Grado en Ingeniería Informática
30213 - Estructuras de datos y algoritmos

Guía docente para el curso 2013 - 2014

Curso: 2, Semestre: 1, Créditos: 6.0

Información básica

Profesores

- **Francisco Javier Campos Laclaustra** jcampos@unizar.es
- **María Piedad Garrido Picazo** piedad@unizar.es
- **Fernando Naranjo Palomino** fnaranjo@unizar.es
- **María Yolanda Villate Pérez** yvillate@unizar.es
- **Jorge Raul Bernad Lusilla** jbernad@unizar.es

Recomendaciones para cursar esta asignatura

El alumno que curse esta asignatura ha de contar con una formación en programación del nivel de la asignatura de Programación II. Por otra parte una adecuada formación matemática del nivel de la asignatura Matemática Discreta resulta muy conveniente.

Actividades y fechas clave de la asignatura

El calendario de exámenes y las fechas de entrega de trabajos se anunciará con suficiente antelación.

Inicio

Resultados de aprendizaje que definen la asignatura

El estudiante, para superar esta asignatura, deberá demostrar los siguientes resultados...

1:
Es capaz de identificar, diseñar y definir Tipos Abstractos de Datos (TADs) independientemente de su implementación.

2:
Diseña e implementa TADs reutilizables y robustos en un lenguaje de programación modular o en un lenguaje orientado a objetos.

3: Diseña e implementa programas robustos de tamaño medio identificando, definiendo e implementando los Tipos Abstractos de Datos (TADs) necesarios.

4:
Es capaz de identificar, utilizar e implementar algunos TADs fundamentales, como: pilas, colas, listas, árboles de búsqueda, tablas hash y grafos.

5:
Es capaz de comparar distintas alternativas de implementación de TADs con respecto al tiempo de ejecución de algoritmos y al uso de la memoria, y de seleccionar la más adecuada en cada problema o contexto.

6:
Conoce y aplica los esquemas algorítmicos básicos (como dividir para vencer, búsqueda con retroceso, voracidad...) a la resolución de problemas.

Introducción

Breve presentación de la asignatura

A la hora de enfrentarse a la resolución de problemas de programación de tamaño medio surge la necesidad de dividir el diseño e implementación de la solución en una serie de módulos o partes, de tamaño y complejidad menor que el problema inicial, pero de forma que dichos módulos interactúen entre si de forma mínima y clara, y constituyan en su conjunto la solución del problema a resolver. Esta aproximación nos permitirá identificar y diseñar módulos con una misión muy clara y acotada, lo que reducirá su grado de dependencia con el resto, y nos permitirá desarrollar módulos reutilizables, eficientes, y robustos, facilitando también que el trabajo de implementación y depuración pueda ser repartido entre varios programadores que trabajen en paralelo.

Gran parte de estos módulos se centrarán en la representación y gestión de la información necesaria para resolver el problema. De esta forma, cada uno de estos módulos encapsulará las estructuras de datos necesarias para representar y almacenar cierta información, ocultando los detalles de su implementación y ofreciendo al exterior únicamente operaciones permitidas (su interfaz), robustas y eficientes, lo que facilitará su reutilización.

En este contexto, se puede definir un Tipo Abstracto de Datos (TAD) como el conjunto de valores que pueden tomar los datos de ese tipo y el conjunto de operaciones para manipularlos, definidos de forma independiente de cualquier representación o implementación. La definición de TADs es por tanto una herramienta de abstracción que intuitivamente responde a la necesidad de enriquecer nuestros lenguajes de programación permitiéndonos proponer nuevos tipos de datos que puedan utilizarse de forma similar a los que suelen ofrecerse predefinidos en un lenguaje de programación, ocultando al usuario de dichos tipos los detalles de su implementación, pero serán de naturaleza mucho más compleja que los tipos habitualmente predefinidos en un lenguaje de programación.

Contexto y competencias

Sentido, contexto, relevancia y objetivos generales de la asignatura

La asignatura y sus resultados previstos responden a los siguientes planteamientos y objetivos:

En esta asignatura el alumno mejorará su capacidad para diseñar y desarrollar programas de ordenador haciendo énfasis en la identificación, diseño y definición de Tipos Abstractos de Datos (TADs) independientemente de su implementación. El alumno aprenderá a diseñar e implementar TADs para que sean reutilizables, eficientes y robustos, y a implementarlos garantizando dichas propiedades. Se presentarán algunos de los TADs fundamentales de uso más frecuente, como: pilas, colas, listas, árboles de búsqueda, tablas, etc., para los que se estudiarán y compararán distintas alternativas de implementación. También se introducirán una serie de esquemas algorítmicos básicos (como dividir para vencer, búsqueda con retroceso, voracidad...) y el alumno aprenderá a reconocer los problemas que requieren este tipo de esquemas para su resolución y cómo aplicarlos.

