Universidad Simón Bolívar

Departamento de Computación y Tecnología de la Información

CI3725 - Traductores e Interpretadores

Diciembre - Marzo 2014-2015

Setlan - Etapa I

Análisis Lexicográfico (5%)

Especificación de la entrega

En la primera etapa de desarrollo del interpretador para el lenguaje `Setlan` se debe implementar su analizador lexicográfico. Siguiendo las especificaciones de la definición del lenguaje deberán identificar los *tokens* relevantes, crear expresiones regulares que los reconozcan e implantar el analizador utilizando la herramienta escogida por el equipo de trabajo.

Deben de considerar los siguientes como *tokens* de `Setlan`:

- Las palabras reservadas del lenguaje `Setlan`.
- Los identificadores de cada variable, recordando que este *token* es único y definido por el nombre del identificador asociado.
- Los valores constantes permitidos por el lenguaje `Setlan`:
    - Los números enteros, cuyo contenido del *token* asociado sea el número reconocido.
    - Las cadenas de caracteres encerradas entre comillas dobles, cuyo *token* asociado tendrá como contenido la cadena en cuestión.
    - Las constantes booleanas.
- Los símbolos utilizados para especificar valores, operadores, separadores y bloques.

Recuerden que las tabulaciones, los espacios en blanco, los saltos de línea y los comentarios **no** se definen como *tokens* del lenguaje, por lo tanto **no** deben de ser reconocidos por el analizador lexicográfico.

Ejecución

Para la ejecución del interpretador su programa deberá llamarse `setlan` y recibirá como primer argumento el nombre del archivo con el código en `Setlan` a analizar.

Por salida, se debe mostrar todos y cada uno de los *tokens* reconocidos por el analizador, especificando -en cada uno- la **línea** y **columna** donde fue encontrado.

Si un código en `Setlan` posee errores léxicos, se debe mostrar por salida **todos** los errores encontrados y **sólo** los errores encontrados. Es decir, la salida **no** puede contener tanto *tokens* como errores léxicos del código. Asimismo, deben recordar que los errores léxicos no son considerados como *tokens* reconocidos por el lenguaje.

Ejemplo de un programa correcto en `Setlan` y su salida de análisis lexicográfico respectiva:

```
program {
    using
        int x,y,z;
    in

    scan x;
    println "Hola, soy la variable x, valgo: ", x;
    y = x+2;
    # Hola, soy un comentario y no un token. Seamos amigos, :).
}
```

Salida:

```
$ ./setlan program.stl
token PROGRAM       value (program) at line 1, column 1
token LCURLY        value ({) at line 1, column 9
token USING         value (using) at line 2, column 5
token INT           value (int) at line 3, column 9
token IDENTIFIER    value (x) at line 3, column 13
token COMMA         value (,) at line 3, column 14
token IDENTIFIER    value (y) at line 3, column 15
token COMMA         value (,) at line 3, column 16
token IDENTIFIER    value (z) at line 3, column 17
token SEMICOLON     value (;) at line 3, column 18
token IN            value (in) at line 4, column 5
token SCAN          value (scan) at line 6, column 5
token IDENTIFIER    value (x) at line 6, column 10
token SEMICOLON     value (;) at line 6, column 11
token PRINTLN       value (println) at line 7, column 5
token STRING        value ("Hola, soy la variable x, valgo: ") at line 7, column 13
token COMMA         value (,) at line 7, column 47
token IDENTIFIER    value (x) at line 7, column 49
token SEMICOLON     value (;) at line 7, column 50
token IDENTIFIER    value (y) at line 8, column 5
token ASSIGN        value (=) at line 8, column 7
token IDENTIFIER    value (x) at line 8, column 9
token PLUS          value (+) at line 8, column 10
token INTEGER       value (2) at line 8, column 11
token SEMICOLON     value (;) at line 8, column 12
token RCURLY        value (}) at line 10, column 1
```

Y, por otro lado, un ejemplo de programa incorrecto con su salida asociada:

```
program {
    #Soy yo otra vez, el comentario, ¿me extrañaste? :)
    using
        int $,x;
    in
    scan y;
    x = true+~;
    print &;
}
```

Salida:

```
Error: Se encontró un caracter inesperado "$" en la Línea 4, Columna 12.
Error: Se encontró un caracter inesperado "~" en la Línea 7, Columna 13.
Error: Se encontró un caracter inesperado "&" en la Línea 8, Columna 10.
```

Noten que para la salida de un código *correcto* en `Setlan`, en esta primera etapa de desarrollo, se imprime el toda la informaición interesante del *token*, como cuál fue el econtrado, su posición y su información asociada. Recuerden que la correctitud de cada entrega determinará el desarrollo de la siguiente. Todas las entregas pertenecen a un mismo proyecto de desarrollo.

Implementación

Para la implementación del interpretador del lenguaje `Setlan`, pueden escoger uno (1) de los tres (3) lenguajes de programación a continuación. Para cada uno de ellos se indica las herramientas disponibles para el desarrollo de un interpretador de código:

- *Python:*
  - *python* 2.7
  - *Python Lex-Yacc (PLY).* Para esta primera etapa de desarrollo utilizarán el submódulo **Lex** de *PLY*. Sin embargo, el submódulo **Yacc** será utilizado en siguientes etapas.
- *Ruby:*
  - *ruby* 1.9
  - Para esta etapa de desarrollo no se utilizará una herramienta en *Ruby* que permita el análisis lexicográfico, por lo tanto, el trabajo para esta entrega se debe realizar a través del manejo de las expresiones regulares del lenguaje. Para entregas posteriores se utilizará *Racc* para el análisis sintáctico.
- *Haskell:*
  - *GHC* 7.6.3 ó 7.8.3
  - *Alex* y *Happy* . Para esta etapa de desarrollo utilizarán el generador de analizadores lexicográficos, *Alex*. Posteriores entregas requerirán de *Happy* para el análisis sintáctico.

Entrega

Formato de Entrega

Deben enviar un correo electrónico a **todos los preparadores** con el asunto: `[CI3725]eXgY` donde `X` corresponde al número de la entrega e `Y` al número del equipo. El correo debe incluir lo siguiente:

- Código fuente debidamente documentado.
- En caso de utilizar Haskell, deben incluir un archivo Makefile o un archivo de configuración para *Cabal*. Si su proyecto no compila, el proyecto no será corregido.
- Un archivo de texto con el nombre `LEEME.txt` donde **brevemente** se expliquen:
  - Decisiones de implementación
  - Estado actual del proyecto
  - Problemas presentes
  - Cualquier comentario respecto al proyecto que consideren necesario
  - Este archivo debe estar identificado con los nombres, apellidos y carné de cada miembro del equipo de trabajo.

Fecha de entrega

La fecha límite de entrega del proyecto es el día **viernes 23** de enero de 2015 (semana 5) *hasta* las **11:50pm**, entregas hechas más tarde tendrán una **penalización del 20%** de la nota, esta penalización aplica por cada día de retraso.