adder

1.0

作成： Doxygen 1.7.3

Wed Feb 16 2011 22:08:29

Contents

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。

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。

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
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 <Synbol.h>`

変数

- CSTLString ∗ type
  行数
- CSTLString ∗ 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

- CSTLString ∗ 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

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
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 <Synbol.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 * initiarizer

  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/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す。 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ます。

```
#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)

4.1.1 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す。 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ます。

作者

faithnh

4.1.2 型定義

4.1.2.1 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

4.1.3 関数

4.1.3.1 void deleteAST (AST ∗ *root*)

指定した AST ノード以下を全て開放させる

引数

output	出力対象の文字列データ

戻り値

なし

4.1.3.2 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

root	探す対象の AST ノード
target	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。

4.1.3.3 void fprintDataFromAST (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

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

戻り値

なし

4.1.3.4 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

output	出力先のファイル構造体
root	指定された AST ノード
line	出力に利用する行数

output_subnode_num	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）
out_block_start	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
out_block_end	ブロックとして表記されている場合は、終わりブのロックへの AST ノードのアドレスを返却する。

戻り値

なし

4.1.3.5 int getArgumentAST (AST ** *output*, AST * *call_function*, int *arg_num*)

指定された call_function に相当する関数から、任意の引数目の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

output	取得する対象のノード
call_function	名前が call_function である AST ノード
arg_num	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

4.1.3.6 int getArgumentString (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

output	出力される引数の内容
call_function	名前が call_function である AST ノード
arg_num	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

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

getArgumentEnableExcept 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

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

getArgumentString 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

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

4.1.3.8 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

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

戻り値

見つけた AST ノードへのアドレス

4.1.3.9 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

output	出力対象の文字列データ
root	新しいノード名

戻り値

なし

4.1.3.10 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

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

戻り値

なし

4.1.3.11 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） 場合、そのノードだけ replace_string に変換して出力させる。

引数

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

戻り値

なし

4.1.3.12 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

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

戻り値

なし

4.1.3.13 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

newname	新しいノード名
new_content	新しい内容
new_line	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

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

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

引数

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

戻り値

なし

4.1.3.15 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

parent	指定された親 AST ノード
target	指定した文字列
traverse_-flag	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
xml_flag	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

4.1.3.16 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

parent	指定された親 AST ノード
child	指定された子 AST ノード

戻り値

なし

4.1.3.17 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

newname	新しいノード名
new_line	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

4.1.3.18 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

root	対象の AST ノード

戻り値

なし

4.1.3.19 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

root	対象の AST ノード

戻り値

なし

4.1.3.20 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

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

4.1.3.21 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

4.2 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)

4.2.1 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

4.2.2 関数

4.2.2.1 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

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

戻り値

なし

4.3 ANSICInformation/DivitionInformation.h

これは除算・剰余算を検出するために使用するのに必要な情報を確保する

```
#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 getDIVITION_INFORMATION_LIST (AST *expression_ast, FUNCTION_INFORMATION_LIST *function_information_list, VARIABLE_TABLE_LIST *vtlist, DIVITION_INFORMATION_LIST *divition_information_list, AST *target_expression, ASTPOINTER_LIST *ignore_ast_list)
- void printDIVITION_INFORMATION_LIST (DIVITION_INFORMATION_LIST *divition_information_list)

4.3.1 説明

これは除算・剰余算を検出するために使用するのに必要な情報を確保する

作者

faithnh

4.3.2 型定義

4.3.2.1 typedef struct divition_information DIVITION_INFORMATION

除算および剰余算に関する情報を格納する

4.3.3 関数

4.3.3.1 void getDIVITION_INFORMATION_LIST (AST * *expression_ast*, FUNCTION_INFORMATION_LIST * *function_information_list*, VARIABLE_TABLE_LIST * *vtlist*, DIVITION_INFORMATION_LIST * *divition_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

expression_ast	指定された式への AST ノードのアドレス
function_information_list	関数に関する情報のリスト
vtlist	変数テーブルリスト
divition_information_list	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
target_expression	対象の式への AST ノードのアドレス
ignore_ast_list	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

4.3.3.2 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

target_expression	対象の式への AST ノードのアドレス
type	除算か剰余かどうかのタイプ　０：除算式　１：剰余式
statement	除算および剰余以下の式
identifiers	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

4.3.3.3 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

target_-expression	対象の式への AST ノードのアドレス
type	除算か剰余かどうかのタイプ　0：除算式　1：剰余式
statement	除算および剰余以下の式
identifiers	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

