CL2001 – *Fundamentos de programación (GM).*

Ciclos formativos para los que se oferta:

- CFGM Sistemas microinformáticos y redes.

Duración y curso: 54 horas, 2º curso.

Objeto:

Conocer los principios básicos de la programación y su aplicación en el desarrollo de software y soluciones informáticas.

Resultados de aprendizaje y criterios de evaluación:

1. Comprende los conceptos fundamentales de la programación, incluyendo variables, estructuras de control, tipos y estructuras de datos, uso de funciones y uso de librerías aplicándolos en la resolución de problemas sencillos.

   Criterios de evaluación:

   a) Se han identificado y descrito los conceptos fundamentales de la programación, como variables, estructuras de control, tipos y estructuras de datos, funciones y librerías
   b) Se han declarado y utilizado variables y constantes aplicando los tipos de datos adecuados según la problemática planteada.
   c) Se han diseñado y aplicado estructuras de control condicionales y bucles en la implementación de programas básicos.
   d) Se han manipulado estructuras de datos como listas, colas, arrays o diccionarios para almacenar y gestionar información.
   e) Se han implementado funciones para estructurar programas y mejorar su legibilidad y reutilización.
   f) Se han incorporado y empleado librerías básicas en el desarrollo de programas para la resolución de problemas concretos.

2. Desarrolla algoritmos básicos para la solución de problemas, utilizando diagramas de flujo y pseudocódigo como herramientas de diseño.

   Criterios de evaluación:

   a) Se han identificado los elementos y las estructuras principales de un algoritmo básico.
   b) Se han elaborado diagramas de flujo que representen soluciones simples a problemas dados.
   c) Se han descrito y utilizado las estructuras básicas de pseudocódigo, incluyendo operaciones secuenciales, condicionales y repetitivas.
   d) Se han relacionado diagramas de flujo con su correspondiente pseudocódigo.
   e) Se han analizado y corregido errores en algoritmos representados mediante diagramas de flujo o pseudocódigo.
   f) Se han resuelto problemas básicos diseñando algoritmos que combinen distintas estructuras de control.
   g) Se ha explicado el funcionamiento de un algoritmo básico a partir de su diagrama de flujo o pseudocódigo.

3. Implementa programas sencillos en un lenguaje de programación específico, demostrando habilidades en la escritura, depuración y ejecución de código.

   Criterios de evaluación:

   a) Se ha seleccionado y configurado un entorno de desarrollo adecuado para la implementación de programas sencillos.
   b) Se ha escrito código que cumpla con los estándares de sintaxis del lenguaje de programación utilizado.
   c) Se han implementado programas que resuelvan problemas específicos utilizando estructuras básicas del lenguaje, como variables, funciones y estructuras de control.
   d) Se han utilizado comentarios en el código para explicar su funcionalidad y mejorar su comprensión.
   e) Se han depurado errores en el código, identificando y corrigiendo problemas de sintaxis o lógica.
   f) Se han probado los programas implementados, verificando su correcto funcionamiento y resolviendo posibles incidencias.
   g) Se ha optimizado código para mejorar su eficiencia o legibilidad, aplicando buenas prácticas de programación.
   h) Se han desarrollado programas funcionales acompañados de una breve documentación que describa su propósito, funcionamiento y limitaciones.

4. Analiza la importancia de la programación en el desarrollo de software y soluciones informáticas, identificando y determinando su impacto en diferentes sectores.

   Criterios de evaluación:

   a) Se han descrito características y funciones principales de la programación en el desarrollo de software.
   b) Se han identificado diferentes áreas de aplicación de la programación en sectores como la industria, la salud, la educación y los servicios.
   c) Se han analizado ejemplos de software y soluciones informáticas que han transformado actividades o procesos en sectores específicos.
   d) Se ha relacionado la programación con tendencias actuales como la inteligencia artificial, el Internet de las cosas y la ciberseguridad.

5. Evalúa y optimiza el rendimiento de los programas desarrollados, aplicando buenas prácticas de programación y técnicas de mejora continua.

   Criterios de evaluación:

   a) Se han identificado factores que afectan el rendimiento de un programa, como la eficiencia de los algoritmos o el uso de recursos del sistema.
   b) Se han detectado cuellos de botella y puntos de mejora en programas a través de herramientas de análisis de rendimiento.
   c) Se han aplicado técnicas de programación específicas para mejorar la claridad, la modularidad y el mantenimiento del código.
   d) Se han optimizado los algoritmos y estructuras de datos para mejorar la eficiencia de los programas desarrollados.
   e) Se han verificado y documentado las mejoras realizadas en términos de rendimiento y funcionalidad.

f) Se han aplicado técnicas de mejora continua, como la refactorización del código, para garantizar la evolución y sostenibilidad del programa.

