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 Silva[1]

*Trabalho apresentado para a disciplina de compiladores.

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

1 ALFABETO Σ

Tabela 1 – Elementos do alfabeto Σ

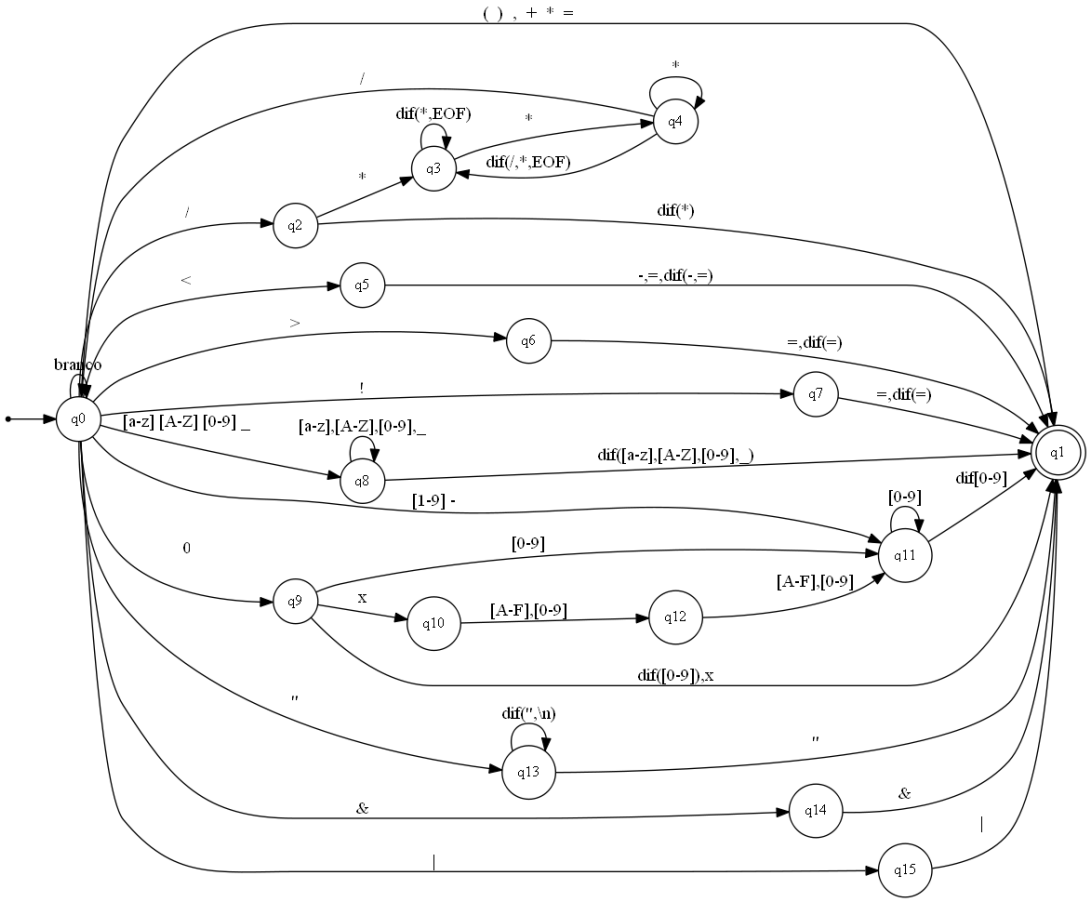
Elemento
final
else
(
<=
;
write
int
&&
)
,
begin
writeln
byte
||
<
+
endwhile
TRUE
string
!
>
-
endif
FALSE
while
<-
!=
*****
endelse
boolean
if
=
>=
/
readln

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 GRÁMÁTICA LIVRE DE CONTEXTO - GLC

Gramática Livre de Contexto LL(1) da Linguagem L

S → **DeclararSA** | λ

A → **ComandosA** | 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 → **ComandoE** | λ

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

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

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

H → **ComandoH** | **Comando**

BlocoElse → “else” **Comando** | λ

Nulo → “;”

Leitura → “readln”“(”“id”“)”“;”

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

ListaExpressoes → ExpressaoI

I → “,” **ExpressaoI |** λ

Expressao → ExpJ

J → “=” **Exp |** “!=” **Exp |** “<” **Exp |** “>” **Exp |** “<=” **Exp |** “>=” **Exp |** λ

Exp → KTL

K → “+” **|** “-” **|** λ

L → “+” **TL |** “-” **TL |** “||” **TL |** λ

T → FM

M → “*” **F |** “&&” **F |** “/” **F |** λ

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