Implementačná dokumentácia k 2. úlohe do IPP 2020/2021

Meno a priezvisko: Ján Homola

Login: xhomol27

1 Interpret XML reprezentácie kódu (interpret.py)

1.1 Kontrola vstupného XML súboru

Na načítanie a následnú kontrolu vstupného XML súboru v jazyku IPPcode21 som použil vstavané metódy z pomocnej knihovne `xml.etree.ElementTree`. Táto kontrola je implementovaná v pomocnej funkcií `parse_xml()`, ktorá kontroluje správne zapísanie hlavičky a ostatné atribúty ako `order` a `opcode`. Ak je daná inštrukcia správne zapísaná uloží sa do listu inštrukcií `instructions` a následne sa tieto inštrukcie zoradia vzostupne pomocou vstavanej funkcie `sorted` na základe atribútu `order`. Ďalej v cykle s podmienkou na začiatku prebieha prvý prechod, ktorý kontroluje jednotlivé inštrukcie podľa zoradenia a to pomocou pomocnej funkcie `check_instruction`, ktorá skontroluje lexikálnu a syntaktickú správnosť zapísania danej inštrukcie ak je táto inštrukcia správne zapísaná vytvorí list inštrukcií `instruc_list`, ktorý uchováva v sebe názov inštrukcie, typ a hodnotu daných argumentov. Počas prvého prechodu sa kontrolujú ešte aj návestia a ukladajú sa do listu `labels`, kde každé návestie obsahuje slovník ktorých v sebe uchováva názov návestia a poradové číslo inštrukcie. Tento list `labels` sa následne používa pri interpretácií skokových inštrukcií.

1.2 Interpretácia kódu

Interpretácia kódu prebieha až počas druhého prechodu, ktorý je tiež implementovaný pomocou cyklu s podmienkou na začiatku , kde interpretácia samotných inštrukcií je riadená počítadlom `order` ak presiahne hodnotu väčšiu ako je počet inštrukcií tak sa cyklus ukonči. Toto počítadlo určuje na ktorej inštrukcií v kóde sa interpret práve nachádza. Na základe počítadla `order` sa z listu `instruc_list` vyberie požadovaná inštrukcia a zavolá sa pomocná funkcia s danými parametrami ktorá túto inštrukciu vykoná. Ak požadovaná inštrukcia je **LABEL** tak táto inštrukcia sa preskočí, pretože bola spracovaná už počas prvého prechodu. Počas interpretácie jednotlivých inštrukcií boli najviac používané pomocné funkcie `get_value()` a `move()`, ktoré boli použité takmer pri každej interpretácii danej inštrukcie. Funkcia `get_value()` získa type a hodnotu konštanty alebo premennej ktoré boli zadané pomocou parametru `symb`. Funkcia `move()` zavolá funkciu `get_value()` a uloží získaný typ a hodnotu do premennej v požadovanom rámci, pokiaľ táto premenná alebo rámec neexistuje dochádza k chybe.

1.3 Implementácia rozšírenia

V programe `interpret.py` som implementoval dva rozšírenia a to **float** a **stack**. Pri rozšírení **float** bolo potrebné pridať prácu s typom **float** v pomocných funkciách ako `arithmetics_instructions()`, ktorá sa zaoberá aritmetickými inštrukciami, vo funkcií `read()`, ktorá načíta hodnotu zo vstupu, a taktiež pri najpoužívanejších inštrukciách `get_value()` a `move()`. Rozšírenie **stack** je implementované pomocou nových pomocných inštrukcií ktoré sa zaoberajú prácou so zásobníkom. Keďže som implementoval aj rozšírenie **float** tak som sa zaoberal aj inštrukciami ako FLOAT2INTS, INT2FLOATS a DIVS.

2 Testovací rámec (test.php)

Tento program slúži na automatizované testovanie programov `parse.php` a `interpret.py`.

2.1 Spracovanie vstupných parametrov

Spracovanie vstupných parametrov programu je kontrolované pomocou funkcie `check_script_param()`, ktorá má dva argumenty a to `$argc` a `$argv`. Táto funkcia kontroluje správnosť zapísania vstupných parametrov a taktiež kontroluje možnosti kombinovania daných vstupných parametrov. Tieto vstupne parametre môžu zmeniť implicitnú hodnotu, a to prostredníctvom nastavania prepínača `$parse-only`, `$int-only` a `$recursive` na hodnotu `true` alebo `false`. Taktiež tieto prepínače môžu zmeniť cestu k daným súborom alebo adresárom.

2.2 Vyhľadanie testov

Vyhľadávanie konkrétnych testov je implementované pomocou externého príkazu `find`, ktorý na základe toho či bol prepínač `--recursive` zapnutý hľadá jednotlivé testy v danom adresári alebo hľadá rekurzívne aj v podadresároch. Následne sa výsledok príkazu `find` uloží do premennej `$file_to_test`.

2.3 Testovanie

Testovanie jednotlivých testov je implementované pomocou cyklu ktorý prechádza cez všetky položky v premennej `$file_to_test` a ak daný súbor má príponu `.src` tak ďalej dochádza ku kontrole či existujú súbory s príponami `.in`, `.out`, `.rc` pokiaľ niektorý z týchto súbor chýba tak sa vytvorí pomocou vstavanej funkcie `fopen()`. Pokiaľ nebol zapnutý ani jeden z prepínačov `--parse-only` a `--int-only` tak pomocou externého príkazu sa spustí program `parse.php` jeho výstup sa uloží do pomocného súboru `parse_out` a jeho návratová hodnota sa uloží do premennej `$return_value`, ak je táto hodnota nulová tak sa pomocou externého príkazu spustí program `interpret.py` a návratovú hodnotu porovná s referenčnou hodnotou daného testu a výstup sa uloží do pomocného súboru `int_out`. Pokiaľ sa tieto hodnoty rovnajú a hodnota nie je nula tak sa potom pomocou pomocnej funkcie `check_int_out()` skontroluje či sa súbor `int_out` rovná referenčnému súboru. Na základe výsledku sa vygeneruje html kód pomocou funkcie `print_html()`. Pokiaľ bol zapnutý prepínač `--parse-only` tak sa porovná výstup uložený v súbore `parse_out` s referenčným súborom s príponou `.out` pomocou pomocnej funkcie `check_parse_out()`, ktorá používa program `jexamxml.jar`. V prípade že bol použití prepínač `--int-only`, tak sa porovná výstup ktorý je uložený v pomocnou súbore `int_out` s referenčným výstupným súborom pomocou funkcie `check_int_out()`, ktorá využíva externý príkaz `diff`.