Universidad de Alcalá

GUÍA DOCENTE

PROCESADORES DEL LENGUAJE

Grados en extinción (sin docencia)

Grado en Ingeniería Informática
Grado en Ingeniería de Computadores

Universidad de Alcalá

Curso Académico 2019/2020

Curso 3º – Cuatrimestre 1º

GUÍA DOCENTE

Nombre de la asignatura:	**Procesadores del lenguaje**
Código:	**780018**
Titulación en la que se imparte:	**Grado en Ingeniería Informática** **Grado en Computadores**
Departamento:	**Departamento Ciencias de la Computación**
Carácter:	**Obligatoria**
Créditos ECTS:	**6**
Curso y cuatrimestre:	**3º Curso / 1º Cuatrimestre**
Profesorado:	Salvador Sánchez Alonso José Luis Cuadrado José Carlos Holgado (información siempre actualizada en la página web del departamento)
Horario de Tutoría:	Se indicará el primer día de clase.
Idioma en el que se imparte:	Español

1a. PRESENTACIÓN

La asignatura Procesadores del Lenguaje estudia el proceso de traducción de los lenguajes con carácter general y los lenguajes de programación en particular, las técnicas de traducción de los mismos, los módulos que componen los procesadores de lenguaje clásicos y las formas en que dichos módulos se comunican entre sí. El objetivo principal es introducir a los estudiantes en la teoría y la práctica de la traducción de lenguajes.

Se organiza como una asignatura de 6 ECTS, donde se explora la teoría de la traducción de lenguajes y se muestra cómo aplicar esta teoría a la construcción de compiladores e intérpretes. Abarca el diseño de gramáticas, la construcción de traductores asistida por generadores automáticos de analizadores léxicos y sintácticos, y el análisis profundo de los problemas que surgen durante el diseño de los traductores.

La asignatura presupone que el estudiante posee conocimientos de programación, específicamente en lenguaje Java o C que son los lenguajes que se necesitan para manejar adecuadamente las herramientas de generación automática de procesadores léxicos y sintácticos empleados. Es por ello por lo que resulta especialmente importante haber cursado con anterioridad las asignaturas Fundamentos de Programación y Programación, siendo también aconsejable un grado de competencia medio-alto en Sistemas operativos y Estructuras de datos.

1b. INTRODUCTION

This topic studies the process of translating languages in general with a special focus on programming languages. The main objective is to introduce students to the theory and practice of languages processing such as scanning, parsing, type checking and code generation. It is organized as a subject of 6 ECTS, where the theory of language translation is studied and applied to the construction of compilers and interpreters. It covers the grammar design, building parsers and scanners by making use of automatic generators, and studies the problems that arise during the design of translators. The course assumes that the student has a good knowledge of programming, specifically Java or C language. It is especially important to have attended previously a course on Fundamentals of Programming, and it is also recommended a background in Operating Systems and Data Structures.

2. COMPETENCIAS

Esta asignatura, y de forma más amplia la materia de la que forma parte, desarrolla en el alumno las siguientes competencias:

Competencias de materia:

CM1) Conocer el concepto de dato como representación y medida de elementos del mundo real.

CM2) Conocer y exponer la estructura física y lógica de datos que representan números, caracteres, registros, ficheros.

CM3) Conocer los conceptos de clase, tipo abstracto de datos y objeto.

CM4) Conocer los fundamentos de la orientación a objetos y ser capaz identificar las diferencias entre la representación basada en objetos y los modelos de flujo de datos.

CM5) Desarrollar la habilidad de crear soluciones algorítmicas a problemas y ser capaz de representarla como programa u objetos.

CM6) Conocer la estrategia de implementación descendente (top-down).

CM7) Ser capaz de realizar una implementación mediante objetos.

CM8) Conocer el diseño modular y los conceptos cohesión y acoplamiento.

CM9) Alcanzar una visión de sistema de la verificación y validación.

