Implementační dokumentace k 2. úloze do IPP 2019/2020

Jméno a příjmení: Dominik Nejedlý

Login: xnejed09

1 Zpracování a interpretace vstupní XML reprezentace kódu IPPcode20

Celkové zpracování a interpretace vstupního zdrojového textu probíhá v souboru `interpret.py`. Nejprve je pomocí třídy `xml.etree.elementTree` načtena vstupní XML reprezentace kódu IPPcode20 a zkontrolován její formát. V případě chybného formátování je skript ukončen s chybovým kódem 31. Poté je kontrolována struktura načteného XML včetně syntaktické a lexikální správnosti reprezentovaných instrukcí kódu IPPcode20 a jejich argumentů. Při nalezení chyby je provádění programu ukončeno s návratovou hodnotou 32. Veškeré instrukce jsou během tohoto procesu ukládány do slovníku, kde klíčem je jejich pořadí a hodnotou objekt instrukce s uloženým operačním kódem a přiřazenými argumenty. Každý argument nese hodnotu daného typu. Pouze v případě, že je argumentem neterminál `<symb>`, je u argumentu zaznamenán také jeho typ.

Současně se sběrem instrukcí probíhá ukládání jejich číselných hodnot do vedlejšího pole `orderArray`, které je seřazeno podle velikosti, což zajišťuje, že instrukce budou prováděny podle pořadí, a sběr všech návěští včetně kontroly jejich unikátnosti. Návěští jsou ukládána do slovníku, kde klíči jsou jejich názvy a hodnotou pořadí jejich instrukce.

Samotná interpretace probíhá v cyklu, kde řídící proměnná začíná s hodnotou 0 a určuje index do pole `orderArray`, kde je získána číselná hodnota instrukce a ta je vložena jako klíč do slovníku s instrukcemi, čímž je získán přístup k objektu instrukce, která je na řadě. Poté je podle operačního kódu volen další postup programu, tedy provedení instrukce. Interpretace končí, když je hodnota řídící proměnné rovna délce pole `orderArray`.

Všechny proměnné jsou ukládány do slovníku rámců, kde klíčem je označení daného rámce (`GF`, `LF`, `TF`). To usnadňuje následnou kontrolu existence proměnné, jelikož je možné rozdělit její reprezentaci na označení rámce a její název (`GL@varName`). Není tedy nutné kontrolovat, v jakém rámci se proměnná nachází přímo, ale jen přeposlat označení rámce do slovníku a pokusit se nalézt proměnnou dle názvu.

Proměnné jsou ukládány jako objekty, které nesou typ proměnné a její hodnotu. Stejným způsobem jsou řešeny také elementy datového zásobníku.

Veškeré skoky jsou řešeny tak, že u dané skokové instrukce je jméno návěští, na které se má skočit, dáno jako klíč do slovníku s návěštími a tím je získáno pořadí instrukce, na kterou se má přejít. Z této hodnoty je získán index do pole `orderArray`, který je dekrementován, aby bylo zajištěno, že v dalším instrukčním cyklu bude prováděna právě instrukce označující dané návěští.

2 Implementovaná rozšíření

Rozšíření STATI umožňuje uložení statistik, tedy počtu instrukcí a definovaných návěští do výstupního souboru a rozšíření STACK zajišťuje podporu pro zásobníkové instrukce. Jedná se o rozšíření skriptu `interpret.py`.

2.1 STATI

Podpora vstupního parametru skriptu `--stats=file` určujícího jméno výstupního souboru se statistikami `file`. V případě, že je tento argument zadán vícekrát vytvoří se odpovídající počet souborů. Pokud není zadáno jméno souboru nebo je shodné se jménem skriptu, jehož je toto rozšíření součástí, provádění programu je ukončeno s chybovým návratovým kódem 10. Uspořádání statistik v souborech záleží na pořadí zadání následujících upřesňujících přepínačů tohoto rozšíření. Tyto přepínače mohou být také zadány opakovaně. Každý záznam je zapsán na samostatném řádku.

Parametr `--insts` zajistí uložení počtu vykonaných instrukcí během interpretace a parametr `--vars` slouží pro uložení počtu všech inicializovaných proměnných ve všech rámcích. Pokud je skript spuštěn s nějakou kombinací těchto parametrů, ale není zadán výstupní soubor, tedy pokud chybí parametr `--stats=file`, je

program ukončen s chybovou návratovou hodnotou 10. Pokud je zadán výstupní soubor, ale žádné další upřesňující parametry, je po úspěšném ukončení interpretace vytvořen prázdný soubor `file`.

