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

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
```

```
<INIT, init>
                                                                    <+>
<ID, a>
                                                                    <NUM, 5>
<,>
                                                                    <->
<ÍD, b>
                                                                    <NUM, 345>
                                                                    <)>
<,>
<ÍD, c>
                                                                    <;>
                                                                    <WRITE, write>
<ID, result>
                                                                    <(>
                                                                    result>
<IS, is>
<T_INTEGER, integer>
                                                                    <)>
                                                                    <;>
<READ, read>
                                                                    <ÉOF>
<(>
<ID, a>
                                                                    Tabela de Símbolos:
                                                                    init = <INIT, init>
<)>
                                                                    a = \langle ID, a \rangle
<;>
<READ, read>
                                                                   b = \langle ID, b \rangle
                                                                   read = <READ, read>
<(>
<ID, c>
                                                                   or = \langle OR, or \rangle
                                                                   c = \langle ID, c \rangle
<)>
                                                                   string = <T_STRING, string>
is = <IS, is>
<;>
<ID, b>
<ATRIB, :=>
                                                                    integer = <T_INTEGER, integer>
<NUM, 10>
                                                                    do = \langle DO, do \rangle
                                                                   while = <WHILE, while>
<ID, result>
                                                                   result = <ID, result>
                                                                    not = <NOT, not>
<ATRIB, :=>
                                                                    stop = <STOP, stop>
<(>
<ÌD, a>
                                                                   else = <ELSE, else>
<*>
                                                                    and = <AND, and>
<ID, c>
                                                                    end = <END, end>
                                                                    if = <IF, if>
<)>
                                                                    begin = <BEGIN, begin>
</>
<(>
                                                                   write = <WRITE, write>
<ÌD, b>
```

3.2 Teste 2

Código:

```
<ID, a>
<,>
<ID, valor>
<,>
<ID, b>
<:>
<T_INTEGER, integer>
<;>
<INIT, init> <READ, read>
<(>
<ÌD, 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>
<E0F>
Tabela de Símbolos:
init = <INIT, init>
a = <ID, a>
b = <ID, b>
read = <READ, read>
or = \langle OR, or \rangle
string = <T_STRING, string>
valor = <ID, valor>
is = \langle IS, is \rangle
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:

```
<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:

<WHILE, while>

```
<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>
```

```
<ID, cont_>
<>>
<NUM, 0>
<)>
<WRITE, write>
<STRING, Media: >
<)>
<;>
<WRITE, write>
<(>
<ID, soma>
</>
<ID, qtd>
<)>
<;>
<STOP, stop>
<E0F>
Tabela de Símbolos:
qtd = <ID, qtd>
init = <INIT, init>
read = <READ, read>
or = \langle OR, or \rangle
string = <T_STRING, string>
soma = <ID, soma>
soma_ = <ID, soma_>
is = <IS, is>
cont_ = <ID, cont_>
integer = <T_INTEGER, integer>
do = \langle DO, do \rangle
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);
```

