UNIVERSIDAD AUTÓNOMA DE MADRID

# 17821 - PROGRAMACIÓN II

## Información de la asignatura

**Código - Nombre:** 17821 - PROGRAMACIÓN II

**Titulación:** 473 - Graduado/a en Ingeniería Informática
474 - Graduado/a en Ingeniería Informática y Matemáticas
722 - Graduado/a en Ingeniería Informática (Modalidad Bilingüe 2018)
734 - Graduado/a en Ingeniería Informática y Matemáticas (2019)

**Centro:** 350 - Escuela Politécnica Superior

**Curso Académico:** 2020/21

## 1. Detalles de la asignatura

### 1.1. Materia

Programación y estructuras de datos

### 1.2. Carácter

Obligatoria

### 1.3. Nivel

Grado (MECES 2)

### 1.4. Curso

1

### 1.5. Semestre

Segundo semestre

### 1.6. Número de créditos ECTS

6.0

### 1.7. Idioma

Español

### 1.9. Recomendaciones

Programación II forma parte de la *Materia 1* del módulo de *Programación y Estructuras de Datos* del plan de estudios. Esta Materia se desglosa en tres asignaturas semestrales que se complementan entre sí: *Programación I*, *Programación II* y *Proyecto de Programación*. La primera de ellas se imparte en el

primer semestre, mientras que las dos últimas lo hacen en el segundo.
Además, los objetivos de la Materia 1 están transversalmente relacionados con la asignatura *Seminario-taller de Software* perteneciente al módulo *Seminarios-taller de Informática* que se imparte en el primer semestre.

Por todo ello, aunque no existen requisitos académicos para cursar la asignatura *Programación II*, es necesario que el alumno disponga de unos conocimientos básicos relativos a los fundamentos de programación y la metodología del diseño de aplicaciones software utilizando el lenguaje C. Esto implica haber superado con éxito las asignaturas de *Programación I* y *Seminario-Taller de Software*.

En concreto, el alumno, antes de cursar la asignatura, debería:

- Ser capaz de diseñar algoritmos y expresarlos en un pseudocódigo independiente de un lenguaje de programación específico. Esto implica:
  - Utilizar correctamente las estructuras de control
  - Agrupar conjuntos de instrucciones algorítmicas de forma coherente mediante funciones y procedimientos.
  - Aplicar los mecanismos básicos de modularidad y ocultación de la información.
- Conocer el funcionamiento de la recursividad y ser capaz de diseñar algoritmos recursivos.
- Ser capaz de transformar un algoritmo expresado en pseudocódigo en un programa C. Esto implica:
  - Conocer la estructura general de un programa en C.
  - Comprender los tipos de datos primitivos de C y utilizar los tipos estructurados del lenguaje (*arrays* y *estructuras)*
  - Utilizar los operadores aritméticos, lógicos, relacionales e incrementales según convenga en las expresiones.
  - Aplicar las estructuras de control y selección de flujo en C.
  - Describir la sintaxis de las funciones básicas de entrada y salida de datos.
  - Hacer uso de los punteros y de la gestión dinámica de memoria.
- Ser capaz de utilizar de manera eficaz y fluida diferentes entornos de programación, incluyendo las herramientas de edición, compilación, ensamblaje y depurado de código.
- Ser capaz de aplicar técnicas elementales de gestión cooperativa de proyectos de software (e.g., gestión de la documentación).

Asimismo, es recomendable que el estudiante disponga de un nivel del idioma inglés que le permita leer la bibliografía de consulta. Finalmente, se requiere iniciativa personal y constancia para el diseño/escritura de programas y la resolución de los ejercicios propuestos durante el curso junto con predisposición y empatía para el trabajo colaborativo en grupo.

