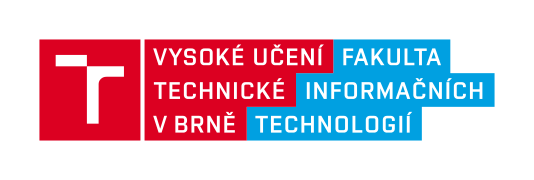
Vysoké učení technické v Brně

# Fakulta informačních technologií



## Dokumentácia k projektu do IFJ a IAL

### Implementácia prekladaču imperatívneho jazyka IFJ18 Tým 076, varianta II

Členovia tímu:  
 Matej Janček (vedúci) xjance00 25%  
 Peter Horňák xhorna14 25%  
 Róbert Hubinák xhubin03 25%  
 Nikolas Patrik xpatri00 25%

Obsah

1. Úvod  
    Cieľom projektu bolo vytvoriť prekladač imperativného jazyka IFJ18, ktorý je založený na základoch jazyka Ruby 2.0, čo je dynamicky typovaný jazyk.   
    Táto dokumentácia popisuje implementáciu prekladača pre jazyk IFJ18. Rozoberá postup pri práci na projekte, spoluprácu v tíme a tak isto voľbu nášho riešenia.  
    Projekt sme si rozdelili na viaceré časti, pričom každý v tíme si zobral na starosť aspoň jednu časť. Podľa týchto častí je rozdelený aj tento dokument.
2. Spolupráca v tíme  
   1. Stretnutia a komunikácia  
        
       Zo začiatku práce na projektu sme sa častejšie stretávali, aby sme sa dohodli na určitých pravidlách pri tvorbe projektu a tak isto rozobratia projektu na menšie časti, ktoré sme dokázali jednoduchšie splniť. Využili sme možnosť pozrieť minuloročné záznamy z prednášok a naštudovať si potrebnú látku dopredu. Keď sme lepšie pochopili celému problému implementácie, stretnutia celého tímu sme obmedzili. Celková komunikácia v tíme nebol problém, keďže takmer celý tím bývame na jednej izbe. Avšak aj napriek tomu sme si zvolili komunikačnú službu *Slack*, cez ktorú sme riešili projekt na diaľku.
   2. Verzovanie kódu  
         
       Z dôvodu potreby pracovať na aktuálnom zdrojovom kóde sme sa zhodli na využitý verzovacieho systému *git* s repozitárom na *Github*-e. Postupom času sa vytvárali vetvy pre rôzne časti projektu, ktoré sa nakoniec zjednotili do jednej vetvi, master. Pokúsili sme sa využiť aj *Issue tracking*, avšak po niekoľkých pokusoch sa od toho upustilo a problémy sa riešili zvyčajne osobne.
3. Lexikálna analýza

Lexikálnu analýzu vykonáva tzv. skener, ktorý funguje na princípoch deterministického konečného automatu, ktorého vstupom je zdrojový kód IFJ18.

Vstupný kód spracováva po jednom znaku, kde biele znaky, ako medzera alebo tabulátor preskočí. Skener sa riadi podľa implementovaného switchu s nekonečným cyklom. Ak znak nevyhovuje žiadnemu prechodu v konečnom automate, skener sa ukončí s lexikálnou chybou alebo ak sa nachádza v konečnom stave, vráti prečítaný nevyhovujúci znak do zdrojového kódu a predá syntaktickému analyzátoru svoj posledný stav.

Jednotlivé znaky spája do lexémov, ktoré následne predáva vo forme tokenov syntaktickému analyzátoru.

Token sme implementovali ako dvojicu (buffer, type). *Buffer* je pointer na miesto v pamäti, kde je uložená postupnosť znakov ukončená končiacou nulou. Táto postupnosť reprezentuje znakovú hodnotu typu. *Type* nám hovorí, akého typu je aktuálny lexém. Všetky možné typy sú uchované v štruktúre, pre jednoduchšiu implementáciu.

Pri spracovaní komentárov sme sa rozhodovali, či je potrebné odovzdávať syntaktickému analyzátoru typ zodpovedajúci komentáru. Rozhodli sme sa pre riešenie s ignoranciou celého textu v komentári a odovzdávaním typu komentára. Toto rozhodnutie sme spravili kvôli tomu, že programátor si píše komentáre len pre vizuálnu kontrolu a nie pre spracovanie prekladačom.