



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Formální jazyky a překladače

Dokumentace projektu do IFJ a IAL

Tým 066, varianta II

Roman Ondráček	xondra58	%	vedoucí
Pavel Raur	xraurp00	%	
František Jeřábek	xjerab25	%	
Radim Lipka	xlipka02	%	

Seznam implementovaných rozšíření

BOOLOP, BASE, CYCLES, FUNEXP, IFTHEN, TABUNARY

Obsah

1	Úvod	2
2	Rozbor částí překladače	2
2.1	Lexikální analýza	2
2.2	Syntaktická analýza	3
2.2.1	LL–Gramatika a LL–Tabulka	3
2.2.2	Precedenční syntaktická analýza	4
2.3	Sémantická analýza	5
2.4	Generátor cílového kódu	5
2.5	Optimalizace	5
3	Implementovaná rozšíření	5
3.1	BOOLOP	5
3.2	BASE	5
3.3	CYCLES	5
3.4	FUNEXP	5
3.5	IFTHEN	5
3.6	TABUNARY	5
4	Závěr	5

2.2 Syntaktická analýza

Syntaktickou analýzu jsme naimplementovali v modulu **parser.c** metodou shora dolů, konkrétněji metodou *rekurzivního sestupu*, která je založena na LL–gramatice a LL–tabulce.

2.2.1 LL–Gramatika a LL–Tabulka

1. $\langle \text{code} \rangle \rightarrow \langle \text{body} \rangle \langle \text{eols} \rangle \langle \text{EOF} \rangle$
2. $\langle \text{body} \rangle \rightarrow \langle \text{definitions} \rangle \langle \text{statements} \rangle$
3. $\langle \text{definitions} \rangle \rightarrow \langle \text{eols} \rangle \langle \text{definition} \rangle \langle \text{definitions} \rangle$
4. $\langle \text{definition} \rangle \rightarrow \epsilon$
5. $\langle \text{definition} \rangle \rightarrow \text{DEF IDENTIFIER} (\langle \text{function_params} \rangle) : \langle \text{eols} \rangle \text{INDENT} \langle \text{statements} \rangle \text{DEDENT}$
6. $\langle \text{definition} \rangle \rightarrow \langle \text{function_call} \rangle$
7. $\langle \text{function_call} \rangle \rightarrow \text{IDENTIFIER} (\langle \text{function_params} \rangle)$
8. $\langle \text{function_params} \rangle \rightarrow \epsilon$
9. $\langle \text{function_params} \rangle \rightarrow \langle \text{function_param} \rangle \langle \text{function_nparam} \rangle$
10. $\langle \text{function_nparam} \rangle \rightarrow , \langle \text{function_param} \rangle \langle \text{function_nparam} \rangle$
11. $\langle \text{function_nparam} \rangle \rightarrow \epsilon$
12. $\langle \text{function_param} \rangle \rightarrow \langle \text{expression} \rangle$
13. $\langle \text{statements} \rangle \rightarrow \epsilon$
14. $\langle \text{statements} \rangle \rightarrow \langle \text{statement} \rangle \text{EOL} \langle \text{eols} \rangle \langle \text{statements} \rangle$
15. $\langle \text{statement} \rangle \rightarrow \langle \text{returnRule} \rangle$
16. $\langle \text{statement} \rangle \rightarrow \langle \text{condition} \rangle$
17. $\langle \text{statement} \rangle \rightarrow \langle \text{assignment} \rangle$
18. $\langle \text{statement} \rangle \rightarrow \langle \text{whileRule} \rangle$
19. $\langle \text{statement} \rangle \rightarrow \text{PASS}$
20. $\langle \text{returnRule} \rangle \rightarrow \text{RETURN} \langle \text{return_expression} \rangle$
21. $\langle \text{return_expression} \rangle \rightarrow \epsilon$
22. $\langle \text{return_expression} \rangle \rightarrow (\langle \text{return_expression} \rangle)$
23. $\langle \text{return_expression} \rangle \rightarrow \langle \text{expression} \rangle$

24. $\langle \text{condition} \rangle \rightarrow \text{IF } \langle \text{condition_expression} \rangle : \text{EOL } \langle \text{eols} \rangle \text{ INDENT } \langle \text{statement} \rangle \text{ DEDENT } \langle \text{else_condition} \rangle$
25. $\langle \text{else_condition} \rangle \rightarrow \epsilon$
26. $\langle \text{else_condition} \rangle \rightarrow \text{ELSE} : \text{EOL } \langle \text{eols} \rangle \text{ INDENT } \langle \text{statement} \rangle \text{ DEDENT}$
27. $\langle \text{condition_expression} \rangle \rightarrow (\langle \text{condition_expression} \rangle)$
28. $\langle \text{condition_expression} \rangle \rightarrow \langle \text{expression} \rangle$
29. $\langle \text{assignment} \rangle \rightarrow \text{IDENTIFIER} = \langle \text{expression} \rangle$
30. $\langle \text{whileRule} \rangle \rightarrow \text{WHILE } \langle \text{condition_expression} \rangle : \text{EOL } \langle \text{eols} \rangle \text{ INDENT } \langle \text{statement} \rangle \text{ DEDENT}$
31. $\langle \text{eols} \rangle \rightarrow \epsilon$
32. $\langle \text{eols} \rangle \rightarrow \text{EOL } \langle \text{eols} \rangle$

