Implementační dokumentace k 2. úloze do IPP 2018/2019

Jméno a příjmení: Martin Chládek
Login: xchlad16

1. Úvod

Cílem druhé části projektu bylo naimplementovat skripty `interpret.php` v jazyce python ve verzi 3.6 a `test.php` v jazyce PHP ve verzi 7.3. První uvedený má za úkol načíst vstupní XML reprezentaci jazyka IPPcode19 a tento program následně interpretovat. Druhý skript slouží pro automatické testování analyzátoru (předmětem 1. úlohy) a interpretu, kde uživatel může zvolit testování obou skriptů ve vzájemné návaznosti či testovat jednotlivé skripty odděleně. Program `test.php` pracuje se vstupními soubory s příponami `.src` (soubor se zdrojovým kódem) `.rc` (návratový kód) `.in` a `.out` (referenční vstup a výstup interpretace), kde soubor s příponou `.src` je povinný, bez kterého nemůže být test proveden. Ostatní soubory jsou případně dogenerovány. Pro nápovědu, jak spouštět dílčí skripty slouží společný parametr `--help`. Implementační detaily jednotlivých skriptů jsou popsány v samostatných kapitolách.

2. Popis implementace `intrepret.py`

Samotná implementace interpretu je v rámci jednoho souboru rozložena do 4 tříd, které reprezentují logické bloky, a 4 samostatně definovaných funkcí.

Mezi samostatně definované funkce patří hlavní funkce `main`, která zaštiťuje celý program a voláním předává řízení běhu programu funkcím z dílčích tříd. Funkcionalita kořenové funkce `main` je rozšířena o analýzu vstupního XML pomocí knihovny `ElementTree`. Nástroj této knihovny uloží celý dokument jako objekt, který je poté předáván funkcím. V mé implementaci nese tento objekt název `program`. Dalšími globálními funkcemi jsou `displayHelp`, která obstarává výpis nápovědy k použití skriptu, `displayError` sloužící pro výpis chybového hlášení na standardní chybový výstup a navrácení příslušného chybového kódu. Tou poslední je pak funkce `closeFiles` zabezpečující uzavření vstupních souborů.

Třídy jsou poté podrobněji popsány v následujících podkapitolách.

2.1 Třída `Arguments`

Primárním úkolem této třídy je klasifikace parametrů, které uživatel zadal při spouštění skriptu. Mimo to, funkce této třídy také zajišťují ověření existence vstupních souborů, případné čtení vstupního XML programu ze standardního vstupu a převedení obsahu souboru na string, se kterým bude pracováno dále v programu.

2.2 Třída `Syntax`

Význam této třídy se dá shrnout do 2 hlavních bodů: provedení syntaktické a lexikální analýzy a uspořádání instrukcí v programu.

Funkce `checkSyntax` realizuje kontrolu formátu hlavičky programu včetně atributů, provádí syntaktickou analýzu iterativním procházením objektu `program` po instrukcích a voláním funkce `check_InstructionSyntax`. Ta na vstupu očekává instrukci jako objekt a seznam referenčních operandů, které syntaxe pro tuto instrukci povoluje. Lexikální analýza je realizována pomocí nástroje pro regulární výrazy `regex`.

Funkce `organize_XML_order` na vstupu očekává objekt `program`, procházením vstupního XML kontroluje vzestupné pořadí jednotlivých instrukcí podle atributu `order` a v případě nesouladu pořadí změní tak, aby bylo odpovídající.

V této třídě se tedy načtený program prochází 2x.

2.3 Třída `FrameOperations`

Uvedená třída obsahuje funkce operující s rámci. Rámce jsou definovány globálně, ale z důvodu logické struktury dokumentace jsem se rozhodl jejich řešení popsat v této kapitole.

Původním úmyslem bylo řešit implementaci rámců jako slovníky, v průběhu jsem však narazil na problém s uchováváním proměnné typu `nil`, který není kompatibilní s jazykem python. Řešením by mohlo být použití typu `None`, ten jsem však již využil pro klasifikaci nedefinovaných proměnných.

Rozhodl jsem se tedy implementovat informace o každé proměnné jako seznam, který obsahuje položky: jméno, hodnota, typ. Rámec je tedy implementován jako seznam, jehož každým prvkem je další seznam.

