UNIVERSIDAD NEBRIJA

Escuela Politécnica Superior

Procesadores de lenguaje

Práctica 1

Desarrolle un analizador léxico que, dado un programa escrito en un lenguaje similar a C, identifique sus componentes léxicos.

El analizador léxico debe cumplir los siguientes requisitos:

- Analizar un programa almacenado en una cadena de caracteres.
- Descartar los espacios, tabuladores y saltos de línea del fichero de entrada.
- Contar las líneas del programa.
- Reconocer las secuencias de dígitos como números enteros o números reales.
- Reconocer las palabras reservadas del lenguaje. Utilice la estructura de datos `Hashtable` de Java para almacenar las palabras `break`, `do`, `else`, `float`, `for`, `if`, `int`, `while`.
- Reconocer los identificadores del programa. Una secuencia que empieza por un carácter y le siguen cero o más caracteres o dígitos es un identificador siempre que no sea una palabra reservada.
- Reconocer los operadores aritméticos, operadores relacionales, operadores lógicos y los caracteres delimitadores: &&, ||, &, |, ==, =, !, !=, <=, <, >=, >, +, -, *, /, %, (,), [,], {, }, ;.
- Utilizar el carácter cero para indicar el fin del programa.

Los componentes léxicos de los operadores y de los caracteres delimitadores.

Lexema	Componente léxico
>	greater_than
>=	greater_equals
<	less_than
<=	less_equals
==	equals
!=	not_equals
=	assignment
+	add
-	subtract
*	multiply
/	divide
%	remainder
&&	and
&	bitwise_and
| |	or
|	bitwise_or
!	not
;	semicolon
(	open_parenthesis
)	closed_parenthesis
[	open_square_bracket
]	closed_square_bracket
{	open_bracket
}	closed_bracket

La clase `ComponenteLexicoBasico`.

```
package practica_1;

public class ComponenteLexicoBasico {
   private String etiqueta;   // etiqueta del token
   private String valor;      // valor asociado a un token id o num

   public ComponenteLexicoBasico(String etiqueta) {
      this.etiqueta = etiqueta;
      this.valor = "";
   }

   public ComponenteLexicoBasico(String etiqueta, String valor) {
      this.etiqueta = etiqueta;
      this.valor = valor;
   }

   public String getEtiqueta() {
      return this.etiqueta;
   }

   // toString()  devuelve una cadena con el contenido del token

   public String toString() {
      if (this.valor.length() == 0)
         return this.etiqueta;
      else
         return this.etiqueta + ", " + this.valor;
   }
}
```

La clase `LexicoBasico`.

```
package practica_1;

import java.util.Hashtable;

public class LexicoBasico {

   // palabrasReservadas: tabla Hash de palabras reservadas
   // posicion: posición del carácter actual
   // lineas: número de líneas del programa
   // caracter: carácter actual devuelto por extraeCaracter()
   // programa: código fuente del programa

   private Hashtable<String, String> palabrasReservadas;
   private int posicion;
   private int lineas;
   private char caracter;
   private String programa;

   public LexicoBasico(String programa) {
      this.posicion = 0;
      this.lineas = 1;

      // la tabla Hash de palabras reservadas almacena el lexema
      // (clave) y el token (valor), la etiqueta del token coincide
      // con el lexema de la palabra reservada

      this.palabrasReservadas = new Hashtable<String, String>();

      this.palabrasReservadas.put("break", "break");
      this.palabrasReservadas.put("do", "do");
      this.palabrasReservadas.put("else", "else");
      this.palabrasReservadas.put("float", "float");
      this.palabrasReservadas.put("for", "for");
      this.palabrasReservadas.put("if", "if");
      this.palabrasReservadas.put("int", "int");
      this.palabrasReservadas.put("while", "while");

      // al final del programa se añade el carácter 0 para indicar
      // el final, cuando el analizador léxico encuentra este carácter
      // devuelve el token “end_program”

      this.programa = programa + (char) (0);
   }
```

```
private char extraeCaracter() {
   return this.programa.charAt(this.posicion++);
}

private void devuelveCaracter() {
   this.posicion--;
}

// extraeCaracter(char c) se usa para reconocer operadores con
// lexemas de dos caracteres: &&, ||, <=, >=, ==, !=

private boolean extraeCaracter(char c) {
   if (this.posicion < this.programa.length() - 1) {
      this.caracter = extraeCaracter();

      if (c == this.caracter)
         return true;
      else {
         devuelveCaracter();

         return false;
      }
   }
   else
      return false;
}

public int getLineas() {
   return this.lineas;
}

// la clase Character de Java ofrece los métodos:

// - Character.isDigit(char c): devuelve true si c es un dígito
// - Character.isLetter(char c): devuelve true si c es una letra
// - Character.isLetterOrDigir(char c): devuelve true si c es una
                                        letra o un dígito

// estos métodos se usan para reconocer identificadores y números.
// aplicando las expresiones regulares:

// - id = letra (letra | digito )*
// - numero = digito+ ( . digito+ )?
```

```
public ComponenteLexicoBasico getComponenteLexico() {

   // el analizador léxico descarta los espacios (código 32),
   // tabuladores (código 9) y saltos de línea (códigos 10 y 13)

   while (true) {
      this.caracter = extraeCaracter();

      if (this.caracter == 0)
         return new ComponenteLexicoBasico("end_program");
      else if (this.caracter == ' ' || (int) this.caracter == 9 ||
            (int) this.caracter == 13)
         continue;
      else if((int) this.caracter == 10)
         this.lineas++;
      else
         break;
   }

   // secuencias de dígitos de números enteros o reales

   if (Character.isDigit(this.caracter)) {
      String numero = "";

      do {
         numero = numero + this.caracter;

         this.caracter = extraeCaracter();
      } while (Character.isDigit(this.caracter));

      if (this.caracter != '.') {
         devuelveCaracter();

         return new ComponenteLexicoBasico("int", numero);
      }

      do {
         numero = numero + this.caracter;

         this.caracter = extraeCaracter();
      } while (Character.isDigit(this.caracter));

      devuelveCaracter();

      return new ComponenteLexicoBasico("float", numero);
   }
```

```
        // identificadores y palabras reservadas

        if (Character.isLetter(this.caracter) ) {
           String lexema = "";

           do {

              lexema = lexema + this.caracter;
              this.caracter = extraeCaracter();

           } while(Character.isLetterOrDigit(this.caracter));


           devuelveCaracter();

           if (this.palabrasReservadas.containsKey(lexema))
              return new
                 ComponenteLexicoBasico((String)
                    this.palabrasReservadas.get(lexema));
           else
              return new ComponenteLexicoBasico("id", lexema);
        }

        // operadores aritméticos, relacionales, lógicos y
        // caracteres delimitadores

        switch (this.caracter) {
        case '=': return new ComponenteLexicoBasico("assignment");
        case '<': return new ComponenteLexicoBasico("less_than");
        case '>': return new ComponenteLexicoBasico("greater_than");
        case '+': return new ComponenteLexicoBasico("add");
        case '-': return new ComponenteLexicoBasico("subtract");
        case '*': return new ComponenteLexicoBasico("multiply");
        case '/': return new ComponenteLexicoBasico("divide");
        case '%': return new ComponenteLexicoBasico("remainder");
        case ';': return new ComponenteLexicoBasico("semicolon");
        case '(': return new ComponenteLexicoBasico("open_parenthesis");
        case ')':
           return new ComponenteLexicoBasico("closed_parenthesis");
        default: return new ComponenteLexicoBasico("invalid_char");
        }
    }
```