```
<INIT, init>
                                                                        <ATRIB, :=>
<ID, i>
                                                                        <NUM, 4>
<,>
                                                                        <+>
<ID, j>
                                                                        <ID, a>
                                                                        Erro na linha 9: Token inválido '$'
<,>
<ID, k>
                                                                        <WRITE, write>
<,>
Erro na linha 3: Token inválido '@'
                                                                        <(>
<ID, total>
                                                                        <İD, i>
<,>
                                                                        <)>
<NUM, 1>
                                                                        <;>
<ID, soma>
<IS, is>
<T_INTEGER, integer>
                                                                        <WRITE, write>
                                                                        <(>
                                                                        <ÌD, j>
<READ, read>
                                                                        <)>
<(>
                                                                        <;>
<ÌD, i>
                                                                        <WRITE, write>
<)>
                                                                        <(>
<;>
<ID, k>
                                                                        <ID, k>
                                                                        <)>
<ATRIB, :=>
                                                                        <;>
<ID, i>
                                                                        <EOF>
<*>
<(>
                                                                        Tabela de Símbolos:
<NUM, 5>
                                                                        init = <INIT, init>
                                                                        a = \langle ID, a \rangle
<->
<ID, i>
                                                                        read = <READ, read>
                                                                        or = <OR, or>
string = <T_STRING, string>
<*>
<NUM, 50>
                                                                        soma = <ID, soma>
</>
<NUM, 10>
                                                                        i = \langle ID, i \rangle
                                                                        is = \langle IS, is \rangle
<ID, j>
                                                                        j = \langle ID, j \rangle
                                                                        integer = <T_INTEGER, integer>
do = <DO, do>
<ATRIB, :=>
<ID, i>
                                                                        k = \langle ID, k \rangle
<*>
<NUM, 10>
                                                                        while = <WHILE, while>
                                                                        not = <NOT, not>
<;>
<ID, k>
                                                                        total = <ID, total>
<ATRIB, :=>
                                                                        stop = <STOP, stop>
else = <ELSE, else>
<ID, i>
<*>
                                                                        and = \langle AND, and \rangle
<ID, j>
                                                                        end = <END, end>
                                                                        if = <IF, if>
</>
<ID, k>
                                                                        begin = <BEGIN, begin>
<;>
                                                                        write = <WRITE, write>
<ID, k>
```

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);
```

```
<INIT, init>
                                                                       <ATRIB, :=>
                                                                      <NUM, 4>
<ID, i>
<,>
                                                                       <+>
<ID, j>
                                                                       <ID, a>
                                                                      Erro na linha 9: Token inválido '$'
<,>
<ID, k>
<,>
                                                                      <WRITE, write>
<ID, total>
                                                                      <(>
<,>
                                                                       <İD, i>
<NUM, 1>
                                                                      <)>
<ID, soma>
                                                                       <;>
                                                                       <WRITE, write>
<IS, is>
<T_INTEGER, integer> <READ, read>
                                                                      <(>
                                                                      <ÌD, j>
<(>
                                                                       <)>
<ID, i>
                                                                      <;>
                                                                      <WRITE, write>
<)>
<;>
                                                                       <(>
<ID, k>
                                                                      <ID, k>
<ATRIB, :=>
                                                                      <)>
<ID, i>
                                                                       <;>
<*>
                                                                       <EOF>
<(>
<NUM, 5>
                                                                      Tabela de Símbolos:
                                                                      init = <INIT, init>
<->
<ID, i>
                                                                      a = \langle ID, a \rangle
<*>
                                                                      read = <READ, read>
                                                                      or = <OR, or>
string = <T_STRING, string>
<NUM, 50>
</>
<NUM, 10>
                                                                      soma = <ID, soma>
                                                                      i = \langle ID, i \rangle
<;>
<ÍD, j>
                                                                      is = <IS, is>
                                                                      j = <ID, j>
integer = <T_INTEGER, integer>
do = <DO, do>
<ATRIB, :=>
<ID, i>
<*>
<NUM, 10>
                                                                      k = \langle ID, k \rangle
                                                                      while = <WHILE, while>
<;>
                                                                      not = <NOT, not>
<ÍD, k>
<ATRIB, :=>
                                                                      total = <ID, total>
                                                                      stop = <STOP, stop>
else = <ELSE, else>
<ID, i>
<*>
<ID, j>
                                                                      and = \langle AND, and \rangle
                                                                      end = <END, end>
</>
                                                                      if = <IF, if>
<ID, k>
<;>
                                                                      begin = <BEGIN, begin>
                                                                      write = <WRITE, write>
<ID, k>
```

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:

<ID, k>

```
<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>
<;>
```

```
<ATRIB, :=>
<NUM, 4>
<+>
<ID, a>
<;>
<WRITE, write>
<(>
<ÎD, i>
<)>
<;>
<WRITE, write>
<(>
<ÌD, j>
<)>
<;>
<WRITE, write>
<(>
<ID, k>
<)>
<;>
<EOF>
Tabela de Símbolos:
init = <INIT, init>
a = <ID, a>
read = <READ, read>
or = <0R, or>
string = <T_STRING, string>
soma = <ID, soma>
i = \langle ID, i \rangle
is = <IS, is>
j = <ID, j>
integer = <T_INTEGER, integer>
do = <DO, do>
k = \langle ID, k \rangle
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, init>
<ID, j>
<,>
<ID, k>
<,>
<ID, m>
<IS, is>
<T_INTEGER, integer>
<;>
<ID, a>
<,>
<ÍD, j>
<IS, is>
<T_STRING, string>
<READ, read>
<(>
<ÌD, 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>
<)>
<;>
<E0F>
Tabela de Símbolos:
init = <INIT, init>
a = \langle ID, a \rangle
read = <READ, read>
or = <OR, or>
string = <T_STRING, string>
is = \langle IS, is \rangle
j = <ID, j>
integer = <T_INTEGER, integer>
do = <DO, do>
k = <ID, k>
while = <WHILE, while>
m = \langle ID, m \rangle
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:

<ATRIB, :=>

```
<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>
<ELSÉ, else>
<BEGIN, begin>
<ID, result>
```

```
<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 = \langle ID, a \rangle
read = <READ, read>
or = <OR, or>
string = <T_STRING, string>
is = <IS, is>
j = <ID, j>
integer = <T_INTEGER, integer>
do = <DO, do>
k = \langle ID, k \rangle
while = <WHILE, while>
m = \langle ID, m \rangle
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:

<ELSE, else>

```
<INIT, init>
<ID, j>
<,>
<ID, k>
<,>
<ÍD, m>
<IS, is>
<T_INTEGER, integer>
<;>
<ID, a>
<,>
<ID, j>
<IS, is>
<T_STRING, string>
<;>
<READ, read>
<(>
<İD, j>
<)>
<;>
<READ, read>
<(>
<ÌD, k>
<)>
<;>
<IF, if>
<(>
<ID, j>
<=>
<=>
<STRING, ok>
<)>
<BEGIN, begin>
<ID, result>
<=>
<ID, k>
</>
<ID, m>
<END, end>
```

```
<BEGIN, begin>
<ID, result>
<ATRIB, :=>
<NUM, 0>
<;>
<WRITE, write>
<STRING, Invalid entry>
<)>
<;>
<END, end>
<WRITE, write>
<(>
\int iD, result>
<)>
<;>
<EOF>
Tabela de Símbolos:
init = <INIT, init>
a = <ID, a>
read = <READ, read>
or = \langle OR, or \rangle
string = <T_STRING, string>
is = \langle IS, is \rangle
j = <ID, j>
integer = <T_INTEGER, integer>
do = <DO, do>
k = <ID, k>
while = <WHILE, while>
m = \langle ID, m \rangle
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:

Resultado:

<ID, c>

```
<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>
<(>
<ÌD, c>
<;>
<ID, maior>
<ATRIB, :=>
<NUM, 0>
<;>
<IF, if>
<(>
<ÌD, a>
<>>
<ID, b>
<AND, and>
<ID, a>
<>>
<ID, c>
<)>
<ID, maior>
<ATRIB, :=>
<ID, a>
<;>
<ELSE, else>
<IF, if>
<(>
<ID, b>
<>>
```

```
<)>
<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>
<(>
<ld><ID, maior>
<)>
<;>
<E0F>
Tabela de Símbolos:
init = <INIT, init>
a = \langle ID, a \rangle
b = \langle ID, b \rangle
read = <READ, read>
or = \langle OR, or \rangle
c = \langle ID, c \rangle
string = <T_STRING, string>
is = <IS, is>
integer = <T_INTEGER, integer>
do = \langle DO, do \rangle
while = <WHILE, while>
idade = <ID, idade>
not = <NOT, not>
stop = <STOP, stop>
maior = <ID, maior>
else = <ELSÉ, else>
and = <AND, and>
end = <END, end>
if = <IF, if>
begin = <BEGIN, begin>
write = <WRITE, write>
```

3.6.2 Segunda execução

Código:

Resultado:

<(>

