

DESARROLLO CURRICULAR DEL MÓDULO

# FUNDAMENTOS DE PROGRAMACIÓN

CICLO FORMATIVO DE GRADO SUPERIOR

**ADMINISTRACIÓN DE SISTEMAS INFORMÁTICOS**

Profesor: Fernando Ruiz Rico

# Índice de contenido

<b>1. INTRODUCCIÓN.....</b>	<b>5</b>
<b>2. OBJETIVOS.....</b>	<b>5</b>
<b>3. CAPACIDADES TERMINALES.....</b>	<b>6</b>
<b>4. ORGANIZACIÓN DE LOS CONTENIDOS.....</b>	<b>7</b>
<b>4.1.- ESTRUCTURA DE LOS CONTENIDOS.....</b>	<b>7</b>
<b>4.2.- RELACIÓN SECUENCIADA DE UNIDADES DE TRABAJO.....</b>	<b>8</b>
<b>4.3.- CONTENIDOS MÍNIMOS.....</b>	<b>9</b>
<b>5. ELEMENTOS CURRICULARES DE CADA UNIDAD.....</b>	<b>10</b>
<b>5.1. PLANTEAMIENTO DE LAS UNIDADES TEMÁTICAS.....</b>	<b>10</b>
<b>5.2. DETALLE DE CADA UNIDAD.....</b>	<b>12</b>
<b>UNIDAD 0. Conceptos básicos sobre programación .....</b>	<b>12</b>
<b>CONTENIDOS.....</b>	<b>12</b>
<b>Conceptos.....</b>	<b>12</b>
<b>Procedimientos.....</b>	<b>12</b>
<b>Actividades de enseñanza-aprendizaje .....</b>	<b>12</b>
<b>Criterios de evaluación.....</b>	<b>12</b>
<b>UNIDAD 1. Toma de contacto con C .....</b>	<b>13</b>
<b>CONTENIDOS.....</b>	<b>13</b>
<b>Conceptos.....</b>	<b>13</b>
<b>Procedimientos.....</b>	<b>13</b>
<b>Actividades de enseñanza-aprendizaje.....</b>	<b>13</b>
<b>Criterios de evaluación.....</b>	<b>13</b>
<b>UNIDAD 2. Tipos de datos básicos.....</b>	<b>14</b>
<b>CONTENIDOS.....</b>	<b>14</b>
<b>Conceptos.....</b>	<b>14</b>
<b>Procedimientos.....</b>	<b>14</b>
<b>Actividades de enseñanza-aprendizaje.....</b>	<b>15</b>
<b>Criterios de evaluación.....</b>	<b>15</b>
<b>UNIDAD 3. Estructuras de control.....</b>	<b>15</b>
<b>CONTENIDOS.....</b>	<b>15</b>
<b>Conceptos.....</b>	<b>15</b>
<b>Procedimientos.....</b>	<b>16</b>
<b>Actividades de enseñanza-aprendizaje.....</b>	<b>16</b>
<b>Criterios de evaluación.....</b>	<b>17</b>
<b>UNIDAD 4. Entrada/salida básica.....</b>	<b>17</b>

<b>CONTENIDOS</b> .....	<b>17</b>
<b>Conceptos</b> .....	<b>17</b>
<b>Procedimientos</b> .....	<b>18</b>
<b>Actividades de enseñanza-aprendizaje</b> .....	<b>18</b>
<b>Criterios de evaluación</b> .....	<b>18</b>
<b>UNIDAD 5. Arrays y estructuras</b> .....	<b>18</b>
<b>CONTENIDOS</b> .....	<b>18</b>
<b>Conceptos</b> .....	<b>18</b>
<b>Procedimientos</b> .....	<b>19</b>
<b>Actividades de enseñanza-aprendizaje</b> .....	<b>19</b>
<b>Criterios de evaluación</b> .....	<b>19</b>
<b>UNIDAD 6. Funciones</b> .....	<b>20</b>
<b>CONTENIDOS</b> .....	<b>20</b>
<b>Conceptos</b> .....	<b>20</b>
<b>Procedimientos</b> .....	<b>20</b>
<b>Actividades de enseñanza-aprendizaje</b> .....	<b>20</b>
<b>Criterios de evaluación</b> .....	<b>20</b>
<b>UNIDAD 7. Punteros y gestión dinámica de memoria</b> .....	<b>21</b>
<b>CONTENIDOS</b> .....	<b>21</b>
<b>Conceptos</b> .....	<b>21</b>
<b>Procedimientos</b> .....	<b>21</b>
<b>Actividades de enseñanza-aprendizaje</b> .....	<b>21</b>
<b>Criterios de evaluación</b> .....	<b>22</b>
<b>UNIDAD 8. Librerías de uso frecuente</b> .....	<b>22</b>
<b>CONTENIDOS</b> .....	<b>22</b>
<b>Conceptos</b> .....	<b>22</b>
<b>Procedimientos</b> .....	<b>22</b>
<b>Actividades de enseñanza-aprendizaje</b> .....	<b>23</b>
<b>Criterios de evaluación</b> .....	<b>23</b>
<b>UNIDAD 9. Ficheros</b> .....	<b>24</b>
<b>CONTENIDOS</b> .....	<b>24</b>
<b>Conceptos</b> .....	<b>24</b>
<b>Procedimientos</b> .....	<b>24</b>
<b>Actividades de enseñanza-aprendizaje</b> .....	<b>25</b>
<b>Criterios de evaluación</b> .....	<b>25</b>
<b>UNIDAD 10. Características avanzadas de C</b> .....	<b>26</b>
<b>CONTENIDOS</b> .....	<b>26</b>
<b>Conceptos</b> .....	<b>26</b>
<b>Procedimientos</b> .....	<b>26</b>

