Trabalho prático de Compiladores

Análise léxica

Douglas Cristiano Alves
2004041166
dcalves@dcc.ufmg.br
Professora Mariza Bigonha

6 de maio de 2009

Sumário

Resumo

Análise léxica é o processo de analisar a entrada de linhas de caracteres (tal como o código-fonte de um programa de computador) e produzir uma seqüência de símbolos chamado "símbolos léxicos"(lexical tokens), ou somente "símbolos"(tokens), que podem ser manipulados mais facilmente por um parser (leitor de saída). O presente trabalho tem como objetivo analisar e produzir um analizador léxico.

1 Introdução

A análise léxica é a primeira fase de um compilador. Sua tarefa principal é a de ler os caracteres de entrada e produzir uma sequência de tokens que o parser utiliza para a análise sintática. Como o analisador léxico é a parte do compilador que lê o texto-fonte, também pode realizar algumas tarefas secundárias ao nível da interface com o usuário. Uma delas é a de remover do programa-fonte os comentários e espaços em branco. Uma outra é a de correlacionar as mensagens de erro do compilador com o programa-fonte

Assim segue-se o analisador léxico dada a definição da linguagem IO.

2 Analisador léxico

2.1 Visão geral

3 Desenvolvimento

3.1 Decisões de implementação

Para o presente trabalho, assim como os anteriores, foi utilizado a linguagem JAVA. Como recurso adicional, particularmente para a implementação do analisador léxico foi utilizado uma ferramenta auxiliar responsável pela geração do analisador léxico através das definições da linguagem: o jflex.

3.2 JFlex

Para a geração do analisador léxico, foi utilizada a ferramenta jflex. Ela é um software livre contruído em JAVA que recebe com o entrada um arquivo ".flex"com a descrição das expressões regulares para o reconhecimento dos tokens e assim eles serão verificados.

O arquivo .flex criado contém basicamente duas seções: uma cmo as definições das expressões regulares referentes a cada token e outra com as ações que devem ser tomadas quando reconhecidos. Abaixo segue um exemplo da definição de uma expressão regular:

```
letter = [A-Za-z]
digit = [0-9]
identifier = {letter}({letter}|{digit})*
```

E assim definimos expressões regulares para uma letra, dígito e identificador. Sendo esses letras de "a"minúsculo a "Z"maiúsculo, números de "0"a "9"e combinações de letras e dígitos, respectivamente.

Na segunda parte do arquivo são definidas as ações que devem ser tomadas quando há o reconhecimento de tokens. Por exemplo:

```
"program" { return (new Token("PROGRAM", yytext(), yyline, yycolumn)); }
```

No exemplo, quando é lido o lexema "program", a ação a ser tomada é criar um novo token "PROGRAM". E assim sucessivamente para todos os outros lexemas.

3.3 Classes

Foram utilizadas três classes:

- Main.java: classe com método principal, é responsável pela abertura do arquivo de entrada, invocação da leitura dos tokens através da classe Lexer e geração da saída.
- Token.java: classe com definição da estrutura dos tokens e definição dos formatos dos atributos a serem impressos na saída.
- Lexer.java: gerado pelo jflex, é responsável pela leitura e reconhecimento dos símbolos a partir do arquivo de entrada de acordo com as regras de reconhecimento no arquivo ".flex".

4 Compilação e execução

Após a utilização da ferramenta jflex para gerar o Lexer.java com a diretiva:

```
$ jflex tp.flex
$
```

O programa pode ser compilado e executado na órdem:

```
$ javac Main.java # compilação
$
$ java Main <arquivo de entrada> # execução
$
```

5 Análise

5.1 Análise de complexidade

- Main.java: Enquanto não encontra o fim do arquivo continua a percorrer o arquivo, assim, se considerarmos "n"linhas e "m"colunas, $O(m * n)$;
- Token.java: $O(1)$ apenas definição do TAD; e
- Lexer.java: $O(1)$, por apenas fazer testes, pode ser considerado linear.

Assim com complexidade final $O(m * n)$ no pior, melhor e caso médio.

