Formato de programa de estudios para la formación y desarrollo de competencias profesionales

1.- DATOS DE LA ASIGNATURA

Nombre de la asignatura:	**Lenguajes y Autómatas I**
Carrera:	**Ingeniería en Sistemas Computacionales**
Clave de la asignatura:	SCD-1015
(Créditos) SATCA[1]:	**2-3-5**

2.- PRESENTACIÓN

Caracterización de la asignatura.

El desarrollo de sistemas basados en computadora y la búsqueda de soluciones para problemas de procesamiento de información son la base tecnológica de la carrera de Ingeniería en Sistemas.

Todo egresado de esta ingeniería debe poseer los conocimientos necesarios para resolver de manera óptima cualquier problema relacionado con procesamiento de información. El conocimiento de las características, fortalezas y debilidades de los lenguajes de programación y su entorno le permitirán proponer las mejores soluciones en problemas de índole profesional y dentro de las realidades de su entorno.

Como parte integral de la asignatura, se debe promover el desarrollo de las habilidades necesarias para que el estudiante implemente sistemas sujetándose en los estándares de desarrollo de software, esto con el fin de incentivar la productividad y competitividad de las empresas donde se desarrollen. Sin duda alguna, los problemas que se abordarán requerirán la colaboración entre grupos interdisciplinarios, por ello el trabajo en grupos es indispensable. Debe quedar claro que los proyectos que serán desarrollados son de diversas áreas y complejidades, y en ocasiones requieren la integración de equipos externos. Esta complejidad debe considerarse una oportunidad para experimentar con el diseño de interfaces hombre-máquina y máquina-máquina.

Como todos sabemos, un mismo problema puede ser resuelto computacionalmente de diversas formas. Una de las condiciones a priori de la asignatura, es el conocimiento de las arquitecturas de computadoras (microprocesadores) y de las restricciones de desempeño que deben considerarse para la ejecución de aplicaciones. Esto aportará los conocimientos que le permitirán al estudiante

[1] Sistema de asignación y transferencia de créditos académicos

desarrollar aplicaciones eficientes en el uso de recursos. De manera adicional, es posible que se integren dispositivos externos dentro de las soluciones. En este aspecto, el papel del profesor como guía es fundamental. Es importante diversificar la arquitectura de las soluciones planteadas. Si la inclusión de algún componente de hardware facilita la solución, se recomienda que sea incluido.

Esta área, por sus características conceptuales, se presta para la investigación de campo. Los estudiantes tendrán la posibilidad de buscar proyectos que les permitan aplicar los conocimientos adquiridos durante las sesiones del curso. El desarrollo de este proyecto es una oportunidad excelente para aplicar todos los conceptos, técnicas y herramientas orientadas al modelado. La formalidad con que se traten estos aspectos dotará al estudiante de nuevos conceptos, procedimientos y experiencia.

En esta asignatura se abordan todos los temas relacionados con teoría de lenguajes formales, algo que permite vislumbrar los procesos inherentes, y a veces, escondidos dentro de todo lenguaje. Las formas de representación formal, procesamiento e implementación de lenguajes de programación se atacan desde un punto de vista de implementación. Los proyectos relacionados y los ejercicios de investigación acercan a los estudiantes al campo de lenguajes formales, base de los procesos de comunicación. Por último se revisan algunos de los puntos eje de la investigación de frontera que aún contienen problemas abiertos, un incentivo para la incorporación de estudiantes a las áreas de investigación.

Las asignaturas directamente vínculadas son estructura de datos por las herramientas para el procesamiento de información que proporciona (árboles binarios, pilas, colas, tablas de Hash), todas aquellas que incluyan lenguajes de programación, porque son las herramientas para el desarrollo de cualquiera de las prácticas dentro de la asignatura y permitirán un enfoque práctico para todos los temas de la misma. La materia de arquitectura de computadoras dota al estudiante de los conocimientos sobre la estructura de registros, modos de direccionamiento, conjunto de operadores, y le da al estudiante una visión sobre cómo mejorar el desempeño de lenguajes.

Esta materia sirve de preámbulo para la asignatura de lenguajes y autómatas II, en la cual se completa el estudio formal de la teoría de lenguajes.

