Plan Docente de la Asignatura

Guía Docente

Curso académico: 2011-2012 **Trimestres:** Segundo y tercero
Nombre de la asignatura: Fundamentos de la Programación
Códigos asignatura: 21406, 21297, 21595
Estudios: Grado en Ingeniería en Informática, Grado en Ingeniería Telemática y Grado en Ingeniería de Sistemas Audiovisuales
Número de créditos ECTS: 8
Número total de horas de dedicación: 200
Temporalización:

Curso: 1er curso
Tipo: Semestral
Periodo: Segundo y tercer trimestres

Profesorado: Javier Agenjo, Jesús Bisbal, Hrvoje Bogunovic, Pol Cirujeda, Eduardo Graells, Jesús Ibáñez, Vicente López, Radwan Qasrawi, Patricia Santos, David Soto, Toni Urcola, Nina Valkanova, Zacharias Vamvakousis, Chong Zhang

1. Datos descriptivos de la asignatura

- **Curso académico:** 2011-2012
- **Nombre de la asignatura:** Fundamentos de la Programación
- **Códigos:** 21406, 21297, 21595
- **Estudios:** Grado en Ingeniería en Informática, Grado en Ingeniería Telemática y Grado en Ingeniería de Sistemas Audiovisuales
- **Número de créditos ECTS:** 8
- **Número total de horas de dedicación:** 200
- **Temporalización:**
  - Curso: 1er curso
  - Tipo: Semestral
  - Periodo: Segundo y tercer trimestres
- **Coordinación:** Jesús Ibáñez y Vicente López
- **Profesorado:** Javier Agenjo, Jesús Bisbal, Hrvoje Bogunovic, Pol Cirujeda, Eduardo Graells, Jesús Ibáñez, Vicente López, Radwan Qasrawi, Patricia Santos, David Soto, Toni Urcola, Nina Valkanova, Zacharias Vamvakousis, Chong Zhang
- **Departamento:** Departamento de Tecnologías de la Información y las Comunicaciones
- **Lengua de docencia:** Catalán y castellano

2. Presentación de la asignatura

Fundamentos de la Programación está integrada en un bloque de asignaturas sobre algorítmica y programación que se llevan a cabo en el primer y segundo curso de los estudios de Grado en Ingeniería en Informática, Grado en Ingeniería Telemática y Grado en Ingeniería de Sistemas Audiovisuales. Para esta primera asignatura del bloque se presupone que los estudiantes no tienen conocimientos previos sobre algorítmica y programación. En ella se establecen las bases de la algorítmica, de las estructuras de datos y de la programación en el lenguaje C. En las asignaturas *Programación Orientada a Objetos* y *Estructuras de Datos y Algoritmos* (primer trimestre del segundo curso) se profundizará en las competencias aquí adquiridas, de forma que al final del primer trimestre de segundo curso el estudiante ha de ser capaz de desarrollar programas de tamaño considerable con las estructuras de datos adecuadas, tanto en forma imperativa como orientada a objetos. Así mismo, es importante destacar que las bases adquiridas en Fundamentos de la Programación son imprescindibles para poder implementar la parte práctica de numerosas asignaturas a lo largo de los estudios.

Las actividades de aprendizaje se han dividido fundamentalmente en función de su tipología en varias categorías:

- sesiones de teoría: los profesores presentan una serie de conceptos y técnicas, además de ejemplos de su utilización. Los estudiantes deberán repasar fuera del aula los apuntes y sus notas para acabar de asimilar los contenidos.
- sesiones de seminarios: los estudiantes deben resolver una serie de ejercicios de pequeño tamaño, poniendo en práctica los conceptos y técnicas presentados en las sesiones de teoría. Estas actividades se comienzan en aulas de ordenadores, y deben completarse fuera del aula.
- sesiones de prácticas: los estudiantes deben resolver unos problemas de un tamaño mayor que los ejercicios antes citados, de forma que deben decidir qué conceptos y técnicas deben utilizar en cada caso. Estas actividades se comienzan en aulas de ordenadores, y deben completarse fuera del aula. Además, en las sesiones de prácticas los estudiantes aprenden a utilizar las herramientas de programación.
- ejercicios de autoevaluación: ejercicios para que el estudiante, al final de cada unidad didáctica, compruebe que ha asimilado los conceptos y técnicas presentados. Esta actividad la realiza el estudiante fuera del aula.

3. Prerrequisitos para el seguimiento del itinerario formativo

Esta asignatura no requiere ningún conocimiento previo de programación ni algorítmica. Para la realización de algunos ejercicios sí se requieren los conocimientos de matemáticas de nivel de bachillerato.

4. Competencias de la asignatura

El objetivo fundamental de esta asignatura es que los alumnos adquieran las bases de la algorítmica y estructuras de datos, así como que sean capaces de desarrollar fluidamente programas de tamaño medio utilizando el lenguaje C.

