Dokumentace úlohy CST: C Stats v PHP 5 v PHP5 do IPP 2013/2014

Jméno a příjmení: Michal Melichar

Login: xmelic17

Zadání:

Cílem projektu bylo vytvořit skript pro analýzu zdrojových souborů v jazyce C dle standardu ISO C99. Skript vypíše v přesně stanoveném formátu statistiky komentářů, klíčových slov, operátorů a řetězců.

Práce s daty:

Skript podporuje načítání dat pouze ze souboru. Vstupní soubor lze zadat pomocí parametru `--input=fileordir`. Přičemž za `fileordir` lze zadat buď samotný soubor, nebo adresář obsahující soubory. Pokud nebude tento parametr zadán, skript analyzuje všechny soubory v aktuálním adresáři. Pokud se nepovede otevřít zadaný soubor, program vrací hodnotu 2 a vypisuje chybové hlášení na standartní chybový výstup (`STDERR`).

Výstup programu je vypsán na standartní výstup (`STDOUT`). Toto chování lze ovlivnit parametrem `--output=filename`. Kdy za parametrem `filename` je nutné zadat požadovaný název výstupního souboru. Pokud se nepovede otevřít výstupní souboru, program vrací hodnotu 3 a vypisuje chybové hlášení na standartní chybový výstup (`STDERR`).

Struktura skriptu:

Tělo skriptu je abstraktně rozděleno na dvě části. V první části jsou ošetřeny správné formáty zadaných parametrů, výpis nápovědy, zpracování zadaných parametrů a volají se obslužné funkce implementované v druhé části těla skriptu. Druhá část těla skriptu obsahuje jednotlivé implementace pomocných funkcí. Netriviální funkce jsou podrobněji rozepsané v kapitole Popis řešené úlohy.

Popis řešené úlohy:

Zpracování argumentů:

Zpracování argumentů probíhá na začátku programu pomocí cyklu `foreach()`, regulárních výrazů a program využívá povolené pomocné funkce jazyka PHP. Pokud při zpracování nastane problém (např. neoprávněná kombinace parametrů nebo špatně zadané názvy souborů), je vrácena odpovídající chybová hodnota. Dále jsou použity proměnné, které se nastavuji na `TRUE` nebo `FALSE` podle stavu jaký nastane při zpracování argumentů. Tyto proměnné jsou využity pro správnou identifikaci vzniklých chyb, případně ovlivňují pozdější chování programu.

Načítání vstupního souboru a výstupního souboru:

Otevření zadaného vstupního souboru / adresáře probíhá až po samotném zpracování všech parametrů programu. Následně se rozlišuje, zda byl zadán soubor nebo adresář, obě možnosti se od sebe liší metodou implementace. Pokud byl zadán soubor, je volána pomocná funkce `open_file()`, která očekává jako parametr zadaný vstupní soubor. Byl-li zadán adresář, je zavolána pomocná funkce `obsah_adresare()`, která očekává jako parametr zadaný vstupní adresář a proměnou určující zda byl zadán parametr `--nosubdir`.

V případě zpracování výstupního souboru je chování odlišné. Nejprve je nutné otestovat, zda je název platný pro soubor, následuje testování, zda program má přístup k tomuto souboru a lze jej otevřít / vytvořit. Dále se po této fázi upraví ukazatel na výstup programu.

Funkce `analyza($vstupni_soubor, $parametr)`

Tato funkce je alfou a omegou programu. Při její implementaci jsem využil znalosti z formálních jazyků a konečných automatů. V úvodu funkce jsou definované dvě pole, které obsahují všechny možné klíčové slova a kombinace jednoduchých operátorů. Dále zde jsou umístěny proměnné pro počítání jednotlivých výskytů a identifikaci určitého stavu. Následně je implementován konečný automat, který správně identifikuje načtený znak a provede příslušnou operaci. Bylo nutné použít funkci `fseek()`, aby nedocházelo v některých případech ke ztrátě znaku a následnému znemožnění správné identifikaci. Program vrací proměnou podle parametru `$parametr`.

Pomocná funkce `obsah_adresare($start_adresar, $nosubdir)`

Funkce slouží ke zpracování zadaného adresáře. Rekurzivně prochází adresářovou strukturu a názvy souborů ukládá do pole, které také vrací. Hloubku zanoření lze ovlivnit parametrem `$nosubdir`. Pokud je tento parametr zadán, omezí se hloubka zanoření pouze na aktuální adresář. V opačném případě se zpracuje kompletní adresářová struktura od aktuálního adresáře hlouběji. Funkce do výstupního pole uloží pouze soubory *.c a *.h.

Pomocná funkce `formatovany_vypis($pole_vysledku, $fp)`

Funkce slouží ke zpracování správného formátovaného výstupu. Parametrem $pole_vysledku je předáno pole obsahující názvy souborů a počet výskytu zvoleného komentáře, klíčových slov, operátorů a řetězce. V parametru `$fp` se očekává ukazatel na otevřený výstupní soubor. Je zde plně využita možnost jazyka PHP mít jako klíč pole řetězec. Pro abecední řazení pole se používá funkce `ksort()`. Nejprve se projdou všechny klíče pole pomocí cyklu `foreach()` a vypočítá se nejdelší délka klíče. Z toho zjistíme, kolik mezer musíme do výpisu přidat. Následuje opětovné procházení pole a validní výpis na zadaný výstup. Pokud proběhlo vše v pořádku, funkce vrací hodnotu 0.

Ostatní pomocné funkce mnou implementované

Ostatní funkce `hledani_retezce($file_in, $hledany_retezec)`, `open_file($adresa_soubor)`, `print_err($cislo)` jsou velmi jednoduché. Tyto funkce vznikly z důvodu zvýšení přehlednosti programu. Z tohoto důvodu si nezaslouží vlastní odstavec a jejich implementace by měla být jasně čitelná přímo ze zdrojového kódu, případně náležitých komentářů.