CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

Departamento de Computação

Compiladores

Terça-feira e quinta-feira: 7h00 às 8h40

Professora: Kécia Aline Marques Ferreira

Analisador Léxico

Gabriel Pires de Miranda Magalhães

Renan Mateus Bernardo do Nascimento

Vinícius Magalhães D'Assunção

Belo Horizonte, CEFET-MG Campus II

Setembro de 2017

Instruções de execução analisador léxico

Junto com o código fonte do analisador léxico está o arquivo *compilador.jar*. Para executar este arquivo, independente do sistema operacional basta abrir o terminal e digitar o comando:

java -jar compilador.jar

Ao executar será exibida uma mensagem solicitando que seja colocado o caminho completo do código que será analisado pelo analisador léxico. Os códigos de teste propostos pela professora estão no diretório testes/, juntamente com um código de teste proposto pelos autores do projeto.

Implementação do analisador léxico

A implementação utilizada codifica o comportamento do AFD direto no código sem usar explicitamente uma variável para armazenar o estado corrente do autômato. O projeto possui as seguintes classes:

- Tag: faz o mapeamento de cada identificador de token para uma constante;
- Word: representa um token de palavras reservadas, identificadores e tokens compostos;
- Num: representa um token número;
- Token: representa um token genérico;
- Lexer: classe que implementa o analisador léxico. Lê cada caractere do arquivo até compor um lexema correspondente à um token. Caso não tenha sido formado um token, é gerada uma exceção, que para esta etapa é tratada na classe Main. exibindo o erro no formato “Error(<linha de ocorrência>): Token ‘<token que gerou o erro>’ inválido”. Esta classe também faz a inserção das palavras reservadas antes de iniciar a leitura do arquivo e dos identificadores na tabela de símbolos durante a leitura do mesmo.
- Main: classe principal que chama a classe Lexer. Há um laço que recebe o token retornado pelo Lexer e este token é impresso no formato <identificador, valor>. É feita a leitura até que o token recebido indique fim de arquivo. Caso ocorra uma exceção, o erro é impresso na tela e é dada continuidade ao processo de leitura dos tokens. Por fim, após a listagem dos tokens é impressa a tabela de símbolos.

Resultados dos testes

Teste 1

- Código Fonte:

```
program
        int a, b;
        int result;
        float a,x,total;

        a = 2;
        x = .1;
        scan (b);
        scan (y)
        result = (a*b ++ 1) / 2;
        print "Resultado: ";
        print (result);
        print ("Total: ");
        total = y / x;
        print ("Total: ";
        print (total);
end
```

- Saída do analisador léxico:

```
**** Tokens lidos ****
< program >
< int >
< id, a >
< virgula >
< id, b >
< ponto_virgula >
< int >
< id, result >
< ponto_virgula >
< id, float >
< id, a >
< virgula >
< id, x >
< virgula >
< id, total >
< ponto_virgula >
< id, a >
< assign >
< num, 2 >
< ponto_virgula >
< id, x >
< assign >
Error(7): Token '.' inválido
< num, 1 >
< ponto_virgula >
< scan >
< abre_parent >
< id, b >
< fecha_parent >
< ponto_virgula >
< scan >
```

```
< abre_parent >
< id, y >
< fecha_parent >
< id, result >
< assign >
< abre_parent >
< id, a >
< mult >
< id, b >
< soma >
< soma >
< num, 1 >
< fecha_parent >
< div >
< num, 2 >
< ponto_virgula >
< print >
< literal, "Resultado: " >
< ponto_virgula >
< print >
< abre_parent >
< id, result >
< fecha_parent >
< ponto_virgula >
< print >
< abre_parent >
< literal, "Total: " >
< fecha_parent >
< ponto_virgula >
< id, total >
< assign >
< id, y >
< div >
< id, x >
< ponto_virgula >
```

```
< print >
< abre_parent >
< literal, "Total: " >
< ponto_virgula >
< print >
< abre_parent >
< id, total >
< fecha_parent >
< ponto_virgula >
< end >
```

```
**** Tabela de símbolos ****
Entrada         |         Mais info
string
int
print
end
program
y
x
result
scan
if
while
do
else
then
b
a
float
total

Process finished with exit code 0
```

