

LALG Compiler

Tratamento de Erros Sintáticos

Obs: baseado completamente nos firsts e follows dos termos, bem como a adição de alguns elementos para facilitar a recuperação, como palavras reservadas e principalmente o ponto e vírgula que facilita encontrar finalizações de partes de código.

<PROGRAM>

Permitir erro na palavra reservada program, desde que se tenha um identificador a frente.

```
1 program id;
2 begin
3   a:=1
4 end.
5
```

No file chosen

Error	Line	Column
program expected	1	1

Informa a falta de ponto e vírgula depois do identificador de programa:

```
1 program id
2 begin
3   a:=1
4 end.
5
```

No file chosen

Error	Line	Column
',' expected	1	11

Fal de ponto no final do programa

<BLOC>

Corresponde a união dos erros em <composite_command>, <variable_declaration_part> e <subroutine_declaration_part>

<VARIABLE_DECLARATION>

Corresponde aos erros em <identifier_list>.

<ASSIGNMENT>

Permite esquecer “:” antes do igual: _____

1 program exemplo;
2 begin
3 a = a+1
4 end.

Choose File

No file chosen

Confirmar

Error	Line	Column
'=' expected	3	4

Permite erro ou falta da expressão depois do igual: _____

1 program exemplo;
2 begin
3 a :=
4 end.

Choose File

No file chosen

Confirmar

Error	Line	Column
expression expected before ':='	3	5

<COMPOSITE_COMMAND>

Palavra reservada begin escrita errada: _____

1 program exemplo;
2 begins
3 a:=1
4 end.

Choose File

No file chosen

Confirmar

Error	Line	Column
begin expected	2	1

Palavra reservada “end” faltando: _____

1 program exemplo;
2 begin
3 read(a)
4 ends.
5

Choose File

No file chosen

Confirmar

Error	Line	Column
end expected	4	1

<CONDITIONAL_COMMAND_1>

Permite escrever a palavra reservada “then” de forma errada, ou mesmo esquecer a mesma:

1

program exemplo;

2

begin

3

if (true)

4

a:= true

5

else

6

a:= false

7

end.

8

Choose File

No file chosen

Confirmar

Error	Line	Column
then expected	4	6

1

program exemplo;

2

begin

3

if (true)

4

a:= true

5

else

6

a:= false

7

end.

8

Choose File

No file chosen

Confirmar

Error	Line	Column
then expected	4	6

<REPETITIVE_COMMAND_1>

Permite escrever a palavra reservada “do” de forma errada, ou mesmo esquecer a mesma:

```
1 program exemplo;
2 begin
3     while (a > b) dos
4         write(a)
5 end.
6
```

Choose File No file chosen

Confirmar

Error	Line	Column
'do' expected	3	16

```
1 program exemplo;
2 begin
3     while (a > b)
4         write(a)
5 end.
6
```

Choose File No file chosen

Confirmar

Error	Line	Column
'do' expected	4	6

<PROCEDURE_CALL>

Permite até 2 caractéres errados antes do primeiro “(“ e trata a falta de “)” e “.””:

1 program exemplo;
2 begin
3 teste..(a, b, c
4 end.
5

Choose File

No file chosen

Confirmar

Error	Line	Column
unexpected '.'	3	7
missing ')'	4	1

1 program exemplo;
2 begin
3 teste a,b)
4 end.
5

Choose File

No file chosen

Confirmar

Error	Line	Column
missing '('	3	8

<COMMAND>

Agrega os erros de <procedure_call>, <assignment>, <procedure_call>, <composite_command>, <repetitive_command> e <conditional_command>.

<EXPRESSION_LIST>

Permite faltar vírgula entre os elementos:

1
2
3
4
5

```
program exemplo;  
begin  
    teste(a b, c)  
end.
```

Choose File

No file chosen

Confirmar

Error	Line	Column
',' expected	3	10

Informa caso se tenha colocado uma vírgula a mais ou esquecido a expressão depois da mesma:

1
2
3
4
5

```
program exemplo;  
begin  
    teste(a > b, not a, )  
end.
```

Choose File

No file chosen

Confirmar

Error	Line	Column
missing expression after ','	3	22

<EXPRESSION>

Informa erro se parte de expressão está faltando ou está errada:

1 program exemplo;
2 begin
3 while (a + b > ###) do
4 write(a)
5 end.
6
7

Choose File

No file chosen

Confirmar

Error	Line	Column
simple expression expected after relation	3	17

Trata até dois caracteres errados entre os membros da expressão:

1 program exemplo;
2 begin
3 while (a + b >&* 12) do
4 write(a)
5 end.
6
7

Choose File

No file chosen

Confirmar

Error	Line	Column
unexpected '&*'	3	16

<SIMPLE_EXPRESSION>

Permite até 3 caracteres inválidos entre os membros, tanto antes quanto depois:

```
1 program exemplo;  
2 begin  
3     if (... + a or b) then  
4         read(a)  
5 end.  
6  
7
```