CM10) Conocer entornos de programación, herramientas de desarrollo y entornos gráficos de desarrollo, variados.

CM11) Conocer los conceptos y técnicas de la manipulación de ficheros mediante ejemplos simples.

CM12) Conocer los conceptos de las estructuras abstractas de datos y su uso en programas y aplicaciones.

CM13) Conocer la resolución de problemas que impliquen uso de ficheros y bases de datos.

CM14) Conocer técnicas de desarrollo, diseño, prueba y depuración aplicadas a problemas.

CM15) Conocer las capacidades y limitaciones de los lenguajes de programación más comunes.

CM16) Describir la evolución de los lenguajes de programación los diferentes paradigmas disponibles hoy día y sus principales características.

CM17) Comparar y contrastar entornos de ejecución interpretados y compilados, con sus ventajas y desventajas, así como la importancia de la abstracción en el contexto de las máquinas virtuales y comprender distintas realizaciones prácticas de dicho concepto.

CM18) Introducir los conceptos de tipos, ámbito y su comprobación (compatibilidad), en el procesamiento de lenguajes de programación.

CM19) Conocer los fundamentos de la programación funcional y lógica identificando las ventajas e inconvenientes de cada paradigma.

CM20) Comprender el concepto de tipo de dato y ser capaz de identificar las características principales de un sistema de tipos.

CM21) Conocer los fundamentos de la programación distribuida, concurrente y paralela, sus algoritmos fundamentales y las ventajas e inconvenientes de cada paradigma.

Competencias genéricas:

CG1) Desarrollar aptitudes para la comunicación oral y escrita

CG2) Desarrollar capacidad de análisis y síntesis

CG3) Desarrollar capacidad para la toma de decisiones

CG4) Desarrollar métodos para la auto-organización y planificación del trabajo individual.

CG5) Motivación por la calidad

CG6) Desarrollar la capacidad de autoaprendizaje

CG7) Desarrollar la capacidad de documentarse y aprender de textos en inglés.

Competencias específicas

CESP1) Reconocer los modelos formales que sostienen la teoría del procesamiento de lenguajes (expresiones regulares, teoría de autómatas y gramáticas).

CESP2) Comparar y contrastar entornos de ejecución interpretados y compilados, con sus ventajas y desventajas, así como la importancia de la abstracción en el contexto de las máquinas virtuales y comprender distintas realizaciones prácticas de dicho concepto.

CESP3) Describir las distintas fases y algoritmos utilizados en la traducción y generación de código desde el programa fuente al ejecutable, incluidas las consideración en la traducción de código dependiente e independiente de la máquina

CESP4) Introducir los conceptos de optimización de código, incluyendo las distintas posibilidades en la elección de código intermedio y fases de optimización.

CESP5) Introducir los conceptos de tipos, ámbito y su comprobación (compatibilidad), en el procesamiento de lenguajes de programación

CESP6) Comprender el concepto de tipo de dato y ser capaz de identificar las características principales de un sistema de tipos.

Resultados del aprendizaje

Al concluir con éxito esta asignatura, los estudiantes serán capaces de…

RA1) Diseñar e implementar analizadores léxicos sencillos que produzcan testigos que puedan ser procesados por un analizador.

RA2) Diseñar la gramática de un lenguaje y utilizarla para generar un analizador sintáctico que junto con un analizador léxico verifiquen los errores estructurales del texto fuente.

RA3) Realizar un análisis sintáctico simple sin utilizar herramientas de generación automática para ello.

RA4) Utilizar herramientas de construcción automática de traductores (Flex/Bison, LEX/YACC, JLex-JFlex/CUP).

RA5) Diseñar analizadores sintácticos para lenguajes artificiales y resolver los posibles conflictos que pudieran aparecer.

RA6) Identificar los tipos de errores que se pueden presentar durante el análisis de un programa, cómo se detectan, cómo se informa de estos errores y cómo un traductor se puede recuperar de ellos.

