Documentation of CHA: C headers analysis script in Python for IPP 2014/2015
Name and surname: Martin Kačmarčík
Login: xkacma03

1 Purpose of the script:

Create XML file with statistics of C header files.

2 Argument and file processing

For argument processing I am using `argparse`. Using this library I set dictionary `argumentArray` where I have stored parameters and their values. For file processing I am using `os.walk` function. It is easy way to find all .h files. All the files are stored into `fileList` variable.

3 Creating XML string

Root elements are automaticly written using `sys.stdout.write` function. To search functions and their parameters I use `Python's` regular expressions functions and my regular expressions patterns. I think it is the most effective way of the search, because with one pattern I can describe any possible occurrence of any function.

3.1 Functions processing

With all files in the list, I call for each file the `returnFunctionString` function which returns me list of functions in the file being processed. Functions are found using regular expression as mentioned. Creating this regex was one of the hardest part of the project. I spent circa 2 days debugging it. Also there was problem that `Python` doesn't support all the tricks that others languages like `PHP` do.

With the list of functions I can call `returnArgumentString` which write down both function elements and their parameters (it is done in here because before this function I don't know the number of parameters, which is needed when `--max-par` parameter is given). This is achieved by almost the same regex as the one for functions, but slightly adjusted.

Also a bit tricky was finding names of functions. Regex from functions needed to be adjusted pretty heavily and one had to be aware of the possibility to have asterisk before name with or even without white spaces between them. This regex took some time too.

4 Few words about parameters

There were some problems with few parameters. In next few lines I will describe them a little.

4.1 Pretty-xml

This one was very tricky. For argument parsing firstly I used getopt, however getopt doesn't support optional options. So I couldn't achieve to have both `--pretty-xml` and `--pretty-xml=k` parameters. I needed to use another parameter parsing function and even in the `argparse` it wasn't the easiest to achieve, although it wasn't that hard either. The execution of the parameter was simple. If it is given just add spaces before string written, one-time for the first level and two times for second level.

4.2 No-inline

Very easy parameter. Using simple regex I found all function that contain `include` and removed them from list so they were not processed further.

4.3 Max-par

This one was relatively easy, although I needed to move the `function` element print into the parameter processing function. After that I just write out these function that have length of their `paramList` equal or lesser than `k`.

4.4 No-duplicates

To achieve this parameter I needed to create list of already processed functions and then, when the parameter was set, always check whether there wasn’t the same name of the function printed before. If the function name was found in the list, it was skipped.

4.5 Remove-whitespace

Maybe the easiest one. Python regex has very nice tools for working with spaces. So I found occurrence of all white-spaces and replaced them with single space. After that I found all asterisks and deleted spaces in left and right from the asterisk.