Implementační dokumentace k projektu do IPP 2017/2018
Jméno a příjmení: Vladimír Dušek
Login: xdusek27

parse.php

První ze sady skriptů, parse.php je napsaný v jazyce PHP 5.6 a analyzuje vstupní kód v jazyce IPPcode18. Provádí lexikální a syntaktické kontroly a generuje výstupní formát tohoto kódu v XML.

Skript jsem implementoval bez využití objektově-orientovaného paradigmatu. Snažil jsem se využívat alespoň principy strukturovaného programování a program členit do funkcí.

Po spuštění skriptu se jako první zavolá funkce `test_args` pro ošetření argumentů programu. Pokud nebyl program spuštěn správně, je zavolána funkce `call_error`, které se přes parametry upřesní s jakým návratovým kódem má program ukončit a jakou chybovou zprávu vypsat na standardní chybový výstup. Vyžádal-li si uživatel nápovědu, je zavolána funkce `call_help`, která nápovědu vypíše.

Po ošetření argumentů je volána funkce `test_intro`, která zkontroluje formát úvodního řádku. Ten musí obsahovat text ".IPPcode18", kde nezáleží an velikosti písmen. Komentáře a všechny bíle znaky před jsou ignorovány.

Po těchto kontrolách již následuje čtení kódu ze standardního vstupu po řádcích pomocí funkce `fgets`. Jsou smazány komentáře (vše do konce řádku za znakem #) a bílé znaky mezi slovy jsou zredukovány na jednu mezeru. Ta je následně použita jako oddělovač a pro každý řádek získáváme pole slov. První z těchto slov je operační kód instrukce, následován jejími operandy. Ve switchy tedy rozpoznáme o jakou instrukci se jedná. Pro dané instrukce jsou poté volány funkce: `test_num_args_inst` pro zkontrolování počtu operandů, `test_var` pro zkontrolování, jestli operand je správně zapsaná proměnná, `test_sym` pro symbol a podobně. Po těchto kontrolách je funkcí `generate_xml` vygerován XML kód pro danou instrukci, která využívá `XMLWriter`.

interpret.py

Interpret.py je skript napsaný v jazyce Python 3.6. Pomocí parametrů je mu předána XML reprezentace kódu v jazyce IPPcode18, který následně lexikálně, syntakticky i sémenticky zkontroluje a v případě jeho správnosti jej interpretuje.

V tomto skriptu jsem se již snažil využívat principy objektově-orientovaného programování.

Nejprve je instanciována třída `Config`, nad kterou je zavolána metoda `parse_options`, která zkontroluje správnost argumentů skriptu. V případě chyby je volána funkce `error`. Té se podobně jako v prvním skriptu přes parametry upřesní s jakým návratovým kódem má program ukončit a jakou chybovou hlášku vypsat.

Dále jsem implementoval třídu `ParserXML` ve které se parsuje vstupní soubor v XML. Je zkontrolována syntaktická správnost souboru a následně je pomocí `xml.etree.ElementTree` vytvořen strom. Jsou zkontrolovány jednotlivé elementy a soubor je uložen jako seznam instrukcí.

Instrukce je reprezentována třídou `Instruction`. Obsahuje atributy pro reprezentaci pořadí instrukce, operačního kódu a seznamu operandů. Dále je definována metoda pro kontrolu počtu operandů.

Operand instrukce je opět třída (`Argument`). Obsahuje atribut (`type` pro typ operandu (var, symb, label, type) a (`content` pro obsah (identifikátor proměnné, hodnota konstanty). Nad třídou jsou implementovány metody pro kontrolu jestli je operand proměnná, symbol, návěští nebo typ.

Třída `Variable` slouží pro vnitřní reprezentaci proměnné. Má funkci pouze jako například struktura z jazyka C. Obsahuje atributy pro typ (int, string, bool) a hodnotu.

Poslední a také nejrozsáhlejší třída, je třída `Interpret`. Obsahuje atributy `GF`, `TF`, `LF` což jsou slovníky a reprezentují rámce. Dále seznam pro instrukce, seznam pro návěští a 3 zásobníky, pro rámce, pro data a pro volání (instrukce CALL a RETURN). Ta je instanciována po naparsování vstupního XML souboru a je ji předán seznam instrukcí. V metodě `execute` pak čte jednu instrukci za druhou a volá příslušně metody. Pro každou instrukci je implementována metoda. V těch se poté provádí většina kontrol a jsou volány další pomocné metody. Například metody pro kontrolování existence rámce, definování proměnné, pro získání nebo uložení hodnoty proměnné nebo získání a uložení jejího typu a podobně.

test.php

Poslední skript je napsán opět v jazyce PHP 5.6 a slouží pro testování předcházejících dvou skriptů.

Přes parametry je skriptu zadán adresář s testama a cesty ke skriptům, které se mají testovat. Dále je možnost hledat testy rekurzivně ve všech podadresářích zadaného adresáře. Každý test obsahuje 4 soubory, případně jsou dogenerovány s výchozíma hodnotama.

Po zpracování argumentů jsou nalezeny cesty ke všem .src souborům pomocí funkce `glob` (případně její rekurzivní variantou) a uloženy do globálního pole `src_paths`.

Dále jsou pro každý test vytvořeny i cesty k ostatním souborům a je načten očekávaný návratový kód. Tyto informace jsou přes parametry předány funkci `evaluate_tests`, která postupně skripty spouští. Jsou li návratové kódy podle očekávání, tak pomocí unixové utility `diff` zkontroluje i výstup. Nakonec jsem smazány dočasné soubory pro výstupy skriptů.

Pro provedení všech testů je zavolána funkce `generate_html`, která vypíše na standardní výstup výslednou přehledovou stránku v HTML 5 o výsledcích jednotlivých testů.