Escuela de Ciencias de la Computación e Informática

Tarea programada 1

Computabilidad y Complejidad – CI0124

Prof. Maureen Murillo R.

I-2025

Descripción del problema

Usted dispone de un archivo de texto diseñado para simular un archivo de registro (log) de un sistema complejo que podría ser generado por una aplicación o sistema operativo real. Incluye una variedad de entradas de log. Este es el desglose de lo que representa:

1. **Comentarios de Encabezado:** El archivo comienza con un encabezado (líneas que comienzan con #) que explica su propósito, fecha y una breve descripción. Estas líneas son tratadas como comentarios y deben ser ignoradas o manejadas de manera diferente por el analizador léxico.
2. **Entradas de Log con Sello de Tiempo:** Cada entrada principal comienza con un sello de tiempo (por ejemplo, [2025-02-15 08:00:00]) seguido de un nivel de log como INFO, WARN, DEBUG o ERROR, y un mensaje breve. Esta estructura imita las entradas de log reales que registran cuándo ocurrió un evento y su gravedad.
3. **Estructuras Anidadas:** Después de la entrada principal del log, el archivo incluye varios bloques anidados (por ejemplo, BEGIN_BOOT_SEQUENCE ... END_BOOT_SEQUENCE). Estos bloques simulan procesos detallados y de múltiples pasos (como secuencias de arranque, diagnósticos, informes de fallos o copias de seguridad). Incluyen:
    - **Pares Clave-Valor:** Elementos como STEP: "Inicializar hardware"; donde un token (por ejemplo, STEP) se asocia con un valor.
    - **Bloques Anidados:** Bloques dentro de bloques (por ejemplo, el bloque BEGIN_MODULES_LOAD ... END_MODULES_LOAD dentro de una secuencia de arranque), lo que añade capas de complejidad.
    - **Arreglos y Objetos:** Estructuras similares a JSON, como arreglos (por ejemplo, ["codec1", "codec2"]) y objetos (por ejemplo, { status: "ok", version: "1.2.3" }).
4. **Variedad de Tokens y Delimitadores:** El archivo incluye una amplia gama de tokens, tales como:
    - **Identificadores y Palabras Clave:** Palabras como BEGIN_BOOT_SEQUENCE, STEP, MODULE, END_MODULES_LOAD, etc.
    - **Operadores y Delimitadores:** Caracteres como dos puntos (:), punto y coma (;), llaves ({ }), corchetes ([ ]), y comillas (").
    - Todos los tipos de tokens deben ser identificados por el analizador léxico.
5. **Escenarios Prácticos:** Cada entrada de log simula un aspecto diferente de las operaciones de un sistema:
    - **Secuencia de Arranque:** Pasos detallados de un sistema iniciándose.
    - **Comprobaciones Diagnósticas:** Verificación de módulos con diferentes resultados.
    - **Informe de Fallo:** Captura de detalles de error, trazas de pila e información de contexto.
    - **Proceso de Copia de Seguridad:** Muestra actualizaciones de progreso anidadas y detalles de archivos.

El objetivo principal de esta tarea es desarrollar un analizador léxico que pueda identificar correctamente tokens (como sellos de tiempo, identificadores, cadenas, números y símbolos) y un analizador sintáctico que pueda entender la estructura jerárquica de los bloques anidados.

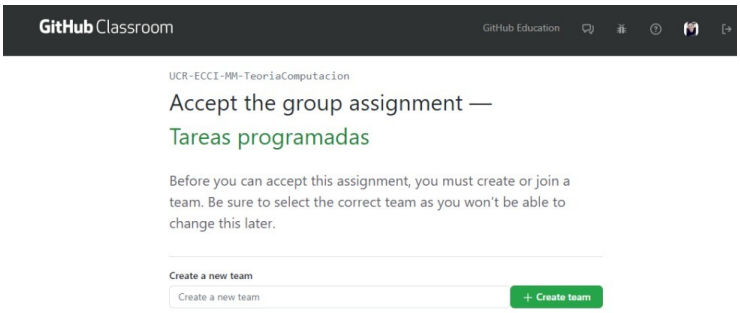
Estos analizadores deben ser parte de una aplicación cuya funcionalidad debe definirla usted, de tal forma que, utilizando todos los datos de entrada, el programa permita a los usuarios obtener en forma amigable información relevante. Los requisitos para su aplicación son:

1. Debe tener al menos tres opciones (funcionalidades) para el usuario.
2. Las funciones de su programa deben utilizar la mayor cantidad de datos del archivo.
3. Debe tener una interfaz gráfica de usuario (GUI) amigable y de fácil uso.
4. Como una cuarta opción su programa debe mostrar al menos un gráfico de datos.
5. Debe tener la mayor robustez posible en la validación de los datos de entrada, es decir, que los analizadores léxico y sintáctico identifiquen errores en el archivo de entrada.

Esta tarea debe ser resuelta en parejas o tríos. Deben usar PLY para crear el analizador léxico y sintáctico. Los analizadores cargarán toda la información necesaria a partir del archivo dado.