4.3.3.4 void printDIVITION_INFORMATION_LIST (DIVITION_INFORMATION_LIST * *divition_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

divition_-information_-list	除算および剰余算に関する情報のリスト

戻り値

なし

4.4 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します。 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します。

```
#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)

4.4.1 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します。 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します。

作者

faithnh

4.4.2 型定義

4.4.2.1 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。

4.4.3 関数

4.4.3.1 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

freemeninfo	取得先のメモリ解放関係の情報
call_function	対象の関数への AST ノード
function_information_list	関数に関する情報のリスト
vtlist	変数テーブルの情報

target_-statement	対象の関数が位置している式への AST ノード

戻り値

なし

4.4.3.2 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

freememinfo	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。

4.4.3.3 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

freememinfo	出力するメモリ解放関係の情報

戻り値

なし

4.5 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ファイルである。

```
#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_informaition_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 getParamInformationFromFunctionDifinition (AST ∗param_info_node, PARAM_INFORMATION_LIST ∗param_information_list)
- int getIN_OUT_FLAG (CSTLString ∗param_type, int pointer_level, int array_level)

4.5.1 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ファイルである。

作者

faithnh

4.5.2 型定義

4.5.2.1 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。

4.5.2.2 typedef struct param_information PARAM_INFORMATION

引数に関する情報

4.5.3 関数

4.5.3.1 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

target_string	対象の関数の定義自体の文字列

戻り値

なし

4.5.3.2 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するし、それらのリスト function_information_list に登録する。

引数

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

戻り値

なし

4.5.3.3 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

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

戻り値

なし

4.5.3.4 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

param_type	パラメータの型名
pointer_level	ポインタのレベル
array_level	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。

4.5.3.5 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

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

戻り値

なし

4.5.3.6 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

target_function_name	指定した関数名

function_information_list	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

4.5.3.7 FUNCTION_INFORMATION* new_FUNCTION_INFORMATION (AST * *function_node*, CSTLString * *return_type*, CSTLString * *function_name*, PARAM_INFORMATION_LIST * *param_informaition_list*)

関数に関する情報を生成する。

引数

function_node	対象の関数へのノード
return_type	返却値のタイプ
function_name	関数名
param_information_list	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

4.5.3.8 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

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

4.5.3.9 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

function_information_list	出力対象の関数に関する情報リスト

戻り値

なし

4.5.3.10 FUNCTION_INFORMATION* searchFUNCTION_INFORMATION (CSTLString * *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

target_function_name	指定した関数名
function_information_list	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。

4.6 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ものです。 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。 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ます。

```
#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)

4.6.1 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ものです。 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。 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ます。

作者

faithnh

4.6.2 型定義

4.6.2.1 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

4.6.3 関数

4.6.3.1 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

root	
sizeof_type	sizeof での型
constant	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

4.6.3.2 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

root	
sizeof_type	sizeof での型
constant	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

4.6.3.3 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

constant	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）
root	malloc などの関数内の式への AST ノード

戻り値

なし

4.6.3.4 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

root	対象の AST ノード
vtlist	対象のプログラムの変数リスト
sizeof_type	sizeof での型
constant	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）
realloc	realloc が対象とするオフセット情報

戻り値

なし

4.6.3.5 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

root	指定された AST ノード
vtlist	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

4.6.3.6 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

sizeof_type	sizeof の型名
realloc_-target	realloc 時のターゲットタイプ
size	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

4.6.3.7 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

sizeof_type	sizeof の型名
realloc_-target	realloc 時のターゲットタイプ
size	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

4.6.3.8 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

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

戻り値

なし

4.7 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オフセットとして格納します。 各次元のオフセットとは動的配列や静的配列などにおいて、配列のどの部分を指しているかという式のことです。

```
#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 deletePointerAndArraySynbol (CSTLString *target)
- void deletePointer (CSTLString *target)
- int searchExpressionOrPointeArrayOrIden (AST *root, AST **output)
- void searchPointerAccessOrIdentifier (AST *root, AST **output, int *getSize)
- 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)

4.7.1 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オフセットとして格納します。 各次元のオフセットとは動的配列や静的配列などにおいて、配列のどの部分を指しているかという式のことです。

作者

faithnh

4.7.2 型定義

4.7.2.1 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

4.7.3 関数

4.7.3.1 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

target_-array_-offset_list	追加先の配列オフセットリスト
array_offset	追加する配列オフセット情報

戻り値

なし

4.7.3.2 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

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

戻り値

なし

4.7.3.3 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

identifier	識別子の名前
offset_level	オフセットレベル
variable_table_list	変数テーブルリスト
ignore_ast_list	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
pointer_level	変数リストから見つけた変数のポインタレベル
array_level	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。

