Implementační dokumentace k 2. úloze do IPP 2019/2020
Jméno a příjmení: Daniel Štěpánek
Login: xstepa61

Postup implementace

1 Interpret (interpret.py)

1.1 Struktura programu

Program je objektově navržený. Není využit žádný návrhový vzor. Překlad a interpretace je rozdělena do tří kroků. V prvním kroku probíhá kontrola vstupního XML a načtení instrukcí. Ve druhém kroku probíhá syntaktická analýza pomocí regulárních výrazů. Ve třetím kroku probíhá sémantická analýza a samotná interpretace.

1.2 Spuštění skriptu

Při spuštění skriptu je provedena kontrola vstupních parametrů. Pokud je vstupni parametr –help, je vypsána nápověda ke skriptu a následné ukončení vykonávání. Jinak, v případě správné kombinace parametrů, s využitím knihovny 'xml.etree.ElementTree' je provedeno parsování vstupního XML kódu. Jednotlivé instrukce jsou uloženy podle pořadového čísla *order* do slovníku.

1.3 Zpracování instrukcí

Nejprve je porovnán operační kód instrukce s klíči ve slovníku instrukcí ve třídě InstructionDict, tedy zda operační kód existuje a je správně zapsán. Pokud je operační kód správný, pomocí informací přiřazených operačnímu kódu ve slovníku klíčových slov, se určí počet argumentů a typy jednotlivých argumentů (var, symb, label, type). Jednotlivé argumenty jsou zkontrolovány regulárnímy výrazy.

1.4 Interpretace kódu

Interpret využívá dvou průchodů programem. Prvním průchodem jsou uložena všechna návěští do slovníku návěští. Druhý průchod je samotné provádění programu. Pro každou instrukci je vytvořena funkce ve třídě *Processor*. Tyto funkce jsou postupně volány v hlavní smyčce podle aktuální hodnoty order. Provádění programu je implicitně sekvenční, případně lze ovlivnit skokovými instrukcemi.

1.5 Třídy Frame, Stack

Pro práci s rámci je vytvořena třída *Frame*, jejíž implementace je řešena pomocí slovníku, s vlastnímy metodami. Zásobník volání (tzv. call stack) i datový zásobník jsou instance třídy *Stack*. Jenž je implementovaný dynamickým polem s vlastnímy metodami.

2 Testovací skript (test.php)

2.1 Spuštění skriptu

Při spuštění skriptu je provedena kontrola vstupních parametrů. Pokud je vstupni parametr –help, je vypsána nápověda ke skriptu a následné ukončení vykonávání. Jinak, v případě správné kombinace parametrů, program provádí testování.

2.2 Načítání testovacích souborů

Pokud je nastaven volitelný parametr –directory, je prohledávána zadaná složka a načítají se všechny soubory s příponou '.src'. Pomocí parametru –recursive, lze zajistit hledání souborů v podadresářích. Implicitně probíhá hledání souborů v aktuálním adresáři.

2.3 Testování skriptu parse.php (- -parse-only)

Explicitně lze nastavit cestu k parse.php pomocí parametru - -parse-script, jinak je použit (pokud je vytvořen) skript parse.php v aktuálním adresáři.
Pro vyhodnocení testování jsou použity nástroje *diff* (s parametry -q -b) pro kontrolu návratových kódů a nástroj *jexamxml* pro kontrolu vygenerovaného XML s referenčním XML. Explicitně lze nastavit cestu k nástroji jexamxml pomocí parametru - - jexamxml.

2.4 Testování skriptu interpret.py (- -int-only)

Explicitně lze nastavit cestu k interpret.py pomocí parametru - -int-script, jinak je použit (pokud je vytvořen) skript interpet.py v aktuálním adresáři.
Pro vyhodnocení testování je použit nástroj *diff* (s parametry -q -b) pro kontrolu návratových kódů i výstupu ze skriptu s referenčím výstupem.

2.5 Test celého překladu

V první části je provedeno parsování vstupního souboru s příponou '.src' skriptem parse.php, jenž vygeneruje XML kód přesměrovaný do souboru s příponou '.par'. Tento soubor je nyní vstupem pro skript interpret.py, jehož výstup je přeměrován do souboru s příponou '.int' a následně je porovnán výstup i návratový kód v souboru s připonou '.intrc' s referenčními soubory pomocí nástroje *diff*.

2.6 Generování HTML

Výstupem programu je generovaný HTML kód na standardní výstup. Pro každý test je dle úspěchu vygenerován HTML kód ve tvaru: Test passed/failed cesta k souboru. Je použito jednoduché stylovaní pomocí CSS.