Implementační dokumentace k 1. úloze do IPP 2020/2021
Jméno a příjmení: Tomáš Čechvala
Login: xcechv03

1 Main body

Pre implementáciu Analyzátoru kódu jazyka IPPcode21, som sa rozhodol použiť metódu, v ktorej spracuvávam text zo štandardného vstupu, ktorý načítavam riadok za riadkom v cykle, ktorý skončí keď je štandardný vstup prázdny, alebo ak nastane chyba. Komentáre začínajúce znakom '#' sú automaticky preskočené, to isté platí aj pre prázdne riadky a ďalej sa s nimi v kóde nepracuje. Program na začiatku skontroluje, či sa na prvom riadku vstupu nachádza hlavička (.IPPcode21), ak sa tam hlavička nenachádza alebo je nesprávna program končí s exit kódom 21. Následne program prechádza na ďalší riadok, ktorý rozdelí podľa počtu výrazov na danom riadku (php funkcia `explode()`). Na začiatku každého riadku by mala byť inštrukcia, ktorá je case insensitive. Platnosť tejto podmienky zisťujeme pomocou `switch(strtoupper($splitted[0]))`, kde inštrukcie sú rozdelené do skupín podľa počtu a typu argumentov. Ak sa tam nenachádza žiadna inštrukcia, alebo inštrukcia nie je definovaná program končí exit kódom 22. Ďalej program zisťuje platnosť argumentov mnou vytvorenými funkciami a ak sú správne realizuje XML výpis pomocou php funkcií `createElement()`,`appendChild()` a `setAttribute()`. XML výpis, ktorý vytváram obsahuje instruction, order, opcode a príslušné argumenty (tak ako bolo uvedné v zadaní).

2 Funkcie

2.1 Spracovanie premenných

Na spracovanie premenných používam funkciu `findArgumentVar()` s argumentom:
`$input` - string, v ktorom hľadáme premennú
Funkcia hľadá znak @ (php funkcia `strpos()`), ak ho nájde rozdelí string na 2 časti a tie porovná pomocou príslušných regulárnych výrazov (php funkciou `preg_match()`), ak podmienky platia znamená to, že sa jedná o premennú a že jej zápis je správny, následne vráti celý string naspäť. Ak by niektorý z argumentov neplatil program sa skončí (exit kód 23).

2.2 Spracovanie type

Na spracovanie správnosti type pri inštrukcii READ používam funkciu `findArgumentType` s argumentom:
`$input` - string, v ktorom hľadáme type
Funkcia hľadá platný regulárny výraz pre type v stringu, ak ho nájde a syntax platí vracia celý string naspäť. Inak program skončí (exit kód 23).

2.3 Spracovanie konštánt

Na spracovanie konštánt používam funkciu `findArgumentConstant()` s argumentom:
`$input` - string, v ktorom hľadáme konštantu
Funkcia hľadá znak @, ak ho nájde rozdelí string na viacero častí a tie porovná pomocou príslušných regulárnych výrazov (php funkciou `preg_match()`), ak podmienky platia znamená to, že sa jedná o konštantu a že jej zápis je správny, následne vráti celý string naspäť. Ak by niektorý z argumentov neplatil program sa skončí (exit kód 23).

2.4 Spracovanie labelu

Na spracovanie labelu používam funkciu `findArgumentLabel()` s argumentom:
`$input` - string, v ktorom hľadáme label
Funkcia hľadá platný regulárny výraz pre label v stringu, ak ho nájde a syntax platí vracia celý string naspäť. Inak program skončí (exit kód 23).