Implementační dokumentace k 2. úloze do IPP 2019/2020

Jméno a příjmení: Nikol Dudová

Login: xdudov01

interpret.py

Po spuštění skriptu se jako první zkontroluje, zda byly zadány správné parametry. Následně dojde k jejich kontrole a zpracování. Soubor se zdrojovým kódem je zkonvertován na strom pomocí knihovny `xml.etree.ElemenTree` a následně postupně čten instrukce po instrukci. Kontroluji, zda je použit správný `tag`, `opcode` a validní `order`, toto si zapisuji do pořadníku instrukcí. Po tom, co je celý strom prohledaný, se seřadí instrukce podle `opcode` v pořadníku a je pro každou vytvořena příslušná instance její třídy (instrukce WRITE má třídu `write` atd.). Všechny třídy obsahují dvě funkce, `__init__` a `run`. V první zmíněné kontroluji, zda má instrukce správný počet argumentů, zda mají očekávaný `tag` a povolenou hodnotu, toto si ukládám do slovníku `args` pro pozdější zpracování ve funkci `run`. V té se instrukce provádí přesně tak, jak je popsána ve specifikaci.

Co se týče práce s argumenty instrukcí, vytvořila jsem pro každý typ argumentu zvláštní třídu, tedy pro konstanty je třída `constant`, pro proměnné `variable` a pro návěští `labelOp`.
V rámcích jsou tedy uložené odkazy na konkrétní instance a při změně hodnoty se mění hodnota přímo v dané instanci pomocí metody `setValue`. Přitom také volám metodu `setType`, aby byla stále udržovaná správná data o typu instrukce. Když chci naopak hodnotu získat, zavolám `getValue` nebo `getType`, podle toho, jestli potřebuji pracovat s hodnotou argumentu nebo jen s jeho typem.

test.php

Zpracování skriptu je rozděleno do několika fází. Hlavní třídou je `SetTestEnvironment`. V té, jak napovídá název, zkontroluji všechny parametry skriptu a nastavím dle nich prostředí, ve kterém testy budou probíhat. Ve funkci `getDirRecursive` je použit kód pro rekurzivní procházení adresářů, který je silně inspirován problémem a jeho řešením, který jsem našla na Stack Overflow. Při procházení adresářů si ukládám všechny soubory, které v nich jsou. Poté tento seznam projdu a pro každý test si uložím pouze jeho název bez příslušných koncovek. Následně zkontroluji, že ke každému testu existují v adresáři soubory `.in`, `.out`, `.src` a `.rc`, případně je vygeneruji. Při spouštění testů je rozděluji podle toho, zda jsou to testy s parametrem `--parse-only`, `--int-only` nebo bez nich. Podle toho pro test vytvořím instanci `POTest`, `IOTest` nebo `Test`. Přičemž první dvě zmíněné dědí ze třídy `Test`. Kromě konstruktoru obsahují třídy ještě funkce `execute`, pro spuštění testu, `validate`, pro jeho vyhodnocení, a funkce `getResult`, `getOutput` a `getReturnCode` pro získání výsledků testu. Nakonec generuji ve třídě `FormHTML` výslednou tabulku. Ta má dva sloupce – `Test Name` a `Result`. Jméno testu je včetně cesty k němu a bez koncovky. Výsledek je pak `passed` nebo `failed`. Celé pole je podbarveno buď zelenou nebo červenou barvou.