En este capítulo se detalla qué es lo que se espera que los estudiantes hayan aprendido al acabar la asignatura. En primer lugar, las competencias generales se refieren a habilidades no directamente relacionadas con la programación en sí, sino al ámbito profesional de un ingeniero. Las competencias específicas son las referidas a aspectos propios de la asignatura.

4.1. Competencias generales

Instrumentales

CG1: Capacidad de síntesis
El estudiante ha de ser capaz de escribir soluciones con los elementos esenciales, de forma simple, elegante y lo más eficiente posible.

CG2: Capacidad de análisis
El estudiante ha de ser capaz de, a partir de un problema concreto, analizarlo y proponer soluciones adecuadas a dicho problema.

Sistémicas

CG3: Capacidad para aplicar el conocimiento en la práctica
El estudiante ha de ser capaz de aplicar los conocimientos adquiridos para resolver problemas concretos, eligiendo la técnica que mejor se ajuste a cada caso.

CG4: Interés por la calidad
El estudiante ha de ser capaz de que su código sea, además de eficiente, fácil de leer y mantener. Así mismo es importante que se documente correctamente, tanto dentro del propio código como en una memoria.

4.2. Competencias específicas

CE1: Capacidad para trabajar con herramientas de programación
El estudiante ha de ser capaz de trabajar con las herramientas básicas para la programación: un compilador y un debugador. Además ha de ser capaz de trabajar con un editor de programación y un entorno de desarrollo integrado. Esta competencia es primordial para el correcto desarrollo de las demás.

CE2: Dominio de los tipos de datos estáticos básicos y compuestos
El estudiante ha de ser capaz de distinguir los diferentes tipos de datos estáticos básicos y compuestos y de decidir el tipo adecuado para cada situación concreta.

CE3: Dominio de las estructuras de control
El estudiante ha de ser capaz de distinguir las diferentes estructuras de control compuestos y de decidir las más adecuadas para resolver problemas concretos.

CE4: Capacidad de resolución de problemas mediante diseño descendente y dominio de la utilización de funciones y librerías
El estudiante ha de ser capaz de resolver problemas de una complejidad considerable utilizando las técnicas del diseño descendente. En particular, el estudiante ha de comprender el funcionamiento de las llamadas a funciones y pasos de parámetros; ha de dominar el uso y creación de librerías; y ha ser capaz de dividir un problema en las unidades adecuadas.

CE5: Dominio de los tipos de datos dinámicos y de la gestión dinámica de memoria
El estudiante ha de comprender el mecanismo de gestión de memoria, así como el uso de punteros y control dinámico de estructuras de datos. Se incluye también el manejo de ficheros de texto.

CE6: Documentación y estructuración de código
El estudiante ha de adquirir el hábito de estructurar y documentar el código de forma adecuada con la finalidad de facilitar su posterior lectura.

CE7: Capacidad de lectura (rápida) de código en C
El estudiante ha de ser capaz de comprender código escrito por otros programadores, de forma relativamente rápida.

CE8: Dominio de los elementos fundamentales de algoritmia
El estudiante ha de conocer y ser capaz de aplicar de forma adecuada los conceptos fundamentales de la algoritmia, tales como la recursividad y los algoritmos de búsqueda y ordenación.

5. Contenidos

Bloque 1: Introducción y conceptos generales

Conceptos	Procedimientos
- Breve historia de la programación y sus lenguajes y paradigmas - Mecanismos de compilación e interpretación - Diferencia entre programa y algoritmo	

Bloque 2: Tipos de datos básicos

Conceptos	Procedimientos
- Variables y constantes - Los diferentes tipos de datos básicos: o Tipos numéricos o Caracteres o Boléanos	- Declaración de constantes y variables de los diferentes tipos

Bloque 3: Expresiones, sentencias y estructuras de control

Conceptos	Procedimientos
- Formación de expresiones - Introducción a la lógica booleana - Sentencias o instrucciones: o Asignaciones o Operaciones de entrada y salida o El orden de precedencia de los operadores - Las estructuras de control: o Las estructuras condicionales o Las estructuras iterativas	- Evaluación de expresiones - Resolución de pequeños problemas utilizando las sentencias y estructuras de control adecuadas

Bloque 4: La descomposición funcional y el diseño descendente

Conceptos	Procedimientos
- Diseño descendente - Declaración de funciones - Definición de parámetros - Llamadas a funciones y paso de parámetros - El tipo void	- Descomposición de problemas en subproblemas - Resolución de pequeños problemas definiendo funciones de forma adecuada

Bloque 5: Tipos de datos compuestos estáticos

Conceptos	Procedimientos
- Arrays unidimensionales - Arrays multidimensionales - Strings o cadenas de caracteres - Estructuras (structs)	- Resolución de pequeños problemas sobre las operaciones típicas con cada uno de los tipos de datos

