IPP Projekt - 2. úloha

Jakub Hlava (xhlava52)

18. dubna 2021

Interpret

Základní návrh

Interpret se skládá z hlavní třídy Interpreter, která obaluje implementace jednotlivých instrukcí, několik pomocných metod a data a struktury nutné pro interpretaci (počítadlo instrukcí, zásobník volání, paměťové rámce, atd.), třídy `Frame`, která implementuje paměťové rámce a metody pro práci s nimi a několika pomocných funkcí pro předzpracování různých prvků zdrojového kódu jako např. řetězců a identifikátorů a funkce `get_input_line`, která tvoří rozhraní pro usnadnění načítání uživatelských vstupů kvůli nutnosti implementovat jak zadávání na standardní vstup, tak čtení ze souboru parametrem `input`.

Zpracování zdrojové reprezentace jazyka v XML

Pro zpracování XML je využita knihovna LXML. Načtený soubor se knihovna pokusí rozložit na strom prvků, se kterými poté pracuje instance třídy `Interpreter`.

Třída `Interpreter`

Obsahuje v sobě odkazy na aktivní globální, lokální a dočasný rámec, zásobník rámců, počítadlo instrukcím, slovník pro známé návěští skoků, zásobník volání, zásobník, odkaz na seznam LXML elementů s instrukcemi a další pomocné atributy.

Průběh vykonávání je řízený metodou execute, která po zavolání vykoná jednu instrukci a případně změní stav interpretu. Stav 0 značí, že je interpret připravený vykonat další instrukci, stav 1, že program skončil.

Každá instrukce je reprezentována jednou metodou s názvem odpovídajícím instrukci (formát `_opcode`). Mimo metody pro jednotlivé instrukce jsou v třídě další pomocné metody, např. `calculate`, která sjednocuje výpočet pro aritmetické a logické instrukce, `getfobj`, která podle identifikátoru proměnné získá objekt paměťového rámce, `forward_search_for_label`, která hledá návěští, nacházející se až za instrukcí skoku nebo volání (namísto dvouprůchodové interpretace) a `parse_symb`, která automaticky rozpozná typ argumentu (literál nebo proměnná) a získá jeho hodnotu.

Třída `Frame`

Paměťové rámce jsou pro usnadnění práce implementovány jako třída `Frame`. Proměnné v rámci jsou uchovávány ve slovníku, druhým atributem je pomocný seznam neinicializovaných proměnných, díky kterému mohu rozlišovat mezi definovanou, ale neinicializovanou proměnnou a proměnnou typu `nil` - obě jsou ve slovníku reprezentovány hodnotou `None`.

Pro práci s rámcem slouží sada metod, které umožňují definovat proměnnou, nastavit nebo načíst její hodnotu, případně získat informaci o tom, zda vůbec proměnná s požadovaným jménem existuje nebo zda je inicializovaná.

Interpretace

Program na základě kombinace parametrů `input` a `source` určí zdroje XML reprezentace zdrojového kódu a standardního vstupu, poté načte XML reprezentaci, pomocí knihovny LXML ji zpracuje, inicializuje interpret (objekt třídy `Interpreter`) a v cyklu vykonává instrukce voláním metody `execute`, dokud interpret nepřejde do stavu 1, tedy konec programu nebo dokud neskončí interpretace s chybou (tedy okamžitě voláním `sys.exit` s odpovídajícím chybovým kódem.

Testovací rámec

Princip a průběh testování

Testovací skript na začátku dle parametrů, se kterými byl spuštěn vyhledá a detekuje dostupné testy v zadané složce, příp. rekurzivně v podsložkách. Skript hledá soubory `.src`, `.rc`, `.in`, `.out`, které podle jejich názvů třídí k jednotlivým testům, chybějící z nich vytvoří s výchozí hodnotou (mimo souboru `.src` se zdrojovým kódem, pak se informace o zbytku zahodí).

Následně jsou testy postupně spouštěny, forma spouštění a kontroly chování je vybrána na základě parametrů spuštění testovacího skriptu. U testů parseru se spouští skript v PHP interpretu, kontroluje se návratový kód a výstupní XML se porovnává za pomoci nástroje `JExamXML`, u testů interpretu se testovaný skript spouští Python interpretem, kontroluje se návratový kód a výstupy se porovnávají nástrojem `diff`. V případě testování obojího se nejprve spustí parser, ověří se návratový kód, poté je spuštěna interpretace, ověří se znovu návratový kód a porovnají se výstupy. U každého testu se uloží jeho název nebo relativní cesta k podsložce, výsledek a u neúspěšného testu i důvod neúspěchu.

Generování výsledkové stránky

Výsledky testů jsou sečteny a testy a kategorie testů jsou převedeny na strom. Sumy úspěšných a neúspěšných testů spolu s celkovým počtem testů jsou vykresleny na začátku výsledkové stránky. Pod nimi jsou přehledně vypsány rozbalitelné kategorie testů s informací o počtu úspěšných a neúspěšných testů v dané kategorii, ve kterých lze poté snadno najít konkrétní test, který uživatele zajímá.