Implementační dokumentace k 1. úloze do IPP 2018/2019
Jméno a příjmení: Michal Plšek
Login: xplsek03

Dokumentace k 1. úloze

Parser kódu IPPcode19 je v php řešen jako konečně stavový automat, který využívá analýzu každé operace (umístěné na každém řádku zvlášť) pomocí regulárních výrazů. Parser ignoruje prázdné řádky a komentáře, které mohou být umístěné za jakoukoliv instrukcí (s předcházejícím libovolným množstvím mezer) nebo na samostatném řádku. V případě že je komentář nebo prázdný řádek (nebo jakýkoliv jiný řádek s obsahem .IPPcode19) umístěn před hlavičkou kódu `.IPPcode19`, skript hlásí na STDERR chybu: je to logické vzhledem k tomu že i komentáře jsou součástí jazyka IPPcode19, a neměly by být obsaženy před tím než vůbec dojde k identifikaci jazyka. Rozšíření implementováno nebylo, moc práce kvůli jednomu bodu.
Pro generování XML výstupu je použit `xmlwriter`. Na začátku skriptu je inicializován nový XML objekt a nastaveno ukládání dat do paměti (memory buffer). Je to z toho důvodu aby se při nalezení případné syntaktické chyby v IPPcode19 na STDOUT nic nevypsalo, k vypsání z bufferu dojde až po načtení všech dat ze STDIN, poté je buffer automaticky vyčištěn, v outputMemory() XML bufferu je defaultní hodnota flush=TRUE.
Kvůli absenci rozšíření skript podporuje pouze vstupní argument `--help`, který vypíše stručnou nápovědu o skriptu. Pokud se v `argv` nachází více argumentů než jeden nebo nějaký jiný, skript vyhodí chybu na STDERR.
Po validaci vstupních argumentů a prvního řádku skriptu přecházíme k validaci jednotlivých instrukcí. Pokud se jedná o vstupní kód, který má validní hlavičku, ale nemá žádné instrukce, XML výstupem je prázdný element /textttprogram. Skript nejprve pomocí regexu zjistí obsah řádku: přednostně jestli se jedná o prázdný řádek nebo řádek s komentářem (takové přeskakuje), poté o kolika adresnou instrukci (počet argumentů instrukce) se jedná (zatím tedy nemáme jistotu že se o instrukci vůbec jedná). Pokud není řádek vhodně rozdělen skript hlásí syntaktickou chybu. Pokud řádek rozdělen je, skript určí na kolik částí (operační instrukce + 0-3 argumenty). Poté začne testovat operační kód instrukce, ve skriptu jsou uložena pole s validními názvy operačních kódů. Pokud skript zjistí že se jedná o validní operační kód, podle typů argumentů (máme 4 typy argumentů: LABEL, TYPE, VAR, CONST) které jsou ve skriptu uloženy k odpovídajícím operačním kódům (toto řešení by bylo nepraktické, pokud by každá operace měla různé typy argumentů, tady je velmi úsporné protože v IPPcode19 mají operace o stejném počtu argumentů velmi podobné typy argumentů) otestuje, zda se jedná opravdu o požadovaný typ. Pokud ano, skript umožní v dalším kroku vytvořit XML element a uložit do něj postupně proměnlivý počet argumentů. Zajímavá část kódu zpracovává ukládání argumentů do XML:

```
for($i = 1; $i < $instruction; $i++) {
        $writer->startElement('arg' . $i);
        $writer->writeAttribute('type', ${'arg' . $i}[0]);
        if(${'arg' . $i}[0] == "var" || ${'arg' . $i}[0] == "string"
        || ${'arg' . $i}[0] == "label")
                $writer->text(xml_escape(${'arg' . $i}[1]));
        else
                $writer->text(${'arg' . $i}[1]);
$writer->endElement();
```

Je zde použito vlastnosti, kdy se spojuje název a proměnná z cyklu, a vytváří název XML elementu, který bude vytvořen. Takhle je ošetřeno vkládání proměnlivého počtu argumentů do XML objektu. Pokud se jedná o typ `string nebo var nebo label`, tak je navíc ošetřeno nahrazení rezervovaných XML znaků v obsahu názvu argumentu: výpis textu je zpracován pomocí funkce writeRaw(). Je to kvůli tomu, že funkce text() převádí automaticky speciální znaky a museli bychom převádět znak zvlášť.
Pokud je parsování dokončeno bez syntaktických chyb, je možné odeslat obsah bufferu v paměti na STDOUT a skript vrátí nulu. Skript vrací následující kódy: 10: chyba parametrů skriptu, 11: chyba při otevírání vstupního souboru, 21: chyba hlavičky IPPcode19, 22: chyba operačního kódu, 23: syntaktická/lexikální chyba kódu.

Detaily podrobné implementace použitých funkcí včetně popisu parametrů a návratových hodnot ve zdrojovém kódu.