Implementační dokumentace k 2. úloze do IPP 2022/23
Jméno a příjmení: Vladimír Hucovič
Login: xhucov00

Popis

Skript nejprve zpracuje argumenty příkazové řádky pomocí knihovny `argparse`. Přečte vstupní XML reprezentaci programu a zkontroluje jeho strukturu ve funkci `checkXML`. Poté z každého elementu `<instruction>` vytvoří v tovární metodě `CreateInstruction` ve třídě `InstructionFactory` objekt typu `Instruction`, který po zbytek běhu interpretu reprezentuje danou instrukci. Třída `InstructionFactory` také obsahuje konstantu se slovníkem operačních kódů instrukcí a jejich příslušných tříd a správnými typy argumentů. Pokud metoda `CreateInstruction` nenalezne daný operační kód v tomto slovníku, vyhodí výjímku. Jinak volá na každou XML reprezentaci metodu `Create` v tovární třídě `ArgumentFactory`, která podle typu argumentu vytvoří objekt `Argument`. Jakmile jsou všechny argumenty připravené, `CreateInstruction` zavolá konstruktor třídy pro danou instrukci.

Kromě XML elementu pro instrukci potřebuje `CreateInstruction` ještě objekt `InterpreterData`, který uchovává data, nad kterými instrukce pracují.

Každá instrukce ve skriptu má svoji třídu. Tyto třídy se liší pouze jejich implementací metody `Execute`. Interpret je tedy snadno rozšiřitelný o další instrukce: stačí vytvořit novou třídu pro nový typ instrukce, implementovat metodu `Execute` a přidat příslušný záznam do slovníku instrukcí.

Jakmile jsou vytvořeny instrukce, je vytvořen objekt `Interpreter` který řídí interpretaci. Jsou do něj přidány objekty instrukcí a je mu přiřazen objekt `InterpreterData`. Poté se volá metoda `Interpret`, která provede program. Pokud interpret narazí na výjímku, ukončí interpretaci, nastaví návratový kód a případně chybovou hlášku. Pokud byla výjímka způsobena chybou ve vstupním programu, chybová hláška se vypíše a skript končí s návratovým kódem v objektu `Interpret`.

Objekt třídy `InterpreterData` poskytuje instrukcím rozhraní, přes který komunikují s interpretem, aniž by vznikla kruhová závislost mezi interpretem a instrukcemi. Tato třída obsahuje zdroj pro instrukci `READ`, ukazatel na aktuálně prováděnou instrukci, globální a dočasný rámec, zásobník rámců, datový zásobník, zásobník volání a slovník návěští spolu s jejich umístěním v seznamu instrukcí v interpretu. Metody této třídy umožňují instrukcím měnit a číst data v těchto strukturách.

Proměnné jsou uchovávány v rámcích, které jsou reprezentovány instancemi třídy `Frame`. Tyto instance obsahují slovník se jmény a hodnotami proměnných. Když se v interpretu přistupuje k hodnotám argumentů v objektech instrukcí, volá se metoda `GetArgumentValue` ve třídě `InterpretData`, která pouze vrátí hodnotu argumentu v případě, že se jedná o konstantu, a v případě, že se jedná o instanci třídy `VarName`, najde rámec, ve kterém se proměnná nachází, a získá její hodnotu.

Datové typy

Protože jazyk Python podporuje různé typové konverze mezi datovými typy a v jazyce IPPCode23 je většina těchto konverzí zakázána, jsou hodnoty argumentů a proměnných reprezentovány třídami `IPPInt`, `IPPFloat`, `IPPString`, `IPPBool` a `Nil`. Tyto přizpůsobené typy dědí od existujících typů v jazyce Python (s výjímkou třídy IPPBool, protože od třídy `bool` se v pythonu nedá dědit) a překrývají jejich standardní tak, aby nebylo možné provádět implicitní konverze, vyvolané různými operacemi nad odlišnými typy (např. sčítání). Při interpretaci kódu jsou pak při pokusu o provedení operace s nepodporovanými typy vyhozeny výjímky, na které se zareaguje ukončením interpretace s příslušným návratovým kódem.

Implementovaná rozšíření

Kromě kompletní implementace základního zadání jsou implementována i všechna rozšíření (STATI, FLOAT, STACK). Implementace rozšíření `FLOAT` byla v tomto návrhu velice snadná - stačilo přidat nový datový typ `IPPFloat` a implementovat instrukce konverze mezi typy `IPPInt` a `IPPFloat`. U obou těchto tříd je taky přepsána metoda `__new__`, která podle typu argumentu, ze kterého se tyto typy tvoří, provede jednoduchou konverzi z nadřazených typů `float, int`, nebo provede konverzi z typu `str`, která zahrnuje i správnou konverzi dané hodnoty z řetězce, ať je číslo zapsáno dekadicky, hexadecimálně, či (v případě typu `IPPInt`) oktalově.

Implementace rozšíření STACK obnášela rozšíření seznamu instrukcí o zásobníkový kód. Jak už bylo popsáno výše, proces zavádění těchto nových instrukcí byl jednoduchý - byly vytvořeny nové třídy pro tyto instrukce, implementována metoda `Execute` a přidán záznam do slovníku instrukcí.

Pro rozšíření STATI byla vytvořena třída `StatisticsCollectingInterpreter`, která dědí od dříve popsané třídy `Interpreter`. Tato třída obsahuje oproti nadřazenému interpretu ještě atributy, do kterých jsou uloženy sbírané statistiky, a metody používané pro zisk těchto statistik. Je zde také překryta metoda `Interpret`, která rozšiřuje původní metodu právě o sběr statistik.

Diagram tříd