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 Sintático

Gabriel Pires de Miranda Magalhães

Renan Mateus Bernardo do Nascimento

Vinícius Magalhães D'Assunção

Belo Horizonte, CEFET-MG Campus II

Novembro de 2017

Instruções de execução do analisador sintático

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

java -jar analisador_sintatico.jar

Ao executar será exibida uma mensagem solicitando que seja colocado o número do teste que será analisado pelo analisador sintático. Os testes possuem a seguinte relação na pasta testes/:

Código fonte	Código fonte consertado
teste1.txt	teste7.txt
teste2.txt	teste8.txt
teste3.txt	teste9.txt
teste4.txt	teste10.txt
teste5.txt	teste11.txt
teste6.txt	teste12.txt

Modificações realizadas na gramática

Primeiramente foram removidas as produções que geravam um token: *letter*; *digit*; *caractere*. As produções correspondentes à um token tornaram-se terminais. É o caso de *identifier*, *literal* e *integer_const*.

A produção *condition* foi removida pois só gerava expression, sendo trocada pela mesma em outras produções que a utilizavam.

A produção *writable* foi removida pois a mesma possuía dois caminhos que levavam à literal. Portanto, writable foi substituído por *simple-expr*.

- writable -> simple-expr -> term -> factor-a -> factor -> constant -> literal
- writable -> literal

A seguir foi removida a recursão à esquerda da gramática para que a mesma se tornasse uma gramática LL1.

Por último as produçẽs com [] ou {} foram substituídas de forma a utilizar λ.

A seguir é apresentada a nova gramática:

```
program          ::= program decl-list stmt-list end
decl-list        ::= decl decl-list | λ
decl             ::= type ident-list ";"
ident-list       ::= identifier ident-list'
ident-list'      ::= "," identifier ident-list' | λ
type             ::= int | string
stmt-list        ::= stmt stmt-list'
stmt-list'       ::= stmt stmt-list' | λ
stmt             ::= assign-stmt ";"  | if-stmt  |  while-stmt | read-stmt ";"  |  write-stmt ";"
assign-stmt      ::= identifier "=" simple_expr
if-stmt          ::= if  expression  then  stmt-list  if-stmt' end
if-stmt'         ::=       else stmt-list | λ
while-stmt       ::= do stmt-list stmt-sufix
stmt-sufix       ::= while expression end
read-stmt        ::= scan  "("  identifier  ")"
write-stmt       ::= print  "("  simple-expr  ")"
expression       ::= simple-expr  expression'
expression'      ::= relop  simple-expr         |         λ
simple-expr      ::= term  simple-expr'
simple-expr'     ::= addop term simple-expr'  | λ
term             ::= factor-a  term'
term'            ::= mulop factor-a term' | λ
factor-a         ::= factor  |  !  factor  |  "-"  factor
factor           ::= identifier  |  constant  |  "("  expression  ")"
relop            ::= "==" |  ">" | "<" | "!=" | ">=" | "<="
addop            ::= "+" |  "-"  |  "||"
mulop            ::= "*"  |  "/"  |  "&&"
constant         ::= integer_const  |  literal
```

Implementação do analisador sintático

A partir da nova gramática foi feita a tabela de FIRST e FOLLOW:

	FIRST	**FOLLOW**
program	program	$
decl-list	int, string	identifier, if, do, scan, print
decl	int, string	int, string, identifier, if, do, scan, print
ident-list	identifier	';'
ident-list'	λ,','	';'
type	int, string	identifier

stmt-list	identifier, if, do, scan, print	end, else, while
stmt-list'	identifier, if, do, scan, print, λ	end, else, while
stmt	identifier, if, do, scan, print	identifier, if, do, scan, print, end, else, while
assign-stmt	identifier	';'
if-stmt	if	identifier, if, do, scan, print, end, else, while
If-stmt'	else, λ	end
while-stmt	do	identifier, if, do, scan, print, end, else, while
stmt-sufix	while	identifier, if, do, scan, print, end, else, while
read-stmt	scan	';'
write-stmt	print	';'
expression	identifier, integer_const, literal, '(', '!', '-'	then, end, ')'
expression'	==', '>', '<', '>=', '<=', '!='	then, end, ')'
simple-expr	identifier, integer_const, literal, '(', '!', '-'	then, end, ')', '==', '>', '<', '>=', '<=', '!=', ';'
simple-expr'	'+', '-', '||', λ	then, end, ')', '==', '>', '<', '>=', '<=', '!=', ';'
term	identifier, integer_const, literal, '(', '!', '-'	'+', '-', '||', then, end, ')', '==', '>', '<', '>=', '<=', '!=', ';'
term'	'*', '/', '&&', λ	'+', '-', '||', then, end, ')', '==', '>', '<', '>=', '<=', '!=', ';'
factor-a	identifier, integer_const, literal, '(', '!', '-'	'*', '/', '&&', '+', '-', '||', then, end, ')', '==', '>', '<', '>=', '<=', '!=', ';'

