UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
Escuela de Ciencias y Sistemas

FICHA TÉCNICA DEL CURSO: Lenguajes Formales y de Programación

<table>
<tr><td>No.</td><td colspan="3">Descripción</td></tr>
<tr><td>1</td><td>Código 796</td><td colspan="2">Créditos 3</td></tr>
<tr><td>2</td><td>Escuela
Ciencias y Sistemas
https://dtt-ecys.org</td><td>Área a la que pertenece
Ciencias de la computación</td><td>Vigencia
Segundo Semestre 2022</td></tr>
<tr><td>3</td><td>2 horas por semana</td><td colspan="2">Horario: Martes 07:10 a 08:50 am.</td></tr>
<tr><td>4</td><td>Pre-requisitos:
770 Introducción a la Programación 1
795 Lógica de sistemas
960 Matemática de Cómputo 1</td><td colspan="2">Postrequistios:
777 Organización de lenguajes y compiladores 1
772 Estructuras de datos</td></tr>
<tr><td>5</td><td colspan="3">Secciones: A+, A-, B+, B-</td></tr>
<tr><td>6</td><td colspan="3">I. Descripción General

Este curso busca introducir al estudiante con los fundamentos teóricos matemáticos y conceptos que fundamentan los lenguajes de programación.
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 libres del contexto, además, de los modelos matemáticos que las resuelven. Adquiriendo conceptos y los pueda relacionar a los aspectos técnicos y prácticos conociendo su aplicación en lenguajes reales conocidos. Y como estos conceptos son base introductoria a los compiladores.
Al finalizar el curso el estudiante estará en la capacidad de comprender la funcionalidad de los compiladores.

II. Objetivos
Objetivo General
Que el estudiante conozca los conceptos teóricos y matemáticos necesarios que fundamentan los lenguajes formales y de programación; mediante la clasificación de gramáticas, y el diseño de lenguajes mediante autómatas, expresiones y gramáticas.

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 formales y gramáticas.
4. Conocer el modelo matemático que resuelve cada tipo de gramática.
5. Diseñar gramáticas que representen lenguajes específicos.
6. Conocer e implementar máquinas de estado finito.
7. Diseñar e implementar gramáticas regulares.
8. Reconocer las fases de un compilador.
9. Desarrollar un analizador léxico con base a los autómatas finitos.
10. Desarrollar un analizador sintáctico por medio de las gramáticas libres del contexto.</td></tr>
</table>

III. Contenido

1. Unidad 1: Introducción (Capítulo 1, libro: Compiladores de Aho) 1.1. Procesadores de lenguaje 1.2. La estructura de un compilador 1.2.1. Análisis léxico 1.2.2. Análisis sintáctico 1.2.3. Análisis semántico 1.2.4. Generación de código intermedio 1.2.5. Optimización de código 1.2.6. Generación de código 1.2.7. Administración de la tabla de símbolos 1.2.8. El agrupamiento de fases en pasadas 1.2.9. Herramientas de construcción de compiladores 1.3. La evolución de los lenguajes de programación 1.3.1. El avance a los lenguajes de alto nivel 1.3.2. Impactos en el compilador 1.4. La ciencia de construir un compilador 1.4.1. Modelado en el diseño e implementación de compiladores 1.4.2. Aplicaciones de la tecnología de compiladores 1.5. Lenguajes formales 1.5.1. Definiciones (capítulo 1, libro de Linz) 1.5.2. Jerarquía de Chomsky (capítulo 11, libro de Linz)	4 períodos
2. Unidad 2: Análisis léxico (Capítulo 3, libro: Compiladores de Aho) 2.1. La función del analizador léxico 2.1.1. Comparación entre análisis léxico y análisis sintáctico 2.1.2. Tokens, patrones y lexemas 2.1.3. Atributos para los tokens 2.1.4. Errores léxicos 2.2. Uso de búfer en la entrada 2.2.1. Pares de búferes 2.2.2. Centinelas 2.3. Especificación de los tokens 2.3.1. Cadenas y lenguajes 2.3.2. Operaciones en los lenguajes 2.3.3. Definiciones regulares (Gramáticas regulares) 2.3.4. Expresiones regulares 2.3.5. Extensiones de las expresiones regulares 2.4. Reconocimiento de tokens 2.4.1. Diagrama de transición de estados 2.4.2. Reconocimiento de las palabras reservadas y los identificadores 2.4.3. Finalización del bosquejo 2.4.4. Arquitectura de un analizador léxico basados en diagramas 2.5. Autómatas finitos 2.5.1. Autómatas finitos no deterministas 2.5.2. Tablas de transición	12 períodos **Parcial 1** **16-Ago.**

