Implementační dokumentace k 1. úloze do IPP 2019/2020

Jméno a příjmení: Tomáš Julina

Login: xjulin08

Program *parse.php* jsem při svém řešení interně rozdělil na dvě hlavní funkce – *scanner* a *syntax*.

Funkce *scanner* se stará o lexikální část analýzy, při každém zavolání funkce nejprve načte celý řádek (pomocí funkce *preg_match* a díky regulárním výrazům rovnou ignoruji řádky, které se skládají pouze z komentářů, či z bílých znaků), načtený řádek poté pomocí funkce *explode* rozdělím na část „s kódem“ a část s případnými komentáři, kterou budu dále ignorovat, takto rozdělený řádek dále pomocí funkce *preg_split* rozdělím na slova, s bílým znakem jako děličem, a toto uložím do pole.

Následně procházím takto zpracovaný řádek po jednotlivých slovech a provádím lexikální kontroly s pomocí bohatých možností regulárních výrazů v jazyce PHP, každé slovo na řádku, v případě, že je lexikálně správně, postupně ukládám do výsledného pole *$output (je výstupem této funkce)*, které obsahuje identifikátor tokenu (*tokenConst, tokenVar*, apod…), datový typ konstanty, případně hodnotu daného slova. V případě, že se jedná o jméno návěští, či instrukci navíc zavolám funkci *checkInstruction*, která porovná aktuální slovo s pomocným polem, které obsahují veškeré podporované funkce jazyka IPPcode20 a na základě výsledku rozhodnu, zda se jedná o instrukci či nikoliv. V případě špatného formátu slova dochází k chybě 23.

Funkce *syntax* pro tvoření výsledného kódu v jazyce XML využívá třídu *DOMDocument*. Funkce nejprve zavolá *scanner* a zkontroluje správný tvar prvního řádku vstupního souboru, poté následuje cyklus *while* který zpracovává soubor až do okamžiku, než od funkce *scanner* obdrží token *EOF (end of file)*, po každém zavolání funkce *scanner* se obdržené pole (array) s lexikálně zpracovaným řádkem dále zpracovává formou konečného automatu prostřednictvím příkazu *switch*. Samotný příkaz *switch* dostává jako parametr číslo instrukce z aktuálně zpracovávaného řádku, jednotlivé případy *case* jsem rozdělil do vhodných skupin dle počtu a druhu vstupních parametrů instrukce. Díky vhodnému uložení informací o jednotlivých řádcích do pole mohu jednoduše provádět syntaktické kontroly a v případě správnosti pouze vytvořím vhodné elementy a nastavím potřebné atributy pro generování výsledného XML kódu
Poznámka: Vzhledem k tomu, že je v případě některých instrukcí zadáním přímo stanovený například datový typ povolené proměnné provádím již některé kontroly v rámci skriptu *parse.php* – např. instrukce „ADD GF@a int@40 string@2“ není korektní a je o tom možné rozhodnout již nyní, tudíž program skončí s chybovým kódem 23 (nikoliv až v interpretu kódem 53). V případě, že všechny kontroly a generování kódu proběhly v pořádku je na konci programu výsledný kód vytisknut na standardní výstup.

Rozšíření *STATP* – na začátku programu zpracovávám vstupní argumenty a ty si ukládám do pomocného pole, současně kontroluji povolené kombinace argumentů, statistiky z většiny implementuji již ve funkci *scanner*, některé pak ve funkce *syntax* – prostřednictvím pomocných počítadel, hotové statistiky následně pomocí funkce *run_stats* vypisuji v pořadí vstupních argumentů do výstupního souboru *file*.