factor	identifier, integer_const, literal, ‘(‘	'*’, ‘/’, ‘&&’, '+’, ‘-’, ‘||’, then, end, ')', '==’, ‘>’, ‘<’, ‘>=’, ‘<=’, ‘!=’, ';'
relop	'==’, ‘>’, ‘<’, ‘>=’, ‘<=’, ‘!=’	identifier, integer_const, literal, ‘(‘, ‘!’, ‘-’
addop	'+’, ‘-’, ‘||’	identifier, integer_const, literal, ‘(‘, ‘!’, ‘-’
mulop	'*’, ‘/’, ‘&&’	identifier, integer_const, literal, ‘(‘, ‘!’, ‘-’
constant	integer_const, literal	'*’, ‘/’, ‘&&’, '+’, ‘-’, ‘||’, then, end, ')', '==’, ‘>’, ‘<’, ‘>=’, ‘<=’, ‘!=’, ';'

A partir da tabela de FIRST e FOLLOW foi feita a tabela do parser. Que está dentro do arquivo zip pois não coube no relatório.

Foi implementado um método para cada produção da gramática. Com o auxílio da tabela do parser, foi definido quais métodos deveriam ser chamados ao ler um token específico.

Resultados dos testes

Para realizar os testes o código fonte utilizado foi aquele que não apresentou erro durante a análise léxica. Caso ocorra erro durante a análise sintática, o código fonte é consertado e o processo de análise é feito novamente. A saída apresenta o seguinte formato:

- Caso sucesso
  Token Consumido(*linha***):** *token*

- Caso insucesso
  Error(*linha***): Token não esperado:** *token*
  Fim de arquivo inesperado.

- **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
```

Resultado da execução

```
**** Inicio Parser ****
Token Consumido(1): < program >
Token Consumido(2): < int >
Token Consumido(2): < id, a >
Token Consumido(2): < virgula >
Token Consumido(2): < id, b >
Token Consumido(2): < ponto_virgula >
Token Consumido(3): < int >
Token Consumido(3): < id, result >
Token Consumido(3): < ponto_virgula >
Token Consumido(4): < id, float >
Error(4): Token não esperado:< id, a >
Fim de arquivo inesperado.
```

Código fonte consertado

```
program
        int a, b;
        int result;
        string 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
```

Resultado da execução

```
**** Inicio Parser ****
Token Consumido(1): < program >
Token Consumido(2): < int >
Token Consumido(2): < id, a >
Token Consumido(2): < virgula >
Token Consumido(2): < id, b >
Token Consumido(2): < ponto_virgula >
Token Consumido(3): < int >
Token Consumido(3): < id, result >
Token Consumido(3): < ponto_virgula >
Token Consumido(4): < string >
Token Consumido(4): < id, a >
Token Consumido(4): < virgula >
Token Consumido(4): < id, x >
Token Consumido(4): < virgula >
Token Consumido(4): < id, total >
Token Consumido(4): < ponto_virgula >
Token Consumido(6): < id, a >
Token Consumido(6): < assign >
Token Consumido(6): < num, 2 >
Token Consumido(6): < ponto_virgula >
Token Consumido(7): < id, x >
Token Consumido(7): < assign >
Token Consumido(7): < num, 1 >
Token Consumido(7): < ponto_virgula >
Token Consumido(8): < scan >
Token Consumido(8): < abre_parent >
Token Consumido(8): < id, b >
Token Consumido(8): < fecha_parent >
Token Consumido(8): < ponto_virgula >
Token Consumido(9): < scan >
Token Consumido(9): < abre_parent >
Token Consumido(9): < id, y >
Token Consumido(9): < fecha_parent >
Token Consumido(9): < ponto_virgula >
Token Consumido(10): < id, result >
Token Consumido(10): < assign >
Token Consumido(10): < abre_parent >
Token Consumido(10): < id, a >
Token Consumido(10): < mult >
Token Consumido(10): < id, b >
Token Consumido(10): < soma >
Token Consumido(10): < num, 1 >
Token Consumido(10): < fecha_parent >
Token Consumido(10): < div >
Token Consumido(10): < num, 2 >
Token Consumido(10): < ponto_virgula >
Token Consumido(11): < print >
Token Consumido(11): < abre_parent >
Token Consumido(11): < literal, "Resultado:
" >
Token Consumido(11): < fecha_parent >
Token Consumido(11): < ponto_virgula >
Token Consumido(12): < print >
Token Consumido(12): < abre_parent >
Token Consumido(12): < id, result >
Token Consumido(12): < fecha_parent >
Token Consumido(12): < ponto_virgula >
Token Consumido(13): < print >
Token Consumido(13): < abre_parent >
Token Consumido(13): < literal, "Total: " >
Token Consumido(13): < fecha_parent >
Token Consumido(13): < ponto_virgula >
Token Consumido(14): < id, total >
Token Consumido(14): < assign >
Token Consumido(14): < id, y >
Token Consumido(14): < div >
Token Consumido(14): < id, x >
Token Consumido(14): < ponto_virgula >
Token Consumido(15): < print >
Token Consumido(15): < abre_parent >
Token Consumido(15): < literal, "Total: " >
Token Consumido(15): < fecha_parent >
Token Consumido(15): < ponto_virgula >
Token Consumido(16): < print >
Token Consumido(16): < abre_parent >
Token Consumido(16): < id, total >
Token Consumido(16): < fecha_parent >
Token Consumido(16): < ponto_virgula >
Token Consumido(17): < end >
```

- **Teste 2**

Código fonte

