Universitat Autònoma de Barcelona

2020/2021

Laboratorio de Programación

Código: 102767
Créditos ECTS: 6

Titulación	Tipo	Curso	Semestre
2502441 Ingeniería Informática	OB	2	1

La metodología docente y la evaluación propuestas en la guía pueden experimentar alguna modificación en función de las restricciones a la presencialidad que impongan las autoridades sanitarias.

Contacto

Nombre: Gemma Sánchez Albaladejo
Correo electrónico: Gemma.Sanchez@uab.cat

Uso de idiomas

Lengua vehicular mayoritaria: catalán (cat)
Algún grupo íntegramente en inglés: No
Algún grupo íntegramente en catalán: Sí
Algún grupo íntegramente en español: Sí

Equipo docente

Jorge Bernal del Nozal
Yael Tudela Barroso
Cristian Da Silva Campello
Estrella María Abril Gutiérrez

Prerequisitos

La asignatura no puede tener ningún prerrequisito oficial por normativa. Pero los estudiantes que no han cursado y aprobado previamente las asignaturas de Fonaments d'informàtica y Metodologia de la Programació, suspenden en mayor proporción esta asignatura. Por ello se recomienda insistentemente que el estudiante haya cursado y superado satisfactoriamente las
asignaturas previas de Fonaments d'Informàtica y Metodologia de la Programació así como Matemàtica Discreta. Por lo tanto, está familiarizado con las estructuras básicas y avanzadas de la programación, Orientación objetos y el concepto de grafo con los diferentes métodos de recorrido sobre ellos.

Objetivos y contextualización

Esta asignatura forma parte de la materia Algorítmica e información y debe verse como la continuación lógica de la asignatura Metodologia de la Programació y la continuación práctica de la asignatura de Matemàtica Discreta. El objetivo básico es profundizar en las nociones de programación orientada a objetos introducidas en Metodologia de la Programació y ampliar con otros conceptos de programación y otras estructuras de datos más complejas, así como algoritmos eficientes para recorrerlas. Se introducirá el concepto de algoritmo recursivo con algoritmos recursivos simples y más complejos como los relacionados con recorridos de árboles y grafos. Además, se introducirán algoritmos de búsqueda y ordenación eficientes y se profundizará en el concepto de coste temporal y espacial de un algoritmo. Al finalizar el curso el estudiante debe ser capaz de diseñar y programar soluciones a problemas complejos de forma óptima.

De esta forma, los objetivos formativos que se proponen para la asignatura son los siguientes:

- Ser capaz de analizar un problema complejo, diseñar una solución óptima, implementarla, calcular su coste y probarla.
- Entender y saber utilizar estructuras de datos complejas como árboles, grafos etc. y utilizarlas correctamente y de una manera eficiente para resolver problemas algorítmicos complejas.
- Entender y aplicar correctamente los principios avanzados de la programación orientada a objetos: templates, clases abstractas, funciones virtuales.
- Dotar al alumno de la capacidad de diseño de algoritmos para la resolución de problemas complejos, viendo algoritmos complejos de recorrido y búsqueda en estructuras de datos complejas. Además de analizar la complejidad temporal y espacialde ellos a fin de elegir la solución que más se adapte a las necesidades de cada momento.
- Introducir el concepto de recursividad y su aplicación al recorrido de estructuras complejas recursivas, además de ser capaz de analizar la complejidad de algoritmos recursivos.
- Programar en un lenguaje de programación real y ser capaz de depurar los propios programas.
- Desarrollar los programas siguiendo unas normas de estilo tendentes a conseguir programas de calidad. Dentro de estas normas de estilo engloban aquellas que facilitan la comprensión del código, como pueden ser el uso de comentarios, el tabulado del código, la utilización de nombres adecuados para variables y funciones, etc. Y el uso de excepciones.

Competencias

- Adquirir hábitos de pensamiento.
- Capacidad para concebir, desarrollar y mantener sistemas, servicios y aplicaciones informáticas empleando los métodos de la ingeniería del software como instrumento para el aseguramiento de su calidad.
- Conocimiento y aplicación de los procedimientos algorítmicos básicos de las tecnologías informáticas para diseñar soluciones a problemas, analizando la idoneidad y complejidad de los algoritmos propuestos.
- Conocimiento, diseño y utilización de forma eficiente los tipos y estructuras de datos más adecuados a la resolución de un problema.

