

# Programación

## 1º DAM

Programación didáctica

Curso: 2018/2019

Departamento de Informática

*Antoni Salort Frasés*

*Nacho Cabanes*

*Nacho Iborra*

## Índice de contenidos

<b>1. Introducción</b>	3
<b>1.1. Contextualización</b>	3
<b>2. Objetivos</b>	4
<b>2.1. Resultados de aprendizaje</b>	4
<b>2.2. Competencias profesionales, personales y sociales</b>	5
<b>3. Contenidos</b>	7
<b>3.1. Secuenciación y temporización</b>	27
<b>4. Metodología didáctica</b>	29
<b>5. Evaluación</b>	30
<b>5.1. Criterios de evaluación</b>	30
<b>5.2. Criterios de calificación</b>	31
<b>5.3. Actividades de refuerzo y ampliación</b>	33
<b>5.4. Evaluación del proceso de enseñanza y aprendizaje</b>	33
<b>6. Criterios de recuperación</b>	35
<b>6.1. Alumnos pendientes</b>	35
<b>7. Medidas de atención a la diversidad y alumnos con N.E.E.</b>	36
<b>8. Fomento de la lectura</b>	38
<b>9. Recursos didácticos</b>	39
<b>10. Bibliografía de referencia</b>	40
<b>11. Actividades complementarias y extraescolares</b>	41
<b>12. Enseñanza bilingüe</b>	42

# 1. Introducción

La siguiente programación didáctica tratará de establecer los conceptos básicos teóricos y prácticos, así como los objetivos que se pretenden alcanzar en el desarrollo del módulo de Programación que se imparte en el primer curso del Ciclo Formativo de grado superior de Desarrollo de Aplicaciones Multiplataforma (DAM). El citado módulo tiene una duración total de 256 horas.

## 1.1. Contextualización

Esta programación está orientada teniendo en cuenta las características del centro en el que se imparte. Estas características son:

- Centro Público, ubicado en un núcleo urbano con una población que ronda los 55.000 habitantes, donde acuden numerosos alumnos de zonas cercanas con menor población en régimen diurno y vespertino.
- El municipio dispone de gran cantidad de empresas del sector servicios que satisfacen las necesidades de todo el sector industrial de la zona. Ante esta situación, existe una creciente demanda de profesionales que sean capaces de desarrollar aplicaciones informáticas, y que son demandados tanto por las industrias como por las empresas de servicios.
- Las asignaturas y los módulos de informática llevan impartándose en este centro diversos años, por lo que está dotado de todos los recursos necesarios para llevar a cabo los contenidos.
- Es un centro ubicado en un municipio muy cercano a una gran ciudad por lo que cuenta con amplias redes de transporte, que facilitarán las posibilidades de desplazamiento para el caso de actividades extraescolares y complementarias, con una amplia oferta cultural.
- En cuanto a la climatología será apacible, propia de la Comunidad Valenciana, que evitará en parte el absentismo escolar.

## 2. Objetivos

En el Real Decreto 450/2010 se indican los objetivos generales del ciclo formativo, de los que se puede extraer, considerando el contexto en el que se impartirá el módulo, los siguientes:

- Seleccionar y emplear lenguajes, herramientas y librerías, interpretando las especificaciones para desarrollar aplicaciones multiplataforma con acceso a bases de datos.
- Gestionar la información almacenada, planificando e implementando sistemas de formularios e informes para desarrollar aplicaciones de gestión.
- Seleccionar y utilizar herramientas específicas, lenguajes y librerías, evaluando sus posibilidades y siguiendo un manual de estilo, para manipular e integrar en aplicaciones multiplataforma contenidos gráficos y componentes multimedia.
- Emplear herramientas de desarrollo, lenguajes y componentes visuales, siguiendo las especificaciones y verificando interactividad y usabilidad, para desarrollar interfaces gráficos de usuario en aplicaciones multiplataforma.
- Seleccionar y emplear técnicas, motores y entornos de desarrollo, evaluando sus posibilidades, para participar en el desarrollo de juegos y aplicaciones en el ámbito del entretenimiento.
- Instalar y configurar módulos y complementos, evaluando su funcionalidad, para gestionar entornos de desarrollo.
- Verificar los componentes software desarrollados, analizando las especificaciones, para completar un plan de pruebas.
- Establecer procedimientos, verificando su funcionalidad, para desplegar y distribuir aplicaciones.

Que se podría plasmar en los siguientes objetivos más concretos para este módulo:

- OM01 Aplicar estrategias de programación estructurada y modular, y de programación orientada a objetos para la resolución de problemas con independencia del lenguaje de programación a utilizar.
- OM02 Identificar estructuras de datos necesarias para la resolución del problema con un lenguaje estructurado.
- OM03 Codificar un módulo de programación en lenguaje C#, empleando las construcciones modulares proporcionadas por dichos lenguajes (funciones, clases, objetos, etc.) y definiendo estructuras de datos apropiadas.
- OM04 Documentar el código desarrollado con comentarios significativos, concisos y legibles.
- OM05 Integrar y enlazar módulos de programación y librerías.
- OM06 Obtener código ejecutable para sistemas operativos Windows y Linux, empleando para ello las distintas opciones de los compiladores, enlazadores, y gestores de configuraciones.
- OM07 Depurar los módulos de programación manejando herramientas específicas.
- OM08 Planificar la realización de una fase de pruebas a partir de las especificaciones establecidas en el diseño y de los resultados esperados para cada módulo.
- OM09 Realizar pruebas para cada módulo de una aplicación y pruebas de integración, detectando errores en la funcionalidad o en la presentación (formato) de los datos de entrada y salida.
- OM10 Medir los rendimientos de la aplicación y evaluar la eficiencia de las prestaciones de la aplicación y el consumo de recursos.

### 2.1. Resultados de aprendizaje

- a. Aplicar estrategias de programación estructurada y modular, y de programación orientada a objetos para la resolución de problemas con independencia del lenguaje de programación a utilizar.
- b. Identificar estructuras de datos necesarias para la resolución del problema con un lenguaje estructurado.
- c. Codificar un módulo de programación en lenguaje C#, empleando las construcciones modulares proporcionadas por dichos lenguajes (funciones, clases, objetos, etc.) y definiendo estructuras de datos apropiadas.
- d. Documentar el código desarrollado con comentarios significativos, concisos y legibles.
- e. Integrar y enlazar módulos de programación y librerías.
- f. Obtener código ejecutable para sistemas operativos Windows y Linux, empleando para ello las distintas opciones de los compiladores, enlazadores, y gestores de configuraciones.
- g. Depurar los módulos de programación manejando herramientas específicas.
- h. Planificar la realización de una fase de pruebas a partir de las especificaciones establecidas en el diseño y de los resultados esperados para cada módulo.
- i. Realizar pruebas para cada módulo de una aplicación y pruebas de integración, detectando errores en la funcionalidad o en la presentación (formato) de los datos de entrada y salida.
- j. Medir los rendimientos de la aplicación y evaluar la eficiencia de las prestaciones de la aplicación y el consumo de recursos.
- k. Provocar y verificar los diversos tratamientos de error.
- l. Elaborar documentación útil sobre la arquitectura y algoritmos diseñados.
- m. Documentar y describir las estructuras de datos utilizadas.
- n. Redactar guías de uso de las aplicaciones.
- o. Identificar los datos y módulos de programación afectados por la modificación de los requerimientos.
- p. Probar que los nuevos datos y módulos no producen pérdidas de eficiencia ni funcionalidad de la aplicación y satisfacen los nuevos requerimientos funcionales.
- q. Actualizar la documentación con los cambios realizados sobre los módulos y estructuras de datos de la aplicación.
- r. Acceder a bases de datos relacionales desde programas realizados usando el lenguaje C#.
- s. Usar la persistencia para almacenar información sobre los objetos empleados por una aplicación.

