Dokumentace k implementaci první části projektu v předmětu IPP

Příjmení a jméno: Valecký David

Logín: xvalec01

Název: parse.php

Předmětem této části projektu je skript, který provede lexikální a syntaktickou analýzu pro jazyk IPPcode19 a převede ho do jazyka XML. Skript je zpracován v jazyce PHP verzi 7.3.

Implementace:

Skript je spouštěný uživatelem a jako vstup je dán soubor přesměrovaný na standardní vstup*:* `php7.3 parse.php <vstupní soubor` nebo je možné psát vstup do konzole, ze které bude skript číst: `php7.3 parse.php`. Další možností je spustit soubor s argumentem `--help`, kdy se vypíše nápověda a shrnutí funkčnosti skriptu: `php7.3 parse.php --help`. Jiné spuštění se bere jako nesprávné a skript vrací `error 11`. S nesprávným počtem argumentů vrací `error 10`.

Skript načítá řádky ze standardního vstupu a upravuje je do vlastní podoby (odebírá bílé znaky) a následně rozdělí přijatý kód na tokeny dle mezer s pomocí funkce `explode()`. Sekvence započatá znakem (`#`) je brána jako komentář, který se při lexikální analýze odebere. Následně jsou tokeny uloženy do pole `$codeLine`, po tom, co projdou žádoucími kontrolami, aby s nimi mohla dál pracovat syntaktická analýza.

První kontrola je, jestli první načtený token obsahuje informaci, jakým jazykem je přijatý kód psán (.IPPcode19). Následně při splnění všech předchozích náležitostí, jsou vytvořené pole (např. `$instructTwoVarSymb`) pojmenované podle počtů atributů (neterminálů) v něm uložených názvů instrukcí (např. `CREATEFRAME`). Rozřazení je systematické pro další kroky.

Pro potřebnou kontrolu správnosti zapsání jednotlivých neterminálů a hodnot jsou použity regexy pro PHP.

Následuje inicializace pole pro využití XMLWriteru a vytvoření hlavičky XML a jejich náležitostí (verze, jazyk, nastavení odsazení, atp.)

Dále je použit už zmíněný cyklus (`while`). Načítané řádky, rozdělené do tokenů, příjme automat (syntaktická analýza), sestrojený z konstrukcí (`if`), a kontroluje syntaxi jazyka .IPPcode19. V těle jednotlivých konstrukcí je kontrola správnosti syntaxe s pomocí funkce `preg_match`, ve které jsou využity dříve zmíněné regexy, která podle druhu chyby vrací `error 21|22|23`.

`Error 21` vrací skript při chybě v hlavičce zdrojového kódu. `Error 22` při chybně zapsaném operačním kódu. `Error 23` je pak jiná lexikální nebo syntaktická chyba.

Při úspěšné analýze se s tokeny vygeneruje XML. Při načítání jednotlivých řádků se inkrementuje proměnná `$lineNumber`, která značí, o kolikátou instrukci v jejich posloupnosti se jedná.

Ve výsledku jsou všechny instrukce vypsány na standardní výstup v jazyce XML.