Implementation documentation for the project in IPP 2017/2018

Name and surname: **Tomáš Zubrik**

Login: **xzubri00**

1. Lexical and syntactic analyzer of code IPPcode18 (parse.php)

1.1. Keywords

Keywords, in other words instruction operation codes in program are represented as keys in associative array named `$keywords`. Value of every key represents sequence of characters, that are abbreviations for obligatory arguments for particular instruction operation code. (`$keywords = {"add" => "vss"}`, where string `"vss"` represents variable and two symbols)

1.2. Regular expressions

Regular expressions are used to check correctness of lexical units and also the syntax. In appropriate way there are used functions `preg_match(), preg_split()`, `preg_explode()` and more.

1.3. XML Writer

There were more possibilities to choose from to parse XML output format of IPPcode18. I have chosen the XML Writer, which had made my work easier. I have implemented function that creates XML element representing the argument of instruction by using XML Writer functions (`add_xml_arg()`). This function was called in loop to creates argument element for every instruction in appropriate number.

1.4. Extensions

I have implemented extension STATS. Important change in program was in parsing input arguments, because STATS works with specific arguments. By adding global variables is was simple to get number of comments and lines of codes of IPPcode18 program.

2. Interpreter of XML representation of code IPPcode18 (interpret.py)

2.1. Parsing input arguments

For parsing input arguments of script there is used `OptionParser` from `optparse.` Without implementing extensions there was need only for parsing argument `--source` and its value. Argument `--help` is created and parsed by `OptionParser` automatically.

2.2. Lexical and syntactic analyzer

Script goes through whole XML representation and does lexical and syntactic analysis of XML elements, its attributes and its values. For analysis there are used regular expressions and dictionaries, that represent each instruction and its expected arguments' number and types.

2.3. Instruction list

After lexical and syntactic analysis, if it has passed, there is created instruction list containing information about every instruction and its arguments. Now is possible to loop over entire instruction list and determine next instruction and let the script control the flow of the program flow of IPPcode18 program. Every instruction in script has its own functionality and it is represented by specific function. For example `ins_jump(), ins_defvar(), ins_createframe()` and so on.

3. Testing script (test.php)

3.1. Parsing arguments

In appropriate way there was used for parsing input arguments the function `getopt`. There was need to deal with 5 input arguments and that was the reason for choosing the getopt. Parsing has become more easier and it takes less lines of code.

3.2. Creating files

For creating files that were missing, I have implemented the function that creates new empty file in specific directory `create_file()`. There are also used temporary files, that are deleted after the script execution. They are used for comparing the expected output files and output from interpreter by using `diff`.

3.3. Running tests

Firstly, very important function that is used is `shell_exec` that runs commands in command line. To find all tests in specific directories there was used `find`, and then executing parser.php script for creating XML representation and interpret.py script for interpreting the code itself. Return codes of both scripts were stored in variables and were checked with expected return codes and eventually with output file.

3.4. HTML output

For HTML output there was used `XML Writer` creating XML elements for every passed and failed test. Every directory that contains tests in appropriate format are in specific section and it is clear and easy to see which tests passed and which didn’t. There is also used little bit of CSS style to make it more readable and user-friendly.