Implementační dokumentace k 1. úloze do IPP 2019/2020
Jméno a příjmení: Jiří Křištof
Login: xkrist22

1 Úvod

Úkolem druhého projektu do předmětu IPP je vytvoření skriptu `interpret.py`, který je určený k interpretaci programu reprezentovaného XML kódem generovaným z jazyka IPPcode20, a skriptu `test.php`, testovacího frameworku pro postupnou aplikaci parse.php a interpret.py.

2 Skript interpret.py

Skript interpret.py využívá funkce obsažené v adresáři `int_lib`. Při spuštění jsou zpracovány vstupní parametry pomocí funkcí ze souboru `xml.py` a ze souborů uvedených v parametrech, případně ze standardního vstupu, jsou získány vstupy programu a XML reprezentace programu. Následně je řízení předáno instanci třídy `interpreter`. Třída je implementována v souboru `interpreter.py`.

2.1 Soubor `arg_check.py`

Soubor obsahuje funkci `arg_check`, která zpracovává parametry předané skriptu při volání z příkazového řádku. Kontroluje správnost kombinací parametrů. V případě využití parametru nastavující zdroj XML kódu a vstupu jsou tyto soubory otevřeny a jejich obsah je vložen do k tomu proměnných.

Druhou funkcí je funkce `print_help` zobrazující nápovědu při využití parametru `--help`.

2.2 Soubor `xml.py`

Soubor obsahuje funkce `load_xml_code` a `replace_escape`. První funkce s pomocí modulu `xml.etree.cElementTree` vytvoří objekt obsahující XML kód a zároveň zkontroluje, zdali je XML kód dobře formátovaný. Po načtení funkce uspořádá algoritmem bubble sort jednotlivé instrukce podle atributu `order`. Skript očekává, že XML kód je již seřazen (např. skript `parse.php` vytváří již seřazenou posloupnost), z tohoto důvodu je řazení implementováno algoritmem bubble sort.

Druhá funkce v souboru `xml.py` je funkce `replace_escape`, která je implementací konečného automatu nahrazující escape sekvence v XML kódu za znaky tímto kódem reprezentované.

2.3 Soubor `hash_table.py`

V souboru je implementována třída `hash_table`. Instance třídy ukládá data pomocí metody `insert` do slovníku, přičemž ukládá vždy dvojici identifikátor a hodnotu. Třída implementuje i další metody pro aktualizaci, mazání, hledání a čtení dat.

2.4 Soubor `stack.py`

Soubor obsahuje implementaci třídy `stack`. Instance této třídy umožňuje ukládat data jako do ADT zásobník. Implementuje veškeré potřebné metody, které umožňují vkládat data na vrchol, odebírat jej z vrcholu a další.

2.5 Soubor `interpreter.py`

Soubor obsahuje implementaci třídy `interpreter`. Instance třídy ukládá privátně tato data:

- `__global_frame`: instance třídy `hash_table` reprezentující globální rámec proměnných
- `__temp_frame`: pokud je existuje dočasný rámec, pak obsahuje instanci třídy `hash_table`, jinak obsahuje hodnotu `None`
- `__frame_stack`: obsahuje instanci třídy `stack` ukládající dočasné rámce; lokálním rámcem je rámec získaný pomocí metody `top`
- `__call_stack`: obsahuje instanci třídy `stack` reprezentující zásobník volání
- `__labels`: instance třídy `hash_table` ukládající návěští

- __data_stack: instance třídy stack reprezentující datový zásobník

Třída implementuje metody pro validaci XML kódu a pro vlastní zpracování instrukce. Pomocí metody check_xml_structure je kontrolována struktura kódu. Metoda check_arg_count vrací počet argumentů instrukce. Pro kontrolu typů je využívána metoda check_types. Metoda exists zjišťuje, zdali existuje určitá proměnná předaná parametrem. Metoda nepřímo kontroluje i existenci rámce. Hodnota proměnné či její typ je možné získat pomocí metody get_var_value, respektive get_var_type. Před voláním této metody je vždy nutné kontrolovat, zdali proměnná existuje.

