Especificação da linguagem

Definição de uma GLC

Seja uma Gramática Livre de Contexto (GLC) G uma quádrupla (V, T, P, S), onde:

V conjunto finito de variáveis ou símbolo não-terminais

T (conjunto de terminais) é um subconjunto de V

P (conjunto de regras (ou produções)) é um subconjunto finito de (V -T) x V*

S (símbolo inicial (ou variável de início)) é um elemento de V

Sabendo disso podemos definir nossa linguagem da seguinte forma:

S → programa nome_programa
block

block → begin variable_declaration
command_sequence
end

variable_declaration → ε | type : id_lists ; | variable_declaration

id_lists → identifier ; | identifier, id_lists

command_sequence → ε | command | command_sequence

command → selection | loop | assignment

selection → if (condition) then
block
else
block
→ if (condition) then
block

loop → while (condition) do
block

→ repeat

block

while (condition)

assigment → identifier := expression

condition → expression relational_operator expression

expression → identifier | constant | (expression) | expression aritmetic_operator expression

constant → number | character

comments → [any_word]

Identificação dos Tokens

Token	**Atributo**
programa	<Não possui>
nome_programa	<Tabela de símbolos>
begin	<Não possui>
end	<Não possui>
if	<Não possui>
then	<Não possui>
else	<Não possui>
while	<Não possui>
repeat	<Não possui>
:=	<Não possui>
identifier	<Tabela de símbolos>
number	<Tabela de símbolos>
character	<Tabela de símbolos>
type	<Tabela de símbolos>
relational_operator	<Tabela de símbolos>
aritmectic_operator	<Tabela de símbolos>
,	<Não possui>
:	<Não possui>
;	<Não possui>
(	<Não possui>
)	<Não possui>
[	<Não possui>

]	<Não possui>
do	<Não possui>

Definição dos Padrões

Token	**Expressão Regular**
programa	programa
identifier	[a-zA-Z]([a-zA-Z0-9])*
nome_programa	[a-zA-Z]([a-zA-Z0-9])*
begin	begin
type	(float|int|char)
:	:
;	;
if	if
(	(
)	)
then	then
else	else
while	while
do	do
repeat	repeat
:=	:=
number	[0-9]([0-9])*(.[0-9]([0-9])*)?([Ee][+-]?[0-9]([0-9])*)?
relational_operator	(=|~=|< | >| <=|>=)
aritmetic_operator	(+|-|*|/|^)
character	\'[a-zA-Z]?\'
any_word	([A-Za-z0-9\s])*
,	,
[	[
]	]