Task documentation CHA: C header analysis in PHP 5 for IPP 2016/2017
Name and surname: Adrián Tóth
Login: xtotha01

My solution of this project can be divided into these following parts: analysis of parameters, file/files managing, file content formatting, file content analysis, output writing. Structure of the script is based on functions. At the beginning of the script there are global variables defined and after them there is a main function call. Every other function is called inside of this main function.

1.) Analysis of parameters:

This is the first part of the script, where global variables, which will be used later, are initialized. Some of them have informative value, if there was some parameter entered, then they have value set true, false if not. Also, the string or number after the given parameter is saved for later usage, when I will have to analyse string after input and output.

Functionality of this part can be described as iterating over every single parameter. Parameters have a specified form, so every one of them has to match at least one regular expression representing the correct type of form. If they don't match, it detects an unknown parameter. If the parameter match the form, badge is set to know it was loaded so it cannot be loaded twice or more times, bypassing duplicates. Entered values after the options such as fileordir(input), filename(output), k(pretty-xml), n(max-par) are saved separately into global variables.

2.) File/Files managing:

First step is to set the output file. If the option output was entered, then check if the filename which was saved in global variable is correct. It means, that the file exists and is writeable. Otherwise, when there was not any output given by user, output is sent to standard output.

Next step is checking given string after input. I verify what represents this string (file or directory) and on this basis my script continues. If it is a file, script analyses it´s content in next following parts. At a directory, script iterate through it, looking for every file with extension .h and file by file analyses it´s content. If one of these files, which were found cannot be open for reading, script ends with an error and the xml output has an undefined format.

3.) File content formatting:

File content is loaded into string. This string goes through filters which remove useless information from it via regular expressions. Filtering is in these steps: remove carriage return sign, remove backslash followed by quote, remove backslash followed by apostrophe, remove block comments, remove strings in quotes, remove strings in apostrophes, remove multiline line comments, remove line comments, remove multiline macros, remove single line macros. After this filtering string is modified in this last step, replacing every sequence consisting from one or more newlines with spaces depending on the size of sequence.

Filtering is provided by regular expression via inbuild functions.

4.) File content analysis:

String entering this part is cleaned from useless data which could do unexpected things. This content includes the function declarations and definitions for what I am searching for. To match these function, I use regular expression which type is:

return_type + S{1,n} + function_name{1} + S{0,n} + ({1} + [SC]{0,n} +){1} + ; OR {

n represents infinity
S represents whitespace character
C represents non-whitespace character
[ ] represents a set
{x,y} represents count from *x* to *y*
function_name is representing a identifier as in C99 (consisting of *[_a-zA-Z][_a-zA-Z0-9]{0,n}*)
return_type is representing a sequention of identifiers and stars (*[*]{0,n}*)

These functions are saved in array (array of strings). After this, this array of strings is iterated trough, searching for strings which have the form as function, but they are not functions. For example, "else if(true) {". These nonfunctions are removed and after this, a transformation will be done.

Array of strings (array of functions) is transformed into array of structures (one structure represents one function). Structure is an array of 7 arrays of characters (7 strings) and 1 array of array of characters (array of strings), where are types of parameters. After this operation, this array of structures is modified depending on parameters indicating what type of functions my script have to print out. Ineligible functions are removed. For example, no-inline or function which have more than n parameters. Whitespace modification is done here too. Array of structures is ready to be printed out at the end of this section.

5.) Output writing:

Incoming array of structures is iterated trough, and structure by structure is printed out. Xmlwriter is used for it. Xmlwriter is customized for printing in the main function.