4.7.3.4 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

ast_data	指定した AST ノード
ignore_ast_list	調べる対象の AST アドレスリスト
***ignore_ast_list* に存在する *AST* アドレスの場合１を返し、そうでない場合は０を返す。**	

4.7.3.5 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

fromlist	コピー元の配列オフセットリスト
tolist	コピー先の配列オフセットリスト
move_start	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

4.7.3.6 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

output	配列式を生成される文字列
array_offset	対象の配列オフセット
output_level	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

4.7.3.7 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

output	生成先の文字列
array_offset	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

4.7.3.8 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

offset_list	

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

offset_list	解放させる offset_list

戻り値

なし

4.7.3.9 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

target	変更対象の変数名

戻り値

なし

4.7.3.10 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

target	変更対象の変数名

戻り値

なし

4.7.3.11 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

root	オフセットリストに該当する AST ノード
array_offset_list	ポインタおよび配列のオフセット情報のリスト
function_information_list	関数に関する情報リスト
vtlist	検証対象の式をマークするための変数リスト
ignore_ast_list	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
target_statement	検証式の対象となるステートメント
ignore_ast_name	無視をするＡＳＴ名

戻り値

なし

4.7.3.12 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

argument	指定した引数に関する AST ノード
function_information_list	関数に関する情報のリスト
vtlist	検証対象の式をマークするための変数リスト
ignore_ast_list	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
array_offset_list	各ポインタおよび配列ごとのオフセットのリスト
target_expression	この左辺式の上位に位置する AST ノード

switch_mode	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

4.7.3.13 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

root	オフセットリストに該当する AST ノード
array_offset_list	ポインタおよび配列のオフセット情報のリスト
function_information_list	関数に関する情報リスト
vtlist	検証対象の式をマークするための変数リスト
ignore_ast_list	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
target_statement	検証式の対象となるステートメント

戻り値

なし

4.7.3.14 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

root	右辺式に関する AST ノード
function_information_list	関数に関する情報のリスト
vtlist	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
array_offset_list	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

ignore_ast_list	同じ位置のポインタが来ても無視するためのリスト
target_statement	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

4.7.3.15 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*)

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

root	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
function_information_list	関数に関する情報のリスト
vtlist	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
array_offset_list	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
ignore_ast_list	同じ位置のポインタが来ても無視するためのリスト
target_statement	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）
inc_dec_flag	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

4.7.3.16 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

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

戻り値

なし

4.7.3.17 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

expression	指定した式に関する AST ノード
function_-information_-list	関数に関する情報のリスト
vtlist	検証対象の式をマークするための変数リスト
ignore_ast_-list	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
array_-offset_list	各ポインタおよび配列ごとのオフセットのリスト
target_-expression	この左辺式の上位に位置する AST ノード
switch_-mode	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
allow_-subeffect	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

4.7.3.18 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

offset_list	生成先のオフセットリスト
variable_tabl	変数テーブルデータ

戻り値

なし

4.7.3.19 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＝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

array_offset	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

4.7.3.20 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

root	指定した AST ノード
output	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。

4.7.3.21 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

root	左辺値に関する AST ノード
function_information_list	関数に関する情報リスト
vtlist	検証対象の式をマークするための変数リスト
offset_level	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。
ignore_ast_list	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
array_offset	ポインタおよび配列のオフセット情報
target_statement	検証式の対象となるステートメント
anpasand_flag	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
inc_dec_flag	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

4.7.3.22 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

expression	指定した式に関する AST ノード
function_information_list	関数に関する情報のリスト
vtlist	検証対象の式をマークするための変数リスト
ignore_ast_list	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
array_offset_list	各ポインタおよび配列ごとのオフセットのリスト

target_-expression	この左辺式の上位に位置する AST ノード
switch_-mode	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

4.7.3.23 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

target	検証対象となっている変数
root	検証対象のノード
output	出力される間接的に関係しているノードへのアドレス
output2	output が minus_expr の場合、minus_expr 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

target	検証対象となっている変数
root	検証対象のノード
output	出力される間接的に関係しているノードへのアドレス
output2	output が minus_expr の場合、minus_expr 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。

4.7.3.24 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

array_-offset_list	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

4.7.3.25 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

array_-offset_list	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

4.7.3.26 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。

対象の配列オフセットリスト 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

4.7.3.27 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

fromlist	移動もとの配列オフセットリスト
tolist	移動先の配列オフセットリスト
move_start	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

4.7.3.28 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

variable_name	変数名
target_statement	ターゲットの statement
variable_address	この変数名が指している AST アドレス
offset_list	各次元のオフセット
pointer_level	この変数のポインタレベル
array_level	この変数の配列レベル
anpasand_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
inc_dec_flag	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

