UNIVERSIDAD AUTÓNOMA DE CHIHUAHUA UNIVERSIDAD AUTÓNOMA DE CHIHUAHUA **Programa analítico de la unidad de aprendizaje:** **Compiladores interpretes**	**DES:**	Ingeniería
	Programa(s) Educativo(s):	Ingeniería en Ciencias de la Computación
	Tipo de materia:	Obligatoria
	Clave de la materia:	CI876
	Semestre:	8°
	Área en plan de estudios:	Ciencias de la Ingeniería
	Créditos	4
	Total de horas por semana:	4
	Teoría:	3
	Práctica	0
	Taller:	0
	Laboratorio:	1
	Prácticas complementarias:	0
	Trabajo extra clase:	0
	Total de horas semestre:	64
	Fecha de actualización:	Febrero 2023
	Materia requisito:	Teoría de la Computación

PROPÓSITO DEL CURSO

Los lenguajes de programación son las herramientas fundamentales para el desarrollo e implementación de soluciones computacionales, para el diseño de estas soluciones, los lenguajes utilizan elementos como compiladores e intérpretes que transforman las instrucciones dadas en lenguaje de alto nivel a instrucciones comprensibles para la computadora. El curso le aporta al estudiante las bases para eficientar sus habilidades de programación y crear analizadores de texto (preámbulos a un compilado), haciendo uso de la lógica, estructura de datos y conceptos abstractos, que son los elementos fundamentales de los compiladores e intérpretes. Esta asignatura tiene vinculación directa con Estructuras de Datos, Teoría de la Computación y Lenguajes de Programación.

COMPETENCIAS (Tipo Y Nombre de la competencias que nutre la materia y a las que contribuye)	**DOMINIOS COGNITIVOS** (Objetos de estudio, temas y subtemas)	**RESULTADOS DE APRENDIZAJE.** (Por objeto de estudio).
El curso promueve las siguientes competencias: **Básicas:** **COMUNICACIÓN** Utiliza diversos lenguajes y fuentes de información para comunicarse efectivamente) **SOLUCIÓN DE PROBLEMAS** Analiza las diferentes componentes de un problema y sus interrelaciones.	Introducción 1.1.- lenguajes de programación. 1.1.1 clasificación de los lenguajes de programación 1.1.1.1 según su grado de independencia de la máquina. 1.1.1.2 según la forma de sus instrucciones 1.1.1.3 por generaciones. 1.1.2 Ventaja de los lenguajes de alto nivel 1.1.3 Inconvenientes de los lenguajes de alto nivel 1.1.4 Otros lenguajes 1.2.- Procesadores de lenguaje 1.2.1 Traductores	Clasifica los procesadores de lenguajes de programación en base a la filosofía que emplean para generar código ejecutable en la computadora.

