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 atributos em comum:

- linha
- coluna

Logo, esses atributos não serão adicionados na tabela a seguir.

Token	Expressão Regular	Atributos
letra	\A | … | \Z | \a | … | \z	lexema
letras	letra+	lexema
digito	0 | … | 9	lexema
digitos	digito+	lexema
fracao	\. digitos	lexema
expoente	(\E (aritmop | ε) digitos) | ε	lexema
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	\> | \< | “>=” | “<=” | “==” | “<>”	tipo
boolop	“and” | “or”	tipo
aritmop	\+ | \- | * | \/	tipo
se	“se”	
entao	“entao”	
senao	“senao”	
principal	“procedure principal”	
inicio	“Inicio”	
fim	“Fim”	
enquanto	“enquanto”	
repita	“repita”	
tipo	“integer” | “char” | “real”	tipo
(	\(	
)	\)	
:=	:=	
;	\;	
:	\:	
,	\,	
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*

}