

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERO TÉCNICO EN INFORMÁTICA DE SISTEMAS

TERCER CURSO

SEGUNDO CUATRIMESTRE

321 - TRADUCTORES, COMPILADORES E INTÉRPRETES		
Departamento: LENGUAJES Y CIENCIAS DE LA COMPUTACION	Horas Lectivas: 45	Obligatoria
OBJETIVOS		
<p>Introducir al alumno en las técnicas de compilación de lenguajes de programación, fundamentalmente imperativos. Aplicaciones generales de dichas técnicas. Estudio de las fases y componentes del proceso de compilación. El alumno deberá ser capaz de:</p> <ul style="list-style-type: none"> - Realizar pequeños compiladores que generen código intermedio convertible a ensamblador. - Realizar intérpretes de comandos. - Analizar y resolver problemas a los que sean aplicables las técnicas propias de la compilación. - Confeccionar analizadores sintácticos y lexicográficos en entornos software que carezcan de herramientas de generadores automáticos. 		
CONTENIDO		
<p>BLOQUE TEMATICO: Introducción</p> <p>1. INTRODUCCIÓN.</p> <p>1.1. Generalidades.</p> <p>1.2. Conceptos de: Traductores, intérpretes y compiladores.</p> <p>1.3. Esquema general de un compilador. Fases.</p> <p>BLOQUE TEMATICO: Etapa de Análisis</p> <p>2. ANÁLISIS LEXICOGRÁFICO.</p> <p>2.1. Funciones del análisis lexicográfico.</p> <p>2.2. Conceptos de: Expresión regular, lexema y token.</p> <p>2.3. El generador de analizadores lexicográficos LEX.</p> <p>3. ANÁLISIS SINTÁCTICO.</p> <p>3.1. Funciones del análisis sintáctico.</p> <p>3.2. Análisis sintáctico descendente.</p> <p>3.2.1. Con retroceso.</p> <p>3.2.2. Análisis con funciones recursivas. Diagramas de sintaxis.</p> <p>3.2.3. LL(1). Manejo de tablas.</p> <p>3.3. Análisis sintáctico ascendente.</p> <p>3.3.1. Con retroceso.</p> <p>3.3.2. LR(1). Manejo de tablas.</p> <p>3.3.3. El generador de analizadores sintácticos YACC.</p> <p>4. GRAMÁTICAS ATRIBUIDAS.</p> <p>4.1. Compilación dirigida por sintaxis. Necesidad de atributos.</p> <p>4.2. Atributos calculados, sintetizados y heredados.</p> <p>4.3. Esquemas de traducción.</p> <p>4.4. Selección de gramáticas.</p> <p>5. LA TABLA DE SÍMBOLOS.</p> <p>5.1. Necesidad de la tabla de símbolos.</p> <p>5.2. Características de la tabla de símbolos.</p> <p>5.3. Lenguajes con estructura de bloques.</p> <p>6. SISTEMAS DE TIPOS.</p> <p>6.1. Expresiones de tipos.</p> <p>6.2. Tipos complejos.</p> <p>6.3. Equivalencia estructural, nominal y funcional.</p> <p>BLOQUE TEMATICO: Etapa de Síntesis</p> <p>7. GENERACIÓN DE CÓDIGO.</p> <p>7.1. El código intermedio.</p> <p>7.2. Máquinas abstractas.</p> <p>7.3. Generación de código en sentencias imperativas.</p> <p>7.4. Necesidad de la optimización.</p> <p>BLOQUE TEMATICO: Tiempo de ejecución</p> <p>8. GESTIÓN DE LA MEMORIA.</p> <p>8.1. Organización de la memoria en tiempo de ejecución.</p> <p>8.2. La memoria estática.</p> <p>8.3. La memoria dinámica. La pila. Registros de activación.</p> <p>8.4. El montón. Recolección de basura.</p>		
BIBLIOGRAFÍA		

