Guía docente
340371 - PRO1-I2O23 - Programación I

Última modificación: 04/07/2024

Unidad responsable: Escuela Politécnica Superior de Ingeniería de Vilanova i la Geltrú
Unidad que imparte: 723 - CS - Departamento de Ciencias de la Computación.

Titulación: GRADO EN INGENIERÍA INFORMÁTICA (Plan 2018). (Asignatura obligatoria).

Curso: 2024 **Créditos ECTS:** 7.5 **Idiomas:** Catalán

PROFESORADO

Profesorado responsable: Bernardino Casas Fernández

Otros: Bernardino Casas Fernández
Alejandro Ríos Jerez

CAPACIDADES PREVIAS

Conocimiento de las técnicas de programación imperativa:
- instrucciones básicas: asignación, alternativa, iteración
- acciones, funciones y paso de parámetros
- vectores, tuplas y secuencias
- esquemas de recorrido y búsqueda

Conocer bien un lenguaje imperativo, preferentmente C++. Experiencia en la edición, compilación y ejecución de programas en C++ en un entorno Linux.

REQUISITOS

Haber aprobado FOPR o al menos estar matriculado.

COMPETENCIAS DE LA TITULACIÓN A LAS QUE CONTRIBUYE LA ASIGNATURA

Específicas:
1. CEFB3. Capacidad para comprender y dominar los conceptos básicos de matemática discreta, lógica, algorítmica y complejidad computacional, y su aplicación para el tratamiento automático de la información por medio de sistemas computacionales y su aplicación para la resolución de problemas propios de la ingeniería.
2. CEFB4. Conocimiento de los fundamentos del uso y programación de los computadores, los sistemas operativos, las bases de datos y, en general, los programas informáticos con aplicación en ingeniería.
3. CEFB5. Conocimiento de la estructura, funcionamiento e interconexión de los sistemas informáticos, así como los fundamentos de su programación.
4. CEFC6. 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.
5. CEFC7. Conocimiento, diseño y utilización de forma eficiente los tipos y estructuras de datos más adecuados a la resolución de un problema.

Transversales:
6. APRENDIZAJE AUTÓNOMO - Nivel 1: Llevar a cabo tareas encomendadas en el tiempo previsto, trabajando con las fuentes de información indicadas, de acuerdo con las pautas marcadas por el profesorado.
7. COMUNICACIÓN EFICAZ ORAL Y ESCRITA - Nivel 1: Planificar la comunicación oral, responder de manera adecuada a las cuestiones formuladas y redactar textos de nivel básico con corrección ortográfica y gramatical.
8. TRABAJO EN EQUIPO - Nivel 1: Participar en el trabajo en equipo y colaborar, una vez identificados los objetivos y las responsabilidades colectivas e individuales, y decidir conjuntamente la estrategia que se debe seguir.
9. USO SOLVENTE DE LOS RECURSOS DE INFORMACIÓN - Nivel 1: Identificar las propias necesidades de información y utilizar las colecciones, los espacios y los servicios disponibles para diseñar y ejecutar búsquedas simples adecuadas al ámbito temático.

METODOLOGÍAS DOCENTES

La asignatura consta de:
- 2 horas a la semana de clases presenciales en el aula (grupo grande) en las que se presentan, de forma participativa, los conceptos y procedimientos asociados a los contenidos de la asignatura, y
- 3 horas a la semana en el aula de laboratorio (grupo pequeño) en las que se aplican los conceptos y técnicas aprendidos.

La resolución de problemes se lleva a cabo tanto dentro como fuera del aula (trabajo personal).

OBJETIVOS DE APRENDIZAJE DE LA ASIGNATURA

El objetivo de la asignatura es consolidar las técnicas básicas de diseño de algoritmos para la resolución de problemas mediante el ordenador, en los ámbitos científico y técnico, y aprender los fundamentos de técnicas avanzadas como la programación recursiva, el diseño modular y la programación basada en objetos.

Al finalizar el curso, el estudiante debe:
- Dominar los conceptos de clase, objeto, atributo y método.
- Conocer los elementos de la especificación de una clase de datos y los elementos de su implementación.
- Ser capaz de construir programas que usen clases correspondientes a estructuras lineales simples (pila, cola, lista, vector) y arborescentes (árbol binario, árbol general).
- Conocer las fases del diseño modular.
- Comprender la recursividad lineal y la múltiple, y su relación con la iteratividad.
- Ser capaz de diseñar programas recursivos e iterativos que sean correctos y eficientes.
- Saber explicar los diferentes tipos de inmersiones de especificación y sus características.
- Implementar una estructura de datos con requisitos específicos sobre sus operaciones y la eficiencia de las mismas usando tipos recursivos de datos (o punteros).