Bloque 6: Declaración de tipos propios

Conceptos	Procedimientos
- Los tipos de las estructuras - Definición de tipos mediante typedef - Conversiones de tipos	- Resolución de pequeños problemas de definición de tipos de datos propios

Bloque 7: Los punteros y la gestión dinámica de memoria

Conceptos	Procedimientos
- Declaración de punteros - Operaciones de dirección e indirección - Asignación de valores a punteros - El puntero nulo - Arrays y punteros en C	- Resolución de pequeños problemas sobre las operaciones típicas con punteros - Resolución de pequeños problemas típicos de arrays utilizando punteros

Bloque 8: La descomposición funcional y el diseño descendente (segunda parte)

Conceptos	Procedimientos
- Paso de parámetros por referencia - La función main y sus argumentos - Visibilidad y alcance - Las librerías y el pre-procesador	- Resolución de problemas mediante descomposición utilizando funciones con parámetros por referencia

Bloque 9: Ficheros de texto

Conceptos	Procedimientos

- El tipo fichero (FILE)
- Operaciones con ficheros:
  o Abrir
  o Cerrar
  o Escribir
  o Leer
- Entrada y salida estándar
	- Resolución de pequeños problemas sobre las operaciones típicas con ficheros de texo

Bloque 10: Estilo y buenas prácticas

Conceptos	Procedimientos
- Estilo, legibilidad y ofuscación - Buenas prácticas - Errores comunes - Análisis de código - Depuración	- Resolución de problemas de detección de errores y mejora de estilo. - Uso de herramientas de análisis de código y depuración.

Bloque 11: La descomposición funcional y el diseño descendente (tercera parte)

Conceptos	Procedimientos
- Programación de librerías - Segmentación del código en ficheros - Ficheros de cabecera	- Resolución de problemas de creación y uso de librerías.

Bloque 12: Algoritmos de búsqueda y ordenación

Conceptos	Procedimientos
- Búsqueda lineal y binaria - Algoritmos básicos de ordenación o Burbuja o Inserción o Selección	- Resolución de problemas sobre algoritmos de búsqueda y ordenación

Bloque 13: Recursividad

Conceptos	Procedimientos
- Concepto de recursividad - Algoritmos recursivos - Caso base y recurrencia - Ventajas e inconvenientes - Recursividad directa e indirecta - Transformación a algoritmos iterativos - Ejemplos concretos de algoritmos recursivos (fractales, torres de Hanoi, Quicksort, etc)	- Resolución de problemas mediante definición de funciones recursivas

Bloque 14: Gestión avanzada de memoria

Conceptos	Procedimientos
- Punteros a punteros - Arrays de punteros - Punteros a funciones - Redimensionamiento de memoria - Ejemplos concretos de gestión avanzada de memoria (listas, árboles binarios, etc)	- Resolución de problemas de gestión avanzada de memoria

6. Metodología de aprendizaje

El proceso habitual de aprendizaje en una unidad didáctica comienza con una sesión de teoría en la que se presentan ciertos fundamentos teórico-prácticos. Esta actividad se realiza en un gran grupo de estudiantes. El estudiante deberá posteriormente completar esta actividad con una lectura detenida de los apuntes. Por ejemplo, una sesión típica de teoría de 2 horas de duración, convenientemente aprovechadas, necesitará un trabajo personal fuera del aula de 1 hora.

A continuación se lleva a cabo una o varias sesiones de seminarios y/o prácticas. En las sesiones de seminarios el estudiante pone en práctica los conceptos y técnicas presentados en la sesión de teoría, mediante la implementación de programas para solucionar pequeños problemas. El objetivo es que el estudiante consolide los fundamentos para posteriormente poder llevar a cabo problemas de mayor tamaño. Esta actividad se realiza individualmente,

en un grupo de alrededor de 15 estudiantes. Cada actividad de este tipo está programada con una duración de 4 horas, de las cuales 2 se realizan con soporte del profesor. Los primeros ejercicios de la sesión, para los cuales se ofrecen las soluciones, deberán resolverse antes de llegar al aula. El profesor puede solicitar la entrega de uno de los ejercicios planteados al final de cada sesión.

En las sesiones de prácticas se proponen unos problemas de mayor tamaño, especialmente en las 3 prácticas evaluables, que requieren un diseño previo de la solución a implementar y que integran diferentes conceptos y técnicas. En la última práctica se reúnen todas las competencias específicas que el estudiante debe adquirir en esta asignatura. Cada actividad de este tipo se realiza por parejas, en un grupo de alrededor de 30 estudiantes, y debe continuarse fuera del aula.

En cada sesión de seminarios y de prácticas se dedica un tiempo para discutir los principales problemas que se han presentado en la sesión anterior.

