Implementační dokumentace k 2. úloze do IPP 2022/2023
Jméno a příjmení: Ondřej Lukášek
Login: xlukas15

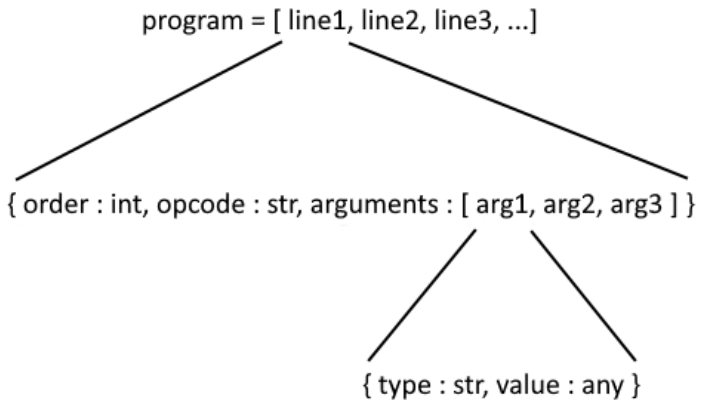
Začátek skriptu

Na úplném začátku skriptu se nejdřív ověří, jaké byly zadány argumenty příkazové řádky (používá se knihovní modul `argparse`) a následně se spustí funkce `main()`. Ta připraví zdrojový soubor, vstupní soubor, případně soubor statistik. Také se provede instanciace třídy `Runtime` a do jejího konstruktoru se předá zdrojový a vstupní soubor (případně stdin). Třída `Runtime` v konstruktoru provede instanciaci třídy `XMLReader` a zavolá její metodu `read()`. Ta jednak ověří správnost a formát vstupního XML souboru a také načte instrukce do `program` (viz Obrázek 1).

Uložení dat v `program` je poměrně komplikované. Jedná se totiž o seznam instrukcí. Každá instrukce se dále dělí na slovníky se třemi položkami typu klíč-hodnota. První položka je `order` a zaznamenává pořadí instrukce, druhá je `opcode` a ta v sobě uchovává, o jakou instrukci se jedná. Poslední položkou je `arguments`, jejíž hodnotou je seznam argumentů instrukce (tento seznam může mít maximálně 3 prvky). Každý argument je slovníkem, který v sobě má typ, jehož hodnotou je řetězec, značící datový typ argumentu, a také má v sobě hodnotu argumentu.

Tento seznam instrukcí se dále uspořádá podle pořadí `order` u každé instrukce a předává se dále. Konstruktor třídy `Runtime` také vytvoří instanci dalších tříd potřebných pro běh programu, kterými jsou `Frame`, `Stack`, `StackFrame`. Také se založí další proměnné, které budou v potřeba v průběhu vykonávání programu, jako například `inst_nr`, která obsahuje číslo zpracovávaného řádku (index do seznamu `program`), proměnné udržující statistiky a další.

Po zpracování zdrojových XML dat se spustí metoda `check_program_and_fill_labels()`, která průchodem najde všechna návěští v programu a uloží si je do slovníku `labels`, který obsahuje indexy do seznamu `program`, což se později hodí pro instrukce skoku. Tato metoda také zkontroluje instrukce v programu, jestli jsou platné, mají správný počet operandů a operandy jsou validních typů. Pro to se používá slovník s instrukčními operandy jako klíči a hodnotou obsahující počet argumentů instrukce (slovník `instruction_operands`). Validní typy jsou uložené v tuple `types`.



Obrázek 1 - struktura seznamu program

Hlavní smyčka

Po načtení programu se spustí metoda `run()` objektu `runtime`. Ta postupně od 0 prochází jednotlivé řádky programu. Pořadí procházení mohou změnit instrukce skoku, které vlastně jen změní aktuální číslo řádku. Pro volání metod, zpracovávajících jednotlivé instrukce se vytvoří slovník `function_map`, který mapuje každé klíčové slovo v `instruction_operands` na odpovídající metodu objektu runtime s názvem začínajícím `do_` následovaným operačním kódem (např `do_ADD`, `do_CALL`, apod.).

Hlavní smyčka také počítá statistiky pro rozšíření STATI.

Pomocné metody

Aby bylo vlastní provádění instrukcí snadnější, jsou vytvořeny různé metody. Metoda `extract_args` načte argumenty instrukce do jednoduchého seznamu. To využívá metoda `get_operands`, která ale jako parametr používá seznam parametrů instrukce v čitelné podobě. To značně zpřehledňuje program.

Například metoda pro zpracování instrukce ADD se jmenuje do_ADD a jako první volá `get_operands(arguments, ('var', 'intfloat', 'intfloat'))`. Tím je dobře vidět, že instrukce potřebuje proměnnou jako první operand a dvě `int` nebo `float` hodnoty pro sečtení. Metoda `get_operands` kromě vlastního načtení operandů instrukce provádí kontroly a případně konverze typu podle požadavku. Podobné metody existují také pro instrukce pracující se zásobníkem.

K tomu mimo jiné používá i další pomocnou metodu `symbol_value`. Ta má na vstupu typ a hodnotu, které i vrátí, pokud se jedná o přímou hodnotu. Pokud je na vstupu proměnná, metoda vrátí její hodnotu.

Paměťový model

Proměnné jsou ukládány do slovníku `variables` třídy `Frame`. Slovník má jako klíč jméno proměnné a hodnota obsahuje její hodnotu. Kromě slovníku `variables` obsahuje třída `Frame` i informaci o tom, zda je daný rámec inicializován. Rámce se mohou ukládat nebo vybírat do zásobníku rámců, který tvoří třída `FrameStack`. S ní pracují například instrukce PUSHFRAME, POPFRAME apod.