Metoda interpreting prochází XML kód a podle hodnoty atributu opcode jednotlivých elementů instruction je volána metoda, která interpretuje danou instrukci. Při prvním průchodu kódem jsou načteny veškeré návěští, při druhém průchodu kódem jsou zpracovávány ostatní instrukce.

3 Skript test.php

Skript test.php po spuštění kontroluje parametry předané z příkazového řádku. Po tomto spouští funkci check_parser, pokud má skript testovat pouze skript parse.php, funkci check_interpreter, pokud má testovat pouze skript interpret.py, nebo funkci check_app, pokud má být testována celá postupná aplikace. Tyto funkce jsou implementovány v souborech v adresáři test_lib.

3.1 Kontrola skriptu parse.php

Funkce pro testování parse.php je implementována v souboru parse_check.php. Při prvním volání je vytvořena instance třídy html_gen, která slouží pro vytváření výstupního HTML dokumentu. Následně funkce iteruje nad každým souborem či adresářem ve složce obsahující testy. Pokud je aktuální položka soubor a má příponu src, pak se skript pokusí nalézt odpovídající soubory s příponami out a rc. Pokud nejsou nalezeny, je vytvořen dočasný prázdný soubor s příponou out a hodnota očekávaného návratového kódu je nastavena na 0. Skript parse.php provede překlad kódu ze souboru s příponou src a s pomocí java aplikace jexamxml je generovaný výstup porovnán s výstupem uvedeným v souboru s příponou out. Pokud jexamxml nenalezne chybu, pak je do výstupního HTML kódu vložena informace o splněném testu. Jinak je vložena informace o nesplněném testu.

3.2 Kontrola skriptu interpret.py

Obdobně jako při kontrole skriptu parse.php je iterováno nad položkami adresáře obsahujícího jednotlivé testy (případně jsou rekurzivně procházeny podadresáře). Stejně jako v předešlé části se pokusíme získat soubory s příponami src, out a rc. Navíc hledáme soubor s příponou in. Pokud není některý ze souborů s příponou out, rc či in nalezen, je místo obsahu souboru využívána implicitní hodnota, tedy prázdný řetězec, 0 a prázdný dočasný soubor postupně. Po spuštění skriptu interpret.py je porovnána získaná návratová hodnota a hodnota očekávaná. Pokud jsou tyto hodnoty shodné a různé od nuly zároveň, pak je test považován za splněný. Pokud jsou návratové kódy získané a očekávané shodné, pak musí být shodný i navrácený a očekávaný výstup programu, aby byl test považován za splněný. Jinak skript testem neprošel.

3.3 Kontrola celé postupné aplikace

Postup je obdobný jako při kontrole skriptu interpret.py. Kód uvedený v souboru s příponou src je první přeložen pomocí skriptu parse.php. Pokud tento skript navrátí hodnotu 0, pak je jeho výstup použit jako vstup pro skript interpret.py. Dále jsou návratové hodnoty porovnávány stejně jako při testování samotného skriptu interpret.py.

Pokud již skript parse.php nalezne chybu a navrátí návratový kód různý od 0, pak kontrolujeme pouze zda-li jsou shodné získané a očekávané návratové kódy. Podle toho poté generujeme do výstupního HTML kódu informaci o projití či neprojití testu.

3.4 Soubor html_gen

Tento soubor obsahuje implementaci třídy, jejíž instance umožňuje generovat, ukládat a vypsat HTML kód na standardní výstup. Při instanciaci je buď navrácen nový objekt této třídy, pokud dosud žádný neexistuje, nebo je vrácen již existující objekt – je využit návrhový vzor jedináček. Využití tohoto návrhového vzoru umožňuje generovat výstupní HTML i z rekurzivně zanořených funkcí stále do jediné instance obsahující výsedky testů z každého volání funkce pro testování.