Modelo de Plan Docente de una materia

I. Descripción y contextualización

Identificación y características de la materia				
Denominación y código	**Laboratorio de Programación II - A4**			
Curso y Titulación	**2º de Ingeniería Informática**			
Área	Lenguajes y Sistemas Informáticos			
Departamento	*Departamento de Ingeniería de Sistemas Informáticos y Telemáticos*			
Tipo	OB			
Coeficientes	Practicidad: 5 alto		Agrupamiento: 2 bajo	
Duración ECTS (créditos)	**2ºC**		**5,45 ECTS (136,36 horas)**	
Distribución ECTS (rangos)	Grupo Grande: 10 %	Seminario-Lab.: 30 %	Tutoría ECTS: 5 %	No presenciales: 55 %
	13 horas	40-41 horas	6-7 horas	75 horas
Descriptores (según BOE)	RD 1459/1990 - I.I. RD 1460 / 1990 - I.T.I.S. RD 1461 / 1990 - I.T.I.G. Denominación: Laboratorio de Programación II Contenidos: Diseño, mantenimiento y desarrollo de programas. Técnicas de verificación y prueba de programas. Créditos: 6 Áreas de Conocimiento: CCIA / LSI			
Coordinador-Profesor/es	Andrés Caro Lindo, José Mª Conejero Manzano, Roberto Rodríguez Echeverría, Encarna Sosa Sánchez			
Tutorías complementarias (1)	Andrés Caro Lindo: Despacho 13 (Ed. Informática) – Martes y Jueves de 9:30 a 12:30 José Mª Conejero Manzano: Despacho 20 (Ed. Telecomunicaciones) – Martes y Miércoles de 11:30 a 13:30 Roberto Rodríguez Echeverría: Despacho 14 (Ed. Telecomunicaciones) – Lunes y Jueves de 11:30 a 14:30 Encarna Sosa Sánchez: Despacho 10 (Ed. Informática) – Martes de 10:00 a 14:00 y Jueves de 18:00 a 20:00 (*) Estos horarios están sujetos a modificaciones en los distintos cursos académicos. Consultar la web del centro: http://epcc.unex.es			

Contextualización profesional

Dentro de los subperfiles definidos en la titulación de Ingeniería Informática existen unos contenidos formativos comunes a la titulación. Dentro de estos contenidos se encuentra las subcategorías Programación e Ingeniería del Software, Sistemas de Información y Sistemas Inteligentes, en cuyo estudio se centra principalmente esta materia. En el libro blanco estas subcategorías deben tratar:

- Fundamentos y metodología de la programación
- Algoritmia
- Computabilidad
- Lenguajes de Programación
- Paradigmas de Programación
- Estructuras de datos
- Desarrollo de software:
  - Procesos
  - Requisitos
  - Especificación y Diseño
- Gestión de Proyectos
- Calidad del Software
- Interacción Persona-Computadora
- Bases de Datos
- Inteligencia Artificial

Son competencias fundamentales de esta asignatura:

- Lenguajes de Programación
- Paradigmas de Programación
- Desarrollo de software (Procesos, Requisitos, Especificación y Diseño)
- Gestión de Proyectos
- Interacción Persona-Computadora

Esta materia tiene como objetivos fundamentales que el alumno consolide sus conocimientos sobre metodología de desarrollo de programas empleando la Orientación a Objetos y sea capaz de participar en todas las fases de los proyectos de desarrollo de software en entornos profesionales.
Como es una asignatura de 2º curso los alumnos tienen un conocimiento previo de programación; en esta asignatura los alumnos debe madurar dichos conocimientos, ampliándolos y desarrollando principalmente una capacidad de abstracción de modo que adquieran los conocimientos necesarios para comprender y aplicar con fluidez las estructuras de datos apropiadas, y desarrollen también algoritmos adecuados.
Las competencias que esta asignatura comparte con las demás asignaturas relacionadas con la Programación son las siguientes:

- Fundamentos y metodología de la programación
- Algoritmia
- Computabilidad
- Estructuras de datos
- Desarrollo de software (Procesos, Requisitos, Especificación y Diseño)
- Gestión de Proyectos
- Calidad del Software
- Interacción Persona-Computadora

Contextualización curricular

Las competencias genéricas y específicas del Título que están relacionadas de una forma directa con los contenidos de esta materia se pueden resumir en las siguientes:

1. Dirigir y coordinar el proyecto de desarrollo y mantenimiento de aplicaciones, supervisando las funciones y recursos de análisis funcional, orgánico y programación, asegurando la adecuada explotación de las aplicaciones.
2. Dominar todas las etapas de la vida de un proyecto (análisis de concepción, análisis técnico, programación, pruebas, documentación y formación de usuarios).
3. Control y seguimiento de plazos, indicadores económicos y de calidad.
4. Analizar y recoger nuevas técnicas y herramientas del mercado estudiando su viabilidad y necesidad. Posibilidad de contratar recursos externos.
5. Control y Gestión del Desarrollo del Proyecto Informático.
6. Realizar el análisis y el diseño detallado de las aplicaciones informáticas.
7. Definir la estructura modular y de datos para llevar a cabo las aplicaciones informáticas que cumplan con las especificaciones funcionales y restricciones del lenguaje de programación.
8. Definición y descripción de procedimientos e interfaz de usuario.
9. Realizar pruebas que verifiquen la validez funcional, la integridad de los datos y el rendimiento de las aplicaciones informáticas.
10. Elaborar y mantener documentación descriptiva de la génesis, producción y operatividad de las aplicaciones informáticas.
11. Participar en el diseño de nuevos sistemas informáticos como consecuencia de la información de áreas de la empresa que utilizan para el desarrollo de sus tareas, métodos y procesos manuales.
12. Integrar sistemas informáticos existentes susceptibles de inter-relacionarse.
13. Escuchar y asesorar a los usuarios en la resolución de los problemas que se les plantean con el uso de los sistemas informáticos.
14. Mantenerse al día en Técnicas, Métodos y Herramientas de Análisis y Diseño.
15. Creación de los tests de pruebas para verificar que los Sistemas Informáticos cumplen los requisitos y especificaciones de Análisis y Diseño.
16. Dirección del arranque o “lanzamiento” de un nuevo sistema.
17. Ayudar al Área de Estudios en la resolución de los fallos que se producen en los Sistemas en Producción.
18. Estudio de Métodos, Técnicas y Herramientas de Análisis y Diseño.
19. Supervisar, controlar y dar validez a los procesos de desarrollo.
20. Asesorar a los programadores en los problemas que se les plantean con la programación de los sistemas.
21. Dirigir el equipo de trabajo compuesto por Analistas Funcionales, Analistas de aplicaciones, Programadores.

