Vysoké učení technické v Brně
Fakulta Informačních technologií

Dokumentace projektu k předmětu IFJ a IAL

Implementace překladače imperativního jazyka IFJ17

Tým 54 varianta II.
datum: 30.11.2017

Seznam autorů:
Petr Marek - 25 % - Vedoucí
Petr Knetl - 25 %
Jakub Štefanišin - 25 %
Matěj Knapovský, xknapo04 - 25%

Rozšíření -

1. Úvod

Tato Dokumentace popisuje návrh a implementaci překladače imperativního jazyka IFJ17. Jazyk IFJ17 jed podmnožinou jazyka FreeBASIC. Naším úkolem bylo napsat překladač a generátor kódu. Dle zadání jsme musely vytvořit tabulku symoblů pomocí hashovací tabulky. Projekt jsme rozdělily do 3 částí. Každá část je popsána v samostatné kapitole.

- Lexikální analýza
- syntaktická analýza
- sémantická analýza a generátor kódu

2. Implementace částí

2.1 Lexikální analyzátor

Lexikální analyzátor (dále jen LA) je část programu, který přímo pracuje se vstupním kódem. Práce LA čtení vstupního kódu, znak po znaku, vynechávání nepotřebených informací (bílé znaky, komentáře) a vyhodnocování lexikálních pravidel, podle kterých poté vygeneruje Token. Token je definován strukturou ve které se nachází typ, stav a řádek na kterém se nachází. Princip chování lexikálního analyzátoru je založen na konečném automatu (obrázek n°1). Načítané znaky z vstupního kódu jsou ukládány do tzv “nekonečného stringu”. LA je řízen pomocí syntaktické analýzy.

2.2 Syntaktická analýza

2.3 Sémantická analýza

2.4 Generátor kódu

3. Algoritmy a funkce

V této kapitole jsou popsány funkce a algoritmy

3.1 Nekonečné stringy

Nekonečné stringy byly použity v lexikální analýze při vytváření tokenu.
Tato funkce vytvoří pole znaků a přiřadí mu pamět, při přiřazení znaku do pole kontroluje jestli pole není plné, pokud ano rozšíří pole a pokračuje dále.

3.2 tabulka symbolů

4. vývoj

V této kapitole je posána komunikace členů týmu, rozdělení práce, použité pomocné programy ve vývoji a metriky.

4.1 rozdělení práce

- Matěj Knapovský - lexikální analyzátor, dokumentace, testy
- Petr Knetl - syntaktická analýza, sémantická analýza
- Petr Marek - lexikální analyzátor, LL tabulka, generátor kódu
- Jakub Štefanišin - tabulka symbolů, syntaktická analýza

4.2 komunikace

Pro komunikaci jsme Nejčastěji používaly **Facebook messenger**, kde jsme měli od začátku projektu zavedenou skupinovou místnost. Dále jsme také plánovaly osobní **schůzky** každý týden, v úterý, v knihovně FITu. Také jsme používaly freeware program **Teamspeak**, který umožňuje hlasovou komunikaci přes internet.

4.3 použité prostředky

Pro společný vývoj jsme používaly verzovací systém **Git**, a jako hosting jsme použily soukromý repozitář na **Githubu**. Na repozitáři sme vytvořili větve pro každou samostatnou část programu, které jsme na konci vývoje spojily do jedné větve. pro překlad jsme použily program **Make**. Pro lepší orientaci ve verzovacím systému Git sme použili program **GitKraken**.