### 1.10. Requisitos mínimos de asistencia

**ITINERARIO CON ASISTENCIA OBLIGATORIA A CLASE**
La asistencia es obligatoria al menos en un 85%.

**ITINERARIO SIN ASISTENCIA OBLIGATORIA A CLASE**
La asistencia es muy recomendable.

### 1.11. Coordinador/a de la asignatura

Rosa Maria Carro Salas

https://autoservicio.uam.es/paginas-blancas/

### 1.12. Competencias y resultados del aprendizaje

#### 1.12.1. Competencias

**B4** Conocimientos básicos sobre el uso y programación de los ordenadores, sistemas operativos, bases de datos y programas informáticos con aplicación en ingeniería.
**C3** Capacidad para comprender la importancia de la negociación, los hábitos de trabajo efectivos, el liderazgo y las habilidades de comunicación en todos los entornos de desarrollo de software.
**C4** Capacidad para elaborar el pliego de condiciones técnicas de una instalación informática que cumpla los estándares y normativas vigentes.
**C5** Conocimiento, administración y mantenimiento sistemas, servicios y aplicaciones informáticas.
**C6** Conocimiento y aplicación de los procedimientos algorítmicos básicos de las tecnologías informáticas para diseñar soluciones a problemas, analizando la idoneidad y complejidad de los algoritmos propuestos.
**C7** Conocimiento, diseño y utilización de forma eficiente los tipos y estructuras de datos más adecuados a la resolución de un problema.
**C14** Conocimiento y aplicación de los principios fundamentales y técnicas básicas de la programación paralela, concurrente, distribuida y de tiempo real.

### 1.12.2. Resultados de aprendizaje

Los **resultados del aprendizaje** que el estudiante adquiere con la asignatura *Programación II* son:

- Comprensión de las técnicas de programación utilizando lenguajes de alto nivel.
- Comprensión de los tipos abstractos de datos, así como su implementación en los lenguajes de programación de alto nivel.
- Comprensión de la evolución de la abstracción de datos: desde los tipos abstractos hasta la orientación a objetos.
- Buenos usos de programación.

### 1.12.3. Objetivos de la asignatura

Los **objetivos generales** son que el estudiante al acabar el curso logre:

- Apreciar los beneficios de la abstracción de datos en el desarrollo de algoritmos y aplicaciones software.
- Distinguir entre un tipo abstractos de datos y su implementación.
- Identificar, diseñar e implementar de forma eficaz los tipos abstractos de datos más adecuados para un problema dado.
- Identificar problemas de naturaleza recursiva: diseñar e implementar soluciones adecuadas a los mismos.
- Aplicar la *modularización* como herramienta conceptual para estructurar programas y aplicaciones software.
- Participar activamente en los análisis y discusiones de grupo que se establezcan al hilo del desarrollo de las actividades propuestas, cooperar con otros compañeros en el desarrollo de trabajos conjuntos y comunicar con propiedad y corrección sus reflexiones sobre los resultados de su trabajo y el de sus compañeros.

Estos objetivos se desglosan en un conjunto de **objetivos específicos** para cada una de las unidades didácticas de las que se compone la asignatura PROG2. Es importante que el estudiante los tenga presente desde el primer momento.
Al finalizar la asignatura el estudiante será capaz de:

| 1. **Unidad: Tipos abstractos de datos** |
|---|
| 1.1 Explicar en qué consiste el proceso de abstracción de datos. |
| 1.2 Discernir entre los tipos primitivos y los tipos abstractos de datos. |
| 1.3 Enumerar los beneficios de utilizar la abstracción de datos (junto con la abstracción funcional) en el desarrollo de algoritmos y aplicaciones software. |
| 1.4 Identificar y especificar informalmente los tipos abstractos de datos adecuados para un problema dado. |
| 1.5 Aplicar la funcionalidad de un tipo abstracto en el diseño de algoritmos. |
| 1.6 Discernir entre un tipo abstracto de datos y su implementación o estructura de datos (EdD). |
| 1.7 Evaluar los beneficios de las diferentes implementaciones de un tipo abstracto de datos en función del espacio o tiempo requeridas para cada una de ellas. |
| 1.8 Implementar en lenguaje C tipos abstractos de datos. |

