Proyecto/Guía docente de la asignatura

Asignatura	PROGRAMACIÓN I		
Materia	PROGRAMACIÓN		
Módulo			
Titulación	GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN		
Plan	727	**Código**	48067
Periodo de impartición	1er CUATRIMESTRE	**Tipo/Carácter**	FORMACIÓN BÁSICA
Nivel/Ciclo	GRADO	**Curso**	1º
Créditos ECTS	6		
Lengua en que se imparte	CASTELLANO		
Profesor/es responsable/s	JOSÉ FERNANDO DÍEZ HIGUERA DAVID GONZÁLEZ ORTEGA		
Datos de contacto (E-mail, teléfono…)	JOSÉ FERNANDO DÍEZ HIGUERA DESPACHO: 2D079 TELÉFONO: 983 18 55 62 E-MAIL: josdie@tel.uva.es DAVID GONZÁLEZ ORTEGA DESPACHO: 2D007 TELÉFONO: 983423000 ext. 5552 E-MAIL: david.gonzalez.ortega@uva.es		
Departamento	TEORÍA DE LA SEÑAL Y COMUNICACIONES E INGENIERÍA TELEMÁTICA		
Fecha de revisión por el Comité de Título	27 de junio de 2025		

1. Situación / Sentido de la Asignatura

1.1. Contextualización

La asignatura *Programación I* es la primera asignatura de naturaleza informática a la que se enfrentan los alumnos del Grado en Ingeniería de Tecnologías de Telecomunicación.

Esta asignatura es una experiencia de aprendizaje diseñada para enseñar a los alumnos cómo desarrollar programas que puedan resolver problemas, convertir datos, almacenar y recuperar información, ayudar a las personas a comunicarse o hacer cualquier cosa que el programador pueda imaginar. En el fondo, esto se hace cuando el programador escribe las instrucciones para el ordenador en un "lenguaje de programación". En otras palabras, los programadores de ordenadores actúan como traductores entre personas y ordenadores, escribiendo las especificaciones de un programa deseado en un lenguaje que el ordenador pueda entender.

1.1.1. ¿Por qué aprender a programar?

Las empresas de telecomunicaciones valoran cada vez más los conocimientos de programación en los ingenieros de telecomunicaciones, ya que el sector está en plena transformación digital. A continuación, se resumen los puntos clave que destacan actualmente:

Conocimientos de programación valorados	♦ *Lenguajes comunes*: C/C++, Python, Java y JavaScript son especialmente útiles para tareas como automatización, análisis de datos, desarrollo de software embebido y aplicaciones web. ♦ *Scripting y automatización*: Bash, PowerShell y Python son esenciales para automatizar tareas de red, mantenimiento de sistemas y pruebas. ♦ *Desarrollo de software y APIs*: Se espera que los ingenieros puedan trabajar con APIs REST, microservicios y herramientas de integración continua. ♦ *Programación orientada a redes*: Conocimientos en SDN (*Software-Defined Networking*) y lenguajes como P4 son cada vez más relevantes. ♦ *Inteligencia Artificial y Big Data*: Las empresas de telecomunicaciones están integrando IA para planificación de redes y atención al cliente, por lo que se valora experiencia en bibliotecas como TensorFlow, PyTorch o herramientas de análisis como Pandas y Spark [1].
Competencias complementarias	♦ *Ciberseguridad*: Programación para detección de intrusos, análisis de tráfico y protección de redes. ♦ *Cloud computing*: Uso de plataformas como AWS, Azure o Google Cloud, y conocimientos en infraestructura como código (IaC) con Terraform o Ansible. ♦ *DevOps*: Integración de conocimientos de programación con herramientas de despliegue y monitoreo continuo.
Tendencias del sector	Según un informe de la consultora EY, las empresas de telecomunicaciones están priorizando la digitalización con tecnologías emergentes como la IA generativa, la automatización y la computación cuántica. Esto implica una creciente necesidad de ingenieros con habilidades de programación para implementar y mantener estas soluciones.
Perfil profesional buscado	Las empresas de telecomunicaciones buscan ingenieros multidisciplinarios, con capacidad de adaptación, pensamiento crítico y habilidades en programación que les permitan: ♦ Automatizar procesos. ♦ Desarrollar soluciones personalizadas. ♦ Integrar sistemas complejos. ♦ Participar en proyectos de transformación digital [2].

1.1.2.¿Por qué empezar con lenguaje C?

