

1. <programa>	::= <declaracoes> algoritmo <corpo> fim_algoritmo
2. <declaracoes>	::= <decl_local_global> <declaracoes> ε
3. <decl_local_global>	::= <declaracao_local> <declaracao_global>
4. <declaracao_local>	::= declare <variavel> constante IDENT : <tipo_basico> = <valor_constante> tipo IDENT : <tipo>
5. <variavel>	::= IDENT <dimensao> <mais_var> : <tipo>
6. <mais_var>	::= , IDENT <dimensao> <mais_var> ε
7. <identificador>	::= <ponteiros_opcionais> IDENT <dimensao> <outros_ident>
8. <ponteiros_opcionais>	::= ^ <ponteiros_opcionais> ε
9. <outros_ident>	::= . <identificador> ε
10. <dimensao>	::= [<exp_aritmetica>] <dimensao> ε
11. <tipo>	::= <registro> <tipo_estendido>
12. <mais_ident>	::= , <identificador> <mais_ident> ε
13. <mais_variaveis>	::= <variavel> <mais_variaveis> ε
14. <tipo_basico>	::= literal inteiro real logico
15. <tipo_basico_ident>	::= <tipo_basico> IDENT
16. <tipo_estendido>	::= <ponteiro_opcional> <tipo_basico_ident>
17. <valor_constante>	::= CADEIA NUM_INT NUM_REAL verdadeiro falso
18. <registro>	::= registro <variavel> <mais_variaveis> fim_registro
19. <declaracao_global>	::= procedimento IDENT (<parametros_opcional>) <declaracoes_locais> <comandos> fim_procedimento funcao IDENT (<parametros_opcional>) : <tipo_estendido> <declaracoes_locais> <comandos> fim_funcao
20. <parametros_opcional>	::= <parametro> ε
21. <parametro>	::= <var_opcional> <identificador> <mais_ident> : <tipo_estendido> <mais_parametros>
22. <var_opcional>	::= var ε
23. <mais_parametros>	::= , <parametro> ε
24. <declaracoes_locais>	::= <declaracao_local> <declaracoes_locais> ε
25. <corpo>	::= <declaracoes_locais> <comandos>
26. <comandos>	::= <cmd> <comandos> ε
27. <cmd>	::= leia (<identificador> <mais_ident>) escreva (<expressao> <mais_expressao>) se <expressao> entao <comandos> <senao_opcional> fim_se caso <exp_aritmetica> seja <selecao> <senao_opcional> fim_caso para IDENT <- <exp_aritmetica> ate <exp_aritmetica> faca <comandos> fim_para enquanto <expressao> faca <comandos> fim_enquanto faca <comandos> ate <expressao> ^ IDENT <outros_ident> <dimensao> <- <expressao> IDENT <chamada_atribuicao> retorne <expressao>
28. <mais_expressao>	::= , <expressao> <mais_expressao> ε
29. <senao_opcional>	::= senao <comandos> ε
30. <chamada_atribuicao>	::= (<argumentos_opcional>) <outros_ident> <dimensao> <- <expressao>
31. <argumentos_opcional>	::= <expressao> <mais_expressao> ε
32. <selecao>	::= <constantes> : <comandos> <mais_selecao>
33. <mais_selecao>	::= <selecao> ε
34. <constantes>	::= <numero_intervalo> <mais_constantes>
35. <mais_constantes>	::= , <constantes> ε
36. <numero_intervalo>	::= <op_unario> NUM_INT <intervalo_opcional>
37. <intervalo_opcional>	::= .. <op_unario> NUM_INT ε
38. <op_unario>	::= - ε
39. <exp_aritmetica>	::= <termo> <outros_termos>
40. <op_multiplicacao>	::= * /
41. <op_adicao>	::= + -
42. <termo>	::= <fator> <outros_fatores>
43. <outros_termos>	::= <op_adicao> <termo> <outros_termos> ε
44. <fator>	::= <parcela> <outras_parcelas>
45. <outros_fatores>	::= <op_multiplicacao> <fator> <outros_fatores> ε
46. <parcela>	::= <op_unario> <parcela_unario> <parcela_nao_unario>
47. <parcela_unario>	::= ^ IDENT <outros_ident> <dimensao> IDENT <chamada_partes> NUM_INT NUM_REAL (<expressao>)
48. <parcela_nao_unario>	::= & IDENT <outros_ident> <dimensao> CADEIA
49. <outras_parcelas>	::= % <parcela> <outras_parcelas> ε
50. <chamada_partes>	::= (<expressao> <mais_expressao>) <outros_ident> <dimensao> ε
51. <exp_relacional>	::= <exp_aritmetica> <op_opcional>
52. <op_opcional>	::= <op_relacional> <exp_aritmetica> ε
53. <op_relacional>	::= = <> >= <= > <
54. <expressao>	::= <termo_logico> <outros_termos_logicos>
55. <op_nao>	::= nao ε
56. <termo_logico>	::= <fator_logico> <outros_fatores_logicos>
57. <outros_termos_logicos>	::= ou <termo_logico> <outros_termos_logicos> ε
58. <outros_fatores_logicos>	::= e <fator_logico> <outros_fatores_logicos> ε
59. <fator_logico>	::= <op_nao> <parcela_logica>
60. <parcela_logica>	::= verdadeiro falso <exp_relacional>

