Universitat de Lleida

GUÍA DOCENTE

PROCESADORES DE LENGUAJE

Coordinación: ALSINET BERNADO, MARIA TERESA

Año académico 2023-24

PROCESADORES DE LENGUAJE 2023-24

Información general de la asignatura

<table>
<tr><td>Denominación</td><td>PROCESADORES DE LENGUAJE</td></tr>
<tr><td>Código</td><td>102043</td></tr>
<tr><td>Semestre de impartición</td><td>2o Q(SEMESTRE) EVALUACIÓN CONTINUADA</td></tr>
<tr><td>Carácter</td><td>
<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>
<tr><td>Grado en Ingeniería Informática</td><td>4</td><td>OPTATIVA</td><td>Presencial</td></tr>
</table>
</td></tr>
<tr><td>Número de créditos de la asignatura (ECTS)</td><td>9</td></tr>
<tr><td>Tipo de actividad, créditos y grupos</td><td>
<table>
<tr><th>Tipo de actividad</th><td>PRALAB</td><td>TEORIA</td></tr>
<tr><th>Número de créditos</th><td>3.6</td><td>5.4</td></tr>
<tr><th>Número de grupos</th><td>1</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>Coordinación</td><td>ALSINET BERNADO, MARIA TERESA</td></tr>
<tr><td>Departamento/s</td><td>INGENIERÍA INFORMÁTICA Y DISEÑO DIGITAL</td></tr>
<tr><td>Distribución carga docente entre la clase presencial y el trabajo autónomo del estudiante</td><td>70 horas de clase presenciales / 115 horas de trabajo autónomo</td></tr>
<tr><td>Información importante sobre tratamiento de datos</td><td>Consulte este enlace para obtener más información.</td></tr>
<tr><td>Idioma/es de impartición</td><td>Catalán</td></tr>
<tr><td>Distribución de créditos</td><td>3 horas a la semana de teoría.
3 horas a la semana de laboratorio.
La asignatura se desarrollará en formato presencial, aunque podrá adaptarse fácilmente a un formato virtual si se requiere.</td></tr>
</table>

PROCESADORES DE LENGUAJE 2023-24

Profesor/a (es/as)	Dirección electrónica\nprofesor/a (es/as)	Créditos impartidos por el profesorado	Horario de tutoría/lugar
ALSINET BERNADO, MARIA TERESA	teresa.alsinet@udl.cat	9	EPS. Despacho 2.13 Enviar un correo electrónico a la profesora para establecer el horario de la tutoría conveniente al estudiante.

Información complementaria de la asignatura

Se recomienda conocimientos previos de:

- Algorítmica y complejidad
- Lenguajes, autómatas y gramáticas
- Modelos de computación y complejidad.

Objetivos académicos de la asignatura

Los resultados de aprendizaje del estudiante en la asignatura son:

- Conocer, diseñar e implementar las etapas que intervienen en el proceso de traducción de los lenguajes de programación,
- Especificar formalmente las características de los lenguajes de programación.
- Saber aplicar las técnicas y algoritmos del proceso de traducción a la implementación de lenguajes formales.
- Utilizar las herramientas de soporte al diseño e implementación de cada etapa de un proceso de traducción.
- Conocer y analizar las principales características y las técnicas de implementación asociadas a lenguajes no imperativos como los lenguajes lógicos, funcionales, de scripting, orientados a objetos, y distribuidos y concurrentes.
- Conocer la evolución de los lenguajes de programación y de los correspondientes intérpretes y traductores.

Competencias

Competencias estratégicas de la Universidad de Lleida

- **CT3: Dominio de las Tecnologías de la Información y la comunicación.**
- **CT2: Dominio de una lengua extranjera.**

Competencias transversales de la titulación

- EPS6. Capacidad de análisis y síntesis.

Competencias específicas de la titulación

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

PROCESADORES DE LENGUAJE 2023-24

Contenidos fundamentales de la asignatura

Estructura de la asignatura en temas:

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

2- Análisis léxica.

3- flex (The Fast Lexical Analyzer)

4 - Análisis sintáctico: Analizadores sintácticos descendentes y ascendentes.

5- Yacc (Yet Another Compiler Compiler)

6- Traducción dirigida por la sintaxis

7- Tablas de símbolos

8- SymTab

9- Comprobación de tipo

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 3 horas semanales orientadas a la resolución de problemas prácticos en el laboratorio y 3 horas semanales de carácter más expositivo donde se presentarán los algoritmos, técnicas y herramientas de traducción de cada etapa del proceso de traducción.

