CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

DANIEL DE PAULA BRAGA LOPES
GUILHERME FERNANDES MARCHEZINI
LAURO CÉSAR JACQUES SANTOS

Relatório Analisador Léxico e Tabela de Símbolos

Belo Horizonte
2017

1. Forma de Utilização

A execução do compilador é feito via terminal do Linux utilizando o .jar que está localizado na pasta dist passando o nome do .jar e o arquivo de entrada que será compilado, da seguinte forma:

```
java -jar compilador.jar <nome_aquivo>
```

Abaixo um exemplo de como compilar o primeiro código de teste disponibilizado na especificação do trabalho:

```
java -jar Compilador.jar ../test/teste1.tst
```

2. Implementação

Abaixo uma breve explicação das classes existentes no compilador:

2.1 Lexer

Classe que implementa o analisador léxico. Seu construtor insere as palavras reservadas na tabela de símbolos. Possui um método **scan** que devolve um Token.

2.2 LexicalException

Classe para imprimir na tela o motivo de ocorrer uma determinada exceção. Existem três casos:
- Token inválido: É passado um token inválido ou não esperado;
- Fim de arquivo inesperado: O arquivo termina quando ainda deveria possuir alguma informação;
- Default: Ocorre quando é um erro diferente dos dois anteriores.

2.3 Num

Classe para representar um Token número.

2.4 Tag

Classe que define as constantes para os tokens.

2.5 Token

Representa um Token genérico. Contém a constante que representa o Token.

2.6 Word

Representa um token de palavras reservadas, identificadores e tokens compostos como != e &&.

3. Testes

A seguir os testes e seus respectivos resultados:

3.1 Teste 1

Código:

```
init
         a, b, c, result is integer;

         read (a);
         read (c);
         b := 10;
         result := (a * c)/(b + 5 - 345);
         write(result);
stop
```

Resultado:

```
<INIT, init>
<ID, a>
<,>
<ID, b>
<,>
<ID, c>
<,>
<ID, result>
<IS, is>
<T_INTEGER, integer>
<;>
<READ, read>
<(>
<ID, a>
<)>
<;>
<READ, read>
<(>
<ID, c>
<)>
<;>
<ID, b>
<ATRIB, :=>
<NUM, 10>
<;>
<ID, result>
<ATRIB, :=>
<(>
<ID, a>
<*>
<ID, c>
<)>
</>
<(>
<ID, b>
<+>
<NUM, 5>
<->
<NUM, 345>
<)>
<;>
<WRITE, write>
<(>
<ID, result>
<)>
<;>
<EOF>

Tabela de Símbolos:
init = <INIT, init>
a = <ID, a>
b = <ID, b>
read = <READ, read>
or = <OR, or>
c = <ID, c>
string = <T_STRING, string>
is = <IS, is>
integer = <T_INTEGER, integer>
do = <DO, do>
while = <WHILE, while>
result = <ID, result>
not = <NOT, not>
stop = <STOP, stop>
else = <ELSE, else>
and = <AND, and>
end = <END, end>
if = <IF, if>
begin = <BEGIN, begin>
write = <WRITE, write>
```

3.2 Teste 2

Código:

```
        a, _valor, b : integer;

init
         read (a);
         b := a * a;
         write (b);
         b = b + a/2 * (a + 5);
         Write (b);
stop
```

Resultado:

```
<ID, a>
<,>
<_>
<ID, valor>
<,>
<ID, b>
<:>
<T_INTEGER, integer>
<;>
<INIT, init>
<READ, read>
<(>
<ID, a>
<)>
<;>
<ID, b>
<ATRIB, :=>
<ID, a>
<*>
<ID, a>
<;>
<WRITE, write>
<(>
<ID, b>
<)>
<;>
<ID, b>
<=>
<ID, b>
<+>
<ID, a>
</>
<NUM, 2>
<*>
<(>
<ID, a>
<+>
<NUM, 5>
<)>
<;>
<WRITE, write>
<(>
<ID, b>
<)>
<;>
<STOP, stop>
<EOF>

Tabela de Símbolos:
init = <INIT, init>
a = <ID, a>
b = <ID, b>
read = <READ, read>
or = <OR, or>
string = <T_STRING, string>
valor = <ID, valor>
is = <IS, is>
integer = <T_INTEGER, integer>
do = <DO, do>
while = <WHILE, while>
not = <NOT, not>
stop = <STOP, stop>
else = <ELSE, else>
and = <AND, and>
end = <END, end>
if = <IF, if>
begin = <BEGIN, begin>
write = <WRITE, write>
```

