Implementačná dokumentácia k 2. úlohe do IPP 2021/2022

Meno a priezvisko: Pavel Kratochvíl
Login: xkrato61

Zadanie:

Cieľom 2. úlohy do predmetu IPP bolo vytvorenie dvojice skriptov `test.php` a `interpret.py`. Skript `interpret.py` interpretoval vstup vo formáte XML daného na užívateľskom vstupe alebo zo súboru. Skript test.php následne testoval správnosť návratových kódov a výstupov skriptov parse.php(z predošlej úlohy) a interpret.py. Zo získaných informácií vytvára správu, kde môže užívateľ nájsť mená a všetky informácie z testovania pre každý vykonaný test.

Skript `interpret.py`

Tento skript vykonáva inštrukcie jazyka IPPcode22 vo formáte XML. Zdrojový kód je spracovávaný z užívateľského vstupu alebo zo zadaného súboru(`--source=`). Ďalej v prípade potreby načítava vstup z užívateľského vstupu(stdin), prípadne zo zadaného súboru(`--input=`). Výstup z programu je presmerovaný na štandardný výstup(stdout). K zobrazeniu nápovedy dochádza pri spustení s flag-om `--help`.

Popis implementácie:

Skript sa začína načítaním vstupných argumentov pomocou štandardnej knižnice argparse, ich kontrolou a kontrolou ich kolízií. Následne som použil štandardnú knižnicu `xml`, ktorá mi umožnila kontrolu a načítanie vstupu vo formáte XML. Vo vygenerovanom XML strome iterujem a ukladám informácie o inštrukciách a ich argumentoch do inštancií tried `Instruction` a `Argument` a ukladám mená náveští spolu s číslami inštrukcií, v ktorých sa nachádzajú do slovníka. Inštrukcie a argumenty jednotlivých inštrukcií sú zoradené na základe atribútu `opcode` prípadne podľa značky xml elementu argumentu(napr. `arg1`).

Následuje samotné vykonávanie inštrukcií. Na vykonanie príslušného kódu zodpovedajúceho aktuálnej inštrukcii využívam jednoduchú konštrukciu `if-else`. Ak inštrukcia využíva nejaké argumenty, tak ich získavam buď priamo zo vstupu(konštanty), prípadne zo zadaného rámca(v názve premennej). Na správne fungovanie a ukladanie rámcov využívam vlastnú implementáciu zásobníka triedou `Stack`.

Spracovanie chyby počas behu programu:

Chcel som v projekte implementovať spôsob pre čo najlepší výpis chybovej hlášky pre užívateľa. Pri akejkoľvek chybe je volaná funkcia `exit_error()` s parametrom chybového kódu určeného miestom výskytu. Vďaka tomu je užívateľ informovaný chybovou hláškou na `STDERR`. Napríklad o neznámej inštrukcii na vstupe. Návratový kód sa zo skriptu vracia podľa špecifikácie.

Skript `test.php`

Poslednou časťou 2. úlohy bolo vytvorenie skriptu `test.php` uľahčujúceho automatickú kontrolu skriptov `parse.php` a `interpret.py`.

Popis implementácie:

Po spustení dochádza k načítaniu vstupných parametrov, kontrole ich kolízií a nastaveniu premenných v inštancii triedy `InputArguments`. Hlavná a jediná inštancia triedy `Tester` si potom uloží odkaz na inštanciu triedy `InputArguments` do svojich atribútov. Tie sú potrebné na neskoršie riadenie behu programu, napríklad ak užívateľ zadá ako vstupný argument `--int-only`, `Tester` vôbec nespustí testy zamerané na skript `parse.php`.

Prehľadávanie priečinku za účelom nájdenia všetkých testov prebieha v inštancií triedy `Tester` vo funkcii `findTests()`. Prehľadávať môže iba koreňový, prípadne zadaný priečinok a ak je zadaný argument `--recursive` tak dochádza aj k prehľadávaniu všetkých podadresárov. Ak bol nájdený testovací súbor s príponou .src ale chýbajú k nemu príslušné súbory s príponami `.in`, `.out`, `.src` tak sú doplnené s príslušným východzím obsahom. Nájdené testy sú uložené v inštanciách triedy `Test`, kde sa o nich priebežne ukladajú všetky získané informácie(napr. meno, cesta k súboru, získané návratové kódy atď.). Tieto inštancie sú agregované v zozname, ktorý je atribútom Testeru. To umožňuje opakované iterovanie cez všetky testovacie súbory pri testoch na zameraných na skript `parse.php`, `interpret.py` prípadne neskôr pri generovaní súhrnnej výstupnej správy. Spúšťanie testovaných skriptov je vykonávané pomocou vstavanej funkcie `exec()` a ich výstupy sú porovnávané s referenčnými pomocou nástroja `JexamXML` alebo `diff`.

Generovanie súhrnnej správy z testovania:

Generovanie prebieha v inštancii triedy `htmlPrinter`. Tá má ako atribúty uložený HTML kód začiatku a konca tabuľky, medzi ktoré sa postupne konkatenujú a vložia výsledky testov všetkých testovacích súborov. Výsledná tabuľka je po ukončení na štandardnom výstupe(`STDOUT`).

Ako malé rozšírenie som doplnil aj jednoduchú súhrnnú tabuľku, v ktorej môže užívateľ zistiť celkový počet vykonaných testov, celkový počet a podiel testov hodnotených ako úspešné.

Spracovanie chyby počas behu programu:

Chcel som v projekte implementovať spôsob pre čo najlepší výpis chybovej hlášky pre užívateľa. Pri akejkoľvek chybe je volaná funkcia `handleError()` s parametrom chybového kódu určeného miestom výskytu. Vďaka tomu je užívateľ informovaný chybovou hláškou na `STDERR`. Napríklad o nedostatočných právach na čítanie alebo zápis . Návratový kód sa zo skriptu vracia podľa špecifikácie.