Entregables

Para trabajar las tareas programadas cada grupo deberá crear un repositorio en Github a través del siguiente enlace: https://classroom.github.com/a/cvusU2ZZ
El primer estudiante de cada equipo que accese este enlace será el que creará el repositorio y deberá usar el nombre del equipo que previamente se asignaron.



Cuando los demás miembros del equipo accesen el enlace anterior deberán unirse al repositorio que ya habrá creado el primer compañero. Una vez creado el repositorio, deben crear una carpeta para cada tarea programada, y dentro de la carpeta para la tarea 1 deben crear una subcarpeta para cada entregable, dentro de la cual quedarán los documentos que se revisarán.

Cada equipo deberá presentar los siguientes entregables:

1. (10%) Este primer avance debe incluir:
   a) Documento pdf con especificación de la funcionalidad de la aplicación y con la explicación de las estructuras de datos específicas que requerirá el programa para almacenar la información a partir del archivo de entrada.
   b) Una aplicación prototipo básica con GUI, en Python, con un menú con las opciones para la funcionalidad definida. Una de las opciones debe mostrar otra ventana con algún tipo de gráfico (barras, pie, etc) que muestre datos inventados. El objetivo es ir conociendo las herramientas de Python.

   Todos los miembros del grupo deberán exponer. **Fecha: Jueves 20 de marzo, 7:00am.**

2. (20%) Analizador léxico funcionando: en esta entrega la herramienta debe ser capaz de reconocer todos los tokens necesarios para procesar el archivo de entrada. **Fecha: Jueves 27 de marzo, 7:00am.**

3. (35%) Analizador sintáctico revisando errores pero sin crear objetos o estructuras de datos dinámicamente: en esta entrega se revisa que el archivo esté sintácticamente correcto y se imprime error en caso contrario. **Jueves 10 de abril, 7:00am.**

4. (15%) Creación dinámica de los objetos o las estructuras de datos: a la entrega anterior, se le agrega la capacidad de crear dinámicamente los objetos o estructuras necesarias para la aplicación. **Lunes 28 de abril, 7:00am.**

5. (20%) Implementación de la aplicación completa: entrega final con toda la funcionalidad. **Jueves 8 de mayo, 7:00am.**

Recomendaciones de trabajo

1. GitHub es un repositorio con control de versiones que permite gestionar proyectos especialmente en casos de trabajo en equipo. El objetivo de uso en este curso no es solamente para realizar las entregas de cada etapa, sino que además se espera que sea utilizado para dar seguimiento al desarrollo de los proyectos, tanto por parte de la docente, del asistente como también de parte de los mismos integrantes de cada equipo. En este sentido, es necesario que para cada etapa quede evidenciado el desarrollo y avance paulatino del trabajo (no solo la entrega final) mediante la planificación y asignación de responsables de cada actividad y la realización de *commits* regulares durante todo el proceso.

2. Tal como se indica en la siguiente sección, debe utilizarse la herramienta *Projects* de la plataforma GitHub para organizar y realizar la planificación detallada de las actividades de cada etapa. Se debe primero crear *issues* y luego utilizarlos en un tablero de *Projects*. Es aceptable si el equipo solo utiliza *issues* para esta planificación, sin embargo se recomienda incorporarlos en un tablero. El siguiente es un buen video para comprender mejor cómo funcionan estas herramientas, aunque no es necesario que apliquen todas las opciones que brindan: https://www.youtube.com/watch?v=m1_O7tIBCy4. La planificación debe ser detallada con actividades específicas, no solo generales. El siguiente ejemplo muestra una buena subdivisión del trabajo:

Aspectos evaluativos

1. La planificación **detallada** del trabajo del equipo (utilizando *Projects*), organizada con actividades específicas y no generales, e indicándose el responsable para cada una, debe realizarse para cada etapa. Tendrá un valor de **2 puntos en cada entregable**.

2. En el repositorio debe haber evidencia del **trabajo individual** de cada uno de los miembros del equipo, mediante la ejecución de *commits* **regulares** (no solo uno o dos *commits* por estudiante) que muestren el **desarrollo incremental** del proyecto. Estos *commits* deben realizarse con regularidad entre la fecha de entrega anterior y la próxima, es decir, no deben realizarse exclusivamente en los dos días anteriores a la fecha límite. Aquel estudiante que no haya realizado *commits* antes de la fecha y hora de entrega límite en las condiciones mencionadas, se considerará como que no trabajó en el proyecto y perderá los puntos del entregable. Deben respetar la fecha y hora límite de entrega indicada en el enunciado de la tarea.

3. Todos los integrantes del equipo deben participar en la exposición oral de cada etapa y deberán responder las preguntas de la profesora. Esta exposición oral tiene un valor de **3 puntos en cada entregable**.

4. En la carpeta de cada entregable debe haber un archivo *readme* que explique claramente cómo ejecutar el programa, incluyendo los paquetes que se requiere tener instalados. Este archivo dentro de la carpeta tendrá un valor de **1 punto en cada entregable**.