Iniciar el aprendizaje de los fundamentos de programación con el **lenguaje C** en titulaciones de ingeniería, y especialmente en **ingeniería de telecomunicaciones**, tiene múltiples argumentos sólidos a favor:

- *Base sólida para comprender cómo funciona un ordenador*: el lenguaje C permite trabajar cerca del *hardware*, lo que ayuda a los estudiantes a entender cómo se gestionan la memoria, los registros, y el procesamiento de instrucciones. Esto es especialmente útil en telecomunicaciones, donde se trabaja con sistemas embebidos, protocolos de red y dispositivos de bajo nivel.
- *Lenguaje fundamental en sistemas embebidos y telecomunicaciones*: Muchos dispositivos de red (*routers*, *switches*, *módems*) y sistemas embebidos están programados en C. Aprender C permite a los futuros ingenieros modificar firmware, optimizar código para hardware específico y desarrollar controladores.

- *Desarrollo del pensamiento algorítmico y estructurado*: C obliga a los estudiantes a pensar de forma lógica y estructurada, sin depender de abstracciones de alto nivel. Esto fortalece la capacidad de resolver problemas complejos, una habilidad clave en cualquier rama de la ingeniería.
- *Transferencia de conocimientos a otros lenguajes*: una vez dominado C, es más fácil aprender otros lenguajes como C++, Java, Python o Rust, ya que muchos conceptos fundamentales (tipos de datos, estructuras de control, funciones) son compartidos.
- *Aplicaciones directas en telecomunicaciones*: protocolos de comunicación, simulaciones de redes, procesamiento de señales y sistemas de tiempo real suelen requerir programación eficiente y de bajo nivel, donde C es ideal. También se utiliza en herramientas como NS-3 (simulador de redes) y en el desarrollo de software para FPGA y DSP.
- *Fomenta la disciplina y el control del código*: C no tiene recolección automática de basura ni manejo automático de errores, lo que obliga a los estudiantes a ser cuidadosos con la gestión de memoria y recursos. Esto desarrolla una mentalidad rigurosa y meticulosa, muy valorada en entornos industriales.
- *Alta demanda en la industria*: empresas de telecomunicaciones y del sector aeroespacial, automoción y electrónica, siguen demandando ingenieros con conocimientos en C para tareas críticas.

En resumen, aprender C en una titulación de ingeniería no es solo aprender un lenguaje de programación, es adquirir una **comprensión profunda de los fundamentos de la computación y de cómo construir sistemas eficientes y robustos**. Proporciona una ventaja significativa en el mercado laboral y prepara al alumno para enfrentar desafíos técnicos complejos.

1.1.3. Referencias

[1] https://www.ey.com/es_es/espana-2025/tendencias-corto-plazo/el-sector-telecomunicaciones-2025-nuevas-estrategias-abordar-nuevos-retos.

[2] https://mx.indeed.com/orientacion-profesional/como-encontrar-empleo/campo-laboral-ingeniero-telecomunicaciones.

1.2. Relación con otras materias

Esta asignatura está especialmente relacionada con la asignatura “Programación II”, en la que se aplican las técnicas y procedimientos de una metodología de desarrollo software concreta al análisis y diseño de un sistema software y amplía los conocimientos sobre lenguajes de programación que siguen el paradigma de orientación a objetos. También está muy relacionada con la asignatura “Fundamentos de Ordenadores y Sistemas Operativos”, la cual incluye programación en lenguaje de bajo nivel (ensamblador) y en lenguaje de alto nivel con llamadas al sistema operativo.

1.3. Prerrequisitos

El alumno que curse esta asignatura ha de poseer unos conocimientos básicos de informática a nivel usuario. En lo referente a programación, y siendo la primera asignatura del Grado que aborda dicha materia, se parte de la base de que el alumno no tiene conocimientos previos de la misma.

2. Resultados del proceso de formación y de aprendizaje (RD 822/2021)

2.1. Conocimientos o contenidos (RD822/2021)

- C3. Conocer y aplicar lenguajes de programación para desarrollar aplicaciones software y conocer alguna metodología de ingeniería de software relevante.

