



PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Instituto de Ciências Exatas e de Informática

Departamento de Ciência da Computação

Disciplina: Compiladores

Compilador para a linguagem de programação L

*

Vinicius Francisco da Silva¹

*Trabalho apresentado para a disciplina de compiladores.

¹Aluno do Programa de Graduação em Ciência da Computação, Brasil – vinicius.silva.1046664@sga.pucminas.br.

1 ALFABETO Σ

Tabela 1 – Elementos do alfabeto Σ

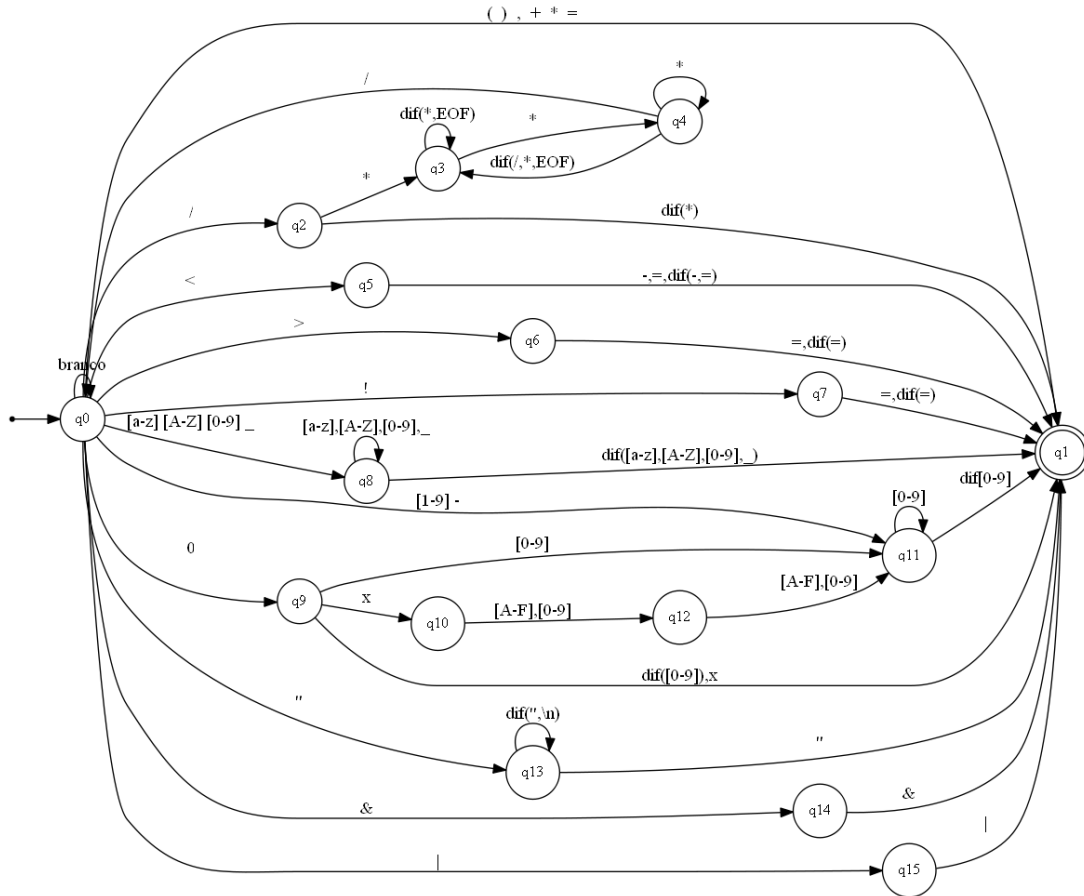
Elemento
<i>final</i>
<i>else</i>
(
<=
;
<i>write</i>
<i>int</i>
&&
)
,
<i>begin</i>
<i>writeln</i>
<i>byte</i>
<
+
<i>endwhile</i>
<i>TRUE</i>
<i>string</i>
!
>
-
<i>endif</i>
<i>FALSE</i>
<i>while</i>
<-
!=
*
<i>endelse</i>
<i>boolean</i>
<i>if</i>
=
>=
/
<i>readln</i>

2 LEXEMAS E PADRÃO DE FORMAÇÃO

Tabela 2 – Lexema x Padrão de formação

Posição	Lexema	Padrão de Formação
1	final	final
2	else	else
3	((
4	<=	<=
5	;	;
6	write	write
7	int	int
8	&&	&&
9))
10	,	,
11	begin	begin
12	writeln	writeln
13	byte	byte
14		
15	<	<
16	+	+
17	endwhile	endwhile
18	TRUE	TRUE
19	string	string
20	!	!
21	>	>
22	-	-
23	endif	endif
24	FALSE	FALSE
25	while	while
26	<-	<-
27	!=	!=
28	*	*
29	endelse	endelse
30	boolean	boolean
31	if	if
32	=	=
33	>=	>=
34	/	/
35	readln	readln

3 ANALISADOR LÉXICO - AFD



4 GRAMÁTICA COM EXPRESSÕES REGULARES - GER

Gramática com Expressões Regulares LL(1) da Linguagem L

S → {**Declarar**}* {**Comando**}*

Declarar → “final” “id” “<-” [“-”] const “;” | (int | boolean | byte | string) id
ListaIds“;”