<b>Actividades de enseñanza-aprendizaje.....</b>	<b>26</b>
<b>Criterios de evaluación.....</b>	<b>27</b>
<b>UNIDAD 11. Ejercicios completos.....</b>	<b>27</b>
<b>CONTENIDOS.....</b>	<b>27</b>
<b>Conceptos.....</b>	<b>27</b>
<b>Procedimientos.....</b>	<b>27</b>
<b>Actividades de enseñanza-aprendizaje.....</b>	<b>27</b>
<b>Criterios de evaluación.....</b>	<b>27</b>
<b>UNIDAD 12. Proyecto final .....</b>	<b>28</b>
<b>CONTENIDOS.....</b>	<b>28</b>
<b>Conceptos.....</b>	<b>28</b>
<b>Procedimientos.....</b>	<b>28</b>
<b>Actividades de enseñanza-aprendizaje.....</b>	<b>28</b>
<b>Criterios de evaluación.....</b>	<b>28</b>
<b>6. TEMPORALIZACIÓN.....</b>	<b>29</b>
<b>7. BIBLIOGRAFÍA.....</b>	<b>30</b>
<b>8. PLANTEAMIENTO DE LA ATENCIÓN A LA DIVERSIDAD.....</b>	<b>31</b>
<b>9. CRITERIOS DE EVALUACIÓN.....</b>	<b>32</b>
<b>10. ACTIVIDADES COMPLEMENTARIAS Y EXTRAESCOLARES.....</b>	<b>33</b>
<b>11. MATERIALES Y RECURSOS DIDÁCTICOS.....</b>	<b>33</b>

## 1. INTRODUCCIÓN

El módulo de **Fundamentos de Programación** es de 288 horas totales (9 horas semanales) y 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 según 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 métodos 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

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. El lenguaje de programación C (inicial)
2. El lenguaje de programación C (avanzado)
3. Mantenimiento de programas

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

### Bloque 1: Metodología de la programación. El lenguaje de programación C (inicial)

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 diagramas de flujo para la resolución de algoritmos que resuelvan esos problemas planteados, siguiendo un diseño modular y estructurado.

Se iniciará el lenguaje relacionando la sintaxis de las estructuras de datos simples, y las sentencias elementales de este lenguaje con respecto al diagrama de flujo. Casi desde el primer día se empezará a trabajar en lenguaje C, para evitar que una carga teórica inicial excesiva pueda hacer la asignatura tediosa para el alumno.

### Bloque 2: 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 3: 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.

## 4.2.- RELACIÓN SECUENCIADA 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.

Bloque 1: Metodología de la programación. El lenguaje de programación C (inicial)

0. Conceptos básicos sobre programación
1. Toma de contacto con C
2. Tipos de datos básicos
3. Estructuras de control
4. Entrada/salida básica
5. Arrays y estructuras
6. Funciones

Bloque 2: El lenguaje de programación C (avanzado)

7. Punteros y gestión dinámica de memoria
8. Librerías de uso frecuente
9. Ficheros
10. Características avanzadas de C

Bloque 3: Mantenimiento de programas

11. Ejercicios completos
12. Proyecto final

### **4.3.- 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

### 5.1. PLANTEAMIENTO DE LAS UNIDADES TEMÁTICAS

#### UT.0. Conceptos básicos sobre programación

La UT.0. tiene como fin presentar al alumno los conceptos básicos sobre la programación de tal manera que comience a familiarizarse con los términos, entornos, materiales y finalidades del Módulo completo. Es una Unidad eminentemente conceptual.

#### UT.1. Toma de contacto con C

Esta unidad acerca al alumno al lenguaje C: la estructura de un programa fundamental, como acceder a pantalla, como mostrar textos prefijados y números enteros, cómo leer números desde el teclado y realizar operaciones aritméticas básicas.

#### UT.2. Tipos de datos básicos

Una vez que el alumno tiene soltura con los números enteros, se le introducen otros tipos de datos imprescindibles, como los números reales y los caracteres, y se le comenta la dificultad que tendrán las cadenas de texto.

#### UT.3. Estructuras de control

En esta unidad se muestra al alumno la forma de comprobar condiciones y de crear bloques de instrucciones que se repitan dentro de un programa.

#### UT.4. Entrada/salida básica

Se concreta más la información que se había presentado en la unidad 1 sobre cómo acceder a pantalla y a teclado, presentando funciones adicionales y más detalles sobre las que ya se conocían.

#### UT.5. Arrays y estructuras

Se muestran al alumnos las estructuras de datos estáticas y su manejo, tanto en lo que se refiere a los arrays unidimensionales, bidimensionales, multidimensionales, como a las estructuras, simples o anidadas. También se acerca al alumno al uso de las cadenas de texto, todavía sin entrar en su uso mediante punteros.