2.1. Habilidades o destrezas (RD822/2021)

- HD6 - Capacidad de razonamiento, análisis y síntesis.
- HD7 - Capacidad para relacionar conceptos y adquirir una visión integrada, evitando enfoques fragmentarios.
- HD8 - Capacidad de toma de decisiones en la resolución de problemas básicos de ingeniería de telecomunicación, así como identificación y formulación de los mismos.
- HD9 - Capacidad para trabajar en grupo, participando de forma activa, colaborando con sus compañeros y trabajando de forma orientada al resultado conjunto, y en un entorno multilingüe.
- HD10 - Conocimiento de materias básicas, científicas y tecnologías, que le capacite para el aprendizaje de nuevos métodos y tecnologías.
- HD15 - Capacidad para resolver problemas con iniciativa, creatividad y razonamiento crítico.
- HD24 - Capacidad de organización, planificación y gestión del tiempo.
- HD25 - Capacidad para comunicar, tanto por escrito como de forma oral, conocimientos, procedimientos, resultados e ideas relacionadas con las telecomunicaciones y la electrónica.
- HD26 - Capacidad para trabajar en cualquier contexto, individual o en grupo, de aprendizaje o profesional, local o internacional, desde el respeto a los derechos fundamentales, de igualdad de sexo, raza o religión y los principios de accesibilidad universal, así como la cultura de paz.

2.1. Competencias (RD822/2021)

- B2. Conocimientos básicos sobre el uso y programación de los ordenadores, sistemas operativos, bases de datos y programas informáticos con aplicación en ingeniería.

3. Objetivos

Al finalizar la asignatura completa, el alumno deberá ser capaz de:

- Codificar y probar algoritmos, aplicando técnicas de programación orientada a procesos (nivel básico) y a datos (nivel avanzado).
- **Dominar la sintaxis y semántica básica del lenguaje C:** Esto incluye variables, tipos de datos, operadores, estructuras de control (condicionales y bucles), funciones y manejo de entrada/salida.
- **Comprender y aplicar conceptos de memoria y punteros:** Entender la asignación de memoria estática y dinámica, el uso de punteros para acceder y manipular datos de forma eficiente, y las implicaciones de seguridad y rendimiento asociadas.
- **Implementar y utilizar estructuras de datos fundamentales:** Desarrollar habilidades para trabajar con vectores, cadenas de caracteres, estructuras (*structs*) y la implementación de estructuras de datos más complejas como listas enlazadas, pilas y colas.
- **Diseñar y desarrollar algoritmos eficientes:** Aplicar principios algorítmicos para resolver problemas computacionales, incluyendo algoritmos de búsqueda y ordenamiento básicos.
- **Utilizar herramientas de desarrollo:** Familiarizarse con entornos de desarrollo integrados (IDEs), compiladores (como GCC) y depuradores para escribir, compilar, ejecutar y depurar programas en C.
- **Desarrollar la capacidad de abstracción y modularización:** Descomponer problemas complejos en módulos más pequeños y manejables utilizando funciones y bibliotecas.

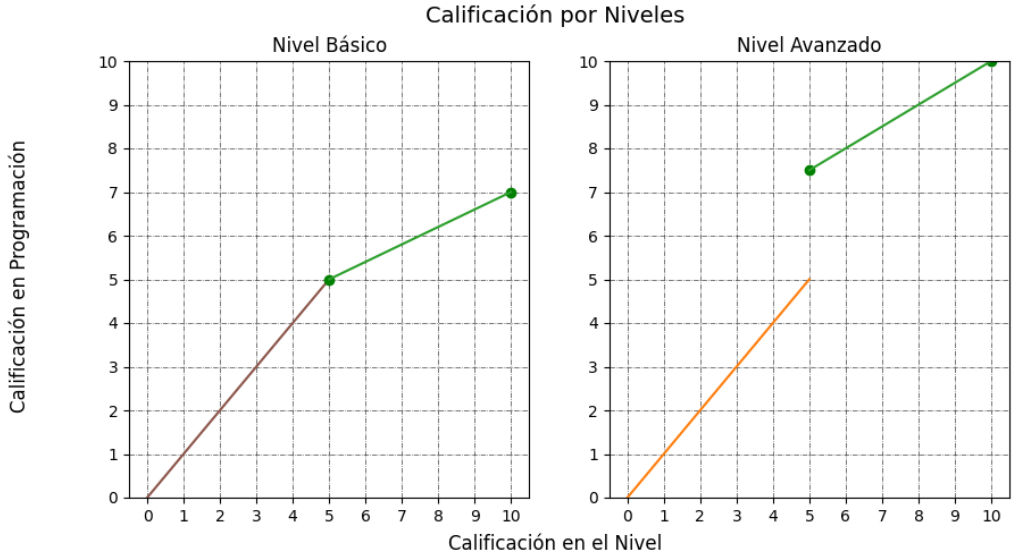
4. Niveles de Aprendizaje

En el presente curso, la asignatura se articula en dos niveles de aprendizaje, a semejanza de los cursos de idiomas, ya que el objetivo principal es el aprendizaje de un lenguaje (de programación): *Nivel Básico* y *Nivel Avanzado*. Con este esquema se pretende que sea el alumno el que decida los conocimientos y habilidades que quiere aprender sobre programación de ordenadores y/o la calificación que desea conseguir para su expediente. En este sentido, las calificaciones que se pueden obtener superando las pruebas de cada uno de los niveles, y la duración de cada nivel, se muestran en la siguiente tabla:

Nivel	Duración	Calificación
Básico	10 semanas	[5.0 – 7.0]
Avanzado	5 semanas	[7.5 - 10.0]

- Al finalizar las semanas correspondientes al nivel básico, se publicarán las calificaciones obtenidas en el nivel básico. Aquellos alumnos que superen este nivel con una nota igual o superior a 5.0 **obtienen una calificación en el intervalo [5.0 – 7.0] como nota de la asignatura**. A partir de ese momento, será potestad de cada alumno decidir si quiere continuar con el nivel avanzado.
- Al finalizar las semanas correspondientes al **nivel avanzado**, se publicarán las calificaciones de aquellos alumnos que hayan decidido cursar el nivel avanzado. Aquellos alumnos que superen este nivel con una nota igual o superior a 5.0 **podrán optar a una calificación entre 7.5 y 10.0**, en función de los resultados de la evaluación de este nivel.

En la siguiente gráfica se pueden observar las escalas de calificaciones correspondientes a cada uno de los niveles de aprendizaje.



NOTA: Como se indica más adelante, en la sección “Sistema y características de la evaluación”, *si un alumno no alcanza los requisitos mínimos de cada elemento de evaluación, su calificación final en la asignatura será el mínimo entre el valor calculado según la ponderación descrita en la tabla y 4.0*.

5. Contenidos y/o bloques temáticos

En consonancia con la sección Niveles de Aprendizaje, los conocimientos y habilidades que el alumno debe adquirir se articulan en dos bloques temáticos, correspondientes a los dos niveles de aprendizaje.

Los objetivos de aprendizaje describen en detalle todo lo que el alumno va a aprender durante este curso. Es importante que el alumno los tenga presentes desde el primer momento, aunque el profesor le irá recordando los objetivos que están implicados en las diferentes actividades del curso.

5.1. Características comunes a todos los bloques

a. Métodos docentes

- Clase expositiva participativa.
- Resolución de problemas.
- Yincanas digitales.

Para más información, consultar la sección “Métodos docentes y principios metodológicos”.

b. Plan de trabajo

El Plan de Trabajo de la asignatura, donde se describe la planificación detallada, se entregará al comienzo de la asignatura.

c. Material docente

La bibliografía propuesta para esta asignatura tiene como objeto servir de soporte al aprendizaje de los conceptos y del lenguaje de programación de la asignatura, por lo que estudiarse alguno (¡o todos!) los libros, no tiene sentido por sí mismo.

Sin embargo, recomendamos el uso de alguno de los libros mencionados en el enlace a la Plataforma Leganto de la Biblioteca de la UVA con la bibliografía recomendada (en el orden en el que aparecen). Existen ejemplares en la biblioteca y todos ellos tratan el mismo tema, con diferentes enfoques, ejemplos y orden de exposición.

§ Otros recursos telemáticos

Los recursos documentales, audiovisuales y de hipermedia elaborados para la implantación de la clase inversa (*flipped classroom*), se enlazarán directamente en las secciones correspondientes del Campus Virtual.

d. Recursos necesarios

- Aula con proyector multimedia y pizarra para sesiones de discusión.
- Laboratorio de prácticas, con un ordenador por alumno, para las sesiones de laboratorio. Cada ordenador debe contar con el entorno de desarrollo para el lenguaje C que se va a utilizar.
- Plataforma educativa para publicar material didáctico, guías de ejercicios, soluciones, tareas, etc.

En esta asignatura se utilizan herramientas docentes online para la docencia y la evaluación. En caso de un transcurso normal de la docencia estarán disponibles las aulas informáticas del centro. En caso de una afección por medidas sanitarias especiales, el alumno debe contar con medios informáticos y telemáticos suficientes para interactuar con el Campus Virtual y con los sistemas de videoconferencia.