5.2 Testes

Para a análise sintática foi criado o seguintes testes:

Teste 1

Entrada:

```
program teste1
 declare
     real a, b, c
 begin
     if a > b then
        c := a + b
     else c := a - b * c
     end
 end
```

Saída:

Token	Lexema	Coluna	Linha
PROGRAM	program	0	0
IDENTIFIER	teste1	0	8
DECLARE	declare	1	1
REAL	real	2	5
IDENTIFIER	a	2	10
COMMA	,	2	11
IDENTIFIER	b	2	13
COMMA	,	2	14
IDENTIFIER	c	2	16
BEGIN	begin	3	1
IF	if	4	5
IDENTIFIER	a	4	8
GT	>	4	10
IDENTIFIER	b	4	12
THEN	then	4	14
IDENTIFIER	c	5	8
ATRIB	:=	5	10
IDENTIFIER	a	5	13
PLUS	+	5	15
IDENTIFIER	b	5	17
ELSE	else	6	5
IDENTIFIER	c	6	10
ATRIB	:=	6	12
IDENTIFIER	a	6	15
MINUS	-	6	17
IDENTIFIER	b	6	19
MULT	*	6	21
IDENTIFIER	c	6	23
END	end	7	5
END	end	8	1

Teste 2

Entrada:

```
program teste2
 declare
     integer i, j, k
     b : boolean
 begin
    i := 4 * (5-3) * -10 / 50)
```

```
   j := i * 88;
   k := i * j / k;
   k := -4 +3;
   b := not false;
   b := 4 (3 and 7) = 0 or 1 != 4
end
```

Saída:

```
            Token     Lexema     Coluna      Linha
          PROGRAM    program          0          0
       IDENTIFIER     teste2          0          8
          DECLARE    declare          1          1
          INTEGER    integer          2          5
       IDENTIFIER          i          2         13
            COMMA          ,          2         14
       IDENTIFIER          j          2         16
            COMMA          ,          2         17
       IDENTIFIER          k          2         19
       IDENTIFIER          b          3          5
       TWO_POINTS          :          3          7
          BOOLEAN    boolean          3          9
            BEGIN      begin          4          1
       IDENTIFIER          i          5          4
            ATRIB         :=          5          6
  INTEGER_CONSTANT          4          5          9
             MULT          *          5         11
      PARENT_OPEN          (          5         13
  INTEGER_CONSTANT          5          5         14
             MINUS          -          5         15
  INTEGER_CONSTANT          3          5         16
     PARENT_CLOSE          )          5         17
             MULT          *          5         19
            MINUS          -          5         21
  INTEGER_CONSTANT         10          5         22
              DIV          /          5         25
  INTEGER_CONSTANT         50          5         27
     PARENT_CLOSE          )          5         29
       IDENTIFIER          j          6          4
            ATRIB         :=          6          6
       IDENTIFIER          i          6          9
             MULT          *          6         11
  INTEGER_CONSTANT         88          6         13
       SEMI_COMMA          ;          6         15
       IDENTIFIER          k          7          4
            ATRIB         :=          7          6
       IDENTIFIER          i          7          9
             MULT          *          7         11
       IDENTIFIER          j          7         13
              DIV          /          7         15
       IDENTIFIER          k          7         17
       SEMI_COMMA          ;          7         18
       IDENTIFIER          k          8          4
            ATRIB         :=          8          6
            MINUS          -          8          9
  INTEGER_CONSTANT          4          8         10
             PLUS          +          8         12
  INTEGER_CONSTANT          3          8         13
       SEMI_COMMA          ;          8         14
       IDENTIFIER          b          9          4
            ATRIB         :=          9          6
              NOT        not          9          9
            FALSE      false          9         13
```

```
      SEMI_COMMA          ;          9         18
      IDENTIFIER          b         10          4
           ATRIB         :=         10          6
 INTEGER_CONSTANT          4         10          9
      PARENT_OPEN          (         10         11
 INTEGER_CONSTANT          3         10         12
             AND        and         10         14
 INTEGER_CONSTANT          7         10         18
     PARENT_CLOSE          )         10         19
               EQ          =         10         21
 INTEGER_CONSTANT          0         10         23
               OR         or         10         25
 INTEGER_CONSTANT          1         10         28
               NE         !=         10         30
 INTEGER_CONSTANT          4         10         33
              END        end         11          1
```

Teste 3

Entrada:

```
program teste3
 declare
     integer j, k;
     array 100 integer a
     array 100 integer b;
     array 100 integer c
 begin
    i := 3; j := 4; k := 2;
    a(i) := a(2*i+3) + b(j) + c(3);
    c(i) := a(i)
 end
```

