Task documentation CHA: C Header Analysis in PHP5 for IPP 2016/2017

Name and Surname: Ján Gula

Login: xgulaj00

Task

Our task was to analyze header files written in C language. We could assume that each header file would fulfill the ISO C99 norm. The goal of our analysis was to find all declarations or definitions of functions in given files. Each function found would then be written into an XML structure, which was the output of our code. Although this project was implemented in the PHP5 language it acts as a script. My implementation is mostly built on regular expressions.

Analyzing input parameters

To begin with our script had to analyze input parameters given from the user. The usage of parameters is described in the script's `––help` parameter. My script loads input parameters with the function `getopts.` Afterwards I check if some parameter isn't defined more than once. If the user sets the `help` parameter we need to check if it's the only parameter set. If it's not, an error is called and a value other than zero is returned.

Removing unnecessary characters

Secondly, if a directory was given, my script would iteratively search through it and look for header files. Those that were found were stored in an array. Then I've cycled through the array and stored the content of each file into a new variable, making it easier to filter the content later. Removing unnecessary characters was a process of performing multiple regular expression matches. At the very beginning I had to delete all macros, comments and strings. Each part of the code needed specific information from a function so first my script had to chunk the content into separate functions. Then each function was cut to contain only the return type and function name. With these information I could fill the XML structure elements. The second part of this section was to cut the functions to leave only the parameters of each function. The parameters were stored in another variable but they were all together and we needed them one-by-one so another regular expression match had to be performed splitting the parameters separately. We took the return type and name from each parameter and filled the remaining XML elements. Each parameter created its own XML element `param.` Only thing left to do in this section was to check if there were any variable arguments. We could do that simply by searching for "`/\.\.\./"` which is a regular expression matching three dots.

Implementing input parameters

When we filtered all functions and filled each XML element, we could proceed into implementing each of our possible input parameters. To implement `remove-whitespace` and `no-inline` I've used `preg_match` and `preg_replace.` To find out which parameters were present at the start I've used `Boolean` variables that were set to false as a default value. In case of parameters like `--max-par` we could simply use an if statement with a continue. Parameter `--pretty-xml` was implemented by multiple commands that touched only the final version of my output. I've added spaces and newlines before every XML element according to the indent given, or with the default indent.

Creating XML document

To create our output XML document I've used the `DOMDocument` function. This created an XML structure that was the parent to all other XML elements. These XML elements were created by the `createElement` function in a cycle after each read function or each read parameter. Then I had to set attributes for these elements that described what does each element represent, e.g. "functions" or "rettype". To ensure that the XML element would be displayed as a child to another element I've used the `appendChild` function included in the `DOMDocument` set.

Conclusion

In this project I had to solve multiple conflicting situations. Primarily it was the definition of a function with no parameters. There are two ways to write this example. **1**: `void func();` **2**: `void func(void).` At first my script thought that the second variant means that there is one parameter called void. To solve this problem I had to implement an if statement with a continue if the parameter is “`void`” or empty. Another problem was that even if the user didn’t set the `pretty-xml` parameter, my XML header would be separated by a newline, which was prohibited according to the assignment. I’ve solved this problem by replacing all “`\n`” with empty strings if `remove-whitespace` parameter wasn’t present. This project showed me that PHP is a very flexible language and that it’s easy to work with variables and strings in general because of the wide span of built in functions supporting work with strings and arrays. I’ve also appreciated built in functions for regular expressions which made it easier to filter data.