Documentation of Project Implementation for IPP 2017/2018
Name and surname: Matej Kňazík
Login: xknazi00

1 Description

Project is based on set of scripts which are together providing interpretation of language IPPcode18. Set of scripts consists of three parts: *parse.php* which performs parsing source code to XML format, *interpret.py* which performs interpreting of source code formatted in XML and *test.php* which is script for testing proper functionality of previous two scripts and its output is provided in HTML format. For usage information of these scripts run them with option *--help*.

2 Implementation details

All three scripts were implemented with procedural approach, because object-oriented approach would be more time demanding and the project itself was relatively small. However, some simple classes were used in script *interpret.py* for various stacks, variables and frames with aim to make the code more readable and easier to debug.

2.1 parse.php

Script *parse.php* is using php DOM library for creating XML structured output. Basic principle of this script is to parse input line by line while ignoring comments and if input is semantically and lexically correct than put it in specified XML format. Lexical analysis is mostly done with regular expressions. Semantic analysis is performed only partially, the rest is expected to be performed by *interpret.py*. This script only checks if instructions have exact number of arguments and whenever they are valid symb or var as specified in IPPcode18 specification.

2.2 interpret.py

Next script *interpret.py* is the biggest part of this project compared to the other two scripts. It has a lot of lines of code which is caused by fact that every instruction of language IPPcode18 has its own function, that provides proper parsing. The reason of this solution is to create more readable code.

Interpret.py expects file with formatted XML structure which is parsed by python's *xml.etree.ElementTree* module. If parsing of XML file went successfully, then it's proper structure is checked. After that, script goes through all instructions and tries to load all labels in python's dictionary containing its names and instruction numbers. Then finally performs pre-runtime semantic, lexical and syntactical analysis. This script doesn't rely on code analysis done in *parse.php,* therefore analyses same things as previous script. In case that everything went successfully, script start performing interpretation of program. Simple classes are used for representing variable, frames, call-stack, data-stack and stack-frame.

2.3 test.php

Last script *test.php* similarly as *parse.php* uses DOM library, but this time for creating HTML representation of output. For simplicity, HTML template is at first represented as string, which is later parsed with function *loadHTML().* This also allows to add some CSS styles for table with test results more visually appealing. Test firstly loads all *.src files and then assign them their *.in, *.rc and *.out files. Then runs test and results saves as *My.out files and output of interpretation redirects from stderr to *.err files. Execution of this tests are done with php function *exec().* Reference output and actual output are compared with diff command. If return codes are identical and so are outputs than test is displayed in result table in green color with status OK otherwise status is FALSE and color of test is red. Lastly, script provides some short summary of tests results under the table.

3 Extensions

3.1 parse.php extensions

parse.php supports STATP extension.

3.2 interpret.py extensions

interpret.py supports STATI extension.

3.3 test,php extensions

test.php doesn’t support any extensions.