Implementační dokumentace k projektu do IPP 2017/2018
Jméno a příjmení: Vladan Kudláč
Login: xkudla15

1. Úvod

Dokumentace je spolu s vestavěnou nápovědou psaná v češtině. Programová dokumentace je stejně jako samotný kód psána v angličtině.

2. Analyzátor kódu v IPPcode18

Program parse.php převádí zdrojový kód v IPPcode18 na XML za pomocí knihovny *XMLWriter*. Při výběru knihovny pro práci s XML jsem se rozhodoval mezi *XMLWriter* a *SimpleXML*. Protože zpracovávám vstupní soubor po řádcích a postupně vytvářím XML, vybral jsem *XMLWriter*, který je méně náročný a pro sekvenční vytváření XML přímo určený.

Program je vytvořen s ohledem na budoucí rozšiřitelnost jazyka IPPcode, neboť veškeré instrukce jazyka jsou uloženy v poli, kde klíčem je název instrukce (s velkými písmeny) a hodnotou je pole určující typy argumentů vyžadované danou instrukcí. Přidání nové instrukce znamená přidání položky do tohoto pole. Ukázka pole instrukcí (pouze některé instrukce):

```
$instructions = [ //var=variable; symb=constant or variable; label=label
      "MOVE" => ['var', 'symb'],
      "CREATEFRAME" => [],
      "WRITE" => ['symb'],
      "DEFVAR" => ['var'],
      "CALL" => ['label']
];
```

U každého řádku se nejprve zjistí instrukce a vyžadované parametry z pole instrukcí. Poté se volají funkce dle typu parametru, které parametry zkontrolují. Výstupní XML se generuje po celou dobu. Při identifikování instrukce se vygeneruje otevírací tag *instruction*, po úspěšném zpracování argumentu se ihned generuje tag *argX*.

V programu používám vlastní třídu *Stats* pro ukládání statistik, kterou předávám funkcím napříč programem. PHP předává objekty odkazem, tím jsem se vyvaroval použití globálních proměnných. Třída obsahuje pouze veřejné atributy: počet řádků kódu, počet komentářů, soubor pro zápis statistik a pořadí výpisu statistik.

Odevzdáno bylo rozšíření STATP, které bylo přidáno dodatečně. U interpretu bylo rozšíření STATI vypisující statistiky již od počátku, čímž jsem se vyhnul problémům s nevhodným návrhem.

3. Interpret XML reprezentace kódu

Pro práci s XML soubory využívám *xml.etree.ElementTree - The ElementTree XML API*. Knihovna mapuje celé XML do objektu. Nejprve získám kořenový element a poté v cyklu procházím vnořené elementy. Objekty získané knihovnou disponují řadou užitečných funkcí, kterých jsem chtěl využívat. Na druhou stranu pořadí elementů není zaručeno, bylo nutné projít vnořené elementy *instruction* a umístit je seřazené do pole. Pole instrukcí obsahuje odkazy na objekty typu *xml.etree.ElementTree.Element*, díky čemuž mohu využívat funkcí pro práci s XML.

Seznam instrukcí procházím 2x. Při prvním průchodu se zpracovávají pouze instrukce *LABEL* a kontroluje se pořadí instrukcí (atribut *order*). Při instrukci skoku musí program znát umístění instrukcí *LABEL*, aby bylo možné na dané místo přesunout ukazatel na následující instrukci.

Před započetím prací jsem navrhl několik objektů, se kterými v programu pracuji. Třída ***Enviroment*** představuje pracovní prostředí jednoho procesu. V celém programu je vytvářena jen jedna instance. Třída obsahuje zásobníky a rámce. ***Variable*** slouží k ukládání hodnot, obsahuje hodnotu a datový typ, ten je vždy řetězec, hodnota je uložena v odpovídajícím datovém typu. Instance třídy *Variable* může ale nemusí vždy představovat proměnnou. Pokud je uložena v proměnné, tak se pracuje s ukazatelem, změna v instanci změní přímo hodnotu uloženou například na zásobníku. Funkce díky tomu získají proměnnou a nemusejí se zajímat, kde se tato proměnná fyzicky nachází. Pokud chci získat hodnotu symbolu, získám ji také jako instanci *Variable*, to proto, aby bylo možné hodnotu jednoduše uložit do proměnné a také pro co nejobecnější návrh systému. Poslední třídou je ***Argument***, ta reprezentuje argument instrukce. Obsahuje typ argumentu (var, type, int, …) a hodnotu argumentu. Z objektu *Argument* lze získat objekt *Variable*.

4. Testovací rámec

Testovací rámec nemá implementované žádné rozšíření. Vytváří přehlednou *HTML* stránku s jednoduchou grafikou využívající *UNICODE* symboly pro znázornění stavu testu. Testy jsou rozčleněny na úspěšné a neúspěšné. Dále jsou tyto kategorie seřazeny podle názvu složky a názvu testu. Výstupní stránka byla testována na 5 prohlížečích pod operačním systémem Windows 10 (Google Chrome, Mozilla Firefox, Opera, Internet Explorer a Microsoft Edge).

Skripty jsou spouštěny pomocí funkce *exec*. Nejdříve se porovná návratová hodnota parseru a pokud se shoduje s očekávanou, přistoupí se ke spuštění druhého skriptu – interpretu. Pokud i ten skončí s očekávanou návratovou hodnotou, dojde k porovnání očekávaného a skutečného výstupu. Výstup porovnává nástrojem *diff*, který spouští rovněž funkcí *exec*.

Program pro testování vytváří během své činnosti 3 dočasné soubory. Pokud nedojde k běhové chybě testovacího programu, jsou soubory opět odebrány. Soubory nejsou odebrány při chybě, neboť obsah může v takovém případě odhalit příčinu selhání skriptu.