Además de las competencias específicas que se han nombrado, existen interrelaciones con otras materias y con otras competencias profesionales, estas son las siguientes:

22. Planificar, supervisar y coordinar el desarrollo, implantación y mantenimiento de los sistemas operativos, software de mercado y propio, básico o de soporte
23. Garantizar una calidad permanente a través de los procedimientos y de las herramientas. Apoyar las demandas cotidianas de los usuarios

*Contextualización personal**

Itinerarios de procedencia y requisitos formativos de los alumnos

Al tratarse de una asignatura de 2º curso de Ingeniería Informática, los alumnos proceden del 1º curso de dicha ingeniería. Deben tener una formación básica en programación, más específicamente, deberían tener conocimientos sobre:

- fundamentos y metodología de la programación
- lenguajes de programación
- algoritmia y computabilidad
- y conocimientos básicos sobre desarrollo de software (procesos, requisitos, especificación y diseño)

Todos estos conocimientos se imparten durante el primer cuatrimestre de 2º curso en la asignatura Estructuras de Datos y Algoritmos (EDA, que continúa durante todo el curso) y durante el 1º curso de la Ingeniería Informática (las asignaturas más relacionadas con Laboratorio de Programación II que se imparten en el 1º curso son: Elementos de Programación (EP) y Laboratorio de Programación I (LPI)). Si los alumnos tuvieran dichos conocimientos, el estudio de esta materia no debería plantear ningún problema para ellos. El problema surge cuando hay alumnos que no han superado las asignaturas de primer curso (especialmente EP y LPI) o no han asimilado los contenidos impartidos en la asignatura EDA, y se matriculan de esta materia, o bien, que aún ni se han matriculado de dichas asignaturas de primer curso. El resultado en estos casos suele ser muy negativo para el alumno, ya que es incapaz de superar los contenidos planteados en la materia (contenidos muy avanzados de programación), el alumno se frustra y en muchos casos esta frustración provoca que el alumno deje las asignaturas de programación por considerarlas "muy difíciles" de aprobar.

Aconsejamos encarecidamente la superación de las materias EP y LPI para matricularse en esta materia, ya que está demostrado que aquellos alumnos que sí han superado las materias de primer curso, especialmente aquellas relacionadas de una forma más directa con programación, suelen superar esta materia sin mayor dificultad.

Otras consideraciones de interés

Nos gustaría resaltar como co-rrequisito a la hora de superar esta materia que el alumno estuviera matriculado durante el mismo curso académico en las asignaturas: Matemática discreta, Estadística y EDA, ya que los contenidos de estas asignaturas están relacionados y pensados para el alumno los estudie conjuntamente.

De forma general, nos gustaría destacar el bajo nivel que demuestran los alumnos en la expresión escrita y oral (además una gran deficiencia en conocimientos elementales relacionados con las matemáticas), necesidades esenciales para un futuro Ingeniero en Informática. Por tanto, será necesario incidir en estas competencias transversales a lo largo de su formación.

Es muy importante que estos alumnos, de segundo (y primer) curso, tengan más información acerca de la titulación en la que se encuentran, puesto que rara vez sus expectativas corresponden con los objetivos de la titulación (con las consecuencias que este hecho conlleva). En las asignaturas de programación muchos alumnos se dan cuenta de que estas asignaturas no tienen ninguna relación con sus espectativas sobre la titulación de Ingenería Informática (que normalmente son todo lo relacionado con internet, navegación web, chats, foros, juegos, etc.) y dado el trabajo que estas asignaturas conllevan, muchos de los alumnos piensan que no podrán superarlas. Es indispensable que los alumnos tengan una guía detallada de la universidad, de la titulación y de cada una de las materias, incluídos sus objetivos, antes de realizar la matrícula.

Por último, nos gustaría resaltar el hecho de que en primer curso se realizan automáticamente algunas convalidaciones a titulados en los ciclos formativos superiores de algunas especialidades relacionadas con Informática. En particular, la asignatura EP se les convalida a algunos alumnos. El resultado es que estos estudiantes no conocen todos los contenidos ni las metodologías que se emplean, lo que puede acarrearles problemas a la hora de superar asignaturas de programación en cursos posteriores, como es el caso de nuestra materia.

