



**Programa de Estudios por Competencias
 COMPILADORES**

I. IDENTIFICACIÓN DEL CURSO

ORGANISMO ACADÉMICO: FACULTAD DE INGENIERÍA						
Programa Educativo: COMPILADORES				Área de docencia: SOFTWARE DE BASE		
Aprobación por los H.H. Consejos Académico y de Gobierno		Fecha:		Programa elaborado por: Dr. José Raymundo Marcial Romero M. en Ing. Felipe Camacho Mejía		Programa actualizado por: Facultad de Ingeniería: Dr. José Raymundo Marcial Romero Dra. Vianney Muñoz Jimenez
				Fecha de elaboración : Noviembre 2009		Fecha de actualización: Noviembre 2013
Clave	Horas de teoría	Horas de práctica	Total de horas	Créditos	Tipo de Unidad de Aprendizaje	Núcleo de formación
L41048	4	1	5	9	CURSO	SUSTANTIVO
Prerrequisitos: Autómatas y Lenguajes Formales, Programación Avanzada		Unidad de Aprendizaje Antecedente Ninguna			Unidad de Aprendizaje Consecuente: Ninguna	
Programas educativos o espacios académicos en los que se imparte: Licenciatura en Ingeniería en Computación (Facultad. de Ingeniería, Centros Universitarios: Atlacomulco, Ecatepec, Texcoco, Valle de Chalco, Valle						



Universidad Autónoma del Estado de México

Secretaría de Docencia

Dirección de Estudios Profesionales

Coordinación de Desarrollo Curricular

de México, Valle de Teotihuacán, Zumpango)

II. PRESENTACIÓN

El uso de lenguajes de programación es una tarea cotidiana de cualquier Ingeniero en Computación. Esta actividad debe ser complementada con la enseñanza de los diferentes paradigmas de lenguajes de programación para tener un amplio criterio del lenguaje a elegir cuando se desea realizar una aplicación. De esta forma, el Ingeniero en Computación debe conocer que, de entre todos los lenguajes de programación que utiliza, hay ciertos lenguajes que pertenecen a la categoría de compilados.

El propósito de esta unidad de aprendizaje es presentar las fases que tienen lugar en el desarrollo de un compilador, esto es con la finalidad de que el alumno pueda identificar y ser capaz de realizar algunas de estas fases para el desarrollo de un compilador. Así mismo, en este curso, se presentan las herramientas que se utilizan para la generación automática de un compilador.



III. LINEAMIENTOS DE LA UNIDAD DE APRENDIZAJE

DOCENTE	DISCENTE
<ul style="list-style-type: none">• Dar a conocer a los alumnos el temario al inicio del semestre.• Cumplir en tiempo y contenido la Unidad de aprendizaje.• Asistir puntualmente a las clases o justificar la ausencia por adelantado.• Asesorar a los alumnos y resolver sus dudas.• Establecer tolerancia para el inicio de clase.• Proponer y respetar formas de evaluación.• Evaluar y calificar a los alumnos.• Preparar el material didáctico para las clases y prácticas.• Respetar número de horas teóricas y prácticas.	<ul style="list-style-type: none">• Contar con el 80% de asistencia para presentar examen ordinario.• Contar con el 60% de asistencia para presentar examen extraordinario.• Contar con el 30% de asistencia para presentar examen a título de suficiencia.• Entregar en tiempo y forma las tareas y proyectos requeridos por el docente• Tener sentido de integración y participación dentro del salón de clases.

IV. PROPÓSITO DE LA UNIDAD DE APRENDIZAJE

El alumno:

Conocerá las teorías, técnicas y metodologías para el diseño y construcción de compiladores con el objetivo de construir de un compilador básico.



Elaborará los programas pertinentes para el desarrollo de cada una de las fases de un compilador, esto le permitirá al final del curso unir las piezas elaborados para construir un compilador básico.

V. COMPETENCIAS GENÉRICAS

El alumno desarrollará compiladores básicos mediante el análisis de escenarios donde pueden o no existan herramientas específicas para un propósito particular.

El alumno utilizará eficazmente las herramientas de programación para el desarrollo de las diferentes fases de un compilador.

El alumno realizará investigación de tecnologías de punto en cuanto a compiladores respecta. Esto le ayudará a conocer el estado del arte de un área tan dinámica como lo es el desarrollo de dichas herramientas.

VI. ÁMBITOS DE DESEMPEÑO PROFESIONAL

Empresas de desarrollo de software.