3.3 Teste 3

3.3.1 Primeira execução

Código:

```
{ Programa de Teste
Calculo de idade
init
         cont_ is int;
         media, idade, soma_ is integer;
         begin
                 cont_ = 5;
                 soma = 0;
                 do

                         write(“Altura:” );
                         read (altura);
                         soma := soma altura;
                         cont_ := cont_ - 1;

                while(cont_ > 0)
                write(“Media: ”);
                write (soma / qtd);
stop
```

Resultado:

```
<EOF>

Tabela de Símbolos:
init = <INIT, init>
read = <READ, read>
or = <OR, or>
string = <T_STRING, string>
is = <IS, is>
integer = <T_INTEGER, integer>
do = <DO, do>
while = <WHILE, while>
not = <NOT, not>
stop = <STOP, stop>
else = <ELSE, else>
and = <AND, and>
end = <END, end>
if = <IF, if>
begin = <BEGIN, begin>
write = <WRITE, write>
```

3.3.2 Segunda execução

Código:

```
{ Programa de Teste
Calculo de idade }
init
         cont_ is int;
         media, idade, soma_ is integer;
         begin
                 cont_ = 5;
                 soma = 0;
                 do

                         write("Altura: ");
                         read (altura);
                         soma := soma altura;
                         cont_ := cont_ - 1;

                while(cont_ > 0)
                write("Media: ");
                write (soma / qtd);
stop
```

Resultado:

```
<INIT, init>
<ID, cont_>
<IS, is>
<ID, int>
<;>
<ID, media>
<,>
<ID, idade>
<,>
<ID, soma_>
<IS, is>
<T_INTEGER, integer>
<;>
<BEGIN, begin>
<ID, cont_>
<=>
<NUM, 5>
<;>
<ID, soma>
<=>
<NUM, 0>
<;>
<DO, do>
<WRITE, write>
<(>
<STRING, Altura: >
<)>
<;>
<READ, read>
<(>
<ID, altura>
<)>
<;>
<ID, soma>
<ATRIB, :=>
<ID, soma>
<ID, altura>
<;>
<ID, cont_>
<ATRIB, :=>
<ID, cont_>
<->
<NUM, 1>
<;>
<WHILE, while>
<(>
<ID, cont_>
<>>
<NUM, 0>
<)>
<WRITE, write>
<(>
<STRING, Media: >
<)>
<;>
<WRITE, write>
<(>
<ID, soma>
</>
<ID, qtd>
<)>
<;>
<STOP, stop>
<EOF>

Tabela de Símbolos:
qtd = <ID, qtd>
init = <INIT, init>
read = <READ, read>
or = <OR, or>
string = <T_STRING, string>
soma = <ID, soma>
soma_ = <ID, soma_>
is = <IS, is>
cont_ = <ID, cont_>
integer = <T_INTEGER, integer>
do = <DO, do>
media = <ID, media>
while = <WHILE, while>
int = <ID, int>
idade = <ID, idade>
not = <NOT, not>
stop = <STOP, stop>
altura = <ID, altura>
else = <ELSE, else>
and = <AND, and>
end = <END, end>
if = <IF, if>
begin = <BEGIN, begin>
write = <WRITE, write>
```

