Implementační dokumentace k 1. úloze do IPP 2018/2019

Jméno a příjmení: Adam Abrahám

Login: xabrah04

1. Array implementation

Firstly, two multidimensional arrays are created, one for storing every allowed instruction and the second for storing every allowed program parameter. The variable which holds the instructions has the instruction name, also known as the opcode, as its key and another array as its value, which holds the instruction parameters. For example: `"add" => array("var", "symb", "symb").`

The second variable which holds the parameters has the program parameter as its key and an array as its value, which holds two items. The first item indicates If the given counter should be counted. The second item holds the counted value. For example: `"comments" => array(true, 3).`

2. Processing user arguments

The first operation the program does, is processing user arguments. A function is called, which compares user written arguments with allowed program arguments. If the comparison does not match the program is terminated with error code 10.

3. Input parsing

For input parsing the `stream_get_contents` function is called, which reads remainder of a stream into a string. Afterwards each line and each word is converted into arrays, using `explode`. Then all unnecessary whitespaces and comments are removed with regular expressions.

4. Instruction and argument processing

After input parsing the program compares the first line with the correct header format. In case of mismatch the program is terminated with error code 21. The program continues with instruction and parameter processing. Each line is processed through a loop. A function gets an array of words, converted from a single line, as a parameter. The first item in the array is the instruction and the rest are the instruction parameters. The program checks if the given instruction exists in the array which holds the allowed instructions, gets the number of parameters and compares it with the number of parameters on the line. After instruction check each parameter is processed. The program determines if the parameter is a variable, constant or a label and performs a specific check on each, which analyzes its correctness. On a lexical or syntactic error, the program is terminated with error code 23.

5. Result printing

The final XML document is created with the `XMLWriter` extension. The variable, which holds the XML document, is initialized at the program start. Each instruction and parameter is written to the document in the exact order. The final XML document is printed out at the end to the standard terminal output.