Implementační dokumentace k projektu do IPP 2017/2018

Jméno a příjmení: David Hříbek

Login: xhribe02

Zadání

Cílem projektu bylo vytvořit sadu skriptů (parse.php, interptet.py, test.php), které mají za úkol interpretovat nestrukturovaný imperativní jazyk IPPcode18.

Skript parse.php

Skript `parse.php` má za úkol načíst zdrojový kód ze standardního vstupu v jazyce IPPcode18, provést lexikální a syntaktickou analýzu kódu, a v případě bezchybného kódu vypsat XML reprezentaci kódu na standardní výstup.

Skript je navrhnut objektově, využívá 3 vlastní třídy: `Instruction`, `Writer` a `Statistics`. Třída `Instructions` zajišťuje načtení instrukce ze standardního vstupu a provedení lexikální a syntaktické analýzy. Třída `Writer` zajištuje generování XML reprezentace instrukce. Třída `Statistics` zajišťuje sběr dat o kódu, potřebných pro rozšíření `STATP`.

Hlavní smyčka programu je založena na jednom nekonečném cyklu, který při každé iteraci požádá o načtení další instrukce, pomocí funkce `loadInstruction`, třídy `Instruction`. Při každém úspěšném načtení instrukce, dojde k vygenerování XML reprezentace dané instrukce, pomocí třídy `Writer`. Pokud již není žádná instrukce na standarndím vstupu, smyčka se ukončí, třída `Writer` vypíše vygenerovanou XML reprezentaci na standardní výstup, a třída `Statistics` vypíše informace o kódu pro rozšíření `STATP` (pokud uživatel zadal příslušné přepínače).

Skript interpret.py

Skript interpret.py má za úkol načíst XML reprezentaci programu a tento program interpretovat s využitím standardního vstupu a výstupu.

Skript je navrhnut objektově a využívá tyto vlastní moduly: `argChecker.py`, `dataStack.py`, `errorHandler.py`, `frameHandler.py`, `instruction.py`, `instructionList.py`, `xmlParser.py`. Hlavním skriptem je `interpet.py`, který využívá většiny těchto modulů.

Zodpovědnost modulů

Modul `argChecker.py`

Kontrola argumentů skriptu.

Modul `xmlParser.py`

Tento modul má na starost načtení XML reprezentace programu ze standardního vstupu, provedení lexikální a syntaktické analýzy a naplnění instrukční pásky danými instrukcemi.

Modul `errorHandler.py`

Na tento modul jsou delegovány všechny výskyty chyb. Po obdržení zprávy o výskytu chyby dojde k vypsání chybové hlášky na standarndí chybový výstup a ukončení programu s příslušným návratovým kódem.

Modul `dataStack.py`

Modul realizuje datový zásobník. Umožňuje vkládat a odebídat hodnoty z datového zásobníku. Tento modul využívají instrukce `PUSHS` a `POPS`.

Modul `frameHandler.py`

Modul realizuje rámce (`LOCAL FRAME`, `TEMPORARY FRAME`, `GLOBAL FRAME`) a umožňuje nad těmito rámci provádět operace. Tento modul využívají zejména instrukce `CREATEFRAME`, `PUSHFRAME`, `POPFRAME`, `BREAK`, `MOVE`, `DEFVAR` a další instrukce, které potřebují zapisovat nebo číst z proměnných.

Modul `instuction.py`

Modul realizující jednu načtenou instrukci. Pro každou instrukci je uchováván operační kód, počet argumentů, a jednotlivé argumenty. Pro argumenty je uchováván typ a hodnota.

Modul `instructionList.py`

Tento modul realizuje instrukční pásku. Uchovává poslopnost načtených instrukcí ze standardního vstupu a číslo právě prováděné instrukce. Každá instrukce je realizována objektem třídy `Instruction`, a má přiřazeno číslo značící pořadí v kódu. Tato instrukční páska slouží pro interpretaci instrukcí. Tento modul přímo ovlivňují instrukce `JUMP`, `LABEL`, `JUMPIFEQ`, `JUMPIFNEQ`, `CALL` a `RETURN`.

Skript `interpret.py` s využitím metod a funkcí poskytovaných výšeuvedenými moduly provede interpretaci programu.

Skript test.php

Skript test.php slouží pro automatické testování skriptů `parse.php` a `interpret.py`.

Skript je navrhnut objektově a využívá 4 vlastní třídy: `Arguments`, `DirectoryScanner`, `TemporaryFile` a `HtmlGenerator`. Třída `Arguments` má za úkol zkontrolovat argumenty programu. Třída `DirectoryScanner` zajišťuje vyhledávání testovacích souborů v zadaných složkách a případné vytvoření chybějících souborů. Třída `TemporaryFile` zapouzdřuje práci s dočasným souborem a třída `HtmlGenerator` zajišťuje vypsání HTML výstupu na standardní výstup.

Skript nejprve zkontroluje argumenty programu a vytvoří dočasný soubor pro výstupy jednotlivých testů. Poté pomocí třídy `DirectoryScanner` vyhledá testovací soubory `.src` v zadaném adresáři, popřípadě v podadresářích. Pro chybějící soubory `.in` a `.out` vytvoří skript prázdné soubory, a pro chybějící soubory `.rc` vytvoří soubor s obsahem "0", vedle `.src` souborů. Poté začne samotné testování.

Pro každý test je daný soubor `.src` poslán na standardní vstup skriptu `parse.php` a XML výstup je přesměrován do dočasného souboru. Pokud návratový kód skriptu `parse.php` není 0, je tento kód porovnán s kódem v souboru `.rc`, pokud se kódy schodují, je **test považován za úspěšný**, **jinak neúspěšný**. Pokud návrtový kód skriptu `parse.php` je 0, přichází na řadu interpretace.

Je spuštěn skript `interpert.py`, který načte XML reprezentaci z dočasného souboru, vstup od uživatele ze souboru `.in` a provede interpretaci. Výstup skriptu je opět přesměrován do dočasného souboru. Pokud návratový kód interpretu není shodný s kódem v souboru `.rc`, je **test považován jako neúspěšný**. Pokud se kódy shodují, je výstup interpretu zapsaný v dočasném souboru porovnán se souborem `.out`. Pokud se výstup interpretu shoduje s očekávaným výstupem, je **test považován jako úspěšný**, **jinak neúspěšný**.

Po otestování všech testovacích souborů je vypsán na standardní výstup souhrn testování, roztříděný podle složek, zahrnující **názvy testů**, **návratové kódy skriptů**, **očekávané návratové kódy skriptů**, **shodnost výstupu interpretu** a **úspěšnost daného testu** ve formě HTML kódu.