# Implementačná dokumentácia k 1. úlohe - IPP 2019/2020

Maroš Geffert (xgeffe00@stud.fit.vutbr.cz)

## 1 Analýza zdrojového kódu IPPcode20 - parse.php

Analýza zdrojového kódu IPPcode20 je implementovanána v triede `Parse`. Zdrojový kód je čitaný po riadkoch zo štandartného vstupu. Využívam jeden while, ktorý spracuje jednotlivé riadky pomocou regulárnych výrazov (ďalej len `REGEX`) a nasledným switchom ktorý definuje o akú inštrukciu sa jedná a aké argumenty (variable, symbol, type...) má vygenerovať. Pri každej inštrukcii sa pomocou `REGEXOV` kontrolujú typy a počty argumentov. Po analýze a po prijatí danného riadku (daný riadok je zapísaný lexikálne a syntakticky správne) je nasledné vegenerovaný XML kód za použitia knižnice `XMLwriter()`. Program končí správne ak prečíta všetky riadky, vygeneruje správny XML kód a skončí v koncovom stave. V prípade chybného vstupu proram vypíše chybu a skončí s príslušnou navratovou hodnotu.

### 1.1 Implementácia rozšírenia

K 1. Úlohe bola možnosť implementácie rozšírenia ktorý zbiera a následne poskytuje štatistiky ktoré zbiera pri preklade. Programu zadávame argument `-stats=file` a za tým dobrovnolné argumenty `-loc` (zistí počet riadkov s inštrukciami), `-comments` (zistí počet riadkov s komentármi) `-labels` (počet definovaných návestí) a `-jump` (počet inštrukcií pre podmienené skoky). Tieto rozšírenia su implementované formou počítadla, ktoré sa inkrementuje za každým kedy riadok príjme inštrukciu/ dostane sa na riadok v ktorom sa nachádza komentár/ na návestie a pod. Na konci analýzy sa zobierané štatistiky uložia do zadaného súboru `file`.

## 2 Interpretácia zdrojového kódu (interpret.py)

Interpretácia zdrojového kódu IPPcode20 je implementovaná v súbore `interpret.py`, skript tak isto používa aj dalšie moduly.

### 2.1 Moduly

- **Modul** `iArgsErrors.py` - Modul kontroluje argumenty skriptu a tak isto tam sú definované error výstupové hlášky.
- **Modul** `iDataStack.py` - Modul definuje dátový zásobník. Umožňuje vkladať/odoberať hodnoty zo zásobníku.
- **Modul** `iInstructionList.py` - Modul, ktorý definuje inštrukčnú pásku, uchováva postupnosť načítaných inštrukcií, počet všetkých inštrukcií, počet dokončených inštrukcii a číslo aktuálne spracovávanej inštrukcie. Tento modul úzko ovlivňuje inštrukcie `JUMP, LABEL, JUMPIFEQ, JUMPIFNEQ, CALL a RETURN` pretože uchováva informácie o pozícií v kóde, a preto sa dá v kóde jednoducho pohybovať.
- **Modul** `iXmlParser.py` - Modul, ktorý načíta vstupný XML kód, skontroluje jeho štruktúru, spracuje jednotlivé inštrukcie do slovníka a tie následne posiela do hlavného "modulu" `interpret.py` ktorý inštrukcie ďalej spracuváva.
- **Modul** `iRootElement.py` - Modul, ktorý vytvorí `root element`, kde sú uložené všetky potrebné informácie o XML štruktúre a posiela ho ďalej do modulu `iXmlParser.py` na ďalšie spracovanie.

`interpret.py` je telo programu, ktorý za pomoci pomocných modulov interpretuje kód na štandartný výstup a končí s očakávaným chybovím výstupom.

# 3 Testovací skript (test.php)

Skript `test.php` slúži pre automatické testovanie skriptov `interpret.py` a `parse.php`. Má viacero prepínačov ako napríklad (`parse/int-script, parse/int-only, directory, jexamxml, recursive`). Skript je navrhnutý objektovo a má 2 triedy.

- Arugments
- iFileScanner

## 3.1 Arguments

V triede `Arguments` sa spracúvávajú jednotlivé argumenty, prepínače a následne spracované informácie posiela do hlavnej triedy `iFileScanner`.

## 3.2 iFileScanner

Trieda `iFileScanner` príjme spracúvané argumenty skontroluje jednotlivé testovacie súbory s príponami (`.src, .rc, .in, .out`) (Ak súbory `.rc, .in, .out` neexistujú tak ich skript dogeneruje) a prejde do funkcie `Scanner()`, kde začína spracovávať jednotlivé vstupy. Funkcia `Scanner()` je rozdelená na tri podfunkcie `iBoth(), iParseOnly(), iIntOnly()`. Ktorá funkcia sa vykoná rozhodujú prepínače zadané v argumentoch. Pre každý test je daný súbor `.src` zaslaný na štandartný vstup do skriptu `parse.php` a výstup je potom uložený do dočasného súboru s príponou `.tmp.in`. Pokiaľ je návratový kód 0 tak sa posiela ďalej ako argument do skriptu `interpret.py` (v prípade prepínača –parse-only kontroluje návratové kódy so súborom `.rc`, ak nájde zhodu tak test je úspešný). Interpret urobí interpretáciu jednotlivého XML súboru a poskytne návratovú hodnotu a výstup, ktorý je zapísaný do dočasného súboru s príponou `tmp.out`. Návratová hodnota sa porovná so súborom `.rc` a výstup z `.out`. Ak obe porovnania sú úspešné test je prijatý ako úspešný a pripočíta sa do predpripravného slovníka ako test "úspešny". Všetky neúspešne testy sa zapíšu ako neúspešné.

## 3.3 Generovanie HTML súboru

Po vykonaní všetkých testov následuje funkcia na vygenerovanie HTML súboru, ktorý poskytuje preľadné informácie o úspešnosti jednotlivých testov.