Plan Docente de una materia

I. Descripción y contextualización

Identificación y características de la materia				
Denominación y código	Elementos de programación			
Curso y Titulación	**1º de Ingeniería Informática**			
Área	Lenguajes y sistemas informáticos			
Departamento	*Informática*			
Tipo	TR			
Coeficientes	Practicidad: 4 (media-alta)		Agrupamiento: 3 (medio)	
Duración ECTS (créditos)	**9 créditos (225 horas)**		**anual**	
Distribución ECTS (rangos)	Grupo Grande: 26,67%	Seminario-Lab.: 13,33%	Tutoría ECTS: 0%	No presenciales: 60%
	60 horas	30 horas	0 horas	135 horas
Descriptores (según BOE)	**(6+3 créditos)** Diseño de algoritmos. Análisis de algoritmos. Lenguajes de programación. Diseño de programas. Técnicas de verificación y prueba de programas.			
Coordinador-Profesor/es	Pedro José Clemente Martín (pjclemente@unex.es) Alberto Gómez Mancha (agomez@unex.es) ***(Coordinador)*** Julia González Rodríguez (juliagon@unex.es) Roberto Rodríguez Echeverría (rre@unex.es) Encarna Sosa Sánchez (esosa@unex.es)			
Tutorías complementarias	e-mail	Teléfono	Despacho	
	agomez@unex.es	927.257.195 Ext:7807	12 (Pab. Informática)	
	pjclemente@unex.es	927.257.195 Ext:7808	11 (Pab. Informática)	
	juliagon@unex.es	927.257.195 Ext:7547	13 (Ed. Telecomunicaciones)	
	rre@unex.es	927.257.195 Ext:7546	14 (Ed. Telecomunicaciones)	
	esosa@unex.es	927.257.195 Ext:7806	10 (Pab. Informática)	
	Horarios por determinar; publicados en la web y en los despachos			

Contextualización profesional

Según el Libro Blanco publicado por la ANECA para el grado de Ingeniero/a en Informática, los **contenidos formativos comunes** de la Ingeniería Informática se deben organizar en las siguientes cuatro categorías:

- Fundamentos científicos
- Contenidos generales de la Ingeniería
- Contenidos específicos de la Ingeniería en Informática
- Proyecto Fin de Carrera (PFC)

Dentro de los contenidos específicos del título se encuentra la subcategoría Programación, en cuyo estudio se centra esta materia. En el Libro Blanco esta subcategoría debe tratar:

- Fundamentos y metodología de la programación
- Algoritmia
- Computabilidad
- Lenguajes de programación
- Paradigmas de programación
- Estructuras de datos

Son competencias fundamentales de esta asignatura:

- Fundamentos y metodología de la programación

En el borrador del título recientemente publicado, el bloque de Programación, con un total de 27 créditos ECTS, incluye los siguientes conocimientos, capacidades y destrezas: "Comprender los fundamentos teóricos de programación: analizar la computabilidad y complejidad algorítmica. Conocer y utilizar lenguajes estructurados para el desarrollo de sistemas software. Conocer las estructuras de datos básicas, sus aplicaciones y propiedades. Determinar las estructuras de datos más adecuadas. Conocer y utilizar los principales paradigmas de programación. Saber emplear técnicas de verificación y validación de programas."

Esta asignatura se encargaría de dar una introducción a la mayoría de los conocimientos y destrezas anteriores, que se afianzarán, desarrollarán y profundizarán en asignaturas posteriores.

Esta materia debe ser tratada independientemente del perfil profesional que pretenda alcanzarse, al ser una materia de primer curso que debe considerarse la primera toma de contacto del estudiante con el ámbito de la programación, por lo que los objetivos fundamentales son que el alumno sepa analizar, diseñar e implementar pequeños programas, iniciándolo en la obtención de una visión crítica que deberá desarrollar posteriormente. También le dotaremos de las herramientas necesarias para la construcción de los programas. Cualquier perfil profesional que desarrolle el estudiante posteriormente se verá influido por su conocimiento de las técnicas básicas y avanzadas de programación y su capacidad para analizar y resolver problemas, presentar y valorar alternativas, competencias desarrolladas ampliamente en esta asignatura.

Al ser la primera asignatura de programación, tendrá una serie de competencias compartidas que deberá introducir y que serán desarrolladas posteriormente en otras asignaturas relacionadas con esta materia.

Competencias compartidas:

- Computabilidad
- Lenguajes de programación
- Paradigmas de programación
- Estructuras de datos

Esta asignatura también se relaciona con muchas de las asignaturas restantes que forman parte de los contenidos específicos de la Ingeniería Informática, ya que en casi todas se usa la programación de ordenadores para conseguir los objetivos formativos propuestos.

Contextualización curricular

Esta es una asignatura anual de primer curso de introducción a la programación, una herramienta básica en la formación de un Ingeniero en Informática. Por tanto, esta asignatura se encargará de iniciar muchas de las competencias genéricas y específicas de la titulación, que otras asignaturas posteriores desarrollarán.

Existe mucha interrelación entre esta asignatura y el resto de las asignaturas del primer curso, así como con el resto de asignaturas de cursos posteriores que utilizan la programación de ordenadores como herramienta.

La asignatura *Laboratorio de programación I* está muy relacionada con *Elementos de programación*, ya que comparten objetivos formativos. De hecho, si se pudiera modificar el plan de estudios, ambas asignaturas deberían ser sólo una, o bien dividirse en dos asignaturas cuatrimestrales sucesivas.

Las asignaturas de *Introducción a los computadores* y *Sistemas digitales* aportan las bases fundamentales del conocimiento de la parte física del ordenador y su funcionamiento interno.

Las asignaturas *Álgebra* y *Cálculo* proporcionan las herramientas matemáticas fundamentales que todo ingeniero debe saber emplear y que le serán de utilidad en asignaturas posteriores (incluidas las de programación).