II. Objetivos

Relacionados con competencias académicas y disciplinares	*Vinculación*
Descripción	*CET*[i]
1. Consolidar la metodología de desarrollo de programas empleando la Orientación a Objetos	6, 8
2. Desarrollar la capacidad de abstracción del alumnado, de modo que adquieran los conocimientos necesarios para comprender y aplicar con fluidez las estructuras de datos apropiadas, y desarrollen también algoritmos adecuados	2, 6, 7, 8
3. Permitir a los alumnos resolver supuestos que supongan la implementación de programas de tamaño medio-grande, con codificaciones correctas y eficientes, utilizando el paradigma orientado a objetos	2, 6, 7, 8
4. Capacitar al alumno para que esté en condiciones de participar en todas las fases de los proyectos de desarrollo de software en entornos profesionales	2-4, 8, 11, 13-19
5. Comprender la necesidad y la función de los lenguajes de programación, así como los diferentes tipos, su utilidad y aplicación. Con el objetivo de ofrecer una panorámica general que permita al alumno, en etapas de desarrollo posterior, elegir el más adecuado	1, 6, 8, 20
6. Conocer y utilizar las herramientas aprendidas para valorar las soluciones aportadas al diseño de un programa, verificar que estas soluciones cumplen con los objetivos propuestos y que realizan las tareas de forma eficaz	3, 15-20
7. Ser capaz de plasmar de forma escrita los pasos realizados en el proceso de desarrollo software, de manera que tanto el usuario de la aplicación, como otros desarrolladores, sean capaces de entender la solución propuesta	5, 8, 10, 15
8. Capacitar al alumno como desarrollador de aplicaciones a adaptarse a los cambios de versiones, aparición de nuevas aplicaciones y nuevas formas de trabajo	9, 12, 14, 18
	1, 13, 17, 20, 21
Relacionados con otras competencias personales y profesionales	*Vinculación*
Descripción	*CET*
9. Provocar el interés por el desarrollo metódico de programas, atendiendo al diseño principalmente antes que a la implementación.	2, 23
10. Capacitar al alumno para trabajar en grupo, compartiendo recursos software y hardware	1, 13, 17, 20, 21, 22

III. Contenidos

*Selección y estructuración de conocimientos generales**

Bloque 1. Programación Orientada a Objetos

Tema 1
Tema 2
Tema 9
Tema 10
Tema 11
Tema 12

Bloque 2. Estructuras de Datos

Tema 3
Tema 4
Tema 5
Tema 6
Tema 8

Bloque 3. Análisis y Diseño

Tema 7
Tema 13

Secuenciación de bloques temáticos y temas

TEMA 1. Programación Orientada a Objetos.

- 1.1. Introducción. Clases y objetos.
  - 1.1.1. Motivaciones y conceptos fundamentales.
  - 1.1.2. Programación Orientada a Objetos como metodología.
- 1.2. Definición de una clase.
  - 1.2.1. Instancias de clases.
  - 1.2.2. Acceso a miembros de la clase: encapsulación.
  - 1.2.3. Métodos y mensajes.
  - 1.2.4. Métodos en línea y fuera de línea.
  - 1.2.5. Archivos de cabecera y clases.
- 1.3. Constructores y destructores.
- 1.4. Punteros a objetos. Creación de objetos dinámicos.
  - 1.4.1. Reserva dinámica de memoria para objetos.
  - 1.4.2. Métodos constructores.
- 1.5. Asignación de objetos dinámicos. Atributos dinámicos.
- 1.6. Destrucción de objetos dinámicos.
  - 1.6.1. Liberación de memoria de objetos.
  - 1.6.2. Métodos destructores.
- 1.7. Referencias.
- 1.8. Aplicación en C++.
- 1.9. Problemas resueltos.
- 1.10. Problemas propuestos.

TEMA 2. Sobrecarga.

- 2.1. Introducción. Concepto de sobrecarga.
- 2.2. Sobrecarga de funciones.
- 2.3. Sobrecarga de métodos.
- 2.4. Sobrecarga de operadores.
  - 2.4.1. Sobrecarga de operadores unarios.
  - 2.4.2. Sobrecarga de operadores binarios.
  - 2.4.3. Sobrecarga del operador de asignación.
  - 2.4.4. Sobrecarga como métodos y como funciones amigas.
- 2.5. Aplicación en C++.
- 2.6. Problemas resueltos.
- 2.7. Problemas propuestos.

TEMA 3. Estructuras de Datos Lineales.

- 3.1. Lista.

3.2. Cola.
3.3. Pila.
3.4. Implementación en C++.
3.5. Problemas resueltos.
3.6. Problemas propuestos.

TEMA 4. Programación genérica. Plantillas (templates).
4.1. Introducción. Genericidad.
4.2. Fundamentos teóricos.
4.3. Plantillas de funciones.
4.3.1. Definición de una plantilla de función.
4.3.2. Ejemplos de plantilla de funciones.
4.4. Plantillas de clases.
4.4.1. Definición de una plantilla de clases.
4.4.2. Instanciación de una plantilla de clases.
4.5. Plantillas frente a polimorfismo.
4.6. Aplicación en C++.
4.7. Problemas resueltos.
4.8. Problemas propuestos.

TEMA 5. Biblioteca de plantillas estándar (STL).
5.1. Introducción a la STL. Conceptos clave.
5.2. Contenedores e iteradores.
5.3. Contenedores y algoritmos.
5.4. Aplicación en C++.
5.5. Problemas resueltos.
5.6. Problemas propuestos.

TEMA 6. Árbol Binario de Búsqueda.
6.1. Repaso conceptos teóricos.
6.2. Recorridos.
6.3. Implementación en C++.
6.4. Problemas resueltos.
6.5. Problemas propuestos.

TEMA 7. Análisis Orientado a Objetos básico.
7.1. Identificación de clases: análisis gramatical.
7.2. Identificación de atributos.
7.3. Identificación de operaciones.

