UNIVERSIDAD AUTÓNOMA DE SINALOA

FACULTAD DE INGENIERÍA MOCHIS

LICENCIATURA EN INGENIERÍA DE SOFTWARE

PROGRAMA DE ESTUDIO

1. DATOS DE IDENTIFICACIÓN			
UNIDAD DE APRENDIZAJE O MÓDULO:	SOFTWARE DE SISTEMAS		
Clave:			
Ubicación:	Semestre VI	Área: Profesionalizante	
Horas y créditos:	Teóricas: 50	Prácticas: 30	Estudio Independiente: 80
	Total de horas: 160	Créditos: 10	
Competencia(s) del perfil de egreso al que aporta:	CG7. Cultiva el compañerismo, el trabajo en equipo y la coordinación de esfuerzos bajo la aspiración de mejorar las tareas académicas, los entornos laborales y la convivencia social en beneficio para la consecución de metas que impactan en las formas de entablar y mantener relaciones humanas positivas. CG10. Asume con responsabilidad y ética el manejo de las Tecnologías del Aprendizaje y el Conocimiento y es capaz de reconducir las Tecnologías de la Información y Comunicación para la adquisición y actualización del conocimiento de manera permanente para su vida y su profesión. CE2. Desarrolla habilidades de abstracción y la expresión de formalismos, además de proporcionar conocimientos específicos fundamentales para la informática y la computación. CE3. Aplica los conceptos básicos de conjuntos, lógica matemática, relaciones, grafos y árboles para resolver problemas afines al área computacional e identifica las estructuras básicas de las matemáticas discretas, cómo aplicarlas en el manejo y tratamiento de las TICS. CE4. Formula y resuelve ecuaciones que permiten asociarlas a fenómenos fundamentales relacionados con las ciencias computacionales y la ingeniería de software como conceptos y operaciones con matrices y vectores que se relacionan con el procesamiento de imágenes y de información		
Unidades de aprendizaje relacionadas:	• Programación • Estructuras de Datos • Teoría de la Computación		
Responsable(s) de elaborar el programa:	Edgar Omar Pérez Contreras	Fecha: 04/noviembre/2025	
Responsable(s) de actualizar el programa:	Edgar Omar Pérez Contreras	Fecha: 4/noviembre/2025	
2. PROPÓSITO			

Que el alumno conozca los fundamentos teóricos del funcionamiento de un compilador mediante el desarrollo de un pequeño lenguaje de programación y su compilador con el objetivo de	
3. SABERES	
Teóricos:	• Identifica los tipos de software de sistemas existentes. • Identifica las fases del compilador. • Conoce lo que es el software de sistemas, la importancia de sus estudios y la diferencia con software de aplicación. • Distingue y diferencia las fases de un compilador. • Diferencia entre u compilador y un intérprete. • Comprende las ventajas de usar expresiones regulares. • Entiende y utiliza la simbología de las expresiones regulares. • Comprende que es un autómata. • Identifica las ventajas de la memoria dinámica y su aplicación. • Comprende y aplica la técnica recursiva. • Maneja eficientemente las estructuras de datos (árboles). • Conocer y utiliza teórica y prácticamente los conceptos de la gramática libre de contexto. • Comprende y aplica algoritmos comunes para la creación de la fase semántica
Prácticos:	• Conoce y aplica técnicas (expresiones regulares, autómatas) para desarrollar un compilador en su fase léxica. • Cono y aplica técnicas para desarrollar un autómata. • Representa un patrón por medio de expresiones regulares y autómatas. • Utiliza la memoria dinámica para algoritmos. • Utiliza técnicas recursivas para resolver problemas. • Maneja árboles eficientemente. • Emplea una gramática libre de contexto para la fase sintáctica. • Utiliza algoritmos para la fase semántica. • Diseña e implementa un compilador en su fase léxica, sintáctica y semántica. • Adquiere habilidades avanzadas de programación.
Actitudinales:	• Trabaja en equipo. • Soluciona problemas. • Planifica, organiza y controla el proyecto. • Demuestra compromiso ético. • Respeta la puntualidad.
4. CONTENIDOS	
UNIDAD I. Fundamentos de Compiladores	

Objetivo: Comprender el papel del compilador, sus fases principales y los programas relacionados con su funcionamiento.