Contexto y sentido de la asignatura en la titulación

Estructuras de Datos y Algoritmos (EDA) es una asignatura obligatoria englobada en la materia de formación común en Programación y Computación.

Esta asignatura completa la formación recibida por el alumno en las asignaturas de Programación I y Programación II, y le prepara para abordar proyectos de programación de cada vez mayor tamaño y complejidad, y a hacerlo aplicando mejores técnicas y estrategias en el diseño e implementación, permitiendo además el reparto efectivo de la carga del trabajo de implementación y desarrollo de las diferentes partes del sistema a desarrollar. Esta línea de formación continuará ampliándose en las asignaturas de Tecnología de Programación, Programación de Sistemas Concurrentes y Distribuidos, e Ingeniería de Software, así como en otras asignaturas posteriores en el plan de estudios

Además, algunos de los TADs y algoritmos estudiados en EDA serán necesarios para diversas asignaturas que tienen a EDA como prerrequisito directo o indirecto, tales como Tecnología de Programación, Bases de Datos, e Inteligencia Artificial.

Al superar la asignatura, el estudiante será más competente para...

1:
Reconocer y aplicar los procedimientos algorítmicos básicos de las tecnologías informáticas para diseñar soluciones a problemas, analizando la idoneidad y complejidad de los algoritmos propuestos.

2:
Resolver problemas reconociendo o diseñando, y utilizando de forma eficiente, los tipos y estructuras de datos más adecuados a la resolución de un problema.

3:
Resolver problemas y tomar decisiones con iniciativa, creatividad y razonamiento crítico.

4:
Aprender de forma continuada y desarrollar estrategias de aprendizaje autónomo.

Importancia de los resultados de aprendizaje que se obtienen en la asignatura:

Estructuras de Datos y Algoritmos constituirá una base sólida en la formación del alumno para el diseño y desarrollo de sistemas o proyectos de programación de cada vez mayor tamaño y complejidad, ya sean basados en el diseño modular o en el diseño orientado a objetos, buscando siempre la encapsulación, calidad, eficiencia y reutilización del software.

Se presentarán además un conjunto de TADs y algoritmos de uso frecuente, y que todo futuro Ingeniero Informático debe conocer y saber utilizar para poder diseñar soluciones en los nuevos contextos o problemas a los que se enfrente.

Evaluación

Actividades de evaluación

El estudiante deberá demostrar que ha alcanzado los resultados de aprendizaje previstos mediante las siguientes actividades de evaluacion

1:
A continuación se describen las actividades que contribuirán a la evaluación del estudiante y la prueba de evaluación global que se realizará en cada convocatoria para evaluar a los estudiantes de la asignatura.

1. A lo largo del semestre se desarrollarán clases de prácticas en laboratorio, para las que se formarán equipos integrados como máximo por dos alumnos. Con los trabajos prácticos de programación se realizará un seguimiento del trabajo realizado por los alumnos durante el semestre y del progreso de su aprendizaje. Los trabajos presentados por el alumno se calificarán con una nota cuantitativa de 0 a 10. Para obtener dichas notas se valorará el funcionamiento de los programas según especificaciones, la calidad de su diseño y su presentación, la adecuada aplicación de los métodos de resolución, el tiempo empleado, así como la

capacidad de los integrantes del equipo para explicar y justificar el diseño realizado.

Los alumnos que hayan cumplido con los plazos de entrega fijados para los trabajos prácticos de programación, y hayan demostrado en ellos un nivel de aprovechamiento y calidad de resultados adecuados, obteniendo en la valoración de su trabajo práctico una nota de 5.0 como mínimo, serán exentos de la realización del examen práctico de programación en laboratorio. Para dichos alumnos, la calificación obtenida con sus prácticas se utilizará como nota de examen práctico de programación en laboratorio, salvo que el alumno decida presentarse a la prueba de examen práctico en laboratorio, en cuyo caso prevalecerá la nota obtenida en el examen práctico individual.

