

DESARROLLO CURRICULAR DEL MÓDULO  
FUNDAMENTOS DE PROGRAMACIÓN

CICLO FORMATIVO DE GRADO SUPERIOR  
ADMINISTRACIÓN DE SISTEMAS INFORMÁTICOS



## ÍNDICE DE CONTENIDOS

1. INTRODUCCIÓN
2. OBJETIVOS
3. CAPACIDADES TERMINALES
4. ORGANIZACIÓN DE LOS CONTENIDOS
  - 4.1 ESTRUCTURA DE LOS CONTENIDOS. BLOQUES.  
TEMPORALIZACIÓN DE LOS BLOQUES
  - 4.2 RELACIÓN DE UNIDADES DE TRABAJO
  - 4.3 RELACIÓN ENTRE UNIDADES DE TRABAJO Y CAPACIDADES TERMINALES
  - 4.4 CONTENIDOS MÍNIMOS
5. ELEMENTOS CURRICULARES DE CADA UNIDAD
6. BIBLIOGRAFÍA RECOMENDADA



## 1.- INTRODUCCIÓN.

El módulo de **Fundamentos de Programación** es de 285 *horas totales (9 horas semanales)* que se encuadra en el primer curso del Ciclo formativo de grado superior correspondiente al título de Administración de Sistemas Informáticos.

La referencia del sistema productivo de este Módulo la encontramos en la unidad de competencia número 4 del correspondiente R.D. de título: "**Proponer y coordinar cambios para mejorar la explotación del sistema y las aplicaciones.**"

## 2.- OBJETIVOS.

Los objetivos del módulo son los siguientes:

- Interpretar y aportar soluciones a las necesidades y requerimientos funcionales formulados por los usuarios.
- Aplicar la metodología de desarrollo, elegir una estructura para los datos y codificar el programa en lenguajes estructurados.
- Establecer y aplicar procedimientos que aseguren la integridad, disponibilidad y confidencialidad de la información.
- Definir y proponer cambios y mejoras en el sistema y aplicaciones encaminadas a optimizar las prestaciones, manteniéndose informado de las innovaciones, tendencias, tecnología y normativa aplicable.
- Organizar y dirigir tareas colectivas cooperando en la superación de las dificultades que se presenten, con una actitud tolerante hacia las ideas de los compañeros y subordinados.
- Mantener relaciones fluidas con los miembros del grupo funcional en el que se integre, responsabilizándose de la consecución de los objetivos asignados al grupo

Este módulo pretende conseguir las siguientes realizaciones profesionales:

- ✓ Formular técnicamente los cambios y mejoras necesarios en el sistema y/o aplicaciones para proporcionar criterios de decisión a la persona autorizada.
- ✓ Realizar, a su nivel, los cambios propuestos en el sistema y/o aplicaciones de acuerdo con las prestaciones requeridas.
- ✓ Realizar pruebas funcionales y de usuario previas a la implantación de los cambios desarrollados en el sistema y/o aplicaciones.
- ✓ Elaborar y mantener la documentación y guías del usuario descriptivas de los cambios y mejoras introducidos en el sistema y/o aplicaciones según las normas y procedimientos establecidos.

### **3.- CAPACIDADES TERMINALES.**

Las capacidades terminales asociadas al módulo están determinadas en los proyectos curriculares y son las siguientes:

#### **1.- Elegir y definir una estructura de datos para resolver un problema con lenguajes estructurados.**

##### **Elementos de Capacidad (Criterios de evaluación).**

- Describir las estructuras de datos típicas que maneja un lenguaje estructurado, su utilidad y ámbito de aplicación.
- Citar operaciones que permite realizar una estructura de datos desde un programa y explicar sus algoritmos.
- Justificar la importancia de la adecuada selección de estructuras de datos para la resolución de problemas en programación.
- Sobre un problema de programación en gestión propuesto:
  - ✓ Elegir las estructuras mas adecuadas para representar y manejar los datos del problema.
  - ✓ Describir los algoritmos de tratamiento de las estructuras para la resolución del problema.

#### **2.- Aplicar la metodología de desarrollo estructurado para el diseño de algoritmos.**

##### **Elementos de Capacidad (Criterios de evaluación).**

- Clasificar los lenguajes de programación segun su nivel de abstracción y los recursos y procedimientos de desarrollo utilizados.
- Describir las características propias de la programación estructurada y justificar las ventajas que comporta.
- Identificar las estructuras básicas de programación.
- Definir las condiciones, el modo de aplicación de algún método de programación estructurada y la sintaxis de un lenguaje gráfico de representación de algoritmos.
- Clasificar las instrucciones típicas de los lenguajes estructurados según su función.
- Sobre un problema de programación en gestión propuesto:
  - ✓ Identificar y definir las estructuras de datos propias del problema.
  - ✓ Elaborar y representar un algoritmo aplicando metodos de programación estructurada.
  - ✓ Elaborar un conjunto de datos de prueba de programa diseñado.

#### **3.- Codificar programas en lenguajes estructurados de tercera generación.**

##### **Elementos de Capacidad (Criterios de evaluación).**

- Interpretar la sintaxis del lenguaje y sus instrucciones.
- Definir las instrucciones, funciones y librerías del lenguaje más básicas y su utilidad.
- Describir el entorno de desarrollo del lenguaje: recursos que se utilizan y procedimiento práctico de desarrollo de programas.
- En un supuesto en el que se dispone de un sistema y de la documentación de referencia del lenguaje y un programa ya diseñado que responde a un programa propuesto:
  - ✓ Interpretar correctamente la información que suministran los manuales.
  - ✓ Codificar un programa fuente en el lenguaje con comentarios significativos y concisos, que defina adecuadamente las estructuras de datos y utilice correctamente las instrucciones, funciones y librerías del lenguaje.

