Dokumentace úlohy CST: C Stats v češtině do IPP 2014/2015
Jméno a příjmení: Ján Jusko
Login: xjusko00

# 1 Analýza zadání

Cílem zadaného projektu bylo vytvořit skript v jazyce PHP 5, který analyzuje zdrojové soubory napsané v jazyku C podle standardu ISO C99. Skript je schopen vypsat počty jednotlivých základních kamenů syntaxe jazyka C (identifikátory, klíčová slova, operátory, komentáře) jednotlivo pro každý analyzovaný zdrojový soubor a spolu pro všechny.

# 2 Postup řešení

## 2.1 Zpracování argumentů

Hned po spuštení skriptu se provádí zpracováni parametrů pomocí funkce *getopt*, která načte zadané argumenty i s jejich hodnotami do asociativního pole *argArray*, pak se ověří správnost parametrů i případné vzájemné vyloučení parametrů a nakonec se pro jednoznakové parametry inicializují globální proměnné jako příznaky, jestli jsou nebo nejsou zadané. Pro získání cesty k vstupnímu nebo výstupnímu souboru se přistupuje přímo do asociatívniho pole *argArray* podle klíče *input* nebo *output*.

## 2.2 Hledání validních souborů

Jedním z hlavních úskalí této úlohy bylo vyhledání všech souborů, které se mají podle aktuálně zadaných parametrů analyzovat. Vplyv na to má kombinace parametrů –input a –nosubdir a vo všeobecnosti mohou nastat 4 případy. Ve všech případech skript naplní asociativní pole *filenames* kde jako klíče jsou názvy souborů buď s absolutní cestou nebo jenom název souboru a v hodnotách pole jsou načtené obsahy příslušejících souborů. Je-li zadán konkrétní soubor, zkontrolujou se jeho práva pro čtení pomocí funkce *is_readable*. Když soubor není možné číst, skript končí s návratovým kódem 2. Když se má prohledávat jenom aktuální/konkrétní adresář, využívam funkce getcwd/scandir, které vrací pole souborů nad kterým pak iteruji, kontroluji správnou příponu a zároveň práva pro čtení a vkládám do pole *filenames*. V případě, že se mají prohledávat i podadresáře, využívam třídu *RecursiveDirectoryIterator*, která je velmi jednoduchá na použití. Pomocí ní iteruji nad všemi soubory v adresářové struktuře a taky kontroluji příponu, práva pro čtení a vkládam do výsledního pole. Když skript narazí při prohledávaní adresářů na soubor, který se nedá číst, tak končí s návratovým kódem 21. Hned po nalezení všech validních souborů, se lexikograficky seřadí pomocí funkce *ksort*.

## 2.3 Počítací funkce

Pro spočítání jednotlivých typů prvků zdrojových kódů je potřeba k souboru přistupovat vždy jinak. Proto jsem si nadefinoval 5 funkce, které řeší počítaní klíčových slov *(keywords)*, identifikátorů *(identifiers)*, znaků komentářů *(comments)*, operátorů *(operators)* a pro nalezení podřetezce slouží funkce *pattern*. Všechny tyto funkce pracují pomocí regulárních výrazů, které vždy nejprve smažou části kódu, které se neanalyzujou (makra, řetezce, literály...). Pak se *match*nou hledané prvky kódu, které se pak sečtou. Ve funkcích *keywords* a *operators* využívam pole naplnené klíčovými slovami/operátormi, z kterého se pak jednotlivo načítají a vyhledávají. Pro vyhledávaní podřetezce slouží šikovná vstavaná funkce *substr_count* která dokonale splňuje specifika ze zadání. Pro spočítání znaků komentářů se nejprve složitým regulárním výrazem matchnou blokové komentáře a pak řádkové do stringu, ve kterým pak sečtu znaky. Počítání identifikátorů je bohužel trochu ťežkopádní, protože se při tom skript snaží vymazat veškeré ostatní prvky kódu a pak sečíst počet slov v celém souboru. Po dlouhém testování to však funguje dostatečně spolehlivě.

## 2.4 Analýza souborů

Funkce analyze pak vezme pole souborů *filenames* a podle zadaného parametru volá počítací funkce pro každý soubor. Výstupem této funkce je opět asociativní pole *analArray*, které obsahuje názvy souborů jako klíče a hledané počty jako hodnoty.

## 2.5 Výpis

O zápis do výstupního souboru nebo na standardní výstup se stará funkce *vypis*, nejzaujímavější část této funkce je ale algoritmus pro vypočtení potrebné mezery mezi názvem a číslem pro každý řádek. Algoritmus funguje na 100% a řeší i případy, když nejdelší název má nejmenší číslo atd. Princip je v tom, že nejprve se vyhledé nejdelší název i číslo. Pak pro každý řádek se výpočte nasledujícim vzorcem kolik mezer je potreba:

$$number\_of\_spaces = (longest\_name - act\_name) + (longest\_number - act\_number) + 1$$

## 2.6 Závěr

Projekt jsem vypracovával a průběžne testoval na operačním systéme *Ubuntu 12.04 LTS*. Používal jsem vlastní i oficiální testy a závěreční testování jsem prováděl na školním serveru Merlin. Doba práce na projektu cca 2 týdny.

## 2.7 Metriky kódu

549 LOC
1 soubor