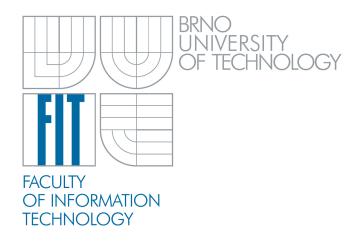
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



Dokumentace projektu do předmětu IFJ a IAL **Interpret jazyka IFJ15**

Tým 043, varianta b/2/I

Obsah

1	Uvod	2											
2	Struktura projektu	2											
3	Lexikální analizátor (scanner) 3.1 Konečný automat	2 2											
4	Syntaktická analizátor (parser) 4.1 LL gramatika	3 3											
5	ntický analizátor												
6	Interpret												
7	Algoritmy z předmětu IAL 7.1 Heapsort												
8	Práce v týmu 8.1 Rozdělení práce												
9	Závěr 9.1 Metriky kódu	5											
10	Literatura	5											

1 Úvod

Dokumentace popisuje návrh a implementaci projektu Interpret jazyka IFJ15 do předmětu IFJ (Formální jazyky a překladače) a IAL (Algoritmy). Vybrali jsme si zadání b/2/I, které nám udávalo, jáký algoritmus máme pro daný problém využít. Pro vyhledávání **Boyer-Mooreův algoritmus**, pro řazení algoritmus Heap sort, který byl využit ve vestavěné funkci **sort**. Poslední ze specifikových pravidel použití, jsme měli využít k implementaci tabulky symbolů **binární vyhledávací strom**.

Cílem projektu bylo vytvořit program, který interpretuje jazyk IFJ15, což je velmi zjednodušenou podmnožinou jazyka C++11.

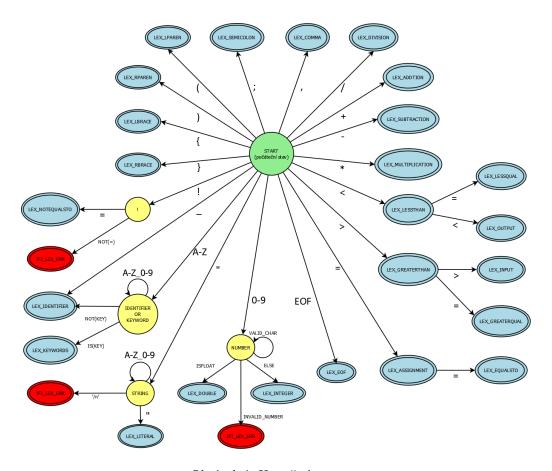
2 Struktura projektu

Překladač je rozdělen do tří hlavních celků. Pomocí **lexikálního analyzátoru** načítá zdrojový kód. **Syntaktický analyzátor** zažádá lexikální o token a ověří syntaxi a sémantiku. Po bezchybné kontrole se spustí **interpret**.

3 Lexikální analizátor (scanner)

Lexikální analyzátor načítá zdrojový kód po znacích a převádí jej na tokeny (funkce *lex_get_token*), přičemž veškeré komentáře ignoruje. Implementováno pomocí konečného automatu (Obrázek 1). Funkce je volána vždy, když syntaktický analizátor zažádá o token.

3.1 Konečný automat



Obrázek 1: Konečný automat

4 Syntaktická analizátor (parser)

Syntaktický analyzátor postupně volá tokeny z lexikálního analyzátoru. Tokeny jsou zpracovávaný dvěma způsoby. Pomocí LL gramatiky (Tabulka 1) a precedenční syntaktické analýzy (Tabulka 2). Při úspěšném dokončení syntaktické analýzy se generuje instrukční páska pro interpret.

4.1 LL gramatika

LL gramatika postupně ověřuje příchozí tokeny.

```
1.
      < program >
                       < declrList > \mathsf{EOF}
2.
      < declrList> \\
                       < func Declr > < declr List >
3.
      < declrList> \\
                       < empty >
4.
     < func Declr >
                       < typeSpec > {\rm ID} \ (< params >)
5.
      < typeSpec >
6.
      < typeSpec >
                       DOUBLE
7.
      < typeSpec >
                       STRING
8.
      < params >
                       < paramItem >
9.
       < params >
                       < paramItem >, < params >
10. < paramItem >
                       < typeSpec > ID
11. < paramItem >
                       < empty >
```

Tabulka 1: LL gramatika

4.2 Precedenční syntaktická analýza

	+	_	*	/	<	>	<=	>=	==	!=	()	id	\$
+	>	>	<	<	>	>	>	>	>	>	<	>	<	>
_	>	>	<	<	>	>	>	>	>	>	<	>	<	>
*	>	>	>	>	>	>	>	>	>	>	<	>	<	>
/	>	>	>	>	>	>	>	>	>	>	<	>	<	>
<	<	<	<	<	<	>	>	>	>	>	<	>	<	>
>	<	<	<	<	<	>	>	>	>	>	<	>	<	>
$\leq=$	<	<	<	<	<	>	>	>	>	>	<	>	<	>
>=	<	<	<	<	<	>	>	>	>	>	<	>	<	>
==	<	<	<	<	<	>	>	>	>	>	<	>	<	>
!=	<	<	<	<	>	>	>	>	>	>	<	>	<	>
(<	<	<	<	<	<	<	<	<	<	<	=	<	
)	>	>	>	>	>	>	>	>	>	>		>		>
id	>	>	>	>	>	>	>	>	>	>		>		>
\$	<	<	<	<	<	<	<	<	<	<	<		<	

Tabulka 2: Precenenční tabulka syntaktické analýzy výrazů

5 Sémantický analizátor

6 Interpret

7 Algoritmy z předmětu IAL

- 7.1 Heapsort
- 7.2 Boyer-Mooreův algoritmus
- 7.3 Binární vyhledávací strom (BVS)

8 Práce v týmu

8.1 Rozdělení práce

- Martin Honza:
- Patrik Jurnečka: Syntaktický analyzátor, dokumentace
- Hana Slámová: Vestavěné funkce
- Frantisek Šumšal: Lexikální analýzátor, syntaktický analyzátor
- Adam Švidroň: Heap sort
- 9 Závěr
- 9.1 Metriky kódu
- 10 Literatura