_</i> - <i>offset_list</i>	追加先の配列オフセットリスト
<i>array_offset</i> (p. 8)	追加する配列オフセット情報

戻り値

なし

4.10.3.2 void checkCallFunction (AST * call_function, int offset_level,
FUNCTION_INFORMATION_LIST * function_information_list)

関数呼び出しを示す AST ノードが、登録されている関数に関する情報に含まれているかどうかを調べ、もし、その関数のポインタレベルがオフセットレベルと一致した場合は、エラーを出力し、強制終了させる。

引数

<i>call_</i> - <i>function</i>	関数呼び出しを示す AST ノード
<i>offset_level</i>	オフセットレベル
<i>function_</i> - <i>information_</i> - <i>list</i>	関数に関する情報のリスト

戻り値

なし

4.10.3.3 int checkIdentifierPointerArrayLevel (AST * identifier, int offset_level,
VARIABLE_TABLE_LIST * variable_table_list, ASTPOINTER_LIST * ignore_ast_list, int *
pointer_level, int * array_level)

識別子の名前を一致する変数を変数リストから探す。このとき、一致する変数を調べたら、ポインタと配列の次元も調べ、オフセットレベル以上であれば、見つけたことになり、1 を返す。そうでなければ、0 を返す。また、ignore_ast_list は無視する IDENTIFIER の AST のアドレスリストを見つけるたびに登録される。

もし、`ignore_ast_list` に登録されているノードなら、それは無視される。また、見つけるのに成功した場合、その該当する変数の配列レベルやポインタのレベルも返す。

引数

<i>identifier</i>	識別子の名前
<i>offset_level</i>	オフセットレベル
<i>variable_table_list</i>	変数テーブルリスト
<i>ignore_ast_list</i>	ポインタでの位置が検証済みである、IDENTIFIER を無視するための AST のアドレスリスト
<i>pointer_level</i>	変数リストから見つけた変数のポインタレベル
<i>array_level</i>	変数リストから見つけた変数の配列レベル

戻り値

識別子の名前およびオフセットレベルが条件を満たしていれば 1 を返し、そうでなければ 0 を返す。

4.10.3.4 int checkIgnoreASTList (AST * ast_data, ASTPOINTER_LIST * ignore_ast_list)

指定した AST ノード `ast_data` が AST アドレスリスト `ignore_ast_list` に存在するかどうかを調べる。存在する場合は 1 をかえす。存在しない場合は、`ast_data` のアドレスを `ignore_ast_list` に追加した上、0 を返す。

引数

<i>ast_data</i>	指定した AST ノード
<i>ignore_ast_list</i>	調べる対象の AST アドレスリスト

戻り値

存在する場合は 1 を返し、そうでない場合は 0 を返す

4.10.3.5 void copyArrayOffsetList (ARRAY_OFFSET_LIST * fromlist, ARRAY_OFFSET_LIST * tolist, int move_start)

配列オフセットリスト `fromlist` 内の `move_start` 以降のデータをすべて、もう一方の配列オフセットリスト `tolist` にコピーさせる。

引数

<i>fromlist</i>	コピー元の配列オフセットリスト
<i>tolist</i>	コピー先の配列オフセットリスト
<i>move_start</i>	移動させたいデータの位置 (先頭から 0 番目とする)

戻り値

なし

4.10.3.6 void createArrayExpression (CSTLString * *output*, ARRAY_OFFSET * *array_offset*, int *output_level*)

配列オフセット情報から、任意の次元までの配列式を生成する。

引数

<i>output</i>	配列式を生成される文字列
<i>array_offset</i> (p. 8)	対象の配列オフセット
<i>output_level</i>	出力したい次元(このとき、配列オフセットを超える値を入れた場合は、配列オフセットが次元までの配列式を出力する)

戻り値

なし

4.10.3.7 int createValidateVariableArrayExpression (CSTLString * *output*, ARRAY_OFFSET * *array_offset*)

配列オフセット情報から、検証用変数に使われる配列式を生成し、オフセットレベルを返す。

引数

<i>output</i>	生成先の文字列
<i>array_offset</i> (p. 8)	対象の配列オフセット

戻り値

配列オフセット情報から生成されたオフセットレベルを返す

配列オフセット情報から、検証用変数に使われる配列式を生成し、オフセットレベルを返す。

引数

<i>output</i>	生成先の文字列
<i>array_offset</i> (p. 8)	対象の配列オフセット

戻り値

配列オフセット情報から生成されたオフセットレベルを返す。

4.10.3.8 void deleteOFFSET_LIST (OFFSET_LIST * *offset_list*)

オフセットリスト *offset_list* の中身を完全に解放させる。

引数

<i>offset_list</i>	
--------------------	--

戻り値

なし

オフセットリスト *offset_list* の中身を完全に解放させる。

引数

<i>offset_list</i>	解放させる <i>offset_list</i>
--------------------	--------------------------

戻り値

なし

4.10.3.9 void deletePointer (CString * *target*)

変数名からポインタを示す記号のみ全て削除する。

引数

<i>target</i>	変更対象の変数名
---------------	----------

戻り値

なし

4.10.3.10 void deletePointerAndArraySynbol (CString * *target*)

変数名から配列およびポインタを示す記号を全て削除する。

引数

<i>target</i>	変更対象の変数名
---------------	----------

戻り値

なし

```
4.10.3.11 void get_ARRAY_OFFSET_LISTIgnoreASTNAME ( AST * root, ARRAY_OFFSET_LIST
* array_offset_list, FUNCTION_INFORMATION_LIST * function_information_list,
VARIABLE_TABLE_LIST * vtlist, ASTPOINTER_LIST * ignore_ast_list, AST *
target_statement, CSTLString * ignore_ast_name )
```

ポインタおよび配列変数の各次元のオフセットリストを取得する。また、無視をする対象のノードを設定可能である。

引数

<i>root</i>	オフセットリストに該当する AST ノード
<i>array_offset_list</i>	ポインタおよび配列のオフセット情報のリスト
<i>function_information_list</i>	関数に関する情報リスト
<i>vtlist</i>	検証対象の式をマークするための変数リスト
<i>ignore_ast_list</i>	ポインタでの位置が検証済みである、IDENTIFIER を無視するための AST のアドレスリスト
<i>target_statement</i>	検証式の対象となるステートメント
<i>ignore_ast_name</i>	無視をする A S T 名

戻り値

なし

```
4.10.3.12 void getArgumentOffsetInfo ( AST * argument, FUNCTION_INFORMATION_LIST *
function_information_list, VARIABLE_TABLE_LIST * vtlist, ASTPOINTER_LIST *
ignore_ast_list, ARRAY_OFFSET_LIST * array_offset_list, AST * target_expression,
int * switch_mode )
```

指定した引数から、必要なオフセット情報を取得する。

引数

<i>argument</i>	指定した引数に関する AST ノード
<i>function_information_list</i>	関数に関する情報のリスト
<i>vtlist</i>	検証対象の式をマークするための変数リスト
<i>ignore_ast_list</i>	ポインタでの位置が検証済みである、IDENTIFIER を無視するための AST のアドレスリスト
<i>array_offset_list</i>	各ポインタおよび配列ごとのオフセットのリスト
<i>target_expression</i>	この左辺式の上位に位置する AST ノード

<i>switch_mode</i>	直接アクセスおよび配列アクセスを探すか、IDENTIFIER を探すかどうかのスイッチフラグ 0 : 両方さがす 1 : direct_ref や array_access のみ探す
--------------------	---

戻り値

なし

4.10.3.13 void `getARRAY_OFFSET_LIST (AST * root, ARRAY_OFFSET_LIST * array_offset_list, FUNCTION_INFORMATION_LIST * function_information_list, VARIABLE_TABLE_LIST * vtlist, ASTPOINTER_LIST * ignore_ast_list, AST * target_statement)`

ポインタおよび配列変数の各次元のオフセットリストを取得する。

引数

<i>root</i>	オフセットリストに該当する AST ノード
<i>array_offset_list</i>	ポインタおよび配列のオフセット情報のリスト
<i>function_information_list</i>	関数に関する情報リスト
<i>vtlist</i>	検証対象の式をマークするための変数リスト
<i>ignore_ast_list</i>	ポインタでの位置が検証済みである、IDENTIFIER を無視するための AST のアドレスリスト
<i>target_statement</i>	検証式の対象となるステートメント

戻り値

なし

4.10.3.14 void `getArrayOffsetInAnpasandInfo (AST * root, FUNCTION_INFORMATION_LIST * function_information_list, VARIABLE_TABLE_LIST * vtlist, ARRAY_OFFSET_LIST * array_offset_list, ASTPOINTER_LIST * ignore_ast_list, AST * target_statement)`

address_ref であるノード内を探索し、それに対するアドレス参照や、識別子を探し出し、見つけたら配列オフセットリスト array_offset_list へ入れる。

引数

<i>root</i>	右辺式に関する AST ノード
<i>function_information_list</i>	関数に関する情報のリスト
<i>vtlist</i>	メモリ確保情報を取得するのに必要なプログラム変数リスト
<i>array_offset_list</i>	左辺式上にあるポインタ参照に対するオフセットリスト

<i>ignore_ast_list</i>	同じ位置のポインタが来ても無視するためのリスト
<i>target_statement</i>	この計算式を属している AST ノードへのアドレス (基本的に expression_statement であるノードが入る)

戻り値

なし

4.10.3.15 void getArrayOffsetIncDecInfo (AST * root, FUNCTION_INFORMATION_LIST * function_information_list, VARIABLE_TABLE_LIST * vtlist, ARRAY_OFFSET_LIST * array_offset_list, ASTPOINTER_LIST * ignore_ast_list, AST * target_statement, int inc_dec_flag)

inc_expr や dec_expr などのインクリメントやデクリメント式であるノード内を探索し、それに対するアドレス参照や、識別子を探し出し、見つけたら配列オフセットリスト array_offset_list へ入れる。

引数

<i>root</i>	inc_expr や dec_expr などのインクリメントやデクリメント式に関する AST ノード
<i>function_information_list</i>	関数に関する情報のリスト
<i>vtlist</i>	メモリ確保情報を取得するのに必要なプログラム変数リスト
<i>array_offset_list</i>	左辺式上にあるポインタ参照に対するオフセットリスト
<i>ignore_ast_list</i>	同じ位置のポインタが来ても無視するためのリスト
<i>target_statement</i>	この計算式を属している AST ノードへのアドレス (基本的に expression_statement であるノードが入る)
<i>inc_dec_flag</i>	インクリメントおよびデクリメントが含まれているかどうかのフラグ 1 : インクリメントが含まれている 2 : デクリメントが含まれている

戻り値

なし

4.10.3.16 void getDeclaratorArrayOffset (ARRAY_OFFSET_LIST * declarator_array_offset_list, AST * declarator_expression, AST * target_expression, VARIABLE_TABLE_LIST * vtlist)

変数テーブルから、変数の定義に対するノードに該当する情報を探し、それに対する配列オフセット情報を取得する。

引数

<i>declarator_-array_-offset_list</i>	取得先の配列オフセット情報
<i>declarator_-expression</i>	変数定義までの AST アドレス
<i>target_-expression</i>	対象の <i>declarator_with_init</i> への AST アドレス
<i>vtlist</i>	調べる先の変数テーブル

戻り値

なし

4.10.3.17 void *getExpressionOffsetInfo* (AST * *expression*, FUNCTION_INFORMATION_LIST * *function_information_list*, VARIABLE_TABLE_LIST * *vtlist*, ASTPOINTER_LIST * *ignore_ast_list*, ARRAY_OFFSET_LIST * *array_offset_list*, AST * *target_expression*, int * *switch_mode*, int *allow_subeffect*)

指定した式から、必要なオフセット情報を取得する。

引数

<i>expression</i>	指定した式に関する AST ノード
<i>function_-information_-list</i>	関数に関する情報のリスト
<i>vtlist</i>	検証対象の式をマークするための変数リスト
<i>ignore_ast_-list</i>	ポインタでの位置が検証済みである、IDENTIFIER を無視するための AST のアドレスリスト
<i>array_-offset_list</i>	各ポインタおよび配列ごとのオフセットのリスト
<i>target_-expression</i>	この左辺式の上位に位置する AST ノード
<i>switch_-mode</i>	直接アクセスおよび配列アクセスを探すか、IDENTIFIER を探すかどうかのスイッチフラグ 0 : 両方さがす 1 : <i>direct_ref</i> や <i>array_access</i> のみ探す
<i>allow_-subeffect</i>	副作用の式を許すかどうかのフラグ 1 : 許す 0 : 許さない

戻り値

なし

4.10.3.18 void *getOFFSET_LISTFromVariableTable* (OFFSET_LIST * *offset_list*, VARIABLE_TABLE * *variable_table*)

変数テーブルデータ *variable_table* からオフセットリスト *offset_list* を生成する。

引数

<i>offset_list</i>	生成先のオフセットリスト
<i>variable_table</i> (p.15)	変数テーブルデータ

戻り値

なし

4.10.3.19 int getOffsetLevelFromArrayOffset (ARRAY_OFFSET * array_offset)

指定した配列オフセットから、演算後のポインタレベルを求める。演算後のポインタレベルはつぎのとおりである。演算後のポインタレベル = この変数の配列とポインタレベルの合計値 + アンパサンドフラグ (挟んでいるなら 1、そうでない場合は 0) - この配列オフセット内のオフセットリスト

引数

<i>array_offset</i> (p.8)	指定した配列オフセット
------------------------------	-------------

戻り値

求めた演算後のポインタレベルを返す

4.10.3.20 void getPointerAccessOrIdentifierList (AST * root, AST *** output, int * getSize)

direct_ref として指定した AST ノード root から、以下のノードを探しだし、それを AST リスト output として取得する。

IDENTIFIER array_access, direct_ref, IDENTIFIER, primary_expression

なお、output の内容を NULL にすることで、root より下位のノードからが検索の対象となる。見つからない場合は 0 である。

引数

<i>root</i>	指定した AST ノード
<i>output</i>	上記の見つけたノードへのアドレス
<i>getSize</i>	取得した値のサイズを返すための変数。見つからない場合は 0 にされる。

戻り値

なし

4.10.3.21 `int getPointerArrayOffset (AST * root, FUNCTION_INFORMATION_LIST * function_information_list, VARIABLE_TABLE_LIST * vtlist, int offset_level, ASTPOINTER_LIST * ignore_ast_list, ARRAY_OFFSET ** array_offset, AST * target_statement, int anpasand_flag, int inc_dec_flag)`

ポインタおよび配列変数の各次元のオフセットとなる式を求める。

引数

<i>root</i>	左辺値に関する AST ノード
<i>function_information_list</i>	関数に関する情報リスト
<i>vtlist</i>	検証対象の式をマークするための変数リスト
<i>offset_level</i>	オフセットレベルを計算するためのところ。基本的に 0 を入力する。1 以上入力すれば、それが最下位レベルとなる。
<i>ignore_ast_list</i>	ポインタでの位置が検証済みである、IDENTIFIER を無視するための AST のアドレスリスト
<i>array_offset</i> (p. 8)	ポインタおよび配列のオフセット情報
<i>target_statement</i>	検証式の対象となるステートメント
<i>anpasand_flag</i>	アンパサンドを挟んでいるかどうかのフラグ 1 : 挟んでいる 0 : 挟んでいない
<i>inc_dec_flag</i>	インクリメントおよびデクリメントが含まれているかどうかのフラグ 0 : 含んでいない 1 : インクリメントが含まれている 2 : デクリメントが含まれている

戻り値

なし

4.10.3.22 `void getSingleExpressionOffsetInfo (AST * expression, FUNCTION_INFORMATION_LIST * function_information_list, VARIABLE_TABLE_LIST * vtlist, ASTPOINTER_LIST * ignore_ast_list, ARRAY_OFFSET_LIST * array_offset_list, AST * target_expression, int * switch_mode)`

指定した式から、必要なオフセット情報を取得する。これは副作用の式を許す。

引数

<i>expression</i>	指定した式に関する AST ノード
<i>function_information_list</i>	関数に関する情報のリスト
<i>vtlist</i>	検証対象の式をマークするための変数リスト
<i>ignore_ast_list</i>	ポインタでの位置が検証済みである、IDENTIFIER を無視するための AST のアドレスリスト
<i>array_offset_list</i>	各ポインタおよび配列ごとのオフセットのリスト

<i>target_expression</i>	この左辺式の上位に位置する AST ノード
<i>switch_mode</i>	直接アクセスおよび配列アクセスを探すか、IDENTIFIER を探すかどうかのスイッチフラグ 0 : 両方さがす 1 : direct_ref や array_access のみ探す

戻り値

なし

4.10.3.23 int getUpperExpressionRelationNode (AST * *target*, AST * *root*, AST ** *output*, AST ** *output2*)

ポインタのオフセットの検証対象となっている変数を示す AST ノード *target* から、間接的にどの関係の中に位置しているかどうかを調べ、そのノードのアドレス *output* として返す。このとき、*target* より明らかに上位である AST ノード *root* を設定しなければならない。また、*output* が *minus_expr* の場合はそのポインタよりひとつ下が左辺か右辺かどうかを調べるために、そのポインタの一つ下のノードを *output2* へ代入する。

引数

<i>target</i>	検証対象となっている変数
<i>root</i>	検証対象のノード
<i>output</i>	出力される間接的に関係しているノードへのアドレス
<i>output2</i>	<i>output</i> が <i>minus_expr</i> の場合、 <i>minus_expr</i> より 1 つ下のノードがここに代入される

戻り値

検索が成功したかどうかのフラグ。成功した場合は 1、そうでない場合は 0 を返す。
なし

ポインタのオフセットの検証対象となっている変数を示す AST ノード *target* から、間接的にどの関係の中に位置しているかどうかを調べ、そのノードのアドレス *output* として返す。このとき、*target* より明らかに上位である AST ノード *root* を設定しなければならない。また、*output* が *minus_expr* の場合はそのポインタよりひとつ下が左辺か右辺かどうかを調べるために、そのポインタの一つ下のノードを *output2* へ代入する。

引数

<i>target</i>	検証対象となっている変数
<i>root</i>	検証対象のノード
<i>output</i>	出力される間接的に関係しているノードへのアドレス
<i>output2</i>	<i>output</i> が <i>minus_expr</i> の場合、 <i>minus_expr</i> より 1 つ下のノードがここに代入される

戻り値

検索が成功したかどうかのフラグ。成功した場合は 1、そうでない場合は 0 を返す。

4.10.3.24 `ARRAY_OFFSET* maxOffsetLevelAddressFromArrayOffsetList (ARRAY_OFFSET_LIST * array_offset_list)`

指定した配列オフセットリストでの演算後のポインタレベルの最大レベルである配列オフセットのアドレスを求める。配列オフセットが空の場合は NULL を代入する。

引数

<code>array_offset_list</code>	指定した配列オフセットリスト
--------------------------------	----------------

戻り値

求めた演算後のポインタレベルが最大である配列オフセットのアドレスを返す

4.10.3.25 `int maxOffsetLevelFromArrayOffsetList (ARRAY_OFFSET_LIST * array_offset_list)`

指定した配列オフセットリストでの演算後のポインタレベルの最大レベルを求める。配列オフセットが空の場合は 0 を代入する。

引数

<code>array_offset_list</code>	指定した配列オフセットリスト
--------------------------------	----------------

戻り値

求めた演算後のポインタレベルを返す。

指定した配列オフセットリストでの演算後のポインタレベルの最大レベルを求める。配列オフセットが空の場合は 0 を代入する。

引数

<code>array_offset_list</code>	指定した配列オフセットリスト
--------------------------------	----------------

戻り値

求めた演算後のポインタレベルを返す

4.10.3.26 void minusArrayOffsetList (ARRAY_OFFSET_LIST * *target_array_offset_list*,
ARRAY_OFFSET_LIST * *delete_array_offset_list*)

対象の配列オフセットリスト *target_array_offset_list* に対して、対象から取り除きたい配列オフセットリスト *delete_array_offset_list* の名前に該当する配列オフセット情報を削除する。

引数

<i>target_array_offset_list</i>	対象の配列オフセットリスト
<i>delete_array_offset_list</i>	対象から取り除きたい配列オフセットリスト

戻り値

なし

4.10.3.27 void moveArrayOffsetList (ARRAY_OFFSET_LIST * *fromlist*, ARRAY_OFFSET_LIST *
tolist, int *move_start*)

配列オフセットリスト *fromlist* 内の *move_start* 以降のデータをすべて、もう一方の配列オフセットリスト *tolist* に移動させる。

引数

<i>fromlist</i>	移動もとの配列オフセットリスト
<i>tolist</i>	移動先の配列オフセットリスト
<i>move_start</i>	移動させたいデータの位置 (先頭から 0 番目とする)

戻り値

なし

4.10.3.28 ARRAY_OFFSET* new_ARRAY_OFFSET (CSTLString * *variable_name*, AST
* *target_statement*, AST * *variable_address*, OFFSET_LIST * *offset_list*, int
pointer_level, int *array_level*, int *anpasand_flag*, int *inc_dec_flag*)

配列やポインタの各次元のオフセット関係を格納するための構造体のデータを生成させる。

引数

<i>variable_name</i>	変数名
<i>target_statement</i>	ターゲットの statement

<i>variable_ - address</i>	この変数名が指している AST アドレス
<i>offset_list</i>	各次元のオフセット
<i>pointer_ - level</i>	この変数のポインタレベル
<i>array_level</i>	この変数の配列レベル
<i>anpasand_ - flag</i>	この変数はアンパサンドを挟んでいるかどうかのフラグ 1 : 挟んでいる 0 : 挟んでいない
<i>inc_dec_flag</i>	インクリメントおよびデクリメントが含まれているかどうかのフラグ 0 : 含んでいない 1 : インクリメントが含まれている 2 : デクリメントが含まれている

戻り値

配列やポインタの各次元のオフセットに関する構造体へのアドレスを返す。

。配列やポインタの各次元のオフセット関係を格納するための構造体のデータを生成させる。

引数

<i>variable_ - name</i>	変数名
<i>target_ - statement</i>	ターゲットの statement
<i>variable_ - address</i>	この変数名が指している AST アドレス
<i>offset_list</i>	各次元のオフセット
<i>pointer_ - level</i>	この変数のポインタレベル
<i>array_level</i>	この変数の配列レベル
<i>anpasand_ - flag</i>	この変数はアンパサンドを挟んでいるかどうかのフラグ 1 : 挟んでいる 0 : 挟んでいない
<i>inc_dec_flag</i>	インクリメントおよびデクリメントが含まれているかどうかのフラグ 0 : 含んでいない 1 : インクリメントが含まれている 2 : デクリメントが含まれている

戻り値

配列やポインタの各次元のオフセットに関する構造体へのアドレスを返す。

4.10.3.29 ARRAY_OFFSET* `new_ARRAY_OFFSET_char (char * variable_name, AST * target_statement, AST * variable_address, OFFSET_LIST * offset_list, int pointer_level, int array_level, int anpasand_flag, int inc_dec_flag)`

