Implementační dokumentace k projektu do IPP 2017/2018
Jméno a příjmení: Jiří Peška
Login: xpeska05

Parse

Úkolem scriptu parse.php je vytvořit xml ze zadaného zdrojového kódu napsaného v jazyce IPPcode18.
Parse postupně prochází vstupní zdrojový kód řádek po řádku, přeskakuje prázdné řádky, počítá komentáře a řádky s kódem vkládá do pole. Nejdůležitější částí scriptu je třída **ParserClass**, která se o parsování stará. Obsahuje v sobě asociativní pole obsahující instrukce, kde pro každou instrukci je seznam operandů, které daná instrukce přijímá. Naproti této mapě je pak testováno, jestli instrukci byl předán odpovídající počet operandů a odpovídající typ operandu.

Když dojde parse na konec vstupního souboru, tak začne postupně procházet vzniklé pole, to parsovat – získá instrukci, operandy, které opět naparsuje, zkontroluje validitu jejich formátu a vytvářet výslednou hierarchickou strukturu instrukcí a jejich operandů, které jsou naparsovány na typ a hodnotu. Formát je kontrolován regulárními výrazy.
Po dokončení parsování a vytvoření výsledné hierarchie se předá výsledné pole instrukcí funkci **toXML()**, která vytvoří výstupní xml pomocí třídy **XMLWriter**.

Interpret

Interpret jako vstup bere zdrojovy kod zapsany v xml a tento zdrojovy kod interpretuje.
Interpret.py obsahuje několik tříd reprezentující entity, vyskytující se při interpretování programu a obsahující operace, které slouží k manipulaci s daty v těchto entitách. Definoval jsem třídy Variable, Frame, LocalFrameContainer a Instruction. Třída Variable slouží k uchování typu promenné a její hodnoty. Třída Frame v sobě obsahuje slovník, ve kterém jsou ukládány proměnné a operace, které slouží pro přístup, vyhledání a úpravu hodnot těchto proměnných. LocalFrameContainer představuje zásobník rámců, co nějž lze operacemi push a pop manipulovat s rámci. Z pohledu proměnných je chování konzistentní se samotným rámcem. Když se hledá proměnná, tak je nahlédnuto do horního rámce na zásobníku. Pokud je proměnná nalezena, tak se provedou patřičné operace, pokud není, je nahlášena chyba.

Krom těchto tříd mám ještě definováno několik funkcí, které se slouží pro parsování proměnných či hodnot, zjištování existence proměnných v rámcích a kontrolu jejich validity. Funkce umožňují jednoduchou implementaci instrukcí, ve kterých se s proměnnými pracuje.

Postup interpretace

Vstupní xml naparsuji a inicializuji pole, ve kterém každý řádek bude reprezentovat řádek kódu. Takže lze pak jednoduše přes programový čítač instrukcí PC postupně procházet program v tomto poli. Přístup na index mi vrátí instrukci, která se má provést. Krom inicializace pole se provádí kontrola formátu řetězců, číselných hodnot apod.
Při načtení řetězce se nahradí všechny escape-sekvence odpovídající hodnotou (odpovídajícím znakem).

Při parsování xml zdrojového kódu inicializuji ještě labely, kde řádek, na němž je instrukce LABEL vložím jako hodnotu do slovníku, kde klíčem je název tohoto labelu. Tento slovník bude sloužit například pro skoky, volání funkcí a pod.

Po úspěšné inicializaci „instrukční pásky“ lze program začít interpretovat. Na začátku je programový čítač PC nastaven na hodnotu nula a podle provedené instrukce se mění jeho hodnota.
Pro každou instrukci je vytvořena funkce, která tuto instrukci reprezentuje a pomocí slovníku instrukcí je vždy zavolána odpovídající funkce, která reprezentuje prováděnou instrukci. Toto volání je implementováno tak, abych mohl volat instrukce jednotně a nebyl závislý na počtu parametrů, se kterými instrukce operují. Zde je úryvek kódu volání instrukcí:

instrukctionCall[opCode][0](*currentParams)

Původně jsem chtěl použít funkci globals(), která v sobě obsahuje všechny definované funkce v souboru, ale z programátorského hlediska mi to nepřišlo nakonec správné, ale líbilo se mi, že bych nemusel mít slovník instrukcí, kde hodnota představuje funkci instrukce. Volání instrukcí by tak bylo otázkou třech řádků kódu.

Pokud při interpretaci nedojde k nějaké chybě, tak je skript ukončen kódem nula a na standartním výstupu jsou vypsaná data z interpretu vypsána instrukcí WRITE.

Test

Skript test.php slouží ke spuštění sady testů nad skripty parse.php a interpret.php.
Dle zadaných parametrů při spuštění testu se bude prohledávat daný adresář buď rekurzivně, což znamená, že bude prohledávat i podadresáře zadaného adresáře, nebo nerekurzivně, přičemž prohledá pouze zadaný adresář.

Při prohledávání adresáře / adresářů hledá soubory, které mají suffix .src. Pokud na nějaký narazí, tak z něj vypreparuje jméno testu, které bude použito později pro přístup k odpovídajícím souborům s koncovkou .out, .in a .rc. Soubor s koncovkou .src předá skriptu parse.php a zjistí návratový kód. Pokud není roven nule, zapíše se do souboru s koncovkou .out a porovná se s referenčním souborem, který končí suffixem .rc.

Pokud se soubor .rc shoduje se souborem .out, tak výsledek testu bude „OK“ a je zapsán do slovníku, který uchovává dvojice test:výsledek, jinak se zapíše jako výsledek hodnota „Fail“.

Pokud návratový kód je roven nule, tak se během parse.php nevyskytla žádná chyba ve zdrojovém kódu a může se pokračovat k interpretu. V tomto případě byl vytvořen soubor končící koncovkou .xml, do nějž byl zapsát zdrojový kód v xml a tento soubor je předán interpretu.

Spustí se interpret do nějž je ze přesměrován soubor .in a analogicky se zkontroluje návratový kód, který se zapíše do souboru .out. Pokud není roven nule, porovná se soubor .out se vzorovým souborem .rc a podle toho, jestli se liší, se do slovníku zapíše patřičná hodnota pro daný test. Jinak se do souboru zapíše hodnota 0 a výsledek interpretace, a až nyní se porovná výsledný soubor .out se souborem .rc a zjistí se rozdíly.

```
exec("python3.6 $param_int_script_file --source=$cwd/$testname.xml < $cwd/$testname.in", $interpret_result, $interpret_exit_value);
```

Soubory se porovnávají příkazem diff, které jsou volány pomocí php funkce exec().
Po provedení všech testů se předá slovník s výsledky funkci, která z něj vygeneruje na výstup přehlednou html stránku (html kód) se shrnutím testů a tabulkou, ve které jsou uspořádány názvy jednotlivých testů a jejich výsledky.
Pokud budeme chtít přímo html stránku, stačí přesměrovat výstup z testu do nějakého souboru a ten v prohlížeči otevřít.