Implementační dokumentace ke 2. úloze do IPP 2018/2019
Jméno a příjmení: Ondřej Studnička
Login: xstudn00

Skript interpret.py:

Úkolem bylo vytvořit skript interpret.py, který provede lexikální analýzu, syntaktickou analýzu, sémantickou analýzu a případnou interpretaci vstupního XML (XML reprezentuje IPPcode19). Skript je vypracován v jazyce Python, konkrétně ve verzi 3.6.

Implementace:

V první řadě bylo třeba zajistit správné spuštění skriptu uživatelem. Skript lze spustit následujícími způsoby: „`python3.6 interpret.py --source=vstupni_soubor --input=vstupni_soubor`“ / „`python3.6 interpret.py --help`“. Pokud je skript spuštěn, tak jako je ukázáno v první variantě, tak může jeden ze vstupních argumentů chybět (v takovém případě je vstup do skriptu načítán ze standardního vstupu), nikdy však nesmí chybět oba argumenty. Druhý z uvedených způsobů vypíše uživateli nápovědu k používání. Jiné spuštění skript ukončí s chybou 10.

Po správném spuštění se do proměnné `program` načtou pomocí knihovny `xml.etree.ElementTree` vstupní informace. Tyto vstupní informace jsou v několika cyklech podrobovány lexikálním a syntaktickým kontrolám. Namátkou lze zmínit několik z nich: kontrola přítomnosti tagu `program`, kontrola atributů tohoto tagu, kontrola jestli u jednotlivé instrukce mají atributy `order` a `opcode`, kontrola jestli je zadaná instrukce platná, kontrola jednotlivých argumentů instrukce, kontrola pomocí regulárních výrazů…

Po těchto kontrolách se provádí seřazování instrukcí podle `opcode` (pokud dvě instrukce mají uvedeno stejné pořadí, tak se skript ukončí s chybou). Po seřazení podle instrukcí probíhá řazení jednotlivých argumentů instrukcí – od 1 do 3 (pokud má argument vyšší číslo, tak je skript ukončen s chybou).

Jakmile je seřazení dokončeno, tak již víme, že lexikální a syntaktická kontrola proběhla bez problému. Potřebná data (`opcode`, `order`, `type`, text argumentů) pro sémantickou analýzu jsou ukládána do pole `all_data`. Tomuto poli je následně vygenerována záloha (kvůli instrukcím `CALL` a `RETURN`).

Před interpretací `interpret.py` hledá v poli `all_data` všechna návěští a kontroluje, jestli se uživatel nesnaží o redefinici návěští.

Interpretace využívá funkce `defvar_function` (pro definici funkce), `check_frame_function` (pro zjištění jestli je proměnná definována), `set_var_value` (pro nastavení hodnoty proměnné), `get_value` (pro získání hodnoty proměnné), `get_type` (pro získání typu proměnné), `pop_all` (pro kompletní vyčištění pole).

Internet pracuje s proměnnými `lokalni_ramec, docasny_ramec, pocet_instrukci_read, k, zasobnik, zasobnik_volani, zasobnik_ramcu`.

Samotná interpretace kódu je uskutečněna pomocí funkce `interpretace`, která za argument bere pole `all_data`. Z pole se postupně v cyklu vybírají jednotlivé instrukce (uloženo v `current_instruction`). Proměnná prochází kaskádou `if-elif-else` a hledá pokračování kódu. Po jeho nalezení se provedou příslušné sémantické kontroly (kontrola jestli je proměnná definována, kontrola jestli je používán správný typ, …), vykonají se operace spjaté s touto instrukcí (např. `ADD` – sečtení) a v drtivé většině případů se výsledek instrukce uloží do zadané proměnné (v ostatních případech se může například vypisovat na standardní výstup (`WRITE`), …).

Skript test.php:

Úkolem bylo vytvořit skript test.php, který otestuje funkčnost předchozích dvou skriptů (`parse.php`, `interpret.py`) a o výsledcích testů informuje uživatele pomocí jednoduché webové stránky v HTML5. Skript je vypracován v jazyce PHP, konkrétně ve verzi 7.3.