5.2. Temporalización

BLOQUE TEMÁTICO	CARGA ECTS	PERIODO PREVISTO DE DESARROLLO
Bloque 1: Nivel Básico	**4.0**	**Semanas 1 a 10**
Bloque 2: Nivel Avanzado	**2.0**	**Semanas 11 a 15**

5.3. Desglose por bloques

Bloque 1: Nivel Básico

Carga de trabajo en créditos ECTS: 4.0

a. Contextualización y justificación

En este primer bloque se introducen los conceptos básicos sobre los diferentes métodos para el desarrollo de aplicaciones informáticas, se presenta C, el lenguaje de programación que se utilizará para la implantación de los programas desarrollados, y se proporcionan los elementos básicos de la **programación orientada a procesos.**

b. Objetivos de aprendizaje

Al final de este bloque el alumno será capaz de:

Concepto	Habilidades
Entorno de desarrollo	• Definir los conceptos de compilación, construcción y ejecución. • Realizar las operaciones necesarias para crear/abrir un proyecto y añadir y eliminar elementos a un proyecto. • Diseñar el archivo *makefile* para una aplicación determinada. • Realizar las operaciones necesarias para editar, compilar, montar y ejecutar un programa, y localizar las carpetas donde están los archivos generados en cada uno de los pasos. • Interpretar adecuadamente los mensajes de error de compilación y corregir el error de compilación correspondiente. • Describir las funcionalidades básicas del *depurador*. • Realizar correctamente las operaciones básicas del depurador (insertar un punto de parada, ejecutar paso a paso y visualizar valores de variables). • Identificar y subsanar los errores de ejecución de un programa, utilizando adecuadamente el depurador.
Tipos de datos y **operaciones**	• Escribir los tipos de datos elementales y las operaciones que actúan sobre ellos. • Escribir la declaración de datos de cualquiera de los tipos elementales. • Indicar el código ASCII de cualquier carácter, con la ayuda de la tabla correspondiente. • Comprender el concepto de puntero y sus diferencias con una variable simple. • Escribir el código necesario para acceder al dato apuntado por un puntero.
Sentencias básicas de asignación y de **entrada/salida**	• Describir el funcionamiento de las sentencias básicas. • Predecir el resultado de una secuencia de sentencias básicas. • Codificar una tarea convenientemente especificada, utilizando la secuencia de sentencias básicas adecuada.
Archivos de texto	• Explicar el concepto de archivo, para qué sirve, y cuáles son las operaciones típicas sobre archivos de texto (crear, abrir, leer, escribir, preguntar por fin de archivo y cerrar). • Escribir las sentencias necesarias para realizar las operaciones básicas con un archivo de texto. • Escribir las sentencias necesarias para determinar el tipo de error que se ha producido al realizar una operación con un archivo de texto.
Archivos binarios	• Escribir las sentencias necesarias para realizar las operaciones básicas con archivos binarios. • Escribir las sentencias necesarias para determinar el tipo de error que se ha producido al realizar una operación con un archivo binario.
Sentencias de control	• Conocer y usar los operadores que se utilizan para construir expresiones lógicas.

	• Usar las sentencias de selección `if`, `if-else` para elegir entre varias acciones alternativas. • Utilizar la sentencia de selección múltiple `switch` para escoger entre muchas alternativas de acción. • Utilizar las etiquetas `case` para identificar las acciones alternativas en las sentencias `switch`. • Ejecutar sentencias repetidamente con la sentencia repetitiva `for`. • Usar la sentencia de repetición `while` para ejecutar sentencias repetidamente.
Funciones	• Definir el concepto de función y su utilidad. • Definir los conceptos: cabecera de función, parámetros formales, variables locales, resultado de la función, llamada a función, parámetros reales, paso de parámetros. • Describir la diferencia entre paso de parámetros por valor o por referencia. • Codificar convenientemente una llamada a función, pasando correctamente los parámetros. • Codificar en forma de función una tarea convenientemente especificada, estableciendo adecuadamente los parámetros necesarios. • Agrupar las funciones de un programa relacionadas entre sí en un módulo.
Estructuras estáticas de datos	• Describir las estructuras de datos fundamentales, y las operaciones típicas sobre ellas. • Escribir la declaración de una estructura de datos convenientemente especificada. • Escribir el código necesario para acceder a un elemento o conjunto de elementos de una estructura de datos. • Elegir la estructura de datos más adecuada para una aplicación determinada.

c. Contenidos

- Elementos básicos del lenguaje C.
- Anatomía de un programa en C.
- Funciones.
- Sentencias de control.
- Vectores numéricos y cadenas de caracteres.
- Algoritmos de recorrido y búsqueda.
- Estructuras estáticas de datos.
  - Registros, uniones, enumeraciones, campos de bits.

