Implementační dokumentace k 2. úloze do IPP 2021/2022
Jméno a příjmení: Tadeáš Kachyňa
Login: xkachy00

Stručný popis programu

Program započne s načtením a kontrolou uživatelem zadaných argumentů. Následně pošle kód určený ke zpracování knihovně `xml.etree`. Ta zkontroluje správnost XML stromu, jestli je tzv. *well-formed*. Taktéž se zkontroluje samotné tělo tohoto stromu, což znamená, jestli jsou každá instrukce a argument správně pojmenováné a také jejich atributy. Poté se provede případné seřazení instrukcí podle atributu *order* a první průchod progamem ke zpracování návěští, aby byla v případě skoku dopředu již definována. Nyní již dochází k vykonávání hlavního těla skriptu, který zastřešuje jeden cyklus *while*, který zpracovává instrukce do té doby, dokud mu nějaké přichází. V každém cyklu dojde nejprve k získání operačního kódu instrukce a taktéž počtu argumentů. Ty se stejně jako samotné instrukce nejprve seřadí dle jejich tagů. Je vytvořena nová instance třídy `Instruction`. Ta si o sobě ukládá svůj operační kód, počet argumentů a také jejich obsah, který následně využije k vytvoření jistého počtu instancí třídy `Argument`. Zde dojde k několika kontrolám, které jsou společné pro všechny instrukce. To znamená, jestli je proměnná definovaná, jejich inicializace, abychom nečetli z prázdné proměnné, nedefinovali proměnnou na neexistujícím rámci. Taktéž je zavolána metoda převádející escape sekvence na jejich ekvivalentní reprezentaci v ASCII za pomocí vestavěné funkce `chr()`. Nejprve vyhledá regulárním výrazem všechny výskyty v daném řetězci. Tyto trojciferné hodnoty si uloží do pole, které následně přetypuje a převede do ASCII reprezentace. Ještě se provede případné přetypování všech hodnot argumentů na jejich skutečnou reprezentaci. Důvod je prostý. Program standartně hodnoty všech proměnných nebo konstant, ať již zadané uživatelem nebo přímo ze zdrojového kódu, ukládá jako řetězec. Proto pokud je třeba, tak celá čísla a pravdivostní hodnoty převádím přetypovávám. Nyní je již vše připravné, aby se mohla daná instrukce provést. Do proměnných si získám z instancí všechny hodnoty a typy argumentů. Za pomocí rozsáhlé "if-else"konstrukce se rozhoduji jakou instrukci právě vykonám. Každý cyklus se inkrementuje hodnota za jejíž pomoci se následně orientuji při skocích v programu. Instrukce jsou implementovány skutečně jednoduše a to i díky tomu, že většinu formalit, aby nedošlo k nedefinovanému chování, kontroluji již dříve. Ze zajímavějších implementací můžu úvest například instrukce pracující s rámci. Protože používám jen jednu strukturu pro uchování všech aktuálně platných proměnných, tak při instrukci `pushframe` si nejprve lokální proměnné, pokud již nějaký takový rámec existuje, převedu pomocí cyklu do pomocné struktury, kterou následně vložím na zásobník proměnných. A pak následně všechny proměnné s označením `TF@*` převedu na `LF@*`. Podobnou postup aplikuji i při instrukci `popframe`.

Datové struktury

Ve skriptu využívám několika datových struktur. Od slovníků až po samostatná pole. K uchování všech aktuálně platných proměnných využívám jeden slovník `varList`, ve kterém mám uložené jejich označení, hodnotu a typ. `labelList` pro návěští, několik polí pro seřazení instrukcí a argumentů. Následně využívám několika zásobníků - datový zásobník `dataStack` využívající instrukce `pushs` a `pops`, zásobník pro uchování proměnných které jsou v aktulně nedostupných LF rámcích `varStack` nebo zásobník pro volání instrukcí `callList`.

Popis tříd a jejich metod

Třída ProgramArgs

Třída starající se o počáteční běh programu, což znamená načtení, kontrolu a zpracování uživatelem zadaných argumentů. Ke zpracování využívám knihovnu `argParse`.
Metody:
`parseProgramArguments` - parsování argumentů
`checkProgramArguments` - kontrola a ověření správné kombinace argumentů
`parseProgramArgumentsPath` - ověření, zda-li soubory existují

Třída Program

Třída Program se stará o základní úkony, které je potřeba provést před samostaným vykonáváním instrukcí. To zahrnuje kontrolu XML stromu, kontrola správného pořadí instrukcí a přednačtení návěští.

Metody:
`checkStructionOfXMLTree`
`orderInstructions`
`findLabels`

Třída Instruction

Uchovává informace o instrukci a jejích argumentech.
Metody:
`checkOpCode` - kotrola správnosti operačního kódu

Třída Argument

Detailněji uchovává informace o jednotlivých argumentech, provádí jejich důslednou kontrolu, převádí escape sekvence a provádí typovou kontrolu jednotlivých hodnot.
Metody:
`checkArgumentsType` - provádí kontrolu zda-li argument odpovídá správnému typu, který by instrukce měla mít
`check` - důsledná kontrola, aby nedošlo k nesprávné manipulaci s těmito argumenty, tzn. neinicializace, nedefinice, špatný rámec, získání hodnoty argumentu, pokud je argument proměnná
`replaceEscapeSequences` - překládá escape sekvence na jejich ekvivalentní reprezentaci v ASCII
`checkTypeConversion` - převádí případnou špatnou reprezentaci hodnoty jejich přetypováním