



CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
Engenharia de Computação

Trabalho Prático - Compiladores

Filipe da Silva Rocha (20193003809)
Matheus Freire Henrique Fonseca (20203002786)
Vitor Laguardia Xavier (201712060554)

Belo Horizonte
Janeiro de 2025

1. Sumário

| | |
|--|----------|
| 1. Sumário | 2 |
| 2. Forma de uso do compilador | 3 |
| 3. Gramática da Linguagem | 3 |
| 4. Execução dos testes | 4 |
| 4.1. Teste 1 | 4 |
| 4.2. Teste 2 | 4 |
| 4.3. Teste 3 | 4 |
| 4.4. Teste 4 | 4 |
| 4.5. Teste 5 | 4 |
| 4.6. Teste 6 | 5 |
| 5. Correção e funcionamento do analisador | 5 |
| 5.1. Teste 1 - corrigido | 5 |
| 5.2. Teste 2 - corrigido | 5 |
| 5.3. Teste 3 - corrigido Parte 1 | 5 |
| 5.4. Teste 3 - corrigido Parte 2 | 6 |
| 5.5. Teste 4 - corrigido | 6 |
| 5.6. Teste 5 - corrigido | 6 |

2. Forma de uso do compilador

Para executar o compilador, é necessário compilar o projeto usando o comando "javac Compiler.java" e após isso executá-lo passando algum arquivo de teste como entrada "java Compiler tests/test1.txt". Como exposto no exemplo, os testes estão na pasta "tests".

3. Gramática da Linguagem

3.1. Gramática Original

```
program      ::= start [decl-list] stmt-list exit
decl-list    ::= decl {decl}
decl         ::= type ident-list ";"
ident-list   ::= identifier {"," identifier}
type         ::= int | float | string
stmt-list    ::= stmt {stmt}
stmt         ::= assign-stmt ";" | if-stmt | while-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
while-stmt   ::= do stmt-list stmt-suffix
stmt-suffix  ::= while condition end
read-stmt    ::= scan "(" identifier ")"
write-stmt   ::= print "(" writable ")"
writable     ::= simple_expr | literal
expression   ::= simple_expr | simple_expr relop simple_expr
simple_expr   ::= term | simple_expr addop term
term         ::= factor-a | term mulop factor-a
factor-a     ::= factor | "!" factor | "-" factor
factor       ::= identifier | constant | "(" expression ")"
```

Padrões de tokens:

```
relop        ::= "==" | ">" | ">=" | "<" | "<=" | "!="
addop        ::= "+" | "-" | "||"
mulop        ::= "*" | "/" | "%" | "&&"
constant     ::= integer_const | float_const | literal
integer_const ::= digit+
float_const  ::= digit+ "."digit+
literal      ::= " { " caractere* " } "
identifier   ::= (letter | _ ) (letter | digit ) *
letter       ::= [A-Za-z]
digit        ::= [0-9]
caractere    ::= um dos caracteres ASCII, exceto quebra de linha
```