- Código Fonte - consertado:

```
program
        int a, b;
        int result;
        float a,x,total;
```

```
        a = 2;
        x = 1;
        scan (b);
        scan (y)
        result = (a*b ++ 1) / 2;
        print "Resultado: ";
        print (result);
        print ("Total: ");
        total = y / x;
        print ("Total: ";
        print (total);
end
```

- Saída do analisador léxico - código fonte consertado

```
**** Tokens lidos ****
< program >
< int >
< id, a >
< virgula >
< id, b >
< ponto_virgula >
< int >
< id, result >
< ponto_virgula >
< id, float >
< id, a >
< virgula >
< id, x >
< virgula >
< id, total >
< ponto_virgula >
< id, a >
< assign >
< num, 2 >
< ponto_virgula >
< id, x >
< assign >
< num, 1 >
< ponto_virgula >
< scan >
< abre_parent >
< id, b >
< fecha_parent >
< ponto_virgula >
< scan >
< abre_parent >
< id, y >
< fecha_parent >
< id, result >
< assign >
< abre_parent >
< id, a >
< mult >
< id, b >
< soma >
< soma >
< num, 1 >
< fecha_parent >
< div >
< num, 2 >
< ponto_virgula >
< print >
< literal, "Resultado: " >
< ponto_virgula >
< print >
< abre_parent >
```

```
< id, result >
< fecha_parent >
< ponto_virgula >
< print >
< abre_parent >
< literal, "Total: " >
< fecha_parent >
< ponto_virgula >
< id, total >
< assign >
< id, y >
< div >
< id, x >
< ponto_virgula >
< print >
< abre_parent >
< literal, "Total: " >
< ponto_virgula >
< print >
< abre_parent >
< id, total >
< fecha_parent >
< ponto_virgula >
< end >
```

```
**** Tabela de símbolos ****
Entrada         |         Mais info
string
int
print
end
program
y
x
result
scan
if
while
do
else
then
b
a
float
total

Process finished with exit code 0
```

Teste 2

- Código Fonte:

```
program

    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;
    a = b + c + d)/2;
    e = val + c + a;
    print ("E: ");
    print (e);
```

- Saída do analisador léxico

```
**** Tokens lidos ****
< program >
< int >
Error(3): Token ':' inválido
< id, a >
< virgula >
< id, c >
< ponto_virgula >
< id, float >
< id, d >
< virgula >
Error(4): Token '_' inválido
< id, e >
< ponto_virgula >
< id, a >
< assign >
< num, 0 >
< ponto_virgula >
< id, d >
< assign >
< num, 3 >
Error(6): Token '.' inválido
< num, 5 >
< id, c >
< assign >
< id, d >
< div >
< num, 1 >
Error(7): Token '.' inválido
< num, 2 >
< ponto_virgula >
< id, Scan >
< abre_parent >
< id, a >
```

```
< fecha_parent >
< ponto_virgula >
< id, Scan >
< abre_parent >
< id, c >
< fecha_parent >
< ponto_virgula >
< id, b >
< assign >
< id, a >
< mult >
< id, a >
< ponto_virgula >
< id, c >
< assign >
< id, b >
< soma >
< id, a >
< mult >
< abre_parent >
< num, 1 >
< soma >
< id, a >
< mult >
< id, c >
< fecha_parent >
< ponto_virgula >
< print >
< abre_parent >
< literal, "Resultado: " >
< fecha_parent >
< ponto_virgula >
< print >
< id, c >
< ponto_virgula >
< id, a >
< assign >
< id, b >
< soma >
< id, c >
< soma >
< id, d >
< fecha_parent >
< div >
< num, 2 >
< ponto_virgula >
< id, e >
< assign >
< id, val >
< soma >
< id, c >
< soma >
< id, a >
< ponto_virgula >
< print >
< abre_parent >
< literal, "E: " >
< fecha_parent >
< ponto_virgula >
< print >
< abre_parent >
< id, e >
< fecha_parent >
< ponto_virgula >
```

**** Tabela de símbolos ****

Entrada	Mais info
string	
int	
print	
end	
program	
scan	
if	
while	
do	
else	
Scan	
e	
then	
d	
c	

b
a
float
val
Process finished with exit code 0

- Código Fonte - consertado

```
program

    int a, c;
    float d, e;

    a = 0; d = 35
    c = d / 12;

    Scan (a);
    Scan (c);
    b = a * a;
    c = b + a * (1 + a*c);
    print ("Resultado: ");
    print c;
    a = b + c + d)/2;
    e = val + c + a;
    print ("E: ");
    print (e);
```