#### UT.6. 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.7. Punteros y gestión 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.8. Librerías de uso frecuente

El alumno necesitará con frecuencia recurrir a bibliotecas de funciones existentes en C y que todavía no se le han presentado, como “math.h” para operaciones matemáticas y generación de números aleatorios, o “ncurses.h” y “conio.h” para acceso avanzado a la pantalla de texto. Esas bibliotecas serán utilizadas en esta Unidad.

#### UT.9. 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.10. Características avanzadas de C

El alumno debe conocer cuales son las funciones del preprocesador y el uso y sintaxis de las principales directivas. También se le presentarán otras características de uso menos frecuente, como los operadores a nivel de bits y el operador coma, y se le enseña a crear ficheros .h.

#### UT.11. Ejercicios completos

Esta unidad trata de enfrentar al alumno con creación de aplicaciones y la adaptación de aplicaciones que ya están hechas para satisfacer nuevos requerimientos. 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.

#### UT.12. Proyecto final

En esta unidad, los alumnos tendrán que enfrentarse a un proyecto de programación de mayor complejidad, y que deberá ser realizado individualmente o en grupo junto con alguno(s) de sus compañeros.

## 5.2. DETALLE DE CADA UNIDAD

### UNIDAD 0. Conceptos básicos sobre programación

#### CONTENIDOS

##### Conceptos

- ✓ Evolución y clasificación de lenguajes: Lenguajes de bajo nivel y de alto nivel.
- ✓ Ensambladores, compiladores e intérpretes
- ✓ Diagramas de flujo

##### Procedimientos

- ✓ Descripción de la evolución de los lenguajes informáticos
- ✓ Ventajas de los lenguajes compilados sobre los interpretados y viceversa.

##### Actividades de enseñanza-aprendizaje

- ✓ Identificación de los lenguajes interpretados de los lenguajes compilados
- ✓ Ejemplo de programa escrito en un lenguaje de bajo nivel
- ✓ Toma de contacto con un lenguaje de alto nivel interpretado (Basic) y otro compilado (Pascal y C).
- ✓ Ejemplo de un algoritmo sencillo con diagramas de flujo.

##### Criterios de evaluación

- ✓ Distinguir las características de los lenguajes de alto y bajo nivel.
- ✓ Describir los tipos de lenguajes, compiladores y traductores de uso mas común.
- ✓ Ser capaz de expresar algoritmos elementales con diagramas de flujo.

## UNIDAD 1. Toma de contacto con C

### CONTENIDOS

#### Conceptos

- ✓ Escribir un texto en C
- ✓ Mostrar números enteros en pantalla
- ✓ Operaciones aritméticas básicas
- ✓ Introducción a las variables: int
- ✓ Identificadores
- ✓ Comentarios
- ✓ Datos introducidos por el usuario: scanf

#### Procedimientos

- ✓ Manejo básico del compilador/entorno escogido para el módulo.
- ✓ Realización de programas sencillos capaces de mostrar textos en pantalla, o valores numéricos enteros, o el resultado de operaciones con números enteros.
- ✓ Previsión del resultado de operaciones matemáticas, teniendo en cuenta la prioridad de los operadores.
- ✓ Lectura de datos tecleados por el usuario del programa, mediante el uso de variables.

#### Actividades de enseñanza-aprendizaje

- ✓ Mostrar en pantalla textos prefijados.
- ✓ Mostrar en pantalla números enteros y realizar operaciones con ellos.
- ✓ Prever el resultado de operaciones matemáticas que implican diversos operadores.
- ✓ Leer números enteros tecleados por el usuario y realizar operaciones aritméticas con ellos.

#### Criterios de evaluación

- ✓ Mostrar en pantalla textos prefijados.
- ✓ Prever el resultado de operaciones matemáticas que implican diversos operadores.

- ✓ Distinguir qué nombres de identificadores son válidos y cuales no.
- ✓ Leer números enteros tecleados por el usuario y realizar operaciones aritméticas con ellos.
- ✓ Comentar programas para aclarar las partes de código más confusas.

## UNIDAD 2. Tipos de datos básicos

### CONTENIDOS

#### Conceptos

- ✓ Tipos de enteros: short/long, signed/unsigned.
- ✓ Sistemas de numeración: binario, octal, hexadecimal.
- ✓ Representación interna de los enteros
- ✓ Incremento y decremento
- ✓ Operaciones abreviadas: +=
- ✓ Modificadores de acceso: const, volatile
- ✓ Tipo de dato real: Simple y doble precisión, cómo mostrar en pantalla.
- ✓ Operador de tamaño: sizeof variable, sizeof (tipo)
- ✓ Operador de molde: (tipo) operando
- ✓ Tipo de dato carácter; cómo mostrar en pantalla
- ✓ Secuencias de escape
- ✓ Introducción a las dificultades de las cadenas de texto

#### Procedimientos

- ✓ Decidir el modificador adecuado para almacenar números de mayor o menor tamaño.
- ✓ Cambiar números enteros de un sistema de numeración a otra (decimal, binario, octal, hexadecimal).
- ✓ Incrementar/decrementar el valor de una variable.
- ✓ Expresar operaciones con notación abreviada.
- ✓ Realizar operaciones con números reales, conociendo la diferencia entre simple y doble precisión y sabiendo cual elegir según la situación.