```
<INIT, init>
<ID, a>
                                                               <>>
<,>
<ID, b>
                                                               <)>
<,>
<ID, c>
<,>
<ID, maior>
                                                               <;>
<IS, is>
<T_INTEGER, integer>
<READ, read>
<(>
                                                               <;>
<ÌD, a>
<)>
<;>
<READ, read>
                                                               <)>
<(>
                                                               <;>
<ID, b>
<)>
                                                               <(>
<;>
<READ, read>
                                                               <)>
<(>
                                                               <;>
<ID, c>
                                                               <EOF>
<;>
<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>
Tabela de Símbolos:
init = <INIT, init>
a = <ID, a>
b = \langle ID, b \rangle
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 = \langle NOT, not \rangle
stop = <STOP, stop>
maior = <ID, maior>
else = <ELSÉ, else>
and = \langle AND, and \rangle
end = <END, end>
if = <IF, if>
begin = <BEGIN, begin>
write = <WRITE, write>
```

3.7 Teste 7

Código:

```
<INIT, init>
<ID, a>
<IS, is> <ID, int>
<;>
<BEGIN, begin>
<READ, read>
<(>
<ID, a>
<)>
<;>
<D0, do>
<ID, a>
<ATRIB, :=>
<ID, a>
<->
<NUM, 2>
<WHILE, while>
<(>
<ÌD, 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 impar.>
<)>
<;>
<EOF>
Tabela de Símbolos:
init = <INIT, init>
a = <ID, a>
read = <READ, read>
or = \langle OR, or \rangle
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
```

<init, init=""></init,>	<atrib, :=""></atrib,>	
	-	
<id, n=""></id,>	<id, proximo=""></id,>	
<is, is=""></is,>	<;>	
<t_integer, integer=""></t_integer,>	<id, proximo=""></id,>	
<pre><;> <td antenion;<="" pre=""></td><td><atrib, :=""></atrib,></td></pre>		<atrib, :=""></atrib,>
<id, anterior=""></id,>	<id, anterior=""></id,>	
<,>	<+>	
<id, proximo=""></id,>	<id, proximo=""></id,>	
<,>	<;>	
<id, aux=""></id,>	<id, anterior=""></id,>	
<,>	<atrib, :=""></atrib,>	
<id, i=""></id,>	<id, aux=""></id,>	
<is, is=""></is,>	<;>	
<t_integer, integer=""></t_integer,>	<id, i=""></id,>	
<;>	<atrib, :=""></atrib,>	
<begin, begin=""></begin,>	<id, i=""></id,>	
<write, write=""></write,>	<+>	
<(>	<num, 1=""></num,>	
Erro na linha 6: Literal não terminado	<;>	
<read, read=""></read,>	<while, while=""></while,>	
<(>	<(>	
<id, n=""></id,>	<id, i=""></id,>	
<)>	<<>	
<if, if=""></if,>	<id, n=""></id,>	
<(>	<)>	
<id, n=""></id,>	<write, write=""></write,>	
<=>	<(>	
<=>	<string, o="" termo:=""></string,>	
<num, 1=""></num,>	<)>	
<)>	<;>	
<id, proximo=""></id,>	<write, write=""></write,>	
<atrib, :=""></atrib,>	<(>	
<num, 0=""></num,>	<id, proximo=""></id,>	
<;>	<)>	
<else, else=""></else,>	<;>	
<if, if=""></if,>	<stop, stop=""></stop,>	
<(>	<eof></eof>	
<id, n=""></id,>		
<=>	Tabela de Símbolos:	
<=>	init = <init, init=""></init,>	
<num, 2=""></num,>	read = <read, read=""></read,>	
<)>	or = <or, or=""></or,>	
<id, proximo=""></id,>	string = <t_string, string=""></t_string,>	
<atrib, :=""></atrib,>	aux = <id, aux=""></id,>	
<num, 1=""></num,>	i = <id, i=""></id,>	
⟨;⟩	is = <is, is=""></is,>	