3.2. Gramática Alterada

1. $\langle \text{begin} \rangle ::= \langle \text{program} \rangle \#$
2. $\langle \text{program} \rangle ::= \text{start} [\text{decl-list}] \langle \text{stmt-list} \rangle \text{exit}$
3. $\langle \text{decl-list} \rangle ::= \langle \text{decl} \rangle \{ \langle \text{decl} \rangle \}$
4. $\langle \text{decl} \rangle ::= \langle \text{type} \rangle \langle \text{ident-list} \rangle ;$
5. $\langle \text{ident-list} \rangle ::= \text{identifier} \{ , \text{identifier} \}$
6. $\langle \text{type} \rangle ::= \text{int} \mid \text{float} \mid \text{string}$
7. $\langle \text{stmt-list} \rangle ::= \langle \text{stmt} \rangle \{ \langle \text{stmt} \rangle \}$
8. $\langle \text{stmt} \rangle ::= \langle \text{assign-stmt} \rangle ; \mid \langle \text{if-stmt} \rangle \mid \langle \text{while-stmt} \rangle \mid \langle \text{read-stmt} \rangle ; \mid \langle \text{write-stmt} \rangle ;$
9. $\langle \text{assign-stmt} \rangle ::= \text{identifier} = \langle \text{simple-expr} \rangle$
10. $\langle \text{if-stmt} \rangle ::= \text{if} \langle \text{condition} \rangle \text{then} \langle \text{stmt-list} \rangle \langle \text{if-stmt-tail} \rangle$
11. $\langle \text{if-stmt-tail} \rangle ::= \text{end} \mid \text{else} \langle \text{stmt-list} \rangle \text{end}$
12. $\langle \text{condition} \rangle ::= \langle \text{expression} \rangle$
13. $\langle \text{while-stmt} \rangle ::= \text{do} \langle \text{stmt-list} \rangle \langle \text{stmt-suffix} \rangle$
14. $\langle \text{stmt-suffix} \rangle ::= \text{while} \langle \text{condition} \rangle \text{end}$
15. $\langle \text{read-stmt} \rangle ::= \text{scan} (\text{identifier})$
16. $\langle \text{write-stmt} \rangle ::= \text{print} (\langle \text{writable} \rangle)$
17. $\langle \text{writable} \rangle ::= \langle \text{simple-expr} \rangle \mid \text{literal}$
18. $\langle \text{expression} \rangle ::= \langle \text{simple-expr} \rangle \langle \text{expression-tail} \rangle$
19. $\langle \text{expression-tail} \rangle ::= \text{relop} \langle \text{simple-expr} \rangle \mid \lambda$
20. $\langle \text{simple-expr} \rangle ::= \langle \text{term} \rangle \langle \text{simple-expr-tail} \rangle$
21. $\langle \text{simple-expr-tail} \rangle ::= \text{addop} \langle \text{term} \rangle \langle \text{simple-expr-tail} \rangle \mid \lambda$
22. $\langle \text{term} \rangle ::= \langle \text{factor-a} \rangle \langle \text{term-tail} \rangle$
23. $\langle \text{term-tail} \rangle ::= \text{mulop} \langle \text{factor-a} \rangle \langle \text{term-tail} \rangle \mid \lambda$
24. $\langle \text{factor-a} \rangle ::= \langle \text{factor} \rangle \mid ! \langle \text{factor} \rangle \mid - \langle \text{factor} \rangle$
25. $\langle \text{factor} \rangle ::= \text{identifier} \mid \text{constant} \mid (\langle \text{expression} \rangle)$

$\text{relop} \quad ::= "=" \mid ">" \mid ">=" \mid "<" \mid "<=" \mid "!="$

$\text{addop} \quad ::= "+" \mid "-" \mid "||"$

$\text{mulop} \quad ::= "*" \mid "/" \mid "\%" \mid "\&\&"$

$\text{constant} \quad ::= \text{integer_const} \mid \text{float_const} \mid \text{literal}$

$\text{integer_const} \quad ::= \text{digit}^+$

$\text{float_const} \quad ::= \text{digit}^+ "." \text{digit}^+$

$\text{literal} \quad ::= "{" \text{caractere}^* "}"$

$\text{identifier} \quad ::= (\text{letter} \mid _) (\text{letter} \mid \text{digit})^*$

$\text{letter} \quad ::= [\text{A-Za-z}]$

$\text{digit} \quad ::= [0-9]$

$\text{caractere} \quad ::= \text{um dos caracteres ASCII, exceto quebra de linha}$

As correções foram realizadas para remover o prefixo comum nas construções if-stmt e expression, e para remover as recursões a esquerda de simple-expr e term.

4. Tabela FIRST e FOLLOW

| | First | Follow |
|------------------|---|---|
| begin | start | \$ |
| program | start | \$ |
| decl-list | int ,float,string | identifier, if, do, scan, print |
| decl | int ,float,string | int ,float,string, identifier, if, do, scan, print |
| ident-list | identifier | ; |
| type | int ,float,string | identifier |
| stmt-list | identifier, if, do, scan, print | exit, end, while, else |
| stmt | identifier, if, do, scan, print | identifier, if, do, scan, print, exit, end, while, else |
| assign-stmt | identifier | ; |
| if-stmt | if | identifier, if, do, scan, print, exit, end, while, else |
| if-stmt-tail | end, else | identifier, if, do, scan, print, exit, end, while, else |
| condition | identifier, constant, (, !, - | then, end |
| while-stmt | do | identifier, if, do, scan, print, exit, end, while, else |
| stmt-suffix | while | identifier, if, do, scan, print, exit, end, while, else |
| read-stmt | scan | ; |
| write-stmt | print | ; |
| writable | identifier, constant, (, !, -, literal |) |
| expression | identifier, constant, (, !, - | then, end,) |
| expression-tail | relop, λ | then, end,) |
| simple-expr | identifier, constant, (, !, - | ;,), relop, then, end |
| simple-expr-tail | addop, λ | ;,), relop, then, end |
| term | identifier, constant, (, !, - | addop, ;,), relop, then, end |
| term-tail | mulop, λ | addop, ;,), relop, then, end |
| factor-a | identifier, constant, (, !, - | mulop, addop, ;,), relop, then, end |
| factor | identifier, constant, (| mulop, addop, ;,), relop, then, end |

