Implementační dokumentace k projektu do IPP 2017/2018
Jméno a přijmení: Jan Kubica
Login: xkubic39

1 Soubor parser.php

1.1 Popis

V souboru `parser.php` se nachází implementace základního zpracování zdrojového kódu napsaného v jazyce *IPPcode18*. Jelikož se jedná o nestrukturovaný jazyk obsahující příkazy jednotlivě každý vždy na novém řádku, mnou implementovaný parser zpracovává daný zdrojový kód právě po řádcích. Každý řádek zdrojového kódu je analyzován samostatně, pro rozpoznávání výrazů na každém řádku jsou využívány jak funkce, které pracují s regulárními výrazy, tak základní funkce pro práci z řetězci jazyka *PHP*. Na počátku je každý řádek předán funkci `parse_line($line)`, která zajišťuje základní filtraci komentářů a bílých znaků, řádky neobsahující příkazy zdrojového kódu jsou zahozeny. Následná data, obsahující příkazy jazyka, jsou pak již ve formě pole předány jako parametr konstruktoru třídy `Instruction`.

1.2 Třída Instruction

V rámci třídy `Instruction` jsou nad těmito daty postupně volány metody `set_instruction($line_arr)`, `check_args_num($line_arr)` a `check_arg_types()`, které mají za úkol ověřit lexikální a syntaktickou správnost jednotlivých příkazů. Tyto metody postupně kontrolují platný název instrukce a operandů, správný počet argumentů a následně typově kompatibilní argumenty k dané instrukci. Takto zpracované objekty jsou ukládány do pole pro pozdější generování *XML* dokumentu.

1.3 Doplňující informace

Je zpravidla vyžadováno, aby se patřičná hlavička, typická pro kód v jazyce `IPPcode18`, nacházela vždy a pouze na prvním řádku zdrojového textu, jak je zmíněno v dokumentaci, a nepředchází ji žádné bílé znaky či komentáře.

K vytváření *XML* dokumentu bylo využito dostupné třídy `DomDocument`. Dané *XML* obsahuje vždy hlavičku s informacemi o kódování znaků a verzi samotného *XML*, i když je podle *XML* dokumentace tato hlavička je pouze doporučená.

Pro přívětivější obsloužení jednotlivých chyb a tisk na standardní chybový výstup při analýze byla implementována třída `myException` rozšiřující třídu `Exception` a metoda `errorHandler()`.

2 Soubor interpret.py

2.1 Použité třídy

Ve skriptu byly implementovány následující třídy: `Parameter` - reprezentující parametr instrukce převzatý z *XML* dokumentu, `Variable` - proměnnou, která se nachází v konkrétním rámci, `Frame` - třída (dočasného) rámce, ze které následně dědí třídy `LocalFrame` a `GlobalFrame`; dále pak `FrameStack`, `CallStack` a `DataStack` postupně jako zásobník rámců, zásobník volání a datový zásobník.

2.2 Popis

Interpret v souboru `interpret.py` navazuje na data zpracovaná parserem. Extrahuje data z vytvořeného *XML* dokumentu a ukládá si je v cyklu do vnitřní reprezentace. Při tomto převodu jsou extrahovány všechny potřebné informace, důležité před během programu, jako je pozice a příp. kontrola redefinice labelů nebo dané instrukce ve správném pořadí a jména parametrů. Vnitřní reprezentace by se dala popsat jako kolekce struktur - pole objektů třídy `Instruction`.

Před samotnou interpretací jsou inicializovány globální proměnné (jedináčci dříve zmíněných tříd) pro zásobník volání, datový zásobník a globální rámec. V následném průchodu vnitřní reprezentace je program interpretován, případné skoky ve vykonávání jsou řešeny změnou indexu v poli. Interpretace probíhá voláním metody `call()` třídy `Instruction`, která invokuje vždy patřičnou funkci pro danou instrukci, která se nachází pod stejným názvem jako samotná instrukce, přebírající parametry *order* a *pole s operandy*. Kontrola správných datových typů operandů se provádí za běhu uvnitř těchto instrukcí.

Mezi často volané funkce v programu patří metody pro práci s rámci, případně pak funkce `look_up_variable(name)` nebo `resolve_parameter_symb(order, f_name, param)`, `resolve_parameter_var(order, f_name, param)`, které vyhodnocují operand instrukce a vrací odkazy na konkrétní požadované proměnné.

2.3 Doplňující informace

Interpret poměrně hluboce kontroluje formát vstupního *XML* dokumentu, který je přesně dán dokumentací. U instrukce nepovoluje žádný jiný atribut než `opcode` a `order`, u programu pak kromě `language` přidává možnost atributu `name` a `description`. Povoluje zpřeházené tagy na stejné úrovní *XML*, pouze ale typu `instruction` a `arg`. Kontroluje správnost pořadí a zamezuje duplicitní hodnotě atributu `order`, více argumentů než tří u instrukce a celkově správného počtu operandů příslušících dané instrukci. Kontroluje validní jména instrukce, validní typy a opět částečně kontroluje textové hodnoty argumentů instrukcí s jejich datovými typy. Interpret bere hlavičku *XML* jako povinnou a její absencí dochází k chybě **31**.

Dále interpret nepovoluje redefinici již existující proměnné v daném rámci instrukcí `DEFVAR` a na tuto skutečnost upozorňuje chybou s návratovým kódem **59**.

3 Soubor test.php

3.1 Popis

V tomto skriptu se nachází dvě třídy `Folder` a `Test`. Pokud je využito rekurzivního prohledávání složky, dojde k vytvoření několika instancí třídy `Folder` naplněných instancemi `Test`. Každý z nalezených testů je spuštěn a výpis je dán na výstup jako stránka ve formátu HTML. Každá složka je na výsledné stránce zvlášť tvořena tabulkou, kde u každé se vpravo dole nachází souhrn o celkovém počtu úspěšných testů. U každého testu jsou zkoumány hodnoty: Test name, Parser processed, Interpret processed, Return code, Return code requested, Stdout diffcheck, Stdout a Stderr. Test je považován za úspěšný, pokud souhlasí návratové hodnoty a jejich výstup je shodný. V takovém případě buňka s názvem testu zezelená. V opačném případě je buňka zbarvená červeně. Celkový počet úspěšných testů ze všech testovaných je uveden na konci stránky vpravo.