Task documentation CHA: C Header in Python 3 IPP 2014/2015
Name and surname: Dávid Mikuš
Login: xmikus15

1 Task

The task was to create script in Python 3 to analyze functions in C language. Subsequently print out path to file, function name, return type and parameters in XML format.

2 Design

2.1 Argument parsing

Argument parsing is done by built-in class `ArgParser`. We have to inherit it and override exit method to return error code `1` instead of `2`. Next one we have to inherit `argparse.Action` for checking if the same argument wasn't entered more than once. At end of argument parsing, it also checks if `--help` was specified with no other argument.

2.2 Analyzing source code

Structure of source code is complex, we have to cut off everything except function headers. We used regular expression to remove macros and afterwards with finite state machine we ignored everything(comments,strings,body of functions ...) except functions headers which we keep them unchanged.

3 Implementation

The code is divided into 4 files.

- `cha.py` - Main file for running the script and printing XML output
- `HeaderParser.py` - Includes class for parsing source code
- `Function.py` - Includes main class for parsing function headers
- `Errors.py` - Contains error codes

3.1 Main script - cha.py

The script starts in `cha.py`. At start parse arguments with `ArgParser`, if some error occured, script is terminated and returns error code `1`. Otherwise every specified file is analyzed. If in `--input` was directory specified, then it recursively traverse through folders and analyze every file with `.h` extension. Finally, print out in XML format information about functions:

- File - In which file was function found
- Name
- Return type
- Varargs - `yes` if has variable arguments, otherwise `no`
- Parameters
  - Param number
  - Type

3.2 Source code parser - HeaderParser.py

At init, it opens specifield file and content is stored into variable. Every sequence of `/\n` is removed to be able remove macros with regular expression. Then the data are parsed with finite state machine. It removes everything except function headers such as strings, comments, global variables, typedefs. Afterwards every function header is parsed by `class Function`. Then it is checked if function satifies argument requirements

- No inline keyword
- Max number of parameters

If the argument `--no-duplicates` is specified and there are more functions with the same name(declaration and definition) only the first one is stored into final list.
If the argument `--remove-whitespace` is specified, it removes every whitespace in return type and parameters of function, but to preserve their context.

3.3 Function parser - Function.py

As a construct arugument it gets string with function header, then it is parsed into name, return type, parameters type and if function got variable number of parameters.

At first lexixal analysis is done by FSM and tokens are stored into list.
Next step is finding function name and index in tokens list. After is applied Right-Left rule. Parser goes through token list, when token in list is used, his value is replaced by char `$` to indicate he was already used.
When parameter list is found, it stores every type into list of parameters. Remaining unused tokens are joined to return type of function.
At the end if is specified, `class Function` can replace every whitespace char with space char and then removes every space, but to preserve their context.