5. Execução dos testes

5.1. Teste 1

| Teste Original: | Teste Final: |
|---|--|
| <pre>start float a; int b, result; float a,x,total; a = 2.0; x = 0.1; scan (b); scan (y) result = (a*b ++ 1) % a; print {Resultado: }; print (result); print ({Total: }); total = y / x; print ({Total: }; print (total); exit</pre> | <pre>start float a; int b, result; float a,x,total; a = 2.0; x = 0.1; scan (b); scan (y); result = (a*b + 1) % a; print ({Resultado: }); print (result); print ({Total: }); total = y / x; print ({Total: }); print (total); exit</pre> |

Execução (A partir do teste original):

- Erro 1

```
Comeu SCAN
Comeu OPEN_PAR
Comeu IDENTIFIER
Comeu CLOSE_PAR
Error: Erro no Stmt at line 10
```

Programa esperava receber o ';' final do scan.

- Erro 2

```
Comeu IDENTIFIER
Comeu ASSIGN
Comeu OPEN_PAR
Comeu IDENTIFIER
Comeu MULT
Comeu IDENTIFIER
Comeu PLUS
Error: Erro no factor a at line 10
```

Programa esperava <term> depois de addop, mas recebeu outro addop.

- **Erro 3**

```
Comeu PRINT  
Error: Erro em write-stmt at line 11
```

Programa esperava o input do print entre parênteses (já resolvendo o erro de não fechamento de parênteses que aconteceria se colocasse só o open_par)

- **Erro 4**

```
Comeu PRINT  
Comeu OPEN_PAR  
Comeu STRING_CONST  
Error: Erro em write-stmt at line 15
```

Programa esperava ")" depois de print.

- **Finalizada correção**

```
Comeu EXIT  
Comeu EOF  
  
----- Compilation Result -----  
Compilation SUCCESS!  
-----
```

5.2. Teste 2

| Teste Original: | Teste Final: |
|--|--|
| <pre>start int a, c_; float d, _e; a = 0; d = 3.5 c = d / 1.2; Scan (a); Scan (c); b = a * a; c = b + a * (1 + a*c); print ({Resultado: }); print c; d = 34.2 e = val + 2.2; print ({E: }); print (e); a = b + c + a)/2;</pre> | <pre>start int a, c_; float d, _e; a = 0; d = 3.5; c = d / 1.2; scan (a); scan (c); b = a * a; c = b + a * (1 + a*c); print ({Resultado: }); print (c); d = 34.2; e = val + 2.2; print ({E: }); print (e); a = (b + c + a)/2; exit</pre> |

Execução (A partir do teste original):

- Erro 1

```
Comeu IDENTIFIER
Comeu ASSIGN
Comeu FLOAT_CONST
Error: Erro no Stmt at line 7
```

O programa esperava ';' depois do float em d=3.5.

- Erro 2

```
Comeu IDENTIFIER
Error: Erro no assign at line 9
```

O programa não conseguiu identificar o 'Scan' como sendo scan e deu erro. Ocorre por ser Case sensitive. (Corrigindo Scan na linha abaixo também).

- Erro 3

```
Comeu PRINT
Error: Erro em write-stmt at line 14
```

O programa esperava a estrutura do print entre parênteses.

- **Erro 4**

```
Comeu IDENTIFIER
Comeu ASSIGN
Comeu FLOAT_CONST
Error: Erro no Stmt at line 16
```

O programa esperava ';' depois do float em d=34.2.

- **Erro 5**

```
Comeu IDENTIFIER
Comeu ASSIGN
Comeu IDENTIFIER
Comeu PLUS
Comeu IDENTIFIER
Comeu PLUS
Comeu IDENTIFIER
Error: Erro no Stmt at line 19
```

O programa recebeu um Close_Par, mas não recebeu um Open_par, ou seja, não formaria a construção Identifier.