A su vez permitirá el desarrollo de las siguientes competencias específicas:
Evaluación de lenguajes de programación: evaluar un conjunto de lenguajes de programación con base en un problema a resolver y elegir el mejor de ellos para el problema en particular.

Análisis y síntesis para la solución de un problema: dado un problema, proponer el mejor lenguaje que se ajusta a las especificaciones del mismo. Si no hay lenguaje disponible, proponer las características del lenguaje ideal para el problema a resolver.

Intención didáctica.

Esta asignatura es de vital importancia para toda la carrera, como es una asignatura sobre lenguajes formales, el enfoque debe coincidir con la formalidad de los mismos. Cada tema debe ser acompañado de una serie de ejercicios y prácticas que permitan redondear los temas revisados en clase. Esta asignatura se presta para la participación activa de los estudiantes en la discusión de los temas y ejemplificación de casos. También permite que el estudiante se acerque al análisis de problemas del área industrial, como diseño, manufactura, tratamiento de lenguaje natural, robótica, inteligencia artificial, procesamiento de consultas en base de datos, procesamiento de consultas en Web, análisis y diseño de algoritmos, entre otros.

En este sentido, el profesor debe guiar, comentar, corregir o completar las investigaciones que el estudiante realice. Estas investigaciones deben buscar como objetivo el desarrollo de la creatividad y la integración del estudiante dentro del grupo. La creatividad permitirá vislumbrar las fronteras dentro de este campo.

Como puede apreciarse, las competencias generales que pueden estimularse son, entre otras:

- Capacidad de discernir los aspectos relevantes de investigaciones documentales
- Comunicación oral y escrita para presentar resultados de investigación documental
- Análisis y sintesis de problemas de procesamiento de información
- Integración de grupos de trabajo, a veces multidisciplinarios
- Solución de problemas a planteamientos específicos
- Toma de decisiones para determinar la mejor forma de resolver un problema
- Uso de Estándares de desarrollo para la implementación de soluciones

3.- COMPETENCIAS A DESARROLLAR

Competencias específicas:	**Competencias genéricas:**
Definir, diseñar, construir y programar las fases del analizador lexico y sintáctico de un traductor o compilador.	**Competencias instrumentales** • Capacidad de análisis y síntesis • Capacidad de organizar y planificar • Conocimientos básicos de la carrera • Comunicación oral y escrita • Habilidades del manejo de la computadora • Habilidad para buscar y analizar

	información proveniente de fuentes diversas • Solución de problemas • Toma de decisiones. **Competencias interpersonales** • Capacidad crítica y autocrítica • Trabajo en equipo • Habilidades interpersonales **Competencias sistémicas** • Estándares de desarrollo para la implementación de soluciones • Capacidad de aplicar los conocimientos en la práctica • Habilidades de investigación • Capacidad de aprender • Capacidad de generar nuevas ideas (creatividad) • Capacidad para diseñar y gestionar proyectos • Habilidad para trabajar en forma autónoma • Búsqueda del logro

4.- HISTORIA DEL PROGRAMA

Lugar y fecha de elaboración o revisión	**Participantes**	**Observaciones (cambios y justificación)**
Instituto Tecnológico de Saltillo Fecha del 5 al 9 de Octubre de 2009 Institutos Tecnológicos	Representantes de los Institutos Tecnológicos de: Representante de la	Reunión nacional de Diseño e innovación curricular de la carrera de Ingeniería en Análisis, enriquecimiento y

Superiores de: Occidente del Estado de Hidalgo y Coatzacoalcos y I.T. de Toluca Fecha 12 de Octubre 2009 al 19 de Febrero de 2010	Academia de Sistemas Computacionales	elaboración del programa de estudio propuesto en la Reunión Nacional de Diseño Curricular de la carrera de
Instituto Tecnológico de fecha	Representantes de los Institutos Tecnológicos participantes en el diseño de la carrera de Ingeniería	Reunión nacional de consolidación de la carrea de ingeniería en

5.- OBJETIVO(S) GENERAL(ES) DEL CURSO (competencias específicas a desarrollar en el curso)

Definir, diseñar, construir y programar las fases del analizador lexico y sintáctico de un traductor o compilador.