- ✓ Depurar el programa fuente y obtener un programa ejecutable.

## **4.- ORGANIZACIÓN DE LOS CONTENIDOS.**

### **4.1.- ESTRUCTURA DE LOS CONTENIDOS. BLOQUES. TEMPORALIZACIÓN DE LOS BLOQUES.**

Teniendo en cuenta la naturaleza del módulo y las capacidades terminales a él ligadas su contenido será de tipo *procedimental* encaminado a conseguir las capacidades terminales del módulo.

La programación está formada por una relación de unidades de trabajo agrupadas bajo los siguientes bloques conceptuales que desarrollan diferentes áreas de programación.

#### **BLOQUES**

1. Metodología de la programación
2. El lenguaje de programación C (inicial)
3. El lenguaje de programación C (avanzado)
4. Mantenimiento de programas

La finalidad de los objetivos que se persiguen con cada bloque son:

##### *Bloque 1: Metodología de la programación*

El objetivo principal es que el alumno adquiera los conceptos básicos de la programación, mediante la adquisición de métodos y técnicas para la resolución de problemas así como las formas descriptoras en forma de pseudocódigo para la resolución de algoritmos que resuelvan esos problemas planteados, siguiendo un diseño modular y estructurado.

##### *Bloque 2: El lenguaje de programación C (inicial)*

Se iniciará el lenguaje relacionando la sintaxis de las estructuras de datos simples, y las sentencias elementales de este lenguaje con respecto al pseudocódigo. El objetivo de este bloque es escribir en C cualquier de los programas vistos en el bloque anterior.

##### *Bloque 3: El lenguaje de programación C (avanzado)*

El objetivo es presentar todo aquello que, considerado como importante para el desarrollo de programas en C, no se ha visto hasta el momento. Permite además aplicar de forma conjunta lo que hasta ahora se ha visto en parcelas independientes, y todo aquello que por su grado de dificultad el profesor ha preferido guardar para cuando el alumno haya adquirido cierta destreza y grado de conocimiento en el manejo del lenguaje C.

##### *Bloque 4: Mantenimiento de programas*

En este bloque se intentará enfrentar al alumno a la creación y/o adaptación de aplicaciones dirigidas a la mejora de programas y/o el sistema. El desarrollo de este bloque dependerá en gran medida del conocimiento que haya adquirido el alumno hasta este momento.

**Temporalización de los Bloques**

La temporalización elegida debe cubrir el número de horas totales del módulo (285h.).

<b>BLOQUE</b>	<b>% Dedicación Mínimo</b>	<b>% Dedicación Máximo</b>
1. METODOLOGÍA DE LA PROGRAMACIÓN	20	35
2. EL LENGUAJE DE PROGRAMACIÓN C (INICIAL)	20	40
3. EL LENGUAJE DE PROGRAMACIÓN C (AVANZADO)	20	40
4. MANTENIMIENTO DE PROGRAMAS	0	10

**4.2.- RELACIÓN DE UNIDADES DE TRABAJO.**

La propuesta de organización de contenidos está constituida por una relación secuenciada de unidades de trabajo donde se integran y desarrollan al mismo tiempo, alrededor de los procedimientos, conceptos, enseñanzas de enseñanza-aprendizaje y criterios de evaluación.

Para el diseño de una programación concreta es necesario tener en cuenta los conocimientos previos del alumno, los recursos materiales del Centro Educativo y los medios utilizados en el entorno productivo.

Del análisis del R.D. 1675/1994, de 22 de julio, por el que se establece el currículo del Ciclo Formativo de grado Superior Administración de Sistemas Informáticos, se pueden considerar 4 grandes bloques, con las siguientes unidades de trabajo:

Los bloques anteriormente descritos estarán constituidos por las siguientes unidades de trabajo:

<b>BLOQUE</b>	<b>UNIDADES DE TRABAJOS</b>
METODOLOGÍA DE LA PROGRAMACIÓN	1. La información y su representación 2. Programación estructurada. 3. Estructuras estáticas de datos 4. Estructuras dinámicas de datos 5. Estructuras externas de datos
EL LENGUAJE DE PROGRAMACIÓN C (INICIAL)	6. Introducción al lenguaje C 7. Arrays y cadenas en C 8. Punteros y asignación dinámica de memoria 9. Funciones
EL LENGUAJE DE PROGRAMACIÓN C (AVANZADO)	10. Estructuras compuestas 11. E/S por ficheros 12. El preprocesador de C 13. Colas, pilas, listas enlazadas y árboles
MANTENIMIENTO DE PROGRAMAS	14. Adaptación y/o creación de aplicaciones y/o funciones sencillas para el sistema

#### 4.3.- RELACIÓN ENTRE LAS CAPACIDADES TERMINALES Y LAS UNIDADES DE TRABAJO.

Desarrollo de las capacidades terminales por Unidades de Trabajo.