4.7.3.29 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

variable_name	変数名
target_statement	ターゲットの statement
variable_address	この変数名が指している AST アドレス
offset_list	各次元のオフセット
pointer_level	この変数のポインタレベル
array_level	この変数の配列レベル
anpasand_flag	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
inc_dec_flag	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

4.7.3.30 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

offset_list	対象のオフセットリスト
string	任意の文字列

戻り値

なし

4.7.3.31 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

astpointer_-list	AST のポインタリスト

戻り値

なし

4.7.3.32 ARRAY_OFFSET* searchARRAY_OFFSET_LIST (ARRAY_OFFSET_LIST * *array_offset_list*, CSTLString * *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、 見つかればその変数名へのアドレスを返す。

引数

array_-offset_list	探索対象の配列オフセットリスト
name	探索したい変数名

戻り値

見つか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。

4.7.3.33 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

root	指定した AST ノード
output	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。

4.7.3.34 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

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

4.7.3.35 void searchPointerAccessOrIdentifier (AST * *root*, AST ** *output*, int * *getSize*)

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

root	指定した AST ノード
output	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。

4.8 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）をさせるものです。 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ます。

```
#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 preProcesser (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)

4.8.1 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）をさせるものです。 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ます。

作者

faithnh

4.8.2 型定義

4.8.2.1 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。

4.8.3 関数

4.8.3.1 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

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

戻り値

なし

4.8.3.2 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

source_file_name	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

4.8.3.3 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

source_file_name	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

4.8.3.4 INCLUDE_DATA* new_INCLUDE_DATA (CSTLString * *include_data*, int *line*)

新しいインクルードファイルを生成する。

引数

include_data	include ファイルの名前
line	その行数

4.8.3.5 int preProcesser (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

source_file_name	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

4.8.3.6 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

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

戻り値

なし

4.9 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ものである

```
#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)

4.9.1 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ものである

作者

faithnh

4.9.2 型定義

4.9.2.1 typedef struct return_info RETURN_INFO

リターン命令に関する情報

4.9.3 関数

4.9.3.1 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

target_-expression	リターン命令自体へのノードへのアドレス
return_-array_-offset_list	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

4.10 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関数が含まれています

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

関数

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

4.10.1 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関数が含まれています

作者

faithnh

4.10.2 関数

4.10.2.1 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

node	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。

4.10.2.2 int getAssignment_TYPE (AST * *assignment_expression_list*)

代入式のタイプを出力させる

引数

assignment_expression_list	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:^=

4.11 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ことができます。 また、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す。

```
#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 *initarizer)
- 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 *initarizer)
- 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)

4.11.1 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ことができます。 また、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す。

作者

faithnh

4.11.2 型定義

4.11.2.1 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。

4.11.2.2 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。

4.11.2.3 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。

4.11.3 関数

4.11.3.1 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

target	変更対象の変数名

戻り値

なし

4.11.3.2 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

target	変更対象の変数名

戻り値

なし

4.11.3.3 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

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

戻り値

見つけられたら、１を返し、そうでなければ０を返す。

4.11.3.4 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

type	指定した型
ast_data	指定された AST ノード
member_list	登録先の変数リスト
enable_start	変数スコープの有効範囲の開始
enable_end	変数スコープの有効範囲の終わり
block_level	この変数のブロックレベル
block_level	ブロックを識別するための識別番号
declaration_-location_-address	この宣言自体への AST アドレス

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

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

4.11.3.5 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

member_list	登録対象のメンバリスト
ast_data	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

4.11.3.6 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

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

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

戻り値

なし

4.11.3.7 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

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

戻り値

なし

4.11.3.8 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

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

戻り値

なし

4.11.3.9 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

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

戻り値

なし

4.11.3.10 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

ast_data	指定された AST ノード
struct_table_data	登録先の構造体テーブルリスト

戻り値

なし

4.11.3.11 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

struct_table_list	登録先の構造体テーブルリスト
ast_data	指定された AST ノード

戻り値

なし

4.11.3.12 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

typedef_table_list	typedef テーブル
typelist	型リストへの AST ノード
identifier	識別への AST ノード

戻り値

なし

4.11.3.13 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

typedef_table_list	指定した typedef テーブルのリスト
token	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。 見つけられなければ、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

typedef_-table_list	指定した typedef テーブルのリスト
token	指定されたトークン
line	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

4.11.3.14 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

variable_-table_list	変数テーブルのリスト
ast_data	対象の AST ノード

戻り値

なし

4.11.3.15 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

line	行数
type	型名 (struct か union のいずれか)
struct_name	構造体の名前
member_list	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

line	行数
type	型名
struct_name	構造体名 (struct か union のいずれか)
member_list	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

4.11.3.16 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

line	行数
type	型名 (struct か union のいずれか)
struct_name	構造体の名前
member_list	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

4.11.3.17 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

target_type	typedef の対象の型
change_type	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

4.11.3.18 TYPEDEF_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

target_type	typedef の対象の型
change_type	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

4.11.3.19 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 * *initarizer*)

新しい変数テーブルのデータを生成させる

引数

enable_start	この変数の有効範囲の始まりの行数
enable_end	この変数の有効範囲の終わりの行数
declaration_location_address	この変数を宣言した場所を示す AST のアドレス
block_level	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）
block_id	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）
idlist	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）
type	型名
variable_name	変数名
initiarizer	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

4.11.3.20 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 * *initarizer*)

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

enable_start	この変数の有効範囲の始まりの行数
enable_end	この変数の有効範囲の終わりの行数

declaration_-location_-address	この変数を宣言した場所を示す AST のアドレス
block_level	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）
block_id	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）
idlist	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）
type	型名
variable_-name	変数名
initiarizer	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

4.11.3.21 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

struct_-table_list	出力対象の構造体テーブルのリスト

戻り値

なし

4.11.3.22 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

typedef_-table_list	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

typedef_table_list	出力対象の typedef テーブルのリスト

戻り値

なし

4.11.3.23 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

variable_table_list	出力対象の変数テーブルのリスト

戻り値

なし

4.11.3.24 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

target_idlist	指定した変数スコープの IDLIST
target_string	対象の変数名
variable_table_list	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。

4.12 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に使用する変数などを追加するためのものである

