Jméno a příjmení: Rostislav Navrátil
Login: xnavra57

1 Analyzátor kódu v IPPcode18 (parse.php)

1.1 Zpracování parametrů

Pro zpracování vstupních parametrů je implementována funkce s názvem `Params`, která jako parametr obsahuje objekt `flags` třídy `flagClass`. Funkce obsahuje cyklus, ve kterém pomocí regulárních výrazů identifikuje vstupní parametry skriptu. Následně dochází k upravení dat v objektu `flags` podle příslušných vstupních parametrů. Při nerozpoznání vstupního parametru je zavolána funkce `erroroutput`, která se stará o ukončení programu s patřičnou návratovou hodnotou.

1.2 Zpracování IPPcode18

Pomocí `file_get_contents("php://stdin")` se načte kód IPPcode18 a pomocí funkce `explode` se uloží po řádcích do datového pole. Dále se provede odstranění mezer a tabulátorů na začátku každého řádku a odstranění komentářů. Až poté se hledá první povinný řádek ".IPPcode18" a proto je možné mít komentáře před prvním povinným řádkem. Dále probíhá zpracování jednotlivých instrukcí. Instrukce jsou ve switch case rozděleny do osmi skupin, podle toho jaký mají operační kód a jaké operandy požadují.

1.3 XML výstup

Po zpracování všech instrukcí se volá funkce `XmlOutput`, která jako parametr obsahuje pole objektů instrukcí `instructions`. Pro generování výstupu je použit modul `DomDocument`. Funkce obsahuje cyklus ve kterém se naplní atributy modulu hodnotami ze zpracovaných instrukcí a výsledek se pomocí funkce echo dostane na standartní výstup.

1.4 Rozšíření STATP

Při vykonávaných jednotlivých instrukcí se inkrementuje hodnota proměnné `LocNumber`, která značí počet zpracovaných instrukcí. Pro zjištění počtu komentářů se inkrementuje hodnota proměnné `CommentNumber`. Komentáře před povinným řádkem IPPcode18 se do výsledného počtu nepočítají. Po zpracování všech instrukcí se ověří zda skript obsahoval vstupní parametr pro výpis statistik a podle toho se provede výpis statistik.

2 Testovací rámec (test.php)

2.1 Zpracování parametrů

Pro zpracování vstupních parametrů je implementována funkce s názvem `Params`, která je implementována stejným způsobem jako funkce `Params` ve skriptu parse.php

2.2 Zpracování zdrojových souborů

Pro zpracování zdrojových souborů s koncovkou .src je implemtována funkce `SourceFile`. Funkce vrací datové pole, které obsahuje jednotlivé cesty k testům. Pro rekurzivní hledání testů v podadresářích je použito `RecursiveDirectoryIterator`.

2.3 Generování chybějících souborů

Další volanou funkcí v řadě je `GenerateMissingFiles`, která jako parametr obsahuje datové pole z předchozí funkce. Probíhá zde generování chybějících souborů s koncovkou .in a .out.

2.4 Zpracování skriptu parse.php a interpret.py

Ve funkci `ParseProcess` probíhá vyhodnocení testů na skriptu parse.php a ve funkci `InterpretProcces` vyhodnocení testů na skriptu interpret.py, které v `ParseProcces` skončili s návratovou hodnotou nula. Spouštění skriptů parse.php a interpret.py se provádí pomocí funkce `exec`. Ve funkci `InterpretProcces` se vytváří dočasný soubor, který obsahuje XML reprezentaci programu a slouží jako vstup pro skript interpret.py. V případě, že jméno dočasného souboru již existuje, vygeneruje se nový název.

2.5 Třídění testů

Pro třídění testů je implemntována funkce `TestSort`, jejíž výstupy jsou datové pole úspěšných a neúspěšných testů.

2.6 Generování HTML5 výstupu

O generování HTML se stará funkce `HTMLgenerate`, která má v parametrech dvě datové pole. Jedno obsahuje úspěšné testy a druhé neúspěšné testy. HTML šablona je uložená v proměnné a obsahuje inline CSS styly. Pomocí cyklů se zpracovávají data z datových polí a vkládají se do HTML šablony.

3 Interpret XML reprezentace kódu (interpret.py)

3.1 Zpracovaní parametrů

Pro zpracování vstupních parametrů je implementována funkce s názvem `params`, která jako parametr obsahuje objekt `flags` třídy `flagsClass`. Funkce obsahuje for cyklus, ve kterém pomocí regulárních výrazů identifikuje vstupní parametry skriptu a následně dojde k upravení dat v objektu `flags` podle příslušného vstupního parametru. Při nerozpoznání vstupního parametru dojde k zavolání funkce `erroroutput`, která se stará o ukončení programu s patřičnou návratovou hodnotou.

3.1 Zpracování XML souboru

Pro zpracování XML souboru je implementována funkce s názvem `xml_process`, která má za parametr objekt `flags`, z kterého použije relativní, či absolutní cestu k XML souboru. Pro načtení samotného XML souboru s jazykovým kódováním UTF-8 je použit modul `codes` a pro samotné zpracování XML souboru je použit modul `xml.etree.ElementTree`. Funke vrací pole instrukcí, které se nazývá `instructions`. Během zpracování XML souboru může dojít k nalezení chyby, která povede k ukončení programu s návratovou hodnotou značící chybný vstupní XML soubor.

3.2 Lexikální analýza

Další volanou funkcí je `lexical_analysis` s parametrem `instructions`, která se stará o lexikální analýzu instrukcí. Operační kódy jsou rozděleny do osmi skupin podle toho jaké operandy obsahují. Podle operačního kódu příchozí nezpracované instrukce se určí do jaké skupiny patří a provede se analýza operandů, které obsahuje. Analýza může objevit chyby, které povedou k ukončení programu na základě lexikální chyby, nebo chybného vstupního XML souboru.

3.3 Interpretace instrukcí

O samotnou interpretaci instrukcí se stará funkce `interpret` s parametrem `instructions`. Obsahuje dva while cykly. Prvním cyklem se v poli instrukcí `instructions` zjistí na jakých pozicích jsou instrukce obsahující návěští. Pozicí je myšleno pořadí v poli instrukcí. Druhým cyklem se opět prohledá pole instrukcí `instructions`, ale tentokrát se budou zpracovávat všechny instrukce. Pomocná proměnná u cyklů má název `x` a značí pořadí právě zpracovávané instrukce. Při skokových instrukcí se prohledá pole návěští `labels` a hodnota `x` se změní na následující pozici za zvoleným návěštím. Proměnnou `x` ke skoku používá i instrukce return a call. Při instrukci defvar dochází k zavolání funkce `is_declared`, která zabraňuje redefinici proměnné a ukončuje program s návratovou hodnotou 59.

3.4 Rozšíření STATI

Při vykonávání jednotlivých instrukcí se inkrementuje pomocná proměnná `stati_count`, která značí počet vykonaných instrukcí. Po vykonání instrukcí createframe, popframe, defvar dochází, nebo může docházet ke změně počtu proměnných, a proto se volá funkce `max_count_var`, která pomocí funkce `count_var` zjistí aktuální počet proměnných a v `max_count_var` se porovná s nejvyšším naměřeným počtem a případně nahradí hodnotu největšího počtu proměnných novou největší hodnotou. K měření počtu proměnných a instrukcí dochází při každém spuštění programu a hodnoty se do souboru vypisují, jen když je na vstupu skriptu požadovaný parametr.