- ✓ Conocer el espacio ocupado por una variable o por un tipo de datos.
- ✓ Manejar variables de tipo carácter, incluyendo secuencias de escape.

### Actividades de enseñanza-aprendizaje

- ✓ Realizar operaciones con números enteros con o sin signo, de mayor o menor tamaño.
- ✓ Cambiar números enteros de un sistema de numeración a otra (decimal, binario, octal, hexadecimal).
- ✓ Incrementar/decrementar el valor de una variable.
- ✓ Expresar operaciones con notación abreviada.
- ✓ Realización de operaciones con números reales, tanto de simple como de doble precisión.
- ✓ Cálculo del espacio ocupado por una variable o por un tipo de datos.
- ✓ Manejo de variables de tipo carácter, incluyendo secuencias de escape.

### Criterios de evaluación

- ✓ Operar con soltura con números enteros con o sin signo, de mayor o menor tamaño.
- ✓ Ser capaz de cambiar números enteros de un sistema de numeración a otra (decimal, binario, octal, hexadecimal).
- ✓ Saber cómo incrementar/decrementar el valor de una variable y cómo expresar operaciones con notación abreviada.
- ✓ Realizar operaciones con números reales, conociendo la diferencia entre simple y doble precisión y sabiendo cual elegir según la situación.
- ✓ Saber cómo conocer el espacio ocupado por una variable o por un tipo de datos.
- ✓ Manejar variables de tipo carácter, incluyendo secuencias de escape.

## UNIDAD 3. Estructuras de control

### CONTENIDOS

#### Conceptos

- ✓ Estructuras alternativas: If, If-else, Switch, Operador condicional

- ✓ Estructuras repetitivas: While, Do ... While, For. Break, continue
- ✓ Sentencia goto
- ✓ Nociones de diseño: diagramas de flujo

### Procedimientos

- ✓ Comprobación de la alternativa correcta entre dos o un número reducido, con if y if-else.
- ✓ Comprobación de la alternativa entre varias posibles con switch.
- ✓ Realización de bloques repetitivos con while (si la condición de salida se comprueba al comienzo), con do..while (si la condición se comprueba al final) y con for (especialmente cuando es un número de iteraciones conocido).
- ✓ Interrupción de bucles con break y con continue.
- ✓ Uso de la sentencia goto y problemática asociada.
- ✓ Creación de diagramas de flujo para programas elementales.
- ✓ Introducción al depurador del entorno escogido.

### Actividades de enseñanza-aprendizaje

- ✓ Realización de programas en los que se deba escoger entre dos opciones usando if.
- ✓ Realización de programas en los que se deba escoger entre un número reducido de opciones usando if-else.
- ✓ Escoger una alternativa entre varias posibles con switch.
- ✓ Realización de bloques repetitivos cuya condición de salida se comprueba al comienzo (con while)
- ✓ Realización de bloques repetitivos cuya condición se comprueba al final (con do..while).
- ✓ Realización de bloques repetitivos con un número de iteraciones conocido (con for).
- ✓ Realización de bloques repetitivos de todo tipo, en los que el alumno deba ser quien decida qué método usar.
- ✓ Ejemplo de uso de la sentencia goto y explicación de la problemática asociada (desorden en el fuente).
- ✓ Creación de diagramas de flujo para programas elementales.
- ✓ Seguimiento del valor de variables usando el depurador del entorno escogido.

## Criterios de evaluación

- ✓ Saber escoger entre dos opciones usando if.
- ✓ Ser capaz de crear programas en los que se deba escoger entre un número reducido de opciones usando if-else.
- ✓ Escoger una alternativa entre varias posibles con switch, conociendo el uso correcto de las sentencias break y default.
- ✓ Corrección en la realización de bloques repetitivos cuya condición de salida se comprueba al comienzo (con while)
- ✓ Corrección en la realización de bloques repetitivos cuya condición se comprueba al final (con do..while).
- ✓ Corrección en la realización de bloques repetitivos con un número de iteraciones conocido (con for).
- ✓ Ser capaz de plantear bloques repetitivos de todo tipo, escogiendo un método adecuado para el problema.
- ✓ Conocer la sentencia goto y los problemas que se pueden derivar de su uso.
- ✓ Ser capaz de crear diagramas de flujo para programas elementales que incluyan condiciones o repetición de bloques de programa.
- ✓ Poder analizar el flujo del programa y comprobar el valor de variables usando el depurador del entorno escogido.

## UNIDAD 4. Entrada/salida básica

### CONTENIDOS

#### Conceptos

- ✓ printf
- ✓ scanf
- ✓ putchar
- ✓ getchar

## Procedimientos

- ✓ Escritura en pantalla, con mayor detalle del que se había visto hasta ahora.
- ✓ Lectura de teclado.

## Actividades de enseñanza-aprendizaje

- ✓ Mostrar textos en pantalla, incluyendo secuencias de escape y códigos de formato.
- ✓ Uso de posibilidades menos básicas de printf: alineación derecha o izquierda, caracteres de relleno, etc.
- ✓ Lectura de teclado con scanf.
- ✓ Manejo a nivel de carácter: getchar y putchar.