Fundamentos físicos de la informática aporta las bases físicas de la electrónica digital que desarrollarán en asignaturas posteriores y que le ayudarán a comprender el funcionamiento interno básico de un ordenador.

Con todas estas asignaturas, el estudiante debería obtener una versión general del campo de actuación del Ingeniero en Informática y de las bases teóricas que le ayudarán en el desarrollo posterior de otras materias.

*Contextualización personal**

Al tratarse de una materia de primer curso los alumnos proceden de niveles inferiores del sistema educativo actual, fundamentalmente de bachillerato tecnológico y de ciclos formativos superiores.

En algunas ocasiones acceden alumnos procedentes de otros itinerarios académicos, que muestran grandes lagunas en algunos temas fundamentales (por ejemplo, alumnos del bachillerato biosanitario con un nivel muy bajo de matemáticas), lo que les impide el seguimiento adecuado de las clases.

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

Sin embargo, en los últimos años, las capacidades relacionadas con la informática a nivel de usuario ya están desarrolladas, lo que facilita el trabajo inicial.

Es muy importante que estos alumnos, de 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). Es indispensable que los alumnos tengan una guía detallada de la universidad, de la titulación y de cada una de las materias, con sus objetivos, antes de realizar la matrícula.

Desde este curso, se realizan automáticamente algunas convalidaciones a titulados en los ciclos formativos superiores de algunas especialidades relacionadas con la Informática. En particular, Elementos de programación de les convalida a algunos alumnos. Aunque estos estudiantes ya poseen parte de los conocimientos y de las competencias que se desarrollan en esta asignatura, no conocen todos los contenidos ni las metodologías que se emplean, lo que puede acarrearles problemas a la hora de superar asignaturas posteriores.

(Actualmente, los alumnos de ciclos formativos se matriculan en las titulaciones técnicas, y no en la ingeniería de ciclo largo, pero con los nuevos títulos de grado, entrarán en el grado.)

II. Objetivos

Relacionados con competencias académicas y disciplinares	***Vinculación***
Descripción	*CET*[i]
1. Contemplar la Informática desde una perspectiva histórica. El estudio de esta disciplina puede ayudar al alumno a entender el desarrollo y tendencias actuales en los distintos campos y ámbitos.	42
2. 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.	42
3. Conocer las técnicas y herramientas empleadas en el diseño y desarrollo de programas, buscando abstracción de los conocimientos para permitidles utilizarlas en cualquier problema con el que deban enfrentarse.	2,11
4. 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.	2,14,34
5. Aprender a determinar los requisitos que un algoritmo necesita para su correcta implementación.	2,10
6. Ser capaz de analizar programas que han sido escritos previamente por otros desarrolladores.	17,19
7. Ser capaz de aportar soluciones óptimas utilizando las herramientas aprendidas.	12,13
8. 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.	2,15,91,104

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.	2,108
10. Capacitar al alumno como desarrollador de aplicaciones a adaptarse a los cambios de versiones, aparición de nuevas aplicaciones y nuevas formas de trabajo.	7
11. Trabajar en grupo, compartiendo recursos software y hardware.	1,43

Competencias de la titulación relacionadas con las competencias de la materia		
Competencia específica de la Titulación (CET)		*CEM*
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.	11
2	Dominar todas las etapas de la vida de un proyecto (análisis de concepción, análisis técnico, programación, pruebas, documentación y formación de usuarios).	3, 4, 5, 8, 9
7	Analizar y recoger nuevas técnicas y herramientas del mercado estudiando su viabilidad y necesidad. Posibilidad de contratar recursos externos.	10
10	Interpretar las especificaciones funcionales encaminadas al desarrollo de las aplicaciones informáticas.	5
11	Realizar el análisis y el diseño detallado de las aplicaciones informáticas.	3
12	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.	7
13	Definición y descripción de procedimientos e interfaz de usuario.	7

14	Realizar pruebas que verifiquen la validez funcional, la integridad de los datos y el rendimiento de las aplicaciones informáticas.	4
15	Elaborar y mantener documentación descriptiva de la génesis, producción y operatividad de las aplicaciones informáticas.	8
17	Estudiar el sistema actual existente y analizar e idear mejores medios para llevar a cabo los mismos objetivos u otros adicionales.	6
19	Integrar sistemas informáticos existentes susceptibles de interrelacionarse.	6
34	Creación de los tests de pruebas para verificar que los Sistemas Informáticos cumplen los requisitos y especificaciones de Análisis y Diseño.	4
42	Estudio de la evolución de las nuevas tecnologías, sobre todo de aquellas que pueden aportar mejoras importantes en los sistemas utilizados en la empresa.	1, 2
43	Planificar, Supervisar y coordinar el desarrollo, implantación y mantenimiento de los sistemas operativos, software de mercado y propio, básico o de soporte.	11
91	Elabora la parte técnica de la propuesta.	8
104	Poner en marcha los procedimientos de prueba y de control de calidad.	8
108	Garantizar una calidad permanente a través de los procedimientos y de las herramientas. Apoyar las demandas cotidianas de los usuarios.	9

III. Contenidos

*Selección y estructuración de conocimientos generales**
Los objetivos principales de esta asignatura es proveer los conocimientos necesarios a los alumnos para que adquieran competencias en la resolución de algoritmos a través de programas implementados en un lenguaje concreto de programación. Para ello es fundamental el análisis de los Tipos Abstractos de Datos (TAD) y de las estructuras de datos requeridas en la manipulación de la información que gestionará el programa.