```
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);
```

Resultado da execução

```
**** Inicio Parser ****
Token Consumido(1): < program >
Token Consumido(3): < int >
Token Consumido(3): < id, a >
Token Consumido(3): < virgula >
Token Consumido(3): < id, c >
Token Consumido(3): < ponto_virgula >
Token Consumido(4): < id, float >
Error(4): Token não esperado:< id, d >
Fim de arquivo inesperado.
```

Código fonte consertado

```
program

        int a, c;
        int d, e;

        a = 0; d = 3;
        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);
end
```

Resultado da execução

```
**** Inicio Parser ****
Token Consumido(1): < program >
Token Consumido(3): < int >
Token Consumido(3): < id, a >
Token Consumido(3): < virgula >
Token Consumido(3): < id, c >
Token Consumido(3): < ponto_virgula >
Token Consumido(4): < int >
Token Consumido(4): < id, d >
Token Consumido(4): < virgula >
Token Consumido(4): < id, e >
Token Consumido(4): < ponto_virgula >
Token Consumido(6): < id, a >
Token Consumido(6): < assign >
Token Consumido(6): < num, 0 >
Token Consumido(6): < ponto_virgula >
Token Consumido(6): < id, d >
Token Consumido(6): < assign >
Token Consumido(6): < num, 3 >
Token Consumido(6): < ponto_virgula >
Token Consumido(7): < id, c >
Token Consumido(7): < assign >
Token Consumido(7): < id, d >
Token Consumido(7): < div >
Token Consumido(7): < num, 12 >
Token Consumido(7): < ponto_virgula >
Token Consumido(9): < scan >
Token Consumido(9): < abre_parent >
Token Consumido(9): < id, a >
Token Consumido(9): < fecha_parent >
Token Consumido(9): < ponto_virgula >
Token Consumido(10): < scan >
Token Consumido(10): < abre_parent >
Token Consumido(10): < id, c >
Token Consumido(10): < fecha_parent >
Token Consumido(10): < ponto_virgula >
Token Consumido(11): < id, b >
Token Consumido(11): < assign >
Token Consumido(11): < id, a >
Token Consumido(11): < mult >
Token Consumido(11): < id, a >
Token Consumido(11): < ponto_virgula >
Token Consumido(12): < id, c >
Token Consumido(12): < assign >
Token Consumido(12): < id, b >
Token Consumido(12): < soma >
Token Consumido(12): < id, a >
Token Consumido(12): < mult >
Token Consumido(12): < fecha_parent >
Token Consumido(12): < ponto_virgula >
Token Consumido(13): < print >
Token Consumido(13): < abre_parent >
Token Consumido(13): < literal, "Resultado:
" >
Token Consumido(13): < fecha_parent >
Token Consumido(13): < ponto_virgula >
Token Consumido(14): < print >
Token Consumido(14): < abre_parent >
Token Consumido(14): < id, c >
Token Consumido(14): < fecha_parent >
Token Consumido(14): < ponto_virgula >
Token Consumido(15): < id, a >
Token Consumido(15): < assign >
Token Consumido(15): < abre_parent >
Token Consumido(15): < id, b >
Token Consumido(15): < soma >
Token Consumido(15): < id, c >
Token Consumido(15): < soma >
Token Consumido(15): < id, d >
Token Consumido(15): < fecha_parent >
Token Consumido(15): < div >
Token Consumido(15): < num, 2 >
Token Consumido(15): < ponto_virgula >
Token Consumido(16): < id, e >
Token Consumido(16): < assign >
Token Consumido(16): < id, val >
Token Consumido(16): < soma >
Token Consumido(16): < id, c >
Token Consumido(16): < soma >
Token Consumido(16): < id, a >
Token Consumido(16): < ponto_virgula >
Token Consumido(17): < print >
Token Consumido(17): < abre_parent >
Token Consumido(17): < literal, "E: " >
Token Consumido(17): < fecha_parent >
Token Consumido(17): < ponto_virgula >
Token Consumido(18): < print >
Token Consumido(18): < abre_parent >
Token Consumido(18): < id, e >
Token Consumido(18): < fecha_parent >
Token Consumido(18): < ponto_virgula >
Token Consumido(19): < end >
```