TEMA 8. Grafo.
8.1. Repaso conceptos teóricos.
8.2. Recorridos.
8.3. Implementación en C++.
8.4. Problemas resueltos.
8.5. Problemas propuestos.

TEMA 9. Patrones de diseño.
9.1. Introducción teórica.
9.2. Patrón de diseño Singleton.
9.3. Implementación en C++.
9.4. Problemas resueltos.
9.5. Problemas propuestos.

TEMA 10. Manejo de Excepciones.
10.1. Introducción. Conceptos generales.
10.2. La gestión de errores.
10.2.1. Excepciones. Lanzamiento de excepciones.
10.2.2. Manejo de excepciones.
10.2.3. Captura de excepciones. Relanzar excepciones.
10.2.4. Restricción de excepciones permitidas.
10.2.5. Excepciones definidas por el usuario.
10.2.6. Excepciones en constructores y destructores.
10.3. Aplicación en C++.
10.4. Problemas resueltos.
10.5. Problemas propuestos.

TEMA 11. Herencia.

- 11.1. Introducción. Concepto de herencia.
  - 11.1.1. Clase base y clase derivada.
  - 11.1.2. Consideraciones de diseño.
- 11.2. Tipos de herencia.
  - 11.2.1. Herencia simple.
  - 11.2.2. Herencia múltiple.
  - 11.2.3. Herencia pública.
  - 11.2.4. Herencia privada.
  - 11.2.5. Herencia protegida.
- 11.3. Constructores y destructores en herencia.
- 11.4. Aplicación en C++.
- 11.5. Problemas resueltos.
- 11.6. Problemas propuestos.

TEMA 12. Polimorfismo.

- 12.1. Introducción. Concepto de polimorfismo.
  - 12.1.1. Punteros de tipos derivados.
- 12.2. Constructores y funciones virtuales.
- 12.3. Destructores y funciones virtuales. Destructor virtual puro.
- 12.4. Polimorfismo. Uso y ventajas.
- 12.5. Ligadura dinámica frente a ligadura estática.
- 12.6. Métodos virtuales puros y tipos abstractos.
- 12.7. Aplicación en C++.
- 12.8. Problemas resueltos.
- 12.9. Problemas propuestos.

TEMA 13. Documentación de Programas.

- 13.1. Introducción a la documentación de programas.
- 13.2. Documentación interna.
  - 13.2.1. Estilo de programación.
- 13.3. Documentación externa.
  - 13.3.1. Manual del Usuario.
    - 13.3.1.1. Descripción General del Sistema.
    - 13.3.1.2. Instalación del Sistema.
    - 13.3.1.4. El Entorno de Trabajo.
    - 13.3.1.3. Puesta en marcha del Sistema.
    - 13.3.1.5. Guía del Operador del Sistema.
    - 13.3.1.6. Lista de errores.
  - 13.3.2. Manual del Programador.
    - 13.3.2.1. Descripción del Manual del Programador.
    - 13.3.2.2. Especificaciones lógicas del sistema.
    - 13.3.2.3. Estructuras de Datos.
    - 13.3.2.4. Algoritmos.
    - 13.3.2.5. Descripción por Módulos.
    - 13.3.2.6. Etapas de desarrollo.
    - 13.3.2.7. Diagramas de Clase.
    - 13.3.2.8. Diagramas de secuencia.
    - 13.3.2.9. Diagramas de actividad.
    - 13.3.2.10. Futuras modificaciones.
    - 13.3.2.11. Listados.
- 13.4. Aplicación en C++.
- 13.5. Problemas resueltos.
- 13.6. Problemas propuestos.

Interrelación			
Requisitos (Rq) y redundancias (Rd)		Tema	*Procedencia*
Conocimiento previo sobre conceptos básicos de Programación	Rq	1	EP
Conocimiento previo sobre conceptos básicos de Programación Orientada a Objetos	Rq	1	LPI

Conocimiento previo sobre Documentación de programas	Rq	13	E.P y LPI
Conocimiento previo sobre ADOO	Rq	7	LPI
Conocimiento previo de E/S en C++	Rq	Todos	LPI
Sobrecarga en C++	Rd	2	LPI
Recursividad	Rq	3, 6 y 8	EP
Estructuras de Datos Lineales	Rd	3	LPI
Arbol Binario de Búsqueda	Rq	6	EDA
Grafo	Rq	8	EDA