## Criterios de evaluación

- ✓ Mostrar textos en pantalla, incluyendo secuencias de escape y códigos de formato.
- ✓ Ser capaz de alinear textos a derecha e izquierda usando printf, así como ajustar la anchura en pantalla, limitar el uso de decimales o usar caracteres de relleno.
- ✓ Leer valores desde teclado con scanf.
- ✓ Leer y escribir caracteres con getchar y putchar.

## UNIDAD 5. Arrays y estructuras

### CONTENIDOS

#### Conceptos

- ✓ Conceptos básicos sobre tablas
- ✓ Arrays unidimensionales
- ✓ Arrays bidimensionales
- ✓ Arrays multidimensionales
- ✓ Arrays de caracteres
- ✓ Arrays indeterminados
- ✓ Estructuras

- ✓ Estructuras anidadas
- ✓ Arrays de estructuras

### Procedimientos

- ✓ Definición de arrays.
- ✓ Carga de datos en un array, en la inicialización o posteriormente.
- ✓ Acceso a los datos de un array.
- ✓ Empleo de las funciones específicas para manejo de arrays de caracteres.
- ✓ Estructuras: definición, carga de datos y acceso a los datos.

### Actividades de enseñanza-aprendizaje

- ✓ Almacenar datos repetitivos empleando arrays.
- ✓ Almacenar datos compuestos empleando estructuras.
- ✓ Manejar cadenas de caracteres, empleando las principales funciones disponibles para ellas: lectura y escritura, asignación de valores y comparación con otras cadenas.
- ✓ Combinar el uso de estructuras y arrays para datos repetitivos formados por unidades más complejas.

### Criterios de evaluación

- ✓ Saber cómo almacenar datos repetitivos empleando arrays, acceder a ellos y manipularlos.
- ✓ Almacenar datos compuestos empleando estructuras, acceder a ellos y manipularlos..
- ✓ Leer cadenas de caracteres desde el teclado, mostrarlas en pantalla, asignarles un valor y comparar con otras cadenas.
- ✓ Combinar el uso de estructuras y arrays para datos repetitivos formados por unidades más complejas.

## UNIDAD 6. Funciones

### CONTENIDOS

#### Conceptos

- ✓ Nociones de diseño modular de programas: Descomposición modular
- ✓ Conceptos básicos sobre funciones: Definición, Declaración, Llamada
- ✓ Formato antiguo de funciones
- ✓ Retorno de una función
- ✓ Clases de funciones
- ✓ Clases de almacenamiento de las variables
- ✓ Parámetros de funciones
- ✓ Funciones y arrays
- ✓ Recursividad

#### Procedimientos

- ✓ Descomposición modular de problemas como método de obtención de funciones.
- ✓ Definición y uso de funciones.
- ✓ Especificación del valor de retorno de una función
- ✓ Uso de parámetros de funciones
- ✓ Creación de funciones recursivas

#### Actividades de enseñanza-aprendizaje

- ✓ Dado un problema, aplicar técnicas de descomposición modular para obtener las funciones más adecuadas para su resolución.
- ✓ Definición y uso de funciones, que devuelvan un valor o no (procedimientos).
- ✓ Definición de variables locales y globales.
- ✓ Uso de parámetros de funciones, por valor y por referencia.
- ✓ Creación de funciones recursivas.

#### Criterios de evaluación

- ✓ El alumno debe ser capaz de, dado un problema, aplicar técnicas de descomposición

modular para obtener las funciones más adecuadas para su resolución.

- ✓ Definir funciones, llamarlas cuando sea necesario, usando valores de retorno o parámetros para intercambiar información.
- ✓ Definir variables locales para el trabajo interno de la función.
- ✓ Definir funciones recursivas, como forma rápida de resolver muchos problemas.

## UNIDAD 7. Punteros y gestión dinámica de memoria

### CONTENIDOS

#### Conceptos

- ✓ Punteros: concepto y definición en C. Operaciones con punteros.
- ✓ Punteros y arrays. Punteros y estructuras.
- ✓ Funciones de asignación dinámica de memoria.
- ✓ Parámetros de “main”
- ✓ Estructuras dinámicas: Listas (contiguas, enlazadas o encadenadas, doblemente enlazadas, circulares), Pilas, Colas, Árboles

#### Procedimientos

- ✓ Definición de punteros y asignación de direcciones.
- ✓ Asignación de punteros y aritmética de punteros.
- ✓ Comparación de punteros y de valores apuntados.
- ✓ Uso de los arrays como punteros.
- ✓ Definición y uso de los arrays de punteros.
- ✓ Definición y uso de los punteros a estructuras.
- ✓ Lectura de los parámetros pasados a un programa, usando los parámetros de “main”.

#### Actividades de enseñanza-aprendizaje

- ✓ Definir punteros, reservarles espacio, asignarles valores, liberar el espacio reservado.
- ✓ Incrementar y decrementar punteros y valores apuntados.

- ✓ Comparar punteros y de valores apuntados.
- ✓ Acceder a los elementos de un array usando punteros.
- ✓ Crear arrays de punteros, almacenar datos y acceder a ellos.
- ✓ Definir punteros a estructuras, almacenar datos y acceder a ellos.
- ✓ Leer los parámetros pasados a un programa, usando los parámetros de “main”.