## 2.2. Competencias profesionales, personales y sociales

La referencia del sistema productivo de este Módulo la encontramos en el Real Decreto 450/2010, de 16 de abril, que especifica, entre otras, las siguientes competencias profesionales, personales y sociales:

- Gestionar entornos de desarrollo adaptando su configuración en cada caso para permitir el desarrollo y despliegue de aplicaciones.
- Desarrollar aplicaciones multiplataforma con acceso a bases de datos utilizando lenguajes, librerías y herramientas adecuados a las especificaciones.
- Desarrollar aplicaciones implementando un sistema completo de formularios e informes que permitan gestionar de forma integral la información almacenada.
- Realizar planes de pruebas verificando el funcionamiento de los componentes software desarrollados, según las especificaciones.
- Desplegar y distribuir aplicaciones en distintos ámbitos de implantación verificando su comportamiento y realizando las modificaciones necesarias.

- Gestionar su carrera profesional, analizando las oportunidades de empleo, autoempleo y de aprendizaje.
- Mantener el espíritu de innovación y actualización en el ámbito de su trabajo para adaptarse a los cambios tecnológicos y organizativos de su entorno profesional.
- Participar de forma activa en la vida económica, social y cultural, con una actitud crítica y responsable.
- Aceptar las normas de comportamiento y trabajo establecidas.
- Participar activamente en los debates y en la formación de grupos de trabajo.
- Valorar la evolución de la técnica para adaptarse al puesto de trabajo.
- Perseverar en la búsqueda de soluciones.
- Valorar la constancia y el esfuerzo propio y ajeno en la realización del trabajo.
- Utilizar los equipos y programas informáticos cumpliendo las normas de seguridad e higiene y requisitos legales.
- Valorar a utilización de técnicas y procedimientos para mantener la seguridad, integridad y privacidad de la información
- Mostrar interés por la utilización correcta del lenguaje informático.
- Realizar su trabajo de forma autónoma y responsable.
- Responsabilizarse de la ejecución de su propio trabajo y de los resultados obtenidos.
- Orden y método en la realización de las tareas.
- Mostrar gusto por una presentación limpia y ordenada de los resultados de los trabajos realizados.
- Demostrar interés por la conclusión total de un trabajo antes de comenzar el siguiente.
- Respeto por otras opiniones, ideas y conductas. Tener conciencia de grupo, integrándose en un grupo de trabajo, participando activamente en las tareas colectivas y respetando las opiniones ajenas.
- Respetar la ejecución del trabajo ajeno en el grupo, compartiendo responsabilidades derivadas del trabajo global.
- Valorar el trabajo en equipo como el medio más eficaz para la realización de ciertas actividades.
- Mantener actitudes de solidaridad y compañerismo.

### 3. Contenidos

Los contenidos básicos que propone el Real Decreto son los siguientes:

- Identificación de los elementos de un programa informático:
  - Estructura y bloques fundamentales.
  - Variables.
  - Tipos de datos.
  - Literales.
  - Constantes.
  - Operadores y expresiones.
  - Conversiones de tipo.
  - Comentarios.
  - Utilización de objetos:
    - Características de los objetos.
    - Instanciación de objetos.
    - Utilización de métodos.
    - Utilización de propiedades.
    - Utilización de métodos estáticos.
    - Constructores.
    - Destrucción de objetos y liberación de memoria.
  
- Uso de estructuras de control:
  - Estructuras de selección.
  - Estructuras de repetición.
  - Estructuras de salto.
  - Control de excepciones.
  
- Desarrollo de clases:
  - Concepto de clase.
  - Estructura y miembros de una clase.

- Creación de atributos.
- Creación de métodos.
- Creación de constructores.
- Utilización de clases y objetos.
- Utilización de clases heredadas.
- Lectura y escritura de información:
  - Tipos de flujos. Flujos de bytes y de caracteres.
  - Clases relativas a flujos.
  - Utilización de flujos.
  - Entrada desde teclado.
  - Salida a pantalla.
  - Ficheros de datos. Registros.
  - Apertura y cierre de ficheros. Modos de acceso.
  - Escritura y lectura de información en ficheros.
  - Utilización de los sistemas de ficheros.
  - Creación y eliminación de ficheros y directorios.
  - Interfaces.
  - Concepto de evento.
  - Creación de controladores de eventos.
- Aplicación de las estructuras de almacenamiento:
  - Estructuras.
  - Creación de arrays.
  - Arrays multidimensionales.
  - Cadenas de caracteres.
  - Listas.
- Utilización avanzada de clases:
  - Composición de clases.
  - Herencia.
  - Superclases y subclases.



- Clases y métodos abstractos y finales.
- Sobreescritura de métodos.
- Constructores y herencia.
- Mantenimiento de la persistencia de los objetos:
  - Bases de datos orientadas a objetos.
  - Características de las bases de datos orientadas a objetos.
  - Instalación del gestor de bases de datos.
  - Creación de bases de datos.
  - Mecanismos de consulta.
  - El lenguaje de consultas: sintaxis, expresiones, operadores.
  - Recuperación, modificación y borrado de información.
  - Tipos de datos objeto; atributos y métodos.
  - Tipos de datos colección.
- Gestión de bases de datos relacionales:
  - Establecimiento de conexiones.
  - Recuperación de información.
  - Manipulación de la información.
  - Ejecución de consultas sobre la base de datos.

## **Concreción**

Estos contenidos básicos propuestos por el Real Decreto se redistribuirán, de modo que su aprendizaje en clase sea progresivo y permita la realización de prácticas desde el primer día. Así, 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. Programación orientada a objetos
3. Acceso a ficheros, persistencia y a acceso a bases de datos
4. Otras características de los lenguajes de programación actuales

## 5. Proyecto final

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

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

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.

Se comenzará relacionando la sintaxis de las estructuras de datos simples, y las sentencias elementales de este lenguaje, comparando con el pseudocódigo. 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: Programación orientada a objetos

Una vez que el alumno conoce la programación estructurada, se le introducirá en la programación orientada a objetos, que le permitirá usar técnicas de diseño más naturales y descomponer problemas de mayor tamaño.

### Bloque 3: Almacenamiento de la información

Como distintas alternativas para que la información quede almacenada de forma permanente, se tratará el acceso a ficheros, la persistencia de objetos y el acceso a bases de datos relacionales.

