Documentation of Project Implementation for IPP 2018/2019

Name and surname: Tomáš Žigo

Login: xzigot00

1. Structure

Script `parse.php` is created by three classes:

Class `arguments`

Class controls all the work with the arguments. It performs check over possible combinations of arguments and count of arguments. This class is also responsible for opening file and writing `STATP` statistics to file.

Class `checker`

Class checker executes partial checking procedures needed in lexical and syntax analysis, f.e. reviews entered instructions syntax.

Class `parse`

Class parse is the center of program as it controls the whole analysis, generates XML document and gathers statistic data for `STATP` extension.

2. Implementation

The parser is handled by `Exceptions`. Every class and their functions are called in `try` block and when error occurs the exception can be thrown within a `catch` block with proper error code and error message. First off all, arguments are checked by `arg_check` function. This function prints help message and end program, or just check validity of arguments set by user. Input from `STDIN` is loaded after the end of input into string. With `do_parse` function, within `parse` class, the string is divided using regular expression. The regular expression split input to an array, after every horizontal whitespace character, *"@"* and *"\n"*. The last two of those delimiters needed to be in result array so every appearance of them is not just divided, but stored, too.
The problem with this solution is with *'@'* characters in string data type where the entered string value must not be split with that delimiter. The problem correction is done by capturing another delimiter `'/string@+[^#\s]*/'` within regular expression. After that the `string@` elements in array are divided manually in loop, also multiple *"\n"* is deleted in this loop.
Lexical and Syntax analysis is done in loop, until the index reaches the last array element. Function `check_instruction`, within `checker` class, checks instruction syntax and returns number of parameters this instruction needs. In loop, switch is created based on number of parameters. Every instruction is handled and controlled by proper sequence of data types and arguments. The XML code is generated simultaneously with analysis, using built-in `xmlwriter` function. If no error appeared, at the end of `do_parse` function the XML document is printed, and function returns array with statistics for `STATP` extension. Parameter names for `STATP` are used as keys to array, for easier access. The extension statistics are written to file in function `write_stats`, within `argument` class. Those data are written in loop, so every extension parameter may be printed more than once.