IIC2323 — Construcción de Compiladores
Entrega 1: Análisis léxico

Primer semestre 2014

1. Descripción general

La entrega consiste en desarrollar un analizador léxico para el lenguaje especificado, utilizando un scanner generado automáticamente. Este recibirá como input el código fuente de un programa, y entregará el análisis en el formato que se detalla a continuación.

2. Formato

Por cada token, se generará una línea con el siguiente formato:

```
char(line,col):type:value
```

donde char corresponde a la posición absoluta dentro del archivo, line, al número de línea, y col al número de columna.

Los valores de *type* y *value* dependen del tipo del token:

2.1. Uppercase identifier

type corresponde a `uppercase_identifier`, y value al valor del identificador.

2.2. Identifier

type corresponde a `identifier`, y value al valor del identificador.

2.3. Iterator name

type corresponde a `iter_name`, y value al nombre.

2.4. Keywords

type es `keyword`, y value el keyword.

2.5. Special symbols

type es `symbol`, y value el símbolo.

2.6. Bool literal

type es `bool`, y value `true` o `false` según corresponda.

secuencia	reemplazo
a	*alert*
b	*backspace*
f	*formfeed*
n	*newline*
r	*carret*
t	*hortab*
v	*verttab*

Cuadro 1: Reemplazos a realizar

2.7. Char literal

type es `char`, y value el valor del caracter de forma explícita, a menos que corresponda a una secuencia escapada, en cuyo caso se debe reemplazar según la tabla 2.7:

2.8. String literal

type es `string[len]`, donde `len` es el largo, y value es el contenido (sin " "), reemplazando las secuencias escapadas de igual forma que en el caso de los chars.

Notar que el largo corresponde al largo real del string, es decir las secuencias escapadas se consideran como un caracter.

2.9. Integer literal

type es `integer`, y value el valor número.

2.10. Lexical errors

Si se detecta un error léxico, se debe imprimir la línea:
`LEXICAL_ERROR(line,col)`
indicando la posición del error. Tras esto, se debe detener el proceso.

3. Ejemplo

Para el siguiente programa en Sather:

```
--Esto es un comentario.
class LIST{T} < COLLECTION{T} is
    attr current:T;
    attr next:SAME;
end
```

el archivo comenzaría con:

```
25(1,0):keyword:class
31(1,6):uppercase_identifier:LIST
35(1,10):symbol:{
36(1,11):uppercase_identifier:T
37(1,12):symbol:}
39(1,14):symbol:<
```

4. Herramientas y entrega

Se deberá desarrollar en Java, utilizando JFlex para generar el scanner[1], y bison para definir los tokens.

La entrega se realizará a través de un repositorio Git o Mercurial, al cual se debe dar acceso al ayudante del curso, del cual se revisará el contenido de la rama principal al momento de la entrega, es decir al día 11 de Abril, a las 23:59 horas.

Junto al código, se debe entregar un script que permita compilar y ejecutar el programa, el que debe seguir un template a ser publicado, y además un archivo de texto con una descripción breve de la solución.

[1] Alternativamente, también se podrá utilizar C99 y Flex, previo aviso via email a `jsnavar1@uc.cl`