6.- COMPETENCIAS PREVIAS

- Diseñar e Interpretar algoritmos computacionales y notaciones matemáticas
- Manejar la programacion para la solución de aplicaciones
- Aplicar las estructuras de datos en la solución de problemas
- Manipular las operaciones básicas de los archivos

7.- TEMARIO

Unidad	Temas	Subtemas
1	Introducción a la Teoría de Lenguajes Formales.	1.1 Alfabeto. 1.2 Cadenas. 1.3 Lenguajes 1.4 Tipos de lenguajes 1.5 Herramientas computacionales ligadas con lenguajes 1.6 Estructura de un traductor 1.7 Fases de un compilador
2	Expresiones Regulares	2.1. Definición formal de una ER 2.2. Operaciones 2.3. Aplicaciones en problemas reales.
3	Autómatas Finitos.	3.1 Definición formal 3.2 Clasificacion de AF 3.3 Conversión de un AFND a AFD 3.4 Representación de ER usando AFND 3.5 Minimización de estados en un AF 3.6 Aplicaciones (definición de un caso de estudio)
4	Máquinas de Turing	4.1 Definición formal MT 4.2 Construcción modular de una MT 4.3 Lenguajes aceptados por la MT.
5	Análisis léxico.	5.1 Funciones del analizador léxico 5.2 Componentes léxicos, patrones y lexemas 5.3 Creación de Tabla de tokens 5.4 Errores léxicos 5.5 Generadores de analizadores Léxicos 5.6 Aplicaciones (Caso de estudio)
6	Análisis Sintáctico	6.1 GLC 6.2 Árboles de derivación. 6.3 Formas normales de Chomsky. 6.4 Diagramas de sintaxis 6.5 Eliminación de la ambigüedad.

		6.6 Generación de matriz predictiva (cálculo first y follow) 6.7 Tipos de analizadores sintácticos 6.8 Manejo de errores 6.9 Generadores de analizadores sintácticos

8.- SUGERENCIAS DIDÁCTICAS (desarrollo de competencias genéricas)

El profesor que imparta esta asignatura debe:

- Desarrollar la capacidad para coordinar y trabajar en equipo; orientar el trabajo del estudiante y potenciar en él la autonomía, el trabajo cooperativo y la toma de decisiones.
- Hacer el seguimiento del proceso formativo y propiciar la interacción entre los estudiantes.
- Proponer investigaciones en diferentes áreas (ciencias sociales, ingeniería, computación, entre otras), por grupos de interés.
- Para promover el desarrollo de capacidades de expresión oral y escrita en los estudiantes, se les invita a que presenten un proyecto de asignatura que incluya los aspectos relevantes de su investigación de campo. El proyecto incluye una presentación escrita y una oral. Todos los integrantes de cada grupo de trabajo deben participar para incentivar y promover el desarrollo de estas capacidades.
- Promover la interacción directa que permita al estudiante aprender nuevas estructuras de programación y técnicas que usan los lenguajes para procesar información.
- Propiciar actividades de metacognición. Ante la ejecución de una actividad, señalar o identificar el tipo de proceso intelectual que se realizó: una identificación de patrones, un análisis, una síntesis, la creación de un heurístico, entre otros. Al principio lo hará el profesor, luego será el estudiante quien lo identifique.
- Propiciar actividades de búsqueda, selección y análisis de información en distintas fuentes.
- Fomentar actividades grupales que propicien la comunicación, el intercambio argumentado de ideas, la reflexión, la integración y la colaboración de y entre los estudiantes.
- Facilitar el contacto directo con lenguajes y herramientas, para contribuir a la formación de las competencias para el trabajo.
- Desarrollar actividades de aprendizaje que propicien la aplicación de los conceptos, modelos y metodologías que se van aprendiendo en el desarrollo de la asignatura.
- Proponer problemas que permitan al estudiante la integración de contenidos de la asignatura y entre distintas asignaturas, para su análisis y solución.
- Propiciar el uso de las nuevas tecnologías en el desarrollo de la asignatura.

