

**Centro Federal de Educação Tecnológica de Minas Gerais**

**Compiladores - Trabalho Prático**

**Parte 2 - Analisador Sintático e Gramática**

Membros: Juliano Martins

Mariana Athayde

Thais Diniz Braz

Belo Horizonte / MG  
 2015

**Implementação**

Nessa segunda parte do trabalho, foram implementados o Analisador Sintático a partir da nova gramática gerada, depois de retirada as recursões.

O objetivo do analisador sintático é identificar se a sequência de tokens é válida segunda a gramática ou não. O analisador sintático vai pegando um a um os token gerados pelo analisador léxico e verificando se as sequências de tokens são válidas segundo a gramática especificada.

Como nós fizemos uma modificação nos valores passados dos tokens para que os mesmos não coincidam com os valores da tabela ASCII, iremos representar brevemente aqui como ficou. Para essa implementação, foi adotado a mesma forma de token do trabalho passado:

par<nome, valor>

Entretanto, houve uma reestruturação dos valores dos tokens como mostrado abaixo:

|  |  |
| --- | --- |
| **Valor** | **Tipo do Token** |
| 257 | DCL: Palavra reservada **declare** |
| 258 | STRT: Palavra reservada **start** |
| 259 | END: Palavra reservada **end** |
| 260 | INT: Palavra reservada **int** |
| 261 | STRG: Palavra reservada **string** |
| 262 | IF: Palavra reservada **if** |
| 263 | THEN: Palavra reservada **then** |
| 264 | ELSE: Palavra reservada **else** |
| 265 | DO: Palavra reservada **do** |
| 266 | WHL: Palavra reservada **while** |
| 267 | READ: Palavra reservada **read** |
| 268 | WRT: Palavra reservada **write** |
| 269 | OR: Palavra reservada **or** |
| 270 | AND: Palavra reservada **and** |
| 271 | EQ: Símbolo de igual (**==**) |
| 272 | GT: Símbolo de maior (**>**) |
| 273 | GE: Símbolo de maior ou igual (**>=**) |
| 274 | LT: Símbolo de menor (**<**) |
| 275 | LE: Símbolo de maior ou igual (**<=**) |
| 276 | NE: Simbolo de diferente (**<>**) |
| 277 | SUM: Sinal de soma (**+**) |
| 278 | MIN: Sinal de subtração (**-**) |
| 279 | MUL: Sinal de multiplicação (**\***) |
| 280 | DIV: Símbolo de divisão (**/**) |
| 281 | PVG: Ponto e vírgula (**;**) |
| 282 | VG: Vírgula (**,**) |
| 283 | DTPS: Dois pontos(**:**) |
| 284 | AP: Abre parênteses (**(**) |
| 285 | FP: Fecha parênteses (**)**) |
| 286 | ATBR: Símbolo de atribuição (**=**) |
| 287 | LIT: Representa um literal |
| 288 | NUM: Representa um número |
| 289 | ID: Representa um identificador |
| 290 | NEG: |
| 291 | ASP: |
| 292 | NESP: Não especificado |

A implementação deste Análisador Sintático foi baseado na gramática mostrada abaixo:

program ::= [ **declare** decl-list] **start** stmt-list **end**

decl-list ::= decl ";" { decl ";"}

decl ::= ident-list “:” type

ident-list ::= identifier {"," identifier}

type ::= **int** | **string**

stmt-list ::= stmt ";" { stmt ";"}

stmt ::= assign-stmt | if-stmt | do-stmt

| read-stmt | write-stmt

assign-stmt ::= identifier "=" simple\_expr

if-stmt ::= **if** condition **then** stmt-list **end**

| **if** condition **then** stmt-list **else** stmt-list **end**

condition ::= expression

do-stmt ::= **do** stmt-list stmt-suffix

stmt-suffix ::= **while** condition

read-stmt ::= **read** "(" identifier ")"

write-stmt ::= **write** "(" writable ")"

writable ::= simple-expr

expression ::= simple-expr | simple-expr relop simple-expr

simple-expr ::= term | simple-expr addop term

term ::= factor-a | term mulop factor-a

fator-a ::= factor | ! factor | "-" factor

factor ::= identifier | constant | "(" expression ")"

relop ::= "==" | ">" | ">=" | "<" | "<=" | "<>"