	1.2.2 Ensambladores 1.2.3 Compiladores 1.2.4 Montadores de enlace 1.2.5 Cargadores 1.2.6 Interpretes 1.2.7 Descompiladores 1.2.8 Desensambladores 1.2.9 Depuradores 1.2.10 Analizadores de rendimiento 1.2.11 Optimizadores de código 1.2.12 Compresores 1.2.13 Preprocesadores 1.2.14 Formateadores 1.3. Fases de un compilador y sus fundamentos teóricos. Aplicación a la construcción de compiladores sencillos de lenguajes simples. 1.3.1 Análisis del programa fuente 1.3.2 Las fases de un compilador 1.3.3 El agrupamiento de las fases 1.4 Tipos de gramáticas 1.5 Forma de backus nour 1.6 Jerarquías de chomsky	
PROFESIONALES: **CIENCIAS FUNDAMENTALES DE LA INGENIERÍA** Utiliza las matemáticas como herramientas para solución de problemas en ingeniería.	Análisis léxico. 2.1 Función del análisis lexicográfico 2.2 Lexemas, expresiones regulares y tokens 2.3 Manejo de buffers de entrada 2.4 Especificación de los componentes léxicos 2.5 Reconocimiento de los componentes léxicos 2.6 Autómatas finitos 2.7 Autómatas finitos no determinísticos 2.8 Autómatas finitos determinísticos 2.9 Paso de una expresión regular a un afn 2.10 Diseño de un generador de analizadores léxicos	Desarrolla analizadores léxicos para el tratamiento de texto que identifiquen elementos básicos tipo tokens.
ESPECÍFICAS: **FUNDAMENTOS DE CIENCIAS DE LA COMPUTACIÓN** Aplica las bases de los lenguajes de programación para generar aplicaciones óptimas	Análisis sintáctico. 3.1 Función del análisis sintáctico 3.2 Gramáticas libres del contexto 3.3 Escritura de una gramática 3.4 Análisis sintáctico descendente 3.5 Análisis sintáctico ascendente 3.6 Análisis sintáctico por procedencia de operadores 3.7 Analizadores sintácticos izquierda-derecha (lr) 3.8 Uso de gramáticas ambiguas	Desarrolla analizadores sintácticos para el tratamiento de texto que comprueban procedencia de operadores como preámbulo a un lenguaje de programación.
	Herramienta para generar compiladores. 4.1 Herramientas tradicionales	Analiza las ventajas de las herramientas para generar

	4.1.1 Ventajas 4.1.2 Desventajas 4.1.3 Ejemplos 4.2 Herramientas de nueva generación 4.2.1 Ventajas 4.2.2 Desventajas 4.2.3 Ejemplos 4.3 Otras herramientas 4.4 Kits para construcción de compiladores	compiladores de lenguajes de programación.
	Traducción dirigida por sintaxis. 5.1 Definiciones dirigidas por la sintaxis 5.2 Análisis de las definiciones dirigidas por sintaxis 5.3 Evaluación ascendente de las definiciones s-atribuidas 5.4 Definiciones l-atribuidas 5.5 traducción durante el análisis descendente 5.6 Evaluación ascendente de atributos heredados 5.7 Asignación de memoria.	Define el concepto de traducción dirigida por sintaxis dentro de los compiladores.
	Análisis semántico. 6.1 Función del análisis semántico 6.2 Reglas semánticas 6.3 Compatibilidad de tipos 6.4 Sistemas de tipos 6.5 Comprobación estática y dinámica de tipos 6.6 Comprobación de tipos en expresiones, sentencias y funciones 6.5 Coerciones, sobrecarga de funciones y operadores, funciones polimórficas	Desarrolla analizadores semánticos que comprueban coherencia de expresiones en un texto de lenguaje de programación.
	Manejo de errores. 7.1 Funcionamiento del manejo de errores 7.2 Técnicas básicas de detección de errores 7.3 Agregando símbolos de error 7.4 Agregando la tabla de ERRORES	Describe el manejo de errores dentro del proceso de compilación de un programa de cómputo.
	Generación de código intermedio 8.1 LENGUAJES INTERMEDIOS 8.2 DECLARACIONES 8.3 PROPOSICIONES DE ASIGNACIÓN 8.4 EXPRESIONES BOOLEANAS 8.5 LLAMADAS A PROCEDIMIENTOS	Explica el proceso de generación de código intermedio durante el proceso de desarrollo de un compilador

OBJETO DE ESTUDIO	**METODOLOGIA** (Estrategias, secuencias, recursos didácticos)	**EVIDENCIAS DE APRENDIZAJE.**
I. INTRODUCCIÓN. II. ANÁLISIS LÉXICO. III. ANÁLISIS SINTÁCTICO.	Aprendizaje interactivo (exposición del profesor) Grupo de discusión.	• Pruebas escritas • Solución de ejercicios (aplicación de conocimientos

