26-27

GRADO EN MATEMÁTICAS
SEGUNDO CURSO

GUÍA DE ESTUDIO PÚBLICA

LENGUAJES DE PROGRAMACIÓN

CÓDIGO 6102210-

UNED

26-27

LENGUAJES DE PROGRAMACIÓN

CÓDIGO 6102210-

ÍNDICE

PRESENTACIÓN Y CONTEXTUALIZACIÓN
REQUISITOS Y/O RECOMENDACIONES PARA CURSAR LA ASIGNATURA
EQUIPO DOCENTE
HORARIO DE ATENCIÓN AL ESTUDIANTE
TUTORIZACIÓN EN CENTROS ASOCIADOS
COMPETENCIAS QUE ADQUIERE EL ESTUDIANTE
RESULTADOS DE APRENDIZAJE
CONTENIDOS
METODOLOGÍA
SISTEMA DE EVALUACIÓN
BIBLIOGRAFÍA BÁSICA
BIBLIOGRAFÍA COMPLEMENTARIA
RECURSOS DE APOYO Y WEBGRAFÍA
IGUALDAD DE GÉNERO

NOMBRE DE LA ASIGNATURA	LENGUAJES DE PROGRAMACIÓN
CÓDIGO	6102210-
CURSO ACADÉMICO	2026/2027
DEPARTAMENTO	INFORMÁTICA Y AUTOMÁTICA
TÍTULO EN QUE SE IMPARTE	GRADO EN MATEMÁTICAS
CURSO	SEGUNDO CURSO
PERIODO	SEMESTRE 2
Nº ETCS	6
HORAS	150.0
IDIOMAS EN QUE SE IMPARTE	CASTELLANO

PRESENTACIÓN Y CONTEXTUALIZACIÓN

La asignatura Lenguajes de Programación se imparte en el segundo semestre, del segundo curso, del Grado en Matemáticas. Se trata de una asignatura de carácter básico, de 6 créditos ECTS, perteneciente a la materia del Grado denominada *Informática*.
Por su temática, la asignatura Lenguajes de Programación guarda relación con la otra asignatura de la materia Informática. Esto es, con la asignatura *Herramientas informáticas para Matemáticas*.
Algunos de los ejemplos y ejercicios prácticos propuestos en la asignatura Lenguajes de Programación consisten en la programación en C++ de algoritmos para la simulación por ordenador de modelos matemáticos de sistemas físicos. En algunos casos, lo aprendido en la asignatura de primer curso *Física* será de ayuda para entender el significado de dichos modelos matemáticos.
Asimismo, la asignatura Lenguajes de Programación proporcionará al alumno los conocimientos de programación necesarios para programar en C++ los métodos numéricos que se le explicarán en las dos asignaturas de la materia *Métodos Numéricos*.

REQUISITOS Y/O RECOMENDACIONES PARA CURSAR LA ASIGNATURA

Se recomienda al alumno que previamente curse la asignatura

•Herramientas informáticas para Matemáticas

Los conceptos sobre manejo de lenguajes de alto nivel proporcionados en esa asignatura, así como la experiencia adquirida en el manejo del ordenador, facilitarán la adquisición de los conocimientos impartidos en la asignatura Lenguajes de programación.

EQUIPO DOCENTE

Nombre y Apellidos	CARLA MARTIN VILLALBA
Correo Electrónico	carla@dia.uned.es
Teléfono	91398-8253
Facultad	ESCUELA TÉCN.SUP INGENIERÍA INFORMÁTICA
Departamento	INFORMÁTICA Y AUTOMÁTICA

Nombre y Apellidos	MIGUEL ANGEL RUBIO GONZALEZ
Correo Electrónico	marubio@dia.uned.es
Teléfono	91398-7154
Facultad	ESCUELA TÉCN.SUP INGENIERÍA INFORMÁTICA
Departamento	INFORMÁTICA Y AUTOMÁTICA

Nombre y Apellidos	ALFONSO URQUIA MORALEDA (Coordinador/a de asignatura)
Correo Electrónico	aurquia@dia.uned.es
Teléfono	91398-8459
Facultad	ESCUELA TÉCN.SUP INGENIERÍA INFORMÁTICA
Departamento	INFORMÁTICA Y AUTOMÁTICA

HORARIO DE ATENCIÓN AL ESTUDIANTE

Las consultas deben dirigirse al Equipo Docente por cualquiera de los tres métodos siguientes:

- La comunicación escrita se realizará preferiblemente a través de los foros del curso virtual de la asignatura. También puede contactarse con el Equipo Docente escribiendo a la dirección de correo electrónico de la asignatura (lp@dia.uned.es), o mediante correo postal, que debe dirigirse a la dirección: “Alfonso Urquía, Dpto. de Informática y Automática, E.T.S. de Ingeniería Informática, UNED, Juan del Rosal 16, 28040, Madrid”.
- Llamando a los números de teléfono 91 398 84 59 / 82 53 / 71 54 cualquier martes lectivo, entre las 10h y las 14h.
- Acudiendo personalmente a la E.T.S. de Ingeniería Informática de la UNED. En este caso, el alumno debe previamente concertar una cita con el Equipo Docente, mediante comunicación telefónica o escribiendo un correo electrónico.

TUTORIZACIÓN EN CENTROS ASOCIADOS

Las sesiones de tutoría se imparten presencialmente y en línea desde el Centro Asociado de referencia o, en su defecto, desde otro Centro Asociado. A continuación se muestran los centros asociados y extensiones en las que se imparten tutorías de la asignatura.

COMPETENCIAS QUE ADQUIERE EL ESTUDIANTE

La metodología, los materiales didácticos y el contenido de esta asignatura contribuyen al desarrollo de competencias genéricas propuestas por la UNED y de competencias específicas del Grado en Matemáticas. Entre las competencias genéricas, se encuentran las siguientes:

- El material docente de la asignatura está especialmente concebido para su uso dentro del modelo educativo a distancia de la UNED. Esto facilita que el alumno pueda estudiar de manera autónoma, potenciando su iniciativa y motivación. El alumno, guiado por la planificación temporal propuesta por el Equipo Docente, desarrolla su capacidad para la gestión y planificación de su propio trabajo, y el manejo adecuado del tiempo.
- Los ejercicios resueltos de autocomprobación permiten al alumno desarrollar su capacidad para realizar el seguimiento y evaluación de su propio trabajo.
- Los trabajos prácticos evaluables permiten al alumno desarrollar su capacidad para la comunicación y expresión escrita en el ámbito científico y tecnológico.
- Las herramientas de comunicación, proporcionadas en el Curso Virtual de la asignatura, permiten al alumno desarrollar su capacidad para la comunicación adecuada y eficaz con otras personas, empleando medios tecnológicos.
- El estudio de la materia y la realización de las actividades propuestas en la asignatura contribuyen al desarrollo de capacidades cognitivas superiores del alumno, como son la capacidad de analizar y resolver problemas, de razonar de manera crítica y tomar decisiones en el contexto del diseño y programación de software, y de aplicar los conocimientos a la práctica.

Asimismo, el contenido de la asignatura contribuye a que el alumno desarrolle las siguientes competencias específicas del Grado en Matemáticas:

- *Conocimientos disciplinares*. CED1 Desarrolla la capacidad del alumno para la comprensión de los conceptos básicos y familiaridad con los elementos fundamentales para el estudio de las Matemáticas superiores. CED2 Destreza en el razonamiento cuantitativo, basado en los conocimientos adquiridos.
- *Competencias profesionales*. CEP4 Resolución de problemas.
- *Competencias académicas*. Destreza en el razonamiento y capacidad para utilizar sus distintos tipos, fundamentalmente por deducción, inducción y analogía. CEA2 Capacidad para tratar problemas matemáticos desde diferentes planteamientos y su formulación correcta en lenguaje matemático, de manera que faciliten su análisis y resolución. CEA3 Habilidad para crear y desarrollar argumentos lógicos, con clara identificación de las hipótesis y las conclusiones. CEA4 Habilidad para detectar inconsistencias de razonamiento ya sea de forma teórica o práctica mediante la búsqueda de contraejemplos. CEA6 Habilidad para extraer información cualitativa a partir de información cuantitativa. CEA7 Habilidad para presentar el razonamiento matemático y sus conclusiones de manera clara y

precisa, de forma apropiada a la audiencia a la que se dirige, tanto en la forma oral como escrita.

•*Otras competencias específicas*. CE1 Razonamiento crítico, capacidad de evaluar trabajos propios y ajenos.

RESULTADOS DE APRENDIZAJE

Como resultado del aprendizaje, se pretende que el alumno adquiera fundamentalmente las capacidades enumeradas a continuación.

•Entender las reglas básicas de sintaxis de la programación.

•Entender y poner ejemplos de variables locales y globales. Saber leer y escribir ficheros externos. Saber manipular listas y cadenas de caracteres.