### Criterios de evaluación

- ✓ Ser capaz de definir punteros, reservar espacio, asignar valores y liberar finalmente el espacio ocupado.
- ✓ Saber comparar, incrementar y decrementar punteros y valores apuntados.
- ✓ Poder acceder a los elementos de un array usando punteros.
- ✓ Crear arrays de punteros, almacenar datos y acceder a ellos.
- ✓ Definir punteros a estructuras, almacenar datos y acceder a ellos.
- ✓ Poder crear programas no interactivos que respondan a los deseos del usuario, leyendo los parámetros de “main”.

## UNIDAD 8. Librerías de uso frecuente

### CONTENIDOS

#### Conceptos

- ✓ Acceso a pantalla en Linux (Unix): ncurses.h
- ✓ Acceso a pantalla en Dos/Windows: conio.h
- ✓ Funciones matemáticas
- ✓ Números aleatorios
- ✓ Fecha y hora
- ✓ Llamada a órdenes del sistema

#### Procedimientos

- ✓ Aprovechamiento de la pantalla de texto en Linux (Unix): borrar, escribir en cierta posición, indicar colores para el texto, leer pulsaciones de tecla con o sin espera,

otras posibilidades.

- ✓ Aprovechamiento de la pantalla de texto en Dos/Windows: borrar, escribir en cierta posición, indicar colores para el texto, leer pulsaciones de tecla con o sin espera, otras posibilidades.
- ✓ Uso de funciones matemáticas de uso frecuente: potencia, raíz cuadrada, trigonométricas, redondeo, valor absoluto.
- ✓ Generación números aleatorios.
- ✓ Acceso a la fecha y hora del sistema.
- ✓ Llamada a órdenes del sistema empleando “system”.

### Actividades de enseñanza-aprendizaje

- ✓ Manipular la pantalla en Linux, borrar, escribir en cierta posición, indicar colores para el texto, leer pulsaciones de tecla con y sin espera.
- ✓ Manipular la pantalla en Dos/Windows, borrar, escribir en cierta posición, indicar colores para el texto, leer pulsaciones de tecla con y sin espera.
- ✓ Calcular potencias con números reales, raíces cuadradas, expresiones trigonométricas y redondear valores, empleando las funciones de “math.h”.
- ✓ Generar números aleatorios.
- ✓ Leer la fecha y hora del sistema.
- ✓ Acceder a órdenes externas del sistema empleando “system”.

### Criterios de evaluación

- ✓ Ser capaz de manipular la pantalla en Linux: borrar, escribir en cierta posición, indicar colores para el texto, leer pulsaciones de tecla con y sin espera.
- ✓ Conocer la pantalla en Dos/Windows: borrar, escribir en cierta posición, indicar colores para el texto, leer pulsaciones de tecla con y sin espera.
- ✓ Ser capaz de calcular potencias con números reales, raíces cuadradas, expresiones trigonométricas y redondear valores.
- ✓ Saber cómo generar números aleatorios y emplearlos.
- ✓ Leer la fecha y hora del sistema.
- ✓ Acceder a órdenes externas del sistema empleando “system”.

## UNIDAD 9. Ficheros

### CONTENIDOS

#### Conceptos

- ✓ Conceptos teóricos sobre ficheros:
  - ✓ Fichero, registro lógico, campo, subcampo, clave, registro físico (bloque)
  - ✓ Clasificación de registros: de longitud fija, de longitud variable
  - ✓ Operaciones con registros: Altas, bajas, modificaciones, consultas
  - ✓ Clasificación de ficheros: Permanentes, temporales
  - ✓ Operaciones con ficheros: creación, apertura, cierre, consulta, actualización; otras menos habituales.
  - ✓ Organización de ficheros y acceso: organización secuencial, organización relativa: Directa (hash); indirecta o aleatoria (clave)
- ✓ Ficheros en C: el puntero a fichero.
- ✓ Apertura y cierre de ficheros.
- ✓ Acceso secuencial (como carácter, cadenas, formateado o bloques de datos).
- ✓ Acceso directo.
- ✓ Ficheros especiales: la impresora, la salida de errores.

#### Procedimientos

- ✓ Apertura y cierre de ficheros.
- ✓ Acceso secuencial como carácter.
- ✓ Acceso secuencial como cadenas de texto.
- ✓ Acceso secuencial formateado.
- ✓ Acceso secuencial como bloques de datos.
- ✓ Acceso directo: conocer la posición actual y saltar a una cierta posición.
- ✓ Manejo de la impresora
- ✓ Envío de información a la salida de errores.

- ✓ Determinación de la estructura de fichero adecuada a un cierto problema.
- ✓ Realización de programas que manipulen las estructuras de fichero escogidas.

### Actividades de enseñanza-aprendizaje

- ✓ Abrir y cerrar ficheros.
- ✓ Acceso secuencial de lectura y de escritura a un fichero como carácter.
- ✓ Acceso secuencial a un fichero como cadenas de texto.
- ✓ Acceso secuencial formateado.
- ✓ Acceso secuencial como bloques de datos.
- ✓ Saltar a una cierta posición del fichero y conocer la posición actual en que nos encontramos.
- ✓ Enviar información textual a la impresora
- ✓ Enviar información a la salida de errores.