Secuenciación de bloques temáticos y temas
1. Introducción a la programación
1.1. Introducción 1.2. Historia de los ordenadores 1.3. Codificación y almacenamiento de la información 1.4. Estructura y funcionamiento básicos de un ordenador 1.5. Historia de los lenguajes de programación 1.6. Tendencias actuales
2. Entorno, expresiones y acciones fundamentales
2.1. Introducción a los algoritmos 2.2. Metodología de resolución de problemas 2.3. Documentación de programas 2.4. Entorno: elementos básicos 2.5. Tipos simples de datos 2.6. Operaciones y expresiones 2.7. Representación de algoritmos 2.8. Acciones elementales 2.9. Aplicación en C++
3. Programación modular
3.1. Introducción al diseño descendente 3.2. Módulos 3.3. Paso de información entre módulos 3.4. Precondiciones y postcondiciones 3.5. Juego de pruebas 3.6. Depuración y ejecución 3.7. Documentación 3.8. Aplicación en C++
4. Programación estructurada
4.1. Introducción 4.2. Estilo de programación

4.3. Estructuras básicas de control 4.4. Otras estructuras de control 4.5. Depuración 4.6. Aplicación en C++
5. Recursividad
5.1. Conceptos 5.2. Recursividad simple final 5.3. Recursividad múltiple 5.4. Recursividad simple no final 5.5. Recursividad simple no final
6. Análisis de algoritmos
6.1. Introducción 6.2. Eficiencia de programas 6.3. Complejidad algorítmica 6.4. Ejemplos de cálculo de complejidades
7. Tipos Abstractos de Datos (TAD)
7.1. Introducción 7.2. Notación de Liskov 7.3. Ejemplos de TAD de uso frecuente 7.4. TAD lineales
8. Estructura de datos registro
8.1. Introducción 8.2. Definición 8.3. Uso 8.4. Aplicación en C++
9. Estructura de datos vector
9.1. Introducción 9.2. Definición 9.3. Operaciones básicas 9.4. Procesamiento secuencial 9.5. Algoritmos de búsqueda 9.6. Algoritmos de ordenación 9.7. Estructuras anidadas 9.8. Implementación de TAD lineales 9.9. Aplicación en C++
10. Gestión dinámica de la memoria
10.1. Introducción 10.2. Punteros y variables dinámicas 10.3. Estructuras de datos dinámicas 10.4. Implementación de TAD lineales 10.5. Aplicación en C++

Interrelación			
Requisitos (Rq) y redundancias (Rd)		Tema	*Procedencia*
No existen requisitos al ser la primera asignatura de programación de primer curso de la titulación			
Conocimiento del funcionamiento interno del ordenador (correquisito)	Rq	1.2, 1.3,1.4	Introducción a los computadores
Conceptos básicos de programación, metodología y lenguajes de programación	Rd*	Todos	Laboratorio de programación I
Recursividad	Rd*	5	Estructura de datos y algoritmos
Análisis de algoritmos	Rd*	6	Estructura de datos y algoritmos
TAD lineales	Rd*	9.9, 10.4	Laboratorio de programación I Laboratorio de programación II

(Rd*) Todos estos temas se vuelven a tratar con más profundidad o con enfoques complementarios en esas asignaturas, no son meras repeticiones.

IV. Metodología docente y plan de trabajo del estudiante

Actividades de enseñanza-aprendizaje				*Vinculación*	
Descripción y secuenciación de actividades	*Tipo*[ii]		*D*[iii]	*Tema*	*Objet.*
	grupo	Carácter			
1. Presentación del Plan docente de la asignatura	GG	C-E (I)	0,6	Todos	
2. Encuesta sobre la procedencia de los alumnos y sus intereses	GG	C-E (I)	0,4		
Tema 1. Introducción a la programación					
3. Lectura previa del resumen del tema	NP	T (II)	0,5	1	1
4. Exposición del mundo de la informática	GG	T (II)	1	1	1,2
5. Búsqueda bibliográfica	NP	T (II)	2	1.5	1,2
6. Iniciación al entorno de programación	S	P (V)	2	1	1,11
Tema 2. Entorno, expresiones y acciones fundamentales					
7. Lectura previa del resumen del tema	NP	T (II)	0,5	2.1., 2.2	3,5
8. Explicación, discusión y ejemplificación en clase	GG	T (II)	2	2.1 - 2.6	3,4,5
9. Resolución de problemas en clase	GG	P (III, IV)	1	2.3 – 2.6	3-6,9
10. Búsqueda bibliográfica y resolución de pequeños problemas	NP	T (II)	2	2.4 – 2.8	3,6,9
11. Preparación guión previo de prácticas	NP	P (IV)	1	2.4 – 2.8	2,3,6
12. Iniciación a un lenguaje de programación mediante ejemplos	S	P (V)	2	2.4 – 2.8	2,3,6
Tema 3. Programación modular					
13. Lectura previa del resumen del tema	NP	T (II)	1	3.1	3-6,9
14. Explicación, discusión y ejemplificación en clase	GG	T (II, III)	2	3.2-3.6	3-6,9
15. Resolución de problemas en clase	GG	P (III, IV)	2	3.3-3.5	3-6,9
16. Resolución de pequeños problemas por grupos	NP	P (III, IV)	3	3.6	7,11
17. Preparación del guión previo de prácticas	NP	P (IV)	1	3	3-6,9
18. Utilización de los conceptos en problemas resueltos con un lenguaje de programación	S	P (V)	2	3.2-3.8	7, 10, 11
19. Realización de cuestionarios de autoevaluación	NP	T-P (I, VI)	1	3	
Tema 4. Programación estructurada					
20. Lectura previa del resumen del tema	NP	T (II)	2	4.1-4.3	9
21. Explicación, discusión y ejemplificación en clase	GG	T (II)	4	4.3-4.5	3,4,5,6
22. Repaso de conceptos teóricos	NP	T (II)	2	4.3-4.5	3,4,5,6
23. Preparación de ejercicios	NP	P (III, IV)	3	4.2-4.5	7,11
24. Resolución de problemas en clase - en grupos	GG	P (III, IV)	4	4.2-4.5	7,11
25. Preparación del guión previo de prácticas	NP	P (IV)	3	4.2 -4.6	3-6,9
26. Utilización de los conceptos en problemas resueltos con un lenguaje de programación	S	P (III, IV)	4	4.6	7,10,11
27. Realización de cuestionarios de autoevaluación	NP	T-P (I, VI)	1	4	3,4,5,6
Tema 5. Recursividad					
28. Lectura previa del resumen del tema	NP	T (II)	1	5.1	3,4
29. Explicación, discusión y ejemplificación en clase	GG	T (I)	1	5.1-5.5	3-5
30. Resolución de problemas en clase	GG	P (III, IV)	1	5.2-5.5	3-5
31. Resolución de pequeños problemas por grupos	NP	P (III, IV)	1	5.2-5.5	7,12
32. Preparación del guión previo de prácticas	NP	P (IV)	1	5.2-5.5	3-6,9
33. Utilización de los conceptos en problemas resueltos con un lenguaje de programación	S	P (V)	2	5.6	7,10,11
34. Realización de cuestionarios de autoevaluación	NP	T-P (I, VI)	2	5	3,4,5,6
Tema 6. Análisis de algoritmos					
35. Lectura previa del resumen del tema	NP	T (II)	1	6.1-6.3	3,4
36. Explicación, discusión y ejemplificación en clase	GG	T (II)	2	6.3-6.4	3-6
37. Resolución de problemas en clase	GG	P (III, IV)	1	6.4	7,11
38. Preparación del guión previo de prácticas	NP	P (IV)	1	6.2-6.4	3-6,9
39. Utilización de los conceptos en problemas resueltos con un lenguaje de programación	S	P (V)	2	6.2-6.4	7,10,11
Tema 7. Tipos Abstractos de Datos (TAD)					
40. Lectura previa del resumen del tema	NP	T (II)	1	7.1-7.3	3,4
41. Explicación, discusión y ejemplificación en clase	GG	T (II)	0,5	7.2-7.4	3-6
42. Preparación de ejercicios	NP	P (III, IV)	1	7.2-7.4	7,11
43. Resolución de problemas en clase	GG	P (III, IV)	1,5	7.4	7,11
44. Preparación del guión previo de prácticas	NP	P (IV)	1	7.2-7.4	3-6,9