2.5.3. Aceptación de las cadenas de entrada mediante los autómatas 2.5.4. Autómatas finitos deterministas 2.6. De las expresiones regulares a los autómatas 2.6.1. Conversión de un AFN a AFD 2.6.2. Simulación de un AFN 2.6.3. Eficiencia de la simulación de un AFN 2.6.4. Construcción de una AFN a partir de una expresión regular 2.6.5. Eficiencia de los algoritmos de procesamiento de cadenas 2.7. Diseño de un generador de analizadores léxicos 2.7.1. La estructura del analizador generado 2.7.2. Coincidencia de patrones con base en los AFNs 2.7.3. AFD para analizadores léxicos 2.7.4. Implementación del operador de preanálisis 2.8. Optimización de los buscadores por concordancia de patrones basados en AFD 2.8.1. Estados significativos de una AFN 2.8.2. Funciones calculadas a partir del árbol sintáctico 2.8.3. Cálculo de anulable, primerapos y ultimapos 2.8.4. Cálculo de siguientepos 2.8.5. Conversión directa de una expresión regular a un AFD 2.8.6. Minimización del número de estados de un AFD 2.8.7. Minimización de estados en los analizadores léxicos 2.8.8. Intercambio de tiempo por espacio en la simulación de un AFD	**Parcial 2** **20-Sep.**
3. Unidad 3: Análisis sintáctico (Capítulo 2, libro: Compiladores de Aho) 3.1. Introducción 3.2. Definición de sintaxis 3.2.1. Definición de gramáticas 3.2.2. Derivaciones 3.2.3. Árboles de análisis sintáctico 3.2.4. Ambigüedad 3.2.5. Asociatividad de los operadores 3.2.6. Precedencia de los operadores 3.3. Autómata de Pila 3.3.1 Definición 3.3.2 Notación gráfica 3.3.3 Diseño de autómatas de pila 3.3.4 Teorema 2.2: Generar AP desde una Gramática tipo 2 3.4. Análisis sintáctico 3.5. Tabla de símbolos	6 períodos **Parcial 3** **18-Oct.**

IV. Metodología:

El curso se desarrollará intercalando clases magistrales para la exposición de conceptos nuevos y clases participativas, en las que se espera que el estudiante realice las lecturas, tareas o ejercicios dejados para realizar fuera de clase, previo al inicio de un nuevo día de clase.

V. Evaluación:

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

3 Evaluaciones de rendimiento (15 puntos c/u).....	45 puntos
Tareas, trabajos en clase, comprobaciones, asistencia, cortos, etc.	10 puntos
Laboratorio (proyectos, prácticas, etc.).....	20 puntos
Evaluación Final.....	25 puntos

Nota Total.....	100 puntos

VI. Observaciones:

- Será necesario contar con un 80% de asistencia y aprobar el laboratorio del curso con una nota mínima de 61 puntos, para tener derecho a la evaluación final.
- Se realizarán exámenes cortos cada día de clase y se debe tener un 80% de exámenes realizados, para tener derecho a la evaluación final.
- En este curso, no se pasan notas de semestres anteriores, no se guardan notas para semestres posteriores, y no se aceptan estudiantes con problemas de prerrequisitos.

7	Bibliografía	1. Aho, A. V., Lam, M. S., Sethi, R., & Ullman, J. D. (2007). **Compilers: Principles, Techniques & Tools** (2nd ed.). Pearson. 2. Linz, P. (2017). **An Introduction to Formal Languages and Automata** (6th ed.). Jones & Bartlett Learning. 3. Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2007). **Introduction Automata Theory, Languages and Computation** (3rd ed.). Pearson.
8	No. De Secciones	4
9	Catedráticos titulares	Sección A+ Ing. Otto Rodríguez Sección B+ Ing. David Morales Sección A- Inga. Damaris Campos – damaris.campos@ingenieria.usac.edu.gt Sección B- Inga. Zulma Aguirre – zaguirre@ingenieria.usac.edu.gt
10	Director de Escuela	**Ing. Carlos Alonzo**