Implementační dokumentace k 2. úloze do IPP 2018/2019

Jméno a příjmení: Václav Trampeška

Login: xtramp00

Skript interpret.py

Skript se spouští příkazem

```
python3.6 interpret.py [--help] [--source=file] [---input=file]
```

, kde parametr `--help` vypíše na standardní výstup nápovědu, parametr `--source=file` očekává soubor s reprezentací kódu ve formátu XML a parametr `--input=file` očekává soubor se vstupy pro instrukci `read`. Pokud jeden ze dvou posledních parametrů chybí, skript očekává jeho vstup na standardním vstupu. Absence obou parametrů, kombinace parametru `--help` nebo zadání nevalidního parametru vede na chybu 10.

Po kontrole parametrů příkazové řádky zavolá skript funkci `parse_xml_to_lists`, které předá, pokud zadaný parametrem, zdrojový soubor. Funkce vrací interní reprezentaci načtených instrukcí a jejich argumentů. Jako první si funkce pomocí knihovny `xml.etree.ElementTree` načte vstupní XML. Nenačtení vede na chybu 31. Následuje kontrola značek (tagů) a atributů značek jak pro hlavičku vstupu, tak pro instrukce a atributy instrukcí. Před ukládáním informací ještě proběhne převedení hodnot u typů `int` a `bool` a zároveň jejich kontrola. Jakýkoliv neočekáváný vstup vede na chybu 32. Jako poslední jsou argumenty a instrukce uloženy do seznamu na základě jejich pořadí zadaného ve vstupu. Pokud se naleznou instrukce nebo argumenty se stejným pořadím nebo skript se ukončí s chybou 32.

Dále skript zavolá funkci `execute_instructions`, která jako parametry očekává interní reprezentaci instrukcí a vstup pro instrukci `read`. Funkce jako první najde všechny instrukce `label` a uloží název návěstí a jeho pozici do slovníku. Bez tohoto slovníku by nebylo možné ve vstupním programu „skákat“ dopředu, ale pouze zpět. Pokud se již návěstí se stejným jménem ve slovníku nachází, jedná se o chybu 52. Funkce zároveň zkontroluje validitu načtených informací stejně jako u ostatních instrukcí v následujícím cyklu. V následujícím cyklu se prochází postupně načtené instrukce. U každé instrukce zkontroluje za pomocí funkce `check_number_of_arguments` počet argumentů a následně na základě typů argumentů volá pro každý argument jednu z funkcí `check_label`, `check_var`, `check_symbol`, `check_type`. Pokud instrukce (kromě `defvar`) využívá proměnnou, zavolá se také instrukce `is_var_defined`, která zkontroluje, zda je proměnná na daném rámci definována. Podle potřeby se volá také funkce `get_value_from_symbol` na získání hodnoty uložené v proměnné. Následně se provede samotná instrukce. Pokud se výsledek instrukce ukládá do proměnné, je jako poslední zavolána funkce `move_value_to_var`, která hodnotu do proměnné uloží. V každé části vykonávání instrukce se také provádí množství sémantických kontrol. Pokud vše proběhne bez problému, skript skončí úspěšně s návratovou hodnotou 0.

Skript test.php

Skript `test.php` pro testování skriptů `parse.php` a `interpret.py` nejprve zkontroluje volitelné parametry `--directory=`, `--recursive`, `--parse-script=`, `--int-script=`, `--parse-only`, `--int-only` a `--help`. Zakázané kombinace parametrů a neočekáváné parametry vedou na chybu 10. Pokud na konci zadané cesty k testům chybí na konci lomítko, skript si tento znak do uložené cesty přidá.

Následně skript prochází jednotlivé soubory v cestě. Pokud nalezne soubor s příponou `.src`, inkrementuje proměnnou s počtem nalezených testů a hledá také soubory s příponami `.in`, `.rc` a `.out` a stejným názvem. Pokud tyto soubory nejsou nalezeny, vygenerují se s implicitním obsahem – prázdný soubor pro `.in` a `.out`, 0 pro `.rc`. Poté se na základě uživatelem zvolených parametrů příkazové řádky test spustí skriptem `parse.php`, interpret.py, nebo postupně obojími v případě, že nebyl zadán ani parametr `--int-only`, ani `--parse-only`. Do skriptu `interpret.py` se předává cesta k souboru s příponou `.in`, kde se nachází vstup pro instrukci `read` v interpretu. Pokud se skript nepodaří nalézt, `test.php` skončí s chybou 11.

Jako další se porovná návratová hodnota skriptu s návratovou hodnotou uloženou v souboru s příponou `.rc`. Pokud se návratové hodnoty liší, skript inkrementuje proměnnou s počtem selhaných testů a vygeneruje a uloží patřičný kód html. Pokud jsou návratové hodnoty stejné, skript přede na kontrolu výstupu skriptu, který se porovnává s referenčním výstupem uloženým v souboru s příponou `.out`. Pokud byl testován skript `parse.php`, probíhá kontrola výstupů pomocí nástroje `jexamxml`. Pokud byl testován skript `interpret.py`, probíhá kontrola výstupů pomocí nástroje příkazu `diff`. Na základě kontroly se vygeneruje patřičný kód html. Pokud se výstupy lišily, skript inkrementuje proměnnou pro počet selhaných testů.

Na konci skriptu se smažou všechny dočasné soubory vytvořené skriptem a následně skript vytiskne kód html se zprávou o proběhlých testech na standardní výstup.