Programa del curso

Semestre 2020-10

Nombre del curso:	ISIS1206 Estructuras de Datos
Course Name:	------
Créditos:	3
Profesor:	Fernando De la Rosa Rosero (fde@uniandes.edu.co) Luis Florez Salamanca (le.florez602@uniandes.edu.co) Carlos Andres Lozano Garzon (calozanog@uniandes.edu.co)
Versión PDF	Click Aquí

Descripción

Estructuras de datos, es el último curso del ciclo básico de programación, el propósito de este curso es generar en el estudiante la habilidad de diseñar e implementar las estructuras de datos en memoria principal necesarias para resolver un problema, teniendo en cuenta los criterios de calidad de eficiencia en tiempo de ejecución y en memoria principal de la solución. [tab title="Secciones"]

Secciones

Sección:	1
Profesor:	Pendiente (...@uniandes.edu.co)
Clases:	Lu & Ju - 6:30 am a 7:50 am
Laboratorio:	Mi 6:30 am a 7:50 am

Sección:	2
Profesor:	Fernando De la Rosa (fde@uniandes.edu.co)
Clases:	Lu & Mi - 9:30 am a 10:50 am
Laboratorio:	Ju 9:30 am a 10:50 am

Sección:	3
Profesor:	Luis Esteban Florez (le.florez602@uniandes.edu.co)

Clases:	Ma & Mi - 12:30 pm a 1:50 pm
Laboratorio:	Vi 12:30 pm a 1:50 pm

Sección:	4
Profesor:	Carlos Andres Lozano Garzon (calozanog@uniandes.edu.co)
Clases:	Lu & Ma - 2:00 pm a 3:20 pm
Laboratorio:	Ju 2:00 pm a 3:20 pm

Sección:	5
Profesor:	Fernando De la Rosa (fde@uniandes.edu.co)
Clases:	Lu & Ma - 3:30 pm a 4:50 pm
Laboratorio:	Ju 3:30 pm a 4:50 pm

Objetivos

Al final del curso, se espera que el estudiante esté en capacidad de:

- Diseñar e implementar una estructura contenedora abstracta, garantizando el desacoplamiento entre su descripción funcional y su representación interna.
- Proponer y justificar el diseño de unas estructuras de datos para resolver un problema, utilizando como argumentos la complejidad de los algoritmos que implementan las operaciones críticas, el espacio ocupado en memoria y la flexibilidad.
- Entender los diferentes tipos de estructuras de datos que se pueden utilizar para modelar los elementos de un mundo.
- Escribir los algoritmos que manipulan las principales estructuras de datos lineales, de acceso directo, recursivas y no lineales.
- Diseñar la solución a problemas integrando estructuras de datos apropiadas para obtener soluciones eficientes en tiempo y en espacio.

Objetivo pedagógico

Objetivo pedagógico	Metas específicas	%
O4	• Escribir el algoritmo que resuelve cada una de las tareas dado un plan de solución de un problema • Descomponer un problema en partes y construir una aplicación como un conjunto de componentes independientes, conectados entre sí por medio de interfaces.	1%

O5	Evaluar y justificar un diseño basado en estructuras de datos para implementar una estructura contenedora abstracta, siguiendo una metodología que tenga en cuenta un conjunto de restricciones impuestas (tiempo, espacio y flexibilidad)	1%
O6	• Evaluar y justificar un diseño basado en estructuras de datos para implementar una estructura contenedora abstracta, siguiendo una metodología que tenga en cuenta un conjunto de restricciones impuestas (tiempo, espacio y flexibilidad) • Evaluar las diferentes complejidades de un algoritmo con el fin de optimizar tareas definidas en un problema dado o cumplir con requerimientos no funcionales preestablecidos • Comprender el impacto que tienen las estructuras de datos que utiliza un algoritmo sobre la complejidad del mismo	1%
O8	• Utilizar el concepto de patrón de algoritmo, como una estructura algorítmica típica para resolver ciertos tipos de problemas bien definidos en estructuras de datos • Escribir el algoritmo que resuelve cada una de las tareas dado un plan de solución de un problema. • Evaluar las diferentes complejidades de un algoritmo con el fin de optimizar tareas definidas en un problema dado o cumplir con requerimientos no funcionales preestablecidos### Descomponer un problema en partes y construir una aplicación como un conjunto de componentes independientes, conectados entre sí por medio de interfaces. • Modelar estructuras de datos abstractas lineales, recursivas y no lineales. • Comprender el impacto que tienen las estructuras de datos que utiliza un algoritmo sobre la complejidad del mismo	25%