- Saída do analisador léxico - código fonte consertado

```
**** Tokens lidos ****
< program >
< int >
< id, a >
< virgula >
< id, c >
< ponto_virgula >
< id, float >
< id, d >
< virgula >
< id, e >
< ponto_virgula >
< id, a >
< assign >
< num, 0 >
< ponto_virgula >
< id, d >
< assign >
< num, 35 >
< id, c >
< assign >
< id, d >
< div >
< num, 12 >
< ponto_virgula >
< id, Scan >
< abre_parent >
< id, a >
< fecha_parent >
< ponto_virgula >
```

```
< id, Scan >
< abre_parent >
< id, c >
< fecha_parent >
< ponto_virgula >
< id, b >
< assign >
< id, a >
< mult >
< id, a >
< ponto_virgula >
< id, c >
< assign >
< id, b >
< soma >
< id, a >
< mult >
< abre_parent >
< num, 1 >
< soma >
< id, a >
< mult >
< id, c >
< fecha_parent >
< ponto_virgula >
< print >
< abre_parent >
< literal, "Resultado: " >
< fecha_parent >
< ponto_virgula >
< print >
< id, c >
< ponto_virgula >
< id, a >
< assign >
< id, b >
< soma >
< id, c >
< soma >
< id, d >
< fecha_parent >
< div >
< num, 2 >
< ponto_virgula >
< id, e >
< assign >
< id, val >
< soma >
< id, c >
< soma >
< id, a >
< ponto_virgula >
< print >
< abre_parent >
< literal, "E: " >
< fecha_parent >
< ponto_virgula >
< print >
< abre_parent >
< id, e >
< fecha_parent >
< ponto_virgula >

**** Tabela de símbolos ****
Entrada        |        Mais info
string
int
print
end
program
scan
if
while
do
else
Scan
e
then
d
c
b
a
```

```
float
val

Process finished with exit code 0
```

Teste 3

- Código Fonte

```
program
	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) and (disponibilidade=="Sim") then
			print("Candidato aprovado");
		else
			print("Candidato reprovado")
		end
	while (pontuação >= 0)end
end
```

- Saída do analisador léxico

```
**** Tokens lidos ****
< program >
< int >
< id, pontuacao >
< virgula >
< id, pontuacaoMaxima >
< virgula >
< id, disponibilidade >
< ponto_virgula >
< string >
< id, pontuacaoMinima >
< ponto_virgula >
```

```
< id, disponibilidade >
< assign >
< literal, "Sim" >
< ponto_virgula >
< id, pontuacaoMinima >
< assign >
< num, 50 >
< ponto_virgula >
< id, pontuacaoMaxima >
< assign >
< num, 100 >
< ponto_virgula >
Error(23): comentário não fechado
```

```
**** Tabela de símbolos ****
Entrada         |         Mais info
print
if
pontuacao
disponibilidade
end
while
do
then
program
string
int
pontuacaoMaxima
else
pontuacaoMinima
scan

Process finished with exit code 0
```

- Código Fonte - consertado

```
program
        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) and (disponibilidade=="Sim") then
                        print("Candidato aprovado");
                else
                        print("Candidato reprovado")
```

```
                end
        while (pontuação >= 0)end
end
*/
```

- Saída analisador léxico - código fonte consertado

```
**** Tokens lidos ****
< program >
< int >
< id, pontuacao >
< virgula >
< id, pontuacaoMaxima >
< virgula >
< id, disponibilidade >
< ponto_virgula >
< string >
< id, pontuacaoMinima >
< ponto_virgula >
< id, disponibilidade >
< assign >
< literal, "Sim" >
< ponto_virgula >
< id, pontuacaoMinima >
< assign >
< num, 50 >
< ponto_virgula >
< id, pontuacaoMaxima >
< assign >
< num, 100 >
< ponto_virgula >
```

```
**** Tabela de símbolos ****
Entrada         |         Mais info
print
if
pontuacao
disponibilidade
end
while
do
then
program
string
int
pontuacaoMaxima
else
pontuacaoMinima
scan

Process finished with exit code 0
```