| |
|---|
| 2. **Unidad: Pilas y sus aplicaciones** |
| 2.1 Describir la organización y el acceso a los datos en una Pila.<br>2.2 Describir ejemplos de aplicaciones del TAD Pila.<br>2.3 Especificar informalmente el TAD Pila.<br>2.4 Diferenciar entre el TAD Pila y sus diferentes implementaciones.<br>2.5 Evaluar los beneficios y desventajas de las diferentes implementaciones del TAD Pila.<br>2.6 Codificar en lenguaje C las primitivas del TAD Pila a partir de diferentes estructuras de datos (EdD).<br>2.7 Evaluar una expresión aritmética expresada en notación infija, prefija o postfija.<br>2.8 Expresar en pseudocódigo los diferentes algoritmos para la evaluación y traducción de expresiones aritméticas.<br>2.9 Codificar en lenguaje C los diferentes algoritmos de evaluación y traducción de expresiones aritméticas. |
| 3. **Unidad: Colas y sus aplicaciones** |
| 3.1 Describir la organización y el acceso a los datos en una Cola.<br>3.2 Identificar las diferencias fundamentales entre Pilas y Colas.<br>3.3 Especificar informalmente el TAD Cola.<br>3.4 Aplicar la funcionalidad del TAD Cola para resolver ciertos problemas paradigmáticos.<br>3.5 Evaluar los beneficios y desventajas de las diferentes implementaciones del TAD Cola.<br>3.6 Codificar en lenguaje C las primitivas de una cola utilizando diferentes estructuras de datos. |
| 4. **Unidad: Listas, listas enlazadas** |
| 4.1 Discernir entre una estructura secuencial y una estructura enlazada.<br>4.2 Especificar informalmente el TAD Lista enlazada simple.<br>4.3 Evaluar los beneficios y desventajas de los diferentes tipos de listas enlazadas.<br>4.4 Codificar en lenguaje C mediante arrays las operaciones primitivas de las listas enlazadas.<br>4.5 Codificar en lenguaje C utilizando memoria dinámica las operaciones primitivas de las listas enlazadas.<br>4.6 Evaluar los beneficios y desventajas de las listas enlazadas cuando son utilizadas como estructuras de datos para implementar otros TAD (por ejemplo, conjuntos, pilas o colas).<br>4.7 Implementar pilas y colas utilizando como estructuras de datos diferentes tipos de listas enlazadas. |
| 5. **Unidad: Árboles. Árboles binarios y árboles ordenados** |
| 5.1 Describir las características de las estructuras jerarquizadas.<br>5.2 Enumerar los diferentes tipos de árboles.<br>5.3 Describir los diferentes recorridos en árboles binarios.<br>5.4 Especificar informalmente el TAD árbol binario.<br>5.5 Codificar en C las primitivas del TAD árbol binario a partir de nodos enlazados<br>5.6 Definir árbol binario de búsqueda (ABdB).<br>5.7 Evaluar el rendimiento en tiempo de las operaciones sobre ABdB.<br>5.8 Utilizar ABdB como estructura de datos para implementar diferentes tipos abstractos de datos.<br>5.9 Construir un árbol binario de expresión a partir de una expresión aritmética. |
| 6. **Unidad: Colas de prioridades y montículos** |
| 6.1 Especificar informalmente el TAD cola de prioridad.<br>6.2 Implementar colas de prioridades con listas enlazadas. |

| |
|---|
| 6.3 Describir la estructura de datos montículo binario.<br>6.4 Codificar en el lenguaje C un montículo binario mediante arrays.<br>6.5 Implementar las operaciones básicas de los montículos con complejidad logarítmica.<br>6.6 Ordenar una lista de elementos con el método del montículo (Heap-sort). |
| 7. **Unidad: Recursión** |
| 7.1 Describir el seguimiento de una función que hace una llamada a sí misma.<br>7.2 Diseñar subprogramas (funciones) recursivas.<br>7.3 Demostrar la eficiencia de un algoritmo recursivo mediante inducción.<br>7.4 Aplicar la técnica de divide y vencerás para la resolución de problemas.<br>7.5 Comparar la resolución de un mismo problema por iteración o por recursión.<br>7.6 Eliminar la recursión de un algoritmo recursivo mediante el uso de pilas. |

### 1.13. Contenidos del programa

El *contenido del programa* es la lista organizada de las unidades que son relevantes para conseguir los objetivos de la asignatura. En el caso de la asignatura *Programación II*, el contenido del programa es el siguiente:

**PROGRAMA SINTÉTICO**

UNIDAD 1. Tipos abstractos de datos
UNIDAD 2. Pilas y sus aplicaciones
UNIDAD 3. Colas y sus aplicaciones
UNIDAD 4. Listas, listas enlazadas
UNIDAD 5. Árboles. Árboles binarios y árboles ordenados
UNIDAD 6. Colas de prioridades y montículos
UNIDAD 7. Recursión avanzada

