**UNIVERSIDAD DE SANTIAGO DE CHILE**
**FACULTAD DE INFORMÁTICA**
**COMPILADORES**

# LABORATORIO Nº2

# ANALIZADOR

# LEX – SIN - SEM

Nombres:
- Rodrigo Araya
- Claudio Hernández
- Marcelo Quiroz

Profesora: Jacqueline Köhler
Ayudante: Patricia Riquelme
Fecha: 09 de julio de 2015

## *Índice:*

## 1. Introducción

En el siguiente informe se detallan las características del programa de generación de un analizador Léxico, Sintáctico y con código intermedio (semántico), correspondiente al Laboratorio Nº2.

Se analizarán los algoritmos de solución explicando cada proceso ordenado por fases de ejecución.

El programa tiene como objetivo analizar un archivo de entrada y determinar si cumple con una gramática dada, con opciones de comentar dicha gramática.

## 2. Planteamiento del Problema

Se requiere analizar un archivo de entrada para determinar si corresponde a la gramática dada.

## 3. Desarrollo de la Solución

Para abordar la problemática se realizaron los siguientes procesos;

### 3.1. Implementación de las librerías JFlex y Java CUP.

Las librerías pueden ser descargadas desde los siguientes sitios web:

- JFLex: https://www.cs.princeton.edu/~appel/modern/java/JLex
- Java_CUP: https://www.cs.princeton.edu/~appel/modern/java/CUP/

### 3.2. Creación del proyecto y copia de las librerías.

Se crea el proyecto el proyecto Java (tipo java aplication)

Dentro de la carpeta src creamos una carpeta llamada JLex donde copiamos la librería Main.java

Desde la librería Java CUP se extrae y copia la carpeta java_cup dentro de la carpeta src

### 3.3. Extracción de librerías.

Ingrese a un terminal, diríjase a la carpeta src y escriba los siguientes comandos para extraer las librerías:

- JFlex: javac JFlex/Main.java
- Java CUP: javac java_cup/*.java

Tras el paso anterior las librerías quedarán disponibles para su uso.

### 3.4. Generación de Reglas Léxicas, Sintácticas y Código Intermedio Semántico.

Se crean dos archivos en blanco, uno llamado Yylex (para reglas léxicas) y parse.cup (para reglas sintácticas y código intermedio semántico).

**Archivo Yylex:**

```
package l2_analizador;
import java_cup.runtime.Symbol;
%%
%cup
%public
%full
%char
%ignorecase
%eofval{
    return new Symbol(sym.EOF ,new String("Fin del Archivo"));
%eofval}
%linea
BINARIO = ([0-1])
NUM = ([2-9])
LETRA=([a-z])
```

```
%%
" " {return new Symbol(sym.ESPACIO, new String(yytext()));}
"#" {return new Symbol(sym.COMENTARIO, new String(yytext()));}
"(" {return new Symbol(sym.APARTENTESIS, new String(yytext()));}
")" {return new Symbol(sym.CPARTENTESIS, new String(yytext()));}
"+" {return new Symbol(sym.UNION, new String(yytext()));}
"*" {return new Symbol(sym.CLAUSURA, new String(yytext()));}
"." {return new Symbol(sym.CONCATENACION, new String(yytext()));}
{BINARIO} {return new Symbol(sym.BINARIO, new String(yytext()));}
{NUM} {return new Symbol(sym.NUM, new String(yytext()));}
{LETRA} {return new Symbol(sym.LETRA, new String(yytext()));}
. {System.out.print(" #ERROR#"); } //Si no encuentra ninguno de los caracteres anteriores marcará error
```

## Archivo parser.java:

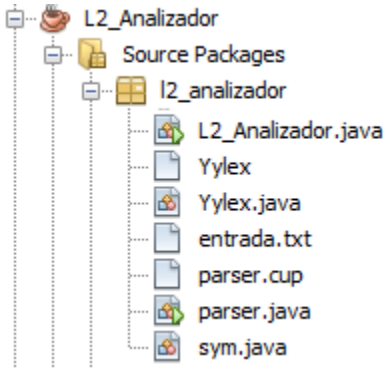
```
package l2_analizador;
import java.io.*;
import java.util.*;
import java_cup.runtime.*;
parser code
{:
   public static void main(String args[]) throws Exception{
      new parser(new Yylex(new FileInputStream(args[0]))).parse();
   }
   public static void cargar(String Archivo) throws Exception{
      new parser(new Yylex(new FileInputStream(Archivo))).parse();
   }
:};
terminal BINARIO, NUM, APARTENTESIS, CPARTENTESIS, UNION, CLAUSURA, CONCATENACION, ESPACIO,
LETRA, COMENTARIO;
non terminal INICIO, ER, FR;
precedence left CLAUSURA, CONCATENACION, UNION;
precedence left COMENTARIO;
start with INICIO;
INICIO::= ER;
ER::= APARTENTESIS {:System.out.print(" /Abre Parent./ "); :} ER CPARTENTESIS {:System.out.print(" /Cierra
Parent./ "); :} ER
   | ER CLAUSURA {:System.out.print(" /Clausura/ "); :} ER
   | ER UNION {:System.out.print(" /Union/ "); :} ER
   | ER CONCATENACION {:System.out.print(" /Concatenacion/ "); :} ER
   | BINARIO {:System.out.print(" /Digito/ "); :}
   | BINARIO COMENTARIO FR{:System.out.print(" /Digito/ "); :}
   | COMENTARIO FR
   |;
FR::= LETRA FR
   | NUM FR
   | BINARIO FR
   | ESPACIO FR
   | COMENTARIO ER;
```

### 3.5. Compilación de archivos Yylex y parse.cup.

Para compilar los archivos ingrese a un terminal, ingrese a la carpeta src y escriba el siguiente código:

- JFlex: JLex/Main Yylex
- Java CUP: java_cup/Main parser.cup

  Tras la ejecuciones anteriores se generarán los archivos Yylex.java (al compilar Yylex), y los archivos parser.java y sym.java (al compilar el archivo parse.cup).




### 3.6. Generación de archivo main.

Generación de código para la ejecución de las reglas léxicas, sintácticas y semánticas, que escaneen un archivo de entrada.

Código Java:

```
package l2_analizador;
public class L2_Analizador {

   public static void main(String[] args)throws Exception{
      System.out.println("ANALIZADOR LEX-SIN-SEM:");
      System.out.println("-----------------------");
      System.out.println("\nCadena Analizada:");
     try{
      parser instancia=new parser();
      instancia.cargar("entrada.txt");
     }catch (Exception e) {
      System.out.println("\n\nResultado: Error en la Gramatica");
     }
    }

}
```

### 3.7. Generación de archivo de entrada.

En la carpeta src se genera un archivo “entrada.txt” con las reglas a analizar.

## 4. Conclusiones

El uso de las librerías JFlex y Java CUP facilitan de manera considerable la generación de un analizador de código; así mismo facilitan la comprensión de cómo interactúan las partes de un analizador al momento de escanear un código dado.