General	
<p>Aho A., Lam M., Sethi R., Ullman J. <i>Compilers: Principles, Techniques and Tools</i> Addison-Wesley 2006 Bennet J.P. <i>Introduction to compiling techniques. A first course using ANSI C, LEX and YACC</i> McGraw-Hill 1990 Gálvez Rojas S. <i>Traductores y Compiladores con Lex/Yacc, JFlex/Cup y JavaCC</i> 2004 Versión electrónica Holub A. <i>Compiler design in C</i> Prentice-Hall 1990 Louden K.C. <i>Construcción de compiladores</i> Thomson 2004 Garrido A., Iñesta J.M., Moreno F., Pérez J.A. <i>Diseño de compiladores</i> Wiley 1981 Alfonseca M., de la Cruz M., Ortega A., Pulido E. <i>Compiladores e intérpretes: teoría y práctica</i></p>	
Específica	
<p>Tremblay J.P., Sorenson P.G. <i>The theory and practice of compiler writing</i> McGraw-Hill 1985 - Mak R. <i>Writing compilers and interpreters</i> -</p>	
METODOLOGÍA DOCENTE	
<p>Las clases serán minoritariamente de pizarra, en las que se explicarán los conceptos fundamentales, instando a la participación del alumno para que sus razonamientos evolucionen a la vez que los conceptos explicados. La mayor parte de las clases teóricas se realizarán con apoyo de ordenador en el que se podrá observar la creación progresiva de pequeños intérpretes que solucionan cada uno de los problemas parciales de los que consta la creación de un compilador.</p> <p>Los ejemplos y clases prácticas se verán apoyados por elementos visuales, fundamentalmente cañón de proyección, con el que se explicarán casos prácticos en los que intervienen los conceptos teóricos. Los ejemplos se harán por entero en tiempo real en la propia clase de forma que el alumno pueda apreciar el orden en que se construye un compilador o intérprete:</p> <ul style="list-style-type: none"> - Determinación informal de programas de ejemplo a reconocer por nuestro traductor - Creación de una gramática de contexto libre acorde al lenguaje a reconocer - Creación de árboles sintácticos que reconozcan sentencias de ejemplo - Asociación de atributos a los distintos componentes de la gramática - Creación de la estructura de la Tabla de Símbolos - Creación del Analizador Léxico y sus acciones léxicas asociadas - Adición de reglas semánticas al Analizador Sintáctico en el orden en que las reglas de producción se reducen <p>Estos programas hechos en la propia clase se darán a los alumnos mediante el Campus Virtual de la asignatura, Campus que se utilizará asiduamente a lo largo de todo el curso para proporcionar material e información de todo tipo.</p> <p>Es especialmente importante al principio de la asignatura que el alumno comience a manejar las herramientas informática necesarias para construir un compilador de un lenguaje imperativo sin demasiadas dificultades. Por ello se hará especial hincapié en que asistan a las tutorías para resolver cualquier duda. El horario y celebración de las tutorías quedará prefijado de antemano al principio de curso. Igualmente se hará uso de los medios proporcionados por la ETSI Informática (apoyo de alumnos veteranos) para que aconsejen y ayuden a los alumnos de la asignatura en cualquier otro aspecto relevante para la adquisición de los conocimientos necesarios.</p> <p>También se suministrarán a los alumnos copias de versiones gratuitas de metacompiladores, para que puedan manipular en la práctica todos los ejercicios propuestos, así como los resueltos en clase.</p> <p>Igualmente se proporcionarán todos los enunciados de los exámenes de cursos pasados, así como sus soluciones prácticas, con el objetivo de que el alumno pueda abordarlos como ejercicios y poder comparar sus resultados con los correctos.</p>	
EVALUACION	
<p>La evaluación se hará mediante una evaluación continua que consta de dos bloques:</p> <p>1.- Tareas semanales.</p> <p>Una o dos veces por semana se propondrá a los alumnos una tarea relacionada con el temario impartido durante las últimas clases. Mediante estas tareas se desea llevar un control de la participación activa del alumnado y de su asistencia real a clase. Se procurará que dichas tareas se encuentren fuertemente acopladas a los pormenores explicados en clase de manera que su correcta resolución sólo pueda conseguirse mediante una asistencia regular a las clases presenciales.</p> <p>Este bloque tiene por objetivo subir la calificación general obtenida en el bloque siguiente y, por tanto, incentivar la asistencia a clase. La proporción en que esta calificación influya en la nota final depende tanto de la calidad como de la cantidad de ejercicios abordados por el alumno.</p> <p>Estos trabajos tratarán sobre los temas principales y anejos al temario oficial y servirán para reforzar conocimientos y mantener más vivo el interés del alumno en la asignatura.</p> <p>2.- Evaluación continua.</p> <p>Este bloque constará de tres pruebas a realizar durante el periodo lectivo, toda vez que se hayan suministrado al alumno los conocimientos o medios necesarios para su total superación. Las pruebas serán prácticas y podrán ser realizadas tanto en laboratorio, en casa o en el aula de teoría ya que se tratará de escribir programas en Lex/Yacc.</p> <p>Con objeto de que el alumno conozca los más rápidamente posible la calificación obtenida planteamos la posibilidad de utilizar la plataforma SIETTE en su versión más actual. Aunque SIETTE era originalmente una plataforma para realizar tests, la última versión permite plantear enunciados al alumno a los cuales éste debe responder con programas completos, encargándose el propio sistema de pasar a dichos programas una batería completa de pruebas y asignarle una puntuación automática en función de las pruebas que haya podido superar. Si la plataforma está lo suficientemente madura desde un punto de vista tecnológico consideraremos su utilización previa comunicación y formación a los alumnos sobre su funcionamiento.</p> <p>Las tres pruebas tendrán un peso del 20%, 30% y 50% sobre la calificación final y versarán sobre Lex, Yacc ampliado (análisis sintáctico y gestión básica de atributos) y Yacc avanzado respectivamente. Se realiza la media ponderada de los tres ejercicios y, el resultado junto con la estimación del bloque anterior da lugar a la calificación final.</p> <p>Para aprobar la asignatura es necesario obtener una media del 50% de los puntos de dichos parciales. En cualquier caso, y especialmente si el alumno no supera dicho 50%, si alguna de estas tres pruebas no es superada satisfactoriamente el alumno dispondrá de una segunda oportunidad para superarla el día del examen final oficial.</p> <p>En cualquier caso, la asistencia a clase es obligatoria con el objetivo de ir adaptando la metodología de evaluación a la de los nuevos grados.</p>	
TÉCNICAS DOCENTES	
Sesiones académicas teóricas:	SI
Sesiones académicas prácticas:	SI
Exposición y debate:	NO