**PROGRAMA DETALLADO**

**1. Tipos abstractos de datos**
1.1 Tipos primitivos de datos
1.2 Tipos abstractos de datos (TAD)
- 1.2.1 El papel de la abstracción: abstracción de datos
- 1.2.2 Beneficios de los TAD
- 1.2.3 Especificación de un TAD
- 1.2.4 Ejemplos de TAD

1.3 Implementación de tipos abstractos de datos
1.4 Modularidad
- 1.4.1 Reglas de modularización en C

1.5 Orientación a objetos
- 1.5.1 Encapsulación y polimorfismo

**2 Pilas y sus aplicaciones**
2.1 Introducción y primeros ejemplos
2.2 Concepto de pila
- 2.2.1 Organización y acceso a datos
- 2.2.2 Operaciones primitivas.

2.3 Implementación del TAD pila
- 2.3.1 Estructuras de datos de pilas basadas en arrays

2.4 Expresiones aritméticas
- 2.4.1 Notación prefija, infija y postfija
- 2.4.2 Evaluación de expresiones aritméticas
- 2.4.3 Traducción de expresiones aritméticas

**3 Colas y sus aplicaciones**
3.1 Introducción y primeros ejemplos

3.2 Concepto de Cola
3.2.1 Organización y acceso a datos
3.2.2 Operaciones primitivas
3.3 Implementación del TAD Cola
3.3.1 Estructuras de datos de colas basadas en arrays
3.3.2 Condiciones de cola llena/cola vacía y aritmética modular
3.4 Aplicaciones del TAD cola

**4 Listas**

4.1 Introducción y primeros ejemplos
4.2 Concepto del TAD Lista enlazada
4.2.1 Organización y acceso a datos
4.2.2 Operaciones primitivas.
4.3 Implementación del TAD Lista enlazada
4.3.1 Implementación estática de listas enlazadas en C
4.3.2 Implementación dinámica de listas enlazadas en C
4.4 Otras estructuras de listas
4.4.1 Listas circulares
4.4.2 Listas doblemente enlazadas
4.5 Listas enlazadas como Estructuras de Datos para Pilas y Colas
4.5.1 Pilas sobre listas enlazadas: análisis e implementación
4.5.2 Colas sobre listas enlazadas: análisis e implementación

**5 Árboles. Árboles Binarios y árboles ordenados**

5.1 Árboles generales
5.1.1 Representación gráfica de un árbol
5.1.2 Terminología fundamental
5.2 Árboles binarios (AB). TAD árbol binario
5.2.1 Equilibrio
5.2.2 AB completos
5.2.3 Especificación del TAD árbol binario
5.3 Recorridos en AB
5.3.1 Recorridos en profundidad (orden previo, orden medio, orden posterior)
5.3.2 Recorrido en anchura
5.4 Implementación de árbol binario
5.4.1 Representación mediante arrays
5.4.2 Representación dinámica de nodos
5.5 Árboles de expresión (ABdE)
5.5.1 Construcción de ABdE
5.6 Árbol binario de búsqueda (ABdB)
5.6.1 Definición de árbol binario de búsqueda
5.6.2 Inserción y búsqueda en ABdB: rendimiento en tiempo
5.6.3 El ABDB como estructura de datos para implementar tipos abstractos de datos

**6 Colas de prioridades y montículos**

6.1 Concepto del TAD Cola de Prioridad
6.2 Implementación de una cola de prioridades
6.3 Montículos
6.3.1 Definición de montículo
6.3.2 Representación de un montículo
6.3.3 Operaciones definidas en un montículo
6.4 Ordenación por montículos

**7 Recursión**

7.1 Definiciones recursivas y procesos recursivos
7.2 Recursión en C
7.3 Escritura de programas recursivos
7.4 Eficiencia de la recursión
7.5 Simulación de la recursión

### 1.14. Referencias de consulta

Para garantizar la adquisición de los contenidos, habilidades y destrezas de PROG2 es necesaria la lectura crítica de los textos de la bibliografía básica y el uso del material electrónico de la asignatura disponible en la plataforma virtual Moodle (http://uam-virtual.es). En esta plataforma se halla la

descripción de las actividades que se realizarán a lo largo del curso junto con su material asociado. Toda esta información se irá incorporando a medida que avance el curso. Es responsabilidad del estudiante consultar de forma periódica la plataforma virtual (una o dos veces por semana).