1. Metodología docente y plan de trabajo del estudiante

Actividades de enseñanza-aprendizaje				*Vinculación*	
Descripción y secuenciación de actividades	*Tipo*[ii]		*D*[iii]	*Tema*	*Objet.*
1. Presentación Asignatura	GG	T	1		
2. Sesión 00 – Entorno, S.O. (Primer proyecto)	S	P	1.5	1	5,10
3. Sesión 01 - Clases y Objetos. Creación y destrucción	S	P	1.5	1	1
4. Ejercicios de consolidación sesiones prácticas	NP	P	1	1	1
5. Trabajo práctico – implementación del proyecto de programación	NP	P	2	1	1
6. Lectura previa del guión del siguiente tema teórico	NP	T	0.5	1, 2	1
7. TEMA 1-2: POO. Composición (Repaso de LP1). Sobrecarga	GG	T	1	1, 2	1
8. Lectura previa del guión de los siguientes temas prácticos	NP	P	1	1, 3	1
9. Sesión 02 - Clases y Objetos. Paso de Objetos por parámetro	S	P	1.5	1	1
10. Sesión 03 - E.D. Lineales (Lista, Cola y Pila)	S	P	1.5	3	2
11. Ejercicios de consolidación sesiones prácticas	NP	P	2	1, 3	2
12. Trabajo práctico – implementación del proyecto de programación	NP	P	2	1, 3	2
13. Lectura previa del guión del siguiente tema teórico	NP	T	0.5	4, 5	2
14. TEMA 4-5: Genericidad y STL	GG	T	1	4, 5	1,2
15. Lectura previa del guión de los siguientes temas prácticos	NP	P	1	1, 3, 4	1,2
16. Sesión 04 - Ejercicios - Actividad Evaluable 1	S	P	1.5	1-3	1,2
17. Sesión 05 – Genericidad – E.D. Genéricas	S	P	1.5	4	1,2
18. Ejercicios de consolidación sesiones prácticas	NP	P	2	1-4	1,2
19. Trabajo práctico – implementación del proyecto de programación	NP	P	2	1-4	1,2
20. Lectura previa del guión del siguiente tema teórico	NP	T	0.5	7	1,2
21. TEMA 7: ADOO	GG	T	1	7	4,9
22. Lectura previa del guión de los siguientes temas prácticos	NP	P	1	5, 6	4,9
23. Sesión 06 – STL - Contenedores	S	P	1.5	5	1,2
24. Sesión 07 – Árbol Binario de Búsqueda	S	P	1.5	6	1,2
25. Ejercicios de consolidación sesiones prácticas	NP	P	2	1-6	1,2
26. Trabajo práctico – implementación del proyecto de programación	NP	P	2	1-6	1,2
27. Preparación EC1	NP	P	2.5	1-6	1,2

28. TEMA: Repaso, dudas y problemas EC1	GG	T	1	1-7	1,2
29. Lectura previa del guión de los siguientes temas prácticos	NP	P	1	6	1,2
30. Sesión 08 – Ejercicios A.B.B.	S	P	1.5	6	1,2
31. Sesión 09 – EC1 - Árbol Binario de Búsqueda	S	P	1.5	1-7	1,2
32. Ejercicios de consolidación sesiones prácticas	NP	P	2	1-7	1,2
33. Trabajo práctico – implementación del proyecto de programación	NP	P	2	1-7	1,2
34. Lectura previa del guión del siguiente tema teórico	NP	T	0.5	10	1,2
35. TEMA 10: Excepciones	GG	T	1	10	1,3
36. Lectura previa del guión de los siguientes temas prácticos	NP	P	1	8, 9	1,3
37. Sesión 10 – Grafo	S	P	1.5	8	1,2
38. Sesión 11 – Patrón Singleton	S	P	1.5	9	1,3,9
39. Lectura previa del guión del siguiente tema teórico	NP	T	0.5	11	1,3,9
40. TEMA 11: Herencia	GG	T	1	11	1,3
41. Lectura previa del guión de los siguientes temas prácticos	NP	P	1	8, 9, 10	1,3
42. Sesión 12 – Excepciones	S	P	1.5	10	1,3
43. Sesión 13 – Ejercicios - Actividad Evaluable 2	S	P	1.5	8, 9	1,3
44. Ejercicios de consolidación sesiones prácticas	NP	P	2	1-9	1,3
45. Trabajo práctico – implementación del proyecto de programación	NP	P	2	1-9	1,3
46. Lectura previa del guión del siguiente tema teórico	NP	T	0.5	12	1,3
47. TEMA 12: Polimorfismo	GG	T	1	11	1,3
48. Lectura previa del guión de los siguientes temas prácticos	NP	P	1	11, 12	1,3
49. Sesión 14 – Herencia	S	P	1.5	11	1,3
50. Sesión 15 – Polimorfismo	S	P	1.5	12	1,3
51. Ejercicios de consolidación sesiones prácticas	NP	P	1	1-12	1,3
52. Trabajo práctico – implementación del proyecto de programación	NP	P	3	1-12	1,3
53. Preparación EC2	NP	P	2.5	1-12	1,3
54. Lectura previa del guión del siguiente tema teórico	NP	T	0.5	12	1,3
55. TEMA: Repaso, dudas y problemas EC2	GG	T	1	8-12	1,3
56. Lectura previa del guión de los siguientes temas prácticos	NP	P	1	8	1,3
57. Sesión 16 – Ejercicios Grafo	S	P	1.5	8	1,2
58. Sesión 17 – EC2 – Grafo	S	P	1.5	8	1,2
59. Ejercicios de consolidación sesiones prácticas	NP	P	1	1-12	1,2
60. Trabajo práctico – implementación del proyecto de programación	NP	P	3	1-12	1,2
61. TEMA: Explicación práctica de ejemplo	GG	T	1	8-12	1,3,9
62. Lectura previa del guión de los siguientes temas prácticos	NP	P	1	3, 11, 12	1,3,9
63. Sesión 18 – E.D. Polimórficas	S	P	1.5	3, 12	1,2
64. Sesión 19 – Ejercicios herencia, polimorfismo	S	P	1.5	11, 12	1,3
65. Ejercicios de consolidación sesiones prácticas	NP	P	1	1-12	1,2
66. Trabajo práctico – implementación del proyecto de programación	NP	P	3	1-12	1,2
67. Lectura previa del guión del siguiente tema teórico	NP	T	0.5	13	1,2
68. TEMA 13: Documentación	GG	T	1	13	7
69. Lectura previa del guión de los siguientes temas prácticos	NP	P	1	8	7
70. Sesión 20 – STL – Algoritmos	S	P	1.5	5	1,2
71. Sesión 21 – Ejercicios - Actividad Evaluable 3	S	P	1.5	9, 10	1,2
72. Ejercicios de consolidación sesiones prácticas	NP	P	1	1-13	1,2
73. Trabajo práctico – implementación del proyecto de programación	NP	P	3	1-13	1,2
74. TEMA: Problemas y dudas	GG	T	1	1-13	1,3
75. Sesión 22 – Habilidades de programación	S	P	1.5	1-13	4,5,6,8
76. Sesión 23 – Trabajo libre y dudas	S	P	1.5	1-13	3
77. Ejercicios de consolidación sesiones prácticas	NP	P	1	1-13	3
78. Trabajo práctico – implementación del proyecto de programación	NP	P	3	1-13	3
79. Sesión 24 – Trabajo libre y dudas	S	P	1.5	1-13	3
80. Sesión 25 – Trabajo libre y dudas	S	P	1.5	1-13	3
81. Ejercicios de consolidación sesiones prácticas	NP	P	1	1-13	3
82. Trabajo práctico – implementación del proyecto de programación	NP	P	2	1-13	3
83. Preparación EC3	NP	P	2.5	1-13	3
84. Sesión 26 – EC3 – Entrega Final	S	P	1.5	11, 12	3
85. Estudio y preparación del examen final	NP	T-P	8	1-13	3

