Task documentation CHA: C header analysis in PHP 5 for IPP 2016/2017
Name and surname: Juraj Ondrej Dúbrava
Login: xdubra03

CHA PHP script was made for analysing C header files and searching for declarations and definitions of C functions in C99 standard. Script has some input parameters from user which are processed at first. For each specified parameter is created index in array of options. If entered parameters does not match one of the specified option or duplicate options are entered, script ends with error and error message is printed on stderr.

File processing

Script searches for functions in a file or directory according to input option. This could be directory or a file. At first script checks if entered input exists and whether we have permission to open file or directory. If control fails, error message is printed on stderr. If input is directory and was correctly processed, script searches for header files in this directory and its subdirectories using function searchdir(). If there is no permission for subdirectory, script ends with error. Output from searching directory is an array of all C header filenames. If input was a file, output array contains only entered file.

Cleaning file

Searching for function definitions and declarations is performed on an array of correct filenames. At first, the process contains removing all unnecessary objects from header file. Script removes singleline comments, multiline comments, single and multiline macros, strings and structure definitions.
Removing is performed in clear_file() function and for removing are used regular expressions and 2 functions macro() and mul_comment() which performs removing multiline macros and comments. Function clear_file returns cleared file for further processing.

Regular expression for matching functions

Searching for functions is performed on output from clear_file() function. In this file could be applied regular expression matching C function constructions. Regular expression is divided into 3 parts: (return type) (function name) (arguments) ({ or ;). After applying regular expression, 3 arrays are created. First array contains return types of found functions, second contains names of functions and the third contains arguments for each matched function.

Function processing

According to number of functions, script performs processing single function. While processing actual function, arrays with function informations need to be checked. Return type and name are checked for containing incorrect constructions as if else statements. Return type is also checked for no-inline option,when inline functions are skipped and remove-whitespaces option, controls are performed by regular expressions. If no-duplicates option is set, array of function names is created and while processing functions, function name which isn't in array is stored there. If actual processed function has name which is already in an array, function is skipped. Function return type and name are then stored and script continues with processing function arguments. Arguments are firstly splitted from record of arguments array, script can then process function arguments independently. Each argument is checked for matching pattern of regular expression (type) (name) for correct argument and variable arguments also. Type of correct parameter is then added to an array of arguments types and counter of arguments is increased. If function has no parameters, any types are added to this array.

Output

Script uses XML Writer for creating XML structure with elements containing function informations.
Before creating new XML function element, check for maximum-parameters option is performed. If value of argument counter is bigger than this number, we skip creating element for this function. For each correct function is created new element function in XML file with its return type and name. If argument counter value is bigger than 0, then for each of arguments in array of arguments is created param element in XML file. Each param element contains its order number and type.