Vytvořil: Michal Řezník
Login: xrezni28

BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

Implementační dokumentace k projektu v předmětu IPP 2020/2021

Zadání:

Cílem projektu je vytvořit skripty: parse.php, interpret.py, test.py, které budou interpretovat nestrukturovaný imperativní jazyk IPPcode20.

Řešení:

Skript parse.php

Skript by měl načíst zdrojový kód v jazyce IPPcode20 ze standartního vstupu. Provede syntaktickou a lexikální analýzu. Za podmínky bezchybného kódu provede výpis XML reprezentace daného kódu.

Pro kontrolu lexikální a syntaktické správnosti jsem použil v cyklu funkci *fgets,* po které následovala funkce *explode,* díky čemuž jsem vždy získal již rozdělené pole na daná slova. Poté už byla práce s polem slov triviální. Použil jsem switch, který zajišťuje všechny správné možnosti a filtruje ty nesprávné.
Pro kontrolu escape sekvencí v řetězci, nahrazujících některé nepovolené znaky, je zde implementována funkce *Check_codes*, která jednoduchým cyklem prochází obsah řetězce a kontroluje znaky následující za znakem \.

Součástí daného switche jsou i části, které v případě správného zápisu ihned tisknou na standartní výstup XML reprezentaci daného kódu.
Hlavní cyklus programu se spouští až po nalezení hlavičky, bez které by nemohl být řádný kód v jazyce IPPcode20 vůbec kontrolován.

Skript interpret.py

Skript je určen k načtení XML reprezentaci programu a jeho následného interpretování.

Interpret je oproti skriptu *parse.php* navržen objektově, díky čemuž bylo daleko snadnější samotný kód procházet a pracovat s jednotlivými instrukcemi a jejich argumenty. Samotný skript obsahuje několik tříd:

Instruction – třída jejímž hlavním úkolem bylo zjednodušení práce během interpretování kódu. Jednotlivé objekty jsme vkládali do seznamu. Který jsme pak mohli snadno procházet a kontrolovat její argumenty a ostatní vlastnosti jako například posloupnost *order* v instrukcích.

Arg – další třída, která patřila k úplnému základu. Objekty dané třídy patřily vždy k samotné instrukci. Díky čemuž samotná kontrola probíhala velmi hladce.

Vytvořil: Michal Řezník
Login: xrezni28

Frames - Jedná se o třídu, jejíž vytvoření mi dalo nejvíce práce. Protože mé dosavadní zkušenosti s OOP byly spíše teoretické. Avšak po pár chvílích strávených nad nefunkčními částmi kódu jsem došel k řešení, které mi přišlo nejelegantnější. Základní instrukce jsem implementoval jako funkce.
V samotné třídě hraje asi nejdůležitější roli funkce **TYPE_VALUE_VAR**, která mi byla nápomocna při každé kontrole nebo při práci s argumenty.

Dále jsem použil třídu **Data**(data na zásobníku), která však není nijak zajímavá.

Pro průchod XML reprezentací jsem využívám knihovnu *xml.etree.ElementTree*. Práce s ní byla příjemná až na chvíle, kdy jsem byl nucen prohledávat dokumentaci pythonu, která se mi zdá velmi nepřehledná oproti ostatním jazykům.

Po základní části, která spočívala ve zpracování argumentů, nahrání jednotlivých instrukcí, návěští a argumentů do seznamů následuje cyklus, jež má za úkol projít celou reprezentaci a současně ji interpretovat.

V případě úspěšného dokončení interpretace se skript ukončí s návratovou hodnotou 1 a případně na standardní datový výstup vypíše výsledek.

Skript test.php

Bohužel vlastní chybou a to špatným odhadem potřebného času na vytvoření jsem daný testovací skript nestihl vytvořit a tak jsem byl odkázaný na malé množství testů, které jsem vymýšlel a zároveň testoval a pracoval s nimi již během implementací jednotlivých částí skriptů.