Comentários entre chaves { } ocupando uma ou várias linhas.

Identificadores, números e cadeias de literais são itens léxicos da forma:

- IDENT: sequência de letras, dígitos e underscore (_), começando por letra ou underscore
- NUM_INT: sequência de dígitos (0 a 9)
- NUM_REAL: pelo menos um dígito seguido de um ponto decimal e de uma sequência de um ou mais dígitos
- CADEIA: sequência de caracteres entre aspas duplas (") de apenas uma linha (= sem mudança de linha – ENTER)

Exemplos de programas SEM erros em LA

algoritmo_6-10_apostila_LA.txt

```
{ dado o comprimento de um arco, calcular seu cosseno pela soma  
cos(x) = S x^i/i! para um dado numero de termos }
```

```
algoritmo
```

```
declare i, termo, baseFatorial, fatorial, numeroTermos: inteiro  
declare cosseno, angulo: real
```

```
{ leitura do arco e do numero de termos }  
leia(angulo, numeroTermos) { angulo em radianos }
```

```
{ calculo da aproximacao do cosseno }  
cosseno <- 0 { acumulador do resultado }  
baseFatorial <- 1  
fatorial <- 1  
termo <- 1  
para i <- 1 ate numeroTermos faca  
{ faz o somatorio }  
se i % 2 = 1 entao  
cosseno <- cosseno + termo { soma termos impares }  
senao  
cosseno <- cosseno - termo { subtrai termos pares }  
fim_se
```

```
{ calcula o proximo termo }  
fatorial <- fatorial * baseFatorial * (baseFatorial + 1)  
baseFatorial <- baseFatorial + 2  
termo <- pot(x, i + 1)/fatorial  
fim_para
```

```
{ resultado calculado }  
escreva("cos(", angulo, ") = ", cosseno)  
fim_algoritmo
```

Saídas geradas para esses programas

Fim da compilacao

algoritmo_9-4_apostila_LA.txt

```
{ exemplificacao de sub-rotinas na forma de funcao e seu uso }
```

```
funcao menorInteiro(valor1: inteiro, valor2: inteiro): inteiro  
{ retorna o menor entre valor1 e valor2; se iguais retorna um deles }
```

```
se valor1 < valor2 entao  
retorne valor1  
senao  
retorne valor2  
fim_se  
fim_funcao
```

```
funcao menorReal(valor1: real, valor2: real): real  
{ retorna o menor entre valor1 e valor2; se iguais, retorna um deles }
```

```
se valor1 < valor2 entao  
retorne valor1  
senao  
retorne valor2  
fim_se  
fim_funcao
```

```
funcao modulo(valor: real): real  
{ retorna o valor absoluto do valor }
```

```
se valor < 0 entao  
valor <- -valor  
fim_se  
retorne valor  
fim_funcao
```

```
{ parte principal }
```

```
algoritmo
```

```
declare  
primeiroInt, segundoInt: inteiro  
declare  
primeiroReal, segundoReal: real
```

```
{ entrada de dados }  
leia(primeiroInt, segundoInt, primeiroReal, segundoReal)
```

```
{ algumas saidas e manipulacoes }  
escreva("O menor inteiro entre", primeiroInt, "e",  
segundoInt, "eh", menor(primeiroInt, segundoInt))
```

```
se menorReal(primeiroReal, segundoReal) <> primeiroReal entao  
escreva(segundoReal, "eh menor que", primeiroReal)  
fim_se
```

```
se modulo(primeiroReal) = primeiroReal e primeiroReal <> 0 entao  
escreva("O valor", primeiroReal, "eh positivo")  
senao  
escreva("O valor", primeiroReal, "nao eh positivo")  
fim_se
```

```
escreva("Considerando-se o modulo, tem-se que o menor entre",  
primeiroReal, "e", segundoReal, "eh",  
menorReal(modulo(primeiroReal), modulo(segundoReal)))
```

```
fim_algoritmo
```