```
Token Consumido(12): < abre_parent >
Token Consumido(12): < num, 1 >
Token Consumido(12): < soma >
Token Consumido(12): < id, a >
Token Consumido(12): < mult >
Token Consumido(12): < id, c >
```

- **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
*/
```

Resultado da execução

```
**** Inicio Parser ****
Token Consumido(1): < program >
Token Consumido(2): < int >
Token Consumido(2): < id, pontuacao >
Token Consumido(2): < virgula >
Token Consumido(2): < id,
pontuacaoMaxima >
Token Consumido(2): < virgula >
Token Consumido(2): < id, disponibilidade
>
Token Consumido(2): < ponto_virgula >
Token Consumido(3): < string >
Token Consumido(5): < literal, "Sim" >
Token Consumido(5): < ponto_virgula >
Token Consumido(6): < id,
pontuacaoMinima >
Token Consumido(6): < assign >
Token Consumido(6): < num, 50 >
Token Consumido(6): < ponto_virgula >
Token Consumido(7): < id,
pontuacaoMaxima >
Token Consumido(7): < assign >
Token Consumido(7): < num, 100 >
Token Consumido(7): < ponto_virgula >
```

Token Consumido(3): < id, pontuacaoMinima > Token Consumido(3): < ponto_virgula > Token Consumido(5): < id, disponibilidade > Token Consumido(5): < assign >	Fim de arquivo inesperado.

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 (( pontuacao > pontuacaoMinima) && (disponibilidade=="Sim")) then
			print("Candidato aprovado");
		else
			print("Candidato reprovado");
		end
	while (pontuacao >= 0)end
end
```

Resultado da execução

**** Inicio Parser **** Token Consumido(1): < program > Token Consumido(2): < int > Token Consumido(2): < id, pontuacao > Token Consumido(2): < virgula > Token Consumido(2): < id, pontuacaoMaxima > Token Consumido(2): < virgula > Token Consumido(2): < id, disponibilidade > Token Consumido(2): < ponto_virgula > Token Consumido(3): < string > Token Consumido(3): < id, pontuacaoMinima >	Token Consumido(15): < scan > Token Consumido(15): < abre_parent > Token Consumido(15): < id, disponibilidade > Token Consumido(15): < fecha_parent > Token Consumido(15): < ponto_virgula > Token Consumido(17): < if > Token Consumido(17): < abre_parent > Token Consumido(17): < abre_parent > Token Consumido(17): < id, pontuacao > Token Consumido(17): < greater_than > Token Consumido(17): < id, pontuacaoMinima > Token Consumido(17): < fecha_parent >

```
Token Consumido(3): < ponto_virgula >
Token Consumido(5): < id, disponibilidade >
Token Consumido(5): < assign >
Token Consumido(5): < literal, "Sim" >
Token Consumido(5): < ponto_virgula >
Token Consumido(6): < id, pontuacaoMinima >
Token Consumido(6): < assign >
Token Consumido(6): < num, 50 >
Token Consumido(6): < ponto_virgula >
Token Consumido(7): < id, pontuacaoMaxima >
Token Consumido(7): < assign >
Token Consumido(7): < num, 100 >
Token Consumido(7): < ponto_virgula >
Token Consumido(11): < do >
Token Consumido(12): < print >
Token Consumido(12): < abre_parent >
Token Consumido(12): < literal, "Pontuacao Candidato: " >
Token Consumido(12): < fecha_parent >
Token Consumido(12): < ponto_virgula >
Token Consumido(13): < scan >
Token Consumido(13): < abre_parent >
Token Consumido(13): < id, pontuacao >
Token Consumido(13): < fecha_parent >
Token Consumido(13): < ponto_virgula >
Token Consumido(14): < print >
Token Consumido(14): < abre_parent >
Token Consumido(14): < literal, "Disponibilidade Candidato: " >
Token Consumido(14): < fecha_parent >
Token Consumido(14): < ponto_virgula >
```

