Implementační dokumentace k 1. úloze do IPP 2018/2019
Jméno a příjmení: Gabriel Quirschfeld
Login: xquirs00

1. Assignment

The goal of the project was to create a parser in PHP 7.3 that would check both the lexical and syntactic correctness of the input code. The input code is written in IPPcode19.

2. Solution

2.1. Input arguments

The input arguments are all handled in the function `handle_arguments()`. This function is called first right after the script is run. I used the `getopt()` function to store the arguments in a global variable that is later used in the `write_stats()` function. I handled the collisions (e.g. missing –stats when other arguments are present).

2.2. Lexical analysis

The lexical analysis runs in the `check_lines()` function. Here an array of lines from the input code is checked first if it contains the .IPPcode19 header (if not the script exits with the code 21). Then it filters out the comments by finding the # symbol and unsetting all the strings after it from the line in the array. The correct instruction name is checked as well.

2.3. Syntactic analysis

The syntactic analysis runs in the `check_lines()` and `check_data()` functions. First it checks whether each instruction has the correct amount of arguments. Then each argument is checked for correct spelling based on the IPPcode19 regulations. This also finds errors in the types of arguments used, meaning one cannot use a label instead of a variable.

2.4. XML generation

The XML document is created after the input is stored as lines in an array with the program element and its attributes generated as well. The `generate_xml($opcode, $element)` is then called every time a new instruction is checked by the lexical and syntactic analysis. The global `$order` variable is used as the order attribute, `$opcode` is used as the opcode attribute and to help with generating the arg elements. The `$element` variable is used for writing the type attribute and getting the values of the strings, ints and so on.

2.5. Addons

The data for the addons is gathered during the runtime of the script. Every time the lexical analysis finds a comment the `$comments` global variable is iterated. The `$jumps`, `$labels` and `$order` global variables work similarly. They are all then used in the `write_stats()` function where they are written into the file.