### Bloque 4: Otras características de los lenguajes de programación actuales

Es interesante el alumno adquiera las nociones básicas de otras características de los lenguajes actuales, como el acceso a la fecha y hora del sistema, la conexión mediante red o a un servidor web, la generación de números aleatorios, el acceso avanzado a la consola o la creación visual de interfaces gráficas. Algunos de estos temas se verán con más detalle en asignaturas de segundo curso, por lo que este bloque apenas tendrá carácter introductorio.

### Bloque 5: Proyecto final

Como forma de que los alumnos apliquen todos los conocimientos adquiridos a un proyecto de programación real, la última parte se dedicará a realizar un proyecto individual, de forma casi totalmente independiente.

## **Elementos curriculares de cada unidad.**

### **Planteamiento de las unidades temáticas**

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

Esta unidad tiene un doble 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.

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

#### UT.2. 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, así como nociones básicas de control de excepciones y depuración de código

#### UT.3. 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, los caracteres o los booleanos, además de otros tipos como las enumeraciones, y una primera toma de contacto con las cadenas de texto.

#### UT.4. Arrays y estructuras

Se muestran al alumno 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 profundiza más en el uso de las cadenas de texto.

#### UT.5. Funciones

El uso de funciones es una característica muy importante en cualquier convencional, pues permite realizar un desarrollo modular del programa al tiempo que facilita su mantenimiento y futuro uso en otras aplicaciones. Por eso, se dedica un tema a la creación y uso de funciones, todavía sin hablar de programación orientada a objetos, de modo que se trate de conocimientos aplicables a otros lenguajes de programación.

#### UT.6. Introducción a la programación orientada a objetos

Se introducirá al alumno en conceptos como “objeto” y “clase”, así como su definición y uso desde C#. También se formalizarán conceptos que hasta esta unidad habían quedado en el aire, como los especificadores de acceso.

#### UT.7. Utilización avanzada de clases

En esta unidad se profundizará en detalles más avanzados, como los constructores y destructores, métodos y atributos static, el polimorfismo y la sobrecarga, la creación de arrays de objetos, y las palabras reservadas override y this. Se establecen también las diferencias a nivel de herencia e implementación entre clases, clases abstractas e interfaces

#### UT.8. Gestión dinámica de la memoria

En esta unidad se introducirá al alumno en la problemática de la gestión dinámica de memoria, los punteros y las “colecciones” dinámicas existentes en los lenguajes de creación reciente.

#### UT.9. Ficheros

Se muestra al alumno la forma de conservar la información, ya sea en fichero de texto, guardando datos tipificados (por ejemplo, el contenido de un struct) o datos de cualquier otro tipo, así como la forma de recuperar esa información que se había guardado, y de comprobar los errores que pueden ocurrir en el proceso. Esto servirá también para repasar el concepto de “excepciones”.

#### UT.10. Persistencia de objetos

Como alternativa a las bases de datos relacionales y a los ficheros convencionales, se introducirá al alumno en las nociones básicas de persistencia de objetos, tanto a nivel de estructuras del lenguaje como utilizando bases de datos orientadas a objetos.

#### UT.11. Acceso a bases de datos relacionales

En grandes y medianos proyectos, es mucho más habitual almacenar la información en bases de datos que en ficheros convencionales. Por eso, se introducirá al alumno en la conexión a bases de datos y la forma de introducir, recuperar y modificar información.

#### UT.12. Bibliotecas de uso frecuente

El alumno necesitará con frecuencia recurrir a bibliotecas de funciones ya existentes y que todavía no se le han presentado, por ejemplo para realizar operaciones matemáticas y generar números aleatorios, para temporización de procesos, o para el acceso avanzado a la pantalla de texto (consola). Esas bibliotecas serán utilizadas en esta Unidad.

#### UT.13. Proyecto final en lenguaje C#

En esta unidad, los alumnos tendrán que enfrentarse a un proyecto de programación individual de mayor complejidad.

### **Detalle de cada unidad**

## **UNIDAD 1. Toma de contacto con C#**

### **CONTENIDOS**

#### **Conceptos**

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

### **Procedimientos**

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

- Identificación de los lenguajes interpretados y 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 en pseudocódigo.
- 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**

- Distinguir las características de los lenguajes de alto y bajo nivel.
- Describir los tipos de lenguajes, compiladores y traductores de uso más común.
- Ser capaz de expresar algoritmos elementales en pseudocódigo.
- 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. 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
- Introducción a las excepciones
- Introducción a la depuración de código

#### 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.
- Interceptación de errores en tiempo de ejecución mediante el mecanismo de excepciones.
- Establecer mecanismos sencillos de depuración de código mediante puntos de ruptura y análisis de expresiones y variables

#### 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.
- Interceptar errores básicos en tiempo de ejecución (conversiones de tipo) usando control de excepciones.
- Depurar el código de un programa incorrecto averiguando dónde está el posible error por ejecución paso a paso

### **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.
- Interceptar de forma correcta errores básicos de conversión de tipo usando control de excepciones.
- Detectar errores sencillos en programas mediante mecanismos básicos de depuración de código

## **UNIDAD 3. 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: +=
- Tipo de dato real: Simple y doble precisión, cómo mostrar en pantalla.
- Tipo de dato carácter; cómo mostrar en pantalla

- Secuencias de escape
- Introducción a las cadenas de texto
- Tipo enumerado y variables de tipo implícito

## **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.
- Conocer la estructura y uso de una cadena de texto
- Saber aplicar el tipo enumerado a los problemas típicos
- Conocer las variables de tipo implícito

## **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.
- Manejo básico de cadenas de texto
- Uso de enumeraciones para problemas de rangos limitados de valores
- Uso de variables de tipo implícito

## **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.
- Manejar textos simples: leer cadenas de caracteres desde el teclado, mostrarlas en pantalla, asignarles un valor y comparar con otras cadenas.
- Manejar tipos enumerados cuando corresponda hacerlo
- Utilizar variables de tipo implícito y ser conscientes de cuándo no es recomendable utilizarlas

## **UNIDAD 4. Arrays y estructuras**

### **CONTENIDOS**

#### **Conceptos**

- Conceptos básicos sobre tablas
- Arrays unidimensionales
- Arrays bidimensionales
- Arrays multidimensionales
- Arrays indeterminados
- Estructuras
- Estructuras anidadas
- Arrays de estructuras
- Cadenas de texto

#### **Procedimientos**

- Definición de arrays.
- Carga de datos en un array, en la inicialización o posteriormente.
- Acceso a los datos de un array.
- Estructuras: definición, carga de datos y acceso a los datos.
- Manipulación y transformación de cadenas de texto

#### **Actividades de enseñanza-aprendizaje**

- Almacenar datos repetitivos empleando arrays.
- Almacenar datos compuestos empleando estructuras.
- Combinar el uso de estructuras y arrays para datos repetitivos formados por unidades más complejas.
- Procesado de cadenas de texto para obtener o modificar la información que contienen.

#### **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..

- Combinar el uso de estructuras y arrays para datos repetitivos formados por unidades más complejas.