```
Token Consumido(17): < and >
Token Consumido(17): < abre_parent >
Token Consumido(17): < id, disponibilidade >
Token Consumido(17): < equals >
Token Consumido(17): < literal, "Sim" >
Token Consumido(17): < fecha_parent >
Token Consumido(17): < fecha_parent >
Token Consumido(17): < then >
Token Consumido(18): < print >
Token Consumido(18): < abre_parent >
Token Consumido(18): < literal, "Candidato aprovado" >
Token Consumido(18): < fecha_parent >
Token Consumido(18): < ponto_virgula >
Token Consumido(19): < else >
Token Consumido(20): < print >
Token Consumido(20): < abre_parent >
Token Consumido(20): < literal, "Candidato reprovado" >
Token Consumido(20): < fecha_parent >
Token Consumido(20): < ponto_virgula >
Token Consumido(21): < end >
Token Consumido(22): < while >
Token Consumido(22): < abre_parent >
Token Consumido(22): < id, pontuacao >
Token Consumido(22): < greater_equals >
Token Consumido(22): < num, 0 >
Token Consumido(22): < fecha_parent >
Token Consumido(22): < end >
Token Consumido(23): < end >
```

- **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
```

Resultado da execução

```
**** Inicio Parser ****
Error(1): Token não esperado:< int >
```

Código fonte consertado

```
program
        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);
end
```

Resultado da execução

```
**** Inicio Parser ****
Token Consumido(1): < program >
Token Consumido(2): < int >
Token Consumido(2): < id, a >
Token Consumido(10): < num, 1 >
Token Consumido(10): < ponto_virgula >
Token Consumido(11): < print >
Token Consumido(11): < abre_parent >
```

```
Token Consumido(2): < virgula >
Token Consumido(2): < id, aux >
Token Consumido(2): < virgula >
Token Consumido(2): < id, b >
Token Consumido(2): < ponto_virgula >
Token Consumido(3): < string >
Token Consumido(3): < id, nome >
Token Consumido(3): < virgula >
Token Consumido(3): < id, sobrenome >
Token Consumido(3): < virgula >
Token Consumido(3): < id, msg >
Token Consumido(3): < ponto_virgula >
Token Consumido(5): < print >
Token Consumido(5): < abre_parent >
Token Consumido(5): < id, Nome >
Token Consumido(5): < fecha_parent >
Token Consumido(5): < ponto_virgula >
Token Consumido(6): < scan >
Token Consumido(6): < abre_parent >
Token Consumido(6): < id, nome >
Token Consumido(6): < fecha_parent >
Token Consumido(6): < ponto_virgula >
Token Consumido(7): < print >
Token Consumido(7): < abre_parent >
Token Consumido(7): < literal,
"Sobrenome: " >
Token Consumido(7): < fecha_parent >
Token Consumido(7): < ponto_virgula >
Token Consumido(8): < scan >
Token Consumido(8): < abre_parent >
Token Consumido(8): < id, sobrenome >
Token Consumido(8): < fecha_parent >
Token Consumido(8): < ponto_virgula >
Token Consumido(9): < id, msg >
Token Consumido(9): < assign >
Token Consumido(9): < literal, "Ola, " >
Token Consumido(9): < soma >
Token Consumido(9): < id, nome >
Token Consumido(9): < soma >
Token Consumido(9): < literal, " " >
Token Consumido(9): < soma >
Token Consumido(9): < id, sobrenome >
Token Consumido(9): < soma >
Token Consumido(9): < literal, "!" >
Token Consumido(9): < ponto_virgula >
Token Consumido(10): < id, msg >
Token Consumido(10): < assign >
Token Consumido(10): < id, msg >
Token Consumido(10): < soma >
Token Consumido(11): < id, msg >
Token Consumido(11): < fecha_parent >
Token Consumido(11): < ponto_virgula >
Token Consumido(13): < scan >
Token Consumido(13): < abre_parent >
Token Consumido(13): < id, a >
Token Consumido(13): < fecha_parent >
Token Consumido(13): < ponto_virgula >
Token Consumido(14): < scan >
Token Consumido(14): < abre_parent >
Token Consumido(14): < id, b >
Token Consumido(14): < fecha_parent >
Token Consumido(14): < ponto_virgula >
Token Consumido(15): < if >
Token Consumido(15): < abre_parent >
Token Consumido(15): < id, a >
Token Consumido(15): < greater_than >
Token Consumido(15): < id, b >
Token Consumido(15): < fecha_parent >
Token Consumido(15): < then >
Token Consumido(16): < id, aux >
Token Consumido(16): < assign >
Token Consumido(16): < id, b >
Token Consumido(16): < ponto_virgula >
Token Consumido(17): < id, b >
Token Consumido(17): < assign >
Token Consumido(17): < id, a >
Token Consumido(17): < ponto_virgula >
Token Consumido(18): < id, a >
Token Consumido(18): < assign >
Token Consumido(18): < id, aux >
Token Consumido(18): < ponto_virgula >
Token Consumido(19): < end >
Token Consumido(20): < print >
Token Consumido(20): < abre_parent >
Token Consumido(20): < literal, "Apos a
troca: " >
Token Consumido(20): < fecha_parent >
Token Consumido(20): < ponto_virgula >
Token Consumido(21): < print >
Token Consumido(21): < abre_parent >
Token Consumido(21): < id, a >
Token Consumido(21): < fecha_parent >
Token Consumido(21): < ponto_virgula >
Token Consumido(22): < print >
Token Consumido(22): < abre_parent >
Token Consumido(22): < id, b >
Token Consumido(22): < fecha_parent >
Token Consumido(22): < ponto_virgula >
Token Consumido(23): < 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
```

