Implementačná dokumentácia k 2. úlohe v IPP 2020/2021
Meno a priezvisko: Jakub Duda
Login: xdudaj02

Interpret.py

1. Popis

Skript *interpret.py* načíta zo štandardného vstupu alebo zo súboru XML reprezentáciu zdrojového kódu v jazyku IPPcode21 a tiež vstup pre tento program. Následne vykoná lexikálnu, syntaktickú a sémantickú kontrolu tohto programu a následne vykoná jeho interpretáciu, resp. ukončí sa so správnym chybovým návratovým kódom.

Na spracovanie XML reprezentácie skript využíva knižnicu *xml.etree.ElementTree*.

2. Spustenie

Skript *interpret.py* prijíma tri rôzne argumenty príkazového riadku. Na zobrazenie nápovedy je možné využiť nekombinovateľný argument: *--help*. Pre normálnu funkcionalitu, je potrebné spúšťať skript s argumentmi *--input=file* a *--source=file*, ktorými sa špecifikuje súbor so zdrojovým kódom a súbor so vstupom pre tento program. Je podporované aj využitie len jedného z týchto argumentov, pričom vtedy bude druhý z týchto vstupov očakávaný na štandardnom vstupe.

3. Implementácia

Na začiatku sa pomocou spomínanej knižnice prevedie lexikálna a syntaktická kontrolu vstupného súboru. Následne prebieha syntaktická kontrola vzhľadom na pravidlá jazyka IPPcode21.

Po overení správnosti vstupného súboru sa pre jednoduchší prístup pri interpretácii kódu zoradia inštrukcie tohto kódu do správneho poradia a potom sa tieto uložia do špeciálnej dátovej štruktúry `InstructionArray`. Táto štruktúra je tvorená rovnomennou triedou, ktorá obsahuje slovník objektov triedy `Instruction`, kde tieto objekty potom obsahujú slovníky objektov triedy `Argument`.

Pred samotnou interpretáciou kódu sa taktiež inicializujú objekty ďalších tried. Premenná `frames` je objektom triedy `FrameStack`, ktorá zahŕňa reprezentáciu dát v programe IPPcode21, a teda globálny rámec, dočasný rámec a zásobník lokálnych rámcov. Pre tieto je využitá trieda `Frame`, ktorá obsahuje slovník objektov triedy `Variable`, ktoré slúžia na uloženie a prácu s jednotlivými premennými programu IPPcode21. Ďalej je inicializovaný objekt triedy `Stack`, ktorá reprezentuje zásobník využívaný v IPPcode21 a trieda `InstructionRegister`, ktorá obsahuje slovník návestí, pole návratových adries a taktiež počítadlo, ktoré pri interpretácií značí číslo aktuálnej inštrukcie.

Pred hlavným behom interpretácie ešte prebehne jeden beh, kedy skript hľadá len inštrukcie *label* a do príslušného slovníka ukladá ich adresy.

Hlavný beh interpretácie je rovnako ako pre-run realizovaný pomocou konštrukcie *while*, kde sa na konci každého cyklu inkrementuje číslo aktuálnej inštrukcie. Jednotlivé inštrukcie sú „odchytávané“ pomocou konštrukcií *if/elif*. Pri interpretácií vstupného kódu skript využíva vhodné funkcie alebo metódy, v ktorých sú ošetrené rôzne chyby ako neexistencia rámca, neexistencia premennej, neexistencia návestia, pokus o redefiníciu premennej, pokus o čítanie z nedefinovanej premennej alebo prázdnej pamäťovej štruktúry a ďalšie.

Zároveň prebieha sémantická analýza vstupného kódu, ktorá odhaľuje chyby použitia argumentov nesprávnych dátových typov pri konkrétnej inštrukcií alebo použitie nesprávnych hodnôt ako napríklad delenie nulou.

Implementačná dokumentácia k 2. úlohe v IPP 2020/2021
Meno a priezvisko: Jakub Duda
Login: xdudaj02

Test.php

1. Popis

Skript test.php testuje funkcionalitu skriptov parser.php a interpret.py a na štandardný výstup generuje súhrn výsledkov v HTML 5.

2. Spustenie

Skript *test.php* pracuje s viacerými nepovinnými argumentmi, ktorých využitím je možné zmeniť niektoré default hodnoty. Na zobrazenie nápovedy sa dá využiť nekombinovateľný argument: *--help*. Pomocou *--directory* je možné špecifikovať cestu k adresáru obsahujúcemu testy, ktoré má skript využiť. Využitie argumentu *--recursive* spôsobí rekurzívne vyhľadanie testov vo všetkých podadresároch daného adresára. Pomocou parametrov *--int-only* a *--parser-only* je možné obmedziť testovanie len na jeden konkrétny skript. Argumenty *--parse-script=file* a *--int-script=file* je možné na využitie špecifikácie ciest ku testovaným skriptom. Argumenty *--jexamxml=file* a *--jexamcfg =file* je možné na využitie špecifikácie ciest ku súborom využívaného nástroja JExamXML.

3. Implementácia

Na spracovanie argumentov je využitá vlastná funkcia `get_from_arg()`, ktorá vracia hodnotu zadanú cez argument alebo ak nebola zadaná vráti defaultú hodnotu.

Na začiatok sa do poľa uložia všetky testovacie súbory so zdrojovým kódom. Ďalej je pre jednoduchšiu orientáciu kód rozdelený na tri časti podľa testovaných skriptov. V každej sa pomocou konštrukcie *foreach* prechádza každá skupina testovacích súborov (každý test). Pri neexistencií *.in* a *.out* súborov sa tieto korektne vytvoria a načíta sa očakávaný výstupný kód. Následne sa pomocou funkcie *exec()* spustí testovací príkaz. Pri odlišnom očakávanom a získanom výstupnom kóde sa test zaznamená ako neúspešný. Ak bol výstupný kód správny a bol nulový, teda program prebehol správne, prechádza sa na testovanie správnosti výstupu. Pri testovaní skriptu *parser.php* sa tu využije nástroj JExamXML a pri testovaní skriptu *interpret.py* sa používa nástroj diff. Na základe rovnakosti očakávaného výstupu a získaného výstupu sa uloží výsledok testu. Pri testovaní obidvoch skriptov sa nerieši korektnosť výstupného kódu a výstupu skriptu *parser.php*, ale len na základe nulovosti výstupného kódu sa buď spustí alebo nespustí ďalší skript.

Na záver sa generuje súhrn výsledkov testovania vo formáte HTML 5. V premenných je uložený začiatok aj koniec tohto výstupu a pomocou konštrukcie *foreach* sa znovu prechádza poľom súborov a poľom ich výstupov a pre každý sa generuje adekvátny HTML výstup, ktorý sa konkatenuje k už definovaným častiam HTML výstupu. Na toto sú využité funkcie `html_add_dir()` a `html_add_test()`. V prvej menovanej sa generuje HTML výstup pre názov adresára a v druhej HTML výstup pre vykonaný test, pričom je rozdielny pre úspešné a neúspešné testy. Na koniec každého adresára sa k jeho názvu v HTML výstupe dopíše počet úspešných a počet všetkých testov v tomto adresári. HTML výstup sa skladá z hlavičky, zhrnutia počtov všetkých , úspešných a neúspešných testov a následne vymenovania jednotlivých testov s ich výsledkami zoradených podľa adresárov, v ktorých sa nachádzajú.