Implementation Documentation for 2. Task to IPP 2018/2019

Name and surname: Ondrej Šajdík
Login: xsajdi01

interpret.py

Program

Script interpret.py interprets instructions loaded in XML format. Center of program is `Interpret` object calling `InputHandler` to receive Instruction and parsing it to `Operation` class to receive method representing instruction functionality and runs it.

Error handling

For handling errors in any part of program is implemented class `ErrorHandler` with static `error()` method which is called in case of any error to print error message on stderr and exit program with specific error code.

Program arguments

For handling program arguments is implemented `ProgramArgumentsHandler` class with method `handle(argv)`, which handles arguments and returns `InterpretSettings` object which is used to adjust interpret behavior.

Interpret

Main body of program works as state automat. Holds information about state of interpretation and runs operations changing this state.

Input

For reading source code is implemented `InputHandler` which calls xml python module to convert xml file into tree structure and then creates list of instructions. Contains `get_instruction(index)` method returning syntactically correct instruction on index from list.

Instruction functionality

To store functionality of all instructions was created `Operation` class with extra function `get(opcode)` returning method from dictionary. For each opcode is implemented method with its functionality.

Syntactical analysis

Class `Analyser` contains static methods to perform syntactical check on each instruction and on instruction list as whole. Analyser is called by `InputHandler`.

Lexical analysis

Lexical checks are called from methods, which represents instruction functionality.

Extensions

Solution include all three extensions. FLOAT and STACK are just extending IPPcode19 so they are implemented alongside with IPPcode19. For STATS is used `stats` object with counters and method `save()` to print results to file.

test.php

Program

Script test.php loads input and expected output of test cases from dir. Generate missing files and runs test cases to test parse.php and interpret.py scripts. Test results are printed on stdout in html format.

Error handling

In case of missing tested program or inserting invalid directory is printed error message on stderr and program is stopped with error code.

Program arguments handling

Implemented function `handleProgramArguments()`, which works as state automat and returns `TestInfo` object saved to global variable adjusting program behavior. In case of –help argument is printed help message and exits program.

Test case search

Implemented function `getTestList()` returning array with relative paths to files with .src suffix. In case of –recursive argument is called `recursiveSeach($dir)`, recursive function searching in all sub directories. Then is checked if all files for each test case exist and if not they are generated.

Test execution

For executing list of test files is used executeTests($testList) function. Function runs each test with its input values and then compares exit code and output file with expected. All test results are saved as `TestResult` returned in array of TestResults.

Output

Output in HTML format is printed on stdout using `printResults($results)` function. Test results are formatted into table with information: 1. Test name with its relative path, 2. Test result: passed/failed 3. Reason of failure. In output is also displayed total number of passed and failed tests.