Documentation of CST: C Stats in PHP for IPP 2014/2015

Name and surname: Martin Kačmarčík

Login: xkacma03

1. Purpose of script:

Write statistics of C source and header files.

2. Argument and file processing

For argument processing I am using in my script the function `getopt` which allows me to easily process arguments. With the processing I also set control flags, like `$foundHelp` which control whether help param occured and in that case any other occurence of params will end with error. I also save arguments into associative array in format: "Name of param => true/false".

After arguments are processed, function `argumentSwitch` is called. This function will determine next steps based on array mentioned above. In all cases this function call `fileOrDir` function which return 1 when dir should be searched, 2 if file should be processed or 3 if input wasnt even given so script should search everything. With this information, `handleMode` function is called. `HandleMode`, based on previous returned value, search (or just open file given) all the directories and put the open stream of files into associative array with name of the file as key and stream of file as value. Next, function returns this array.

With array full of names and streams, script iterate trough it using foreach cycle and call for each file appropriate function (`k`, `o`, etc..), save returned number into variable (number represent occurences found) and put it into new associative array with only changed value - the number returned (representing occurences found). With this `$arrayOfResults`, function `processResult` is called. This function change directory into directory with script, delete and create new file if output was given, calculate the spaces that need to be put between number of occurences and the path of file and write it into file or stdout. Spaces are calculated using this algorithm:

```
$numberOfSpaces = $greatestLenght - $currentLenght + ($biggestNumberLenght
- $currentNumberLenght);
```

where `$greatestLenght` is greatest lenght of string with filename (and path if needed), `$currentLenght` is lenght of the current string processed and `$biggestNumberLenght` is lenght of greatest number etc.

3. Statistic making functions

I have five functions (based on params) that create the statistics. It is k, o, i, w and c. All functions are using regular expresions to find occurences of their tasks. In next lines I will describe each function one by one. Many functions get rid of comments and macros using `removeCommentsFromString` function. This function was the hardest to make, because it use regular expresions to get rid of comments and it needed tons of debuging to create the propper one.

3.1 The `k` function

In `$arrayOfKeyWords` there are all keywords of C99 standard. Then `foreach` cycle iterate and try to find them using regex and `preg_match_all` function. Next it return count with number of occurences.

3.2 The `o` function

This function iterate trough array with operators and using regular expression it count the occurence of each operator. Regular expressions containts some rules that operator has to stick to.

3.3 The `i` function

It finds all identificators, even the keywords. Next `k` function is called and the returned value is substracted from final result. It use regular expressions.

3.4 The `w` function

This function just call `preg_match_all` for pattern given and return number of occurences.

3.5 The `c` function

It benefit from the function which delete all the comments. Except from deleting them, it save them and then use strlen to count the lenght of comments. Then it just return the value with lenght.

4. Last note

I just want to add that all regular expressions are clever and they use the rules of C language. For example we can easily determine whether the '*' character is used in decleration or not. The regular expression for it can look like this: `"~".$type."\s*\*+\s*[a-zA*Z_]~"` (find all occurences that start with type then 0~n white spaces, the '*' character, 0~n with spaces, then identificator has to start.)