Docencia a cualquier nivel de aprendizaje escolarizado.

Investigación de nuevas tecnologías de compiladores.



Universidad Autónoma del Estado de México

Secretaría de Docencia

Dirección de Estudios Profesionales

Coordinación de Desarrollo Curricular

VII. ESCENARIOS DE APRENDIZAJE

Aula, laboratorio de computadores (uso de las herramientas flex, bison, javacup o javacc).

VIII. NATURALEZA DE LA COMPETENCIA

(Inicial, entrenamiento, complejidad creciente, ámbito diferenciado)

Entrenamiento y complejidad creciente.



IX. ESTRUCTURA DE LA UNIDAD DE APRENDIZAJE

1. Conocer las fases de la metodología de compilación para la creación de lenguajes.
2. Utilizar las técnicas de desarrollo de autómatas finitos y expresiones regulares para la construcción de analizadores léxicos.
3. Emplear las técnicas de desarrollo de las gramáticas libres de contexto para la construcción de analizadores sintácticos.
4. Formular el análisis semántico de un pequeño compilador.
5. Construir la representación intermedia del compilador básico para la generación de código del programa ejecutable
6. Conocer técnicas de optimización de código para un compilador

X. DESARROLLO DE LA UNIDAD DE APRENDIZAJE

UNIDAD DE COMPETENCIA I	ELEMENTOS DE COMPETENCIA		
	Conocimientos	Habilidades	Actitudes/ Valores
Conocer las fases de la metodología de compilación para la creación de lenguajes.	<ol style="list-style-type: none">1. Programas relacionados con un compilador.2. El proceso de compilación.3. Estructuras de datos principales de un compilador.	<ul style="list-style-type: none">• Analizar las diferentes fases de un compilador.• Entender las diferentes herramientas de compilación.	<ul style="list-style-type: none">- Cumplir con las actividades asignadas.- Tolerancia y participación activa- Actitud propositiva..



	4. Herramientas para la construcción de un compilador.		
ESTRATEGIAS DIDÁCTICAS: Uso de diagramas, resúmenes, cuestionarios, ejercicios, presentaciones acompañadas de apuntes preparados por el profesor, trabajos en equipo.	RECURSOS REQUERIDOS Libros de texto, apuntes del docente, pizarrón, proyector (cañón o transparencias), computadora.	TIEMPO DESTINADO 4 hrs.	
CRITERIOS DE DESEMPEÑO I	EVIDENCIAS		
	DESEMPEÑO	PRODUCTOS	
Utilizando un esquema de reactivos, el alumno debe diferenciar las diferentes fases de un compilador (50%)	Manejo de conceptos básicos	Documento	
Resolver una serie de ejercicios prácticos propuestos en clase (50%)	Aplicación de conceptos	Listado de ejercicios resueltos	



UNIDAD DE COMPETENCIA II	ELEMENTOS DE COMPETENCIA		
	Conocimientos	Habilidades	Actitudes/ Valores
Utilizar las técnicas de desarrollo de autómatas finitos y expresiones regulares para la construcción de analizadores léxicos.	5. Proceso de Análisis léxico. 6. Funciones del Analizador léxico. 7. Expresiones Regulares- 8. Autómatas finitos (deterministas y no deterministas). 9. Diseño de un generador de análisis léxico. 10. Uso de una herramienta para generar automáticamente un analizador léxico.	- Utilizar las herramientas existentes para la construcción de analizadores léxicos. Diseñar un analizador léxico tomando como base las especificaciones del lenguaje dado.	- Cumplir las actividades asignadas. - Desarrollar la capacidad analítica ante nuevos problemas - Participación activa.
ESTRATEGIAS DIDÁCTICAS: Uso de máquinas de estado finito, resúmenes, cuestionarios, ejercicios, presentaciones acompañadas de apuntes preparados por el profesor, trabajos en equipo.	RECURSOS REQUERIDOS Libros de texto, apuntes del docente, pizarrón, proyector (cañón o transparencias), computadora, herramienta para el desarrollo de analizadores léxicos (por ejemplo Lex)	TIEMPO DESTINADO 14 hrs	



