Implementační dokumentace k 1.úloze do IPP 2023/2024

Jméno a příjmení: Jindřich Halva

Login: xhalva05

Úvod

Cílem první úlohy bylo vytvořit skript parse.py v pythonu (verze 3.10.), který na vstupu obsahuje instrukce v jazyce IPPcode24 a převede jej do formátu XML.

Implementace

Kód je rozdělen do tří oddělených částí. V prvním oddílu s názvem FUNCTIONS jsou naimplementovány důležité funkce jako například funkce kontroly proměnných `check_var()`, funkce generující xml `generate_xml()` a další. U funkcí kontrolující formát proměnných, symbolů či typu jsem postupoval prvně logicky a moc jsem nepřemýšlel nad tím, jak si práci ulehčit, tudíž jsem postupoval metodou postupných větvení. Po čase jsem více nastudoval regulární výrazy a jak s nimi pracovat. Regulární výrazy jsem použil poprvé při kontrole symbolů.

V druhé části kódu ARGUMENTS_HANDLING probíhá kontrola argumentů. Jde pouze o velmi jednoduchou a jasnou sekci, bez potřeby komentáře.

Hlavní část MAIN_PART začíná hledáním hlavičky vstupního textu a inicializací seznamu s názvem `program`. Do tohoto seznamu se později vkládají prvky obsahující informace o aktuálně zpracovávané instrukci a jejích argumentech. Tento seznam slouží jako vstup pro již zmíněnou funkci `generate_xml()`. Každý prvek v tomto seznamu obsahuje “opcode” (název instrukce) a “argN” (argument daného typu “type” a hodnoty “value”). Text na vstupu se načítá po řádcích. Z Každého řádku se ořežou komentáře a rozdělí se na jednotlivé části podle mezer, vzniká seznam `instruction_list` obsahující vždy název instrukce a případně její argumenty. Do python obdoby switche vstupuje název instrukce. Jednotlivé case jsou uspořádány podle toho, jaký formát mají argumenty jednotlivých instrukcí. Pokud má například instrukce dva argumenty, jež mají být proměnná a symbol, hned poté co kód vleze do dané case, zavolá se funkce na kontrolu počtu argumentů `operands_num_check()`, poté proběhne kontrola proměnné a symbolu. Pokud projdou argumenty kontrolou, u symbolu se ověří, zda se v něm neskrývá proměnná (“type” zapisovaný do XML výstupu je u proměnné “var” a ne například “GF”). Pokud se nejedná o proměnnou, “type” argumentu je výraz před znakem “@”. Následuje přidání prvku na konec seznamu `program`. Ukázka z kódu může napomoct lepší představě o principu implementace:

```
    case "DEFVAR" | "POPS":
        operands_num_check(1)
        check_var(instruction_list[1])
        program.append({
            "opcode": first,
            "arg1": {"type": "var", "value": instruction_list[1]}})
```

Instrukce `DEFVAR` a `POPS` očekávají jeden argument a to typu “label”, provedou se tedy zmíněné kontroly a pokud projdou bez chyby, zapíše se tato instrukce na konec seznamu `program`.

Program končí vypsáním výsledného textu v podobě XML na standartní výstup, nebo ukončením s chybou danou příslušným návratovým kódem a chybovou hláškou vypsanou na standartní chybový výstup.