Dokumentace úlohy CST: C Stats v PHP 5 do IPP 2014/2015
Jméno a příjmení: Jakub Pastuszek
Login: xpastu00

Dokumentace k projektu IPP 2014/2015, CST: C Stats

Analýza zadání

Cílem projektu bylo vytvořit skript v jazyce PHP, který bude zobrazovat statistiky zdrojových souborů jazyka C dle standardu ISO C99 a bude spustitelný na serveru Merlin. Skript lze spouštět s různými přepínači, a tím upravovat podobu výstupu.

Postup řešení

Prvotní událostí při spuštění skriptu je definice vstupních parametrů, které budou převzaty jako přepínače skriptu a jejich následná kontrola. Dále definice funkce pro obsluhu chybových stavů, která ve skriptu plní důležitou roli. Poté jsou definovány funkce pro vyhledávání a počítání výskytu vzorků v daném řetězci a funkce pro jejich odstranění. Každý krátký přepínač má svou obslužnou funkci, která může volat pomocné funkce.

Mezi pomocné funkce patří dvě nejdůležitější funkce na vyhledávání regulárního výrazu v daném řetězci, kde jedna vrací počet nalezených výskytu vzorku a druhá celkovou délku nalezených vzorků. Další neméně důležité funkce jsou pro vymazání části řetězce, opět podle regulárního výrazu. Volající funkce si nastaví, zda chce vymazat jeden nebo více z následujících možnosti, kterými jsou makra, řetězce, literály, či komentáře. Poslední pomocná funkce počítá, a následně vkládá, počet mezer na výstup pro správné zarovnání.

Dále je třeba nastavit výstupní soubor, který může být zadán přepínačem *--output*, jinak se vypisuje na standardní výstup. Následně při procházení jednotlivých souborů se počítá maximální délka názvu souboru (buď s cestou, nebo bez - dle přepínače *-p*) pro správné zarovnání výstupu a celkový počet výskytů počítaných hodnot.

Pro průchod adresářem jsem použil třídu *DirectoryIterator* a pro podadresáře třídu *RecursiveDirectoryIterator* s iterátorem *RecursiveIteratorIterator*.

Nakonec se provede seřazení podle názvů souborů a jejich následné vypsání i s počítanou hodnotou na výstup. Za výpis souborů se vypíše také celkový počet nalezených hodnot.

Funkce

Jedna z hlavních funkcí se stará o zpracování přepínače *-k*. Hledá klíčová slova s některými omezeními, kdy např. klíčové slovo nemusí být odděleno bílými znaky, ale může být v závorkách (přetypování) nebo bezprostředně za ním může následovat hvězdička (ukazatel). Dále volá pomocnou funkci pro nalezení počtu výskytu vzorku.

Funkce pro přepínač *-o* (vyhledání operátorů) nejprve projde řetězec a do jisté míry vymaže deklarace ukazatelů (* v deklaraci není operátor) a až poté zahájí vyhledávání počtu výskytů daného vzorku. V případě zadání přepínače *-s* (rozšíření TER) se počítá i počet výskytů ternárních operátorů.

Přepínač *-i* (vyhledání identifikátorů) a její funkce nejprve naleznou neplatné identifikátory začínající číslem a hexadecimální číslice, které se vymažou, a poté se spočítá počet všech identifikátorů včetně klíčových slov, které se musí následně odečíst. Toho se docílí tak, že se zavolá funkce pro spočítání klíčových slov a odečtením vrácené hodnoty od celkové získám přesný počet identifikátorů.

Funkce pro vyhledávání vzorku (přepínač *-w*) prochází celý řetězec a hledá nepřekrývající se vzorky, přitom sčítá počet výskytů.

Počet znaků komentářů (přepínač *-c*) zpracovává jak řádkové, tak i blokové komentáře včetně rozšíření COM, které počítá znaky komentáře i v makrech.

Zpracování parametrů

Parametry jsou zpracovávány pomocí funkce *getopt*. Tato funkce ulehčuje zpracování parametrů, ale má i své úskalí, kdy např. nezjistí žádné jiné parametry, než ty, které byly definovány. Tento problém jsem musel řešit u zpracování přepínače *--help*, kde kontroluji počet parametrů pomocí globální proměnné *$argv*. Každý jednotlivý přepínač se může při volání skriptu vyskytnout pouze jednou, což je patřičně ošetřeno.

Zpracování vstupních souborů

Vstupní soubor je načten jako souvislé pole znaků (řetězec), v PHP reprezentováno datovým typem *string*. Toto řešení jsem si zvolil, protože se mi lépe pracovalo s celým řetězcem, než kdybych měl pole řetězců rozdělených novým řádkem.

Pokud je zadána složka, jsou postupně procházený a analyzovány soubory v dané složce a poté i rekurzivně v jejich podsložkách (pokud není uveden přepínač *--nosubdir*).

Závěr

Projekt jsem z počátku řešil na svém PC, kde jsem měl nainstalovaný PHP server, na kterém jsem si zkoušel funkčnost skriptu. Před pokusným odevzdáním jsem začal testovat na školním serveru Merlin. Testování regulárních výrazů jsem prováděl na serveru www.regex101.com.

S jazykem PHP jsem se již setkal dříve, proto jsem se sémantickou stránkou jazyka neměl problém. Projekt jsem z počátku řešil pomocí konečných automatů, ale po zjištění jednoduchosti použití regulárních výrazu jsem je do projektu začlenil. Paradoxně největší problémy při tvorbě projektu nastaly v regulárních výrazech, kdy při testování jsem postupně zjišťoval a opravoval jejich nedostatky.