Etapa 1 - Projeto da Linguagem

Aluno: Felipe Alves Belisário
Matrícula: 11721BCC030

➢ Identificação dos tokens e expressões regulares:

O token para espaços em branco e quebra de linha foi colocado na tabela no intuito de que no momento em que o analisador léxico identificar o lexema que representa esse token a análise será reiniciada e começará o processo de reconhecimento do próximo lexema.

A mesma lógica se aplica para os tokens de comentário, a partir do momento que for reconhecido o token de início do comentário tudo que vier em seguida será ignorado até que seja reconhecido um token de fim de comentário.

Todos os tokens terão de tipos de atributos em comum:

- linha
- coluna

Logo, os atributos serão um tipo estruturado com várias informações adicionais (os tipos acima não serão adicionados na tabela a seguir para todos os tokens).

Token	Expressão Regular	Atributos
letra	\A | … | \Z | \a | … | \z	valor
letras	letra+	valor
digito	0 | … | 9	valor
digitos	digito+	valor
fracao	\. digitos	valor
expoente	(\E (aritmop | ε) digitos) | ε	valor
cnt_real	\-? digitos fracao expoente	valor
cnt_int	\-? digitos	valor

cnt_char	\’ letra \’	valor
id	letra (letra | digito | _)*	posição na tabela de símbolos
ws	“\t” | “\n” | “ “	
relop	\> | \< | “>=” | “<=” | “==” | “<>”	valor (*GT* ou *LT* ou *GE* ou *LE* ou *EQ* ou *NE*)
boolop	“and” | “or”	valor (*AND* ou *OR*)
aritmop	\+ | \- | * | \/	valor (*PL* ou *MN* ou *MT* ou *DV*)
se	“se”	
entao	“entao”	
senao	“senao”	
principal	“procedure principal”	
inicio	“Inicio”	
fim	“Fim”	
enquanto	“enquanto”	
repita	“repita”	
tipo	“integer” | “char” | “real”	valor (*INT* ou *CHAR* ou *REAL*)
(	\(	
)	\)	
:=	:=	
;	\;	
:	\:	
,	\,	
inicio_comentario	\{	
fim_comentario	\}	

➢ Gramática Livre de Contexto (GLC):

G = (V, T, P, main)

V = {

ids,
declaracao ,
expr_aritm ,
expr_relop ,
expr_bool ,
termo ,
atrib ,
if_sozinho ,
if ,
while ,
do_while,
expr_bloco ,
exprs_bloco ,
bloco ,
main

}

T = {

cnt_real ,
cnt_int ,
cnt_char ,
id ,
tipo ,
aritmop ,
relop ,
boolop ,
se ,
entao ,
senao ,
enquanto ,
repita ,

inicio ,
fim ,
principal ,
: ,
, ,
(,
) ,
; ,
:=

}

P = {

ids	→	*ids* **,** **id** | **id**
declaracao	→	**tipo :** *ids* ***;***
expr_aritm	→	*expr_aritm* **aritmop** *termo* | *termo*
expr_relop		*expr_aritm* **relop** *expr_aritm* | **(** *expr_relop* **)**
expr_bool	→	*expr_bool* **boolop** *expr_relop* | *expr_relop* | **(** *expr_bool* **)**
termo	→	**id** | **cnt_int** | **cnt_real** | **cnt_char** | **(** *expr_aritm* **)**
atrib	→	**id :=** *expr_aritm*
if_sozinho	→	**se (** *expr_bool* **) entao** *bloco*
if	→	*if_sozinho* **senao** *bloco* | *if_sozinho*
while	→	**enquanto (** *expr_bool* **)** *bloco*
do_while	→	**repita** *bloco* **enquanto (** *expr_bool* **);**
expr_bloco	→	*declaracao* | *atrib* | *if* | *while* | *do_while*
exprs_bloco	→	*exprs_bloco expr_bloco* | ε
bloco	→	**inicio** *exprs_bloco* **fim**

main → **principal** *bloco*

}