## **UNIDAD 5. Funciones**

### **CONTENIDOS**

#### **Conceptos**

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

#### **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
- Obtención de tipos básicos con TryParse para capturar excepciones
- Pruebas simples de programas mediante aserciones

#### **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.
- Obtención de distintos tipos básicos capturando excepciones en una sola línea con TryParse
- Probar el funcionamiento de programas sencillos mediante aserciones

#### **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.
- Saber utilizar TryParse para el parseo de datos simples libre de excepciones
- Saber emplear aserciones para pruebas sencillas del funcionamiento de programas

## **UNIDAD 6. Introducción a la programación orientada a objetos**

### **CONTENIDOS**

#### **Conceptos**

- Conceptos de “objeto” y “clase”
- Definición de clases y de objetos desde C#
- Especificadores de acceso
- Métodos y atributos. Propiedades.
- Encapsulación. Herencia.
- Nociones básicas de UML. Generadores de código.

#### **Procedimientos**

- Descripción de un enunciado en términos de clases, objetos y relaciones de herencia.
- Elaboración de diagramas de clases para especificar la solución a un problema concreto.
- Desarrollo de programas en que se apliquen los conceptos de la programación orientada a objetos.

#### **Actividades de enseñanza-aprendizaje**

- Extraer las clases, objetos y relaciones de herencia existentes en un problema real.
- Elaborar diagramas de clases para especificar la solución a un problema real.
- Desarrollar programas aplicando los conceptos de la programación orientada a objetos.
- Rediseñar programas realizados anteriormente.

#### **Criterios de evaluación**

- Manejar con soltura conceptos como los de clase, herencia, encapsulación, polimorfismo y sobrecarga.
- Ser capaz de extraer las clases, objetos y relaciones de herencia en un enunciado basado en el mundo real.
- Elaborar diagramas de clases para especificar la solución a un problema real.

- Saber desarrollar programas aplicando los conceptos de la programación orientada a objetos.
- Poder rediseñar programas realizados en puntos anteriores del curso, aplicando en ellos las técnicas de la programación orientada a objetos.

## **UNIDAD 7. Utilización avanzada de clases**

### **CONTENIDOS**

#### **Conceptos**

- Composición de clases
- Herencia, superclases y subclases
- Polimorfismo. Sobrecarga.
- Sobreescritura de métodos.
- Métodos abstractos y finales.
- Constructores y herencia.
- Clases abstractas e interfaces
- Arrays de objetos.
- La palabra "this"
- Sobrecarga de operadores

#### **Procedimientos**

- Resolución de problemas más complejos que incluyan herencia a varios niveles.
- Reescritura de métodos en subclases.
- Herencia en constructores.
- Uso de clases abstractas e interfaces para código incompleto o por definir
- Diferencias entre un array de struct y uno de objetos. Forma de uso de los arrays de objetos.
- Mayor legibilidad del código fuente, gracias a la sobrecarga de operadores

#### **Actividades de enseñanza-aprendizaje**

- A partir de diagramas de clases, plasmar fuentes que incluyan herencia a varios niveles.
- Distinguir mediante casos prácticos cuándo utilizar clases normales, clases abstractas o interfaces
- Crear subclases que redefinan algunos de los métodos existentes en las superclases.
- Crear constructores que se apoyen en otros constructores o en otros métodos de la superclase.
- Crear arrays de objetos.
- Crear clases que contengan operadores sobrecargados.

#### **Criterios de evaluación**

- Ser capaz de crear programas con más de un nivel de herencia.
- Poder redefinir correctamente métodos existentes en las superclases.
- Identificar cuándo emplear clases abstractas o interfaces en la resolución de un problema
- Saber crear constructores que se apoyen en otros constructores de la misma clase.
- Poder crear constructores que se apoyen en otros constructores de la superclase.
- Saber crear, inicializar y consultar arrays de objetos.
- Ser capaz de clases que contengan operadores sobrecargados.

## **UNIDAD 8. Gestión dinámica de la memoria**

### **CONTENIDOS**

#### **Conceptos**

- Memoria dinámica: motivación.
- Pilas
- Colas
- Listas
- Tablas hash
- Enumeradores
- Punteros. Operaciones con punteros.

#### **Procedimientos**

- Creación de pilas usando las estructuras existentes en el lenguaje
- Colas usando las estructuras existentes en el lenguaje
- Listas: nociones generales; ArrayList; SortedList
- Características y uso de las tablas hash
- Empleo de enumeradores
- Acceso a punteros. Incremento y decremento de punteros y sus efectos.

#### **Actividades de enseñanza-aprendizaje**

- Crear pilas usando las estructuras existentes en el lenguaje
- Crear colas usando las estructuras existentes en el lenguaje
- Crear ArrayList, introducir en ellos datos simples y extraerlos
- Introducir y extraer structs y objetos en un ArrayList
- Manipulaciones y recorrido de un ArrayList
- Crear SortedList, introducir datos y extraer datos
- Crear tablas hash, introducir datos y extraer datos
- Empleo de enumeradores para recorrer estructuras dinámicas
- Obtener punteros a datos. Recorrer estructuras usando aritmética de punteros.
- Imitar pilas, colas y listas usando arrays y ArrayList

#### **Criterios de evaluación**

- Saber crear de pilas usando las estructuras existentes en el lenguaje
- Poder crear colas usando las estructuras existentes en el lenguaje
- Manejar correctamente ArrayList, tanto con datos simples como complejos
- Guardar y obtener información de un SortedList
- Saber guardar y leer información de una tabla hash
- Poder emplear enumeradores para recorrer una estructura dinámica
- Ser capaz de imitar pilas, colas y listas usando arrays y ArrayList

## **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)
- Apertura y cierre de ficheros.
- Acceso secuencial (como carácter, cadenas, formateado o bloques de datos).
- Acceso directo.
- Formas de acceso a un fichero binario usando C#

#### **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.
- 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.

### **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.
- 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. Persistencia de objetos**

### **CONTENIDOS**

#### **Conceptos**

- Características de las bases de datos orientadas a objetos
- Creación de bases de datos
- El lenguaje de consultas: sintaxis, expresiones, operadores
- Recuperación, modificación y borrado de información
- Tipos de datos objeto; atributos y métodos
- Tipos de datos colección.

#### **Procedimientos**

- Instalación del gestor de bases de datos.
- Creación de bases de datos.
- Recuperación, modificación y borrado de información.

#### **Actividades de enseñanza-aprendizaje**

- Instalar un gestor de bases de datos orientado a objetos.
- Crear bases de datos.
- Introducir información en la base de datos.
- Recuperación de información de la base de datos.
- Borrado de información de la base de datos.

### **Criterios de evaluación**

- Ser capaz de instalar un gestor de bases de datos orientado a objetos.
- Crear bases de datos.
- Saber introducir información en la base de datos.
- Poder recuperación información de la base de datos.
- Ser capaz de borrar información de la base de datos.

