IPP Projekt - 1. úloha

Jakub Hlava (xhlava52)

17. března 2021

Základní popis návrhu

Při návrhu analyzátoru kódu v jazyce IPPcode21 jsem se snažil o co nejjednodušší a nejspolehlivější řešení. Výsledkem byl návrh analyzátoru, který provádí analýzu v několika oddělených krocích: načtení celého kódu do paměti, očištění zdrojového kódu od neužitečných částí (např. komentáře, nadbytečné mezery, apod.), ověření přítomnosti korektní hlavičky, ověření lexikální a syntaktické správnosti pomocí šablon instrukcí a regulárních výrazů a vygenerování výsledného XML.

Příprava kódu k analýze

Celý kód je načten do paměti a při načítání rozdělen po řádcích, protože v jazyce IPPcode21 mohou mít instrukce maximálně řádek a usnadní mi to práci později. Poté jsou všechny řádky "očištěny", to znamená, že jsou odstraněny všechny komentáře, všechny sekvence bílých znaků jsou zredukovány na jednu mezeru, mezery na začátku a konci se oříznou a nakonec se z pole řádků odstraní prázdné řádky, tím jsem si zajistil, že řádky obsahují pouze instrukce (a pokud na řádku není instrukce, pak můžu analýzu spolehlivě ukončit s chybovým kódem). Na závěr se provede kontrola přítomnosti hlavičky `.IPPcode21` na první pozici v poli a její případné odstranění.

Analýza zdrojového kódu

Jádrem analýzy zdrojového kódu je třída `InstructionTemplate`, která uchovává počet a typy argumentů instrukce a obsahuje metodu `match` pro srovnání řádku s instrukcí s šablonou, činnost této metody je blíže popsána níže. Objekty třídy `InstructionTemplate` jsou uchovávány v poli, které je klíčované operačními kódy známých instrukcí, neznámé instrukce tedy lze vyloučit pouhou kontrolou přítomnosti operačního kódu mezi klíči.

Sestavení výstupního XML

Výstupem metody `match` jsou objekty tříd `Argument` a `Instruction`. Z nich je jednoduše pomocí knihovny `SimpleXML` vygenerován XML dokument, dosazením dat z objektů na správná místa dle specifikace v zadání a pomocí knihovny `DOMDocument` je dokument zformátován, aby byl přehledný a snadno čitelný i člověkem.

Pomocné třídy a metody

`InstructionTemplate` a metoda `match`

Objektům třídy `InstructionTemplate` se v konstruktoru předává pole typů neterminálů, které instrukce přijímá (tím je udán i počet a pořadí). Při analýze řádku, na kterém čekáme instrukci potom z řádku oddělíme operační kód, tento zkusíme použít jako klíč v poli šablon. Pokud je operační kód validní, získáme tím objekt šablony, na kterém voláme metodu `match`.

Metoda `match` v argumentu přijímá řádek s instrukcí. Tento rozdělí na operační kód a operandy. Operandy poté iteruje a pokouší se na ně aplikovat regulární výrazy, kterými postupně ověří jak lexikální správnost, tak např. analyzuje neterminál `<symb>` a blíže určí, zda jde o proměnnou či konstantu. V případě, že se nepodaří aplikovat regulární výraz podle typu neterminálu, pak analýza končí chybou.

Z validních argumentů, které odpovídají regulárním výrazům se tvoří objekty třídy `Argument`, které obsahují typy a hodnoty ve formátu pro vložení do XML dle specifikace v zadání. Po kontrole celé instrukce se z operačního kódu vytvoří a vrátí objekt třídy `Instruction`.

`Argument` a `Instruction`

Jedná se o jednoduché pomocné třídy s metodami, které usnadňují a zpřehledňují ukládání výstupů analýzy a generování výstupního XML z těchto výstupů.