45. Utilización de los conceptos en problemas resueltos con un lenguaje de programación	S	P (V)	2	7.2-7.4	7,10,11
46. Realización de cuestionarios de autoevaluación	NP	T-P (I, VI)	1	7	3,4,5,6
Tema 8. Estructura de datos registro					
47. Lectura previa del resumen del tema	NP	T (II)	1	8.1	3,4
48. Explicación, discusión y ejemplificación en clase	GG	T (II,III,IV)	0,5	8.2-8.3	3-6
49. Resolución de problemas en clase	GG	P (III, IV)	0,5	8.2-8.3	3-7
50. Preparación del guión previo de prácticas	NP	P (IV)	1	8.2-8.4	3-6,9
51. Utilización de los conceptos en problemas resueltos con un lenguaje de programación	S	P (IV, V)	2	8.4	7,10,11
52. Realización de cuestionarios de autoevaluación	NP	T-P (I, VI)	1	8	3,4,5,6
Tema 9. Estructura de datos vector					
53. Lectura previa del resumen del tema	NP	T (II)	2	9.1,9.2	3,4
54. Explicación, discusión y ejemplificación en clase	GG	T (II,III,IV)	5	9.3, 9.4	3-5
55. Resolución de problemas en clase	GG	P (III, IV)	2	9.3, 9.4	3-5
56. Preparación de ejercicios	NP	P (III, IV)	1	9.3-9.4	7,11
57. Resolución de pequeños problemas por grupos	NP	P (III, IV)	2	9.3, 9.4	3-5,7,11
58. Preparación del guión previo de prácticas	NP	P (IV)	1	9.3, 9.4	3-6,9
59. Utilización de los conceptos en problemas resueltos con un lenguaje de programación	S	P (IV, V)	2	9.3, 9.4	7,10,11
60. Preparación del examen parcial	NP	C-E (I)	9	1-9.4	
61. Examen parcial TEÓRICO	***GG***	***C-E (I)***	***2***	***1-9.4***	
62. Lectura previa del resumen de los apartados de utilización de TADs estáticos	NP	T (II)	2	9.9 -9.10	3,4
63. Explicación, discusión y ejemplificación en clase	GG	T (II, III)	2	9.9 -9.10	3-5
64. Implementación de TADs estáticos	GG	T (II, III)	3	9.9 -9.10	3-5,7,11
65. Preparación del guión previo de prácticas	NP	P (IV)	1	9.9 -9.10	3-6,9
66. Utilización de los conceptos en problemas resueltos con un lenguaje de programación	S	P (IV, V)	2	9.9 -9.10	7,10,11
67. Realización de cuestionarios de autoevaluación	NP	T-P (I, VI)	1	9.1-4, 9.9-10	3,4,5,6
68. Análisis, diseño e implementación de pequeño programa	***NP***	***P (IV)***	***40***	***2,3-9***	***3,5,8,9***
69. Búsqueda bibliográfica de algoritmos propios de vectores: ordenación y búsqueda	NP	T (II)	3	9.3 - 9.8	3-5
70. Preparación del guión previo de prácticas	NP	P (IV)	1	9.3 - 9.8	3-6,9
71. Utilización de los conceptos en problemas resueltos con un lenguaje de programación	S	P (IV,V)	4	9	7,10,11
72. Evaluación del programa realizado	***S***	***C-E (I)***	***2***	***2,3-9***	
73. Realización de cuestionarios de autoevaluación	NP	T-P (I, VI)	2	9	3,4,5,6
Tema 10. Gestión dinámica de la memoria					
74. Lectura previa del resumen del tema	NP	T (II)	2	10.1	3,4
75. Explicación, discusión y ejemplificación en clase	GG	T (II,III,IV)	5	10.2,10.3	3-7
76. Preparación de ejercicios	NP	P (III, IV)	4	10.2,10.3	7,11
77. Resolución de problemas en clase	GG	P (III, IV)	3	10.2,10.3	5,6,7
78. Preparación del guión previo de prácticas	NP	P (IV)	2	10.4	3-6,9
79. Utilización de los conceptos en problemas resueltos con un lenguaje de programación	S	P (IV)	2	10.4	3-7,10
80. Implementación de TADs lineales con estructuras dinámicas	GG	P (IV, V)	4	10.5	3-7,10
81. Realización de cuestionarios de autoevaluación	NP	T-P (I, VI)	2	10	3,4,5,6
Actividades de cohesión					
82. Preparación de ejercicios	NP	P (III, IV)	3	1,2-10	7,11
83. Resolución de problemas en clase	GG	P (III, IV)	6	1,2-10	5,6,7
Examen final					
84. Estudio y preparación del examen final	NP	C-E (I)	20	1,2-10	Todos
85. Examen final TEÓRICO	***GG***	***C-E (I)***	***3***	***1,2-10***	***Todos***
86. Examen final TEÓRICO - Test de conocimientos	***GG***	***C-E (I)***	***1***	***1,2-11***	***Recuper.***
87. Examen final PRÁCTICO	***GG***	***C-E (I)***	***2***	***1,2-10***	***Recuper.***