86. Examen final	S	T-P	2.5	1-13	3

Distribución del tiempo (ECTS)			*Dedicación del alumno*		*Dedicación del profesor*	
Distribución de actividades		*Nº alumnos*	*H. presenciales*	*H. no presenc.*	*H. presenciales*	*H. no presenc.*
Grupo grande (Más de 20 alumnos)	Coordinac./evaluac. (I)	-	-	-	7	28
	Teóricas (II y III)	50	12	4	12	-
	Prácticas (IV, V y VI)	-	-	-	-	-
	Subtotal	**50**	**12**	**4**	**19**	**28**
Seminario-Laboratorio (6-20 alumnos)	Coordinac./evaluac. (I)	-	6	4.5	14	55
	Teóricas (II y III)	-	-	-	-	
	Prácticas (IV, V y VI)	20	34.5	68	34.5	84
	Subtotal	**20**	**40.5**	**72.5**	**48.5**	**139**
Tutoría ECTS (1-5 alumnos)	Coordinac./evaluac. (I)		-	-		
	Teóricas (II y III)		-	-		
	Prácticas (IV, V y VI)		-	-		
	Subtotal		-	-		
Tutoría comp. y preparación de ex. (VII)			-	-	2.5	5
Totales			**52.5**	**76.5**	**70**	**144**

*Otras consideraciones metodológicas**

Recursos y metodología de trabajo en las actividades presenciales

Las sesiones presenciales serán de dos tipos diferenciados: sesiones de teoría y sesiones de práctica. En las sesiones de teoría, el alumno dispone (con la suficiente antelación) de un documento con apuntes previos sobre el tema que el profesor explicará en la sesión de teoría. Sobre este material, el alumno podrá realizar consultas al profesor sobre aquellos conceptos que no han quedado claros. Por tanto, la metodología seguida para las sesiones de teoría requiere una lectura y trabajo previo por parte del alumno (la dedicación aconsejada es aproximadamente de media hora) de los apuntes dejados por el profesor para cada uno de los temas vistos en la sesión de teoría. Una vez resueltas las dudas que los alumnos puedan tener sobre el tema de la sesión, el profesor explicará el tema teórico con ejemplos prácticos, de modo que los alumnos puedan asimilar de una manera más sencilla los conceptos que se presentan en el tema de teoría.

En cuanto a las sesiones prácticas, el alumno parte de un guión previo en el que tiene detallados los conceptos que se van a explicar en la sesión, una serie de ejercicios previos aconsejados así como una serie de ejercicios que deberá realizar en dicha sesión. En este guión, se detallarán también los ejercicios propuestos que el alumno debe realizar para afianzar los conceptos vistos en la sesión presencial

De este modo, los alumnos disponen desde la primera sesión tanto de teoría como de prácticas de una planificación completa de los temas y trabajos prácticos que van a explicarse en cada una de las sesiones durante todo el curso.

Recursos y metodología de trabajo en las actividades semi-presenciales y no presenciales

Como se ha comentado anteriormente, en las sesiones presenciales, los alumnos dispondrán de un guión previo en el que se detallarán los conceptos a desarrollar en la sesión y los ejercicios propuestos para ser realizados posteriormente por el alumno de manera no presencial. De este modo, la metodología de trabajo que se sigue en la asignatura será la misma durante todo el curso. El alumno leerá y comprenderá el guión previo de la sesión, de este modo conocerá en todo momento el tema que será explicado en la sesión. Posteriormente, el alumno asiste a la sesión presencial, asimilando los conceptos explicados por el profesor. Por último, el alumno realiza los ejercicios propuestos en el guión de la sesión, de manera que pueda afianzar los conceptos explicados en la sesión presencial.

Durante todo el curso el alumno deberá desarrollar un proyecto práctico (de una forma incremental, a lo largo de todo el cuatrimestre). Hasta ahora el desarrollo de dicho proyecto práctico se ha ido realizando mediante distintas entregas (tres) a lo largo del curso. A partir del próximo curso, nuestra idea será ir aplicando técnicas

de PBL, de forma que se entregará un único enunciado de proyecto a principio del curso y los alumnos irán realizando entregas del proyecto en grupos de 3 ó 4 alumnos.
Se hace uso además de una plataforma virtual en la que se van publicando tanto los temas teóricos como las sesiones prácticas, así como la siguiente información útil para los alumnos:

- ejercicios resueltos, ejercicios con errores para que los alumnos los solucionen, etc.
- material bibliográfico aconsejado (por temas específicos para las sesiones impartidas en cada semana) como ayuda al alumno, este material está formado por algunos temas de libros o links en los que aparecen ejemplos y explicaciones sobre las sesiones relacionadas
- manuales de consulta en internet
- código implementado de ayuda para realizar el trabajo práctico
- plantillas para realizar la documentación solicitada
- enunciados de exámenes propuestos en otros cursos anteriores

Además, se proponen actividades para que los alumnos se impliquen de una forma más directa en la asignatura: foros, cuestionarios sobre algunos temas o aspectos de la materia, etc.