## **UNIDAD 11. Acceso a bases de datos relacionales**

### **CONTENIDOS**

#### **Conceptos**

- Nociones básicas sobre bases de datos relacionales
- Conexiones a bases de datos
- Formas de introducción de información
- Métodos de recuperación de información
- Manipulación de la información: cambios, borrados

#### **Procedimientos**

- Establecimiento de conexiones a una base de datos
- Introducción de información
- Recuperación de la información introducida
- Recuperación de información calculada
- Alteración de la información
- Borrado de la información existente

#### **Actividades de enseñanza-aprendizaje**

- Establecimiento de conexiones a una base de datos previamente creada
- Introducción de información desde un programa
- Recuperación de la información introducida anteriormente
- Recuperación de información calculada a partir de datos existentes
- Alteración de la información que contiene la base de datos
- Borrado de la información existente
- Creación de bases de datos desde programa

#### **Criterios de evaluación**

- Ser capaz de establecer conexiones a una base de datos
- Poder introducción información a la base de datos
- Recuperar información introducida



- Saber obtener información calculada a partir de los datos almacenados
- Ser capaz de modificar la información existente
- Poder borrar la información existente

## **UNIDAD 12. Bibliotecas de uso frecuente**

### **CONTENIDOS**

#### **Conceptos**

- La consola (pantalla en modo texto)
- Nociones básicas de entornos gráficos
- Fecha y hora. Temporización
- El entorno. Llamadas al sistema
- Servicios de red
- Nociones básicas sobre juegos
- Introducción a las expresiones regulares
- Introducción a la librería LINQ

#### **Procedimientos**

- Acceso mejorado a la consola: posicionamiento, colores, lectura directa de teclas
- Creación visual de entornos gráficos. Componentes elementales: botones, etiquetas, casillas de texto, listas.
- Lectura de la fecha y hora del sistema.
- Temporización y pausas
- Acceso a datos sobre el sistema operativo y el entorno de ejecución.
- Llamadas a otras órdenes del sistema
- Conexión con otro ordenador mediante red
- Acceso a un servidor web desde un programa
- Nociones básicas sobre juegos: el bucle de juego, representación de elementos gráficos, detección básica de colisiones, descomposición en clases de objetos que cooperan
- Utilizar expresiones regulares para identificar patrones en un texto
- Emplear LINQ para manipulación de colecciones de datos

#### **Actividades de enseñanza-aprendizaje**

- Escribir texto en colores y en distintas posiciones
- Detener el programa hasta que se pulsen ciertas teclas
- Lectura del teclado sin detener el programa
- Contacto con diseñadores visuales de entornos gráficos
- Introducción de botones, etiquetas, casillas de texto y listas a un interfaz de usuario
- Asignación de eventos a los componentes visuales de un interfaz de usuario
- Lectura de la fecha y hora del sistema
- Realización de pausas temporizadas en un programa

- Acceder a datos sobre el sistema operativo y el entorno de ejecución.
- Llamar a otras órdenes del sistema desde el programa actual
- Establecer un diálogo con otro ordenador mediante una conexión de red
- Acceder a la información de una página web desde un programa
- Crear un esqueleto de juego en el que un elemento gráfico se mueve respondiendo al teclado.
- Ampliar el esqueleto de juego para que un segundo elemento gráfico se mueva de forma independiente.
- Ampliar el esqueleto de juego para añadir detección simple de colisiones.
- Ampliar el esqueleto de juego, descomponiéndolo en varios objetos que cooperan
- Definir y aplicar distintos patrones elementales de expresiones regulares en distintos textos
- Utilizar LINQ para filtrar o mapear datos de determinadas colecciones

### **Criterios de evaluación**

- Ser capaz de acceder a la consola de forma mejorada
- Ser capaz de crear entornos gráficos simples, de modo que el programa responda a las acciones del usuario sobre botones
- Poder leer la fecha y la hora del sistema
- Saber obtener información del entorno
- Llamar a otras órdenes del sistema
- Acceder a información disponible en una página web
- Comunicar con otro ordenador mediante red
- Ser capaz de crear juegos sencillos
- Saber definir expresiones regulares sencillas y aplicarlas a textos para buscar patrones
- Saber emplear el lenguaje LINQ para consultas y filtros sencillos sobre colecciones de datos

## **UNIDAD 13. Proyecto final en lenguaje C#**

### **CONTENIDOS**

#### **Conceptos**

- No se aportarán conceptos nuevos, sólo técnicas que ayuden a los alumnos a trabajar, tanto individualmente como en grupo, de la forma más coordinada y más eficiente posible.
- Procedimientos
- Nociones de planificación de productos software.
- 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 individual o 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 el alumno.

## 3.1. Secuenciación y temporización

En la siguiente tabla se especifican las unidades y el tiempo de clase asociado para la modalidad presencial y semipresencial. La diferencia de tiempo en algunas unidades se debe a la planificación para hacer coincidir los exámenes trimestrales del semipresencial con la finalización aproximada de los temas. Además, el último tema (proyecto final), exige una presencialidad y un seguimiento diario en la modalidad presencial que no se tiene en la semipresencial, por lo que se pueden dedicar más semanas a dar con más calma otros contenidos:

TEMA	SEMANAS (PRESENCIAL)	SEMANAS (SEMIPRES.)
1. Toma de contacto con C#	2	2
2. Estructuras de control	4	4
3. Tipos de datos básicos	2	2
4. Arrays y estructuras	3	3
5. Funciones	3	3
6. Introducción a la programación Orientada a Objetos	2	2
7. Utilización avanzada de clases	3	4
8. Gestión dinámica de memoria	3	3
9. Ficheros	3	3
10. Persistencia de objetos	1	1
11. Acceso a bases de datos relacionales	1	1
12. Bibliotecas de uso frecuente	2	2
13. Proyecto final en lenguaje C#	3	2
	32	32

Dicho tiempo de clase puede variar ligeramente en función de los festivos que se tengan

durante el curso, y de las características del grupo, que puede que requieran de algún día extra más en algún tema para terminar de afianzar conceptos. Además, en la modalidad semipresencial no se dedicarán tantas semanas de clase al proyecto final (tema 13) propiamente dicho, sino que se irá avanzando éste a medida que se avanza en los temas finales. Esto servirá para dejar también alguna semana "extra" para completar ejercicios pendientes de temas importantes que puedan ir más justos de tiempo.

En cuanto a la secuenciación, se impartirán las siguientes unidades en cada trimestre del curso:

**1º trimestre:** unidades 1 a 5 (presencial) y 1 a 4 (semipresencial)

**2º trimestre:** unidades 6 a 9 (presencial) y 5 a 7 (semipresencial).

**3º trimestre:** resto de unidades, junto con el proyecto final que se irá planteando y desarrollando ya desde el 2º trimestre.

La diferencia de tiempos entre la modalidad presencial y semipresencial viene dada por la semana de exámenes para la modalidad semipresencial, que suele ser una o dos semanas antes de la evaluación, dando menos tiempo para finalizar los temas previstos en cada trimestre. En realidad, en la modalidad semipresencial el tema 5 se deja empezado antes de finalizar el 1º trimestre, y el tema 8 se comienza antes de finalizar el segundo.

## 4. Metodología didáctica

El método que se seguirá para el desarrollo de las clases será el siguiente:

Exposición de conceptos teóricos en clase a partir de los apuntes y los materiales complementarios que el profesor estime convenientes para una mejor comprensión de los contenidos.

Planteamiento de ejercicios y actividades en el aula de aquellos temas que lo permitan, resolviéndose aquellos que se estime oportuno.

Presentación en clase de las diversas prácticas a realizar y desarrollo en la misma de todas aquellas que el equipamiento permita.