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)	60	6		7	68
	Teóricas (II y III)	60	28	23	28	14
	Prácticas (IV, V y VI)	60	26	29	26	69,5
	Subtotal	**60**	**60**	**52**	**61**	**151,5**
Seminario-Laboratorio (6-20 alumnos)	Coordinac./evaluac. (I)	20	2		12	45
	Teóricas (II y III)	20				
	Prácticas (IV, V y VI)	20	28	54	84	21
	Subtotal	**60**	**30**	**54**	**96**	**66**
Tutoría ECTS (1-5 alumnos)	Coordinac./evaluac. (I)	5				
	Teóricas (II y III)	5				
	Prácticas (IV, V y VI)	5				
	Subtotal	**60**	**0**	**0**	**0**	**0**
Tutoría comp. y preparación de ex. (VII)		1		**29**	**30**	
Totales			**90**	**135**	**214**	**152**

En la distribución del tiempo para el alumno se han tenido en cuenta las actividades marcadas en la tabla anterior. Para el cálculo del tiempo que dedica el profesor, se ha estimado un grupo de 60 personas para los grupos grandes, GG, de 20 para los seminarios y de 5 personas para las tutorías.
Para el cálculo de la carga del profesor en horas no presenciales, se ha seguido con los parámetros expuestos en la guía extensa proporcionada por el Vicerrectorado:

- Revisión del plan docente a principio de curso : 2 horas
- Revisar y prepara los resúmenes y otros materiales de prácticas y apoyo: 3 horas
- Completar la evaluación de los trabajos dirigidos o grupos autorizados: 30 minutos por grupo.
- Corregir exámenes: 30 minutos por alumno, teniendo en cuenta que se realiza un examen parcial y un examen final.
- Dos sesiones de revisión de resultado y elaboración de las actas: 2 horas
- Clases de teoría: 30 minutos cada una
- Clases de prácticas: 45 minutos cada una, no repetida.

*Otras consideraciones metodológicas**

Recursos y metodología de trabajo en las actividades presenciales

Ante el número de alumnos matriculados en la titulación, hasta ahora muy elevado, y debido a los pocos recursos con los que cuentan los centros en los que se imparte, no es posible prescindir de la clase magistral como metodología de trabajo.

Buscando la interacción con los alumnos y posibilitar el debate dentro de las clases, se ofrecerá a los alumnos, antes de comenzar cada tema un resumen del mismo. Estos resúmenes deberán ser leídos por el alumno antes de ir a clase, lo que permitirá el debate en el aula, y que las clases puedan centrarse en los conceptos claves.

Al tratarse de una asignatura de primer curso, se tratarán conceptos básicos que deberán ser asimilados para poder cursar adecuadamente el resto de asignaturas de cursos posteriores de la titulación. Estamos, por tanto, en un aprendizaje concéntrico donde cada concepto incluye al anterior y en el que, si cometemos el error de realizar un estudio progresivo es prácticamente imposible el ser capaz de asimilar de forma desordenada o en poco tiempo los conceptos tratados.

La clase expositiva no es suficiente en esta materia. La reflexión, la capacidad de creación y de aportar ideas son características fundamentales y propias de la misma, por ello, la clase de desarrollará en un entorno de continuo debate, de preguntas y respuestas, donde es necesario la participación activa del alumno, contestando a las preguntas formuladas o bien resolviendo los problemas propuestos.

En un grupo grande es fácil que los alumnos eviten la participación activa, por lo que en las actividades dedicadas a la resolución de problemas, se pretende la creación de grupos de trabajo, (de tres o cuatro personas), para que una vez que hayan resuelto problemas

propuestos, un portavoz pueda mostrarlos en clase. Además estas actividades serán consideradas en la evaluación final del alumno.

Con respecto a los seminarios, éstos se desarrollarán en grupos de 20 personas en el aula de informática. Los alumnos se agruparán en equipos de dos personas, que se mantendrá durante todo el cuatrimestre, fomentando así el trabajo en grupo.

Se aplicará una metodología de *aprendizaje por proyectos*, fomentando así una de las capacidades que deben adquirirse: el análisis y la proposición de una solución.

Es imprescindible que estos trabajos se apoyen en un trabajo realizado de forma no presencial, y que será propuesto por el profesor y resuelto parcialmente antes de llegar al aula.

Es importante que el profesor dirija al alumno para ayudarle a encontrar la solución más óptima, pero a medida que el curso avanza, la intervención del profesor será cada vez menor, buscando la creatividad y la autosuficiencia del alumno en la propuesta de soluciones.