	<i>Capacidad terminal 1</i>	<i>Capacidad terminal 2</i>	<i>Capacidad terminal 3</i>
<b>U.T.1</b>			
<b>U.T. 2</b>			
<b>U.T. 3</b>			
<b>U.T. 4</b>			
<b>U.T. 5</b>			
<b>U.T. 6</b>			
<b>U.T. 7</b>			
<b>U.T. 8</b>			
<b>U.T. 9</b>			
<b>U.T. 10</b>			
<b>U.T. 11</b>			
<b>U.T. 12</b>			
<b>U.T. 13</b>			
<b>U.T. 14</b>			

#### 4.4.- CONTENIDOS MÍNIMOS.

Los contenidos mínimos que deben alcanzar los alumnos en el módulo de Fundamentos de Programación están establecidos en el Real Decreto del Título, y su referencia son las capacidades terminales que el alumno debe conseguir y sus correspondientes criterios de evaluación, que marcan los niveles de consecución aceptable de dichas capacidades terminales.

Los alumnos deben ser capaces de resolver cuestiones teóricas y prácticas que indiquen que han adquirido las capacidades terminales. Para ello deben demostrar que son capaces de realizar las actividades de Enseñanza/Aprendizaje y alcanzar los Criterios de Evaluación desarrollados en cada Unidad de Trabajo.

### 5.- ELEMENTOS CURRICULARES DE CADA UNIDAD.

#### **UT.1: La información y su representación.**

La UT.1. tiene como fin presentar al alumno los conceptos básicos de la programación de tal manera que comience a familiarizarse con los términos, entornos, materiales y finalidades del Módulo completo. Es una Unidad eminentemente conceptual que pretende presentar, de forma global, el contenido que se verá durante todo el Módulo.

#### **UT.2: Programación estructurada.**

La UT.2 presenta al alumno los métodos y técnicas que le permitirán desarrollar buenos programas. Esto es la metodología de la programación., **parte fundamental y básica para la asimilación de la lógica de programación**. Asimismo se expondrán las herramientas de diseño de algoritmos así como las pautas que debe seguir en su diseño y las técnicas de programación utilizadas en la actualidad.

#### **UT.3: Estructuras estáticas de datos.**

La UT3. presenta las primeras estructuras complejas de datos: las estructuras internas estáticas. Se enseñara al alumno a aplicar alguna de las herramientas de diseño de algoritmos a este tipo de estructuras .

#### **UT.4: Estructuras dinámicas de datos.**

La UT.4 presenta al alumno las estructuras internas dinámicas de datos. Este tipo de estructuras son características del lenguaje C y le proporcionará una forma muy flexible de gestionar la memoria.

#### **UT.5: Estructuras externas de datos.**

Esta unidad presenta al alumno las estructuras externas de datos. La aplicación de la herramienta de diseño de algoritmos elegida al tratamiento de ficheros, estudio de las características y peculiaridades de los ficheros

#### **UT.6: Introducción al lenguaje C.**

Esta unidad. presenta un lenguaje de programación procedimental estructurado, el C. Se pretende dar una visión general del lenguaje, sus características, utilidad, ventajas, inconvenientes e implantación actual y de la estructura de un programa en C. Asimismo se pretende que el alumno adquiera los suficientes conocimientos sobre el compilador y su entorno de trabajo (editor, depurador, librerías, etc.) como para poder empezar a codificar desde este momento.

En esta unidad también se pretende en su inicio que el alumno descubra los diferentes tipos de datos que se utilizan en C para dedicarse posteriormente a la descripción de los tipos de datos sencillos que maneja el C y su forma de utilizarlos. También se ocupa de las estructuras de programación características del lenguaje de manera que el alumno pueda empezar a resolver problemas sencillos siguiendo siempre los mismos pasos: interpretación de los problemas, diseño del algoritmo, codificación en C, pruebas, depuración y documentación.

#### **UT.7: Arrays y cadenas en C.**

Es fundamental que el alumno las comprenda dada la variedad de estas estructuras en C, su importancia y continua utilización de aquí en adelante. Al finalizar la Unidad el alumno debe encontrarse en situación de poder resolver problemas de una cierta entidad. El contenido de esta Unidad es eminentemente procedimental.

#### **UT.8: Punteros y asignación dinámica de memoria.**

Los punteros tienen una gran utilidad en el lenguaje C y también cierta complejidad conceptual lo que obliga a dedicarle una unidad de trabajo en exclusiva para conocer su uso. Al finalizar la unidad el alumno debe encontrarse en situación de poder manejar punteros evitando los errores habituales que se suelen cometer en su uso

#### **UT.9: Funciones.**

El uso de funciones es una característica peculiar y muy importante en el lenguaje C, pues permite realizar un desarrollo modular del programa al tiempo que facilita su mantenimiento y futuro uso en otras aplicaciones.

#### **UT.10: Estructuras compuestas.**

Las estructuras compuestas son estructuras de datos basadas en otros tipos de datos más simples ya vistos en la parte de metodología. El alumno debe ser capaz de declarar, definir y utilizar las estructuras compuestas.

#### **UT.11: E/S por ficheros.**

El alumno estudiará el manejo y tratamiento de la entrada y salida de la información mediante el uso de ficheros de almacenamiento externo. Al finalizar la Unidad el alumno debe haber adquirido los conocimientos y destrezas necesarios para el manejo de estas estructuras en problemas de gestión.

#### **UT.12: El preprocesador de C.**

El preprocesador de C consta de las llamadas “directivas de compilación” que son ordenes dirigidas al compilador. El alumno debe conocer cuales son las funciones del preprocesador y el uso y sintaxis de las principales directivas.

