

FICHA TÉCNICA DEL CURSO: ORGANIZACIÓN DE LENGUAJES Y COMPILADORES 1

No.	Descripción		
	Código 777	Créditos 4	
1	Escuela Ciencias y Sistemas	Área a la que pertenece Ciencias de la Computación	Vigencia Segundo Semestre 2025
2	Horas por semana: 4 horas	Horario Secciones B y C; Martes y Jueves 07:10 – 08:50 Sección N: ; Martes y Jueves 17:20 – 19:00	
3	Prerrequisitos: 771 (Introducción a la Programación y Computación 2) 796 (Lenguajes Formales y de Programación) 962 (Matemática de Computo 2)		
4	Post-requisitos: 781 (Organización de Lenguajes y Compiladores 2)		
5	Sección: A, B y C		
6	I. Descripción General Este curso estudia los principios básicos de un compilador y / o intérprete, partiendo de la estructura interna del proceso de compilación, y describiendo las fases de este proceso. Se tratan en detalle las primeras fases del proceso: análisis lexicográfico, análisis sintáctico y traducción dirigida por la sintaxis. Para poner en práctica los conceptos aprendidos se realizan varias tareas y proyectos prácticos. II. Objetivos <i>Objetivo General</i> Que el estudiante adquiera una base teórica fundamental para el entendimiento de la estructura interna del proceso de compilación. <i>Objetivos Específicos</i> • Que el estudiante aprenda con detalle las primeras fases del proceso de compilación, principalmente el análisis lexicográfico y el análisis sintáctico. • Capacitar y ejercitar al estudiante en los principios del análisis, diseño e implementación de compiladores, para lo cual se realizarán varias tareas y proyectos III. Contenido I. Introducción a. Procesadores de lenguaje b. Estructura de un compilador i. Análisis léxico ii. Análisis sintáctico iii. Análisis semántico iv. Generación de código intermedio v. Optimización de código vi. Generación de código c. Evolución de los lenguajes de programación i. Avance a los lenguajes de alto nivel ii. Impacto de un compilador d. La ciencia de construir un compilador i. Modelado en el diseño e implementación de compiladores ii. La ciencia de la optimización de código e. Aplicaciones de la tecnología de compiladores i. Implementación de lenguajes de programación ii. Optimización arquitectura computadoras iii. Diseño de nuevas arquitecturas iv. Traducciones de programas v. Herramientas de productividad de software f. Fundamentos de los lenguajes de programación i. Distinción entre estático y dinámico ii. Entornos y estados iii. Alcance estático y estructura de bloques iv. Control de acceso explícito v. Alcance dinámico vi. Mecanismo para el paso de parámetros vii. Uso de alias II. Análisis léxico a. La función del analizador léxico i. Tokens, patrones y lexemas ii. Atributos de los tokens iii. Errores léxicos b. Uso de buffer en la entrada i. Pares de búferes y centinelas c. Especificación de los tokens i. Cadenas y lenguajes ii. Operaciones en los lenguajes iii. Expresiones regulares iv. Definiciones regulares v. Extensiones de las expresiones regulares d. Reconocimiento de tokens i. Diagrama de transición de estados ii. Reconocimiento de palabras reservadas e identificadores		