1.1. Conceptos generales de compiladores
- 1.1.1. Definición y función de un compilador
- 1.1.2. Relación entre compilador, intérprete y ensamblador

1.2. Fases del proceso de compilación
- 1.2.1. Análisis léxico
- 1.2.2. Análisis sintáctico
- 1.2.3. Análisis semántico
- 1.2.4. Generación de código intermedio

1.3. Administración y control del proceso
- 1.3.1. Tabla de símbolos
- 1.3.2. Manejo y reporte de errores

1.4. Programas relacionados
- 1.4.1. Preprocesadores
- 1.4.2. Ensambladores
- 1.4.3. Cargadores y editores de enlace

Práctica: Elaborar un diagrama que muestre las fases de un compilador y analizar el flujo de un compilador real o educativo (por ejemplo, TinyC, PLY o MiniLang).

UNIDAD II. Análisis Léxico

Objetivo: Diseñar e implementar el analizador léxico de un compilador utilizando expresiones regulares y autómatas finitos.

2.1. Conceptos del análisis léxico
- 2.1.1. Función y objetivo del analizador léxico
- 2.1.2. Identificación de tokens, lexemas y patrones

2.2. Expresiones regulares
- 2.2.1. Definición formal
- 2.2.2. Expresiones regulares aplicadas a lenguajes de programación

2.3. Autómatas finitos
- 2.3.1. Autómatas finitos determinísticos (DFA)
- 2.3.2. Autómatas finitos no determinísticos (NFA)
- 2.3.3. Implementación básica de autómatas en código

2.4. Construcción del analizador léxico
- 2.4.1. Diseño manual o mediante herramientas (Lex, Flex, JLex)
- 2.4.2. Implementación y prueba de reconocimiento de tokens

Práctica: Implementar un analizador léxico que reconozca identificadores, números, operadores y palabras reservadas de un mini lenguaje.

UNIDAD III. Análisis Sintáctico

Objetivo: Comprender y aplicar los principios del análisis sintáctico mediante gramáticas libres de contexto y métodos descendentes.

3.1. Fundamentos del análisis sintáctico
- 3.1.1. Proceso y función del análisis sintáctico

3.1.2. Comparación con el análisis léxico
3.2. Gramáticas libres de contexto
3.2.1. Definición y elementos básicos
3.2.2. Derivaciones y árboles de análisis
3.2.3. Árboles sintácticos abstractos (AST)
3.3. Métodos de análisis sintáctico
3.3.1. Análisis sintáctico descendente recursivo
3.3.2. Análisis predictivo simple (LL(1))
3.3.3. Manejo de errores sintácticos
3.4. Implementación práctica
3.4.1. Especificación y codificación de un analizador sintáctico
3.4.2. Integración con el analizador léxico

Práctica: Construir un analizador sintáctico descendente que reconozca expresiones y estructuras de control básicas (asignaciones, if, while).

UNIDAD IV. Análisis Semántico
Objetivo: Aplicar técnicas básicas de análisis semántico, manejo de la tabla de símbolos y verificación de tipos.

4.1. Conceptos del análisis semántico
4.1.1. Propósito del análisis semántico
4.1.2. Momento de ejecución en el proceso del compilador
4.2. Gramáticas con atributos
4.2.1. Tipos de atributos: sintetizados y heredados
4.2.2. Evaluación de atributos y dependencias
4.3. Tabla de símbolos
4.3.1. Estructura y organización
4.3.2. Declaraciones, ámbitos y alcance
4.4. Verificación de tipos
4.4.1. Equivalencia e inferencia de tipos
4.4.2. Validación de operaciones y declaraciones

Práctica: Extender el compilador para incluir verificación de variables declaradas, tipos de datos y control semántico básico.

UNIDAD V. Generación de Código Intermedio
Objetivo: Traducir las estructuras sintácticas a una representación intermedia que permita generar código ejecutable simple.

5.1. Fundamentos de la generación de código
5.1.1. Objetivos y etapas de traducción
5.1.2. Representaciones intermedias
5.2. Código intermedio
5.2.1. Código de tres direcciones
5.2.2. Estructuras de datos para su representación
5.3. Generación de código a partir del AST
5.3.1. Expresiones aritméticas y lógicas
5.3.2. Sentencias de control (if, while)

5.4. Traducción a código ejecutable o interpretado
- 5.4.1. Ejecución simbólica o simulada
- 5.4.2. Integración con las fases anteriores del compilador

Práctica: Implementar un generador de código intermedio que traduzca expresiones y sentencias de control a instrucciones de tres direcciones.