#### **UT.13: Colas, pilas, listas enlazadas y árboles.**

Forman las estructuras dinámicas de datos cuyo concepto el alumno estudió en la UT.3. En esta unidad se estudiará como mediante el lenguaje C se pueden utilizar este tipo de estructuras.

#### **UT.14: Adaptación y/o creación de aplicaciones y/o funciones sencillas para el sistema.**

Esta unidad trata de enfrentar al alumno con la adaptación de aplicaciones que ya están hechas para satisfacer nuevos requerimientos. En una palabra, se trata de hacer un mantenimiento de aplicaciones informáticas en uso. Para el desarrollo de esta Unidad será definitivo el grado de conocimiento que haya adquirido el alumno hasta este momento, así como la ayuda que pueda prestar el profesor.

<b>UNIDAD DE TRABAJO 1</b> <b>LA INFORMACIÓN Y SU REPRESENTACIÓN.</b>
<b>CONCEPTOS</b>
<ul style="list-style-type: none"><li>• Conceptos básicos sobre programación</li><li>• Evolución y clasificación de los lenguajes</li><li>• Fases de elaboración de un programa informático</li><li>• Ensambladores, compiladores e interpretes</li><li>• Documentación de programas</li><li>• Estructuras básicas de datos</li></ul>
<b>PROCEDIMIENTOS</b>
<ul style="list-style-type: none"><li>• Descripción del ciclo de vida de una aplicación informática</li><li>• Descripción de la evolución de los lenguajes informáticos</li><li>• Ventajas de los lenguajes compilados sobre los interpretados y viceversa</li></ul>
<b>ACTIVIDADES DE ENSEÑANZA-APRENDIZAJE</b>
<ul style="list-style-type: none"><li>• Reconocimiento de los distintos componentes de los Sistemas de Información disponibles</li><li>• Identificación y discusión de las distintas fases del ciclo de vida de una aplicación informática</li><li>• Identificación de los lenguajes interpretados de los lenguajes compilados</li><li>• Reconocimiento de los tipos básicos de datos que componen un programa</li></ul>
<b>CRITERIOS DE EVALUACIÓN</b>
<ul style="list-style-type: none"><li>• Describir los aspectos fundamentales de la programación.</li><li>• Identificar y describir las fases de una aplicación informática.</li><li>• Describir las estructuras de datos típicas que maneja un lenguaje estructurado, su utilidad y ámbito de aplicación.</li><li>• Describir los tipos de lenguajes, compiladores y traductores de uso mas común.</li></ul>

<b>UNIDAD DE TRABAJO 2</b> <b>PROGRAMACIÓN ESTRUCTURADA.</b>
<b>CONCEPTOS</b>
<ul style="list-style-type: none"><li>• Partes de un programa</li><li>• Algoritmos</li><li>• Tipos de instrucciones.</li><li>• Instrucciones de Entrada/Salida</li><li>• Instrucciones de control</li></ul>
<b>PROCEDIMIENTOS</b>
<ul style="list-style-type: none"><li>• Interpretación del problema</li><li>• Elección de las estructuras de programación necesarias para la resolución del problema</li><li>• Construcción del algoritmo utilizando las estructuras elegidas</li><li>• Realización de pruebas</li><li>• Corrección de errores</li><li>• Documentación del programa</li></ul>
<b>ACTIVIDADES DE ENSEÑANZA-APRENDIZAJE</b>
<ul style="list-style-type: none"><li>• Comparación de las distintas herramientas de diseño de algoritmos</li><li>• Aplicación de las herramientas de diseño de algoritmos a la utilización de estructuras estáticas</li><li>• Justificación de la importancia de las estructuras estáticas</li><li>• Planteamiento de ejercicios</li><li>• Resolución de ejercicios en grupo</li><li>• Documentación de los ejercicios resueltos</li></ul>
<b>CRITERIOS DE EVALUACIÓN</b>
<ul style="list-style-type: none"><li>• Identificar las estructuras básicas de programación.</li><li>• Describir las características propias de la programación estructurada y justificar las ventajas que comporta.</li><li>• Clasificar las instrucciones típicas de los lenguajes estructurados según su función.</li><li>• Saber diseñar el algoritmo, con pseudocódigo, para un planteamiento de un problema dado.</li><li>• Mejorar la eficiencia de los algoritmos.</li><li>• Realizar pruebas y corrección de errores de algoritmos.</li></ul>

<b>UNIDAD DE TRABAJO 3</b> <b><i>ESTRUCTURAS ESTÁTICAS DE DATOS.</i></b>
<b>CONCEPTOS</b>
<ul style="list-style-type: none"><li>• Concepto de array o vector</li><li>• Tipos de arrays</li><li>• Definición de un array</li><li>• Operaciones sobre arrays</li></ul>
<b>PROCEDIMIENTOS</b>
<ul style="list-style-type: none"><li>• Interpretación del problema</li><li>• Elección de las estructuras de programación necesarias para la resolución del problema</li><li>• Construcción del algoritmo utilizando las estructuras elegidas</li><li>• Realización de pruebas</li><li>• Corrección de errores</li><li>• Documentación del programa</li></ul>
<b>ACTIVIDADES DE ENSEÑANZA-APRENDIZAJE</b>
<ul style="list-style-type: none"><li>• Comparación de las distintas herramientas de diseño de algoritmos</li><li>• Aplicación de las herramientas de diseño de algoritmos a la utilización de estructuras estáticas</li><li>• Justificación de la importancia de las estructuras estáticas</li><li>• Planteamiento de ejercicios</li><li>• Resolución de ejercicios en grupo</li><li>• Documentación de los ejercicios resueltos</li></ul>
<b>CRITERIOS DE EVALUACIÓN</b>
<ul style="list-style-type: none"><li>• Elegir correctamente las estructuras estáticas de datos que se van a utilizar.</li><li>• Definir y utilizar correctamente los arrays, conociendo y sabiendo realizar las diferentes operaciones sobre ellos.</li><li>• Describir las estructuras de datos estáticas que maneja un lenguaje estructurado, su utilidad y ámbito de aplicación.</li><li>• Aplicar una metodología de desarrollo estructurado para el diseño de algoritmos.</li></ul>

