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) 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_list ; | variable_declaration

id_list → identifier ; | identifier, id_list

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)

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')