Teste 4

- Código Fonte

```
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)
end
```

- Saída do analisador léxico

```
**** Tokens lidos ****
< int >
Error(1): Token ':' inválido
< id, a >
< virgula >
< id, aux >
Error(1): Token '$' inválido
< virgula >
< id, b >
< ponto_virgula >
< string >
< id, nome >
< virgula >
< id, sobrenome >
< virgula >
< id, msg >
< ponto_virgula >
< print >
< abre_parent >
< id, Nome >
Error(4): Token ':' inválido
< fecha_parent >
< ponto_virgula >
< scan >
```

< abre_parent >
< id, nome >
< fecha_parent >
< ponto_virgula >
< print >
< abre_parent >
< literal, "Sobrenome: " >
< fecha_parent >
< ponto_virgula >
< scan >
< abre_parent >
< id, sobrenome >
< fecha_parent >
< ponto_virgula >
< id, msg >
< assign >
< literal, "Ola, " >
< soma >
< id, nome >
< soma >
< literal, " " >
< soma >
< id, sobrenome >
< soma >
< literal, "!" >
< ponto_virgula >
< id, msg >
< assign >
< id, msg >
< soma >
< num, 1 >
< ponto_virgula >
< print >
< abre_parent >
< id, msg >
< fecha_parent >
< ponto_virgula >
< scan >
< abre_parent >
< id, a >
< fecha_parent >
< ponto_virgula >
< scan >
< abre_parent >
< id, b >
< fecha_parent >
< ponto_virgula >
< if >
< abre_parent >
< id, a >
< greater_than >
< id, b >
< fecha_parent >
< then >
< id, aux >
< assign >
< id, b >
< ponto_virgula >
< id, b >
< assign >
< id, a >
< ponto_virgula >
< id, a >
< assign >
< id, aux >
< ponto_virgula >
< end >
< ponto_virgula >
< print >
< abre_parent >
< literal, "Apos a troca: " >
< fecha_parent >
< ponto_virgula >
< id, out >
< abre_parent >
< id, a >
< fecha_parent >
< ponto_virgula >
< id, out >
< abre_parent >
< id, b >
< fecha_parent >
< end >

```
**** Tabela de símbolos ****
Entrada         |         Mais info
string
int
print
end
msg
program
sobrenome
scan
if
while
do
nome
out
else
then
b
a
aux
Nome

Process finished with exit code 0
```

- Código Fonte - consetado

```
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)
end
```

- Saída do analisador léxico - código fonte consertado

```
**** Tokens lidos ****
< int >
< id, a >
< virgula >
< id, aux >
< virgula >
< id, b >
< ponto_virgula >
< string >
< id, nome >
< virgula >
< id, sobrenome >
< virgula >
< id, msg >
< ponto_virgula >
< print >
< abre_parent >
< id, Nome >
< fecha_parent >
< ponto_virgula >
< scan >
< abre_parent >
< id, nome >
< fecha_parent >
< ponto_virgula >
< print >
< abre_parent >
< literal, "Sobrenome: " >
< fecha_parent >
< ponto_virgula >
< scan >
< abre_parent >
< id, sobrenome >
< fecha_parent >
< ponto_virgula >
< id, msg >
< assign >
< literal, "Ola, " >
< soma >
< id, nome >
< soma >
< literal, " " >
< soma >
< id, sobrenome >
< soma >
< literal, "!" >
< ponto_virgula >
< id, msg >
< assign >
< id, msg >
< soma >
< num, 1 >
< ponto_virgula >
< print >
< abre_parent >
< id, msg >
< fecha_parent >
< ponto_virgula >
< scan >
< abre_parent >
< id, a >
< fecha_parent >
< ponto_virgula >
< scan >
< abre_parent >
< id, b >
< fecha_parent >
< ponto_virgula >
< if >
< abre_parent >
< id, a >
< greater_than >
< id, b >
< fecha_parent >
< then >
< id, aux >
< assign >
< id, b >
```

```
< ponto_virgula >
< id, b >
< assign >                                  **** Tabela de símbolos ****
< id, a >                                   Entrada         |       Mais info
< ponto_virgula >                           string
< id, a >                                   int
< assign >                                  print
< id, aux >                                 end
< ponto_virgula >                           msg
< end >                                     program
< ponto_virgula >                           sobrenome
< print >                                   scan
< abre_parent >                             if
< literal, "Apos a troca: " >               while
< fecha_parent >                            do
< ponto_virgula >                           nome
< id, out >                                 out
< abre_parent >                             else
< id, a >                                   then
< fecha_parent >                            b
< ponto_virgula >                           a
< id, out >                                 aux
< abre_parent >                             Nome
< id, b >
< fecha_parent >                            Process finished with exit code 0
< end >
```

