Fecha: 16 de octubre

Teoría de la Computación

Semestre II-2025

Tarea 3

Investigación, análisis, diseño e implementación (TID)

Objetivo de la tarea

Diseñar e implementar un analizador léxico y sintáctico para análisis de códigos fuentes en FORTRAN77, empleando una metodología de desarrollo de software que incluya las etapas de análisis, diseño, implementación y pruebas.

Objetivos Específicos

- Aplicar conceptos de gramáticas libres de contexto (GLC) para el caso.
- Diseñar un autómata o parser LL(1) o LR(0) que reconozca un subconjunto del lenguaje definido.
- Implementar un analizador léxico (tokenizador) que identifique correctamente los lexemas de entrada.
- Generar un árbol sintáctico o una traza de derivación para las entradas válidas.
- Integrar las etapas del análisis léxico y sintáctico en una aplicación funcional.

Descripción de la Tarea

Cada grupo deberá construir un programa (en **Python o C++ o C**) que implemente un analizador léxico y sintáctico para un lenguaje definido. Debe analizar gramáticas formales definidas (por ejemplo, expresiones aritméticas, asignaciones, estructuras condicionales o bucles simples, otros).

Metodología de trabajo

El desarrollo debe considerar una metodología de software (por ejemplo: Ágil, RUP o en cascada) con las siguientes fases:

1. Análisis de requisitos: Definir el tipo de lenguaje y sus reglas léxicas y sintácticas.
2. Diseño: Especificar la gramática formal, los tokens y la estructura del analizador (diagramas o pseudocódigo).
3. Implementación: Codificar el analizador léxico y sintáctico.
4. Pruebas: Crear casos de prueba válidos e inválidos para validar el funcionamiento.
5. Documentación: Redactar informe técnico describiendo el proceso y resultados.

Fecha: 16 de octubre

Entregables

1. Código fuente del analizador, debe ser indicado en un LINK en la sección **Anexo** que lleve a un Git. (recuerde invitar al Profesor)
2. Documento en formato PDF se sube al Item de la Tarea.

Presentación de Código

Presentación del código: Se realiza en hora MIXTA según llamado por Bloque.
(Un integrante a quien asigne deberá mostrar el código, si los otros le acompañan es opcional)

Recomendaciones adicionales

- Validar las expresiones con una tabla LL(1) o mediante un parser recursivo descendente.
- Aplicar control de versiones (GitHub o GitLab) para registrar avances.
- Documentar cada etapa del desarrollo en el informe final.
- Implementar detección de errores léxicos y sintácticos.
- Generar un árbol sintáctico visual.
- Añadir una interfaz gráfica simple.

Ver en plataforma educa:

- ✓ Carpeta: Formatos y rúbricas
- ✓ Plazo

Prof. M. Lévano,

,