

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 símbolos não-terminais

T (conjunto de terminais) é um subconjunto de V

P (conjunto de regras (ou produções)) é um subconjunto finito de $(V - T) \times V^*$

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

Sabendo disso podemos definir nossa linguagem da seguinte forma:

$S \rightarrow \text{programa nome_programa}$

block

$\text{block} \rightarrow \text{begin}$

$\text{variable_declaration}$

command_sequence

end

$\text{variable_declaration} \rightarrow \epsilon \mid \text{type} : \text{id_list} ; \mid \text{variable_declaration}$

$\text{id_list} \rightarrow \text{identifier} ; \mid \text{identifier}, \text{id_list}$

$\text{command_sequence} \rightarrow \epsilon \mid \text{command} \mid \text{command_sequence}$

$\text{command} \rightarrow \text{selection} \mid \text{loop} \mid \text{assignment}$

$\text{selection} \rightarrow \text{if (condition) then}$

block

else

block

$\rightarrow \text{if (condition) then}$

block

loop → while (condition) do

 block

→ repeat

 block

while(condition)

assignment → identifier := expression;

condition → expression relational_operator expression

expression → identifier | constant | (expression) | expression arithmetic_operator expression

constant → number | character

Identificação de 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>
arithmetic_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	(= ~= < > <= >=)
arithmetic_operator	(+ - / * ^)
character	'[a-zA-Z]'
comments	[([A-Za-z0-9\s])*)
,	,
[[
]]
space	(' ' '\n' '\t')