- **Erro 6**

```
Comeu IDENTIFIER
Comeu ASSIGN
Comeu OPEN_PAR
Comeu IDENTIFIER
Comeu PLUS
Comeu IDENTIFIER
Comeu PLUS
Comeu IDENTIFIER
Comeu CLOSE_PAR
Comeu DIV
Comeu INT_CONST
Comeu SEMICOLON
java.lang.Exception: Erro sintático: esperava EXIT
, mas encontrou EOF LINHA: 20
```

O programa finalizou e tentou encontrar o Exit, mas como não tinha, encontrou EOF.

```
Comeu EXIT
Comeu EOF

----- Compilation Result -----
Compilation SUCCESS!
-----
```

5.3. Teste 3

| Teste Original: | Teste Final: |
|--|--|
| <pre>int pontuacao, pontuacaoMaxima, disponibilidade; string pontuacaoMinima; disponibilidade = Sim; pontuacaoMinima = 50; pontuacaoMaxima = 100; /* Entrada de dados */ Verifica aprovação de candidatos do print({Pontuacao Candidato: }); scan(pontuacao); print({Disponibilidade Candidato: }); scan(disponibilidade); if ((pontuação > pontuacaoMinima) && (disponibilidade=={Sim})) then out({Candidato aprovado}); else out({Candidato reprovado}); end while (pontuação >= 0)end exit</pre> | <pre>start int pontuacao, pontuacaoMaxima, disponibilidade; string pontuacaoMinima; disponibilidade = Sim; pontuacaoMinima = 50; pontuacaoMaxima = 100; /* Entrada de dados Verifica aprovação de candidatos */ do print({Pontuacao Candidato: }); scan(pontuacao); print({Disponibilidade Candidato: }); scan(disponibilidade); if ((pontuação > pontuacaoMinima) && (disponibilidade=={Sim})) then print({Candidato aprovado}); else print({Candidato reprovado}); end while (pontuação >= 0)end exit</pre> |

Execução (A partir do teste original):

- Erro 1

```
java.lang.Exception: Erro sintático: esperava STAR
T, mas encontrou INT LINHA: 1
```

O programa não possui Start

- Erro 2

```
Comeu IDENTIFIER
Error: Erro no assign at line 10
```

“Verifica aprovação de candidatos” tinha ficado fora do comentário.

- **Erro 3**

```
Comeu IF
Comeu OPEN_PAR
Comeu OPEN_PAR
Comeu IDENTIFIER
Comeu GREATER
Comeu IDENTIFIER
Comeu CLOSE_PAR
Comeu AND
Comeu OPEN_PAR
Comeu IDENTIFIER
Comeu EQ
Comeu STRING_CONST
Comeu CLOSE_PAR
Error: Erro em term at line 17
```

O programa não recebeu o close_par no if então não conseguiu ler o then por considerar que o if ainda não tinha terminado.

- **Erro 4**

```
Comeu THEN
Comeu IDENTIFIER
Error: Erro no assign at line 18
```

Após o Then, o programa recebe a estrutura out(literal), que não é definida na gramática, então retorna erro. Corrigindo para print. (corrigindo o out que aparecia após else também)

- **Erro 5**

```
Comeu ELSE
Comeu PRINT
Comeu OPEN_PAR
Comeu STRING_CONST
Comeu CLOSE_PAR
Error: Erro no Stmt at line 21
```

Além do out, faltou um ';' nessa construção

```
Comeu WHILE
Comeu OPEN_PAR
Comeu IDENTIFIER
Comeu GREATER_EQ
Comeu INT_CONST
Comeu CLOSE_PAR
Comeu END
Comeu EXIT
Comeu EOF

----- Compilation Result -----
Compilation SUCCESS!
-----
```

5.4. Teste 4

| Teste Original: | Teste Final: |
|--|--|
| <pre> start Int a, aux, b; string nome, sobrenome, msg; print(Nome); scan (nome); print({Sobrenome: }); scan (sobrenome); msg = {Ola, } + nome + { } + sobrenome + {!}; msg = msg + 1; print (msg); scan (a); scan(b); if (a>b) then aux = b; b = a; a = aux; end; print ({Apos a troca: }); out(a); out(b) exit </pre> | <pre> start int a, aux, b; string nome, sobrenome, msg; print(Nome); scan (nome); print({Sobrenome: }); scan (sobrenome); msg = {Ola, } + nome + { } + sobrenome + {!}; msg = msg + 1; print (msg); scan (a); scan(b); if (a>b) then aux = b; b = a; a = aux; end print ({Apos a troca: }); print(a); print(b); exit </pre> |

Execução (A partir do teste original):

- Erro 1

```

Comeu START
Comeu IDENTIFIER
Error: Erro no assign at line 2

```

O programa não reconhece Int como type, já que é case sensitive

- Erro 2

```

Comeu END
java.lang.Exception: Erro sintático: esperava EXIT
, mas encontrou SEMICOLON LINHA: 20

```

O programa recebeu o ';' após o end e não conseguiu interpretar, já que após o end não há ';' na construção da gramática. O parser tenta retornar para a raiz que chama o if-stmt, ou seja, a stmt-list, e por isso vemos a tentativa de ler o token Exit.

