

FICHA TÉCNICA DEL CURSO: Lenguajes Formales y de Programación

No.	Descripción	
.	<b>Código</b> 796	<b>Créditos</b> 3
1	<b>Escuela</b> Ciencias y Sistemas <a href="https://dtc-ecys.org">https://dtc-ecys.org</a>	<b>Área a la que pertenece</b> Ciencias de la computación  <b>Vigencia</b> Primer Semestre 2021
2	<b>Horas por semana</b> 2	<b>Horario</b> Martes 7:10 a 8:50
3	Pre-requisitos: 770 introducción a la Programación 1 795 lógica de sistemas 960 Matemática de Cómputo 1	
4	Post requisitos: 777 Organización de lenguajes y compiladores 1 772 Estructuras de datos	
5	Secciones: A+, A-, B+, B-	
6	<p><b>I. Descripción General</b></p> <p>Este curso busca introducir al estudiante con los fundamentos teóricos matemáticos y conceptos que fundamentan los lenguajes de programación. Busca introducir a los estudiantes a los conceptos de lenguajes y compiladores.</p> <p>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.</p> <p>Es de primordial importancia que pueda reconocer cualquier tipo de gramática, pero, sobre todo, pueda manejar y diseñar gramáticas para lenguajes regulares, 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.</p> <p>Al finalizar el curso el estudiante estará en la capacidad de poder recibir y comprender sin problema un curso avanzado de compiladores.</p> <p><b>II. Objetivos</b></p> <p><b>Objetivo General</b> Que el estudiante conozca los conceptos teóricos y matemáticos necesarios que fundamentan los lenguajes formales y de programación así como los compiladores; mediante la clasificación de gramáticas, y el diseño de lenguajes mediante autómatas, expresiones y gramáticas.</p> <p><b>Objetivos Específicos</b></p> <p>Al final del curso el estudiante deberá:</p> <ul style="list-style-type: none"> <li>• Definir cualquier lenguaje formal</li> <li>• Reconocer las características que identifican a cualquier tipo de gramática.</li> <li>• Manejar la terminología de los lenguajes formales y gramáticas.</li> <li>• Conocer el modelo matemático que resuelve cada tipo de gramática.</li> <li>• Diseñar gramáticas que representen lenguajes específicos.</li> <li>• Conocer e implementar máquinas de estado finito.</li> <li>• Diseñar e implementar gramáticas regulares.</li> </ul>	

### III. Contenido

Contenido	Planificación
<p><b>1. Unidad 1: Introducción</b></p> <ul style="list-style-type: none"><li>1.1. Procesadores de lenguaje</li><li>1.2. La estructura de un compilador<ul style="list-style-type: none"><li>1.2.1. Análisis léxico</li><li>1.2.2. Análisis sintáctico</li><li>1.2.3. Análisis semántico</li><li>1.2.4. Generación de código intermedio</li><li>1.2.5. Optimización de código</li><li>1.2.6. Generación de código</li><li>1.2.7. Administración de la tabla de símbolos</li><li>1.2.8. El agrupamiento de fases en pasadas</li><li>1.2.9. Herramientas de construcción de compiladores</li></ul></li><li>1.3. La evolución de los lenguajes de programación<ul style="list-style-type: none"><li>1.3.1. El avance a los lenguajes de alto nivel</li><li>1.3.2. Impactos en el compilador</li></ul></li><li>1.4. La ciencia de construir un compilador<ul style="list-style-type: none"><li>1.4.1. Modelado en el diseño e implementación de compiladores</li><li>1.4.2. Lenguajes formales</li><li>1.4.3. Definiciones</li><li>1.4.4. Jerarquía de Chomsky</li></ul></li></ul>	
<p><b>2. Unidad 2: Análisis léxico</b></p> <ul style="list-style-type: none"><li>2.1. La función del analizador léxico<ul style="list-style-type: none"><li>2.1.1. Comparación entre análisis léxico y análisis sintáctico</li><li>2.1.2. Tokens, patrones y lexemas</li><li>2.1.3. Atributos para los tokens</li><li>2.1.4. Errores léxicos</li></ul></li><li>2.2. Uso de búfer en la entrada<ul style="list-style-type: none"><li>2.2.1. Pares de búferes</li><li>2.2.2. Centinelas</li></ul></li><li>2.3. Especificación de los tokens<ul style="list-style-type: none"><li>2.3.1. Cadenas y lenguajes</li><li>2.3.2. Operaciones en los lenguajes</li><li>2.3.3. Definiciones regulares (Gramáticas regulares)</li><li>2.3.4. Expresiones regulares</li><li>2.3.5. Extensiones de las expresiones regulares</li></ul></li><li>2.4. Reconocimiento de tokens<ul style="list-style-type: none"><li>2.4.1. Diagrama de transición de estados</li><li>2.4.2. Reconocimiento de las palabras reservadas y los identificadores</li><li>2.4.3. Finalización del bosquejo</li><li>2.4.4. Arquitectura de un analizador léxico basados en diagramas</li></ul></li><li>2.5. Automatas finitos<ul style="list-style-type: none"><li>2.5.1. Automatas finitos no deterministas</li><li>2.5.2. Tablas de transición</li><li>2.5.3. Aceptación de las cadenas de entrada mediante los autómatas</li><li>2.5.4. Automatas finitos deterministas</li></ul></li><li>2.6. De las expresiones regulares a los autómatas<ul style="list-style-type: none"><li>2.6.1. Conversión de un AFN a AFD</li><li>2.6.2. Simulación de un AFN</li></ul></li></ul>	<ul style="list-style-type: none"><li>• <b>Primera evaluación</b></li></ul>

