Implementační dokumentace k 1. úloze do IPP 2021/2022

Jméno a příjmení: Dominik Pop

Login: xpopdo00

Základní popis:

Skript parse.php přijímá na standartní vstup zdrojový kód napsaný v jazyce IPPcode22. Tento soubor je v hlavním cyklu načítán řádek po řádku až do konce souboru. Každý načtený řádek je průběžně zpracováván, kontrolován a zapisován ve formátu XML na standartní výstup, popřípadě přesměrován do souboru.

Načítání:

O načítání se stará hlavní cyklus v pseudo části skriptu main. Ten pomocí funkce fgets načítá řádky a ukládá je jako řetězec do proměnné. Tento řádek je pak zbaven nepotřebných částí. Tedy komentářů a zbytečných bílých znaků. Řádek je pak rozdělen pomocí funkce explode na pole řetězců.

Kontrola:

Lexikální a syntaktická kontrola probíhá v průběhu hlavního cyklu. Začíná kontrolou povinné hlavičky, pokud je tato kontrola úspěšná přechází se na kontrolu instrukcí a jejich argumentů. O kontrolu instrukcí se stará slovník instrukcí implementovaný pomocí switche.

K následné kontrole argumentů pak slouží implementované funkce **check_params/symb/type/var/label**. Funkce check_params se stará o kontrolu správného počtu argumentů. Ostatní funkce mají pak na starost lexikální kontrolu datových typů argumentů a využívají k tomu hlavně regulární výrazy. Při této kontrole se tyto argumenty zároveň zpracovávají a ukládají ve správném formátu, tak aby mohly být později využity při zápisu.

Zápis:

Zápis do formátu XML probíhá po lexikální a syntaktické kontrole rovněž v rámci hlavního cyklu. Pro účely zápisu byla implementována třída **XML**, jejíž atributy a metody se starají o dodržení správného formátu. O vytvoření povinné hlavičky XML se stará metoda **make_head,** která je volaná ihned po kontrole hlavičky vstupního souboru. O zápis instrukcí metoda **make_instruction**, která volá metodu **add_argument**, pro přidání argumentů instrukce. O nastavení atributů, které využívají tyto metody se starají metody **__construct** a **setter**. XML soubor pak ukončuje metoda **end_xml**, volaná po konci hlavního cyklu.

Seznam funkcí:

- check_args($argc, $argv) – kontrola vstupních argumentů skriptu
- trim_comments($line) – oříznutí komentáře z řádku
- check_head($line_parsed, $xml) – kontrola povinné hlavičky
- check_params($line_parsed, $correct_params, $err_msg) – kontrola správného počtu argumentů
- check_symb/type/var/label($str) – kontrola datového typu
- replace_special_chars($str) – nahrazení speciálních znaků formátu XML

Třída XML:

Atributy:

- $encoding
- $version
- $order
- $opcode
- $num_args
- $types
- $param

Metody:

- __construct($encoding, $version) - konstruktor
- setter($opcode, $num_args, $types, $params) – nastavení atributů
- make_head($language) – zápis hlavičky XML formátu
- make_instruction() – zápis instrukcí
- add_arguments($num) – zápis agrumentů instrukce
- end_xml() – ukončení zápisu XML

Implementační dokumentace k 2. úloze do IPP 2021/2022

1. Interpret.py:

Základní popis:

Skript interpret.py buď pracuje se vstupním souborem s příponou ‚.src', nebo se standartním vstupem, ve formátu XML, který byl předpracován parserem. Z tohoto vstupu pak čte instrukce, zpracuje je, zkontroluje jejich validitu a vykoná je. Interpret dál může pracovat se soubory formátu ‚.in', nebo opět standartním vstupem, které slouží jako vstup instrukce READ. Pokud jsou ve vstupu obsaženy instrukce pro výpisy dat, je výsledek posílán na standartní výstup, popřípadě přesměrován do souboru.

Načítání:

Interpret načte vstup ze souboru/standartního vstupu a rozdělí jej na části pomocí knihovny **xml.etree.ElementTree**. Následně prochází jednotlivé části a získává z nich potřebné informace, u kterých ověřuje jejich validitu.

Instrukce a jejich argumenty:

Instrukce jsou zpracovávány v cyklu for, umístěném v pseudo části main. V tomto cyklu se volá funkce **check_instruction**, která má na starosti získání informací z instrukce, částečnou kontrolu těchto informací a následné uložení informací do objektu třídy **Instruction**. Tyto objekty jsou pak ukládány do listu instrukcí. Funkce check_instruction se pak stará o zpracování argumentů instrukce, voláním funkce **check_arg**, která zkontroluje informace o argumentu a uloží je do objektu třídy **Argument**. Tyto objekty jsou pak ukládány do listu argumentů jednotlivých instrukcí. List instrukcí je pak následně seřazen podle čísla pořadí instrukcí.

Návěští:

Návěští jsou zpracovávána při zpracovávání instrukcí v cyklu for. Jsou tedy zpracovávány při prvním průchodu vstupu interpretu. Jejich jména a pozice se pak ukládají do objektů třídy **Label** a tyto objety se ukládají do listu návěští.