Bloque 2: Nivel Avanzado

Carga de trabajo en créditos ECTS: 2.0

a. Contextualización y justificación

El bloque 2 se centra en la introducción de los conceptos relacionados con la **programación orientada a datos dinámicos**: tipos de datos estructurados, registros, estructuras de datos dinámicas, y asignación dinámica de memoria, vectores dinámicos, estructuras dinámicas y estructuras autorreferenciadas.

b. Objetivos de aprendizaje

Al final de este bloque el alumno será capaz de:

Concepto	Habilidades y conocimientos
Estructuras dinámicas de datos	• Diferenciar claramente entre la asignación de memoria **estática** (en tiempo de compilación) y **dinámica** (en tiempo de ejecución). • Dominar las funciones de gestión de memoria dinámica en C. • Trabajar con punteros para acceder y manipular memoria dinámica. • Implementar y gestionar vectores dinámicos (uni- y multidimensionales). • Manejar estructuras de datos con asignación dinámica de memoria.

	• Desarrollar buenas prácticas en la gestión de memoria.
Estructuras de datos autorreferenciadas	• Definir qué son las estructuras de datos autorreferenciadas y explicar su importancia en la gestión dinámica de la memoria y en la resolución de problemas donde el tamaño de los datos no se conoce de antemano. • Comprender la relación entre nodos y punteros en la construcción de estructuras autorreferenciadas, y cómo los punteros son esenciales para "enlazar" elementos entre sí. • Identificar las ventajas y desventajas de usar estructuras autorreferenciadas frente a arreglos estáticos, considerando aspectos como la flexibilidad, el uso de memoria y la complejidad de implementación.

c. Contenidos

- Aritmética de punteros.
- Asignación dinámica de memoria.
- Estructuras dinámicas de datos.
  - Vectores dinámicos.
  - Registros dinámicos.
  - Estructuras autorreferenciadas: listas enlazadas, pilas y colas.

6. Métodos docentes y principios metodológicos

Siguiendo una de las recomendaciones de la Dirección de la ETSIT en cursos anteriores, el flujo de trabajo semanal de esta asignatura está edificado sobre los cimientos del modelo pedagógico denominado *Flipped Classroom* (FC), complementado con técnicas de *Just-in-time Teaching y ConcepTest.*

- *Flipped Classroom* (o *clase inversa*) es "un modelo pedagógico que transfiere el trabajo de determinados procesos de aprendizaje fuera del aula y utiliza el tiempo de clase, junto con la experiencia del docente, para facilitar y potenciar otros procesos de adquisición y práctica de conocimientos dentro del aula".
- Cuestionarios *Just-in-time (JIT) Teaching (o enseñanza justo a tiempo)* son una estrategia pedagógica que utiliza la retroalimentación entre las actividades de aula y el trabajo que el alumnado hace en casa para preparar las sesiones que se dan en el aula. Los objetivos son: aumentar el aprendizaje durante el tiempo que se está en el aula, fomentar la motivación de los alumnos, animar a los estudiantes a que se preparen las clases y permitir al profesor encontrar las actividades más adecuadas para las necesidades de sus estudiantes.
- Cuestionarios *ConcepTests* (o "pruebas de concepto" en castellano) son una *estrategia de enseñanza interactiva* diseñada para *evaluar rápidamente la comprensión de los estudiantes* sobre un concepto específico durante una clase. Fueron popularizados por el físico Eric Mazur en la Universidad de Harvard como parte de su método de "instrucción por pares" (*Peer Instruction*).

Como ya se ha comentado en la sección 4, la asignatura se estructura en niveles de aprendizaje. Cada nivel se compone de varias etapas (por lo general, de **dos semanas** de duración), que a su vez se articulan en una serie de tramos de diferentes actividades. A continuación, se describe la hoja de ruta de una etapa estándar (sin jornadas no lectivas), con la enumeración de los tramos a cubrir.

