Implementační dokumentace k 19.04.2022 úloze do IPP 2021/2022

Jméno a příjmení: Artur Suvorkin

Login: xsuvor00

1.1 Foundation (interpret.py)

Basis of the program is a variable **`instructions`** of type **`dictionary`**, in which information regarding existing IPPcode22 language instructions is stored. The key values in this field are **`instructions`** names. Value stored for each key is an array of arguments that function can take, as well as a pointer to the function that will be called when code is interpreting.

1.2 Loading and checking XML input (interpret.py)

Loading and input control is performed in the function **`checkXML`**. For load **`XML`** data library is used **`xml.etree.ElementTree`**. Function returns structure in which operation codes and instruction arguments are stored.

1.3 Code interpretation (interpret.py)

Frames that store program variables are represented as dictionaries that can be searched by variable name. Frames (TF | LF | GF) are stored in a single field for ease of access.

Helping Functions is created for each instruction that exists in the IPPcode22 language, which is called during interpretation. Links to these functions are stored in the variable **`instructions`** and inputs of these functions are arguments for individual instructions.

Jump instructions change global transformation called **`currentInstruction`** which controls the flow of the program. It is set according to the signal to be jumped to. Labels are stored in the dictionary. Each label is assigned a value that determines where the label is located in the program being executed. The label dictionary is already created during XML input processing.

2. test.php

The script starts by creating the root DOM element. We also introduce global variables including test counters.

From the very beginning, we create an empty HTML page and add styles to it.
Thus, when the script is run without parameters, if it does not find a folder with tests, an empty HTML page will still be generated.

In order to reduce the possible number of errors, reverse command line arguments done with PHP getopt function. This, however, leads to the fact that the program does not react in any way to unfamiliar arguments.

A temporary directory is created for the tests to work, which will be deleted at the end if the **`--noclean`** parameter is missing.

Main test function mainTest. It runs parse.php and interpret.py.
In order to improve the readability of the code, mainTest uses the **`getFolders`** helper functions to get the names of the files and directories that the script works with.

Test outputs are compared by the **`compareOutputs`** function. It also raises the counters of passed and failed tests.

The result table is generated with **`addTableRow`**.
This function receives as parameters the path to the test being performed, information if it passed (this is necessary for a more visual representation of the results)
as well as a possible reason why the test failed (test and reference rc code or information that the diff result is different).

At the end of the file, a line with general test statistics is added and the root element is saved.

```
Usage: ... >out.html
```