配列やポインタの各次元のオフセット関係を格納するための構造体のデータを生成させる。

引数

<i>variable_name</i>	変数名
<i>target_statement</i>	ターゲットの statement
<i>variable_address</i>	この変数名が指している AST アドレス
<i>offset_list</i>	各次元のオフセット
<i>pointer_level</i>	この変数のポインタレベル
<i>array_level</i>	この変数の配列レベル
<i>anpasand_flag</i>	アンパサンドを挟んでいるかどうかのフラグ 1 : 挟んでいる 0 : 挟んでいない
<i>inc_dec_flag</i>	インクリメントおよびデクリメントが含まれているかどうかのフラグ 0 : 含んでいない 1 : インクリメントが含まれている 2 : デクリメントが含まれている

戻り値

配列やポインタの各次元のオフセットに関する構造体へのアドレスを返す

配列やポインタの各次元のオフセット関係を格納するための構造体のデータを生成させる。

引数

<i>variable_name</i>	変数名
<i>target_statement</i>	ターゲットの statement
<i>variable_address</i>	この変数名が指している AST アドレス
<i>offset_list</i>	各次元のオフセット
<i>pointer_level</i>	この変数のポインタレベル
<i>array_level</i>	この変数の配列レベル
<i>anpasand_flag</i>	アンパサンドを挟んでいるかどうかのフラグ 1 : 挟んでいる 0 : 挟んでいない
<i>inc_dec_flag</i>	インクリメントおよびデクリメントが含まれているかどうかのフラグ 0 : 含んでいない 1 : インクリメントが含まれている 2 : デクリメントが含まれている

戻り値

配列やポインタの各次元のオフセットに関する構造体へのアドレスを返す。

4.10.3.30 void OFFSET_LIST_push_back_alloc (OFFSET_LIST * *offset_list*, char * *string*)

任意の文字列を、動的変数としてオフセットリストに追加する。

引数

<i>offset_list</i>	対象のオフセットリスト
<i>string</i>	任意の文字列

戻り値

なし

4.10.3.31 void printASTPOINTER_LIST (ASTPOINTER_LIST * *astpointer_list*)

AST のポインタリストの内容を出力させる。

引数

<i>astpointer_list</i>	AST のポインタリスト
------------------------	--------------

戻り値

なし

4.10.3.32 ARRAY_OFFSET* searchARRAY_OFFSET_LIST (ARRAY_OFFSET_LIST * *array_offset_list*, CString * *name*)

配列オフセットリスト *array_offset_list* から、指定した変数名を探索し、見つければその変数名へのアドレスを返す。

引数

<i>array_offset_list</i>	探索対象の配列オフセットリスト
<i>name</i>	探索したい変数名

戻り値

見つければ変数名へのアドレスを返す。そうでなければ NULL を返す。

4.10.3.33 int searchExpressionOrPointeArrayOrIden (AST * *root*, AST ** *output*)

primary_expression として指定した AST ノード *root* から、その次の下位である次のノード名を探し出し、そのアドレスを *output* へ出力させ、1 を返す。

minus_expr, *plus_expr*, *array_access*, *direct_ref*, IDENTIFIER, *primary_expression*

なお、*output* の内容を NULL にすることで、*root* より下位のノードからが検索の対象となる。また、ポインタ計算の関係上、++演算演算子を示すようなものや CONSTANT(定数) が来た場合のみ、-1 を返す。見つからない場合は 0 である。

引数

<i>root</i>	指定した AST ノード
<i>output</i>	上記の見つけたノードへのアドレス

戻り値

上記の条件で値を返却する。

4.10.3.34 `ARRAY_OFFSET* searchOffsetLevelAddressFromArrayOffsetList (AST * root, ARRAY_OFFSET_LIST * array_offset_list, int pointer_level, int array_level)`

対象の AST ノードから、演算後のポインタレベルが指定されたポインタレベルと配列レベルの合計と一致するような変数の配列オフセットを指定された配列オフセットリストから探し出し、見つかったらアドレスを取得する。

引数

<i>root</i>	対象の AST ノード
<i>array_offset_list</i>	対象の配列オフセットリスト
<i>pointer_level</i>	指定するポインタレベル
<i>array_level</i>	指定する配列レベル

戻り値

演算後のポインタレベルと指定されたポインタレベルと配列レベルの合計が一致するような変数を返す。失敗した場合は NULL を返す。

4.10.3.35 `int searchPointerAccessOrIdentifierOrPrimary (AST * root, AST ** output)`

`direct_ref` として指定した AST ノード `root` から、その次の下位である次のノード名を探し出し、そのアドレスを `output` へ出力させ、1 を返す。

IDENTIFIER array_access, direct_ref, IDENTIFIER, primary_expression inc_after_expression, inc_expr, dec_after_expr, inc_expr assignment_expression

なお、`output` の内容を NULL にすることで、`root` より下位のノードからが検索の対象となる。見つからない場合は 0 である。

引数

<i>root</i>	指定した AST ノード
<i>output</i>	上記の見つけたノードへのアドレス

戻り値

上記の条件で値を返却する。

4.11 ANSICInformation/PreProcess.h

このファイルは構文解析を通せるように、Gcc で C 言語にプリプロセス (前処理) をさせるための命令が含まれている。このファイルでできることとしては、`::include` をコメントアウトすることで省いて、プリプロセスを掛けることができる。

```
#include <cstl/list.h>
#include "../Library/CSTLString.h"
```

データ構造

- struct **include_data**

型定義

- typedef struct **include_data** **INCLUDE_DATA**

関数

- **INCLUDE_DATA * new_INCLUDE_DATA** (CSTLString ***include_data**, int line)
- int **preProcessor** (char *source_file_name)
- int **includeComment** (char *source_file_name)
- int **adjustProgramStart** (char *source_file_name)
- void **readIncludeDataFromFile** (char *file_name, INCLUDE_LIST *include_list)
- void **addIncludeDataFromFile** (char *file_name, INCLUDE_LIST *include_list)

4.11.1 説明

このファイルは構文解析を通せるように、Gcc で C 言語にプリプロセス (前処理) をさせるための命令が含まれている。このファイルでできることとしては、`::include` をコメントアウトすることで省いて、プリプロセスを掛けることができる。

作者

faithnh

4.11.2 型定義

4.11.2.1 typedef struct include_data INCLUDE_DATA

インクルードファイルに関する情報が格納される。これは、検証式付加時にインクルードファイルを付加するのに使用する。

4.11.3 関数

4.11.3.1 void addIncludeDataFromFile (char * *file_name*, INCLUDE_LIST * *include_list*)

インクルードリストを基に、対象のファイルにインクルードを追加する。

引数

<i>file_name</i>	開くファイル名
<i>include_list</i>	インクルードリスト

戻り値

なし

4.11.3.2 int adjustProgramStart (char * *source_file_name*)

プログラムの始まりを示す位置まですべて削除する。

引数

<i>source_file_name</i>	対象のソースファイル名
-------------------------	-------------

戻り値

成功したかどうかを示す 成功した場合は 1 をかえし、そうでない場合は 0 を返す。

4.11.3.3 int includeComment (char * *source_file_name*)

"#include"にコメントをかけておく。コメントアウトをかけた結果はファイル名 *_out.c* として出力される。

引数

<i>source_file_name</i>	対象のソースファイル名
-------------------------	-------------

戻り値

成功したかどうかを示す 成功した場合は 1 をかえし、そうでない場合は 0 を返す。

4.11.3.4 INCLUDE_DATA* new_INCLUDE_DATA (CString * *include_data*, int *line*)

新しいインクルードファイルを生成する。

引数

<i>include_data</i> (p. 11)	include ファイルの名前
<i>line</i>	その行数

4.11.3.5 int preProcessor (char * source_file_name)

実行させたいソースにプリプロセッサをかける。プリプロセッサのかけたファイルはファイル名_out_pre.c として出力される。

引数

<i>source_file_name</i>	実行させたいソースファイル名
-------------------------	----------------

戻り値

成功したかどうかを示す 成功した場合は 1 をかえし、そうでない場合は 0 を返す。

4.11.3.6 void readIncludeDataFromFile (char * file_name, INCLUDE_LIST * include_list)

"#include"にコメントをかけたもののみを取り出し、取りだしたものをインクルードファイルのリスト include_list にいれる。

引数

<i>file_name</i>	開くファイル名
<i>include_list</i>	インクルードリスト

戻り値

なし

4.12 ANSICInformation/Return_Info.h

これはリターン命令に関する情報を取得するための命令が含まれている。

```
#include "PointerArrayControl.h"
```

データ構造

- struct **return_info**

型定義

- typedef struct **return_info** **RETURN_INFO**

関数

- **RETURN_INFO** * **new_RETURN_INFO** (**AST** *target_expression, **ARRAY_OFFSET_LIST** *return_array_offset_list)

4.12.1 説明

これはリターン命令に関する情報を取得するための命令が含まれている。

作者

faithnh

4.12.2 型定義

4.12.2.1 typedef struct return_info RETURN_INFO

リターン命令に関する情報が含まれている。

4.12.3 関数

4.12.3.1 RETURN_INFO* new_RETURN_INFO (AST * target_expression, ARRAY_OFFSET_LIST * return_array_offset_list)

リターン命令に関する情報を生成させる。

引数

<i>target_expression</i>	リターン命令自体へのノードへのアドレス
<i>return_array_offset_list</i>	リターン命令で表記された式の配列オフセットリスト

戻り値

生成されたリターン命令に関する情報へのアドレスを返す。

4.13 ANSICInformation/SubEffectCheck.h

このファイルには副作用式のチェックする関数や代入式のタイプを求める命令が含まれている。

```
#include "AST.h"
```

関数

- `int checkContainSubEffectStatement (AST *node)`
- `int getAssignment_TYPE (AST *assignment_expression_list)`

4.13.1 説明

このファイルには副作用式のチェックする関数や代入式のタイプを求める命令が含まれている。

作者

faithnh

4.13.2 関数

4.13.2.1 `int checkContainSubEffectStatement (AST * node)`

指定された AST ノード `node` から、副作用式 (インクリメント式・デクリメント式・代入式) が含まれているかどうかチェックする。

引数

<i>node</i>	対象の AST ノード
-------------	-------------

戻り値

含まれていた式がインクリメント式の場合 1、デクリメント式の場合 2、代入式の場合 3 とする。含まれていない場合は 0 を返す。

4.13.2.2 `int getAssignment_TYPE (AST * assignment_expression_list)`

代入式のタイプを出力させる。

引数

<i>assignment_expression_list</i>	代入式に関する AST ノードのリスト
-----------------------------------	---------------------

戻り値

代入式のタイプに応じた値を返却する。0: =, 1: + =, 2: - =, 3: * =, 4: / =, 5: =, 6: < < =, 7: > > =, 8: & =, 9: | =, 10: ^ =

4.14 ANSICInformation/Synbol.h

このファイルは、構文解析によって生成された抽象構文木 (AST) から、変数・typedef テーブル・構造体テーブルを生成させるための命令が含まれている。とくに、typedef テーブルの生成は、C 言語の構文解析では必須な処理である。

```
#include <stdio.h>
#include <cstl/list.h>
#include "../Library/CSTLString.h"
#include "../Library/IdList.h"
#include "AST.h"
```

データ構造

- struct **typedef_table**
- struct **variable_table**
- struct **struct_table**

型定義

- typedef struct **typedef_table** **TYPEDEF_TABLE**
- typedef struct **variable_table** **VARIABLE_TABLE**
- typedef struct **struct_table** **STRUCT_TABLE**

関数

- **TYPEDEF_TABLE * new_TYPEDEF_TABLE** (CSTLString *target_type, CSTLString *change_type)
- **VARIABLE_TABLE * new_VARIABLE_TABLE** (int enable_start, int enable_end, AST *declaration_location_address, int block_level, int block_id, IDLIST *idlist, CSTLString *type, CSTLString *variable_name, AST *initializer)
- **STRUCT_TABLE * new_STRUCT_TABLE_with_char** (int line, char *type, char *struct_name, VARIABLE_TABLE_LIST *member_list)
- **TYPEDEF_TABLE * new_TYPEDEF_TABLE_with_char** (char *target_type, char *change_type)
- **VARIABLE_TABLE * new_VARIABLE_TABLE_with_char** (int enable_start, int enable_end, AST *declaration_location_address, int block_level, int block_id, IDLIST *idlist, char *type, char *variable_name, AST *initializer)
- **STRUCT_TABLE * new_STRUCT_TABLE** (int line, CSTLString *type, CSTLString *struct_name, VARIABLE_TABLE_LIST *member_list)
- void **getTYPEDEF_TABLE_DATA** (TYPEDEF_TABLE_LIST *typedef_table_list, AST *typelist, AST *identifier)
- **AST * getTYPEDEFfromAST** (TYPEDEF_TABLE_LIST *typedef_table_list, char *token, int line)

- void **printTYPEDEF_TABLE_LIST** (TYPEDEF_TABLE_LIST *typedef_table_list)
- void **getSTRUCT_TABLE_DATA** (STRUCT_TABLE_LIST *struct_table_list, AST *ast_data)
- int **find_STRUCT_TABLE_DATA** (STRUCT_TABLE_LIST *struct_table_list, CSTLString *target)
- void **getSTRUCT_DATA** (AST *ast_data, STRUCT_TABLE_LIST *struct_table_data)
- void **getMemberList** (VARIABLE_TABLE_LIST *member_list, AST *ast_data)
- void **getDeclaratorFromAST** (char const *type, AST *ast_data, VARIABLE_TABLE_LIST *member_list, int enable_start, int enable_end, int block_level, int block_id, AST *declaration_location_address)
- void **printSTRUCT_TABLE_LIST** (STRUCT_TABLE_LIST *struct_table_list)
- void **getVARIABLE_TABLE_LIST** (VARIABLE_TABLE_LIST *variable_table_list, AST *ast_data)
- void **getParameterData** (VARIABLE_TABLE_LIST *variable_table_list, AST *ast_data, AST *enable_start, AST *enable_end)
- void **getParameterVARIABLE_TABLE_LIST** (VARIABLE_TABLE_LIST *variable_table_list, AST *ast_data)
- void **printVARIABLE_TABLE_LIST** (VARIABLE_TABLE_LIST *variable_table_list)
- void **getPointerLevelAndArrayLevelFromVARIABLE_TABLE** (VARIABLE_TABLE *variable_table_data, int *output_pointer_level, int *output_array_level)
- void **getPointerLevelAndArrayLevel** (CSTLString *target_identifier, int *output_pointer_level, int *output_array_level)
- **VARIABLE_TABLE * searchVARIABLE_TABLE_LIST** (IDLIST *target_idlist, CSTLString *target_string, VARIABLE_TABLE_LIST *variable_table_list)
- void **deletePointerAndArraySynbol** (CSTLString *target)
- void **deletePointer** (CSTLString *target)

4.14.1 説明

このファイルは、構文解析によって生成された抽象構文木 (AST) から、変数・typedef テーブル・構造体テーブルを生成させるための命令が含まれている。とくに、typedef テーブルの生成は、C 言語の構文解析では必須な処理である。

作者

faithnh

4.14.2 型定義

4.14.2.1 typedef struct struct_table STRUCT_TABLE

構造体に関する情報であり、プログラム中の構造体の識別するのに用いられる。

4.14.2.2 typedef struct typedef_table TYPEDEF_TABLE

型定義に関する情報で、BISON による構文解析時の型定義の認識に用いられる。

4.14.2.3 typedef struct variable_table VARIABLE_TABLE

プログラム中の変数に関する情報であり、検証式生成時に変数を識別するのに用いられる。

4.14.3 関数

4.14.3.1 void deletePointer (CSTLString * *target*)

変数名からポインタを示す記号のみ全て削除する。

引数

<i>target</i>	変更対象の変数名
---------------	----------

戻り値

なし

4.14.3.2 void deletePointerAndArraySynbol (CSTLString * *target*)

変数名から配列およびポインタを示す記号を全て削除する。

引数

<i>target</i>	変更対象の変数名
---------------	----------

戻り値

なし

4.14.3.3 int find_STRUCT_TABLE_DATA (STRUCT_TABLE_LIST * *struct_table_list*, CSTLString * *target*)

構造体テーブルリストに同じ定義がないかどうかを調べる。

引数

<i>struct_table_list</i>	検索対象の構造体テーブルリスト
<i>target</i>	検索する文字列

戻り値

見つけれたら、1 を返し、そうでなければ 0 を返す。

4.14.3.4 void getDeclaratorFromAST (char const * *type*, AST * *ast_data*,
VARIABLE_TABLE_LIST * *member_list*, int *enable_start*, int *enable_end*, int *block_level*,
int *block_id*, AST * *declaration_location_address*)

指定された AST ノードから、declarator を探し、それを見つけたら指定した型の変数として変数リストに登録する。

引数

<i>type</i>	指定した型
<i>ast_data</i>	指定された AST ノード
<i>member_list</i>	登録先の変数リスト
<i>enable_start</i>	変数スコープの有効範囲の開始
<i>enable_end</i>	変数スコープの有効範囲の終わり
<i>block_level</i>	この変数のブロックレベル
<i>block_id</i>	ブロックを識別するための識別番号
<i>declaration_location_address</i>	この宣言自体の AST へのアドレス (検証式の生成に必要)

戻り値

なし

4.14.3.5 void getMemberList (VARIABLE_TABLE_LIST * *member_list*, AST * *ast_data*)

指定された AST ノードから、メンバリストを生成する。

引数

<i>member_list</i>	登録対象のメンバリスト
<i>ast_data</i>	指定された AST ノード

戻り値

作成された構造体データへのアドレスを返却する

指定された AST ノードから、メンバリストを生成する。

引数

<i>member_list</i>	登録対象のメンバリスト
<i>ast_data</i>	指定された AST ノード

戻り値

作成された構造体データへのアドレスを返却する。

4.14.3.6 void getParameterData (VARIABLE_TABLE_LIST * *variable_table_list*, AST * *ast_data*,
AST * *enable_start*, AST * *enable_end*)

関数のパラメータリストを示す AST ノードから、parameter_declaration を見つけ、そこから変数テーブルのリストに登録させる。

引数

<i>variable_table_list</i>	変数テーブルのリスト
<i>ast_data</i>	対象の AST ノード
<i>enable_start</i>	有効範囲の開始を示す AST ノードのアドレス
<i>enable_end</i>	有効範囲の終了を示す AST ノードのアドレス

戻り値

なし

関数のパラメータリストを示す AST ノードから、parameter_declaration を見つけ、そこから変数テーブルのリストに登録させる。

引数

<i>variable_table_list</i>	変数テーブルのリスト
<i>ast_data</i>	対象の AST ノード
<i>enable_start</i>	有効範囲の開始
<i>enable_end</i>	有効範囲の終了

戻り値

なし

4.14.3.7 void getParameterVARIABLE_TABLE_LIST (VARIABLE_TABLE_LIST *
variable_table_list, AST * *ast_data*)

対象の AST ノードから関数を探し、関数内の引数を変数テーブルのリストに登録する。

引数

<i>variable_table_list</i>	変数テーブルのリスト
<i>ast_data</i>	対象の AST ノード

戻り値

なし

4.14.3.8 void getPointerLevelAndArrayLevel (CSTLString * *target_identifier*, int * *output_pointer_level*, int * *output_array_level*)

対象の識別子のポインタの次元および配列の次元を取得する。

引数

<i>target_identifier</i>	対象の識別子
<i>output_pointer_level</i>	出力されるポインタレベル
<i>output_array_level</i>	出力される配列レベル

戻り値

なし

4.14.3.9 void getPointerLevelAndArrayLevelFromVARIABLE_TABLE (VARIABLE_TABLE * *variable_table_data*, int * *output_pointer_level*, int * *output_array_level*)

変数テーブルから、ポインタの次元および配列の次元を取得する。

引数

<i>variable_table_data</i>	変数テーブルのリスト
<i>output_pointer_level</i>	出力されるポインタレベル
<i>output_array_level</i>	出力される配列レベル

戻り値

なし

4.14.3.10 void getSTRUCT_DATA (AST * *ast_data*, STRUCT_TABLE_LIST * *struct_table_data*)

指定された AST ノードから、構造体データを作成させ、構造体テーブルのリストへ登録させる。

引数

<i>ast_data</i>	指定された AST ノード
<i>struct_-table_data</i>	登録先の構造体テーブルリスト

戻り値

なし

4.14.3.11 void getSTRUCT_TABLE_DATA (STRUCT_TABLE_LIST * *struct_table_list*, AST * *ast_data*)

指定された AST ノードから、構造体テーブルリストに構造体データを登録させる。

引数

<i>struct_-table_list</i>	登録先の構造体テーブルリスト
<i>ast_data</i>	指定された AST ノード

戻り値

なし

4.14.3.12 void getTYPEDEF_TABLE_DATA (TYPEDEF_TABLE_LIST * *typedef_table_list*, AST * *typelist*, AST * *identifier*)

指定した AST ノードから参照し、もし typedef 宣言の場合は、typedef テーブルに入れる。

引数

<i>typedef_-table_list</i>	typedef テーブル
<i>typelist</i>	型リストへの AST ノード
<i>identifier</i>	識別への AST ノード

戻り値

なし

4.14.3.13 AST* getTYPEDEFfromAST (TYPEDEF_TABLE_LIST * *typedef_table_list*, char * *token*, int *line*)

指定した typedef テーブルのリストから参照し、指定されたトークンに一致するような typedef テーブルデータが存在するかどうか調べる。もし、見つければ、内

容が指定されたトークンで、名前が TYPE_NAME である AST ノードを生成し、それへのアドレスを返す。

引数

<i>typedef_table_list</i>	指定した typedef テーブルのリスト
<i>token</i>	指定されたトークン

戻り値

生成された AST ノードへのアドレスを返す。

指定した typedef テーブルのリストから参照し、指定されたトークンに一致するような typedef テーブルデータが存在するかどうか調べる。もし、見つければ、内容が指定されたトークンで、名前が TYPE_NAME である AST ノードを生成し、それへのアドレスを返す。見つけれなければ、名前が IDENTIFIER である AST ノードを生成し、それへのアドレスを返す。

引数

<i>typedef_table_list</i>	指定した typedef テーブルのリスト
<i>token</i>	指定されたトークン
<i>line</i>	指定された行数

戻り値

生成された AST ノードへのアドレスを返す。

4.14.3.14 void getVARIABLE_TABLE_LIST (VARIABLE_TABLE_LIST * *variable_table_list*, AST * *ast_data*)

対象の AST ノードから変数テーブルのリストを登録する。

引数

<i>variable_table_list</i>	変数テーブルのリスト
<i>ast_data</i>	対象の AST ノード

戻り値

なし

4.14.3.15 STRUCT_TABLE* new_STRUCT_TABLE (int *line*, CSTLString * *type*, CSTLString * *struct_name*, VARIABLE_TABLE_LIST * *member_list*)