3.4 Teste 4

3.4.1 Primeira execução

Código:

```
init

        i, j, k, @total, 1soma is integer

                read (I);
                k := i * (5-i * 50 / 10;
                j := i * 10;
                k := i* j / k;
                k := 4 + a $;
                write(i);
                write(j);
                write(k);
```

Resultado:

```
<INIT, init>
<ID, i>
<,>
<ID, j>
<,>
<ID, k>
<,>
Erro na linha 3: Token inválido '@'
<ID, total>
<,>
<NUM, 1>
<ID, soma>
<IS, is>
<T_INTEGER, integer>
<READ, read>
<(>
<ID, i>
<)>
<;>
<ID, k>
<ATRIB, :=>
<ID, i>
<*>
<(>
<NUM, 5>
<->
<ID, i>
<*>
<NUM, 50>
</>
<NUM, 10>
<;>
<ID, j>
<ATRIB, :=>
<ID, i>
<*>
<NUM, 10>
<;>
<ID, k>
<ATRIB, :=>
<ID, i>
<*>
<ID, j>
</>
<ID, k>
<;>
<ID, k>
<ATRIB, :=>
<NUM, 4>
<+>
<ID, a>
Erro na linha 9: Token inválido '$'
<;>
<WRITE, write>
<(>
<ID, i>
<)>
<;>
<WRITE, write>
<(>
<ID, j>
<)>
<;>
<WRITE, write>
<(>
<ID, k>
<)>
<;>
<EOF>

Tabela de Símbolos:
init = <INIT, init>
a = <ID, a>
read = <READ, read>
or = <OR, or>
string = <T_STRING, string>
soma = <ID, soma>
i = <ID, i>
is = <IS, is>
j = <ID, j>
integer = <T_INTEGER, integer>
do = <DO, do>
k = <ID, k>
while = <WHILE, while>
not = <NOT, not>
total = <ID, total>
stop = <STOP, stop>
else = <ELSE, else>
and = <AND, and>
end = <END, end>
if = <IF, if>
begin = <BEGIN, begin>
write = <WRITE, write>
```

3.4.2 Segunda execução

Código:

```
init

        i, j, k, total, 1soma is integer

                read (I);
                k := i * (5-i * 50 / 10;
                j := i * 10;
                k := i* j / k;
                k := 4 + a $;
                write(i);
                write(j);
                write(k);
```

Resultado:

```
<INIT, init>
<ID, i>
<,>
<ID, j>
<,>
<ID, k>
<,>
<ID, total>
<,>
<NUM, 1>
<ID, soma>
<IS, is>
<T_INTEGER, integer>
<READ, read>
<(>
<ID, i>
<)>
<;>
<ID, k>
<ATRIB, :=>
<ID, i>
<*>
<(>
<NUM, 5>
<->
<ID, i>
<*>
<NUM, 50>
</>
<NUM, 10>
<;>
<ID, j>
<ATRIB, :=>
<ID, i>
<*>
<NUM, 10>
<;>
<ID, k>
<ATRIB, :=>
<ID, i>
<*>
<ID, j>
</>
<ID, k>
<;>
<ID, k>
<ATRIB, :=>
<NUM, 4>
<+>
<ID, a>
Erro na linha 9: Token inválido '$'
<;>
<WRITE, write>
<(>
<ID, i>
<)>
<;>
<WRITE, write>
<(>
<ID, j>
<)>
<;>
<WRITE, write>
<(>
<ID, k>
<)>
<;>
<EOF>

Tabela de Símbolos:
init = <INIT, init>
a = <ID, a>
read = <READ, read>
or = <OR, or>
string = <T_STRING, string>
soma = <ID, soma>
i = <ID, i>
is = <IS, is>
j = <ID, j>
integer = <T_INTEGER, integer>
do = <DO, do>
k = <ID, k>
while = <WHILE, while>
not = <NOT, not>
total = <ID, total>
stop = <STOP, stop>
else = <ELSE, else>
and = <AND, and>
end = <END, end>
if = <IF, if>
begin = <BEGIN, begin>
write = <WRITE, write>
```

