Implementační dokumentace k 2. úloze do IPP 2018/2019

Jméno a příjmení: Michal Zelenák

Login: xzelen24

Interpret.py

Skript interpret.py začína získaním a spracovaním argumentov skriptu pomocou funkcie `getopt`. Ihneď po spracovaní argumentov nasleduje spracovanie XML vstupu do štruktúry, pomocou `xml.etree.ElementTree.parse`. Potom sú skontrolované tagy koreňového uzlu a pokračuje sa triedením jeho pod-elementov(inštrukcií), podľa ich atribútu „order“. Pri každej inštrukcii prebieha kontrola, či daná inštrukcia patrí jazyku IPPcode19, a taktiež sú kontrolované atribúty jej pod-elementov(argumentov). Ak ide o inštrukciu „label“, atribút argumentu, je skontrolovaný regulárnym výrazom a uložený do poľa spolu s číslom poradia inštrukcie.
Pomocou cyklu potom vykonávam inštrukcie programu od prvej až po poslednú podľa ich poradia v poli. Index poľa, pod ktorým sa nachádza v poli posledná inštrukcia je zhodný s dĺžkou poľa vytriedených inštrukcií(konečná podmienka cyklu). Ak je potrebné v niektorej inštrukcii pracovať s reťazcom, reťazec je skontrolovaný pomocou funkcie `checkString()`, ktorá na jeho kontrolu využíva regulárny výraz. Podobne sú kontrolované názvy premenných, pričom je použitý rovnaký regulárny výraz ako pri inštrukcii „label“.
Pri používaní inštrukcií `Jump`, `Jumpifeq`, `jumpifneq` a `call`, je použitá funkcia `jump()`, ktorá zmení hodnotu používanú cyklom(index poľa inštrukcií). Pri inštrukcii „call“ sa navyše uloží do zásobníka pozícia inštrukcie ktorá nasleduje ihneď po inštrukcii „call“. Táto poloha sa použije opäť pri inštrukcii „return“.
Ak inštrukcia ukladá niečo do premennej je použitá funkciu `saveItemToVar()`, ktorá skontroluje existenciu rámca, a tiež existenciu danej premennej. Pri akejkoľvek chybe je použitá funkcia `callErr(msg, code)`, kde atribút „`msg`“ je správa, ktorá bude vytlačená na chybový výstup, a atribút „`code`“ obsahuje hodnotu s ktorou bude ukončený skript.

Test.php

Skript začína spracovaním argumentov pomocou funkcie `getopt()`. Hneď po ich spracovaní sa skontrolujú ich zakázané kombinácie, poprípade sa vytlačí chyba. Inicializuje sa pole súborov so vstupmi pre testy, a podľa zadaných argumentov sa toto pole filtruje.
Nasleduje testovanie oboch skriptov. Ako meno testu je použitá kombinácia cesty k jednotlivým testom a názov súboru s testom bez prípony. S každým testom sa inkrementuje počítadlo testov, a taktiež jedno z počítadiel pre úspešné alebo neúspešné testy.
Najskôr sa získa obsah súboru s príponou „rc“, ktorý sa uloží do premennej `$rc`. Nasleduje hľadanie a prípadne vytvorenie súborov s príponami „in“ a „out“.
Ak nebol zadaný parameter „--int-only“, je použitá funkcia `exec()`, pre vykonanie jednotlivej skúšky skriptu parse.php. Ak je návratová hodnota skriptu rôzna od nuly, kontroluje sa zhoda s premennou `$rc`. Podľa toho sa do premennej `htmlCode`, pridá html kód ktorý zodpovedá jednému testu. Ak je návratová hodnota nula a zároveň bol zadaný parameter „--parse-only“ porovná sa XML výstup so súborom s príponou „out“. Podľa toho sa opäť generuje html kód.
Ak bol zadaný parameter „--int-only“ získa sa obsah súboru s príponou „src“, ktorý sa použije ako vstup pre interpret.py, namiesto výstupu zo skriptu „parse.php“.
Ak parameter „--parse-only“ nebol zadaný použije sa opäť príkaz `exec()` pre vykonanie skúšky skriptu interpret.py. Ak nie je návratová hodnota skriptu rovná nule, opäť sa porovná s premennou `$rc`, a podľa toho sa k premennej `htmlCode` pridá výstup pre daný test. Ak je návratová hodnota rovná nule, porovná sa výstup interpretu s obsahom súboru s príponou “out“ a podľa toho sa opäť generuje výstupný html kód.

Po vykonaní všetkých testov sa vygeneruje hlavička html kódu a následne sa vygeneruje kód jednotlivých testov(premenná `htmlCode`), následne sa ukončí html telo a taktiež program.