Implementační dokumentace k 1. úloze do IPP 2021/2022

Jméno a příjmení: Jiří Mládek

Login: xmlade01

Struktura skriptu

Na začátku skriptu definuji konstanty pro chybový návratový kód. Po definicích následují pomocné funkce, po nich funkce pro kontrolu a zápis operandů instrukcí. Následují funkce pro skupiny instrukcí, a nakonec hlavní tělo skriptu. Pro vytváření xml reprezentace dat používám knihovnu **`XMLWriter`**.

Průběh skriptu

Nejprve zkontroluji argumenty, se kterými je skript spuštěn. Každý řádek ze standardního vstupu ukládám postupně do pole `$lines[]` pomocí funkce `file()`. Z každého řádku poté odstraním komentáře (vše, co je za znakem #, až do konce řádku.) pomocí funkce `preg_replace` (ta nahradí požadované prázdným řetězcem). Poté ještě odstraním z každého řádku přebytečné bílé znaky na začátku a na konci, pomocí funkce `trim()`.

Poté už následuje hlavní smyčka programu, kde postupně procházím řádky vstupu. Jelikož jsem odstranil komentáře a odstranil zbytečné bílé znaky (na začátku a na konci řádku), tak **procházím pouze řádky, které se nerovnají prázdnému řetězci**. Podle bílých znaků rozdělím obsah řádků na tokeny (pole `$tokens[]`) pomocí funkce `preg_split()`. Zkontroluji, zda je první token neprázdného řádku hlavička programu `".IPPCODE22"`. Pokud ano, nastavím příznak `$correct_header` na true. První token každého řádku změním na velké písmo pomocí funkce `strtoupper()`, kvůli tomu, že u instrukce nezáleží na velikosti písma a na výstup se musí vypsat velkými písmeny.

Poté zpracovávám instrukce s jejich operandy pomocí konstrukce `switch`, která se řídí prvním tokenem na řádku. Instrukce jsem si rozdělil do 8 skupin podle toho, jaké operandy přijímají. Instrukce bez operandů zpracovává funkce `empty_instr()`, instrukce přijímající jeden operand <var> funkce `var_instr()`, atd. Každá skupina instrukcí má pevně daný počet operandů, který instrukce očekává. V těchto funkcích dochází ke kontrole počtu operandů, zapsání elementu <instruction> a jeho atributů. Nakonec jsou z těchto funkcí postupně volány funkce pro zpracování proměnných, symbolů, návěští a datových typů (podle skupiny instrukcí).

O kontrolu proměnné se stará funkce `is_var()`, o kontrolu návěští `is_label()` a o kontrolu datového typu funkce `is_type()`. Zde již přímo pomocí regexu (pomocí funkce `preg_match()`) dochází ke kontrole správnosti zápisu. Pokud dojde ke shodě, tak se potřebné zapíše do xml.

Pokud se jedná o operand typu <symb>, tak dojde k zavolání funkce `is_symb()`. Ta rozliší, zda se jedná o proměnnou nebo konstantu. O kontrolu konstanty se stará funkce `is_constant()`, která podle datového typu konstanty zavolá s konkrétním parametrem funkci `write_constant()`. Ta zkontroluje pomocí regexu správný zápis konstanty. Popíšu zde, jak jsem vyřešil kontrolu konstanty typu „string“, tu totiž neřeším pouze podle regexu. Nejprve odstraním escape sekvence pomocí funkce preg_replace(). Díky tomu poté mohu pomocí regexu zkontrolovat, aby se v textu nenacházel znak ‚\‘.

Jak u proměnné, tak u návěští a konstanty typu string dojde k nahrazení speciálních znaků implicitně díky knihovně `XMLWriter`.

Konec skriptu

Po ukončení smyčky `for` mám už v paměti výsledný výstup. Jelikož používám knihovnu `XMLWriter` a ta do svého výstupu implicitně přidává nový řádek, tak kvůli testování tento nový řádek odstraním pomocí funkce `trim()`. Nakonec vypíšu dokončenou xml reprezentaci z paměti na standardní výstup.