VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Dokumentace projektu z předmětů IFJ a IAL

# Implementace překladače imperativního jazyka IFJ22

Tým xbarto0g, varianta – TRP

| | |
|---|---|
| Petr Bartoš – xbarto0g (vedoucí) | 40 % |
| Tomáš Rajsigl – xrajsi01 | 30 % |
| Lukáš Zedek – xzedek03 | 30 % |
| Dmytro Afanasiev – xafana01 | 0 % |

2. prosince 2022

# 1 Úvod

Cílem projektu je vytvořit program v jazyce C, který načte zdrojový kód zapsaný ve zdrojovém jazyce IFJ22 a přeloží jej do cílového jazyka IFJcode22. Jazyk IFJ22 je zjednodušenou podmnožinou jazyka PHP. Konkrétně se jedná o variantu zadání s implementací tabulky symbolů pomocí tabulky s rozptýlenými položkami.

# 2 Práce v týmu

Práci jsme si rozdělili jakožto čtyřčlenný tým, avšak z důvodu nespolupracování a nezájmu jednoho člena jsme byli nuceni si práci přerozdělit a projekt vypracovat pouze ve třech. Vzhledem k časové náročnosti a složitosti daných částí jsme se rozhodli pro nerovnoměrné rozdělení bodů.

## 2.1 Rozdělení práce

- Petr Bartoš
  - Syntaktická a sémantická analýza, kostra pro testovaní, tabulka symbolů, Makefile
- Tomáš Rajsigl
  - Lexikální analýza, dokumentace
- Lukáš Zedek
  - Generování cílového kódu, testování, dokumentace

Kontrole kódu, jeho čitelnosti a opravě chyb jsme se věnovali všichni.

## 2.2 Vývojový cyklus

Při vypracovávání projektu jsme využívali verzovací systém Git. Pro dané části projektu byly vytvořeny konkrétní branche, kde jsme je testovali a upravovali. Před zahrnutím do master branche byl vyžadován pull request, následný code review a schválení od jednoho či více členů. Součástí vývojového cyklu byl i unit testing. Při každém commitu byly automaticky spuštěny testy a bylo tak hned možné vidět, jaký dopad bude commit mít na výsledný program.

# 3 Návrh a implementace překladače

Projekt byl rozdělen na několik konrétních částí, které jsou představeny v této kapitole.

## 3.1 Lexikální analýza

### 3.1.1 Dynamický řetězec

## 3.2 Syntaktická analýza

### 3.2.1 Precedenční syntaktická analýza

## 3.3 Sémantická analýza

## 3.4 Generování cílového kódu

## 3.5 Přílohy

### 3.5.1 Diagram konečného automatu

### 3.5.2 LL – gramatika

1. <program> -> <body_main> <program_e>
2. <program_e> -> $\varepsilon$
3. <program_e> -> ?>
4. <type_p> -> float
5. <type_p> -> int
6. <type_p> -> string
7. <type> -> null
8. <type> -> <type_p>
9. <type> -> ?<type_p>
10. <return> -> return <return_p>
11. <return_p> -> expr
12. <return_p> -> $\varepsilon$
13. <body_main> -> <body>
14. <body_main> -> <func_def> <body>
15. <body> -> $\varepsilon$
16. <body> -> <if> ; <body>
17. <body> -> <while> ; <body>
18. <body> -> expr ; <body>
19. <body> -> identifier_var = expr ; <body>
20. <body> -> indentifier_var = <func> ; <body>
21. <body> -> <func> ; <body>
22. <body> -> <return> ; <body>
23. <if> -> if expr { <body> } else { <body> }
24. <while> -> while expr { <body> }
25. <params> -> $\varepsilon$
26. <params> -> <type> identifier_var <params_n>
27. <params> -> identifier_var <params_n>
28. <params> -> expr <params_n>
29. <params_n> -> $\varepsilon$
30. <params_n> -> , <params_p> <params_n>
31. <params_p> -> <type> identifier_var
32. <params_p> -> identifier_var
33. <params_p> -> expr
34. <func_def> -> function identifier_func ( <params> ) : <type> { <body> }
35. <func> -> identifier_func ( <params> )

Tabulka 1: LL – gramatika řídící syntaktickou analýzu

### 3.5.3 LL – tabulka

### 3.5.4 Precedenční tabulka

# 4 Reference