新しい構造体テーブルのデータを生成させる。

引数

<i>line</i>	行数
<i>type</i>	型名 (struct か union のいずれか)
<i>struct_name</i>	構造体の名前
<i>member_list</i>	メンバリスト (変数テーブルより)

戻り値

新しく生成された構造体テーブルのデータへのアドレスが返される。

新しい構造体テーブルのデータを生成させる。

引数

<i>line</i>	行数
<i>type</i>	型名
<i>struct_name</i>	構造体名 (struct か union のいずれか)
<i>member_list</i>	メンバリスト (変数テーブルより)

戻り値

新しく生成された構造体テーブルのデータへのアドレスが返される。

4.14.3.16 `STRUCT_TABLE* new_STRUCT_TABLE_with_char (int line, char * type, char * struct_name, VARIABLE_TABLE_LIST * member_list)`

新しい構造体テーブルのデータを生成させる (char 文字列対応)。

引数

<i>line</i>	行数
<i>type</i>	型名 (struct か union のいずれか)
<i>struct_name</i>	構造体の名前
<i>member_list</i>	メンバリスト (変数テーブルより)

戻り値

新しく生成された構造体テーブルのデータへのアドレスが返される。

4.14.3.17 `TYPEDEF_TABLE* new_TYPEDEF_TABLE (CSTLString * target_type, CSTLString * change_type)`

新しいtypedef テーブルのデータを生成させる。

引数

<i>target_type</i>	typedef の対象の型
<i>change_type</i>	typedef で割り当てた後の新しい型名

戻り値

新しく生成された typedef テーブルのデータへのアドレスが返される。

4.14.3.18 **TYPDEF_TABLE*** new_TYPEDEF_TABLE_with_char (char * *target_type*, char * *change_type*)

新しい typedef テーブルのデータを生成させる (char 文字列対応)。

引数

<i>target_type</i>	typedef の対象の型
<i>change_type</i>	typedef で割り当てた後の新しい型名

戻り値

新しく生成された typedef テーブルのデータへのアドレスが返される。

4.14.3.19 **VARIABLE_TABLE*** new_VARIABLE_TABLE (int *enable_start*, int *enable_end*, AST * *declaration_location_address*, int *block_level*, int *block_id*, IDLIST * *idlist*, CSTLString * *type*, CSTLString * *variable_name*, AST * *initializer*)

新しい変数テーブルのデータを生成させる。

引数

<i>enable_start</i>	この変数の有効範囲の始まりの行数
<i>enable_end</i>	この変数の有効範囲の終わりの行数
<i>declaration_location_address</i>	この変数を宣言した場所を示す AST のアドレス
<i>block_level</i>	この変数のブロックレベル (グローバル変数なら 0 とし、関数の中での定義なら 1、その関数内の for 文などのブロック文ないでの宣言なら 2 とする)
<i>block_id</i>	ブロックごとの ID (基本的には 0 から始まり、ブロックレベル 2 が 2 回目にくると、1 となる)
<i>idlist</i>	ブロックごとの ID (これは変数スコープを識別するために使用する)
<i>type</i>	型名
<i>variable_name</i>	変数名
<i>initializer</i>	初期定義式への AST ノード

戻り値

新しく生成された変数テーブルのデータへのアドレスが返される。

4.14.3.20 VARIABLE_TABLE* new_VARIABLE_TABLE_with_char (int *enable_start*, int *enable_end*, AST * *declaration_location_address*, int *block_level*, int *block_id*, IDLIST * *idlist*, char * *type*, char * *variable_name*, AST * *initializer*)

新しい変数テーブルのデータを生成させる (char 文字列対応)。

引数

<i>enable_start</i>	この変数の有効範囲の始まりの行数
<i>enable_end</i>	この変数の有効範囲の終わりの行数
<i>declaration_location_address</i>	この変数を宣言した場所を示す AST のアドレス
<i>block_level</i>	この変数のブロックレベル (グローバル変数なら 0 とし、関数の中での定義なら 1、その関数内の for 文などのブロック文ないでの宣言なら 2 とする)
<i>block_id</i>	ブロックごとの ID (基本的には 0 から始り、ブロックレベル 2 が 2 回目にくると、1 となる)
<i>idlist</i>	ブロックごとの ID (これは変数スコープを識別するために使用する)
<i>type</i>	型名
<i>variable_name</i>	変数名
<i>initializer</i>	初期定義式への AST ノード

戻り値

新しく生成された変数テーブルのデータへのアドレスが返される。

4.14.3.21 void printSTRUCT_TABLE_LIST (STRUCT_TABLE_LIST * *struct_table_list*)

構造体テーブルのリストの内容を出力させる。

引数

<i>struct_table_list</i>	出力対象の構造体テーブルのリスト
--------------------------	------------------

戻り値

なし

4.14.3.22 void printTYPEDEF_TABLE_LIST (TYPEDEF_TABLE_LIST * *typedef_table_list*)

typedef テーブルのリストに登録されているものを、次のような形式で出力させる。

target_type change_type

引数

<i>typedef_-table_list</i>	出力対象の typedef テーブルのリスト
----------------------------	------------------------

戻り値

なし

typedef テーブルのリストに登録されているものを出力させる。

引数

<i>typedef_-table_list</i>	出力対象の typedef テーブルのリスト
----------------------------	------------------------

戻り値

なし

4.14.3.23 void printVARIABLE_TABLE_LIST (VARIABLE_TABLE_LIST * *variable_table_list*)

変数テーブルのリストの内容を出力させる。

引数

<i>variable_-table_list</i>	出力対象の変数テーブルのリスト
-----------------------------	-----------------

戻り値

なし

4.14.3.24 VARIABLE_TABLE* searchVARIABLE_TABLE_LIST (IDLIST * *target_idlist*, CSTLString * *target_string*, VARIABLE_TABLE_LIST * *variable_table_list*)

変数テーブルリスト *variable_table_list* から、指定した変数スコープの IDLIST *target_idlist* と *target_string* に該当ような変数テーブルへのアドレスを返す。

引数

<i>target_idlist</i>	指定した変数スコープの IDLIST
<i>target_-string</i>	対象の変数名
<i>variable_-table_list</i>	変数テーブルリスト

戻り値

上記の処理から見つけた変数テーブルへのアドレスを返す。見つからなければ NULL を返す。

4.15 ANSICInformation/Varidate_statement.h

これは C 言語プログラム上から、不具合を検証するための検証式や検証用を使用する変数などを追加するための命令が含まれている。

```
#include <cstl/list.h>
#include "Synbol.h"
#include "PointerArrayControl.h"
#include "MemallocInfo.h"
#include "FreeMemInfo.h"
#include "ForInformation.h"
#include "DivitionInformation.h"
#include "PreProcess.h"
#include "../ProgramSlicing/ProgramSlicingInformation.h"
```

データ構造

- struct **validate_variable**
- struct **validate_statement**

型定義

- typedef struct **validate_variable** **VALIDATE_VARIABLE**
- typedef struct **validate_statement** **VALIDATE_STATEMENT**

関数

- **VALIDATE_STATEMENT * new_VALIDATE_STATEMENT_char** (int target_id, int check_or_modify, int used, char *statement, AST *target_statement)
- **VALIDATE_STATEMENT * new_VALIDATE_STATEMENT** (int target_id, int check_or_modify, int used, CSTLString *statement, AST *target_statement)
- **VALIDATE_VARIABLE * new_VALIDATE_VARIABLE** (int used, int enable_start, int enable_end, int declaration_location, int block_level, int block_id, CSTLString *type, CSTLString *variable_name, CSTLString *target_variable_name, int offset_level)
- **VALIDATE_VARIABLE * new_VALIDATE_VARIABLE_with_char** (int used, int enable_start, int enable_end, int declaration_location, int block_level, int

- block_id, char *type, char *variable_name, char *target_variable_name, int offset_level)
- void **initVALIDATE_STATEMENT_flag** (VALIDATE_STATEMENT_LIST *validate_statement_list)
 - void **getValidate_Variable** (VARIABLE_TABLE_LIST *variable_table_list, VALIDATE_VARIABLE_LIST *validate_variable)
 - void **printVALIDATE_VARIABLE_LIST** (VALIDATE_VARIABLE_LIST *validate_variable_list)
 - void **createValidateStatement** (AST *root, FUNCTION_INFORMATION_LIST *function_information_list, VARIABLE_TABLE_LIST *vtlist, VALIDATE_VARIABLE_LIST *validate_variable_list, VALIDATE_STATEMENT_LIST *validate_statement_list, FOR_INFORMATION_LIST *for_information_list, int undefined_control_check, int zero_divition_check, int array_unbound_check, int free_violation_check)
 - void **getValidateStatementFromInitializer** (AST *root, FUNCTION_INFORMATION_LIST *function_information_list, VARIABLE_TABLE_LIST *vtlist, VALIDATE_VARIABLE_LIST *validate_variable_list, VALIDATE_STATEMENT_LIST *validate_statement_list, ASTPOINTER_LIST *ignore_ast_list, AST *target_expression, int undefined_control_check, int zero_divition_check, int array_unbound_check)
 - void **getValidateStatementFromMallocNumber** (VALIDATE_STATEMENT_LIST *validate_statement_list, AST *call_function, ARRAY_OFFSET_LIST *right_array_offset_list, MEMALLOC_INFO *memalloc_info)
 - void **getValidateStatementFromForIteration** (VALIDATE_STATEMENT_LIST *validate_statement_list, FOR_INFORMATION_LIST *for_information_list, FUNCTION_INFORMATION_LIST *function_information_list, VARIABLE_TABLE_LIST *vtlist, VALIDATE_VARIABLE_LIST *validate_variable_list, ASTPOINTER_LIST *ignore_ast_list, int undefined_control_check, int zero_divition_check, int array_unbound_check, int free_violation_check)
 - void **getValidateStatementFromAssignStatement** (AST *root, FUNCTION_INFORMATION_LIST *function_information_list, VARIABLE_TABLE_LIST *vtlist, VALIDATE_VARIABLE_LIST *validate_variable_list, VALIDATE_STATEMENT_LIST *validate_statement_list, ASTPOINTER_LIST *ignore_ast_list, AST *target_expression, int undefined_control_check, int zero_divition_check, int array_unbound_check, int free_violation_check)
 - void **getValidateStatementFromPointerOperator** (VALIDATE_STATEMENT_LIST *validate_statement_list, ARRAY_OFFSET_LIST *left_array_offset_list, ARRAY_OFFSET_LIST *right_array_offset_list, AST *right_expression, int a_op_flag)
 - void **getValidateStatementFromCallFunction** (AST *root, VALIDATE_STATEMENT_LIST *validate_statement_list, ARRAY_OFFSET_LIST *left_array_offset_list, ARRAY_OFFSET_LIST *right_array_offset_list, FUNCTION_INFORMATION_LIST *function_information_list)
 - void **getBasisLocationFromAssignmentExpression** (CSTLString *output, ARRAY_OFFSET_LIST *left_array_offset_list, ARRAY_OFFSET_LIST *right_array_offset_list, AST *right_expression_ast, int a_op_flag)
 - void **getBasisLocationFromExpression** (CSTLString *output, ARRAY_OFFSET *array_offset, AST *expression_ast)

- void **createValidateStatementForMallocAction** (VALIDATE_STATEMENT_LIST *validate_statement_list, MEMALLOC_INFO *memalloc_info, ARRAY_OFFSET_LIST *array_offset_list, VALIDATE_VARIABLE_LIST *validate_variable_list)
- void **createValidateStatementFromIncDecExpr** (VALIDATE_STATEMENT_LIST *validate_statement_list, ARRAY_OFFSET_LIST *array_offset_list)
- void **createValidateStatementForFreeAction** (VALIDATE_STATEMENT_LIST *validate_statement_list, FREEMEMINFO *freememinfo, VALIDATE_VARIABLE_LIST *validate_variable_list)
- void **createValidateStatementFromArrayDefine** (VALIDATE_VARIABLE_LIST *validate_variable_list, VALIDATE_STATEMENT_LIST *validate_statement_list, VARIABLE_TABLE_LIST *variable_table_list, FUNCTION_INFORMATION_LIST *function_information_list)
- void **createVaridateStatementFromPointerDefine** (VALIDATE_STATEMENT_LIST *validate_statement_list, VARIABLE_TABLE_LIST *variable_table_list, FUNCTION_INFORMATION_LIST *function_information_list)
- void **ArrayOffsetToValidateStatement** (CSTLString *output, VALIDATE_VARIABLE_LIST *validate_variable_list, VARIABLE_TABLE *variable_table, OFFSET_LIST *offset_list)
- void **createCheckUnboundAndUndefineOperationCheck** (VALIDATE_STATEMENT_LIST *validate_statement_list, ARRAY_OFFSET_LIST *array_offset_list, int array_unbound_check, int undefined_control_check)
- void **createViolentFreeOperation** (VALIDATE_STATEMENT_LIST *validate_statement_list, FREEMEMINFO *freememinfo)
- void **createZeroDivitionCheck** (VALIDATE_STATEMENT_LIST *validate_statement_list, DIVITION_INFORMATION_LIST *divition_information_list)
- int **getNewValidateStatementID** (VALIDATE_STATEMENT_LIST *validate_statement_list, AST *target_statement)
- void **getLeftAssignmentInfo** (AST *left_expression, FUNCTION_INFORMATION_LIST *function_information_list, VARIABLE_TABLE_LIST *vtlist, ASTPOINTER_LIST *ignore_ast_list, ARRAY_OFFSET_LIST *array_offset_list, AST *target_expression, int *switch_mode)
- void **getRightAssignmentInfo** (AST *root, FUNCTION_INFORMATION_LIST *function_information_list, VARIABLE_TABLE_LIST *vtlist, MEMALLOC_INFO **memalloc_info, ARRAY_OFFSET_LIST *array_offset_list, ASTPOINTER_LIST *ignore_ast_list, AST *target_statement)
- void **printProgramDataWithValidateStatement** (AST *root, VALIDATE_VARIABLE_LIST *validate_variable_list, VALIDATE_STATEMENT_LIST *validate_statement_list, FOR_INFORMATION_LIST *for_information_list)
- void **fprintProgramDataWithValidateStatement** (FILE *output, AST *root, VALIDATE_VARIABLE_LIST *validate_variable_list, VALIDATE_STATEMENT_LIST *validate_statement_list, FOR_INFORMATION_LIST *for_information_list)
- void **fprintValidateStatement** (FILE *output, VALIDATE_STATEMENT_LIST *validate_statement_list, AST *target_ast, int check_or_modify, int allow_output_used_statement)
- void **fprintValidateStatement_not_assert** (FILE *output, VALIDATE_STATEMENT_LIST *validate_statement_list, AST *target_ast, int check_or_modify, int allow_output_used_statement)

- void **setValidateVariableFromExprSlicing** (VALIDATE_VARIABLE_LIST *validate_variable_list, EXPR_SLICING_LIST *expr_slicing_list)
- void **fprintProgramDataWithPSIVaridateStatement** (FILE *output, EXPR_SLICING_LIST *expr_slicing_list, VALIDATE_STATEMENT_LIST *validate_statement_list, VALIDATE_VARIABLE_LIST *validate_variable_list, FOR_INFORMATION_LIST *for_information_list, AST *check_target_ast)
- void **createValidateStatementAdderFileEachCheck** (EXPR_SLICING_LIST *expr_slicing_list, VALIDATE_STATEMENT_LIST *validate_statement_list, VALIDATE_VARIABLE_LIST *validate_variable_list, FOR_INFORMATION_LIST *for_information_list, INCLUDE_LIST *include_list)
- void **VALIDATE_STATEMENT_LIST_sort_ast** (VALIDATE_STATEMENT_LIST *validate_statement_list)
- void **getASTList_FromVALIDATE_STATEMENT_LIST** (VALIDATE_STATEMENT_LIST *validate_statement_list, ASTPOINTER_LIST *ast_node_list)

4.15.1 説明

これはC言語プログラム上から、不具合を検証するための検証式や検証用に変数などを追加するための命令が含まれている。

作者

faithnh

4.15.2 型定義

4.15.2.1 typedef struct validate_statement VALIDATE_STATEMENT

実際に検証式として挿入するための情報である。

4.15.2.2 typedef struct validate_variable VALIDATE_VARIABLE

ポインタや配列変数に対する検証用の変数リストを作成するための構造体である。

4.15.3 関数

4.15.3.1 void ArrayOffsetToValidateStatement (CSTLString * output, VALIDATE_VARIABLE_LIST * validate_variable_list, VARIABLE_TABLE * variable_table, OFFSET_LIST * offset_list)

配列のオフセットリストを基に、検証式を作成する。

引数

<i>output</i>	出力する検証式
---------------	---------

<i>VALIDATE_- VARIABLE_- LIST</i>	検証用変数リスト
<i>variable_- table</i> (p. 15)	対象の変数データ
<i>offset_list</i>	オフセットリスト

戻り値

なし

配列のオフセットリストを基に、検証式を作成する。

引数

<i>output</i>	出力する検証式
<i>validate_- variable_list</i>	検証用変数リスト
<i>variable_- table</i> (p. 15)	対象の変数データ
<i>offset_list</i>	オフセットリスト

戻り値

なし

**4.15.3.2 void createCheckUnboundAndUndefineOperationCheck (VALIDATE_STATEMENT_LIST
* *validate_statement_list*, ARRAY_OFFSET_LIST * *array_offset_list*, int
array_unbound_check, int *undefined_control_check*)**

配列やポインタなどのオフセット情報のリスト *array_offset_list* から、配列の範囲外参照のチェックをするための検証式 や、未定義状態で処理をチェックするための検証式を生成し、VALIDATE_STATEMENT_LIST へ追加する。

引数

<i>VALIDATE_- STATEMENT_ LIST</i>	追加先の検証式リスト
<i>array_- offset_list</i>	配列やポインタなどのオフセット情報のリスト
<i>array_- unbound_- check</i>	配列が範囲外を参照していないかどうかを検証するための式を生成するかどうか 1 : 生成する 0 : 生成しない

<i>undefined_- control_- check</i>	未定義な処理を行っていないかどうかを検証するための式を生成するかどうか 1 : 生成する 0 : 生成しない
--	--

戻り値

なし

配列やポインタなどのオフセット情報のリスト *array_offset_list* から、配列の範囲外参照のチェックをするための検証式 や、未定義状態で処理をチェックするための検証式を生成し、*validate_statement_list* へ追加する。

引数

<i>validate_- statement_- list</i>	追加先の検証式リスト
<i>array_- offset_list</i>	配列やポインタなどのオフセット情報のリスト
<i>array_- unbound_- check</i>	配列が範囲外を参照していないかどうかを検証するための式を生成するかどうか 1 : 生成する 0 : 生成しない
<i>undefined_- control_- check</i>	未定義な処理を行っていないかどうかを検証するための式を生成するかどうか 1 : 生成する 0 : 生成しない

戻り値

なし

**4.15.3.3 void createValidateStatemenFromIncDecExpr (VALIDATE_STATEMENT_LIST *
validate_statement_list, ARRAY_OFFSET_LIST * array_offset_list)**

指定した配列オフセット *array_offset_list* から、インクリメントおよびデクリメント式を含んでいたら、それに応じて *basis_location* に反映させるための検証式を生成し、*VALIDATE_STATEMENT_LIST* に追加する。

引数

<i>VALIDATE_- STATEMENT_ LIST</i>	追加先の検証式
<i>array_- offset_list</i>	配列オフセットリスト

戻り値

なし

指定した配列オフセットリスト `array_offset_list` から、インクリメントおよびデクリメント式を含んでいたら、それに応じて `basis_location` に反映させるための検証式を生成し、`validate_statement_list` に追加する。

引数

<code>validate_statement_list</code>	追加先の検証式
<code>array_offset_list</code>	配列オフセットリスト

戻り値

なし

4.15.3.4 `void createValidateStatement (AST * root, FUNCTION_INFORMATION_LIST * function_information_list, VARIABLE_TABLE_LIST * vtlist, VALIDATE_VARIABLE_LIST * validate_variable_list, VALIDATE_STATEMENT_LIST * validate_statement_list, FOR_INFORMATION_LIST * for_information_list, int undefined_control_check, int zero_divition_check, int array_unbound_check, int free_violation_check)`

基本的な検証式の生成を行う。

引数

<code>root</code>	検証式生成対象の AST ノード
<code>function_information_list</code>	関数に関する情報のリスト
<code>vtlist</code>	検証対象の式をマークするための変数リスト
<code>VALIDATE_STATEMENT_LIST</code>	検証用変数リスト
<code>VALIDATE_VARIABLE_LIST</code>	取得した検証式が格納するところ
<code>for_information_list</code>	for 文に関する情報
<code>undefined_control_check</code>	未定義な処理（未定義ポインタの参照など）を行っていないかどうかを検証するための式を生成するかどうか 1：生成する 0：生成しない

<i>zero_-divition_-check</i>	0 で割っていないかどうかを検証するための式を生成するかどうか 1 : 生成する 0 : 生成しない
<i>array_-unbound_-check</i>	配列が範囲外を参照していないかどうかを検証するための式を生成するかどうか 1 : 生成する 0 : 生成しない
<i>free_-violation_-check</i>	メモリ解放関係で不正な処理を行っていないかどうかを検証するための式を生成するかどうか 1 : 生成する 0 : 生成しない

戻り値

なし

基本的な検証式の生成を行う。

引数

<i>root</i>	検証式生成対象の AST ノード
<i>function_-informaiton_-list</i>	関数に関する情報のリスト
<i>vtlist</i>	検証対象の式をマークするための変数リスト
<i>validate_-statement_-list</i>	検証用変数リスト
<i>validate_-variable_-list</i>	取得した検証式が格納するところ
<i>for_-information_-list</i>	for 文に関する情報
<i>undefined_-control_-check</i>	未定義な処理（未定義ポインタの参照など）を行っていないかどうかを検証するための式を生成するかどうか 1 : 生成する 0 : 生成しない
<i>zero_-divition_-check</i>	0 で割っていないかどうかを検証するための式を生成するかどうか 1 : 生成する 0 : 生成しない
<i>array_-unbound_-check</i>	配列が範囲外を参照していないかどうかを検証するための式を生成するかどうか 1 : 生成する 0 : 生成しない
<i>free_-violation_-check</i>	メモリ解放関係で不正な処理を行っていないかどうかを検証するための式を生成するかどうか 1 : 生成する 0 : 生成しない

戻り値

なし


```
4.15.3.5 void createValidateStatementAdderFileEachCheck ( EXPR_SLICING_LIST
* expr_slicing_list, VALIDATE_STATEMENT_LIST * validate_statement_list,
VALIDATE_VARIABLE_LIST * validate_variable_list, FOR_INFORMATION_LIST *
for_information_list, INCLUDE_LIST * include_list )
```