Resultados de aprendizaje

1. Aplicar estrategias de depuración, prueba y corrección de programas.
2. Conocer y comprender los paradigmas de programación.
3. Desarrollar la capacidad de análisis, síntesis y prospectiva.
4. Desarrollar programas con un buen estilo de programación y con la documentación apropiada.
5. Identificar la complejidad computacional de un algoritmo en términos de recursos de memoria y tiempo de ejecución.
6. Identificar posibles estrategias de solución a problemas con los conceptos propios del paradigma de programación orientación a objetos.
7. Seleccionar y aplicar la combinación de estructuras de datos y estrategia de resolución más apropiada para resolver de modo eficiente un problema informático.

Contenido

0. Introducción
Objetivos y presentación de la asignatura. Repaso de Programación Orientada a Objetos y Estructuras de Datos Dinámicas.

1. Orientación a objetos
Paradigma de objetos avanzado. Templates. Herencia, clases abstractas, funciones virtuales y polimorfismo.

2. Estructuras de datos no lineales. hash

Técnicas de "Hashing". Matrices hash y listas hash. Funciones hash.

3. Recursividad y Algoritmos de ordenación

Introducción a los algoritmos recursivos. Método de la burbuja, QuickSort, mergesort. Recursividad. Cálculo complejidad.

4. Estructuras de datos no lineales. grafos
Representaciones y recorridos. BFS, DFS, Resolución de problemas con grafos.

5. Estructuras de datos no lineales. Árboles.

Definición y representación de un árbol. Recorridos de árboles. Binary Heaps. Red-Black Tree.

6. Conceptos básicos de Python.

Conceptos básicos de programación en Python.

Metodología

NOTA INFORMATIVA DEL CURSO DEBIDO AL Covid: Esta asignatura se hará en modo on-line debido a la par

La metodología docente de la asignatura parte del principio que dice que *"programar es la única forma de aprender a programar"* y, por tanto, estará centrada principalmente en el trabajo práctico del estudiante. También se basa en aprovechar al máximo el tiempo presencial que el estudiante está con el profesor. De este modo conceptos teóricos descriptivos, fácilmente alcanzados por el estudiante mediante la visualización de vídeos o lectura de artículos, se harán de manera autónoma (y guiada) por parte del estudiante. Mientras que la puesta en práctica de estos conceptos o la ampliación de los mismos se harán en clase con ayuda del profesor. El objetivo principal de la asignatura es que el estudiante sepa resolver un problema dado, de manera eficiente, utilizando estructuras de datos complejas, si es necesario. Por esta razón el aprendizaje se centrará en acompañar al estudiante en su tarea de resolución de problemas a partir de unos conceptos teóricos estudiados previamente de manera autónoma. Se utilizará principalmente el lenguaje de programación C ++, y se darán algunos conceptos de programación en python.

La metodología general de la asignatura se puede dividir en tres fases:

Preparación de la clase: el objetivo de estafase es que el alumnado pueda aprender los conceptos que se trabajarán en la sesión siguiente mediante diversas actividades ofrecidas por el profesorado como puede ser el visionado de vídeos, la lectura de textos, etc.

Clase presencial: el objetivo de esta fase es la de consolidar los conceptos vistos y ponerlos en valor dentro del contexto de la asignatura. El profesorado velará paraque el alumnado profundice en estos conceptos mediante ejercicios (más o menos) guiados durante la sesión, y añadiendo nuevos matices a los conceptos aprendidos de manera autónoma cuando sea necesario. De esta manera la clase presencial se aprovecha mucho más dado que los conceptos descriptivos ya son conocidos por el estudiante y se puede entrar de lleno en su utilización y ampliación que son los puntos en los que el estudiante puede necesitar más la ayuda del profesor.

Trabajo autónomo: para que el alumnado tome soltura en la utilización de estructuras complejas y los algoritmos asociados a las mismas éste tendrá que hacer una parte del trabajo por su cuenta, ya sean ejercicios sueltos o dentro de un proyecto.

El trabajo del proyecto deberá hacerse en grupos de 2 personas.

