Documentation of Project Implementation for IPP 2017/2018

Name and surname: Ivan Eštvan

Login: xestva00

Script parse.php:

Script implements a parser which loads source code from standard input (`STDIN`), and proceeds with syntax analysis and lexical analysis following the rules of `IPPcode18` language.

Input:

Input is parsed by finite-state machine line after line. Each line is split into characters and parsed. Parsing is split into 6 stages (0 to 5) each representing different part of the process. Stage 0 to 3 are used to create instruction, arg1, arg2 and arg3, stage 4 ignores comments and stage 5 represents the end of possible instruction. If there is more characters once in stage 5, then it results in an error.

Main functions:

- `analyze (instruction, arg1, arg2, arg3)` – checks whether instruction matches operation codes of IPPcode18 language, instruction is a result of input parsing. Does this by matching the `instruction` to a reference list of IPPcode18's instructions.
- `analyzeArgs (case, arg1, arg2, arg3)` – uses custom functions to check validity of all arguments for operation code groups defined by the case, there are 8 case groups based of a IPPcode18 specifications

Custom functions:

- `issymbol(arg), iscontant(arg), istype(arg), islabel(arg), isvar(arg)` – in all cases `arg` is split into array of characters which are parsed and checked for validity

Output:

Script outputs valid XML representation of the source code to the standard output (`STDOUT`), to achieve this is uses library `Array2XML` for easier creation of the XML elements.

Error codes:

- `21` – lexical or syntactical error was found in `STDIN`
- `10` – wrong script arguments

Script interpret.py:

Script implements an interpreter which reads a source file and produces a result by using standard input and output. In this case I don't use any special library only the ones available on school server.

Input:

Since source file is expected to be in XML format file, it is parsed by a xml.etree.ElementTree library, then checked for any lexical or syntactical errors. Result of this parsing is a list of instructions, each of class Instruction which match IPPcode18 specification.

Classes:

- `class Instruction` – contains variables `order, opcode, child_count` and 3 pairs of `argN_type, argN_value`, where N is argument position from 1 to 3. On its creation all variables are filled with data provided by the initial parsing.
- `class IPPvariable` – contains variables `type` and `value`, where `type` is IPPcode18's variable type and `value` is IPPcode18's variable value. It is used to store IPPcode18's variables

Main functions:

- `analyze_instruction_args (instruction_to_check, case)` – checks whether instruction matches operation codes of IPPcode18 language, instruction is a result of input parsing. Process is very similar to the one in parse.php.
- `parse_string (string)` – parses the `string` which may contain Unicode values, replaces these with characters representing the Unicode value and returns the `string`
- `main()` – is not only the entry point into `interpret.py`, but also contains an entire interpretation itself. This was not the bright move, but once I realized it is too big and not easy to understand, it was already too late for a change.

Custom functions:

- `custom_isstring(arg_value), custom_isbool(arg_value), custom_isint(arg_value), istype(arg_value), isvar_or_label(arg_value)` – in all cases `arg_value` is split into array of characters which are parsed and checked for validity

Output:

Script outputs the result of the interpreted IPPcode18.

Error codes:

- `31- not valid source file - error in XML formating`
- `32` – lexical or syntactical error was found in `XML file`
- `52-58` – problem during interpretation
- 10 – wrong script arguments

Script test.php:

Script implements automated testing of scripts parse.php and interpret.py. Doesn't use any special frameworks.

Output:

Output of this script is in HTML format. Contains a table with information on which tests passed or failed, their exit codes and the overall number of passed/ failed tests. If forwarded into file, it can be open in any web browser.