El trabajo desarrollado en las aulas se apoyará en el desarrollo parcial de un proyecto de programación. Se le propondrá al alumno un pequeño trabajo práctico que deberá desarrollar en tres fases, coincidentes cada una de ellas con un tema (7, 8 y 9). Cada una de las fases deberá ser integrada con lo hecho anteriormente, además de ser documentada. El desarrollo del proyecto deberá reflejarse en un portafolio personal, disponible a través de la web.

Nos apoyaremos en las nuevas tecnologías para proporcionar apoyo al alumno, por un lado a través de foros y chats, y otras actividades, a través de la plataforma virtual de la UEX. Cada alumno poseerá un proyecto propio, que irá rellenando que tras la realización de una sesión de laboratorio, tendrá que se completada como si de un cuaderno se tratase.

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

Toda la metodología propuesta se basa en un aprendizaje mixto semipresencial, donde la asistencia a las clases de grupo grande, y a los seminarios es obligatoria, pero ambas deben apoyarse en el trabajo no presencial realizado por el alumno, antes y después de ir al aula.

Además del estudio individual, se proponen al alumno una serie de actividades que les ayuden a aumentar su participación y a fomentar el estudio activo y continuo durante toda la duración del curso. Entre estas actividades el alumno contará con un sistema de autoevaluación con cuestionarios on-line sobre cada uno de los temas tratados, que permitirán al alumno comprobar si los conocimientos obtenidos.

A través de la plataforma virtual de la UEX se abrirá la asignatura a la que tendrán acceso todos los alumnos mediante una clave previamente facilitada.

En la plataforma estarán disponibles todos los materiales con los que se vaya a trabajar. Entre estos materiales se incluye los resúmenes de cada uno de los temas, el calendario previsto para la asignatura, así como los portafolios, cada uno de los cuestionarios de revisión y el resto de actividades previstas.

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

Se propondrán ejercicios y actividades complementarias que podrán desarrollar los alumnos que no vayan adquiriendo las competencias y conocimientos necesarios. También se les dará apoyo individual en las tutorías complementarias de la asignatura.

Además, tendrán pruebas de recuperación al final de curso de dos actividades importantes, con la intención de que con el tiempo y trabajo adicional hayan adquirido los conocimientos y competencias que no demostraron en las primeras actividades de evaluación.

Recursos y metodología de trabajo para desarrollar competencias transversales

Se incluirán documentos y referencias a sitios web donde se darán las pautas necesarias para desarrollar competencias transversales como la escritura de documentos técnicos, búsqueda de información, trabajo en grupo, etc.

Además, se incluirán en las clases (tanto de grupo grande como de laboratorio), técnicas docentes que ayuden a desarrollar esas competencias.

V. Evaluación

*Criterios de evaluación**	*Vinculación**	
Descripción	*Objetivo*	*CC*[iv]
Ser capaz de analizar un problema y aportar una solución correcta	4, 5, 7, 9	40%
Capacidad de evaluación y emisión de juicios de valor sobre algoritmos propuestos y escritos	4, 6	10%
Capacidad de escribir ordenada y claramente sus ideas	8	15%
Manejo del software elegido para edición, creación y depuración de programas	3	10%
Capacidad de escribir un programa que satisfaga los requerimientos	5, 6, 10	20%
Trabajo en grupo	11	5%
		100%

Actividades e instrumentos de evaluación		
Laboratorio	Elaboración de un portafolio de las sesiones de laboratorio	15% NR
Laboratorio	Elaboración de un proyecto de programación en varias fases, con la documentación generada y una prueba de modificación del proyecto (en la recuperación de la actividad, se podrá obtener, como máximo, 2/3 de la nota máxima).	20% R
Examen parcial	Se tratarán los contenidos desarrollados en las primeras 9 semanas del curso, usando preguntas de test o respuestas cortas y resolución de problemas. Para superarlo debe obtenerse un mínimo de 6 sobre 10. (La recuperación de esta actividad se realizará junto con el examen final, mediante un test, y se deberá sacar un mínimo de 5 sobre 10.)	20% R
Examen final	Se integrarán en este examen todos los contenidos de las asignaturas y las competencias trabajadas. Se propondrán varios problemas.	35% R
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 (como mínimo, 15): 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, pudiendo conseguirse un máximo de 10.	10% NR

VI. Bibliografía

Bibliografía de apoyo seleccionada

- Walter Savitch , /Resolución de problemas con C++. 2ª edición. /Prentice Hall, 2000
- Luis Joyanes, /Fundamentos de programación. Algoritmos, estructuras de datos y objetos. 3ª edición. / McGraw-Hill, 2003
- Luis Joyanes, /Fundamentos de programación.Libro de problemas. 2ª edición. /McGraw-Hill, 2003
- H.M. Deitel y P.J. Deitel, /Como programar en C/C++. 4ª ed./ Prentice Hall, 2003
- H. Schildt, /C++ Manual de referencia/. McGraw-Hill, 95.
- J. Castro y otros, /Curso de programación/. McGraw-Hill, 93
- Luis Joyanes, /Programación en C++. Algoritmos, estructuras de datos y objetos. /McGraw-Hill, 2000

Bibliografía o documentación de lectura obligatoria

Resumen de cada uno de los temas elaborados por el profesor y publicados en la web de la asignatura.

Además, aparecerán en la web enlaces a distintos documentos que expliquen y completen los distintos temas.