Bibliografía básica:

- Estructuras de datos con C y C++, Langsam, Augenstein, Tenenbaum, Prentice Hall Hispanoamericana 1997 INF/681.3.01/LAN (Libro de referencia para todas las unidades del curso)

Bibliografía complementaria sobre *Tipos abstractos y estructuras de datos*

- Introduction to algorithms, Cormen, Leiserson, Rivest, The MIT Press 2001. INF/510.5/COR (Consulta para todas las unidades del curso)
- Estructuras de datos y algoritmos, Aho, Hopcroft, Ullman, Addison-Wesley 1998. INF/681.3.01/AHO (Consulta para todas las unidades del curso)
- Algoritmos y estructuras de datos: una perspectiva en C, Joyanes Aguilar L., Zahonero Martínez I., Mc Graw Hill 2003. INF/510.5/JOY (Consulta)
- Data Structures and algorithm analysis in C, Weiss Mark Allen, Addison Wesley-Longman 1998 INF/681.3.01/WEI (Consulta para todas las unidades del curso)

Bibliografía complementaria sobre el lenguaje C

- The C programming language. Brian W. Kernighan, Dennis M. Ritchie. Prentice Hall, 1988. INF/681.3.062-C/KER (Referencia).
- A book on C. Programming in C (fourth edition). Kelley, Al and Pohl, Ira. Addison-Wesley, 2004. INF/681.3.062-C/KEL (Referencia).

Bibliografía complementaria sobre Metodología de Programación

- La práctica de la programación. B.W. Kernighan. Pearson Educación, 2000. INF/681.3.06/KER (Referencia para prácticas).

Bibliografía complementaria sobre el entorno de programación

- Consultar el Tutorial sobre el entorno de programación Visual C++ en Windows (editor de texto, compilador y depurador) para C que se encuentra en la plataforma virtual Moodle (http://moodle.uam.es) del curso *Seminarios-taller de Informática*

## 2. Metodologías docentes y tiempo de trabajo del estudiante

### 2.1. Presencialidad

| | #horas |
|---|---|
| Porcentaje de actividades presenciales (mínimo 33% del total) | 79 |
| Porcentaje de actividades no presenciales | 71 |

### 2.2. Relación de actividades formativas

| Actividades presenciales | Nº horas |
|---|---|
| Clases teóricas en aula | 42 |
| Seminarios | |
| Clases prácticas en aula | |
| Prácticas clínicas | |
| Prácticas con medios informáticos | 26 |
| Prácticas de campo | |
| Prácticas de laboratorio | |
| Prácticas externas y/o practicum | |
| Trabajos académicamente dirigidos | |
| Tutorías | |
| Actividades de evaluación | 11 |
| Otras | 71 |

## 3. Sistemas de evaluación y porcentaje en la calificación final

### 3.1. Convocatoria ordinaria

La asignatura se organiza en clases teóricas, clases de problemas, prácticas de laboratorio y pruebas individualizadas.

En las **clases teóricas** se presentarán los conceptos de manera clara y concisa utilizando para ello las herramientas docentes más adecuadas al alcance del profesor. Para cada tema, el alumno deberá trabajar ciertos contenidos de forma individual, con ayuda del material propuesto, estimulando, de esta forma, su aprendizaje autónomo.

En las **clases de problemas** se presentarán y resolverán problemas de complejidad superior a los planteados en las sesiones teóricas.

En las **prácticas de laboratorio** se realizarán ejercicios relacionados con el temario teórico utilizando el lenguaje de programación C. Se fomentará el aprendizaje cooperativo, inculcando además en los alumnos el sentido ético que debe primar en los estudios universitarios, de forma que eviten comportamientos fraudulentos como la copia de prácticas o el absentismo.

En las **pruebas teóricas individualizadas** (pruebas parciales y prueba final) se evaluará individualmente los conocimientos adquiridos por los estudiantes.