addop ::= "+" | "-" | **or**

mulop ::= "\*" | "/" | **and**

constant ::= integer\_const | literal

integer\_const ::= digit {digit}

literal ::= " **“**" {caractere} "**”**"

identifier ::= letter {letter | digit }

letter ::= [A-Za-z]

digit ::= [0-9]

caractere ::= *um dos 256 caracteres do conjunto ASCII, exceto* ” e

quebra de linha

E apartir dessa gramática foi construída a seguinte tabela FIRST/FOLLOW:

|  |  |  |
| --- | --- | --- |
|  | **First** | **Follow** |
| **Program** | declare, start | $ |
| **Decl-list** | id, lambda | start |
| **Decl** | id | ; |
| **Ident-list** | id, lambda | : |
| **Type** | int, string | ; |
| **Stmt-list** | id, do, read, write, if, lambda | end, else |
| **Stmt** | id, do, read, write, if | ; |
| **Assign-stmt** | id | ; |
| **If-stmt\_A** | if | ; |
| **If-stmt\_B** | end, else | ; |
| **Condition** | “, int, (, not, - | then |
| **Do-stmt** | do | ; |
| **Stmt-suffix** | while | ; |
| **Read-stmt** | read | ; |
| **Write-stmt** | write | ; |
| **Writeable** | “, int, id, (, not, - | ) |
| **Expression\_A** | “, int, id, (, not, - | then, ) |
| **Expression\_B** | <, <=, >, >=, <>, = = | then, ) |
| **Simple-expr\_A** | “, int, id, (, not, - | then, ), ==, >, >=, <, <=, <> |
| **Simple-expr\_B** | +, -, or, lambda | then, ), ==, >, >=, <, <=, <> |
| **Term1** | “, int, id, (, not, - | or,+, -,then, ), ==, >, >=, <, <=, <> |
| **Term2** | \*, /, and, lambda | or,+, -,then, ), ==, >, >=, <, <=, <> |
| **Factor-a** | “, int, id, (, not, - | \*, /, and, +, -, or, then, ), ==, >, >=, <, <=, <> |
| **Factor** | “, int, id, ( | \*, /, and, +, -, or, then, ), ==, >, >=, <, <=, <> |
| **Relop** | <, = =, <=, >, >=,<> | “, int, id, not, -, ( |
| **Addop** | +, -, or | “, int, id, not, -, ( |
| **Mulop** | \*, /, and | “, int, id, not, -, ( |
| **Constant** | “, int | \*, /, and, +, -, or, then, ), ==, >, >=, <, <=, <> |

Os principais aspectos a serem ressaltados em nosso código são:

* **A classe Lexer.java**

Esta classe lê o arquivo de entrada, caractere a caractere, e retorna os tokens encontrados.

* **A classe parser.java**

Que recebe os token gerados pela classe Lexer um a um e vai validando ou não as sequências de tokens geradas.

Além dos pontos acima, as demais classes servem de auxílio às principais: A classe TAG.java apenas declara e instância os valores inteiros que representam qual o tipo do token utilizado, a classe Token.java representa os tokens que serão reconhecidos, a classe Word extende a classe Token e refere-se ao identificadores, assim como a classe Num que refere-se aos números.

**Forma de uso do compilador**

Para utilizar o compilador deve-se chamar o arquivo compilador.jar, passando como parâmetro os arquivos que serão compilados. Lembrando que nessa primeira parte do trabalho, compilar consiste apenas em imprimir os Tokens reconhecidos (ou os erros para os não reconhecidos) e as entradas da Tabela de Símbolos.

**Exemplo:**

O comando ‘java -jar compilador.jar teste1’ compila o arquivo teste1.

**Testes**

Nesta etapa foram realizados os testes no analisador léxico, para tal foram considerados 8 arquivos, dentre eles 7 contendo erros léxicos. Abaixo estarão as evidências de execução de cada um dos testes antes e depois da correção dos erros, é importante ressaltar que aqui não são corrigidos os outros erros que não dizem respeito ao analisador léxico (erro sintático/semântico).