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

Tema 1. Introducción a la programación

[Horw89] **"Data Structures in Pascal". Horowitz E. Sahni, S.**
Ed. Computer Science Press, 1986. 2ª Edición
Chapter 1. Introduction. Pág 1 – 30

[Sav00] **"Resolución de problemas con C++." Savitch W.**
Ed. Prentice-Hall, 2000. 2ª Edición
Capítulo 1. Introducción a las computadoras. Pág 1 – 36

[HIST1] http:// www.monografías.com
Historia de las computadoras
Tendencias de los lenguajes de programación
[HIST2] http://www.cienciasmisticas.com.ar/informatica.ofimatica/historia.index.html
Historia de la informática
[HIST3] http://www.millbury.k12ma.us/%7Ehs/techrepair/gen.html
Computer generation time

Tema 2. Entorno, expresiones y acciones fundamentales

[Cas93] **"Curso de programación". Castro J., Cucker F., Messeguer X., Rubio A., Solano L., Balles B. Ed. McGraw-Hill, 1993**
Capítulo 1. Estructuras y algoritmos básicos. Pág. 1-38

[Gar98] **"Aprenda ANSI C como si estuviese en primero". García de Jalón J, Rodríguez J.I., Serie Aprenda Informática. Colección Aprenda…como si estuviese en primero. Ed. Universidad de Navarra, 1998.**
Capítulo 1. INTRODUCCIÓN. Pág. 1-12
2. TIPOS DE DATOS FUNDAMENTALES. VARIABLES. Pág. 13-20
3. CONSTANTES. Pág.21-24
4. OPERADORES, EXPRESIONES Y SENTENCIAS. Pág. 25-32

[Joy00] **"Problemas de metodología de la programación". Joyanes L. Serie Computación. La enseñanza del futuro Ed. McGraw-Hill, 2000**
Capítulo 1. Algoritmos y programas. Pág. 1-36
Capítulo 2. Los datos y las operaciones básicas. Pág. 37-70
Capítulo 3. Estructura general de un programa. Pág. 71-114

[Ker91] **"El lenguaje de programación C". Kernigham B, Ritchie D. Ed. Prentice-Hall, 1991, 2ª Edición.**
Capítulo 2. Tipos, operadores y expresiones. Pág. 39-60

[Ker00] **"La práctica de la programación". Kernigham B, Pike R. Ed. Prentice-Hall, 2000.**
Capítulo 1. Estilo. Pág. 1-29

[Sav00] **"Resolución de problemas con C++." Savitch W. Ed. Prentice-Hall, 2000. 2ª Edición**
Capítulo 1. Introducción a las computadoras y a la programación en C++. Pág. 1-36
Capítulo 2. Fundamentos de C++. Pág. 38-104

Tema 3. Programación modular

[Bal93] **"Programación metódica". Balcázar, J. L. Ed. McGraw-Hill, 1993**
Capítulo 1. Especificación y corrección. Pág. 1-20

[Cas93] **"Curso de programación". Castro J., Cucker F., Messeguer X., Rubio A., Solano L., Balles B. Ed. McGraw-Hill, 1993**
Capítulo 3. Tratamiento de datos no elementales. Pág. 71-106

[Joy96] **"Fundamentos de programación. Algoritmos y estructuras de datos". Joyanes L., 2002. Ed. McGraw-Hill, 1996. 2ª Edición**
Capítulo 5. Subprogramas: procedimientos y algoritmos. Pág. 163-202

[Joy01a] **"Metodología, estructuras de datos y objetos. Programación en C". Joyanes L., Zahonero I. Ed. McGraw-Hill, 2002**
Capítulo 7. Funciones. Pág. 208 - 257

[Ker00] **"La práctica de la programación". Kernigham B., Pike R. Ed. Prentice-Hall, 2000**
Capítulo 5. Depuración. Pág. 117 – 138
Capítulo 6. Pruebas. Pág. 139 – 164

[Llan02] **"Ejercicios de programación creativos y recreativos en C++". Llana G., Martínez, Palao y Pareja. Ed. Prentice-Hall, 2002**

[Sav00] **"Resolución de problemas con C++." Savitch W. Ed. Prentice-Hall, 2000. 2ª Edición**
Capítulo 3. Abstracción de procedimientos y funciones que devuelven un valor.
Capítulo 4. Funciones para todas las subtareas. Pág. 167-214

Tema 4. Programación estructurada

[Cas93] **"Curso de programación". Castro J., Cucker F., Messeguer X., Rubio A., Solano L., Balles B. Ed. McGraw-Hill, 1993**
Capítulo 1. Estructuras y algoritmos básicos. Pág. 1-38

[Joy00] **"Problemas de metodología de la programación". Joyanes L. Serie Computación. La enseñanza del futuro. Ed. McGraw-Hill, 2000**
Capítulo 4. Programación estructurada. Pág. 115-164

[Joy01a] **"Metodología, estructuras de datos y objetos. Programación en C". Joyanes L., Zahonero I. Ed. McGraw-Hill, 2002**
Capítulo 5. Estructuras de selección. Pág. 137-167
Capítulo 6. Estructuras de control: Bucles. Pág. 168-207

[Joy01b] **"Libro de problemas. Programación en C". Joyanes L., Zahonero I.**

Ed. McGraw-Hill, 2001
Capítulo 4. Estructuras de selección. Pág. 41 –56
Capítulo 5. Estructuras de control: bucles. Pág. 57-78

[Llan02] **"Ejercicios de programación creativos y recreativos en C++". Llana G., Martínez, Palao y Pareja. Ed. Prentice-Hall, 2002**

[Sav00] **"Resolución de problemas con C++." Savitch W. Ed. Prentice-Hall, 2000. 2ª Edición**
Capítulo 2. Fundamentos. Pág. 37-104

Tema 5. Recursividad

[Joy96] **"Fundamentos de programación. Algoritmos y estructuras de datos". Joyanes L., 2002. Ed. McGraw-Hill, 1996. 2ª Edición**
Capítulo 5. Subprogramas: procedimientos y algoritmos. Pág. 163-202

[Joy01a] "Metodología, estructuras de datos y objetos. Programación en C". Joyanes L., Zahonero I. Ed. McGraw-Hill, 2002
Capítulo 7. Funciones. Pág. 208 - 257

[Sav00] **"Resolución de problemas con C++." Savitch W. Ed. Prentice-Hall, 2000. 2ª Edición**
Capítulo 12. Recursión. Pág. 701-749

Tema 6. Análisis de algoritmos

[Bras97] "Fundamentos de algoritmia". Brassard G., Bratley P. Ed. Prentice-Hall, 1997
Capítulo 3. Notación asintótica. Pág. 91 - 110