El último paso de la unidad didáctica es el de la resolución de unos ejercicios de autoevaluación mediante los cuales el estudiante puede comprobar si ha adquirido las competencias que posteriormente se evaluarán mediante los exámenes parcial y final.

6.1. Unidades didácticas

Unidad didáctica 1: Primeros pasos: introducción, tipos de datos básicos y estructuras de control

Bloques de contenido	Actividades de aprendizaje			
1: Introducción y conceptos generales 2: Tipos de datos básicos 3: Expresiones, sentencias y estructuras de control	Sesiones de Teoría	Sesiones de Seminarios	Sesiones de Prácticas	Autoevaluación
	T1, T2 y T3	S1	P1	A1

Dedicación total Unidad Didáctica 1: 17 horas (10 en el aula, 7 fuera)

Detalle de las actividades:

Sesión de teoría T1. 3 horas (2 en el aula, 1 fuera): breve historia de la programación y explicando los modelos de compilación e interpretación.

Sesiones de teoría T2 y T3. 6 horas (por cada sesión 2 en el aula, 1 fuera): explicación de los tipos de datos básicos y principales ejemplos de utilización. Explicación de la formación de expresiones y sentencias básicas con los principales ejemplos de su utilización. Estructuras de control condicionales e iterativas y principales ejemplos de su utilización.

Sesión de seminarios S1. 4 horas (2 en el aula, 2 fuera): ejercicios sobre formación de expresiones y estructuras de control condicionales e iterativas.

Sesión de prácticas P1. 3 horas (2 en el aula, 1 fuera): se explicará cómo instalar el compilador, y cómo editar, compilar y ejecutar un programa. Se dará a los alumnos pequeños programas con errores para que se habitúen a los típicos mensajes de error del compilador.

Autoevaluación A1. 1 hora (fuera del aula): resolución de ejercicios de autoevaluación sobre formación de expresiones y estructuras de control.

Unidad didáctica 2: Funciones y diseño descendente

Bloques de contenido	Actividades de aprendizaje			
4: La descomposición funcional y el diseño descendente	Sesiones de Teoría	Sesiones de Seminarios	Sesiones de Prácticas	Autoevaluación
	T4	S2		A2

Dedicación total Unidad Didáctica 2: 8 horas (4 en el aula, 4 fuera)

Detalle de las actividades:

Sesión de teoría T4. 3 horas (2 en el aula, 1 fuera): explicación de las bases del diseño descendente así como la definición de funciones y el mecanismo de llamada y paso de parámetros por valor, con ejemplos de utilización.

Sesión de seminarios S2. 4 horas (2 en el aula, 2 fuera): ejercicios sobre diseño descendente y paso de parámetros por valor.

Autoevaluación A2. 1 hora (fuera del aula): Resolución de ejercicios de autoevaluación sobre funciones y paso de parámetros por valor.

Unidad didáctica 3: Tipos de datos compuestos estáticos y declaración de tipos

Bloques de contenido	Actividades de aprendizaje			
5: Tipos de datos compuestos estáticos	Sesiones de Teoría	Sesiones de Seminarios	Sesiones de Prácticas	Autoevaluación
6: Declaración de tipos propios	T5 y T6	S3	P2 y P3	A3

Dedicación total Unidad Didáctica 3: 29 horas (10 en el aula, 19 fuera)

Detalle de las actividades:

Sesiones de teoría T5 y T6. 6 horas (por cada sesión 2 en el aula, 1 fuera): explicación de los tipos de datos compuestos (arrays, cadenas y estructuras) con ejemplos de utilización. Explicación de la definición de tipos de datos propios por parte del programador, con ejemplos de utilización.

Sesión de seminarios S3. 4 horas (2 en el aula, 2 fuera): ejercicios sobre tipos de dato compuestos.

Sesiones de prácticas P2 y P3 (primera práctica evaluable). 18 horas (por cada sesión, 2 en el aula, 7 fuera): el estudiante ha de resolver (programando en C) unos problemas medianos, utilizando arrays, cadenas y estructuras, y en algunos casos definiendo sus propios tipos. Incluye la confección de una memoria explicativa del trabajo.

Autoevaluación A3. 1 hora (fuera del aula): Resolución de ejercicios de autoevaluación sobre tipos de datos compuestos.

Unidad didáctica 4: Punteros y paso de parámetros por referencia

Bloques de contenido	Actividades de aprendizaje			
7: Los punteros y la gestión dinámica de memoria	Sesiones de Teoría	Sesiones de Seminarios	Sesiones de Prácticas	Autoevaluación
8: La descomposición funcional y el diseño descendente (segunda parte)	T7 y T8	S4	P4	A4

Dedicación total Unidad Didáctica 4: 18 horas (8 en el aula, 10 fuera)

Detalle de las actividades:

Sesiones de teoría T7 y T8. 8 horas (por cada sesión 2 en el aula, 2 fuera): explicación del funcionamiento de los punteros con ejemplos de utilización. Explicación del paso de parámetros por referencia con ejemplos de utilización. Explicación de las reglas de visibilidad.