**Arquivo de Teste 1**

Esse arquivo contém um erro léxico que consiste em um caracter não contido no alfabeto da gramática.

|  |  |
| --- | --- |
| **Programa: teste1**  program teste1  declare  a, b, c: int;  result: int  start  read (a);  read (c);  b = 10;  result = (a \* c)/(b + 5 - 345);  write(result);  teste1 = 1;  end | <),34>  <;,30>  <b,41>  <=,35>  <10,40>  <;,30>  <result,41>  <=,35>  <(,33>  <a,41>  <\*,28>  <c,41>  <),34>  </,29>  <(,33>  <b,41>  <+,26>  <5,40>  <-,27>  <345,40>  <),34>  <;,30>  <write,41>  <(,33>  <result,41>  <),34>  <;,30>  <teste1,41>  <=,35>  <1,40>  <;,30>  <end,41>  <------ TABELA DE SÍMBOLOS ------->  result  a  b  c  teste1  program  CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo) |
| **Resultado do Teste 1:**  <program,41>  <teste1,41>  <declare,41>  <a,41>  <,,31>  <b,41>  <,,31>  <c,41>  <:,32>  <int,41>  <;,30>  <result,41>  <:,32>  <int,41>  <start,41>  <read,41>  <(,33>  <a,41>  <),34>  <;,30>  <read,41>  <(,33>  <c,41> |

**Arquivo de Teste 2**

Esse arquivo contém um erro léxico que consiste em um caracter não contido no alfabeto da gramática.

|  |  |
| --- | --- |
| **Programa: teste2**  declare  a: int;  b: int;  nome: string;  start  read (a);  read (nome);  b = a \* a;  b = b + a/2 \* (3 + 5);  write(nome);  write(b);  end. | **Programa sem erros léxicos: teste2\_Corrigido**  declare  a: int;  b: int;  nome: string;  start  read (a);  read (nome);  b = a \* a;  b = b + a/2 \* (3 + 5);  write(nome);  write(b);  end |
| **Resultado do Teste 2:**  <declare,41>  <a,41>  <:,32>  <int,41>  <;,30>  <b,41>  <:,32>  <int,41>  <;,30>  <nome,41>  <:,32>  <string,41>  <;,30>  <start,41>  <read,41>  <(,33>  <a,41>  <),34>  <;,30>  <read,41>  <(,33>  <nome,41>  <),34>  <;,30>  <b,41>  <=,35>  <a,41>  <\*,28>  <a,41>  <;,30>  <b,41>  <=,35>  <b,41>  <+,26>  <a,41>  </,29>  <2,40>  <\*,28>  <(,33>  <3,40>  <+,26>  <5,40>  <),34>  <;,30>  <write,41>  <(,33>  <nome,41>  <),34>  <;,30>  <write,41>  <(,33>  <b,41>  <),34>  <;,30>  <end,41>  ERRO - Caractere não especificado na linha 12 ch: ".".  <.,42>  <------ TABELA DE SÍMBOLOS ------->  a  b  nome  CONSTRUÍDO COM SUCESSO (tempo total: 2 segundos) | **Resultado do Teste 2 Corrigido:**  <declare,41>  <a,41>  <:,32>  <int,41>  <;,30>  <b,41>  <:,32>  <int,41>  <;,30>  <nome,41>  <:,32>  <string,41>  <;,30>  <start,41>  <read,41>  <(,33>  <a,41>  <),34>  <;,30>  <read,41>  <(,33>  <nome,41>  <),34>  <;,30>  <b,41>  <=,35>  <a,41>  <\*,28>  <a,41>  <;,30>  <b,41>  <=,35>  <b,41>  <+,26>  <a,41>  </,29>  <2,40>  <\*,28>  <(,33>  <3,40>  <+,26>  <5,40>  <),34>  <;,30>  <write,41>  <(,33>  <nome,41>  <),34>  <;,30>  <write,41>  <(,33>  <b,41>  <),34>  <;,30>  <end,41>  <------ TABELA DE SÍMBOLOS ------->  a  b  nome  CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo) |

**Arquivo de Teste 3**

Esse arquivo contém um erro léxico que consiste em um caracter não contido no alfabeto da gramática.

