Implementační dokumentace k 1. úloze do IPP 2020/2021

Jméno a příjmení: Michal Pyšík
Login: xpysik00

Popis průběhu analýzy a překladu

Analyzátor implementovaný skriptem `parse.php` čte zdrojový kód napsaný v jazyce `IPPcode21` po jednotlivých řádcích pomocí `while` cyklu, dokud nenarazí na konec souboru, přičemž vždy ignoruje řádky obsahující pouze bílé znaky a komentáře (ty v případě sběru statistik samozřejmě započítá). Po kontrole povinné hlavičky očekává pouze řádky s jednotlivými instrukcemi a jejich operandy.

Jelikož jsou syntaktická pravidla v jazyce IPPcode21 velice striktní (každá instrukce musí mít vlastní řádek, počet jejích operandů je vždy fixně daný), nebylo nutné na syntaktickou kontrolu nutné např. sestrojovat zásobníkový automat, ani sepisovat všechna pravidla bezkontextové gramatiky generující daný jazyk. Každý řádek je rozdělen na jednotlivé lexémy funkcí `explode`. Přepínačem `switch` se podle počtu těchto lexémů rozhodne o kolika argumentovou funkci se jedná, pomocí druhého přepínače `switch` se potom první lexém (vždy název instrukce) porovnává s jednotlivými instrukcemi očekávající daný počet argumentů.

Lexikání analýza jednotlivých lexémů poté ve voláné funkci `lex_arg_check` probíhá většinou pomocí regulárních výrazů, v případě názvů instrukcí a literálů s velice omezenou kardinalitou (třeba validní bool je pouze `bool@true` nebo `bool@false`) probíhá pomocí přímého porovnávání řetězců. V případě vybraných instrukcí které přijímají jako některý argument `symbol` zjistíme o jaký typ literálu se jedná, a to pomocí funkce `symbol_check`.

Po kontrole celého řádku a za předpokladu, že nebyla nalezena žádná chyba, probíhá vždy výpis názvu instrukce a jejích jednotlivých argumentů do bufferu výstupního XML dokumentu, který je však na standarní výstup vypsán až po kontrole celého vstupního souboru. Pro tyto účely byla použita knihovna `XMLWriter`.

Zmíněné pomocné funkce

- `Lex_arg_check($arg_type, $argument)` – Vrací `true` pokud je řetězec `$argument` platným lexémem daného druhu (předaným jako řetězec parametrem `$arg_type`), jindy vrací `false`
- `symbol_check($symbol)` – Zjistí zda je řetězec `$symbol` platným symbolem, a pokud ano, vrací řetězec obsahující název jeho typu (například `'int'`), jinak vrací prázdný řetězec

Rozšíření STATP

Zpracování argumentů skriptu probíhá bez použití externích knihoven. Jednotlivé statistiky jsou reprezentovány třídou `Stat`, které obsahují název dané statistiky, název souboru kam se má vypsat, a seznam čísel všech řádků kam se mají v daném souboru vypsat.

Jelikož jsem měl původně problém s přesným pochopením zadání, bylo možné jakoukoli danou statistiku zapisovat pouze do jednoho souboru najednou. To jsem však poté doplnil implementací pole `$extra_stat`, kam se v případě výpisu již obsazené statistiky do dalšího souboru ukládají řetězce ve tvaru `'název_statistiky''název_souboru''číslo_řádku'` a ty se při jejich pozdějším užití oddělí funkcí `explode`.

Počítadla jednotlivých statistik se samozřejmě inkrementují za použití různých technik při samotné analýze. Za zmínku stojí ošetření statistik o skocích, jestliže přečtená skoková instrukce odkazuje na již definované návěstí, jedná se o zpětný skok. Jinak se uloží do pomocného pole a až po přečtení celého zdrojového kódu se rozhodne zda se jedná o dopředný nebo špatný skok podle toho, zda bylo dané návěstí kdekoli v kódu definováno.

Na závěr probíhá otevření, výpis statistik, a zavření jednotlivých souborů. Ukazatele na otevřené soubory jsou ukládány do asociativního pole `$opened_files`, kde klíč vždy tvoří název daného souboru. Tímto se zjednodušuje výpis statistik do souborů a také lze po dokončení výpisu jednoduše všechny soubory postupně zavírat při vyprazdňování tohoto pole.