Implementační dokumentace k 1. úloze do IPP 2020/2021

Jméno a příjmení: Vasil Poposki

Login: xpopos00

První úloha (skript `parse.php`) byla navržena jako filtr, který na vstup dostane program v jazyce IPPcode21, což je imperativní jazyk typu assembler a na výstup vypíše XML reprezentaci programu. Samotná implementace se skládá ze dvou částí – lexikální a syntaktická analýza a generování XML kódu.

V průběhu své implementace jsem se nejvíc zaměřil na ověření správnosti programu použitím regulárních výrazů. Pro generování XML výstupu jsem v programu použil metody z třídy `XMLWriter`, což umožnilo snadné generování formátované XML reprezentace programu. Hlavní část skriptu je `while` cyklus, během kterého se provede lexikální a syntaktická analýza voláním pomocných funkcí pro ověření správnosti operandů u instrukcí zdrojového kódu a úpravou vstupního kódu za účelem efektivního provádění analýzy. Během cyklu se zároveň generuje XML kód.

Popis pomocných funkcí:

`line_arrange($l)`

- Provádí úpravu načteného řádku v IPPcode21. Použitím regulárních výrazů odstraní z řádku komentáře, zbytečné bíle znaky a jiné speciální znaky.

`p_var($var), p_const($const), p_label($label), p_type($type)`

- Používají vestavěnou funkci `preg_match`, která pomocí regulárního výrazu hledá vzor v daném vstupním parametru. Funkce pro jednotlivé typy operandů (<var>, <const>, <label>, <type>) ověřují, zda typ operandu odpovídá dané instrukci. Taktéž se ověřuje korektní zápis operandu, např. escape sekvence u typu `string` nebo správná hodnota u typu `bool a nil`.

`instr_get_type($instr, $instr_arg)`

- Pomocná funkce sloužící k výpisu XML reprezentace programu. Na základě předaných parametrů zjistí typ operandu.

`instr_get_text($instr_arg)`

- Funkce vrací obsah operandu. Pokud je operand konstanta(int, string, bool, nil) vezme se pouze podřetězec za znakem '@'.

Tok programu:

Volá se funkce pro úpravu právě načteného řádku. Pro kontrolu instrukcí a jejích parametrů je načten řádek rozdělen na podřetězce. Název instrukce je pak na indexu 0 v řetězci (`$arr[0]`).

```
$arr = explode(" ", trim($l, "\n"));
```

Během switch-case konstrukce kontroluje se název instrukce, počet operandů u instrukcí a volají se pomocné funkce pro lexikální a syntaktickou analýzu. Pokud program narazí na chybu, skončí odpovídajícím chybovým kódem a chybovým hlášením. Zároveň se generuje XML kód. Pokud analýza proběhla v pořádku, vypíše se XML reprezentace na standardní výstup.