```
#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 createValidateStatemenFromIncDecExpr (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)

4.12.1 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使用する変数などを追加するためのものである

作者

faithnh

4.12.2 型定義

4.12.2.1 typedef struct validate_statement VALIDATE_STATEMENT

実際に検証式として挿入するための情報

4.12.2.2 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

4.12.3 関数

4.12.3.1 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

output	出力する検証式
VALIDATE_-VARIABLE_-LIST	検証用変数リスト
variable_tabl	対象の変数データ
offset_list	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

output	出力する検証式
validate_-variable_list	検証用変数リスト

variable_tabl	対象の変数データ
offset_list	オフセットリスト

戻り値

なし

4.12.3.2 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

VALIDATE_STATEMENT_LIST	追加先の検証式リスト
array_offset_list	配列やポインタなどのオフセット情報のリスト
array_unbound_check	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
undefined_control_check	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

ArrayOffsetToVaridateStatement の内部処理

引数

output	出力する検証式
validate_variable_list	検証用変数リスト
variable_tabl	対象の変数データ
offset_list	オフセットリスト
offset_level	今参照しているオフセットレベル

戻り値

なし

4.12.3.3 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 arrya_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

VALIDATE_STATEMENT_LIST	追加先の検証式
array_offset_list	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 arrya_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

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

戻り値

なし

4.12.3.4 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

root	検証式生成対象の AST ノード
function_-informaiton_-list	関数に関する情報のリスト
vtlist	検証対象の式をマークするための変数リスト
VALIDATE_-STATEMENT_-LIST	検証用変数リスト
VALIDATE_-VARIABLE_-LIST	取得した検証式が格納するところ
for_-information_-list	for 文に関する情報
undefined_-control_-check	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
zero_-divition_-check	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
array_-unbound_-check	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
free_-violation_-check	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

root	検証式生成対象の AST ノード
function_-informaiton_-list	関数に関する情報のリスト
vtlist	検証対象の式をマークするための変数リスト
validate_-statement_-list	検証用変数リスト
validate_-variable_list	取得した検証式が格納するところ
for_-information_-list	for 文に関する情報

undefined_control_check	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
zero_divition_check	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
array_unbound_check	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
free_violation_check	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

4.12.3.5 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を生成する

引数

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

4.12.3.6 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

variable_-statement_-list	生成先の検証式リスト
freememinfo	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

variable_-statement_-list	生成先の検証式リスト
freememinfo	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

4.12.3.7 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

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

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

4.12.3.8 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

validate_-statement_-list	追加先の検証式
validate_-table_list	変数テーブルリスト
function_-information_-list	関数定義に関する情報リスト

戻り値

なし

4.12.3.9 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 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

validate_statement_list	追加先の検証式
validate_table_list	変数テーブルリスト
function_information_list	関数定義に関する情報リスト

