Implementačná dokumentácia k 1.úlohe do IPP 2020/2021

Meno a priezvisko: Jakub Sokolík
Login: xsokol14

1 Cieľ úlohy

Cieľom tejto úlohy bolo implementovať parser pre jazyk IPPcode21. Implementácia riešenia sa nachádza v súbore **parse.php**. K úlohe bolo implementované rozšírenie **STATP**, ktoré sa nachádza v tom istom súbore ako samotné riešenie úlohy.

2 Spustenie skriptu a kontrola prepínačov

Chovanie skriptu ovplyvňujú argumenty, s ktorými je skript spustený. Pomocou prepínača –help je možné na štandardný výstup vypísať hlásanie, takzvaný Usege, ktorý popisuje spustenie skriptu a funkciu jednotlivých prepínačov. Skript podporuje všetky prepínače, z rozšírenia **STATP**.

O spracovanie jednotlivých prepínačov sa stará funkcia **checkopt()**. Pokiaľ je pri volaní skriptu zadaný prepínač –help, nie je kontrolované nič a skript vracia navrtovú hodnotu 0 spolu s hlásením uvedením vyššie. V opačnom prípade sa kontroluje poradie poradie argumentov. V prípade zlého poradia je skript ukončený s chybou 10. V prípade použitia viacerých prepínačov –stats=*file*, je kontrolované či sú súbory pre zápis jedinečné, v opačnom prípade je skript ukončený s chybou 12. Funkcia **checkopt()** je implementovaná prostredníctvom for cyklu nad poľom prepínačov.

3 Spracovanie vstupu

Skript spracováva vstup zo štandardného vstupu. Očakáva sa, že na vstupe je kód v jazyku IPPcode21. Vstupný kód hneď na prvom neokomentovanom riadku obsahuje hlavičku .IPPcode21. Túto skutočnosť kontroluje while spolu s funkciou **throwcom()**, ktorá z každého riadku odstráni komentár a vracia to čo z riadku ostalo. Ďalej funkcia **iswhite()**, kontroluje pomocou regulárneho výrazu či riadok obsahuje iba biele znaky. Po narazení na požadovanú hlavičku, sa while ukončí a vygeneruje začiatok výstupného kódu v UML. V prípade, že while prečítal celý súbor a hlavičku nenašiel, skript vracia chybu 21.

Po úspešnej kontrole hlavičky, je čítaný každý riadok samostatne, je odstránený komentár, a reťazec je rozdelený pomocou medzier na slová. Následuje switch, ktorý na základe prvého slova určí o akú inštrukciu ide a následne kontroluje argumenty, ktoré daná inštrukcia obsahuje. Na kontrolu bolo implementovaných niekoľko pomocných funkcii. **isvar()** kontroluje pomocou regulárneho výrazu (ďalej iba regex), či ide o správny zápis premennej. **issym()** kontroluje pomocou regexu, či ide o správne zapísaný symbol. Na základe regexu ďalej fungujú aj funkcie **islabel()** a **istype()**. Vždy po kontrole jednej inštrukcie je vygenerovaný výstup pre danú inštrukciu v jazyku UML. Chybný zápis inštrukcie vedie na chybu 23. Po prečítaní celého vstupu, je vygenerovaný koniec kódu v UML. Pred konečným ukončením skriptu sú ešte zapísané štatistiky z rozšírenia **STATP**.

Bolo potrebné implementovať funkciu **problemchar()**, ktorá vo vstupnom kóde nahádza znaky <, >, &, pretože by mohli byť reprezentované ako znaky jazyka UML, čím by došlo k porušeniu pôvodného kódu. Táto funkcia sa využíva iba pri zapisovaní symbolov.

4 Rozšírenie STATP

Počítanie jednotlivých štatistik nebolo ťažké implementovať, napríklad o počítanie komentárov sa stará funkcia **throwcom()**, ktorá ma statické počítadlo. Ostatné štatistiky sa počítajú priamo pri kontrole jednotlivých inštrukcii. Najťažšie bolo implementovať počítanie dopredných a spätných skokov. Za týmto účelom bol vytvorený zoznam definovaných a nedefinovaných návestí. Pokiaľ sa skáče na už definované návestie, potom sa musí skákať dozadu. Ak sa skáče na nedefinované návestie, tak je pridané do zoznamu a je jasné, že sa skáče niekde dopredu alebo sa jedná o zlý skok. Pri definícii návestia sa kontroluje či návestie je v zozname nedefinovaných návestí. To koľko krát tam je zapísané, značí počet dopredných skokov na toto návestie. Pokiaľ na konci skriptu nie je zoznam s nedefinovanými návestí prázdny, je jasné, že sa jedná o neplatné skoky.