Documentation of Project Implementation for IPP 2019/2020
Name and surname: Pavel Yadlouski
Login: xyadlo00

1 Basic information about script implementation

Main body of script is in the file `parse.php`. There are defined variables for inner needs and loop for reading input code from `STDIN`. This loop is represented as `while` loop with `switch case` inside.

First cases made for simple processing of header line (using function `checkHeader`), empty lines and comments on separated lines. In my opinion, this kind of implementation speed up process of lexical analyze in case, that insignificant lines don't use function for general cases, but parsed in as specific case as fast as it can be.

2 Lexical analyze of input code

In default case processing of operations from input code is held. Basically, there are some functions that carrying about code operations. First function them `checkArgsCount` checks number of arguments for current operation. For this purpose there is dictionary with all operations as a key in sting format and number of arguments as a value (simple integer value). Then follows adding new element to XML with method `addElement` of class `Writer` (extends `XMLWriter` class for for simplifying adding elements of operations)

After control of number of arguments creating of new element follows. Then there is loop for parsing argument of operations. There is another switch case with all valid commands. Cases groped by number of required parameters (for example command `(`ADD) and `SUB` requires 3 parameters) and by type of parameters (variable, label, etc.). For each group based on number and types of parameters called two functions for processing variables and symbols (`var_symb`) and labels and type (`label_type`).

I made this division because it seamed to me that processing this groups of arguments types pretty same and it would be comfortable to process them together. Division is made based on symbol '@' in current argument. If it presents, so it would be a variable or a constant, if not - it should be a type or a label.

This way of processing arguments insures that on each place there would be appropriate positional argument. For example for command `JUMPIFEQ` required parameters are:

1. `label` as first argument
2. `symbol` as second argument
3. `symbol` as thread argument

So, as first would be called function `label_type` for processing label, then two times function `var_symb` for processing symbols.

I made this division because it seamed to me that processing this groups of arguments types pretty same and it would be comfortable to process them together. Division is made based on symbol '@' in current argument. If it presents, so it would be a variable or a constant, if not - it should be a type or a label.

3 Extension – STATP

During parsing input code, if flag for collecting statistics is set, then established values are counted is the moment of writing down to XML representation, t.e in functions `var_const` and `label_type`.

As parameters for counting different statstics can be in any order an can be duplicated, so I process them in the following way. In the end of main program `parse.php` there is loop that go through every elements in built in array `$argv` (previously first element containing name of the script is removed). Inside this loop there are some `if` cases, that checks whether current argument is one of available parameter for collecting statistics or not, and write down to given file corresponding values