|  |  |
| --- | --- |
| **Programa: teste3**  declare  cont: int;  media, altura, soma: int;  start  cont = 5;  soma = 0;  do  write(“Altura: ”);  read (altura);  soma = soma # altura;  cont = cont - 1;  while(cont);  write(“Media: ”);  write (soma / qtd);  end | **Programa sem erros léxicos: teste3\_Corrigido**  declare  cont: int;  media, altura, soma: int;  start  cont = 5;  soma = 0;  do  write(“Altura: ”);  read (altura);  soma = soma \* altura;  cont = cont - 1;  while(cont);  write(“Media: ”);  write (soma / qtd);  end |
| **Resultado do Teste 3:**  <declare,41>  <cont,41>  <:,32>  <int,41>  <;,30>  <media,41>  <,,31>  <altura,41>  <,,31>  <soma,41>  <:,32>  <int,41>  <;,30>  <start,41>  <cont,41>  <=,35>  <5,40>  <;,30>  <soma,41>  <=,35>  <0,40>  <;,30>  <do,41>  <write,41>  <(,33>  <“Altura: ”,36>  <),34>  <;,30>  <read,41>  <(,33>  <altura,41>  <),34>  <;,30>  <soma,41>  <=,35>  <soma,41>  ERRO - Caractere não especificado na linha 10 ch: "#".  <#,42>  <altura,41>  <;,30>  <cont,41>  <=,35>  <cont,41>  <-,27>  <1,40>  <;,30>  <while,41>  <(,33>  <cont,41>  <),34>  <;,30>  <write,41>  <(,33>  <“Media: ”,36>  <),34>  <;,30>  <write,41>  <(,33>  <soma,41>  </,29>  <qtd,41>  <),34>  <;,30>  <end,41>  <------ TABELA DE SÍMBOLOS ------->  qtd  altura  soma  media  cont | **Resultado do Teste 3 Corrigido:**  <declare,41>  <cont,41>  <:,32>  <int,41>  <;,30>  <media,41>  <,,31>  <altura,41>  <,,31>  <soma,41>  <:,32>  <int,41>  <;,30>  <start,41>  <cont,41>  <=,35>  <5,40>  <;,30>  <soma,41>  <=,35>  <0,40>  <;,30>  <do,41>  <write,41>  <(,33>  <“Altura: ”,36>  <),34>  <;,30>  <read,41>  <(,33>  <altura,41>  <),34>  <;,30>  <soma,41>  <=,35>  <soma,41>  <\*,28>  <altura,41>  <;,30>  <cont,41>  <=,35>  <cont,41>  <-,27>  <1,40>  <;,30>  <while,41>  <(,33>  <cont,41>  <),34>  <;,30>  <write,41>  <(,33>  <“Media: ”,36>  <),34>  <;,30>  <write,41>  <(,33>  <soma,41>  </,29>  <qtd,41>  <),34>  <;,30>  <------ TABELA DE SÍMBOLOS ------->  qtd  altura  soma  media  cont  CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos) |

**Arquivo de Teste 4**

Esse arquivo contém três erros léxicos que consiste em identificador mal formado (inicalizados de um númer) e caracteres nao especificados no alfabeto da gramática.

