Implementační dokumentace k 1. úloze do IPP 2020/2021

Jméno a příjmení: David Hudák
Login: xhudak03

1 Úvod

Tato dokumentace se zabývá řešením prvního projektu v rámci předmětu principy programovacích jazyků. Jedná se o první zpracování IPPcode21 do XML dokumentu. Programová dokumentace je v českém jazyce a kód včetně komentářů v jazyce anglickém.

Projekt je řešen v programovacím jazyce PHP ve verzi 7.4 a k implementaci zapisování XML do souboru jsem využil funkcí knihovny XMLWriter. K použití jejích funkcí jsem přímo využil úryvky z webové dokumentace[1]. Ostatní funkcionality byly používány na základě přednášené látky v tomto předmětu.

2 Instrukční sada

V projektu je nutné řešit existenci daných instrukcí (příkazů) jazyka IPPcode21. Dále je nutné u nich kontrolovat správný počet argumentů a jejich typy[2]. Toho jsem dosáhl s využitím asociativního pole obsahující jako své prvky pole. Tedy nejprve se vyzkouší, zdali se prvek nachází v poli – jméno instrukce na vstupu se zvětší na velká písmena a následně se použije jako klíč asociativního pole. Pokud dojde k nálezu (s pomocí funkce `array_key_exists()`), použije se znovu jako klíč k přístupu a asociativní pole vrátí pole o dynamické velikosti. To obsahuje dvě základní informace – počet argumentů funkce ve formě čísla na první pozici a následně výčet argumentů na pozicích za instrukcí tak, jak jdou za sebou.

Jelikož PHP ve verzi 7.4 neobsahuje výčtový datový typ, byl pro rozlišování datových typů použit nepříliš efektivní, ale za mě dobře přehledný, datový typ `string`. Pro rozlišení návěští bylo použito slovo `label`, pro typ u instrukce `read` slovo `type`, pro nutnost proměnné slovo `var` a pro možnost čehokoliv krom návěští slovo `nlabel`.

3 Načítání vstupu

Kód musí nutně začínat klíčovým slovem `.IPPcode21`. Dojde tedy k odstranění řádků/komentářů na začátku souborů a vyzkouší se, zdali je první kus kódu `.IPPcode21`. Pokud ano, nastaví se potřebné hlavičkové údaje XML souboru a přejde se k dekódování instrukcí.

Hlavní cyklus programu provádí postupné načítání řádků. Z každého řádku jsou nejprve odstraněny funkcí `substr()` s pomocí funkce `strpos()` komentáře (řetězce začínající znakem `'#'`). Následně se pomocí regulárního výrazu `/\s+/` a funkce `preg_split()` řádky rozbíjí na pole. Pak se s pomocí postupu uvedeného v kapitole 2 rozliší existence instrukce a její potřebné údaje a následně se pokračuje lexikální a syntaktickou kontrolou (viz 4 a 5).

4 Lexikální kontrola

V programu se provádí kontroly lexémů nejprve rozdělením podle přítomnosti znaku `@` na návěští a typy, nebo na proměnné a konstanty. U návěští a u proměnných se kontroluje, zdali řetězec začíná na speciální znak, nebo na písmeno. U konstant bylo komplikovanější akorát rozlišit legální escape sekvence – toho bylo dosaženo s pomocí získání podřetězců po znaku `\` a následně spočítání korektního počtu cifer (3).

5 Syntaktická kontrola

Syntaktická kontrola běží společně s lexikální (viz 4). Tedy dochází k načtení řádku, přečtení instrukcí. Po rozložení řádku na slova s pomocí dříve uvedeného regulárního výrazu pak program spočítá počet slov a očekávaný počet slov. Následně se v cyklu kontroluje správnost typů (každý argument má svou iteraci). Po rozlišení argumentu na typy a návěští, nebo na konstanty a proměnné se provádí porovnání zjištěného typu (na základě předpon `GF`, `LF`, `int` atd.) a očekávaného typu (opět viz 2).

6 Závěr

Výsledkem řešení projektu je program, který je schopen převést soubor ze standardního vstupu na jeho XML konverzi. Program byl testován na Ubuntu v systému WSL a na školním serveru Merlin.

[1] Viz `https://www.php.net/manual/en/example.xmlwriter-simple.php`

[2] (nikoliv po stránce sémantiky stylu, že sčítání může sčítat pouze dvě čísla typu int, nýbrž po stránce kontroly, že například instrukce JUMP má jako argument návěští)