<else, else=""></else,>	<pre>integer = <t_integer, integer=""></t_integer,></pre>	
<id, anterior=""></id,>	do = <do, do=""></do,>	
<atrib, :=""></atrib,>	while = <while, while=""></while,>	
<num, 0=""></num,>	n = <id, n=""></id,>	
<;>	<pre>anterior = <id, anterior=""></id,></pre>	
<id, proximo=""></id,>	not = <not, not=""></not,>	
<atrib, :=""></atrib,>	stop = <stop, stop=""></stop,>	
<num, 1=""></num,>	else = <else, else=""></else,>	
<;>	and = <and, and=""></and,>	
<id, i=""></id,>	<pre>proximo = <id, proximo=""></id,></pre>	
<atrib, :=""></atrib,>	end = <end, end=""></end,>	
<num, 3=""></num,>	if = <if, if=""></if,>	
<;> `	begin = <begin, begin=""></begin,>	
<d0, do=""></d0,>	write = <write, write=""></write,>	
<id, aux=""></id,>	· ·	

3.8.2 Segunda execução

Código:

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

<init, init=""></init,>	<id, aux=""></id,>	
	•	
<id, n=""></id,>	<atrib, :=""></atrib,>	
<is, is=""></is,>	<id, proximo=""></id,>	
<t_integer, integer=""></t_integer,>	<;>	
<pre><;> <td antenion;<="" pre=""></td><td><id, proximo=""></id,></td></pre>		<id, proximo=""></id,>
<id, anterior=""></id,>	<atrib, :=""></atrib,>	
<,> .TD manufact	<id, anterior=""></id,>	
<id, proximo=""></id,>	<+>	
<,>	<id, proximo=""></id,>	
<id, aux=""></id,>	<;>	
<,>	<id, anterior=""></id,>	
<id, i=""></id,>	<atrib, :=""></atrib,>	
<is, is=""></is,>	<id, aux=""></id,>	
<t_integer, integer=""></t_integer,>	<;>	
<;>	<id, i=""></id,>	
<begin, begin=""></begin,>	<atrib, :=""></atrib,>	
<pre><write, write=""></write,></pre>	<id, i=""></id,>	
<(>	<+>	
<string, a="" digite="" posicao:=""></string,>	<num, 1=""></num,>	
<)>	<;>	
<;>	<while, while=""></while,>	
<read, read=""></read,>	<(>	
< <u>(</u> >	<id, i=""></id,>	
<id, n=""></id,>	<<>	
<)>	<id, n=""></id,>	
<;>	<)>	
<if, if=""></if,>	<write, write=""></write,>	
<(>	<(>	
<id, n=""></id,>	<string, o="" termo:=""></string,>	
<=>	<)>	
<=>	<;>	
<num, 1=""></num,>	<write, write=""></write,>	
<)>	<(>	
<id, proximo=""></id,>	<id, proximo=""></id,>	
<atrib, :=""></atrib,>	<)>	
<num, 0=""></num,>	<;>	
<;>	<stop, stop=""></stop,>	
<else, else=""></else,>	<eof></eof>	
<if, if=""></if,>	Tabala da Cémbalas	
<(>	Tabela de Símbolos:	
<id, n=""></id,>	init = <init, init=""></init,>	
<=>	read = <read, read=""></read,>	
<=>	or = <or, or=""></or,>	
<num, 2=""></num,>	string = <t_string, string=""></t_string,>	
<)>	aux = <id, aux=""></id,>	
<id, proximo=""></id,>	i = <id, i=""></id,>	
<atrib, :=""></atrib,>	is = <is, is=""></is,>	
<num, 1=""></num,>	<pre>integer = <t_integer, integer=""></t_integer,></pre>	
<;>	do = <do, do=""></do,>	
<else, else=""></else,>	while = <while, while=""></while,>	
<id, anterior=""></id,>	n = <id, n=""></id,>	
<atrib, :=""></atrib,>	anterior = <id, anterior=""></id,>	
<num, 0=""></num,>	not = <not, not=""></not,>	
<;> <td nnovimo=""></td> <td>stop = <stop, stop=""></stop,></td>		stop = <stop, stop=""></stop,>
<id, proximo=""></id,>	else = <else, else=""></else,>	
<atrib, :=""></atrib,>	and = <and, and=""></and,>	
<num, 1=""></num,>	proximo = <id, proximo=""></id,>	
<id, i=""></id,>	end = <end, end=""></end,>	
<atrib, :=""></atrib,>	if = <if, if=""></if,>	
<num, 3=""></num,>	begin = <begin, begin=""></begin,>	
<;> <do, do=""></do,>	write = <write, write=""></write,>	
\bu_, \u0/		