## Linguagem LALG

```
2. <corpo> ::= <dc> begin <comandos> end
3. <dc> ::= <dc_v> <dc_p>
4. <dc_v> ::= var <variaveis> : <tipo_var> ; <dc_v> | \lambda
5. <tipo_var> ::= real | integer
6. <variaveis> ::= ident <mais_var>
7. <mais_var> ::= , <variaveis> | \lambda
8. \langle dc_p \rangle ::= procedure ident \langle parametros \rangle ; \langle corpo_p \rangle \langle dc_p \rangle | \lambda
9. 9.  ( lista_par> ) | \lambda
10. lista_par> ::= <variaveis> : <tipo_var> <mais_par>
11. <mais par> ::= ; sta par> |\lambda|
12. <corpo p> ::= <dc loc> begin <comandos> end;
13. <dc loc> ::= <dc v>
14. < lista arg> ::= ( < argumentos> ) | \lambda
15. <argumentos> ::= ident <mais_ident>
16. <mais_ident> ::= ; <argumentos> |\lambda|
17. \langle pfalsa \rangle ::= else \langle cmd \rangle | \lambda
18. <comandos> ::= <cmd> ; <comandos> | \lambda
19. <cmd> ::= read ( <variaveis> ) |
                 write ( <variaveis> ) |
                 while <condicao> do <cmd> |
                 if <condicao> then <cmd> <pfalsa> |
                ident := <expressao> |
                ident < lista_arg > |
                begin <comandos> end
20. <condicao> ::= <expressao> <relacao> <expressao>
21. <relacao> ::= = | <> | >= | <= | > | <
22. <expressao> ::= <termo> <outros_termos>
23. \langle op\_un \rangle := + | - | \lambda
24. \langle outros\_termos \rangle ::= \langle op\_ad \rangle \langle termo \rangle \langle outros\_termos \rangle | \lambda
25. \langle op_ad \rangle := + 1 -
26. <termo> ::= <op un> <fator> <mais fatores>
27. <mais_fatores> ::= <op_mul> <fator> <mais_fatores> |\lambda|
28. <op mul> ::= * | /
29. <fator> ::= ident | numero int | numero real | ( <expressao> )
```

Além disso, inclusão do comando repeat-until.

Comentários entre chaves { }

Identificadores e números são itens léxicos da forma:

- ident: sequência de letras e dígitos, começando por letra
- número inteiro: sequência de dígitos (0 a 9)
- número real: pelo menos um dígito seguido de um ponto decimal e de uma seqüência de um ou mais dígitos

## Exemplos de programas LALG

```
program nome1;
                                                                 program nome2;
{exemplo 1}
                                                                 {exemplo 2}
var a, a1, b: integer;
                                                                 var a: real;
begin
                                                                 var b: integer;
read(a, b);
                                                                 procedure nomep(x: real);
a1 := a + b;
                                                                 var a, c: integer;
while a1>a do
                                                                 begin
begin
                                                                 read(c, a);
write(a1);
                                                                 if a<x+c then
a1 := a1-1;
                                                                 begin
end:
                                                                 a := c + x;
if a <> b then write(a);
                                                                 write(a);
end.
                                                                 end
                                                                 else c := a + x;
                                                                 begin {programa principal}
                                                                 read(b);
                                                                 nomep(b);
                                                                 end.
```