RA7) Incorporar a un analizador sintáctico la gestión de errores y de la tabla de símbolos.

RA8) Describir e interpretar cómo se aplican los sistemas de tipos en un lenguaje de programación.

RA9) Describir los distintos tipos de modelos de control de errores, y saber aplicar los correspondientes algoritmos de control.

RA10) Construir los módulos de un compilador encargados de las fases de análisis léxico, sintáctico, semántico y de generación de código intermedio

RA11) Dominar los fundamentos de la generación de código final y las técnicas esenciales de optimización del código.

RA12) Determinar la posición que ocuparán los datos durante la ejecución de un programa y el modo de acceso a los mismos.

3. CONTENIDOS

Bloques de contenido (se pueden especificar los temas si se considera necesario)	Total de clases, créditos u horas
Introducción a los traductores	0,5 ECTS
Análisis léxico	1,5 ECTS
Análisis sintáctico	2 ECTS
Análisis semántico	1 ETTS
Generación y optimización de código	0,5 ECTS
Fase final o back-end	0,5 ECTS

4. METODOLOGÍAS DE ENSEÑANZA-APRENDIZAJE.- ACTIVIDADES FORMATIVAS

4.1. Distribución de créditos (especificar en horas)

Número de horas presenciales:	30 horas de teoría 30 horas de resolución de problemas y trabajo en laboratorio
Número de horas del trabajo	90 horas, incluyendo tutorías

propio del estudiante:	
Total horas	150 horas

4.2. Estrategias metodológicas, materiales y recursos didácticos

La asignatura se organiza como una asignatura cuatrimestral de 6 ECTS, empleándose en el proceso de enseñanza-aprendizaje de sus contenidos las siguientes actividades formativas:

- Clases Teóricas presenciales.
- Clases Prácticas presenciales.
- Prácticas en Laboratorio presenciales.
- Tutorías: individuales y/o grupales.
- Trabajos individuales.

Además, en función de la naturaleza de las distintas partes de la materia objeto de estudio, se podrán utilizar, entre otras, las siguientes actividades formativas:

- Elaboración de trabajos en equipo.
- Puesta en común de la información, problemas y dudas que aparezcan en la realización de los trabajos.
- Organización y realización de jornadas públicas con presentaciones orales y discusión de resultados.
- Utilización de la Plataforma de Aula Virtual.

Actividades presenciales:

- En el aula: Exposición y discusión de los conocimientos básicos de la asignatura. Planteamiento y resolución teórica de ejercicios y supuestos relacionados. Actividades (lecturas, discusiones, casos, etc.) orientadas a la enseñanza de las competencias específicas de la asignatura.
- En el laboratorio: Planteamiento, desarrollo y solución de ejercicios prácticos utilizando herramientas, técnicas y métodos objeto de estudio de la asignatura, contribuyendo al desarrollo de la capacidad de análisis, razonamiento crítico y comprensión de las prácticas realizadas.

Actividades no presenciales:

- Análisis y asimilación de los contenidos de la materia, resolución de problemas, consultas bibliográficas, preparación de trabajos individuales y/o grupales, realización de autoevaluaciones. Orientadas especialmente al desarrollo de métodos para la organización y planificación del trabajo individual y en equipo.
- Tutorías: asesoramiento individual y en grupos durante el proceso de enseñanza-aprendizaje, bien en forma presencial o a distancia.

Para la adquisición de las competencias prácticas, se utilizará una metodología de aprendizaje basada en una práctica evolutiva e incremental.