2.2 STACK

Podpora zásobníkových verzí instrukcí základního řešení. Tyto instrukce pracují s datovým zásobníkem. Vstupní operandy vybírají od konce, tedy na vrcholu zásobníku je vždy poslední argument. Výsledky těchto instrukcí jsou opět ukládány na vrchol zásobníku.

3 Testování

Veškeré testování skriptu `interpret.py` a `parse.php` je zajištěno testovacím skriptem `test.php`. Při spuštění je zadán adresář k prohledání, a pokud je v něm nalezen nějaký test, je podle případných dalších parametrů tohoto skriptu spuštěn pouze nad jedním z testovaných skriptů, nebo nad oběma tak, že vstup je poslán do programu `parse.php` a výstup tohoto skriptu směřuje jako vstup do programu `interptert.py`.

Každý test se skládá ze 4 souborů s příponami `in`, `out`, `rc` a `src`. Soubor s příponou `src` je zdrojovým souborem, jedná se tedy o vstup pro testované skripty. Soubor s příponou `rc` obsahuje návratový kód testovaného skriptu. Pokud v adresáři chybí je vytvořen testovacím skriptem a je do něj uložena hodnota 0. Soubor s příponou `in` je vstupním souborem pro interpretovanou XML reprezentaci programu v IPPcode20 skriptem `interpret.py`. Pokud v adresáři chybí, je dotvořen testovacím skriptem a zůstává prázdný. Soubor s příponou `out` je soubor obsahující referenční výstup daného testovacího zdrojového souboru. Pokud v adresáři chybí, je dotvořen a zůstává prázdný.

Celkové testování poté probíhá na principu porovnávání získaných výstupů a návratových kódů z testovaných skriptů s referenčními hodnotami v souborech s testy, přičemž je nakonec vygenerována zpráva v HTML5 s výsledky na standardní výstup.

3.1 Vstupní parametry

Dovolují univerzálněji pracovat se skriptem a ovlivnit testovací scénář. Parametr `--directory=path` určuje adresář, ve kterém jsou testy hledány. Chybí-li je prohledáván aktuální adresář. Parametr `--recursive` určuje, že testy budou hledány i v podadresářích prohledávaného adresáře. Parametr `--parse-script=file` umožňuje zadat soubor pro analýzu zdrojového kódu. Není-li zadán je implicitně hledán soubor `parse.php` v aktuálním adresáři. Parametr `--int-script=file` je analogií předchozího parametru pro skript `interpret.py`. Parametr `--parse-only` určuje, že se bude testovat jen skript `parse.php`. Parametr `--int-only` je opět analogií předchozího parametru pro `interpret.py` a parametr `--jexamxml=file` určuje umístění souboru s balíčkem pro porovnávání XML.

Každý parametr lze zadat i vícekrát, ale musí nést stejnou hodnotu, tedy jména, či cesty k souborům musí být stejné, jinak je skript ukončen a chybovým kódem 10.

3.2 Průběh testu

Při spuštění testovacího skriptu jsou nejprve načteny a zkontrolovány parametry. Jedná se zejména o existenci vstupních souborů a adresáře, ale také o kontrolu kombinací parametrů. Poté je prohledáván adresář s testy, které jsou ukládány v poli jako hodnota do slovníku, kde klíč tvoří název adresáře, ve kterém se nachází. V této fázi jsou v daném adresáři (či adresářích při rekurzivním prohledávání), dotvářeny potřebné soubory k testování.

V další fázi dochází k postupnému spouštění jednotlivých testů a do slovníku, jehož klíčem je jméno adresáře a hodnotou slovník testů, jsou ukládány výsledné hodnoty tak, že jméno testu je klíčem k přístupu ke slovníku, kde je uložen výsledek porovnání návratového kódu získaného a referenčního, přes klíč `rc` a výsledek porovnání získaného výstupu s referenčním přes klíč `out`. Výstup se porovnává ale jen v případě, že je návratový kód stejný jako referenční a zároveň je roven 0.

V poslední fázi se prochází přes tyto vnořené slovníky a do standardního výstupu je generována zpráva s výsledky ve formátu HTML5.

Každá úspěšná hodnota je označena zeleně a neúspěšná červeně.