Realización de trabajos y exposiciones por parte del alumnado, de aquellos temas que se brinden a ello o que se propongan directamente por el profesor, con lo que se conseguirá una activa participación y un mayor acercamiento a los conceptos y contenidos del módulo.

En una etapa inicial del curso, se seguirá el método tradicional de exposición por parte del profesor, con el fin de explicar los conceptos básicos que éste módulo necesita para poder arrancar.

Tan pronto como sea posible, la metodología pasará a ser fundamentalmente procedimental, con la realización de prácticas, invitando al alumno, bien individualmente o bien en grupo, a que exponga su planteamiento ante sus compañeros, para efectuar los pertinentes comentarios, intercambio de pareceres y discusión de las soluciones propuestas.

En la medida de lo posible se tendrá una atención individualizada por parte de los profesores a cada alumno o grupo de alumnos. Se fomentará que cada uno plantee sus dudas o problemas sobre el ejercicio. Finalmente se comentarán las posibles soluciones, así como los fallos y errores que suelen cometerse de forma más habitual.

En el módulo **semipresencial o a distancia**, la metodología necesita ser algo diferente, al carecer de horas de clase propiamente dichas. En este caso, la mayor parte del seguimiento se realizará a través del Aula Virtual. Se publicará en dicha Aula el trabajo a realizar durante cada semana: apuntes de teoría que leer y revisar, y ejercicios a realizar. También cada semana se dedicarán las tutorías colectivas a resolver dudas de carácter general sobre las tareas a realizar esa semana, y a la explicación de conceptos o problemas más complejos que requieran de dichas explicaciones. También se dispondrá de horas de tutoría individual para atender las dudas específicas que tengan los alumnos sobre lo visto hasta ahora. Pero, a diferencia del módulo presencial, en este módulo es importante la parte autodidacta del alumnado, siendo el Aula Virtual la principal vía de comunicación, y las horas de tutoría un complemento para repasar o solucionar lo que haya quedado menos claro.

## 5. Evaluación

A continuación se exponen los criterios de evaluación y calificación, así como las actividades propuestas a realizar como refuerzo o ampliación.

### 5.1. Criterios de evaluación

Para superar el módulo es necesario:

- En el caso del grupo presencial, las ausencias a clase no superarán el 15% 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

En el grupo **presencial**, la asistencia a clase es obligatoria (por ley), debido al elevado contenido práctico del módulo, por tanto, aquellos alumnos que no asistan como mínimo al 85% de las horas, no tendrán derecho a la evaluación continua y deberán realizar un examen final de todo el módulo. Por ello, cada falta de asistencia **no justificada** (en el grupo presencial) penalizará rebajando en 0,35 puntos la nota de la correspondiente evaluación, como forma efectiva de que un 15% de las faltas (13 horas de las cerca de 85 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. Deberán presentarse a una prueba final en junio los alumnos cuya nota media del curso sea inferior a 5,00.

Con relación al proyecto final, es importante insistir en que, como requisito **IMPRESINDIBLE** para superar la asignatura, se deberá completar el proyecto (y su memoria), de modo que el proyecto sea “realmente utilizable”, tenga un nivel de dificultad razonable para un proyecto final, y muestre buenas costumbres de programación (en cuanto a estructuración, uso de comentarios y limpieza de código)

**En la modalidad semipresencial**, las prácticas serán de dos tipos: unas (la mayoría) de autocorrección, donde pasado el plazo de finalización y entrega se publicarán las soluciones en el Aula Virtual para que los propios alumnos revisen su práctica, y otras (normalmente unas 2 o 3 por trimestre) de corrección por parte del profesor. Las primeras no tendrán una calificación

asociada, sino que su entrega es recomendable para llevar el módulo al día.

Además, en relación a las prácticas en el caso de la modalidad semipresencial, debido al gran número de alumnos y entregas semanales de ejercicios que puede haber, se exigirá que dichas entregas cumplan un patrón determinado, en cuanto a ficheros entregados, nombres de los mismos, y estructuración en carpetas, para así facilitar el proceso de corrección, y acostumbrar al alumnado a cumplir unas pautas de entrega que luego también se le podrán exigir en otros módulos o en el mundo laboral. Así, y en cuanto a las prácticas corregidas por el profesor, la segunda entrega de prácticas que no cumpla los requisitos se evaluará al 50% de la nota máxima (es decir, se podrá obtener como mucho un 5), y a partir de la tercera entrega incorrecta, la nota en esa(s) práctica(s) mal entregada(s) será de cero.

En cuanto a la **actitud en la modalidad semipresencial**, obviamente no se puede valorar el comportamiento y participación en clase, pero sí en los foros, de forma que se valorarán positivamente tanto las preguntas meditadas y razonadas que se hagan, como las respuestas con cierta entidad, o comentarios que sean importantes. Quedan fuera de esta valoración preguntas simples (por ejemplo, "¿Cómo se realiza el ejercicio X?") y respuestas triviales ("No lo sé", "Mira en esta web..."), así como cualquier otra forma de participación que sólo busque aparecer en el foro. También se valorará en el apartado de actitud la participación (voluntaria) del alumnado en tareas que se puedan ir proponiendo (concursos o retos de programación, olimpiadas informáticas, etc.).

## 5.2. Criterios de calificación

En la modalidad **presencial**, los baremos detallados son los siguientes:

Calificación del **primer** trimestre:

- 70% nota ponderada de los exámenes de la siguiente manera:
  - Examen tema 0 y 1: 10%
  - Examen tema 2 y 3: 30%
  - Examen tema 4: 30%
  - Examen tema 5: 30%
- 30% ejercicios realizados en clase

Calificación del **segundo** trimestre:

- 40% primera evaluación
- 60% segunda evaluación, desglosada de la siguiente forma:
  - 50% media de los exámenes realizados
  - 50% ejercicios realizados en clase

Calificación **final**:

- 80% de las evaluaciones, ponderadas de la siguiente forma:
- 20% primera evaluación
- 40% segunda evaluación
- 40% tercera evaluación, que se puede desglosar como:
  - 40% media de los exámenes realizados
  - 60% ejercicios realizados en clase
- 10% proyecto globalizador
- 10% actitud

En la modalidad **a distancia**, se han procurado establecer unos criterios de calificación que resulten equivalentes para los alumnos que siguen el curso puntualmente, y aquellos que deciden acudir directamente a las pruebas oficiales de junio. Por otra parte, se pretende que, aquellos alumnos que se examinen por parciales trimestrales, puedan conservar su nota, en caso de ser buena, y no jugárselo todo a una carta en el examen final. Así, los baremos son los siguientes:

En el caso de alumnos de seguimiento regular y trimestral:

Calificación de cada **trimestre**:

- 70% nota de examen, donde:
  - Si la nota de un trimestre es superior a la de algún trimestre anterior, automáticamente se asume esa nota para el trimestre anterior con menos nota
  - Si la nota de un trimestre es inferior a la de algún trimestre anterior, el alumno conserva la nota del antiguo trimestre sin verse disminuida por el nuevo
  - De esta forma, se premia a los alumnos que acuden a exámenes trimestrales y obtienen buena nota, y se da una nueva oportunidad de recuperar a los alumnos que no obtuvieron buenos resultados en trimestres previos pero luego obtienen mejores notas.
- 30% ejercicios obligatorios realizados en el Aula Virtual

Calificación **final**:

- 15% nota del primer trimestre
- 35% nota del segundo trimestre
- 35% nota del tercer trimestre
- 15% proyecto final
- 5% actitud (extra)

En el caso de alumnos que acudan directamente al examen final, su **calificación final** será:

- 85% examen y prácticas, de los cuales:



- 70% nota examen final
- 30% prácticas propuestas
- 15% proyecto final
- 5% actitud (extra)

En ningún caso se podrá aprobar una evaluación si:

- La nota media de prácticas o ejercicios es inferior a 4.
- La nota de exámenes es inferior a 4.
- La nota de una de las prácticas o ejercicios es inferior a 3.