HORAS TOTALES DE DEDICACIÓN DEL ESTUDIANTADO

Tipo	Horas	Porcentaje
Horas aprendizaje autónomo	112,5	60.00
Horas grupo grande	30,0	16.00
Horas grupo pequeño	45,0	24.00

Dedicación total: 187.5 h

CONTENIDOS

1.- Diseño modular y diseño basado en objetos

Descripción:
Abstracción y su necesidad. Descomposición funcional y por datos. Módulos. Ocultación de la información. Encapsulado. Fases del diseño modular: distinción entre especificación e implementación. Tipo de módulos y su uso. Bibliotecas. Principios básicos del diseño basado en objetos: clases y objetos; campos y métodos. Implementación de diseños modulares en C++. Compilación separada y montaje. Depuración, prueba y documentación de programas modulares.

Objetivos específicos:
- Diferenciar los roles de usuario, especificador e implementador de clases de datos. Enumerar los elementos de la especificación de una clase de datos. Enumerar los elementos de la implementación de una clase de datos.
- Diseñar una clase de datos con una clara independencia entre su especificación y su implementación. Justificar que la única forma de crear, consultar o modificar un objeto de una clase de datos sea a través de las operaciones de la especificación de la misma.
- Resolver en C++ cualquier ejercicio basado en la aplicación de un esquema algorítmico sencillo sobre un vector formado por elementos de una clase de datos.
- Dada una implementación para una clase de datos sencilla, introducir mejoras en su representación y en sus operaciones.
- Diseñar de forma individual un programa modular en C++ a partir de las abstracciones de datos extraídas del enunciado de un problema. Modificar o añadir alguna funcionalidad a un programa modular dado escrito en C++.
- Preparar un programa en C++ que utilice tipos simples y clases C++ (algunas de ellas predefinidas y otras definidas por el propio alumno) para poder ser ejecutado.

Actividades vinculadas:
Actividad 1 y Control Parcial.

Dedicación: 20h
Grupo grande/Teoría: 2h
Grupo mediano/Prácticas: 3h
Grupo pequeño/Laboratorio: 2h
Actividades dirigidas: 1h
Aprendizaje autónomo: 12h

2.- Estructuras de datos lineales y arborescentes

Descripción:
Pilas, colas y listas: especificación y uso (operaciones de búsqueda y recorrido). Iteradores: definición y uso.
Árboles binarios, n-arios y generales: especificación y uso (operaciones de búsqueda y recorrido).

Objetivos específicos:
- Conocer los tipos de datos utilizados habitualmente para representar y manejar estructuras de datos lineales y su especificación. Diseñar algoritmos iterativos y recursivos para resolver problemas de búsqueda y recorrido en pilas, colas y listas, usando las operaciones del tipo de datos correspondiente e iteradores (cuando sea recomendable).
- Conocer tipos de datos utilizados para representar y manejar estructuras de datos arborescentes y su especificación. Diseñar algoritmos recursivos para resolver problemas de búsqueda y recorrido en árboles binarios usando las operaciones del tipo de datos correspondiente.

Actividades vinculadas:
Actividad 2, Control Parcial y Control Final.

Dedicación: 26h
Grupo grande/Teoría: 3h
Grupo mediano/Prácticas: 4h
Grupo pequeño/Laboratorio: 2h
Actividades dirigidas: 1h
Aprendizaje autónomo: 16h

3.- Programación recursiva metódica

Descripción:
Justificación de la corrección de algoritmos recursivos. Inmersión (o generalización) de una función. Inmersiones de especificación: por debilitamiento de la postcondición y por reforzamiento de la precondición. Relación entre algoritmos recursivos lineales finales y algoritmos iterativos.

Objetivos específicos:
- Describir los principales pasos del diseño de algoritmos recursivos. Justificar la corrección de algoritmos recursivos relativamente sencillos.
- Conocer el concepto de inmersión de una función y saber explicar la diferencia entre inmersiones de especificación e inmersiones de eficiencia. Conocer los diferentes tipos de inmersiones de especificación y los distintos tipos de inmersiones de eficiencia.
- Dado un algoritmo recursivo, determinar si existe una manera sencilla de obtener un algoritmo iterativo equivalente, y si es así, escribirlo.

Actividades vinculadas:
Control Parcial, Control Final y Práctica.

Dedicación: 39h 30m
Grupo grande/Teoría: 4h
Grupo mediano/Prácticas: 6h
Grupo pequeño/Laboratorio: 4h
Actividades dirigidas: 1h
Aprendizaje autónomo: 24h 30m

4.- Programación iterativa metódica

Descripción:
Invariante de un bucle. Justificación de la corrección de algoritmos iterativos.

Objetivos específicos:
- Describir los principales pasos del diseño de algoritmos iterativos. Justificar la corrección de algoritmos iterativos relativamente sencillos.

Actividades vinculadas:
Control Parcial.

Dedicación: 24h 30m
Grupo grande/Teoría: 2h
Grupo mediano/Prácticas: 3h 30m
Grupo pequeño/Laboratorio: 3h
Actividades dirigidas: 1h
Aprendizaje autónomo: 15h

5.- Mejoras de eficiencia en programas recursivos e iterativos

Descripción:
Detección de la repetición de cálculos en programas recursivos e iterativos. Inmersiones de eficiencia: nuevos datos (parámetros de entrada) y/o resultados (valores de retorno o parámetros de salida) en operaciones recursivas para mejorar la eficiencia. Nuevas variables locales que utilizan sus valores en iteraciones anteriores para mejorar la eficiencia de operaciones iterativas.