2. Examen práctico e individual de programación, en laboratorio. En el examen práctico se le plantearán al alumno ejercicios de programación de naturaleza similar a los realizados en las prácticas o vistos en clase. Se calificará con una nota de 0 a 10, para la que se valorará el correcto funcionamiento y rendimiento de los programas según especificaciones, la calidad de su diseño, la adecuada aplicación de los métodos de resolución y el tiempo empleado. Para aquellos alumnos que resulten exentos de la realización de este examen y opten por presentarse al mismo prevalecerá la nota obtenida en el examen práctico individual. Será necesaria una calificación mínima de 5.0 puntos en el examen práctico para aprobar la asignatura y, en tal caso, la calificación obtenida pondera un 30% de la nota final de la asignatura.

3. Examen escrito en el que se deberán resolver problemas de programación y, en su caso, responder preguntas conceptuales o resolver algún ejercicio. Se calificará con una nota de 0 a 10. En general, se valorará la calidad y claridad de las respuestas y soluciones propuestas, su adecuación a las especificaciones y restricciones planteadas, la calidad del diseño, la adecuada aplicación de los métodos de resolución y el tiempo empleado. Será necesario obtener una calificación mínima de 4.0 puntos en el examen escrito para aprobar la asignatura. En tal caso la calificación obtenida pondera un 70% de la nota final de la asignatura.

Evaluación global

La prueba global de evaluación de la asignatura consta de dos partes:

- Examen práctico de programación en laboratorio e individual. En cada convocatoria se realizará un examen práctico de programación en laboratorio, en el que se le plantearán al alumno ejercicios de programación de naturaleza similar a los realizados en las prácticas o vistos en clase. Es necesario una calificación mínima de 5.0 puntos en el examen práctico para aprobar la asignatura. En tal caso la calificación obtenida pondera un 30% de la nota final de la asignatura.

  Los alumnos que hayan cumplido con los plazos de entrega fijados para los trabajos prácticos de programación, y hayan demostrado en ellos un nivel de aprovechamiento y calidad de resultados adecuados obteniendo una valoración de su trabajo práctico de como mínimo 5.0, serán exentos de la realización del examen práctico de programación en laboratorio convirtiéndose automáticamente la nota numérica obtenida en la evaluación de sus trabajos prácticos en su nota final de examen práctico de programación. No obstante, para los alumnos exentos del examen práctico que se presenten al mismo, en cualquier convocatoria, prevalecerá la nota obtenida en el examen práctico individual de programación.

- Examen escrito en el que se deberán resolver problemas de programación y, en su caso, responder preguntas *conceptuales* o resolver algún ejercicio. Es necesaria una calificación mínima de 4.0 puntos en el examen escrito para aprobar la asignatura. En tal caso la calificación obtenida pondera un 70% de la nota final de la asignatura.

Si la calificación obtenida por el alumno en el examen escrito es igual o superior a 4.0 y su calificación en el examen práctico de programación es igual o superior a 5.0, entonces la calificación del alumno en la asignatura se obtendrá como la suma ponderada de las calificaciones del examen escrito con ponderación del 70%, y del examen práctico con ponderación del 30%. Si la calificación en el examen escrito es inferior a 4.0, la calificación del alumno en la asignatura será la obtenida en el examen escrito, independientemente de la nota obtenida en el examen práctico. Si, por el contrario, no se alcanza la calificación mínima de 5.0 en el examen práctico, entonces la calificación del alumno en la asignatura se obtendrá como el mínimo entre: la nota del examen práctico, y la suma ponderada de ambas partes. Las calificaciones obtenidas en las dos partes en la primera convocatoria se guardan para la siguiente convocatoria del mismo curso académico en el caso de que el alumno no logre aprobar la asignatura.

Actividades y recursos

Presentación metodológica general

El proceso de aprendizaje que se ha diseñado para esta asignatura se basa en lo siguiente:

1. El estudio y trabajo continuado desde el primer día de clase.
2. El aprendizaje de conceptos y metodologías para el diseño e implementación de TADs correctos, reutilizables y eficientes a través de las clases magistrales, en las que se favorecerá la participación de los alumnos.
3. La aplicación de tales conocimientos al diseño y análisis de algoritmos y programas en las clases de problemas. En estas clases los alumnos desempeñarán un papel activo en la discusión y resolución de los problemas.
4. Las clases de prácticas en laboratorio en las que el alumno deberá poner en práctica la tecnología necesaria para desarrollar proyectos de programación de tamaño pequeño o medio, utilizando un lenguaje de programación determinado y aplicando los conceptos y técnicas estudiadas en esta asignatura.
5. El trabajo en equipo desarrollado para resolver las prácticas de la asignatura, equipos integrados como máximo por dos alumnos, y cuyo resultado se plasma en la entrega de programas resultantes convenientemente diseñados y documentados, así como en la explicación y justificación del diseño realizado y decisiones adoptadas, que se expondrán al profesor tutor de prácticas.
6. Un trabajo continuado en el que se conjugue la comprensión de conceptos, el análisis y la resolución de problemas de programación utilizando “lápiz y papel” y la puesta a punto en computador de algunos proyectos de programación de tamaño pequeño o medio.