- **Erro 3**

```
Comeu PRINT
Comeu OPEN_PAR
Comeu STRING_CONST
Comeu CLOSE_PAR
Comeu SEMICOLON
Comeu IDENTIFIER
Error: Erro no assign at line 22
```

O programa recebe a estrutura out(identifier), que não é definida na gramática, então retorna erro. Corrigindo para print. (corrigindo o out que aparecia após else também).

- **Erro 4**

```
Comeu SEMICOLON
Comeu PRINT
Comeu OPEN_PAR
Comeu IDENTIFIER
Comeu CLOSE_PAR
Error: Erro no Stmt at line 24
```

Além do out, faltou um ';' nessa construção

```
Comeu SEMICOLON
Comeu EXIT
Comeu EOF

----- Compilation Result -----
Compilation SUCCESS!
-----
```

5.5. Teste 5

| Teste Original: | Teste Final: |
|--|--|
| <pre>start int a, b, c, maior, outro; do print({A}); scan(a); print({B}); scan(b); print({C}); scan(c); //Realizacao do teste if ((a>b) && (a>c)) maior = a else if (b>c) then maior = b; else maior = c; end end end print({Maior valor:}); print (maior); print ({Outro? }); scan(outro); while (outro >= 0); exit</pre> | <pre>start int a, b, c, maior, outro; do print({A}); scan(a); print({B}); scan(b); print({C}); scan(c); //Realizacao do teste if ((a>b) && (a>c))then maior = a; else if (b>c) then maior = b; else maior = c; end end end print({Maior valor:}); print (maior); print ({Outro? }); scan(outro); while (outro >= 0) end exit</pre> |

Execução (A partir do teste original):

- Erro 1

```
Comeu IF
Comeu OPEN_PAR
Comeu OPEN_PAR
Comeu IDENTIFIER
Comeu GREATER
Comeu IDENTIFIER
Comeu CLOSE_PAR
Comeu AND
Comeu OPEN_PAR
Comeu IDENTIFIER
Comeu GREATER
Comeu IDENTIFIER
Comeu CLOSE_PAR
Comeu CLOSE_PAR
Error: Erro no ifstmt at line 14
```

Esperava a construção then.

- Erro 2

```
Comeu THEN
Comeu IDENTIFIER
Comeu ASSIGN
Comeu IDENTIFIER
Error: Erro no Stmt at line 16
```

O programa esperava receber o ';' na definição maior = a.

- Erro 3

```
Comeu WHILE
Comeu OPEN_PAR
Comeu IDENTIFIER
Comeu GREATER_EQ
Comeu INT_CONST
Comeu CLOSE_PAR
Error: Erro em stmt-sufix at line 28
```

O programa recebeu um ';' após o while, construção incorreta.

- Erro 4

```
Comeu WHILE
Comeu OPEN_PAR
Comeu IDENTIFIER
Comeu GREATER_EQ
Comeu INT_CONST
Comeu CLOSE_PAR
Error: Erro em stmt-sufix at line 29
```

O programa não recebeu o end do While.

```

Comeu WHILE
Comeu OPEN_PAR
Comeu IDENTIFIER
Comeu GREATER_EQ
Comeu INT_CONST
Comeu CLOSE_PAR
Comeu END
Comeu EXIT
Comeu EOF

----- Compilation Result -----
Compilation SUCCESS!
-----

```

5.6. Teste 6

| Teste Original: | Teste Final: |
|--|--|
| <pre> start int x, y, z float a, b; string nome; x = 10 y = x + 5; z = y * 2; a = 3.14; b = a / 2.0; nome = {Teste}; if x > y then print({x é maior que y}); else print({y é maior ou igual a x}); end do x = x - 1; y = y + 1; while x > 0 end scan(z); print({Valor de z: }, z); x = y = 5; if x = y then print({x e y são iguais}); end </pre> | <pre> start int x, y, z; float a, b; string nome; x = 10; y = x + 5; z = y * 2; a = 3.14; b = a / 2.0; nome = {Teste}; if x > y then print({x é maior que y}); else print({y é maior ou igual a x}); end do x = x - 1; y = y + 1; while x > 0 end scan(z); print({Valor de z: }); print(z); x = y - 5; if x == y then print({x e y são iguais}); end </pre> |

| | |
|---|--|
| <pre>print({Fim do programa}); exit</pre> | <pre>end print({Fim do programa}); exit</pre> |
|---|--|

Execução (A partir do teste original):

- Erro 1

```
Comeu START
Comeu INT
Comeu IDENTIFIER
Comeu COMMA
Comeu IDENTIFIER
Comeu COMMA
Comeu IDENTIFIER
Error: Erro na decl at line 3
```

Esperava “;” no final da decl.