La asignatura se desarrollará en formato presencial, aunque podrá adaptarse fácilmente a un formato virtual si se requiere.

Los estudiantes resolverán ejercicios prácticos durants las sesiones de laboratorio y abordarán de manera grupal la preparación y presentación de tres trabajos:

- Uso de las expresiones regulares en los lenguajes de programación
- Aspectos de diseño y implkementació de un lenguaje de programación particular
- Fase de síntesis de los traductores: Gestión de memoria en el entorno de ejecución y Optimización de código

Plan de desarrollo de la asignatura

El temario de la asignatura se estructura en **dos partes**.

La **primera** 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 e 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 generar código para una máquina objeto.

Además, el estudiante abordará el estudio y aplicación de las expresiones regulares a los lenguajes de programación y escogerá un lenguaje de programación y analizará las principales características de diseño e implementación del lenguaje. Los resultados de aprendizaje serán presentados de forma oral al resto de grupo.

Semana	Descripción	Actividad	Trabajo supervisado/autónomo
1	Clase expositiva y problemas	Tema 1, 2	6h/9h
2	Clase expositiva y problemas	Tema 2, 3	6h/9h
3	Actividad 1 y práctica 1	Trabajo en grupo	6h/9h
4	Presentaciones grupales	Actividad 1: Expressiones regulares y lenguajes	6h/9h
5	Clase expositiva y problemas	Tema 4	6h/9h
6	Clase expositiva y problemas	Tema 5	6h/9h
7	Actividad 2 y práctica 2	Trabajo en grupo	6h/9h
8	Presentaciones grupales	Actividad 2: Diseño de lenguajes	6h/9h
9		Primer parcial	
10	Clase expositiva y problemas	Tema 6, 7	6h/9h
11	Clase expositiva y problemas	Tema 8, 9	6h/9h
12	Práctica 3	Trabajo en grupo	6h/9h
13	Clase expositiva y problemas	Tema 10, 11	6h/9h
14	Clase expositiva y problemas	Tema 12, 13	6h/9h
15	Práctica 4	Trabajo en grupo	6h/9h

Sistema de evaluación

La evaluación continua de la asignatura se basa en 5 bloques:

1. **Bloque de actividades: 20%.** Consiste en dos actividades: **Presentación 1 (10%) y Presentación 2 (10%)**. No son recuperables. No tienen nota mínima. Fecha de la Presentación 1: Durante la 4a semana. Fecha de la Presentación 2: Durante la 8a semana.
2. **Bloque de análisis léxico: 20%**. Consiste en una práctica: **Práctica 1**. Recuperable. No tiene nota mínima. Fecha entrega de la Práctica 1: Durante la 6a semana. La fecha para la recuperación de la Práctica 1 será la fijada por la EPS para la realización del examen del 1er parcial de la asignatura.
3. **Bloque de análisis sintáctico: 20%**. Consiste en una práctica: **Práctica 2**. Recuperable. No tiene nota mínima. Fecha entrega de la Práctica 2: Durante la 10a semana. La fecha para la recuperación de la Práctica 2 será la fijada por la EPS para la realización del examen del 2º parcial de la asignatura.
4. **Bloque de análisis semántico: 20%**. Consiste en una práctica: **Práctica 3**. Recuperable. No tiene nota mínima. Fecha entrega de la práctica 3: Durante la 14a semana. La fecha para la recuperación de la Práctica 3 será la fijada por la EPS para la realización del examen de recuperación de la asignatura.
5. **Bloque de generación de código: 20%**. Consiste en una práctica: **Práctica 4**. Recuperable. No tiene nota mínima. Fecha entrega de la práctica 4: Fecha del examen del 2º parcial fijada por la EPS. La fecha para la recuperación de la Práctica 4 será la fijada por la EPS para la realización del examen de recuperación de la asignatura.

Recuperación de los Bloques: Consiste en una nueva fecha para entregar la práctica correspondiente. La realización de la recuperación de los bloques no condiciona la calificación máxima alcanzable en la asignatura.