<b>UNIDAD DE TRABAJO 4</b> <b><i>ESTRUCTURAS DINÁMICAS DE DATOS.</i></b>
<b>CONCEPTOS</b>
<ul style="list-style-type: none"><li>• Punteros</li><li>• Listas</li><li>• Estructuras dinámicas lineales (pilas, colas)</li><li>• Estructuras dinámicas no lineales(árboles)</li></ul>
<b>PROCEDIMIENTOS</b>
<ul style="list-style-type: none"><li>• Elección de las estructuras dinámicas necesarias para la resolución del problema</li><li>• Construcción del algoritmo utilizando las estructuras dinámicas elegidas</li></ul>
<b>ACTIVIDADES DE ENSEÑANZA-APRENDIZAJE</b>
<ul style="list-style-type: none"><li>• Aplicación de las herramientas de diseño de algoritmos a la utilización de estructuras estáticas</li><li>• Justificación de la importancia de las estructuras estáticas</li><li>• Planteamiento de ejercicios</li><li>• Resolución de ejercicios en grupo</li><li>• Documentación de los ejercicios resueltos</li></ul>
<b>CRITERIOS DE EVALUACIÓN</b>
<ul style="list-style-type: none"><li>• Describir las estructuras de datos dinámicas que maneja un lenguaje estructurado, su utilidad y ámbito de aplicación.</li><li>• Justificar la importancia de la adecuada selección de estructuras de datos dinámicas para la resolución de problemas en programación.</li><li>• Evaluar la importancia de la claridad y legibilidad de los programas para facilitar el mantenimiento y el trabajo en equipo.</li><li>• Documentar el código de un módulo de programación con comentarios significativos, concisos y legibles.</li></ul>

<b>UNIDAD DE TRABAJO 5</b> <b><i>ESTRUCTURAS EXTERNAS DE DATOS.</i></b>
<b>CONCEPTOS</b>
<ul style="list-style-type: none"><li>• Conceptos y definiciones</li><li>• Clasificación de registros (fijos y variables)</li><li>• Operaciones con registros</li><li>• Clasificación de ficheros</li><li>• Organización y acceso</li></ul>
<b>PROCEDIMIENTOS</b>
<ul style="list-style-type: none"><li>• Elección de las estructuras externas necesarias para la resolución del problema</li><li>• Construcción del algoritmo utilizando las estructuras externas elegidas</li></ul>
<b>ACTIVIDADES DE ENSEÑANZA-APRENDIZAJE</b>
<ul style="list-style-type: none"><li>• Aplicación de las herramientas de diseño de algoritmos a la utilización de estructuras externas</li><li>• Justificación de la importancia de las estructuras externas</li><li>• Planteamiento de ejercicios</li><li>• Resolución de ejercicios en grupo</li><li>• Documentación de los ejercicios resueltos</li></ul>
<b>CRITERIOS DE EVALUACIÓN</b>
<ul style="list-style-type: none"><li>• Describir las estructuras de ficheros de datos típicas que maneja un lenguaje estructurado, su utilidad y ámbito de aplicación.</li><li>• Justificar la importancia de la adecuada selección de tipos de ficheros para la resolución de problemas en programación.</li><li>• Utilizar herramientas de diseño de programas.</li><li>• Documentar el código de un módulo de programación con comentarios significativos, concisos y legibles.</li></ul>

<b>UNIDAD DE TRABAJO 6 INTRODUCCIÓN AL LENGUAJE C.</b>
<b>CONCEPTOS</b>
<ul style="list-style-type: none"> <li>• Características del lenguaje C</li> <li>• Tipos de datos</li> <li>• Identificadores</li> <li>• Definición de variables</li> <li>• Modificadores de acceso</li> <li>• Tipos de datos</li> <li>• Operadores (orden de prioridad)</li> <li>• Tipos de instrucciones</li> <li>• Estructuras de control</li> <li>• La compilación y el linkado en C</li> <li>• Librerías de C</li> </ul>
<b>PROCEDIMIENTOS</b>
<ul style="list-style-type: none"> <li>• Descripción e identificación de los distintos elementos del listado de un programa fuente escrito en C</li> <li>• Identificación de las distintas estructuras de programación que aparecen en el listado fuente</li> <li>• Utilización del editor empleado</li> <li>• Utilización del compilador de C empleado</li> <li>• Realización de pruebas</li> <li>• Corrección de errores</li> </ul>
<b>ACTIVIDADES DE ENSEÑANZA-APRENDIZAJE</b>
<ul style="list-style-type: none"> <li>• Convertir algoritmos sencillos elaborados previamente en pseudocódigo a programas fuentes de C</li> <li>• Utilización práctica del compilador, del enlazador, de las librerías y del depurador, a partir de los listados de programas fuentes anteriores</li> <li>• Obtención y prueba del código ejecutable</li> </ul>
<b>CRITERIOS DE EVALUACIÓN</b>
<ul style="list-style-type: none"> <li>• Identificar y utilizar los elementos del lenguaje C.</li> <li>• Definir las instrucciones, funciones y librerías del lenguaje más básicas y su utilidad.</li> <li>• Codificar programas en un lenguaje estructurado a partir de los algoritmos diseñados.</li> <li>• Realizar pruebas para cada módulo de una aplicación y pruebas de integración.</li> <li>• Comprobar que los formatos de entrada y salida de la aplicación son los esperados.</li> </ul>