5. ACTIVIDADES PARA DESARROLLAR LAS COMPETENCIAS

Actividades del docente:

- Explicación inicial de la unidad en clases presenciales
- Brinda asesorías extra clase
- Habilito foro temático y modera la discusión entre los estudiantes para la construcción del conocimiento
- Organiza las actividades del estudiante para ser evaluadas
- Revisa y evalúa las actividades de aprendizajes
- Actualiza el libro de calificaciones del entorno virtual
- Brinda una introducción a los Fundamentos del Lenguaje C
- Brinda una introducción a las Estructuras de Control del Lenguaje C
- Brinda una introducción a los Arreglos en el Lenguaje C

Actividades del estudiante:

- Asiste a clases presenciales.
- Revisa, analiza y estudia los materiales proporcionados por el docente
- Participa activamente en el foro temático
- Expone sus dudas y ayuda a resolver dudas de otros compañeros en el foro de dudas
- Colabora en la edición de la Wiki agregando entradas nuevas o mejorando las ya existentes

6. EVALUACIÓN DE LAS COMPETENCIAS

6.1. Criterios de desempeño	6.2 Portafolio de evidencias
Todas las unidades: ❖ Asistencia del 80% a clases ❖ Participación en Foros y Wikis en Plataforma Virtual ❖ Participación en clases ❖ Trabajo en equipo ❖ Exposiciones ❖ Desarrollo de tareas ❖ Desarrollo de prácticas ❖ Desarrollo de los productos finales de cada unidad	**Unidad 1:** ❖ Infografía sobre compiladores incluyendo: análisis de programa fuente y fases del compilador así como programas relacionados. ❖ Instalación y configuración de CodeBlocks y un programa “Hola Mundo” en Lenguaje C **Unidad 2:** ❖ Diagrama de Flujo sobre el proceso de análisis léxico ❖ Mapa conceptual sobre el Analizador Léxico ❖ Examen Unidad 2 ❖ Especificación de su Lenguaje de Programación ❖ Compilador-Fase 1: Analizador Léxico ❖ Resumen de esta unidad **Unidad 3:** ❖ Compilador-Fase 2: Especificación del Analizador Sintáctico ❖ Examen Unidad 3

	Unidad 4: ❖ Compilador-Fase 2: Analizador Sintáctico ❖ Resumen de la Unidad ❖ Examen Unidad 4 **Unidad 5:** ❖ Compilador-Fase 3: Analizador Semántico ❖ Exposición en Equipos de 2 ❖ Examen Unidad 5 ❖ Compilador-Fase 5: Generador de Código (puntos extra si cumple con esta actividad)
6.3. Calificación y acreditación:	
Parcial: ❖ Parcial 1: Unidad 1 y Unidad 2 • 20% ❖ Parcial 2: Unidad 3 • 30% ❖ Parcial 3: Unidad 4 y Unidad 5 • 30%	**Final:** ❖ 3 parciales: 80% ❖ Presentación Compilador: 20%

7. RECURSOS DIDÁCTICOS

- ❖ Google Classroom
- ❖ Google Drive,
- ❖ Correo electrónico,
- ❖ WhatsApp,
- ❖ Video Proyector,
- ❖ Internet,
- ❖ Artículos científicos y de difusión,
- ❖ Tutoriales,
- ❖ Materiales didácticos,
- ❖ Recursos tecnológicos o auditivos,
- ❖ Páginas web oficiales

8. FUENTES DE INFORMACIÓN

Bibliografía básica

Autor(es)	Título	Editorial	Año	URL o biblioteca digital donde está disponible
Alfred V. Aho, Monica S. Lam, Ravi	Compiladores. Principios, Técnicas y		2006	

Sethi y también Jeffrey D. Ullman	Herramientas (Dragon Book)			
Kenneth C. Louden	Uned Construccion de Compiladores. Principios y Practica	Thomson	2004	
J. Clenn Brookshear	Teoría de la Computación. Lenguajes formales, autómatas y complejidad	Addison Wesley Iberoamericana		

Bibliografía complementaria

Autor(es)	Título	Editorial	Año	URL o biblioteca digital donde está disponible
Allen I. Holub	Compiler Design in C	Prentice All	1990	
Torben Ægidius Mogensen	Basics of Compiler Design	Department of Computer Science University of Copenhagen	2010	

9. PERFIL DEL DOCENTE

Académicos:
Ingeniero en Software, Licenciado en Sistemas Computacionales, Licenciado en Informática o carrera afín, preferentemente con maestría o doctorado en una carrera similar. Con experiencia docente en la enseñanza de asignaturas relacionadas con el Software de Sistemas.

Experiencia Laboral:
Contar con experiencia en el Desarrollo de Software de Sistemas, experiencia en la industria de la tecnología de la información (deseable), experiencia en Gestión de Proyectos de Software (deseable)

Conocimientos:
Arquitectura de Software; Diseño de Software de Sistemas; Mantenimiento de Software; Lenguajes de Programación (C, C++, Java, Python, etc); Sistemas Operativos; Bases de Datos, Redes de Computadoras.

Habilidades:
Comunicación efectiva; Trabajo en equipo; Resolución de problemas; Liderazgo.