Actividades de evaluación

Acronimo	Actividad de evaluación	Ponderación	Nota Mínima	En grupo	Obligatoria	Recuperable
PRE1	*Presentación 1*	10%	NO	SI (<=3)	NO	NO
PRE2	*Presentación 2*	10%	NO	SI (<=3)	NO	NO
PRA1	*Práctica 1*	20%	NO	SI (<=3)	NO	SI
PRA2	*Práctica 2*	20%	NO	SI (<=3)	NO	SI
PRA3	*Práctica 3*	20%	NO	SI (<=3)	NO	SI
PRA4	*Práctica 4*	20%	NO	SI (<=3)	NO	SI
Para aprobar la asignatura, la nota final debe ser >= 5						
Nota Final = 0,1*PRE1 + 0,1*PRE2 + 0,2*PRA1 + 0,2**PRA2* + 0,2**PRA3* + 0,2**PRA4*						

Las actividades del curso consisten en:

- **Práctica 1**: El estudiante debe hacer uso de la herramienta de generación automática de analizadores léxicos lex.
- **Práctica 2**: El estudiante debe hacer uso de la herramienta de generación automática de analizadores sintácticos yacc.
- **Práctica 3**: El estudiante debe hacer uso de la herramienta de implementación de la tabla de símbolos y su integración con las herramientas lex y yacc.
- **Práctica 4**: El estudiante debe implementar un traductor a código intermedio (3-direcciones) para un lenguaje imperativo reducido.
- **Presentación 1**: Expresiones regulares. Estudio y aplicación de las expresiones regulares en los lenguajes de programación, herramientas de generación de analizadores léxico, implementación de la fase de análisis léxico. Presentación oral al resto de grupo.
- **Presentación 2**: Estudio de los aspectos de diseño de los lenguajes de programación. Presentación oral al resto de grupo.
- Las **prácticas** se desarrollarán de forma individual o en grupo de dos o tres personas. En caso de realizarse en grupo, cada estudiante autoevaluará su participación en el desarrollo de la práctica.
- Las **presentaciones** se desarrollarán en grupo de dos o tres personas. La evaluación de las presentaciones será colaborativa entre todos los asistentes a las presentaciones. Además, en cada grupo cada estudiante autoevaluará su participación en el desarrollo de la actividad.

Evaluación alternativa (estudiantes que renuncian a la evaluación continua):

- **Práctica 1: 25%**. Individual. Recuperable. No tiene nota mínima. Fecha de entrega de la Práctica 1: Fecha del examen del 1er parcial fijada por la EPS. La fecha para la recuperación de la Práctica 1 será la fijada por la EPS para la realización del examen del 2º parcial de la asignatura.
- **Práctica 2: 25%**. Individual. Recuperable. No tiene nota mínima. Fecha de entrega de la Práctica 2: Fecha del examen del 1er parcial fijada por la EPS. La fecha para la recuperación de la Práctica 2 será la fijada por la EPS para la realización del examen del 2º parcial de la asignatura.
- **Práctica 3: 25%**. Individual. Recuperable. No tiene nota mínima. Fecha de entrega de la Práctica 3: Fecha del examen del 2º parcial fijada por la EPS. La fecha para la recuperación de la Práctica 3 será la fijada por la EPS para la realización del examen de recuperación de la asignatura.
- **Práctica 4: 25%**. Individual. Recuperable. No tiene nota mínima. Fecha de entrega de la Práctica 3: Fecha del examen del 2º parcial fijada por la EPS. La fecha para la recuperación de la Práctica 4 será la fijada por la EPS para la realización del examen de recuperación de la asignatura.
- **Recuperación de las Prácticas**: Consiste en una nueva fecha para entregar la práctica correspondiente. La presentación de la práctica en la fecha de recuperación no condiciona la calificación máxima alcanzable en la asignatura.

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 de generación en C/C++:

- Flex: http://flex.sourceforge.net/
- Yacc: http://www.gnu.org/software/bison/

Herramienta de especificación de llenguatges:

- JFLAP: http://www.jflap.org/jflaptmp/

Herramientas de generación en Java:

- JFlex: http://jflex.de/
- Cup: http://www2.cs.tum.edu/projects/cup/
- Ant: http://ant.apache.org/
- ANTLR**:** http://www.antlr.org/

Herramientas de generación en Phyton:

- https://pypi.org/project/ply/

Herramientas de generación en Haskell:

- https://www.haskell.org/alex/
- https://www.haskell.org/happy/