2.2.2 Precedenční syntaktická analýza

Precedenční syntaktická analýza je využita pro zpracování výrazů a je řízena precedenční tabulkou. Jelikož jsme se rozhodli pro implementaci rozšíření TABUNARY, museli jsme se vypořádat s rozdílem mezi unárními operátory plus a mínus a jejich binární podobou, protože do této části syntaktické analýzy přicházejí jako od sebe nerozpoznatelné tokeny. V precedenční analýze hojně pracujeme se zásobníkem, na který se uloží každý nově příchozí token, který následně porovnáme s vrcholem zásobníku a podle predenční tabulky provedeme náležitou operaci. Výraz se takto postupně redukuje podle redukčních pravidel a vzniká tak strom.

$E \rightarrow id$	$E \rightarrow E + E$	$E \rightarrow E < E$
$E \rightarrow (E)$	$E \rightarrow E - E$	$E \rightarrow E > E$
$E \rightarrow id()$	$E \rightarrow E * E$	$E \rightarrow E \leq E$
$E \rightarrow id(E)$	$E \rightarrow E / E$	$E \rightarrow E \geq E$
$E \rightarrow id(E, \dots)$	$E \rightarrow E // E$	$E \rightarrow E \text{ and } E$
$E \rightarrow -E$	$E \rightarrow E != E$	$E \rightarrow E \text{ or } E$
$E \rightarrow +E$	$E \rightarrow E == E$	$E \rightarrow \text{not } E$

Tabulka 1: Redukční pravidla

	<i>un+</i>	<i>un−</i>	<i>*</i>	<i>/</i>	<i>//</i>	<i>+</i>	<i>−</i>	<i><</i>	<i>></i>	<i><=</i>	<i>>=</i>	<i>==</i>	<i>and</i>	<i>or</i>	<i>not</i>	<i>!=</i>	<i>(</i>	<i>)</i>	<i>id</i>	<i>\$</i>
<i>un+</i>	<	<	>	>	>	>	>	>	>	>	>	>	>	>	−	>	<	>	<	>
<i>un−</i>	<	<	>	>	>	>	>	>	>	>	>	>	>	>	−	>	<	>	<	>
<i>*</i>	<	<	>	>	>	>	>	>	>	>	>	>	>	>	<	>	<	>	<	>
<i>/</i>	<	<	>	>	>	>	>	>	>	>	>	>	>	>	<	>	<	>	<	>
<i>//</i>	<	<	>	>	>	>	>	>	>	>	>	>	>	>	<	>	<	>	<	>
<i>+</i>	<	<	<	<	<	>	>	>	>	>	>	>	>	>	<	>	<	>	<	>
<i>−</i>	<	<	<	<	<	>	>	>	>	>	>	>	>	>	<	>	<	>	<	>
<i><</i>	<	<	<	<	<	<	<	−	−	−	−	−	>	>	<	−	<	>	<	>
<i>></i>	<	<	<	<	<	<	<	−	−	−	−	−	>	>	<	−	<	>	<	>
<i><=</i>	<	<	<	<	<	<	<	−	−	−	−	−	>	>	<	−	<	>	<	>
<i>>=</i>	<	<	<	<	<	<	<	−	−	−	−	−	>	>	<	−	<	>	<	>
<i>==</i>	<	<	<	<	<	<	<	−	−	−	−	−	>	>	<	−	<	>	<	>
<i>and</i>	<	<	<	<	<	<	<	<	<	<	<	<	>	>	<	<	<	>	<	>
<i>or</i>	<	<	<	<	<	<	<	<	<	<	<	<	<	>	<	<	<	>	<	>
<i>not</i>	−	−	>	>	>	>	>	>	>	>	>	>	>	>	<	>	<	>	<	>
<i>!=</i>	<	<	<	<	<	<	<	−	−	−	−	−	>	>	<	−	<	>	<	>
<i>(</i>	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	=	<	−
<i>)</i>	>	>	>	>	>	>	>	>	>	>	>	>	>	>	>	>	−	>	−	>
<i>id</i>	−	−	>	>	>	>	>	>	>	>	>	>	>	>	>	>	=	>	−	>
<i>\$</i>	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	−	<	−

Tabulka 2: Tabulka precedenční syntaktické analýzy

2.3 Sémantická analýza

2.4 Generátor cílového kódu

2.5 Optimalizace

3 Implementovaná rozšíření

3.1 BOOLOP

3.2 BASE

3.3 CYCLES

3.4 FUNEXP

3.5 IFTHEN

3.6 TABUNARY

4 Závěr