3.4.3 Terceira execução

Código:

```
init

        i, j, k, total, 1soma is integer

                read (I);
                k := i * (5-i * 50 / 10;
                j := i * 10;
                k := i* j / k;
                k := 4 + a ;
                write(i);
                write(j);
                write(k);
```

Resultado:

```
<INIT, init>
<ID, i>
<,>
<ID, j>
<,>
<ID, k>
<,>
<ID, total>
<,>
<NUM, 1>
<ID, soma>
<IS, is>
<T_INTEGER, integer>
<READ, read>
<(>
<ID, i>
<)>
<;>
<ID, k>
<ATRIB, :=>
<ID, i>
<*>
<(>
<NUM, 5>
<->
<ID, i>
<*>
<NUM, 50>
</>
<NUM, 10>
<;>
<ID, j>
<ATRIB, :=>
<ID, i>
<*>
<NUM, 10>
<;>
<ID, k>
<ATRIB, :=>
<ID, i>
<*>
<ID, j>
</>
<ID, k>
<;>
<ID, k>
<ATRIB, :=>
<NUM, 4>
<+>
<ID, a>
<;>
<WRITE, write>
<(>
<ID, i>
<)>
<;>
<WRITE, write>
<(>
<ID, j>
<)>
<;>
<WRITE, write>
<(>
<ID, k>
<)>
<;>
<EOF>

Tabela de Símbolos:
init = <INIT, init>
a = <ID, a>
read = <READ, read>
or = <OR, or>
string = <T_STRING, string>
soma = <ID, soma>
i = <ID, i>
is = <IS, is>
j = <ID, j>
integer = <T_INTEGER, integer>
do = <DO, do>
k = <ID, k>
while = <WHILE, while>
not = <NOT, not>
total = <ID, total>
stop = <STOP, stop>
else = <ELSE, else>
and = <AND, and>
end = <END, end>
if = <IF, if>
begin = <BEGIN, begin>
write = <WRITE, write>
```

3.5 Teste 5

3.5.1 Primeira execução

Código:

```
init
// Programa com if

                j, k, m is integer;
                a, j is string;

        read(j);
        read(k);

        if (j == “ok”)
        begin
                result = k/m
        end
        else
        begin
                result := 0;
                write (“Invalid entry”);
        end

        write(result);
```

Resultado:

```
<INIT, init>
<ID, j>
<,>
<ID, k>
<,>
<ID, m>
<IS, is>
<T_INTEGER, integer>
<;>
<ID, a>
<,>
<ID, j>
<IS, is>
<T_STRING, string>
<;>
<READ, read>
<(>
<ID, j>
<)>
<;>
<READ, read>
<(>
<ID, k>
<)>
<;>
<IF, if>
<(>
<ID, j>
<=>
<=>
Erro na linha 11: Token inválido '“'
<ID, ok>
Erro na linha 11: Token inválido '”'
<)>
<BEGIN, begin>
<ID, result>
<=>
<ID, k>
</>
<ID, m>
<END, end>
<ELSE, else>
<BEGIN, begin>
<ID, result>
<ATRIB, :=>
<NUM, 0>
<;>
<WRITE, write>
<(>
Erro na linha 18: Token inválido '“'
<ID, Invalid>
<ID, entry>
Erro na linha 18: Token inválido '”'
<)>
<;>
<END, end>
<WRITE, write>
<(>
<ID, result>
<)>
<;>
<EOF>

Tabela de Símbolos:
init = <INIT, init>
a = <ID, a>
read = <READ, read>
or = <OR, or>
string = <T_STRING, string>
is = <IS, is>
j = <ID, j>
integer = <T_INTEGER, integer>
do = <DO, do>
k = <ID, k>
while = <WHILE, while>
m = <ID, m>
result = <ID, result>
entry = <ID, entry>
not = <NOT, not>
stop = <STOP, stop>
else = <ELSE, else>
and = <AND, and>
invalid = <ID, Invalid>
end = <END, end>
ok = <ID, ok>
if = <IF, if>
begin = <BEGIN, begin>
write = <WRITE, write>
```

