Implementační dokumentace k 2. úloze do IPP 2021/2022
Jméno a příjmení: Simona Češková
Login: xcesko00

1 Interpret

1.1 Běh programu

Program začíná kontrolou XML kódu a uložením každé instrukce s jejími argumenty do třídy Instruction, Argument Všechny instrukce jsou uložené v listu Instructions. Pak následuje inicializací rámců a první průchod přes list instrukcí a ukládáním návěstích do LABEL_FRAME.

Hlavní průchod programu je obsažen ve while cyklu pro možnost použití instrukcí se skoky. Zde se už vykonávají jednotlivé instrukce, tak že se zavolá funkce getattr(), která přijme opcode názvu instrukce a podle toho vyhledá příslušnou metodu.

1.2 Rámce

1.2.1 Rámce pro ukládání proměnných

Vytvořený list FRAMES v sobě obsahuje na 0. indexu globální rámec, na 2. indexu zásobník lokálních rámců. Při použití instrukce CREATEFRAME se inicializuje na 3. indexu dočasný rámec.

Při vytváření lokálních rámců, se rámec uloží do zásobníku lokálních rámců, tak že se uloží na index, který se rovná délce listu, který představuje zásobník.

Každý z těchto tří rámců je implementován jako slovník. Klíčem se stává jméno ukládané proměnné a hodnotou instance třídy Variables.

1.2.2 Datový zásobník

Datový zásobník je inicializovaný už od začátku spolu s globálním rámcem, proto se nachází na 1. indexu rámce rámců FRAMES.

1.2.3 Rámec pro návěstí

LABEL_FRAME. Jednotlivé návěští nelze ukládat do globálního rámce, kdyby se proměnná tam uložené jmenovala stejně jako návěští. Proto je vytvořen tento rámec, který se využije pouze jeden krát při prvním průchodu všemi instrukcemi. V tomto průchodu se ukládají pouze návěstí a jejich čísla instrukce, kam ukazují. Stejně jako GF, LF a TF je tvořen slovníkem. Klíč je jméno návěští a hodnotou je návěští uložené do instance třídy Variables. Nepoužívané části třídy jsou zapsány jako None.

1.2.4 Zásobník pro ukládání pozicí

CALL_FRAME. Zde se uloží při zavolání instrukce CALL vždy na nejsvrchnější pozici zásobníku číslo reprezentující order instrukce, kam se při zavolání instrukce RETURN vrátí běh programu.

1.2.5 Hledání správného rámce

Při práci s proměnnou se musí použít nejaktuálnější hodnota, která je uložená v některém z rámců a pomocí funkce frameFinding se najde zadaný rámec podle atributu v instanci třídy Argument, která je uložené v listu, který se nachází v třídě Instruction

1.3 STACK

Implementace rozšíření STACK je pomocí vytvořeného datového zásobníku, kam ukládají používané proměnné pomocí třídy Variables. Takto uložené hodnoty lze bez komplikací později ukládat do rámců, protože v rámcích je používaná stejná struktura pro ukládání proměnných.

Narozdíl od práce s instrukcemi, které pracují s globálním, lokálním, nebo s dočasným rámcem, se při provádění operací nekontroluje, zda proměnná s kterou pracuje instrukce byla deklarovaná, nebo definovaná, protože pomocí instrukce PUSHS nelze přidat do datového zádobníku nedefinovanou, nebo nedeklarovanou proměnnou.

1.4 STATI

Pro výpis do souboru se spouští funkce `statsOutput`, do které jsou předávány hodnoty z `main`. Pro parametr `--insts` se připočívátá +1 před každým vykonáním instrukce. `--vars` inkrementuje při každém zavolání `DEFVAR` a pokud přesáhne svoje maximum z dřívějška, tak se přepíše maximum na momentální počet. Pro `--hot` se ukládají do seznamu `hotList` a ve funkci `statsOutput` se spočítá, které `order` číslo se vyskytlo nejvícekrát.

1.5 NVI

Třída `Instruction` má v sobě metodu pro přidání argumentů z třídy `Argument` a také metody pro výpočet všech instrukcí programu. Třída `Variables` slouží k přechovávání informací z instrukcí. Jako vzor je použita metoda skladba, protože všechny tři třídy tvoří hierarchii a jsou mezisebou propojené. Třída `Argument` spolu s `Instrukce` tvoří strukturální celek.

Třída `Checks` je podle vzoru observer a její metody kontrolují deklarace a definice proměnných. Třída `Returns` slouží jako adaptér mezi ostatními třídami. Její metody vrací hodnotu, nebo typ proměnné či konstanty.

2 Test

2.1 Rozdělení programu

Podle toho jaké parametry jsou zadané, tak je rozdělený i program na tři části. Všechny tři části `interpret-only`, `parse-only`, `both` (testy bez zadaného parametru) mají podobnou strukturu.

Nejdříve se spracují argumenty a zvolí správné cesty k adresářům testů a ostatním souborů. Poté se prohledává složka v které se program právě nachází a pokud dojde na soubor s příponou .src, tak se posune dále. Je-li zadaná rekurze, tak vkročí do ní. Jsou to funkce `recursionParse/Int/Both`.
tab Jinak pouze vykonává kontrolu. Před kontrolou vytvoří chybějící soubory (jejich existence se kontroluje pomocí `file_exists()`.

2.2 Testování parseru

Nejdříve se musí vygenerovat `XML` kód pro porovnávání návratových kódů a výstupu. Generování používá funkci `exec()`. a testování pomocí `java -jar`.

2.3 Společné testy

Vygenerovaný `XML` kód si převezme `interpret`.

Pokud je zadán parametr `--clean`, tak pomocí funkce `unlink()` se vymažou dočasné generované soubory během testu (pouze pomocne soubory maji priponu `.gen` a), po skončení porovnávání výstupů.