Implementation documentation for 1. task of IPP 2023/2024

First and last name: Nestor Baraniuk
Login: xbaran21

1 Script structure

The script consists of three parts: **lexical analysis**, **parsing** and **generating an XML document**. At the beginning of the script, two classes are created - **Token** and **Stats**. After that, regex patterns are defined in the form of macros and an IPPcode24 instruction dictionary is created.

2 Lexical analysis

Lexical analysis is defined by the `instruction_scan` function. This function takes a single line of code and turns it into an array of tokens. For each token, it determines its type and value, and checks for the presence of an operational code in the dictionary. If it is missing or the token type cannot be set, the script exits with error 23. Otherwise, the array of tokens is passed to the next function for parsing.

3 Parsing and generating

Parsing is defined by the `parse_instruction` function. It takes as an argument an array containing instructions and tokens and checks them against the corresponding instruction in the dictionary. Each operating code in the dictionary has its own pattern of argument types. In case of any error this function exits with an error code 23. During parsing, both the types and the number of arguments are checked. After that, the function calls `print_xml` to generate an instruction in XML format. It generates an instruction, its attributes, and elements. The main function will first check for the header, then call instruction_scan for each line which contains the code.

4 STATP

To collect statistics, I used the Stats class, which would contain all the necessary information according to the task specification. It consists of variables and dictionaries and is mostly filled during the `print_xml` function. When working with the most frequent instructions, where their number has not been specified, the script will print them until their number reaches 35% of the total number of instructions.