Vykonání instrukcí:

Vykonání instrukcí probíhá v hlavním cyklu while, který se nachází v pseudo části main. V tomto cyklu se prochází přes list instrukcí a jednotlivé instrukce jsou jedna za druhou vykonávány. Před vykonáním instrukce probíhá extrakce informací z argumentů, kdy je za potřebí rozhodnout, jestli je argument konstanta nebo proměnná. O tom rozhoduje funkce **get_value_type**, která vrací hodnotu a typ proměnné/konstanty. Ukládání informací o proměnné má na starosti třída **Variable**. Po získání dat z argumentů probíhá sémantická kontrola informací, která nemohla být vykonána při načítání instrukce. U instrukcí, které potřebují k uložení své hodnoty proměnnou, také probíhá kontrola existence této proměnné. To je umožněno funkcí **does_var_exist**, která vrací referenci na objekt třídy Variable. Pokud proběhla kontrola v pořádku je instrukce vykonána, pokud ne je interpret ukončen s chybovou hláškou a vhodným výstupním kódem. O to se stará třída **Error** a její metoda **exit_with_error**.

Rámce a zásobníky:

Rámce jsou zpracovávány pomocí objektů třídy **Frame**. Kontrolu jejich existence má na starost funkce **does_frame_exist**, která vrací True/False na základě existence. Zásobník rámců je implementovaný pomocí **deque**, stejně jako všechny ostatní zásobníky (zásobník volání/dat).

Seznam funkcí:

- get_type(opcode, num) – vrací jakých datových typů mohou nabývat argumenty dané instrukce
- check_head(root) – kontroluje hlavičku vstupního XML souboru
- check_instruction(instruction, valid_ins) – kontroluje validitu instrukce, vytváří objekt instrukce, volá zpracování argumentů instrukce, přidává argumenty do instrukce
- check_arg(arg, type_list, type_ins, name) – kontroluje argument, vytváří objekt argumentu
- check_arg_value(value, type) – kontroluje správnost hodnoty argumentu
- does_label_exist(list, name) – kontroluje existenci návěští
- does_frame_exist(frame) – kontroluje existenci rámce

- does_var_exist(var) – kontroluje existenci proměnné
- convert_escape_seq(string) – mění escape sekvence na znaky
- get_value_type(type, value) – získá hodnotu a její datový typ z proměnné/konstanty
- both_int(type1, type2) – kontroluje jestli jsou obě hodnoty typu int

Seznam tříd:

- Args – zpracování argumentů
- Instruction – zpracování instrukcí
- Argument – zpracování argumentů
- Variable – informace o proměnné
- Label – informace o návěští
- Frame – reprezentace rámce
- Error – error handling

2. Test.php:

Základní popis:

Skript test.php je určen k otestování validity výstupů skriptů parse.php a interpret.py. Tester spustí jeden (-parse-only), druhý (-int-only), či oba skripty a porovná jejich výstupy s referenčními výstupy. Tester porovnává výstupový kód skriptů a jestliže je jejich i referenční kód nulový, tak porovnává i výstupní data, které si přesměrovává do ‚.out' souborů. Tyto soubory po svém dokončení maže, za předpokladu, že není specifikováno jinak příkazem ‚--noclean'. Tester má také možnost rekurzivního procházení podadresářů, které může být specifikováno příkazem ‚--recursive'. Výsledky testů pak tester posílá na standartní výstup ve formátu HTML5, který je zapotřebí si přesměrovat do souboru ‚index.html' a zobrazit v prohlížeči.

Kontrola vstupních argumentů:

O načtení vstupních argumentů se stará třída **Args**. Po načtení argumentů jsou použity metody **check_bool**, pro ověření validity argumentů, a **check_files**, pro ověření existence vstupních souborů. Jestliže je vstup v pořádku, přechází tester k procházení vstupního adresáře.

Procházení adresářů:

O procházení adresářů se stará funkce go_through_dir, volaná v pseudo části skriptu main nad vstupním adresářem (--directory). Tato funkce uloží všechny adresáře do listu adresářů a všechny soubory do listu souborů, které jsou seřazeny abecedně. Jestliže byl při spuštění zadán argument ‚--recursive', tak tester začne procházet rekurzivně všechny adresáře z listu adresářů, opět pomocí funkce go_through_dir. Poté co skončí procházení adresářů, začne funkce procházet všechny soubory z listu souborů a spustí nad nimi danou kombinaci skriptů.

Kontrola testů:

Poté co byli nad souborem spuštěny skripty je volána funkce **compare_output**, která porovná referenční výstupní kód, uložený v souboru ‚.rc', s výstupním kódem skriptu. Jestliže jsou oba kódy nulové, pak porovnává výstupy v souborech ‚.out'. Porovnávání výstupů je vykonáno buď přes Jexam, v případě kontroly parseru, nebo přes diff, v případě kontroly obou, nebo pouze interpretu. Jestli test uspěl, nebo ne, je následně vypsáno na standartní výstup ve formátu HTML5. Následně je volána funkce **clean_up**, která se stará o smazání všech souborů, které si tester při testování vytvořil. Volání této funkce je možné zakázat argumentem ‚--noclean'.

Seznam funkcí:

- go_through_dir ($dir, $args) – procházení adresářů, spuštění skriptů
- check_for_files($file, $dir) – kontrola existence testovacího příkladu
- compare_output($exit_code, $path, $args) – porovnání výstupu s referenčním výstupem
- clean_up($dir) – vyčištění souborů, které byly vytvořeny při testování
- print_wrong_tests($args) – vypíše všechny špatné testy

Seznam tříd:

- Args – zpracování argumentů