O9	• Identificar las estructuras de datos que hacen parte de la arquitectura de un programa. Justificar dicha arquitectura. • Comparar arquitecturas de una aplicación teniendo en cuenta criterios basados en requerimientos no funcionales • Incorporar en la arquitectura de un programa elementos externos, de los cuales sólo existe una especificación funcional (componentes pre-hechos con una especificación clara) • Modelar, diseñar, construir y utilizar estructuras de datos genéricas, en las cuales sea posible parametrizar el tipo de los elementos contenidos. • Utilizar un componente como parte de la implementación de otro componente • Utilizar la técnica de descomposición de un modelo en términos de los elementos que lo componen	7%
O10	Diseñar y construir pruebas automáticas en una aplicación teniendo en cuenta la definición aleatoria de casos de prueba	3%

O13	• Expresar la arquitectura de un programa en un lenguaje gráfico semi-formal, en el que se hagan explícitos todos los elementos que la componen. • Incorporar en la arquitectura de un programa elementos externos, de los cuales sólo existe una especificación funcional (componentes pre-hechos con una especificación clara) • Evaluar e incorporar patrones básicos de desacoplamiento entre los elementos de una arquitectura • Modelar, diseñar, construir y utilizar estructuras de datos genéricas, en las cuales sea posible parametrizar el tipo de los elementos contenidos. • Crear una aplicación, utilizando una arquitectura de múltiples capas según su contexto • Diseñar y construir pruebas automáticas en una aplicación teniendo en cuenta la definición aleatoria de casos de prueba • Implementar el modelaje y la solución de un problema en un subconjunto del lenguaje de programación java • Entender las ventajas de un lenguaje canónico y estándar como XML para el intercambio y estandarización de información entre aplicaciones y utilizarlo en una aplicación • Utilizar frameworks de alto nivel y/o librerías empaquetadas para facilitar y/o integrar el desarrollo de nuevas funcionalidades a una aplicación • Entender y utilizar herramientas que faciliten y/o automaticen el proceso de despliegue de una aplicación • Utilizar con alguna facilidad un ambiente de desarrollo de software • Utilizar la técnica de descomposición de un modelo en términos de los elementos que lo componen • Utilizar la técnica de patrones de algoritmos para construir algoritmos utilizando el esqueleto de un patrón como base para luego refinar dicho esqueleto • Utilizar patrones de desacoplamiento para estructurar el diseño de una aplicación	56%
O14	• Expresar el modelo del mundo del problema en un subconjunto del lenguaje UML • Documentar un programa utilizando tecnología apropiada (i.e. javadoc)	3%

O15	Seguir un proceso (en grupos) en general, leer y entender un formato. Crear entregables.	2%

Objetivos pedagógicos transversales

Objetivo Pedagógico	Metas específicas
OT1	• Utilizar frameworks de alto nivel y/o librerías empaquetadas para facilitar y/o integrar el desarrollo de nuevas funcionalidades a una aplicación • Entender y utilizar herramientas que faciliten y/o automaticen el proceso de deploy de una aplicación • Utilizar con alguna facilidad un ambiente de desarrollo de software
OT2	• Evaluar las diferentes complejidades de un algoritmo con el fin de optimizar tareas definidas en un problema dado o cumplir con requerimientos no funcionales preestablecidos • Comprender el impacto que tienen las estructuras de datos que utiliza un algoritmo sobre la complejidad del mismo • Evaluar y justificar un diseño basado en estructuras de datos para implementar una estructura contenedora abstracta, siguiendo una metodología que tenga en cuenta un conjunto de restricciones impuestas (tiempo, espacio y flexibilidad)
OT8	• Utilizar frameworks de alto nivel y/o librerías empaquetadas para facilitar y/o integrar el desarrollo de nuevas funcionalidades a una aplicación • Entender y utilizar herramientas que faciliten y/o automaticen el proceso de deploy de una aplicación • Utilizar con alguna facilidad un ambiente de desarrollo de software