Sesión de seminarios S4. 4 horas (2 en el aula, 2 fuera): ejercicios sobre punteros y paso de parámetros por referencia.

Sesión de prácticas P4. 5 horas (2 en el aula, 3 fuera): el estudiante ha de resolver (programando en C) unos pequeños problemas, utilizando punteros y paso de parámetros por referencia.

Autoevaluación A4. 1 hora (fuera del aula): Resolución de ejercicios de autoevaluación sobre punteros y paso de parámetros por referencia.

Unidad didáctica 5: Ficheros de texto

Bloques de contenido	Actividades de aprendizaje			
9: Ficheros de texto	Sesiones de Teoría	Sesiones de Seminarios	Sesiones de Prácticas	Autoevaluación
	T9	S5		A5

Dedicación total Unidad Didáctica 5: 8 horas (4 en el aula, 4 fuera)
Detalle de las actividades:

Sesión de teoría T9. 3 horas (2 en el aula, 1 fuera): explicación de los ficheros de texto y ejemplos de utilización.

Sesión de seminarios S5. 4 horas (2 en el aula, 2 fuera): Ejercicios sobre ficheros de texto.

Autoevaluación A5. 1 hora (fuera del aula): Resolución de ejercicios de autoevaluación sobre ficheros.

Unidad didáctica 6: Buenas prácticas y programación de librerías

Bloques de contenido	Actividades de aprendizaje			
10: Estilo y buenas prácticas 11: La descomposición funcional y el diseño descendente (tercera parte)	Sesiones de Teoría	Sesiones de Seminarios	Sesiones de Prácticas	Autoevaluación
	T10 y T11		P5, P6, P7 y P8	A6

Dedicación total Unidad Didáctica 6: 33 horas (12 en el aula, 21 fuera)
Detalle de las actividades:

Sesión de teoría T10. 3 horas (2 en el aula, 1 fuera): explicación de buenas prácticas para programar de forma legible y obtener programas menos propensos a errores. Explicación de los errores comunes al codificar un programa. Explicación de las técnicas de análisis y depuración de código.

Sesión de teoría T11. 3 horas (2 en el aula, 1 fuera): explicación del proceso de creación de librerías y segmentación del código en varios ficheros. Repaso de las reglas de visibilidad.

Sesión de prácticas P5. 4 horas (2 en el aula, 2 fuera): se comenzará a usar un entorno de desarrollo integrado y se explicará cómo debugar un programa utilizando el entorno de desarrollo. Los estudiantes debugarán varios programas.

Sesión de prácticas P6. 4 horas (2 en el aula, 2 fuera): se explicará como crear un programa compuesto de varios módulos (o ficheros) utilizando un entorno de desarrollo integrado. Los estudiantes crearán un programa dividido en varios módulos.

Sesiones de prácticas P7 y P8 (segunda práctica evaluable). 18 horas (por cada sesión, 2 en el aula, 7 fuera): el estudiante ha de resolver (programando en C) un problema de tamaño considerable que integre punteros, paso de parámetros por referencia, ficheros de texto y segmentación del código en varios ficheros. La solución debe programarse siguiendo las recomendaciones de buenas prácticas. Incluye la confección de una memoria explicativa del trabajo.

Autoevaluación A6. 1 hora (fuera del aula): Resolución de ejercicios de autoevaluación sobre estilo, buenas prácticas, y segmentación del código en módulos o ficheros.

Unidad didáctica 7: Búsqueda, ordenación y recursividad

Bloques de contenido	Actividades de aprendizaje			
12: Algoritmos de búsqueda y ordenación 13: Recursividad	Sesiones de Teoría	Sesiones de Seminarios	Sesiones de Prácticas	Autoevaluación
	T12, T13, T14 y T15	S6 y S7		A7

Dedicación total Unidad Didáctica 7: 24 horas 12 en el aula, 12 fuera)
Detalle de las actividades:

Sesión de teoría T12. 3 horas (2 en el aula, 1 fuera): explicación de los algoritmos de búsqueda lineal y binaria, así como de los algoritmos básicos de ordenación (burbuja, inserción y selección).
Sesiones de teoría T13, T14 y T15. 12 horas (T13 2 en el aula, 2 fuera; T14 2 en el aula, 3 fuera; T15 2 en el aula y 1 fuera): explicación del concepto de recursividad. Explicación de las bases de los algoritmos recursivos y sus ventajas e inconvenientes. Explicación de los diferentes tipos de recursividad. Explicación de la transformación de algoritmos recursivos a algoritmos iterativos. Discusión de diversos ejemplos concretos de algoritmos recursivos (incluyendo los de ordenación avanzada).
Sesión de seminarios S6. 4 horas (2 en el aula, 2 fuera): Ejercicios sobre algoritmos de búsqueda, algoritmos de ordenación y algoritmos recursivos.
Sesión de seminarios S7. 4 horas (2 en el aula, 2 fuera): Ejercicios sobre algoritmos recursivos.
Autoevaluación A7. 1 hora (fuera del aula): Resolución de ejercicios de autoevaluación sobre algoritmos de búsqueda, algoritmos de ordenación y recursividad.

