C Static Analyzer
Backend Specifications

Frederico Falcao

v0.3
Nov 2019

Global Variables

i (int) - this is a generic integer to be used globally on the code.

j (int) - this is a generic integer to be used globally on the code.

name (char*) - this is the name of the user.

() -

Makefile

```
SHELL=/bin/bash

compile: doc/global-variables.tex
  cd doc; pdflatex main.tex || true
  #cd doc; pdflatex main.tex || true
  mv doc/main.pdf main.pdf
clean:
  rm doc/*.aux doc/*.log doc/*.out doc/*.pdf doc/global-variables.tex
   precompile.db

doc/global-variables.tex: precompile.db
  sqlite3 --separator ' ' precompile.db 'SELECT name,type,description FROM
    globalVars' | doc/global-variables.tex.php > doc/global-variables.tex

precompile.db:
  echo > precompile.db # Clear the database
  sqlite3 precompile.db 'CREATE TABLE globalVars(type text , name text,
   description text, version text)'
  sqlite3 precompile.db 'INSERT INTO globalVars(type,name,description,
   version)VALUES("int","i","this is a generic integer to be used globally
    on the code.","1.0")'
  sqlite3 precompile.db 'INSERT INTO globalVars(type,name,description,
   version)VALUES("int","j","this is a generic integer to be used globally
    on the code.","1.0")'
  sqlite3 precompile.db 'INSERT INTO globalVars(type,name,description,
   version)VALUES("char*","name","this is the name of the user.","1.0")'
```

Main.tpl.c

```c
#include <stdio.h>

int main(int argc, char**argv) {


  return 0;
}
```

1 BACKLOG

```
----------
v4.0 - AUTO DOCUMENTATION
-----------

v4.2 - Include a script to list all global variables

v4.3 - Includes CHANGE LOG ( the description of all previous commits, i.e. git log outpu

v4.4 - Includes source code
- the template files inside modules (*.tpl.c)
- the main script to trigger the make process (i.e. Makefile)



v0.3.5  - ABSTRACT VARIABLE DEFINITIONS:

- New File: precompile.php

- the script contains mostly i/o functions, except:
- header code:
- connection to local database sqlite3 file
---- SAMPLE PSEUDO CODE ----
- $dbHandle = new SQLite3($filename);
-  dbQuery($sqlQuery)
 $dbHandle;
 (strpos($sqlQuery, "SELECT") === 0)
$result = $dbHandle->query($sqlQuery)
 $return->fetchArray(SQLITE3_ASSOC)

 $dbHandle->query($sqlQuery)


- footer code:
---- SAMPLE PSEUDO CODE ----
- check if filename passed in command line is valid
- parse in the current environment
- i/o functions
- receive one or more arguments with C keywords, variable names, or literals
- store the passed data into the sqlite database
- return C-code (to be passed to the compiler)
- two functions:
```

```
- declareGlobalVar(string type, string name, string version)
- declareLocalVar (string type, string name, string version)


#
# Use a script to generate boilerplate comments.
#
# - the comments help programmers writing the modules to have context
# - the script implements the functions:
#  (1) declareGlobalVar()
#  (2) declareLocalVar()
#  (3) defineFunction()
#  (4) endFunctionDef()
#  (5) triggerEvent()
# - the comments include:
#  - local variables available
#  - global variables available
#  - functions available
#  - the current code block (while loop, if block, function, etc..)

v0.3.99 - MODULARIZE:

New Architecture: (1) Event Driven Modules, (2) Replace C preprocessor with PHP script

Each Sub-folder is a module
- each module can respond to one or several events
- (kind of like a library, a collection of functions)
- each module can declare variables that its functions can use
- each responding event function starts with "will..." or "did..." - inspired by Swift

- the precompiler will figure out things based on the extension of each file inside the

v0.4.99 - IMPROVEMENTS

- NEW MOULE: print the number of words
- as well as the number of lines
```