[Ker00] **"La práctica de la programación". Kernigham B., Pike R. Ed. Prentice-Hall, 2000**
Capítulo 2. Algoritmos y estructuras de datos. Pág. 29 - 60

[Peñ98] **"Diseño de programas. Formalismos y abstracción".Ricardo Peña Marí. Ed. Prentice-Hall, 1998. 2ª Edición.**
Capítulo 1. La eficiencia de los algoritmos. Pág. 1-24

[Shif96] **"Data Structures in C++. Including Breadth and Laboratories". Angela B. Shiflet. Ed. West Publishing Company, 1996**
Chapter 2. Fundamentals. Pág 31 – 94

Tema 7. Tipos Abstractos de Datos (TAD)

[Sav00] **"Resolución de problemas con C++." Savitch W. Ed. Prentice-Hall, 2000. 2ª Edición**
Capítulo 6. Definiicón de clases y tipos de datos abstractos. Pág. 291-356

[Shif96] **"Data Structures in C++. Including Breadth and Laboratories". Angela B. Shiflet. Ed. West Publishing Company, 1996**
Chapter 3. Elementary data structures. Pág 95 – 170

[Wirth80] **"Algoritmos + Estructuras de datos = Programas" Wirth N. Ed. Ediciones del Castillo, 1980. 4ª Edición**
Capítulo 1. Estructuras fundamentales de datos. Pág. 1-60

Tema 8. Estructura de datos registro

[Sav00] **"Resolución de problemas con C++." Savitch W. Ed. Prentice-Hall, 2000. 2ª Edición**
Capítulo 6. Definiicón de clases y tipos de datos abstractos. Pág. 291-356

[Shif96] **"Data Structures in C++. Including Breadth and Laboratories". Angela B. Shiflet. Ed. West Publishing Company, 1996**
Chapter 3. Elementary data structures. Pág 95 – 170

[Wirth80] **"Algoritmos + Estructuras de datos = Programas" Wirth N. Ed. Ediciones del Castillo, 1980. 4ª Edición**
Capítulo 1. Estructuras fundamentales de datos. Pág. 1-60

Tema 9. Estructura de datos vector

[Cairó02] **"Estructuras de datos". Cairó O., Guadarti S. Ed. McGraw- Hill, 2002. 2ª Edición**
Capítulo 1. Estructuras fundamentales. Pág 2 –17
Capítulo 2. Representación lineal de estructuras no lineales. Pág 55 –78
Capítulo 8. Métodos de ordenación. Pág. 319-355
Capítulo 9. Métodos de búsqueda. Pág 385 –394

[Joy02] **"Algoritmos , estructuras de datos y objetos. Programación en C++". Joyanes L. Ed. McGraw-Hill, 2002**
Capítulo 7. Arrays. Pág 161 –178
Capítulo 12. Ordenación y búsqueda. Pág 247 –261

[Joy01b] "Libro de problemas. Programación en C". Joyanes L., Zahonero I. Ed. McGraw-Hill, 2001
Capítulo 7. Arrays. Pág 115 –152

[Peñ98] **"Diseño de programas. Formalismos y abstracción".Ricardo Peña Marí. Ed. Prentice-Hall, 1998. 2ª Edición.**
Búsqueda dicotómica. Pág. 77-81
Búsqueda lineal no acotada Pág. 129-130
Búsqueda lineal acotada. Pág. 151

	Quicksort. Pág. 144
	Ordenación por selección. Pág. 2
	Ordenación por inserción. Pág. 21
[Sav00]	**"Resolución de problemas con C++." Savitch W. Ed. Prentice-Hall, 2000. 2ª Edición**
	Capítulo 9. Arreglos. Pág 512 –533
	Capítulo 10. Cadenas y arreglos multidimensionales. Pág 587 –617
[Shif96]	**"Data Structures in C++. Including Breadth and Laboratories". Angela B. Shiflet. Ed. West Publishing Company, 1996**
	Chapter 3. Elementary Data Structures.
	Chapter 9. Sorting. Pág. 599– 672
[EDUPM]	**http://www-oei.eui.upm.es/Asignaturas/ED-I/tablas.zip**
	Estructuras de datos I. Algoritmos de búsqueda. Simulación
	Escuela Universitaria Politécnica de Madrid.
Tema 10. Gestión dinámica de la memoria	
[Deit03]	**"Como programar en C++". Deitel & Deitel. Ed. Pearson Educación, 2003. 4ª Edición**
	Capítulo 5. Apuntadores y cadenas. Pág. 319 – 403
[Joy01a]	**"Metodología, estructuras de datos y objetos. Programación en C". Joyanes L., Zahonero I. Ed. McGraw-Hill, 2002**
	Capítulo 10. Punteros. Pág. 322 - 352
[Joy02]	**"Programación en C++. Algoritmos, estructuras de datos y objetos". L. Joyanes Aguilar. Ed. Prentice-Hall, 2002.**
	Capítulo 9. Punteros. 9.1 – 9.7. Pág 197 – 205
	Capítulo 10. Asignación dinámica de memoria. 10.1 – 10.5. Pág 217 – 223
[Llan02]	**"Ejercicios de programación creativos y recreativos en C++". Llana G., Martínez, Palao y Pareja. Ed. Prentice-Hall, 2002**
	Capítulo 6. Memoria dinámica. Pág. 291 - 364
[Sav00]	**"Resolución de problemas con C++." Savitch W. Ed. Prentice-Hall, 2000. 2ª Edición**
	Capítulo 11. Apuntadores y arreglos dinámicos. Pág. 654 – 668
	Capítulo 14. Apuntadores y listas enlazadas. Pág. 777 – 814
[Shif96]	**"Data Structures in C++. Including Breadth and Laboratories". Angela B. Shiflet. Ed. West Publishing Company, 1996**
	Chapter 3. Elementary Data Structures.

***Códigos*.-**

[i] *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.