4.4 metriky

souborů: TBA
řádků kódu:TBA

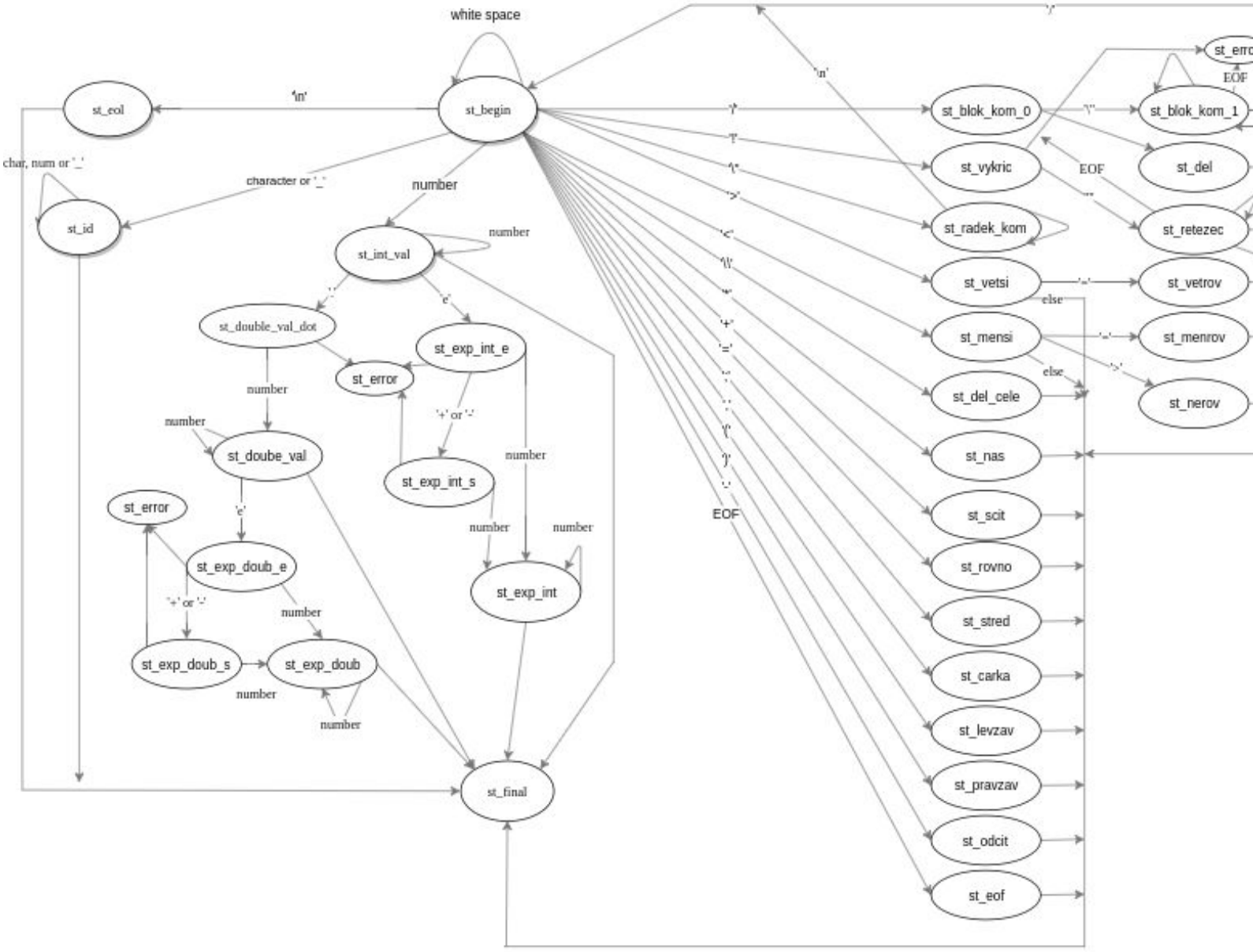
5. Závěr

5.1 zdroje

- Přednášky, skripta a podklady k předmětům IFJ a IAL

6. Přílohy

6.1 Konečný automat lexikální analýzy



6.2 LL pravidla

1. <start state> -> <function> <scope>
2. <scope> -> ‘Scope’ <st-list> ‘End’ ‘Scope’
3. <function> -> <function-head> <st-list> <function-tail> <function>
4. <function> -> <function-dec> <function>
5. <function> -> ε
6. <function-dec> -> ‘Declare’ ‘Function’ <function-id> ‘(‘ <par> ‘)’ ‘As’ <type>
7. <function-head> -> ‘Function’ <function-id> ‘(‘ <par> ‘)’ ‘As’ <type> ‘EOL’
8. <function-tail> -> ‘End’ ‘Function’
9. <par> -> <id> ‘As’ <type> <next-par>
10. <par> -> ε
11. <next-par> -> ‘,’ <par>
12. <next-par> -> ε
13. <type> -> ‘Integer’
14. <type> -> ‘Double’
15. <type> -> ‘String’
16. <st-list> -> <stat> <st-list>
17. <st-list> -> ε
18. <stat> -> ‘Dim’ <id> ‘As’ <Type> <eval>
19. <eval> -> ‘=’ <expr>
20. <eval> -> ε
21. <stat> -> <id> ‘=’ <assign>
22. <assign> -> <expr>
23. <assign> -> <function-id> ‘(‘ <call-par> ‘)’
24. <call-par> -> <id> <call-next-par>
25. <call-par> -> ε
26. <call-next-par> -> ‘,’ <call-par>
27. <call-next-par> -> ε
28. <stat> -> ‘Input’ <id>
29. <stat> -> ‘Print’ <expr> ; <pr-expr>
30. <pr-expr> -> <expr> ; <pr-expr>
31. <pr-expr> -> ε
32. <stat> -> ‘If’ <expr> ‘Then’ ‘EOL’ <st-list> ‘Else’ ‘EOL’ <st-list> ‘End’ ‘If’
33. <stat> -> ‘Do’ ‘While’ <expr> ‘EOL’ <st-list> ‘Loop’
34. <stat> -> ‘Return’ <expr>

6.3 LL tabulka

	Scope	End	Declare	Function	(	)	ID	FUN_ID	As	EOL	,	Integer	Double
<start state>	1		1	1									
<scope>	2												
<function>			4	3									
<function-dec>			6										
<function-head>				7									
<function-tail>		8											
<par>						10	9						
<next-par>						12					11		
<type>												13	14
<st-list>		17					16						
<stat>							21						
<eval>		20					20						
<assign>								23					
<call-par>						25	24						
<call-next-par>						27					26		
<pr-expr>		27					27						

	String	Dim	=	Input	Print	;	If	Then	Else	Do	While	Loop	Return	$	<expresion>
<start state>															
<scope>															
<function>															
<function-dec>															
<function-head>															
<function-tail>															
<par>															
<next-par>															
<type>	15														
<st-list>		16		16	16		16			16			16		
<stat>		18		28	29		32			33			34		
<eval>		20	19	20	20		20			20			20		
<assign>															22
<call-par>															
<call-next-par>															
<pr-expr>		27		27	27		27			27			27		30

6.4 Precedenční tabulka

	*,/,\	+,-	=,<>,<	<=,>,>=	(	)	i	nm,sr,bl	$	R
*,/,\	>	>	>	>	<	>	<	<	>	<
+,-	<	>	>	>	<	>	<	<	>	<
=,<>,<	<	<	>	>	<	>	<	<	>	<
<=,>,>=	<	<	>	>	<	>	<	<	>	<
(	<	<	<	<	<	=	<	<		<
)	>	>	>	>		>				
i	>	>	>	>		>		<	>	
nm,sr,bl	>	>	>	>		>			>	
$	<	<	<	<	<		<	<		
R	>	>	>	>		>				