En cuanto a la actitud extra en la modalidad semipresencial, al no poder valorar comportamiento y actitud en clase, se valorará la participación activa en los foros, concursos de programación y propuestas realizadas por los profesores del módulo.

A efectos de la calificación, también se debe tener en cuenta que la copia (de trabajos o en exámenes) es considerada una falta grave, y se tomarán las medidas oportunas en todas las partes implicadas.

### **5.3. Actividades de refuerzo y ampliación**

Se dispone de diversidad de actividades de refuerzo y ampliación por unidad didáctica. Con este tipo de actividades pretendemos dar respuesta a los diferentes ritmos de aprendizaje que presentan los alumnos. Las actividades de refuerzo permitirán que alumnos con un ritmo de aprendizaje menor lleguen a alcanzar las capacidades de la unidad, mientras que las actividades de ampliación permitirán que alumnos con un ritmo de aprendizaje mayor puedan profundizar en los contenidos de la unidad una vez alcanzadas las capacidades.

### **5.4. Evaluación del proceso de enseñanza y aprendizaje**

Para la evaluación del proceso de aprendizaje se tienen, entre otros, los siguientes aspectos:

- La evaluación se realizará tomando como referencia las capacidades y criterios de evaluación establecidos.
- La aplicación del proceso de evaluación continua del alumnado requiere su asistencia regular a las clases y a las actividades.

Para la evaluación del proceso de enseñanza, entre otros, los siguientes aspectos:

**¿Qué evaluar?**

Se debe evaluar la programación, la intervención del profesor, los recursos, los espacios y tiempos previstos, la participación de alumnos, los criterios e instrumentos de evaluación aplicados, etc. Pero además, se debe evaluar la coordinación docente, la adecuación de las decisiones del Proyecto curricular de etapa y la coherencia entre los Proyectos curriculares de cada etapa así como con el Proyecto educativo de centro.

### **¿Cómo evaluar?**

En relación a los procedimientos e instrumentos para la evaluación de la enseñanza, utilizaremos los siguientes:

- El contraste de experiencias con otros compañeros del equipo docente o de otros centros.
- La reflexión a partir del análisis comparativo entre resultados esperados y los obtenidos.
- Los cuestionarios contestados por los propios profesores y por los alumnos sobre asuntos que afecten a la marcha general del centro y del módulo.

### **¿Cuándo evaluar?**

La intervención educativa debe ser continua y conviene tomar datos a lo largo del proceso para hacer los cambios pertinentes en el momento adecuado. No obstante, dadas las características de los diferentes elementos del proceso y de los documentos en que se plasman, hay momentos especialmente indicados para recoger la información que sirve de base para la evaluación.

- La evaluación inicial al comienzo de curso para situar tanto el punto de partida del grupo aula como la del equipo docente, así como los recursos materiales y humanos de que dispone el centro.
- Tras la finalización de cada unidad didáctica para tomar decisiones sobre posibles cambios en la propia unidad o siguientes.
- Al final del módulo, los datos tomados permitirán evaluar y tomar decisiones de modificación de las programaciones.

## 6. Criterios de recuperación

En el bloque de programación estructurada y orientada a objetos, dado que se trata de conocimientos claramente incrementales, en los que cada nuevo tema se apoya en los anteriores, no habrá actividades extraordinarias de recuperación durante el curso: las posibles notas negativas se deberán compensar con ejercicios y exámenes posteriores (que, además, tendrán más ponderación a medida que avance el curso). Si aun así, algún alumno llega a final de curso sin haber logrado una media de aprobado, podrá realizar el examen final de junio, en el que se le evaluará de todos los conocimientos adquiridos a lo largo del curso. En caso de suspender dicho examen final, se propondrá al alumno una serie de actividades a realizar, y deberá realizar un examen final extraordinario, también de todo el contenido de este bloque. Tanto en el caso de tener que realizar el examen final de junio como el extraordinario, el alumno deberá realizar también un proyecto final, con una carga cercana a las 30 horas, que ponga en práctica la mayor parte de lo aprendido durante el curso. El contenido de dicho proyecto final deberá ser aceptado previamente por el profesor. Además, el profesorado puede requerir que el alumno entregue algunas prácticas que tenga pendientes de aprobar.

Concretando más el caso de la **modalidad semipresencial**:

- En el examen de la 3ª evaluación se incluirán, además de los contenidos propios de dicha evaluación, los de las evaluaciones previas, de forma que será un examen global del módulo. Para aquellos alumnos que tengan aprobados los parciales trimestrales previos, este examen sólo les ponderará como trimestral, teniendo ya ganada una parte de la nota previa. Los alumnos que hayan suspendido, o no se hayan presentado, a los exámenes trimestrales, se jugarán el 100% de la nota de examen con esta prueba final.
- En cuanto a las prácticas, se propondrán prácticas de recuperación de la 1ª y 2ª evaluaciones, a entregar durante la 3ª, para aquellos alumnos que no pudieron entregarlas en su día, o las suspendieron con menos de un 4.

Esto posibilitará que los alumnos que no hayan podido llevar el módulo al día, y presentar los trabajos y exámenes trimestrales, tengan también opción de superar la asignatura, acumulando todo el trabajo exigido hasta el final de curso.

### 6.1. Alumnos pendientes

En el caso de la modalidad presencial, el número de horas del módulo impide que el alumnado lo tenga pendiente mientras cursa segundo.

En el caso de la modalidad semipresencial, los alumnos con el módulo pendiente serán evaluados como si se tratara de una matriculación ordinaria (y no un módulo pendiente), por lo que no cabe ninguna distinción respecto al resto de alumnos.

## 7. Medidas de atención a la diversidad y alumnos con N.E.E.

### Introducción y objetivos

Esta etapa educativa debe atender las necesidades educativas de los alumnos y alumnas, tanto de los que requieren un refuerzo porque presentan ciertas dificultades en el aprendizaje como de aquellos cuyo nivel esté por encima del habitual.

Escalonar el acceso al conocimiento y graduar los aprendizajes constituye un medio para lograr responder a la diversidad del alumnado, de manera que se puedan valorar progresos parciales. Representa también un factor importante el hecho de que los alumnos y alumnas sepan qué es lo que se espera de ellos.

De los objetivos generales del módulo, se tendrá en cuenta que, la adquisición de las capacidades presentará diversos grados, en función de esta diversidad del alumnado.

Por último será el profesor o profesora el que adopte la decisión de que objetivos, contenidos, metodología, actividades, instrumentos y criterios de evaluación adaptará según las características del alumnado de los grupos que imparta.