Implementace:

V první řadě bylo třeba zajistit správné spuštění skriptu uživatelem. Skript lze spustit následujícími způsoby: „`php7.3 test.php --help`“ / „`php7.3 test.php prepinace`“. Pokud je skript spuštěn prvním způsobem, tak bude uživateli na standardní výstup vytisknuto krátké shrnutí o tomto skriptu. Pokud je skript spuštěn druhým způsobem, tak uživatel doplní namísto „`prepinace`“ některý z parametrů: `--directory=, --recursive, --parse-script=, --int-script=, --parse-only, --int-only`. Některé kombinace parametrů jsou však zakázány (např.: `int-only` a `parse-only`). Osobně doporučuji ještě za přepínače přidat „ > `report.html`“, aby pak uživatel mohl rovnou zobrazit výsledky testu v prohlížeči.

Skript test.php načte zadané argumenty a na jejich základě rozhodne o svém pokračování. Pokud je zadán jeden z argumentů `--directory=/--parse-script=/--int-script=`, tak se prvně otestuje, jestli cesta k zadaným souborům existuje. Pokud ano, tak bude skript pracovat se zadanými složkami/soubory. Jinak se použije výchozí nastavení. Při zadání `--recursive` bude skript prohledávat složku postupným zanořováním. Přepínače `--parse-only` a `--int-only` určují, který skript bude otestován. Pokud není zadán ani jeden z nich, tak bude otestována funkčnost obou skriptů.

Po nastavení chování skriptu bude `test.php` hledat soubory, se kterými bude pracovat (hledá přípony `.src`). Pokud žádné nenalezne, tak bude uživatel informován ve výsledné webové stránce.

V případě nalezení souborů se skript pokusí najít pro každý `.src` soubor ještě soubory se stejným názvem, ale s příponami `.rc, .out, .in`. Pokud takové soubory nenalezne, tak je vygeneruje jako prázdné (pouze do `.rc` vepíše 0).

Následně jsou volány funkce `parse_test, int_test, test_both` (podle přepínačů). A jejich výstupy jsou ukládány do pole pro výpis `$test_info`.

Funkce `parse_test` vykoná pro zadaný vstupní soubor otestování pomocí skriptu `parse.php`. Příkazem `exec` spustí skript a do určených proměnných vloží návratovou hodnotu skriptu a jeho případný výstup. Pak následují možnosti pro vyhodnocení testu. Pokud skript vrátí 0, tak je toto číslo porovnáváno s očekávanou hodnotou. Pokud je i ta 0, tak se spustí `JExamXML`, pro porovnání výstupu z `parse.php` a očekávaného výstupu. Pokud se tyto hodnoty rovnají, tak test uspěl. V opačném případě ne. Podobně `test.php` vyhodnocuje i ostatní možné stavy.

Funkce `int_test` vykoná pro zadaný vstupní soubor otestování pomocí skriptu `interpret.py`. Příkazem `exec` spustí skript a do určených proměnných vloží návratovou hodnotu skriptu a jeho případný výstup. Pak následují možnosti pro vyhodnocení testu. Pokud skript vrátí 0, tak je toto číslo porovnáváno s očekávanou hodnotou. Pokud je i ta 0, tak se spustí `diff`, pro porovnání výstupu z `interpret.py` a očekávaného výstupu. Pokud `diff` vrátí hodnotu 0, tak test uspěl. V opačném případě ne. Podobně `test.php` vyhodnocuje i ostatní možné stavy.

Funkce `test_both` vykoná pro zadaný vstupní soubor otestování pomocí skriptu obou skriptů. Příkazem `exec` spustí `parse.php` a do určených proměnných vloží návratovou hodnotu skriptu a jeho případný výstup. Pokud skript vrátí 0, tak je volána funkce `int_test`, která provede stejný test, jako je popsán výše. Vyhodnotí se úspěšnost testu.

Nakonec `test.php` pomocí příkazu `echo` vypíše `HTML5` kód, ve kterém jsou uvedeny informace o vykonaných testech. Pozadí stránky se mění podle toho, jestli test uspěl/neuspěl/neproběhl.