Dokumentace úlohy CST: C Stats v pythonu 3 do IPP 2014/2015
Jméno a příjmení: Jiří Zahrandík
Login: xzahra22

1. Basic description

This script serves as analyzer of C language header and source files.

2. Controls

To control this script, several parameters can be used. Each of this parameters toggles special functionality of script. To count single operators, use `-o`. To count keywords, use `-k`. To count commented characters, use `-c`. The last listed parameter will count commented characters even in preprocessor's macros and directives. To count used identificators, use `-i`. To truncate absolute paths from output, use `-p`. By default, script analyzes even files from every subdirectory of specified directory. To turn this feature off, parameter `--nosubdir` must be used. Also, this script provides fulltext search, which is being turned on by this syntax: `-w=patternToFind`.

3. Implementation

For implementation, the object oriented design has been chosen. Mainly for it's high level of abstraction, readability and easy maintenance. Each component is represented by its own method. The `main` has only 1 line. It is where the object is created from class `parser`. After creation, methods are being called in constructor. First of all, argument check is called. Then, scripts checks whether input is a directory or a file. In case of file, file will be open and analyzed no matter what suffix it has. In case of directory, script goes through subdirectories recursively, and appends every C source or header file to array of files, which are going to be analyzed later. After this operation, rights to read from input files and right to write to output file must be checked. If one of rights is missing, error is raised and script exits with appropriate exit code. Then, based on argument, a method is called. Most of methods are using finite state machines to achieve certain target, e.g. count commented characters, strip comments, macros and string. As final part of constructor, method which prints the output is called. Implementation of each method is described in next paragraph.

3.1 Methods

`error(self, message, exitCode)`: basic output of error messages and exit codes

`help(self)`: prints help, if the reader is fan of Monty Python, the reader will like it.

`argCheck(self)`: this method uses regular expressions to match parameters passed to script. Whenever t he argument is matched, it is deleted from argument vector. This serves as control that parameters passed to script are correct. First, it checks for `--help`, then for `–nosubdir`. Next is parameter `--input` and parameter `--output`. In next step , method checks parameter `-p`. At this moment, only flag signaling what operation to do should be in argument vector. Therefore size of argument vector should be 2. If not, error is raised.

`scanFileOrDir(self, directory)`: this is recursive method, that searches through directory and subdirectories. Firstly, it checks if passed directory is file. If it is file and it has `.c` or `.h` suffix. It is appended to the end of file array. If it is any other file, method returns nothing and ends. Therefore emerges from one step of recursion. Then method iterates through directory and if finds directory, recursively submerges into this directory and search continues. If it finds a C source or header file, it is appended to the end of file vector.

`rights(self, array, mode)`: method iterates through array of passed files and depending on mode passed, checks if script can read or write into file(s)

`commentedChars(self, string)`: this method counts commented chars in whole header/source file including preprocessor macros. It is implemented as finite state machine

`stripStrings(self, string)`: this method strips strings from source/header file. FSM as well.

`stripComments (self, string)`: this method strips comments from source/header file. FSM as well.

`stripMacros(self, string)`: same as stripComments() but for macros

`stripKeyWords(self, string)`: same as stripComments() but for key words

`cutPath(self)`: cuts path and leaves only file name `printFiles(self, fileToWrite, filesArray, countArray, totalCount)`: this method prints analyzed files, count and total count in files into specified output file. Between longest file and longest number is 1 space.

`xFlagSet(self)` (where x equals flag set in parameters): this method iterates through files, if needed, performs regular expression match and sets object variables to counts of things specified in parameters.