プログラムをチェック式ごとにプログラムスライシングと検証式付加を行ったファイル生成する。

引数

<i>expr_slicing_list</i>	プログラムスライシング情報
<i>validate_variable_list</i>	検証用変数リスト
<i>validate_statement_list</i>	検証式リスト
<i>for_information_list</i>	for 文に関する情報
<i>include_list</i>	インクルードリスト

戻り値

なし

```
4.15.3.6 void createValidateStatementForFreeAction ( VALIDATE_STATEMENT_LIST
* validate_statement_list, FREEMEMINFO * freememinfo,
VALIDATE_VARIABLE_LIST * validate_variable_list )
```

メモリ解放関係の情報 *freememinfo* から検証式を生成し、VALIDATE_STATEMENT_LIST に追加する。

引数

<i>validate_statement_list</i>	生成先の検証式リスト
<i>freememinfo</i>	メモリ確保関係の情報
VALIDATE_VARIABLE_LIST	検証用変数リスト

戻り値

なし

メモリ解放関係の情報 *freememinfo* から検証式を生成し、*validate_statement_list* に追加する。

引数

<i>variable_-statement_-list</i>	生成先の検証式リスト
<i>freememinfo</i>	メモリ確保関係の情報
<i>validate_-variable_-list</i>	検証用変数リスト

戻り値

なし

4.15.3.7 void createValidateStatementForMallocAction (VALIDATE_STATEMENT_LIST * *validate_statement_list*, MEMALLOC_INFO * *memalloc_info*, ARRAY_OFFSET_LIST * *array_offset_list*, VALIDATE_VARIABLE_LIST * *validate_variable_list*)

メモリ確保関係の情報 *memalloc_info* や配列やポインタのオフセット情報 *array_offset_list* から検証式を生成し、VALIDATE_STATEMENT_LIST に追加する。

引数

<i>variable_-statement_-list</i>	生成先の検証式リスト
<i>memalloc_-info</i>	メモリ確保関係の情報
<i>array_-offset_-list</i>	配列やポインタのオフセット情報
VALIDATE_-VARIABLE_-LIST	検証用変数リスト

戻り値

なし

メモリ確保関係の情報 *memalloc_info* や配列やポインタのオフセット情報 *array_offset_list* から検証式を生成し、*validate_statement_list* に追加する。

引数

<i>variable_-statement_-list</i>	生成先の検証式リスト
<i>memalloc_-info</i>	メモリ確保関係の情報

<i>array_offset_list</i>	配列やポインタのオフセット情報
<i>validate_variable_list</i>	検証用変数リスト

戻り値

なし

4.15.3.8 void createValidateStatementFromArrayDefine (VALIDATE_VARIABLE_LIST *
validate_variable_list, VALIDATE_STATEMENT_LIST * validate_statement_list,
VARIABLE_TABLE_LIST * variable_table_list, FUNCTION_INFORMATION_LIST *
function_information_list)

変数定義リストで、配列生成時の検証用変数の更新するための検証式を作成し、
validate_statement_list に追加する。また、検証用変数リスト validate_variable_list
に、配列のオフセットを生成するのに使用する変数 vviterator_2 ~ vviterator_n (n
は配列の最大次元数) を生成し、追加する。

引数

<i>validate_variable_list</i>	検証用変数リスト
<i>validate_statement_list</i>	追加先の検証式
<i>variable_table_list</i>	変数テーブルリスト
<i>function_information_list</i>	関数定義に関する情報リスト

戻り値

なし

4.15.3.9 void createVaridateStatementFromPointerDefine (VALIDATE_STATEMENT_LIST
* validate_statement_list, VARIABLE_TABLE_LIST * variable_table_list,
FUNCTION_INFORMATION_LIST * function_information_list)

変数定義リストでポインタ変数に対する検証用変数を更新するための検証式を作
成し、validate_statement_list に追加する。

引数

<i>validate_statement_list</i>	追加先の検証式
--------------------------------	---------

<i>variable_- table_list</i>	変数テーブルリスト
<i>function_- information_ list</i>	関数定義に関する情報リスト

戻り値

なし

**4.15.3.10 void createViolentFreeOperation (VALIDATE_STATEMENT_LIST *
validate_statement_list, FREEMEMINFO * freememinfo)**

メモリ解放関係の情報 freememinfo から、free 関数に関する違反行為を行っていないかどうかをチェックするための検証式を生成し、VALIDATE_STATEMENT_LIST へ追加する。

引数

<i>VALIDATE_- STATEMENT_ LIST</i>	追加先の検証式リスト
<i>freememinfo</i>	メモリ解放関係の情報

戻り値

なし

メモリ解放関係の情報 freememinfo から、free 関数に関する違反行為を行っていないかどうかをチェックするための検証式を生成し、validate_statement_list へ追加する。

引数

<i>validate_- statement_- list</i>	追加先の検証式リスト
<i>freememinfo</i>	メモリ解放関係の情報

戻り値

なし

4.15.3.11 void createZeroDivitionCheck (VALIDATE_STATEMENT_LIST * *validate_statement_list*,
DIVITION_INFORMATION_LIST * *divition_information_list*)

除算および剰余式の情報から、ゼロ除算および剰余になっていないかどうかの検証式を生成する。

引数

<i>VALIDATE_STATEMENT_LIST</i>	格納先の検証式リスト
<i>divition_information_list</i>	対象の除算および剰余式の情報

戻り値

なし

除算および剰余式の情報から、ゼロ除算および剰余になっていないかどうかの検証式を生成する。

引数

<i>validate_statement_list</i>	格納先の検証式リスト
<i>divition_information_list</i>	対象の除算および剰余式の情報

戻り値

なし

4.15.3.12 void fprintProgramDataWithPSIVaridateStatement (FILE * *output*,
EXPR_SLICING_LIST * *expr_slicing_list*, VALIDATE_STATEMENT_LIST *
validate_statement_list, VALIDATE_VARIABLE_LIST * *validate_variable_list*,
FOR_INFORMATION_LIST * *for_information_list*, AST * *check_target_ast*)

プログラムスライシング情報をもとに検証式を追加しながら出力させる。

引数

<i>output</i>	出力する先のファイル
<i>expr_slicing_list</i>	プログラムスライシング情報
<i>validate_variable_list</i>	検証用変数リスト

<i>validate_-statement_-list</i>	検証式リスト
<i>for_-information_-list</i>	for 文に関する情報
<i>check_-target_ast</i>	チェック検証式の対象への AST ノード

戻り値

なし

4.15.3.13 void fprintProgramDataWithValidateStatement (FILE * *output*, AST * *root*, VALIDATE_VARIABLE_LIST * *validate_variable_list*, VALIDATE_STATEMENT_LIST * *validate_statement_list*, FOR_INFORMATION_LIST * *for_information_list*)

検証式リストや検証用変数をもとにプログラムデータを生成し、指定したファイル *output* に出力する。

引数

<i>output</i>	出力先のファイル構造体
<i>root</i>	プログラムへの AST ノード
<i>VALIDATE_-VARIABLE_-LIST</i>	検証用変数リスト
<i>VALIDATE_-STATEMENT_-LIST</i>	検証式リスト
<i>for_-information_-list</i>	for 文に関する情報のリスト

戻り値

なし

検証式リストや検証用変数をもとにプログラムデータを生成し、指定したファイル *output* に出力する。

引数

<i>output</i>	出力先のファイル構造体
<i>root</i>	プログラムへの AST ノード
<i>validate_-variable_list</i>	検証用変数リスト

<i>validate_statement_list</i>	検証式リスト
<i>for_information_list</i>	for 文に関する情報のリスト

戻り値

なし

4.15.3.14 void fprintfValidateStatement (FILE * *output*, VALIDATE_STATEMENT_LIST * *validate_statement_list*, AST * *target_ast*, int *check_or_modify*, int *allow_output_used_statement*)

式に対応する検証式を出力させる。

引数

<i>output</i>	出力先のファイル構造体
<i>VALIDATE_STATEMENT_LIST</i>	出力対象の検証式リスト
<i>target_ast</i>	対象の AST ノード
<i>check_or_modify</i>	検証式をチェックするタイプか、プログラムを元に編集するタイプかを判断するフラグ。0 : チェックするタイプ、1 : 編集するタイプ
<i>allow_output_used_statement</i>	使用済みの検証式も含めて出力するかどうかのフラグ 0 : 出力しない 1 : 出力する

戻り値

なし

式に対応する検証式を出力させる。

引数

<i>output</i>	出力先のファイル構造体
<i>validate_statement_list</i>	出力対象の検証式リスト
<i>target_ast</i>	対象の AST ノード
<i>check_or_modify</i>	検証式をチェックするタイプか、プログラムを元に編集するタイプかを判断するフラグ。0 : チェックするタイプ、1 : 編集するタイプ

<i>allow_ output_ used_ statement</i>	使用済みの検証式も含めて出力するかどうかのフラグ 0 : 出力しない 1 : 出力する
---	---

戻り値

なし

4.15.3.15 void fprintfValidateStatement_not_assert (FILE * *output*, VALIDATE_STATEMENT_LIST * *validate_statement_list*, AST * *target_ast*, int *check_or_modify*, int *allow_output_used_statement*)

式に対応する検証式を assert(0); を削除したうえ出力させる。

引数

<i>output</i>	出力先のファイル構造体
<i>validate_ statement_ list</i>	出力対象の検証式リスト
<i>target_ast</i>	対象の AST ノード
<i>check_or_ modify</i>	検証式をチェックするタイプか、プログラムを元に編集するタイプかを判断するフラグ。 0 : チェックするタイプ、 1 : 編集するタイプ
<i>allow_ output_ used_ statement</i>	使用済みの検証式も含めて出力するかどうかのフラグ 0 : 出力しない 1 : 出力する

戻り値

なし

4.15.3.16 void getASTList_FromVALIDATE_STATEMENT_LIST (VALIDATE_STATEMENT_LIST * *validate_statement_list*, ASTPOINTER_LIST * *ast_node_list*)

検証式リストのチェック式から、AST ノードを取り出し、AST リストとしてまとめる。

引数

<i>validate_ statement_ list</i>	取り出し先の検証式リスト
<i>ast_node_ list</i>	まとめる先の AST ノードリスト

戻り値

なし

4.15.3.17 void getBasisLocationFromAssignmentExpression (CSTLString * *output*,
ARRAY_OFFSET_LIST * *left_array_offset_list*, ARRAY_OFFSET_LIST *
right_array_offset_list, AST * *right_expression_ast*, int *a_op_flag*)

ポインタ演算式に対して、ポインタ演算における基本的な位置の式を文字列として求め、output に入れる。

引数

<i>output</i>	出力先の CSTL 文字列
<i>left_array_offset_list</i>	左辺値の配列オフセットリスト
<i>right_array_offset_list</i>	右辺式の配列オフセットリスト
<i>right_expression_ast</i>	右辺式への AST アドレス
<i>a_op_flag</i>	代入演算子が何かを示すフラグ 0: =, 1: +=, 2: -=, 3: *=, 4: /=, 5: =, 6: <=, 7: >=, 8: &=, 9: =, 10: ^=

戻り値

なし

4.15.3.18 void getBasisLocationFromExpression (CSTLString * *output*, ARRAY_OFFSET
* *array_offset*, AST * *expression_ast*)

指定した式 *expression_ast* から、ポインタ演算における基本的な位置の式を文字列として求め、output に入れる。このとき、*array_offset* を見つけたらそれに該当する式を 0 に変換する。

引数

<i>output</i>	出力先の CSTL 文字列
<i>array_offset</i> (p. 8)	指定した識別子の配列オフセット
<i>expression_ast</i>	指定した式への AST アドレス

戻り値

なし

4.15.3.19 void getLeftAssignmentInfo (AST * *left_expression*, FUNCTION_INFORMATION_LIST * *function_information_list*, VARIABLE_TABLE_LIST * *vtlist*, ASTPOINTER_LIST * *ignore_ast_list*, ARRAY_OFFSET_LIST * *array_offset_list*, AST * *target_expression*, int * *switch_mode*)

代入式の左辺値について、検証式に必要な情報を取得する。

引数

<i>left_expression</i>	左辺値に関する AST ノード
<i>function_information_list</i>	関数に関する情報のリスト
<i>vtlist</i>	検証対象の式をマークするための変数リスト
<i>ignore_ast_list</i>	ポインタでの位置が検証済みである、IDENTIFIER を無視するための AST のアドレスリスト
<i>array_offset_list</i>	各ポインタおよび配列ごとのオフセットのリスト
<i>target_expression</i>	この左辺式の上位に位置する AST ノード
<i>switch_mode</i>	直接アクセスおよび配列アクセスを探すか、IDENTIFIER を探すかどうかのスイッチフラグ 0 : 両方さがす 1 : direct_ref や array_access のみ探す

戻り値

なし

4.15.3.20 int getNewValidateStatementID (VALIDATE_STATEMENT_LIST * *validate_statement_list*, AST * *target_statement*)

検証式リスト VALIDATE_STATEMENT_LIST から、*target_statement* と同じ AST のアドレスを持ったものを探し出し、それを基に重複しないようにするための新しい検証式の識別番号を取得する。新しい検証式を作るにはこの関数から新しい識別番号を取得すること。

引数

<i>VALIDATE_STATEMENT_LIST</i>	対象の検証式リスト
<i>target_statement</i>	対象の検証式から確認するための AST のアドレス

戻り値

新しい識別番号を出力する。すでに同じ AST のアドレスを持っている検証式

がなければ 0 を返す。

検証式リスト `validate_statement_list` から、`target_statement` と同じ AST のアドレスを持ったものを探し出し、それを基に重複しないようにするための新しい検証式の識別番号を取得する。新しい検証式を作るにはこの関数から新しい識別番号を取得すること。

引数

<code>validate_statement_list</code>	対象の検証式リスト
<code>target_statement</code>	対象の検証式から確認するための AST のアドレス

戻り値

新しい識別番号を出力する。すでに同じ AST のアドレスを持っている検証式がなければ 0 を返す。

4.15.3.21 `void getRightAssignmentInfo (AST * root, FUNCTION_INFORMATION_LIST * function_information_list, VARIABLE_TABLE_LIST * vtlist, MEMALLOC_INFO ** memalloc_info, ARRAY_OFFSET_LIST * array_offset_list, ASTPOINTER_LIST * ignore_ast_list, AST * target_statement)`

代入式の右辺式について、検証式に必要な情報を取得する。

引数

<code>root</code>	右辺式に関する AST ノード
<code>function_information_list</code>	関数に関する情報のリスト
<code>vtlist</code>	メモリ確保情報を取得するのに必要なプログラム変数リスト
<code>memalloc_info</code>	malloc 関係の情報が出力される
<code>array_offset_list</code>	左辺式上にあるポインタ参照に対するオフセットリスト
<code>ignore_ast_list</code>	同じ位置のポインタが来ても無視するためのリスト
<code>target_statement</code>	この計算式を属している AST ノードへのアドレス (基本的に <code>expression_statement</code> であるノードが入る)

戻り値

なし

4.15.3.22 void getValidate_Variable (VARIABLE_TABLE_LIST * *variable_table_list*,
VALIDATE_VARIABLE_LIST * *validate_variable*)

プログラムの変数リストをもとにプログラムの検証用の変数を設定する。

引数

<i>variable_table_list</i>	プログラムの変数リスト
<i>validate_variable</i> (p. 14)	検証用の変数リスト

戻り値

なし

4.15.3.23 void getValidateStatementFromAssignStatement (AST * *root*,
FUNCTION_INFORMATION_LIST * *function_information_list*, VARIABLE_TABLE_LIST
* *vtlist*, VALIDATE_VARIABLE_LIST * *validate_variable_list*,
VALIDATE_STATEMENT_LIST * *validate_statement_list*, ASTPOINTER_LIST *
ignore_ast_list, AST * *target_expression*, int *undefined_control_check*, int
zero_divition_check, int *array_unbound_check*, int *free_violation_check*)

指定した AST ノード *root* から、*assign_expression* を探しだし、そこから VARIDATE_STATEMENT に関する情報を取得する。

引数

<i>root</i>	指定したノード
<i>function_information_list</i>	関数に関する情報のリスト
<i>vtlist</i>	対象の変数リスト
<i>validate_variable_list</i>	検証用変数リスト
<i>validate_statement_list</i>	取得した検証式が格納するところ
<i>ignore_ast_list</i>	ポインタでの位置が検証済みである、IDENTIFIER を無視するための AST のアドレスリスト
<i>target_expression</i>	<i>assign_expression</i> が属している <i>expression_statement</i>
<i>undefined_control_check</i>	未定義な処理（未定義ポインタの参照など）を行っていないかどうかを検証するための式を生成するかどうか 1：生成する 0：生成しない
<i>zero_divition_check</i>	0 で割っていないかどうかを検証するための式を生成するかどうか 1：生成する 0：生成しない

<i>array_unbound_check</i>	配列が範囲外を参照していないかどうかを検証するための式を生成するかどうか 1 : 生成する 0 : 生成しない
<i>free_violation_check</i>	メモリ解放関係で不正な処理を行っていないかどうかを検証するための式を生成するかどうか 1 : 生成する 0 : 生成しない

戻り値

なし

```
4.15.3.24 void getValidateStatementFromCallFunction ( AST * root,
VALIDATE_STATEMENT_LIST * validate_statement_list, ARRAY_OFFSET_LIST
* left_array_offset_list, ARRAY_OFFSET_LIST * right_array_offset_list,
FUNCTION_INFORMATION_LIST * function_information_list )
```

式から、関数呼出を探しだし、関数呼出に対する検証式を追加する。

引数

<i>root</i>	探索対象の AST ノード
<i>validate_statement_list</i>	検証式リスト
<i>left_array_offset_list</i>	左辺値の配列オフセットリスト
<i>right_array_offset_list</i>	右辺式の配列オフセットリスト
<i>function_information_list</i>	関数に関する情報リスト

戻り値

なし

```
4.15.3.25 void getValidateStatementFromForIteration ( VALIDATE_STATEMENT_LIST *
validate_statement_list, FOR_INFORMATION_LIST * for_information_list,
FUNCTION_INFORMATION_LIST * function_information_list, VARIABLE_TABLE_LIST
* vtlist, VALIDATE_VARIABLE_LIST * validate_variable_list, ASTPOINTER_LIST
* ignore_ast_list, int undefined_control_check, int zero_divition_check, int
array_unbound_check, int free_violation_check )
```

for 文の末尾の情報から、検証式を取得し、検証式リストに入れる。

引数

<i>validate_statement_list</i>	取得した検証式が格納するところ
<i>for_information_list</i>	for 文に関する情報のリスト
<i>function_information_list</i>	関数に関する情報のリスト
<i>vtlist</i>	対象の変数リスト
<i>validate_variable_list</i>	検証用変数リスト
<i>ignore_ast_list</i>	ポインタでの位置が検証済みである、IDENTIFIER を無視するための AST のアドレスリスト
<i>undefined_control_check</i>	未定義な処理を行っていないかどうかを検証するための式を生成するかどうか 1 : 生成する 0 : 生成しない
<i>zero_divition_check</i>	0 で割っていないかどうかを検証するための式を生成するかどうか 1 : 生成する 0 : 生成しない
<i>array_unbound_check</i>	配列が範囲外を参照していないかどうかを検証するための式を生成するかどうか 1 : 生成する 0 : 生成しない
<i>free_violation_check</i>	メモリ解放関係で不正な処理を行っていないかどうかを検証するための式を生成するかどうか 1 : 生成する 0 : 生成しない

戻り値

なし

4.15.3.26 void getValidateStatementFromInitializer (AST * root, FUNCTION_INFORMATION_LIST * function_information_list, VARIABLE_TABLE_LIST * vtlist, VALIDATE_VARIABLE_LIST * validate_variable_list, VALIDATE_STATEMENT_LIST * validate_statement_list, ASTPOINTER_LIST * ignore_ast_list, AST * target_expression, int undefined_control_check, int zero_divition_check, int array_unbound_check)

指定した AST ノード root から、init_declarator を探しだし、そこから VARIDATE_STATEMENT に関する情報を取得する。

引数

<i>root</i>	指定したノード
<i>function_information_list</i>	関数に関する情報のリスト

<i>vtlist</i>	対象の変数リスト
<i>VALIDATE_- VARIABLE_- LIST</i>	検証用変数リスト
<i>VALIDATE_- STATEMENT_ LIST</i>	取得した検証式が格納するところ
<i>ignore_ast_- list</i>	ポインタでの位置が検証済みである、IDENTIFIER を無視するための AST のアドレスリスト
<i>target_- expression</i>	assign_expression が属している expression_statement
<i>undefined_- control_- check</i>	未定義な処理を行っていないかどうかを検証するための式を生成するかどうか 1 : 生成する 0 : 生成しない
<i>zero_- division_- check</i>	0 で割っていないかどうかを検証するための式を生成するかどうか 1 : 生成する 0 : 生成しない
<i>array_- unbound_- check</i>	配列が範囲外を参照していないかどうかを検証するための式を生成するかどうか 1 : 生成する 0 : 生成しない

戻り値

なし

指定した AST ノード root から、init_declarator を探しだし、そこから VARIDATE_STATEMENT に関する情報を取得する。

引数

<i>root</i>	指定したノード
<i>function_- information_- list</i>	関数に関する情報のリスト
<i>vtlist</i>	対象の変数リスト
<i>validate_- variable_list</i>	検証用変数リスト
<i>validate_- statement_- list</i>	取得した検証式が格納するところ
<i>ignore_ast_- list</i>	ポインタでの位置が検証済みである、IDENTIFIER を無視するための AST のアドレスリスト
<i>target_- expression</i>	assign_expression が属している expression_statement
<i>undefined_- control_- check</i>	未定義な処理を行っていないかどうかを検証するための式を生成するかどうか 1 : 生成する 0 : 生成しない

<i>zero_</i> <i>division_</i> <i>check</i>	0 で割っていないかどうかを検証するための式を生成するかどうか 1 : 生成する 0 : 生成しない
<i>array_</i> <i>unbound_</i> <i>check</i>	配列が範囲外を参照していないかどうかを検証するための式を生成するかどうか 1 : 生成する 0 : 生成しない

戻り値

なし

4.15.3.27 void `getValidateStatementFromMallocNumber` (`VALIDATE_STATEMENT_LIST`
* *validate_statement_list*, `AST` * *call_function*, `ARRAY_OFFSET_LIST` *
right_array_offset_list, `MEMALLOC_INFO` * *memalloc_info*)

malloc 用識別番号を付加するための関数に変換するための検証式を追加させる。

引数

<i>VALIDATE_</i> <i>STATEMENT_</i> <i>LIST</i>	追加先の検証式リスト
<i>call_</i> <i>function</i>	関数呼び出しに対するノード
<i>right_</i> <i>array_</i> <i>offset_list</i>	左辺値に関する配列オフセットリスト
<i>memalloc_</i> <i>info</i>	メモリ確保情報