•Saber diseñar, programar, utilizar, explicar la utilidad y depurar funciones (incluyendo funciones recurrentes) y programas sencillos.

A continuación se detallan los resultados del aprendizaje que el alumno debe alcanzar tras estudiar cada uno de los temas.

TEMA 1: FUNDAMENTOS DE PROGRAMACIÓN

Resultados del aprendizaje referidos a principios básicos

•Discutir las ideas generales que han guiado la evolución de los lenguajes de programación.

•Discutir qué influencia ha tenido la arquitectura de la máquina de von Neumann en los primeros lenguajes de programación imperativos.

•Discutir cómo se produce la ejecución de un programa en la máquina de von Neumann.

•Discutir qué características tienen los lenguajes de bajo nivel (lenguaje máquina y ensamblador). Discutir las diferencias con los lenguajes de alto nivel y las ventajas de éstos frente a aquéllos.

•Discutir qué necesidades motivaron el desarrollo de los lenguajes de programación siguientes: FORTRAN, ALGOL, LISP, COBOL, BASIC, Visual BASIC, Prolog, SIMULA 67, Pascal, C, Modula-2, Ada, Smalltalk, C++, Java, sh, awk, Perl, JavaScript, PHP y Python. Discutir las principales características de estos lenguajes y qué nuevas capacidades aportó cada uno de ellos.

•Discutir en qué consiste la metodología de la programación estructurada.

•Discutir las características básicas de los cuatro paradigmas de programación fundamentales: programación imperativa, funcional, lógica y orientada a objetos.

•Discutir las características, y las ventajas y desventajas, de cada uno de los tres métodos siguientes de implementación de lenguajes de alto nivel: interpretación pura, compilación, y sistema híbrido de interpretación y compilación.

•Discutir en qué consiste el preprocesado del programa.

Resultados del aprendizaje referidos a programación en C++

- Discutir el significado de cada una de las líneas de código de un programa sencillo, como es el caso del programa ``Hola mundo!'', que escribe en la consola una frase.
- Discutir cómo se construyen literales de tipo string. Discutir el uso de caracteres especiales en la construcción de literales de tipo string.
- Discutir cómo se ponen datos en el flujo estándar de salida, empleando manipuladores.
- Discutir por qué los flujos de entrada y salida funcionan como buffers.
- Discutir la finalidad de los flujos de salida, entrada y error estándar.

TEMA 2: VARIABLES Y TIPOS DE DATOS

Resultados del aprendizaje referidos a principios básicos

- Discutir qué es una variable y qué atributos tiene. En particular, discutir el significado de los atributos nombre, dirección en memoria, valor y tipo de dato.
- Discutir qué es la declaración e inicialización de una variable.
- Discutir qué son las variables constantes. Discutir qué significa que el lenguaje permita establecer ligaduras estáticas y dinámicas sobre el valor de las constantes.
- Discutir qué es un bloque de código, y los conceptos de ámbito y visibilidad de una variable.
- Discutir la diferencia entre variables locales y globales.
- Discutir qué motivaciones tiene la declaración de los tipos de datos de las variables.
- Discutir qué son los tipos de datos primitivos. En particular, qué son los tipos número entero, número real, Booleano y carácter.
- Discutir las facilidades que los diferentes lenguajes proporcionan al programador para que éste defina sus propios tipos de datos.
- Discutir qué es un array, qué significa declarar e inicializar un array, cómo se accede en el programa a los componentes de un array y qué maneras tienen las implementaciones de almacenar los arrays bidimensionales en memoria.
- Discutir la manera en que diferentes lenguajes soportan las cadenas de caracteres, bien almacenándolas en arrays, cuyos componentes son de tipo carácter, o bien mediante objetos del tipo de dato string.
- Discutir qué son los punteros y cuáles son algunos de sus principales usos.
- Discutir qué son las variables en memoria dinámica, y cuál es su diferencia con las variables locales y globales.
- Discutir algunos de los errores típicos de programación que se cometen al trabajar con arrays, punteros y variables en memoria dinámica. Explicar las facilidades que proporcionan diferentes lenguajes para la prevención y diagnosis de este tipo de errores, que en ocasiones son cometidos por el programador.

Resultados del aprendizaje referidos a programación en C++

- Discutir cómo y en qué partes del programa se realiza la declaración e inicialización de variables en C++.