Saída:

```
            Token     Lexema     Coluna       Linha
          PROGRAM    program          0           0
       IDENTIFIER     teste3          0           8
          DECLARE    declare          1           1
          INTEGER    integer          2           5
       IDENTIFIER          j          2          13
            COMMA          ,          2          14
       IDENTIFIER          k          2          16
       SEMI_COMMA          ;          2          17
            ARRAY      array          3           5
 INTEGER_CONSTANT        100          3          11
          INTEGER    integer          3          15
       IDENTIFIER          a          3          23
            ARRAY      array          4           5
 INTEGER_CONSTANT        100          4          11
          INTEGER    integer          4          15
       IDENTIFIER          b          4          23
       SEMI_COMMA          ;          4          24
            ARRAY      array          5           5
 INTEGER_CONSTANT        100          5          11
          INTEGER    integer          5          15
       IDENTIFIER          c          5          23
            BEGIN      begin          6           1
       IDENTIFIER          i          7           4
            ATRIB         :=          7           6
 INTEGER_CONSTANT          3          7           9
       SEMI_COMMA          ;          7          10
       IDENTIFIER          j          7          12
```

```
            ATRIB         :=          7         14
  INTEGER_CONSTANT          4          7         17
        SEMI_COMMA          ;          7         18
        IDENTIFIER          k          7         20
            ATRIB         :=          7         22
  INTEGER_CONSTANT          2          7         25
        SEMI_COMMA          ;          7         26
        IDENTIFIER          a          8          4
       PARENT_OPEN          (          8          5
        IDENTIFIER          i          8          6
      PARENT_CLOSE          )          8          7
            ATRIB         :=          8          9
        IDENTIFIER          a          8         12
       PARENT_OPEN          (          8         13
  INTEGER_CONSTANT          2          8         14
              MULT          *          8         15
        IDENTIFIER          i          8         16
              PLUS          +          8         17
  INTEGER_CONSTANT          3          8         18
      PARENT_CLOSE          )          8         19
              PLUS          +          8         21
        IDENTIFIER          b          8         23
       PARENT_OPEN          (          8         24
        IDENTIFIER          j          8         25
      PARENT_CLOSE          )          8         26
              PLUS          +          8         28
        IDENTIFIER          c          8         30
       PARENT_OPEN          (          8         31
  INTEGER_CONSTANT          3          8         32
      PARENT_CLOSE          )          8         33
        SEMI_COMMA          ;          8         34
        IDENTIFIER          c          9          4
       PARENT_OPEN          (          9          5
        IDENTIFIER          i          9          6
      PARENT_CLOSE          )          9          7
            ATRIB         :=          9          9
        IDENTIFIER          a          9         12
       PARENT_OPEN          (          9         13
        IDENTIFIER          i          9         14
      PARENT_CLOSE          )          9         15
               END        end         10          1
```

Teste 4

Entrada:

```
program teste4
  declare
      integer divisor, number;
      boolean nofactor, prime
  begin
      read(number);
      write(number);
      divisor := number;
      nofactor := true;
      while nofactor and (divisor > 1) do
          if (number mod divisor) = 0 then nofactor := false
          else divisor := divisor - 1
          end
      end;
      prime := nofactor;
      if prime then write('S') else write('N') end
  end
```

Saída:

