Implementation documentation for the project to IPP 2017/2018
Name and surname: **Vladimír Marcin**
Login: **xmarci10**

1 Code Analyzer (parse.php)

1.1 Processing Arguments

We used `getopt` to detect options and their values, if any. In this step we check entered options for various errors. If some error is occured, script returns error code with value 10 and prints error message to `stdout`.

1.2 Syntax and Lexical Analysis

After processing the arguments we continue with syntax and lexical analysis. We get the line by line the using builtin function `fgets()`. All empty lines and lines, which contains only comment are skipped. If line contains something else, we split it by white characters and all parts of that we save to array. So, in first element of the array should be save name of instruction and in other elements arguments of the instruction. After that our script is looking for instruction in another array where are saved all instructions together with important informations about the instruction. This array has as key name of instruction and as value has another array with function names, that check arguments of given instruction using regular expressions. For every argument competent function is called and if argument is not right, the function reports error. This process is executed for every line in input file.

1.3 Extensions

We decided implements extension STATP. Extenstion is implemented using regular expressions. Every line of input file is matched with regular expressions, that checks the presence of comment or empty line or line of code. If it is a comment internal counter of comments is incremented and if it is a line of code internal instruction's counter is incremented.

2 Interpret of XML code representation (interpret.py)

2.1 Processing Arguments

In this case we used module `argparse` to detect options. When checking the arguments we had to establish the order of arguments, whether the extension STATI was implemented.

2.2 Syntax and Lexical Analysis

Syntax and lexical analysis are almost the same as in the parse.php. Difference is, that interpret.py instead of array uses dictionary for saving information about instrucitons. Interpret also checks correctness of XML file in contrast with parser. For parsing XML file module `xml.etree.ElementTree` was used.

2.3 Semantic Analysis

Semantic analysis takes place during the course of the interpret. Interpret checks and report all semantic error like bad opperands types, wrong operations with variables or labels an so on.

2.4 Interpretation

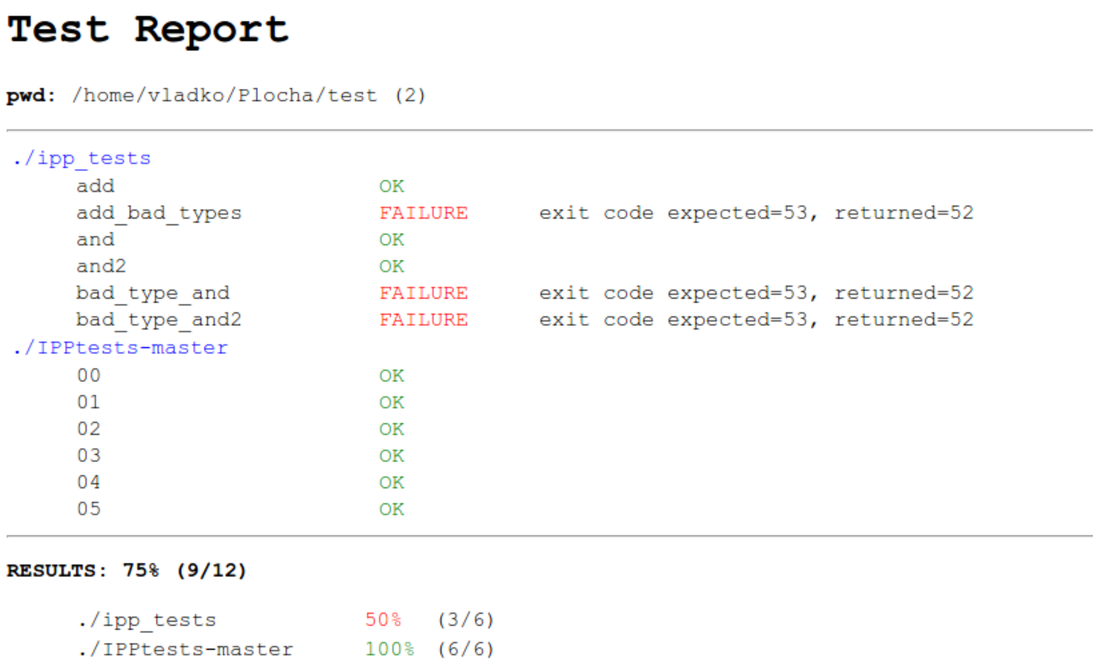
Every instruction in our code id represented as function. In this function is defined what will do given instruction. Each of these functions take as parameter, list of the instruction arguments. Instruction argument is represented as dictionary, which contains type and value of given argument. This simplifies semantic controls.

2.5 Extensions

We decided implements extension STATI. It is implemented in one function, which is called after every instruction and count number of executed instructions and number of initialized variabled in all frames.

3 Test (test.php)

At the beginig scrip finds all test in given directory and if recursive flag is set it will looking for tests in all subdirectories. All tests are saved in array, where as key is used directory and as value is another array which contains all tests in given directory. This will help us work with individual directories. Then the individual directories are processed. Script computes statistics for every directory and put it to the output html file. Reference html file is shown below.



```
Test Report

pwd: /home/vladko/Plocha/test (2)

./ipp_tests
     add                     OK
     add_bad_types           FAILURE     exit code expected=53, returned=52
     and                     OK
     and2                    OK
     bad_type_and            FAILURE     exit code expected=53, returned=52
     bad_type_and2           FAILURE     exit code expected=53, returned=52
./IPPtests-master
     00                      OK
     01                      OK
     02                      OK
     03                      OK
     04                      OK
     05                      OK

RESULTS: 75% (9/12)

     ./ipp_tests            50%   (3/6)
     ./IPPtests-master      100%  (6/6)
```

Figure 1: HTML output from test.php