Jako pomocná metoda pro práci s proměnnými slouží `get_frame_var()`, která jako vstup dostane jméno proměnné ve tvaru používaném jazykem IPPcode23 (například “GF@auto“) a vrací tuple, který obsahuje objekt správného rámce a jméno proměnné za znakem @.

Zásobník hodnot a zásobník volání je tvořen objekty třídy `Stack`. Ta implementuje metody `push()`, `pop()`, `clear()`. Vlastní hodnoty jsou uložené v objektu třídy `deque()`, která se hodí pro implementaci zásobníku lépe než klasický seznam. Stejnou třídu `deque()` používá i třída `FrameStack`.

Ukončení skriptu

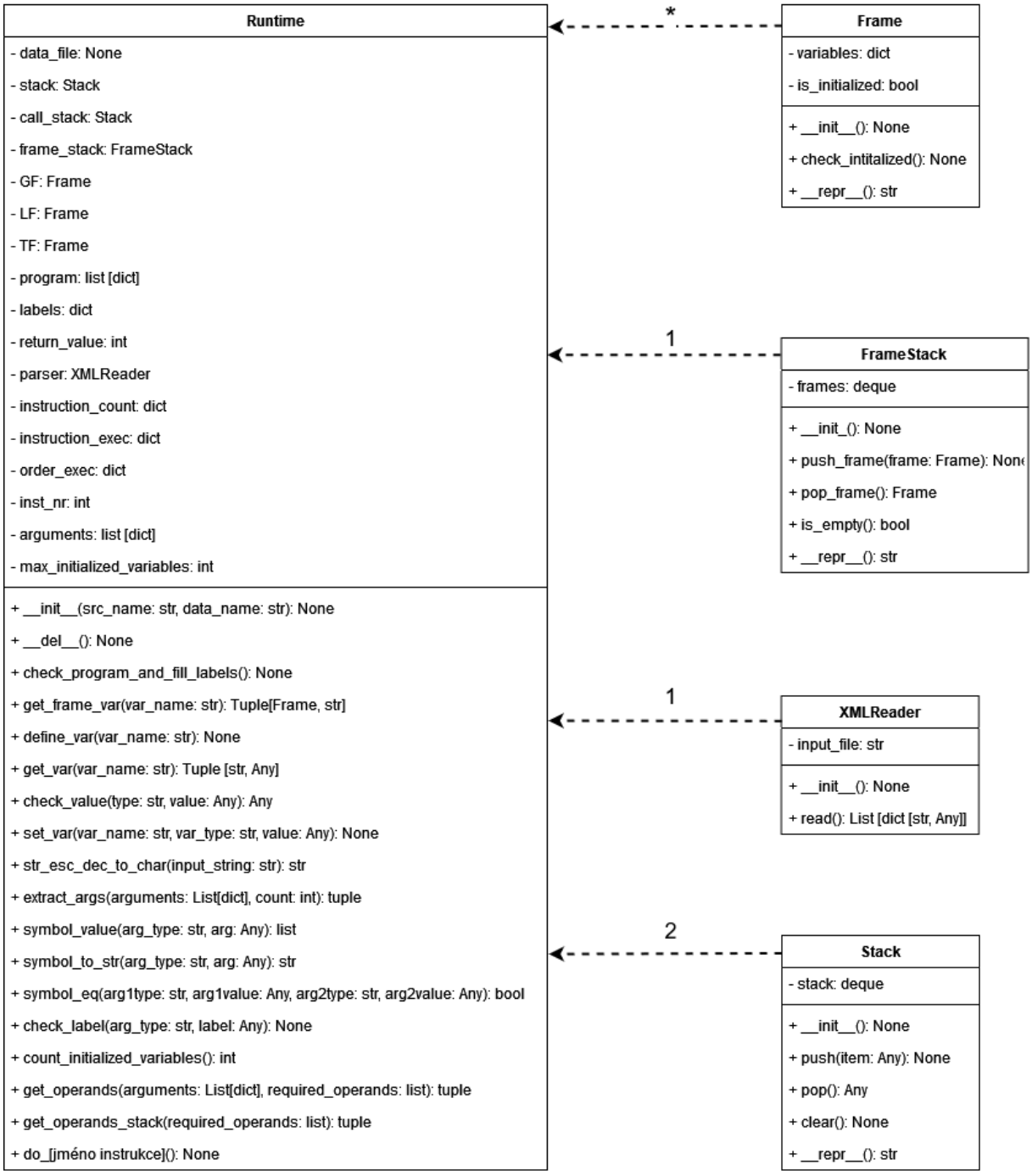
Po ukončení metody `run()` objektu `runtime` se uloží a uzavře soubor se statistikami a program ukončí s návratovou hodnotou. Ta je obvykle 0, pokud ji nezmodifikovala instrukce EXIT.

Implementovaná rozšíření:

Projekt implementuje veškerá možná rozšíření (FLOAT, STACK, STATI). Jsou implementovány instrukce pro práci se zásobníkem, proměnné typu float a požadované statistiky.

Testování

Na testování byly použity testy, které se šířily mezi studenty. Jsou poměrně komplexní, obsahují okolo 700 testů zahrnujících interpret, které jsem pouze doplnil o několik specifických testů a některé upravil tam, kde mi to přišlo vhodné.



Obrázek 2 - diagram tříd pro interpret