Documentation of Project Implementation for IPP 2021/2022

Name and surname: Vladyslav Kovalets

Login: xkoval21

Processing and analysis

The script starts with a function `handle_arguments()` that checks the arguments entered by the user. After that, a function `handle_stdin()` is called, which first takes data from stdin using a loop and removes unnecessary characters or lines. For example, comments, tabs, or a sequence of spaces. Next, it checks for the presence of a header `.IPPcode22` and, if successful, creates and sets up the `DOMDocument` class. I chose this method for XML format output. In an unsuccessful case, the script will send an error message. After all this, the edited string is divided into lexical tokens and passed to the next function `handle_instructions()` as an array. It is responsible for the distribution of instructions depending on the number and content of the arguments. The IPPcode22 language is case insensitive. Each letter of the token is converted to uppercase and compared with instructions using switch.

Each group of instructions has its own functions. For example `defvar_pops()` or `call_label_jump()`. They parse and populate the XML output with additional functions such as `is_variable()` checks the variable for validity, `is_label()` checks the label for validity, `is_type()` defines the data type. Functions `create_xml_functions()` and `create_xml_arguments()` create and set up elements for XML.

At the end of the script, the prepared XML output is printed.

Extension

The STATP extension allows statistics to be collected and stored in an output file.

I have implemented statistics for instructions, comments, labels and jumps. This is implemented using the `Counters` class. The instruction counter is incremented before dividing instructions into groups. The comment count is incremented by one after a comment is deleted. One is added to the jump counter if a `"RETURN"`, `"CALL"` or `"JUMP"` has been detected and processed. When `"LABEL"` is found, the script checks the found name in a special array in order to avoid repetitions and then writes it there, followed by incrementing the label counter.

The `print_stats()` function is responsible for writing statistics to the file specified by the user. This is implemented using a loop and the `stats_arr()` array, which ensures the correct sequence of statistics writing.