戻り値

なし

4.12.3.10 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

VALIDATE_STATEMENT_LIST	追加先の検証式リスト
freememinfo	メモリ解放関係の情報

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

validate_statement_list	追加先の検証式リスト
array_offset_list	配列やポインタなどのオフセット情報のリスト
array_unbound_check	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
undefined_control_check	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 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

validate_statement_list	追加先の検証式リスト
freememinfo	メモリ解放関係の情報

戻り値

なし

4.12.3.11 void createZeroDivitionCheck (VALIDATE_STATEMENT_LIST * *validate_statement_list*, DIVITION_INFORMATION_LIST * *division_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

VALIDATE_STATEMENT_LIST	格納先の検証式リスト
division_information_list	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

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

戻り値

なし

4.12.3.12 void fprintProgramDataWithPSIValidateStatement (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

output	出力する先のファイル
expr_slicing_list	プログラムスライシング情報
validate_variable_list	検証用変数リスト
validate_statement_list	検証式リスト
for_information_list	for 文に関する情報
check_target_ast	チェック検証式の対象への AST ノード

4.12.3.13 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

output	出力先のファイル構造体
root	プログラムへの AST ノード
VALIDATE_VARIABLE_LIST	検証用変数リスト
VALIDATE_STATEMENT_LIST	検証式リスト
for_information_list	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

output	出力先のファイル構造体
root	プログラムへの AST ノード
validate_variable_list	検証用変数リスト
validate_statement_list	検証式リスト
for_information_list	for 文に関する情報のリスト

戻り値

なし

4.12.3.14 void fprintValidateStatement (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

output	出力先のファイル構造体
VALIDATE_STATEMENT_LIST	出力対象の検証式リスト
target_ast	対象の AST ノード
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
allow_output_used_statement	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

output	出力先のファイル構造体
validate_statement_list	出力対象の検証式リスト
target_ast	対象の AST ノード
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
allow_output_used_statement	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

4.12.3.15 void fprintValidateStatement_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

output	出力先のファイル構造体
validate_statement_list	出力対象の検証式リスト
target_ast	対象の AST ノード
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
allow_output_used_statement	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

4.12.3.16 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

validate_statement_list	取り出し先の検証式リスト
ast_node_list	まとめる先の AST ノードリスト

戻り値

なし

4.12.3.17 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

output	出力先の CSTL 文字列
left_array_-offset_list	左辺値の配列オフセットリスト
right_-array_-offset_list	右辺式の配列オフセットリスト
right_-expression_-ast	右辺式への AST アドレス
a_op_flag	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

4.12.3.18 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

output	出力先の CSTL 文字列
array_offset	指定した識別子の配列オフセット
expression_-ast	指定した式への AST アドレス

4.12.3.19 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

left_-expression	左辺値に関する AST ノード
function_-information_-list	関数に関する情報のリスト
vtlist	検証対象の式をマークするための変数リスト
ignore_ast_-list	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

array_offset_list	各ポインタおよび配列ごとのオフセットのリスト
target_expression	この左辺式の上位に位置する AST ノード
switch_mode	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

4.12.3.20 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

VALIDATE_STATEMENT_LIST	対象の検証式リスト
target_statement	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

validate_statement_list	対象の検証式リスト
target_statement	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。

4.12.3.21 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

root	右辺式に関する AST ノード
function_information_list	関数に関する情報のリスト
vtlist	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
memalloc_info	malloc 関係の情報が出力される
array_offset_list	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
ignore_ast_list	同じ位置のポインタが来ても無視するためのリスト
target_statement	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

4.12.3.22 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

variable_table_list	プログラムの変数リスト
struct_table_list	構造体リスト
typedef_table_list	typedef リスト
validate_vari	検証用の変数リスト

戻り値

なし

4.12.3.23 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

root	指定したノード
function_-information_-list	関数に関する情報のリスト
vtlist	対象の変数リスト
validate_-variable_list	検証用変数リスト
validate_-statement_-list	取得した検証式が格納するところ
ignore_ast_-list	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
target_-expression	assign_expressionが属しているexpression_statement
undefined_-control_-check	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
zero_-divition_-check	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
array_-unbound_-check	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
free_-violation_-check	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

4.12.3.24 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

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

戻り値

なし

4.12.3.25 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

validate_statement_list	取得した検証式が格納するところ
for_information_list	for 文に関する情報のリスト
function_information_list	関数に関する情報のリスト
vtlist	対象の変数リスト
validate_variable_list	検証用変数リスト

ignore_ast_-list	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
undefined_-control_-check	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
zero_-divition_-check	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
array_-unbound_-check	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
free_-violation_-check	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