ListaIds → [**Atrib**] {“,” “id” [**Atrib**]}*

Atrib → “<-” [“-”] const

Comando → **Atribuicao** | **Repeticao** | **Teste** | **Nulo** | **Leitura** | **Escrita**

Atribuicao → “id” “<-” **Expressao** “;”

Repeticao → “while”“(“**Expressao**“)” **Blocowhile**

Blocowhile → “begin”{**Comando**}*“endwhile” | **Comando**

Teste → “if”“(“**Expressao**“)” (**Blocoif** | **Comando** [**BlocoElse**])

Blocoif → “begin”{**Comando**}+“endif”“else”“begin”{**Comando**}+“endelse”

BlocoElse → “else” **Comando**

Nulo → “;”

Leitura → “readln”“(“id“)”“;”

Escrita → “write”“(“**ListaExpressoes**“)”“;” | “writeln”“(“**ListaExpressoes**“)”“;”

ListaExpressoes → **Expressao** {“,” **Expressao**}*

Expressao → **Exp** [(“=” | “!=” | “<” | “>” | “<=” | “>=”) **Exp**]

Exp → [+ | -] **T** {(“+” | “-” | “||”) **T** }*

T → **F** {(“*” | “&&” | “/”) **F** }*

F → “!”**F** | (“**Expressao**“) | [“-”] const | id

5 GRÁMATICA LIVRE DE CONTEXTO - GLC

Gramática Livre de Contexto LL(1) da Linguagem L

S → **Declarar**SA | λ

A → **Comandos**A | λ

Declarar → “final” “id” “<-” **B** const “,” | **C** “id” **ListaIds** “,”

B → “-” | λ

C → “int” | “boolean” | “byte” | “string”

ListaIds → **Atrib** **D**

Atrib → “<-” **B** “const” | λ

D → “,” “id” **Atrib** | λ

Comando → **Atribuicao** | **Repeticao** | **Teste** | **Nulo** | **Leitura** | **Escrita**

Atribuicao → “id” “<-” **Expressao** “,”

Repeticao → “while” “(”**Expressao**“)” **BlocoWhile**

BlocoWhile → “begin” **E** “endwhile” | **Comando**

E → **Comando****E** | λ

Teste → “if” “(”**Expressao**“)” **G**

G → **Blocoif** | **Comando** **BlocoElse**

Blocoif → “begin” **H** “endif” “else” “begin” **H** “endelse”

H → **Comando****H** | **Comando**

BlocoElse → “else” **Comando** | λ

Nulo → “,”

Leitura → “readln”“(”“id”“)”“,”

Escrita → “write”“(”**ListaExpressoes**“)”“,” | “writeln”“(”**ListaExpressoes**“)”“,”

ListaExpressoes \rightarrow **ExpressaoI**

I \rightarrow “,” **ExpressaoI** | λ

Expressao \rightarrow **ExpJ**

J \rightarrow “=” **Exp** | “!=” **Exp** | “<” **Exp** | “>” **Exp** | “<=” **Exp** | “>=” **Exp** | λ

Exp \rightarrow **KTL**

K \rightarrow “+” | “-” | λ

L \rightarrow “+” **TL** | “-” **TL** | “||” **TL** | λ

T \rightarrow **FM**

M \rightarrow “*” **F** | “&&” **F** | “/” **F** | λ

F \rightarrow “!” **F** | “(”**Expressao**“)” | **B** const | “id”

6 GRAMÁTICA COM EXPRESSÕES REGULARES - GER

Gramática com Expressões Regulares LL(1) da Linguagem L

S → {**Declarar**}* {**Comando**}*

Declarar → “final” “id” “<-” [“-”] const “;” | (int | boolean | byte | string) id **ListaIds**“;”

ListaIds → [**Atrib**] {“,” “id” [**Atrib**]}*

Atrib → “<-” [“-”] const

Comando → **Atribuicao** | **Repeticao** | **Teste** | **Nulo** | **Leitura** | **Escrita**

Atribuicao → “id” “<-” **Expressao** “;”

Repeticao → “while”“(“**Expressao**“)” **Blocowhile**

Blocowhile → “begin”{**Comando**}*“endwhile” | **Comando**

Teste → “if”“(“**Expressao**“)” (**Blocoif** | **Comando** [**BlocoElse**])

Blocoif → “begin”{**Comando**}+“endif”“else”“begin”{**Comando**}+“endelse”

BlocoElse → “else” **Comando**

Nulo → “;”

Leitura → “readln”“(“id“)”“;”

Escrita → “write”“(“**ListaExpressoes**“)”“;” | “writeln”“(“**ListaExpressoes**“)”“;”

ListaExpressoes → **Expressao** {“,” **Expressao**}*

Expressao → **Exp** [(“=” | “!=” | “<” | “>” | “<=” | “>=”) **Exp**]

Exp → [+ | -] **T** {(“+” | “-” | “||”) **T** }*

T → **F** {(“*” | “&&” | “/”) **F** }*

F → “!”**F** | (“**Expressao**“) | [“-”] const | id
