Dokumentace úlohy CHA: C header analysis v Python3 do IPP 2015/2016
Jméno a příjmení: Michael Halinár
Login: Hhalin01

Návrh:

Fáze návrhu probíhala současně s programováním. Začal jsem s jednotlivými moduly a samostatně je testoval. Nejdříve bylo nutné zpracování parametrů skriptu, pak nastavení vstupu a výstupu a s tím spojené ošetřování práv a existencí souborů. Po úspěšném otevření vstupu a výstupu následuje ořezání vstupního souboru pomocí 7 regulárních výrazů. Následně jsou ze souboru vybrány všechny deklarace pomocí dalšího regulárního výrazu. Z těchto deklarací pak skript vybírá návratové typy, jména a argumenty, ty jsou následně vypisovány do XML formátu. Pokud byli zadány nějaké jiné přepínače, hlavní funkce `printStuff()` se o ne postará.

Postup řešení:

První fáze je zpracování parametrů programu, o tuto část se stará funkce `checkParams()`, která v cyklu podle počtu parametrů zjišťuje názvy parametrů a jejich správnost, zdali u přepínačů –max-par a –pretty-xml byli zadány čísla. Funkce si pamatuje jestli daný parametr už byl jednou nastaven, tímto se ošetřuje duplicita přepínačů. Hodnoty těchto přepínačů se ukládají do slovníku.

Po zpracování parametrů následuje nastavení výstupu funkcí `setStdout()`. Nejdříve se kontroluje zdali je výstup nastaven do adresáře, pokud ano skript končí chybou. Pokud je jako výstup zadán soubor, skript pak ověřuje práva na zápis, pokud je má tak soubor bez milosti přepíše. Pokud soubor neexistuje a skript nemá práva na zápis – chyba, jinak se soubor vytvoří a přesměřuje se do něj `stdout`.

Další funkcí v pořadí je `checkInput()`, tato funkce kontroluje správnost vstupu skriptu. Nejdříve se kontroluje existence souboru nebo adresáře pak práva na přístup.

Po úspěšné kontrole všech argumentů skriptu následuje nastavení odsazení výsledného XML souboru pomocí funkce `setPretty()`, která vrací dvě hodnoty a to `pretty` a `newline`, kde `pretty` značí počet odsazovaných mezer a `newline` značí nový řádek.

Následuje hlavní funkce `printstuff()`, tato funkce používá mnoho dalších modulů. První fází této funkce je vypsání hlavičky a zjištění zdali vstup je soubor nebo adresář. Pokud je to soubor , zavolá se funkce `openFile` která ho otevře a načte jeho obsah. Následuje už zmíněné ořezání souboru pomocí funkce `trimmTheFile()`. Tato funkce používá 7 regulárních výrazů – ořezání komentářů "`//`" které pokračují s "`/`"na dalším řádku, ořezání jednořádkových komentářů, pak víceřádkové komentáře, stringy v `""` a stringy v `''`, pak define které pokračují na následujícím řádku a zbylé jednořádkové define.

Po odstraněrní přebytečných znaků je volána funkce `getFunctions()` která pomocí jednoho regulárního výrazu vyhledá všechny deklarace funkcí a vrátí je v poli. Pokud byl zadán přepínač `--remove-whitespace`, funkce `removeWhitespaces()` se postará o odstranění přebytečných bílých znaků pomocí regulárního výrazu ,který najde všechny posloupnosti bílých znaků a nahradí je jednou mezerou.

O deklarace s proměnným počtem parametrů se stará funke `getVarArgs()` která vrací pole s hodnotami 'no' nebo 'yes'. Další používanou funkcí je `getArgs()` ta získává argumenty z jednotlivých deklarací a vrací je v dvou rozměrném poli – pro každou deklaraci jedno pole argumentů. Tato funkce vyhledá obsah závorek, ten si následně rozdělí na pole v místech čárek, jednotlivé části tohoto pole pak rozdělí na tokeny kde poslední token je název argumentu a předcházející tokeny jsou návratové typy, tyto uloží do zmíněného dvourozměrného pole.

Dvě podobné funkce `getNames()` a get `retTypes()` fungují podobně jako získávání argumentů, z deklarace se odstraní argumenty a zbytek se rozdělí na tokeny, kde poslední token je název funkce a předcházející tokeny jsou návratové typy.

O výpis se stará funkce `printOut()` ,která dostává jako argumenty předem získané názvy, návratové typy, argumenty atd. Nachází se v ní kontrola inline specifikátorů, kontrola duplicity a kontrola počtu parametrů kvůli přepínači `--max-par` .

Pokud byl zadán skriptu na vstup soubor, použije se vstavěná funkce `os.walk()` , tato funkce prohledá všechny soubory v daném adresáři a jeho podadresářích následně pak můžeme pracovat s polem souborů. Každý soubor s příponou `.h` se pak zpracovává již popsaným postupem.