Implementační dokumentace k projektu do IPP 2017/2018
Jméno a příjmení: Pavel Janko
Login: xjanko10

parser.php

Skript se skládá ze 3 hlavních tříd. První třídou je `Token` – ta slouží k reprezentaci jednotlivých tokenů zdrojového kódu. Mezi její atributy patří `name` (reprezentující název) a `value` (reprezentující hodnotu – ta může být prázdná). Metody této třídy slouží pouze k získání nebo nastavení hodnot atributů této třídy (čili getter a setter metody).

Další třídou je `Scanner`. Tato třída slouží k určení druhu tokenů po načtení sekvence znaků ze zdrojového souboru a následné vytvoření instance třídy `Token`, přičemž skrze konstruktor rovněž nastavuje počáteční hodnoty atributů těchto instancí. Takto vytvořené instance rovněž ukládá do pole tokenů, které se poté používá při vytvoření výstupního XML souboru, který obsahuje mezikód. Třída obsahuje pomocnou metodu pro rozdělení řetězců – ta se používá zejména u lexikálních kontrolách názvu proměnných, konkrétně oddělení identifikátoru rámce od samotného názvu proměnné.

Poslední třídou je `Parser`, tedy syntaktický analyzátor. Ta kontroluje, zdali posloupnost tokenů vyplývající ze zdrojového kódu je syntakticky platná, tedy že ji lze vygenerovat z gramatiky jazyka IPPcode18. Obsahuje metody pro zpracování jednotlivých částí gramatiky, jako například instrukce, její operandy apod. Rovněž generuje výstupní mezikód ve formátu XML. Rozhodl jsem se využít knihovnu `DOMDocument`.

V hlavní části programu (mimo třídy) se pouze kontrolují vstupní parametry skriptu a v případě jejich správnosti se vytváří instance třídy `Parser`.

interpret.py

Skript interpretu se skládá také ze 3 tříd. První třída pojmenovaná `Argument` slouží pro zpracování jak syntaktické, tak lexikální stránky argumentu (operandu) instrukce.

Následuje třída `Instruction`, která je důležitá zejména u exekuce jednotlivých instrukcí, kdy se na začátku každého kódového bloku instrukce invokuje metoda `check_args()`, která podle parametrů zajišťuje sémantickou správnost instrukcí (respektive jejích argumentů).

Poslední a hlavní třída skriptu je `Interpret`, která vykonává samotný zdrojový kód jednotlivých instrukcí a stará se o exekuci instrukcí ve správném pořadí s tím, že toto pořadí mohou některé instrukce, jako například `RETURN`, měnit. Nejprve se vykonají instrukce `LABEL`, aby se naplnil seznam návěstí spolu s informaci o pozici instrukce a tím zajistila správná funkcionalita skokových instrukcí apod. Interpret při redefinici proměnných jednoduše přepíše jejich původní obsah.

Mimo třídy se pak provádí mimo jiné kontrola parametrů skriptu – například specifikace zdrojového souboru s mezikódem v XML a následně také platnost toho XML souboru. Rozhodl jsem se využít knihovnu `etree`, která obsahuje metody pro jednoduchou práci s XML souborem, jako například vyhledávání podle tagu.

test.php

Struktura

Spouštění testů probíhá po adresářích s tím, že při použití rekurzivního přepínače se do jednotlivých adresářů zanořuje. Pojmem test je myšlen každý soubor, který končí příponou `.src` s tím, že se následně případně dogenerují soubory s příponami, které jsou pro test zapotřebí – tedy `.in`, `.out` a `.rc`.

Po provedení skriptů se vypíše souhrn výsledků v HTML s tím, že jsou výsledky všech testů zapsány v tabulce pod sebou, přičemž na levé straně je název složky, ze které soubor pocházel, a na pravé straně je název testu (respektive souboru obsahující test) a výsledku.

Omezení

Testovací skript není psán objektově, jelikož by dle mého názoru OO návrh nijak výrazně nevylepšil jak přehlednost, tak práci s programem.