Token	Lexema	Coluna	Linha
PROGRAM	program	0	0
IDENTIFIER	teste4	0	8
DECLARE	declare	1	2
INTEGER	integer	2	6
IDENTIFIER	divisor	2	14
COMMA	,	2	21
IDENTIFIER	number	2	23
SEMI_COMMA	;	2	29
BOOLEAN	boolean	3	6
IDENTIFIER	nofactor	3	14
COMMA	,	3	22
IDENTIFIER	prime	3	24
BEGIN	begin	4	2
READ	read	5	6
PARENT_OPEN	(	5	10
IDENTIFIER	number	5	11
PARENT_CLOSE	)	5	17
SEMI_COMMA	;	5	18
WRITE	write	6	6
PARENT_OPEN	(	6	11
IDENTIFIER	number	6	12
PARENT_CLOSE	)	6	18
SEMI_COMMA	;	6	19
IDENTIFIER	divisor	7	6
ATRIB	:=	7	14
IDENTIFIER	number	7	17
SEMI_COMMA	;	7	23
IDENTIFIER	nofactor	8	6
ATRIB	:=	8	15
TRUE	true	8	18
SEMI_COMMA	;	8	22
WHILE	while	9	6
IDENTIFIER	nofactor	9	12
AND	and	9	21
PARENT_OPEN	(	9	25
IDENTIFIER	divisor	9	26
GT	>	9	34
INTEGER_CONSTANT	1	9	36
PARENT_CLOSE	)	9	37
DO	do	9	39
IF	if	10	10
PARENT_OPEN	(	10	13
IDENTIFIER	number	10	14
IDENTIFIER	mod	10	21
IDENTIFIER	divisor	10	25
PARENT_CLOSE	)	10	32
EQ	=	10	34
INTEGER_CONSTANT	0	10	36
THEN	then	10	38
IDENTIFIER	nofactor	10	43
ATRIB	:=	10	52
FALSE	false	10	55
ELSE	else	11	10
IDENTIFIER	divisor	11	15
ATRIB	:=	11	23
IDENTIFIER	divisor	11	26
MINUS	-	11	34
INTEGER_CONSTANT	1	11	36
END	end	12	10
END	end	13	6

```
        SEMI_COMMA          ;         13          9
        IDENTIFIER      prime         14          6
             ATRIB         :=         14         12
        IDENTIFIER   nofactor         14         15
        SEMI_COMMA          ;         14         23
                IF         if         15          6
        IDENTIFIER      prime         15          9
              THEN       then         15         15
             WRITE      write         15         20
       PARENT_OPEN          (         15         25
      CHAR_CONSTANT        'S'         15         26
       PARENT_CLOSE          )         15         29
              ELSE       else         15         31
             WRITE      write         15         36
       PARENT_OPEN          (         15         41
      CHAR_CONSTANT        'N'         15         42
       PARENT_CLOSE          )         15         45
                END        end         15         47
                END        end         16          2
```

Teste 5

Entrada:

```
program test5
  declare
     boolean b;
     procedure p(procedure f)
     declare
           integer x;
           procedure q begin write(x) end
     begin
   x := if b then 5 else 0 end;
           b := not b;
           if b then p(q) end;
           f
     end
     procedure r begin write('.') end
  begin
     write('x'); write('=');
     b :=false;
     p(r)
  end
```

Saída:

```
            Token     Lexema     Coluna      Linha
          PROGRAM    program          0          0
       IDENTIFIER      test5          0          8
          DECLARE    declare          1          2
          BOOLEAN    boolean          2          5
       IDENTIFIER          b          2         13
       SEMI_COMMA          ;          2         14
        PROCEDURE  procedure          3          5
       IDENTIFIER          p          3         15
      PARENT_OPEN          (          3         16
        PROCEDURE  procedure          3         17
       IDENTIFIER          f          3         27
     PARENT_CLOSE          )          3         28
          DECLARE    declare          4          5
          INTEGER    integer          5         11
       IDENTIFIER          x          5         19
       SEMI_COMMA          ;          5         20
```

```
       PROCEDURE  procedure          6         11
      IDENTIFIER          q          6         21
           BEGIN      begin          6         23
           WRITE      write          6         29
     PARENT_OPEN          (          6         34
      IDENTIFIER          x          6         35
    PARENT_CLOSE          )          6         36
             END        end          6         38
           BEGIN      begin          7          5
      IDENTIFIER          x          8          4
           ATRIB         :=          8          6
              IF         if          8          9
      IDENTIFIER          b          8         12
            THEN       then          8         14
 INTEGER_CONSTANT          5          8         19
            ELSE       else          8         21
 INTEGER_CONSTANT          0          8         26
             END        end          8         28
      SEMI_COMMA          ;          8         31
      IDENTIFIER          b          9         11
           ATRIB         :=          9         13
             NOT        not          9         16
      IDENTIFIER          b          9         20
      SEMI_COMMA          ;          9         21
              IF         if         10         11
      IDENTIFIER          b         10         14
            THEN       then         10         16
      IDENTIFIER          p         10         21
     PARENT_OPEN          (         10         22
      IDENTIFIER          q         10         23
    PARENT_CLOSE          )         10         24
             END        end         10         26
      SEMI_COMMA          ;         10         29
      IDENTIFIER          f         11         11
             END        end         12          5
       PROCEDURE  procedure         13          5
      IDENTIFIER          r         13         15
           BEGIN      begin         13         17
           WRITE      write         13         23
     PARENT_OPEN          (         13         28
    CHAR_CONSTANT        '.'         13         29
    PARENT_CLOSE          )         13         32
             END        end         13         34
           BEGIN      begin         14          2
           WRITE      write         15          5
     PARENT_OPEN          (         15         10
    CHAR_CONSTANT        'x'         15         11
    PARENT_CLOSE          )         15         14
      SEMI_COMMA          ;         15         15
           WRITE      write         15         17
     PARENT_OPEN          (         15         22
    CHAR_CONSTANT        '='         15         23
    PARENT_CLOSE          )         15         26
      SEMI_COMMA          ;         15         27
      IDENTIFIER          b         16          5
           ATRIB         :=         16          7
           FALSE      false         16          9
      SEMI_COMMA          ;         16         14
      IDENTIFIER          p         17          5
     PARENT_OPEN          (         17          6
      IDENTIFIER          r         17          7
    PARENT_CLOSE          )         17          8
```

```
               END        end         18          2
```