Fim da compilacao

Exemplos de programas COM erros em LA

40-algoritmo_6-10_apostila_LA_erro_linha_26_acusado_linha_27.txt

```
1. { dado o comprimento de um arco, calcular seu cosseno pela soma  
cos(x) = S x^i/i! para um dado numero de termos }  
2.  
3. algoritmo  
4. declare i, termo, baseFatorial, fatorial, numeroTermos: inteiro  
5. declare cosseno, angulo: real  
6.  
7. { leitura do arco e do numero de termos }  
8. leia(angulo, numeroTermos) { angulo em radianos }  
9.  
10. { calculo da aproximacao do cosseno }  
11. cosseno <- 0 { acumulador do resultado }  
12. baseFatorial <- 1  
13. fatorial <- 1  
14. termo <- 1  
15. para i <- 1 ate numeroTermos faca  
16. { faz o somatorio }  
17. se i % 2 = 1 entao  
18. cosseno <- cosseno + termo { soma termos impares }  
19. senao  
20. cosseno <- cosseno - termo { subtrai termos pares }  
21. fim_se  
22.  
23. { calcula o proximo termo }  
24. fatorial <- fatorial * baseFatorial * (baseFatorial + 1)  
25. baseFatorial <- baseFatorial + 2  
26. termo <- pot(x, i + 1/fatorial  
27. fim_para  
28.  
29. { resultado calculado }  
30. escreva("cos(", angulo, ") = ", cosseno)  
31. fim_algoritmo
```

54-algoritmo_9-4_apostila_LA_erro_linha_3.txt

```
1. { exemplificacao de sub-rotinas na forma de funcao e seu uso }  
2.  
3. funcao (valor1: inteiro, valor2: inteiro): inteiro  
4. { retorna o menor entre valor1 e valor2; se iguais retorna um deles }  
5.  
6. se valor1 < valor2 entao  
7. retorne valor1  
8. senao  
9. retorne valor2  
10. fim_se  
11. fim_funcao  
12.  
13. funcao menorReal(valor1: real, valor2: real): real  
14. { retorna o menor entre valor1 e valor2; se iguais, retorna um deles }  
15.  
16. se valor1 < valor2 entao  
17. retorne valor1  
18. senao  
19. retorne valor2  
20. fim_se  
21. fim_funcao  
22.  
23. funcao modulo(valor: real): real  
24. { retorna o valor absoluto do valor }  
25.  
26. se valor < 0 entao  
27. valor <- -valor  
28. fim_se  
29. retorne valor  
30. fim_funcao  
31.  
32. { parte principal }  
33. algoritmo  
34. declare  
35. primeiroInt, segundoInt: inteiro  
36. declare  
37. primeiroReal, segundoReal: real  
38.  
39. { entrada de dados }  
40. leia(primeiroInt, segundoInt, primeiroReal, segundoReal)  
41.  
42. { algumas saidas e manipulacoes }  
43. escreva("O menor inteiro entre", primeiroInt, "e",  
44. segundoInt, "eh", menor(primeiroInt, segundoInt))  
45.  
46. se menorReal(primeiroReal, segundoReal) <> primeiroReal  
entao  
47. escreva(segundoReal, "eh menor que", primeiroReal)  
48. fim_se  
49.  
50. se modulo(primeiroReal) = primeiroReal e primeiroReal <> 0  
entao  
51. escreva("O valor", primeiroReal, "eh positivo")  
52. senao  
53. escreva("O valor", primeiroReal, "nao eh positivo")  
54. fim_se  
55.  
56. escreva("Considerando-se o modulo, tem-se que o menor entre",  
57. primeiroReal, "e", segundoReal, "eh",  
58. menorReal(modulo(primeiroReal), modulo(segundoReal)))  
59. fim_algoritmo
```

Saídas geradas para esses programas com erros

Linha 27: erro sintatico proximo a fim_para
Fim da compilacao

Linha 3: erro sintatico proximo a (
Fim da compilacao