GUÍA DOCENTE

PROCESADORES DE LENGUAJE

Coordinación: PLANES CID, JORDI

Año académico 2017-18

PROCESADORES DE LENGUAJE 2017-18

Información general de la asignatura

Denominación	PROCESADORES DE LENGUAJE
Código	102043
Semestre de impartición	2o Q(SEMESTRE) EVALUACIÓN CONTINUADA
Carácter	<table><tr><th>Grado/Máster</th><th>Curso</th><th>Carácter</th><th>Modalidad</th></tr><tr><td>Grado en Ingeniería Informática</td><td>4</td><td>OBLIGATORIA</td><td>Presencial</td></tr></table>
Número de créditos ECTS	9
Grupos	1GG
Créditos teóricos	2
Créditos prácticos	7
Coordinación	PLANES CID, JORDI
Departamento/s	INFORMATICA I ENGINYERIA INDUSTRIAL
Distribución carga docente entre la clase presencial y el trabajo autónomo del estudiante	70 horas de clase presencial / 115 horas de trabajo autónomo
Información importante sobre tratamiento de datos	Consulte este enlace para obtener más información.
Idioma/es de impartición	Catalán
Horario de tutoría/lugar	Para concretar la hora de las tutorias enviar un correo electrónico al professor.

PROCESADORES DE LENGUAJE 2017-18

Profesor/a (es/as)	Dirección electrónica profesor/a (es/as)	Créditos impartidos por el profesorado	Horario de tutoría/lugar
ALSINET BERNADÓ, MARIA TERESA	tracy@diei.udl.cat	2	
PLANES CID, JORDI	jplanes@diei.udl.cat	7	

Información complementaria de la asignatura

Se recomienda conocimientos previos de Modelos de computación y complejidad.

Objetivos académicos de la asignatura

Los objetivos de aprendizaje de la asignatura son:

- Conocer las etapas, técnicas y algoritmos que intervienen en el proceso de traducción de los lenguajes de programación,
- Utilizar herramientas de soporte en el diseño e implementación de cada etapa.
- Analizar las características principales y las técnicas de implementación asociadas a lenguajes no imperativos como los lenguajes lógicos, funcionales, de scripting, orientados a objetos, y distribuidos y concurrentes.

Competencias

Competencias estratégicas de la Universidad de Lleida

- **CT3**. Adquirir capacitación en el uso de las nuevas tecnologías y de las tecnologías de la información y la comunicación.
- **CT2**. Adquirir un dominio significativo de una lengua extranjera, especialmente del inglés.

Competencias transversales de la titulación

- **EPS6**. Capacidad de análisis y síntesis.

Competencias específicas de la titulación

- **GII-C2**. Capacidad para conocer los fundamentos teóricos de los lenguajes de programación y las técnicas de procesamiento léxico, sintáctico y semántico asociadas, y saber aplicarlas para la creación, diseño y procesamiento de lenguajes.

Contenidos fundamentales de la asignatura

Estructura de la asignatura en temas:

1- Introducción a los lenguajes de programación y a las técnicas de traducción.

2- Análisis léxico.

3- Herramienta: flex (The Fast Lexical Analyzer)

4 - Análisis sintáctico: Analizadores sintácticos descendentes y ascendentes.

5- Herramienta: Yacc (Yet Another Compiler Compiler)

6- Traducción dirigida por la sintaxis

7- Tablas de símbolos

8- Herramienta: SymTab

9- Comprobación de tipos

10- Representaciones intermedias y generación de código intermedio

11- Gestión de memoria en el entorno de ejecución

12- Optimización de código

13- Generación de código objeto

Ejes metodológicos de la asignatura

Las clases de la asignatura se estructuran en 4 horas semanales de resolución de problemas prácticos en el laboratorio y 2 horas semanales de carácter expositivo donde se presentarán los algoritmos, técnicas y herramientas de traducción de cada etapa del proceso de traducción.

Los estudiantes resolverán ejercicios prácticos durante las sesiones de laboratorio, abordarán la preparación y presentación de un lenguaje de programación no clásico en el resto de grupos, e implementarán un compilador.

Plan de desarrollo de la asignatura

El temario de la asignatura se estructura en dos partes.

La **primera parte** aborda la especificación y reconocimiento de los componentes léxicos de los lenguajes de programación, estudia las técnicas de análisis sintáctico y muestra cómo integrar las rutinas semánticas con los algoritmos de análisis sintáctico. La formación del estudiante se complementa con el estudio de herramientas especializadas de apoyo al diseño y implementación de componentes específicos de los sistemas de traducción. Bajo este marco presentamos la herramienta de especificación y reconocimiento de lenguajes JFLAP, la herramienta de generación de analizadores léxicos flex, de generación de analizadores sintácticos ascendentes yacc y de gestión de tablas de símbolos SymTab.