<b>UNIDAD DE TRABAJO 7 ARRAYS Y CADENAS EN C.</b>
<b>CONCEPTOS</b>
<ul style="list-style-type: none"><li>• Arrays unidimensionales</li><li>• Arrays bidimensionales</li><li>• Arrays multidimensionales</li><li>• Arrays indeterminados</li><li>• Cadenas de caracteres</li></ul>
<b>PROCEDIMIENTOS</b>
<ul style="list-style-type: none"><li>• Elección de los tipos de arrays necesarios para la resolución del problema</li><li>• Construcción de los algoritmos utilizando los arrays elegidos</li><li>• Codificación en C de los algoritmos</li><li>• Compilación de los programas fuentes</li><li>• Corrección de los errores observados</li><li>• Documentación del programa</li></ul>
<b>ACTIVIDADES DE ENSEÑANZA-APRENDIZAJE</b>
<ul style="list-style-type: none"><li>• Aplicación de las herramientas de diseño de algoritmos a la utilización de arrays</li><li>• Planteamiento de ejercicios</li><li>• Elección de las estructuras de arrays más adecuadas para representar y manejar los datos del problema propuesto</li><li>• Codificación el programa utilizando correctamente los arrays y el conjunto de instrucciones, funciones y librerías del lenguaje</li><li>• Depuración del programa fuente y obtener un programa ejecutable</li><li>• Elaboración de un conjunto de datos de prueba</li></ul>
<b>CRITERIOS DE EVALUACIÓN</b>
<ul style="list-style-type: none"><li>• Ejercicios donde se usen arrays lineales y quede claro la diferencia del índice del contenido de lo apuntado por él.</li><li>• Debe ser capaz el alumno de identificar y distinguir los distintos tipos de arrays a emplear y cuándo conviene optar por uno u otro dependiendo del ejercicio propuesto.</li><li>• Los propios trabajos realizados en clase de codificación, prueba y depuración de programas que utilicen estas estructuras darán completa información sobre las aptitudes del alumno</li></ul>

<p><b>UNIDAD DE TRABAJO 8</b> <b>PUNTEROS Y ASIGNACIÓN DINÁMICA DE MEMORIA.</b></p>
<p><b>CONCEPTOS</b></p> <ul style="list-style-type: none"> <li>• Conceptos básicos (indirección)</li> <li>• Operaciones con punteros (aritmética de punteros)</li> <li>• Punteros y arrays</li> <li>• Arrays de punteros</li> <li>• Indirección múltiple</li> <li>• Funciones de asignación dinámica de memoria</li> </ul>
<p><b>PROCEDIMIENTOS</b></p> <ul style="list-style-type: none"> <li>• Conocimiento de la utilidad de los punteros en C</li> <li>• Realización de diversas actividades con punteros</li> <li>• Utilización de los punteros en arrays y cadenas de caracteres</li> <li>• Uso de los arrays de punteros</li> <li>• Utilización de la indirección múltiple</li> <li>• Uso de las funciones para el manejo y asignación dinámica de memoria</li> </ul>
<p><b>ACTIVIDADES DE ENSEÑANZA-APRENDIZAJE</b></p> <ul style="list-style-type: none"> <li>• Aplicación de las herramientas de diseño de algoritmos a la utilización de punteros</li> <li>• Planteamiento de ejercicios</li> <li>• Elegir las estructuras de punteros más adecuadas para representar y manejar los datos del problema propuesto</li> <li>• Codificar el programa utilizando correctamente los punteros y el conjunto de instrucciones, funciones y librerías del lenguaje</li> <li>• Depurar el programa fuente y obtener un programa ejecutable</li> <li>• Elaborar un conjunto de datos de prueba</li> </ul>
<p><b>CRITERIOS DE EVALUACIÓN</b></p> <ul style="list-style-type: none"> <li>• Realización de ejercicios donde se evidencie el manejo y uso de punteros.</li> <li>• Debe ser capaz el alumno de identificar y distinguir cuándo conviene usar asignación dinámica de memoria dependiendo del ejercicio propuesto</li> <li>• Los propios trabajos realizados en clase de codificación, prueba y depuración de programas que utilicen punteros y asignación dinámica de memoria darán completa información sobre las aptitudes del alumno</li> </ul>

