UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERIA
ESCUELA DE CIENCIAS Y SISTEMAS

PROGRAMA DEL CURSO DE LENGUAJES FORMALES DE PROGRAMACION

CODIGO:	796	**CRÉDITOS:**	3
ESCUELA:	Ciencias y sistemas	**ÁREA A LA QUE PERTENECE**	Ciencias de la Computación
PRE REQUISITO:	770, 795, 960	**POST REQUISITO**	777, 964
CATEGORÍA:	Obligatorio	**SECCION:**	
HORAS POR SEMANA DEL CURSO:	2	**HORAS POR SEMANA DEL LABORATORIO:**	2
DÍAS QUE SE IMPARTE EL CURSO:	Martes	**DÍAS QUE SE IMPARTE EL LABORATORIO:**	Viernes
HORARIO DEL CURSO:		**HORARIO DEL LABORATORIO:**	

I. Descripción General

Este curso busca introducir al estudiante con los fundamentos teóricos matemáticos y conceptos que fundamentan los lenguajes de programación. El estudiante debe adquirir la base teórica necesaria y requerida para que pueda llevar un curso avanzado de lenguajes y compiladores.

Se busca, además, definir los modelos matemáticos asociados a la representación de los diferentes tipos de lenguajes para luego implementar estos conceptos en lenguajes de programación.

Es de primordial importancia que pueda reconocer cualquier tipo de gramática, pero sobre todo, pueda manejar y diseñar gramáticas para lenguajes regulares y para lenguajes libres de contexto, además, de los modelos matemáticos que las resuelven. Se busca que el estudiante tenga mucha práctica en el diseño de gramáticas para representar lenguajes y que adquiera la habilidad de diseñarlas sin problema. Adquiriendo conceptos y los pueda relacionar a los aspectos técnicos y prácticos conociendo su aplicación en lenguajes reales conocidos. El estudiante debe aprender la teoría que esta atrás de los diferentes componentes de un compilador, las técnicas de programación usadas para poner esta teoría en práctica. El curso se enfoca y trata con profundidad la teoría de autómatas finitos buscando que el estudiante entienda el proceso matemático para encontrar los autómatas finitos y su implementación en un lenguaje de programación.

II. Objetivos

- **Objetivo Generale**
  - Que el estudiante tenga los conceptos teóricos y matemáticos necesarios que fundamentan los lenguajes de programación y el diseño de lenguajes y compiladores.
- **Objetivos Específicos**

  Al final del curso el estudiante deberá:
  1. Definir cualquier lenguaje formal
  2. Reconocer las características que identifican a cualquier tipo de gramática.

3. Manejar la terminología de los lenguajes y compiladores.
4. Conocer el modelo matemático que resuelve cada tipo de gramática.
5. Conocer el funcionamiento de un analizador léxico y su implementación
6. Conocer e implementar maquinas de estado finito
7. Diseñar e implementar gramáticas libres de contexto
8. conocer los conceptos que fundamentan el análisis sintáctico
9. Aplicar los modelos matemáticos que resuelven gramáticas de tipo 0

IV. Evaluación del Rendimiento Académico:

La nota final estará compuesta de 100 puntos, distribuidos de la siguiente manera:

Tres Evaluaciones de Rendimiento (15 pts c/u)	**45 puntos**
Tareas, trabajo en clase, asistencia, comprobaciones, etc.	**10 puntos**
Laboratorio (Proyectos, Practicas, etc.)	**20 puntos**
Zona	**75 puntos**
Examen Final	**25 puntos**
Total	**100 puntos**

- Página oficial de la asignatura:
http://ecys.ingenieria-usac.edu.gt/UV

VII. CONTENIDO PROGRAMATICO

Contenido

Unidad 1. Lenguajes Formales

1. Definiciones
   Lenguajes de programación, Compiladores e interpretes
   Generaciones de lenguajes
   Partes del compilador: Análisis sintáctico
   Análisis léxico
   Definición de tokens, lexemas, palabras reservadas.
2. Lenguajes formales
   - 2.1 Definición
   - 2.2 Símbolos terminales
   - 2.3 Símbolos no terminales
   - 2.4 Gramática
   - 2.5 Estado inicial

Unidad 2 : Jerarquia de Chomsky

3. Jerarquía de Chomsky
   - 3.1 Lenguajes Recursivamente enumerarbles
     Maquinas de turing
   - 3.2 Lenguajes Sensibles al Contexto
     Autómatas lineales limitados
   - 3.3 Lenguajes Libres de contexto
     Autómatas Descendentes
   - 3.4 Lenguajes regulares
     Autómatas finitos

Unidad 3: Lenguajes regulares

1. Lenguajes regulares
2. Gramáticas Regulares (tipo 3)
3. Diseño de gramáticas regulares
   - 3.1 Ejemplos y ejercicios de gramáticas regulares
   - 3.2 Aplicación de gramáticas regulares
4. Expresiones regulares
5. Autómatas Finitos

5.1 Grafos para representación de autómatas
5.2 Autómatas Finitos No Determinísticos NFA
5.3 Autómatas Finitos Determinísticos DFA

6. Método para pasar de gramáticas a expresiones regulares y de expresiones regulares a gramáticas
7. Métodos para Calcular DFAs
   7.1 Construcción de Thomson y minimización de estados
   7.2 Método del árbol
8. Ejemplos y ejercicios

Unidad 4: Lenguajes libres de contexto

1. Lenguajes Libres de contexto
2. Gramáticas Libres de contexto (Tipo 2)
3. Diseño de gramáticas libres de contexto
   3.1 Ejemplos y ejercicios de gramáticas libres de contexto
   3.2 Aplicación de lenguajes libres de contexto
4. Recursividad por la izquierda y recursividad por la derecha
5. Gramáticas ambiguas
7. Parser recursivos descendentes
10. Ejemplos y ejercicios

VIII. BIBLIOGRAFIA

- Andrew W. Appel “Modern Compiler Implementation in Java” Second Edition Cambridge University Press

- Brookshear, J. Glenn. Teoría de la Computación - Lenguajes formales, autómatas y complejidad. Addison-Wesley Iberoamericana.

- Appleby, Doris. Vandekopple, Julius J. Lenguajes de Programación - Paradigma y práctica. Editorial McGraw Hill, México, 1,