3.5.2 Segunda execução

Código:

```
init
// Programa com if

                j, k, m is integer;
                a, j is string;


        read(j);
        read(k);

        if (j == "ok")
        begin
                result = k/m
        end
        else
        begin
                result := 0;
                write (“Invalid entry”);
        end


        write(result);
```

Resultado:

```
<INIT, init>
<ID, j>
<,>
<ID, k>
<,>
<ID, m>
<IS, is>
<T_INTEGER, integer>
<;>
<ID, a>
<,>
<ID, j>
<IS, is>
<T_STRING, string>
<;>
<READ, read>
<(>
<ID, j>
<)>
<;>
<READ, read>
<(>
<ID, k>
<)>
<;>
<IF, if>
<(>
<ID, j>
<=>
<=>
<STRING, ok>
<)>
<BEGIN, begin>
<ID, result>
<=>
<ID, k>
</>
<ID, m>
<END, end>
<ELSE, else>
<BEGIN, begin>
<ID, result>
<ATRIB, :=>
<NUM, 0>
<;>
<WRITE, write>
<(>
Erro na linha 18: Token inválido '“'
<ID, Invalid>
<ID, entry>
Erro na linha 18: Token inválido '”'
<)>
<;>
<END, end>
<WRITE, write>
<(>
<ID, result>
<)>
<;>
<EOF>

Tabela de Símbolos:
init = <INIT, init>
a = <ID, a>
read = <READ, read>
or = <OR, or>
string = <T_STRING, string>
is = <IS, is>
j = <ID, j>
integer = <T_INTEGER, integer>
do = <DO, do>
k = <ID, k>
while = <WHILE, while>
m = <ID, m>
result = <ID, result>
entry = <ID, entry>
not = <NOT, not>
stop = <STOP, stop>
else = <ELSE, else>
and = <AND, and>
invalid = <ID, Invalid>
end = <END, end>
if = <IF, if>
begin = <BEGIN, begin>
write = <WRITE, write>
```

3.5.3 Terceira execução

Código:

```
init
// Programa com if

                 j, k, m is integer;
                 a, j is string;


        read(j);
        read(k);

        if (j == "ok")
        begin
                result = k/m
        end
        else
        begin
                result := 0;
                write ("Invalid entry");
        end


        write(result);
```

Resultado:

```
<INIT, init>
<ID, j>
<,>
<ID, k>
<,>
<ID, m>
<IS, is>
<T_INTEGER, integer>
<;>
<ID, a>
<,>
<ID, j>
<IS, is>
<T_STRING, string>
<;>
<READ, read>
<(>
<ID, j>
<)>
<;>
<READ, read>
<(>
<ID, k>
<)>
<;>
<IF, if>
<(>
<ID, j>
<=>
<=>
<STRING, ok>
<)>
<BEGIN, begin>
<ID, result>
<=>
<ID, k>
</>
<ID, m>
<END, end>
<ELSE, else>
<BEGIN, begin>
<ID, result>
<ATRIB, :=>
<NUM, 0>
<;>
<WRITE, write>
<(>
<STRING, Invalid entry>
<)>
<;>
<END, end>
<WRITE, write>
<(>
<ID, result>
<)>
<;>
<EOF>

Tabela de Símbolos:
init = <INIT, init>
a = <ID, a>
read = <READ, read>
or = <OR, or>
string = <T_STRING, string>
is = <IS, is>
j = <ID, j>
integer = <T_INTEGER, integer>
do = <DO, do>
k = <ID, k>
while = <WHILE, while>
m = <ID, m>
result = <ID, result>
not = <NOT, not>
stop = <STOP, stop>
else = <ELSE, else>
and = <AND, and>
end = <END, end>
if = <IF, if>
begin = <BEGIN, begin>
write = <WRITE, write>
```