CONVOCATORIA ORDINARIA:

- Para aprobar la asignatura es obligatorio obtener una nota mayor o igual a 5 puntos, tanto en la parte de teoría como en la de práctica. En caso contrario, la nota final que figurará en actas será

  Calificación = min (4.9, 0.4* Prácticas + 0.6*Teoría)

- La calificación correspondiente a la parte de teoría (nota Teoría), se obtiene a partir de las calificaciones obtenidas por el estudiante durante el cuatrimestre. El estudiante puede optar por dos itinerarios diferenciados:

*Itinerario de Evaluación Continua-Teoría (EC).*

En este itinerario se debe realizar:

- Asistencia continuada a las clases de teoría (por encima del 85% de las sesiones programadas).
- Una primera prueba parcial (PL1). Esta prueba se realizará en torno a la 6ª semana de clase. Para superar esta prueba el estudiante deberá obtener una nota >= 5.
- Una segunda prueba parcial (PL2). Esta prueba se realizará al final del cuatrimestre. Sólo podrán presentarse a esta prueba aquellos alumnos que hubiesen superado la PL1. Para superar la PL2 el estudiante deberá obtener una nota >= 5.
- El Examen Final Individual de la asignatura. Incluirá el contenido de todo el curso. El examen Final constará de dos partes (P1 y P2). A la primera parte (P1) deberán presentarse aquellos alumnos que no hayan superado la prueba parcial PL1. A la segunda parte deberán presentarse todos aquellos alumnos que no hayan superado el examen parcial PL2. También podrán presentarse aquellos estudiantes que piensen que pueden obtener una nota mayor de la que obtuvieron en PL1 o PL2. En ese caso, la nota obtenida en la parte o partes correspondientes del examen final anula(n) la(s) obtenida(s) anteriormente.
  La nota final será la media ponderada de las dos partes P1 (o PL1 en caso de haber aprobado esa parte y no haber hecho P1 en el final) y P2 (o PL2 en caso de haber aprobado esa parte y no haber hecho P2 en el final). Para superar la teoría de la asignatura la nota mínima de cada parte deberá ser igual o superior a 5.
- En la modalidad de evaluación continua, el número mínimo de pruebas a las que el estudiante se ha de presentar para recibir una calificación numérica es dos tercios del número total de pruebas (entre teóricas y prácticas). Por debajo de este mínimo, el estudiante recibirá la calificación en actas de "No evaluado". Siempre que se haya presentado a este número mínimo de pruebas, recibirá una calificación numérica.
- En el caso de que el estudiante no cumpla con alguno de los requisitos de la modalidad EC, perderá el derecho a continuar en dicha modalidad y pasará a EF, perdiendo cualquier derecho sobre las calificaciones obtenidas hasta ese momento.

*Itinerario de Evaluación Final Teoría (EF)*

Pertenecen a este itinerario los estudiantes que voluntariamente hayan solicitado su pertenencia al mismo o que, habiendo comenzado en el EC, no hayan cumplido con los requisitos de permanencia en el mismo. En este itinerario:

- No se requiere una cuota de asistencia mínima.
- La calificación del estudiante se obtendrá a partir del resultado obtenido en el Examen Final Individual de la asignatura (ver el apartado correspondiente de la sección de la evaluación continua para la descripción de esta prueba). La prueba final escrita constará de dos partes que deberán aprobarse individualmente.
- Se aconseja asistir a clase y realizar los ejercicios propuestos. El estudiante tendrá acceso a la misma información y recursos facilitados a los estudiantes del itinerario de evaluación continua.

- La calificación correspondiente a la parte de Prácticas es la que resulte de realizar las prácticas programadas en el curso y las actividades que se especifiquen. La normativa específica de las prácticas figurará antes del comienzo del curso en la página de la asignatura en http://moodle.uam.es. El estudiante puede optar por dos itinerarios diferenciados:

I*tinerario de Evaluación Continua-Práctica (EC).*

En este itinerario se debe realizar:

- Asistencia continuada a las clases de prácticas. El estudiante deberá, al menos, asistir al 85% de las sesiones de prácticas. En caso contrario pasará a la modalidad de Evaluación Final Práctica (EF).
- Entregar todo el material en el plazo y condiciones establecido en la normativa específica de prácticas.
- La calificación de las prácticas tendrá en cuenta la calidad de los diseños realizados, el nivel de los resultados obtenidos y la defensa del trabajo realizado ante el profesor.
- La calificación final de prácticas será:

  CFP = Baremación (P1,P2,…,Pn)),
- n = número de prácticas realizadas cada curso. Usualmente n=4.
- La baremación dependerá de la planificación de las prácticas que se propongan en cada curso. En consecuencia, pueden variar de un curso para el siguiente. Se publicará en la página de prácticas correspondiente a la asignatura en Moodle al inicio de cada curso.
- En caso de no cumplir los requisitos mínimos para mantenerse en EC, el estudiante pasará a EF.