•Discutir qué restricciones impone C++ a los nombres de los identificadores.
•Discutir cómo se realiza la declaración de variables constantes y cómo se les asigna valor.
•Discutir qué tipos de datos básicos (también llamados primitivos) proporciona C++.
•Realizar programas sencillos, en los cuales se declaren e inicialicen variables de los tipos de datos básicos, y se vuelque el valor de dichas variables a la consola.
•Discutir qué son los límites numéricos de los tipos de datos básicos y realizar programas para obtenerlos.
•Discutir cómo la implementación de C++ realiza la inicialización por defecto de las variables locales y globales de los tipos de datos básicos.
•Discutir cómo se declaran tipos enumerados y declarar variables de este tipo en programas.
•Discutir qué son las estructuras, y realizar la declaración e inicialización de estructuras en programas.
•Declarar e inicializar arrays en programas. Acceder a los componentes de un array.
•Declarar e inicializar variables y constantes de los tipos de datos std::string y std::vector, que están declarados en la librería estándar de C++.
•Discutir las diferencias entre arrays y vectores.
•Discutir de qué tipo son los flujos predefinidos de entrada y salida.
•Discutir cómo se declaran los punteros y cómo se emplean para referenciar variables, incluyendo las variables en memoria dinámica.
•Discutir cómo se declaran variables y arrays en memoria dinámica, usando el operador new, y cómo se libera el espacio ocupado por esas variables usando el operador delete. Reservar y liberar espacio para este tipo de variables en los programas.
•Discutir cómo se define en C++ el ámbito y la visibilidad de las variables. Discutir el acceso a variables globales mediante el operador ámbito. Aplicar estos conceptos en el desarrollo de programas.

TEMA 3: ASIGNACIONES Y EXPRESIONES

Resultados del aprendizaje referidos a principios básicos

•Discutir qué es una sentencia de asignación.
•Discutir qué es una expresión. Clasificar las expresiones en aritméticas, relacionales, Booleanas y condicionales.
•Discutir qué son los operadores relacionales y los Booleanos. Indicar cuáles son.
•Discutir la diferencia de significado entre el operador de asignación y el operador de comparación de igualdad.
•Clasificar los operadores, en función del número de sus operandos, en unarios, binarios y ternarios.
•Clasificar la notación de las expresiones, en función de la posición relativa de los operadores respecto a los operandos, en notación prefija, sufija e infija.

- Emplear en asignaciones los operadores que combinan una operación aritmética con la asignación (+=, -=, *=, /=, %=).
- Emplear en expresiones los operadores incremento ++ y decremento --, usados tanto como prefijo como postfijo.
- Discutir qué finalidad tienen las reglas de asociatividad y precedencia. Discutir las reglas de precedencia que comúnmente se aplican en los lenguajes de programación.
- Discutir qué es el sistema de tipos de un lenguaje de programación.
- Discutir qué es la sobrecarga de los operadores.
- Discutir la diferencia entre las conversiones de tipo explícitas e implícitas. Discutir las reglas que aplica el compilador de C para realizar las coerciones (conversiones implícitas).
- Discutir en qué consiste la verificación de tipos, y la diferencia entre verificación estática y dinámica.
- Discutir qué se entiende por lenguaje fuertemente tipado.

Resultados del aprendizaje referidos a programación en C++

- Discutir cómo son las sentencias de asignación en C++ y las conversiones de tipo que se producen automáticamente al ejecutar dichas sentencias.
- Construir expresiones empleando los operadores aritméticos de C++. Decidir, aplicando las reglas de precedencia, el orden de evaluación de los operadores en las expresiones aritméticas.
- Construir expresiones lógicas empleando los operadores aritméticos, relacionales y lógicos de C++. Decidir, aplicando las reglas de precedencia, el orden de evaluación de los operadores que intervienen en las expresiones lógicas. Discutir en qué consiste la evaluación en cortocircuito de los operadores lógicos.
- Discutir el significado de los operadores >>y <<cuando se aplican a palabras de bits y a flujos.
- Emplear las funciones matemáticas declaradas en la cabecera estándar cmath.
- Emplear punteros para el direccionamiento de variables. Emplear los operadores dirección-de e indirección. Discutir la relación existente entre punteros y arrays. Emplear punteros para direccionar los componentes de un array.
- Declarar variables del tipo std::complex, std::string y std::vector, inicializarlas y usar dichas variables en la construcción de expresiones, empleando las funciones miembro y los operadores definidos en la librería estándar para este tipo de datos.
- Discutir qué son los iteradores. Declarar iteradores a vectores y emplearlos para direccionar los componentes de un vector.
- Emplear variables de tipo estructura en expresiones y sentencias de asignación. Emplear punteros para referenciar las estructuras y sus miembros. Emplear arrays cuyos componentes sean de tipo estructura. Discutir qué es una estructura autorreferenciada y cuál es su utilidad.