Tutorías especializadas:	NO
Visitas y excursiones:	NO
Controles de lectura obligatorias:	NO
Otros:	Razonamiento evolutivo
Desarrollo y Justificación:	

Segundo Semestre

Actividad	Nº de horas
Clases Teóricas:	30
Clases Prácticas:	15
Exposiciones y Seminarios:	
Tutorías Especializadas (presenciales o virtuales):	
A) Colectivas:	
B) Individuales:	
Realización de Actividades Académicas Dirigidas:	
A) Con presencia del profesor:	
B) Sin presencia del profesor:	
Otro Trabajo Personal Autónomo:	
A) Horas de estudio	55
B) Preparación de Trabajo Personal:...	5
C) ?	
Realización de Exámenes:	
A) Examen escrito:	10
B) Exámenes orales (control del Trabajo Personal):	

DESCRIPTOR

- Lenguajes de Programación
- Compiladores e Intérpretes
- Fases de Compilación
- Semántica de sentencias imperativas
- Gramáticas atribuidas
- Estructuras de datos

SITUACIÓN

Contexto dentro de la situación

Esta asignatura supone una culminación en el proceso de formación de un informático de sistemas, por lo que se halla sustentada por los conocimientos relativos a ciertas áreas teóricas que ven su utilidad práctica en esta y otras asignaturas. De un lado, el conocimiento adquirido por la teoría de autómatas y lenguajes formales es básica para poder construir gramáticas que reconozcan lenguajes de programación y comprender el proceso automático de reconocimiento. De otro, las estructuras de datos y los algoritmos de manipulación juegan un papel importantísimo en la comprensión de los mecanismos de ejecución de las subrutinas cuya ejecución se entrelaza para realizar el proceso de compilación.

COMPETENCIAS TRANSVERSALES/GENÉRICAS

La Programación es la base de la Informática actual desde el punto de vista del Software. Por ello una asignatura como ésta dota al alumno de una visión general de todos los aspectos de la Informática que se ven influenciados por el uso de los lenguajes de programación. No es, por tanto, una asignatura Transversal General sino Específica en el campo de la Informática.

COMPETENCIAS ESPECÍFICAS

Cognitivas(Saber)

El alumno deberá aprender el funcionamiento básico de un compilador y de un intérprete, las fases que lo componen y los objetivos de cada una. Asimismo debe darse cuenta de las relaciones entre cada fase y la siguiente, así como entre cada fase y los recursos de la plataforma informática a la que va destinada el traductor.

El alumno hará uso de conceptos de asignaturas anteriores para comprender la importancia de las estructuras de datos de alto rendimiento, así como de las estructuras básicas de control computacional.

Procedimentales/Instrumentales (Saber hacer)

El alumno deberá ser capaz de diseñar lenguajes de programación mediante reglas de producción y diagramas de sintaxis, así como trocear las cadenas de entrada a reconocer sintácticamente y a comunicar las dos primeras fases de la etapa de análisis.

También deberá conocer los distintos tipos de análisis sintáctico existentes y sus limitaciones dentro de la teoría de los lenguajes formales y de autómatas finitos deterministas.

Por último aprenderá las reglas básicas que rigen el funcionamiento de las gramáticas atribuidas y de los esquemas de traducción lo que abrirá las puertas a un nuevo mundo en el control y manipulación de los lenguajes generados por gramáticas formales. A este respecto deberá saber aplicar ciertos patrones comunes en el uso de atributos y de acciones semánticas asociadas a las reglas de producción de una gramática. En resumen:

- Saber plantear soluciones algorítmicas a problemas concretos
- Visualizar e interpretar adecuadamente soluciones ya dadas
- Diseñar e implementar algoritmos ad hoc
- Adaptarse al diseño construido por otros grupos de trabajo
- Identificar, localizar y corregir errores
- Conectar los aspectos vistos en teoría con la práctica en ordenador
- Utilización de herramientas (Lex y Yacc)