Implementační dokumentace k 2. úloze do IPP 2020/2021

Jméno a příjmení: Tomáš Tomala
Login: xtomal02

1 interpret.py

1.1 Načtení argumentů

Zpracování argumentů z příkazové řádky je implementováno ve funkci `executeArgs`. Pro načtení argumentů je použita knihovna `argparse`, která použité argumenty uloží do pole argumentů. Funkce toto pole zpracuje, otestuje jestli vstupní soubory existují a se zadanými argumenty interpret dále pracuje. V případě že je v argumentech nepovolená kombinace, je skript ukončen s chybou 10.

1.2 Načtení XML

Po zpracování argumentů skript načte XML reprezentaci IPPcode21. Reprezentace je získána ze standartního vstupu, nebo ze souboru získaného při zpracování argumentů. XML reprezentace je zpracována pomocí knihovny `ElementTree`, která strukturu rozdělí do objektů. Po úspěšné kontrole hlavičky jsou načteny jednotlivé instrukce.

Ke každé instrukci jsou zjištěny údaje o argumentech, operačním kódu a jejím pořadí. Z těchto údajů se skladají jednotlivé objekty reprezentující instrukce. Při zpracování těchto instrukcí jsou kontrolovány nepovolené konstrukce XML reprezentace. Při načítaní XML se také ukládají názvy návěští a jejich pozice. V posledním kroku se instrukce seřadí v poli podle pozice.

1.3 Paměťový model

Skript reprezentuje proměnné pomocí třídy `Variable`. Třída obsahuje informace o jejím názvu, typu a hodnotě. Proměnné se ukládají do rámců, které jsou reprezentovány jako pole objektů této třídy. Konstanty jsou reprezentovány jako slovníky s informacemi o typu a hodnotě.

1.4 Emulace programu

Na začátku emulace se inicializují zásobníky volání, rámců a dat. Dále se inicializují rámce. O běh programu se stará cyklus, který prochází všechny instrukce získané z XML reprezentace IPPcode21. Cyklus také udržuje informace o běhu, jako je počet zpracovaných instrukcí. Tato informace je využita při emulaci instrukce `BREAK`. Pro každou instrukci je nejprve provedena kontrola typů a počtu argumentů. Poté se provádí emulace každé instrukce.

2 test.php

2.1 Načtení argumentů

Načítání argumentů z příkazového prostředí je zajištěno pomocí funkce getopt. Funkce `executeArgs` tyto argumenty uloží, otestuje, jestli vstupní soubory existují a případně při neexistujícím souboru nebo chybné kombinaci argumentů ukončí skript.

2.2 Hledání testů

Na základě argumentu `recursive` probíhá hledání testů pouze v lokální složce, nebo i ve všech podadresářích. Rekurzivní prohledávání je zajištěno pomocí třídy RecursiveDirectoryIterator, RecursiveIteratorIterator a RegexIterator. Funkce `findTests` hledá všechny soubory s příponou src, které značí zdrojové soubory pro skripty interpret a parse. Pro nalezený zdrojový soubor jsou také zkontrolovány náležité soubory pro vstup, výstup a návratový kód. Jestliže tyto soubory neexistují, jsou vygenerovány skriptem. Seznam všech nalezených testů je uložen do pole, které je poté předáno ke zpracování.

2.3 Testování skriptů

Nejprve je vytvořen dočasný soubor pro výstup skriptů, který poté bude použit k porovnání výsledků s referenčním výstupem. Dále na základě argumentů parse-only a int-only je spuštěn odpovídající skript. Pokud není ani jeden z těch argumentů zadán, je spuštěn jak skript pro interpret, tak i parser.

Skripty jsou spouštěny pomocí příkazu exec se soubory s příponami `in` a `src` jako vstup. Výstup těchto skriptů je uložen do dočasně vytvořeného souboru na začátku testování. Je porovnán návratový kód těchto skriptů a referenčního návratového kódu. Jestliže se návratové kódy nerovnají, je aktuální test ukončen jako neúspěšný. Pokud se návratové kódy shodují a jsou rozdílné od 0, je test ukončen jako úspěšný. Pokud se návratový kod shoduje a je roven 0, jsou porovnány také výstupy.

Při testování parse-only je pro porovnání výstupu použit nástroj `JExamXML`, jinak je použit nástroj `diff`. Pokud se výstupy nerovnají, je test vyhodnocen jako neúspěšný. Dočasný soubor je odstraněn na konci každého testu nebo při detekované chybě.

2.4 Generování HTML

Po dokončení testování nastává generování HTML. Přehled testů obsahuje základní informace jako cestu k parseru, cestu k interpretu, typ testování, čas testu a prohledávaný adresář s testy. Úspěšné a neúspěšné testy jsou rozděleny do vlastnich tabulek. Tabulka obsahuje informace o testech jako cesta k souboru, návratové kódy a případně typ chyby.