I. Briefing

Tipo de actividad: no presencial

Al comienzo de cada tramo, y a través del Campus Virtual de la asignatura, el profesor proporciona a los alumnos la hoja de ruta correspondiente a dicho tramo, con la siguiente información:

- Fecha de inicio y final de tramo.
- Objetivos de aprendizaje.
- Enlaces a los recursos documentales necesarios para conseguir los objetivos de aprendizaje previstos.
- Actividades presenciales y de trabajo individual no presencial a realizar durante el tramo, indicando cuáles de ellas son *formativas*, y cuáles tendrán una calificación *sumativa*.
- Criterios específicos de calificación de las actividades evaluables.

II. Periplo a través de los recursos

Tipo de actividad: no presencial	*No evaluable*

En este tramo el alumno debe utilizar los recursos proporcionados por el profesor, y/o leer PDFs interactivos, tutoriales y otros tipos de documentación escrita, relacionados con el material que se abordará durante la próxima sesión de clase.

III. Aplicación de la enseñanza *Just-in-time*

Tipo de actividad: no presencial	Tipo de evaluación: diagnóstica y formativa

Para esta actividad el profesor habrá desarrollado previamente un conjunto de preguntas efectivas que se publicarán en el Campus Virtual para que los estudiantes respondan antes de la siguiente sesión de teoría. Estas preguntas son generalmente de opción múltiple, y requieren que los estudiantes hayan completado el tramo anterior. Después de la fecha límite de publicación, pero siempre antes de que comience la clase, el profesor examina las respuestas de los estudiantes. En la mayoría de los casos, el profesor puede utilizar esta revisión para hacer ajustes en las actividades de aula previstas. Si el profesor cree que el alumnado ha dominado un tema, puede reducir o eliminar la discusión de dicho tema durante la clase. De igual manera, si las respuestas muestran que los alumnos tienen dificultades concretas, estas serán vistas durante la clase con más detenimiento.

IV. Sesión de clase

A. Procedimiento de desarrollo de la sesión

En circunstancias normales, y atendiendo a las características específicas de esta asignatura, la sesión de clase en aula se desarrolla mediante el siguiente procedimiento: el profesor conecta su portátil al proyector del aula, con el fin de que todos los alumnos puedan visualizar en la pantalla del aula una presentación (*PowerPoint, Google Colab, H5P, etc.*), el desarrollo de un ejercicio en el portátil del profesor utilizando el entorno de desarrollo del lenguaje de programación utilizado, y cualquier otro documento o aplicación que el profesor desee mostrar. En otras palabras, el profesor **comparte** la pantalla de su portátil con los alumnos a través del proyector y la pantalla de proyección instalados en el aula. Con cierta frecuencia, el profesor puede 'invitar' a algunos alumnos a salir a la pizarra para realizar un ejercicio, lo que en programación se traduce en utilizar el ordenador del profesor para editar el código, de forma que el resto de la clase pueda seguir el desarrollo, sugiriendo soluciones, detectando errores en el código, etc., en lo que se podría calificar como una actividad de *aprendizaje cooperativo*, ya que todos los alumnos cooperan en la resolución del problema.

B. Desarrollo de la sesión

Como se ha comentado en la sección III, el profesor, después de analizar los resultados de la enseñanza *Just-in-time*, habrá planificado la sesión con recursos y actividades que profundicen en conceptos deficientemente asimilados por los alumnos. Una vez desplegados dichos recursos, el profesor utilizará la técnica de *ConcepTest* para recabar nueva información sobre la asimilación grupal de conceptos durante la sesión (evaluación diagnóstica), con aplicaciones del tipo **WooClap**, *Kahoot*, *Socrative*, etc., y al final de cada sesión (evaluación sumativa), mediante un cuestionario de Moodle realizado con la app para móviles de dicha plataforma (se plantea esta alternativa ya que es posible que algún alumno no disponga de portátil, pero es casi seguro que todos tienen un teléfono móvil).

V. Sesión de laboratorio

A. Procedimiento de desarrollo de la sesión

Las sesiones de laboratorio se emplearán fundamentalmente en aplicar el aprendizaje basado en la resolución de problemas, en el que se incluye el formato de yincana *digital*. En una yincana digital, los alumnos tienen que ir superando una serie de etapas, hasta llegar al ***trofeo*** que les identifica como ganadores. Una vez conseguida una posible solución, se introducirá en un programa proporcionado por el profesor (*unit-test*, *stubs & drivers, etc.*) con el siguiente resultado:

- Si la solución es correcta, le proporcionará el recurso necesario para acceder a las especificaciones de la siguiente etapa.
- Si la solución no es correcta, se lo indicará al alumno, para que revise sus resultados.