Teste 5

- Código Fonte

```
program
	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
		print("Maior valor:"");
		print (maior);
		print ("Outro? ");
		scan(outro);
	while (outro >= 0)
end
```

- Saída do analisador léxico

```
**** Tokens lidos ****
< program >
< int >
< id, a >
< virgula >
< id, b >
< virgula >
< id, c >
< virgula >
< id, maior >
< virgula >
< id, outro >
```

< ponto_virgula >
< do >
< print >
< abre_parent >
< literal, "A" >
< fecha_parent >
< ponto_virgula >
< scan >
< abre_parent >
< id, a >
< fecha_parent >
< ponto_virgula >
< print >
< abre_parent >
< literal, "B" >
< fecha_parent >
< ponto_virgula >
< scan >
< abre_parent >
< id, b >
< fecha_parent >
< ponto_virgula >
< print >
< abre_parent >
< literal, "C" >
< fecha_parent >
< ponto_virgula >
< scan >
< abre_parent >
< id, c >
< fecha_parent >
< ponto_virgula >
< if >
< abre_parent >
< abre_parent >
< id, a >
< greater_than >
< id, b >
< fecha_parent >
< and >
< abre_parent >
< id, a >
< greater_than >
< id, c >
< fecha_parent >
< fecha_parent >
< id, maior >
< assign >
< id, a >
< else >
< if >
< abre_parent >
< id, b >
< greater_than >
< id, c >
< fecha_parent >
< then >
< id, maior >
< assign >
< id, b >
< ponto_virgula >
< else >
< id, maior >
< assign >
< id, c >
< ponto_virgula >
< end >
< end >
< print >
< abre_parent >
< literal, "Maior valor:" >
Error(24): Token "" inválido
< print >
< abre_parent >
< id, maior >
< fecha_parent >
< ponto_virgula >
< print >
< abre_parent >
< literal, "Outro? " >
< fecha_parent >
< ponto_virgula >
< scan >
< abre_parent >

```
< id, outro >
< fecha_parent >
< ponto_virgula >
< while >
< abre_parent >
< id, outro >
< greater_equals >
< num, 0 >
< fecha_parent >
< end >
```

```
**** Tabela de símbolos ****
Entrada         |         Mais info
print
if
end
while
do
then
program
maior
string
int
outro
c
b
else
a
scan

Process finished with exit code 0
```

- Código Fonte - consertado

```
program
        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
                print("Maior valor:");
                print (maior);
                print ("Outro? ");
                scan(outro);
            while (outro >= 0)
end
```

- Saída do analisador léxico - código fonte consertado

```
**** Tokens lidos ****
< program >
< int >
< id, a >
< virgula >
< id, b >
< virgula >
< id, c >
< virgula >
< id, maior >
< virgula >
< id, outro >
< ponto_virgula >
< do >
< print >
< abre_parent >
< literal, "A" >
< fecha_parent >
< ponto_virgula >
< scan >
< abre_parent >
< id, a >
< fecha_parent >
< ponto_virgula >
< print >
< abre_parent >
< literal, "B" >
< fecha_parent >
< ponto_virgula >
< scan >
< abre_parent >
< id, b >
< fecha_parent >
< ponto_virgula >
< print >
< abre_parent >
< literal, "C" >
< fecha_parent >
< ponto_virgula >
< scan >
< abre_parent >
< id, c >
< fecha_parent >
< ponto_virgula >
< if >
< abre_parent >
< abre_parent >
< id, a >
< greater_than >
< id, b >
< fecha_parent >
< and >
< abre_parent >
```

```
< id, a >
< greater_than >
< id, c >
< fecha_parent >
< fecha_parent >
< id, maior >
< assign >
< id, a >
< else >
< if >
< abre_parent >
< id, b >
< greater_than >
< id, c >
< fecha_parent >
< then >
< id, maior >
< assign >
< id, b >
< ponto_virgula >
< else >
< id, maior >
< assign >
< id, c >
< ponto_virgula >
< end >
< end >
< print >
< abre_parent >
< literal, "Maior valor:" >
< fecha_parent >
< ponto_virgula >
< print >
< abre_parent >
< id, maior >
< fecha_parent >
< ponto_virgula >
< print >
< abre_parent >
< literal, "Outro? " >
< fecha_parent >
< ponto_virgula >
< scan >
< abre_parent >
< id, outro >
< fecha_parent >
< ponto_virgula >
< while >
< abre_parent >
< id, outro >
< greater_equals >
< num, 0 >
< fecha_parent >
< end >

**** Tabela de símbolos ****
Entrada         |         Mais info
print
if
end
while
do
then
program
maior
string
int
outro
c
b
else
a
scan

Process finished with exit code 0
```