戻り値

なし

malloc 用識別番号を付加するための関数に変換するための検証式を追加させる。

引数

<i>validate_</i> <i>statement_</i> <i>list</i>	追加先の検証式リスト
<i>call_</i> <i>function</i>	関数呼び出しに対するノード
<i>right_</i> <i>array_</i> <i>offset_list</i>	左辺値に関する配列オフセットリスト
<i>memalloc_</i> <i>info</i>	メモリ確保情報

戻り値

なし

```
4.15.3.28 void getValidateStatementFromPointerOperator ( VALIDATE_STATEMENT_LIST
               * validate_statement_list, ARRAY_OFFSET_LIST * left_array_offset_list,
               ARRAY_OFFSET_LIST * right_array_offset_list, AST * right_expression, int a_op_flag
               )
```

ポインタ演算式後の内容を検証用変数に反映するための検証式を追加する。

引数

<i>VALIDATE_STATEMENT_LIST</i>	検証式リスト
<i>left_array_offset_list</i>	左辺値の配列オフセットリスト
<i>right_array_offset_list</i>	右辺式の配列オフセットリスト
<i>right_expression</i>	右辺式へのASTアドレス
<i>a_op_flag</i>	代入演算子が何かを示すフラグ 0:=,1:+=,2:-=,3:*=,4:/=,5:=,6:<=,7:>=,8:&=,9: =,10:^=

戻り値

なし

ポインタ演算式後の内容を検証用変数に反映するための検証式を追加する。

引数

<i>validate_statement_list</i>	検証式リスト
<i>left_array_offset_list</i>	左辺値の配列オフセットリスト
<i>right_array_offset_list</i>	右辺式の配列オフセットリスト
<i>right_expression</i>	右辺式へのASTアドレス
<i>a_op_flag</i>	代入演算子が何かを示すフラグ 0:=,1:+=,2:-=,3:*=,4:/=,5:=,6:<=,7:>=,8:&=,9: =,10:^=

戻り値

なし

4.15.3.29 void initVALIDATE_STATEMENT_flag (VALIDATE_STATEMENT_LIST *
validate_statement_list)

検証式リストの使用フラグを未使用状態に初期化する。

引数

validate_ statement_ list	* 初期化対象の検証式リスト
---------------------------------	----------------

戻り値

なし

4.15.3.30 VALIDATE_STATEMENT* new_VALIDATE_STATEMENT (int target_id, int
check_or_modify, int used, CSTLString * statement, AST * target_statement)

実際に検証式として挿入するための情報を生成する。

引数

target_id	この検証式の識別 ID(どの順序でこの検証式を入れていくかを確認するための ID)
check_or_ modify	検証式をチェックするタイプか、プログラムを元に編集するタイプかを判断するフラグ。0 : チェックするタイプ、1 : 編集するタイプ
used	この検証式は使用しているかどうかのフラグ 1:使用 0:未使用
statement	この検証式の内容
target_ statement	この検証式のターゲットとなる AST ノードへのアドレス

戻り値

実際に検証式として挿入するための情報へのアドレスを返す。

4.15.3.31 VALIDATE_STATEMENT* new_VALIDATE_STATEMENT_char (int target_id,
int check_or_modify, int used, char * statement, AST * target_statement)

実際に検証式として挿入するための情報を生成する。

引数

target_id	この検証式の識別 ID(どの順序でこの検証式を入れていくかを確認するための ID)
-----------	---

<i>check_or_modify</i>	検証式をチェックするタイプか、プログラムを元に編集するタイプかを判断するフラグ。 0 : チェックするタイプ、 1 : 編集するタイプ
<i>used</i>	この検証式は使用しているかどうかのフラグ 1:使用 0:未使用
<i>statement</i>	この検証式の内容
<i>target_statement</i>	この検証式のターゲットとなる AST ノードへのアドレス

戻り値

実際に検証式として挿入するための情報へのアドレスを返す。

4.15.3.32 VALIDATE_VARIABLE* new_VALIDATE_VARIABLE (int *used*, int *enable_start*, int *enable_end*, int *declaration_location*, int *block_level*, int *block_id*, CSTLString * *type*, CSTLString * *variable_name*, CSTLString * *target_variable_name*, int *offset_level*)

新しい検証用変数テーブルのデータを生成させる。

引数

<i>used</i>	この検証用変数テーブルを使用したかどうか
<i>enable_start</i>	この変数の有効範囲の始まりの行数
<i>enable_end</i>	この変数の有効範囲の終わりの行数
<i>declaration_location</i>	この変数を宣言した場所の行数
<i>block_level</i>	この変数のブロックレベル (グローバル変数なら 0 とし、関数の中での定義なら 1、その関数内の for 文などのブロック文ないでの宣言なら 2 とする)
<i>block_id</i>	ブロックごとの ID (基本的には 0 から始まり、ブロックレベル 2 が 2 回目にくると、 1 となる)
<i>type</i>	型名
<i>variable_name</i>	変数名
<i>target_variable_name</i>	検証対象の変数名
<i>offset_level</i>	この変数の配列やポインタの次元レベル

戻り値

新しく生成された検証用変数のデータへのアドレスが返される。

4.15.3.33 `VALIDATE_VARIABLE* new_VALIDATE_VARIABLE_with_char (int used, int enable_start, int enable_end, int declaration_location, int block_level, int block_id, char * type, char * variable_name, char * target_variable_name, int offset_level)`

新しい検証用変数テーブルのデータを生成させる (char 対応版)。

引数

<i>used</i>	この検証用変数テーブルを使用したかどうか
<i>enable_start</i>	この変数の有効範囲の始まりの行数
<i>enable_end</i>	この変数の有効範囲の終わりの行数
<i>declaration_location</i>	この変数を宣言した場所の行数
<i>block_level</i>	この変数のブロックレベル (グローバル変数なら 0 とし、関数の中での定義なら 1、その関数内の for 文などのブロック文ないでの宣言なら 2 とする)
<i>block_id</i>	ブロックごとの ID (基本的には 0 から始まり、ブロックレベル 2 が 2 回目になると、1 となる)
<i>type</i>	型名
<i>variable_name</i>	変数名
<i>target_variable_name</i>	検証対象の変数名
<i>offset_level</i>	この変数の配列やポインタの次元レベル

戻り値

新しく生成された検証用変数のデータへのアドレスが返される。

4.15.3.34 `void printProgramDataWithValidateStatement (AST * root, VALIDATE_VARIABLE_LIST * validate_variable_list, VALIDATE_STATEMENT_LIST * validate_statement_list, FOR_INFORMATION_LIST * for_information_list)`

検証式リストとともにプログラムデータを出力する。

引数

<i>root</i>	プログラムへの AST ノード
<i>VALIDATE_VARIABLE_LIST</i>	検証用変数リスト
<i>VALIDATE_STATEMENT_LIST</i>	検証式リスト

<i>for_- information_- list</i>	for 文に関する情報のリスト
---	-----------------

戻り値

なし

検証式リストとともにプログラムデータを出力する。

引数

<i>root</i>	プログラムへの AST ノード
<i>validate_- variable_list</i>	検証用変数リスト
<i>validate_- statement_- list</i>	検証式リスト
<i>for_- information_- list</i>	for 文に関する情報のリスト

戻り値

なし

4.15.3.35 void printVALIDATE_VARIABLE_LIST (VALIDATE_VARIABLE_LIST *
validate_variable_list)

検証用変数テーブルのリストの内容を出力させる。

引数

<i>validate_- variable_list</i>	出力対象の検証用変数テーブルのリスト
-------------------------------------	--------------------

戻り値

なし

4.15.3.36 void setValidateVariableFromExprSlicing (VALIDATE_VARIABLE_LIST *
validate_variable_list, EXPR_SLICING_LIST * expr_slicing_list)

プログラムスライシング情報の変数定義をもとに、検証用変数リストの出力を設定する。

引数

<i>validate_- variable_list</i>	検証用変数リスト
<i>expr_- slicing_list</i>	プログラムスライシング情報のリスト

戻り値

なし

**4.15.3.37 void VALIDATE_STATEMENT_LIST_sort_ast (VALIDATE_STATEMENT_LIST *
validate_statement_list)**

検証式リストを AST ノードごとにソートする。

引数

<i>validate_- statement_- list</i>	ソート対象の検証式リスト
--	--------------

戻り値

なし

4.16 Library/CharStringExtend.h

このファイルは、string.h にないような処理を実現させるための命令が含まれている。具体的には文字列からある場所からある場所までを抽出させるといったことができる。

関数

- int **str_extract** (char *dest, char *source, int start, int str_length)
- int **isExpression** (char *source)

4.16.1 説明

このファイルは、string.h にないような処理を実現させるための命令が含まれている。具体的には文字列からある場所からある場所までを抽出させるといったことができる。

作者

faithnh

4.16.2 関数

4.16.2.1 int isExpression (char * source)

文字列 source は式であるかどうかを調べる。

引数

<i>source</i>	対象の文字列
---------------	--------

戻り値

式である場合は 1、そうでない場合は 0 を返す。

文字列 source は式または識別子であるかどうかを調べる。

引数

<i>source</i>	対象の文字列
---------------	--------

戻り値

式または識別子である場合は 1、そうでない場合は 0 を返す。

4.16.2.2 int str_extract (char * dest, char * source, int start, int str_length)

文字列 source から開始文字数 start から指定された文字数 strlen 分の文字列を抽出し、その結果を文字列 dest へ入れる。また終端子も付くので、抜き出した文字 + 1 個分の文字列が必要

引数

<i>dest</i>	抽出結果を入れる文字列
<i>source</i>	抽出対象の文字列
<i>start</i>	抽出の開始位置
<i>str_length</i>	抽出する文字数

戻り値

成功したかどうかのフラグ 成功した場合は 1 途中で抽出に失敗した場合は 0 を返す。

4.17 Library/CSTLString.h

このファイルは、CSTL ライブラリを用いた文字列型 CSTLString を生成したり、それに関する操作を行うための命令が含まれている。

```
#include <cstl/string.h>
```

関数

- void **CSTLString_replace_string** (CSTLString *target, char *search, char *replace)
- int **CSTLString_compare_with_char** (CSTLString *target1, char *target2)
- int **CSTLString_delete_tail_str** (CSTLString *target1, char *del_str)
- void **CSTLString_printf** (CSTLString *target, int add_flag, char *source,...)
- void **CSTLString_ltrim** (CSTLString *target)

4.17.1 説明

このファイルは、CSTL ライブラリを用いた文字列型 CSTLString を生成したり、それに関する操作を行うための命令が含まれている。

作者

faithnh

4.17.2 関数

4.17.2.1 int CSTLString_compare_with_char (CSTLString * target1, char * target2)

指定した CSTL 文字列 target1 と比較対象の普通の文字列 target2 と比較する。

引数

<i>target1</i>	指定した CSTL 文字列
<i>target2</i>	比較対象の普通の文字列

戻り値

比較した結果を返す、一致した場合は 0、そうでない場合は 0 以外の値を返す。

4.17.2.2 int CSTLString_delete_tail_str (CSTLString * target1, char * del_str)

指定した CSTL 文字列 target1 から、指定した文字列 del_str の最初に出現する箇所以降を全て削除する。

引数

<i>target1</i>	指定した CSTL 文字列
<i>del_str</i>	削除する場所を指定するための文字列

戻り値

成功したかどうか否かを返す。成功した場合は 1、そうでない場合は 0 を返す。

4.17.2.3 void CSTLString_ltrim (CSTLString * *target*)

指定した文字列の前の半角スペース・改行文字・タブを削除する。

引数

<i>target</i>	指定した文字列
---------------	---------

戻り値

なし

4.17.2.4 void CSTLString_printf (CSTLString * *target*, int *add_flag*, char * *source*, ...)

指定した文字列 *source* を簡易版の書式フォーマット形式で指定した CSTL 文字列 *target* に入力する。簡易版なので、d、f、s、d、%にしか対応していない。

は可能。また、*add_flag* を指定することで、追加・代入の選択もできる。

引数

<i>target</i>	入力先の指定した CSTL 文字列（あらかじめ初期化する必要あり）
<i>add_flag</i>	追加挿入するかどうかのフラグ 1：追加 0：代入
<i>source</i>	指定した文字列（書式フォーマット形式）
...	書式に対応した任意の引数

戻り値

なし

4.17.2.5 void CSTLString_replace_string (CSTLString * *target*, char * *search*, char * *replace*)

指定した CSTL 文字列 *target* に対して、置換対象の文字列 *search* から置換したい文字列 *replace* に置換を行う。

引数

<i>target</i>	指定した CSTL 文字列
<i>search</i>	置換対象の文字列
<i>replace</i>	置換したい文字列

戻り値

なし

4.18 Library/FlagDatabase.h

このファイルは、本体に関するフラグ設定を格納するための命令が含まれている。たとえば、xml として出力するのかなどといった設定が含まれている。

関数

- `int getFlagDatabase (int argc, char *argv[])`
- `int isXmlMode (int flag_database)`
- `int isHelpMode (int flag_database)`
- `int isArrayUnboundCheckMode (int flag_database)`
- `int isUndefineControlCheckMode (int flag_database)`
- `int isZeroDivitionCheckMode (int flag_database)`
- `int isFreeViolationCheckMode (int flag_database)`
- `int isProgramSlicingMode (int flag_database)`

4.18.1 説明

このファイルは、本体に関するフラグ設定を格納するための命令が含まれている。たとえば、xml として出力するのかなどといった設定が含まれている。

作者

faithnh

4.18.2 関数

4.18.2.1 `int getFlagDatabase (int argc, char * argv[])`

プログラムの引数からフラグデータベースを取得する。ここで不正にフラグが設定されていた場合（存在しないフラグがある・フラグが二重に定義されている場合） エラーを返し、強制終了される。

引数

<i>argc</i>	引数フラグの数
<i>argv</i>	引数フラグの文字列

戻り値

引数から取得したフラグを取得する

プログラムの引数からフラグデータベースを取得する。ここで不正にフラグが設定されていた場合（存在しないフラグがある・フラグが二重に定義されている場合） エラーを返し、強制終了される。

引数

<i>argc</i>	引数フラグの数
<i>argv</i>	引数フラグの文字列

戻り値

引数から取得したフラグを取得する。

4.18.2.2 int isArrayUnboundCheckMode (int *flag_database*)

フラグデータベースから、ARRAY_UNBOUND_CHECK_MODE が含まれているかどうか確認する。

引数

<i>flag_database</i>	フラグデータベース
----------------------	-----------

戻り値

ARRAY_UNBOUND_CHECK_MODE が含まれていたなら 1 を返し、そうでなければ 0 を返す。

4.18.2.3 int isFreeViolationCheckMode (int *flag_database*)

フラグデータベースから、FREE_VIOLATION_CHECK_MODE が含まれているかどうか確認する。

引数

<i>flag_database</i>	フラグデータベース
----------------------	-----------

戻り値

FREE_VIOLATION_CHECK_MODE が含まれていたなら 1 を返し、そうでなければ 0 を返す。

4.18.2.4 int isHelpMode (int *flag_database*)

フラグデータベースから、HELP_MODE が含まれているかどうか確認する。

引数

<i>flag_database</i>	フラグデータベース
----------------------	-----------

戻り値

HELP_MODE が含まれていたら 1 を返し、そうでなければ 0 を返す。

4.18.2.5 int isProgramSlicingMode (int *flag_database*)

フラグデータベースから、PROGRAM_SLICING_MODE が含まれているかどうか確認する。

引数

<i>flag_database</i>	フラグデータベース
----------------------	-----------

戻り値

PROGRAM_SLICING_MODE が含まれていたら 1 を返し、そうでなければ 0 を返す。

4.18.2.6 int isUndefineControlCheckMode (int *flag_database*)

フラグデータベースから、UNDEFINE_CONTROL_CHECK_MODE が含まれているかどうか確認する。

引数

<i>flag_database</i>	フラグデータベース
----------------------	-----------

戻り値

UNDEFINE_CONTROL_CHECK_MODE が含まれていたら 1 を返し、そうでなければ 0 を返す。

4.18.2.7 int isXmlMode (int *flag_database*)

フラグデータベースから、XML_MODE が含まれているかどうか確認する。

引数

<i>flag_database</i>	フラグデータベース
----------------------	-----------

戻り値

XML_MODE が含まれていたら 1 を返し、そうでなければ 0 を返す。

4.18.2.8 int isZeroDivitionCheckMode (int *flag_database*)

フラグデータベースから、ZERO_DIVITION_CHECK_MODE が含まれているかどうか確認する。

引数

<i>flag_database</i>	フラグデータベース
----------------------	-----------

戻り値

ZERO_DIVITION_CHECK_MODE が含まれていたら 1 を返し、そうでなければ 0 を返す。

4.19 Library/IdList.h

このファイルは、この変数は C 言語プログラム上のどのブロックに宣言しているかどうかを確認するための情報を生成するための命令が含まれている。

```
#include "Stack_int.h"
#include <cstl/list.h>
```

関数

- int **IDLIST_compare_with** (IDLIST *target1, IDLIST *target2)
- void **SET_STACK_INTToIDLIST** (IDLIST *dest, STACK_INT *source, int num)
- void **printIDLIST** (IDLIST *idlist, int new_line_flag)

4.19.1 説明

このファイルは、この変数は C 言語プログラム上のどのブロックに宣言しているかどうかを確認するための情報を生成するための命令が含まれている。

作者

faithnh

4.19.2 関数

4.19.2.1 int IDLIST_compare_with (IDLIST * *target1*, IDLIST * *target2*)

二つの IDLIST target1,target2 を比較し、次の状態であれば、1 を返す。そうでなければ 0 を返す。 target2 の一番最初のリストの値が 0 である。 target2 のリスト内の値が全て、target1 のリスト内の値と一致する場合

引数

<i>target1</i>	比較対象の IDLIST 1
<i>target2</i>	比較対象の IDLIST 2

戻り値

比較して上記道理になると 1 を返し、そうでなければ、0 を返す。

4.19.2.2 void printIDLIST (IDLIST * *idlist*, int *new_line_flag*)

IDLIST の内容を出力する。

引数

<i>idlist</i>	出力対象の IDLIST
<i>new_line_flag</i>	改行するかどうかのフラグ 1 : 改行する 0 : 改行しない

戻り値

なし

4.19.2.3 void SET_STACK_INTtoIDLIST (IDLIST * *dest*, STACK_INT * *source*, int *num*)

整数スタックの内容 *source* を入れたい対象の IDLIST*dest* へ入れる。

引数

<i>dest</i>	対象の IDLIST
<i>source</i>	入れる整数スタック内容
<i>num</i>	整数スタックに入れる数

戻り値

なし

4.20 Library/Stack_int.h

このファイルは整数スタックに関する操作をするための命令が含まれている。このファイルで使用される主な用途として、C 言語ソースファイルにおけるブロックレベルを設定するのに使用される。

```
#include <cstl/vector.h>
```

関数

- int **STACK_INT_at_and_alloc** (STACK_INT *stack_int, int index)
- int **STACK_INT_inclement_at** (STACK_INT *stack_int, int index)

4.20.1 説明

このファイルは整数スタックに関する操作をするための命令が含まれている。このファイルで使用される主な用途として、C 言語ソースファイルにおけるブロックレベルを設定するのに使用される。

作者

faithnh

4.20.2 関数

4.20.2.1 int STACK_INT_at_and_alloc (STACK_INT * *stack_int*, int *index*)

対象の整数スタック *stack_int* から、指定したインデックスの内容を出力する。インデックスが範囲外である場合は、そのサイズ分まで確保し、0 を返す。

引数

<i>stack_int</i>	対象の整数スタック
<i>index</i>	対象の座標

戻り値

指定したインデックスでの整数スタックの値を出力する。インデックスが範囲外である場合は 0 を返す。

4.20.2.2 int STACK_INT_inclement_at (STACK_INT * *stack_int*, int *index*)

対象の整数スタック *stack_int* から、指定したインデックスの内容をインクリメントする。

引数

<i>stack_int</i>	対象の整数スタック
<i>index</i>	指定したインデックス

戻り値

インクリメントに成功したかどうかを示す。成功した場合は 1、そうでない場合は 0 を返す。

4.21 Library/StoreInformation.h

これはファイル名などといったプログラム内で共通する部分を確保するための命令が含まれている。主にファイル名などを確保する。

関数

- void **setFileName** (char *str)
- char * **getFileName** ()

4.21.1 説明

これはファイル名などといったプログラム内で共通する部分を確保するための命令が含まれている。主にファイル名などを確保する。

作者

faithnh

4.21.2 関数

4.21.2.1 char* getFileName ()

設定したファイル名を呼び出す。

戻り値

呼び出されたファイル名を返す

4.21.2.2 void setFileName (char * str)

ファイル名を設定する。4096 バイト分までの文字が確保でき、それ以上は切り捨てられる。

引数

<i>str</i>	ファイル名
------------	-------

戻り値

なし

4.22 Main/Help.h

このファイルはヘルプ出力関係の命令が含まれている。

関数

- void **viewHelp** (void)

4.22.1 説明

このファイルはヘルプ出力関係の命令が含まれている。

作者

faithnh

4.22.2 関数

4.22.2.1 void viewHelp (void)

ANSIC Varidate Statement Adder tool ヘルプを出力させる。

戻り値

なし

4.23 ProgramSlicing/DeclarationPSI.h

このファイルは Declaration Statement Program Slicing Information の略である。宣言式から、プログラムスライシングに関する情報を抽出するための命令が含まれている。

```
#include "ProgramSlicingInformation.h"
#include "../ANSICInformation/FunctionInformation.h"
#include "../ANSICInformation/Synbol.h"
```

関数

- void **getDeclarationtPSI** (EXPR_SLICING_LIST *expr_slicing_list, AST *declaration, EXPR_SLICING *parent, VARIABLE_TABLE_LIST *vtlist, FUNCTION_INFORMATION_LIST *function_information_list, ASTPOINTER_LIST *ignore_ast_list, AST *target_statement)

4.23.1 説明

このファイルは Declaration Statement Program Slicing Information の略である。宣言式から、プログラムスライシングに関する情報を抽出するための命令が含まれている。

作者

faithnh

4.23.2 関数

4.23.2.1 void getDeclarationtPSI (EXPR_SLICING_LIST * *expr_slicing_list*, AST * *declaration*, EXPR_SLICING * *parent*, VARIABLE_TABLE_LIST * *vtlist*, FUNCTION_INFORMATION_LIST * *function_information_list*, ASTPOINTER_LIST * *ignore_ast_list*, AST * *target_statement*)

宣言式の AST ノード *declaration* から、関数に対するプログラムスライシングを抽出し、プログラムスライシングリスト *expr_slicing_list* に追加する。