### Criterios de evaluación

- ✓ Ser capaz de acceder secuencialmente para lectura y escritura a un fichero de carácter en carácter.
- ✓ Poder acceder secuencialmente para lectura y escritura a un fichero formado por cadenas de texto.
- ✓ Acceder secuencialmente a un fichero formado por información formateada.
- ✓ Acceso secuencial a un fichero para lectura y escritura bloques de datos.
- ✓ Ser capaz de saltar a una cierta posición del fichero y conocer la posición actual en que se encuentra.
- ✓ Saber cómo enviar información textual a la impresora
- ✓ Poder enviar información a la salida de errores.
- ✓ Ser capaz de determinar la estructura de fichero adecuada a un cierto problema.
- ✓ Ser capaz de realizar un programa que manipule la estructura de fichero escogida.

## UNIDAD 10. Características avanzadas de C

### CONTENIDOS

#### Conceptos

- ✓ Operadores para tratamiento de bits
- ✓ Operador coma
- ✓ Otros tipos de datos: Campos de bits, Uniones, Enumeraciones
- ✓ El preprocesador: concepto y directivas habituales (`#define`, `#include`, `#if`, `#pragma`).
- ✓ Creación de ficheros de cabecera

#### Procedimientos

- ✓ Operaciones a nivel de bit, empleando los operadores adecuados.
- ✓ Uso del operador coma, tanto en for como en asignaciones.
- ✓ Manejo de otros tipos de datos que no se habían empleado hasta ahora: Campos de bits, Uniones, Enumeraciones.
- ✓ Uso de las distintas posibilidades del preprocesador: constantes simbólicas, inclusión de ficheros, compilación condicional, directivas exclusivas del compilador.
- ✓ Creación de ficheros de cabecera.

#### Actividades de enseñanza-aprendizaje

- ✓ Realizar operaciones AND, OR, XOR, NOT a nivel de bit.
- ✓ Usar el operador coma para encadenar asignaciones.
- ✓ Crear y manejar datos enumerados.
- ✓ Crear y manejar uniones.
- ✓ Crear y manejar campos de bits.
- ✓ Definir constantes simbólicas.
- ✓ Usar la directiva `#if` para compilación condicional.
- ✓ Crear nuevos ficheros de cabecera e incluirlos.
- ✓ Aprovechar características avanzadas del compilador usando `#pragma`.

## Criterios de evaluación

- ✓ Saber realizar operaciones AND, OR, XOR, NOT a nivel de bit.
- ✓ Poder definir y manejar datos enumerados, uniones y campos de bits.
- ✓ Saber definir y utilizar constantes simbólicas.
- ✓ Ser capaz de aplicar la directiva #if (y relacionadas) para compilación condicional.
- ✓ Saber crear nuevos ficheros de cabecera y cómo incluirlos en un programa.

## UNIDAD 11. Ejercicios completos

### CONTENIDOS

#### Conceptos

- ✓ Aplicación de los conocimientos previos a la creación de aplicaciones de pequeño tamaño.
- ✓ Mantenimiento de aplicaciones ya creadas.

#### Procedimientos

- ✓ Creación de aplicaciones completas desde cero.
- ✓ Uso del depurador para encontrar fallos en aplicaciones.

#### Actividades de enseñanza-aprendizaje

- ✓ Se propondrá a los alumnos diversos retos, de complejidad creciente, que deberán resolver en forma de programa en lenguaje C.
- ✓ Se propondrá también la corrección, mejora o modificación de programas ya existentes.

## Criterios de evaluación

- ✓ Capacidad para desarrollar un programa completo como respuesta a un problema propuesto.
- ✓ Capacidad para encontrar fallos de problemas ya existentes, o ampliar sus funcionalidades.

## UNIDAD 12. Proyecto final

### CONTENIDOS

#### Conceptos

- ✓ No se aportarán conceptos nuevos, sólo técnicas que ayuden a los alumnos a trabajar en grupo de la forma más coordinada y más eficiente posible.

#### Procedimientos

- ✓ Trabajo en grupo en paralelo (cada alumno desarrollando una parte distinta, que luego se integran).
- ✓ Trabajo en grupo colaborativo (un alumno teclea mientras el otro se sienta a su lado, comprueba la corrección y da ideas; al cabo de cierto tiempo, se intercambian los papeles).

#### Actividades de enseñanza-aprendizaje

- ✓ Realización de un proyecto de mayor dificultad, gracias al trabajo colaborativo de varios alumnos. Los propios alumnos podrán proponer el ejercicio que desean realizar, si bien el profesor tendrá la última palabra en caso de que no se proponga nada, o se trate de problemas de dificultad demasiado alta o demasiado baja.

#### Criterios de evaluación

- ✓ Se valorará la dificultad del proyecto escogido, su resolución y el trabajo realizado por cada uno de los integrantes del grupo.