- Erro 2

```
Comeu START
Comeu INT
Comeu IDENTIFIER
Comeu COMMA
Comeu IDENTIFIER
Comeu COMMA
Comeu IDENTIFIER
Comeu SEMICOLON
Comeu FLOAT
Comeu IDENTIFIER
Comeu COMMA
Comeu IDENTIFIER
Comeu SEMICOLON
Comeu STRING
Comeu IDENTIFIER
Comeu SEMICOLON
Comeu IDENTIFIER
Comeu ASSIGN
Comeu INT_CONST
Error: Erro no Stmt at line 7
```

Esperava “;” no final da stmt.

- **Erro 3**

```
Comeu INT_CONST
Comeu SEMICOLON
Comeu IDENTIFIER
Comeu ASSIGN
Comeu FLOAT_CONST
Comeu SEMICOLON
Comeu IDENTIFIER
Comeu ASSIGN
Comeu IDENTIFIER
Comeu DIV
Comeu FLOAT_CONST
Comeu SEMICOLON
Comeu IDENTIFIER
Comeu ASSIGN
Comeu STRING_CONST
Comeu SEMICOLON
Comeu IF
Comeu IDENTIFIER
Comeu GREATER
Comeu IDENTIFIER
Comeu THEN
Comeu PRINT
Comeu OPEN_PAR
Error: Erro no writable at line 14
```

Esperava “}” (CLOSE_PAR) no final da string.

- **Erro 4**

```
Comeu PLUS
Comeu INT_CONST
Comeu SEMICOLON
Comeu WHILE
Comeu IDENTIFIER
Comeu GREATER
Comeu INT_CONST
Comeu END
Comeu SCAN
Comeu OPEN_PAR
Comeu IDENTIFIER
Comeu CLOSE_PAR
Comeu SEMICOLON
Comeu PRINT
Comeu OPEN_PAR
Comeu STRING_CONST
Error: Erro em write-stmt at line 25
```

Esperava um término do “print” com “)” mas como foram passados dois argumentos no print, uma string “,” e uma variável. Ele apresenta erro.

- **Erro 5**

```
Comeu IDENTIFIER
Comeu CLOSE_PAR
Comeu SEMICOLON
Comeu PRINT
Comeu OPEN_PAR
Comeu STRING_CONST
Comeu CLOSE_PAR
Comeu SEMICOLON
Comeu PRINT
Comeu OPEN_PAR
Comeu IDENTIFIER
Comeu CLOSE_PAR
Comeu SEMICOLON
Comeu IDENTIFIER
Comeu ASSIGN
Comeu IDENTIFIER
Error: Erro no Stmt at line 28
```

Erro de uma dupla atribuição. Nesse caso “x = y = 5;” ele encontrou erro no stmt já que a primeira atribuição não foi terminada corretamente.

- **Erro 6**

```
Comeu STRING_CONST
Comeu CLOSE_PAR
Comeu SEMICOLON
Comeu PRINT
Comeu OPEN_PAR
Comeu IDENTIFIER
Comeu CLOSE_PAR
Comeu SEMICOLON
Comeu IDENTIFIER
Comeu ASSIGN
Comeu IDENTIFIER
Comeu MINUS
Comeu INT_CONST
Comeu SEMICOLON
Comeu IF
Comeu IDENTIFIER
Error: Erro no ifstmt at line 29
```

Esperava uma comparação relap mas encontrou somente “=” e deu erro no ifstmt.

Programa corrigido:

```
Comeu PRINT
Comeu OPEN_PAR
Comeu STRING_CONST
Comeu CLOSE_PAR
Comeu SEMICOLON
Comeu END
Comeu PRINT
Comeu OPEN_PAR
Comeu STRING_CONST
Comeu CLOSE_PAR
Comeu SEMICOLON
Comeu EXIT
Comeu EOF

----- Compilation Result -----
Compilation SUCCESS!
-----
```