引数

<i>expr_slicing_list</i>	追加先のプログラムスライシングリスト <i>expr_slicing_list</i>
<i>declaration</i>	式全般に関する AST ノード
<i>parent</i>	ノードを追加するときの親ノード
<i>vtlist</i>	変数テーブルリスト
<i>function_information_list</i>	関数に関する情報リスト
<i>ignore_ast_list</i>	重複防止のために無視するノードリスト
<i>target_statement</i>	<i>expression_statement</i> を指し示すノード

戻り値

なし

4.24 ProgramSlicing/ExpressionStatementPSI.h

このファイルは Expression Statement Program Slicing Information の略である。式全般から、プログラムスライシングに関する情報を抽出するための命令が含まれている。

```
#include "ProgramSlicingInformation.h"
#include "../ANSICInformation/FunctionInformation.h"
#include "../ANSICInformation/Synbol.h"
```

関数

- void **getExpressionStatementPSI** (EXPR_SLICING_LIST **expr_slicing_list*, AST **expression_statement*, EXPR_SLICING **parent*, VARIABLE_TABLE_

- LIST *vtlist, FUNCTION_INFORMATION_LIST *function_information_list, ASTPOINTER_LIST *ignore_ast_list, **AST** *target_statement)
- void **getInputFunctionPSI** (**AST** *root, ARRAY_OFFSET_LIST *target_variable, ARRAY_OFFSET_LIST *dependences, FUNCTION_INFORMATION_LIST *function_information_list, VARIABLE_TABLE_LIST *vtlist, ASTPOINTER_LIST *ignore_ast_list, **AST** *target_statement)
 - void **getASI_ARRAY_OFFSET_LIST** (**AST** *root, ARRAY_OFFSET_LIST *left_array_offset_list, ARRAY_OFFSET_LIST *right_array_offset_list, FUNCTION_INFORMATION_LIST *function_information_list, VARIABLE_TABLE_LIST *vtlist, ASTPOINTER_LIST *ignore_ast_list, **AST** *target_statement, int allow_side_effect)
 - void **setARGUMENT_NUMBER** (ARRAY_OFFSET_LIST *argument_list, int arg_num, **AST** *call_function)

4.24.1 説明

このファイルは Expression Statement Program Slicing Information の略である。式全般から、プログラムスライシングに関する情報を抽出するための命令が含まれている。

作者

faithnh

4.24.2 関数

- 4.24.2.1 void **getASI_ARRAY_OFFSET_LIST** (**AST** * root, ARRAY_OFFSET_LIST * target_variable, ARRAY_OFFSET_LIST * dependences, FUNCTION_INFORMATION_LIST * function_information_list, VARIABLE_TABLE_LIST * vtlist, ASTPOINTER_LIST * ignore_ast_list, **AST** * target_statement, int allow_side_effect)

expression_statement および、expression に相当するノードから、対象の変数および依存関係の配列オフセットリストを取得する。

引数

<i>root</i>	オフセットリストに該当する AST ノード
<i>target_variable</i>	左辺式のポインタおよび配列のオフセット情報のリスト
<i>dependences</i>	右辺式のポインタおよび配列のオフセット情報のリスト
<i>function_information_list</i>	関数に関する情報リスト
<i>vtlist</i>	検証対象の式をマークするための変数リスト
<i>ignore_ast_list</i>	ポインタでの位置が検証済みである、IDENTIFIER を無視するための AST のアドレスリスト
<i>target_statement</i>	検証式の対象となるステートメント

<i>allow_side_effect</i>	副作用式を許すかどうかのフラグ 1 : 許す 0 : 許さない
--------------------------	---------------------------------

戻り値

なし

4.24.2.2 void getExpressionStatementPSI (EXPR_SLICING_LIST * *expr_slicing_list*, AST * *expression_statement*, EXPR_SLICING * *parent*, VARIABLE_TABLE_LIST * *vtlist*, FUNCTION_INFORMATION_LIST * *function_information_list*, ASTPOINTER_LIST * *ignore_ast_list*, AST * *target_statement*)

式全般の AST ノード *expression_statement* から、関数に対するプログラムスライシングを抽出し、プログラムスライシングリスト *expr_slicing_list* に追加する。

引数

<i>expr_slicing_list</i>	追加先のプログラムスライシングリスト <i>expr_slicing_list</i>
<i>expression_statement</i>	式全般に関する AST ノード
<i>parent</i>	ノードを追加するときの親ノード
<i>vtlist</i>	変数テーブルリスト
<i>function_information_list</i>	関数に関する情報リスト
<i>ignore_ast_list</i>	重複防止のために無視するノードリスト
<i>target_statement</i>	<i>expression_statement</i> を指し示すノード

戻り値

なし

4.24.2.3 void getInputFunctionPSI (AST * *root*, ARRAY_OFFSET_LIST * *target_variable*, ARRAY_OFFSET_LIST * *dependences*, FUNCTION_INFORMATION_LIST * *function_information_list*, VARIABLE_TABLE_LIST * *vtlist*, ASTPOINTER_LIST * *ignore_ast_list*, AST * *target_statement*)

call_function の AST ノードから入力関係の関数を探し出し、これらの引数の変数をすべて、対象変数として登録する。

引数

<i>root</i>	対象の AST ノード
-------------	-------------

<i>target_</i> <i>variable</i>	対象の変数の配列オフセット情報リスト
<i>dependences</i>	依存関係の配列オフセット情報リスト
<i>function_</i> <i>information_</i> <i>list</i>	関数に関する情報リスト
<i>vtlist</i>	検証式の式をマークするための変数リスト
<i>ignore_ast_</i> <i>list</i>	配列オフセットリストの重複登録を防止するための、IDENTIFIER を無視するための AST のアドレスリスト
<i>target_</i> <i>statement</i>	検証式の対象となるステートメント

戻り値

なし

call_function の AST ノードから入出力関係の関数を探し出し、これらの引数の変数をすべて、対象変数として登録する。

引数

<i>root</i>	対象の AST ノード
<i>target_</i> <i>variable</i>	対象の変数の配列オフセット情報リスト
<i>dependences</i>	依存関係の配列オフセット情報リスト
<i>function_</i> <i>information_</i> <i>list</i>	関数に関する情報リスト
<i>vtlist</i>	検証式の式をマークするための変数リスト
<i>ignore_ast_</i> <i>list</i>	配列オフセットリストの重複登録を防止するための、IDENTIFIER を無視するための AST のアドレスリスト
<i>target_</i> <i>statement</i>	検証式の対象となるステートメント

戻り値

なし

4.24.2.4 void setARGUMENT_NUMBER (ARRAY_OFFSET_LIST * *argument_list*, int *arg_num*, AST * *call_function*)

参照先の AST ノードが指定した関数呼び出し call_function である配列オフセットリスト *argument_list* に対して、指定した引数の番号 *argument_number* を以下の形で付加する。

名前 名前::引数番号

引数

<i>argument_-list</i>	付加対象の配列オフセットリスト
<i>arg_num</i>	引数
<i>call_-function</i>	関数呼び出しに対する AST ノード

戻り値

なし

4.25 ProgramSlicing/ForStatementPSI.h

このファイルは For Statement Program Slicing Information の略である。for 文から、プログラムスライシングに関する情報を抽出するための命令が含まれている。

```
#include "ProgramSlicingInformation.h"
#include "../ANSICInformation/FunctionInformation.h"
#include "../ANSICInformation/Symbol.h"
```

関数

- `EXPR_SLICING * getForStatementPSI (EXPR_SLICING_LIST *expr_slicing_list, AST *for_statement, EXPR_SLICING *parent, VARIABLE_TABLE_LIST *vtlist, FUNCTION_INFORMATION_LIST *function_information_list, ASTPOINTER_LIST *ignore_ast_list)`

4.25.1 説明

このファイルは For Statement Program Slicing Information の略である。for 文から、プログラムスライシングに関する情報を抽出するための命令が含まれている。

作者

faithnh

4.25.2 関数

4.25.2.1 `EXPR_SLICING* getForStatementPSI (EXPR_SLICING_LIST * expr_slicing_list,
AST * for_statement, EXPR_SLICING * parent, VARIABLE_TABLE_LIST * vtlist,
FUNCTION_INFORMATION_LIST * function_information_list, ASTPOINTER_LIST *
ignore_ast_list)`

for 文の AST ノード *for_statement* から、関数に対するプログラムスライシングを抽出し、プログラムスライシングリスト *expr_slicing_list* に追加する。

引数

<i>expr_slicing_list</i>	追加先のプログラムスライシングリスト <i>expr_slicing_list</i>
<i>for_statement</i>	for 文に関する AST ノード
<i>parent</i>	ノードを追加するときの親ノード
<i>vtlist</i>	変数テーブルリスト
<i>function_information_list</i>	関数に関する情報リスト
<i>ignore_ast_list</i>	重複防止のために無視するノードリスト

戻り値

なし

4.26 ProgramSlicing/FunctionPSI.h

このファイルは Function Program Slicing Information の略である。関数定義から、プログラムスライシングに関する情報を抽出するための命令が含まれている。

```
#include "ProgramSlicingInformation.h"
```

関数

- `EXPR_SLICING * getFunctionPSI (EXPR_SLICING_LIST *expr_slicing_list, AST *function_definition, EXPR_SLICING *parent)`
- `void getParameterPSI (EXPR_SLICING *expr_slicing, AST *parameter_node, AST *basis_parameter_node)`

4.26.1 説明

このファイルは Function Program Slicing Information の略である。関数定義から、プログラムスライシングに関する情報を抽出するための命令が含まれている。

作者

faithnh

4.26.2 関数

4.26.2.1 `EXPR_SLICING* getFunctionPSI (EXPR_SLICING_LIST * expr_slicing_list, AST * function_definition, EXPR_SLICING * parent)`

関数定義の AST ノード *function_definition* から、関数に対するプログラムスライシングを抽出し、プログラムスライシングリスト *expr_slicing_list* に追加する。

引数

<i>expr_slicing_list</i>	追加先のプログラムスライシングリスト <i>expr_slicing_list</i>
<i>function_definition</i>	関数定義に関する AST ノード
<i>parent</i>	ノードを追加するときの親ノード

戻り値

追加した関数のスライシングへのアドレスを返す

4.26.2.2 `void getParameterPSI (EXPR_SLICING * expr_slicing, AST * parameter_node, AST * basis_parameter_node)`

パラメータリストに関する AST ノード *parameter_node* から関数定義のプログラムスライシングリスト *expr_slicing_list* に格納していく

引数

<i>expr_slicing</i>	関数定義のプログラムスライシング
<i>parameter_node</i>	各パラメータの定義リスト
<i>basis_parameter_node</i>	パラメータを指す AST ノード

戻り値

なし

4.27 ProgramSlicing/IfStatementPSI.h

このファイルは If Statement Program Slicing Information の略である。if 文もしくは、if と else 文から、プログラムスライシングに関する情報を抽出するための命令が含まれている。

```
#include "ProgramSlicingInformation.h"
#include "../ANSICInformation/FunctionInformation.h"
#include "../ANSICInformation/Synbol.h"
```

関数

- `EXPR_SLICING * getIfStatementPSI (EXPR_SLICING_LIST *expr_slicing_list, AST *if_statement, EXPR_SLICING *parent, VARIABLE_TABLE_LIST *vtlist, FUNCTION_INFORMATION_LIST *function_information_list, ASTPOINTER_LIST *ignore_ast_list)`

4.27.1 説明

このファイルは If Statement Program Slicing Information の略である。if 文もしくは、if と else 文から、プログラムスライシングに関する情報を抽出するための命令が含まれている。

作者

faithnh

4.27.2 関数

4.27.2.1 `EXPR_SLICING* getIfStatementPSI (EXPR_SLICING_LIST * expr_slicing_list,
AST * if_statement, EXPR_SLICING * parent, VARIABLE_TABLE_LIST * vtlist,
FUNCTION_INFORMATION_LIST * function_information_list, ASTPOINTER_LIST *
ignore_ast_list)`

if 文もしくは、if と else 文の AST ノード `if_statement` から、関数に対するプログラムスライシングを抽出し、プログラムスライシングリスト `expr_slicing_list` に追加する。

引数

<code>expr_slicing_list</code>	追加先のプログラムスライシングリスト <code>expr_slicing_list</code>
<code>if_statement</code>	if 文もしくは、if と else 文に関する AST ノード
<code>parent</code>	ノードを追加するときの親ノード
<code>vtlist</code>	変数テーブルリスト
<code>function_information_list</code>	関数に関する情報リスト
<code>ignore_ast_list</code>	重複防止のために無視するノードリスト

戻り値

なし

if 文もしくは、if と else 文の AST ノード `if_statement` から、関数に対するプログラムスライシングを抽出し、プログラムスライシングリスト `expr_slicing_list` に追加する。

引数

<code>expr_slicing_list</code>	追加先のプログラムスライシングリスト
<code>if_statement</code>	if 文もしくは、if と else 文に関する AST ノード
<code>parent</code>	ノードを追加するときの親ノード
<code>vtlist</code>	変数テーブルリスト

<i>function_information_list</i>	関数に関する情報リスト
<i>ignore_ast_list</i>	重複防止のために無視するノードリスト

戻り値

なし

4.28 ProgramSlicing/JumpStatementPSI.h

このファイルは Jump Statement Program Slicing Information の略である。GOTO、continue、break 文から、プログラムスライシングに関する情報を抽出するための命令が含まれている。

```
#include "ProgramSlicingInformation.h"
#include "../ANSICInformation/FunctionInformation.h"
#include "../ANSICInformation/Symbol.h"
```

関数

- void **getJumpStatementPSI** (EXPR_SLICING_LIST *expr_slicing_list, AST *jump_statement, EXPR_SLICING *parent)

4.28.1 説明

このファイルは Jump Statement Program Slicing Information の略である。GOTO、continue、break 文から、プログラムスライシングに関する情報を抽出するための命令が含まれている。

作者

faithnh

4.28.2 関数

4.28.2.1 void getJumpStatementPSI (EXPR_SLICING_LIST * *expr_slicing_list*, AST * *jump_statement*, EXPR_SLICING * *parent*)

GOTO、continue、break 文の AST ノード expression_statement から、関数に対するプログラムスライシングを抽出し、プログラムスライシングリスト expr_slicing_list に追加する。

引数

<i>expr_slicing_list</i>	追加先のプログラムスライシングリスト <i>expr_slicing_list</i>
<i>jump_statement</i>	GOTO、continue、break 文に関する AST ノード
<i>parent</i>	ノードを追加するときの親ノード

戻り値

なし

4.29 ProgramSlicing/LabeledStatementPSI.h

このファイルは Labeled Statement Program Slicing Information の略である。goto ラベル文、case ラベル文、default ラベル文から、プログラムスライシングに関する情報を抽出するための命令が含まれている。

```
#include "ProgramSlicingInformation.h"
#include "../ANSICInformation/FunctionInformation.h"
#include "../ANSICInformation/Synbol.h"
```

関数

- `EXPR_SLICING * getLabeledStatementPSI (EXPR_SLICING_LIST *expr_slicing_list, AST *labeled_statement, EXPR_SLICING *parent, VARIABLE_TABLE_LIST *vtlist, FUNCTION_INFORMATION_LIST *function_information_list, ASTPOINTER_LIST *ignore_ast_list)`

4.29.1 説明

このファイルは Labeled Statement Program Slicing Information の略である。goto ラベル文、case ラベル文、default ラベル文から、プログラムスライシングに関する情報を抽出するための命令が含まれている。

作者

faithnh

4.29.2 関数

4.29.2.1 **EXPR_SLICING*** `getLabeledStatementPSI (EXPR_SLICING_LIST * expr_slicing_list,
AST * labeled_statement, EXPR_SLICING * parent, VARIABLE_TABLE_LIST * vtlist,
FUNCTION_INFORMATION_LIST * function_information_list, ASTPOINTER_LIST *
ignore_ast_list)`

goto ラベル文、case ラベル文、default ラベル文の AST ノード `expression_statement` から、関数に対するプログラムスライシングを抽出し、プログラムスライシングリスト `expr_slicing_list` に追加する。

引数

<i>expr_slicing_list</i>	追加先のプログラムスライシングリスト <code>expr_slicing_list</code>
<i>labeled_statement</i>	goto ラベル文、case ラベル文、default ラベル文に関する AST ノード
<i>parent</i>	ノードを追加するときの親ノード
<i>vtlist</i>	変数テーブルリスト
<i>function_information_list</i>	関数に関する情報リスト
<i>ignore_ast_list</i>	重複防止のために無視するノードリスト

戻り値

なし

4.30 ProgramSlicing/ProgramSlicing.h

このファイルはプログラムスライシングを行うための命令が含まれている。指定した識別子名および行数を入れることで、それに基づいてプログラムスライシングを行う。

```
#include "../ANSICInformation/AST.h"
#include "../ANSICInformation/PointerArrayControl.h"
#include "../ANSICInformation/FunctionInformation.h"
#include "../ANSICInformation/Synbol.h"
#include "ProgramSlicingInformation.h"
#include <cstl/list.h>
```

関数

- `void * createStatementNodeList (AST *root, EXPR_SLICING_LIST *expr_slicing_list, EXPR_SLICING *parent, VARIABLE_TABLE_LIST *vtlist, FUNCTION_`

- INFORMATION_LIST *function_information_list)
- void **createDD_list_in_global** (EXPR_SLICING *expr_slicing, EXPR_SLICING_LIST *program_expr_slicing_list)
- void **createDD_list** (EXPR_SLICING *expr_slicing)
- void **createDD_listAll** (EXPR_SLICING_LIST *expr_slicing_list, FUNCTION_INFORMATION_LIST *function_information_list, EXPR_SLICING_LIST *program_expr_slicing_list)
- void **createDD_list_in_Function** (EXPR_SLICING *expr_slicing, EXPR_SLICING_LIST *expr_slicing_list, FUNCTION_INFORMATION_LIST *function_information_list)
- void **createDD_list_in_argument** (EXPR_SLICING *expr_slicing, CSTLString *argument_name, EXPR_SLICING *function_definition_expr_slicing)
- void **staticSlicing** (EXPR_SLICING *expr_slicing)
- int **startStaticSlicing** (AST *inst_ast, EXPR_SLICING_LIST *expr_slicing_list)

4.30.1 説明

このファイルはプログラムスライシングを行うための命令が含まれている。指定した識別子名および行数を入れることで、それに基づいてプログラムスライシングを行う。

作者

faithnh

4.30.2 関数

4.30.2.1 void createDD_list (EXPR_SLICING * expr_slicing)

指定した対象の命令に対して、データ依存関係の命令を探し、その命令を結びつける。

引数

<i>expr_slicing</i>	指定した対象の命令
---------------------	-----------

戻り値

なし

4.30.2.2 void createDD_list_in_argument (EXPR_SLICING * expr_slicing, CSTLString * argument_name, EXPR_SLICING * function_definition_expr_slicing)

関数へのスライシング情報に対して、仮引数名が依存関係として用いている命令を探し、それを指定したプログラムスライシングのデータ依存関係として追加する。

引数

<i>expr_slicing</i>	関数呼び出しのプログラムスライシング
<i>argument_name</i>	仮引数名
<i>function_definition_expr_slicing</i>	関数宣言へのプログラムスライシング

戻り値

なし

4.30.2.3 void createDD_list_in_Function (EXPR_SLICING * *expr_slicing*, EXPR_SLICING_LIST * *expr_slicing_list*, FUNCTION_INFORMATION_LIST * *function_information_list*)

関数呼び出しから、呼び出している関数へのデータ依存関係を生成する。

引数

<i>expr_slicing</i>	関数呼び出しが含まれている処理名
<i>expr_slicing_list</i>	プログラム全体のプログラムスライシングリスト
<i>function_information_list</i>	関数に関する情報リスト

戻り値

なし

4.30.2.4 void createDD_list_in_global (EXPR_SLICING * *expr_slicing*, EXPR_SLICING_LIST * *program_expr_slicing_list*)

指定した対象の命令とグローバル変数に関してデータ依存関係があるかどうか調べる。存在すれば、グローバル変数宣言に対して、データ依存関係として結びつける。

引数

<i>expr_slicing</i>	指定した対象の命令
<i>program_expr_slicing_list</i>	プログラム全体のスライシングリスト

戻り値

なし

指定した対象の命令とグローバル変数宣言に関してデータ依存関係があるかどうか調べる。存在すれば、グローバル変数宣言に対して、データ依存関係として結びつける

引数

<i>expr_slicing</i>	指定した対象の命令
<i>program_expr_slicing_list</i>	プログラム全体のスライシングリスト

戻り値

なし

4.30.2.5 void createDD_listAll (EXPR_SLICING_LIST * *expr_slicing_list*,
FUNCTION_INFORMATION_LIST * *function_information_list*, EXPR_SLICING_LIST *
program_expr_slicing_list)

プログラムスライシングリスト中のすべての命令に対し、データ依存関係の命令を探し、その命令を結びつける。

引数

<i>expr_slicing_list</i>	指定した対象のプログラムスライシングリスト
<i>function_information_list</i>	関数に関する情報リスト
<i>program_expr_slicing_list</i>	プログラム全体へのスライシングリスト

戻り値

なし

4.30.2.6 void* createStatementNodeList (AST * *root*, EXPR_SLICING_LIST *
expr_slicing_list, EXPR_SLICING * *parent*, VARIABLE_TABLE_LIST * *vtlist*,
FUNCTION_INFORMATION_LIST * *function_information_list*)

指定したプログラム A S T ノードから次のようなノードを取得し、そこからスライシングを行うための構造体リスト *expr_slicing_list* を生成する。 *function_definition_type_a*, *function_definition_type_b*, *expression_statement*, *declaration_with_init*, *if_statement*, *ifelse_statement*, *switch_statement*, *while_statement*, *dowhile_statement*, *for_statement_type_a*, *for_statement_type_b*, *for_statement_type_c*, *for_statement_type_d*, *goto_statement*, *continue_statement*, *break_statement*, *return_statement*, *return_expr_statement* *goto_labeled_statement*, *case_labeled_statement*,

default_labeled_statement

引数

<i>root</i>	指定したプログラム A S T ノード
<i>expr_slicing_list</i>	スライシングを行うための構造体
<i>parent</i>	親のスライシングデータ
<i>vtlist</i>	変数テーブルリスト
<i>function_information_list</i>	関数に関する情報のリスト

戻り値

なし

4.30.2.7 int startStaticSlicing (AST * *inst_ast*, EXPR_SLICING_LIST * *expr_slicing_list*)

指定した命令への AST ノードに基づいて *expr_slicing_list* に対してスタティックスライシング処理を行う。

引数

<i>inst_ast</i>	指定した命令への AST ノード
<i>expr_slicing_list</i>	スタティックスライシング処理を行うための構造体リスト

戻り値

成功したかどうかを返却する。 1 : 成功 0 : 失敗

4.30.2.8 void staticSlicing (EXPR_SLICING * *expr_slicing*)

指定した識別子の命令に対してスタティックスライシング処理を行う。

引数

<i>expr_slicing</i>	指定した命令に対するプログラムスライシング情報
---------------------	-------------------------