Resultado da execução

```
**** Inicio Parser ****
Token Consumido(1): < program >
Token Consumido(2): < int >
Token Consumido(2): < id, a >
Token Consumido(2): < virgula >
Token Consumido(2): < id, b >
Token Consumido(2): < virgula >
Token Consumido(2): < id, c >
Token Consumido(2): < virgula >
Token Consumido(2): < id, maior >
Token Consumido(2): < virgula >
Token Consumido(8): < abre_parent >
Token Consumido(8): < id, b >
Token Consumido(8): < fecha_parent >
Token Consumido(8): < ponto_virgula >
Token Consumido(9): < print >
Token Consumido(9): < abre_parent >
Token Consumido(9): < literal, "C" >
Token Consumido(9): < fecha_parent >
Token Consumido(9): < ponto_virgula >
Token Consumido(10): < scan >
Token Consumido(10): < abre_parent >
```

```
Token Consumido(2): < id, outro >
Token Consumido(2): < ponto_virgula >
Token Consumido(4): < do >
Token Consumido(5): < print >
Token Consumido(5): < abre_parent >
Token Consumido(5): < literal, "A" >
Token Consumido(5): < fecha_parent >
Token Consumido(5): < ponto_virgula >
Token Consumido(6): < scan >
Token Consumido(6): < abre_parent >
Token Consumido(6): < id, a >
Token Consumido(6): < fecha_parent >
Token Consumido(6): < ponto_virgula >
Token Consumido(7): < print >
Token Consumido(7): < abre_parent >
Token Consumido(7): < literal, "B" >
Token Consumido(7): < fecha_parent >
Token Consumido(7): < ponto_virgula >
Token Consumido(8): < scan >
```

```
Token Consumido(10): < id, c >
Token Consumido(10): < fecha_parent >
Token Consumido(10): < ponto_virgula >
Token Consumido(13): < if >
Token Consumido(13): < abre_parent >
Token Consumido(13): < abre_parent >
Token Consumido(13): < id, a >
Token Consumido(13): < greater_than >
Token Consumido(13): < id, b >
Token Consumido(13): < fecha_parent >
Token Consumido(13): < and >
Token Consumido(13): < abre_parent >
Token Consumido(13): < id, a >
Token Consumido(13): < greater_than >
Token Consumido(13): < id, c >
Token Consumido(13): < fecha_parent >
Token Consumido(13): < fecha_parent >
Error(14): Token não esperado:< id, maior
>
Fim de arquivo inesperado.
```

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) ) then
			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
end
```

Resultado da execução