## 6. TEMPORALIZACIÓN

TEMA	HORAS
0. Conceptos básicos sobre programación	3
1. Toma de contacto con C	10
2. Tipos de datos básicos	15
3. Estructuras de control	24
4. Entrada/salida básica	10
5. Arrays y estructuras	18
6. Funciones	24
7. Punteros y gestión dinámica de memoria	22
8. Librerías de uso frecuente	27
9. Ficheros	35
10. Características avanzadas de C	30
11. Ejercicios completos	20
12. Proyecto globalizador y final	50
	288

Esta temporalización no se seguirá en orden estrictamente lineal, alternando en ocasiones conceptos de temas más teóricos con conceptos de temas posteriores más prácticos, a fin de que las clases resulten más dinámicas. Es el caso del tema 2 y tema 3, que realmente se solaparán en el tiempo.

## 7. BIBLIOGRAFÍA

El manual que cubrirá gran parte de los contenidos es el siguiente:

- "Aprenda ANSI C como si estuviera en primero", de la Escuela Superior de Ingenieros Industriales, Universidad de Navarra.

Para el proyecto globalizador se utilizará el siguiente manual:

- Programación de videojuegos con SDL, por Alberto García Serrano.

Adicionalmente, se recomendará las siguientes lecturas adicionales a los alumnos que deseen profundizar:

- Quero E., *Fundamentos de programación*, Paraninfo 2001
- 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
- Kernighan, B., Ritchie, D., *El lenguaje de programación C*, Prentice-Hall

Así como otros documentos disponibles en Internet, entre los que se puede citar:

- Curso de C por Angel Salas (Universidad de Zaragoza).
- Curso de C (en línea) de "El rincón del C".
- Curso de C, por Nacho Cabanes.

## 8. PLANTEAMIENTO DE LA ATENCIÓN A LA DIVERSIDAD

Es evidente que el ritmo de desarrollo de las capacidades no tiene por qué ser el mismo en todo un colectivo como es el grupo de alumnas y alumnos. En un proceso de aprendizaje en el que lo principal o exclusivo es la adquisición de conocimientos, las adaptaciones curriculares a los diferentes ritmos de aprendizaje deben realizarse actuando sobre el método (entendido aquí como un elemento curricular más), proponiendo actividades diversas que conduzcan a metas semejantes.

Por ello, existirá una serie de conceptos teóricos y de ejercicios prácticos que todos los alumnos deberán realizar.

Para aquellos alumnos y alumnas con nivel elevado de conocimientos o con un ritmo de enseñanza-aprendizaje más rápido, se plantearán una serie de actividades de ampliación.

Finalmente, también se plantearán actividades que puedan servir de refuerzo para aquellos alumnos y alumnas con un menor ritmo de aprendizaje.

## 9. CRITERIOS DE EVALUACIÓN

Los criterios específicos de evaluación de los contenidos de cada unidad temática se han indicado con anterioridad. Aquí se hace referencia a la evaluación global del curso.

La **calificación** de los alumnos se valorará según los siguientes porcentajes:

- Actividades de enseñanza aprendizaje..... 10%
- Actividades específicas de evaluación..... 40%
- Proyecto globalizador..... 50%

Además para superar un módulo es necesario:

- Que las ausencias a clase no superen el 20% del horario lectivo, según lo regulado en el Reglamento de Régimen Interior de Centro.
- No tener actitudes contrarias a las normas de convivencia.
- Haber realizado satisfactoriamente las actividades programadas como indispensables, por el profesor.
- En el proyecto globalizador deberá haber obtenido al menos la calificación de 5 sobre 10.

Con relación a las faltas de asistencia no justificadas, cada una penalizará rebajando en 0,25 puntos la nota de la correspondiente evaluación, como forma efectiva de que un 20% de las faltas (20 horas de las casi 100 que integran cada evaluación) supongan un suspenso casi automático en la evaluación en cuestión.

Durante la evaluación se realizarán frecuentes ejercicios con nota, como forma de comprobar la evolución del alumnado. Al final de cada evaluación se realizarán actividades teóricas y prácticas de evaluación que será necesario superar (al menos obtener 5 sobre 10) para aprobar esa evaluación. Se realizará una recuperación de estas pruebas de la 1ª y 2ª evaluación para los alumnos que, habiéndose presentado a ellas, no las hayan superado. Deberán presentarse a una prueba final en junio los alumnos que tengan pendientes alguna de las evaluaciones.

## 10. ACTIVIDADES COMPLEMENTARIAS Y EXTRAESCOLARES.

Sería conveniente, si existiera la posibilidad, visitar las instalaciones de alguna empresa de desarrollo de software.

Se encuentran programadas visitas a la Universidad para participar en talleres de desarrollo de videojuegos con librerías XNA.

## 11. MATERIALES Y RECURSOS DIDÁCTICOS.

Los materiales y recursos que se emplearán en la asignatura son:

HARDWARE: Un aula equipada con

- Un servidor Pentium IV.
- Veinticinco estaciones de trabajo Pentium IV conectadas en red.
- Una impresoras láser.
- Dos switches de 24 puertos en el aula de mayor tamaño.
- Un sistema de proyección (proyector SVGA y pantalla).
- Conexión a Internet ADSL.

SOFTWARE:

- Sistemas operativos en red: Windows XP, Linux Kubuntu.
- Entornos de desarrollo: MinGW y Code::Blocks para Windows, y GCC y Kdevelop para Linux.