I*tinerario de Evaluación Final Práctica (EF)*

- En este caso, el estudiante deberá entregar todas las prácticas realizadas, realizar un control escrito (CF) y, en el caso de que haya superado los requesitos anteriores, realizar una prueba práctica en el laboratorio (PP). En este itinerario, la calificación final de prácticas será:

  CFP = 0,5*CF + 0,5*PP

**ATENCIÓN:** Cualquier acto de copia descubierto a lo largo del curso, tanto en cualquiera de las actividades de teoría como en cualquiera de los apartados de las prácticas, penalizará por igual, tanto a los alumnos que copian como a los copiados. La penalización por copia implica la aplicación de la sanción prevista en la normativa de evaluación de la EPS.

### 3.2. Convocatoria extraordinaria

C*onvocatoria Extraordinaria Teoría*

En la convocatoria extraordinaria **no** se tendrán en cuenta ninguna de las calificaciones parciales que el alumno haya podido obtener en la Convocatoria Ordinaria. La nota correspondiente (Teoría) se determina mediante una prueba final escrita individual.

*Convocatoria Extraordinaria Práctica*

En la convocatoria extraordinaria la evaluación será similar a la que se lleva a cabo en la convocatoria ordinaria para la modalidad de evaluación final prácticas.

- Consideraciones generales:
- El hecho de pasar del itinerario de EC al de EF en teoría o prácticas no implica que se deba pasar a EF en ambas. Es posible seguir el itinerario de EF en una parte de la asignatura (teoría o prácticas) y continuar en EC en la otra, si se cumplen los requisitos indicados.
- Las notas de teoría o prácticas superadas en la convocatoria ordinaria (>=5) se conservan únicamente para la convocatoria extraordinaria del mismo curso académico.

## 4. Cronograma orientativo

En la página http://moodle.uam.es se encuentra la información actualizada de la asignatura. En ella se incluye la información detallada de todas las actividades que se mencionan en este cronograma, tanto la referente a la parte de teoría como a las prácticas. Esta información se incorporará a medida que el curso progrese. Es fundamental que el estudiante consulte con asiduidad (una o dos veces por semana) dicha página.

| Semana | **Contenido** | Horas presenciales | Horas no prese |
|---|---|---|---|
| 1 | Presentación y motivación de la asignatura<br>**Repaso de C.** | 1 | 3<br>Lectura de las<br>de teoría y prá |