IV. HERRAMIENTAS PARA GENERAR COMPILADORES. V. TRADUCCIÓN DIRIGIDA POR SINTAXIS. VI. ANÁLISIS SEMÁNTICO. VII. MANEJO DE ERRORES. VIII. GENERACIÓN DE CÓDIGO INTERMEDIO.	Auto aprendizaje (búsqueda y análisis de información) Inductivo • Observación • Comparación • Experimentación Deductivo • Aplicación • Comprobación • Demostración Sintético • Recapitulación • Definición • Resumen • Esquemas • Modelos matemáticos • Conclusión. **Material de Apoyo didáctico: Recursos** • Literatura citada en el programa del curso • Manual de Instrucción para prácticas de laboratorio • Materiales gráficos: artículos y libros, entre otros • Cañón • Pizarrón, pintarrones	• Exposición • Trabajo de laboratorio de cómputo. • Respeto y participación al trabajo dentro del salón de clase. • Interés por la asignatura .

FUENTES DE INFORMACIÓN (Bibliografía, Direcciones electrónicas)	**EVALUACIÓN DE LOS APRENDIZAJES** (Criterios e instrumentos)
1. Mak, R. (2009**).** *Writing Compileres and Interpreters*. (3ª.ed.). Wiley. 2. Cooper, K. y Torczon, L. (2009*). Engineering a Compiler.* (2ª.ed.). Morgan Kaufmann. 3. Aho, A.V., Lam, M.S., Sethi, R. y Ullman J.D. (2006). *"Compilers: Principles, Techniques, and Tools"*. (2ª.ed.). Addison-Wesley.	**INSTRUMENTOS:** • Prueba escrita • Solución de ejercicios (aplicación de conocimientos • Prácticas de laboratorio de cómputo (ejercicios, problemas, desarrollo de software). • Lista de cotejo (Respeto y participación al trabajo dentro del salón de clase, interés por la asignatura **CRITERIOS DE DESEMPEÑO:** • La solución de ejercicios se realiza en clase en forma individual o por equipos según amerite. • **Exposición:** presentadas en orden lógico: 1. Introducción resaltando el objetivo a alcanzar

	2. Desarrollo temático, responder preguntas y aclarar dudas 3. Concluir. • **Los trabajos extracurriculares** Toda actividad complementaria al curso se podrá llevar a cabo en forma individual o por equipo según amerite el tema. Estos se reciben únicamente en tiempo y forma previamente establecidos. • **Exámenes escritos**: Se realizan 3 exámenes escritos durante el semestre y las fechas se establecen por la secretaría académica • **Prácticas de laboratorio**: Las prácticas de laboratorio por unidad consistirán en el desarrollo de software del tema visto. Esta práctica se especificará mediante una lista de puntos a cumplir que contemplen: • Objetivo de la práctica. • Tarea especifica a desarrollar. • Resultados esperados. Se toma en cuenta para integrar **calificaciones parciales:** Prueba escrita 50% Solución de ejercicios (incluyendo práctica de laboratorio) 50% Fecha de exámenes parciales: 1º. Parcial: 2º. Parcial: 3er Parcial: **La acreditación del curso:** • Promedio de Calificaciones parciales: 70% • Cuestionarios, resúmenes, participación en exposiciones, discusión individual, por equipo y grupal: 30% **LAS ACTIVIDADES NO REALIZADAS EN TIEMPO Y FORMA SE CALIFICAN CON CERO**. **Nota:** para acreditar el curso se deberá tener calificación aprobatoria tanto en la teoría como en las prácticas.

Cronograma del Avance Programático

S e m a n a s

Unidades de aprendizaje	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
I.INTRODUCCIÓN.																
II.ANALISIS LÉXICO.																
III. ANÁLISIS SINTACTICO.																
IV HERRAMIENTAS PARA GENERAR COMPILADORES.																
V TRADUCCIÓN DIRIGIDA POR LA SINTAXIS.																
VI. ANÁLISIS SEMÁNTICO																
VII. MANEJO DE ERRORES																
VIII. GENERACIÓN DE CÓDIGO INTERMEDIO.																