UNIVERSIDAD DE SAN CARLOS DE
GUATEMALA FACULTAD DE INGENIERIA
ESCUELA DE CIENCIAS Y SISTEMAS

NOMBRE DEL CURSO: Laboratorio de Lenguajes Formales y de Programación

CODIGO:	796	**CREDITOS:**	3
ESCUELA:	Ciencias y Sistemas	**AREA A LA QUE PERTENECE:**	Ciencias de la computación
PRE REQUISITOS:	770 – Introducción a la programación 1 795 – Lógica de sistemas 960 – Matemática para computación 1	**POST REQUISITO:**	777 - Organización de Lenguajes y Compiladores 1 772 – Estructuras de Datos
CATEGORIA:	Obligatorio	**SEMESTRE:**	1er. 2018
CATEDRATICO (A):	Ing. Otto Rodríguez	**TUTOR ACADEMICO:**	Robin Navarro
EDIFICIO:	T-3	**SECCION:**	A+
SALON DEL CURSO:	212	**SALON DEL LABORATORIO:**	T-3 211
HORAS POR SEMANA DEL CURSO:	2	**HORAS POR SEMANA DEL LABORATORIO:**	2
DIAS QUE SE IMPARTE EL CURSO:	Martes	**DIAS QUE SE IMPARTE EL LABORATORIO:**	Miércoles
HORARIOS DEL CURSO:	7:10 - 8:50	**HORARIO DEL LABORATORIO:**	7:10 – 8:50

DESCRIPCION DEL CURSO

El laboratorio tiene como propósito introducir al estuante de ciencias de la computación al estudio, análisis, comprensión e implementación de lenguajes de programación bajo una estructura genérica que contribuya al desarrollo de un compilador básico y funcional, abarcando las fases de análisis léxico, análisis sintáctico e introducción al análisis semántico.

Comprende también, mostrar métodos que faciliten la creación de analizadores léxicos por medio de algoritmos programados. La parte final del laboratorio se enfocará en la enseñanza de técnicas para la creación de analizadores sintácticos, modelados mediante una gramática formal, y las técnicas para la recuperación de errores durante el análisis.

OBJETIVOS:

Objetivo General

- Que el estudiante adquiera los conocimientos básicos sobre el funcionamiento de un compilador, y los ponga en práctica en la construcción de las primeras fases de este.

Objetivos Específicos

- Aplicar las técnicas y conocimientos adquiridos en clase en la implementación de un analizador léxico.
- Crear un analizador sintáctico basándose en las técnicas y herramientas aprendidas en clase.
- Aprender los principios básicos de la creación y utilización eficiente de la tabla de símbolos, relacionándola con las fases de análisis.

METODOLOGIA

- Se impartirán clases presenciales con material audiovisual que se proporcionara al estudiante al terminar la clase.
- Durante el semestre se elaborarán tareas y exámenes cortos para complementar el contenido del curso, y que el estudiante mejore sus habilidades en el diseño y construcción de compiladores.
- Se realizarán prácticas y proyectos para poder evaluar los conceptos adquiridos en clase, tomando en cuenta que pueden incluirse temas de cursos pre requisitos.
- Además de la bibliografía recomendada, se proporcionará al alumno material complementario que le ayude a conocer distintas técnicas en la elaboración de compiladores.

RESTRICCIONES

- Copiar parciales o totales de información de internet, textos y otros, en la elaboración de las tareas serán sancionadas.
- Copias en las prácticas y proyectos serán sancionadas y reportadas al catedrático y escuela de sistemas.
- Las tareas enviadas deben poseer el formato establecido y ser enviadas en la fecha y hora establecidas.

OBSERVACIONES

- El laboratorio se debe aprobar con una nota mínima de 61 puntos, además de contar con un 80% de asistencia.
- Solo se calificarán exámenes, proyectos y demás actividades, a estudiantes asignados.
- No se guarda notas para semestres posteriores, no se pasan notas de semestres anteriores, y no se aceptan estudiantes con problemas de prerrequisitos.