4.12.3.26 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

root	指定したノード
function_-information_-list	関数に関する情報のリスト
vtlist	対象の変数リスト
VALIDATE_-VARIABLE_-LIST	検証用変数リスト
VALIDATE_-STATEMENT_-LIST	取得した検証式が格納するところ
ignore_ast_-list	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
target_-expression	assign_expression が属している expression_statement

undefined_control_check	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
zero_division_check	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
array_unbound_check	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

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

root	指定したノード
function_information_list	関数に関する情報のリスト
vtlist	対象の変数リスト
validate_variable_list	検証用変数リスト
validate_statement_list	取得した検証式が格納するところ
ignore_ast_list	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
target_expression	assign_expression が属している expression_statement
undefined_control_check	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
zero_division_check	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
array_unbound_check	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

4.12.3.27 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

VALIDATE_STATEMENT_LIST	追加先の検証式リスト
call_function	関数呼び出しに対するノード
right_array_offset_list	左辺値に関する配列オフセットリスト
memalloc_info	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

validate_statement_list	追加先の検証式リスト
call_function	関数呼び出しに対するノード
right_array_offset_list	左辺値に関する配列オフセットリスト
memalloc_info	メモリ確保情報

戻り値

なし

4.12.3.28 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

VALIDATE_STATEMENT_LIST	検証式リスト

left_array_-offset_list	左辺値の配列オフセットリスト
right_-array_-offset_list	右辺式の配列オフセットリスト
right_-expression	右辺式へのＡＳＴアドレス
a_op_flag	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

validate_-statement_-list	検証式リスト
left_array_-offset_list	左辺値の配列オフセットリスト
right_-array_-offset_list	右辺式の配列オフセットリスト
right_-expression	右辺式へのＡＳＴアドレス
a_op_flag	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

4.12.3.29 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

4.12.3.30 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

4.12.3.31 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)
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

4.12.3.32 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

used	この検証用変数テーブルを使用したかどうか
enable_start	この変数の有効範囲の始まりの行数
enable_end	この変数の有効範囲の終わりの行数
declaration_location	この変数を宣言した場所の行数
block_level	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）
block_id	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）
type	型名
variable_name	変数名
target_variable_name	検証対象の変数名
offset_level	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

4.12.3.33 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

used	この検証用変数テーブルを使用したかどうか
enable_start	この変数の有効範囲の始まりの行数
enable_end	この変数の有効範囲の終わりの行数
declaration_location	この変数を宣言した場所の行数
block_level	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）

block_id	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）
type	型名
variable_name	変数名
target_variable_name	検証対象の変数名
offset_level	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

4.12.3.34 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

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

root	プログラムへの AST ノード
validate_variable_list	検証用変数リスト
validate_statement_list	検証式リスト

for_information_list	for 文に関する情報のリスト

戻り値

なし

4.12.3.35 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

variable_table_list	出力対象の検証用変数テーブルのリスト

戻り値

なし

4.12.3.36 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

validate_variable_list	検証用変数リスト
expr_slicing_list	プログラムスライシング情報のリスト

戻り値

なし

4.12.3.37 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

validate_-statement_-list	ソート対象の検証式リスト

戻り値

なし

4.13 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ものです。 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たりといったことができます。

関数

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

4.13.1 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ものです。 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たりといったことができます。

作者

faithnh

4.13.2 関数

4.13.2.1 int isExpression (char * *source*)

文字列 source は式であるかどうかを調べる

引数

source	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

source	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

4.13.2.2 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

dest	抽出結果を入れる文字列
source	抽出対象の文字列
start	抽出の開始位置
strlen	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

4.14 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います。

`#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)

4.14.1 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います。

作者

faithnh

4.14.2 関数

4.14.2.1 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

target1	指定した CSTL 文字列
target2	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

4.14.2.2 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

target1	指定した CSTL 文字列
del_str	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。

4.14.2.3 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

target	指定した文字列

4.14.2.4 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

target	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）
add_flag	追加挿入するかどうかのフラグ　１：追加　０：代入
source	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

4.14.2.5 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

target	指定した CSTL 文字列
search	置換対象の文字列
teplace	置換したい文字列

戻り値

なし

4.15 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ものです。 たとえば、xml として出力するとかなどといった設定はこちらに入ります。

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)

4.15.1 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ものです。 たとえば、xml として出力するとかなどといった設定はこちらに入ります。