Objetivos específicos:
- Determinar si el coste de un algoritmo iterativo o recursivo sencillo dado que trabaje sobre vectores, pilas, colas, listas o árboles es lineal o si es cuadrático (suponiendo que el coste sea de uno de estos dos tipos).
- Determinar si se puede mejorar la eficiencia de un algoritmo recursivo sencillo dado y, en caso de que sea posible, diseñar un algoritmo recursivo alternativo más eficiente usando inmersiones de eficiencia.
- Determinar si se puede mejorar la eficiencia de un algoritmo iterativo sencillo dado y, en caso de que sea posible, diseñar un algoritmo iterativo alternativo más eficiente.

Actividades vinculadas:
Control Final y Práctica.

Dedicación: 20h
Grupo grande/Teoría: 2h
Grupo mediano/Prácticas: 3h
Grupo pequeño/Laboratorio: 2h
Actividades dirigidas: 1h
Aprendizaje autónomo: 12h

6.- Tipo recursivos de datos

Descripción:
Introducción al uso de tipo recursivos de datos. El constructor de tipos puntero y la gestión de memoria dinámica. Implementación de estructuras de datos enlazadas mediante tipos recursivos de datos. Algoritmos iterativos y recursivos para resolver problemas de búsqueda y recorrido en estructuras de datos enlazadas accediendo directamente a la representación basada en nodos y punteros a nodos.

Objetivos específicos:
- Implementar una estructura de datos con requisitos específicos sobre sus operaciones y/o la eficiencia de las mismas usando tipos recursivos de datos (o estructuras de datos enlazadas).
- Diseñar algoritmos iterativos y recursivos para resolver problemas de búsqueda y recorrido en estructuras de datos enlazadas utilizando directamente su representación.

Actividades vinculadas:
Control Final.

Dedicación: 20h
Grupo grande/Teoría: 2h
Grupo mediano/Prácticas: 3h
Grupo pequeño/Laboratorio: 2h
Actividades dirigidas: 1h
Aprendizaje autónomo: 12h

SISTEMA DE CALIFICACIÓN

C1 = Control Parcial. Prueba individual (2 horas).
C2 = Control Final. Prueba individual (máx. 3 horas) de caràcter global.
A1 = Actividad individual (propuesta de actividad o resolución de problemas de la plataforma Jutge).
A2 = Actividad individual (resolución de problemas de verificación).
Práctica = Nota resultante correspondiente a las entregas de la práctica.

Calificación Final = (A1*0,5 + A2*0,5)*0,15 + máx((C1*0,5 + C2*0,5), C2) * 0,60 + Práctica*0,25

Módulo de prácticas de laboratorio:
Se compone del trabajo práctico que el alumno realiza para la asignatura. En concreto, la realización de una práctica de programación en equipos de dos personas donde se utilizarán los conocimientos que se aprenden en la asignatura. La práctica se evaluará a partir de: la ejecución del programa realizado, el código presentado y una Prueba de Validación Individual (PVI). La Prueba de Validación Individual puede realizarse mediante una entrevista, el seguimiento en las sesiones de laboratorio o bien en el Control Final de la asignatura. La realización y presentación de la práctica (que recoge el trabajo de prácticas realizado durante el curso) será condición necesaria para la superación de la asignatura. En caso contrario, la calificación final de toda la asignatura será de 'No Presentado'.

La reevaluación, prueba de máximo 3 horas, sustituye la nota de los dos controles, por tanto corresponde a un 60% de la calificación final.

NORMAS PARA LA REALIZACIÓN DE LAS PRUEBAS.

Las pruebas Control Parcial, Control Final y Reevaluación, son presenciales e individuales.
Las pruebas Actividad 1 y Actividad 2 son no presenciales e individuales.
La Práctica se realiza en equipos de dos personas. Se entrega de forma no presencial y se evalúa tanto de forma presencial como de forma no presencial a partir de la documentación presentada.

BIBLIOGRAFÍA

Básica:
- Musser, David R.; Derge, Gillmer J.; Saini, Atul. STL tutorial and reference guide : C++ programming with the standard template library. 2nd ed. Boston [etc.]: Addison-Wesley publishing company, 2000. ISBN 9780321702128.
- Weiss, Mark Allen. Data structures and problem solving using C++. 2a ed. Upper Saddle River: Pearson Education International, 2003. ISBN 0321205006.

Complementaria:
- Balcázar, José Luis. Programación metódica. Madrid [etc.]: McGraw-Hill, 1993. ISBN 8448119576.
- Castro, Jorge. Curs de programació. Madrid [etc.]: McGraw-Hill, 1992. ISBN 844810031X.

RECURSOS

Enlace web:
- http://c.conclase.net/curso/index . Curso de C++. Útil como manual de referencia.
- http://www.cplusplus.com/reference/stl/. Manual de referencia sobre los containers de la librería STL de C++.