
# Haskell: Lexer for C Language

Iván Aveiga
Kevin Campuzano

Escuela Superior Politécnica del Litoral

4 de septiembre de 2013

# Lexer 0.1

Implementamos la idea básica de un analizador léxico, fue escrito .ª mano", hay herramientas que podían usarse como Alex para generar análisis léxico y Parsec para *parsear*. Partimos definiendo los tokens que vamos a reconocer siendo los siguientes:

- Operator: +, -, /, *, ^, /, %
- Reserved Word: auto, break, case, const, continue, default, do, else, enum, extern, for, goto, if, register, return, signed, sizeof, static, struct, switch, typedef, union, unsigned, void, volatile, while.
- Operadores de Incremento y Decremento.
- Operadores lógicos.
- Operadores relacionales.
- Primitve types: int, float, double, char, long
- Integers.
- Reals.
- Strings.
- Todo lexeme que no entre en la categoría de los tokens anteriores entra a la categoría de UNKNOWN TOKEN.

# Lexer 0.1

El lexer opera bajo las siguientes condiciones:

1. Solo funciona con espacios en blanco, es decir, no detecta la siguiente expresión en sus respectivos lexemas

$$(3 * 2) + 3$$

2. No Detecta punteros.

# Lexer 0.1

El algoritmo básico para generar la lista de (lexeme, tokens) es el siguiente:

1. Leer el archivo fuente .c .
2. Generar la lista de lexemas usando el espacio en blanco como delimitador.
3. Asignar a cada lexema en la lista su respectivo token.
4. Guardar en un archivo .csv la lista de (lexema, token).