•Realizar programas sencillos en los que se realicen operaciones de entrada por teclado y salida por consola, se declaren e inicialicen variables, se manipulen los datos almacenados en variables empleando expresiones y se guarden en variables los resultados, usando para ello sentencias de asignación.

TEMA 4: CONTROL DEL FLUJO DEL PROGRAMA

Resultados del aprendizaje referidos a principios básicos

•Discutir qué finalidad tienen las sentencias de selección y las sentencias iterativas, y cuál es la diferencia entre ambas.

•Discutir cómo se controla el flujo del programa en las sentencias de selección if, case y switch. Discutir las diferencias entre ellas.

•Discutir en qué consiste el problema del else ambiguo y de qué maneras evitan los lenguajes que surja este problema.

•Discutir el significado y qué usos tienen las sentencias break y continue.

•Discutir cómo se controla el flujo del programa en las diferentes formas de la sentencia iterativa for, y en las sentencias controladas mediante expresión Booleana. Respecto a este último tipo de sentencias, discutir la diferencia entre las sentencias con precondición (por ejemplo, sentencia while) y con postcondición (por ejemplo, sentencia do-while).

•Discutir qué son las excepciones y en qué consiste, a grandes rasgos, la captura y tratamiento de las excepciones.

Resultados del aprendizaje referidos a programación en C++

•Emplear las sentencias de selección (if y switch) e iterativas (for, while y do-while), así como las sentencias break y continue, en la escritura de programas en C++.

•Realizar programas en los que se capturen y traten excepciones dentro del código de la función main. Las excepciones capturadas podrán ser tanto las lanzadas por el programador, como las lanzadas por las funciones y operadores de la librería estándar de C++.

•Declarar variables en memoria dinámica, capturando y tratando la excepción std::bad_alloc.

•Programar la entrada de datos a través del flujo std::cin, empleando un bucle while, analizando el tipo de error producido y descartando el contenido del flujo de entrada cuando proceda. Discutir cómo cambia el estado del flujo de entrada, dependiendo de si la operación sobre el flujo se ha realizado o no con éxito. Emplear en los programas las funciones que proporcionan información acerca del estado del flujo de entrada y que cambian dicho estado.

•Escribir programas en C++ en los cuales se realice entrada y salida a fichero de texto.

TEMA 5: SUBPROGRAMAS

Resultados del aprendizaje referidos a principios básicos

•Discutir qué es un subprograma y qué ventajas proporciona el uso de subprogramas.

•Discutir de qué partes consta la definición de una función.
•Discutir cómo se realiza la invocación de una función. Definir qué son los parámetros formales de la función y qué son los parámetros actuales usados en la invocación.
•Discutir cómo se realiza la evaluación de una función. Discutir en qué consiste el paso de parámetros por valor y por referencia, y la diferencia entre ambas formas de invocación de funciones.
•Discutir cómo se realiza el paso de parámetros a funciones en los lenguajes C y C++.
•Reconocer el ámbito y la visibilidad de las variables locales y de los parámetros declarados en funciones.
•Reconocer si una función es recursiva, si tiene recursividad lineal y si tiene recursividad de cola.
•Discutir la diferencia entre una función y un procedimiento.
•Discutir de qué partes consta la definición de un procedimiento y cómo se realiza su invocación.

Resultados del aprendizaje referidos a programación en C++

•Discutir cómo se definen e invocan funciones en lenguaje C++. Discutir cómo se realiza el paso de parámetros a las funciones en lenguaje C++.
•Realizar programas en los cuales se definan e invoquen funciones.
•Dada una función, reconocer el ámbito y la visibilidad de los parámetros formales y las variables locales.
•Discutir cómo son tratadas, a efectos de su almacenamiento en memoria, las variables estáticas declaradas en el cuerpo de una función. Discutir cuál es el ámbito de dichas variables. Emplear este tipo de variables en la programación de funciones.
•Discutir el significado de la sentencia return y emplear dicha sentencia en la programación de funciones.
•Realizar programas en los cuales se definan funciones que lancen excepciones y en los cuales estas excepciones sean capturadas y tratadas.
•Discutir qué es la declaración y la definición de una función, y cuál es la diferencia entre ambas. Realizar programas en los cuales las funciones sean declaradas y definidas.
•Escribir programas organizados en varios ficheros.
•Discutir cuál es el propósito de los espacios de nombres, y realizar programas en los cuales se definan espacios de nombres y se usen las entidades declaradas en ellos.