Unidad didáctica 8: Gestión avanzada de memoria

Bloques de contenido	Actividades de aprendizaje			
14: Gestión avanzada de memoria	Sesiones de Teoría	Sesiones de Seminarios	Sesiones de Prácticas	Autoevaluación
	T16, T17 y T18	S8	P9 y P10	A8

Dedicación total Unidad Didáctica 8: 33 horas (12 en el aula, 21 fuera)
Detalle de las actividades:

Sesiones de teoría T16 y T17. 7 horas (T16 2 en el aula, 2 fuera; T17 2 en el aula, 1 fuera): explicación de los mecanismos y técnicas para la gestión avanzada de memoria (punteros a punteros, arrays de punteros, punteros a funciones, redimensionamiento de memoria). Discusión de ejemplos concretos que ilustran la gestión avanzada de memoria.
Sesión de teoría T18. 3 horas (2 en el aula, 1 fuera): repaso general y resolución de dudas.
Sesión de seminarios S8. 4 horas (2 en el aula, 2 fuera): Ejercicios sobre gestión avanzada de memoria.
Sesiones de prácticas P9 y P10 (tercera práctica evaluable). 18 horas (por cada sesión, 2 en el aula, 7 fuera): el estudiante ha de resolver (programando en C) un problema de tamaño considerable que integre todos los conceptos y técnicas de la asignatura (incluyendo algoritmos recursivos y gestión avanzada de memoria). La solución debe programarse siguiendo las recomendaciones de buenas prácticas. Incluye la confección de una memoria explicativa del trabajo.
Autoevaluación A8. 1 hora (fuera del aula): Resolución de ejercicios de autoevaluación sobre gestión avanzada de memoria.

Nota: además de las horas de trabajo previstas e indicadas en la descripción de las unidades didácticas, se reservan 10 horas (8.5+1.5) para preparar y realizar el examen parcial, y 20 (17.5+2.5) horas para el examen final.

Dedicación total de la asignatura: 200 horas (76 en el aula, 124 fuera)

7. Programación de actividades

7.2. Horas de dedicación de los alumnos

	Actividad	Horas en el aula			Horas fuera del aula	
		Grupo grande	Grupo mediano	Grupo pequeño		
Unidad didáctica 1	T1	2			1	
	T2	2			1	
	P1		2		1	
	T3	2			1	
	S1			2	2	
	A1				1	
Unidad didáctica 2	T4	2			1	
	S2			2	2	
	A2				1	
Unidad didáctica 3	T5	2			1	
	S3			2	2	
	T6	2			1	
	P2		2		7	
	P3		2		7	
	A3				1	
Unidad didáctica 4	T7	2			2	
	S4			2	2	
	T8	2			2	
	P4		2		3	
	A4				1	
Unidad didáctica 5	T9	2			1	
	S5			2	2	
	A5				1	
Unidad didáctica 6	T10	2			1	
	P5		2		2	
	T11	2			1	
	P6		2		2	
	P7		2		7	
	P8		2		7	
	A6				1	
Unidad didáctica 7	T12	2			1	
	T13	2			2	
	T14	2			3	
	S6			2	2	
	T15	2			1	
	S7			2	2	
	A7				1	
Unidad didáctica 8	T16	2			2	
	T17	2			1	
	S8			2	2	
	P9		2		7	
	T18	2			1	
	P10		2		7	
	A8				1	
Exámenes	Parcial	1,5			8,5	
	Final	2,5			17,5	
Total		40	20	16	124	200 (ECTS * 25)

7.3. Programación de sesiones presenciales

En esta sección se incluyen tablas con la programación de las sesiones presenciales. En las tablas, las sesiones T son de teoría, S son de seminario, y P son de prácticas. Nótese que, para cada grupo, la mitad de sus subgrupos de seminario tiene las sesiones de seminario un día, y la otra mitad las tiene otro día diferente. En cambio, todos los subgrupos de prácticas de cada grupo tienen las sesiones de prácticas el mismo día y a la misma hora, aunque cada subgrupo las tiene en un aula diferente. Para ver las aulas de cada sesión, consúltese **http://www.upf.edu/esup/docencia/graus/**

7.4. Listado de actividades (con fecha de entrega)

Nombre	Bloques de contenido	Fecha de enunciado	Fecha límite de entrega	Competencias trabajadas y evaluadas
Primera práctica evaluable	B1, B2, B3, B4, B5 y B6	03-02-2012	27-02-2012 (a las 9:00 am)	CE2, CE3, CE4, CE6
Segunda práctica evaluable	B7, B8, B9, B10 y B11	20-04-2012	14-05-2012 (a las 9:00 am)	CE2, CE3, CE4, CE5 y CE6
Tercera práctica evaluable	B12, B13, B14	25-05-2012	18-06-2012 (a las 9:00 am)	CE2, CE3, CE4, CE5, CE6 y CE8

Dado el carácter incremental de la asignatura, aunque en cada práctica evaluable se trabajan principalmente los bloques de contenido que se indican en la tabla, también se trabajan indirectamente los bloques anteriores (al ser necesarios para la resolución de los problemas).