3.6 Teste 6

3.6.1 Primeira execução

Código:

```
init
        a, b, c, maior is integer;
                read(a);
                read(b);
                read(c;
                maior := 0;
                if ( a>b and a>c )
                        maior := a;
                else
                        if (b>c)
                                maior := b;
                        else
                                maior := c;
                write(“Maior idade: ”);
                write(maior);
end
```

Resultado:

```
<INIT, init>
<ID, a>
<,>
<ID, b>
<,>
<ID, c>
<,>
<ID, maior>
<IS, is>
<T_INTEGER, integer>
<;>
<READ, read>
<(>
<ID, a>
<)>
<;>
<READ, read>
<(>
<ID, b>
<)>
<;>
<READ, read>
<(>
<ID, c>
<;>
<ID, maior>
<ATRIB, :=>
<NUM, 0>
<;>
<IF, if>
<(>
<ID, a>
<>>
<ID, b>
<AND, and>
<ID, a>
<>>
<ID, c>
<)>
<ID, maior>
<ATRIB, :=>
<ID, a>
<;>
<ELSE, else>
<IF, if>
<(>
<ID, b>
<>>
<ID, c>
<)>
<ID, maior>
<ATRIB, :=>
<ID, b>
<;>
<ELSE, else>
<ID, maior>
<ATRIB, :=>
<ID, c>
<;>
<WRITE, write>
<(>
Erro na linha 20: Token inválido '“'
<ID, maior>
<ID, idade>
<:>
Erro na linha 20: Token inválido '”'
<)>
<;>
<WRITE, write>
<(>
<ID, maior>
<)>
<;>
<EOF>

Tabela de Símbolos:
init = <INIT, init>
a = <ID, a>
b = <ID, b>
read = <READ, read>
or = <OR, or>
c = <ID, c>
string = <T_STRING, string>
is = <IS, is>
integer = <T_INTEGER, integer>
do = <DO, do>
while = <WHILE, while>
idade = <ID, idade>
not = <NOT, not>
stop = <STOP, stop>
maior = <ID, maior>
else = <ELSE, else>
and = <AND, and>
end = <END, end>
if = <IF, if>
begin = <BEGIN, begin>
write = <WRITE, write>
```

3.6.2 Segunda execução

Código:

```
init
        a, b, c, maior is integer;

                read(a);
                read(b);
                read(c;

                maior := 0;
                if ( a>b and a>c )
                        maior := a;

                else
                        if (b>c)
                                maior := b;

                        else
                                maior := c;
                write("Maior idade: ");
                write(maior);
end
```

Resultado:

```
<INIT, init>
<ID, a>
<,>
<ID, b>
<,>
<ID, c>
<,>
<ID, maior>
<IS, is>
<T_INTEGER, integer>
<;>
<READ, read>
<(>
<ID, a>
<)>
<;>
<READ, read>
<(>
<ID, b>
<)>
<;>
<READ, read>
<(>
<ID, c>
<;>
<ID, maior>
<ATRIB, :=>
<NUM, 0>
<;>
<IF, if>
<(>
<ID, a>
<>>
<ID, b>
<AND, and>
<ID, a>
<>>
<ID, c>
<)>
<ID, maior>
<ATRIB, :=>
<ID, a>
<;>
<ELSE, else>
<IF, if>
<(>
<ID, b>
<>>
<ID, c>
<)>
<ID, maior>
<ATRIB, :=>
<ID, b>
<;>
<ELSE, else>
<ID, maior>
<ATRIB, :=>
<ID, c>
<;>
<WRITE, write>
<(>
<STRING, Maior idade: >
<)>
<;>
<WRITE, write>
<(>
<ID, maior>
<)>
<;>
<EOF>

Tabela de Símbolos:
init = <INIT, init>
a = <ID, a>
b = <ID, b>
read = <READ, read>
or = <OR, or>
c = <ID, c>
string = <T_STRING, string>
is = <IS, is>
integer = <T_INTEGER, integer>
do = <DO, do>
while = <WHILE, while>
not = <NOT, not>
stop = <STOP, stop>
maior = <ID, maior>
else = <ELSE, else>
and = <AND, and>
end = <END, end>
if = <IF, if>
begin = <BEGIN, begin>
write = <WRITE, write>
```