### Metodología

La atención a la diversidad es uno de los elementos fundamentales a la hora del ejercicio de la actividad educativa, pues se trata de personalizar el proceso de enseñanza-aprendizaje, adecuándolo a las necesidades y al ritmo de trabajo y desarrollo del alumnado.

Se pueden ofrecer vías para la atención a la particular evolución de los alumnos y alumnas, tanto proponiendo una variada escala de dificultad en sus planteamientos y actividades como manteniendo el ejercicio reforzado de las habilidades básicas. La atención a la diversidad se podrá contemplar de la siguiente forma:

- Desarrollando **cuestiones de diagnóstico previo**, al inicio de cada unidad didáctica, para detectar el nivel de conocimientos y de motivación del alumnado que permita valorar al profesor el punto de partida y las estrategias que se van a seguir. Conocer el nivel del que partimos nos permitirá saber qué alumnos y alumnas requieren unos conocimientos previos antes de comenzar la unidad, de modo que puedan abarcarla sin dificultades. Asimismo, sabremos qué alumnos y alumnas han trabajado antes ciertos aspectos del contenido para poder emplear adecuadamente los criterios y actividades de ampliación, de manera que el aprendizaje pueda seguir adelante.
- Incluyendo **actividades de diferente grado de dificultad**, bien sean de contenidos mínimos, de ampliación o de refuerzo o profundización, permitiendo que el profesor seleccione las más oportunas atendiendo a las capacidades y al interés de los alumnos

y alumnas.

- Ofreciendo **textos de refuerzo o de ampliación** que constituyan un complemento más en el proceso de enseñanza-aprendizaje.
- Programando **actividades de refuerzo** cuando sea considerado necesario para un seguimiento más personalizado.

## 8. Fomento de la lectura

A fin de que el alumno desarrolle su comprensión lectora, se aplicarán estrategias que le faciliten su consecución:

- Favorecer que los alumnos activen y desarrollen sus conocimientos previos.
- Permitir que el alumno busque por sí solo la información, jerarquice ideas y se oriente dentro de un texto.
- Activar sus conocimientos previos tanto acerca del contenido como de la forma del texto.
- Relacionar la información del texto con sus propias vivencias, con sus conocimientos, con otros textos, etc.
- Jerarquizar la información e integrarla con la de otros textos.
- Reordenar la información en función de su propósito.
- Ayudar a que los alumnos elaboren hipótesis sobre el tema del texto que se va a leer con apoyo de los gráficos o imágenes que aparecen junto a él.
- Realizar preguntas específicas sobre lo leído.
- Formular preguntas abiertas, que no puedan contestarse con un sí o un no.
- Coordinar una discusión acerca de lo leído.

Para la enseñanza y el aprendizaje de la lectura vamos a trabajar con:

- Lectura de textos cortos relacionados con el tema y preguntas relacionadas con ellas.
- Lectura de materiales que se habilitarán en la plataforma moodle del centro educativo.
- Lectura en voz alta motivadora de materiales de clase con su explicación correspondiente.
- Lectura silenciosa que antecede a la comprensión, estudio y memorización.
- Lectura de periódicos y comentarios en clase de informaciones relacionadas con la materia.

En cada sesión se dedicarán entre 10-15 minutos a la lectura de textos relacionados con los contenidos de la unidad que se esté tratando, tanto aquellos provistos por los libros y materiales, como los elaborados por los propios alumnos (ejercicios realizados como deberes para casa, actividades de investigación, etc.). Se incrementará el tiempo en función del nivel de progresión de los grupos.

Diseño y aplicación de las estrategias de comprensión lectora:

- Se realizarán actividades en cada unidad didáctica leyendo individualmente para ejercitar la comprensión.

## **9. Recursos didácticos**

El material necesario para impartir este módulo es cuantioso. Por un lado se dispone de un aula específica de informática con al menos 20 ordenadores conectados en red y un servidor, que permitirán la realización de prácticas sobre los sistemas operativos de las familias Microsoft y Linux. En el aula hay también pizarra de plástico, para evitar el polvo de tiza. Se contará, así mismo, con un proyector conectado al ordenador del profesor, lo que ayudará a las exposiciones y a la ejemplificación directa sobre el ordenador cuando sea necesario.

Por otro lado, se debe disponer de acceso a Internet desde cualquier ordenador para las numerosas prácticas que lo requieren. Incluso se deberá disponer de espacio Web.

## **10. Bibliografía de referencia**

Material proporcionado por el departamento en la plataforma Moodle.



## **11. Actividades complementarias y extraescolares**

Se fomentará entre el alumnado la labor de investigación personal sobre los diferentes temas tratados a lo largo del curso y la realización de actividades complementarias que permitan conocer casos reales de implantación de los diversos aspectos abordados en el módulo.

Además, se propondrán visitas a exposiciones, organismos o empresas del entorno en los que los alumnos puedan observar en la práctica los aspectos teóricos vistos. En todo caso, estas visitas dependerán de las posibilidades que se vayan descubriendo en el entorno y de cómo se vaya desarrollando el módulo a lo largo del curso.

También se asistirá a diversas charlas y exposiciones realizadas por expertos en el propio centro.

## 12. Enseñanza bilingüe

En el citado Real Decreto, en su Artículo 6, “Enseñanza bilingüe” se detalla que:

El currículo de este ciclo formativo incorpora la lengua inglesa de forma integrada en al menos dos módulos profesionales de entre los que componen la totalidad del ciclo formativo. Estos módulos se impartirán por el profesorado con atribución docente en los mismos y que, además, posea la habilitación lingüística correspondiente al nivel B2 del Marco Común Europeo de referencia para las lenguas.

Al objeto de garantizar que la enseñanza bilingüe se imparta en los dos cursos académicos del ciclo formativo de forma continuada se elegirán módulos profesionales de ambos cursos.

Los módulos susceptibles de ser impartidos en lengua inglesa son los señalados el anexo III.

Con carácter excepcional, y para quienes lo soliciten, en el caso de alumnos o alumnas con discapacidad que puedan presentar dificultades en su expresión oral (parálisis cerebral, sordera...), se establecerán medidas de flexibilización y/o alternativas en el requisito de impartición de módulos en lengua inglesa, de forma que puedan cursar todas las enseñanzas de los módulos profesionales en su lengua materna.

En el caso del ciclo de Desarrollo de Aplicaciones Multiplataforma en el I.E.S. San Vicente, se ha optado por impartir en inglés los módulos de Entornos de Desarrollo (para el grupo de enseñanza presencial vespertino), y Bases de Datos (para los grupos semipresenciales).

Esto supone que, para los alumnos del grupo presencial, los apuntes elaborados por el equipo docente estarán realizados en inglés para dichos módulos, así como los exámenes y gran parte de los ejercicios de clase.

No se espera que el alumno pueda llegar a dominar la lengua inglesa con sólo 3 horas a la semana, pero sí que coja la costumbre de manejar textos técnicos en inglés, y que sea capaz de entender un enunciado de un problema o un texto explicativo básico.

En cualquier caso, este aspecto no afecta durante el presente curso 2018-2019 al módulo de Programación.