9.- SUGERENCIAS DE EVALUACIÓN

Como se ha especificado en los apartados anteriores, el aprendizaje en esta asignatura debe ser acompañado por el desarrollo de ejercicios prácticos. Cada unidad incluye ejercicios de esta naturaleza. El buen desarrollo de los mismos permitirá un aprendizaje significativo de esta asignatura. Las sugerencias son las siguientes:

- Aplicar evaluaciones diagnósticas.
- Desarrollar proyectos usando un lenguaje de programación, donde se aplique el manejo de expresiones regulares, autómatas y gramáticas formales para la construcción de las fases del analizador léxico y sintáctico de un compilador.
- Realizar investigaciones documentales referentes a la asignatura usando los diferentes medios bibliográficos o electrónicos, para desarrollar posteriormente: cuadros comparativos, mapas conceptuales, cuadros sinópticos, resúmenes y ensayos.
- Diseñar Expresiones regulares y transformarlo en AF.
- Representar, comparar, reflexionar sobre teorías o conceptos.
- Clasificar los componentes léxicos de un lenguaje, obtener su alfabeto y el lenguaje al que pertenece.
- Aplicar exámenes teórico-prácticos para detectar que tanto se ha comprendido del tema analizado.
- Realizar prácticas y ejercicios en los diferentes tópicos de la asignatura.
- Evaluar el desempeño del estudiante en el grupo utilizando instrumentos de autoevaluaciones y coevaluaciones (por ejemplo: rúbricas o listas de cotejo).
- Delimitar las especificaciones de los proyectos.

10.- UNIDADES DE APRENDIZAJE

Unidad 1: Introducción a la Teoría de Lenguajes Formales.

Competencia específica a desarrollar	**Actividades de Aprendizaje**
Expresar la notación matemática de un lenguaje formal. Identificar las fases de un compilador. Relacionar los componentes léxicos con el alfabeto.	• Identificar alfabetos y lenguajes en un cao de estudio. • Investigar la función de cada traductor. • Conocer las fases de un compilador. • Obtener un alfabeto a partir de un lenguaje. • Determinar la identificación de lexemas y componentes léxicos a partir de un lenguaje.

Unidad 2: Expresiones Regulares

Competencia específica a desarrollar	**Actividades de Aprendizaje**
Crear y reconocer ER mediante un lenguaje de programación o un analizador léxico.	• Investigar las expresiones regulares y sus operaciones. • Generar cadenas a partir de una expresión regular. • Obtener una expresión regular a partir de un grupo de cadenas o visceversa. • Obtener una expresión regular a partir de la descripción de un caso de estudio. • Elaborar por equipo, el reconocimiento de expresiones regulares mediante un lenguaje de programación o un analizador léxico.

Unidad 3: Autómatas Finitos

Competencia específica a desarrollar	**Actividades de Aprendizaje**
Crear un AF mediante un lenguaje de programación. .	• Determinar la notación formal de un AF. • Conocer la diferencia entre un AFN y AFD. • Construir un AF a partir de un ER. • Construir un AF a partir de la descripción de un caso de estudio. • Convertir un AFN a AFD. • Minimizar estados en un AF.

	• Elaborar por equipo, la simulación de un AF mediante un lenguaje de programación.

Unidad 4: Máquinas de turing

Competencia específica a desarrollar	**Actividades de Aprendizaje**
Diseñar y construir o simular una MT	• identificar la notación formal de una MT • Construir una MT a partir de un caso • Simular a través de un lenguaje de alto nivel, la representación de una MT.

Unidad 5: Análisis Léxico

Competencia específica a desarrollar	**Actividades de Aprendizaje**
Construir un analizador léxico apartir de un lenguaje de programación o un analizador léxico (p. e. Flex, Lex, JavaCC).	• Elaborar por equipo, la identificación de lexemas, componentes léxicos y patrones a partir de un lenguaje • Conocer los elementos de una tabla de tokens. • Distinguir los Errores léxicos. • Definir las reglas de un lenguaje de programación propio. • Identificar patrones válidos, generar autómatas y tabla de tokens del lenguaje propuesto. • Construir un analizador léxico (utilizar un generador de analizador léxico o un LP).