Recursos y materiales:

- Bibliografía de referencia.
- Ordenadores personales.
- Software de generación automática de analizadores léxicos (JLex, JFlex, Lex, etc.) y sintácticos (Yacc, CUP, etc.) y manuales de uso de los mismos.
- Entornos de desarrollo.
- Conexión a Internet.
- Plataforma de Aula Virtual y manuales de uso.
- Proyectores

5. EVALUACIÓN: Procedimientos, criterios de evaluación y de calificación

La evaluación de la asignatura programación se realizará siguiendo la Normativa Reguladora de los Procesos de Evaluación de los Aprendizajes aprobada en Consejo de Gobierno de 24 de marzo de 2011.

Tanto en la convocatoria ordinaria como en la convocatoria extraordinaria la evaluación se basará en una prueba única, compuesta de varias partes, en la que se determinará el grado de dominio de las competencias de la asignatura. El estudiante ha de atestiguar (además de superar la prueba) la adquisición de las competencias prácticas.

Procedimientos de Evaluación

Criterios de Evaluación

Los Criterios de Evaluación deben atender al grado de adquisición de las competencias por parte del estudiante. Para ello se definen los siguientes criterios:

> **CE1:** El alumno reconoce los fundamentos de la traducción automática, distingue e identifica las diferentes fases y ha asimilado los conceptos subyacentes esenciales a la misma
>
> **CE2:** El alumno es capaz de diseñar una gramática para un lenguaje artificial

CE3: El alumno demuestra aptitud en el manejo de generadores automáticos de analizadores léxicos y sintácticos

CE4: El alumno sabe cómo añadir acciones semánticas a una gramática

CE5: El alumno es capaz de diseñar estrategias de tratamiento de errores en la traducción de lenguajes

CE6: El alumno domina desde el punto de vista teórico los diferentes enfoques para el análisis sintáctico, tanto ascendente como descendente, e identifica las diferencias, ventajas e inconvenientes entre los distintos métodos existentes

CE7: El alumno es capaz de implementar y utilizar de manera óptima una tabla de símbolos.

CE8: El alumno reconoce la importancia del código intermedio y sabe aplicar estrategias de optimización al código generado por un traductor

CE9: El alumno identifica la necesidad de relacionar el código estático de un programa con las acciones a desarrollar en ejecución, y es capaz de implementar a nivel teórico las abstracciones que aparecen en el código fuente

Instrumentos de Evaluación

Esta sección indica los instrumentos de evaluación que serán aplicados a cada uno de los criterios de evaluación.

1. Prueba de Evaluación Final Teórica (PEF-T): consistente en la resolución de supuestos teórico-prácticos, asi como en la solución de problemas complejos, sobre conceptos de traducción, análisis léxico y sintáctico, análisis semántico, tablas de símbolos, comprobación de tipos, generación y optimización de código, y ambientes para la ejecución.
2. Prueba de Evaluación Final Práctica (PEF-P): consistente en la elaboración de un traductor (léxico+sintáctico+semántico) utilizando generadores automáticos, e implementando una tabla de símbolos.

Criterios de Calificación

Esta sección cuantifica los criterios de evaluación para la superación de la asignatura.

Convocatoria ordinaria y extraordinaria

Competencia		Resultado Aprendizaje	Criterio de Evaluación	Instrumento de Evaluación	Peso en la calificación
CM17-18, CM20, CG1-7	CESP 1-6	RA1-RA12	CE1-9	PEF-T	50%

CM18, CM20, CG1-7	CESP 3-6	RA1-2, RA4, RA5, RA6, RA7, RA8, RA9, RA10	CE3, CE4, CE5, CE7	PEF-P	50%

Todas las pruebas podrán realizarse en las aulas de teoría o laboratorio, o a través del Aula Virtual.