Aparte de estas prácticas evaluables, en las sesiones de seminarios los profesores pedirán la entrega voluntaria de un ejercicio al final de cada sesión.

Evaluación

7.1. Criterios de evaluación generales

Esta asignatura consta de dos partes principales:

- Fundamentos teórico-prácticos, que engloba las actividades relacionadas con las sesiones de teoría y de seminarios
- Prácticas

Cada una supone un 50% de la nota final, aunque es necesario aprobar ambas para superar la asignatura.

Fundamentos teórico-prácticos

Habrá dos exámenes, uno a mitad de la asignatura (al final del segundo trimestre) y otro al final (al final del tercer trimestre). El primero supone un 25% de la nota y se centra en el bloque de ejercicios realizados en las sesiones de ejercicios del segundo trimestre. El segundo examen supone el 75% restante y abarca cuestiones teórico-prácticas de todo el conjunto de la asignatura.

En las sesiones de ejercicios los profesores irán revisando el trabajo de los estudiantes a lo largo del trimestre, y podrán pedir la entrega de un ejercicio al final de cada sesión. En función de dichas revisiones y entregas, se podrá subir (pero nunca bajar) la nota de la parte de fundamentos teórico-prácticos hasta 1.5 puntos.

Además, se han programado varias actividades de auto-evaluación, una al final de cada unidad didáctica, con el objetivo de que el estudiante vaya valorando su propio progreso. Estas actividades constan de una serie de preguntas test de respuesta múltiple, de un nivel similar a las de los exámenes. Así mismo, en las actividades de seminarios el estudiante puede también evaluar su progreso, comparando sus soluciones con los resultados publicados. En todo caso, estas actividades de auto-evaluación no inciden en la nota.

Prácticas

La nota de esta parte se obtiene a partir de tres prácticas entregables, distribuidas a lo largo de la asignatura. Cada una de estás prácticas tiene el mismo peso en la nota (es decir, cada práctica evaluable tiene un peso de un 33.3%). En cuanto a la nota de las prácticas entregables, a parte de la valoración global, los profesores podrán revisar el progreso en el trabajo de los estudiantes, a lo largo de las sesiones de prácticas. En el caso de que algún grupo no hiciese dicha revisión, o que ésta no fuese satisfactoria, tendrá que hacer una defensa de su práctica con posterioridad a su entrega.

Aparte de las prácticas entregables, las otras prácticas servirán a los estudiantes para evaluar su propio progreso. Estas actividades no inciden en la nota.

Recuperación de Septiembre

En el caso de no aprobar ambas partes en la convocatoria de Junio, sólo se guardará para septiembre una parte aprobada en el caso de que la nota de la otra parte sea igual o superior al 3. En caso contrario, habrá que volver a examinarse y entregar una nueva práctica (con un enunciado diferente).

7.2. Concreción

Primera práctica evaluable

Se evaluará la resolución de una práctica en la que el estudiante podrá mostrar el grado de asimilación de los conceptos y competencias adquiridos durante el primer tercio de la asignatura. En particular se evaluarán las competencias CE2, CE3 y ciertos aspectos de la CE4 y la CE6 (estructuración de código comentado). La práctica consistirá en la resolución de un problema, incluyendo su análisis y la programación de una solución adecuada en lenguaje C. El estudiante deberá decidir las estructuras de datos estáticas así como el flujo de operaciones de la solución y realizar un sencillo diseño descendente. Así mismo se valorarán las cuatro competencias generales definidas.

Segunda práctica evaluable

Se evaluará la resolución de una práctica en la que el estudiante podrá mostrar el grado de asimilación de los conceptos y competencias adquiridos durante los dos primeros tercios de la asignatura, y especialmente durante el segundo. En particular se evaluarán las competencias CE2, CE3, CE4, CE5 y CE6. La práctica consistirá en la resolución de un problema considerablemente complejo, incluyendo su análisis, la elección de las estructuras de datos (estáticas y dinámicas) adecuadas, el diseño descendente, la programación de una solución adecuada en lenguaje C (siguiendo las recomendaciones de buenas prácticas) y su correcta documentación en una memoria en la que también se justificarán las decisiones adoptadas. Así mismo se valorarán las cuatro competencias generales definidas.