|  |  |
| --- | --- |
| **Programa: teste4**  declare  i, j, k, @total, 1soma: int;  start  i = 4 \* (5-3 \* 50 / 10;  j = i \* 10;  k = i\* j / k;  k = 4 + a $;  write(i);  write(j);  write(k);  end | **Programa sem erros léxicos: teste4\_Corrigido**  declare  i, j, k, total, soma1: int;  start  i = 4 \* (5-3 \* 50 / 10;  j = i \* 10;  k = i\* j / k;  k = 4 + a;  write(i);  write(j);  write(k);  end |
| **Resultado do Teste 4:**  <declare,41>  <i,41>  <,,31>  <j,41>  <,,31>  <k,41>  <,,31>  ERRO - Caractere não especificado na linha 2 ch: "@".  <@,42>  <total,41>  <,,31>  Erro! Identificador incorreto na linha 2  <1,40>  <soma,41>  <:,32>  <int,41>  <;,30>  <start,41>  <i,41>  <=,35>  <4,40>  <\*,28>  <(,33>  <5,40>  <-,27>  <3,40>  <\*,28>  <50,40>  </,29>  <10,40>  <;,30>  <j,41>  <=,35>  <i,41>  <\*,28>  <10,40>  <;,30>  <k,41>  <=,35>  <i,41>  <\*,28>  <j,41>  </,29>  <k,41>  <;,30>  <k,41>  <=,35>  <4,40>  <+,26>  <a,41>  ERRO - Caractere não especificado na linha 7 ch: "$".  <$,42>  <;,30>  <write,41>  <(,33>  <i,41>  <),34>  <;,30>  <write,41>  <(,33>  <j,41>  <),34>  <;,30>  <write,41>  <(,33>  <k,41>  <),34>  <;,30>  <end,41>  <------ TABELA DE SÍMBOLOS ------->  a  total  soma  i  j  k  CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo) | **Resultado do Teste 4 Corrigido:**  <declare,41>  <i,41>  <,,31>  <j,41>  <,,31>  <k,41>  <,,31>  <total,41>  <,,31>  <soma1,41>  <:,32>  <int,41>  <;,30>  <start,41>  <i,41>  <=,35>  <4,40>  <\*,28>  <(,33>  <5,40>  <-,27>  <3,40>  <\*,28>  <50,40>  </,29>  <10,40>  <;,30>  <j,41>  <=,35>  <i,41>  <\*,28>  <10,40>  <;,30>  <k,41>  <=,35>  <i,41>  <\*,28>  <j,41>  </,29>  <k,41>  <;,30>  <k,41>  <=,35>  <4,40>  <+,26>  <a,41>  <;,30>  <write,41>  <(,33>  <i,41>  <),34>  <;,30>  <write,41>  <(,33>  <j,41>  <),34>  <;,30>  <write,41>  <(,33>  <k,41>  <),34>  <;,30>  <end,41>  <------ TABELA DE SÍMBOLOS ------->  a  total  i  soma1  j  k  CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo) |

**Arquivo de Teste 5**

Esse arquivo contém um erro onde o comentário é iniciado entretanto, não é encerrado.

|  |  |
| --- | --- |
| **Programa: teste5**  declare  /\*Este e um programa com if  j, k: int;  a, j: real;  start  read(j);  read(k);  if (k != 0)  result = j/k  else  result = 0  end;  write(res);  end | **Programa sem erros léxicos: teste5\_Corrigido**  declare  /\*Este e um programa com\*/ if  j, k: int;  a, j: real;  start  read(j);  read(k);  if (k == 0)  result = j/k  else  result = 0  end;  write(res);  end |
| **Resultado do Teste 5:**  <declare,41>  WARNING! Comentário não encerrado!  <------ TABELA DE SÍMBOLOS ------->  CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo) | **Resultado do Teste 5 Corrigido:**  <declare,41>  <if,41>  <j,41>  <,,31>  <k,41>  <:,32>  <int,41>  <;,30>  <a,41>  <,,31>  <j,41>  <:,32>  <real,41>  <;,30>  <start,41>  <read,41>  <(,33>  <j,41>  <),34>  <;,30>  <read,41>  <(,33>  <k,41>  <),34>  <;,30>  <if,41>  <(,33>  <k,41>  <==,20>  <0,40>  <),34>  <result,41>  <=,35>  <j,41>  </,29>  <k,41>  <else,41>  <result,41>  <=,35>  <0,40>  <end,41>  <;,30>  <write,41>  <(,33>  <res,41>  <),34>  <;,30>  <end,41>  <------ TABELA DE SÍMBOLOS ------->  result  a  res  j  k  real |

**Arquivo de Teste 6**

Esse arquivo contém dois erros léxicos: possui identificador maior do que o tamanho máximo permitido e extrapola o limite de tamanho do tipo inteiro.