TEMA 6: ESTRUCTURAS DE DATOS

Resultados del aprendizaje referidos a principios básicos

• Discutir qué son los tipos abstractos de datos y qué utilidad tienen.
• Para las estructuras de datos lista, pila, cola, mapa y árbol, discutir:
•Qué características tiene cada estructura. Los conceptos y la terminología básicos relacionados con cada estructura de datos.

•Qué utilidad puede tener cada una de estas estructuras de datos en la realización de programas.
•Qué operaciones se realizan comúnmente sobre cada una de estas estructuras de datos.
•Cómo puede implementarse cada una de estas estructuras de datos mediante un array y mediante estructuras autorreferenciadas. Pros y contras en cada caso de estas dos opciones de implementación.
• Programar en C++ listas, incluyendo pilas y colas, mediante estructuras autorreferenciadas, así como programar funciones que realicen operaciones básicas sobre listas.

Resultados del aprendizaje referidos a programación en C++

•Planteado un problema de programación, discutir qué estructuras de datos sería más adecuado usar de entre las siguientes: listas, pilas, colas y mapas.
•Discutir conceptos básicos de los componentes fundamentales de la Standard Template Library (STL) de C++. Esto es, de los contenedores, algoritmos e iteradores.
•Emplear en la realización de programas en C++ los tipos lista, cola, pila y mapa declarados en la STL, sabiendo declarar e inicializar variables de esos tipos, insertar, eliminar, modificar y buscar elementos, y manipular el contenido de las variables de dichas estructuras de datos empleando funciones miembro.
•Emplear iteradores para acceder a los elementos de listas, colas, pilas y mapas.

TEMA 7: ALGORITMOS

Resultados del aprendizaje referidos a principios básicos

•Discutir qué es un algoritmo.
•Discutir las características básicas de los siguientes paradigmas para el diseño de algoritmos: fuerza bruta o búsqueda exhaustiva, divide y vencerás, programación dinámica, programación lineal, programación entera, y búsqueda y enumeración.
•Discutir las principales características de las dos formas siguientes de describir los algoritmos: mediante pseudocódigo y mediante diagrama de flujo.
•Discutir cómo se estima la complejidad de un algoritmo y qué es la notación O.
•Discutir cuál es la finalidad de los algoritmos de ordenación.
•Programar en C++ los tres algoritmos de ordenación siguientes: método de la burbuja, ordenación por inserción y ordenación por mezcla.

Resultados del aprendizaje referidos a programación en C++

•Realizar programas en C++ empleando los algoritmos siguientes, que se encuentran declarados en la STL de C++: count, count_if, remove_copy, replace_copy, reverse, transform, sort y find_if.
•Emplear expresiones lambda para definir funciones anónimas y pasarlas como argumento a algoritmos.

CONTENIDOS

TEMA 1. FUNDAMENTOS DE PROGRAMACIÓN

TEMA 2. VARIABLES Y TIPOS DE DATOS

TEMA 3. ASIGNACIONES Y EXPRESIONES

TEMA 4. CONTROL DEL FLUJO DEL PROGRAMA

TEMA 5. SUBPROGRAMAS

TEMA 6. ESTRUCTURAS DE DATOS

TEMA 7. ALGORITMOS

METODOLOGÍA

El **texto base** de la asignatura es una Unidad Didáctica editada por la UNED. Este texto está adaptado para la educación a distancia y cubre totalmente el temario de la asignatura. En la **página web de la asignatura** (https://www.uned.es/6102210/) están disponibles los objetivos docentes de cada tema y el temario detallado, de modo que aquellos alumnos que lo deseen puedan preparar la asignatura empleando otros recursos diferentes al texto base.
Se recomienda al alumno que aprenda a manejar algún **entorno integrado de desarrollo (IDE) de C++** y que realice por sí mismo la programación y ejecución del código explicado en el texto base, así como que emplee dicho entorno de desarrollo para resolver los ejercicios y las actividades propuestas. En la página web de la asignatura puede encontrarse información acerca de varios entornos integrados de desarrollo gratuitos para C++.
En la página web de la asignatura hay **ejercicios de autoevaluación** y la solución a los mismos, así como **exámenes y trabajos de anteriores convocatorias resueltos**. También pueden encontrarse enlaces a **recursos de uso opcional**, que pueden ser útiles para aquellos alumnos que voluntariamente deseen profundizar en la materia más allá de los objetivos planteados en la asignatura.