Tercera práctica evaluable

Se evaluará la resolución de una práctica en la que el estudiante podrá mostrar el grado de asimilación de los conceptos y competencias adquiridos durante toda la asignatura. En particular se evaluarán las competencias CE2, CE3, CE4, CE5, CE6 y CE8. La práctica consistirá en la resolución de un problema considerablemente complejo, incluyendo su análisis, la elección de las estructuras de datos (estáticas y dinámicas) adecuadas, el diseño descendente, la programación de una solución adecuada en lenguaje C (siguiendo las recomendaciones de buenas prácticas) y su correcta documentación en una memoria en la que también se justificarán las decisiones adoptadas y donde se incluirá una parte de análisis de la complejidad algorítmica. Así mismo se valorarán las cuatro competencias generales definidas.

Examen parcial

Se evaluará la comprensión y aplicación de los conceptos y técnicas adquiridos durante la primera mitad de la asignatura (correspondiente al segundo trimestre). En concreto se evaluarán las competencias CE2, CE3, CE4, CE5 y especialmente la CE7. El método de evaluación consiste en un examen tipo test de alrededor de 20 preguntas con 4 opciones por cada pregunta con sólo una respuesta válida. Las preguntas estarán extraídas (con ligeras modificaciones) de las propuestas para las sesiones de ejercicios y de las actividades de auto-evaluación. La evaluación tendrá lugar durante el período de exámenes del segundo trimestre.

Examen final

Se evaluará la comprensión y aplicación de los conceptos adquiridos a lo largo de toda la asignatura en pequeños programas. En concreto se evaluarán las competencias CE2, CE3, CE4, CE5, CE7 y CE8. El método de evaluación consiste en un examen tipo test de alrededor de 30 preguntas con 4 opciones por cada pregunta con sólo una respuesta válida. La evaluación tendrá lugar durante el período de exámenes del tercer trimestre.

8. Fuentes de información y recursos didácticos

8.1 Fuentes de información para el aprendizaje. Bibliografía básica

- Toni Navarrete Terrasa. *Introducción a la programación con lenguaje C*.
- Jesús Bisbal Riera. *Manual d'algorísmica: Recursivitat, complexitat i disseny d'algorismes*. Editorial UOC. ISBN: 978-84-9788-570-6
- Brian W. Kernighan, Dennis M. Ritchie: *El lenguaje de programación C*. Segunda edición. Prentice-Hall. ISBN: 968-880-205-0

8.2 Fuentes de información para el aprendizaje. Bibliografía complementaria

Otros libros sobre C:

- Herbert Schildt: *C Manual de referencia*. Mc Graw Hill. 84-481-0335-1
- James L. Antonakos, Kenneth C. Mansfield Jr.: *Programación estructurada en C*. Prentice-Hall. ISBN: 84-89660-23-9
- Marco A. Peña, José M. Cela: *Introducción a la programación en C*. Edicions UPF. ISBN: 84-8301-429-7
- Luis Joyanes, Ignacio Zahonero: *Programación en C*. Mc Graw Hill. ISBN: 84-481-3013-8
- Félix García, Jesús Carretero, Javier Fernández, Alejandro Calderón: *El lenguaje de programación C. Diseño e implementación de programas*. Prentice-Hall. ISBN: 84-205-3178-2
- P.J. Plauger, Jim Brodie: *C Estándar. Guía de referencia rápida para programadores*. Anaya. ISBN: 84-7614-264-1

Otros libros "clásicos" sobre algoritmia (más avanzados):

- Alfred V. Aho, Jeffrey D. Ullman, John E. Hopcroft: *Estructuras de datos y algoritmos*. Addison Wesley, 1988. ISBN: 968-444-345-5
- Niklaus Wirth: *Algoritmos + estructuras de datos = programas*. Ediciones del Castillo, 1980. ISBN: 84-219-0172-9
- Niklaus Wirth: *Algoritmia y estructuras de datos*. Prentice Hall, 1987. ISBN: 968-880-113-5
- D.E. Knuth: *El arte de programar ordenadores* (3 volums). Editoria Reverté. ISBN: 84-291-2661-9
- Terrence W. Pratt, Marvin V. Zelkowitz: *Lenguajes de programación. Diseño e implementación*. Prentice Hall, 3ª edic., 1997. ISBN: 970-17-0046-5
- G. Brassard, P. Bratley: *Fundamentos de algoritmia*. Prentice-Hall. ISBN: 84-89660-00-X

8.3 Recursos didácticos. Material docente de la asignatura

En el aula Moodle de la asignatura (en el Aula Global) se colgará el material docente de la asignatura. En particular se colgarán:

- Apuntes
- Enunciados para las sesiones de seminario
- Enunciados para las sesiones de prácticas
- Ejercicios de autoevaluación