	<ul style="list-style-type: none"> <li>2.6.3. Eficiencia de la simulación de un AFN</li> <li>2.6.4. Construcción de una AFN a partir de una expresión regular</li> <li>2.6.5. Eficiencia de los algoritmos de procesamiento de cadenas</li> <li>2.7. Diseño de un generador de analizadores léxicos <ul style="list-style-type: none"> <li>2.7.1. La estructura del analizador generado</li> <li>2.7.2. Coincidencia de patrones con base en los AFNs</li> <li>2.7.3. AFD para analizadores léxicos</li> <li>2.7.4. Implementación del operador de preanálisis</li> </ul> </li> <li>2.8. Optimización de los buscadores por concordancia de patrones basados en AFD <ul style="list-style-type: none"> <li>2.8.1. Estados significativos de una AFN</li> <li>2.8.2. Funciones calculadas a partir del árbol sintáctico</li> <li>2.8.3. Cálculo de anulable, primera posición y última posición</li> <li>2.8.4. Cálculo de siguiente posición</li> <li>2.8.5. Conversión directa de una expresión regular a un AFD</li> <li>2.8.6. Minimización del número de estados de un AFD</li> <li>2.8.7. Minimización de estados en los analizadores léxicos</li> <li>2.8.8. Intercambio de tiempo por espacio en la simulación de un AFD</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• <b>Segunda evaluación</b></li> </ul>
	<p><b>3. Unidad 3: Introducción a la sintaxis</b></p> <ul style="list-style-type: none"> <li>3.1. Introducción</li> <li>3.2. Definición de sintaxis <ul style="list-style-type: none"> <li>3.2.1. Definición de gramáticas</li> <li>3.2.2. Derivaciones</li> <li>3.2.3. Árboles de análisis sintáctico</li> <li>3.2.4. Ambigüedad</li> <li>3.2.5. Asociatividad de los operadores</li> <li>3.2.6. Precedencia de los operadores</li> </ul> </li> <li>3.3. Traducción orientada a la sintaxis <ul style="list-style-type: none"> <li>3.3.1. Notación prefija</li> <li>3.3.2. Atributos sintetizados</li> <li>3.3.3. Definiciones simples orientadas a la sintaxis</li> <li>3.3.4. Recorrido de árboles</li> <li>3.3.5. Esquemas de traducción</li> </ul> </li> </ul>	
	<p><b>4. Unidad 4: Análisis sintáctico</b></p> <ul style="list-style-type: none"> <li>4.1.1. Análisis sintáctico tipo arriba-abajo</li> <li>4.1.2. Análisis sintáctico predictivo</li> <li>4.1.3. Cuando usar las producciones épsilon</li> <li>4.1.4. Diseño de un analizador sintáctico predictivo</li> <li>4.1.5. Recursividad por la izquierda</li> <li>4.2. Un traductor para las expresiones simples <ul style="list-style-type: none"> <li>4.2.1. Sintaxis abstracta y concreta</li> <li>4.2.2. Adaptación del esquema de traducción</li> <li>4.2.3. Procedimientos para los no terminales</li> <li>4.2.4. Simplificación del traductor</li> <li>4.2.5. El programa completo</li> </ul> </li> <li>4.3. Tabla de símbolos <ul style="list-style-type: none"> <li>4.3.1. Tabla de símbolos por alcance</li> <li>4.3.2. El uso de las tablas de símbolos</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• <b>Tercera evaluación</b></li> </ul>

#### 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:

Descripción	Puntos
3 evaluaciones de rendimiento (15 puntos c/u).	45
Tareas, trabajos en clase, comprobaciones, asistencia, cortos, etc.	10
Laboratorio (incluye 2 proyectos de programación.)	20
Evaluación Final	25
Nota total	100

#### 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 debe tener un 80% actividades realizadas en clase.
- 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	<ol style="list-style-type: none"><li>1. Alfred V. Aho, Mónica S. Lam, Ravi Sethi, Jeffrey D. Ullman - <b>Compiladores: Principios, Técnicas y Herramientas</b> Pearson Addison Wesley 2006</li><li>2. Peter Linz <b>Una Introducción a los Lenguajes Formales y Autómatas</b> [ 6th ed. Jones &amp; Bartlett 2017</li><li>3. John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman <b>Introducción a la Teoría de Autómatas y lenguajes y Computación</b> Prentice Hall. 2006</li></ol>
8	No. De Secciones	4
9	Catedráticos	Sección A+ Ing. Otto Rodríguez Sección B+ Ing. David Morales Sección A- Inga. Damaris Campos Sección B- Inga. Zulma Aguirre
10	Director de Escuela	<b>Ing. Carlos Alonzo</b>