3.7 Teste 7

Código:

```
init
        a is int;

        begin
                read (A);

                DO
                        A := A - 2
                WHiLE (A >= 2);

                iF (a = 0)
                        write (A);
                        write (" é par");
                ELSE
                        write (A);
                        write (" é ímpar.");
```

Resultado:

```
<INIT, init>
<ID, a>
<IS, is>
<ID, int>
<;>
<BEGIN, begin>
<READ, read>
<(>
<ID, a>
<)>
<;>
<DO, do>
<ID, a>
<ATRIB, :=>
<ID, a>
<->
<NUM, 2>
<WHILE, while>
<(>
<ID, a>
<GTE, >=>
<NUM, 2>
<)>
<;>
<IF, if>
<(>
<ID, a>
<=>
<NUM, 0>
<)>
<WRITE, write>
<(>
<ID, a>
<)>
<;>
<WRITE, write>
<(>
<STRING,  é par>
<)>
<;>
<ELSE, else>
<WRITE, write>
<(>
<ID, a>
<)>
<;>
<WRITE, write>
<(>
<STRING,  é ímpar.>
<)>
<;>
<EOF>

Tabela de Símbolos:
init = <INIT, init>
a = <ID, a>
read = <READ, read>
or = <OR, or>
string = <T_STRING, string>
is = <IS, is>
integer = <T_INTEGER, integer>
do = <DO, do>
while = <WHILE, while>
int = <ID, int>
not = <NOT, not>
stop = <STOP, stop>
else = <ELSE, else>
and = <AND, and>
end = <END, end>
if = <IF, if>
begin = <BEGIN, begin>
write = <WRITE, write>
```

3.8 Teste 8

3.8.1 Primeira execução

Código:

```
init
        n is integer;
        anterior, proximo, aux, i is integer;

        begin
                write ("Digite a posicao: );
                read (n);

                if ( n == 1)
                        proximo := 0;
                else
                        if ( n == 2)
                                proximo := 1;
                        else
                                anterior := 0;
                                proximo := 1
                                i := 3;
                                do
                                        aux := proximo;
                                        proximo := anterior + proximo;
                                        anterior := aux;
                                        i := i + 1;
                                while (i < n)

        write ("O termo: ");
                write (proximo);
stop
```

Resultado:

```
<INIT, init>
<ID, n>
<IS, is>
<T_INTEGER, integer>
<;>
<ID, anterior>
<,>
<ID, proximo>
<,>
<ID, aux>
<,>
<ID, i>
<IS, is>
<T_INTEGER, integer>
<;>
<BEGIN, begin>
<WRITE, write>
<(>
Erro na linha 6: Literal não terminado
<READ, read>
<(>
<ID, n>
<)>
<IF, if>
<(>
<ID, n>
<=>
<=>
<NUM, 1>
<)>
<ID, proximo>
<ATRIB, :=>
<NUM, 0>
<;>
<ELSE, else>
<IF, if>
<(>
<ID, n>
<=>
<=>
<NUM, 2>
<)>
<ID, proximo>
<ATRIB, :=>
<NUM, 1>
<;>
<ELSE, else>
<ID, anterior>
<ATRIB, :=>
<NUM, 0>
<;>
<ID, proximo>
<ATRIB, :=>
<NUM, 1>
<;>
<ID, i>
<ATRIB, :=>
<NUM, 3>
<;>
<DO, do>
<ID, aux>
<ATRIB, :=>
<ID, proximo>
<;>
<ID, proximo>
<ATRIB, :=>
<ID, anterior>
<+>
<ID, proximo>
<;>
<ID, anterior>
<ATRIB, :=>
<ID, aux>
<;>
<ID, i>
<ATRIB, :=>
<ID, i>
<+>
<NUM, 1>
<;>
<WHILE, while>
<(>
<ID, i>
<<>
<ID, n>
<)>
<WRITE, write>
<(>
<STRING, O termo: >
<)>
<;>
<WRITE, write>
<(>
<ID, proximo>
<)>
<;>
<STOP, stop>
<EOF>

Tabela de Símbolos:
init = <INIT, init>
read = <READ, read>
or = <OR, or>
string = <T_STRING, string>
aux = <ID, aux>
i = <ID, i>
is = <IS, is>
integer = <T_INTEGER, integer>
do = <DO, do>
while = <WHILE, while>
n = <ID, n>
anterior = <ID, anterior>
not = <NOT, not>
stop = <STOP, stop>
else = <ELSE, else>
and = <AND, and>
proximo = <ID, proximo>
end = <END, end>
if = <IF, if>
begin = <BEGIN, begin>
write = <WRITE, write>
```