- iii. Finalización del bosquejo
 - iv. Arquitectura de un analizador léxico
 - e. Autómatas finitos (Práctico)
 - i. Autómatas finitos no deterministas
 - ii. Tablas de transiciones
 - iii. Aceptación de las cadenas de entrada mediante los autómatas
 - iv. Autómatas finitos deterministas
 - f. De las expresiones regulares a los autómatas (Práctico)
 - i. Conversión de un AFN a AFD
 - ii. Simulación de un AFN
 - iii. Eficiencia de la simulación de un AFN
 - iv. Construcción de una AFN a partir de una expresión regular
 - g. Diseño de un generador de analizadores léxicos
 - i. La estructura del analizador generado
 - ii. Coincidencia de patrones con base en los AFN
 - iii. AFD para analizadores léxicos
 - iv. Implementación del operador de preanálisis
 - h. Optimización de los buscadores de concordancia
 - i. Estados significativos de una FN
 - ii. Funciones calculadas a partir del árbol sintáctico
 - iii. Cálculo de anulable, primerapos, y ultimapos
 - iv. Cálculo de siguientes
 - v. Conversión directa de una expresión regular a un AFD
 - vi. Minimización del número de estados de una AFD
 - vii. Minimización de estados en los analizadores léxicos
 - viii. Intercambio de tiempo por espacio en la simulación de un AFD
- III. Análisis sintáctico
 - a. Introducción
 - i. La función del analizador sintáctico
 - ii. Representación de gramáticas
 - iii. Manejo de errores sintácticos
 - iv. Estrategias para recuperarse de los errores
 - b. Gramáticas libres de contexto
 - i. Definición formal y notación
 - ii. Árboles de análisis sintáctico y derivaciones
 - iii. Ambigüedad
 - iv. Verificación del lenguaje generado
 - v. Comparación entre gramáticas y expresiones
 - c. Escritura de una gramática
 - i. Comparación entre análisis léxico y sintáctico
 - ii. Eliminación de la ambigüedad
 - iii. Eliminación de la recursividad por la izquierda
 - iv. Factorización por la izquierda
 - v. Construcción de lenguajes no tipo 2
 - d. Análisis sintáctico descendente
 - i. Análisis sintáctico de descenso recursivo
 - ii. Primero y siguiente
 - iii. Gramáticas LL(1)
 - iv. Análisis sintáctico predictivo no recursivo
 - v. Recuperación de errores en el análisis sintáctico predictivo
 - e. Análisis sintáctico ascendente
 - i. Reducciones
 - ii. Poda
 - iii. Análisis sintáctico de desplazamiento-reducción
 - iv. Conflictos
 - f. Introducción al análisis sintáctico LR Simple
 - i. Elementos y el autómata LR(0)
 - ii. Algoritmo de análisis sintáctico LR
 - iii. Construcciones de tablas de análisis sintáctico SLR
 - iv. Prefijos viables
 - g. Analizadores sintácticos LR poderosos
 - i. Elementos LR(1) canónicos
 - ii. Construcción de conjuntos de elementos LR(1)
 - iii. Tablas de análisis sintáctico LR(1) canónico
 - iv. Construcción de tablas de análisis sintáctico LALR
 - v. Construcción eficiente de tablas LALR
 - vi. Compactación de tablas LR
 - h. Uso de gramáticas ambiguas
 - i. Precedencia y asociatividad para resolver conflictos
 - ii. Ambigüedad del else colgante
 - iii. Recuperación de errores en el análisis sintáctico LR

IV. Metodología

Clase Virtual para explicación de teoría.

Resolución de tareas, problemas y autoestudio

Tareas de investigación

Proyectos de programación

V. Evaluación:

Clase (76 puntos)		
	3 Exámenes parciales (15 puntos c/u)	45
	Examen final	25
	Tareas y cortos	06
	Total Clase	76
Laboratorio (24 puntos)		
	2 Proyectos	24
	Total Laboratorio	24

VI. Observaciones

El curso y el laboratorio se aprueban con 61 puntos.

Sitio del curso: <http://ecys.ingenieria-usac.edu.gt/UV/index.php>

Las notas y tareas del curso también serán publicadas en UEDI.

Para tener derecho a exámenes parciales y final es necesario cumplir con el 80% de la asistencia, así como para solicitar reposición de exámenes parciales (sólo se da reposición de exámenes parciales).

Exámenes Parciales Jueves en horario de clase

Primer Examen Parcial Jueves 14/Agosto/2025 en horario de clase* (Día y Hora propuestos pueden cambiar)**

Incluye

- i. Introducción a la compilación
- ii. Análisis Léxico
- iii. De las expresiones regulares a los autómatas

Segundo Examen Parcial Jueves 18/Septiembre/2025 en horario de clase* (Día y Hora propuestos pueden cambiar)**

Incluye

- iv. Análisis Léxico
- v. Análisis Sintáctico
- vi. Análisis sintáctico descendente

Tercer Examen Parcial Jueves 16/Octubre/2025 en horario de clase* (Día y Hora propuestos pueden cambiar)**

Incluye

- vii. Análisis Sintáctico
- viii. Introducción al análisis sintáctico LR Simple
- ix. Analizadores sintácticos LR poderosos - LR(1)

7	Bibliografía	Libro de Texto Compiladores principios, técnicas y herramientas Aho, lam, Sethi y Ullman. (2008).. 2 ed. (Capítulos 1, 3 y 4)
9	Catedráticos titulares	Ing. Manuel Castillo Ing. Keving Lajpop Ing. Mario Bautista