Implementačná dokumentácia k 2.úlohe do IPP 2020/2021

Meno a priezvisko: Ivana Colníková

Login: xcolni00

Skript *test.php*

Tento skript stojí na veľkej funkcii *check_arguments()* a na dlhom cykle *foreach*.

V *check_arguments()* sa kontroluje či sú zadané parametre správne a či nedochádza k nesprávnej kombinácii parametrov. Pokiaľ je jedným zo spúšťaných parametrov *--recursive*, do pomocnej premennej sa ukladá o tom informácia. Takto podobne je riešené aj zadávanie parametrov v cykle *foreach*. Kontroluje sa taktiež aj *--int-script* a *--parse-only*, pomocou ktorých viem či sa bude testovať len parser alebo interpret, prípadne oba naraz.

Na záver sa generuje prehľadná štatistika o vykonaných testoch, kde je uvedené koľko testov prešlo správne a koľko testov neprešlo. Vždy sa vypisuje cesta daného testu či už test úspešne prešiel alebo zlyhal.

Rozšírenie FILES u *test.php*

Najskôr sa kontroluje či sa daný súbor dá otvoriť, teda či daný súbor existuje. Po úspešnej kontrole je súbor čítaný riadok po riadku a podľa toho, či sa na danom riadku nachádza názov súboru alebo názov adresára pridávam do zoznamu src súborov daný súbor alebo src súbory z daného adresára. Pri použití prepínača *--match* sa pred pridaním do tohto zoznamu kontroluje či daný názov vyhovuje zadanému regulárnemu výrazu.

Skript *interpret.py*

Mám tu implementovaných hneď niekoľko pomocných funkcií. Jednou z nich je aj *check_arguments(),* v ktorej s pomocou *getopt()* spracovávam argumenty.

Po spracovaní argumentov nasleduje načítanie vstupného XML súboru, na ktorý je využitá knižnica *xml.parsers.expat*. Radí sa medzi nízkoúrovňové knižnice, čo znamená, že na správne spracovanie a uloženie údajov z XML štruktúry sme si vytvorili svoje vlastné pomocné triedy *Argument* a *Instruction*. Na samotné spracovanie elementov sú vytvorené obslužné funkcie *handle_instructions()* a *handle_argument_value().* V nich sa kontroluje lexikálna a syntaktická správnosť XML elementov.

Po načítaní vstupného XML súboru sa vytvára zoznam inštrukcií a ich príslušných argumentov, cez ktoré sa potom prechádza a volajú sa príslušné funkcie pre ich spracovanie. V nich sa najskôr kontroluje počet očakávaných argumentov. Následne prebieha sémantická typová kontrola premenných.

Trieda *FrameVariable* je použitá na uloženie informácií o načítanej premennej. Tieto premenné sa ukladajú do rámcov, ktoré mám implementované ako slovníky. Pri každom čítaní a zápise sa kontroluje existencia danej premennej v rámci. Pre zatiaľ nedefinované hodnoty premenných sa používa reťazec *undef.*

Najdôležitejšou časťou môjho interpreta je nekonečný cyklus *while*. Spracovávajú sa v ňom jednotlivé inštrukcie zo zoznamu. Podľa mena práve spracovávanej inštrukcie sa rozhoduje, ktorú funkciu na jej spracovanie zavolať.

Rozšírenie STACK, STATI u *interpret.py*

STACK: Implementácia tohto rozšírenia spočíva vo vytvorení zoznamu, ktorý som použila ako zásobník a následne definovala funkcie, ktoré s týmto zásobníkom pracujú. V zásobníku sú uložené typy a hodnoty pomocou triedy *FrameVariable*.

STATI: Implementácia tohto rozšírenia spočíva v pridaní ďalších prepínačov do zoznamu pre funkciu *getopt().* Pre sledovanie počtu vykonaných inštrukcií som si vytvorila pomocnú premennú, ktorá sa zvyšuje pri každej novo načítanej inštrukcii. Na zistenie najviac krát použitej inštrukcie som si vytvorila pomocnú funkciu, ktorá ukladá do slovníka meno inštrukcie a k nej prislúchajúcu dvojicu: číslo označujúce počet, koľkokrát bola daná inštrukcia prevedená a číslo označujúce jej poradie. Na zistenie maximálneho počtu premenných v rámcoch som si vytvorila ďalšiu pomocnú funkciu. Prechádza jednotlivé slovníky a počíta premenné v nich. Táto funkcia je volaná pred spracovaním každej jednej inštrukcie. Výslednú štatistiku zapisujem do súboru zadaného užívateľom.