Implementační dokumentace k 2. úloze do IPP 2019/2020

Jméno a příjmení: Tomáš Julina

Login: xjulin08

Interpret.py

V první části programu je zpracování vstupních argumentů k čemuž je využívána knihovna „*getopt*“, kontrolují se povolené kombinace argumentů a případně program se program ukončí s chybovým kódem.

Samotný interpret je rozložen na pět pomocných tříd:

Třída *Frames* – tato třída zastřešuje veškerou práci s rámci – jelikož je každý rámec společný pro celý program, jsou v ní využívány pouze třídní proměnné (nikoliv instanční). Tato třída obsahuje 3 slovníky (pro každý druh rámce jeden) a také zásobník rámců *frameStack*, který je avyužíván při zpracování instrukcí *PUSHFRAME* a *POPFRAME*, tato třída dále obsahuje tři pomocné třídní funkce – *addVarToFrame*, která vytvoří novou proměnnou a nastaví jí hodnotu *None* a její datový typ na „uninitialised“, dále pak funkci *changeVarValueInFrame*, pomocí které je možné přistupovat k uloženým proměnným v libovolném rámci a následně měnit jejich hodnotu (a datový typ), funkci *getVarValueFromFrame*, pomocí které je možné načíst hodnotu (a datový typ) libovolné proměnné z rámce a také funkci *is_Initialized*, pomocí které lze zjistit, zda vytvořená proměnná byla od vytvoření inicializována.

Třída *Labels* – tato třída slouží k práci s návěštími – má jednu třídní proměnnou typu slovník, která obsahuje název návěští spolu s „adresou“, kterou je možné využít při instrukci skoku na dané návěští.

Třída *Stacks* – tato třída slouží pro práci se zásobníky (nacházející se ve třídě *Interpret*) – zásobník volání (instrukce *CALL*, …) a datový zásobník (instrukce *PUSHS*, …). Při práci s touto třídou jsou využívány její instance, obsahuje dvě jednoduché funkce *push* a *pop*, jejichž funkčnost je zjevná.

Třída *Interpret* – jedná se o hlavní třídu celého interpretu. Hlavní řídící funkce *doEverything* se stará o celý chod interpretu a volá potřebné pomocné funkce – funkci *findLabels*, která projde seznam instrukcí (viz. dále) a zpracuje všechny výskyty instrukce LABEL (kvůli možným potřebám „skákat“ v programu dříve, než by na tyto instrukce došla řada), dále pak funkci *executeInstructions*, která se volá po nalezení návěští a provede veškeré zbylé instrukce včetně kontrol jejich operačních kódů a operandů a také důležitou funkci *sortInstructionList*, díky který je interpret schopen přijmout instrukce v jakémkoliv pořadí (atribut *order*).

Třída *Instruction* – tato třída představuje jednotlivé instrukce k interpretaci. Jsou využívány její instance, ve kterých je uložen operační kód, počet operandů (argumentů) a samotné operandy jsou uloženy ve formě slovníku (obsahující typ argumentu, jeho hodnotu a případě rámec – pro proměnnou). Tato třída obsahuje funkce *checkArguments*, kterou interpret využívá ke kontrole datových typů a počtu operandů, mimo to také kontroluje, že aktuálně zpracovávaná instrukce je podporovaná jazykem IPPcode20. V případě neočekávaného operandu u instrukce kontroluje výjimku *KeyError* (toto je možné díky tomu, že jsou operandy uloženy ve slovníku), která tuto chybu odhalí. Dále funkce *exec*, která na základě již zkontrolovaného operačního kódu zavolá potřebnou funkci reprezentující danou instrukci. V neposlední řadě tato funkce obsahuje pomocnou funkci pro každou instrukci – každá pomocná funkce při načítání argumentů kontroluje jejich správnost (obsahově) a poté je instrukce vykonána.

Funkce *main* – tato funkce vytvoří instanci třídy Interpret a následně interpret spustí se vstupními daty v jazyce XML.

Funkce *parseInput* – tato funkce se na začátku pokusí s využitím knihovny *ElementTree* načíst vstupní XML data buď ze souboru nebo ze standardního vstupu (funkce také zkontroluje, že se jedná o dobře formátovaný soubor). V první řadě je zkontrolováno, zda je v kořenovém elementu atribut *language* s hodnotou IPPcode20 (případně další povolené atributy), poté je iterováno přes jednotlivé „child“ prvky (dále instrukce) celého stromu a je kontrolováno, že každý obsahuje „tag“ *instruction*, značící instrukci (a zároveň je to také jejich jediný povolený „tag“). Jednotlivé

instrukce jsou ukládány do listu instrukcí (což je také výstupem této funkce), u každé instrukce je kontrolováno, že obsahuje povinné atributy – *opcode* a *order*. Atribut *order* je kvůli potřebě kontrolovat opakující se hodnoty a záporné hodnoty (které také považuji za chybné) ukládán do proměnné *orderList* typu list. U každé instrukce jsou dále ukládány její očíslované (na jejich pořadí zápisu v XML kódu v rámci interpretu nezáleží – interpret se dokáže poprat s jakýmkoliv pořadím – důležitý je jejich počet) operandy spolu s typem a obsahem (případně rámcem u proměnné) do slovníku, jednotlivé hodnoty operandů jsou kontrolovány pomocí regulárních výrazů (tak jako u skriptu *parse.php*). Na konci této funkce je pro každou instrukci vytvořena instanci třídy *Instruction* a tato je uložena do listu instrukcí (zatím neseřazený – k seřazení dochází ve třídě *Interpret*).

Sporné případy:

- V případě, že instrukci *JUMPIEFQ / JUMPIFNEQ* je zadán neexistující *LABEL* a podmínka pro skok je vyhodnocena negativně – program pokračuje dál (jako chybu je to přijímáno v případě, že by se mělo skutečně „skákat“)

Test.php

V první řadě jsou zpracovávány vstupní parametry pomocí funkce *getopt*, dále jsou kontrolovány povolené kombinace vstupních argumentů a provedeny nastavení pomocných proměnných.

Funkce *parse_only* – tato funkce si do pole uloží všechny soubory s příponou *.src*, vygeneruje hlavičku výstupního HTML souboru s výsledky testů a případně vytvoří, resp. otevře soubory se vstupy týkající se konkrétního testu – poté je spuštěn skript *parse.php* nad těmito vstupními daty a případně jsou výsledky porovnány pomocí programu JExamXML. Průběžně je generována tabulka s výsledky testování, na konci se zavřou otevřené soubory a smažou se pomocné (temporary) soubory a navíc se vygeneruje shrnutí výsledků testů (pod shrnutím jsou ještě opětovně vypsány pouze neúspěšné testy kvůli snadnější orientaci).

Funkce *int_only* – tato funkce má prakticky naprosto stejnou funkčnost jako funkce *parse_only* s tím rozdílem, že nad vstupními daty je spuštěn program *interpret.py* a k porovnání výstupů dochází pomocí programu *diff*.

Funkce *do_both* – tato funkce nejprve spustí nad vstupními daty skript *parse.php* a poté vyhodnotí výsledek – pokud je potřeba, výstup je použit jako vstup pro program *interpret.py* – konečný výsledek je pomocí programu *diff* porovnán s očekávaným výstupem a průběžně je generována tabulka s výsledky testování.