戻り値

なし

4.31 ProgramSlicing/ProgramSlicingInformation.h

このファイルはプログラムスライシングに関する情報を取り扱う命令が含まれている。

```
#include "../ANSICInformation/AST.h"
#include "../ANSICInformation/PointerArrayControl.h"
#include <cstl/list.h>
```

関数

- **EXPR_SLICING * new_EXPR_SLICING** (int expr_slicing_number, **AST** *expression, ARRAY_OFFSET_LIST *target_variable, ARRAY_OFFSET_LIST *dependences, EXPR_SLICING_LIST *children1, EXPR_SLICING_LIST *children2, EXPR_SLICING *parent)
- **DD_INFORMATION * new_DD_INFORMATION** (CSTLString *dd_variable_name, EXPR_SLICING *dd_target)
- int **searchDD** (CSTLString *variable_name, DD_INFORMATION_LIST *dd_information_list, DD_INFORMATION **find_dd_informaiton)
- int **searchDeclarationDD** (CSTLString *variable_name, EXPR_SLICING *expr_slicing)
- void **initExprSlicingListFlag** (EXPR_SLICING_LIST *expr_slicing_list)
- **EXPR_SLICING * searchFunctionPSI** (CSTLString *function_name, EXPR_SLICING_LIST *expr_slicing_list)
- void **getFunctionGrobalVariable** (ARRAY_OFFSET_LIST *output_global_variable_list, EXPR_SLICING_LIST *function_expr_slicing_list, ARRAY_OFFSET_LIST *global_variable_list)
- void **getVariableDeclarationFromEXPR_SLICING_LIST** (ARRAY_OFFSET_LIST *global_variable_list, EXPR_SLICING_LIST *program_expr_slicing_list)
- void **setGlobalVariable** (EXPR_SLICING_LIST *expr_slicing_list, EXPR_SLICING_LIST *program_expr_slicing_list)
- void **print_EXPR_SLICING_LIST** (EXPR_SLICING_LIST *expr_slicing_list)
- void **print_tree_EXPR_SLICING_LIST** (EXPR_SLICING_LIST *expr_slicing_list, int program_slicing_mode)
- void **registerIncDecVariable** (ARRAY_OFFSET_LIST *dependences, ARRAY_OFFSET_LIST *target_variable)

4.31.1 説明

このファイルはプログラムスライシングに関する情報を取り扱う命令が含まれている。

作者

faithnh

4.31.2 関数

4.31.2.1 void `getFunctionGlobalVariable` (`ARRAY_OFFSET_LIST` * *output_global_variable_list*,
`EXPR_SLICING_LIST` * *function_expr_slicing_list*, `ARRAY_OFFSET_LIST` *
global_variable_list)

関数内で扱うグローバル変数を抽出し、配列オフセットリストとして抽出する。

引数

<i>output_global_variable_list</i>	抽出先の配列オフセットリスト
<i>function_expr_slicing_list</i>	関数呼び出しに対するプログラムスライシングリスト
<i>global_variable_list</i>	グローバル変数一覧を示す配列オフセットリスト

戻り値

なし

4.31.2.2 void `getVariableDeclarationFromExprSlicingList` (`ARRAY_OFFSET_LIST` *
global_variable_list, `EXPR_SLICING_LIST` * *program_expr_slicing_list*)

対象のプログラムスライシングリストから、変数宣言部分を抽出し、配列オフセット情報として抽出する。

引数

<i>global_variable_list</i>	グローバル変数一覧を示す配列オフセットリスト
<i>program_expr_slicing_list</i>	プログラム全体のプログラムスライシングリスト

戻り値

なし

4.31.2.3 void `initExprSlicingListFlag` (`EXPR_SLICING_LIST` * *expr_slicing_list*)

プログラムスライシングの flag を初期化する。

引数

<i>expr_slicing_list</i>	対象のプログラムスライシング情報のリスト
--------------------------	----------------------

戻り値

なし

4.31.2.4 DD_INFORMATION* **new_DD_INFORMATION** (**CSTLString** * *dd_variable_name*,
EXPR_SLICING * *dd_target*)

データ依存関係に関する情報を生成する。

引数

<i>dd_variable_name</i>	データ依存関係に関する変数名
* <i>dd_target</i>	データ依存関係のターゲットとなる命令

戻り値

生成されたデータ依存関係に関する情報の構造体へのアドレスを返す。

データ依存関係に関する情報を生成する。

引数

<i>dd_variable_name</i>	データ依存関係に関する変数名
* <i>dd_target</i>	データ依存関係のターゲットとなる命令

戻り値

生成されたデータ依存関係に関する情報の構造体へのアドレスを返す

4.31.2.5 EXPR_SLICING* **new_EXPR_SLICING** (**int** *expr_slicing_number*, **AST** * *expression*,
ARRAY_OFFSET_LIST * *target_variable*, **ARRAY_OFFSET_LIST** * *dependences*,
EXPR_SLICING_LIST * *children1*, **EXPR_SLICING_LIST** * *children2*, **EXPR_SLICING** *
parent)

スライシングを行うための構造体を生成する。

引数

<i>expr_slicing_number</i>	対象の式に対する AST 番号
<i>expression</i>	対象の式への A S T ノード
<i>target_variable</i>	変数のオフセットリスト
<i>dependences</i>	依存関係の配列オフセットリスト

<i>children1</i>	if や while、for に対するスライシングリスト
<i>children2</i>	else 文に対するスライシングリスト
<i>parent</i>	親をたどるためのノード

戻り値

生成されたスライシングを行うための構造体へのアドレスを返す。

スライシングを行うための構造体を生成する。

引数

<i>expr_-slicing_-number</i>	対象の式に対する AST 番号
<i>expression</i>	対象の式への A S T ノード
<i>target_-variable</i>	変数のオフセットリスト
<i>dependences</i>	依存関係の配列オフセットリスト
<i>children1</i>	if や while、for に対するスライシングリスト
<i>children2</i>	else 文に対するスライシングリスト
<i>parent</i>	親をたどるためのノード

戻り値

生成されたスライシングを行うための構造体へのアドレスを返す

4.31.2.6 void print_EXPR_SLICING_LIST (EXPR_SLICING_LIST * *expr_slicing_list*)

スライシングに関する情報リスト *expr_slicing_list* を出力させる。

引数

<i>expr_-slicing_list</i>	スライシングに関する情報リスト
---------------------------	-----------------

戻り値

なし

4.31.2.7 void print_tree_EXPR_SLICING_LIST (EXPR_SLICING_LIST * *expr_slicing_list*, int *program_slicing_mode*)

スライシングに関する情報リスト *expr_slicing_list* をツリー形式で出力させる。

引数

<i>expr_-slicing_list</i>	スライシングに関する情報リスト
<i>program_-slicing_mode</i>	プログラムスライシング後のモードにするかどうかのフラグ 1 : 有効 0 : 無効

戻り値

なし

4.31.2.8 void registerIncDecVariable (ARRAY_OFFSET_LIST * *dependences*,
ARRAY_OFFSET_LIST * *target_variable*)

dependences でインクリメントやデクリメントのフラグが立っている変数を見つけたら、それを *target_variable* に登録する。

引数

<i>dependences</i>	インクリメントやデクリメントを探す対象の依存変数
<i>target_variable</i>	登録先の定義対象変数

戻り値

なし

4.31.2.9 int searchDD (CSTLString * *variable_name*, DD_INFORMATION_LIST *
dd_information_list, DD_INFORMATION ** *find_dd_informaiton*)

指定したデータ依存関係のリストから、指定された変数名が存在するか調べる。

引数

<i>variable_name</i>	指定された変数名
<i>dd_information_list</i>	指定したデータ依存関係のリスト
<i>find_dd_informaiton</i>	見つけた場合に返すデータ依存関係の情報

戻り値

存在する場合は 1、そうでない場合は 0 を返す。

指定したデータ依存関係のリストから、指定された変数名が存在するか調べる。

引数

<i>variable_name</i>	指定された変数名
<i>dd_information_list</i>	指定したデータ依存関係のリスト
<i>find_dd_information</i>	見つけた場合に返すデータ依存関係の情報

戻り値

存在する場合は 1、そうでない場合は 0 を返す

4.31.2.10 `int searchDeclarationDD (CSTLString * variable_name, EXPR_SLICING * expr_slicing)`

指定したプログラムスライシング情報のデータ依存関係から、指定された変数名である変数宣言が存在するか調べる。

引数

<i>variable_name</i>	指定された変数名
<i>expr_slicing</i>	指定したプログラムスライシング情報のデータ

戻り値

存在する場合は 1、そうでない場合は 0 を返す。

4.31.2.11 `EXPR_SLICING* searchFunctionPSI (CSTLString * function_name, EXPR_SLICING_LIST * expr_slicing_list)`

プログラム全体のプログラムスライシングリスト *expr_slicing_list* から、関数呼び出しに対応する関数定義のプログラムスライシング情報を取得する。

引数

<i>function_name</i>	関数名
<i>expr_slicing_list</i>	プログラム全体のプログラムスライシングリスト

戻り値

取得に成功した場合、関数呼び出しに対応する関数定義へのプログラムスライシング情報を返す、失敗した場合は NULL を返す

プログラム全体のプログラムスライシングリスト `expr_slicing_list` から、関数呼び出しに対応する関数定義のプログラムスライシング情報を取得する。

引数

<code>function_name</code>	関数名
<code>expr_slicing_list</code>	プログラム全体のプログラムスライシングリスト

戻り値

取得に成功した場合、関数呼び出しに対応する関数定義へのプログラムスライシング情報を返す、失敗した場合は `NULL` を返す。

4.31.2.12 `void setGlobalVariable (EXPR_SLICING_LIST * expr_slicing_list, EXPR_SLICING_LIST * program_expr_slicing_list)`

関数呼び出しから、取り扱うグローバル変数を抽出し、対象の変数一覧に追加していく。

引数

<code>expr_slicing_list</code>	追加対象のプログラムスライシングリスト
<code>program_expr_slicing_list</code>	プログラム全体のプログラムスライシングリスト

関数呼び出しから、取り扱うグローバル変数を抽出し、対象の変数一覧に追加していく。

引数

<code>expr_slicing_list</code>	追加対象のプログラムスライシングリスト
<code>program_expr_slicing_list</code>	プログラム全体のプログラムスライシングリスト

戻り値

なし

4.32 ProgramSlicing/ReturnStatementPSI.h

このファイルは Return Statement Program Slicing Information の略である。 `return` 文から、プログラムスライシングに関する情報を抽出するための命令が含まれて

いる。

```
#include "ProgramSlicingInformation.h"
#include "../ANSICInformation/FunctionInformation.h"
#include "../ANSICInformation/Synbol.h"
```

関数

- void **getReturnStatementPSI** (EXPR_SLICING_LIST *expr_slicing_list, AST *return_statement, EXPR_SLICING *parent, VARIABLE_TABLE_LIST *vtlist, FUNCTION_INFORMATION_LIST *function_information_list, ASTPOINTER_LIST *ignore_ast_list)

4.32.1 説明

このファイルは Return Statement Program Slicing Information の略である。return 文から、プログラムスライシングに関する情報を抽出するための命令が含まれている。

作者

faithnh

4.32.2 関数

4.32.2.1 void getReturnStatementPSI (EXPR_SLICING_LIST * *expr_slicing_list*, AST * *return_statement*, EXPR_SLICING * *parent*, VARIABLE_TABLE_LIST * *vtlist*, FUNCTION_INFORMATION_LIST * *function_information_list*, ASTPOINTER_LIST * *ignore_ast_list*)

return 文の AST ノード *expression_statement* から、関数に対するプログラムスライシングを抽出し、プログラムスライシングリスト *expr_slicing_list* に追加する。

引数

<i>expr_slicing_list</i>	追加先のプログラムスライシングリスト <i>expr_slicing_list</i>
<i>return_statement</i>	GOTO、continue、break 文に関する AST ノード
<i>parent</i>	ノードを追加するときの親ノード
<i>vtlist</i>	変数テーブルリスト
<i>function_information_list</i>	関数に関する情報リスト
<i>ignore_ast_list</i>	重複防止のために無視するノードリスト

戻り値

なし

4.33 ProgramSlicing/SwitchStatementPSI.h

このファイルは Switch Statement Program Slicing Information の略である。 switch 文から、プログラムスライシングに関する情報を抽出するための命令が含まれている。

```
#include "ProgramSlicingInformation.h"
#include "../ANSICInformation/FunctionInformation.h"
#include "../ANSICInformation/Synbol.h"
```

関数

- `EXPR_SLICING * getSwicthStatementPSI (EXPR_SLICING_LIST *expr_slicing_list, AST *switch_statement, EXPR_SLICING *parent, VARIABLE_TABLE_LIST *vtlist, FUNCTION_INFORMATION_LIST *function_information_list, ASTPOINTER_LIST *ignore_ast_list)`

4.33.1 説明

このファイルは Switch Statement Program Slicing Information の略である。 switch 文から、プログラムスライシングに関する情報を抽出するための命令が含まれている。

作者

faithnh

4.33.2 関数

4.33.2.1 `EXPR_SLICING* getSwicthStatementPSI (EXPR_SLICING_LIST * expr_slicing_list, AST * switch_statement, EXPR_SLICING * parent, VARIABLE_TABLE_LIST * vtlist, FUNCTION_INFORMATION_LIST * function_information_list, ASTPOINTER_LIST * ignore_ast_list)`

switch 文の AST ノード `switch_statement` から、関数に対するプログラムスライシングを抽出し、 プログラムスライシングリスト `expr_slicing_list` に追加する。

引数

<i>expr_slicing_list</i>	追加先のプログラムスライシングリスト <code>expr_slicing_list</code>
<i>switch_statement</i>	switch 文に関する AST ノード

<i>parent</i>	ノードを追加するときの親ノード
<i>vtlist</i>	変数テーブルリスト
<i>function_information_list</i>	関数に関する情報リスト
<i>ignore_ast_list</i>	重複防止のために無視するノードリスト

戻り値

なし

4.34 ProgramSlicing/WhileStatementPSI.h

このファイルは While Statement Program Slicing Information の略である。while 文から、プログラムスライシングに関する情報を抽出するための命令が含まれている。

```
#include "ProgramSlicingInformation.h"
#include "../ANSICInformation/FunctionInformation.h"
#include "../ANSICInformation/Symbol.h"
```

関数

- `EXPR_SLICING * getWhileStatementPSI (EXPR_SLICING_LIST *expr_slicing_list, AST *while_statement, EXPR_SLICING *parent, VARIABLE_TABLE_LIST *vtlist, FUNCTION_INFORMATION_LIST *function_information_list, ASTPOINTER_LIST *ignore_ast_list)`

4.34.1 説明

このファイルは While Statement Program Slicing Information の略である。while 文から、プログラムスライシングに関する情報を抽出するための命令が含まれている。

作者

faithnh

4.34.2 関数

4.34.2.1 **EXPR_SLICING*** `getWhileStatementPSI (EXPR_SLICING_LIST * expr_slicing_list,
AST * while_statement, EXPR_SLICING * parent, VARIABLE_TABLE_LIST * vtlist,
FUNCTION_INFORMATION_LIST * function_information_list, ASTPOINTER_LIST *
ignore_ast_list)`

while 文や DoWhile 文の AST ノード `while_statement` から、関数に対するプログラムスライシングを抽出し、プログラムスライシングリスト `expr_slicing_list` に追加する。

引数

<i>expr_slicing_list</i>	追加先のプログラムスライシングリスト <code>expr_slicing_list</code>
<i>while_statement</i>	if 文もしくは、if と else 文に関する AST ノード
<i>parent</i>	ノードを追加するときの親ノード
<i>vtlist</i>	変数テーブルリスト
<i>function_information_list</i>	関数に関する情報リスト
<i>ignore_ast_list</i>	重複防止のために無視するノードリスト

戻り値

なし

Index

abstract_syntax_tree, 7
addIncludeDataFromFile
 PreProcess.h, 69
adjustProgramStart
 PreProcess.h, 69
ANSIC_CODE.h
 ASTLIST_ITERATOR_1, 17
 ASTLIST_ITERATOR_2, 18
 ASTLIST_ITERATOR_3, 18
 ASTLIST_ITERATOR_4, 18
 ASTLIST_ITERATOR_5, 18
 ASTLIST_ITERATOR_6, 19
 ASTLIST_ITERATOR_7, 19
ANSICInformation/ANSIC_CODE.h, 17
ANSICInformation/AST.h, 19
ANSICInformation/DivitionDeclarator.h, 28
ANSICInformation/DivitionInformation.h,
 29
ANSICInformation/ForInformation.h, 32
ANSICInformation/FreeMemInfo.h, 35
ANSICInformation/FunctionInformation.h,
 37
ANSICInformation/MallocNumber.h, 42
ANSICInformation/MemallocInfo.h, 44
ANSICInformation/PointerArrayControl.h,
 48
ANSICInformation/PreProcess.h, 68
ANSICInformation/Return_Info.h, 70
ANSICInformation/SubEffectCheck.h, 71
ANSICInformation/Synbol.h, 73
ANSICInformation/Varidate_statement.h, 85
ARRAY_OFFSET
 PointerArrayControl.h, 50
array_offset, 8
ARRAY_OFFSET_LIST_push_back_ref_-
 not_dup
 PointerArrayControl.h, 51
ArrayOffsetToValidateStatement
 Varidate_statement.h, 88
AST
 AST.h, 21
AST.h
 AST, 21
 deleteAST, 21
 findASTAddress, 21
 fprintDataFromAST, 21
 fprintfStatement, 21
 getArgumentAST, 22
 getArgumentString, 22
 getArgumentStringEnableExcept, 23
 getASTwithString, 23
 getStringFromAST, 23
 getStringFromASTEnableExcept, 24
 getStringReplaceASTtoString, 24
 multi_push_back_childrenAST, 24
 new_AST, 25
 printDataFromAST, 25
 printTargetASTNode, 25
 push_back_childrenAST, 26
 same_new_AST, 26
 setASTBlocklevelAndId, 27
 setASTReturnType, 27
 traverseAST, 27
 traverseASTwithXML, 28
ASTLIST_ITERATOR_1
 ANSIC_CODE.h, 17
ASTLIST_ITERATOR_2
 ANSIC_CODE.h, 18
ASTLIST_ITERATOR_3
 ANSIC_CODE.h, 18
ASTLIST_ITERATOR_4
 ANSIC_CODE.h, 18
ASTLIST_ITERATOR_5
 ANSIC_CODE.h, 18
ASTLIST_ITERATOR_6
 ANSIC_CODE.h, 19
ASTLIST_ITERATOR_7
 ANSIC_CODE.h, 19
CharStringExtend.h
 isExpression, 115
 str_extract, 115

- checkCallFunction
 - PointerArrayControl.h, 51
- checkContainSubEffectStatement
 - SubEffectCheck.h, 72
- checkIdentifierPointerArrayLevel
 - PointerArrayControl.h, 51
- checkIgnoreASTList
 - PointerArrayControl.h, 52
- copyArrayOffsetList
 - PointerArrayControl.h, 52
- createArrayExpression
 - PointerArrayControl.h, 53
- createCheckUnboundAndUndefineOperationCheck
 - Varidate_statement.h, 89
- createDD_list
 - ProgramSlicing.h, 137
- createDD_list_in_argument
 - ProgramSlicing.h, 137
- createDD_list_in_Function
 - ProgramSlicing.h, 138
- createDD_list_in_global
 - ProgramSlicing.h, 138
- createDD_listAll
 - ProgramSlicing.h, 139
- createStatementNodeList
 - ProgramSlicing.h, 139
- createValidateStatementFromIncDecExpr
 - Varidate_statement.h, 90
- createValidateStatement
 - Varidate_statement.h, 91
- createValidateStatementAdderFileEachCheck
 - Varidate_statement.h, 92
- createValidateStatementForFreeAction
 - Varidate_statement.h, 93
- createValidateStatementForMallocAction
 - Varidate_statement.h, 94
- createValidateStatementFromArrayDefine
 - Varidate_statement.h, 95
- createValidateVariableArrayExpression
 - PointerArrayControl.h, 53
- createVaridateStatementFromPointerDefine
 - Varidate_statement.h, 95
- createViolentFreeOperation
 - Varidate_statement.h, 96
- createZeroDivitionCheck
 - Varidate_statement.h, 96
- CSTLString.h
 - CSTLString_compare_with_char, 116
 - CSTLString_delete_tail_str, 116
 - CSTLString_ltrim, 116
 - CSTLString_printf, 117
 - CSTLString_replace_string, 117
 - CSTLString_compare_with_char, 116
 - CSTLString.h, 116
 - CSTLString_delete_tail_str, 116
 - CSTLString.h, 116
 - CSTLString_ltrim, 116
 - CSTLString.h, 116
 - CSTLString_printf, 117
 - CSTLString.h, 117
 - CSTLString_replace_string, 117
 - CSTLString.h, 117
- DeclarationPSI.h
 - getDeclarationtPSI, 126
- deleteAST
 - AST.h, 21
- deleteOFFSET_LIST
 - PointerArrayControl.h, 54
- deleteParameterDefine
 - FunctionInformation.h, 38
- deletePointer
 - PointerArrayControl.h, 54
 - Synbol.h, 75
- deletePointerAndArraySynbol
 - PointerArrayControl.h, 54
 - Synbol.h, 75
- DIVITION_INFORMATION
 - DivitionInformation.h, 30
- divition_information, 9
- DivitionDeclarator.h
 - OutputSourceAfterDivitionDeclarator, 29
- DivitionInformation.h
 - DIVITION_INFORMATION, 30
 - getDIVITION_INFORMATION_LIST, 30
 - new_DIVITION_INFORMATION, 31
 - new_DIVITION_INFORMATION_-char, 31
 - printDIVITION_INFORMATION_LIST, 31
- ExpressionStatementPSI.h
 - getASI_ARRAY_OFFSET_LIST, 127
 - getExpressionStatementPSI, 128
 - getInputFunctionPSI, 128
 - setARGUMENT_NUMBER, 129
- find_STRUCT_TABLE_DATA