Třída obsahuje několik funkcí, které mají na starost ověřit existenci rámce, vkládání proměnné do rámce, aktualizaci hodnoty proměnné či nalezení proměnné v daném rámci a navrácení její hodnoty a datového typu.

2.4 Třída `Interpret`

Jak už název napovídá, tato třída zajišťuje samotnou interpretaci. Obsahuje funkci pro každou instrukci jazyka IPPcode19, pomocné funkce a hlavní funkci `runProgram`.

Úlohou hlavní funkce `runProgram` je iterativní procházení programem a volání příslušné funkce pro každou instrukci. Iterace je prováděna, dokud je indexace v intervalu celkového počtu instrukcí v programu. Skoky na návěští (na určitou pozici v programu), jsou zajištěny tak, že funkce dané instrukce vrátí index, na který chce skočit (index odpovídá pozici určité instrukce v kódu). V případě, že instrukce nemá zájem provést skok v programu, vrátí 0 a cyklus se přesune na další instrukci v pořadí.

Z pomocných funkcí jsou nejvýznamnější dvě:

- `get_Value_and_Type` – Funkce na vstupu očekává argument a zajistí navrácení hodnoty a typu konstanty/proměnné. V případě proměnné přitom využívá funkcí implementovaných ve třídě `FrameOperations`
- `get_Frame_and_Name` – Funkce rozdělí reprezentaci proměnné v XML a vrátí její rámec a název.

3. Popis implementace `test.php`

Implementace skriptu `test.php` je rozdělena do 5 tříd. Jednotlivé třídy představují logické bloky programu.

3.1 Třída `Arguments`

Třída Arguments obsahuje jedinou funkci a tou je funkce `ParseArgument`. Její úlohou je klasifikovat uživatelem zadané parametry, zkontrolovat jejich správnost a povolené kombinace.

3.2 Třída `Tests`

Tato třída je pro `test.php` zásadní, protože v ní probíhá veškeré testování. Obsahuje 4 funkce, které se volají v jednotném pořadí.

Na počátku stojí funkce `prepareTest`, ta zajišťuje potenciální rekurzi skrz vnořené adresáře. Rekurzivně hledá soubor s příponou `.src` a zavolá funkci `doOneTest`, přičemž jí předá aktuální adresář a název nalezeného `.src` souboru.

Funkce `doOneTest` hledá podle předaného názvu soubory s příponami `.rc`, `.in`, `.out`. V případě jejich neexistence je vytvoří. Vytvoří objekt `test` a ten předává v závislosti na zadaných argumentech funkci `testParser` nebo `testInterpret`.

Existují 2 různé scénáře, jak bude probíhat testování ve funkci `testParser`. Pro oba scénáře platí, že nejprve je spuštěn analyzátor příkazem `exec` a do dočasného souboru je uchován jeho výstup. V případě zvolené možnosti testování pouze analyzátoru jsou porovnávány návratové kódy a v případě jejich rovnosti a kódu 0 jsou porovnávány výstupy nástrojem `jExam`. V opačném případě se porovnávají pouze návratové kódy a rozhoduje se, zda byl test neúspěšný již v analyzátoru, nebo bude volána funkce `testInterpret`.

Pokud uživatelem byla zvolena varianta testování obou skriptů, je zdrojovým souborem pro skript interpretu výstup analyzátoru z předchozí funkce, jinak je načten `.src` soubor. Nad zdrojovým souborem je spuštěn interpret a poté jsou porovnávány výstupní kódy a pomocí nástroje `diff` porovnány výstupní soubory. Na základě porovnání je rozhodnuto o úspěšnosti testů.

3.3 Třída `HTML`

V uvedené třídě existují 3 funkce, `WriteHTML` pro zápis výstupu v jazyce HTML5, `WriteCSS` doplní HTML výstup o jednoduchý CSS styl a `WriteTestResults`, která slouží pro výpis výsledků testování.

3.4 Třída `GlobalClass`

Funkce v této třídě obstarávají případný výpis nápovědy k používání skriptu a smazání dočasných souborů použitých během testování.

3.5 Třída `HandlingErrors`

Tato třída pouze zajišťuje výpis chybového hlášení na standardní chybový výstup a navrácení příslušného kódu.