Como resultado del proceso de evaluación el alumno obtendrá una calificación que dependerá de su actividad en las distintas pruebas de la asignatura. El resultado de cada prueba arrojará información bien mediante indicadores cuantitativos de adquisición de competencias, bien mediante una calificación cualitativa, que a modo de orientación podrá determinarse en función del grado de dominio mostrado en las tareas propuestas por los profesores responsables de la asignatura:

Sobresaliente	Notable	Aprobado	Suspenso
Excelente dominio de los conocimientos básicos. Elaboración de ideas a partir de la reflexión y aplicación de los conocimientos adquiridos. Cumplimiento de todas las tareas programadas.	Dominio de los conocimientos básicos. Alto nivel de reflexión. Cumplimiento adecuado de la mayoría de las tareas programadas.	Domina los conocimientos básicos. Nivel medio de reflexión. Cumplimiento de un número suficiente de las tareas programadas.	Bajo nivel de comprensión y aplicación de ideas. Nivel bajo de reflexión. Falta de implicación en las tareas propuestas por el profesor.

6. BIBLIOGRAFÍA

Bibliografía Básica

- AHO, A.V., SETHI, R. y ULLMAN, J.D. (2008) Compiladores: Principios, técnicas y herramientas, 2ª edición. Addison–Wesley. Clásico de la materia que trata con mucha profusión y profundidad la mayoría de los temas –si bien puede alcanzarse una buena comprensión de los compiladores sin necesidad de comprender todos los detalles de cada algoritmo-.

- LOUDEN, K. C. 1997. Construcción de compiladores: Principios y práctica. Uno de los libros más claros y didácticos sobre el tema, aborda todas las fases de la traducción con la profundidad exigida a un curso de grado pero sin perder por ello la claridad y la visión didáctica. Es posiblemente el mejor libro para hacer un estudio poco dirigido por un profesor, por lo que se recomienda especialmente dada la naturaleza de las nuevas titulaciones en el EEES.

Bibliografía Complementaria

- BENNETT, J. P. (1996). Introduction to Compiling Techniques. McGraw-Hill, 1996. Texto fácil de leer y ameno de nivel introductorio. Puede ser un complemento como texto de introducción para aquellos que buscan un libro fácil y didáctico con una fuerte componente práctica al mismo tiempo.
- DOMINGUEZ, J. (2008). Compiladores: Teoría Y práctica Con Java, Jlex, Cup Y Ens2001. Lulu.com. Describe completamente el proceso de creación de un compilador en Java con las herramientas JLex, Cup y Ens2001. Es un excelente complemento para el estudio del análisis sintáctico y una imprescindible referencia para los trabajos prácticos de las Pruebas de Evaluación Continua.
- GRUNE, D y Jacobs, C.J.H. (2008). Parsing techniques: a practical guide, 2nd edition. New York: Springer. Se dedica íntegramente a una de las partes más complejas para los alumnos, el análisis sintáctico, que constituye en la práctica el primer escalón verdaderamente fuerte en su curva de aprendizaje en la asignatura.
- MAK, R. (2009). Writing Compilers and Interpreters: A Software Engineering Approach, 3rd edition. Wiley. Desarrolla muy bien las técnicas para construir un analizador, un intérprete, un depurador a nivel de código fuente y un compilador para la máquina virtual Java Virtual Machine. El libro es más para el que quiere aplicar los compiladores que para el teórico del compilador, por lo que resulta especialmente recomendable como compañero a la hora de implementar las partes más prácticas de la asignatura.
- MUCHNICK, S. (1997). Advanced Compiler Design and Implementation. Morgan Kaufmann. Cubre temas avanzados en todas las áreas fundamentales del diseño de compiladores, pero es especialmente interesante como guía de estudio en los temas relacionados con la optimización de código pues describe una amplia gama de posibles algoritmos de optimización y determina la importancia relativa de las optimizaciones. Excelente complemento para estudiar la fase de optimización.
- WIRTH, N. 1996. Compiler construction. Addison–Wesley. Presenta un ejemplo completo de compilador con el lenguaje Oberon para una arquitectura RISC lo cual resulta muy útil para el alumno cuando se enfrenta a la tarea de realizar su propia implementación pues tiene ejemplos similares de los que aprender. Interesante complemento para el estudio de la asignatura.