La gestión de la docencia de la asignatura se hará a través de Caronte (http://caronte.uab.cat/), que servirá para poder ver los materiales, gestionar los grupos de prácticas, hacer las entregas correspondientes, ver las notas , comunicarse con los profesores, etc.

competencias Transversales

- T01.02 Desarrollar la capacidad de análisis, síntesis y prospectiva. Esta competencia se desarrollará durante las sesiones presenciales, donde el estudiante, después de haber consolidado unos conceptos teóricos, deberá analizar un problema práctico y proponer una solución óptima. Se tendrá en cuenta en la evaluación de las diferentes partes de la asignatura.

Actividades

Título	Horas	ECTS	Resultados de aprendizaje
Tipo: Dirigidas			
Clases presenciales	50	2	1, 2, 3, 4, 5, 6, 7
Tipo: Supervisadas			
Tutorías	1	0,04	1, 2, 3, 4, 5, 6, 7
Tipo: Autónomas			
Estudio Individual	13	0,52	2, 3, 5, 6
Preparación previa de las clases	34	1,36	2, 3, 5, 6
Trabajo autónomo	46	1,84	1, 3, 4, 5, 6, 7

Evaluación

La evaluación de la asignatura tendrá en cuenta tres tipos de actividades de evaluación: entrega de problemas, evaluación individual y proyecto de programación. La nota final de la asignatura se obtiene combinando la evaluación de estas 3 actividades de la siguiente manera:

Nota Final = (0.2 * Evaluación Problemas) + (0.4 * Proyecto) + (0.4 * Evaluación Individual)

- Entrega de problemas: en este apartado se incluye la entrega de los ejercicios que se propongan a lo largo del curso y otras actividades que se realicen en las sesiones de problemas.

No hay una nota mínima en esta actividad para poder aprobar la asignatura.
Los ejercicios que se entreguen fuera de plazo o que tengan una evaluación de suspendido se podrán recuperar y volver a entregar en cualquier momento del curso antes de la fecha del examen final de la asignatura, con una reducción sobre la nota del 20%. Los problemas estarán ponderados según el peso del tema al conjunto de la asignatura, y el número de problemas que se tengan que entregar por cada tema.

- Evaluación individual: en este apartado se incluye el resultado de las pruebas individuales que se harán a lo largo del curso. Habrá dos pruebas parciales que se realizarán durante el período lectivo del curso en horarios de clase y una prueba final durante el periodo oficial de exámenes. Esta prueba final será de recuperación y sólo la tendrán que hacer los estudiantes que no hayan superado alguno de los dos parciales. Si se ha superado uno de los dos parciales, pero el otro no, en esta prueba sólo se ha de recuperar la parte de la asignatura correspondiente al parcial que no se haya superado.

Se deberá conseguir una nota mínima de 4 en cada uno de los dos parciales y una nota promedio mínima de 5 para poder aprobar la asignatura.

La nota final será la media de los dos parciales: Evaluación Individual = (0.5 * Parcial1) + (0.5 * Parcial2)

- Proyecto: incluye todo el trabajo del proyecto de programación. Incluye la evaluación de las dos entregas del proyecto (una entrega parcial a mitad de curso y la entrega final) y la evaluación del seguimiento del proyecto que se hará a las sesiones presenciales que corresponda. La nota final se calculará de la forma siguiente:

Proyecto = (0.2 * Evaluación seguimiento proyecto) + (0.3 * Entrega Parcial 1) + (0.5 * Entrega Final)

- Se deberá conseguir una nota mínima de 4 en la evaluación del seguimiento del proyecto y una nota mínima de 5 en el entrega final del proyecto para poder aprobar el proyecto.
- Se deberá conseguir una nota mínima de 5 en el proyecto para poder aprobar la asignatura.
- La nota de la entrega final del proyecto se podrá recuperar si la nota del proyecto es> = 3 y la nota de la evaluación individual es> = 5.

No evaluable: Un alumno se considerará no evaluable (NA) si no hace como mínimo el 50% de las entregas de ejercicios y no hace ninguna de las pruebas de evaluación: parcial 1, parcial 2, prueba final de recuperación, entrega final de la práctica.

suspendidos: Si el cálculo de la nota final es igual o superior a 5 pero no se llega al mínimo exigido en alguna de las actividades de evaluación, la nota final serà suspendido y se pondrá un 4.5 en la nota del expediente del alumno .

convalidaciones: Para los alumnos repetidores se convalidará la nota del proyecto del año anterior (curso 2018-19) si se cumplen estas condiciones:

a) La nota final del proyecto del curso anterior es mayor o igual a 7

b) La nota de la evaluación individual del curso anterior es mayor o igual a 3

MH: Se darán tantas matrículas como puedan dentro de la normativa de la universidad, empezando por las notas más altas y siempre y cuando la nota mínima sea un 9.

