Implementační dokumentace k 2. úloze do IPP 2018/2019
Jméno a Příjmení: Šimon Matyáš
Login: xmatya11

Skript interpret.py
Úkolem skriptu je provést interpretaci XML reprezentujici zdrojový kód v IPPcode19.

Vstupní XML
Interpret pracuje s výstupem skriptu parse.php musí být ale připraven že XML bude obsahovat chyby, kontroluje proto attributy tagů, upraví se pořadí instrukcí podle attributu „order“ , podle tohoto attributu zkontroluje i jestli nějaká instrukce nechybí, dále upraví pořadí argumentů jednotlivých instrukcí pokud jsou přeházené. Vyhledá i přebytečný text mezi tagy.

Proměnné
Na začátku skriptu se inicializují tyto proměnné:

- **GF -** slovník slovníků definovaných proměnných v globalním rámci.
  Příklad proměnné X s datovým typem int a hodnotou 5 => {X:{type:int,val:5}}
- **LF -** seznam slovníků proměnných v lokálním rámci, na začátku skriptu je nedefinovan
- **TF -** seznam proměnných v dočasném rámci, také není na začátku definován
- **FrameCount –** počet založených rámců
- **ThisFrame –** aktuálni rámec
- **Stack –** seznam hodnot a typů vložených do zásobníku používaný zásobníkovými instrukcemi
- **Labels –** slovník návěstí, klíč je název návěstí a hodnota je pozice instrukce
- **Next_instr –** bool jestli se má přejít na další proměnnou v pořadí nebo byl proveden skok
- **Active_instr –** index aktuální instrukce
- **CallStack –** seznam pozic odkud byla použita instrukce CALL. Po použití instrukce RETURN je další instrukce nastavena na poslední pozici v seznamu CallStack.

Switch instrukci
Cyklus while běží dokud index aktualni instrukce není vyšší než počet instrukcí ve vstupním XML. Podle attributu „opcode“ zpracovávané instrukce se ve switchy vyhledá příslušný kód, proběhne kontrola jestli instrukce má správný počet argumentů a jestli jsou argumenty správného typu. Pokud instrukce pracuje s proměnnou tak je nejprve zjištěno jestli byla proměnná již definována.
Při provedení skoku je index následující instrukce přepsán na pozici příslušného návěsí, pokud žádný skok neproběhl je následující instrukce zvolena podle pořadí.

Instrukce CALL a RETURN
Při volání instrukce CALL je aktualní pozice přidána na zásobník **CallStack** a je vykonán skok na příslušné návěstí.
Po použití instrukce RETURN je další instrukce nastavena na poslední pozici v zásobníku **CallStack**

Zásobník
Skript obsahuje rozšíření pro práci se zásobníkem, instrukce při zavolání zkontroluje jestli zásobník obsahuje dostatek prvků potřebných pro vykonání instrukce a jeji výsledek uloží na vrchol zásobníku

Skript test.php

Skript slouží pro automaticke testování skriptů parse.php a interpret.py

Argumenty

Nejprve skript zjistí jestli byl volán s vyhovujícími argumenty.

Hledání testu + Rekurze

Skript prohledává zadaný adresář a hledá soubory s koncovkou .src, pokud soubor s takovou koncovkou najde tak hledá stejnojmenné soubory s koncovkami .in .out a .rc, pokud některý z těchto souborů není nalezen tak je vygenerován s obsahem specifikovaným v zadání.
Pokud byl skript volán s argumentem **–recursive** tak po nalezení adresáře je na tento adresář rekurzivně použita funkce pro hledání testů

Průběh testu

Po vyhledání testu je podle argumentů se kterými byl skript spuštěn určeno jestli bude testován skript parse.php, interpret.py nebo oba. Pokud je test úspěšný tak se počítadlo úspěšných testů inkrementuje a jmeno testu je zapsáno do pole s úspěšnými testy, v opačném případě jsou aktualizovány proměnné pro neúspěšné testy.

- **parse only -** V případě testování skriptu parse.php je příkazem **exec** skript spuštěn s testem. Nejprve je exit code daného testu porovnán s testovým souborem s příponu .rc a pokud jsou shodné znamená to že buď test objevil chybu a vrátil správný exit code nebo pokud je exit code roven 0 znamená to, že skript proběhl v pořádku a je potřeba porovnat ještě výstup skriptu, výstup je tedy uložen do dočasného souboru který je poté porovnán pomocí nástroje **JExemXML** s testovým souborem s příponou .out.
- **int only –** testování skriptu interpret.py probíhá podobně jako u předchozího skriptu. Nejprve se příkazem exec spustí, porovnají se exit code a pokud proběhl skript bez chyb tak proběhne porovnání výsledků pomocí příkazu **diff.**
- **oba skripty –** při testování obou skriptů je nejprve použit skript parse.php, pokud je exit code roven 0 tak je jeho výstup nahrán do dočasného souboru a ten použit jako vstup pro skript interpret.py, před porovnáním výstupu je opět nejprve porovnán exit code s testovým souborem s příponou .rc a až poté je dočasný soubor přemazán výstupem skriptu interpret.py a porovnán příkazem **diff** se souborem .out.

Na konci testu je použitý dočasný soubor smazán.

Generování HTML

Po průběhu všech nalezených testů jsou pole úspěšných a neúspěšných testů upravena na string vhodný pro tisk, jsou z nich odděleny názvy adresářů které se přidávají zvlášť kvůli lepší přehlednosti.
Dále výsledné HTML vygeneruje tabulku obsahující celkový počet testů a počet úspěšných/neúspěšných testů.
HTML je generováno na standartní výstup pomocí příkazu **echo()**.