Recursos y metodología de trabajo para los alumnos que no han alcanzado los requisitos

Se plantean durante el curso algunas actividades no recuperables, de forma que se valora al alumno que asiste continuadamente tanto a las sesiones prácticas como a las teóricas y además realiza todas las tareas planteadas. Además, se plantean tres entregas distintas del trabajo práctico (que se realiza de una forma incremental) que pueden ser superadas cada una por separado. Si algún alumno no supera alguna de las entregas prácticas, podrá recuperarla (también de forma individual) en la convocatoria de junio. En las siguientes convocatorias se realizará un examen completo sobre todo el trabajo práctico desarrollado durante el curso. Previamente a cada entrega, los alumnos deberán superar un "cuestionario sobre conocimientos básicos" para poder presentarse a cada una de las pruebas prácticas. Este cuestionario puede ser recuperado en cada convocatoria a la que se presente el alumno.
Al mismo tiempo, los alumnos que no alcancen los requisitos establecidos deberán asistir a las respectivas revisiones de evaluación que el profesor realizará después de cada convocatoria de evaluación. En estas reuniones de revisión (realizadas de manera individual con el alumno), el profesor le explicará al alumno los conceptos que debe reforzar para poder aprobar la asignatura, de modo que el alumno pueda reforzar esos conceptos de manera individual. Para reforzar los conceptos comentados, el profesor puede facilitar al alumno bibliografía recomendada si así lo considera oportuno.
Creemos fundamental en esta materia valorar el trabajo continuado del alumno, ya que es una materia que requiere mucho trabajo práctico (tanto en laboratorios como trabajo no presencial; durante este año hemos realizado distintas encuestas a los alunos y hemos constatado que superan en un alto número el número de horas estimadas). Además, creemos fundamental valorar más a aquellos alumnos que siguen la materia durante el curso y llegan a los objetivos durante el cuatrimestre o incluso en convocatoria de junio (ese es el motivo de plantear actividades no recuperables durante todo el cuatrimestre), ya que aquellos alumnos que se presentan a convocatoria de septiembre o posteriores, además de haber tenido más tiempo para el desarrollo práctico de la asignatura pueden haber solicitado "ayuda externa", en nuestro centro es un grave problema los trabajos prácticos que son realizados en academias y son entregados en las distintas convocatorias.

Recursos y metodología de trabajo para desarrollar competencias transversales

Durante este curso estamos utilizando de una forma continuada la plataforma Avuex, creemos que esta plataforma es fundamental para el desarrollo de ciertas competencias como:

- comentar con los compañeros temas teóricos y prácticos
- búsqueda de documentos
- utilización de bibliografía
- lectura y comentarios sobre código implementado por otra persona (en nuestro caso y por ahora siempre por los profesores, en un futuro lo realizaremos con código desarrollado por los alumnos)
- aprender a esquematizar la información que dejan los profesores en Avuex como referencia
- organización del trabajo (planning disponible desde el primer día de curso)

En un futuro, nuestra idea es aplicar a esta materia PBL, creemos que de esta forma, además se desarrollan competencias transversales como: trabajo en grupo, enfrentarse a un problema completo y aprender a

dividir el trabajo, presentación pública y defensa de un trabajo desarrollado por un grupo de personas, lectura de material y optimización de información recibida, etc.

V. Evaluación

*Criterios de evaluación**	*Vinculación**	
Descripción	*Objetivo*	*CC*[iv]
Capacidad de enfrentarse a un problema de tamaño medio-grande, participando y resolviendo cada una de las fases de problema	2-4, 8, 11, 13-19	30%
Utilización correcta de Estructuras de Datos y resolución de algoritmos	3, 6, 8, 15-20	15%
Capacidad de plasmar de forma escrita todos los pasos realizados en el proceso de desarrollo software	5, 8, 10, 15	10%
Manejo del software de edición, creación y depuración de programas	2, 3, 6, 7, 8, 15-20	5%
Capacidad de modificar (utilizando correctamente todos los conceptos sobre Programación Orientada a Objetos - POO aprendidos en la materia) el software creado según nuevas especificaciones	9, 12, 14, 18	30%
Capacidad de trabajar en grupo	1, 13, 17, 20, 21	10%

