Compilador lexicográfico básico

“El Automatador”

Instructor

Roberto Flórez Rueda

Estudiantes

Juan Cardona - 1044508271

Roy Maestre - 1214715506

UdeA

Facultad de Ingeniería

Departamento de Ingeniería de Sistemas

Teoría de Lenguajes y Laboratorio

2020-1

Descripción General

En este documento expondremos un compilador léxicográfico llamado Automatador, basado en el uso de autómatas finitos, este permite seleccionar un archivo escrito en el lenguaje de programación Java, para Identificar si está estructuralmente correcto, (no se evalúa lógicamente) y mostrar un mensaje que indica si el código está escrito con una estructura correcta, de no ser el caso dirá el porqué y en dónde se encuentra el error que hace invalida la estructura del código.

El programa permite abrir un archivo *.java para analizarlo, una vez haya finalizado el proceso de análisis y retornado si el código es estructuralmente correcto o no, permitirá seleccionar otro archivo *.java para realizar el mismo análisis.

Con el fin de ilustrar de una manera práctica los tipos de secuencias que nuestro compilador lexicográfico admite, a continuación se listan ejemplos de cada tipo de estructura válidos:

```
String b;                       #Instanciación de variable
int a , b , c ;                 #Instanciación múltiple de variables
float b = 2.7f;                 #Instanciación de variable con asignación
String x = “pato”, z =“gato”;   #Instanciación múltiple de variables con asignación
boolean c=true, d=false, x;     #Instanciación múltiple de variables con o sin asignación
String b = (a*5) && true;       #Instanciación de variable con asignación de expresión

n = “Textodeejemplo”;           #Asignación de valor
a = (a+c-b)*3 ;                 #Asignación de expresión
My_nUmber_b -= b ;              #Modificación de variable por variable
n += (3-7*b) ;                  #Modificación de variable por expresión

a--;                            #Incremento 1 unidad
```

Para verificar que una línea escrita en el archivo que ingrese a nuestro compilador lexicográfico tiene una estructura válida, este utiliza 4 autómatas, uno que reconoce las variables y las identifica como palabra reservada , otro que identifica las variables, otro para símbolos y operadores y otro para los números, así verifica si las sentencias que están en el código corresponden a:

- **Tipo de variable**: int, double, float, bool y string
- **Variable**: cualquier palabra (no inicia con número, contiene solo letras mayusculas, minusculas, numeros y _)
- **Palabra reservada**: Tipos de variables, false, true, if, while, for y else
- **Separador**: ,

- **Asignación**: =
- **Modificador**: +=, -=,*=, /*, %=
- **Operadores lógico y aritméticos**: +, -, *, /, %, ==, !=, <=, >=, |, ||, &, &&
- **Incremento**: ++, --
- **Fin de línea**: ;

El archivo que analizaremos será el siguiente:

```
1  String b;
2  int a , b , c ;
3  float b = 2.7f;
4  String x = “pato”, z =“gato”;
5  boolean c=true, d=false, x;
6  String b = (a*5) && true;
7
8  n = “Textodeejemplo”;
9  a = (a+c-b)*3 ;
10 My_nUmber_b -= b ;
11 n += (3-7*b) ;
12
13 a--;
```

Restricciones

Antes de comenzar con el análisis de las líneas, hay que hacer algunas claridades acerca de las restricciones que va a tener el código que nuestro corrector lexicográfico admitirá.

Solo se admitirán archivos *.java aunque también puede analizar archivos *.txt pero solo si el código que está allí escrito corresponde al lenguaje Java.

Solo se admitirán los tipos de variables, variables, símbolos y operadores listados anteriormente en las sentencias válidas. Se garantiza que siempre se deben cerrar todos las comillas abiertas, pero no los paréntesis.

De esta manera luego de analizar una línea como la siguiente:

```
3  float b = 2.7f;
```

El corrector léxico gráfico utilizando los autómatas finitos nos arrojará una lista ligada como la siguiente:

```
[Tipo | float ] [Variable | b ] [Asignacion | = ] [float | 2.7f ] [Fin de linea | ; ]
```

En caso de que haya algún error en la estructura de la línea, significa que la línea contiene alguna combinación de caracteres que no corresponde a alguno de los tipos listados anteriormente, por ejemplo:

```
10  1My_nUmber_b -= b ;
```

De esta manera nos arrojará una advertencia en donde indique el de error que corresponda y lo presentara, así:

“

1My_nUmber_b

^ ERROR: Invalid variable

”

Fecha límite de entrega: 01/Nov/2020 23:59:00
correo: david.mejia3@udea.edu.co
asunto: “Primera Práctica TdeL 2020 - 1”
link de su repositorio: https://github.com/Royk8/TdLAutomataInfinito