La clase `TestLexicoBasico`.

```
package practica_1;

public class TestLexicoBasico1 {

   public static void main(String[] args) {
      ComponenteLexicoBasico etiquetaLexica;

      String programa = "int k; for (int i=0; i<10; i=i+1) k=k*2;";

      LexicoBasico lexico = new LexicoBasico(programa);

      int c = 0;

      System.out.println("Test léxico basico \t" + programa + "\n");

      do {
         etiquetaLexica = lexico.getComponenteLexico();

         System.out.println("<" + etiquetaLexica.toString() + ">");

         c++;
      } while (!etiquetaLexica.getEtiqueta().equals("end_program"));

      System.out.println("\nComponentes léxicos: " + c +
                         ", lineas: " + lexico.getLineas());
   }

}
```

El programa de prueba.

```
{
   float suma = 0.0;

   int [10] v;

   for (int k=0; k<10; k=k+1) {
      if (k % 2 == 0)
         suma = suma + k*10.5;
      else
         suma = suma + k*15.75;

      v[i] = suma;
   }

   if (suma <= 25.0)
      suma = suma / 2.5;
   else
      suma = suma * 4.5;

}
```

Los componentes léxicos:

```
<open_bracket>
<float>
<id, suma>
<assignment>
<float, 0.0>
<semicolon>
<int>
<open_square_bracket>
<int, 10>
<closed_square_bracket>
<id, v>
<semicolon>
<for>
<open_parenthesis>
<int>
<id, k>
<assignment>
<int, 0>
<semicolon>
<id, k>
<less_than>
<int, 10>
<semicolon>
<id, k>
<assignment>
<id, k>
<add>
<int, 1>
<closed_parenthesis>
<open_bracket>
<if>
<open_parenthesis>
<id, k>
<remainder>
<int, 2>
<equals>
<int, 0>
<closed_parenthesis>
<id, suma>
<assignment>
<id, suma>
<add>
<id, k>
<multiply>
```

```
<float, 10.5>
<semicolon>
<else>
<id, suma>
<assignment>
<id, suma>
<add>
<id, k>
<multiply>
<float, 15.75>
<semicolon>
<id, v>
<open_square_bracket>
<id, i>
<closed_square_bracket>
<assignment>
<id, suma>
<semicolon>
<closed_bracket>
<if>
<open_parenthesis>
<id, suma>
<less_equals>
<float, 25.0>
<closed_parenthesis>
<id, suma>
<assignment>
<id, suma>
<divide>
<float, 2.5>
<semicolon>
<else>
<id, suma>
<assignment>
<id, suma>
<multiply>
<float, 4.5>
<semicolon>
<closed_bracket>
<end_program>

Componentes léxicos: 84, lineas: 20
```