Teste 6

- Código Fonte

```
program

        Teste6
        */

        // comentario
        //
        int a, b, c, d;

        if ((c==a) | (c==b)) then
        print("Valor c: ", c)
        else
        if ( c<=d )
        print("Valor d d);
        end

        if (c-d != a) then
        print(!falseç)
        end

end
```

- Saída analisador léxico

```
**** Tokens lidos ****
< mult >
< div >
< int >
< id, a >
< virgula >
< id, b >
< virgula >
< id, c >
< virgula >
< id, d >
< ponto_virgula >
< if >
< abre_parent >
< abre_parent >
< id, c >
< equals >
< id, a >
< fecha_parent >
Error(7): Token '|' inválido
< abre_parent >
< id, c >
< equals >
< id, b >
< fecha_parent >
< fecha_parent >
< then >
< print >
< abre_parent >
< literal, "Valor c: " >
```

```
< virgula >
< id, c >
< fecha_parent >
< else >
< if >
< abre_parent >
< id, c >
< less_equals >
< id, d >
< fecha_parent >
< print >
< abre_parent >
Error(11): Token "" inválido
< end >
< if >
< abre_parent >
< id, c >
< menos >
< id, d >
< not_equals >
< id, a >
< fecha_parent >
< then >
< print >
< abre_parent >
< not >
< id, false >
Error(14): Token 'ç' inválido
< fecha_parent >
< end >
< end >

**** Tabela de símbolos ****
Entrada         |         Mais info
print
if
end
while
do
then
program
string
int
d
c
b
else
a
false
scan

Process finished with exit code 0
```

- Código Fonte - consertado

```
program

    /*
    Teste6
    */

    // comentario
    //
    int a, b, c, d;

    if ((c==a) || (c==b)) then
```

```
        print("Valor c: ", c)
        else
        if ( c<=d )
        print("Valor d" d);
        end

        if (c-d != a) then
        print(!false)
        end

end
```

- Saída do analisador léxico - código fonte consertado

```
**** Tokens lidos ****
< program >
< int >
< id, a >
< virgula >
< id, b >
< virgula >
< id, c >
< virgula >
< id, d >
< ponto_virgula >
< if >
< abre_parent >
< abre_parent >
< id, c >
< equals >
< id, a >
< fecha_parent >
< or >
< abre_parent >
< id, c >
< equals >
< id, b >
< fecha_parent >
< fecha_parent >
< then >
< print >
< abre_parent >
< literal, "Valor c: " >
< virgula >
< id, c >
< fecha_parent >
< else >
< if >
< abre_parent >
< id, c >
< less_equals >
< id, d >
< fecha_parent >
< print >
< abre_parent >
< literal, "Valor d" >
< id, d >
< fecha_parent >
< ponto_virgula >
< end >
< if >
< abre_parent >
< id, c >
< menos >
< id, d >
< not_equals >
< id, a >
< fecha_parent >
< then >
< print >
```

< abre_parent >
< not >
< id, false >
< fecha_parent >
< end >
< end >

**** Tabela de símbolos ****
Entrada | Mais info
print
if
end
while
do
then
program
string
int
d
c
b
else
a
false
scan

Process finished with exit code 0