VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Documentation of Project Implementation for IPP 2021/2022

Name and surname: Vladyslav Kovalets

Login: xkoval21

Interpreting input XML code IPPcode22

The main file is `interpret.py`, it interprets the input source text. For convenience, minor files are placed in the `lib` folder. The `init.py` file initializes most data types, `functions.py` stores all functions, `class.py` is created for all injected classes, and `errors.py` contains all sorts of errors.

Solving the problem of interpreting xml data, thanks to object-oriented programming, several classes were implemented for instructions, arguments, variables and the stack.

An example of the completed Instruction class, which is in the instruction list at number three:

```
instruction[3].opcode    = "MOVE"

instruction[3].arg1.type = "var"

instruction[3].arg2.type = "int"

instruction[3].arg1.val  = "GF@var2"

instruction[3].arg2.val  = 2
```

The class is incrementally populated and used in `handle_structure` and `handle_instructions`.

The first operation that the program starts is processing each command line options using the additional `getopt` library. The source file, if present, is used in a function that parses and logically validates the XML data using the additional `xml.etree.ElementTree` library.

Each instruction is stored in the `functions` dictionary. The key is the queue number, and the value is the `Instruction` object. To ensure that instructions are processed one by one, their numbers are stored in the `orders_instructions_list` list. Each label is also stored in a dictionary and has a similar key with a number.

Each element in the sorted list of instructions is processed in the `handle_instructions` function. The loop goes through each instruction until the list ends. The values are stored in a dictionary where the key is `GF`, `LF` or `TF` frame.

Extension

STATI extension allows collecting code interpretation statistics.

The extension is not fully implemented, but only a report on the number of instructions executed `--insts` and listing the maximum number of initialized variables `--vars`. When processing the command line, the file for displaying statistics is stored in a variable `stats_file` and the required parameters to the `stats_parameters` list, thanks to which the sequence of statistics recording is preserved. If the user entered statistics parameters, but didn’t specify a file for recording, then an error will be displayed. The `stats_writes` function writes statistical data to a file and is executed before program termination, if needed. Also, in the `"EXIT"` instruction similarly.