| | | | |
|---|---|---|---|
| 2 | **Unidad 1: Tipos abstractos de datos**.<br>- Se examinarán los conceptos de modularidad y abstracción de datos y se discutirán los beneficios de utilizar la abstracción de datos en el desarrollo de algoritmos y aplicaciones software.<br>- El alumno aprenderá a: (i) especificar informalmente un tipo de datos abstractos de dato, (ii) usar los TAD en una aplicación determinada, (iii) distinguir entre un TAD y su implementación y (iv) utilizar los ficheros de cabecera para especificar en lenguaje C un TAD determinado | 3 | 2<br>- Estudio indivi<br>- Lectura del m<br>propuesto Uni |
| 3 | **Unidad 2: Pilas y sus aplicaciones (2.1-2.3).**<br>- Se estudiará pormenorizadamente el TAD Pila y se describirán ejemplos de aplicaciones elementales de las pilas (e.g, inversión de cadenas de caracteres, cambio de base,…).<br>- Se analizarán los beneficios y desventajas de las implementaciones del TAD Pila basadas en memoria estática.<br>- Se codificarán en lenguaje C diferentes implementaciones del TAD pila. | 3 | 2<br>- Estudio indivi<br>-Lectura del ma<br>propuesto Uni |
| 4 | Cont. **Unidad 2: Pilas y sus aplicaciones (2.4)**<br>- Se aplicarán las pilas para resolver algunos problemas “clásicos” (e.g., utilización de las pilas en algoritmos para evaluar y traducir expresiones algebraicas) | 3 | 2<br>- Estudio indivi<br>-Lectura del ma<br>propuesto Uni |
| 5 | **Unidad 3: Colas y sus aplicaciones (3.1-3.4).**<br>- Se estudiará pormenorizadamente el TAD Cola describiendo ejemplos de aplicaciones del mismo.<br>- Se discuten los beneficios y desventajas de las implementaciones del TAD Cola basadas en *arrays* y *arrays* circulares.<br>- Se analizarán diferentes implementaciones del TAD cola y se programarán en lenguaje C | 3 | 2<br>- Estudio indivi<br>- Lectura del m<br>propuesto Uni |
| 6 | **Unidad 4: Listas enlazadas (4.1-4.3.1)**<br>- Se introducirá el concepto de lista secuencial frente al de lista enlazada<br>- Se describirán diferentes Implementaciones de las listas enlazadas: con arrays y utilizando memoria dinámica. | 3 | 2<br>- Estudio indivi<br>- Lectura del m<br>propuesto Uni |
| 7 | Cont. **Unidad 4: Listas enlazadas (4.3.2-4.5)**<br>-Se describirán otros tipos de listas enlazadas (e.g., las listas circulares, y las doblemente enlazadas)<br>- Se analizará la idoneidad de las listas como estructuras de datos para implementar otros TADs (e.g., pilas y colas)<br><u>Parcial Liberatorio (PL)</u> | 3<br>2 | 2<br>- Estudio indivi<br>- Lectura del m<br>propuesto Uni |
| 8 | -Comienzo **Unidad 5: Árboles, árboles ordenados (5.1-5.4)**<br>-Se estudiará en concepto de árbol, su terminología y los recorridos en profundidad y en anchura.<br>-Se analizarán diferentes implementaciones del TAD árbol binario (con arrays y utilizando memoria dinámica) | 3 | 2<br>- Estudio indivi<br>- Lectura del m<br>propuesto Uni |
| 9 | -Cont. **Unidad 5: Árboles, árboles ordenados (5.5, 5.6)**<br>-Se describirán los árboles de expresión como un método eficiente para representar fórmulas algebraicas<br>- Se introducen los árboles binarios de búsqueda y se estudiarán algunas aplicaciones habituales de los mismos. | 3 | 2<br>- Estudio indivi<br>Lectura del ma<br>propuesto Uni |
| 10 | **Unidad 6: Colas de prioridad, montículos (6.1, 6.3.1)**<br>-Se estudiará el TAD colas de prioridades y se presentarán algunas de sus aplicaciones informáticas.<br>-Se introduce el montículo como estructura de datos para implementar las colas de prioridades | 3 | 2<br>Estudio individ<br>Lectura del ma<br>propuesto Uni |

| | | | |
|---|---|---|---|
| 11 | Cont**. 6: Colas de prioridad, montículos (6.3.2-6.4)**<br>-Se analizan el coste de las operaciones básicas (insertar y eliminar mínimo) en un montículo.<br>- Se describe el algoritmo de ordenación Heap-sort | 3 | 2<br>Estudio individ<br>Lectura del ma<br>propuesto Uni |
| 12 | **Unidad 7: Recursión (7.1-7.4)**<br>-Se describirán los pasos que siguen los lenguajes de programación al llamar a un sub-programa<br>-Se compararán las soluciones iterativas y recursivas de diferentes problemas.<br>-Se describirán diferentes técnicas algorítmicas basadas en la recursión | 3 | 2<br>Estudio individ<br>Lectura del ma<br>propuesto Uni |
| 13 | TEORÍA<br>Cont. **Unidad 7: Recursión (7.5)**<br>- Se estudiará un procedimiento algorítmico para eliminar la recursión, simulándola mediante procedimientos iterativos. | 3 | 2<br>Estudio individ<br>Lectura del ma<br>propuesto Uni |
| 14 | **Repaso y preparación del examen**<br>Parcial Liberatorio (PL) | 3<br>2 | 2<br>Estudio individ<br>- |
| | **Examen prácticas**<br>**Examen Final (P1 / P2)** | 4 | 10h |
| | **Examen extraordinaria** | 3 | |