|  |  |
| --- | --- |
| **Programa: teste6**  declare  // Um programa para calcular a maior idade  a, b, c, maior: int;  umavariavelcomonomemuitogrande: string;  start  read(a);  read(b);  read(c);  maior = 12345678901234567890123456789;  if ( a>b and a>c ) then  maior = a;  else  if (b>c) then  maior = b;  else  maior = c;  end  end  write(“Maior idade: ”);  write(maior);  end  end | **Programa sem erros léxicos: teste6\_Corrigido**  declare  // Um programa para calcular a maior idade  a, b, c, maior: int;  variavel: string;  start  read(a);  read(b);  read(c);  maior = 123456;  if ( a>b and a>c ) then  maior = a;  else  if (b>c) then  maior = b;  else  maior = c;  end  end  write(“Maior idade: ”);  write(maior);  end  end |
| **Resultado do Teste 6:**  <declare,41>  <a,41>  <,,31>  <b,41>  <,,31>  <c,41>  <,,31>  <maior,41>  <:,32>  <int,41>  <;,30>  ERRO! Identificador excedeu o número permitido de caracteres!  <umavariavelcomonomem,41>  <uitogrande,41>  <:,32>  <string,41>  <;,30>  <start,41>  <read,41>  <(,33>  <a,41>  <),34>  <;,30>  <read,41>  <(,33>  <b,41>  <),34>  <;,30>  <read,41>  <(,33>  <c,41>  <),34>  <;,30>  <maior,41>  <=,35>  Erro! Valor de inteiro maior que o maximo permitido  <12345678901234567890123456789,40>  <;,30>  <if,41>  <(,33>  <a,41>  <>,21>  <b,41>  <and,41>  <a,41>  <>,21>  <c,41>  <),34>  <then,41>  <maior,41>  <=,35>  <a,41>  <;,30>  <else,41>  <if,41>  <(,33>  <b,41>  <>,21>  <c,41>  <),34>  <then,41>  <maior,41>  <=,35>  <b,41>  <;,30>  <else,41>  <maior,41>  <=,35>  <c,41>  <;,30>  <end,41>  <end,41>  <write,41>  <(,33>  <“Maior idade: ”,36>  <),34>  <;,30>  <write,41>  <(,33>  <maior,41>  <),34>  <;,30>  <end,41>  <end,41>  <------ TABELA DE SÍMBOLOS ------->  a  b  c  maior  umavariavelcomonomem  uitogrande  CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo) | **Resultado do Teste 6 Corrigido:**  <declare,41>  <a,41>  <,,31>  <b,41>  <,,31>  <c,41>  <,,31>  <maior,41>  <:,32>  <int,41>  <;,30>  <variavel,41>  <:,32>  <string,41>  <;,30>  <start,41>  <read,41>  <(,33>  <a,41>  <),34>  <;,30>  <read,41>  <(,33>  <b,41>  <),34>  <;,30>  <read,41>  <(,33>  <c,41>  <),34>  <;,30>  <maior,41>  <=,35>  <123456,40>  <;,30>  <if,41>  <(,33>  <a,41>  <>,21>  <b,41>  <and,41>  <a,41>  <>,21>  <c,41>  <),34>  <then,41>  <maior,41>  <=,35>  <a,41>  <;,30>  <else,41>  <if,41>  <(,33>  <b,41>  <>,21>  <c,41>  <),34>  <then,41>  <maior,41>  <=,35>  <b,41>  <;,30>  <else,41>  <maior,41>  <=,35>  <c,41>  <;,30>  <end,41>  <end,41>  <write,41>  <(,33>  <“Maior idade: ”,36>  <),34>  <;,30>  <write,41>  <(,33>  <maior,41>  <),34>  <;,30>  <end,41>  <end,41>  <------ TABELA DE SÍMBOLOS ------->  a  b  c  maior  variavel  CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo) |

**Arquivo de Teste 7**

Esse arquivo contém um erro léxico que consiste em um termo literal que é inicializado entretanto, não é finalizado.

