Práctica. Primera Fase

Desarrollo de analizadores léxicos

En esta primera parte debe realizarse el siguiente trabajo:

1) Desarrollo manual de un analizador léxico para **Tiny(0)**, un subconjunto de **Tiny** (véase el **apéndice A**). Para ello, deberá entregarse:
   - Un apartado en la memoria con las siguientes secciones:
     - Enumeración de las clases léxicas de **Tiny(0)**. Para cada clase debe incluirse, además, una descripción informal, en lenguaje natural.
     - Una especificación formal del léxico del lenguaje mediante definiciones regulares.
     - Diseño de un analizador léxico para el lenguaje mediante un diagrama de transiciones.
   - Una implementación manual, en Java, del analizador léxico. Debe proporcionarse, además, un programa de prueba que acepte como argumento el archivo a procesar, y genere como salida una descripción legible de la secuenciad de tokens reconocida, y, si procede, un mensaje legible de error léxico detectado.
2) Desarrollo de un analizador léxico completo para **Tiny** mediante la herramienta **JFlex**. Para ello, deberá entregarse:
   - Un apartado en la memoria con las siguientes secciones:
     - Enumeración de las clases léxicas de **Tiny**. Para cada clase debe incluirse, además, una descripción informal, en lenguaje natural.
     - Una especificación formal del léxico del lenguaje mediante definiciones regulares.

     Importante: (obviamente) en esta segunda parte de la memoria **no** hay que incluir diagrama de transiciones
   - Una implementación basada en **JFlex** del analizador léxico. Para ello deberá entregarse la especificación de entrada a **JFLex**, y todos los archivos Java adicionales requeridos por la implementación. Debe proporcionarse, además, un programa de prueba que acepte como argumento el archivo a procesar, y genere como salida una descripción legible de la secuencia de tokens reconocida, y, si procede, un mensaje legible de error léxico detectado.

La memoria deberá incluir una portada en la que aparezcan los nombres y apellidos de los integrantes del grupo, y el número de grupo.

Fechas:

- Presentación en el laboratorio de las memorias: **lunes 5 de febrero 2024**
- Presentación en el laboratorio de las implementaciones: **lunes 12 de febrero de 2024**
- Fecha límite de entrada: **Viernes 16 de febrero de 2024, a las 23:59h**.

La entrega final se realizará:

- A través del campus virtual, en un único .zip. Dicho archivo debe contener: (i) un documento PDF `memoria_lexico.pdf` con la memoria requerida; (ii) una carpeta `implementacion_manual`, en el interior de la cuál debe incluirse la implementación manual del analizador léxico para **Tiny(0)**; (iii) una carpeta `implementacion_jflex`, en el interior de la cuál debe incluirse la implementación jflex del analizador léxico para **Tiny**; (iv) una carpeta `pruebas_tiny_0` con distintos programas de prueba que permiten probar el analizador léxico para **Tiny(0)**; y (v) una carpeta `pruebas_tiny` con distintos programas de prueba que permitan probar el analizador léxico para **Tiny**. La entrega debe ser realizada solamente por un miembro del grupo.
- A través del juez **DomJudge** de la asignatura se entregará la implementación manual del analizador léxico para Tiny(0), y la implementación JFlex del analizador léxico completo para Tiny. Ambos programas estarán acondicionados para su funcionamiento y prueba a través del juez.

Apéndice A

Tiny(0) es un subconjunto de **Tiny** que incluye únicamente las siguientes características:

- Declaraciones de variables, con tipos básicos **int**, **real** y **bool**.
- Únicamente instrucciones eval. Las expresiones asociadas incluyen únicamente:
    - Literales enteros, reales, y booleanos (**true** y **false**).
    - Variables.
    - Operadores aritméticos +,-,* y /, menos unario (-), los operadores lógicos **and**, **or** y **not**, los operadores relacionales (<,>,<=,>=,==,!=), y el operador de asignación (=).
- Las prioridades y asociatividades de estos operadores son las de **Tiny**. Como en **Tiny**, pueden utilizarse paréntesis para alterarlas.

Ejemplo de programa **Tiny(0)**

```
real peso;
bool pesado
&&
@ peso = (45.0 * 12e-56) / -2.05;
@ pesado = (peso >  10.0) or (peso / 2 <= +0.0)
```