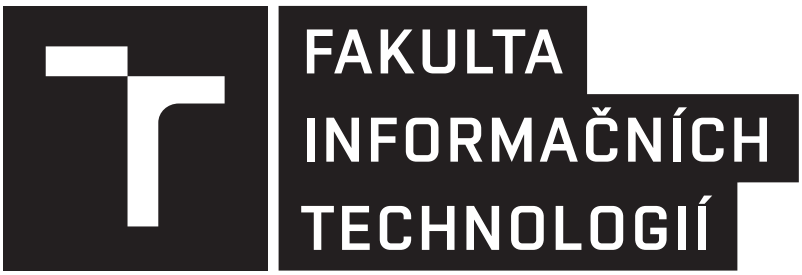
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ



Dokumentace ke společnému projektu do IFJ a IAL

Implementace interpretu jazyka IF21

Tým 143, varianta I

Vedoucí týmu:
Susík Josef xsusik00

Další členové:
Putala Marek xputal00
Popelář Samuel xpopel22
Kniazkin Daniil xkniaz00

Contents

1 Úvod

Dokumentace popisuje implementace programu v jazyce C,která načte zdrojový kód zapsaný ve zdrojovém jazyce IFJ21 a přeloží jej do cílového jazyka IFJcode21 (mezikód).Projekt se skládá ze 4 částí:

- Lexikální analy
- Syntaktický analy
- Sémantický anal
- Interpet

2 Práce v týmu

2.1 Motivace

Každý z nás měl vysokou motivaci.

2.2 Rozdělení práce

Vzhledem k tomu, že projekt vyžaduje spoustu času práce byla rozdělena rovnoměrně mezi všechny členy týmu .

- -bla-bla
- -bla-bla
- -bla-bla
- blala

3 Struktura projektu

3.1 Lexikální analyzátor

První věc, kterou bylo třeba implementovat, je Lexikální analyzátor.Hlavním cílem lexikálního analyzátoru je čtení zdrojového souboru a podle lexikálních pravidel jazyka rozdělit posloupnosti znaků souboru na lexikální části - lexémy.

Funkce **getNextToken** čte jednotlivý znaky a převádí je na strukturu **Token**.Struktura Token obsahuje informace o typu tokenu a atribut který navazuje na ten typ.

Lexikální analyzátor implementován podle dříve vytvořeného konečného automatu (příloha 4.1).

3.2 Syntaktický analyzátor

Syntaktický analyzátor implementována tak že pro každé pravidlo v LL gramatice je vlastní funkce a každá z nich pracuje se strukturou **Parser**.V této struktuře jsou všechny potřebné proměnné pro syntaktickou a sémantickou analýzu.