CRITERIOS DE DESEMPEÑO II	EVIDENCIAS	
	DESEMPEÑO	PRODUCTOS
	Construcción de Maquinas de estado y expresiones regulares a partir de especificaciones reales.	Diseño de autómatas finitos
El profesor elegirá un lenguaje de programación pequeño para que el alumno construya la menos las primeras 4 fases su compilador. Se recomienda MiniJava que se puede obtener en la tercer referencia bibliográfica. En esta parte del curso, el alumno implementará el analizador léxico para el lenguaje elegido. La herramienta sugerida para MiniJava es JavaCC pero se deja a elección del alumno y profesor.	Manejo de herramientas para generación de analizadores léxicos	Programa



UNIDAD DE COMPETENCIA III	ELEMENTOS DE COMPETENCIA		
	Conocimientos	Habilidades	Actitudes/ Valores
Emplear las técnicas de desarrollo de las gramáticas libres de contexto para la construcción de analizadores sintácticos.	11. Proceso de análisis sintáctico. 12. Gramáticas independientes del contexto. 13. Análisis sintáctico descendente. 14. Análisis sintáctico ascendente. 15. Gramáticas Ambiguas. 16. Uso de una herramienta para generar automáticamente un analizador sintáctico.	<ul style="list-style-type: none"> • Aplicar diferentes técnicas de análisis sintáctico. • Diseñar un analizador sintáctico tomando como base las especificaciones del lenguaje dado. 	- Cumplir las actividades asignadas. - Desarrollar la capacidad analítica ante nuevos problemas - Participación activa.
ESTRATEGIAS DIDÁCTICAS: Uso de árboles, resúmenes, cuestionarios, ejercicios, presentaciones acompañadas de apuntes preparados por el profesor, trabajos en equipo.	RECURSOS REQUERIDOS Libros de texto, apuntes del docente, pizarrón, proyector (cañón o transparencias), computadora, herramienta para el desarrollo de analizadores léxicos (por ejemplo Yacc)	TIEMPO DESTINADO 16 hrs	



CRITERIOS DE DESEMPEÑO III	EVIDENCIAS	
	DESEMPEÑO	PRODUCTOS
	A partir de especificaciones reales, el alumno construirá gramáticas y árboles de análisis gramatical.	Diseño de gramáticas libres de contexto
El alumno implementará el analizador sintáctico para el lenguaje elegido. La herramienta sugerida para Mnijava es JavaCC pero se deja a elección del alumno y profesor.	Manejo de herramientas para generación de analizadores sintácticos	Programa



UNIDAD DE COMPETENCIA IV	ELEMENTOS DE COMPETENCIA		
	Conocimientos	Habilidades	Actitudes/ Valores
Formular el análisis semántico de un compilador pequeño	17. Análisis Semántico 18. La tabla de símbolos. 19. Árbol Sintáctico Abstracto 20. Tipos de datos y verificación de tipos 21. Creación de un analizador semántico para un lenguaje básico.	<ul style="list-style-type: none"> • Diseñar una tabla de símbolos. • Escribir un verificador de tipos para un lenguaje dado. • Anotar árboles para la creación de analizadores semánticos. 	<ul style="list-style-type: none"> • Tolerancia • Perseverancia • Participación activa • Cumplir las actividades asignadas. • Desarrollar la capacidad analítica ante nuevos problemas • Participación activa.
ESTRATEGIAS DIDÁCTICAS: Uso de algoritmos para cálculos de atributos, resúmenes, cuestionarios, ejercicios, presentaciones acompañadas de apuntes preparados por el profesor, trabajos en equipo.	RECURSOS REQUERIDOS Libros de texto, apuntes del docente, pizarrón, proyector (cañón o transparencias), computadora, lenguaje de programación (C o Java).	TIEMPO DESTINADO 17 hrs	
CRITERIOS DE DESEMPEÑO IV	EVIDENCIAS		
	DESEMPEÑO	PRODUCTOS	
A partir de especificaciones reales, el	Diseño de gramáticas libres de contexto	Documento de ejercicios resueltos	



alumno construirá gramáticas y árboles de análisis gramatical.		
El alumno implementará el analizador semántico para el lenguaje elegido. La herramienta sugerida para Minijava es JavaCC pero se deja a elección del alumno y profesor. El analizador semántico debe contemplar: la tabla de símbolos, el verificador de tipos, las acciones semánticas de las producciones.	Manejo de herramientas para generación de analizadores sintácticos	Programa