Metodología

El curso consiste en 3 horas semanales de clase presencial con el profesor, 1½ horas de trabajo supervisado en el laboratorio y 4½ horas de trabajo individual por fuera de clase. El estudiante que no asista al menos al 80% de las clases y sesiones de trabajo supervisado no podrá aprobar el curso. El curso esta organizado en 3 partes que corresponden a objetivos pedagógicos específicos y a un conjunto de conocimientos y habilidades que se introducen o refuerzan. Los medios de comunicación oficiales del curso son el sitio WEB del curso, la lista de correo electrónico y las aulas virtuales en SICUA+. [tab title="Evaluaciones"]

Evaluaciones

Durante el semestre se realizarán 3 exámenes escritos individuales. Además, el estudiante deberá desarrollar 3 proyectos, los cuales se desarrollarán en parejas, aunque la sustentación y la nota serán individuales. Adicionalmente los estudiantes desarrollaran individualmente talleres de refuerzos de los temas vistos. Sólo se aceptan las entregas de los proyectos y los talleres hechos según el medio electrónico definido para el curso y que es informado en los enunciados respectivos en los plazos establecidos. Sobre este último punto no hay excepciones. Los porcentajes de calificación son los siguientes:

Concepto	Porcentaje
Examen 1	17%
Examen 2	17%
Examen 3	17%
Proyecto 1	10%
Proyecto 2	12%
Proyecto 3	15%
Talleres	12%

Para aprobar el curso se debe obtener una nota numérica superior o igual a 3.0, lo cual implica que no hay ningún tipo de aproximación.

Notas finales

La nota final se calculará como la Nota Ponderada de las evaluaciones realizadas por su porcentaje respectivo, expresada con sus dos primeros decimales. **Caso Especial**: NO hay aproximación a 3.00 con una nota ponderada inferior a 3.00. En particular, las notas ponderadas con 2.99 No se aproximan a 3.00.

Calendario

Calendario Global del Curso

Semana	Clase	Tema	Laboratorio/Proyectos
1	1	Introducción	T0: Ambiente de desarrollo del curso, Manejo de repositorios GIT, Manejo de consola, Envío de talleres (estructura),
	2	Java Generics	
2	3	Java Generics	T1: Listas, Enunciado General del proyecto **Enunciado proyecto 1**
	4	Listas	

Semana	Clase	Tema	Laboratorio/Proyectos
3	5	Pilas y Colas	T2: Pilas y Colas
	6	Algoritmos de ordenamiento básicos	
4	7	Shellsort	Trabajo Proyecto 1
	8	MergeSort	
5	9	Quicksort	T3: Ordenamiento
	10	Colas de prioridad: Binary Heap, HeapSort	
6		**Parcial 1 (Semana 1-5)**	
7	13	Colas de prioridad: Binary Heap, HeapSort	Entrega proyecto 1 **Enunciado proyecto 2** T4: Colas de prioridad: Binary Heap, HeapSort
	14	Búsquedas: tabla de símbolos, Tabla de Hash	
8	15	Tabla de Hash	T5: Tablas de Hash
	16	Árboles binarios	
9	17	Árboles balanceados	T6: Árboles Balanceados Trabajo Proyecto 2
	18	Árboles balanceados	
10		**Parcial 2 (Semana 5-9)**	
11	21	Grafos no dirigidos	**Entrega proyecto 2 Enunciado proyecto 3**
	22	Grafos no dirigidos	
12	23	Algoritmos sobre Grafos no dirigidos	T7: Grafos No dirigidos
	24	Algoritmos sobre Grafos no dirigidos	
13	25	Grafos Dirigidos	Grafos Dirigidos
	26	Grafos Dirigidos	
14	27	Algoritmos sobre grafos	Trabajo Proyecto 3 Grafos Dirigidos
	28	Algoritmos sobre grafos	
15	29	Algoritmos sobre grafos	
	30	Algoritmos sobre grafos	

Semana	Clase	Tema	Laboratorio/Proyectos
16		**Parcial 3 (Semana 11-15)**	**Preparacion parcial 3 Entrega proyecto 3**

[tab title="Guías"]

Guías

- Envío talleres
- Envío proyectos
- Ambiente de desarrollo
- Manejo de repositorios (GIT)
- Snippets de código
- FAQ Proyectos

[tab title="Recursos"]

Recursos

- Eclipse (IDE)
- SourceTree (Cliente Git)
- Bitbucket (Repositorio Git)