EVALUACIÓN DEL REDIMIENTO ACADÉMICO: El laboratorio se evalúa sobre una nota de 100 puntos, la nota mínima de promoción es de 61 puntos, y se debe cumplir con las normas establecidas para poder aprobar el curso:

La nota final de laboratorio estará ponderada de la siguiente manera:

Aspecto	**Valor:**
Exámenes cortos	5 pts.
Practicas:	
Practica 1	15 pts.
Proyectos:	
Proyecto 1	30 pts.
Proyecto 2	40 pts.
Examen final	10 pts.

BIBLIOGRAFIA

Texto recomendado:

- Aho, Alfred V., Sethi, Ravi y Ullman. *Compiladores, principios, técnicas y herramientas*. Segunda Edición. México: Pearson Educación, 2008.

Adicional:

John E Hopcroft. Introducción a la Teoría de Autómatas, Lenguajes y computación.

Louden, Kenneth C. Construcción de compiladores/ Construction of compilers: Principios y práctica. 2004.

Eliza Viso G. Teoría de Computación. UNAM 2002

CONTENIDO:

1. **Lenguajes formales**
   - 1.1. Definición Lenguajes Formales
   - 1.2. Características de lenguajes formales
   - 1.3. Lenguajes de programación
   - 1.4. Características de lenguajes de programación.
   - 1.5. Gramáticas
2. **Introducción a compiladores**
   - 2.1. ¿Qué es un compilador?
   - 2.2. ¿Qué es un intérprete?
   - 2.3. Diferencias entre compiladores e interpretes
   - 2.4. Fases de un compilador
   - 2.5. Ejemplos de compiladores e interpretes
3. **Análisis léxico**
   - 3.1. Análisis lexicográfico
   - 3.2. Token, Patrón y Lexema
   - 3.5 Errores léxicos
4. **Jerarquía de Chomsky**
   - 4.1 Niveles
   - 4.2 Restricciones
   - 4.3 Ejemplos
5. **Lenguajes regulares y gramáticas regulares**
   - 5.1. Definición.
   - 5.2. Diseño y construcción
   - 5.3. Arboles de derivación
   - 5.4. Ejemplos y aplicaciones
6. **Expresiones regulares**
   - 6.1. Definición y propiedades
   - 6.2. Diseño y construcción
   - 6.3. Relación gramáticas regulares – Expresiones regulares
7. **Autómatas finitos**
   - 7.1. Definición
   - 7.2. Límites de los Autómatas Finitos Deterministas (AFD)
   - 7.3. Autómatas Finitos Deterministas (AFD)
   - 7.4. Autómatas Finitos No Deterministas (AFND)
   - 7.5. Implementación de AFD’s

8. **Método del árbol**
   - 8.1 Construcción de arboles
   - 8.2 Calculo de primeros, últimos y siguientes
   - 8.3 Subconjuntos y transiciones
   - 8.4 Implementación del método del árbol
9. **Implementación de un analizador léxico**
   - 9.1 Tabla de símbolos
   - 9.2 Programación de autómata finito
   - 9.3 Manejo de errores
10. **Lenguajes libres de contexto**
   - 10.1 Definición y derivaciones
   - 10.2 Diseño y construcción de gramáticas libres de contexto
   - 10.3 Ambigüedad y recursividad
   - 10.4 Factorización por la izquierda
   - 10.5 Eliminación de recursividad por la izquierda
   - 10.6 Implementación de autómatas de pila
   - 10.7 Analizado sintáctico descendente
     - 10.7.1 Analizador sintáctico descendente LL(1) predictivo
     - 10.7.2 Primero y siguientes
     - 10.7.3 Construcción de tabla de análisis sintáctico
     - 10.7.4 Validación de cadenas
     - 10.7.5 Recuperación de errores sintácticos