Implementační dokumentace k 2. úloze do IPP 2018/2019

Jméno a příjmení: Adam Abrahám

Login: xabrah04

# 1. Interpret.py

## a) Class and array implementation

At the beginning two classes are implemented, one for frames and one for stack of frames, then a multidimensional array is created, for storing every allowed instruction and their arguments with allowed data types. The array which holds the instructions has the instruction name, also known as the opcode, as its key and another array as its value, which holds the instruction parameters and data types.

For example: `"add": {"var": [], "symb1": ["int"], "symb2": ["int"]}`.

The `Frames` class has two attributes, attribute `init` shows whether the frame is initialized, and attribute `variables` is an array which holds frame variables. This class implements methods that work with frames, such as `append` and `clearFrame`. The `Stack` class has one stack attribute and implements methods that work with the stack. For example: `push, pop, top`.

## b) User argument processing and source parsing

The first operation the program does, is processing user arguments through `argparser` module.
For source parsing the `ElementTree` module is used, which parses the input XML into a variable. Firstly, the parsed XML goes through syntax check, afterwards each instruction and its arguments are converted into arrays.

## c) Instruction and argument processing

After input parsing the stack and the global frame is initialized. The program continues with instruction preparation, all instruction and arguments get sorted and processed through a loop. A function gets the instruction name, and its sorted arguments from the parsed XML, as parameters. The program checks whether the given instruction exists in the array which holds the allowed instructions, gets the number of parameters and compares them with the number of parameters in the given array. After instruction check each parameter is processed. The program determines whether the parameter is a variable, constant or a label and performs syntax and semantic check on each. On success, each implemented instruction is executed in the correct order.

# 2. Test.php

## a) User argument processing

The program starts with user argument processing. Arguments are processed through a built-in function called `getopt`. In case of invalid argument combination, such as `–int-script` combined with `--parse-only` or vice versa, the program is terminated.

## b) Directory search and test execution

The program looks for each file with `.src, .in, .out, .rc` extension in the given directory or directories. Each `.src` file is saved into an array with its real path, and each `.in, .out, .rc` file is created when not found.
After the directory search, test function is executed for each `.src` file. The test function compares the script file content and exit code with the estimated file content and exit code, based on user arguments, and saves the result.

### c) HTML output

At the program start a variable is initialized, which holds the HTML code, a table with test results and a summary of all tests. Each test is stored in this table, which contains the test number, its name and the result. At the end this HTML code is printed out to the standard output.