Implementation documentation for the 1-st task of IPP 2021/2022
Name and Last Name: Andrei Shchapaniak
Login: xshcha00

1 Processing of the IPPcode22 source code

The source code is read line by line from standard input. Then the new line is removed from each line at the end and multiple spaces are replaced with a single space. The header of the source code is checked after it all. If it is not a line with a header, the line will be separated by space.

Lexical analysis is performed by PCRE regular expressions and syntax analysis is performed by expecting order of arguments for every instruction.

To generate an XML representation of the IPPcode22 source code the `DOMDocument` library is used.

2 Command line parameters parsing

It is implemented using function `getopt()`. In case of entered only one parameter `--help` will be written a help message. In case of more than 1 entered argument one of them must be `--stats=filename`. In case of wrong sequence of arguments will be return error `10`.

3 Implemented extensions

3.1 STATP

This extension adds to the script a possibility of gathering different statistics about the source code. To activate this extension parameter `--stats=filename` must be passed to the script. At the end of the analysis, the statistics are stored in the specified file in order according to other specified parameters.

Brief description of the meaning of the parameter. `--loc` count the number of rows with instructions, `--comments` count the number of comments, `--labels` count the number of defined unique labels, `--jumps` count the number of instructions for jumps, including instructions `CALL` and `RETURN`, `--backjumps` count the number of back jumps, `--badjumps` count the number of jumps on a non-existent label, `--fwjumps` count the number of forward jumps.

The structure `Stats` is responsible for all collected statistics. To count all jumps and labels are used functions `checkLabels()` and `sortJumps()`.

3.2 NVP

When implementing this script, the `Singleton` design pattern was used. I have used this design pattern because it is appropriate in my case when a class should have just a single instance available. `Singleton` design pattern helped to solve access global variables and easily access the sole instance of a class. The files `parse.php` and `stats.php` have classes `outXML` and `Stats` respectively. Each of these classes has a private constructor to prevent the direct creation of objects and a public static method to create an instance only if it was not already created.