

Grupo:

Gustavo Luiz Kohler

Cleverson Borges dos passos

Breno Juliano Sayão

Gramática Transformada:

MAIN	-> STMT FLIST ϵ
FLIST	-> FDEF FLIST_LINHA
FLIST_LINHA	-> FDEF FLIST_LINHA ϵ
FDEF	-> def id (PARLIST) { STMTLIST }
PARLIST	-> int id PARLIST_LINHA ϵ
PARLIST_LINHA	-> , PARLIST ϵ
VARLIST	-> id VARLIST_LINHA
VARLIST_LINHA	-> , VARLIST ϵ
STMT	-> int VARLIST ; ATRIBST ; PRINTST ; RETURNST ; IFSTMT { STMTLIST } ;
ATRIBST	-> id = ATRIBST_CONTENT
ATRIBST_CONTENT	-> num TERM_LINHA NUMEXPR_LINHA EXPR_LINHA (NUMEXPR) TERM_LINHA NUMEXPR_LINHA
EXPR_LINHA	id ATRIBST_ID
ATRIBST_ID	-> (PARLISTCALL) TERM_LINHA NUMEXPR_LINHA EXPR_LINHA
FCALL	-> id (PARLISTCALL)
PARLISTCALL	-> id PARLISTCALL_LINHA ϵ
PARLISTCALL_LINHA	-> , PARLISTCALL ϵ
PRINTST	-> print EXPR
RETURNST	-> return RETURNST_LINHA
RETURNST_LINHA	-> id ϵ
IFSTMT	-> if (EXPR) { STMT } IFSTMT_LINHA
IFSTMT_LINHA	-> else { STMT } ϵ
STMTLIST	-> STMT STMTLIST_LINHA
STMTLIST_LINHA	-> STMT STMTLIST_LINHA ϵ
EXPR	-> NUMEXPR EXPR_LINHA
EXPR_LINHA	-> < NUMEXPR <= NUMEXPR > NUMEXPR >= NUMEXPR == NUMEXPR != NUMEXPR

	ϵ
NUMEXPR	-> TERM NUMEXPR_LINHA
NUMEXPR_LINHA	-> + TERM NUMEXPR_LINHA - TERM NUMEXPR_LINHA ϵ
TERM	-> FACTOR TERM_LINHA
TERM_LINHA	-> * FACTOR TERM_LINHA / FACTOR TERM_LINHA ϵ
FACTOR	-> num (NUMEXPR) id

Justificativas:

Remoção de Recursão à Esquerda:

- Aplicada em NUMEXPR e TERM (criando NUMEXPR_LINHA e TERM_LINHA) para evitar loops infinitos no analisador.

Fatoração à Esquerda:

- IFSTMT: Fatorado para resolver o conflito entre if com e sem else.
- EXPR: Todas as comparações começavam com NUMEXPR. Fatoramos NUMEXPR para EXPR_LINHA.
- ATRIBST: Resolvemos o prefixo comum id = criando ATRIBST_CONTENT.
- PARLIST, VARLIST, PARLISTCALL: Fatorados para lidar com a vírgula.
- RETURNST: Fatorado para suportar return com ou sem valor.
- FLIST e STMTLIST também foram ajustados para remover a ambiguidade.

Justificativa da Alteração em ATRIBST:

- Ao montar a tabela, encontramos um conflito FIRST-FIRST na regra ATRIBST_CONTENT: tanto EXPR quanto FCALL podiam começar com o token id, o que impedia o analisador preditivo de decidir qual produção usar.
- Reescrevemos a regra para que o id seja lido primeiro, e a escolha entre EXPR ou FCALL seja feita depois, usando o novo não terminal ATRIBST_ID.

FIRSTs:

MAIN = { int, id, print, return, if, {, ;, def, ϵ }

FLIST = { def }

FLIST_LINHA = { def, ϵ }

FDEF = { def }

PARLIST = { int, ϵ }

PARLIST_LINHA = { ,, ϵ }

STMTLIST = { int, id, print, return, if, {, ; }

STMTLIST_LINHA = { int, id, print, return, if, {, ;, ϵ }

STMT = { int, id, print, return, if, {, ; }
 VARLIST = { id }
 VARLIST_LINHA = { ,, ε }
 ATRIBST = { id }
 ATRIBST_CONTENT = { num, (, id }
 ATRIBST_ID = { (, *, /, +, -, <, <=, >, >=, ==, !=, ε }
 FCALL = { id }
 PARLISTCALL = { id, ε }
 PARLISTCALL_LINHA = { ,, ε }
 PRINTST = { print }
 RETURNST = { return }
 RETURNST_LINHA = { id, ε }
 IFSTMT = { if }
 IFSTMT_LINHA = { else, ε }
 EXPR = { num, (, id }
 EXPR_LINHA = { <, <=, >, >=, ==, !=, ε }
 NUMEXPR = { num, (, id }
 NUMEXPR_LINHA = { +, -, ε }
 TERM = { num, (, id }
 TERM_LINHA = { *, /, ε }
 FACTOR = { num, (, id }