Actividades de aprendizaje programadas (Se incluye programa)

El programa que se ofrece al estudiante para ayudarle a lograr los resultados previstos comprende las siguientes actividades...

1:
En las clases impartidas en el aula se desarrollará el temario de la asignatura.

2:
En las clases de problemas se resolverán problemas de aplicación de los conceptos y técnicas presentadas en el programa de la asignatura. Se propondrán problemas y ejercicios para ser resueltos antes de la clase de problemas en la que se presentarán y discutirán diferentes soluciones a dichos problemas. También se propondrán ejercicios durante la sesión de problemas para ser resueltos durante la misma, algunos de forma individual y otros para ser trabajados en grupo.

3:
Las sesiones de prácticas de desarrollan en un laboratorio informático. En estas sesiones el alumno deberá trabajar en equipo y realizar una serie de trabajos de programación directamente relacionados con los temas estudiados en la asignatura. Para las prácticas de laboratorio se propondrán una serie de trabajos o ejercicios de programación para que el alumno los resuelva, desarrolle parte del trabajo en el laboratorio, los complete como trabajo en casa, y los entregue dentro de los plazos de tiempo que se fijen en cada caso.

Planificación y calendario

Calendario de sesiones presenciales y presentación de trabajos

La organización docente de la asignatura prevista es la siguiente.

- Clases teóricas (2 horas semanales)
- Clases de problemas (1 hora semanal)
- Prácticas de la asignatura: Habrá una primera sesión presencial de dos horas en laboratorio. El resto de las prácticas consistirán en trabajos tutelados por un profesor, en las que participan los alumnos de cada uno de los subgrupos en los que se divide el grupo.

Presentación de trabajos prácticos de programación:

Los trabajos de programación a desarrollar en las prácticas de la asignatura deberán ser realizados y presentados de acuerdo a lo especificado para cada uno de ellos, y dentro de las fechas límite que se anunciarán en el enunciado de cada uno de los trabajos propuestos o con suficiente antelación.

Programa de la asignatura

Temario

1. Programación con Tipos Abstractos de Datos.
2. Tipos de datos lineales.
3. Tipos de datos arborescentes.
4. Tipos de datos funcionales.
5. Introducción a los grafos.
6. Introducción a los esquemas algorítmicos.

Trabajo del estudiante

La dedicación del estudiante para alcanzar los resultados de aprendizaje en esta asignatura se estima en 157 horas distribuidas del siguiente modo:

- 49 horas, aproximadamente, de actividades presenciales (clases teóricas, de problemas y prácticas en laboratorio)
- 41 horas de trabajo de programación en equipos de 2 personas para desarrollar los programas propuestos en las prácticas de laboratorio
- 61 horas de estudio personal efectivo (estudio de apuntes y textos, resolución de problemas, preparación clases y prácticas, desarrollo de programas)
- 6 horas de examen final de teoría escrito y de prácticas en laboratorio

Bibliografía

Bibliografía de la asignatura

Bibliografía Básica:

- Weiss, M.A.: *Data Structures and Algorithm Analysis in Java, 3rd Edition*, Pearson/Addison Wesley, 2011.
- Hernández, Z.J. y otros: *Fundamentos de Estructuras de Datos. Soluciones en Ada, Java y C++*, Thomson, 2005.
- Weiss, M.A.: *Estructuras de datos en Java*, Pearson/Addison Wesley, 2000.

Ejercicios:

- Martí Ollet, N., Ortega Mallén, Y., Verdejo López, J.A.: Estructura de datos y algoritmos. Ejercicios y problemas resueltos, Pearson Prentice Hall, 2003.
- Joyanes, L., Zahonero, I., Fernández, M. y Sánchez, L.: Estructura de datos. Libro de problemas, McGraw Hill, 1999.

Bibliografía sobre Java:

- Deitel, P.J. y Deitel, H.M.: *Java. Cómo programar (7ª edición)*, Prentice Hall, 2008.

