Documentation of project implementation for 2. task IPP 2018/2019
Name and surname: Dominik Vagala
Login: xvagal00

Interpret.py

The goal was to create script, that will interpret xml representation of IPPcode19 source code.

Solution

Main script **interpret.py** uses multiple classes from separate scripts. Syntax analysis is not implemented by any strict scheme, because of relatively simple syntax in source language IPPcode19. All syntax rules for instructions are stored in list of Instruction object in this form: STRI2INT <var> <symb> <symb>. The main script provides following functionality:

1. Parse program arguments with modified Argparse, so if error occurred while parsing arguments script can exit with proper exit code.
2. Read xml source code and store it to list of lines
3. Check xml source code syntax
4. Get list of Instruction objects from xml source code
5. Do lexical analysis on list of instructions
6. Do syntax analysis on list of instructions
7. Do semantics analysis on list of instructions - check if there is redefinition of label, or not define label
8. Replace escape sequences in string types in source code to normal symbols.
9. Interpret code

Code interpretation

Whole code interpretation do **interpretCode(instructionsList, inputLines)** function. It takes list of Instruction objects and list of input lines from user. If user not specified input file, function **readInput(inputLines)** ensures that STDIN will be used instead. Interpretation go on line by line in loop by incrementing **currentInstructionIndex** and do interpretation on **instructionsList[currentInstructionIndex]**. Function **getLablesIndexes(instructionsList)** return dictionary with labels names and it's index in list of instructions. So when interpret should perform jump to label, it just finds label name in dictionary with corresponding index and change **currentInstructionIndex** to that. For generalization purposes it always store argument data from given instruction to **destinationData**, **sourceDataFirst** and **sourceDataSecond** even before it knows what instruction it should perform. That's because a lot of instructions has this format: INSTRUCTION dest [source] [source].

Classes

Argument
Represents argument in IPPcode19. Attributes:
- type: Type of argument. E.g.: symb, var, string
- name: Name of argument
- order: Position of instruction in instructions

Instruction
Represents instruction in IPPcode19. Attributes:
- name: Name of instruction
- arguments: List of Argument objects
- order: Position of argument in instruction arguments

Error
Represents one error type according to project specification. Attributes:
- description: Friendly description of error type that will be printed to stderr
- code: Exit code

Test.php

The goal was to create script, that will automate test process of parse.php and interpret.py and print test results in html code to STDOUT.

Solution

Script sequentially test source files from directory that user specified. These files are accessed by DirectoryIterator or RecursiveDirectoryIterator. Steps:

1. Parse program arguments
2. Print html starting code
3. For each test in source files do following:
   (a) Print test name to html table
   (b) Create default rc, out and in files if not exists
   (c) Execute parser or interpret or both scripts with source file and store results
   (d) Check if result are the same as expected
   (e) Print results to html table
   (f) Delete temporary files that has been created during test
4. Print html summary ending code

Classes

HtmlGenerator
Print various html code parts to STDOUT