FOLLOWs:

MAIN = { \$ }
 FLIST = { \$ }
 FLIST_LINHA = { \$ }
 FDEF = { \$, def }
 PARLIST = {) }
 PARLIST_LINHA = {) }
 STMTLIST = { } }
 STMTLIST_LINHA = { } }
 STMT = { int, id, print, return, if, {, ;, }, \$ }
 VARLIST = { ; }
 VARLIST_LINHA = { ; }
 ATRIBST = { ; }
 ATRIBST_CONTENT = { ; }
 ATRIBST_ID = { ; }
 FCALL = { ; }
 PARLISTCALL = {) }
 PARLISTCALL_LINHA = {) }

```

PRINTST = { ; }
RETURNST = { ; }
RETURNST_LINHA = { ; }
IFSTMT = { int, id, print, return, if, {, ;, }, $ }
IFSTMT_LINHA = { int, id, print, return, if, {, ;, }, $ }
EXPR = { ;, ) }
EXPR_LINHA = { ;, ) }
NUMEXPR = { <, <=, >, >=, ==, !=, ;, ) }
NUMEXPR_LINHA = { <, <=, >, >=, ==, !=, ;, ) }
TERM = { +, -, <, <=, >, >=, ==, !=, ;, ) }
TERM_LINHA = { +, -, <, <=, >, >=, ==, !=, ;, ) }
FACTOR = { *, /, +, -, <, <=, >, >=, ==, !=, ;, ) }

```

Exemplo de aplicação:

String: print 10 ;

Passo	Pilha	Entrada Restante	Regra Aplicada
1	MAIN \$	print num ; \$	MAIN -> STMT
2	STMT \$	print num ; \$	STMT -> PRINTST ;
3	PRINTST ; \$	print num ; \$	PRINTST -> print EXPR
4	print EXPR ; \$	print num ; \$	Match (print)
5	EXPR \$	num ; \$	EXPR -> NUMEXPR EXPR_LINHA

6	NUMEXPR EXPR_LINHA ; \$	num ; \$	NUMEXPR -> TERM NUMEXPR_LINHA
7	TERM NUMEXPR_LINHA EXPR_LINHA ; \$	num ; \$	TERM -> FACTOR TERM_LINHA
8	FACTOR TERM_LINHA NUMEXPR_LINHA EXPR_LINHA ; \$	num ; \$	FACTOR -> num
9	num TERM_LINHA NUMEXPR_LINHA EXPR_LINHA ; \$	num ; \$	Match (num)
10	TERM_LINHA NUMEXPR_LINHA EXPR_LINHA ; \$; \$	TERM_LINHA -> ϵ
11	NUMEXPR_LINHA EXPR_LINHA ; \$; \$	NUMEXPR_LINHA -> ϵ
12	EXPR_LINHA ; \$; \$	EXPR_LINHA -> ϵ
13	; \$; \$	Match (;)
14	\$	\$	ACEITA

Tabela de reconhecimento sintático:

	\$	def	id
S	S ::= MAIN \$	S ::= MAIN \$	S ::= MAIN \$
MAIN	MAIN ::= ε	MAIN ::= FLIST	MAIN ::= STMT
FLIST		FLIST ::= FDEF FLIST_LINHA	
FLIST_LINHA	FLIST_LINHA ::= ε	FLIST_LINHA ::= FDEF FLIST_LINHA	
FDEF		FDEF ::= def id (PARLIST) { STMTLIST }	
PARLIST			
PARLIST_LINHA			
STMTLIST			STMTLIST ::= STMT STMTLIST_LINHA
STMTLIST_LINHA			STMTLIST_LINHA ::= STMT STMTLIST_LINHA
STMT			STMT ::= ATRIBST ;
VARLIST			VARLIST ::= id VARLIST_LINHA
VARLIST_LINHA			
ATRIBST			ATRIBST ::= id = ATRIBST_CONTENT
ATRIBST_CONTENT			ATRIBST_CONTENT ::= id ATRIBST_ID
ATRIBST_ID			
PARLISTCALL			PARLISTCALL ::= id PARLISTCALL_LINHA
PARLISTCALL_LINHA			
PRINTST			
RETURNST			
RETURNST_LINHA			RETURNST_LINHA ::= id
IFSTMT			
IFSTMT_LINHA	IFSTMT_LINHA ::= ε		IFSTMT_LINHA ::= ε
EXPR			EXPR ::= NUMEXPR EXPR_LINHA
EXPR_LINHA			
NUMEXPR			NUMEXPR ::= TERM NUMEXPR_LINHA
NUMEXPR_LINHA			
TERM			TERM ::= FACTOR TERM_LINHA
TERM_LINHA			
FACTOR			FACTOR ::= id