Bibliografía Complementaria:

- Campos Laclaustra, J.: *Estructuras de Datos y Algoritmos*, Prensas Universitarias de Zaragoza, Colección Textos Docentes, 1995.
- Franch Gutiérrez, X.: Estructuras de Datos. Especificación, Diseño e Implementación, 3ª edición, Ed. Edicions UPC, 2001.
- Mehta, D.P. y Sahni, S.: *Handbook of Data Structures and Applications*, Chapman & Hall/CRC, 2005.

Referencias bibliográficas de la bibliografía recomendada

Escuela Universitaria Politécnica

- Campos Laclaustra, Javier. Estructuras de datos y algoritmos / Javier Campos Lacaustra . - 1ª ed., 1ª reimp. Zaragoza : Prensas Universitarias de Zaragoza, 2001
- Deitel, Paul J.. Java : cómo programar / P. J. Deitel, H. M. Deitel ; traducción Alfonso Vidal Romero Elizondo ; revisión técnica Gabriela Azucena Campos García, Roberto Martínez Román, Jorge Armando Aparicio Lemus. - 7ª ed. Naucalpan de Juárez (Estado de México) : Pearson Educación, 2008
- Estructura de datos. Libro de problemas / Luis Joyanes Aguilar [et al.] Madrid [etc.] : McGraw-Hill, D.L.1999
- Franch Gutiérrez, Xavier. Estructuras de datos : Especificación, diseño e implementación / Xavier Franch Gutiérrez . - 2a. ed. Barcelona : UPC, 1996
- Fundamentos de estructura de datos : soluciones en Ada, Java y C++ / Zenón José Hernández Figueroa ... [et al.] Madrid : Thomson, D.L. 2005
- Handbook of data structures and applications / edited by Dinesh P. Mehta and Sartaj Sahni Boca Raton, Florida [etc.] : Chapman & Hall/CRC, cop. 2005
- MARTÍ OLIET, N. Estructura de datos y algoritmos. Ejercicios y problemas resueltos / Nardiso Martí Oliet, Y. Ortega Mallén, J.A. Verdejo López,. Madrid : Prentice Hall, 2003
- Weiss, Mark Allen. Data structures and algorithm analysis in Java / Mark Allen Weiss Reading, Massachusetts [etc.] : Addison-Wesley, cop. 1999
- Weiss, Mark Allen. Estructuras de datos en Java : compatible con Java 2 / Mark Allen Weiss . - 1a ed. en español Madrid : Addison Wesley, cop. 2000

Escuela Politécnica Superior

- [B. Básica] - Fundamentos de estructura de datos : soluciones en Ada, Java y C++ / Zenón José Hernández Figueroa ... [et al.] . Madrid : Thomson, D.L. 2005
- [B. Básica] - Weiss, Mark Allen. Data structures and algorithm analysis in Java / Mark Allen Weiss . 3rd ed.,International ed. / contributions by Arup Kumar Bhattacharjee, Soumen Mukherjee Harlow, England : Pearson Education, cop. 2012
- [B. Básica] - Weiss, Mark Allen. Estructuras de datos en Java : compatible con Java 2 / Mark Allen Weiss . 1a ed. en español Madrid : Addison Wesley, cop. 2000
- [B. Complementaria] - Campos Laclaustra, Javier. Estructuras de datos y algoritmos / Javier Campos Lacaustra . [1a. ed.] Zaragoza : Prensas Universitarias de Zaragoza, 1995
- [B. Complementaria] - Franch Gutiérrez, Xavier. Estructuras de datos : especificación, diseño e implementación / Xavier Franch Gutiérrez . 3a ed. Barcelona : Edicions UPC, 1999
- [B. Complementaria] - Handbook of data structures and applications / edited by Dinesh P. Mehta and Sartaj Sahni. Boca Raton, Florida [etc.] : Chapman & Hall/CRC, cop. 2005
- [Ejercicios] - Estructura de datos. Libro de problemas / Luis Joyanes Aguilar [et al.] . Madrid [etc.] : McGraw-Hill, D.L.1999 [Ejercicios]
- [Ejercicios] - Martí Oliet, Narciso. Estructuras de datos y métodos algorítmicos : ejercicios resueltos / Narciso Martí Oliet, Yolanda Ortega Mallén, José Alberto Verdejo López . Madrid [etc.] : Prentice Hall, D.L. 2003 [Ejercicios]
- [Java] - Deitel, Paul J.. Java : cómo programar / P. J. Deitel, H. M. Deitel ; traducción Alfonso Vidal Romero Elizondo ; revisión técnica Gabriela Azucena Campos García, Roberto Martínez Román, Jorge Armando Aparicio Lemus. 7ª ed. Naucalpan de Juárez (Estado de México) : Pearson Educación, 2008 [Java]