- Synbol.h, 75
- findASTAddress
 - AST.h, 21
- FlagDatabase.h
 - getFlagDatabase, 118
 - isArrayUnboundCheckMode, 119
 - isFreeViolationCheckMode, 119
 - isHelpMode, 119
 - isProgramSlicingMode, 120
 - isUndefineControlCheckMode, 120
 - isXmlMode, 120
 - isZeroDivitionCheckMode, 120
- FOR_INFORMATION
 - ForInformation.h, 33
- for_information, 9
- ForInformation.h
 - FOR_INFORMATION, 33
 - getFOR_INFORMATION_LIST, 33
 - new_FOR_INFORMATION, 33
 - print_FOR_INFORMATION_LIST, 34
 - searchFOR_INFORMATION_FromAST, 34
- ForStatementPSI.h
 - getForStatementPSI, 130
- fprintDataFromAST
 - AST.h, 21
- fprintfStatement
 - AST.h, 21
- fprintProgramDataWithPSIVaridateStatement
 - Varidate_statement.h, 97
- fprintProgramDataWithValidateStatement
 - Varidate_statement.h, 98
- fprintValidateStatement
 - Varidate_statement.h, 99
- fprintValidateStatement_not_assert
 - Varidate_statement.h, 100
- freemem_info, 10
- FREEMEMINFO
 - FreeMemInfo.h, 36
- FreeMemInfo.h
 - FREEMEMINFO, 36
 - getFreememInfo, 36
 - new_FREEMEMINFO, 36
 - printFREEMEMINFO, 36
- FUNCTION_INFORMATION
 - FunctionInformation.h, 38
- function_information, 10
- FunctionInformation.h
 - deleteParameterDefine, 38
 - FUNCTION_INFORMATION, 38
 - getFunctionInformation, 38
 - getFunctionInformationFromFile, 39
 - getIN_OUT_FLAG, 39
 - getParamInformationFromFunctionDif-
initiation, 39
 - getPointerLevelFromFUNCTION_INFORMATION_-
LIST, 40
 - new_FUNCTION_INFORMATION, 41
 - new_PARAM_INFORMATION, 41
 - PARAM_INFORMATION, 38
 - printFUNCTION_INFORMATION_-
LIST, 41
 - searchFUNCTION_INFORMATION, 42
- FunctionPSI.h
 - getFunctionPSI, 131
 - getParameterPSI, 132
- generateMallocNumber
 - MallocNumber.h, 43
- get_ARRAY_OFFSET_LISTIgnoreASTNAME
 - PointerArrayControl.h, 54
- getArgumentAST
 - AST.h, 22
- getArgumentOffsetInfo
 - PointerArrayControl.h, 55
- getArgumentString
 - AST.h, 22
- getArgumentStringEnableExcept
 - AST.h, 23
- getARRAY_OFFSET_LIST
 - PointerArrayControl.h, 56
- getArrayOffsetInAnpasandInfo
 - PointerArrayControl.h, 56
- getArrayOffsetInIncDecInfo
 - PointerArrayControl.h, 57
- getASI_ARRAY_OFFSET_LIST
 - ExpressionStatementPSI.h, 127
- getAssignment_TYPE
 - SubEffectCheck.h, 72
- getASTList_FromVALIDATE_STATEMENT_-
LIST
 - Varidate_statement.h, 100
- getASTwithString
 - AST.h, 23
- getBasisLocationFromAssignmentExpression
 - Varidate_statement.h, 101
- getBasisLocationFromExpression
 - Varidate_statement.h, 101

- getCallocInformation
 - MemallocInfo.h, 45
- getDeclarationtPSI
 - DeclarationPSI.h, 126
- getDeclaratorArrayOffset
 - PointerArrayControl.h, 57
- getDeclaratorFromAST
 - Synbol.h, 76
- getDIVISION_INFORMATION_LIST
 - DivitionInformation.h, 30
- getExpressionOffsetInfo
 - PointerArrayControl.h, 58
- getExpressionStatementPSI
 - ExpressionStatementPSI.h, 128
- getFileName
 - StoreInformation.h, 124
- getFlagDatabase
 - FlagDatabase.h, 118
- getFOR_INFORMATION_LIST
 - ForInformation.h, 33
- getForStatementPSI
 - ForStatementPSI.h, 130
- getFreememInfo
 - FreeMemInfo.h, 36
- getFunctionGrobalVariable
 - ProgramSlicingInformation.h, 142
- getFunctionInformation
 - FunctionInformation.h, 38
- getFunctionInformationFromFile
 - FunctionInformation.h, 39
- getFunctionPSI
 - FunctionPSI.h, 131
- getIfStatementPSI
 - IfStatementPSI.h, 133
- getIN_OUT_FLAG
 - FunctionInformation.h, 39
- getInputFunctionPSI
 - ExpressionStatementPSI.h, 128
- getJumpStatementPSI
 - JumpStatementPSI.h, 134
- getLabeledStatementPSI
 - LabeledStatementPSI.h, 136
- getLeftAssignmentInfo
 - Varidate_statement.h, 101
- getMallocInformation
 - MemallocInfo.h, 45
- getMallocMaxsize
 - MemallocInfo.h, 46
- getMemberList
 - Synbol.h, 76
- getNewValidateStatementID
 - Varidate_statement.h, 102
- getOFFSET_LISTFromVariableTable
 - PointerArrayControl.h, 58
- getOffsetLevelFromArrayOffset
 - PointerArrayControl.h, 59
- getParameterData
 - Synbol.h, 77
- getParameterPSI
 - FunctionPSI.h, 132
- getParameterVARIABLE_TABLE_LIST
 - Synbol.h, 77
- getParamInformationFromFunctionDifinition
 - FunctionInformation.h, 39
- getPointerAccessOrIdentifierList
 - PointerArrayControl.h, 59
- getPointerArrayOffset
 - PointerArrayControl.h, 59
- getPointerLevelAndArrayLevel
 - Synbol.h, 78
- getPointerLevelAndArrayLevelFromVARIABLE_ -
TABLE
 - Synbol.h, 78
- getPointerLevelFromFUNCTION_INFORMATION_ -
LIST
 - FunctionInformation.h, 40
- getReallocInformation
 - MemallocInfo.h, 46
- getReturnStatementPSI
 - ReturnStatementPSI.h, 148
- getRightAssignmentInfo
 - Varidate_statement.h, 103
- getSingleExpressionOffsetInfo
 - PointerArrayControl.h, 60
- getStringFromAST
 - AST.h, 23
- getStringFromASTEnableExcept
 - AST.h, 24
- getStringReplaceASTtoString
 - AST.h, 24
- getSTRUCT_DATA
 - Synbol.h, 78
- getSTRUCT_TABLE_DATA
 - Synbol.h, 79
- getSwicthStatementPSI
 - SwitchStatementPSI.h, 149
- getTYPEDEF_TABLE_DATA
 - Synbol.h, 79
- getTYPEDEFfromAST
 - Synbol.h, 79

- getUpperExpressionRelationNode
 - PointerArrayControl.h, 61
- getValidate_Variable
 - Varidate_statement.h, 103
- getValidateStatementFromAssignStatement
 - Varidate_statement.h, 104
- getValidateStatementFromCallFunction
 - Varidate_statement.h, 105
- getValidateStatementFromForIteration
 - Varidate_statement.h, 105
- getValidateStatementFromInitializer
 - Varidate_statement.h, 106
- getValidateStatementFromMallocNumber
 - Varidate_statement.h, 108
- getValidateStatementFromPointerOperator
 - Varidate_statement.h, 109
- getVARIABLE_TABLE_LIST
 - Synbol.h, 80
- getVariableDeclarationFromEXPR_SLICING_LIST
 - ProgramSlicingInformation.h, 142
- getWhileStatementPSI
 - WhileStatementPSI.h, 151
- Help.h
 - viewHelp, 125
- IdList.h
 - IDLIST_compare_with, 121
 - printIDLIST, 122
 - SET_STACK_INTToIDLIST, 122
- IDLIST_compare_with
 - IdList.h, 121
- IfStatementPSI.h
 - getIfStatementPSI, 133
- INCLUDE_DATA
 - PreProcess.h, 68
- include_data, 11
- includeComment
 - PreProcess.h, 69
- initExprSlicingListFlag
 - ProgramSlicingInformation.h, 142
- initVALIDATE_STATEMENT_flag
 - Varidate_statement.h, 110
- insertMallocNumberHeadder
 - MallocNumber.h, 43
- isArrayUnboundCheckMode
 - FlagDatabase.h, 119
- isExpression
 - CharStringExtend.h, 115
- isFreeViolationCheckMode
 - FlagDatabase.h, 119
- isHelpMode
 - FlagDatabase.h, 119
- isProgramSlicingMode
 - FlagDatabase.h, 120
- isUndefineControlCheckMode
 - FlagDatabase.h, 120
- isXmlMode
 - FlagDatabase.h, 120
- isZeroDivitionCheckMode
 - FlagDatabase.h, 120
- JumpStatementPSI.h
 - getJumpStatementPSI, 134
- LabeledStatementPSI.h
 - getLabeledStatementPSI, 136
- Library/CharStringExtend.h, 114
- Library/CSTLString.h, 115
- Library/FlagDatabase.h, 118
- Library/IdList.h, 121
- Library/Stack_int.h, 122
- Library/StoreInformation.h, 124
- Main/Help.h, 124
- MallocNumber.h
 - generateMallocNumber, 43
 - insertMallocNumberHeadder, 43
- maxOffsetLevelAddressFromArrayOffsetList
 - PointerArrayControl.h, 62
- maxOffsetLevelFromArrayOffsetList
 - PointerArrayControl.h, 62
- MEMALLOC_INFO
 - MemallocInfo.h, 45
- MemallocInfo.h
 - getCallocInformation, 45
 - getMallocInformation, 45
 - getMallocMaxsize, 46
 - getReallocInformation, 46
 - MEMALLOC_INFO, 45
 - memoryAllocationAnarysis, 47
 - new_MEMALLOC_INFO, 47
 - new_MEMALLOC_INFO_char, 47
 - searchSizeof, 48
- memory_allocation_info, 11
- memoryAllocationAnarysis
 - MemallocInfo.h, 47
- minusArrayOffsetList
 - PointerArrayControl.h, 62

- moveArrayOffsetList
 - PointerArrayControl.h, 63
- multi_push_back_childrenAST
 - AST.h, 24
- new_ARRAY_OFFSET
 - PointerArrayControl.h, 63
- new_ARRAY_OFFSET_char
 - PointerArrayControl.h, 64
- new_AST
 - AST.h, 25
- new_DD_INFORMATION
 - ProgramSlicingInformation.h, 143
- new_DIVITION_INFORMATION
 - DivitionInformation.h, 31
- new_DIVITION_INFORMATION_char
 - DivitionInformation.h, 31
- new_EXPR_SLICING
 - ProgramSlicingInformation.h, 143
- new_FOR_INFORMATION
 - ForInformation.h, 33
- new_FREEMEMINFO
 - FreeMemInfo.h, 36
- new_FUNCTION_INFORMATION
 - FunctionInformation.h, 41
- new_INCLUDE_DATA
 - PreProcess.h, 69
- new_MEMALLOC_INFO
 - MemallocInfo.h, 47
- new_MEMALLOC_INFO_char
 - MemallocInfo.h, 47
- new_PARAM_INFORMATION
 - FunctionInformation.h, 41
- new_RETURN_INFO
 - Return_Info.h, 71
- new_STRUCT_TABLE
 - Synbol.h, 80
- new_STRUCT_TABLE_with_char
 - Synbol.h, 81
- new_TYPEDEF_TABLE
 - Synbol.h, 81
- new_TYPEDEF_TABLE_with_char
 - Synbol.h, 82
- new_VALIDATE_STATEMENT
 - Varidate_statement.h, 110
- new_VALIDATE_STATEMENT_char
 - Varidate_statement.h, 110
- new_VALIDATE_VARIABLE
 - Varidate_statement.h, 111
- new_VALIDATE_VARIABLE_with_char
 - Varidate_statement.h, 111
- new_VARIABLE_TABLE
 - Synbol.h, 82
- new_VARIABLE_TABLE_with_char
 - Synbol.h, 82
- OFFSET_LIST_push_back_alloc
 - PointerArrayControl.h, 65
- OutputSourceAfterDivitionDeclarator
 - DivitionDeclarator.h, 29
- PARAM_INFORMATION
 - FunctionInformation.h, 38
- param_information, 11
- PointerArrayControl.h
 - ARRAY_OFFSET, 50
 - ARRAY_OFFSET_LIST_push_back_-
ref_not_dup, 51
 - checkCallFunction, 51
 - checkIdentifierPointerArrayLevel, 51
 - checkIgnoreASTList, 52
 - copyArrayOffsetList, 52
 - createArrayExpression, 53
 - createValidateVariableArrayExpression,
53
 - deleteOFFSET_LIST, 54
 - deletePointer, 54
 - deletePointerAndArraySynbol, 54
 - get_ARRAY_OFFSET_LISTIgnoreASTNAME,
54
 - getArgumentOffsetInfo, 55
 - getARRAY_OFFSET_LIST, 56
 - getArrayOffsetInAnpasandInfo, 56
 - getArrayOffsetInIncDecInfo, 57
 - getDeclaratorArrayOffset, 57
 - getExpressionOffsetInfo, 58
 - getOFFSET_LISTFromVariableTable,
58
 - getOffsetLevelFromArrayOffset, 59
 - getPointerAccessOrIdentifierList, 59
 - getPointerArrayOffset, 59
 - getSingleExpressionOffsetInfo, 60
 - getUpperExpressionRelationNode, 61
 - maxOffsetLevelAddressFromArrayOff-
setList, 62
 - maxOffsetLevelFromArrayOffsetList,
62
 - minusArrayOffsetList, 62
 - moveArrayOffsetList, 63
 - new_ARRAY_OFFSET, 63

- new_ARRAY_OFFSET_char, 64
- OFFSET_LIST_push_back_alloc, 65
- printASTPOINTER_LIST, 66
- searchARRAY_OFFSET_LIST, 66
- searchExpressionOrPointerArrayOrIdentifier, 66
- searchOffsetLevelAddressFromArray-OffsetList, 67
- searchPointerAccessOrIdentifierOrPrimary, 67
- PreProcess.h
 - addIncludeDataFromFile, 69
 - adjustProgramStart, 69
 - INCLUDE_DATA, 68
 - includeComment, 69
 - new_INCLUDE_DATA, 69
 - preProcessor, 70
 - readIncludeDataFromFile, 70
- preProcessor
 - PreProcess.h, 70
- print_EXPR_SLICING_LIST
 - ProgramSlicingInformation.h, 144
- print_FOR_INFORMATION_LIST
 - ForInformation.h, 34
- print_tree_EXPR_SLICING_LIST
 - ProgramSlicingInformation.h, 144
- printASTPOINTER_LIST
 - PointerArrayControl.h, 66
- printDataFromAST
 - AST.h, 25
- printDIVISION_INFORMATION_LIST
 - DivisionInformation.h, 31
- printFREEMEMINFO
 - FreeMemInfo.h, 36
- printFUNCTION_INFORMATION_LIST
 - FunctionInformation.h, 41
- printIDLIST
 - IdList.h, 122
- printProgramDataWithValidateStatement
 - Varidate_statement.h, 112
- printSTRUCT_TABLE_LIST
 - Synbol.h, 83
- printTargetASTNode
 - AST.h, 25
- printTYPEDEF_TABLE_LIST
 - Synbol.h, 83
- printVALIDATE_VARIABLE_LIST
 - Varidate_statement.h, 113
- printVARIABLE_TABLE_LIST
 - Synbol.h, 84
- ProgramSlicing.h
 - createDD_list, 137
 - createDD_list_in_argument, 137
 - createDD_list_in_Function, 138
 - createDD_list_in_global, 138
 - createDD_listAll, 139
 - createStatementNodeList, 139
 - startStaticSlicing, 140
 - staticSlicing, 140
- ProgramSlicing/DeclarationPSI.h, 125
- ProgramSlicing/ExpressionStatementPSI.h, 126
- ProgramSlicing/ForStatementPSI.h, 130
- ProgramSlicing/FunctionPSI.h, 131
- ProgramSlicing/IfStatementPSI.h, 132
- ProgramSlicing/JumpStatementPSI.h, 134
- ProgramSlicing/LabeledStatementPSI.h, 135
- ProgramSlicing/ProgramSlicing.h, 136
- ProgramSlicing/ProgramSlicingInformation.h, 141
- ProgramSlicing/ReturnStatementPSI.h, 147
- ProgramSlicing/SwitchStatementPSI.h, 149
- ProgramSlicing/WhileStatementPSI.h, 150
- ProgramSlicingInformation.h
 - getFunctionGlobalVariable, 142
 - getVariableDeclarationFromEXPR_SLICING_LIST, 142
 - initExprSlicingListFlag, 142
 - new_DD_INFORMATION, 143
 - new_EXPR_SLICING, 143
 - print_EXPR_SLICING_LIST, 144
 - print_tree_EXPR_SLICING_LIST, 144
 - registerIncDecVariable, 145
 - searchDD, 145
 - searchDeclarationDD, 146
 - searchFunctionPSI, 146
 - setGlobalVariable, 147
- push_back_childrenAST
 - AST.h, 26
- readIncludeDataFromFile
 - PreProcess.h, 70
- registerIncDecVariable
 - ProgramSlicingInformation.h, 145
- RETURN_INFO
 - Return_Info.h, 71
- return_info, 12
- Return_Info.h
 - new_RETURN_INFO, 71
 - RETURN_INFO, 71

- ReturnStatementPSI.h
 - getReturnStatementPSI, 148
- same_new_AST
 - AST.h, 26
- searchARRAY_OFFSET_LIST
 - PointerArrayControl.h, 66
- searchDD
 - ProgramSlicingInformation.h, 145
- searchDeclarationDD
 - ProgramSlicingInformation.h, 146
- searchExpressionOrPointerArrayOrIden
 - PointerArrayControl.h, 66
- searchFOR_INFORMATION_FromAST
 - ForInformation.h, 34
- searchFUNCTION_INFORMATION
 - FunctionInformation.h, 42
- searchFunctionPSI
 - ProgramSlicingInformation.h, 146
- searchOffsetLevelAddressFromArrayOffsetList
 - PointerArrayControl.h, 67
- searchPointerAccessOrIdentifierOrPrimary
 - PointerArrayControl.h, 67
- searchSizeof
 - MemallocInfo.h, 48
- searchVARIABLE_TABLE_LIST
 - Synbol.h, 84
- SET_STACK_INTToIDLIST
 - IdList.h, 122
- setARGUMENT_NUMBER
 - ExpressionStatementPSI.h, 129
- setASTBlocklevelAndId
 - AST.h, 27
- setASTReturnTypes
 - AST.h, 27
- setFileName
 - StoreInformation.h, 124
- setGlobalVariable
 - ProgramSlicingInformation.h, 147
- setValidateVariableFromExprSlicing
 - Varidate_statement.h, 113
- Stack_int.h
 - STACK_INT_at_and_alloc, 123
 - STACK_INT_inclement_at, 123
- STACK_INT_at_and_alloc
 - Stack_int.h, 123
- STACK_INT_inclement_at
 - Stack_int.h, 123
- startStaticSlicing
 - ProgramSlicing.h, 140
- staticSlicing
 - ProgramSlicing.h, 140
- StoreInformation.h
 - getFileName, 124
 - setFileName, 124
- str_extract
 - CharStringExtend.h, 115
- STRUCT_TABLE
 - Synbol.h, 74
- struct_table, 12
- SubEffectCheck.h
 - checkContainSubEffectStatement, 72
 - getAssignment_TYPE, 72
- SwitchStatementPSI.h
 - getSwicthStatementPSI, 149
- Synbol.h
 - deletePointer, 75
 - deletePointerAndArraySynbol, 75
 - find_STRUCT_TABLE_DATA, 75
 - getDeclaratorFromAST, 76
 - getMemberList, 76
 - getParameterData, 77
 - getParameterVARIABLE_TABLE_LIST, 77
 - getPointerLevelAndArrayLevel, 78
 - getPointerLevelAndArrayLevelFromVARIABLE_TABLE, 78
 - getSTRUCT_DATA, 78
 - getSTRUCT_TABLE_DATA, 79
 - getTYPEDEF_TABLE_DATA, 79
 - getTYPEDEFfromAST, 79
 - getVARIABLE_TABLE_LIST, 80
 - new_STRUCT_TABLE, 80
 - new_STRUCT_TABLE_with_char, 81
 - new_TYPEDEF_TABLE, 81
 - new_TYPEDEF_TABLE_with_char, 82
 - new_VARIABLE_TABLE, 82
 - new_VARIABLE_TABLE_with_char, 82
 - printSTRUCT_TABLE_LIST, 83
 - printTYPEDEF_TABLE_LIST, 83
 - printVARIABLE_TABLE_LIST, 84
 - searchVARIABLE_TABLE_LIST, 84
 - STRUCT_TABLE, 74
 - TYPEDEF_TABLE, 74
 - VARIABLE_TABLE, 75
- traverseAST
 - AST.h, 27

- traverseASTwithXML
 - AST.h, 28
- TYPDEF_TABLE
 - Synbol.h, 74
- typedef_table, 13
- VALIDATE_STATEMENT
 - Varidate_statement.h, 88
- validate_statement, 13
- VALIDATE_STATEMENT_LIST_sort_ast
 - Varidate_statement.h, 114
- VALIDATE_VARIABLE
 - Varidate_statement.h, 88
- validate_variable, 14
- VARIABLE_TABLE
 - Synbol.h, 75
- variable_table, 15
- Varidate_statement.h
 - ArrayOffsetToValidateStatement, 88
 - createCheckUnboundAndUndefineOperationCheck, 89
 - createValidateStatementFromIncDecExpr, 90
 - createValidateStatement, 91
 - createValidateStatementAdderFileEachCheck, 92
 - createValidateStatementForFreeAction, 93
 - createValidateStatementForMallocAction, 94
 - createValidateStatementFromArrayDe-viewHelp, 95
 - createVaridateStatementFromPointerDefine, 95
 - createViolentFreeOperation, 96
 - createZeroDivitionCheck, 96
 - fprintProgramDataWithPSIVaridateStatement, 97
 - fprintProgramDataWithValidateStatement, 98
 - fprintValidateStatement, 99
 - fprintValidateStatement_not_assert, 100
 - getASTList_FromVALIDATE_STATEMENT_LIST, 100
 - getBasisLocationFromAssignmentExpression, 101
 - getBasisLocationFromExpression, 101
 - getLeftAssignmentInfo, 101
 - getNewValidateStatementID, 102
 - getRightAssignmentInfo, 103
 - getValidate_Variable, 103
 - getValidateStatementFromAssignStatement, 104
 - getValidateStatementFromCallFunction, 105
 - getValidateStatementFromForIteration, 105
 - getValidateStatementFromInitializer, 106
 - getValidateStatementFromMallocNumber, 108
 - getValidateStatementFromPointerOperator, 109
 - initVALIDATE_STATEMENT_flag, 110
 - new_VALIDATE_STATEMENT, 110
 - new_VALIDATE_STATEMENT_char, 110
 - new_VALIDATE_VARIABLE, 111
 - new_VALIDATE_VARIABLE_with_char, 111
 - printProgramDataWithValidateStatement, 112
 - printVALIDATE_VARIABLE_LIST, 113
 - setValidateVariableFromExprSlicing, 113
 - VALIDATE_STATEMENT, 88
 - VALIDATE_STATEMENT_LIST_sort_ast, 114
 - VALIDATE_VARIABLE, 88
- viewHelp
 - Help.h, 125
- WhileStatementPSI.h
 - getWhileStatementPSI, 151