Choose File No file chosen

Confirmar

Error	Line	Column
unexpected '...'	3	6

```
1 program exemplo;  
2 begin  
3     if ($#&+ a or b or c or. d) then  
4         read(a)  
5 end.  
6  
7
```

Choose File No file chosen

Confirmar

Error	Line	Column
unexpected '\$#&'	3	6
unexpected factor '.'	3	25

<THERM>, <FACTOR>

Trata até 3 caracteres inválidos entre os operadores e operandos:

```
1 program exemplo;  
2 begin  
3     res:=a*&[ b  
4 end.  
5  
6
```

Choose File No file chosen

Confirmar

Error	Line	Column
unexpected factor '&'	3	9
unexpected factor '['	3	10

Termo faltando depois do not:

```
1 program exemplo;  
2 begin  
3     res:=not ;  
4     write(res)  
5 end.  
6  
7
```

Choose File No file chosen

Confirmar

Error	Line	Column
expected factor before not	3	11

Faltando fechamento de parenteses no fator:

```
1 program exemplo;  
2 begin  
3     res:=c*(a+b;  
4     write(res)  
5 end.  
6  
7
```

Choose File No file chosen

Confirmar

Error	Line	Column
')' expected	3	13

<VARIABLE_DECLARATION_PART>

Permite a ocorrência de caracteres inválidos antes do ponto e vírgula:

```
1 program exemplo;  
2 int a, b **;  
3 boolean c;  
4 begin  
5     res:=a+b;  
6     write(res)  
7 end.  
8  
9
```

Choose File No file chosen

Confirmar

Error	Line	Column
unexpected '**' before ';'	2	10

Também trata a falta do ponto e vírgula no final de uma declaração de variável:

```
1 program exemplo;  
2 int a, b  
3 boolean c;  
4 begin  
5     res:=a+b;  
6     write(res)  
7 end.  
8  
9
```

Choose File No file chosen

Confirmar

Error	Line	Column
';' expected	3	1

<VARIABLE_DECLARATION>

Corresponde aos erros para <identifier_list>, pois necessita obrigatoriamente da apresentação de um tipo simples para ser reconhecida.

<IDENTIFIER_LIST>

Permite que a vírgula esteja faltando entre os identificadores:

```
1 program exemplo;
2 int a b;
3 boolean c d;
4 begin
5     res:=a+b;
6     write(res)
7 end.
8
9
```

Choose File No file chosen

Confirmar

Error	Line	Column
';' expected	2	7
';' expected	3	11

Também acusa a falta de identificador depois da vírgula:

```
1 program exemplo;
2 int a, ;
3 boolean c,;
4 begin
5     res:=a+b;
6     write(res)
7 end.
8
9
```

Choose File No file chosen

Confirmar

Error	Line	Column
missing identifier after ','	2	8
missing identifier after ','	3	11

<SUBROUTINE_DECLARATION_PART>

Corresponde aos erros da <procedure_declaration> e tratamentos dos caracteres inválidos antes do ponto e vírgula, bem como a sua falta:

```
1 program exemplo;  
2 procedure proc1(var i, j : int);  
3 begin  
4     a:=i+j  
5 end &;  
6 begin  
7     proc1(1,2)  
8 end.  
9  
10
```

Choose File No file chosen

Confirmar

Error	Line	Column
unexpected '&#' before ';'	5	5

Ponto e vírgula faltando:

```
1 program exemplo;  
2 procedure proc1(var i, j : int);  
3 begin  
4     a:=i+j  
5 end  
6 begin  
7     proc1(1,2)  
8 end.  
9  
10
```

Choose File No file chosen

Confirmar

Error	Line	Column
';' expected	6	1

<PROCEDURE_DECLARATION>

Corresponde a concatenação dos erros para <formal_parameters> e <bloc> permitindo que se digite a palavra reservada “procedure”, desde que se tenha depois outro identificador:

1 program exemplo;
2 procedur proc1(var i, j : int);
3 begin
4 a:=i+j
5 end;
6 begin
7 proc1(1,2)
8 end.
9
10

Choose File

No file chosen

Confirmar

Error	Line	Column
keyword 'procedure' expected	2	1

Permite ponto e vírgula faltando:

1 program exemplo;
2 procedure proc1(var i, j : int)
3 begin
4 a:=i+j
5 end;
6 begin
7 proc1(1,2)
8 end.
9
10

Choose File

No file chosen

Confirmar

Error	Line	Column
expected ';'	2	31

Trata até dois caracteres inválidos antes do ponto e vírgula:

1 program exemplo;
2 procedure proc1(var i, j : int) # \$;
3 begin
4 a:=i+j
5 end;
6 begin
7 proc1(1,2)
8 end.
9
10

Choose File

No file chosen

Confirmar

Error	Line	Column
unexpected '# \$' before ';'	2	33

<FORMAL_PARAMETERS>

Corresponde a união dos erros para a <formal_parameters_section> e alguns tratamentos relacionados ao ponto e vírgula e os parenteses:

1 program exemplo;
2 procedure proc1 var i, j : int);
3 begin
4 a:=i+j
5 end;
6 begin
7 proc1(1,2)
8 end.
9
10

Choose File

No file chosen

Confirmar

Error	Line	Column
expected '('	2	17

Erro para fecha parenteses:

1 program exemplo;
2 procedure proc1 (var i, j : int;
3 begin
4 a:=i+j
5 end;
6 begin
7 proc1(1,2)
8 end.
9
10

Choose File

No file chosen

Confirmar

Error	Line	Column
expected ')'	2	32

Obs: quando tanto abre parenteses quanto fecha parenteses estão faltando, um erro é devolvido.

Podem ocorrer caracteres inválidos tanto antes do abre parenteses quanto do fecha parenteses e também antes do ponto e vírgula de separação dos membros:


```
1 program exemplo;
2 procedure proc1 $$ (var i, j : int $$; is_valid: boolean $$);
3 begin
4     a:=i+j
5 end;
6 begin
7     proc1(1,2)
8 end.
9
10
```

Choose File No file chosen

Confirmar

Error	Line	Column
unexpected '\$\$' before '('	2	17
unexpected '\$\$' before ';'	2	35
unexpected '\$\$' before ')'	2	57

<FORMAL_PARAMETER_SECTION>

Corresponde aos união com os erros para <identifier_list> bem como tratamento se algum caracter inválido foi passado antes da palavra reservada “var” ou do primeiro identificador:

1 program exemplo;
2 procedure proc1 (\$11 var i, j : int);
3 begin
4 a:=i+j
5 end;
6 begin
7 proc1(1,2)
8 end.
9
10

Choose File

No file chosen

Confirmar

Error	Line	Column
unexpected '\$11'	2	18

Também permite o tratamento da falta dos dois pontos informando o tipo dos parâmetros:

1 program exemplo;
2 procedure proc1 (var i, j int);
3 begin
4 a:=i+j
5 end;
6 begin
7 proc1(1,2)
8 end.
9
10

Choose File

No file chosen

Confirmar

Error	Line	Column
':' expected	2	28

Também realiza o tratamento de até dois caracteres errados antes dos dois pontos ou antes do tipo dos parâmetros:

```
1 program exemplo;
2 procedure proc1 (var i, j ##:## int);
3 begin
4     a:=i+j
5 end;
6 begin
7     proc1(1,2)
8 end.
9
10
```

Choose File No file chosen

Confirmar

Error	Line	Column
unexpected '##' before ':'	2	27
unexpected '##' before type	2	30

E finalmente, informa caso se tenha esquecido o tipo dos parâmetros formais:

```
1 program exemplo;
2 procedure proc1 (var i, j:);
3 begin
4     a:=i+j
5 end;
6 begin
7     proc1(1,2)
8 end.
9
10
```

Choose File No file chosen

Confirmar

Error	Line	Column
type expected	2	27

O analisador também consegue tratar erros maiores descartando parte do programa, porém não consegue dar informações de erro detalhadas nesse caso, como por exemplo:

```
1 program exemplo;  
2 begin  
3     while;  
4     read(a)  
5 end.  
6  
7
```

Choose File No file chosen

Confirmar

Error	Line	Column
expression expected after while keyword	3	7

```
1 program exemplo;  
2 begin  
3     if;  
4     read(a)  
5 end.  
6  
7
```

Choose File No file chosen

Confirmar

Error	Line	Column
expression expected after if keyword	3	4

Análise Semântica

A Análise Semântica se baseia em tokens validados pela análise sintática. A cada execução de um método da análise sintática, métodos da análise semântica são chamados para validar semanticamente a expressão atual da linguagem.

O seguinte programa dispararia um erro indicando que a variável a,b e c não foram utilizadas, pois a variável a de proc foi utilizada, mas b e c de proc não foram (e a do escopo global também não). Além disso, proc não foi utilizado, portanto procedure não declarado é disparado:

```

1 program correto;
2 int a, b, c;
3 procedure proc(var a1 : int);
4 int a, b, c;
5 begin
6   a:=1
7 end;
8 begin
9   b:=4
10 end.

```

No programa abaixo, o erro de variável não inicializada seria disparado.

```

1 program prog;
2 int a, b;
3 begin
4   a:= b
5 end.
6

```

No programa abaixo, um erro de tipo seria disparado, pois o valor esperado era inteiro, mas recebeu um valor real

```

1 program prog;
2 int a;
3 real b;
4 begin
5   b:= 10
6   a:= b
7 end.
8

```

No exemplo abaixo, haverá um erro pois o tipo esperado para o operador and é boolean, mas o tipo usado foi real

```
1 program prog;  
2 int a, b, c;  
3 begin  
4   a:= 10  
5   c:= 20  
6   b:= a and c  
7 end.  
8
```