|  |  |
| --- | --- |
| **Programa: teste7**  declare  i, j, k, declare: int;  start  i = 4 ;  j = \*;(((\*  tt=2;  write(j);  write(j)\*\*\*\*/  write("Altura é /\* & demais);  k = i\* j / k;  k = 4 + a $;  write(j);  write(k\*);  end;  end | **Programa sem erros léxicos: teste7\_Corrigido**  declare  i, j, k, declare: int;  start  i = 4 ;  j = \*;(((\*  tt=2;  write(j);  write(j)\*\*\*\*/  write("Altura é &demais");  k = i\* j / k;  k = 4 + a;  write(j);  write(k\*);  end;  end |
| **Resultado do Teste 7:**  <declare,41>  <i,41>  <,,31>  <j,41>  <,,31>  <k,41>  <,,31>  <declare,41>  <:,32>  <int,41>  <;,30>  <start,41>  <i,41>  <=,35>  <4,40>  <;,30>  <j,41>  <=,35>  <\*,28>  <;,30>  <(,33>  <(,33>  <(,33>  <\*,28>  <tt,41>  <=,35>  <2,40>  <;,30>  <write,41>  <(,33>  <j,41>  <),34>  <;,30>  <write,41>  <(,33>  <j,41>  <),34>  <\*,28>  <\*,28>  <\*,28>  <\*,28>  </,29>  <write,41>  <(,33>  ERRO! String literal não finalizada!  <------ TABELA DE SÍMBOLOS ------->  tt  i  j  k | **Resultado do Teste7 Corrigido:**  <declare,41>  <i,41>  <,,31>  <j,41>  <,,31>  <k,41>  <,,31>  <declare,41>  <:,32>  <int,41>  <;,30>  <start,41>  <i,41>  <=,35>  <4,40>  <;,30>  <j,41>  <=,35>  <\*,28>  <;,30>  <(,33>  <(,33>  <(,33>  <\*,28>  <tt,41>  <=,35>  <2,40>  <;,30>  <write,41>  <(,33>  <j,41>  <),34>  <;,30>  <write,41>  <(,33>  <j,41>  <),34>  <\*,28>  <\*,28>  <\*,28>  <\*,28>  </,29>  <write,41>  <(,33>  <"Altura é &demais",36>  <),34>  <;,30>  <k,41>  <=,35>  <i,41>  <\*,28>  <j,41>  </,29>  <k,41>  <;,30>  <k,41>  <=,35>  <4,40>  <+,26>  <a,41>  <;,30>  <write,41>  <(,33>  <j,41>  <),34>  <;,30>  <write,41>  <(,33>  <k,41>  <\*,28>  <),34>  <;,30>  <end,41>  <;,30>  <end,41>  <------ TABELA DE SÍMBOLOS ------->  tt  a  i  j  k  CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo) |

**Arquivo de Teste 8**

Esse arquivo contém alguns erros léxicos e foram inseridos caracteres de formatação (\t) para verificar o comportamento do analisador léxico.

|  |  |
| --- | --- |
| **Programa: teste8**  programa teste8  declare  a, b, c: int;  result: int;  start  read [a];  read (b);  c = -30;  a = 15,7;  wr ite (a+c);  write (" sucesso!);  end | **Programa sem erros léxicos: teste8\_Corrigido**  programa teste8  declare  a,b,c: int;  result: int;  start  read a;  read (b);  c = -30;  a = 15,7;  wr ite (a+c);  write (" sucesso!");  end |
| **Resultado do Teste 8:**  <programa,41>  <teste8,41>  <declare,41>  <a,41>  <,,31>  <b,41>  <,,31>  <c,41>  <:,32>  <int,41>  <;,30>  <result,41>  <:,32>  <int,41>  <;,30>  <start,41>  <read,41>  ERRO - Caractere não especificado na linha 6 ch: "[".  <[,42>  <a,41>  ERRO - Caractere não especificado na linha 6 ch: "]".  <],42>  <;,30>  <read,41>  <(,33>  <b,41>  <),34>  <;,30>  <c,41>  <=,35>  <-,27>  <30,40>  <;,30>  <a,41>  <=,35>  <15,40>  <,,31>  <7,40>  <;,30>  <wr,41>  <ite,41>  <(,33>  <a,41>  <+,26>  <c,41>  <),34>  <;,30>  <write,41>  <(,33>  ERRO! String literal não finalizada!  <------ TABELA DE SÍMBOLOS ------->  result  a  teste8  b  c  programa  wr  ite  CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo) | **Resultado do Teste 8 Corrigido:**  <programa,41>  <teste8,41>  <declare,41>  <a,41>  <,,31>  <b,41>  <,,31>  <c,41>  <:,32>  <int,41>  <;,30>  <result,41>  <:,32>  <int,41>  <;,30>  <start,41>  <read,41>  <a,41>  <;,30>  <read,41>  <(,33>  <b,41>  <),34>  <;,30>  <c,41>  <=,35>  <-,27>  <30,40>  <;,30>  <a,41>  <=,35>  <15,40>  <,,31>  <7,40>  <;,30>  <wr,41>  <ite,41>  <(,33>  <a,41>  <+,26>  <c,41>  <),34>  <;,30>  <write,41>  <(,33>  <" sucesso!",36>  <),34>  <;,30>  <end,41>  <------ TABELA DE SÍMBOLOS ------->  result  a  teste8  b  c  programa  wr  ite  CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos) |