Implementační dokumentace k 1. úloze do IPP 2020/2021
Jméno a přijmení: Dominik Horký
Login: xhorky32

Dokumentace ke skriptu 'parse.php'

Skript typu filtr, který načte ze standardního vstupu text – kód v jazyce IPPcode21, na kterém provede syntaktické a lexikální kontroly a na výstup vytvoří XML reprezentaci programu.

Hlavní část skriptu

Samotný skript je rozdělen na části oddíly „main" a funkce. Je to víceméně pouze ze zvyku (programování v C). Oddíl „main" obsahuje definice globálních proměnných použitých při parsování vstupu. Zároveň se zde provádí ihned po začátku skriptu kontrola argumentů, se kterými byl skript spuštěn. V případě, že kontrola argumentů proběhne úspěšně (a není program spuštěn s parametrem `--help`) se vypíše hlavička XML souboru. Následuje parsování vstupu.

Funkce parse

Ke každé funkci byl uveden krátký komentář, který by měl stručně vystihovat její účel. Nejdůležitější funkcí skriptu je funkce `parse`, která načítá ze standardního vstupu řádky kódu, dokud nenarazí na konec souboru (vstupu). Jelikož si myslím, že je pro uživatele pohodlnější, když překladač hlásí, na jakém řádku je která chyba, rozhodl jsem se i tento skript vést podobným duchem, a proto skript počítá načtené řádky ze vstupu (i když to zadání nevyžaduje).
Tuto informaci využívám poté při volání funkce `parseError`, která se volá, pokud skript najde syntaktickou či lexikální chybu ve vstupu. Nalezená chyba ve vstupu ukončí skript s vhodným chybovým kódem a chybovou hláškou (pomocí funkce `error_log`).
Funkce `parse` mimo jiné rozdělí načtený řádek vstupu do pole `$splitted` tak, aby byly vynechány bílé znaky a dalo se se vstupem potom lépe pracovat. Pokud vstup-text obsahuje více mezer mezi slovy (parametry, instrukcemi) než 1, v poli se to projeví tak, že prvky budou prázdným řetězcem, proto se prvky na těchto indexech vynechají a použije se první index, který neobsahuje prázdný řetězec.
Parsování vstupu funguje na principu switche, který je komentáři rozdělen na část instrukcí bez parametrů, s jedním, dvěma a třemi parametry. Dále jsou v "casech" pospolu instrukce, které mají stejné pořadí a typy parametrů (samozřejmě i počet). Vždy se na začátku parsování zkontroluje správný počet parametrů pro danou instrukci, poté se zkontrolují samotné parametry – ty mohou byt proměnnou, symbolem, návěštím či datovým typem. Poté parsování skončí a načítá se další řádek vstupu.
Speciálními případy jsou komentáře, mezery, konce řádku (v takovém případě se načítá další řádek vstupu) a hlavička. Pokud se nejedná o žádnou instrukci, může to být jedině hlavička nebo neplatná instrukce.
Pří kontrolách vstupu jsem vzal v úvahu i komentáře (připojené k posledímu argumentu funkce, oddělené či zakomentované instrukce, apod.). Takové kontroly se provádí napříč většinou funkcí a velkou roli zde hraje hlavně funkce `checkForHashtag`, která vráti index prvního nalezeného znaku `'#'` (začátek komentáře) v řetězci.

Pomocné funkce

Funkce `parse[Label/Symbol/Type/Variable]` používají na kontrolu správného formátu daného vstupu regex, jehož platnost se ověřuje pomocí funkce `preg_match`. Nejsložitější funkcí je `parseSymbol`, jelikož v případě, že se jedná o nějakou hodnotu, se provádí i kontrola typu (tedy, že pokud je před `@` uveden například `int`, tak že daná hodnota je opravdu pouze celým číslem). Zároveň v případě, že jde o hodnotu, se provádí nahrazení problematických znaků pro XML za jejich ekvivalenty (jedná se o znaky `&, <, >`).
Funkce `parseArguments` provádí kontroly správných parametrů skriptu a jejich správného počtu a také vypisuje nápovědu v případě argumentu `--help`. Text pro nápovědu byl převzat ze zadání a vhodně upraven.

Použité návratové kódy

10: neplatné spuštění skriptu (tzn. špatný počet argumentů nebo neplatný argument)
21: špatný formát hlavičky nebo neexistující hlavička `.IPPcode21`
22: neplatná instrukce
23: špatný parametr či jeho formát, neočekávaná hodnota, neplatný typ proměnné nebo špatný počet parametrů instrukce