Implementační dokumentace k 2. úloze do IPP 2018/2019

Jméno a příjmení: Daniel Štěpán
Login: xstepa60

Skript interpret.py

Podstatou skriptu `interpret.py` je načtení programu reprezentovaného v XML a tento program je dále interpretován.

Moduly

Kód je z hlediska čitelnosti kódu a principů objektově orientovaného programování rozdělen do několika (šesti) spolupracujících modulů a to: `arg_parse_helper.py`, `frame.py`, `instruction.py`, `interpret.py`, `stack.py`, `xml_parse_helper.py`. Každý modul má na starost nějakou část interpretace a výsledek předává dle potřeby ostatním modulům.

Přijímané parametry

Ihned po spuštění skriptu dochází ke kontrole a zpracování parametrů příkazové řádky. Tato kontrola probíhá v rámci modulu `arg_parse_helper.py`. Třída ArgParseHelper zde využívá služeb knihovny argparse. Kontrolovanými parametry jsou `--source` - nastavením tohoto parametru určíme skriptu vstupní soubor s kódem v XML reprezentaci. Pokud tento parametr není nastaven, interpret očekává vstup kódu na standardní vstup.
Dalším parametrem je `--input` - nastavením tohoto parametru určíme skriptu soubor obsahující vstupy důležité pro interpretaci zdrojového kódu (například při přijetí instrukce s operačním kódem `READ`).
Posledním hlídaným parametrem je `--help` - jedná se o výpis nápovědy k programu. Vždy musí být zadán alespoň jeden z parametrů `--source` nebo `--input`, jinak se jedná o chybu. Stejně tak se parametr `--help` nesmí kombinovat s žádnými jinými parametry.

Parsování vstupního XML

Parsování vstupního XML zajišťuje modul `xml_parse_helper.py`, metoda load_xml() třídy XMLParseHelper využívá ke zpracování knihovnu xml.etree.ElementTree.
Po načtení do proměnné a parsování se prochází jednotlivé elementy XML a načtené instrukce a argumenty se převádí do interní reprezentace objektů třídy Instruction, která je implementována v modulu `instruction.py`.
Při převodu na interní reprezentaci pomocí objektů probíhá zároveň i syntaktická a lexikální analýza vstupního XML kódu. U každé instrukce se hlídá i požadovaný počet parametrů.

Paměťový model interpretu

Interpret využívá v podstatě tři druhy rámců: `lokální rámce`, `dočasné rámce` a `globální rámec`. Globální rámec (GF) slouží pro ukládání globálních proměnných a jeho implementace je zajištěna pomocí slovníku. Každá proměnná v rámci musí být nejprve definovaná pomocí instrukce `DEFVAR` a její typ se určí až s prvním přiřazením nějaké hodnoty.

Interpretace kódu

Po parsování XML a převodu instrukcí na interní reprezentaci je výsledkem pole se seřazenými instrukcemi (dle parametru `order`). Toto pole se dále prochází pomocí cyklu while. Cyklus while byl vybrán proto, protože nám umožňuje pracovat a měnit hodnotu řídící proměnné, kterou zároveň používáme jako index do pole s instrukcemi. Toto nám umožnilo implementovat skokové instrukce způsobem, že zjistíme, na jaké pozici je hledaný label a změníme hodnotu řídící proměnné na tuto pozici.
Uvnitř cyklu je dlouhá if-elif-else konstrukce, kde na základě operačního kódu právě zpracovávané instrukce provádíme patřičné operace (práce s pamětí, aritmetické či logické operace, výpis, skoky atd.).

Využití návrhových vzorů

Při zpracování instrukce `READ`, čtení ze souboru, je využit návrhový vzor Singleton. Ten nám umožnil při prvním volání otevřít daný soubor a při dalších voláním vrací referenci na otevřený soubor. Implementace Singletonu je v třídě UserFile v modulu `arg_parse_helper.py`.

Skript test.php

Podstatou skriptu `test.php` je automatické testování aplikací `parse.php` a `interpret.py`.

Přijímané parametry

Ihned po spuštění skriptu dochází ke kontrole a zpracování parametrů příkazové řádky. Ke zpracování se používá vestavěná knihovna `getopt`.
Kontrolovanými parametry jsou `--help` - tento parametr vypíše nápovědu k programu a zároveň nesmí být kombinován s žádným jiným parametrem.
Další parametr je `--directory=path` - path je cesta k adresáři, ve kterém skript vyhledá testy. Pokud tento parametr chybí, tak skript hledá testy v aktuálním adresáři.
Dalším parametrem je `--recursive` - nastavením tohoto parametru docílíme, že skript hledá testy i v podadresářích zadaného `--directory`, případně podadresářích aktuálního adresáře.
Parametr `--parse-script=file` - určíme soubor s PHP skriptem pro analýzu zdrojového kódu. Při nezadání parametru je použita defaultní hodnota `parse.php` v aktuálním adresáři.
Parametr `--parse-script=file` - určíme soubor s Python skriptem pro interpretaci kódu v XML reprezentaci. Při nezadání parametru je použita defaultní hodnota `interpret.py` v aktuálním adresáři.
Parametry `--parse-only` a `--int-only` testují pouze skript pro parsování zdrojového kódu IPPcode19, respektive skript pro interpret XML reprezentace IPPcode19, přičemž se nesmí kombinovat.

Průběh skriptu po spuštění

Po spuštění proběhne zpracování parametrů a ověření formálních požadavků na parametry (viz výše). Inicializuje se sada proměnných buď defaultními hodnotami nebo hodnotami předaných přes parametry při spuštění. Proběhne kontrola, zda zadaný adresář existuje a je validní.
Poté se spustí stěžejní funkce skriptu `test.php` a to funkce `test($directory, $parse_script, $parse_only, $int_script, $int_only, $recursive)`.

Funkce test(...)

Na začátku funkce se pomocí cyklu `foreach(scandir($directory) as $file)` prochází všechny soubory v adresáři, pokud se narazí na adresář a zároveň byl zadán při spuštění parametr `--recursive`, tak se na daný adresář volá rekurzivně funkce `test(...)`.
Jediný soubor týkající se testů, který musí být zadaný, je soubor s koncovkou `.src`, obsahující zdrojový kód, ať již v XML reprezentaci IPPcode19 nebo v IPPcode19. Skript si tedy prvně zjistí a vypíše všechny *.src soubory. Dále se podívá, jestli k tomuto souboru existuje soubor s koncovkou `.in`, `.out` a `.rc`. Pokud neexistuje, vytvoří je jako prázdné, respektive s defaultní hodnotou 0 v případě *.rc.
Následně, dle zadaných parametrů, se provede jedna z funkcí parse_only(...), int_only(...) nebo both(...).

Společná implementace funkcí parse_only(...), int_only(...), both(...)

Vestavěnou funkcí `shell_exec()` se zavolá parsovací skript a vloží se mu *.src soubor na standardní vstup a výstup se uloží do dočasného `output_temp` souboru. Tato funkce se zavolá ještě jednou, tentokrát je však výstup přesměrován do `/dev/null 2>&1` a následně se volá příkaz `echo $?`. Takto uložíme návratovou hodnotu do proměnné. Nyní se porovnají skutečné výstupy a návratová hodnota s očekávanými hodnotami.

Generování výstupního HTML

Na závěr je vygenerována přehledná barevná dvousloupcová tabulka s testy, které prošly a které neprošly.