()	{
		S ::= MAIN \$
		MAIN ::= STMT
	PARLIST ::= ε	
	PARLIST_LINHA ::= ε	
		STMTLIST ::= STMT STMTLIST_LINHA
		STMTLIST_LINHA ::= STMT STMTLIST_LINHA
		STMT ::= { STMTLIST }
ATRIBST_CONTENT ::= (TERM NUMEXPR_LINHA) TERM_LINHA NUMEXPR_LINHA EXPR_LINHA		
ATRIBST_ID ::= (PARLISTCALL)		
	PARLISTCALL ::= ε	
	PARLISTCALL_LINHA ::= ε	
		IFSTMT_LINHA ::= ε
EXPR ::= NUMEXPR EXPR_LINHA		
	EXPR_LINHA ::= ε	
NUMEXPR ::= TERM NUMEXPR_LINHA		
	NUMEXPR_LINHA ::= ε	
TERM ::= FACTOR TERM_LINHA		
	TERM_LINHA ::= ε	
FACTOR ::= (NUMEXPR)		

	}	int	,	;	=
		S ::= MAIN \$		S ::= MAIN \$	
		MAIN ::= STMT		MAIN ::= STMT	
		PARLIST ::= int id PARLIST_LINHA			
			PARLIST_LINHA ::= , PARLIST		
		STMTLIST ::= STMT STMTLIST_LINHA		STMTLIST ::= STMT STMTLIST_LINHA	
STMTLIST_LINHA ::= ε		STMTLIST_LINHA ::= STMT STMTLIST_LINHA		STMTLIST_LINHA ::= STMT STMTLIST_LINHA	
		STMT ::= int VARLIST ;		STMT ::= ;	
			VARLIST_LINHA ::= , VARLIST	VARLIST_LINHA ::= ε	
				ATRIBST_ID ::= TERM_LINHA NUMEXPR_LINHA EXPR_LINHA	
			PARLISTCALL_LINHA ::= , PARLISTCALL		
				RETURNST_LINHA ::= ε	
IFSTMT_LINHA ::= ε		IFSTMT_LINHA ::= ε		IFSTMT_LINHA ::= ε	
				EXPR_LINHA ::= ε	
				NUMEXPR_LINHA ::= ε	
				TERM_LINHA ::= ε	

num	print	return
	S ::= MAIN \$	S ::= MAIN \$
	MAIN ::= STMT	MAIN ::= STMT
	STMTLIST ::= STMT STMTLIST_LINHA	STMTLIST ::= STMT STMTLIST_LINHA
	STMTLIST_LINHA ::= STMT STMTLIST_LINHA	STMTLIST_LINHA ::= STMT STMTLIST_LINHA
	STMT ::= PRINTST ;	STMT ::= RETURNST ;
ATRIBST_CONTENT ::= num TERM_LINHA NUMEXPR_LINHA EXPR_LINHA		
	PRINTST ::= print EXPR	
		RETURNST ::= return RETURNST_LINHA
	IFSTMT_LINHA ::= ε	IFSTMT_LINHA ::= ε
EXPR ::= NUMEXPR EXPR_LINHA		
NUMEXPR ::= TERM NUMEXPR_LINHA		
TERM ::= FACTOR TERM_LINHA		
FACTOR ::= num		

if	else	<
S ::= MAIN \$		
MAIN ::= STMT		
STMTLIST ::= STMT STMTLIST_LINHA		
STMTLIST_LINHA ::= STMT STMTLIST_LINHA		
STMT ::= IFSTMT		
		ATRIBST_ID ::= TERM_LINHA NUMEXPR_LINHA EXPR_LINHA
IFSTMT ::= if (EXPR) { STMT } IFSTMT_LINHA		
IFSTMT_LINHA ::= ε	IFSTMT_LINHA ::= else { STMT }	
		EXPR_LINHA ::= < NUMEXPR
		NUMEXPR_LINHA ::= ε
		TERM_LINHA ::= ε

[illegible]

$=$	\neq	$+$
ATRIBST_ID ::= TERM_LINHA NUMEXPR_LINHA EXPR_LINHA	ATRIBST_ID ::= TERM_LINHA NUMEXPR_LINHA EXPR_LINHA	ATRIBST_ID ::= TERM_LINHA NUMEXPR_LINHA EXPR_LINHA
EXPR_LINHA ::= == NUMEXPR	EXPR_LINHA ::= != NUMEXPR	
NUMEXPR_LINHA ::= e	NUMEXPR_LINHA ::= e	NUMEXPR_LINHA ::= + TERM NUMEXPR_LINHA
TERM_LINHA ::= e	TERM_LINHA ::= e	TERM_LINHA ::= e

[illegible]