UNIDAD DE COMPETENCIA V	ELEMENTOS DE COMPETENCIA		
	Conocimientos	Habilidades	Actitudes/ Valores
<p>Construir la representación intermedia del compilador básico para la generación de código de un programa ejecutable.</p>	<p>22. Técnicas básicas de generación de código. 23. Código Intermedio. 24. Generación de código.</p>	<ul style="list-style-type: none"> • Conocer y aplicar los tipos de código intermedio más comunes. • Diseñar soluciones para la generación de código intermedio de un compilador • Convertir de código intermedio a código de máquina 	<ul style="list-style-type: none"> • Tolerancia • Perseverancia • Participación activa • Cumplir las actividades asignadas. • Desarrollar la capacidad analítica ante nuevos problemas • Participación activa..
<p>ESTRATEGIAS DIDÁCTICAS: Uso de lenguajes de programación compilados, resúmenes, cuestionarios, ejercicios, presentaciones acompañadas de apuntes preparados por el profesor, trabajos en equipo.</p>	<p>RECURSOS REQUERIDOS Libros de texto, apuntes del docente, pizarrón, proyector (cañón o transparencias), computadora, lenguaje de programación (C o Java).</p>	<p>TIEMPO DESTINADO 16 hrs</p>	
CRITERIOS DE DESEMPEÑO V	EVIDENCIAS		
	DESEMPEÑO	PRODUCTOS	
Utilizando una herramienta para generar	Construcción de un generador de código	Programa que ejemplifique las diferentes	



*Secretaría de Docencia
Coordinación General de Estudios Superiores
Programa Institucional de Innovación Curricular*

código, el alumno debe escribir el programa para convertir la especificación de su lenguaje en código	para un lenguaje pequeño.	fases de generación de código de un compilador.
Resolver una serie de ejercicios prácticos propuestos en clase.	Aplicación de conceptos	Listado de ejercicios resueltos



UNIDAD DE COMPETENCIA VI	ELEMENTOS DE COMPETENCIA		
	Conocimientos	Habilidades	Actitudes/ Valores
Conocer técnicas de optimización de código para un compilador	25. Definiciones básicas 26. Tipos de optimización 27. Optimización básica de programas.	<ul style="list-style-type: none"> Diferenciar entre técnicas básicas y avanzadas de optimización de código. Identificar la técnica de optimización aplicable a un lenguaje de programación. 	<ul style="list-style-type: none"> Tolerancia Perseverancia Participación activa Cumplir las actividades asignadas. Desarrollar la capacidad analítica ante nuevos problemas Participación activa..
ESTRATEGIAS DIDÁCTICAS: Uso de un lenguajes de programación orientado a objetos (por ejemplo Java), resúmenes, cuestionarios, ejercicios, presentaciones acompañadas de apuntes preparados por el profesor, trabajos en equipo.	RECURSOS REQUERIDOS Libros de texto, apuntes del docente, pizarrón, proyector (cañón o transparencias), computadora, lenguaje de programación (C o Java).	TIEMPO DESTINADO 13 hrs	
CRITERIOS DE DESEMPEÑO VI	EVIDENCIAS		
	DESEMPEÑO	PRODUCTOS	
Resolver una serie de ejercicios prácticos propuestos en clase.	Aplicación de conceptos	Listado de ejercicios resueltos	



El alumno realizará una investigación sobre las técnicas de optimización desarrolladas recientemente para la creación de un compilador. Su investigación debe considerar por lo menos 3 técnicas para poder contrastarlas.	Conocimiento sobre técnicas de optimización.	Ensayo.
--	--	---------

XI. EVALUACIÓN Y ACREDITACIÓN

Evaluación:

Calificaciones parciales (al menos 2)	50%
Examen Final	30 %
Proyecto Final	20 %

Calificación parcial:

Examen parcial escrito	50 %
Investigaciones y tareas	25 %
Proyecto parcial	25 %

Acreditación:

Cumplir con el 80% de asistencia
Promedio final de 6.0



XII REFERENCIAS

1. Aho, A. V., Sethi R., Ullman, D. J., *Compiladores Principios Técnicas y herramientas*, Addison Wesley, 2006, 2da. Edición.
2. Grune, D., Reeuwijk, K. V., Bal, H., Jacobs C. J. H., *Modern Compiler Design*, Springer, 2012, 2da. Edición.
3. Louden, K. C., *Construcción de Compiladores principios y práctica*, Internacional Thomson Editores, 1997.
4. Appel, A. W., Palsberg, J., *Modern Compiler implementation in Java*, Cambridge University Press, 2002, 2da. Edición.
5. Wilhelm, R., *Compiler Design*, Addison-Wesley, 1995.