Documentation of Project Implementation for IPP 2023/2024

Name and surname: Aurel Strigáč
Login: xstrig00

Introduction

This script is a Python implementation of a parser for the IPPcode24 language. The primary function of this script is to read IPPcode24 instructions from stdin (standard input), perform lexical and synthetical analysis, and output the corresponding XML representation of the instructions to stdout (standard output). This script (parser.py) is designed to be user-friendly, offering helpful error messages for common syntax and lexical mistakes, and includes "`--help`" option for users in need of an assistance.

Implementation

This script is written in Python 3.10, utilizing standard libraries for XML handling (`xml.etree.ElementTree` and `xml.dom.minidom`), regular expressions (`re`), and system interactions (`sys`). Included here is the overview of the implementation, see the source code for more detailed insight.

Firstly, the script checks if the "`--help`" argument is present. If it is, the script prints out the help message, otherwise it removes any parts of code that are unnecessary for our purpose (spaces, comments,…). It then verifies the presence of the required header. Once the header has been verified, the program goes through each line of the input file. Each instruction is then put into the XML tree, which is then printed to the standard output. This happens only if the syntax of the instruction is correct. To check syntax, the program utilizes cases for each instruction that share a syntax share, grouped by the types of variables accepted by the instruction. The program also checks whether correct number of instruction arguments has been used. If any of these check come out as unsuccessful, the program prints an error message and exits with error code.

The script is organized into several key functions, each responsible for a different part of the parsing process:

`print_help()`: Displays a help message explaining how t use the script, its options, and exits the program.

Regular Expression Patterns: Defines patterns for validating the correct format of variables, labels and symbols in the IPPcode24 source code.

Syntax Checking Functions: A set of functions to check the syntactical correctness of different elements(e.g., variables, symbols, types, labels) within the code. It includes following functions:

`check_length(instruction, number, expected)`: Compares the expected number of arguments with the actual number according to IPPcode24 instructions.

`check_symb_syn(token), check_type_syn(token), check_var_syn(token)`: Checks if the given variable token is syntactically correct, performs different checks to ensure the variable is valid depending on type of the argument (symbol, type, variable). These functions facilitate the aforementioned regular expressions in order to perform these operations.

XML Handling Functions: Includes functions for initializing the XML document, inserting elements corresponding to IPPcode24 instructions, and generating the final XML output. It includes following functions:

`XML_initialize()`: Initializes the XML tree and sets the root element for IPPcode24.

`XML_write_arg(root, num, token)`: Inserts the XML representation of an argument into the XML tree.

`XML_body(line, xml_tree, root)`: Processes IPPcode24 instructions from the input lines, creating corresponding 'instruction' elements in the XML tree and validating syntax and arguments. This function uses the `XML_write_arg()` function to achieve this.

`XML_finish(xml_tree)`: Finalizes the XML document, prepares it for output, and prints it with proper formatting to the stdout (standard output).

Conclusion

In summary, this IPPcode24 parser, crafted in Python, delivers a foundational approach to parsing the IPPcode24 language. It uses a collection of utility functions for validating the language's syntax and outputs each instruction's XML representation to stdout (standard output).