Implementační dokumentace k 1. úloze do IPP 2018/2019
Jméno a přijímení: Petr Marek
Login: xmarek69

Projekt jsem se rozhodl vypracovat v několika samostatných souborech, kde každý soubor (s vyjímkou vstupního – *parse.php*) sestává z nějaké třídy, ze které se vytváří objekty pro snazší práci. Pro všechny možné hodnoty „exit kódu“ jsou definované konstanty. Následuje popis jednotlivých souborů a způsob jejich spolupráce.

Parse.php
Program začíná své provádění zavoláním funkce `main()` v *parse.php*. Ve funkci `main()` se nejprve zavolá funkce `checkProgramArgs()`, která otestuje, jestli jsou správně zadané argumenty (tzn. správný počet a případně formát, dle zadání). Potom se zavolá funkce `getSTDIN()`, která načte a vrátí standardní vstup (jako soubor) a vrácený soubor se uloží do proměnné `$source`. Poté se vytvoří objekt typu `Parser`, který slouží již k samotné kontrole lexikální a syntaktické správnosti kódu. Za účelem této kontroly se na zmíněném objektu zavolá funkce `parse()`, které se předá jako argument soubor `$source` (standardní vstup). Až `parser` dokoncí svou činnost (funkci `parse()`), zavoláme na něm z `main()` ještě funkci, která uloží do proměnné `$parsedInstructions` pole instrukcí, které objekt parser vytvořil (pouze pokud funkce `parse()` nezjistila žádnou lexikální či syntaktickou chybu ve vstupním souboru). Po získání pole instrukcí se vytvoří další objekt, objekt typu `IppXML`, který slouží pro konečné vytvoření a vypsání XML reprezentace programu na standardní výstup. Na tomto objektu se zavolá funkce `instructionsToXML()`, které se jako argument předá pole (dříve získaných) instrukcí `$parsedInstructions`. Tato funkce vytvoří XML dokument a vypíše jej na standardní výstup. Nakonec se zavolá exit() s odpovídajícím kódem (tedy 0), čímž se program ukončí.

Parser.php
Jak již bylo zmíněno, tento soubor je složený z třídy `Parser` a obsahuje funkci `parse()`. V této funkci se nejprve načítají řádky předaného souboru (standardního vstupu), dokud program nenarazí na povinnou hlavičku .IPPcode19. Pokud není nalezena (případně je vložena pouze jako komentář), program skončí a vrátí odpovídající chybu. V opačném případě začne postupně (v cyklu) načítat jednotlivé řádky předaného souboru a provádět s nimi následující akce: Nejprve se zavolá nad načteným řádkem funkce `removeComment()`, která odstraní komentář, pokud se na řádku vyskytuje. Vzápětí jsou z takto vzniklého řádku odstraněny bílé znaky na okrajích řetězce. Poté se zkontoluje, zda řádek není prázdný. Pokud ano, tak se přeskočí na další iteraci cyklu, jinak se zavolá funkce této třídy `parseLine()`, které se jako argument předá získaný (a případně upravený řádek). Funkce `parseLine()` rozdělí řádek do pole, přičemž jako oddělovač použije bílé znaky. Tím získáme pole ve kterém (v případě korektně zapsaného zdrojového kódu v jazyku IPPcode19) bude na první pozici název instrukce a případně na dalších pozicích argumenty, pokud instrukce nějaké potřebuje. Z tohoto pole vytovříme další pole, ve kterém budou pouze argumenty (pro pozdější snadnou práci). Nakonec v této funkci zavoláme funci `checkInstructionName()` a jako argumenty ji předáme název instrukce a pole argumentů. Ve funkci `checkInstructionName()` vytvoříme z každého prvku pole argumentů objekt typu Argument. Dále se rozhodne na základě instrukce co bude následovat. Pokud není název instrukce rozpoznaný, tak se program ukončí s chybou 22. Jinak se porovná očekávaný počet argumentů se skutečným. Pokud je rozdílný, tak program skončí s chybou 22, jinak se přesune k další kontrole – zda odpovídají typy jednotlivých argumentů očekávaným typům. Pokud ne, tak program skončí opět s chybou 22. Pokud typy odpovídají, tak se vytvoří objekt typu `Instruction` (kterému se jako argumenty pro konstruktor zašle název instrukce a pole argumentů (objektů typu `Argument`)). Typ argumentů se samozřejmě kontroluje a vyhodnocuje pouze u instrukcí, které očekávají nějaké argumenty. Nakonec se vytvořený objekt instrukce přidá do globálního pole instrukcí, které si objekt parser uchovává. Nakonec se vrátí provádění programu do cyklu ve funkci `parse()` a cyklus přejde k další iteraci.

Instruction.php
Tento soubor obsahuje třídu, reprezentující instrukci. Objekty typu Instruction vytváří objekt typu Parser a využívá je objekt typu IppXML pro vytvoření XML dokumentu.

Argument.php
V tomto souboru je třída Argument, která velmi usnadňuje práci (kontroly argumentů) v objektu typu `Parser`. Obsahuje množství pomocných funkcí (např. různé kontroly argumentu).

XML.php
Tento poslední soubor obsahuje třídu `IppXML`, obsahující funkci `instructionsToXML()`. Ta slouží pro konečné vytvoření a vypsání XML reprezentace programu na standardní výstup. Jako parametr vyžaduje pole instrukcí (tj. pole objektů typu Instruction). Jelikož případné chyby jsou ošetřeny v dříve zmíněných souborech, tak se zde již neošetřují chyby. Funkce pracuje s knihovnou `XMLWriter`. Po vytvoření objektu typu `XMLWriter` a počáteční inicializaci se vytvoří XML hlavička a element „program“ (s atributem vyžadovaným zadáním). Poté se v cyklu vytváří z předaného pole instrukcí postupně elementy „instruction“ a v případě, že příslušná instrukce obsahuje nějaké argumenty, tak také tyto se vypíší. Nakonec se XML dokument vytiskne na standardní výstup.