```
**** Inicio Parser ****
Token Consumido(1): < program >
Token Consumido(2): < int >
Token Consumido(2): < id, a >
Token Consumido(2): < virgula >
Token Consumido(2): < id, b >
Token Consumido(2): < virgula >
Token Consumido(2): < id, c >
Token Consumido(2): < virgula >
Token Consumido(2): < id, maior >
Token Consumido(2): < virgula >
Token Consumido(2): < id, outro >
Token Consumido(2): < ponto_virgula >
Token Consumido(4): < do >
Token Consumido(5): < print >
Token Consumido(5): < abre_parent >
Token Consumido(5): < literal, "A" >
Token Consumido(5): < fecha_parent >
Token Consumido(5): < ponto_virgula >
Token Consumido(6): < scan >
Token Consumido(6): < abre_parent >
Token Consumido(6): < id, a >
Token Consumido(6): < fecha_parent >
Token Consumido(6): < ponto_virgula >
Token Consumido(7): < print >
Token Consumido(7): < abre_parent >
Token Consumido(7): < literal, "B" >
Token Consumido(7): < fecha_parent >
Token Consumido(7): < ponto_virgula >
Token Consumido(8): < scan >
Token Consumido(8): < abre_parent >
Token Consumido(8): < id, b >
Token Consumido(8): < fecha_parent >
Token Consumido(8): < ponto_virgula >
Token Consumido(9): < print >
Token Consumido(9): < abre_parent >
Token Consumido(9): < literal, "C" >
Token Consumido(9): < fecha_parent >
Token Consumido(9): < ponto_virgula >
Token Consumido(10): < scan >
Token Consumido(10): < abre_parent >
Token Consumido(10): < id, c >
Token Consumido(10): < fecha_parent >
Token Consumido(10): < ponto_virgula >
Token Consumido(13): < if >
Token Consumido(13): < fecha_parent >
Token Consumido(13): < then >
Token Consumido(14): < id, maior >
Token Consumido(14): < assign >
Token Consumido(14): < id, a >
Token Consumido(14): < ponto_virgula >
Token Consumido(16): < else >
Token Consumido(17): < if >
Token Consumido(17): < abre_parent >
Token Consumido(17): < id, b >
Token Consumido(17): < greater_than >
Token Consumido(17): < id, c >
Token Consumido(17): < fecha_parent >
Token Consumido(17): < then >
Token Consumido(18): < id, maior >
Token Consumido(18): < assign >
Token Consumido(18): < id, b >
Token Consumido(18): < ponto_virgula >
Token Consumido(20): < else >
Token Consumido(21): < id, maior >
Token Consumido(21): < assign >
Token Consumido(21): < id, c >
Token Consumido(21): < ponto_virgula >
Token Consumido(22): < end >
Token Consumido(23): < end >
Token Consumido(24): < print >
Token Consumido(24): < abre_parent >
Token Consumido(24): < literal, "Maior
valor:" >
Token Consumido(24): < fecha_parent >
Token Consumido(24): < ponto_virgula >
Token Consumido(25): < print >
Token Consumido(25): < abre_parent >
Token Consumido(25): < id, maior >
Token Consumido(25): < fecha_parent >
Token Consumido(25): < ponto_virgula >
Token Consumido(26): < print >
Token Consumido(26): < abre_parent >
Token Consumido(26): < literal, "Outro? " >
Token Consumido(26): < fecha_parent >
Token Consumido(26): < ponto_virgula >
Token Consumido(27): < scan >
Token Consumido(27): < abre_parent >
Token Consumido(27): < id, outro >
Token Consumido(27): < fecha_parent >
```

```
Token Consumido(13): < abre_parent >
Token Consumido(13): < abre_parent >
Token Consumido(13): < id, a >
Token Consumido(13): < greater_than >
Token Consumido(13): < id, b >
Token Consumido(13): < fecha_parent >
Token Consumido(13): < and >
Token Consumido(13): < abre_parent >
Token Consumido(13): < id, a >
Token Consumido(13): < greater_than >
Token Consumido(13): < id, c >
Token Consumido(13): < fecha_parent >
Token Consumido(27): < ponto_virgula >
Token Consumido(28): < while >
Token Consumido(28): < abre_parent >
Token Consumido(28): < id, outro >
Token Consumido(28): < greater_equals >
Token Consumido(28): < num, 0 >
Token Consumido(28): < fecha_parent >
Token Consumido(28): < end >
Token Consumido(29): < end >
```

- **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 ) then
         print("Valor d" + d);
   end

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

end
```