3.2.1 Stack

3.3 Sémantický analyzátor

3.3.1 - - - - - - -

3.3.2 - - - - - - -

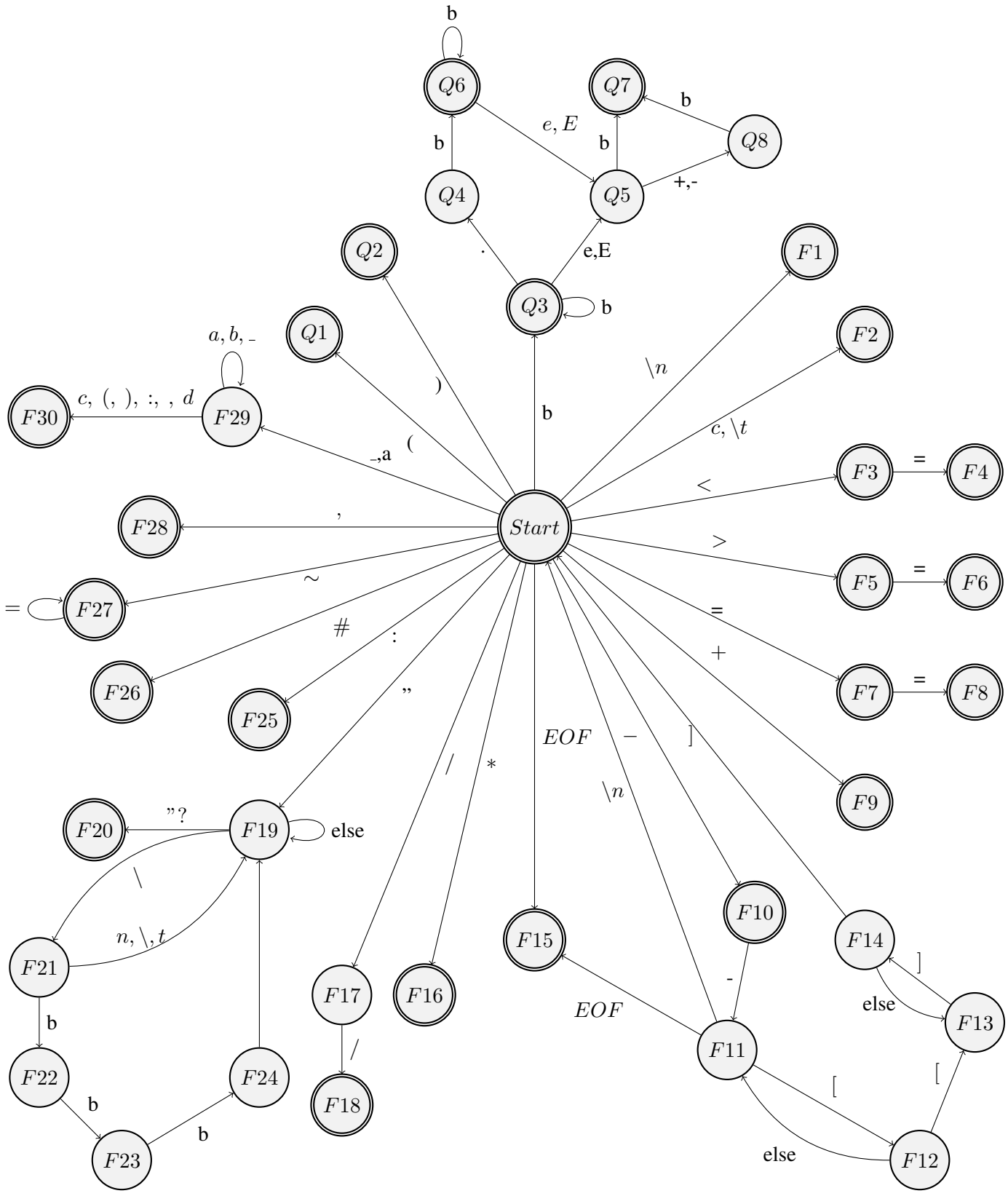
3.4 Interpret

3.4.1 - - - - - - -

3.4.2 - - - - - - -

4 Přílohi

4.1 Koneční automat



F1 TOKEN_TYPE_EOL
F2 TOKEN_TYPE_EMPTY
F3 TOKEN_TYPE_LESS
F4 TOKEN_TYPE_LESS_EQ
F5 TOKEN_TYPE_MORE
F6 TOKEN_TYPE_MORE_EQ
F7 TOKEN_TYPE_ASSIGN
F8 TOKEN_TYPE_EQUALS
F9 TOKEN_TYPE_PLUS
F10 TOKEN_TYPE_MINUS
F11 TOKEN_TYPE_LINE_COMMENTARY
F12 TOKEN_TYPE_BLOCK_COMMENTARY_START
F13 TOKEN_TYPE_BLOCK_COMMENTARY_END
F14 TOKEN_TYPE_BLOCK_COMMENTARY_END_2
F15 TOKEN_TYPE_EOF
F16 TOKEN_TYPE_MULTIPLY
F17 TOKEN_TYPE_DIVISION
F18 TOKEN_TYPE_DIVISION_INT
F19 TOKEN_TYPE_STRING_START
F20 TOKEN_TYPE_STRING
F21 TOKEN_TYPE_STRING_ESCAPE
F22 TOKEN_TYPE_STRING_ESCAPE_TWO
F23 TOKEN_TYPE_STRING_ESCAPE_THREE
F24 TOKEN_TYPE_STRING_ESCAPE_WRITEF
F25 TOKEN_TYPE_COLONS
F26 TOKEN_TYPE_LENGTH
F27 TOKEN_TYPE_EG_ASSIGN
F28 TOKEN_TYPE_COMMA
F29 TOKEN_TYPE_IDENTIFIER
F30 TOKEN_TYPE_KEYWORD
Q1 TOKEN_TYPE_LEFT_PAR
Q2 TOKEN_TYPE_RIGHT_PAR
Q3 TOKEN_TYPE_DIGIT
Q4 TOKEN_TYPE_DOUBLE_DOT
Q5 TOKEN_TYPE_EXPONENT
Q6 TOKEN_TYPE_DOUBLE
Q7 TOKEN_TYPE_EXPONENT_EXPONENT
Q8 TOKEN_TYPE_EXPONENT_SIGN