<b>UNIDAD DE TRABAJO 9</b> <b><i>FUNCIONES.</i></b>
<b>CONCEPTOS</b>
<ul style="list-style-type: none"><li>• Conceptos básicos sobre funciones</li><li>• Clases de almacenamiento de las variables</li><li>• Retorno de una función</li><li>• Clases de funciones</li><li>• Parámetros de funciones</li><li>• Funciones y arrays</li><li>• Funciones recursivas</li></ul>
<b>PROCEDIMIENTOS</b>
<ul style="list-style-type: none"><li>• Conocimiento de la definición y declaración de funciones</li><li>• Conocimiento del ámbito y duración de las variables</li><li>• Uso del retorno de una función</li><li>• Uso de los parámetros</li><li>• Uso del paso de arrays a funciones</li><li>• Uso de parámetros desde el SO</li><li>• Uso de funciones recursivas</li></ul>
<b>ACTIVIDADES DE ENSEÑANZA-APRENDIZAJE</b>
<ul style="list-style-type: none"><li>• Aplicación de las herramientas de diseño de algoritmos a la utilización de funciones</li><li>• Planteamiento de ejercicios</li><li>• Elegir las funciones más adecuadas para representar y manejar los datos del problema propuesto</li><li>• Codificar el programa utilizando correctamente las funciones necesarias y el conjunto de instrucciones y librerías del lenguaje</li><li>• Depurar el programa fuente y obtener un programa ejecutable</li><li>• Elaborar un conjunto de datos de prueba</li></ul>
<b>CRITERIOS DE EVALUACIÓN</b>
<ul style="list-style-type: none"><li>• Diseño y codificación de programas creando las funciones de usuario y utilizando las funciones predefinidas.</li><li>• Aparte de conocer los parámetros de llamada al gestor y la sintaxis de las funciones, con ayuda de documentación suficiente, el alumno debe ser capaz de implementar un programa con los requerimientos de funciones que se les pida.</li></ul>

<b>UNIDAD DE TRABAJO 10</b> <b>ESTRUCTURAS COMPUESTAS.</b>
<b>CONCEPTOS</b>
<ul style="list-style-type: none"><li>• Estructuras: Definición, referencia y carga de datos</li><li>• Estructuras anidadas</li><li>• Arrays de estructuras</li><li>• Punteros y estructuras</li><li>• Funciones y estructuras</li><li>• Tipos de datos definidos por el usuario</li></ul>
<b>PROCEDIMIENTOS</b>
<ul style="list-style-type: none"><li>• Conocimiento de la declaración definición y uso de las estructuras compuestas.</li><li>• Uso de los arrays de estructuras compuestas</li><li>• Paso de estructuras compuestas a funciones</li><li>• Uso de uniones y enumeraciones</li><li>• Definición de tipos de datos con nombre</li></ul>
<b>ACTIVIDADES DE ENSEÑANZA-APRENDIZAJE</b>
<ul style="list-style-type: none"><li>• Aplicación de las herramientas de diseño de algoritmos a la utilización de estructuras compuestas</li><li>• Planteamiento de ejercicios</li><li>• Elegir las estructuras compuestas más adecuadas para representar y manejar los datos del problema propuesto</li><li>• Codificar el programa utilizando correctamente las estructuras compuestas y el conjunto de instrucciones, funciones y librerías del lenguaje</li><li>• Depurar el programa fuente y obtener un programa ejecutable</li><li>• Elaborar un conjunto de datos de prueba</li></ul>
<b>CRITERIOS DE EVALUACIÓN</b>
<ul style="list-style-type: none"><li>• Realización de ejercicios semejantes a los explicados en clase donde se evidencie el manejo y uso de estructuras</li><li>• Los propios trabajos realizados en clase de codificación, prueba y depuración de programas que utilicen estructuras de datos y la de datos darán completa información sobre las aptitudes del alumno.</li></ul>

<b>UNIDAD DE TRABAJO 11</b> <b>E/S POR FICHEROS.</b>
<b>CONCEPTOS</b>
<ul style="list-style-type: none"> <li>• Puntero a ficheros</li> <li>• Funciones para el tratamiento de ficheros</li> <li>• Acceso secuencial</li> <li>• Acceso directo</li> </ul>
<b>PROCEDIMIENTOS</b>
<ul style="list-style-type: none"> <li>• Conocimiento de las distintas herramientas que proporciona el lenguaje C para el tratamiento de ficheros</li> <li>• Diseño de programas para el tratamiento de ficheros secuenciales</li> <li>• Conocimiento de las ventajas e inconvenientes del sistema de ficheros en C, así como sus limitaciones</li> </ul>
<b>ACTIVIDADES DE ENSEÑANZA-APRENDIZAJE</b>
<ul style="list-style-type: none"> <li>• Aplicación de las herramientas de diseño de algoritmos a la utilización de ficheros</li> <li>• Planteamiento de ejercicios</li> <li>• Elegir las accesos a ficheros mas adecuados para representar y manejar los datos del problema propuesto</li> <li>• Codificar el programa utilizando correctamente la E/S a ficheros y el conjunto de instrucciones, funciones y librerías del lenguaje</li> <li>• Depurar el programa fuente y obtener un programa ejecutable</li> <li>• Elaborar un conjunto de datos de prueba</li> </ul>
<b>CRITERIOS DE EVALUACIÓN</b>
<ul style="list-style-type: none"> <li>• Debe ser el alumno capaz de realizar por si mismo cada una de las operaciones diseñadas, en una situación de tener que crear o disponer de un fichero con organización secuencial.</li> <li>• Planteado un problema de necesidad de almacenar registros a través de una clave, el alumno deberá hacer todo el proceso que permita crear, inicializar, transformar la clave, cargar, consultar,... y demás operaciones del ciclo de vida de este fichero.</li> <li>• Los propios trabajos realizados en clase de codificación, prueba y depuración de programas que utilicen ficheros de entrada y salida de datos darán completa información sobre las aptitudes del alumno.</li> </ul>