Resultado da execução

```
**** Inicio Parser ****
Token Consumido(1): < program >
Token Consumido(9): < int >
Token Consumido(9): < id, a >
Token Consumido(9): < virgula >
Token Consumido(9): < id, b >
Token Consumido(9): < virgula >
Token Consumido(9): < id, c >
Token Consumido(14): < less_equals >
Token Consumido(14): < id, d >
Token Consumido(14): < fecha_parent >
Token Consumido(14): < then >
Token Consumido(15): < print >
Token Consumido(15): < abre_parent >
Token Consumido(15): < literal, "Valor d" >
Token Consumido(15): < soma >
```

```
Token Consumido(9): < virgula >
Token Consumido(9): < id, d >
Token Consumido(9): < ponto_virgula >
Token Consumido(11): < if >
Token Consumido(11): < abre_parent >
Token Consumido(11): < abre_parent >
Token Consumido(11): < id, c >
Token Consumido(11): < equals >
Token Consumido(11): < id, a >
Token Consumido(11): < fecha_parent >
Token Consumido(11): < or >
Token Consumido(11): < abre_parent >
Token Consumido(11): < id, c >
Token Consumido(11): < equals >
Token Consumido(11): < id, b >
Token Consumido(11): < fecha_parent >
Token Consumido(11): < fecha_parent >
Token Consumido(11): < then >
Token Consumido(12): < print >
Token Consumido(12): < abre_parent >
Token Consumido(12): < literal, "Valor c: " >
Token Consumido(12): < soma >
Token Consumido(12): < id, c >
Token Consumido(12): < fecha_parent >
Token Consumido(12): < ponto_virgula >
Token Consumido(13): < else >
Token Consumido(14): < if >
Token Consumido(14): < abre_parent >
Token Consumido(14): < id, c >
Token Consumido(15): < id, d >
Token Consumido(15): < fecha_parent >
Token Consumido(15): < ponto_virgula >
Token Consumido(16): < end >
Token Consumido(18): < if >
Token Consumido(18): < abre_parent >
Token Consumido(18): < id, c >
Token Consumido(18): < menos >
Token Consumido(18): < id, d >
Token Consumido(18): < not_equals >
Token Consumido(18): < id, a >
Token Consumido(18): < fecha_parent >
Token Consumido(18): < then >
Token Consumido(19): < print >
Token Consumido(19): < abre_parent >
Token Consumido(19): < not >
Token Consumido(19): < id, false >
Token Consumido(19): < fecha_parent >
Token Consumido(19): < ponto_virgula >
Token Consumido(20): < end >
Token Consumido(22): < end >
Fim de arquivo inesperado.
```

Código fonte consertado

```
program

   /*
      Teste6
   */

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

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

```
        end
    end

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

end
```

Resultado da execução

```
**** Inicio Parser ****
Token Consumido(1): < program >
Token Consumido(9): < int >
Token Consumido(9): < id, a >
Token Consumido(9): < virgula >
Token Consumido(9): < id, b >
Token Consumido(9): < virgula >
Token Consumido(9): < id, c >
Token Consumido(9): < virgula >
Token Consumido(9): < id, d >
Token Consumido(9): < ponto_virgula >
Token Consumido(11): < if >
Token Consumido(11): < abre_parent >
Token Consumido(11): < abre_parent >
Token Consumido(11): < id, c >
Token Consumido(11): < equals >
Token Consumido(11): < id, a >
Token Consumido(11): < fecha_parent >
Token Consumido(11): < or >
Token Consumido(11): < abre_parent >
Token Consumido(11): < id, c >
Token Consumido(11): < equals >
Token Consumido(11): < id, b >
Token Consumido(11): < fecha_parent >
Token Consumido(11): < fecha_parent >
Token Consumido(11): < then >
Token Consumido(12): < print >
Token Consumido(12): < abre_parent >
Token Consumido(12): < literal, "Valor c: " >
Token Consumido(12): < fecha_parent >
Token Consumido(12): < ponto_virgula >
Token Consumido(13): < print >
Token Consumido(13): < abre_parent >
Token Consumido(13): < id, c >
Token Consumido(13): < fecha_parent >
Token Consumido(13): < ponto_virgula >
Token Consumido(14): < else >
Token Consumido(15): < if >
Token Consumido(15): < abre_parent >
Token Consumido(15): < id, c >
Token Consumido(15): < less_equals >
Token Consumido(15): < id, d >
Token Consumido(15): < fecha_parent >
Token Consumido(15): < then >
Token Consumido(16): < print >
Token Consumido(16): < abre_parent >
Token Consumido(16): < literal, "Valor d" >
Token Consumido(16): < fecha_parent >
Token Consumido(16): < ponto_virgula >
Token Consumido(17): < print >
Token Consumido(17): < abre_parent >
Token Consumido(17): < id, d >
Token Consumido(17): < fecha_parent >
Token Consumido(17): < ponto_virgula >
Token Consumido(18): < end >
Token Consumido(19): < end >
Token Consumido(21): < if >
Token Consumido(21): < abre_parent >
Token Consumido(21): < id, c >
Token Consumido(21): < menos >
Token Consumido(21): < id, d >
Token Consumido(21): < not_equals >
Token Consumido(21): < id, a >
Token Consumido(21): < fecha_parent >
Token Consumido(21): < then >
Token Consumido(22): < print >
Token Consumido(22): < abre_parent >
Token Consumido(22): < not >
Token Consumido(22): < id, false >
Token Consumido(22): < fecha_parent >
Token Consumido(22): < ponto_virgula >
Token Consumido(23): < end >
Token Consumido(25): < end >
```