La **segunda parte** de la asignatura aborda las fases de análisis semántico, optimización de código y generación de código objeto. Mostramos cómo incorporar al proceso de análisis rutinas semánticas que permitan la gestión de ámbitos, la comprobación de tipos, la generación de código intermedio para las principales construcciones de los lenguajes imperativos y la asignación de memoria. Se estudian optimizaciones de código dependientes de la representación intermedia y de la máquina objeto, y cómo generar código para una máquina objeto.

Finalmente, además de los lenguajes imperativos clásicos analizaremos las características principales y las técnicas de implementación asociadas a otros lenguajes como los lenguajes lógicos, funcionales, de scripting, orientados a objetos, y distribuidos y concurrentes. Para su estudio y análisis, los estudiantes escogerán un lenguaje el que presentarán al resto del grupo.

Semana	Descripción	Actividad Presencial Grupo Grande	Trabajo presencial/autónom
1	Clase magistral y problemas	Tema 1	6h/9h

Semana	Descripción	Actividad Presencial Grupo Grande	Trabajo presencial/autónom
2	Clase magistral y problemas	Tema 2,3	6h/9h
3	Clase magistral y problemas	Tema 2,3	6h/9h
4	Clase magistral y problemas	Tema 2,3	6h/9h
5	Clase magistral y problemas	Tema 4,5	6h/9h
6	Clase magistral y problemas	Tema 4,5	6h/9h
7	Clase magistral y problemas	Tema 4,5	6h/9h
8	Clase magistral y problemas	Tema 4,5	6h/9h
9		**Primer parcial**	
10	Clase magistral y problemas	Tema 6	6h/9h
11	Clase magistral y problemas	Tema 7,8	6h/9h
12	Clase magistral y problemas	Tema 9,10	6h/9h
13	Clase magistral y problemas	Tema 11,12	6h/9h
14	Clase magistral y problemas	Tema 13	6h/9h
15	Pràctiques	Presentación Práctica	6h/9h
16		**Segundo parcial**	
17		**Segundo parcial**	
18		Semana de estudio	
19		**Recuperación**	

Sistema de evaluación

Acrónimo	Actividades de evaluación	Ponderación	Nota mínima	En grupo	Obligatoria	Recuperable
PR1	JFLAP	10%	-	No	Sí	Sí
P1	Lex	10%	-	No	Sí	Sí
P2	Yacc	15%	-	No	Sí	Sí
P3	Proyecto	45%	-	Sí	Sí	Sí
A1	Análisis lenguage	15%	-	Sí	Sí	Sí
A2	Expresiones regulares	5%	-	No	Sí	Sí

Las actividades del curso consisten en:

- JFLAP: Resolución de una colección de problemas de especificación y reconocimiento de lenguajes empleando la herramienta JFLAP
- Lex: uso de la herramienta de generación automática de analizadores léxicos lex.
- YACC uso de la herramienta de generación automática de analizadores sintácticos yacc.
- Proyecto: uso de la herramienta de implantación de tablas de símbolos e integración con las herramientas lex y yacc. Se propone desarrollar un traductor en código intermedio (3-direcciones) para un lenguaje imperativo reducido.
- Análisis de lenguaje: Análisis de las características propias de un lenguaje de programación no clásico. Aspectos de implementación. Trabajo a realizar en grupo de dos o tres personas. Presentación oral al resto de grupo.
- Expresiones regulares: Actividad de expresiones regulares: realizar los ejercicios de la Práctica 1 en un lenguaje de programación.

Bibliografía y recursos de información

Además del material disponible en el apartado de Recursos, puede consultar las siguientes referencias:

Referencias:

[1] A.V. Aho, M. Lam, R. Sethi, and J.D. Ullman. Compilers: Principles, Techniques, and Tools. Addison-Wesley Series in Computer Science, Reading, Massachusetts. Second Edition. 2006.

[2] Dick Grune, Henri E. Bal, Ceriel J. H. Jacobs, Koen G. Langendoen. Modern Compiler Design. Jonh Wiley and Sons, England, 2000.

[3] Andrew W. Appel, Maia Ginsburg. Modern Compiler Implementation in C. Cambridge University Press, 1998.

[4] John Levine. Flex & bison: Text Processing Tools. O'Reilly, 2009.

[5] Reinhard Wilhelm, Helmut Seidl, Sebastian Hack: Compiler Design - Syntactic and Semantic Analysis. Springer 2013.

[6] Helmut Seidl, Reinhard Wilhelm, Sebastian Hack: Compiler Design - Analysis and Transformation. Springer 2012.

[7] Reinhard Wilhelm, Helmut Seidl: Compiler Design - Virtual Machines. Springer 2010

Herramientas:

- Flex: http://flex.sourceforge.net/
- Yacc: http://www.gnu.org/software/bison/
- JFLAP: http://www.jflap.org/jflaptmp/
- JFlex: http://jflex.de/
- Cup: http://www2.cs.tum.edu/projects/cup/
- Ant: http://ant.apache.org/
- ANTLR**:** http://www.antlr.org/