a isaplha
b isdigit
c isspace
d EOF

4.2 LL-gramatika

LL-GRAMATIKA

1. $< require > \ -> \ REQUIREIDEOL < prog >$
2. $< prog > \ -> \ FUNCTIONID(< params >) < return\_type > EOL < body > EOLEND < prog >$
3. $< prog > \ -> \ GLOBALID : FUNCTION(< type >< type\_2 >) < return\_type > EOL < prog >$
4. $< prog > \ -> \ ID(< args >)EOL < prog >$
5. $< prog > \ -> \ EOL < prog >$
6. $< prog > \ -> \ EOF$
7. $< params > \ -> \ \varepsilon$
8. $< params > \ -> \ ID :< type >< params\_2 >$
9. $< params\_2 > \ -> \ , ID :< type >< params\_2 >$
10. $< params\_2 > \ -> \ \varepsilon$
11. $< return\_type > \ -> \ \varepsilon$
12. $< return\_type > \ -> \ :< type >< return\_type\_2 >$
13. $< return\_type\_2 > \ -> \ , < type >< return\_type\_2 >$
14. $< return\_type\_2 > \ -> \ \varepsilon$
15. $< type > \ -> \ INTEGER$
16. $< type > \ -> \ NUMBER$
17. $< type > \ -> \ STRING$
18. $< type > \ -> \ NIL$
19. $< type\_2 > \ -> \ , < type >< type\_2 >$
20. $< type\_2 > \ -> \ \varepsilon$
21. $< body > \ -> \ \varepsilon$
22. $< body > \ -> \ EOL < body >$
23. $< body > \ -> \ IF < expression > THENEOL < body >< return > EOLELSEEOL < body >< return > EOLENDEOL < body >$
24. $< body > \ -> \ WHILE < expression > DOEOL < body >< return > EOLENDEOL < body >$
25. $< body > \ -> \ < return >$
26. $< body > \ -> \ ID(< args >)EOL < body >$
27. $< body > \ -> \ < id >=< exp >< body >$
28. $< body > \ -> \ < scope > ID :< assign >< body >$
29. $< return > \ -> \ \varepsilon$
30. $< return > \ -> \ RETURN < return\_value > EOL < body >$
31. $< return\_value > \ -> \ \varepsilon$
32. $< return\_value > \ -> \ < exp >$
33. $< exp > \ -> \ < expression >< expression\_2 >< exp\_2 >$
34. $< exp > \ -> \ ID(< args >) < exp\_2 >$
35. $< exp\_2 > \ -> \ \varepsilon$
36. $< exp\_2 > \ -> \ , < exp >< exp\_2 >$
37. $< expression\_2 > \ -> \ \varepsilon$
38. $< expression\_2 > \ -> \ , < expression >< expression\_2 >$
39. $< args > \ -> \ \varepsilon$
40. $< args > \ -> \ < value >< args\_2 >$
41. $< args\_2 > \ -> \ \varepsilon$
42. $< args\_2 > \ -> \ , < value >< args\_2 >$
43. $< value > \ -> \ ID$
44. $< value > \ -> \ INTEGER\_VALUE$
45. $< value > \ -> \ NUMBER\_VALUE$
46. $< value > \ -> \ STRING\_VALUE$
47. $< value > \ -> \ NIL$
48. $< id > \ -> \ ID < id\_2 >$

49. $< id\_2 > \ -> \ \varepsilon$
50. $< id\_2 > \ -> \ ,< id >< id\_2 >$
51. $< assign > \ -> \ < type >$
52. $< assign > \ -> \ < type >=< exp >$
53. $< assign > \ -> \ < type >=< value >$
54. $< scope > \ -> \ SCOPE$

4.3 LL-tabulka

	REQUIRE	ID	FUNCTION	EXPRESSION	GLOBAL	(	)	:	,	INTEGER	NUMBER	STRING	NIL	IF	WHILE	=	RETURN	INTEGER_VALUE	NUMBER_VALUE	STRING_VALUE	NIL	SCOPE	EOL	EOF	END	$
require	1																									
prog		4	2		3																		5	6		
params		8					7																			
params_2		10					10		9																	
return_type								12															11			
return_type_2									13														14			
type										15	16	17	18													
type_2							20		19																	
body		26		27										23	24		25					28	22		21	
return																	30									29
return_value		32		32																						31
exp		34		33																						
exp_2									36																	35
expression_2				37					38																	
args							39											40	40	40	40					
args_2							41		42																	
value																		43	44	45	46					
id		47																								
id_2									50																	49
assign										51	51	51	51			52		53	53	53			52			
scope																						54				

4.4 Precedenční tabulka