Las sesiones serán de dos tipos:

- **Formativas**: los alumnos contarán con el apoyo del profesor. El objetivo de estas sesiones es que los alumnos vayan adquiriendo progresivamente los conocimientos y habilidades de cada nivel de aprendizaje, siguiendo la modalidad "*learning by doing*". Como ya postulaba Aristóteles, "lo que tenemos que aprender a hacer, lo aprendemos haciendo".

- **Sumativas**: los alumnos tienen que realizar los programas correspondientes de forma autónoma. El objetivo de estas sesiones es evaluar el grado de adquisición de conocimientos y habilidades de cada alumno.

Los resultados de ambos tipos de sesiones se tendrán en cuenta en la calificación de las prácticas de laboratorio.

Además, al comienzo de cada sesión de laboratorio se realizarán los **cuestionarios de conocimientos básicos**.

7. Tabla de dedicación del estudiantado a la asignatura

ACTIVIDADES PRESENCIALES o PRESENCIALES A DISTANCIA (1)	HORAS	ACTIVIDADES NO PRESENCIALES	HORAS
Sesiones de aula	20	Visionado y/o lectura de recursos didácticos	35
Resolución de problemas en laboratorio	40	Cuestionarios de enseñanza JIT	10
		Cuestionarios de Aprendizaje	20
		Estudio autónomo	25
Total presencial	**60**	Total no presencial	**90**
		TOTAL presencial + no presencial	**150**

(1) Actividad presencial a distancia es cuando un grupo sentado en un aula del campus sigue una clase por videoconferencia de forma síncrona, impartida por el profesor.

8. Sistema y características de la evaluación

La evaluación de la asignatura se articula en dos elementos: **cuestionarios** y **resolución de problemas en laboratorio**.

INSTRUMENTO/PROCEDIMIENTO	PESO EN LA NOTA FINAL	OBSERVACIONES
TEORÍA: Cuestionarios	35%	Para superar cada cuestionario de conocimientos básicos el alumno dispone de cuatro intentos durante el curso, más un intento adicional en las convocatorias ordinaria y extraordinaria. Para promediar con la nota de prácticas, el alumno debe obtener una calificación mínima en teoría de 5 sobre 10.
PRÁCTICAS: resolución de problemas en laboratorio	65%	Para promediar con la nota de teoría, el alumno debe obtener una calificación mínima en prácticas de 5 sobre 10.
Si un alumno no alcanza los requisitos mínimos descritos en la tabla anterior, su calificación final en la asignatura será el mínimo entre el valor calculado según la ponderación descrita en la tabla y 4.0.		

CRITERIOS DE CALIFICACIÓN
La **Convocatoria Ordinaria** se realiza en la fecha prevista en el calendario de exámenes. Esta convocatoria está indicada para: 1. Alumnos que no han aprobado alguno de los niveles en la Evaluación Continua. 2. Alumnos que, habiendo aprobado alguno de los niveles en la evaluación continua, deseen mejorar su calificación presentándose a un nivel superior al aprobado. En este caso, si un alumno no supera el

nivel al que se presenta en esta convocatoria, mantiene la calificación obtenida en la Evaluación Continua. En cualquier caso, el alumno se puede presentar a cualquiera de los dos niveles, pero solamente a uno de ellos. Solamente se tiene que examinar de los cuestionarios asociados al nivel del que se quiere examinar, y resolver los problemas asociados a dicho nivel.
La **Convocatoria Extraordinaria** se realiza en la fecha prevista en el calendario de exámenes. Esta convocatoria está indicada solamente para alumnos que no han aprobado ninguno de los niveles en la Evaluación Continua ni en la Convocatoria Ordinaria. El alumno se puede presentar a cualquiera de los dos niveles, pero solamente a uno de ellos. Solamente se tiene que examinar de los cuestionarios asociados al nivel del que se quiere examinar, y resolver los problemas asociados a dicho nivel.

El uso de herramientas de Inteligencia Artificial generativa (como ChatGPT, Copilot, Gemini, etc.) está permitido, exclusivamente, como apoyo para aclarar dudas o mejorar la comprensión de conceptos de la asignatura, de forma que ayude en el aprendizaje.

No está permitido, no obstante, utilizar estas herramientas para generar de forma automática programas entregables, o partes de ellos, requeridos en la asignatura.

El incumplimiento de esta norma podrá considerarse una infracción a la honestidad profesional y tendrá una penalización sobre la calificación de hasta el 100% de la nota obtenida.

9. Consideraciones finales