作者

faithnh

4.15.2 関数

4.15.2.1 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

argc	引数フラグの数
argv	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

4.15.2.2 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

flag_-database	フラグデータベース

戻り値

ARRAY_UNBOUND_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

4.15.2.3 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

flag_-database	フラグデータベース

戻り値

FREE_VIOLATION_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

4.15.2.4 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

flag_-database	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

4.15.2.5 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

flag_-database	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

4.15.2.6 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

flag_-database	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

4.15.2.7 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

flag_-database	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

4.15.2.8 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

flag_-database	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

4.16 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します。

```
#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)

4.16.1 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します。

作者

faithnh

4.16.2 関数

4.16.2.1 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

target1	比較対象の IDLIST １
target2	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。

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

IDLIST の内容を出力する

引数

idlist	出力対象の IDLIST
new_line_flag	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

4.16.2.3 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 IDLISTdest へ入れる。

引数

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

戻り値

なし

4.17 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ものです。 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ます。

```
#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)

4.17.1 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ものです。 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ます。

作者

faithnh

4.17.2 関数

4.17.2.1 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

stack_int	対象の整数スタック
index	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。

4.17.2.2 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

stack_int	対象の整数スタック

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。

4.18 Main/Help.h

このファイルはヘルプ出力関係のファイルである

関数

- void viewHelp (void)

4.18.1 説明

このファイルはヘルプ出力関係のファイルである

作者

faithnh

4.18.2 関数

4.18.2.1 void viewHelp (void)

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

戻り値

なし

4.19 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操作が含まれている

```
#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)

4.19.1 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操作が含まれている

作者

faithnh

4.19.2 関数

4.19.2.1 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

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

4.20 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操作が含まれている

```
#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)

4.20.1 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操作が含まれている

作者

faithnh

4.20.2 関数

4.20.2.1 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

expr_slicing_list	追加先のプログラムスライシングリスト expr_slicing_list
function_definition	関数定義に関する AST ノード
parent	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

4.20.2.2 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

expr_slicing_list	関数定義のプログラムスライシング
parameter_node	各パラメータの定義リスト

basis_-parameter_-node	パラメータを指す AST ノード

4.21 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操作が含まれている

```
#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)

4.21.1 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操作が含まれている

作者

faithnh

4.21.2 関数

4.21.2.1 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

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

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

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

4.22 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操作が含まれている

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

関数

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

4.22.1 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操作が含まれている

作者

faithnh

4.22.2 関数

4.22.2.1 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

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

4.23 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操作が含まれている

```
#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)

4.23.1 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操作が含まれている

作者

faithnh

4.23.2 関数

4.23.2.1 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

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

4.24 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ものである。 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)

4.24.1 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ものである。 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

4.24.2 関数

4.24.2.1 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

expr_slicing	指定した対象の命令

戻り値

なし

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

expr_slicing	指定した対象の命令

4.24.2.2 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

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

戻り値

なし

4.24.2.3 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

expr_slicing	関数呼び出しが含まれている処理名
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
function_information_list	関数に関する情報リスト

戻り値

なし

4.24.2.4 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

expr_slicing	指定した対象の命令

program_-expr_-slicing_list	プログラム全体のスライシングリスト

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

expr_slicing	指定した対象の命令
program_-expr_-slicing_list	プログラム全体のスライシングリスト

4.24.2.5 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

expr_-slicing_list	指定した対象のプログラムスライシングリスト
function_-information_-list	関数に関する情報リスト
program_-expr_-slicing_list	プログラム全体へのスライシングリスト

戻り値

なし

4.24.2.6 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ＡＳＴ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

root	指定したプログラムＡＳＴノード
expr_-slicing_list	スライシングを行うための構造体
parent	親のスライシングデータ
vtlist	変数テーブルリスト
function_-information_-list	関数に関する情報のリスト

戻り値

なし

4.24.2.7 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

inst_ast	指定した命令への AST ノード
expr_-slicing_list	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

4.24.2.8 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

identifiers	指定した識別子の配列オフセットリスト
line	指定した行数
expr_-slicing_list	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

なし

4.25 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操作が含まれている

```
#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)

4.25.1 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操作が含まれている

作者

faithnh

4.25.2 関数

4.25.2.1 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

expr_slicing_list	追加先のプログラムスライシングリスト expr_slicing_list
return_statement	GOTO、continue、break 文に関する AST ノード
parent	ノードを追加するときの親ノード
vtlist	変数テーブルリスト
function_information_list	関数に関する情報リスト

ignore_ast_-list	重複防止のために無視するノードリスト

4.26 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操作が含まれている

```
#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)

4.26.1 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操作が含まれている

作者

faithnh

4.26.2 関数

4.26.2.1 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

expr_-slicing_list	追加先のプログラムスライシングリスト expr_slicing_list

switch_statement	switch 文に関する AST ノード
parent	ノードを追加するときの親ノード
vtlist	変数テーブルリスト
function_information_list	関数に関する情報リスト
ignore_ast_list	重複防止のために無視するノードリスト

Index