Teste 6

Entrada:

```
program teste7
  declare
      label L1, L2;
      integer x;
  begin
      x := 0;
      L1: x := x + 1;
      declare
          integer y;
      begin
          y := 1;
          declare
              integer z;
          begin
              z := x + y;
              if z < 10 then goto L1 else goto L2 end
          end
      end;
      L2: write(x)
  end
```

Saída:

```
             Token     Lexema     Coluna      Linha
           PROGRAM    program          0          0
        IDENTIFIER     teste7          0          8
           DECLARE    declare          1          2
             LABEL      label          2          6
        IDENTIFIER         L1          2         12
             COMMA          ,          2         14
        IDENTIFIER         L2          2         16
        SEMI_COMMA          ;          2         18
           INTEGER    integer          3          6
        IDENTIFIER          x          3         14
        SEMI_COMMA          ;          3         15
             BEGIN      begin          4          2
        IDENTIFIER          x          5          6
             ATRIB         :=          5          8
  INTEGER_CONSTANT          0          5         11
        SEMI_COMMA          ;          5         12
        IDENTIFIER         L1          6          6
        TWO_POINTS          :          6          8
        IDENTIFIER          x          6         10
             ATRIB         :=          6         12
        IDENTIFIER          x          6         15
              PLUS          +          6         17
  INTEGER_CONSTANT          1          6         19
        SEMI_COMMA          ;          6         20
           DECLARE    declare          7          6
           INTEGER    integer          8         10
        IDENTIFIER          y          8         18
        SEMI_COMMA          ;          8         19
             BEGIN      begin          9          6
        IDENTIFIER          y         10         10
             ATRIB         :=         10         12
  INTEGER_CONSTANT          1         10         15
        SEMI_COMMA          ;         10         16
```

```
          DECLARE    declare         11         10
          INTEGER    integer         12         14
       IDENTIFIER          z         12         22
       SEMI_COMMA          ;         12         23
            BEGIN      begin         13         10
       IDENTIFIER          z         14         14
            ATRIB         :=         14         16
       IDENTIFIER          x         14         19
             PLUS          +         14         21
       IDENTIFIER          y         14         23
       SEMI_COMMA          ;         14         24
               IF         if         15         14
       IDENTIFIER          z         15         17
               LT          <         15         19
  INTEGER_CONSTANT         10         15         21
             THEN       then         15         24
             GOTO       goto         15         29
       IDENTIFIER         L1         15         34
             ELSE       else         15         37
             GOTO       goto         15         42
       IDENTIFIER         L2         15         47
              END        end         15         50
              END        end         16         10
              END        end         17          6
       SEMI_COMMA          ;         17          9
       IDENTIFIER         L2         18          6
       TWO_POINTS          :         18          8
            WRITE      write         18         10
      PARENT_OPEN          (         18         15
       IDENTIFIER          x         18         16
     PARENT_CLOSE          )         18         17
              END        end         19          2
```

6 Conclusão

A contrução de um analisador léxico é o passo inicial para a produção de um compilados. Com os dados mostrados e testes, viu-se a validade do mesmo.

7 Bibliografia

1 A. V. Aho, M. S. Lam, R. Sethi & J.D. Ullman, Compilers Principles, Techniques, & Tools, Addison Wesley, First Edition, 1986.

2 Deitel, H. M. - Java, como programar, Bookman, Quarta Edição, 2003

3 Roberto S. Bigonha & Mariza A. S. Bigonha, Notas de Aula de Compiladores, UFMG, 2009

4 Jflex - the fast scanner generator for java. http://jflex.de/.

8 Anexo

A seguir se encontra o códifo fonte do software e da entrada do jflex.