Contenidos:

1. Bases fundamentales de la programación.
    a) Definición y propósito de la programación. Variables y constantes. Estructuras de control.
    b) Tipos y estructuras de datos. Tipos de datos compuestos. Operaciones básicas.
    c) Funciones en programación. Ámbito de las variables. Buenas prácticas en el diseño de funciones.
    d) Concepto y utilidad de librerías. Incorporación de librerías básicas. Instalación y uso de librerías externas.
    e) Resolución de problemas. Metodología. Estructuración del programa. Validación de soluciones.
2. Algoritmos: conceptos fundamentales, diseño y aplicación.
    a) Definición y características de los algoritmos. Elementos de un algoritmo. Clasificación de algoritmos.
    b) Herramientas para el diseño de algoritmos: diagramas de flujo y pseudocódigo. Reglas y diseño de diagramas de flujo. Estructuras básicas en pseudocódigo.
    c) Relación entre diagramas de flujo y pseudocódigo. Traducción de diagramas de flujo a pseudocódigo. Comparativa.
    d) Análisis y corrección de algoritmos. Identificación de errores comunes. Herramientas para la corrección de errores.
    e) Resolución de problemas mediante diseño de algoritmos. Metodología. Ejercicios prácticos de diseño de algoritmos.
    f) Explicación y documentación de algoritmos. Comunicación de algoritmos. Ejemplos prácticos.
3. Primeros programas: manejo de un entorno de desarrollo integrado (IDE) e introducción de código.
    a) Configuración y uso del entorno de desarrollo integrado (IDE). Selección, instalación y configuración del IDE. Familiarización con el entorno.
    b) Introducción de código en el lenguaje de programación seleccionado. Sintaxis del lenguaje. Uso de estructuras básicas. Mejora de la legibilidad del código.
    c) Depuración y resolución de errores. Identificación de errores comunes. Herramientas de depuración. Corrección de errores en programas.
    d) Prueba y validación de programas. Métodos de prueba básica. Diseño de casos de prueba. Resolución de incidencias detectadas.
    e) Documentación de programas. Elaboración de documentación básica. Inclusión de comentarios en el código.
    f) Ejercicios prácticos. Desarrollo de programas funcionales. Pruebas y corrección colaborativa.

4. Impacto de la programación en la sociedad actual.
    a) Introducción a la programación y su relevancia en el desarrollo de software. Funciones principales de la programación en el desarrollo de software. Beneficios generales de la programación.
    b) Aplicaciones de la programación en diferentes sectores: industria, salud, educación, servicios y comercio. Impacto de la programación en sectores específicos. Ejemplos de software transformador.
    c) Relación de la programación con tendencias tecnológicas actuales. Inteligencia artificial (IA). Internet de las cosas (IoT). Ciberseguridad.
    d) Ejercicios prácticos y análisis de casos. Análisis de software conocido. Creación de proyectos básicos orientados a sectores específicos.
5. Rendimiento y optimización de los programas desarrollados.
    a) Factores que afectan el rendimiento de un programa. Factores clave. Métodos de evaluación del rendimiento.
    b) Análisis del rendimiento y detección de cuellos de botella. Herramientas.
    c) Buenas prácticas de programación para optimizar el rendimiento. Mejora de la legibilidad y modularidad del código. Reducción de redundancias y optimización de procesos.
    d) Optimización de algoritmos y estructuras de datos. Técnicas de optimización. Selección y optimización de estructuras de datos.
    e) Refactorización y mejora continua del código. Concepto y principios de refactorización. Técnicas de refactorización. Garantía de sostenibilidad del programa.
    f) Documentación y verificación de las mejoras. Registro de mejoras realizadas. Técnicas de verificación del rendimiento.
    g) Ejercicios prácticos y análisis de casos. Ejercicios prácticos de optimización. Estudio de casos.

Especialidades del Profesorado:

- Cuerpo/s: 0511/0590 Catedráticos/Profesores de enseñanza secundaria - Especialidad: 107 - Informática.
- Cuerpo/s: 0590/0591 Profesores de enseñanza secundaria/Profesores técnicos de formación profesional (a extinguir) - Especialidad: 227 - Sistemas y aplicaciones informáticas.
- Para la impartición del módulo optativo «Fundamentos de programación (GM)» en centros de titularidad privada o de titularidad pública de otras administraciones distintas de las educativas, se exigirán las mismas condiciones de formación inicial que para impartir cualquiera de los módulos que incluyan estándares de competencia adscritos a la misma familia profesional que el correspondiente título.