Actividades e instrumentos de evaluación		
Grupo Grande	Se realizarán tres cuestionarios a los alumnos de “conocimientos básicos”. Dichos cuestionarios constarán de 5-8 preguntas cortas sobre todos los contenidos estudiados en la asignatura. Este cuestionario se realizará antes de cada prueba práctica en el laboratorio y la no superación de cada cuestionario imposibilitará al alumno a presentarse a cada prueba práctica. (La recuperación de esta actividad se realizará junto con el examen final, con un único cuestionario de 15-20 preguntas y deberá aprobarse de forma obligatoria para poder realizar las pruebas prácticas de la asignatura)	10% R
Laboratorio	Resolución de tres actividades evaluables a lo largo del cuatrimestre. Serán pruebas cortas sobre los contenidos de la asignatura realizadas en algunas de las sesiones prácticas de la asignatura.	10% NR
Laboratorio	Elaboración en grupo de un proyecto de programación en tres fases, con la documentación generada y una prueba de modificación del proyecto en cada fase (en la recuperación de la actividad, se podrá obtener, como máximo, un 80% de la nota máxima). Evaluación conjunta (sólo en exámenes parciales y conv. Junio): Si todos los alumnos de un mismo grupo superan cada una de las fases del proyecto (EC1, EC2, EC3) se subirá 1 punto a todos los miembros de ese grupo.	70% R
No Presencial	Utilización y resolución de cuestionarios de autoevaluación que se dejarán propuestos en el aula virtual de la asignatura.	5% NR
Otras Actividades	A lo largo del curso, en las clases de grupo grande, de laboratorio y en la web de la asignatura se propondrán varias actividades (entre 5-8): resolución de problemas, propuesta de nuevos problemas y preguntas de test, participación en foros, desarrollo de contenidos, etc. La resolución de manera correcta de una de esas actividades supondrá un punto.	5% NR
Examen Parcial	Constará de la resolución del cuestionario de “conocimientos básicos” (GG) + Proyecto de programación en tres fases (LAB). Este examen se realizará en tres fases a lo largo de todo el cuatrimestre. La evaluación se realizará de forma	70% + 10%

	conjunta.	
Examen Final (Junio)	Constará de la resolución del cuestionario de “conocimientos básicos” (GG) + Proyecto de programación en tres fases (LAB). La evaluación se realizará de forma conjunta. El alumno deberá presentarse sólo a aquella fase(s) que no haya superado en los exámenes parciales.	70% + 10%
Examen Final (demás convocatorias)	Constará de la resolución del cuestionario de “conocimientos básicos” (GG) + Proyecto de programación en tres fases (LAB). La evaluación se realizará de forma individual. El alumno deberá presentarse a las tres fases del proyecto de programación.	70% + 10%

VI. Bibliografía

Bibliografía de apoyo seleccionada

"Programación en C++. Algoritmos, estructuras de datos y objetos". Luis Joyanes Aguilar. Editorial McGraw-Hill
"Programación en C++. Un enfoque práctico". Serie Schaum. Luis Joyanes Aguilar y Lucas Sánchez García. Editado por McGraw-Hill. 2006
"El Lenguaje de Programación C++", Bjarne Stroustrup, Addison-Wesley, 2002
C++ estándar. Enrique Hernández Orallo, José Hernández Orallo, Mª Carmen Juan Lizandra. . Paraninfo. 2001
Problemas Resueltos de Programación en C++. José Daniel García Sánchez, José María Pérez Menor, Luis Miguel Sánchez García, Félix García Carballeira. Paso a Paso. Thomson. 2004
“Estructuras de Datos y A lgoritmos”. Roberto Hernández, Juan Carlos Lázaro, Raquel Dormido, Salvador Ros. Universidad Nacional de Educación a Distancia (Prentice Hall)

*Bibliografía o documentación de lectura obligatoria**

"Programación Orientada a Objetos". Roberto Rodríguez Echeverría, Encarna Sosa Sánchez y Álvaro Prieto Ramos. Editado por Librería Álvaro (Cáceres). 2004

*Bibliografía o documentación de ampliación, sitios web…**

“Object-Oriented Analysis & Design”. McLaughlin, Pollice and West. Head First. O'reilly. 2006

“Thinking in C++”. Bruce Eckel. http://www.mindview.net/Books/TICPP/ThinkingInCPP2e.html

“Navigating C++ and Object-Oriented Design”. Google books. http://books.google.es/books?id=b-NiT6w8FTAC&dq=c%2B%2B+prentice+hall&pg=PP3&ots=EwQvKn02iN&sig=aOox022oQexN2AM9BHm1q4aSYiY&prev=http://+www.google.es/search%3Fhl%3Des%26client%3Dfirefox-a%26rls%3Dcom.ubuntu%+253Aes-ES%253Aofficial%26hs%3DHiP%26q%3Dc%252B%252B%2Bprentice%2Bhall%2+6btnG%3DB%25C3%25BAsqueda%26meta%3D&sa=X&oi=print&ct=result&cd=2#PPP1,M+1

“Entrada C++ en Wikipedia”: http://en.wikipedia.org/wiki/C++

“Cuadernos Didácticos Análisis y Diseño Orientado a Objetos”, Cueva Lovelle, J.M. Editorial Servitec. Portal e-libro: http://site.ebrary.com/lib/extremasp/Top?channelName=extremasp&cpage=1&f00=text&frm=smp.x&hitsPerPage=10&id=10061284&layout=document&p00=orientada+a+objetos&sch=sch&sch.x=0&sch.y=0&sortBy=score&sortOrder=desc

---

[i] ***Códigos*.-**
CET: Competencias Específicas del Título (véase el apartado de Contextualización curricular)

[ii] *Tipos de actividades*: GG (Grupo Grande); S (Seminario o Laboratorio); Tut (Tutoría ECTS); No presenciales (NP); C-E, I (Coordinación o evaluación); T, II (Teórica de carácter expositivo o de aprendizaje a partir de documentos); T, III (Teórica de discusión); P, IV (Prácticas basadas en la solución de problemas); P, V (Prácticas basadas en la observación, experimentación, aplicación de destrezas, estudio de casos…); P, VI (Prácticas con proyectos o trabajos dirigidos); T-P, VII (Otras teórico-prácticas).

[iii] *D: Duración* en sesiones de 1 hora de trabajo presencial o no presencial (considerando en cada hora 50-55 minutos de trabajo neto y 5-10 de descanso).

[iv] *CC: Criterios de Calificación* (ponderación del criterio de evaluación en la calificación cuantitativa final).

[v] *NR*: actividad "no recuperable" o que no permite evaluación extraordinaria.

(*) Apartados no obligatorios.