Implementačná dokumentácia k 2. úlohe do IPP 2019/2020
Meno a priezvisko: Dominik Boboš
Login: xbobos00

1 Skript interpret.py

Skript *interpret.py* uskutočňuje preklad vstupného XML súboru s jazykom IPPcode20 a následne kód interpretuje s využitím štandardného vstupu a výstupu. Skript je implementovaný v troch moduloch `interpret.py`, `ippcode_bank.py` a `ippcode_dependencies.py`.

1.1 Modul interpret.py

Modul `interpret.py` je spúšťaný ako prvý. Na začiatku skriptu sa parsujú vstupné argumenty z príkazového riadku. Trieda **`XMLparser`** obsahuje 2 metódy *checkXML(source)* a *checkBody(source)*. V týchto metódach sa pomocou knižnice *xml.etree.ElementTree* parsuje vstupný XML súbor s IPPcode20 v parametri *source*. Overuje sa či je „well-formed“, správnosť hlavičky a ostatné formálne záležitosti. Funkcia *checkBody(source)* vracia list inštrukcií s danými argumentami - `instrList`, spoločne vracia aj správne poradie jednotlivých inštrukcí podľa *order* v premennej `orderList`. Tieto parametre sú esenciálne pre modul `ippcode_bank.py`.

1.2 Modul ippcode_bank.py

Tento modul je „mozog“ celého skriptu, nakoľko sa v ňom overujú konkrétne inštrukcie a spúšťa potrebné funkcie na interpretáciu kódu. Obsahuje triedu **`Interpret`** s početnými metódami. Najdôležitejšie sú metódy *checkInstr((self, order, xmlInstr)* a *interpret(self, inputFile, inputBool)*.

V *checkInstr* sa uskutočňuje prvý prechod. Prebieha v cykle while a skončí pri prechode všetkými inštrukciami. Kontroluje sa či každá inštrukcia dostala očakávaný počet argumentov, definujú sa návesti do *self.labels*, overuje sa validita hodnôt symbolov či názvov premenných, alebo návestí, metóda vytvára list inštrukcií *self.instructions* v tvare **[[instr1, [[arg1(type), arg1(value)][..., ...]]], [instr2 [[..., ...]]], [...[[...]]]]**, kde sú inštrukcie zoradené v správnom poradí.

Samotná interpretácia kódu sa uskutočňuje až druhým prechodom v metóde *interpret(self, inputFile, inputBool)*. Prechod sa opäť uskutočňuje cyklom while, kde riadiaca premenná *current* označuje momentálnu pozíciu v kóde a cyklus sa ukončí pokiaľ *current* dosiahne väčšiu hodnotu ako je počet inštrukcií. Parametre funkcie *inputFile* a *inputBool* slúžia inštrukcií READ k prípadnému načítaniu vstupu zo súboru. Táto metóda slúži predovšetkým ako switch a prepája skript s ďalším modulom `ippcode_dependencies.py`.

1.3 Modul ippcode_dependencies.py

Tento modul spracováva jednotlivé inštrukcie jazyka IPPcode20. Uchováva závislosti ako je zásobník pre jednotlivé rámce, zoznamy pre jednotlivé rámce, zásobník pre inštrukcie PUSHS, POPS a pomocná premená pre inštrukciu READ, kde sa uchovávajú jednotlivé riadky zo súboru v argumente `--input`.

Dôležité metódy v triede **`Dependencies`** sú *foundVar(self, var, symbBool)* a *setTypeValue(self, frame, index, typeVar, value)*. Prvá vyhľadá premennú s názvom arg[1] v rámci uloženom vo var[0], druhá vloží hodnotu *value* a typ *typeVar* do premennej na indexe *index*. Tieto funkcie sa využívajú vo väčšine zvyšných funkcií tohoto modulu na vykonanie jednotlivých inštrukcií jazyka.

1.4 Error handling

Správa chybových hlásení je uskutočňovaná pomocou jednotlivých tried nakonci modulu `ippcode_bank.py`, kde pre každý návratový kód chyby je vytvorená jedna trieda. Pri zachytávaní chyby je s výnimkou zavolaná žiadaná trieda s vhodnou chybovou hláškou a následne ukončuje skript s danou návratovou hodnotou.

1.5 Implementované rozšírenia

V skripte sú implementované rožšírenia **`FLOAT, STACK, STATI`**. Tým, že sú implementované aj FLOAT aj STACK, tak sa oproti inštrukciám v zadaní pre rozšírenie pridal aj DIVS, INT2FLOATS, FLOAT2INTS (podľa zadania do predmetu IFJ).

V triede `Dependencies` sa implementovaním STACK pridal do metód ďalší argument *stack* typu bool, ktorý je implicitne nastavený na **False**, v prípade, že je **True**, tak sa výsledok riešenia neukladá do premennej *var*, ale funkciou *pushs(self, symb)* sa uloží na zásobník.

Štatistické údaje `--insts` sa zbierajú vo funkcií *interpret(self, inputFile, inputBool)*. Údaj `--vars` sa zbiera vo funkcií *isInitialized(self, var)*, ktorá sa volá v jednotlivých metódach modulu ippcode_dependencies.py. Výsledné zozbierané štatistiky sa vypíšu do súboru predaného v `--stats` v module interpret.py.

2 Skript test.php

Skript *test.php* slúži k automatickému testovaniu skriptov *parse.php* a *interpret.py*.

V hlavnom tele programu prebieha parsovanie vstupných argumentov, ktoré prípadne zmenia implicitné hodnoty globálnych premenných: *$directory, $int_file, $parse_file, $jexamxml_file, $recursive_flag, $parse_only_flag, $int_only_flag*, takže buď zmenia cesty, alebo nastavia prepínač.

Samotné testovanie prebieha vo funkcií `TestFiles($source)`. Argument *$source* je `.src` súbor, vďaka ktorému zisťujeme existenciu ostatných testových súborov `.out`, `.rc`, `.in`. Následne spustíme najskôr skript `parse.php` pomocou príkazu `exec` a jeho výstup smeruje do dočasného súboru `temp_output`. V prípade, že návratová hodnota z parse.php bola rovná 0, tak pokračujeme so spúšťaním skriptu `interpret.py`, do ktorého ako vstup používame dočasný súbor `temp_output` a súbor s príponou `.in` a výstup smeruje do dočasného súboru `temp_output2`. Následne sa porovnajú výstupné návratové hodnoty s predpokladanou hodnotou. V prípade, že sa rovnajú, tak sa programom `diff` s prepínačom `-q` porovná súbor `temp_output2` s `*.out` súborom.

V prípade, že je zapnutý prepínač `--parse-only`, tak na začiatku sa pomocou funkcie `isXmlStructureValid($file)` zistí či sa náhodu nepúšťa k testovaniu súbor s XML reprezentáciou, v takom prípade sa test preskočí, inak sa pokračuje ďalej a `temp_output` sa pomocou programu `jexamxml.jar` porovná s `*.out` súborom.

Prepínač `--recursive` sa implementuje pomocou `RecursiveIteratorIterator`, súbory, ktoré takto vyhľadá potom posiela do funkcie *TestFiles($dir_source)*

Generovanie výsledného html súboru prebieha vo funkcií `HTMLgen($filename, $exp_rc, $test_rc, $result, $test_count)`. Tá sa volá pri každom teste a postupne vkladá do html tabuľky informácie o čísle testu, o názve testu, úspešnosť, žiadaný návratový kód, kód z výsledního testu. Po vykonaní testov celú túto html reprezentáciu vypíše na štandardný výstup.