Dokumentace úlohy CHA: C Header Analysis v PHP 5 do IPP 2016/2017
Jméno a příjmení: Milan Augustín
Login: xaugus09

Popis problému

Úlohou bylo vytvořit skript v jazyce PHP, které analyzuje soubory jazyka C do výstupu v XML formátu. Hlavním předpokladem je, že vstupní soubory splňují normu ISO C99. Řešení využívá kombinaci konečného automatu a regulárních výrazů.

Zpracování argumentů

Celý skript má několik možných konfigurací, které jsou zpracovávané prostřednictvím funkce `getopt`. Ty jsou následně kontrolovány na duplicitu a validitu. Současně se vytvoří rekurzivním průchodem seznam (pole) souborů, určených pro analýzu ze zadaného vstupního adresáře, anebo implicitně z aktuálního adresáře. V případě, že byl zadaný vstup konkrétní soubor, je rovnou přidaný do pole a vznikne pouze jednoprvkové pole. Tím se úloha zjednoduší a analýza se provádí pouze nad jedním souborem.

Odstranění nepodstatných částí

Důležitou částí je odstranění nepodstatných částí, kvůli možným kolizím s vyhledáváním funkcí. Zde jsem použil kombinaci regulárních výrazů a velmi jednoduchý konečný automat.

Na začátek jsem odstranil každou dvojici znaků, která označuje, že řádek pokračuje na dalším řádku. Tím mám například zaručené, že všechny makra budou pouze na jednom řádku. Dále je potřebné odstranit jednoduché a blokové komentáře, navíc ještě textové řetězce. Vytvořit správný regulární výraz se mi zdálo velmi komplikované, a proto jsem zvolil jednoduší cestu formou konečného automatu (Obrázek 1). Následně se odstraní makra pomocí regulárního výrazu.

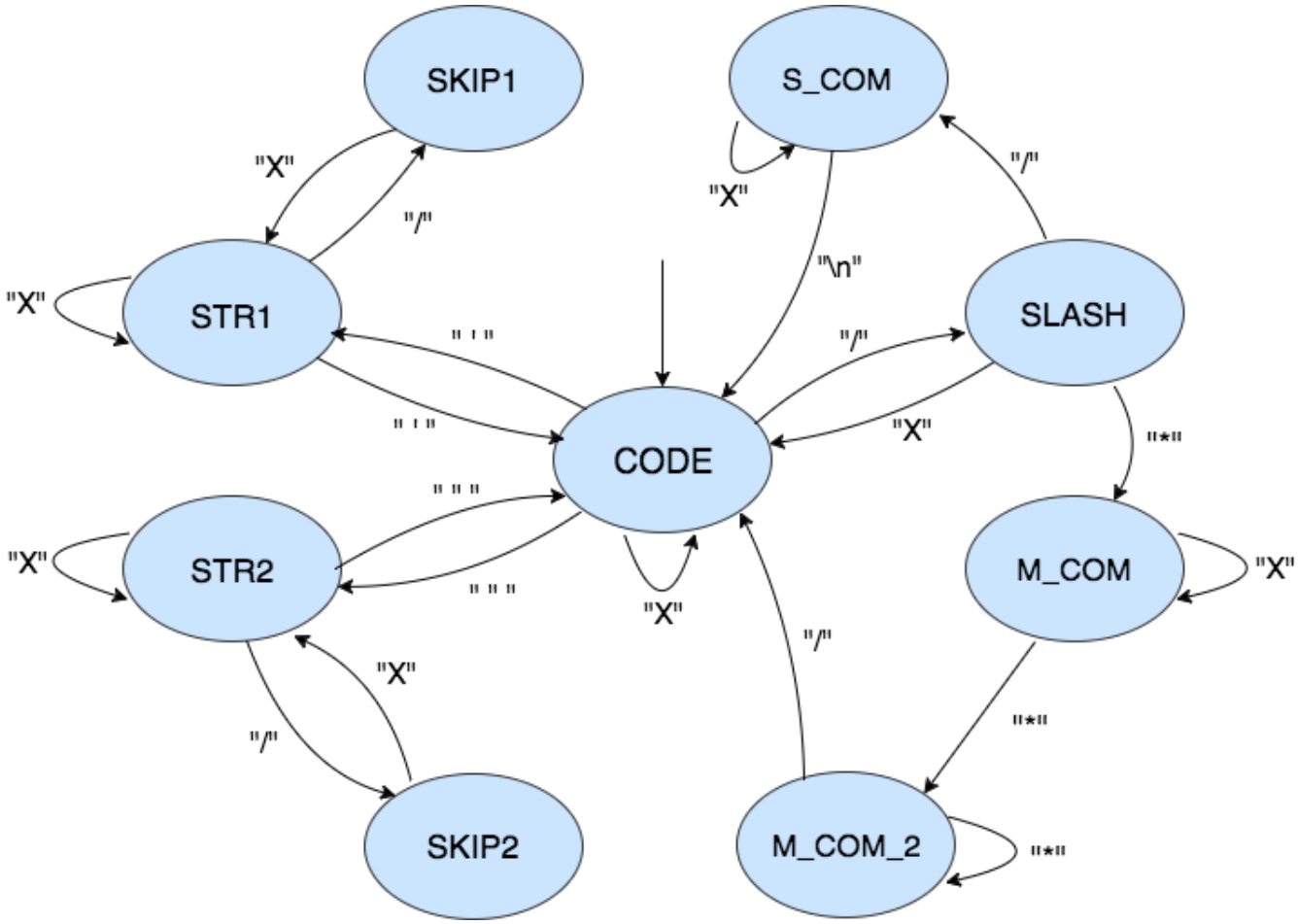
Jelikož se můžou vyskytovat v definicích funkcí anonymní funkce (jejich deklarace), střetl jsem se s menším problémem, protože se v těle funkce objevil následující kus kódu: `int var1 = var2 * foo(var3);` . Regulární výraz, který používám na vyhledávání vzal `var2 * foo(var3);` , což je chyba. Proto jsem všechny přiřazení pomocí dalšího regulárního výrazu odstranil a nahradil znakem `;` jako případnou zarážku.

Analýza souboru

Po odstranění nepodstatných částí je zbytek souboru připravený na analýzu. Po bližším nastudování regulárních výrazů jsem si našel rovnou celé funkce, kde jsem si jednotlivé shody rozdělil do třech částí pomocí klíčů na návratový typ, název a argumenty. Tato věc mi výrazně zpřehlednila a zjednodušila práci nad těmito údaji.

Formátovaní výstupu

Na formátování výstupu jsem použil „XMLWriter“, se kterém se velmi dobře manipuluje. Celé vkládání údajů do XML dokumentu probírá současně s analýzou, kde se postupně sbírají informace. Po ukončení analýzy se celý XML dokument uzavře a převede na řetězec. Následně se zkontroluje přítomnost přepínače `--pretty-indent` , pokud je zvolen, příslušně se odsadí jednotlivé elementy a výstup je tak kompletně zpracován.



Obrázek 1: Diagram konečného automatu. (Poznámka: "X" – značí všechny ostatní znaky)