Revisiones: Para cada actividad de evaluación, se indicará un lugar, fecha y hora de revisión en la que el estudiante podrá revisar la actividad con el profesor. En este contexto, se podrán hacer reclamaciones sobre la nota de la actividad, que serán evaluadas por el profesorado responsable de la asignatura. Si el estudiante no sepresenta en esta revisión, no se revisará posteriormente esta actividad.

Nota importante sobre copias y plagios:
Sin perjuicio de otras medidas disciplinarias que se estimen oportunas, y de acuerdo con la normativa académica vigente, las irregularidades cometidas por un estudiante que puedan conducir a una variación de la calificación se calificarán con un cero (0). las actividades de evaluación calificadas de esta forma y por este procedimiento no serán recuperables. Si es necesario superar cualquiera de estas actividades de evaluación para aprobar la asignatura, esta asignatura quedará suspendida directamente, sin oportunidad de recuperarla en el mismo curso. Estas irregularidades incluyen, entre otras:

- la copia total o parcial de una práctica, informe, o cualquier otra actividad de evaluación;
- dejar copiar;
- presentar un trabajo de grupo no hecho íntegramente por los miembros del grupo;
- presentar como propios materiales elaborados por un tercero, aunque sean traducciones o adaptaciones, y en general trabajos con elementos no originales y exclusivos del estudiante;
- tener dispositivos de comunicación (como teléfonos móviles, smart watches, etc.) accesibles durante las pruebas de evaluación teórico Prácticas individuales (exámenes).

En estos casos, la nota numérica del expediente será el valor menor entre 3.0 y la media ponderada de las notas (y por tanto no será posible el aprobado por compensación).
En la evaluación de las entregas de problemas y prácticas se utilizarán herramientas de detección de copia del código del programa.

Nota sobre la planificación de las actividades de evaluación:
Las fechas de evaluación continua y entrega de trabajos se publicarán a principio de curso y pueden estar sujetas a cambios de programación por motivos de adaptación a posibles incidencias. Siempre se informará a Cerbero sobre estos cambios ya que se entiende que esta es la plataforma habitual de intercambio de información entre profesores y estudiantes.

Actividades de evaluación

Título	Peso	Horas	ECTS	Resultados de aprendizaje
Entrega problemas	20% Nota final	0	0	2, 3, 5, 6, 7
Examen de recuperación	ver la descripción del método de evaluación	2	0,08	1, 2, 3, 4, 5, 6, 7
Primer parcial	50% Nota ev. individual	2	0,08	1, 2, 3, 4, 5, 6, 7
Proyecto Programación	40% Nota final	0	0	1, 3, 4, 6, 7
Segundo parcial	50% Nota Ev. individual	2	0,08	1, 2, 3, 4, 5, 6, 7

Bibliografía

- http://www.cplusplus.com/ : The C++ Resources Network
- https://es.wikibooks.org/wiki/Programaci%C3%B3n_en_C%2B%2B: Programación en C++ - Wikilibros
- Mark Allen Weiss. Data Structures and Data Analysis in C++. Pearson. 2014.
- B. Eckel. Thinking in C++, Volume 1: Introduction to Standard C++, Prentice-Hall, 1999
- B. Eckel. Thinking in C++, Volume 2: Standard Libraries and Advanced Topics, Prentice-Hall, 1999
- F. Xhafa, P. Vázquez, J. Marco, X. Molinero, A. Martín: Programación en C++ para ingenieros. Thomson, 2006
- Thinking in PYTHON Bruce Eckel (se puede descargar de http://www.bruceeckel.com).
- Learning PYTHON 2nd Edition. Mark Lutz and David Ascher, Safari Tech Books Online.
- Manuals de Python (de la pagina web oficial).
- Llibres electronics interactius de python:
    - http://interactivepython.org/runestone/static/thinkcspy/toc.html#t-o-c
    - http://interactivepython.org/runestone/static/pythonds/index.html
      http://www.pythontutor.com/