SISTEMA DE EVALUACIÓN

TIPO DE PRUEBA PRESENCIAL

Tipo de examen	Examen de desarrollo
Preguntas desarrollo	5
Duración del examen	120 (minutos)

Material permitido en el examen

Ninguno

Criterios de evaluación

En el enunciado del examen se indica la puntuación de cada pregunta.

% del examen sobre la nota final	50
Nota del examen para aprobar sin PEC	0
Nota máxima que aporta el examen a la calificación final sin PEC	0
Nota mínima en el examen para sumar la PEC	5

Comentarios y observaciones

El examen presencial escrito obligatorio se celebrará en todos los Centros Asociados, de manera coordinada, según el calendario previsto. El examen tendrá una duración de 2 horas, no se permitirá el uso de ningún material y constará de varios ejercicios, que el alumno deberá resolver de manera argumentada.

El examen será calificado con una nota comprendida entre 0 y 10. Para aprobar el examen debe obtenerse una nota igual o superior a 5. Para aprobar la asignatura es necesario aprobar el examen.

PRUEBAS DE EVALUACIÓN CONTINUA (PEC)

¿Hay PEC?

Descripción

El trabajo práctico obligatorio consistirá en una serie de ejercicios de programación que el alumno deberá realizar individualmente. Se propondrá un trabajo para la convocatoria ordinaria y otro trabajo diferente para la convocatoria extraordinaria. El enunciado del trabajo, incluyendo información acerca de los plazos y la forma de entrega, se publicará en el curso virtual de la asignatura. La entrega y evaluación del trabajo se realiza también a través del curso virtual. El trabajo práctico será calificado con una nota comprendida entre 0 y 10. Para aprobar el trabajo práctico debe obtenerse una nota igual o superior a 5.

Criterios de evaluación

La puntuación de cada ejercicio se especifica en el enunciado del trabajo.

Ponderación de la PEC en la nota final	50%
Fecha aproximada de entrega	16 de abril (conv. ordinaria) y 10 de septiembre (conv. extraordinaria)

Comentarios y observaciones

El trabajo práctico es obligatorio. Para aprobar la asignatura es necesario aprobar el trabajo práctico.

OTRAS ACTIVIDADES EVALUABLES

¿Hay otra/s actividad/es evaluable/s?

Descripción

La participación en los foros del curso virtual no es obligatoria. No obstante, se valorará positivamente la participación constructiva en los foros del curso virtual, entendiendo como tal la contribución a la resolución de dudas planteadas por otros alumnos, proporcionar información o comentarios útiles para que otros alumnos comprendan la materia, etc.

Criterios de evaluación

La participación constructiva en los foros se valorará con una nota comprendida entre 0 y 1, siempre y cuando se haya aprobado el examen y el trabajo práctico obligatorio.

Ponderación en la nota final 0

Fecha aproximada de entrega

Comentarios y observaciones

¿CÓMO SE OBTIENE LA NOTA FINAL?

Para superar la asignatura debe aprobar tanto el examen presencial como el trabajo práctico obligatorio.

La nota de aquellos alumnos que hayan aprobado el examen y el trabajo práctico se calculará, como se indica a continuación, de las notas obtenidas en el examen y en el trabajo, y de la evaluación de la actividad del alumno en los foros:

Nota = mín(10, 0.5*notaExamen + 0.5*notaTrabajo + actividadForos)

La nota del examen o del trabajo obtenida en la convocatoria ordinaria se guardará para la convocatoria extraordinaria del mismo curso académico. Sin embargo, no se guardarán notas de un curso académico al siguiente.

BIBLIOGRAFÍA BÁSICA

ISBN(13):9788436276916
Título:LENGUAJES DE PROGRAMACIÓN2ª, 2021
Autor/es:Carla Martín Villalba ; Alfonso Urquía Moraleda ; Miguel Ángel Rubio González ;
Editorial:Editorial UNED

El texto base de la asignatura es:

Título: Lenguajes de programación

Autores: Carla Martín Villalba, Alfonso Urquía Moraleda y Miguel Ángel Rubio González

Editorial UNED (SEGUNDA EDICIÓN, 2021)