<b>UNIDAD DE TRABAJO 12</b> <b>EL PREPROCESADOR DE C.</b>
<b>CONCEPTOS</b>
<ul style="list-style-type: none"><li>• Directiva para la definición de constantes simbólicas</li><li>• Directiva para definir macros</li><li>• Directiva para inclusión de ficheros</li><li>• Directivas condicionales</li><li>• Otras directivas</li></ul>
<b>PROCEDIMIENTOS</b>
<ul style="list-style-type: none"><li>• Conocimiento sobre lo que significa el preprocesador</li><li>• Uso de las funciones del preprocesador</li><li>• Conocimiento de la sintaxis de las principales directivas de compilación</li></ul>
<b>ACTIVIDADES DE ENSEÑANZA-APRENDIZAJE</b>
<ul style="list-style-type: none"><li>• Introducir sentencias de compilación en los programas de C que permitan:</li><li>• Insertar código de otros ficheros fuente</li><li>• Poder establecer criterios de compilación condicional</li><li>• Definición de macros y constantes simbólicas</li></ul>
<b>CRITERIOS DE EVALUACIÓN</b>
<ul style="list-style-type: none"><li>• Los propios trabajos realizados en clase de codificación, prueba y depuración de programas que utilicen directivas del preprocesador insertadas en el código fuente de los programas darán completa información sobre las aptitudes del alumno.</li></ul>

<p><b>UNIDAD DE TRABAJO 13</b>  <b>COLAS, PILAS, LISTAS ENLAZADAS Y ÁRBOLES.</b></p>
<p><b>CONCEPTOS</b></p> <ul style="list-style-type: none"> <li>• Colas</li> <li>• La cola circular</li> <li>• Pilas</li> <li>• Listas enlazadas (simplemente y doblemente enlazadas)</li> <li>• Árboles binarios</li> </ul>
<p><b>PROCEDIMIENTOS</b></p> <ul style="list-style-type: none"> <li>• Gestión de la memoria del sistema</li> <li>• Elección de las estructuras dinámicas necesarias para la resolución de problemas</li> <li>• Construcción del algoritmo utilizando las estructuras dinámicas elegidas</li> <li>• Realización de pruebas</li> <li>• Corrección de los errores observados</li> </ul>
<p><b>ACTIVIDADES DE ENSEÑANZA-APRENDIZAJE</b></p> <ul style="list-style-type: none"> <li>• Justificar la importancia de la adecuada selección de estructuras de datos dinámicas para la resolución de problemas en programación</li> <li>• Describir las estructuras de datos dinámicas que maneja C, su utilidad y ámbito de aplicación</li> <li>• Codificar el programa utilizando correctamente las estructuras dinámicas (pilas, colas, etc...) y el conjunto de instrucciones, funciones y librerías del lenguaje</li> <li>• Depurar el programa fuente y obtener un programa ejecutable</li> <li>• Elaborar un conjunto de datos de prueba</li> </ul>
<p><b>CRITERIOS DE EVALUACIÓN</b></p> <ul style="list-style-type: none"> <li>• Reconocimiento de las estructuras dinámicas: listas, pilas, colas, listas doblemente enlazadas y circulares, árboles, grafos,...</li> <li>• Resolución de ejercicios con estructuras dinámicas de una dificultad semejante a los hechos en clase con la ayuda del profesor</li> </ul>

<b>UNIDAD DE TRABAJO 14</b> <b>ADAPTACIÓN Y/O CREACIÓN DE APLICACIONES Y/O FUNCIONES SENCILLAS PARA EL SISTEMA.</b>
<b>CONCEPTOS</b>
<ul style="list-style-type: none"><li>• Utilización de los conocimientos adquiridos en las UT anteriores.</li><li>• Ficheros de cabecera</li><li>• Librerías de funciones</li></ul>
<b>PROCEDIMIENTOS</b>
<ul style="list-style-type: none"><li>• Interpretación del problema o modulo que se desea modificar</li><li>• Elección de las estructuras necesarias para la resolución del nuevo problema</li><li>• Codificación de la nueva aplicación o en su caso de los módulos afectados por la modificación</li><li>• Realización de pruebas</li><li>• Corrección de errores</li></ul>
<b>ACTIVIDADES DE ENSEÑANZA-APRENDIZAJE</b>
<ul style="list-style-type: none"><li>• Interpretación del problema o modulo que se desea modificar</li><li>• Elección de las estructuras necesarias para la resolución del nuevo problema</li><li>• Codificación de la nueva aplicación o en su caso de los módulos afectados por la modificación</li><li>• Realización de pruebas</li><li>• Corrección de errores</li></ul>
<b>CRITERIOS DE EVALUACIÓN</b>
<ul style="list-style-type: none"><li>• Integrar y enlazar módulos de programación, rutinas y utilidades siguiendo las especificaciones del diseño.</li><li>• Codificar y depurar los diferentes módulos.</li><li>• Realizar pruebas para cada módulo y pruebas de integración.</li><li>• Documentar los cambios realizados sobre los datos, módulos y estructuras de datos y control de la aplicación.</li></ul>

## **6.- BIBLIOGRAFÍA RECOMENDADA.**

Alcalde, E., García,M., *Metodología de la programación*, McGraw-Hill,1992.

Joyanes,L., *Fundamentos de Programación*, McGraw-Hill

Ceballos,F.J. *Curso de programación con C. Ra-Ma*, Madrid 1993

Quero E., *Fundamentos de programación*, Paraninfo 2001

Kernighan, B.,Ritchie, D., *El lenguaje de programación C*, Prentice-Hall