3.8.2 Segunda execução

Código:

```
init
        n is integer;
        anterior, proximo, aux, i is integer;

        begin
                write ("Digite a posicao: ");
                read (n);

                if ( n == 1)
                        proximo := 0;
                else
                        if ( n == 2)
                                proximo := 1;
                        else
                                anterior := 0;
                                proximo := 1
                                i := 3;
                                do
                                        aux := proximo;
                                        proximo := anterior + proximo;
                                        anterior := aux;
                                        i := i + 1;
                                while (i < n)

        write ("O termo: ");
                write (proximo);
stop
```

Resultado:

```
<INIT, init>
<ID, n>
<IS, is>
<T_INTEGER, integer>
<;>
<ID, anterior>
<,>
<ID, proximo>
<,>
<ID, aux>
<,>
<ID, i>
<IS, is>
<T_INTEGER, integer>
<;>
<BEGIN, begin>
<WRITE, write>
<(>
<STRING, Digite a posicao: >
<)>
<;>
<READ, read>
<(>
<ID, n>
<)>
<;>
<IF, if>
<(>
<ID, n>
<=>
<=>
<NUM, 1>
<)>
<ID, proximo>
<ATRIB, :=>
<NUM, 0>
<;>
<ELSE, else>
<IF, if>
<(>
<ID, n>
<=>
<=>
<NUM, 2>
<)>
<ID, proximo>
<ATRIB, :=>
<NUM, 1>
<;>
<ELSE, else>
<ID, anterior>
<ATRIB, :=>
<NUM, 0>
<;>
<ID, proximo>
<ATRIB, :=>
<NUM, 1>
<ID, i>
<ATRIB, :=>
<NUM, 3>
<;>
<DO, do>
<ID, aux>
<ATRIB, :=>
<ID, proximo>
<;>
<ID, proximo>
<ATRIB, :=>
<ID, anterior>
<+>
<ID, proximo>
<;>
<ID, anterior>
<ATRIB, :=>
<ID, aux>
<;>
<ID, i>
<ATRIB, :=>
<ID, i>
<+>
<NUM, 1>
<;>
<WHILE, while>
<(>
<ID, i>
<<>
<ID, n>
<)>
<WRITE, write>
<(>
<STRING, O termo: >
<)>
<;>
<WRITE, write>
<(>
<ID, proximo>
<)>
<;>
<STOP, stop>
<EOF>

Tabela de Símbolos:
init = <INIT, init>
read = <READ, read>
or = <OR, or>
string = <T_STRING, string>
aux = <ID, aux>
i = <ID, i>
is = <IS, is>
integer = <T_INTEGER, integer>
do = <DO, do>
while = <WHILE, while>
n = <ID, n>
anterior = <ID, anterior>
not = <NOT, not>
stop = <STOP, stop>
else = <ELSE, else>
and = <AND, and>
proximo = <ID, proximo>
end = <END, end>
if = <IF, if>
begin = <BEGIN, begin>
write = <WRITE, write>
```