ISBN 978-84-362-7691-6

Este texto cubre totalmente el temario y es suficiente para preparar la asignatura.
El contenido de la asignatura consta de 7 temas. El texto base de la asignatura tiene 14 capítulos, ya que cada punto del temario es desarrollado en dos capítulos consecutivos del texto base.
El texto base está organizado de manera que van alternándose los capítulos en los cuales se explican conceptos generales de los lenguajes de programación, con aquellos en los cuales se muestra la aplicación de estos conceptos en el lenguaje C++.
Se pretende con ello que el alumno adquiera unos sólidos conocimientos de los fundamentos de los lenguajes de programación en general y que, a la vez, adquiera la destreza suficiente en el manejo de un lenguaje de programación en particular (en este caso, C++) como para poder diseñar, programar y ejecutar aplicaciones sencillas en el ámbito de la computación con aplicación a la Matemática.
Los capítulos del texto base están estructurados de la forma siguiente:

1. Al comienzo de cada capítulo se enumeran los objetivos docentes que el alumno debe alcanzar una vez haya estudiado el tema y realizado por sí mismo los ejercicios de autocomprobación.
2. En los capítulos dedicados a conceptos generales, se muestra cómo diferentes lenguajes de programación soportan dichos conceptos. Se presta especial atención a los lenguajes FORTRAN, Pascal, Modula-2, Ada, C, C++, Java y Python. En los capítulos dedicados a la práctica en C++, se aplican dichos conceptos a la resolución de *casos prácticos* en el ámbito de la Matemática.
3. Al final de cada capítulo se encuentra una colección de ejercicios de autocomprobación, así como la solución a los mismos.

El alumno puede emplear el entorno de desarrollo de C++ que desee para escribir y ejecutar los programas. En la página web de la asignatura se dan indicaciones adicionales a este respecto.

BIBLIOGRAFÍA COMPLEMENTARIA

ISBN(13):9780072226805
Título:C++: THE COMPLETE REFERENCE2003
Autor/es:Herbert Schildt ;
Editorial:McGraw-Hill

ISBN(13):9780136073475
Título:CONCEPTS OF PROGRAMMING LANGUAGES2009
Autor/es:Robert W. Sebesta ;
Editorial:ADDISON WESLEY

ISBN(13):9780201700732

Título:THE C++ PROGRAMMING LANGUAGE2007
Autor/es:Bjarne Stroustrup ;
Editorial:PEARSON EDUCACIÓN

ISBN(13):9780321751041
Título:THE ART OF COMPUTER PROGRAMMING3rd Edition
Autor/es:Donald E. Knuth ;
Editorial:ADDISON-WESLEY

En el texto (Sebesta, 2009) puede encontrarse información adicional acerca de la evolución de los lenguajes de programación, los distintos paradigmas de programación, y conceptos relacionados con las variables y tipos de datos, las asignaciones y expresiones, el control del flujo del programa y los subprogramas. Los cuatro volúmenes que componen (Knuth, 2011) son una excelente referencia sobre algoritmos.
Los textos (Schildt, 2003) y (Stroustrup, 2007) son excelentes referencias para aprender C++. Asimismo, dado que C++ es un lenguaje de programación muy ampliamente usado, en Internet puede encontrarse abundante documentación, ejemplos de uso, entornos integrados de desarrollo (IDE) gratuitos, etc. En la página web de la asignatura (https://www.uned.es/6102210) hay algunos enlaces de interés.

RECURSOS DE APOYO Y WEBGRAFÍA

En el **curso virtual** de la asignatura puede encontrarse:
•La guía del curso.
•Los foros, que proporcionan un medio de comunicación entre los alumnos, y entre los alumnos y el profesorado.
•El enunciado de los trabajos obligatorios: el de convocatoria ordinaria y el de convocatoria extraordinaria. La entrega y evaluación del trabajo se realiza también a través del curso virtual.
•Noticias, como puede ser el anuncio de las fechas de las videoconferencias de los tutores.

En la **página web** de la asignatura (https://www.uned.es/6102210) puede encontrarse:
•Información más detallada acerca del contenido y los objetivos docentes de la asignatura.
•Ejercicios de autoevaluación resueltos.
•Soluciones a los exámenes de las convocatorias anteriores.
•Soluciones a los trabajos prácticos planteados en convocatorias anteriores.
•Enlaces a sitios de descarga de software gratuito de simulación, enlaces a cursos y otros recursos relacionados con C++, etc.

IGUALDAD DE GÉNERO

En coherencia con el valor asumido de la igualdad de género, todas las denominaciones que en esta Guía hacen referencia a órganos de gobierno unipersonales, de representación, o miembros de la comunidad universitaria y se efectúan en género masculino, cuando no se hayan sustituido por términos genéricos, se entenderán hechas indistintamente en género femenino o masculino, según el sexo del titular que los desempeñe.