Unidad 6: Análisis Sintáctico

Competencia específica a desarrollar	**Actividades de Aprendizaje**
Construir un analizador sintáctico apartir de un lenguaje de programación o un analizador sintáctico para el reconocimiento de gramáticas (p.e. YACC).	• identificar la notación formal de una gramática. • Buscar la sintaxis de la construcción de los LP por medio de GCL o utilizando notación BNF (Backus-Naur Form). • Investigar las formas normales de Chomsky. • Conocer la notación de los diagramas

	de sintaxis. • Construir diagramas de sintaxis de un lenguaje. • Construir una GLC a partir de los diagramas de sintaxis. • Eliminar la ambigüedad de una gramática. • Distinguir los Errores sintácticos. • Construir un analizador sintáctico (utilizar un generador de analizador sintáctico o un LP).

11.- FUENTES DE INFORMACIÓN

1. Aho, Sethi, Ullman, Compiladores Principios, técnicas y herramientas, Ed. Addison Wesley.
2. Hopcroft John E., Introducción a la Teoría de Autómatas, Lenguajes y Computación, 2da ed, Ed. Addison Wesley, 2004.
3. Lemote Karen A. , Fundamentos de compiladores Cómo traducir al lenguaje de computadora, Ed. Compañía Editorial Continental.
4. Martin John, Lenguajes formales y teoría de la computación, Ed. Mc Graw Hill.
5. Kelley, Dean, *Teoría de Automatas y Lenguajes Formales,* Prentice Hall.
6. Brookshear. *Teoría de la Computación, Lenguajes Formales,Autómatas y Complejidad.* Addison Wesley.
7. Isasi, Martínez y Borrajo. *Lenguajes, Gramáticas y Autómatas.* Addison Wesley.
8. Sipser, Michael, *Introduction to the Theory of Computation,* PWS Publishing Company.
9. Cohen, Daniel I.A, *Introduction to Computer Theory,* Ed. Wie Wiley.
10. Davis, Martín D., Weyuker, Elaine. *Computability, Complexity and Languages Fundamentales of Teorical Computer Science,* Academic Press.
11. Denning, Peter J. *Machines, Languages and Computation,* Prentice Hall.
12. Dr. Sergio Gálvez Rojas y Miguel Ángel Mora Mata ,Compiladores “Traductores y Compiladores con Lex/Yacc, JFlex/Cup y JavaCC”, , http://www.lcc.uma.es/~galvez/Compiladores.html, 3/nov/2009
13. Dr. Sergio Gálvez Rojas y Miguel Ángel Mora Mata, http://www.lcc.uma.es/~galvez/tci.html,, 3/nov/2009
14. Descargar PCLEX y PCYACC, [publicación en línea], http://www.abxsoft.com/, 3/nov/2009
15. 2006, Introduction to Automata Theory, Languages, and Computation, [publicación en línea], http://www.-db.stanford.edu/~ullman/ialc,html, 22/feb/2010

12.- PRÁCTICAS PROPUESTAS

1 Realizar un cuadro comparativo de los traductores que incluya ventajas, desventajas y características.
2 Clasificar un lista de lenguajes, ambientes de desarrollo y utilerías en herramientas computacionales
3 Clasificar los componentes léxicos en un código de programa
4 Obtener un alfabeto a partir de un lenguaje y viceversa.
5 Relacionar los componentes léxicos con una Expresión regular.
6 Obtener expresiones regulares a partir de casos de estudio.
7 Realizar un programa que implemente una expresión regular
8 Realizar programas que implemente lenguajes simples representados con AFD´s
9 Realizar ejercicios de construcción de AF a partir de ER o casos de estudio
10 Realizar conversiones de AFN a AFD
11 Construir MT a partir de casos de estudio.
12 Simular a través de un lenguaje de alto nivel, la representación de una máquina de Turing
13 Definir las reglas de un lenguaje de programación propio
14 Generar el autómata correspondiente al lenguaje definido
15 Analizar la funcionalidad de diferentes generadores para análisis léxico de compilador.
16 Realizar prácticas en algún generador para analizadores léxico.
17 Construir un analizador léxico (utilizar un generador de analizador léxico o un LP)
18 Construir diagramas de sintaxis para el lenguaje propuesto.
19 Construir una GLC para el lenguaje propuesto.
20 Analizar la funcionalidad de diferentes generadores para análisis sintáctico.
21 Realizar prácticas en algún generador para analizadores sintáctico.
22 Construir un analizador sintáctico (utilizar un generador de analizador sintáctico o un LP)